STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor: 18. Informatika

Coopmaster - systém pro kontrolu a automatizaci kurníku

Jaroslav Němec

Hradec Králové 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

COOPMASTER - SYSTÉM PRO KONTROLU A AUTOMATIZACI KURNÍKU

COOPMASTER - SYSTEM FOR COOP CONTROL AND AUTOMATION

AUTOR	Jaroslav Němec
ŠKOLA	Střední škola a vyšší odborná škola aplikované kybernetiky
KRAJ	Královehradecký
ŠKOLITEL	Ing. David Podzimek
OBOR	18. Informatika

Hradec Králové 2025

Prohlášení

Prohlašuji, že svou práci na téma *Coopmaster - systém pro kontrolu a automatizaci kurníku* jsem vypracoval/a samostatně pod vedením Ing. Davida Podzimka a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Dále prohlašuji, že tištěná i elektronická verze práce SOČ jsou shodné a nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a změně některých zákonů (autorský zákon) v platném znění.

V Hradci Králové dne: _____ _____

Jaroslav Němec

Poděkování

Děkuji svému školiteli Ing. Davidu Podzimkovi za obětavou pomoc, podnětné připomínky a nekonečnou trpělivost, kterou mi během práce poskytoval.

Anotace

Ve své práci jsem se zabýval návrhem a realizací kompletního řešení pro malé i střední zemědělce a hospodáře, které má pomoci plnit běžné činnosti. Cílem této práce bylo vytvořit a realizovat systém pro kontrolu a automatizaci chovu hospodářských zvířat.

Klíčová slova

programování, automatizace, zemědělství, neuronové sítě, python, mikroservisní architektura

Annotation

In my work, I have been involved in the design and implementation of a complete solution for small and medium farmers and homesteaders to help them carry out their day-to-day activities. The aim of this work was to design and implement a system for control and automation of livestock farming.

Keywords

programming, automation, agriculture, neural networks, python, microservice architecture

Obsah

Kapitola 1

Úvod

V dnešní době je populární využívat moderní technologie a umělou inteligenci v různých aplikacích jako například monitoring pohybu zákazníků v obchodech nebo ve formě textových modelů čímž je například známe ChatGPT. Zároveň dnes ubývá lidí, kteří by se chtěli zabývat staráním se o hospodářská zvýřata a tudíž je potřeba, aby něco zaujmulo jejich místo. Zároveň jsem sám člověk, který rád zkoumá nové věci v oblasti informatiky a velice ho baví automatizování nejrůznějších úloh. Na základě těchto faktů jsem se rozhodl vytvořit téma práci popisující využití moderních technologií při chovu hospodářských zvýřat. Nápad na tuto práci vznikl díky mojí babičce, která chová doma slepice. Když odjede na dovolenou, stávám se já tím, kdo se o ně musí starat. Slepice je třeba dojet ráno a večer zkontrolovat a spočítat. Tato činnost je časově náročná kvůli cestování a kolikrát i zbytečná, protože většinou se nic neděje. A vzhledem k tomu, že jsem povoláním informatik, budu toto práci věnovat tomu jaké je moje řešení automatizace babiččina chovu kura domácího.

Práce se dělí na teoretickou a praktickou část. V teoretické části jsou popsány základní pojmy a principy, jež jsou důležité pro plné chápání práce. Praktická část popisuje konkrétní implementaci a nasazení asistenčního systému pro chov hospodářských zvýřat. Čtenář může využít znalosti, které nasbírá během čtení teoretické části a jež mu následně pomůže chápat praktická část,

jako návod k realizaci vlastního systému dle jeho potřeb.
Svojí konkrétní implementaci jsem zaměřil na chov Kura domácího z výše popsaných osobních důvodů, a protože byl pro mě nejdostupnější testovací zvíře. Systém jsem realizoval mikroservisní architekturou a jednotlivé služby jsou psány v populárnímu jazyce Python. Jako GUI rozhraní je šikovně použit open source software pro chytrou domácnost Home Assistant.

Kapitola 2

Teoretická část

2.1 Virtualizace

Virtualizace virtualizace-Martin-Polednik je technologie, která se využívá pro efektivnější rozdělení hardwarových zdrojů fyzického počítače mezi virtualizované/hostované počítače. Jednotlivé systémy běží v podstatě na jiném hardwaru, díky čemuž jsou jejich procesy izolovány, což přispívá k bezpečnosti řešení. Virtuální stroj je zároveň jednodužší spravovat, díky tomu, že máme úplnou kontrolu nad daným virtuálním strojem, což pomáhá například při klonování nebo restartu.

2.2 Kontejnerizace

Kontejnerizace virtualizace-Martin-Polednik je technologie umožňující zapouzdření funkcionality jedné aplikace společně se všemi jejími závislostmi a zdroji do jednotky nazývané česky obrazy. Konkrétní běžící instance jednoho obrazu se nazývá kontejner. Tyto kontejnery pak lze spouštět v prakticky neomezeném množsví nezávisle na platformě a výhodou je, že jsou jejich běhy navzájem izolované, takže běh jednoho neovlivní ostatní. Je to v podstatě forma virtualizace se všemi jejími výhodami, s tím rozdílem, že daný kontejner obsahuje jen nezbytné věci pro běh dané aplikace.

2.2.1 Docker Engine

Jedna z technologií používaných pro kontejnerizaci je Docker Engine kontejnerizace-docker. Používá se pro tvorbu, správu, orchestraci, verzování a nasazování jednotlivých kontejnerů.

2.2.2 Docker Compose

Docker compose kontejnerizace-docker-compose je nástroj pro Docker, který nám umožňuje tvořit komplikované ekosystémy jednotlivých kontejnerů, jež spolu v rámci něho mohou spolupracovat. Pomáhá nám například se síťováním nebo konfigurací kontejnerů.

2.3 Mikroservisní architektura

Mikroservisní architektura je přístup k vývoji softwarových aplikací, kdy je celek rozdělen na malé, nezávislé služby nazývané mikroservisy. Každá služba běží samostatně, komunikuje s ostatními službami pomocí dobře definovaného API a je zaměřena na konkrétní funkcionalitu. Tento model umožňuje snadnější údržbu, škálovatelnost a flexibilitu systému.

2.4 Jazyk Python

Python je vysokoúrovňový programovací jazyk známý svou jednoduchou a čitelnou syntaxí. Podporuje několik programovacích paradigmat, včetně objektově orientovaného, procedurálního a funkcionálního programování. Python je široce používán v různých oblastech, jako je webový vývoj, vědecké výpočty, umělá inteligence, strojové učení a datová analýza. Disponuje rozsáhlou standardní knihovnou a množstvím externích modulů, které usnadňují vývoj komplexních aplikací. Je multiplatformní, což znamená, že programy v Pythonu lze spouštět na různých operačních systémech.

2.5 Jazyky Wire a C++

2.5.1 Wiring

Wiring je open-source programovací jazyk a vývojové prostředí určené pro práci s mikrokontroléry. Jeho cílem je usnadnit programování elektronických zařízení a interaktivních projektů, zejména pro umělce, designéry a studenty. Syntaxe Wiringu vychází z jazyka C++ a je navržena tak, aby byla snadno pochopitelná i pro začátečníky. Wiring poskytuje intuitivní prostředí pro psaní kódu, jeho kompilaci a nahrávání přímo do mikrokontroléru.

2.5.2 C++

C++ je výkonný, objektově orientovaný programovací jazyk vyvinutý jako rozšíření jazyka C. Umožňuje kombinovat nízkoúrovňové programování blízké hardwaru s vysokou úrovní abstrakce, což z něj činí univerzální nástroj pro různé typy softwaru. C++ podporuje více programovacích paradigmat, včetně procedurálního, objektově orientovaného a generického programování. Je široce využíván pro vývoj systémového softwaru, her, grafiky, aplikací s vysokým výkonem a vestavěných systémů.

2.6 Git

Git je systém pro správu verzí, který umožňuje sledovat změny v zdrojovém kódu během vývoje softwaru. Tento systém se stal již dnes standardem moderního vývoje softwaru. Vytvořil ho Linus Torvalds v roce 2005 pro potřeby vývoje jádra operačního systému Linux. Git umožňuje více vývojářům pracovat současně na jednom projektu bez rizika přepsání či ztráty práce. Každá kopie repozitáře obsahuje kompletní historii projektu, což zajišťuje možnost vrátit se k předchozím verzím a pracovat i bez připojení k internetu. Důležité funkce Gitu zahrnují rychlost, efektivitu při práci s velkými projekty a podporu nelineárního vývoje prostřednictvím větvení.

2.7 Github workflow

GitHub Workflow je nástroj pro automatizaci procesů při vývoji softwaru na platformě GitHub. Umožňuje vytvářet a spravovat tzv. workflow pomocí souborů ve formátu YAML, které definují jednotlivé kroky. Tyto kroky nebo také akce, se mohou spouštět na základě různých událostí, jako je push kódu nebo vytvoření pull requestu. GitHub Workflow podporuje kontinuální integraci a nasazení (CI/CD), automatické testování, nasazení aplikací a další úlohy.

2.8 Mqtt

MQTT (Message Queuing Telemetry Transport) je lehký messagingový protokol pro publikování a odebírání zpráv, navržený pro komunikaci mezi zařízeními v sítích s omezenou kapacitou nebo vysokou latencí. Je často využíván v oblasti Internetu věcí (IoT) pro přenos dat mezi senzory, akčními členy a centrálními systémy. MQTT pracuje na principu architektury klient–server, kde klienti(publisheři) publikují zprávy na určité téma(topic) a broker tyto zprávy distribuuje odběratelům(subscriberům), kteří jsou na dané téma přihlášeni.

2.9 Home Assistant

Home Assistant je open-source platforma pro automatizaci domácnosti napsaná v jazyce Python. Umožňuje centrálně ovládat a monitorovat různá zařízení v chytré domácnosti, jako jsou osvětlení, termostaty, bezpečnostní systémy a další IoT zařízení. Home Assistant podporuje integraci s více než tisíci různými komponentami a službami, což umožňuje vytvořit komplexní a přizpůsobené automatizační scénáře. Platforma běží lokálně, což zajišťuje vyšší úroveň soukromí a nezávislost na cloudových službách. Uživatelé mohou využít webové rozhraní pro správu a nastavování automatizací nebo psát vlastní konfigurace pomocí YAML souborů.

2.10 Arduino

Arduino je open-source platforma pro prototypování elektroniky založená na snadno použitelném hardwaru a softwaru. Skládá se z mikroprocesorové desky a vývojového prostředí Arduino IDE, které využívá jazyk podobný C/C++. Arduino desky umožňují komunikaci s různými senzory a akčními členy, což usnadňuje tvorbu interaktivních projektů. Podporuje řadu rozšiřujících modulů, tzv. shieldů, které rozšiřují jeho funkčnost například o bezdrátovou komunikaci, ovládání motorů či připojení k internetu. Arduino usnadňuje rychlý vývoj a testování elektronických aplikací bez hlubokých znalostí elektroniky.

2.11 Ip kamera a RTSP

IP kamera je digitální zařízení, které přenáší obraz a zvuk přes IP sítě, jako je internet nebo lokální síť. To umožňuje vzdálený přístup k živému vysílání nebo záznamům bez nutnosti speciálního kabelového připojení. Pro efektivní přenos multimediálních dat v reálném čase se často využívá protokol RTSP (Real Time Streaming Protocol).

2.11.1 RTSP

RTSP je síťový protokol, který umožňuje kontrolu nad streamováním médií, jako je spouštění, zastavování nebo přetáčení videa.

2.12 Cloudflare tunneling

Cloudflare Tunneling je služba poskytovaná společností Cloudflare, která umožňuje bezpečné a jednoduché propojení lokálního serveru s internetem. Využívá tunelování k vytvoření šifrovaného spojení mezi vaším interním systémem a infrastrukturou Cloudflare bez nutnosti otevírat porty na firewallu nebo nastavovat složité síťové konfigurace. To usnadňuje publikování

webových aplikací nebo služeb hostovaných na lokálních serverech, aniž by byla odhalena jejich skutečná IP adresa.

2.13 Flask

Flask je lehký webový framework pro Python, který umožňuje rychlé a jednoduché vytváření webových aplikací. Patří mezi mikroframeworky, což znamená, že poskytuje pouze základní funkce potřebné pro webový vývoj, jako je směrování URL a zpracování HTTP požadavků. Díky své modularitě umožňuje vývojářům přidávat rozšíření a knihovny podle potřeby, například pro práci s databázemi, autentizaci či validaci. Flask využívá šablonovací systém Jinja2 pro vytváření dynamických HTML stránek a nástroj Werkzeug pro WSGI kompatibilitu.

2.14 APScheduler

APScheduler (Advanced Python Scheduler) je knihovna pro programovací jazyk Python, která umožňuje plánovat a spouštět úlohy v určených časech nebo intervalech. Poskytuje nástroje pro definování plánů úloh, jako jsou jednorázové spuštění, opakované intervaly nebo specifické časové výrazy.

2.15 Paho-mqtt

Paho-MQTT je oficiální klientská knihovna pro protokol MQTT určená pro programovací jazyk Python. Tato knihovna umožňuje aplikacím v Pythonu komunikovat s MQTT brokerem, což je centrální bod pro výměnu zpráv v rámci MQTT messaging systému.

2.16 Python-dotenv

Python-dotenv je knihovna pro programovací jazyk Python, která umožňuje načítat proměnné prostředí ze souboru.env. Tento soubor obsahuje konfiguraci ve formátu klíč–hodnota, kde jsou uloženy citlivé informace, jako jsou hesla, API klíče nebo nastavení aplikace.

2.17 Pyserial

PySerial je knihovna pro programovací jazyk Python, která umožňuje komunikaci přes sériové porty. Poskytuje jednotné rozhraní pro přístup k sériovým portům na různých operačních systémech, jako jsou Windows, Linux nebo macOS. PySerial umožňuje otevírání, čtení a zápis dat do sériového portu přímo z Python aplikací.

2.18 Strojové učení

Strojové učení je oblast umělé inteligence zaměřená na tvorbu algoritmů a modelů, které umožňují počítačům se samostatně učit z dat bez explicitního naprogramování každého kroku. Využívá statistických metod a matematických modelů k analýze velkého množství dat s cílem odhalit vzory a vztahy. Tyto vzory pak slouží k predikci nebo rozhodování v různých situacích. Strojové učení se dělí na učení s učitelem, bez učitele a posilované učení. Praktické aplikace zahrnují rozpoznávání hlasu a obrazu, doporučovací systémy, detekci podvodů či autonomní řízení vozidel. Díky strojovému učení mohou systémy neustále zlepšovat svou výkonnost na základě nových dat.

2.19 Yolo Ultralytics

Ultralytics YOLO je pokročilý systém pro detekci objektů v reálném čase založený na hlubokém učení. Jedná se o implementaci architektury YOLO

(You Only Look Once), kterou vyvinula společnost Ultralytics. Tento model využívá konvoluční neuronové sítě k analýze obrazových dat a současné identifikaci více objektů během jediného průchodu sítí. Díky své vysoké rychlosti a přesnosti je ideální pro aplikace náročné na čas, jako je autonomní řízení, bezpečnostní systémy nebo analýza videa. Ultralytics YOLO je dostupný jako open-source software, což umožňuje jeho široké využití ve výzkumu i v průmyslových aplikacích.

2.20 Tenzometrický senzor

Tenzometrický senzor je zařízení sloužící k měření mechanického napětí nebo deformace v materiálu či konstrukci. Základním principem tenzometru je využití změny elektrického odporu vodivého materiálu při jeho mechanickém zatížení. Nejčastěji se používají tenzometry s kovovou fólií nebo drátkem, který je pevně spojen s měřeným objektem. Když je objekt zatížen silou, dojde k jeho deformaci, což způsobí prodloužení nebo zkrácení tenzometru a tím i změnu jeho elektrického odporu. Tato změna je úměrná velikosti aplikovaného napětí a může být přesně změřena pomocí elektrických obvodů, jako je Wheatstoneův můstek. Tenzometrické senzory nacházejí uplatnění v oblastech jako je strojírenství, stavebnictví, letectví či při vývoji nových materiálů, kde je důležité sledovat mechanické vlastnosti a bezpečnost konstrukcí.

2.21 Power over Ethernet (PoE)

Power over Ethernet (PoE) je technologie umožňující přenos elektrické energie společně s daty prostřednictvím standardního ethernetového kabelu typu kroucená dvojlinka. Tato technologie umožňuje napájet síťová zařízení, jako jsou IP kamery, bezdrátové přístupové body nebo VoIP telefony, bez potřeby samostatného napájecího zdroje. PoE využívá nevyužité páry vodičů v kabelu nebo kombinuje napájení s datovými signály na stejných vodičích.

Existují různé standardy PoE, jako IEEE 802.3af, 802.3at a 802.3bt, které definují maximální výkon a kompatibilitu zařízení. Použití PoE zjednodušuje instalaci, snižuje náklady na kabeláž a umožňuje flexibilnější umístění zařízení bez závislosti na elektrických zásuvkách.

2.22 Wifi extender

Wi-Fi extender, také známý jako repeater nebo zesilovač signálu, je zařízení určené k rozšíření dosahu bezdrátové sítě. Přijímá existující Wi-Fi signál z routeru a znovu ho vysílá do oblastí s nedostatečným pokrytím. Tím eliminuje ẅrtvé zónyẅ domácnosti nebo kanceláři, kde je signál slabý nebo žádný. Instalace je obvykle jednoduchá a nevyžaduje dodatečné kabely. Wi-Fi extendery podporují různé standardy Wi-Fi, jako 802.11n nebo 802.11ac, a nabízejí přenosové rychlosti odpovídající těmto standardům. Pro efektivní rozšíření sítě je důležité umístit extender tam, kde ještě přijímá silný signál z routeru.

2.23 TailScale vpn

Tailscale VPN je moderní služba pro vytváření virtuálních privátních sítí, která využívá protokol WireGuard k zajištění bezpečné komunikace mezi zařízeními. Umožňuje snadné nastavení sítě bez složitých konfiguračních procesů tradičních VPN řešení. Tailscale vytváří šifrované peer-to-peer spojení mezi zařízeními na základě jejich identity, spravované prostřednictvím cloudu. To umožňuje uživatelům bezpečně přistupovat k interním sítím a službám odkudkoli na světě. Díky automatické správě síťových konfigurací a firewallu snižuje nároky na údržbu a zvyšuje celkovou bezpečnost. Tailscale je vhodný pro jednotlivce, týmy i organizace hledající efektivní a jednoduché VPN řešení pro propojení svých zařízení.

2.24 Raspberry PI

Raspberry Pi je řada malých jednočipových počítačů vyvinutých nadací Raspberry Pi ve Velké Británii s cílem podpořit výuku informatiky a programování. Tyto cenově dostupné zařízení nabízejí plnohodnotné funkce počítače na kompaktním hardwaru. Raspberry Pi je vybaven procesorem ARM, grafickým výstupem HDMI, USB porty, Ethernetem a dalšími rozhraními pro připojení periferií a senzorů. Nejčastěji na něm běží operační systém Raspbian, který je založen na Linuxu. Díky své univerzálnosti a přístupnosti je široce využíván nejen ve vzdělávání, ale i v projektech IoT, domácí automatizace, robotiky a dalších oblastech vyžadujících flexibilní a výkonnou výpočetní platformu.

2.25 Raspberry PI OS

Raspberry Pi OS je oficiální operační systém pro počítače Raspberry Pi, vyvinutý nadací Raspberry Pi Foundation. Je založen na distribuci Debian Linux a je optimalizován pro hardware Raspberry Pi. Systém poskytuje stabilní a uživatelsky přívětivé prostředí s podporou široké škály aplikací. Raspberry Pi OS obsahuje předinstalované nástroje pro výuku programování, jako jsou Python, Scratch či Sonic Pi, což podporuje vzdělávací poslání platformy. Nabízí grafické uživatelské rozhraní, ale i možnost běhu v terminálu pro pokročilé uživatele. Díky pravidelným aktualizacím a široké komunitní podpoře je ideální volbou pro projekty v oblasti IoT, domácí automatizace či robotiky.

2.26 Yaml

YAML (YAML Ain't Markup Language) je formát pro serializaci dat navržený pro snadnou čitelnost a zápis člověkem. Používá se převážně pro zápis konfigurací v různých aplikacích a systémech. Jeho syntaxe je založena na odsazení a struktuře klíč-hodnota, což umožňuje vytvářet přehledná a hierarchicky uspořádaná data. YAML podporuje různé datové typy, jako jsou řetězce,

číselné hodnoty, seznamy a slovníky. Díky své jednoduchosti a flexibilitě je oblíbený mezi vývojáři a administrátory. Využívá se například v nástrojích pro správu kontejnerů, jako je Docker Compose, nebo v automatizačních systémech typu Ansible pro definování infrastruktury jako kódu.

2.27 GUI

GUI (Grafické uživatelské rozhraní) je způsob interakce člověka s počítačem pomocí grafických prvků, jako jsou ikony, tlačítka, menu a okna. Na rozdíl od textového rozhraní, kde uživatel zadává příkazy prostřednictvím textu, GUI umožňuje ovládání systému pomocí myši, dotykové obrazovky či jiného ukazovacího zařízení. Tento přístup zvyšuje intuitivnost a uživatelskou přívětivost aplikací a operačních systémů. GUI je založeno na konceptu WIMP (Window, Icon, Menu, Pointing device), který standardizuje prvky rozhraní pro konzistentní uživatelský zážitek. Vývoj grafických rozhraní vyžaduje znalost programovacích jazyků a knihoven, jako jsou Qt, GTK nebo Windows API.

2.28 HTTP a REST

HTTP (Hypertext Transfer Protocol) je základní komunikační protokol pro web, který umožňuje přenos dat mezi klientem a serverem pomocí žádostí a odpovědí.

2.28.1 REST (Representational State Transfer)

REST(Representational State Transfer) je architektonický styl pro tvorbu webových služeb, který využívá protokol HTTP. REST definuje sadu principů pro uspořádání API tak, aby byly škálovatelné, flexibilní a snadno použitelné. V rámci RESTful služeb se využívají standardní HTTP metody jako GET, POST, PUT a DELETE k manipulaci s prostředky. Tento přístup usnadňuje komunikaci mezi různými systémy a umožňuje efektivní integraci aplikací v rámci webu.

2.29 Latex

LaTeX je vysoce kvalitní systém pro sazbu dokumentů, který umožňuje tvorbu profesionálně vypadajících vědeckých a technických textů. Je založen na systému TeX, který vyvinul Donald Knuth. LaTeX využívá značkovací jazyk, ve kterém uživatelé definují strukturu dokumentu pomocí příkazů a prostředí. Tento přístup umožňuje automatickou správu číslování kapitol, odkazů, bibliografií a tvorbu obsahu. LaTeX je oblíbený zejména v akademickém prostředí pro svou schopnost precizně sázet matematické vzorce a rovnice. Díky své flexibilitě je vhodný pro vytváření diplomových prací, odborných článků, knih a dalších publikací, kde je kladen důraz na typografickou kvalitu.

2.30 Backend

Backend je část softwarové aplikace, která běží na serveru a stará se o logiku, zpracování dat a komunikaci s databázemi. Je to neviditelná vrstva pro uživatele, ale klíčová pro funkčnost aplikace. Backend přijímá požadavky od frontendu (uživatelského rozhraní), zpracovává je a odesílá zpět potřebné informace.

2.31 Environment proměnné

Environment proměnné jsou nastavení v operačním systému, která ovlivňují chování běžících programů a skriptů. Umožňují předávat důležité informace, jako jsou cesty k souborům, konfigurace nebo přístupové údaje, bez potřeby je pevně zakomponovat do kódu.

Kapitola 3

Praktická část

Tato část práce předvede čtenáři konkrétní realizaci myšlenky probírané v této prácí. Zároveň v se v ní čtenář dozví jak v praxi využít poznatky nabyté v teoretické části.

3.1 Rozvržení person

Před tím než se započne tvorba jakékoli aplikace, je třeba určit, pro koho danou aplikaci tvoříme a jak bychom chtěli, aby jí tento uživatel používal. V našem případě jde o to, že systém by měl být schopný používat člověk znalý v chovu hospodářských zvířat, ale často méně zdatný v používání informačních technologií a mobilních aplikací. Na základě toho, že už víme, kdo je cílový uživatel můžeme se zamýšlet jakou funkcionalitu do řešení implementovat. Jedná se tedy o to, aby aplikace umožňovala uživateli vzdálený dohled na jeho chov a pomáhala mu šetřit čas zautomatizováním každodenních činnosti. Rozhodlo se tedy, že aplikace bude uživateli poskytovat následující funkce

- Vzdálený přístup zkrze internet odkudkoli
- Pohledy z bezpečnostních kamer ve vnitřním i venkovním výběhu
- Dálkové ovládání a automatizace světla a bezpečnostních dvířek v kurníku

- Intuitivní vizualizace stavů jednotlivých hnízd(sedící slepice, v opačném případě počet vajec)
- Automatické upozornění notifikacemi v telefonu na vetřelce ve výběhu
- Vizualizace aktuálních povětrnostních podmínek v kurníku (teplota a vlhkost)

3.2 Návrh architektury a volba technologií

Díky tomu, že si přesně určíme, kdo je cílový uživatel naší aplikace, a k tomu dáme do hromady, co uživatel od naší aplikace očekává. Jsme nyní schopni bez větších problémů teoreticky navrhnout architekturu našeho systému a přesně popsat požadovanou funkcionalitu, jakou budou disponovat jeho jednotlivé části. Celý ekosystém se skládá z několika samostatných celků

- Fyzická zařízení jako kamery a senzory
- Backend systému obsahující hlavní funkcionalitu(sekce **??**)
- GUI(sekce **??**)
- Moduly pro komunikaci mezi GUI a Backendem
- Modul pro propagaci a zpřístupnění aplikace z internetu

3.2.1 Backend

Jako teoretický model na základě kterého, budeme organizovat backend a třídit jeho funkcionalitu, jsem zvolil mikroservisní architekturu(sekce **??**). Tento způsob rozvržení zodpovědnosti jednotlivých částí systému jsem zvolil, kvůli velké možnosti rozšíření, zapouzdření funkcionality a snadné úpravě jednotlivých služeb bez nutnosti kompletního restartu systému případně znovunasazení. Služby jsou psány v programovacím jazyce Python(sekce **??**) a

využívají jeho knihovny. Na základě určeného způsobu zapouzdření funkcionality je funkcionalita backendu rozdělena do následujících služeb, které jsou pojmenovány dle jejich účelu

- Camera driver
- Scale driver
- Room driver
- Health checker
- Room assistant
- Nest watcher
- Dog alarm
- Chicken watch guard

Camera driver

Camera driver je služba zodpovědná za komunikaci s ip kamerou(sekce **??**).
S ní komunikuje pomocí protokolu RTSP(sekce **??**). Url ip kamery, k níž je driver přiřazen, je službě předávána pomocí envirnoment proměnných(sekce **??**).
Pro komunikaci se zbytkem backendu poskytuje služba své REST(sekce **??**) api. Konkrétně při zavolání na endpoint driver stáhne nejnovější obrázek z kamery a vrátí ho jako odpověď na volání.

Scale driver

Scale driver zodpovídá za komunikaci mezi fyzickou váhou a službami, které využívají data o vážení.
Protože jako řídící jednotka váhy je použito Arduino(**??**) tento modul komunikuje přes serialový port pomocí protokolu USB a na dotaz přijmutý RESTovým api poskytne jako odpověď hodnotu načtenou z váhy. Pro služby v systému tento driver poskytuje data o hmotnosti opět pomocí REST api.

Room driver

Room driver zařizuje komunikaci mezi ostatnímy službami a řídící jednotkou v kurníku, která ovládá dveře a světlo.
Jako mozek řídící jednotky je použito opět Arduino a tomu je třeba přizpůsobyt architekturu služby. Tento modul má tedy za úkol přes serialový port pomocí protokolu USB posílat příkazy a načítat stavy řídící jednotky na základě requestů příchozích na REST api služby. Pro služby v systému služba na vystavuje GET a POST endpointy

Health checker

Health checker je malá služba určená pro správce systému.
Poskytuje informace o tom zda všechny potřebné služby běží, aby správce nebyl nucen přihlašovat se vzdáleně na server a manuálně kontrolovat každou službu. Výpis stavů jednotlivých služeb je poskytován RESTovým api [GET] /status

Room assistant

Room assistant je zpřístupňuje komunikaci mezi GUI tedy konkrétně Home Assistantem a Room driverem.
S Home Assistantem je komunikace realizována pomocí MQTT(**??**) a s Room driverem pomocí HTTP(**??**) protokolu. Základní funkcí je zpracování a přeposlání příkazů do Room Driveru odebíraných z témat Služba očekává, že na tato témata budou chodit zprávy open / close pro dveře a on / off jako příkazy pro světlo. Další úlohou je periodické načítání a aktualizace informací o stavu dveří, světla a dat z teplotního a vlhkostního senzorů. Tato data by měl Room assistant pravidelně načítat a posílat přes MQTT do Home Assistanta.

Nest watcher

Tato služba interpretuje stavy jednotlivých hnízd v kurníku pro Home Assistanta. Hlavní funkcí je načítání a analýza dat z jednotlivých vah v hnízdech,

která jsou reprezentována Scale Drivery.
Data jsou načítána několikrát do minuty a ukládána do databáze s časovým údajem, kdy byl záznam vytvořen. Následně se jednou za minutu vyhodnotí průměrná hodnota během posledních několika vážení. Na základě tohoto údaje jsme schopni zjisti několik případů

- hnízdo je prázdné (hodnota na váze nepřevyšuje 50 g)
- v hnízdě se nacházejí vejce (hmotnost jednoho vejce je průměrně 50 g)
- v hnízdě sedí slepice (hmotnost slepice se pohybuje okolo 1200 g a více)

Pokud je na váze průměrně méně než 50 g, vzhledem k možným chybám měření, takový případ vyhodnotíme jako, že je hnízdo prázdné. Jesliže se hodnota pohybuje mezi 50 a 1200 gramy, znamená to, že v hnízdě jsou pravděpodobně vejce, a jejich počet je vypočítán vydělením celkové hmotnosti a hmotnosti jednoho vejce. V případě, že je na váze více jak 1200 g, vyhodnotí služba, že v hnízdě sedí slepice. Tyto tři zmíněné informace služba následně pomocí MQTT předává do Home Assistanta.

Dog alarm

Služba Dog alarm má detekovat nebezpečí ve výběhu a poslat tuto zprávu do Home Assistanta.
Aktuální záběry jsou pomocí GET requestů stahovány z konkrétní instance služby Camera driver, která je přiřazena ke kameře ve výběhu. Analýza probíhá v určitých intervalech za pomocí umělé inteligence, kde je konkrétně použita metoda detekce objektů. Jakmile jako výsledek klasifikace vyjde jednoznačně, že v záběru byl spatřen pes nebo jiný predátor, je zpráva poslána pomocí MQTT do Home Assistanta společně s konkrétním záběrem, na němž byl predátor detekován. Tato služba zároveň přes MQTT posílá do Home Assistanta aktuální záběr z kamery.

Chicken watch guard

Úkolem služby Chicken watch guard je sledovat stav a počet slepic v kurníku. Aktuální záběry jsou stejně jako u Dog alarmu stahovány z konkrétního Camera driveru v kurníku. Následně po získání záběru proběhne detekce objektů a počet těchto objektů udává počet detekovaných slepic v obraze. Tato hodnota je publikována pomocí MQTT do Home Assistanta. Vedlejší funkcí služby je průběřné posílání aktuánlho pohledu z kamery v kurníku do Home Assistanta.

3.2.2 GUI

Aplikace musí mít rozhodně i grafické rozhraní. Pro tento účel byla zvolena open source aplikace Home Assistant(sekce **??**). Home assistant byl zvolen kvůli jeho univerzálnosti, rozsáhlé podpoře, komunitě bohaté na custom řešení a obrovské možnosti konfigurace a přispůsobení, čehoš pro ovládání a vizualizaci dat z našeho systému hojně využijeme.

3.2.3 Komunikace mezi Backendem a Frontendem

Je třeba zajistit komunikaci mezi Home Assistantem a Backendem. Tuto úlohu musíme přijmout velice zodpověďně a navrhnout řešení, které půjde opět snadno rozšířit a modifikovat. Nelze proto použít klasické HTTP(sekce **??**), z toho důvodu že bychom komunikaci vázali na buď doménové jméno a port nebo ip adresu a port. Toto řešení má problém v tom, že pokud bychom potřebovali změnit, buď umístění částí Backendu nebo port jedné ze služeb, bylo by třeba překonfigurovat i home assistanta. Dalším problém nastane, když potřebujeme z Backendu poslat například notifikaci do Home Assistanta, je pro to potřeba, aby jednotlivé služby na Backendu věděli, kde na síti Home Assistant běží, což je věc, která se může měnit, a museli bychom naopak přenastavovat jednotlivé služby. Jako řešení se nabízí použít messaging konkrétně třeba technologii MQTT(sekce **??**). Tato technologie se běžně používá u IoT zařízení a pro naše použití bude vynikajícím řešením.

- aplikace se rozvrhla do několika modulů
- jako první moduly pro komunikaci s hardware nazvané drivery; takže vzniknul scale driver pro komunikaci s váhou, kamerami a arduinem pro dveře, světlo a senzory
- následně vzniky služby na základě požadovaných funkcí nest watcher, chicken watch guard, dog alarm, room assistant a ještě drobná službyčka health checker pro kontrolu a reportování systému a jeho chyb
- drivery komunikují s výkonými službami pomosí http protokolu
- výkonné služby komunikují s home assistantem pomocí mqtt realizovaným mosquitto serverem

3.2.4 Výběr technologií

- vybrali jsme si python a pro clasifikaci framework od ultralitics což znamená modely YOLO
- jako databáze pro sběr statistik byl vybrán postgres pro svou robustnost a dobrou cenu
- mohlo se to psát klidně i v C# a použít třeba azure pro classifikaci
- pro kontejnerizaci jsme zvolili docker; dalo se i třeba podman nebo podobné ale toto se učíme ve škole a rád si to zopakuji a zlepším tak své dovednosti
- python pro mě byla trochu výzva ale dopadlo to dobře

3.3 Implementace jednotlivých modulů

- implementace probíhala v jazyce python za využití vypsaných knihoven viz readmečka modulů
- jak se konfiguroval logger, flask blueprinty, jak propojit python a arduino, jak na to s cronem/schedulerem, jak posílat mqtt a přijímat(jaká je struktura našich topiců, jak se to pojmenovává) - verzuje se to na github - na githubu běží workflow které vytváří jednolivé docker image pro každý modul a uploaduje je na můj docker hub kvůli snadnému deployi a distribuci po internetu

3.4 Tvorba GUI rozhraní

- vybral se teda ten home assistant a ted ho nakonfit
- konfiguruje se to přes yaml configuration.yaml což je hlavní konfigurák HA
- byla výzva přijít na to jak se přidávají vlastní mqtt senzory viz seznam závad v trellu
- po přidání senzorů jsem si hrál s vizualizací jednotlivých hnízd tak aby to pro uživatele bylo přívětivě
- napsal jsem proto vlastní komponentu pro HA viz foto a trello
- zajímavé zjištění bylo že state který jsem využíval pro předávání textových zpráv má maximální velikost 255 bytů
- takže jsem z jsonu přešel na csv
- pokecat trochu o tom jak vytvořit takovou komponentu jaké to má části a předpoklady a jak se to následně přidává a konfiguruje v HA
- následně pak konfigurace automations pro dveře, přepínačů pro manuální ovladání světla a dvěří, obrázků z kamer které taky chodí pomocí mqtt, a následně ještě dog alert aby se poslalo upozornění pokud je mobilní apka a na dashboardu vyskočil daný obrázek a výstrahou
- závěrem pak výpis teploty, vlhkosti a počtu slepic v kurníku které systém poznal

3.5 Rozmyšlení konkrétní implementace

- budeme potřebovat váhu pro kontrolu hnízd zda tam slepice je a nebo kolik je tam vajec
- budeme potřebovat ideálně ip kamery s vhodným IP krytím abychom mohly monitorovat kurník vevnitř a venku
- budeme potřebovat ovladačku asi arduino pro dveře, světlo, senzory teploty a vlhkosti
- předchozí věci je potřeba dostat na síť takže nějaké rpi pro předávání komunikace a směrování

- a všechno to musí řídit něco s dostatečným výkonem pro klasifikace třeba my tu máme nucka s RTX2080
- uživatelské rozhraní se rozhodlo že je zbytečné vyrábět vlastní a ztrácet tím čas lepší bude použít HA který disponuje všemi funkcemi je open source a má komunitu která ho udržuje - první řešení ale bude na stole takže se nemusíme zaobírat detaily kontkrétní instalace, alespoň pro zatím

3.6 První pracovní zapojení a běh

- zjistilo se že bez poe je to špatná volba
- kamery žerou dost proudu
- bylo potřeba odladit nastavení kamer a jejich statické ip adresy, kamery měli zabezpečení na blokování ip address a tak
- byl problém s kontakty na váze takže se nakonec museli vyměnit lisované za pájené
- na stole to většinou funguje dobře

3.7 Nasazení do kurníku

- bylo potřeba natahat elektřinu a internet
- elektřina nebyla problém vzala se z kůlny zkrz díru ve zdi
- horší to bylo s netem protože wifi signál do kurníku nedosáhne
- vyřešilo se to wifi extenderem od tplinku který má zároveň i ethernet výstup díky čemuž nemuselisme dlouhého tahati kabelu a šlo nám to vzduhem přes dvůr a ve stodole pak drátem
- kamery bylo potřeba napájet a taky připojit přes internet; zprvu jsem si myslel že poe nebude potřeba ale taha 230 by bylo zbytečné námahy takže jsem pořídil poe switch od tplinku a ten připojuje kamery
- bylo někde potřeba udělat místo kam se nainstaluje celá technologie kurníku
- zvolila se plechová bedna ze starého domovního rozvaděče do níž se pro

jednotlivé prvky vytvožili na míru držáčky a pouzdra ps.: fotky z tisku a pak z rozvaděče
- jak se zrealizovala váha
- jak vypadá řídící jednotka pri room assistanta
- jaké tam jsou dveře pro slepice
- celé je to propojené s rpi 5 které to v kurníku řídí a s ním pak komunikuje nuc pomocí tail scailu ale to zas v další kapitole
- bylo potřeba zkalibrovat hmotnosti slepic a vajec

3.8 Konfigurace vzdáleného přístupu

- rpi propojene s nuckem a dev kompama přes tailscail
- v docker compose na nuckoj se vyrobila nová network jenom pro home assistanta a jeho propagaci na venek
- do vzniklé sítě se přidal ještě kontejner Cloudflared který zajišťuje tunel ven na cloudflare a přes jeho firewall a proxyny do internetu
- cloudflare tunel bylo potřeba namapovat na doménu kterou vlastníme
- pronajmul jsem si doménu u doméhového registrátora forpsi
- zmínění jaká plynou nebezpečí z tohoto řešení

3.9 Rozvržení práce do budoucna

- jak by se řešení dalo zoptimalizovat
- vylepšení modelu pro classifykaci
- vylepšení a zpřesnění vah
- kontrola napajedla
- kontrola krmítka - implementace autonomního chování do Room Assistanta

3.10 Ekonomická stránka projektu

Kapitola 4

Zamyšlení nad možnými dalšími příležitostmi pro automatizaci

4.1 Co vše se dá automatizovat?

- skleník
- akvarium
- kravín a pastviny
- záhony a zahrada

4.2 Automatizace skleníku

- kontrola vlhkosti, větrání na základě teploty ve skleníku, přidávání hnojiva do zálivky

4.3 Chytré akvárium

- chytré topení, filtrace, krmítko, kontrola kvality vody

4.4 Monitoring kravína a pastvin

- zranění kráva
- cizí pes
- cizí člověk

4.5 Monitoring záhonů

- pes krade mrkev
- utekly slepice
- zajíc
- plíseň na bramborách
- sucho

Kapitola 5

Závěr

Závěrem lze říct, že jsme úspěšně vyvinuli systém, který je schopný v reálném čase pomáha plnit každodení úkony, které by hospodář musel jinak dělat sám. Díky naší aplikaci může člověk kontrolovat situaci v kurníku případně i ve výběhu. Systém ho sám informuje, pokud je něco v nepořádku, například v případě, že je ve výběhu vetřelec, který by mohl představovat nebezpečí pro chované slepice. Dále se nám na podobném principu povedlo implementovat kontrolu počtu slepic v kurníku, což lze použít například při automatizaci zavírání dvířek. Systém také poskytuje aktuální záběry z bezpečnostních kamer díky, kterým si může hospodář sám kontrolovat situaci svého chovu a nemusí díky tomu být přítomen fyzicky. V poslední řadě jsme také implementovali část systému, která je schopna kontrolovat stavy jednotlivých hnízd a zjistit zda v něm sedí slepice případně kolik vajec v hnízdě je. Tato data lze také použít ke tvorbě statistik, díky nimž může člověk analizovat různé aspekty svého chovu.

Seznam obrázků

Seznam tabulek

Seznam rovnic