

**POLITECHNIKA POZNAŃSKA**  
**WYDZIAŁ ELEKTRYCZNY**

Dokumentacja projektowa z Podstaw Teleinformatyki

***Face Recognition System***

**PROJEKT APLIKACJI WYKORZYSTUJĄCEJ KAMERĘ IP  
DO ROZPOZNAWANIA I ZLICZANIA LUDZI**

**Monika Grądzka**  
**Robert Kazimierczak**

<https://github.com/gradzka/FaceRecognitionSystem>

# SPIS TREŚCI

<b>OPIS PROJEKTU</b>	<b>3</b>
<b>UZASADNIENIE WYBRANIA PROJEKTU</b>	<b>4</b>
<b>PODZIAŁ PRACY W ZESPOLE</b>	<b>5</b>
<b>OPIS WYMAGAŃ FUNKCJONALNYCH</b>	<b>6</b>
<b>OPIS WYMAGAŃ NIEFUNKCJONALNYCH</b>	<b>9</b>
<b>WYMAGANIA SPRZĘTOWE</b>	<b>10</b>
<b>ŚRODOWISKO, BIBLIOTEKI ZEWNĘTRZNE, NARZĘDZIA DODATKOWE</b>	<b>10</b>
<b>ZAGADNIENIA PROBLEMATYCZNE</b>	<b>11</b>
<b>ROZWIĄZANIA ZAGADNIENI PROBLEMATYCZNYCH</b>	<b>11</b>
<b>WYBRANE FRAGMENTY KODU</b>	<b>14</b>
<b>INSTRUKCJA UŻYTKOWANIA APLIKACJI</b>	<b>18</b>
KONFIGURACJA POŁĄCZENIA Z KAMERĄ IP	18
KONFIGURACJA MANUALNA	19
KONFIGURACJA Z UŻYCIEM PLIKU KONFIGURACYJNEGO	20
UTWORZENIE PLIKU KONFIGURACYJNEGO	21
MODUŁY FACE RECOGNITION SYSTEM	22
UTWORZENIE PLIKU KONFIGURACYJNEGO	23
WYKONYWANIE ZDJĘĆ DO KORPUSU	23
ZLICZANIE LUDZI	25
TRENOWANIE FACE RECOGNIZER	25
ROZPOZNAWANIE LUDZI	26
HELP	27
WYJŚCIE Z PROGRAMU	27
<b>PLANY NA ROZWINIĘCIE APLIKACJI</b>	<b>27</b>
<b>TESTOWANIE APLIKACJI</b>	<b>28</b>

# 1. OPIS PROJEKTU

Projekt ma na celu utworzenie 32-bitowej aplikacji konsolowej do rozpoznawania twarzy oraz zliczania ludzi. Pobocznym celem projektu, nie mniej ważnym, jest zadbanie o wygodę użytkownika i bezpieczeństwo wrażliwych danych oraz kompatybilność z systemami Linux i Windows.

Najważniejszym modulem projektu jest komunikacja z kamerą IP. Ze względów finansowych wybrano niskobudżetową kamerę IP Media-Tech MT4051. Do komunikacji z kamerą sieciową wykorzystano bibliotekę programistyczną cURL. Użytkownik o odpowiednich uprawnieniach może sterować kamerą przy użyciu klawiatury, czy robić zdjęcia.

Kolejną ważną częścią aplikacji jest logowanie użytkownika. Użytkownik może logować się poprzez ręczne podanie danych, tj. adresu IP kamery, loginu oraz hasła lub za pomocą pliku konfiguracyjnego zaszyfrowanego przy pomocy symetrycznego szyfru blokowego AES. Chcąc skorzystać z logowania przy użyciu pliku konfiguracyjnego, użytkownik musi go najpierw utworzyć, wybrawszy odpowiednią opcję w programie. Po wybraniu opcji utworzenia pliku użytkownik jest proszony o podanie tych danych, co przy manualnym logowaniu i dodatkowo o tajny klucz. Klucz może być dowolnej długości, ponieważ do szyfrowania danych używany jest skrót klucza uzyskany po zastosowaniu funkcji skrótu SHA256.

Przechodząc do głównego celu projektu - rozpoznawanie twarzy, należy wziąć pod uwagę przygotowanie odpowiednich modułów oraz główną bibliotekę projektu - OpenCV. Pierwszym z nich jest mechanizm robienia zdjęć potrzebny do zbudowania korpusu. Użytkownik najpierw musi podać nazwę folderu wyjściowego oraz określić liczbę zdjęć, którą wykona kamera. Zdjęcia są wykonywane w odstępie 3 sekund. Długi sygnał dźwiękowy poprzedza i kończy serię zdjęć, krótkie informują o wykonanym zdjęciu, co daje czas użytkownikowi, np. na zmienienie mimiki twarzy. Zdjęcia mają określony format, kolor oraz rozszerzenie, a także nie są przypadkowe. Przed wykonaniem zdjęcia następuje detekcja twarzy i sprawdzenie, czy w kadrze nie znajduje się więcej niż jedna osoba za pomocą modułu zliczającego ludzi. Detekcja twarzy uzyskana jest dzięki klasyfikatorowi kaskadowemu LBP (ang. *Local Binary Patterns*). Po wykryciu jednej twarzy zdjęcie jest przycinane i zapisane w odcieniach szarości w formacie .png. Mając przygotowany korpus, użytkownik może rozpocząć trenowanie rozpoznawania twarzy.

W początkowej fazie trenowania utworzony jest odpowiedni plik, który zawiera informacje o zdjęciach znajdujących się w przygotowanych przez użytkownika folderach w folderze screenshots. Etykieta przy każdym wpisie w pliku specyfikuje zdjęcia, które znajdują się w danym folderze (dotyczą jednej osoby). Dzięki niej rozróżniane są zdjęcia innych osób. Następnie plik ten zostanie odczytany. Na sam koniec model rozpoczyna trenowanie. Gdy model zostanie poprawnie wytrenowany, użytkownik jest pytany o to, czy chce zapisać ustawienia wytrenowanego modelu do pliku .xml. Gdy zapiszemy ustawienia, w przyszłości możemy z nich skorzystać. Mając wytrenowany model, możemy przejść do modułu rozpoznawania twarzy.

Moduł rozpoznawania twarzy oparty jest na algorytmie LBPH (ang. *local binary patterns histograms*). Histogramy lokalnych wzorców binarnych pozwalają wydobywać cechy z obrazów cyfrowych, służą do opisu kształtów, czy pokrycia powierzchni obiektu w przetwarzanym zdjęciu. Metoda ta nie wymaga dużej liczby zdjęć w korpusie, aby uzyskać

zadowalające wyniki w klasyfikowaniu<sup>1</sup>. Moduł ten można uruchomić tylko jeśli wcześniej został wczytany lub wytrenowany. Mając przygotowaną odpowiednią bazę ze zdjęciami różnych osób, jesteśmy w stanie zidentyfikować osobę, której zdjęcie wykona dla nas kamera. Jeżeli na wykonanym zdjęciu znajduje się więcej niż jedna osoba, to rozpoznawanie odbywa się dla każdej z twarzy. Dane o rozpoznanych oraz nierozpoznanych osobach wyświetlają się na ekranie. Dodatkowo, rezultaty działania algorytmu (lista rozpoznanych osób oraz liczba nieudanych prób rozpoznawania twarzy) trafiają do odpowiedniego pliku tekstowego.

Ostatnim modułem, wykorzystywanym w innych modułach jest zliczanie ludzi. Opiera się on na dziesięciokrotnym powtórzeniu kroku: pobraniu klatki i detekcji twarzy. Zliczone twarze przy każdym kroku są dodawane do sumy wykrytych twarzy i na koniec dzielone przez liczbę wykonanych klatek. Wynik jest zaokrąglany do liczby całkowitej.

**Słowa kluczowe:** kamera IP Media-Tech MT4051, cURL, OpenCV, AES, SHA256, klasyfikator kaskadowy LBP, algorytm LBPH.

## 2. UZASADNIENIE WYBRANIA PROJEKTU

Pomysł na wykonanie projektu wziął się z doświadczenia z pracą z kamerami IP obydwóch członków zespołu. Doświadczenie opierało się na komunikacji z kamerami IP, stworzeniu prostego pliku konfiguracyjnego oraz wywołaniu programu z parametrami. Projekt ten jednak zawierał szereg rozwiązań, które należało poprawić. Do komunikacji z kamerą używano programu Wget. Dane wprowadzane do pliku nie były sprawdzane, co więcej plik nie był szyfrowany. Należało więc znaleźć inne rozwiązania i dostosować komunikację do innego modelu kamery IP.

Innym głównym powodem wybrania projektu jest plan wykorzystania go w pracy inżynierskiej jako jednego z modułów. Z tego powodu wynika dbanie o bezpieczeństwo użytkownika. Założono również, że interfejs graficzny nie pełni tutaj dużej roli - stąd decyzja o aplikacji konsolowej.

Ważnym powodem jest również chęć zapoznania się z parametrami kamery niskobudżetowej i dodanie do aplikacji modułów, które nie są dostarczane w darmowych aplikacjach dołączanych do takich kamer.

Projekt również wybrano, gdyż pobocznym planem jest zrealizowanie aplikacji kompatybilnej z systemami Windows i Linux. Z drugiej strony jest również chęć zapoznania się z kompatybilnością różnych modeli kamer IP, w ramach czego, konieczne jest zapoznanie się ze standardem ONVIF<sup>2</sup> i sprawdzenie ich interfejsów. Należy zadać sobie pytanie, czy inne modele kamer będą obsługiwały zaimplementowane funkcje. Odpowiedź pozostawiamy jako jedną ze ścieżek rozwoju aplikacji.

---

<sup>1</sup> <http://pe.org.pl/articles/2016/9/36.pdf>

<sup>2</sup> Otwarty, globalny standard interfejsu do komunikacji urządzeń w sieci komputerowej

### 3. PODZIAŁ PRACY W ZESPOLE

Do udokumentowania podziału prac w projekcie wykorzystano modyfikację wykresu Gantt<sup>3</sup>. Przedstawiono go w formie *Tabeli 3.1.*. Znajdziemy w niej informacje na temat zadania, które zaimplementował podany członek zespołu wraz z określeniem tygodnia pracy. Dokładne informacje na temat przebiegu pracy w implementowaniu aplikacji można znaleźć na stronie: <https://github.com/gradzka/FaceRecognitionSystem/commits/master>.

*Tabela 3.1. Wykres Gantt (Tn \_ n-ty tydzień pracy, M \_ Monika, R \_ Robert)*

Zadania:	Osoba:	T1	T2	T3	T4	T5	T6
Wybór tematu projektu	M, R	*					
Utworzenie repozytorium na github.com	M	*					
Wstępne rozeznanie tematu i podział prac	M	*					
Prezentacja nr 1.	M,R	*					
Dodanie biblioteki libcurl, komunikowanie się z kamerą IP (podstawowe sterowanie kamerą za pomocą klawiatury)	R		*				
Dodanie biblioteki OpenCV 3, robienie zdjęć kamerą IP na żądanie, nazewnictwo zdjęć i folderów	M		*				
Pobieranie od użytkownika danych niezbędnych do nazywania zdjęć i folderów, zadbanie o czyszczenie zawartości strumienia wejściowego, dodanie komendy HELP	R		*				
Zmiana nazw folderów, interfejs użytkownika,	M		*				
Dodanie regex do adresu IP, prototyp trybów logowania	M		*				
Obsługa komend HOME i PICTURE, COMMAND	R		*				
Prezentacja nr 2	M, R		*				
Logowanie manualne	M			*			
Metoda testująca połączenie z kamerą	R			*			
Tworzenie pliku konfiguracyjnego, logowanie za pomocą pliku konfiguracyjnego	M			*			
Zabezpieczenia logowania przy użyciu pliku konfiguracyjnego	R			*			
Detekcja twarzy	R			*			
Zliczanie ludzi jako moduł wykorzystujący detekcję twarzy	M			*			

<sup>3</sup> Graficzny sposób planowania i kontroli

Tworzenie pliku .csv	R			*			
Prezentacja nr 3.	M, R			*			
Dodanie biblioteki AES, SHA256	R				*		
Przebudowa libcurl, zmiana wersji OpenCV na 2.7	R				*		
Tworzenie pliku konfiguracyjnego z zaszyfrowanymi danymi przy użyciu AES	M				*		
Prototyp trenowania modelu	M				*		
Prototyp rozpoznawania twarzy	R				*		
Czasowe pobieranie zdjęć z kamery, trenowanie FaceRecognizer	M				*		
Prototyp rozpoznawania twarzy (“zdjęcia na sztywno”)	R				*		
Tworzenie dokumentacji projektu	M, R				*	*	*
Prezentacja nr 4.	M, R				*		
Rozpoznawanie twarzy na podstawie bieżących klatek z kamery IP	R					*	
Dodanie słowników związanych z rozpoznawaniem osób	M					*	
Zapis rezultatów działania modułu rozpoznawania twarzy do pliku	M					*	
Tryby działania modułu rozpoznawania twarzy (nieograniczony i ograniczony czasowo)	R					*	
Interfejs graficzny	M					*	
Prezentacja nr 5.	M, R					*	
Poprawa wykrytych błędów w aplikacji	M, R						*
Prezentacja nr 6.	M, R						*

## 4. OPIS WYMAGAŃ FUNKCJONALNYCH

Wszystkie wymagania funkcjonalne związane są z aktorem, którym jest użytkownik aplikacji. Niektóre z nich nie są dostępne jako opcja aplikacji, a ukryte są w innych opcjach. Wymagania funkcjonalne podzielono względem obszaru działania użytkownika.

Tabela 4.1. Opis wymagań funkcjonalnych w danych obszarach działania

L.p.	Wymaganie funkcjonalne	Obszar działania
1.	<p><i>Logowanie</i> (dostępne po uruchomieniu aplikacji, przed zalogowaniem)</p> <p>Logowanie może być manualne lub za pomocą pliku konfiguracyjnego. Wybrawszy logowanie manualne, użytkownik proszony jest o podanie adresu IP kamery, loginu oraz hasła. Wybrawszy plik konfiguracyjny, użytkownik musi podać tajny klucz, który wprowadził podczas procesu tworzenia pliku konfiguracyjnego. Niezależnie od wyboru trybu logowania następuje walidacja danych, tj. sprawdzane jest połączenie z kamerą IP. Jeżeli test zakończy się sukcesem, użytkownik zostanie zalogowany, w przeciwnym przypadku - nie.</p>	Logowanie
2.	<p><i>Utworzenie pliku konfiguracyjnego</i> (dostępne po uruchomieniu aplikacji, przed zalogowaniem)</p> <p>Może być utworzony wyłącznie przed zalogowaniem. Użytkownik oprócz podania adresu IP kamery, loginu i hasła musi podać tajny klucz, którego skrót utworzony dzięki funkcji skrótu SHA256 zostanie wykorzystany do zaszyfrowania danych z użyciem symetrycznego szyfru blokowego AES. Przed utworzeniem pliku następuje walidacja danych, tj. sprawdzane jest połączenie z kamerą IP. Jeżeli ten test zakończy się sukcesem, plik zostanie utworzony, a użytkownik zalogowany, w przeciwnym wypadku - nie.</p>	Plik konfiguracyjny
3.	<p><i>Sterowanie</i></p> <p>Kamerą IP można sterować za pomocą odpowiednich klawiszy. Dostępne jest wykonywanie ruchów: w lewo, w prawo, do góry, na dół oraz powrót do pozycji domowej.</p>	Kamera IP
4.	<p><i>Wykonywanie zdjęć</i></p> <p>Dzięki odpowiedniemu przyciskowi użytkownik może wykonać zdjęcia do korpusu. Po wybraniu modułu, użytkownik proszony jest o podanie nazwy folderu, gdzie będą przechowywane zdjęcia i ich liczbę. Zdjęcia do korpusu zapisywane są tylko, jeśli twarz na pobranej klatce zostanie wykryta, co więcej sprawdzana jest liczba wykrytych twarzy. Jeżeli zostanie wykryta tylko jedna twarz, wtedy zdjęcie jest przycinane do odpowiedniego rozmiaru i zapisywane w odcieniach szarości w odpowiednim formacie, w przeciwnym przypadku, zdjęcie jest sklasyfikowane jako niepoprawne. Czas pomiędzy pobraniem kolejnej klatki z kamery wynosi 3 sekundy. Rozpoczęcie i zakończenie procesu wykonywania zdjęć sygnalizowane jest sygnałem ciągłym, wykonanie poprawnego zdjęcia - sygnałem krótkim.</p>	
5.	<p><i>Detekcja twarzy</i></p> <p>Detekcja twarzy na zdjęciu opiera się na klasyfikatorze kaskadowym LBP (ang. <i>Local Binary Patterns</i>). Wykonywana jest ona na klatce pobrania z kamery. Nie jest dostępna jako opcja aplikacji.</p>	Zdjęcie
6.	<p><i>Rozpoznawanie twarzy</i></p> <p>Wybrawszy odpowiedni klawisz, możemy uruchomić mechanizm rozpoznawania twarzy, który opiera się na algorytmie LBPH. Ważne jest, by przed procesem rozpoznawania twarzy, użytkownik wytrenował lub wczytał model na wcześniej przygotowanym korpusie. Zdjęcia w korpusie muszą spełniać określone wymagania. Rozpoznawanie twarzy następuje w locie, to znaczy bezpośrednio z klatek pobieranych z kamery. Dzięki zastosowaniu kontenera map przed zapisaniem informacji</p>	

	o rozpoznanych osobach do pliku, eliminowane są redundancje wystąpień osób. Nierozpoznanie osoby powoduje zwiększenie odpowiedniego licznika. Na końcu pliku, dzięki licznikowi osób nierozpoznanych, zapisywana jest informacja o liczbie prób nierozpoznanych osób. Rozpoznawanie twarzy ma ustawiony próg odchylenia od oryginalnych zdjęć z korpusu: 100. Powyżej tego progu osoba uznawana jest za nierozpoznaną.	
7.	<p style="text-align: center;"><i>Liczenie ludzi</i></p> <p>Wybrawszy odpowiedni klawisz możemy uruchomić mechanizm liczenia ludzi. Wykorzystany jest do niego moduł detekcji twarzy. Wynikowa liczba, którą otrzymuje użytkownik związana jest z 10 krotnym powtórzeniem detekcji twarzy na różnych, pobranych klatkach, dodaniu do siebie liczby wykrytych twarzy i podzieleniu otrzymanej sumy przez 10. Wynik końcowy jest zaokrąglany do liczby całkowitej.</p>	
8.	<p style="text-align: center;"><i>Trenowanie modelu FaceRecognizer:</i></p> <p>Po wybraniu opcji trenowania modelu FaceRecognizer, aplikacja sprawdza, czy istnieje odpowiedni plik, w którym znajdują się dane na temat wytrenowanego modelu. Jeżeli istnieje plik o odpowiedniej nazwie i nie jest on pusty, użytkownik ma zadane pytanie, czy chce załadować model, czy rozpocząć trenowanie nowego modelu. Jeżeli użytkownik wybierze rozpoczęcie trenowania na nowo, następuje utworzenie pliku .csv na podstawie bazy zdjęć, odczytanie utworzonego pliku i wytrenowanie modelu. Jeżeli model zostanie wytrenowany poprawnie (informacja o procesie trenowania zakończonym sukcesem), użytkownik jest pytany, czy chce zapisać dany model do pliku.</p>	Baza zdjęć (znajdująca się w folderze /screenshots)
9.	<p style="text-align: center;"><i>Operacje na pliku .csv</i></p> <p>Moduł ten nie jest dostępny jako opcja aplikacji, a zaszyty jest w module trenowania FaceRecognizer. Plik tworzą informacje o zdjęciach danych osób, które znajdują się w folderach o unikalnych nazwach. Folder, w którym znajdują się zdjęcia danej osoby, specyfikuje etykieta. Etykieta pozwala rozróżnić ludzi podczas działania modułu rozpoznawania twarzy.</p>	
10.	<p style="text-align: center;"><i>Utworzenie listy obecności</i></p> <p>Pliki obecności są zapisywane jako pliki .txt nazwach [data] [godzina]. Znajduje się w nim lista numerowana osób rozpoznanych i informacja o liczbie osób nierozpoznanych. W pliku nie ma redundancji danych. Dzięki zastosowaniu kontenera map przed zapisem rozpoznanych osób do pliku, wyeliminowano powtarzane rozpoznawanie danej osoby. Na końcu pliku znajduje się liczba prób nierozpoznanych osób.</p>	Pliki obecności (znajdujące się w folderze /presence)
11.	<p style="text-align: center;"><i>Uruchomienie trybu nieskończonego lub ograniczonego czasowo</i></p> <p>Po wybraniu opcji rozpoznawania ludzi, użytkownik może określić tryb działania algorytmu. Wybrawszy opcję nieskończonego działania, program działa do momentu przerwania przez użytkownika. Zdecydowawszy się na drugą opcję, użytkownik musi podać czas działania programu zgodnie z formatem HH:mm:ss. Po zakończeniu się programu wyświetlany jest komunikat na ekranie informujący o pomyślnym zakończeniu rozpoznawania ludzi.</p>	Tryb nieskończony oraz ograniczony czasowo
12.	Pokazanie listy komend dostępnych dla użytkownika.	Komendy dodatkowe
13.	Wyłączenie aplikacji.	



## 5. OPIS WYMAGAŃ NIEFUNKCJONALNYCH

W tym rozdziale zaprezentowane zostaną ograniczenia, które nałożone są na aplikację. Ograniczenia te dotyczą zarówno tego ilu bitowa ma być aplikacja, w jakim języku programowania napisana, po sposób komunikowania się stacji klienckiej z kamerą IP, czy sposoby realizacji zabezpieczenia danych.

*Tabela 5.1. Opis wymagań niefunkcjonalnych z uzasadnieniem*

L.p.	Wymaganie niefunkcjonalne	Uzasadnienie
1.	Aplikacja 32-bitowa, konsolowa	Wybór podyktowany był 32-bitową wersją cURLa - planem na przyszłość jest aplikacja 64-bitowa.
2.	Język programowania C/C++	Wybór podyktowany był chęcią wykorzystania modułów w pracy inżynierskiej. Innym powodem jest próba uzyskania kompatybilności z systemem Windows i Linux..
3.	Komunikacja z kamerą IP powinna odbywać się za pomocą komunikatów HTTP oraz RTSP.	Sposób komunikacji podyktowany jest podsłuchaniem jej przy użyciu programu Wireshark.
4.	Klatki pobierane z kamery IP powinny być zapisane w formacie .png.	Wybrano format .png ze względu na konwersję zrzutu: niewielki rozmiar, kolor szary.
5.	Dane w pliku konfiguracyjnym powinny być zaszyfrowane symetrycznym szyfrem blokowym AES.	Szyfrowanie danych w pliku zapewnia, że osoba niepożądana ich nie wykradnie - zapewnienie bezpieczeństwa.
6.	Klucz użyty do szyfrowania danych w pliku konfiguracyjnym powinien powstać w wyniku utworzenia skrótu klucza podanego przez użytkownika po zastosowania funkcji skrótu SHA256.	Takie rozwiązanie pozwala zastosować symetryczny szyfr blokowy AES, który potrzebuje 256-bitowego klucza.
7.	Aplikacja powinna używać klasyfikatora kaskadowego LBP (ang. <i>Local Binary Patterns</i> ).	Klasyfikator LBP jest ok. 3 razy szybszy od klasyfikatora Haar, osiągając zbliżone rezultaty.
8.	Aplikacja powinna używać algorytmu rozpoznawania twarzy LBPH.	Algorytm LBPH osiąga najlepsze rezultaty spośród dostępnych algorytmów do rozpoznawania twarzy w bibliotece OpenCV (pozostałe to Eigenfaces oraz Fisherfaces).
9.	Angielska wersja językowa aplikacji.	Brak kodowania polskich znaków.
10.	Aplikacja powinna być kompatybilna z systemami Windows oraz Linux.	Chęć wykorzystania modułu do pracy inżynierskiej.

## 6. WYMAGANIA SPRZĘTOWE

Rozdział prezentuje zebrane w Tabeli 6.1. wymagania sprzętowe, które są niezbędne do prawidłowego uruchomienia aplikacji. Ze względu na brak sprawdzenia standardu ONVIF i kompatybilności innych modeli kamer z aplikacją, zaleca się użycia kamery IP o podanym modelu, ewentualnie kamery określonego producenta.

*Tabela 6.1. Wymagania sprzętowe*

L.p.	Wymaganie sprzętowe
1.	Kamera IP (Media-Tech MT4051)
2.	Komputer z systemem Windows lub Linux
3.	Sieć lokalna

## 7. ŚRODOWISKO, BIBLIOTEKI ZEWNĘTRZNE, NARZĘDZIA DODATKOWE

Rozdział prezentuje środowisko, w którym napisano aplikację, zewnętrzne biblioteki, które wykorzystano oraz narzędzia dodatkowe, które ułatwiają pracę z aplikacją.

*Tabela 7.1. Środowisko, biblioteki zewnętrzne, narzędzia dodatkowe*

L.p.	Środowisko
1.	Visual Studio 2017, wykorzystując język programowania C++ 17
L.p.	Biblioteki zewnętrzne
1.	cURL 7.54 - biblioteka do wysyłania zapytań HTTP - komunikacja z kamerą IP
2.	OpenCV 2.4.13.2 - biblioteka do przetwarzania obrazów oraz pobierania klatek z kamery IP
3.	<a href="https://github.com/B-Con/crypto-algorithms">https://github.com/B-Con/crypto-algorithms</a> - biblioteka zawierająca podstawową implementację standardowych algorytmów kryptograficznych (AES, SHA256)
L.p.	Narzędzia dodatkowe
1.	Connectify Hotspot 2017 w wersji darmowej ( <a href="http://www.connectify.me/">http://www.connectify.me/</a> ) - program do stworzenia sieci lokalnej

## 8. ZAGADNIENIA PROBLEMATYCZNE

Rozdział prezentuje wybrane zagadnienia problematyczne, z jakimi zetknięto się podczas zbierania materiałów potrzebnych do implementacji aplikacji oraz te, które wynikły w czasie implementacji projektu.

*Tabela 8.1. Zagadnienia problematyczne*

L.p.	Zagadnienie problematyczne
1.	Zainstalowanie najnowszej wersji biblioteki OpenCV
2.	Kompilacja biblioteki cURL
3.	Komunikacja z kamerą
4.	Obsługa pliku konfiguracyjnego
5.	Pobieranie klatek w locie
6.	Szyfrowanie danych przy użyciu symetrycznego szyfru blokowego AES
7.	Tworzenie pliku .csv
8.	Wybór algorytmu rozpoznawania twarzy oraz ustalenie progu rozpoznania osoby
9.	Wielokrotne rozpoznawanie tych samych osób w czasie jednej sesji
10.	Zapamiętanie wytrenowanego modelu
11.	Plik zawierający podsumowanie działania modułu rozpoznawania osób
12.	Tryb ograniczony czasowo podczas działania algorytmu rozpoznawania twarzy
13.	Mierzenie parametru FPS (liczba klatek na sekundę)

## 9. ROZWIĄZANIA ZAGADNIEŃ PROBLEMATYCZNYCH

Rozdział opisuje rozwiązania oraz opis zagadnień problematycznych.

*Tabela 9.1. Zagadnienia problematyczne wraz z ich rozwiązaniem*

L.p.	Zagadnienie problematyczne	Rozwiązanie
1.	Zainstalowanie najnowszej wersji biblioteki OpenCV	Początkowo pobrano najnowszą (3.2.0) wersję biblioteki z oficjalnej strony OpenCV. Niestety po instalacji i konfiguracji, okazało się, że moduły odpowiedzialne za rozpoznawanie twarzy nie znajdują się w oficjalnej paczce i należy je skompilować wraz z kodem źródłowym podstawowej

		wersji. Niestety po wielu próbach nie udało się uzyskać działającej wersji rozszerzonej biblioteki, dlatego zdecydowano się na wersję 2.4.13.2, ponieważ ona zawiera wbudowane potrzebne moduły do aplikacji i jest ona dość aktualna (data wydania: 16.12.2016).
2.	Kompilacja biblioteki cURL	Skompilowano bibliotekę cURL 7.54 w wersji 32-bitowej dla trybu Debug oraz Release z wykorzystaniem narzędzia x86 Native Tools Command Prompt for VS 2017 na podstawie poradnika wideo dla Visual Studio 2015: <a href="https://youtu.be/Eu7NFeg43T4">https://youtu.be/Eu7NFeg43T4</a> .
3.	Komunikacja z kamerą	Do komunikowania się z kamerą IP wykorzystano zewnętrzną bibliotekę sieciową cURL oraz OpenCV. cURL wykorzystano ją do: <ul style="list-style-type: none"> <li>- testowania połączenia z kamerą IP,</li> <li>- sterowania kamerą IP,</li> <li>- ustawiania kamery IP w pozycji domowej,</li> </ul> zaś OpenCV do: <ul style="list-style-type: none"> <li>- testowania połączenia z kamerą IP,</li> <li>- pobierania klatek z kamery IP.</li> </ul>
4.	Obsługa pliku konfiguracyjnego	<p><i>Utworzenie pliku</i></p> <p>Należało dokonać częściowej walidacji danych podawanych przez użytkownika - dane logowania do kamery IP są sprawdzane poprzez nawiązanie połączenia z kamerą. Adres IP sprawdzany jest wyrażeniem regularnym. Ważne jest również zadbanie o bezpieczeństwo tych danych - z tego względu użytkownik proszony jest o dodatkowy tajny klucz, z którego utworzony jest skrót przy użyciu funkcji skrótu SHA256, a następnie z użyciem otrzymanego skrótu dane szyfrowane są przy użyciu symetrycznego szyfru blokowego AES.</p> <p><i>Konfigurowanie połączenia z kamerą IP przy pomocy pliku</i></p> <p>Użycie pliku konfiguracyjnego wymaga podania tajnego klucza, z którego tworzony jest skrót i którym deszyfrowane są dane. Należało wpaść na pomysł utworzenia skrótu o długości 256 bitów - ze względu na długość klucza, którym szyfrowane/deszyfrowane są dane.</p>
5.	Pobieranie klatek w locie	Rozwiązanie tego problemu opiera się na bibliotece OpenCV. Należało najpierw połączyć się z kamerą IP, tworząc odpowiedni komunikat RTSP, w którym zawarte są informacje: login, hasło, adres IP, port, parametr live oraz kanał. Po pomyślnym nawiązaniu połączenia w pętli <code>while(true)</code> następuje pobieranie klatek za pomocą metody <code>read</code> i zapisywanie ich do zmiennej <code>image</code> typu <code>Mat</code> .
6.	Szyfrowanie/deszyfrowanie danych przy użyciu symetrycznego szyfru blokowego AES	Ważnym krokiem było wybranie odpowiedniej biblioteki, która pozwoli na wykorzystanie gotowych modułów do szyfrowania i deszyfrowania danych oraz pozwoli utworzyć skrót o pożądanej długości. Z tego względu zdecydowano się na bibliotekę dostępną na stronie: <a href="https://github.com/B-Con/crypto-algorithms">https://github.com/B-Con/crypto-algorithms</a> . Problematyczna okazuje się różna długość danych, które może wprowadzać użytkownik - należy pamiętać, że szyfrowanie/deszyfrowanie danych wymaga podzielenia ich na paczki po 32 bajty.

7.	Tworzenie pliku .csv	Trudnością było rekursywne przechodzenie po plikach oraz katalogach. W C++ 17 dostępny jest iterator <code>std::filesystem::recursive_directory_iterator</code> , który wraz z pętlą <code>for</code> rozwiązał ten problem. Wykorzystano <a href="http://stackoverflow.com/a/37494654">http://stackoverflow.com/a/37494654</a> .
8.	Wybór algorytmu rozpoznawania twarzy oraz ustalenie progu rozpoznania osoby	<p>Biblioteka OpenCV zawiera 3 zaimplementowane algorytmu do rozpoznawania twarzy:</p> <ul style="list-style-type: none"> <li>- Eigenfaces,</li> <li>- Fisherfaces,</li> <li>- Local Binary Patterns Histograms (LBPH).</li> </ul> <p>Wybór najlepszego z nich odbył się w sposób doświadczalny:</p> <ul style="list-style-type: none"> <li>- utworzono prostą bazę zdjęć (2 osoby, każda z nich zawierała 10 zdjęć twarzy różniących się m.in. nachyleniem twarzy, mimiką),</li> <li>- trenowano każdy z modeli z użyciem jednego z algorytmów,</li> <li>- uruchomiono moduł rozpoznawania twarzy z wykorzystaniem wytrenowanego modelu na czas ok. 1 minuty i wyświetlano na ekran bieżące rezultaty działania modułu tj. imię i nazwisko rozpoznanej osoby oraz odchylenie od oryginalnych zdjęć z korpusu.</li> </ul> <p>Algorytm LBPH osiąga najlepsze rezultaty spośród dostępnych algorytmów do rozpoznawania twarzy w bibliotece OpenCV. Na podstawie obserwacji wartości odchyleń podczas tego samego testu, ustalono również próg maksymalnego odchylenia od oryginalnych zdjęć z bazy na 100. W przypadku wykrycia twarzy, dla której wartość odchylenia jest większa, program zwróci komunikat o osobie nierozpoznanej.</p>
9.	Wielokrotne rozpoznawanie tych samych osób w czasie jednej sesji	<p>Okazuje się, że rozpoznawanie osób, które znajdują się w danym miejscu, może być utrudnione poprzez ich przemieszczanie się. Problem ten może być zauważalny zwłaszcza, gdy wybierzemy czasowe działanie aplikacji. Z tego względu w programie zastosowano dwa kontenery <code>map</code>. Pierwszy z nich: <code>std::map&lt;int, std::string&gt; peopleBase;</code> zawiera informacje o ludziach z bazy zdjęć znajdującej się w folderze <code>/screenshots</code> w postaci <code>&lt;etykieta, nazwa folderu&gt;</code>. Drugi kontener: <code>std::map&lt;int, std::string&gt; predictedPeople;</code> zawiera informacje o ludziach, których rozpoznał algorytm. Mapa ma postać <code>&lt;etykieta, nazwa folderu&gt;</code>. Po rozpoznaniu osoby przez algorytm zwracana jest etykieta przypisana do niej. Dzięki etykietom, która jest kluczem w obydwóch mapach, możemy łatwo sprawdzić, czy znajduje się ona w mapie, która zawiera osoby rozpoznane w czasie działania modułu rozpoznawania osób. Sprawdziwszy, że etykiety nie ma w mapie osób rozpoznanych, sprawdzamy wartość dla niej z mapy bazy ludzi i dodajemy wpis o noworozpoznanej osobie. Takie postępowanie eliminuje redundancję w mapie osób rozpoznanych.</p>
10.	Zapamiętanie wytrenowanego modelu	Do wykorzystania modułu rozpoznawania ludzi, konieczne jest wytrenowanie modelu. Im większa baza zdjęć w tym dłuższy czas potrzebny na wytrenowanie modelu. Z tego względu dano

		wybór użytkownikowi zapisania danych wytrenowanego modelu do pliku <i>trainFR.xml</i> (metoda z OpenCV <code>save(&lt;nazwa_pliku&gt;)</code> ). Przy wybraniu opcji trenowania modelu, posiadając plik z wytrenowanym modelem, można go wczytać (metoda z OpenCV <code>load(&lt;nazwa_pliku&gt;)</code> ) lub ponownie wytrenować model.
11.	Plik zawierający podsumowanie działania modułu rozpoznawania osób	Problematiczną kwestią jest zaznaczenie w pliku informacji o osobach nierozpoznanych. Należy wziąć pod uwagę przemieszczanie się ludzi w pomieszczeniu - jedna osoba nierozpoznana, może powodować przy każdym przemieszczeniu dodanie informacji o nierozpoznaniu. Z tego względu zdecydowano się na zaznaczenie informacji o próbach rozpoznania osoby zakończonych niepowodzeniem. Każde nierozpoznanie osoby powoduje inkrementowanie licznika osób nierozpoznanych, który, po zakończeniu procesu rozpoznawania, wykorzystywany jest do dodania na końcu pliku informacji o osobach nierozpoznanych.
12.	Tryb ograniczony czasowo podczas działania algorytmu rozpoznawania twarzy	Aby pobrać od użytkownika informację o czasie trwania działania modułu, wykorzystano rozwiązanie: <a href="http://stackoverflow.com/a/24271424">http://stackoverflow.com/a/24271424</a> , które zakłada wykorzystanie metody <code>sscanf</code> . Rozwiązanie to oprócz szybkiej konwersji odpowiedzi użytkownika przechowywanej w zmiennej typu string, pozwala na sprawdzenie poprawnego formatu wpisanych danych (HH:mm:ss). Do samego mierzenia czasu wykonywania algorytmu wykorzystano bibliotekę <code>chrono</code> ( <a href="http://stackoverflow.com/a/18685338">http://stackoverflow.com/a/18685338</a> ).
13.	Mierzenie parametru FPS (liczba klatek na sekundę)	Wbudowana metoda <code>get(CV_CAP_PROP_FPS)</code> działa poprawnie jedynie, gdy źródłem klatek jest plik wideo z dysku. W przypadku, gdy źródłem jest strumień klatek, należało wyznaczyć liczbę FPS ręcznie. Trzeba najpierw połączyć się z kamerą IP i zmierzyć czas pobrania 100 klatek w sekundach. Na koniec należy podzielić liczbę 100 przez zmierzoną liczbę sekund i w ten sposób otrzymano szukany parametr. Powyższe rozwiązanie oparto na kodzie zawartym w artykule: <a href="https://www.learnopencv.com/how-to-find-frame-rate-or-frame-s-per-second-fps-in-opencv-python-cpp/">https://www.learnopencv.com/how-to-find-frame-rate-or-frame-s-per-second-fps-in-opencv-python-cpp/</a> .

## 10. WYBRANE FRAGMENTY KODU

W tym rozdziale znajdują się wybrane fragmenty kodu wraz z krótkim opisem. Fragmenty kodu przedstawione są w postaci *Listingów*.

**Listing 10.1.** prezentuje sposób pobierania klatek. Wykorzystano do tego protokół RTSP (kontrola strumieni multimedialnych). W wiadomości przesyłanej jest żądanie RTSP z loginem, hasłem, adresem IP kamery IP i wybranym kanałem - kanał 0 (pobierane klatki w rozdzielczości HD).

```

const std::string videoStreamAdress = "rtsp://" + this->USERPWD + "@" +
this->IPAddress + "/live/ch0";
...
if (!vcap.open(videoStreamAdress))
{
    ...
    while (true)
    {
        ...
        if (vcap.read(image))
        {
            ...
            cv::waitKey(1);
        }
    }
}

```

**Listing 10.1.** Nawiązywanie połączenia i pobieranie klatek z kamery IP

**Listing 10.2.** prezentuje testowanie połączenia z kamerą IP. Po tym, jak dane takie jak login, hasło oraz adres IP kamery zostaną wczytane od użytkownika lub z pliku konfiguracyjnego, następuje utworzenie na ich podstawie komunikatu HTTP z zapytaniem o dostępność urządzenia oraz wysłanie go z oczekiwaniem na odpowiedź. Jeżeli wczytane/wprowadzone dane są niepoprawne, kod odpowiedzi jest różny od 200, co oznacza niepowodzenie. Kod odpowiedzi równy 200 oznacza sukces i możliwość korzystania z głównych funkcjonalności programu.

```

CURLcode res;
curl_easy_setopt(curl, CURLOPT_USERPWD, USERPWD.c_str());
curl_easy_setopt(curl, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, false);
curl_easy_setopt(curl, CURLOPT_USERAGENT, "Mozilla/5.0 (Windows NT 6.3;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.28
Safari/537.36");
std::string URL = "http://" + IPAddress +
"/hy-cgi/ptz.cgi?cmd=ptzctrl&act=";
curl_easy_setopt(curl, CURLOPT_URL, URL.c_str());
curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, Utilities::write_data);
res = curl_easy_perform(curl);
if (res == 0)
{
    int httpCode = 0;
    res = curl_easy_getinfo(curl, CURLINFO_HTTP_CODE, &httpCode);
    if (httpCode == 200)
    {
        curl_easy_cleanup(curl);
        return true;
    }
}
curl_easy_cleanup(curl);

```

**Listing 10.2.** Testowanie połączenia z kamerą IP

**Listing 10.3.** prezentuje tworzenie pliku csv. Każdy wers pliku składa się z tych samych elementów: screenshots\[nazwa folderu\[nazwa zdjecia].png;[etykieta]. Do

odczytania nazw folderów wykorzystano rekurencję. Etykietą są kolejne liczby naturalne (włączając 0).

```
bool ImgProc::createCSV()
{
    std::fstream file;
    file.open("corp.csv", std::ios::out);
    if (file.good() != true)
    {
        std::cout << "Error!" << std::endl;
        return false;
    }

    std::string path = "screenshots/";
    int numberPerson = -1;
    std::string folderName = "";
    for (auto & p :
std::experimental::filesystem::recursive_directory_iterator(path))
    {
        if (!std::experimental::filesystem::is_directory(p.path()))
        {
            file << p << ";" << numberPerson << std::endl;
        }
        else
        {
            numberPerson++;
            peopleBase[numberPerson] = p.path().string().substr(12,
p.path().string().length()-12);
        }
    }
    file.close();
    return true;
}
```

**Listing 10.3.** Tworzenie pliku csv

**Listing 10.4.** prezentuje metodę wyznaczania wartości parametru FPS kamery IP. Po nawiązaniu połączenia z kamerą IP następuje zmierzenie czasu pobrania 100 klatek z kamery IP. Końcowa wartość obliczana jest jako iloraz liczby pobranych klatek przez potrzebny do tego czas w sekundach.

```
double Camera::getFPS()
{
    std::cout << "Configuration of IP camera settings. Wait... " <<
std::endl;
    Utilities::dashes();
    cv::VideoCapture vcap;
    cv::Mat image;
    const std::string videoStreamAddress = "rtsp://" + this->USERPWD + "@" +
this->IPAddress + "/live/ch1";

    if (!vcap.open(videoStreamAddress))
    {
        std::cout << "Error opening video stream or file" << std::endl;
    }
}
```



```

        return 0;
    }
    int num_frames = 100;
    time_t start, end;
    time(&start);
    for (size_t i = 0; i < num_frames; i++)
    {
        vcap >> image;
    }
    time(&end);

    double seconds=difftime(end, start);
    vcap.release();
    return num_frames / seconds;
}

```

**Listing 10.4.** Obliczanie wartości parametru FPS kamery IP

**Listing 10.5.** prezentuje sposób rozwiązania podzielenia danych wprowadzanych przez użytkownika na paczki po 32 bajty. Zmienna `secDataPointer` początkowo zawiera rozmiar danych. Po zaszyfrowaniu paczki, odejmowane są od niej 32 bajty, jeżeli otrzymana wartość jest ujemna, proces szyfrowania jest zakończony. Przy każdym przejściu pętli, kopiowana jest do tablicy bajtów `plaintext` o rozmiarze 32 kolejna paczka bajtów wiadomości do zaszyfrowania. Dzięki wykorzystaniu zmiennej `packetNo` można wykryć, od którego bajtu zaczyna się paczka do zaszyfrowania. Metoda `aes_encrypt_cbc(...)` dokonuje szyfrowania paczki danych i zwraca wartość poprawnego lub błędnego działania. Każda zaszyfrowana paczka jest zapisywana do pliku konfiguracyjnego.

```

while (secDataPointer > 0)
{
    memcpy(plaintext, ByteSecretData + packetNo * 32, 32);

    if (aes_encrypt_cbc(plaintext, 32, enc_buf, key_schedule, 256,
initialValue) == false)
    {
        fclose(file);
        printf("Something went wrong with encryption!");
        return;
    }
    fwrite(&enc_buf, 1, 32, file);
    packetNo++;
    secDataPointer -= 32;
}

```

**Listing 10.5.** Szyfrowanie danych z użyciem symetrycznego szyfru blokowego AES - tryb CBC

**Listing 10.6.** przedstawia deszyfrowanie danych. Wykorzystuje ono podobny sposób działania, co szyfrowanie - to samo zastosowanie zmiennej `secDataPointer` oraz `packetNo`. Różnicą jest zapisywanie odszyfrowanych danych do pamięci.

```

while (dataPointer > 0)
{
    memcpy(enc_buf, data + packetNo * 32, 32);
}

```

```
    aes_decrypt_cbc(enc_buf, 32, dec_buf, key_schedule, 256, initialValue);  
    std::string plaintextTemp((char *)dec_buf);  
    plaintext += plaintextTemp.substr(0,32);  
    packetNo++;  
    dataPointer -= 32;  
}
```

**Listing 10.6.** Deszyfrowanie danych z użyciem symetrycznego szyfru blokowego AES - tryb CBC

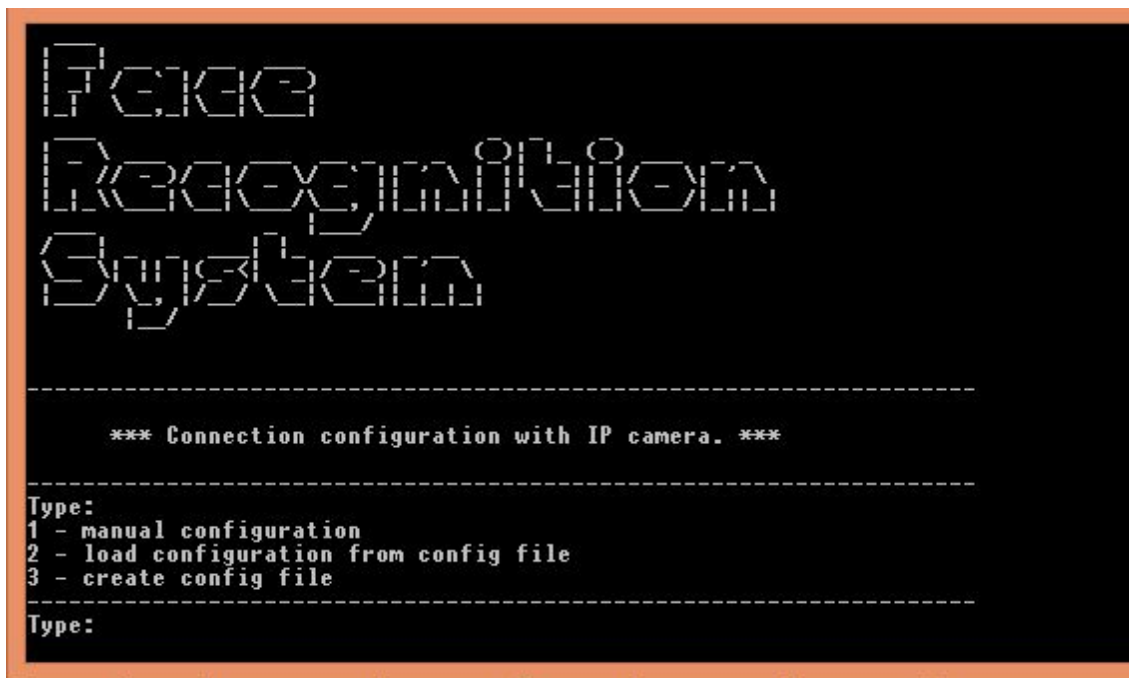
## 11. INSTRUKCJA UŻYTKOWANIA APLIKACJI

W tym rozdziale znajduje się instrukcja użytkowania aplikacji. Aplikacja posiada prosty interfejs graficzny. Użytkownik informowany jest o dostępnych komendach, które może wpisać na każdym etapie działania aplikacji. Aplikacja domagając się działania ze strony użytkownika, informuje go poprzez komendę `Type`.

Instrukcja użytkownika będzie miała postać zrzutów ekranu z opisem dostępnych komend wybranych przez użytkownika. Podzielona jest na dwie części, które wyodrębnione są w programie: konfiguracja połączenia z kamerą IP oraz działanie na modułach dostępnych po zalogowaniu - moduły *Face Recognition System*.

### KONFIGURACJA POŁĄCZENIA Z KAMERĄ IP

*Rysunek 11.1.* prezentuje ekran powitalny aplikacji. Użytkownik musi rozpocząć od konfiguracji połączenia z kamerą IP. Ma do wyboru 3 opcje: konfigurację manualną, załadowanie konfiguracji z pliku lub utworzenie pliku konfiguracyjnego. Zdecydowawszy się na daną opcję, należy wybrać jedną z podanych cyfr. Każda z opcji kończy się zalogowaniem do kamery IP, co związane jest z posiadaniem dostępu do modułów programu.



*Rys.11.1. Ekran powitalny aplikacji*

## KONFIGURACJA MANUALNA

Wybrawszy opcję 1., użytkownik proszony jest o podanie danych, potrzebnych do nawiązania komunikacji z kamerą IP, tj. adres IP kamery, login oraz hasło, przy czym podawszy adres IP, sprawdzana jest poprawność jego zapisu. Użytkownik jest informowany o nawiązaniu komunikacji, czy problemie z nawiązaniem komunikacji. Napotkawszy problem, możemy ponownie wybrać opcję konfiguracji połączenia z kamerą IP. Konfigurowanie połączenia zakończone sukcesem, pozwoli na przejście do kolejnych modułów programu.

```
-----  
*** Connection configuration with IP camera. ***  
-----  
Type:  
1 - manual configuration  
2 - load configuration from config file  
3 - create config file  
-----  
Type: 1  
-----  
Type camera adress IP compatible with format: xxx.xxx.xxx.xxx  
Type: 192.168.18.102  
-----  
Type login: admin  
-----  
Type password:  
-----  
Test connection. Wait...  
-----  
Test connection has been succeeded  
-----  
*** Welcome in Face Recognition System ***  
-----  
HELP - C  
-----  
Type:
```

*Rys.11.2. Konfiguracja manualna*

## KONFIGURACJA Z UŻYCIEM PLIKU KONFIGURACYJNEGO

Opcja 2. przyspiesza proces logowania, gdyż korzysta z pliku konfiguracyjnego. Ważne jest, by przed skorzystaniem z tej opcji, utworzyć plik konfiguracyjny. Mając wygenerowany wcześniej plik, podajemy tajny klucz i czekamy na rezultat nawiązania połączenia. Po poprawnym skonfigurowaniu połączenia z kamerą IP dostępne są kolejne moduły programu. W przypadku problemu z nawiązaniem połączenia z kamerą IP możemy ponownie wybrać jedną z trzech opcji konfiguracyjnych.

```
-----
*** Connection configuration with IP camera. ***
-----
Type:
1 - manual configuration
2 - load configuration from config file
3 - create config file
-----
Type: 2
-----
Type secret key:
-----
Test connection. Wait...
-----
Test connection has been succeeded
-----
*** Welcome in Face Recognition System ***
-----
HELP - C
-----
Type:
```

*Rys.11.3. Konfiguracja z użyciem pliku konfiguracyjnego*

## UTWORZENIE PLIKU KONFIGURACYJNEGO

Opcja 3. zawiera utworzenie pliku konfiguracyjnego, dzięki któremu będziemy mogli zalogować się do kamery IP. Chcąc utworzyć plik konfiguracyjny, należy najpierw podać dane, które podajemy w przypadku konfiguracji manualnej, tj. adres IP kamery, login oraz hasło. Po podaniu danych sprawdzane jest połączenie z kamerą. W przypadku niepowodzenia, użytkownik będzie mógł ponownie wybrać jedną z 3 opcji konfiguracyjnych. Poprawne nawiązanie połączenia skutkuje zapytaniem użytkownika o tajny klucz, użyty do zaszyfrowania danych w pliku konfiguracyjnym. Tylko podanym kluczem można odszyfrować dane z pliku. Gdy użytkownik poda tajny klucz, informowany jest o zaszyfrowaniu danych i zapisaniu ich w pliku `config.bin`, a następnie może skorzystać z modułów programu.

```
-----
*** Connection configuration with IP camera. ***
-----
Type:
1 - manual configuration
2 - load configuration from config file
3 - create config file
-----
Type: 3
-----
Type camera adress IP compatible with format: xxx.xxx.xxx.xxx
Type: 192.168.18.102
-----
Type login: admin
-----
Type password:
-----
Test connection. Wait...
-----
Test connection has been succeeded
-----
Type key:
-----
Your data was encrypted with AES algorithm and saved in config.bin
-----
*** Welcome in Face Recognition System ***
-----
HELP - C
-----
Type:
```

Rys.11.4. Utworzenie pliku konfiguracyjnego

Plik konfiguracyjny to plik binarny o stałej nazwie: config.bin. Zapisany jest zgodnie z podaną ścieżką:  
?\\FaceRecognitionSystem\\FaceRecognitionSystem\\FaceRecognitionSystem, gdzie ? należy uzupełnić o miejsce, w którym znajduje się aplikacja.

config.bin	2017-05-18 12:54	Plik BIN	1 KB
------------	------------------	----------	------

Rys.11.5. Widok na plik konfiguracyjny config.bin

Zawartość pliku konfiguracyjnego jest zaszyfrowana. Tylko dzięki mając odpowiedni klucz, możemy go odszyfrować.



Rys.11.6. Widok na zawartość pliku konfiguracyjnego config.bin

## MODUŁY FACE RECOGNITION SYSTEM

Aplikacja posiada różne moduły, do których użytkownik ma dostęp po zalogowaniu zakończonym sukcesem. Użytkownik może sterować kamerą, robić zdjęcia do korpusu, zliczać ludzi, którzy znajdują się w kadrze kamery IP, wytrenować model, rozpocząć

rozpoznawanie ludzi, czy wyświetlać listę pomocy. Każdy z modułów może być wykonywany dowolną liczbę razy, ale nie jest możliwe wybranie modułów do pracy równoległej. Konieczne do zapoznania się w aplikacji z dostępnymi modułami jest wyświetlenie pomocy.

## UTWORZENIE PLIKU KONFIGURACYJNEGO

Użytkownik może wybrać opcję wyświetlenia pomocy. W lewej kolumnie pomocy dostępne są litery, które należy wprowadzać z klawiatury, by uzyskać efekt opisany w prawej kolumnie. Wielkość litery jest dowolna. Polecenia: W, S, A, D, H służą do sterowania kamerą IP i ich potwierdzenie wyboru nie pojawi się w oknie konsoli. Wybór pozostałych opcji skutkuje komentarzem wyświetlanym na konsoli lub w przypadku komendy Q - zamknięciem aplikacji.

```
*** Welcome in Face Recognition System ***
-----
HELP - C
-----
Type: c
-----
<BUTTON>      <Description>
W             Move the camera up
S             Move the camera down
A             Move the camera left
D             Move the camera right
H             Move the camera to home position
P             Take pictures
E             Count people
T             Train face recognizer
R             Start face recognizer
C             Show command list
Q             Exit the application
-----
Type:
```

Rys.11.7. Pomoc

## WYKONYWANIE ZDJĘĆ DO KORPUSU

Wybrawszy opcję P, użytkownik znajduje się w module robienia zdjęć. Użytkownik proszony jest o podanie nazwy folderu, w którym znajdują się zdjęcia. W przypadku robienia zdjęć do korpusu, pożądane jest, by nazywać foldery w formacie <imię nazwisko>, jednakże nie ma wymuszenia na użytkowniku takiej praktyki. Zalecaną liczbą zdjęć do wykonania jest 10. Wykonane zdjęcia mają określony format i są w odcieniu szarości. O rozpoczęciu procesu robienia zdjęć użytkownik informowany jest dłuższym sygnałem dźwiękowym. Krótki sygnał dźwiękowy generowany jest po wykonaniu zdjęcia. O zakończeniu robienia zdjęć użytkownik informowany jest długim sygnałem dźwiękowym. Sygnały dźwiękowe generowane są na komputerze, na którym uruchomiona jest aplikacja.

```

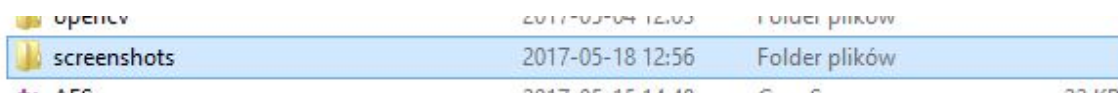
*** Welcome in Face Recognition System ***

HELP - C
Type: p
Type directory name where the pictures will be stored
Type: imie nazwisko
Type number of pictures that will be made
Type: 10
Configuration of IP camera settings. Wait...
Screenshots are being taking...
Success! Your screenshot/s in dir imie nazwisko
Type:

```

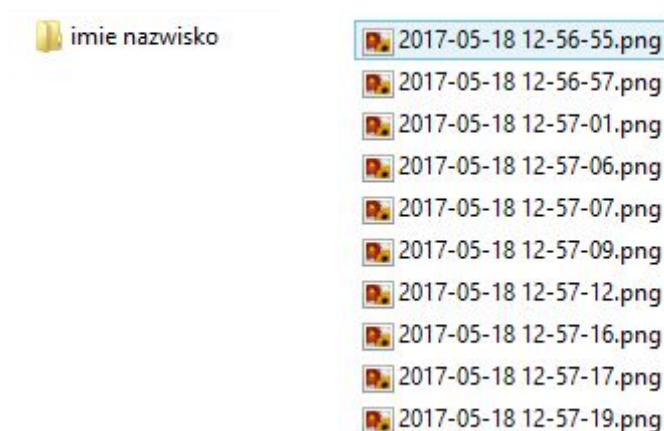
Rys.11.8. Wykonywanie zdjęć do korpusu

Folder screenshots, jeżeli nie istnieje, utworzony zostaje przy pierwszym uruchomieniu modułu wykonywania zdjęć do korpusu i zapisany jest zgodnie z podaną ścieżką: ?\FaceRecognitionSystem\FaceRecognitionSystem\FaceRecognitionSystem, gdzie ? należy uzupełnić o miejsce, w którym znajduje się aplikacja.



Rys.11.9. Podgląd na folder screenshots

Zawartość folderu screenshots to foldery o nazwach nadanych przez użytkownika. Foldery te zawierają zdjęcia, które tworzą korpus. Nazywane są one datą wykonania zdjęcia.

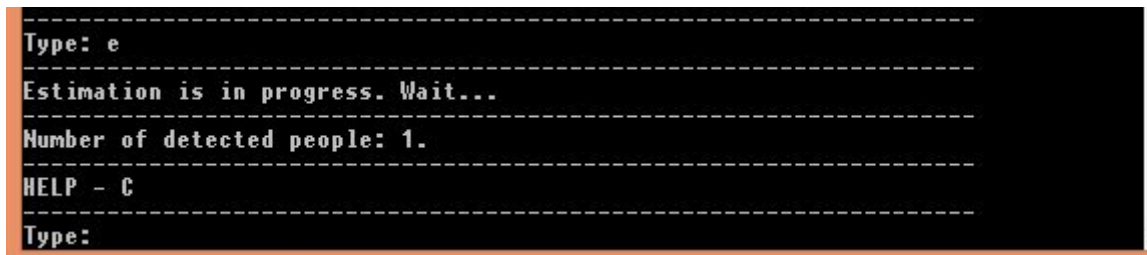


Rys.11.10. Widok na przykładową zawartość folderu screenshots wraz z jego zawartością



## ZLICZANIE LUDZI

Wybrawszy opcję E, użytkownik znajdzie się w module służącym do zliczenia ludzi. Ze względu na to, że możliwe jest nierozpoznanie wszystkich ludzi przy jednokrotnym wywołaniu modułu (przechylenie głowy, zasłonięcie twarzy, odwrócenie się, wykonywanie gwałtownych ruchów), po wybraniu opcji następuje 10 krotne powtórzenie operacji zliczania osób. Opiera się ono na liczeniu rozpoznania twarzy, które znajdują się w kadrze kamery oraz obliczeniu średniej wartości, która jest zaokrąglana. Użytkownik informowany jest o rezultacie działania modułu poprzez wyświetlenie informacji na ekranie.



```
-----
Type: e
-----
Estimation is in progress. Wait...
-----
Number of detected people: 1.
-----
HELP - C
-----
Type:
```

*Rys.11.11. Zliczanie ludzi*

## TRENOWANIE FACE RECOGNIZER

Wybrawszy opcję T, znajdziemy się w module trenowania modelu Face Recognizer. Początkowo użytkownik informowany jest o rozpoczęciu trenowania modelu i proszony o oczekiwanie na jego zakończenie.

**Uwaga!** Mając wytrenowany i zapisany model do pliku `trainFR.xml` będziemy poproszeni o zdecydowanie, czy chcemy załadować model z pliku, czy wytrenować model ponownie.

Jeżeli model zostanie poprawnie wytrenowany, użytkownik jest informowany o sukcesie. Następnie proszeni jesteśmy o zdecydowanie, czy chcemy zapisać model do pliku `trainFR.xml`, jeżeli wybierzemy Y, musimy poczekać na zapis modelu do pliku. Poprawne zapisanie modelu do pliku skutkuje wyświetleniem na ekranie informacji o sukcesie.

```

-----
Type: t
-----
Face recognizer model is being trained. Wait...
-----
Face recognizer model has been trained successfully!
-----
Do you want to save training data in trainFR.xml file?
Type Yes[Y]/No[N]: y
-----
Training data is being saved in trainFR.xml file. Wait...
-----
Training data has been saved successfully!
-----
HELP - C
-----
Type:

```

Rys.11.12. Trenowanie modelu Face Recognizer

## ROZPOZNAWANIE LUDZI

Wybrawszy opcję R, znajdziemy się w module rozpoznawania ludzi. Użytkownik proszony jest o wybranie jednego z dwóch trybów - tryb nieskończony działania aplikacji (1) oraz tryb działający w określonym czasie (2).

Wybrawszy opcję (1), należy poczekać na zakończenie procedur konfiguracyjnych, po których rozpocznie się rozpoznawanie ludzi. Osoby rozpoznane wyświetlane są na ekranie z podaniem wartości określającej odchylenie od wartości 0, która oznacza 100% zgodność osoby rozpoznanej ze zdjęciem z bazy ludzi.

Tryb czasowy różni się tym, że użytkownik musi podać czas działania aplikacji zgodnie z określonym formatem. Po tym okresie, działanie modułu rozpoznawania ludzi zostaje zakończony.

```

-----
HELP - C
-----
Type:
1 - run FaceRecognizer in endless mode
2 - run FaceRecognizer in time mode
-----
Type: 2
-----
Type duration time of FaceRecognizer in format HH:mm:ss
Type: 00:00:30
-----
Configuration procedures are ongoing. Wait...
-----
People recognition module has started...
-----
Predicted person: Robert Kazimierczak with confidence: 980.68.
Predicted person: Monika Gradzka with confidence: 2733.82.
Predicted person: Robert Kazimierczak with confidence: 1494.6.
Predicted person: Monika Gradzka with confidence: 1361.03.
Predicted person: Robert Kazimierczak with confidence: 1251.41.
Predicted person: Monika Gradzka with confidence: 632.002.
Predicted person: Monika Gradzka with confidence: 2165.09.
Predicted person: Monika Gradzka with confidence: 915.845.
-----
Face recognition has been ended.
-----
HELP - C
-----
Type:

```

Rys.11.13. Rozpoznawanie ludzi - tryb czasowy

**Uwaga!** Ważne jest aby przed rozpoczęciem modułu rozpoznawania ludzi wytrenować model, w przeciwnym przypadku zostanie wyświetlony komunikat informujący o niewytrenowaniu modelu.

```
-----
Type: Error! Model is not trained!
-----
```

Rys.11.14. Komunikat o niewytrenowaniu modelu

## HELP

Użytkownik przed lub po zakończeniu działania modułów może wyświetlić pomoc na ekranie, komenda C, dzięki której może dowiedzieć się o komendach.

```
-----
HELP - C
-----
Type: c
-----
<BUTTON>      <Description>
W             Move the camera up
S             Move the camera down
A             Move the camera left
D             Move the camera right
H             Move the camera to home position
P             Take pictures
E             Count people
T             Train face recognizer
R             Start face recognizer
C             Show command list
Q             Exit the application
-----
```

Rys.11.15. Widok pomocy

## WYJŚCIE Z PROGRAMU

Użytkownik może zakończyć działanie aplikacji poprzez wybranie komendy Q lub wybranie przycisku X zamykającego konsolę.

## 12. PLANY NA ROZWINIĘCIE APLIKACJI

Aplikacja przeznaczona jest do dołączenia, jako moduł, do projektu monitoringu domu, który będzie zrealizowany jako praca inżynierska. Moduły, które można dołączyć do aplikacji zaprezentowano w Tabeli 12.1..

Tabela 12.1. Kierunki rozwoju aplikacji

L.p.	Kierunek rozwoju
1.	Interfejs graficzny, który umożliwi użytkownikowi wygodne i szybkie nawigowanie w aplikacji
2.	Aplikacji z wykorzystaniem wątków - np. równoczesne sterowanie kamerą i rozpoznawanie osób
3.	Wykorzystanie mikrofonu i głośników kamery do komunikacji
4.	Sygnal dźwiękowy nadawany z kamery, a nie z komputera
5.	Czasowy obieg kamery względem ustawionych presetów
6.	Nagrywania w nocy i sprawdzenie poprawności rozpoznawania ludzi w takich warunkach
7.	Rozpoznawanie obiektów
8.	Użycie kamery zewnętrznej do rozpoznawania aut, tablic rejestracyjnych
9.	Zapoznanie się z kompatybilnością różnych modeli kamer - zapoznanie się ze standardem ONVIF i sprawdzenie interfejsów innych kamer
10.	Kompatybilność aplikacji z systemem operacyjnym Linux.

## 13. TESTOWANIE APLIKACJI

Rozdział prezentuje wybrane testy aplikacji.

Na *Rysunku 13.1.* pokazano test rozpoznawania twarzy w czasie rzeczywistym. W korpusie znajdowały się dwa foldery z 10 zdjęciami twarzy Moniki Grądzkiej i Roberta Kazimierczaka. Zdjęcia robione do korpusy robione były przy naturalnym oświetleniu. Należy pamiętać, że im mniejsza wartość *confidence*, tym większa podobieństwo między osobą rozpoznaną z klatki, a zdjęciem znajdującym się w korpusie.



Rys.13.1. Test modułu rozpoznawania twarzy w czasie rzeczywistym



*Rysunki 13.2 - 13.4* przedstawiają wybraną wymianę komunikatów HTTP pomiędzy komputerem klienckim, a kamerą IP w trakcie sterowania kamerą z poziomu aplikacji. *Rysunki 13.5* oraz *13.6* pokazują przykładowe komunikaty podczas pobierania kolejnych klatek z kamery IP. Do podsłuchania komunikacji wykorzystano program *Wireshark*. Adres IP komputera klienckiego to 192.168.18.50, zaś kamery IP - 192.168.18.102.

*Rysunek 13.2.* przedstawia zapytanie *HTTP GET* wysyłane z komputera klienckiego do kamery IP. Zaznaczone zapytanie zawiera *URI: /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left*, które oznacza komendę rozpoczęcia ruchu kamery w lewo.

No.	Time	Source	Destination	Protocol	Length	Info
1261	2017-05-28 13:14:13.269011	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1263	2017-05-28 13:14:13.311123	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1271	2017-05-28 13:14:13.350387	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1273	2017-05-28 13:14:13.391184	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1281	2017-05-28 13:14:13.665857	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1283	2017-05-28 13:14:13.711790	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1291	2017-05-28 13:14:13.746479	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1293	2017-05-28 13:14:13.799460	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1301	2017-05-28 13:14:14.335241	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1
1303	2017-05-28 13:14:14.379404	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1311	2017-05-28 13:14:14.416655	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1

```

> Frame 1261: 264 bytes on wire (2112 bits), 264 bytes captured (2112 bits) on interface 0
> Ethernet II, Src: HonHaiPr_d8:8b:69 (10:08:b1:d8:8b:69), Dst: Shenzhen_79:1d:51 (28:f3:66:79:1d:51)
> Internet Protocol Version 4, Src: 192.168.18.50, Dst: 192.168.18.102
> Transmission Control Protocol, Src Port: 3289, Dst Port: 80, Seq: 1, Ack: 1, Len: 210
> Hypertext Transfer Protocol
  > GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1\r\n
    Host: 192.168.18.102\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.28 Safari/537.36\r\n
    Accept: */*\r\n
    \r\n
    [Full request URI: http://192.168.18.102/hy-cgi/ptz.cgi?cmd=ptzctrl&act=left]
    [HTTP request 1/1]
    [Response in frame: 1263]

```

*Rys.13.2. Zapytanie HTTP GET bez autoryzacji*

Wynikiem niewysłania danych logowania potrzebnych do komunikacji z kamerą IP jest odpowiedź *HTTP 401 Unauthorized*. Na *Rysunku 13.3.* można zauważyć m.in. rodzaj wymaganej autoryzacji (*Digest*) oraz ciąg pseudolosowy do wykorzystania podczas kolejnej próby autoryzacji.

No.	Time	Source	Destination	Protocol	Length	Info
1261	2017-05-28 13:14:13.269011	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1263	2017-05-28 13:14:13.311123	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1271	2017-05-28 13:14:13.350387	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1273	2017-05-28 13:14:13.391184	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1281	2017-05-28 13:14:13.665857	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1283	2017-05-28 13:14:13.711790	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1291	2017-05-28 13:14:13.746479	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1293	2017-05-28 13:14:13.799460	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1301	2017-05-28 13:14:14.335241	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1
1303	2017-05-28 13:14:14.379404	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1311	2017-05-28 13:14:14.416655	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1

```

> Frame 1263: 664 bytes on wire (5312 bits), 664 bytes captured (5312 bits) on interface 0
> Ethernet II, Src: Shenzhen_79:1d:51 (28:f3:66:79:1d:51), Dst: HonHaiPr_d8:8b:69 (10:08:b1:d8:8b:69)
> Internet Protocol Version 4, Src: 192.168.18.102, Dst: 192.168.18.50
> Transmission Control Protocol, Src Port: 80, Dst Port: 3289, Seq: 1, Ack: 211, Len: 610
> Hypertext Transfer Protocol
  > HTTP/1.1 401 Unauthorized\r\n
    WWW-Authenticate: Digest realm="IPCamera Login", nonce="2ce9e06c15f8459ddcc1b2b1a4fcaee5", qop="auth"\r\n
    Content-Type: text/html\r\n
    Content-Length: 351\r\n
    Connection: close\r\n
    Date: Sun, 28 May 2017 11:14:13 GMT\r\n
    Server: lighttpd/1.4.35\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.042112000 seconds]
    [Request in frame: 1261]

```

*Rys.13.3. Odpowiedź HTTP 401 Unauthorized*

*Rysunek 13.4* przedstawia zapytanie *HTTP GET* wysyłane z komputera klienckiego do kamery IP. Różnicą pomiędzy tym zapytaniem, a zapytaniem z *Rysunku 13.2.* to

bezpieczna wersja autoryzacji - *Digest*. Zaletą tego rozwiązania jest to, że dane nie są przesyłane w formie jawnej, zaś sama komunikacja nie wymaga użycia SSL w celu zapewnienia bezpieczeństwa. Po wysłaniu poprawnych danych logowania do kamery IP, komputer kliencki otrzymuje odpowiedź *HTTP 200 OK* (następny komunikat).

No.	Time	Source	Destination	Protocol	Length	Info
1261	2017-05-28 13:14:13.269011	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1263	2017-05-28 13:14:13.311123	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1271	2017-05-28 13:14:13.350387	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1
1273	2017-05-28 13:14:13.391184	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1281	2017-05-28 13:14:13.665857	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1283	2017-05-28 13:14:13.711790	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1291	2017-05-28 13:14:13.746479	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=stop HTTP/1.1
1293	2017-05-28 13:14:13.799460	192.168.18.102	192.168.18.50	HTTP	231	HTTP/1.1 200 OK
1301	2017-05-28 13:14:14.335241	192.168.18.50	192.168.18.102	HTTP	264	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1
1303	2017-05-28 13:14:14.379404	192.168.18.102	192.168.18.50	HTTP/X...	664	HTTP/1.1 401 Unauthorized
1311	2017-05-28 13:14:14.416655	192.168.18.50	192.168.18.102	HTTP	520	GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=down HTTP/1.1

```

> Frame 1271: 520 bytes on wire (4160 bits), 520 bytes captured (4160 bits) on interface 0
> Ethernet II, Src: HonHaiPr_d8:8b:69 (10:08:b1:d8:8b:69), Dst: Shenzhen_79:1d:51 (28:f3:66:79:1d:51)
> Internet Protocol Version 4, Src: 192.168.18.50, Dst: 192.168.18.102
> Transmission Control Protocol, Src Port: 3290, Dst Port: 80, Seq: 1, Ack: 1, Len: 466
> Hypertext Transfer Protocol
  > GET /hy-cgi/ptz.cgi?cmd=ptzctrl&act=left HTTP/1.1\r\n
    Host: 192.168.18.102\r\n
    [truncated] Authorization: Digest username="admin",realm="IPCamera Login",nonce="2ce9e06c15f8459ddcc1b2b1a4fcae5",uri="/hy-cgi/ptz.cgi?cmd=ptzctrl&act=left",c
    User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.28 Safari/537.36\r\n
    Accept: */*\r\n
    \r\n
    [Full request URI: http://192.168.18.102/hy-cgi/ptz.cgi?cmd=ptzctrl&act=left]
    [HTTP request 1/1]
    [Response in frame: 1273]

```

Rys.13.4. Zapytanie HTTP GET z autoryzacją typu Digest

Rysunek 13.5. przedstawia komunikat *RTSP PLAY* wysyłany z komputera klienckiego do kamery IP. Oznacza on rozpoczęcie odbioru klatek z kamery. Ważne parametry to: *URL: rtsp://192.168.18.102:554/live/ch1/* - źródło klatek (kanał 1 oznacza jakość klatek 480p) oraz *Authorization: Basic*, czyli podstawowa forma autoryzacji wykorzystująca *Base64* jako formę kodowania znaków (1 znak - 6 bitów).

No.	Time	Source	Destination	Protocol	Length	Info
964	2017-05-28 14:37:16.279374	192.168.18.50	192.168.18.102	RTSP	246	PLAY rtsp://192.168.18.102:554/live/ch1/ RTSP/1.0
965	2017-05-28 14:37:16.367771	192.168.18.102	192.168.18.50	RTSP	447	Reply: RTSP/1.0 200 OK
969	2017-05-28 14:37:18.166556	192.168.18.102	192.168.18.50	RTP	1371	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=360, Time=11442330
970	2017-05-28 14:37:18.166816	192.168.18.102	192.168.18.50	RTCP	142	Sender Report Source description Application specific ( qtsi ) subtype=1
972	2017-05-28 14:37:18.174167	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=361, Time=11442330
973	2017-05-28 14:37:18.174538	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=362, Time=11442330
975	2017-05-28 14:37:18.174727	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=363, Time=11442330
976	2017-05-28 14:37:18.194944	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=364, Time=11442330
978	2017-05-28 14:37:18.199171	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=365, Time=11442330
979	2017-05-28 14:37:18.199276	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=366, Time=11442330
981	2017-05-28 14:37:18.199577	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=367, Time=11442330
982	2017-05-28 14:37:18.201744	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=368, Time=11442330

```

> Frame 964: 246 bytes on wire (1968 bits), 246 bytes captured (1968 bits) on interface 0
> Ethernet II, Src: HonHaiPr_d8:8b:69 (10:08:b1:d8:8b:69), Dst: Shenzhen_79:1d:51 (28:f3:66:79:1d:51)
> Internet Protocol Version 4, Src: 192.168.18.50, Dst: 192.168.18.102
> Transmission Control Protocol, Src Port: 3595, Dst Port: 554, Seq: 1098, Ack: 2281, Len: 192
> Real Time Streaming Protocol
  > Request: PLAY rtsp://192.168.18.102:554/live/ch1/ RTSP/1.0\r\n
    Method: PLAY
    URL: rtsp://192.168.18.102:554/live/ch1/
    Range: npt=0.000-\r\n
    CSeq: 7\r\n
    User-Agent: Lavf57.2.100\r\n
    Session: 9202640500397323178
    Authorization: Basic YWlRtaW46a29zdHJvbG10YmlzQWMA==\r\n
    \r\n

```

Rys.13.5. Komunikat RTSP Play

Pojedyncze klatki z kamery IP pobierane są z użyciem komunikatów *RTP*. Rysunek 13.6. pokazuje sekwencję komunikatów *RTP*. Można również zaobserwować użycie protokołu *RTCP*, który wspiera protokół *RTP*. Monitoruje dane oraz zapewnia funkcje sterujące i identyfikacyjne.

No.	Time	Source	Destination	Protocol	Length	Info
964	2017-05-28 14:37:16.279374	192.168.18.50	192.168.18.102	RTSP	246	PLAY rtsp://192.168.18.102:554/live/ch1/ RTSP/1.0
965	2017-05-28 14:37:16.367771	192.168.18.102	192.168.18.50	RTSP	447	Reply: RTSP/1.0 200 OK
969	2017-05-28 14:37:18.166556	192.168.18.102	192.168.18.50	RTP	1371	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=360, Time=11442330
970	2017-05-28 14:37:18.166816	192.168.18.102	192.168.18.50	RTCP	142	Sender Report Source description Application specific ( qtsi ) subtype=1
972	2017-05-28 14:37:18.174167	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=361, Time=11442330
973	2017-05-28 14:37:18.174538	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=362, Time=11442330
975	2017-05-28 14:37:18.174727	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=363, Time=11442330
976	2017-05-28 14:37:18.194944	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=364, Time=11442330
978	2017-05-28 14:37:18.199171	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=365, Time=11442330
979	2017-05-28 14:37:18.199276	192.168.18.102	192.168.18.50	RTP	1372	PT=DynamicRTP-Type-96, SSRC=0x4E153BA4, Seq=366, Time=11442330

Frame 969: 1371 bytes on wire (10968 bits), 1371 bytes captured (10968 bits) on interface 0  
 Ethernet II, Src: Shenzhen\_79:1d:51 (28:f3:66:79:1d:51), Dst: HonHaiPr\_d8:8b:69 (10:08:b1:d8:8b:69)  
 Internet Protocol Version 4, Src: 192.168.18.102, Dst: 192.168.18.50  
 Transmission Control Protocol, Src Port: 554, Dst Port: 3595, Seq: 2674, Ack: 1290, Len: 1317  
 RTSP Interleaved Frame, Channel: 0x00, 1313 bytes  
 Real-Time Transport Protocol

10.. .... = Version: RFC 1889 Version (2)  
 ..0. .... = Padding: False  
 ...0 .... = Extension: False  
 .... 0000 = Contributing source identifiers count: 0  
 0... .... = Marker: False  
 Payload type: DynamicRTP-Type-96 (96)  
 Sequence number: 360  
 Timestamp: 11442330  
 Synchronization Source identifier: 0x4e153ba4 (1310014372)  
 Payload: 7c858883004ffe3ee969bf93077e298b320ac96994738d4a...

*Rys.13.6. Komunikat RTP zawierający pojedynczą klatkę z kamery IP*