ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Vývoj software pro kamerový systém vozidel hromadné dopravy

Bc. Martin Lufinka **2019**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin LUFINKA**
Osobní číslo:	**E16N0075P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Vývoj software pro kamerový systém vozidel hromadné dopravy**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

Navrhněte softwarové vybavení pro řídicí počítač kamerového systému. Software musí umožňovat:

1. Záznam streemu videa/audia z několika zdrojů včetně ukládání časových značek.
2. Optimalizujte systém ukládání dat s ohledem na "opotřebení" paměťového média.
3. Monitorovat stav interní zálohy napájení a adekvátně reagovat při výpadku napájení.
4. Komunikace se stacionárním serverem - posílání záznamů z událostí nebo zaslání dat na základě žádosti. Tato komunikace proběhne v případě dostupnosti ethernetového připojení.

Navrhněte požadavky na konfiguraci řídicího počítače s ohledem na počet kamer a délku a kvalitu požadovaného záznamu.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah kvalifikační práce:	**40 - 60 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

Visual C# .NET krok za krokem, ISBN 80-86593-27-4

Vedoucí diplomové práce:	**Ing. Kamil Kosturik, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**5. října 2018**
Termín odevzdání diplomové práce:	**30. května 2019**

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 5. října 2018

Abstrakt

Předkládaná diplomová práce se zabývá komplexním návrhem softwaru prototypu kamerového systému pro vozidla hromadné dopravy. Jedná se zejména o návrh záznamové aplikace pro ukládání příchozích dat z kamer s ohledem na jejich synchronizaci a ochranu použitého záznamového média, aplikace server-klient umožňující získání požadovaných záznamů či jejich částí prostřednictvím klienta, ale práce se zabývá také problematikou kamerových systémů obecně a formáty kódování videa, a to hlavně formátem H.264 a kontejnerem MP4. Dále je zde řešeno praktické sestavení kamerového systému a experimentální ověření požadovaných výsledků.

Klíčová slova

Disk, ethernet, IP, H.264, kamera, kamerový systém, klient, MP4, PoE switch, program, server, software, soket, záznamová aplikace, záznamová jednotka, zdroj.

Abstract

The Diploma Thesis presents a complex design of software of the prototype of the camera system for public transport vehicles. This is especially design of the recording application for storing incoming data from the cameras with regard to their synchronization and protection of the used recording drive, then design of the server-client application which allows to obtain the required records or their parts through the client. But this thesis also deals with CCTV systems and IP cameras generally and with video coding formats, mainly with H.264 format and MP4 container. In the end it gives the practical configuration of the camera system and the experimental verification of the required results.

Key words

Disk, ethernet, IP, H.264, camera, camera system, client, MP4, PoE switch, program, server, software, socket, recording application, recording unit, power supply.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

.....

podpis

V Plzni dne 30.4.2019 Martin Lufinka

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Kamilu Kosturikovi, Ph.D. a odbornému konzultantovi ze Škody Transportation a.s. Ing. Vítovi Veselému za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Seznam symbolů a zkratek

CCD ..... Charged Coupled Device = zařízení s vázaným nábojem
CCTV..... Closed Circuit Television = uzavřený televizní okruh
CPU..... Central Processing Unit = centrální procesorová jednotka
CMOS ..... Complementary Metal–Oxide–Semiconductor
FE..... fotoelektrický jev
fps..... snímková frekvence
FTP..... File Transfer Protocol
DRAM ..... Dynamic Random Access Memory
DSP ..... Digital Signal Processor = digitální signálový procesor
DVR..... Digital Video Recorder = digitální video rekordér
IEEE ..... Institute of Electrical and Electronics Engineers
IP..... Internet Protocol = internetový protokol
JPEG ..... Joint Photographic Experts Group
HD..... High Definition
HDD..... Hard Drive Disk = pevný disk
HTTP ..... Hyper Text Transfer Protocol
LAN ..... Local Area Network
LED..... Light Emitting Diode
MPEG ..... Moving Picture Experts Group
OS ..... operační systém
PoE..... Power over Ethernet = napájení po ethernetu
PTZ ..... pan-tilt-zoom
RAM ..... Random Access Memory
RTP ..... Real Time Procol
SSD ..... Solid-state Drive
TCP ..... Transmission Control Protocol = přenosový kontrolní protokol
TVL..... Television Lines = televizní řádky
UDP ..... User Datagram Protocol
USB..... Universal Serial Bus
UTP..... Unshielded Twisted Pair = nestíněná kroucená dvoulinka
VGA..... Video Graphics Array

Úvod

Práce popisuje projekt zadaný společností Škoda Transportation a.s., oddělením nadřazeného řízení, na jaře roku 2018. Cílem projektu bylo vyvinout řídicí systém, tedy software, pro kamerový systém vozidel hromadné dopravy a vytvořit tak pro společnost vlastní levnější alternativu k dostupným kamerovým systémům od externích společností. Projekt byl od začátku řešen jako izolovaný systém, tedy nezávislý na dalších zařízeních vozidla jako je řídící počítač nebo informační systém.

Text diplomové práce je členěn, kromě úvodu a závěru, do pěti kapitol. První kapitola se zabývá úvodem do problematiky kamerových systémů, je zde uvedena stručná historie vývoje kamer a dohledových systémů a základní popis analogových a digitálních kamer, jejich vlastnosti, rozdělení, využití a vzájemné porovnání. Zmíněny jsou taktéž komunikační protokoly, které kamery využívají. Druhá kapitola, rovněž teoretická, řeší problematiku formátů kódování videa, je zde hlavně rozebrán formát H.264 a multimediální kontejner MP4, který je využit ve vyvíjeném kamerovém systému. Třetí kapitola se věnuje popisu vlastního kamerového systému, nejdříve je nastíněna drážní norma EN50155, kterou musí všechny komponenty (IP kamery, záznamová jednotka a PoE switch) splňovat, a následně jsou jednotlivé komponenty systému rozebrány, je popsána jejich základní funkčnost, vlastnosti a parametry a taktéž je zdůvodněn jejich výběr. Čtvrtá kapitola je věnována již softwaru, zabývá se záznamovou aplikací psanou v programovacím jazyce C v programu Visual Studio Code, která běží automaticky v záznamové jednotce pod operačním systémem na bázi Linux. Kapitola obsahuje podrobný popis celé aplikace, včetně vývojových diagramů. Je zde taktéž popsána konfigurace OS a všechny použité soubory a shell skripty. Aplikace by měla umožňovat synchronní nahrávání video záznamů z devíti IP kamer, ukládání a mazání dat šetrné k záznamovému médiu a monitorování stavu napájení záznamové jednotky a umět reagovat na jeho výpadky a znovuobnovení. V páté kapitole je řešena aplikace server-klient, kdy na záznamové jednotce běží serverový program psaný v jazyce C a umožňuje klientu spuštěnému na notebooku se k serveru připojit přes ethernet a požadovat od něj data.

Po dořešení komunikace záznamové jednotky s řídícím počítačem vozidla a informačním systémem, případně vylepšení klienta na systém používaný společností, by se systém v budoucnu mohl nasadit do kolejových vozidel jako plně funkční systém.

1 Kamerový systém

Úvodní teoretická kapitola se věnuje nejdříve stručnému vývoji a použití různých kamerových systémů, následuje rozdělení kamerových systémů podle jejich funkce na analogové a digitální a popis jejich funkce a dřívějšího a nynějšího použití. U analogových a digitálních kamer jsou taktéž uvedeny jejich výhody a nevýhody a je provedeno jejich stručné porovnání.

1.1 Historie vývoje a použití kamerových systémů

V této podkapitole je rozebrán stručný vývoj a použití různých kamerových systémů od konce 19. století do současnosti. V roce 1880 byly vynalezeny první video kamery, o což se zasloužili Thomas Edison a William Dickson, kteří spolu v roce 1893 předvedli první ukázku pohybujícího se obrazu. Dalším důležitým milníkem byl rok 1939, kdy se pro potřeby moderního válečného úsilí objevily první miniaturní přenosné kamery, jako například Univex 8 mm. V roce 1942 byl v Německu poprvé použit kamerový systém CCTV pro sledování startů raket V2. Roku 1951 byl vynalezen záznam na magnetický pásek – systém VTR (Video Tape Recorder). O pár let později se tato technologie stala komerčně dostupnou a mohla být spárována s CCTV kamerami pro záznam obrazu a jeho přehrání. [2, 9, 20, 22]

Rané CCTV systémy a televizní vybavení měly stejné technologie. Oba kamerové systémy podporovaly černo-bílé video na definované výkony používající NTSC standardy v USA a evropské standardy PAL. V 50. letech se staly dostupnými barevné kamery. Roku 1969 se objevil první systém pro domácí zabezpečení. Od 70. let minulého století, kdy byly představeny videorekordéry pro záznam obrazu a videa na videokazety, a ty se staly běžně dostupné, se CCTV kamery rozšiřovaly i do soukromé sféry. Tento trend pokračoval i v 80. letech. V roce 1976 je vynalezena CCD technologie (bude popsána podrobněji v kapitole 1.3.1) pro snímání obrazu založena na čipu, která tak umožnila snímání obrazu i za horších světelných podmínek, potažmo v noci a umožnila tak sledování 24 hodin denně. [2, 9, 20, 22]

Dalším milníkem kamerových systémů byla 90. léta, kdy se objevil multiplexing, který umožnil zobrazení více video signálů z různých kamer na jednom monitoru. V roce 1992 byla vynalezena první „chůva“ kamera, a to díky rychlému rozvoji kamer, které se zmenšovaly a zároveň umožňovaly vyšší rozlišení. Ve stejném roce byly vynalezeny první digitální IP

kamery, které mohly posílat a přijímat informace skrze počítačovou síť. V roce 1996 představila svojí IP kameru komunikující přes ethernet společnost Axis. S příchodem milénia a postupující digitalizací byly rovněž VCR (Video Cassette Recorder) nahrazeny DVR, díky čemuž se i CCTV systémy staly uživatelsky přívětivějšími. [2, 9, 20, 22]

První IP kamery poskytovaly 4 CIF (704 × 480 pixelů) nebo VGA (640 × 480 pixelů), což bylo srovnatelné se staršími analogovými kamerami. V roce 2002 byla představena první megapixelová IP kamera společností IQinvision a poskytovala tak více než čtyřikrát větší rozlišení než staré VGA kamery. V roce 2004 byly představeny 2 Mpix kamery, a tak už byl rozdíl ve kvalitě oproti starým CCTV markantní. V roce 2014 už bylo na světě použito více IP kamerových systémů než starých CCTV systémů. Objevily se rovněž síťové videorekordéry (NVR), které jsou založeny na kódování a zpracování videa v kamerách a následném vysíláním záznamů do NVR pro ukládání nebo vzdálené sledování. V roce 2015 byly představeny kamerové systémy s vysokým 4K rozlišením. [2, 9, 20, 22]

1.2 Analogové kamery

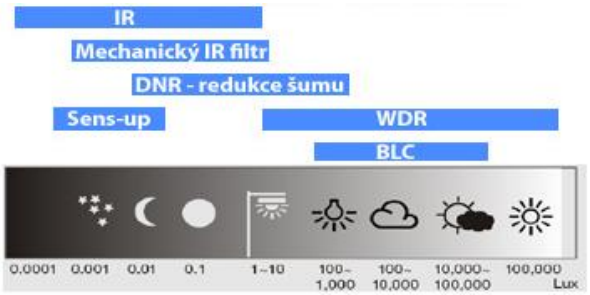
Pro kamerové systémy využívající analogové kamery se vžilo označení CCTV (uzavřené televizní okruhy – k vysílání se nedostane každý veřejně). Tento systém má stále mnohá využití, ať už se jedná o sledování objektů nebo osob či vozidel, monitorování veřejných prostor nebo interiérů budov. U kamer je sledováno několik důležitých parametrů, které mohou hrát roli při výběru správné kamery pro konkrétní aplikaci. Jedná se o rozlišení kamery, citlivost, typ objektivu, způsob přenosu signálu a způsob záznamu. [17]

Pro měření kvality obrazu kamer se nejčastěji používají televizní řádky jako hodnota horizontálního rozlišení. Pomocí nich lze analogové kamery rozdělit do 4 kategorií:

- nízké rozlišení – do 420 TVL,
- střední rozlišení – 420 až 500 TVL,
- vysoké rozlišení – 500 až 600 TVL,
- ultra vysoké rozlišení – 600 až 700 TVL – nejnovější. [17]

Citlivost kamer je parametr, který udává minimální světelné podmínky, za kterých je ještě čip v kameře schopen snímat obraz. Kamery lze podle citlivosti rozdělit na 2 druhy. Prvním jsou kamery se standardní citlivostí (1 lux u barevných, 0,1 Lux u černobílých). Tato

citlivost vyhovuje pro běžné aplikace za denního nebo dostatečného umělého světla. Oproti tomu existují kamery ultracitlivé s vysokou citlivostí (0,01 Lux u barevných a u 0,001 Lux u černobílých), které jsou vhodné pro snímání za snížené viditelnosti – za šera a v noci. Aby se daly výhody barevných i černobílých kamer spojit, byly vyvinuty kamery pracující v režimu den/noc. Čip přes den pracuje v barevném režimu a v noci v režimu černobílého snímání s ultravysokou citlivostí. Ráno se kamera opět přepne do režimu den, čehož se využívá hlavně pro nepřetržité sledování. Pokud je potřeba snímat nějaký objekt za velmi špatné světelnosti, některé kamery disponují možností IR přisvícení pomocí IR diod integrovaných do těla kamery, které se mohou automaticky spouštět. Citlivost kamer je možno zvýšit obrazovou integrací (SENS UP) - digitální zvýšení citlivosti snímacího prvku CCD prodloužením doby integrace náboje v čipu (citlivost v režimu den/noc až 0,00045 Lux). Přehled funkcí kamer při různém osvětlení je zobrazen na *Obr. 1.1*. [17]



Obr. 1.1: Přehled funkcí kamer při různém osvětlení [17]

Objektivy tvoří volitelné příslušenství většiny kamer. Platí pravidlo, že pokud máme kvalitní kameru, měli bychom mít i kvalitní objektiv, abychom nedegradovali vlastnosti kamery (obrazový výkon). Používají se převážně objektivy pevné, s manuální/automatickou clonou, varifokální s manuální/automatickou clonou, mikroobjektivy a Mpix objektivy. [17]

Pro přenos analogového signálu z kamer se využívají běžné metalické koaxiální kabely, optická vlákna nebo kroucené dvoulinky sdělovacích kabelů (UTP 5). Jsou umožněny i mikrovlnné přenosy signálu. Pro záznam z kamer se mohou používat jednoúčelová zařízení pouze pro záznam z analogových kamer, jako jsou třeba DVR, které obsahují pevný HDD disk, nebo se využívají počítače se speciální rozšiřovací kartou pro připojení kamer. Moderní systémy jsou hybridní a umožňují připojení analogových i digitálních IP kamer a systém lze obměňovat a rozšiřovat. Většina videorekordérů je vybavena videoserverem, díky čemuž lze sledovat živý obraz přes internet nebo systém konfigurovat. [17, 29]

1.2.1 Typy kamer, výhody a nevýhody

Kamery se vyrábějí se snímacím CCD čipem v barevném nebo černobílém provedení. Rozlišujeme několik typů kamer podle provedení. Nejčastěji se jedná o standardní kamery s CS-závitem pro výměnný objektiv, kompaktní kamery s vestavěným objektivem, dome-kamery pro montáž na strop a další. [15]

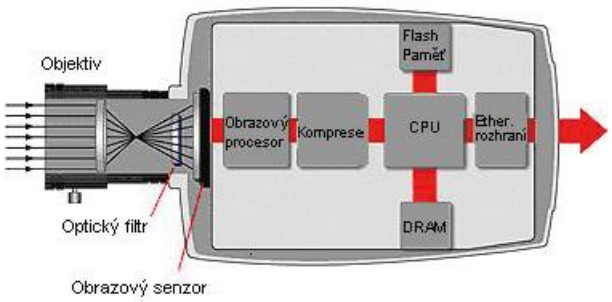
Výhodou analogových kamerových systémů je nesporně mnoho dostupných typů kamer a jejich kompaktní rozměry, levné komponenty a příslušenství, možnost dlouhých přenosových tras, hybridní záznamové jednotky pro analogové i IP kamery, velmi kvalitní obraz v noci, u malých systémů levné pořizovací náklady a snadná obsluha. Oproti tomu je zde však značné množství nevýhod. Jedná se o zastaralou technologii s omezeným rozlišením, u velkých systémů jsou vysoké pořizovací náklady a náročná instalace, chybí zde možnost záznamu zvuku integrovaná v kamerách (nutný zvláštní kabel), možnost rušení z okolních sítí, při výpadku záznamové jednotky dojde k výpadku celého systému. [15]

1.3 IP kamery

V této kapitole jsou probrány digitální IP kamery a kamerové systémy a to detailněji, protože v navrhovaném kamerovém systému pro vozidla hromadné dopravy jsou využity právě IP kamery. Je zde popsána jejich vnitřní struktura, komponenty, princip činnosti, dále jsou zběžně probrány CCD a CMOS snímače, dále pak typické vlastnosti, výhody a nevýhody, porovnání s analogovými kamerami, typy IP kamer, a nakonec jsou rozebrány využívané přenosové linky a komunikační protokoly.

Jak je zmíněno v předchozí kapitole, analogové kamerové systémy jsou stále hojně využívané, ale zároveň jsou i pomalu na ústupu a pokračujícím trendem je digitalizace a využívání digitálních kamerových systémů. Hlavním cílem digitalizace je dosažení bezproblémové integrace jednotlivých systémů do větších celků. Kamery těchto digitálních sledovacích systémů se nazývají síťové kamery, webkamery nebo IP kamery podle komunikačního protokolu, který využívají. Hlavní rozdíl oproti analogovým kamerám tkví v používaných komunikačních protokolech, způsobu zpracování video signálu a možnosti zapojení inteligentních analýz. [28, 29]

Následující odstavce jsou zaměřeny na konstrukční provedení a princip fungování IP kamer. V podstatě se jedná o integraci kamery a řídícího počítače. Hlavními komponenty jsou objektiv, obrazový senzor, řídící procesor, paměti a komunikační rozhraní. Dále mohou obsahovat příslušenství, které umožňuje například použití kamery ve venkovním nebo nehostinném prostředí, zabraňuje poškození nebo poskytuje nějaké dodatečné speciální funkce. Vnitřní schéma IP kamery je zobrazeno na *Obr. 1.2* pod tímto odstavcem. [10, 28]



Obr. 1.2: Vnitřní schéma IP kamery [10]

Skrze objektiv a optický filtr dopadá světlo o různých vlnových délkách na snímací senzor, což může být CCD nebo CMOS čip. Když světlo prochází skrze objektiv, což je optická soustava, dochází k tzv. vykreslování scény, což znamená přizpůsobení obrazu dané aplikaci. Tuto kreslící operaci objektivu lze popsat modulační přenosovou funkcí, která popisuje zkreslení dopadajícího světla vzhledem k clonovým číslům a ohniskovým vzdálenostem objektivu. Za objektivem se nachází optický IR filtr, který zajistí průchod světla pouze o vyžadovaných vlnových délkách. Čip informaci o dopadajícím světle transformuje na elektrický náboj, který je akumulován ve světločivných buňkách. Výsledkem je analogový signál (CCD čip), který je pomocí AD převodníku převeden na digitální signál, nebo rovnou digitální signál (CMOS čip), který je posléze odeslán do řídícího obrazového procesoru DSP, který zajišťuje zpracování signálu v digitální podobě s využitím funkcí pro zlepšení kvality výsledného obrazu. Do zpracování je možno zakomponovat i funkce inteligentní analýzy obrazu nebo detekce pohybu. Podle používaného kompresního algoritmu je pak obraz zkomprimován zejména pro snížení potřebné kapacity úložišť a potřebné šířky pásma pro další přenos. Dále následuje centrální řídící počítač CPU, operační dynamická paměť DRAM a Flash paměť. CPU společně s paměťmi obstarávají komunikaci s ostatními zařízeními a přenos dat. CPU taktéž zprostředkovává všechny operace dějící se v kameře (ovládání, nastavení). Signál je poté v digitální podobě odeslán skrze komunikační rozhraní. [10, 28]

1.3.1 CCD vs. CMOS

Oba snímací senzory jsou založeny na fotoelektrickém jevu, kdy v některých látkách způsobuje dopad světla zvýšení energie atomů, a tak vznik páru elektron-díra. Po přiložení napětí dojde k pohybu těchto nosičů, a tak vzniku elektrického proudu. Oba senzory jsou založeny na vnitřním fotoelektrickém jevu, kdy dochází k uvolňování elektronů uvnitř materiálu, což vede ke změně elektrické vodivosti, přičemž množství uvolněných elektronů je dáno intenzitou a spektrem dopadajícího záření.

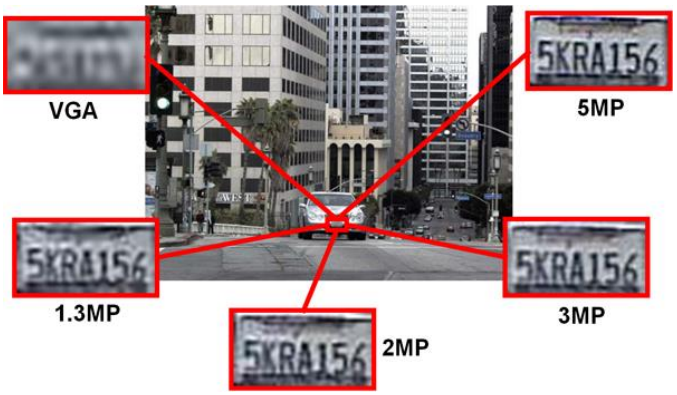
Zkratka CCD pochází z anglického Charged Coupled Device (prvek s vázaným nábojem). Snímací senzor je založen na technologii MOS a funguje na principu vytváření náboje v závislosti na míře osvětlení a následném posunu náboje pro zpracování jako u posuvného registru. Přiložené kladné napětí vyvolá oddálení minoritních nosičů (děr) a vznik potenciálové jámy. To této jámy jsou přesunuty elektrony vzniklé při vnitřním FE jevu. Náboj je po vzniku nutné přesunout pro zpracování, což je prováděno postupným přeléváním náboje, čehož se dosáhne postupnou změnou napětí. Náboj se takto přesune až k poslední elektrodě, kde probíhá samotné zpracování. Přelévání je dvou, tří nebo čtyř fázové. [24]

Technologie CMOS (Complementary Metal–Oxide–Semiconductor) využívá stejně jako CCD vnitřní FE jev. Následně dochází k plnění nábojů do potenciálových jam. Odvod nábojů je realizován paralelně z každé buňky zvlášť. Nejvíce se využívají 2 typy – APS CMOS (Active Pixel Sensor), jehož princip spočívá v doplnění jednotlivých buněk snímače o řízení, což vede k omezení šumu a lepší citlivosti snímače. Druhou variantou je FF CMOS (Full Frame). Oproti CCD snímačům mají CMOS snímače nižší spotřebu, nižší produkci tepla (šum) a větší rychlost přenosu ze snímačů větších rozměrů. [24]

1.3.2 Vlastnosti IP kamer, porovnání s analogovými kamerami

V této podkapitole jsou probrány vlastnosti IP kamer, je provedeno porovnání s analogovými kamerami, určení výhod a nevýhod. První vlastností je rozlišení obrazu. To určuje, kolika obrazovými body je tvořen výsledný obraz. Je zřejmé, že vyšší rozlišení přináší kvalitnější a ostřejší obraz poskytující více detailů. U analogových kamer již bylo v rozlišení dosaženo maxima – 0,7 Mpix. U IP kamer je oproti tomu rozlišení stále navyšováno, běžné

jsou kamery s rozlišením 1,3 až 2 Mpix. Špičkové kamery ale mohou mít rozlišení i 10krát vyšší. Porovnání rozlišení a kvality obrazu různých kamer je vidět na *Obr. 1.3* níže. [16, 29]



Obr. 1.3: Porovnání kamer s různým rozlišením a kvalitou obrazu [16]

Dalším důležitým aspektem je citlivost, která udává, při jaké nejnižší intenzitě osvětlení je kamera schopna snímat a poskytovat čistý a kvalitní obraz. V této vlastnosti IP kamery obecně za analogovými zaostávají, což je dáno hlavně fyzikálními vlastnostmi – když porovnáme kamery se stejnou velikostí snímacího čipu, je zřejmé že u IP kamer, které mají vyšší rozlišení a tím pádem musí mít na čipu více bodů, na 1 čip potom dopadá méně světla než u čipů analogových kamer. Snímková frekvence (fps) určuje, kolik snímků za 1 sekundu je kamera schopna poskytnout. Čím je fps větší, tím je obraz plynulejší a méně trhaný. Limitní hranicí, při které lidské oko už nevnímá obraz jako nespojitý, je přibližně 15 fps. Analogové i IP kamery běžně poskytují obraz 25 fps, což je u analogových kamer zároveň i maximální hodnota, oproti tomu IP kamery jsou schopné poskytovat fps vyšší (až 60 fps). [16]

Schopnost kamery detekovat změny v obraze a informovat o tom uživatele či další zařízení mají jak analogové, tak IP kamery, které ale předčí analogové v možnosti inteligentní analýzy. Další vlastností, kterou lze porovnávat, je úroveň zabezpečení. Zde se již nejedná o vlastnost samotné kamery, ale celého analogového/digitálního systému. Na zabezpečení se lze dívat ze 3 hledisek. Nejdříve se jedná o zabezpečení přenosové cesty mezi kamerou a záznamovým zařízením a nemožností se nabourat do přenosové trasy. U analogových systémů je jedinou možností ochrany fyzická ochrana kabeláže, na druhou stranu u digitálních systémů lze přenos chránit šifrováním komunikace. Druhým hlediskem je ochrana samotného záznamu. U obou systémů jsou uložená data chráněna vodoznakem proti pozměnění záznamu. Proti neoprávněnému přečtení jsou více chráněna data digitálních systémů, protože

jejich úložiště mohou být šifrována. Posledním hlediskem je ochrana mezi záznamovou jednotkou a místem přehrávání záznamu. Zde je úroveň zabezpečení obou systému stejná. Souhrnné porovnání vlastností obou systémů je vidět v *Tab. 1*. [10, 16]

Tab. 1: Porovnání vlastností analogových a digitálních kamerových systémů. [16]

Vlastnost	Analogový systém	Digitální systém
Rozlišení kamer	Nižší (do 0,7 Mpix)	Vyšší (standardně 1 až 2 Mpix)
Citlivost kamer	Vyšší	Nižší
Snímková frekvence	25 fps	Až 60 fps
Detekce pohybu v obraze	Ano	Ano
Inteligentní analýza	Ne	Ano
Standardizace	Vyšší	Nižší
Úroveň zabezpečení	Nižší	Vyšší
Nároky na diskovou kapacitu	Nižší	Vyšší
Kabeláž	Vyhrazená	Sdílená
Pořizovací cena	Nižší	Vyšší

1.3.3 Typy IP kamer

IP kamer existuje celá řada různých druhů a typů. V této podkapitole je předloženo možné dělení a strukturalizace kamer. Lze determinovat 2 základní druhy – fixní a PTZ IP kamery. Do kategorie fixních kamer spadají fixní dome IP kamery. PTZ IP kamery lze rozdělit na 3 podkategorie – mechanické, nemechanické a PTZ IP dome kamery. [19, 28]

Fixní kamery mají pevně nastavený směr natočení a nelze toto nastavení dálkově měnit a ovládat. Jedná se tedy o tradiční kameru, u které lze vybírat ze široké škály objektivů od obyčejných, přes širokoúhlé po teleobjektivy, což zajišťuje všestranné použití, vhodné jsou třeba pro venkovní aplikace. Mají možnost i přidělání ochranných krytů. [19, 28]

- ***Fixní dome IP kamery***

Jedná se o fixní kamery, které jsou již v základním vybavení opatřeny ochranným dome krytem. Kryt je z neprůhledného materiálu, takže nelze jednoduše pozorovat, jaký prostor daná kamera sleduje. Nevýhodou je nemožnost výměny objektivu, přičemž tento problém je zmírněn použitím objektivu s proměnnou ohniskovou vzdáleností. [19, 28]

PTZ IP kamery mají svůj název odvozený z anglických slov Pan (panorámovat – pohyb po horizontální ose), Tilt (náklon – pohyb po vertikální ose) a Zoom (zvětšení). Mají možnost

využít všechny zmíněné polohovací mechanismy (automaticky či manuálně), pro přesné nastavení kamery pro danou aplikaci. Díky tomu jsou velice variabilní a lze je využít v široké škále aplikací. Tyto kamery jsou používané v pokrokových dohledových systémech, protože umožňují automatické polohování na základě inteligentní analýzy a díky tomu mohou efektivně pokrýt co největší prostor. [19, 28]

- ***Mechanické PTZ IP kamery***

Obsluha těchto kamer je zajištěna operátorem a vhodné jsou tak zejména do vnitřních prostor. Oproti nemechanickým PTZ či dome PTZ kamerám nemají žádné výhody, naopak mají horší dynamiku. [19, 28]

- ***Nemechanické PTZ IP kamery***

Tyto kamery mají díky absenci mechanických prvků neslyšně tichý chod a prostorově nenáročné provedení, jsou diskrétnější. Obvykle jsou využívané s širokoúhlými objektivy a megapixelovými snímacími senzory. Jsou schopné pokrýt i díky zoomu velký prostor při zachování schopnosti detailnosti. Oproti tomu mají nevýhodu v omezeném pohybu. [19, 28]

- ***PTZ IP dome kamery***

Jedná se o nejlepší typ IP kamer, umožňují neomezený pohyb ve všech osách pohybu, což v kombinaci s diskrétním dome krytem, který neumožňuje rozpoznat směr natočení, vede k možnosti monitoringu velkého prostoru v pravidelných intervalech s nemožností určení okamžitého sledovaného prostoru. Jsou vhodné pro vnitřní i vnější aplikace. [19, 28]

1.3.4 Komunikace kamer, přenosová média

Komunikace IP je kamer je ta hlavní část, kterou se odlišují od analogových kamer. IP kamera má vlastní IP adresu pro přístup na síť a vlastní hardware realizující síťovou komunikaci zabudovaný v jednotce kamery. Kamera může mít různé konektory pro připojení média zprostředkovávajícího datový přenos. Může se jednat o konektor pro připojení koaxiálního nebo UTP kabelu, konektor pro připojení antény, konektor pro napájení nebo konektor pro připojení mikrofonu. Velikou výhodou je možnost přenosu videa, programování funkcí, nastavování kamery, napájení a ovládání přes jedno rozhraní. [19, 28]

Řídící počítač obsahuje veškerý software pro přístup na webový sever, FTP server, emailový klient a další. IP protokol je dnes nejvyužívanějším počítačovým komunikačním protokolem, a to zejména pro jeho škálovatelnost, což znamená, že je stejně dobře využitelný pro velmi malé i velmi velké instalace, a proto jeho využití ke komunikaci s kamerami dává

největší smysl. Na většině míst se dnes vyskytují lokální drátové LAN sítě či bezdrátové WLAN sítě komunikující pomocí TCP/IP protokolu, proto je instalace kamerového systému velmi jednoduchá. Pro připojení jednotlivých IP kamer se využívá zpravidla standardu 100BASE-T (100 Mbit/s) prostřednictvím kroucené dvojlinky. Pokud chceme připojit více kamer, k tomu pak poslouží síťový přepínač (switch), který taktéž zajišťuje kompatibilitu se staršími standardy jako je 10BASE-T, případně pokud mají kamery možnost napájení po ethernetu, je nutno využít PoE switch. Od switche dál je často nutná vyšší přenosová rychlost, a tak je nutno využít jiného standardu a někdy i přenosového média, než je kroucená dvoulinka. Jedná se o standard 1000BASE-T (SX, LX, LH) s přenosovou rychlostí 1 Gbit/s, přenosovým médiem je kroucená dvoulinka (1000BASE-T) nebo optické vlákno 1000BASE-SX (LX, LH). Pokud nechceme nebo v daném místě není možné použít fyzická média pro přenos, jako vhodná alternativa se jeví využití bezdrátového přenosu dat. V tomto případně je nutno využít bezdrátového přístupového bodu, který je drátově spojen se switchem a prostřednictvím bezdrátového rozhraní s ostatními prvky sítě. Sítě založené na TCP/IP protokolu podléhají standardu IEEE 802.11, který má několik specifikací. Je to standard 802.11a, který používá pásmo 5 GHz a poskytuje rychlost až 54 Mbit/s do vzdálenosti 30 m ve vnitřním prostředí. Nejvíce používaným je však standard 802.11b pracující v pásmu 2,4 GHz a poskytující rychlost 11 Mbit/s na vzdálenost 100 m i ve venkovním prostředí. Třetím využívaným standardem je 802.11g, který využívá pásma 2,4 GHz a má vyšší přenosovou rychlost (54 Mbit/s) než předchozí na stejnou vzdálenost. [19, 28]

1.4 Komunikační protokoly pro přenos dat

V této kapitole jsou zmíněny a popsány dva nejhojněji používané komunikační transportní protokoly IP kamer. Jedná se o TCP a UDP protokol. TCP je zkratka pro Transmission Control Protocol (přenosový kontrolní protokol), který je založen na připojení a dokáže rozdělit velké objemy dat do menších paketů a poskytuje spolehlivý přenosový kanál. Oproti tomu UDP (User Datagram Protocol) není založen na připojení a negarantuje doručení zaslaných dat, neobsahuje kontrolní mechanismy. IP protokol zajišťuje komunikaci v síťové vrstvě, TCP a UDP se starají o pokrytí transportní vrstvy modelu ISO-OSI. [25]

TCP protokol vznikl v roce 1974, tedy ještě před vznikem referenčního komunikačního modelu ISO-OSI (1977), a proto ho nepoužívá, ale využívá protokolový standard DoD, který má pouze 4 vrstvy. Jedná se o protokol transportní vrstvy. Hlavním úkolem je vytváření,

rušení a údržba transportních spojení, které dále využívají aplikační procesy pro komunikaci. Při komunikaci je vytvořen plně duplexní virtuální okruh. Používá se v tom případě, kdy je hlavním požadavkem konzistence a nepřerušenost dat, nikoliv rychlost přenosu. Je zabalen do IP paketu s typovým číslem PROT = 6. [25]

Transportní TCP protokol zároveň funguje jako nosič pro další běžně používané protokoly aplikační vrstvy na určitých portech. Jedná se o FTP protokol na portu 21, který slouží k přenosu souborů přes internet/intranet, čehož lze u kamerových systémů využít pro přenos záběrů nebo videa z kamer na FTP server nebo do aplikace. Dalším používaným protokolem je SMTP (Send Mail Transfer Protocol) na portu 25, který slouží pro odesílání emailů, díky čemuž může kamera posílat například upozornění pomocí emailového klienta. Následuje HTTP na portu 80, který se používá pro prohlížení webových stránek. Zároveň slouží k nejběžnějšímu způsobu přenosu videa z kamery/video serveru. Možno je také použít HTTPS (HTTP přes Secure Socket Layer) na portu 443, který umožňuje zabezpečený přístup k webovým stránkám, potažmo i zabezpečený přenos videa. Nakonec to může být RTP nebo jeho odvozené varianty, který má standardizovaný formát paketů pro poskytování videa a zvuku přes internet. Je to běžný způsob přenosu videa ve formátu MPEG (typicky na portu 554 přes RTSP). Tento protokol může být navázán i na UDP protokol. [25]

Stejně jako TCP protokol, je i UDP protokol protokolem transportní vrstvy. Jedná se o nespojovanou službu, což znamená, že po odeslání dat se již odesílatel nestará, zdali data v pořádku dorazila příjemci nebo ne (o to se stará protokol aplikační vrstvy). Používá se v aplikacích, které vyžadují spojitý datový tok nehledě na občasné výpadky dat s rychlým nezabezpečeným přenosem bez potvrzování příjmu. Je zabalen do IP paketu s typovým číslem PROT = 17. [25]

2 Formáty videa

Nejprve jsou v této kapitole uvedeny důležité pojmy, které souvisejí s přenosem a komprimací video formátu. Proč je vlastně komprese video signálu tak důležitá a nezbytná? Odpověď je jednoduchá. Kdyby komprese neexistovala, byla by potřeba enormní velikost datového toku a enormní velikost úložišť, a to jednoduše není možné. Signál obrazu se skládá z jednotlivých bodů – pixelů, které jsou zase složeny ze 3 subpixelů, které odpovídají základním barvám – červené, zelené a modré. Celé video je potom složeno z jednotlivých snímků jdoucích za sebou. Počet snímků za vteřinu se v angličtině označuje jako Frame rate, odkud je pak i zkratka fps. Typickými hodnotami jsou 24, 25 a 30 fps. Dalším důležitým údajem u formátu videa je rozlišení, které udává počet pixelů v obrazu, a poměr stran, což značí rozložení pixelů mezi sloupci a řádky. Historicky nejrozšířenějším poměrem stran byl poměr 4:3. V dnešní době je nejrozšířenějším poměrem stran poměr 16:9 a rozlišení HD 1280 × 720 pixelů, nebo Full HD – 1920 × 1280 pixelů nebo nejnověji rozlišení 4K UHD. [1]

Dalším důležitým údajem u video signálu je datový tok, který udává množství dat potřebné pro 1 vteřinu videa v Mbit/s. Pro video signál je rovněž důležitý YUV model, písmeno Y značí jasovou složku, písmena U a V udávají barevnou složku. Model byl vytvořen pro kompatibilitu černobílého a barevného vysílání. Je založen na principu, že 3 barevné složky lze vyjádřit jako 2 barvené a 1 jasovou složku. Formát je vyjádřen číslem, které udává počet bitů na jednu složku a 3 čísly, které reprezentují informaci o vzorkování. [1]

Datový soubor, ve kterém je zapouzdřen komprimovaný video signál společně s audio signálem, se nazývá multimediální kontejner. Lze takto uložit více různých signálů do jednoho souboru (například více zvukových stop a titulků) a je zajištěna vzájemná synchronizace. Různé kontejnery se liší podle jejich schopností pojmout různé typy multimediálních dat. Kontejner sám o sobě neříká nic o použité kompresi video a audio signálu, ta je dána použitým kodekem. Kontejner MP4 je probrán v kapitole 2.2. [1]

2.1 Formáty kódování videa

Formátů pro kódování videa je mnoho, zde jsou proto vyjmenovány pouze ty nejznámější a nejpoužívanější typy. Jedná se například o MPEG-2 Part 2, MJPEG, Xvid, Divx Codec,

H.264 (H.265), WMV9, Theora, Dirac, Vc-1 a další. V síťových kamerách se nejvíce používá MJPEG nebo H.264 a tak jsou dále probrány podrobněji. [13]

2.1.1 MJPEG

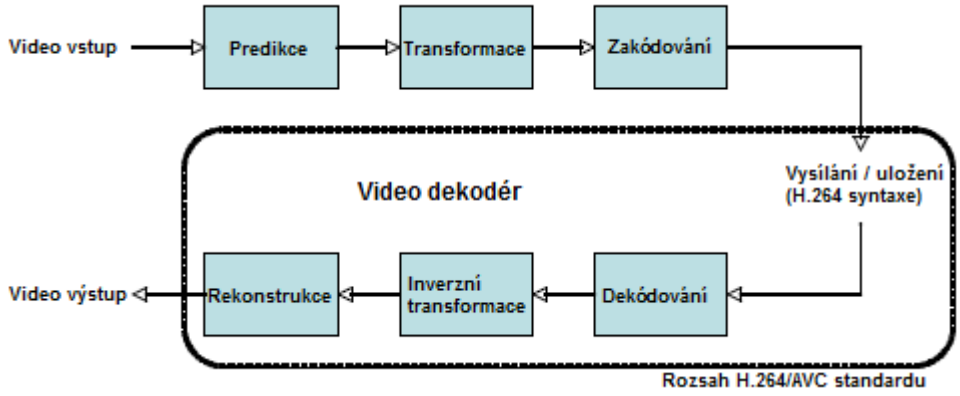
Motion JPEG je video kodek, který spočívá v kódování každého snímku zvlášť, kdy je každý snímek odděleně komprimován do JPEG obrazu. Z toho vyplývá nezávislost výsledné kvality video komprese na množství pohybu v obrazu (na rozdíl od MPEG, kde velké množství pohybu může způsobit pokles kvality obrazu). Každý snímek je zde klíčový, což má své výhody i nevýhody. Výhodou je rychlý posun ve videu. Nevýhodou je větší velikost výsledného videa. Sám o sobě nemá oficiální standard, což může vést k nekompatibilitě jednotlivých kodeků. Ale implementuje ho několik knihoven, například knihovna libavcodec z frameworku ffmpeg. [23]

Při dostupnosti pouze malé šířky pásma je priorita poskytována na zachování kvality obrazu, což může vést ale k zahození nějakých obrázků. Výhodou je minimální zpoždění při zpracování obrazu. Jednotlivé snímky mají konzistentní velikost. Kamera dokáže zachytit a komprimovat určitý počet snímků za vteřinu a poté z nich vytvoří spojitý tok obrazů a odešle je přes síť na zobrazovací stanici. Všechny snímky mají stejnou garantovanou kvalitu obrazu určenou úrovní komprese pro danou kameru nebo video server. Tato technologie je jednodušší, a tak je nižší i cena kamery/video serveru. Využití tohoto způsobu kódování videa také zmenšuje zpoždění zvuku a tím je jednodušší synchronizace videa a zvuku. [23]

2.1.2 H.264

H.264 (někdy též zvaný MPEG-4 part 10) je průmyslový standard pro kódování a kompresi videa, jehož finální verze byla vytvořena v roce 2003 a někdy se nazývá MPEG-4 AVC (advanced video coding) a je součástí sady standardů MPEG-4, ze které toho hodně přebírá a přidává k tomu různá vylepšení. Je tak zachováno mezisnímkové prediktivní kódování, omezení mezibodové redundance, výstupní bezeztrátové kódování nebo omezení redundance predikční chyby. Jeho hlavní pointou je přenášení obrazu ve vyšší kvalitě při nižší přenosové rychlosti a nižším objemu dat. Je hojně užívaný v širokém spektru oblastí, lze zmínit například Blue-Ray disky, mobilní TV, DVD-Video nebo internetové streamování, a je tak nejpoužívanějším standardem současnosti. Byl vyvinut v kooperaci dvou skupin ITU-T Video Coding Experts Group a ISO/IEC JTC1 Moving Picture Experts Group. [1, 21]

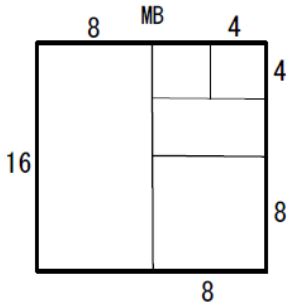
V praxi nejprve video kodér konvertuje video do komprimovaného formátu a poté ho opět dekodér dekóduje zpět do nekomprimovaného formátu. Veškerá standardizace je v dokumentu vydaném ITU-T a ISO/IEC a je zde definován formát pro komprimované video a metody pro dekódování této syntaxe k vytvoření zobrazitelné video sekvence. Dokument přímo nespecifikuje, jak komprimovat video signál – to je ponecháno výrobcům kodérů, prakticky se ale jedná o zrcadlový proces k dekódování. Na *Obr. 2.1* pod tímto odstavcem je zobrazen blokový diagram kódování a dekódování videa a je tam vyznačeno, ve kterých částech procesu se uplatňuje H.264 standard. [3]



Obr. 2.1: Proces kódování a dekódování videa formátu H.264

Kódování videa ve formátu H.264 se skládá ze 3 částí. Vstupní video signál je složen z jednotlivých snímků. Úkolem kodéru je předpovídání, transformace a kódování a vytvoření komprimovaného datového toku. Kodér rozdělí jednotlivé snímky do skupin snímků GOP (Group of Picture). Počet snímků ve skupině není pevně daný. Základním prvkem ve skupině je I-Frame (Intra-Frame), který je kódovaný pomocí formátu JPEG. Dalšími typy snímků ve skupině jsou P-Frames, které udávají změny I-Frames, a B-Frames. Kodér zpracovává video snímky v jednotkách predikovaných makrobloků o velikosti 16 × 16 pixelů (u H.264 mohou mít nově makrobloky i jiné velikosti), přičemž H.264 umožňuje rovněž i dělení makrobloku 16 × 16 na makro-partions, což je zobrazeno na *Obr. 2.2*. Makrobloky se vytvářejí na základě predikce – buď je to intrapredikce, která využívá již zakódovaných a použitých okolních pixelů v rámci daného snímku, využívá bloky o velikosti 16 × 16 nebo 4 × 4 pixelů, nebo je to interpredikce, která využívá předchozí zakódované celé snímky, využívá bloky o velikosti od 16 × 16 po 4 × 4 pixelů. Predikce může využívat různých technik odhadu pohybu, a hlavně pohybové vektory, které pomocí souřadnic udávají přesnou polohu makrobloku. Predikce vektorů vychází z existující korelace mezi vektory sousedních bloků a samotným vektorem.

Poté se kódují jen tyto souřadnice, které mají přesnost až na ¼ pixelu, což vede k velké úspoře objemu dat. [1, 3, 7, 13, 21]



Obr. 2.2: Dělení makrobloku formátu H.264 [21]

Následně dojde k odečtení predikovaného makrobloku od aktuálně zpracovávaného makrobloku, což se nazývá kompenzace pohybu a výsledkem je reziduální makroblok, který je posléze transformován s využitím integrální transformace. Výsledkem transformace je množina koeficientů, z nichž každý je váhovou hodnotou pro standardní bázi. Toho se využívá při dekódování u inverzní transformace – podle vah lze sestavit obrazový blok. Transformační koeficienty jsou následně kvantovány, tj. jsou děleny celočíselnými hodnotami dle kvantizačního kroku. Větší kvantizační krok způsobí více nulových koeficientů a tím i větší kompresi, ovšem na úkor špatné kvality dekódovaného obrazu. Nižší kvantizační krok vede k nižší kompresi, ale k větší kvalitě. [3, 7, 13]

Tento proces vytváří celou řadu hodnot, které musí být ve výsledném bitovém proudu obsaženy. Jsou to kvantované transformační koeficienty, informace umožňují dekodéru znovu vytvořit predikci, informace o struktuře komprimovaných dat a kompresních nástrojích použitých při kódování a informace o kompletní sekvenci videa. Všechny tyto hodnoty a parametry jsou převedeny na binární kódy, a to pomocí aritmetického kódování nebo kódování s proměnnou délkou. Kódovaný bitový tok pak může být uložen, anebo přenášen. Zároveň s transformací a kódováním probíhá ve zpětné větvi kodéru inverzní transformace, kdy je výsledný residuální makroblok přičten predikovanému a uchován pro další predikce. P-snímky mají jiný způsob kódování, protože nesou méně informace – udávají pouze změny I-snímků. B-snímky fungují obdobně – udávají změnu oproti I nebo P-snímku. [3, 7, 13]

Při dekódování video dekodér přijímá komprimovaný bitový tok ve formátu H.264 a dekóduje každý ze syntaktických prvků a extrahuje všechny potřebné informace uvedené

výše. Tyto informace se následně využívají pro zvrácení procesu kódování a obnovení sekvence obrazů. Inverzní transformace kombinuje standardní vzory založené na změněných koeficientech k znovuvytvoření bloků zbytkových dat. Tyto bloky jsou spojeny dohromady a vytvoří residuální makroblok. Pro každý makroblok dekodér vytváří stejnou predikci jako byla ta vytvořená kodérem. Dekodér přidává předpověď k dekódovanému zbytku pro rekonstrukci dekódovaného makrobloku, který pak může být zobrazen jako součást video rámce. [3, 7]

Standard H.264 obsahuje také různé nástroje a postupy pro kódování videa. Konkrétní sada definic se nazývá profil. Existují 4 profily, spolu se kterými jsou definovány úrovně, které určují parametry videa, které je dekodér ještě schopen zpracovat. Těchto úrovní je pět, ale každá může mít ještě mezikroky. Tyto parametry jsou například maximální bitová rychlost, maximální velikost snímku, či počet makrobloků za sekundu. Prvním profilem je základní profil Baseline, který je určen pro přenos videa v reálném čase. Obsahuje pouze I a P-snímky. Druhým profilem je rozšířený profil (Extended), který se podobá tomu základnímu, akorát má dokonalejší přenos. Třetím profilem je profil hlavní (Main), který je určený pro DVBT vysílání a DVD přehrávače. Používá lepší, ovšem výpočetně náročnější kódování. Tento profil také kromě I, P a B-snímků obsahuje ještě SP a SI-snímky (switching P/I), které slouží ke snížení datového toku při použití přepínání mezi proudy nebo pro zprostředkování náhodného přístupu. Poslední je vysoký profil (High), který klade důraz na kvalitu výsledného obrazu za cenu vyšší HW náročnosti dekodéru nebo delší doby kódování. [14, 21]

2.2 Multimediální kontejner MP4

Multimediální kontejnery představují formát pro uložení několika proudů multimediálních dat do jednoho souboru. To znamená, že do jednoho souboru lze uložit třeba jednu video stopu, několik audio stop a několik stop titulků. Je zajištěna jejich vzájemná synchronizace. Různé kontejnery se vzájemně liší podle schopnosti pojmout multimediální data. Pro přehrávání jednotlivých kontejnerů se používají demuxery (splittery), které rozdělí datové proudy do různých kodeků a následně do výstupních zařízení. Kontejner sám o sobě neříká nic o vnitřní kompresi dat, která je určena použitým kodekem. Některé kontejnery mohou mít v sobě uloženy pouze určité formáty, jiné jich mohou mít uloženo více. Mezi nejpoužívanější multimediální kontejnery patří formáty AVI (Audio Video Interleave),

MPEG Program Stream, MPEG Transport Stream, OGG, MP4, Flash Video, MOV, Matroška, WemM nebo VOB. Pro streamování přes internet se nejvíce používá kontejner MP4 (MPEG-4 part 14), a tak je popsán dále. [8, 18]

MP4 multimediální kontejner, též známý pod názvem MPEG-4 part 14, je definovaným standardem ISO/IEC 14496-14:2003, z čehož vyplývá, že je součástí standardu MPEG-4, a tudíž umožňuje využití kompresních formátů MPEG-1, MPEG-2 a MPEG-4 pro video a MP3 a AAC pro zvuk. Jeho jedinou oficiální modifikací je kontejner 3GPP (a jeho další verze), který používá pro obraz kompresi H.263 a AMR kompresi pro zvuk. Povoluje kombinovat digitální video, zvuk a data do jednoho souboru. Je navržen k pojmutí synchronizovaných mediálních informací ve flexibilním formátu pro účely výměny, správy a úpravy multimediálních souborů. Kontejner je identický s kontejnerem MOV od Apple. Standardní příponou je MP4, ale mohou se objevit i další, jako jsou M4A pro nezašifrované zvukové záznamy a M4P pro zašifrované zvukové soubory. [4, 8]

Umožňuje také streamování přes internet, díky čemuž je oblíbený a hojně používaný, navíc ho podporuje většina chytrých telefonů. Používá se tak pro uchování digitálního obrazu a zvuku na webových stránkách. Uložený obsah může být sledován v reálném čase. Dokáže poskytnou kompresní poměr 4 MB/min, který je srovnatelný s DVD kvalitou. Dnes je často používán i pro ukládání videa v HD kvalitě v digitálních fotoaparátech a kamerách. [4, 8]

3 Vlastní kamerový systém

Cílem této diplomové práce je vytvořit řídící software pro kamerový systém vozidel hromadné dopravy. Aby bylo možno tento úkol realizovat, bylo nejdříve nutno si ujasnit, jaké komponenty takový kamerový systém obsahuje, jaké jsou na ně kladeny požadavky, dále tyto komponenty sehnat, sestavit z nich zkušební systém a na něm vyvíjet a testovat řídící systém. Základními komponenty pro sestavení jednoduchého nezávislého kamerového systému jsou bezesporu samotné IP kamery, zaznamenávající obraz a zvuk ze sledovaného prostoru, dále PoE switch sloužící pro napájení a komunikaci kamer a samozřejmě záznamová jednotka s operačním systémem sloužící jako řídící jednotka systému a úložiště pro příchozí data. Ještě je také potřeba napájecí zdroj pro záznamovou jednotku. Všechny části systému musí pro nasazení v reálných podmínkách splňovat drážní normu ČSN EN 50155. Základní popis normy je uveden v následují kapitole 3.1.

Systém by měl umožňovat synchronní nahrávání obrazu a zvuku ze všech kamer, monitorovat stav napájení záznamové jednotky a v případě výpadku zjistit, že došlo k přepnutí na záložní zdroj, ukončit nahrávání, všechna data uložit a jednotku vypnout a informaci o tomto výpadku (jeho čase) uložit. Měl by také monitorovat funkčnost kamer a jejich výpadky. Řídící software by měl také pamatovat na opotřebení záznamového disku a zvolit vhodný ukládací systém tak, aby nedocházelo k ničení disku častým přepisováním a mazáním dat. Program by měl umožňovat také komunikaci záznamové jednotky se stacionárním serverem, pokud je k dispozici ethernetové připojení, a umožnit posílání požadovaných záznamů z daných kamer pro libovolný provozní den vozidla, či vyříznout a odeslat pouze požadovaný kus záznamu. Pro tento účel se nejlépe hodí komunikace prostřednictvím aplikace server-klient a posíláním dat přes TCP protokol.

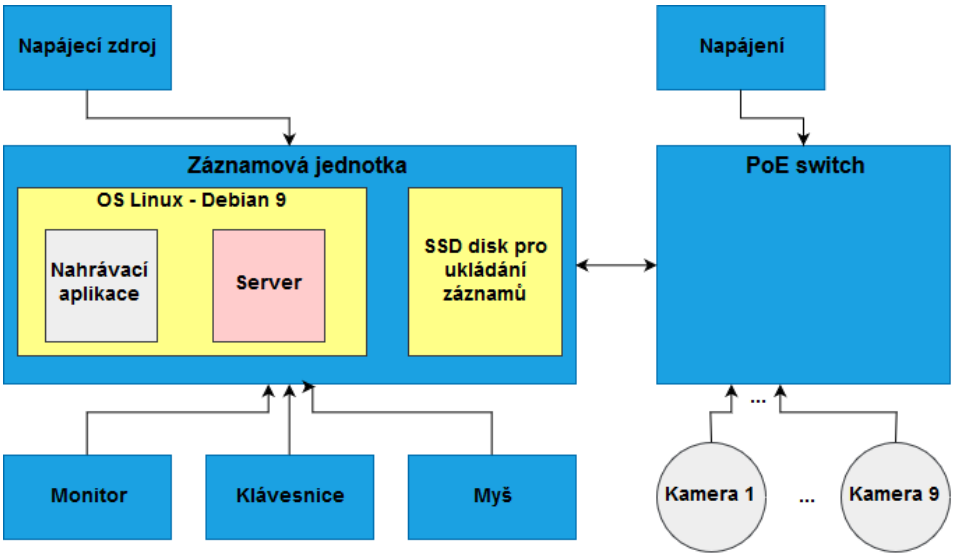
Pro ucelenější fungování systému by se ještě měla dořešit komunikace záznamové jednotky s informačním systémem a řídící jednotkou vozidla, která by umožnila například do kamerového záznamu zahrnout textové informace z informačního systému o poloze vozidla ve formě GPS souřadnic, předchozí a následující zastávce, číslu řidiče atd. Komunikace s řídící jednotkou by zase umožnila přidávat informace o klíčových událostech. Nicméně tato funkčnost není součástí této diplomové práce, kamerový systém byl řešen jako izolovaný systém. Tato funkčnost může být implementována později.

3.1 Drážní norma EN50155

Tato kapitola se zabývá drážní normou ČSN EN 50155 ed.3, která je českou verzí evropské normy EN 50155:2007. Jedná se o technickou normu, která se zabývá elektronickými a elektrickými drážními zařízeními. Byla vyhotovena technickou komisí TC 9X CENELEC. Platí pro všechna elektronická zařízení pro ochranu, napájení, regulaci a řízení instalovaná na kolejových vozidlech a spojená s akumulátorovou baterií vozidla nebo s napájecím zdrojem nízkého napětí s přímým připojením k trakčnímu vedení nebo bez přímého připojení k tomuto trakčnímu vedení s výjimkou elektronických výkonových obvodů, na které se vztahuje EN 50207. Tato norma zahrnuje pracovní podmínky, návrh, konstrukci a zkoušení elektronických zařízení a také základní požadavky na hardware a software, které jsou považovány za nezbytné pro kvalitní a bezporuchová zařízení. [31]

3.2 Komponenty kamerového systému

Jak již bylo zmíněno výše, izolovaný kamerový systém pro vývoj řídícího software se skládá ze záznamové jednotky, napájecího zdroje, IP kamer a PoE switche. Protože záznamová jednotka nebyla k dispozici na začátku vývoje, byla dříve nahrazena notebookem. Blokové schéma kompletního systému je vidět *Obr. 3.1* pod tímto odstavcem. Pro umožnění vývoje software a ovládání záznamové jednotky je k ní přes VGA připojen externí monitor a přes USB klávesnice a myš. Jednotlivé komponenty jsou rozebrány následně.



Obr. 3.1: Blokové schéma kamerového systému

3.2.1 IP kamery

Úkolem IP kamer v kamerovém sledovacím systému v kolejovém vozidle je snímat daný prostor a digitalizovaný obraz posílat k uložení do záznamové jednotky po ethernetové sběrnici, po které je řešeno i napájení. Při komplexním řešení systému, které není součástí této diplomové práce, je poté možno online data z kamer zobrazit na IP monitorech v kabině a případně i na displeji nadřazeného řízení. Počet kamer v systému pro simulaci věrohodné situace a dostatečné odzkoušení vzájemné synchronizace a konfigurace měl být původně 15. Nicméně se objevily problémy se zapojením a napájením takového množství kamer, a tak je výsledný počet kamer v systému 9. Nicméně to je stále dostatečné množství pro navození reálné situace a později nebude problém počet navýšit. Těchto 9 kamer si lze ve tříčlánkové tramvaji se 4 dveřmi a 1 kabinou představit takto: v každém článku tramvaje bude kamera monitorující vnitřní salón, což jsou 3 kamery, 1 bude umístěna v kabině řidiče a bude monitorovat jeho chování, nad každými dveřmi bude jedna kamera sledující nástupní prostor (4 kamery) a 1 kamera může být vně a sledovat pantograf.

Pro vyvíjený kamerový systém byly společností Škoda Transportation poskytnuty IP kamery Axis P3904-R Mk II (*Obr. 3.2*). Jedná se o vysoce výkonné robustní síťové kamery navržené pro mobilní sledování videa na palubách vozidel a kolejových vozidel, jako jsou autobusy, vlaky, tramvaje nebo metro. Jsou vybaveny ochranou proti prachu a vodě (krytí IP66/IP77) a mohou odolávat náročným podmínkám jako jsou vibrace, otřesy, nárazy a kolísání teplot. Zvládají dobrý obraz i v měnících se světelných podmínkách. Byly vybrány z prostého důvodu, bylo jich na skladě nejvíc a byly k dispozici a zároveň splňují normu EN 50155 a jejich parametry jsou pro tuto aplikaci dostatečné. Jejich problémem ale je, že neumožňují připojení analogového mikrofonu, a tak poskytovat i zvukový záznam. Proto by bylo lepší použít IP kamery Axis P3905-R, které to umožňují, ty ale nebyly k dispozici. Kompletní technická specifikace kamery se nachází v příloze A. [5]

Obr. 3.2: IP kamera Axis P3904-R MK II

Podporované formáty video komprese jsou H.264, který je použit pro nahrávání záznamu řídícím systémem, a MPJEG. Nejvyšší možné rozlišení je 1280 × 720 bodů, nejnižší pak 16 × 90 bodů. Snímková frekvence je 25 fps při 50 kHz nebo 30 fps při 60 kHz. Kamera je napájena po ethernetu (PoE) – standard IEEE802.3af/IEEE802.3at typ 1 třída 2, typický odběr má 2,9 W, maximálně pak 3,6 W. Zde právě nastal problém při napájení všech kamer. Použitý PoE switch má 16 konektorů – 8 z nich podporuje mid-spanové napájení (napájení na pinech 4/5 (+) a 7/8 (-) UTP kabelu) a 8 z nich podporuje end-spanové napájení (napájení na pinech 1/2 (+) a 3/6 (-) UTP kabelu). Tyto kamery by podle standardu měly být schopny býti napájeny oběma způsoby, nicméně reálně zvládají pouze end-spanové napájení, a proto jich lze z PoE switche napájet pouze 8. Zbývající devátá použitá kamera je Axis P3905-R, která zvládá i mid-spanové napájení. Mají konektory M12 female, a proto je nutno použít redukci M12 male na RJ45 male, aby bylo možno kamery připojit do PoE switche, který má konektory RJ45 female. Reálné zapojení kamer do switche je vidět na *Obr 3.3* níže. Normální provozní podmínky jsou od – 40 °C do 60 °C. [5]

Obr. 3.3: Zapojení IP kamer Axis P3904-R MK II do PoE switche

3.2.2 PoE switch

IP kamery musí nějak komunikovat se záznamovou jednotkou, být napájeny a posílat data pro živý přenos a uložení. K těmto funkcím jim dopomáhá switch (síťový přepínač), který zároveň umožňuje funkci napájení po ethernetu – tzv. PoE (Power over Ethernet) switch. Společností Škoda Transportation byl jako vhodný vybrán GSW 1600-HP PoE switch od Planet (vyobrazen níže na *Obr. 3.4*). Tento vysokorychlostní přepínač disponuje šestnácti 10/100/1000 Mbits ethernetovými RJ45 female porty. Každý port umožňuje napájení zařízení 52 V ss po ethernetu. Umožňuje napájení vysokými výkony podle IEEE 802.3af (až 15,4 W) a podle IEEE 802.3 af (až 30,8 W). Disponuje funkcí autodetekce připojeného zařízení. Obsahuje ochranu obvodů a umožňuje napájení až do vzdálenosti 100 m. [12]

Obr. 3.4: PoE switch GSW 1600-HP od Planet [12]

Podporuje Energeticky efektivní ethernetové sítě (EEE) podle normy IEEE802.3az. Zařízení je navrženo s ohledem na šetření energie, čehož je dosaženo pomocí dvou technologií. První z nich je využití klidového režimu (Idle Mode Link Down power saving), který automaticky sníží energii pro daný port, pokud v něm není zapojené zařízení. Druhou je inteligentní škálování napájení podle délky kabelu (Intelligent Scales Power). Tato technologie aktivně určuje příslušnou úroveň výkonu podle délky kabelu připojeného zařízení. Zařízení s přípojným kabelem menším než 20 m mohou dosáhnout největší úspory energie. Zařízení má i LED displej pro indikaci napájení (pokud je napájení přítomno – zelená LED dioda) a indikaci připojených zařízení na jednotlivých portech. Je ho nutno napájet střídavým napětím 100 až 240 V a maximálním proudem 3,5 A. Maximální odběr činí 260 W. Hmotnost zařízení je 2,54 kg. Technická specifikace je k nahlédnutí v příloze B. [12]

3.2.3 Záznamová jednotka

Záznamová jednotka tvoří základ celého kamerového systému. Plní funkci řídícího počítače, kdy na ní běží operační systém se záznamovou aplikací a serverem, umožňuje nastavování a ovládání kamer a také plní funkci diskového úložiště, kdy se na ní ukládají zaznamenaná data. Musí také umožnit získání požadovaných záznamů či jejich částí bez přístupu k disku, tzn. pomocí nějakého klienta – například pomocí aplikace server-klient na portu 8080 a posílání dat přes TCP protokol. Byl vybrán VBOX-3620-M12X od společnosti Sintrones. Reálná záznamová jednotka je vidět na *Obr. 3.5* níže.

Obr. 3.5: Záznamová jednotka VBOX-3620-M12X, vlevo zepředu, vpravo zezadu

Jednotka použitá ve vyvíjeném systému disponuje procesorem Intel Gen6 Core i5-6300U 2,4 až 3 GHz a dvěma GB operačními paměťmi. Má implementovány 3 chipsety pro LAN s třemi 10/100/1000 Mb/s M-12 x-code porty. Veškeré technické údaje jsou k dohledání v příloze C. Zařízení disponuje 4 sériovými porty a 4 USB porty. Pro práci s videem obsahuje 2 DP porty a 1 VGA port. Jednotka má integrovaný 16 GB SATA DOM disk typu FLASH, na kterém běží operační systém, záznamová aplikace a server pro komunikaci s externím PC. Dále disponuje sloty pro 2 disky – 2,5“ sloty pro disky od SATA typu HDD nebo SSD. Obecně u takovýchto záznamových jednotek platí, že tyto HDD/SSD disky slouží pouze pro ukládání dat. V mém případě je k dispozici jeden 256 GB 2,5“ SSD. [30]

Záznamová jednotka má variabilní rozsah napájecího napětí – je umožněno stejnosměrné napájecí napětí v rozsahu 9 až 36 V. V mém případě je k napájení jednotky využíván spínaný zdroj se stejnosměrným napětím 24 V. Použitý zdroj je zobrazen na *Obr. 3.6* pod tímto odstavcem. Zařízení disponuje ochranou proti zkratu, automatickým obnovením, v BIOS lze softwarově nastavit zpoždění vypnutí a jednotka také obsahuje interní baterii, která zaručí 10 minut provozu po výpadku napájení, což je důležitá funkčnost, protože třeba při výpadku jističe mezi baterií vozidla a záznamovou jednotkou, by při absenci této vnitřní baterie došlo k okamžitému vypnutí jednotky a ztrátě všech dat z daného dne, případně i všech dat ze záznamové jednotky. Takto lze při ztrátě napájení softwarově vyhodnotit, že došlo ke ztrátě napájecího napětí a došlo k přechodu na záložní baterii, ukončit záznamovou aplikaci, uložit všechna data, zaznamenat čas výpadku a jednotku vypnout. Záznamová jednotka je schopna pracovat v rozsahu teplot od -40 °C do 70 °C. Aby se dala záznamová jednotka použít jako počítač pro vývoj software – tzn. disponovala grafickým rozhraním a měla k dispozici ovládací prvky, byl k ní připojen monitor a klávesnice a myš. [30]

Obr. 3.6: Napájecí zdroj záznamové jednotky – spínaný zdroj stejnosměrných 24 V

4 Záznamová aplikace

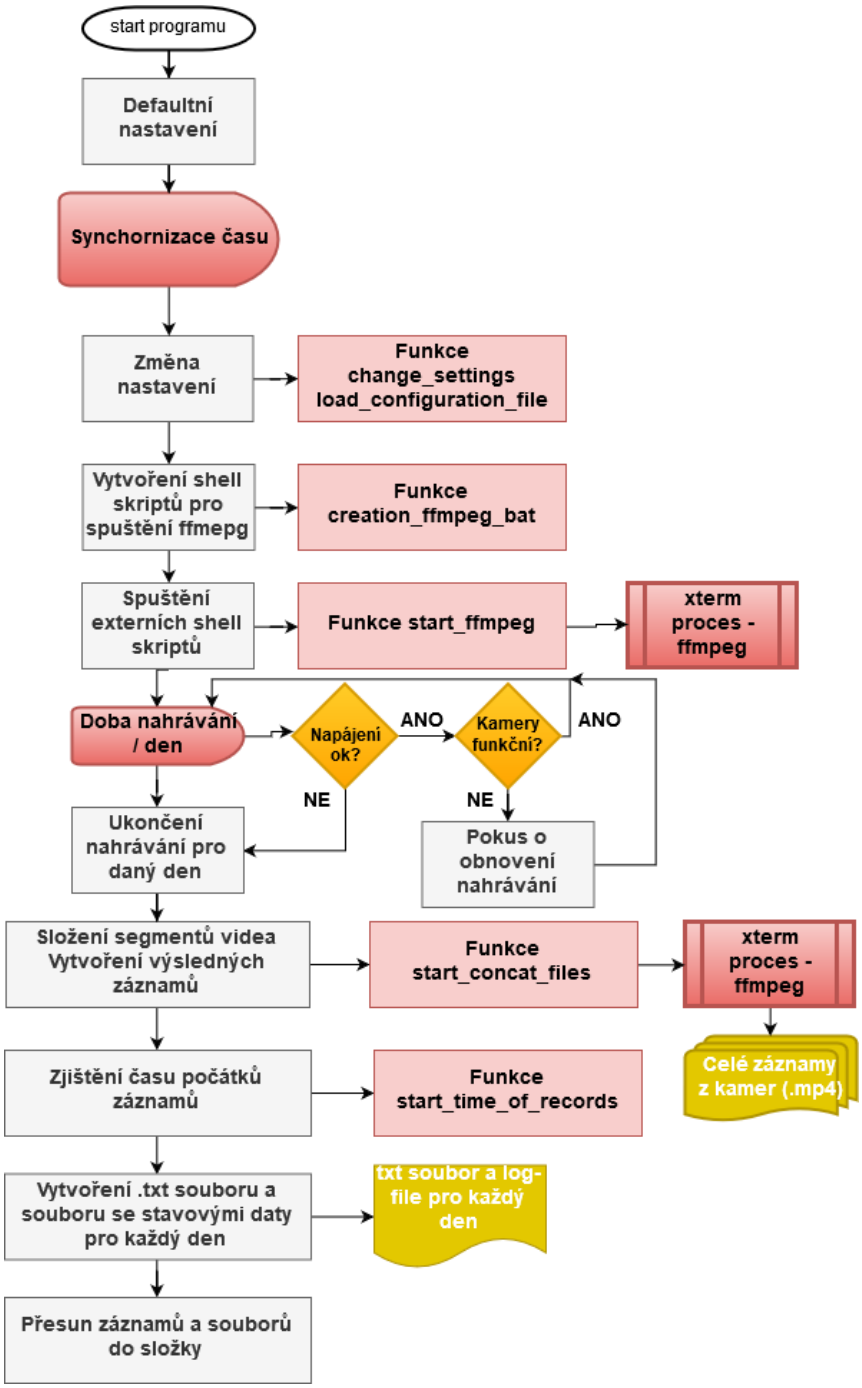
Záznamová aplikace je stěžejní částí celého kamerového systému. Když se kamerový systém začal vyvíjet, nebyla záznamová jednotka ještě k dispozici, a tak byla její funkce dočasně nahrazena notebookem s operačním systémem Windows 7. Aby byl poté kód snadno přenositelný na záznamovou jednotku, pro kterou byl vybrán operační systém na bázi linuxu – Debian 9 Stretch, byl zvolen programovací jazyk C, který je pro danou aplikaci zcela dostačující. Jako vývojový nástroj byl ve Windows použitý software Microsoft Visual Studio 2013 Professional a v OS Debian pak Visual Studio Code. Součástí záznamové aplikace byly také batch soubory ve Windows, které byly následně nahrazeny shell skripty v OS Debian. Pro samotné nahrávaní je využit opensource program ffmpeg.

Před samotným programováním bylo nutno ještě správně nastavit všechny použité IP kamery – nastavit jim správné statické IP adresy, byly zvoleny IP adresy 192.168.0.101 až 192.168.0.109. Poté, co byly kamerám přiřazeny statické IP adresy, je možno se ke kameře připojit zadáním její IP adresy do webového prohlížeče. Zde je možno nastavovat požadované parametry. Důležité je nastavit správně parametry formátu H.264, který je využit pro ukládaní záznamů. Dále je potřeba nastavit, co se bude zobrazovat ve výsledném záznamu – byl zvolen doprovodný text (jméno kamery) a aktuální čas s přesností na setiny sekund. Následně je potřeba správně nastavit číslo portu pro RTSP protokol, který je využíván pro nahrávání záznamů. Jedná se o port 443. Důležité je také nastavení synchronizace času kamery – je možno nastavit kameře čas manuálně, nebo zvolit synchronizaci s hodinami počítače, anebo synchronizaci s NTP serverem. Byla zvolena poslední možnost, protože ta zajistí, aby byly všechny kamery synchronizovány na stejný čas s velkou přesností. Jako NTP server byla zvolena záznamová jednotka, a to pomocí její IP adresy 192.168.0.100. Důležité bylo, nastavit všechny kamery stejně, a také je manuálně zaostřit, aby byl obraz dostatečně kvalitní.

4.1 Celkový popis nahrávací aplikace

V této kapitole je blokově popsána principiální funkce celé záznamové aplikace, jejíž kompletní blokový diagram je zobrazen na *Obr. 4.1* níže. Program je psán v jazyce C a má typickou strukturu. V programu jsou použity 3 funkce sloužící k získání povolení od systému k monitorování portů a stavu napájení, ke čtení a zápisu nastavení zpoždění pro vypnutí pro každý port. Důležitých je 10 uživatelských funkcí, které jsou volány při běhu programu. Jedná

se o funkci pro manuální nastavení parametrů kamerového systému přímo z konzole, funkci pro načtení konfiguračního souboru, funkci pro vytvoření shell skriptů pro každou kameru s parametry pro ffmpeg, funkci pro vytvoření shell skriptu pro spuštění procesu ffmpeg. Následuje funkce pro zjištění počátečních časů záznamů, funkce pro složení videí dohromady, funkce pro monitorování funkčnosti kamer, funkce pro vytvoření textového souboru s počátečními časy, funkce pro vytvoření shell skriptu pro přesunutí příslušných souborů do složky a funkce pro vytvoření textového log_file souboru se stavovými daty.



Obr. 4.1: Blokový diagram celé záznamové aplikace

Poté, co se program spustí, provede se defaultní nastavení systému – je nastaven počet kamer v systému a každé kameře je přiřazena správná IP adresa, parametry nahrávaného obrazu (rozlišení, fps, čas mezi klíčovými snímky, dobra trvání segmentu záznamu a celková doba nahrávání) a nastavení se zobrazí na konzoli. Počet kamer v systému je nastaven na 9, snímková frekvence na 25 fps, rozlišení 800 × 450 pixelů, klíčový snímek přítomen minimálně každou 1 s, doba trvání segmentu záznamu 60 s a celková doba nahrávání 57600 s (16 h). Následně program vstoupí do čekací smyčky vytvořené pomocí while-cyklu, která trvá 30 s a slouží k tomu, aby byl poskytnut čas kamerám pro synchronizaci času se záznamovou jednotkou. Následuje pak nekonečná smyčka, kde je umožněna změna nastavení, buď manuálně z konzole, nebo načtením konfiguračního souboru. Pokud zde není vstup od uživatele, program automaticky načte konfigurační soubor a pokračuje dále. Celý proces změny nastavení je podrobně popsán v následující kapitole 4.2.

Poté, co je změněno nastavení, nebo je načten konfigurační soubor, následuje část programu, ve které probíhá nahrávání záznamů a volají se uživatelské funkce. Nejdříve se zavolá funkce *creation_ffmpeg_bat*, která vytvoří spustitelné shell skripty pro ffmpeg, její podrobný popis se nachází v kapitole 4.3. Následně se volá funkce *start_ffmpeg*, která spustí shell skripty a tím nahrávání segmentů záznamů ze všech kamer pomocí ffmpeg. Tato funkce je popsána detailněji v kapitole 4.4. Výsledkem je tedy spuštěný proces ffmpeg pro každou kameru zvlášť, který běží na pozadí a vytváří se 60 s segmenty záznamů a výstupní textové soubory ke každému záznamu, všechny tyto soubory jsou ukládány na pevný SSD disk do pracovní složky DATA. Program se dále větví na 2 části. Buď pokračuje částí pro nahrávání 1 provozního dne vozidla (v konfiguračním souboru je uvedena požadovaná doba nahrávání), přičemž po uplynutí provozního dne se program ukončí a záznamová jednotka se vypne, nebo pokračuje módem nepřetržitého nahrávání (v konfiguračním souboru je uvedena doba nahrávání 24 h), kdy nahrávání probíhá nepřetržitě a vždy po půlnoci se zpracují data pro uplynulý den. Program tak buď vstoupí do čekací smyčky, která trvá po dobu trvání provozního dne, nebo do 2 nekonečných smyček pro nepřetržitý mód.

Až uplyne požadovaná doba nahrávání nebo skončí den, program ze smyčky vystoupí a pokračuje dál (v nepřetržitém módu mezitím pokračuje nahrávání dalšího dne). Avšak v průběhu nahrávání v čekací smyčce ještě dochází k neustálému monitorování stavu napájení záznamové jednotky. Každou minutu se pomocí funkce *Funkcnost_kamer_obnoveni_kamer* taktéž monitoruje funkčnost všech kamer, a pokud by se zjistil výpadek, tak každé 2 minuty

se program pokouší nahrávání z nefunkční kamery obnovit. Pokud dojde k výpadku kamery, je její obraz nahrazen černou obrazovkou, aby se zachovala spojitost záznamu a všechny záznamy měly stejnou délku. Tato funkčnost je rozebrána v kapitole 4.5. V další fázi programu se volá uživatelská funkce *start_concat_files*, která vede k vytvoření textových souborů pro concat funkci ffmpeg pro jednotlivé kamery a spuštění ffmpeg podle těchto souborů, což vede k sestavení segmentů videa dohromady a vytvoření tak výsledných záznamů. Funkce je popsána v kapitole 4.6.

Dále je potřeba vytvořit textový soubor pro daný nahrávací den, k čemuž slouží funkce *create_text_file,* který se jmenuje podle daného data (rok-měsíc-den.txt) a obsahuje počáteční časy všech kamerových záznamů pro daný den. Tento soubor je důležitý později pro vyhledávání a vyřezávání požadovaných záznamů. Tato část programu je rozebrána v kapitole 4.7. Po vytvoření výstupního textového souboru pro daný den program vstupuje do své poslední fáze. Tou je vytvoření spustitelného shell skriptu, který přesune výsledné video záznamy ze všech kamer a další potřebné soubory, jako výstupní textový soubor pro každou kameru s časovými značkami, textové soubory pro concat, konfigurační soubor a výsledný textový soubor pro daný nahrávací den, do složky pro daný nahrávací den pojmenované podle data (rok-měsíc-den). Vytvoření skriptu je realizováno pomocí funkce *move_files*. Nakonec se pomocí funkce *create_log_file* pro nahrávací den vytvoří log-file s informacemi o datu, délce nahrávání, počtu funkčních a nefunkčních kamer v systému, jejich parametrech, o případném výpadku kamery nebo napájení (čas výpadku a obnovení). Tento poslední krok programu je rozebrán v kapitole 4.8. Po ukončení tohoto kroku a spuštění shell skriptu, se spustí příkaz na vypnutí jednotky v případě nahrávání po dobu provozního dne nebo se program vrátí zpět do nekonečné smyčky a čeká na konec dalšího dne.

Aby vše probíhalo automaticky a nahrávání se spustilo samo a nebylo potřeba zásahů uživatele a taktéž nebylo potřeba žádné grafické prostředí, je nutné, aby se nejprve po zapnutí napájení jednotky načetl systém automaticky bez vyžadování hesla. To bylo provedeno pomocí úpravy souboru /etc/lightdm/lightdm.conf a poté provedení dvou příkazů v terminálu s kompetencí práv root.

```
[Seat:*]
pam-service=lightdm
pam-autologin-service=lightdm-autologin
autologin-user=Martin
autologin-user-timeout=0
session-wrapper=/etc/X11/Xsession
greeter-session=lightdm-greeter
```

```
# groupadd -r autologin
# gpasswd -a Martin autologin
```

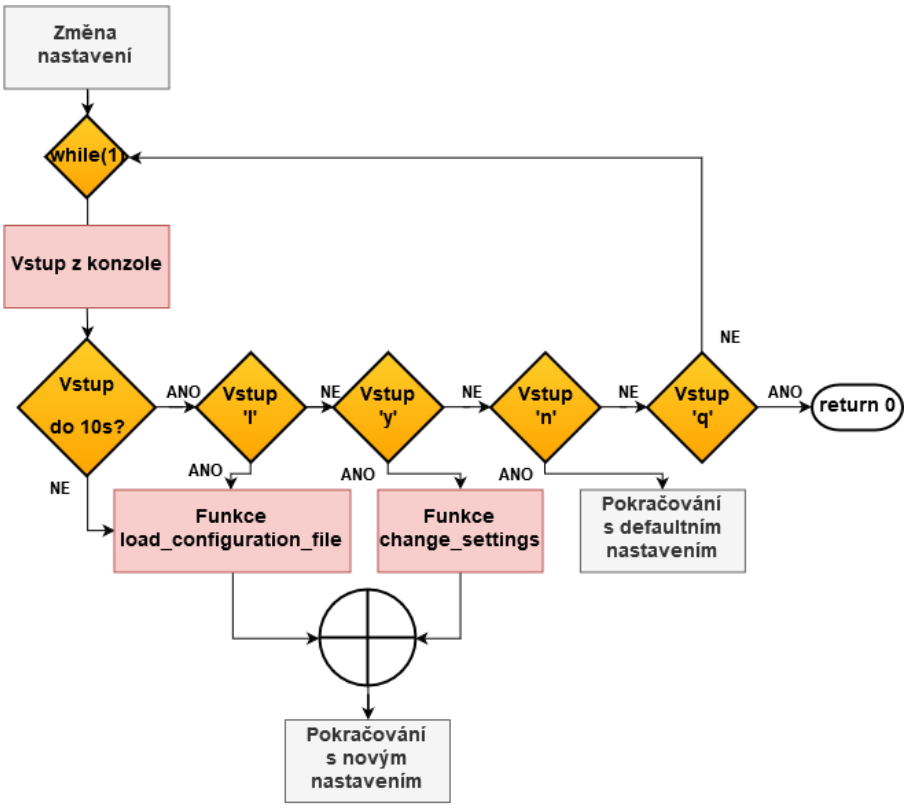
Poté byl vytvořen shell skript run.sh, který se automaticky spustí po naběhnutí systému. Obsahuje příkaz pro připojení pevného SSD disku, přiřazení správné IP adresy ethernetové LAN síti, do které je připojen PoE switch s kamerami, příkaz pro spuštění NTP serveru, příkaz na přistoupení do správné složky, příkaz pro odstranění souborů ze složky, následuje příkaz pro přeložení .c kódu a vytvoření spustitelného souboru, a nakonec příkaz spouštějící nahrávací a serverovou aplikaci. Celý shell skript je vidět na *Obr. 4.2* níže.

```
#!/bin/bash
sudo mount /dev/sda1 /mnt/newhd                          # pripojeni pevneho disku
sudo ip address add 192.168.0.100/24 brd + dev enp1s0    # prirazeni IP adresy eth 0 LAN siti
sudo /etc/init.d/ntp start                               # spusteni ntp serveru pro synchronizaci kamer
cd /home/martin//Program                                 # pristup do spravne slozky
sudo rm -rf ./Camera*.sh                                 # odstraneni zbylych shell skriptu z predchoziho dne
sudo rm -rf ./start.sh
sudo rm -rf ./concat*.sh
gcc -v -static Program.c -lm -g -o Program               # prelozeni nahravaci a ridici aplikace v .c jazyku
gcc Program_cut_videos.c -o Server                       # prelozeni serverove aplikace pro ziskavani zaznamu klientem
chmod +x Program                                         # ucineni vystupnich souboru spustitelnymi
chmod +x Server
sudo ./Program                                           # spusteni nahravaci aplikace
sudo gnome-terminal -x ./Server                          # spusteni serveru
```

Obr. 4.2: Shell skript pro automatický běh aplikace

4.2 Změna nastavení systému

Poté, co je provedeno defaultní nastavení systému, je umožněna jeho změna. Vývojový diagram této části programu je zobrazen na *Obr. 4.3* dále. Pro změnu slouží nekonečný cyklus. Čeká se na uživatelský vstup – uživatel může zadat znak ‚y' pro manuální změnu parametrů – zavolá se funkce *change_settings* pro to určená, její popis se nachází v následující kapitole 4.2.1. Nové nastavení je vypsáno na konzoli a program vyskočí ze smyčky. Nebo může zadat ‚n' pro pokračování s defaultním nastavením, kdy program rovnou vyskočí ze smyčky, nebo zadat ‚l' pro načtení konfiguračního textového souboru (*Obr. 4.5*). Případně je možno zadat ‚q' pro ukončení programu. Nicméně aby byl zajištěn automatický chod aplikace a v případě žádného vstupu program nečekal do nekonečna, je zde zajištěno, že pokud není žádný vstup do 10 s, program automaticky pokračuje s načtením konfiguračního souboru. Tato funkčnost je realizována pomocí funkce *poll(&mypoll, 1, 10000)*.



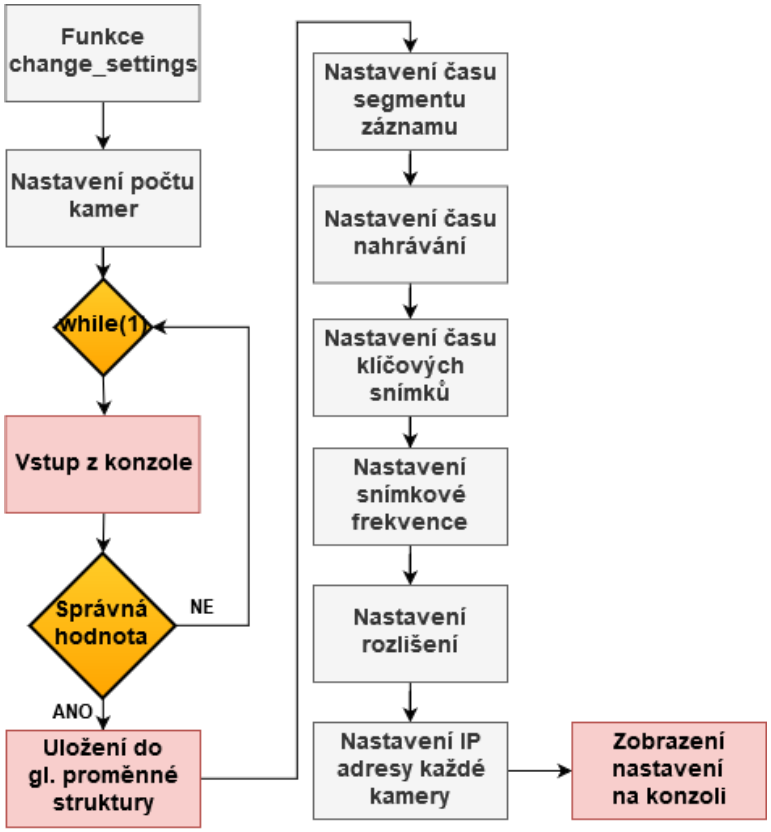
Obr. 4.3: Vývojový diagram bloku změna nastavení systému

4.2.1 Nastavování parametrů systému uživatelem

Pokud nechceme nechat defaultní nastavení systému, nebo načíst konfigurační soubor pro změnu nastavení, je zde možnost programu změnit nastavení ručně zadáním hodnot z konzole. Tato možnost je realizována pomocí funkce *change_settings*, jejíž vývojový diagram je vidět na *Obr. 4.4* pod tímto odstavcem. Pokud dojde k zavolání funkce, program nejdříve skočí do nekonečného while-cyklu pro zadání požadovaného počtu kamer v systému, zde se čeká na uživatelský vstup z konzole. Je nutno zadat přirozené číslo v rozmezí 1 až 9, jinak je to neplatný vstup a je nutno zadat novou hodnotu, z while-cyklu se vyskočí pomocí příkazu *break* po zadání korektní hodnoty. Poté program skočí do dalšího while-cyklu.

Při zadávání hodnoty požadovaného času segmentu záznamu v sekundách je nutno zadat přirozené číslo v rozmezí od 1 do 600. Poté program pokračuje nastavením celkového času nahrávání záznamů. Je nutno zadat přirozené číslo v rozmezí 1 až 72000 (v sekundách). Časování klíčových snímků (v angličtině time of keyframe) je parametr určující, v jakých intervalech se při nahrávání budou objevovat klíčové snímky. Pokud se nastaví tento parametr například na 1 s, znamená to, že klíčový snímek bude v záznamu přítomný minimálně každou vteřinu. Při snímkové frekvenci 25 fps to znamená, že v jedné sekundě záznamu bude vždy

přítomný alespoň 1 klíčový snímek. Tomu se říká parametr GoP (Group of Picture = skupina snímků). Je nutno zadat přirozené číslo v rozmezí 1 až 900 (v sekundách). Poté program skočí do nekonečného while-cyklu pro nastavení snímkové frekvence, kde je potřeba zadat konkrétní přirozené číslo (10, 15, 20, 25, 30)). Následuje zadání rozlišení – šířky obrazu, tolerovány jsou pouze hodnoty 1200, 800 a 600 pixelů, aby bylo nastaveno typicky používané rozlišení. Poté program skočí do sekce zadání rozlišení – výšky obrazu. Zde je stejný proces, uživatel musí zadat správnou výšku rozlišení v pixelech – povolené jsou hodnoty 340, 450 a 720 pixelů. Nakonec program skočí do nekonečného while-cyklu pro nastavení jednotlivých IP adres kamer. Toto nastavení se provádí pro každou kameru zvlášť, a to zadáním správné IP adresy ve tvaru xxx.xxx.xxx.xxx.



Obr. 4.4: Vývojový diagram funkce pro ruční změnu nastavení

4.2.2 Načtení konfiguračního souboru

Načtení konfiguračního souboru (*Obr. 4.5*) se provádí zavoláním funkce *load_configuration_file*. Otevřený textový soubor se načítá řádek po řádce pomocí *fgets* a ukládá se do pole ve while-cyklu až do konce souboru. Je zajištěna ochrana pro správné

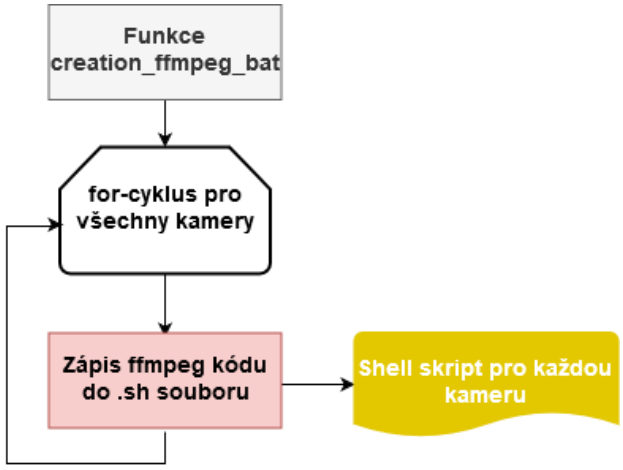
otevření a správné ukončení souboru. Když se soubor podaří řádně zavřít, nové nastavení systému se zobrazí na konzoli a program vyskočí ze smyčky pomocí příkazu *break*.

```
9               Number of cameras used in the system
192.168.0.101   IP address1
192.168.0.102   IP address2
192.168.0.103   IP address3
192.168.0.104   IP address4
192.168.0.105   IP address5
192.168.0.106   IP address6
192.168.0.107   IP address7
192.168.0.108   IP address8
192.168.0.109   IP address9
192.168.0.110   IP address10
192.168.0.111   IP address11
192.168.0.112   IP address12
192.168.0.113   IP address13
192.168.0.114   IP address14
192.168.0.115   IP address15
800             Resolution - weight
450             Resolution - height
25              Frame rate (fps)
1               Time of keyframe
60              Time of recorded segment stream (in seconds)
86400           Time of whole recorded stream (in seconds)
```

Obr. 4.5: Ukázka konfiguračního textového souboru pro nastavení kamerového systému

4.3 Funkce pro vytvoření shell skriptů pro ffmpeg

Poté, co program pokračuje s defaultním nebo se změněným nastavením, dostane se do fáze, kdy se volá uživatelská funkce pro vytvoření shell skriptů pro jednotlivé kamery s parametry pro ffmpeg. Co je to ffmpeg a jak funguje, je popsáno v kapitole 4.3.1. Jedná se o funkci *creation_ffmpeg_bat*, která se volá s parametrem počet kamer v systému. Její vývojový diagram je zobrazen na *Obr. 4.6*.



Obr. 4.6: Vývojový diagram funkce pro vytvoření shell skriptů pro ffmpeg

Kód funkce proběhne tolikrát, kolik je počet kamer v systému – to je zajištěno pomocí for-cyklu. Pokud se podaří otevřít soubor pro čtení, zapíše se tam požadovaný kód pro ffmpeg pomocí funkce *fprintf*, který zařídí nahrávání záznamů a synchronizaci videa a audia a synchronizaci jednotlivých záznamů. Co každá část kódu znamená, je vysvětleno v následující kapitole 4.3.1. Poté, co je práce se souborem ukončena, dojde k jeho zavření pomocí *fclose*. Je zde ošetřeno řádné otevření a zavření souboru. Skript je pro kameru č.1 zobrazen na *Obr. 4.7* pod tímto odstavcem.

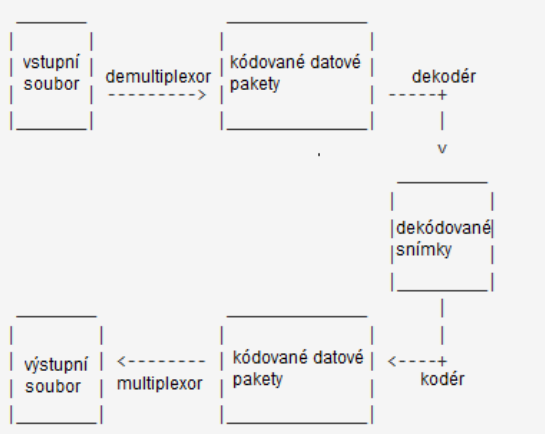
```
#!/bin/bash
ffmpeg -rtsp_transport tcp -i "rtsp://root:pass@192.168.0.101:554/axis-media/media.amp?
videocodec=h264&h264profile=main&resolution=800x450&fps=25" -c:v copy -r 25 -force_key_frames
"expr:gte(t, n_forced * 1)" -c:a copy -flags +global_header -copyts -vsync cfr -f segment -
strftime 1 -segment_list /mnt/newhd/DATA/out1.txt -segment_list_type csv -reset_timestamps 1 -
segment_atclocktime 1 -segment_time 60 -segment_format mp4 -segment_time_delta 0.02 -t
315360000 /mnt/newhd/DATA/Cam1_%Y-%m-%d_%H-%M-%S.mp4
```

Obr. 4.7: Výsledná podoba shell skriptu pro každou kameru

4.3.1 Fmpeg

Ffmpeg je kolekce svobodného softwaru, který umožňuje konverzi, nahrávání a streamování digitálního audia i videa. Vyvíjen je primárně pro operační systémy na bázi Linux, ale je podporován v různých verzích i pro většinu ostatních OS. Je založen na knihovně libavcodec – nejdůležitější knihovně pro kompresi videa a audia. Kolekce je složena z několika komponent. Ffmpeg je utilita pro kompresi video a audio formátů, nahrávání a umožňuje i kódování v reálném čase. Ffplay představuje multimediální přehrávač. Ffserver dává uživateli možnost streamování pro živá broadcastová vysílání. Ffprobe je nástroj pro analýzu multimediálních streamů. Byl vybrán pro jeho jednoduchost, rychlost, široké možnosti využití, podporu napříč operačními systémy a free licenci. [11]

Pro nahrávání z kamer je využita utilita ffmpeg. Princip ffmpeg nástroje je zobrazen na *Obr. 4.8* pod tímto odstavcem. Ffmpeg volá knihovnu libavformat, která obsahuje demuxery, pro čtení vstupních souborů a získání paketů obsahující zakódovaná data. Při více vstupních souborech se ffmpeg pokouší udržovat jejich synchronizaci sledováním nejnižší časové známky na jakémkoliv aktivním datovém proudu. Následně jsou kódované pakety předané dekodéru, který vytváří nekomprimované snímky, které lze dále filtrovat. Po filtrování jsou snímky předané kodéru, který je zakóduje a vygeneruje kódované pakety, které jsou předány muxeru, který je zapíše do výstupního souboru. [11]



Obr. 4.8: Princip funkce nástroje ffmpeg z kolekce Ffmpeg

Nyní následuje rozbor příkazu pro ffmpeg vygenerovaném z programu. Příkaz ffmpeg říká systému, aby využil právě tuto utilitu z FFMPEG. Následuje příkaz -rtsp_transport tcp, který určuje, jaký typ nižšího transportního protokolu bude použit pro RTSP, v našem případě je použit TCP protokol, protože se hodí pro tuto aplikaci lépe (jeho výhody jsou popsány v kapitole 1.4). Následuje příkaz -i, který určuje, že to, co následuje, je vstupní soubor. Nejdříve je sděleno, jaký typ protokolu se používá pro komunikaci s kamerou – konkrétně se jedná o RTSP protokol, který je nejvhodnější pro nahrávání živého záznamu z IP kamer. Poté je zadána internetová adresa pro živý záznam z kamery – skládá se z uživatelského jména a hesla, IP adresy, typu média, použitého video kodeku (je použitý kodek H.264, hlavní profil), rozlišení (použito 800 × 450 pixelů) a snímkové frekvence (25 fps).

Následuje příkaz -c:v copy, který programu říká, aby pro výstupní soubor použil stejný video kodek, jako je video kodek vstupního souboru, tedy H.264 s hlavním profilem. Parametr -r 25 určuje, aby měl výstupní soubor přesnou snímkovou frekvenci 25 fps. Následující příkaz -force_key_frames "expr:gte(t, n_forced * 1)" vnucuje klíčové snímky každou 1 s, čímž je zaručeno, že to tak u výstupního souboru vždy přesně bude, a klíčové snímky po 1 s pak zaručují vysokou přesnost při vyřezávání požadovaného záznamu v požadovaném časovém intervalu, zároveň to mají záznamy ze všech kamer stejné, což je důležité pro synchronizaci. Následující příkaz -c:a copy říká, aby měl výstupní soubor použitý stejný audio kodek, jako vstupní soubor. Nicméně v našem případě zde žádné audio není, protože ho kamery neumožňují, avšak díky tomuto příkazu bude v budoucnu při použití lepších kamer umožněno i synchronní nahrávání audia. Další příkaz -flags +global_header vynucuje globální záhlaví ve H.264 paketech generovaných kodérem libx264. Příkaz -copyts určuje, aby nebyly zpracovány vstupní časové značky, ale aby jejich hodnoty byly uchovány

beze změny, tedy aby nebyly například odstraňovány posuny počátečního času. Dále je přítomen příkaz -vsync cfr, který určuje metodu synchronizace videa. Parametr cfr říká, že snímky mohou být duplikovány či zahozeny, aby se dosáhlo přesně požadovaného fps.

Příkaz -f segment programu říká, že má využít základní segmenter streamu. Tento muxer rozděluje vstupní stream do řady samostatných souborů s téměř pevnou dobou trvání. Vzor výstupního souboru může být nastaven pomocí parametru -sfrtime 1, což je i náš případ, který určuje název nových segmentů. Tento muxer se používá pro zápis do výstupních datových streamů, je ideální pro výstup do segmentů přenosu streamu MPEG. Každý segment začíná klíčovým snímkem vybraného referenčního toku. Pokud chceme dosáhnout přesného rozdělení video souboru, je třeba, aby vstupní klíčové snímky odpovídaly přesným časům rozdělení očekávaných segmenterem, toho je dosaženo daným nastavením. Příkaz -segment_list /media/newhd/DATA/out1.txt slouží ke generování seznamu vytvořených segmentů – je určeno kam a pod jakým názvem se má list generovat a příkazem -segment_list_type csv je určen jeho typ.

Příkaz -reset_timestamps 1 nařizuje, aby byly časové značky obnovovány na začátku každého segmentu tak, aby každý segment začal s nulovou časovou značkou, což usnadňuje přehrávání generovaných segmentů. Nastavení -segment_atclocktime 1 -segment_time 60 určuje, že každý segment je dlouhý přesně 60 s a vstupní datový proud může být dělen pouze v přesných pravidelných časových intervalech – v tomto případě každou minutu. Reálně první segment začne, když se pustí ffmpeg (je tedy kratší než 60 s) a každý další pak již splňuje dané nastavení. Příkaz -segment_format mp4 slouží k přepsání formátu vnitřního kontejneru pro výstupní segmenty na formát mp4. Příkaz -segment_time_delta 0.02 je v naší aplikaci důležitý, protože určuje čas přesnosti při výběru časového začátku segmentu. Když je zadána delta, klíčový snímek zahájí nový segment, pokud jeho PTS splňuje vztah: PTS ≥ start_time - time_delta. Tato volba je užitečná při rozdělení videoobsahu, který je vždy rozdělen na hranicích GOP, v případě, že se klíčový rámec nachází těsně před stanoveným časem rozdělení. Toto nastavení se používá zejména v kombinaci s volbou -force_key_frames.

Na závěr již jen příkaz -t určuje celkovou dobu trvání nahrávání v sekundách a /mnt/newhd/DATA/Cam1_%Y-%m-%d_%H-%M-%S.mp4 určuje do jaké složky a pod jakým názvem se mají výsledné segmenty vstupního datového streamu ukládat. Při tomto

čase je každý segment uložen do pracovní složky DATA na pevném SSD disku pod názvem odvozeným od čísla kamery a počátečního data a času segmentu.

Toto výsledné nastavení bylo zjištěno experimentálně zkoušením různých kombinací nastavení různých parametrů vstupu a segmenteru podle dokumentace FFMPEG. Ukázalo se jako nejvhodnější pro vzájemnou synchronizaci všech kamerových záznamů a zároveň nedochází při segmentování ke ztrátě nebo nadbytku dat, to znamená, že po složení všech segmentů dohromady odpovídá výsledná délka záznamu požadované době a odpovídá i reálnému času hodin. Výhoda segmentace vstupu do 60 s bloků a následného složení dohromady spočívá ve vysoké přesnosti časových značek a klíčových snímků, což pak velmi zpřesňuje posun a umožňuje výřez požadovaného časového intervalu s vysokou přesností.

4.4 Spuštění nahrávání pomocí ffmpeg

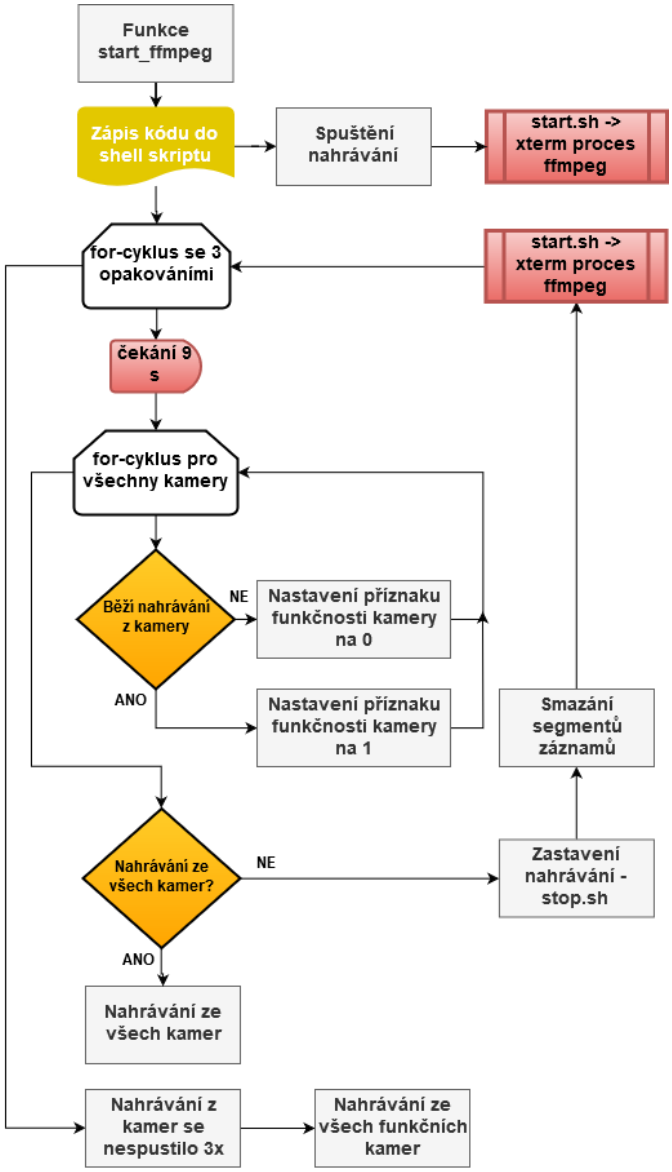
Poté, co jsou vytvořeny shell skripty pro ffmpeg pro všechny kamery, je nutno je spustit. K tomu slouží funkce *start_ffmpeg*, jejíž vývojový diagram je zobrazen na *Obr. 4.10* níže. Je založena na tom, že se do shell skriptu start.sh zapíší všechny shell skripty pro všechny kamery, které se mají spustit, učiní se spustitelnými soubory, a tento skript se spustí a tím se spustí i všechny ostatní. Nejdříve se otevře soubor start.sh pro zápis a pomocí *fprintf* se do něj zapíše požadovaný kód. Výsledná podoba toho shell skriptu je vidět na *Obr. 4.9* níže.

Poté, je shell skript spuštěn, program vstoupí do for-cyklu s možností 3 opakování ve kterém se testuje, zdali je spuštění nahrávání ze všech kamer úspěšné. Pokud není, provede se dvakrát nový pokus – aby se zjistilo, zdali je kamera opravdu nefunkční nebo se jen špatně spustilo nahrávání. Pokud ani 3 pokusy nestačí pro začátek nahrávání ze všech kamer, znamená to, že jedna či více kamer je nefunkčních, každé kameře se přiřadí 1 nebo 0 podle toho, jestli je na počátku nahrávání funkční, nebo není, a spočítá se počet funkčních kamer v systému na začátku nahrávání. Ověření, že nahrávání z každé kamery běží správně, se provádí ve for-cyklu pro všechny kamery testováním přítomnosti výstupního textového souboru pro každou kameru v příslušné složce. Samozřejmě je v programu přítomné ošetření na správné otevření a uzavření souborů. Pokud ve složce nějaký výstupní soubor chybí, znamená to, že nahrávání pro danou kameru neprobíhá a nahrávání se zastaví pomocí spuštění externího procesu stop.sh (*Obr. 4.11*), v pracovní složce se odstraní všechny soubory a nahrávání se spustí znovu pomocí start.sh. Pokud jsou v pracovní složce DATA přítomné

všechny výstupní soubory, nahrávání probíhá úspěšně, funkce se ukončí a program vstoupí do čekací smyčky po dobu nahrávání nebo do nekonečné smyčky v případě nekonečného nahrávání.

```
#!/bin/bash
for i in `seq 1 9`;
do
chmod +x Camera_${i}.sh &
done &
for i in `seq 1 9`;
do
gnome-terminal -x ./Camera_${i}.sh &
done &
```

Obr. 4.9: Shell skript pro spuštění nahrávání pomocí ffmpeg



Obr. 4.10: Vývojový diagram funkce pro spuštění nahrávání pomocí ffmpeg

```
killall ffmpeg &
rm -rf /mnt/newhd/DATA/out*.txt &
rm -rf /mnt/newhd/DATA/Cam*.mp4 &
exec bash
```

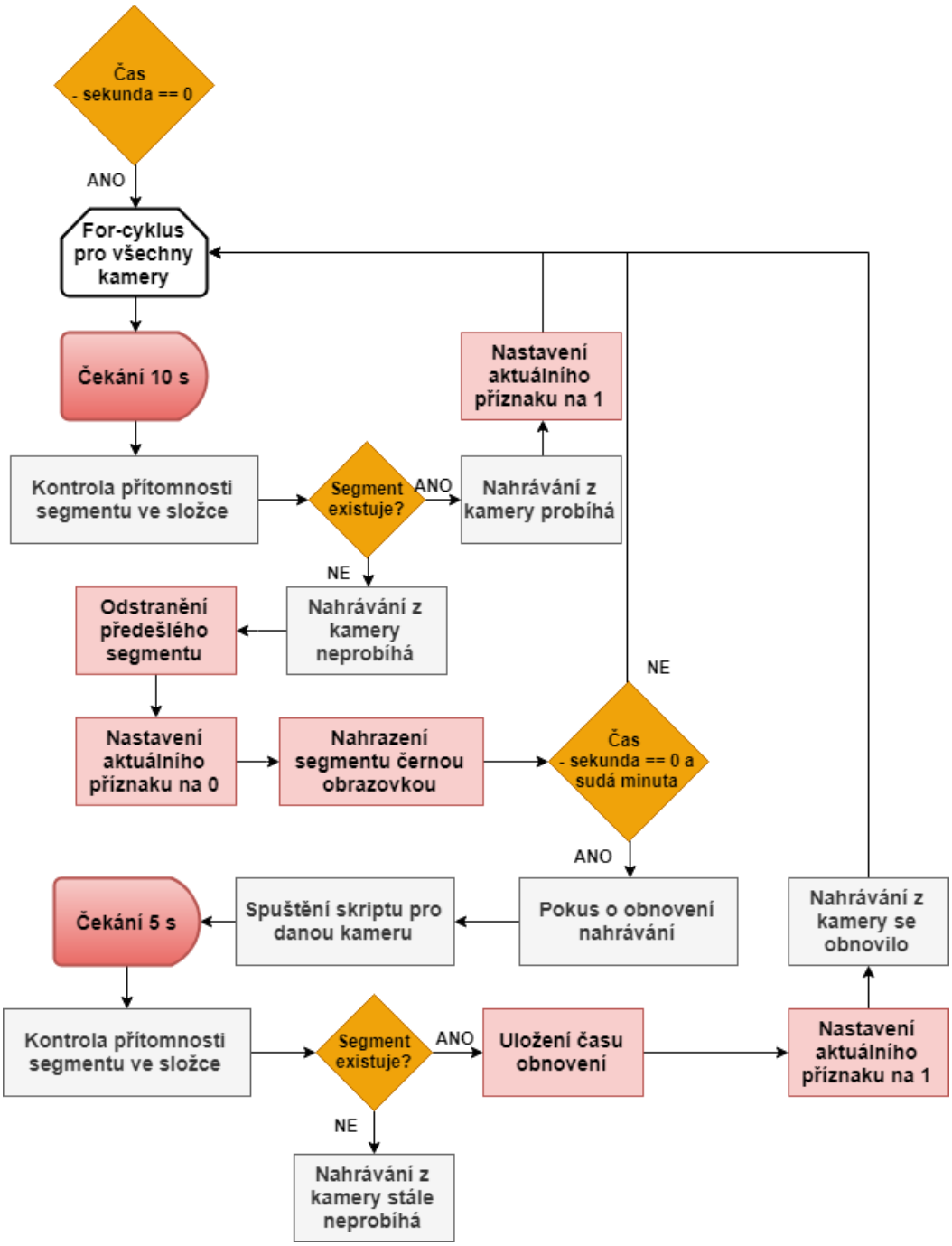
Obr. 4.11: Shell skript pro zastavení nahrávání

4.5 Monitorování stavu napájení a funkčnosti kamer

Monitorování stavu napájení a funkčnosti kamer probíhá v čekací smyčce programu, kdy běží externí procesy ffmpeg s nahráváním segmentů záznamů. Vyhodnocuje se hodnota proměnné „Status“ ve větvi switch-case, která získává informace o napájení pomocí čtení stavu příslušného registru zařízení. Pokud je napájení v pořádku, z větve se vyskočí pomocí *break* a provede se inkrementace času. Tato kontrola probíhá každý takt. Ovšem pokud je hodnota proměnné rovna 7, což je stav, kdy došlo k výpadku napájení a přepnutí napájení na záložní UPS zdroj (10 minut chodu), přepne se bool proměnná s informací o stavu napájení na hodnotu false, uloží se aktuální čas, kdy k přepnutí došlo, nastaví se nová doba trvání záznamu pro každou kameru. Poté se ukončí všechny ffmpeg procesy a zastaví se nahrávání a poté se pomocí příkazu *break* vyskočí z nahrávací čekací smyčky a program se ukončí a jednotka se vypne. Poté, co je napájení jednotky obnoveno, se jednotka opět zapne a nahrávání pokračuje dále. Až při správném běhu pak následně po ukončení nahrávání program pokračuje dále, tzn. sloučí dílčí záznamy dohromady a přesune je do složky pro daný den a smaže dílčí záznamy a vytvoří textový soubor a log-file. Výsledné záznamy jsou tak zkrácené o dobu výpadku napájení oproti požadované době (provozní den / celý den).

V čekací smyčce se zároveň volá každou minutu funkce *Funkcnost_kamer_obnoveni_kamer*, kde se testuje, zda nahrávání z každé kamery běží správně (vývojový diagram funkce je vidět na *Obr. 4.12* níže). Zjišťování funkčnosti kamery se provádí ve for-cyklu pro všechny kamery testováním příslušného souboru (segment záznamu pro danou minutu) ve složce, pokud segment existuje, nahrávání běží správně a proměnná informující o aktuální funkčnosti kamery se nastaví do 1, pokud neexistuje, nahrávání z kamery již neběží, proměnná informující o aktuální funkčnosti kamery se nastaví do 0, uloží se čas výpadku kamery (minutová přesnost) a smaže se segment záznamu, ve kterém došlo k výpadku (je tím poškozený a nešel by otevřít). Místo smazaného segmentu se pro danou minutu vytvoří 60 s prázdný záznam s černou obrazovkou, což vede k tomu, že i přes výpadek mají všechny kamery ve výsledku stejnou dobu trvání záznamu.

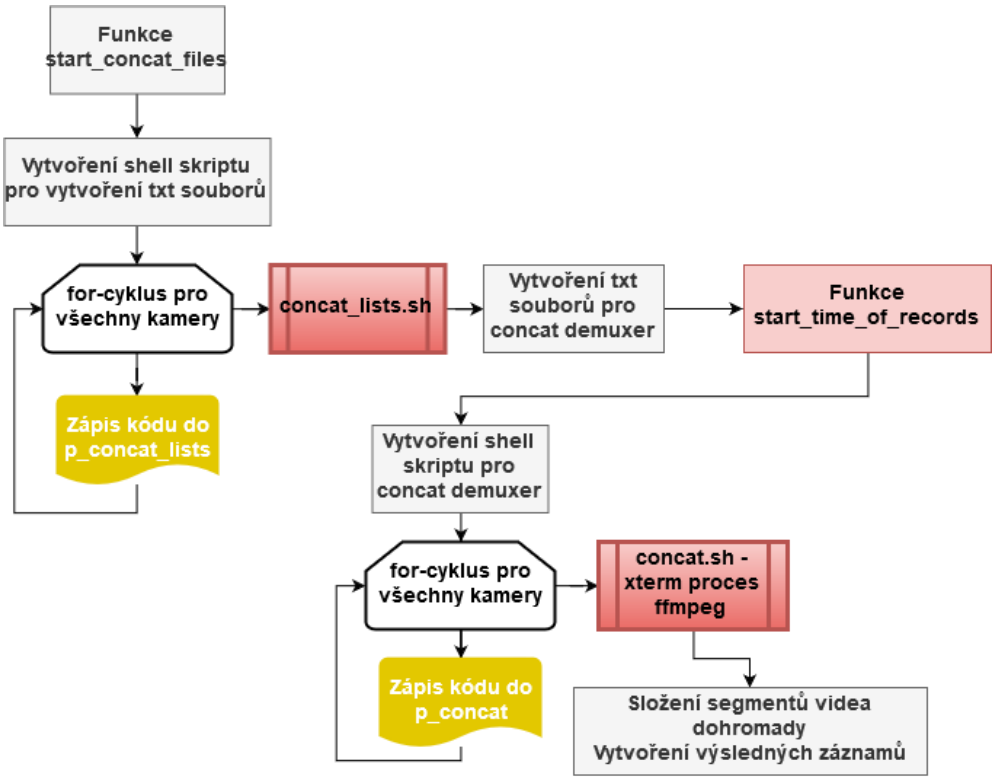
Zároveň se program každé 2 minuty pokouší obnovit nahrávání z nefunkčních kamer. Jednoduše se do shell skriptu start.sh zapíše jméno shell skriptu pro požadovanou kameru a skript se pustí a tím se pustí i nahrávání, pokud je již kamera funkční. Následně se otestuje, zda spuštění proběhlo úspěšně, pokud ano, nastaví se proměnná informující o aktuální funkčnosti kamery do 1, uloží se čas obnovení nahrávání z kamery, pokud ne, zůstane nastavená na 0 a za další 2 minuty se pokus opakuje.



Obr. 4.12: Vývojový diagram testování a obnovení funkčnosti kamer – funkce Funkcnost_kamer_obnoveni_kamer

4.6 Složení videí ve výsledné záznamy

Poté, co uplyne požadovaná doba nahrávání záznamů (případ jednodenního nahrávání), nebo uplyne celý den (případ nekonečného nahrávání), program vyskočí z čekací smyčky a pokračuje dále do fáze vytvoření textových souborů pro jednotlivé kamery a spuštění složení segmentů záznamů dohromady ve výsledné soubory, k čemuž slouží uživatelská funkce *start_concat_files*. Vývojový diagram celé funkce je zobrazen na *Obr. 4.13* později v této kapitole. Funkce slouží k vytvoření shell skriptu pro vytvoření textových souborů pro všechny kamery pro concat demuxer, který je součástí ffmpeg nástroje a který provede složení segmentů záznamů dohromady ve výsledné záznamy podle textových souborů. Požadovaný kód se do souboru zapíše pomocí *fprintf* ve for-cyklu. Výsledná podoba tohoto skriptu je vidět níže na *Obr. 4.14*. Pro každou kameru se vezmou jména všech segmentů záznamu a vloží se postupně do textového souboru – na každý řádek jedno jméno. Protože externí procesy pro ffmpeg pro každou kameru se pouští v různých vláknech paralelně, je zajištěno, že nahrávání ze všech kamer započne v tentýž okamžik. Je tedy v pořádku vzít všechny segmenty videa a není třeba první segment zahazovat. Výsledným záznamům se přiřadí do názvu správná délka trvání. Poté se skript učiní spustitelným a je spuštěn, čímž dojde k vytvoření všech textových souborů pro concat demuxer (*Obr. 4.15* níže – pouze část).



Obr. 4.13: Vývojový diagram funkce start_concat_files

```
cd /mnt/newhd/DATA/
for f in Cam1*.mp4; do echo "file $f" >> concat_cam1.txt; done
for f in Cam2*.mp4; do echo "file $f" >> concat_cam2.txt; done
for f in Cam3*.mp4; do echo "file $f" >> concat_cam3.txt; done
for f in Cam4*.mp4; do echo "file $f" >> concat_cam4.txt; done
for f in Cam5*.mp4; do echo "file $f" >> concat_cam5.txt; done
for f in Cam6*.mp4; do echo "file $f" >> concat_cam6.txt; done
for f in Cam7*.mp4; do echo "file $f" >> concat_cam7.txt; done
for f in Cam8*.mp4; do echo "file $f" >> concat_cam8.txt; done
for f in Cam9*.mp4; do echo "file $f" >> concat_cam9.txt; done
exit 0
```

Obr. 4.14: Shell skript pro vytvoření textových souborů pro concat demuxer

```
file Cam1_2019-04-29_09-27-51.mp4
file Cam1_2019-04-29_09-28-00.mp4
file Cam1_2019-04-29_09-29-00.mp4
file Cam1_2019-04-29_09-30-00.mp4
file Cam1_2019-04-29_09-31-00.mp4
file Cam1_2019-04-29_09-32-00.mp4
file Cam1_2019-04-29_09-32-59.mp4
file Cam1_2019-04-29_09-34-00.mp4
file Cam1_2019-04-29_09-35-00.mp4
file Cam1_2019-04-29_09-36-00.mp4
file Cam1_2019-04-29_09-37-00.mp4
file Cam1_2019-04-29_09-38-00.mp4
file Cam1_2019-04-29_09-39-00.mp4
file Cam1_2019-04-29_09-40-00.mp4
file Cam1_2019-04-29_09-41-00.mp4
file Cam1_2019-04-29_09-42-00.mp4
```

Obr. 4.15: Textový soubor pro concat demuxer

Následuje vytvoření shell skriptu, který spustí samotné složení segmentů pomocí ffmpeg programu. Nejdříve se zavolá funkce *start_time_of_record* pro zjištění startovních časů požadovaných zápisů (je popsána detailněji v kapitole 4.6.1) a do souboru se zapíše požadovaný kód ve for-cyklu pomocí funkce *fprintf*. Výsledná podoba tohoto shell skriptu je vidět na *Obr. 4.16* níže. Poté se soubor spustí čímž se spustí externí proces s programem ffmpeg, který zajistí vytvoření výsledných záznamů. Concat demuxer složí všechny segmenty, jejichž jména jsou v příslušném textovém souboru, dohromady za sebe ve výsledné video, které má stejné parametry kodeku, jako jednotlivé segmenty. Výsledný video záznam se jmenuje podle čísla kamery, počátečního data, času a délky záznamu.

```
cd /mnt/newhd/DATA/
ffmpeg -f concat -safe 0 -i concat_cam1.txt -c copy Camera1_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam2.txt -c copy Camera2_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam3.txt -c copy Camera3_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam4.txt -c copy Camera4_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam5.txt -c copy Camera5_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam6.txt -c copy Camera6_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam7.txt -c copy Camera7_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam8.txt -c copy Camera8_2019-4-29_9-27-51_14280.mp4
ffmpeg -f concat -safe 0 -i concat_cam9.txt -c copy Camera9_2019-4-29_9-27-51_14280.mp4
exit 0
```

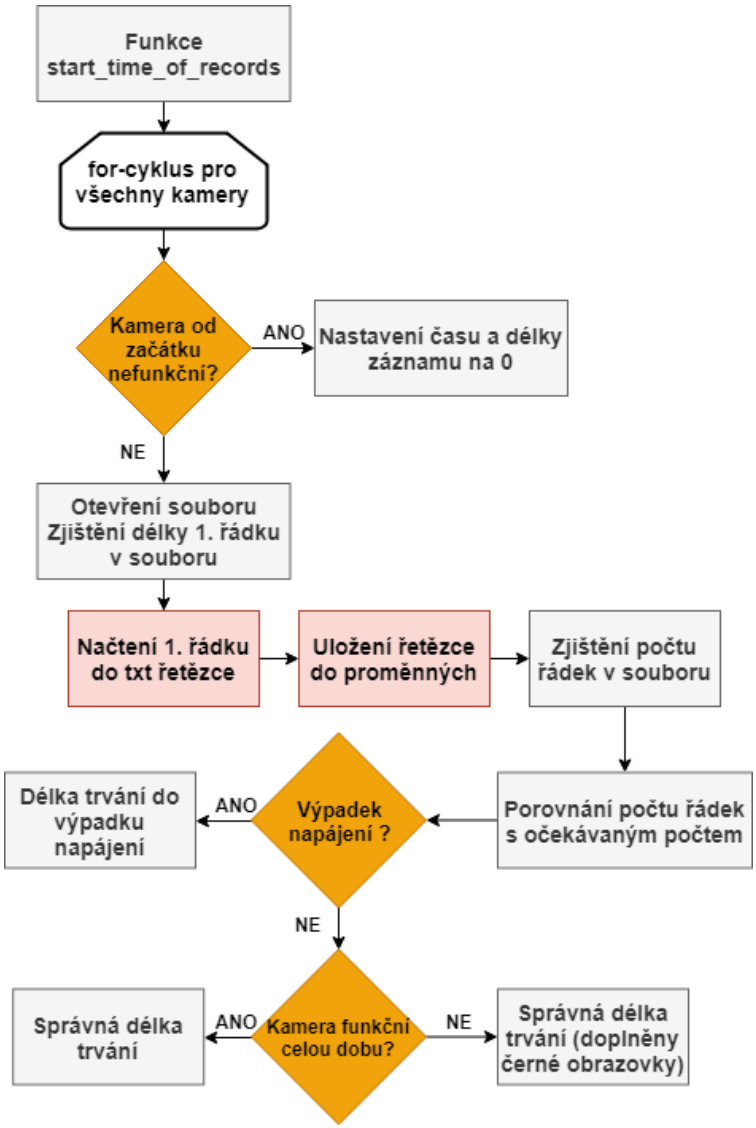
Obr. 4.16: Shell skript pro složení videí dohromady

4.6.1 Funkce pro zjištění počátečního času záznamů

Funkce *start_time_of_records* je volána v průběhu předchozí funkce po vytvoření textových souborů pro concat demuxer pro zjištění počátečních časů jednotlivých kamerových záznamů. Funkce probíhá postupně pro každou kameru pomocí for-cyklu. Pokud je kamera po celý den nahrávání nefunkční (neexistuje pro něj textový soubor pro concat demuxer), nastaví se jí počáteční čas i délka trvání na nulovou hodnotu. Její vývojový diagram je vidět na *Obr. 4.17* pod tímto odstavcem.

Nejdříve program otevře pro čtení textový soubor pro concat a zjistí délku prvního řádku v daném souboru. Pomocí funkce *fseek* se zjistí počátek souboru a postupně se do bufferu načte pomocí funkce *fread* obsah prvního řádku souboru. Obsah bufferu se uloží do jednotlivých proměnných pomocí *sscanf*. Získá se tak číslo kamery a dále počáteční rok, měsíc, den, hodina, minuta, vteřina a požadovaná délka daného záznamu. Dříve program fungoval tak, že se spočetl počet řádek v souboru – musel se shodovat s předpokládaným počtem, pokud byl nižší, znamenalo to, že došlo v průběhu nahrávání k výpadku kamery a předčasnému ukončení nahrávání z dané kamery. Kameře se přiřadila správná dobra trvání – pokud běžela správně po celu dobu, byl to požadovaný čas, pokud došlo v průběhu dne k výpadku a případně opětovnému nahození, byla to délka nahrávání z celého dne zmenšená o čas, kdy kamera nefungovala (počet segmentů se vynásobil dobou trvání segmentu), což ale vedlo k tomu, že v důsledku výpadku kamer nebyly výsledné záznamy stejně dlouhé, což znepříjemňovalo synchronní sledování záznamů. Proto je nyní při nefunkčnosti kamery její záznam nahrazen černou obrazovkou, což vede k tomu, že i při výpadcích jsou výsledné záznamy ze všech kamer stejně dlouhé a časově synchronní. Pokud došlo k výpadku napájení, je to doba nahrávání do tohoto výpadku. Časy se posléze převedou na hodnoty v sekundách. Kód obsahuje ošetření pro práci se soubory.

Nakonec se do proměnné „day_list“ zapíše pomocí funkce sprintf textový řetězec „rok-měsíc-den“. Pokud došlo k přerušení napájení a vypnutí nahrávání a jednotky, tak se pouze ukončilo nahrávání a soubor se nevytváří, vytváří se až po obnovení napájení a řádném ukončení nahrávání. Tato proměnná je později využita při vytváření výstupního textového souboru pro daný nahrávací den.



Obr. 4.17: Vývojový diagram funkce pro zjištění počátečních časů záznamů

4.7 Vytvoření výstupního textového souboru

Poté, co jsou vytvořeny všechny výsledné záznamy pro daný pracovní den vozidla, je nutno vytvořit výstupní textový soubor pro daný den obsahující názvy všech kamerových záznamů pro daný den, což se provede zavoláním funkce *create_text_file*. Pokud se soubor podaří otevřít, zapíše se do souboru požadovaný text pomocí funkce *fprintf*. Jak vypadá takový výstupní soubor, je vidět na *Obr. 4.18* pod tímto odstavcem. Může také nastat situace, kdy běží nahrávání daného pracovního dne a někdy v průběhu dojde k výpadku napájení, což vede k přepnutí záznamové jednotky na její UPS a ukončení nahrávání, v této situaci se nicméně výstupní textový soubor nevytvoří. Porucha může být opravena a později dojde opět k zapnutí záznamové jednotky a zapnutí nahrávání. Dílčí záznamy se opět nahrávají do

pracovní složky DATA na SSD disku, po uplynutí dne dojde ke sloučení záznamů a teprve pak se vytváří výstupní textový soubor.

```
Camera1_2019-4-29_9-27-51_14280
Camera2_2019-4-29_9-27-51_14280
Camera3_2019-4-29_9-27-51_14280
Camera4_2019-4-29_9-27-51_14280
Camera5_2019-4-29_9-27-51_14280
Camera6_2019-4-29_9-27-51_14280
Camera7_2019-4-29_9-27-51_14280
Camera8_2019-4-29_9-27-51_14280
Camera9_2019-4-29_9-27-51_14280
```

Obr. 4.18: Výsledný textový soubor pro daný nahrávací den

4.8 Přesun záznamů do složky pro daný den a vytvoření log-file

Když je vytvořen textový soubor pro daný nahrávací den, program vstupuje do své poslední fáze a tou je přesun výsledných záznamů pro daný den do příslušné složky, k čemuž slouží shell skript, který se vytváří ve volané funkci *move_files*. Shell skript se vytvoří pouze v případě, že nedošlo k výpadku napájení a program je na konci dne nebo provozního dne. Pokud se soubor podaří otevřít pro zápis, nejdříve se zkontroluje přítomnost výstupního souboru. Následně se pomocí funkce *fprintf* ukládají požadované příkazy do shell skriptu. Jedná se o vytvoření složky na SSD disku pro daný nahrávací den, přesunutí výsledných záznamů z pracovní složky do vytvořené složky, zkopírování výstupního textového souboru a log-file pro daný den do vytvořené složky, přesunutí výstupních textových souborů s časovými značkami každého záznamu do vytvořené složky, zkopírování konfiguračního souboru do složky, přesunutí concat textových souborů pro každý záznam do vytvořené složky a nakonec smazání všech souborů z pracovní složky DATA. Součástí skriptu je také příkaz, který na disku smaže všechny složky starší než 10 dní, což zaručuje dostatečnou kapacitu pro nové záznamy. Pokud si tedy klient o záznamy nepožádá, po 10 dnech jsou nenávratně smazány a klient je již nemůže získat. Tato doba se odvíjí od velikosti použitého SSD disku a zvolené kvality záznamů (rozlišení 800 × 450 pixelů) a počtu kamer v systému.

Poté, co je shell skript vytvořen, dojde k zavolání funkce *create_log_file* a tím k vytvoření textového souboru log-file pro daný den se všemi podstatnými informacemi – obsahuje datum nahrávacího dne, počátek záznamů, počet funkčních kamer na začátku a na konci nahrávání, informaci o čase případného výpadku napájení záznamové jednotky, informace o časech výpadků a nahození kamer a stavové informace jednotlivých kamer – číslo, IP adresa, stav na začátku a na konci, rozlišení, počátek záznamu, délka trvání záznamu.

Následně se shell skript učiní spustitelným a spustí se. Výsledný shell skript je vidět na *Obr. 4.19* pod tímto odstavcem.

```
mkdir /mnt/newhd/2019-4-29

mv /mnt/newhd/DATA/Camera*2019-4-29_9*.mp4 /mnt/newhd/2019-4-29
mv 2019-4-29.txt /mnt/newhd/2019-4-29
cp configuration.txt /mnt/newhd/2019-4-29
mv /mnt/newhd/DATA/concat*.txt /mnt/newhd/2019-4-29
mv /mnt/newhd/DATA/log_file.txt /mnt/newhd/2019-4-29
sudo rm -rf /mnt/newhd/DATA/concat*.txt &
sudo rm -rf concat.sh &
sudo rm -rf concat_lists.sh. &
sudo rm -rf /mnt/newhd/DATA/Cam1_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam2_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam3_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam4_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam5_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam6_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam7_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam8_2019-04-29*.mp4 &
sudo rm -rf /mnt/newhd/DATA/Cam9_2019-04-29*.mp4 &
```

Obr. 4.19: Výsledný shell skript pro přesun souborů

4.8.1 Ochrana SSD úložiště

U starších paměťových disků bylo potřeba řešit správný systém ukládání souborů a způsoby přemazávání, poněvadž časté mazání a přehrávání dat na určitém místě disku mohlo vést k jeho degradaci a poškození. Jednou z metod ochrany disku je metoda wear levelling, což je technika, která slouží k prodloužení životnosti přemazatelných médií, jako je flash paměť, která je použitá v SSD discích. Existuje několik mechanizmů wear levellingu opotřebení, které poskytují různé úrovně zdokonalení životnosti.

EEPROM a flash paměti mají jednotlivé segmenty, které lze vymazat, přičemž každý z nich může být omezen počtem cyklů vymazání, po němž se stává nespolehlivým. Wear levelling se pokouší obejít tato omezení uspořádáním dat tak, aby vymazání a opětovné zápisy dat byly distribuovány rovnoměrně po médiu, díky čemuž žádný segment předčasně nedegraduje vlivem vysoké koncentrace zápisových cyklů. Konvenční souborové systémy, jako jsou FAT, UFS, HFS, ext2 a NTFS, byly původně navrženy pro magnetické disky, což vedlo k častému přepisování mnoha datových struktur opakovaně ve stejné oblasti. Pokud by se tyto souborové systémy využívaly na paměťových médiích typu flash (potažmo tak na SSD discích), způsobilo by to problém. Problém je umocněn sledováním přístupových časů, což může vést k neustálému přepisování metadat souboru na místě. Paměťové médium může být buď úplně bez wear levellingu nebo může obsahovat systém dynamického wear levellingu nebo statického wear levellingu.

Nicméně moderní SSD disky využívají pro ukládání dat souborový systém ext3 nebo ext4, což je i případ SSD disku použitého v záznamové jednotce (ext3). U těchto souborových systému již není problém častého mazání a přepisování dat třeba řešit, protože si to řeší souborový systém sám, tedy nezapisuje data na stále stejné místo na disku, ale zapisuje je do náhodně zvolených segmentů a opotřebování je tak rovnoměrné.

Jelikož vždy dochází k nahrávání pracovního dne, tak i pokud by se po nahrání data odeslala a smazala a začalo se na stejné místo zapisovat znovu, dané místo na disku by se přepsalo je 365krát za rok, což je 3650 přepsání za 10 let, což by disk měl stále zvládnout. Při dnešních cenách disků je 10 let používání naprosto dostatečné.

5 Aplikace server-klient

Když dojde k nasazení kamerového systému do provozu, záznamová jednotka bude uzamčena a nebude k ní přístup pomocí grafického prostředí, takže bylo nutno vyvinout jiný způsob, jak s jednotkou komunikovat a získat z ní požadovaná data. Byl proto vytvořen server (napsaný v jazyce C), který běží nepřetržitě na záznamové jednotce, ke kterému se může připojit klient přes ethernet na portu 8080 pomocí TCP protokolu. Pro klienta byla vytvořena konzolová aplikace napsaná opět v jazyce C.

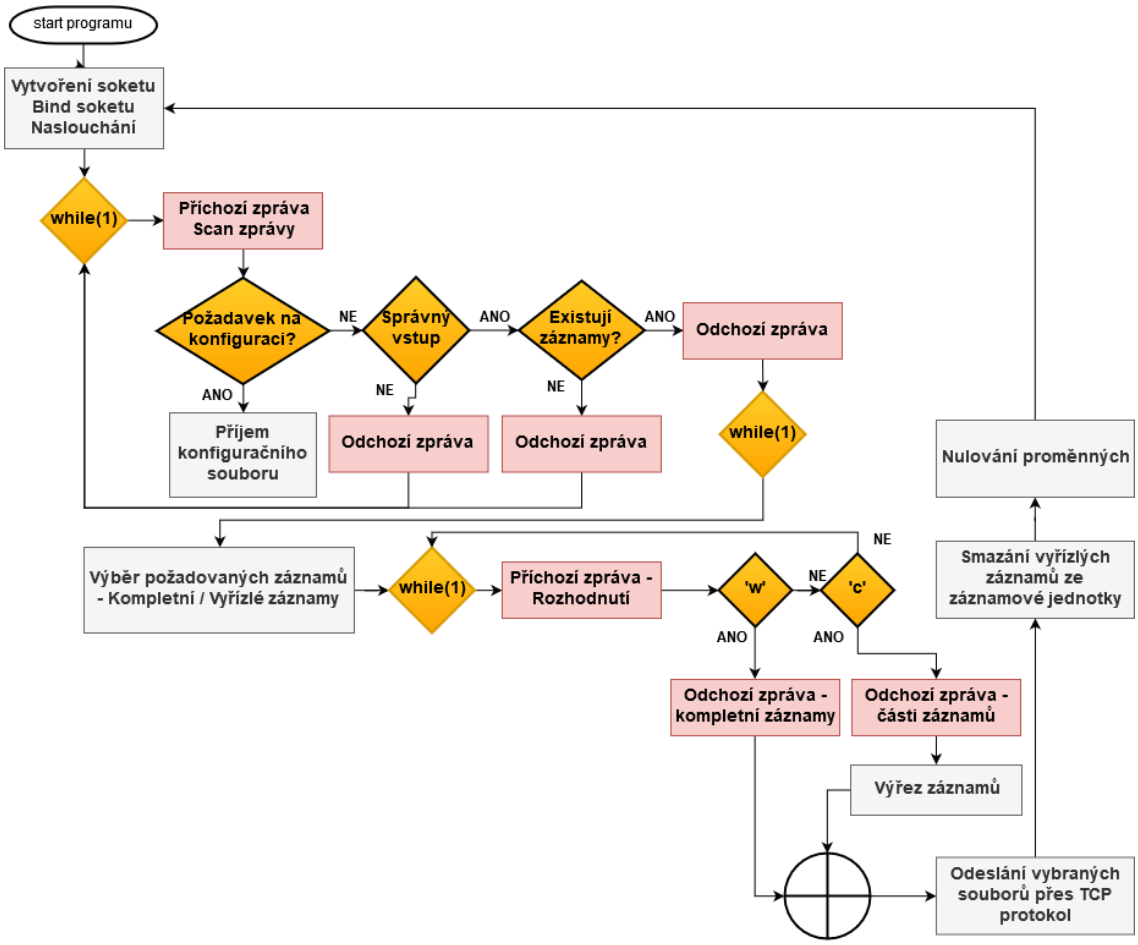
5.1 Serverová aplikace

Serverová aplikace běžící na záznamové jednotce je, stejně jako záznamová aplikace, napsána v programovacím jazyce C a je spuštěna ihned po startu záznamové jednotky pomocí shell skriptu run.sh. V následujících odstavcích je program systematicky popsán, vývojový diagram celé aplikace je vidět na *Obr. 5.1* později. Složitější části programu jsou nejdříve jen stručně komplexně popsány a následně jsou detailněji rozebrány v následujících kapitolách.

Po startu programu se nejdříve program pokouší vytvořit soket zavoláním funkce *SocketCreate*. Pokud se soket vytvoří, program pokračuje. Následně se pokouší soket spojit (anglicky bind) a to pomocí funkce *BindCreatedSocket*. Pokud se soket spojí, program pokračuje naslouchání – funkce *listen*. Poté program vstoupí do hlavní nekonečné smyčky a čeká na příchozí zprávu od klienta, což je realizováno pomocí funkce *incoming_socket*. Pokud nějaká zpráva přichází, program nejdříve akceptuje komunikaci (*accept)* a přijme zprávu a uloží jí pomocí *recv*.

První příchozí zpráva od klienta obsahuje buď požadavek na příjem nového konfiguračního souboru pro kamerový systém (textový řetězec „conf“), nebo požadovaný den, ze kterého chce získat záznamy ve formátu yyyy-mm-dd. Poté se soket uzavře a uvolní se paměť a následuje vyhodnocení vstupu. Pokud by byla příchozí zpráva rovna textovému řetězci „conf“, program vstoupí do části pro příjem konfiguračního souboru. Tato část je popsána detailně v kapitole 5.1.2. Jinak vstup musí být zadaný v přesném tvaru a musí dávat smysl. Proto je zde ošetření, aby vstup byl korektní. Pokud je vstup nekorektní, vytvoří se odchozí zpráva pro klienta pomocí funkce *outcoming_socket*, pomocí funkce *strcpy* se do proměnné zapíše požadovaná zpráva a pomocí funkce *send_message_and_free_memory* se

odešle zpráva pomocí funkce *send* klientu a program se vrací na začátek a očekává příchozí zprávu od klienta s nově zadaným požadovaným datem.



Obr. 5.1: Vývojový diagram serverové aplikace

Pokud je vstup korektní, vyhodnocuje se, zda soubor pro daný den na příslušném místě na disku existuje. Pokud neexistuje, znamená to, že pro klientem zvolený den na záznamové jednotce neexistují žádné záznamy a je vytvořena a odeslána zpráva s touto informací a požadavkem na zadání nového data. Pokud soubor existuje, znamená to, že pro vybraný den existují kamerové záznamy. Tato skutečnost je klientovi předána odesláním zprávy spolu s textovým souborem log-file se stavovými informacemi pro daný den a spolu s dotazem, jestli požaduje celé kamerové záznamy nebo bude chtít vyříznout požadovanou část. Zavolá se uživatelská funkce *start_time_of_records*, pomocí které se otevře textový soubor pro daný nahrávací den z příslušné složky, který obsahuje názvy a počáteční časy a dobu trvání všech záznamů pro daný den. Funkce je v podstatě stejná jako u nahrávací aplikace. Informace se později využijí pro ověření správnosti intervalu zadaného uživatelem pro výřez videí. Dále se po uživateli vyžaduje zadání znaku 'w' pro celkové záznamy nebo 'c' pro vyříznutí

požadované části záznamu. Zde se uživatel pomocí klienta rozhodne, zda chce celodenní záznamy pro všechny kamery nebo pouze pro jednu kameru, anebo jestli chce pouze požadovaný čas pro všechny kamery nebo pouze pro jednu kameru. Tato část programu je detailněji rozebrána v kapitole 5.1.3.

Poté, co si uživatel zvolí jednu z možností, jsou mu pomocí funkce *SendFileToClient* odeslány přes TCP protokol postupně jména a všechny požadované soubory nebo soubor. Pomocí příkazů *break* se vyskočí z nekonečných smyček, vynulují se některé stavové proměnné, smažou se případné vyříznuté záznamy ze záznamové jednotky a program pokračuje znovu od začátku.

5.1.1 Tvorba soketů, navázání komunikace

Soket je vytvořen pomocí funkce *SocketCreate*, ve které se volá systémová funkce *socket* se 3 parametry – doménou, typem a protokolem. Doména (nebo také adresová rodina) se liší podle používaného protokolu. Pro IP protokol je to AF_INET. Parametr typ určuje typ používané služby. SOCK_STREAM je přesně ten typ, který aplikace vyžaduje – služba virtuálního obvodu. Poslední parametr protokol indikuje konkrétní protokol, který se použije při podpoře provozu soketů. Pro TCP/IP sokety chceme specifikovat rodinu IP adres a službu virtuálních obvodů. Vzhledem k tomu, že existuje pouze jedna forma virtuálních obvodů, neexistují žádné varianty protokolu, takže poslední argument, protokol, je nulový.

Dalším krokem po vytvoření soketu je jeho pojmenování, což znamená, že je mu přidělena transportní adresa (číslo portu v IP síti). V angličtině se tato operace nazývá „binding“ a je pro ni použita systémová funkce *bind*. Transportní adresa je definována v adresové struktuře soketu. Jelikož jsou sokety navrženy tak, aby mohly spolupracovat s různými typy komunikačního rozhraní, je rozhraní velmi obecné. Místo, aby se soketu jednoduše přiřadilo číslo portu, je potřeba struktura „sockaddr“, jejíž skutečný formát je určen dle rodiny adres, která je použita. Tato operace je realizována funkcí *BindCreatedSocket*.

Prvním parametrem pro systémovou funkci *bind*, je soket vytvořený v předešlém kroku. Druhým parametrem je struktura „sockaddr“, která je obecným kontejnerem, který jednoduše dovoluje operačnímu systému číst prvních pár bytů, které identifikují rodinu adres. Rodina adres určuje, jaká varianta struktury bude použita. Pro sítě IP se používá struktura sockaddr_in definována v záhlaví <netinet/in.h>. Před samotným zavoláním funkce *bind* je

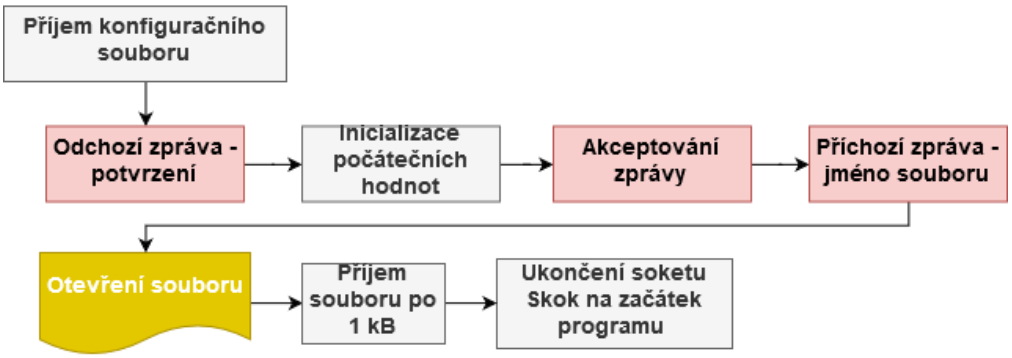
nutno strukturu naplnit. Je jí přiřazena AF_INET rodina adres, číslo portu 8080 a také adresa soketu, což je IP adresa záznamové jednotky, parametrem INADDR_ANY říkáme, že se nestaráme o specifikaci konkrétního rozhraní a necháme operační systém používat, co chce. Protože se struktura adresy může lišit podle typu použitého transportu, třetí parametr definuje délku použité struktury. Předtím, než se klient k serveru může připojit, je nutné, aby server měl soket, který je připraven k přijetí připojení od klienta. K tomu slouží systémová funkce *listen*, která říká soketu, že má být připraven k přijetí připojení.

Po těchto krocích tedy je soket vytvořen, pojmenován a je mu řečeno, že může přijímat příchozí připojení. Příchozí připojení je akceptováno pomocí systémové funkce *accept*. Čekání na příchozí připojení, jeho akceptování a přijetí zprávy je v serverové aplikaci realizováno funkcí *incoming_socket*. Když je příchozí zpráva od klienta úspěšně přijata, je pomocí funkce *recv* uložena. Po přijetí a uložení zprávy je nutno uvolnit alokovanou paměť a soket uzavřít pomocí funkce *close*. Pokud není vyžadováno přijetí zprávy od klienta, ale naopak server chce poslat zprávu s odpovědí klientovi, postupuje se obdobně. V serverové aplikaci k odeslání zprávy slouží dvě funkce. První z nich je funkce *outcoming_socket*, se vstupním int parametrem, který udává délku odchozí zprávy. Ve funkci je akceptován soket pomocí funkce *accept* a poté je k odeslání použita druhá funkce *send_message_and_free_memory*, ve které je provedeno odeslání zprávy pomocí systémové funkce *send*. Poté je opět uvolněna paměť a soket je uzavřen pomocí *close*.

5.1.2 Příjem konfiguračního souboru

Pokud je první příchozí zpráva od klienta rovna textovému řetězci „conf“, znamená to, že klient chce serveru, a tedy záznamové jednotce, zaslat nový textový konfigurační soubor pro rekonfiguraci kamerového systému. Vývojový diagram požadavku je vidět na *Obr. 5.2* níže.

Pomocí funkce *outcoming_socket* je vytvořen soket pro odchozí zprávu se sdělením klientu, že požadavek na příjem souboru je akceptován a pomocí funkce *send_message_and_free_memory* je zpráva odeslána. Následuje akceptování příchozí zprávy od klienta a pokud je úspěšné, pomocí funkce *recv* se přijme ze zprávy jméno souboru. Ověří se řádné otevření souboru a dojde ke vstupu programu do while-smyčky, která trvá, dokud probíhá příjem dat souboru. Dokud přicházejí nějaká data, jsou do souboru zapisována pomocí funkce *fwrite*. Až jsou zapsána všechna data a další již nepřicházejí, soubor a soket je uzavřen a program pokračuje od začátku příkazem *continue*.



Obr. 5.2: Vývojový diagram pro příjem konfiguračního souboru

5.1.3 Výběr požadovaného záznamu

Poté, co server vytvořil soket, pojmenoval ho a určil ho k přijetí soketu od klienta a ten odeslal zprávu se správným požadovaným datem pro získání záznamů z kamer, což znamená, že datum bylo zadáno v korektní podobě yyyy-mm-dd a zároveň pro požadované datum existují v záznamové jednotce kamerové záznamy, program vstoupí do fáze, kdy nejprve server odešle klientu textový log_file soubor se všemi stavovými daty pro zvolený den informující o funkčních kamerách, jejich parametrech a případných výpadcích. Poté klient musí zvolit, jestli požaduje celé záznamy ze všech kamer nebo pouze z jedné kamery, anebo pouze části záznamů ze všech kamer či z jedné kamery. Program vstoupí do dalšího nekonečného while-cyklu, kde se čeká na příchozí zprávu od klienta s požadovaným znakem. Pokud je příchozí znak špatný, je vytvořena odchozí zpráva a tato informace je klientovi odeslána a čeká se na novou příchozí zprávu od klienta s novým znakem.

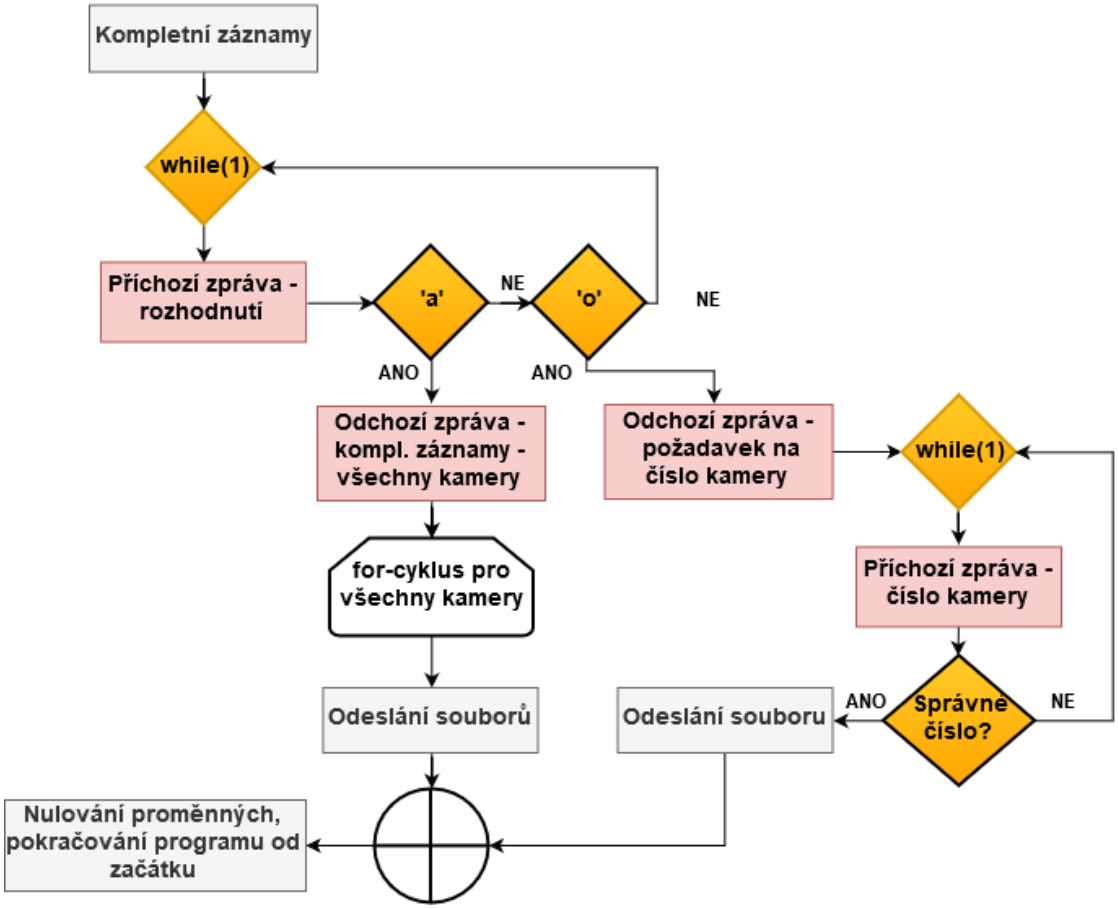
5.1.3.1 Požadavek na celodenní kompletní záznamy

Pokud je příchozí znak 'w', znamená to požadavek od klienta na záznam/záznamy z celého provozního dne vozidla. Kompletní vývojový diagram pro tuto situaci je zobrazen na *Obr. 5.3* na konci této kapitoly. Je vytvořena odchozí zpráva s informací o výběru možnosti pro celodenní záznamy a s dotazem pro klienta, zda chce obdržet záznamy ze všech kamer pro daný den nebo pouze z jedné vybrané kamery, zpráva je odeslána a čeká se v dalším while-cyklu na odpověď klienta se znakem 'a' znamenající požadavek na všechny záznamy nebo se znakem 'o' znamenající požadavek na celodenní záznam z jedné kamery.

Pokud přijde znak ‚a‘, znamená to, že uživatel chce všechny celodenní záznamy. Pomocí for-cyklu pro všechny kamery se postupně pomocí funkce *SendFileToClient* odešle jméno souboru a poté celý soubor pro každou kameru. Pomocí příkazů *break* se vyskočí

z nekonečných smyček, vynulují se stavové proměnné a program pokračuje znovu od začátku. Kód obsahuje všechna ošetření pro zadání správného vstupu, správného otevření a uzavření souborů a kontrolu odeslání souborů.

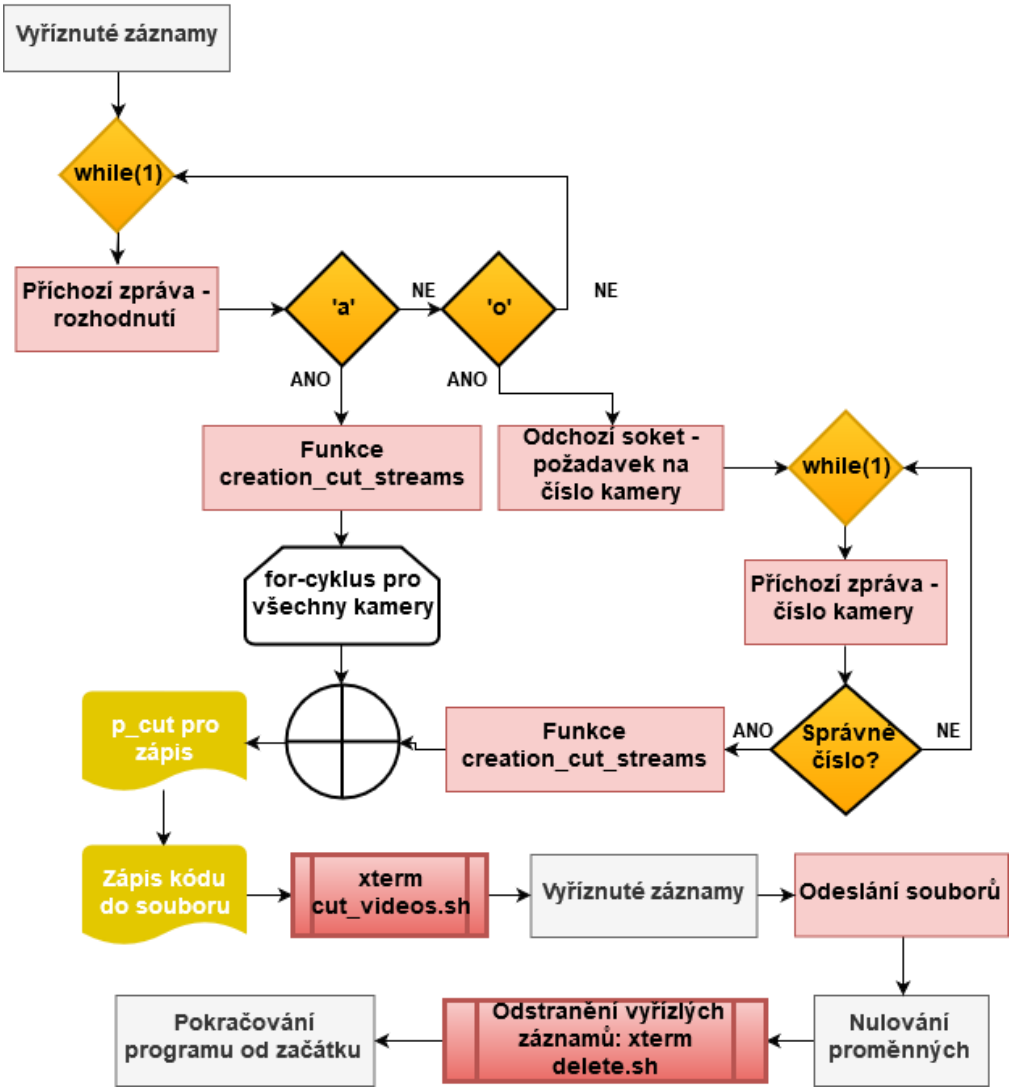
Pokud přichízí znak ‚o‘, znamená to, že uživatel chce celodenní záznam pouze z jedné kamery. Odešle se dotaz na číslo kamery, jakou klient vyžaduje pro celodenní záznam a program vstoupí do nekonečného while-cyklu, kde čeká na reakci klienta s požadovaným číslem kamery. Příchozí číslo musí být přirozené číslo od 1 do 9. Pokud je příchozí číslo kamery správné, pomocí funkce *SendFileToClient* se odešle klientu jméno souboru a poté celý soubor pro danou kameru. Pomocí příkazů *break* se vyskočí z nekonečných smyček, vynulují se stavové proměnné a program pokračuje od začátku.



Obr. 5.3: Vývojový diagram pro výběr požadavku pro kompletní celodenní záznamy

5.1.3.2 Požadavek na části záznamů

Pokud je příchozí znak 'c', znamená to požadavek od klienta na záznam/záznamy pouze pro určitý čas provozního dne vozidla. Kompletní vývojový diagram pro tuto situaci je zobrazen na *Obr. 5.4* pod tímto odstavcem.



Obr. 5.4: Vývojový diagram pro výběr požadavku pro vyříznutí částí záznamů

Je vytvořena odchozí zpráva s informací o výběru možnosti pro části záznamů a s dotazem pro klienta, zda chce obdržet části záznamů ze všech kamer pro daný den nebo pouze z jedné vybrané kamery, zpráva je odeslána a čeká se v dalším nekonečném while-cyklu na odpověď klienta se znakem 'a' znamenající požadavek na část záznamů ze všech kamer nebo se znakem 'o' znamenající požadavek na část záznamu z jedné kamery. Pokud přijde znak 'a', znamená to, že uživatel chce části záznamů ze všech kamer. Je zavolána

uživatelská funkce *creation_cut_streams*, která slouží k zadání požadovaných časů záznamů klientem. Funkce je podrobněji popsána v následující kapitole. Poté se pomocí for-cyklu pro všechny kamery naplní struktura stejnými hodnotami a vyskočí se ze smyčky.

Pokud přijde znak 'o', znamená to, že uživatel chce část záznamu pouze z jedné kamery. Odešle se zpráva s požadavkem na číslo kamery. Program vstoupí do další nekonečné smyčky a očekává příchozí zprávu s číslem kamery. Vstupem musí být přirozené číslo od 1 do 9. Poté, co přijde správný požadavek na číslo kamery, je zavolána uživatelská funkce *creation_cut_streams*, která slouží k zadání požadovaných časů záznamu klientem opět pomocí soketů, vyskočí se ze smyčky pomocí příkazu *break*.

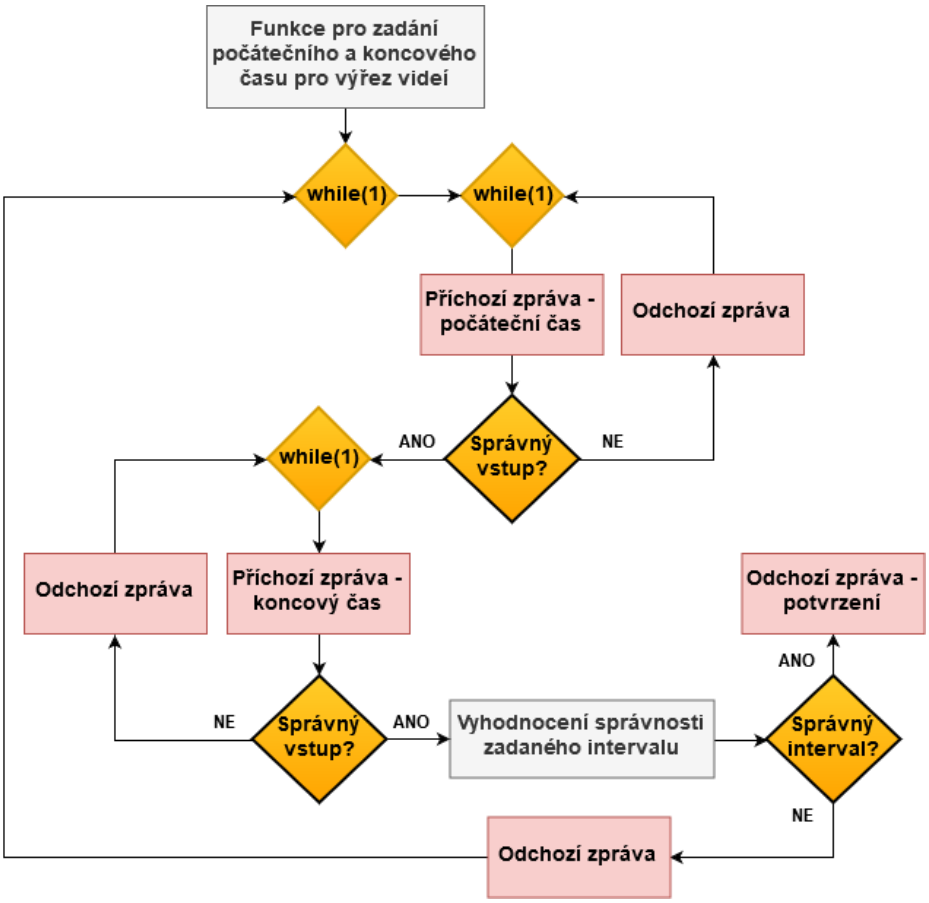
Ať už uživatel požaduje vyříznuté záznamy ze všech kamer, nebo pouze z jedné konkrétní, program pokračuje skoro stejně. Soubor pro vyříznutí videí je otevřen pro zápis a zapíše se do něj pomocí funkce *fprintf* požadovaný kód pro ffmpeg (buď pro všechny kamery, nebo pouze pro jednu kameru), který posléze provede výřez videí. Shell skript pro výřez videí pro konkrétní časy pro 1 kameru je vidět na *Obr. 5.5* pod tímto odstavcem. Skript se spustí, čímž se spustí proces s ffmpeg, který provede výřez požadovaných videí podle požadovaných časů. Následně se zavolá funkce *SendFileToClient*, která provede odeslání jmen souborů a souborů klientu. Pomocí příkazů *break* se vyskočí z nekonečných smyček, vynulují se některé stavové proměnné a spustí se shell skript delete.sh, který provede smazání. Program pokračuje znovu od začátku.

```
ffmpeg -ss 6360 -i /mnt/newhd/2019-1-8/Camera2_2019-1-8_15-14-0_57600.mp4
-t 3600 -vcodec copy -acodec copy -y
/mnt/newhd/2019-1-8/required_cut_Camera2_2019-1-8_17-00-00_18-00-00.mp4
```

Obr. 5.5: Výsledný shell skript pro výřez videí pomocí ffmpeg pro 1 kameru a konkrétní časy

5.1.4 Funkce pro zadávání časů pro výřez

Když uživatel pomocí klienta zadá správné datum, pro které existují záznamy a zvolí možnost získání pouze částí celodenních záznamů, je nutné zvolit počáteční a koncový čas požadovaného záznamu. K tomu slouží uživatelská funkce *creation_cut_streams*, která se volá s různými parametry v závislosti na tom, zda uživatel požaduje vyříznout záznamy pouze pro jednu kameru nebo pro všechny kamery. Vývojový diagram této funkce je zobrazen na *Obr. 5.6* pod tímto odstavcem.



Obr. 5.6: Vývojový diagram funkce pro zadání počátečního a koncového času pro výřez videí

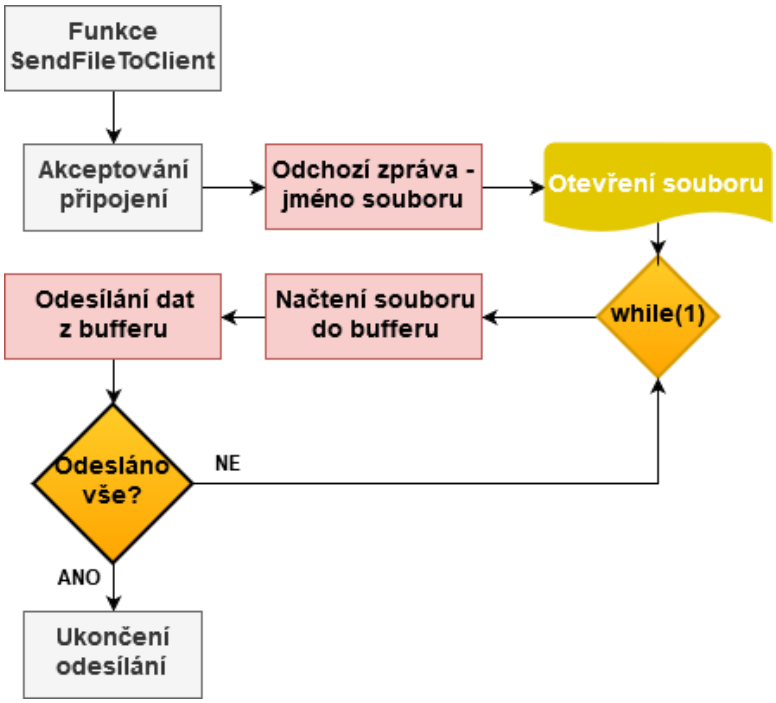
Poté, co se zavolá funkce, je klientu odeslána zpráva s požadavkem na zadání počátečního času ve tvaru hh-mm-ss a program vstoupí do své nekonečné smyčky a ihned do další nekonečné smyčky pro zadání počátečního času. Čeká se na odpověď klienta s požadovaným časem a vyhodnocuje se jeho správnost, příchozí vstup musí být ve správném tvaru a musí dávat smysl. Pokud je počáteční čas správný, program vyskočí ze smyčky a je odesláno klientu potvrzení a požadavek na zadání koncového času. Program posléze vstoupí do další nekonečné smyčky pro zadání koncového času, kde čeká na odpověď klienta s požadovaným časem a vyhodnocuje se jeho správnost. Pokud je koncový čas správný, program vyskočí ze smyčky a je odesláno klientu potvrzení.

Když jsou oba časy zadány správně, vyhodnocuje se jejich správnost z logického hlediska. Koncový čas musí být samozřejmě větší nežli počáteční čas a zároveň zadaný interval musí náležet do časového intervalu kamerového záznamu. Pokud je nějaká podmínka nesplněna, je klientu tato informace odeslána a funkce se vrací na začátek a opět je požadováno zadání počátečního a koncového času. Když je zadaný interval správný, program

vyskočí ze smyčky a je odeslána potvrzující zpráva o správnosti zadaného intervalu a funkce se ukončí a program pokračuje odesíláním souborů.

5.1.5 Funkce pro odesílání souborů

Když si uživatel vybral, jaké záznamy chce obdržet, následuje jejich odeslání, které se provede pomocí funkce *SendFileToClient*. Vývojový diagram celé funkce je zobrazen pod tímto odstavcem na *Obr. 5.7*. Na začátku je nutno vytvořit spojení mezi serverem a klientem. Poté, co je spojení správně navázáno, se klientu odešle pomocí funkce *write* jméno odesílaného souboru. Funkce pokračuje vstupem do nekonečné smyčky. V ní se plní buffer daty souboru pomocí funkce *fread* a dochází k odesílání dat pomocí funkce *write*. Odesílání probíhá až do té doby, dokud jsou k dispozici nějaká data. Až se dojde na konec souboru, program vyskočí ze smyčky, uzavře se soket a funkce se ukončí.

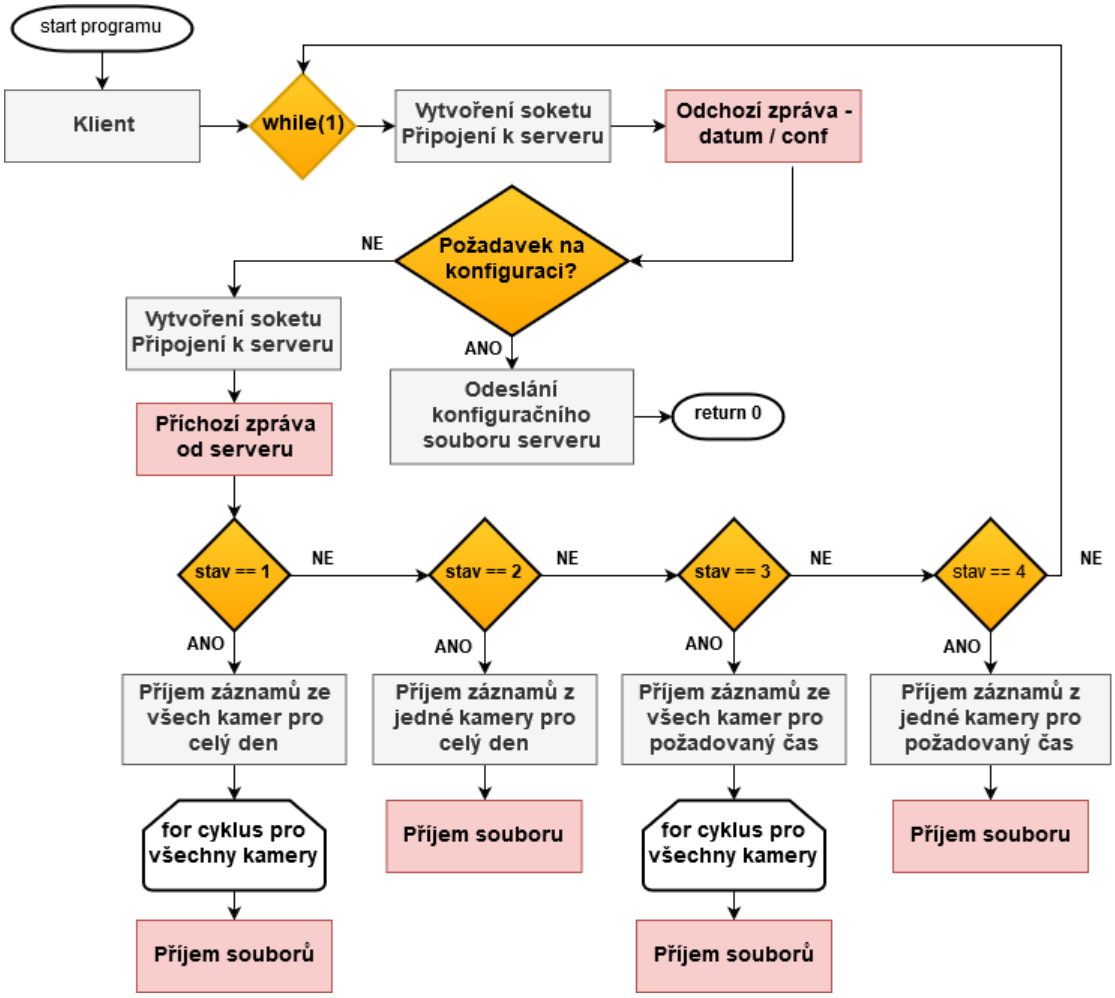


Obr. 5.7: Vývojový diagram funkce pro posílání souborů

5.2 Klient

Pro navázání komunikace se serverem běžícím automaticky na záznamové jednotce je potřeba nějaký klient, který je schopen se k serveru připojit. Tento klient je konsolová aplikace napsaná v jazyce C. Klient posílá serveru požadavky – buď mu může odeslat konfigurační soubor s novým nastavením kamerového systému, nebo může vyžadovat příjem záznamů. Může požadovat všechny celodenní záznamy ze všech kamer nebo pouze z jedné

vybrané kamery, nebo může odeslat požadavek pouze na omezený časový interval pro všechny kamery nebo pouze pro jednu kameru. Server pak provede požadovaný výřez a záznamy odešle. Vývojový diagram celé aplikace je vidět na *Obr. 5.8* níže.



Obr. 5.8: Vývojový diagram aplikace klient

Program je realizován v nekonečné smyčce. Nejdříve se klient pokusí vytvořit soket pomocí funkce *Socket_creation*. Pokud se soket podařilo vytvořit, zavolá se funkce *SocketConnect* pro navázání spojení se serverem, po úspěšném připojení se vypíše informace na konzoli a program pokračuje. Jsou zde ošetřeny všechny možnosti – tedy neúspěch vytvoření soketu a selhání připojení k serveru.

Následuje část, kde se odesílá zpráva serveru. Uživatel zadá zprávu a ta se z konzole načte pomocí funkce *gets*. Od uživatele se očekává buď požadavek na datum, ze kterého chce záznamy ve tvaru „yyyy-mm-dd“, nebo řetězec „conf,“ který vede k odeslání konfiguračního souboru serveru pro nové nastavení kamerového systému. Ať uživatel zadá správný vstup,

nebo špatný, zpráva se odešle serveru pomocí funkce *SocketSend* přes TCP protokol. Pokud byl odeslán řetězec „conf," program vstoupí do části pro odeslání konfiguračního souboru, která je detailněji rozebrána v kapitole 5.2.2. Pakliže uživatel zadal požadované datum, případně nějaký nesmyslný vstup, dojde k uvolnění paměti a uzavření soketu a program očekává příchozí odpověď serveru. Vytvoří se nový soket a spojí se serverem a pomocí funkce *SocketReceive* se přijme odpověď serveru. Tato odpověď se pomocí *printf* vypíše na konzoli. Z příchozí zprávy se pomocí *sscanf* skenuje do proměnné „stav" hodnota ze zprávy. Dojde k uvolnění paměti a uzavření soketu. Pokud proměnná „stav" není rovna 1 nebo 2 nebo 3 nebo 4 znamená to, že nedojde k přenosu souborů ze serveru a program je stále ve fázi komunikace, vrátí se zpět na začátek smyčky, klient odešle serveru žádost, server pošle odpověď, a tak to pokračuje, dokud server neodešle soubory klientu.

Pokud je „stav" roven 1, dojde k přenosu celodenních záznamů ze všech kamer, pokud je roven 2, dojde k přenosu celodenního záznamu z jedné vybrané kamery. Jestliže je „stav" roven 3, dojde k přenosu požadovaných částí záznamů ze všech kamer. Pokud je „stav" roven 4, dojde k přenosu pouze části záznamu z jedné vybrané kamery. Příjem souborů je realizován funkcí *receive_files_from_server*. Funkce pro příjem souborů je rozebrána v kapitole 5.2.3.

5.2.1 Vytvoření soketu, navázání spojení

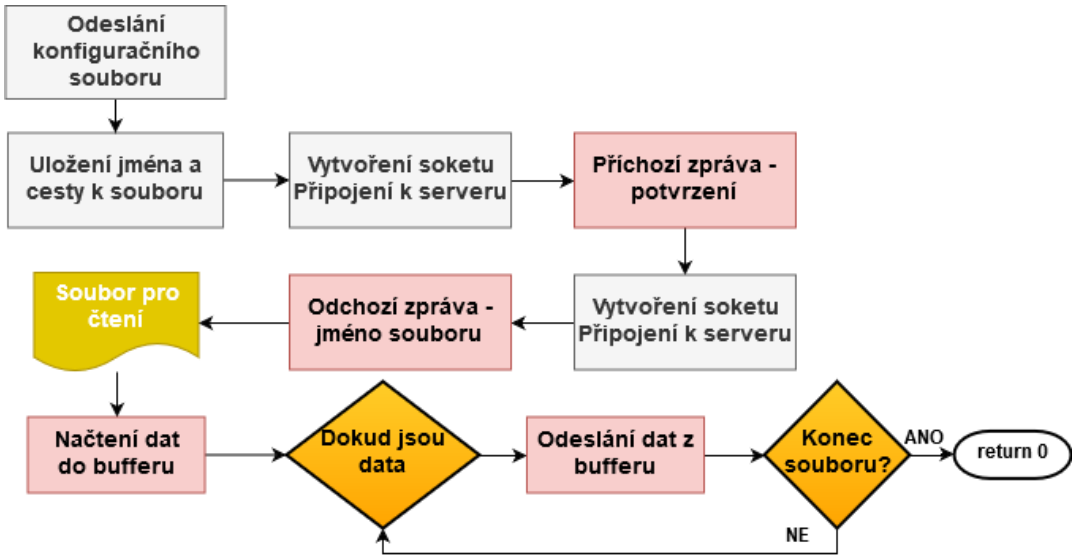
K vytvoření soketu v aplikaci klienta slouží funkce *Socket_creation,* vnitřně volá funkci *socket* s parametry AF_INET, SOCK_STREAM, 0, což je stejné, jako v případě serveru. K navázání spojení se serverem slouží funkce *SocketConnect.* Uvnitř funkce je zadáno číslo portu, přes které se komunikuje se serverem, v našem případě se jedná o port číslo 8080. Taktéž je zde deklarována struktura sockaddr_in, která je naplněna správnými hodnotami – pomocí funkce inet_addr je zadána IP adresa serveru, jako adresová rodina je zvolena rodina AF_INET a pomocí funkce *htons* je zadáno číslo portu pro komunikaci. Uvnitř se volá systémová funkce *connet pro* navázání spojení.

Pro posílání zprávy z klienta na server slouží funkce *SocketSend,* kde vstupy jsou požadovaný soket, požadovaná zpráva k odeslání a délka zprávy. Ve funkci je nastaven timeout 20 s pro maximální dobu pro odeslání zprávy. Pokud je vše v pořádku, provede se odeslání zprávy pomocí funkce *send.* K přijímání zprávy od serveru slouží funkce *SocketReceive.* Stejně jako u funkce pro odesílání zprávy, je i zde struktura typu timeval pro

vyhodnocení uplynutí definovaného času (timeout = 20 s) pro přijetí zprávy. Pokud je vše v pořádku, zpráva se přijme pomocí *recv*.

5.2.2 Odeslání konfiguračního souboru

Pakliže je první zprávou od klienta serveru odeslán textový řetězec „conf“, program klienta vstoupí do fáze, ve které se provede odeslání konfiguračního souboru pro kamerový systém. V programu musí být zadáno správné jméno a cesta k příslušnému souboru. Celý vývojový diagram této části programu je zobrazen na *Obr. 5.9* pod tímto odstavcem.



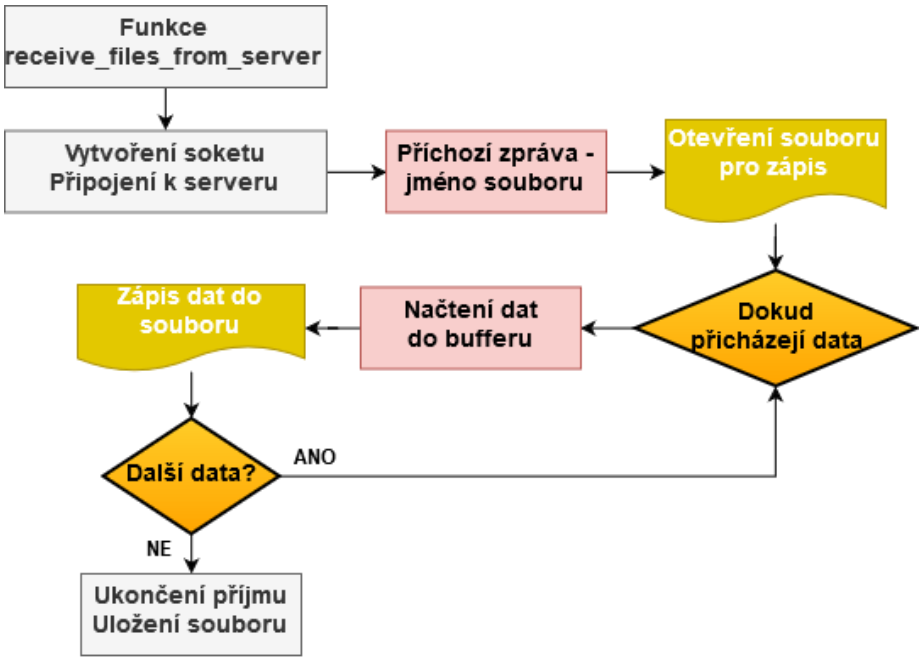
Obr. 5.9: Vývojový diagram pro odeslání konfiguračního souboru

Vytvoří se soket a naváže se připojení se serverem, pokud je k dispozici. Program obsahuje nezbytná ošetření pro výjimky. Pokud se klient připojí k serveru, klient přijme odpověď serveru s potvrzením přijmutí konfiguračního souboru. Vytvoří se nový soket a klient odešle nejdříve jméno souboru a poté následuje odeslání daného souboru. Ten se nejdříve otevře pro čtení a pomocí funkce *fread* se načte do bufferu a posléze ve while-cyklu, který trvá, dokud jsou data k dispozici, se provádí odesílání dat z bufferu pomocí *send*, až se dojde na konec souboru, z cyklu se vyskočí a odesílání se ukončí a program skončí s 0.

5.2.3 Funkce pro příjem souborů

Jestliže klient neprovedl odeslání konfiguračního souboru, po výměně zpráv mezi klientem a serverem o požadavcích dojde k odeslání požadovaných souborů serverem a klient

je musí přijmout. K tomu je nadefinovaná funkce *receive_files_from_server*. Vývojový diagram této funkce je zobrazen na pod tímto odstavcem na *Obr. 5.10*.



Obr. 5.10: Vývojový diagram funkce pro příjem souborů ze serveru

Nejdříve se vytvoří soket pomocí funkce *Socket_creation*, samozřejmě s testováním úspěšnosti vytvoření, poté se klient zkouší připojit k serveru, opět se testuje úspěšnost připojení a pokud je vše v pořádku, pomocí funkce *read* dojde k přečtení příchozí zprávy ze serveru se jménem souboru a uložení tohoto jména. Soubor se otevře pro zápis a pokud se podaří soubor řádně otevřít, následuje while-cyklus, ve kterém dochází k příjmu dat ze serveru – data se získávají pomocí funkce *read* a ukládají se do bufferu. Dokud přicházejí nějaká data, z bufferu se zapíší do souboru pomocí funkce *fwrite*. Až jsou přijata a zapsána všechna data, smyčka se ukončí, dojde k uzavření souboru a soketu a k vyskočení z funkce.

Závěr

V této diplomové práci jsem provedl rešerši analogových a zejména digitálních IP kamer, od stručné historie jejich použití, přes jejich principy, rozdělení, výhody a nevýhody, po využívaná přenosová média a komunikační protokoly. Dále jsem popsal formát kódování videa H.264 a kontejner MP4. Následně jsem vytvořil fungující model kamerového systému, skládající se ze záznamové jednotky, napájecího zdroje, PoE switche a devíti IP kamer. Všechny komponenty jsem popsal, rozebral jejich vlastnosti a funkce. Po sestavení jsem systém zprovoznil – na záznamovou jednotku jsem nainstaloval OS a nakonfiguroval ji, IP kamerám jsem přiřadil statické IP adresy, manuálně je zaostřil, provedl jsem nutné nastavení jejich parametrů a nastavil jejich časovou synchronizaci se záznamovou jednotkou.

Po zprovoznění systému přišel na řadu samotný vývoj řídícího software. Aplikace psaná v jazyce C umožňuje konfiguraci kamerového systému s ohledem na počet kamer v systému, požadované rozlišení a délku požadovaného záznamu. Pro samotné nahrávání záznamů z kamer jsem využil opensource program ffmpeg, který je volán z aplikace pro každou kameru. Synchronizace jednotlivých záznamů je zajištěna nastavením parametrů pro ffmpeg, které bylo zjištěno experimentálně. V průběhu nahrávání je monitorován stav napájení, výpadek a přepnutí na záložní UPS je vyhodnoceno a nahrávání je ukončeno a data jsou uložena. Aplikace umí vyhodnotit (ne)funkčnost kamery, její výpadek během nahrávání a cyklicky se snaží nahrávání při výpadku obnovit. Výstupem je textový soubor s počátečními časy a délkou trvání záznamů a textový log-file obsahující stavová data o funkčnosti kamer a výpadcích. Výsledné záznamy jsou ukládány na SSD disk do složky pro daný den. Následně jsem vytvořil konzolovou aplikaci server-klient, sloužící pro získání požadovaných dat ze záznamové jednotky pomocí posílání zpráv a dat přes TCP protokol. Klient může jednotce odeslat konfigurační soubor nebo od jednotky požadovat data pro daný den, buď ze všech kamer, nebo pouze z jedné konkrétní. Může žádat buď celodenní záznamy, nebo pouze jejich části. Jednotka se postará o vyříznutí záznamů a odešle klientu celé, nebo částečné záznamy.

Vytvořil jsem funkční kamerový systém zajišťující nahrávání videa z 9 IP kamer automaticky po zapnutí záznamové jednotky, který umí řešit výpadky a ukládá stavová data. Umí na požádání odeslat data a záznamy, takže po dořešení komunikace s řídícím počítačem nebo informačním systém může nalézt uplatnění v nasazení do nějakého kolejového vozidla.

Seznam literatury a informačních zdrojů

[1] ADOLF, Jindřich. *Porovnání účinnosti moderních standardů pro kódování videa.* Bakalářská práce, České vysoké učení technické v Praze, Praha, 2014.

[2] *An Incredibly Unboring History of IP Cameras*. ProtectAmerica [online]. 2016, 09.12.2016 [cit. 2019-03-10]. Dostupné z: https://www.protectamerica.com/home-security-blog/tech-tips/draft-an-incredibly-unboring-history-of-ip-cameras-draft_11713

[3] *An Overview of H.264 Advanced Video Coding*. VCODEX [online]. [cit. 2019-03-10]. Dostupné z: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/

[4] *AVI vs MP4, rozdíl mezi AVI a MP4*. Wondershare [online]. [cit. 2019-03-10]. Dostupné z: http://cs.wondershare.com/mp4/avi-vs-mp4.html

[5] *AXIS P3904-R Mk II Network Camera: Datasheet and User Manual*. AXIS Communications [online]. [cit. 2019-03-10]. Dostupné z: https://www.axis.com/products/axis-p3904r-mkii/support-and-documentation

[6] BALA, Sid. *H.264 is Magic*. Sidbala.com [online]. 02.11.2016 [cit. 2019-03-10]. Dostupné z: https://sidbala.com/h-264-is-magic/

[7] BÍLEK, Jan. *Moderní video kodeky.* Bakalářská práce, Vysoké učení technické v Brně, Brno, 2015.

[8] *Co je formát MP4*. Apowersoft [online]. 14.01.2016 [cit. 2019-03-10]. Dostupné z: https://www.apowersoft.cz/co-je-format-mp4.html

[9] DELGADO, Rick. *From Edison to internet: A history of video surveillance*. Business 2 Community [online]. 2013, 14.08.2013 [cit. 2019-03-10]. Dostupné z: https://www.business2community.com/tech-gadgets/from-edison-to-internet-a-history-of-video-surveillance-0578308

[10] *Encyklopedie síťového videa*. Netcam.cz [online]. 02.11.2018 [cit. 2019-03-10]. Dostupné z: https://netcam.cz/reseni.php

[11] *Ffmpeg Documentation*. [online]. [cit. 2019-03-10]. Dostupné z: https://ffmpeg.org/ffmpeg-all.html

[12] *GSW-1600HP*. PLANET Networking and Communication [online]. [cit. 2019-03-10]. Dostupné z: https://www.planet.com.tw/en/product/gsw-1600hp-v2

[13] *H.264: Jeden standard pro všechna videa*. CHIP [online]. 09.06.2009 [cit. 2019-03-10]. Dostupné z: https://www.chip.cz/casopis-chip/earchiv/vydani/r-2009/chip-04-2009/h-264/

[14] *H264 profiles and levels*. MediaCoder [online]. 2008, 28.04.2008 [cit. 2019-03-10]. Dostupné z: http://blog.mediacoderhq.com/h264-profiles-and-levels/

[15] CHYLÍK, Eduard. *CCTV - kamerové systémy.* Falco Computer [online]. 2012 [cit. 2019-03-10]. Dostupné z: http://www.falcocomputer.cz/elektroinstalace/cctv-kamerove-systemy

[16] *IP vs. analog kamery a základní pojmy*. Stasanet.cz [online]. [cit. 2019-03-10]. Dostupné z: https://www.stasanet.cz/IP-vs-analog-kamery-a-zakladni-pojmy/

[17] *Kamerové Systémy CCTV*. Stasanet.cz [online]. [cit. 2019-03-10]. Dostupné z: https://www.stasanet.cz/Kamerove-Systemy-CCTV/

[18] KOSTOLÁNYOVÁ, Kateřina. *Úvod do multimédií: (grafika, hudba a zvuk)*. Vyd. 1. Ostrava: Ostravská univerzita, Pedagogická fakulta, 2003, 54 s. Systém celoživotního vzdělávání Moravskoslezska. ISBN 80-704-2924-0.

[19] KŘEČEK, Stanislav. *Příručka zabezpečovací techniky*. Vyd. 2. S.l. Cricetus, 2003, 351 s. ISBN 80-902-9382-4.

[20] LONGDIN, Anna. *The history of CCTV - from 1942 to present: From monitoring V-2 rockets to digital video recorders*. PCR [online]. 2014, 02.09.2014 [cit. 2019-03-10].

Dostupné z: https://www.pcr-online.biz/retail/the-history-of-cctv-from-1942-to-present

[21] LOUDIL, Jiří. *Metody komprese digitálního videa.* Bakalářská práce, Západočeská univerzivta v Plzni, Plzěň, 2011.

[22] MESNIK, Bob. *The History of Video Surveillance: Evolution from Closed Circuit TV to Ubiquitous IP Camera Surveillance*. KINTRONICS: IP Security Solutions [online]. 2016, 20.06.2016 [cit. 2019-03-10]. Dostupné z: https://kintronics.com/the-history-of-video-surveillance/

[23] *MJPEG vs MPEG4: Understanding the differences, advantages and disadvantages of each compression technique*. On-Net Surveillance Systems Inc [online]. [cit. 2019-03-10]. Dostupné z: https://onssi.com/downloads/OnSSI_WP_compression_techniques.pdf

[24] HOLOTA, Radek. *Přednášky EZO*. Západočeksá univerzita v Plzni, Plzeň, 2018.

[25] KOSTURIK, Kamil. *Přednášky RIS*. Západočeksá univerzita v Plzni, Plzeň, 2018.

[26] RICHARDSON, Iain E. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. 1st edition, 2003. Chichester. 320 s. ISBN 978-0-470-84837-1

[27] RICHARDSON, Iain E. *The H.264 Advanced Video Compression Standard*. London: Wiley-Blackwell, 2010. ISBN 978-0470516928.

[28] ŠEVČÍK, Jiří. *Princip činnosti, typy a komunikační rozhraní IP kamer*. ATP journal [online]. 2012, 06.09.2012 [cit. 2019-03-10]. Dostupné z: https://www.atpjournal.sk/budovy/rubriky/prehladove-clanky/princip-cinnosti-typyakomunikacnirozhrani-ip-kamer.html?page_id=15814

[29] ŠVEJDA, Adam. *V čem se liší IP a AHD kamery*. Secutek [online]. 2018, 02.11.2018 [cit. 2019-03-10]. Dostupné z: https://secutek.cz/blog/53/v-cem-se-lisi-ip-a-ahd-kamery-.html

[30] *VBOX-3620-M12X: Datasheet*. SINTRONES [online]. [cit. 2019-03-10]. Dostupné z: https://www.sintrones.com/VBOX-3620-M12X.html

[31] ČSN EN 50155. *Drážní zařízení – Elektronická zařízení drážních vozidel*. Ed. 2. Praha: Český normalizační institut, 2002

Přílohy

Příloha A – Technické údaje kamery AXIS P3904-R Mk II

www.axis.com T10095723/EN/M4.2/1712

AXIS P3904-R Mk II Network Camera

Models	**AXIS P3904-R Mk II:** RJ45 **AXIS P3904-R Mk II:** M12
Camera	
Image sensor	1/2.9" Progressive scan RGB CMOS
Lens	3.6 mm, F2.0 Horizontal field of view: 87° Vertical field of view: 47° M12 mount, fixed iris See Optional accessories for exchangeable lenses
Light sensitivity	HDTV 720p 25/30 fps with Lightfinder **Color:** 0.15 lux at 50 IRE F2.0 **Color:** 0.06 lux at 30 IRE F2.0
Shutter time	1/32500 s to 2 s
Camera angle adjustment	Pan: ±30° Tilt: 15–90° Rotation: ±175°
Video	
Video compression	H.264 (MPEG-4 Part 10/AVC) Baseline, Main and High Profiles Motion JPEG
Resolution	1280x720 to 160x90
Wide dynamic range	WDR - forensic capture
Frame rate	Up to 25/30 fps (50/60 Hz) in all resolutions
Video streaming	Multiple, individually configurable streams in H.264 and Motion JPEG Axis Zipstream technology in H.264 Controllable frame rate and bandwidth VBR/MBR H.264
Image settings	Compression, color, brightness, sharpness, contrast, white balance, exposure control, exposure zones, fine tuning of behavior at low light, rotation: 0°, 90°, 180°, 270° including Corridor Format, text and image overlay, 20 individual privacy mask, mirroring of images, Traffic Light mode
Pan/Tilt/Zoom	Digital PTZ, preset positions Guard tour, control queue
Network	
Security	Password protection, IP address filtering, HTTPS[a] encryption, network access control, digest authentication, user access log, centralized certificate management
Supported protocols	IPv4/v6, HTTP, HTTPS[a], SSL/TLS[a], QoS Layer 3 DiffServ, FTP, SFTP, CIFS/SMB, SMTP, Bonjour, UPnP™, SNMP v1/v2c/v3 (MIB-II), DNS, DynDNS, NTP, RTSP, RTP, TCP, UDP, IGMP, RTCP, ICMP, DHCP, ARP, SOCKS, SSH
System integration	
Application Programming Interface	Open API for software integration, including VAPIX® and AXIS Camera Application Platform; specifications at *www.axis.com* AXIS Video Hosting System (AVHS) with One-Click Connection ONVIF® Profile S and ONVIF® Profile G, specification at *www.onvif.org*
Analytics	Included AXIS Video Motion Detection, active tampering alarm[b] Supported AXIS Perimeter Defender Support for AXIS Camera Application Platform enabling installation of third-party applications, see *www.axis.com/acap*
Event triggers	Analytics, time scheduled, edge storage events
Event actions	Record video: SD card and network share Upload images or video clips: FTP, SFTP, HTTP, HTTPS, network share and email Pre- and post-alarm video or image buffering for recording or upload Notification: email, HTTP, HTTPS, TCP and SNMP trap PTZ: PTZ preset, start/stop guard tour Overlay text
Data streaming	Event data
Built-in installation aids	Pixel counter
General	
Casing	Aluminum and polycarbonate casing
Sustainability	PVC free
Mounting	Inside vehicles and rolling stock[c]
Memory	512 MB RAM, 256 MB Flash
Power	Power over Ethernet (PoE) IEEE 802.3af/802.3at Type 1 Class 2 Typical 2.9 W, max 3.6 W
Connectors	RJ45: male, 10BASE-T/100BASE-TX **M12:** female, rugged, D-coded with rotatable coupling nut All connectors support PoE
Storage	Support for microSD/microSDHC/microSDXC card Support for SD card encryption Support for recording to network-attached storage (NAS) For SD card and NAS recommendations see *www.axis.com*
Operating conditions	Normal: -40 °C to 60 °C (-22 °F to 140 °F) Maximum (intermittent): 70 °C (158 °F) Arctic Temperature Control: start-up at -40 °C (-40 °F) Humidity: 10-100% RH
Storage conditions	-40 °C to 65 °C (-40 °F to 149 °F)
Approvals	EMC EN 55032 Class A, EN 55024, EN 61000-6-1, EN 61000-6-2, FCC Part 15 Subpart B Class A, ICES-003 Class A, VCCI Class A, RCM AS/NZS CISPR 32 Class A, KCC KN32 Class A, KN35, EN 50121-4, EN 50121-3-2, IEC 62236-4, ECE R10 rev.05 (E approval), EN 50498, EAC Safety IEC/EN/UL 62368-1, EN 45545, UN ECE R118, parts of NFPA 130[d] Environment IEC/EN 61373 Category 1 Class B, IEC/EN 60529 IP66/67, NEMA 250 Type 4X, IEC/EN 62262 IK08, IEC 60721-3-5 Class 5M3 (vibration and shock), EN 50155:2017 Class OT2/ST2 (EN 50155:2007 Class TX), IEC 60068-2-1, IEC 60068-2-2, IEC 60068-2-27, IEC 60068-2-64, IEC 60068-2-78
Dimensions	Height: 49 mm (1 15/16 in), ø 109 mm (4 3/16 in)
Weight	RJ45: 241 g (0.53 lb) **M12:** 248 g (0.55 lb)
Included accessories	Installation guide, Windows decoder 1-user license, drill hole template, top cover tool, lens tool, Allen key Resistorx®
Optional accessories	Lenses 2.1 mm, F2.2: horizontal field of view 147° 2.8 mm, F2.0: horizontal field of view 115° 6 mm, F1.6: horizontal field of view 52° 8 mm, F1.6: horizontal field of view 40° Other AXIS T96B05 Outdoor Housing, AXIS T94D01S Mount Bracket, AXIS T94D02S Mount Bracket, Network coupler IP66, Network cable coupler indoor For more accessories, see *www.axis.com*
Video management software	AXIS Companion, AXIS Camera Station, video management software from Axis' Application Development Partners available on *www.axis.com/support/downloads*
Languages	English, German, French, Spanish, Italian, Russian, Simplified Chinese, Japanese, Korean, Portuguese, Traditional Chinese
Warranty	Axis 3-year warranty and AXIS Extended Warranty option, see *www.axis.com/warranty*

a. *This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (www.openssl.org), and cryptographic software written by Eric Young (eay@cryptsoft.com).*
b. *For detection of tampering attempts in static and non-crowded scenes.*
c. *For outdoor applications, use AXIS T96B05 Outdoor Housing.*
d. *ASTM E162 and the non-flaming parts of ASTM E662*

Environmental responsibility:

www.axis.com/environmental-responsibility

AXIS COMMUNICATIONS

Příloha B – Technické údaje PoE switche

PLANET Networking & Communication

GSW-1600HP

Product Specifications

Model	GSW-1600HP 16-Port 10/100/1000Mbps 802.3at PoE+ Ethernet Switch
Hardware Specification	
Network Port	16 10/100/1000Base-T Auto-MDI/MDI-X RJ-45 ports
PoE Inject Port	16-Port with 802.3af / 802.3at PoE injector function
LED Display	System Power (Green) PoE Ethernet Interfaces PoE In-Use (Orange) 10/100/1000Base-T Ports 1000 (LNK/ACT, Green), 10/100 (LNK/ACT, Orange)
Switch Architecture	Store and Forward switch architecture
MAC Address	8K MAC address table with Auto learning function
Switch Fabric	32Gbps
Switch Throughput	23.8Mpps@64Bytes
Jumbo Packet Size	9K Bytes
Flow Control	Back pressure for Half-Duplex. IEEE 802.3x Pause Frame for Full-Duplex
Power Requirement	AC 100~240V, 50/60Hz, 3.5A max.
Power Consumption	Max. 260 Watts / 885 BTU
Dimension (W x D x H)	445 x 207 x 45 mm (1U height)
Weight	2.54kg
Power over Ethernet	
PoE Standard	IEEE 802.3af Power over Ethernet / PSE IEEE 802.3at Enhancement Power over Ethernet / PSE
PoE Power Output	Per Port 52V DC, 300mA. Max. 15.4 Watts (IEEE 802.3af) Per Port 52V DC, 590mA. Max. 30 Watts (IEEE 802.3at)
PoE Power Budget	220 Watts
Number of PD, 7Watts	16
Number of PD, 15.4Watts	14
Number of PD, 30.8Watts	7
Environment	
Operating environment	0 ~ 50 Degree C
Storage environment	-40 ~ 70 Degree C
Operating Humidity	5 ~ 95%, relative humidity, non-condensing
Storage Humidity	5 ~ 95%, relative humidity, non-condensing
Standard Conformance	
Standard Compliance	IEEE 802.3 Ethernet IEEE 802.3u Fast Ethernet IEEE 802.3x Flow Control IEEE 802.3af Power over Ethernet IEEE 802.3at Enhancement Power over Ethernet IEEE 802.3az Pre-Energy-Efficient Ethernet
Regulation Compliance	FCC Class A, CE

Ordering Information

GSW-1600HP	16-Port 10/100/1000Mbps 802.3at PoE+ Ethernet Switch

Příloha C – Technické údaje záznamové jednotky VBOX-3620-M12X

VBOX-3620-M12X

Intel Gen6 Core i7-6600U CPU with Isolated Input 3 x GbE and 1 x iAMT In-Vehicle Computer

In-Vehicle Computing

Delay Time
Power Ignition
Wide Range
Wireless
Battery Backup

Taiwan Patent No. M447854
Taiwan Patent No. M448011

Features

- Intel Core i7-6600U / i5-6300U / i3-6100U
- Wireless support LTE, 3.5G w/ SIM Card, WLAN GPS Dead Reckoning, GSM/GPRS, BT
- 3 x GbE including 1 with iAMT
- Isolated wide Input Range 9-36VDC
- Smarter Vehicle Power Ignition for Variety Vehicle
- Dual Hot Swappable SATA Storage, RAID 0,1,5
- EN50155 Tx

System

CPU	Intel Gen6 Core I7-6600U 2.6GHz up to 3.4GHz Intel Gen6 Core I5-6300U 2.4GHz up to 3GHz Intel Gen6 Core I3-6100U 2.3GHz Intel Gen 6 Dual Core 3955U 2.0GHz
Memory	2 x SO-DIMM DDR4 up to 32GB
LAN Chipset	2 x Intel I210AT and 1 x Intel I219LM
Audio	Realtek ALC662 HD Codec Onboard
Watchdog	Watchdog Timer Support, Offer 1 – 255 Step

I/O

Serial Port	4 x RS-232/422/485, w/ Isolation (Auto Direction Control)
USB Port	2 x USB 3.0 Ports, 2 x USB 2.0 Ports
LAN	3 x 10/100/1000 Mb/s w/ M12 x-code (1 port with IAMT)
Video Port	2 x DP Port, 1 x VGA (Support Triple Independent Display)
DIO Port	8 x GPI and 4 x GPO w/ Isolation
Audio	1 x Line-out and 1 x Mic-in (Line-in Optional)
SIM Card Socket	2 x SIM Card Sockets, supported onboard with eject 6 x SIM Card Sockets (Optional)
Expansion Bus	3 x Mini-card slots 1 x M.2 slot

Graphics

Graphics	Intel® HD Graphics 520 DirectX Video Acceleration (DXVA) for Accelerating Video Processing - Full AVC/VC1/MPEG2 HW Decode Supports DirectX 11/10.1/10/9 and OpenGL 4.0
Resolution	Up to 4096 x 2304@60 Hz

Storage

Type	2 x 2.5" Drive Bay for SATA Type HDD/SSD, RAID 0, 1, 5 1 x SATA DOM

Power Requirement

Power Input	9V - 36V DC Power Input with Isolation
Power Protection	Automatics Recovery Short Circuit Protection
Power Management	Vehicle Power Ignition for Variety Vehicle
Power Off Control	Power off Delay Time Setting by Software and BIOS
Battery (UPS)*	Internal Battery Kit for 10 Mins Operating (Optional) Patent No. : M447854 - Build-in Battery

* with 6 x SIM card slots option cannot have Battery option

Environmental

Operating Temp.	-40°C ~ 70°C
Storage Temp.	-40°C ~ 80°C
Relative Humidity	0% RH – 95% RH
Vibration(random)	IEC60068-2-64, random, 2.5G@5~500Hz, 1hr/axis with SSD
Vibration Operating	MIL-STD-810G, Method 514.6, Procedure I, Category 4
Shock	Operating: MIL-STD-810G, Method 516.6, Procedure I, Trucks and semi-trailers-15G (11ms) with SSD
Certifications	CE, FCC Class A, EN50155, EN50121

Mechanical

Construction	Aluminum Alloy
Mounting	Supports Both of Wall-mount/VESA-mount
Weight	1900g
Dimensions	250 x 150 x 55 mm

Ordering Information

Part Number	VBOX-3620-M12X-000
Description	Intel Gen6 Core i7-6600U CPU with Isolated Input 3x GbE with 1 x IAMT In-Vehicle Computer