Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Kanał optyczny transmisji danych dla urządzeń przenośnych maksymalizacja przepustowości

Arkadiusz Lewandowski
Nr indeksu: 208836

Praca inżynierska napisana
pod kierunkiem
Dr. inż. Łukasza Krzywieckiego

Politechnika
Wrocławska
Wrocław 2016

Spis treści

Wstęp

Praca swoim zakresem obejmuje zagadnienia bezprzewodowej wymiany danych pomiędzy urządzeniami przenośnymi bez wykorzystania infrastruktury sieciowej. Poprzez infrastrukturę sieciową rozumie się tutaj infrastrukturę sieci bezprzewodowych realizowanych za pomocą standardu IEEE 802.11*, w ramach których urządzenia wymieniające się danymi muszą być podłączone do tej samej sieci. Brak wykorzystania infrastruktury zakłada, że urządzenia łączą się ze sobą w ramach sieci „ad-hoc"w kanałach elektromagnetycznych. (Bluetooth, IrDA, NFC) lub kanałach alternatywnych- dźwiękowych za pomocą modulacji sygnału bądź kanału optycznego. Z racji tego, że współczesne smartfony posiadają kamery i urządzenia rejestrujące, w pracy tej rozważa się kanał optyczny. Celem pracy będzie zbadanie efektywności przesyłania danych za pomocą kanału optycznego, ergonomia nawiązywania połączenia oraz przepustowość transmisji (teoretyczna przepustowość wynikająca z teoretycznych ograniczeń oraz z ograniczeń wynikających z programistycznych implementacji).
Szczegółowymi celami pracy są

- ocenienie maksymalnej przepustowości
- projekt systemu do przesyłania danych w kanale optycznym z wykorzystaniem technologii QR kodów jako nośnika zakodowywanych bitów po stronie nadawczej i odbiorczej
- implementacja projektu na platformie iOS z wykorzystaniem bibliotek standardowych lub łatwo dostępnych na wszystkich platformach marki Apple.
- przeprowadzenie testów weryfikujących przepustowość uzyskaną z przepływności, która została wyliczona

Istnieje szereg aplikacji wykorzystujących kody QR do potwierdzania tożsamości obiektów, przy czym ich przeznaczenie zakłada statyczność kodów oraz brak interakcji z urządzeniem czytającym. Natomiast aplikacji wykorzystujących kody te do wymiany danych lub do szybkiego ich czytania nie ma wielu. Warto zatem przyjrzeć się scenariuszom, w których mogłyby ułatwiać codzienne czynności.

Obiekt indentyfikujący się takim kodem może być *statyczny*, w rozumieniu, że nie wymaga on mocy obliczeniowej. Raz wygenerowana sekwencja QR kodów może przedstawiać zbiór danych, co można wykorzystać jako tworzenie fizycznych kopii zapasowych dokumentów, które mogłyby w prosty sposób być czytane za pomocą urządzenia elektronicznego. Takim rozwiązaniem można zapobiec utracie cyfrowych danych przez czynniki zewnętrzne oraz uszkodzenia złośliwym oprogramowaniem, drukując wygenerowane dane i tym samym oddzielić informacje od sieci, wciąż mając do niej łatwy dostęp. Urzędy przechowujące dane o obywatelach mogą wydrukować wszystkie potrzebne informacje o nich i przechowywać bezpiecznie, zachowując przy tym łatwość ich ponownego wczytania. Zapewniając sobie tym samym bezpieczeństwo

danych i brak ingerencji ze strony cyfrowej. Przechowywanie i wczytywanie faktur mogłoby stać się o wiele szybsze i bezpieczniejsze. Użytkownik nie musiałby jej pobierać, a jedynie nakierować urządzenie czytające na monitor lub fizyczny wydruk i miałby dostęp do swoich danych. W tym przypadku naszą uwagę może zwrócić zagrożenie często omawiane przy temacie kodów QR i ich czytania. Problemem może się okazać złośliwy kod wstrzyknięty w naszą sekwencję QR, jednakże nie uruchamiając czytanych symboli, a jedynie wyświetlając je odkodowane, blokujemy jedną z dwóch dróg ataku(zobacz []). Druga natomiast, często nazywana przepełnieniem bufora, jest wyeliminowana przez możliwość techniczną urządzenia. Dla kodu QR pojedynczy symbol nie przekracza 3000 bajtów- mieści się on zatem w zwykłej zmiennej typu *String* (zobacz [1]).

Brak możliwości wymiany danych za pomocą kabli lub komunikacji radiowej. W takim przypadku użytkownik mógłby wygenerować na swoim urządzeniu ciąg danych, które w sposób jednoznaczny i dekodowalny przedstawiałyby plik lub ciąg znaków, do odczytania przez inne urządzenie. Użytkownik minimalizuje ryzyko wycieku danych w sieci oraz uniezależnia się od platformy, którą miałby te dane przesłać. Weźmy przykład, w którym użytkownik nie ma dostępu do sieci oraz jego urządzenia nie są w stanie połączyć się za pomocą takich technologii jak bluetooth, wifi lub nawet USB oraz platformy nie przewidują komunikacji z innymi urządzeniami spoza swojej marki. W takim przypadku komunikacja za pomocą standardowych QR kodów może zostać alternatywnym kanałem transmisji danych. Sytuacja ta pokazuje także, że użytkownicy tej techniki są niezależni od infrastruktury takich cyfrowych technologii jak telefoniczna sieć komórkowa.

Praca ta ma za zadanie pokazać, że efektywna transmisja za pomocą kodów QR jest możliwa oraz nie wymaga dodatkowej infrastruktury, co umożliwia wykorzystanie w warunkach polowych bez uprzedniego przygotowania.
Projekt jako całośc zakłada kolejno:

- W rozdziale pierwszym zostaje omówiony problem i jego teoretyczne rozwiązanie.
- W rozdziale drugim zostają omówione założenia teoretyczne samej transmijsi danych, które nie uwzględniają technicznych możliwości urządzenia wraz z jego ograniczeniami programistycznymi.
- W trzecim rozdziale przedstawiony jest projekt prototypu aplikacji, która spełnia wymagania teoretycznego rozwiązania problemu.
- W czwartym rozdziale jest budowa aplikacji od podstaw wraz z oceną możliwości programistycznych, takich jak obsługa wątkowości dla pojedynczego urządzenia wejścia - kamery, a także ograniczenia wynikające ze wspólnego czytania/nadpisywania pamięci. Zostają także omówione biblioteki i wzorce projektowe użyte/odrzucone w aplikacji. Czytelnik zostaje wprowadzony w świat optymalizacji aplikacji mobilnej pod względem zmniejszenia wykorzystania pamięci podręcznej.
- W piątym zostają przeprowadzone badania na różnych urzadzeniach mobilnych, platformy iOS. Uwzględniając szybkości procesorów i możliwości interakcji z kartą graficzną przez bibliotekę AVFoundation. Zostają także zaprezentowane i omówione jednokierunkowe transfery: osobno z punktu widzenia nadającego i odbierającego urządzenia, a

także perspektywa komunikacji między urządzeniami za pomocą kamer niższej rozdzielczości. W tym rozdziale pojawiają się też wykresy osiągniętych zakresów dla poszczególnych urządzeń i ich specyfikacji, takich jak rozdzielczość, częstotliwość odświeżania ekranu oraz nagrywania.

- W szóstym - ostatnim - podsumowywane są wszystkie aspekty pracy w sposób uogólniony i następuje próba oceny możliwości urządzeń w niedalekiej przyszłości oraz przypadki użycia alternatywnego kanału w komunikacji bez interferencji.

1 Analiza problemu

W tym rozdziale przedstawiana jest analiza problemu transmisji danych QR kodami oraz rozważania nad jego podproblemami. (zobacz [1]). Najpierw zostaje omówiony nośnik danych - kod QR, następnie sposoby i rozważania nad jego użyciem. Przeanalizowanie wymogów jakie musiałaby spełniać aplikacja umożliwiająca zbadanie parametrów maksymalizujących przepustowość kanału.

Problem

W celu maksymalizacji przepustowości najpierw trzeba poznać wszystkie czynniki mające na nią wpływ. W przypadku transmisji takim kanałem alternatywnym jak kanał optyczny, szczególnymi czynnikami są przede wszystkim możliwości ustalonego nośnika, algorytmów używanych w jego eksploatacji oraz parametrów technicznych urządzeń, na których jest on wykorzystywany. W tym badaniu wybranym nośnikiem jest kod QR. Pozostałe czynniki zostają omawiane w następnych rozdziałach pracy.

Kod QR

W porównaniu z konwencjonalnym kodem kreskowym, kod QR jest w stanie zawrzeć w sobie od kilkudziesięciu do kilkuset razy więcej informacji. Przez swoją budowę i sposób kodowania - można w nim przechować dane w dowolnej postaci cyfrowej. Główną zaletą jest to, że pojedynczy kod QR jest w stanie reprezentować sobą 7,089 cyfr albo 4,296 alfranumerycznych symboli, albo 2,953 bajtów lub 1,817 znaków Kanji/ pełnej szerokości Kanę. Kod QR posiada zdolność korekcji błędów. Zatem dane zatracone w wyniku częściowego przysłonięcia lub zniszczenia - mogą zostać odzyskane. Maksymalnie każdy pojedynczy QR jest w stanie zakodować 30% danych korekcyjnych ze wszystkich bajtów sobą reprezentowanych w postaci słów kodowych.

W pojedynczym kodzie modułami, z których się on składa, nazywamy kwadraty przybierające jeden z dwóch kolorów- biały lub czarny. Zbiory modułów na kodzie tworzą słowa kodowe, które przedstawiają informacje jako poszczególne znaki. Wygląd pojedynczego kodu jest zależny od jego wersji. Odróżniamy je na podstawie poziomu korekcji błędów oraz ilości danych w nich zapisanych.

W odczytywaniu kodów QR wykorzystuje się wzór wyszukiwania umożliwiający czytnikowi odnalezienie poszczególnych miejsc w QR kodzie, względem których odczytywane są pozostałe jego części. W pierwszej kolejności wyszukiwane są duże moduły [4.1], które muszą być

oddzielone od krawędzi całego kodu białą ramką szerokości co najmniej jednego modułu. Kolejnym szukanym wzorcem jest [4.3] wzór synchronizacji, który pozwala na określenie wersji, gęstości oraz współrzędnych poszczególnych danych zapisanych w kodzie. Kolejnym szukanym elementem jest lub są tzw. wyrównania [4.2]. Może ich być wiele w zależności od ilości danych zapisanych w QR. Na podstawie [2.] można ustalić wielkość kodu, a ilość danych w obszarze [1.] wskazuje na wersję QR kodu, gdyż jest ona zależna od maskowania i ilości danych. [3.] prezentują dane i korekcję błędów dla tych danych.

Kody QR są w pewnym sensie odporne na zniszczenia i przesłonięcia. Zawdzięczają to korekcji błędów, która to duplikuje dane w nich zawarte. Dotyczy to jednak szczególnych miejsc. Słowo kodowe w przypadku kodu QR ma długość 8 bitów, więc jeśli zniszczenie dotknie wszystkich powtórzeń tego słowa- kod nie zostaje odczytany.

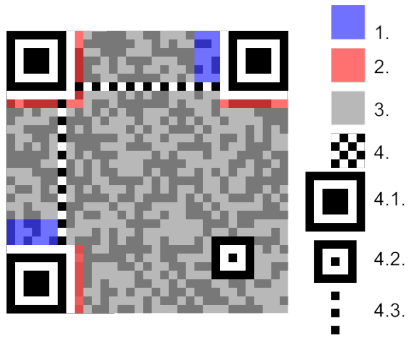
QR są czytelne pod każdym kątem obrotu w osi prostopadłej do płaszczyzny na której się znajdują.

Kolejną zaletą jest możliwość podziału zbyt dużego symbolu na mniejsze, przy jednoczesnym zachowaniu danych. Ta własność w tej pracy jest wielokrotnie wykorzystywana.

Cechą szczególną kodów QR jest mechanizm maskowania, który powoduje, że średnia kolorów na dowolnych obszarach złożonych z modułów jest sobie jak najbliższa. Przez takie zrównoważenie występowania białych i czarnych modułów wzrasta szybkość odszukiwania i dekodowania obrazów przez skanery.

W czerwcu 2000 roku organizacja ISO włączyła kod QR jako standard ISO/IEC18004.

Schemat



1. Informacja o wersji
2. Informacja o formatowaniu
3. Dane i korekcja błedów
4. Wzorce:
    4.1. Pozycje
    4.2. Wyrównanie
    4.3. Chronometraż

Sposoby użycia

Kod QR można wykorzystać jako nośnik danych na kilka różnych sposobów.

Przedstawienie wielu kodów QR tej samej wielkości, przy czym każdy z nich będzie zawierał jak najwięcej danych. Implikuje to sporej wielkości kody.

Generowanie pomniejszych kodów QR tej samej wielkości, które pozwalają na jednoczesną prezentację ich większej ilości na jednym ekranie.

Generowanie przemiennych kodów QR, losowej wielkości. Pozwoli to stworzyć dla urządzenia czytającego tzw kontrast i zmusi je do ponownego wyszukiwania położenia QR na ekranie.

Wymogi prototypu

Do rozwiązania problemu wymagana jest aplikacja, która spełniałaby następujące wymogi:

- Urządzenie nadające tłumaczy plik na ciągi znaków, by następnie wyświetlić je jako kody QR.
- Urządzenie odbierające za pomocą wbudowanej kamery odbiera przesyłane dane, zapisując obraz z wyświetlacza urządzenia nadającego i następnie z pojedynczych klatek transmisji odczytuje kody QR.

Aplikacja w trybie nadawania musi pozwalać na modyfikacje takich cech jak:

- czas między kolejnymi sekwencjami QR
- wielkość samych QR kodów
- ilość QR kodów na ekranie i wątków

Aplikacja w trybie odbierania musi zmaksymalizować względem czytania danych następujące:

- nagrywanie z maksymalną dostępną szybkością
- nagrywanie z maksymalną dostępną rozdzielczością

1.1 Założenia teoretyczne

Ten rozdział został przeznaczony na rozważania nad problemem samej transmisji, nie uwzględnia on przeszkód wynikających ze specyfikacji samej platformy, na której realizowany jest projekt.

Problemem przeprowadzanego badania jest szukanie optymalnych parametrów dla urządzeń w komunikacji, którego znalezienie wymaga wstępnych założeń i zbadania ich poprawności empirycznie. Urządzenia mobilne korzystające natywnie ze swoich komponentów, takich jak karta graficzna lub procesor, posiadają ograniczenia specyfikacji. Rozdzielczość wraz z częstotliwością odświeżania wyświetlacza nie są jedynymi składowymi problemu. Urządzenie nie umożliwia pełnego wykorzystania jego komponentów pojedynczej aplikacji, ale również zależnie od ilości zainstalowanych aplikacji korzystających z procesora w tle - wydajność testowanego oprogramowania spada. Zatem potrzebne są techniki zmian kolejkowania zadań platformy, by zminimalizować użycie podzespołów dla procesów rozpraszających pracę testowanej aplikacji. Kolejnym niezbędnym do rozstrzygnięcia problemem jest ten leżący między urządzeniami komunikującymi się - widoczność sekwencji kodów. Jako, że aplikacja nadająca może przyjąć, że jest w stanie nadawać z maksymalną przepustowością na największej dostępnej jej rozdzielczości, to z kolei odbierająca nie gwarantuje, że w jej kadrze znajdą się wszystkie kraty z kodami, ale również odwrotna sytuacja może miec miejsce. Kamera urządzenia powinna nagrywać z maksymalną dostępną jej rozdzielczością i maksymalną ilością klatek na sekundę. Wynika to z tego, że urządzenie odbierające nie jest w stanie przewidzieć z jaką częstotliwością będzie nadawało urządzenie nadające. Jeśli byłoby to możliwe od razu, to można by założyć, że dwukrotna przewaga częstości odbierania wystarczy do tego zadania (zobacz [**?**]).

Problem nie jest trywialny, gdyż po nawiązaniu połączenia, rozumianego tutaj jako odnalezienie punktów skupiających QR kody, należy rozróżnić kiedy transfer będzie większy. Większy w rozumieniu ilości danych przesłanych w określonej jednostce czasu. Dobranie parametrów by zmaksymalizować przepustowość jest jednak trudne do osiągnięcia, gdyż przy niewiele mniejszej rozdzielczości - ilość klatek na sekundę może wzrosnąć prawie dwukrotnie. Mając na względzie powyższe utrudnienia należy przyjąć, że osiągnięte wyniki będą optymalnymi tylko na urządzeniach tutaj wykorzystanych - wciąż zakładając, że ich producent nie wprowadził w nich zmian *w trakcie* lub *po* przeprowadzeniu tych badań.

1.2 Urządzenia

Urządzenia wybrane na docelową serię badań zostały dobrane ze względu na trzy aspekty.

Pierwszym jest ich wyświetlacz retina. Jego zaletą w badaniu jest wysoka gęstość piksli, dzięki której możliwe jest zbadanie przypadku dla dużej ilości małych QR kodów oraz małej ilości dużych QR kodów.

Drugim aspektem jest zróżnicowanie procesora i jego zintegrowanej karty graficznej. Pozwala to na ustalenie czy odkodowanie kodu po jego znalezieniu wymaga dużego nakładu pracy.

Trzecim rodzaj kamery, tzn. ilość klatek na sekundę oraz jej rozdzielczość. Dobór odpowiedniego urządzenia czytającego pozwala w naturalny sposób zwiększyć ilość danych w transmisji.

Wybrane zostały urządzenia mobilne, by wskazać, że dziś każdy użytkownik ogólnodostępnej technologii jest w stanie wymieniać się danymi z innymi użytkownikami bez względu na platformę czy typ urządzenia.

Parametry urządzeń

Przedstawione tutaj dane dotyczą parametrów nagrywania i wyświetlania są dostępne w specyfikacji urządzeń marki Apple, z dnia 2 lipca 2016 roku.

model	ppx_{max}(przód)	ppx_{max}(tył)	fps_{min}	fps_{max}
iPhone 5c	720	1080	30	60
iPhone 5s	720	1080	30	120
iPhone 6	720	1080	30	240
iPhone 6s	720	4K	30	240
iPhone SE	720	4K	30	240

1.3 Odbieranie

Przepustowością odbierania danych dla wybranych urządzeń mobilnych jest ich maksymalna ilość klatek na sekundę czytanych przez kamerę (tylną), na co nałożony jest czas potrzebny na odczytanie pojedynczego symbolu QR z klatek. Przyjmując, że algorytm używany przez Apple w AVFoundation framework jest algorytmem optymalnym dla testowanej platformy. Biorąc pod uwagę, iż pojedyncze klatki obrazu są chwytane i dopiero na nich odszukiwany jest symbol QR oraz następuje jego dekodowanie, można zauważyć, że wciąż istnieje problem po stronie nadającego. Jest nim jego sposób odświeżania ekranu. Może się okazać, że kod QR w momencie chwytania klatki kamerą natrafia już na obraz w jakiejś części złożony z dwóch symboli, zamiast jednego.

1.4 Nadawanie

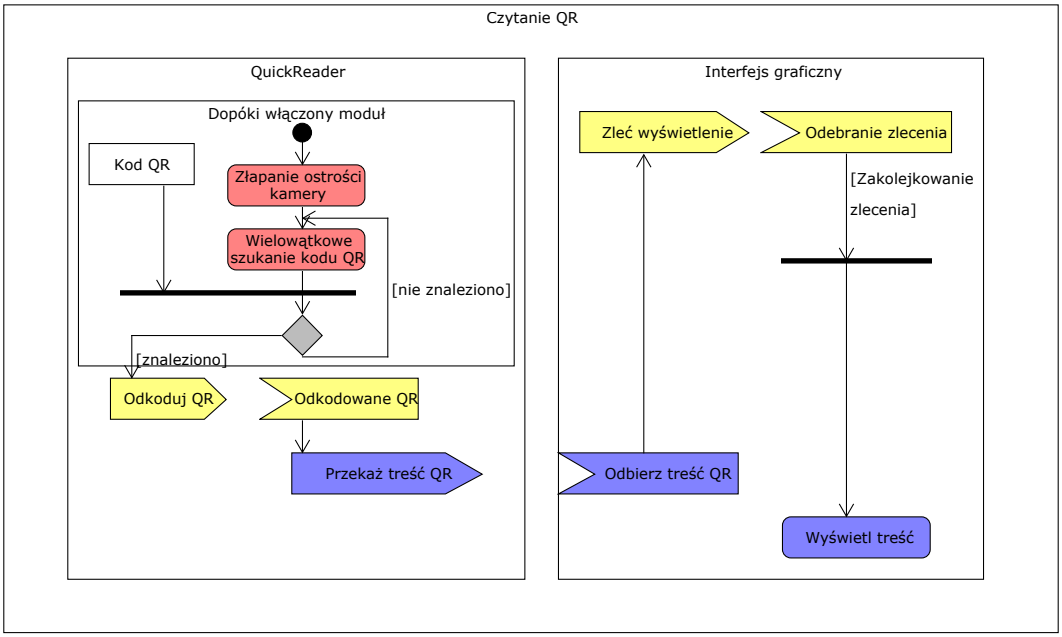
Wyświetlanie kodów QR jest procedurą wymagającą zgrania z wyświetlaniem. Platforma *iOS* nie pozwala na prezentację grafiki nie będącej w całości załadowanej do pamięci podręcznej, zatem można przyjąć pierwszy problem za rozstrzygnięty. Kolejną kwestią do rozważenia jest sposób postrzegania kodów QR przez urządzenie odbierające. Jako wysyłający nie możemy spowodować, że urządzenie odbierające przestanie zauważać kody QR lub nie będzie w stanie ich odczytać przez zbyt małą dokładkość ich prezentacji. Zatem kody QR muszą mieć rozsądną wielkość, czytelną z takiej odległości, która zapewniałaby najlepszy transfer.

2 Projekt systemu

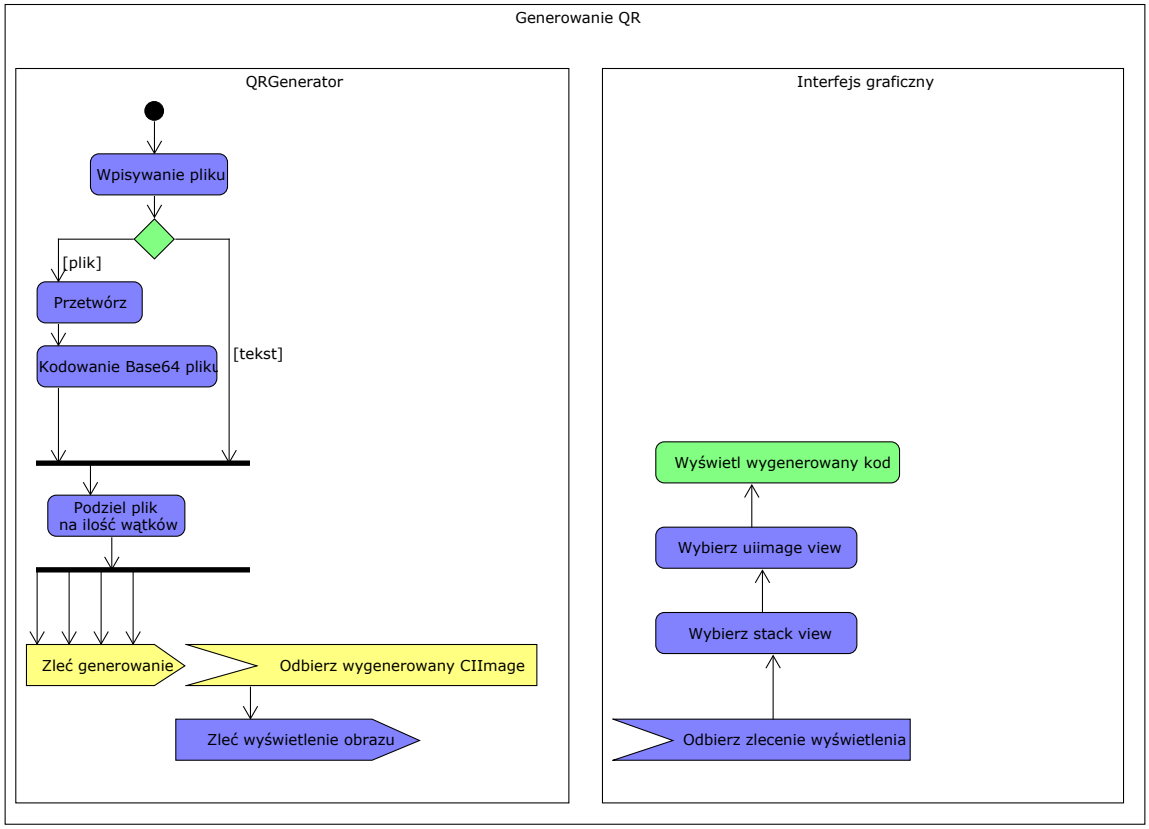
Ten rozdział przedstawia szczegółowy projekt aplikacji w notacji UML, uwzględnia on założenia funkcjonalne opisane w rozdziale 1.1. Opis relacji między składowymi systemu wyrażono diagramami.

2.1 Struktury prototypu

Diagramy aktywności



Rysunek 2.1: Caption



Rysunek 2.2: Caption

Diagramy klas

Przepraszam, ze jeszcze nie dodalem

Diagramy sekwencji

Przepraszam, ze jeszcze nie dodalem

Diagramy stanów

Przepraszam, ze jeszcze nie dodalem

3 Implementacja systemu

3.1 Opis technologii

Język programowania

Swift®, wersje kolejno 2, 2.1, 2.2, 2.3, 3.0, 3.0.1. Apple® LLVM version 8.0.0 (clang-800.0.42.1) Target: x86_64-apple-darwin15.6.0 Thread model: posix

IDE

Xcode, wersje odpowiednio wzrastające od 7.0 (7A220) do 8.1 (8B62). Tak częste zmiany wynikały z dużej ilości poprawek i większych zmian w całym środowisku. Włączając w to duże zmiany w platformach zarówno *iOS* jak i *OSX*. Pisząc pracę wymagane były dwie migracje, które pomimo wielu zmian pozwoliły na zbadanie natywnych bibliotek i ich sposobów czytania kodów QR. Ostateczną wersją, która była użyta w pracy jest - 8.1 (8B62).

3.1.1 Biblioteki

W pracy zostały użyte: - UIKit framework - konstrukcja i zarządzanie interfejsem *iOS*. Reakcje na interakcje użytkownika i wydarzenia systemowe. Dostęp do natywnych cech urzadzenia, takich jak widoki, przejścia i animacje.

- AVFoundation framework - nagrywanie, przychwytywanie sesji kamery. Framework ten zapewnia interfejs dla metod napisanych w języku Objective-C, dzięki któremu możliwe jest używanie audio-wizualnych właściwości platformy.

3.1.2 Wzorce

Swift® przez swą budowę wymaga użytkowania wzorca Model View Controller. Ważnym też w prototypie aplikacji był wzorzec Fasada, dzięki któremu dostępny jest zbiór strategii dla transmisji kodów QR.

3.2 Omówienie kodów źródłowych

Kod źródłowy 3.2 przedstawia klasę tworzącą generyczny `view controller`, który jest wykorzystywany w implementacjach poszczególnych modułów aplikacji. Interfejs ten pozwala

na uruchomienie kamery urządzenia z dowolnymi parametrami oraz umożliwia czytanie i dekodowanie kodów QR.

Kod źródłowy 3.1: Szybkie czytanie kodów QR: `QuickReader`.

```
import UIKit
import AVFoundation
internal class QuickReader : UIViewController, AVCaptureMetadataOutputObjectsDelega

    internal let session: AVCaptureSession

    internal var previewLayer: AVCaptureVideoPreviewLayer?

    internal var captureDevices: AVCaptureDevice?

    internal let queue: DispatchQueue

    internal let backgroundQueue: DispatchQueue

    internal var thread_number: Int

    internal var maxFrameRate: Double

    internal var cameraPositionOverloaded: AVCaptureDevicePosition

    @IBOutlet weak internal var QR1: UILabel!

    @IBOutlet weak internal var QR2: UILabel!

    @IBOutlet weak internal var QR3: UILabel!

    @IBOutlet weak internal var QR4: UILabel!

    internal var uilabel: [UILabel] { get }

    internal var packets: [String]

    internal func addPreviewLayer()

    internal func previewLayerHide()

    internal func previewLayerShow()

    internal func hideQRLabels()

    internal func showQRLabels()

    internal func configureDevice()
```

```
    internal func startSession()

    internal func setCamera(cameraPosition: AVCaptureDevicePosition)

    internal func setCamBefore()

    override internal func viewDidLoad()

    override internal func viewWillAppear(_ animated: Bool)

    override internal func didReceiveMemoryWarning()

    override internal func viewWillDisappear(_ animated: Bool)

    internal func somethingToDoWhileCapturingMetadata(stringFromMetaData: String, t

    internal func captureOutput(_ captureOutput: AVCaptureOutput!, didOutputMetada

    internal func sortPackets()
}
```

Kod źródłowy 3.1 przedstawia opisy poszczególnych metod interfejsu: `QRCodeGeneratorViewControlle`
Kompletne kody źródłowe znajdują się na płycie CD dołączonej do niniejszej pracy w katalogu `Kody` (patrz Dodatek A).

Kod źródłowy 3.2: Generowanie QR kodów na wątkach `QRCodeGenerator`.

```
import UIKit
import MobileCoreServices

internal class QRCodeGeneratorViewController : UIViewController, UIDocumentPickerD

    internal var qrcodeImage: CIImage!

    internal var target: String

    internal var thread_number: Int

    internal var text_spread: Int

    internal var stepDelay: Double

    internal var dataObjects: Int

    internal var swBool: Bool

    internal var files: [AnyObject]

    @IBOutlet weak internal var stackViewAll: UIStackView!
```

```
    @IBOutlet weak internal var stackViewSubUp: UIStackView!
    @IBOutlet weak internal var stackViewSubMid: UIStackView!
    @IBOutlet weak internal var stackViewSubDown: UIStackView!
    @IBOutlet weak internal var imgQRCode: UIImageView!
    @IBOutlet weak internal var imgQRCode2: UIImageView!
    @IBOutlet weak internal var imgQRCode3: UIImageView!
    @IBOutlet weak internal var imgQRCode4: UIImageView!
    @IBOutlet weak internal var imgQRCode5: UIImageView!
    @IBOutlet weak internal var imgQRCode6: UIImageView!
    @IBOutlet weak internal var textField1: UITextField!
    internal var t_imgQRCodes: [UIImageView] { get }
    internal var tField: [UITextField] { get }
    internal func documentPicker(_ controller: UIDocumentPickerViewController, didP
    internal func documentPickerWasCancelled(_ controller: UIDocumentPickerViewCo
    override internal var prefersStatusBarHidden: Bool { get }
    override internal func viewDidLoad()
    override internal func viewDidAppear(_ animated: Bool)
    internal func hideStackViews()
    internal func showStackViews()
    @IBAction internal func performButtonAction(_ sender: AnyObject)
    internal func generateTestGivenInString(str: String)
    internal func displayQRCodeImage(_ imageIter: Int)
}
```

4 Testy

Do tych badań potrzebne jest urządzenie, które w sposób stały wyświetla kod QR oraz aplikacja prototyp, umożliwiająca empiryczne zbadanie technicznych możliwości urządzeń wziętych pod uwagę w tym badaniu. Wyświetlający - urządzenie wyświetlające ten sam symbol QR, propaguje dane, nie zmieniając ich. Odbierający - urządzenie czytające symbole QR z maksymalną dla swoich parametrów prędkością. W ten sposób pozwalamy by odbierający miał możliwość przetestowania swojej prędkości odbierania bez zakłóceń zewnętrznych.

4.1 Badanie przepustowości nadawania

Nadawanie jest niezależne względem czynników zewnętrznych. Ograniczeniami urządzenia są jego własne możliwości podane w specyfikacji oraz jego procesy uruchomione w tle. Zakładając, że urządzenie nie będzie przechodziło w trakcie testów w stan inny niż aktywny i pierwszoplanowy, przyjęty zostaje brak przerw w nadawaniu.
Nadawanie danych wyświetlaczem:
Oznaczenia:

- Niech b będzie bajtem zawierającym jeden symbol ASCII po zakodowaniu danych używając BASE64.
- Niech s będzie sekundą (jednostką czasu).
- Niech i oznacza ilość bajtów w jednym QR kodzie.

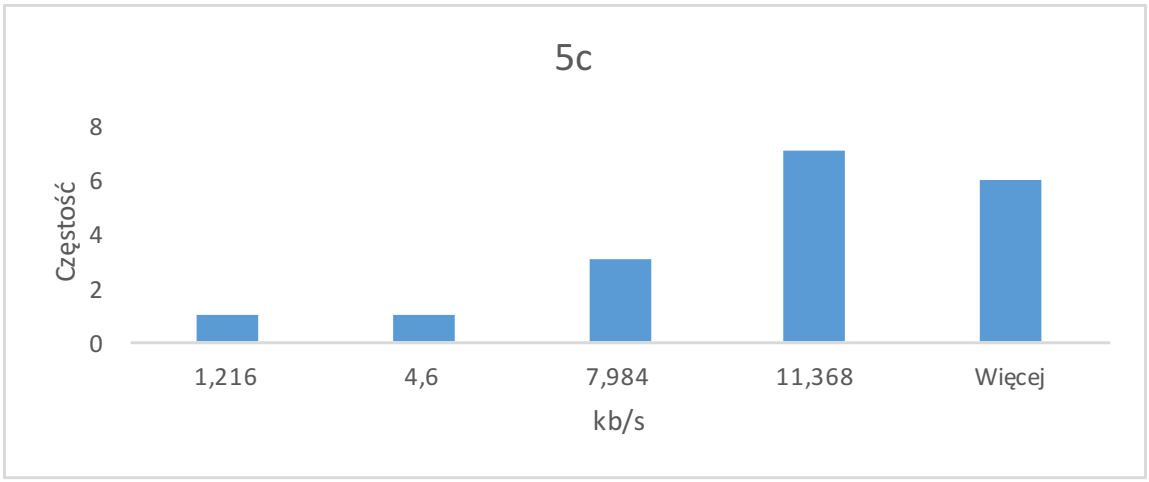
Mamy następujący wzór dla przesyłania danych.

- Zatem $V_1 = ((b * i)/s)$

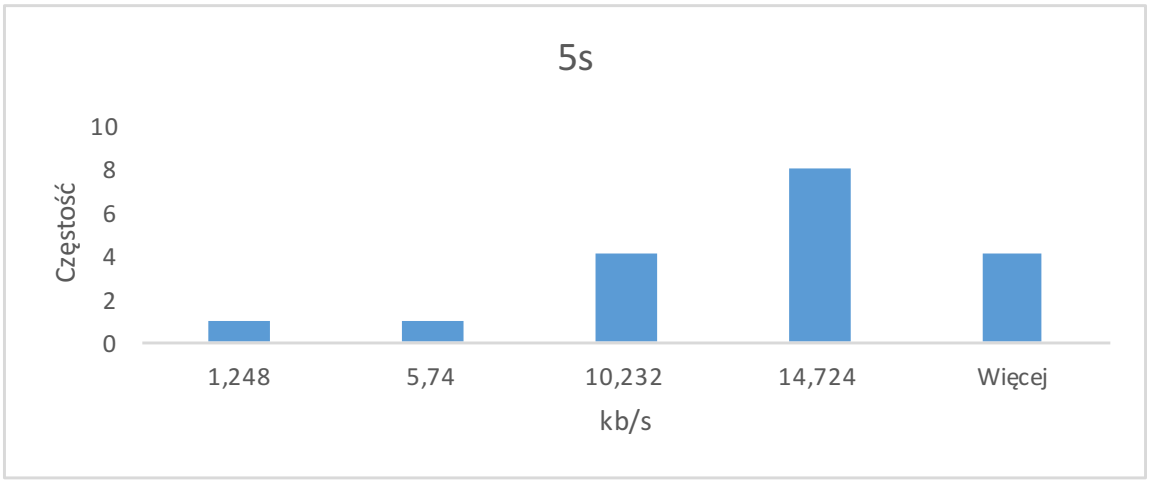
Następnie biorąc pod uwagę, że można przesyłać jeden duży QR kod lub wiele pomniejszych, wzór można przekształcić następująco:

- Niech q oznacza ilość QR kodów, jakie są wyświetlane w tej samej jednostce czasu.
- Zatem $V_2 = ((b * i)/s) * q$

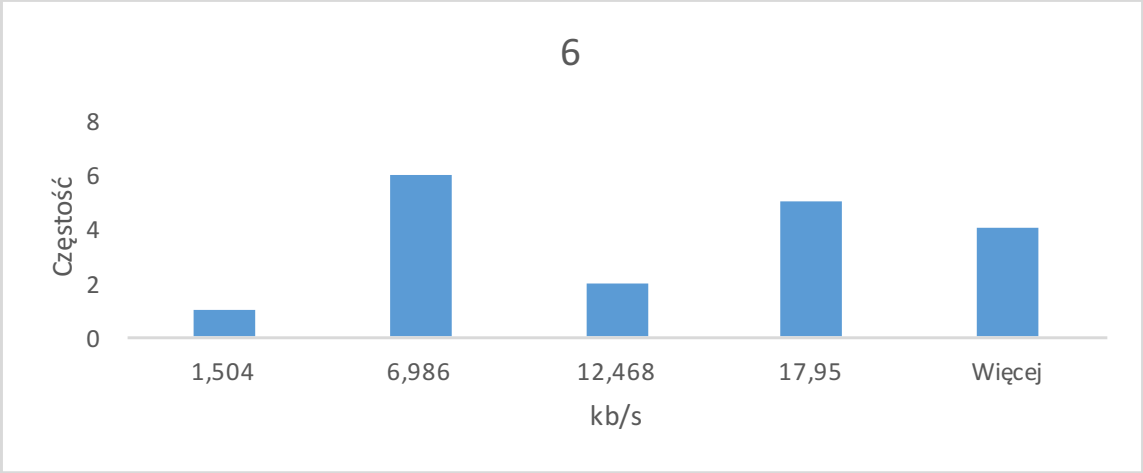
Należy jednak wziąć pod uwage, że kod QR posiada korekcję błędów na czterech różnych poziomach. Zatem część obszaru przez niego pokrywanego nie będzie nośnikiem danych, a powtórzeniem już istniejących, co także uwzględniamy. Wobec czego powyższy wzór należy przedstawić następująco: $V_3 = ((b * i)/s) * q - V_2 * C$,
gdzie C jest jednym z czterech { 7%, 15%, 25%, 30% }



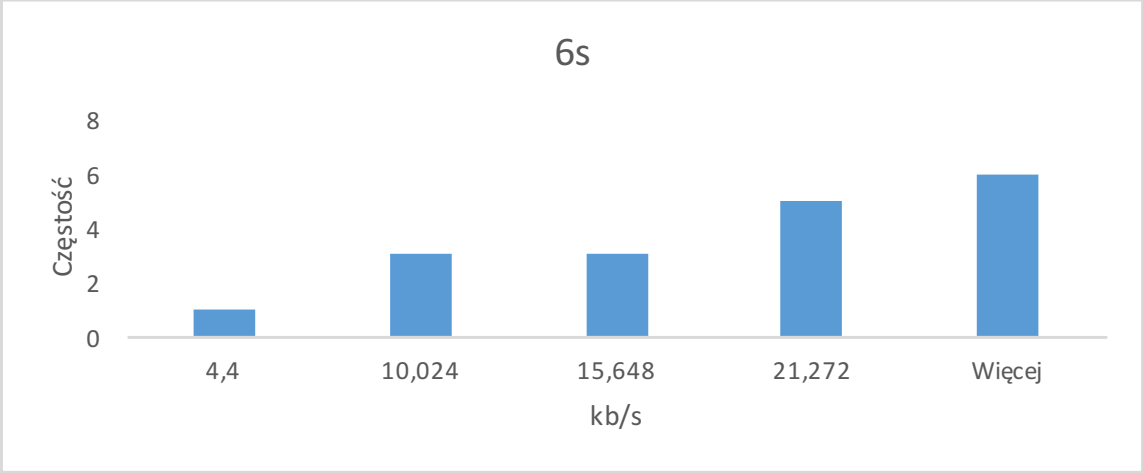
Rysunek 4.1: Prędkość nadawania QR kodów na iPhone 5c. Dolna oś przedstawia górną granicę przedziału dla transferu danych.



Rysunek 4.2: Prędkość nadawania QR kodów na iPhone 5s. Dolna oś przedstawia górną granicę przedziału dla transferu danych.



Rysunek 4.3: Prędkość nadawania QR kodów na iPhone 6. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

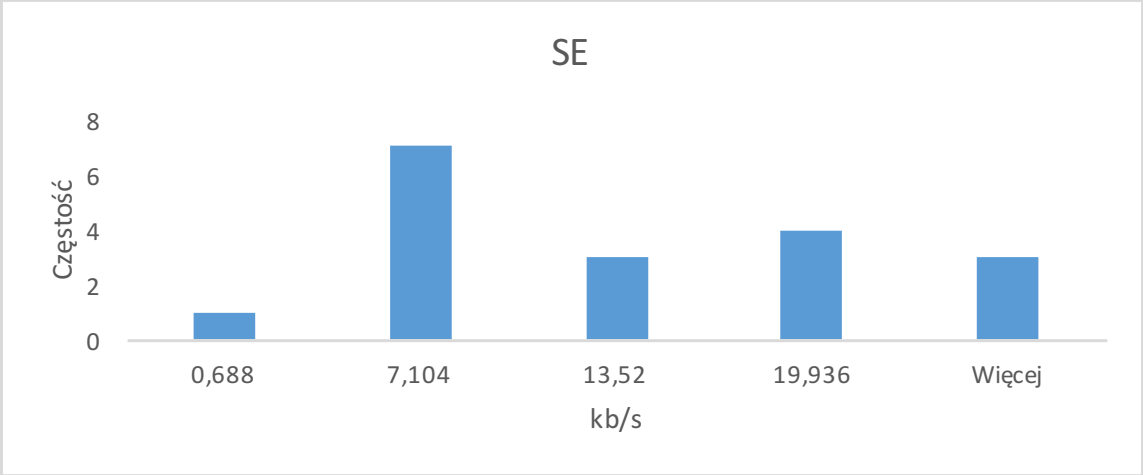


Rysunek 4.4: Prędkość nadawania QR kodów na iPhone 6s. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

¡ TABELKA SVG z zasiegami w metrach dla nadawania dla kazdego urzadzenia ¿

Opisz jeszcze właściwość dual domain pixels w 6,6s,7... Jego zużycie procesora graficznego. Opisz jeszcze wyświetlacze 5,5s,5c,SE... Które perspektywy mają silne, które gupią wartości rgba. ¡ TABELKA SVG z iloscia danych przeslanych w 10 sekund - akurat 10s, zeby focus zlapac ¿

Uśredniona względem zbadanych odległości liczba kilobitów na sekundę dla nadawania w czasie rzeczywistym danych prezentując tą samą ilość danych:
Urządzenie 5c - 6.7 Kb/s
Urzadzenie 5s - 8.0 Kb/s
Urzadzenie 6 - 13.3 Kb/s
Urzadzenie 6s - 16.0 Kb/s
Urzadzenie SE - 11.4 Kb/s

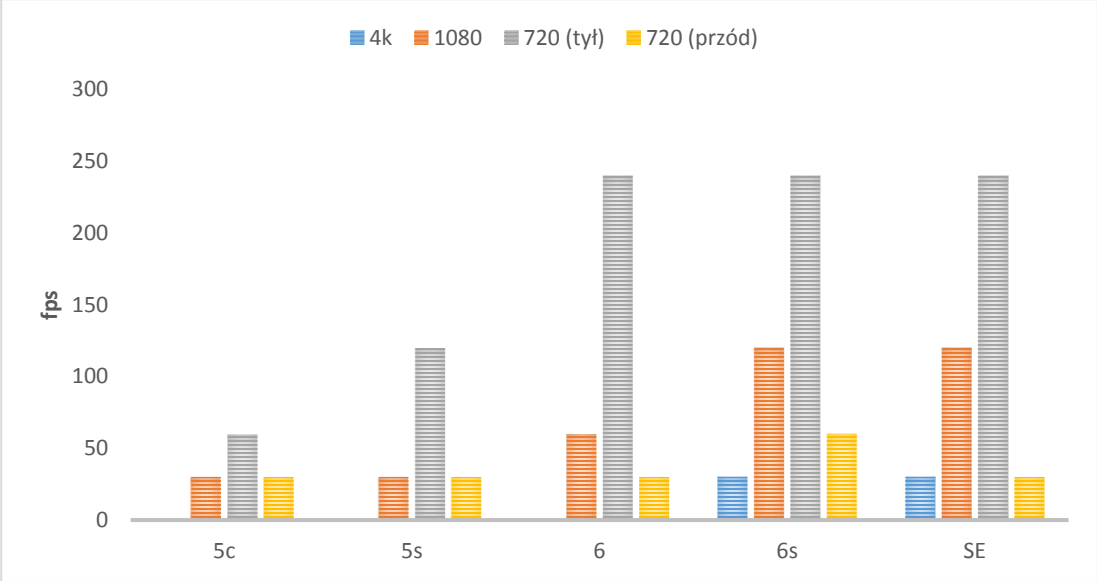


Rysunek 4.5: Prędkość nadawania QR kodów na iPhone SE. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

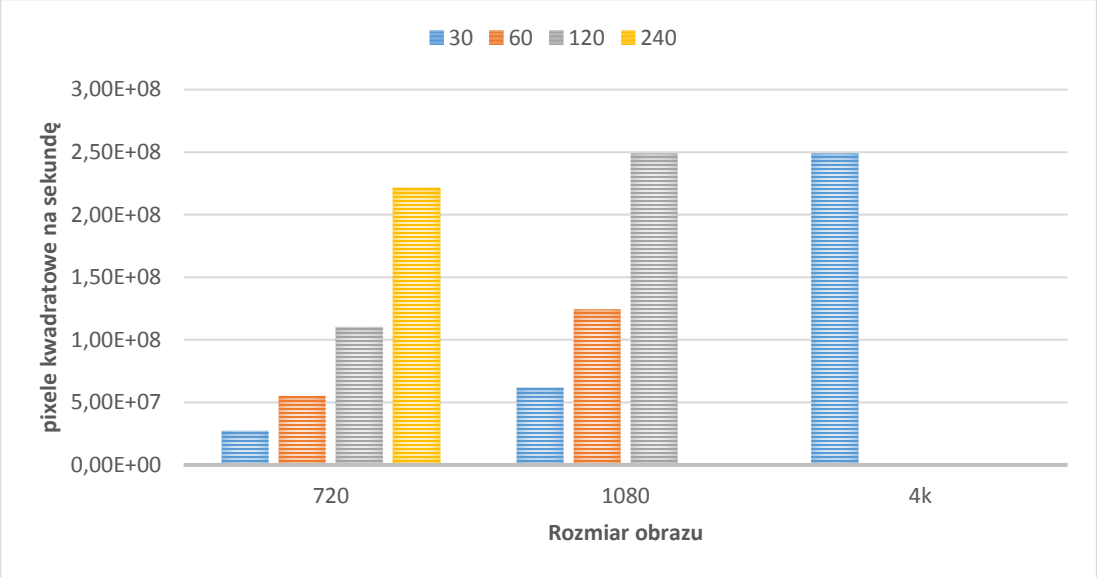
Badanie przepustowości odbioru

Ze względu na środowisko badania należy przyjąć, że urządzenie nadające jest w stanie propagować poprawnie wszystkie bity transmisji oraz połączenie między urządzeniami jest nieprzerywalne.

Powyższy wykres umożliwia wykonanie poniższego, dzięki któremu można przyjąć, która kamera jest potencjalnym maksymalnym przepływem.

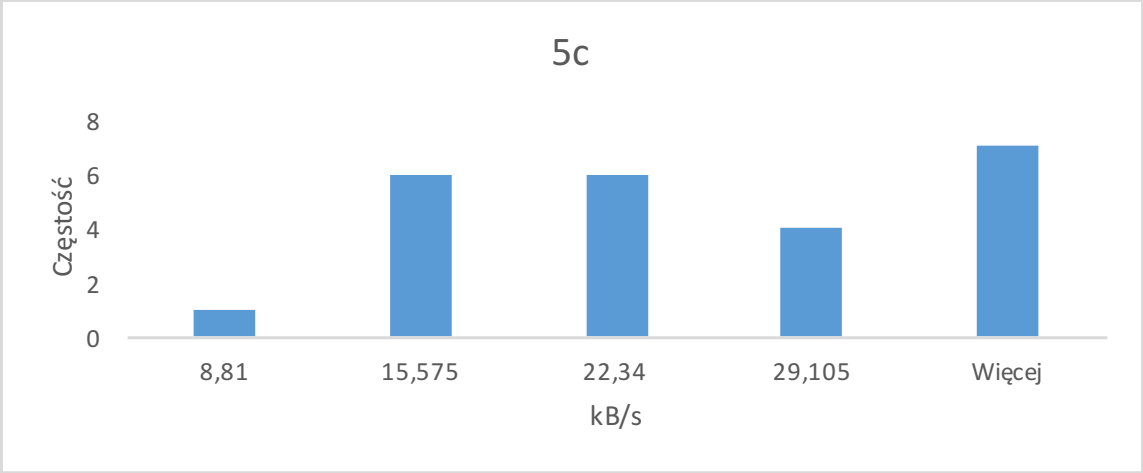


Rysunek 4.6: Wykres przedstawia możliwości poszczególnych ustawień kamery dla odbierania danych. Oś pionowa przedstawia ilość klatek na sekundę, pozioma natomiast rozdzielczość nagrywania.

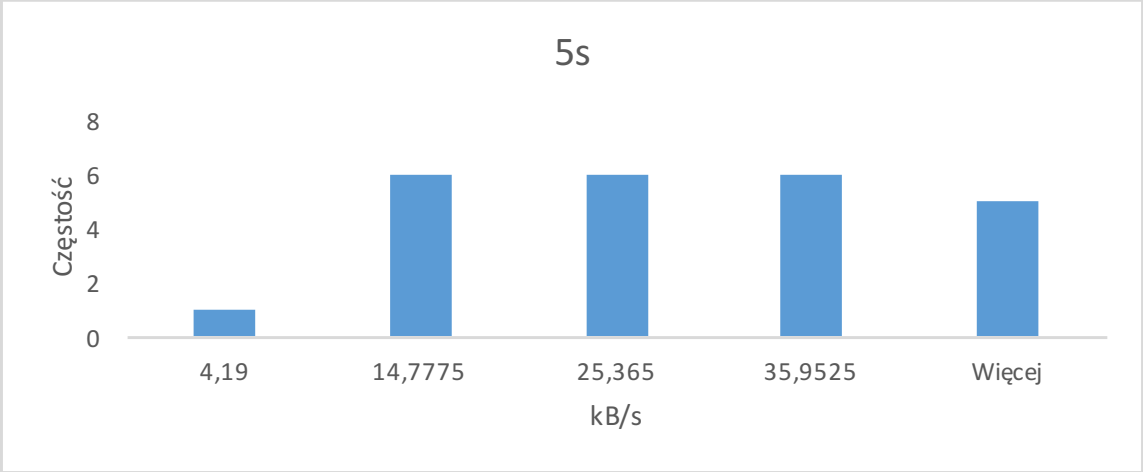


Rysunek 4.7: Wykres przedstawia zależność rozdzielczości obrazu względem potencjalnej ilości przesyłanych danych. Przy założeniu, że kazdy piksel jest w stanie nieść informację.

¡ ilosc danych taka sama ¿ ¡ TABELKA SVG z predkosciami odbioru danych odleglosc obojetna, max ilosc danych jaka jest w stanie chwycic metadata ¿



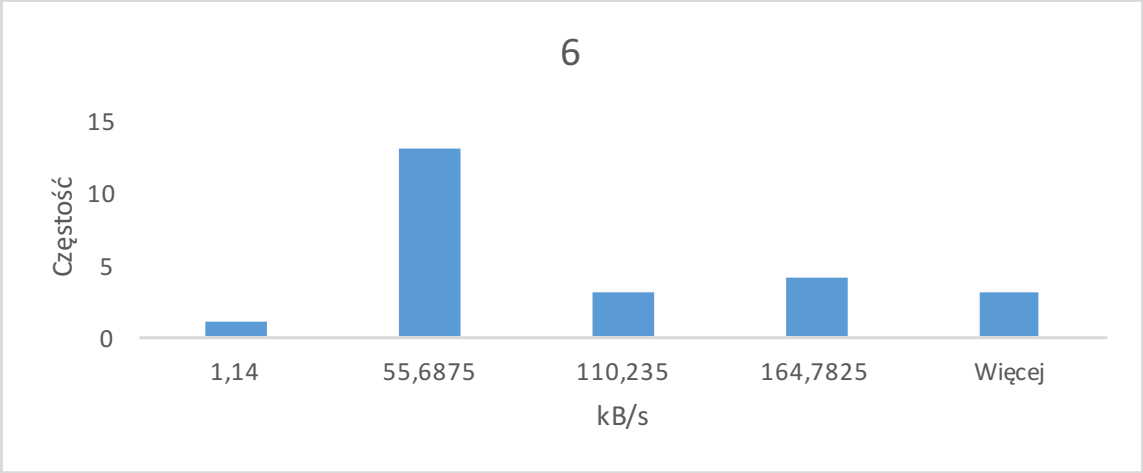
Rysunek 4.8: Prędkość czytania danych ze statycznego QR kodu na urządzeniu iPhone 5c. Dolna oś przedstawia górną granicę przedziału dla transferu danych.



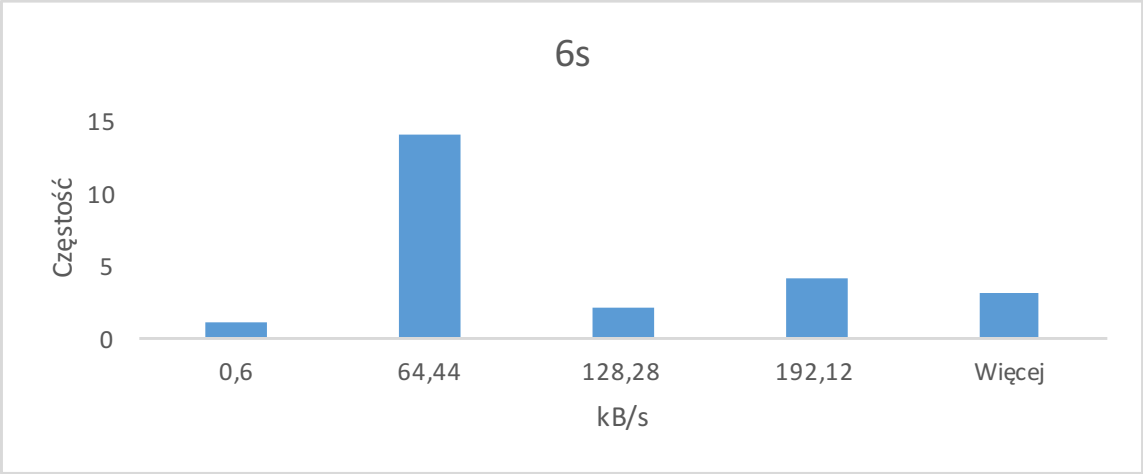
Rysunek 4.9: Prędkość czytania danych ze statycznego QR kodu na urządzeniu iPhone 5s. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

¡ TABELKA SVG z predkosciami odbioru danych dla takiej samej ilosci QR kodow na ekranie ¿

Liczba kilobajtów na sekundę dla odbierania w czasie rzeczywistym danych z drugiego urządzenia prezentującego w sposób ciągły ten sam symbol QR:
Urządzenie 5c - 31.0 KB/s
Urzadzenie 5s - 34.0 KB/s
Urzadzenie 6 - 49.0 KB/s
Urzadzenie 6s - 58.0 KB/s
Urzadzenie SE - 58.0 KB/s



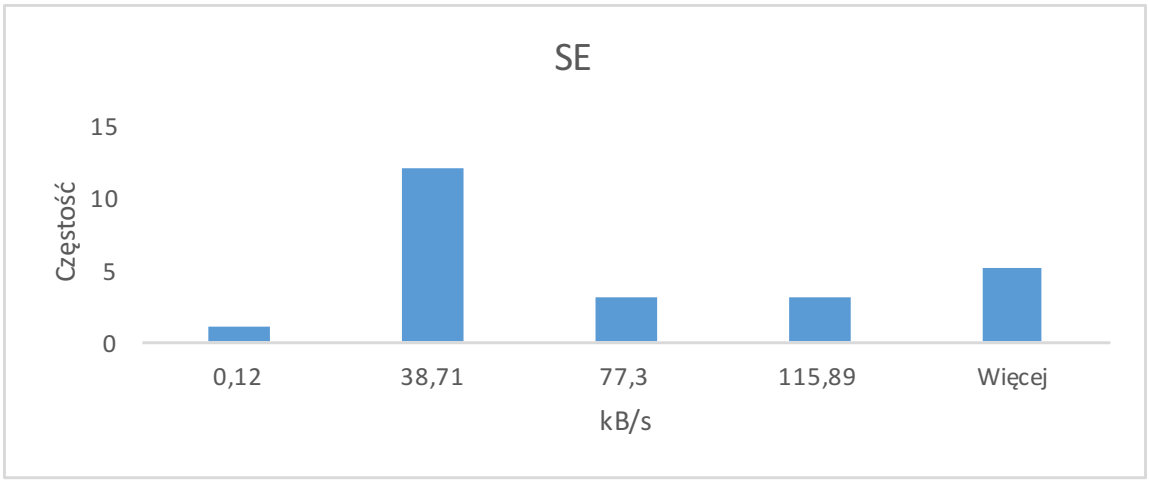
Rysunek 4.10: Prędkość czytania danych ze statycznego QR kodu na urządzeniu iPhone 6. Dolna oś przedstawia górną granicę przedziału dla transferu danych.



Rysunek 4.11: Prędkość czytania danych ze statycznego QR kodu na urządzeniu iPhone 6s. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

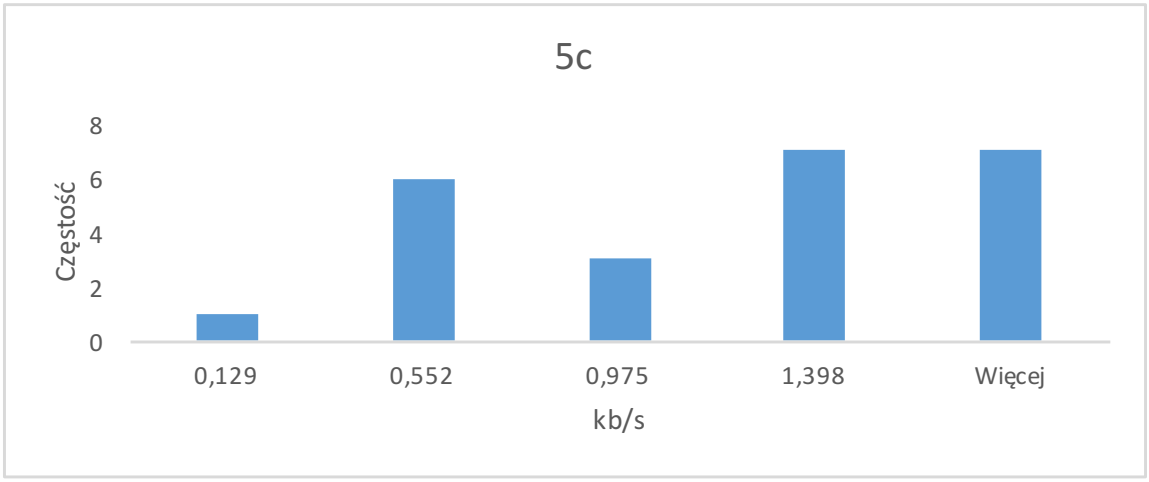
Badanie przepustowości w komunikacji

W tym kroku należy przyjąć, że optymalna przepustowość to minimum z przepustowości nadawania i odbierania dla obu urządzeń.



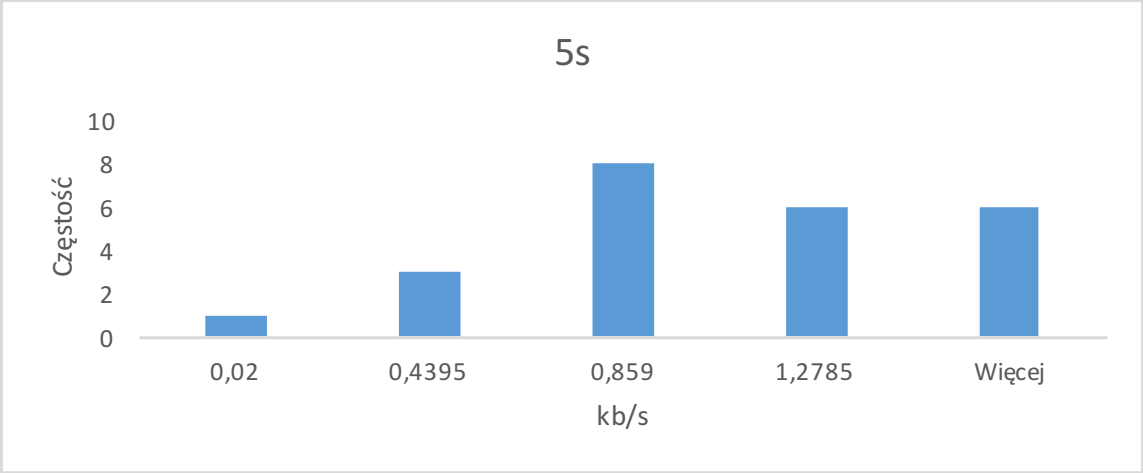
Rysunek 4.12: Prędkość czytania danych ze statycznego QR kodu na urządzeniu iPhone SE. Dolna oś przedstawia górną granicę przedziału dla transferu danych.

¡ TABELKA SVG z histo dla predkosci w komunikacji dla poszczegolnych badan ¿

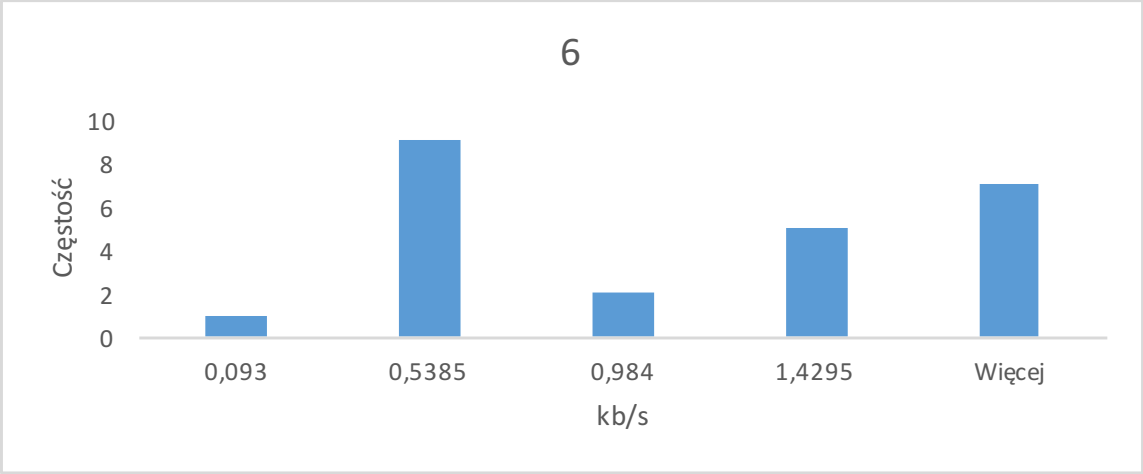


Rysunek 4.13: Caption

¡ TABELKA SVG z czasem potrzebnym by wymienic sie kluczem takiej samej dlugosci 4096 ¿



Rysunek 4.14: Caption



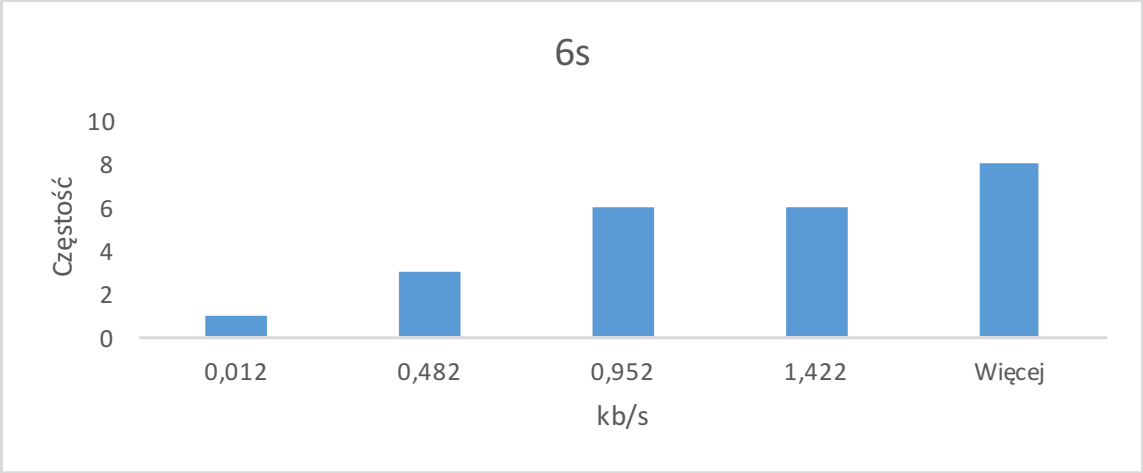
Rysunek 4.15: Caption

4.2 Możliwości techniczne transmisji danych w praktyce

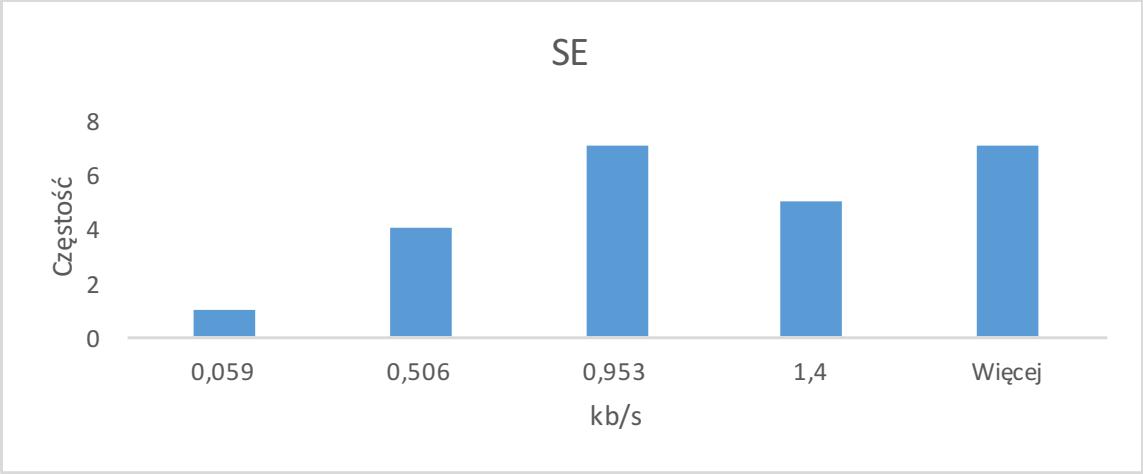
Wykresy i opisy

Urządzenia są w stanie wymienić się danymi dowolnej długości, ograniczonej jedynie pamięcią potrzebną do zaalokowania w aplikacji.

Każdy telefon ma określoną ilość ramu, więc żeby zapisywać potrzeba robic przerwę w nagrywaniu, by odciążyć go na czas dostępu do pamięci wbudowanej i zapisac dotychczasową kopię odebranych danych. Jest to ważny zabieg, gdyż zbliżając się do limitu danych jaki jest w stanie przechować urządzenie, zwiększamy ryzyko utraty innych tymczasowych danych. Urzadzenia marki Apple w odróżnieniu od innych dostępnych na rynku(2016r.) smartfonów są bardzo dobrze zoptymalizowane pod kątem dostępów do pamięci wbudowanej. Cała pamięć jest zbudowana w oparciu o technologię Flash Drive. Zapewnia ona szybszy dostep do pamięci, przez jej statyczność, która w odróżnieniu od Hard Drive, nie opóźnia transmisji danych



Rysunek 4.16: Caption



Rysunek 4.17: Caption

podczas odczytu i zapisu.

4.3 Testy prototypu i dyskusja

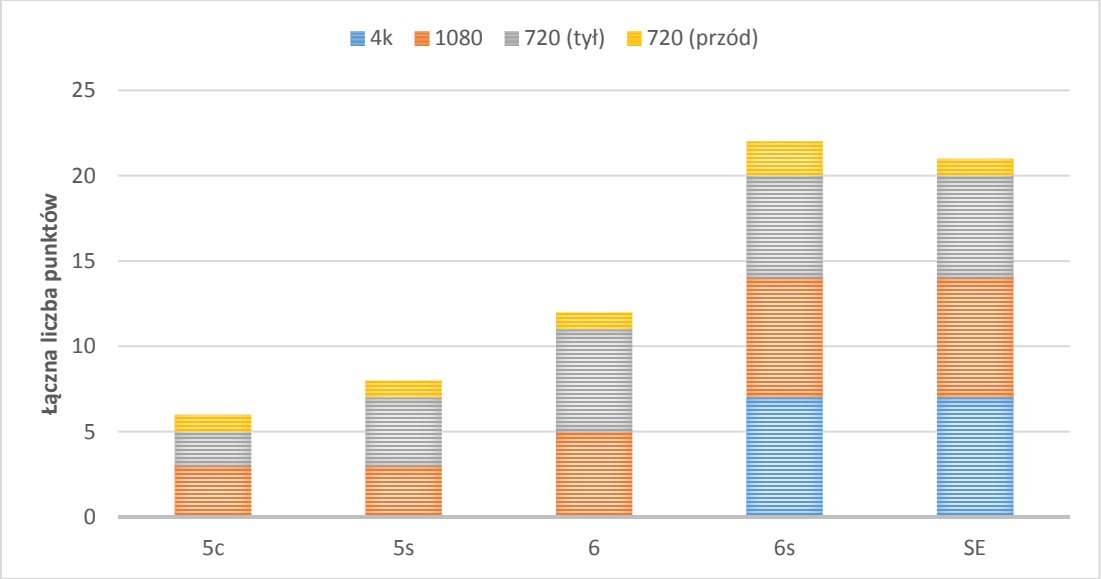
Podjęte próby - osiągnięcia

Podczas implementacji projekt nie przewidywał migracji między trzema wersjami języka Swift i kompilatora Clang. Wymagało to zapoznania się z nowymi wymogami platformy oraz przeprojektowaniem kolejkowania wątków i dystrybucji zadań w systemie. Zmiany te były na tyle znaczące, że zmusiły do ponownego przeprowadzenia badań, w celu ustalenia poprawności obliczeń. Sama zmiana platformy na nowszą nie ulepszyła żadnej z funkcjonalności aplikacji. Zmieniło to jedynie sposób dostępu do plików systemowych i kolejkowania wątków w całej platformie iOS. Wszelkie odwołania języka Swift do Objective-C pozostały podobne

z wyróżnieniem etykietowania argumentów funkcji w sposób jawny. Kompilator wspierający język otrzymał jedną znaczącą poprawkę - nie próbuje poprawiać kodu po pierwszym dużym błędzie kompilacji, co przyspiesza kompilację większych projektów.

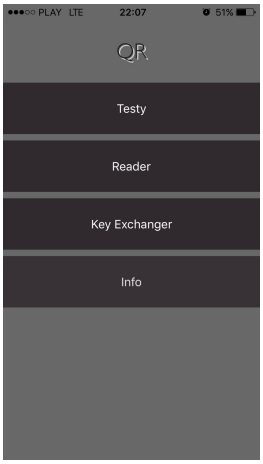
5 Podsumowanie

Projektując aplikację głównym założeniem było przesłanie jak największej ilości bajtów w jak najkrótszym czasie. Badanie obejmowało pięć telefonów marki Apple oraz przedziały czasu w jakich były one wydawane, można było spodziewać się znacznego wzrostu wydajności kolejnych względem poprzednich modeli. Tymczasem platforma iOS zmieniając programistyczne implementacje poszczególnych klas udostępnionych deweloperom, pozostawiała algorytmy i optymalizacje samemu kompilatorowi clang i jego domyślnym ustawieniom. Biorąc to wszystko pod uwagę, udało się zakończyć badania sukcesem i przedstawić zależności między poszczególnymi aspektami technicznymi oraz ograniczeniami samej platformy. Aplikacja oraz jej możliwości pozwalają na przesłanie niewielkiej ilości danych, dobranych proporcjonalnie do potrzeb. Umożliwia ona wymianę kluczy alternatywnym kanałem, odpornym na interferencje i nadmiar infrastruktur. Nie wymaga uruchamiania bluetooth, czy wifi, przez co uniemożliwia wykrycie i podsłuchanie transmisji nawet z bliska. Wartym uwagi mogłoby być nasłuchiwanie pracy samego procesora w celu ustalenia, czy w danym momencie następuje wymiana informacji, czy jest to jego stan bezczynności. Aplikacja umożliwia dalszy rozwój przez implementację różnych algorytmów wymiany danych czy protokołów inicjalizujących bezpieczną transmisję danych. Urządzenia, które były celem badania zostały ocenione na podstawie ich własności technicznych i można je przedstawić następująco:

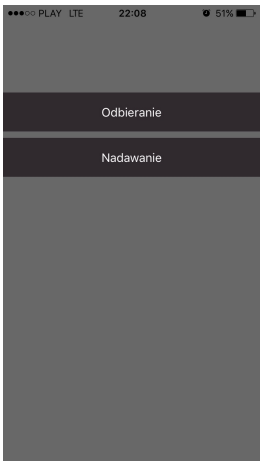


Rysunek 5.1: Wykres sporządzono na podstawie ilości pikseli kwadratowych na sekundę w transmisji danych dla maksymalnych ustawień kamery.

Wycinki ekranów:

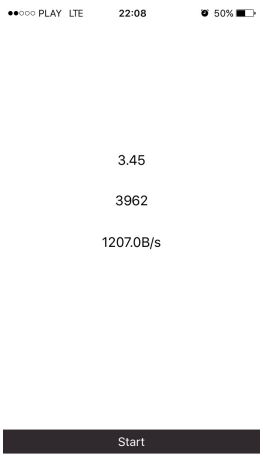


Rysunek 5.2: Menu aplikacji

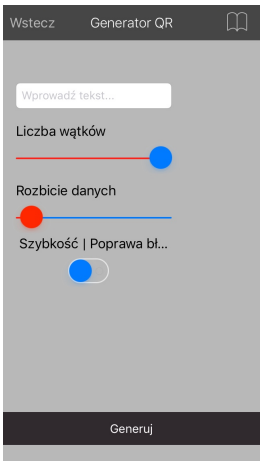


Rysunek 5.3: Wybór dla Testów

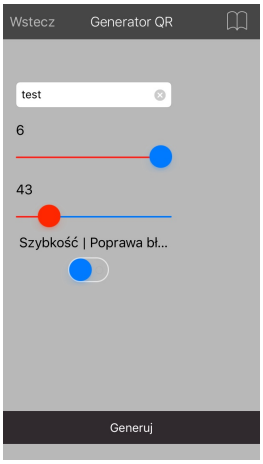
Rysunek 5.4: Moduł czytania czterech QR na raz w czasie rzeczywistym



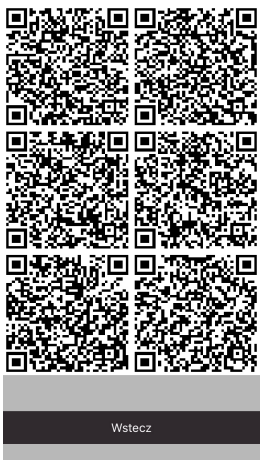
Rysunek 5.5: Testowanie transferu czytania QR kodów



Rysunek 5.6: Widok parametryzacji dla generowania QR kodów



Rysunek 5.7: Ustawione parametry dla generowania QR kodów



Rysunek 5.8: Generowanie QR kodów

Rysunek 5.9: Wymiana klucza za pomocą QR kodów

Bibliografia

[1]

A Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD. Na płycie znajduje się projekt Xcode, zawierający aplikację testową oraz jej dokumentację.