

Факультет программной инженерии и компьютерной техники

Компьютерные сети

Лабораторная работа №1

Преподаватель: Маркина Татьяна Анатольевна

Выполнил: Кульбако Артемий Юрьевич Р33112

# Задание

1. Ознакомиться с постановкой задачи и изучить необходимые теоретические сведения.
2. Сформировать исходное сообщение в соответствии с этапом 1.
3. Выполнить физическое кодирование исходного сообщения не менее, чем тремя способами, включая, в качестве обязательного, манчестерское кодирование. Рассчитать частотные характеристики передаваемого сигнала для рассматриваемых способов кодирования и определить требуемую для эффективной передачи сообщения пропускную способность канала связи (этап 2).
4. Выполнить логическое кодирование исходного сообщения, используя избыточное кодирование 4В/5В и скремблирование. Рассчитать частотные характеристики передаваемого сигнала для рассматриваемых способов кодирования и определить требуемую для эффективной передачи сообщения пропускную способность канала связи (этапы 3 и 4).
5. Выполнить сравнительный анализ рассмотренных способов кодирования и выбрать наилучший способ для передачи исходного сообщения (этап 5).
6. Оформить отчёт и сдать его на проверку.
7. В назначенное преподавателем время защитить задание.

# Выполнение

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I этап** | | | | | | | | | | | | | | |
| К | у | л | ь | б | а | к | | о |  | А | . |  | Ю | . |
| CA | F3 | EB | FC | E1 | E0 | EA | | EE | 20 | C0 | 2E | 20 | DE | 2E |
| 11001010 | 11110011 | 11101011 | 11111100 | 11100001 | 11100000 | 11101010 | | 11101110 | 00100000 | 11000000 | 00101110 | 00100000 | 11011110 | 00101110 |
| *Вес = 14 байт (112 бит)* | | | | | | | | | | | | | | |
| **II этап** | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | |
|  | | | | | | | | 6.6 ГГц | | | | | | |
|  | | | | | | | |  | | | | | | |
|  | | | | | | | |  | | | | | | |
|  | | | | | | | |  | | | | | | |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | | | | | | |  |  |  | | NRZ | 500 | | | | | | | 62.5 | 219 | 3500 | | RZ | 1000 | | | | | | | 500 | 750 | 6600 | | AMI | 500 | | | | | | | 250 | 454 | 3300 | | Манчестерский код | 1000 | | | | | | | 500 | 797 | 6600 | | NRZI | 500 | | | | | | | 167 | 344 | 3400 | |  | | NRZ | RZ | M | AMI | NRZI | | Минимизация спектра | | + | - | - | + | + | | Самосинхронизация | | - | + | + | - | - | | Отсутствие постоянной составляющей | | - | + | + | - | - | | Обнаружение ошибок | | - | + | + | + | + | | Низкая стоимость | | + | - | + | - | + |   По сравнительной таблице можно сделать вывод о том, следующие способы физического кодирования в качестве наилучших: Манчестерский RZ и NRZI (по характеристикам, на его месте мог бы быть и NRZ, но он проигрывает в стоимости реализации). | | | | | | | | | | | | | | |
| **III этап** | | | | | | | | | | | | | | |
| Логическое кодирование применяется для ликвидации длинных последовательностей у AMI, NRZI, MLT-3. | | | | | | | | | | | | | | |
| *4B/5B* | | | | | | | | | | | | | | |
| 1101010110 | 1110110101 | 1110010111 | 1110111010 | 1110001001 | 1110011110 | 1110010110 | | 1110011100 | 1010011110 | 1101011110 | 1010011100 | 1010011110 | 1101111100 | 1010011100 |
| D5BB5E5FBAE279EE5B9CA7B5EA729EDF29C | | | | | | | | | | | | | | |
| *Вес = 17.5 байт (140 бит)*  *Избыточность = 25%* | | | | | | | | | | | | | | |
|  | | | | | | |  | | | | | | | |
|  | | | | | | |  | | | | | | | |
| **IV этап** | | | | | | | | | | | | | | |
| Автоматизируем скремблирование для предложенных полиномов с помощью скрипта:  List<String> code = *this*.args[0].toCharArray().collect **{** it.toString().toBoolean() **}** *def* polynomial = **{** x, y, z = *null* **->** *if* (z) x ^ y ^ z *else* x ^ y **}** *def* scramble = **{** arr, a, b **->** arr.withIndex().collect **{** el, i **->** *if* (i *in* 3..b - 1) polynomial(el, code[i - a])  *else if* (i >= b) polynomial(el, code[i - a], code[i - b])  *else* el  **}}** *def* toReadable = **{** (it.collect **{** n **->** n ? 1 : 0 **}**.join()) **}** println([[3, 5], [5, 7]]  .collect **{** scramble(code, it[0], it[1]) **}** .collect**{** toReadable it **}**) | | | | | | | | | | | | | | |
| *Скремблирование (Bi = Ai⊕Bi-3⊕Bi-5)* | | | | | | | | | | | | | | |
| 1101010111111010000010011101110010011010110100111111000011100100100101011101111000101010100101011100001100011010 | | | | | | | | | | | | | | |
| D5FA09DC9AD3F0E495DE2A95C31A | | | | | | | | | | | | | | |
| *Скремблирование (Bi = Ai ⊕Bi-5 ⊕Bi-7)* | | | | | | | | | | | | | | |
| 1101110100110001100100110111010011111111001011000010110001101100100011011000011110101111000011011001100101100011 | | | | | | | | | | | | | | |
| DD319374FF2C2C6C8D87AF0D9963 | | | | | | | | | | | | | | |
| *Вес = 14 байт (112 бит)* | | | | | | | | | | | | | | |
| Выберем первый полином, т.к. в нём максимальная длина последовательности меньше. | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | |
|  | | | | | | | |  | | | | | | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | | AMI (4B/5B) | 500 | 250 | 485 | 3100 | | NRZI (4B/5B) | 500 | 167 | 344 | 3200 | | AMI (скремблирование) | 500 | 100 | 407 | 3100 | | NRZI (скремблирование) | 500 | 84 | 84 | 3300 |  |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | AMI (4B/5B) | NRZI (4B/5B) | AMI (скремблирование) | NRZI (скремблирование) | | Минимизация спектра | + | + | + | + | | Самосинхронизация | - | - | - | - | | Отсутствие постоянной составляющей | - | - | - | - | | Обнаружение ошибок | + | + | + | + | | Низкая стоимость | - | + | - | + |   По таблице можно сделать вывод о том, что на данной последовательности сигналов лучше всего отработала пара NRZI-скремблирование, она же выигрывает и по характеристикам. | | | | | | | | | | | | | | |