

Министерство науки и высшего образования РФ  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра телекоммуникаций и основ радиотехники (ТОР)

**Исследование технологий NarrowBand Internet of Things для  
устройств телеметрии с низкими объемами обмена данными**

Отчет по производственной практике: Научно-исследовательская работа

Выполнил  
студент гр.168-М:  
\_\_\_\_\_ Горелкин. Б.К.  
« » \_\_\_\_\_ 2019 г.

Руководитель  
Доцент каф. ТОР:  
\_\_\_\_\_ Рогожников Е.В.  
« » \_\_\_\_\_ 2019 г.

Проверил:  
Доцент кафедры ТОР  
\_\_\_\_\_ Литвинов Р.В  
«\_\_» \_\_\_\_\_ 2019 г.

Томск 2019

## Оглавление

Введение.....	3
1. Реализация передатчика при моделирование канала N-BCCH в DLink.....	4
1.1 Моделирование канала N-BCCH в DLink.....	4
1.2 Реализация CRC.....	5
1.3 Реализация свёрточного кодера.....	6
1.4 Реализация Puncturing.....	8
1.5 Реализация перемежителя.....	8
1.6 Реализация модулятора.....	9
2. Канал связи.....	11
3. Реализация приёмника при моделирование канала N-BCCH в DLink.....	12
3.1 Демодулятор QPSK.....	12
3.2 Реализация деперемежителя.....	13
3.3 Реализация восстановления проколотых бит.....	13
3.4 Реализация декодирования по алгоритму Витерби.....	14
Заключение.....	15
Список использованных источников.....	16

## Введение

С каждым днем устройств с возможностью выхода в интернет становится все больше и больше, а следовательно, по закону Меткалфа (полезность любой системы равняется квадрату элементов этой системы) «интернет вещей» с каждым днем становится перспективнее. Интернет вещей не просто связывает миллиарды устройств в одну сеть, как когда-то Интернет объединил все компьютеры. Реальная инновация и потенциал Интернета вещей в том, чтобы трансформировать бизнес-модели, позволять компаниям продавать продукты, по-новому принося дополнительную пользу как компании, так и клиенту.

С увеличением количества «умных» устройств, которые выходят в сеть и взаимодействуют друг с другом – все меньше подходят привычные нам стандарты передачи данных (Wi-Fi, LTE и др.) по нескольким причинам. Во-первых большое количество устройств подключенных к одной базовой станции (например LTE) значительно ухудшат использование этой станции по её прямому назначению, да и использовать высокоскоростные каналы для обмена низкими объемами данных – не целесообразно, во-вторых далеко не каждое устройство можно подключить к сети электропитания, а значит питание будет производиться от батареи, которую нужно экономить. Для реализации межмашинного взаимодействия (M2M) необходимо рассмотреть технологии LPWAN (Low Power Wide Area Network) т.к. они позволяют объединить в одну систему очень большое количество устройств. Именно это и является целью данной работы – обзор технологий и существующих решений для системы «интернет вещей».

Для этого в данной работе рассматриваются теоретические сведения о системах для связи IoT, а также сравнение существующих решений.

## 1. Реализация передатчика при моделирование канала N-BCCH в DLink

### 1.1 Моделирование канала N-BCCH в DLink

Задача NB-IoT – возможность устройств работать в низких уровнях сигнала и при высоком уровне шумов, а также экономия питающей батареи. Данный стандарт был разработан на базе существующих стандартов мобильной связи. NB-IoT (от англ. NarrowBand IoT – «узкополосный интернет вещей»)

Для моделирования был выбран канал N-BCCH параметры которого представлены в таблице 1.1, а структурная схема на рисунке 1.1.

Таблица 1.1 – Параметры N-BCCH канала

N-BCCH Coding parameters	Size [bits]
Payload	152
CRC	16
Tail bits	8
Input bits for channel coding	176
Convolutional coding	1/3
Encoded Bits	528
Puncturing	80
Constraint Length	7

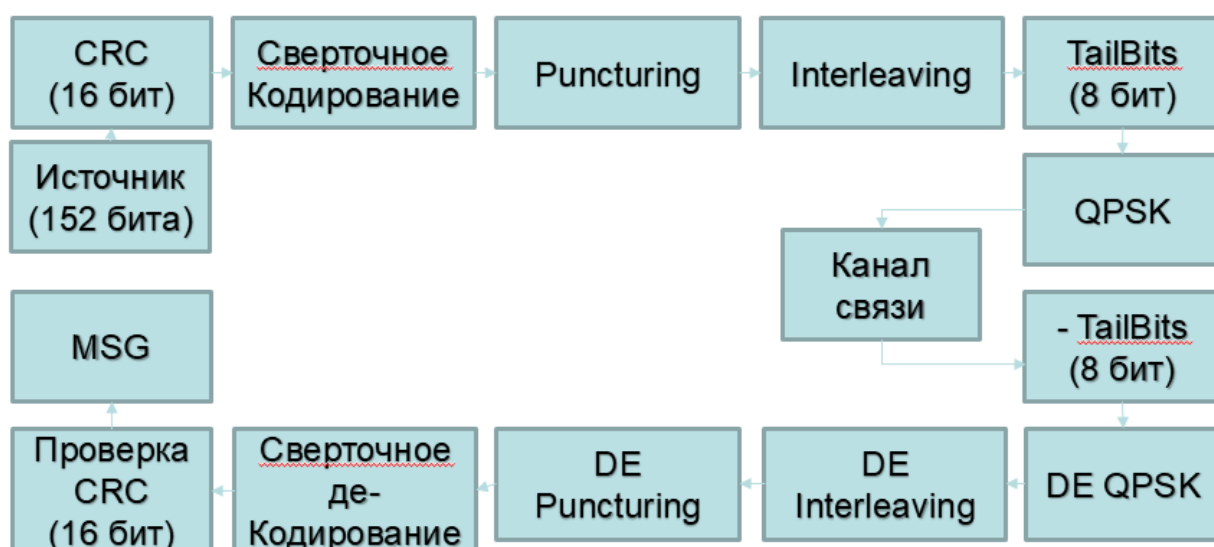
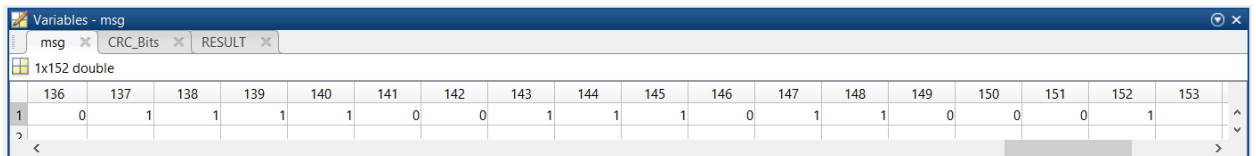


Рисунок 1.1 – Структурная схема канала N-BCCH

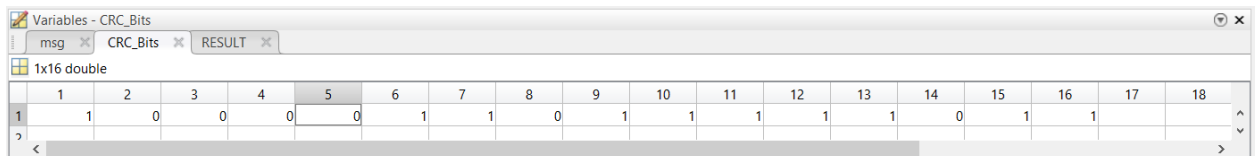
## 1.2 Реализация CRC

Основная идея алгоритма контрольной суммы состоит в представлении сообщения в виде двоичного числа, полученного путем деления его на фиксированное двоичное число и использовании остатка от этого деления в качестве контрольной суммы CRC. На рисунке 2.1 показано исходное сообщение, на рисунке 2.2 рассчитанная для данного сообщения контрольная сумма, на рисунке 2.3 сообщение вместе с контрольной суммой CRC.



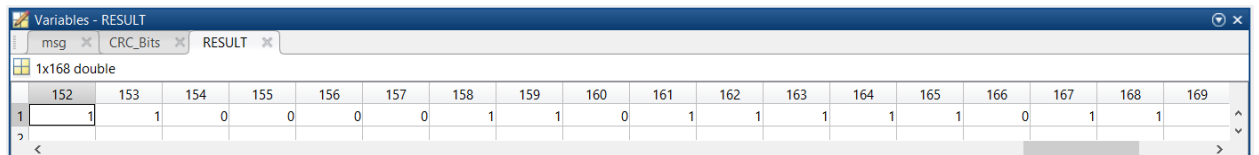
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
1	0	1	1	1	1	0	0	1	1	1	0	1	1	0	0	0	1

Рисунок 1.2 – Исходное сообщение



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	1	1	

Рисунок 1.3 – Рассчитанная CRC



152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169
1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	1	1

Рисунок 1.4 – Сообщение вместе с CRC

На рисунке 2.4 представлен код, реализующий расчет контрольной суммы по полиному указанному в рекомендации к стандарту 3GPP [1].

```

Payload = 152;
msg = randi([0 1], 1, Payload); %Длина массива (Полезная нагрузка)
DATA_CODEC = msg;
CRC_Type=16;
ENCODER=1; % 1 - кодер; 0 - декодер.
crc_poly = [1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1];          crc_len = 16;
crc_rem = zeros(1,crc_len+1);
if (ENCODER==1) tmp_array = [DATA_CODEC, zeros(1,crc_len)]; else tmp_array=DATA_CODEC; end;
% Вычисление CRC
for n=1:length(tmp_array)
    for m=1:crc_len crc_rem(m) = crc_rem(m+1); end;
    crc_rem(crc_len+1) = tmp_array(n);

    if(crc_rem(1) ~= 0)
        for(m=1:crc_len+1) crc_rem(m) = mod(crc_rem(m)+crc_poly(m), 2); end;
    end;
end;
CRC_Bits=crc_rem(2:end);
if (ENCODER==1) RESULT=[DATA_CODEC, CRC_Bits]; ERROR=0; else
if ((ENCODER==0) && (bi2de(CRC_Bits)~=0)) RESULT=DATA_CODEC(1:(length(tmp_array)-crc_len)); ERROR=1; else
if ((ENCODER==0) && (bi2de(CRC_Bits)==0)) RESULT=DATA_CODEC(1:(length(tmp_array)-crc_len)); ERROR=0;end;end;end;

```

Рисунок 1.5 – Код, реализующий расчет CRC

### 1.3 Реализация свёрточного кодера

К передаваемому сообщению, кроме контрольной суммы, добавляется защитный интервал длиной в 8 бит, который не несет в себе информации. Tail bits необходим для устранения «обрезания» передаваемого сообщения на сверточном кодере, в итоге на сверточный кодер приходят данные длиной в 176 бит (22 байта). По условию скорость кодера равняется 1/3, Кодирование производится по полиномам:

$$G4 = 1 + D2 + D3 + D5 + D6$$

$$G5 = 1 + D + D4 + D6$$

$$G6 = 1 + D + D2 + D3 + D4 + D6$$

В результате после процедуры сверточного кодирования получается блок длиной 528 бит (66 байт). Результат декодирования – последовательность, содержащая столько же символов, сколько и входной вектор. Скалярный входной параметр *tblen* – положительное целое число, задающее глубину просмотра решетки при декодировании. В случае если на вход канального кодера поступит последовательность без защитного интервала, тогда при декодировании «потеряется» часть данных на величину *tblen*

На рисунке 3.1 представлен входной поток данных на сверточный кодер, из рисунка видно, что начиная со 152го бита начинается рассчитанная CRC, а со 169го символа начинается защитный интервал, длина которого составляет 8бит. На рисунке 3.2 показан выходной поток данных после прохождения сверточного кодера. На рисунке 3.3 представлена реализация сверточного кодера.

	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Рисунок 1.6 – Входной поток данных на сверточном кодере

	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528
1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0

Рисунок 1.7 – Выходной поток данных после прохождения сверточного кодера

```
%% Кодирование массива
% _____Сверточне кодирование_____ %

trell = poly2trellis(7,[171 133 165]);
tblen = TailBits; % Глубина просмотра при декодировании
zero = zeros(1,tblen); %последовательность нулей, длиной tblen
BITunited = [BIT,zero]; %к последовательности BIT добавили массив длиной tblen
                        %для того чтобы при декодировании не обрезались биты
codeBIT = convenc(BITunited,trell); %Закодировали массив (сверточное кодирование)
```

Рисунок 1.8 – Реализация сверточного кодирования

## 1.4 Реализация Puncturing

После сверточного кодера данные подвергаются «прокалыванию» таким образом, что из последовательности длиной в 528 бит – 80 закодированных бит не передаются, т.е. каждый бит находящийся на  $J_{\text{puncturing}}=1:80$ ;  $C=23+5*J_{\text{puncturing}}$ ; Вырезается из последовательности и в результате получается последовательность из 448 символов

The screenshot shows the MATLAB Variables window with two variables: codeBIT and codeBIT\_punct. codeBIT is a 1x528 double array, and codeBIT\_punct is a 1x448 double array. The arrays contain binary data. The codeBIT array is displayed as a row of 528 elements, with the first 36 elements shown in the image. The codeBIT\_punct array is displayed as a row of 448 elements, with the first 36 elements shown in the image.

codeBIT	codeBIT_punct																
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	0

Рисунок 1.9 – Данные на входе «прокалывателя»

Variables - codeBIT\_punct

codeBIT codeBIT\_punct

1x448 double

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	1

Рисунок 1.10 – Данные на выходе прокалывателя

```
%% Puncturing
%Значения для выкалывания *Описание: Рабочий стол? диссертация? документ 45820-d10 ? страница 137
Jpuncturing=1:80;
C=23+5*Jpuncturing;
%Код прокололся таким образом, что следующие 80 закодированных битов не передаются.
codeBIT_punct = codeBIT;
%В результате получается блок из 448 кодированных и проколотых битов, P (0) ... P (447).
codeBIT_punct (C)=[];
```

Рисунок 1.11 – Реализация «прокалывателя»

## 1.5 Реализация перемежителя

Перемежитель (Интерливер от англ. Interleaver) — блок, реализующий перемежение - один из способов борьбы с ошибками. Предназначен для борьбы с пакетированием ошибок путём их разнесения во времени. Использует перемешивание (перемежение) символов передаваемой последовательности на передаче и восстановление её исходной структуры на приёме. Может использоваться как самостоятельно, так и вместе с помехоустойчивым кодом, являясь в таком случае его составным



компонентом. Благодаря перемежению на входе декодера ошибки равномерно распределяются во времени, в идеале образуя поток независимых ошибок. На рисунке 5.1 показаны входные данные на перемеживающем устройстве, на рисунке 5.2 показаны выходные данные после перемеживания. Перемеживание осуществляется по алгоритму указанному в стандарте 3GPP[2].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0

Рисунок 1.12 – входные данные на перемеживающем устройстве

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1	1	1

Рисунок 1.13 – выходные данные после перемеживания

```
%% Перемежитель %
zer16x28 = zeros(16,28);
for k = 1:448;
    B=mod(12*(k-1)+floor((k-1)/2)+mod((k-1),2),16);
    j=mod(23*mod(5*(k-1),28)+ floor(7*(k-1)/16),28);
    B=B+1;
    j=j+1;
    inter(B,j)= codeBIT_punct(k); %Матрица Вxj для перемеживания
end
Interleaving = reshape (inter', 1,448);
y = Interleaving;
```

Рисунок 1.14 – Реализация перемежителя

## 1.6 Реализация модулятора

Согласно стандарту в канале N-BCCH используется QPSK модуляция, созвездие которой представлено на рисунке 6.1, реализация модулятора продемонстрирована на рисунке 6.2, входной и выходной поток данных показан на рисунках 6.3 и 6.4 соответственно.

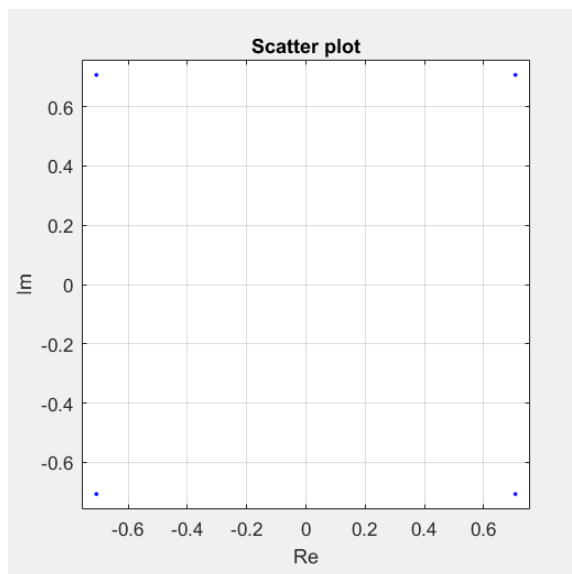


Рисунок 1.15 – QPSK модуляция

```

%% Модулятор
y = Interleaving;
for i = 1 : length(y) / 2    % Length - сам рассчитывает от длины y
    if y(2*(i-1)+1)==1 && y(2*i)==1
        outmod(i)=0.707+1i*0.707;
    end
    if y(2*(i-1)+1)==1 && y(2*i)==0
        outmod(i)=-0.707+1i*0.707;
    end
    if y(2*(i-1)+1)==0 && y(2*i)==0
        outmod(i)=-0.707-1i*0.707;
    end
    if y(2*(i-1)+1)==0 && y(2*i)==1
        outmod(i)=0.707-1i*0.707;
    end
end
end

```

Рисунок 1.16 – реализация QPSK модуляции

Variables - Interleaving

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1	1	1

Рисунок 1.17 – Данные на входе модулятора

Variables - outmod

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.7070 - 0.7070i	0.7070 - 0.7070i	0.7070 + 0.7070i	-0.7070 - 0.7070i	-0.7070 + 0.7070i	-0.7070 + 0.7070i	-0.7070 - 0.7070i	0.7070 + 0.7070i	0.7070 + 0.7070i	0.7070 - 0.7070i	0.7070 + 0.7070i	0.7070 + ...

Рисунок 1.18 – Данные на выходе модулятора

## 2. Канал связи

После модулятора данные поступают в канал связи, где могут возникать различные шумы, но они уже не так страшны, поскольку сообщение было подвержено обработке помехоустойчивым кодом. (сверточное кодирование + перемежение). На данном этапе промоделированный сигнал можно подать в радио-устройство для дальнейших исследований различных параметров. На рисунке 7.1 показана структурная схема дальнейшей работы.

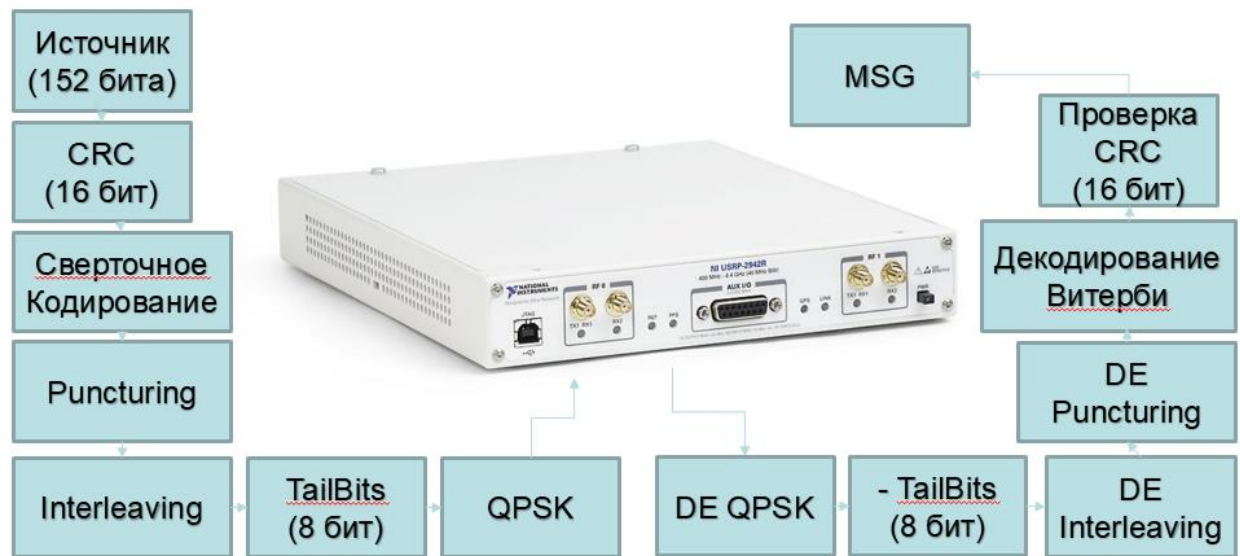


Рисунок 2.1 – Структурная схема

### 3. Реализация приёмника при моделирование канала N-BCCH в DLink

#### 3.1 Демодулятор QPSK

```
%__Демодулятор
for i = 1 : length(SpecOFDM1)
    if real(SpecOFDM1(i))>0 && imag(SpecOFDM1(i))>0
        outdemod((i-1)*2+1)=1
        outdemod(2*i)=1;
    end
    if real(SpecOFDM1(i))<0 && imag(SpecOFDM1(i))>0
        outdemod((i-1)*2+1)=1
        outdemod(2*i)=0;
    end
    if real(SpecOFDM1(i))<0 && imag(SpecOFDM1(i))<0
        outdemod((i-1)*2+1)=0
        outdemod(2*i)=0;
    end
    if real(SpecOFDM1(i))>0 && imag(SpecOFDM1(i))<0
        outdemod((i-1)*2+1)=0
        outdemod(2*i)=1;
    end
end
end
```

Рисунок 3.1 – Реализация демодулятора

Variables - outdemod

Interleaving x outdemod x outmod x

1x448 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1	1	1

Рисунок 3.2 – Данные на выходе демодулятора

Variables - Interleaving

Interleaving x outmod x

1x448 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1	1	1

Рисунок 3.3 – Данные на входе модулятора

### 3.2 Реализация деперемежителя

```

DEInterleaving = reshape (outdemod', 28,16);
DEInterleaving = DEInterleaving';
for k = 1:448;
    B=mod(12*(k-1)+floor((k-1)/2)+mod((k-1),2),16);
    j=mod(23*mod(5*(k-1),28)+ floor(7*(k-1)/16),28);
    B=B+1;
    j=j+1;
    decodeBIT2(k)= DEInterleaving(B,j);
end

```

Рисунок 3.4 – Реализация деперемежителя

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0	1

Рисунок 3.5 – Поток данных на выходе деперемежителя

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0	1

Рисунок 3.6 – Поток данных на входе перемежителя

### 3.3 Реализация восстановления проколотых бит

На данном этапе необходимо вставить на место удаленных битов «0», для того чтобы восстановить размерность сообщения

```

z = zeros(1,528);
z1 = [decodeBIT2(1:23)]
z2 = [decodeBIT2(344:end)]
n = 24;
for i = 24:5:423
    z(i) = decodeBIT2(n);
    n=n+1;
    z(i+1) = decodeBIT2(n);
    n=n+1;
    z(i+2) = decodeBIT2(n);
    n=n+1;
    z(i+3) = decodeBIT2(n);
    n=n+1;
end
codeBIT_DEpunct = [z1(1:end), z(24:423), z2(1:end)];

```

Рисунок 3.7 – Реализация восстановления проколотых бит

[illegible]

**Рисунок 3.8 – Данные на выходе депрокальвателя**

Variables - codeBIT

codeBIT\_DEpunct x codeBIT x

1x528 double

	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	1	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0	1	1

**Рисунок 3.9 – Данные на входе прокалывателя**

### 3.4 Реализация декодирования по алгоритму Витерби

```
% %_____Сверточное ДЕкодирование_____%
```

**Рисунок 3.10 – Реализация алгоритма Витерби**

Variables - decodeBIT

decodeBIT

1x176 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

**Рисунок 3.11 – Данные на выходе декодера**

The screenshot shows a window titled "Variables - BITunited". Inside, there is a tab labeled "decodeBIT" and another labeled "BITunited". Below the tabs, there is a label "1x176 double". The main area displays a single row of 176 bits, indexed from 1 to 18 (and then 0 at the end). The bits are: 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0. The first bit (index 1) is highlighted with a black background.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	0
1	1	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0

**Рисунок 3.12 – Данные на входе кодера**

### **Заключение**

В ходе данной работы был осуществлен процесс моделирования канала N-ВССН и детально рассмотрены каждые этапы. Смоделированная модель готова для дальнейших исследований на приёмо-передающих устройствах.

**Список использованных источников**

1. Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding // 3GPP TS 36.212 version 15.4.0 Release 15. 2019-01-11
2. Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT) // 3GPP TR 45.820 version 13.10 Release 13. 2016-08-16