Dokumentacja końcowa

1. Treść zadania

System komunikacji pomiędzy bramą radiową a zbiorem kamer wykonujących zdjęcia.

Kamery wysyłają do bramy zdjęcia zgodnie z ustawionymi parametrami (rozdzielczość, ogniskowa, okres i zmienność opóźnienia «*jitter*»). Zaproponować procedury instalacyjne, testowe i konfiguracyjne. W warstwie transportowej używamy wyłącznie UDP. Niezawodność obsługiwana jest w warstwie aplikacji. Należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów.

2. Nazwa projektu: Camera-Link

3. Założenia funkcjonalne i niefunkcjonalne.

Założenia funkcjonalne:

- Administrator może dokonać instalacji, testowania lub konfiguracji kamer.
- Procedura instalacyjna pozwala na sparowanie kamer z bramą radiową.
- Procedura konfiguracyjna pozwala na przesłanie parametrów działania do kamer (rozdzielczość, ogniskowa, okres i zmienność opóźnienia «*jitter*»).
- Procedura testowa pozwala na sprawdzenie poprawności parowania, wykonanie i przesłanie testowego zdjęcia na żądanie.
- Kamery cyklicznie przesyłają zdjęcia do serwera.
- Możliwość podglądu komunikatów za pomocą Wiresharka.

Założenia niefunkcjonalne:

- Do komunikacji wykorzystuje się protokół UDP.
- System działa zarówno dla adresów IPv4 jak i adresów IPv6.
- Aplikacja sprawdza kompletność danych.
- Procedura instalacyjna musi uniemożliwiać lub utrudniać podłączenie fałszywych źródeł informacji podczas parowania urządzeń (uwierzytelnianie serwera oraz kamer).
- Komunikacja między kamerami i serwerem jest szyfrowana.
- Aplikacja sprawdza poprawność parametrów komunikatów odbieranych z sieci.
- Na serwerze i kliencie znajduje się klucz publiczny serwera.

4. Podstawowe przypadki użycia.

a. Instalacja systemu:
   - Administrator wprowadza adres IP kamery.
   - Bramka wysyła komunikat rozpoczynający połączenie do kamery wraz z 4 znakowym hasłem kamery.
   - Kamera po odebraniu komunikatu i sprawdzeniu poprawności hasła, odsyła komunikat o stanie instalacji. W przypadku pozytywnej instalacji zapamiętuje adres bramki.
   - Bramka, w przypadku pozytywnej odpowiedzi, dodaje kamerę do swojej listy.

b. Testowanie systemu:
   - Administrator wprowadza adres IP kamery.
   - Bramka wysyła wiadomość testową do kamery i czeka na odpowiedź.
   - Kamera odsyła wynik autotestu.

c. Konfiguracja kamery:
   - Administrator wprowadza adres IP kamery oraz parametry konfiguracyjne.
   - Bramka przesyła parametry do kamery i czeka na potwierdzenie.
   - Kamera zapisuje parametry w pamięci i wysyła potwierdzenie konfiguracji wraz z wynikiem autotestu.

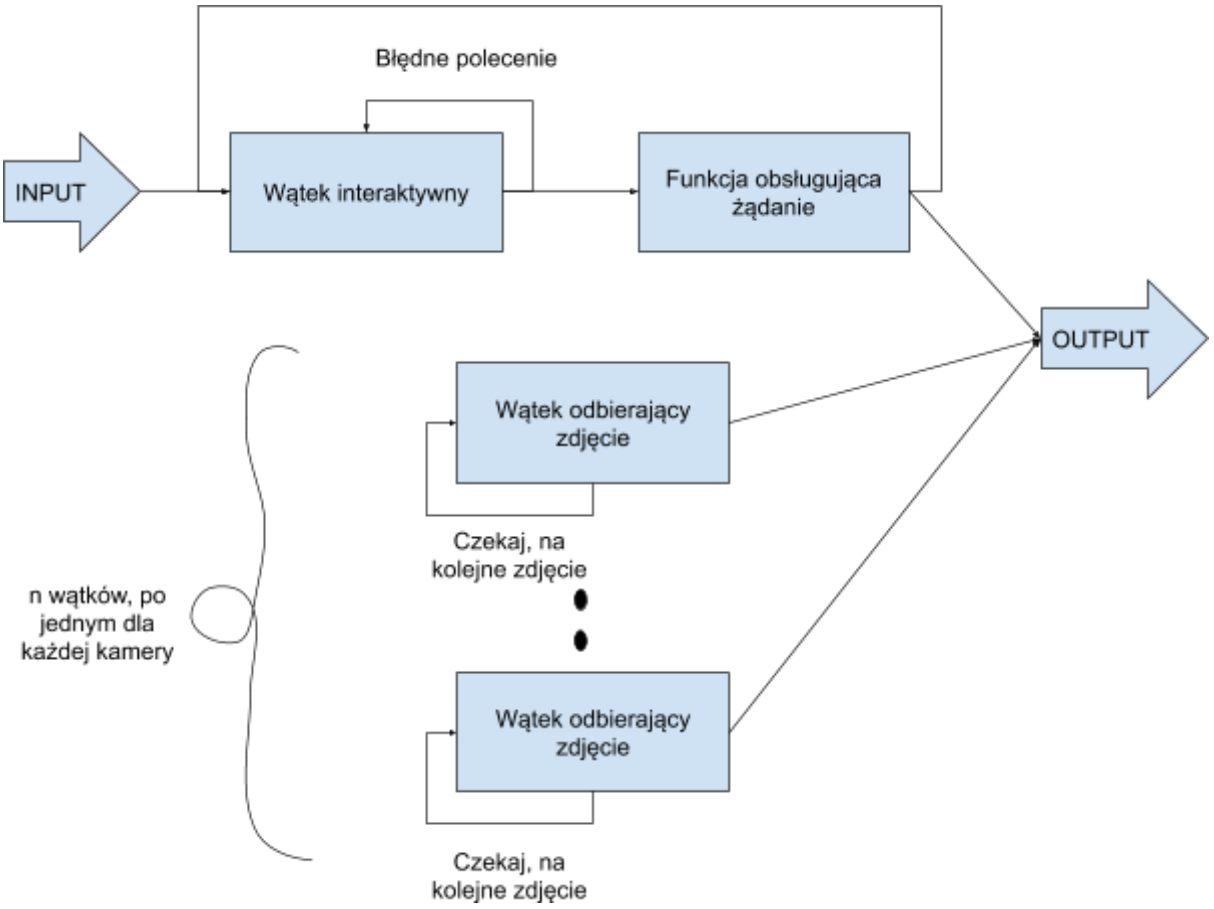
d. Przesyłanie pliku:
   - Kamera fragmentuje przesyłany plik i rozpoczyna ich przesyłanie.
   - Bramka odbiera kolejne fragmenty, zapamiętując które z nich dostała.
   - Jeśli po upływie określonego czasu bramka dalej nie posiada wszystkich pakietów, wysyła żądanie retransmisji brakujących pakietów.
   - Jeśli kamera dostała żądanie retransmisji, to przesyła ponownie wybrane pakiety.
   - Bramka po otrzymaniu wszystkich pakietów wysyła do kamery potwierdzenie odebrania pliku.

5. Wybrane środowisko sprzętowo-programowe (systemy operacyjne, biblioteki programistyczne) i narzędziowe (debugowanie, testowanie).

- System operacyjny: ubuntu 16.04 wzwyż
- korzystanie z biblioteki do obsługi wątków : `pthread.h`
- Inne wykorzystywane biblioteki: `ctime`, `algorithm`, `fstream`, `stdexcept`, `string`, `string.h`, `stdio.h`, `stdlib.h`, `unistd.h`, `arpa/inet.h`, `sys/socket.h`, `sys/types.h`, `netinet/in.h`, `netdb.h`
- korzystanie z bibliotek do testów jednostkowych Boost C++

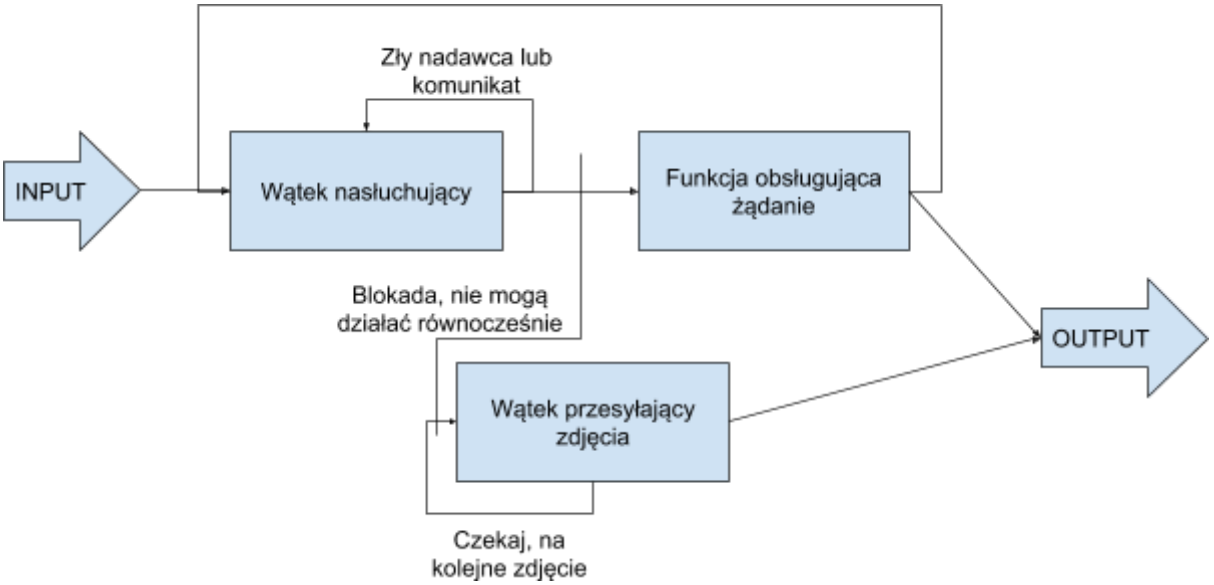
6. Architektura rozwiązania.

Bramka i kamery komunikują się używając portów z zakresu 4000-5000.
Struktura działania oprogramowania bramki:



Na serwerze pracuje jeden wątek/proces główny, który po odebraniu żądania od administratora zajmuje się jego obsługą. Jeśli w wyniku żądania połączono się z nową kamerą zostaje utworzony wątek odbierający zdjęcia z tej kamery. Analogicznie jeśli doszło do rozłączenia kamery, odpowiedni wątek jest zabijany. W momencie rozpoczęcia obsługi zdarzenia nakładana jest blokada na odpowiedni wątek, aby nie doszło do błędnego odczytu danych.

Struktura działania oprogramowania kamery:



Istnieje wątek nasłuchujący, który zajmuje się przyjmowaniem komunikatów z zewnątrz. Odpowiada on za odrzucanie komunikatów z niewłaściwego źródła lub o niewłaściwej treści. Jeśli komunikat jest poprawny, to wątek ten zajmuje się obsługą danego zdarzenia. Równolegle istnieje drugi wątek, odpowiedzialny za przesyłanie zdjęć do bramki. Działa on tylko wtedy gdy spełniony jest szereg warunków, tzn. gdy bramka jest podłączona, kamera jest poprawnie skonfigurowana oraz nie istnieje żadne obsługiwane żądanie. Wtedy wątek ten fragmentuje i przesyła zdjęcie, a następnie zwalnia blokadę i zostaje uśpiony na wcześniej określony czas. Dodatkowo ta struktura jest powielona dla adresów IPv6.

7. Lista komunikatów z określeniem nadawców i odbiorców.

Wszystkie komunikatu zaczynają się od jednego bajtu zawierającego ich kod, dlatego nie będzie on dalej wymieniany. Typy całkowitoliczbowe i zmiennoprzecinkowe są kodowane w konwencji *little endian*.

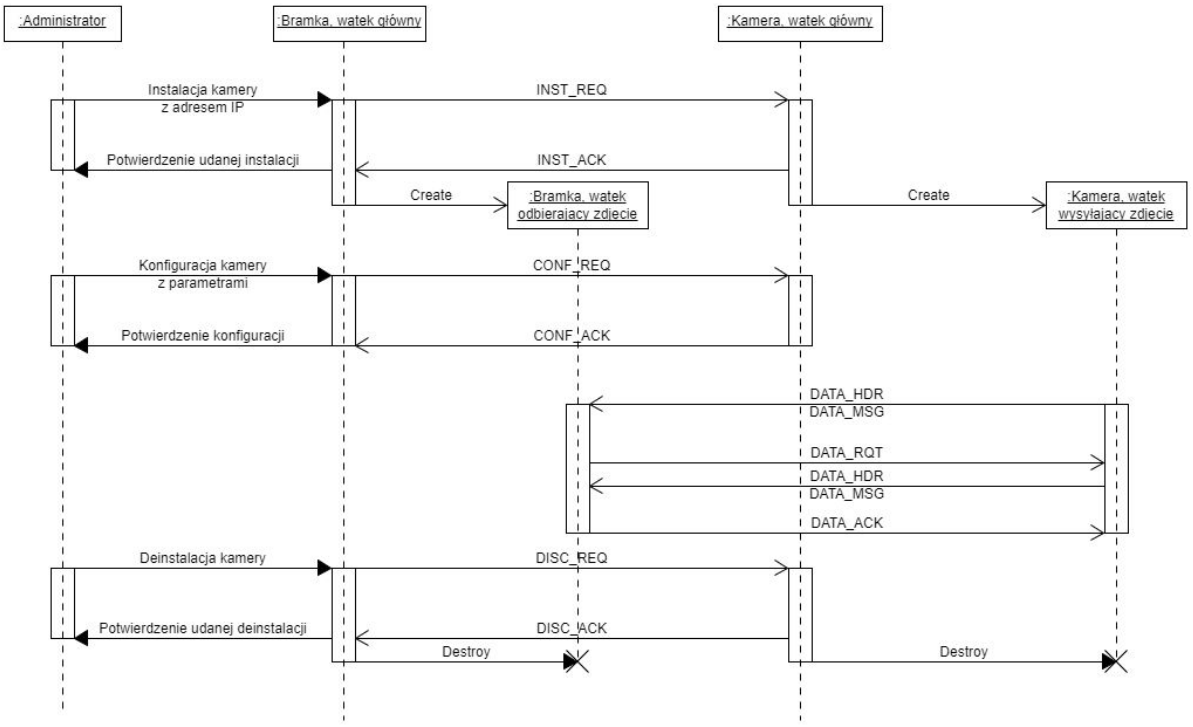
Kod	Nazwa	Nadawca	Odbiorca	Struktura	Opis
1	TEST_REQ	Bramka	Kamera	-	Inicjalizuje test połączenia z kamerą
2	TEST_ACK	Kamera	Bramka	1B(z ustawionymi flagami dla poszczególnych parametrów)	Potwierdza połączenie i przesyła wynik autotestu
3	CONF_REQ	Bramka	Kamera	4B(int)(rozdzielczo	Inicjalizuje

				ść pozioma) +4B(int)(rozdzielczość pionowa) +4B(float)(ogniskowa) +4B(float)(okres) +4B(float)(zmienność opóźnienia)	konfigurację parametrów kamery
4	CONF_ACK	Kamera	Bramka	1B(z ustawionymi flagami dla poszczególnych parametrów)	Potwierdza konfigurację i zwraca wynik autotestu
5	INST_REQ	Bramka	Kamera	4B(zawierające hasło)	Inicjalizuje podłączenie kamery do bramki, kamera sprawdza zgodność hasła
6	INST_ACK	Kamera	Bramka	-	Potwierdzenie udanego podłączania
7	DISC_REQ	Bramka	Kamera	-	Inicjalizacja odłączenia kamery
8	DISC_ACK	Kamera	Bramka	-	Potwierdzenie odłączenia
9	DATA_MSG	Kamera	Bramka	4B(int)(numer pakietu) +4B(int)(liczba wszystkich pakietów) +512B(dane)	Pakiet zawierający dane przesyłanego pliku
10	DATA_RQT	Bramka	Kamera	4B(int)(numer pakietu do ponownego przesłania)	Komunikat służący do żądania retransmisji danego pakietu(numer pakietu = 0, oznacza żądanie retransmisji nagłówka)
11	DATA_ACK	Bramka	Kamera	-	Potwierdzenie odebrania całego pliku

12	DATA_HDR	Kamera	Bramka	4B(int)(liczba znaków nazwy pliku) +4B(int)(liczba wszystkich pakietów) +nB(nazwa pliku)	Komunikat zawierający nazwę przesyłanego pliku
13	NO_PAIR	Kamera	Bramka	-	Przesyłany w odpowiedzi na DISC_REQ, jeśli kamera nie jest połączona z żadną bramką, lub gdy kamera dostała niepoprawne hasło w INST_REQ
14	IS_PAIR	Kamera	Bramka	-	Przesyłany w odpowiedzi na INST_REQ, jeśli kamera jest już połączona z jakąś bramką

8. Przykładowe scenariusze komunikacji.

Scenariusz komunikacji prezentujący instalację, konfigurację, przesyłanie danych i dezinstalację.



9. Sposób testowania.

Przy każdym punkcie kontrolnym projektu będą dokonywane testy integracyjne pomiędzy poszczególnymi modułami projektu tworzonymi przez członków zespołu. W trakcie testów przesyłane komunikaty będą monitorowane za pomocą modułu Wiresharka w celu potwierdzenia ich poprawności.

Testy jednostkowe będą polegały na sprawdzeniu działania systemu dla przypadków typowych, skrajnych oraz złośliwych (takich jak np. celowo niepoprawnie wprowadzane dane, mające na celu zakłócenie działania systemu).

Testy które system przeszedł we wcześniejszym etapie projektu będą powtarzane przy dodawaniu nowych elementów, aby upewnić się, że nie powodują one nowych błędów.

10. Sposób demonstracji rezultatów, tj. scenariusze testów akceptacyjnych do zaprezentowania przy odbiorze projektu.

- Test zaniku napięcia dla serwera

- Test zaniku napięcia dla klienta
- Test obciążeniowy komunikacji między wieloma klientami a serwerem

11. Wnioski z testowania.

Klasy Camera, Message i FileMessage przechodzą swoje testy jednostkowe. Kamera przechodzi test zaniku prądu. Nie udało nam się przeprowadzić pozostałych dwóch zaplanowanych testów, z powodu wybrakowanej funkcjonalności serwera. Ręcznie przetestowaliśmy przesyłanie zdjęcia, które działa.

12. Analiza podatności bezpieczeństwa.

W naszym systemie do ataków może dochodzić głównie na kamery. Pierwszym słabym punktem jest parowanie kamery z bramką, które jest zabezpieczone 4 znakowym hasłem(ale bez większych problemów można je wydłużyć). Kolejnym problemem jest to, że hasło to jest przechowywane jako plik tekstowy w pamięci masowej kamery, co oznacza, że w przypadku włamania się do kamery może ono zostać wykradzione. Hasło to, jak również wszelkie inne dane, można podsłuchać w sieci, ponieważ nie jest w żaden sposób szyfrowane. Po instalacji kamery, identyfikacja kolejnych komunikatów następuje tylko poprzez sprawdzenie adresu IP nadawcy, co stwarza kolejną lukę bezpieczeństwa.

Podsumowywując, nasz system nie jest dobrze zabezpieczony przed celowymi atakami profesjonalistów, ale jest odporny na przypadkowe ataki. Nie jesteśmy zadowoleni z tego rozwiązania, ale niestety zabrakło nam czasu na wykonanie pełnoprawnej identyfikacji na podstawie certyfikatów z szyfrowaniem.

13. Instrukcja instalacji i dezinstalacji stworzonego systemu.

W celu instalacji oprogramowania należy wywołać polecenie *make*, które wyprodukuje dwa pliki wykonywalne: *gate* i *camera*. Pierwszy z nich należy umieścić na bramce, drugi na kamerze wraz z plikiem tekstowym *password.txt* zawierający 4 znakowe hasło dostępu do kamery.
W celu przeprowadzenia testów jednostkowych należy wywołać polecenie *make test*.
W celu usunięcia plików należy użyć polecenia *make clean*.

14. Podział prac w zespole.

- Moduł do Wiresharka : Adam Bieniek
- Testy akceptacyjne: Adam Bieniek
- Bramka: Konrad Meysztowicz-Wiśniewski
- Kamera: Maciej Puchalski
- Procedury: Łukasz Rombel

- Dokumentacja końcowa: Adam Bieniek

15. Harmonogram prac.

a. 17 kwietnia - wstępna wersja serwera i klienta umożliwiająca podstawowa komunikacje i testowanie przy pomocy Wiresharka
b. 7 maja - implementacja procedury instalacyjnej i konfiguracyjnej
c. 25 maja - złożenie wszystkich elementów systemu, końcowe testy integracyjne

16. Podsumowanie.

Niestety czytając założenia naszego systemu można łatwo zauważyć, że nie udało nam się zrealizować ich wszystkich. Przyczyną tego stanu rzeczy jest źle ułożony harmonogram prac. Udało nam się zaimplementować obsługę IPv6 tylko na kamerach, więc nie byliśmy w stanie tego przetestować. Nie udało nam się zaimplementować wielowątkowości na bramce, przez co obecna wersja bramki jest tylko wersją demonstracyjną. Najbardziej brakującą funkcjonalnością jest brak obsługi wielu kamer naraz.

Projekt był bardzo edukacyjny. Dowiedzieliśmy się o wielu trudnościach jakie mogą napotkać twórcy oprogramowania sieciowego, m. in. różnice formatów(big endian, little endian) czy komunikaty o niestałej długości. Doceniamy również programy do nasłuchu sieci (tutaj Wireshark), które są bardzo pomocne w wykrywaniu błędów.

Jak już wspomniałem wyżej, nasza współpraca nie była idealna, więc na pewno wyciągniemy z tego wnioski na przyszłość. Systematyczna komunikacja i kontrola postępu są kluczem do stworzenia udanego projektu na czas.

17. Adres projektu na serwerze kontroli wersji.

https://github.com/koniczyn5/TIN_cameras