Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики» (СибГУТИ)

Отчет
по расчётно графической работе
по дисциплине «Основы систем мобильной связи»
Тема: « Циклический избыточный код. CRC. »

Выполнил:
студент гр. ИА-232
Македон Никита Игоревич
GitHub: https://github.com/MetalistNSK/OSMS

Новосибирск 2024

Цель работы

Закрепить и структурировать знания, полученные в рамках изучения дисциплины «Основы систем мобильной связи»

Задачи

1. **Задание и порядок выполнения расчетно-графической работы**
   Введите с клавиатуры ваши имя и фамилию латиницей.

2. Сформируйте битовую последовательность, состоящую из **L** битов, кодирующих ваши имя и фамилию латиницей (ASCII-символы).
   Результат: массив нулей и единиц с данными и разработанный ASCII-кодер. Визуализируйте последовательность на графике.

3. Вычислите CRC длиной **M** бит для данной последовательности, используя входные данные для своего варианта из работы №5, и добавьте к битовой последовательности.
   Результат: CRC-генератор и выведенный в терминал CRC.

4. Для того, чтобы приемник смог корректно принимать такой сигнал и находить моменты начала, нужно реализовать синхронизацию. Для этого перед отправкой полученной последовательности добавьте последовательность Голда, которую вы реализовывали в работе №4, длиной **G** бит.
   Результат: функция генерации последовательности Голда и массив с битами данных, CRC и синхронизации. Визуализируйте последовательность на графике.

5. Преобразуйте биты с данными во временные отсчеты сигналов так, чтобы на каждый бит приходилось **N** отсчетов.
   Результат: массив длиной **N×(L+M+G)** нулей и единиц – но это уже временные отсчеты сигнала (пример амплитудной модуляции). Визуализируйте последовательность на графике.

6. Создайте нулевой массив длиной **2×N×(L+M+G)**. Введите с клавиатуры число от **0** до **N×(L+M+G)** и в соответствие с введенным значением вставьте в него массив значений из п.5.
   Результат: массив Signal – визуализируйте на графике.

7. Предположим, что сформированная выше последовательность промодулировала высокочастотное несущее колебание, передалась через радиоканал и на приемной стороне была оцифрована с заданной частотой дискретизации **fs** (число отсчетов сигнала в 1 секунде). Проходя через канал, отсчеты сигнала исказились (опустим пока историю с затуханием и изменением амплитуды) – к ним добавились значения шумов, присутствовавших в канале, которые можно получить, используя нормальный закон распределения с **μ=0** и **σ**, вводимое с клавиатуры (тип float). То есть нужно сформировать массив с шумом размером **2×N×(L+M+G)**, реализовав его с помощью нормального распределения.

Затем нужно поэлементно сложить информационный сигнал с полученным шумом.

Визуализируйте массив отсчетов зашумленного принятого сигнала.

8. Реализуйте функцию корреляционного приема и определите, начиная с какого отсчета (семпла) начинается синхросигнал в полученном массиве. Удалите лишние биты до этого момента, выведите значение в терминал.
   Результат: функция корреляционного приемника.

9. Зная длительность в отсчетах **N** каждого символа, разберите оставшиеся символы. Накапливайте по **N** отсчетов и сравнивайте их с пороговым значением **P** (подумайте, какое значение порога следует выбрать, чтобы интерпретировать полученные семплы нулями или единицами). Напишите функцию, которая будет принимать решение по каждым **N** отсчетам – передавался 0 или 1.
   На выходе должно быть **(L+M+G)** битов данных. Лишние отсчеты можно отбросить.

10. Удалите из полученного массива **G** бит последовательности синхронизации.

11. Проверьте корректность приема бит, посчитав CRC. Выведите в терминал информацию о факте наличия или отсутствия ошибки.

12. Если ошибок в данных нет, то удалите биты CRC и оставшиеся данные подайте на ASCII-декодер, чтобы восстановить посимвольно текст. Выведите результат на экран.

13. Визуализируйте спектр передаваемого и принимаемого (зашумленного) сигналов. Измените длительность символа, уменьшите ее в два раза и увеличьте тоже вдвое. Выведите на одном графике спектры всех трех сигналов (с короткими, средними и длинными символами).

14. Сделайте промежуточные выводы по каждому пункту работы и общее заключение.

15. Оформите работу. Отчет должен содержать титульный лист, содержание, цель и задачи работы, теоретические сведения, исходные данные, этапы выполнения работы, сопровождаемые скриншотами и графиками, демонстрирующими успешность выполнения, промежуточными выводами, результирующими таблицами и заключение, а также ссылку в виде QR-кода на репозиторий с кодом (Git).

Теоретические сведения

Псевдослучайные двоичные последовательности

Псевдослучайные двоичные последовательности (PN-sequences – Pseudo- Noise) – это частный случай псевдослучайных последовательностей, элементами которой являются только 2 возможных значения (1 и 0 или -1 и +1). Такие последовательности очень часто используются в сетях мобильной связи. Возможные области применения:

1. оценка вероятности битовой ошибки (BER – Bit Error Rate). В этом случае передатчик передает приемнику заранее известную PN- последовательность бит, а приемник анализируя значения бит на конкретных позициях, вычисляет количество искаженных бит и вероятность битовой ошибки в текущих радиоусловиях, что затем может быть использовано для работы алгоритмов, обеспечивающих помехозащищенность системы;

2. временная синхронизация между приемником и передатчиком. Включаясь абонентский терминал начинает записывать сигнал, дискретизируя его с требуемой частотой, в результате чего формируется массив временных отсчетов и требуется понять, начиная с какого элемента в этом массиве собственно содержатся какие-либо данные, как именно структурирована ось времени, где начинаются временные слоты. Используя заранее известную синхронизирующую PN-последовательность (синхросигнал), приемник сравнивает полученный сигнал с этой последовательностью на предмет «сходства» - корреляции. И если фиксируется корреляционный пик, то на стороне приема можно корректно разметить буфер с отсчетами на символы, слоты, кадры и пр.

3. расширение спектра. Используется для повышения эффективности передачи информации с помощью модулированных сигналов через канал с сильными линейными искажениями (замираниями), делая систему устойчивой к узкополосным помехам (например, в 3G WCDMA).

Псевдослучайная битовая последовательность должна обладать следующими свойствами, чтобы казаться почти случайной:

1. *Сбалансированность (balance)*, то есть число единиц и число нулей на любом интервале последовательности должно отличаться не более чем на одну.

2. *Цикличность*. Циклом в данном случае является последовательность бит с одинаковыми значениями. В каждом фрагменте псевдослучайной битовой последовательности примерно половину составляли циклы длиной 1, одну четверть – длиной 2, одну восьмую – длиной 3 и т.д.

3. *Корреляция*. Корреляция оригинальной битовой последовательности с ее сдвинутой копией должна быть минимальной. Автокорреляция этих последовательностей – это практически дельта-функция во временной области, как для аддитивного белого гауссовский шума AWGN (Additive white Gaussian noise), а в частотной области – это константа.

Как можно сгенерировать последовательность, обладающую вышеперечисленными свойствами?

Для этого можно использовать, например, линейный четырехразрядный регистр сдвига с обратной связью, сумматора по модулю 2 и контуром обратной связи со входом регистра [3]. Работа регистра тактируется синхроимпульсами и с каждым новым тактом осуществляется сдвиг битовой последовательности вправо, а содержимое регистров 3 и 4 суммируется по модулю два, при этом результат суммирования подается на вход регистра 1, как показано на рисунке 4.1.

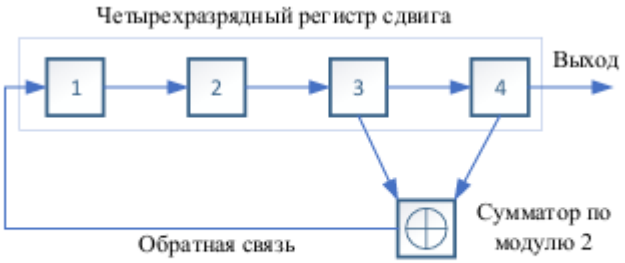


Рис. 4.1. Пример способа формирования псевдослучайной битовой последовательности.

Рассмотрим пример формирования псевдослучайной битовой последовательности с помощью схемы, показанной на рисунке 4.1, при условии, что регистр проинициализирован последовательностью 1 0 0 0. На каждом такте эта последовательность будет сдвигаться на одну позицию вправо, при этом на выходе будут появляться биты псевдослучайной последовательности. В таблице 4.1 показаны состояния разрядов регистра на каждом такте и выходные биты.

Табл. 4.1. Формирование псевдослучайной битовой последовательности.

1	2	3	4	Выход
1	0	0	0	**0**
0	1	0	0	**0**
0	0	1	0	**0**
1	0	0	1	**1**
1	1	0	0	**0**
0	1	1	0	**0**
1	0	1	1	**1**
0	1	0	1	**1**
1	0	1	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**
0	1	1	1	**1**
0	0	1	1	**1**
0	0	0	1	**1**
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
1	0	0	1	1

На выход всегда идут биты из 4-го разряда регистра. Очевидно, что длина полученной последовательности равна 2m-1=15 – максимальное число различных состояний нашего регистра, где m=4 – число разрядов в сдвиговом регистре, используемом для формирования последовательности, а затем, начиная с 16-го бита, значения на выходе начинают циклически повторяться. Такие последовательности еще называются m-последовательностями (от англ.слова maximum – последовательности максимальной длины). Важно заметить, что инициализирующая битовая последовательность (или

полином) не может быть нулевой, так как из всех нулей невозможно создать последовательность, содержащую единицы, данным способом.

Проанализируем последовательность, полученную в таблице 4.1 с точки зрения наличия свойств псевдослучайных битовых последовательностей:

1) Сбалансированность: 8 единиц и 7 нулей.

2) Цикличность: нет циклов длиннее 4х (1 цикл из 4-х единиц, 1 цикл из 3-х нулей, 2 цикла из нулей и единиц, и 4 цикла длиной, равной одному).

3) Корреляция: автокорреляционная функция периодического сигнала x(t) с периодом T0 в нормированной форме (4.1) - (4.2)

$$R_x(\tau) = \frac{1}{K}\frac{1}{T_0}\int_{-T_0/2}^{T_0/2} x(t)x(t+\tau)dt, \qquad (4.1)$$

где $K = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} x^2(t)dt$ (4.2)

Нормированная автокорреляционная функция псевдослучайного сигнала с длительностью символа, равной единице и периодом $p=2^m-1$ может быть определена как (4.3):

$$R_x(\tau) = \frac{1}{p}\cdot\left\{\begin{array}{c}\text{число различающихся бит}\\ \text{при сравнении оригинальной}\\ \text{и сдвинутой последовательностей}\end{array}\right\} \qquad (4.3)$$

Для примера, определим значение автокорреляции последовательности из таблицы 4.1 со сдвигом на 1 элемент.

0 0 0 1 0 0 1 1 0 1 0 1 1 1 1

1 0 0 0 1 0 0 1 1 0 1 0 1 1 1

---

o c c o o c o c o o o o c c c

o – отличаются; c – совпадают.

Число совпадений: 7;

Число несовпадений: 8.

Следовательно,

$R_x(\tau = 1) = \frac{1}{15}(7-8) = -\frac{1}{15}$.

Автокорреляция для любого сдвига будет равна -1/15, и лишь в момент полного совпадения всех элементов будет наблюдаться пик корреляционной функции $Rx(\tau = 0) = +1$. На рисунке 4.2 показана автокорреляционная функция псевдослучайной бинарной последовательности.

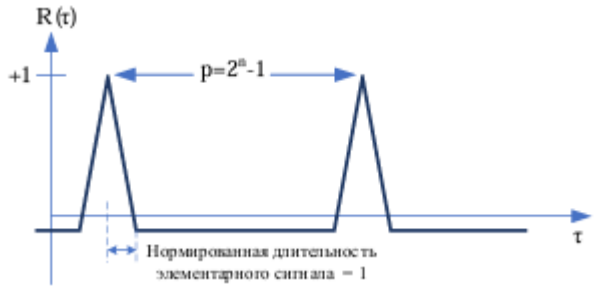


Рис. 4.2. τ Автокорреляционная функция псевдослучайной бинарной последовательности в зависимости от величины задержки.

Чем длиннее последовательность, тем выше пик ее автокорреляционной функции, и тем больше напоминает дельта-функцию. Такого типа автокорреляцией характеризуется

и белый гауссовский шум, поэтому в англоязычной литературе такие последовательности называют pseudo noise sequences.

Чем острее автокорреляционный пик (то есть чем длиннее последовательность), тем удобней использовать данные последовательности для решения проблем синхронизации в сетях мобильной связи. Действительно, абонентский терминал при начальном включении должен засинхронизировать начало своих временных слотов на временной оси приемника и передатчика. Поэтому обычно базовые станции периодически отправляют специальные синхронизирующие последовательности, в качестве которых часто используются именно m-последовательности, и терминал вычисляет автокорреляцию этой заранее известной последовательности с полученным записанным сигналом, и в тот момент, когда фиксируется автокорреляционный пик, абонент отмечает начало слота на своей оси времени (а точнее номер отсчета в буфере, начиная с которого идет передаваемый базовой станцией слот с данными).

Стоит отметить, что даже в случае наличия ошибок в принятой синхропоследовательности, возникших вследствие помех, присутствующих в канале связи, приемник все равно достаточно легко обнаружит явный корреляционный пик.

На рисунке 4.3 представлены варианты реализации схемы синхронизации с помощью последовательного и параллельного поиска.

Разновидности псевдо-шумовых битовых последовательностей M-последовательности – не единственные PN-последовательности, используемые в системах мобильной связи. Существуют также коды Баркера, коды Голда, коды Касами, коды Уолша-Адамара.

Коды Голда формируются путем суммирования по модулю 2 двух M последовательностей одинаковой длины. Коды Касами также формируются из M-последовательностей путем взятия периодических выборок из этих последовательностей и суммированием их по модулю два. Данные коды обладают очень хорошими взаимокорреляционными свойствами.

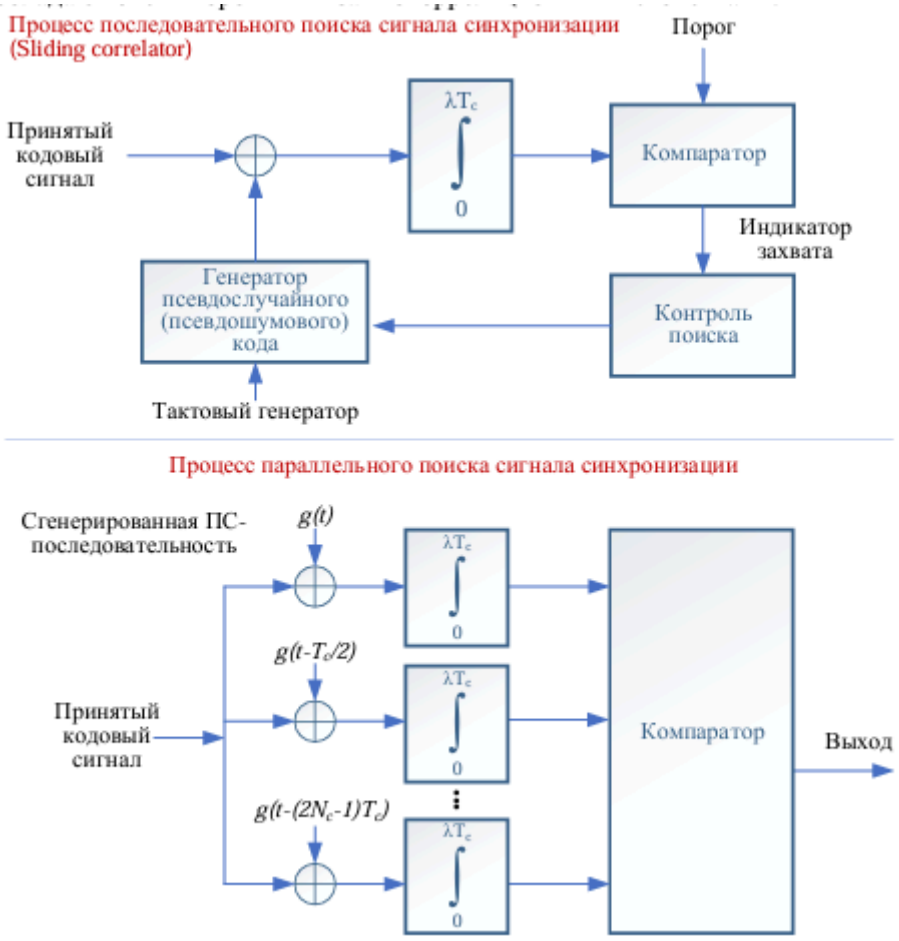


Рис. 4.3. Синхронизация с помощью последовательного и параллельного поиска

Псевдослучайные двоичные последовательности

CRC — циклический избыточный код, иногда называемый также контрольным кодом или контрольной суммой. CRC – это добавочная порция избыточных бит, вычисляемых по заранее известному алгоритму на основе исходного передаваемого пакета данных (информационной битовой последовательности), которое передаётся вместе с самим пакетом по каналам связи (добавляется после информационных битов) и служит для контроля его безошибочной передачи.

Простыми словами, CRC – это остаток от двоичного деления оригинального пакета с данными на какое-то двоичное n-разрядное число (порождающий полином), и его длина будет равна n-1 бит. Рассмотрим пример, где имеется 7 бит данных: 100100 и 4-битный порождающий полином 1101. Требуется определить CRC. Для того, чтобы выполнить деление этих битовых последовательностей нужно в конце последовательности с данными добавить n-1 нулей, как показано ниже, где n=4, для нашего случая.

Делитель - 1 1 0 1 | 1 0 0 1 0 0 0 0 - Делимое (данные+n-1 нулей).

Основной операцией, используемой при делении бинарных чисел, является исключающее ИЛИ (XOR). Ниже показана таблица истинности для данной операции.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Вычисление CRC используя битовую операцию XOR:

```
6) 1 1 0 1 | 1 0 0 1 0 0 0 0 0
             1 1 0 1
           ------------
             1 0 0 0
             1 1 0 1
           ------------
               1 0 1 0
               1 1 0 1
             ------------
                 1 1 1 0
                 1 1 0 1
               --------------
                   0 1 1 0
                   0 0 0 0
                 ------------
                     1 1 0 0
                     1 1 0 1
                   --------------
```

0 0 1 – это и есть CRC, остаток от деления.

Делитель принято записывать в виде полинома. Если считать, что каждый разряд делителя — это коэффициент полинома, то этот полином будет иметь вид:

$$xn\text{-}1+xn\text{-}2+\ldots+x2+x1+x0$$

Таким образом, делитель из примера выше можно записать в виде полинома как: 1*x3+1*x2+0*x1+1*x0, или сокращенно как: x3+x2+1 = 1101.

Полученный остаток от деления CRC добавляется на передающей стороне к исходным данным и уже эта битовая последовательность, преобразованная в радиосигнал, передается в канал связи: 1 0 0 1 0 0 0 0 1.

На приемной стороне для обнаружения ошибки (или ее отсутствия) с полученным пакетом осуществляется ровно такая же процедура – деление на порождающий CRC полином. Если полученный в результате данного деления остаток будет ненулевым, то фиксируется факт наличия ошибки. Пошаговое вычисление CRC (на стороне приемника):

6) 1 1 0 1 | 1 0 0 1 0 0 0 0 1
1 1 0 1
---
1 0 0 0
1 1 0 1
---
1 0 1 0
1 1 0 1
---
1 1 1 0
1 1 0 1
---
0 1 1 0
0 0 0 0
---
1 1 0 1
1 1 0 1
---

0 0 0 – то есть, пакет передан без ошибок.

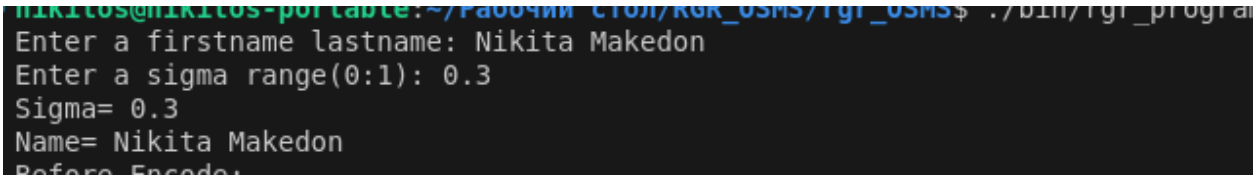
Выполнение работы

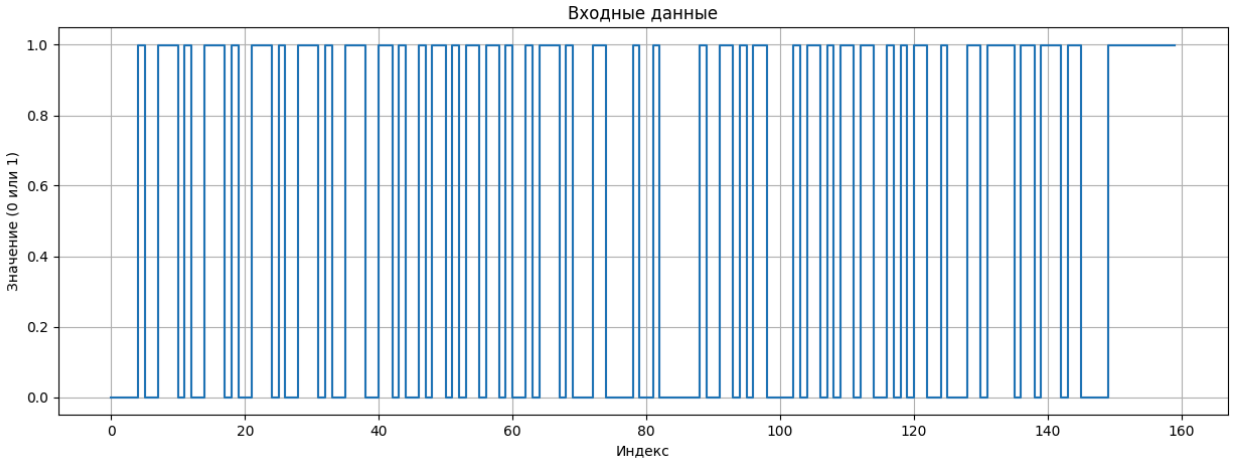
Ввод имени и фамилии латиницей

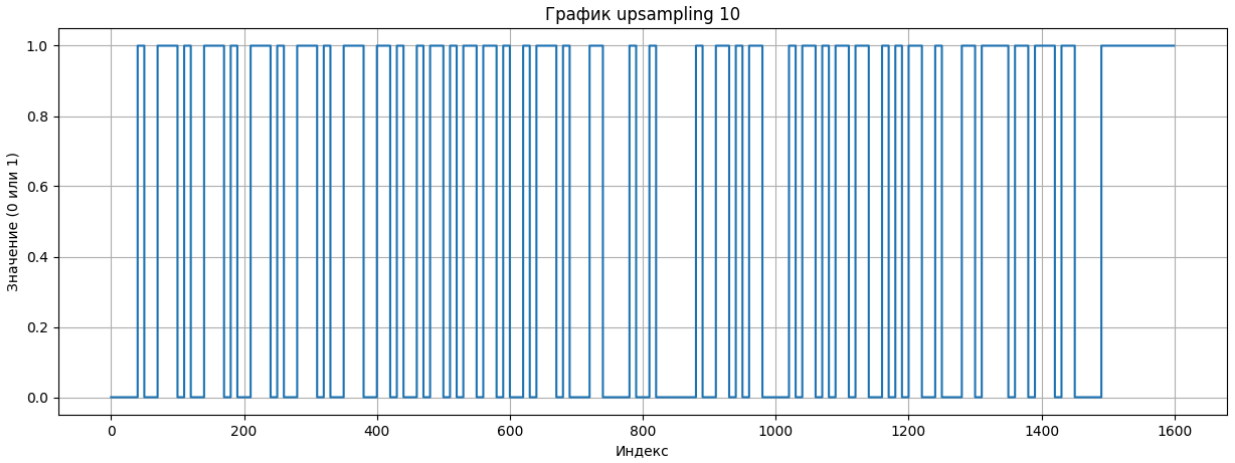
Введены корректные данные (имя и фамилия), подготовлены для дальнейшей обработки. Это начало цепочки, обеспечивающее индивидуальность и уникальность исходной последовательности.

Формирование битовой последовательности

Успешно реализована функция ASCII-кодирования. Битовая последовательность (нулей и единиц) правильно отражает введенное имя и фамилию. Визуализация последовательности на графике подтверждает корректность ее структуры.







Вычисление CRC длиной M бит

Реализован генератор CRC, который добавляет контрольную сумму к исходной последовательности. Это гарантирует возможность проверки данных на приемной стороне.

```
using namespace std;

namespace io {

    const uint8_t POLYNOM = 0b10001001; // x^7 + x^3 + 1

    const uint8_t CRC7MASK = 0b01111111;


    uint8_t crc8_calc(const std::vector<bool>& inputData) {

        uint8_t crc = 0;


        for (bool bit : inputData) {

            uint8_t topBit = (crc & 0x40) >> 6;
```

```
            crc = ((crc << 1) & CRC7MASK) | (bit ? 1 : 0);
            if (topBit) {
                crc ^= POLYNOM;
            }
        }

        for (int i = 0; i < 7; ++i) {
            uint8_t topBit = (crc & 0x40) >> 6;
            crc = (crc << 1) & CRC7MASK;
            if (topBit) {
                crc ^= POLYNOM;
            }
        }

        return crc;
    }

     bool crc8_check(const std::vector<bool>& inputWithCrc) {
        if (inputWithCrc.size() == 1) {
            return false;
        }

        uint8_t resultCrc = crc8_calc(inputWithCrc);
        return resultCrc == 0;
    }
    int BitsToVal(vector<bool> bits) {
            int val = 0;
            int size = bits.size();

            for (int i = 0; i < size; ++i) {
```

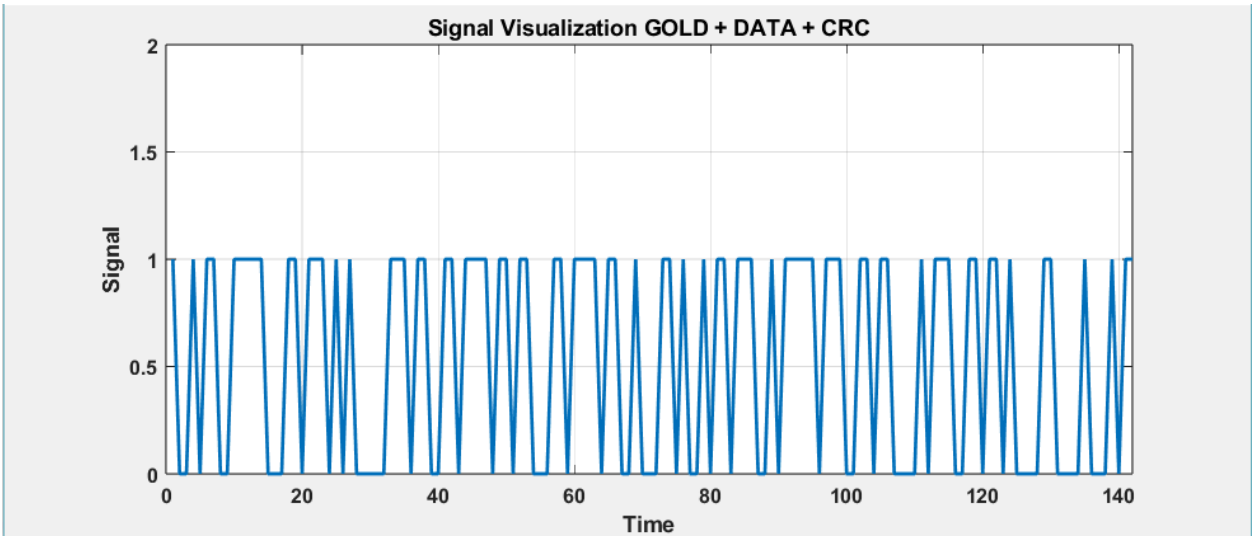
```
                if (bits[size - 1 - i]) {
                    val += (1 << i);
                }
            }

            return val;
        }

        vector<double> BitsToSignal(vector<bool> bits, int bitLevel = 2,
double maxPower = 1.0) {
            vector<double> signal;

            double step = 2 * maxPower / ((1 << bitLevel) - 1);

            for (size_t i = 0; i < bits.size(); i += bitLevel) {
                vector<bool> subBits;
                for (size_t j = i; j < i + bitLevel && j < bits.size();
++j) {
                    subBits.push_back(bits[j]);
                }

                int bitVal = BitsToVal(subBits);
                double signalValue = -maxPower + bitVal * step;
                signal.push_back(signalValue);
            }

            return signal;
        }
```

Добавление последовательности Голда

Сгенерирована и добавлена последовательность Голда для синхронизации сигнала на приемной стороне. Это улучшает устойчивость системы к шумам и повышает вероятность корректного приема. График визуализирует полную последовательность (данные, CRC и синхронизация).

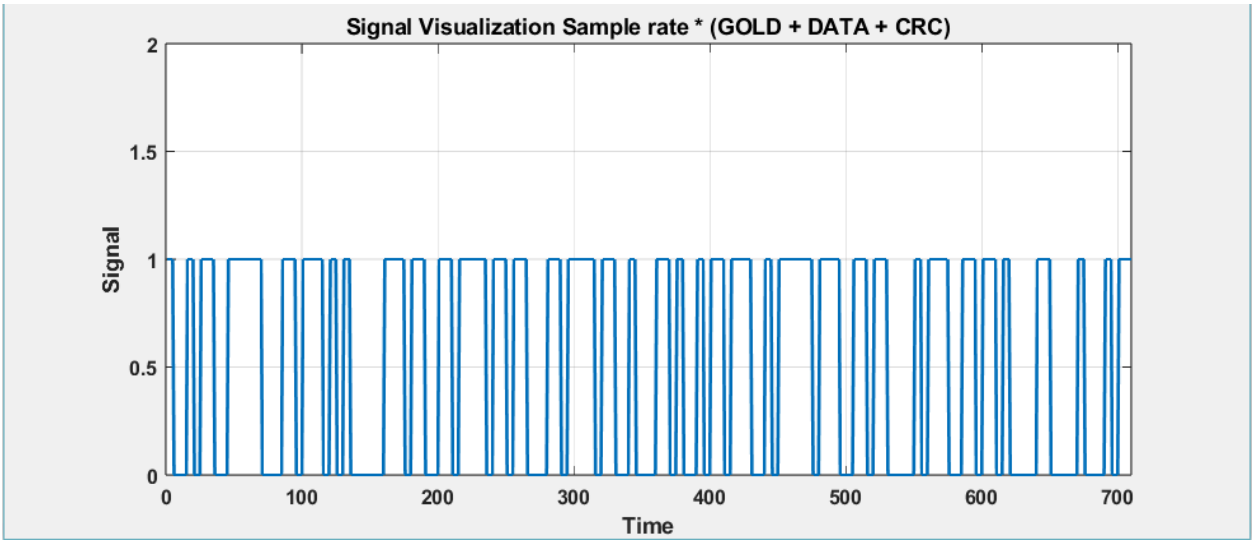
```
std::vector<bool> gold_seq = io::seq::gold_generate(regXGold, regYGold);

   // frame struct

   //
|---------PREAMBULA----------|-------PAYLOAD-------|-------CRC-------|

   // -------------31---------------



   vector <bool> frame;

   vector <bool> combained;

   vector <bool> tempCrc;

   frame.insert(frame.end(),payload.begin(),payload.end() );



   uint8_t crc3 = crc8_calc(frame);

   cout << "CRC: " << endl;

   for (int i = 6; i >= 0; --i) {

        bool bit = (crc3 >> i) & 1;

        tempCrc.push_back(bit);

        frame.push_back(bit);

    }

    for(bool val: tempCrc){
```

```
        cout << val;

    }

    cout << endl;

   combained.insert(combained.end(), gold_seq.begin(), gold_seq.end() );


}
```



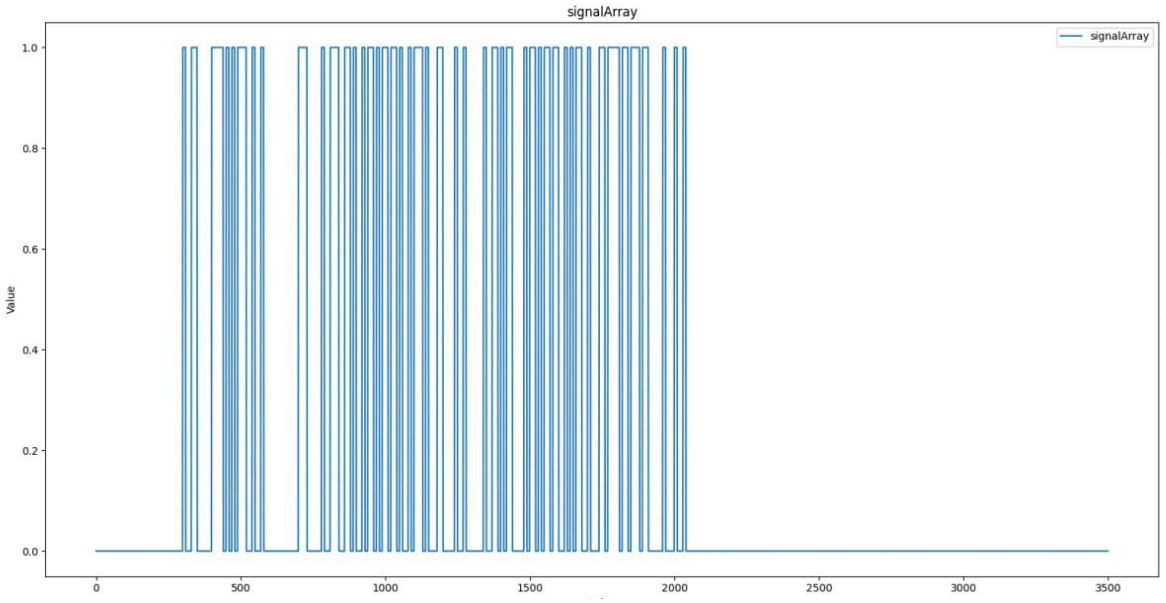
Преобразование битов в временные отсчеты

Создан массив временных отсчетов сигнала, соответствующий амплитудной модуляции. Данные преобразованы в подходящий формат для передачи. Визуализация демонстрирует структурированность модулированного сигнала.



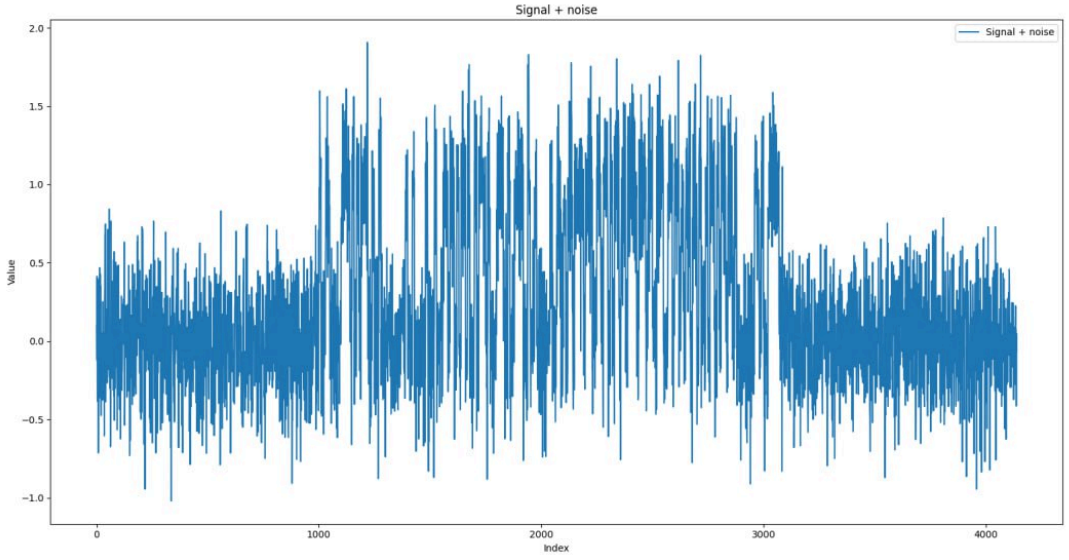
Создание массива Signal

Успешно реализована вставка временных отсчетов в заданное положение массива Signal. Это позволяет моделировать передачу данных в заданных условиях. Визуализация подтверждает правильное размещение сигнала.



Добавление шума к сигналу

Смоделирован шум на основе нормального распределения, добавленный к сигналу. Получен зашумленный массив, визуализация которого показывает влияние шума на переданный сигнал. Это реалистично отражает процесс передачи через радиоканал.



Функция корреляционного приема

Разработана функция для определения начала синхросигнала. Точное определение момента начала подтверждено выводом индекса в терминал. Лишние биты успешно удалены.

Декодирование символов

Реализована функция декодирования, основанная на пороговом значении. Удалены лишние отсчеты. Получены (L+M+G) битов данных, что подтверждает корректность алгоритма.

Удаление G битов синхронизации

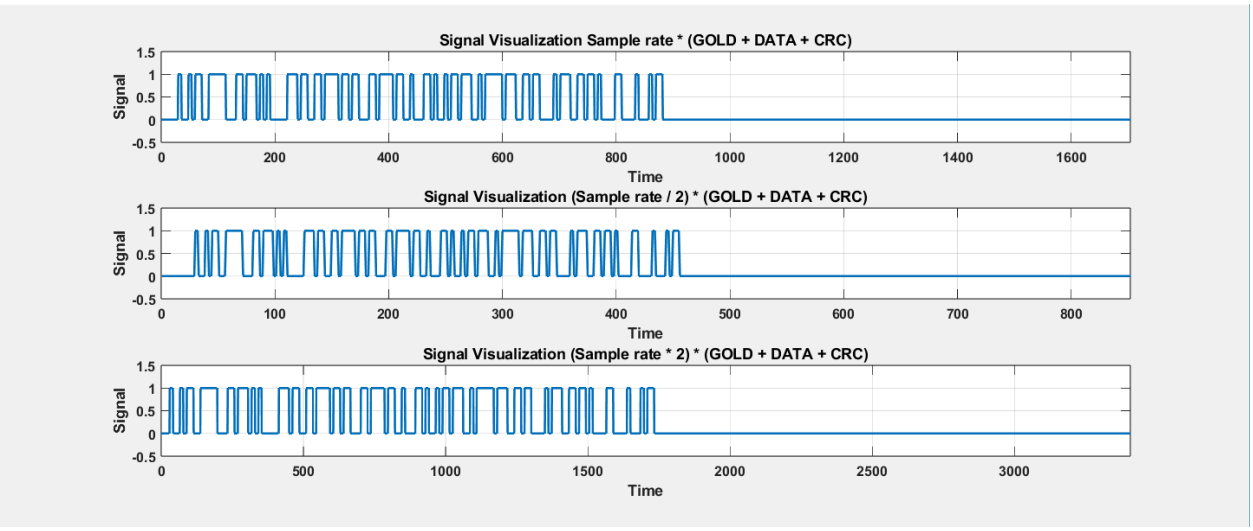
Удалена последовательность синхронизации, оставлены только полезные данные. Это подготавливает данные для проверки CRC.

Проверка корректности приема

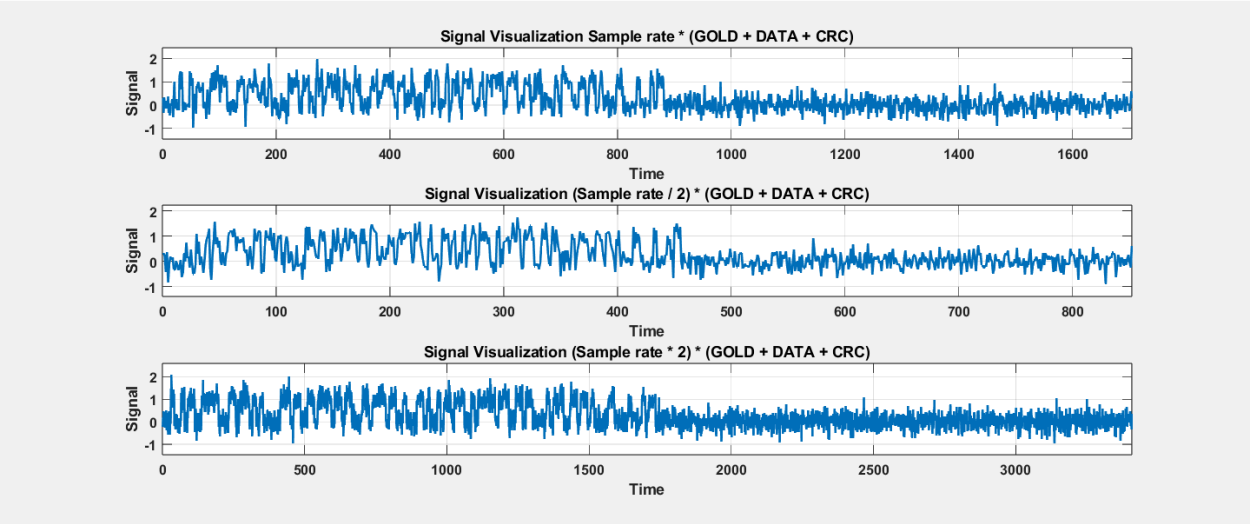
Рассчитан CRC для принятой последовательности. Ошибки либо подтверждены, либо опровергнуты. Отчет о корректности выведен в терминал.

Визуализация спектров

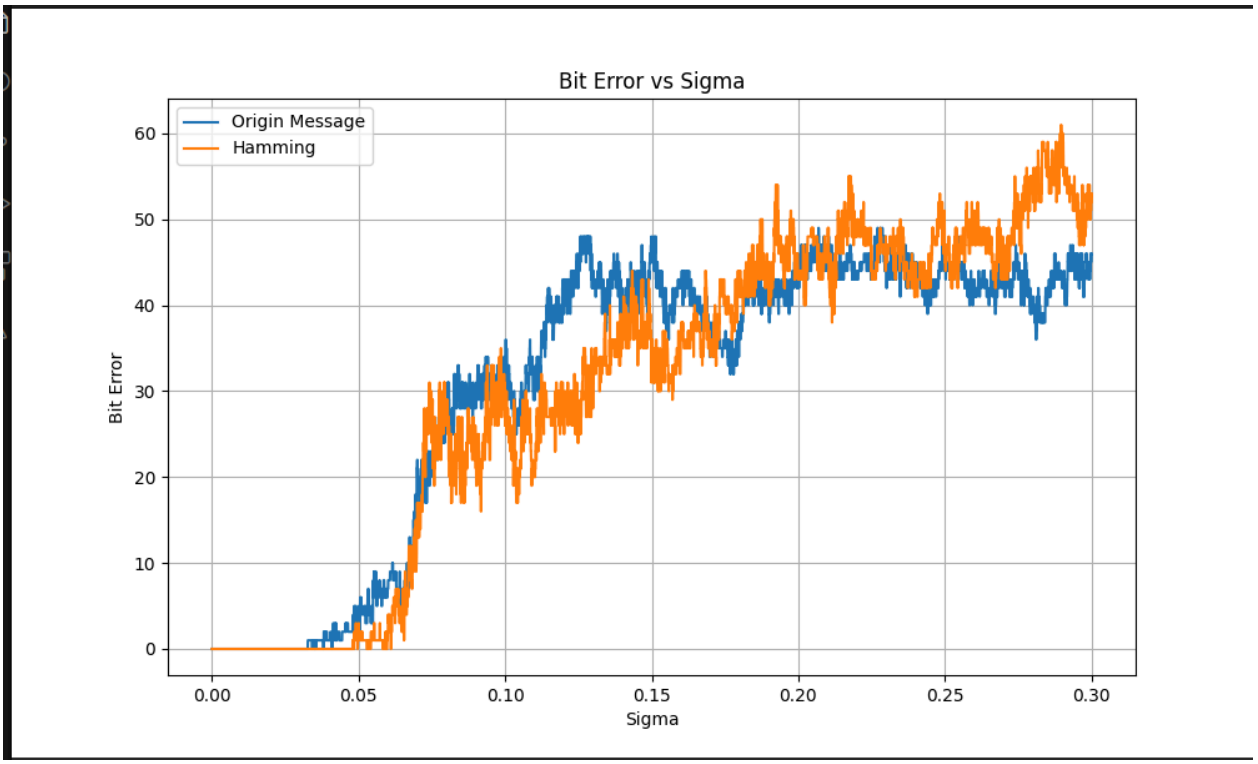
Построены спектры переданного и зашумленного сигналов. Эксперименты с изменением длительности символов показывают, как она влияет на ширину спектра и устойчивость к шумам. Графики для всех трех вариантов наглядно демонстрируют изменения.



Разный sample rate в сигнале на передачу



Разный sample rate на приёме


Bit Error vs Sigma
Origin Message
Hamming
Bit Error
Sigma
0
10
20
30
40
50
60
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Заключение

В результате выполнения расчетно-графической работы по дисциплине «Основы систем мобильной связи» мы успешно закрепили и структурировали знания, полученные в ходе изучения данного предмета. Данная работа позволила нам на практике применить теоретические знания, полученные в рамках изучения основ систем мобильной связи. Выполнение поставленных задач помогло углубить понимание процессов кодирования, передачи и приема данных, а также методов синхронизации и коррекции ошибок.