Detekce objektů v obraze pro rozšířenou realitu

Tomáš Hanáček

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Hanáček**
Osobní číslo:	**A20005**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Detekce objektů v obraze pro rozšířenou realitu**
Téma práce anglicky:	**Object Detection for Augmented Reality**

Zásady pro vypracování

1. Prostudujte vizuální kódy pro značkování objektů, které podporují snímání kamerou v různých úhlech natočení, např. kruhový čárový kód, obrázkové kódy, QR a AR kódy atd.
2. Ověřte spolehlivost detekce několika typů vizuálních kódů v různých podmínkách a vyberte značkovací kód, který pro zadanou aplikaci rozšířené reality poskytuje nejlepší výsledky.
3. Pro zadanou aplikaci rozšířené reality implementujte knihovnu pro detekci a dekódování značek v obraze.
4. Pro zadanou aplikaci rozšířené reality navrhněte možnou HW architekturu a způsob komunikace mezi použitými HW zařízeními.
5. Implementujte ukázku detekce objektů pomocí Raspberry PI s kamerou a odesílání detekovaných objektů přes rozhraní WiFi nebo Bluetooth.

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

Seznam doporučené literatury:

1. DONAT, Wolfram. Learn Raspberry Pi Programming with Python: Learn to Program on the World's Most Popular Tiny Computer. 2nd ed. 2018. Imprint: Apress, 2018. ISBN 9781484237694.
2. BRADSKI, Gary R. a Adrian KAEHLER. Learning OpenCV. Sebastopol: O'Reilly, c2008. ISBN 9780596516130.
3. SUMMERFIELD, Mark. Python 3: výukový kurz. Brno: Computer Press, 2010. ISBN 9788025127377.
4. KABELOVÁ, Alena a Libor DOSTÁLEK. Velký průvodce protokoly TCP/IP a systémem DNS. 5., aktualiz. vyd. Brno: Computer Press, 2008. ISBN 9788025122365.
5. KAEHLER, Adrian a Gary R. BRADSKI. Learning OpenCV 3: computer vision in C++ with the OpenCV library. Sebastopol: O'Reilly, 2016. ISBN 9781491937990.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

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

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15.5.2023

Tomáš Hanáček v.r.
…….
podpis studenta

ABSTRAKT

Bakalářská práce se zabývá zpracováním obrazu z kamery na Raspberry Pi pomocí jazyka Python a knihovny OpenCV a řešením komunikace s chytrým zařízením (mobilním telefonem, tabletem) pro přenos získaných dat. Data jsou na chytrém zařízení prezentována pomocí ukázkové aplikace implementované ve frameworku Unity, která je základem pro implementaci rozšířené reality. Praktická část obsahuje testování a výběr vizuálních kódů značek pro detekci a identifikaci fyzických předmětů v obraze kamery a dále také testování a implementaci bezdrátové komunikace mezi Raspberry Pi a chytrým zařízením, na kterém jsou data přijímána pomocí ukázkové mobilní aplikace ve frameworku Unity. Výsledkem bakalářské práce je zařízení stolní herní plochy s Raspbery Pi, které po připojení k napájení vytvoří veřejný přístupový bod a odesílá souřadnice hracích kostek na herní ploše na telefon nebo tablet připojený k tomuto přístupovému bodu, zatímco tablet nebo telefon pomocí své kamery vykreslí 3D objekty rozšířené reality na získaných souřadnicích, tj. nad hracími kostkami.

Klíčová slova: Python, OpenCV, Raspberry Pi, detekce, video, komunikace, data, virtuální přístupový bod, chytré zařízení.

ABSTRACT

The bachelor thesis deals with the image processing of the video from camera on Raspberry Pi using Python and OpenCV and the communication with a smart device (mobile phone, tablet) for data transmission. The data is presented on the smart device using a sample application implemented in the Unity framework, which is the basis for the implementation of augmented reality. The practical part includes testing and selection of visual marker codes for detection and identification of physical objects in the camera image. It also includes testing and implementing wireless communication between the Raspberry Pi and the smart device on which the data is received using a sample mobile application in the Unity framework. The result of the bachelor thesis is a desktop game board device with a Raspbery Pi that, when connected to power, creates a public access point and sends the coordinates of the detected physical objects on the game board to a phone or tablet connected to this access point, while the tablet or phone uses its camera to render 3D augmented reality objects on the received coordinates, hence over the detected physical objects.

Keywords: Python, OpenCV, Raspberry Pi, Detection, Video, Communication, Data, Virtual Access Point, Smart Device

Chtěl bych poděkovat Ing. Tomášovi Dulíkovi, Ph.D. za odborné vedení, vstřícnost při konzultacích a cenné rady při zpracování této práce

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Od dob, kdy nám výpočetní technika umožnila zpracování velkých objemů dat udělal obor počítačového vidění obrovský pokrok. Nyní je strojové vidění znatelně rozšířeno například v průmyslové výrobě, např. k vizuální kontrole nebo počítání a sledování jednotlivých polotovarů na lince. Rozšiřování strojového vidění se dostává do fáze, kde jej bude možno spatřit na každém kroku, ať už při identifikaci osob ve veřejných prostorech, nebo vozidel na silnicích. Další využití má v medicíně, kde je získávání dat z mikroskopů a snímků z vyšetření velmi nápomocné při záchraně lidských životů.

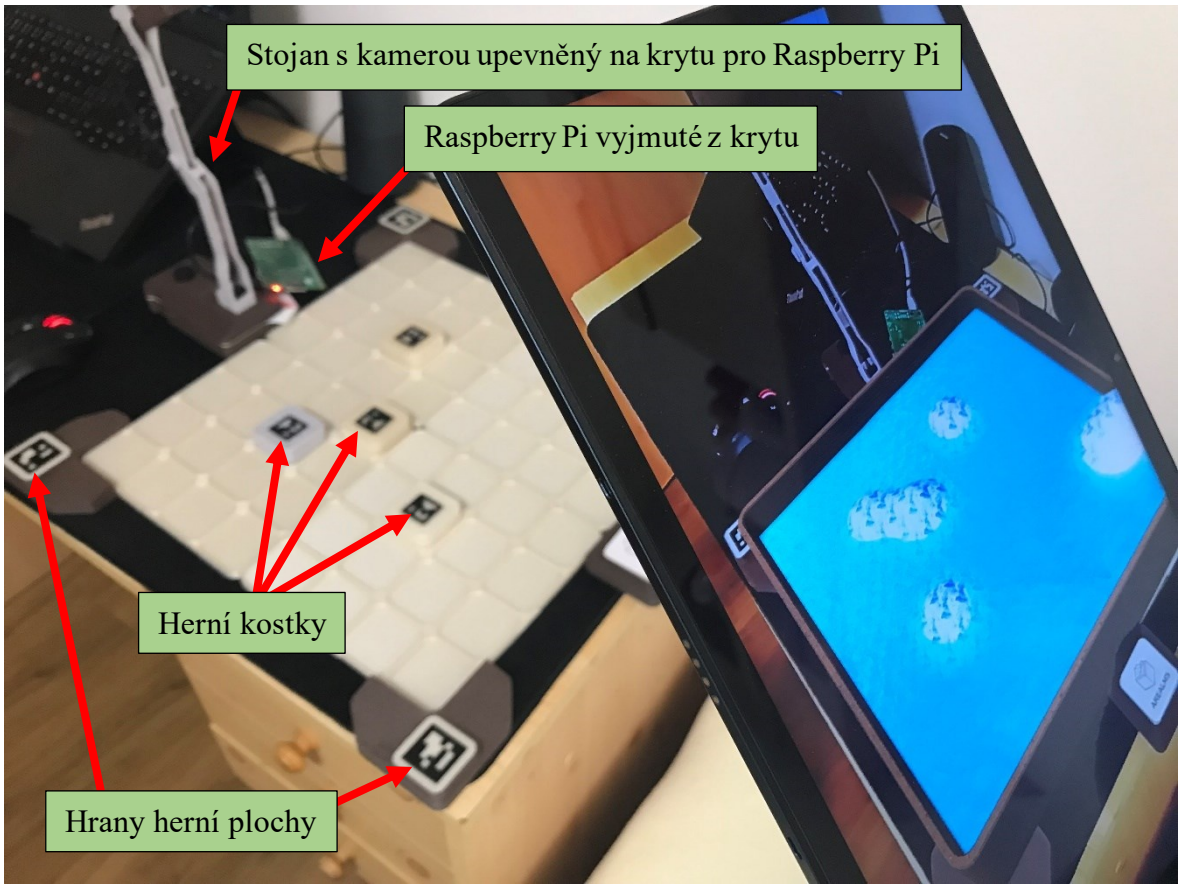
Tato bakalářské práce se zabývá strojovým viděním pro implementaci rozšířené reality. Celý projekt je realizován ve spolupráci s Fakultou multimediálních komunikací (FMK) a je podpořen Interní grantovou agenturou (IGA) UTB ve Zlíně. Cílem tohoto IGA projektu je vytvoření fyzické platformy stolní herní plochy s rozšířenou realitou, která může sloužit jako výuková pomůcka nebo zábavná hra určená pro uživatele nižší věkové kategorie. Příkladem jedné z výukových aplikací je hra na procvičení přírodopisu či zeměpisu.

Zatímco implementaci grafické části projektu s rozšířenou realitou realizuj e nositel projektu – Ing. Štěpán Dlabaja, Ph.D. z FMK, tato bakalářská práce se zabývá návrhem hardware a implementací software pro zpracování obrazu z kamery a pro komunikaci.

Prvním úkolem této bakalářské práce byl průzkum a testování vizuálních kódů pro označení herních kostek a hranic stolní herní plochy, aby byly snadno detekovatelné ve videu z kamery a mohly být zobrazovány a manipulovány v rozšířené realitě. Dalším cílem je výběr a implementace bezdrátové komunikace pro přenos získaných dat do vizualizační aplikace. V neposlední řadě je potřeba vytvořit ukázkovou aplikaci pro příjem a zobrazení dat na mobilním telefonu.

Teoretická část práce uvede čtenáře do návrhu hardwarové architektury projektu, včetně výběru potřebných komponent pro bezproblémovou funkčnost projektu. Zde spadá výběr vhodného výpočetního zařízení, kamery pro snímání obrazu a chytrého zařízení pro získání a zobrazení dat v reálném čase. Po návrhu HW architektury v teoretické části práce představíme základní vizuální kódy a jejich podskupiny. Dále následuje popis zvoleného programovacího jazyka, vývojového prostředí a seznam použitých knihoven, které byly použity pro vypracovávání praktické části.

Praktická část obsahuje podrobný popis při vypracovávání práce a zákoutí, se kterými bylo nutné se vypořádat pro bezproblémovou funkčnost. Tato část začíná přípravou prostředí, jako instalací OS na výpočetní zařízení, instalací potřebných knihoven, testování dostupných kamerových modulů a jejich kalibrací. Dále se praktická část práce ubírá k testování vizuálních kódů a následnému přečtení základních informací jako ID a poloha herních kostek a hranic herní plochy, které je potřeba získat pro splnění cíle práce. Další částí je implementace komunikace potřebné pro přenos informací z Raspbery PI do mobilního telefonu. V poslední kapitole je popsáno vytvoření jednoduché mobilní aplikace pro ukázku příjmu dat na chytrém zařízení coby základu pro vytvoření finální softwarové aplikace s rozšířenou realitou, kterou tvoří spoluřešitel IGA projektu Ing. Štěpán Dlabaja, Ph.D. na FMK UTB ve Zlíně.



Obrázek 1: Spuštěná aplikace na tabletu snímající herní plochu s herními kostkami

I. TEORETICKÁ ČÁST

1 ROZŠÍŘENÁ REALITA

Rozšířená realita (augmented reality) má za úkol propojit fyzický a digitální svět. Je to vylepšená verze reálného prostředí, vytvářená pomocí vizuálních digitálních prvků.

Existuje několik druhů realit: [23]

- Rozšířená realita (AR) - přidává digitální prvky do zobrazení reálného světa
- Virtuální realita (VR) - cílem je izolovat uživatele od reálného světa, obvykle prostřednictvím náhlavní soupravy a sluchátek určených pro tyto činnosti.
- Hybridní realita (MR) - kombinuje prvky rozšířené reality a virtuální reality tak, aby digitální objekty mohly interagovat s reálným světem
- Kombinovaná realita (XR) - zahrnuje všechny typy technologií, které zlepšují naše smysly, včetně tří výše uvedených typů

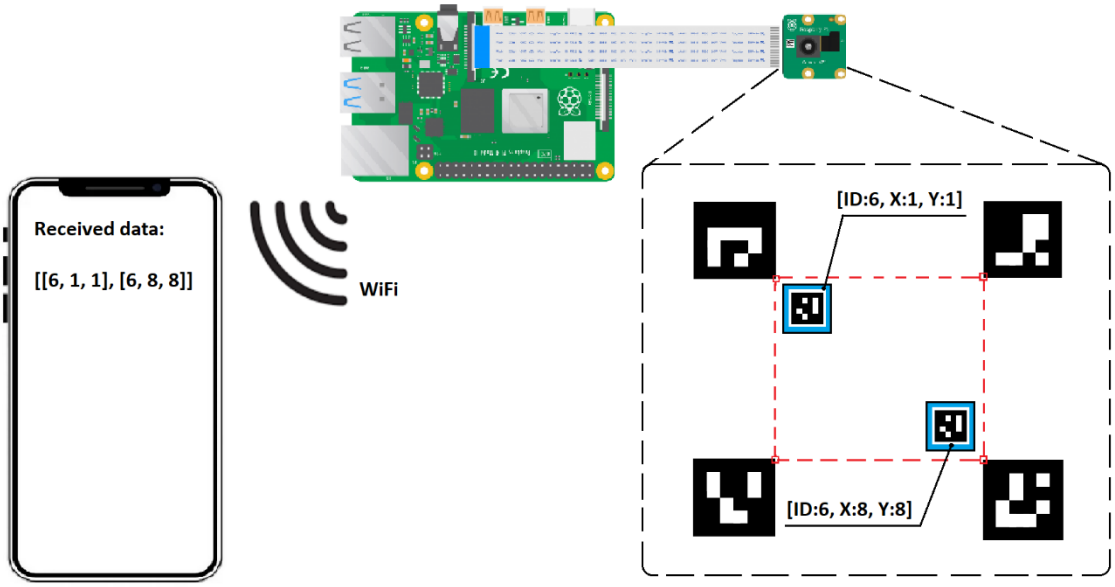
Příklad využití AR je možné pozorovat na sociálních sítí, kde existují filtry využívající kameru mobilního zařízení, která snímá obličej uživatele a aplikace dokresluje uživateli knírek, brýle, případně čepici. Dále je možné AR využívat pro školení pracovníků, při používání GPS navigace, při vytváření realistických simulací atd.

V bakalářské práci budeme pomocí AR snímat značení, nad kterým budou přes kameru telefonu či tabletu zobrazovat grafické objekty, se kterýma může uživatel interagovat posouváním herních kostek. Takováto aplikace bude využívat tzv. kotevního bodu. Existují aplikace nevyužívající kotevních bodů, které jsou založené pouze na aktuální poloze zařízení.

Obrázek 2: Příklad využití AR [24]

2 NÁVRH ARCHITEKTURY HARDWARE

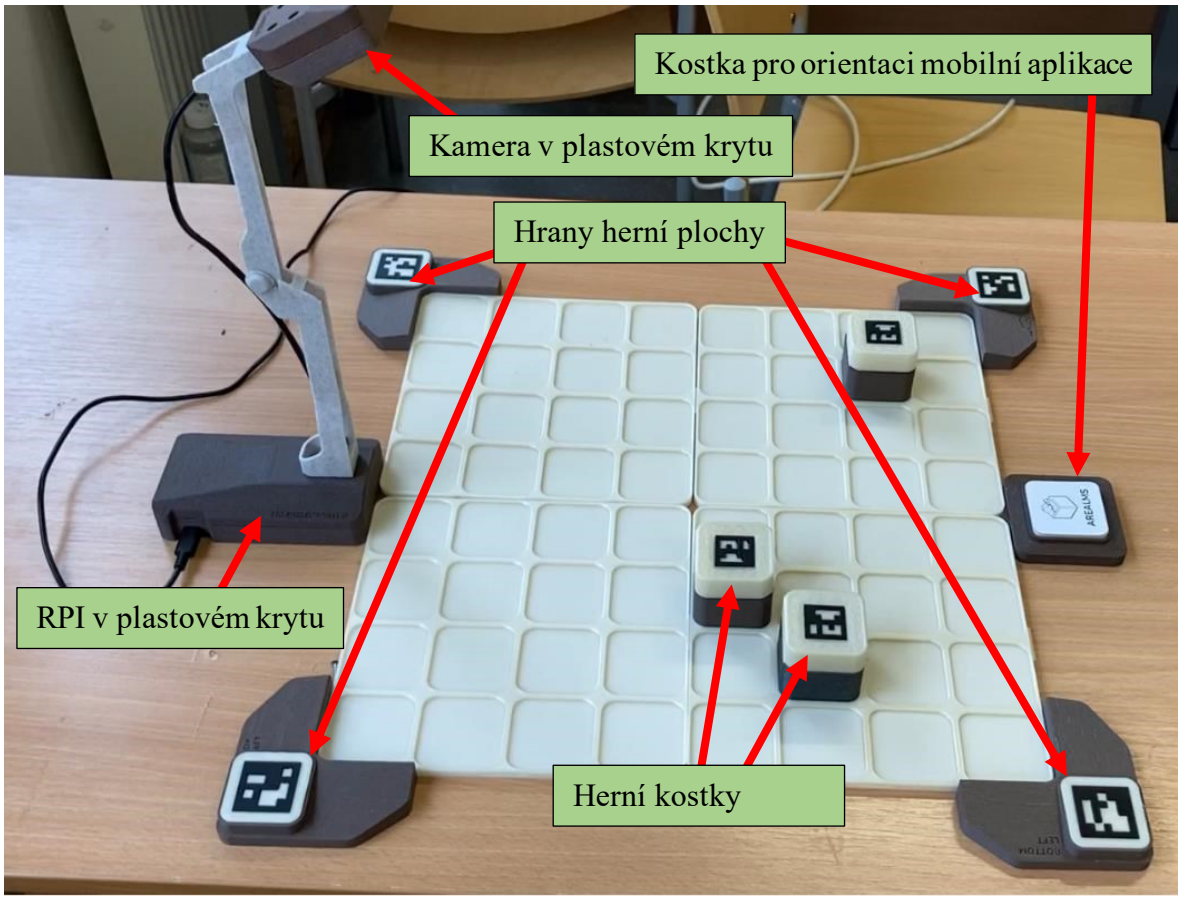
Z důvodu rozmanitosti různých typů kamer používaných v chytrých zařízeních a toho, že každé zařízení nedisponuje stejným výpočetním výkonem, bylo na začátku grantového projektu IGA rozhodnuto, že pro zajištění nejlepší uživatelské přívětivosti celého systému a zachování nízké ceny HW stolní herní plochy bude HW architektura postavena ze třech základních komponent. Tyto komponenty umožňují zaznamenat obraz v reálném čase (kamera), dodávají výpočetní výkon pro zpracování získaného snímku (Raspberry Pi) a zprostředkovávají rozhraní pro zobrazení získaných dat (chytré zařízení).



Obrázek 3: Schéma architektury hardware

2.1 Herní plocha

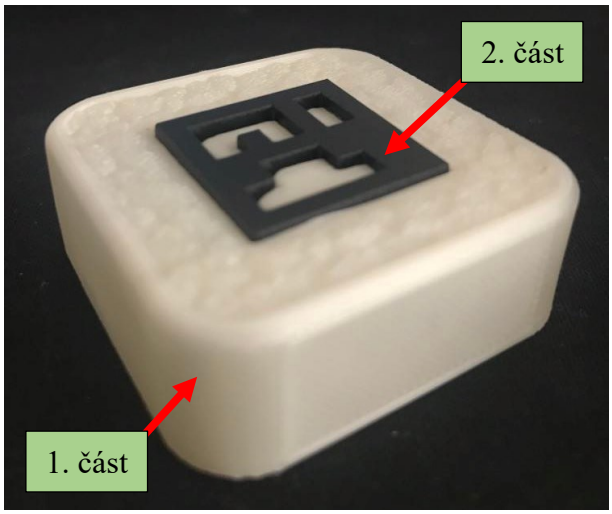
Herní plocha představuje předem vyhrazený herní prostor, do kterého je možné vkládat herní kostky a přemisťovat je. Skládá se ze čtyř plastových částí, vytisknutých na 3D tiskárně a přiložených k sobě tak, aby tvořily čtvercovou herní plochu. Jednotlivé díly mají po povrchu zapuštěné části pro snadnější ukládání kostek a lze díky nim definovat i přesnou herní pozici kostky. Herní plocha pro umisťování kostek je rozšířena o plastové hrany, taktéž vytisknutých pomocí 3D tiskárny, nesoucí ArUco markery, které slouží pro definování herní plochy z obrazu v Raspberry Pi. Vedle herní plochy se nachází plastový kryt pro Raspberry Pi a stojan na kameru, díky kterému je možné herní desku snímat z požadované výšky. Na druhé straně herní plochy, naproti kamery je umístněná kostka sloužící pro orientaci mobilní aplikace v prostoru a pro vykreslení herní grafiky nad hrací plochou.



Obrázek 4: Herní plocha a její jednotlivé části

2.2 Herní kostky

Herní kostky umožňují uživateli interagovat s aplikací a dochází díky nim k vykreslení konkrétního grafického objektu na dané pozici. Kostky jsou stejně jako herní plocha vytvořené na 3D tiskárně a skládají se ze dvou plastových částí zapadajících do sebe.



Obrázek 5: Herní kostka rozdělená na dvě části

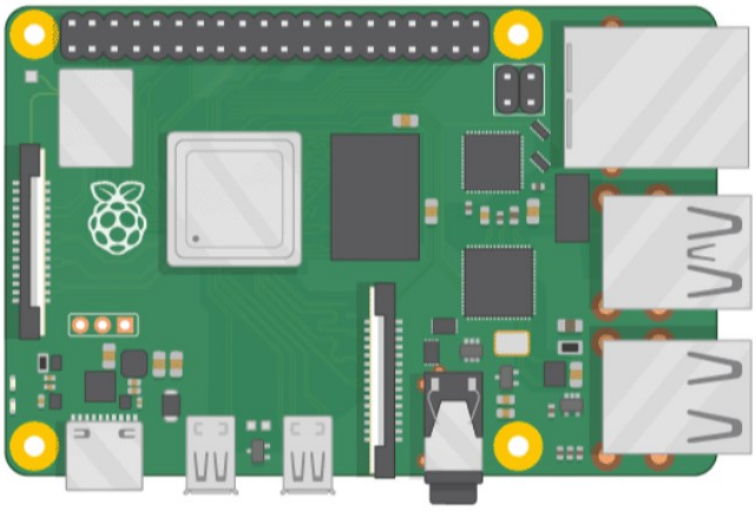
2.3 Raspberry Pi

Jedná se o malý jednodeskový počítač, ke kterému je možné připojit periferie jako monitor, klávesnici, myš atd. Je to zařízení, které zprostředkovává nenáročné uvedení do světa programování například pomocí jazyka Python. Od Raspberry Pi může uživatel očekávat téměř vše co očekává od stolního počítače, s výjimkou omezeného výpočetního výkonu.

Eben Christopher Upton vytvořil Raspberry Pi ve Velké Británii. První model byl uveden v únoru roku 2012. Jméno „Raspberry" je poctou slavným počítačovým společnostem jako např. Apple, Blueberry a Tangerine Computer Systems.

2.3.1 Raspberry Pi 4 Model B - 8GB RAM

Model 4B je momentálně nejvýkonnější z rodiny Raspberry. Velikost operační paměti typu LPDDR4 čítá 8GB RAM. Obsahuje 1.5GHz čtyřjádrový procesor ARM Cortex-A72 zhotoven 28nm technologií, díky čemuž tento model dosáhnul významného zvýšení výkonu. Napájecí konektor je typu USB-C, nachází se zde dva microHDMI konektory, ke kterým může uživatel připojit dva monitory zároveň a další konektory jako USB 3.0, Ethernet konektor, atd. [25]

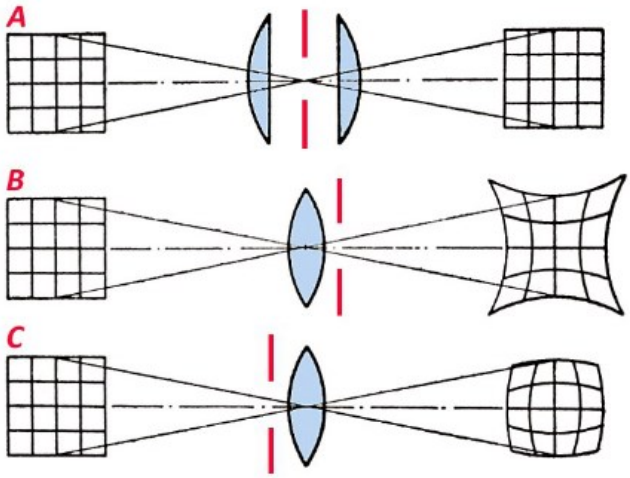


Obrázek 6: Raspberry Pi 4 Model B [25]

2.4 Kamera

Kamera je typ zařízení, které pořizuje za krátký časový úsek velké množství snímků, díky čemuž se snímky jeví jako souvislý obraz. Slouží tedy k zachycení pohyblivého obrazu. Kamera je pro tuto práci zvolená typu 190° rybí oko, z důvodu širokého spektra záběru při

zachování nízké vzdálenosti od snímaného objektu. Zachycený obraz obsahuje značnou deformaci a je nutné ho proto zkalibrovat a odstranit tak deformaci za pomocí kalibračního vzoru čtvercové tabule. Deformace obrazu u zvolené kamery je soudkového typu.



Obrázek 7: typ A – obraz bez zkreslení, typ B – zkreslení obrazu typu poduška, typ C – zkreslení obrazu typu soudek [26]

2.4.1 Raspberry Pi kamera V2

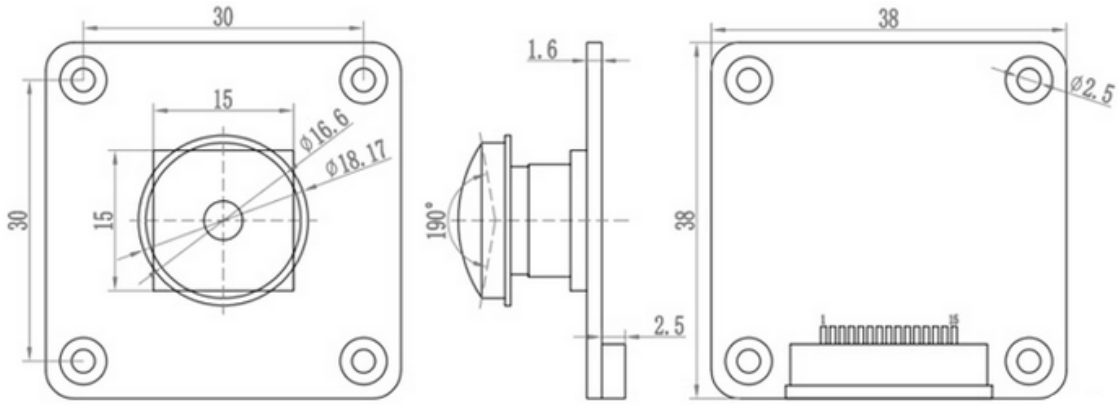
Jedná se o oficiální kamerový modul pro Rapsberry Pi s vyšším rozlišením než jeho předchozí verze. Kamera umožňuje natáčet HD videa a pořizovat fotografie v rozlišení až 3280x2464 px. Vzhledem k úzkému záběru kamery není tento model vhodným adeptem pro tuto práci. [1]

Obrázek 8: Raspberry Pi kamera V2 [1]

2.4.2 Waveshare IMX378-190 Fisheye

Jedná se o 12,3 Mpx kameru pro Raspberry Pi s objektivem typu rybí oko se 190° zorným polem a jasným obrazem, která je připojena k Raspberry Pi pomocí 16žilového plochého

kabelu. Tato kamera byla vybrána jako jediný akceptovatelný model pro tuto práci vzhledem k největšímu možnému širokoúhlému záběru. [2]



Obrázek 9 : schéma kamery pro Raspberry Pi - Waveshare IMX378-190 [2]

2.4.3 Arducam 12Mpx IMX477 kamerový modul

Kamerový senzor nabízí vyšší rozlišení a snímkovací frekvenci než Raspberry Pi kamera V2, při zachování stejné velikosti desky. Kamerový modul je ideální pro využití na dronech, strojové učení, robotiku, rozpoznávání obličeje a detekci defektů. Rozlišení je 12Mpx 4056x3040. Typ senzoru IMX477. Přestože se kamera pyšní lepším rozlišením a snímkovací frekvencí než modul V2, není vhodným adeptem pro tento projekt. [3]

Obrázek 10: Kamerový modul Arducam 12Mpx IMX 477 [3]

2.5 Chytré zařízení

Chytrým zařízením se rozumí malé přenosné zařízení s displejem, na kterém lze instalovat a spustit vytvořenou Unity aplikaci, určenou k obdržení dat odeslaných z Raspberry Pi a zároveň zařízení disponuje i kamerovým objektivem pro pomocné snímání obrazu. Zároveň

by zařízení mělo splňovat podmínky snadné manipulace, a i možnost pohybu s ním. Takovéto podmínky dokonale splňuje mobilní telefon či tablet.

Chytré zařízení bude v projektu sloužit k zobrazování trojdimenzionálních grafických objektů, které se budou zobrazovat nad herními kostkami umístěnými v herním prostoru. Takovéto zařízení obdrží z výpočetního zařízení data v podobě souřadnic, díky kterým následně umístí do herní plochy grafické prvky, které budou zároveň umístěné nad detekovanými kostkami a zobrazeny na displeji telefonu při namíření kamery na definovanou herní plochu. Vzhledem k unikátním identifikátorům kostek, bude přiřazen univerzální grafický objekt kostkám podle jejich identifikačního čísla, které nesou. Takovýchto univerzálních čísel je omezené množství, podle zadání, avšak je možné je během vývoje přidávat, či odebírat dle libosti a potřeb.

3 VIZUÁLNÍ KÓDY

Vizuální kódy jsou nedílnou součástí moderního života a setkáváme se s nimi denně na každém rohu. Využití čárových kódů můžeme spatřit například v obchodech, ve zdravotnictví, nebo v průmyslu. Čárový kód je strojově čitelné značení, které v sobě nese informaci.

Čárové kódy se dále dělí na jednodimenzionální 1D kódy a dvoudimenzionální 2D kódy. Hlavní rozdíl mezi jednodimenzionálními a dvoudimenzionálními kódy je v množství dat, které lze do nich zapsat. Zatímco do běžných čárových kódů vložíte jen malé množství znaků, u 2D kódů jste značně méně limitovaní, díky čemuž jsou složitější. Čím větší množství informací do 2D (například QR) kódu vložíte, tím je kód náročnější na přečtení.

3.1 Jednodimenzionální čárové kódy

Tyto kódy se dělí na dvě hlavní skupiny: souvislé a diskrétní, přičemž charaktery diskrétních čárových kódů začínají čárkou a končí čárkou. Čáry jsou odděleny určitým množstvím mezer mezi čáry. Každý znak lze dekódovat, aniž byste se museli dívat na zbytek čárového kódu. Souvislé čárové kódy obsahují charaktery začínající čárkou a končící mezerou. Jednotlivé znaky nemohou být interpretovány samy o sobě. Šířka konečné mezery jednoho znaku je určena začátkem první mezery dalšího znaku.

Mezi nejpoužívanější druhy jednodimenzionálních kódů se řadí: Code 39 a Code 39 Mod 43, U.P.C. A, U.P.C. E0 a U.P.C. E1, EAN 13 a EAN 8, Code 93, Code 128, Codabar, MSI.

Pro kódování znaků do čárových kódů se používají kódovací tabulky dané pro určitý typ kódů. [4]

Obrázek 11: Struktura jednodimenzionálního čárového kódu [4]

3.1.1 Code 39 a Code 39 Mod 43

Code 39 byl vyvinut v roce 1974. Jedná se o nejčastěji využívanou symboliku u čárových kódů, protože umožňuje zakódovat číslice, písmena a některé interpunkční znaky. Code 39 je typu diskrétní s proměnlivou délkou. Každý znak v tomto kódu obsahuje 5 čar a 4 mezery. Z těchto 9 čar jsou vždy 3 široké a 6 úzkých. Znak start a stop je reprezentován hvězdičkou. Code 39 Mod 43 obsahuje navíc oproti Code 39 i kontrolní znak. [5]

Obrázek 12: Čárový kód Code 39 [6]

3.1.2 EAN-13 a EAN-8

European Article Numbering (EAN) má dvě verze: EAN-13 a EAN-8. Obě varianty jsou numerické a mají pevně stanovenou délku. Dle názvu můžeme poznat kolik číslic oba typy kódují. EAN-13 kóduje 13 číslic a EAN-8 kóduje 8 číslic. Existuje organizace EAN, která provádí správu kódu se sídlem v Belgii. [5]

Z kódu EAN-13 lze například identifikovat zemi původu výrobce nebo způsob užití daného zboží. Méně jsou používány kódy EAN-8, které jsou vyhrazeny a používány pro menší položky, na které je problém umístit 13místný kód, jako jsou třeba sladkosti. [7]

Obrázek 13: Čárový kód EAN-13 [4]

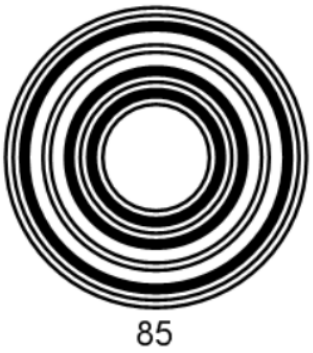
3.1.3 Code 93

Jedná se o souvislou alfanumerickou symboliku proměnné délky, která kóduje všechny znaky ASCII. Jedná se o kód s vysokou hustotou. [5]

Obrázek 14: Čárový kód Code 93 [6]

3.1.4 Kruhový kód

Kruhové kódy byly široce používány pro značení CD/DVD. Struktura těchto kruhových kódů spočívala v sérii vystředěných kružnic typicky založených na symbolice tradičních čárových kódů. Hlavní výhodou kruhového kódu je možnost čtení z kteréhokoliv úhlu. [8]



Obrázek 15: Kruhový kód [8]

3.2 Dvoudimenzionální čárové kódy

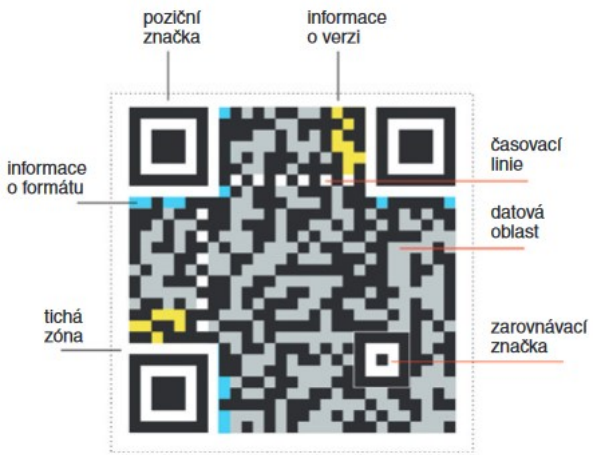
V roce 1987 vyvinul David Allais první dvoudimenzionální čárový kód který nesl označení Code 49. Brzy byly vyvinuty další typy kódů jako například PDF-417, QR, Aztec, Data matrix, Maxi Code a další. Všechny 2D kódy disponují možností automatické korekce chyb, takže dokážou být přečteny i špatně vytištěné nebo znečištěné symboly. V současné době je nejpoužívanější kód tohoto typu QR kód vyvinutý ve firmě Toyota ke značení materiálu při výrobě vozidel. Dvoudimenzionální kódy se dělí do dvou hlavních podskupin: skládané a maticové. Skládané dvoudimenzionální kódy fungují na principu složení dvou a více jednodimenzionálních kódů. Můžou být též označovány jako víceřádkové (Code 49, PDF-417). Maticové kódy jsou zpravidla čtvercového tvaru, sestavené do matice (QR, Aztec, Data Matrix).

3.2.1 QR kód

Jedná se o jeden z nejpoužívanějších dvoudimenzionálních čárových kódů. Název QR kód je odvozen od anglického Quick Response Code. Vyvinut byl v Japonské společnosti Toyota, která neuplatňuje patentová práva a je tudíž volně k dispozici. Jedná se o maticový kód ve tvaru čtverce, který obsahuje poziční značky ve třech vrcholech, informace o formátu, zarovnávací značky umožňující zjistit natočení kódu a časovou linii informující o rozměrech. Zbylá část kódu je vyplněna daty. Nejmenší QR kód má 21x21 modulů, největší QR kód obsahuje 177x177 modulů. QR kód může obsahovat numerické, alfanumerické znaky a binární (8-bit) data, avšak jeho kapacita závisí na rozměrech a počtech modulů. Úroveň obnovy poškozených dat je nastavitelná až na 30% z celkového počtu znaků v symbolu. [4]

Výhody QR kódu oproti jednodimenzionálním kódům: [9]

a) Větší objem uložených dat
b) Menší velikost kódu pro stejný objem dat
c) Možnost skenovat kód z kteréhokoliv úhlu
d) Rozmanitost kódovatelných znaků
e) QR kód umožňuje korekci chyb
f) Možnost ošetření skenovaných dat heslem

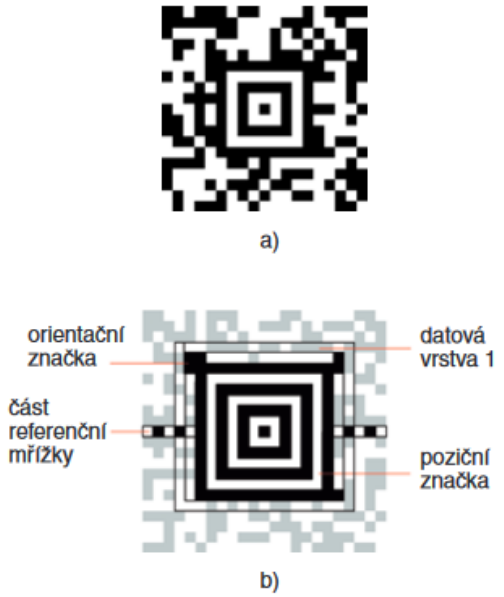


Obrázek 16: QR kód [4]

3.2.2 Aztec kód

Jedná se o jeden z nejmenších 2D kódů čtvercového maticového tvaru, který nepotřebuje okolo sebe volné místo. Tento kód vyvinul v roce 1995 Andrew Longacre a Robert Hussey. Název kódu je odvozen od vzhledu aztécké pyramidy z ptačí perspektivy. Aztec kód

podporuje 255 ASCII charakterů (čísla [0-9], text, binární data). Stejně jako QR kód disponuje korekcí chyb. Aztec kód se používá v kompaktní nebo úplné verzi. Kompaktní verze má dva prstence uvnitř poziční značky a jen čtyři datové vrstvy. Úplná verze má tři prstence uvnitř poziční značky a obsahuje navíc referenční mřížku. Největší Aztec kód může obsahovat až 32 datových vrstev. Oproti QR kódu nepodporuje Kanji znaky. [10]



Obrázek 17: Aztec kód: a) kompaktní verze, b) úplná verze [4]

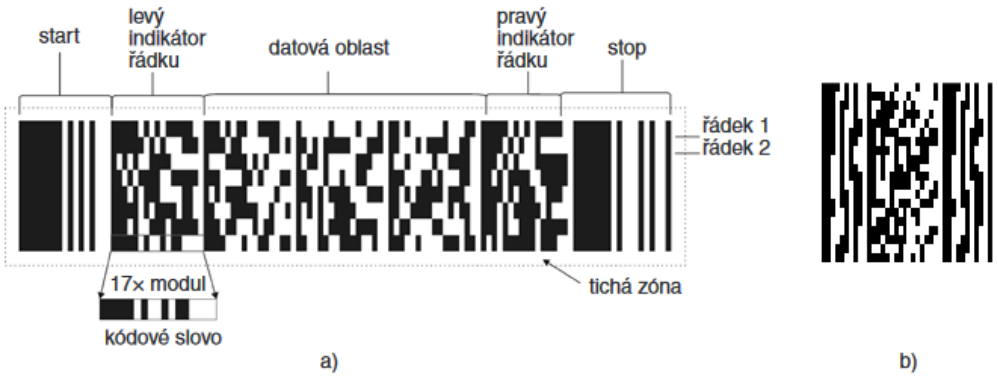
3.2.3 Data Matrix

Jedná se o dvoudimenzionální, velikostně variabilní kód ve tvaru čtvercové matice vyvinut v roce 1989. Datová matice může být vytisknuta černě na bílém podkladu, nebo černě na bílém podkladu. Obsahuje datovou oblast, poziční značky a tichou zónu. Velikost kódů závisí na množství vložených dat. Existují dva typy, které se dělí na základě detekce a korekce chyb (ECC200, ECC000-140) [11]

Obrázek 18: Struktura kódu Data Matrix [4]

3.2.4 PDF-417

Portable Data File 417 je tzv. skládaný čárový kód vyvinut v roce 1990 jako první 2D čárový kód. Jedná se o nejpoužívanější skládaný čárový kód. Výška může být od 3 do 90 řádků. Řádek obsahuje stejně jako u 1D kódů znaky pro start a stop hned vedle indikátoru řádku, který obsahuje údaje jako číslo řádku, celkový počet řádků a sloupců a úroveň detekce chyb. Jeden kódový znak je tvořen čtyřmi čarami a čtyřmi mezerami, dohromady vždy o celkové šířce sedmnáct modulů. Odtud pochází označení PDF 417. [4]



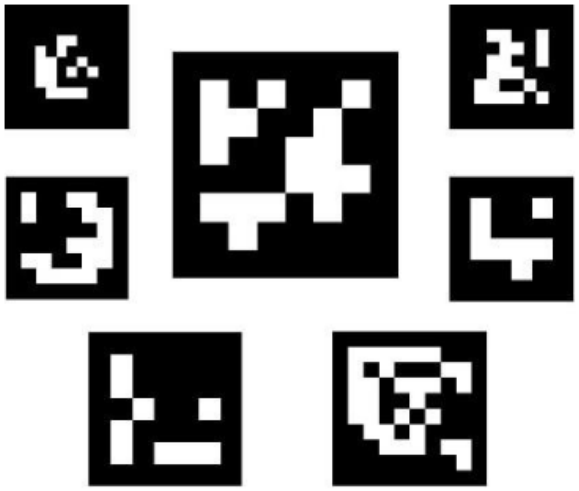
Obrázek 19: Čárový kód PDF-417: a) úplná verze, b) modifikovaná verze Micro PDF 417 [4]

3.2.5 ArUco marker

ArUco marker je jednoduchý, snadno a rychle detekovatelný kód. Obrys této značky je černý, kvůli snadné identifikaci a vnitřek vyplňuje jednoduchá černobílá matice nesoucí informaci. Velikost markeru je daná počtem bitů. Čím větší je počet bitů, tím větší je počet unikátních markerů k detekování. Jako následek vysokého rozlišení matice (např. 8x8) může být problém v čitelnosti. Markery jsou čteny a porovnávány s markery z databáze. Při použití objemově menších databází a nižšího rozlišení je chybová detekce méně pravděpodobná než za použití objemné databáze s vysokým rozlišením markerů.

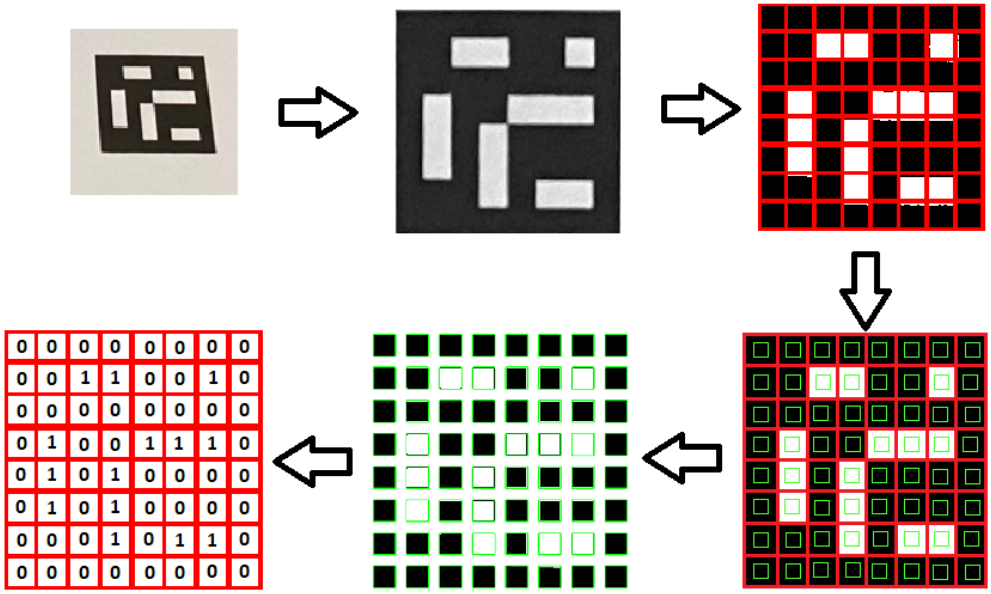
Toto značení je odolné vůči nežádoucím efektům, například vůči nízkému osvětlení, pohybovému rozmazání, odrazům a dalším. Zatímco výše uvedené 2D čárové kódy jsou určené spíše pro přenos informací, ArUco markery jsou využívány hlavně pro určování polohy. Z toho důvodu nesou minimální množství informací v následku čehož jsou dobře

detekovatelné i na větší vzdálenost a můžeme snadno zjistit jejich rozložení, identifikační číslo a natočení.



Obrázek 20: ArUco markery

Jakmile je marker v obraze detekovaný za pomocí kontrastu mezi černými a bílými hrany je marker navzorkovaný do mřížky 6x6, kde každému z třiceti šesti sektorů je přiřazen digitální symbol 0 nebo 1. Výsledek je následně porovnán s obsahem knihovny a získáme tak odpovídající marker.



Obrázek 21: Postup detekce markeru

4 VÝVOJOVÉ PROSTŘEDÍ

Pro vývoj aplikace byl využit jazyk Python. Na zařízení s Windows OS bylo využité Spyder vývojové prostředí a na Raspberry Pi Thonny IDE. Při vývoji na RPI byl využitý VNC software určený pro práci na vzdálené ploše. Vývoj zahrnoval i využití níže zmíněných knihoven, které umožnili práci z různými typy dat a předdefinovanými funkcemi. V podkapitolách následuje stručný popis jednotlivých prostředí a použitých knihoven.

4.1 Python

OpenCV podporuje širokou škálu programovacích jazyků jako C++, Python, Java a další. Bavíme-li se o volbě jazyka pro programování a práci se strojovým viděním, tak byl zvolen právě Python. Důvod této volby je následující. Python je ideální volbou pro rychlé testování použitých knihoven a dle internetových zdrojů je zároveň i nejpoužívanější v oblasti zpracování obrazu, což zajišťuje dostatečné množství návodů. Raspberry Pi OS má již předinstalované Python IDE Thonny a není tak nutné se zabývat instalací vývojového prostředí.

Autorem programovacího jazyka Python je Guido van Rossum. Tento programovácí jazyk se těší velké oblibě hlavně kvůli jednoduchosti a snadné čitelnosti kódu. Je to objektově orientovaný jazyk. Podporuje polymorfismus, přetěžování a vícenásobnou dědičnost a je možné jej zkompilovat na téměř každé platformě. V porovnání s jinými programovacími jazyky umožňuje programátorovi psát krátký kód. Sic je v porovnání s C/C++ Python pomalejší, můžeme skrz něj využívat Python moduly, které při zavolání spustí v pozadí právě C++ kód, což je výhodnější hlavně po rychlostní stránce. Další výhoda je, že programování v Pythonu je pro mnohé jednodušší než v C/C++.

4.2 Virtual Network Computing

Virtual Network Computing, zkráceně VNC je software, který umožňuje uživateli připojení na vzdálené grafické rozhraní přes počítačovou síť. Pracuje na principu klient-server, kde server, v tomto případě Raspberry Pi, generuje grafické rozhraní a komunikuje s klientem ve stejné síti tím, že klient si zobrazí toto grafické rozhraní. Jedná se tudíž o vzdálený přístup k ploše jiného počítače.

Při využívání VNC je nutné mít nainstalovaný VNC Server program na serveru a u klienta VNC Viewer. V dnešní době je již VNC součástí Raspberry Pi operačního systému a není nutné jej tak instalovat dodatečně. Základní verze VNC Viewer je k dispozici zdarma.

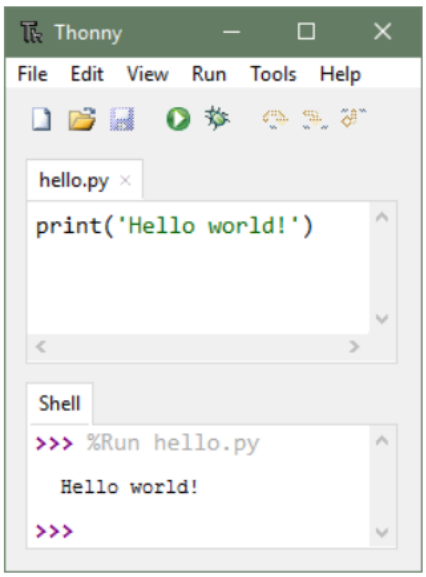
4.3 Spyder

Spyder je vědecké vývojové prostředí napsané v jazyce Python pro Python navržené vědci, inženýry a datovými analytiky. Spyder obsahuje nespočet specifických funkcí, obsahuje konzoli pro získání například aktuálního výsledku kódu, textový editor a mnoho dalších. Spyder byl využitý pro projekt při práci na Windows zařízení, zatímco při práci na Raspberry Pi zařízení bylo využíváno prostředí Thonny.

Spyder uživateli poskytuje vícejazyčný panel editoru s mnoha funkcemi. Editor nabízí automatické dokončování, analýzu v reálném čase, zvýraznění syntaxe a mnoho dalšího. Python konzole zas uživateli umožní spouštět příkazy, skripty a pracovat s daty. Dále uživateli průzkumník proměnných umožňuje interaktivně spravovat a procházet objekty vygenerované kódem. Plot panel zase poskytuje statické obrázky a obrázky vytvořené kódem, jako jsou například grafy. Samozřejmostí je i možnost provádět debugging v editoru, který ulehčuje ladění programu. [12]

4.4 Thonny

Thonny (IDE) je společně s Geany předinstalovaný editor v Raspberry Pi OS, který byl využitý pro psaní praktické části bakalářské práce. Jedná se o jednoduché a intuitivní vývojové prostředí, do kterého stačí napsat program v jazyce Python, stisknout tlačítko „Run“ a výstup spuštěného programu se objeví v „output“ okně ve spodní části prostředí.



Obrázek 22: Ukázka „Hello World“ programu v Thonny IDE

4.5 Knihovny

Podkapitoly níže popisují použité knihovny v jazyce Python, které byly využity za účelem vypracování praktické části bakalářské práce. Knihovny vyžadují před jejich použitím instalaci, kterou je možné provézt pomocí příkazu „pip install“ a import každé z instalovaných knihoven na začátku Python kódu příkazem „import“. Po provedení instalace a importu knihovny do kódu, je možné knihovnu využívat spolu s jejími funkcemi.

4.5.1 OpenCV

Jedná se o open source knihovnu určenou pro práci s počítačovým viděním a strojovým učením. Knihovna obsahuje více než 2500 algoritmů určených pro detekci obličejů, objektů, sledování pohybu, získání 3D bodů za použití stereo kamer, spojení obrázků pro zlepšení rozlišení, sledování pohybu očí a mnoho dalších. OpenCV je napsáno v jazyce C++ a je rozděleno na základní moduly, např. core, imgproc, imgcodecs, videoio, highgui, video, calib3d, features2d a další. [13]

4.5.2 Math

Standartní modul „math“ zprostředkovává přístup k matematickým funkcím. Pro používání matematických funkcí z tohoto modulu, je zapotřebí modul importovat do svého projektu vložením „import math“ na začátek skriptu. Tento modul nepodporuje komplexní datové typy. Mezi základní funkce se řadí například ceil(x), floor(x), faktorial(x), exp(x) a mnoho dalších. [14]

4.5.3 Socket

Socket modul umožňuje přístup k BSD socket aplikačnímu programovému rozhraní, kde můžeme provádět volání základních funkcí a můžeme tak přidat internetovou komunikaci do našeho projektu. Mezi základní funkce a metody API se řadí socket(), bind(), listen(), accept(), connect(), send(), recv(), close(). [15]

4.5.4 Numpy

Numpy je open source knihovna, která se zaměřuje na numerického počítání v jazyce Python. Knihovna byla vytvořen v roce 2005. Zprostředkovává uživateli multidimenzionální pole, maskování pole a matice, funkce pro rychlé operace v poli, včetně matematických, logických a tvarových manipulací. [16]

4.5.5 Subprocess

Subprocess modul je nástroj pro spuštění programu nebo příkazu z již spuštěného Python kódu. Může být využit pro spuštění nového programu, kterému zprostředkuje data a získá zpět data, která jsou nějakým způsobem zpracované. Například pomocí subprocess může uživatel používat příkazy jako ls, ping, call, a získat jejich výstup. [17]

4.5.6 Imutils

Imutils modul umožňuje uživateli provádět základní funkce pro zpracování obrazu, jako je například rotace, změna velikosti, třídění obrysů, detekce hran a další. Autorem této knihovny je Adrian Rosebrock.

4.5.7 Picamera2

Picamera2 je Python knihovna poskytující pohodlný přístup ke kamerovému systému Raspberry Pi, která je určena pro kamery připojené plochým kabelem přímo na Raspberry Pi kamerový konektor a není určena pro jiné typy kamer. Přesto existuje limitovaná podpora pro USB kamery.

Picamera2 je postavena na open source projektu libcamera, který umožňuje komplexní podporu kamerového systému v Linuxu. Picamera2 je náhrada za legacy PiCamera Python knihovnu. [18]

4.5.8 Matplotlib

Matplotlib je knihovna určená pro vykreslování grafů v jazyce Python, která slouží jako nástroj pro vizualizaci. Knihovna Matplotlib byla vyvinuta Johnem D. Hunterem a je napsaná převážně v jazyce Python. Jedná se o open source knihovnu, kterou mohou uživatelé využívat zdarma bez poplatků. Knihovna obsahuje funkce jako například plot(), která je určená pro vykreslení bodů v diagramu. Defaultně tato funkce vykresluje přímku z bodu do bodu.

4.5.9 Pyzbar

Pyzbar je knihovna, která je určená pro čtení jednodimenzionálních kódů z obrazu, která spolupracuje společně s knihovnou OpenCV, Numpy atd.

4.5.10 Sys

Tento sys modul poskytuje uživateli přístup k proměnným používaných interpreterem. Například funkce „sys.argv“ vypíše list vstupních parametrů, které byly zadané při spuštění python skriptu.

4.5.11 Time

Time modul poskytuje uživateli funkce umožňující práci s časem. Modul obsahuje funkce pro získání aktuálního času, formátování času, případně uspání běžícího procesu po určitou dobu. Funkce lze volat názvy „ctime()“, „localtime()“, nebo například „sleep()“.

4.5.12 Nmap

Nmap je bezplatná knihovna poskytující možnost skenování síťových portů. Knihovna je často využívána například systémovými administrátory. V projektu je knihovna využívána pro testování a získání aktuálně připojených zařízení do Wi-Fi sítě. Knihovna obsahuje funkce, jako jsou například „scan()“, nebo „all_hosts()“.

5 KOMUNIKACE

5.1 Bluetooth

Bluetooth slouží pro komunikaci mezi zařízeními na blízkou vzdálenost v řádu desítek metrů, oproti komunikaci přes internet, kde vzdálenost nehraje roli. Princip komunikace je u obou variant stejný. Bluetooth byl vyvinut od základů, nezávisle na protokolech Ethernet a TCP/IP, přestože pracují v zásadě na stejném principu komunikace dvou zařízení.

Každý vyrobený Bluetooth čip má přiřazenou unikátní identifikační 48-bit adresu, která je velmi podobná s MAC adresou a je označována jako Bluetooth adresa, nebo adresa zařízení. Tyto adresy jsou čipům přiřazeny již při výrobě a jsou statické po celou dobu životnosti čipu, přičemž pro komunikaci mezi zařízeními přes Bluetooth je nutné znát právě takovouto MAC adresu zařízení. Tyto adresy jsou spravovány organizací IEEE (Institute of Electrical and Electronics Engineers).

Pro člověka může být obtížnější orientovat se pomocí 48-bit adres a tak, můžeme Bluetooth zařízení přiřadit název, který bude sloužit k identifikaci zařízení. V mnoha případech existuje možnost, kdy si uživatel může název zařízení sám modifikovat. Ovšem takový uživatel by měl brát zřetel na to, aby bylo zařízení i nadále identifikovatelné a mělo by mu být přiřazeno unikátní jméno.

Po připojení dvou zařízení je nutné specifikovat, který protokol bude využit pro přenos dat. Níže je uvedena stručná specifikace tří hlavních Bluetooth protokolů. [19]

5.1.1 RFCOMM

Bluetooth RFCOMM je jednoduchý transportní protokol vytvořený nad protokolem L2CAP, který se funkčně podobá TCP. V praxi se u aplikací využívajících TCP očekává že budou využívat point-to-point připojení, přes které se budou data odesílat. V případě že data nebudou přijata na druhém zařízení v určitém časovém úseku, připojení bude ukončeno. A stejný princip využívá i Bluetooth RFCOMM protokol pro komunikaci mezi zařízeními.

Zatímco TCP podporuje až 65 535 otevřených portů na jednom zařízení, RFCOMM podporuje pouze 30 portů. [19]

5.1.2 L2CAP

Bluetooth L2CAP (layer common access protocol) je podobný internetovému UDP. UDP se využívá ze předpokladu, že není požadováno spolehlivé doručení všech zaslaných packetů. L2CAP ve výchozím nastavení poskytuje individuální odesílání datagramů pevně stanovené délky. Tento protokol je poměrně přizpůsobivý a lze jej nakonfigurovat pro různé úrovně spolehlivosti.

Přenosový protokol, na kterém je L2CAP postaven využívá princip, kde jsou nepotvrzené packety znovu vysílány. Existují zde 3 zásady aplikace:

1.) Nikdy packety znovu neodesílat.
2.) Znovu odeslat packet do úspěchu, nebo selhání komunikace (defaultní varianta).
3.) Zahodit packet nebylo-li potvrzeno přijetí po určité době a přejít na další ve frontě.

Nikdy znovu neodesílat a zahodit packet po určité době je označováno jako snaha o nejlepší úsilí při komunikaci. Znovu odesílání packetu do úspěchu nebo úplného selhání připojení je označováno za spolehlivou komunikaci. Je nutné brát v úvahu to, že pokud bude mít zařízení otevřeno více Bluetooth komunikačních kanálů, je zde pravděpodobnost, že všechny ostatní komunikace budou negativně ovlivněny. Tento problém se ale netýká zařízení, kde existuje pouze jedno připojení. [19]

5.1.3 OBEX

OBEX (Object Exchange) je poslední z uvedených protokolů a je jeden z mnoha dalších protokolů, které zde nejsou zmíněny. Jedná se o hojně využívaný protokol pro výměnu informací mezi Bluetooth zařízeními. Namísto přenosu kolekce bajtů OBEX přenáší objekt, který je rozeznatelný operačním systémem, jako například obrázek, zvukový klip, vyzváněcí tón, a další. [19]

5.2 Internetové transportní protokoly

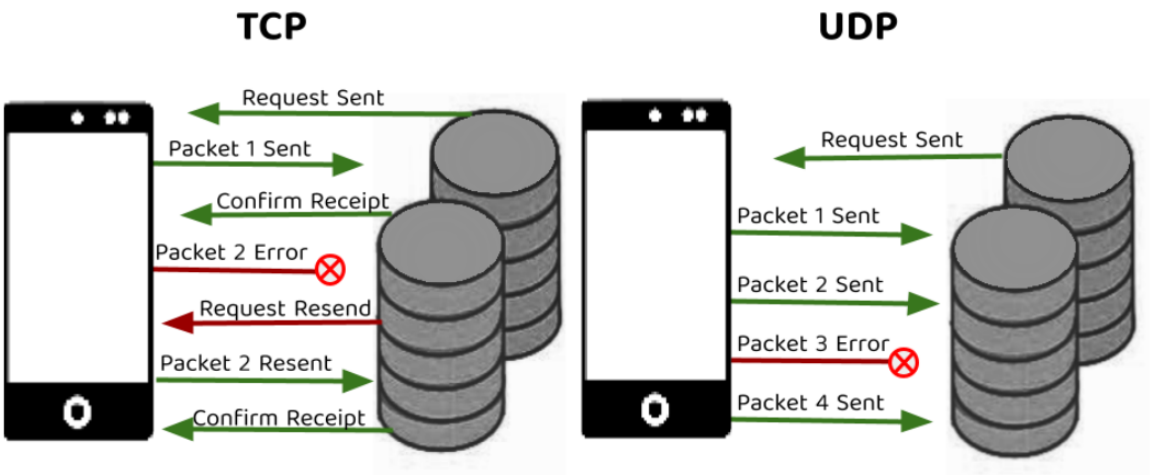
Internetové transportní protokoly se nachází v transportní vrstvě modelu síťové architektury OSI a poskytují spolehlivý přenos dat s požadovanou kvalitou. Mezi takové protokoly se řadí UDP nebo TCP. Pro projekt, vzhledem k požadavku streamování dat jedním směrem, kde nehraje velkou roli ztráta packetů je nejvýhodnější protokol UDP.

5.2.1 TCP

TCP (Transmission Control Protoco) poskytuje kontrolovaný a spolehlivý transportní servis. Jedná se o protokol transportní vrstvy. U TCP protokolu je spojení stanoveno před započnutím odesílání dat a je zaručený příjem všech odeslaných paketů. Data jsou odesílána bez duplikací a jsou přijímána ve stejném pořadí v jakém byly odeslány. Ve chvíli, kdy je packet ztracen je znovu odeslán k cílovému uživateli. Práce na protokolu TCP/IP začali v roce 1973 a je rychlostně pomalejší než UDP, protože dochází ke kontrole přijímaných packetů. TCP protokol je například ideální při načítání webových stránek nebo stahování souborů, kdy je špatné pořadí a ztráta přijímaných packetů nežádoucí. [20]

5.2.2 UDP

UDP (User Datagram Protocol) poskytuje lehký a nespolehlivý transportní servis. Jedná se stejně jako u TCP o protokol transportní vrstvy, který uživatel využije pro vytvoření spojení s jiným uživatelem, nebo serverem. Služba neposkytuje žádné záruky, to znamená, že data mohou být ztracena, duplikována nebo mohou dojít v jiném pořadí, než byly odeslány. Například pokud server posílá data klientovi, tak pošle všechny data klientovi a dále se nestará o to, jestli došlo ke ztrátám. Tento způsob není ideální například pro stahování souborů, protože může vést k chybě, jejímž následkem by bylo neúplné stáhnutí souboru. Dalším špatným příkladem použití UDP protokolu může být načítání webových stránek. Na druhou stranu UDP protokol najde využití například u streamování nebo videohovoru kde uživatel nebude postrádat ztracené pixely. UDP je tak rychlejší než TCP jelikož nemusí čekat na odezvu druhé strany. [20]



Obrázek 23: Princip TCP a UDP transportních protokolů

6 ZPRACOVÁNÍ DAT

V projektu dochází ke zpracování dat se souřadnicemi, které slouží k následnému vykreslování grafického rozhraní na chytrém zařízení. Ke zpracování dat je využíváno aplikace Unity, ve kterém je využíván programovací jazyk C#. Data jsou odesílána z RPI zařízení na telefon, či tablet v podobě již zmíněných souřadnic přes UDP, přičemž RPI k odesílání využívá Python kód, který čeká na připojení telefonu do sítě a zapnutí Unity aplikace. Zatímco chytré zařízení telefon, nebo tablet využijí zmíněný C# kód. Vytvořený C# kód bude mít za úkol odeslat RPI zařízení zprávu, že je aplikace spuštěná a že je připravena přijímat data v podobě seznamu, obsahující vždy list s unikátním číslem a souřadnicemi na ose x a y.

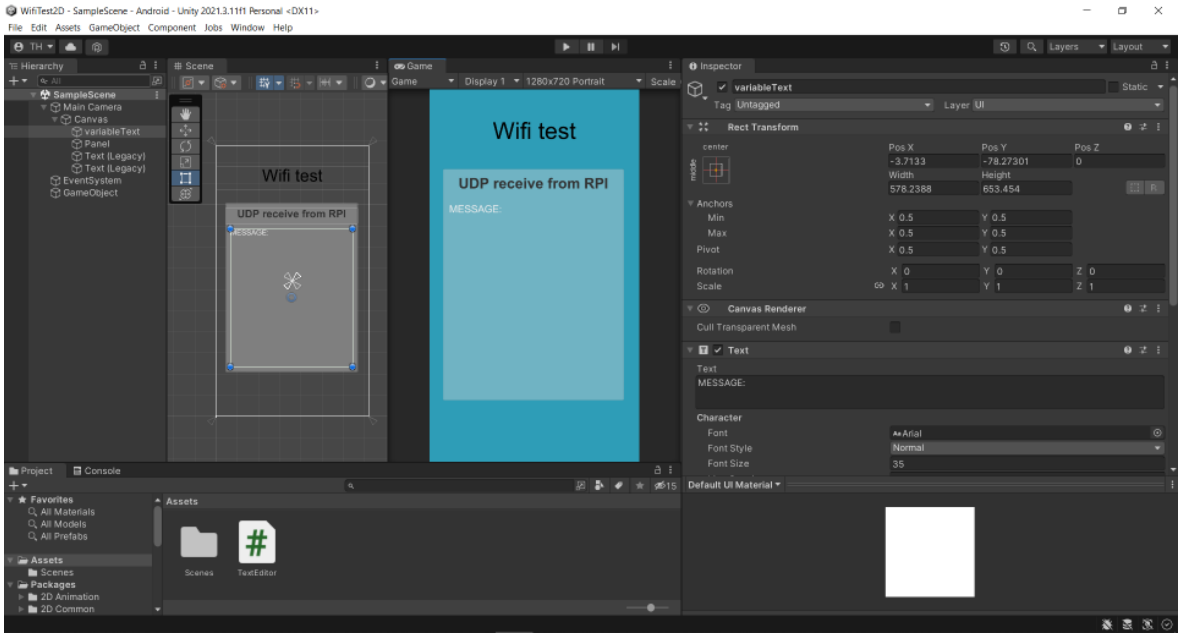
6.1 Unity

Unity je engine umožňující rozmanitou škálu úkonů a je zvláště využívaný širokou veřejností při vývoji her pro počítače, konzole, mobilní zařízení, webové aplikace a v dnešní době je možné vyvinout hru i na oblíbené VR platformy, nebo Smart TV. Pomocí tohoto enginu lze vytvořit 2D nebo 3D hry. Jedná se o komplexní vývojové prostředí, které obsahuje pomůcky pro práci s objekty, zvukem, světlem, textury a mnoho dalších pomůcek.

Součástí Unity prostředí je i vývojářská konzole napomáhající programátorovi při vývoji aplikace a jejího testování s výpisem chybových hlášek a zachycených výjimek. Testování aplikace je možné přímo v Unity prostředí. Není tedy příliš nutné vytvářet ve vývojové fázi spustitelné aplikace a můžeme tak aplikaci spustit přímo v tomto prostředí. Takovéto spuštění aplikace ve vývojovém prostředí umožňuje uživateli libovolně pozastavovat scénu, manipulovat s ní a s objekty v ní. Umožňuje například měnit rozlišení obrazovky a najít taky potřebné rozlišení pro aplikaci i za běhu.

Následné vyexportování spustitelného souboru pro instalaci aplikace na danou platformu je velice jednoduché a zahrnuje základní konfiguraci, kterou vás provede manuál k tomu určený umístěný v Unity dokumentaci.

Vyexportovaný soubor stačí už jen přenést, například pomocí USB kabelu nebo flash disku na zařízení, kam budeme naši aplikaci instalovat. Na takovém zařízení spustíme přenesený soubor a provedeme jeho instalaci. Po úspěšné instalaci by měla aplikace běžet a uživatel si tak může užívat vlastnoručně vytvořené hry.



Obrázek 24: Ukázka Unity vývojového prostředí

6.2 Visual Studio

Visual studio je nejznámější vývojové prostředí vyvinuté společností Microsoft, umožňující vytvářet širokou škálu různých typů aplikací. Prostředí je velmi používané vývojáři využívající Microsoft produkty po celém světě.

Visual Studio, uživateli zkráceně označováno jako VS podporuje hned několik platforem a programovacích jazyků, bez nutnosti instalace dodatečných balíčků, pluginů, rozšířeních a komponent.

Tento produkt Microsoftu byl vyvinut pro vývojové účely. V roce 1997 a později Microsoft vyvinul několik programovacích jazyků pro operační systém Windows a k programovacím jazykům vytvořil IDE (Visual Studio) pro jejich podporu.

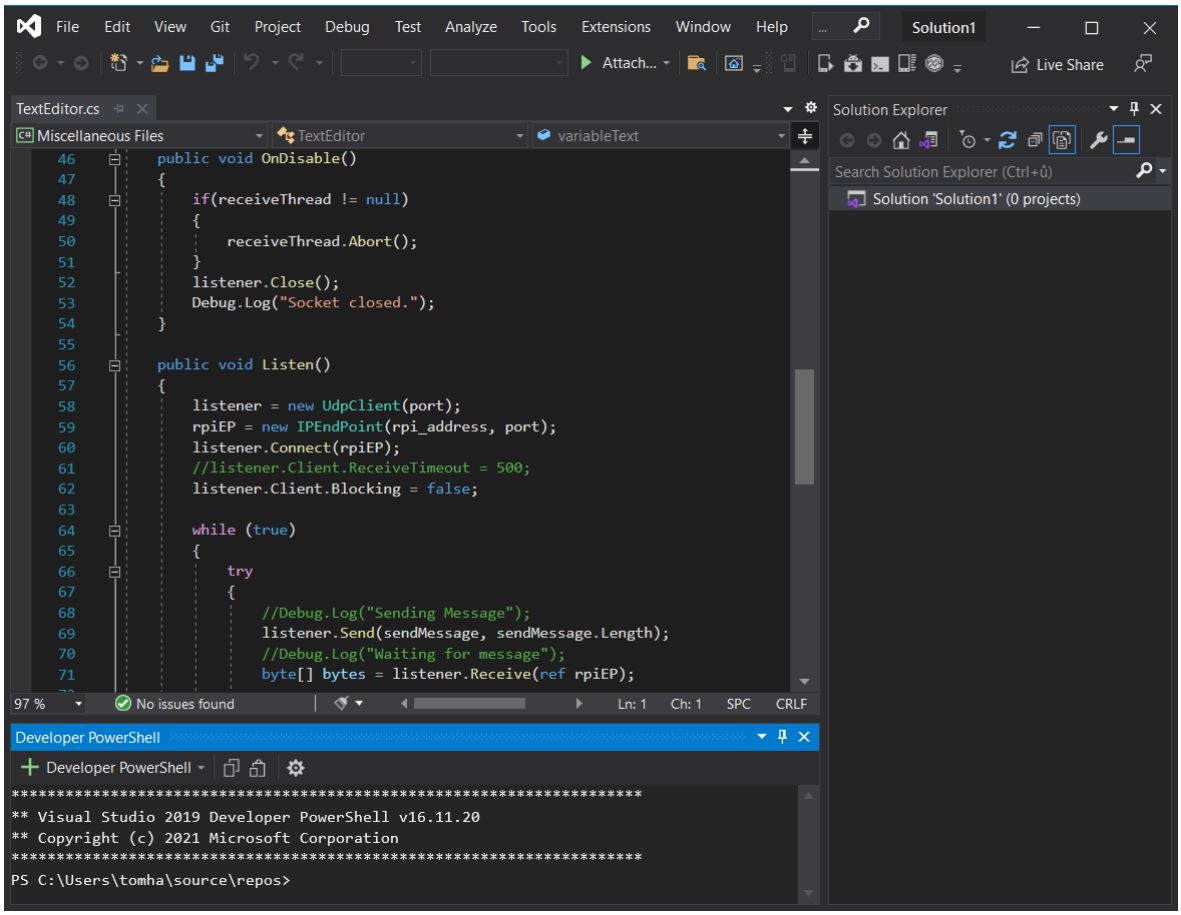
Visual studio existuje ve více verzích, kde Visual Studio Community je zdarma k dispozici pro individuální vývojáře, Visual Studio Professional a Visual Studio Enterprise jsou již k zakoupení pro společnosti, přičemž Visual Studio Professional verze je mírně omezená v podporovaných funkcích oproti Enterprise verzi.

Vývojové prostředí je dále členěno podle toho, na jakém OS jej chcete využívat. Visual Studio je určeno pro Windows platformu, Visual Studio for Mac je určeno pro Apple produkty a třetí Visual Studio Code je možné spustit na Windows, Apple a Linux zařízeních. [21]

6.3 C#

C# je objektově orientovaný vyšší programovací jazyk vytvořený firmou Microsoft. Syntaxe jazyka C# je velmi podobná syntaxi C++ nebo Javy, ovšem podobnosti jsou jen povrchové a C# se oproti těmto jazykům v mnoha ohledech liší. C# je čistě objektový jazyk. Na rozdíl od C++ pracujeme v C# pouze s dynamickými objekty (tzn. vytváříme je pomocí „new“). O mazání vytvořených objektů se programátor starat nemusí, protože zde existuje garbage collector, který se stará o odstranění nepoužívaných objektů.

Zdrojový kód programu napsaný v jazyce C# se nepřeloží přímo do strojového kódu, ale do jazyka CIL (Common Intermediate Language). Z CIL je následně kód převeden do strojového kódu v okamžiku spuštění programu. Na počítači musí být též nainstalováno prostředí .NET, které kromě překladačů obsahuje i další komponenty potřebné pro běh programů. [22]



Obrázek 25: Ukázka C# kódu ve vývojovém prostředí Visual Studio 2019

II. PRAKTICKÁ ČÁST

7 POPIS

Praktickou část bakalářské práce lze rozdělit na čtyři části, kde první část se zabývá přípravou hardware prvků projektu, výběrem vyhovujícího kamerového modulu a jeho kalibrací. V druhé části dochází k testování vybraných vizuálních kódů přes výřez obrazu až po výběr jednoho vyhovujícího kódu spolu se získáním souřadnic ze zvoleného univerzálního značení. Třetí část je zaměřená na komunikaci dvou zařízení přes Wi-Fi a Bluetooth, kde dochází k testování jednotlivých typů komunikace a k výběru nejideálnějšího způsobu. Poslední čtvrtá část řeší zpracování souřadnic získaných z Raspberry Pi v aplikaci Unity a jejich výpis na displej telefonu, spolu s automatizací Raspberry Pi.

7.1 Aplikace

Cílem projektu je vytvoření herní desky a aplikace pro chytré zařízení. Herní deskou se rozumí vymezený hrací prostor, do kterého uživatel umístí jednu či více herních kostech vytvořených pomocí 3D tisku, které na sobě mají unikátní identifikátor. Unikátních identifikátorů (vizuálních kódů) existuje osm, a tudíž existuje osm typů univerzálních kostek. Spuštěná aplikace na chytrém zařízení je následně schopná pomocí namířené kamery na herní desku nad jednotlivými typy kostek vykreslit 3D grafický tvar a zobrazit ho na displeji zařízení.

Tato samotná aplikace ale není schopná detekce unikátních identifikátorů, a tak za pomocí prostředníka, což je Raspberry Pi s kamerou, ve kterém dochází k získání souřadnic kostek, tyto souřadnice obdrží a využije k vykreslení 3D grafiky na displeji.

Herní plocha je vymezená čtyřmi ArUco markery a je čtvercového tvaru. Při zpracování obrazu v Raspberry Pi je plocha rozdělena jako matice 8x8 na 64 částí. Podle toho, v jaké části se herní kostka aktuálně nachází dochází k získání souřadnic. Například pokud se herní kostka nachází v levém horním rohu, obdrží chytré zařízení data, která informují, že kostka s identifikačním číslem „1“ leží na souřadnicích v ose x:1 a v ose y:1. Za předpokladu že bude kostka umístěna do pravého dolního rohu, obdrží zařízení informaci o kostce opět s identifikačním číslem jedna, která leží nyní na souřadnicích v ose x:8 a v ose y:8.

Unity aplikace nainstalovaná v telefonním zařízení potřebuje pro grafické vykreslení pouze jeden bod, který pro ni definuje herní plochu. Přítomnost dalších tří bodů, definující hranici herní plochy není nutná, protože v Unity aplikaci lze využít funkce, sloužící pro orientaci v 3D reálném prostoru, které si vytvořenou herní plochu dokážou definovat pouze za pomocí

jednoho bodu a pozici této herní plochu si budou udržovat i při namíření kamery jiným směrem. Od tohoto bodu, kterým může být například jeden z ArUco markerů, definujících herní plochu, bude aplikace vykreslovat podle získaných souřadnic grafické objekty na displeji.

Těmito grafickými objekty může být cokoliv. Může se například jednat o grafické figurky různých tvarů, které je možné posouvat po hrací ploše, nebo lze pomocí spojení určitých typů herních kostek vytvořit různé tvary, které se nad použitými kostkami zobrazí. Užitečnější využití pro projekt může být například za předpokladu, kdy existuje určité množství univerzálních kostek a každá z těchto kostek bude díky svému ID detekována a bude jí přiřazen podle předdefinování určitý typ grafického vykreslení krajiny. Například herní kostka, nesoucí ID s číslem 1, bude spojena s grafickým vykreslením vody. Dále herní kostka s ID číslem 2, bude spojena s grafickým vykreslením země (písku), atd. Umístěním takovýchto dvou kostek vedle sebe na herní plochu by mělo za výsledek vykreslení grafiky, zobrazující písek spojený s vodou, tudíž by došlo k vytvoření pláže. Takovýmto způsobem by docházelo k vytvoření celé mapy, kterou si může uživatel přestavovat a utvářet dle své libosti.

8 PŘÍPRAVA PROSTŘEDÍ A HARDWARE

Prvním krokem praktické části je připravení hardware architektury, která se skládá z prvků popsaných v teoretické části bakalářské práce a instalace všeho potřebného software pro bezproblémový provoz.

8.1 Příprava Raspberry Pi

Pro Raspberry Pi je vyvíjen operační systém Raspbian optimalizovaný pro jeho hardware. Tento operační systém je odvozen z operačního systém Debian a je potřebný pro následující práci na Raspberry Pi.

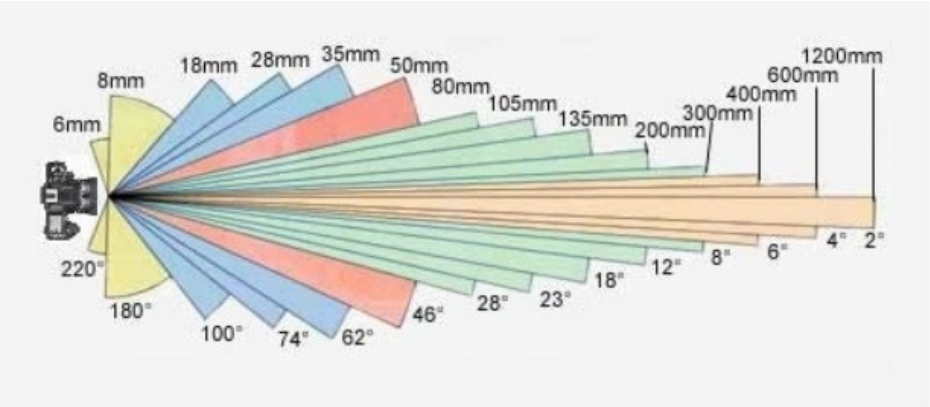
Do počítače nebo notebooku je vložena SD karta o velikosti 32 GB, která je součástí balíčku Raspberry Pi a z oficiálních webových stránek je stažen program „Raspberry Pi Imager". V programu je po spuštění zvolen operační systém s desktopovým rozhraním a uložiště typu SD karta na kterou je operační systém nahraný. Po nahrání operačního systému vložíme SD kartu do Raspberry Pi slotu, načež je možné tento malý počítač spustit.

K Raspberry Pi je připojen kamerový modul pomocí 16žilového plochého kabelu, napájení přes USB typu C a síťový kabel, který spojuje Raspberry Pi a notebook. Notebook má díky programu VNC Viewer přístup ke grafickému rozhraní Raspberry Pi, které lze nyní ovládat pomocí klávesnice a myši z notebooku. Nyní je tento malý počítač připravený k práci.

Po základní konfiguraci je čas nainstalovat potřebné knihovny, jako OpenCV, numpy, socket, math, picamera2 a další, přes terminál pomocí příkazu „pip install", nebo „apt install". Jakmile jsou knihovny nainstalovány je možné psát do předinstalovaného Thonny IDE první kód, který bude sloužit pro zobrazení obrazu z kamery.

8.2 Výběr kamery

Hlavním faktorem při výběru kamery je ohnisková vzdálenost, protože platí, čím menší ohnisková vzdálenost kamery, tím větší má kamera zorný úhel. Na široký zorný úhel kamery je kladen velký důraz, z důvodu zachování krátké vzdálenosti kamery od hrací plochy, aniž by došlo k nežádoucímu ořezání obrazu.



Obrázek 26: Jak se liší zorný úhel na základě ohniskové vzdálenosti

Zobrazení obrazu z kamery vyžaduje pár řádků kódu a základní konfiguraci Raspberry Pi. Do terminálu je vložen příkaz „sudo raspi-config", který otevře konfigurační prostředí, kde povolíme vstup kamery. Po povolení kamery otestujeme příkazem „vcgencmd get_camera", zda je kamera řádně detekována.

```
$ vcgencmd get_camera
supported=1 detected=1, libcamera interfaces=0
```

Obrázek 27: Výstup příkazu "vcgencmd get_camera"

Úspěšně detekovanou kameru je nyní možné využít pro zobrazení snímaného obrazu. Níže uvedený Python kód obsahuje implementaci knihovny OpenCV a zachytává obraz z defaultní kamery na Raspberry Pi, přičemž tento obraz vykresluje příkazem „cv2.imshow" uživateli. Uživatel může spuštěný Python kód ukončit stisknutím klávesy Q nebo Esc.

```
import cv2

source = cv2.VideoCapture(0)
if not source.isOpened():
    print("Cannot open camera")
    exit()

while True:
    has_frame, frame = source.read()
    if not has_frame:
        print("Camera not connected!")
        break

    cv2.imshow("Camera view", frame)

    key = cv2.waitKey(1)
    if key == ord('Q') or key == ord('q') or key == 27:
        break

source.release()
cv2.destroyAllWindows()
```

Zatímco obraz z kamerového modulu V2 lze zpracovat pouze za pomocí OpenCV knihovny a kódu zobrazeného výše, u dalších dvou modulů je zapotřebí využít Python knihovnu Picamera2. Níže uvedený kód slouží pro zobrazení obrazu z kamer Waveshare IMX378-190 Fisheye a Arducam 12Mpx IMX477. Kromě samotného Python kódu je nutné upravit soubor „/boot/config.txt“ podle snímače právě využívané kamery a vložit do něj buďto „dtoverlay=imx477“ nebo „dtoverlay=imx378“.

```
import cv2
from picamera2 import Picamera2

picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration\
                 (main={"format": 'RGB888', "size": (1280, 960)}))
picam2.start()

while True:
    frame = picam2.capture_array()
    if frame is None:
        break
    cv2.imshow("Camera", frame)

    key = cv2.waitKey(1)
    if key == ord('Q') or key == ord('q') or key == 27:
        break

cv2.destroyAllWindows()
```

Po otestování je kamera Waveshare IMX378-190 Fisheye vybrána jako nejvíce vyhovující z důvodu nejnižší ohniskové vzdálenosti, tudíž s největším zorným úhlem.

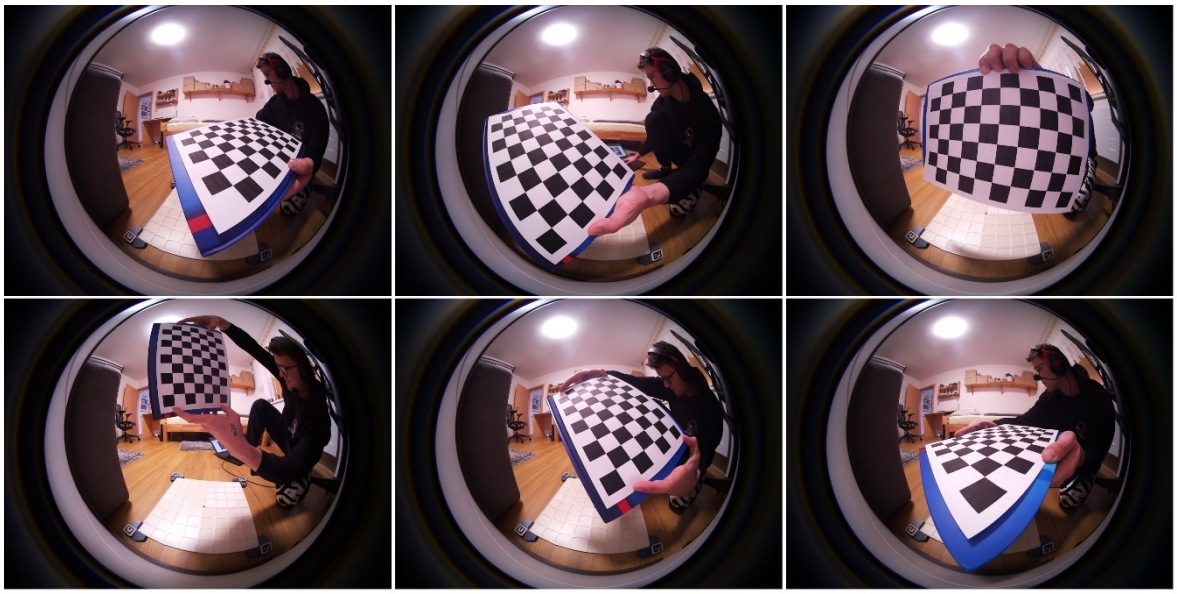
Obrázek 28: Zobrazení pohledu kamery Waveshark IMX378-190 Fisheye

8.3 Kalibrace kamery

Vzhledem k výběru kamerového modulu je nutné provést kalibraci obrazu, která předejde nežádoucím deformacím, které jsou součástí kamer typu rybí oko a bude možné z obrazu bezproblémově detekovat ArUco markery.

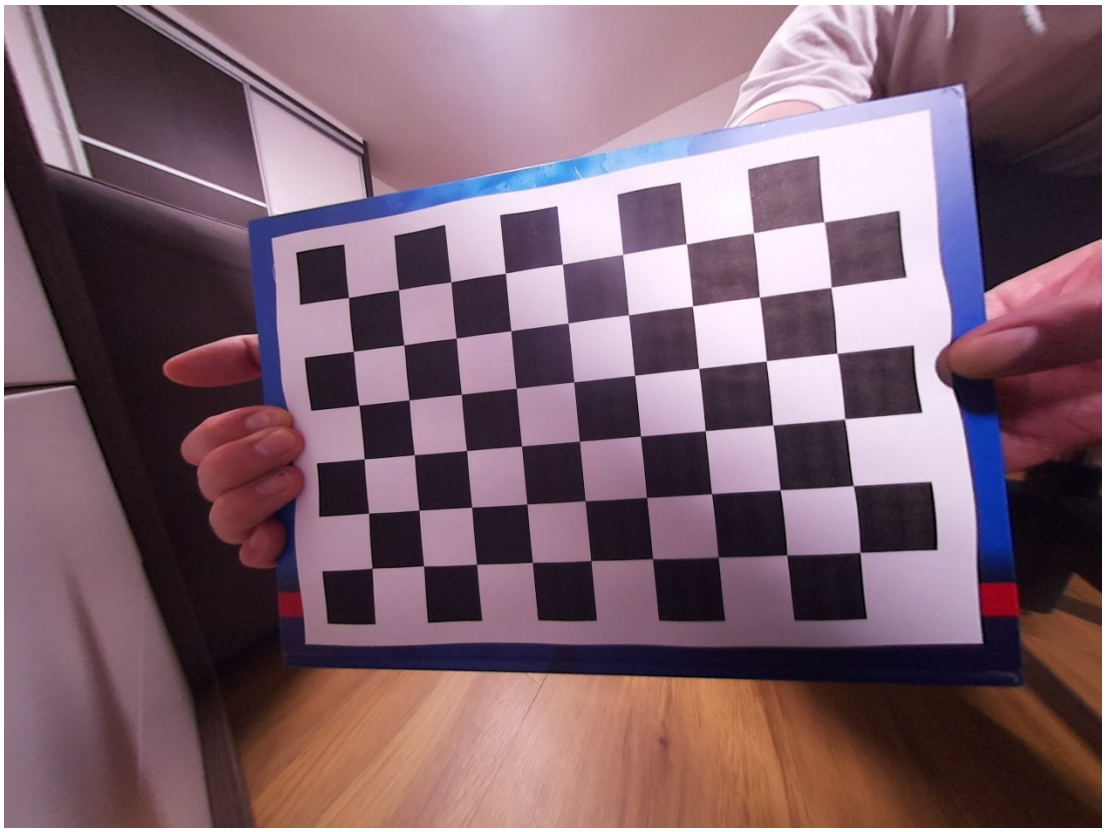
Taková kalibrace je provedena pomocí několika snímků šachovnicového vzoru umístěného před kameru v různých polohách. Snímky jsou díky kalibračnímu Python kódu *calibrate.py* zmapovány a výstupem takového kódu jsou tři parametry DIM, K a D. Tři výstupní parametry jsou dále využity ke kalibraci obrazu v Python funkci zobrazené níže, kde vstupním parametrem funkce je zdeformovaný obraz z kamery a výstupním parametrem je obraz již zpracovaný a nezdeformovaný.

```
DIM=(1280, 960)
K=np.array([[290.98819482567455, 0.0, 634.223974611061],\
            [0.0, 292.864973219883, 481.3538809773127], [0.0, 0.0, 1.0]])
D=np.array([[0.11700104482415771], [-0.13856085817812192], \
            [0.07756993368964876], [-0.0185412487827757]])

def undistort(img):
    h,w = img.shape[:2]
    map1, map2 = cv2.fisheye.initUndistortRectifyMap\
                 (K, D, np.eye(3), K, DIM, cv2.CV_16SC2)
    undistorted_img = cv2.remap(img, map1, map2,\
                                interpolation=cv2.INTER_LINEAR,\
                                borderMode=cv2.BORDER_CONSTANT)
    return undistorted_img
```



Obrázek 29: Snímky z kamery se šachovnicovým vzorem pro kalibraci obrazu

Obrázek 30: Snímek před kalibrací obrazu



Obrázek 31: Snímek po kalibraci obrazu

9 OVEŘENÍ SPOLEHLIVOSTI DETEKCE VIZUÁLNÍCH KÓDŮ

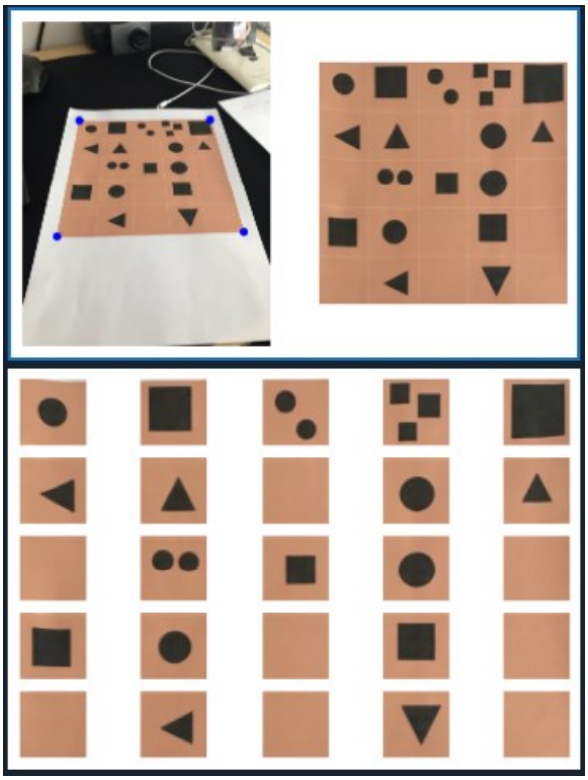
Kalibrace obrazu je hotová a nyní je možné obraz využívat k detekci různých, již nezkreslených objektů. Před detekcí herních kostek je ale nejdříve nutné spolehlivě detekovat herní plochu jako výřez obrazu, ve kterém bude docházet k detekci objektů na herní ploše.

Pro úkol detekce herní plochy jsme předpokládali a testovali použití několika různých řešení, nakonec i zde vyšla jako nejspolehlivější metoda použití ArUco značek, umístěných v rozích hrací plochy. Průběh testování různých metod řešení výřezu obrazu je popsán v podkapitole 9.1. Na to v podkapitole 9.2 navazuje testování spolehlivosti metod detekce a dekódování vizuálních kódů objektů, umístěných v herní ploše.

9.1 Testování spolehlivosti metod výřezu obrazu

9.1.1 Výřez z obrazu se statickými body

Získaný obraz je složen z pixelů, kde každý z pixelů nese hodnotu a má svou X a Y pozici. Právě díky X a Y pozicím je možné z obrazu provést výřez ve tvaru čtverce a odřezky tak odstranit ze zpracovávaného obrazu. Takový ořez lze provést pomocí pevně definovaných bodů, které předáme programu a následně je proveden ořez. Na obrázku níže je možné vidět pevně stanovené body, označené modrou tečkou. S ořezem je možné následně manipulovat jako s běžným obrázkem. Dále dochází k segmentaci obrazu na 25 dílů.



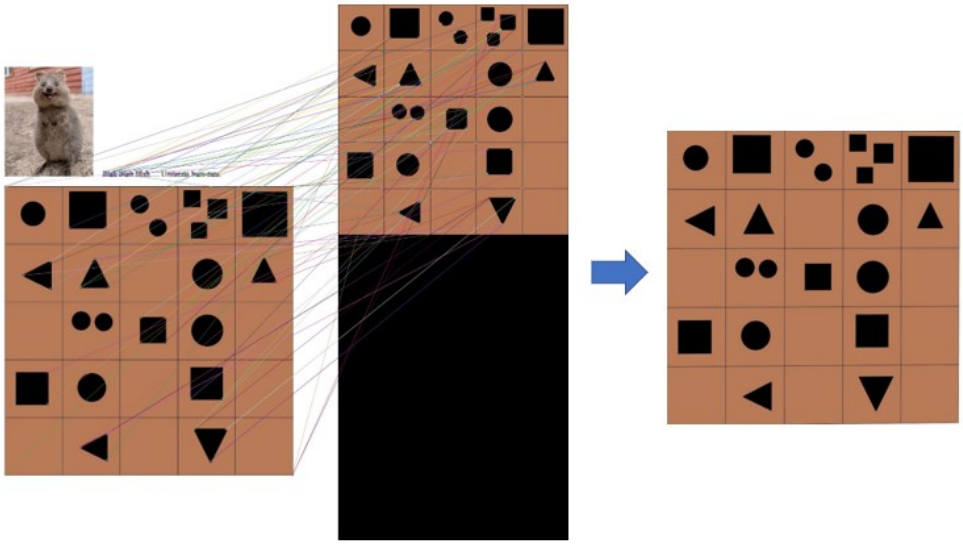
Obrázek 32: Příklad ořezu obrázku s pevnými body a rozdělení na segmenty

Tyto pevně definované body ovšem zamezují jakémukoliv pohybu kamery, který by měl za následek špatně definovanou herní plochu, a proto tato varianta není vhodná pro projekt.

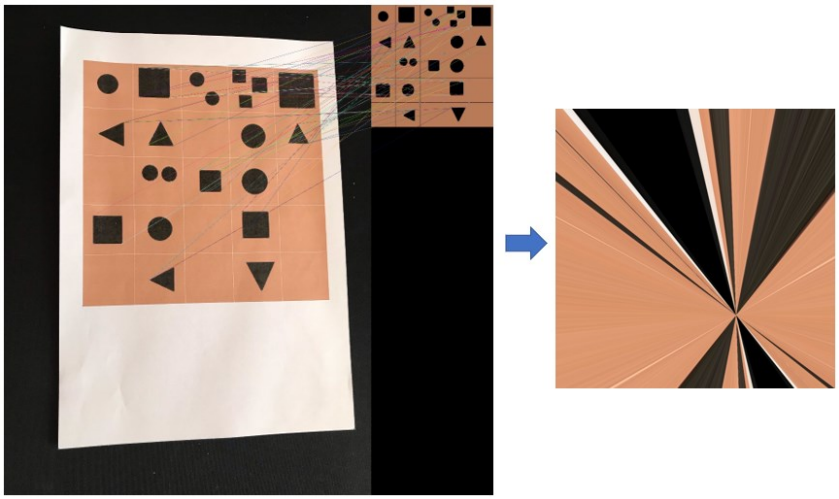
9.1.2 Výřez obrazu pomocí Image alignment

Image aligment je proces, kde jsou na vstupu vyžadovány dva parametry, dva obrázky zobrazující stejný předmět. Na prvním vzorovém obrázku může být například naskenovaný dokument a na druhém snímek z fotoaparátu stejného dokumentu. Následně jsou tyto dva obrázky porovnány a nalezeny spojité body, které vyfotografovaný obrázek přetransformují, aby byl zobrazený jako obrázek naskenovaný.

Pro tento způsob byly provedeny tři pokusy transformace, kde byl úspěšný pouze jeden. Na obrázku níže je možné vidět na levé straně stránku vytvořenou ve Word aplikaci, která obsahuje obrázek zvířete, text a čtvercové zobrazení plochy. Ve středu je obrázek, se kterým se hledají spojité body a výsledek této operace je zobrazený v pravé části obrázku.



Obrázek 33: Úspěšný pokus Image alignment z Word dokumentu



Obrázek 34: Neúspěšný pokus Image alignment z fotografie

Image alignment byl shledán kvůli nespolehlivému chování jako nevhodný způsob výřezu pro tento projekt.

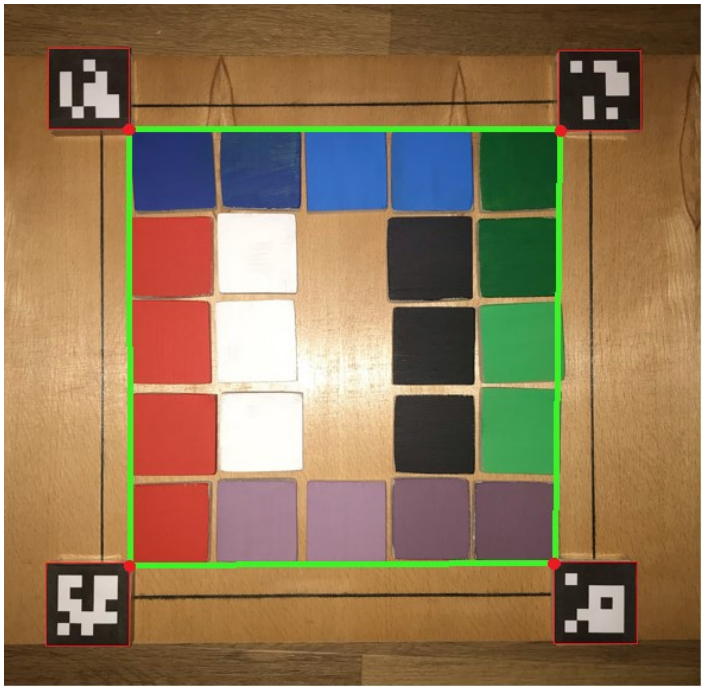
9.1.3 Výřez obrazu pomocí největší spojité uzavřené hrany

Při požadavku výřezu části obrazu se nabízí možnost oříznout největší spojitou konturu, která se v obraze nachází. Tato možnost byla otestována a shledána za nedostačující, z důvodu možného narušení kontury a to pouhým stínem z jakéhokoliv objektu.

9.1.4 Výřez obrazu pomocí ArUco značek

ArUco značky jsou snadno detekovatelné, lze snadno získat souřadnice hran a i natočení a tak jsou ideálními značkami pro vytvoření výřezu, přičemž není příliš limitovaný pohyb ani

úhel kamery, ze kterého je hrací plocha snímána. Výřez vytvořený takovýmto způsobem je narušitelný pouze při nečitelnosi jedné ze čtyř značek. Z těchto důvodů byla metoda výřezu pomocí ArUco značení zvolena jako metoda s níž se bude pracovat v projektu.



Obrázek 35: Herní plocha vymezená čtyřmi ArUco značkami

9.2 Testování spolehlivosti detekce objektů na hrací ploše

Detekce objektů probíhá z již ořezaného obrazu a tím jsou odstraněné rušivé elementy, nacházející se v blízké vzdálenosti od herní plochy. Na samotnou detekci je kladen důraz hlavně z hlediska spolehlivosti a přesnosti, kdy je požadováno, aby byl objekt detekován i za špatného osvětlení a nedocházelo například k záměně unikátních identifikátorů. V kapitolách níže jsou uvedeny pokusy o nalezení nejspolehlivějšího způsobu detekce a výběr jednoho, který byl použit pro tento projekt.

9.2.1 Detekce QR kódu

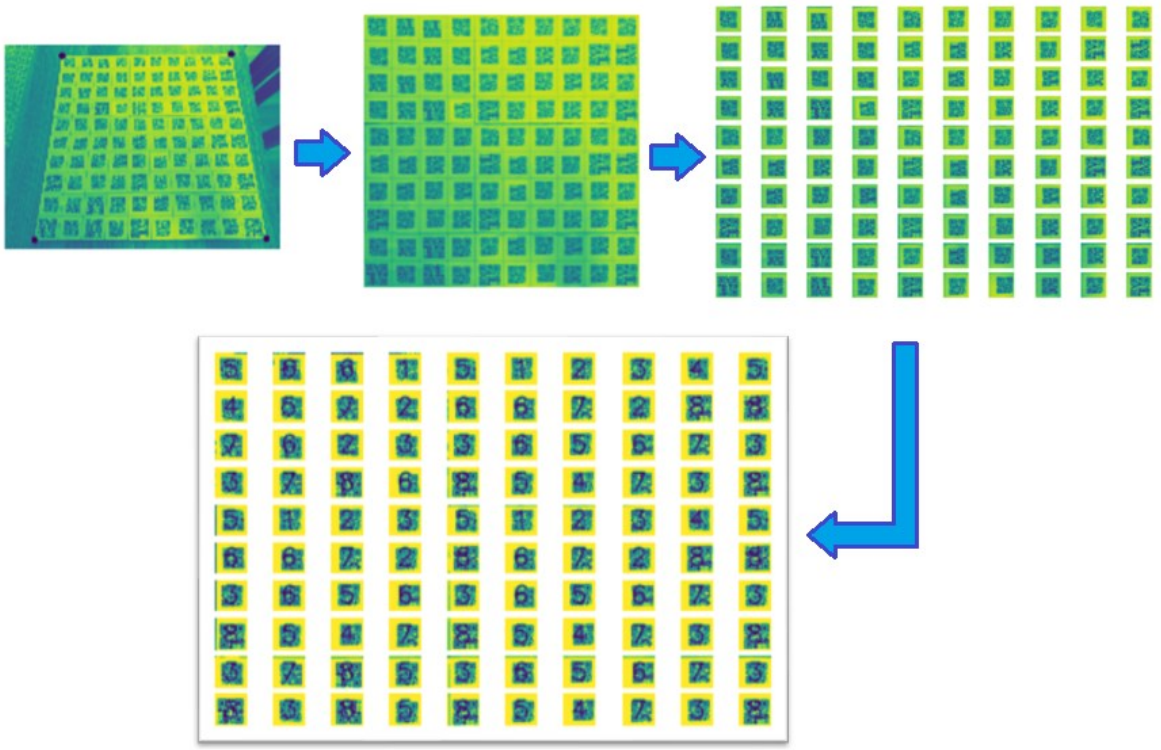
První unikátní identifikátor, který přistane člověku na mysli je QR kód, jelikož se s ním dnes a denně setkáváme v běžném životě. Pro identifikování QR kódu bylo využito dvou způsobů detekce. První způsob detekce QR kódu z obrázku je možné vidět v kódu níže, kde je pro detekci využíváno OpenCV knihovny a data získána z detekce jsou následně zobrazena.

```
import cv2
import numpy as np

img = cv2.imread('C:\OpenCV_image\QR_codes\photo2.jpg', cv2.IMREAD_COLOR)
qrDecoder = cv2.QRCodeDetector()

# Detect and decode the qrcode
data,bbox,rectifiedImage = qrDecoder.detectAndDecode(img)
if len(data)>0:
    print("Decoded Data : {}".format(data))
else:
    print("QR Code not detected")

cv2.putText(img, data[0], org=(70, 100),\
            fontFace=cv2.FONT_HERSHEY_SIMPLEX,\
            fontScale=2, color=(0, 255, 0),thickness=3)
cv2.imshow('original', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Druhý způsob detekce byl uskutečněn za pomocí knihovny Pyzbar a OpenCV, kde OpenCV knihovna sloužila pouze k načtení obrázku a Pyzbar knihovna sloužila k samotné detekci QR kódu a data byla následně opět zobrazena. Python kód pro detekci QR kódu je možné vidět níže.

```
from pyzbar.pyzbar import decode
import cv2

imgOriginal = cv2.imread('C:\OpenCV_image\qrcode.jpg', 0)
decoded = decode(imgOriginal)
data = decoded[0].data.decode("utf-8")
print (data)
```

Při porovnání těchto dvou variant detekce QR kódu vyšlo najevo, že spolehlivější způsob detekce proběhl za použití knihovny Pyzbar. Následuje tedy vytvoření provizorní herní desky, čítající sto QR kódů umístěných vedle sebe a testování. První testování je provedeno z fotky pořízené mobilním fotoaparátem za dobrých světelných podmínek a vysokého rozlišení. Prvním krokem při detekci bylo provedení výřezu pomocí statických bodů umístěných na hranách provizorní herní plochy. Druhý krok spočíval v rozčlenění jednotlivých QR kódů do samostatných jednotek a poslední krok prováděl samotnou detekci jednotlivých QR kódů, přičemž došlo k převedení všech částí na binární zobrazení pomocí Image Thresholdingu, pro usnadnění a zvýšení úspěšnosti detekce. Výsledek detekce QR kódů byl stoprocentní a následovalo testování s kamerou.



Obrázek 36: Průběh detekce QR kódů z fotografie pomocí statických bodů

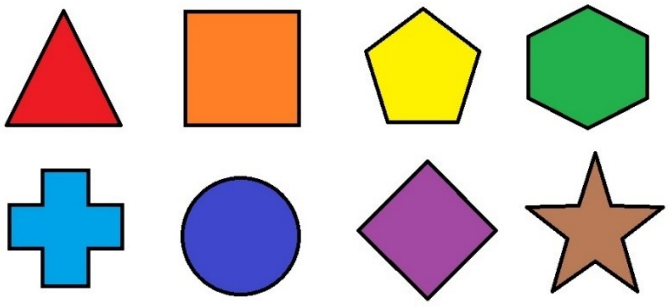
Detekce QR kódů pomocí kamery probíhala na stejném principu jako detekce z fotografie. Opět bylo prvním krokem provedení výřezu, následovala segmentace jednotlivých QR kódů a detekce každého z QR kódů, který byl před samotnou detekcí převeden na binární obrázek za pomocí Image Thresholdingu. Obrázek níže zobrazuje ve svém středu herní plochu, která čítá sto QR kódů. Po provedení ořezové části je zobrazen levý horní QR kód v levé části obrázku před aplikováním Image Thresholdingu a následovalo očištění obrázku od šumu a dalších negativních vlivů právě zmíněným procesem Image Thresholding. Výsledek takovéhoto odstranění nežádoucích efektů je možné vidět v pravé části obrázku, takzvané převedení obrázku na binární obrázek. Následující proces obsahuje segmentaci a detekci, při které je nad každým QR kódem zobrazeno číslo, které jednotlivé kódy nesou. Za předpokladu že je nad kódem zobrazeno písmeno X, nebyl QR kód detekován.

Obrázek 37: Průběh detekce sta QR kódů za pomocí kamery

Z výsledků viditelných výše na obrázku je patrné, že detekce pomocí QR kódů není vhodná za podmínek, kdy je QR kód umístěn ve větší vzdálenosti při nedostatečném rozlišení, a tak není vhodné využít QR kódy pro tento projekt.

9.2.2 Detekce základních tvarů

Základní geometrické tvary jsou dobře viditelné, jednoduché a mohou být i dostatečně veliké, aby zabrali celou plochu herní kostky. Pro pokus bylo vybráno osm nejjednodušších geometrických tvarů, které byly následně podrobeny detekci.



Obrázek 38: Osm základních geometrických tvarů pro detekci

Pro testování byla herní plocha vytištěna na papír o velikosti A4, která obsahovala 25 geometrických tvarů s černým ohraničením. Získaný výřez obrazu byl rozčleněn stejně jako při předchozích testech na x částí a nad každou samostatnou částí probíhala detekce objektu.

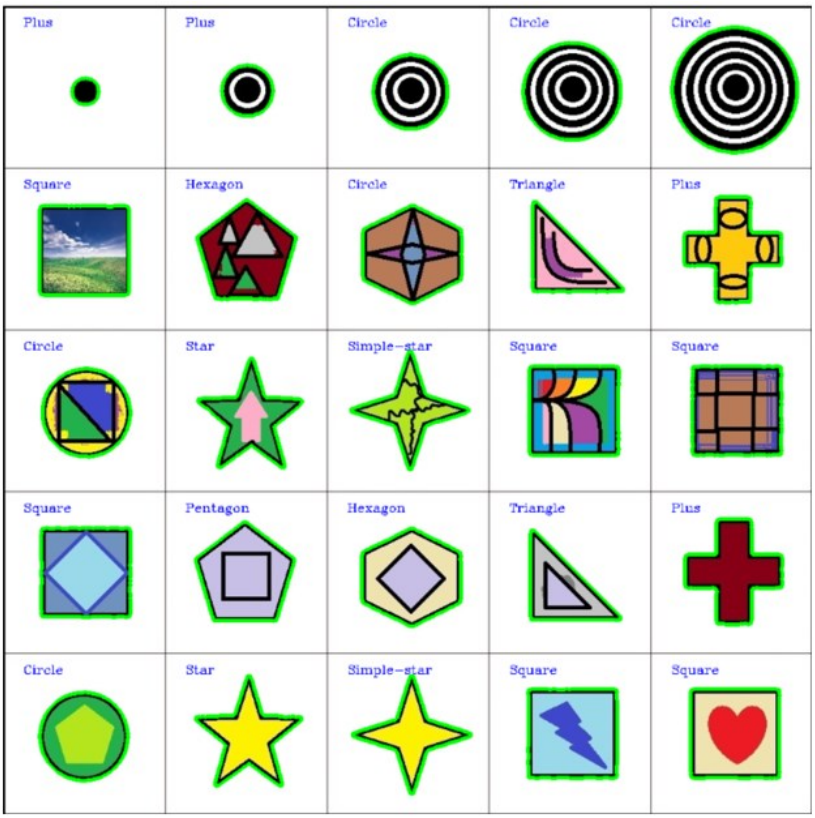
Před samotnou detekcí byly jednotlivé segmenty převedeny na černobílé obrázky („cv2.COLOR_BGR2GRAY“), bylo aplikované Gaussovské rozostření pro snížení obrazového šumu („cv2.GaussianBlur“) a Canny filtr určený pro zobrazení hran objektu („cv2.Canny“). Obrázek, který byl upraven všemi efekty a filtry, mohl nyní sloužit pro detekci kontur, za využití funkce „cv2.findContours“ z knihovny OpenCV. Níže lze vidět Python ukázku kódu pro zpracování jednoho z dvaceti pěti segmentů obrazu a aplikování funkce pro nalezení spojitých vnějších kontur objektu a následné vykreslení. Po vykreslení probíhá implementace Douglas-Peucker algoritmu, který má za úkol z křivky tvořené přímkami vytvořit stejnou křivku složenou z méně křivek.

```
gray = cv2.cvtColor(imageArray[i], cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (7, 7), 0)
edged = cv2.Canny(blur, 75, 200)

contours, _ = cv2.findContours(edged, cv2.RETR_EXTERNAL,\
                               cv2.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv2.contourArea, reverse=True)

cv2.drawContours(imageArray[i], contours, -1, (0,255,0), 3)

for contour in contours:
    peri = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.01 * peri, True)
```

Podle proměnné „approx“, která nese počet křivek dané kontury, je následně rozhodnuto, o jaký tvar se jedná. Takovéto rozdělení je možné vidět v kódu níže, kde je každému počtu křivek přiřazen tvar. Například pokud se jedná o konturu se třemi křivkami, jedná se o tvar trojúhelníku, nebo pokud se jedná o konturu tvořenou čtyřmi přímkami, tak je tvar čtverec. Vzhledem k pevně definovaným tvarům je patrné, že geometrický tvar složený z dvanácti přímek bude znak znaménka plus. Tento znak plus se skládá ze dvanácti přímek a je tudíž nepochybné, že jakýkoliv jiný tvar s větším počtem přímek bude kruh. Kruh se totiž skládá z několika desítek malých spojených přímek utvářející výsledný tvar kruhu. V kódu níže je také možné spatřit vykreslení názvu jednotlivých geometrických tvarů za pomocí funkce „putText“, které jsou předány parametry jako umístění textu, font, barva, tloušťka a velikost textu.

```
if len(approx) == 3:
    cv2.putText(imageArray[i], "Triangle", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 4 :
    cv2.putText(imageArray[i], "Square", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 5 :
    cv2.putText(imageArray[i], "Pentagon", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 6 :
    cv2.putText(imageArray[i], "Hexagon", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 8 :
    cv2.putText(imageArray[i], "Simple-star", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 10 :
    cv2.putText(imageArray[i], "Star", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
elif len(approx) == 12 :
    cv2.putText(imageArray[i], "Plus", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
else:
    cv2.putText(imageArray[i], "Circle", (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 0, 0))
```

Výsledek takovéto detekce byl uspokojivý, avšak ne stoprocentní. Při vykreslování kontur a názvů detekovaných obrazců bylo patrné, že geometrické tvary jsou cestou, ale ne natolik spolehlivou, aby byly využity pro tento projekt.

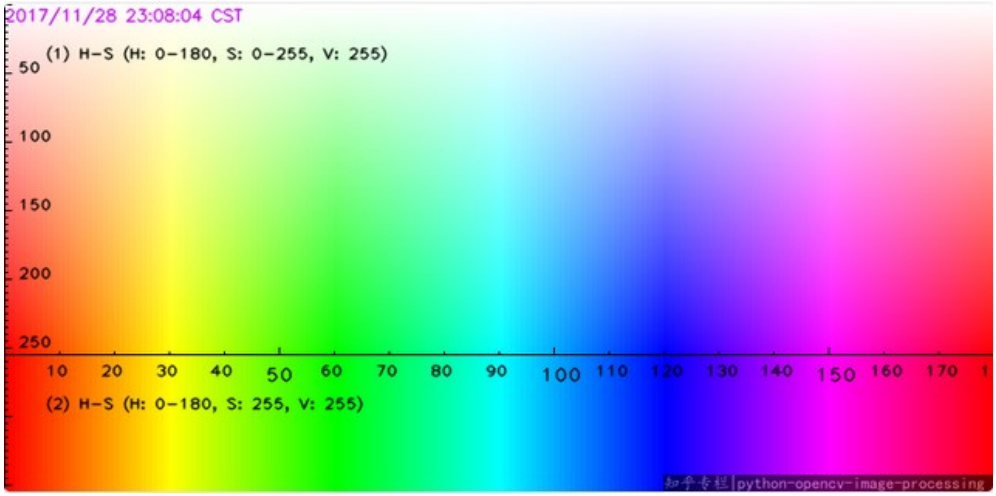


Obrázek 39: Ukázka detekce geometrických tvarů

9.2.3 Detekce barvy

Předposledním způsobem detekce je způsob detekce podle barvy objektu. Na první pohled způsob, který je možné snadno ovlivnit vnějšími faktory, se nezdá jako spolehlivý, ale byl taktéž otestován a podroben zkouškou osvětlení.

V prvním kroku dojde k převodu RGB (Red, Green, Blue) obrázku na HSV (Hue, Saturation, Value) obrázek a k výběru oblasti, ze které bude barva rozpoznávána. Vzhledem k tomu, že kostky jsou celé pokryté barvou, může dojít k výběru přímo ze středu kostky. Takový výběr spočívá v definování souřadnic a získání pixelu na těchto souřadnicích s jeho hodnotami.



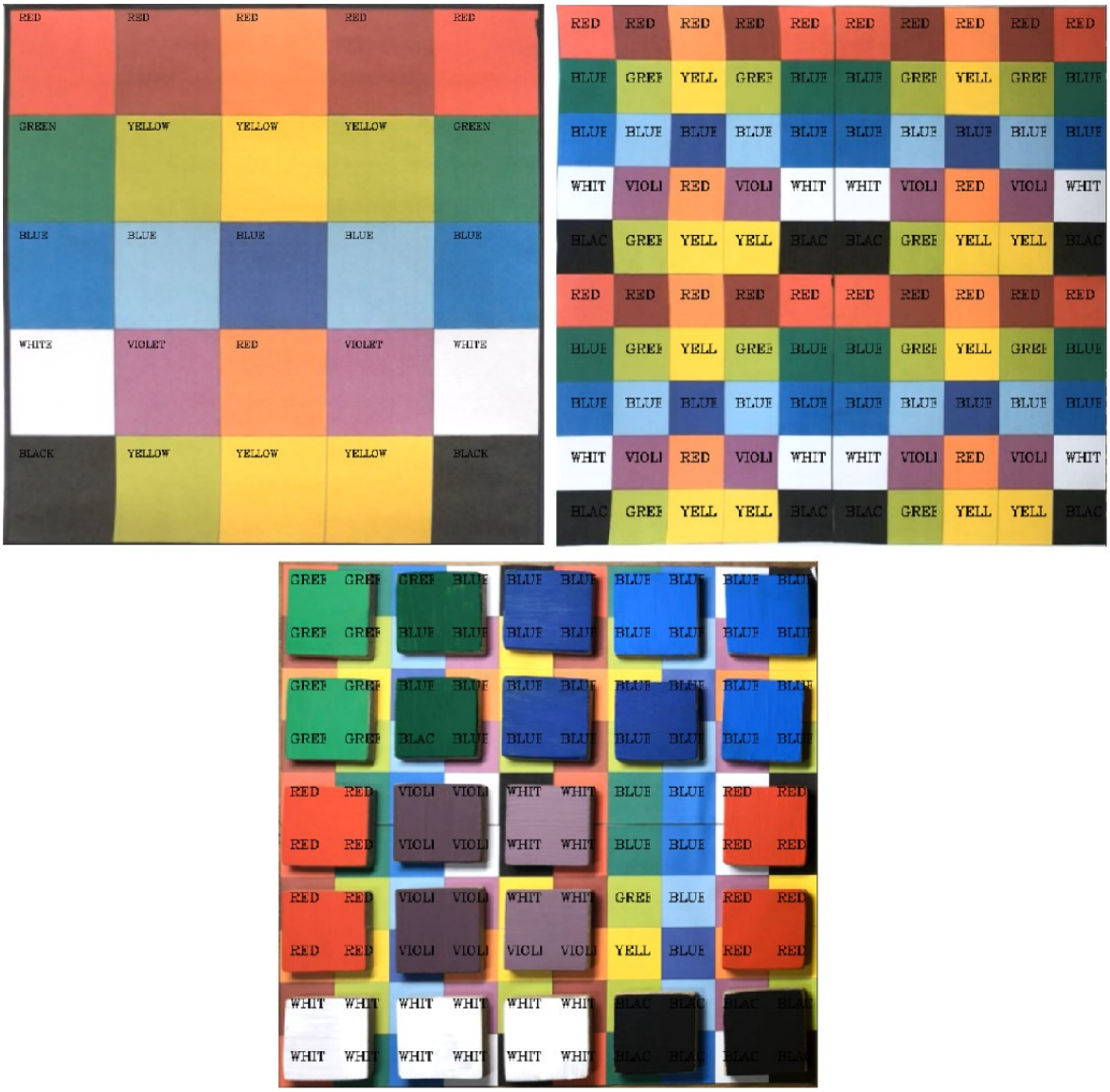
Obrázek 40: HSV mapa pro získání barvy na základě hodnot

Následně podle získaných hodnot pixelu na dané souřadnici, dochází k definici, o jakou barvu se jedná podle HSV mapy. V kódu pod textem je možné vidět ukázku, jaké byly nastavené hranice barev pro testování.

Velkou nevýhodou při detekci barev je již zmíněné světlo, které dokáže barvy na kostkách zesvětlit, nebo případně ztmavit. Takovéto náhlé změny při staticky definovaných barvách jsou nežádoucí a znemožňují spolehlivou detekci. Ovšem za předpokladu, možného zajištění statického osvětlení by tato možnost byla efektivní.

```
pixel_center = hsv_frame[cy, cx]
hue = pixel_center[0]
saturation = pixel_center[1]
value = pixel_center[2]

color = "Undefined"
if value < 70:
    color = "BLACK"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255))
elif saturation < 40:
    color = "WHITE"
    cv2.putText(imageArray[i], color, (20, 20),,\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
elif hue < 18:
    color = "RED"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
elif hue < 30:
    color = "YELLOW"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
elif hue < 78:
    color = "GREEN"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
elif hue < 131:
    color = "BLUE"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
elif hue < 175:
    color = "VIOLET"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
else:
    color = "NONE"
    cv2.putText(imageArray[i], color, (20, 20),\
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
```

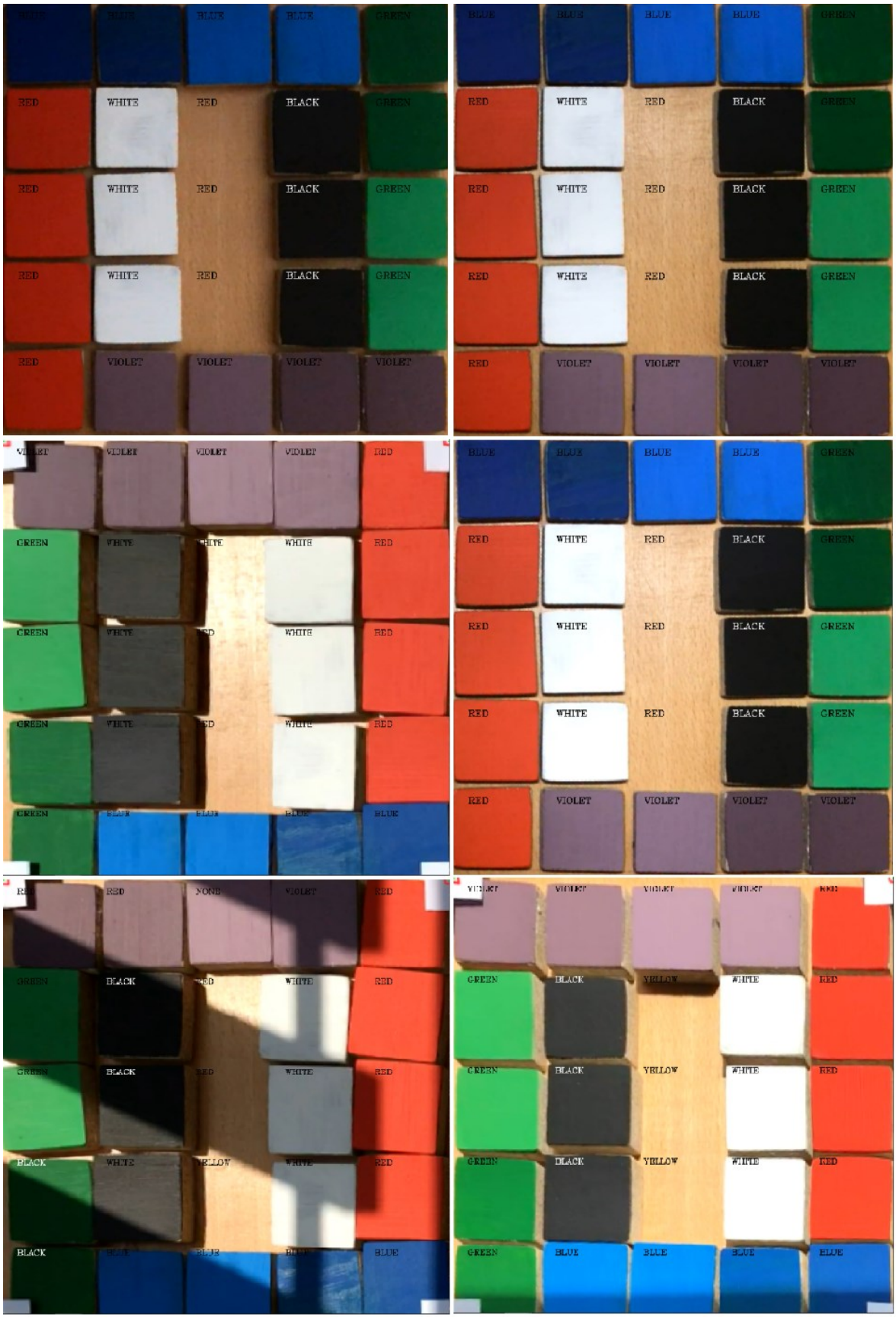
Následuje samotné testování barev vytištěných na papíře a kostek natřených barvou. Na následujících snímcích je možné zahlédnout text na každé z kostek, který definuje barvu při detekci. Nepatrnou nepřesnost je možné pozorovat při chybové detekci fialové ve třetím snímku, kdy je mylně označována za bílou barvu.



Obrázek 41: Detekce barev při statickém osvětlení

Po překvapivě úspěšné detekci se statickým osvětlením přichází na řadu scéna s rozdílným osvětlením, při kterém bude na herní desku uvržen stín a přímé sluneční záření. Výsledky tohoto testování je možné vidět níže na pořízených snímcích.

Celkový výsledek detekce barev byl shledán stejně jako předchozí typy detekcí za nespolehlivý, z důvodu možného narušení různým typem osvětlení, jelikož není možné za stanovených podmínek docílit statického osvětlení.

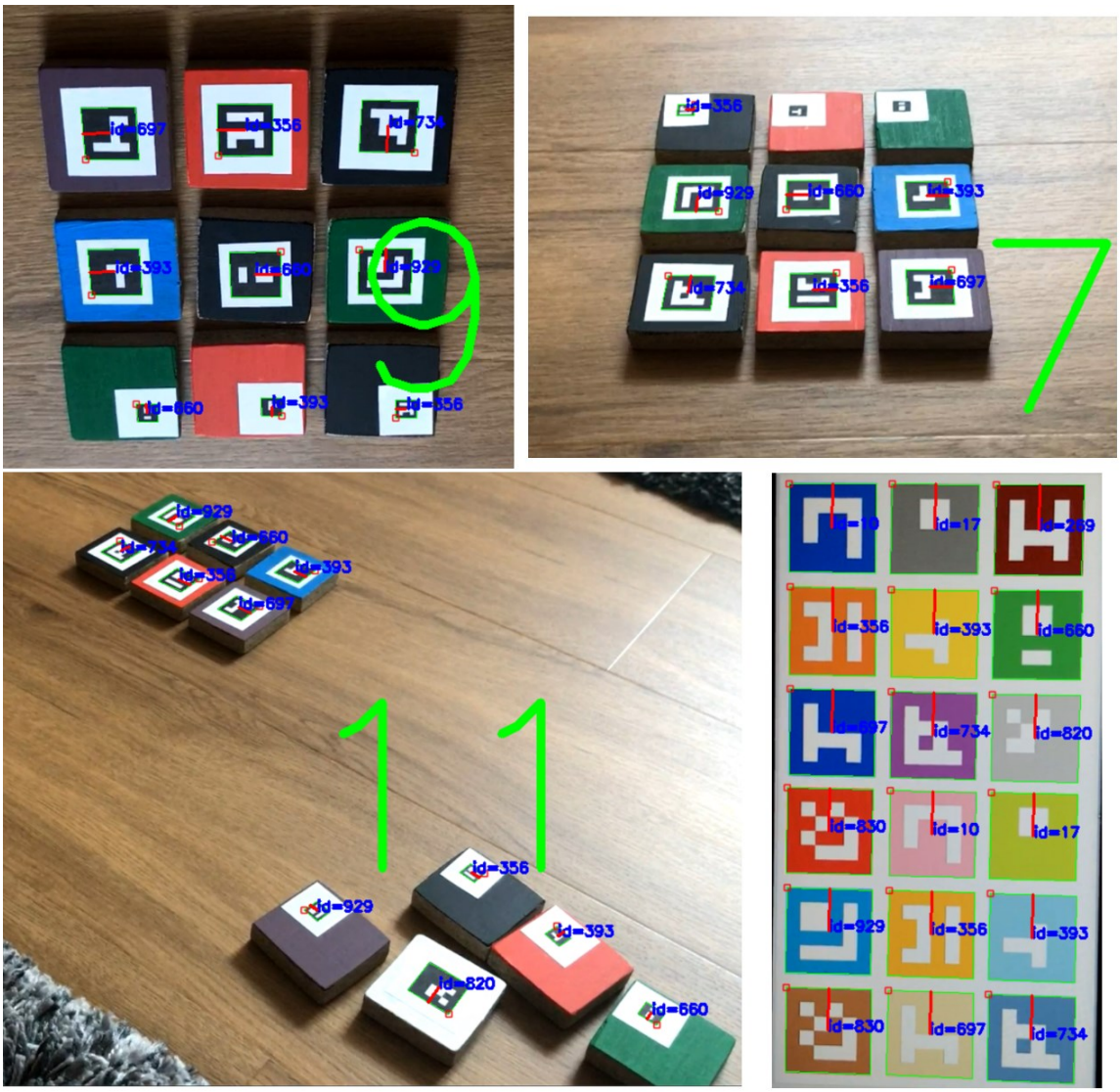


Obrázek 42: Detekce barev s různým osvětlením

9.2.4 Detekce ArUco značek

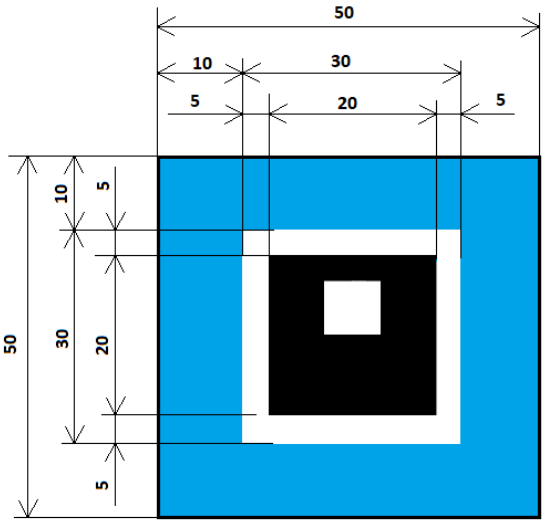
ArUco značení se osvědčilo jako spolehlivé už při použití výřezu herní plochy, a tak bylo vybráno i pro samotnou detekci unikátních kostek. Přestože toto značení bylo do jisté míry otestováno samotným výřezem, bylo nutné zjistit požadovanou velikost značení pro bezproblémovou detekci s ohledem na vzdálenost kamery od herní plochy.

Pro takové testování bylo vytištěno několik velikostí ArUco značek a nalepeno na kostky dříve využité pro detekci podle barvy. Vytištěné ArUco značky se dále lišily i prostorem a jeho velikostí okolo značek, protože při detekci ArUco je nezbytné zajistit bílý prostor okolo samotného značení, pro zajištění úspěšné detekce. Snímky zobrazeny pod textem zobrazují kostky s ArUco značením s vykresleným ID, které značky nesou, směrem natočení a v neposlední řadě velké zelené číslo, které udává celkový počet detekovaných kostek z obrazu.



Obrázek 43: Detekce ArUco značení

Výsledkem testován bylo zjištění, že pro bezproblémovou detekci je nutné zajistit okolo značení ArUco prostor o velikosti alespoň pět milimetrů. Samotné značení by přitom mělo být veliké jako čtverec se stranou dvacet milimetrů.



Obrázek 44: Definovaná velikost značení na herní kostce

Při testování vyšlo taktéž najevo, že detekované značení by mělo pocházet z ArUco knihovny čítající nejmenší možný počet unikátních znaků o nejmenší možné velikosti. Čím větší je množství unikátních značení, tím je větší možnost chybové detekce, která může vzniknout například záměnou jednoho identifikátoru za druhý. Tento druh chyby je možné eliminovat i samotnou velikostí ArUco, kde značení se liší i velikostí vnitřní matice, která při větší velikosti může taktéž zapříčinit chybnou detekci.

Funkce „findArucoMarkersFive“ zobrazuje průběh detekce ArUco značek v obraze. Funkci je předán parametr ve formě snímku z kamery, který je následně převeden na černobílý obrázek. Probíhá načtení ArUco knihovny o velikosti padesáti unikátních značení a pro značky s vnitřní maticí 5x5. Následuje samotná detekce pomocí funkce „detectMarkers“ z OpenCv knihovny. Funkce vrací „bboxs“, což je pole rohů značek a samotné ID.

```
def findArucoMarkersFive(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #Load the dictionary that was used to generate the markers.
    arucoDict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_50)
    # Initialize the detector parameters using default values
    arucoParam = cv2.aruco.DetectorParameters_create()
    # Detect the markers
    bboxs, ids, _ = cv2.aruco.detectMarkers\
                    (gray, arucoDict, parameters = arucoParam)
    # cv2.aruco.drawDetectedMarkers(img, bboxs)
    return bboxs, ids
```

10 IMPLEMENTACE DETEKCE A DEKÓDOVÁNÍ ZNAČEK

Ve 3. bodu zadání této bakalářské práce se předpokládala implementace detekce a dekódování značek pomocí knihovny. Vzhledem k celkové architektuře systému, kde detekce, dekódování a komunikace běží na Raspberry Pi, má výsledná implementace formu aplikace v jazyce Python, běžící na Raspberry Pi.

V následujícím textu je tato aplikace podrobněji popsána.

10.1 Dekódování značek a získání souřadnic

Díky zvolenému způsobu detekce objektů není získání souřadnic, na kterých značení leží, náročné. Hlavně díky funkci pro detekci, jejímž výsledkem jsou rohy detekované značky. Právě díky těmto rohům je možné získat střed kostky. Definovaný výřez herní plochy je po získání veliký 1000x1000 pixelů. V tomto rozmezí se nachází všechny herní kostky.

Jeden z požadavků je rozčlenit hrací plochu na matici 8x8, tudíž se bude skládat ze 64 čtverců o velikosti 125x125 pixelů. Podle toho, na jaké souřadnici se kostka nachází na herní ploše jí bude přiděleno místo v matici 8x8. Níže definovaná funkce „getMarkerCenter“ má dva vstupní parametry. Jedním vstupním parametrem je množina nalezených rohů funkcí „cv2.aruco.detectMarkers“ a druhým vstupním parametrem jsou identifikační čísla, taktéž nalezených stejnou funkcí jako množina rohů. Ve funkci je vypočítán střed značky pomocí čtyř rohů a jejich souřadnic. Po získání souřadnice středu kostky jsou poděleny získané souřadnice x a y číslem 125, díky čemuž získáme pozici v matici 8x8. Výstupem funkce je množina souřadnic s přiděleným identifikačním číslem.

```
def getMarkerCenter(corners, ids):
    centers = []
    for i in range(0,len(corners)):
        px = math.floor((corners[i][0][0][0] + corners[i][0][1][0]\
                         + corners[i][0][2][0] + corners[i][0][3][0]) * 0.25)
        py = math.floor((corners[i][0][0][1] + corners[i][0][1][1]\
                         + corners[i][0][2][1] + corners[i][0][3][1]) * 0.25)
        px = math.ceil(px/125)
        py = math.ceil(py/125)
        center = [ids[i][0], px, py]
        centers.append(center)
    return centers
```

S takto získanými souřadnicemi není třeba dále manipulovat a jsou tudíž připraveny k přenosu na chytré zařízení, kde poslouží k vykreslení grafiky.

11 NÁVRH ZPŮSOBU KOMUNIKACE

Komunikace mezi Raspberry PI a mobilním telefonem přenáší data v podobě souřadnic herních kostek. Tato komunikace je jedním z hlavních pilířů, potřebných pro funkčnost celého projektu. Proto je v práci kladen důraz na plynulost a spolehlivost zvolené komunikace, která zároveň udává i počet snímků za vteřinu při vykreslování grafiky. Testovány byly dva možné způsoby komunikace: Bluetooth a Wi-Fi.

11.1 Bluetooth

Prvním ze dvou způsobů je Bluetooth komunikace. Bluetooth modulem disponuje jak Raspberry Pi, tak i námi použité chytré zařízení. Komunikace přes Bluetooth by byla výhodná pro zachování funkce Wi-Fi připojení k internetu na straně mobilního telefonu.

Při komunikaci přes Bluetooth musíme zajistit snadné připojení zařízení, přičemž Raspberry Pi má automaticky Bluetooth zapnutý po spuštění a není třeba nic nastavovat.

Následující kód slouží pro komunikaci dvou zařízení. Jedním zařízením je Raspberry Pi, který se v aktuální situaci chová jako server a druhým zařízením je telefon, který je v roli klienta. První dojde k zapnutí serveru, který čeká na připojení na portu číslo 29 a umožňuje připojení jednomu vzdálenému zařízení s jakoukoliv adresou. Dále přichází klient, který se spuštěním skriptu připojí na server pomocí MAC adresy a portu, kde hned přijímá data odeslaná od serveru po každé půl vteřině, dokud nedojde k přerušení.

Server

```
import socket
import time

port = 29
s = socket.socket(socket.AF_BLUETOOTH,\
                  socket.SOCK_STREAM,\
                  socket.BTPROTO_RFCOMM)
s.bind((socket.BDADDR_ANY,port))
s.listen(1)
print("listening")
client, address = s.accept()
print(address)
try:
    print("Sending data")
    while 1:
        time.sleep(0.5)
        client.send(bytes("M", 'UTF-8'))
except:
    print("Closing socket")
    s.close()
```

Client

```
import socket

# Raspberry Pi MAC address
raspberryMACAddress = 'E4:5F:01:BC:2B:10'
port = 29
s = socket.socket(socket.AF_BLUETOOTH,\
                  socket.SOCK_STREAM,\
                  socket.BTPROTO_RFCOMM)
# connect to Raspberry pi to port 29
s.connect((raspberryMACAddress,port))
while 1:
    data = s.recv(1024) # receive data
    print(data) # print data into console

s.close()
```

Obrázek 45: RFCOMM komunikace dvou zařízení

Výše ukázaná komunikace pracuje podle předpokladů a nyní následuje testování na telefonu, které vyžaduje přepsání klientské strany do C# kódů. V níže uvedeném C# kódu je využita externí knihovna „InTheHand“ zajišťující Bluetooth komunikaci. Dále je možné v kódu vidět staticky definovanou MAC adresu Raspberry Pi, zvolené Guid, které udává zvolený způsob komunikace, port a následné připojení na Endpoint (Raspberry Pi), od kterého získává zaslaná data, které jsou zobrazována do terminálu.

```
using System;
using System.Text;
using InTheHand.Net;
using InTheHand.Net.Sockets;
using System.IO;

namespace testTesttestTest
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string rpiMAC_str = "E4:5F:01:BC:2B:10";
            BluetoothAddress rpiMAC = BluetoothAddress.Parse(rpiMAC_str);
            Guid serviceClass = new Guid("34B1CF4D-1069-4AD6-89B6-E161D79BE4D8");
            int port = 29;
            // Guid serviceClass = BluetoothService.SerialPort;

            BluetoothEndPoint ep = new BluetoothEndPoint(rpiMAC, serviceClass, port);
            BluetoothClient cli = new BluetoothClient();
            cli.Connect(ep);
            Stream peerStream = cli.GetStream();
            byte[] received = new byte[1024];
            while (true)
            {
                peerStream.Read(received, 0, received.Length);
                string data = Encoding.ASCII.GetString(received);
                Console.WriteLine(data);
            }
        }
    }
}
```

Při pokusech o přenesení C# kódu do Unity aplikace došlo k potížím, způsobených knihovnou „InTheHand“, která není v Unity podporovaná. Následovaly tedy pokusy o zprovoznění Bluetooth klienta na Android telefonu přes aplikaci Unity pomocí několika bezplatných knihoven pro Bluetooth komunikaci, které byly odzkoušeny. Bohužel žádná z knihoven nefungovala. Problém byl konzultován s jedním autorem použité knihovny, který byl ochoten poradit, avšak ani poté nedošlo k vyřešení problému. Následovaly neúspěšné pokusy o vytvoření vlastní knihovny pro Bluetooth a konzultace s učitelem zaměřujícím se na výuku Unity,

ani to však nepomohlo k vyřešení daného problému. Od Bluetooth komunikace bylo upuštěno a nastal přesun na komunikaci přes Wi-Fi.

Během testování Bluetooth docházelo i k potížím na straně RPI, kde nastával problém, kdy Bluetooth zařízení RPI nebylo viditelné pro ostatní zařízení a bylo nutné použít příkaz „sudo service dbus restart“, načež Bluetooth bylo detekovatelné, ale po restartování RPI bylo tento proces potřeba opakovat. Tato skutečnost taktéž přispěla k volbě alternativní komunikace.

11.2 Wi-Fi

Neúspěšné pokusy o zprovoznění Bluetooth komunikace zapříčinily nutnost jiného řešení, kterým je komunikace přes WiFi resp. TCP a UDP. Stejně jako u Bluetooth bylo prvním krokem zprovoznění jednoduchého odesílání zpráv a příjem na druhé straně. Komunikace přes Wi-Fi vyžaduje připojení obou komunikujících zařízení na stejné síti. Takovou komunikaci není problém zprovoznit v domácích podmínkách, avšak například při připojení na školní Wi-Fi síť nastal problém, který zapříčinil, že jedno zařízení nebylo schopné odeslat informace na druhé zařízení, které nebylo viditelné v rámci sítě ostatním zařízením.

Chytré zařízení

```
import socket

sock = socket.socket(socket.AF_INET,\
                     socket.SOCK_DGRAM)
# Raspberry Pi address
address = ("192.168.1.54",11000)
sock.bind(address)

while True:
    data,addr = sock.recvfrom(1024)
    print(data.decode("utf-8"))
```

Raspberry Pi

```
import socket

sock = socket.socket(socket.AF_INET,\
                     socket.SOCK_DGRAM)
for i in range(0,10):
    sock.sendto(bytes("WOW", "utf-8"),\
                ("192.168.1.37",8000))
    i += 1
print("konec")
```

Obrázek 46: UDP komunikace ve stejné síti

Stejně jako v Bluetooth případě je nutné přenést Python kód pro chytré zařízení do C# kódu pro mobilní aplikaci v Unity a Python kód pro Raspberry Pi upravit podle potřeb. Takový výsledný C# kód je možné vidět v ukázce níže, která slouží pro příjem dat z Raspberry Pi. Funkce „Listen“ na telefonu komunikuje s Raspberry Pi přes UDP. Souřadnice získané s Raspberry Pi jsou ve tvaru pole. Pokud na Raspberry Pi nedojde k detekci herních kostek, například vlivem zakrytí jedné ze čtyř kostek sloužící k vyhrazení herní plochy, obdrží telefon prázdné pole. Pokud nastane tento scénář, kde není žádná kostka detekovaná, C# kód si bude

udržovat poslední získané souřadnice, které obdržel a nenastane tak situace, kde by nebyla vykreslená žádná kostka.

Takovéto řešení značně zpříjemní manipulaci s herními kostkami, při které může snadno dojít k zakrytí kostky pro vyhrazení herní plochy a nebude to mít ve výsledku vliv na zobrazení ostatních kostek, již detekovaných v předchozích krocích.

```
public void Listen()
{
    listener = new UdpClient(port);
    rpiEP = new IPEndPoint(rpi_address, port);
    listener.Connect(rpiEP);
    listener.Client.Blocking = false;

    while (true)
    {
        try
        {
            //Debug.Log("Sending Message");
            listener.Send(sendMessage, sendMessage.Length);
            //Debug.Log("Waiting for message");
            byte[] bytes = listener.Receive(ref rpiEP);

            //Debug.Log("Received message from " + groupEP);
            result = Encoding.ASCII.GetString(bytes, 0, bytes.Length);
            if (!result.Equals("[]"))
            {
                coordinations = result;
            }
            //Debug.Log(Encoding.ASCII.GetString(bytes, 0, bytes.Length));
        }
        catch (SocketException e)
        {
            Debug.Log(e);
        }
    }
}
```

Komunikace je nyní zprovozněna a zbývá zajistit, aby telefon a Raspberry Pi byly připojené ve stejné viditelné síti, přes kterou by mohlo dojít ke komunikaci. Zde je problém vyřešený tím, že Raspberry Pi bude zároveň sloužit jako Wi-Fi přístupový bod. Takovýto přístupový bud vytvoří viditelný hotspot s názvem, na který se může připojit jakékoliv zařízení po zadání nastaveného hesla. Raspberry Pi tak bude mít na svém hotspotu připojený telefon, ne který bude odesílat data.

Wi-Fi access point je nakonfigurovaný, aby povolil připojení pouze jednomu zařízení ve stejnou dobu, kterému bude přiřazena IP adresa 192.168.5.100. Na tuto adresu budou po připojení zařízení na hotspot odesílána data. Po odpojení zařízení z hotspotu je nastavený minimální možný čas (2 minuty) pro odebrání IP adresy dříve připojenému zařízení. Nastane

tedy odebrání IP adresy a poté je možné adresu přidělit jinému zařízení, které se pokouší o připojení. Pokud se jiné zařízení bude pokoušet o připojení do sítě kde, již jiné zařízení připojené je, jednoduše mu nebude přidělena IP adresa a nebude mu tak umožněno připojení.

Důvod použití pouze jedné IP adresy, která je zařízením nastavována je následující. Za předpokladu, že se připojí zařízení na hotspot a je mu přiřazena IP adresa, která není jasně definovaná, nastane chvíle, kdy máme na Raspberry Pi připojené zařízení a neznáme jeho IP adresu, která se může pohybovat v rozmezí 1-255. IP adresu právě připojeného zařízení je možné zjistit například následujícími způsoby.

```
import socket
import os
import nmap
from subprocess import check_output

# 1. zpusob
"""
nm = nmap.PortScanner()
nm.scan('192.168.43.0/24', arguments='-sn')
print(nm.all_hosts())

"""

# 2. zpusob
nm = nmap.PortScanner()
get_IP = check_output(['hostname','-I']).decode()
get_IP = get_IP.split(" ")[1]
IP_base = get_IP.split(".")[:-1]
IP_target = IP_base[0] + "." + IP_base[1] + "." + IP_base[2] + ".0/24"
print(IP_target)

nm.scan(IP_target, arguments='-sn')
print(nm.all_hosts())
```

Při testování skenování všech připojených zařízení, nebo určitého počtu zařízení, docházelo k velkým časovým prodlevám. Z toho důvodu je povoleno připojení pouze jednomu zařízení a je mu přiřazena vždy stejná IP adresa.

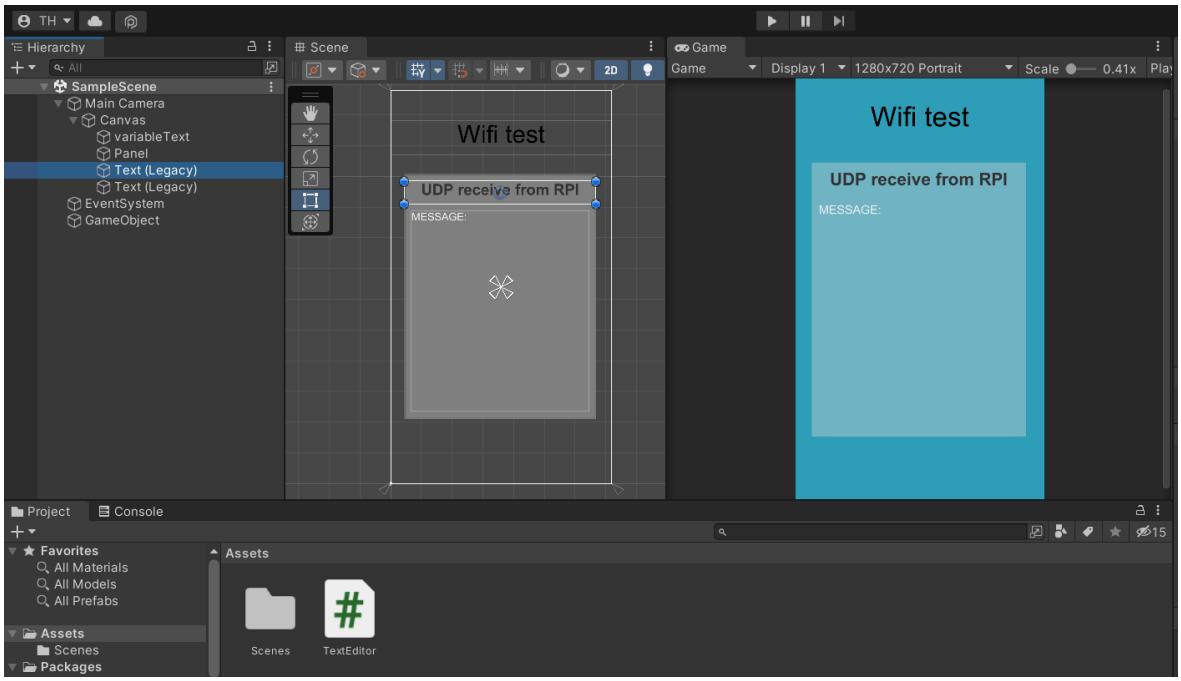
Principiálně funguje komunikace tak, že Raspberry Pi je spuštěno spolu s Python skriptem pro detekci a je vytvořený hotspot na který je možné se připojit pomocí nastaveného hesla. Po samotném připojení telefonu na hotspot nedochází k odesílání žádných dat. K odesílání dat dochází až v okamžiku spuštění Unity aplikace na mobilním telefonu. Unity aplikace totiž po svém spuštění začne odesílat zprávy na Raspberry Pi, kterému tak dává informaci, že je telefon připravený na příjem dat. Raspberry Pi obdrží zprávu od Unity aplikace a začne odesílat souřadnice herních kostek na již známou IP adresu. Jakmile je Unity aplikace

na telefonu ukončena, je přerušeno odesílání zpráv na Raspberry Pi a zastaví se tak komunikace. Komunikace bude obnovena při opětovném zapnutí Unity aplikace na telefonu.

12 UKÁZKA DETEKCE OBJEKTŮ A KOMUNIKACE

Cílem ukázky je předvést detekci objektů pomocí Raspberry PI s kamerou, kde souřadnice detekovaných objektů jsou z Raspberry PI odesílány a na mobilním telefonu přijímány a zobrazovány.

Ukázková aplikace, běžící na mobilním telefonu nebo tabletu, je implementována pomocí frameworku Unity. Pro příjem a zobrazení souřadnic bylo vytvořeno základní grafické prostředí, které na displeji telefonu vypisuje data. Grafické rozhraní je tvořeno z nadpisu, podnadpisu a variabilního textu, který se mění na základě přijímaných dat.



Obrázek 47: Vytvořené grafické prostředí pro telefon a příjem dat

Vytvoření scény, která je viditelná na obrázku výše, předcházelo vytvoření C# kódu pro příjem dat a jeho napojení na danou scénu, přičemž C# kód vypisuje aktuální souřadnice kostek na displej. Tato data jsou vypisována jako seznam záznamů, obsahujících vždy tři informace: identifikační číslo, souřadnici na ose x a souřadnici na ose y.

Po otestování funkčnosti Unity aplikace na PC následovalo testování na Android telefonu. Vytvořená scéna byla vyexportovaná jako „.apk“ soubor. Vyexportovaný instalační soubor byl přenesen na telefon přes USB kabel a nainstalován. Aplikace po spuštění zobrazuje na displeji souřadnice. Unity aplikace je tedy nainstalována a je zajištěn příjem dat po jejím spuštění.

Tímto práce v Unity programu v této bakalářské práci končí. Další vývoj projektu bude probíhat na straně Fakulty multimediálních komunikací UTB ve Zlíně, kde kolega v rámci spolupráce na společném projektu IGA vytvoří grafické objekty vykreslované na základě přijímaných dat na displeji telefonu či tabletu pomocí souřadnic, při namíření kamery na herní desku.

13 AUTOMATIZACE

Vytvořený projekt je třeba upravit, aby jeho spuštění bylo snadné, rychlé a nevyžadovalo zásah uživatelem. Je potřeba zajistit, aby uživatel Raspberry Pi pouze připojil ke zdroji a následovalo jeho spuštění společně se skriptem pro detekci a vytvořilo hotspot, na který se bude připojovat zařízení. Zároveň by se RPI mělo vypnout po určité době nečinnosti. To znamená, že pokud se na RPI hotspot nepřipojí po určitou dobu nějaké zařízení, dojde k ukončení veškeré činnosti a následnému vypnutí. Po úplném vypnutí je možné RPI opět zapnout vytažením a znovuzapojením napájecího kabelu do zásuvky.

Automatické zapnutí vytvořeného Python skriptu lze docílit konfigurací souboru „rc.local“. Do souboru přidáme řádek „python3 /home/pi/utb/Arducam.py &“, který obsahuje cestu ke skriptu a spustí ho po zapnutí RPI. Ampersand na konci přidaného řádku umožňuje příkaz spustit v nezávislém procesu a může tak pokračovat bootování s již běžícím procesem. Skript je navíc opožděn příkazem „sleep 60“, který je vložený před samotné spuštění skriptu a spuštění opozdí o 60 vteřin. Takovéto opoždění slouží k úplnému spuštění RPI.

Jakmile je skript po zapnutí RPI spuštěn, čeká na zprávu od telefonu, který informuje, že je Unity aplikace zapnutá a je možné na něj začít posílat souřadnice. Raspberry Pi čeká na zprávu 360 vteřin. Po uplynutí tohoto časového intervalu předpokládáme, že se k RPI nepřipojí již žádné zařízení a je tedy veškerá jeho činnost ukončena. Pro vypnutí Raspberry Pi je na konci skriptu volán příkaz „sudo shutdown -h now“ pomocí funkce „call“.

Takováto konfigurace nám umožňuje pracovat s projektem bez jakékoliv nutnosti zasahovat do Raspberry Pi a usnadňuje práci s ním. V případě nutnosti jakýchkoliv úprav je možné Raspberry Pi napojit na notebook přes síťový kabel a opět pomocí VNC vieweru zobrazit jeho plochu, přes kterou můžeme uživatel pomocí terminálu spuštěný Python skript ukončit příkazem „sudo pkill python“. K ukončení Python skriptu je ale nutné, aby byl samotný skript již spuštěný, protože není možné ukončit něco, co ještě neběží. Proto při zadávání příkazu pro ukončení skriptu musí uživatel vyčkat, dokud se skript nezapne, což může sledovat v terminálu pomocí příkazu „htop“. Jakmile je spuštěný Python kód ukončen uživatelem, je možné RPI využívat opět běžným způsobem. Ovšem po vypnutí a opětovném zapnutí RPI se Python kód znova po chvíli spustí. Chce-li uživatel RPI využívat například pro úpravu kódu, může v souboru „rc.local“ zakomentovat řádek, který Python kód po spuštění RPI spouští.

ZÁVĚR

Výstupem bakalářské práce je software aplikace napsaná v jazyce Python pro Raspberry Pi, který za pomocí kamery snímá herní desku spolu s herními kostkami a souřadnice získané vůči hrací ploše desky odesílá na připojené zařízení. V ukázkové aplikaci využíváme 8 různých unikátních identifikátorů (vizuálních kódů). Finální aplikace na chytrém zařízení je implementována ve frameworku Unity a vykresluje 3D grafický tvar rozšířené reality do obrazu z kamery mobilního zařízení nad jednotlivé detekované herní kostky.

Na úvod teoretické části došlo k návrhu hardware architektury, která se skládá z Raspberry Pi, kamery a chytrého zařízení a herní desky společně s herními kostkami. Jednotlivé HW komponenty byly v této teoretické části podrobněji popsány spolu se základními parametry. Teoretická část bakalářské práce se dále v kapitole s názvem „Vizuální kódy“ zabývala druhy vizuálních kódů a jejich stručným popisem. Kódy byly rozděleny na jednodimenzionální a dvoudimenzionální spolu s dalším rozdělením a konkrétními příklady a ukázkami kódů. Sekce „Vývojové prostředí“ popisovala použité prostředky pro vývoj aplikace. Následně byl čtenáři přiblížen způsob komunikace mezi dvěma zařízeními, konkrétně typy Bluetooth a Wi-Fi, kde byly popsány základní protokoly obou typů komunikací. Poslední část teoretické části nesoucí název „Zpracování dat“ uvedla čtenáře do použitých prostředků při zpracovávání dat přijímaných na chytrém zařízení.

Praktická část bakalářské práce obsahuje na úvod rekapitulaci funkčnosti projektu a jeho základní popis. V praktické části je na prvním místě popsána příprava hardware, která obsahovala přípravu samotného Raspberry Pi pro zajištění základní funkčnosti. Následoval výběr jedné ze tří dostupných kamer pro projekt. Vybraná kamera byla zkalibrována a byl tak zajištěn nezdeformovaný obraz, určený pro následné zpracovaní. Po kalibraci kamery následoval výběr vhodného značení a testování, které mělo za úkol snadnou identifikaci typu kostky a její polohy v herní ploše. Prvními kroky při zpracování obrazu byly výřez části obrazu, ze kterého jsme dále získali detekovatelné objekty. Takovýto způsob výřezu byl taktéž vybírán z více způsobů. Vzhledem k nespolehlivým výsledkům ostatních testovaných kódů došlo k výběru výřezu pomocí ArUco značení. Jakmile byl způsob výřezu vytvořen, unikátní herní kostky byly podrobeny testování, které mělo za úkol odhalit nejspolehlivější značení, detekovatelné i ve zhoršených světelných podmínkách. Výsledkem tohoto testování byl výběr ArUco značení, které se i v tomto případě ukázalo jako nejspolehlivější ze všech testovaných typů vizuálních kódů. Dále se praktická část bakalářské práce po zvoleném

způsobu značení ubírá ke komunikaci mezi dvěma zařízeními. Pro komunikaci bylo z patřičných důvodů uvedených v této části využito Wi-Fi komunikace pomocí UDP. Následovala základní práce v Unity, která zajistila v projektu příjem dat na Android telefonu a výpis přijatých dat na displeji telefonu. Poslední část popisuje způsob automatizace spuštění Raspberry Pi, která zajistí uživateli snadnou manipulaci s RPI a není tak nutné, aby uživatel jakkoliv zasahoval do kódu, či snad konfiguroval nastavení.

Cílová skupina uživatelů výsledného produktu našeho IGA projektu je spíše nižší věková kategorie dětí, které takovýto projekt mohou využít pro hraní nebo vzdělávání se. Takovýto uživatel může po hrací ploše libovolně přesouvat kostky a utvářet tak různé tvary díky spojování a umisťování kostek do vzájemné blízkosti. Uživatel může tak rozvíjet svou kreativitu a zároveň si rozšíří i povědomí o virtuálním světě. Projekt mohou zajisté využívat i uživatelé vyššího věku, kterým aplikace může taktéž pomoci, například při vytváření jednoduché mapy pro hru, nebo jen při touze si zahrát a vytvořit něco kreativního.

SEZNAM POUŽITÉ LITERATURY

[1] Raspberry Pi kamera V2. *RPishop.cz* [online]. Copyright © Copyright 2023 RPishop.cz. [cit. 02.04.2023]. Dostupné z: https://rpishop.cz/mipi-kamerove-moduly/329-raspberry-pi-kamera-modul-v2.html

[2] Waveshare IMX378-190 Fisheye kamera pro Raspberry Pi. *RPishop.cz* [online]. Copyright © Copyright 2023 RPishop.cz. [cit. 02.04.2023]. Dostupné z: https://rpishop.cz/mipi-kamerove-moduly/5031-waveshare-imx378-190-fisheye-kamera-pro-raspberry-pi.html

[3] Arducam 12Mpx IMX477 kamerový modul. *RPishop.cz* [online]. Copyright © Copyright 2023 RPishop.cz. [cit. 02.04.2023]. Dostupné z: https://rpishop.cz/mipi-kamerove-moduly/3386-arducam-12mpx-imx477-kamerovy-modul.html

[4] ŠOUSTEK, Petr; MATOUŠEK, Radek. *Moderní čárové kódy* [online]. [cit. 2.4.2023]. Dostupný na WWW: https://automa.cz/cz/casopis-clanky/moderni-carove-kody-2012_05_0_9585/

[5] Čárové kódy (teorie). *Gaben.cz* [online]. Copyright © 2016 GABEN, spol. s r.o. [cit. 02.04.2023]. Dostupné z: https://www.gaben.cz/cz/faq/carove-kody-teorie

[6] ESP holding a.s. Čárový kód – vše, co potřebujete vědět. *Redirecting to https://esp.cz/cs* [online]. Copyright © 2011 [cit. 02.04.2023]. Dostupné z: https://esp.cz/cs/blog/carovy-kod-vse-potrebujete-vedet-moderni-automaticke-identifikaci

[7] Codeware s.r.o. Lineární čárové kódy - Čárový kód.info. *Portál o technologii čárového kódu - Čárový kód.info* [online]. [cit. 02.04.2023]. Dostupné z: https://www.carovy-kod.info/carovy-kod/linearni-carove-kody_213.html

[8] Circular Barcode. *TechnoRiver* [online]. Copyright © 2004 [cit. 02.04.2023]. Dostupné z: https://www.technoriversoft.com/CircularBarcode.html

[9] QR Code History: Story of evolution of the very popular 2D Barcode. *QR Code Solutions | QR Code Generator | Customized QR Code | Scanova* [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://scanova.io/blog/qr-code-history/

[10] Aztec Code. [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://barcode-guide.seagullscientific.com/content/Symbologies/Aztec_Code.htm

[11] Data Matrix. [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://barcode-guide.seagullscientific.com/Content/Symbologies/Data_Matrix.htm?Highlight=Data%20matrix

[12] Home — Spyder IDE. *Home — Spyder IDE* [online]. Copyright © 2022 Spyder Website Contributors [cit. 02.04.2023]. Dostupné z: https://www.spyder-ide.org/

[13] About - OpenCV. *Home - OpenCV* [online]. Dostupné z: https://opencv.org/about/

[14] Mathematical functions — Python 3.11.2 documentation. *python.org* [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://docs.python.org/3/library/math.html

[15] Socket — Low-level networking interface — Python 3.11.2 documentation. *python.org* [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://docs.python.org/3/library/socket.html

[16] NumPy documentation — NumPy v1.24 Manual. *NumPy* [online]. Copyright © Copyright 2008 [cit. 02.04.2023]. Dostupné z: https://numpy.org/doc/stable/

[17] Subprocess management — Python 3.11.2 documentation. *python.org* [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://docs.python.org/3/library/subprocess.html

[18] *Raspberry Pi Datasheets* [online]. Copyright © [cit. 02.04.2023]. Dostupné z: https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf

[19] HUANG, Albert S. a Larry RUDOLPH. *Bluetooth essentials for programmers*. New York, NY: Cambridge University Press, 2007. ISBN 978-0-521-70375-8.

[20] Internet Core Protocols: The Definitive Guide: Help for Network Administrators - Eric Hall - Knihy Google. *Knihy Google* [online]. Dostupné z: https://books.google.cz/books/about/Internet_Core_Protocols_The_Definitive_G.html?id=kdtFBAAAQBAJ&redir_esc=y

[21] Professional Visual Studio Extensibility - Keyvan Nayyeri - Knihy Google. *Knihy Google* [online]. Dostupné z: https://books.google.cz/books?id=rlilUH8kAh0C&printsec=frontcover&hl=cs#v=onepage&q&f=false

[22] VIRIUS, Miroslav. *Programování v C#: od základů k profesionálnímu použití*. Praha: Grada Publishing, 2021. Knihovna programátora (Grada). ISBN 978-80-271-1216-6.

[23] Co je rozšířená realita (AR) | Microsoft Dynamics 365. *Object moved* [online]. Dostupné z: https://dynamics.microsoft.com/cs-cz/mixed-reality/guides/what-is-augmented-reality-ar/

[24] AR technology for Android — Part 1 | by Loredana Zdrânc | Zipper Studios | Medium. *Medium – Where good ideas find you.* [online]. Dostupné z: https://medium.com/zipper-studios/ar-technology-for-android-part-1-85ff45114620

[25] Raspberry Pi 4 Model B - 8GB RAM. RPishop.cz [online]. Copyright © [cit. 05.05.2023]. Dostupné z: https://rpishop.cz/raspberry-pi-4/2611-raspberry-pi-4-model-b-8gb-ram-0765756931199.html

[26] KOTEK, Kamil. Kalibrační proces ve 3D. fm.tul.cz [online]. [cit. 5.5.2023]. Dostupný na WWW: https://www.fm.tul.cz/files/pages/other/MTI/obr15/obr15_fcc_kotek.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RPI	Raspberry Pi
VNC	Virtual Network Computing
EAN	European Article Number
QR	Quick Response
CV	Computer Vision
RFCOMM	Radio Frequency Communication
L2CAP	Logical Link Control and Adaptation Protocol
OBEX	Object Exchange
UDP	User Datagram Protocol
TCP	Transmission Control Protocol

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD

Přiložené CD obsahuje:

- Soubor **fulltext** – text bakalářské práce ve formátu PDF/A
- Adresář **Prilohy**
    - Soubor **Arducam.py** – skript pro Raspberry Pi
    - Soubor **calibrate.py** – skript pro kalibraci kamery
    - Soubor **unityApp.cs** – skript pro unity aplikaci