МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств (ТС и ВС)

Отчет по производственной практике
по дисциплине
SDR

по теме:

АРХИТЕКТУРА SDR-УСТРОЙСТВ. ПРИМЕРЫ ФОРМИРОВАНИЯ I/Q-СЭМПЛОВ ПРОИЗВОЛЬНОЙ ФОРМЫ. РАБОТА С БУФЕРОМ ПРИЕМА SDR

Студент:
Группа ИА-331 *Я.А Гмыря*

Предподаватели:
Лектор *Калачиков А.А*
Семинарист *Ахпашев А.В*
Семинарист *Попович И.А*

Новосибирск 2025 г.

СОДЕРЖАНИЕ

ЦЕЛЬ И ЗАДАЧИ

Цель:

Повторить архитектуру систем цифровой связи, изучить базовые типы модуляции BPSK и QPSK, программно реализовать их.

Задачи:

1. Прослушать лекцию.
2. На основе полученных знаний выполнить программную реализацию BPSK и QPSK модуляции.
3. Создать треугольный сигнал и сигнал параболической формы, отправить его с SDR, а потом принять и посмотреть, что получилось.

ЛЕКЦИЯ

Введение

На этом занятии вспомним архитектуру цифровой системы связи и рассмотрим BPSK и QPSK модуляции.

Архитектура цифровой системы связи

Задачи цифровой системы связи

Какие задачи у системы связи? Задача системы связи заключается в надежном передать поток бит на заданной скорости по каналу связи. Для передачи по каналу связи мы используем электромагнитные колебания - sin и cos какой-то частоты.

Архитектура передатчика

Базовая архитектура цифровой системы связи в простейшем случае состоит из приемника, передатчика и радиоканала. Рассмотрим упрощенную архитектуру передатчика:

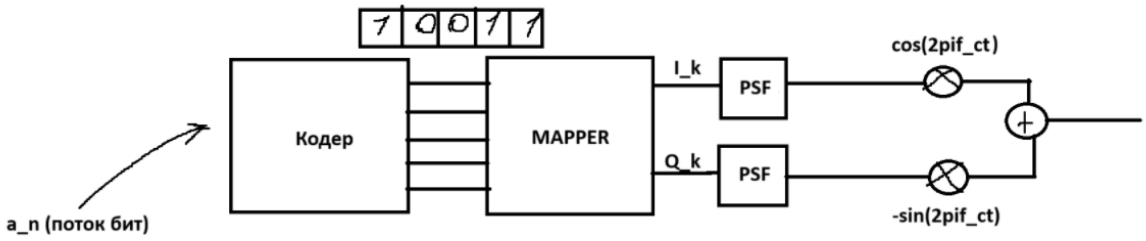

Рисунок 1 — Архитектура передатчика

Формируется поток бит, который поступает на кодер. В нашем случае кодер просто делит поток бит на блоки определенной длины. У кодера один вход, по которому последовательно поступают биты, а на выходе N-битная шина, которая уже параллельно передает биты на мапер. Кодер можно представить в виде схемы следующим образом:

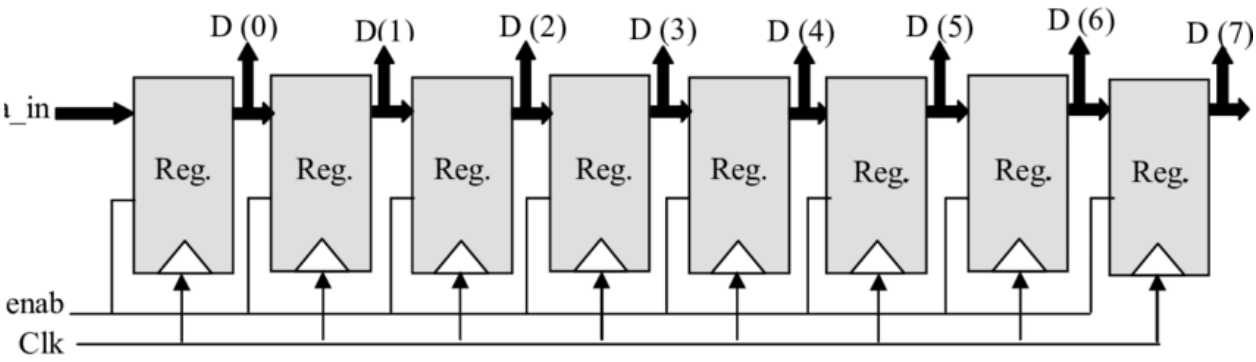

Рисунок 2 — Схема последовательно-параллельного преобразователя

Имеется N триггеров с общим тактовым сигналом. По каждому тактовому сигналу поток бит будет ”продвигаться”по тригерам и попадать на N-битную шину, т.е параллельно выводиться из устройства.

Далее блоки битов попадают на мапер, который ставит в соответствие каждому блоку числа I и Q. Если блок бит имеет размерность N, то в мапере будет заложено 2^N комбинаций. На выходе маппера параллельно получим числа I и Q.

Сигнал, который мы хоти сформировать имеет вид имеет вид $S_k(t) = I_k cos(2\pi f_c t) - Q_k sin(2\pi f_c t)$, где k - номер текущего символа, I_k, Q_k - координаты символа в сигнальной диаграмме. Эти координаты мы получаем на выходе маппера. Эти числа в будущем станут парамтерами колебания.

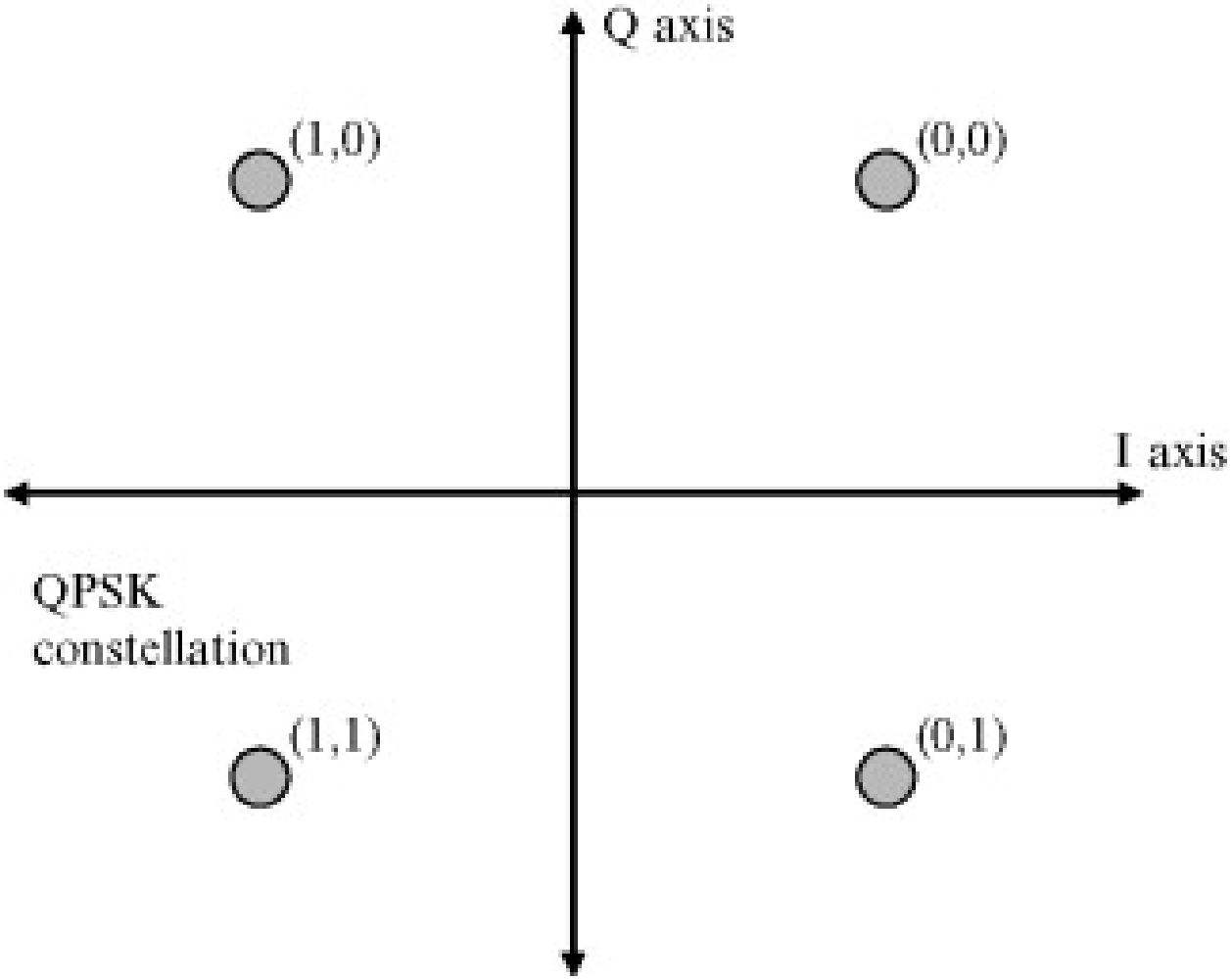

Рисунок 3 — Пример сигнальной диаграммы для QPSK модуляции

Далее I и Q попадают на Pulse Shaping Filter (PSF). Фильтр выполняет 2 задачи: определяет ширину спектра радиосигнала и его форму, а также применяется на приемной стороне для символьной синхронизации. Фильтр характеризуется важным параметром во временной области - импульсной характеристикоой $g(t)$. Импульсная характеристика может быть различной, но для простоты восприятия будем рассматривать только прямоугольную характеристику. На выходе фильтра получим прямоугольные отрезки сигналов.

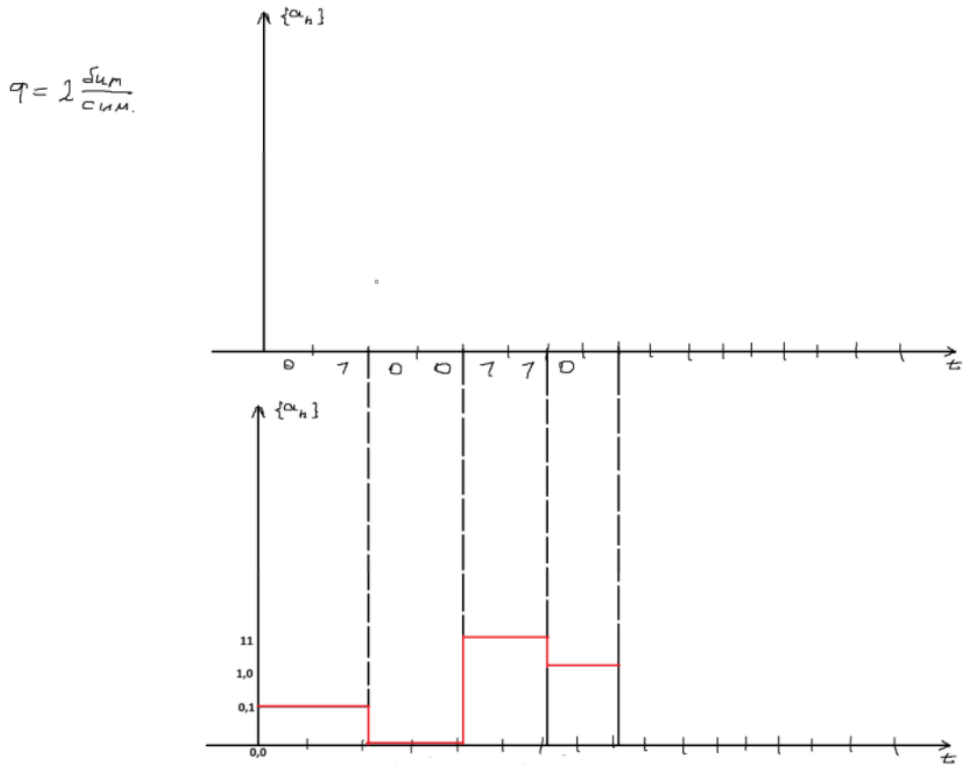

Рисунок 4 — Пример формирования символов

Длительность символов вычисляется как $1/R_s$, где R_s - символьная скорость. В свою очередь символьная скорость R_s вычисляется как $R_b/log2(M)$, где R_b - битовая скорость (та скорость, с которой биты поступают на маппер), M - кол-во точек созвездия. После мапера битовая скорость переходит в символьную, которая определяет длительность прямоугольного импульса.

Схемы модуляции

BPSK

BPSK (Binary Phase Shift Keying) - двухпозиционная фазовая манипуляция. В такой схеме модуляции на 1 бит приходится 1 символ.

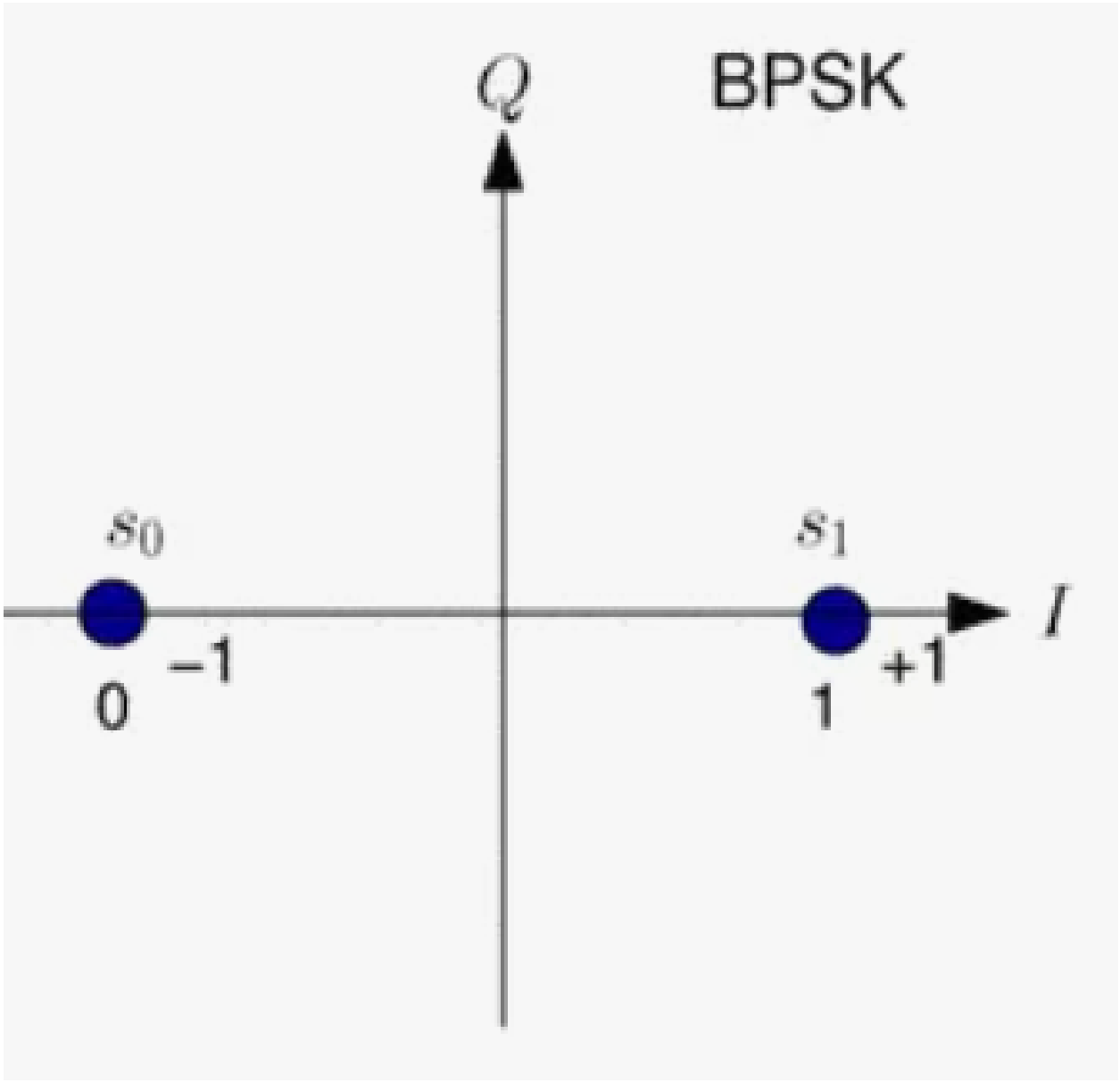

Рисунок 5 — Созвездие BPSK модуляции

Логика BPSK следующая: если бит имеет сосотяние 0, то он будет соответсвовать сигналу с начальной фазой ϕ равной 0, т.е точке с координатой (1, 0), если бит находится в сосотянии 1, то биту будет соответствовать сигнал с начальной фазой ϕ равной π,т.е точке с координатой (-1, 0). Можем заметить, что квадратурная составляющая всегда равна 0, это значит, что в сигнале будет только cos составляющая, т.к sin составляющая занулится (видно из уравнения сигнала $S_k(t) = I_k cos(2\pi f_c t) - Q_k sin(2\pi f_c t)$).

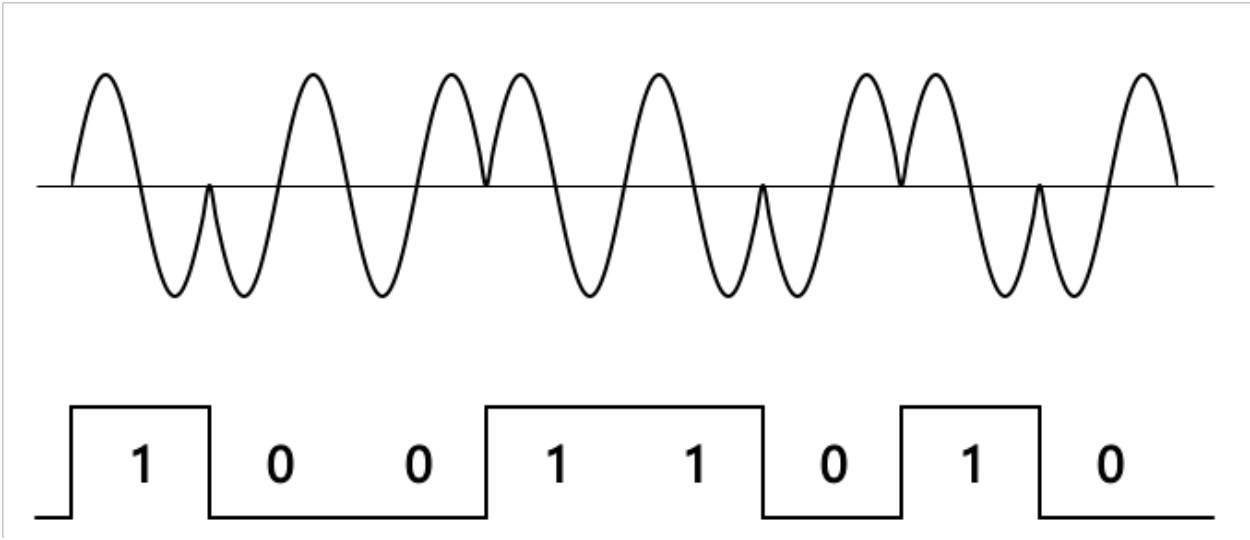

Рисунок 6 — Пример кодирования модуляцией BPSK

QPSK

QPSK (Quadrature Phase Shift Keying) - квадратурная фазовая манипуляция. 2 бита кодируются в одном символе. Используется 4 различных фазы для представления пар битов. Логика этой схемы модуляции такая же, как у BPSK, но уже добавляется квадратурные составляющие, т.е в сигнале будут как cos составляющая, так и sin составляющая. Точке (0, 0) будет соответсвовать фаза $\frac{\pi}{4}$, (1, 0) будет соответсвовать фаза $\frac{3\pi}{4}$, (1, 1) будет соответсвовать фаза $\frac{5\pi}{4}$, (0,1) будет соответсвовать фаза $\frac{7\pi}{4}$,

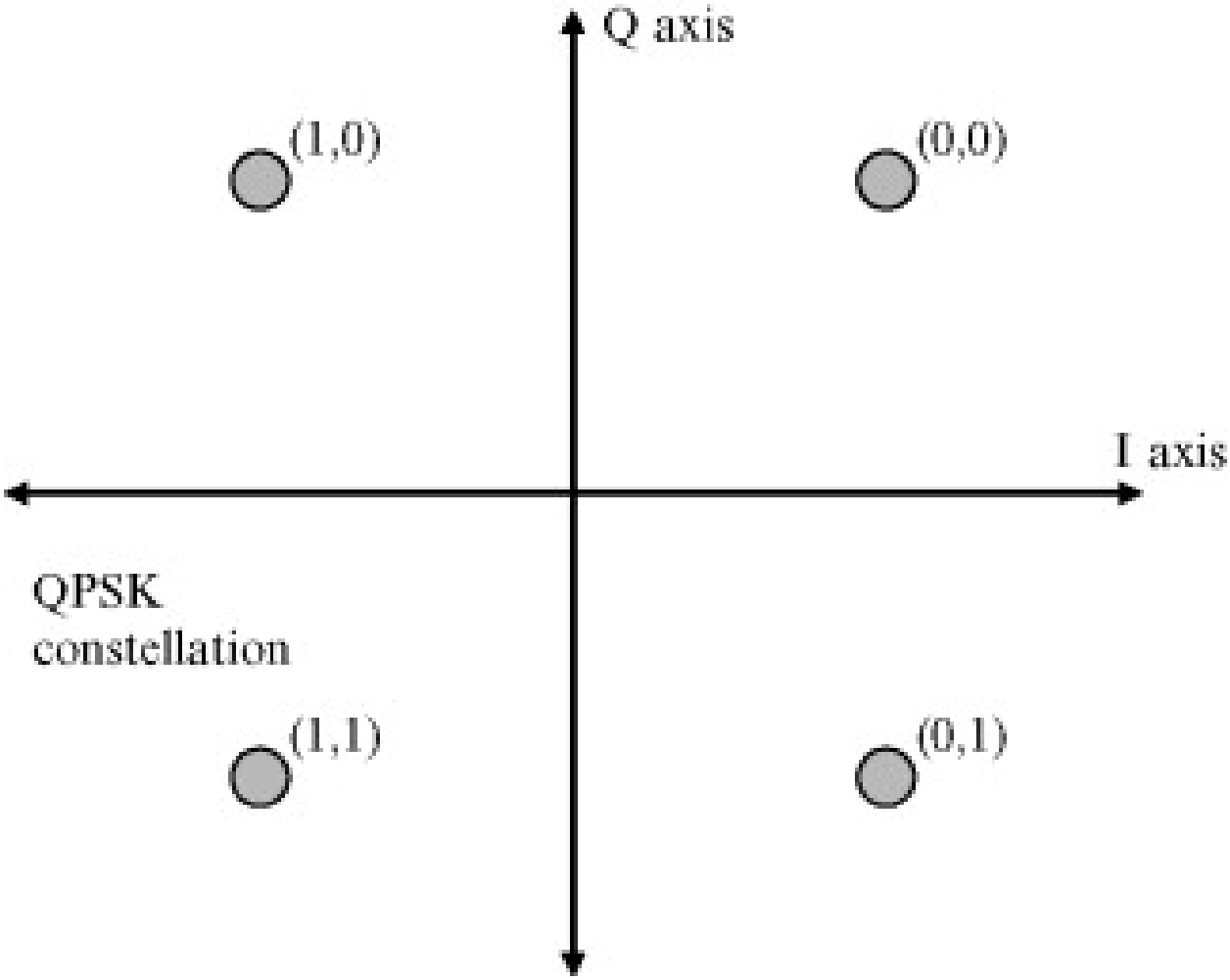

Рисунок 7 — QPSK созвездие

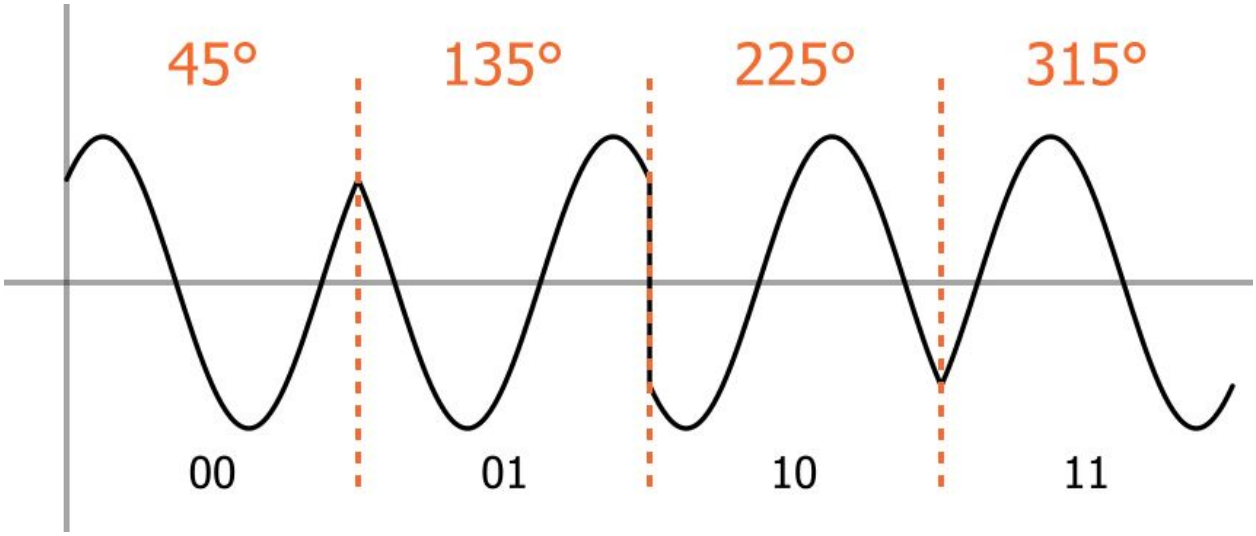

Рисунок 8 — Пример кодирования модуляцией QPSK

ПРАКТИЧЕСКАЯ ЧАСТЬ

Введение

На практике продолжаем реализовывать сигналы разной формы. На этом занятии реализую сигнал параболической формы и треугольный сигнал.

Треугольный сигнал

Я буду формировать такой сигнал по следующему принципу:

1. Если бит равен единице, то в момент длительности сигнала $\frac{\tau}{2}$ буду задавать максимальное значение, а в остлаьных случаях буду задавать нули. Таким образом в одной точке будет образовываться пик, который и будет придавать сигналу форму треугольного.
2. Если бит равен 0, то на всей длительности τ сигнал будет принимать 0.

Программная реализация

```
#define TAU 10
#define TAU_ON_ELEMENT 20

int16_t* bits_to_triangle_signal(uint8_t* bits, int
   bits_count, int tx_mtu){

// allocate memory
int16_t* tx_buff = (int16_t*)malloc(sizeof(int16_t) * tx_mtu
   * 2);

// iterate on bitts
for (int i = 0; i < bits_count; ++i)
{
    // fill tx_buff with samples
    for(int j = i*TAU_ON_ELEMENT; j < i*TAU_ON_ELEMENT + 20
       && j < tx_mtu*2; j+=2){
```

```
            if(bits[i] && j == i*TAU_ON_ELEMENT + 8){
                tx_buff[j] = 2047 << 4;     // I
                tx_buff[j+1] = -2047 << 4; // Q
            } else{
                tx_buff[j] = 0;       //I
                tx_buff[j+1] = 0;     //Q
            }
        }
    }

    return tx_buff;
}
```

Код полностью идентичен коду, который использовался для создания прямоугольного сигнала. Единственное отличие заключается в условии

```
    j == i*TAU_ON_ELEMENT + 8
```

Это условие как раз и позволяет выбрать одну конкретную точку, в которой задается максимальное значение.

Визуализируем сигнал:

Отправим сформированный сигнал и посмотрим на результат после приема.

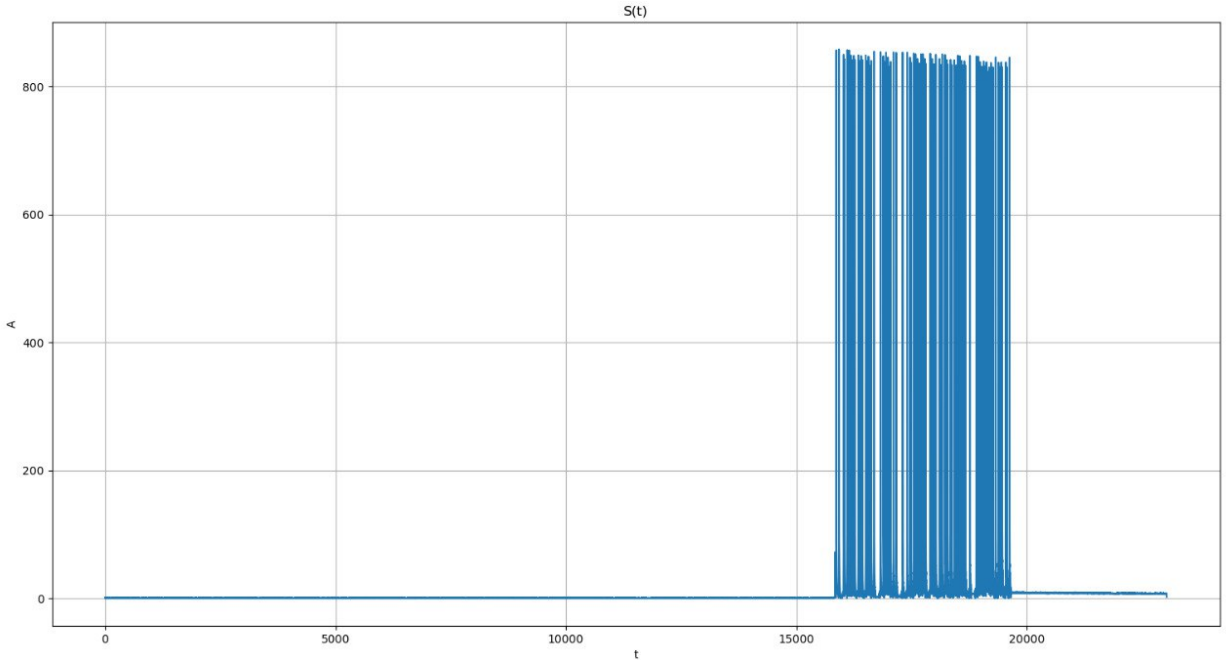

Рисунок 9 — Пример принятого треугольного сигнала

Параболический сигнал

Сформируем параболу, перебирая точки от $-\frac{tx\_mtu}{10}$ до $-\frac{tx\_mtu}{10}+2\frac{tx\_mtu}{10}$, где tx_mtu - кол-во семплов, передаваемых за раз. Деление на 10 используется для масштабирования, чтобы не возводить большие числа в квадрат.

Программная реализация

```
    int16_t* parabola_signal(int tx_mtu){

    // allocate memory
    int16_t* tx_buff = (int16_t*)malloc(sizeof(int16_t) * tx_mtu
       * 2);

    float coef = -tx_mtu / 10;

    for (int i = 0; i < 2 * tx_mtu; i+=2)
    {
        tx_buff[i] = int16_t(coef * coef);   // I
        tx_buff[i+1] = int16_t(coef * coef); // Q

        coef += 0.1;
    }

    return tx_buff;
}
```

Визуализация

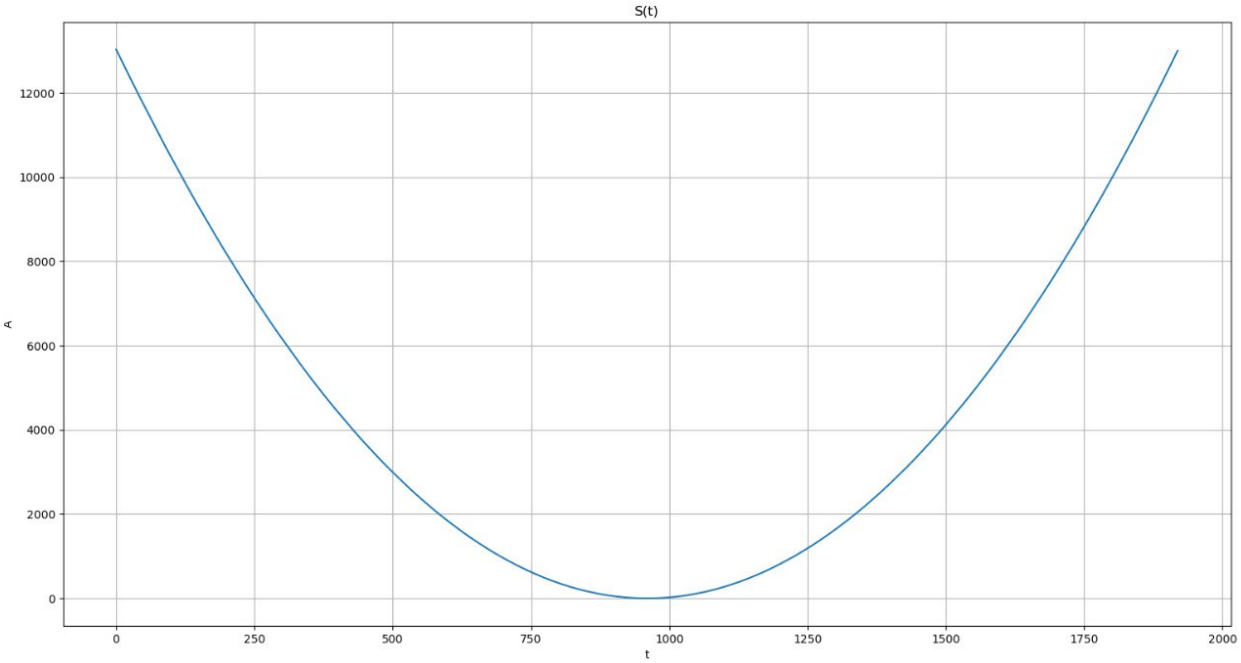

Рисунок 10 — Пример сигнала в виде параболы

Визуализация после приема

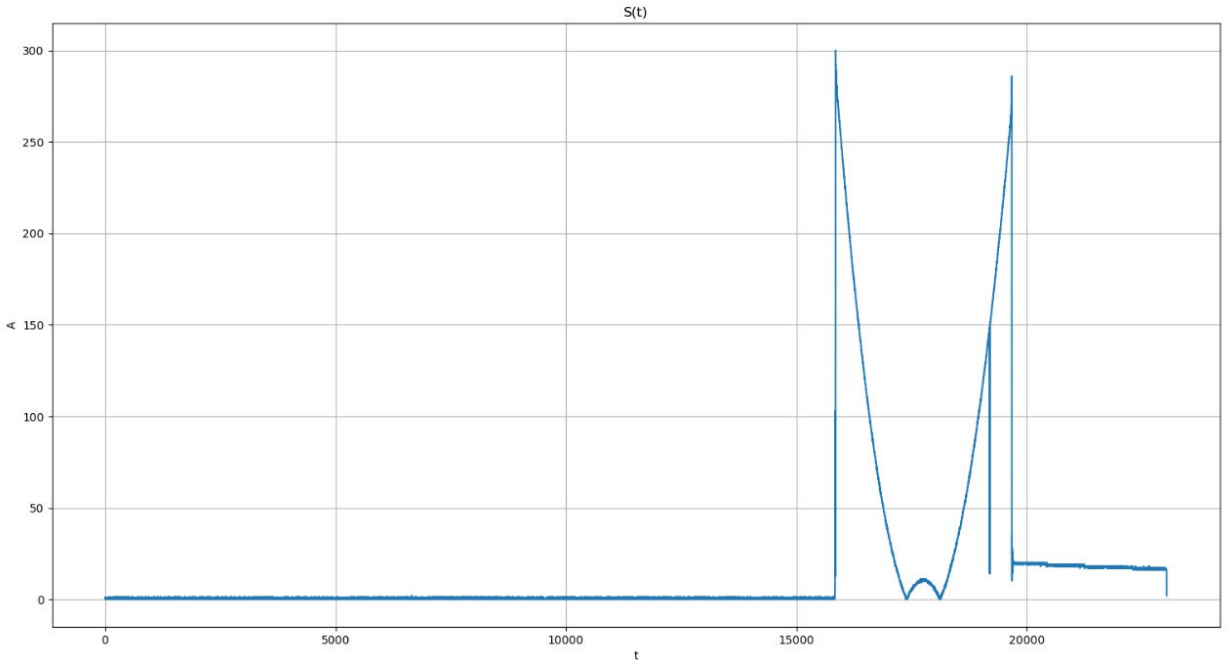

Рисунок 11 — Пример сигнала в виде параболы на приеме

Можем наблюдать параболу в нашем сигнале, по центру парабола отображена вверх, потому что я строил график модуля семплов.

Реализация BPSK модуляции

Шаги по реализации:

1. Написать BPSK модуляцию (логика маппера)
2. На основе I и Q, полученных на прошлом шаге, сгенерировать прямоугольные импульсы (логика формирующего фильтра)
3. Перемножить прямоугольные импульсы на несущее колебание
4. Проанализировать полученные результаты и сделать вывод

BPSK модулятор

Реализация на языке C:

```
double* BPSK_modulation(int* bits, int bits_count){
    //allocate memory
    double* IQ_samples = (double*)malloc(sizeof(double) *
       bits_count * 2);
    //iterate on bits
    for(int i,j = 0; i < bits_count * 2; i+=2){
        if(bits[j]){
            IQ_samples[i] = 1;      // I
            IQ_samples[i + 1] = 0;  // Q
        }else{
            IQ_samples[i] = -1;     // I
            IQ_samples[i + 1] = 0;  // Q
        }

        ++j;
    }

    return IQ_samples;
}
```

Сначала выделяем память под IQ семплы, т.к на каждый бит приходится 1 семпл, а на 1 семпл - 2 числа (I и Q), то размер массива с семплами должен быть вдвое больше кол-ва бит. Далее перебираем биты и в зависимости от его значения формируем семпл. В результирующем массиве I и Q идут в строгой последовательности I_0,Q_0,I_1,Q_1,...,I_N,Q_N. Массив в дальнейшем запишется в файл.

Формирующий фильтр

На языке Python парсим файл с семплами из прошлого шага. Далее формируем прямоугольные импульсы. Визуализация импульсов:

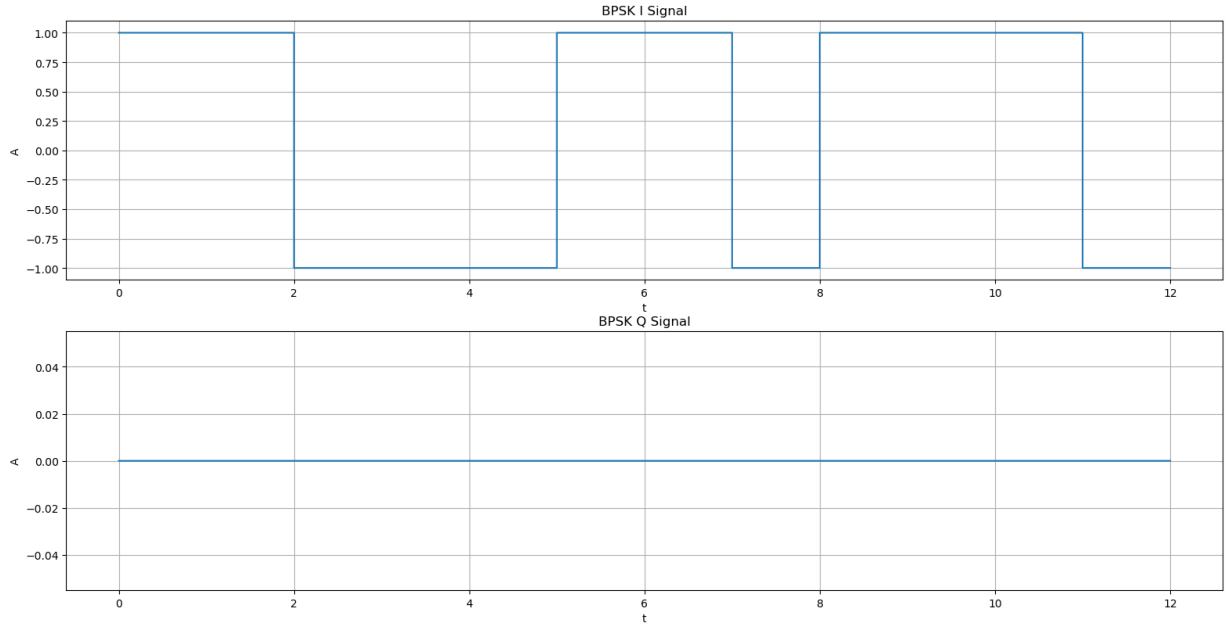

Рисунок 12 — Символы

Здесь явно видно, что в BPSK модуляции квадратурная составляющая всегда равна 0. Я передавал последовательность в 12 бит и это заняло 12 секунд, т.е на каждый бит приходился 1 символ, который длился 1 секунду.

Перемножение с несущим колебанием

Теперь перемножим символы с несущим колебанием. В роли несущего колебания я использовал $cos(2\pi ft)$ и $-sin(2\pi ft)$ с частотой $f = 1$. В реальности частоты несущих колебаний в разы больше, но для наглядности я взял маленькое значение. Этот процесс можно сравнить с наложением маски, где маской является прямоугольный сигнал. Если посмотреть на схему цифровой системы связи, которая была дана в разделе с теорией, то этот процесс происходит в цепи, которая следует после формирующего фильтра. Визуализация результата:

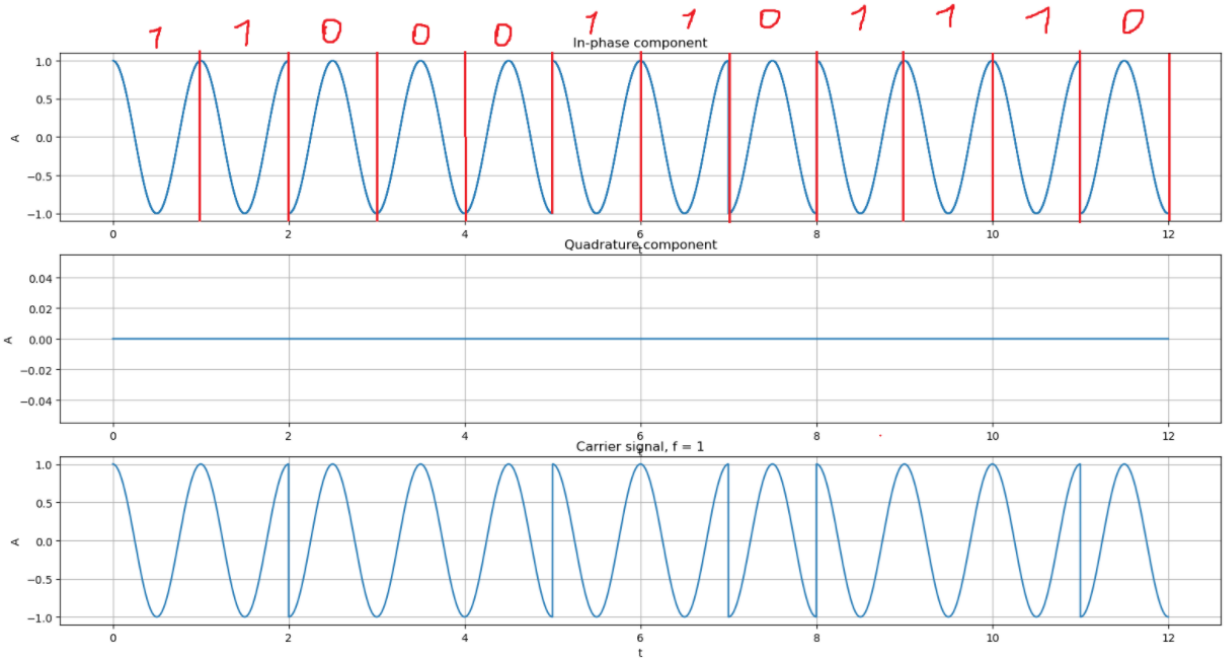

Рисунок 13 — Перемножение с несущим сигналом

Можем видеть, что квадратурная компонента по прежнему равна 0, т.е в сигнале присутствует только cos. В синфазной компоненте можем видеть, как меняется фаза колебания, которая отражает значение бита. Если фаза равна 0, то передается 1, если фаза равна π, то передается 0. В местах, где меняется фаза, происходит смена значения бита. Таким образом можно восстановить передаваемую битовую последовательность. Результирующий сигнал будет иметь вид синфазной компоненты, т.к квадратурная всегда равна 0.

Реализация QPSK модуляции

Шаги по реализации будут в точности такие же, как у BPSK модуляции, но слегка будет отличаться реализация.

Реализация на языке C:

```
double* QPSK_modulation(int* bits, int bits_count){
    //allocate memory
    double* IQ_samples = (double*)malloc(sizeof(double) *
       bits_count);
    //iterate on bits
    for(int i = 0; i < bits_count; i+=2){
        if(bits[i]){
            IQ_samples[i] = -1;           //I
```

```
            if(bits[i + 1]){
                IQ_samples[i + 1] = -1;   //Q
            }else{
                IQ_samples[i + 1] = 1; // Q
            }
        }else{
            IQ_samples[i] = 1;          //I
            if(bits[i + 1]){
                IQ_samples[i + 1] = 1;   //Q
            }else{
                IQ_samples[i + 1] = -1; //Q
            }
        }
    }


    return IQ_samples;
}
```

Заметим, что в случае QPSK модуляции массив под семплы вдвое меньше, т.к в QPSK модуляции на 1 семпл приходится 2 бита.

Формирующий фильтр

Визуализация импульсов:

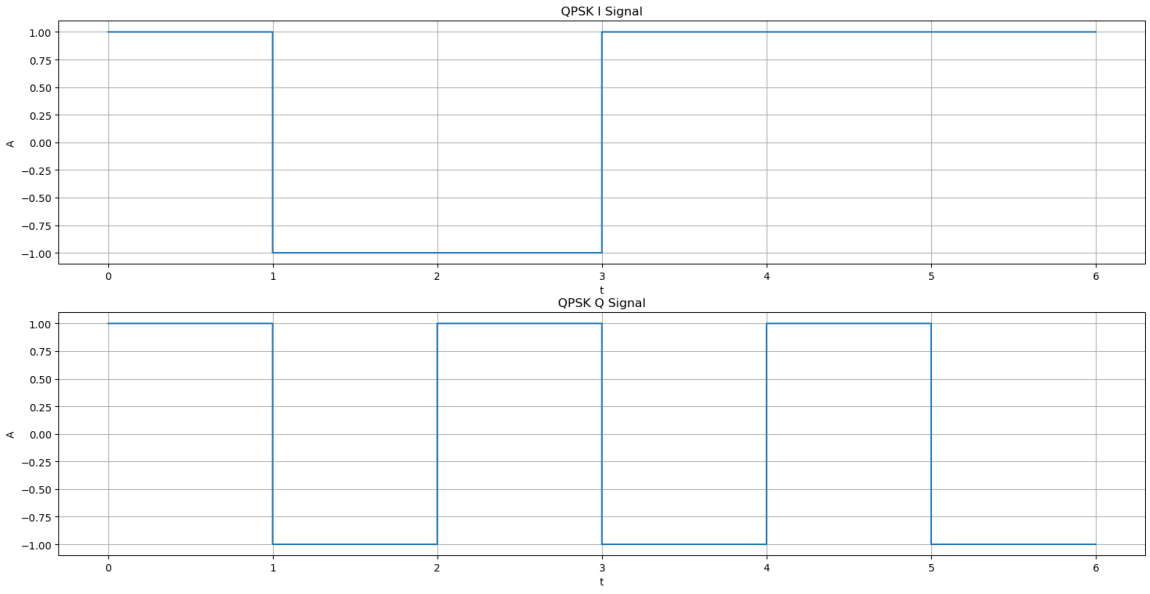

Рисунок 14 — Символы

Можем заметить, что появляется квадратурная часть. Также можно заметить, что та же последовательность из 12 бит передается уже за 6 секунд. Это связано с тем, что в QPSK на семпл приходится 2 бита. Можно сделать вывод о том, что чем больше точек в сигнальном созвездии, тем выше скорость передачи данных. 1 символ длится уже не 1 секунду, а 0.5 секунд (на графике чуть неверный масштаб)

Перемножение с несущим колебанием

Визуализация результата:

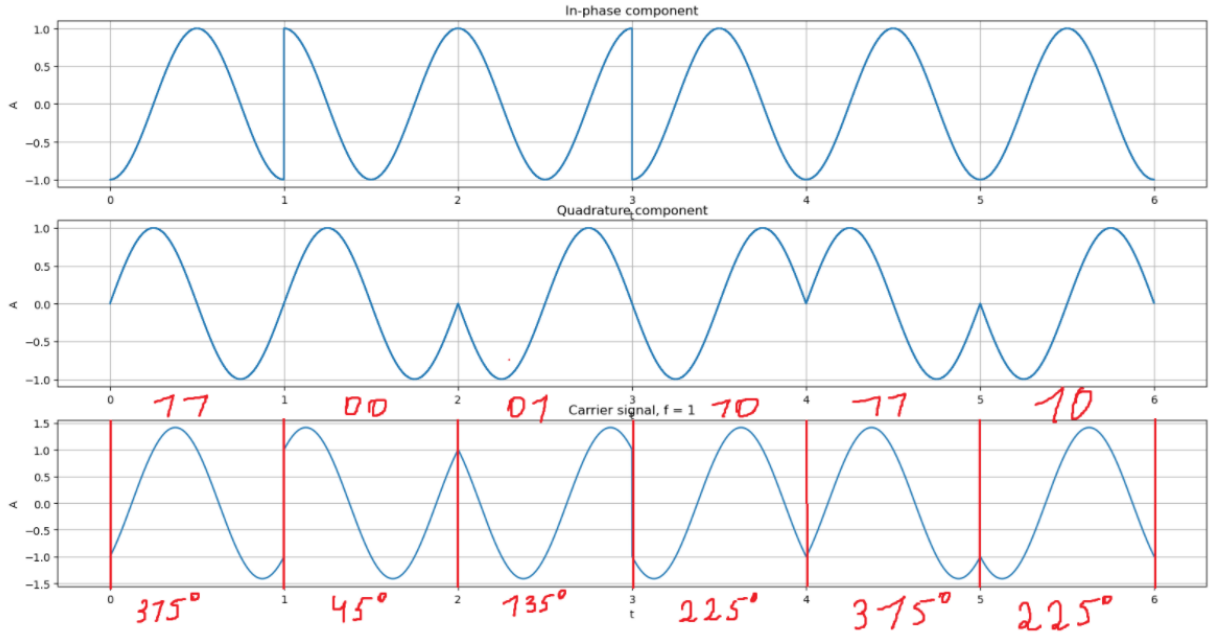

Рисунок 15 — Перемножение с несущим сигналом

Можем видеть, что квадратурная компонента теперь не равна 0, т.е в сигнале присутствует cos и sin. Еще можно заметить, что максимальная амплитуда в результирующем сигнале уже не равна 1, а равна $\sqrt{I^2 + Q^2} = \sqrt{1^2 + 1^2} = \sqrt{2}$. Результирующий сигнал будет иметь вид $Icos(2\pi f_c t) - Qsin(2\pi f_c t)$.

ВЫВОД

В ходе проделанной работы я реализовал сигналы разной формы, отправил их в радиоканал, потом принял и визуализировал форму сигнала. Также реализовал BPSK и QPSK модуляцию.