

Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий



**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ  
РАБОТА БАКАЛАВРА**

**ПРИМЕНЕНИЕ НОВЫХ АЛГОРИТМОВ ОБРАБОТКИ РАДАРНЫХ  
ДАННЫХ В РЕШЕНИИ ПРОБЛЕМЫ НЕОДНОЗНАЧНОСТИ  
ОПРЕДЕЛЕНИЯ СКОРОСТЕЙ**

Студент гр. 3530901/60101 Е.С. Ильин

Санкт-Петербург

2020

Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский политехнический университет Петра Великого Институт  
компьютерных наук и технологий  
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Работа допущена к защите  
Заведующий кафедрой  
\_\_\_\_\_ В.М. Ицыксон  
« \_\_\_ » \_\_\_\_\_ 2020 г.

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

### **Применение новых алгоритмов обработки радарных данных в решении проблемы неоднозначности определения скоростей**

по направлению 09.03.01 «Информатика и вычислительная техника»  
по образовательной программе  
09.03.01\_01 «Вычислительные машины, комплексы, системы и сети»

Выполнил  
студент гр.3530901/60101

Е.С. Ильин

Научный руководитель,  
старший преподаватель

А.А. Федотов

Консультант по нормоконтролю,  
старший преподаватель

С.А. Нестеров

Санкт-Петербург  
2020

**САНКТ – ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**Институт компьютерных наук и технологий**

**Высшая школа интеллектуальных систем**

**и суперкомпьютерных технологий**

**УТВЕРЖДАЮ**

“ \_\_\_\_\_ ” \_\_\_\_\_ 2020 г.

*Директор ВШИСиСТ* \_\_\_\_\_ **В.М.Ицыксон**

**ЗАДАНИЕ**

**по выпускной бакалаврской работе**

студенту Ильину Евгению Сергеевичу

1. Тема работы Применение новых алгоритмов обработки радарных данных в решении проблемы неоднозначности определения скоростей
2. Срок сдачи студентом законченной работы 15.06.2020
3. Исходные данные к проекту (работе) \_\_\_\_\_
1. Содержание научной статьи, описывающей алгоритм
2. Список доступных аппаратных платформ для реализации  
Функциональные требования: \_\_\_\_\_
1. Радарная установка настроена и обладает требуемыми параметрами
2. Стенд собирает данные и передает их для дальнейшей обработки  
Требования к средствам реализации: \_\_\_\_\_
1. Программный язык написания программы взаимодействия с радаром: C++
4. Содержание расчетно-пояснительной записки \_\_\_\_\_
1. Теоретическая база
2. Выбор аппаратной платформы
3. Работа с оборудованием
4. Запуск и настройка сенсора используя API

## 5. Тестирование установки

5.Дата выдачи задания 28.02.2020\_\_\_\_\_

Руководитель \_\_\_\_\_ ( Федотов А.А. )  
(подпись руководителя) (Фамилия и инициалы)

Задание принял к исполнению “ 28 ” февраля 2020 г.

\_\_\_\_\_ ( Ильин Е.С. )  
(подпись студента) (Фамилия и инициалы)

## **РЕФЕРАТ**

На 50 с., 25 рисунков, 1 таблицу, 4 приложения.

**УСТРОЙСТВО, РАДАР, МИЛЛИМЕТРОВЫЙ РАДАР, FMCW РАДАР,  
ОТСЛЕЖИВАНИЕ НЕСКОЛЬКИХ ЦЕЛЕЙ**

Выпускная квалификационная работа посвящена изучению технологии FMCW радаров миллиметрового диапазона, построению экспериментального стенда, обладающего требуемыми параметрами, соответствующими результатам исследования алгоритма обработки радарных данных. Результатом работы является рабочий стенд, обладающий необходимыми параметрами сигнала, собирающий и передающий данные для дальнейшей постобработки.

## **THE ABSTRACT**

50 pages, 25 pictures, 1 table, 4 appendices.

**DEVICE, RADAR, MMWAVE RADAR, FMCW RADAR, MULTITARGET  
DETECTION**

The final qualification work is devoted to the studying of FMCW radar technology, building experimental device, with required parameters, corresponding to the research on radar data processing algorithm. Result of this work is an operating unit, capable of capturing and transmitting the data for further post-processing.

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>7</b>
<b>1. ТЕОРЕТИЧЕСКАЯ БАЗА.....</b>	<b>9</b>
1.1 Принцип работы FMCW радара .....	9
1.2 Описание алгоритма.....	19
1.3 Определение ключевых параметров .....	20
<b>2. ВЫБОР АППАРАТНОЙ ПЛАТФОРМЫ .....</b>	<b>22</b>
2.1 Анализ предоставленного оборудования .....	22
2.2 Выводы по разделу.....	23
<b>3. РАБОТА С ОБОРУДОВАНИЕМ.....</b>	<b>24</b>
3.1 Описание и сборка.....	24
3.2 Подключение к ПК и настройка .....	26
3.3 Проверка работоспособности.....	28
3.4 Алгоритм на практике.....	29
3.5 Получение и анализ сырых данных.....	31
<b>4. ЗАПУСК И НАСТРОЙКА СЕНСОРА ИСПОЛЬЗУЯ API.....</b>	<b>33</b>
4.1 Описание файла конфигурации .....	33
4.2 Программа загрузки параметров .....	33
<b>5. ТЕСТИРОВАНИЕ УСТАНОВКИ.....</b>	<b>37</b>
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>39</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>40</b>
<b>ПРИЛОЖЕНИЕ 1. ПРИМЕР ФАЙЛА КОНФИГУРАЦИИ .....</b>	<b>42</b>
<b>ПРИЛОЖЕНИЕ 2. ПОРЯДОК ВЫЗОВА ФУНКЦИЙ.....</b>	<b>45</b>
<b>ПРИЛОЖЕНИЕ 3. ЛИСТИНГ КОДА ОСНОВНОЙ ФУНКЦИИ .....</b>	<b>46</b>
<b>ПРИЛОЖЕНИЕ 4. РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ .....</b>	<b>49</b>

## ВВЕДЕНИЕ

Радар, или Радиолокационная станция (РЛС) – это система для обнаружения, определения скорости, дальности и геометрических параметров воздушных, морских и наземных целей. Основным принципом работы является в излучении радиоволн, и последующей регистрации их отражений от объектов, или регистрации сигнала, излучаемого самим объектом радиолокации.

Классифицировать РЛС можно по разным параметрам [1]:

1. по типу сигнала, принимаемым установкой (РЛС с пассивным ответом, или первичные; РЛС с активным ответом, или вторичные; смешанные);
2. по диапазону используемых радиоволн (РЛС миллиметрового, сантиметрового, дециметрового, метрового и декаметрового диапазона);
3. по виду излучаемого (зондирующего) сигнала (РЛС с непрерывным (ЛЧМ или немодулированным) или импульсным излучением);
4. числу используемых каналов приема и излучения сигналов (одно- и многоканальные).

Точное определение местоположения объекта, как и его относительной и радиальной скоростей, необходимо во множестве технических приложений, в том числе автомобильной связи. Причем в отличие от ультразвукового или оптического метода, микроволновый радар не так ограничен в максимальной рабочей дистанции [2].

Целью данной работы являлось проведение эксперимента по реализации алгоритма, приведенного в пункте «описание алгоритма», с расчетом необходимых параметров, а также разработка приложения автоматической настройки сенсора с пользовательскими настройками.

Для достижения указанной цели, нужно решить следующие задачи:

1. Изучить основные принципы работы FMCW радара и выделить ключевые параметры, необходимые для реализации алгоритма.
2. Провести анализ имеющихся на рынке радаров миллиметрового диапазона и выбрать основу для экспериментального стенда.
3. Провести сборку, установку и освоить работу с радарной системой.
4. Разработать ПО для автоматического запуска, подключения и настройки сенсора.

## 5. Протестировать полученную установку.

Результатом работы является экспериментальная установка, которая, обладая необходимыми параметрами, собирает и передает радарные данные для дальнейшей обработки. Задание параметров, а также запуск установки должны производиться автоматически, с помощью специально разработанного ПО.



## 1. ТЕОРЕТИЧЕСКАЯ БАЗА

Так как суть данной работы заключается в практическом применении данных, полученных в ходе исследования [3], нужно определить какие из параметров системы имеют значимость и должны быть учтены при аппаратной реализации.

### 1.1 Принцип работы FMCW радара

Так как FMCW радар является ключевым объектом работы, для понимания дальнейших действий необходимо изучить какие принципы лежат в основе его работы, и каким образом происходит определение таких характеристик цели, как дальность, скорость, и угол отклонения.

Аббревиатура FMCW расшифровывается как Frequency Modulated Continuous Wave, что означает, что у такого радара зондирующим является непрерывный частотно модулированный сигнал. К ключевым особенностям такого радара относится [4]:

1. Возможность измерения крайне низких частот, сравнимых с длиной излучаемой волны.
2. Высокая точность измерения расстояния до цели.
3. Возможность одновременно оценить как расстояние, так и относительную скорость цели.
4. Возможность обнаружения неподвижных целей.
5. Обработка сигнала в диапазоне низких частот, что существенно упрощает задачу реализации схемы обработки.

FMCW радар излучает сигнал, который называется «чирп». Чирп – это синусоида, частота которой линейно изменяется со временем, как показано на рис. 1.1.

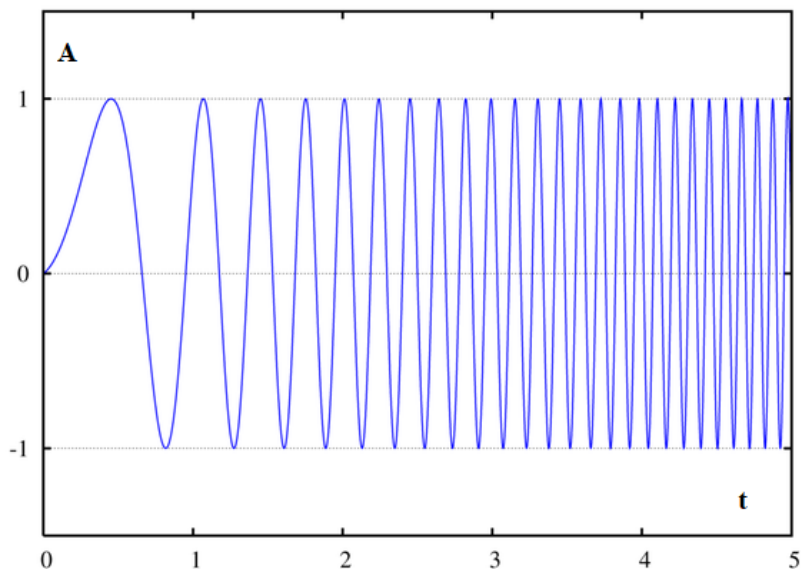


Рис. 1.1. График зависимости амплитуды сигнала от времени

Для удобства представления данный сигнал можно представить как график отношения частоты ко времени, который, в силу линейности закона изменения частоты, будет представлять собой прямую, как представлено на рис. 1.2.

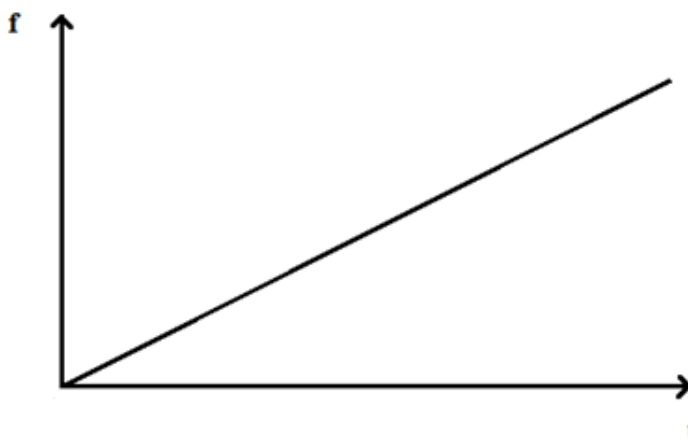


Рис. 1.2. График зависимости частоты сигнала от времени

Чирп характеризуется такими параметрами, как начальная частота  $f_c$ , ширина спектра  $B$  и длительность  $T_c$ . Наклон чирпа  $S$  определяет темп его роста. Например, при ширине спектра в 4 ГГц и длительности 40 микросекунд, наклон чирпа будет составлять 100 МГц/мкс. Как будет доказано далее, ширина спектра и наклон чирпа являются важными параметрами, определяющими эффективность всей радарной системы.

Упрощенная схема FMCW радара представлена на рис. 1.3.

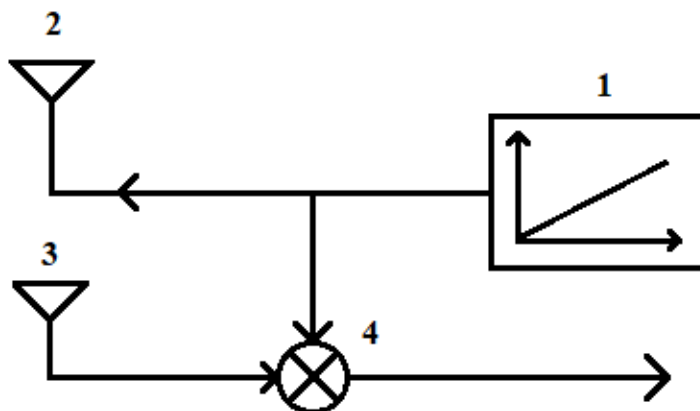


Рис. 1.3. Упрощенная схема FMCW радара

В данном примере приведен радар с одной антенной передатчика и приемника. Обозначения обозначенных на рис. 1.3. элементов, а также поэтапное описание работы радара:

1. Синтезирующее устройство, формирующее чирп.
2. Антенна передатчика, излучающая полученный сигнал.
3. Антенна приемника, на которую попадает чирп, отраженный от объекта.
4. Смеситель, на вход «Гетеродин» которого подается чирп передатчика, а на вход «ВЧ» - чирп, принятый антенной приемника. На выходе же получается сигнал, именуемый «ПЧ», или промежуточная частота.

Полученный в итоге ПЧ сигнал обладает постоянной частотой, равной разности частот сигналов передатчика и приемника, а также фазой, равной разности фаз входных сигналов смесителя. Работу смесителя можно представить в виде уравнения:

$$x_{tx} = \sin[\omega_1 + \varphi_1]$$

$$x_{rx} = \sin[\omega_2 + \varphi_2]$$

$$x_{out} = \sin[(\omega_1 - \omega_2) + (\varphi_1 - \varphi_2)] \quad (1)$$

В качестве примера рассмотрим случай одиночной неподвижной цели. Отобразим полученные чирпы приемника и передатчика на графике частоты ко времени, рис. 1.4.

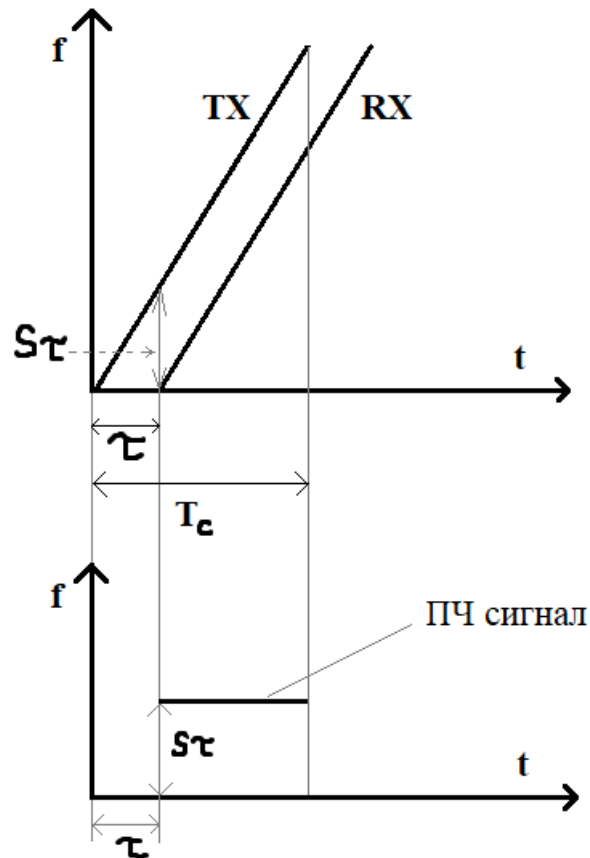


Рис. 1.4. Графики зависимости частоты от времени сигналов приемника, передатчика и ПЧ сигнала

В верхней части рис. 1.4. изображены чирп передатчика и идентичный ему, но сдвинутый по времени на величину  $\tau$  чирп приемника. График ПЧ сигнала в нижней части рис. 1.4. получен путем вычитания графика сигнала приемника из графика передатчика, так как частота сигнала на выходе смесителя равна разности частот передатчика и приемника. Однако информативной является только часть, лежащая после интервала  $\tau$ , до конца интервала  $T_c$ . Из этого можно сделать следующий вывод: одиночная статичная цель, находящаяся перед радаром, производит ПЧ сигнал постоянной частоты. Частота этого сигнала равна:

$$f_{\text{ПЧ}} = S\tau = \frac{S2d}{c} \quad (2)$$

Последнее равенство объясняется тем, что время задержки между излучением сигнала приемником и попаданием его отражения на антенну передатчика, равное  $\tau$ , можно представить как удвоенное расстояние до объекта, поделенное на скорость света  $c$ . Отсюда следует вывод, что частота полученного ПЧ сигнала прямо пропорциональна расстоянию до цели.

При нахождении перед радаром нескольких целей, частота соответствующего ему ПЧ сигнала будет тем выше, чем дальше находится объект. Но так как на выходе смесителя получается единый сигнал, необходимо провести над ним преобразование Фурье, для определения частот составляющих сигналов. В результате преобразования на спектрограмме ПЧ сигнала можно будет наблюдать пики, соответствующие наблюдаемым объектам. Такое преобразование также называется «Range FFT», или БПФ расстояния. Пример результата преобразования такого сигнала приведен на рис. 1.5.

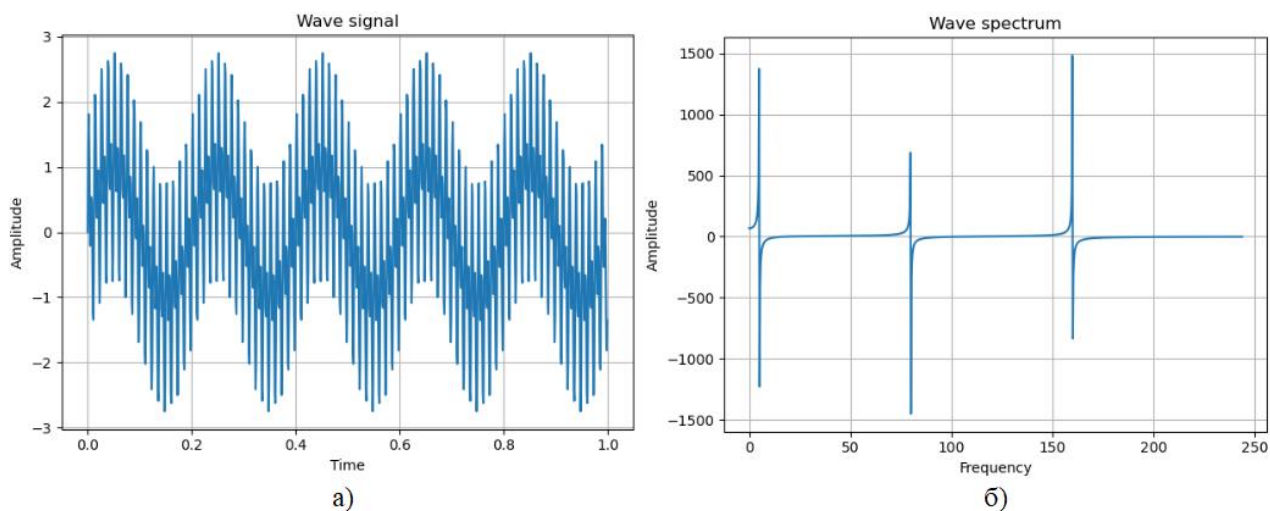


Рис. 1.5. а) Сигнал с тремя разными составляющими частоты; б) результат БПФ с тремя соответствующими пиками

Из этого метода так же следует и разрешение радара по расстоянию, так как для того, чтобы при БПФ два сигнала были выражены двумя пиками, разница их частот должна удовлетворять условию:

$$\Delta f > \frac{1}{T} \quad (3)$$

Следовательно, для улучшения разрешения по дальности, следует увеличить длительность чирпов, так как длительность ПЧ сигнала определяется как  $T_c - \tau$ .

Воспользовавшись формулами (1) и (2), получаем:

$$\Delta f = \frac{S2\Delta d}{c}$$

$$\Delta f > \frac{1}{T_c}$$

$$\frac{S2\Delta d}{c} > \frac{1}{T_c}$$

$$\Delta d > \frac{c}{2ST_c}$$

Так как произведение наклона на длительность является шириной спектра, получаем неравенство:

$$\Delta d > \frac{c}{2B}, \quad (4)$$

Где  $\Delta d$  – минимальная дистанция между целями, при которой их возможно различить как два отдельных объекта.

Так как ПЧ сигнал – аналоговый, для обработки его необходимо пропустить через АЦД с ФНЧ, а затем передать сигнальному процессору для выполнения необходимых преобразований и расчетов. При этом, частота семплирования АЦП ограничивает максимальную дальность работы радара, влияя на ширину спектра ПЧ сигнала. Из формулы (1) можно сделать следующий вывод:

$$F_s \geq \frac{S2d_{max}}{c},$$

тогда

$$d_{max} = \frac{F_s c}{2S} \quad (5)$$

Следовательно, для двух чирпов с равной шириной спектра, но разной длительностью, чирп с большим  $T_c$  потребует меньшей частоты семплирования АЦП, но займет больше времени.

При анализе расстояния до цели проводилось БПФ ПЧ сигнала, то есть задействовалась только реальная часть так называемой комплексной амплитуды, так как анализировалась лишь частота. Оценка скорости же основана на анализе фазы ПЧ сигнала – комплексной составляющей. Как было указано в уравнении (1), фаза ПЧ сигнала, создаваемого объектом, равна разности фаз сигналов приемника и передатчика. Если цель переместится, то изменится не только частота ПЧ сигнала, но и его фаза, так как новый чирп придет с отличающейся

от предыдущего задержкой. Формула зависимости разности фаз от пройденного объектом расстояния []:

$$\Delta\varphi = 2\pi f_c \Delta\tau = \frac{4\pi\Delta d}{\lambda} \quad (6)$$

Рассмотрим крайне малое перемещение цели, сравнимое с длиной волны несущего сигнала. Для стандартного 77ГГц радара с наклоном чирпа в 50МГц/мкс и длительностью в 40 мкс перемещение цели на 1мм будет являться примерно четвертью длины волны. Тогда, соответствии с формулами (2) и (6), получается:

$$\Delta\varphi = \frac{4\pi \times \frac{\lambda}{4}}{\lambda} = \pi = 180^\circ$$

$$\Delta f = \frac{50 \times 10^{12} \times 2 \times 1 \times 10^{-3}}{3 \times 10^8} = 333\text{Гц}$$

Из полученных значений видно, что при столь малом перемещении частота ПЧ сигнала практически не изменилась (приращение составляет порядка 1.3%), тогда как фаза сигнала изменилась существенным образом.

Для измерения скорости объекта излучается два чирпа, с интервалом  $T_c$ . Несмотря на то, что в результате БПФ расстояния будет виден лишь один пик, разность фаз полученных сигналов будет соответствовать пройденному расстоянию, которое равно  $vT_c$ , где  $v$  – скорость цели. Из (6) следует:

$$\omega = \frac{4\pi v T_c}{\lambda} \Rightarrow v = \frac{\lambda \omega}{4\pi T_c}$$

Максимальная же скорость определяется неравенством []:

$$\frac{4\pi v T_c}{\lambda} < \pi \Rightarrow v < \frac{\lambda}{4T_c},$$

следовательно,

$$v_{\max} = \frac{\lambda}{4T_c}.$$

Отсюда можно сделать вывод, то для увеличения максимальной определяемой скорости достаточно уменьшить интервал между чирпами.

Для определения скорости двух и более равноудаленных целей двух чирпов недостаточно, так как значение в пике БПФ расстояния будет содержать составляющие обоих объектов. Для этого применяется последовательность не из двух, а из  $N > 2$  чирпов, которая называется фрейм. Тогда будет получено не одно значение, а последовательность значений фазеров, соответствующих пикам на БПФ расстояния. Над этим дискретным набором значений также можно произвести БПФ, которое называется доплеровским. Пики, полученные в результате такого преобразования, отражают частоту изменения фазы  $\omega$ , и позволяют определить скорость сразу нескольких объектов:

$$v_1 = \frac{\lambda \omega_1}{4\pi T_c}, v_2 = \frac{\lambda \omega_2}{4\pi T_c}$$

Разрешение по скорости, из свойств БПФ, определяется неравенством:

$$\Delta v > \frac{\lambda}{2NT_c},$$

где  $N$  – количество семплов, равное количеству чирпов, а  $T_c$  – время между чирпами. Тогда произведение  $NT_c$  можно заменить на  $T_F$  – общую продолжительность фрейма:

$$v_{res} = \frac{\lambda}{2T_F}$$

В FMCW радарх применяется так называемое двумерное БПФ. Оцифрованные данные с каждого чирпа фрейма сохраняются в матрицу, как показано на рис. 1.6.

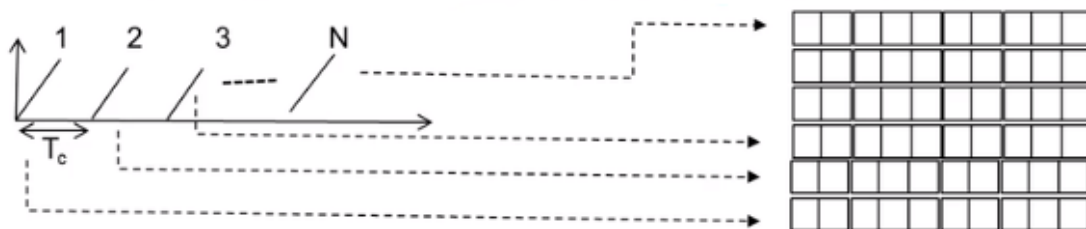


Рис. 1.6. Сохранение данных чирпов в матрицу

После того, как только данные с чирпа получены, СП производит БПФ расстояния, в итоге получается матрица как на рис. 1.7.



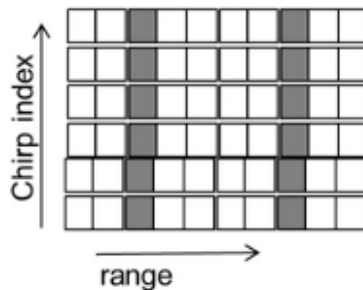


Рис. 1.7. Матрица данных с чирпов после БПФ расстояния

Когда матрица полностью заполнена и столбцы пиков окончательно сформированы, над ними производится доплеровское БПФ, результат которого приведен на рис. 1.8.

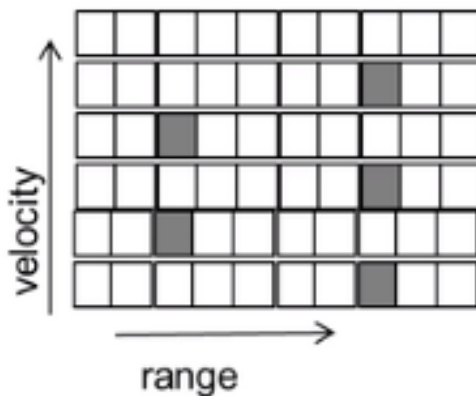


Рис. 1.8. Результат доплеровского БПФ

Данная последовательность действий называется двумерное БПФ.

Последняя характеристика цели, которую осталось определить – угол отклонения. Для его определения потребуется более одной антенны приемника, так как оценка угла основана на том факте, что расстояние от цели до разных антенн слегка различается. А как было установлено ранее, даже самое малое изменение расстояние существенно влияет на фазу сигнала. Пример для двух передатчиков приведен на рис. 1.9.

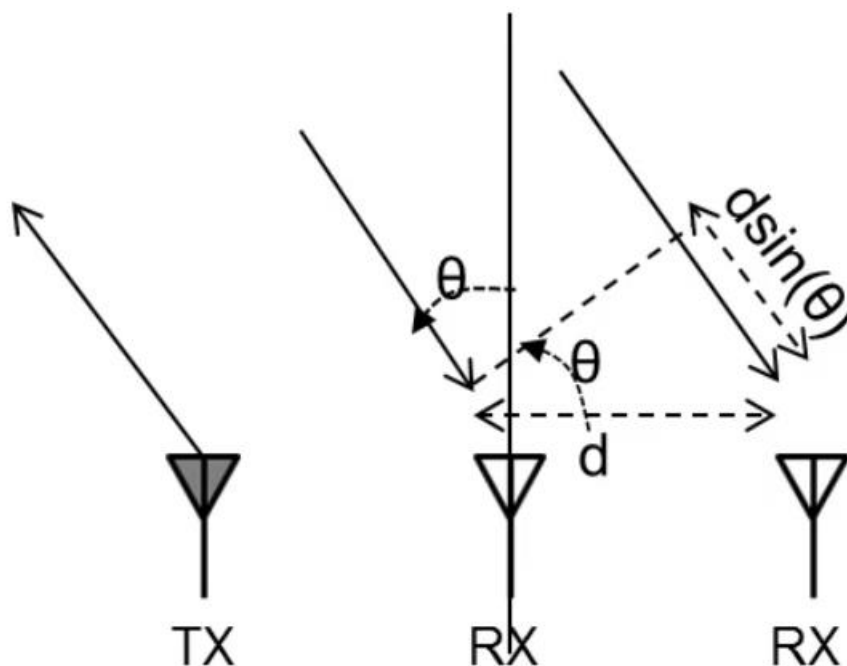


Рис. 1.9. Передача и прием сигнала радаром с 1 передатчиком и 2 приемниками

После передачи фрейма и двумерного БПФ полученных сигналов, пики у обоих приемников будут в одном и том же месте, однако их фазы будут различаться, на разницу пути сигнала, зависящую от расстояния между антеннами и углом отклонения:

$$\omega = \frac{2\pi d \sin(\theta)}{\lambda} \Rightarrow \theta = \sin^{-1} \left( \frac{\lambda \omega}{2\pi d} \right)$$

Следует отметить, что в отличие от предыдущих величин, зависимость разности фаз от угла наклона не линейна. Также, с увеличением угла  $\theta$  уменьшается чувствительность к его изменению, вплоть до нулевой чувствительности при  $\theta = 90^\circ$ , т.к. при таком значении синус угла будет равен нулю. Данный метод определения угла отклонения работает только для одиночной цели, если целей две и более, то необходим иной подход. Аналогично тому, как для определения скорости нескольких целей вместо двух чирпов использовалось большее количество, для определения угла отклонения нескольких целей воспользуемся не двумя, а большим количеством приемников. Тогда на каждом из них получим фазер с различной разностью фаз. Тогда произведя над ними БПФ, получим пики, соответствующие разности фаз для разных объектов. Такое БПФ называется угловым. Полученные значения углов будут определяться следующими формулами:

$$\theta_1 = \sin^{-1} \left( \frac{\lambda \omega_1}{2\pi d} \right), \theta_2 = \sin^{-1} \left( \frac{\lambda \omega_2}{2\pi d} \right)$$

Угловое разрешение, опять же в силу свойств БПФ, будет равняться:

$$\theta_{res} = \frac{\lambda}{N d \cos(\theta)}$$

Причем заметим, что в данном случае, в отличие от остальных, разрешение по параметру зависит от самого параметра. Это значит, что чем больше угол  $\theta$ , тем хуже разрешение по углу, и наоборот, наилучшее разрешение достигается при  $\theta=0$ , то есть, когда цели находятся прямо перед радаром.

Подытожив, выделим влияние параметров сигнала на эффективность работы радара:

$$v_{max} = \frac{\lambda}{4T_c} \quad (6)$$

$$v_{res} = \frac{\lambda}{2T_F} \quad (7)$$

$$d_{res} = \frac{c}{2B} \quad (8)$$

## 1.2 Описание алгоритма

Алгоритм обработки принятого сигнала включает одномерное быстрое комплексное преобразование Фурье с окном Хэмминга (Hamming Window), оценку частот максимумов амплитудного спектра, расчет дальностей и скоростей целей по оценке частот максимумов и применение алгоритма спаривания дальности и скорости целей. На промежутке, где частота излучаемого сигнала постоянна ( $T_0$ ) определяется скорость всех целей по максимумам амплитудного спектра. Разрешающая способность определения частоты определяется длительностью соответствующего промежутка. Таким образом, разрешающая способность по скорости и дальности соответственно равны

$$\delta V = \frac{c}{2T_0 f_0}; \delta R = \frac{c}{2\Delta f}$$

Каждой доплеровской частоте цели соответствует 2D-матрица частот биений для промежутков  $T_1$  и  $T_2$ .

### 1.3 Определение ключевых параметров

Как было установлено в разделе 1.1, ключевыми параметрами миллиметрового FMCW радара, напрямую влияющими на его работу, являются параметры генерируемого зондирующего сигнала. Однако, помимо этого при выборе платформы для аппаратной реализации необходимо также учитывать и следующее:

1. Устройство должно поддерживать желаемую ширину спектра ПЧ сигнала.
2. Устройство должно быть способно генерировать необходимый наклон чирпа.
3. Необходимо учитывать требования устройства к минимальному времени простоя между чирпами, которое необходимо генератору для возврата к опорной частоте.
4. Так как для получения фазового спектра (в п.1.1 - доплеровское БПФ) необходимо сначала собрать данные об амплитудном спектре каждого из чирпов фрейма, которых в реальных условиях может быть более тысячи, то есть устройство должно обладать достаточным объемом памяти.
5. Так как реальное устройство накладывает некоторые ограничения на те или иные параметры сигнала, для получения нужных значений разрешения и максимальных обрабатываемых дальности, скорости, и угла отклонения, приходится подстраивать параметры сигнала. Иногда приходится жертвовать, например, максимальной дальностью для лучшего разрешения, и наоборот.

Тогда, опираясь на данные компьютерного эксперимента, проведенного в рамках исследования [3], а также учитывая применяемую форму зондирующего сигнала, приведенного на рис. 1.10, составим таблицу необходимых параметров сигнала.

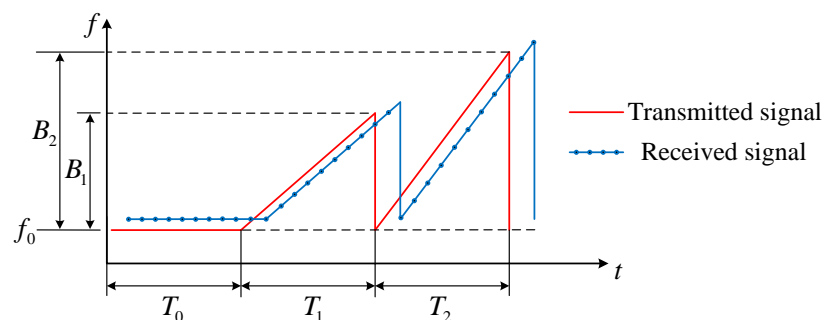


Рис. 1.10. Применяемая последовательность чирпов.

## Требуемые параметры системы

	Constant frequency	First chirp	Second chirp
Modulation Period, ms	7 ( $T_0$ )	7 ( $T_1$ )	7 ( $T_2$ )
Sampling Frequency, MHz	1,2	1,2	1,2
FFT point	8400	8400	8400
Carrier Frequency $f_0$ , GHz	76,5	76,5	76,5
Sweep Bandwidth, MHz	0	350 ( $\Delta f_1$ )	500 ( $\Delta f_2$ )

## 2. ВЫБОР АППАРАТНОЙ ПЛАТФОРМЫ

### 2.1 Анализ предоставленного оборудования

Для реализации необходимо выбрать одно из решений, предложенных на рынке миллиметровых FMCW радаров. Для этого был произведен сравнительный анализ представителей различных крупных производителей:

1. Texas Instruments – AWR1243 с внешним СП.
2. Infineon – position2go.
3. Silicon Radar – SiRad easy.

Для выбора подходящей платформы сравним представленные решения по интересующим нас параметрам. Сразу стоит отметить, что Infineon position2go обладает частотой зондирующего сигнала 24ГГц [5] при требуемых 76,5ГГц, а значит данное решение нам точно не подходит.

SiRad easy поставляется как с 24ГГц, так и 120ГГц и обладает некоторыми преимуществами [6]:

1. Наличие Wi-Fi модуля упрощает эксплуатацию устройства, хотя и накладывает некоторые ограничения.
2. Гибкая настройка и информативное ПО постобработки.
3. Готовые профили настроек для различных сценариев использования.
4. Возможность настройки канала связи, например изменение порта или адреса соединения.

Однако, обладает ограниченным объемом памяти и не способен вести длительную запись, что довольно важно для проведения разного рода испытаний.

AWR1243 обладает диапазоном рабочих частот от 76ГГц до 81ГГц [7], работает в паре с внешним процессором, передавая ему сырые данные через CSI2. В качестве такого внешнего устройства можно использовать DCA1000, либо TWR1400. Последнее, хотя и позволяет вести обработку в реальном времени, может записывать данные лишь в ограниченную внутреннюю память, которая для начала новой записи должна быть очищена.

В связке с внешним процессором DCA1000, в свою очередь, возможно вести практически не ограниченную по времени запись, поскольку данные на физическом передаются через Ethernet-соединение на ПК [8], объем памяти которого практически не ограничен.

Помимо этого, решение Texas Instruments обладает еще одним важным плюсом: TI предоставляет библиотеки для работы с фронтендом сенсора через специальные API, что позволяет разработать пользовательские приложения не только для обработки данных, но и для взаимодействия с самим радаром.

## **2.2 Выводы по разделу**

Основываясь на вышеперечисленных качествах, а также в связи с заданием, было решено в качестве основной аппаратной платформы использовать решение Texas Instruments – AWR1243 с внешним процессором DCA1000.

### 3. РАБОТА С ОБОРУДОВАНИЕМ

Экспериментальный стенд было решено построить на основе радарной платы AWR1243 и внешнего процессора DCA1000, так как в самом радаре отсутствуют как сигнальный процессор, так и микроконтроллер. Данное решение было выбрано в виду подходящего диапазона рабочих частот, а также возможности вести длительную, или даже непрерывную запись.

#### 3.1 Описание и сборка

Стенд состоит из двух плат, изображенных на рис. 3.1 и 3.2.

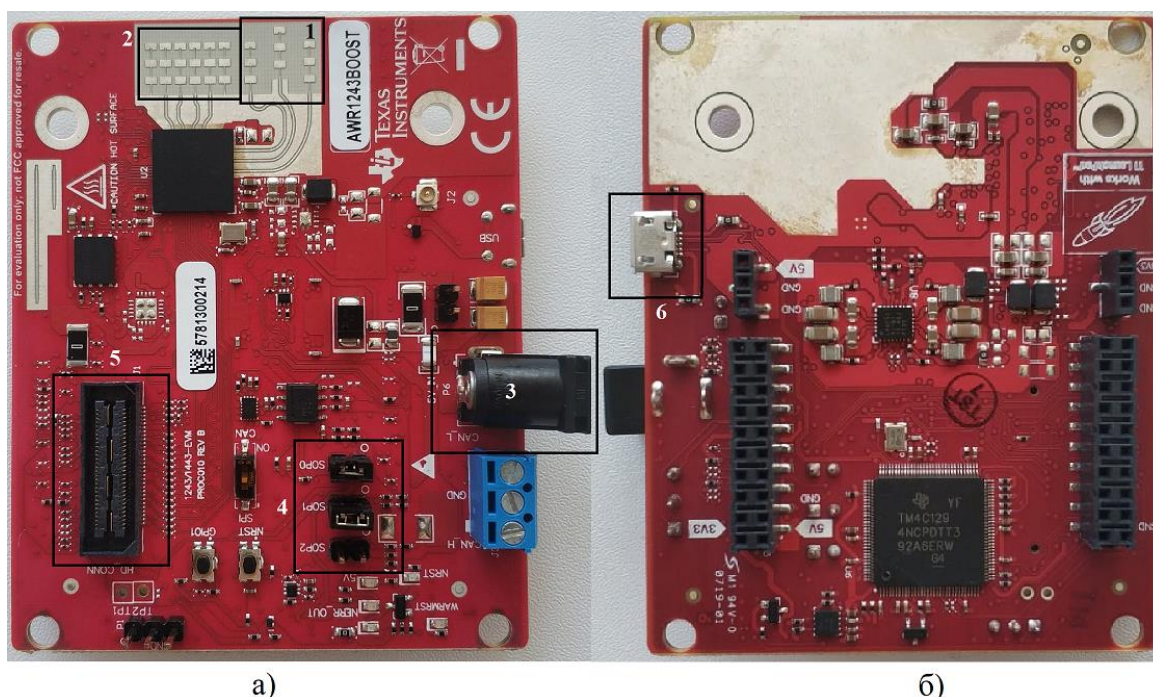


Рис. 3.1. AWR1243; а) вид спереди; б) вид сзади

Ниже описаны обозначенные на рис. 3.1 основные элементы, с которыми, в основном, происходит взаимодействие в данной работе:

1. Антенны передатчика.
2. Антенны приемника.
3. Разъем power jack для подключения внешнего источника питания 5В/2.5А.
4. Блок контактов, отвечающие за выбор режима работы платы. Разным режимам соответствуют разные комбинации замыкания.
5. Разъем для подключения внешнего процессора.
6. Разъем mini-USB для подключения к ПК.



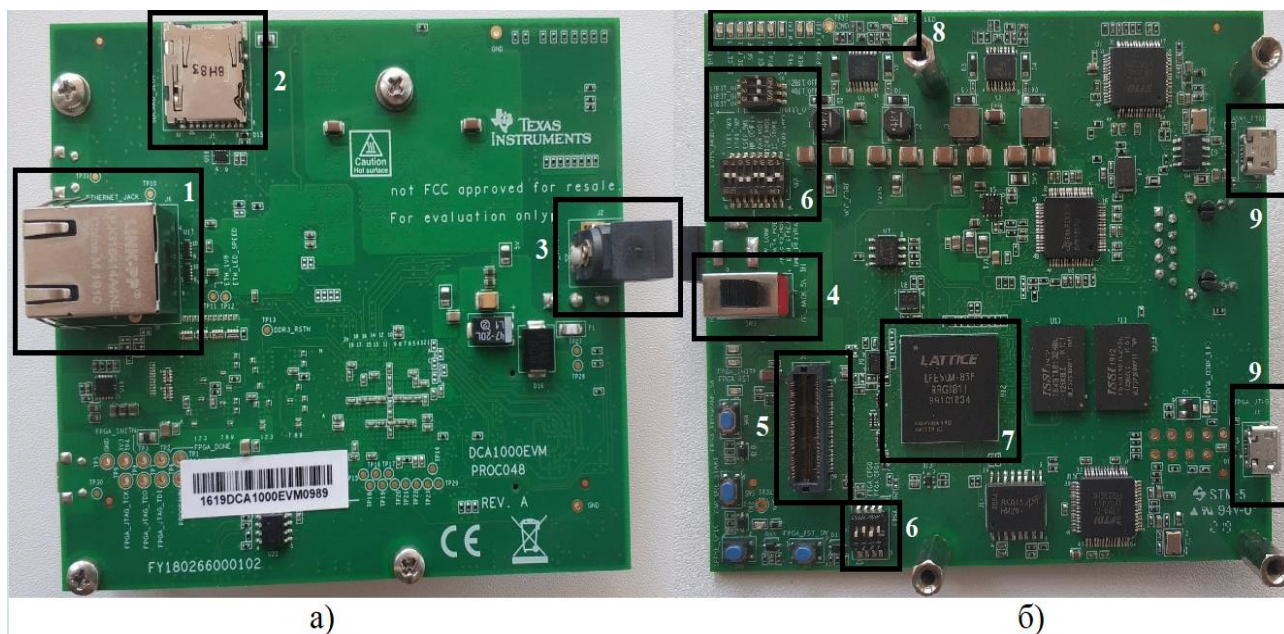


Рис. 3.2. DCA1000; а) вид сзади; б) вид спереди

Ниже приведено описание выделенных на рис. 3.2 элементов:

1. Ethernet-разъем, по которому данные передаются на ПК.
2. Разъем для карты памяти Micro-USB.
3. Разъем power jack для подключения внешнего источника питания.
4. Переключатель режима питания: от внешнего источника или от радара.
5. Разъем для подключения радара.
6. В данных областях располагаются переключатели для аппаратной конфигурации платы.
7. Основная ПЛИС – LATTICE FPGA.
8. Светодиоды индикации.
9. Разъемы mini-USB для подключения к ПК, нижний для управления платой через JTAG, верхний для настройки через поставляемое ПО.

В данном случае переключатель питания выставлен в режим питания от внешнего источника.

Платы соединяются друг с другом с помощью четырех ножек, которые можно увидеть на рис. 3.2.б возле элементов 6, 8 и 9. Помимо этого, в комплекте поставляются специальные опоры для более устойчивого положения плат на поверхности. Собранный установка с соединенными платами представлена на рис. 3.3.

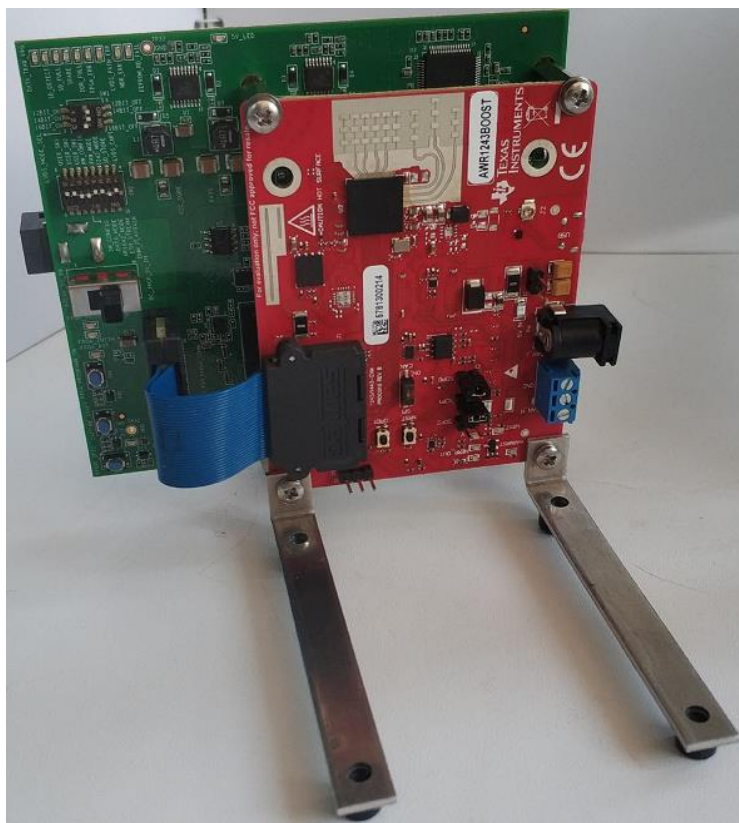


Рис. 3.3. Собранная установка на опорах

### 3.2 Подключение к ПК и настройка

Для подключения к ПК каждая из плат подключается к хост-ПК с помощью кабеля USB – mini-USB, DCA1000 дополнительно подключается через ethernet-кабель. При первом подключении установка не будет корректно отображаться в диспетчере устройств, так как не установлены соответствующие драйвера. Пример содержимого окна диспетчера устройств показан на рис. 3.4.

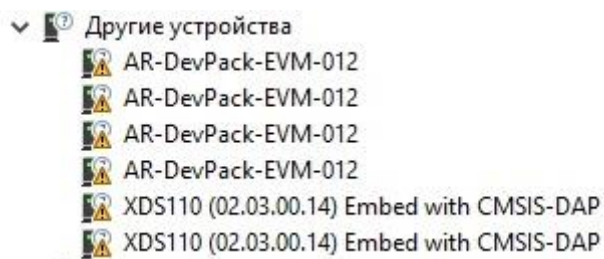


Рис. 3.4. Устройства неверно определяются до установки драйверов

После установки и обновления драйверов до последней версии устройства начинают определяться корректно, и теперь в диспетчере отображается COM-порт, к которому подключено устройство. Номер порта для каждого ПК получается свой, поэтому важно верно установить драйвера, потому что позже

будет необходимо указать номер СОМ-порта, к которому подключен радар. Корректное отображение представлено на рис. 3.5.



Рис. 3.5. Содержимое окна диспетчера устройств после установки драйверов

Драйвера для устройств семейства xWR поставляются вместе с ПО.

Для работы с платой было установлено специальное ПО, поставляемое TI, mmWave Studio. Однако, для корректной работы радара и самой студии, необходимо установить также [9]:

1. Matlab Runtime Engine v8.5.1.
2. Matlab 2015A, для установки и работы необходимой версии Runtime Engine.
3. XDS Emulation Software Package v6.0.579.0
4. Code Composer Studio v7.1

Для указанных программ необходимо установить конкретную приведенную версию, иначе корректная работа устройства производителем не гарантирована.

Так как DCA1000 запрограммирована отправлять данные через ethernet на определенный адрес, необходимо было настроить свойства подключения. Необходимый адрес – IPv4 192.168.33.30, маска – 255.255.255.0. Процесс настройки соединения показан на рис. 3.6.

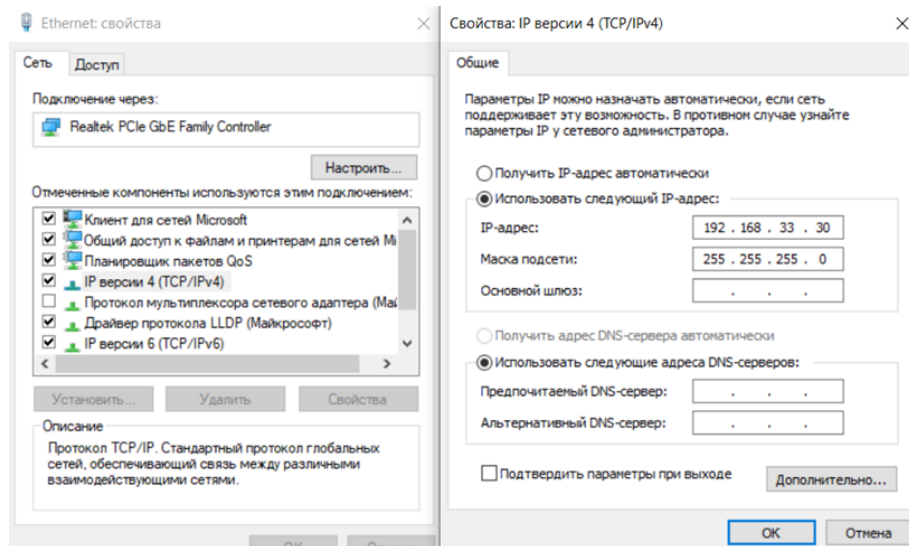


Рис. 3.6. Содержимое окна настройки соединения

После настройки соединения устройство готово к работе.

### 3.3 Проверка работоспособности

Для проверки работоспособности установки устройство было проверено с помощью mmWave Studio. После запуска программы и выполнения скрипта startup.lua появляется окно RadarAPI, в котором производится настройка радара.

После выбора нужных параметров и загрузки на плату необходимого аппаратного ПО статус отображения соединения меняется на Connected, а напротив радарной и управляющей подсистем отображается версия прошивки. Окно mmWave Studio после настройки продемонстрировано на рис. 3.7.

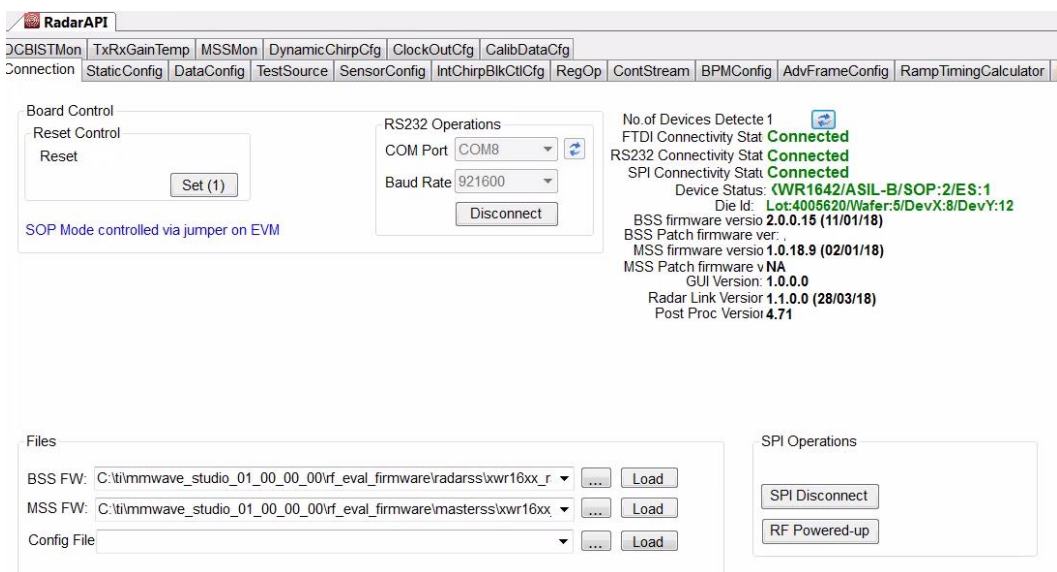


Рис. 3.7. Содержимое окна mmWave Studio после подключения



Для общей оценки работоспособности системы изначальные параметры сигнала не менялись. После проведения эксперимента, постобработка была проведена встроенными средствами mmWave Studio. Результаты постобработки продемонстрированы на рис. 3.8.

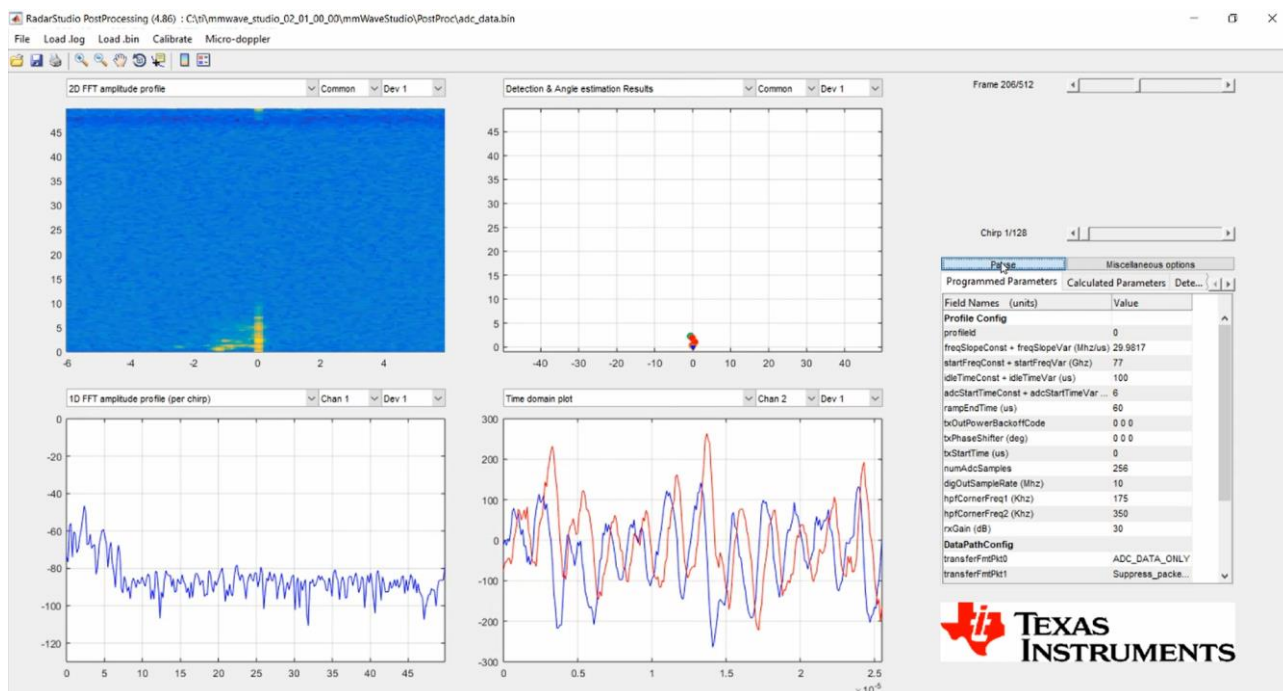


Рис. 3.8. Результат постобработки

В данном эксперименте небольшой объект перемещался на небольшом расстоянии от радара. Как видно по графику амплитудного спектра (левый нижний), объект был распознан радаром, хотя и довольно неточно. Это объясняется тем, что эксперимент проводился в помещении, а следовательно, имели место множественные отражения сигнала, а также присутствием статичных объектов в помещении.

В целом, несмотря на не совсем точный результат, эксперимент признан успешным, а стенд рабочим.

### 3.4 Алгоритм на практике

После успешной проверки работоспособности экспериментального стенда были рассчитаны необходимые параметры сигнала, используя в том числе инструмент RampTimingCalculator, включенный в mmWave Studio, и показанный на рис. 3.9.

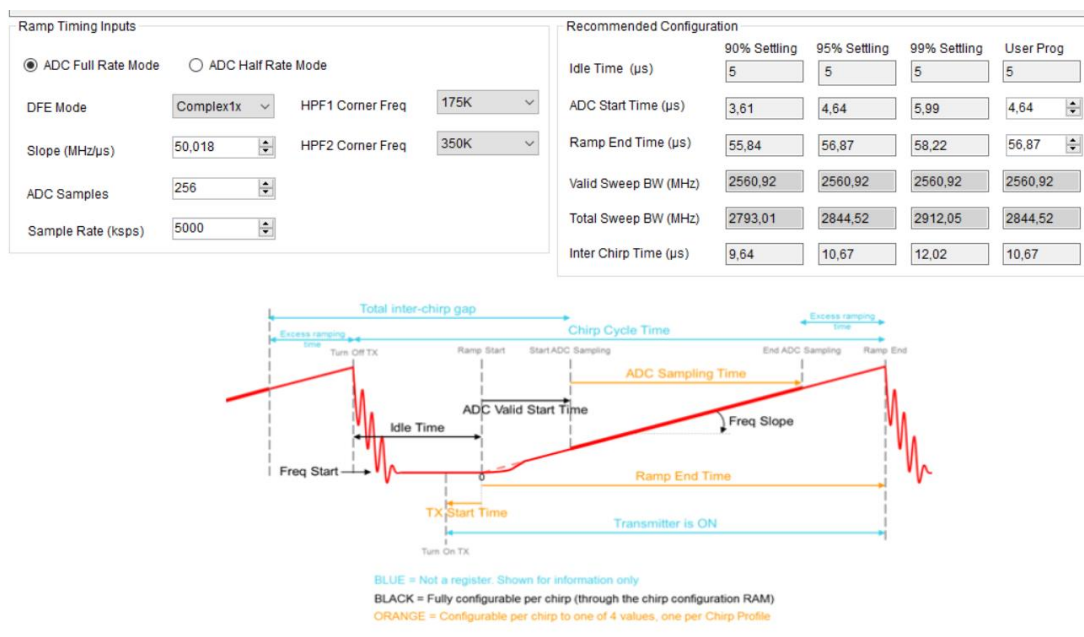


Рис. 3.9. Калькулятор конфигурации сигнала

С помощью данного инструмента можно упростить часть расчетов, так как рекомендованные параметры автоматически будут показаны в правой части окна. Также в нижней части расположена удобная памятка с указанием как тот или иной параметр влияет на конечную форму чирпов.

Так как возможности платы не безграничны, то при настройке через интерфейс mmWave Studio на некоторые настраиваемые значения накладываются некоторые искусственные программные ограничения. Помимо этого, при настройке и постобработке через встроенное средство студии нельзя выбрать количество точек для БПФ, которое также является важным параметром, который влияет на результативность измерений.

В конечном итоге, учитывая вышеописанные ограничения, а также значения табл. 1 и рис. 1.10 из раздела 1.2, были получены три профиля чирпов, конфигурации, представленной на рис. 3.10.

Profile ID	Start Freq(GHz)	Frequency Slope(MHz/us)	Idle Time(us)	TX Start Time(us)	ADC Start Time(us)	ADC Samples	Sample Rate(kpsps)	Ramp End Time(us)	RX Gain(dB)	RX Gain Target(dB)
0	76,5	0	100	0	6	4096	10000	5000	30	30
1	76,5	0,0840000...	100	0	6	4096	10000	5000	30	30
2	76,5	0,0970000...	100	0	6	4096	10000	5000	30	30

Рис. 3.10. Содержимое окна менеджера профилей чирпов

Не совсем удобно каждый раз пользоваться mmWave Studio для прохождения процедуры подключения радара, загрузки аппаратного ПО,

настройки АЦП и формата данных, параметров чирпов и фреймов. Функционал программы весьма большой и даже избыточный для целей данной работы, к тому же ручной ввод параметров и запуск установки можно автоматизировать.

### 3.5 Получение и анализ сырых данных

Так как встроенное средство постобработки не позволяет настроить сам процесс обработки, это накладывает определенные ограничения. Значит, нужна возможность получить сырые данные с АЦП, переданные платой на ПК, но еще не обработанные встроенными средствами mmWave Studio.

Данные записываются в бинарные файлы, которые вмещают до 1 Гб данных, названия файлов `adc_data_Raw_0.bin` для первого файла, и `adc_data_Raw_1.bin` для второго, и так далее. В бинарных файлах сохраняются пакеты, полученные через ethernet. Каждый пакет содержит [10]:

1. 4 байта, указывающие на номер UDP пакета.
2. 4 байта поля длины данных, указывает на количество байт в пакете.
3. 6 байт счетчика, указывающего, сколько всего байт было передано вплоть до данного пакета.
4. Поле данных, размером от 48, до 1462 байт.

Данные хранятся в формате, приведенном на рис. 3.11.

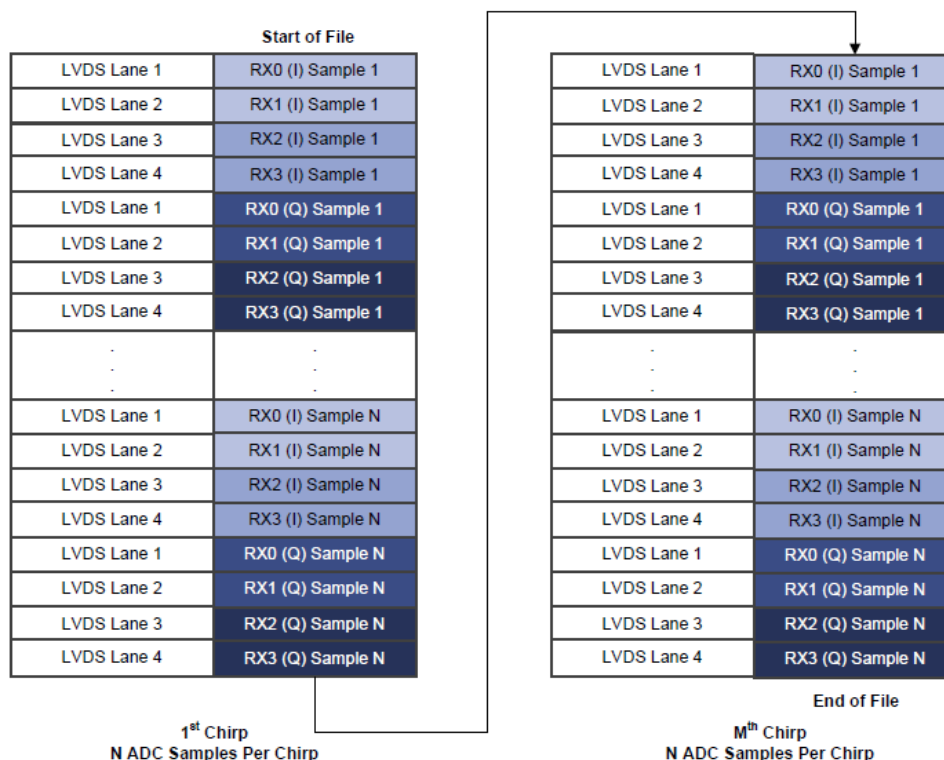


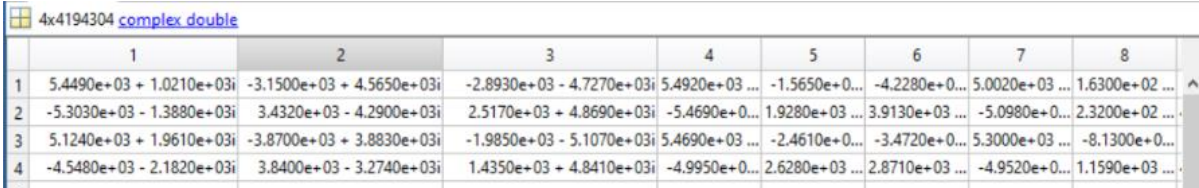
Рис. 3.11. Формат комплексных данных, полученных через Ethernet

Для упрощения последующей обработки, а так же для лучшего понимания результата был написан скрипт на языке Matlab [11], принимающий файл сырых данных, и создающий на его основе матрицу размером  $4 \times N$ , где 4 – число каналов приемника, а  $N$  – общее число семплов. Таким образом значения проще интерпретировать, так как данные представлены более наглядно. Если какие-то из приемников неактивны, то соответствующие им строки будут заполнены нулями. Текст скрипта приведен в листинге 3.1.

Листинг 3.1. Скрипт для создания матрицы значений

```
function [retVal] = readDCA1000(fileName)
%% global variables
% change based on sensor config
numADCBits = 16; % number of ADC bits per sample
numLanes = 4; % do not change. number of lanes is always 4 even if only 1 lane is
used. unused
isReal = 0; % set to 1 if real only data, 0 if complex data are populated with 0 %
read file and convert to signed number
% read .bin file
fid = fopen(fileName,'r');
% DCA1000 should read in two's complement data
adcData = fread(fid, 'int16');
% if 12 or 14 bits ADC per sample compensate for sign extension
if numADCBits ~= 16
    l_max = 2^(numADCBits-1)-1;
    adcData(adcData > l_max) = adcData(adcData > l_max) - 2^numADCBits;
end
fclose(fid);
%% organize data by LVDS lane
% for real only data
if isReal
    % reshape data based on one samples per LVDS lane
    adcData = reshape(adcData, numLanes, []);
    %for complex data
else
    % reshape and combine real and imaginary parts of complex number
    adcData = reshape(adcData, numLanes*2, []);
    adcData = adcData([1,2,3,4], :) + sqrt(-1)*adcData([5,6,7,8], :);
end
%% return receiver data
retVal = adcData;
```

В результате обработки данным скриптом файла данных, полученного в ходе тестирования установки, была получена матрица, приведенная на рис. 3.12.



	1	2	3	4	5	6	7	8
1	5.4490e+03 + 1.0210e+03i	-3.1500e+03 + 4.5650e+03i	-2.8930e+03 - 4.7270e+03i	5.4920e+03 ...	-1.5650e+0...	-4.2280e+0...	5.0020e+03 ...	1.6300e+02 ...
2	-5.3030e+03 - 1.3880e+03i	3.4320e+03 - 4.2900e+03i	2.5170e+03 + 4.8690e+03i	-5.4690e+0...	1.9280e+03 ...	3.9130e+03 ...	-5.0980e+0...	2.3200e+02 ...
3	5.1240e+03 + 1.9610e+03i	-3.8700e+03 + 3.8830e+03i	-1.9850e+03 - 5.1070e+03i	5.4690e+03 ...	-2.4610e+0...	-3.4720e+0...	5.3000e+03 ...	-8.1300e+0...
4	-4.5480e+03 - 2.1820e+03i	3.8400e+03 - 3.2740e+03i	1.4350e+03 + 4.8410e+03i	-4.9950e+0...	2.6280e+03 ...	2.8710e+03 ...	-4.9520e+0...	1.1590e+03 ...

Рис. 3.12. Полученная матрица значений семплов, распределенные по каналам приемника.



## 4. ЗАПУСК И НАСТРОЙКА СЕНСОРА ИСПОЛЬЗУЯ API

Так как компания Texas Instruments предлагает готовую библиотеку для работы с API радара, а также предоставляет необходимые интерфейсы, остается только в правильном порядке вызвать соответствующие методы, и составить файл конфигурации, для которого так же есть готовый парсер [12].

### 4.1 Описание файла конфигурации

Файл конфигурации – текстовый документ формата .txt, в котором описаны необходимые параметры системы. Пример описания профиля чирпа приведен в листинге 4.1

Листинг 4.1. Настройка сигнала с помощью файла конфигурации

```
profileId=0;
startFreqConst=1439117143;
idleTimeConst=1000;
adcStartTimeConst=600;
rampEndTime=6000;
txOutPowerBackoffCode=0;
txPhaseShifter=0;
freqSlopeConst=621;
txStartTime=0;
numAdcSamples=256;
digOutSampleRate=10000;
hpfCornerFreq1=0;
hpfCornerFreq2=0;
txCalibEnCfg=511;
rxGain=30;
```

В файле описываются все параметры, которые нужно настроить, от количества активных приемников и передатчиков, до продвинутых параметров фрейма. Пример полного файла конфигурации приведен в приложении 1.

### 4.2 Программа загрузки параметров

Программа парсер и методы, которые вызывают парсер для получения конкретных значений из файла конфигурации, методы обработки асинхронных сигналов и методы непосредственно взаимодействия API уже реализованы и описаны в документации, посвященной программированию параметров. Для создания программы, загружающей нужные параметры, нужно лишь вызвать требуемые методы в правильном порядке [13]. Также необходимо подключить все готовые библиотеки:

1. mmw\_config.h. Данная библиотека содержит методы для получения значений требуемых параметров из текстового файла конфигурации.

2. mmwavelink.h. Основная библиотека, позволяющая программе взаимодействовать с радаром.
3. rls\_osi.h. Вспомогательная библиотека, позволяющая основной взаимодействовать с ОС Windows.
4. rls\_studio.h. Библиотека, содержащая интерфейсы соединения с устройством.

Разработанная часть ПО состоит из двух функций:

1. int Execute(). Поочередно выполняет все необходимые операции.
2. void main(void). Вызывает функцию выше и выводит сообщение об успешности выполнения программы.

Описание функции main приведено в листинге 4.2.

Листинг 4.2. Разработанная функция main

```
void main(void)
{
    int retVal;

    retVal = Execute();
    if(retVal == RL_RET_CODE_OK)
    {
        printf("Success!\n");
    }
    else
    {
        printf("Fail...\n");
    }
    printf("Press any key to exit\n");
    getchar();
    printf("Exit\n");
}
```

Заводится переменная, которая приравнивается результату выполнения основной функции, после чего сравнивается с кодом ответа об успешной операции. Если значения совпадают, то выводится сообщение об успешном выполнении программы, иначе – об ошибке. После чего программа ждет ввода любого символа, а получив его – закрывает окно консоли.

Начало функции Execute приведено в листинге 4.3.

## Листинг 4.3. Начало основной функции

```
int Execute()
{
    int retVal = RL_RET_CODE_OK;
    unsigned char deviceMap = RL_DEVICE_MAP_CASCADED_1;

    retVal = MMWL_openConfigFile();
    if (retVal != RL_RET_CODE_OK)
    {
        printf("failed to Open configuration file Error code: %d\n", retVal);
        return -1;
    }
}
```

В начале, как и в функции `main`, создается переменная, которая будет приравниваться к результату выполнения различных функций. Помимо этого, создается экземпляр устройства, который будет передаваться последующим функциям в качестве аргумента. Первой вызывается функция открытия и парсинга файла конфигурации для получения всех необходимых значений. Следующая функция относится непосредственно к работе с радаром и приведена в листинге 4.4.

## Листинг 4.4. Вызов функции запуска управляющей подсистемы

```
retVal = MMWL_powerOnMaster(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("mmWave Device Power on failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("mmWave Device Power on success\n\n");
}
```

Теперь переменная результата приравнивается результату выполнения функции запуска управляющей подсистемы. Следующим по порядку должен быть осуществлен вызов функции загрузки аппаратного ПО, и проверка его версии, приведенный в листинге 4.5.

### Листинг 4.5. Вызов функции загрузки аппаратного ПО

```
printf("---Firmware Download---\n");
retVal = MMWL_firmwareDownload(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Firmware update failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("---Firmware update successful---\n");
}
```

Последующие вызовы функций отличаются только самой вызываемой функцией, и выполнены в порядке, описанном в руководстве, предоставленном производителем. Полное описание порядка вызова функций приведены в приложении 2, а полный код основной функции в приложении 3. Результаты работы полученной программы приведены в приложении 4.

## 5. ТЕСТИРОВАНИЕ УСТАНОВКИ

Для оценки точности полученных результатов необходимо знать измеряемые величины, чего крайне трудно добиться в «комнатных» условиях, в силу отражений и того, что скорость реальных движущихся объектов зачастую заранее неизвестна. Для решения этой проблемы была использована возможность создания виртуальных целей с заданными параметрами.

Для создания виртуальных целей воспользуемся интерфейсом mmWave Studio, продемонстрированным на рис. 5.1

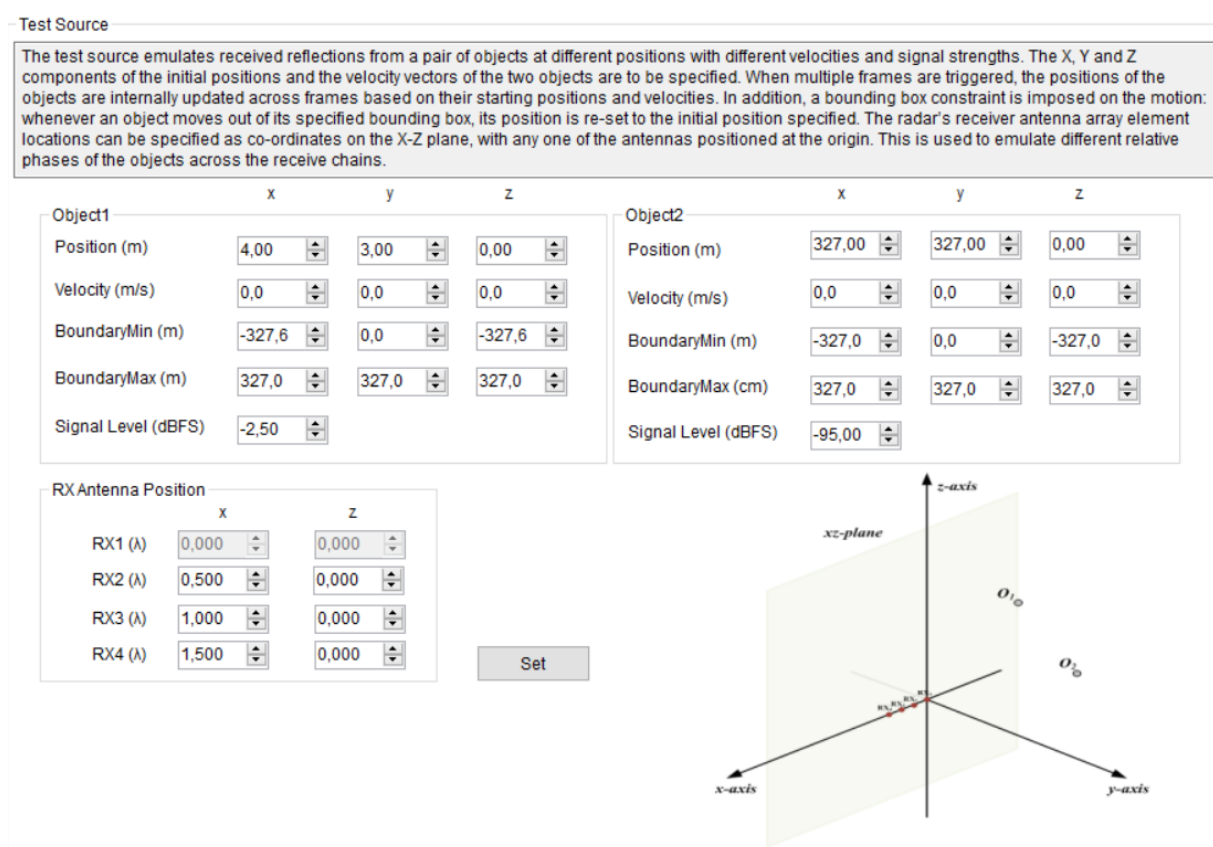


Рис. 5.1. Интерфейс создания искусственных целей в mmWave Studio

Для настройки установки была использована разработанная программа. Для тестирования были созданы две виртуальные цели по разные стороны от радара, движущиеся в разные стороны, со скоростями близкими друг к другу.

В результате несмотря на крайне малую разницу в скорости и равном расстоянии, объекты все равно распознаются отдельно, что заметно на рис. 5.2.

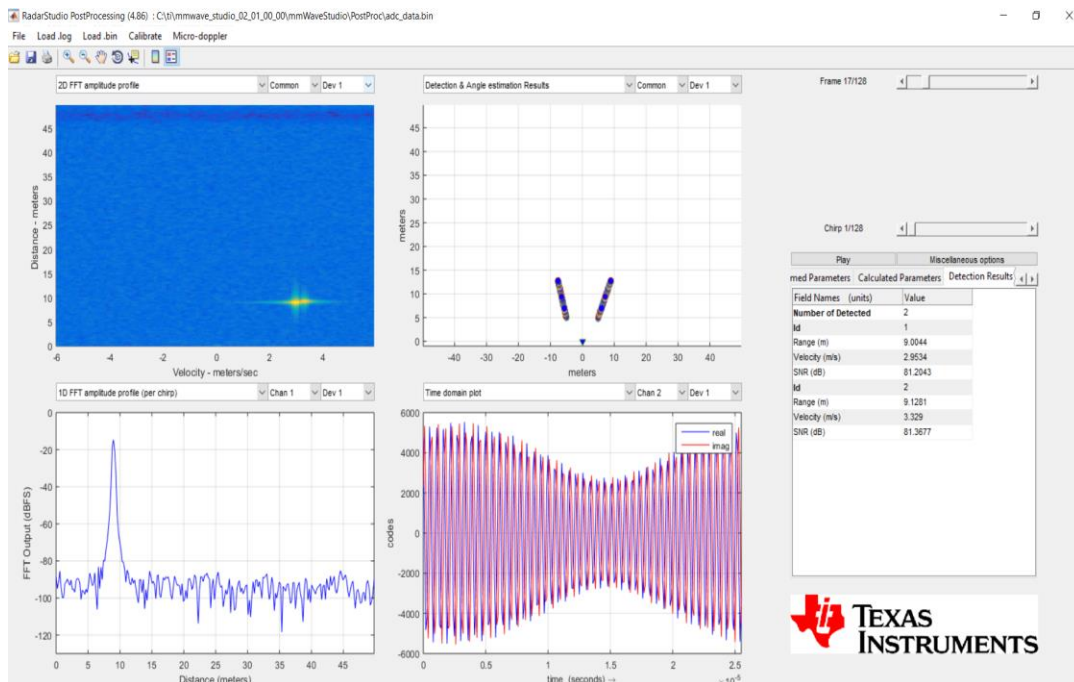


Рис. 5.2. Результат обработки сигнала, при отслеживании двух целей

Как видно из правой части окна, разница скоростей объектов составляет около 0.4 м/с, что совпадает с разрешением по скорости, заявленным в статье, посвященной алгоритму.

Для сравнения повторим опыт со стандартными параметрами, при тех же характеристиках и положении объектов. Результат на рис. 5.3.

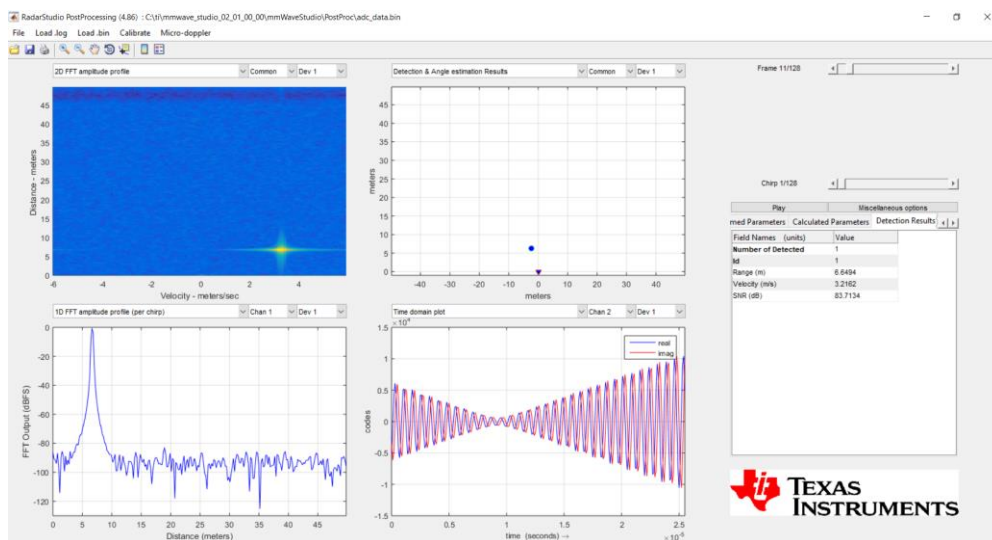


Рис. 5.3. Отслеживание двух целей, используя стандартные параметры

Как видно из результатов постобработки, в этот раз виден только один объект.

## ЗАКЛЮЧЕНИЕ

В рамках данной работы были решены следующие задачи:

1. Детально изучены принципы, лежащие в основе работы FMCW радара, методы, с помощью которых вычисляются те или иные параметры, выведены ключевые соотношения параметров сигнала и работоспособности радара. Определены требования к значениям различных параметров.
2. Проанализированы три кандидата на роль аппаратной платформы стенда. В итоге было выбрано решение Texas Instruments, так как оно обладает наиболее подходящими рабочими частотами, а также упрощает разработку ПО для работы с сенсором, благодаря имеющимся готовым библиотекам.
3. Была собрана экспериментальная радарная установка, произведена необходимая конфигурация ПК, настройка самих плат, протестирована работоспособность стенда.
4. На основе имеющихся библиотек для работы с API радара, было разработано ПО, автоматизирующее настройку радара и запись данных. Необходимые значения параметров записываются в специальный текстовый файл конфигурации, который затем обрабатывается программой. Благодаря этому отпадает необходимость в использовании громоздкой mmWave Studio при проведении лабораторных испытаний.
5. При помощи инструмента создания виртуальных целей с заранее известными параметрами, была протестирована точность измерений полученной установки.

В результате решения указанных задач, был получен экспериментальный стенд, соответствующий требованиям алгоритма, и упрощающий проведение экспериментов. А значит, поставленные цели работы были успешно достигнуты.

Так же в процессе работы были изучены общие принципы работы радаров, получены практические навыки работы с системами радиолокации, что упрощает возможные дальнейшие исследования на эту тему.

В перспективе после всех испытаний возможно создание полноценного устройства на единой плате, с учетом всех ограничений, которые накладывает текущая реализация, а также разработка собственного средства постобработки и визуализации данных.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бакулев П.А. Радиолокационные системы. Учебник для вузов. – М.: Радиотехника, 2004, 320 с. – с. 4 – 14.
2. Радиолокационные сигналы для аэрокосмических, оборонных и автомобильных РЛС – Режим доступа: <https://www.astrosoft.ru/articles/radar/radiolokatsionnye-signal-y-dlya-aerokosmicheskikh-oboronnykh-i-avtomobilnykh-rls/> (дата обращения: 18.04.2020)
3. S.I. Ivanov, V.D. Kuptsov, A.A. Fedotov. Features of Multi-target Detection Algorithm for Automotive FMCW Radar
4. Радиолокатор непрерывного излучения с частотной модуляцией – Режим доступа: <https://www.radartutorial.eu/02.basics/rp08.ru.html> (дата обращения: 23.05.2020).
5. 24 GHz radar tools and development environment user manual – Режим доступа: [https://www.infineon.com/dgdl/Infineon-24GHz\\_Radar\\_Tools\\_and\\_Development\\_Environment\\_User\\_Manual-ApplicationNotes-v01\\_00-EN.pdf](https://www.infineon.com/dgdl/Infineon-24GHz_Radar_Tools_and_Development_Environment_User_Manual-ApplicationNotes-v01_00-EN.pdf) (дата обращения: 15.03.2020).
6. User Guide SiRad Simple Evaluation Kit – Режим доступа: [https://siliconradar.com/wp-content/uploads/2019/01/20181220\\_UserGuide\\_SiRad\\_Simple.pdf](https://siliconradar.com/wp-content/uploads/2019/01/20181220_UserGuide_SiRad_Simple.pdf) (дата обращения 29.03.2020).
7. AWR1243 Single-Chip 77- and 79-GHz FMCW Transceiver – Режим доступа: <https://www.ti.com/lit/ds/symlink/awr1243.pdf> (дата обращения: 19.04.2020).
8. DCA1000EVM Data Capture Card – Режим доступа: <https://www.ti.com/lit/ug/spruij4a/spruij4a.pdf> (дата обращения: 11.06.2020).
9. mmWave Sensor Raw Data Capture Using the DCA1000 Board and mmWave Studio – Режим доступа: [https://training.ti.com/sites/default/files/docs/mmwave\\_sensor\\_raw\\_data\\_capture\\_using\\_dca1000\\_v02.pdf](https://training.ti.com/sites/default/files/docs/mmwave_sensor_raw_data_capture_using_dca1000_v02.pdf) (дата обращения 17.04.2020).
10. Mmwave Radar Device ADC Raw Data Capture – Режим доступа: <https://www.ti.com/lit/an/swra581b/swra581b.pdf> (дата обращения: 30.05.2020)
11. Векторы и матрицы в MatLab – Режим доступа: [http://scask.ru/a\\_lect\\_matlab.php](http://scask.ru/a_lect_matlab.php) (дата обращения: 08.05.2020)



12. Programming Chirp Parameters in TI Radar Devices – Режим доступа: <https://www.ti.com/lit/an/swra553a/swra553a.pdf> (дата обращения: 02.06.2020)
13. MMWAVE DFP User Guide – Режим доступа: [http://software-dl.ti.com/r-processors/esd/MMWAVE-DFP/01\\_00\\_00\\_01/exports/mmwave\\_dfp\\_user\\_guide.pdf](http://software-dl.ti.com/r-processors/esd/MMWAVE-DFP/01_00_00_01/exports/mmwave_dfp_user_guide.pdf) (дата обращения: 08.05.2020)

## ПРИЛОЖЕНИЕ 1. ПРИМЕР ФАЙЛА КОНФИГУРАЦИИ

### Листинг 1. Файл конфигурации параметров радара

```
#cascade mode enable
#
cascade_enable=0;
#END

#
#power on master arguments, please modify if needed.
#rlClientCbs_t: crcType 0:16Bit/1:32Bit/2:64Bit, ackTimeout
#
crcType=1;
ackTimeout=1000;
#END

#
#channel config parameters, please modify if needed.
#rlChanCfg_t
#
channelTx=3;
channelRx=4;
cascading=0;
#END

#
#ADC out config parameters, please modify if needed.
#rlAdcOutCfg_t
#
adcBits=2;
adcFormat=2;
#END

#
#DATA format config parameters, please modify if needed.
#rlDevDataFmtCfg_t
#
rxChanEn=15;
adcBitsD=2;
adcFmt=1;
iqSwapSel=0;
chInterleave=0;
#END

#
#Low power config Parameters, please modify if needed.
#rlLowPowerModeCfg_t
#
anaCfg=0;
lpAdcMode=0;
#END
```

```

#
#Data Path config parameters, please modify if needed
#rlDevDataPathCfg_t
#
intfSel=1;
transferFmtPkt0=1;
transferFmtPkt1=0;
cqConfig=2;
cq0TransSize=64;
cq1TransSize=64;
cq2TransSize=64;
#END

#
#LVDS clock config parameters, please modify if needed
#rlDevDataPathClkCfg_t
#
laneClk=1;
dataRate=1;
#END

#
#SET HSI clock parameters, please modify if needed.
#rlDevHsiClk_t
#
hsiClk=9
#END

#
#LANE config parameters, please modify if needed.
#rlDevLaneEnable_t
#
laneEn=15;
#END

#
#LVDS Lane Config parameters, please modify if needed.
#rlDevLvdsLaneCfg_t
#
laneFmtMap=0;
laneParamCfg=1;
#END

#
#Profile config parameters, please modify if needed.
#rlProfileCfg_t
#
profileId=0;
startFreqConst=1439117143;
idleTimeConst=1000;
adcStartTimeConst=600;
rampEndTime=6000;
txOutPowerBackoffCode=0;
txPhaseShifter=0;
freqSlopeConst=621;
txStartTime=0;
numAdcSamples=256;
digOutSampleRate=10000;
hpfCornerFreq1=0;
hpfCornerFreq2=0;
txCalibEnCfg=511;
rxGain=30;
#END

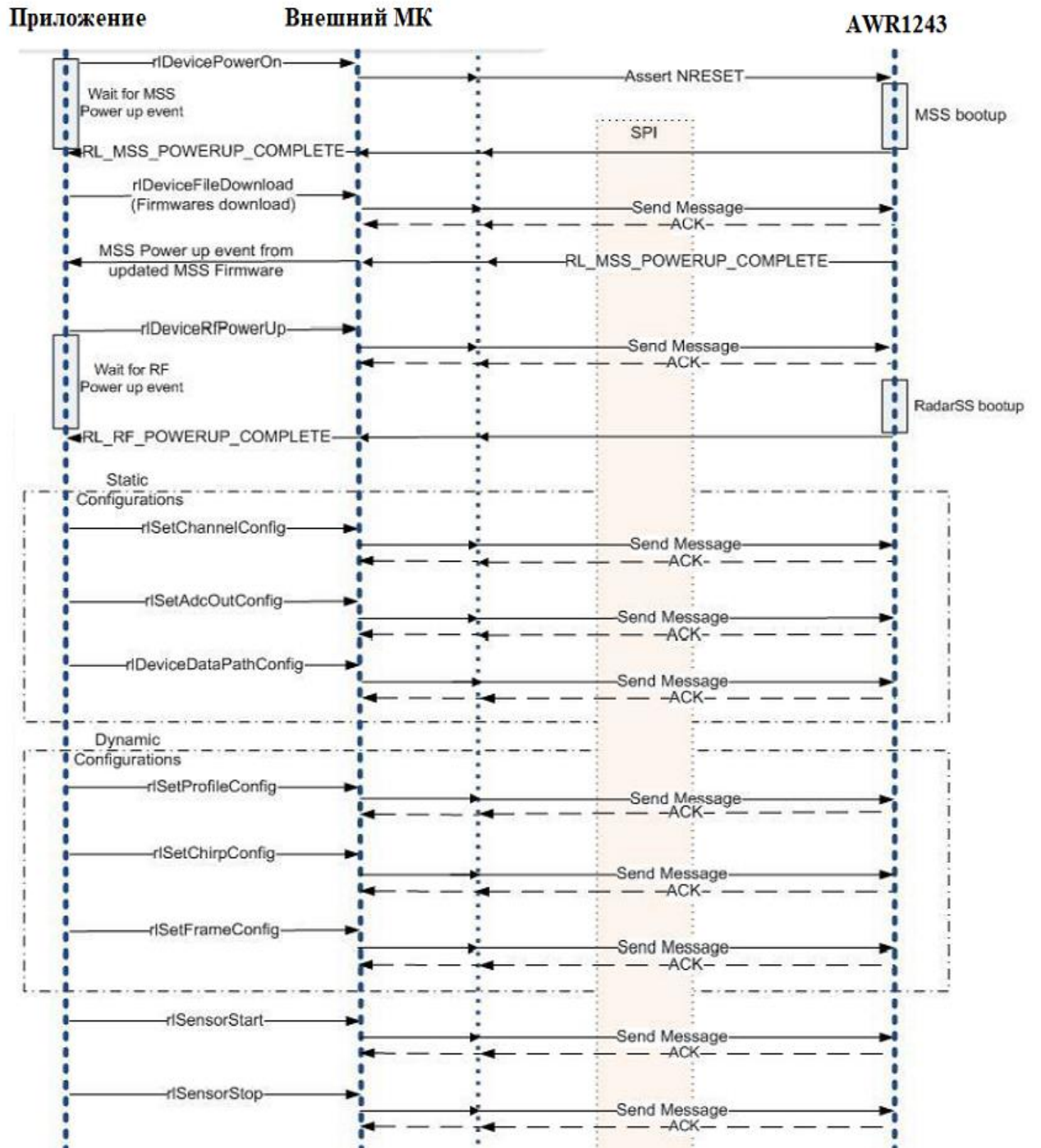
#
#Chirp Configuration parameters, please modify if needed.
#rlChirpCfg_t
#

```

```
chirpStartIdx=0;
chirpEndIdx=127;
profileIdCPCFG=0;
startFreqVar=0;
freqSlopeVar=0;
idleTimeVar=0;
adcStartTimeVar=0;
txEnable=1;
#END

#
#Frame configuration parameters, please modify if needed.
#rlFrameCfg_t
#
chirpStartIdxFCF=0;
chirpEndIdxFCF=127;
frameCount=64;
loopCount=1;
periodicity=20000000;
triggerDelay=0;
numAdcSamples=512;
triggerSelect=1;
#END
```

## ПРИЛОЖЕНИЕ 2. ПОРЯДОК ВЫЗОВА ФУНКЦИЙ



**ПРИЛОЖЕНИЕ 3. ЛИСТИНГ КОДА ОСНОВНОЙ ФУНКЦИИ**

```
int Execute()
{
    int retVal = RL_RET_CODE_OK;
    unsigned char deviceMap = RL_DEVICE_MAP_CASCADED_1;

    retVal = MMWL_openConfigFile();
    if (retVal != RL_RET_CODE_OK)
    {
        printf("failed to Open configuration file Error code: %d\n", retVal);
        return -1;
    }

    retVal = MMWL_powerOnMaster(deviceMap);
    if (retVal != RL_RET_CODE_OK)
    {
        printf("mmWave Device Power on failed Error code: %d\n", retVal);
        return -1;
    }
    else
    {
        printf("mmWave Device Power on success\n\n");
    }

    printf("---Firmware Download---\n");
    retVal = MMWL_firmwareDownload(deviceMap);
    if (retVal != RL_RET_CODE_OK)
    {
        printf("Firmware update failed Error code: %d\n", retVal);
        return -1;
    }
    else
    {
        printf("---Firmware update successful---\n");
    }

    retVal = MMWL_setDeviceCrcType(deviceMap);
    if (retVal != RL_RET_CODE_OK)
    {
        printf("CRC Type set for MasterSS failed Error code: %d\n", retVal);
        return -1;
    }
    else
    {
        printf("CRC Type set for MasterSS success\n\n");
    }

    retVal = MMWL_rfEnable(deviceMap);
    if (retVal != RL_RET_CODE_OK)
    {
        printf("Radar/RF subsystem Power up failed Error code: %d\n", retVal);
        return -1;
    }
    else
    {
        printf("Radar/RF subsystem Power up successful\n\n");
    }
}
```

```

printf("---Basic/Static Configuration---\n");
retVal = MMWL_basicConfiguration(deviceMap, 0);
if (retVal != RL_RET_CODE_OK)
{
    printf("Basic/Static configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("---Basic/Static configuration success---\n",
        deviceMap);
}

retVal = MMWL_rfInit(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("RF Initialization/Calibration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("RF Initialization/Calibration successful\n");
}

printf("---FMCW Configuration---\n");
retVal = MMWL_profileConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Profile Configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Profile Configuration success\n\n");
}

retVal = MMWL_chirpConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Chirp Configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Chirp Configuration success\n\n");
}

printf("---Data Path(LVDS/CSI2) Configuration---\n");
retVal = MMWL_dataPathConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Data Path Configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Data Path Configuration successful\n\n");
}

retVal = MMWL_hsiClockConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("CSI2/LVDS Clock Configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("CSI2/LVDS Clock Configuration success\n\n");
}

```

```

}

retVal = MMWL_hsiLaneConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("CSI2/LVDS Lane Config failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("CSI2/LVDS Lane Configuration success\n");
}

retVal = MMWL_frameConfig(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Frame Configuration failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Frame Configuration success for deviceMap %u \n", deviceMap);
}

retVal = MMWL_sensorStart(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Sensor Start failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Sensor Start successful\n\n");
}

osiSleep(frameCount*framePeriodicity);

retVal = MMWL_sensorStop(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    if (retVal == RL_RET_CODE_FRAME_ALREADY_ENDED)
    {
        printf("Frame is already stopped when sensorStop CMD was issued\n");
    }
    else
    {
        printf("Sensor Stop failed Error code: %d\n", retVal);
        return -1;
    }
}
else
{
    printf("Sensor Stop successful\n\n");
}

retVal = MMWL_powerOff(deviceMap);
if (retVal != RL_RET_CODE_OK)
{
    printf("Device power off failed Error code: %d\n", retVal);
    return -1;
}
else
{
    printf("Device power off success\n\n");
}
MMWL_closeConfigFile();
return 0;
}

```



## ПРИЛОЖЕНИЕ 4. РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

```

rComIfOpen Callback Called for Device Index [0]
Got 4 devices connected
Device 0: AR-DevPack-EVM-012 D , SN: FT3EOYP9D
Device 1: AR-DevPack-EVM-012 A , SN: FT3EOYP9A
Device 2: AR-DevPack-EVM-012 B , SN: FT3EOYP9B
Device 3: AR-DevPack-EVM-012 C , SN: FT3EOYP9C
SPI port opened (PortNum=1)
IRQ & I2C port opened (PortNum=2)
Board Control port opened (PortNum=3)
Generic GPIO port opened (PortNum=0)
rDeviceEnable Callback is called by mmWaveLink for Device Index [0]
mmWave Device Power on success

---Firmware Download---
Meta Image download started for deviceMap 1
Download in Progress: 0%..10%..20%..30%..40%..50%..60%..70%..80%..90%..Done!
Meta Image download complete ret = 0

Waiting for firmware update response from mmWave device
Firmware update successful
Debug: Finished rDeviceSetMiscConfig
CRC Type set for MasterSS success

RF Version [ 2. 0. 0. 1]
MSS version [ 1.10. 0.20]
mmWaveLink version [ 1. 2. 5.16]

RF Patch Version [ 1. 2. 5. 2]
MSS Patch version [ 1. 2. 5. 1]
Radar/RF subsystem Power up successful

---Basic/Static Configuration---
Calling rSetChannelConfig With [4]Rx and [7]Tx Channel Enabled
Channel Configuration success for deviceMap 1

Calling rSetAdcOutConfig With [2]ADC Bits and [2]ADC Format
AdcOut Configuration success for deviceMap 1

Calling rRfSetLdoBypassConfig With Bypass [0]
LDO Bypass Configuration success for deviceMap 1

Data format Configuration success for deviceMap 1
Low Power Configuration success for deviceMap 1
Debug: Finished rSetAsyncEventDir
AsyncEvent Configuration success for deviceMap 1

Basic/Static configuration success for deviceMap 1
Async event: RF-init calibration status
RF Initialization/Calibration successful
---FMCW Configuration---
Calling rSetProfileConfig with
ProfileId[0]
Start Frequency[77.200256] GHz
Ramp Slope[29.981732] MHz/uS
Profile Configuration success

Calling rSetChirpConfig with
ProfileId[0]
Start Idx[0]

```

```
End Idx[127]
Debug: Get chirp configurations are matching with parameters configured during
rlSetChirpConfig
Chirp Configuration success

---Data Path(LVDS/CSI2) Configuration---
Calling rlDeviceSetDataPathConfig with HSI Interface[1] Selected
Data Path Configuration successful

Calling rlDeviceSetDataPathClkConfig with HSI Data Rate[1] Selected
MMWL_hsiDataRateConfig success for deviceMap 1

Calling rlDeviceSetHsiClk with HSI Clock[9]
MMWL_setHsiClock success for deviceMap 1

CSI2/LVDS Clock Configuration success

LaneConfig success for deviceMap 1
LvdsLaneConfig success for deviceMap 1
CSI2/LVDS Lane Configuration success
Calling rlSetFrameConfig with
Start Idx[0]
End Idx[127]
Loops[1]
Periodicity[100]ms
Frame Configuration success for deviceMap 1
Async event: Frame trigger
Sensor Start successful

Async event: Frame stopped
Frame is already stopped when sensorStop CMD was issued
rlDeviceDisable Callback is called by mmWaveLink for Device Index [0]
Device power off success

Success!
Press any key to exit
```