Uniwersytet Wrocławski
Wydział Fizyki i Astronomii

Krzysztof Kukiz

Inteligentne powitanie na podstawie rozpoznawania twarzy

Smart greeting based on face recognition

Praca inżynierska na kierunku
Informatyka Stosowana i Systemy Pomiarowe

Opiekun
dr hab. Maciej Matyka, prof. UWr

Wrocław, 26 maja 2022

Spis treści

Streszczenie

Niniejsza praca przedstawia projekt systemu inteligentnego rozpoznawania osób oraz podejmowania przez system wcześniej zdefiniowanych działań, w zależności od rozpoznanej osoby. W przypadku braku rozpoznania osoby, system poda także odpowiedni komunikat głosowy oraz wizualny.

Projekt oparty jest na Raspberry Pi 4B, na języku programowania python 3, oraz wykorzystuje elementy wydrukowane na drukarce 3D Creality Ender-3 v.2.

Projekt łączy ze sobą w całość 3 warstwy niezbędne do wykonania wszystkich założonych zadań w taki sposób aby w przyszłości można było rozbudować system o kolejne funkcjonalności: warstwę sprzętową (kamery, mikro-komputera Raspberry Pi, głośnika, wyświetlacza), warstwę programistyczną odpowiedzialną za obróbkę oraz optymalizację odczytanego obrazu, interpretację pobranego obrazu oraz porównanie go z wcześniej zdefiniowaną bazą zdjęć, podjęcie decyzji o tym jakie działanie ma być podjęte oraz wygenerowanie właściwego sygnału skutkującego podjęciem określonych, warstwę produktową w postaci dedykowanej, zaprojektowanej specjalnie dla tego projektu obudowy będącej jednocześnie opakowaniem produktu, pełniącej jego funkcje organizacyjną, ochronną, informacyjną.

Głównym celem projektu było stworzenie systemu mobilnego, o jak najmniejszych wymiarach, który jest gotowy do działania natychmiast po podłączeniu zasilania.

Abstract

This work presents the design of a system for the intelligent recognition of persons and for the system to take predefined actions, depending on the recognised person. If the person is not recognised, the system will also give an appropriate voice and visual message.

The project is based on a Raspberry Pi 4B, the python 3 programming language, and uses components printed on a Creality Ender-3 v.2 3D printer.

The project brings together the 3 layers needed to carry out all the tasks in such a way that the system can be extended in the future with further functionalities: A hardware layer (the camera, the Rasppery Pi micro-computer, a speaker, a display), the programming layer responsible for processing and optimisation of the read image, interpretation of the retrieved image and comparison with a predefined image database, deciding on the action to be taken and generating the right signal to take the specified action, a product layer in the form of a dedicated enclosure (Etui), designed specifically for this project, performing its organisational, protective, informative functions.

The main goal of the project was to create a mobile system, with the smallest possible dimensions, which is ready to operate immediately after connecting the power supply.

1 Wstęp

1.1 Wprowadzenie

Systemy inteligentne są stosowane obecnie na całym świecie. Znajdują coraz szersze zastosowanie w życiu codziennym każdego z nas zarówno w pracy jak i w domu. Potrafią reagować na nasze zachowania, odczytywać intencje oraz gromadzić i wykorzystywać dane, które im udostępnimy zarówno na poziomie jednostki, jak i całych grup społecznych.

W naszym codziennym życiu są elementem składowym inteligentnych budynków opartych między innymi o systemy LOXONE, KNX, AMPIO, czy też FIBARO[1] i znacznie ułatwiają codzienne czynności poprzez sterowanie klimatem, ogrzewaniem, rekuperacją, czy też podlewaniem ogrodów, sprzątaniem, a nawet optymalnym wykorzystaniem energii elektrycznej.

Systemy inteligentne pomimo swoich kosztów zdobywają coraz większą popularność nie tylko w zastosowaniach profesjonalnych, ale także w zastosowaniach prywatnych i mogą być oparte zarówno o infrastrukturę przewodową (patrz rys. 1), jak i mogą być oparte na rozwiązaniach bezprzewodowych.

Rysunek 1: Przykład okablowania domu inteligentnego [13].

Żaden system nie posiada jednak wszystkich funkcjonalności, które są niezbędne danemu użytkownikowi. W większości z nich brakuje mobilnego systemu pozwalającego na rozpoznanie osób oraz podjęcia konkretnych działań w zależności od osoby, która została zidentyfikowana.

[1]LOXONE, KNX, AMPIO, FIBARO są to nazwy producentów oferujących rozwiązania stosowane w budowie domów inteligentnych

System inteligentnego rozpoznawania osób, który jest tematem niniejszego projektu, docelowo ma umożliwić identyfikację osób wchodzących do pomieszczenia na podstawie zapisanej wcześniej bazy zdjęć oraz powitanie ich indywidualnym komunikatem zarówno głosowym, jak i wizualnym.

Do realizacji projektu wykorzystałem mikro-komputer Raspperry Pi 4B oraz język programowania Python 3. Jako elementu wejściowego sygnału użyłem kamery Raspberry Pi Camera HD v2 8MPx zgodnej z Raspperry Pi 4B, oraz głośnika pod wejscie jack 3,5mm.

1.2 Cel i zakres pracy

Celem głównym projektu jest stworzenie prototypu inteligentnego, mobilnego systemu rozpoznawania osób, na podstawie wcześniej zgromadzonej bazy zdjęciowej, a także wygenerowanie unikatowego powitania głosowego oraz wizualnego w zależności od rozpoznanej osoby.

Celami szczegółowymi projektu są: wybór technologii sprzętowej, wybór technologii programistycznej oraz zaprojektowanie i wykonanie obudowy prototypu, mieszczącej wszystkie elementy projektu.

Praca swoim zakresem obejmuje zarówno tematykę z dziedziny elektroniki, informatyki projektowania oraz prototypowania 3D.

W zakresie elektroniki wykorzystane zostały układy i sensory zgodne z płytką Rasppery 4B, w zakresie informatyki napisany został program w języku Python 3 i wykorzystujący do realizacji zadań dostępne biblioteki, natomiast w zakresie prototypowania z wykorzystaniem druku 3D, użyta została drukarka 3D Creality Ender v2 oraz materiał EASY PET-G. Do celów zaprojektowania obudowy wykorzystano program SOLIDWORKS.

Projekt z jednej strony pokazuje, że połączenie wiedzy z zakresu elektroniki, informatyki oraz projektowania dostarcza każdemu studentowi odpowiednią wiedzę oraz umiejętności, do wykonania prototypu od fazy projektowej do fazy praktycznego zastosowania, z drugiej udowadnia, że wiedza ta pozwala na wykonania projektu z wykorzystaniem elementów ogólnodostępnych na rynku.

1.3 Struktura pracy

Rozdział pierwszy zawiera wstęp, wprowadzenie, opis celu oraz zakresu pracy, a także strukturę pracy. Znajdują się w nim podstawowe informacje opisujące główne elementy składowe projektu, oraz umożliwia szybkie zapoznania się z projektem na poziomie ogólnym.

Rozdział drugi zawiera założenia odnośnie szczegółowych wymagań stawianych projektowi i umożliwia zrozumienie koncepcji projektu zarówno na poziomie ogólnym i bardziej szczegółowym.

Rozdział trzeci zawiera zestawienie założone funkcjonalności projektu, które będą zaimplementowane w projektowanym systemie oraz sposób korzystania z tych funkcjonalności.

Rozdział czwarty zawiera szczegółowy opis warstwy sprzętowej z wyszczególnieniem użytych w projekcie podzespołów oraz ich parametrów. Znajdują się w nim szczegółowe

opisy użytej do projektu kamery, mikro-komputera Raspberry Pi, głośnika oraz ekranu służącego do komunikacji z użytkownikiem.

W rozdziale piątym znajduje się opis warstwy programistycznej projektu z uzasadnieniem wyboru języka Python jako języka programowania użytego w projekcie, opis zastosowanych bibliotek, algorytm oraz główne elementy kodu odpowiedzialne za działanie całego projektu.

W rozdziale szóstym znajduje się opis warstwy produktowej projektu z opisem użytej technologii druku 3D, kroków projektowania obudowy, procesu wykonania obudowy oraz szczegółów samej obudowy.

W rozdziale siódmym opisane zostały kwestie związane z samą realizacją projektu, napotkane problemy oraz sposoby jakimi problemy zostały rozwiązane. Omówiono także potencjalne możliwości rozbudowy projektu o kolejne funkcjonalności i zastosowania.

W rozdziale ósmym zawarto wnioski oraz „Lessons learned" z wykonanego projektu. Stanowią one podstawę do dalszego doskonalenia podejścia przy kolejnych projektach.

2 Wymagania stawiane projektowi

W celu realizacji pracy w pierwszej kolejności określone zostały wymagania, które stanowiły punkt docelowy projektu, jaki należało osiągnąć w ramach prac projektowych, a następnie pozwoliły na zaplanowanie drogi dojścia do założonego celu.

Na podstawie założonego celu zostały określone kroki milowe, które należało osiągnąć. Kroki te dzieliły cały projekt na mniejsze etapy, z jednej strony pozwalające na lepszy nadzór nad postępami pracy, oraz natychmiastowe określenie punktów stanowiących problem i wstrzymujących postępy, a z drugiej strony pozwalały na elastyczne reagowanie i zmiany projektowe bez konieczności zmian poprzednich etapów.

Takie podejście miało zapewnić iż ewentualny błąd popełniony na dalszych etapach nie będzie miał wpływu na etapy poprzednie projektu.

Rysunek 2: Etapy prac projektowych [13].

2.1 Wymagania dla prototypu

W ramach realizacji projektu określone zostały następujące wymagania dla prototypu.

- **Mobilność.** Głównym założeniem projektu jest mobilność prototypu. Oznacza to, iż będzie posiadał wymiary oraz wagę umożliwiającą łatwe przenoszenie w dowolne miejsce,
- **Łatwość zasilania.** Prototyp powinien mieć możliwość zasilania z ogólnodostępnej sieci 230V, tak aby istniała możliwość stałego podłączenia do gniazda elektrycznego. Opcjonalnie założono, że prototyp będzie posiadał możliwość podłączenia do zasilania z powerbanku, tak aby mógł znaleźć zastosowanie w miejscach, w których dostęp do sieci elektrycznej jest utrudniony,
- **Dostępność.** Prototyp zostanie wykonany ze standardowych elementów dostępnych na rynku polskim,
- **Relatywnie niski koszt.** Wykonanie prototypu powinno mieścić się w kwocie nie większej niż 1000 zł,
- **Łatwość obsługi.** Prototyp będzie umożliwiał obsługę bez konieczności znajomości zagadnień technicznych. Zadaniem użytkownika będzie jedynie wprowadzenie bazy zdjęć do określonego folderu, oraz uruchomienie dwóch programów:
  - program "zapisujący" zdjęcia twarzy do zakodowanego pliku;
  - główny program rozpoznający osoby przechodzące przed kamerą;

  i odpowiednie nakierowanie kamery prototypu,
- **Łatwość rozbudowy.** Platforma na jakiej zostanie zbudowany prototyp będzie pozwalała na jego łatwą rozbudowę,
- **Łatwość programowania.** System będzie oparty o jeden z popularnych języków programowania, w którym dostępne są biblioteki niezbędne do realizacji wszystkich zadań, tak aby użytkownik mógł skorzystać z dostępnych standardowych rozwiązań programistycznych, bez ponoszenia dodatkowych kosztów.

2.2 Wymagania dla procesu projektowego

W celu wykonania prototypu zostały określone następujące wymagania dla procesu projektowego.

- **Elastyczność.** Projekt zostanie podzielony na mniejsze etapy, tak aby w każdej chwili istniała możliwość modyfikacji założeń lub działań przy jednoczesnym zapewnieniu realizacji założonych celów,
- **Dostępność narzędzi.** Dla każdego etapu projektu zostaną użyte narzędzia dostępne na rynku, tak aby uniknąć konieczności projektowania samych narzędzi,

- **Przejrzystość.** Na każdym etapie status prac będzie łatwo weryfikowalny aby zidentyfikować potencjalne przeszkody mogące powodować opóźnienie realizacji projektu,
- **Backup.** W celu uniknięcia utraty danych lub kodu źródłowego zostanie wykorzystane repozytorium Git. Wersja dostępna online znajduje się na Githubie [12].

3 Oczekiwane funkcjonalności projektu

W ramach realizacji projektu zostały zaimplementowane następujące funkcjonalności.

- **Zapis zdjęć twarzy** do postaci numerycznej. Użytkownik do imiennych folderów wprowadza zdjęcia twarzy osoby (np. zdjęcia Jana wrzucam do folderu "Jan Nowak") i uruchamia program kodujący zamieniając zdjęcie twarzy na wartości liczbowe,
- **Użycie syntezatora mowy.** System korzysta z syntezatora mowy w celu dodania domyślnego powitania do pliku .mp3,
- **Wykrycie osoby.** System skanuje wybrany obszar przestrzeni i wykrywa czy w nadzorowanym obszarze znajduje się osoba. W przypadku wykrycia aktywowane są kolejne kroki programu. System pomija zwierzęta oraz przedmioty,
- **Skanowanie twarzy.** Po wykryciu obiektu kamera koncentruje się na twarzy wykrytej osoby i koduje twarz w celu porównania z lokalną bazą,
- **Porównanie (Identyfikacja) twarzy.** System porównuje zakodowane zdjęcia z wcześniej zakodowanymi zdjęciami znajdującymi się w bazie danych. System przekształca dane wejściowe do macierzy liczb. Funkcja z biblioteki porównuje wszystkie macierze i patrzy czy dana osoba istnieje,
- **Komunikat głosowy.** Z puli powitań plików .mp3 (domyślnie jeden, można wrzucić własne dzwięki), program losuje jeden i go odpala (w przypadku gdy jest tylko jeden plik, losowanie się nie odbywa),
- **Komunikat wizualny.** Na ekranie dodatkowo pojawia się komunikat graficzny z imieniem i nazwiskiem (Patrz rys. 3a). W przypadku wykrycia osoby nieznanej pojawia się stosowny komunikat (Patrz rys. 3b).

(a) Rozpoznanej osoby [13].

(b) NIE rozpoznanej osoby [13].

Rysunek 3: Komunikaty graficzne

Powyższe funkcjonalności pozwalają na realizację funkcji

- powitania,

- nadzór obszaru.

Przykłady realizacji funkcji powitania:

- "urządzenie rozpoznało Jana, który ma 3 wersje powitania. Program wylosuje i powie jedno z:"
  - "Cześć Jan, miło Cię widzieć",
  - "Witaj w domu Janek",
  - "Panie Janie, Panie Janie ..."
- "urządzenie rozpoznało Adama Nowaka, który ma tylko podstawowe pytanie. Program bez losowania odrazu je wybierze i powie:"
  - "Witaj Adam Nowak"
- dla nierozpoznanej osoby możemy mieć tylko domyślne przywitanie, ale możemy dać ich więcej. Program zachowa się analogicznie do wcześniej wymienionych przypadków:
  - "Witaj nieznajomy"
  - "Witamy w domostwie Nowaków"

4 Warstwa sprzętowa

W ramach warstwy sprzętowej projekt składa się z 4 elementów. Kamery, mikro-komputera Raspberry Pi, głośnika oraz ekranu.

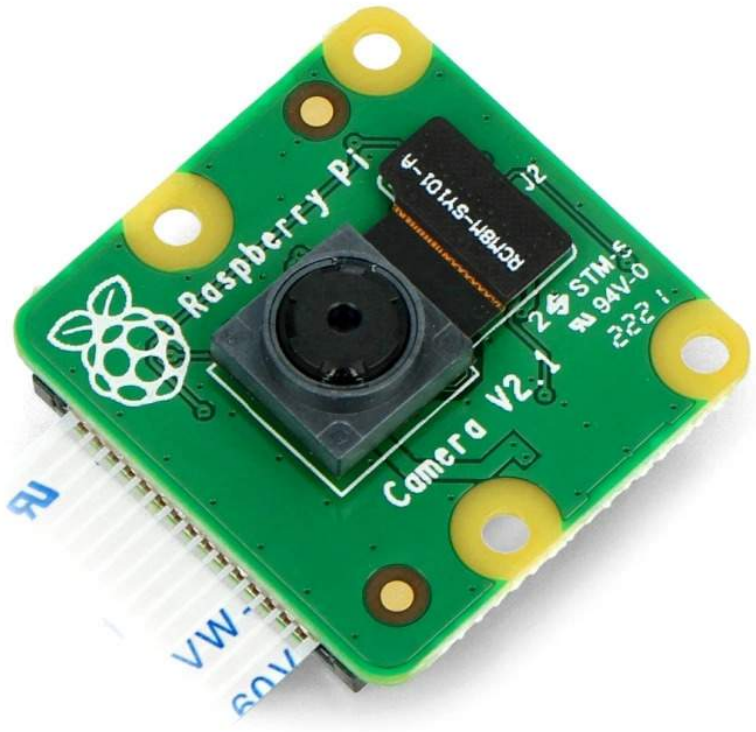
Podstawą systemu jest Raspberry Pi, do którego podłączone są wszystkie elementy wejściowe oraz wyjściowe prototypu, tak aby stanowiły spójną całość. W ten sposób przygotowana warstwa sprzętowa będzie następnie oprogramowana (patrz rozdział 5) oraz zabezpieczona obudową pełniącą jednocześnie funkcje opakowania (patrz rozdział 6).

4.1 Kamera jako element wejściowy

Spośród dostępnych kamer w projekcie zastosowano kamerę HD v2 8MPx dedykowaną do mikro-komputera Raspberry Pi. Urządzenie posiada matrycę o rozdzielczości 8 Mpx, wspiera tryb HD 1080p, 720p oraz 640 x 480p oraz jest w stanie wykonywać zdjęcia w wyższej rozdzielczości 3280 x 2464 px.

Aby uniknąć potencjalnych problemów w przesyle sygnału zastosowano oficjalny moduł Raspberry Pi Camera HD v2, który jest całkowicie zgodny ze sterownikami, dostarczanymi wraz z systemem Raspberry Pi OS.

Dzięki takiemu rozwiązaniu możliwe jest jednocześnie szerokie wsparcie w zakresie dokumentacji, tutoriali oraz bibliotek programistycznych.



Rysunek 4: Kamera HD [3].

Parametry kamery

- 1080p / 30 fps,
- 720p / 60 fps,
- 640 x 480p / 90 fps.

4.2 Płyta Raspberry Pi

Spośród dostępnych modeli Raspberry Pi do projektu wykorzystano model w wersji 4B, który charakteryzuje się zwiększoną pojemnością pamięci RAP (4GB) oraz wydajniejszym procesorem (Broadcom BCM2711) quad-core 64-bitowy ARM-8.

Posiada on dwa złącza microHDMI, dwa złączami USB 3.0 i 2 złącza USB 2.0. Zasilanie realizowane jest przez złącze USB C.

Mikro-komputer wyposażony jest także w dwuzakresowe WiFi 2,4 GHz i 5 GHz, Bluetooth w standardzie 5, oraz port Ethernet o prędkości do 1000 Mb/s.

Rysunek 5: Raspberry Pi 4B [4].

Specyfikacja techniczna

- Pamięć RAM - 1GB, 2GB lub 4GB LPDDR4,
- Procesor - Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz,
- 4 × USB,
- mini jack 3.5mm,
- Karta pamięci - microSD,
- Systemy operacyjne - Linux Raspbian, Windows 10 loT i wiele innych,
- Temperatura pracy - 0–50°C,
- Zasilanie - złącze USB C minimum 5V/3A, złącza GPIO minimum 5V/3A PoE przy zastosowaniu dedykowanej nakładki,
- Wymiary - 85 x 56 x 17 mm.

4.3 Głośnik, jako element wyjściowy

W zakresie komunikatów głosowych w projekcie został zastosowany głośnik LOGIC CONCEPT LS-09.

Logic Concept LS-09, to zestaw dwóch głośników, które można podłączyć zarówno do komputerów stacjonarnych, laptopów, jak i różnego rodzaju urządzeń multimedialnych. Głośniki charakteryzują się niewielkimi rozmiarami, co było jednym z warunków niezbędnych do realizacji projektu.

Każdy głośnik o mocy 3W. Dzięki niewielkim rozmiarom oraz lekkiej konstrukcji w łatwy i wygodny sposób można je przetransportować.

Zasilane realizowane jest poprzez kabel USB podłączany do Raspberry Pi.

Rysunek 6: Głośniki LOGIC CONCEPT LS-09 [13].

4.4 Ekran, jako element wyjściowy

W projekcie został zastosowany ekran Ekran dotykowy Waveshare 9904. Jest to ekran rezystancyjny LCD TFT o przekątnej 3,5” i rozdzielczości 320x480px kompatybilny z Raspberry Pi 4/3/2/B+/Zero.

Oprogramowanie ekranu dotykowego przygotowane przez producenta zawiera sterowniki.

Rysunek 7: Ekran Waveshare 9904 [2].

Specyfikacja ekranu dotykowego Waveshare

- Typ: ekran dotykowy, rezystancyjny,
- Przekątna: 3.5",
- Rozdzielczość: 320px x 480px,
- Komunikacja SPI,
- Współpracuje z: Raspberry Pi w wersji 4B, 3B+, 3B, 2B, B+, Zero i Zero,
- Wymiary ekranu: 85mm x 56.5mm.

4.5 Radiatory

W celu odprowadzenia ciepła z podzespołów został zastosowany zestaw radiatorów do Raspberry Pi - z taśmą termoprzewodzącą.

Radiatory zbudowane są ze stopów metali dobrze przewodzących ciepło. Ich powierzchnia od strony zewnętrznej jest rozwinięta w postaci żeber.

5 Warstwa programistyczna

5.1 Python, jako język programowania

W celu realizacji funkcji prototypu zostały wzięte pod uwagę języki C++ oraz Python. Po wstępnej analizie został wybrany język Python ze względu na wsparcie producenta Raspberry Pi [14], większy wybór bibliotek [10] oraz łatwiejszy do pisania i debugowania kod [15]. Jednocześnie język programowania Python charakteryzuje się wolniejszym działaniem w porównaniu do języka c++, co w moim przypadku nie jest problemem, gdyż najbardziej wymagająca biblioteka jest napisana pod pythona właśnie w języku c++. Dzięki czemu nie tracę na szybkości.

Z tego powodu biorąc pod uwagę przede wszystkim:

- większy wybór bibliotek,
- wsparcie Raspberry Pi,
- przejrzystość kodu,

to język Python stanowi lepszy wybór.

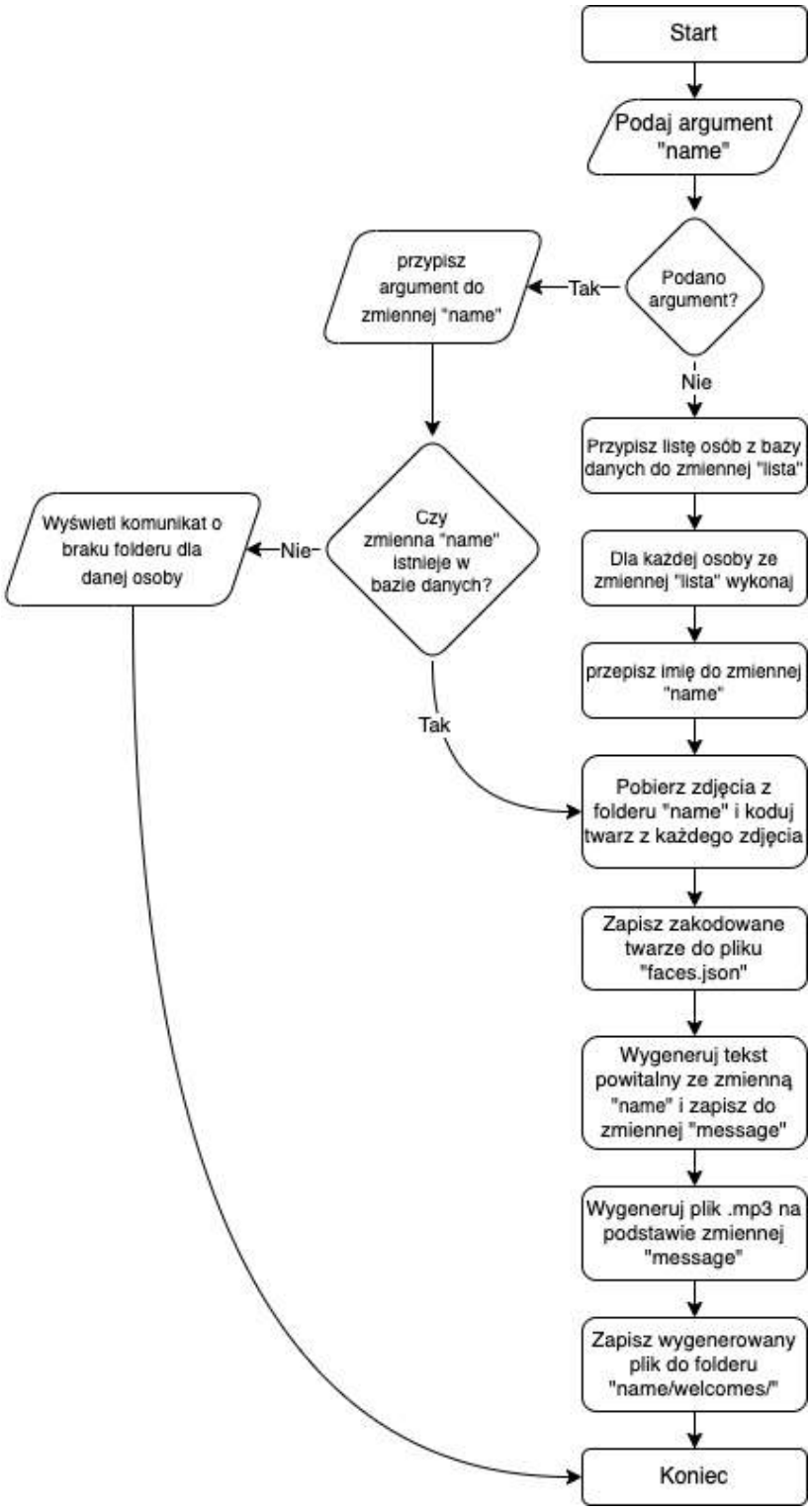
5.2 Biblioteki

W celu realizacji poszczególnych zadań zostały użyte następujące biblioteki języka Python:

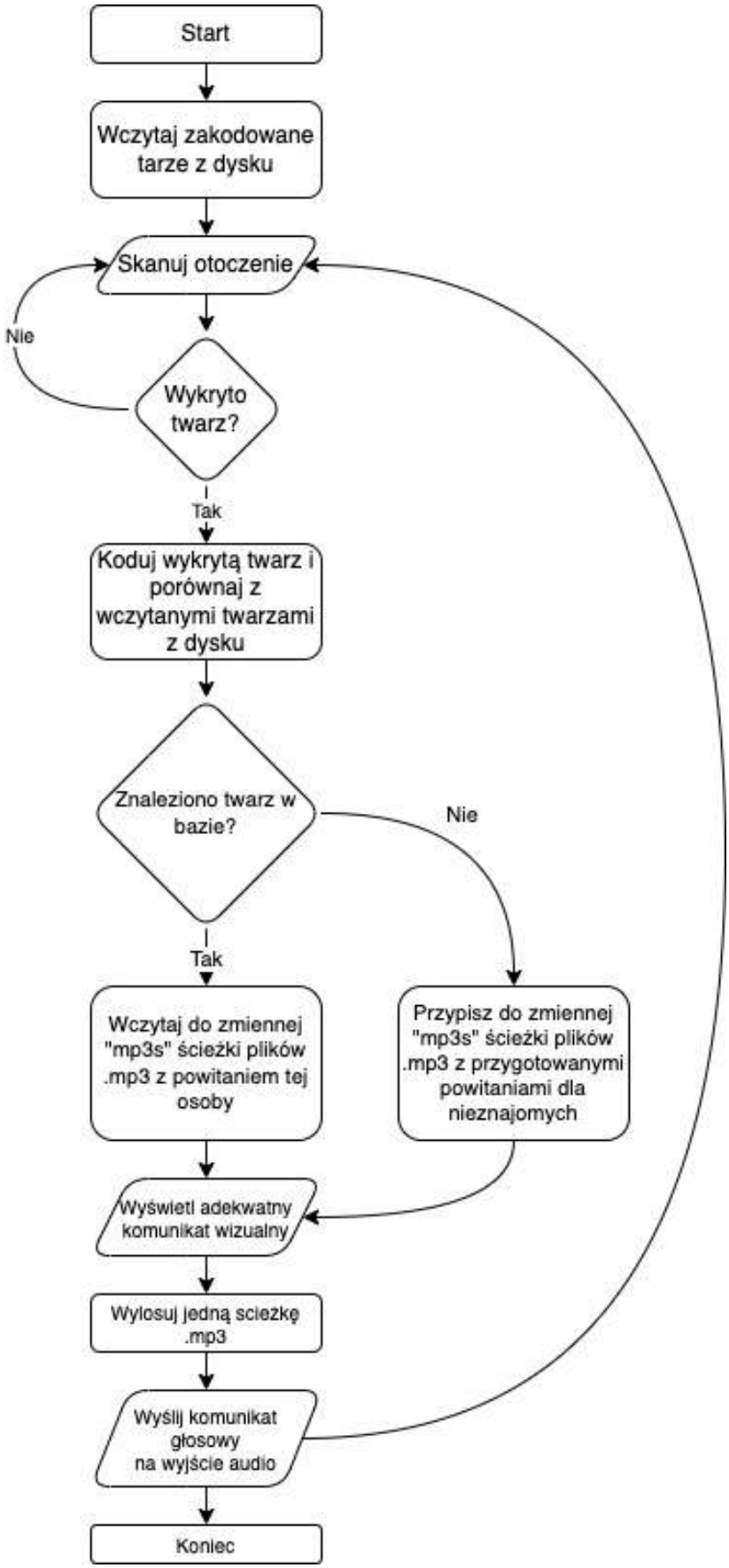
- cv2, odpowiadająca za pobieranie obrazu z kamerki,
- face_recognition, odpowiadająca za rozpoznawanie twarzy [5] [8] [9],
- gtts, odpowiadająca za stworzenie deflautowego powitania do pliku .mp3,
- json, odpowiadająca za zapisywanie i odczytywanie zakodowanych twarzy do plików,
- numpy, odpowiadająca za tworzenie zoptymalizowanych pod bibliotekę rozpoznawania twarzy macierzy,
- os, odpowiadająca za poruszanie się po folderach,
- pygame, odpowiadająca za odtworzenie powitalnego dźwięku,
- random, odpowiadająca za wybranie losowej ścieżki do otworzenia,
- sys, odpowiadająca za pobranie wartości przy uruchomieniu programu sprawdzających czy zakodować wszystkich, czy tylko jedną osobę,
- threading, odpowiadająca za wykorzystywanie przez program wszystkich dostępnych wątków procesora,
- tkinter - odpowiada za wyświetlenie obrazu.

5.3 Algorytm

Poniżej znajdują się algorytmy programów zastosowanych w projekcie:



Rysunek 8: Program kodujący zdjęcia do plików .json [13].



Rysunek 9: Algorytm programu głównego [13].

5.4 Kod

W tym rozdziale znajdują się listingi kodu odpowiedzialnego za realizację warstwy softwarowej projektu. Prezentowany kod podzielony jest na następujące elementy:

Zamiana obrazu twarzy na postać cyfrową (kodowanie) (Patrz listing 1).

```
video_capture = cv2.VideoCapture(CAM_REF)

# inicjacja zmiennych
face_names = []
process_this_frame = True

# start
while True:
    # wez klatke z kamerki
    ret, frame = video_capture.read()
    # zeskaluj klatke do 1/4 rozmiaru dla szybszego procesu
    small_frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
    # konwertuj klatke z BGR (uzywanego przez OpenCV)
    # do RGB (uzywanego przez face_recognition)
    rgb_small_frame = small_frame[:, :, ::-1]
    # przetwarzaj tylko co druga klatke
    thread = threading.Thread(target=encodeThisFrameFaces,
    args=(rgb_small_frame,))
    if process_this_frame:
        print("appActiveColor: ", appActiveColor, end="\t\t")
        thread.start()
    (...)
```

Listing 1: Optymalizacja klatki wejściowej.

Przypisanie rozpoznanej twarzy do zdefiniowanego w bazie danych imienia i nazwiska (Patrz listing 2, 3).

```
def encodeThisFrameFaces(frame):
    timeStart = time.time()
    # znajdz wszystkie pozycje twarzy
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame,
    face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # sprawdz czy twarz jest znana
        matches = face_recognition.compare_faces(
        known_face["encodings"], face_encoding)
        name = UNKNOWN_NAME

        # jak nie jest znana, to sprawdz najlepsze porownanie
        face_distances = face_recognition.face_distance(
        known_face["encodings"], face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face["names"][best_match_index]

        face_names.append(name)
```

Listing 2: Rozpoznanie osoby (cz. 1 / 2).

```
# sprawdz czy osoba nie powtarza się zbyt często
face_names_to_alert = []
time_differences = {}
for regognized_face in face_names:
    if regognized_face in ALERT_PEOPLE.keys():
        # sprawdz czas ostatniego wykrycia
        time_difference = time.time() - ALERT_PEOPLE[
        regognized_face]
        time_differences[regognized_face] = time_difference
        ALERT_PEOPLE[regognized_face] = time.time()
        if time_difference >= ALERT_DELAY_IGNORE:
            # powiadom
            face_names_to_alert.append(regognized_face)
            pass
        else:
            # pomin
            pass
        pass
    else:
        # dodaj to ALERT_PEOPLE & powiadom
        ALERT_PEOPLE[regognized_face] = time.time()
        face_names_to_alert.append(regognized_face)
        pass
print(f"[{threading.active_count()}] ", *face_names_to_alert,
f" ({(time.time() - timeStart):.4}s)\t\t\t{time_differences}")

for face in face_names_to_alert:
    say_hello(face)
pass
```

Listing 3: Rozpoznanie osoby (cz. 2 / 2).

```
def create_welcome_voice_for_known_person(name, path2root=
PATH_TO_ROOT):
    person_dir_path = os.path.join(path2root,
    KNOW_PEOPLE_DIR_PATH_NAME, name)
    welcomes_dir = os.path.join(person_dir_path,
    WELCOMES_DIR_PATH_NAME)
    basic_welcome_file = os.path.join(welcomes_dir,
    BASIC_WELCOME_FILE_NAME)
    create_and_save_welcome_message(f"Witaj {name}",
    basic_welcome_file)

def create_welcome_voice_for_unknown_person(path2root=PATH_TO_ROOT):
    # stworz folder dla nieznanych osob
    unknowe_dir_name = f"{WELCOMES_DIR_PATH_NAME}_unknown"
    know_people_dir = os.path.join(path2root,
    KNOW_PEOPLE_DIR_PATH_NAME)
    unknown_dir_path = os.path.join(know_people_dir,
    unknowe_dir_name)
    if not os.path.isdir(unknown_dir_path):
        if os.path.isfile(unknown_dir_path):
            os.remove(unknown_dir_path)
        os.chdir(know_people_dir)
        os.mkdir(unknowe_dir_name)

    # stworz plik z wiadomoscia
    file_path = os.path.join(unknown_dir_path,
    BASIC_WELCOME_FILE_NAME)
    create_and_save_welcome_message("Witaj nieznajomy nieznajoma",
    file_path)
```

Listing 4: Generowanie domyślnego komunikatu głosowego

```
def screenVisualization():
    global lastMods2Show
    global lGreeting
    global lName
    if len(mods2ShowQueue) > 0:
        mods2ShowQueue.sort()
        # sprawdzenie wykorzystania limitu czasu
        if time.time() > mods2ShowQueue[0].time_end:
            # pop
            mods2ShowQueue.pop(0)
            lGreeting.pack_forget()
            lName.pack_forget()
            screenVisualization()
        else:
            activeMode2show = mods2ShowQueue[0]
            # zmien tylko gdy mod2show jest nowy
            if activeMode2show != lastMods2Show:
                # kolor tła
                Tk.configure(background=
                    appBackground[activeMode2show.activeMode])
                lGreeting.configure(background=
                    appBackground[activeMode2show.activeMode])
                lName.configure(background=
                    appBackground[activeMode2show.activeMode])

                # zmien imie textu
                lName.config(text=activeMode2show.name)

                # wyswietl text
                lGreeting.pack()
                lName.pack()

                Tk.update()
                if activeMode2show.activeMode == Mods.UNKNOWN:
                    say_hello("Unknown")
                else:
                    say_hello(activeMode2show.name)
                lastMods2Show = activeMode2show
                appLastMode = activeMode2show.activeMode;
    else:
        # zmien na idle
        Tk.configure(background=appBackground[Mods.IDLE])
        Tk.update()
```

Listing 5: Wysłanie komunikatu graficznego na ekran

Obsługa sytuacji w zależności od tego czy twarz została rozpoznana (Patrz listing 6).

```
def say_hello(name, path2root=PATH_TO_ROOT):
    global addPersoneMode2queue

    appActiveName = name

    # pobierz folder z mp3
    mp3s = []
    path_to_welcomes_for_name = os.path.join(path2root,
    KNOW_PEOPLE_DIR_PATH_NAME)
    if name == "Unknown":
        addPersoneMode2queue(Mods.UNKNOWN)
        path_to_welcomes_for_name = os.path.join(
        path_to_welcomes_for_name, "welcomes_unknown")
    else:
        addPersoneMode2queue(Mods.KNOWN, name)
        path_to_welcomes_for_name = os.path.join(
        path_to_welcomes_for_name, name, WELCOMES_DIR_PATH_NAME)

    # pobierz sciezki do plikow mp3
    for file in os.listdir(path_to_welcomes_for_name):
        print(file)
        extension = os.path.splitext(file)[1] # rozszerzenie pliku
        print(extension)
        if extension.lower() == ".mp3":
            mp3s.append(os.path.join(path_to_welcomes_for_name,
            file))
    print(mp3s)
```

Listing 6: Przygotowanie komunikatów głosowych.

Wysłanie komunikatu głosowego do głośników (Patrz listing 7).

```
# wylosuj plik z wiadomoscia mp3
mp3_to_paly = random.choice(mp3s)
print(mp3_to_paly)

# odtworz plik mp3
pygame.mixer.init()
pygame.mixer.music.load(mp3_to_paly)
pygame.mixer.music.play()
while pygame.mixer.get_busy() == True:
    continue
```

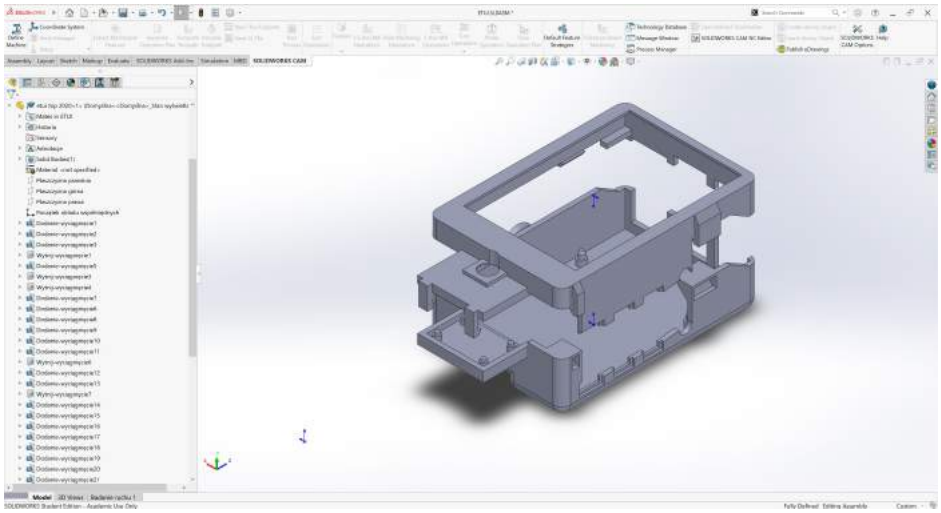
Listing 7: Odtworzenie pliku mp3.

6 Warstwa produktowa

Prototyp to nie tylko elektronika oraz oprogramowanie. Bardzo ważnym elementem jest stworzenie całościowego produktu, do czego niezbędne jest zaprojektowanie oraz wykonanie odpowiedniej obudowy. W celu realizacji tego zadania zostały użyte poniższe programy:

- SOLIDWORKS – do tworzenia projektu obudowy (patrz rys. 10,
- ultimaker cura – do przygodowania wexportowanego z SOLIDWORKS pliku do postaci g-code którą czytają drukatki 3D,
- octoprint (+ Raspbarry pi) – do nadzorowania wydruku

oraz prototypowanie przy pomocy drukarki 3D.



Rysunek 10: Okno programu SOLIDWORKS z projektem obudowy [13].

6.1 Technologia druku 3D

Druk 3D został wynaleziony z konieczności szybkiego i stosunkowo taniego przygotowywania wszelkiego rodzaju prototypów praktycznie we wszystkich branżach działalności gospodarczej. Obecnie prototypowanie przy użyciu wydruków 3D zdecydowanie ułatwia i przyspiesza całość prac od projektu do wykonania wyrobu końcowego.

Zalety prototypowania przy pomocy druku 3D [1]:

- łatwość modyfikacji projektu,
- możliwość natychmiastowego rozpoczęcia drukowania projektu,
- brak konieczności oczekiwania na realizację zadania przez firmy zewnętrzne,
- możliwość zastosowania materiałów o różnych własnościach,
- relatywnie niska cena.

Drukarki 3D używane do prototypowania, wykorzystują 2 podstawowe technologie przyrostowe [1].

- SLA, oraz
- FDM.

W technologii SLA do wydruku wykorzystuje się żywicę, która jest materiałem światło utwardzalnym. Poszczególne warstwy nanosi się na specjalnie zaprojektowanej płytce. Poszczególne warstwy utwardzane są wiązką laserową [7].

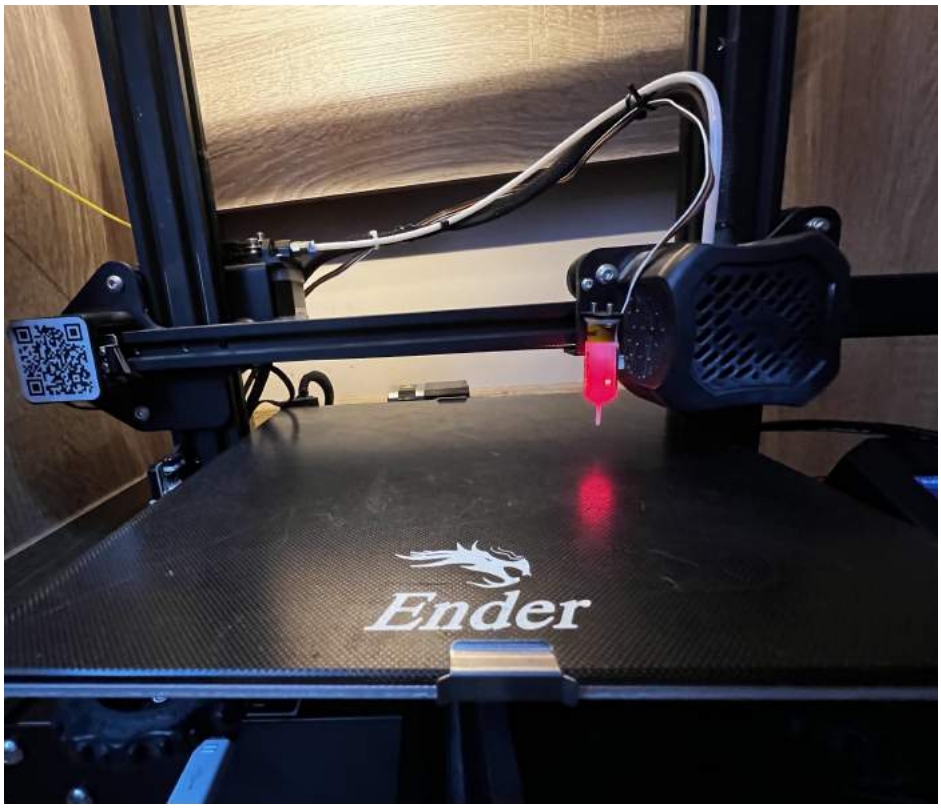
Zaletą tej metody jest duża dokładność otrzymanego modelu, natomiast podstawową wadą tej technologii jest:

- stosunkowo wysoki koszt użycia w warunkach domowych, oraz
- konieczność stosowania wyciągów mechanicznych w celu odprowadzenia powstających w trakcie procesu oparów.

W technologii FDM drukarki do druku korzystają z tworzyw plastikowych takich jak ABS oraz PLA [6] [7].

Do celów realizacji niniejszego projektu wybrałem metodę FDM ze względu na jej niewielki koszt oraz już posiadaną drukarkę 3D wykorzystującą właśnie tą technologię druku. Do druku drukarki wykorzystujące technologię FDM korzystają z tworzyw plastikowych takich jak ABS oraz PLA. Filament, czyli materiał drukujący ("Tusz"), przechodzi przez głowicę drukarki, gdzie następuje jego podgrzanie aż do uzyskania formy płynnej materiału. Następnie jest on wyciskany przez dedykowaną dyszę (stosowane są różne średnice dysz) na stół roboczy. Dysza rozprowadza w przestrzeni roboczej filament w postaci nakładania na siebie kolejnych warstw, zgodnie z przygotowanym wcześniej projektem. Po nałożeniu warstwy następuje jej schłodzenie i filament twardnieje. Po naniesieniu ostatniej warstwy wydruku, wydruk (i elementy drugarki) musi się schłodzić, a następnie sam schodzi z powierzchni drukującej.

Zgodnie z [1] „(...) prototypowanie z wykorzystaniem druku 3D pozwala na wypracowanie oszczędności – czasu i pieniędzy. Umożliwia szybkie testowanie prototypów i przyspiesza czas, jaki niezbędny jest do wprowadzenia docelowego produktu na rynek. Dodatkowo na każdym etapie procesu prototypowania można otrzymać natychmiast podgląd wydruku i śledzić proces powstawania modelu (...)".

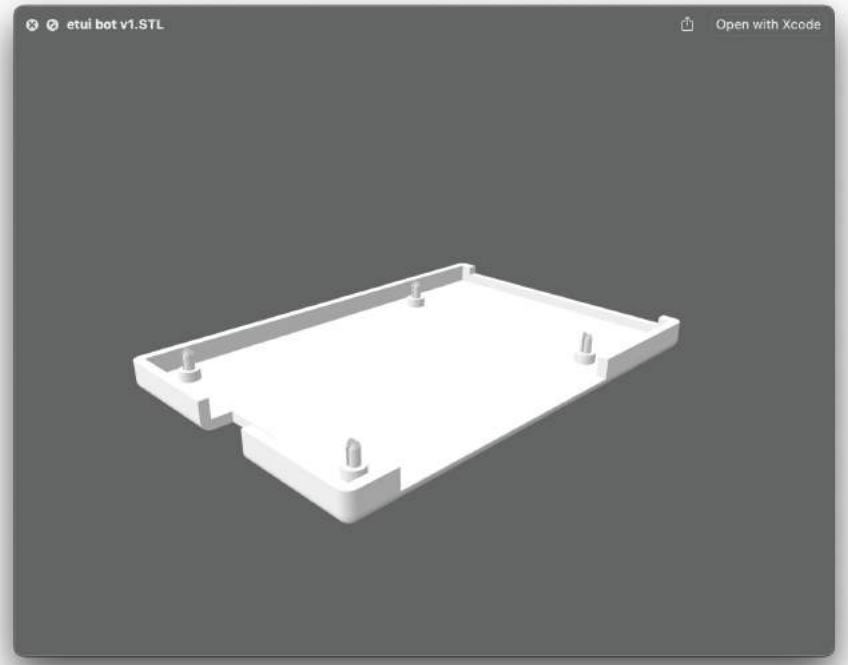


Rysunek 11: Drukarka 3D Creality Ender wykorzystana w projekcie [13].

6.2 Projekt obudowy

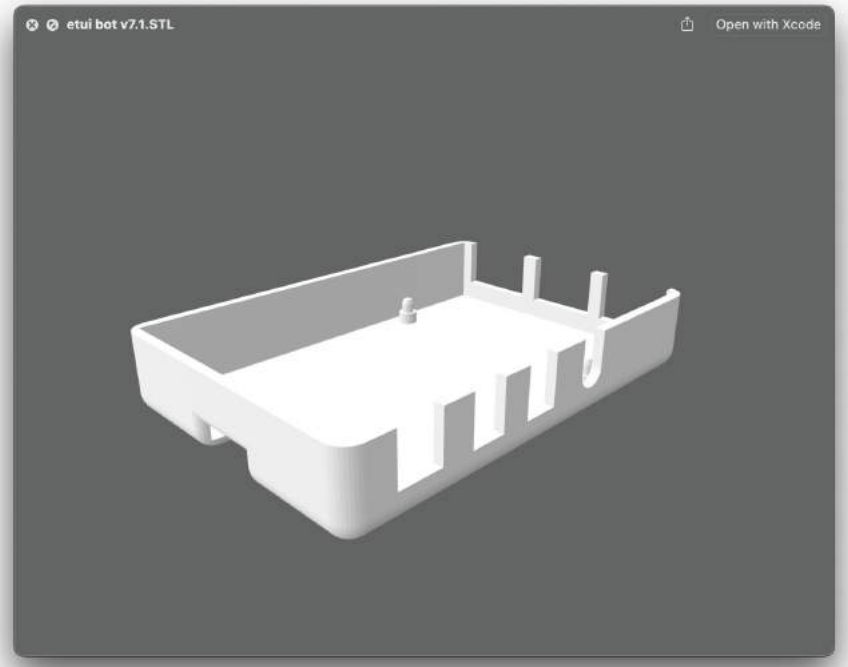
W celu wykonania projektu obudowy zostały wykorzystane oprogramowanie SOLIDWORKS w wersji 20. W początkowej fazie użyto oprogramowania Autodesk Fusion 360, jednak nie mogłem znaleźć w nim wielu narzędzi których się nauczyłem korzystać w SOLIDWORKS.

W ramach wykonania projektu ostatecznego były wprowadzane poprawki do projektu zgodnie z zasadą PDCA[2] stanowiącą podstawę zwinnego projektowania.

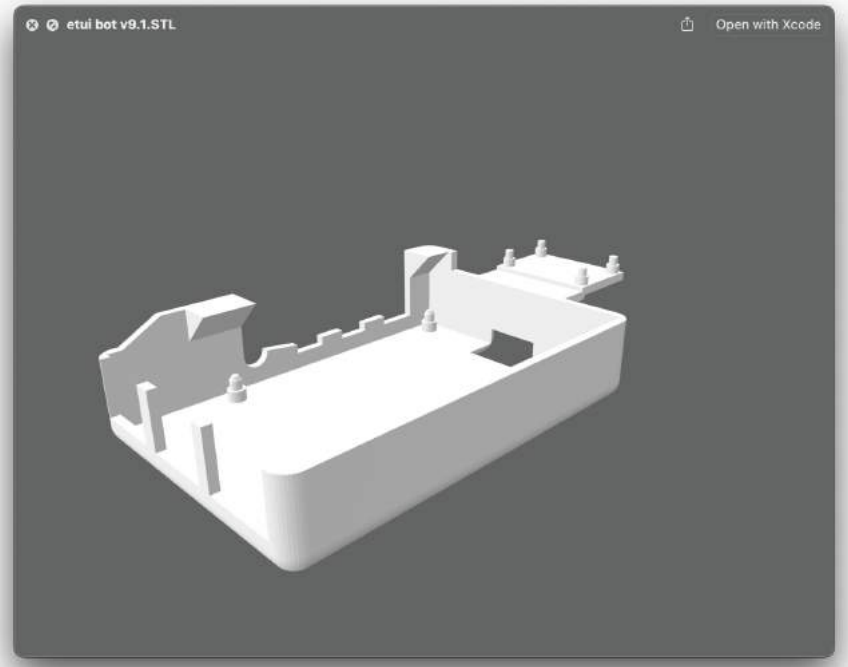


Rysunek 12: Podstawa obudowy. Wersja 1 [13].

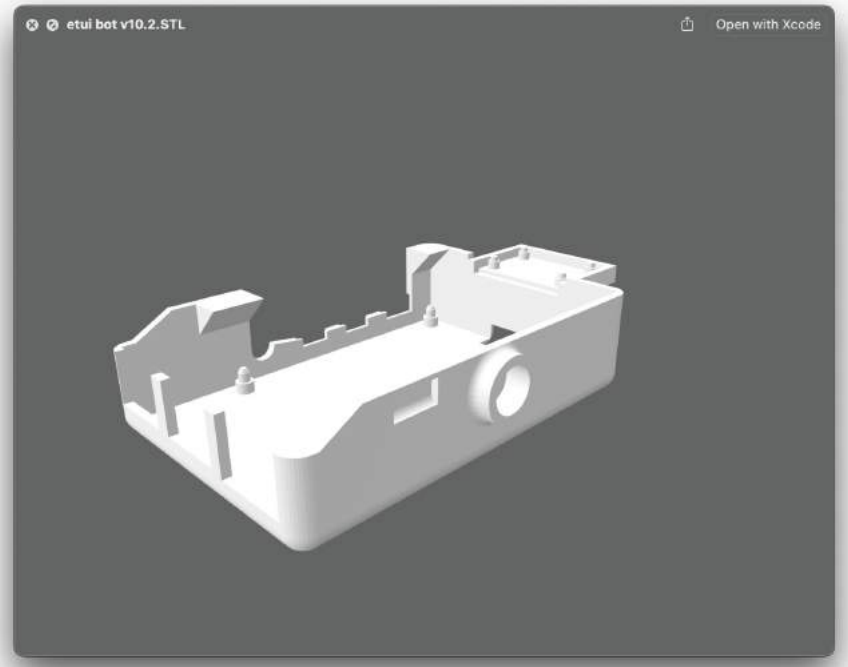
[2]P – PLAN (zaplanuj), D – DO (wykonaj), C – CHECK (sprawdź), A – ACT (popraw).



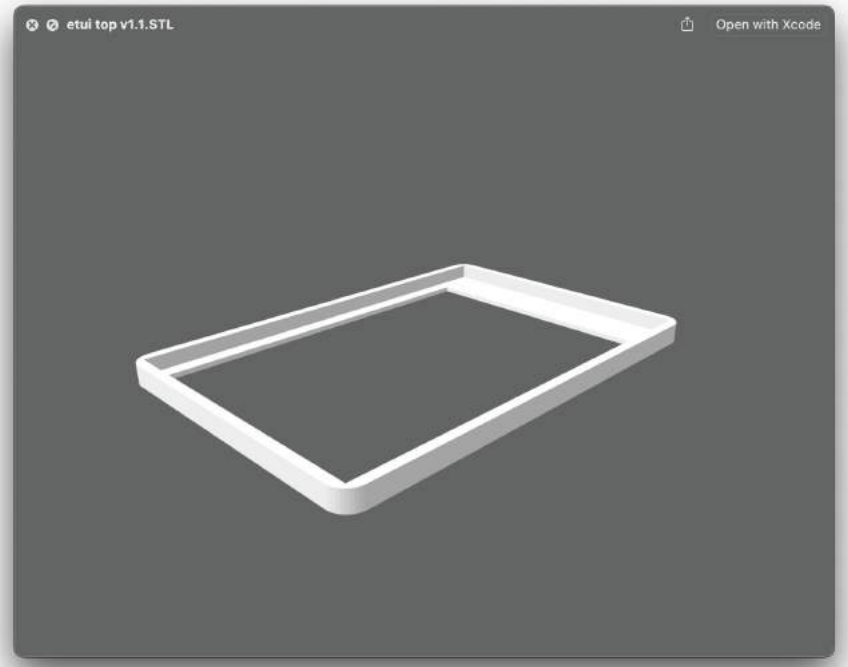
Rysunek 13: Podstawa obudowy. Wersja 7 [13].



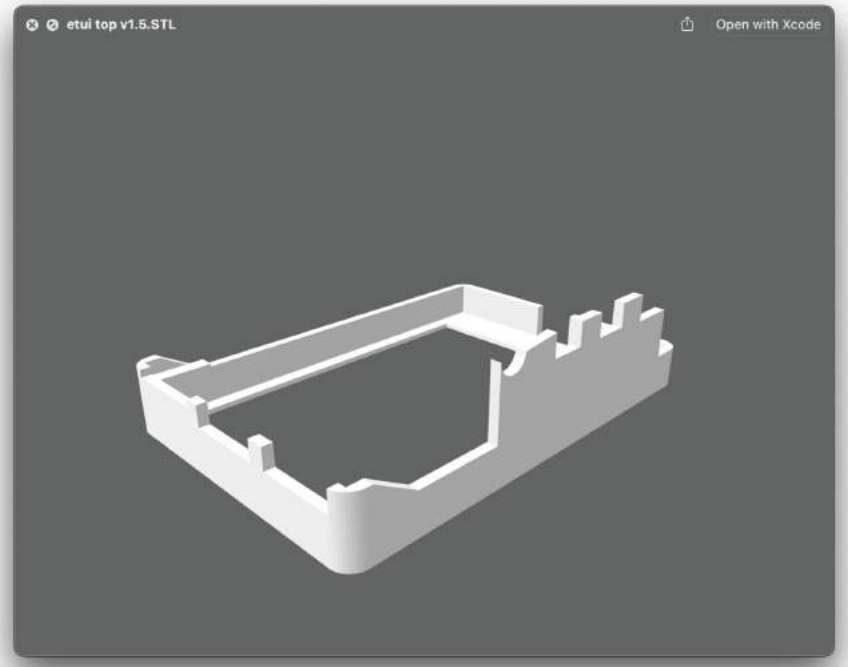
Rysunek 14: Podstawa obudowy. Wersja 9 [13].



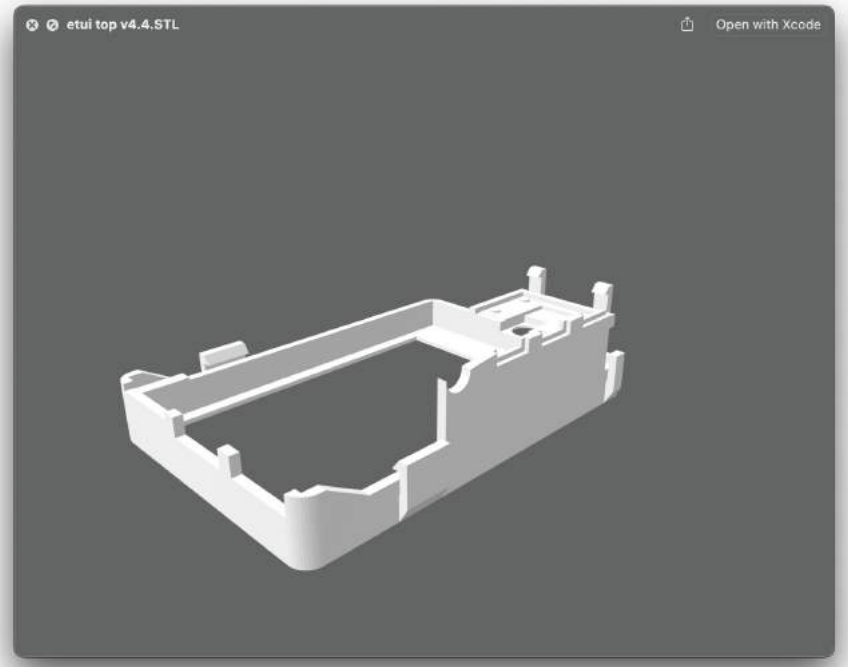
Rysunek 15: Podstawa obudowy. Wersja 10 [13].



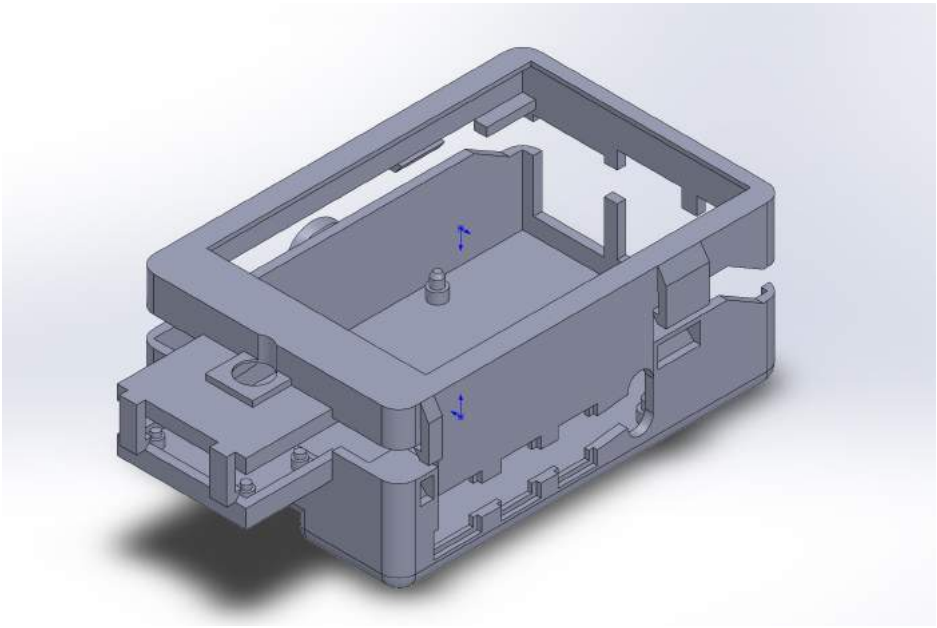
Rysunek 16: Pokrywa obudowy. Wersja 1.1 [13].



Rysunek 17: Pokrywa obudowy. Wersja 1.5 [13].



Rysunek 18: Pokrywa obudowy. Wersja 4 [13].



Rysunek 19: Rysunek złożeniowy kompletnej obudowy [13].

6.3 Wykonanie obudowy

W celu osiągnięcia ostatecznego wyniku przeprowadzono kilka iteracji cyklu Deminga[3] (PDCA) [11]. W tym celu dokonywano także poprawek ustawień drukarki.

Rysunek 20: Iteracja Początkowa [13].

W pierwszej iteracji (Patrz rys. 20) okazało się, że wymiary i rozmieszczenie bolców jest lekko przesunięte względem Raspberry Pi.

[3]Cykl Deminga – sposób strukturalnego wykonywania pracy w postaci 4 kroków (zaplanuj, wykonaj, sprawdź, popraw)

W jednej z pośrednich iteracji (Patrz rys. 21) okazało się, że otwory na panelach wyjścia / wejścia posiadają zbyt małą średnicę, oraz ekran nie był zamocowany stabilnie, ze względu na brak podpór.

Rysunek 21: Iteracja pośrednia 1 [13].

Rysunek 22: Iteracja pośrednia 2 [13].

Następna iteracja (Patrz rys. 22) miała na celu wydrukowanie poprawnych wymiarów dla kamery. Z powodu niewielkich rozmiarów urządzenia nie było łatwo dokładnie zmierzyć otworów. Nawet z uwzględnieniem niepewności pomiarowej trzeba było wydrukować komponent kilka razy z różnicą kilku dziesiątych mm.

Rysunek 23: Iteracja pośrednia 3 [13].

W jednej z końcowych iteracji (Patrz rys. 23) celem było wpasowanie obu części obudowy w siebie nawzajem. Na "Papierze" wszystkie wymiary się zgadzały i powinny pasować. Jednak z względu niedoskonałości wydruku było inaczej.

Ostatnia iteracja (Patrz rys. 24) złożona z zamontowanymi w niej częściami, oraz śrubą aparatową pod stojak.

Rysunek 24: Iteracja końcowa [13].

Wszystkie wydrukowane Iteracje PDCA (Patrz rys. 25).

Rysunek 25: Wszystkie wydrukowane Iteracje [13].

7 Realizacja projektu

7.1 Napotkane problemy

W trakcie realizacji projektu napotkano następujące problemy.

- **Wykorzystanie tylko jednego rdzenia procesora** (wątku). Raspberry Pi wyposażony jest w procesor czterordzeniowym, dzięki czemu można jednocześnie realizować 4 równoległe operacje.
  Problem:
  Okazało się że procesor Raspberry ma problemy z wykorzystaniem większej liczby wątków w ramach jednego programu.
  Rozwiązanie:
  Użyto biblioteki threading do pracy na wątkach.

- **Brak możliwość realizacji komunikatów głosowych.**
  Problem:
  Pierwotnie zastosowany głośnik nie posiadał wsparcia dla realizacji komunikatów głosowych zapisanych w pamięci prototypu. Jedyną możliwością było odtwarzanie plików przez Spotify.
  Rozwiązanie:
  Zastosowano zewnętrzne głośniki podpinane przez złącze jack. Plusem tego rozwiązanie jest równiesz dowolność wyboru rozmiaru głośników.

- **Zbyt duża temperatura prototypu ze względu na konieczność wykonywania dużej liczby obliczeń.**
  Problem:
  W początkowej fazie projektu planowano użyć wyświetlacza zintegrowanego z głośnikiem. Okazało się iż takie rozwiązanie powoduje, że brakuje miejsca na radiator.
  Rozwiązanie:
  Znaleziono nowy wyświetlacz oraz wyprowadzono głośniki na zewnątrz przy pomocy gniazda typu jack.

7.2 Możliwości rozbudowy

W ramach projektu powstał pierwszy w pełni funkcjonalny prototyp inteligentnego rozpoznawania twarzy. Na etapie prototypu zostały zaimplementowane funkcjonalności, zgodnie z założeniami projektu.

Nie mniej sam system jest elastyczny i można go z czasem rozszerzyć o kolejne funkcjonalności:

- integrację z istniejącymi na rynku systemami Smart Home,
- wprowadzenie funkcji nadzoru obszaru ze śledzeniem osób przemieszczających się w danych strefach,
- dodawanie zdjęć twarzy do bazy danych z wykorzystaniem kamery w którą wyposażony jest prototyp,

- zintegrowanie funkcji dostępu do pomieszczeń po rozpoznaniu twarzy, lub dzięki dotykowemu ekranowi, w który wyposażony jest projekt,
- generowanie sygnału uruchomiającego sekwencję zdarzeń po rozpoznaniu danej osoby. (na przykład włączenie czajnika lub ekspresu do kawy, czy też zapalenie światła w określonym pomieszczeniu),
- stworzenie centrali pobierającej dane z kilku - kilkunastu punktów zbudowanych na bazie obecnego prototypu.

7.3 Koszty realizacji projektu

Element projektu	**Cena [zł]**
Raspberry Pi 4b	435.00 zł
Kamera	175.90 zł
Ekran	115.00 zł
Głośnik	49.99 zł
Radiatory	3,70 zł
Obudowa	50.00 zł
Podstawka	34.99 zł
Razem	**865.58 zł**

Całkowity koszt projektu wyniósł 865.58 zł, co oznacza iż całość kosztów zawiera się w założonym budżecie.

8 Wnioski

Stworzenie projektu inteligentnego rozpoznawania twarzy pozwala na sformułowanie kilku podstawowych wniosków, które powstały na bazie doświadczania przy realizacji zadania.

1. Wybrany temat pracy wymagał skutecznego połączenia wiedzy nabytej w trakcie studiów z zakresu elektroniki, programowania oraz projektowania z wykorzystaniem oprogramowania typu CAD.

2. Nawet przy ograniczonym budżecie możliwa jest realizacja zadań związanych z rozpoznawaniem twarzy i zbudowania prototypu, który w przyszłości będzie można integrować z rozwiązaniami typu Smart Home.

3. W trakcie realizacji projektu napotkano problemy, które należało rozwiązać. Okazało się, iż każdy problem może być rozwiązany przy zastosowaniu odpowiednich środków zaradczych.

4. Zastosowana technika druku 3D jest techniką zbyt wolną, aby produkować obudowy prototypów seryjnie. Nadaje się świetnie natomiast do prototypowania.

5. Zastosowane oprogramowanie nie może być na tym etapie użyte w zakresie ochrony dostępu, ponieważ nie zaimplementowano rozróżnienia żywej osoby od zdjęcia.

Literatura

[1] Prototypowanie w druku 3d – zalety. https://b3d.com.pl/szybkie-prototypowanie-w-druku-3d/. [Online; Odwiedzono 2022-05-13].

[2] Zdjęcie ekranu. https://cdn2.botland.com.pl/16493-large_default/ekran-dotykowy-a-rezystancyjny-lcd-tft-35-320x480px-gpio-dla-raspberry-pi-4-3-2-b-zero-waveshare-9904.jpg. [Online; Odwiedzono 2022-05-13].

[3] Zdjęcie kamery. https://cdn3.botland.com.pl/106920-large_default/raspberry-pi-camera-hd-v2-8mpx-oryginalna-kamera-do-raspberry-pi.jpg. [Online; Odwiedzono 2022-05-13].

[4] Zdjęcie mikro-komputera. https://cdn3.botland.com.pl/78044-large_default/raspberry-pi-4-model-b-wifi-dualband-bluetooth-8gb-ram-15ghz.jpg. [Online; Odwiedzono 2022-05-13].

[5] Api rozpoznawania twarzy. https://github.com/ageitgey/face_recognition, 2018. [Online; Odwiedzono 2022-05-13].

[6] Jak wybrać najlepsze tworzywo do drukarek 3d? https://up3d.pl/technologia/jak-wybrac-najlepsze-tworzywo-do-drukarek-3d/, 2021. [Online; Odwiedzono 2022-05-13].

[7] Jakie materiały wykorzystuje się w druku 3d? , https://b3d.com.pl/jakie-materialy-wykorzystuje-sie-w-druku-3d/. [Online; Odwiedzono 2022-05-13].

[8] Manav Bansal. Face recognition implementation on raspberrypi using opencv and python. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3557027, 2019. [Online; Odwiedzono 2022-05-13].

[9] Adam Geitgey. Recognize faces from python or from the command line. https://pypi.org/project/face-recognition/, 2020. [Online; Odwiedzono 2022-05-13].

[10] Maliha Khan, Sudeshna Chakraborty, Rani Astya, and Shaveta Khepra. Face detection and recognition using opencv. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 116–119. IEEE, 2019.

[11] Jacek Kukiz. *Narzędzia Lean. Podejście praktyczne.* LeanCert sp z o.o., 1 edition, 2018.

[12] Krzysztof Kukiz. Github: Cookiez (krzysztof kukiz). https://github.com/Cooookiez, 2022. [Online; Odwiedzono 2022-05-13].

[13] Krzysztof Kukiz. Archiwum prywatne, 2022-05-05.

[14] Rob. Raspberry pi - python threading. http://robsraspberrypi.blogspot.com/2016/01/raspberry-pi-python-threading.html. [Online; Odwiedzono 2022-05-13].

[15] Shantnu Tiwari. Face recognition with python, in under 25 lines of code. https://realpython.com/face-recognition-with-python/. [Online; Odwiedzono 2022-05-13].