Jakub HYLA[1 2], Wojciech SUŁEK[1]

Silesian University of Technology (1), TKH Technology Poland Sp. z.o.o. (2)

doi:10.15199/48.2023.04.23

Dekoder LDPC implementowany w mikrokontrolerze dla systemów Internetu Rzeczy

Streszczenie. *Artykuł dotyczy projektowania systemów kodowania korekcyjnego dla protokołów komunikacyjnych w Internecie Rzeczy, które są implementowane na platformach o mocno ograniczonych zasobach obliczeniowych. W artykule zaproponowano wykorzystanie nowoczesnych kodów LDPC (Low Density Parity Check), zaprezentowano algorytm dekodujący oraz przedstawiono wyniki eksperymentalne implementacji w układzie mikrokontrolera. Przeprowadzono testy dla różnych wielkości słów kodowych oraz zebrano wyniki związane z czasem dekodowania, przepustowością, jak również liczbą iteracji potrzebną do zdekodowania jednego bloku.*

Abstract. *The article concerns the design of correction coding systems for communication protocols in the Internet of Things, which are implemented on platforms with very limited computing resources. The article proposes the use of modern LDPC (Low Density Parity Check) codes, presents the decoding algorithm and presents the experimental results of the implementation in a microcontroller device. Experiments were performed for different codeword sizes and the results were collected concerning the decoding time, throughput as well as the number of iterations needed to decode one block.* ***(LDPC decoder for Internet of Things systems)***

Słowa kluczowe: IoT, LDPC, QC-LDPC, dekodowanie
Keywords: IoT, LDPC, QC-LDPC, decoding

Wprowadzenie

Współczesny rozwój technologiczny determinuje wzrost znaczenia poprawnego przesyłania danych w wielu rozmaitych systemach teleinformatycznych. Wymagana jest transmisja coraz większej liczby informacji, w coraz krótszym czasie, przy wykorzystaniu jak najmniejszych porcji energii, co skutkuje nieuniknionym zjawiskiem błędów w transmisji. Tymczasem poprawność danych przesyłanych jest jednym z kluczowych elementów w wielu sektorach. Dzięki kodowaniu kanałowemu możliwe jest wykrywanie błędów, a także automatyczna rekonstrukcja prawidłowych bloków danych (korekcja). Jest to realizowane poprzez dołączanie do bitowych bloków informacyjnych pewnej liczby symboli (bitów) nadmiarowych (kontrolnych), które mogą być wykorzystane w odbiorniku do detekcji i korekcji błędów.

Kody LDPC (Low Density Parity Check) są obecnie jednymi z najlepszych metod kodowania blokowego o bardzo dobrych własnościach korekcyjnych. Kody LDPC dzięki swoim własnościom znajdują się blisko granicy Shannona [11]. Historia kodów LDPC sięga roku 1962, gdzie zostały zaproponowane przez R. G. Gallagera [5]. W ówczesnych czasach poziom technologiczny nie pozwalał na ich zastosowanie. Tym samym straciły one na znaczeniu aż do roku 1999, gdzie D.J.C. MacKay zwrócił uwagę na znakomite własności kodów LDPC [10]. Od tego czasu zdecydowanie zyskały one na popularności, czego przykładami są zastosowania takie jak: DVB-S2 i DVB-T2 [4], WiFi [1], WiMAX [7], sieci 5G [13, 2, 12, 15].

Projekt badawczy, którego częścią są wyniki przedstawione w niniejszym artykule, ma na celu pokazanie, że kody LDPC mogą być z powodzeniem stosowane także w systemach o silnie ograniczonych zasobach sprzętowych, którymi zwykle cechują się urządzenia systemów Internetu rzeczy (ang. IoT). W ramach projektu powstała efektywna implementacja kodera LDPC w układzie mikrokontrolera, która została opisana w [6], a także implementacja dekodera, oparta na koncepcji opisanej w niniejszym artykule. Artykuł dotyczący kodera opisuje problem złożoności obliczeniowej kodera LDPC oraz porusza problem związany z przetwarzaniem dużych liczby danych przez urządzenia o niewielkich możliwościach obliczeniowych. Jednocześnie autorzy proponują zmianę podejścia do zapisu tak zwanych macierzy rzadkich, czyli takich, w których występuje niewiele elementów niezerowych na poczet wykorzystania innych metod zapisu tych samych informacji. Analiza i rozwiązanie tego problemu znalazło zastosowanie w urządzeniu typu Internetu Rzeczy bazującym na mikrokontrolerze o ograniczonych możliwościach obliczeniowych. Wyniki liczbowe zajętości pamięci, czasu potrzebnego na zakodowanie bloku informacyjnego, jak również przepustowość zostały podsumowane w wynikach cytowanego artykułu. Należy zwrócić uwagę, że zyski związane z efektywną implementacją algorytmu kodowania są według autorów bardzo dobre i wpisują się w trend urządzeń typu Internetu Rzeczy zasilanych bateryjne ze względu na zyski związana z energooszczędnością.

Kody LDPC - podstawowe definicje

W systemie systematycznego kodowania blokowego, określony jest wektor informacyjny $\mathbf{u} = [u_1, u_2, ...u_K]$ będący częścią wektora kodowego $\mathbf{c} = [c_1, c_2, ...c_N]$, gdzie $N = K + M$, a M oznacza liczbę bitów nadmiarowych. Taki kod blokowy jest oznaczany $C(N, K)$, a kolejne słowa strumienia transmitowanej informacji – wektory informacyjne $\mathbf{u}$ są przekształcane w zakodowane wektory $\mathbf{c}$. Elementy wektorów $\mathbf{u}$, $\mathbf{c}$ mogą w ogólności należeć do ciała Galois $GF(q)$, jednakże powszechnie jest stosowane kodowanie binarne nad $GF(2)$, gdzie $u_k, c_n \in \{0, 1\}$.

Pojęcie sprawności kodu $R = K/N$ definiuje poziom nadmiarowości kodowania. Sprawność zawsze zawiera się w przedziale $0 < R < 1$. Zwiększenie liczby elementów nadmiarowych poprawia własności korekcyjne kodu, jednocześnie pogarszając sprawność kodu.

Dany kod LDPC definiowany jest przez macierz kontroli parzystości $\mathbf{H}$, która jest macierzą rzadką, natomiast $\mathbf{Hc}^T = 0$ to równanie kontrolne, które pozwala na weryfikację czy dany wektor $\mathbf{c}$ jest prawidłowy. Jeśli nie, to iteracyjny algorytm dekodowania LDPC pozwala na korekcję (pewnej liczby) błędów.

W przypadku kodów LDPC macierz kontrolna $\mathbf{H}$ jest macierzą rzadką – o małej liczbie elementów niezerowych. Kody LDPC dzielą się na dwa typy – kody regularne i nieregularne. Odróżnia je stała bądź też zmienna w przypadku kodów nieregularnych liczba elementów niezerowych w każdej kolumnie $\mathbf{H}$. W przypadku kodów regularnych jako zaletę można wskazać uproszczenie struktury sprzętowego kodera, natomiast w przypadku kodów nieregularnych o odpowiednio dobranej macierzy $\mathbf{H}$, zaletą w ogólności są lepsze własności korekcyjne od kodów regularnych [14]. W większości standardów telekomunikacyjncych wykorzysty-

wane są kody nieregularne.

Algorytmy dekodowania wykorzystywane w praktyce są algorytmami iteracyjnymi. Alternatywnym sposobem reprezentacji kodu dla macierzy $\mathbf{H}$, na potrzeby prezentacji algorytmu dekodowania, jest graf Tannera [10]. Graf Tannera dla liniowego kodu blokowego o macierzy kontrolnej $\mathbf{H}$ jest grafem dwudzielnym. Składa się on z wierzchołków kontrolnych i bitowych. W procesie dekodowania elementarne obliczenia reprezentowane są przez wierzchołki, natomiast sposób przesyłania wyników tych obliczeń reprezentowany jest poprzez krawędzie. Wierzchołki kontrolne odpowiadają wierszom macierzy $\mathbf{H}$ – czyli równaniom kontrolnym. Wierzchołki bitowe odpowiadają kolumnom macierzy $\mathbf{H}$ – czyli bitom słowa kodowego. Wartości prawdopodobieństw są reprezentowane przez wiadomości – poziomy wiarygodności (ang. beliefs), że dany bit jest równy 0 lub 1 (wiadomości z wierzchołków bitowych do kontrolnych) oraz informacje o spełnieniu równania kontrolnego, przy określonej wartości bitu (wiadomości z wierzchołków kontrolnych do bitowych).

Algorytmy iteracyjne polegają na zbliżaniu się do prawidłowego rozwiązania z każdą iteracją dekodowania. Inaczej nazywane są algorytmami propagacji poziomów wiarygodności BP (ang. Belief Propagation) i szczegółowo opisane w licznej literaturze [8], [10].

Model systemu IoT

Obecnie urządzenia systemów tzw. Internetu rzeczy są coraz bardziej popularne i zazwyczaj są postaci małego urządzenia zasilanego bateryjnie, z bardzo ograniczonymi funkcjami, lecz czasem także znacznie bardziej rozbudowanego urządzenia posiadającego system operacyjny, dającymi możliwość bezpośredniej wymiany danych z chmurą [9]. W niniejszym artykule uwagę skupiamy a urządzeniach bazujących na prostych mikrokontrolerach (np. STM32L4), o ograniczonych zasobach obliczeniowych. Dla wspomnianego mikrokontrolera producent (STM) daje do wykorzystania narzędzie wspomagające programowanie, które wykorzystano w opisanych pracach badawczych oraz biblioteki do obsługi peryferiów.

Model przykładowego rozwiązania komunikacji w systemie IoT jest przedstawiony na Rys. 1. Urządzenie końcowe Internetu Rzeczy wysyła / odbiera komunikaty, np. w aplikacji chmurowej, jako węzeł pośredni wykorzystując urządzenie centralne – bramkę IoT. Poziom oznaczony na czerwono, związany z urządzeniem IoT, jest poziomem o ograniczonych zdolnościach obliczeniowych, reprezentującym zwykle niskokosztowe i energooszczędne urządzenia końcowe.

Urządzenie IoT wysyła komunikaty w kierunku tzw. w górę (ang. uplink) oraz odbiera w kierunku tzw. w dół (ang. downlink). W przypadku komunikacji uplink urządzenie IoT gromadzi dane, przetwarza je, koduje oraz zamienia na odpowiedni sygnał dostosowany do kanału komunikacyjnego. Dane kodowane są za pomocą kodera LDPC, które zostało zaprojektowane dla urządzeń typu IoT, wyposażonych w mikrokontroler [6]. Komunikacja w kierunku downlink realizowana jest od bramki do urządzenia IoT, więc urządzenie odbiera sygnał, przetwarza go do postaci bitów lub tzw. miękkich decyzji (demodulator), a następnie wykorzystuje programowy dekoder do korekcji błędów.

Realizacja algorytmów kodeka LDPC w urządzeniach IoT

Algorytmy kodowania i dekodowania mają wymagania co do pojemności pamięci, jak również mocy obliczeniowej. Algorytm kodowania zrealizowany dla mikrokontrolera w urządzeniu typu IoT został opisany w artykule [6]. Na potrzeby urządzeń o ograniczonych zasobach pamięci i mocy obliczeniowej został zaproponowany algorytm kodujący oparty o indeksy niezerowych elementów macierzy $\mathbf{H}$. Możliwe jest również zrealizowanie algorytmu kodowania w postaci programowej wymagającego mniejszej liczby pamięci mikrokontrolera. Taki algorytm może realizować dekodowanie na potrzeby kodów LDPC jak i QC-LDPC (ang. Quasi Cyclic LDPC), jak przedstawiono w [6]

Kolejnym krokiem jest opracowanie realizacji programowej dekodera kodów LDPC dla układu mikrokontrolera. Do implementacji wybrano wersję algorytmu BP (ang Be-

Algorithm 1: Dekodowanie LDPC [16] dla układu mikrokontrolera.

Input:
– Macierz kontroli parzystości $\mathbf{H}$ zapisana w postaci indeksów niezerowych elementów.
– Maksymalna liczba iteracji dekodowania.
Output:
– Informacja o powodzeniu lub niepowodzeniu dekodowania.
– Wektor zdekodowanych danych.

1 Inicjalizacja - Przypisanie wartości wejściowych. Są one prawdopodobieństwami LLR (ang. Log-Likelyhood Ratio) (dla wszystkich $m \in M(n)$ i $n \in (1, M)$)

$$Q_{nm} := L(c_n|y_n) = \ln\left(\frac{P(c_n = 0|y_n)}{P(c_n = 1|y_n)}\right) \quad (1)$$

2 Krok horyzontalny, w którym wyznaczane są wierzchołki kontrolne. Wyznaczanie wartości minimalnych (dla wszystkich $n \in N(m)$ i $m \in (1, N)$):

$$R_{mn} := \left[\prod_{k \in N(m)\setminus n} \operatorname{sgn}(Q_{km})\right] \min_{k \in N(m)\setminus n} |Q_{km}| \quad (2)$$

Algorytm dekodujący wykorzystany w implementacji to algorytm Normalized Min-Sum

3 Krok wertykalny, w którym wyznaczane są wierzchołki bitowe. Sumowanie wartości minimalnych i wejściowych LLR (dla wszystkich $m \in M(n)$ i $n \in (1, M)$):

$$Q_{nm} := L(c_n|y_n) + \sum_{k \in M(n)\setminus m} R_{kn} \quad (3)$$

4 Dekodowanie wstępne Wyznaczanie prawdopodobieństw pseudo-posteriori (dla wszystkich n):

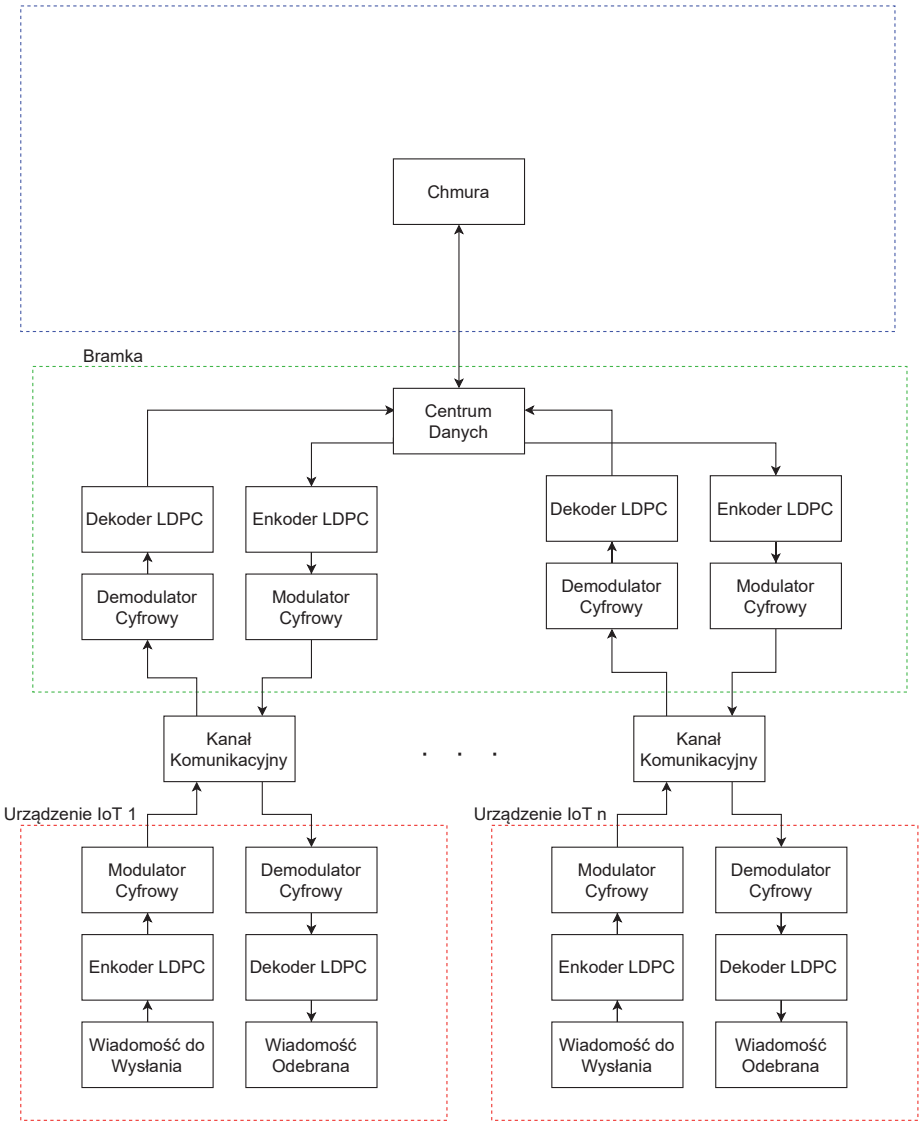
$$Q_n := L(c_n|y_n) + \sum_{k \in M(n)} R_{kn} \quad (4)$$

5 Wyznaczenie twardych decyzji.Podejmowanie próbnych, twardych decyzji (dla wszystkich $n \in (1, M)$):

$$\hat{c}_n := \begin{cases} 1 & \text{gdy } Q_n < 0 \\ 0 & \text{gdy } Q_n \geq 0 \end{cases} \quad (5)$$

6 Warunek zakończenia

$$\mathbf{H}\hat{c}^T = 0 \quad (6)$$



Rys. 1. Uproszczony model komunikacji z wykorzystaniem kodowania korekcyjnego LDPC w systemie IoT o topologii gwiazdy

lief Propagation) znaną jako Normalized Min-Sum [3], która jest przedstawiona jako algorytm 1. Przedstawione kroki powtarzane są iteracyjnie tak długo, aż zostanie osiągnięty warunek zakończenia (6) lub zostanie osiągnięta maksymalna liczba iteracji (co oznacza dekodowanie zakończone niepowodzeniem). Schemat blokowy algorytmu pokazany jest na Rys. 2. Należy ona do klady iteracyjnych algorytmów propagacji wartości wiarygodności BP.

W opisie algorytmu 1 wykorzystano następujące oznaczenia:

- $\mathbf{x} = [c_1, c_2, \ldots, c_n]$ jest wektorem kodowym,
- $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ jest wektorem wartości odebranych (miękkie decyzje),
- $L(c_n|y_n)$ - LLR prawdopodobieństw *a priori* dla bitu n-tego,
- $N(m)$ - numery kolumn w macierzy kontrolnej $\mathbf{H}$ zawierających jedynkę w wierszu m-tym,
- $M(n)$ - numery wierszy w macierzy kontrolnej $\mathbf{H}$ zawierających jedynkę w kolumnie n-tej,
- Q_{nm} - wartość LLR (*Log-Likelihood Ratio*) wiarygodności z n-tego wierzchołka bitowego do m-tego wierzchołka kontrolnego grafu Tannera,
- R_{mn} - wiadomość z m-tego wierzchołka kontrolnego do n-tego wierzchołka bitowego,
- $\hat{c}_n$ - wektor zdekodowany.

Implementacja programowa algorytmu dokowania na urządzeniu IoT

Zaproponowany algorytm został zaimplementowany z wykorzystaniem języka C, przy czym opis utworzono optymalizując dla kompilacji do układu mikrokontrolera. Znajduje on zastosowanie w dekodowaniu danych dla macierzy kontroli parzystości $\mathbf{H}$ regularnych oraz nieregularnych kodów LDPC, w tym klasy QC-LDPC. Jest to zatem uniwersalne rozwiązanie, które wymaga jedynie odpowiedniego zapisu macierzy $\mathbf{H}$, z wykorzystaniem indeksów elementów niezerowych tej macierzy.

Pierwszym krokiem algorytmu dekodowania jest przekazanie danych do procedury inicjalizującej zmienne oraz bufory do przechowywania danych tymczasowych. Przy wywołaniu funkcji dekodowania trafią one na stos w obszarze pamięci RAM. Pamięć RAM mikrokontrolera zwykle ograniczona jest do kilkudziesięciu kilobajtów w zależności od typu mikrokotrolera. W tym kroku przypisywane są wartości wejściowe (LLR prawdopodobieństw skojarzonych z poszczególnymi bitami), dla wszystkich węzłów bitowych. Na tym etapie wywoływana jest również funkcja odpowiadająca za podjęcie tzw. twardych decyzji, czyli określenie na podstawie odebranych symboli najbardziej prawdopodobnych wartości poszczególnych bitów (0 / 1). Tworzona i inicjalizowana jest macierz (dwuwymiarowa

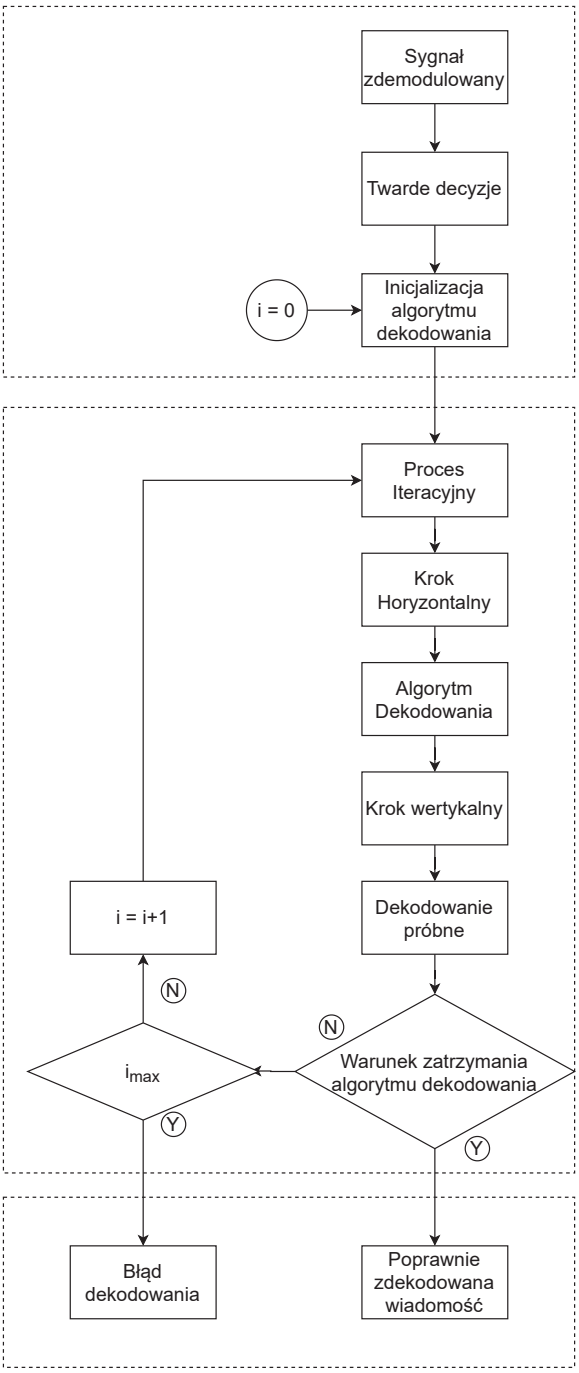
Macierz	K	N	P	R	Rodzaj
H1	512	1024	32	$\frac{1}{2}$	Regularna
H2	512	1024	32	$\frac{1}{2}$	Nieregularna
H3	256	512	32	$\frac{1}{2}$	Regularna
H4	256	512	32	$\frac{1}{2}$	Nieregularna
H5	320	512	32	$\frac{5}{8}$	Regularna
H6	384	512	32	$\frac{3}{4}$	Regularna
H7	144	288	16	$\frac{1}{2}$	Regularna
H8	144	288	16	$\frac{1}{2}$	Nieregularna

Tablica 1. Parametry eksperymentalnych macierzy kontroli parzystości $\mathbf{H}$ dla kodowania LDPC.

tablica), która przechowuje wiadomości z wierzchołków kontrolnych do bitowych (macierz $\mathbf{rllr}$ oraz macierz (tablica) przechowująca wiadomości z wierzchołków bitowych do kontrolnych (macierz $\mathbf{qllr}$).

Następnie rozpoczynana jest pętla o ograniczonej liczbie iteracji, w której realizowany jest algorytm dekodowania. Ostatnim krokiem pętli jest zawsze sprawdzanie warunku zakończenia (spełnienia wszystkich równań kontrolnych) i w zależności od rezultatu możliwe jest przerwanie obliczeń.

Drugim krokiem jest krok horyzontalny, w którym obliczane są wartości wierzchołków kontrolnych oraz wykonywana jest realizacja algorytmu Normalized Min-Sum za pomocą którego dane są dekodowane. Dla każdego wiersza macierzy kontrolnej zapisywane są w pamięci indeksy elementów niezerowych macierzy kontrolnej $\mathbf{H}$. Patrząc na graf Tannera zapisywane są w pamięci indeksy wierzchołków bitowych połączonych krawędziami z wierzchołkami kontrolnymi. Następnie zapisywane w pamięci są wartości bezwzględne twardych decyzji oraz ich znak w osobnych tablicach pomocniczych. Algorytm iteruje po wszystkich wierszach macierzy (H), gdzie dla każdego wiersza obliczana jest wartość minimalna z elementów o indeksach innych niż aktualny, natomiast bazując na znakach obliczany jest iloczyn znaków wektora wejściowego bez brania pod uwagę elementu o obliczanym indeksie lub innymi słowy jest to XOR bitów znaku. Obliczona wartość minimalna jest normalizowana poprzez przemnożenie ich przez wartość w zakresie 0.5-1. Na potrzeby demonstracyjne przyjęto doświadczalnie wartość 0.75. Po normalizacji wartości zaokrąglane są do liczb całkowitych. W trakcie pracy nad algorytmem zauważono, że wartość parametru mniejsza od 1 wpływa pozytywnie, gdy jest dużo błędów i algorytm działa na granicy swoich możliwości. W analogicznym przypadku algorytm dekodujący Min-Sum z parametrem 1 nie potrafił zdekodować danych, natomiast algorytm dekodujący Normalized Min-Sum z parametrem 0.75 dekodował je poprawnie. Końcowym etapem jest połączenie obliczonego znaku z wartością modułem. Tym samym wraz kolejnymi iteracjami aktualizowane są wartości macierzy $\mathbf{rllr}$.



Rys. 2. Parameters of the experimental parity-check matrices $\mathbf{H}$ for LDPC. The λ_d represents the fraction of degree-d variable nodes in the code graph.

Trzecim krokiem jest krok wertykalny, w którym obliczane są wierzchołki bitowe. Wykonywana jest ona zgodnie z Tab. 1 N razy, czyli tyle ile mamy kolumn macierzy $\mathbf{H}$. Początkowo wybieramy indeksy niezerowych elementów w pierwszej kolumnie macierzy $\mathbf{H}$. Bazując na macierzy $\mathbf{rllr}$ obliczanej w poprzednim kroku wyciągany jest jej odpowiedni fragment bazując na indeksach niezerowych elementów kolumn macierzy $\mathbf{H}$. Następnie wybrany fragment jest sumowany bez brania pod uwagę elementu o obliczanym indeksie, do niego dodawana jest wartość inicjująca algorytm. Obliczony wektor tymczasowy jest ograniczany do co

wartości po to, aby wartości nie dążyły do +/-∞, a w przypadku obliczeń komputerowych przekroczenia zakresu zmiennej.

Bazując na wyznaczonych wartościach macierzy $\mathbf{qllr}$ oraz $\mathbf{rllr}$ realizowane jest wstępne dekodowanie bazujące na wartości tymczasowej z poprzedniego kroku algorytmu. Na jego podstawie wyznaczane są twarde decyzje bazując na znaku. Końcowym etapem jest obliczenie równania kontrolnego oraz sprawdzenie warunku zakończenia. Jeżeli równanie kontrolne jest spełnione oznacza to, że zostało zakończone dekodowanie tego słowa.

W przypadku, równanie kontrolne nie zostanie spełnione oraz zostanie osiągnięta maksymalna liczba iteracji algorytm przerwie działanie i zwróci błąd. Wtedy jest wiadome, że słowo jest niepoprawne i obarczone błędami.

Wyniki liczbowe działania algorytmu dokowania na urządzeniu IoT

Implementacja dekodera została zrealizowana z wykorzystaniem mikrokontrolera firmy ST o oznaczeniu STM32L476, należącym do serii o bardzo niskim poborze mocy. Mikrokontroler jest oparty o wydajny 32-bitowy rdzeń Arm Cortex M4, pracujący z częstotliwością do 80MHz i posiada wbudowane 1MB pamięci Flash oraz 128KB pamięci SRAM. Dzięki swoim możliwościom obliczeniowym oraz trybom niskiego poboru energii, układ jest dobrym rozwiązaniem dla urządzeń Internetu Rzeczy, stosowanym w wielu przemysłowych rozwiązaniach.

Z wykorzystaniem zaprezentowanego mikrokontrolera przeprowadzono analizy dla różnych macierzy $\mathbf{H}$ w dwóch wariantach:

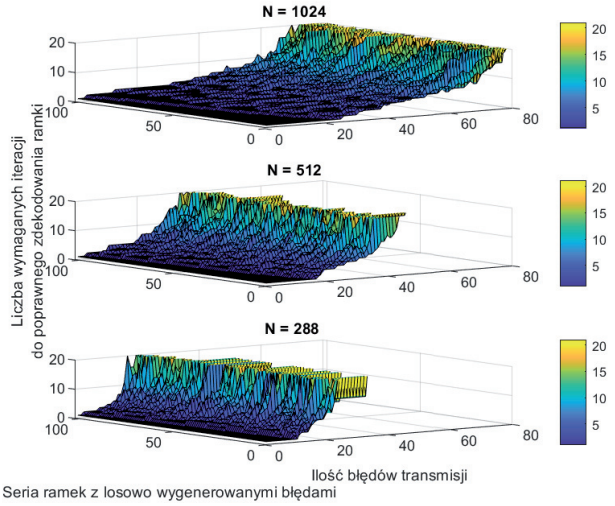
- stała sprawność kodu o różne długości słowa
- stała długość słowa o różnej sprawności

Stała sprawność

Porównano, jak zachowuje się algorytm dekodowania w zależności od liczby błędnych bitów w odebranym słowie kodowym. Błędy były wprowadzane w sposób pseudo-losowy, w liczbie od 0 do 80 bitów. Dla każdej macierzy $\mathbf{H}$ wykonano 100 prób. Na rys. 3 przedstawiono porównanie wyników dekodowania dla sprawności $\frac{1}{2}$, dla różnych długości słowa kodowego N, przy maksymalnej liczbie iteracji dekodowania ustalonej na 20. Przedstawiono liczbę iteracji potrzebnych do zdekodowania poszczególnych słów kodowych. Osiągnięcie limitu 20 iteracji wskazuje bloki, które nie zostały prawidłowo zdekodowane. Do uzyskania tych wyników eksperymentalnych wykorzystano macierze $\mathbf{H2}$, $\mathbf{H4}$, $\mathbf{H8}$, wymienione w tab. 1. Zauważyć można, jak długość bloku kodowego, a tym samym liczba bitów nadmiarowych wpływa na możliwości dekodera. Najdłuższy blok z 512 bitami nadmiarowymi potrafi skorygować do ok. 80 błędów, blok z 256 bitami nadmiarowymi potrafi skorygować do około 40 błędów, natomiast blok z 144 bitami nadmiarowymi potrafi skorygować do około 30 błędów.

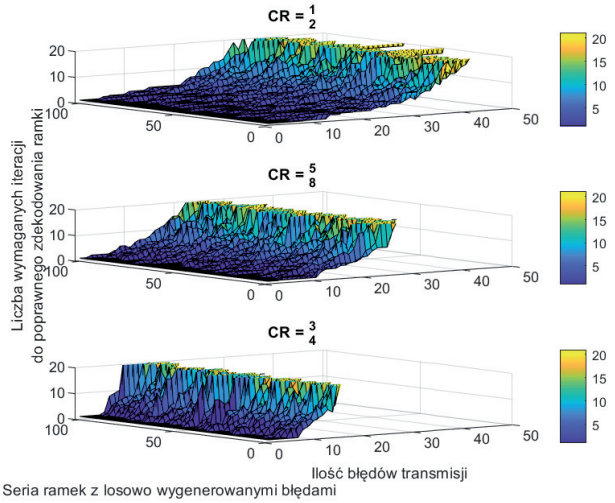
Stała długość słowa

Porównano, jak zachowuje się algorytm dekodowania w zależności od liczby błędów transmisji. Błędy dla każdego wektora testowego były wprowadzane w sposób pseudo-losowy, w liczbie od 0 do 50. Dla każdej macierzy $\mathbf{H}$ wykorzystano 100 wektorów testowych. Na wykresie 4 przedstawiono porównanie dla długości słowa $N = 512$ dla różnych sprawności kodu R. Do porównania wykorzystano macierze $\mathbf{H3}$, $\mathbf{H5}$, $\mathbf{H6}$ z tab. 1. Zauważalne jest jak liczba bitów nadmiarowych wpływa na możliwości dekodera.



Rys. 3. Wyniki procedury iteracyjnego algorytmu dekodowania LDPC dla stałej sprawności

Kod o sprawności $\frac{1}{2}$ potrafi osiągnąć najmniejszą liczba iteracji dla tej samej liczby błędów. Potrafi zdekodować nawet 50 błędów transmisji w zadanej liczbie iteracji. Kod o sprawności $\frac{5}{8}$ potrafi zdekodować do około 25 błędów transmisji. Kod o sprawności $\frac{3}{4}$ potrafi zdekodować do 15 błędów transmisji.



Rys. 4. Wyniki procedury iteracyjnego algorytmu dekodowania LDPC dla stałej długości słowa

Wyniki czasowe oraz przepustowość algorytmu dekodowania

Przepustowość dekodera (ang. throughput – TH) można określić z zależności (1), gdzie K jest długością słowa informacyjnego, a T_{exec} to czas działania dekodera. Czas dekodowania można rozłożyć na 3 elementy: czas inicjalizacji T_{init}, czas dekodowania T_{decode}, czas generowania informacji wyjściowej T_{out}, jak w (2). Według testów proces inicjalizacji może zostać wykonany raz na początku, a czas generowania informacji wyjściowej jest znikomo mały w porównaniu do czasu dekodowania. Zatem $T_{exec} \approx T_{decode}$. Czas dekodowania obejmuje czas wykonywania instrukcji oraz sumaryczny czas dostępów do pamięci, przy czym oba czasy są proporcjonalne do liczby niezerowych elementów w macierzy kontrolnej. Przedstawiono to za pomocą zależności (4), gdzie C określa średnią liczbę elementów niezerowych w każdym wierszu macierzy $\mathbf{H}$, natomiast I to maksymalna

liczba iteracji. Wreszcie przepustowość można wyznaczyć równaniem (6), gdzie α jest określonym eksperymentalnie współczynnikiem proporcji złożoności obliczeń i opóźnienia dostępu do pamięci. Wynika z tego, że przepustowość zależna jest od współczynnika α, średniej liczby niezerowych elementów w każdym wierszu macierzy $\mathbf{H}$, liczby iteracji oraz sprawności kodu. Współczynnik α został wyznaczony doświadczalnie na podstawie czasu dekodowania jednej iteracji i dla każdej z macierzy $\mathbf{H}$ jest zbliżony. Średnia wartość współczynnika α dla eksperymentów przeprowadzonych na wybranych macierzach wyniosła $1,83 * 10^{-6}$. Korzystając z tej wartości współczynnika α, błąd określenia przepustowości z zależności (6) nie przekracza 5% dla wykorzystanych w pracach badawczych macierzy kontrolnych.

$$TH = \frac{N}{T_{exec}} \tag{1}$$

$$T_{exec} = T_{init} + T_{decode} + T_{out} \tag{2}$$

$$T_{decode} = T_{operate} + T_{memory} \tag{3}$$

$$T_{decode} \approx \propto \times N \times C \times I \tag{4}$$

$$TH = \frac{N}{\alpha \times C \times I \times N} = \frac{1}{\alpha \times C \times I} \tag{5}$$

Typ macierzy H	Czas dekodowania jednej iteracji [ms]	Przepustowość [bit/s]
H1	230	4452
H2	245	4180
H3	115	4452
H4	118	4339
H5	150	3012
H6	185	2116
H7	65	4431
H8	65	4431

Tablica 2. Wyniki czasowe oraz przepustowość algorytmu dekodowania

Typ macierzy H	Długość słowa kodowego N	Średnia liczba elementów niezerowych w każdym wierszu C	Współczynnik złożoności obliczeniowej α
H1	1024	6	1,87E-06
H2	1024	6,5	1,84E-06
H3	512	6	1,87E-06
H4	512	6,5	1,77E-06
H5	512	9	1,84E-06
H6	512	13	1,82E-06
H7	288	6	1,88E-06
H8	288	6,5	1,74E-06

Tablica 3. Parametry potrzebne do wyznaczenia przepustowości algorytmu dekodowania

Istotnym jest, że przepustowość nie jest zależna od wielkości słowa kodowego, a jedynie od liczby elementów niezerowych w kolumnie, co dla algorytmu bazującego na indeksach elementów niezerowych jest poprawne, a jedynie czas dekodowania jest zależny od wielkości słowa kodowego. Dla kodów nieregularnych zastosowano wartość średnią obliczoną na podstawie liczby jedynek w każdym wierszu.

Wnioski

W artykule przedstawiono implementację nowoczesnego kodowania korekcyjnego LDPC w układzie mikrokontrolera, które może być wykorzystane w protokołach komunikacyjnych przetwarzających niewiele danych, przy ograniczonych zasobach obliczeniowych, co jest charakterystyczne dla urządzeń typu IoT. W szczególności skupiono się na układzie dekodera. Z przeprowadzonych eksperymentów można wyciągnąć pewne praktyczne wnioski projektowe. W szczególności zauważono, że kody nieregularne mają lepsze własności korekcyjne. Dla tych samych parametrów macierzy $\mathbf{H}$ w przypadku macierzy regularnej oraz nieregularnej, różnica jest znaczna i kształtuje się na poziomie około od 50% do nawet 90% większej liczby zdekodowanych błędów w tej samej liczby iteracji. Pokazano, że możliwa jest implementacja dekodera kodów LDPC oraz QC-LDPC, także nieregularnych, w układzie mikrokontrlera, mimo znacznych wymagań dotyczących mocy obliczeniowej dla kodów LDPC. Wskazano, jaka przepustowość może być uzyskana oraz zaproponowano przybliżony wzór obliczania przepustowości dekodera zaimplementowanego w typowym mikrokontrolerze.

***Autorzy**: Ph.D. Wojciech Sułek, Jakub Hyla Institute of Automatic Control and Robotics, Electrinics and Telefomunication, Silesian University of Technology ul. Akademicka 2a, 44-100 Gliwice, Poland, email: jakubhyla93@gmail.com,*

This research was co-financed by the Ministry of Education and Science of Poland under grant No. DWD/3/7/2019.

LITERATURA

[1] Ieee 802.11-2016. ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2016.

[2] Salima Belhadj and Moulay Lakhdar Abdelmounaim. On error correction performance of ldpc and polar codes for the 5g machine type communications. In *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–4, 2021.

[3] Xiaoheng Chen and Chang-Li Wang. High-Throughput Efficient Non-Binary LDPC Decoder Based on the Simplified Min-Sum Algorithm. *IEEE Trans. Circuits Syst. I*, 59:2784–2794, November 2012.

[4] ETSI Standard: EN 302 307 v1.1.1, *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications*, 2005.

[5] Robert G. Gallager. Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, IT-8:21–28, January 1962.

[6] Jakub Hyla, Wojciech Sułek, Weronika Izydorczyk, Leszek Dziczkowski, and Wojciech Filipowski. Efficient ldpc encoder design for iot-type devices. *Applied Sciences*, 12(5), 2022.

[7] IEEE Standard: IEEE P802.11n=D10. Draft IEEE Standard for Local Metropolitan Networks – Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC), and Physical Layer (PHY) specifications: Enhancements for Higher Throughput, March 2006.

[8] Shu Lin and Daniel J. Costello, Jr. *Error Control Coding: Fundamentals and Applications, 2nd Edition*. Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458, 2004.

[9] Bing Liu, Rongke Liu, Zhanxian Liu, and Ling Zhao. An efficient implementation of ldpc decoders on arm processors. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 1–5, 2018.

[10] David J. C. MacKay. Good Error-Correcting Codes Based on Very Sparse Matrices. *IEEE Trans. Inf. Theory*, 45:399–431, March 1999.

[11] David J. C. MacKay and Radford M. Neal. Near Shannon Limit Performance of Low Density Parity Check Codes. *Electronics Letters*, 32:1645–1646, August 1996.

[12] Jérémy Nadal and Amer Baghdadi. Parallel and flexible 5g ldpc decoder architecture targeting fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(6):1141–1151, 2021.
[13] Vladimir L. Petrović, Dragomir M. El Mezeni, and Andreja Radošević. Flexible 5g new radio ldpc encoder optimized for high hardware usage efficiency. *Electronics*, 10(9), 2021.
[14] Thomas J. Richardson, M. Amin Shokrollahi, and Rüdiger L. Urbanke. Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes. *IEEE Trans. Inf. Theory*, 47:619–637, February 2001.
[15] Chance Tarver, Matthew Tonnemacher, Hao Chen, Jianzhong Zhang, and Joseph R. Cavallaro. Gpu-based, ldpc decoding for 5g and beyond. *IEEE Open Journal of Circuits and Systems*, 2:278–290, 2021.
[16] Saleh Usman and Mohammad M. Mansour. Fast column message-passing decoding of low-density parity-check codes. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(7):2389–2393, 2021.