Федеральный государственный автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт инженерной физики и радиоэлектроники
Радиотехника

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Саломатов Ю.П.
подпись
«____» _____ 20___ г.

БАКАЛАВРСКАЯ РАБОТА

11.03.01 – Радиотехника

Разработка лабораторного макета для изучения навигационных систем

Руководитель _____ профессор, д-р, ф-м н. С.П. Царев
подпись, дата должность, ученая степень инициалы, фамилия

Науч. Консультант _____ доцент, канд. т. н. А.С. Пустошилов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Д. Р. Соловьев
подпись, дата инициалы, фамилия

Красноярск 2023

РЕФЕРАТ

Дипломная работа по теме «Разработка лабораторного макета для изучения навигационных систем» содержит 64 страницs текстового документа, 41 рисунок, 4 формулы, 9 используемых источников.

Работа включает в себя документацию, которая содержит:

- введение, где описываются цели и задачи проекта;
- анализ функциональных и нефункциональные требований к макету;
- теоретическую часть исследования;
- описание архитектуры и компонентов учебного макета;
- описание разработанных алгоритмов;
- результаты и их анализ;
- заключение, где подводятся итоги работы и делаются выводы;
- список литературы, включающий использованные источники информации и стандарты;
- приложения, содержащие код разработанных алгоритмов.

Содержание

ВВЕДЕНИЕ

Лабораторный макет для изучения навигационных систем позволит изучить принцип, по которому определяется расстояние между спутником и объектом в Глобальных Навигационных Спутниковых Системах.

Цель работы: разработать устройство, использующее звук для определения расстояния, измеряя временную задержку между отправленным и принятым сигналами. Также необходимо написать программу, рассчитывающую координаты приемника по полученным с помощью устройства данным.

Для выполнения поставленной цели были поставлены следующие задачи:

− выделить наиболее подходящие для реализации макета кодовые последовательности;

− разработать алгоритмы обработки звуковых сигналов для определения задержки звука. Цель состоит в том, чтобы учебный макет точно и надежно определял задержку между сигналами, излучаемыми источниками звука, для последующего расчета расстояния.

− разработать алгоритмы вычисления координат на основе полученного расстояния.

− разработать функционал для построения графика на основе найденных значений, что поможет в понимании и анализе данных.

− макетирование устройства. Важно определить и интегрировать необходимые компоненты, такие как микрофон, динамики, отладочную плату. Это позволит учебному макету полноценно функционировать и выполнять требуемые измерения.

Актуальность разработки состоит в том, что лабораторный стенд предоставит студентам и школьникам возможность визуального изучения работы навигационных систем. Он позволит экспериментировать с различными кодовыми последовательностями и анализировать их влияние на точность и скорость измерений.

Использование стенда позволит обучающимся применить теоретические знания, полученные во время изучения навигационных систем, на практике. Они могут проводить эксперименты, анализировать полученные данные и сравнивать их с ожидаемыми результатами. Такой практический опыт способствует более глубокому пониманию материала и развитию навыков анализа и решения проблем.

Реализация учебного макета навигационной системы в слышимом диапазоне не имеет аналогов, а весь код находится в открытом доступе и может быть доработан под нужды учебного процесса, что делает разработку лабораторного стенда гибкой и адаптируемой. Преподаватели и студенты могут модифицировать код и настраивать параметры экспериментов в соответствии с конкретными целями и требованиями курса. Это позволяет лучше интегрировать лабораторный стенд в образовательную программу и настраивать его под конкретные потребности исследований.

АНАЛИЗ ТРЕБОВАНИЙ К МАКЕТУ

1. Требования к функциональности. Разработанный макет должен осуществлять:

− измерение расстояния по задержке звука между динамиками и микрофоном;

− вычисление координат приемника на основе замеров задержки звука;

− построение графика, отображающего изменение координат источника звука в реальном времени;

2. Требования к аппаратной части:

− наличие микрофона и динамиков, расположенных в разных точках для измерения задержки звука. Чувствительность микрофона и громкость динамиков должны быть достаточными для точного измерения задержки звука;

− поддержка аналогового и цифрового ввода/вывода микроконтроллером для передачи данных с микрофона и на динамики;

3. Требования к программному обеспечению:

− разработанные алгоритмы должны обеспечить достаточную точность для определения координаты искомой точки и последующего построения графика;

− поддержка передачи данных между устройством и компьютером посредством серийного интерфейса UART;

Техническое задание представляет собой основу для разработки учебного макета и помогает определить требования к его функциональности, аппаратной части и программному обеспечению.

Раздел 1. Теоретическая часть

1.1 Принцип работы системы:

Сигнал со спутников непрерывно поступает в терминал, программный комплекс которого анализирует время задержки для разных спутников. На основе анализа ответной информации определяются координаты объекта, на котором установлено приемное оборудование. [4] Пример системы приведен на рисунке 1

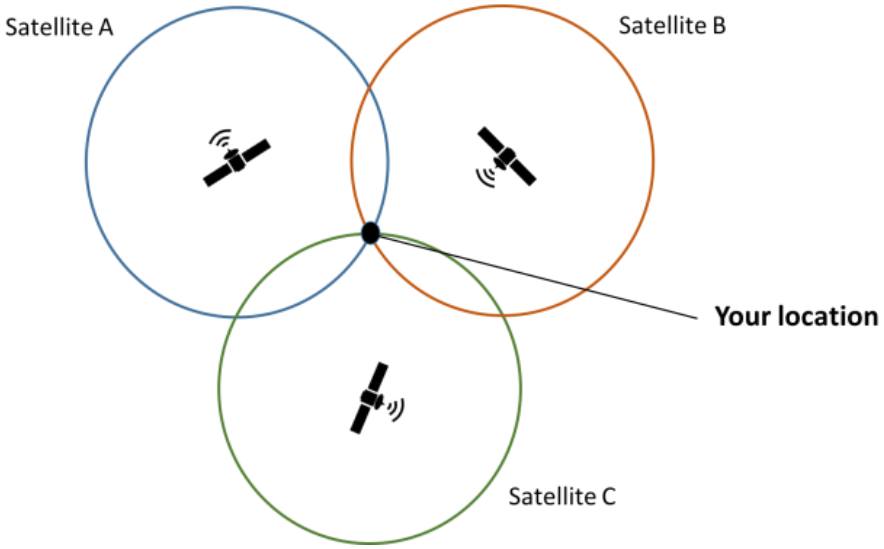

Рисунок 1 – Пример работы навигационной системы

При постоянной работе терминала система ГЛОНАСС может определять не только положение, но и скорость движения объекта. При движении точность позиционирования снижается, но все равно остается достаточной для того, чтобы навигационное оборудования могло выполнить привязку координат объекта к электронной карте местности и построить маршрут. [1,2]

Глобальная навигационная спутниковая система (Global Navigation Satellite System – ГНСС) – это спутниковые системы (наиболее распространены GPS и ГЛОНАСС), используемые для определения местоположения в любой точке земной поверхности с применением специальных навигационных или геодезических приемников. ГНСС-технология нашла широкое применение в геодезии, городском и земельном кадастре, при инвентаризации земель, строительстве инженерных сооружений, в геологии и т.д.

ГЛОНАСС – это российская разработка, которая обеспечивает точное позиционирование объекта в пространстве с минимальной погрешностью. Для определения координат используется специальное оборудование, которое при поддержке наземной инфраструктуры связывается с сетью спутников, выведенных на околоземную орбиту.

Основой системы ГЛОНАСС являются 24 космических аппарата, которые движутся в трёх орбитальных плоскостях по 8 аппаратов в каждой плоскости, наклоненных к экватору под углом 64,8°, с высотой орбит 19100 км и периодом обращения 11 ч 15 мин 44 с. Выбранная структура орбитальной группировки обеспечивает движение всех космических аппаратов по единой трассе на поверхности Земли с ее повторяемостью через 8 суток. Такие характеристики обеспечивают высокую устойчивость орбитальной группировки системы ГЛОНАСС, что практически позволяет обходиться без коррекции орбит космических аппаратов в течение всего срока их активного существования. [4]

Для определения расстояния в навигационных системах применяют дальномерные коды (псевдослучайные последовательности).[1]

В случае учебного макета терминалом является микрофон, а динамики – выполняют роль спутников, относительно которых и будет определятся координата микрофона. Блок-схема работы лабораторного стенда представлена на рисунке 2.

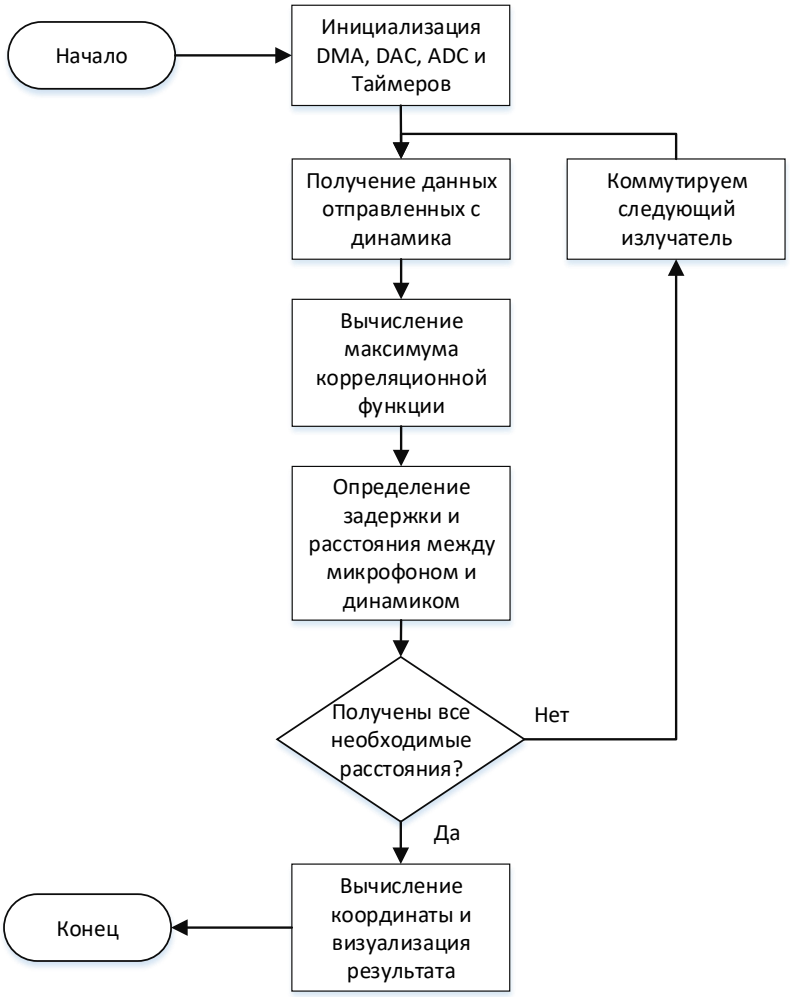

Рисунок 2 – Блок-схема работы лабораторного стенда

1.2 Псевдослучайные последовательности:

Псевдослучайные последовательности (ПСП) играют важную роль в навигации, особенно в глобальных навигационных спутниковых системах (ГНСС) таких, как GPS и ГЛОНАСС. Эти системы используют сигналы, создаваемые спутниками, для определения местоположения приемника на Земле.

Каждый спутник ГНСС генерирует сигналы, которые содержат коды, известные как кодовые последовательности. Коды представляют собой бинарные последовательности нулей и единиц, которые кажутся случайными. Однако эти последовательности скорее являются псевдослучайными, то есть они создаются с помощью детерминированных алгоритмов, которые могут генерировать такие последовательности.

Приемник ГНСС использует сигналы от нескольких спутников для определения своего местоположения. Приемник сравнивает фазу сигнала, получаемого от спутника, с фазой сигнала, создаваемого внутренней кодовой последовательностью приемника, чтобы определить эту разницу.

ПСП также используются для устранения ошибок измерения в ГНСС. Измерения могут быть искажены различными факторами, такими как многолучевое распространение и эффекты атмосферы. Однако, если заранее известны кодовые последовательности, сгенерированные спутником, и коды, сгенерированные приемником, то приемник может устранить эти ошибки и получить более точное измерение расстояния до спутника.

В целом, псевдослучайные последовательности очень важны для навигации и ГНСС. Они позволяют определять расстояние до спутников и устранять ошибки измерений, что делает системы ГНСС более точными и надежными. Для передачи псевдослучайной последовательности используется фазовая манипуляция.

Фазовая манипуляция (ФМ) – это вид модуляции, в котором информация кодируется изменением фазы несущего сигнала. При ФМ используются различные дискретные значения фазы, каждое из которых представляет определенное состояние символа или бита данных. [7,8]

Рассмотрим некоторые из них:

М-последовательности

М-последовательности, или последовательности максимальной длины, являются особыми фазоманипулированными сигналами. Они имеют ряд характеристик, которые делают их полезными в навигационных системах:

1. Периодичность: М-последовательность является периодической с периодом, состоящим из N импульсов или символов.

2. Боковые пики автокорреляционной функции М-последовательностей равны -1/N.

3. Псевдослучайность: М-последовательность в общем случае состоит из нескольких видов импульсов, которые распределены в периоде равновероятно. Это делает М-последовательности псевдослучайными, поскольку они имеют свойства случайных последовательностей. Это особенно полезно в навигационных системах, где требуется иметь сложную, но предсказуемую последовательность для различения и идентификации сигналов.

4. Малые боковые пики: при усечении М-последовательности, то есть взятии непериодической последовательности длиной в период N, боковые пики автокорреляционной функции приближаются к $1/\sqrt{N}$. Это означает, что при увеличении N величина боковых пиков уменьшается, что делает М-последовательности более эффективными.

Из-за перечисленных свойств М-последовательности они широко используются в навигационных системах. Например, системы GPS и

ГЛОНАСС используют М-последовательности для идентификации спутниковых сигналов. [8]

М-последовательности обладают хорошими автокорреляционными характеристиками, но их использование в качестве псевдослучайной последовательности для лабораторного макета нежелательно по следующим причинам:

− Долгая обработка данных: М-последовательности обычно имеют длину, состоящую из множества символов. Использование такой длинной последовательности может привести к неэффективному использованию ресурсов устройства и требовать больше вычислительной мощности

− Переполнение памяти микроконтроллера: Использование М-последовательностей может требовать значительного объема памяти для хранения этих последовательностей и выполнения вычислений с ними. Ограниченные ресурсы памяти микроконтроллера могут повлиять на функциональность учебного макета.

Код Баркера

Код Баркера, также известный как последовательность Баркера, является одним из популярных типов псевдослучайных последовательностей, которые могут применяться в лабораторных макетах с участием радиосвязи или навигации. Вот некоторые преимущества его использования:

1. Код Баркера имеет относительно небольшое количество элементов или символов по сравнению с другими типами псевдослучайных последовательностей. При использовании занимают меньше памяти и требуют меньше вычислительных ресурсов для их обработки. На рисунке 3 изображена временная диаграмма сигнала, модулированного кодовой последовательностью Баркера из пяти символов.

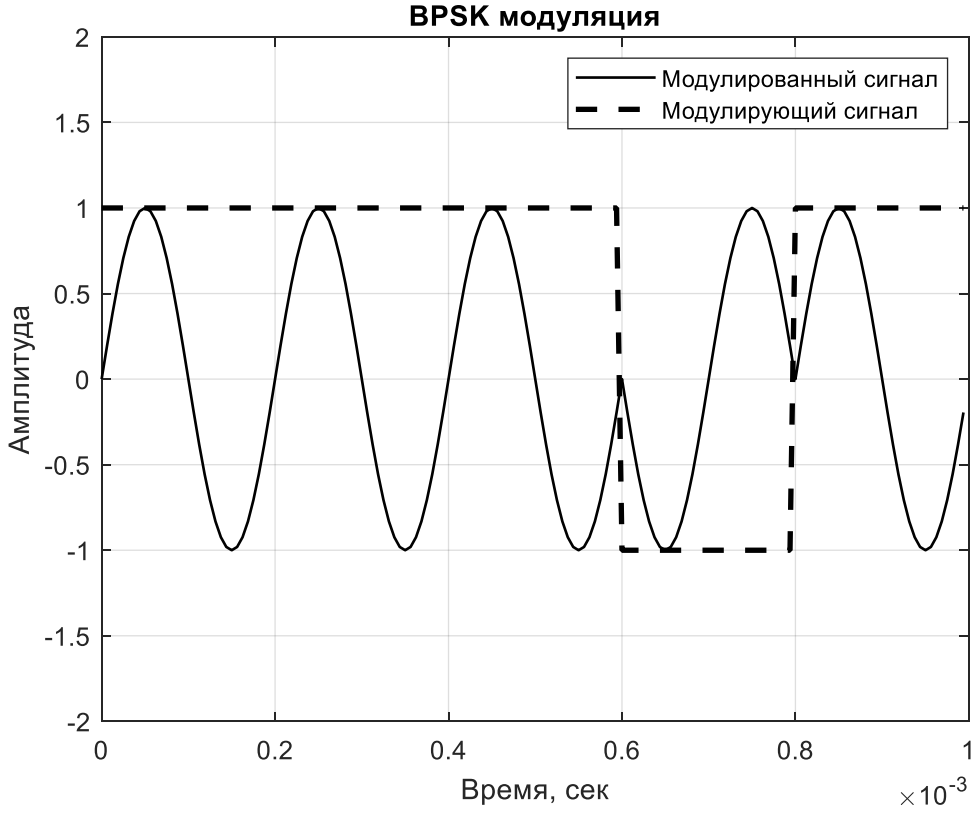

Рисунок 3 – Временная диаграмма сигнала, модулированного кодовой последовательностью Баркера

2. Код Баркера обладает отличными автокорреляционными свойствами. Это значит, что при сравнении принятого сигнала с оригинальным можно точно определить задержку сигнала. Последовательности Баркера имеют минимальный уровень боковых лепестков автокорреляционной функции $1/N$. На рисунке 4 показана автокорреляционная характеристика сигнала, модулированного кодовой последовательностью Баркера.

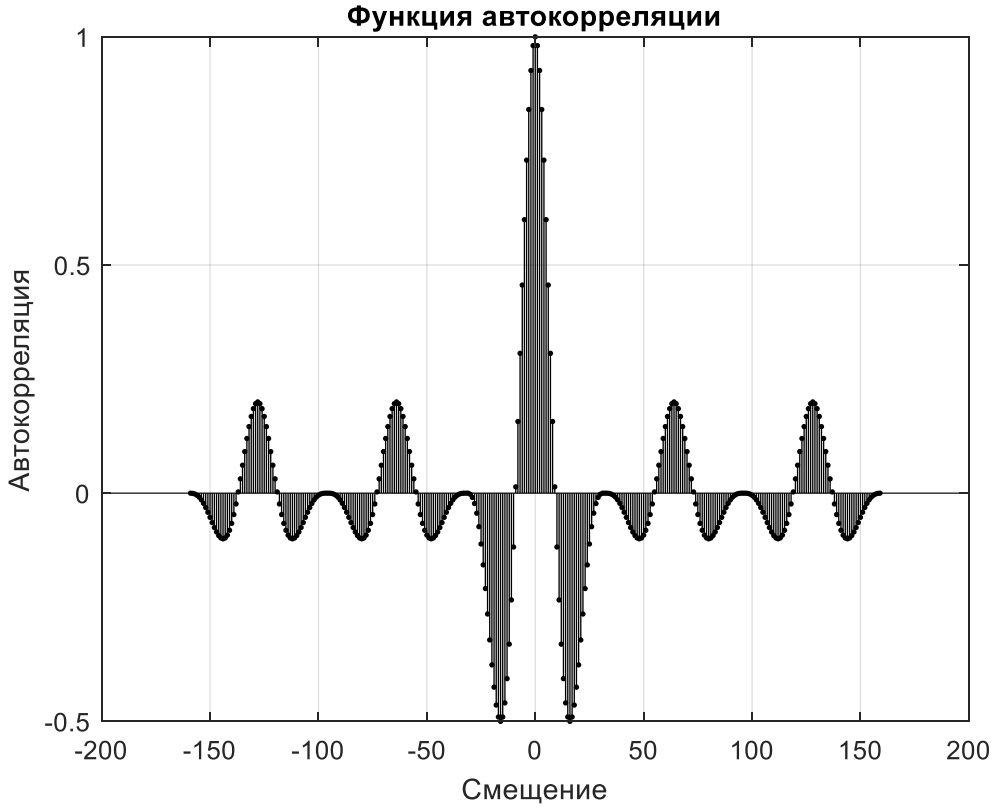

Рисунок 4 – Автокорреляционная характеристика сигнала, модулированного кодовой последовательностью Баркера

3. Устойчивость к помехам: Коды Баркера обладают хорошей способностью сопротивляться помехам. Их структура позволяет легко отличать их от других сигналов или шума. Это особенно полезно в условиях с шумом или многолучевым распространением сигнала. [7]

Код Баркера представляет собой компактную и эффективную псевдослучайную последовательность и использование для определения расстояния сигнала упрощает реализацию системы, а низкий уровень боковых лепестков автокорреляционной функции позволит точнее определять задержку.

Помимо Баркера и М-последовательностей, в радиосвязи и навигации широко применяются следующие типы псевдослучайных последовательностей:

1. Голдовские последовательности: Голдовские последовательности получаются путем комбинирования двух М-последовательностей с использованием операции XOR. Они обладают хорошими корреляционными свойствами и используются в системах CDMA (Code Division Multiple Access), таких как GSM и GPS.

2. Каскадные последовательности: Каскадные последовательности, также известные как Соломоновские последовательности, являются комбинацией нескольких последовательностей различной длины, полученных из семейства Фибоначчи или других базовых последовательностей. Они используются в радиолокации, системах связи и других приложениях.

3. Шумовые последовательности: Шумовые последовательности, также известные как псевдослучайные шумы, генерируются с использованием статистических алгоритмов или физических источников шума. Они обладают случайными свойствами и широко используются в системах радиосвязи для модуляции и демодуляции сигналов.

4. Коды Хэдемарка: Коды Хэдемарка являются математическими конструкциями, представляющими собой матрицы с определенными свойствами. Они используются в радиолокации, спектральном разделении сигналов и других приложениях, где требуется устойчивость к помехам и хорошая спектральная эффективность.[8]

Это лишь некоторые из популярных типов псевдослучайных последовательностей, используемых в радиосвязи и навигации. Каждый тип последовательности имеет свои особенности и применения в различных системах и технологиях.

1.3 ЦАП, АЦП и DMA микроконтроллера STM32:

Для передачи сгенерированного сигнала на динамик необходимо регулярно преобразовывать его в аналоговый вид, желательно не нагружая при этом микроконтроллер. С этой задачей отлично справится встроенный в STM32 12-ти битный ЦАП, который, кроме того, может быть использован в сочетании с блоком прямого доступа к памяти DMA.

Цифроаналоговый преобразователь (ЦАП) представляет собой устройство, позволяющее получить аналоговый сигнал необходимой формы из соответствующего цифрового кода. Фактически он производит обратную операцию, выполняемую аналогоцифровым преобразователем (АЦП). ЦАП и АЦП являются интерфейсами между дискретным цифровым миром и аналоговыми сигналами. Любой ЦАП характеризуется разрядностью, производительностью и динамическим диапазоном.[5]

Структурная схема ЦАП представлена на рисунке 5:

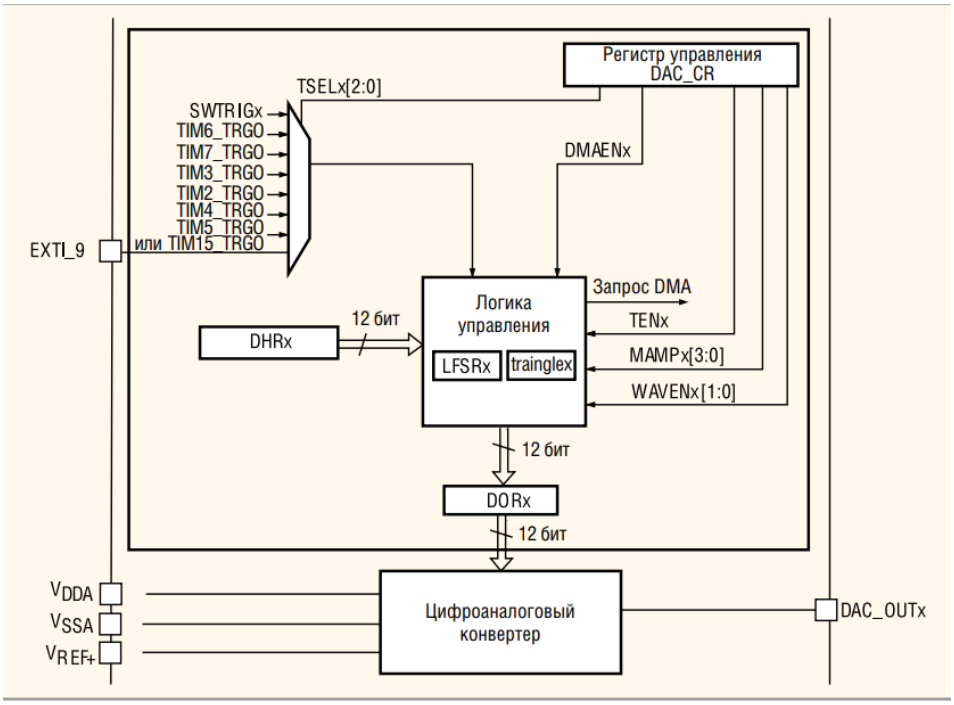

Рисунок 5 – Структурная схема ЦАП

Цифроаналоговый преобразователь микроконтроллеров серии STM32 представляет собой 12-разрядный преобразователь цифровых данных в выходное напряжение от 0 В до опорного напряжения Vref+. ЦАП поддерживает как 12-разрядный режим, так и 8-разрядный.

ЦАП имеет два канала, каждый из которых содержит независимый преобразователь. Канал 1 подключён к выводу PA4, а канал 2 – к выводу PA5. В двухканальном режиме преобразование может выполняться независимо или одновременно, когда оба канала группируются для синхронного запуска. В 12-разрядном режиме данные могут выравниваться по правому или по левому краю 16-разрядных слов.

Запуск преобразования возможен программно либо от внешних источников. В качестве таких источников запуска могут служить таймеры или внешний вход EXTI_9. Вход опорного напряжения Vref+ является общим с блоком АЦП. [5]

DMA (Direct Memory Access – прямой доступ к памяти) – позволяет передавать данные без участия ядра. [6]

На рисунке 6 изображена структурная схема каналов DMA и ее контролера:

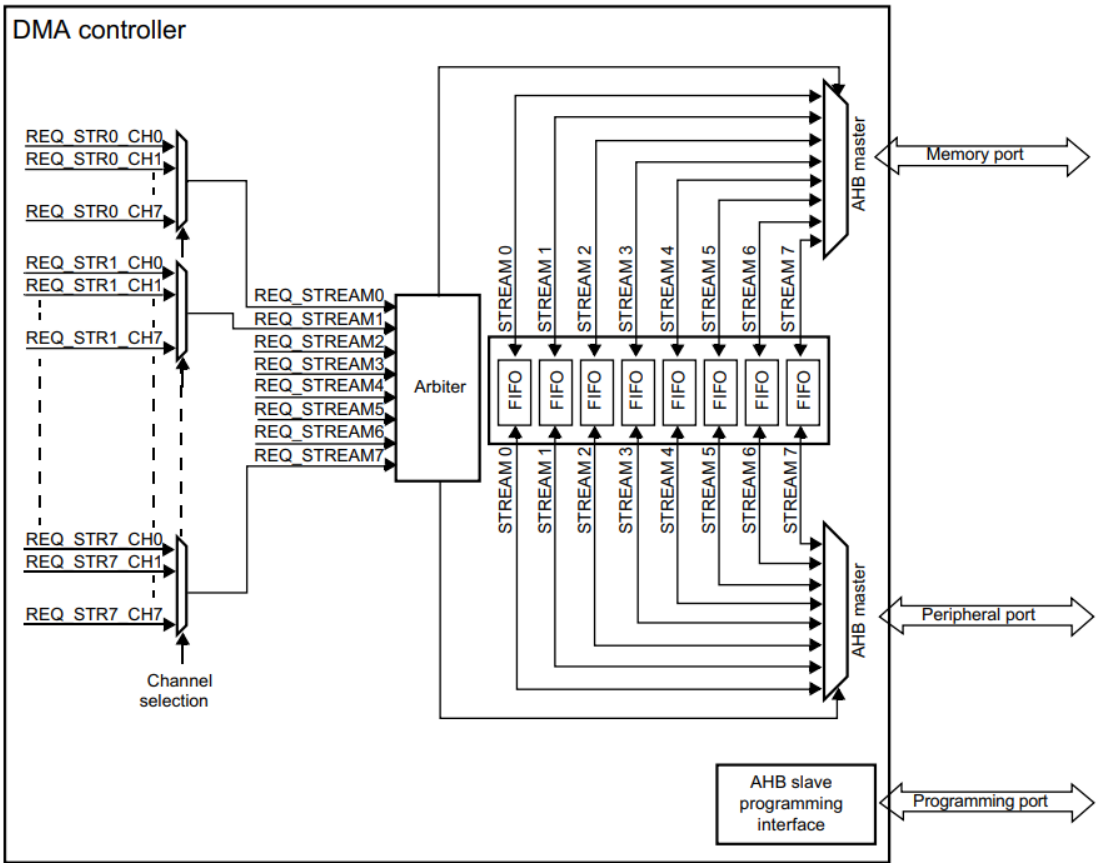

Рисунок 6 – Структурная схема контроллера DMA

Встроенный в микроконтроллеры STM32 цифроаналоговый преобразователь (DAC) может работать по сигналам таймера и получать данные напрямую из массива памяти через DMA. Таким образом, можно сконфигурировать контроллер так, что DAC будет работать без участия программы, не затрачивая ресурсы процессора, за исключением инициализации системы. [5,6]

Этого достаточно для реализации BPSK модуляции не нагружая контроллер лишними операциями.

Прием сигнала в проекте осуществляется с помощью модуля микрофона. Для обработки принятого аналогового сигнала необходимо преобразовать его обратно в цифровой.

12-ти разрядный АЦП (аналого-цифровой преобразователь) микроконтроллера STM32 относится к встроенной аналоговой подсистеме, которая предназначена для преобразования аналоговых сигналов в цифровой формат. На функциональной схеме АЦП (Рисунок 7) показаны основные компоненты модуля.

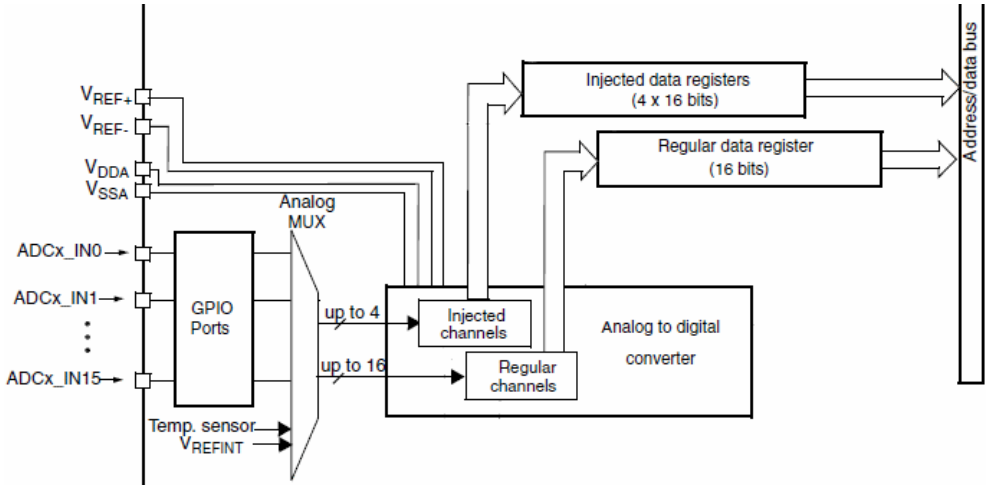

Рисунок 7 – Структурная схема модуля АЦП

В модуле АЦП имеется один встроенный аналого-цифровой преобразователь. Через коммутатор Analog MUX на его вход можно подключать аналоговые сигналы с выводов микроконтроллера IN0...IN15. Существует определенная логика работы модуля, по которой АЦП автоматически переключает коммутатор и запускает преобразования.

Опрос каналов АЦП осуществляется по логике, которая разделяет их на два типа: регулярные и инжектированные. Оба типа каналов работают с общими аналоговыми входами микроконтроллера, но отличаются приоритетом, последовательностью и способом инициализации опроса.

Для регулярных каналов формируется список последовательности работы в специальных регистрах. Согласно этому списку, АЦП поочередно опрашивает регулярные каналы. Порядок опроса не имеет значения, и до 16 регулярных преобразований могут быть выполнены. Один и тот же канал может быть опрошен несколько раз в одной последовательности. Последовательности могут вызываться циклически, то есть непрерывно.

Результаты преобразования регулярных каналов сохраняются в одном 16-разрядном регистре данных (Regular data register). Чтобы избежать потери данных, результат должен быть считан из регистра до завершения следующего преобразования.

У инжектированных каналов максимальное количество измерений в цикле составляет 4. Для сохранения результатов преобразования каждого канала используются отдельные 16-разрядные регистры данных (Injected data registers). Всего имеется 4 таких регистра.

В проекте входные данные необходимо принимать регулярно с определенной периодичностью, поэтому был выбран регулярный тип опроса одного канала по событию от таймера.

Раздел 2. Реализация навигационной системы с помощью микроконтроллера

2.1 Выбор сигнала для измерения расстояния

Во время работы над макетом возникла необходимость в исследовании корреляционных свойств сигнала, модулированного последовательностью Баркера, с помощью которого мы определяем задержку, а также в изменении его параметров.

По причине экономии места, отчеты излучаемого сигнала хранятся в постоянной памяти и во время работы устройства не изменяются, было решено записать их в массив констант при загрузке прошивки в устройство. Чтобы избежать ручного формирования массива, была написана специальная программа на языке MATLAB (код программы прикреплен в Приложении А).

Перед запуском программы, необходимо написать параметры требуемого сигнала в соответствующие переменные (Рисунок 8):

```
% кодируемая последовательность
code = barker2;
% количество отсчетов одного периода синуса
m = 12;
% длительность импульса в периодах синусойды
num_of_periods_per_bit = 1;
% частота сигнала в Гц
fc = 5000;
```

Рисунок 8 – Параметры для настройки генерируемого с помощью программы сигнала

Во время выполнения кода программа рассчитает отчеты синусоидального сигнала, сформирует модулирующий сигнал и, путем перемножения отсчетов, смоделирует требуемый фазово-манипулированный

сигнал. Полученный сигнал и его параметры записываются в текстовый файл. Предварительно амплитуда сигнала домножается на 2048 (половина от максимального значения 12-ти разрядного ЦАП микроконтроллера) и инкрементируется на это же значение для того, чтобы убрать отрицательный полупериод.

Одновременно с этим вычисляется автокорреляция сигнала и строится временная диаграмма (Рисунок 9) и автокорреляционная функция (Рисунок 10):

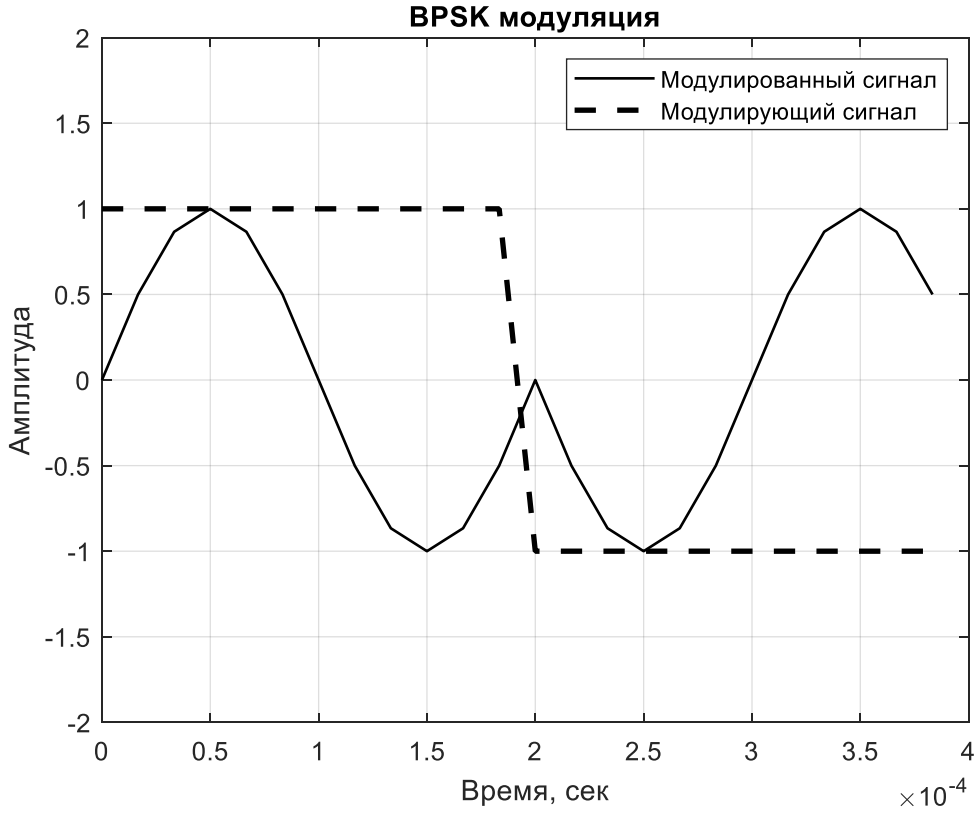

Рисунок 9 – Временная диаграмма последовательности Баркера из двух символов и сигнала, получившегося в результате выполнения программы

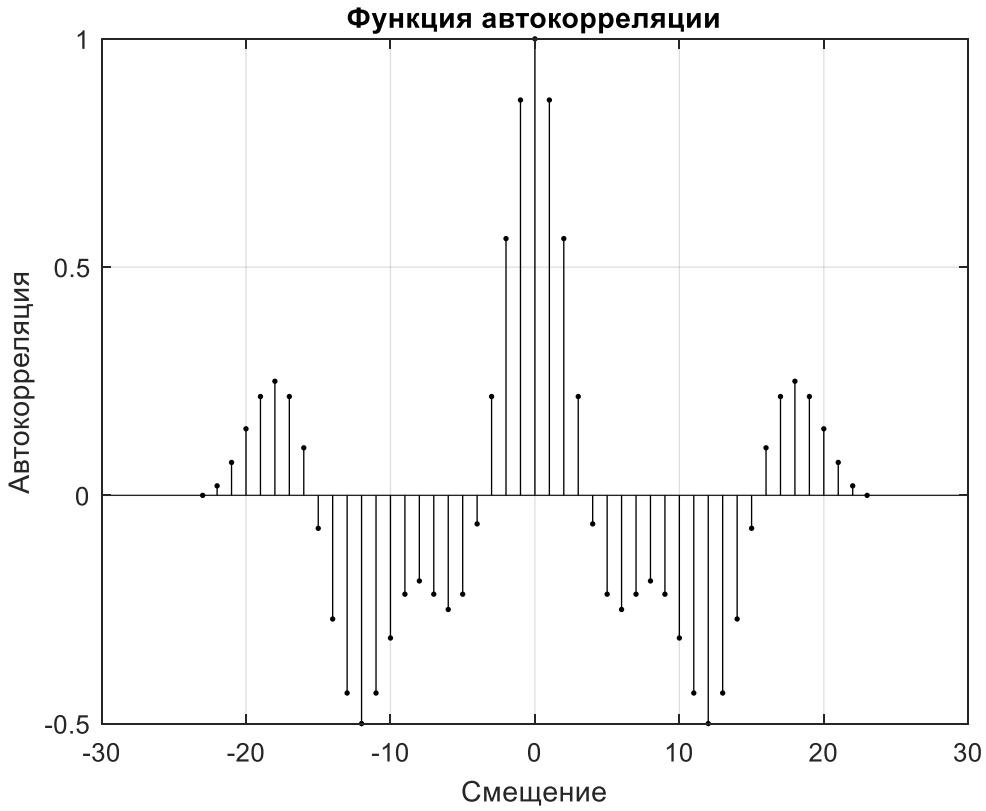

Рисунок 10 – Автокорреляционная функция сигнала, моделируемого кодом Баркера длиной 2 элемента

Результат, записанный в текстовый файл, представлен на рисунке 11:

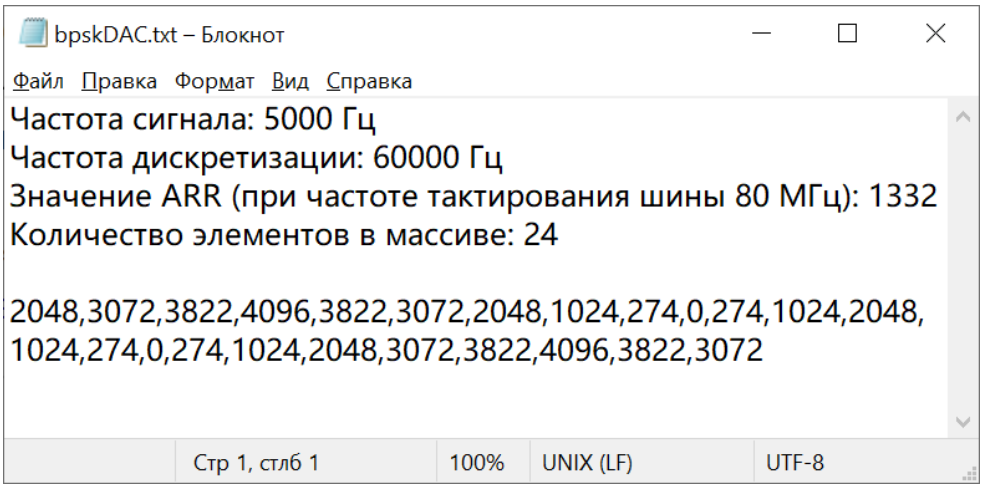

Рисунок 11 – Содержимое текстового файла с результатами выполнения программы

Пик автокорреляционной функции при небольшом количестве элементов недостаточно выражен, поэтому дополнительно были смоделированы сигналы с следующими параметрами:

Точность может быть увеличена путем увеличения дискретизации сигнала. Увеличим количество отсчетов сигнала на один период синусоиды с 12 до 32, при этом частота дискретизации для сигнала с собственной частотой 5 кГц будет равна 160 кГц. Временная диаграмма представлена на рисунке 12:

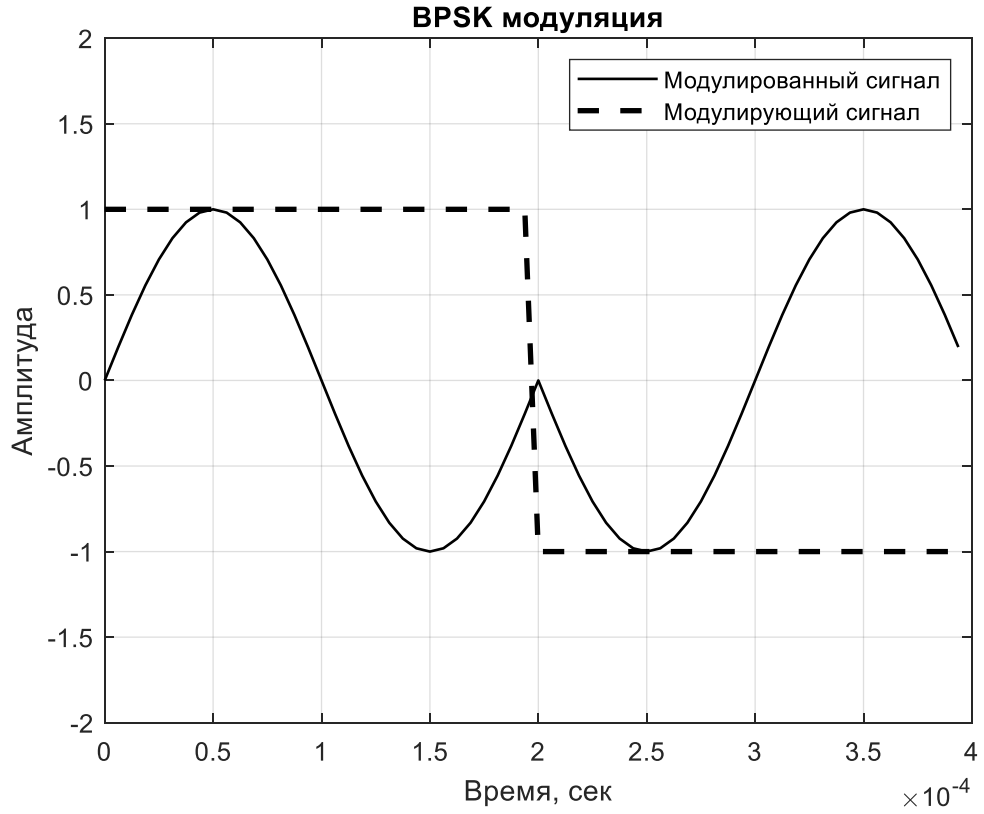

Рисунок 12 – Временная диаграмма последовательности Баркера из двух символов и сигнала, получившегося в результате выполнения программы при увеличении частоты дискретизации

Сформированный сигнал стал заметно плавнее, дальнейшее увеличение частоты дискретизации не имеет особого смысла, так как это будет сопровождаться увеличением размера сформированного массива. Автокорреляционная функция такого сигнала изображена на рисунке 13:

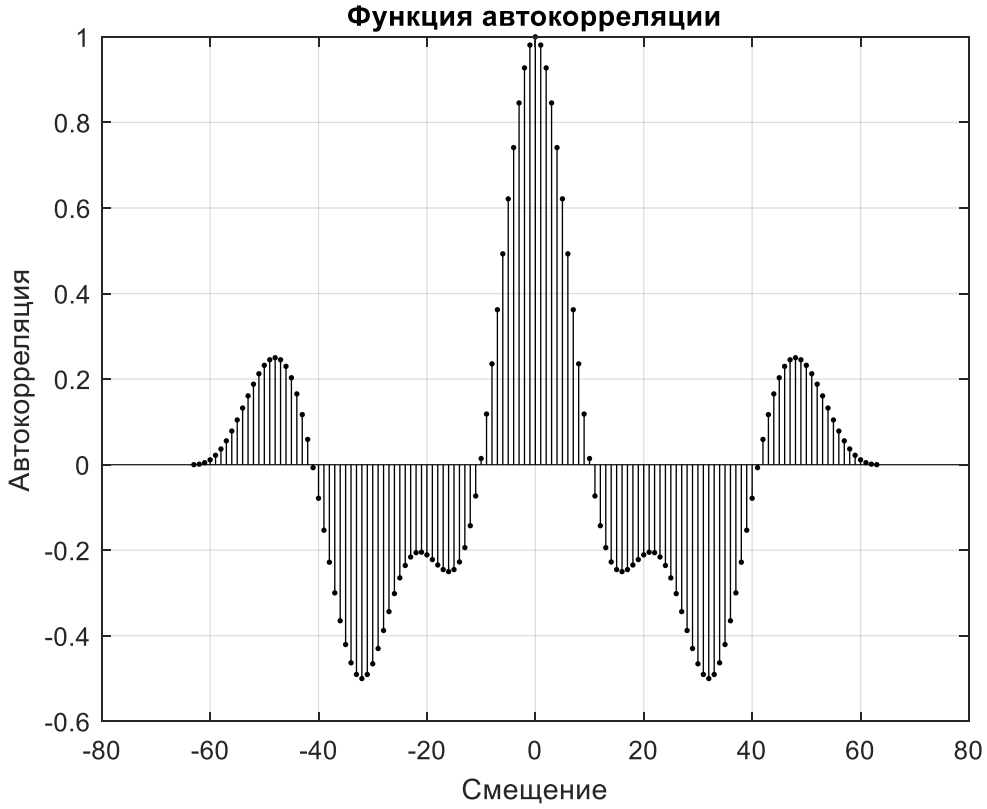

Рисунок 13 – Автокорреляционная функция сигнала, моделируемого кодом Баркера длиной 2 элемента при увеличении частоты дискретизации

Повышение частоты дискретизации улучшает точность оценки автокорреляции за счет более плотной выборки сигнала. Это может помочь выявить более слабые корреляционные связи или различать сигналы с близкими задержками.

Однако следует учитывать, что увеличение частоты дискретизации требует большего объема вычислительных ресурсов и памяти для обработки и хранения сигнала. Кроме того, дальнейшее повышение частоты дискретизации может привести к увеличению шума и спектральной ширины сигнала, что также может повлиять на автокорреляционную характеристику.

Для более явного выделения пика автокорреляционной характеристики была увеличена длина кода Баркера с двух элементов до семи. Временная диаграмма и автокорреляционная характеристика сигнала представлена на рисунках 14 и 15 соответственно:

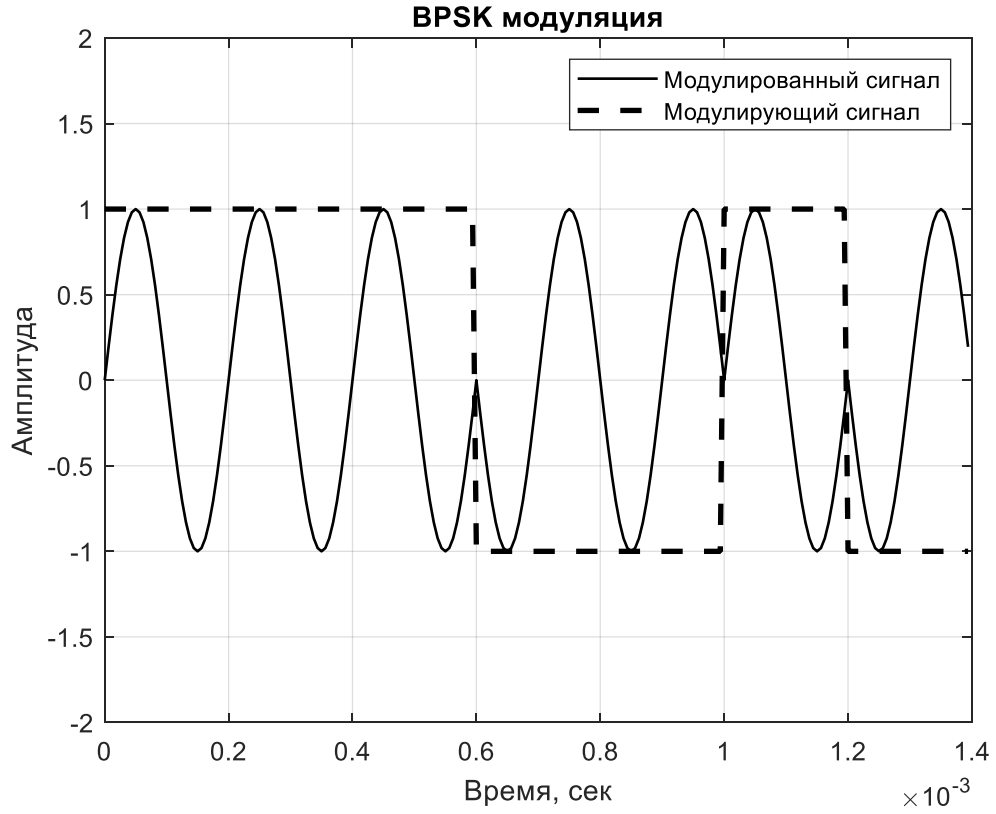

Рисунок 14 – Временная диаграмма последовательности Баркера из семи символов и сигнала, получившегося в результате выполнения программы

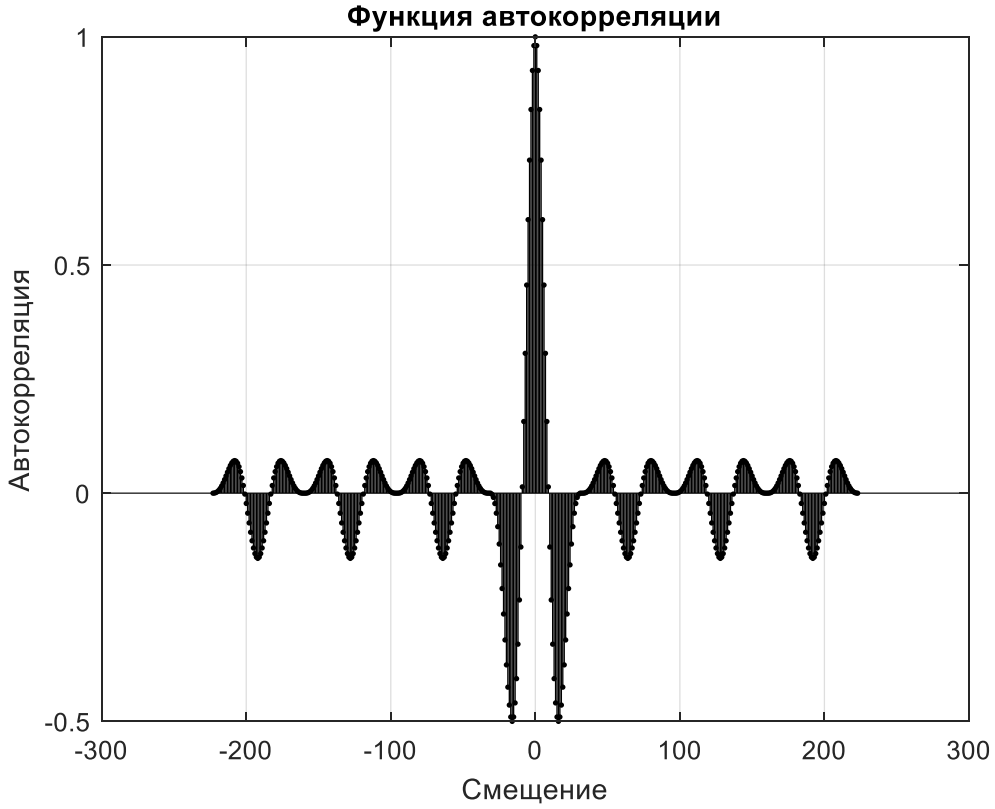

Рисунок 15 – Автокорреляционная функция сигнала, моделируемого кодом Баркера длиной 7 элементов

По характеристике видно, что разрешающая способность увеличилась еще сильнее. При увеличении длины последовательности Баркера возможно более точное определение задержек между сигналами, автокорреляционная характеристика показывает более явно выраженный и отчетливый пик, что облегчает определение задержки.

Увеличим длину последовательности до 11 символов. Рассмотрим полученные временные диаграммы (Рисунок 16) и автокорреляционную характеристику (Рисунок 17):

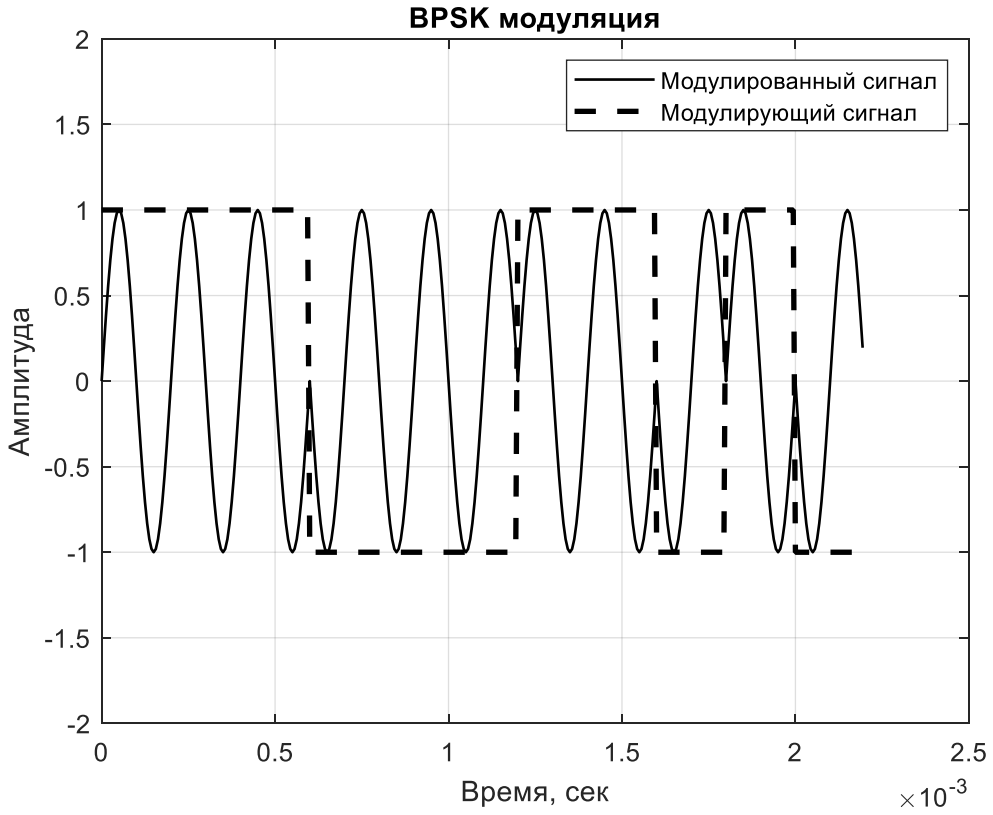

Рисунок 16 – Временная диаграмма последовательности Баркера из одиннадцати символов и сигнала, получившегося в результате выполнения программы

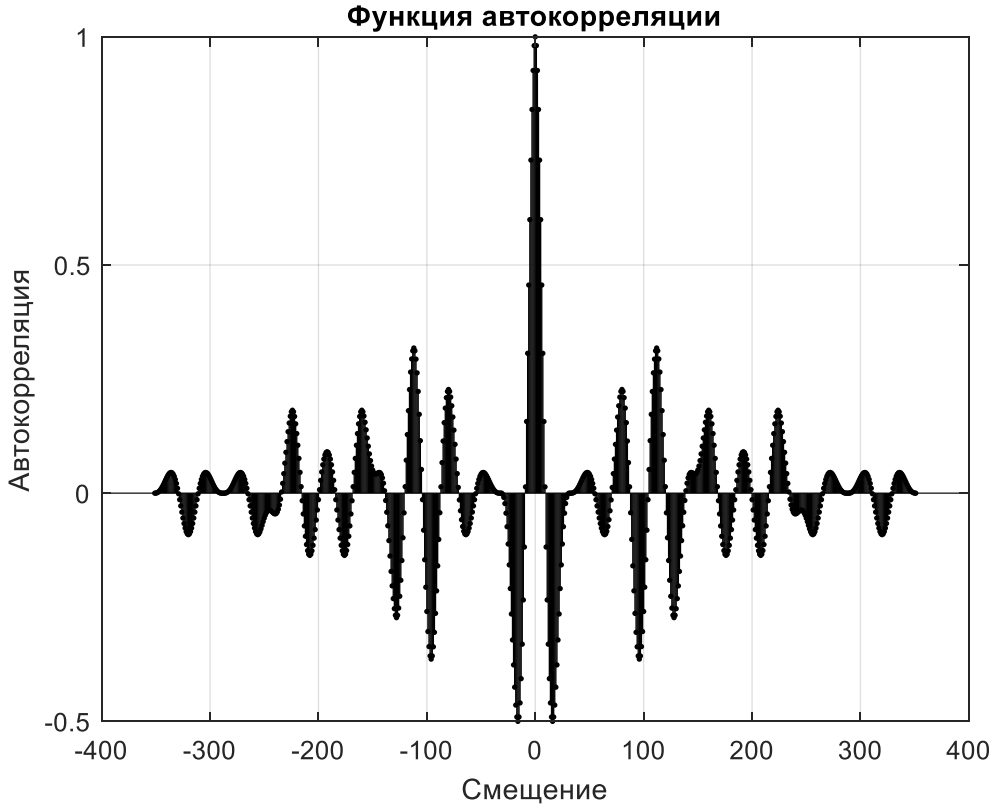

Рисунок 17 – Автокорреляционная функция сигнала, моделируемого кодом Баркера длиной 11 элементов

При еще большем увеличении количества символов была замечена и другая особенность: может происходить расширение боковых лепестков в автокорреляционной характеристике. Это связано с увеличением числа возможных сдвигов (лагов) в автокорреляционной функции. Боковые лепестки могут представлять собой помехи и приводить к ухудшению способности различения пиков автокорреляции.

Очевидно, что увеличится и вычислительная сложность. Обработка и анализ более длинных последовательностей Баркера может быть более ресурсоемкой задачей и требовать большего времени выполнения.

Используем последовательность Баркера длиной в 13 символов (Рисунки 18 и 19):

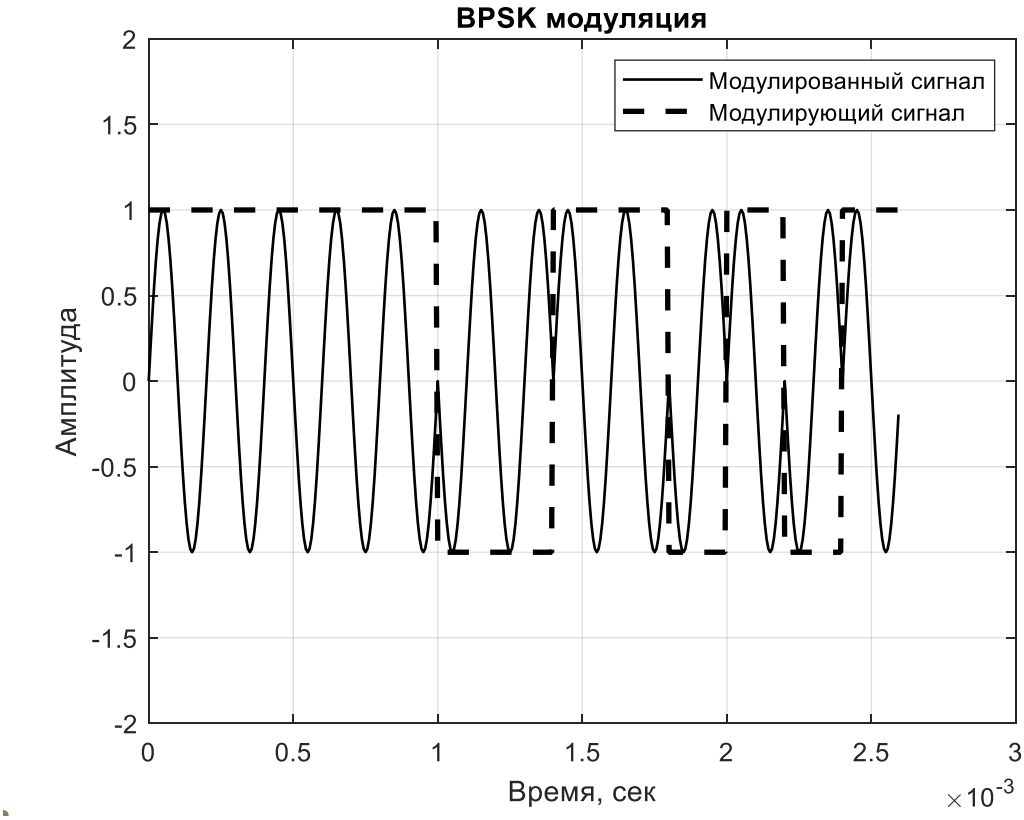

Рисунок 18 – Временная диаграмма последовательности Баркера из семи символов и сигнала, получившегося в результате выполнения программы

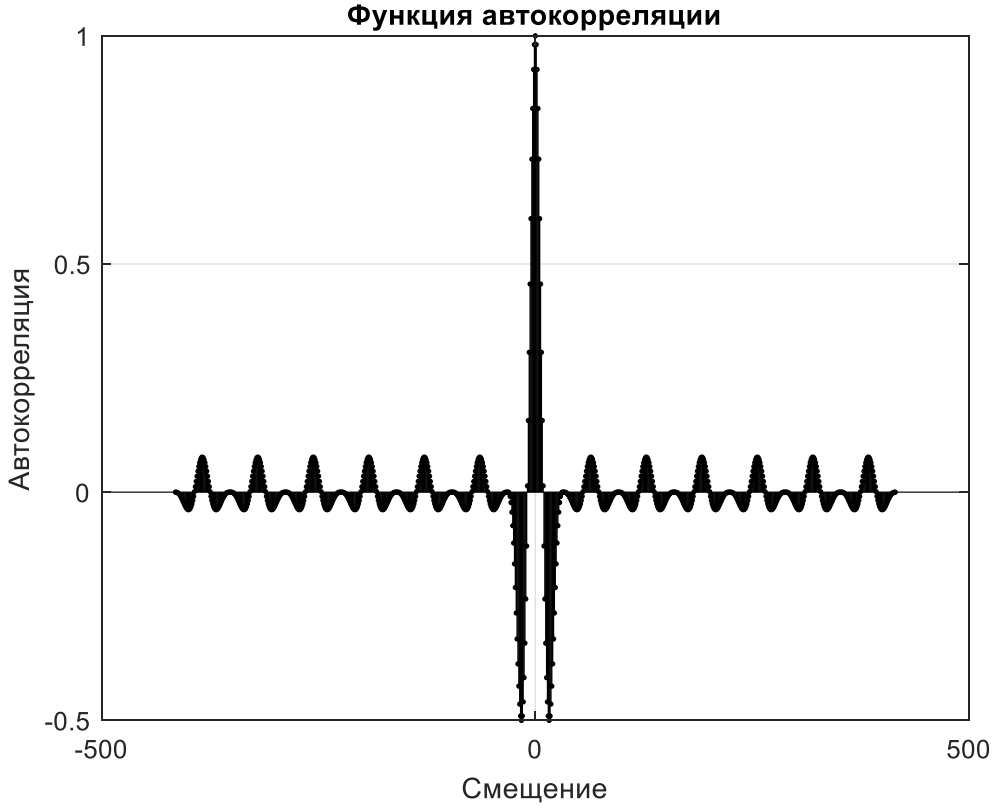

Рисунок 19 – Автокорреляционная функция сигнала, моделируемого кодом Баркера длиной 13 элементов

В целом, увеличение длины последовательности Баркера может улучшить разрешающую способность автокорреляционной характеристики, однако может также привести к появлению дополнительных помех в виде боковых лепестков. По полученным характеристикам было отмечено, что ярко выраженные боковые лепестки при использовании последовательностей длиной 7 и 13 символов имеют отрицательные значения корреляции.

При выборе оптимальной длины следует учитывать требования к разрешающей способности и вычислительным ресурсам системы. Исходя из результатов моделирования и выдвинутых критериев оптимальным является сигнал, модулируемый кодом Баркера из семи элементов. Массив, хранящий выборку с таким сигналом, поместится в память микроконтроллера, скорость его обработки будет достаточно высокой, а ярко выраженный пик корреляционной функции позволит точнее определять расстояние.

2.2 Инициализация DAC, DMA и таймеров микроконтроллера:

Для работы системы требуется микроконтроллер STM32 (имеющий DMA, DAC и ADC), микрофон и динамик. Для удобной настройки микроконтроллера и генерации кода, инициализирующего периферию, использована программа CubeMX.

Так как для генерации гармонического сигнала используются табличные значения синуса и ЦАП микроконтроллера, именно делением частоты тактирования микроконтроллера определяется частота обновления таймера, отвечающего за пересылку значений по DMA – это значение и будет частотой дискретизации сигнала. Контроллер STM32F407VET6 используемый в макете тактируется внешним кварцевым резонатором 80 МГЦ. [5, 6]

Тактирования таймера вычисляется по формуле 1:

$$\mathrm{Update}_{\mathrm{event}} = \frac{\mathrm{TIM}_{\mathrm{CLK}}}{(PSC + 1)(ARR + 1)}, \quad (1)$$

где $\mathrm{TIM}_{\mathrm{CLK}}$ – частота тактирования шины (80 МГц), PSC – делитель частоты, ARR – значение, после которого таймер будет обновляться.

Определившись с сигналом, который будет использован в макете и сгенерировав его отсчеты с помощью программы из приложения А, получим необходимое значение регистра ARR = 499. Выставляем это значение в окне настройки при инициализации таймера для тактирования АЦП и пересылки данных в ЦАП (Рисунок 20).

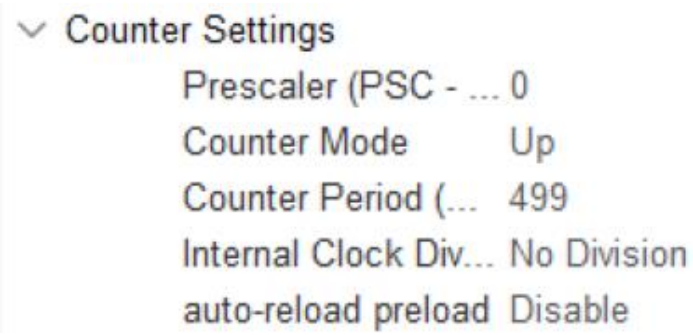

Рисунок 20 – Окно настройки таймера, вызывающего пересылку данных из памяти в ЦАП и тактирующего АЦП

Если необходимо, чтобы расстояние измерялось регулярно со строго заданной периодичностью, инициализируем еще один таймер. Например, для того, чтобы таймер обновлялся два раза в секунду, выставляем: PSC = 9999, ARR = 1999 (Рисунок 21).

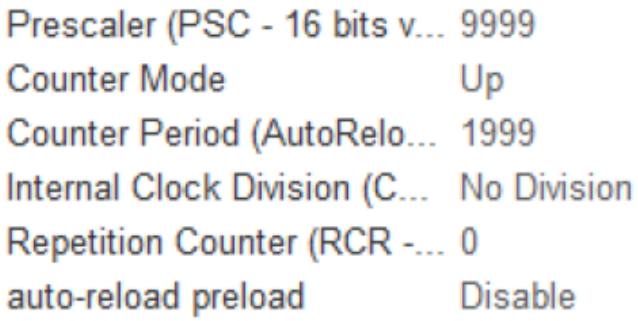

Рисунок 21 – Окно настройки таймера, регулирующего частоту измерений

При инициализации АЦП, необходимо в качестве триггера указать соответствующий таймер, а также удостоверится в том, что выключен режим непрерывной передачи (Рисунок 22):

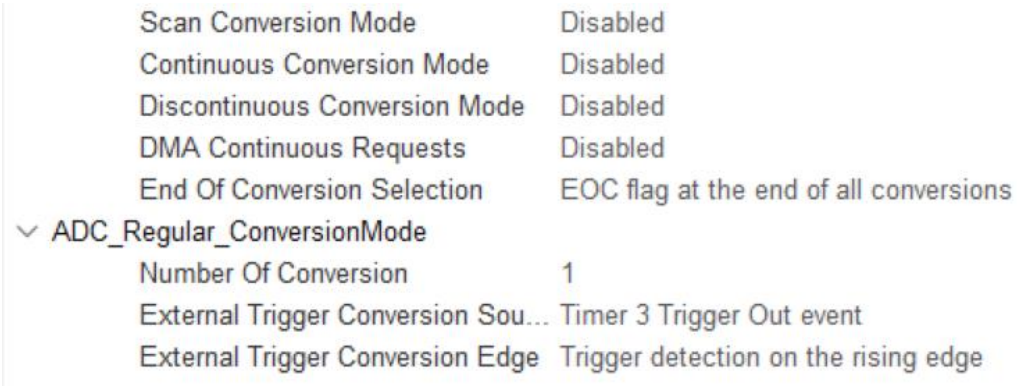

Рисунок 22 – Окно инициализации ADC

При инициализации ЦАП нужно также установить в качестве триггера таймер (Рисунок 23) и инициализировать DMA для передачи значений из памяти в периферию (Рисунок 24)

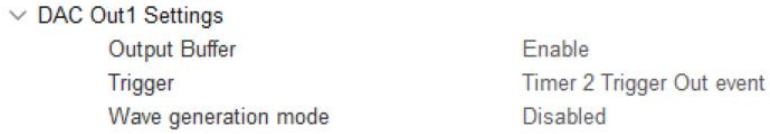

Рисунок 23 – Окно инициализации ЦАП

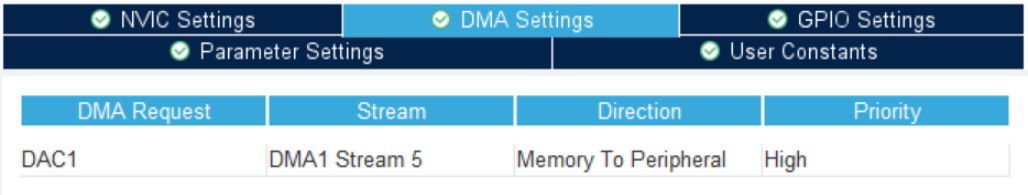

Рисунок 24 – Окно инициализации DMA

В основной программе для запуска системы достаточно, используя функции из библиотеки HAL, запустить таймеры, DMA и ADC.

При подаче питания на микроконтроллер DMA, инициализированная в режиме пе, будет непрерывно отправлять значения из памяти на ЦАП.

Использование ЦАП совместно с DMA, позволяет освободить ресурсы микроконтроллера для последующего приема сигнала и его обработки.

2.3 Определение расстояния между динамиком и микрофоном:

Для удобства дальнейшей отладки системы все ее компоненты (микрофон со встроенным усилителем, динамики, усилители к ним, Bluetooth модуль для связи по UART) были распаяны на сэндвич-плате, которая устанавливается сверху на отладочную плату STM32. Результат представлен на рисунке 25.

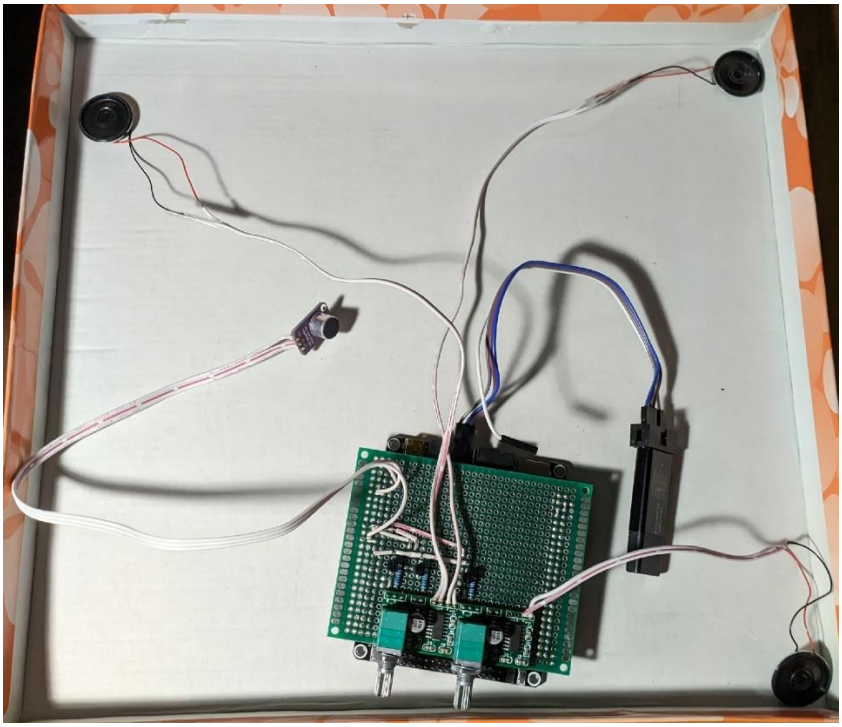

Рисунок 25 – Макет для проверки работы системы

Каналы усилителей коммутируются биполярными транзисторами, базы которых, подключены к ножкам контроллера. После подачи питания на макет инициализируются АЦП, ЦАП, таймеры и DMA. Затем микроконтроллер излучает сигнал из выбранного динамика, одновременно принимая его с АЦП. После переполнения массива хранящего принятые значения сигнала запускается процесс вычисления корреляционной функции. Во время отладки системы значения массива хранящего отсчеты принятого сигнала передаются

по UART. Для доступа к ним достаточно выбрать нужный виртуальный COM порт в любом терминале последовательного порта. На рисунке 26 в качестве примера приведено окно терминала Termite.

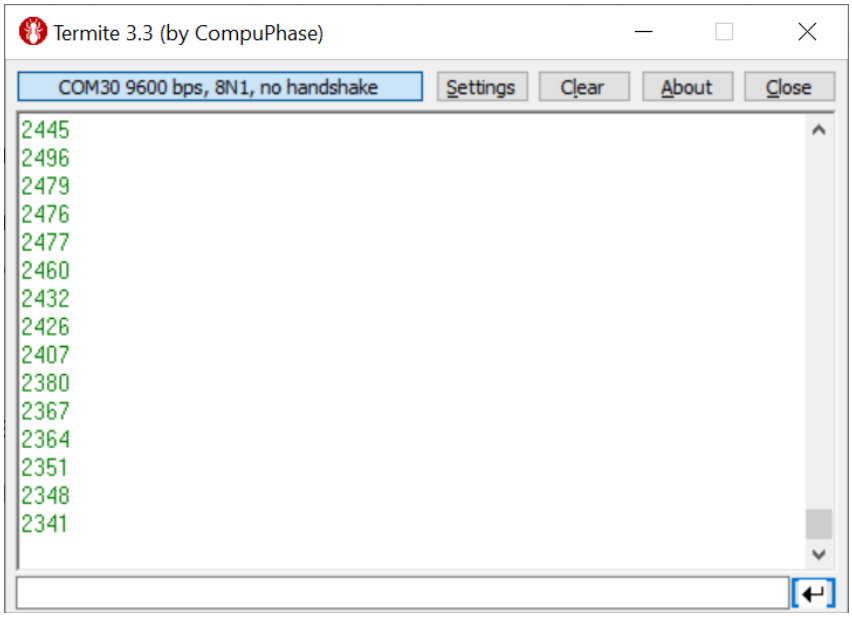

Рисунок 26 – Отсчеты сигнала, переданные с помощью UART

Пришедшие отсчеты копируются в текстовый файл, к которому имеет доступ программа из приложения А. В процессе выполнения программы строится временная диаграмма сигнала, принятого с микрофона (Рисунок 27):

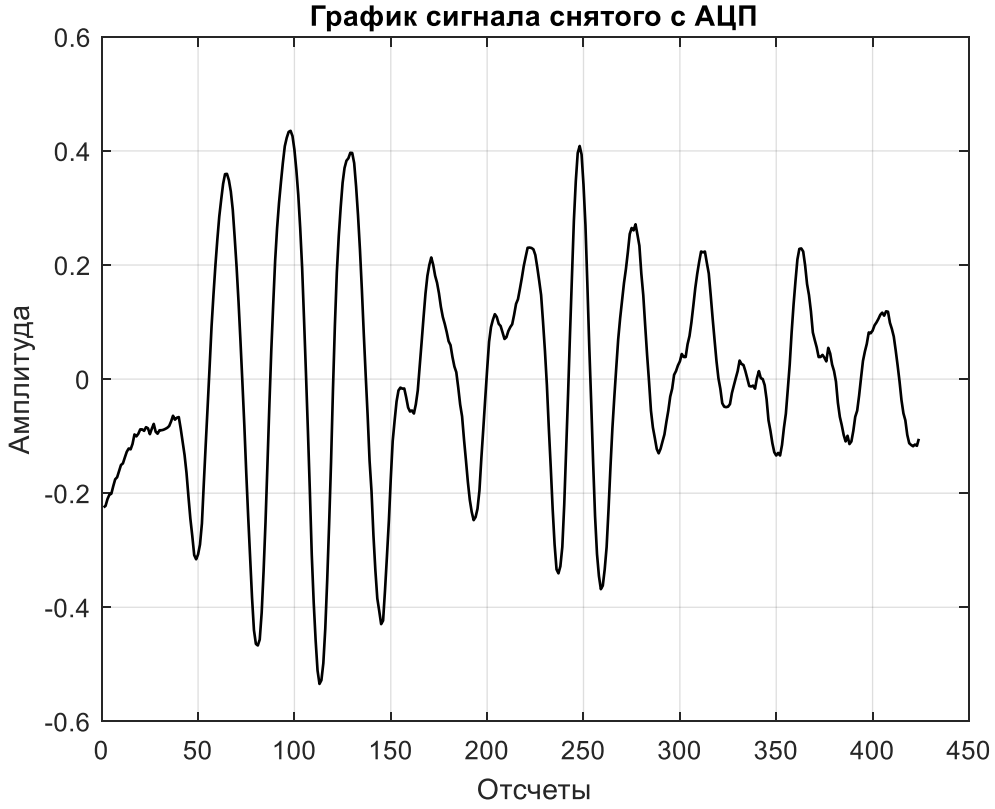

Рисунок 27 – Временная диаграмма сигнала, модулируемого семью элементами Баркера, принятого с микрофона

После приема сигнала необходимо определить время задержки. Для этого рассчитывается взаимная корреляционная функция между пришедшим и отправленным сигналами. Для дискретных или цифровых сигналов — это сумма произведений совпадающих (перекрывающих друг друга) субимпульсов:

$$C_{xy}(m) = \sum_{n=-\infty}^{\infty} x_n y_{n-m}, \quad (2)$$

где x_n – элемент первого массива, y_{n-m} – элемент второго массива смещенного на m.

На языке С (Приложение В) и на языке MATLAB (Приложение А) была написана функция, которая рассчитывает корреляцию между массивом

пришедшего и отправленного сигнала. Адаптация функции в MATLAB позволяет визуально оценить корреляционную характеристику, которая аналогичным образом рассчитывается с помощью микроконтроллера в процессе работы лабораторного стенда. Взаимная корреляционная характеристика сигнала, принятого микрофоном, изображена на рисунке 28.

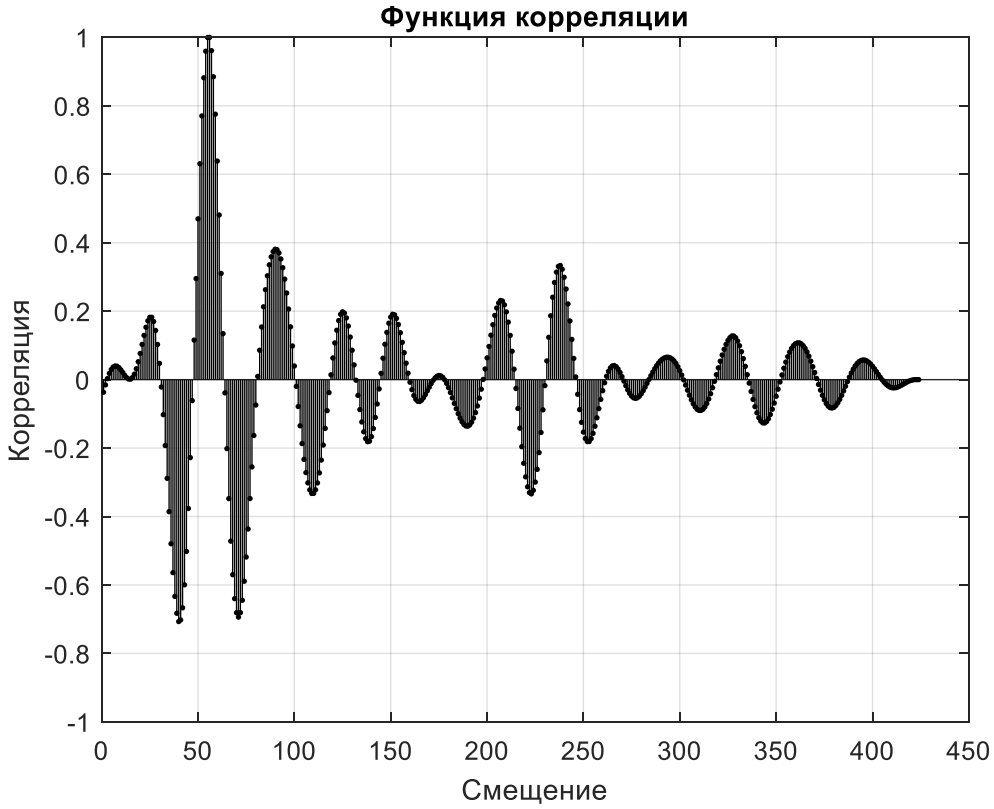

Рисунок 28 – Взаимная корреляционная характеристика сигнала, модулируемого семью элементами Баркера, принятого микрофоном

По характеристике заметно, что уровень боковых лепестков заметно возрос: это связанно с наличием шумов, которые вместе с полезным сигналом пропускаются микрофоном.

Функция определяет задержку по максимальному значению корреляционной характеристики. В последствии по этому значению и определяется расстояние.

Накладывая исходный массив на массив отсчетов принятых с помощью АЦП микроконтроллера (Рисунок 29), программа перемножает отсчеты, суммирует результаты, записывает их в переменную и смещает массив. Эти операции повторяются до тех пор, пока исходный массив не сместиться до конца массива с отсчетами АЦП. Результат вычислений изображен на рисунке 30.

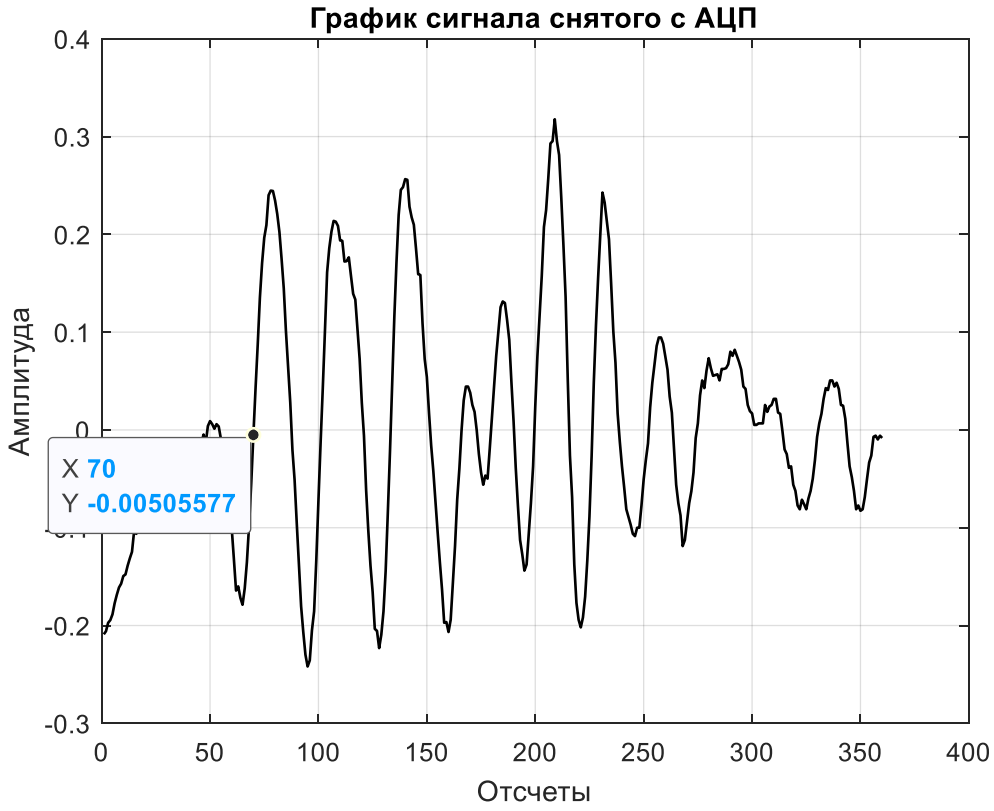

Рисунок 29 – Отсчеты сигнала, модулируемого пятью элементами Баркера, принятого с микрофона

Маркером на графике отмеченно начало пришедшего сигнала.

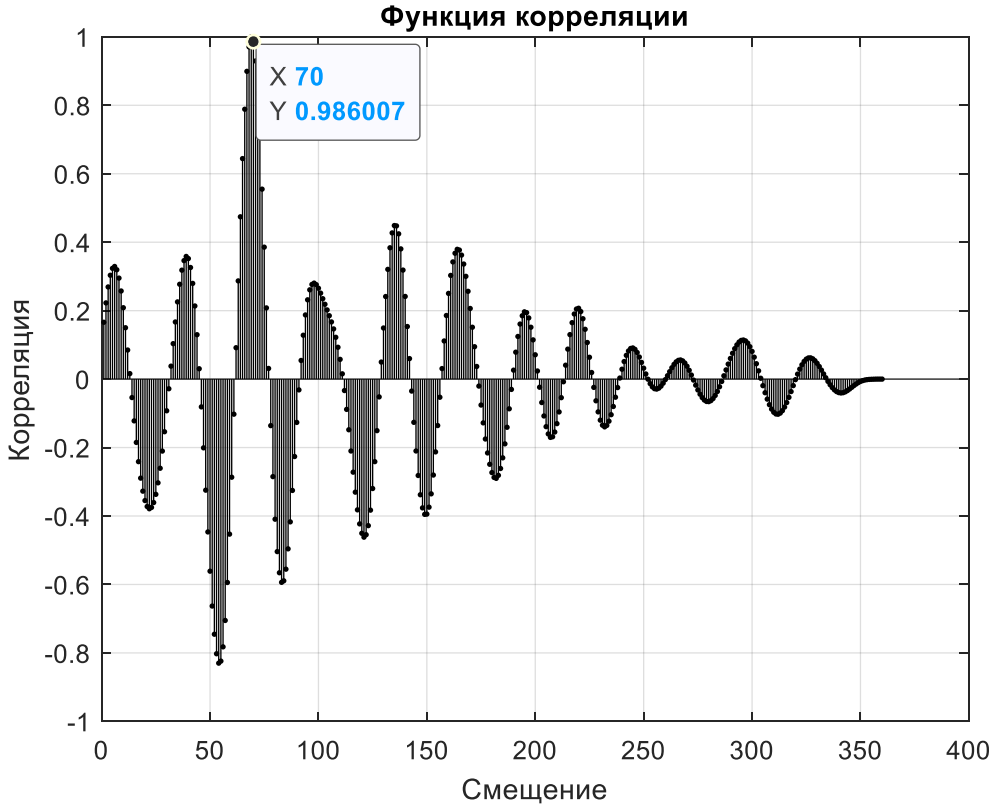

Рисунок 30 – Взаимная корреляционная характеристика сигнала, модулируемого пятью элементами Баркера, принятого микрофоном

Смещение, при котором был достигнут максимум корреляционной функции (отмечен маркером) и является той самой задержкой распространения сигнала – остается лишь перевести это значение в секунды и помножить на скорость звука.

Макет позволяет анализировать работу системы при различных кодовых последовательностях. Для примера рассмотрим, как определяется расстояние при различных кодовых последовательностях. На рисунках 31, 32 представлены временная диаграмма и взаимная корреляционная функция сигнала, моделированного последовательности Баркера длиной 3 символа, принятого микрофоном.

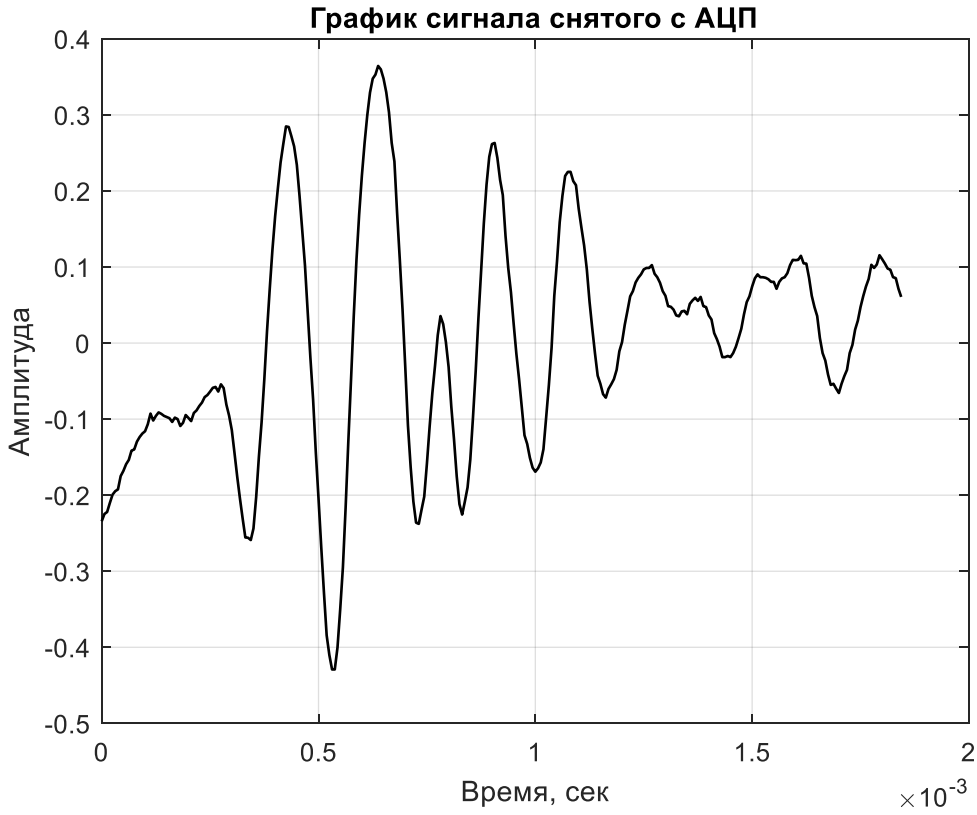

Рисунок 31 – Временная диаграмма сигнала, модулируемого тремя элементами Баркера, принятого с микрофона

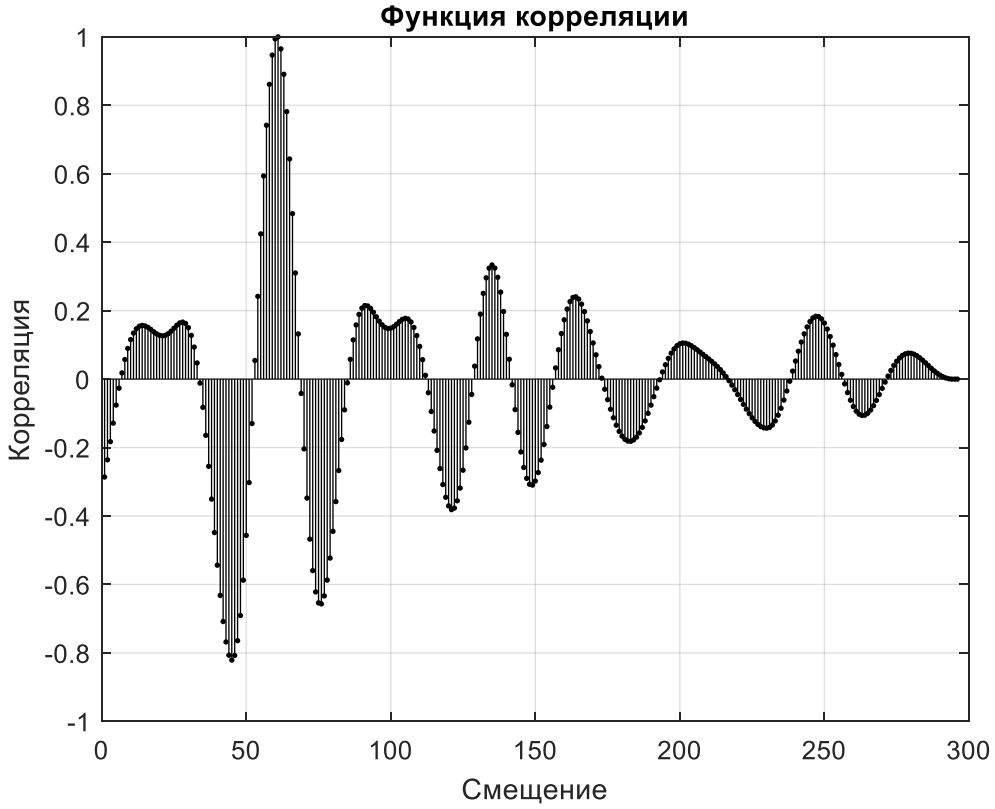

Рисунок 32 – Взаимная корреляционная характеристика сигнала, модулируемого тремя элементами Баркера, принятого микрофоном

По рисункам очевидно, что точность определения задержки станет гораздо ниже по причине того, что главный пик расширился, а уровень бокового лепестка практически сопоставим с уровнем максимума корреляционной характеристики.

Увеличим количество элементов до пяти (Рисунок 33 и 34):

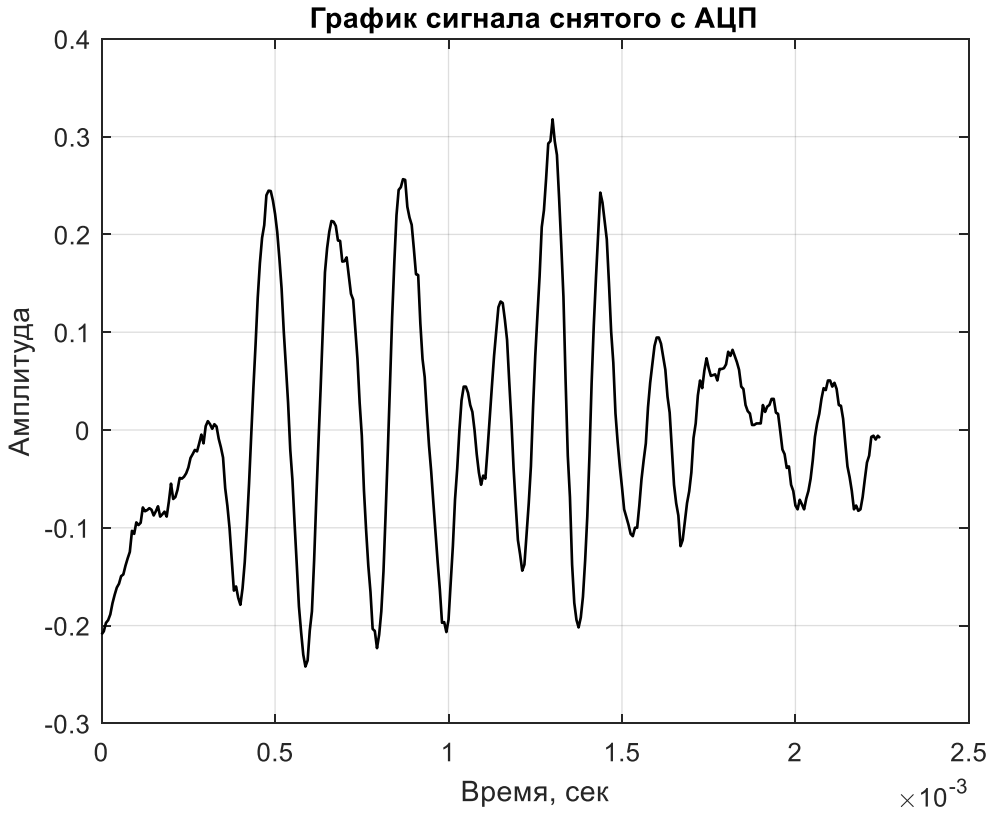

Рисунок 33 – Временная диаграмма сигнала, модулируемого пятью элементами Баркера, принятого с микрофона

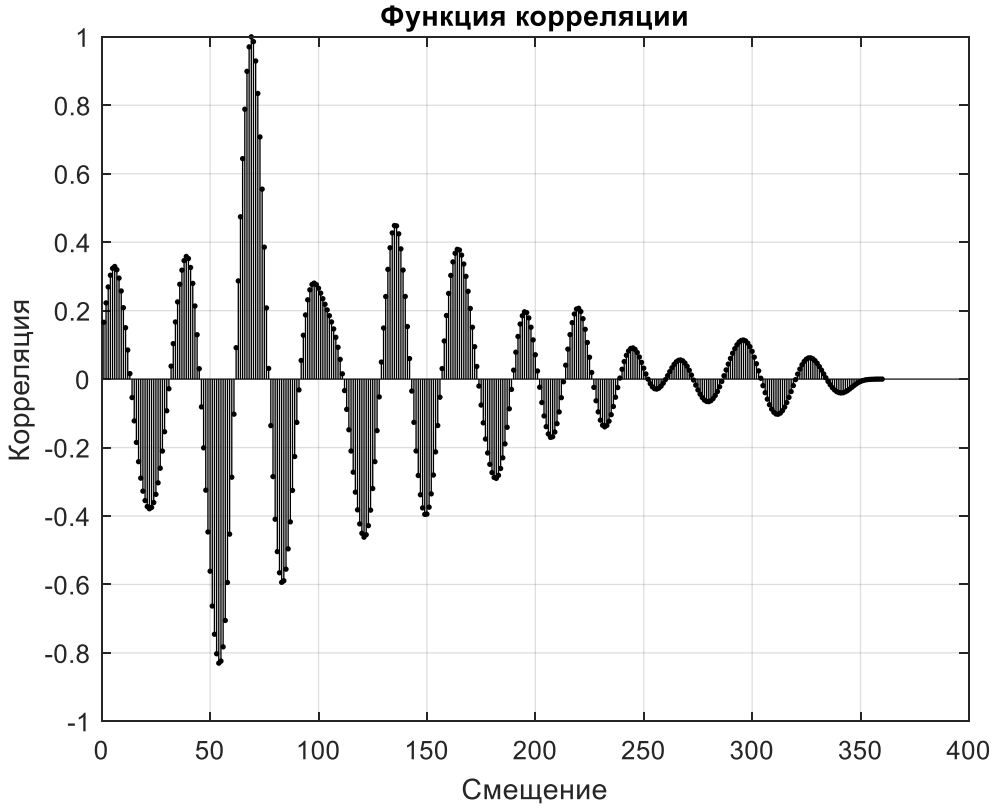

Рисунок 34 – Взаимная корреляционная характеристика сигнала, модулируемого пятью элементами Баркера, принятого микрофоном

Получившееся характеристика имеет большую разрешающую способность, а максимальный уровень боковых лепестков составил всего половину от максимума корреляционной функции.

Исследуем работу системы при более длинных последовательностях. На рисунках 35 и 36 изображены временная диаграмма и взаимная корреляционная характеристика сигнала, модулируемого одиннадцатью элементами Баркера.

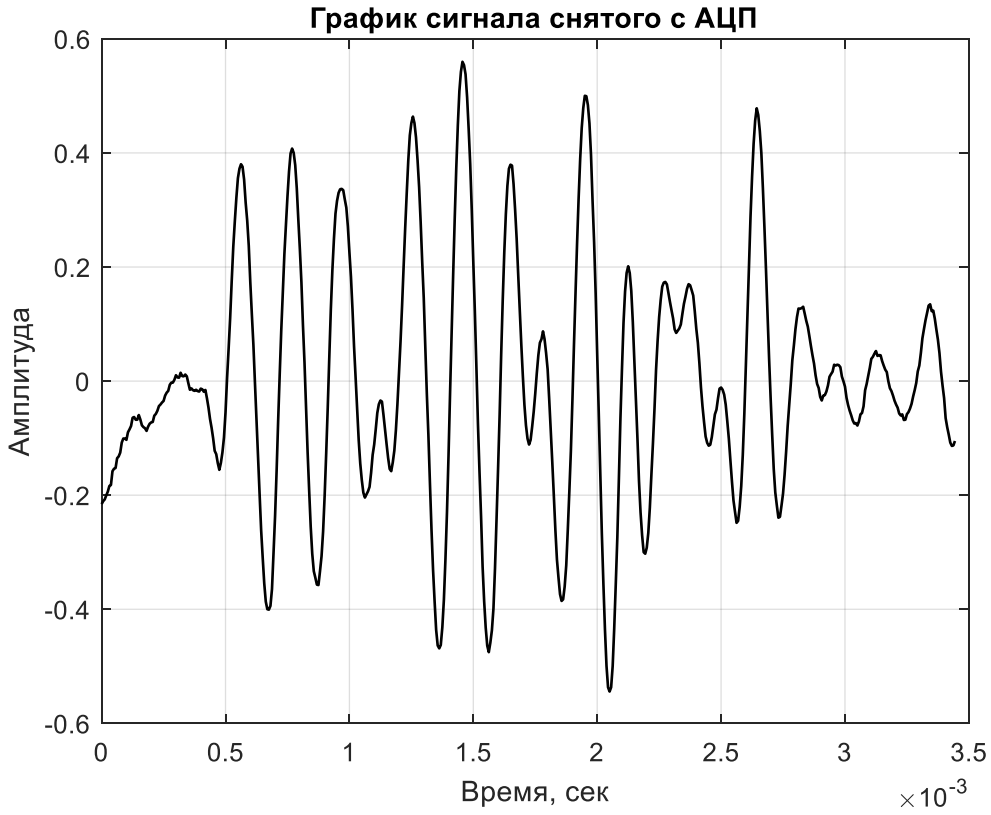

Рисунок 35 – Временная диаграмма сигнала, модулируемого одиннадцатью элементами Баркера, принятого с микрофона

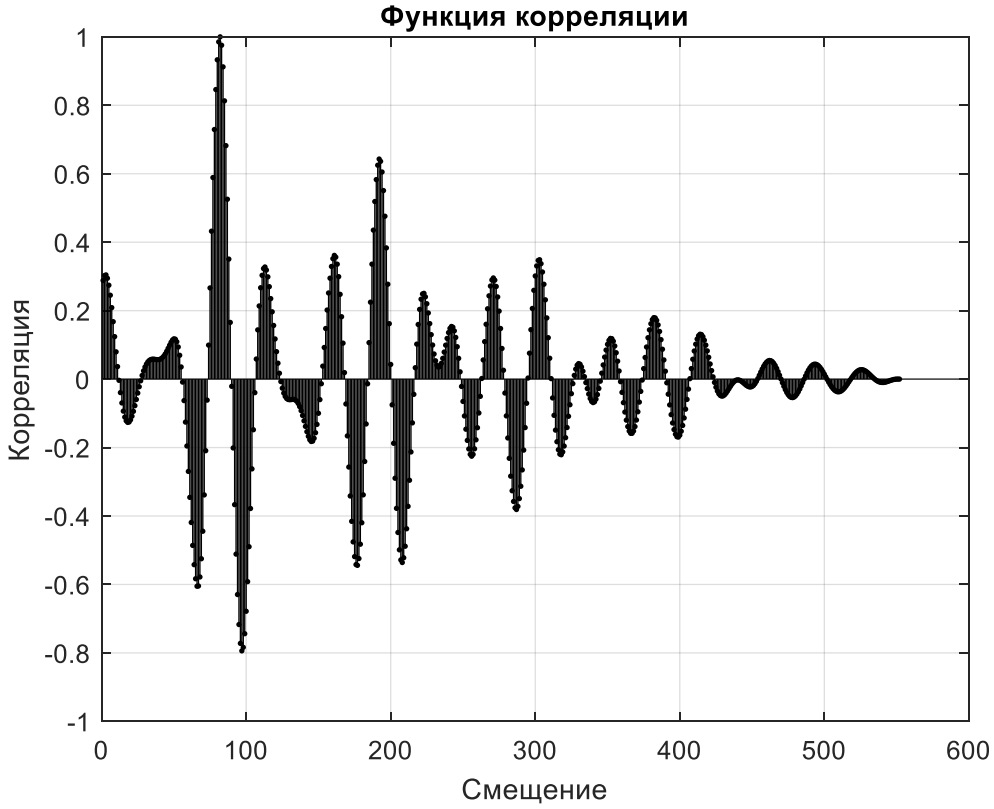

Рисунок 36 – Взаимная корреляционная характеристика сигнала, модулируемого одиннадцатью элементами Баркера, принятого микрофоном

На рисунках 37 и 38 изображены временная диаграмма и взаимная корреляционная характеристика сигнала, модулируемого тринадцатью элементами Баркера.

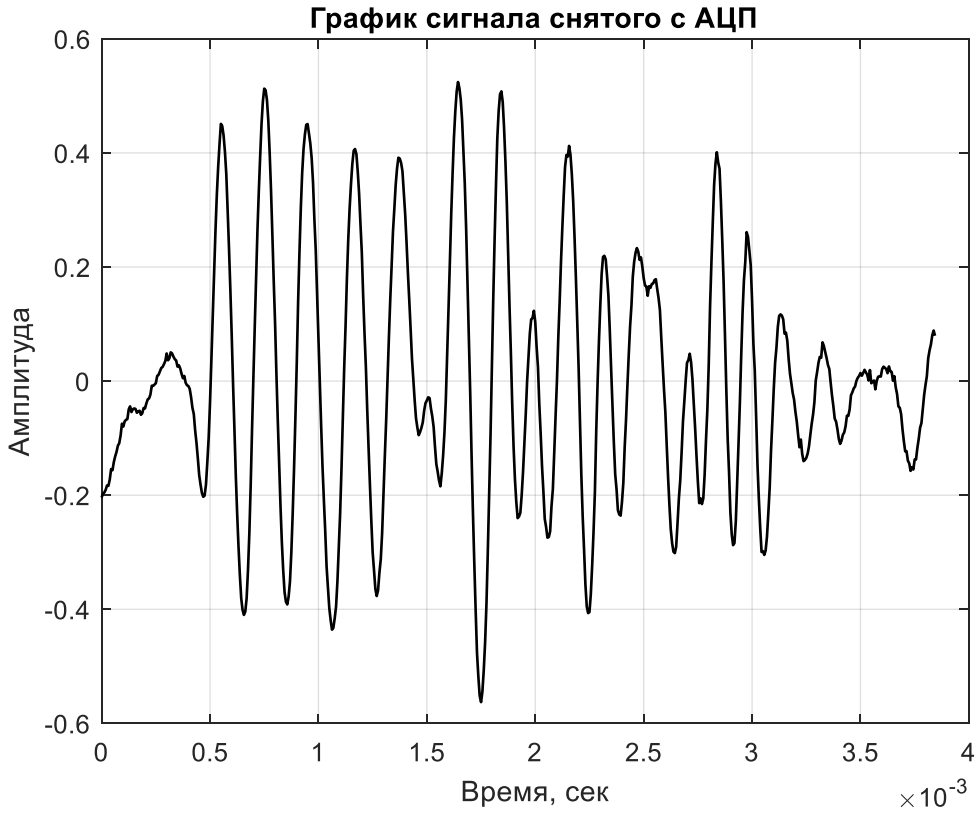

Рисунок 37 – Временная диаграмма сигнала, модулируемого тринадцатью элементами Баркера, принятого с микрофона

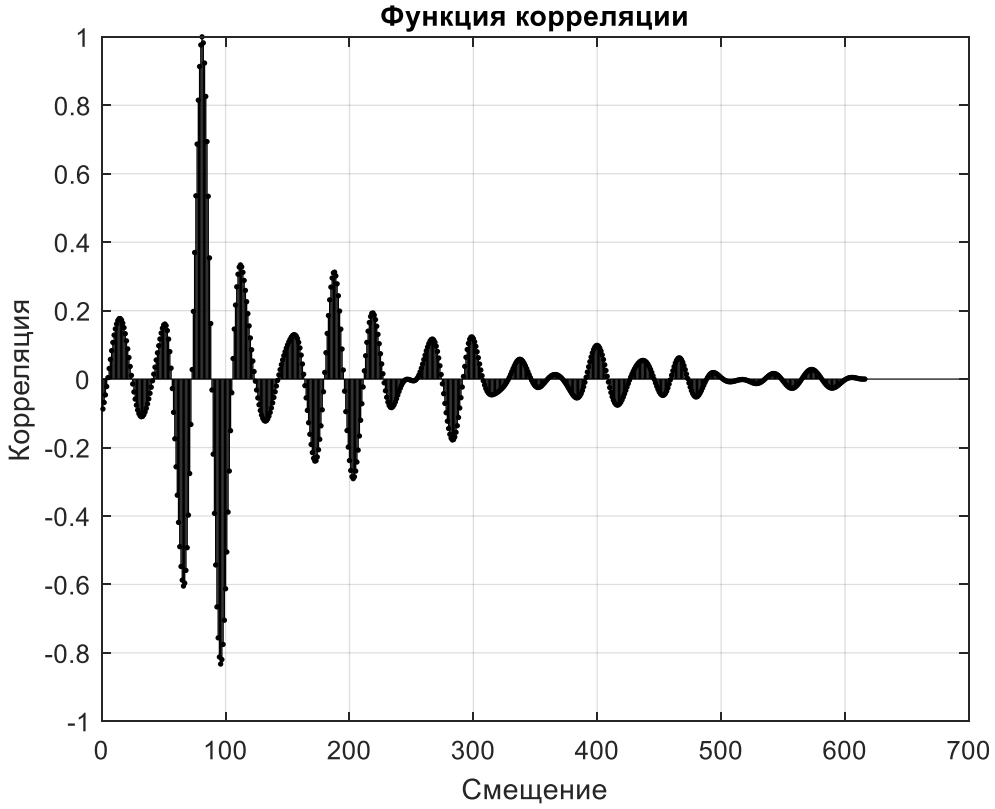

Рисунок 38 – Взаимная корреляционная характеристика сигнала, модулируемого тринадцатью элементами Баркера, принятого микрофоном

По полученным графикам можно подтвердить ранее сделанный вывод, что увеличение разрешающей способности при использовании более длинных последовательностей Баркера является избыточным. Положительный эффект от ярко выраженных пиков корреляционной функции нивелируется возрастающим уровнем боковых лепестков и долгой обработкой.

2.4 Решение задачи трилатерации:

Для определения местоположения используются несколько излучателей (обычно два или три), которые находятся на известном расстоянии друг от друга. Когда источники излучают звуковые сигналы, они распространяется во все стороны, и приходят к приемнику в разное время.

Приемник записывает время прихода сигнала, и затем с помощью трилатерации определяется координата приемника. Для этого необходимо провести окружности с центром в каждом источнике и радиусом, равным времени задержки сигнала между источником и приемником. Точка пересечения окружностей будет соответствовать местоположению приемника звука.

Для решения данной задачи нужно найти точку пересечения трёх окружностей, следовательно получаем систему из трёх уравнений окружностей, а именно (Формула 3):

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = r_1^2, \\ (x_2 - x)^2 + (y_2 - y)^2 = r_2^2, \\ (x_3 - x)^2 + (y_3 - y)^2 = r_3^2; \end{cases} \quad (3)$$

где $x_1, y_1, x_2, y_2, \quad x_3, y_3$ — координаты динамиков, r_1, r_2 и r_3 — расстояния до соответствующих динамиков, x и y — координаты микрофона.

В процессе выражения из системы координат микрофона получим (Формула 4):

$$\begin{cases} x = \dfrac{C * E - F * B}{E * A - B * D}, \\ y = \dfrac{C * D - A * F}{B * D - A * E}; \end{cases} \quad (4)$$

где $A = 2(x_2 - x_1)$, $B = 2(y_2 - y_1)$, $C = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2$, $D = 2(x_3 - x_2)$, $E = 2(y_3 - y_2)$, $F = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2$.

Написана программа, рассчитывающая координаты приемника (Приложение Б). Для проверки ее работы динамики коммутировались по очереди, позволяя обеспечить временное разделение при передаче сигналов, измеряющих расстояние и отправить измеренное расстояние по UART на компьютер.

Во время проверки работы программы не ставилась задача проверки точности полученных расчетов, поэтому координата микрофона при измерении расстояния измерялась приблизительно. Координаты динамиков были строго определены ([0 мм, 250 мм], [250 мм, 250 мм], [250 мм, 0 мм]) и занесены в программу. В окне программы необходимо указать правильной номер последовательного порта и при желании откалибровать производимые вычисления (Рисунок 39):

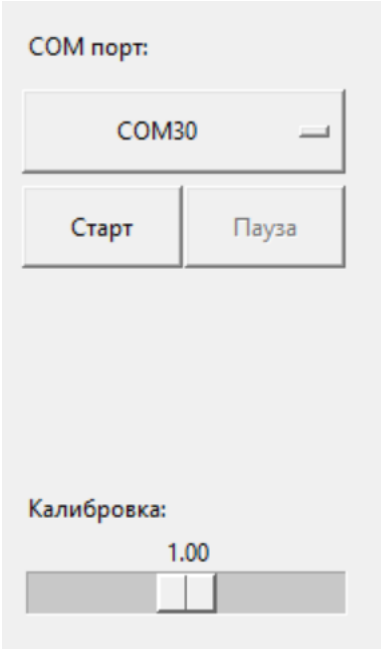

Рисунок 39 – Параметры необходимые для корректной работы системы

Результаты обработки данных о местоположении выводятся на графике (Рисунок 40, 41):

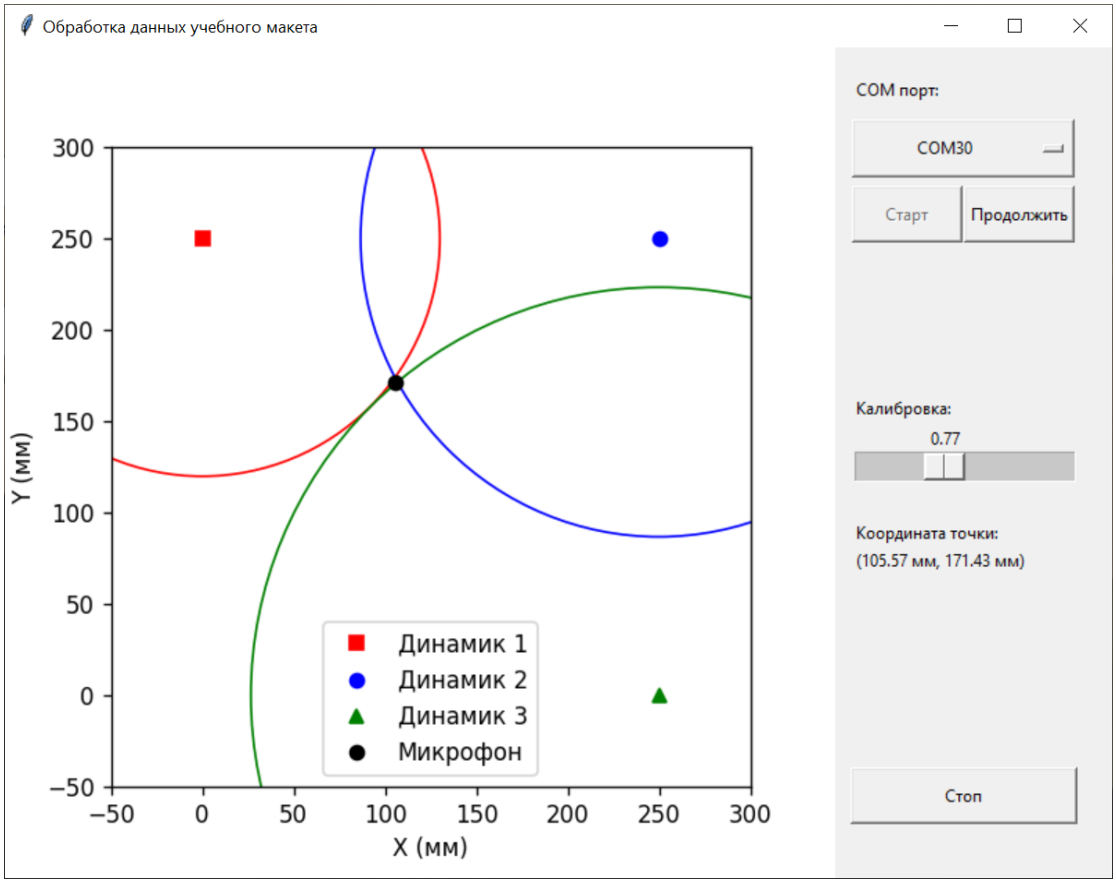

Рисунок 40 – Результат обработки данных при реальной координате микрофона (105, 170)

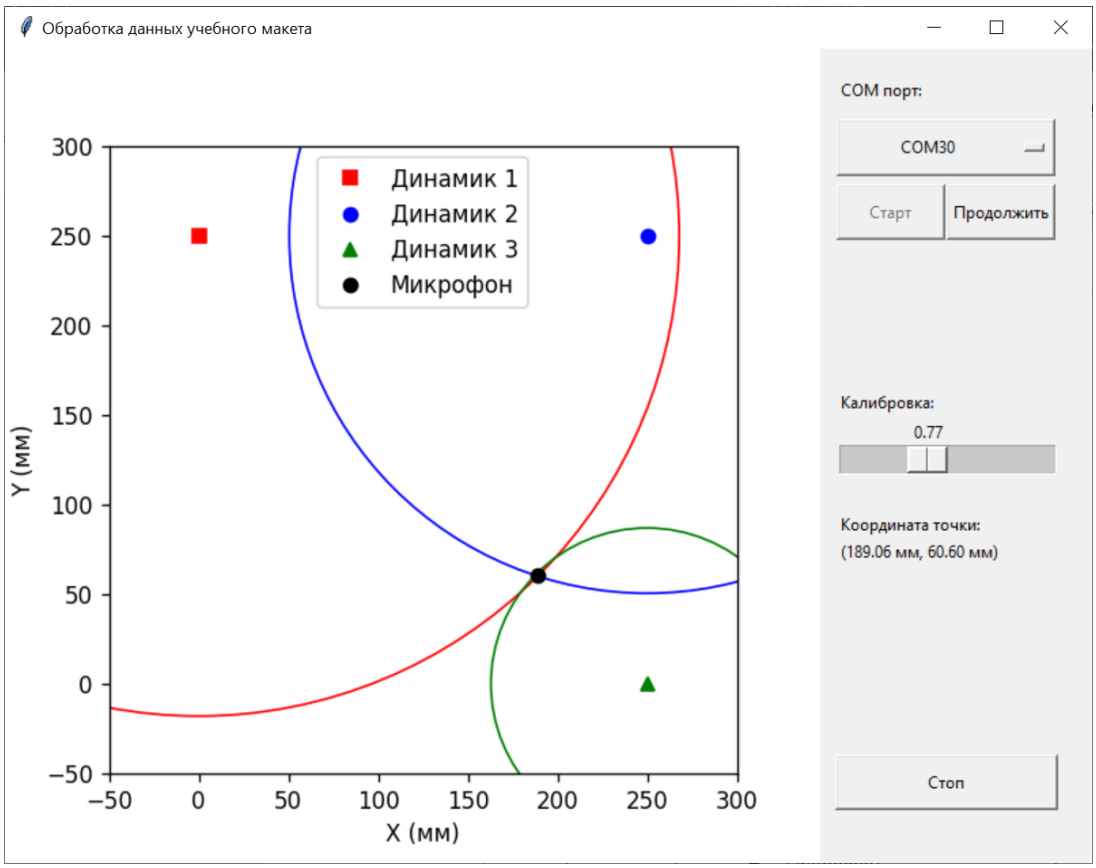

Рисунок 41 – Результат обработки данных при реальной координате микрофона примерно (190, 60)

В окне программы визуализируются следующие данные: заранее известные координаты источников сигнала, окружности с радиусами, равными измеренному с помощью макета расстоянию, точка пересечения окружностей и её координата, соответствующая искомой координате приемника сигнала.

График обновляется в реальном времени, скорость вычислений и точность получаемого результата, зависит от выбора псевдослучайной последовательности. Анализ оптимальных последовательностей был произведен в прошлом разделе.

Заключение

В результате проделанной работы было выяснено, что при проектировании навигационной системы в слышимом диапазоне подходят короткие псевдослучайные последовательности – в частности коды Баркера.

Был разработан алгоритм обработки полученных звуковых сигналов, позволяющий определять его задержку. В результате моделирования и экспериментального исследования было выяснено, как влияет длина последовательности на точность и скорость проводимых измерений. Также были определены оптимальные (по критерию стабильности работы системы и быстроты обработки данных при их использовании) коды Баркера: последовательности длиной в 5 и 7 символов.

Был разработан алгоритм вычисления координаты приемника сигналов. Программа с его реализацией способна принимать значения расстояния по последовательному порту, рассчитывать его координату по пересечении трех окружностей и в случае отсутствия явной точки пересечения усреднять полученные значения для нахождения искомой координаты.

Написанные программы на Python и MATLAB визуализируют обрабатываемые данные, что позволяет наглядно изучить работу навигационной системы и могут помочь в анализе различных псевдослучайных последовательностей.

Разработанная в ходе работы плата и спроектированный корпус лабораторного стенда позволяют учебному макету полноценно функционировать и выполнять требуемые измерения.

Данное устройство может использоваться как учебный макет студентами и школьниками для изучения принципов работы навигационных систем и возможностей микроконтроллера.

Список используемых источников

1. СПУТНИКОВАЯ РАДИОНАВИГАЦИОННАЯ СИСТЕМА "ГЛОНАСС" – Режим доступа: https://cyberleninka.ru/article/n/sputnikovaya-radionavigatsionnaya-sistema-glonass-1.

2. Сайт Роскосмоса: Статья про ГЛОНАСС — российская глобальная навигационная система – Режим доступа: https://www.roscosmos.ru/21923/.

3. Применение устройства некогерентной демодуляции "в целом" фазоманипулированных сигналов в радиосистемах управления, Глушков Алексей Николаевич – Режим доступа: https://cyberleninka.ru/article/n/primenenie-ustroystva-nekogerentnoy-demodulyatsii-v-tselom-fazomanipulirovannyh-signalov-v-radiosistemah-upravleniya/viewer.

4. Прикладной потребительский центра ГЛОНАСС – Режим доступа: https://www.glonass-iac.ru/guide/glonass.php.

5. Современные 32-разрядные ARM-микроконтроллеры серии STM32: цифроаналоговый преобразователь, Олег Вальпа.

6. Reference manual на микроконтроллер STM32F40x.

7. Дубинин А.Е. Анализ фазовой модуляции при передаче сигналов Баркера. – Самара: СамГУПС, 2011.

8. Варакин Л. Е. Системы связи с шумоподобными сигналами. — М.: Радио и связь, 1985.

9. Триангуляционная система определения координат источника звука – Режим доступа: https://cyberleninka.ru/article/n/triangulyatsionnaya-sistema-opredeleniya-koordinat-istochnika-zvuka

Приложение А

Код программы на языке MATLAB:

```
clear, clc, close all

load('BarkerCodes.mat');
% кодируемая последовательность
code = barker7;
% количество отсчетов одного периода синуса
m = 32;
% длительность импульса в периодах синусойды
num_of_periods_per_bit = 1;
% частота сигнала в Гц
fc = 5000;

fs = fc*m;
ts = 0 : 1/fs : (m*length(code)*num_of_periods_per_bit)/fs-1/fs;
N = length(ts);


sinus = sin(2*pi*fc*ts);

% длина одного бита в отсчётах

n_for_bit = m*num_of_periods_per_bit;


% формируем модулирующий сигнал
fm = zeros(1,N);
for i=1:length(code)
    for j=n_for_bit*(i-1)+1:n_for_bit*i
        fm(j) = code(i);
    end
end


x = sinus.*fm;

filename = 'bpskDAC.txt';  % Имя файла
fid = fopen(filename, 'w'); % Открываем файл для записи

% Запись частоты сигнала, частоты дискретизации и значение ARR в файл
fprintf(fid, 'Частота сигнала: %d Гц\n', fc);
fprintf(fid, 'Частота дискретизации: %d Гц\n', fs);
ARR = round((80*10^6)/(fs))-1;
fprintf(fid, 'Значение ARR (при частоте тактирования шины 80 МГц): %d
\n', ARR);
fprintf(fid, 'Количество элементов в массиве: %d \n\n', length(x));
% запись отсчетов для 12 разрядного DAC в текстовый файл
dlmwrite(filename, round(1736*x)+1736,'-append', 'delimiter', ',');
fclose(fid);

% построение графиков
plot(ts,x,'black','LineWidth',1), grid on, hold on;
plot(ts,fm,'--black','LineWidth',2), grid on;
title ('BPSK модуляция');
ylim([-2 2]);
xlabel('Время, сек'), ylabel('Амплитуда');
```

```
legend({'Модулированный сигнал';'Модулирующий сигнал'});

% Вычисление автокорреляционной функции с помощью функции myAutocorr
autocorr = Autocorr(x);

% Создание оси времени
time = -(length(x)-1):(length(x)-1);

% Построение графика автокорреляционной функции
figure;
stem(time, autocorr,'.',"black",'LineWidth',0.5);
xlabel('Смещение');
ylabel('Автокорреляция');
title('Функция автокорреляции');
grid on;

% Чтение отчетов принятого микрофоном сигнала
% Указываем путь к текстовому файлу
file_path = 'bpskADC.txt';

% Открываем текстовый файл для чтения
fileID = fopen(file_path, 'r');

% Чтение данных из текстового файла
dataadc = fscanf(fileID, '%f');
dataadc = (dataadc-mean(dataadc))./mean(dataadc);
% Закрываем файл
fclose(fileID);

% Создаем вектор времени для оси X
tsadc = 1:numel(dataadc);

% Построение графика
figure;
plot(tsadc, dataadc,'black','LineWidth',1);
xlabel('Отсчеты');
ylabel('Амплитуда');
title('График сигнала снятого с АЦП');
grid on;

crosscorr = Crosscorr(dataadc,x);

% Построение графика автокорреляционной функции
figure;
stem(tsadc, crosscorr,'.',"black",'LineWidth',0.5);
xlabel('Смещение');
ylabel('Корреляция');
ylim([-1 1]);
title('Функция корреляции');
grid on;
```

Приложение Б

Код на языке Python, определяющий координату и визуализирующий ее на координатной плоскости:

```
import tkinter as tk
import serial
import serial.tools.list_ports
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from matplotlib.patches import Circle

class TrilaterationApp:
    def __init__(self):
        self.ser = None  # Соединение с UART портом
        self.running = False  # Флаг для определения состояния программы
(запущена/приостановлена)
        self.distances = []  # Список расстояний

        # Известные точки и их координаты
        self.point1 = (0, 250)
        self.point2 = (250, 250)
        self.point3 = (250, 0)

        # Создание графика
        self.fig = Figure(figsize=(5, 5), dpi=120)
        self.ax = self.fig.add_subplot(111)

        # Отображение известных точек
        self.ax.plot(self.point1[0], self.point1[1], 'rs', label='Динамик 1')
        self.ax.plot(self.point2[0], self.point2[1], 'bo', label='Динамик 2')
        self.ax.plot(self.point3[0], self.point3[1], 'g^', label='Динамик 3')

        # Отображение орбит
        self.circle1 = Circle(self.point1, 0, fill=False, color='r')
        self.circle2 = Circle(self.point2, 0, fill=False, color='b')
        self.circle3 = Circle(self.point3, 0, fill=False, color='g')
        self.ax.add_patch(self.circle1)
        self.ax.add_patch(self.circle2)
        self.ax.add_patch(self.circle3)

        # Отображение неизвестной точки
        self.unknown_point, = self.ax.plot([], [], 'ko', label='Микрофон')

        # Настройка графика
```

```
        self.ax.set_aspect('equal')
        self.ax.set_xlim([-50, 300])
        self.ax.set_ylim([-50, 300])
        self.ax.set_xlabel('X (мм)')
        self.ax.set_ylabel('Y (мм)')
        self.ax.legend()

        # Создание окна выбора порта
        self.root = tk.Tk()
        self.root.geometry("800x600")
        self.root.title("Обработка данных учебного макета")

        # Получение доступных портов
        self.available_ports = [port.device for port in
serial.tools.list_ports.comports()]

        # Создание подписи для списка портов
        self.coordinate_units = tk.Label(self.root, text="COM порт:")
        #self.coordinate_units.grid(row=0, column=1, padx=3, pady=3)
        self.coordinate_units.place(x=612, y=20)

        # Создание выпадающего списка с доступными портами
        self.port_var = tk.StringVar(self.root)
        self.port_var.set("")  # По умолчанию не выбран порт
        self.port_menu = tk.OptionMenu(self.root, self.port_var,
*self.available_ports)
        self.port_menu.config(width=20, height=2)
        self.port_menu.place(x=610, y=50)

        # Кнопки для управления программой
        self.start_button = tk.Button(self.root, text="Старт",
command=self.start_program)
        self.start_button.config(width=10, height=2)
        self.start_button.place(x=612, y=100)

        self.pause_button = tk.Button(self.root, text="Пауза",
command=self.pause_program, state=tk.DISABLED)
        self.pause_button.config(width=10, height=2)
        self.pause_button.place(x=693, y=100)

        self.stop_button = tk.Button(self.root, text="Стоп",
command=self.stop_program, state=tk.DISABLED)
        self.stop_button.config(width=22, height=2)
        self.stop_button.place(x=611, y=520)

        # Переменная, значение которой меняется с помощью ползунка
        self.slider_var = tk.DoubleVar()
        self.slider_var.set(1)  # Установка значения по умолчанию
```

```
        # Создание текстового поля
        self.coordinate_label = tk.Label(self.root, text="Калибровка:")
        self.coordinate_label.place(x=612,y=250)

        # Создание ползунка
        self.slider = tk.Scale(self.root, variable=self.slider_var,
orient=tk.HORIZONTAL, from_=0, to=2, resolution=0.01,
command=self.slider_changed)
        self.slider.config(width=20,length=160)
        self.slider.place(x=611,y=270)

        # Создание текстового поля
        self.coordinate_label = tk.Label(self.root, text="Координата точки:")
        self.coordinate_label.place(x=612,y=340)

        self.coordinate_value = tk.StringVar()
        self.coordinate_text = tk.Label(self.root,
textvariable=self.coordinate_value)
        self.coordinate_text.place(x=612,y=360)



        # Добавление графика в интерфейс программы
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.root)
        self.canvas.get_tk_widget().place(x=0,y=0)

        self.plot_initialized = False

    def start_program(self):
        selected_port = self.port_var.get()  # Получить выбранный COM порт
        if selected_port:
            # Подключение к выбранному COM порту
            self.ser = serial.Serial(selected_port, 9600)  # Укажите
правильную скорость передачи данных

            # Отключение кнопки "Старт" и активация кнопок "Пауза" и
"Отключить"
            self.start_button.config(state=tk.DISABLED)
            self.pause_button.config(state=tk.NORMAL)
            self.stop_button.config(state=tk.NORMAL)

            self.running = True  # Установка флага состояния программы
            self.update_plot()

    def pause_program(self):
        self.running = not self.running  # Изменение флага состояния
программы
        if self.running:
            self.pause_button.config(text="Пауза")
```

```
        else:
            self.pause_button.config(text="Продолжить")

    def stop_program(self):
        self.running = False  # Установка флага состояния программы
        self.ser.close()  # Закрытие соединения с UART портом

        # Активация кнопки "Старт" и отключение кнопок "Пауза" и "Отключить"
        self.start_button.config(state=tk.NORMAL)
        self.pause_button.config(state=tk.DISABLED)
        self.stop_button.config(state=tk.DISABLED)

    def update_plot(self):
        if self.running:
            # Чтение расстояний от UART порта
            self.distances = []
            while True:
                start=self.ser.readline()
                start = start.decode('utf8')
                if start == 's\n':
                    print('start of conv\n')
                    break
            while len(self.distances) < 3:
                if self.ser.in_waiting > 0:
                    # Чтение данных из порта
                    data = self.ser.readline()
                    data = data.decode('utf8')
                    try:
                        distance = float(data)* self.slider_var.get()  #
Преобразовать прочитанные данные в число
                        self.distances.append(distance)
                    except ValueError:
                        continue
                    print(data)
            # Вычисление координат неизвестной точки
            result = self.trilaterate(self.point1, self.point2, self.point3,
self.distances[0], self.distances[1], self.distances[2])

            # Обновление орбит и координат неизвестной точки
            self.circle1.set_radius(self.distances[0])
            self.circle2.set_radius(self.distances[1])
            self.circle3.set_radius(self.distances[2])
            self.unknown_point.set_data(result[0], result[1])
            self.coordinate_value.set(f"({result[0]:.2f} мм, {result[1]:.2f}
мм)")

            if not self.plot_initialized:
                self.plot_initialized = True
                self.ax.set_xlim([-50, 300])
```

```
                self.ax.set_ylim([-50, 300])
                #self.ax.relim()
                self.ax.legend()

            # Обновление графика
            self.ax.figure.canvas.draw()
        # Планирование следующего обновления графика через 100 миллисекунд
        self.root.after(10, self.update_plot)

    def trilaterate(self, p1, p2, p3, d1, d2, d3):
        x1, y1 = p1
        x2, y2 = p2
        x3, y3 = p3

        A = 2 * (x2 - x1)
        B = 2 * (y2 - y1)
        C = d1**2 - d2**2 - x1**2 + x2**2 - y1**2 + y2**2
        D = 2 * (x3 - x2)
        E = 2 * (y3 - y2)
        F = d2**2 - d3**2 - x2**2 + x3**2 - y2**2 + y3**2

        x = (C*E - F*B) / (E*A - B*D)
        y = (C*D - A*F) / (B*D - A*E)

        return x, y

    def slider_changed(self, value):
        # Преобразовать значение ползунка в число и обновить переменную
        self.slider_var.set(float(value))

    def start(self):
        self.root.mainloop()

# Создание и запуск приложения
app = TrilaterationApp()
app.start()
```

Приложение В

Основной цикл программы, загруженной на микроконтроллер STM32:

```
while (1)
  {
//      debug1 = 0;
      if(num==ADCIn)
      {
//          for(uint16_t c = 0;c<ADCIn;c++)
//          {
//              debug1 = -buff[c];
//              HAL_Delay(3);
//          }
          //Передача значений принятых с микрофона по UART. Раскоментировать
при отладке.
          /*
          for(uint16_t c = 0;c<ADCIn;c++)
          {
              debug1 = buff[c];
              for( a=0; a<4; a++ ) micdebug[a] = '0';
              while( debug1>=1000 )    { micdebug[0]++; debug1 = debug1-
1000; }
              while( debug1>=100 )     { micdebug[1]++; debug1 = debug1-
100; }
              while( debug1>=10 )      { micdebug[2]++; debug1 = debug1-10; }
              micdebug[3] += debug1;
              HAL_UART_Transmit(&huart1,(uint8_t*)micdebug, a, 1000);
              HAL_UART_Transmit(&huart1,"\n", 1, 1000);
          }
          */
          //Конец сегмента для отладки
          //Вычисление постоянной составляющей и взаимной корреляции
          int32_t mean = calculateAverage(buff,ADCIn);
          for(uint16_t c = 0;c<ADCIn;c++)
          {
              buff[c] = buff[c]-mean;
          }
          autocorr(buff,Bark);
          //Конец сегмента
          num=0;
          //Пересылка расстояния по UART. Закоментировать при отладке
          for( a=0; a<3; a++ ) buf[a] = '0';
          while( distance>=100 )     { buf[0]++; distance = distance-100; }
          while( distance>=10 )       { buf[1]++; distance = distance-10; }
          buf[2] += distance;
          if (speakerNum==0){HAL_UART_Transmit(&huart1,"s\n", 2, 1000);}
          HAL_UART_Transmit(&huart1,(uint8_t*)buf, a, 1000);
          HAL_UART_Transmit(&huart1,"\n", 1, 1000);
          HAL_Delay(30);
          //Конец сегмента передачи расстояния по UART
          //Переключаем светодиод для индикации следующего замера
          HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_6);
          //Выбираем следующий динамик (необходимо раскоментировать
необходимое)
          /*(1)Переключение динамиков (основной режим работы*/
          speakerNum++;
          /*(2)Только первый*/
```

```
            //speakerNum=0;
            /*(3)Только второй-*/
            //speakerNum=1;
            /*(4)Только третий-*/
            //speakerNum=2;
            if (speakerNum > 2){speakerNum = 0;}
            //Включаем необходимый динамик
            switch(speakerNum)
            {
            case 0:
                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_SET);
                break;
            case 1:
                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_SET);
                break;
            case 2:
                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
                break;
            default :
                break;
            }

            //Повторно запускаем DMA
            HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, (uint32_t*)Bark, MasSize,
DAC_ALIGN_12B_R);
            //Повторно запускаем ADC
            HAL_ADC_Start_IT(&hadc1);
        }
```

Обработчик прерываний ADC:

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)

{
if(hadc->Instance == ADC1) //check if the interrupt comes from ACD1
    {
              //Заполняем массив значениями с микрофона
              if(num<(ADCIn))
              {
                  buff[num]=HAL_ADC_GetValue(&hadc1);
                  num++;
              }
              else
              {

              //Приостанавливаем передачу и прием данных
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET);

              //HAL_DAC_Stop_DMA(&hdac, DAC_CHANNEL_1);
              HAL_ADC_Stop_IT(&hadc1);
              }
    }
}
```

Автокорреляционная функция:

```
void autocorr(int32_t *mas1,const uint32_t *mas2)
{


    volatile int32_t maxcorr = 0;

    volatile int32_t ans = 0;

    volatile int32_t sdvig = 0;
    for(uint16_t t = 0; t<ADCIn; t++)
    {
        volatile int32_t vnutr = 0;
        for(volatile uint16_t i = 0;i<MasSize;i++)
        {
            sdvig = i+t;
            if(sdvig<ADCIn)
            {
            vnutr=vnutr+(mas1[sdvig])*((int16_t)mas2[i]-1736);
            }
        }
        corr = vnutr;
        if(corr>maxcorr){maxcorr=corr;ans=t;}
    }
    distance=round((ans*33/16));
}
```