Niezawodność i diagnostyka układów cyfrowych 2

Transmisja w systemie Hybrid ARQ (HARQ) - Projekt

Radosław Zimoch 263963
Joanna Zoglowek 264452

24 czerwca 2023

1. Cele projektu

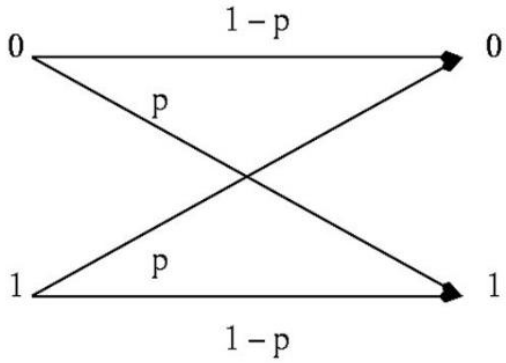
Celem projektu jest implementacja kanału komunikacyjnego opartego na modelach Binary Symmetric Channel (BSC) i Gilberta-Elliotta oraz systemu transmisji Hybrid Automatic Repeat Request (HARQ) z wykorzystaniem kodów detekcyjnych i korekcyjnych. Następnie przeprowadzenie symulacyjnej oceny skuteczności transmisji dla różnych parametrów kanałów i kodów.

2. Opis projektu

W ramach projektu stworzyliśmy program symulujący działanie systemu transmisji HARQ, który W ramach projektu opracowaliśmy program symulujący działanie systemu transmisji Hybrid ARQ (HARQ), który umożliwia elastyczną modyfikację różnych zmiennych projektu.

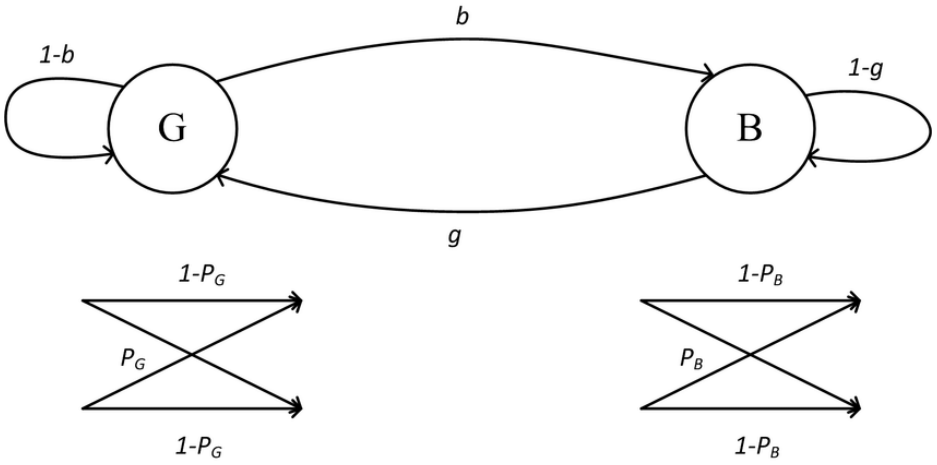
W naszym projekcie zaimplementowaliśmy dwa modele kanałów: Binary Symmetric Channel (BSC) oraz Gilbert-Elliott. Każdy z tych modeli ma swoje parametry, które użytkownik może modyfikować, takie jak prawdopodobieństwo błędu bitowego, prawdopodobieństwo przejścia kanału z dobrego na zły oraz zły na dobry itp.

a) Binary Symmetric Channel (BSC) jest modelem kanału, w którym występują błędy bitowe niezależnie od wartości przesyłanych bitów. To znaczy, że każdy przesłany bit może być zakłócony z pewnym określonym prawdopodobieństwem p (Rys.1). BSC jest prostym modelem, który znajduje zastosowanie w wielu dziedzinach komunikacji.



Rysunek 1 Schemat modelu kanału BSC

b) Gilbert-Elliott to bardziej zaawansowany model kanału, który uwzględnia zmienność jakości transmisji. W tym modelu istnieją dwa stany kanału: dobry i zły. Kanał może przejść z jednego stanu do drugiego z określonym prawdopodobieństwem (b lub g na Rys.2). Model Gilberta-Elliotta znajduje zastosowanie w sytuacjach, w których jakość kanału może ulegać zmianom, na przykład w transmisji bezprzewodowej.



Rysunek 2 Schemat modelu kanału Gilberta - Elliota

Wykorzystaliśmy kody detekcyjne takie jak bit parzystości, CRC8, CRC16 i CRC32. Każdy z tych kodów generuje sumę kontrolną na podstawie przesyłanych danych, która jest dołączana do danych i pozwala na wykrycie błędów w procesie odbioru. Bit parzystości sprawdza parzystość liczby bitów danych, umożliwiając wykrycie prostych błędów transmisji. Kody CRC (Cyclic Redundancy Check) generują bardziej skomplikowane sumy kontrolne, uwzględniając różne kombinacje bitów danych, co pozwala na wykrycie szerszej gamy błędów.

Wielomiany jakie dobraliśmy dla naszych kodów zostały wybrane jako najlepsze wielomiany ze strony Carnegie Mellon University i wyglądają następująco:

- CRC-8:
  $x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + x + 1$
- CRC-16:
  $x^{16} + x^{12} + x^5 + 1$
- CRC-32:
  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Dodatkowo, w celu nie tylko wykrywania, ale także korekcji błędów w przesyłanych danych, zaimplementowaliśmy kod Reed-Solomon (kod RS). Kod RS jest jednym z popularnych kodów korekcyjnych stosowanych w transmisji danych. Działa na zasadzie dodawania nadmiarowych informacji do danych, co umożliwia wykrycie i naprawę błędów w procesie odbioru. Kod RS jest efektywny w korygowaniu różnych rodzajów błędów, takich jak błędy pojedynczych bitów, uszkodzenia pakietów danych czy błędy przekłamań.

W ramach naszych testów skupimy się na Typie II Hybrid ARQ. W tym trybie pierwsza wiadomość jest kodowana tylko kodem detekcyjnym. Jeśli pierwsza transmisja zostanie odebrana bez błędów, proces jest zakończony. Jeżeli jednak zostaną wykryte błędy, wiadomość jest wysyłana ponownie, tym razem z dodatkową częścią korekcyjną.

3. Planowane efekty projektu

Poza implementacją systemu transmisji HARQ, planujemy przeprowadzić symulacyjne badania, które pozwolą nam ocenić skuteczność transmisji dla różnych parametrów kanałów i różnych kodów detekcyjnych W przypadku testów, planujemy wykorzystać modele kanału z różnymi parametrami, takimi jak:

- Prawdopodobieństwo błędu bitowego
- Prawdopodobieństwo przejścia kanału ze stanu dobrego na zły oraz zły na dobry
- Długość przesyłanej wiadomości
- Liczba potrzebnych retransmisji

Wyniki badań pozwolą nam ocenić skuteczność systemu transmisji HARQ dla różnych przypadków, co pozwoli nam określić optymalne parametry systemu transmisji dla danego kanału.

4. Implementacja

Zdecydowaliśmy się na zaimplementowanie naszego systemu transmisji w języku programowania Java. Projekt jest dostępny na repozytorium github:

https://github.com/radziu2402/HybridARQ-NIDUC2-projekt

Program pobiera zdjęcie w formacie BMP (wybór tego formatu pozwala na mniejsze problemy przy ponownym odtwarzaniu obrazka z przesłanych bitów), a następnie dzieli je na paczki i po kolei przesyła, jeśli wiadomość zakodowana detekcyjnie przejdzie bezbłędnie przez kanał to przechodzimy do kolejnej paczki, natomiast jeśli wykryty zostanie błąd to paczka wysyłana jest ponownie z część korekcyjna, która próbuje naprawić błędy, a jeśli jej się nie uda to następuje retransmisja, aż do skutku. Używany do testów obrazek miał 7,77 KB (bajtów: 7 958).

5. Zakres testów

Mamy do dyspozycji wiele parametrów, którymi możemy manipulować, testy są przeprowadzane z podziałem na:

a) Rodzaj wybranego kanału BSC i Gilberta-Elliotta
b) Prawdopodobieństwo błędu w tym kanale
c) Rodzaj wybranego kodu detekcyjnego (niektóre przepuszczają więcej błędów co powoduje zmienione piksele w wyjściowym obrazie)
d) Długość jednej paczki bitów (musimy się jednak ograniczyć do wielokrotności 1 bajta, czyli 8,16,24 bity itd. z uwagi na implementację kodu RS)
e) Długość części korekcyjnej kodu RS, tutaj również są to wielokrotności 1 bajta

6. Wyniki testów

Kanał BSC:

Aby obrazek w ogóle został wygenerowany prawdopodobieństwo błędu musi być dość niskie dlatego nasze badania przeprowadzane są dla trzech prawdopodobieństw błędu:

- 1%
- 3%
- 5%

Po konsultacji z prowadzącym postanowiliśmy przyjąć następujące parametry:

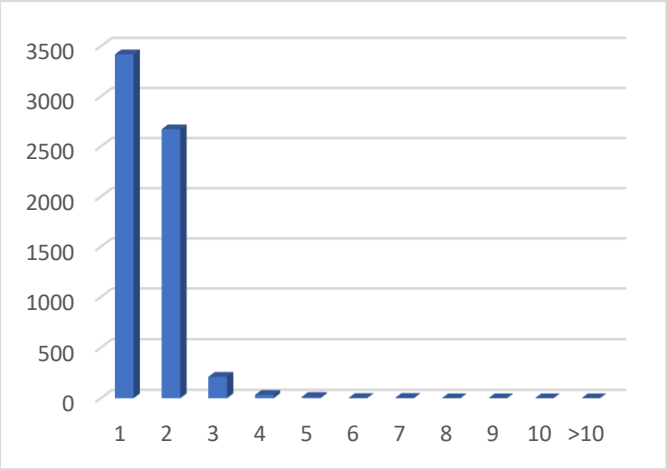
- Długość pakietu: 16 bajtów
- Długość części korekcyjnej: 6 bajtów

Poniższe wykresy przedstawiają histogramy ilości transmisji danego pakietu:

- oś X - Ilość transmisji, która była potrzebna do przesłania pakietu
- oś Y - Ilość pakietów, które zostały wysłane

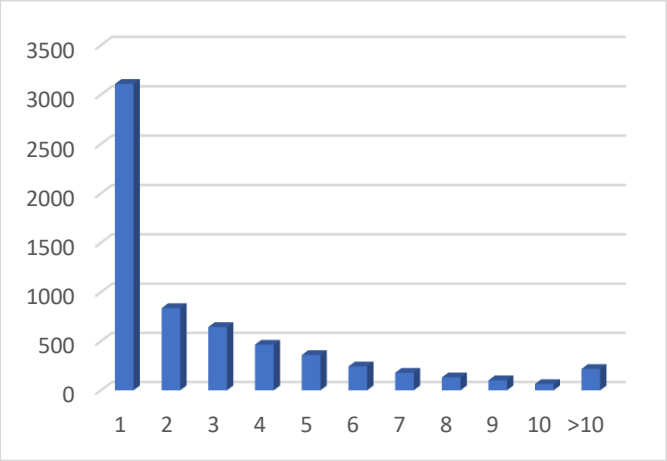
Kod detekcyjny – bit parzystości

Prawdopodobieństwo błędu 1%:



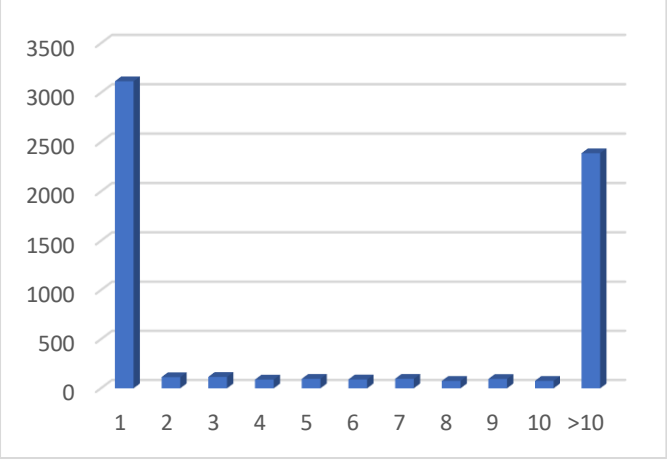
Wykres 1

Prawdopodobieństwo błędu 3%:



Wykres 3

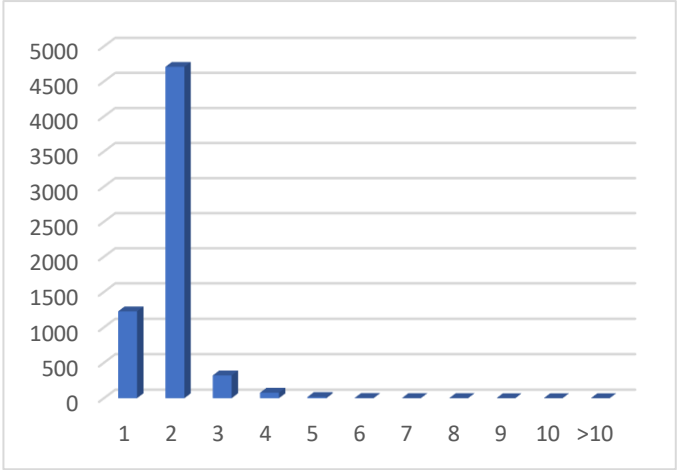
Prawdopodobieństwo błędu 5%:



Wykres 5

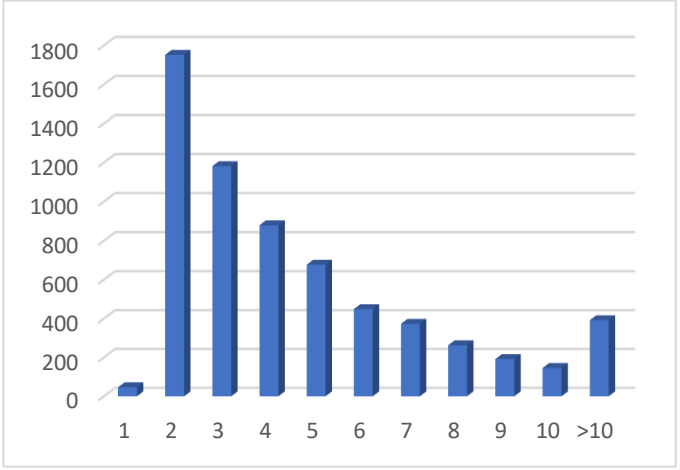
Kod detekcyjny – CRC32

Prawdopodobieństwo błędu 1%:



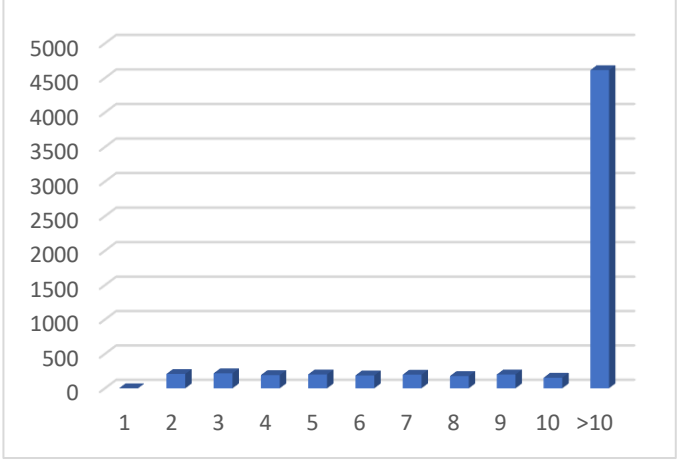
Wykres 2

Prawdopodobieństwo błędu 3%



Wykres 4

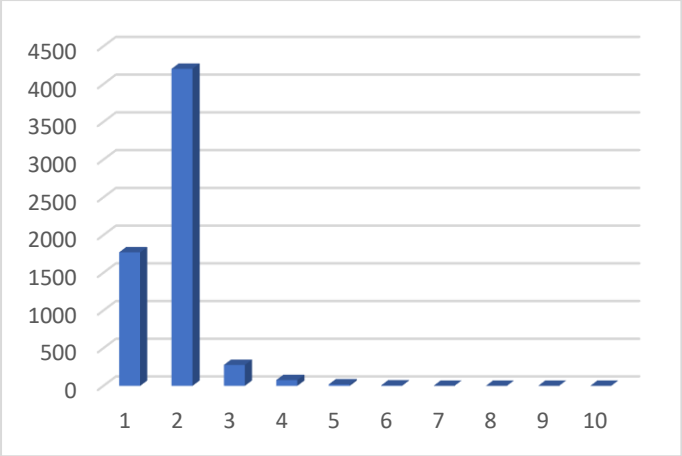
Prawdopodobieństwo błędu 5%



Wykres 6

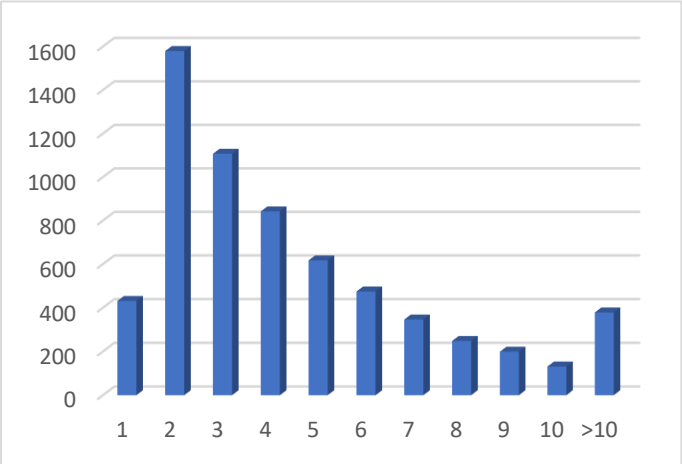
Kod detekcyjny – CRC8

Prawdopodobieństwo błędu 1%:



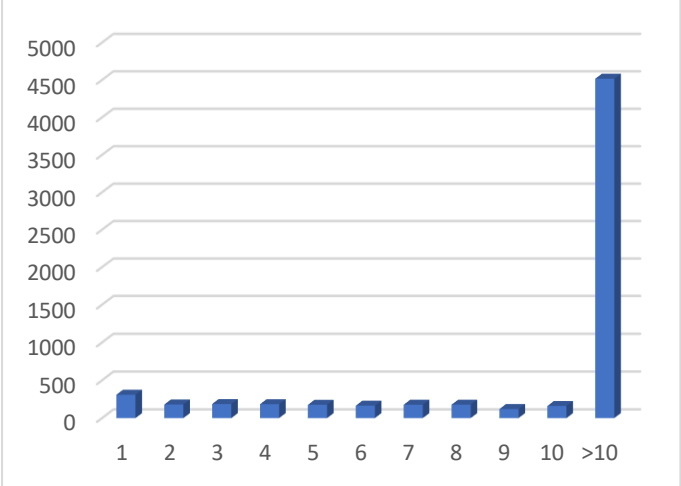
Wykres 7

Prawdopodobieństwo błędu 3%:



Wykres 9

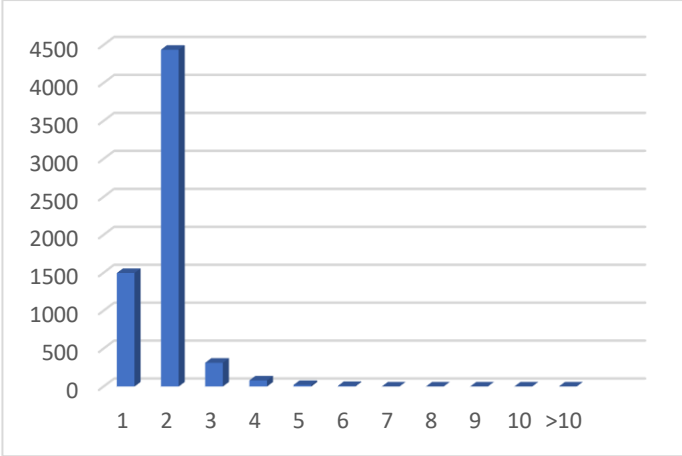
Prawdopodobieństwo błędu 5%:



Wykres 11

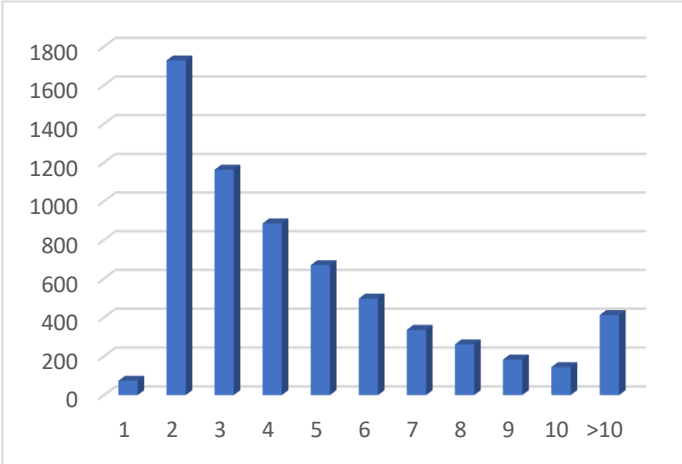
Kod detekcyjny – CRC16

Prawdopodobieństwo błędu 1%:



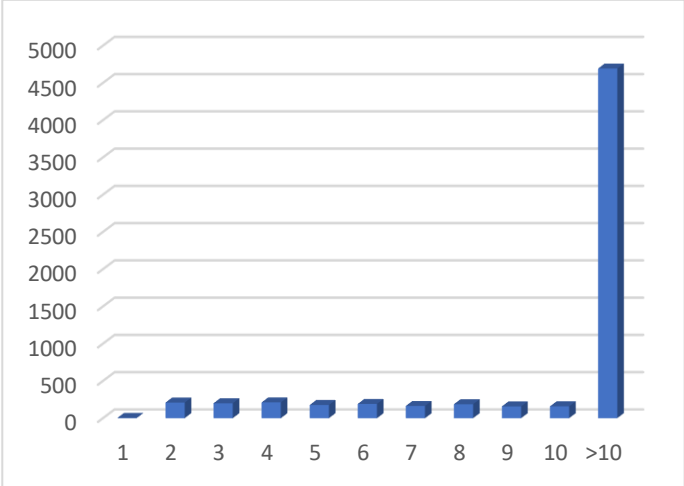
Wykres 8

Prawdopodobieństwo błędu 3%:



Wykres 10

Prawdopodobieństwo błędu 5%:

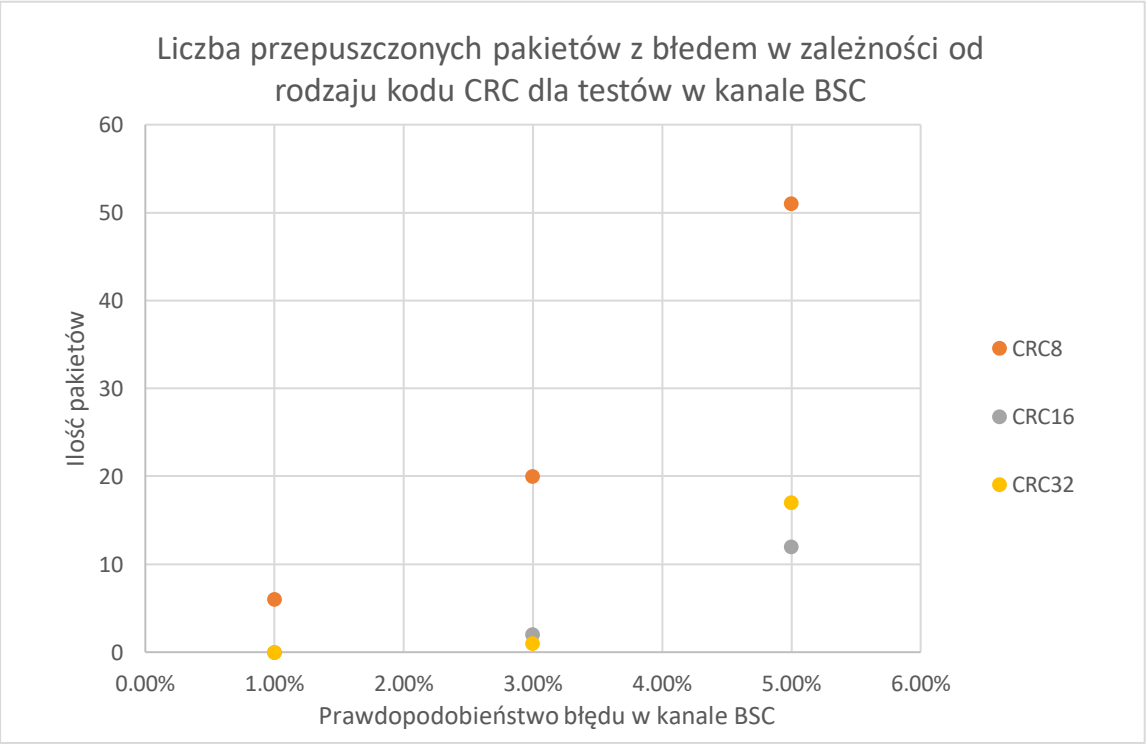


Wykres 12

Na wykresach możemy zaobserwować - jak wraz ze wzrostem prawdopodobieństwa, maleje liczba pakietów przesłanych poprawnie za jednym razem na rzecz większej ilości retransmisji pakietów. Jest to spodziewane zachowanie przy tym typie transmisji. Przy prawdopodobieństwie 1%, kod korekcyjny wykazuje dużą skuteczność, zgodnie z oczekiwaniami. Możemy również zauważyć, że tak dobrane parametry czyli: 16 bajtowa wiadomość i tylko 6 bajtowa część korekcyjna nie radzi sobie z naprawą błędów już przy prawdopodobieństwie około 5% i na pewno wtedy transmisja jest nieefektywna.

Tabela 1 Liczba przepuszczonych pakietów z błędami

Prawdopodobieństwo błędu w kanale	bit parzystości	CRC8	CRC16	CRC32
1%	1706	6	0	0
3%	2951	20	2	1
5%	3241	51	12	7



Wykres 13

Zgodnie z oczekiwaniami, bit parzystości nie jest w stanie zabezpieczyć tak długiej wiadomości i przepuszcza wiele błędów. Natomiast poszczególne wersje kodu CRC, wykazują liniową zależność przepuszczonych błędów od prawdopodobieństwa ich wystąpienia. Im dłuższy kod CRC tym skuteczność jest większa. W przypadku tego kodu nie wykrycie niektórych błędów, wynika z tego, że istnieją takie kombinację błędów, które powodują, że reszta modulo przy dzieleniu przez wielomian CRC jest taka sama jak reszta dla wiadomości prawidłowej.

Kanał Gillberta-Elliota:

W drugiej części przeprowadzonych testów, użyto kanału Gillberta-Elliota który charakteryzuje się tym, że wprowadza do transmisji błędy typu burst. Zmienianym parametrem dla owego kanału było prawdopodobieństwo przejścia do stanu złego. Prawdopodobieństwo wyjścia ze stanu błędu było
ustawione jako stałe i wynosiło 10%. Natomiast prawdopodobieństwa błędów w stanach były następujące:

- stan dobry – 1%
- stan zły – 10%

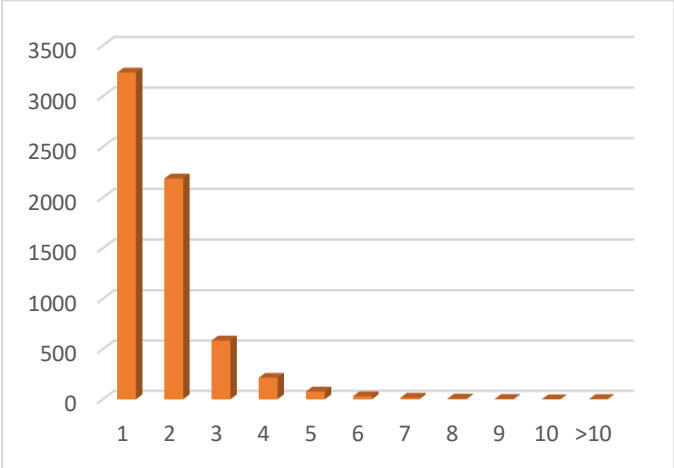
Poniższe wykresy przedstawiają histogramy ilości transmisji danego pakietu:

- oś X - Ilość transmisji, która była potrzebna do przesłania pakietu
- oś Y - Ilość pakietów, które zostały wysłane

Kody CRC8 i CRC16 zostały pominięte na wykresach z powodu praktycznie identycznych wyników jak CRC32

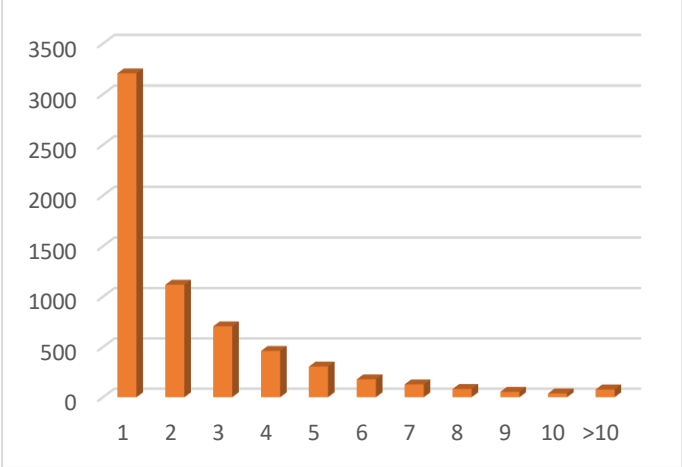
Kod detekcyjny – bit parzystości

Prawdopodobieństwo przejścia w stan zły 1%:



Wykres 74

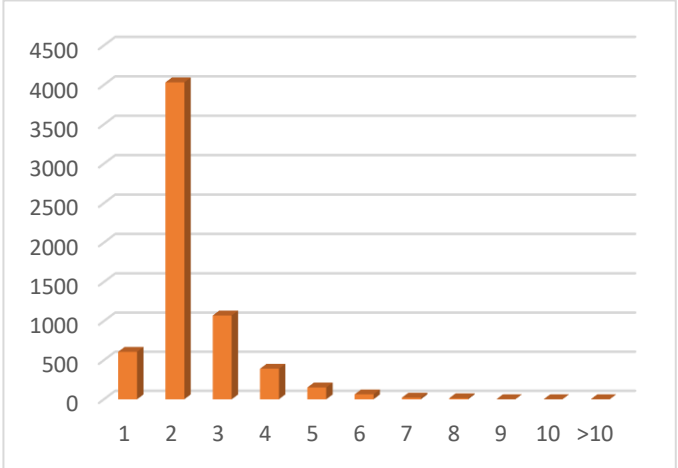
Prawdopodobieństwo przejścia w stan zły 3%:



Wykres 16

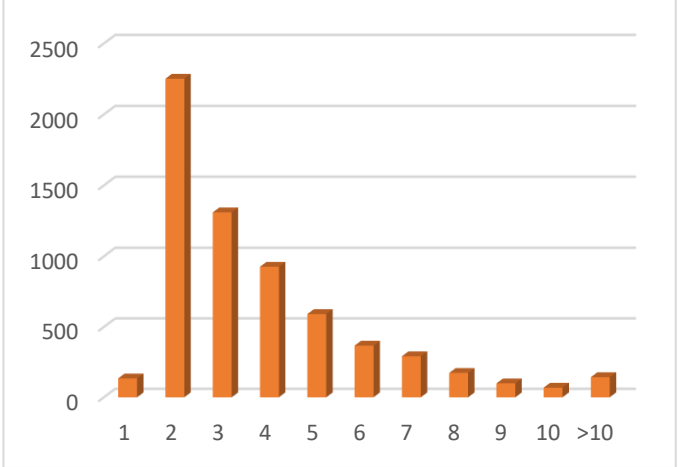
Kod detekcyjny – CRC32

Prawdopodobieństwo przejścia w stan zły 1%:



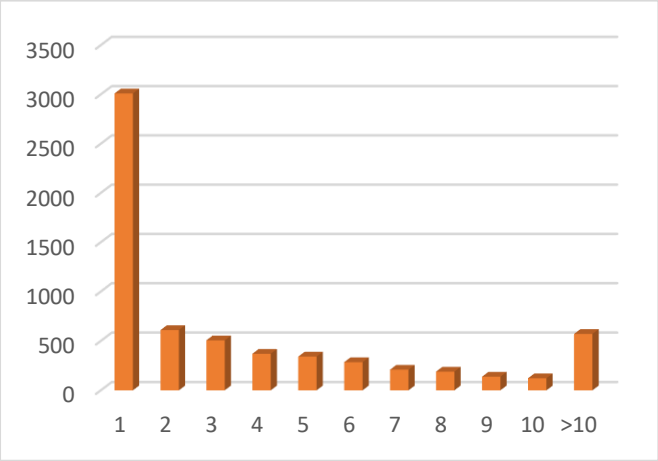
Wykres 15

Prawdopodobieństwo przejścia w stan zły 3%



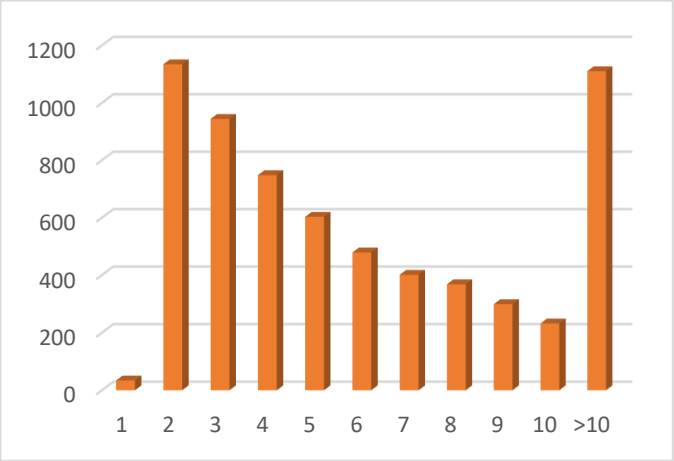
Wykres 17

Prawdopodobieństwo przejścia w stan zły 5%:



Wykres 18

Prawdopodobieństwo w stan zły 5%:



Wykres 19

Tabela 2 Liczba przepuszczonych pakietów z błędami

Prawdopodobieństwo przejścia w stan zły	bit parzystości	CRC8	CRC16	CRC32	% czasu transmisji w stanie dobrym	% czasu w transmisji w stanie złym
1,00%	2332	3	0	0	90,94	9,06
3,00%	2879	12	1	0	76,88	23,12
5,00%	3142	21	3	2	66,62	33,38

Zgodnie z założeniami bit równie słabo zabezpiecza paczki zarówno przy tego typu kanale. Natomiast dla błędów typu burst kody CRC radziły sobie nieznacznie lepiej, co jest logicznym wynikiem, ponieważ trudniej jest uzyskać odpowiednią kombinację błędów, które ten rodzaj kodu przepuszcza w momencie gdy następowałyby one po sobie.
Porównując wykresy z poprzednimi wynikami dla kanału BSC możemy dojść do wniosku że kod RS równie dobrze radzi sobie z błędami typu burst.

7. Podsumowanie

Wyniki eksperymentów potwierdziły, że wprowadzenie mechanizmu HARQ znacznie poprawia jakość transmisji danych. Dzięki możliwości powtórnego przesyłania uszkodzonych pakietów, system HARQ umożliwia skuteczną naprawę błędów i zwiększa niezawodność transmisji. Wykazano, że implementacja HARQ pozwala na redukcję liczby błędów bitowych i poprawia ogólną wydajność.

Analiza różnych parametrów kanałów i kodów detekcyjnych wykazała, że skuteczność transmisji jest w dużej mierze zależna od charakterystyki kanału i zastosowanych kodów. Model Gilberta-Elliotta, uwzględniający zmienność jakości transmisji, potwierdził swoje znaczenie jako bardziej realistyczny model kanału w porównaniu do prostszego modelu BSC.

Wprowadzenie kodów detekcyjnych, takich jak bit parzystości, CRC8, CRC16 i CRC32, pozwoliło na skuteczną detekcję błędów transmisji. Szczególnie kod CRC32 wyróżniał się wysoką skutecznością w wykrywaniu błędów.

W przypadku korekcji błędów, kod Reed-Solomon (kod RS) okazał się efektywnym narzędziem, umożliwiającym naprawę różnych rodzajów błędów i poprawę integralności danych.

Podsumowując, wyniki tego projektu potwierdziły, że wprowadzenie systemu HARQ oraz odpowiednich kodów detekcyjnych i korekcyjnych przyczynia się do znaczącej poprawy jakości transmisji danych. Dalsze badania i optymalizacje tych mechanizmów mogą jeszcze bardziej zwiększyć niezawodność systemów komunikacyjnych w przyszłości.

8. Źródła

https://devopedia.org/hybrid-arq
https://en.wikipedia.org/wiki/Hybrid_automatic_repeat_request
https://www.youtube.com/watch?v=MN8KIdGYnic&ab_channel=FOREACH
https://www.youtube.com/watch?v=1pQJkt7-R4Q&ab_channel=vcubingx
https://www.youtube.com/watch?v=fBRMaEAFLE0&t=584s&ab_channel=Computerphile
https://www.youtube.com/watch?v=aSqGKNRJRDg&ab_channel=KhanAcademyPoPolsku
https://chat.openai.com/
https://users.ece.cmu.edu/~koopman/crc/index.html
https://pl.frwiki.wiki/wiki/Code_de_Reed-Solomon
https://phdopen.mimuw.edu.pl/zima12/guruswami-notes6.pdf