VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE LESNÍ ZVĚŘE S VYUŽITÍM PLATFORMY ESP32

WILDLIFE DETECTION USING THE ESP32 PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ŠTĚPÁN VONDRÁČEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ GOLDMANN, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

162914

Ústav: Ústav inteligentních systémů (UITS)
Student: **Vondráček Štěpán**
Program: Informační technologie
Název: **Detekce lesní zvěře s využitím platformy ESP32**
Kategorie: Umělá inteligence
Akademický rok: 2024/25

Zadání:

1. Seznamte se s dostupnými metodami pro detekci lesní zvěře, například srny, divoká prasata. Dále sumarizujte informace o moderních fotopastech.
2. Zjistěte, jaké metody je možné použít na nízkoenergetických mikrokontrolérech.
3. Navrhněte nízkoenergetický kamerový modul s ESP32, který bude provádět detekci vybrané lesní zvěře a metadata posílat prostřednictvím LoRaWAN sítě The Things Network.
4. Kamerový modul sestavte a implementujte obslužný firmware. Dále implementujte jednoduchou uživatelskou aplikaci, která bude stahovat data ze serveru sítě The Things Network.
5. Funkčnost systému otestujte v reálném prostředí. Vyhodnoťte energetickou spotřebu celého systému.

Literatura:

- DE CARVALHO SILVA, Jonathan, et al. LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities. In: *2017 2nd International multidisciplinary conference on computer and energy science (SpliTech)*. IEEE, 2017. p. 1-6.
- MA, Ding; YANG, Jun. Yolo-animal: An efficient wildlife detection network based on improved yolov5. In: *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*. IEEE, 2022. p. 464-468.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Goldmann Tomáš, Ing., Ph.D.**
Vedoucí ústavu: Kočí Radek, Ing., Ph.D.
Datum zadání: 1.11.2024
Termín pro odevzdání: 14.5.2025
Datum schválení: 31.10.2024

Abstrakt

Tato práce se zabývá vývojem nízkoenergetického zařízení založeného na platformě ESP32, schopné detekovat lesní zvěř pomocí neuronových sítí běžících přímo na mikrokontroléru. Zařízení využívá síť LoRaWAN k přenosu informací o detekovaných zvířatech na server The Things Network. Díky modelu detekce FOMO se snižuje počet zbytečných fotografií bez výskytu zvěře a zvyšuje se efektivita ukládání dat. Součástí práce je návrh a implementace jak hardwarového řešení, tak obslužného programu a jednoduché desktopové aplikace pro vizualizaci přijatých dat. Výsledný systém byl otestován v reálném prostředí.

Abstract

This thesis deals with the development of a low-power device based on the ESP32 platform, capable of detecting wildlife using neural networks running directly on a microcontroller. The device uses the LoRaWAN network to transmit information about the detected animals to The Things Network server. Thanks to the FOMO detection model, the number of unnecessary photos without wildlife is reduced and the efficiency of data storage is increased. The work includes the design and implementation of both a hardware solution and firmware, and a simple desktop application for visualizing the received data. The resulting system was tested in a real environment.

Klíčová slova

ESP32, Fotopast, TinyML, TensorFlow Lite for Microcontrollers, Edge Impulse, YOLO, FOMO, Faster R-CNN, LoRaWAN, The Things Network, ElectronJS, MQTT

Keywords

ESP32, Camera trap, TinyML, TensorFlow Lite for Microcontrollers, Edge Impulse, YOLO, FOMO, Faster R-CNN, LoRaWAN, The Things Network, ElectronJS, MQTT

Citace

VONDRÁČEK, Štěpán. *Detekce lesní zvěře s využitím platformy ESP32*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann, Ph.D.

Detekce lesní zvěře s využitím platformy ESP32

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Goldmanna, Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Štěpán Vondráček
12. května 2025

Obsah

Kapitola 1

Úvod

Detekce lesní zvěře je důležitá z mnoha důvodů, ať už z pohledu myslivosti, ochrany přírody či prevence dopravních nehod. Myslivci mohou efektivněji monitorovat populaci zvěře, což je prospěšné pro určení přemnožení zvěře v lese. Při přemnožení některého z druhů lesní zvěře hrozí větší nebezpečí kolize s vozidly, jelikož se více zvěře bude pohybovat po silnici. Myslivcům může detekce pomoci lépe naplánovat, kdy je vhodné zvěř lovit. Moderní technologie mohou hrát zásadní roli při řešení těchto výzev.

Existují různé metody pro detekci a monitorování lesní zvěře, jako jsou fotopasti, pozorování přímo myslivci, případně sledování pomocí GPS obojků. Fotopasti poskytují vysokou kvalitu obrazu, při detekci pohybu vyfotí a uloží fotografii. Některé moderní fotopasti již dovolují bezdrátové ovládání pomocí mobilní sítě GSM. Nevýhoda je, že nemají žádný způsob, jak si ověřit, že na fotografii se nachází zvěř, a proto je pořízeno spousta fotografií, kde se nic nenachází, a fotografie se musí manuálně promazávat. Pozorování myslivci je časově náročné. GPS obojky poskytují přesné informace o pohybu zvěře, ale jejich použití je omezeno na menší skupiny sledovaných jedinců, jelikož nejprve musíme zvěři nasadit obojek. Nasazování obojků všem zvířatům by bylo také časově náročné a nezjistíme polohu těch, kteří obojek nemají nasazen.

Cílem této práce je vyvinout zařízení pro detekci lesní zvěře založené na platformě ESP32. Toto zařízení bude využívat technologii LoRaWAN k bezdrátové komunikaci přes server The Things Network, což umožní vzdálené monitorování a upozornění na detekci zvěře v reálném čase. Zařízení bude fungovat na principu fotopasti, ale bude používat objektovou detekci za použití umělých neuronových sítí. To dovolí, aby při detekci byl rovnou znám druh zvěře. Sníží se tak počet falešných detekcí a zlehčí se kontrola pořízených fotografií. Dále je cílem implementovat jednoduchou aplikaci pro zobrazení výsledků přijatých na server The Things Network a otestovat vyrobený prototyp zařízení v reálném prostředí lesa.

Práce je rozdělena do následujících kapitol. V kapitole 2 jsou popsány metody detekce lesní zvěře. Dále se kapitola zaměří na informace o moderních fotopastech. Kapitola 3 je zaměřena na základní informace o fungování neuronových sítí a detektorech objektů využívajících neuronové sítě. Dále jsou zde popsány metody, které umožňují spuštění modelů detekce objektů na nízkoenergetických mikrokontrolérech. V kapitole 4 jsou informace o platformě ESP32 a síti LoRaWAN. Byl zde vytvořen návrh obslužného programu pro zařízení s čipy ESP32, návrh jednoduché aplikace zobrazující data z detekce zařízením a samotný návrh realizace zařízení včetně zapojení. Implementace obslužného programu, aplikace pro zobrazování dat a realizace zařízení jsou v kapitole 5. Kapitola 6 je o testování zařízení a analýze energetické spotřeby.

Kapitola 2

Detekce lesní zvěře a fotopasti

Monitorování a detekce volně žijící lesní zvěře představuje zásadní nástroj pro řízení její populace, snižování škod způsobovaných na zemědělských a lesních porostech a udržení rovnováhy mezi lidskou činností a přírodními procesy. Populace zvěře jsou vázány na dostupnost potravy, přičemž v kulturní krajině často dochází ke konzumaci hospodářsky významných rostlin. To přirozeně vede k ekonomickým ztrátám, kterým se i přes preventivní opatření zcela zabránit nelze. Změny v krajině a způsobu hospodaření v posledních letech vedly ke zvýšenému výskytu zvěře na zemědělské půdě. Mezi hlavní faktory tohoto přesunu patří například rušení v lesních komplexech, vyšší atraktivita kulturních plodin oproti přirozené potravě, možnost celoročního úkrytu v porostech a vyšší kvalita potravy na polích, zejména během zimních měsíců . Každoroční odhad početnosti zvěře je nezbytným podkladem pro tvorbu mysliveckého plánu. Evidence škod a ulovených jedinců slouží k průběžné aktualizaci minimálního plánu lovu. Efektivní metodika detekce a prevence škod umožňuje nejen minimalizovat hospodářské ztráty, ale rovněž podporuje udržitelné hospodaření s přírodními zdroji [18].

2.1 Metody detekce lesní zvěře

Mezi používané metody detekce a monitorování zvěře dle patří [18]:

- **Sčítání zvěře při pochůzkách**, často na vybraných liniích či plochách.
- **Pozorování zvěře z posedů**, případně z jiných pozorovacích míst.
- **Vyhánění zvěře**, z předem vymezených a obestavených oblastí za účelem sčítání.
- **Statické sčítání**, z pevných pozic během definovaného časového úseku.
- **Letecké sčítání**, za použití lehkých letadel, vrtulníků nebo bezpilotních prostředků (dronů), často doplněné o kamerové nebo termovizní zařízení.
- **Noční sčítání se světlomety**, zejména v otevřeném terénu.
- **Termovizní přímé sčítání**, z pozorovacích míst s termovizním zařízením.
- **Detekce pomocí termovize**, sloužící k lokalizaci mláďat (zajíčků, srnčat) v porostech (například systém VMT-VÚZT[1] nebo drony s termokamerami).

[1] Více informací na stránce https://w-technika.cz/products/vmt-vuzt-termovizni-vyhledavac-srncat-behem-senosece

- **Metoda sčítání trusu**, zahrnující kvantifikaci hromádek trusu na určených plochách. Může být provedena jednorázově, nebo formou opakovaného měření s čištěním ploch.
- **Fotopasti**, jsou digitální zařízení aktivovaná pohybem, která po aktivaci pořídí fotografii. Umožňují sledování pohybu zvěře, vyhodnocení jejího výskytu a chování, a často se využívají také ke stanovení vhodného umístění posedů.

2.2 Fotopasti

Fotopasti se používají k monitorování pohybu nebo událostí v určitém prostoru, a to i bez přímého dohledu člověka. Toto monitorování je prospěšné pro různou sortu lidí. Například pro vědce pro sběr dat, pro myslivce pro zjišťování migračních tras, počtu jedinců v dané oblasti nebo k pozorování chování zvířat v jejich přirozeném prostředí. Dále je mohou používat i zemědělci, kteří je využívají k hlídání polí proti poškození plodin od zvěře [26].

Fotopasti si navíc nevyužívají pouze pro myslivost a zemědělství. Lidé tento vynález používají také pro ochranu svého soukromého majetku. Zabezpečují si své pozemky v jejich nepřítomnosti. Fotopasti zachytí zloděje nebo vandala při činu. Díky pořízeným fotografiím se zvýší šance na navrácení ukradených předmětů a dostihnutí viníka [26].

Fotopast se skládá z kamery, elektroniky pro uložení dat na paměťové médium, jako je například SD karta. Dále nějaký druh přísvitu pro focení v noci nebo horší viditelnosti. Většinou je to infračervený přísvit, který zapříčiní, že budou fotky pouze černobílé [3]. Nejdůležitější, co z fotopasti dělá fotopast, je spoušť, díky které se bez přítomnosti člověka vytvoří fotografie a uloží ji [24].

V historii se k tomuto používal natažený drát, kde musela zvěř o drát zavadit, aby se vytvořila fotografie. Teď se jako spoušť využívají jiné metody. Natažený drát může na podobném principu nahradit světelná závora. Ta se skládá ze dvou zařízení. Jedno zařízení vysílá většinou infračervený paprsek, ale může jít i o viditelné světlo. Druhé zařízení přijímá tento paprsek. Pokud je paprsek přerušen a přijímač přestane přijímat paprsek, tak se fotopast sepne. Toto má řadu nevýhod. Stejně jako u obyčejného nataženého drátu musí procházet zvěř přesně vymezeným prostorem, aby se vytvořila fotografie. Dalším problémem této metody je spotřeba energie, kde závora musí být stále napájena [29]. To znamená, že baterie musí být větší a tím je celá fotopast nákladnější. Pro tyto účely se používá čidlo PIR (anglicky *passive infrared sensor*). Další informace o něm jsou v sekci 2.2.

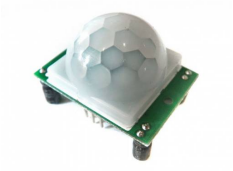
U výběru fotopasti jsou důležité některé zásadní parametry. V první řadě typ přísvitu. Lze si vybrat fotopasti s viditelným infračerveným přísvitem, tak i neviditelným. Neviditelný je nejvhodnější pro monitorování lesní zvěře, jelikož ji nevyplaší. Dále je důležitá rychlost spouště. Pokud je cílem monitorovat zvěř za pohybu, je potřeba, aby fotopast spustila a vytvořila fotografii co nejrychleji. Doba spouště se liší podle různých modelů a může být nízká jako 0,5 sekundy. Velikost u fotopastí pro monitorování zvěře je také důležitá. Menší velikost dovolí fotopast lépe ukrýt, aby si jí zvěř nevšimla. Fotopasti mají možnost vyfotit několik fotografií po sepnutí čidla PIR nebo rovnou natočit video. Vybrat si můžeme, zda chceme fotopast, která dokáže komunikovat pomocí mobilní sítě GSM. Toto umožňuje v momentě detekce informovat majitele a dokonce mu odeslat pořízenou fotografii nebo video. U kamery je hlavním parametrem rozlišení. Výrobci často uvádějí rozlišení fotografie, která je při pořízení interpolována na vyšší rozlišení, to ale nezvýší kvalitu pořízené fotografie. Takže kamera, která je použitá ve fotopasti, pořizuje fotografie o rozlišení 5 Mpx, ale díky interpolaci jsou zvětšeny na 20 Mpx. Dalším aspektem je i úhel záběru fotopasti, kdy při pořizování fotografií z blízkosti je potřeba mít širokoúhlou kameru. Baterie se v jed-

notlivých fotopastech může lišit. Podle typu fotopasti může být zařízení napájeno pomocí tužkových baterií AA nebo dobíjecích Li-Ion baterií. Některé umožňují i připojení externí olověné 12 V baterie z automobilu. Výdrž se liší podle typů napájení a může se pohybovat od řádů týdnů až po rok [24].

Čidlo PIR

Čidlo slouží k detekci pohybu objektů vyzařujících tepelné záření, jako jsou lidé nebo zvířata. PIR čidlo pracuje na základě detekce změn v úrovni infračerveného záření v prostředí. Každý teplokrevný objekt (člověk, zvíře) vyzařuje infračervené záření, které čidlo dokáže zachytit [12].

O zachycení infračerveného záření se stará pyroelektrický materiál. Tento materiál mění svůj elektrický náboj při změně teploty. Z tohoto materiálu se vytvoří snímače, které se poskládají v čidlu tak, aby se snímače zároveň vyrušily, pokud budou mít stejný náboj. Čidlo je takto postaveno, aby prostředí, ve kterém se nachází, neovlivnilo snímání. Lidské nebo zvířecí tělo spustí nejdřív jeden snímač a pak další při pohybu. Proto se dá použít pouze pro detekci pohybu. Náboj, který tyto snímače vygenerují, se zesilovačem zesílí a poté se zjistí, zda byl detekován pohyb. Pro zvětšení zorného pole a ochranu snímačů od vlivů vlhkosti a prachu se využívají plastové Fresnelovy čočky [12]. Na obrázku 2.1 je Fresnelova čočka na čidlu PIR vidět.



Obrázek 2.1: Modul čidla PIR HC – SR501 pro použití s mikrokotroléry. Převzato z [5].

Existující fotopasti

Zde jsou uvedeny příklady existujících moderních fotopastí.

Bunaty Mini Full HD

Tato fotopast představuje kompaktní zařízení určené pro sledování přírody a ochranu majetku. Zařízení je 10 cm vysoké, 80 cm široké a 40 cm hluboké. Díky jeho malým rozměrům a maskovanému designu je snadno skrytelná v terénu. Disponuje kamerou schopným pořizovat fotografie v rozlišení až 32 Mpx interpolovaně a natáčet video ve kvalitě Full HD se záznamem zvuku. Kamera má zorný úhel 52° a spouští se s reakční dobou přibližně 0,5 s. Neviditelný infračervený přísvit s vlnovou délkou 940 nm poskytuje dosvit 14 m. Zařízení je napájeno čtyřmi bateriemi typu AA, přičemž je možné použít také externí napájecí zdroj. Pro ukládání záznamů slouží micro SD karta o kapacitě až 32 GB. Ovládání probíhá prostřednictvím 2"barevného LCD displeje umístěného na přední straně a menu je kompletně v češtině. Mezi doplňkové funkce patří funkce časovače a časosběru. Časovač určuje čas, kdy je fotopast aktivní. Časosběr pravidelně v nastavených intervalech pořídí fotografii.

Lze nastavit jak dlouhé video se má při natáčení pořídit. Lze také nastavit, aby se pořídila fotografie a video zároveň. Případně ještě poskytuje možnost sériového pořízení fotografií, které při jednom sepnutí fotopasti pořídí za sebou více fotografií. Fotopast je možné zaheslovat, aby se nikdo neoprávněný nedostal k fotografiím [16]. Na obrázku 2.2 je vidět tento model fotopasti.

Obrázek 2.2: Fotopast Bunaty Mini Full HD. Převzato z [16].

Bunaty WIDE Full HD GSM

Tato fotopast díky širokému úhlu záběru 80° pokrývá větší oblast než běžné fotopasti, což je ideální pro sledování rozsáhlých ploch z krátké vzdálenosti. Snímač s rozlišením 5 Mpx a možností ukládat fotografie s rozlišení 24 Mpx interpolovaně umožňuje pořizovat kvalitní fotografie a videa v kvalitě Full HD. Neviditelný infračervený přísvit o vlnové délce 940 nm zajišťuje dosvit až 25 m. Integrovaný GSM modul s podporou 4G umožňuje okamžité odesílání snímků na e-mail, cloudové úložiště nebo mobilní telefon, a to do jedné minuty od zaznamenání pohybu. Zařízení je vybaveno 2,4"barevným LCD displejem pro snadné nastavení a prohlížení záznamů, laserovým ukazovátkem pro přesné zaměření pozorované oblasti. Doplňkové funkce a režimy pořízení záznamu jsou stejné jako u modelu Bunaty Mini. Napájení zajišťuje osm AA baterií s možností připojení externího zdroje. Fotopast je odolná vůči prachu a vodě (IP67) a její robustní konstrukce chrání vnitřní komponenty před nárazy a pády. Podporuje paměťové karty až do kapacity 32 GB [17]. Na obrázku 2.3 je možné tuto fotopast vidět.

Obrázek 2.3: Fotopast Bunaty WIDE Full HD GSM. Převzato z [17].

Kapitola 3

Detektory objektů pomocí neuronových sítí

V této kapitole jsou shrnuty základní informace o neuronových sítích, a je rozdělena následovně. V sekci 3.1 jsou informace o neuronových sítích a počítačovém vidění. Modely objektové detekce jsou popsány v sekci 3.2. Možnosti, jak tyto modely použít na mikrokontrolérech, se nacházejí v sekci 3.3. Existující detektory lesní zvěře jsou v sekci 3.4. Dataset s lesní zvěří se nachází v sekci 3.5.

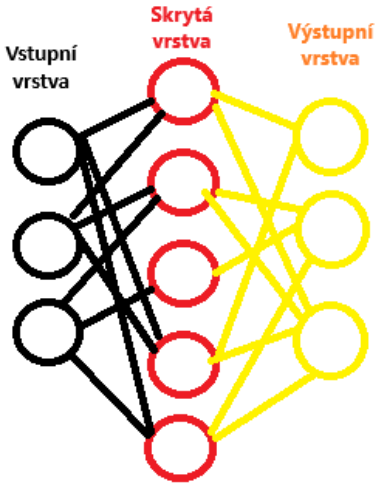
3.1 Detekce objektů

V této sekci jsou rozebrány základní informace o umělých neuronových sítích a počítačovém vidění.

Umělé neuronové sítě

Detekce objektů je založena na umělých neuronových sítích. Tyto sítě jsou tvořeny umělými neurony. Umělý neuron funguje tak, že pomocí vstupů do umělého neuronu se vrací požadovaný výstup. Toho je dosaženo díky aktivační funkci, která rozhodne, jaký bude mít výstup. Jako vstup do aktivační funkce se dají součiny vah jednotlivých vstupů a hodnoty vstupu. Vstupem neuronu bývá jiný neuron [15].

Připojením umělých neuronů za sebou nám vzniká umělá neuronová síť. Na vstupu umělé neuronové sítě je vstupní vrstva neuronů, která přijímá data k zpracování. Dále jsou data zpracována přes skrytou vrstvu neuronů a ta posílá data do výstupní vrstvy, kde lze vyčíst hodnoty výstupu po zpracování [10]. Pomocí správného nastavení vah vstupů neuronů budeme dostávat lepší výsledky na výstupech sítě. Jednoduchá neuronová síť je ilustrována na obrázku 3.1.



Obrázek 3.1: Obrázek jednoduché umělé neuronové sítě rozdělené na vrstvy. Kolečka jsou umělé neurony, který jsou mezi sebou propojeny a každé propojení má svou váhu.

Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) jsou jako umělé neuronové sítě, ale u vyhodnocování výstupů používají konvoluci. Tato funkce je vhodná pro zpracovávání obrázků. Díky konvoluci můžeme extrahovat rysy (nebo také vlastnosti, anglicky *features*) obrázku. Například zvýraznění hran objektů na obrázku. V neuronové síti se přidají takzvané konvoluční vrstvy, kde probíhá zpracování konvoluce. Základem konvoluce je takzvané jádro (anglicky *kernel*). Jedná se o matici, která funguje jako filtr pro extrakci nějaké určité vlastnosti z původního obrázku. Skalárním součinem původního obrazu a konvolučního jádra vzniká obrázek, kde jsou zvýrazněny ty vlastnosti, jaké určíme jádrem. V konvoluční síti bývá více konvolučních vrstev [28].

Velikost výstupu po konvoluční vrstvě je menší podle velikosti matice použitého jádra pro extrakci. Výstup po zpracování konvoluční vrstvou se nazývá aktivační mapa (*feature map*).

Konvoluční neuronové sítě většinou obsahují i *pooling* vrstvy, které zmenšují výstupy konvoluční vrstvy, pro zlepšení rychlosti výpočtu. Tyto vrstvy provádí takzvaný *pooling*, což znamená, že se použije maticový filtr, který vrátí maximální hodnotu oblasti, na kterou je matice aplikována. Matice je velká, aby redukovala velikost výstupů konvolučních vrstev. Toto má za následek, že může být ztracena informace a může být znehodnocena práce konvoluční vrstvy [28].

Hluboké učení

Hluboké učení je aplikováno z principu na neuronovou síť, která má mnoho skrytých vrstev. Proces učení probíhá upravováním vah na jednotlivých skrytých vrstvách.

Upravování vah neuronů probíhá pomocí algoritmu zpětné šíření chyby. Je využíván při učení neuronové sítě s učitelem. Tento algoritmus se snaží snížit při každém průchodu sítí chybovou funkci. Učení s učitelem probíhá u počítačového vidění tak, že připravíme obrázky, kde je nějaký předmět, který chceme detekovat, a označíme ho. Označování probíhá

pomocí takzvaných bounding boxů. Pro co nejlepší výsledky potřebujeme co nejvíce takto připravených obrázků [13].

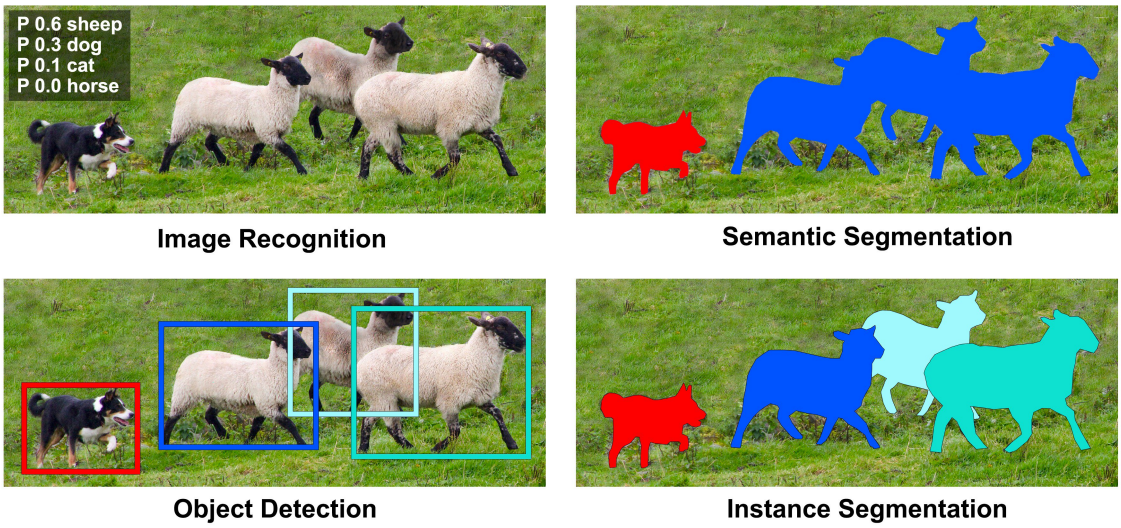
Když trénujeme neuronovou síť moc dlouho pouze na stejných datech, tak může nastat takzvaný *overfitting*. To znamená, že je síť přetrénovaná a funguje pouze pro data, na kterých byla trénována [37].

Počítačové vidění

V počítačovém vidění se zabývá tím, jak využít neuronové sítě a hluboké učení na to, aby se automaticky detekovaly objekty, které jsme neuronovou síť naučili. Po počítačovém vidění je vyžadováno těchto úloh:

- **detekce objektu**, kde se nachází v obrázku. Označí ho bounding boxem [2].
- **klasifikace**, kde z detekovaných objektů se rozhoduje, o jaký objekt se zrovna jedná. Ke klasifikovaným objektům přidá značku například stůl nebo židle [2].
- **segmentace**, která slouží k tomu, aby přiřadila pixely z obrázku, které patří přímo detekovanému objektu. Ta může být podle aktuální detekce (instance segmentation) nebo podle klasifikace objektu (*semantic segmentation*) [2].

Na obrázku 3.2 jsou vidět jednotlivé úlohy počítačového vidění.



Obrázek 3.2: Nahoře vlevo je vidět klasifikace, dole vlevo detekce objektů, nahoře vpravo segmentace objektů podle klasifikace, dole vpravo segmentace podle detekovaných objektů. Převzato z [36].

3.2 Modely detekce objektů

Modely objektové detekce jsou klíčovým prvkem v oblasti počítačového vidění. Tyto modely lze obecně rozdělit na tři hlavní komponenty: páteř (*backbone*, dále jako extraktor rysů) a hlavu (*head*). Podle architektury jednotlivých detektorů se zde nachází ještě vrstva, která propojuje extraktor rysů a *head*. Tato vrstva se nazývá FPN (*feature pyramid network*). Každá z těchto částí má specifickou funkci, kterou přispívá k celkovému výkonu modelu.

Extraktor rysů

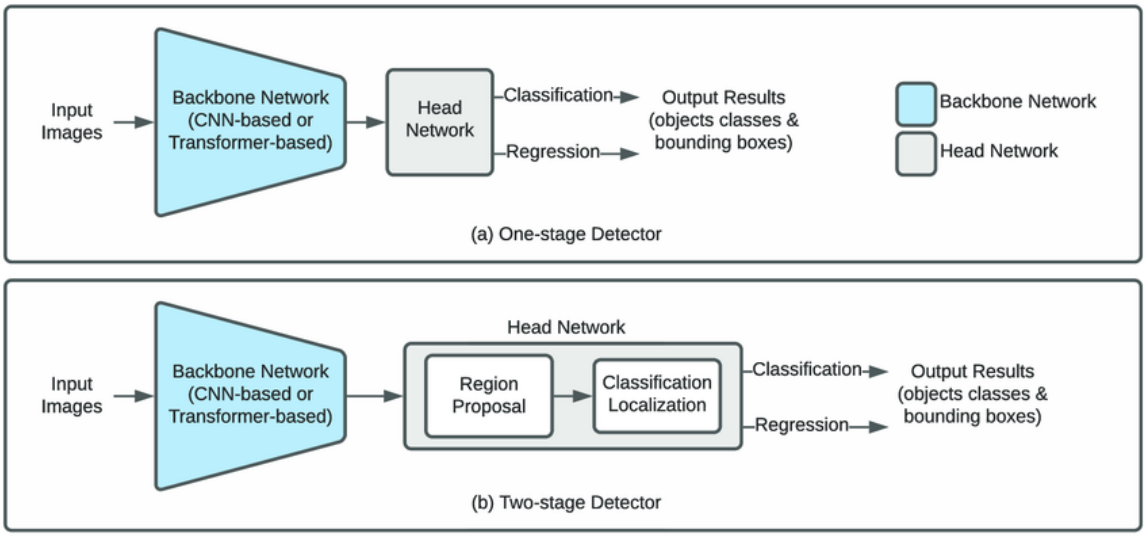
Extraktor rysů je klíčovou částí modelu, která extrahuje významné rysy z obrazu. Typicky se jedná o hluboké konvoluční neuronové sítě původně navržené pro klasifikaci obrazů, jako jsou ResNet, VGG nebo EfficientNet. Tyto sítě převádějí vstupní obraz na hierarchii rysů, které zachycují různé úrovně abstrakce od základních hran po složité textury a objekty. Výběr extraktoru má přímý vliv na přesnost a rychlost modelu [11].

Head

Head modelu je zodpovědná za interpretaci extrahovaných rysů. Tato část modelu provádí úkoly, jako je lokalizace objektů (detekce hranic) a klasifikace jednotlivých objektů v obraze. Existují různé přístupy ke konstrukci, například [19]:

- **Jednofázové detektory (*Single-stage detectors*)**: Jako jsou YOLO nebo SSD, které kombinují lokalizaci a klasifikaci do jednoho kroku, což vede k rychlým predikcím.
- **Dvoufázové detektory (*Two-stage detectors*)**: Například Faster R-CNN, který nejprve generuje regiony zájmu (*Region Proposals*) pomocí vrstvy *Region Proposal Network* (RPN) a poté provádí přesnou klasifikaci a lokalizaci.

Na obrázku 3.3 lze vidět rozdělení konstrukce *head*.



Obrázek 3.3: v bodu (a) je vidět, že část *head* udělá detekci a klasifikaci najednou. V bodu (b) je vidět, že se nejdříve vygenerují regiony zájmu a pak teprve detekuje a klasifikuje objekt. Převzato z [19].

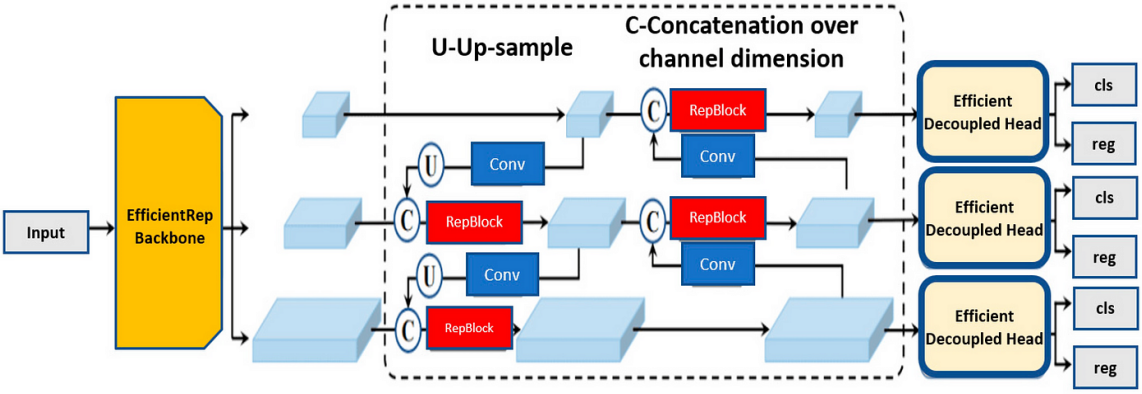
Feature Pyramid Network

V některých modelech se používají specializované detekční *head*, které jsou optimalizovány pro specifické úkoly, například pro víceúrovňovou detekci objektů různých velikostí nebo pro predikci klíčových bodů. Tyto *head* často zahrnují další vrstvy, jako jsou FPN (Feature Pyramid Networks), které pomáhají zlepšit detekci malých objektů [22].

Úspěch modelu objektové detekce závisí na efektivní spolupráci všech tří hlavních částí. Extraktor rysů musí poskytovat kvalitní a informativní rysy, které *head* dokáže efektivně zpracovat. Vrstva FPN a její integrace do modelu pak může výrazně ovlivnit celkový výkon, zejména v náročných scénářích, jako je detekce malých objektů nebo práce s nekvalitními daty.

YOLO

YOLO (*You Only Look Once*) je běžná architektura v oblasti počítačového vidění. Vyznačuje se rychlostí a efektivitou, díky čemuž je vhodná pro aplikace v reálném čase. Základní princip YOLO spočívá v tom, že rozdělí vstupní obraz na mřížku a poté předpovídá jak ohraničující boxy, tak klasifikuje objekty při jediném průchodu neuronovou sítí jako jednofázový detektor. Na obrázku 3.4 je nakreslená architektura [1].



Obrázek 3.4: Architektura modelu YOLOv8. Vstup jde do extraktoru rysů, která vytvoří aktivační mapy, které jdou do vrstvy FPN. Výstupy z ní zpracovává *head*. Převzato z: [1].

Vývoj a srovnání verzí YOLO

Architektura YOLO prošla od své první verze až po současné modely značným vývojem. Na těchto vybraných verzích je vývoj popsán [1]:

- **YOLOv1**: První verze YOLO byla zveřejněna v roce 2016. Měla rychlou detekci, ale nižší přesnost u malých objektů a překrývajících se objektů.
- **YOLOv2 a YOLOv3**: Tyto verze přinesly vylepšení ve formě lepšího rozpoznávání víceúrovňových rysů díky použití technik, jako je FPN. YOLOv3 je stále často používán díky své vyváženosti mezi přesností a rychlostí.
- **YOLOv4 a YOLOv5**: Tyto verze se zaměřily na optimalizaci trénovacího procesu. YOLOv5 je široce využíván díky své jednoduchosti a podpoře běhu na zařízeních s málými výpočetními zdroji.
- **YOLOv7 a YOLOv8**: Verze se zaměřují na zlepšení detekce malých objektů a integraci segmentace.

Páteřní sítě a použití FPN

V architektuře YOLO hraje důležitou roli extraktor rysů. V různých verzích YOLO byly využity různé extraktory rysů, například [1]:

- **YOLOv3**: Používá Darknet-53, hlubokou konvoluční síť optimalizovanou pro rychlost a přesnost.
- **YOLOv4**: Integruje CSPDarknet53, což je vylepšená verze Darknet-53 s *cross-stage partial connections*, které zlepšují efektivitu výpočtů.
- **YOLOv5**: Umožňuje flexibilní volbu extraktoru rysů včetně lehčích variant pro vestavěných zařízení.
- **YOLOv7 a YOLOv8**: Zavádějí další optimalizace a pokročilé architektury pro zvýšení přesnosti a rychlosti detekce.

FPN se objevuje od verze YOLOv3 a vyšších. FPN umožňuje modelu efektivně kombinovat rysy z různých úrovní páteřní sítě, což je zvláště užitečné pro detekci objektů různých velikostí. Díky tomu jsou moderní verze YOLO schopny dosáhnout vysoké přesnosti při detekci malých i velkých objektů [1].

Použitelnost na mikrokontrolérech

Verze jako YOLOv5 a YOLOv7 mají odlehčené modely (nano nebo tiny), které jsou navrženy speciálně pro tato omezení [39].

Pro nasazení na mikrokontrolérech, jako je Raspberry Pi nebo ESP32, se často využívají modely kvantizované pomocí TensorFlow Lite Micro (další informace v sekci 3.3. Tyto optimalizace umožňují běh detekce objektů v reálném čase při minimální spotřebě energie a bez potřeby velkých výpočetních zdrojů.

Faster R-CNN

Faster R-CNN je vylepšení předchůdce Fast R-CNN. Konkrétně vylepšili model o modul RPN. Toto výrazně zlepšilo rychlost učení modelu a jeho provozu v reálném čase. Jedná se o dvoufázový detektor. Jeho přesnost detekce je hodně vysoká ve srovnání s jednofázovými detektory. I když se výpočetní rychlost a rychlost učení dost zlepšily, tak ne na tolik, aby se dal použít na zařízeních s omezenými výpočetními zdroji, jako jsou mikrokontroléry ESP32. Dá se použít na zařízeních s grafickou kartou, kde dosahoval model 5 snímků za sekundu [32].

Původní verze R-CNN používala algoritmus selective search, aby predikovaly regiony zájmu. Tyto pravděpodobné výskyty se zpracovaly konvolučními neuronovými sítěmi a výsledky prošly ke klasifikaci a regresi. Každý region zájmu prošel celým procesem, než se mohl zpracovat další. Fast R-CNN přišlo s tím, že se regiony zájmu vygenerují přímo z aktivační mapy. Regiony zájmu se generují stále stejným způsobem, ale vytvoří se všechny najednou. Poté projde vrstvou *Region of Interest Pool* (RoI Pool), která regiony zájmu zarovná s extrahovanými rysy. To dovolilo, že mohl model postupovat v architektuře pouze ke klasifikaci a regresi. V Faster R-CNN k generování regionů zájmu byl použit již modul RPN. Zde již používali předtrénované páteřní sítě pro generování aktivační mapy. A dále se napojuje na stejnou konstrukci jako Fast R-CNN [34].

FOMO

FOMO představuje lehkou architekturu pro detekci objektů, která byla navržena speciálně pro zařízení s omezenými výpočetními zdroji. Na rozdíl od modelů detekce objektů, jako jsou YOLO nebo Faster R-CNN, FOMO neidentifikuje ohraničující rámečky, ale místo toho lokalizuje objekty v místě zvaném *centroid*. *Centroid* je bod v centru pravděpodobného výskytu objektu (*heat map*). Obrázek se rozdělí do mřížky a při detekci objektu je jako centroid označena nejbližší buňka [6].

Hlavní vlastnosti

FOMO je navrženo s ohledem na nízké požadavky na výpočetní zdroje, a proto může běžet na zařízeních s méně než 200 KB RAM. I přes tuto úspornost nabízí vysokou rychlost. Na výkonnějších zařízeních, jako je Raspberry Pi 4, může dosahovat až 60 snímků za sekundu. Implementace modelu je navíc velmi jednoduchá díky přímé integraci s platformou Edge Impulse (viz sekce 3.3), což umožňuje snadné trénování i nasazení modelů přímo na cílovém zařízení. FOMO se soustředí na efektivní detekci objektů pomocí určení jejich polohy. Neposkytuje detailní informaci o velikosti, ale detekuje pouze centroid objektu, což je dostatečné pro mnoho praktických aplikací [21]. Tento princip je ilustrován na obrázku 3.5.

Obrázek 3.5: Data k učení musí být označeny bounding boxem, jak je vidět nahoře. Dole je vidět, že model detekuje a klasifikuje pouze centroid objektu. Převzato z [6].

FOMO je ideální pro aplikace, kde objekty mají podobnou velikost a nejsou příliš blízko u sebe. Když jsou objekty moc u sebe, tak se může stát, že se budou detekovat jako jeden objekt. To se dá vyřešit tím, že mřížka bude nastavena tak, aby každý pixel byl jednou buňkou v mřížce. To však zvýší náročnost výpočtu. FOMO také neumožňuje, aby se detekovaly objekty, které se překrývají, pokud mají centroid ve stejné buňce mřížky [6].

3.3 Možnosti provozu detekčních modelů na nízkoenergetických mikrokontrolérech

Kvůli výpočetní náročnosti strojového učení a jeho modelů se většina práce vykonává ve cloudových službách nebo na výkonných počítačích. Díky technologii nazvané TinyML lze modely strojového učení spouštět i na nízkoenergetických mikrokontrolérech.

TinyML

Zmenšení potřeby výpočetních zdrojů je v tinyML realizováno pomocí těchto praktik:

- **Kvantizace** – Kvantizace převádí reprezentaci modelových parametrů (například váhy a aktivace) z vyšší bitové hloubky (32bitových čísel s pohyblivou řádovou čárkou) na nižší bitovou hloubku (například 8bitová celá čísla). Tento postup dokáže mnohonásobně zmenšit velikost celého modelu. Při tomto se ztrácí však přesnost výpočtu modelu. To však díky algoritmům, které jsou dále popsány v této práci [31] nemá až tak moc velký dopad na přesnost.

- ***Pruning*** (prořezávání) – *Pruning* odstraňuje nevýznamné nebo málo využívané parametry (váhy) a spoje neuronové sítě. Analýzou modelu se identifikují váhy s malým absolutním vlivem na výsledek, to jsou třeba váhy spojů mezi neurony blížící se nule. Tím se sníží počet počítaných spojů neuronů s minimálním vlivem na přesnost a zároveň velikost modelu se zmenší [20].

- ***Fusing*** (spojování) – Spojuje některé operátory modelu do jednoho operátoru, který se bude chovat stejně jako jejich kombinace a lze vykonat v jednom kroku [31].

Některé mikrokontroléry používají hardwarovou akceleraci. Ta používá SIMD (*Single Instruction, Multiple Data* – dovoluje v jedné instrukci vypočítat více dat) instrukce. Případně, pokud má zařízení modul NPU (neurální procesorová jednotka), tak ho využívá k výpočtům modelů. Hardwarová akcelerace zrychluje samotný výpočet modelů.

TinyML můžeme používat pomocí implementací v knihovnách nebo v cloudových službách. Zde jsou příklady [31]:

- **TensorFlow Lite Micro (TFLM)** – Rozšíření TensorFlow Lite pro mikrokontroléry. Tato knihovna umožňuje a běh modelů v řádech stovek KB velikosti. Podporuje modely vytvořené v knihovně TensorFlow. Umožňuje použití modelu v jazycích Python, C, C++ a Java. Neumožňuje trénovat model přímo na zařízení, takže musíme použít předtrénovaný model [30].

- **uTensor** – Lehký framework zaměřený na nasazení modelů na IoT (Internet of Things) zařízeních. Modely trénované v Keras jsou konvertovány do C++ pro snadnou integraci.

- **Edge Impulse**[1] – Cloudová platforma pro trénování a nasazení modelů TinyML, například model FOMO. Podporuje export trénovaných modelů na edge zařízení. To znamená, že je možný udělat na této platformě celý proces od tvoření dat pro trénování modelu, přes trénování, až po vytvoření modelu pro nasazení na zařízení.

[1] Tato služba lze používat na webové stránce: https://edgeimpulse.com/

- **NanoEdge AI Studio** – Nabízí optimalizaci knihoven a emulaci výkonu před nasazením modelů. Podporuje detekci anomálií a klasifikaci na platformách jako STM32 a Arduino .

TinyML umožnilo využívat síly modelů strojového učení, bez potřeby připojení k internetu a používání vytrénovaného modelu v cloudových službách. To zlepší odezvu na naměřené hodnoty ve svém okolí a vykonaného výpočtu. Znamená to, že mohou zařízení dostat data ze svého prostředí (například fotografii) a ihned ji zpracovat, a po dokončení výpočtů mohou přejít do režimu spánku. Díky tomu mohou být tyto mikrokontroléry nezávislé na ostatních zařízeních [31].

3.4 Existující detektory zvěře

Existují již funkční detektory zvířat. Většinou se ovšem jedná o detektory, co jsou naučeny na exotickou zvěř, jako jsou tygři, hroši a tak podobně. Detektory by se museli naučit přímo na detekci lesní zvěře, která se vyskytuje na území České republiky.

YOLO-Animal

YOLO-Animal je vylepšená verze založená na YOLOv5s, která byla speciálně navržena pro detekci divokých zvířat v reálném čase [23]. Klíčová vylepšení zahrnují:

- **BiFPN (*Weighted Bidirectional Feature Pyramid Network*)**: Tento modul umožňuje pokročilé slučování rysů z různých úrovní páteřní sítě. Díky tomu se zlepšuje detekce malých a částečně zakrytých objektů. BiFPN vylepšuje FPN [35].
- **ECA (*Effective Channel Attention*)**: Tento mechanismus přidává selektivní pozornost na aktivační mapy, což zlepšuje extrakci klíčových rysů a odolnost vůči šumu v obraze [38].

Tyto vylepšení činí YOLO-Animal ideálním řešením pro detekci divokých zvířat v komplexních podmínkách, jako jsou špatné osvětlení, částečné zakrytí nebo nejasné rozdíly mezi blízkými a vzdálenými objekty. Zvýšilo to přesnost detekce oproti YOLOv5s [23].

WilDect-YOLO

WilDect-YOLO je pokročilý model založený na architektuře YOLOv4, navržený speciálně pro detekci ohrožených druhů zvěře v reálném čase. Klíčová vylepšení tohoto modelu zahrnují [33]:

- **DenseNet bloky**: Díky propojení všech vrstev v rámci bloku se efektivně přenášejí a znovu využívají diskriminační rysy, což zvyšuje přesnost modelu a zlepšuje jeho schopnost detekovat objekty v složitých podmínkách.
- **Reziduální bloky CSPX1-n a CSPX2-n**: Tyto modifikované reziduální bloky vylepšují extraktor rysů CSPDarknet53, čímž zajišťují hlubší extrakci prostorových rysů a zvyšují rozlišovací schopnosti modelu.
- **SPP (*Spatial Pyramid Pooling*)**: Rozšiřuje receptivní pole modelu a zajišťuje lepší detekci objektů v různých měřítkách bez nutnosti změny velikosti vstupních obrázků.

- **Vylepšený PANet**: Zajišťuje efektivní sloučení vysokých a nízkých úrovní rysů ve více měřítkách, čímž zachovává jemně lokalizované informace a zlepšuje přesnost detekce.

3.5 Existující datasety

Většina dostupných datasetů na internetu obsahuje fotografie různých druhů zvířat, ale většinou zde nejsou zahrnuta ta důležitá pro tuto práci. Pro účely této práce jsou potřeba datasety, které obsahují fotografie srnek a divokých prasat.

Na webových stránkách Roboflow[2] existují datasety vytvořené uživateli. Zde se nachází některé datasety, které by byly vhodné pro trénování modelu. Následující datasety budou využity k trénování modelu:

- **My Game Pics** – Jedná se o dataset od uživatele *My Game Pics*. Tento dataset byl vytvořen pomocí záběrů z fotopastí. Objevují se jak barevné fotografie vyfocené za dne, tak i černobílé fotografie vyfocené v noci za infračerveného přísvitu. Nachází se zde fotografie srnek a divokých prasat. Jsou zde i fotografie kojotů, králíků a mývalů, které však v rámci této práce nejsou potřebné. Počet fotografií divokých prasat je 1 348 a fotografií srnek je 2 280. Všechny fotografie jsou anotovány, což umožní jednodušší práci [25].

- **Forest** – Tento dataset je od uživatele *Nicolasfree*. V tomto datasetu se nachází fotografie lesů v různých ročních období. Fotografie jsou foceny za různých světelných podmínek. Jednotlivé fotografie jsou anotovány podle ročních období a počasí. Je tu 7 463 fotografií lesa. Tento dataset by naučil model, aby si nepletl pozadí fotografie (les) se zvířetem a nedocházelo k častým falešným detekcím zvěře [27].

[2] https://universe.roboflow.com/

Kapitola 4

Návrh

V této kapitole popíšu informace o platformě ESP32 v sekci 4.1. V sekci 4.2 je popsána síť LoRaWAN. Následně v sekcích 4.3, 4.4, 4.5 a 4.6 detailně popíšu návrh dvou zařízení na platformě ESP32, která by byla schopná detekce lesní zvěře a zaslání informace o detekci pomocí sítě LoRaWAN na server The Things Network. V sekci 4.7 je popsán návrh jednoduché aplikace, která by informaci o detekci stáhla ze serveru a zobrazila ji uživateli.

4.1 Platforma ESP32

Výběr mikrokontrolérů z řady ESP32 od firmy Espressif Systems je v současnosti velmi rozsáhlý. V této práci se zaměřuji pouze na dvě specifické řady, a to ESP32-S3 a ESP32-P4, které jsou reprezentovány vývojovými deskami ESP32-S3-EYE a ESP32-P4-Function-EV-Board. Tyto desky disponují dostatečným výpočetním výkonem pro nasazení a provoz modelů strojového učení pomocí TinyML.

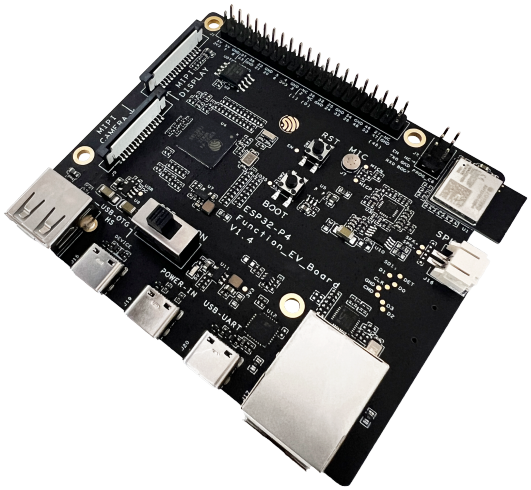
ESP32-S3-EYE

Vývojová deska ESP32-S3-EYE je rozdělena na dvě části – hlavní část s ESP32 modulem a samostatnou část s displejem. Srdcem celé desky je modul ESP32-S3-WROOM-1, který obsahuje čip ESP32-S3R8. Tento čip je vybaven dvoujádrovým Xtensa® 32bitovým LX7 mikroprocesorem taktovaným na 240 MHz a nabízí konektivitu Wi-Fi i Bluetooth 5 Low Energy. Dále obsahuje specializované SIMD instrukce pro urychlení výpočtů neuronových sítí a zpracování signálů. Modul má integrovanou paměť 8 MB PSRAM typu Octal SPI a rovněž 8 MB flash paměti pro ukládání dat. Deska je vybavena kamerou OV2640 s rozlišením 2 MPx a úhlem záběru 66,5°, přičemž maximální rozlišení činí 1600 na 1200 pixelů. Pro záznam zvuku je integrován digitální MEMS mikrofon s rozhraním I2S. Zobrazovací jednotku tvoří 1,3"LCD displej připojený přes SPI rozhraní. Nechybí ani slot pro micro SD karty a tříosý akcelerometr QMA7981. Napájení je možné prostřednictvím připojení Li-ion baterie, kterou lze nabíjet přímo přes USB díky integrované lineární nabíječce [9]. Na obrázku 4.1 lze toto zařízení vidět.

Obrázek 4.1: Vývojová deska ESP32-S3-EYE. Převzato z: [9].

ESP32-P4-Function-EV-Board

ESP32-P4-Function-EV-Board je velmi výkonná vývojová deska, která se svou výbavou přibližuje jednočipovým počítačům, jako je Raspberry Pi. Je vybavena dvoujádrovým 32bitovým RISC-v procesorem s taktovací frekvencí až 400 MHz. K dispozici je až 32 MB PSRAM a 16 MB flash paměti. Deska obsahuje modul ESP32-C6-mini, který umožňuje komunikaci pomocí Wi-Fi 6 a Bluetooth 5 Low Energy. Pro vizuální výstup je osazen 7"kapacitní dotykový displej s MIPI DSI. Kamera SC2336 s rozlišením 2 MPx a maximálním rozlišením 1920 na 1080 pixelů komunikuje přes rozhraní MIPI CSI. Pro zpracování audia je k dispozici audio kodek ES8311, vestavěný mikrofon a výstup pro reproduktor. Napájet lze přes USB-C s napětím 5 V. Má vyvedených 28 vstupně-výstupních pinů a slot pro micro SD kartu [8]. Na obrázku 4.2 lze tuto vývojovou desku vidět.



Obrázek 4.2: Vývojová deska ESP32-P4-Function-EV-Board. Převzato z [8].

Možnosti programování

Tyto vývojové desky lze programovat v různých vývojových prostředích a pomocí různých jazyků, jako je C++ nebo MicroPython. Pro jednodušší projekty je vhodné využít prostředí **Arduino IDE**, které nabízí velké množství předpřipravených knihoven a používá Arduino framework. Další možností je **PlatformIO**, které funguje jako rozšíření například pro Visual Studio Code a také podporuje Arduino framework. Pro pokročilejší vývoj je určeno vývojové prostředí **ESP-IDF**. To je oficiální vývojové prostředí od společnosti Espressif Systems. Ten nově obsahuje systém ESP Component Registry, který umožňuje snadné přidávání komponent (knihoven) do projektů podobně jako v Arduino IDE nebo PlatformIO. ESP-IDF lze používat prostřednictvím vlastního rozhraní Espressif-IDE nebo pomocí rozšíření pro Visual Studio Code.

4.2 LoRaWAN

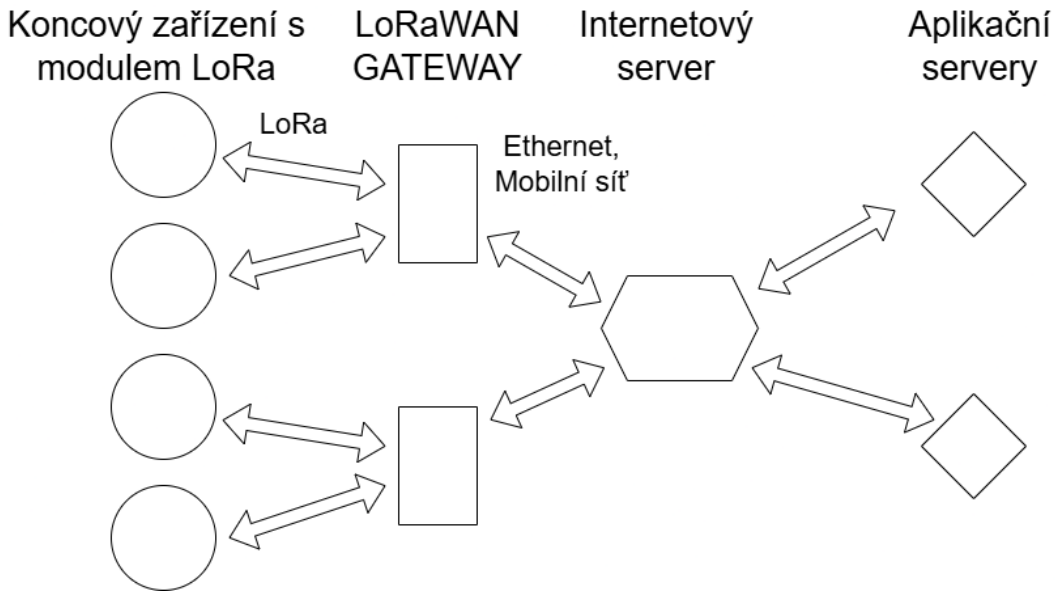
LoRa (Long Range) je bezdrátová přenosová technologie, která umožňuje energeticky úsporný přenos dat na velmi dlouhé vzdálenosti. Na základě této technologie vznikla síťová vrstva známá jako LoRaWAN, která nachází široké uplatnění v oblasti internetu věcí (IoT). V České republice síť pracuje v frekvenčním pásmu 868 MHz [4].

Srovnání LoRaWAN s jinými LPWAN technologiemi

Ve srovnání s jinými technologiemi LPWAN, jako jsou Sigfox, NB-IoT nebo LTE-M, nabízí LoRaWAN řadu výhod. Především vyniká dosahem – v městských oblastech pokrývá vzdálenosti 2 až 5 kilometrů, v příměstských oblastech kolem 15 kilometrů a ve venkovských oblastech dokonce až 45 kilometrů. Další předností je nízká spotřeba energie, díky čemuž mohou zařízení napájená z baterie vydržet v provozu po mnoho let. LoRaWAN klade důraz i na bezpečnost, využívá šifrování AES-128 a odděluje bezpečnostní vrstvy pro síť a aplikaci. Přenosové rychlosti se pohybují od 290 b za sekundu až do 50 kb za sekundu, což umožňuje přizpůsobit síť podle konkrétních potřeb. Technologie také vykazuje vysokou odolnost vůči rušení díky použití modulace chirp spread spectrum (CSS) a je velmi dobře škálovatelná, což znamená, že je vhodná i pro rozsáhlé sítě [4].

Topologie LoRaWAN

Topologie LoRaWAN sítě je hierarchická a označuje se jako **hvězda hvězd**. V této struktuře hrají klíčovou roli koncová zařízení, která odesílají data buď jednosměrně, nebo obousměrně. Tato zařízení komunikují s bránami (gateway), které slouží jako mezičlánek a předávají data dál do sítě pomocí Ethernetu, Wi-Fi nebo mobilní sítě. Následuje síťový server, který odpovídá za dekódování paketů, kontrolu bezpečnosti a správu přenosových parametrů. Typickým příkladem takového serveru je The Things Network. Na nejvyšší úrovni architektury se nachází aplikační vrstva, která zpracovává přijatá data a vykonává specifické úlohy. Tato síťová topologie přispívá k nízké spotřebě energie a umožňuje budování rozsáhlých a výkonných IoT sítí [4]. Topologie je znázorněna na obrázku 4.3.



Obrázek 4.3: Na obrázku je vyobrazena topologie sítě LoRaWAN.

Architektura LoRaWAN

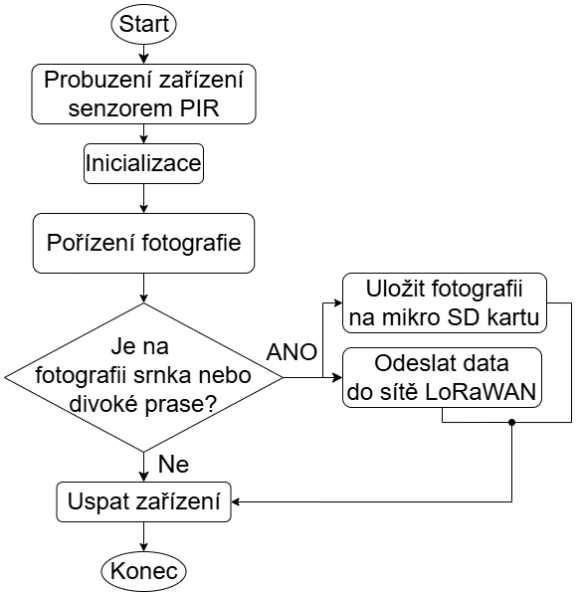
Architektura LoRaWAN je rozdělena do dvou základních vrstev. Fyzická vrstva využívá zmíněnou CSS modulaci, která zajišťuje dlouhý dosah a zároveň vysokou odolnost proti rušení. Druhou vrstvou je MAC vrstva, která se stará o řízení přístupu do sítě a optimalizaci komunikace mezi zařízeními a servery. Mezi klíčové funkce této architektury patří adaptivní úprava datové rychlosti (ADR), která přispívá k efektivnímu využívání sítě a prodloužení životnosti baterií v zařízeních. Bezpečnostní aspekt je zajištěn pomocí šifrování AES-128 přímo na úrovni datových paketů [4].

Třídy koncových zařízení

Důležitou součástí je plánování komunikace koncových zařízení se sítí, které v síti LoRaWAN probíhá podle definovaných tříd A, B nebo C. Každá třída představuje odlišný způsob řízení přenosu a rovnováhu mezi energetickou náročností a dostupností komunikace. Třída A je nejúspornější, protože zařízení umožní síti komunikovat zpátky se zařízením pouze krátce po odeslání zprávy. Třída B přidává možnost přijímat zprávy v plánovaných intervalech. Třída C pak umožňuje téměř nepřetržité přijímání zpráv, což snižuje latenci, ale zároveň výrazně zvyšuje spotřebu energie [14].

4.3 Základní koncepce obslužného programu zařízení

Z hlediska spotřeby by mělo být Zařízení většinu času uspáno a pracovat by mělo pouze, když PIR zaznamená pohyb. Poté se spustí program, který inicializuje kameru, LoRaWAN modul a čtečku micro SD karet. Dále pořídí fotografii. Tuto fotografii nechám zpracovat pomocí detekčního modelu. Podle výsledků se program rozhodne, co bude dělat dále. Buď na obrázku nenachází žádné zvíře a obslužný program uvede zařízení zpět do spánku, nebo se pokusí zaslat data s detekovanými zvířaty a zároveň, pokud je vložena micro SD karta do čtečky, tak do ní uloží originální fotku. Toto schéma je znázorněno i na obrázku 4.4.



Obrázek 4.4: Schéma obslužného programu.

Tento návrh by mohl zahltit velice rychle paměťovou kartu a zbytečně by zasílal zprávy o detekci. To by se stalo v případě, kdyby se zvíře před senzorem neustále pohybovalo, tak by se provádělo odeslání a zapsání dat do doby, než by odešlo ze záběru. Proto je potřeba také omezit časově, jak často má zařízení obraz zpracovávat. Například po každé úspěšné detekci si zapíše aktuální čas a na začátku programu po probuzení se zkontroluje, zda již minuta uplynula, jinak se zařízení znovu uspí. Tímto by se částečně ušetřila energie.

4.4 Formát dat k odeslání na server

Kvůli úspoře přenášených dat pomocí LoRaWAN sítě jsem se rozhodl posílat informaci pouze o tom, jaké zvíře bylo detekováno a jejich počet. Model také je omezen na rozeznávání srnek a divokých prasat. Formát vypadá následovně. Každý druh zvířete, co byl detekován, se zakóduje jako 2 bajty. První bajt určuje, o jaký druh jde, a druhý bajt určuje počet výskytů daného druhu. Srnka má v tomto formátu bajtovou hodnotu 0 a divoké prase má bajtovou hodnotu 1. Například když přijdou 4 bajty s hodnotami `[0, 2, 1, 1]`, to znamená, že model zaznamenal dvě srnky a jedno divoké prase.

4.5 Návrh zařízení

Pro srovnání a určení vhodnější vývojové desky pro bateriový provoz a zpracování obrazu jsem navrhl dvě zařízení. První zařízení je postavené okolo vývojové desky ESP32-S3-EYE. Druhé zařízení využívá větší vývojovou desku ESP32-P4-Function_EV_Board, konkrétně ve verzi 1.4. Obě zmíněná zařízení jsou vhodná pro lokální zpracovávání modelů objektové detekce. Jako model jsem z důvodu malé velikosti a vysoké rychlosti zvolil model FOMO. Jeho limitace z hlediska kolizí objektů by neohrozila detekci zvěře v obraze a následné zaslání dat na server. Ke komunikaci se sítí LoRaWAN jsem vybral modul, který již sám dodržuje specifikace sítě. Jedná se o modul Unit LoRaWAN EU868 od firmy M5Stack. Tento modul se připojuje pomocí rozhraní UART a ovládá se pomocí AT příkazů. To zaručuje správnou

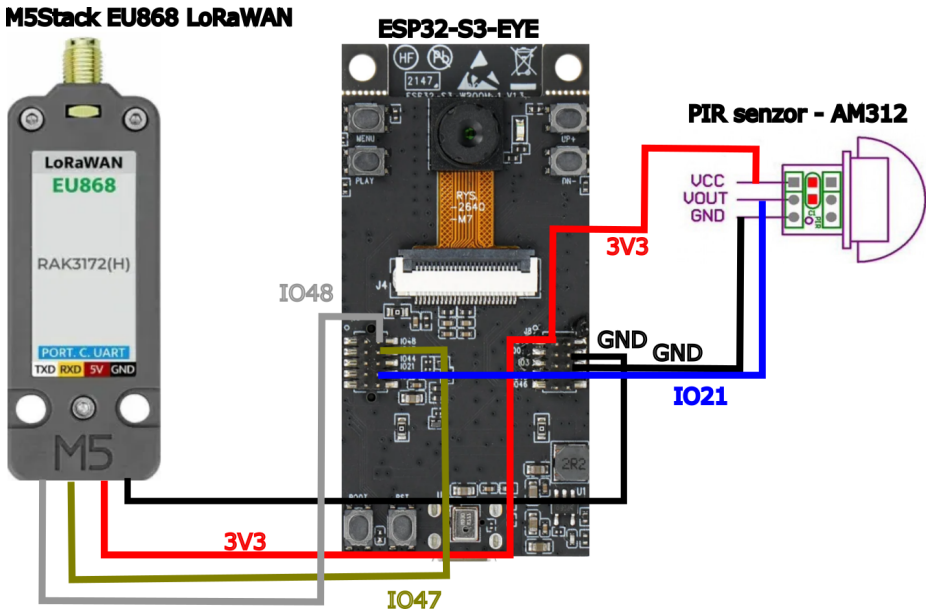
komunikaci se sítí. Toto řešení dovoluje komunikovat se sítí každému zařízení, které toto rozhraní má. Aby výsledné zařízení šetřilo energií a mohlo pracovat na baterii, je potřeba zakomponovat do systému také senzor PIR, který probudí zařízení ze spánku. Jako výborná volba se ukazuje senzor AM312. Tento senzor při detekci pohybu na výstupu pinu VOUT má logickou „1“. Vzdálenost detekce pohybu je přibližně 5 metrů. Jelikož ESP32-S3-EYE má již zabudovanou nabíječku Li-Ion baterií, tak jsem zvolil tento typ baterie i pro verzi s ESP32-P4-Function_EV_Board.

4.6 Zapojení

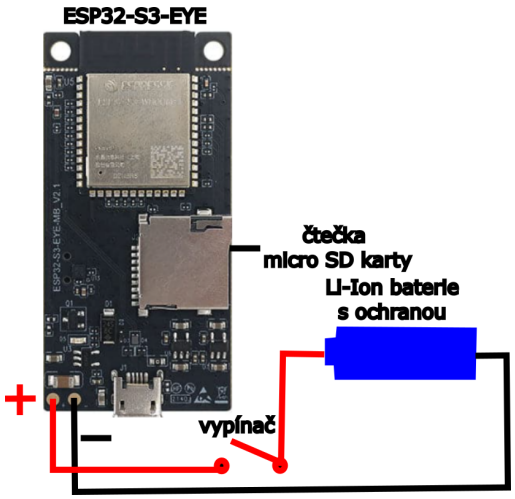
V této sekci bude popsáno zapojení obou zařízení.

Zařízení s ESP32-S3-EYE

U této vývojové desky nejsou vyvedeny volné piny, které by se daly použít k zapojení ostatních zařízení, proto se musí displej, který je s touto vývojovou deskou dodáván, odebrat. Díky tomu se zpřístupní deset GND (z ground – zem) pinů, pět vstupně-výstupních pinů, čtyři piny, které určují, jakým způsobem se zařízení zapne (tyto piny jsou nevhodné pro použití) a jeden napájecí 3,3 V pin. Napájení modulu Unit LoRaWAN EU868 se připojí k napájecímu 3,3 V pinu a GND se propojí s libovolným GND pinem na vývojové desce. Vývod TXD se připojí k pinu IO48 a vývod RXD zase k pinu IO47. Následuje připojení senzoru PIR AM312, výstup se připojí k pinu IO21, napájení k 3,3 V a GND na jakýkoliv GND pin. Výhoda této desky je, že má integrovanou nabíječku Li-Ion baterií, a proto ze zadní strany jsou plochy pro připájení přívodních kabelů od baterie. V dokumentaci je uvedeno, že použitá baterie musí být vybavena vlastním ochranným obvodem proti zkratu, podvybití a přebití [9]. Celé zapojení je vidět na obrázcích 4.5 a 4.6.



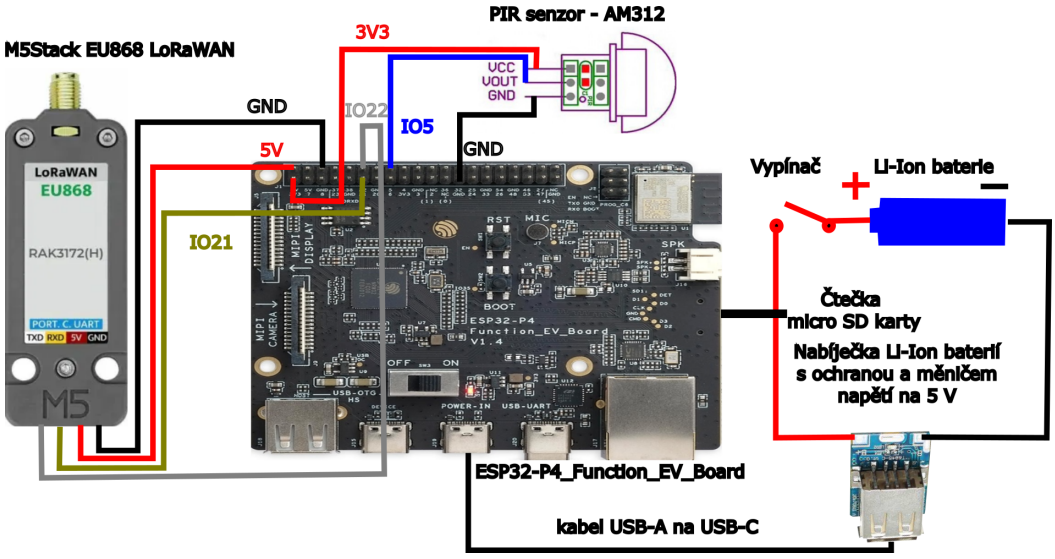
Obrázek 4.5: Zapojení zařízení z předního pohledu na vývojové desce ESP32-S3-EYE.



Obrázek 4.6: Zapojení zařízení ze zadního pohledu na vývojové desce ESP32-S3-EYE.

Zařízení s ESP32-P4_Function_EV_Board

Zapojení tohoto zařízení je značně jednodušší, jelikož piny, pro připojení ostatních modulů, jsou vyvedeny volně na desce a je jich dostatečné množství. Deska má vyvedených několik napájecích pinů, a to jak piny s 5 V, tak i 3,3 V. Proto můžeme napájení komunikačního modulu připojit na 5 V výstup a napájení senzoru AM312 k výstupu s 3,3 V. Piny GND můžeme připojit na libovolný výstup GND na desce. Výstupy rozhraní UART komunikačního modulu jsou připojeny následujícím způsobem. TXD je připojeno k výstupu IO21 a RXD je připojeno k IO22. Výstup senzoru AM312 je připojen k IO5. S připojením baterie je to tentokrát složité. Tato vývojová deska nemá jednoduchý způsob, jak by se dala připojit baterie. Je určená ke stálému napájení z konektoru USB-C. Navíc potřebuje být napájena 5 V. Řešení, které jsem navrhl, aby bylo možné napájet systém pouze z Li-Ion baterie, vyžaduje modul nabíječky Li-Ion baterií s ochranou a měničem napětí na 5 V. Poté výstup takového modulu propojíme s deskou pomocí kabelu USB. Na obrázku 4.7 je zapojení ukázáno.



Obrázek 4.7: Zapojení zařízení s vývojovou deskou ESP32-P4_Function_EV_Board.

4.7 Návrh aplikace pro zobrazování dat detekce

Pro účely vizualizace dat odesílaných pomocí koncových zařízení v LoRaWAN síti je potřeba navrhnout aplikaci. Tato aplikace se musí propojit se serverem The Things Network. Pro propojení aplikace se serverem The Things Network existuje několik možností.

Data lze přijímat přes protokol MQTT, který umožňuje odběr zpráv v reálném čase na základě odebírání konkrétního tématu. Téma (*topic*) v protokolu MQTT označuje komunikační kanál, kde se nachází zprávy určitého typu. V tomto případě by aplikace odebírala pouze zprávy uplink (zprávy přijaté od koncového zařízení). The Things Network

Další možností je využití HTTP Webhook, kdy server The Things Network při přijetí nové zprávy odešle požadavek na definovanou URL adresu aplikace. Umožní to jednoduché přijetí zprávy a případné uložení dat do databáze. Toto se vyplatí, pokud by aplikace byla hostovaná na internetu.

Pro jednoduchost jsem zvolil variantu, kdy vytvořím desktopovou aplikaci, implementovanou pomocí frameworku **ElectronJS**. Pro tento styl se více hodí propojení pomocí protokolu MQTT. Nevýhodou tohoto přístupu je, že může přijímat zprávy pouze, pokud je zrovna spuštěná a připojená k MQTT brokerovi (MQTT server sítě The Things Network). Aplikace bude pouze zobrazovat aktuálně přijaté zprávy a nebude je ukládat. Data bude ukládat do tabulky pod sebe, kde v jednom sloupci bude čas přijetí a v druhém dekódovaná zpráva podle formátu dat popsaném v sekci 4.4. Takto navrhnutá aplikace nemusí být nikde hostovaná na internetu a bude fungovat lokálně v počítači.

Aplikace umožní uživateli zadat přihlášení do MQTT brokera pro aplikaci vytvořenou v The Things Network. Po úspěšném přihlášení k brokerovi se aplikace přihlásí k odběru uplink zpráv a bude čekat do příjmu nové zpravy z koncového zařízení. Aplikace si bude pamatovat přihlašovací údaje k brokerovi i po vypnutí aplikace. To vyžaduje bezpečný uložení hesla. Při zapnutí aplikace si načte naposledy zadané přihlašovací údaje a pokusí se ihned připojit k MQTT brokerovi.

Kapitola 5

Implementace

V této kapitole popíšu postup implementace obslužného programu zařízení v sekci 5.1 a desktopové aplikace pro zobrazování přijatých dat v sekci 5.2. V sekci 5.3 se nachází realizace prototypů zařízení.

5.1 Obslužný program zařízení

Pro vývoj obslužného programu pro obě zařízení v této části bylo využito vývojové prostředí ESP-IDF v novější verzi 5.4. Starší verze nepodporují vývojové desky s čipem ESP32-P4. V textovém editoru Visual Studio Code jsem používal rozšíření ESP-IDF, které celé vývojové prostředí ESP-IDF stáhlo, nainstalovalo a nakonfigurovalo pro jednodušší programování a ovládání prostředí. Nainstaluje si automaticky i potřebné nástroje pro překlad, jako je program CMake a další. Celý projekt je přeložen v jazyce C++17.

Vlastní implementace programu je napsána v adresáři `main`. V adresáři `components` se nachází knihovny, které nejsou registrovány v systému ESP Component Registry. Takovéto knihovny musí být manuálně vloženy do projektu. Adresář `managed_components` se vygeneruje vývojovým prostředím při překladu nebo nastavování konfigurace. Obsahuje knihovny, které lze přidat do projektu pomocí ESP Component Registry. Automaticky se stáhnou všechny, které jsou zadány v souboru `idf_component.yml` v adresáři `main`.

Konfigurace projektu

Jelikož implementace kódu pro obě zařízení je napsaná v jednom projektu, je potřeba před překladem vybrat správné zařízení pro překlad. To se dá provést pomocí příkazu v systémové konzoli „idf.py set-target <target>“, kdy za „<target>“ se nahradí buď „esp32s3“, nebo „esp32p4“. Tímto se využijí přednastavené parametry uložené v souboru `sdkconfig.defaults.esp32s3` nebo `sdkconfig.defaults.esp32p4` podle zvoleného zařízení. Tyto soubory jsou nastaveny tak, aby povolily optimalizaci pro vyšší výkon. A zároveň další parametry pro správný chod aplikace. Aby vývojové prostředí mohlo využít přednastavených hodnot v souborech, musí být ještě vytvořen soubor `sdkconfig.defaults`. Do tohoto souboru se píšou parametry stejné pro všechna zařízení. V mé implementaci jsem ponechal tento soubor prázdný.

V mém projektu jsou ještě potřeba nastavit mé vlastní parametry, které vývojové prostředí předá mé aplikaci při překladu. Tyto parametry mohou být a jsou napsány v souboru `Kconfig` v adresáři `main`. Vývojové prostředí díky tomuto souboru automaticky přidá mou vlastní sekci do konfigurace. Sekci jsem pojmenoval „LoRaWAN Configuration“ a slouží pro

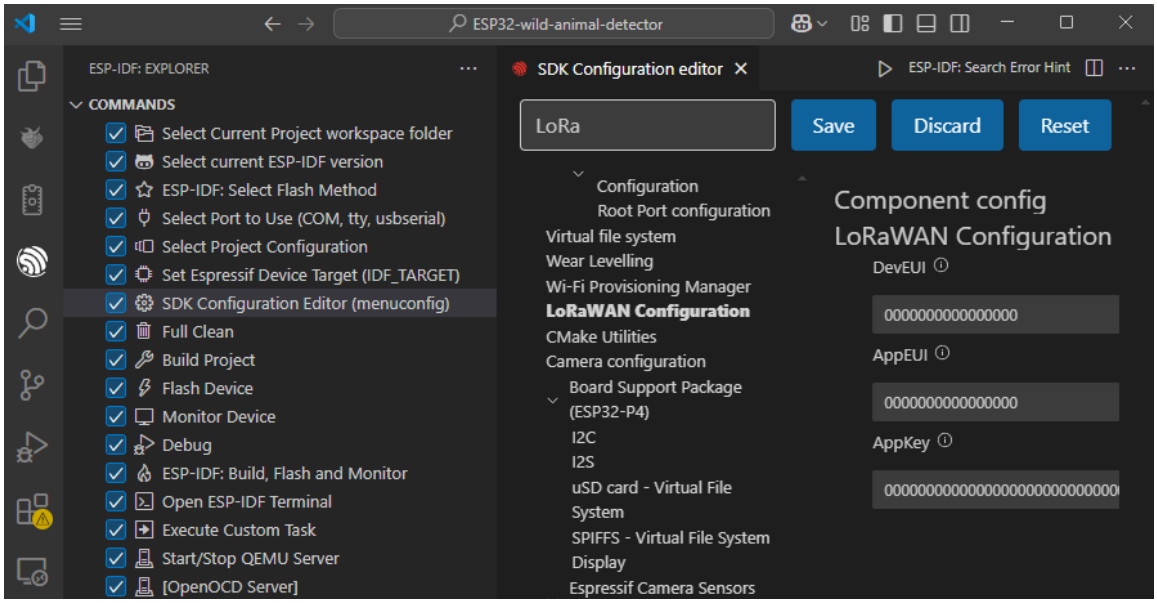
nastavení identifikátorů zapsaných hexadecimálně pro připojení do sítě LoRaWAN k serveru The Things Network.

V sekci se dají nastavit tyto parametry:

- **DevEUI** – 64bitový identifikátor zařízení v rámci LoRaWAN sítě.
- **AppEUI** – 64bitový identifikátor aplikace, ke který se zařízení připojuje.
- **AppKey** – 128bitový identifikátor, který je použit jako heslo ke konkrétní aplikaci.

Všechny parametry jsou dostupné při vytvoření aplikace a přidání koncového zařízení do ní na serveru The Things Network. **AppEUI** je možné také nalézt pod názvem **JoinEUI**. Všechny tyto hodnoty lze zvolit podle sebe, a kromě **JoinEUI** vám stránka dovolí vygenerovat nové identifikátory.

Nastavit tyto parametry lze v systémové konzoli pomocí příkazu „idf.py menuconfig“. Jednodušší a přehlednější se mi zdá využít funkce rozšíření ve Visual Studio Code, kde stačí kliknout na kolonku „SDK Configuration Editor (menuconfig)“. Na následujícím obrázku 5.1 je možné vidět otevřené konfigurační okno v programu Visual Studio Code.



Obrázek 5.1: Nastavování konfiguračního souboru v sekci „LoRaWAN Configuration“ v programu Visual Studio Code.

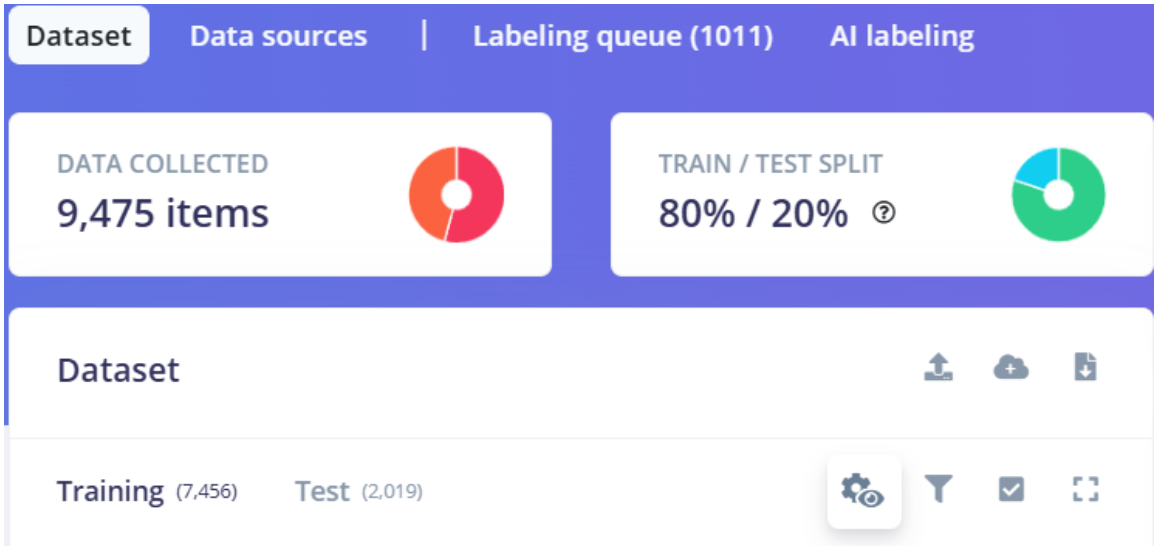
Model detekce lesní zvěře

Celý projekt je implementován tak, aby se použila platforma Edge Impulse a jejich SDK[1]. Základem pro vytvoření tohoto projektu byla šablona aplikace pro vývojové prostředí ESP-IDF. Šablona je volně dostupná v repozitáři na serveru GitHub a v dokumentaci [7] je napsáno, jak ji použít.

Po vytvoření projektu podle šablony se musí model vytvořit a natrénovat na fotografiích divokých prasat, jelenů a srnek. To se provádí již přímo na webových stránkách platformy Edge Impulse. Prvním krokem je vytvořit si projekt. Po vytvoření projektu musíme nahrát

[1]SDK – anglicky software development kit, je sada nástrojů pro vývoj aplikací pro určitou platformu

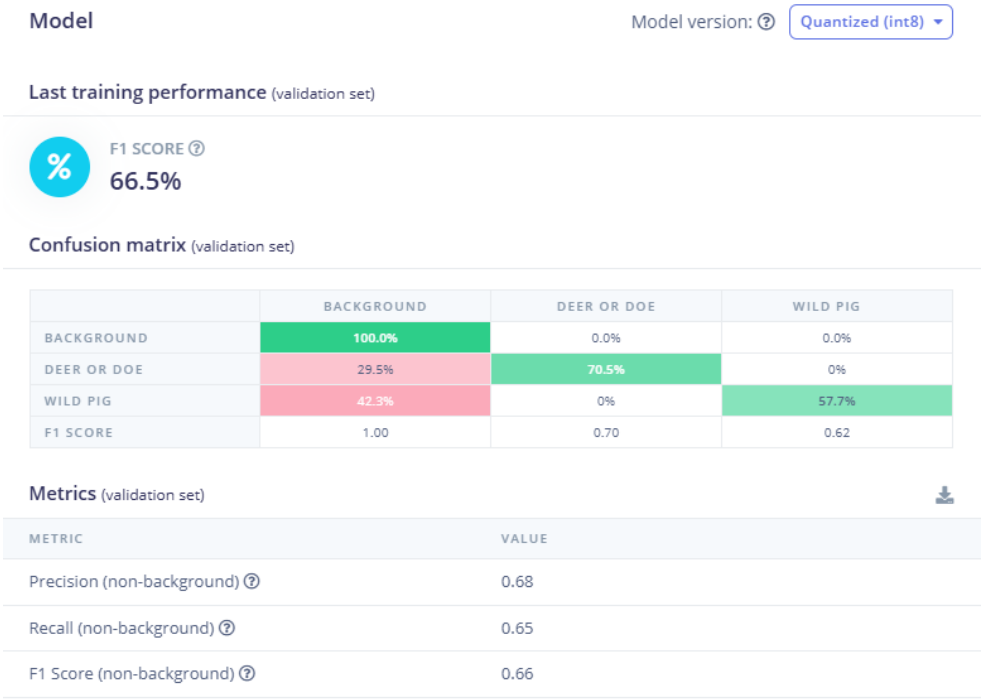
trénovací data na jejich server. První dataset obsahuje pouze fotografie lesa a jmenuje se **Forest**. Druhý dataset obsahuje fotografie různých zvířat zachycených pomocí fotopastí a jmenuje se **My Game Pics**. Oba datasety jsou popsány v sekci 3.5. Edge Impulse dovoluje anotovaný dataset se zvěří přímo importovat a nemusíme každou fotografii znovu anotovat ve webové aplikaci. Platforma si automaticky rozdělí fotografie na skupinu pro trénování a skupinu pro otestování, kde se snaží je rozdělit v poměru 80 % pro trénování a 20 % pro otestování . Po importování jsem musel ještě přejmenovat jednotlivé štítky určující zvířata, protože byly naimportovány jako čísla „0“, „1“. Následně jsem pomocí filtrování mezi fotografiemi vymazal zvířata, která jsem nechtěl rozpoznávat. Fotografie lesa jsem nechal bez štítků (*labels*). Celkem jsem použil pro trénování modelu 9 475 fotografií. Toto lze vidět včetně rozdělení dat na obrázku 5.2.



Obrázek 5.2: Nahrané a rozdělené fotografie na platformu Edge Impulse.

K testování se musí nejprve vytvořit nový „impuls“, kde nastavíme, jaké vstupní rozlišení bude model přijímat a jakým způsobem bude rozlišení obrázku měněno na vstupní rozlišení. V mém případě jsem zvolil velikost 96 na 96 pixelů, kde se obrázky budou zmenšovat bez zanechání poměru stran. Od tohoto rozlišení jsem si sliboval vysokou rychlost detekce a zároveň rychlejší trénování modelu při nižší velikosti modelu. Dále musíme přidat blok pro zpracování obrázku. To znamená, že vytvoří z obrázku „vlastnosti“, které model očekává jako vstup. Pro rychlejší trénink využiji toho, že bude model zpracovávat obrázky černobíle. Následuje samotné nastavení trénování modelu. Lze zde nastavit počet trénovacích cyklů (kolikrát se bude model učit z celé kolekce). Důležité nastavení je „learning rate“, které určuje, jak rychle se model učí z každého cyklu. Dále výběr, zda chceme nechat trénovat model na procesoru nebo na grafické kartě. Nastavit lze také, jaké procento z fotografií se použije pro validaci správnosti detekce modelu. Nastavit lze i hodnotu „batch size“. Tato hodnota určuje, kolik fotografií najednou vyhodnotí, než upraví váhy neuronové sítě. Jako detekční model jsem použil FOMO s páteří MobileNetV2. Po mnoha pokusech jsem dospěl k tomu, že nejlepší výsledky mi přineslo, když jsem nastavil 60 trénovacích cyklů, 0,001 „learning rate“ a 128 „batch size“. Trénoval jsem model na procesoru. K validaci jsem použil 20 % fotografií z trénovací kolekce. Výsledky trénování lze vidět na obrázku 5.3. Přesnost modelu se určuje pomocí hodnot **precision** a **recall**. **Precision** určuje, kolikrát se

model trefil, pokud tvrdil, že detekoval objekt. **Recall** určuje, kolik objektů úspěšně označil ze všech, které jsou v trénovacích datech anotovány. Platforma Edge Impulse ještě hodnotí **F1 score** a tato hodnota ukazuje poměr mezi **precision** a **recall** v procentech. Model, který jsem vytrénoval, má **precision** 68 % a **recall** 65 % a **F1 score** 66,5 %. Výsledky lze vidět na obrázku 5.3.



Obrázek 5.3: Výsledné hodnocení modelu pro detekci lesní zvěře.

Při testování na testovací kolekci fotografií však dosahoval model detekce lesní zvěře pouze **precision** 63 % a **recall** 64 %. Model si často plete pozadí se zvířetem, pokud není fotografie přímo v lese.

V posledním kroku se musí nasadit vytrénovaný model z webového prohlížeče do projektu. To se dělá v sekci „Deployment“. Stránka nabídne spoustu možností, jak model nasadit. Pro nasazení modelu do projektu s vývojovým prostředím ESP-IDF musíme zvolit možnost „C++ library“. Nyní se musí vybrat, zda chceme model zkompilovat pomocí kompilátoru EON, což sníží velikost celého modelu a sníží spotřebu paměti při jeho práci. Ještě lze vybrat, zda použít model používající 32bitové číslo s plovoucí řadovou čárkou nebo kvantizovaný model využívající 8bitová celá čísla. Zde je velký rozdíl v rychlosti a náročnosti na paměť. Kvantizovaný model je čtyřikrát menší, rychlost je téměř třikrát rychlejší a přesnost je téměř stejná. Z tohoto důvodu jsem využil kvantizovaný model. V původní verzi implementace jsem chtěl použít zkompilovanou verzi, avšak při následném spuštění na zařízení celý obslužný program přestal pracovat a ve chybovém výstupu bylo napsáno, že se nepodařilo alokovat dostatek paměti pro práci modelu i v momentě, kdy zařízení mělo volnou většinu paměti. Toto jsem se v první fázi implementace snažil vyřešit tím, že jsem záměrně dovolil modelu alokovat více paměti, to se však také nepovedlo. Nakonec jsem nechal vytvořit knihovnu s nezkompilovanou verzí modelu v souboru *tflite_learn_27.tflite*. To však vedlo k problémům s uložením souboru na zařízení a načtením do programu, jelikož vygenerovaná knihovna používá jinou knihovnu pro načtení modelu, která na zařízení ne-

funguje. Řešením je na webových stránkách v sekci „Dashboard“ v „Administrative zone“ zaškrtnout možnost „Use xxd instead of INCBIN to link TFLite/ONNX files“. Toto zapříčiní, že model bude uložen přímo v proměnné. Zkomprimovanou knihovnu jsem pouze extrahoval do projektu.

Komunikační modul

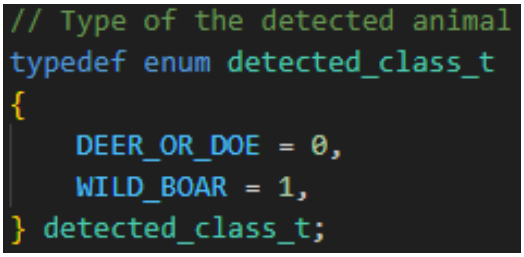
Ke komunikaci se sítí LoRaWAN jsem použil oficiální knihovnu k Unit LoRaWAN EU868. Umístěna je v adresáři `components` a pojmenovaná `M5-LoRaWAN-RAK`. Ta však je napsána pro framework Arduino a nelze ji použít ve vývojovém prostředí ESP-IDF. Musel jsem udělat implementaci funkcí a tříd z frameworku Arduino, které tato knihovna využívá. Jedná se konkrétně o třídy `HardwareSerial` a `String`. `HardwareSerial` se stará o komunikaci s modulem pomocí rozhraní UART. Třída `String` se stará o práci s textem a nemá stejně pojmenované metody jako třída `std::string` ze standardní knihovny jazyka C++. Adaptoval jsem knihovnu pro ESP-IDF tímto způsobem, aby byl co nejmenší zásah do vlastního kódu knihovny. Jediná změna provedená na samotné knihovně je změna názvu souboru načítaného direktivou `#include` z `Arduino.h` na `ArduinoAdapter.h`, kde je má implementace uložena.

Knihovna používá takzvané „AT příkazy“. To jsou textové příkazy určené k ovládání komunikačního modulu. Při odeslání jakéhokoliv příkazu modul odpovídá zpět také v textové formě. Knihovna zasílání příkazů a příjem odpovědí od modulu značně zjednodušuje.

V adresáři `lorawan` se nachází následující soubory:

- **lorawan.hpp** a **lorawan.cpp** – Obsahují funkce pro inicializaci LoRaWAN modulu, připojení k síti a zaslání zprávy. Inicializace spoléhá na správné zapojení modulu, které je popsáno v sekci 4.6. Stav připojení k síti je uložen v globální proměnné `lorawan_joined`. Tato proměnná je uložená do paměti RTC. To umožňuje udržet aktuálně nastavenou hodnotu do odpojení od napájení. Při inicializaci se spustí na pozadí proces, který se zpracovává asynchronně a kontroluje, zda komunikační modul nezaslal zprávu. Konkrétně zda se úspěšně připojil k síti nebo nikoliv. Tyto zprávy automaticky nastaví hodnotu globální proměnné `lorawan_joined`. Modul se pokusí o připojení maximálně desetkrát každých 7 sekund. Zařízení je v síti nastaveno jako třída A.

- **sender.hpp** a **sender.cpp** – Obsahují funkci, která zjistí, zda byla detekována zvěř z výsledku detekce. Dále funkci, která výsledek detekce zformátuje podle popisu ze sekce 4.4 a pokusí se jí odeslat. Poslední funkce se stará o to, aby se pokusila zaslat zprávu, případně znovu připojit k síti. Připojení k síti je omezeno maximálně na 70 sekund a poté se program přestane pokoušet o zaslání a zprávu nepošle. V hlavičkovém souboru definuje výčtový datový typ, který slouží pro označení zvířat číslem podle formátu pro zaslání do sítě. Ten lze vidět na obrázku 5.4.



Obrázek 5.4: Výčtový datový typ popisující detekované zvíře číslem.

Kamera

Implementace kódu používajícího kameru je uložena v adresáři `camera`. Je rozdělena do souborů následovně.

Soubor **photo_trap_camera.hpp** obsahuje deklarace funkcí, které slouží k inicializaci a ukončení kamery, alokaci a uvolnění paměti a uložení fotografie do paměti. Dále obsahuje deklarace globálních proměnných, do kterých se fotografie ukládá a jejich velikost v bajtech. Do proměnné `image_buffer` je ukládána fotografie v původní velikosti a do `image_detection_buffer` se obrázek ukládá zmenšený na vstup do detekčního modelu.

V souboru **camera_common.cpp** se nachází implementace funkcí alokujících paměť pro ukládání fotografií.

Implementace funkcí pro využívání kamery na zařízení ESP32-S3-EYE je v souboru **camera_s3.cpp**. K implementaci používám knihovny `esp32_camera`, která podporuje zařízení s čipy ESP32-S3 a kamerou OV2640. Tato knihovna je přidána pomocí ESP Component Registry. V ukázkových projektech od firmy Edge Impulse se právě tato knihovna používá, a proto při implementaci k obsluze jsem použil částečně jejich kód. Ve funkci pro uložení obrázků jsem upravil pouze to, aby se ukládala jak originální fotografie, tak upravená pro vstup do modelu. Originální fotografii ukládá ve formátu JPEG s rozlišením 1280 na 720 pixelů. Upravená fotografie je převedena na formát RGB888 a zmenšena na rozlišení 96 na 96 pixelů. Poslední je zde funkce, která je vyžadována Edge Impulse SDK a převádí upravenou fotografii do formátu 32bitového čísla s plovoucí řadovou čárkou. Tuto funkci jsem také použil z ukázkového projektu. Byl zde vyřešen problém knihovny `esp32_camera`, který při konverzi fotografie z formátu JPEG do formátu RGB888 ukládal barevné kanály v pořadí modrá–zelená–červená (BGR) namísto očekávaného pořadí červená–zelená–modrá (RGB).

Posledním souborem je **camera_p4.cpp** pro zařízení s čipem ESP32-P4. Zde byl problém s kompatibilitou knihovny `esp32_camera`, která nepodporuje tento čip a ani samotnou kameru SC2336. Pokusil jsem se ji nahradit knihovnou `esp-video` přidanou pomocí ESP Component Registry a nadstavbou této knihovny knihovnou `who_cam`, která je součástí frameworku ESP WHO. Tuto třídu jsem musel přidat do projektu manuálně do adresáře `components` pod názvem `who_cam`. Změna oproti implementaci na předchozí zařízení je, že kamera zaznamenává obraz ve formátu RGB888, a tak se rovnou může zmenšit na požadovanou velikost. Jelikož není využit převod formátů z knihovny `esp32_camera`, tak jsem upravil načítání barevných kanálů zpět na pořadí červená–zelená–modrá.

Problém se zařízením s čipem ESP32-P4

Při zkoušení implementace na zařízení s ESP32-P4 jsem zjistil, že obslužný program nikdy nic nedetekoval. To bylo zvláštní, jelikož implementace pro ESP32-S3 fungovala. Při zkou-

šení ukázkových projektů jsem vyzkoušel jeden, který umožňuje připojit zařízení k počítači a používat v něm jeho kameru. Zjistil jsem, že výstup kamery byl příliš přesvícený, až téměř bílý, a použití v přírodě nepřipadalo v úvahu. Po mnoha pokusech nastavení kamery se mi nepodařilo naleznout řešení s tímto problémem. Tento problém mi zamezil testování tohoto zařízení a je třeba buď využít jiné kamery, nebo jiných knihoven, které jsem nenalezl. Dalším problémem byla i nekompatibilita knihovny `esp_video` a knihovny spravující inicializaci micro SD karty, kdy nelze inicializovat micro SD kartu. Z těchto důvodů testování na tomto zařízení probíhalo pouze na fotografiích nahraných do paměti.

Ukládání fotografií na paměťové médium

Poslední částí před implementací samotného smyčky obslužného programu je zápis pořízené fotografie na micro SD kartu. K tomuto používám knihovnu `esp32_s3_eye_noglib` na ESP32-S3 nebo obdobnou knihovnu `esp32_p4_function_ev_board_noglib` pro ESP32-P4. Tento typ knihoven umožňuje používání jednotlivých komponent stejnými funkcemi pro každou podporovanou vývojovou desku. V tomto případě obě zařízení mají čtečku micro SD karet a mohu pro obě zařízení použít stejný kód a rozdíly mezi nimi vyřeší knihovna. Tento typ knihoven se nazývá Board Support Package. Do projektu jsou přidány pomocí ESP Component Registry.

Implementace se nachází v adresáři `sdcard` v souborech **sdcard.hpp** a **sdcard.cpp**. V souboru **sdcard.hpp** se nachází deklarace funkce, která inicializuje souborový systém na micro SD kartě a uloží obrázek. Dále ještě deklaruje binární semafor, který bude sloužit pro synchronizaci procesů obslužného programu. Soubor **sdcard.cpp** obsahuje implementaci této funkce. Nejdříve inicializuje micro SD kartu. Poté využije pomocné funkce. Pomocné funkce jsou určeny pro práci s NVS[2]. Konkrétně funkce, které NVS inicializuje, přečte hodnotu čítače, inkrementuje hodnotu čítače a ukončí NVS. Čítač se používá pro pojmenování nového souboru s fotografií. Pro zařízení s ESP32-P4 je potřeba ještě fotografii překonvertovat z formátu RGB888 na formát JPEG. Následně se pokusí zapsat fotografii do souboru pojmenovaného číslovkou s aktuální hodnotou čítače a při úspěšném zapsání inkrementuje hodnotu čítače. Na konci odpojí micro SD kartu, ukončí NVS a nastaví hodnotu binárního semaforu, aby se mohl obslužný program ukončit. Při nevložené micro SD kartě nebo chybné inicializaci nastaví hodnotu binárního semaforu, aby obslužný program mohl také ukončit.

Hlavní logika programu

V souboru **main.cpp** se nachází hlavní logika obslužného programu. Zde se využívají implementace komponent popsané v předchozích sekcích. Program je implementován podle návrhu, který je na obrázku 4.4. Je určen, aby většinu času byl uveden do režimu hlubokého spánku (deep sleep). Je to energeticky nejúspornější režim, kdy se ukončí skoro všechny periferie vývojové desky a čipu. Na začátku programu se proto musí nastavit, jak se bude zařízení probouzet, jinak by nebylo možné již zařízení probudit. Probuzení je implementováno na nástupní hranu vstupu, kde je připojen senzor PIR. Aby se více šetřila energie, zkontroluje se na začátku, zda poslední úspěšná detekce proběhla před minutou. To znamená, že první reálná detekce může být provedena po minutě. To umožňuje umístit zařízení na požadované místo, zapnout ho a odejít dříve, než se spustí první detekce. Je to možné díky proměnné `last_detection_time`, která je uložena v paměti RTC a uchovává

[2]NVS – non-volatile storage (úložiště, kde uložená data přetrvají i po odpojení zařízení od napájení)

si svou hodnotu ve spánku zařízení. Při prvním spuštění nastaví aktuální čas do této proměnné a kontroluje rozdíl s hodnotou času, která je naměřena při každém spuštění zařízení. Tento rozdíl musí být 60 sekund a více. Do proměnné `last_inference_time` se zapíše nový čas pouze, pokud byla úspěšná detekce později v programu.

Při splnění časové podmínky začne vykonávání funkčního kódu. Při každé chybě, která ovlivní fungování zařízení, jako je neúspěšná inicializace některé z komponent, tak přejde zařízení zpět do režimu spánku. První se inicializuje kamera. Zde při testování jsem zjistil, že výstup kamery po inicializaci byl celý nazelenalý. Pomohlo přidat čas po inicializaci kamery, kdy se čeká, než kamera na pozadí provede korekci barev a poté teprve pořídí fotografii. Tento čas jsem nastavil na 750 ms. Zlepšilo to jak úspěšnost detekce, tak i vzhled fotografie, která se uloží na micro SD kartu. Dále program alokuje paměť pro uložení fotografie.

Přichází část kódu, kde se již pořídí fotografie samotná. Ta je uložena jak v původním rozlišení, tak i ve velikosti a formátu fotografie pro detekci. Po pořízení fotografie je kamera ukončena, aby se snížila spotřeba energie. Před spuštěním modelu detekce se musí vytvořit struktura `signal_t`. Zde musí být nastavena celková velikost fotografie. Ta se nastaví na počet pixelů fotografie. To je $96 \times 96 = 9216$ pixelů. Do struktury se musí nastavit funkce, která je volána implementací detektoru modelu. Jedná se o funkci, která převádí fotografii na požadovaný formát v detektoru. Tato funkce je pojmenována `camera_get_data` a je implementována v komponentě kamery, která je popsána v sekci 5.1. Poté se předá tato struktura detektoru, který provede detekci a výsledek zapíše do struktury `ei_impulse_result_t` do proměnné `result`. Model se spustí funkcí `run_classifier`.

Výsledek detekce je zkontrolován, zda obsahuje nějaké zvíře, a pokud ano, tak se provede zapsání času poslední úspěšné detekce, jak bylo napsáno výše. Dale se inicializuje komunikační modul. Díky systému FreeRTOS lze spustit proces, který uloží fotografii na micro SD kartu, paralelně s procesem pro odeslání dat na server The Things Network. Konkrétně odeslání dat je spuštěno ve hlavním procesu a pro zapsání na micro SD kartu je vytvořen nový proces. Aby se nestalo, že by hlavní proces dokončil zasílání dříve, než by skončil proces pro zapsání, tak hlavní proces musí počkat, než proces pro zapsání uvolní binární semafor. Binární semafor se musí uvolnit i v případě, že nebylo detekováno žádné zvíře. Po dokončení obou procesů se uvolní paměť obrázků a zařízení přejde do režimu hlubokého spánku.

5.2 Aplikace pro zobrazování dat detekce

Aplikace je implementována pomocí frameworku **ElectronJS**. Tento framework umožňuje vyvíjet desktopové aplikace podobným stylem, jako jsou vyvíjeny webové aplikace. Využívá technologii **Node.js** pro backendovou logiku a **Chromium** pro zobrazování uživatelského rozhraní, což umožňuje použití značkovacího jazyka HTML, CSS pro nastavení vzhledu elementů a jazyka JavaScriptu. Výhodou tohoto přístupu je multiplatformní kompatibilita, díky které lze výslednou aplikaci spustit na operačních systémech Windows, Linux i macOS bez nutnosti zásadních úprav kódu. ElectronJS zároveň poskytuje přístup k systémovým prostředkům, jako je souborový systém, čímž rozšiřuje možnosti běžných webových technologií pro použití v prostředí klasických desktopových aplikací. Výhodou také je, že umožňuje použití různých balíčků. Balíčky jsou obdoba knihoven v jiných jazycích. Pro správu, instalaci, aktualizaci balíčku je v **Node.js** nástroj **npm**. Pro vyhledávání balíčků má nástroj **npm** webovou stránku[3].

[3] Vyhledávání balíčků v nástroji **npm** je dostupné na webové stránce: https://www.npmjs.com/

Struktura aplikace

V projektu jsou klíčové tři soubory `main.js`, `preload.js` a `renderer.js`, které společně tvoří jádro aplikace. Soubor `main.js` spouští aplikaci a podle souboru `index.html` vytvoří uživatelské prostředí a soubor `index.css` jeho vzhled. Má zároveň přístup k systému. Soubor `preload.js` vytváří komunikační můstek mezi hlavním procesem v `main.js` a souborem `renderer.js`. Zde se nachází většinou kód, který komunikuje se systémovými prostředky. V mé implementaci tento soubor používám pro načtení balíčků a používání balíčků. Soubor `renderer.js` přímo interaguje se zobrazovaným uživatelským rozhraním a má přístup ke zdrojům, které jsou mu v souboru `preload.js` pomocí `contextBridge` zpřístupněny. Toto rozdělení implementace do tří souborů je ve frameworku **ElectronJS** standardní a pomáhá to aplikaci s bezpečností, jelikož oddělí zobrazovací proces od procesu, který má přístup k systémovým prostředkům.

Implementace aplikace

K stahování dat ze serveru The Things Network využívá aplikace MQTT klienta z balíčku **mqtt**[4]. Tento balíček umožňuje jednoduchou komunikaci s MQTT brokerem. Tento balíček je načten v souboru `preload.js`. Pomocí `contextBridge` vytvoří objekt `mqttBridge`, který je dostupný v souboru `renderer.js`. Objekt mqttBridge má metodu `connect`, která vytvoří nového MQTT klienta a pokusí se připojit k MQTT brokerovi. Tato metoda nastaví funkce, které jsou zavolány při připojení k brokerovi, při chybě MQTT klienta, při odpojení od brokera a pro příjem nové zprávy. Při úspěšném připojení se přihlásí klient automaticky k odběru tématu, do kterého se zasílají uplink zprávy (zprávy odesílané koncovými zařízeními), které jsou v rámci jedné aplikace na serveru The Things Network. Téma vypadá následovně „v3/photo-trap@ttn/devices/+/up“. Část „photo-trap@ttn“ je přihlašovací jméno k MQTT brokerovi a „photo-trap“ reprezentuje jedinečný identifikátor aplikace. Takže je díky přihlašovacímu formuláři možné se automaticky přihlásit i k odběru zpráv bez dalšího zadávání tématu uživatelem. Všechny funkce, které jsou volány zpětně MQTT klientem, vytváří událost, která může být přijata v souboru `renderer.js`. Při přijetí nové zprávy se zpráva ve formátu JSON deserializuje a odešle se událostí pouze čas, kdy byla zpráva přijata na server The Things Network, a samotný text zprávy, který je kódován v Base64.

Při připojování k brokerovi nelze znovu spustit proces připojování a v uživatelském rozhraní lze vidět zašedlé tlačítko „Connect“. Připojování je omezeno na maximálně 10 sekund. Poté se lze znovu pokusit o připojení.

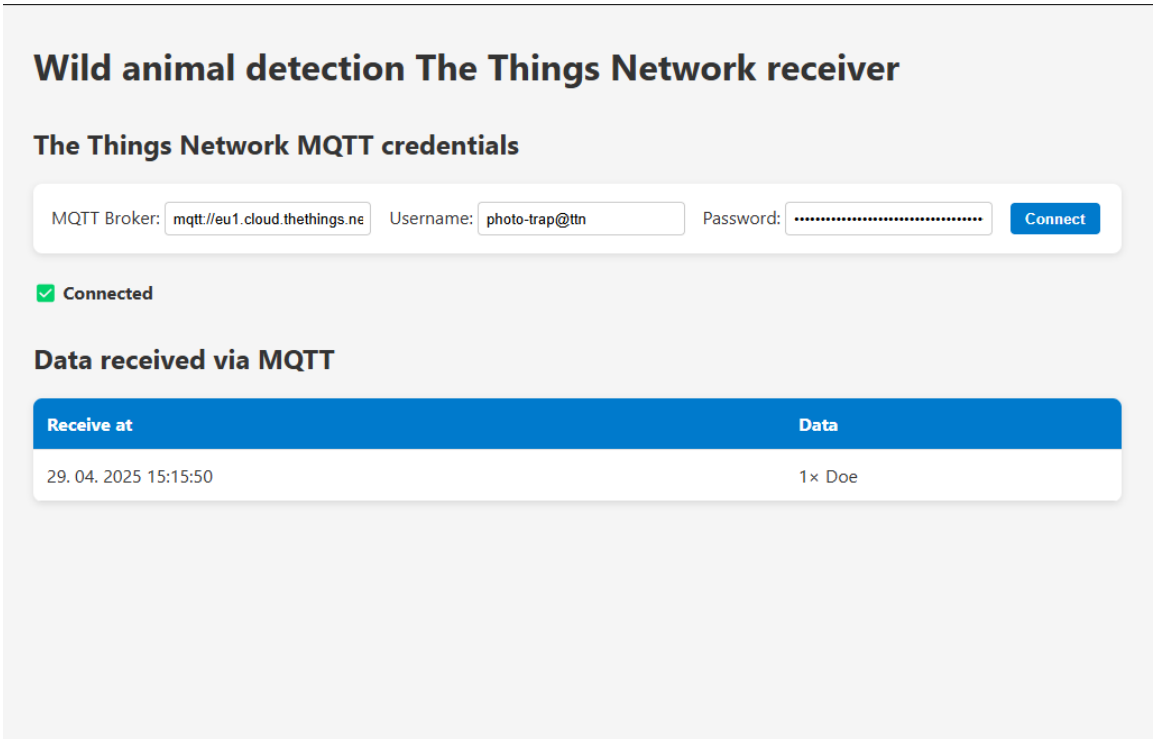
V souboru `preload.js` aplikace ještě využívá balíčku **keytar**[5]. Umožňuje také přes `contextBridge` ukládat bezpečně hesla. Objekt, který je dostupný ve vykreslovacím procesu, je pojmenován `passwordStore`. Zde implementuje metody pro uložení, načtení a smazání hesla.

V souboru `renderer.js` se implementuje metoda `window.onload`, která je zavolána po startu aplikace. Zde se pokusí z pomocí `localStorage` a `passwordStore` načíst poslední zadané přihlašovací údaje. Pokud jsou načteny, tak se pokusí automaticky připojit k MQTT brokerovi. Dále nastaví funkci, která se zavolá po zadání nových přihlašovacích údajů do formuláře. Ta se pokusí připojit k odběru zpráv a uloží nové údaje pomocí `localStorage` a `passwordStore`.

[4] https://www.npmjs.com/package/mqtt
[5] https://www.npmjs.com/package/keytar

Soubor `renderer.js` se přihlásí k odběru MQTT událostí a podle typu událostí mění text stavu připojení aplikace. Při příjmu zprávy zavolá funkci, která převede zprávu kódovanou v Base64 na bajtové pole. Podle tohoto pole vytvoří podle formátu ze sekce 4.4 textový řetězec. Text na výstupu může vypadat například takto: „2× Doe, 3× Wild Boar“. Ještě, než je tento text zobrazen, tak se převede přijatý čas na český formát. Následně se text a čas přidá do tabulky v uživatelském rozhraní.

Vzhled a rozvržení uživatelského rozhraní lze vidět na obrázku 5.5.

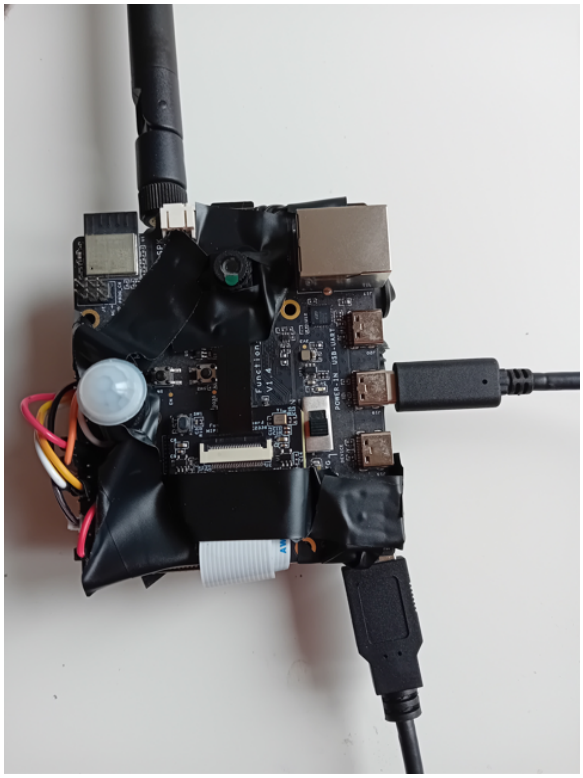


Obrázek 5.5: Na obrázku je vidět uživatelské rozhraní aplikace pro příjem dat ze serveru The Things Network. V tabulce je vidět přijatá zpráva z testování na obrázku srnky.

5.3 Realizace zařízení

Vytvořeny byly dva prototypy zařízení, které jsou určeny pro detekci lesní zvěře, za účelem je spolu porovnat. Avšak nepodařilo se vyřešit problém s kamerou na zařízení s čipem ESP32-P4. Proto bylo otestováno v reálném prostředí pouze zařízení s čipem ESP32-S3. Prototyp zařízení s čipem ESP32-P4 byl sestaven dříve, než se na problém s kamerou přišlo. Zařízení s čipem ESP32-P4 jsem otestoval pouze na rychlost zpracování fotografie, která je nahrána přímo do paměti zařízení, což nevyžaduje funkčnost kamery. Vzhledem k menší velikosti se zdá varianta s čipem ESP32-S3 jako lepší varianta do lesa. Na obrázcích 5.6 a 5.7 můžete vidět obě zařízení.

Obrázek 5.6: Prototyp s vývojovou deskou ESP32-S3-EYE.



Obrázek 5.7: Prototyp s vývojovou deskou ESP32-P4-Function-EV-Board.

Kapitola 6

Testování a experimenty

V sekcích 6.1 a 6.2 jsou popsány testy. Sekce 6.3 obsahuje naměřené metriky. V sekci 6.4 je analyzována energetická spotřeba zařízeních.

6.1 Testování na obrázcích lesní zvěře

Při tomto testování jsem musel pro zařízení s čipem ESP-P4 nahrávat obrázky přímo do paměti zařízení. Na zařízení s čipem ESP-S3 jsem vyzkoušel funkcionalitu včetně kamery. Prototypům jsem předkládal obrázky, kde se nacházela zvěř a chtěl jsem otestovat, zda funguje zasílání a ukládání fotografií.

Zařízení s čipem ESP-S3 bylo schopné z obrázků detekovat zvířata. Mělo však problém s detekcí objektů na pozadí. Model byl vytrénován na fotografiích lesa a zvěře. Proto často detekoval běžné objekty interiéru domu jako srnku nebo divoké prase. Dále mělo zařízení problém detekovat správný počet zvěře, pokud byla blízko u sebe. Toto může být způsobeno také menší přesností modelu.

Zařízení s čipem ESP-P4 na obrázcích předané do paměti dokázal detekovat zvěř stejně jako druhé zařízení. Toto potvrdilo problém s kamerovým modulem. Aby bylo testování zařízení vypovídající ponechal jsem inicializaci kamery a pořízení fotografie v implementaci. Jak bylo již u implementačních problémů napsáno, tak zařízení kvůli nekompatibilitě kamerové knihovny a knihovny ovládající čtečku micro SD karet nedokázalo obrázek uložit na paměťové médium.

6.2 Testování v reálným prostředí

Testování bylo prováděno po dobu 5 dní v lese přibližně 1,5 km od LoRaWAN brány. Na obrázku 6.1 lze vidět lokaci testování poblíž obce Čechtice.

Obrázek 6.1: Lokace testování zařízení nedaleko obce Čechtice.

Nejprve bylo zařízení umístěno na mýtině. To se však neosvědčilo jako dobré místo, protože se zařízení neustále probouzelo pomocí senzoru PIR, který byl moc citlivý a při vyšší rychlosti větru ho spínal pohybující se travní porost. Navíc po spuštění zařízení senzorem občas zařízení detekovalo zvíře, které na fotografii nebylo. Na obrázku 6.2 je vidět mýtina, kterou zařízení sledovalo.

Po tomto bylo zařízení umístěno do hustějšího jehličnatého lesa. Strom, který byl vybrán k umístění zařízení, byl před místem, kde vypadalo, že se dříve vyskytovala divoká prasata. Na obrázku 6.3 je přidělané zařízení ke stromu v lese.

Obrázek 6.2: Obrázek z pohledu kamery zařízení na mýtině.

Obrázek 6.3: Umístění zařízení v lese na stromě.

Během testování muselo být zařízení opraveno, jelikož došlo k uvolnění kontaktu ke komunikačnímu modulu, byly všechny kontakty připájeny za použití cínu a páječky. Po opravě stále žádná zpráva o detekci zvěře nepřicházela, tak jsem v rámci testování nasypal krmení před zařízení. Konkrétně siláž a směs obilí, kukuřice, pšenice a ječmene.

Po dvou dnech v brzkých hodinách kolem půl 6 ráno přišla zpráva na server The Things Network a do aplikace. V rozmezí 5 minut přišly další dvě zprávy. Zařízení jsem se rozhodl ponechat ještě do večera, ale žádná zpráva již nepřišla. Další den měl přijít déšť, takže jsem zařízení musel odnést pryč, aby se nepoškodilo. Z fotografií uložených na micro SD kartě

byla vidět dvě divoká prasata. Na obrázku 6.4 je vidět fotografie pořízená zařízením po detekci divokých prasat. Fotografie byla hodně tmavá, proto jsem fotografii zvýšil jas.

Obrázek 6.4: Fotografie pořízena zařízením po detekci dvou divokých prasat.

V lese se zařízení neprobouzelo tak často a nedocházelo k častým falešným detekcím. Tento test dokázal, že zařízení je schopné detekovat divoká prasata. Bohužel se nepodařilo detekovat žádnou srnku. To může být tím, že se v této oblasti nepohybovala nebo je zařízení nerozpoznalo.

6.3 Metriky

Při testování byl pozorován čas, jak dlouho trvali jednotlivé akce na zařízení. Konkrétně rychlost inicializace kamery, čas pořízení fotografie, rychlost zpracování pořízené fotografie modelem detekce a celkový potřebný čas od probuzení po detekci zvěře. Tyto časy určí, jak rychle dokáže zařízení pracovat. Čas odeslání zprávy do sítě LoRaWAN není tolik důležitý, jelikož se může lišit podle počtu pokusů, než se komunikační modul připojí do sítě a úspěšně odešle zprávu. Po inicializaci je kvůli zlepšení kvality pořízené fotografie zařízení opožděno o 750 ms. Tyto naměřené časy obou zařízeních jsou zapsané v tabulce 6.1.

Akce	**ESP32-S3 [ms]**	**ESP32-P4 [ms]**
Inicializace kamery	188	197
Pořízení fotografie	724	642
Čas po pořízení fotografie	1 654	1 586
Provedení detekce zvěře	137	194
Celkový naměřený čas po detekci	**1 809**	**1 812**

Tabulka 6.1: Porovnání časů jednotlivých segmentů v milisekundách mezi zařízeními s čipem ESP32-S3 a zařízení s čipem ESP32-P4.

Tyto naměřené hodnoty jsou překvapivé, jelikož byl předpoklad, že zařízení s čipem ESP32-P4 bude výrazně rychlejší jak zařízení s čipem ESP32-S3. Může to být tím, že Edge Impulse SDK není optimalizované pro toto zařízení.

Obě zařízení si vedla z časového hlediska podobně a detekce zvěře je dokončena 1,8 sekundy od probuzení zařízení z režimu hlubokého spánku.

6.4 Energetická spotřeba

Energetická spotřeba byla vyhodnocena pro obě cílová zařízení, tedy pro variantu s čipem ESP32-S3 i pro variantu s čipem ESP32-P4. Spotřeba byla analyzována s ohledem na jednotlivé fáze provozu zařízení při scénáři, kdy po probuzení dojde k úspěšné detekci objektu. V obou případech se předpokládá, že připojení k síti LoRaWAN neproběhne napoprvé, a proto jsou v úvahu zahrnuty dva pokusy o připojení. Zápis dat na SD kartu není v tabulkách samostatně uveden, jelikož probíhá paralelně s komunikací a při měření nebyl detekován významný nárůst odběru. V tabulkách 6.2 a 6.3 je zaznamenána spotřeba zařízení v jednotlivých fázích a vypočítána celková spotřeba za jeden detekční cyklus.

Fáze	**Proud [mA]**	**Doba [s]**	**Spotřeba [mAs]**
Probuzení a inicializace	60	0,93	55,8
Pořízení fotografie a detekce	110	0,88	96,8
Připojení (2 pokusy)	95	14,0	1 330
Špičky při připojení	170	0,6	102
Odeslání zprávy	170	0,3	51
Celkem	–	16,71	**1 635,6**

Tabulka 6.2: Spotřeba energie zařízení s čipem ESP32-S3 dle jednotlivých fází provozu.

Fáze	**Proud [mA]**	**Doba [s]**	**Spotřeba [mAs]**
Probuzení a inicializace	120	0,94	112,8
Pořízení fotografie a detekce	160	0,86	137,6
Připojení (2 pokusy)	130	14,0	1 820
Špičky při připojení	180	0,6	108
Odeslání zprávy	180	,3	54
Celkem	–	16,7	**2 232,4**

Tabulka 6.3: Spotřeba energie zařízení s čipem ESP32-P4 dle jednotlivých fází provozu.

Celková spotřeba energie za jeden detekční cyklus se přepočítá na jednotky mAh v rovnicích 6.1 a 6.2.

$$\frac{1635{,}6}{3600} \approx 0{,}454\,\text{mAh} \text{ (ESP32-S3)} \tag{6.1}$$

$$\frac{2232{,}4}{3600} \approx 0{,}620\,\text{mAh} \text{ (ESP32-P4)} \tag{6.2}$$

Za předpokladu pěti detekčních cyklů denně je denní spotřeba v aktivní fázi vypočítána podle rovnice 6.3.

$$\text{ESP32-S3: } 5 \times 0,454 = 2,27\,\text{mAh}, \quad \text{ESP32-P4: } 5 \times 0,620 = 3,10\,\text{mAh} \tag{6.3}$$

Doba trvání všech pěti aktivních cyklů je vypočítána v rovnici 6.4.

$$\text{ESP32-S3: } 5 \times 16,71 = 83,55\,\text{s}, \quad \text{ESP32-P4: } 5 \times 16,7 = 83,5\,\text{s} \tag{6.4}$$

Čas strávený v aktivním cyklu je podobný pro obě zařízení a čas zbytku dne se vypočítá podle rovnice 6.5.

$$\frac{86400 - 83,5}{3600} \approx 23,98\,\text{hodiny} \tag{6.5}$$

Zařízení s čipem ESP32-S3 má spotřebu 18 mA v režimu hlubokého spánku, kde zařízení s čipem ESP32-P4 má v tomto režimu 87 mA. Z toho se vypočítá denní spotřeba ve spánku pro zařízení s čipem ESP32-S3 v 6.6 a 6.7.

$$\text{ESP32-S3: } 23,98 \times 18 = 431,64\,\text{mAh} \tag{6.6}$$

$$\text{ESP32-P4: } 23{,}98 \times 87 = 2086,26\,\text{mAh} \tag{6.7}$$

Celková denní spotřeba je vypočítána v 6.8.

$$\text{ESP32-S3: } 431,64 + 2,27 = \mathbf{433{,}91\,mAh}, \quad \text{ESP32-P4: } 2086,26 + 3,10 = \mathbf{2089{,}36\,mAh} \tag{6.8}$$

Zařízení je napájeno akumulátorem s kapacitou 3 200 mAh. Výdrž zařízení při tomto režimu provozu je vypočtena v 6.9.

$$\text{ESP32-S3: } \frac{3200}{433,91} \approx \mathbf{7{,}37\,dne}, \quad \text{ESP32-P4: } \frac{3200}{2089,36} \approx \mathbf{1{,}53\,dne} \tag{6.9}$$

Z výpočtu je patrné, že zařízení s čipem ESP32-S3 je z hlediska spotřeby výrazně úspornější, především díky nízkému odběru v režimu hlubokého spánku. Naopak zařízení ESP32-P4 má v režimu hlubokého spánku podstatně vyšší spotřebu, což se zásadně odráží na výdrži při napájení z baterie. Zařízení s čipem ESP32-P4 je nevhodné pro provoz na baterii.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo navrhnout, implementovat a otestovat nízkoenergetické zařízení schopné detekce lesní zvěře s využitím mikrokontroléru ESP32 a bezdrátové komunikace pomocí technologie LoRaWAN do sítě The Things Network. Ačkoliv v návrhu bylo naplánováno vytvořit dva funkční prototypy, kvůli implementačním problémům na jednom ze zařízení nakonec funguje pouze jeden prototyp. Tento prototyp splňuje záměr této práce. Dalším cílem byla implementace aplikace, která by data zaslaná do sítě stahovala a zobrazovala je. Tento cíl byl také splněn.

Při testování bylo zjištěno že zařízení s čipem ESP32-P4 není vhodné pro bateriový provoz. Překvapivě toto zařízení nebylo v testování rychlejší než druhý prototyp i přesto, že má mnohem větší výpočetní výkon.

Zařízení s čipem ESP32-S3 je schopné provozu a myslím si, že po vylepšení modelu detekce by mohlo být použito k monitorování lesní zvěře. Tímto vylepšením je myšleno naučit model i na jiné prostředí, než je les. Užitečné by bylo naučit model rozpoznávat zvěř na loukách a mýtinách tak, aby nebylo příliš mnoho falešných detekcí. Dále by se dal model rozšířit o více druhů lesní zvěře.

Zařízení s čipem ESP32-S3 by mohlo mít ještě menší v režimu hlubokého spánku. Vývojová deska funkčního prototypu má problém se zabudovanou LED diodou, která svítí, když je zařízení v režimu hlubokého spánku. Zde se nabízí možnost odebrat tuto diodu z vývojové desky a tím ušetřit energii. Dále vypnout kompletně komunikační modul, když přechází zařízení do režimu hlubokého spánku.

Díky této práci jsem si rozvinul programovací schopnosti a to zejména v oblasti vestavěných systémů. Vyzkoušel jsem si různé technologie při vývoji – LoRaWAN, Edge Impulse, ElectronJS, MQTT.

Literatura

[1] Barozai, D. K. *What is YOLOV8? A Step-By-Step Guide* [online]. [cit. 2025-18-01]. Dostupné z: https://www.folio3.ai/blog/what-is-yolov8-architecture/.

[2] Brownlee, J. *A Gentle Introduction to Object Recognition With Deep Learning* [online]. 2021 [cit. 2025-18-01]. Dostupné z: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[3] Chen, Y.-T., Chuang, J.-H., Teng, W.-C., Lin, H.-H. a Chen, H.-T. Robust license plate detection in nighttime scenes using multiple intensity IR-illuminator. In: *2012 IEEE International Symposium on Industrial Electronics*. 2012, s. 893–898. DOI: 10.1109/ISIE.2012.6237207.

[4] de Carvalho Silva, J., Rodrigues, J. J. P. C., Alberti, A. M., Solic, P. a Aquino, A. L. L. LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities. In: *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. 2017, s. 1–6. ISBN 978-953-290-071-2.

[5] ECLIPSERA s.r.o.. *PIR Pohybové čidlo pro jednodeskové počítače HC – SR501* [online]. [cit. 2025-09-05]. Dostupné z: https://dratek.cz/arduino/839-hc-sr501-pohybove-cidlo-pro-jednodeskove-pocitace.html.

[6] Edge Impulse. *FOMO: Object detection for constrained devices* [online]. 2024 [cit. 2025-18-01]. Dostupné z: https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices.

[7] Edge Impulse. *On your Espressif ESP-EYE (ESP32) development board* [online]. 2025 [cit. 2025-02-05]. Dostupné z: https://docs.edgeimpulse.com/docs/run-inference/cpp-library/running-your-impulse-esp32.

[8] Espressif Systems. *ESP32-P4-Function-EV-Board* [online]. [cit. 2025-20-01]. Dostupné z: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32p4/esp32-p4-function-ev-board/user_guide.html#getting-started.

[9] Espressif Systems. *ESP32-S3-EYE v2.2* [online]. [cit. 2025-20-01]. Dostupné z: https://github.com/espressif/esp-who/blob/master/docs/en/get-started/ESP32-S3-EYE_Getting_Started_Guide.md.

[10] Feng, J. wen a Wu, Y. chen. Development and Application of Artificial Neural Network. *Wireless Personal Communications*. Září 2018, sv. 102, č. 2, s. 1645–1656. DOI: 10.1007/s11277-017-5224-x. ISSN 1572-834X.

[11] Fořtová, K. *Analýza konvolučních neuronových sítí pro detekci a klasifikaci poškození otisku prstu*. Brno, CZ, 2022. [cit. 2025-18-01]. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/25006/.

[12] Glolab Corporation. *How Infrared motion detector components work* [online]. [cit. 2025-16-01]. Dostupné z: http://www.glolab.com/pirparts/infrared.html.

[13] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning* [online]. MIT Press, 2016 [cit. 2025-17-01]. Dostupné z: http://www.deeplearningbook.org.

[14] Hrdličková, K. *LoRaWAN: Vše, co potřebujete vědět* [online]. 2025 [cit. 2025-27-4]. Dostupné z: https://www.acrios.com/cs/blog/lorawan-vse-co-potrebujete-vedet.

[15] Hu, Y. H. a Hwang, J.-N. *Handbook of Neural Network Signal Processing*. CRC PRESS, 2002. ISBN 0-8493-2359-2.

[16] INFOTO s.r.o.. *Bunaty Mini Full HD fotopast* [online]. [cit. 2025-09-05]. Dostupné z: https://www.infoto.cz/fotopasti/bunaty-mini-full-hd-fotopast/.

[17] INFOTO s.r.o.. *Fotopast BUNATY WIDE Full HD GSM + 32GB karta* [online]. [cit. 2025-09-05]. Dostupné z: https://www.infoto.cz/fotopasti/fotopast-bunaty-wide-full-hd-gsm/.

[18] Kamler, J., Plhal, R., Drimaj, J., Mikulka, O. a Marada, P. *Metodika prevence škod působených zvěří na polních plodinách*. 1. vyd. Brno, Czech Republic: Mendelova univerzita v Brně, 2019 [cit. 2025-10-05]. ISBN 978-80-7509-696-8. Dostupné z: https://ldf-test.mendelu.cz/wp-content/uploads/2022/03/metodika_prevence_kod_zv_na_polch.pdf.

[19] Kang, J., Tariq, S., Oh, H. a Woo, S. S. A Survey of Deep Learning-Based Object Detection Methods and Datasets for Overhead Imagery. *IEEE Access*. IEEE. 2022, sv. 10, s. 20118–20134. DOI: 10.1109/ACCESS.2022.3149052.

[20] Kevin, T. *TinyML — Post-Training Pruning* [online]. 2024 [cit. 2025-19-01]. Dostupné z: https://medium.com/@thommaskevin/tinyml-post-training-pruning-3c20a3053e80.

[21] Kumar, A. D. A. S. a Patil, P. N. Object Detection at Edge Using TinyML Models. *SN Computer Science*. Listopad 2023, sv. 5, č. 11. DOI: 10.1007/s42979-023-02304-z. ISSN 2661-8907. Dostupné z: https://doi.org/10.1007/s42979-023-02304-z.

[22] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. et al. Feature Pyramid Networks for Object Detection. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, s. 936–944. DOI: 10.1109/CVPR.2017.106.

[23] Ma, D. a Yang, J. YOLO-Animal: An efficient wildlife detection network based on improved YOLOv5. In: *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*. 2022, s. 464–468. DOI: 10.1109/ICICML57342.2022.10009855.

[24] Mrocek, J. a Holásek, L. *Jak vybrat tu správnou fotopast?* [online]. Listopad 2022 [cit. 2025-10-05]. Dostupné z: https://www.gearcheckers.com/cs/fotopasti/jak-vybrat-fotopast-pruvodce-nakupem/#nocni-prisvit.

[25] My Game Pics. *My Game Pics Dataset* [online]. Roboflow, duben 2025 [cit. 2025-05-02]. Dostupné z: https://universe.roboflow.com/my-game-pics/my-game-pics.

[26] Nejfotopasti. *Fotopast — Pro koho se hodí a k čemu ji využít?* [online]. Září 2018 [cit. 2025-10-05]. Dostupné z: https://www.nejfotopasti.cz/blog/12-fotopast-pro-koho-se-hodi-a-k-cemu-ji-vyuzit.

[27] Nicolasfree. *Forest Dataset* [online]. Roboflow, říjen 2024 [cit. 2025-05-02]. Dostupné z: https://universe.roboflow.com/nicolasfree/forest-jitth.

[28] O'Shea, K. a Nash, R. An Introduction to Convolutional Neural Networks. *ArXiv preprint arXiv:1511.08458*. Listopad 2015, [cit. 2025-17-01]. DOI: 10.48550/arXiv.1511.08458.

[29] O'Connell, A. F., Nichols, J. D. a Karanth, K. U. Camera Traps in Animal Ecology. In:. Springer, Leden 2011, kap. A History of Camera Trapping, s. 9–26. DOI: 10.1007/978-4-431-99495-4. ISBN 978-4-431-99494-7.

[30] Pykes, K. *What is TinyML? An Introduction to Tiny Machine Learning* [online]. 2023 [cit. 2025-19-01]. Dostupné z: https://www.datacamp.com/blog/what-is-tinyml-tiny-machine-learning.

[31] Ray, P. P. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University – Computer and Information Sciences*. 2022, sv. 34, č. 4, s. 1595–1623. DOI: 10.1016/j.jksuci.2021.11.019. ISSN 1319-1578. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1319157821003335.

[32] Ren, S., He, K., Girshick, R. a Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016 [cit. 2025-19-01]. DOI: 10.48550/arXiv.1506.01497.

[33] Roy, A. M., Bhaduri, J., Kumar, T. a Raj, K. WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. *Ecological Informatics*. 2023, sv. 75, s. 101919. DOI: https://doi.org/10.1016/j.ecoinf.2022.101919. ISSN 1574-9541. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1574954122003697.

[34] singh, P. *Evolution of Object Detection: RCNN, Fast RCNN, and Faster RCNN* [online]. 2023 [cit. 2025-19-01]. Dostupné z: https://medium.com/@2003priyanshusingh/evolution-of-object-detection-rcnn-fast-rcnn-and-faster-rcnn-90cc872e6dae.

[35] Tan, M., Pang, R. a Le, Q. V. *EfficientDet: Scalable and Efficient Object Detection*. 2020. DOI: 10.48550/arXiv.1911.09070.

[36] Tedrake, R. Robotic Manipulation: Perception, Planning, and Control. In: [Course Notes for MIT 6.421]. 2024, kap. Object Detection and Segmentation [cit. 2025-09-05]. Dostupné z: https://manipulation.csail.mit.edu/segmentation.html.

[37] Twin, A. *Overfitting* [online]. 2021 [cit. 2025-17-01]. Dostupné z: https://www.investopedia.com/terms/o/overfitting.asp.

[38] Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W. et al. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, s. 11531–11539. DOI: 10.1109/CVPR42600.2020.01155.

[39] Wong, A., Famuori, M., Mohammad, J. S., Li, F., Chwyl, B. et al. YOLO Nano: a Highly Compact You Only Look Once Convolutional Neural Network for Object Detection. In: *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing – NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, s. 22–25. DOI: 10.1109/EMC2-NIPS53020.2019.00013.

Příloha A

Obsah pamětového média

/ esp32_wildlife_detector
/ aplikace_wildlife_detector_receiver
/ technicka_zprava
/ technicke_dokumentace
/ readme.md