Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра компьютерных систем в управлении и проектировании (КСУП)

Домаскин Дмитрий Александрович

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СТАЦИОНАРНОГО КОМПЛЕКСА КОНТРОЛЯ СКВАЖИН ОБОРУДОВАННЫХ ШГНУ

Направление 09.04.01 «Информатика и вычислительная техника»
Магистерская программа 09.04.01 «Информационное обеспечение аппаратно-программных комплексов»

Диссертация
на соискание степени
магистра

Научный руководитель:
Доцент каф. КСУП ТУСУР,
Канд. техн. наук
_____ А.А. Калентьев

Консультант от профильной
организации
Заместитель технического
директора ООО ТНПВО «СИАМ»
_____ В.Н. Фомин

Томск 2020

РЕФЕРАТ

Выпускная квалификационная работа 80 с., 29 рис., 28 табл., 20 источников.

МОБИЛЬНОЕ ПРИЛОЖНИЕ, КРОССПЛАТФОРМЕННАЯ РАЗРАБОТКА, ШГНУ, ДАТЧИК, XAMARIN.

Объектом исследования является стационарный комплекс контроля скважин оборудованных ШГНУ.

Цель работы – разработка программного обеспечения стационарного комплекса контроля скважин оборудованных ШГНУ.

В процессе работы проводилось исследование стационарного комплекса контроля скважин оборудованных ШГНУ. Было проведено изучение датчиков входящих в состав комплекса, технологий обмена данными с датчиками. Был выполнен выбор технологии для разработки ПО. Инструментом для разработки был выбран язык программирования C# и фреймворк Xamarin. В качестве компонента для внедрения зависимостей использовался IoC-контейнер Autofac. В качестве хранилища данных использовалась база данных SQLite. Были разработаны программные модули сканирования эфира Bluetooth, взаимодействия с устройствами по Bluetooth, взаимодействия с датчиками, внедрения зависимостей, работы с данными, а также графического интерфейса.

Результатом работы является программное обеспечение верхнего уровня, позволяет в реальном времени отслеживать состояние датчиков стационарного комплекса контроля скважин. Осуществлять управление датчиками, путем запуска исследований. Загружать из датчиков данные результатов исследований для предоставления их для дальнейшего анализа.

ABSTRACT

Graduation work 80 pp., 29 fig., 28 tab., 20 sources.

MOBILE APPLICATIONS, CROSSPLATFORM DEVELOPMENT, SHNU, SENSOR, XAMARIN.

The object of the study is a stationary monitoring complex of wells equipped with a deep-well pumping unit.

The purpose of the work is to develop software for a stationary well monitoring complex equipped with a deep-well pumping unit.

In the process, a study was conducted of a stationary well monitoring complex equipped with a deep-well pumping unit. A study was conducted of the sensors included in the complex, technology for exchanging data with sensors. The choice of technology for software development was made. The development tool was the C # programming language and the Xamarin framework. An Autofac IoC container was used as a component for dependency injection. The SQLite database was used as a data warehouse. Software modules were developed for scanning Bluetooth broadcasts, interacting with devices via Bluetooth, interacting with sensors, injecting dependencies, working with data, and also a graphical interface.

The result of the work is top-level software, which allows real-time monitoring of the state of sensors of a stationary well monitoring complex. Manage sensors by launching research. Download data from the research results from the sensors to provide them for further analysis.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

УТВЕРЖДАЮ
Зав. каф. КСУП
д-р техн. наук, проф.,
_____Ю.А. Шурыгин
«____»_____2020 г.

ЗАДАНИЕ
на магистерскую диссертацию

студенту гр.588-М1 факультета вычислительных систем

Домаскину Дмитрию Александровичу

1. Тема работы: Программное обеспечение стационарного комплекса контроля скважин оборудованных ШГНУ.

(утверждена приказом по вузу от_____ № _____)
2. Срок сдачи студентом законченного проекта _____

3. Назначение и область применения системы: Управление и контроль с помощью управляющего блока (смартфона) стационарным комплексом контроля скважин оборудованных ШГНУ.

4. Требования к работе: В разработанном программном обеспечении должен быть реализован функционал, позволяющий сканировать эфир Bluetooth для поиска устройств; функционал, позволяющий подключаться к датчикам типов ДДИМ, ДДИН, ДУ по беспроводному каналу связи Bluetooth; функционал для отображения текущих данных датчиков; функционал для запуска на датчиках длительного исследования физических величин; функционал для загрузки из датчиков данных длительных исследований; функционал визуализации результатов длительных исследований; функционал для отправки результатов исследований по почте. Программное обеспечение должно предусматривать

изменения платформы управляющего блока.

5. Перечень вопросов, подлежащих разработке: Датчики компании “СИАМ”, протокол обмена данными с датчиками “СИАМ”, технология Bluetooth, выбор технологии для разработки ПО, разработка модуля сканирования Bluetooth эфира, разработка модуля взаимодействия с устройствами по Bluetooth, разработка модуля внедрения зависимостей, разработка модуля взаимодействия с датчиками, разработка модуля работы с данными, разработка графического интерфейса приложения.

6. Перечень графического материала (с точным указанием обязательных чертежей): Презентационный материал.

ЗАДАНИЕ СОГЛАСОВАНО:
Консультант по нормам и требованиям ЕСКД
Хабибулина Н.Ю. канд. техн. наук, доцент _____

Руководитель ВКР
Калентьев А.А. канд. техн. наук, доцент _____
Ф.И.О. должность, место работы
«_____»_____ 20__ г. Подпись_____

Консультант (с предприятия)
Фомин В.Н. зам. тех. директора ООО ТНПВО «СИАМ» г. Томск_____
Ф.И.О. должность, место работы
«_____»_____ 20__ г. Подпись_____

Задание принято к исполнению
«_____»_____ 20__ г. Студент_____
подпись

Оглавление

Диск CD-R	В конверте на обороте обложки
Пояснительная записка к ВКР	
Презентация дипломного проекта ВКР «Программное обеспечение стационарного комплекса контроля скважин оборудованных ШГНУ»	
Отчет о проверке на плагиат	

Графический материал:	15 слайд.
Презентация ВКР «Программное обеспечение стационарного комплекса контроля скважин оборудованных ШГНУ»	5 экз.
Твердая копия презентации	

1 Введение

В настоящее время актуальной является тема нефтедобычи. В связи с развитием технического прогресса человечество стремится осваивать все новые и новые источники энергоресурсов, а также совершенствовать методы добычи этих ресурсов. На фоне прочих источников энергоресурсов, таких как солнечная энергия, энергия солнца, атомная энергия, главную роль играют топливные ресурсы. Топливно-энергетический комплекс тесно связан с экономикой любой современной страны.

Среди трех отраслей топливной промышленности, включающей газовую, нефтяную и угольную, особняком стоит нефтяная промышленность. На протяжении многих лет спрос на нефть опережает предложение.

В связи с вышесказанным любая нефтяная компания стремится увеличить объемы добываемого сырья. Это можно сделать разными способами. Главные из них это георазведка и разработка новых нефтяных месторождений, а также совершенствования методов добычи.

Совершенствование оборудования, методов добычи сырья, оптимизация производственных процессов в нефтедобывающей промышленности, как и в любой другой сфере требует научного подхода.

В настоящее время компания ТНПВО “СИАМ” специализируется на решении подобных инженерных задач. Одним из проектов в разработке данной компании является “Стационарный комплекс контроля скважин, оборудованных ШГНУ”. Комплекс позволяет вести мониторинг состояния скважины в реальном времени, а также помогает определять дебит скважины. Дебит скважины является основной характеристикой, отражающей производительность скважины. Дебит скважины позволяет определить количество выкачиваемой жидкости из скважины в единицу времени. Низкое значение дебита указывает на низкую производительность скважины, причиной чего могут являться различные неполадки ШГНУ.

Данная работа посвящена разработке программного обеспечения для стационарного комплекса контроля скважин оборудованных ШГНУ.

2 Описание предметной области

Комплекс контроля скважин, оборудованных ШГНУ, предназначен для оперативного контроля уровня жидкости в скважине и динамограммы. Комплекс имеет раздельное исполнение измерительных блоков (датчиков) и управляющего блока (смартфона). Смартфон должен обеспечивать управление датчиками, визуализацию графиков, а также просмотр накопленных в памяти измерений. Связь между смартфоном и датчикам осуществляется по беспроводному соединению Bluetooth. По каналу Bluetooth осуществляется управление, конфигурация, а также сбор данных по средствам смартфона. Смартфон имеет встроенный GSM-модем для передачи результатов исследований по каналу сотовой связи.

Датчики представляют собой электронные приборы без индикаторов и клавиатур. Датчики полностью автономны и обеспечивают все возможности полнофункциональных приборов (кроме индикации и управления измерениями).

Исследования проводятся на добывающих скважинах, оборудованных погружными штанговыми насосами любого типа и любого конструктивного исполнения с приводом от станка-качалки серии СКН по ГОСТ 5866-56, СК по ГОСТ 5866-76, СКД по ОСТ 26-16-08-87 всех типоразмеров, и аналогичных зарубежного производства [1].

Для измерения физических величин в комплексе используются три типа датчиков.

1 Датчик уровня (ДУ-1) для определения уровня жидкости в скважине и контроля давления на устье.

2 Датчик динамометрирования межтравесный (ДДИМ-2) для измерения перемещения и величины нагрузки на полированный шток при работе ШГНУ.

3 Датчик динамометрирования накладной (ДДИН-2) для измерения перемещения и величины нагрузки на полированный шток при работе ШГНУ.

Для получения данных с датчиков в комплекс включен смартфон DEXP Ixion P350 Tundra под управлением операционной системы Android 7.0 с поддержкой BLE. На рисунке 2.1 приведена структурная схема стационарного комплекса контроля скважин оборудованных ШГНУ.

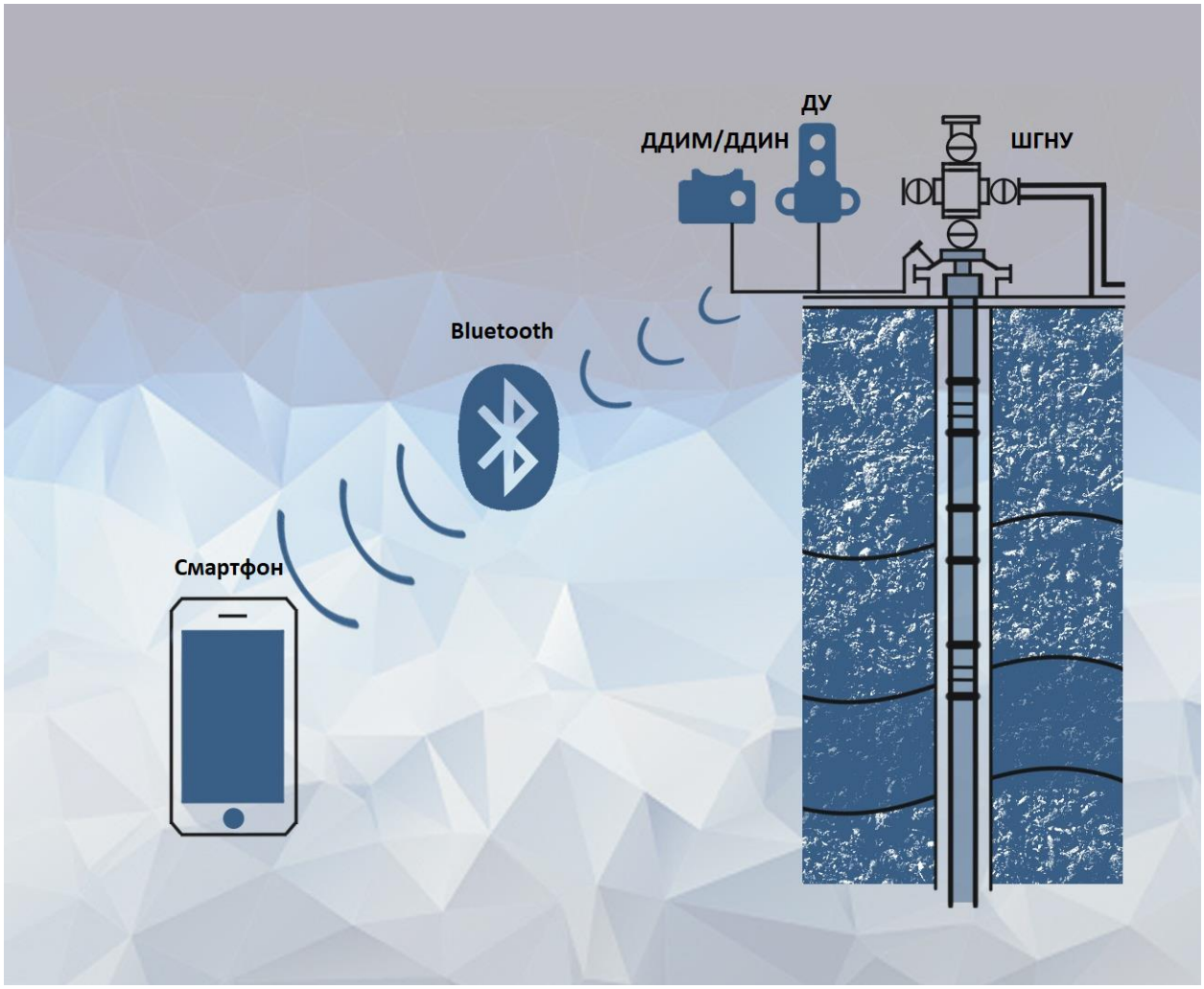

Рисунок 2.1 – Структурная схема стационарного комплекса контроля скважин оборудованных ШГНУ

2.1 ШГНУ

Использование шланговых глубинных насосных установок (ШГНУ) является одним из самых распространенных методов добычи нефти.

ШГНУ включает:

– наземное оборудование – станок-качалка (СК), оборудование устья, блок управления;

– подземное оборудование – насосно-компрессорные трубы (НКТ), штанги насосные (ШН), штанговый скважинный насос (ШСН) и различные защитные устройства, улучшающие работу установки в осложненных условиях.

На рисунке 2.2 представлен общая схема ШГНУ.

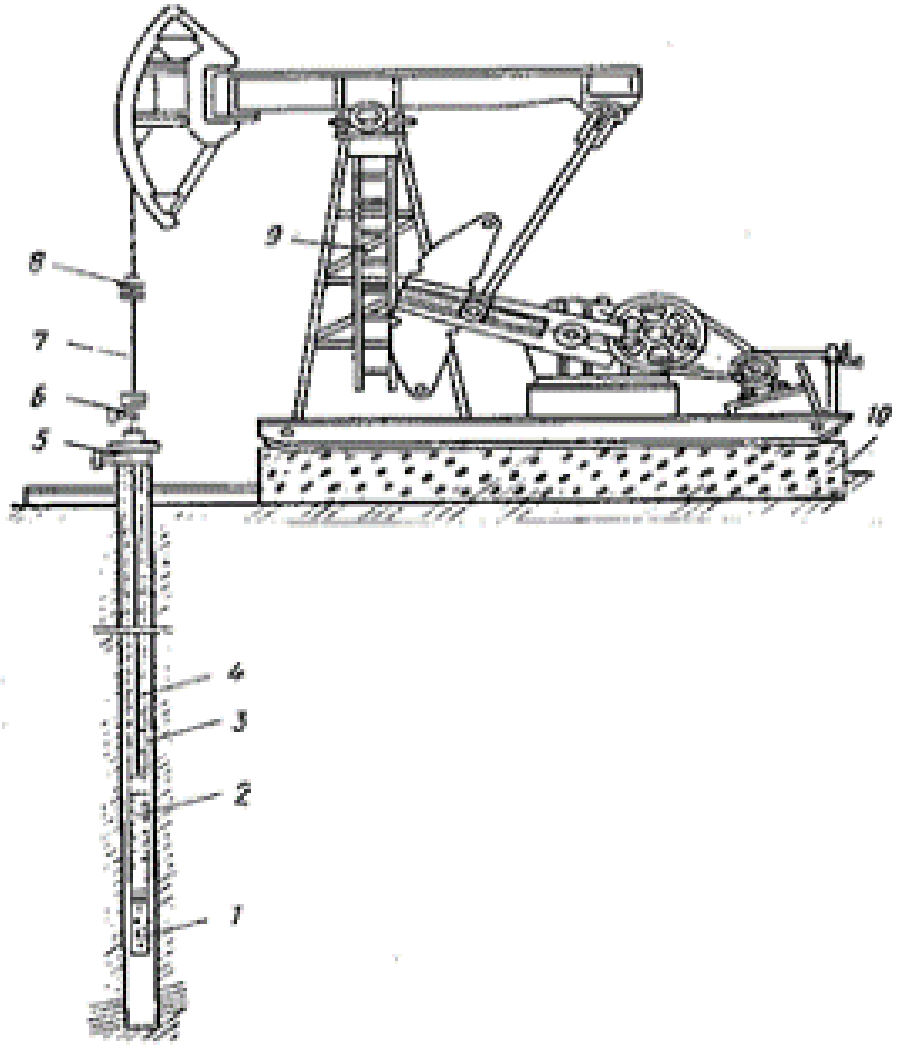

Рисунок 2.2 – Общий вид ШГНУ

Штанговая глубинная насосная установка (рисунок 1) состоит из скважинного насоса 2 вставного или невставного типов, насосных штанг 4, насосно-компрессорных труб 3, подвешенных на планшайбе или в трубной подвеске 8 устьевой арматуры, сальникового уплотнения 6, сальникового штока 7, станка качалки 9, фундамента 10 и тройника 5. На приеме скважинного насоса устанавливается защитное приспособление в виде газового или песочного фильтра 1.

2.2 Датчик уровня

Датчик уровня предназначен для генерации акустических импульсов в затрубном пространстве, приема, преобразования и анализа акустического отклика (эхо-сигнала), определения уровня жидкости в скважине и контроля давления на устье. Для записи эхограммы датчик устанавливается на измерительный патрубок устьевой арматуры скважины и не требует использования соединительных кабелей. Выпускной клапан может вращаться без ограничения вокруг продольной оси для установки выкидного сопла в сторону от оператора. На рисунке 2.3 Представлен общий вид датчика уровня (ДУ).

Рисунок 2.3 – Общий вид датчика уровня

Устройство предназначено для эксплуатации на устье скважины на месторождениях нефтяной и газовой отрасли промышленности и имеет функции, которые перечислены ниже.

1 Измерение уровня жидкости в затрубном пространстве скважины в пределах (20 – 6000) м, избыточного давления в пределах (0 – 100) кгс/см2.

2 Запись и сохранение измеряемых параметров в энергонезависимом запоминающем устройстве.

3 Передачу сохраненных параметров во внешнее устройство по беспроводному соединению, посредством специализированного ПО.

4 Подключение к мультирежимному терминалу по каналу Bluetooth.

5 Активацию и обновление ПО датчика по каналу Bluetooth

6 Устройство сохраняет работоспособность при температуре окружающей среды от минус 40 °С до + 50 °С.

7 Устройство функционирует автономно и питается от специального внутреннего аккумулятора с напряжением 3,6 В или 3,7 В. Минимальное рабочее напряжение аккумуляторной батареи, не приводящее к потере работоспособности устройства, составляет 2,7 В [2].

Основные технические характеристики указаны в таблице 2.1.

Таблица 2.1 – Основные характеристики датчика уровня

Наименование параметра	Норма по ТУ
Маркировка взрывозащиты	1Ex ib IIB T3 Gb X
Диапазон контроля уровня, м	20 – 6000
Память эхограмм, КБ	5
Разрешающая способность контроля уровня (при скорости звука в газе 340 м/с), % от ВПИ	0,03
Диапазон измерения давления, кгс/см2	0 – 100
Разрешающая способность контроля давления, кгс/см	0,1
Допустимый диапазон скоростей звука в затрубном газе, м/с	250 – 450
Время непрерывной работы в режиме регистрации, час	100
Максимальная потребляемая мощность, мВт	200
Присоединительная резьба	2” НКТ 60
Масса, кг	2,3

Состояния датчика уровня

Датчик уровня может находиться в состояниях, которые перечислены ниже.

1 Свободен. В этом состоянии датчик не производит никаких действий и готов отвечать на запросы чтения и записи данных доступных регистров памяти.

2 Замер шумов. В этом состоянии датчик производит замер шумов в скважине.

3 Ожидание нажатия на ручной клапан. В этом состоянии датчик ожидает физического нажатия на ручной клапан для генерации акустических импульсов в затрубном пространстве.

4 Запись эхограммы. В этом состоянии датчик производит запись эхограммы.

5 Экспорт измерения. Данное состояние свидетельствует о завершении процесса записи эхограммы, и датчик подготовил все данные для экспорта.

На рисунке 2.4 отображена диаграмма переходов между состояниями датчика уровня.

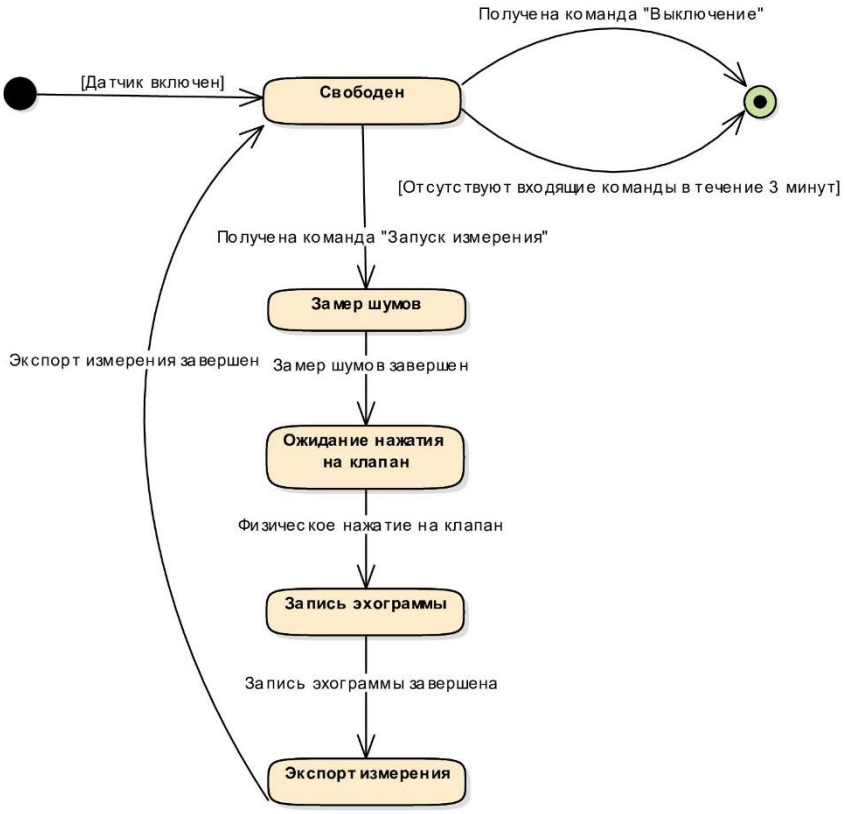

Рисунок 2.4 – Диаграмма переходов между состояниями датчика уровня

2.2 Датчик динамометрирования

Датчик динамометрирования предназначен для измерения и записи перемещения и величины нагрузки на полированный шток при работе ШГНУ.

На рисунке 2.4 представлен общий вид датчик динамометрирования.

Рисунок 2.5 – Общий вид датчика динамометрирования

Устройство предназначено для эксплуатации на устье скважины, оборудованной штанговой глубинно-насосной установкой (ШГНУ), на месторождениях нефтяной и газовой отрасли промышленности и имеет функции, которые перечислены ниже.

1 Измерение изменения нагрузки на полированном штоке ШГНУ в пределах (0 – 10000) кгс, перемещения полированного штока в пределах (0,5 – 9,999) м не менее одного цикла качания балансира ШГНУ.

2 Запись и сохранение измеряемых параметров в энергонезависимом запоминающем устройстве.

3 Передачу сохраненных параметров во внешнее устройство по беспроводному соединению посредством специализированного ПО.

4 Подключение к мультирежимному терминалу по каналу Bluetooth.

5 Активацию и обновление программного обеспечения датчика по каналу Bluetooth.

Устройство сохраняет работоспособность при температуре окружающей среды от минус 40 °С до + 50 °С. Устройство функционирует автономно и питается от специального внутреннего аккумулятора с напряжением 3,6 В или 3.7В. Минимальное рабочее напряжение аккумуляторной батареи, не приводящее к потере работоспособности устройства, составляет 3,0 В. Основные технические характеристики указаны в таблице 2.2 [3].

Таблица 2.2 – Основные характеристики датчика уровня

Наименование параметра	Норма по ТУ
Маркировка взрывозащиты	1Ex ib IIB T3 Gb X
Верхний предел изменения нагрузки, кгс	10000
Разрешающая способность канала нагрузки, (кгс) / перемещения, (м)	10 / 0,010
Точность измерения нагрузки, % от ВПИ	1%
Диапазон измерения перемещения, м	0,5 -10
Допустимый темп качаний балансира ШГНУ, кач/мин	0,4 – 15
Допустимый диаметр полированного штока ШГНУ, мм	16 – 37
Минимальный интервал записи динамограммы, мин	1
Время непрерывной работы в режиме регистрации, час	280
Время работы без замены элемента питания, не менее, год	1
Максимальная потребляемая мощность, мВт	200
Масса, кг	1,35

Необходимо отметить, что существует две вариации датчика динамометрирования. Датчик динамометрирования межтравесный (ДДИМ) и датчик динамометрирования накладной (ДДИН) имеют минимальные отличия в обусловленные типом крепления скважин и методикой эксплуатации.

Состояния датчика динамометрирования

Датчик динамометрирования может находиться в нескольких состояниях, которые перечислены ниже.

1 Свободен. В этом состоянии датчик не производит никаких действий и готов отвечать на запросы чтения и записи данных доступных регистров памяти.

2 Измерение. В этом состоянии датчик производит измерение нагрузки и ускорения.

3 Расчет. В этом состоянии датчик производит расчет динамограммы.

4 Экспорт измерения. Это состояние свидетельствует о том, что расчет динамограммы завершен и датчик подготовил все данные для экспорта.

На рисунке 2.6 отображена диаграмма переходов между состояниями датчика динамометрирования.

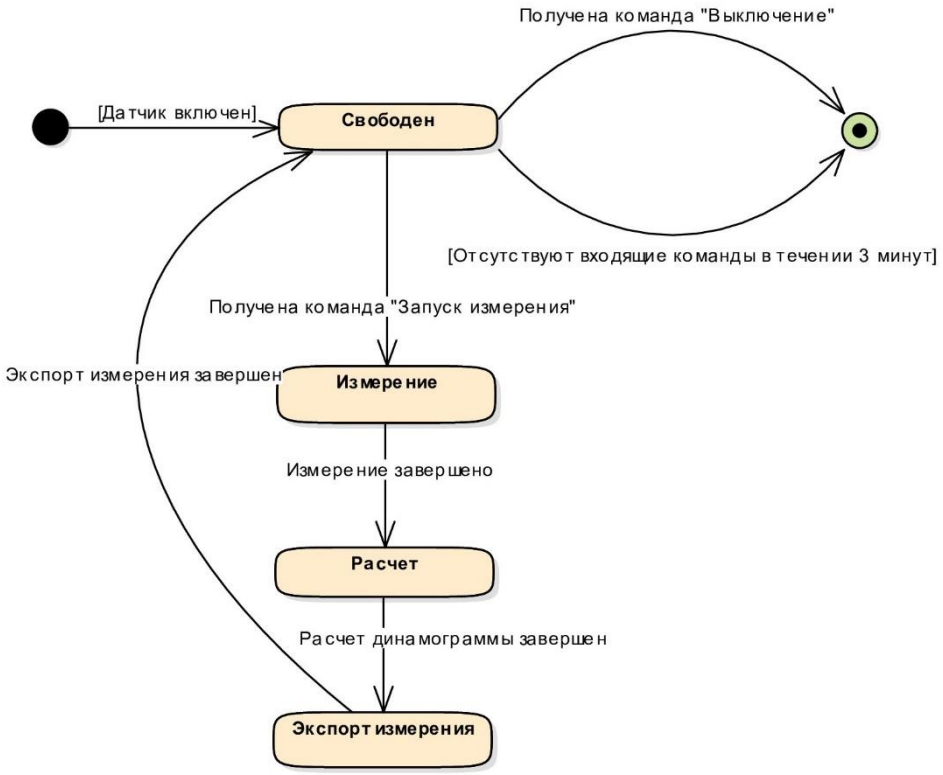

Рисунок 2.6 – Диаграмма переходов между состояниями датчика динамометрирования

2.3 Смартфон под управлением ОС Android

В базовую комплектацию стационарного комплекса контроля скважин оборудованных ШГНУ включен смартфон DEXP Ixion P350 Tundra под управлением операционной системы Android 7.0. Данный смартфон имеет поддержку технологий Bluetooth и Bluetooth Low Energy, что позволяет ему поддерживать связь со всеми датчиками комплекса. А также имеет модуль GSM для установления связи с сетью интернет для передачи данных измерений.

В таблицу 2.3 представлены основные характеристики смартфона DEXP Ixion P350 Tundra.

Таблица 2.3 – Основные характеристики смартфона DEXP Ixion P350 Tundra.

Наименование характеристики	Значение
Версия ОС	Android 7.0 Nougat
Модель процессора	MediaTek MT6580A
Объем оперативной памяти	1 ГБ
Версия Bluetooth	4.0
Поддержка сетей 3G	WCDMA 2100, WCDMA 1900, WCDMA 850, WCDMA 900
Поддержка сетей 2G	GSM 900, GSM 1900, GSM 1800, GSM 850
Емкость аккумулятора	5000 мАч

2.4 Протокол Bluetooth

В зависимости от конкретной модели датчика он может быть оборудован различным от другой модели того же датчика модулем Bluetooth. Существуют две вариации реализации модуля связи в датчиках ДУ, ДДИМ и ДДИН.

1 Модуль Laird BT900 использующий протокол Bluetooth 2.0.

2 Модуль Laird BL654 использующий протокол Bluetooth 4.0 (BLE).

Протокол Bluetooth предназначен для обеспечения беспроводной передачи данных в реальном времени между портативными и стационарными точками связи. Устройства в сети Bluetooth делятся на ведущие (master) и подчиненные (slave). Обмен данными может осуществляться только между ведущим устройством и подчиненным. Каждое устройство в сети может выступать и в роли ведущего и в роли подчиненного. В таблице 2.4 представлена радиоспецификация Bluetooth.

Таблица 2.4 – Радиоспецификация Bluetooth

Характеристика	Значение
Диапазон частот	2.4 ГГц, диапазон ISM 2.4000 - 2.4835 ГГц F = 2402 + k MHz, k = 0,…,78 – радиоканалы
Модуляция	GFSK – основная (BT = 0.5) π/4 - DQPSK, 8DPSK – для EDR * (roll – off = 0.4)
Максимальная скорость передачи данных в канале	1 Мбит/с – для GFSK 2 Мбит/с – для π/4 - DQPSK 3 Мбит/с – для 8DPSK
Схемы доступа	TDMA, FDMA, CDMA
Схема передачи	TDD
Расширение спектра	FHSS
Число радионесущих	79

Окончание таблицы 2.4

Характеристика	Значение
Расстояние между несущим	1 МГц
Скорость перестройки частоты	1600 скачков в секунду в режиме передачи; 3200 скачков в секунду в режимах опроса и запроса
Максимальная мощность передач	1 мВт - 100 мВт – класс 1; 0.25 мВт - 2.5 мВт – класс 2; до 1 мВт – класс 3

Состояния Bluetooth

Основные состояния:

– холостое состояние – низкое энергопотребление, работают только часы устройства;

– состояние соединения – устройство подключено к пикосети;

– состояние парковки – состояние подчинённого устройства, от которого не требуется участия в работе пикосети, но которое должно оставаться её частью.

Промежуточные состояния (для подключения к пикосети новых подчинённых устройств):

– опрос – определение устройством наличия других устройств в пределах его досягаемости;

– поиск опроса – ожидание устройством опроса;

– ответ на опрос–устройство, получившее опрос, отвечает на него;

– запрос – посылается одним устройством другому для установления с ним соединения (запрашивающее устройство становится ведущим, запрашиваемое – подчинённым);

– поиск запроса – устройство ожидает запрос;

– ответ подчинённого устройства – подчинённое устройство отвечает на запрос ведущего;

– ответ ведущего устройства – ведущее устройство отвечает подчинённому после получения от него ответа на запрос [3].

На рисунке 2.7 представлена структурная схема переходов между состояниями Bluetooth.

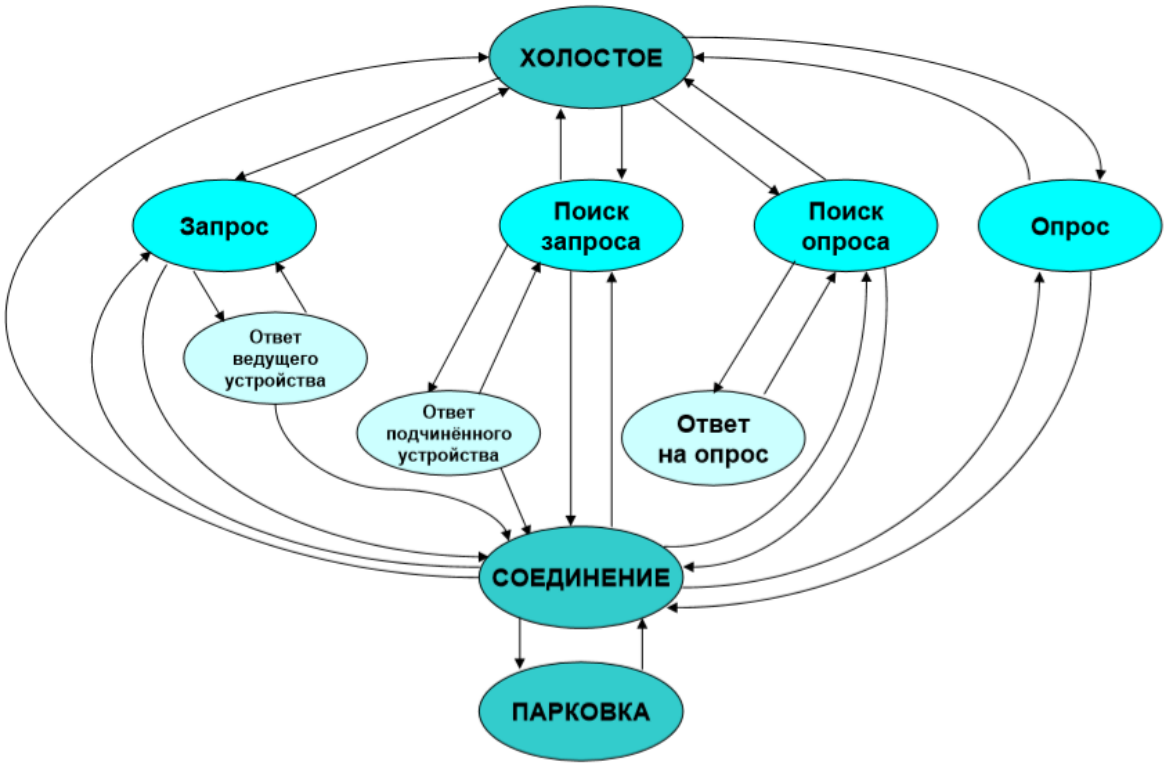

Рисунок 2.7 – Структурная схема переходов между состояниями Bluetooth

Режимы Bluetooth

1) Активный режим. Подчинённое устройство активно участвует в работе пикосети, ожидая, передавая и принимая пакеты. Ведущее устройство периодически передаёт подчинённому устройству пакеты для поддержания синхронизации.

2) Режим удержания. Устройство не поддерживает работу по асинхронным каналам, но может участвовать в обмене по каналам SCO/eSCO. В периоды неактивности устройство может переходить в режим пониженного энергопотребления, делать опросы, запросы, сканировать поисковые каналы или участвовать в работе другой пикосети. Режим удержания активен в

течение заранее определённого времени, по истечении которого устройство возвращается в предыдущий режим.

3) Режим подслушивания. Для передачи подчинённому устройству, находящемуся в режиме подслушивания, ведущее устройство выделяет по каналам ACL меньше слотов, чем обычно. Доступность синхронных каналов SCO и eSCO при этом не уменьшается. В периоды неактивности ACL-канала устройство может переключаться на другой физический канал (другая пикосеть) или переходить в режим энергосбережения.

4) Режим повышенной скорости передачи (EDR – Enhanced Data Rate). В данном режиме устройство может обмениваться информацией по каналам ACLU и eSCO-S с повышенной скоростью (до 3 Мбит/с) и поддерживать дополнительные типы пакетов.

2.5 Протокол обмена данными приборов ТНПВО “СИАМ”

Протокол предоставляет внешним устройствам возможность управления приборами и получения данных по каналу связи.

С точки зрения внешнего устройства все данные представлены в двоичном виде в адресном пространстве 0x00000000 - 0xFFFFFFFF (разрядность адреса – 4 байта). Обмен данными осуществляется с помощью операций записи и чтения. Все приборы имеют набор регистров, которые представлены в таблице 2.5.

Таблица 2.5 – Общие регистры

Регистр	Размер, байт	Адрес	Доступ
Тип прибора	2	0x00000000	Только чтение
Версия модели памяти	2	0x00000002	Только чтение
Адрес названия прибора	4	0x00000004	Только чтение
Размер названия прибора	2	0x00000008	Только чтение
Номер прибора	4	0x0000000A	Чтение/запись
Уникальный номер прибора	8	0x0000000E	Только чтение
Сетевой адрес прибора	2	0x00000016	Чтение/запись
Скорость обмена	4	0x00000018	Чтение/запись

Старший байт регистра 0x00 (тип прибора) определяет группу, к которой относится прибор, младший – модель прибора (номер разработки). В таблице 2.6 представлены коды групп приборов.

Таблица 2.6 – Группы приборов

Код	Группа приборов
0x0100, 0x0200	Уровнемеры
0x0300	Накладные динамографы
0x0400	Межтраверсные динамографы
0x0500	Устьевые манометры-термометры
0x0600	Глубинные манометры-термометры

Входящие сообщения для датчиков можно разделить на несколько типов приведенных ниже.

1 Сообщение запроса чтения. Сообщение представляет запрос на чтение данных из определенного участка памяти датчика исходящий от внешнего устройства. На рисунке 2.8 представлен формат сообщения запроса чтения.

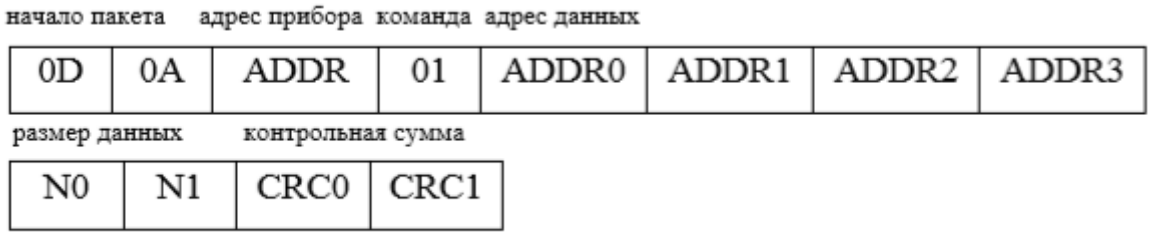

Рисунок 2.8 – Формат сообщения запроса чтения

2 Сообщение запроса записи. Сообщение представляет запрос на запись данных в определенный участок памяти датчика исходящий от внешнего устройства. На рисунке 2.9 представлен формат сообщения запроса записи.

Рисунок 2.9 – Формат сообщения запроса записи

3 Сообщение положительного ответа на запрос чтения. Сообщение представляет положительный ответ датчика внешнему устройству на запрос чтения и содержит запрашиваемые данные. На рисунке 2.10 представлен формат сообщения положительного ответа на запрос чтения.

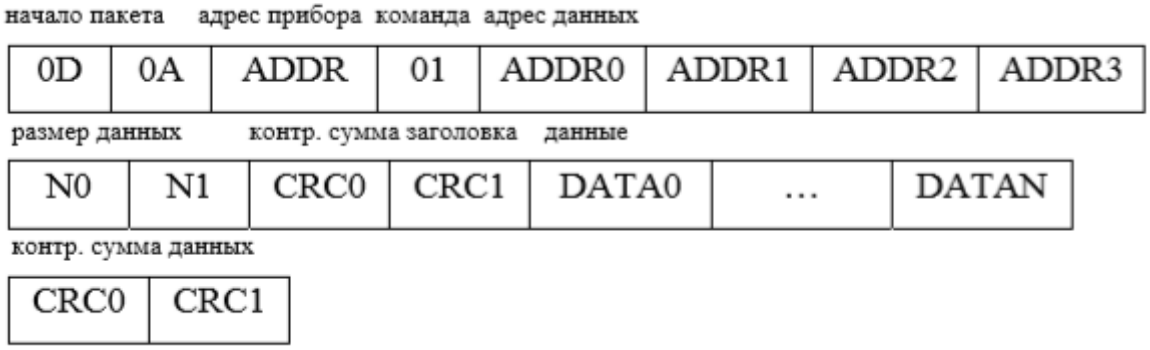

Рисунок 2.10 – Формат сообщения положительного ответа на запрос чтения

4 Сообщение положительного ответа на запрос записи. Сообщение представляет положительный ответ датчика внешнему устройству на запрос записи в определенный участок памяти. На рисунке 2.11 представлен формат сообщения положительного ответа на запрос записи.

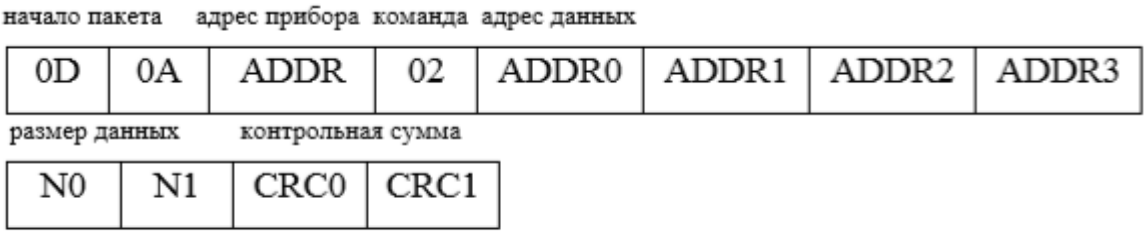

Рисунок 2.11 – Формат сообщения положительного ответа на запрос записи

5 Сообщение отрицательного ответа на запрос чтения. Сообщение представляет отрицательный ответ на запрос чтения данных из определенного участка памяти и содержит код ошибки. На рисунке 2.12 представлен формат сообщения отрицательного ответа на запрос чтения.

Рисунок 2.12 – Формат сообщения отрицательного ответа на запрос чтения

6 Сообщение отрицательного ответа на запрос записи. Сообщение представляет отрицательный ответ на запрос записи данных в определенный участок памяти и содержит код ошибки. На рисунке 2.13 представлен формат сообщения отрицательного ответа на запрос записи.

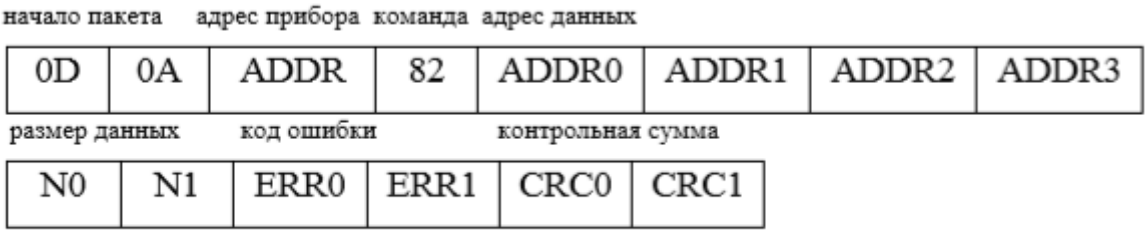

Рисунок 2.13 – Формат сообщения отрицательного ответа на запрос записи

Данные передаются младшим байтом вперед. Контрольная сумма подсчитывается по алгоритму CRC16 (CRC-16-IBM, CRC-16-ANSI). Начало пакета (0x0D, 0x0A) при подсчете контрольной суммы не учитывается. При

неверной контрольной сумме заголовка команда не выполняется и ответ не передается. При неизвестной команде никаких действий не производится и ответ не отправляется. При неверной контрольной сумме данных команда не выполняется, но передается отрицательный ответ с кодом ошибки. В таблице 2.7 представлены коды общих ошибок.

Таблица 2.7 – Коды общих ошибок

Код	Ошибка
0x0001	Недопустимый адрес
0x0002	Недопустимый размер данных
0x0003	Недопустимые данные
0x0004	Недопустимая операция (запись только читаемого регистра)
0x0005	Неверная контрольная сумма данных

Коды других ошибок определяются типом прибора и назначаются с номера 0x0100 [5].

3 API датчиков

Используя описанный в главе 2.5 протокол можно обмениться данными с любым датчиком стационарного комплекса контроля скважин оборудованных ШГНУ. Но нужно учитывать, что каждый тип датчика имеет свою структуру организации регистров памяти и соответственно отличный API. Далее будет рассмотрен API датчика динамометрирования и датчика уровня.

Вне зависимости от типа все датчики имеют одинаковый интерфейс доступа для регистров в адресном пространстве 0x00000000 – 0x00001000. Регистры в данном адресном пространстве называются общими и приведены в таблице 3.1.

Таблица 3.1 – Общие регистры

Адрес	Назначение	Тип данных	Режим доступа	Комментарий
0x00000000	Тип прибора	UInt16	Только чтение	0x1101 – ДУ 0x1302 – ДДИН, 0x1401 - ДДИМ
0x00000002	Версия модели данных	UInt16	Только чтение	
0x00000004	Адрес названия прибора	UInt32	Только чтение	
0x00000008	Размер названия прибора	UInt16	Только чтение	
0x0000000A	Заводской номер	UInt32	Только чтение	

3.1 API датчика уровня (ДУ)

В данном разделе будут рассмотрены основные регистры датчика ДУ, которые требуются для осуществления работы. В таблице 3.2. приведены основные регистры датчика уровня.

Таблица 3.2 – Основные регистры датчика уровня

Адрес	Назначение	Тип данных	Режим доступа
Регистры текущих данных			
0x00008400	Напряжение аккумулятора	UInt16	Только чтение
0x00008404	Давление, атм	UInt16	Только чтение
Операционные регистры			
0x00008800	Запуск процесса	UInt8	Только запись
0x00008802	Состояние датчика	UInt16	Только чтение
Данные исследования			
0x80000000	Заголовок исследования	UInt16[2]	Только чтение
0x81000000	Эхограмма	UInt8[3000]	Только чтение

Отдельно требуется прокомментировать регистр “*Запуск процесса*”. Датчик ДУ имеет возможность путем передачи в регистр запуска процесса определенного кода запускать соответствующий коду процесс. Коды процессов датчика ДУ приведены в таблице 3.3.

Таблица 3.3 – Коды процессов датчика уровня

Код процесса	Описание
1	Запуск измерения
2	Запуск очистки данных
3	Обнуление состояния
4	Выключение датчика

3.2 API датчиков динамометрирования (ДДИМ\ДДИН)

В данном разделе будут рассмотрены основные регистры датчиков динамометрирования (ДДИМ\ДДИН), которые требуются для осуществления работы. В таблице 3.4. приведены основные регистры датчика динамометрирования.

Таблица 3.4 – Основные регистры датчика динамометрирования

Адрес	Назначение	Тип данных	Режим доступа
Регистры текущих данных			
0x00008400	Напряжение аккумулятора, В	UInt16	Только чтение
0x00008402	Температура, °C	UInt16	Только чтение
0x00008404	Нагрузка, мВ	Float	Только чтение
0x00008408	Ускорение, мВ	Float	Только чтение
Операционные регистры			
0x00008800	Запуск процесса	UInt16	Только запись
0x00008802	Состояние датчика	UInt16	Только чтение
Данные исследования			
0x80000000	Заголовок исследования	UInt16[14]	Только чтение
0x81000000	Динамограмма	Byte[3000]	Только чтение

Отдельно требуется прокомментировать регистр “*Запуск процесса*”. Датчики ДДИМ и ДДИН имеют возможность путем передачи в регистр запуска процесса определенного кода запускать соответствующий коду процесс. Коды процессов датчика ДДИМ и ДДИН приведены в таблице 3.5.

Таблица 3.5 – Коды процессов датчика уровня

Код процесса	Описание
1	Запуск измерения
4	Выключение датчика

4 Постановка задачи

Для управляющего блока (смартфона) стационарного комплекса контроля скважин оборудованных ШГНУ требуется разработать программное обеспечение, функциональность перечисленную ниже.

1 Сканирование эфира Bluetooth для поиска устройств.

2 Подключение к датчикам типов ДДИМ, ДДИН, ДУ по беспроводному каналу связи Bluetooth.

3 Отображение текущих данных датчиков ДДИМ, ДДИН, ДУ.

4 Запуск процесса длительного исследования физических величин на датчиках ДДИМ, ДДИН, ДУ.

5 Загрузка результатов длительных исследований из датчиков.

6 Визуализация результатов исследований.

7 Отправка результатов исследований по почте.

В связи с динамически меняющимися запросами и требованиями бизнеса одним из важнейших требований является возможность без больших временных и денежных затрат обеспечить замену любого модуля в любой системе. В контексте комплекса контроля скважин оборудованных ШГНУ требуется предусмотреть возможность замены управляющего блока, представленного в настоящее время смартфоном под управлением операционной системы Android 7.0, другим устройством, в том числе и на другой платформе.

Разработку приложения можно декомпозировать на условные модули:

1 разработка модуля сканирования Bluetooth эфира;

2 разработка модуля взаимодействия с устройствами по Bluetooth;

3 разработка модуля внедрения зависимостей;

4 разработка модуля взаимодействия с датчиками;

5 разработка модуля работы с данными;

6 разработка графического интерфейса приложения.

5 Выбор технологий разработки

Один из основных моментов, которые необходимо учитывать при разработке мобильных приложений - операционная система, под управление которой функционирует устройство. Так как именно тип операционной системы во многом определяет процесс разработки программного обеспечения для мобильных устройств. В настоящий момент можно выделить несколько наиболее популярных операционных систем. В таблице 5.1 представлены популярные операционные системы для мобильных устройств и их характеристики, важные для разработки программного обеспечения.

Таблица 5.1 – Популярные операционные системы

Операционная система	Разработчик	Исходный код	Язык программирования для нативной разработки
Android	Google	Открытый	Java, Kotlin
iOS	Apple	Закрытый	Objective-C, Swift
UWP	Microsoft	Закрытый	C#

Исходя из таблицы 5.1 нужно отметить, что для каждой операционной системы имеется свой язык разработки приложений. Из этого вытекает серьезная проблема – каждая платформа имеет свой язык программирования для разработки программного обеспечения. То есть в случае, когда стоит задача разработки приложения на несколько платформ, необходимо описывать одну и ту же бизнес логику на каждом из языков конкретной платформы. Такой подход влечет за собой существенные проблемы при сопровождении и масштабировании приложения, а также требует больших затрат временных и, как следствие, денежных ресурсов.

Решением данной проблемы является применение кроссплатформенного подхода к процессу разработки.

5.1 Общие сведения о кроссплатформенной разработке

Кроссплатформенная разработка приложений - это процесс или подход к разработке приложений, которые могут работать на нескольких платформах. Эти приложения используют повторно используемый код, который можно повторно использовать для создания одной и той же формы для нескольких платформ. Кроссплатформенная разработка обычно используется разработчиками приложений для сокращения времени разработки и получения экономически эффективных результатов. Однако кроссплатформенная разработка приложений может привести к ряду проблем, связанных с производительностью.

Тем не менее, этот метод подхода имеет свои преимущества и недостатки.

Плюсы кроссплатформенного подхода разработки

Ниже перечислены положительные стороны кроссплатформенного подхода к процессу разработки приложений.

1 Переиспользуемый код. Одним из наиболее значительных преимуществ этого метода разработки приложений является повторно используемый код, который разработчик может использовать на разных платформах. Поэтому вместо разработки отдельных приложений для отдельных платформ разработчик может использовать и настраивать один код для разных платформ. Более того, этот же код можно использовать и для будущих проектов.

2 Сокращенное время разработки. Как упоминалось в предыдущем пункте, повторно используемый код позволяет разработчикам использовать один код на всех платформах. Это приводит к значительному сокращению времени разработки и позволяет разработчикам быстро завершать проекты разработки.

3 Экономическая эффективность. Поскольку один код можно использовать на всех платформах, владельцам приложений не нужно тратить деньги на наем отдельных разработчиков для разных платформ. Это экономит

ресурсы. Например, разработка приложений для iPhone и разработка приложений для Android, если они выполняются отдельно, обходится дорого.

4 Расширенный охват рынка. Одним из ограничений разработки единой платформы является ограниченный охват. Если приложение разработано специально для iPhone, Android или Windows, пользователи не смогут получить к нему доступ. А поскольку разработка единой платформы является дорогостоящей, владельцы приложений воздерживаются от разработки одного и того же приложения на разных платформах. Но кроссплатформенная разработка предоставляет новые возможности для расширения пользовательской базы и охват рынка, что в других случаях ограничено в одноплатформенном приложении.

5 Простота сопровождения. Кроссплатформенные приложения легко редактировать и обновлять. Поскольку все приложения используют стандартный код, обновление может выполняться на всех платформах, что приводит к бесперебойной работе.

Минусы кроссплатформенного подхода разработки

Хотя кроссплатформенная разработка имеет массу интересных преимуществ, у этого подхода также есть несколько недостатков, о которых речь пойдет ниже.

1 Ограниченные обновления. Операционная система может не поддерживать все функции, используемые фреймворком. Например, если iOS выпускает обновление или добавляет новый элемент, необходимо соответствующим образом обновить версию iOS для вашего приложения. Однако вы не можете сделать то же самое с приложением для Android, пока Google не выпустит аналогичное обновление.

2 Слабосвязанный код. Разработка кроссплатформенных приложений не так проста, как кажется, особенно для начинающих разработчиков. Разработчики должны использовать инверсию контроля и прочие подходы в процессе разработки, которые могут привести к слабосвязанному коду и, следовательно, к медленному применению [6].

5.2 Сравнение фреймворков кроссплатформенной разработки

Кроссплатформенная разработка является непростым, но правильным подходом для решения поставленной задачи. Правильный выбор фреймворка является ключевым моментом при проектировании и реализации программных систем. В этой главе будет произведен краткий обзор фреймворков кроссплатформенной разработки приложений.

React Native

React Native позволяет создавать мобильные приложения, используя только JavaScript. React Native использует тот же дизайн, что и React, позволяя создавать богатый мобильный интерфейс из декларативных компонентов. С React Native вы не создаете «мобильное веб-приложение», «приложение HTML5» или «гибридное приложение». Вы создаете настоящее мобильное приложение, которое неотличимо от приложения, созданного с использованием Objective-C или Java. React Native использует те же основные строительные блоки пользовательского интерфейса, что и обычные приложения для iOS и Android. Вы просто соединяете эти строительные блоки, используя JavaScript и React. React Native плавно сочетается с компонентами, написанными на Objective-C, Java или Swift. Просто перейти к нативному коду, если вам нужно оптимизировать несколько аспектов вашего приложения. Также легко создать часть вашего приложения в React Native, а часть приложения - напрямую с использованием нативного кода - так работает приложение Facebook [7].

Flutter

Фреймворк от компании Google для создания кроссплатформенных приложений, которые используют общий код. Фреймворк позволяет создавать веб-приложения, мобильные приложения под iOS и Android, а также десктопные приложения операционных систем Windows, MacOS и Linux.

Особенность фреймворка Flutter заключается в том, что приложения под разные платформы могут иметь общий программный код. Но по причине неэквивалентности платформ некоторые отдельные модули необходимо

настраивать под конкретную операционную систему, несмотря на это большая часть кода может совпадать. Эта особенность позволяет сэкономить время и ресурсы на разработку приложений.

В качестве языка программирования используется относительно молодой язык программирования Dart. При сборке приложения Flutter транслирует кода на Dart в нативный код приложения, которое можно запускать на любой из платформ [8].

Ionic

IonicFramework - это SDK с открытым исходным кодом для разработки гибридных мобильных приложений с использованием веб-технологий. С Ionic можно создавать кроссплатформенные гибридные приложения. Он тесно взаимодействует с фреймворком Apache Cordova, который преобразовывает веб-приложения в мобильные программы.

Ionic завоевал признание среди разработчиков мобильных приложений, потому что с ним просто работать. Фреймворк построен на ECMAScript 6 и TypeScript, поэтому его можно использовать в любой IDE, поддерживающей эти языки, например в Visual Studio Code, Atom или Angular IDE. Ionic, как и React Native и Flutter, предлагает концепцию единого кода для разных платформ, но на новом уровне. Все его компоненты автоматически адаптируются к платформе, на которой запускается приложение – а значит, разработка становится быстрее. Также с Ionic есть возможность использовать JavaScript, Angular, React или Vue.

В производительности Ionic серьезно проигрывает от React Native и Flutter, поскольку для визуализации приложений он использует веб-технологии и совсем не применяет нативные компоненты. Такой подход значительно снижает скорость. Но со стороны разработки есть и плюсы: Ionic позволяет проводить быстрое тестирование, которое можно запустить прямо в браузере [9].

Xamarin

Xamarin расширяет платформу для разработчиков .NET инструментами и библиотеками специально для создания приложений для Android, iOS, tvOS, watchOS, macOS и Windows.

Независимо от того, разрабатываете ли вы унифицированный интерфейс для разных платформ или создаете собственный пользовательский интерфейс, ваши приложения будут вести себя так, как ожидают пользователи.

Благодаря возможности доступа ко всему спектру функциональности, предоставляемой базовой платформой и устройством, а также использованию аппаратного ускорения для конкретной платформы, приложения Xamarin компилируются для собственной производительности.

Xamarin.Android и Xamarin.iOS наделяют приложение теми же возможностями и интерфейсом, которые есть у нативных решений. В случае Xamarin.iOS программа компилируется непосредственно в машинный код (AOT-компиляция), тогда как в Xamarin.Android сначала происходит компиляция в байт-код, который затем интерпретируется виртуальной машиной (JIT-компиляция).

Если же нужно ускорить процесс написания кода, лучше использовать Xamarin.Forms – более простой инструмент, в котором почти все элементы полностью совместимы с любыми платформами [10].

PhoneGap

Как и Ionic, PhoneGap позволяет использовать веб-технологии JavaScript в связке с HTML и CSS в мобильной разработке. Он является дистрибутивом Apache Cordova.

Приложение PhoneGap, по сути, представляет собой набор HTML-страниц, обёрнутых в нативную оболочку. Страницы хранятся в локальном каталоге или в облаке, а во время запуска на смартфоне они получают доступ к функциям устройства через плагины. Это делает приложения PhoneGap довольно лёгкими, но они выглядят менее естественно, а качество

пользовательского интерфейса будет в большей степени зависеть от веб-представления конкретной ОС. PhoneGap отличается невысокой производительностью по сравнению с нативными инструментами по причине использования веб-технологий [11].

Итог обзора технологий

В таблице 5.2 представлена краткая сводка по фреймворкам для кроссплатформенной разработки.

Таблица 5.2 – Популярные операционные системы

Характеристика	React Native	Flutter	Ionic	Xamarin	PhoneGap
Язык	JavaScript + React	Dart	JavaScript + HTML, CSS + Angular, React, Vue	C# + .NET	JavaScript, HTML, CSS
Приложения	Кроссплатформенные	Кроссплатформенные	Кроссплатформенные гибридные	Кроссплатформенные	Кроссплатформенные гибридные
Первый релиз	2015	2017	2013	2011	2009
Разработчик	Facebook + сообщество	Google + сообщество	Drifty Co	Microsoft	Adobe
Сообщество	Очень большое	Небольшое, но активно развивается	Большое	Большое	Большое
Платформы	Android, iOS, UWP	Android, iOS, Google Fuchsia, Web, Desktop	Android, iOS, Web	Android, iOS, UWP	Android, iOS, Windows Phone 8
Открытый исходный код	Да	Да	Да + платные пакеты	Да + платные пакеты	Да

Окончание таблицы 5.2

Характеристика	React Native	Flutter	Ionic	Xamarin	PhoneGap
Инструменты фронтенда	Компоненты Native + Declarative UI	Встроенные виджеты	HTML, CSS + виджеты	Xamarin.Android/Xamarin.iOS/Xamarin.Forms	HTML, CSS
Производительность	Высокая	Очень высокая	Средняя	Высокая	Средняя

Важный момент, на который стоит обратить это язык программирования, фреймворка. Фреймворки React Native, Ionic и PhoneGap базируются на веб-технологиях и языке программирования JavaScript. Применение языков программирования с динамической типизацией, таких как JavaScript, в приложениях, направленных на решения проблем бизнеса, влечет за собой возможные проблемы при сопровождении, так как такие языки больше пригодны для прототипирования, чем для корпоративной разработки. Внедрение решений на основе языков с динамической типизацией требует более длительного тестирования.

Фреймворки Flutter и Xamarin предоставляют примерно одинаковые возможности. Flutter базируется на языке с статической типизаций Dart, а Xamarin языке программирования C# также с статической типизацией, и эта особенность делает эти инструменты более надежными, по сравнению с прочими фреймворками использующими динамическую типизацию. Первая версия языка Dart вышла в 2014 году, а сам фреймворк Flutter в 2017. На этом фоне Xamarin является более надежным, так как его поддержка и обновление ведется с 2013 года, за это время вокруг этого инструмента успело сформироваться большое сообщество разработчиков, что немаловажно. Основывая на всех вышеприведенных аргументах можно остановить выбор технологии для разработки кроссплатформенных приложений на Xamarin от компании Microsoft.

5.3 Краткое описание фреймворка Xamarin

Фреймворк Xamarin предоставляет два подхода разработке приложений, которые приведены ниже.

1 Xamarin.Essentials предоставляет разработчикам кроссплатформенные API-интерфейсы для мобильных приложений. Android, iOS и UWP (универсальная платформа Windows) предоставляют разные API-интерфейсы операционной системы и платформы, к которым разработчики могут обращаться из кода C# с помощью Xamarin. Xamarin.Essentials обеспечивает единый кроссплатформенных API-интерфейс, который предоставляет доступ из общего кода для любого приложения Xamarin.Forms, Android, iOS или универсальной платформы Windows независимо от используемого метода создания пользовательского интерфейса.

2 Xamarin.Forms — это платформа пользовательского интерфейса с открытым кодом. С помощью Xamarin.Forms разработчики могут создавать приложения для Android, iOS и Windows на основе общей базы кода. Xamarin.Forms позволяет разработчикам создавать пользовательские интерфейсы в XAML с помощью кода программной части в C#. Эти интерфейсы на каждой платформе подготавливаются к просмотру как собственные элементы управления [10].

Исходя из поставленной задачи в приложении не требуется реализовывать сложный интерфейс, требуется лишь предусмотреть возможность расширения приложения на другие платформы. По этой причине более оптимально использовать Xamarin.Forms. Xamarin.Forms позволит единожды описать пользовательский интерфейс, который будет корректно использоваться на всех платформах. К тому же в Xamarin.Forms используется шаблон проектирования MVVM (Model-View-ViewModel).

Разработка приложения в Xamarin.Forms предполагает описание графического интерфейса с помощью XAML, и последующую имплементацию программных компонентов использующий ранее описанный интерфейс. В процессе развития приложений и расширения функционала

нередко возникают проблемы обсуживания. Такие проблемы темно связаны с взаимодействием между бизнес-логикой и пользовательским интерфейсом, что увеличивают затрачиваемое количество ресурсов на внесение изменений в пользовательский интерфейс, а также отрицательно сказывается на сложности модульного тестирования.

Шаблон Model-View-ViewModel (MVVM) позволяет четко разграничить бизнес-логику и модель представления приложения от пользовательского интерфейса. Придерживаясь принципа чистого разделения бизнес-логики приложения и пользовательского интерфейса можно устранить многие проблемы разработки и упростить тестирование, а также сопровождение и развитие приложения. Шаблон MVVM позволяет значительно улучшить возможность повторного использования кода. Немаловажным аспектом является возможность упростить совместную работу дизайнеров пользовательского интерфейса и разработчиков программных модулей при разработке приложения.

В шаблоне MVVM есть три основных компонента, которые приведены ниже.

1 Модель (Model). Компонент отвечает з классы, которые не являются визуальными и описываю модель предметной области, включая модель данных предметной области приложения и бизнес-процессы.

2 Представление (View). Компонент отвечает за пользовательский интерфейс, который описан с помощью XAML и минимального количества программного кода, не затрагивающего бизнес-процессы.

3 Модель представления (ViewModel). Компонент отвечает за связь между моделью и представлением уведомляя представление о изменениях состояния модели. Модель представления предоставляет свойства и команды, которые определяют функциональные возможности пользовательского интерфейса, но ответственность за то, как эти функции должны отображаться, лежит на представлении.

На рисунке 5.1 показаны связи между компонентами шаблона MVVM.

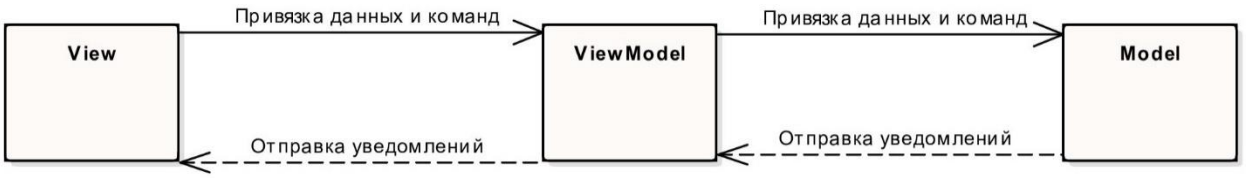

Рисунок 5.1 – Связи между компонентами MVVM

Ниже приведены преимущества использования шаблона проектирования MVVM.

1 Нередко, когда изменение существующей реализации модели, которая инкапсулирует описание бизнес-процесса, является сложной или рискованной, модель представления является адаптером для классов модели, позволяя избежать существенных изменений существующей кодовой базы.

2 Простота реализации модульного тестирования модели и модели представления без использования представления. Модульные тесты для модели представления могут использовать те же команды и свойства, которые используются в представлении.

3 Пользовательский интерфейс, при условии, что он полностью реализован с помощью XAML, можно легко заново переконструировать, при этом никак не затронув модель и модель представления.

4 Разработчики программных моделей и дизайнеры пользовательского интерфейса могут работать одновременно и независимо, сосредоточившись только на собственных ответственностях [12].

6 Практическая часть

На рисунке 6.1 представлена концептуальная диаграмма компонентов разрабатываемого решения.

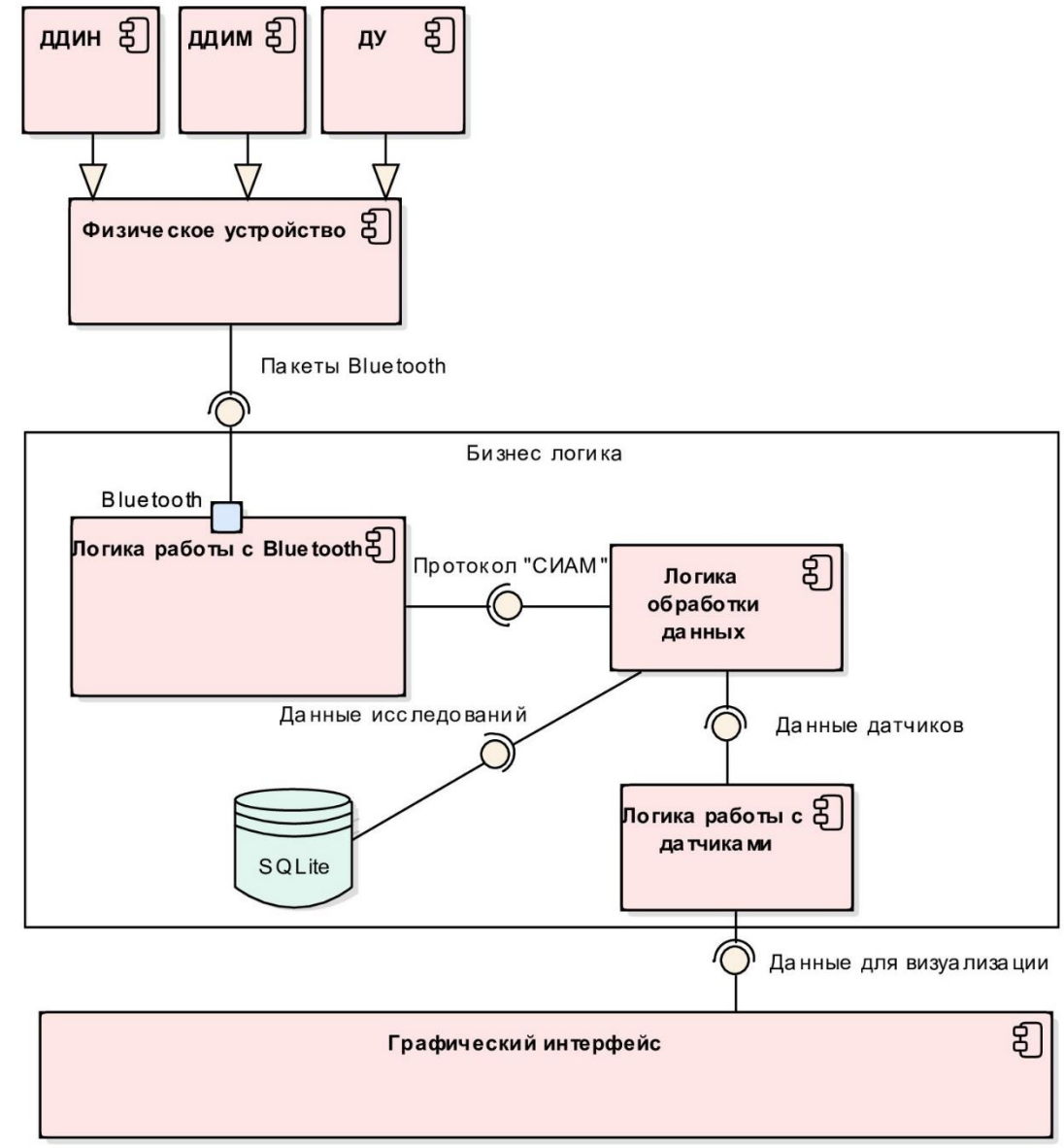

Рисунок 6.1 – Концептуальная диаграмма компонентов решения

Пакеты Bluetooth от датчиков поступают в приложение. Обработанные пакеты Bluetooth представляют из себя информационные сообщения по протоколу “СИАМ”. Информационные сообщения интерпретируются компонентом, ответственным за логику работы с датчиками, и передаются для обработки, для последующей записи в базу данных SQLite или отображения на графическом интерфейсе.

В связи с тем, что платформа iOS редко применяется в сфере решения бизнес задач по причине высокой стоимости устройств под управлением данной операционной системы, было принято решение, для доказательства кроссплатформенности разрабатывать конечный продукт на платформах Android и UWP.

Проект разделен на три логических решения, которые с пояснениями приведены ниже.

1 SiamService. В этом решении находится весь программный код, который не зависит от платформы, на который будет выполняться.

2 SiamService.Android. В этом решении находится программный код, который описывает объекты, зависимые от платформы Android.

3 SiamService.UWP. В этом решении находится программный код, который описывает объекты, зависимые от платформы UWP.

6.1 Описание работы Bluetooth

Реализация алгоритмов работы с Bluetooth использует драйвера конкретной операционной системы. Эта особенность делает программный код взаимодействия с Bluetooth платформозависимым, то есть зависящими от реализации конкретной платформы. Также важным аспектом является то, что реализация протокола Bluetooth 2 отлична от реализации протокола Bluetooth 4 (BLE), но протоколы можно обобщить единым интерфейсом.

На рисунке 6.2 представлена UML-диаграмма классов, реализованных для взаимодействия с Bluetooth.

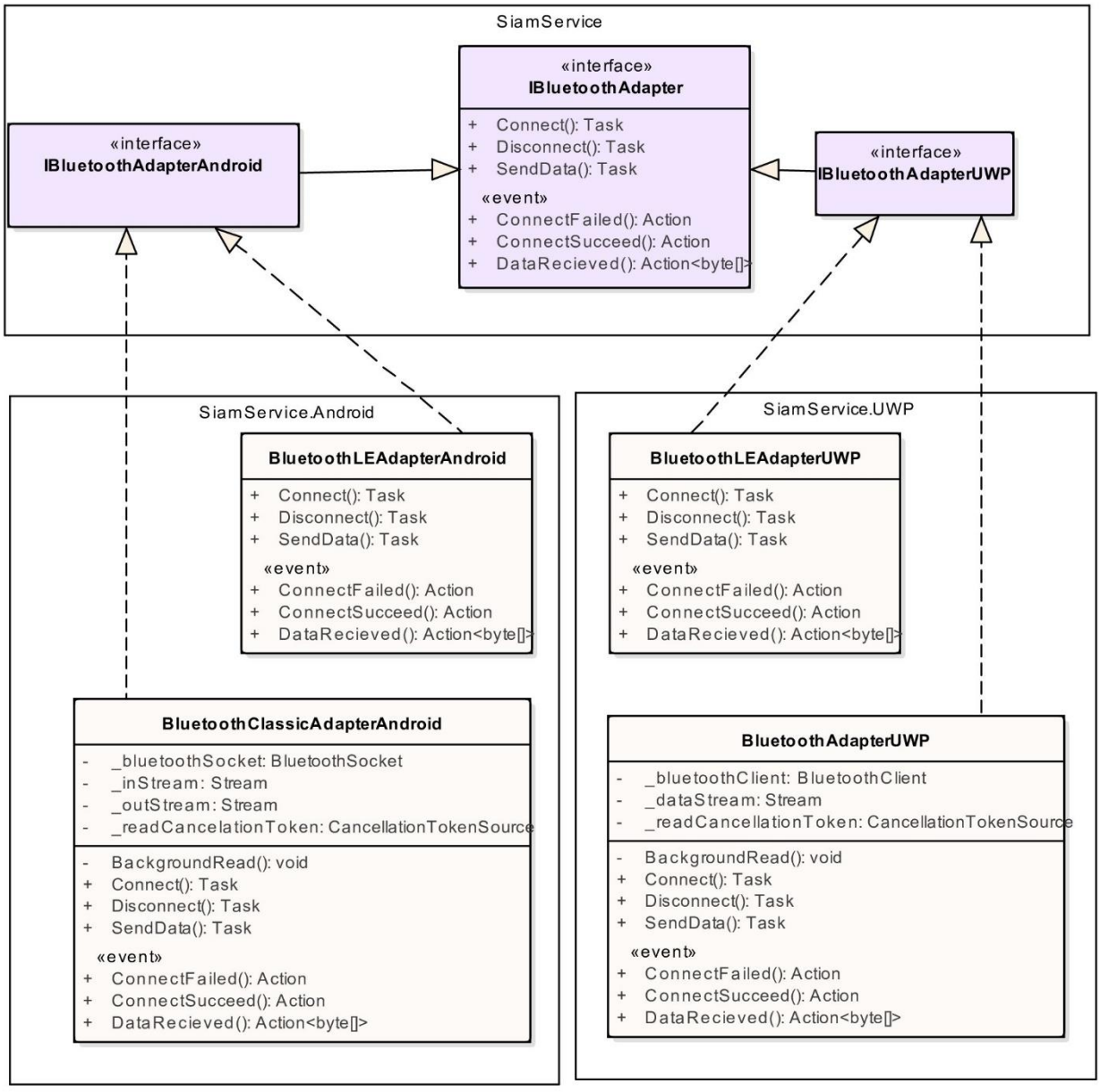

Рисунок 6.2 – UML-диаграмма бизнес логики Bluetooth

Объект *IBluetoothAdapter* представляет обобщенный интерфейс для работы с объектами реализации бизнес логики Bluetooth. В таблице 6.1 представлено описание интерфейса *IBluetoothAdapter*.

Таблица 6.1 – События и методы интерфейса *IBluetoothAdapter*

Название	Тип	Описание
События интерфейса		
ConnectFailed	Action	Событие разрыва физического соединения
ConnectSucceed	Action	Событие успешного соединения с физическим устройством
DataRecieved	Action<byte[]>	Событие приема данных от физического устройста
Методы интерфейса		
Connect()	Task	Установить соединение с физическим устройством
Disconnect()	Task	Разорвать соединение с физическим устройством
SendData(byte[])	Task	Отправить данные физическому устройству

Интерфейс *IBlutoothAdapterAndroid* и *IBluetoothAdapterUWP* наследуют интерфейс *IBlutoothAdapter* и представляют абстракции интерфейса работы с Bluetooth для соответствующих платформ.

Класс *BluetoothLEAdapterAndroid* реализующий интерфейс *IBluetoothAdapterAndroid* берет на себя ответственность по работе с устройствами по протоколу Bluetooth 4 (BLE) на платформе Android. В реализации этого класса используется компонент с открытым исходным кодом Plugin.BLE для реализации взаимодействия с драйвером BLE [13].

Класс *BluetoothClassicAdapterAndroid* реализующий интерфейс *IBluetoothAdapterAndroid* берет на себя ответственность по работе с

устройствами по протоколу Bluetooth 2 на платформе UWP. Для взаимодействия с драйвером Bluetooth Classic используются стандартные нативные инструменты из пространства имен Android.Bluetooth. В таблице 6.2 представлены поля и методы класса *BluetoothClassicAdapterAndroid.*

Таблица 6.2 – Поля и методы класса *BluetoothClassicAdapterAndroid*

Название	Тип	Описание
Поля класса		
_bluetoothSocket	BluetoothSocket	Bluetooth сокет
_inStream	Stream	Входящий поток данных
_outStream	Stream	Исходящий поток данных
_readCancellationTocken	CancelletionTokenSource	Токен отмены процесса чтения
Методы класса		
BackgroungRead	void	Чтение потока данных с физического устройства

Класс *BluetoothLEAdapterUWP* реализующий интерфейс *IBluetoothAdapterUWP* берет на себя ответственность по работе с устройствами Bluetooth 4 (BLE) на платформе UWP. Для взаимодействия с драйвером BLE используются стандартные инструменты из пространства имен Windows.Devices.Bluetooth.

Класс *BluetoothClassicAdapterUWP* реализующий интерфейс *IBluetoothAdapterUWP* берет на себя ответственность по работе с устройствами 2 на платформе UWP. Для взаимодействия с драйвером Bluetooth Classic используются компонент с открытым исходным кодом 32feet.NET [14]. В таблице 6.3 представлены поля и методы класса *BluetoothClassicAdapterUWP*.

Таблица 6.4 – Поля и методы класса *BluetoothClassicAdapterUWP*

Название	Тип	Описание
Поля класса		
_bluetoothClient	BluetoothClient	Bluetooth клиент
_dataStream	Stream	Входящий поток данных
_readCancellationTocken	CancelletionTokenSource	Токен отмены процесса чтения
Методы класса		
BackgroungRead	void	Чтение потока данных с физического устройства

При обобщении протокола BLE и Bluetooth Classic возникает следующая проблема. Реализация инструментов работы с драйвером Bluetooth Classic не предусматривает события приема данных, так как Bluetooth Classic использует абстракцию Stream для обмена данными. Данную проблему решает приватный метод *BackgroungRead*. Блок-схема метода *BackgroungRead* представлена на рисунке 6.3.

Рисунок 6.3 – Блок-схема метода *BackgroungRead*

6.2 Поиск Bluetooth устройств

Для сканирования эфира Bluetooth для дальнейшего подключения разработана архитектура классов, UML-диаграмма которой приведена на рисунке 6.4.

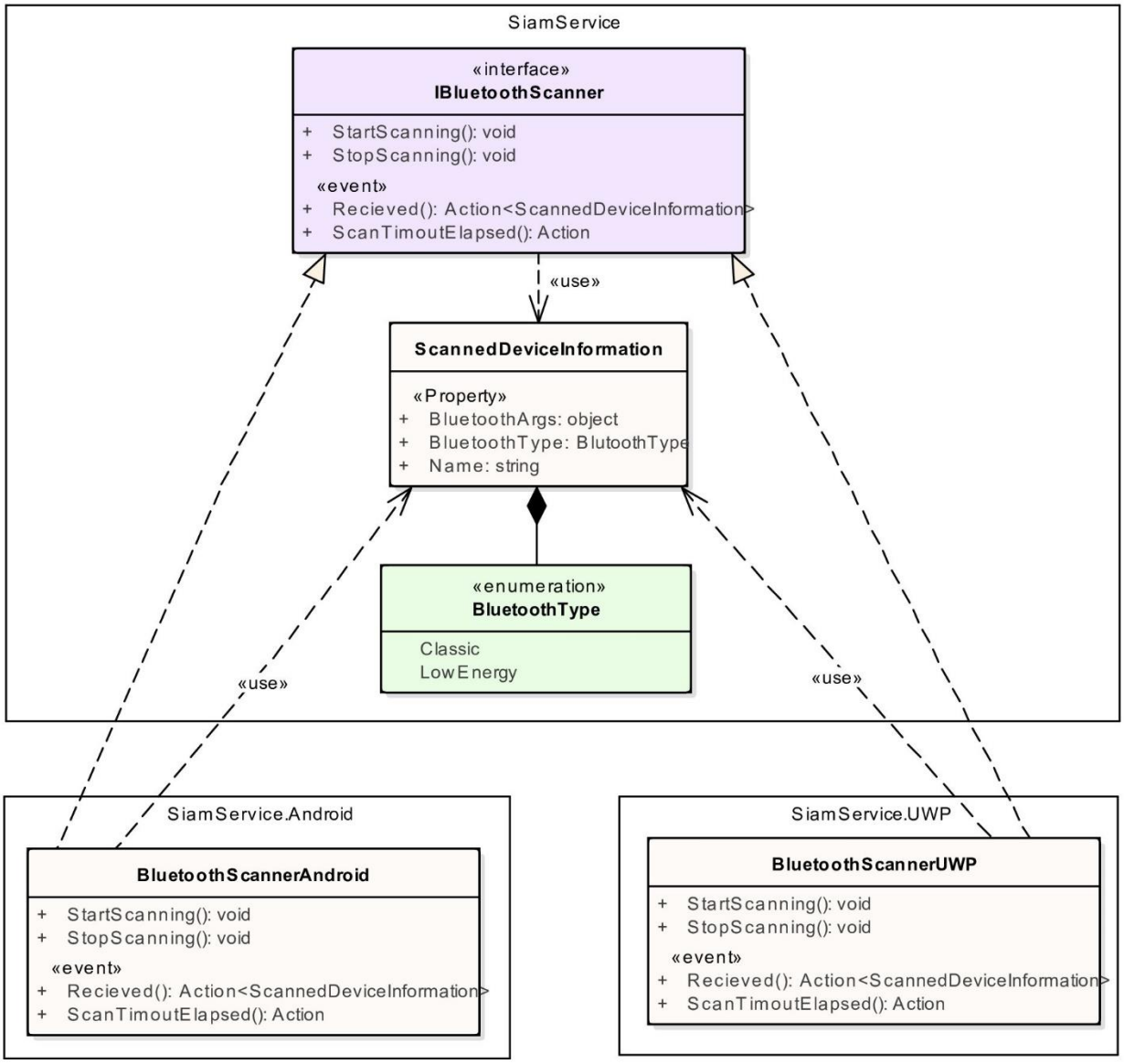

Рисунок 6.4 – UML-диаграмма бизнес логики сканирования эфира Bluetooth

Интерфейс *IBluetoothScanner* определяет общие методы и события у платформозависимых объектов *BluetoothScannerAndroid* и *BluetoothScannerUWP*. В таблице 6.5 представлены события и методы интерейса *IBluetoothScanner*.

Таблица 6.5 – События и методы интерфейса *IBluetoothScanner*

Название	Тип	Описание
События интерфейса		
Recieved	Action <ScannedDeviceInformation>	Событие успешного обнаружения устройства в Bluetooth эфире
ScanTimeoutElapsed	Action	
Методы интерфейса		
StartScanning	void	Запустить сканирование
StopScanning	void	Остановить сканирование

Класс *ScannedDeviceInformation* определяет аргументы описывающие характеристики найденного устройства. В таблице 6.6 представлены поля класса *ScannedDeviceInformation*.

Таблица 6.6 – Поля класса *ScannedDeviceInformation*

Название	Тип	Описание
BluetoothArgs	object	Системные параметры устройства
BluetoothType	BluetoothType	Тип протокола
Name	string	Имя устройства

6.3 Внедрение зависимостей

В главах 6.1 и 6.2 были описаны платформозависимые модули приложения. Для того, чтобы использовать платформозависимые модули требуется применить внедрение зависимостей. В качестве IoC-контейнера используется Autofac.

Autofac управляет зависимостями между классами, чтобы приложения оставались легко меняющимися по мере роста и сложности. Это достигается путем обработки обычных классов .NET как компонентов. В программной инженерии инверсия управления (IoC) является принципом разработки, в котором пользовательские части компьютерной программы получают поток управления из общей структуры. Архитектура программного обеспечения с этим дизайном инвертирует контроль по сравнению с традиционным процедурным программированием: в традиционном программировании пользовательский код, который выражает цель программы, вызывает в библиотеки многократного использования, чтобы заботиться об общих задачах, но с инверсией управления, это структура который вызывает в пользовательском или заданном конкретном коде [15].

На рисунке 6.5 приведена UML-диаграмма логики внедрения зависимостей.

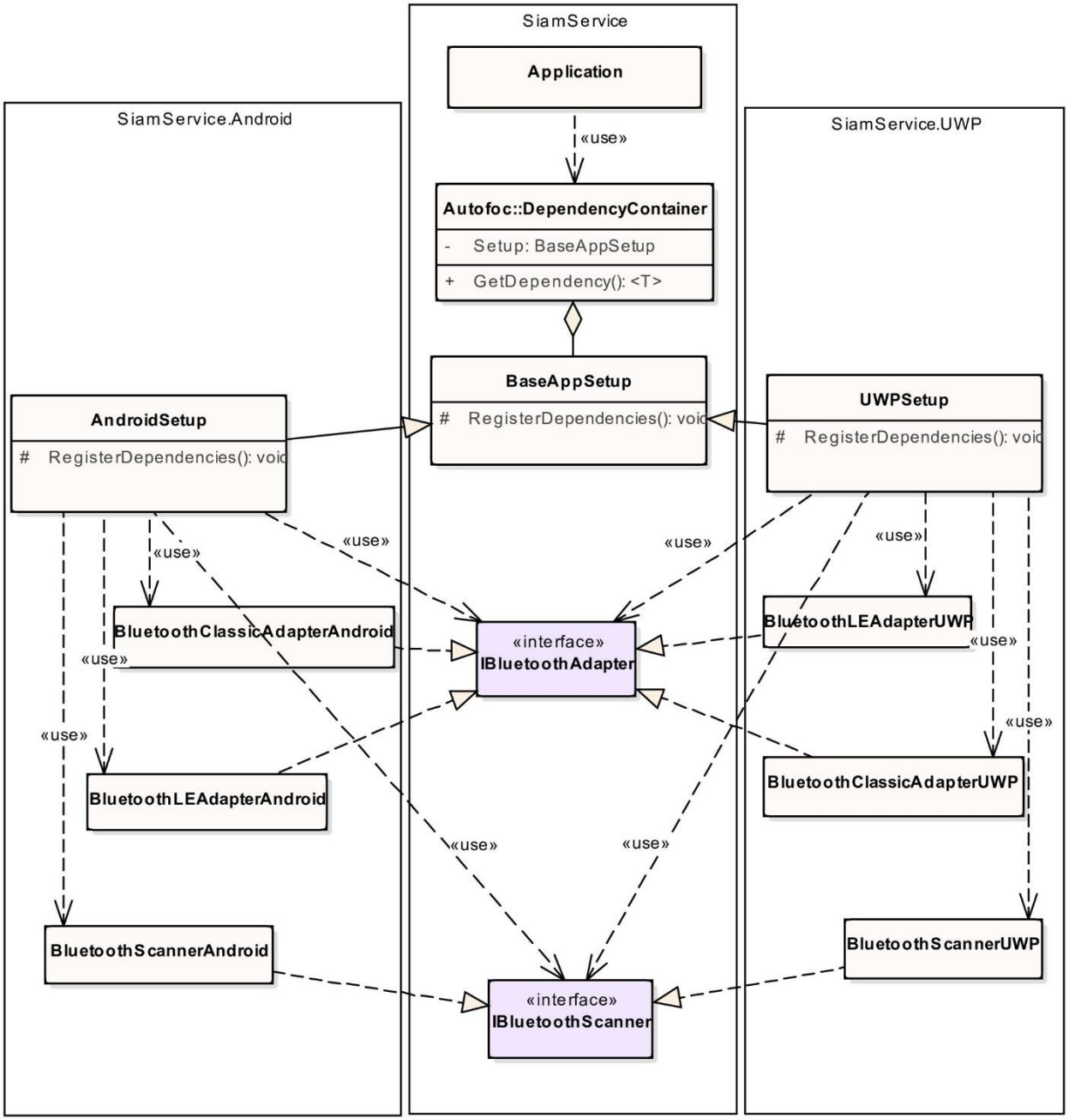

Рисунок 6.5 – UML-диаграмма логики внедрения зависимостей

Объекты *AndroidSetup* и *UWPSetup* принимают на себя ответственность за регистрацию конкретных реализаций объектов для абстракций *IBluetoothAdapter* и *IBluetoothScanner*. Объект *DependencyContainer* из пространства имен *Autofac* содержит все зависимости и по требованию клиента, который на диаграмме обозначен классом *Application,* может разрешить любую из них через метод *GetDependency*.

6.4 Взаимодействие с датчиками

Разработка программного модуля для взаимодействия с датчиками заключается в описании программных сущностей датчиков ДУ, ДДИМ и ДДИН, каждый из которых использует собственный API, и может иметь реализацию модуля связи базируясь на технологии Bluetooth Classic или BLE. При всех различиях часть алгоритмов взаимодействия с устройствами и алгоритмов синтаксического анализа сообщений является общим вне зависимости от типа датчика.

Доводы, приведенные выше, создают предпосылки к применению принципа разработки DRY (Don’t Repeat Youself). Принцип DRY – это принцип разработки программного обеспечения с множеством слоев абстрагирования, который нацелен на снижение количества повторяемой информации [16].

В программном модуле взаимодействия с датчиками принцип DRY реализуется путем комбинации поведенческого паттерна *Шаблонный метод* и порождающего паттерна проектирования *Фабричный метод*. Паттерн проектирования *Шаблонный метод* определяет алгоритм, некоторые этапы которого делегируются подклассам, позволяя подклассам переопределить эти этапы, не меняя структуру алгоритма. Паттерн *Фабричный метод* определяет интерфейс для создания объекта, но позволяет подклассам определять, какой класс инстанцировать [17].

Для реализации частично общего для всех типов датчиков синтаксического анализа информационных сообщений используется *Шаблонный метод*. В контексте приложения данный паттерн позволяет использовать общую структуру алгоритма синтаксического анализа сообщений, но, так как датчики имеют различный API, реализация алгоритма синтаксического анализа для каждого конкретного устройства может переопределять этапы базовой реализации данного алгоритма. В связи с этим возникает необходимость в объекте, обязанность которого заключается в создании сложных программных объектов датчиков, определяя при этом

соответствующую конкретную реализацию алгоритма синтаксического анализа, но скрывающего его от клиентов. Для этого реализован паттерн *Фабричный метод*.

Архитектура модуля взаимодействия с датчиками представлена на рисунке 6.6.

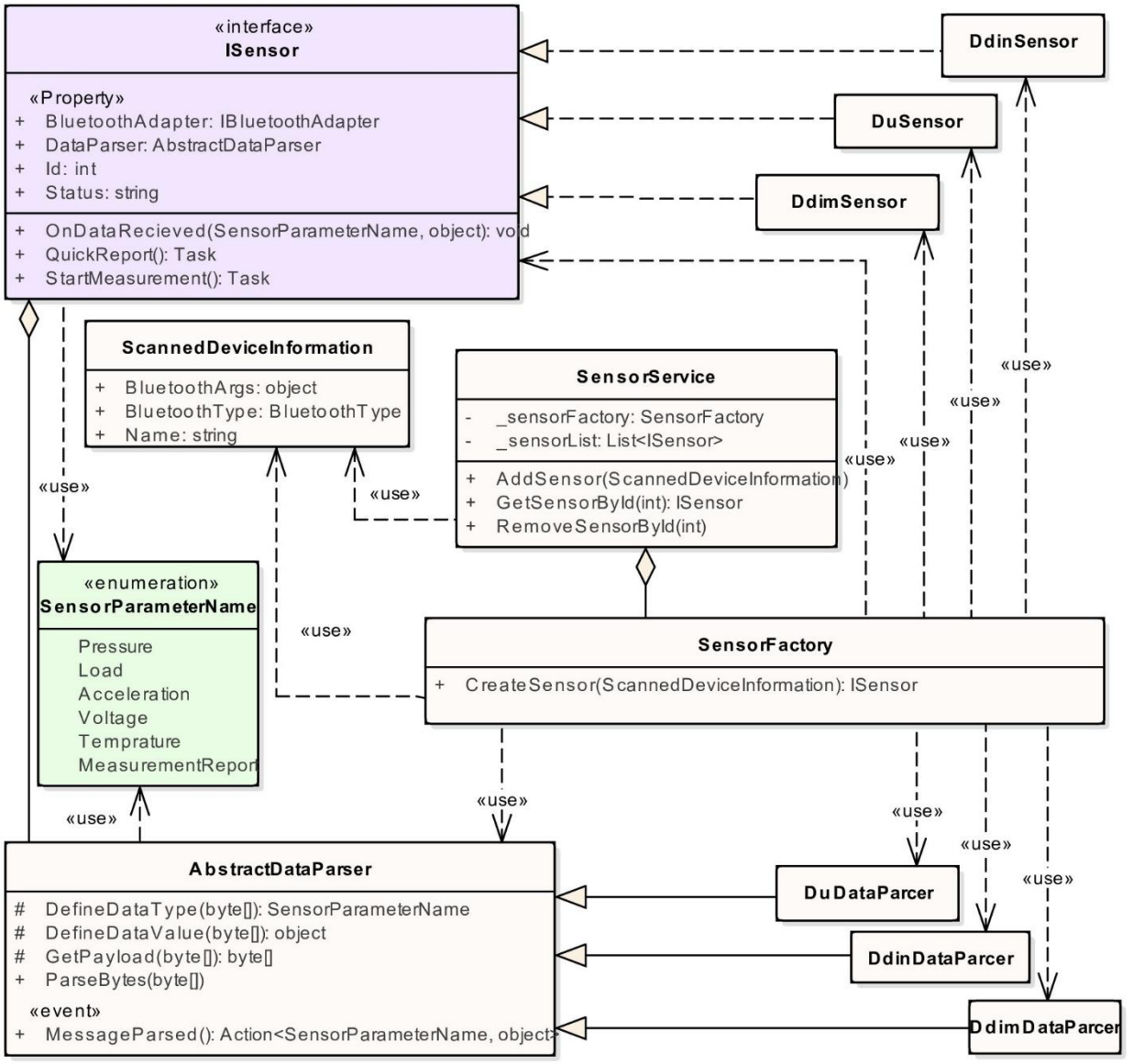

Рисунок 6.6 – UML-диаграмма бизнес логики взаимодействия с датчиками

На диаграмме объект *ISensor* определяет интерфейс, описывающий ряд общих свойств и методов, реализованных в датчиках. Классы *DdimSensor*, *DdinSensor* и *DuSensor* являются реализациями данного интерфейса. В таблице 6.7 представлены свойства и методы интерфейса *ISensor*.

Таблица 6.7 – Свойства и методы интерфейса *ISensor*

Название	Тип	Описание
Свойства интерфейса		
BluetoothAdapter	IBluetoothAdapter	Адаптер протокола Bluetooth
DataParser	AbstractDataParser	Синтаксический анализатор сообщений
Id	int	Уникальный присвоенный номер
Status	string	Строковое представление статуса устройства
Методы интерфейса		
OnDataRecieved	void	Обработчик события приема сообщения
QuickReport	Task	Запрос у физического устройства быстрого отчета
StartMeasurement	Task	Запуск измерения на физическом устройстве

Класс SensorService является объектом-контейнером для датчиков. В таблице 6.8 представлены его поля и методы.

Таблица 6.8 – Поля и методы интерфейса SensorService

Название	Тип	Описание
Поля класса		
_sensorFactory	SensorFactory	Фабрика для создания датчиков
_sensorList	List<ISensor>	Список добавленных датчиков
Методы класса		
AddSensor	void	Добавить датчик для работы
RemoveSensorById	void	Удалить датчик по Id
GetSensorById	ISensor	Добавить датчик по Id

AbstractDataParser это объект, обязанность которого определение структуры алгоритма синтаксического анализа информационных сообщений датчиков по протоколу “СИАМ”. Его дочерние объекты DuDataParser, DdimDataParser и DdinDataParcer при необходимости переопределяю этапы алгоритма синтаксического анализа, что является реализацией паттерна проектирования *Шаблонный метод*. В таблице 6.9 представлены методы и события класса *AbstractDataParser*.

Таблица 6.9 – Методы и события класса *AbstractDataParser*

Название	Тип	Описание
Методы класса		
DefineDataType	SensorParameterName	Определить тип данных в сообщении
DefineDataValue	object	Определить значение параметра
GetPayload	byte[]	Получить полезные данные из сообщения
ParseBytes	void	Анализировать входящие байты данных
События класса		
MessageParsed	Action<SensorParameterName, object>	Добавить датчик для работы

Для создания сложных объектов датчиков реализован класс *SensorFactory*. Получая от клиента на вход метода *CreateSensor* информацию об устройстве, данный класс определяет реализацию алгоритма синтаксического анализа и конкретную реализацию объекта *ISensor,* что является воплощением шаблона проектирования *Абстрактная фабрика*.

6.5 Работа с данными

Важнейшим модулем приложения является модуль обработки и хранения результатов исследований датчиков. Для хранения результатов измерений используется легковесная база данных SQLite.

SQLite – это библиотека на языке C, которая реализует небольшой, быстрый, автономный, высоконадежный, полнофункциональный механизм базы данных SQL. SQLite – самый распространенный в мире модуль баз данных. SQLite встроен во все мобильные телефоны и большинство компьютеров и поставляется внутри множества других приложений, которые люди используют каждый день.

Формат файла SQLite является стабильным, кроссплатформенным и обратно совместимым, и разработчики обязуются сохранять его таким, по крайней мере, до 2050 года. Файлы базы данных SQLite обычно используются в качестве контейнеров для передачи богатого контента между системами и в качестве долговременного архивного формата данных.

Исходный код SQLite находится в свободном доступе и может использоваться всеми для любых целей [18].

На рисунке 6.7 приведена UML-диаграмма модели базы данных для хранения результатов исследований.

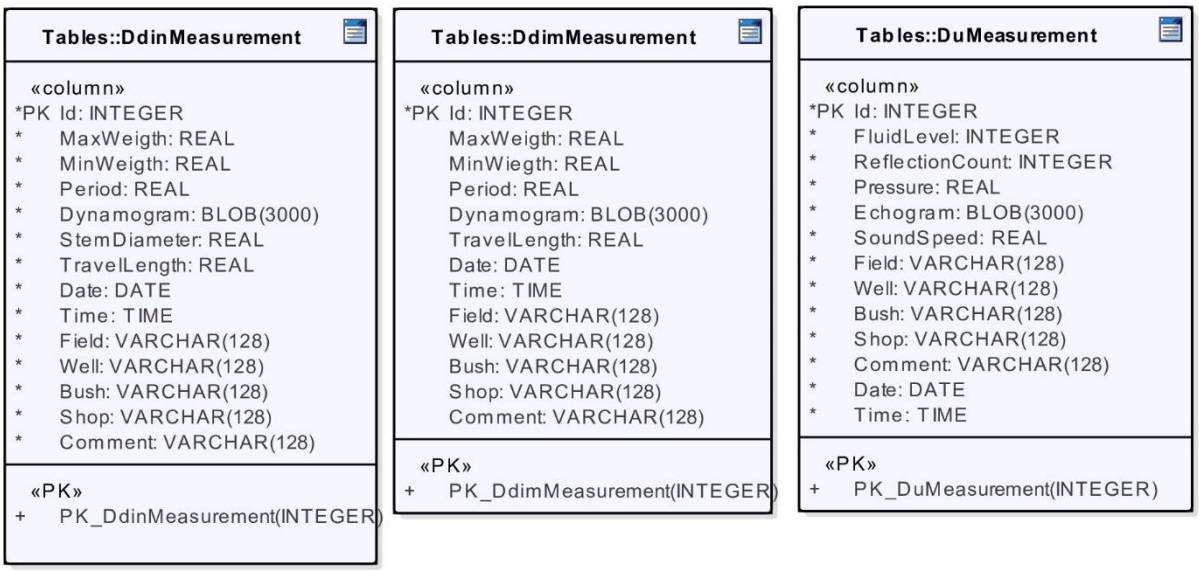

Рисунок 6.7 – UML-диаграмма модели базы данных

В таблице 6.10 представлено писание полей таблицы DdinMeasurement.

Таблица 6.10 – Описание полей таблицы DdinMeasurement

Имя	Тип	Описание
MaxWeigth	REAL	Максимальная нагрузка
MinWeigth	REAL	Минимальная нагрузка
Period	REAL	Период
Dynamogram	BLOB(3000)	Динамограмма
StemDiameter	REAL	Диаметр штока
TravelLength	REAL	Длина хода
Date	DATE	Дата
Time	TIME	Время
Field	VARCHAR(128)	Месторождение
Well	VARCHAR(128)	Скважина
Bush	VARCHAR(128)	Куст
Shop	VARCHAR(128)	Цех
Comment	VARCHAR(128)	Комментарий

В таблице 6.11 представлено писание полей таблицы DuMeasurement.

Таблица 6.11 – Описание полей таблицы DuMeasurement

Имя	Тип	Описание
FluidLevel	INTEGER	Уровень жидкости
ReflectionCount	INTEGER	Количество отражений
Pressure	REAL	Давление
Echogram	BLOB(3000)	Эхограмма
SoundSpeed	REAL	Скорость звука
Field	VARCHAR(128)	Месторождение
Well	VARCHAR(128)	Скважина
Shop	VARCHAR(128)	Куст

Окончание таблицы 6.11

Имя	Тип	Описание
Comment	VARCHAR(128)	Цех
Date	DATE	Дата
Time	TIME	Время

В таблице 6.12 представлено писание полей таблицы DdimMeasurement.

Таблица 6.12 – Описание полей таблицы DdimMeasurement

Имя	Тип	Описание
MaxWeigth	REAL	Максимальная нагрузка
MinWeigth	REAL	Минимальная нагрузка
Period	REAL	Период
Dynamogram	BLOB(3000)	Динамограмма
TravelLength	REAL	Длина хода
Date	DATE	Дата
Time	TIME	Время
Field	VARCHAR(128)	Месторождение
Well	VARCHAR(128)	Скважина
Bush	VARCHAR(128)	Куст
Shop	VARCHAR(128)	Цех
Comment	VARCHAR(128)	Комментарий

Для взаимодействия с объектами базы данных в приложении как с классами используется компонент Dapper ORM. Dapper - это объектно-реляционное отображение (ORM) для платформы .NET. Он обеспечивает основу для сопоставления объектно-ориентированной модели предметной области с традиционной реляционной базой данных. Dapper обеспечивает мапинг между базами данных и объектами .NET [19].

На рисунке 6.8 представлена UML-диаграмма классов модуля работы с данными.

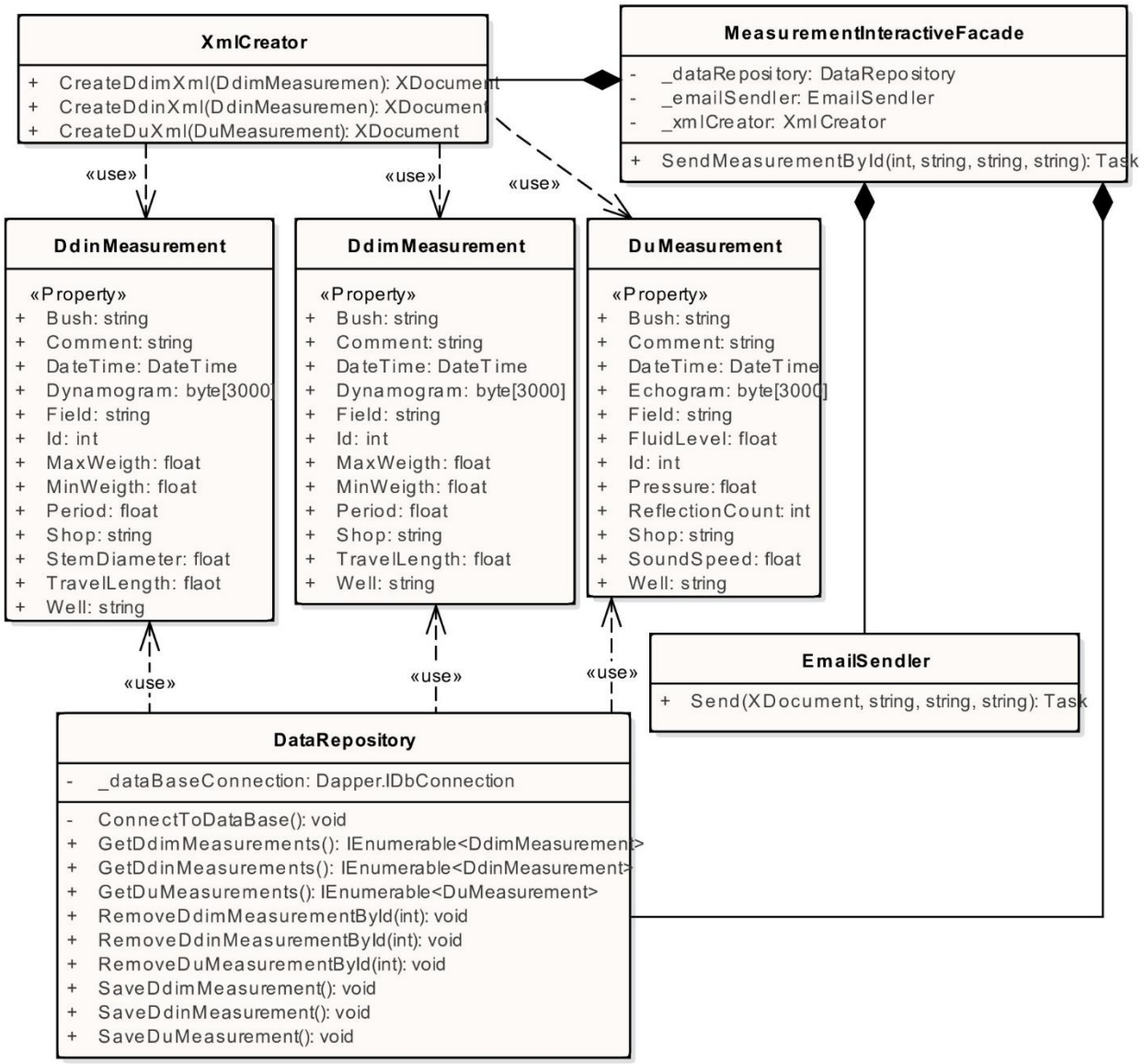

Рисунок 6.8 – UML-диаграмма бизнес-логики работы с данными

Классы *DdinMeasurement*, *DdimMeasurement* и *DuMeasurement* являются объектным представлением реляционных объектов базы данных.

Класс *DataRepository* является объектом-оберткой над базой данных, беря на себя ответственность взаимодействия с ней. В таблице 6.13 приведены поля и методы класса *DataRepository*.

Таблица 6.13 – Поля и методы класса *DataRepository*

Название	Тип	Описание
Поля		
_dataBaseConnection	Dapper.IDbConnection	Соединение с базой данных
Методы		
ConnectToDataBase	void	Соединиться с базой данных
GetDdimMeasurements	IEnumerable <DdimMeasurement>	Получить результаты всех измерений ДДИМ
GetDdinMeasurements	IEnumerable <DdinMeasurement>	Получить результаты всех измерений ДДИН
GetDuMeasurements	IEnumerable <DuMeasurement>	Получить результаты всех измерений ДУ
SaveDdimMeasurement	void	Сохранить измерение ДДИМ
SaveDdinMeasurement	void	Сохранить измерение ДДИН
SaveDuMeasurement	void	Сохранить измерение ДУ
RemoveDdimMeasurementById	void	Удалить измерение ДДИМ по Id
RemoveDdinMeasurementById	void	Удалить измерение ДДИН по Id
RemoveDuMeasurementById	void	Удалить измерение ДУ по Id

Класс *XmlCreator* берет на себя ответственность за конвертацию результатов исследований всех датчиков в формат XML.

Класс *EmailSandler* берет на себя ответственность отправки писем с приложенным файлом по на заданную почту получателя от имени заданного отправителя.

Класс *MeasurementInteractiveFacade* предоставляет высокоуровневый интерфейс к подсистеме взаимодействия с данными. Принимая на вход метода *SendMeasurementById* целочисленный идентификатор измерения, адрес получателя, адрес отправителя и пароль, объект *MeasurementInteractiveFacade* реализует отправку конкретного результат исследования, который, будучи извлеченным из базы данных, конвертируется в формат XML.

6.6 Графический интерфейс

Для использования всех возможностей приложения пользователем необходимо разработать графический интерфейс. На рисунке 6.9 представлена UML диаграмма вариантов использования.

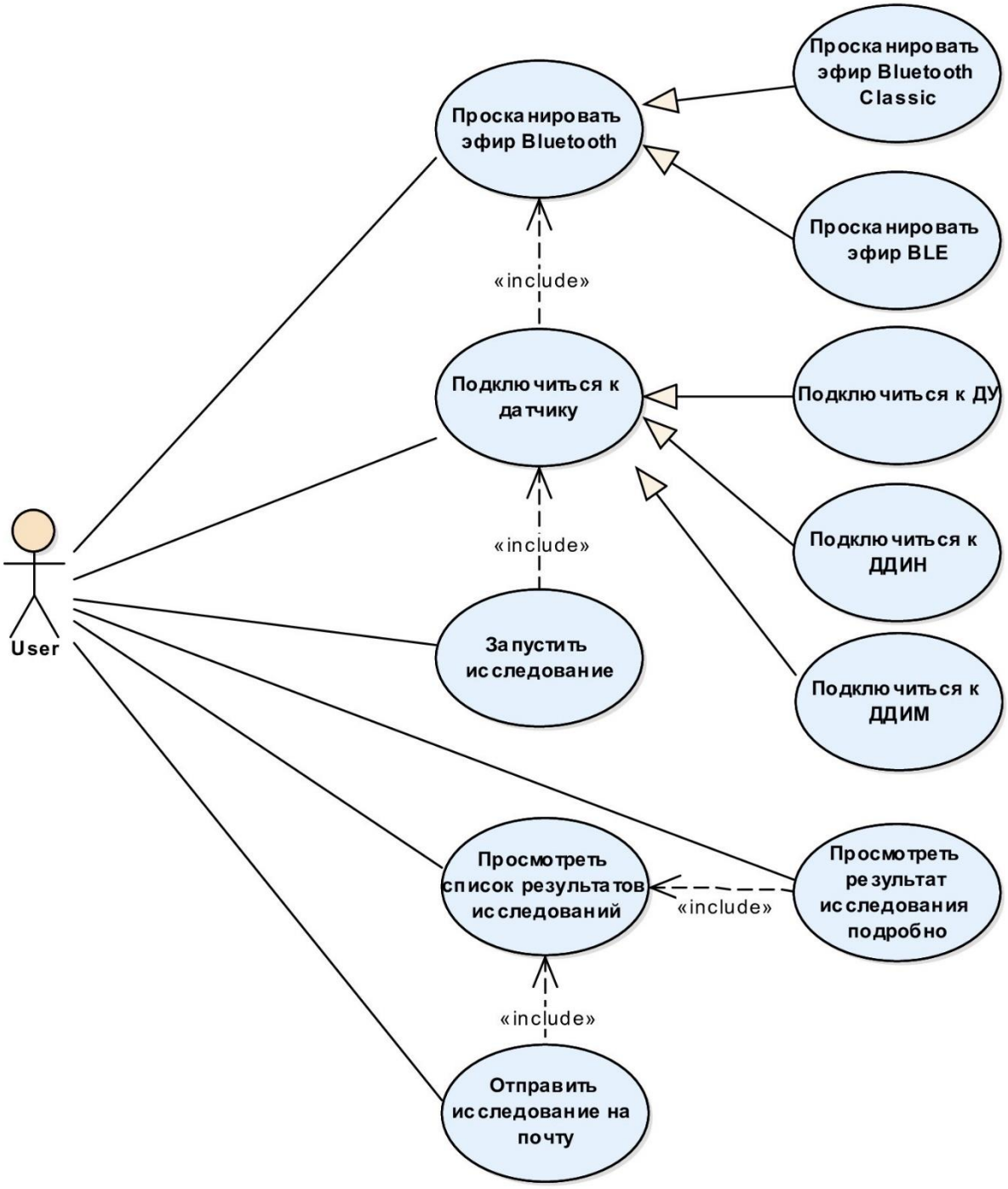

Рисунок 6.9 – Диаграмма вариантов использования

В Xamarin.Forms для разработки графического интерфейса используется XAML. XAML позволяет разработчикам определять

пользовательские интерфейсы в Xamarin.Forms приложениях, используя разметку, а не код. Код XAML никогда не нужен в Xamarin.Forms программе, но часто является более сжатым и более наглядным, чем эквивалентный код, и, возможно, Инструментарий. XAML хорошо подходит для использования с популярной архитектурой приложения MVVM (Model-View-ViewModel): XAML определяет представление, связанное с кодом ViewModel через привязки данных на основе XAML.

В XAML-файле Xamarin.Forms разработчик может определять пользовательские интерфейсы, используя все Xamarin.Forms представления, макеты и страницы, а также пользовательские классы. XAML может быть скомпилирован или внедрен в исполняемый файл. В любом случае данные XAML анализируются во время сборки, чтобы определить именованные объекты, и снова во время выполнения для создания и инициализации объектов, а также для установки связей между этими объектами и программным кодом [20].

Для удобства и повышения качества пользовательского опыта маршрутизация пользователя по графическому интерфейсу приложения реализована в качестве бокового всплывающего меню с помощью элемента *MasterDetailPage*.

Для визуализации поиска Bluetooth устройств помощью элемента *ContentPage* разработана страница “Поиск”. Страница содержит элемент *TabbedPage*, который является родительским элементом для страниц *Low Energy* и *Classic,* которые визуализируют списки имен активных устройств в эфире BLE и Bluetooth Classic соответственно. Контейнером имен устройств является элемент *ListView.* При нажатии на представление найденного устройства, оно будет добавлено на панель управления.

Для визуализации подключенных и готовых к работе устройств с помощью элемента *ContentPage* разработана страница *“Панель управления”*. Данная страница визуализирует информацию о каждом из подключенных датчиков, контейнером для которой является элемент *ListView.* При нажатии

на представление в списке конкретного датчика откроется страница *“Запуск исследования”*, которая описана ниже.

Разработанная помощью элемента *ContentPage* страница *“Запуск исследования”* предназначена для получения от пользователя параметров старта длительного исследования. Для визуализации ввода и редактирования параметров старта длительного исследования используется свой экземпляр элемента *Entry* для каждого параметра.

Для визуализации списка результатов измерений разработана страница “*Измерения*” помощью элемента *ContentPage*. Список результатов измерений реализован с помощью контейнера *ListView,* каждый элемент которого кратко описывает результат измерения. При нажатии на конкретный результат измерения будет осуществлен переход на страницу *“Просмотр измерения”*, которая описана ниже. В шапке страницы “*Измерения*” с помощью элемента *ToolbarItems* реализованы кнопки “Выделить”, “Отправить по почте”, “Удалить”. Для отправки измерений по почте удаления пользователь должен выбрать желаемые элементы и нажать кнопку “Отправить по почте” или “Удалить”.

Для визуализации подробного результата длительного измерения разработана помощью элемента *ContentPage* страница *“Просмотр измерения”*. На данной странице пользователь может видеть график эхограммы или динамограммы, а также побочные параметры исследования. Для визуализации эхограммы и динамограммы используется компонент *SKCanvasView*.

6.7 Приемочное тестирование

С целью проверки разработанного программного обеспечения на предмет готовности к эксплуатации было выполнено приемочное тестирование основной функциональности. Был составлен план приемочного тестирования, охватывающий основной функционал приложения необходимый для работы. Составленный план приемочного тестирования представлен в таблице 6.14.

Таблица 6.14 – План приемочного тестирования

Тест	Ожидаемый результат
Запуск приложения	Запустится приложение с развернутым боковым меню
Сканирование Bluetooth эфира	Приложение просканирует Bluetooth эфир и отобразит на интерфейсе доступные устройства
Подключение к датчикам	Приложение должно установить соединение с датчиками соединение, визуализировав подключенные устройства на странице *“Панель управления”*
Запуск длительного исследования	Приложение должно отобразить окно запуска измерения для ввода пользователем параметров исследования и запустить на выбранном датчике исследование.
Визуализация результата длительного исследования	Приложение должно отобразить на графическом интерфейсе подробную информацию о результатах длительного исследования.
Отправка результатов длительных исследований по почте	Приложение должно отобразить на графическом интерфейсе список результатов длительных исследований, выбрав желаемые из которых, пользователь сможет отправить их по почте.

Для реализации тестового случая “Запуск приложения” необходимо запустить приложение. На рисунке 6.10 представлена демонстрация работы приложения в тестовом случае “Запуск приложения”.

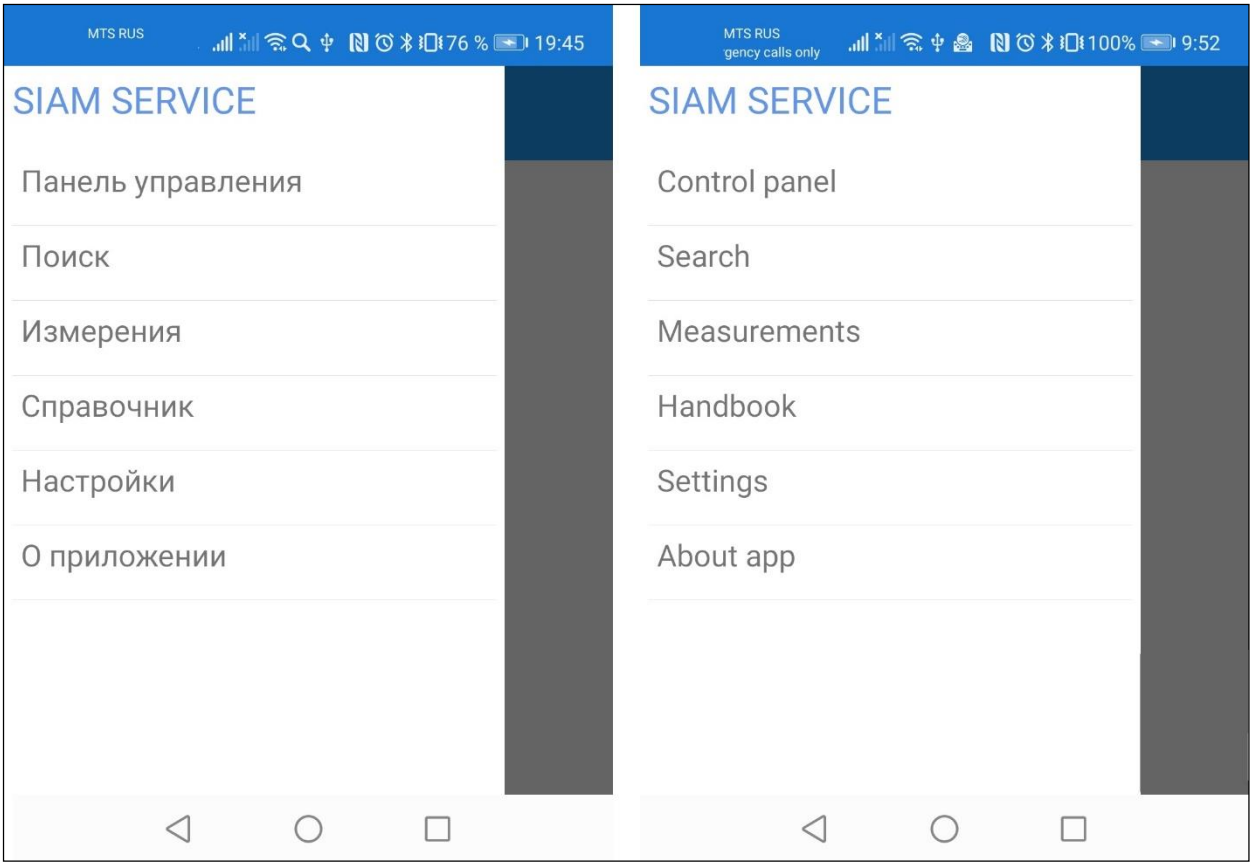

Рисунок 6.10 – Тестовый случай “Запуск приложения”

Приложение успешно запустилось и предоставило пользователю меню для перехода к следующему действию. Тест “Запуск приложения” признается пройденным.

Для реализации тестового случая “Сканирование Bluetooth эфира” необходимо перейти в главном меню по пункту “Поиск”. Далее открывается страница “Поиск”. На рисунке 6.11 представлена демонстрация работы приложения в тестовом случае “Сканирование Bluetooth эфира”.

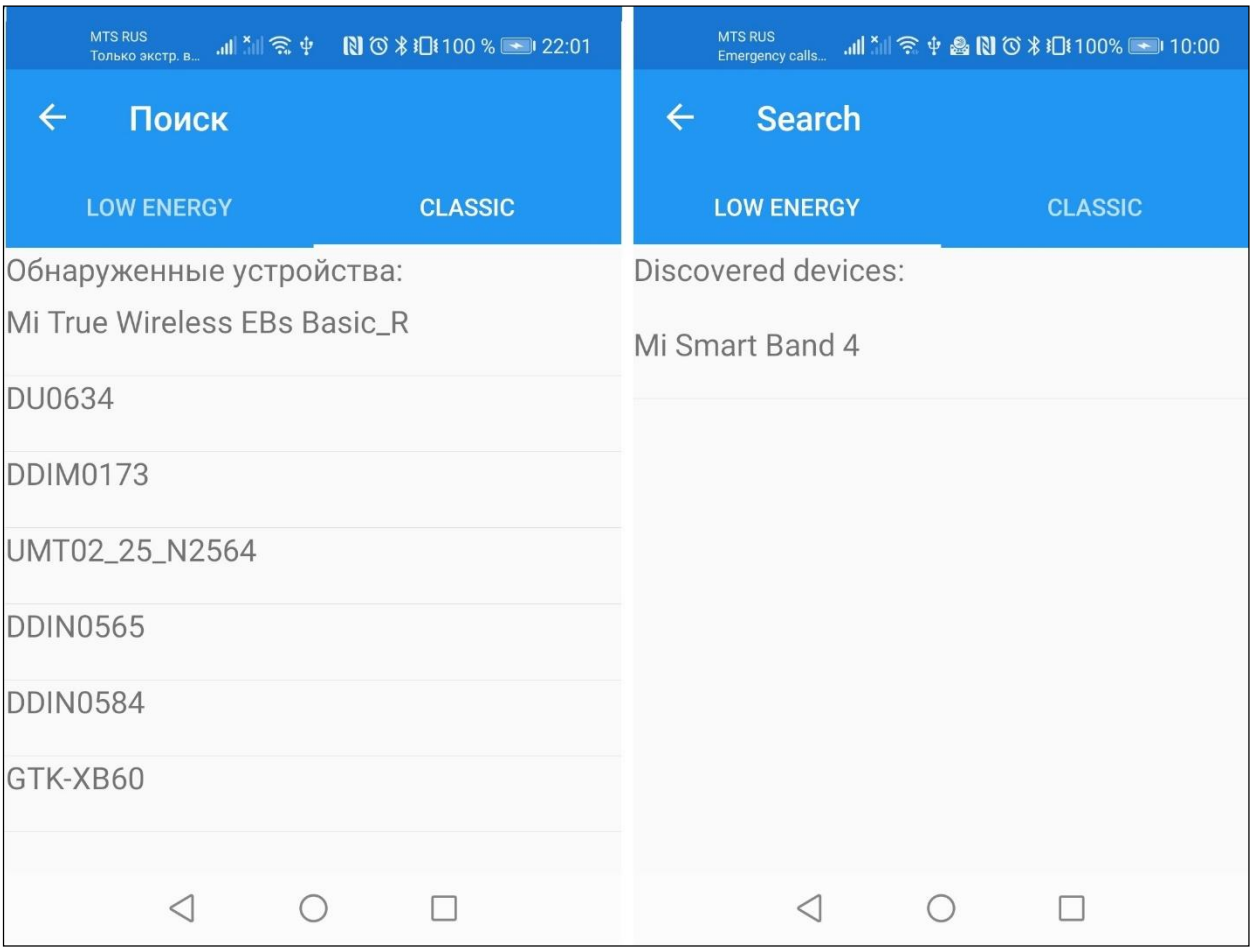

Рисунок 6.11 – Тестовый случай “Сканирование Bluetooth эфира”

На странице “Поиск” отображаются обнаруженные устройства в эфире BLE и эфире Bluetooth Classic. Тест “Сканирование Bluetooth эфира” признается пройденным.

Для реализации тестового случая “Подключение к датчикам” на странице “Поиск” необходимо выбрать желаемое для подключения устройство. На рисунке 6.12 представлена демонстрация работы приложения в тестовом случае “Подключение к датчикам”.

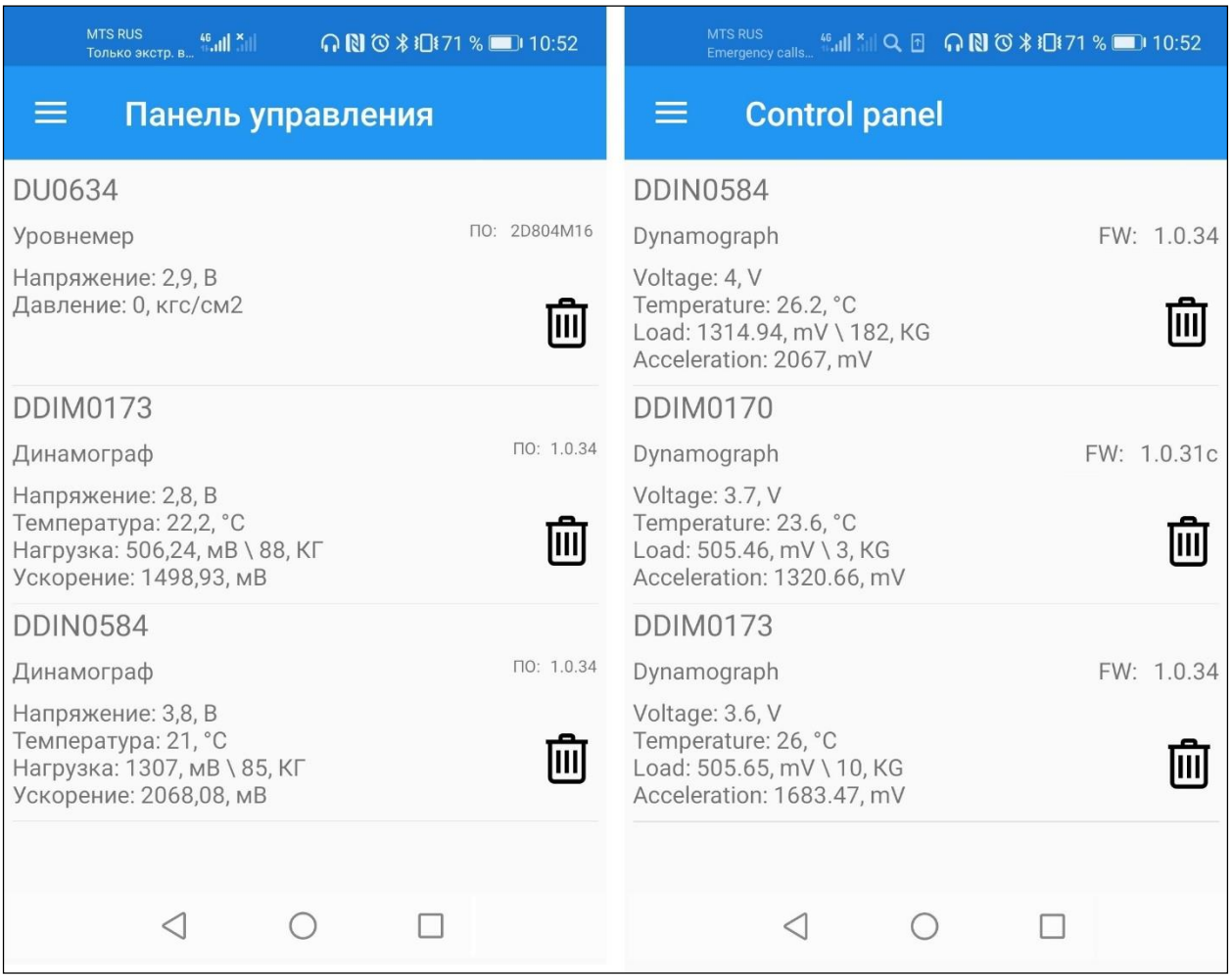

Рисунок 6.12 – Тестовый случай “Подключение к датчикам”

После выбора устройств на странице “Поиск” приложение автоматически переходит на страницу “Панель управления”, где визуализируются подключенные устройства, а также их текущие параметры. Тест “Подключение к датчикам” признается пройденным.

Для реализации тестового случая “Запуск длительного исследования” необходимо на странице “Панель управления” выбрать путем нажатия желаемый для исследования датчик. На рисунке 6.13 представлена демонстрация работы приложения в тестовом случае “Запуск длительного исследования”.

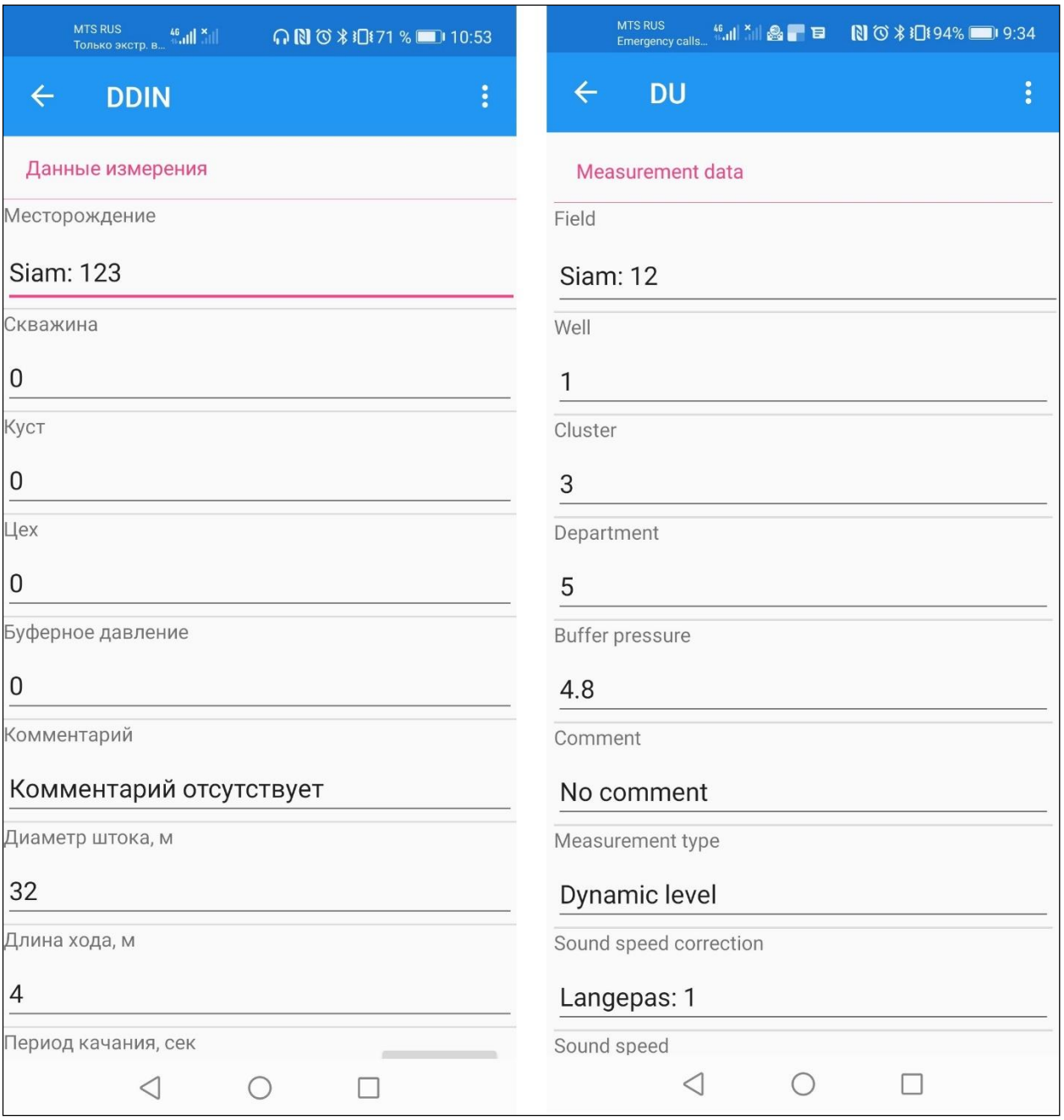

Рисунок 6.13 – Тестовый случай “Запуск длительного исследования”

После выбора на странице “Панель управления” датчика для исследования приложение открывает страницу для заполнения необходимых параметров исследований, после чего запускается исследование. Тест “Запуск длительного исследования” признается пройденным.

Для реализации тестового случая “Визуализация результата длительного исследования” нужно дождаться окончания длительного исследования. На рисунке 6.14 представлена демонстрация работы

приложения в тестовом случае “Визуализация результата длительного исследования”.

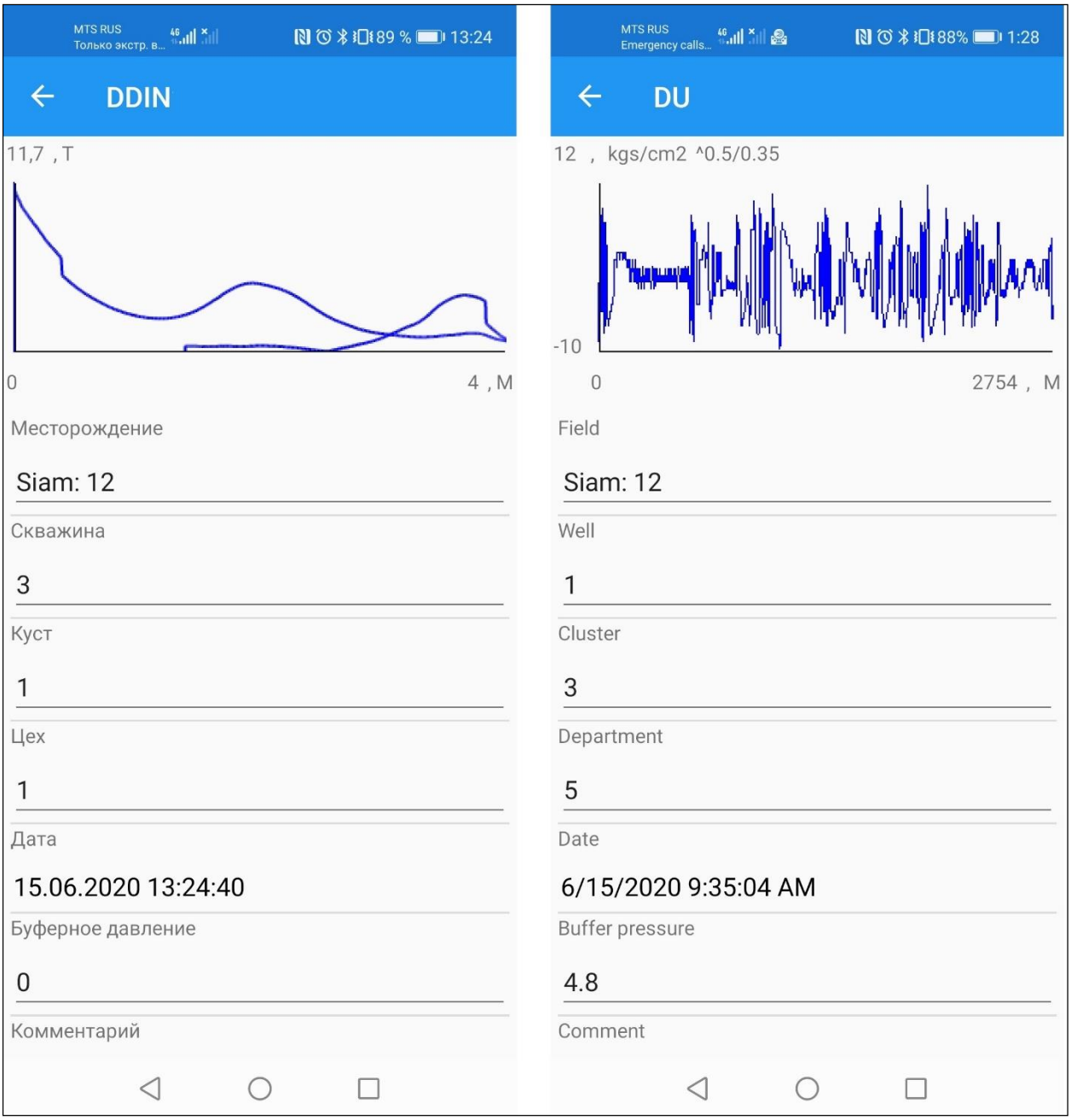

Рисунок 6.14 – Тестовый случай “Визуализация результата длительного исследования”

После окончания длительного исследования приложение автоматически вывело на экран смартфона страницу, с визуализированным результатом исследования. Тест “Визуализация результата длительного исследования” признается пройденным.

Для реализации тестового случая “Отправка результатов длительных исследований по почте” необходимо в главном меню перейти на страницу “Измерения”. На рисунке 6.15 представлена демонстрация работы приложения в тестовом случае “Отправка результатов длительных исследований по почте”.

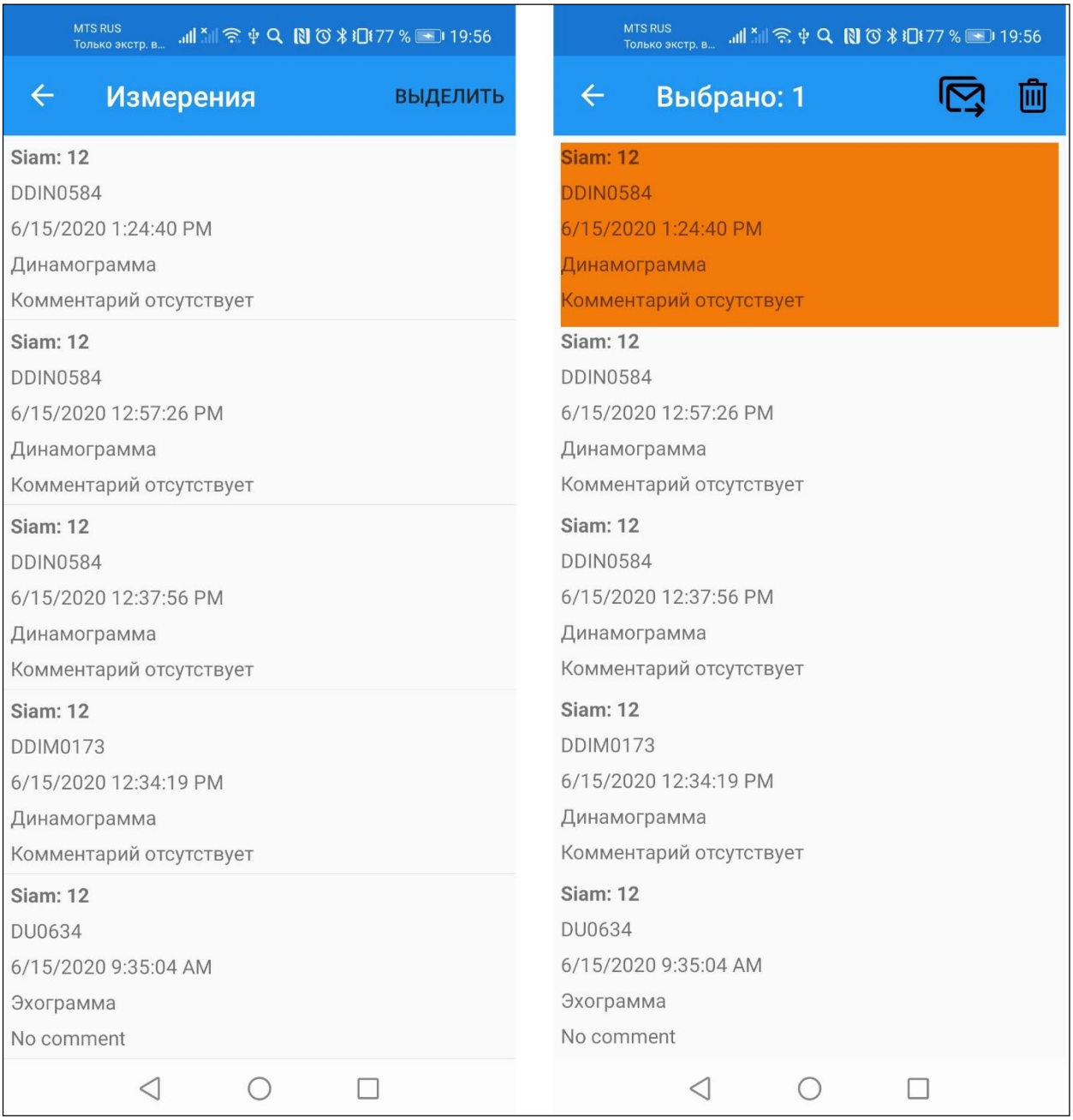

Рисунок 6.15 – Тестовый случай “Отправка результатов длительных исследований по почте”.

После перехода на страницу “Измерения” потребовалось с выбрать желаемые для отправки измерения и нажать кнопку “Отправить измерения по

почте”. После чего сконвертированный отчет измерения был отправлен по электронной почте. Тест “Отправка результатов длительных исследований по почте” признается пройденным.

В таблице 6.15 представлены результаты приемочного тестирования.

Таблица 6.15 – Результаты приемочного тестирования

Тест	Статус
Запуск приложения	Успешно пройден
Сканирование Bluetooth эфира	Успешно пройден
Подключение к датчикам	Успешно пройден
Запуск длительного исследования	Успешно пройден
Визуализация результата длительного исследования	Успешно пройден
Отправка результатов длительных исследований по почте	Успешно пройден

Заключение

В ходе выполнения магистерской диссертации было произведено исследование, включающая изучение датчиков компании “СИАМ”, изучение протоколов и технологий обмена данными с датчиками. Был произведен обзор технологий для разработки кроссплатформенного программного обеспечения, которое необходимо было разработать для управляющего блока стационарного комплекса контроля скважин ШГНУ. На основании обзора технологий для разработки программного обеспечения был произведен подбор наиболее подходящей технологии для разработки.

Процесс реализации программного обеспечения был декомпозирован на программные модули, которые были реализованы:

1 модуль сканирования Bluetooth эфира;
2 модуль взаимодействия с устройствами по Bluetooth;
3 модуль внедрения зависимостей;
4 модуль взаимодействия с датчиками;
5 модуль работы с данными;
6 модуль графического интерфейса приложения.

Было выполнено приемочное тестирование разработанного программного обеспечения.

Результатом работы является программное обеспечение верхнего уровня, позволяет в реальном времени отслеживать состояние датчиков стационарного комплекса контроля скважин. Осуществлять управление датчиками, путем запуска исследований. Загружать из датчиков данные результатов исследований для предоставления их для дальнейшего анализа путем отправки на почту. Разработанное программное обеспечение не привязано к конкретной платформе управляющего блока, что позволяет легче в дальнейшем расширять и масштабировать приложение.

Обозначения и сокращения

В настоящей магистерской диссертации применяются обозначения и сокращения:

ШГНУ – Шланговая глубинная насосная установка;

ДДИН – Датчик динамометрирования накладной;

ДДИМ – Датчик динамометрирования межтраверсный;

ДУ – Датчик уровня;

ПО – Программное обеспечение;

BLE – Bluetooth Low Energy;

MVVM – Model-View-ViewModel.

Список использованных источников

1 СИАМ. Производитель оборудования для исследования скважин // Официальный сайт [Электронный ресурс]. – Режим доступа: http://www.siamoil.ru/ (дата обращения: 21.04.2020).

2 СИАМ. Уровнемеры // Официальный сайт [Электронный ресурс]. – Режим доступа: http://www.siamoil.ru/Levelmeters (дата обращения: 16.04.2020).

3 СИАМ. Динамографы // Официальный сайт [Электронный ресурс]. – Режим доступа: http://www.siamoil.ru/dynamometers (дата обращения: 21.04.2020).

4 Технология Bluetooth // Официальный сайт [Электронный ресурс]. – Режим доступа: https://www.bluetooth.com/ (дата обращения: 25.04.2020)

5 Протокол обмена данными приборов ТНПВО “СИАМ”. ООО ТНПВО “СИАМ”, редакция 3, 2007. – 7 с.

6 Ain Shams Engineering Journal. Taxonomy of Cross-Platform Mobile Applications Development Approaches // Официальный сайт [Электронный ресурс]. – Режим доступа: https://www.sciencedirect.com/science/article/pii/S2090447915001276#b0165 (дата обращения: 1.05.2020).

7 React Native // Официальный сайт [Электронный ресурс]. – Режим доступа: https://react-native.org/ (дата обращения: 5.05.2020).

8 Flutter // Официальный сайт [Электронный ресурс]. – Режим доступа: https://flutter.dev/ (дата обращения: 5.05.2020).

9 Ionic // Официальный сайт [Электронный ресурс]. – Режим доступа: https://ionicframework.com/ (дата обращения: 5.05.2020).

10 Документация по Xamarin // Официальный сайт [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin/ (дата обращения: 5.05.2020).

11 PhoneGap // Официальный сайт [Электронный ресурс]. – Режим доступа: https://phonegap.com/ (дата обращения: 15.05.2020).

12 Шаблоны корпоративного приложения с использованием Xamarin.Forms. Шаблон MVVM // Официальный сайт [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm (дата обращения: 16.05.2020).

13 Github. Bluetooth LE plugin for Xamarin // Официальный сайт [Электронный ресурс]. – Режим доступа: https://github.com/xabre/xamarin-bluetooth-le (дата обращения: 25.04.2020).

14 Github. 32feet.NET – Personal Area Networking for .NET // Официальный сайт [Электронный ресурс]. – Режим доступа: https://github.com/inthehand/32feet (дата обращения: 26.04.2020).

15 Autofac // Официальный сайт [Электронный ресурс]. – Режим доступа: https://autofac.org/ (дата обращения: 28.04.2020).

16 A. Hunt, D. Thomas. The Pragmatic Programmer: From Journeyman to Master / Hunt. A, D. Thomas.: Addison-Wesley Professional, 1999. – 352 с.

17 Э. Фримен, Э. Робсон. Head First. Паттерны проектирования / Э. Фримен, Э. Робсон.: Питер, 2020. – 651 с.

18 SQLite. // Официальный сайт [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/index.html (дата обращения: 29.05.2020).

19 Dapper ORM. // Официальный сайт [Электронный ресурс]. – Режим доступа: https://dapper-tutorial.net/step-by-step-tutorial (дата обращения: 2.06.2020).

20 Xamarin.Forms. Основы XAML // Официальный сайт [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/xaml/xaml-basics/ (дата обращения: 3.06.2020).