МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ РОССИИ)

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рыбинский государственный авиационный технический университет
имени П. А. Соловьева»
(РГАТУ имени П. А. Соловьева)

Институт информационных технологий и систем управления
Кафедра вычислительных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Дипломный проект

Разработка программы для обработки физиологических данных о сне человека с целью определения оптимального момента его пробуждения.

на соискание квалификации бакалавра
по направлению 09.03.01 Информатика и вычислительная техника

Пояснительная записка

Соискатель, студент группы ИВБ1–19 Чунаков М. А.

Руководитель ст. преп. кафедры ВС РГАТУ Каленов А. С.

Нормоконтролер канд. техн. наук, доцент Гусаров А. В.

К защите допустить
Зав. кафедрой канд. техн. наук, профессор Комаров В. М.

ВКР передана в ГЭК __ июня 2023 г.

Секретарь ГЭК канд. техн. наук, доцент каф. ВС Малышев Р. А.

Рыбинск 2023

Рыбинский государственный авиационный технический университет
имени П. А. Соловьева

УТВЕРЖДАЮ
Зав. кафедрой ВС
_____ Комаров В. М.
«____»_____2023 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

Студенту Чунакову Михаилу Алексеевичу

1 Тема дипломного проекта Разработка программы для обработки физиологических данных о сне человека с целью определения оптимального момента его пробуждения.

Утверждена приказом по университету от «___»_____ № _____

2 Срок сдачи студентом законченного проекта:

3 Назначение и актуальность темы:

В повседневной жизни человеку важно хорошо высыпаться. Для этого предпочтительно, чтобы он проснулся в так называемую фазу быстрого сна. Фаза быстрого сна характерна бурной деятельностью мозга, повышением артериального давления, температуры тела и ускорением сердечных сокращений. Разрабатываемая программа будет определять фазу сна на основе данных, присылаемых, с пульсометра на смартфон по *Bluetooth*. В существующих аналогах данной программы имеются следующие недостатки:

- Определение фазы сна по косвенным параметрам;
- Сложность настройки;
- Реализация в виде отдельного дорогостоящего устройства.

Для устранения отмеченных недостатков необходимо разработать программу для обработки физиологических данных о сне человека с целью определения оптимального момента пробуждения. Данная ВКР посвящена решению этой задачи, этим и обусловлена актуальность её темы.

4 Исходные данные к проекту:

4.1 Входные данные программы:

-информация с пульсометра (частота сердечных сокращений, уровень заряда пульсометра);

-заданное время пробуждения;

-текущее время на смартфоне.

4.2 Функции программного обеспечения:

- определение оптимального момента пробуждения;
- сбор физиологических данных о сне человека;

-подача сигнала на пробуждение;

-задание максимально раннего момента пробуждения;

-задание максимально позднего момента пробуждения;

- подключение пульсометра по *Bluetooth.*

4.3 Выходные данные программы:

-отчёт о качестве сна;

-звуковой и/или вибрационный сигнал будильника.

4.4 Требования к аппаратно-программному обеспечению:

смартфон системы Android (не ниже 6 версии) с поддержкой *Bluetooth.*

4.5 Среда разработки: *Android Studio Electric Eel*

5 Содержание основной части пояснительной записки (перечень подлежащих разработке вопросов)

Введение

1 Патентно-информационный поиск

2 Анализ технического задания

3 Техническое проектирование

3.1 Разработка алгоритмов

3.2 Декомпозиция программы.

3.2 Кодирование программы

3.3 Разработка контрольного примера

4 Экономическая часть (определение себестоимости и цены разработки)

Заключение

Список использованных источников

6 Перечень графического материала (не менее 6 листов формата А1):

1 Описание предметной области 1л.

2 Схема информационных потоков 1л.

3 Сравнения аналогов 1л.

4 Блок-схема взаимодействия процессов приложения 1л.

5 Статическая модель программы 1л.

6. Интерфейс программы. Контрольный пример. 1л.

7 Прочие требования: Демонстрация работающей программы при защите проекта

8 Консультанты по проекту (фамилия, имя, отчество)

Нормоконтроль: канд. техн. наук, доцент Гусаров А. В.

Дата выдачи задания «____»_____2023 г.

Руководитель: ст. преп. кафедры ВС РГАТУ Каленов А. С.

Подпись руководителя _____

Подпись студента _____

Примечание – Задание прилагается к законченной выпускной квалификационной работе и вместе с выпускной квалификационной работой представляется в ГАК.

Приложение к заданию на дипломный проект

КАЛЕНДАРНЫЙ ПЛАН

работы студента Чунакова Михаила Алексеевича.

Тема дипломного проекта: Разработка программы для обработки физиологических данных о сне человека с целью определения оптимального момента его пробуждения.

Наименование этапов дипломного проектирования	Сроки выполнения этапов	Примечание
1. Патентно-информационный поиск	15.04.2023	
2. Анализ технического задания	05.05.2023	
3. Утверждение задания ВКР	20.05.2023	
4. Смотр ВКР	24.05.2023	
5. Декомпозиция программы	25.05.2023	
6. Кодирование программы	30.05.2023	
7. Разработка контрольного примера	31.05.2023	
8. Экономическая часть	01.06.2023	
9. Оформление ПЗ и документации	08.06.2023	
10. Сдача проекта руководителю	09.06.2023	
11. Нормоконтроль	19.06.2023	
12. Предварительная защита ВКР	20.06.2023	
13. Рецензирование	23.06.2023	
14. Защита ВКР	26.06.2023	

Студент-дипломник _____ Рук. проекта _____

Примечание – Календарный план работы над дипломным проектом разрабатывается до начала проектирования

ХОД ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Смотр ВКР _____

% выполнения, дата, подпись председателя комиссии, Ф. И. О. председателя комиссии

Предварительная защита _____

заключение, дата и подпись председателя комиссии, Ф. И. О. председателя комиссии

Защита проекта на заседании ГАК _____

дата и подпись зав. кафедрой ВС

Примечание – Протокол изменения задания прилагается к законченной выпускной квалификационной работе (ВКР) и вместе с ВКР представляется в ГАК..

Реферат

Выпускная квалификационная работа. Дипломный проект на тему: «Разработка приложения для обработки физиологических данных о сне человека с целью определение оптимального момента его пробуждения».

Пояснительная записка содержит 81 страницы, 26 рисунков, 6 таблиц, 17 источников, 5 приложений.

Цель работы – разработка программы, предназначенной для обработки физиологических данных о сне человека с целью определения оптимального момента его пробуждения.

В процессе работы была разработана модель обработки данных, произведено проектирование программы и осуществлено кодирование программы.

В результате выполнения работы была разработана документация на программное обеспечение.

Программа должна выводить сигнал для пробуждения пользователя и отправлять уведомления на смартфон о текущем пульсе пользователя.

Содержание

Введение

Сон человека состоит из двух фаз: быстрой фазы сна (REM) и медленной (NREM). Быстрая фаза сна получила своё название благодаря тому, что на этой стадии сна глазные яблоки человека быстро двигаются под закрытыми веками. REM это аббревиатура от Rapid Eye Movement, что в переводе означает «быстрое движение глаз». REM-сон играет важную роль в поддержке нормальной работы организма. В том числе, фаза быстрого сна отвечает за то, насколько человек высыпается. Быстрая фаза занимает около 25 % от всего времени сна, медленная – до 75 %.

В фазу медленного сна входят три стадии: NREM1 (дремота), NREM2 (поверхностный, легкий сон), NREM3 (глубокий сон), в фазу быстрого сна – одна REM. Медленный сон переходит в быстрый примерно через 60 – 90 минут после засыпания. В процессе ночного сна мы несколько раз проходим через все его этапы, включающие три стадии медленного сна, за которыми следует быстрый сон. С каждым новым циклом мы проводим все больше времени в REM-сне, причем его основная часть приходится на вторую половину ночи. Каждая из стадий сна по-своему необходима нашему организму.

Быстрый сон играет важную роль в формировании сновидений, памяти, обработке эмоций, правильном развитии мозга, подготовке к пробуждению. Быстрый сон, благодаря активации центральной нервной системы, может помочь нам подготовиться к пробуждению. Это объясняет, почему люди проводят все больше времени в фазе быстрого сна во второй части ночи и почему нам легче проснуться на этой стадии сна.

1 Патентно-информационный поиск.

На первом этапе дипломного проектирования производился поиск прототипов и аналогов разрабатываемой программы. Поиск производился в сети Интернет.

В результате произведённого поиска полных аналогов не было найдено, найдены отдельные устройства, поддерживающие аналогичные функции и программы, работающие на других принципах, но выполняющие похожие функции.

1.1 Устройства, реализующие функцию будильника с пульсометром.

1.1.1 Фитнес-браслеты фирмы *Fitbit*.

Фитнес-браслеты этой фирмы (а именно модели *Fitbit Charge HR, Fitbit Alta HR* и *Fitbit Blaze*) постоянно контролируют изменение сердечного ритма и частоту сердечных сокращений (ЧСС).

Устройства используют данные об изменении ЧСС и ваших движений, собранные в предыдущую ночь, чтобы оценить циклы сна с полученными метриками сегодняшним утром. Таким образом, трекер сравнивает и анализирует прошлые и нынешние результаты для полной информации о качестве вашего сна. *Fitbit* не претендует на распознавание стадий сна, гаджеты довольно точно обнаруживают, насколько чутко или крепко спит пользователь.

1.1.2 Фитнес-трекер фирмы *Misfit* модели *Shine 2*.

Misfit Shine 2 – это водонепроницаемый фитнес-трекер, ориентированный, как и большинство спортивных браслетов, на активный образ жизни. Однако гаджет предлагает довольно хорошие возможности отслеживания сна.

Shine 2 различает лёгкий и глубокий сон, предоставляет данные об общем количестве часов сна.

1.1.3 Фитнес-браслеты *Xiaomi Mi Band 2* и *Mi Band 3*.

Mi Band 2 и *Mi Band 3* определяют фазы сна, поскольку имеют пульсометры.

Функция умного будильника в браслетах отсутствует, но с помощью сторонних приложений её можно реализовать, но настройка сторонних приложений сложна. Из-за этого корректнее при анализе будет рассматривать сторонние приложения XSmart.

1.2 Приложения для отслеживания сна.

Несмотря на то, что приложения умного будильника и существуют, но они отслеживают сон на основе косвенных признаков, а именно:

- датчик движения;
- микрофон (шорох, разговоры, храп).

Эти способы определения продолжительности сна, но они не могут:

- определить фазу сна;
- начать будить пользователя в фазу быстрого сна (rem — «быстрое движение глаз»);
- не быть специализированными на небольшом числе моделей датчиков.

1.3 Анализ существующих вариантов.

В ходе анализа существующих вариантов находим плюсы и минусы существующих аналогов программы. Эти плюсы и минусы заносим в одну таблицу 1.1.

Таблица 1.1 – Результаты анализа существующих вариантов

Название варианта	Плюсы	Минусы
Fitbit	Удобное использование	Дороговизна
	Имеются другие функции	Отдельное устройство
		Не определяет фазы сна
Misfit Shine 2	Удобное использование	Дороговизна
	Имеются другие функции	Отдельное устройство
	Определяет фазы сна	
Приложения для отслежывание качества сна	Собирает и анализирует информацию о качестве сна	Не определяет фазы сна
	Существует на все системы смартфонов (*Android* и *IOS*)	Не все функции доступны бесплатно
Приложение Xsmart	Определяет фазы сна	Сложная настройка
	Множество дополнительных функций	Работает только с браслетом Mi Band 2
		Существует только для Android

2 Анализ технического задания

Проект выполняется в среде разработки *Android Studio Electric Eel* ввиду того, что в ней есть удобная функция преобразования программы в .App файл, с дальнейшей установкой приложения на смартфон.

Данная среда разработки имеет возможность проводить отладку на физическом смартфоне, а не только на эмулированном.

В среде разработки предусмотрены переменные для хранения: *Bluetooth GATT* сервисов, характеристик и дескрипторов, а также для *UUID* этих переменных. *UUID* – это универсальный уникальный индексатор предназначенный для возможности обращения к отдельным элементам профиля. Он представляет собой 16-битный и 128-битный код. Например, сервис обслуживания батареи имеет код *0x180F* в 16-битном формате, а в 128-битном формате 0000180F-0000-1000-8000-00805F9B34FB. *Bluetooth GATT* сервис это как папка с данными в ней, все характеристики и дескрипторы хранятся в соответствующих сервисах. Например, сервис сердечного ритма включает три характеристики (*0x180D*):

- обязательная характеристика частоты сердечных сокращений (*0x2a37*);
- опционная характеристика положения датчика тела (0x2a38);
- условная характеристика контрольной точки сердечного ритма (0x2a39).

Характеристика – это данные, которые посылает или получает BLE устройство. *BLE* это *Bluetooth Low Energy*, что означает *Bluetooth* который потребляет меньше энергии устройства. Дескриптор – это атрибут с дополнительной информацией о характеристике.

Создание будильника осуществляется при помощи механизмов предусмотренных средой разработки, и необходимость написания его не требуется.

Анализ технического задания показывают, что все требования обоснованы и могут быть выполнены.

Для определения входных и выходных данных проанализируем представление ПО в виде чёрной сферы, изображенное на рисунке 2.1.

Рисунок 2.1 – Представление ПО в виде чёрной сферы

Входными данными для программы является:

- пульс с пульсометра. Для отслеживания фазы cnal

- заряд батареи пульсометра. Для того чтобы предупредить пользователя, когда заряд батареи пульсометра заканчивается;

- текущее время на смартфоне. Для правильного срабатывания будильника.

- время, на которое поставить будильник;

- максимально возможное время отклонения от времени срабатывания будильника. Время насколько раньше будильник может сработать.

Выходными данными для программы является:

- сигнал сработавшего будильника. Вибрация и мелодия звонка;

- отчёт о качестве сна. Отчёт содержит продолжительность сна, и фаза сна, в которой Вы проснулись.

3 Техническое проектирование

Для начала технического проектирования создадим *UseCase*-диаграмму. Эта диаграмма помогает определить функционал программы, элементы программы и с чем эти элементы будут взаимодействовать, а также помогает определить события, которые будут выполняться в синхронном или асинхронном режиме. Эта диаграмма изображена на рисунке 3.1.

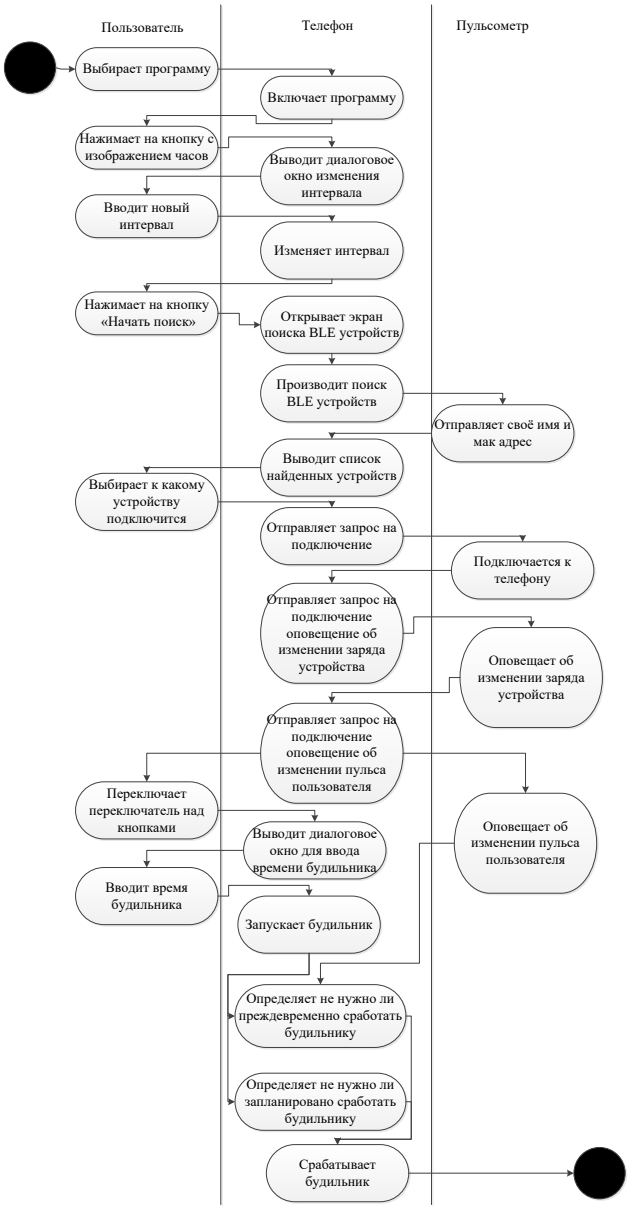

Рисунок 3.1 – *UseCase*-диаграмма

3.1 Декомпозиция программы

Декомпозиция программы выполняется для определения данных программы и упрощения написания алгоритмов программы.

В первом этапе декомпозиции программа представляется в виде чёрной сферы, и определяются входные и выходные данные. Результат первого этапа декомпозиции изображён на рисунке 3.2.

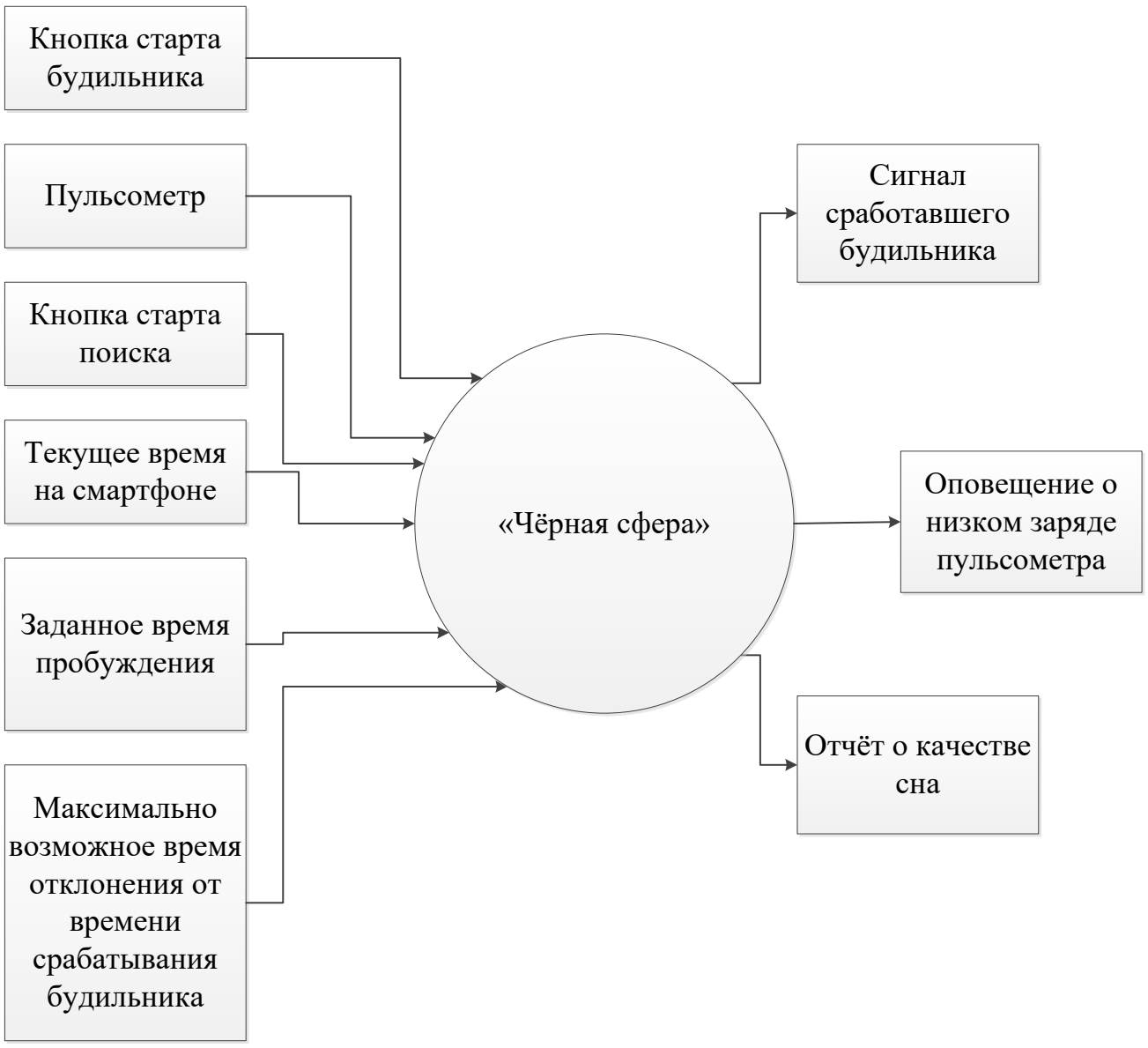

Рисунок 3.2. – Результат первого этапа декомпозиции

Следующим этапом нудно разделить задачу всей программы на несколько подзадач (в данном случае на две подзадачи) и подписать какие данные передаются между этими подзадачами.

Результат второго этапа декомпозиции программы изображён на рисунке 3.3.

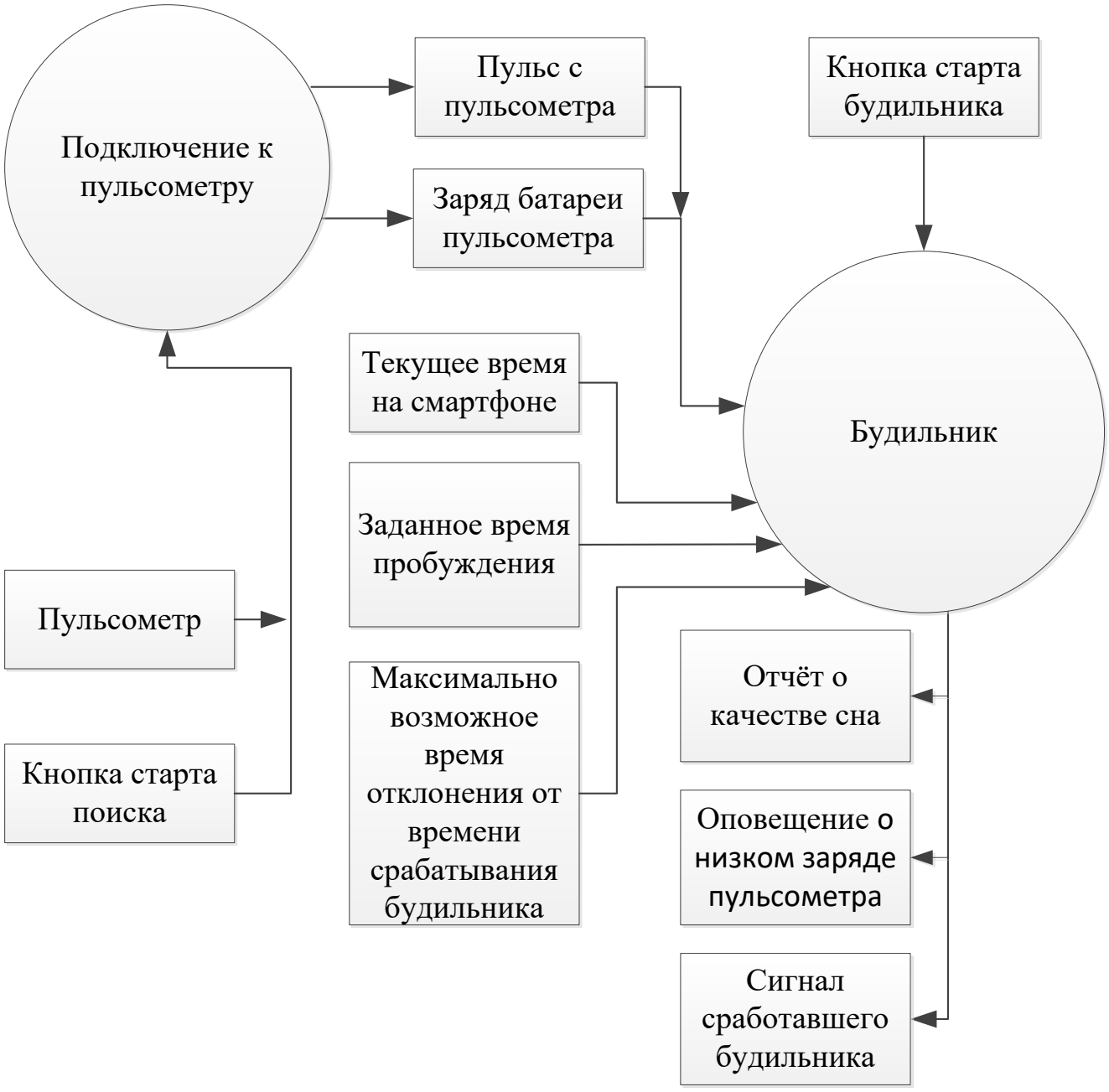

Рисунок 3.3 – Результат второго этапа декомпозиции программы

Следующий этап декомпозиции – это разделить получившиеся подзадачи на ещё более мелкие подзадачи.

Результат третьего этапа декомпозиции изображён на рисунке 3.4.

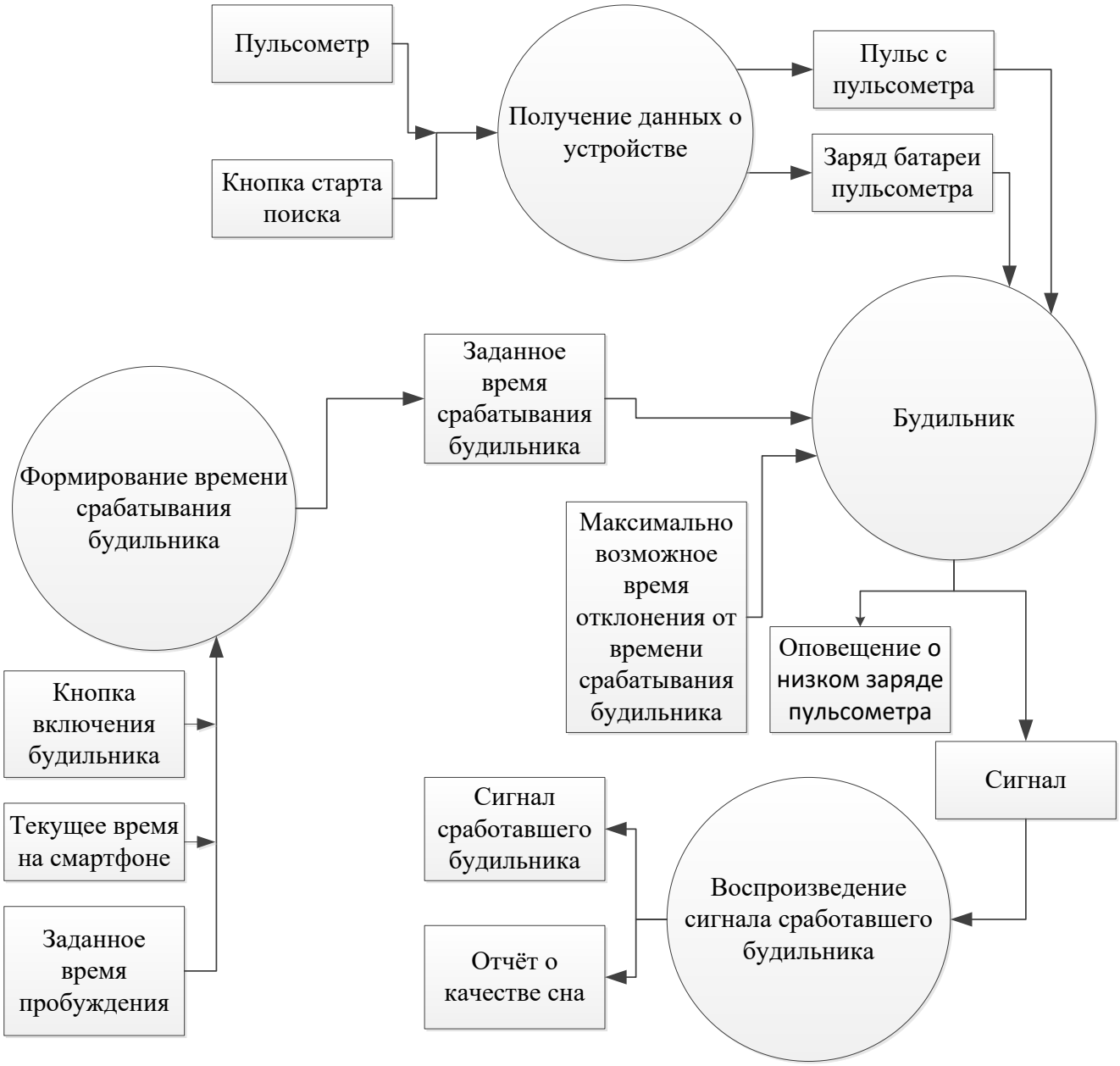

Рисунок 3.4 – Результат третьего этапа декомпозиции программы

Последующие этапы выполняются аналогично первым двум. Результат четвёртого этапа декомпозиции изображён на рисунке 3.5.

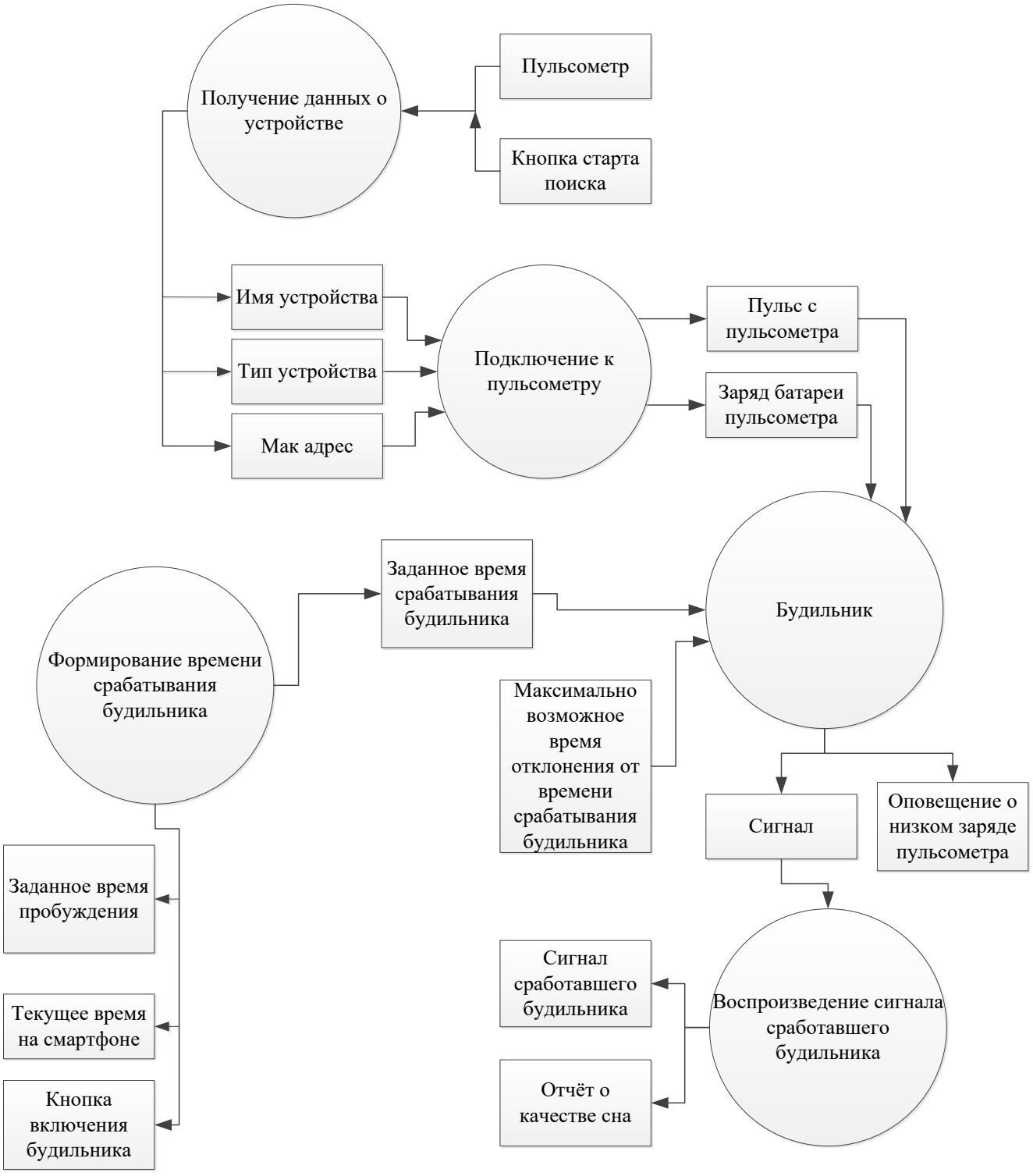

Рисунок 3.5 – Результат четвёртого этапа декомпозиции программы

Результат пятого этапа декомпозиции изображён на рисунке 3.6.

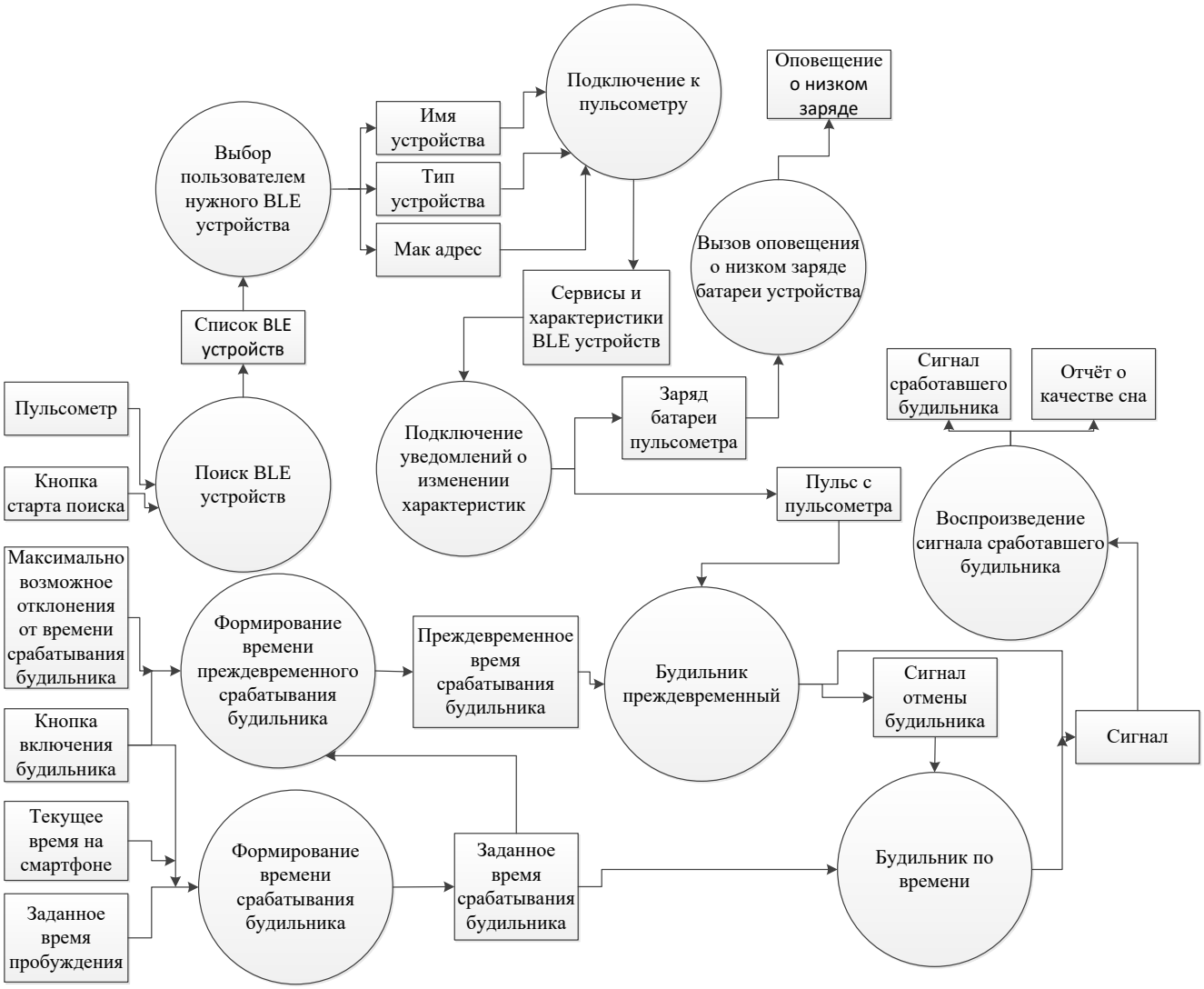

Рисунок 3.6 – Результат пятого этапа декомпозиции программы

Подзадачи после пятого этапа декомпозиции являются простыми и не требуют дальнейшего упрощения для кодирования. В связи с этим прекращаем выполнение декомпозиции и начинаем разработку алгоритмов программы.

3.2 Разработка алгоритмов

Разработка алгоритмов программы производится с целью определения количества основных функций и определения основного порядка работы программы.

Общий алгоритм программы представлен на рисунке 3.6.

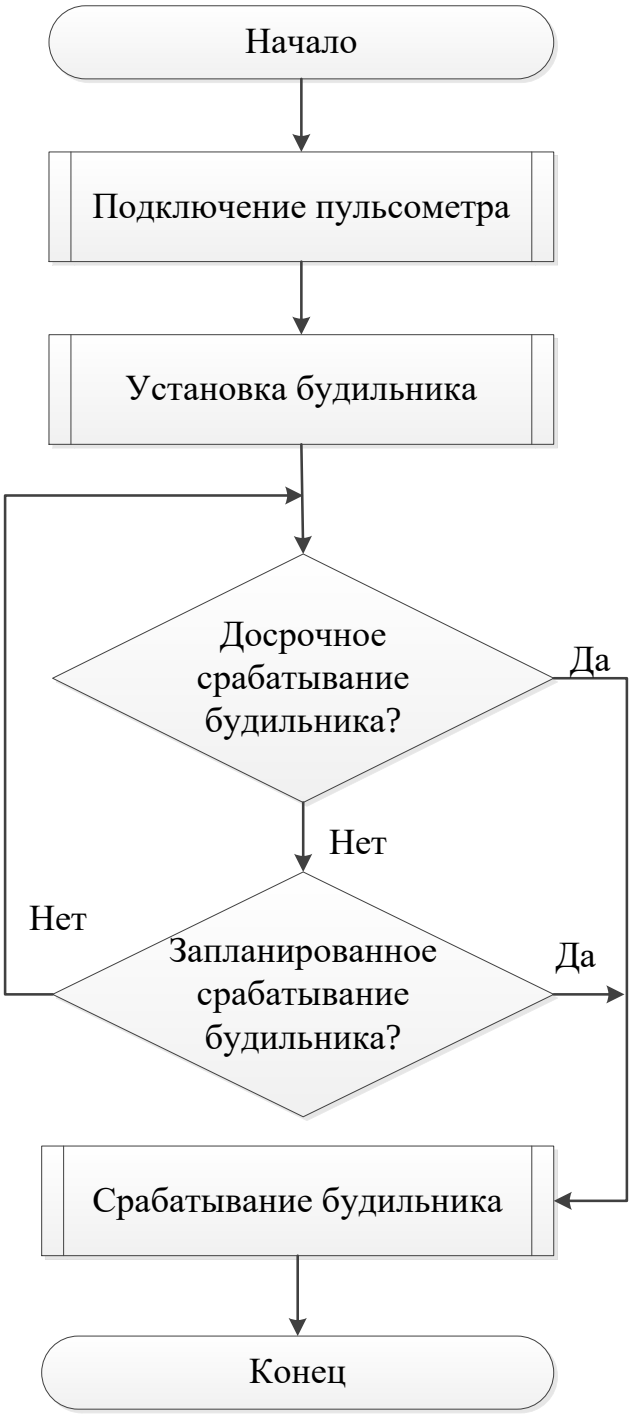

Рисунок 3.6 – Общий алгоритм программы

Алгоритм подключения к пульсометру представлен на рисунке 3.8. После подключения к BLE устройству сразу же включим оповещение об изменении значения характеристик.

Рисунок 3.8 – Алгоритм подключения к пульсометру

Алгоритм включения оповещения об изменении заряда устройства изображён на рисунке 3.9. Этот алгоритм будет работать только тогда, когда Bluetooth активен, и имеется *Bluuetooth GATT* подключение.

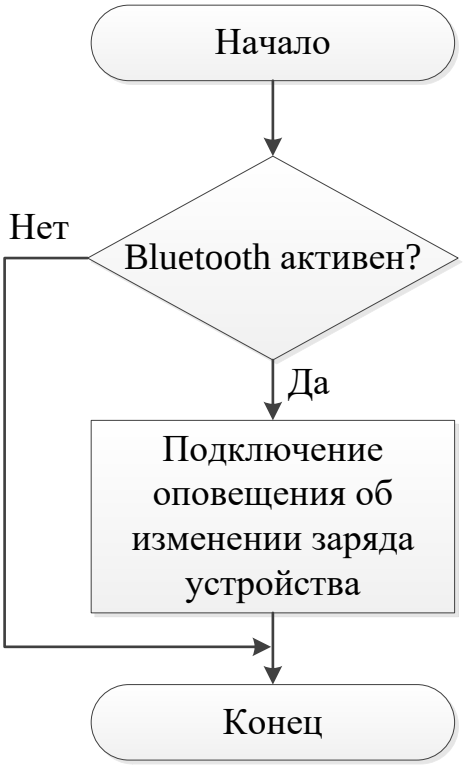

Рисунок 3.9 – Алгоритм включения оповещения об изменении заряда устройства

Алгоритм включения оповещения об изменении пульса пользователя изображён на рисунке 3.10. Этот алгоритм будет работать только тогда, когда Bluetooth активен, и имеется *Bluuetooth GATT* подключение. Из-за особенностей характеристики пульса в сервисе пульсометра этот алгоритм будет отправить пульсометру подтверждающий дескриптор.

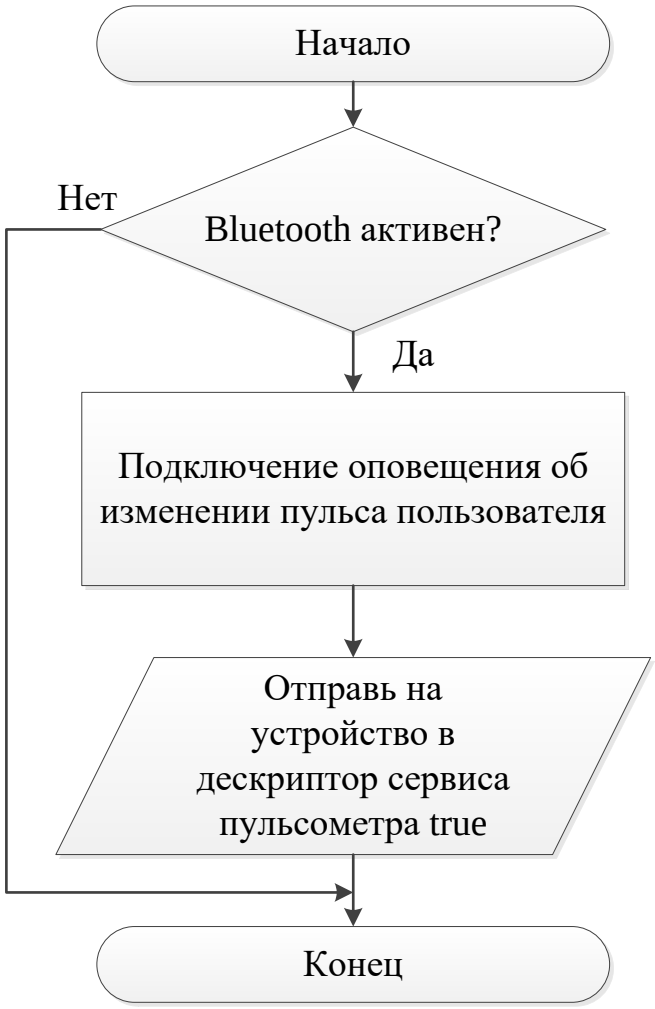

Рисунок 3.10 – Алгоритм включения оповещения об изменении пульса пользователя

Алгоритм установки будильника изображён на рисунке 3.11. Ввод времени срабатывания будильника осуществляется с помощью специального диалогового окна.

Рисунок 3.11 – Алгоритм установки будильника

Алгоритм установки будильника на сегодня изображён на рисунке 3.12. Этот алгоритм будет, вызывается для установки будильника на текущий день и переменная дня, на который установлен будильник, важна только для определения времени преждевременного срабатывания будильника.

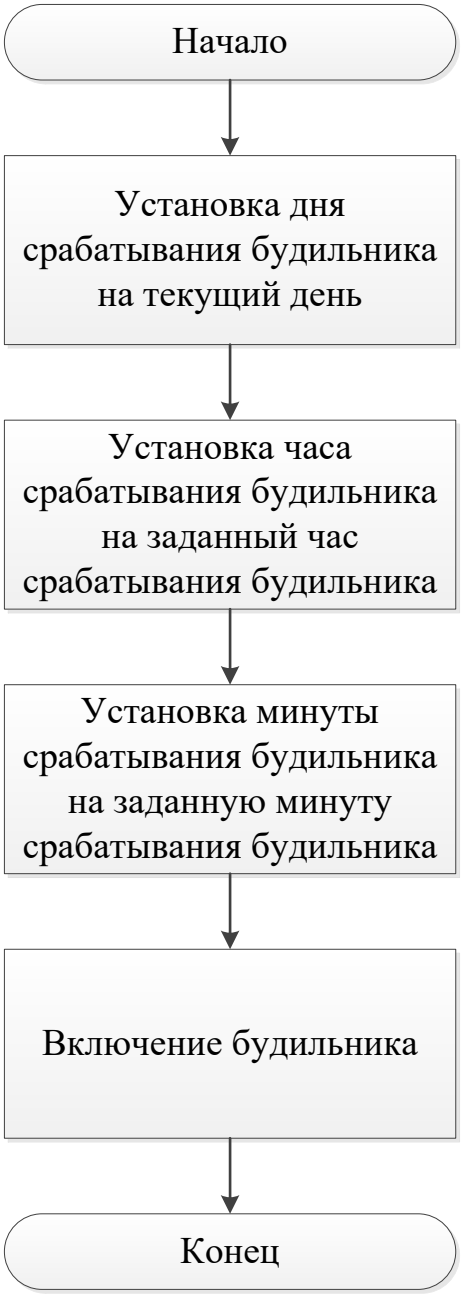

Рисунок 3.12 – Алгоритм установки будильника на сегодня

Алгоритм установки будильника на завтра изображён на рисунке 3.13. Важно то, что если сейчас последний день месяца, то переменная сама определит это, и при прибавлении 1 установит день срабатывания на первое число следующего месяца.

Рисунок 3.13 – Алгоритм установки будильника на завтра

Алгоритм определения минимального времени срабатывания будильника изображён на рисунке 3.14.

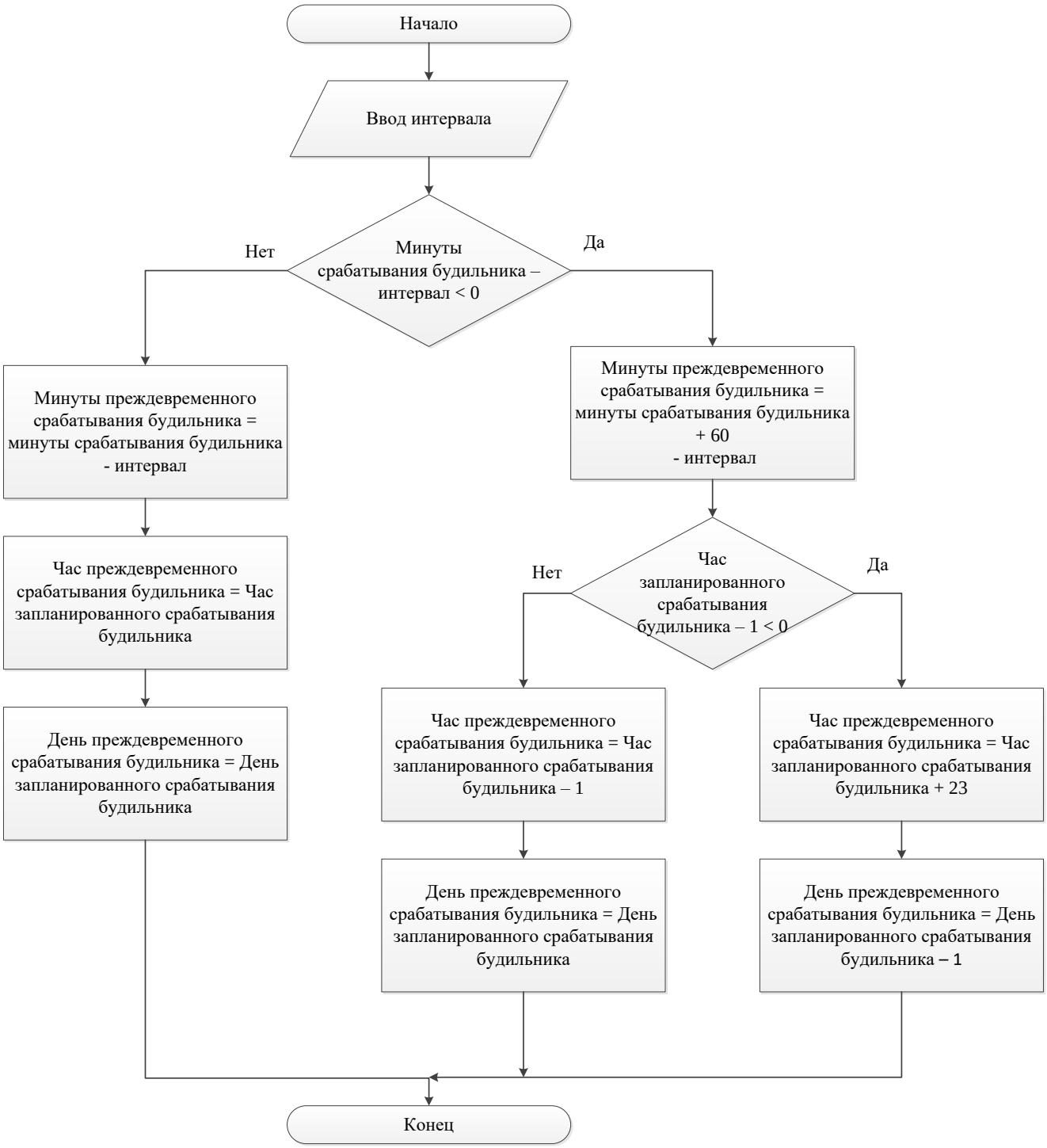

Рисунок 3.14 – Алгоритм определения минимального времени срабатывания будильника

Алгоритм срабатывания будильника изображён на рисунке 3.15. Вибрация от сработавшего будильника будет воспроизводиться до тех пор, пока вы не нажмёте на картинку.

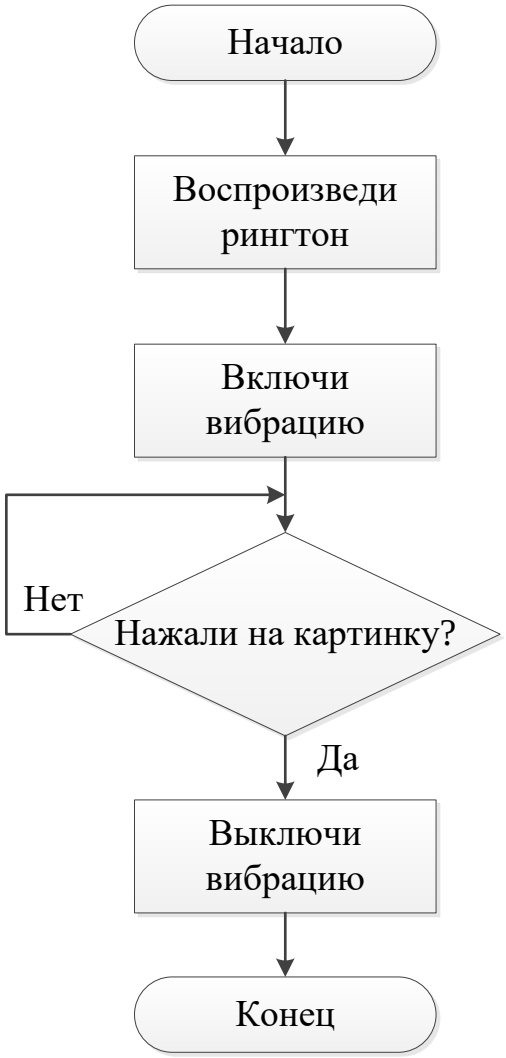

Рисунок 3.15 – Алгоритм срабатывания будильника

3.3 Кодирование программы

В ходе анализа технического задания средой разработки была выбрана *Android Studio*, в которой используется язык программирования JavaScript и язык программирования *HTML*.

HTML используется для стилизации элементов. С помощью него можно задать размеры элементов и их положение, цвет, толщину и размер шрифта, отступы. Резюмируя, HTML задает внешний вид приложения.

В JavaScript есть: *Bluetooth Manager* (используется для подключения Bluetooth устройств), *Alarm Manager* (даёт возможность управлять будильником) и *Notification Manager* (используется для отправки уведомлений пользователю).

При получении данных с пульсометра с помощью функций *Bluetooth Manager* Данные обрабатываются и демонстрируются пользователю с помощью функций *Notification Manager*. Данные от пульсометра приходят в переменную *characteristic* откуда и записываются в отдельные переменные.

Полученные данные о заряде устройства изображены на рисунке 3.16. На этом рисунке в строке *mValue* хранится значение заряда устройства в виде байтовой ячейки массива.

Рисунок 3.16 – Полученные данные о заряде устройства

Полученные данные о пульсе пользователя изображены на рисунке 3.17. На этом рисунке в строке *mValue* самой переменной *characteristic* хранится значение пульса пользователя (в ячейке массива под номером 1 в данном случае это 86). В строке *mValue mDescriptors* хранится подтверждающий включение оповещения дескриптор.

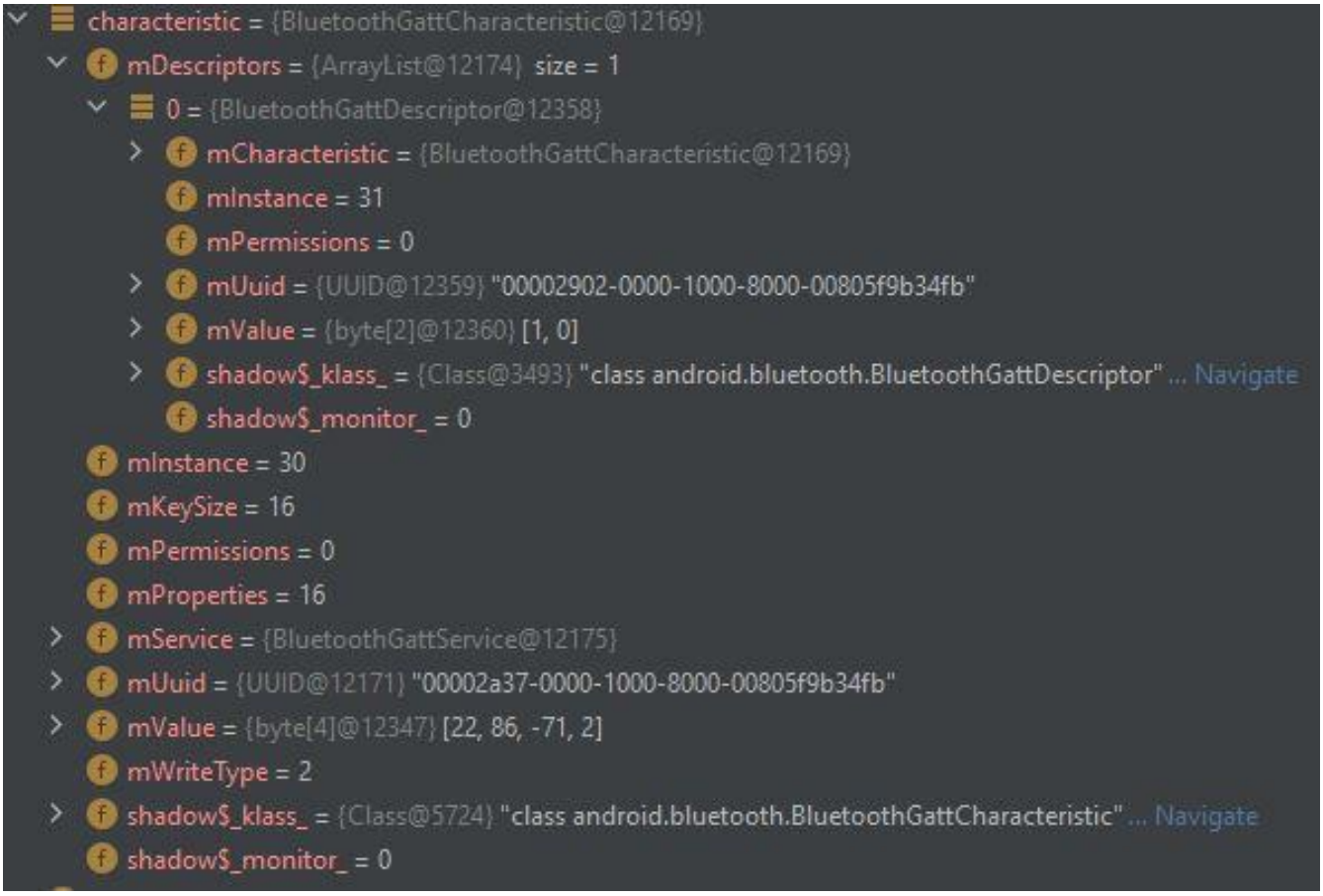

Рисунок 3.17 – Полученные данные о пульсе пользователя

Интерфейс программы кодируется на языке *HTML*. Ниже приведены примеры экранов с кодами этих экранов. На рисунке 3.18 приведён код экрана сработавшего будильника. На рисунке 3.19 изображён сам экран сработавшего будильника.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AlarmActivity">
    <TextView
        android:id="@+id/alarm_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Alarm!!!"
        android:textSize="72dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.681" />
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="254dp"
        android:layout_height="373dp"
        android:onClick="cansel"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/_2b26cbe731acaeeb40ca07625737b65"
        tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3.18 – Код экрана сработавшего будильника

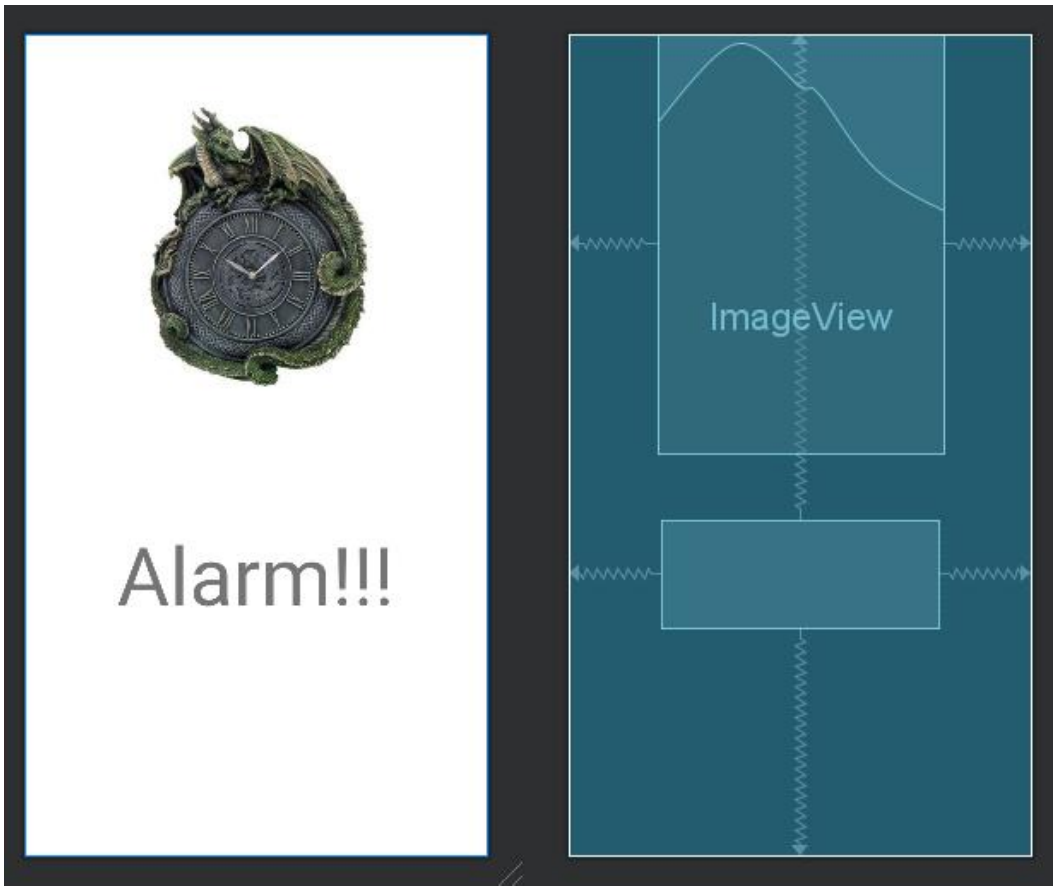

Рисунок 3.19 – Экран сработавшего будильника

Код экрана поиска BLE устройств изображён на рисунке 3.20. Изображение этого экрана представлено на рисунке 3.21.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context="com.example.budilnik.MainBluetooth">
    <TextView
        android:id="@+id/back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:onClick="oBacking"
        android:text="Назад"
        android:textStyle="italic" />
    <Button
        android:id="@+id/scan"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Начать поиск" />
    <ListView
        android:id="@+id/lelist"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Рисунок 3.20 – Код экрана поиска BLE устройств

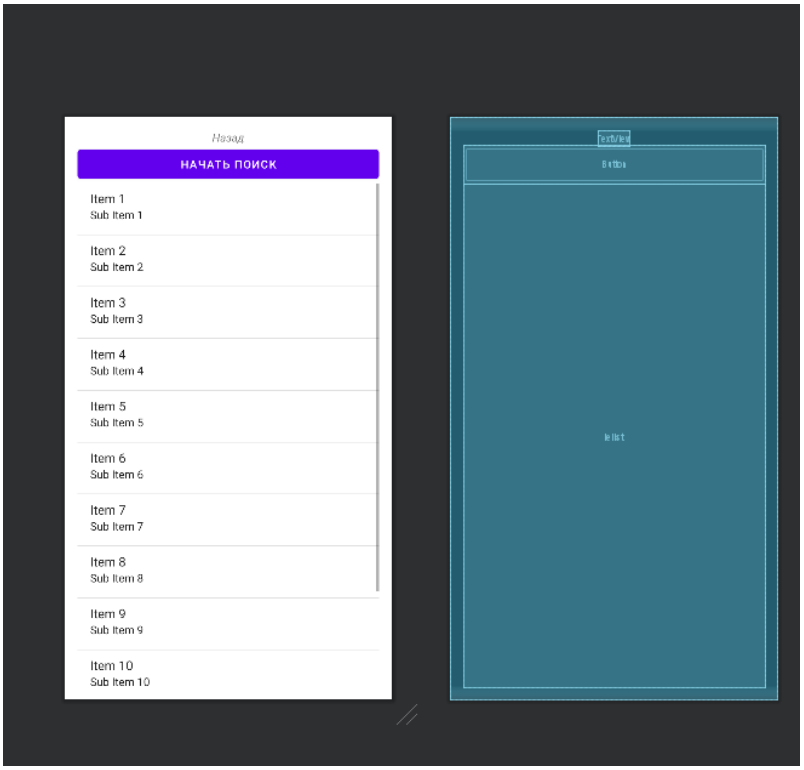

Рисунок 3.21 – Экран поиска BLE устройств

Код остальных экранов пишется аналогично. Изображение главного экрана представлено на рисунке 3.22.

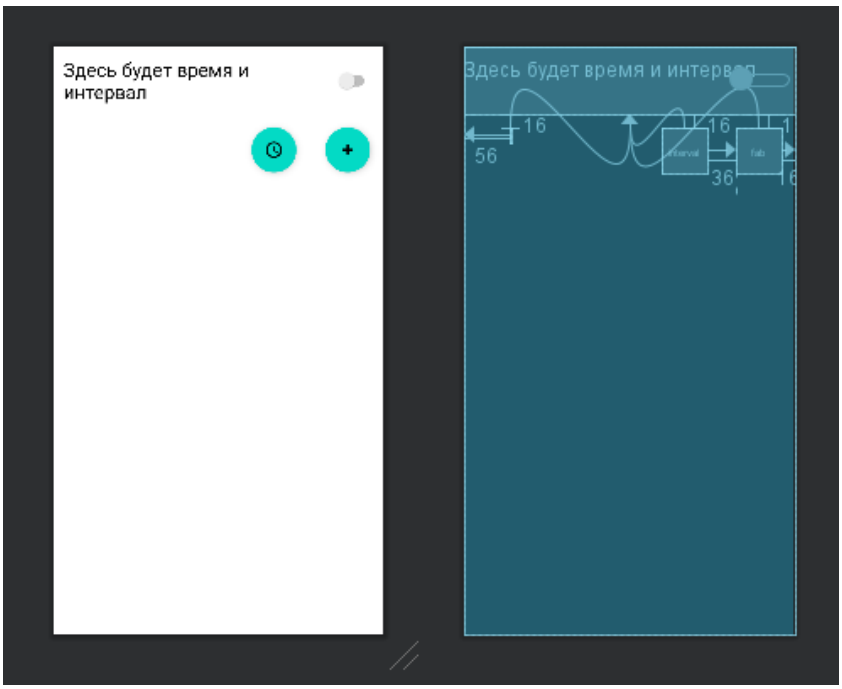

Рисунок 3.22 – Главный экран

Экран подключения к устройству изображен на рисунке 3.23.

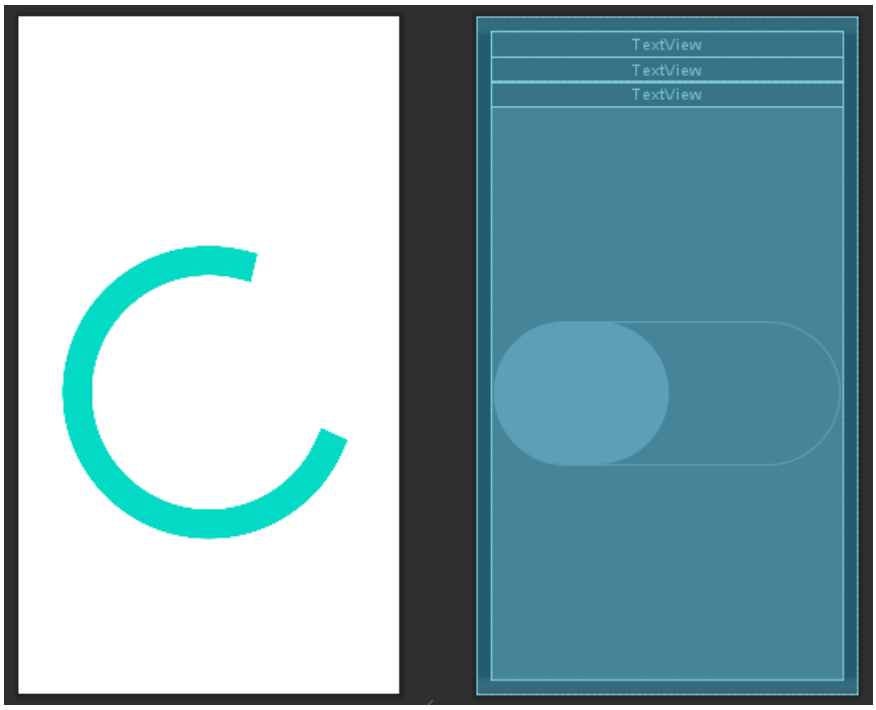

Рисунок 3.23 – Экран подключения к устройству

Частота сердечных сокращений здорового человека 20 – 25 лет в покое имеет значение от 60 – 72 ударов в минуту. При этом если человек стоит, то пульс имеет значение от 80 – 100. Во время разработки программы была собрана статистика о пульсе человека во сне. Для этого производилась запись значений пульса в отдельный файл. По этой статистике пульс человека при быстрой фазе сна от 60 до 80. Для предотвращения ложного срабатывания пульс должен быть в этом промежутке значений некоторое время.

3.4 Разработка контрольного примера

Для демонстрации возможностей были проведены испытания приложения. В течение месяца пользования этим приложением, оно будило меня раньше времени два раза. Из результатов этого эксперимента следует, что приложение работоспособно. Результаты эксперимента изображены на рисунке 3.23. и 3.24.

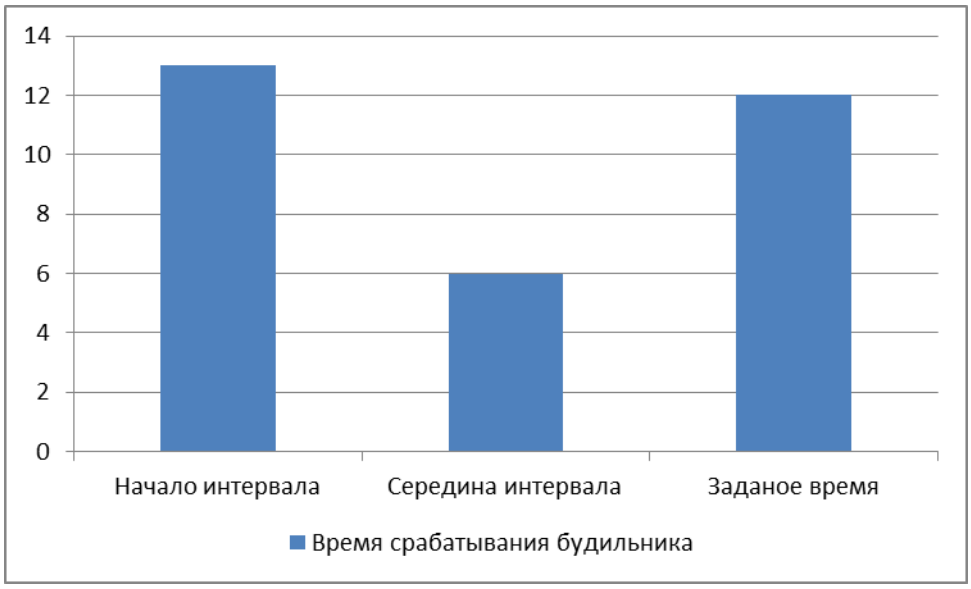

Рисунок 3.23 – В какое время срабатывал будильник во время экспериментов

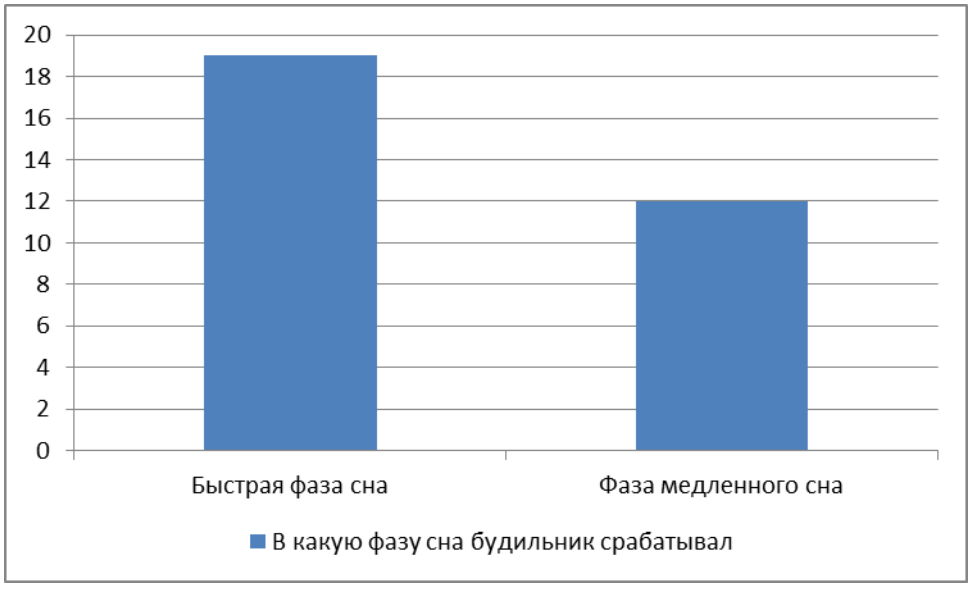

Рисунок 3.24 – В какую фазу срабатывал будильник

4 Экономическая часть

4.1 Применение и потенциальные пользователи

Каждому человеку важно высыпаться. Ведь от того выспался человек или нет зависит его работоспособность и внимательность в течение всего дня. Учёными доказано, что человек просыпаясь в фазу быстрого сна, чувствует себя выспавшимся, а если человек проснётся в фазу медленного сна, то он будет ощущать вялость. В связи с этим важно, если нужно проснуться к какому-то определённому времени, просыпаться в фазу быстрого сна. В этом и помогает разработанное приложение. Данное приложение могут использовать как студенты, так и уже отучившиеся люди.

4.2 Расчет затрат на разработку приложения

Для определения затрат на разрабатываемое приложение необходимо рассчитать и сложить основные статьи доходов, приведенные ниже:

- материальные затраты;
- основная заработная плата;
- дополнительная заработная плата;
- отчисления в страховые фонды;
- амортизационные расходы;
- эксплуатационные расходы;
- накладные расходы;
- прочие расходы.

В раздел материальные затраты входит стоимость материалов, использованных на создание приложения. На данный раздел будет приведена смета в таблице 4.1.

Таблица 4.1 – Материальные затраты

Вид материала	Количество, штук	Цена за единицу, руб.	Стоимость, руб.
Бумага формата А4	500	0,76	380
Картридж для принтера	1	1500	1500
Папка со скоросшивателем	1	100	100
Диск	1	30	30
Итого:			2010

Расходы на данный раздел составили 2010 р.

В данной ВКР была произведена разработка ПО. Следовательно, нужно опираться на заработную плату программиста, которая равна 23000 руб. в месяц. Произведем расчет основной заработной платы (ОЗП):

$$ОЗП = \sum_{i=1}^{n} P_{дi} \cdot T_i, \quad (4.1)$$

где n – количество работников, занимающихся разработкой ПО;

$P_{чi}$ – заработная плата в день определенного исполнителя;

T_i – количество дней для выполнения работы для определенного исполнителя;

Необходимо составить перечень работ, которые должны быть выполнены при разработке приложения. Перечень, приведённый в таблице 4.2, определяет трудоемкость работы.

Таблица 4.2 – Перечень трудоемкости работы

№ этапа	Этап разработки	Трудоемкость работы, дни
1	Информационно - патентный поиск	3
2	Анализ ТЗ	2
3	Анализ предметной области	2
4	Разработка модели обработки данных	2
5	Проектирования программы	4
6	Кодирование и тестирование программы	30
7	Разработка контрольного примера	1
8	Оформление ПЗ и графической части	14
9	Нормоконтроль и утверждение документации	4

Перечень, приведённый в таблице 4.3, определяет размер основной заработной платы, составляется на основании таблицы 4.2.

Таблица 4.3 – Основная заработная плата

№ этапа	Средняя дневная ставка, р.	Трудоемкость, дни	Сумма зарплаты, р.
1	1 152	3	3 456
2	1 152	2	2 304
3	1 152	2	2 304
4	1 152	2	2 304
5	1 152	4	4 608
6	1 152	30	34 560
7	1 152	1	1 152
8	1 152	14	16 128
9	1 152	4	4 608
Итого:	1 152	62	71424

Количество дней для выполнения работы равно 62 , дневная зарплата $P_{дi} = 1320$ руб./д. Подставим значение в формулу (4.1) и тогда получим значение основной заработной платы:

$$ОЗП = 1152 \cdot 62 = 71424.$$

Так как была определена основная заработная плата, можно вычислить дополнительную заработную плату, которая равняется 10 % от ОЗП. Дополнительная заработная плата составит:

$$ДЗП = 0.1 \cdot ОЗП. \quad (4.2)$$

$$ДЗП = 0.1 \cdot 71424 = 7142{,}4 \text{ р.}$$

Необходимо рассчитать отчисления в страховые фонды, равные 30 % от суммы общей заработной платы, то есть от суммы основной заработной платы и дополнительной заработной платы. Отчисления в страховые фонды составляет:

$$ОСФ = 0.3\ (ОЗП + ДЗП). \quad (4.3)$$

Подставив известные значения в формулу (4.3), получим что

$$ОСФ = 0.3\ (71\ 424 + 71\ 42{,}4) = 23\ 569{,}92 \text{ р.}$$

Далее определим амортизационные расходы. Для этого составим перечень оборудования, которое использовалось для разработки приложения. Для разработки данного приложения использовался ноутбук. Расчет амортизационных расходов приведен в таблице 4.4.

Таблица 4.4 – Амортизационные расходы

Использованное средство	Изначальная стоимость, р.	Срок гарантийного использования, месяцев	Время использования, месяцев	Итоговая сумма расходов, р.
Персональный компьютер	25000	12	2	4166,6

В итоге, исходя из таблицы 4.4, амортизационные расходы равны 4166,6 рублей.

Следующим шагом необходимо рассчитать эксплуатационные расходы. Поскольку использовался только ноутбук для расчета будет взяты затраты на электроэнергию. Для расчета примем время рабочего дня 8 часов, 21 рабочий день, цена одного киловатт-часа равна 4,55 руб., мощность ноутбука 65 Вт. Расчет произведен в таблице 4.5.

Таблица 4.5 – Эксплуатационные расходы

Использованное средство	Мощность, Вт.	Время использования, ч.	Потребляемая электроэнергия, кВт/ч	Итоговая сумма, р.
Ноутбук	65	336	21,84	99,37

Исходя из таблицы 4.5, эксплуатационные расходы обошлись в 99,37 рубля.

Для организаций, занимающихся разработкой новых проектов, статья накладные расходы равна 15 %, рассчитаем эти расходы по формуле (4.4).

$$НР = 0{,}15\ (ОЗП + ДЗП). \quad (4.4)$$

Подставляя необходимые значения в формулу (4.4) получим:

$$НР = 0{,}15\ (71\,424 + 71\,42{,}4) = 11\,784{,}96.$$

Затраты по статье накладные расходы составляют 11 784,96 рублей.

Прочие расходы рассчитаем по формуле (4.5).

$$ПР = 0{,}01 \cdot ОЗП. \quad (4.4)$$

Итак, подставив нужные значения в формулу (4.4) получим:

$$ПР = 0{,}01 \cdot 71\,424 = 714{,}24\ р.$$

Теперь, когда все статьи расходов подсчитаны, можем рассчитать себестоимость разработки приложения.

Себестоимость разработки приложения равна 120 911,49 рублей.

Далее рассчитаем цену разработки, для этого сначала рассчитаем прибыль с разработки равную 20 % от себестоимости.

Прибыль равна 24 182,3 рублей, тогда цена разработки равна 145 093,49 рубля.

Также необходимо рассчитать НДС (20 % от цены).

$$НДС = 0{,}2 \cdot 145\,093{,}49 = 29\,018{,}76\ р.$$

Итоговая цена разработки приложения с НДС равна 174 112,55 рублей.

Заключение

В ходе данной работы была произведена разработка программы для обработки физиологических данных о сне человека с целью определения оптимального момента его пробуждения. Для написания программы использовалось приложение Android Studio. В ходе разработки была произведена декомпозиция задачи и разработка алгоритмов программы. После разработки программы была произведена оценка себестоимости и цены разработки приложения.

Данное приложение превосходит аналоги в том, что оно:

- определяет фазу сна;
- просто в использовании;
- работает с дешёвым пульсометром.

При дальнейшем улучшении приложения можно создать его для IOS или оптимизировать расход заряда смартфона. Есть возможность добавить функцию отображения графика фаз сна, которые пользователь прошёл во время своего предыдущего сна.

Список использованных источников

1. Азбука Сна и Дыхания — статья о быстрой фазе сна [Электронный ресурс]: URL: https://sleep-resp.ru/statji-pro-son/zdorovy-son/faza-bystrogo-sna/ (дата обращения: 01.04.2023 г.).

2. *ismart wath.ru* — обзор фитнес-браслетов и смарт-часов с умным будильником и пульсометром [Электронный ресурс]: URL: https://ismartwatch.ru/8673-fitnes-braslety-s-umnym-budilnikom (дата обращения: 01.04.2023 г.).

3. *ixbt.com* — обзор *Xiaomi Mi Band* 2 [Электронный ресурс]: URL: https://www.ixbt.com/mobile/xiaomi-mi-band-2.shtml (дата обращения: 01.04.2023 г.).

3. *mobile-review.com* — обзор фитнес-трекера *Xiaomi Mi Band 2* [Электронный ресурс]: URL: https://mobile-review.com/articles/2017/xiaomi-mi-band-2.shtml (дата обращения: 01.04.2023 г.).

4. *fitbit* — официальный сайт *fitbit* [Электронный ресурс]: URL: https://www.fitbit.com/global/us/home (дата обращения: 01.04.2023 г.).

5. *mobile-review.com* — обзор умных часов *FitBit* [Электронный ресурс]: URL: https://mobile-review.com/articles/2018/fitbit-ionic.shtml (дата обращения: 01.04.2023 г.).

6. *medgadgets.ru* — Misfit Shine 2: обзор от *Medgadgets* [Электронный ресурс]: URL: https://medgadgets.ru/fitness/misfit-shine-2-obzor-ot-medgadgets.html (дата обращения: 01.04.2023 г.).

7. *mobile-review.com* — обзор трекера *Misfit Shine 2* [Электронный ресурс]: URL: https://mobile-review.com/articles/2016/misfit-shine-2.shtml (дата обращения: 01.04.2023 г.).

8. *droid-top* — Топ приложений «умных» будильников на телефон [Электронный ресурс]: URL: https://droid-top.ru/podborki/umniy-budilnik/ (дата обращения: 01.04.2023 г.).

9. *androfon.ru* — Лучшие умные будильники для андроид [Электронный ресурс]: URL: https://androfon.ru/article/podborka-umnyh-budilnikov (дата обращения: 01.04.2023 г.).

10. *Developer Android* — официальный сайт *Android Studio* [Электронный ресурс]: URL: https://developer.android.com/ (дата обращения: 02.04.2023 г.).

11. *Android-er* — блог с руководством как производить поиск Bluetooth устройств[Электронный ресурс]: URL: http://android-er.blogspot.com/2016/06/scan-specified-ble-devices-with.html (дата обращения: 16.04.2023 г.).

12. *Android-er* — блог с руководством как подключатся к BLE устройствам [Электронный ресурс]: URL: http://android-er.blogspot.com/2016/07/bluetooth-le-example-connect-to.html (дата обращения: 16.04.2023 г.).

13. *Android-er* — блог с руководством как определить что найденное устройство является BLE устройством [Электронный ресурс]: URL: http://android-er.blogspot.com/2016/06/bluetooth-le-gatt-example-step-by-step.html (дата обращения: 16.04.2023 г.).

14. *Android-er* — блог с руководством как включать оповещение о изменении значений характеристик подключённого BLE устройства [Электронный ресурс]: URL: http://android-er.blogspot.com/2016/06/bluetooth-le-gatt-example-to-link-with.html (дата обращения: 16.04.2023 г.).

15. *developer.alexanderklimov.ru* — блог с руководством как использовать уведомления смартфона [Электронный ресурс]: URL: https://developer.alexanderklimov.ru/android/notification.php (дата обращения: 19.04.2023 г.).

16. *c-sharpcorner.com* — руководство как использовать *Alarm Manager* в *Android Studio* [Электронный ресурс]: URL: https://www.c-sharpcorner.com/article/create-alarm-android-application/ (дата обращения: 25.04.2023 г.).

17. СТРОЙФОРА — тарифы ЖКХ [Электронный ресурс]: URL: https://stroyfora.ru/tariff/area-292c0a6a-47ce-435d-8a75-f80e9ce67fba/year-2023/type-16 (дата обращения: 09.06.2023 г.)

Приложение А

Код класса *MainActivity*

```
package com.example.budilnik;

import static com.example.budilnik.BluetoothLeService.heartRate;
import android.annotation.SuppressLint;
import android.app.AlarmManager;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.timepicker.MaterialTimePicker;
import com.google.android.material.timepicker.TimeFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Objects;

public class MainActivity extends AppCompatActivity   {
    public static int heartRate;
    SimpleDateFormat sdf = new SimpleDateFormat("dd LLLL HH:mm:ss", Locale.getDefault());
    public static Integer hour;//Переменная хранящая час запланированного срабатывания
    public static Integer min; //Переменная хранящая минуты запланированного срабатывания
    public static Integer day; //Переменная хранящая день запланированного срабатывания
    public static Integer hour2;//Переменная хранящая час преждевременного срабатывания
    public static Integer min2;//Переменная хранящая минуты преждевременного срабатывания
    public static Integer day2;//Переменная хранящая день преждевременного срабатывания
    final Context context = this;
```

```
    static AlarmManager alarmManager;
    public static final int REQUEST_CODE_LOC=1;
    static Switch switchView;
    Calendar calendar_zavtro = Calendar.getInstance();
    Calendar calendar_segodny = Calendar.getInstance();
    public static boolean flagVkl=false;
    private int interval;
    public static boolean flagVklB=false;
    public static boolean flagVklI=false;


    @Override
    protected void onDestroy() {
        flagVkl=false;
        SharedPreferences savedData =
getSharedPreferences("SavedData",MODE_PRIVATE);//сохранение данных
        SharedPreferences.Editor editor=savedData.edit();//сохранение данных
        editor.putBoolean("flag",flagVkl);
        editor.apply();
        AlarmActivity.hour= hour;
        AlarmActivity.min= min;

        super.onDestroy();
    }

    @SuppressLint("SetTextI18n")
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        razrysheniy();
        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final Intent intent = new Intent(MainActivity.this,
                        MainBluetooth.class);//Переход к процессу MainBluetooth
                startActivity(intent);
            }
        });
        FloatingActionButton interval2 = findViewById(R.id.interval);
        interval2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                LayoutInflater li = LayoutInflater.from(context);
                View promptsView = li.inflate(R.layout.prompts, null);
                AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                        context);//Вызов диалогового окна для изменения интервала
                alertDialogBuilder.setView(promptsView);

                final EditText userInput = (EditText) promptsView
```

```
                .findViewById(R.id.editTextDialogUserInput);

        // Устанавливаем как выглядит диалоговое окно
        alertDialogBuilder
                .setCancelable(false)
                .setPositiveButton("OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,int id) {
                                // get user input and set it to result
                                // edit text


                                String im =String.valueOf(userInput.getText());
                                try {
                                    interval=Integer.parseInt (im);// изменение интервала
                                }catch (NumberFormatException e){//обработка того что ввели не правильно
число
                                    Toast.makeText(MainActivity.this, "Вы ввели неправильное число.\n(Вы
могли ввести число с буквой или дробное число)", Toast.LENGTH_LONG).show();
                                interval=20;
                                }


                                SharedPreferences savedData =
getSharedPreferences("SavedData",MODE_PRIVATE);//Загрузка сохранённых данных
                                SharedPreferences.Editor editor=savedData.edit();//Загрузка данных
                                editor.putInt("interval", interval);
                                editor.apply();
//Определение как должно отображается время
                                if (min<10){
                                    if (hour<10){
                                        switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
                                    }else{
                                        switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
                                    }}else{
                                    if (hour<10){
                                        switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
                                    }else{
                                        switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
                                    }
                                }
                            }
                        })
                .setNegativeButton("Отмена",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,int id) {
                                dialog.cancel();
                            }
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
```

```
            // Показ диалогового окна
            alertDialog.show();
        }
    });
    SharedPreferences savedData = getSharedPreferences("SavedData",MODE_PRIVATE);//Загрузка
данных
    interval = savedData.getInt("interval", 30);
    hour=savedData.getInt("Hours",0);
    min=savedData.getInt("Min",0);
    flagVkl=savedData.getBoolean("flag",false);

    switchView=findViewById(R.id.switch_bn1);
    if (min<10){
        if (hour<10){
            switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
        }else{
            switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
        }}else{
        if (hour<10){
            switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
        }else{
            switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
        }
    }
    proverka();
    switchView.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
                budilka();//вызов установки будильника
               flagVklB=true;
            } else {
                otmena();//вызов отмены будильника
            }
        }
    });
}

@SuppressLint("UnspecifiedImmutableFlag")
private PendingIntent getAlarmInfoPendingIntent(){
  Intent alarmInfoIntent  = new Intent(this,MainActivity.class);

alarmInfoIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|Intent.FLAG_ACTIVITY_NEW_TAS
K);
   return PendingIntent.getActivity(this,0,alarmInfoIntent,PendingIntent.FLAG_UPDATE_CURRENT);
}

@SuppressLint("UnspecifiedImmutableFlag")
public PendingIntent getAlarmActionPendingIntent(){//Что произойдёт когда сработает будильник
    Intent intent = new Intent(this,AlarmActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|Intent.FLAG_ACTIVITY_NEW_TASK);
    return PendingIntent.getActivity(this,1,intent,PendingIntent.FLAG_UPDATE_CURRENT);
```

```
    }
    @SuppressLint("SetTextI18n")
    public void budilka(){



        SharedPreferences savedData = getSharedPreferences("SavedData",MODE_PRIVATE);//загрузка
данных

        hour=savedData.getInt("Hours",0);
        min=savedData.getInt("Min",0);
        if (min<10){
            if (hour<10){
                switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
            }else{
                switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
            }}else{
            if (hour<10){
                switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
            }else{
                switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
            }
        }

        MaterialTimePicker materialTimePicker =new MaterialTimePicker.Builder()//Вызов диалогового
окна для установки времени срабатывания будильника
                .setTimeFormat(TimeFormat.CLOCK_24H)
                .setHour(hour)
                .setMinute(min)

                .setTitleText("Выбирите время для будильника")
                .build();
        materialTimePicker.addOnPositiveButtonClickListener(view ->{


            Date currentTime = Calendar.getInstance().getTime();
if (currentTime.getHours()>materialTimePicker.getHour()) {
    zavtro(materialTimePicker,currentTime);
    }else {
        if (currentTime.getMinutes()>materialTimePicker.getMinute()){
            zavtro(materialTimePicker,currentTime) ;
            }else{
            segodny(materialTimePicker,currentTime) ;
            }
        }
            interval();
            SharedPreferences.Editor editor=savedData.edit();//сохранение данных
            editor.putInt("Hours", hour);
            editor.putInt("Min", min);
            editor.putBoolean("flag",flagVkl);
            editor.apply();
            Log.d("asd","min="+savedData.getInt("Min",0)+"Hours"+savedData.getInt("Hours",0));
```

```
        if(Objects.equals(day, "0")){
            switchView=findViewById(R.id.switch_bn1);

            if (min<10){
                if (hour<10){
                    switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
                }else{
                    switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
                }}else{
                if (hour<10){
                    switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
                }else{
                    switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
                }
            }
        }else {
            switchView = findViewById(R.id.switch_bn1);
            if (min<10){
                if (hour<10){
                    switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
                }else{
                    switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
                }}else{
                if (hour<10){
                    switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
                }else{
                    switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
                }
            }

        }
    });

    materialTimePicker.show(getSupportFragmentManager(),"Tag_piker");

}
@SuppressLint("SetTextI18n")
public void otmena(){


    SharedPreferences savedData =
getSharedPreferences("SavedData",MODE_PRIVATE);//сохранение данных

  hour=savedData.getInt("Hours",0);
  min=savedData.getInt("Min",00);

  switchView=findViewById(R.id.switch_bn1);
if (min<10){
  if (hour<10){
    switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");
  }else{
```

```
        switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");
    }}else{
    if (hour<10){
        switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");
    }else{
        switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");
    }
    }
    alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
    if (getAlarmActionPendingIntent()!=null){
    alarmManager.cancel(getAlarmActionPendingIntent());}//Отменяем будильник, связанный с
интентом данного класса
   // Toast.makeText(this, "Будильник отменён", Toast.LENGTH_LONG).show();
    flagVkl=false;
    switchView.setChecked(flagVkl);
    SharedPreferences.Editor editor=savedData.edit();//сохранение данных
    editor.putInt("Hours", hour);
    editor.putInt("Min", min);
    editor.putBoolean("flag",flagVkl);
    editor.apply();
    }
    public void zavtro(MaterialTimePicker materialTimePicker,Date currentTime){//Установка времени
срабатывания на завтра
        flagVkl=true;
        calendar_zavtro.set(Calendar.SECOND, currentTime.getSeconds());
        calendar_zavtro.set(Calendar.MILLISECOND, 0);
        calendar_zavtro.set(Calendar.MINUTE, materialTimePicker.getMinute());
        calendar_zavtro.set(Calendar.HOUR_OF_DAY, materialTimePicker.getHour());

        calendar_zavtro.set(Calendar.DAY_OF_MONTH,currentTime.getDate()+1);
        day=calendar_zavtro.getTime().getDate();

        min = materialTimePicker.getMinute();//сохранение данных
        hour = materialTimePicker.getHour();

        alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        AlarmManager.AlarmClockInfo alarmClockInfo = new
AlarmManager.AlarmClockInfo(calendar_zavtro.getTimeInMillis(), getAlarmInfoPendingIntent());
        alarmManager.setAlarmClock(alarmClockInfo, getAlarmActionPendingIntent());
        Toast.makeText(this, "Будильник установлен на: " + sdf.format(calendar_zavtro.getTime()),
Toast.LENGTH_LONG).show();

    }
    public  void segodny(MaterialTimePicker materialTimePicker,Date currentTime){//установка
времени срабатывания будильника на сегодня
        flagVkl=true;
        calendar_segodny.set(Calendar.SECOND, currentTime.getSeconds());
        calendar_segodny.set(Calendar.MILLISECOND, 0);
        calendar_segodny.set(Calendar.MINUTE, materialTimePicker.getMinute());
        calendar_segodny.set(Calendar.HOUR_OF_DAY, materialTimePicker.getHour());
        calendar_segodny.set(Calendar.DAY_OF_MONTH,currentTime.getDate());
```

```
        min = materialTimePicker.getMinute();//загрузка данных
        hour = materialTimePicker.getHour();
        day=calendar_segodny.getTime().getDate();


        alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        AlarmManager.AlarmClockInfo alarmClockInfo = new
AlarmManager.AlarmClockInfo(calendar_segodny.getTimeInMillis(), getAlarmInfoPendingIntent());
        alarmManager.setAlarmClock(alarmClockInfo, getAlarmActionPendingIntent());
        Toast.makeText(this, "Будильник установлен на: " + sdf.format(calendar_segodny.getTime()),
Toast.LENGTH_LONG).show();



    }
    public void proverka(){//Проверка на то выданы и нужные разрешения программе
        switchView=findViewById(R.id.switch_bn1);
       switchView.setChecked(flagVkl);
    }
    public void razrysheniy(){
        int accessCoarseLocation =
this.checkSelfPermission(android.Manifest.permission.ACCESS_COARSE_LOCATION);
        int accessFineLocation   =
this.checkSelfPermission(android.Manifest.permission.ACCESS_FINE_LOCATION);
        int
acessWRITEEXTERNALSTORAGE=this.checkSelfPermission(android.Manifest.permission.WRITE_E
XTERNAL_STORAGE);
        List<String> listRequestPermission = new ArrayList<String>();

        if (accessCoarseLocation != PackageManager.PERMISSION_GRANTED) {
            listRequestPermission.add(android.Manifest.permission.ACCESS_COARSE_LOCATION);
        }
        if (accessFineLocation != PackageManager.PERMISSION_GRANTED) {
            listRequestPermission.add(android.Manifest.permission.ACCESS_FINE_LOCATION);
        }
        if (acessWRITEEXTERNALSTORAGE != PackageManager.PERMISSION_GRANTED) {
            listRequestPermission.add(android.Manifest.permission.WRITE_EXTERNAL_STORAGE);
        }

        if (!listRequestPermission.isEmpty()) {
            String[] strRequestPermission = listRequestPermission.toArray(new
String[listRequestPermission.size()]);
            this.requestPermissions(strRequestPermission, REQUEST_CODE_LOC);
        }
    }
    public void interval(){//Высчитывание минимального времени срабатывания будильника

     if (min-interval<0){
         min2=min-interval+60;
         if (hour-1<0){
             hour2=hour+23;
            day2=day-1;
```

```
        }else{
            hour2=hour-1;
            day2= day;
        }
    }else{
        min2=min-interval;
        hour2=hour;
        day2=day;
    }
   }
}
```

Приложение Б

Код класса *MainBluetooth*

```
package com.example.budilnik;

import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanCallback;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.PowerManager;
import android.provider.Settings;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.List;

public class MainBluetooth extends AppCompatActivity {

    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothLeScanner mBluetoothLeScanner;
    private boolean mScanning;
    private static final int RQS_ENABLE_BLUETOOTH = 1;
    Button btnScan;
    ListView listViewLE;
    List<BluetoothDevice> listBluetoothDevice;
    List<String> listBluetoothDeviceString;
    ListAdapter adapterLeScanResult;
    private Handler mHandler;
    private static final long SCAN_PERIOD = 20000;

    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main_bluetooth);
   Intent intent = new Intent();
   String packageName = getPackageName();
   PowerManager pm = (PowerManager) getSystemService(POWER_SERVICE);
   if (!pm.isIgnoringBatteryOptimizations(packageName)) {
      Toast.makeText(this,
            "Пожалуйста, отключите оптимизацию расхода батареи для этого приложения. " +
                  "\n Это нужно для правильной работы приложения.",
            Toast.LENGTH_LONG).show();
   intent.setAction(Settings.ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS);
      //intent.setData(Uri.parse("package:" + packageName));
      startActivity(intent); }

   // Check if BLE is supported on the device.
   if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
      Toast.makeText(this,
            "BLUETOOTH_LE not supported in this device!",
            Toast.LENGTH_SHORT).show();
      finish();
   }

   getBluetoothAdapterAndLeScanner();

   // Checks if Bluetooth is supported on the device.
   if (mBluetoothAdapter == null) {
      Toast.makeText(this,
            "bluetoothManager.getAdapter()==null",
            Toast.LENGTH_SHORT).show();
      finish();
      return;
   }

   btnScan = (Button)findViewById(R.id.scan);
   btnScan.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         scanLeDevice(true);
      }
   });
   listViewLE = (ListView)findViewById(R.id.lelist);

   listBluetoothDevice = new ArrayList<>();
   listBluetoothDeviceString = new ArrayList<>();
   adapterLeScanResult = new ArrayAdapter<String>(
         this, android.R.layout.simple_list_item_1, listBluetoothDeviceString);
   listViewLE.setAdapter(adapterLeScanResult);
   listViewLE.setOnItemClickListener(scanResultOnItemClickListener);

   mHandler = new Handler();
```

```
}
AdapterView.OnItemClickListener scanResultOnItemClickListener =
      new AdapterView.OnItemClickListener(){

         @Override
         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            final BluetoothDevice device = (BluetoothDevice) listBluetoothDevice.get(position);

            String msg = device.getAddress() + "\n"
                  +"Bluetooth класс "+ device.getBluetoothClass().toString() + "\n"
                  + getBTDevieType(device);
            new AlertDialog.Builder(MainBluetooth.this)
                  .setTitle(device.getName())
                  .setMessage(msg)
                  .setPositiveButton("Назад", new DialogInterface.OnClickListener() {
                     @Override
                     public void onClick(DialogInterface dialog, int which) {

                     }
                  })
                  .setNeutralButton("Подключится", new DialogInterface.OnClickListener() {
                     @Override
                     public void onClick(DialogInterface dialog, int which) {
                        final Intent intent = new Intent(MainBluetooth.this,
                              ControlActivity.class);
                        intent.putExtra(ControlActivity.EXTRAS_DEVICE_NAME,
                              device.getName());
                        intent.putExtra(ControlActivity.EXTRAS_DEVICE_ADDRESS,
                              device.getAddress());
                        if (mScanning) {
                           mBluetoothLeScanner.stopScan(scanCallback);
                           mScanning = false;
                           btnScan.setEnabled(true);
                        }
                        startActivity(intent);
                     }
                  })
                  .show();

         }
      };

private String getBTDevieType(BluetoothDevice d){
   String type = "";

   switch (d.getType()){
      case BluetoothDevice.DEVICE_TYPE_CLASSIC:
         type = "DEVICE_TYPE_CLASSIC";
         break;
      case BluetoothDevice.DEVICE_TYPE_DUAL:
         type = "DEVICE_TYPE_DUAL";
         break;
```

```
            case BluetoothDevice.DEVICE_TYPE_LE:
                type = "DEVICE_TYPE_LE";
                break;
            case BluetoothDevice.DEVICE_TYPE_UNKNOWN:
                type = "DEVICE_TYPE_UNKNOWN";
                break;
            default:
                type = "unknown...";
        }

        return type;
    }

    @Override
    protected void onResume() {
        super.onResume();

        if (!mBluetoothAdapter.isEnabled()) {
            if (!mBluetoothAdapter.isEnabled()) {
                Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, RQS_ENABLE_BLUETOOTH);
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        if (requestCode == RQS_ENABLE_BLUETOOTH && resultCode ==
Activity.RESULT_CANCELED) {
            finish();
            return;
        }

        getBluetoothAdapterAndLeScanner();

        // Checks if Bluetooth is supported on the device.
        if (mBluetoothAdapter == null) {
            Toast.makeText(this,
                    "bluetoothManager.getAdapter()==null",
                    Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        super.onActivityResult(requestCode, resultCode, data);
    }

    private void getBluetoothAdapterAndLeScanner(){
        // Get BluetoothAdapter and BluetoothLeScanner.
        final BluetoothManager bluetoothManager =
                (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
```

```
        mBluetoothAdapter = bluetoothManager.getAdapter();
        mBluetoothLeScanner = mBluetoothAdapter.getBluetoothLeScanner();

        mScanning = false;
    }

    /*
    to call startScan (ScanCallback callback),
    Requires BLUETOOTH_ADMIN permission.
    Must hold ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission to get
results.
     */
    private void scanLeDevice(final boolean enable) {
        if (enable) {
            listBluetoothDevice.clear();
            listBluetoothDeviceString.clear();
            listViewLE.invalidateViews();

            // Stops scanning after a pre-defined scan period.
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    mBluetoothLeScanner.stopScan(scanCallback);
                    listViewLE.invalidateViews();

                    Toast.makeText(MainBluetooth.this,
                            "Scan timeout",
                            Toast.LENGTH_LONG).show();

                    mScanning = false;
                    btnScan.setEnabled(true);
                }
            }, SCAN_PERIOD);

            //mBluetoothLeScanner.startScan(scanCallback);
            //scan specified devices only with ScanFilter
            ScanFilter scanFilter =
                    new ScanFilter.Builder()
                            .setServiceUuid(BluetoothLeService.ParcelUuid_Puls_ledService)
                            .build();
            List<ScanFilter> scanFilters = new ArrayList<ScanFilter>();
            scanFilters.add(scanFilter);

            ScanSettings scanSettings =
                    new ScanSettings.Builder().build();

            mBluetoothLeScanner.startScan(scanFilters, scanSettings, scanCallback);
            mScanning = true;
            btnScan.setEnabled(false);
        } else {
            mBluetoothLeScanner.stopScan(scanCallback);
            mScanning = false;
```

```
            btnScan.setEnabled(true);
        }
    }

    private ScanCallback scanCallback = new ScanCallback() {
        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);

            //TODO
            addBluetoothDevice(result.getDevice());
        }

        @Override
        public void onBatchScanResults(List<ScanResult> results) {
            super.onBatchScanResults(results);
            for(ScanResult result : results){
                addBluetoothDevice(result.getDevice());
            }
        }

        @Override
        public void onScanFailed(int errorCode) {
            super.onScanFailed(errorCode);
            Toast.makeText(MainBluetooth.this,
                    "onScanFailed: " + String.valueOf(errorCode),
                    Toast.LENGTH_LONG).show();
        }

        private void addBluetoothDevice(BluetoothDevice device){
            if(!listBluetoothDevice.contains(device)){
                listBluetoothDevice.add(device);
                listBluetoothDeviceString.add(device.getName());
                listViewLE.invalidateViews();
            }
        }
    };
    public void oBacking(View view) {
        final Intent intent = new Intent(MainBluetooth.this,
                MainActivity.class);
        if (mScanning) {
            mBluetoothLeScanner.stopScan(scanCallback);
            mScanning = false;
            btnScan.setEnabled(true);
        }
        startActivity(intent);
    }
}
```

Приложение В

Код класса *ControlActivity*

```
package com.example.budilnik;

import static com.example.budilnik.BluetoothLeService.String_Name_switchChar;
import static com.example.budilnik.BluetoothLeService.heartRate;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattService;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.opengl.Visibility;
import android.os.Bundle;
import android.os.Environment;
import android.os.IBinder;
import android.util.Log;
import android.view.View;
import android.widget.ExpandableListView;
import android.widget.SimpleExpandableListAdapter;
import android.widget.TextView;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.UUID;

public class ControlActivity extends AppCompatActivity {
    public static String Name_Device;
    final String FILENAME = "LOG.txt";
    private final static String TAG = "321";
    public static final String EXTRAS_DEVICE_NAME = "DEVICE_NAME";
    public static final String EXTRAS_DEVICE_ADDRESS = "DEVICE_ADDRESS";
    private static final String DIR_SD = "MyFile";
    private String mDeviceName;
    private String mDeviceAddress;

    private boolean mConnected = false;
    private BluetoothGattCharacteristic mNotifyCharacteristic;
    private BluetoothGattCharacteristic mBatteryNotifyCharacteristic;
    private BluetoothGattCharacteristic mPulsNotifyCharacteristic;
```

```
    public static BluetoothLeService mBluetoothLeService;
    static TextView textViewState;
    private ExpandableListView mGattServicesList;
    private final String LIST_NAME = "Пульсометр";
    private final String LIST_UUID = "0000180d-0000-1000-8000-00805f9b34fb";
    private ArrayList<ArrayList<BluetoothGattCharacteristic>> mGattCharacteristics = new
ArrayList<ArrayList<BluetoothGattCharacteristic>>();
    private final ServiceConnection mServiceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder service) {
            mBluetoothLeService = ((BluetoothLeService.LocalBinder) service).getService();
            if (!mBluetoothLeService.initialize()) {
                Log.e(TAG, "Unable to initialize Bluetooth");
                finish();
            }
            // Automatically connects to the device upon successful start-up initialization.
            mBluetoothLeService.connect(mDeviceAddress);
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            mBluetoothLeService = null;
        }
    };

    // Handles various events fired by the Service.
    // ACTION_GATT_CONNECTED: connected to a GATT server.
    // ACTION_GATT_DISCONNECTED: disconnected from a GATT server.
    // ACTION_GATT_SERVICES_DISCOVERED: discovered GATT services.
    // ACTION_DATA_AVAILABLE: received data from the device.  This can be a result of read
    //                        or notification operations.
    private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();
            if (BluetoothLeService.ACTION_GATT_CONNECTED.equals(action)) {
                    mConnected = true;
                updateConnectionState("GATT_CONNECTED");
            } else if (BluetoothLeService.ACTION_GATT_DISCONNECTED.equals(action)) {
                mConnected = false;
                updateConnectionState("GATT_DISCONNECTED");
                View backing = findViewById(R.id.Backing);
                View bar = findViewById(R.id.progressBar);
                backing.setVisibility(View.VISIBLE);
                bar.setVisibility(View.GONE);
                clearUI();
            } else if (BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)) {
                // Show all the supported services and characteristics on the user interface.
                displayGattServices(mBluetoothLeService.getSupportedGattServices());
            } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
            }
        }
```

```
};


private void clearUI() {
    mGattServicesList.setAdapter((SimpleExpandableListAdapter) null);
}

private void updateConnectionState(final String st) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            textViewState.setText(st);
        }
    });
}

public static void displayData(String data) {
    if (data != null) {
        textViewState.setText(data);
    }
}

// Demonstrates how to iterate through the supported GATT Services/Characteristics.
// In this sample, we populate the data structure that is bound to the ExpandableListView
// on the UI.
private void displayGattServices(List<BluetoothGattService> gattServices) {

    if (gattServices == null) return;
    String uuid = null;
    String unknownServiceString = "Unknown Service";
    String unknownCharaString = "Unknown Characteristic";
    ArrayList<HashMap<String, String>> gattServiceData =
            new ArrayList<HashMap<String, String>>();
    ArrayList<ArrayList<HashMap<String, String>>> gattCharacteristicData
            = new ArrayList<ArrayList<HashMap<String, String>>>();
    mGattCharacteristics = new ArrayList<ArrayList<BluetoothGattCharacteristic>>();


    // Loops through available GATT Services.
    for (BluetoothGattService gattService : gattServices) {
        HashMap<String, String> currentServiceData = new HashMap<String, String>();
        uuid = gattService.getUuid().toString();
        currentServiceData.put(
                LIST_NAME, lookup(uuid, unknownServiceString));
        currentServiceData.put(LIST_UUID, uuid);
        gattServiceData.add(currentServiceData);

        ArrayList<HashMap<String, String>> gattCharacteristicGroupData =
                new ArrayList<HashMap<String, String>>();
        List<BluetoothGattCharacteristic> gattCharacteristics =
                gattService.getCharacteristics();
        ArrayList<BluetoothGattCharacteristic> charas =
```

```
            new ArrayList<BluetoothGattCharacteristic>();

        // Loops through available Characteristics.
        for (BluetoothGattCharacteristic gattCharacteristic : gattCharacteristics) {
            charas.add(gattCharacteristic);
            HashMap<String, String> currentCharaData = new HashMap<String, String>();
            uuid = gattCharacteristic.getUuid().toString();
            currentCharaData.put(
                    LIST_NAME, lookup(uuid, unknownCharaString));
            currentCharaData.put(LIST_UUID, uuid);
            if (uuid==String_Name_switchChar){
               Name_Device= String.valueOf(gattCharacteristic.getValue());
            }
            gattCharacteristicGroupData.add(currentCharaData);


        }
        mGattCharacteristics.add(charas);
        gattCharacteristicData.add(gattCharacteristicGroupData);

    }
    final BluetoothGattCharacteristic mBateriCharakteristic =
            mGattCharacteristics.get(6).get(0);

    mBluetoothLeService.readCharacteristic(mBateriCharakteristic);

    mBluetoothLeService.setCharacteristicNotification(
            mBateriCharakteristic, true);
    mBatteryNotifyCharacteristic = mBateriCharakteristic;
    try {
       Thread.sleep(100);
    } catch (InterruptedException e) {
       throw new RuntimeException(e);
    }
    final BluetoothGattCharacteristic mPulsCharakteristic =
            mGattCharacteristics.get(5).get(0);

    mBluetoothLeService.readCharacteristic( mPulsCharakteristic);

    mBluetoothLeService.setCharacteristicNotification(
            mPulsCharakteristic, true);
    mPulsNotifyCharacteristic =  mPulsCharakteristic;
    if (heartRate!=0){
MainActivity.flagVklI=true;}

    BackingGl();
    SimpleExpandableListAdapter gattServiceAdapter = new SimpleExpandableListAdapter(
            this,
            gattServiceData,
            android.R.layout.simple_expandable_list_item_2,
            new String[] {LIST_NAME, LIST_UUID},
            new int[] { android.R.id.text1, android.R.id.text2 },
```

```
            gattCharacteristicData,
            android.R.layout.simple_expandable_list_item_2,
            new String[] {LIST_NAME, LIST_UUID},
            new int[] { android.R.id.text1, android.R.id.text2 }
        );
        mGattServicesList.setAdapter(gattServiceAdapter);

    }
    void writeFileSD() {
        // проверяем доступность SD
        if (!Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
            Log.d(TAG, "SD-карта не доступна: " + Environment.getExternalStorageState());
            return;
        }
        // получаем путь к SD
        File sdPath = Environment.getExternalStorageDirectory();
        // добавляем свой каталог к пути
        sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
        // создаем каталог
        sdPath.mkdirs();
        // формируем объект File, который содержит путь к файлу
        File sdFile = new File(sdPath, FILENAME);
        try {
            // открываем поток для записи
            BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile));
            // пишем данные
            bw.write("Содержимое файла на SD\n");
            // закрываем поток
            bw.close();
            Log.d(TAG, "Файл записан на SD: " + sdFile.getAbsolutePath());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private final ExpandableListView.OnChildClickListener servicesListClickListner = new
ExpandableListView.OnChildClickListener() {
            @Override
            public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,
                                        int childPosition, long id) {
                if (mGattCharacteristics != null) {
                    writeFileSD();
                    final BluetoothGattCharacteristic characteristic =
                            mGattCharacteristics.get(groupPosition).get(childPosition);

                    final int charaProp = characteristic.getProperties();
                    if ((charaProp | BluetoothGattCharacteristic.PROPERTY_READ) > 0) {
                        // If there is an active notification on a characteristic, clear
                        // it first so it doesn't update the data field on the user interface.
                        if (mNotifyCharacteristic != null) {
                            mBluetoothLeService.setCharacteristicNotification(
```

```
                        mNotifyCharacteristic, false);
                mNotifyCharacteristic = null;
            }

            mBluetoothLeService.readCharacteristic(characteristic);
            //TODO characteristic.getValue();
        }
        if ((charaProp | BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {




            mNotifyCharacteristic = characteristic;
            mBluetoothLeService.setCharacteristicNotification(
                    characteristic, true);


        }
        return true;

      }
      return false;
    }
  };

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_control);
  View backing = findViewById(R.id.Backing);
  View bar = findViewById(R.id.progressBar);
  bar.setVisibility(View.VISIBLE);
  backing.setVisibility(View.GONE);

  startService(new Intent(this, BluetoothLeService.class));
  final Intent intent = getIntent();
  mDeviceName = intent.getStringExtra(EXTRAS_DEVICE_NAME);
  mDeviceAddress = intent.getStringExtra(EXTRAS_DEVICE_ADDRESS);

  TextView textViewDeviceName = (TextView)findViewById(R.id.textDeviceName);
  TextView textViewDeviceAddr = (TextView)findViewById(R.id.textDeviceAddress);
  textViewState = (TextView)findViewById(R.id.textState);

  textViewDeviceName.setText(mDeviceName);
  Name_Device=mDeviceName;
  textViewDeviceAddr.setText(mDeviceAddress);

  mGattServicesList = (ExpandableListView) findViewById(R.id.gatt_services_list);
  mGattServicesList.setOnChildClickListener(servicesListClickListner);

  Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);
  bindService(gattServiceIntent, mServiceConnection, BIND_AUTO_CREATE);
```

```
    }

    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());
        if (mBluetoothLeService != null) {
            final boolean result = mBluetoothLeService.connect(mDeviceAddress);
            Log.d(TAG, "Connect request result=" + result);
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        unregisterReceiver(mGattUpdateReceiver);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unbindService(mServiceConnection);
        //mBluetoothLeService = null;
    }
    private static IntentFilter makeGattUpdateIntentFilter() {
        final IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_CONNECTED);
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_DISCONNECTED);
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED);
        intentFilter.addAction(BluetoothLeService.ACTION_DATA_AVAILABLE);
        return intentFilter;
    }
    private static HashMap<String, String> attributes = new HashMap();
    public void oBacking(View view) {
        BackingGl();
    }
    public void BackingGl(){
        final Intent intent = new Intent(ControlActivity.this,
                MainActivity.class);
        startActivity(intent);
    }
}
```

Приложение Г

Код класса *AlarmActivity*

```
package com.example.budilnik;

import android.annotation.SuppressLint;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.View;
import android.widget.Switch;
import androidx.appcompat.app.AppCompatActivity;
import org.jetbrains.annotations.Nullable;

public class AlarmActivity extends AppCompatActivity {

    Ringtone ringtone;
    Vibrator vibrator;
    public static Integer hour;
    public static Integer min;
    public boolean flagVkl;
    static Switch switchView;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (MainActivity.hour!=null&&MainActivity.min!=null){
            hour=MainActivity.hour;
            min=MainActivity.min;
        }else {
            SharedPreferences savedData = getSharedPreferences("SavedData",MODE_PRIVATE);

            hour=savedData.getInt("Hours",0);
            min=savedData.getInt("Min",0);
        }
        otmena();
        vibrator = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);


        setContentView(R.layout.activity_alarm);
```

```
        Uri notificationUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
        ringtone = RingtoneManager.getRingtone(this,notificationUri);
        if (ringtone==null){
            notificationUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
            ringtone = RingtoneManager.getRingtone(this,notificationUri);
        }
        if (ringtone!=null){
            ringtone.play();
        }
        long[] pattern = {0, 2000, 1000, 2500, 1500, 3000};
        vibrator.vibrate(pattern,1);

    }

    @Override
    protected void onStop() {
        super.onStop();
        otmena();
    }

    @Override
    protected void onDestroy() {
        SharedPreferences savedData =
getSharedPreferences("SavedData",MODE_PRIVATE);//сохранение данных
        flagVkl=false;
        SharedPreferences.Editor editor=savedData.edit();//сохранение данных
        editor.putBoolean("flag",flagVkl);
        editor.apply();
        if (ringtone!=null&&ringtone.isPlaying()){
            ringtone.stop();

        }
        vibrator.cancel();
        MainActivity.hour=this.hour;
        MainActivity.min=this.min;
        otmena();
        super.onDestroy();
    }


    public void cansel(View view) {
        Intent intent = new Intent(this,MainActivity.class);
        if (ringtone!=null&&ringtone.isPlaying()){
            ringtone.stop();

        }
        vibrator.cancel();
        otmena();
        startActivity(intent);
    }
```

```
    public void otmena(){
SharedPreferences savedData = getSharedPreferences("SavedData",MODE_PRIVATE);//сохранение
данных
        hour=savedData.getInt("Hours",0);
        min=savedData.getInt("Min",0);
        switchView=findViewById(R.id.switch_bn1);
        MainActivity.alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        if (getAlarmActionPendingIntent()!=null){
            MainActivity.alarmManager.cancel(getAlarmActionPendingIntent());}//Отменяем будильник,
связанный с интентом данного класса
       // Toast.makeText(this, "Будильник отменён", Toast.LENGTH_LONG).show();
        MainActivity.flagVkl=false;
        MainActivity.switchView.setChecked(false);
        SharedPreferences.Editor editor=savedData.edit();//сохранение данных
        editor.putInt("Hours", hour);
        editor.putInt("Min", min);
        editor.putBoolean("flag",flagVkl);
        editor.apply();
    }

    @SuppressLint("UnspecifiedImmutableFlag")
    private PendingIntent getAlarmInfoPendingIntent(){
        Intent alarmInfoIntent  = new Intent(this,MainActivity.class);

alarmInfoIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|Intent.FLAG_ACTIVITY_NEW_TAS
K);
        return
PendingIntent.getActivity(this,0,alarmInfoIntent,PendingIntent.FLAG_UPDATE_CURRENT);
    }
    @SuppressLint("UnspecifiedImmutableFlag")
    PendingIntent getAlarmActionPendingIntent(){
        Intent intent = new Intent(this,AlarmActivity.class);
      intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|Intent.FLAG_ACTIVITY_NEW_TASK);
        return PendingIntent.getActivity(this,1,intent,PendingIntent.FLAG_UPDATE_CURRENT);
    }
}
```

Приложение Д

Код класса *BluetoothLeService*

```
package com.example.budilnik;

import static androidx.core.app.NotificationCompat.PRIORITY_DEFAULT;
import static androidx.core.app.NotificationCompat.PRIORITY_HIGH;
import static androidx.core.app.NotificationCompat.PRIORITY_MAX;
import android.annotation.SuppressLint;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCallback;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattDescriptor;
import android.bluetooth.BluetoothGattService;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothProfile;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.os.Binder;
import android.os.Build;
import android.os.Environment;
import android.os.IBinder;
import android.os.ParcelUuid;
import android.util.Log;
import android.view.View;
import androidx.core.app.NotificationCompat;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.UUID;
public class BluetoothLeService extends Service {
    public float batteryPct=0;
    private final static String TAG = "123";
```

```
    private static final String DIR_SD = "MyFile";
    final String FILENAME = "LOG.txt";
    public byte[] data;
    private NotificationManager notificationManager;
    // Идентификатор уведомления
    private static final int NOTIFY_ID = 101;
    private static final int NOTIFYZU_ID = 102;
    private static final int NOTIFYZT_ID = 103;
    private static final int NOTIFYO_ID = 104;
    // Идентификаторы каналов уведомлений
    private static final String CHANNEL_ID = "Оповещения о входящих данных";
    private static final String CHANNELZU_ID = "Оповещение о малом уровне заряда пульсометра";
    private static final String CHANNELZT_ID = "Оповещение о недостаточном уровне заряда
телефона";
    private static final String CHANNELO_ID = "Оповещение об отключении пульсометра от
телефона";
    private BluetoothManager mBluetoothManager;
    private BluetoothAdapter mBluetoothAdapter;
    private String mBluetoothDeviceAddress;
    private BluetoothGatt mBluetoothGatt;
    private int mConnectionState = STATE_DISCONNECTED;
    public static int heartRate;
    private static final int STATE_DISCONNECTED = 0;
    private static final int STATE_CONNECTING = 1;
    private static final int STATE_CONNECTED = 2;
    public final static String ACTION_GATT_CONNECTED =
            "android-er.ACTION_GATT_CONNECTED";
    public final static String ACTION_GATT_DISCONNECTED =
            "android-er.ACTION_GATT_DISCONNECTED";
    public final static String ACTION_GATT_SERVICES_DISCOVERED =
            "android-er.ACTION_GATT_SERVICES_DISCOVERED";
    public final static String ACTION_DATA_AVAILABLE =
            "android-er.ACTION_DATA_AVAILABLE";
    public final static String EXTRA_DATA =
            "android-er.EXTRA_DATA";
    public static Integer timer=0;
    private static final UUID Battery_Level_UUID =
            UUID.fromString("00002a19-0000-1000-8000-00805f9b34fb");
    public static String String_Name_ledService =
            "00001800-0000-1000-8000-00805f9b34fb";
    public static String String_Name_switchChar = "00002a00-0000-1000-8000-00805f9b34fb";
    public final static UUID UUID_Name_switchChare =
            UUID.fromString(String_Name_switchChar);
    public static String String_Puls_ledService =
            "0000180d-0000-1000-8000-00805f9b34fb";
    public final static ParcelUuid ParcelUuid_Puls_ledService =
            ParcelUuid.fromString(String_Puls_ledService);
    public static String String_Puls_switchChar = "00002a37-0000-1000-8000-00805f9b34fb";
    public final static UUID UUID_Puls_switchChare =
            UUID.fromString(String_Puls_switchChar);
    private byte zaryd=0;
    public BluetoothDevice device;
```

```
    private int priv=0;
    private boolean zarydFlag=true;
    private boolean zarydFlag2=true;


    @SuppressLint("MissingPermission")
    public void readCharacteristic(BluetoothGattCharacteristic characteristic) {
        if (mBluetoothAdapter == null || mBluetoothGatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        mBluetoothGatt.readCharacteristic(characteristic);
    }
    @SuppressLint("MissingPermission")
    public void writeCharacteristic() {
        if (mBluetoothAdapter == null || mBluetoothGatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        BluetoothGattService mCustomService =
mBluetoothGatt.getService(UUID.fromString(String_Name_ledService));
        if(mCustomService == null){
            Log.w(TAG, "Custom BLE Service not found");
            return;
        }

        BluetoothGattCharacteristic characteristic1 =
mCustomService.getCharacteristic(UUID_Name_switchChare);
        byte[] value = ControlActivity.Name_Device.getBytes();
        characteristic1.setValue(value);
        mBluetoothGatt.writeCharacteristic(characteristic1);
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {

        writeCharacteristic();
        return START_REDELIVER_INTENT;
    }
    private final BluetoothGattCallback mGattCallback = new BluetoothGattCallback() {
        @SuppressLint("MissingPermission")
        @Override
        public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
            String intentAction;
            if (newState == BluetoothProfile.STATE_CONNECTED) {
                intentAction = ACTION_GATT_CONNECTED;
                mConnectionState = STATE_CONNECTED;
                broadcastUpdate(intentAction);
                Log.i(TAG, "Connected to GATT server.");
                // Attempts to discover services after successful connection.
                Log.i(TAG, "Attempting to start service discovery:" +
                        mBluetoothGatt.discoverServices());
```

```
        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            intentAction = ACTION_GATT_DISCONNECTED;
            mConnectionState = STATE_DISCONNECTED;
            Log.i(TAG, "Disconnected from GATT server.");
            notifykO();
            broadcastUpdate(intentAction);
        }
    }

    @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
            broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED);
        } else {
            Log.w(TAG, "onServicesDiscovered received: " + status);
        }
    }

    @Override
    public void onCharacteristicRead(BluetoothGatt gatt,
                                     BluetoothGattCharacteristic characteristic,
                                     int status) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
            try {
                broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }

    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt,
                                        BluetoothGattCharacteristic characteristic) {
        try {
            broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
};
private void broadcastUpdate(final String action) {
    final Intent intent = new Intent(action);
    sendBroadcast(intent);
}
@SuppressLint("SetTextI18n")
private void broadcastUpdate(final String action,
                             final BluetoothGattCharacteristic characteristic) throws IOException {
    final Intent intent = new Intent(action);

    UUID uuid = characteristic.getUuid();
    if (Battery_Level_UUID.equals(uuid)) {
```

```
            data = characteristic.getValue();
            if (data != null && data.length > 0) {
                zaryd = data[0];
                intent.putExtra(EXTRA_DATA, new String(data, StandardCharsets.UTF_8) + "\n" + data[0]);
                Log.d(TAG, "Zaryd" + new String(data, StandardCharsets.UTF_8) + "  " + data[0]+"gh
"+batteryPct+"%");
            return;
            }
        } else if (UUID_Puls_switchChare.equals(uuid)) {
            if (timer >= 10) {
                writeCharacteristic();
                timer = 0;
                notifyk();
            } else {
                timer = timer + 1;
                Log.d(TAG, String.valueOf(timer));
            }
            if (MainActivity.flagVklB) {
                budilnik();
            }
            int flag = characteristic.getProperties();
            int format = -1;
            if ((flag & 0x01) != 0) {
                format = BluetoothGattCharacteristic.FORMAT_UINT16;
                Log.d(TAG, "Heart rate format UINT16.");
            } else {
                format = BluetoothGattCharacteristic.FORMAT_UINT8;
                Log.d(TAG, "Heart rate format UINT8.");
            }
            heartRate = characteristic.getIntValue(format, 1);
            Log.d(TAG, String.format("Received heart rate: %d", heartRate));
            IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
            Intent batteryStatus = this.registerReceiver(null, ifilter);
            int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
            batteryPct = level * 100 / (float)scale;
            Date cutTime = Calendar.getInstance().getTime();
            if (cutTime.getHours()>20){
            if (batteryPct<60){
                if (zarydFlag){
            notifykZT();
                zarydFlag=false;}
            }else {
                zarydFlag=true;
            }}
            if (zaryd<4){
                if (zarydFlag2){
                    notifykZP();
                    zarydFlag2=false;}
                else{
                    zarydFlag2=true;
                }
```

```
        }
        // проверяем доступность SD
        if (!Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
            Log.d(TAG, "SD-карта не доступна: " + Environment.getExternalStorageState());
            return;
        }
        // получаем путь к SD
        File sdPath = Environment.getExternalStorageDirectory();
        // добавляем свой каталог к пути
        sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
        // создаем каталог
        sdPath.mkdirs();
        // формируем объект File, который содержит путь к файлу
        File sdFile = new File(sdPath, FILENAME);

        // отрываем поток для записи
        BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile, true));
        // пишем данные
        SimpleDateFormat sdf = new SimpleDateFormat("'Date :'dd.MM' and Time :'HH.mm.ss z");
        String currentDateAndTime = sdf.format(new Date());
        bw.write("Пульс "+heartRate + " :  " + currentDateAndTime +"  Заряд "+zaryd+ "  Заряд
телефона "+batteryPct+ "%\n");
        // закрываем поток
        bw.close();
        Log.d(TAG, "Файл записан" + sdFile.getAbsolutePath());

        intent.putExtra(EXTRA_DATA, String.valueOf(heartRate));

    } else {// For all other profiles, writes the data formatted in HEX.
        data = characteristic.getValue();
        if (data != null && data.length > 0) {

            intent.putExtra(EXTRA_DATA, new String(data, StandardCharsets.UTF_8) + "\n" + data[0]);
            Log.d(TAG, "{{{}}}}" + new String(data, StandardCharsets.UTF_8) + "  " + data[0]);
        }
    }

    //remove special handling for time being
    Log.w(TAG, "broadcastUpdate()");



    sendBroadcast(intent);


        //ControlActivity.displayData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));

}
public static void createChannelIfNeeded(NotificationManager manager){
    if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.O){
        NotificationChannel notificationChannel=new
```

```
NotificationChannel(CHANNEL_ID,CHANNEL_ID, NotificationManager.IMPORTANCE_LOW);
            manager.createNotificationChannel(notificationChannel);
            NotificationChannel notificationChannelZU=new
NotificationChannel(CHANNELZU_ID,CHANNELZU_ID,
NotificationManager.IMPORTANCE_DEFAULT);
            manager.createNotificationChannel(notificationChannelZU);
            NotificationChannel notificationChannelZT=new
NotificationChannel(CHANNELZT_ID,CHANNELZT_ID,
NotificationManager.IMPORTANCE_DEFAULT);
            manager.createNotificationChannel(notificationChannelZT);
            NotificationChannel notificationChannelO=new
NotificationChannel(CHANNELO_ID,CHANNELO_ID,
NotificationManager.IMPORTANCE_DEFAULT);
            manager.createNotificationChannel(notificationChannelO);
    }
}
    public class LocalBinder extends Binder {
        BluetoothLeService getService() {
            return BluetoothLeService.this;
        }
    }
    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
    @Override
    public boolean onUnbind(Intent intent) {
        close();
        return super.onUnbind(intent);
    }
    private final IBinder mBinder = new LocalBinder();
    public boolean initialize() {
        if (mBluetoothManager == null) {
            mBluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
            if (mBluetoothManager == null) {
                Log.e(TAG, "Unable to initialize BluetoothManager.");
                return false;
            }
        }

        mBluetoothAdapter = mBluetoothManager.getAdapter();
        if (mBluetoothAdapter == null) {
            Log.e(TAG, "Unable to obtain a BluetoothAdapter.");
            return false;
        }

        return true;
    }
    @SuppressLint("MissingPermission")
    public boolean connect(final String address) {
        if (mBluetoothAdapter == null || address == null) {
            Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
```

```
            return false;
        }

        // Previously connected device.  Try to reconnect.
        if (mBluetoothDeviceAddress != null && address.equals(mBluetoothDeviceAddress)
                && mBluetoothGatt != null) {
            Log.d(TAG, "Trying to use an existing mBluetoothGatt for connection.");
            if (mBluetoothGatt.connect()) {
                mConnectionState = STATE_CONNECTING;
                return true;
            } else {
                return false;
            }
        }

        device = mBluetoothAdapter.getRemoteDevice(address);
        if (device == null) {
            Log.w(TAG, "Device not found.  Unable to connect.");
            return false;
        }
        // We want to directly connect to the device, so we are setting the autoConnect
        // parameter to false.
        mBluetoothGatt = device.connectGatt(this, false, mGattCallback);
        Log.d(TAG, "Trying to create a new connection.");
        mBluetoothDeviceAddress = address;
        mConnectionState = STATE_CONNECTING;

        return true;
    }
    @SuppressLint("MissingPermission")
    public void disconnect() {
        if (mBluetoothAdapter == null || mBluetoothGatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        mBluetoothGatt.disconnect();
    }
    @SuppressLint("MissingPermission")
    public void close() {
        if (mBluetoothGatt == null) {
            return;
        }
        mBluetoothGatt.close();
        mBluetoothGatt = null;
    }
    @SuppressLint("MissingPermission")
    public void setCharacteristicNotification(BluetoothGattCharacteristic characteristic, boolean enabled)
    {
        if (mBluetoothAdapter == null || mBluetoothGatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
```

```
        mBluetoothGatt.setCharacteristicNotification(characteristic, enabled);

        // This is specific to Heart Rate Measurement.
        if (UUID_Puls_switchChare.equals(characteristic.getUuid())) {
            BluetoothGattDescriptor descriptor = characteristic.getDescriptor(UUID.fromString("00002902-
0000-1000-8000-00805f9b34fb"));
            descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
            mBluetoothGatt.writeDescriptor(descriptor);
        }
    }
    public List<BluetoothGattService> getSupportedGattServices() {
        if (mBluetoothGatt == null) return null;

        return mBluetoothGatt.getServices();
    }
    public void notifyk(){

    //readCustomCharacteristic();
    notificationManager=(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
    Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);
    intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
    PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(),0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
    NotificationCompat.Builder notificationBulder=
            new NotificationCompat.Builder(getApplicationContext(),CHANNEL_ID)
                    .setAutoCancel(false)
                    .setSmallIcon(R.drawable.ic_stat_name)
                    .setWhen(System.currentTimeMillis())
                    .setContentIntent(pendingIntent)
                    .setContentTitle("Входные данные")
                    .setContentText("Пульс = "+ heartRate+"\nЗаряд = "+zaryd+"\nЗаряд телефона =
"+batteryPct)
                    .setPriority(PRIORITY_MAX);
    createChannelIfNeeded(notificationManager);
    notificationManager.notify(NOTIFY_ID,notificationBulder.build());
    MainActivity.heartRate=heartRate;
    }
    public void notifykZP(){

        //readCustomCharacteristic();
        notificationManager=(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
        Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);
        intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
        PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(),0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
        NotificationCompat.Builder notificationBulder=
                new NotificationCompat.Builder(getApplicationContext(),CHANNELZU_ID)
                        .setAutoCancel(false)
                        .setSmallIcon(R.drawable.ic_chard_name)
                        .setWhen(System.currentTimeMillis())
```

```
                .setContentIntent(pendingIntent)
                .setContentTitle("Заряд пульсометра кончается")
                .setContentText("Заряда пульсометра скорее всего не хватит на всю ночь")
                .setPriority(PRIORITY_HIGH);
        createChannelIfNeeded(notificationManager);
        notificationManager.notify(NOTIFYZU_ID,notificationBulder.build());
    }
    public void notifykZT(){


        notificationManager=(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
        Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);
        intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
        PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(),0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
        NotificationCompat.Builder notificationBulder=
            new NotificationCompat.Builder(getApplicationContext(),CHANNELZT_ID)
                .setAutoCancel(false)
                .setSmallIcon(R.drawable.baseline_charging_station_24)
                .setWhen(System.currentTimeMillis())
                .setContentIntent(pendingIntent)
                .setContentTitle("Заряд телефона")
                .setContentText("Заряда телефона Скорее всего не хватит на всю ночь")
                .setPriority(PRIORITY_MAX);
        createChannelIfNeeded(notificationManager);
        notificationManager.notify(NOTIFYZT_ID,notificationBulder.build());
    }
    public void notifykO(){


        notificationManager=(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
        Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);
        intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
        PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(),0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
        NotificationCompat.Builder notificationBulder=
            new NotificationCompat.Builder(getApplicationContext(),CHANNELO_ID)
                .setAutoCancel(false)
                .setSmallIcon(R.drawable.ic_chard_name_disconection)
                .setWhen(System.currentTimeMillis())
                .setContentIntent(pendingIntent)
                .setContentTitle("Отключилось")
                .setContentText("Пульсометр отключился от телефона.\n Для полноценной работы
приложения подключите пульсометр обратно")
                .setPriority(PRIORITY_DEFAULT);
        createChannelIfNeeded(notificationManager);
        notificationManager.notify(NOTIFYO_ID,notificationBulder.build());
    }
    public void budilnik(){
        if (MainActivity.flagVkl){
```

```
          Date cutTime = Calendar.getInstance().getTime();
         if (heartRate>=60){
            if (heartRate<=80){
               if (priv>254){
            priv=0;
            if (MainActivity.day2<=cutTime.getDate()){
         if (MainActivity.hour2<=cutTime.getHours()){
            if (MainActivity.min2<=cutTime.getMinutes()){
               MainActivity.flagVklB=false;
               final Intent intent = new Intent(this,
                     AlarmActivity.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|Intent.FLAG_ACTIVITY_NEW_TASK);
               startActivity(intent);
            }
             }
            }
               }else {
                  priv=priv+1;
               }
            }else {
               priv=0;
            }
      }
      else {
            priv=0;
      }
      }
}

}
```