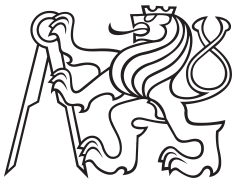
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Pokročilý plánovač pro autonomní ovládání experimentu WILLIAM

Michal Ščupák

Školitel: Ing. Petr Janout, Ph.D.
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ščupák** Jméno: **Michal** Osobní číslo: **460513**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Multimediální technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Pokročilý plánovač pro autonomní ovládání experimentu WILLIAM

Název bakalářské práce anglicky:

Advanced Scheduler for Autonomous Control of WILLIAM Experiment

Pokyny pro vypracování:

Seznamte se s celooblohovým kamerovým systémem WILLIAM pro monitoring povětrnostních podmínek. Seznamte se s metodami plánování úloh, prioritizace a dobou taktu. Na základě poznatků navrhněte a implementujte algoritmus pro plánování definovaných úloh zahrnující zmíněné podmínky. Vytvořte plánovač, který bude schopen autonomně ovládat kamery systému WILLIAM s možností uživatelského vkládání řádných i mimořádných událostí. Navržený plánovač otestujte pomocí systému WILLIAM.

Seznam doporučené literatury:

[1] P. Janout, P. Páta, J. Bednář, E. Anisimova, M. Blažek, and P. Skala, “Stellar objects identification using wide-field camera,” in Photonics, Devices, and Systems VI, 2015, vol. 9450, p. 94501I.
[2] S. G. Kochan, Programming in C, 3rd ed. Indianapolis, Ind: Sams Pub, 2005.
a dle doporučení vedoucího práce.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Janout, Ph.D., katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **01.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Petr Janout, Ph.D.	prof. Mgr. Petr Páta, Ph.D.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Především bych rád poděkoval vedoucímu mé bakalářské práce panu Ing. Petru Janoutovi, Ph.D. za konzultace, podporu a aktivní přístup k vedení práce. Mé poděkovaní také patří panu prof. Mgr. Petru Pátovi, Ph.D. za nabídku zpracování zajímavého tématu práce. Dále chci poděkovat své rodině a blízkým, především za podporu a finální kontrolu formální stránky této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze 20.5.2019

.....

Abstrakt

Práce se zabývá návrhem a implementací rozšíření experimentu WILLIAM o pokročilý plánovač událostí snímání. Systém WILLIAM snímá oblohu a vyžadoval plánovač událostí snímání. Implementovaný plánovač poskytuje uživatelsky přívětivé rozhraní pro správu řádných a mimořádných událostí snímání a detekci kolizí mezi nimi. V teoretické části je tak věnován prostor metodám plánování a prioritizace procesů a následně jsou tyto poznatky využity při samotném vývoji plánovače. Na základě řádných událostí, které plánovač autonomně generuje, a uživatelem vložených mimořádných událostí, plánovač řídí snímání experimentu WILLIAM.

Klíčová slova: C++, PHP, plánování procesů, prioritizace, WILLIAM

Školitel: Ing. Petr Janout, Ph.D.

Abstract

This thesis deals with the design and implementation of an advanced capture tasks scheduler as an extension of the WILLIAM experiment. The WILLIAM experiment captures the sky and required a capture task scheduler. The implemented scheduler provides a user friendly interface for administration of regular/non regular capture tasks and their collision detection. The theoretical part is focused on task scheduling and prioritisation methods. Those findings were later used for the development of the scheduler. Image data acquisition by WILLIAM experiment is based on regular tasks that are autonomously generated by the scheduler and non regular tasks that are defined by user.

Keywords: C++, PHP, task scheduling, prioritisation, WILLIAM

Title translation: Advanced Scheduler for Autonomous Control of WILLIAM Experiment

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Sledování oblohy a astronomie má kořeny v dávné minulosti. S vývojem moderních technologií se dříve velice drahá zařízení, využívaná ke zkoumání oblohy, stávají stále dostupnějšími. Tím také vznikají stále nové inovativní aplikace a možnosti využití těchto zařízení i v každodenním životě.

V případě experimentu Wide-field all-sky image analyzing monitoring system (WILLIAM) je sledování oblohy využito pro detekci vhodných povětrnostních podmínek pro astronomická pozorování a zaznamenaná data jsou také dále využita pro pozdější analýzu. Lze tak poměrně dostupným zařízením, jako je digitální kamera, eliminovat vystavení drahých astronomických přístrojů (jako jsou například kvalitní optické dalekohledy či teleskopy) nepříznivým podmínkám, ve kterých může dojít k jejich poškození. Také lze například lépe optimalizovat pozorovací cyklus teleskopu dané astronomické observatoře, na které je kamera umístěna.

Tato práce se zabývá základním konceptem již fungujícího systému, který se stará o snímání oblohy a rozšířením tohoto zařízení o pokročilý plánovač pro autonomní ovládání experimentu. Experiment WILLIAM je vyvíjen na ČVUT FEL v Praze. Základními funkcemi systému jsou pokročilé techniky snímání a detekce, jako je například rozpoznávání a klasifikace mraků, nebo také statistické určení pravděpodobnosti deště apod. Následně je tak systém schopen detekovat vhodné povětrnostní podmínky v místě instalace zařízení. Experiment WILLIAM je vyvíjen a testován pro pozdější využití spolu se systémem Meteor Automatic Imager and Analyzer stations (MAIA), ze kterého vychází. Dosavadní výstupy experimentu a další informace je možné nalézt na webových stránkách projektu. [1] V současné době však zařízení nemá implementováno jednoduché uživatelsky přívětivé prostředí pro administraci jednotlivých událostí snímání, na základě kterých je zmíněné snímání oblohy, detekce objektů a analýza snímků prováděna. Rozšíření systému o pokročilý plánovač pro autonomní ovládání experimentu má za hlavní úkol vyřešit zadávání řádných (pravidelných) či mimořádných (uživatelských) událostí snímání. Dále je však nutné provést detekci kolizí mezi jednotlivými událostmi a jejich následné vyhodnocení, na základě kterého je možné jednotlivé události snímat. Při návrhu je tedy klíčové stanovit vhodnou metodu plánování a prioritizace těchto událostí. Navíc je také potřeba zajistit co nejvyšší dostupnost dříve zmíněného plánovače a nezávislost jeho jednotlivých částí. Nabízí se

tak využití webové aplikace poskytující uživatelské rozhraní, které pomocí architektury client-server zajistí výše zmíněné funkce i dostupnost a výrazně se tak usnadní celé ovládání experimentů.

Text práce je rozdělen na několik částí, ve kterých je nejdříve blíže popsáno stávající provedení experimentu WILLIAM. Následující kapitola je zaměřena na metody plánování procesů a jejich prioritizaci. Poté jsou shrnuty požadavky na rozšíření experimentu WILLIAM a představen samotný návrh rozšíření systému o pokročilý plánovač. Dále jsou zmíněny použité technologie, vývojové prostředí a podrobný popis implementace plánovače a jeho součástí. Na závěr je stručně věnována pozornost nasazení již navrženého rozšíření plánovače do provozu.

Kapitola 2

Experiment WILLIAM

První generace experimentálního zařízení využívá klasický Digital Single-Lens Reflex (DSLR) fotoaparát s vysokým rozlišením a širokoúhlým objektivem. Novější druhá a třetí generace již obsahuje speciální astronomické snímače Astronomy Imaging camera (ASI), díky kterým lze dosáhnout ještě lepších výsledků detekce objektů na obloze. [4] Dále systém využívá Global Positioning System (GPS) pro určení přesné polohy a synchronizaci času. Celá konstrukce zařízení je pak zapouzdřena do ochranného krytu, který jednak brání jeho poškození, ale také udržuje stabilní prostředí pro snímání. Celé snímací zařízení včetně fotoaparátu a objektivu je možno vidět dále na obrázku 2.0.1, který je dostupný online [8].

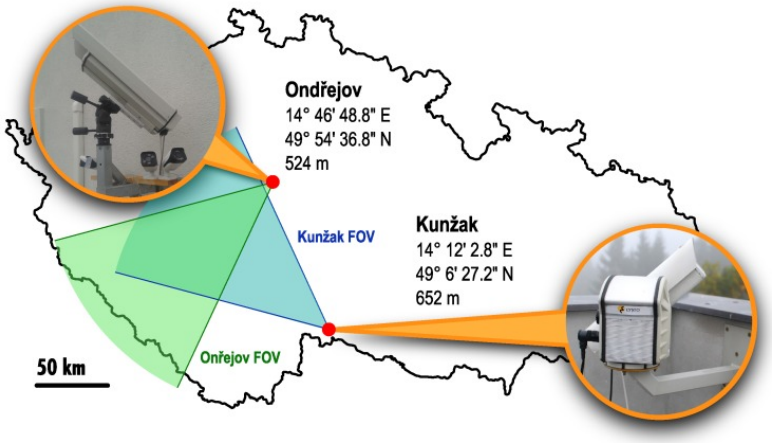
Obrázek 2.0.1: Fotografie zařízení WILLIAM na střeše FEL ČVUT [8]

Princip detekce vhodných podmínek pro noční astronomická pozorování spočívá v analýze snímků, které jsou v určitém časovém intervalu pořizovány. Podmínky pro pozorování jsou určeny na základě porovnání počtu detekovaných hvězd na zaznamenaném snímku s teoretickým počtem viditelných hvězd v daném místě a čase dle katalogu. Systém WILLIAM však také disponuje výčtem dalších funkcí. Mezi ty patří například detekce a klasifikace mraků, kterou se analýza vhodných povětrnostních podmínek může ještě zdo-

konalovat, nebo také statistické metody předpovědi pravděpodobnosti deště apod. V případě detekce nevhodných podmínek pro pozorování oblohy zůstává například kopule astronomické observatoře, kde je zařízení experimentu WILLIAM umístěno, uzavřená a nedochází tak k vystavení drahé pozorovací techniky nepříznivým povětrnostním vlivům. Lze také při částečně vhodných podmínkách systém využít pro vhodné nasměrování teleskopu hvězdárny. [5]

2.1 Projekt MAIA

Systém WILLIAM je experimentální rozšíření projektu Meteor Automatic Imager and Analyzer stations (MAIA). [6] Stejně jako u projektu WILLIAM i projekt MAIA je založen na snímání oblohy pomocí kamerového zařízení, které je ovládáno speciálně navrženým softwarem. MAIA, skládající se ze dvou kamer na dvou vzdálených místech, snímá oblohu s cílem zaznamenat meteory či meteorický roj. Umístění jednotlivých kamer je ilustrováno níže na obrázku 2.1.1, který byl převzat z [2]. Díky dvou kamerám a vysoké rychlosti snímání jednotlivých snímků je systém schopen nejen detekovat letící meteory, ale také analyzovat dráhu meteoru apod. [2]



Obrázek 2.1.1: Umístění zařízení systému MAIA, FOV [2]

2.2 Hardwarové komponenty

Základní komponentou je fotoaparát neboli snímací zařízení, které pořizuje vyhodnocované snímky. První generace systému WILLIAM byla osazena klasickým DSLR fotoaparátem od výrobce Nikon[1]. Původně modelem D5100. Ten byl později však nahrazen modelem D5300, který poskytl vyšší obrazové rozlišení. V obou případech byl na fotoaparát připojen diagonální objektiv typu rybí oko Sigma 10 mm F2,8 EX DC 2, který zajišťuje velice široké zorné pole (zorné pole 180°, pouze diagonálně). Výhoda využití diagonálního objektivu spočívá v lepším využití rozlišení senzoru samotného fotoaparátu.

[1] https://www.nikon.com/

V případě využití cirkulárního objektivu značnou část snímku zaujímají černé okraje (zorné pole 180°, všemi směry). [3]

Později byla vyvinuta druhá a třetí generace zařízení WILLIAM využívající již speciální astronomické kamery čínského výrobce ZWO[2]. První z nich je model ASI 178MC obsahující barevný senzor (využívající Bayerovu masku) typu Complementary Metal–Oxide–Semiconductor (CMOS) s rozlišením 3096 x 2080 pixelů. Druhým modelem je ASI 1600MM-Cool vybavený chlazeným monochromatickým senzorem typu CMOS s rozlišením 4656 x 3520 pixelů. Jeho výhodou je nejen větší rozlišení a velikost senzoru, ale také absence Bayerova filtru. Nedochází tak k interpolaci barev v jednotlivých kanálech (RGB), což přispívá k lepším rozlišovacím schopnostem kamery. Pro potřebu zaznamenat barevné snímky pro detekci mraků je nutné využít filtrů Luminance, Red, Green and Blue (LRGB)[3] a následné kompozice pořízených snímků. [4] Kamery ZWO využívají širokoúhlý objektiv stejného výrobce, a to konkrétně CCTV Lens 2,5 mm 170 Degree, nebo již dříve využívaný objektiv Sigma. [7]

Celé snímací zařízení je zapouzdřeno do voděodolného krytu udržující stabilní podmínky pro snímání a také splňující ochranou funkci před povětrnostními vlivy.

Ovládání experimentu WILLIAM první generace probíhalo prostřednictvím routeru, na němž byl nainstalován speciálně navržený software zajišťující komunikaci mezi kamerou, GPS, serverem a datovým uložištěm. [3] V druhé generaci byl router nahrazen jednočipovým počítačem Rasberry Pi 2 a 3. [7]

2.3 Software a využívané technologie

Hlavní řídící jednotkou je Raspberry Pi[4] se standardním operačním systémem Raspbian jež je odvozen ze základní distribuce Debianu[5] (GNU/Linux). [9] Cron[6], neboli plánovač úloh známý ze systémů této platformy, se stará o spouštění skriptů, které řídí chod fotoaparátu a také komunikaci se vzdáleným serverem. [7]

Operační systém GNU/Linux zajišťuje také chod serveru, ten poskytuje zabezpečenou komunikaci pomocí SSH a SFTP připojení pro bezpečný přenos dat. Pro náročnost detekčních algoritmů se veškeré zpracovávání snímků provádí až na zmíněném serveru. Snímky jsou ukládány v obrazovém formátu RAW, nebo NEF[7] (v případě fotoaparátů Nikon) ve vysoké kvalitě s minimální kompresí, většinou spolu s JPEG[8] náhledy. [7] Při tvorbě nového rozšíření systému WILLIAM o pokročilý plánovač bylo nutné brát ohled na možnosti stávajícího softwaru, kterým je WILLIAM řízen, aby bylo možné navržené

[2] https://astronomy-imaging-camera.com/
[3] http://foto.astronomy.cz/comp_LRGB.htm
[4] https://www.raspberrypi.org/
[5] https://www.debian.org/
[6] https://manpages.debian.org/stretch/cron/cron.8.en.html
[7] https://www.nikonimgsupport.com/eu/BV_article?articleNo=000004883&lang=cs
[8] https://jpeg.org/

skripty a aplikace jednoduše implementovat.

Kapitola 3

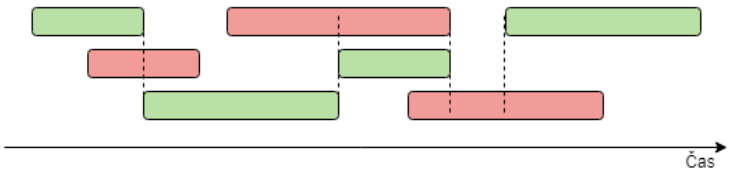
Metody plánování procesů

Plánování procesů je velice komplexní záležitostí a pro stanovení vhodné metody je potřeba analyzovat vstupní data, která potřebujeme uspořádat, naplánovat, přiřadit jednotlivým procesům určité priority (jeli to nutné), a také stanovit, jaký má být výstup plánovacího procesu. Snímání události (pořízení snímku oblohy) můžeme také nazvat procesem, který systém WILLIAM v reálném čase provádí a správná funkce systému tak závisí na provedení daného procesu v odpovídající čas události. Samotné snímání oblohy je tedy z pohledu klasifikace systémů na základě provádění procesů systém reálného času (Real-time system). V případě systémů reálného času závisí správná funkce systému nejen na proměnných definující jednotlivé procesy a jejich provedení, ale především na čase, kdy má být proces prováděn. Lze také říci, že událost (pořízení snímku oblohy) je takzvaný hard real-time proces. Tato klasifikace procesu znamená, že provedení události (procesu) v jiný, než definovaný čas nesmí nastat, jinak dojde k nesprávné funkci systému. [11]

Když se zaměříme pouze na plánování událostí (uživatelské rozhraní) a tedy přidávání nových mimořádných událostí a případně generování řádných událostí, což je hlavním cílem této práce, je nutné se zaměřit především na prioritizaci událostí (procesů), detekci vzájemných kolizí a jejich následné vyhodnocení. Nutnost stanovení priorit mezi řádnými a mimořádnými událostmi vyplývá již z požadavků experimentu WILLIAM. Abychom zvolili vhodnou metodu (algoritmus) pro plánování jednotlivých úloh snímání je potřeba provést analýzu vstupu a výstupu plánovače, jak již bylo zmíněno na začátku této kapitoly. Existuje velké množství třídících a plánovacích algoritmů, ne všechny jsou však vhodné pro aplikaci v našem případě. Některé metody plánování procesů využívají také strojového učení a umělé inteligence pro ještě vyšší optimalizaci výsledku. Tyto metody se však spíše využívají při aplikacích jako je řízení průmyslové výroby nebo například procesní řízení podniků a pro současné řešení plánování úloh experimentu WILLIAM nemají smysl. Pozornost byla tedy upřena především na koncept datových struktur jako je prioritní fronta (Priority queue) a fronta obecně. Dále se autor zaměřil na optimalizační algoritmy, a to konkrétně na hladový algoritmus (Greedy algorithm), který funguje na principu hledání optimální množiny splňující určitou vlastnost a jeho modifikace jsou hodně využívány v plánovacích algoritmech. [12]

First-in, first-out (FIFO) je velice jednoduchý koncept, který je využíván frontou. Fronta je datová struktura obsahující objekty, v našem případě procesy, které jsou uspořádány v pořadí, v jakém jsou do fronty vloženy. Prioritní fronta je speciálním případem fronty jejíž objekty navíc obsahují přidělenou prioritu. Objekty (procesy) jsou následně vykonávány nejen na základě jejich pořadí, ale také jsou porovnávány jejich priority. Pokud některý z objektů (procesů) má vyšší prioritu než prvky před ním, přeskočí je. [12]

Optimalizací myslíme nalezení nejvhodnějšího řešení pro námi definovaný problém. Hladový algoritmus nelze jednoznačně definovat, existuje totiž množství modifikací, které jsou využívány pro různé aplikace. Ve stručnosti jde o nalezení optimálního řešení v jednotlivých krocích podle rozhodnutí na základě specifikované podmínky. Jednou z modifikací je hladový algoritmus, který nalezne *i-tý* proces jehož konec nastane nejdříve (aby další proces mohl být co nejdříve prováděn) a přidá jej do seznamu naplánovaných procesů. Poté jsou všechny konfliktní procesy s *i-tým* procesem přeskočeny a celá smyčka se opakuje. Ilustrace algoritmu je znázorněna níže na obrázku 3.0.1.



Obrázek 3.0.1: Ilustrace funkce výše popsané varianty hladového algoritmu (zeleně je vyznačen výsledek algoritmu, červeně pak procesy, které jsou algoritmem vynechány). [21]

Další jeho modifikace je například založena na úlohách, které nemají pevně stanovený čas a lze tak s nimi v rámci definované meze (deadline) posunovat. V průběhu algoritmu se pak snažíme najít řešení s minimálním časovým posunem jednotlivých procesů. Podobně jako v předešlém příkladu v průběhu algoritmu hledáme proces, jehož deadline nastane nejdříve, toto pravidlo je nazýváno Earliest Deadline First. [12]

Existují i jiné algoritmy a jejich modifikace, které zde dále nepopisuji z důvodu jejich nevhodnosti pro námi požadovanou aplikaci a případně je lze nalézt v [12].

Díky tomu, že je nutné pro experiment WILLIAM zajistit, aby při zadávání nových událostí snímání nedocházelo k nežádoucím změnám již zadaných událostí, nelze přímo využít žádnou z výše popsaných metod, neboť nejsou vhodné. Pro požadované řešení plánování a prioritizaci řádných a mimořádných událostí snímání byla navržena metoda plánování, jenž je kombinací výše zmíněných metod plánování. Ta je blíže popsaná v následujících kapitolách.

Kapitola 4
Návrh plánovače

Současný stav experimentu WILLIAM neposkytoval uživatelsky přívětivou metodu, jak systém jednoduše ovládat, měnit parametry snímání nebo případně také přidávat snímání mimořádné události. S postupným vývojem systému tak vznikla potřeba vyvinout aplikaci poskytující uživatelské rozhraní, které vyhoví těmto požadavkům.

4.1 Funkční požadavky na plánovač

Při návrhu plánovače bylo nutné plně zachovat funkci stávajícího zařízení. GNU/Linux[1] je platforma, která poskytuje základ řídícího systému umožňuje využití velkého množství programovacích jazyků pro tvorbu rozšíření systému WILLIAM, které lze jednoduše implementovat. [10]

Důležitým aspektem je oddělit uživatele (uživatelské rozhraní) od autonomního řízení snímání systému WILLIAM z důvodu zajištění nezávislosti jednotlivých procesů a zajištění bezpečnosti. Je tedy nutné zajistit přenos dat mezi uživatelem a samotným kamerovým systémem. Dále je nezbytné zajistit bezpečný přístup do aplikace s autorizací uživatelů, kteří mohou naplánované události upravovat či přidávat nové. Stávající stav zařízení poskytoval šest přednastavení jednotlivých snímání, a to tři přednastavení pro noční režim a stejný počet pro denní režim. V případě rozšíření o pokročilý plánovač tomu je jinak a je nutné brát v úvahu teoreticky nekonečné množství kombinací různých nastavení událostí snímání. Plánovač musí být také schopen autonomně generovat řádné události, nezávisle na uživateli. K tomu je potřeba přednastavení řádných událostí pro denní a noční snímání. S tím souvisí nezbytné určení času západu a východu slunce pro danou lokalitu, kde se zařízení nachází.

Na základě výše zmíněných požadavků vzniká potřeba vytvořit seznam naplánovaných událostí spolu s jejich parametry a časem. Při změně parametrů řádných snímání nebo vkládání snímání nových mimořádných událostí může dojít ke kolizi jednotlivých časů, kdy se daná snímání mají provádět. Je potřeba tedy zajistit jednak detekci těchto kolizí, ale také jejich vyhodnocení, jako je například uzpůsobení času kolizních událostí. Pro vyhodnocení kolizí je

[1] https://www.gnu.org/

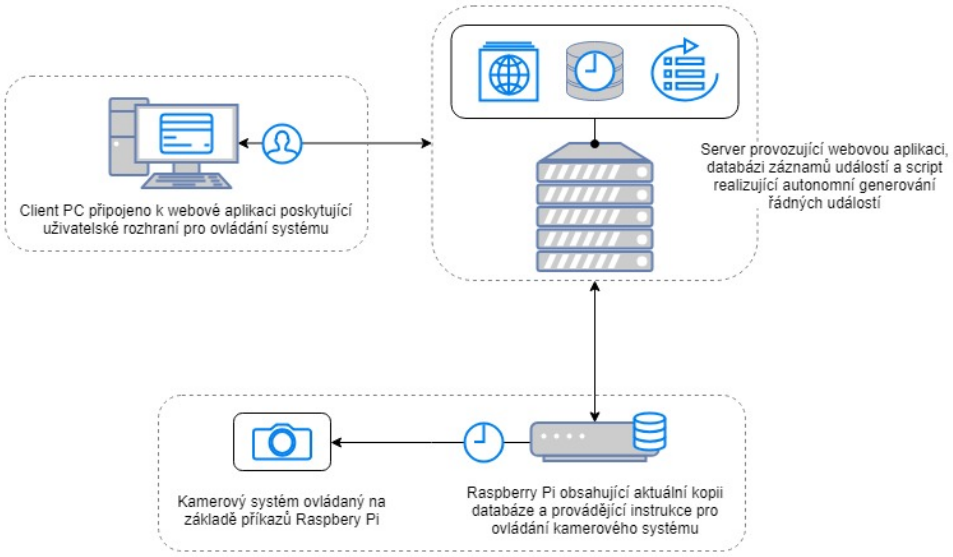
nutné přiřadit jednotlivým událostem snímání priority. Pro určení a přiřazení priorit existuje velké množství metod a modelů. V našem případě není potřeba využívat žádné komplexní modely, vystačíme si s rozdělením priorit pro řádné a mimořádné události. Přičemž řádné události jsou podřadné mimořádným událostem, jež jsou definovány uživatelem, což vyplývá z požadavků experimentu WILLIAM. Dalším požadavkem je, aby měl uživatel možnost kdykoliv některou z událostí odstranit. Všechny dříve zmíněné operace vyžadují změnu v záznamech naplánovaných událostí. Je tedy nutné (pro přehled nad provedenými událostmi) uchovávat záznam o tom, jaké události byly skutečně v daném čase provedeny.

4.2 Návrh komponent plánovače

Samotný návrh komponent plánovače musel navázat na dříve popsané funkční požadavky. Při jeho realizaci se nabízely různé možnosti, jak dospět k požadovanému řešení, ze kterých byl nakonec vytvořen následující návrh. Již při prvním zamyšlení nad návrhem autonomního plánovače bylo zřejmé, že je logické jej rozdělit na několik nezávislých částí, které se nebudou vzájemně ovlivňovat. Generování řádných (pravidelných) událostí snímání na určitou dobu dopředu a samotný proces provádění snímků kamerovým zařízením nemají spolu z pohledu funkce moc společného. Stejně tak přidávání nových mimořádných událostí snímání či změna parametrů snímání nemá téměř žádnou společnou funkcionalitu s dříve zmíněnými procesy.

Celý autonomní plánovač je tak rozdělen na dvě, respektive tři, autonomní části, uprostřed nichž stojí databáze (jde tedy o tzv. centralizovanou architekturu[2]). Plánovací systém je takto navržen z důvodu zajištění nezávislosti jednotlivých částí, které používají jako prostředníka databázi, a tudíž se navzájem nijak neovlivňují. Přičemž hlavní část plánování úloh, detekce kolizí a jejich vyhodnocování, se provádí pomocí serverově orientované webové aplikace. Softwarová architektura client-server[2], je jednou z nejpoužívanějších v případě distribuovaných systémů. Je založena na centralizovaném výpočetním zařízení označovaném server, které zpracovává požadavky (operace) druhého zařízení označovaného jako client a navrací pouze výsledky. O veškeré výpočty a práci s daty se tak stará server a client zařízení, kterých může být teoreticky neomezený počet, slouží pouze jako zprostředkovatel informace. Vzhledem k tomu, že je požadována maximální dostupnost možnosti upravovat a přidávat snímání pomocí navrženého autonomního plánovače, je pro nás jedna z modifikací architektury client-server za použití webové aplikace nejvhodnější. Při využití architektury samostatné aplikace (Standalone application), jejíž instalace by byla nutná na každém uživatelském zařízení, by se řešení značně zkomplikovalo (i vzhledem k práci s daty).[13] Jednotlivé komponenty jsou blíže popsány dále v textu spolu se schématickým znázorněním plánovacího systému, které je možno vidět na obrázku 4.2.1.

[2] https://www.tutorialspoint.com/software_architecture_design/



Obrázek 4.2.1: Základní systémové schéma plánovače [21]

Navržená centralizovaná databáze obsahuje pro dané sestavení dvě oddělené tabulky záznamů. První tabulka obsahuje pouze základní nastavení řádných (pravidelných) událostí spolu s časovým intervalem v jakém se má daná událost provádět. Ve druhé databázové tabulce naplánovaných událostí jsou již zařazeny záznamy o událostech, které mají být nebo již byly provedeny. Tyto záznamy již obsahují přidělený čas, kdy se má daná událost provést (neobsahující kolize) spolu s nastavením potřebných parametrů události a jsou využity pro výsledné snímání. První databázová tabulka nastavení řádných událostí je oddělena od tabulky naplánovaných událostí z důvodu zajištění jednoduché administrace základního nastavení řádných událostí a nijak se nepodílí na provádění událostí snímání. Přičemž rozlišujeme nastavení pro typ události nočního a denního snímání.

O načtení základního nastavení řádných událostí z databáze, přiřazení času provedení jednotlivých řádných událostí v daném intervalu a jejich následné zařazení do tabulky naplánovaných událostí, se stará autonomní generátor řádných událostí. Ten je umístěn odděleně na serveru a pomocí systémového plánovače pravidelně spouštěn, aby generoval řádné události na nadcházející dny. Počet dnů, na které generátor plánuje dopředu, je dle návrhu stanoven na sedm dní dopředu.

Další autonomní částí celého systému je webová aplikace poskytující uživatelské rozhraní, jehož prostřednictvím správce systému (uživatel) může experiment WILLIAM ovládat. Jednak lze měnit základní nastavení řádných událostí včetně intervalu jejich provádění, ale také přidávat další uživatelem definované události. Změna parametrů řádných událostí pak má za následek nutnost vyvolání požadavku na vygenerování pozměněných řádných událostí snímání, vypořádání se vzniklými kolizemi a vložení nového plánu (seznamu prováděných událostí) do databáze. Při přidání mimořádné události je nutné nalézt v případě kolize vhodné řešení. Dle návrhu a požadavků na plánovač

mají mimořádné události snímání vyšší prioritu než řádné. V případě kolize dvou mimořádných událostí pak plánovač upřednostní dříve vložené snímání. Objekty s nižší prioritou jsou pak zařazeny na nejbližší místo (posunuty vpřed v čase). Dojde-li tímto k posunu řádné události (událost A) tak, že nastane kolize s následující řádnou událostí (událost B), je řádná událost A odstraněna.

Poslední částí plánovače je skript, který již setříděné a naplánované události načítá z již dříve zmíněné databáze a v čase provedení dané události vyšle příkaz o provedení do kamerového systému. Prováděné události jsou uloženy v lokální kopii, je proto nutné zajisti periodickou kontrolu aktuálnosti těchto dat porovnáním s databází.

Kapitola 5

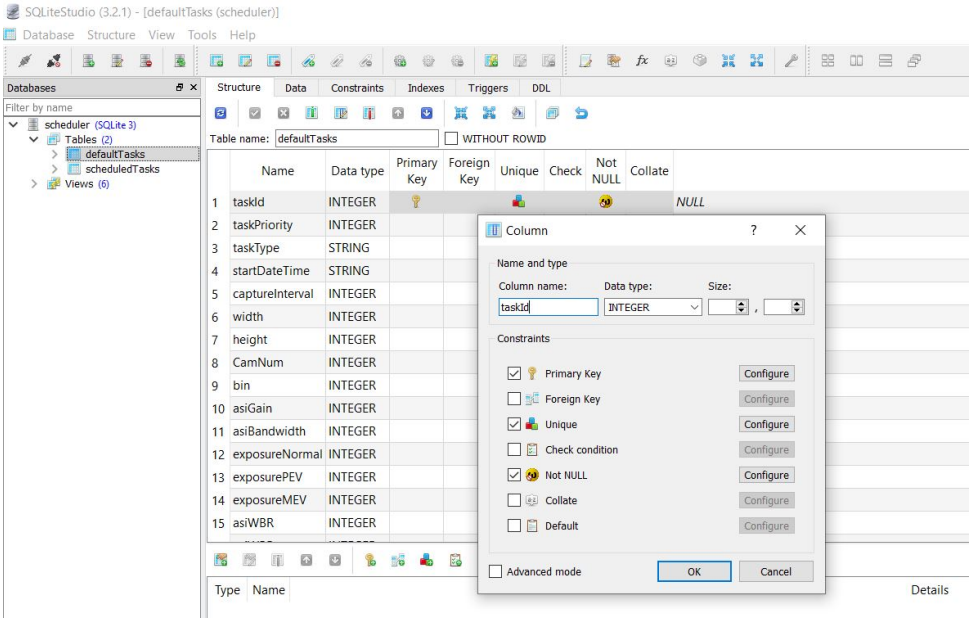
Využité technologie a vývojové prostředí

Pokročilý plánovač pro autonomní ovládání experimentu WILLIAM byl vyvíjen na zařízení s operačním systémem Microsoft Windows 10 Pro[1]. Bylo tedy nutné zaručit co nejjednodušší nasazení plánovacího systému do provozu a také jeho funkčnost na cílovém zařízení s operačním systémem platformy GNU/Linux a softwarovým webovým serverem Apache[2].

5.1 SQLite databáze, SQLiteStudio

Jak bylo v předchozí kapitole zmíněno, jednou z hlavních komponent plánovače je databáze, která je prostředníkem jednotlivých komponent zajišťující jednotlivé funkce plánovače. Pro tvorbu databáze byl zvolen Structured Query Language (SQL)[3] a konkrétně jednoduchý relační databázový systém SQLite[4]. Pro jeho funkci není potřeba databázový server, na kterém by byla databáze nainstalována, a proto si vystačíme s jedním databázovým souborem reprezentujícím samotnou databázi. SQLite databáze je také systémově nezávislá a tak i jednoduše přenositelná, což velice usnadňuje její implementaci.[14] K funkci databáze a také následnému využití spolu s dalšími komponenty pokročilého plánovače je nutné nejdříve provést jednoduchou instalaci balíčku SQLite, který je možno nalézt na internetu spolu s instrukcemi k jeho instalaci.[15] Vytvoření databáze, její administrace a testování bylo prováděno pomocí volně dostupného databázového manažeru SQLiteStudio[5], jež poskytuje velice přehledné prostředí a správu SQLite databází.

[1] https://www.microsoft.com/en-us/windows
[2] https://httpd.apache.org/
[3] https://www.w3schools.com/sql/sql_intro.asp
[4] https://www.sqlite.org/
[5] https://sqlitestudio.pl/



Obrázek 5.1.1: Snímek obrazovky SQLiteStudio manažeru, tabulka defaultTasks

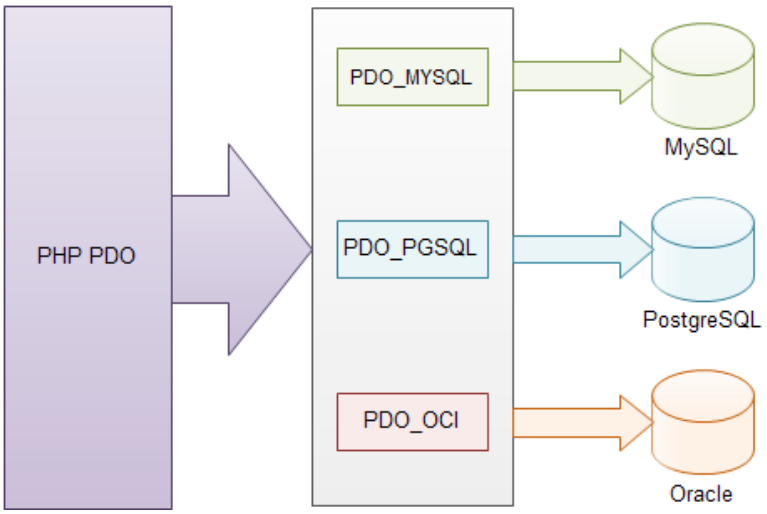
5.2 C++, Windows Subsystem for Linux

Další z komponent plánovače je takzvaný generátor řádných událostí, ten využívá základní nastavení pro noční a denní událost snímání, které je uloženo ve dříve zmíněné databázi. Následně tak s daným nastavením autonomně generuje události pro den a noc v závislosti na východu a západu slunce a vygenerované záznamy pak ukládá do databáze. Pro výpočet času východu a západu slunce byl využit skript SunSet[6], dostupný pod licencí GPL2. Ten pomocí souřadnic místa, kde se zařízení nachází a nastavení lokálního času určí dobu soumraku.[16] Implementace generátoru je uskutečněna pomocí multiparadigmatického programovacího jazyka C++[7]. Stejně tak samotné autonomní pořizování jednotlivých snímků (provádění jednotlivých událostí) na základě předem vytvořených záznamů v databázi je implementováno pomocí programovacího jazyka C++. Oba skripty byly vyvíjeny a kompilovány na platformě Windows v prostředí Windows Subsystem for Linux (WSL)[8] a to z důvodu zajištění funkce aplikace na cílovém zařízení (server s operačním systémem GNU/Linux). WSL je nadstavbou operačního systému Microsoft Windows, jež poskytuje vývojové prostředí systémů GNU/Linux spolu s příkazovou řádkou, aplikacemi apod., bez nutnosti virtualizace zařízení. Toto prostředí rovněž umožňuje doinstalování hojně využívaného GCC kompilátoru[9], kterým byla prováděna kompilace C++ skriptů. [17]

[6] https://github.com/buelowp/sunset
[7] https://en.cppreference.com/
[8] https://docs.microsoft.com/en-us/windows/wsl/about
[9] https://gcc.gnu.org/

5.3 PHP, PDO, IIS Express

Uživatelské prostředí pro ovládání pokročilého plánovače snímání zajišťuje zmíněná webová aplikace. Experiment tak lze ovládat vzdáleně z jakéhokoli místa s internetovým připojením a není tedy nutná přítomnost uživatele v místě snímání. Přístup do aplikace je nutné zabezpečit autorizací uživatelů pro zajištění bezpečnosti a udělení přístupu k samotnému plánovači. Díky umístění webové aplikace na již existující doméně experimentu WILLIAM, kde je provozován redakční systém WordPress[10] poskytující autorizaci uživatelů, není nutné již tuto autorizaci provádět v samotné aplikaci plánovače. Pro realizaci webové aplikace byl využit skriptovací programovací jazyk Hypertext Preprocessor (PHP)[11] spolu s Hypertext Markup Language (HTML), JavaScriptem a případně dalšími styly. Webová aplikace zaručuje potřebnou interaktivitu prostředí s uživatelem a spojení s databází, které je nutné pro vkládání a úpravu událostí snímání. [18] Při vývoji, testování a emulaci webového prostředí byl využit softwarový webový server IIS Express[12] (Internet Information Services 10.0 Express) jež je součástí operačního systému Microsoft Windows (případně lze jednoduše doinstalovat) a je určený pro vývoj webových aplikací na platformě Windows. Podrobnější popis instalace a následné konfigurace IIS Express a PHP lze nalézt v dokumentaci, viz [19]. Propojení PHP a databáze je zajištěno pomocí rozhraní PHP Data Objects (PDO)[13], to umožňuje připojení několika různých databázových systémů za použití specifického ovladače použité databáze. Lze tedy v případě potřeby poměrně jednoduše provést záměnu databázového systému bez nutnosti rozsáhlé úpravy původních funkcí pracujících s databází. [20]



Obrázek 5.3.1: Schéma PHP PDO rozhraní [20]

[10] https://wordpress.com/
[11] https://www.php.net/
[12] https://docs.microsoft.com/en-us/iis/extensions/introduction-to-iis-express/
[13] https://www.php.net/manual/en/book.pdo.php

Kapitola 6

Implementace komponent plánovače

Implementace pokročilého plánovače pro autonomní ovládání experimentu WILLIAM probíhala částečně v emulovaném vývojovém prostředí WSL (v případě programování skriptů v C++) a částečně pomocí IIS Express (v případě PHP skriptů). V následujících podkapitolách je stručně popsána funkce jednotlivých komponent a jejich implementace.

6.1 Implementace databáze

Jak bylo dříve uvedeno, SQLiteStudio poskytuje velice jednoduchou a efektivní správu SQLite databází. Vytvořená databáze obsahuje dvě oddělené tabulky. První z nich je tabulka s názvem defaultTasks. Ta obsahuje dva záznamy nastavení řádných událostí snímání, a to konkrétně denní a noční nastavení (odlišeny typem defaultDay, defaultNight a obsahující interval provedení captureInterval). Tuto tabulku především využívá skript nazvaný generator pro autonomní generování řádných snímání (tomu je věnována pozornost v následující podkapitole). V budoucnu je možno zařízení teoreticky jednoduše rozšířit o větší množství základních přednastavení řádných událostí. Druhou z tabulek je tabulka nazvaná scheduledTasks, která již obsahuje reálné záznamy o událostech snímání s přiděleným časem (startDateTime), které byly buď vygenerovány autonomním generátorem řádných událostí nebo byly přidány uživatelem spolu s časovým vyjádřením trvání události v milisekundách (fullTaskDuration). Jedná se tedy o databázovou tabulku, která již nerozlišuje mimořádné a řádné události snímání (informace o prioritě a typu události defaultDay, defaultNight a userTask zůstává v tabulce zachována) a zároveň již nesmí obsahovat kolize událostí. Na základě záznamů z této tabulky je následně prováděno reálné snímání kamerovým systémem. Níže lze nahlédnout do ukázky struktury databázových tabulek vytvořených pomocí softwaru SQLiteStudio, kde lze také vidět všechny zmíněné a další potřebné parametry událostí (spolu s výchozími hodnotami), viz tabulky 6.1 a 6.2.

Parametr	Denní nastavení	Noční nastavení
taskId	1	2
taskPriority	1	1
taskType	defaultDay	defaultNight
startDateTime	NULL	NULL
captureInterval	600 (s)	600 (s)
width	1	1
height	1	1
CamNum	NULL	NULL
bin	1	1
asiGain	0	150
asiBandwidth	40	40
exposureNormal	100	22000000
exposurePEV	0	0
exposureMEV	0	0
asiWBR	75	75
asiWBB	80	80
asiGamma	50	50
asiBrightness	40	10
asiFlip	0	0
collingOn	1	1
capturingOn	1	1
setTemp	18	18
printİnfo	0	0

Tabulka 6.1: Výchozí hodnoty nastavení řádných událostí (defaultTask)

Parametr	Příklad nastavení
taskId	123
parentTaskId	0
taskPriority	1
taskType	defaultDay
startDateTime	2019-05-15 12:50:35
fullTaskDuration	1100
...	...

Tabulka 6.2: Příklad záznamu naplánované události (scheduledTask, neobsahuje všechny parametry)

6.2 Impementace generátoru řádných událostí

Autonomní generování řádných událostí je zajištěno skriptem generator.cpp. Ten ke své funkci využívá rozhraní sqlite3[1] pro připojení a práci s SQLite databází. Dále byl vytvořen předpis pro vznik objektu reprezentující událost

[1] https://sqlite.org/c3ref/intro.html

snímání, třídu task (soubory task.cpp, task.h), jež obsahuje atributy představující parametry události a funkce pro práci s nimi. Podrobněji jsou jednotlivé části a jejich funkce popsány v následujícím textu.

6.2.1 Třída task

Hlavní funkcí třídy task je reprezentace objektu události, v tomto případě řádné události snímání. Atributy objektu třídy task jsou téměř totožné s parametry obsaženými ve dříve zmíněných databázových tabulkách. Protože databázové tabulky defaultTasks a scheduledTasks se v některých parametrech liší, obsahuje třída task ke své plnohodnotné funkci atributy, které jsou kombinací parametrů obou databázových tabulek.

Dále třída obsahuje metody pro základní přístup k atributům objektu, takzvané getters. Jde o metody navracející hodnotu požadovaného atributu objektu. Podstatnou metodou třídy task pro generátor řádných událostí je především metoda `void setStartDateTime(time_t)` umožňující nastavení času konkrétní události před zařazením do naplánovaných událostí snímání do databáze. Další neméně důležitou metodou je `time_t getTime_tStartDateTime()`, která je však především potřebná ve skriptu run.cpp, kde je využívána pro přístup k hodnotě startDateTime objektu třídy task, potřebné k provedení dané události v požadovaný čas. Podrobněji je její využití popsáno v následující podkapitole věnující se real-time provádění událostí.

6.2.2 Skript generující řádné události

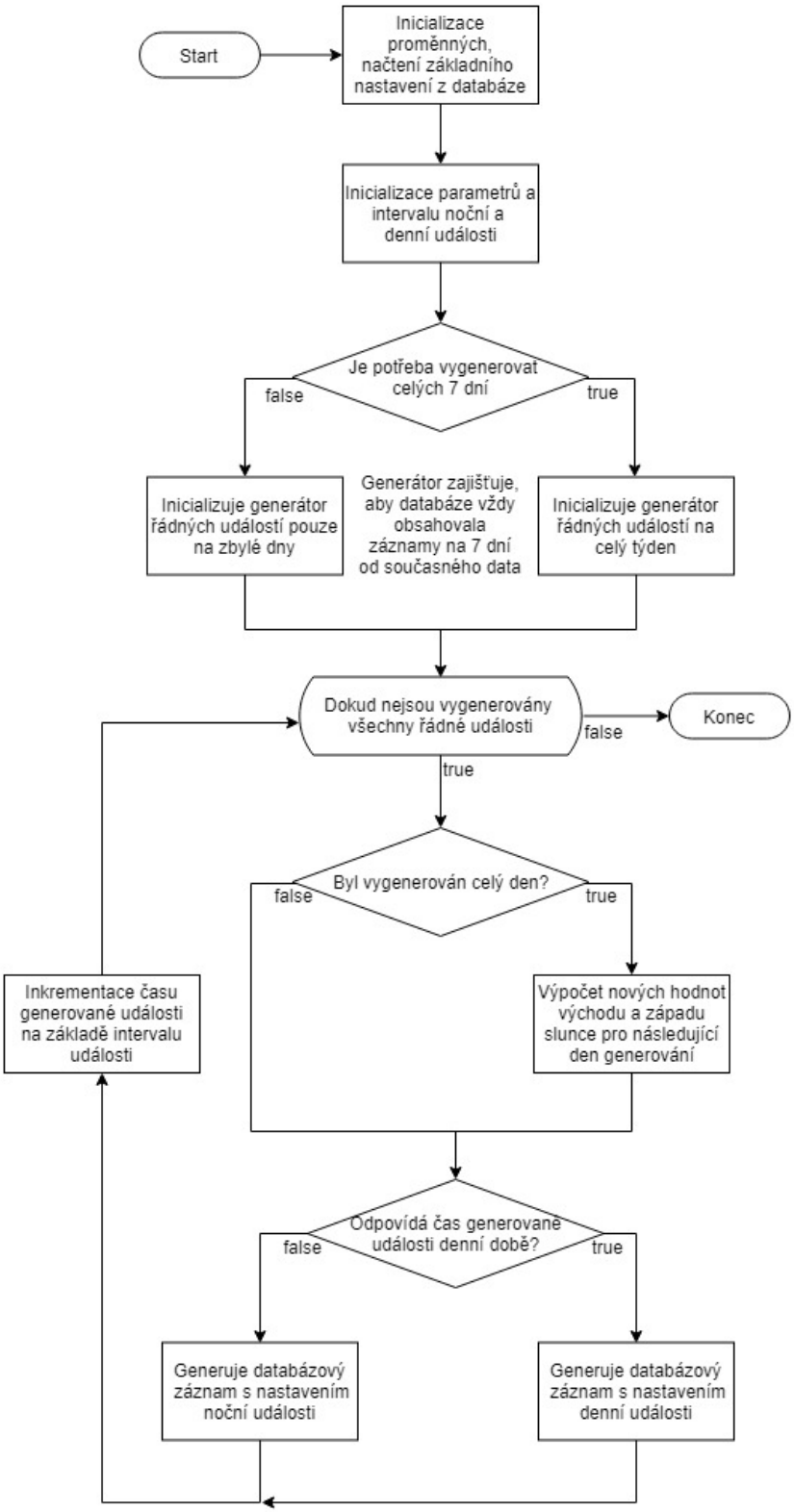
Skript generator.cpp využívá několik periferií pro svou funkci. Základem je připojení k SQLite databázi pomocí dříve zmíněného rozhraní a zavolání funkce `void generateDefaultSchedule(sqlite3*, SunSet)`, jejíž parametry jsou pointer (ukazatel) na instanci databázového spojení a instance třídy SunSet.

Pomocí převzatého skriptu SunSet je tedy nutné vytvořit ještě instanci třídy SunSet, která slouží pro určení času východu a západu slunce. Je potřeba poskytnout souřadnice zařízení spolu s lokálním časem a následně se pomocí funkce `void sunsetSetup()` a `double setCurrentDate(int, int, int)` inicializuje její instance. Hodnoty východu a západu slunce jsou pak vypočítány prostřednictvím metod `double calcSunrise()` a `double calcSunset()` a využity pro vymezení denní a noční doby a následné generování vhodného nastavení řádných událostí snímání v danou dobu. [16]

Funkce `void generateDefaultSchedule(sqlite3*, SunSet)` nejdříve inicializuje současný čas a následně načte základní nastavení snímání defaultDay a defaultNight z databáze. K tomu slouží funkce `task getDayTaskSettings(sqlite3*)` a `task getNightTaskSettings(sqlite3*)`, které pomocí metody `sqlite3_exec(...)`, blíže popsané v dokumentaci rozhraní sqlite3[2] , vytvoří instanci třídy task odpovídající parametrům obdrženým z databáze. Tyto dvě instance jsou dále použité ke generaci řádných událostí

[2] http://sqlite.org/c3ref/exec.html

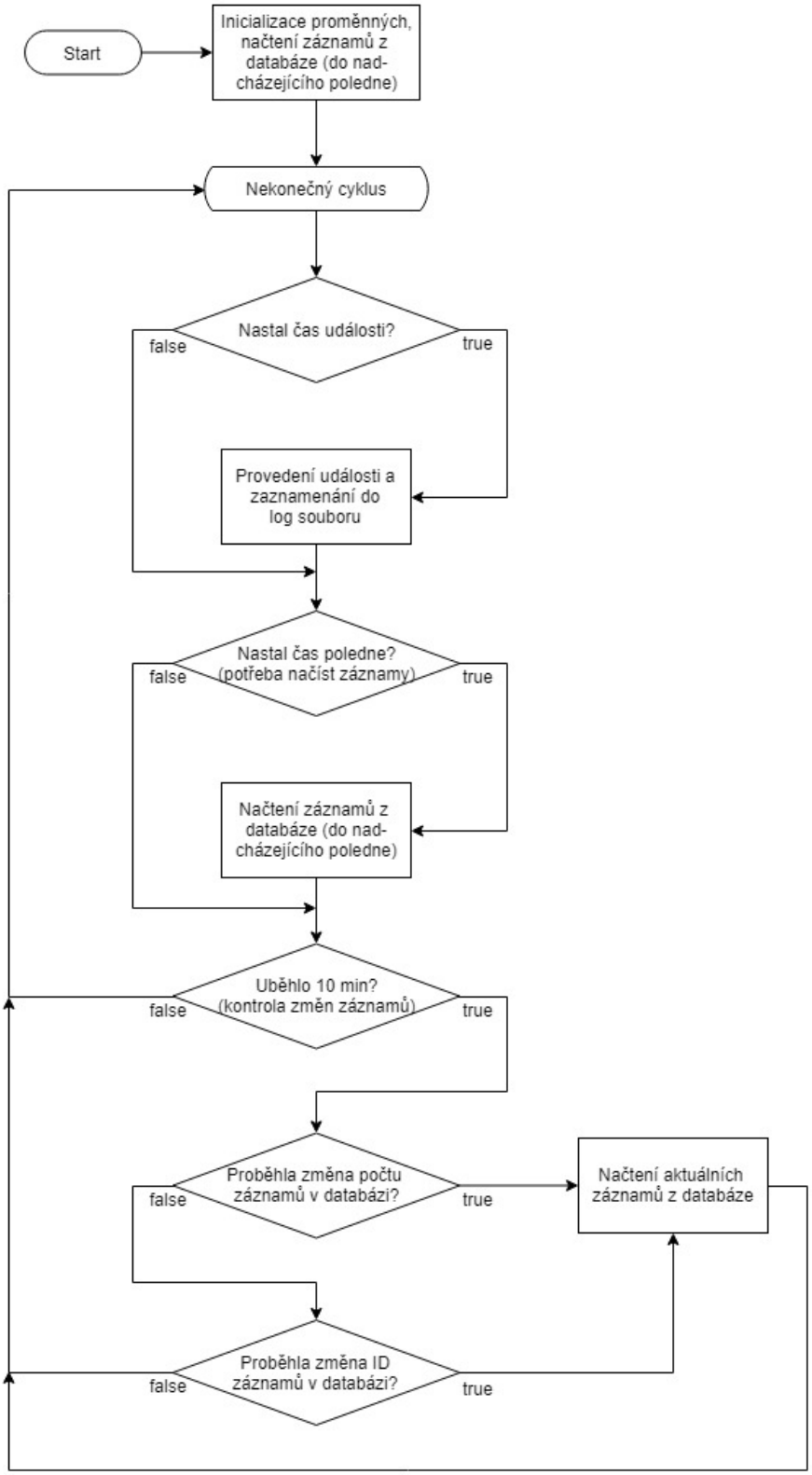
snímání. Algoritmus generování událostí zajišťuje vždy existenci vygenerovaných řádných událostí na následujících sedm dní (uživatel smí zadávat mimořádné události taktéž na sedm dní dopředu). Protože je generátor dle návrhu spouštěn každý den systémovým plánovačem procesů Cron, není nutné, aby generoval pokaždé všech sedm dnů. Zároveň však v případě změny parametrů řádných událostí může nastat potřeba přegenerovat celých sedm dnů. Tato změna je vyvolána webovou aplikací, při změně nastavení řádných událostí snímání v uživatelském prostředí. Proto ještě před samotným generováním řádných událostí, dojde ke kontrole poslední řádné události (přesněji jejího času provedení), která se nachází v databázi, prostřednictvím funkce `time_t getLastDefaultTaskTime(sqlite3*)`. Tato funkce na základě předání pointeru na instanci spojení s SQLite databází vrátí hodnotu času poslední naplánované řádné události. Na základě tohoto času je pak rozhodnuto kolik dní je potřeba vygenerovat pro splnění dříve zmíněné podmínky sedmi dní. Následně je již nastaven počáteční a koncový čas a datum generování událostí. Postupně je tak přidělován čas (startDateTime) jednotlivým událostem pomocí metody třídy task `void setStartDateTime(time_t)` na základě jejich definovaného intervalu (captureInterval) a na základě denní nebo noční doby, pro kterou je daná událost generována (defaultDay nebo defaultNight). Tyto události jsou jednotlivě přidávány do databáze pomocí funkce `void addTaskToScheduled(sqlite3*, task)`, jejíž parametry jsou pointer na instanci spojení s SQLite databází a instance třídy task reprezentující vygenerovanou událost. Aplikuje se opět funkce `sqlite3_exec(...)` poskytovaná SQLite rozhraním pro vložení záznamu do databáze. Zjednodušený diagram funkce znázorňuje obrázek 6.2.1.



Obrázek 6.2.1: Flowchart diagram skriptu generate.cpp [21]

6.3 Implementace real-time provádění událostí

Provádění událostí (inicializaci komunikace s kamerovým systémem pro reálné pořízení snímků) zajišťuje skript run.cpp. Při spuštění skriptu se inicializuje současný čas a čas poledne následujícího dne. Současný čas je důležitý pro provádění událostí snímání v definovaný čas. Čas poledne byl zvolen jako čas načítání záznamů na následujících 24 hodin. Důvodem pro zvolení tohoto času načítání záznamů z databáze je teoreticky menší vytížení systému snímání, a proto je vhodné provádět aktualizaci databázových záznamů na další den v tuto dobu. Zároveň dojde k načtení naplánovaných událostí (načtení části databázové tabulky scheduledTasks a vytvoření dočasné lokální kopie záznamů) z SQLite databáze podobně jako tomu bylo v předešlých případech. O načtení záznamů se stará funkce `list<task> getDayScheduledTasksList(time_t, time_t)` jejíž první a druhý parametr vymezuje počáteční a koncový čas intervalu načtených záznamů naplánovaných událostí. Její návratovou hodnotou je list objektů třídy task (reprezentující list jednotlivých naplánovaných událostí snímání). Následně dojde k inicializaci záznamového (log) souboru a skript vstupuje do nekonečné smyčky, ve které se již jednotlivé události provádí. V každém cyklu smyčky je aktualizován současný čas, který je okamžitě porovnán s první událostí (parametr startDateTime této události) v seřazeném listu naplánovaných snímání. Pokud si dané časy odpovídají je událost provedena, vytvořen záznam v log souboru a následně je událost odebrána z čela listu. Tento postup se opakuje, dokud není běh procesu přerušen. Smyčka obsahuje, mimo dříve zmíněné vykonávání událostí snímání, jejich opětovné načtení z databáze pro zajištění aktuálnosti záznamů. Tato aktualizace probíhá na základě ověření změny v databázi pomocí funkce `list<int> getLastIdAndCount()`, která navrací list o dvou hodnotách, a to hodnotu posledního taskId (identifikátor události) a celkový počet událostí zaznamenaných v databázové tabulce scheduledTasks. Pokud bychom kontrolovali pouze změnu hodnoty taskId není zaručeno, že nebyla některá z událostí v databázové tabulce smazána, a tudíž došlo ke změně. Stejně nevypovídající je pouze kontrola změny počtu naplánovaných událostí v případě přidání jedné a smazání jiné události snímání. Na základě těchto dvou parametrů lze tedy s jistotou detekovat změnu v databázových záznamech scheduledTasks, jejíž kontrola probíhá každých deset minut běhu celého cyklu (těchto deset minut je bráno následně v úvahu ve webové aplikaci, která slouží k přidávání událostí snímání). Vždy jsou však načteny aktualizované záznamy pouze do nadcházejícího poledne v rámci dříve nastaveného intervalu. Při dosažení konce zmíněného intervalu (poledne) je v cyklu navíc zajištěno načtení záznamů událostí snímání na dalších 24 hodin (interval je nastaven opět do nadcházejícího poledne a je vytvořen nový log soubor).



Obrázek 6.3.1: Flowchart diagram skriptu run.cpp [21]

6.4 Implementace webové aplikace, uživatelského rozhraní

Webová aplikace poskytující uživatelské rozhraní je vytvořena pomocí HTML, a především skriptovacího jazyka PHP zajišťující veškerou interaktivitu, práci s databází s minoritním využitím skriptovacího jazyka JavaScript (zejména pro lokální upozornění na akce uživatele).

Hlavním úkolem vytvořené webové aplikace je umožnit přidávání mimořádných událostí snímání experimentu WILLIAM. To je zajištěno pomocí skriptu add_new_task.php, který obsahuje formulář pro zadávání parametrů snímání a pro svou funkci využívá třídu task definovanou souborem task.php. Třída task podobně jako v případě C++ skriptů zajišťuje reprezentaci jednotlivých událostí snímání a metody pro práci s jejími atributy. Dále skript add_new_task.php využívá třídy database definovanou souborem database.php, jež jak již vyplývá z názvu poskytuje prostředky pro navázání komunikace s SQLite databází a práci s ní. Posledním skriptem, který je nutný pro úplnost skriptu add_new_task.php je skript dialog.php. Ten je využíván pro potvrzení přidání událostí snímání po vyhodnocení kolizí atd.

Druhým úkolem webové aplikace je umožnění změny základních nastavení řádných snímání. To je zajištěno skriptem edit_default_settings.php, který opět využívá dříve zmíněné třídy task a database pro svou funkci.

Oba skripty (add_new_task.php a edit_default_settings.php) spojuje jednoduchý rozcestník poskytnutý skriptem index.php. Blíže jsou jednotlivé PHP skripty popsány dále v textu.

6.4.1 Reprezentace události snímání

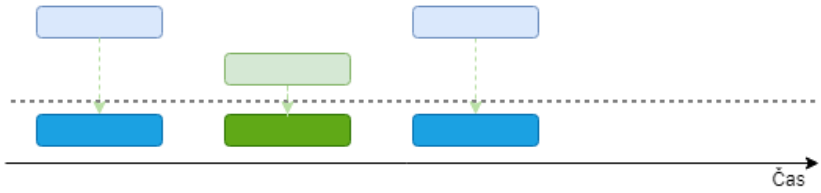
Pomocí skriptu task.php je definována třída task obsahující většinu atributů totožnou s databázovými parametry událostí snímání. Jedním z atributů, který třída obsahuje navíc je například parametr endDateTime, jež představuje čas konce události a je využit při vyhodnocování kolizí událostí. Zároveň třída obsahuje reprezentace časových hodnot nejen ve formátu string (textový řetězec), ale také vyjádření typu DateTime (datový typ pro vyjádření data a času) pro zjednodušení práce s časy. Její metody lze opět rozdělit na metody typu get (metody pro přístup k jednotlivým atributům a jejich hodnotám) a následně metody set zajišťující přidělení hodnot jejím atributům (`setStartDateTime(DateTime)`, `setEndDateTime(DateTime)` a `setTaskDelete()`). Jak již vyplývá z názvu metody `setStartDateTime(DateTime)` a `setEndDateTime(DateTime)`, jde o metody využívané pro nastavení hodnot startDateTime a endDateTime události snímání. V případě kolize dvou řádných událostí je potřeba první z nich odstranit (teoreticky není důvod pořizovat dvě řádná snímáni s velice blízkými časy provedení). K indikaci, že má být daná úloha smazána slouží metoda `setTaskDelete()`, která nastaví atribut taskPriority (hodnotu priority události) na hodnotu -1, jež je později využit k indikaci smazání události. Dalšími neméně důležitými metodami jsou konstruktor (`_construct(...)`) definující inicia-

lizaci objektu třídy task a komparátor (`comparator(task, task)`) určený pro porovnání dvou instancí třídy task.

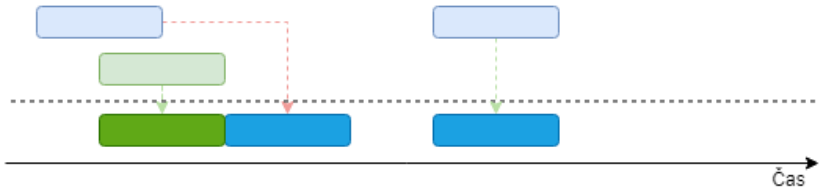
6.4.2 Práce s databází, plánování

Třída database obsahuje pouze dva atributy. Atribut databasePath (specifukující cestu k databázi), ve kterém je uložena hodnota cesty k databázovému souboru a atribut dbSetup (PDO spojení s databází) reprezentující spojení s databází. Atribut databasePath je nastaven v konstruktoru třídy database (`_construct(...)`), který slouží pro inicializaci před samotným spojením s databází. Následně je potřeba navázat spojení pomocí metody `dbOpen()`, jež inicializuje PDO spojení s databází (atribut dbSetup). Po ukončení práce s databází je nutné zavolat metodu `dbClose()`, která zajistí kontrolu případných chyb a ukončení spojení s databází.

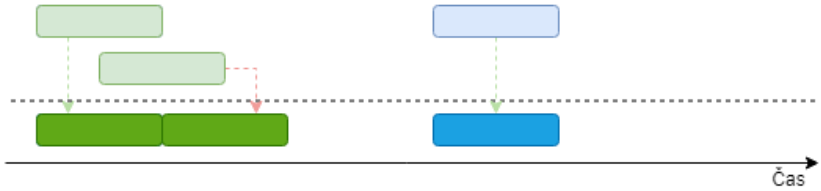
Třída database dále obsahuje množství metod pro načítání a přidávání záznamů do databáze, jejich princip funkce je poměrně jednoduchý. Je důležité zaměřit se především na metodu `scheduleNewTask(task)`, jež se stará o detekci a vyhodnocování kolizí v případě jejich vzniku při přidávání nové mimořádné události snímání či sekvence snímání (v tomto případě je sekvence rozdělena na jednotlivé události a ty jsou vyhodnocovány postupně). Algoritmus, který metoda obsahuje, byl navržen na základě poznatků shrnutých ve třetí kapitole. Celé rozhodování algoritmu spočívá v porovnávání řádných a mimořádných událostí snímání, které jsou již naplánovány a vkládané mimořádné události. Podmínky rozhodování závisí na počátečním a koncovém čase a prioritě každé události (startDateTime, endDateTime a taskPriority). Mimořádné události mají hodnotu taskPriority rovnu dvěma, a tedy vyšší prioritu než události řádné, u kterých je taskPriority rovna jedné. Pokud není detekována kolize vkládané mimořádné události s již naplánovanými událostmi (v datavázové tabulce scheduledTasks), je vkládané události přidělen požadovaný čas určený uživatelem. Jestliže naopak dojde ke kolizi, snaží se algoritmus najít nejbližší volný časový úsek, kde je možné událost (události) umístit. Jestliže jde o kolizi s mimořádnou událostí tak algoritmus nejdříve hledá volný časový úsek v rámci mimořádných událostí (dříve naplánované mimořádné události mají vyšší prioritu), jakmile jej nalezne přidělí vkládané události modifikovaný čas provedení. Poté je ještě potřeba zjistit, zda vkládaná událost není v kolizi s řádnou událostí, pokud ano, je naopak řádná událost posunuta do nejbližšího volného časového intervalu. Pokud je řádná událost posunuta natolik, že se překrývá s následující řádnou událostí, dojde k jejímu odstranění. Celý algoritmus řazení událostí a jednotlivé případy vyhodnocení kolizí jsou názorně ilustrovány níže na obrázku 6.4.1.



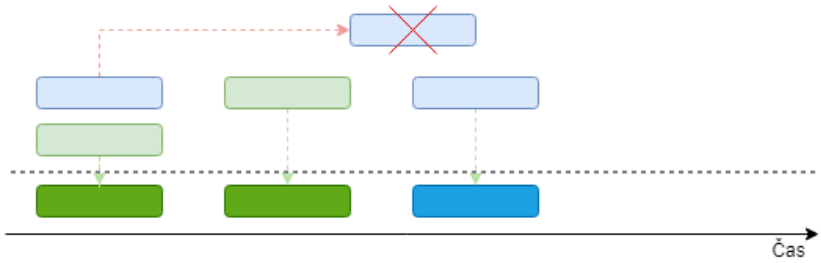
(a) : Jednoduchý bezkolizní případ vložení události



(b) : Příklad upřednostnění vkládané mimořádné události



(c) : Případ kolize dvou mimořádných událostí



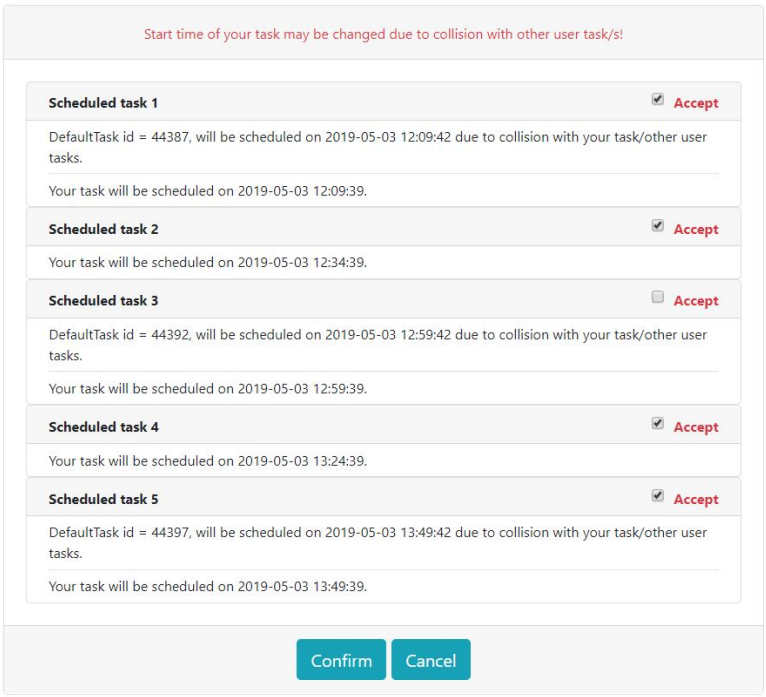
(d) : Složitější příklad, kde dochází k odstranění řádné události

Obrázek 6.4.1: Realizace vyhodnocování kolizí při přidávání mimořádné události snímání. Modře jsou vyznačeny řádné události, zeleně pak mimořádné události. První řádek reprezentuje již naplánované události, druhý představuje vkládanou událost a třetí výsledné řešení situace. [21]

6.4.3 Dialog zadání snímání

Skript dialog.php slouží jako mezikrok při přidávání mimořádné události do databáze (či sekvence mimořádných událostí). Jedná se o zobrazení přidávané události, případně přidávaných událostí v případě sekvence, a především zobrazení přiděleného času provedení snímání a kolizní události (včetně jejího posunutí) v případě kolize. Již při samotném návrhu byl kladen důraz ponechat uživateli co největší možnost rozhodování a kontroly při přidávání událostí snímání. Uživateli je tak poskytnuta rekapitulace událostí, které chce

přidat, zároveň s vyobrazením změn, jež bylo nutné pro vhodné naplánování událostí provést. Na pravé straně každé přidávané události je možné případně jednoduše některou z nich vyřadit a konkrétní změny tak nebudou provedeny. Kliknutím na tlačítko Confirm uživatel potvrdí přidání událostí snímání a je tak pro každou z vybraných událostí zavolána metoda třídy database `addTaskToDB(task)` případně `updateTaskInDB(task)` či `deleteTaskFromDB(...)`, jež provede změny v databázi (v případě sekvence událostí jsou události přidávány postupně). Vše je ilustrováno na obrázku číslo 6.4.2.



Obrázek 6.4.2: Náhled obrazovky pro potvrzení přidání událostí (dialog.php)

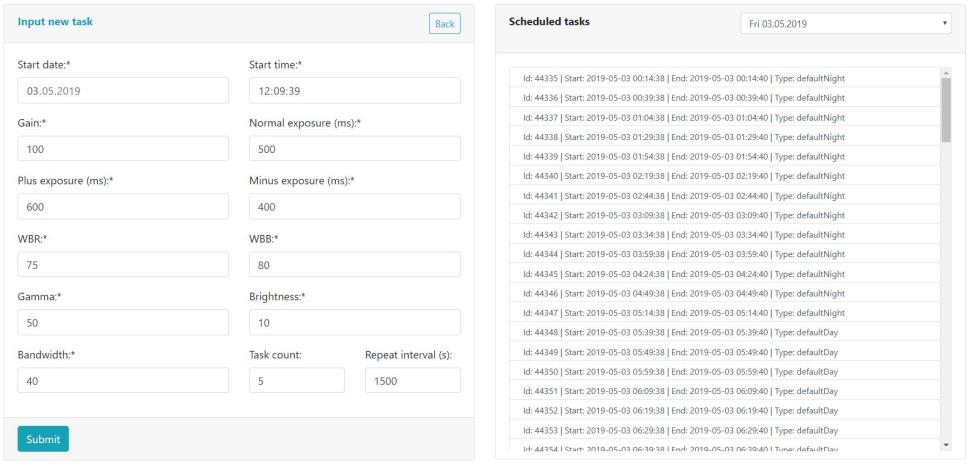
6.4.4 Přidání nové události snímání

Na začátku kapitoly bylo zmíněno, že skript add_new_task.php zajišťuje uživatelské prostředí pro přidávání nových mimořádných událostí snímání. Náhled na celé řešení je k vidění níže na snímku 6.4.3. Prostřednictvím poměrně jednoduchého formuláře lze definovat neomezené množství nastavení mimořádné události či sekvence mimořádných událostí. Při potvrzení formuláře tlačítkem Submit dojde k volání metody `scheduleNewTask(task)` třídy database popsané v podkapitole 6.4.2. Data událostí spolu s přidělenými časy jsou pak předána dále skriptu dialog.php pomocí PHP Session[3], kterým lze přidání událostí potvrdit či zamítnout.

Zároveň jsou zde vyobrazeny již naplánované události, které jsou pomocí metody `getScheduledWeekFromDB(string)` třídy database (jejímž jedním parametrem je současný čas v textovém formátu) načteny z SQLite data-

[3] https://www.php.net/manual/en/intro.session.php

báze, pro poskytnutí přehledu naplánovaných událostí snímání. Intuitivně lze pomocí rozbalovacího listu vybrat požadovaný den, pro který chceme naplánované události procházet a tím je usnadněno prohlížení velkého počtu databázových záznamů. V seznamu naplánovaných událostí je každému snímání přiděleno tlačítko Del-task, které slouží ke smazání dané události, respektive Del-seq pro smazání celé sekvence událostí (sekvence událostí jsou identifikovány parametrem parentTaskId). Tato možnost je poskytnuta z důvodu teoretické potřeby smazat především danou mimořádnou událost či sekvenci událostí, v případně nutnosti umístění jiné události v tentýž časový interval.



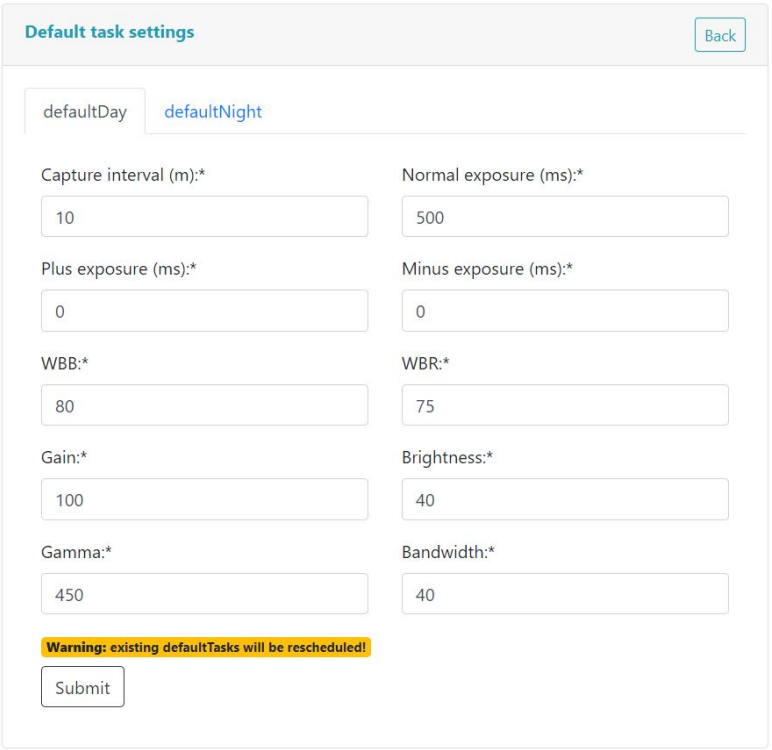
Obrázek 6.4.3: Náhled obrazovky pro přidávání mimořádných událostí (add_new_task.php)

6.4.5 Změna základního nastavení řádných událostí

Změnu základního nastavení řádných událostí snímání poskytuje uživatelské prostředí a funkce definované skriptem edit_default_settings.php. Zde jsou využívány metody `getDefaultSettingsFromDB()` třídy database pro načtení základních parametrů řádných událostí. Dále je poskytnut formulář pro změnu parametrů, jak pro událost typu defaultDay, tak typu defaultNight. Při změně parametrů řádné události a potvrzení tlačítkem Submit dojde hned k několika úkonům.

Nejdříve se provede vytvoření lokální kopie seznamu mimořádných událostí snímání, jež byly naplánovány, pomocí metody `getScheduledUserTasksFromDB(string)`. Metoda `getScheduledUserTasksFromDB(string)` třídy database obdrží jako parametr současný čas v textovém řetězci a zajistí načtení mimořádných událostí z databáze. Následně jsou smazány databázové záznamy naplánovaných událostí z tabulky scheduledTasks, upravena základní nastavení řádných událostí a aktualizována v databázi pomocí metody `updateDefaultTaskInDB(task)`. Jakmile jsou změněny parametry řádných událostí v databázi je zavolána PHP funkce `exec(...)`, která umožňuje vykonat zadaný příkaz jako je tomu v příkazové řádce. Funkce `exec(...)` vyvolá generování řádných událostí (příkazem `./generator`), jež je definováno C++

skriptem generator. Ve finálním kroku je pak využita dříve vytvořená lokální kopie seznamu mimořádných událostí snímání. Ty jsou postupně přidávány do databázové tabulky naplánovaných událostí (scheduledTasks) stejným principem jako při manuálním přidávání mimořádné události. Postupně je jim přiřazen čas provedení metodou `scheduleNewTask(task)`, a následně jsou aktualizovány záznamy databáze metodou `addTaskToDB(task)`, respektive `updateTaskInDB(task)` či `deleteTaskFromDB(...)`. Tím je celý proces změny základního nastavení řádné události snímání ukončen.



Obrázek 6.4.4: Náhled obrazovky pro změnu nastavení řádných událostí (edit_default_settings.php)

Kapitola 7

Nasazení plánovače do provozu a testování

Výsledné testování pokročilého plánovače pro autonomní ovládání experimentu WILLIAM bylo prováděno na snímacím zařízení, jež je umístěno v Praze Dejvicích na budově ČVUT FEL. Zde je rovněž umístěn server zpracovávající úlohy snímání, kde byl nasazen implementovaný plánovač.

V rámci nasazovaní pokročilého plánovače do provozu a jeho umístění na cílové zařízení (server), bylo potřeba částečně uzpůsobit jak skripty webové aplikace (pro využití v redakčním systému WordPress), tak také C++ skripty. Následně bylo nutné zkompilovat zdrojové kódy C++ pro cílovou platformu (GNU/Linux) a přidat automatické spouštění generátoru řádných událostí do systémového plánovače Cron.

Při instalaci databáze bylo nutné zajistit dostatečná oprávnění přístupu do adresáře, ve kterém je samotná databáze umístěna. Rozhraní PDO poskytované programovacím jazykem PHP (webová aplikace) pro navázání spojení s databází a práci s ní k tomu využívá metodu Write-Ahead Logging (WAL)[1]. Tato metoda zachovává původní databázi (databáze A) a vytváří její dočasnou kopii (databáze B), ve které jsou prováděny požadované změny. Jestliže je proces zapisování změn úspěšně dokončen, jsou změny přeneseny do původní databáze (databáze A). Je tedy nutné oprávnění k zapisování (případně čtení), nejen do samotného souboru databáze, ale také do složky, v níž se databázový soubor nachází (kde je vytvářena dočasná kopie databáze).

Pro přímočaré nasazení a kompilaci C++ skriptů byl vytvořen Makefile[2], který definuje jednotlivé závislosti mezi zdrojovými soubory a jejich kompilaci. Jediným příkazem `make` v příkazové řádce GNU/Linux systému, tak lze zkompilovat všechny zdrojové soubory do spustitelné formy.

Testování plánovače bylo zaměřeno především na funkci generování řádných událostí a přidávání mimořádných událostí pomocí webové aplikace (případně na změnu parametrů řádných událostí snímání). Na závěr byla správnost funkce plánovače ověřena kontrolou skutečně realizovaných snímků dle nastavení plánovače, jejich parametrů apod.

[1] https://www.sqlite.org/wal.html

[2] https://www.gnu.org/software/make/manual/make.html

Kapitola 8

Závěr

V rámci práce bylo dosaženo cíle, který byl stanoven na základě požadavků a zadání práce. Během prvotního seznamování se systémem WILLIAM a metodami plánování procesů, se autor seznámil s novými způsoby, jak řešit problémy spojené s plánováním real-time procesů (událostí snímání oblohy). Bylo nutné zvolit co nejvhodnější metodu a tu následně implementovat. Při samotné přípravě návrhu a implementaci řešení přišel autor do styku s novými programovacími jazyky (především PHP) a novými technologiemi (GNU/Linux). V průběhu vývoje plánovače také vzniklo několik komplikací, například s kompatibilitou platforem Microsoft Windows a GNU/Linux, nebo s uchováváním dat webové aplikace pomocí cookie (PHP Session). Obdobně tomu bylo při nasazení pokročilého plánovače do provozu a jeho testování, kde vznikly především komplikace se zapouzdřením webové aplikace do redakčního systému WordPress. Tyto záležitosti však byly vyřešeny a podařilo se úspěšně plánovač událostí zprovoznit v reálném prostředí.

Webová aplikace spolu s C++ skripty, jež vytváří kompletní systém pokročilého plánovače pro autonomní ovládání experimentu WILLIAM, je již dnes nasazena v testovacím provozu na zařízení umístěném na budově ČVUT FEL v Praze Dejvicích. Plánovač je navržen tak, aby poskytl množství možných rozšíření a je tedy očekáváno jeho další využití a vývoj. Případné rozšíření plánovače by mohlo umožnit nejen plánování snímání oblohy, ale také plánování měření elektrických či neelektrických veličin (například pomocí připojené meteorologické stanice a dalších senzorů). Jako jedno z dalších rozšíření aplikace se nabízí odlišení jednotlivých uživatelů, kteří mohou spravovat události snímání. Také by bylo například možné přidat více základních nastavení řádných událostí snímání pro docílení ještě lepších výsledků pravidelného snímání apod.

Literatura

[1] WILLIAM – WIde-field aLL-sky Image Analyzing Monitoring system. *william.multimediatech.cz* [online]. [cit. 2019-04-15]. Dostupné z: http://william.multimediatech.cz/

[2] VÍTEK, S., P. PÁTA, P. KOTEN a K. FLIEGEL. *Long-Term Continuous Double Station Observation of Faint Meteor Showers*. Sensors(Basel), 2016. DOI 10.3390/s16091493

[3] JANOUT, Petr a PÁTA Petr. *Celooblohová kamera s extrémně širokoúhlým zorným polem*. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky.

[4] JANOUT, P., M. BLAŽEK a P. PÁTA. *New generation of meteorology cameras*. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Radioelectronics.

[5] JANOUT, P., P. PÁTA, J. BEDNÁŘ, E. ANISIMOVA, M. BLAŽEK, P. SKALA. *Stellar objects identification using wide-field camera*. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Radioelectronics.

[6] BAXA, Jan. *Využití digitálního fotoaparátu pro snímání oblohy*. Praha, 2013. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra měření.

[7] KRAUZ, Lukáš. *Detekce mraků podle barevné informace z celooblohových kamer*. Praha, 2019. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky.

[8] Installation to Prague. *william.multimediatech.cz* [online]. [cit. 2019-04-15]. Dostupné z: http://william.multimediatech.cz/

[9] Raspbian FAQ. *Raspbian.org* [online]. [cit. 2019-04-15]. Dostupné z: https://www.raspbian.org/RaspbianFAQ

[10] KADLEC Josef. Obecné pojednání o programovacích jazycích. In: *Linuxsoft.cz* [online]. 2004-07-16 [cit. 2019-04-15]. Dostupné z: https://www.linuxsoft.cz/article.php?id_article=268

[11] MOHAMMADI, Arezou a Selim G. AKL. Scheduling algorithms for real-time systems. [online]. 2005 [cit. 2019-04-23]. Dostupné z: http://research.cs.queensu.ca/home/akl/techreports/scheduling.pdf

[12] KLEINBERG, Jon a Éva TARDOS. *Algorithm design*. Pearson Education, Inc., 2006. ISBN 0-321-29535-8

[13] Software Architecture & Design Tutorial. *tutorialspoint.com* [online]. ©2019 [cit. 2019-05-03]. Dostupné z: https://www.tutorialspoint.com/software_architecture_design/

[14] About SQLite. *SQLite.org* [online]. [cit. 2019-05-03]. Dostupné z: https://www.sqlite.org/about.html

[15] SQLite Download Page. *SQLite.org* [online]. [cit. 2019-05-03]. Dostupné z: https://sqlite.org/download.html

[16] BUELOWP *SunSet* [software]. [přístup 3. května 2019]. Dostupné z: https://github.com/buelowp/sunset

[17] COOLEY, S., M. WOJCIAKOWSKI, M. SATRAN, T. RAJ, TASSOMAN, RENZOK. Windows Subsystem for Linux Documentation. In *Microsoft Docs* [online] 2016-07-11 [cit. 2019-05-03]. Dostupné z: https://docs.microsoft.com/en-us/windows/wsl/about

[18] What can PHP do? *PHP.net*[online]. ©2001-2019 [cit. 2019-05-03]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php

[19] NEWMAN, K., R. MCMURRAY, M. WENZEL and S. PATEL. Configure a PHP Website on IIS. In *Microsoft Docs* [online] 2013-04-14. [cit. 2019-05-03]. Dostupné z: https://docs.microsoft.com/en-us/iis/application-frameworks/scenario-build-a-php-website-on-iis/configure-a-php-website-on-iis

[20] PHP PDO Tutorial. *ZenTut.com* [online]. ©2019 [cit. 2019-05-03]. Dostupné z: http://www.zentut.com/php-pdo/

[21] Flowchart Maker & Online Diagram Software. *draw.io* [online]. ©2005-2019 [cit. 2019-04-15]. Dostupné z: https://www.draw.io/

Seznam použitých zkratek a pojmů

ASI Astronomy Imaging camera - astromický snímač.

C++ Multiparadigmatický programovací jazyk C++.

captureInterval Časové vyjádření intervalu řádné události v milisekundách.

CMOS Complementary Metal–Oxide–Semiconductor.

defaultDay Řádná (pravidelná) denní událost snímání.

defaultNight Řádná (pravidelná) noční událost snímání.

defaultTask Řádná (pravidelná) událost snímání.

DSLR Digital Single-Lens Reflex - digitální zrcadlovka.

endDateTime Koncový (end) čaš události snímání.

FIFO First In, First Out - metoda řízení, fronta.

FOV Field of view - zorné pole.

fullTaskDuration Časové vyjádření trvání události v milisekundách.

GPS Global Positioning System - globální polohový systém.

HTML Hypertext Markup Language - standardizovaný značkovací jazyk.

IIS Internet Information Services - Internetová informační služba.

JPEG Joint Photographic Experts Group - souborový formát.

LRGB Luminance, Red, Green and Blue - Jas, červená, zelená a modrá.

MAIA Meteor Automatic Imager and Analyzer stations.

Makefile Soubor definující kompilaci a závislosti zdrojových souborů.

NEF Nikon Electronic File - souborový formát.

PDO PHP Data Objects - rozhraní pro spojení s databází.

PHP Hypertext Preprocessor - skriptovací programovací jazyk.

RAW Souborový formát.

scheduledTask Naplánovaná událost snímání.

SFTP SSH File Transfer Protocol - zabezpečený protokol pro přenos souborů.

SQL Structured Query Language - standardizovaný strukturovaný dotazovací jazyk.

SSH Secure Shell - zabezpečený komunikační protokol.

startDateTime Počáteční (start) čaš události snímání.

taskId Jedinečný numerický identifikátor události.

taskPriority Priorita události snímání.

userTask Mimořádná (uživatelem definovaná) událost snímání.

WAL Write-Ahead Logging - metoda přístupu k databázi.

WILLIAM Wide-field all-sky image analyzing monitoring system.

WSL Windows Subsystem for Linux.

ČVUT FEL České vysoké učení technické v Praze, Fakulta elektrotechnická.

Přílohy

Příloha A - C++ zdrojové kódy

- task.cpp, task.h
- generator.cpp
- run.cpp
- Makefile

Příloha B - PHP skripty

- task.php
- database.php
- dialog.php
- add_new_task.php
- edit_default_settings.php
- index.php

Příloha C - SQLite databázový soubor

- scheduler.db

Příloha D - Zdrojové kódy SunSet

- sunset-master.zip