|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Караник Андрей Александрович

*фамилия, имя, отчество*

Группа ИУ9-62Б

Тип практики Производственная

Название предприятия ООО «ИЭГРУПП»

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Рекомендуемая оценка: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Руководитель практики   
от предприятия: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2024г.*

Оглавление

[Введение 3](#_Toc175665987)

[О предприятии 6](#_Toc175665988)

[Реализация 8](#_Toc175665989)

[Заключение 16](#_Toc175665990)

[Список использованных источников 16](#_Toc175665991)

# Введение

Задача состоит в том, чтобы изучить android-приложение MercuryScales и обновить в нем протокол работы весов, а также добавить возможность изменять язык интерфейса. Из-за ограничений в области конфиденциальности и коммерческой тайны информация о конкретных деталях проекта, в том числе протокола, будет изменена или опущена.

Весы передают по Bluetooth LE каждые 100мс следующую информацию в виде 8 байт:

* 1-2 байт – статус состояния (см. таблицу 1)
* 3-4 байт – время в миллисекундах между измерениями
* 5-8 байт – вес в 0.1 г

|  |  |
| --- | --- |
| Номер бита | Описание |
| 0 | Стабильность (если нет стабильности массы, расчет стоимости не производится, тара и ноль не берутся) |
| 1 | Ноль массы (готовность к новому взвешиванию) |
| 2 | Подается внешнее питание |
| 3 | Идет заряд аккумулятора |
| 4 | Перегрузка АЦП (возникает, если нарушается связь с датчиком, т.е. обычно неисправен кабель или разъем) |
| 5 | АЦП не отвечает на запросы (требуется ремонт платы BLE) |
| 6 | Калибровка не производилась ни разу. Требуется первичная калибровка. |
| 7 | Ошибка записи во Flash-память в КР(требуется ремонт платы BLE). |
| 8 | Требуется установить эталонную массу на платформу в КР. В разрядах веса в этот момент передается требуемое значение массы (для сведения). |
| 9 | Калибровочный режим (КР). В данном режиме производится тестирование датчика с помощью эталонной массы. Режим активируется специальным переключателем (КВ) на корпусе весов. Во время калибровки расчет стоимости не производится. |
| 10-15 | Резерв. |

Таблица 1. Статусы состояний.

После выключения питания микроконтроллер (МК) может продолжать работу до 5 – 10 минут на остаточном заряде. При повторном включении питания в указанном интервале времени МК продолжит работу без сброса. Для более быстрого разряда остаточного питания необходимо на некоторое время (несколько секунд) включить КВ, затем выключить.

Процедура калибровки

1. Выключить все. Отключить кабель зарядки (используется питание от батареи). Освободить платформу от груза. Включить КВ в положение «On».

2. Включить питание. Установить соединение с программой на Андроиде. Убедиться, что старший бит статуса установлен (КР – см. таблицу 1).

3. Дождаться, когда в разрядах веса появится значение калибровочной массы(10000,0). Выключить КВ и сразу, без задержки установить указанную массу на платформу.

4. Если старшие биты статуса сбросятся, а в разрядах массы младшие разряды начнут изменяться («живой» вес), калибровку можно считать законченной.

После обновления протокола необходимо добавить в раздел “Параметры работы” возможность изменять язык интерфейса.

# О предприятии

Компания Incotex - это крупнейший в России разработчик и производитель уникальной радиоэлектронной продукции, который лидирует в РФ по четырём направлениям деятельности, насчитывает более 2000 человек, в том числе более 200 человек в разработке.

Инкотекс производит более 800 типов разнообразной продукции и исключительно собственной разработки, которые создаются в R&D подразделениях:

* г. Москва
* г. Маркс, Саратовская обл.

Высокая квалификация разработчиков и программистов, общей численностью более 200 человек позволяет нам создавать конкурентную на мировом рынке продукцию и осуществлять экспорт в более чем 30 стран Мира, а также создавать продукцию в столь разных направлениях:

* счётчики электроэнергии (более 200 типов) и системы учёта;
* торговое оборудование (кассовые машины, принтеры, электронные весы);
* тахографы;
* LED освещение (более 200 типов) и лампы различного назначения и мощностей (до 2 кВт и более);
* светодиодные драйверы;
* автоматизированные системы управления освещением;

История Incotex начинается в 1989 г. с контрактной разработки различных радиоэлектронных устройств. Основу компании составляет авторитет в техническом мире и безупречная репутация ее создателя радиоинженера и изобретателя к.т.н. Юрия Соколова.

Инкотекс применяет более 2000 единиц новейшего высокопроизводительного оборудования, в том числе:

* 19 SMT линий общей производительностью более 1 млн. компонентов в час
* 8 линий селективной пайки
* 5 линий пайки волной
* 3 широкоформатных лазера по металлу
* 5 лазерных установок по пластмассе
* 3 автоматизированные и 2 ручные линии покраски металла и пластика
* 5 роботов нанесения клеевых составов и заливки
* 6 обрабатывающих центров по металлу
* 4 раскроечных автоматизированных станка по металлу
* 49 литьевых машин по пластмассе

а также множество другого оборудования для производства штамповки, гибки, резки металлов, гальваники, измерительно-испытательное, контрольное оборудование.

# Реализация

Android-приложение MercuryScales использует язык программирования Kotlin и архитектуру в соответствии с MVVM. Поскольку помимо протокола работы весов также была обновлена модель самих весов, то для начала нужно было изменить поиск весов в приложении. Для решения этой проблемы, была найдена константа, т.е. имя устройства, по которому осуществлялся этот поиск.



Рисунок 1. Константа со значением названия устройств

При сканировании устройств в методе обработки устройство не добавляется в список, если его имя не совпадает со значением константы.

Разбор данных по протоколу происходит в методе parseByProtocol.

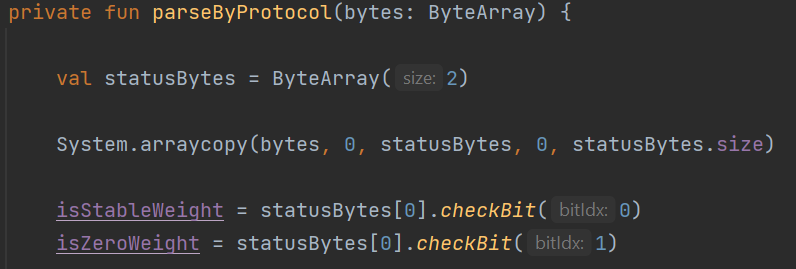


Рисунок 2. Фрагмент метода parseByProtocol.

Метод parseByProtocol принимает массив bytes, т.е. те самые 8 байт, которые весы передают по Bluetooth LE каждые 100мс. В нем создается новый массив байт statusBytes, состоящий из 2 элементов, который представляет собой статусы состояний весов. Для копирования фрагмента массива был использован метод arraycopy, который принимает: исходный массив, начальный индекс в исходном массиве, целевой массив, начальный индекс в целевом массиве, количество элементов для копирования. После копирования массива происходит обновление переменных, объявленных как тип Boolean в классе WorkspaceViewModel, означающие статусы состояний: isStableWeight, isZeroWeight и т.д. Для этого идет обращение к нужному байту массива statusBytes и у него вызывается метод checkBit с параметром индекса в байте, который возвращает true, если бит установлен, в противном случае – false.

В ходе разработки был использован MutableLiveData, класс в архитектурном компоненте Android, который используется для хранения и управления изменяемыми данными, которые могут быть наблюдаемыми. LiveData и его подкласс MutableLiveData часто используются в архитектуре MVVM для обновления пользовательского интерфейса в ответ на изменения данных в модели.

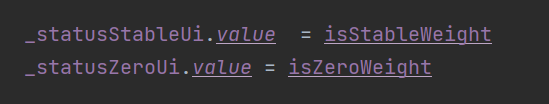


Рисунок 3. Обновление значений статусов состояний

Например, \_statusStableUi и \_statusZeroUi являются переменными типа MutableLiveData<Boolean>. Эти переменные обновляются на основе значений переменных isStableWeight и isZeroWeight. Свойство value используется для получения и установки текущего значения MutableLiveData. При изменении значения через value, все наблюдатели, подписанные на LiveData, автоматически уведомляются об изменении, что позволяет синхронизировать пользовательский интерфейс.

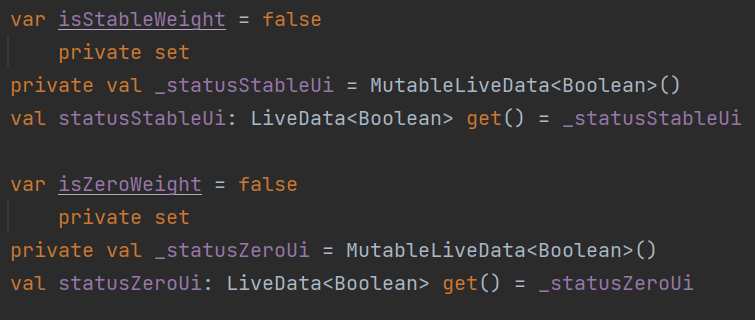


Рисунок 4. Фрагмент кода представления статусов состояний

Для получения значения веса из массива байтов использовался метод преобразования данных, включающий использование ByteBuffer. Сначала создавался массив байтов размером 4, и с помощью метода arraycopy из исходного массива байтов извлекались 4 байта, относящиеся к весу. Затем этот массив байтов оборачивался в объект ByteBuffer с помощью метода wrap. Устанавливался порядок байтов в ByteOrder.LITTLE\_ENDIAN, что соответствует порядку байтов, используемому в контексте протокола. Наконец, из буфера извлекалось целое число, которое преобразовывалось в строку для отображения веса.

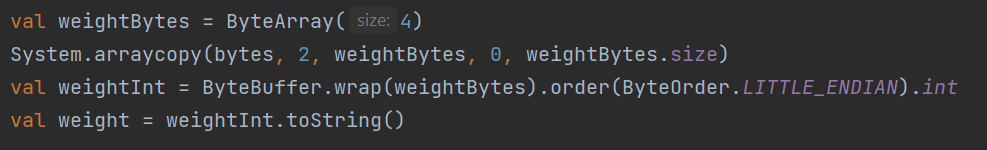


Рисунок 5. Получение значения веса

Также была произведена модификация кода, касающаяся работы с Bluetooth-устройствами, в частности, изменен UUID для характеристик весов, который используется для уникальной идентификации служб и характеристик Bluetooth-устройств, позволяющий точно определять нужную характеристику, обеспечивая корректное взаимодействие с устройством. В коде были обновлены методы, которые обрабатывают найденные характеристики Bluetooth и настраивают уведомления от весов. В методе handleFoundCharacteristic добавлена обработка новой UUID характеристики SampleGattAttributes.UUID\_CHARACTERISTIC\_DATA. Это изменение позволило корректно идентифицировать и обрабатывать уведомления от устройства весов. Теперь, когда характеристика с указанным UUID обнаруживается, соответствующий объект BluetoothGattCharacteristic присваивается переменной charaScalesNotif, и отправляется широковещательное сообщение об обработке этой характеристики.

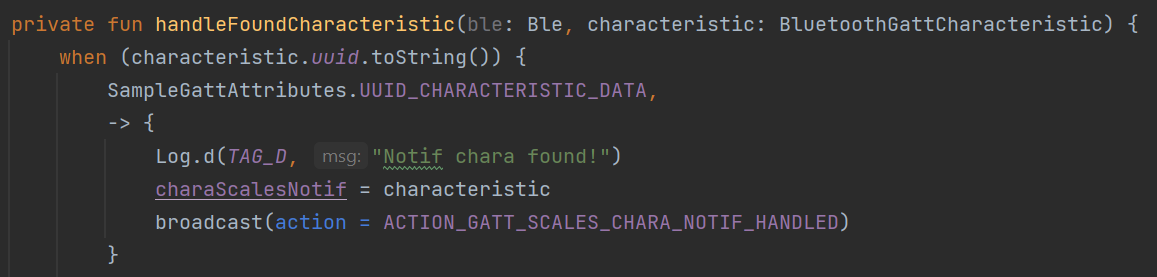


Рисунок 6. Фрагмент метода handleFoundCharacteristic

Кроме того, в методе startScalesNotif настроена активация уведомлений для новой характеристики. С помощью метода setCharacteristicNotification включены уведомления для обнаруженной характеристики, что позволяет корректно получать данные от весов и обрабатывать их в приложении.



Рисунок 7. Фрагмент метода startScalesNotif

Для изменения языка интерфейса приложения была разработана графическая часть, код которой представлен на рисунке 8:



Рисунок 8. Код графической части изменения языка интерфейса

Это позволяет пользователю выбирать предпочитаемый язык из выпадающего списка, обеспечивая удобную настройку локализации приложения.

Рассмотрим данный код подробнее:

* LinearLayout — это контейнер для размещения других UI-элементов. В данном случае, это TextView и Spinner. Контейнер располагает их горизонтально или вертикально в зависимости от ориентации (по умолчанию - горизонтально).
  + android:id="@+id/languageContainer" — присваивает уникальный идентификатор для LinearLayout, что позволяет обращаться к нему программно.
  + android:layout\_width="wrap\_content" и android:layout\_height="wrap\_content" — указывают, что контейнер будет занимать столько места, сколько необходимо для отображения его содержимого.
  + android:layout\_marginTop="16dp" — добавляет верхний отступ в 16dp, чтобы создать пространство между этим контейнером и элементом вышеConstraint Layoutint Layout (ограничения макета):
  + app:layout\_constraintBottom\_toBottomOf="parent" — привязывает нижнюю часть LinearLayout к нижней части родительского элемента.
  + app:layout\_constraintEnd\_toEndOf="parent" — привязывает правую сторону LinearLayout к правой стороне родительского элемента.
  + app:layout\_constraintStart\_toStartOf="parent" — привязывает левую сторону LinearLayout к левой стороне родительского элемента.
  + app:layout\_constraintTop\_toBottomOf="@+id/btnChangePass" — привязывает верхнюю часть LinearLayout к нижней части элемента с идентификатором btnChangePass.
* TextView — это элемент, который отображает текст "Выбранный язык" или другой текст, соответствующий строковому ресурсу @string/settings\_selected\_language.
  + android:layout\_width="wrap\_content" и android:layout\_height="wrap\_content" — указывают, что TextView будет занимать столько места, сколько необходимо для отображения его содержимого.
  + android:layout\_gravity="left" — задает выравнивание текста по левому краю внутри LinearLayout.
  + android:paddingEnd="24dp" — добавляет отступ в 24dp с правой стороны, создавая пространство между TextView и следующим элементом (Spinner).
  + android:textColor="@color/Black" — устанавливает черный цвет текста.
  + android:textSize="24sp" — определяет размер текста в 24sp, делая его достаточно крупным для удобного чтения.
* Spinner — это выпадающий список, который позволяет пользователю выбирать язык интерфейса.
  + android:id="@+id/languageSpinner" — присваивает уникальный идентификатор элементу, чтобы можно было программно управлять его содержимым и действиями.
  + android:layout\_width="match\_parent" — устанавливает ширину Spinner так, чтобы он занимал всю доступную ширину контейнера (LinearLayout).
  + android:layout\_height="wrap\_content" — автоматически подстраивает высоту элемента под его содержимое.
  + android:background="@android:drawable/btn\_dropdown" — задает фоновое изображение, которое придает Spinner внешний вид стандартного выпадающего списка.
  + android:spinnerMode="dropdown" — указывает, что список будет отображаться в виде выпадающего меню, где пользователь может выбрать один из предложенных языков.

В проекте для управления текстовым содержимым использовались строковые ресурсы. Это позволяет централизованно хранить текстовые данные, что упрощает их изменение и адаптацию для различных языков. В частности, для каждого текстового элемента в приложении были заданы строковые ресурсы, которые позволяют легко переключаться между языковыми версиями, такими как русский и английский.

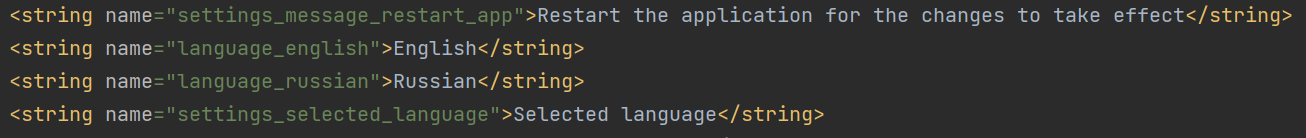


Рисунок 9. Фрагмент файла strings.xml для английского языка в пакете values

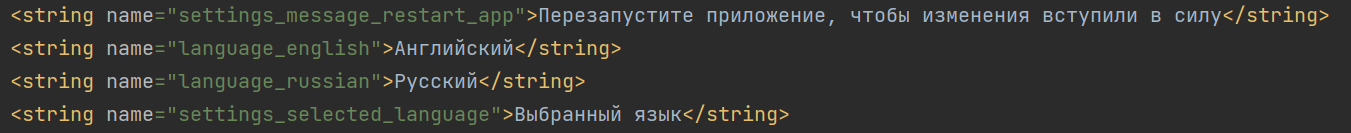


Рисунок 10. Фрагмент файла strings.xml для русского языка в пакете values-ru

Для реализации обработки изменения языка интерфейса через выпадающий список был использован следующий подход. Сначала был создан массив, содержащий названия доступных языков, которые извлекаются из строковых ресурсов, что позволяет адаптировать их в зависимости от локализации приложения. Затем был настроен адаптер для элемента Spinner, который связывает этот массив с интерфейсом и обеспечивает корректное отображение списка языков.

После этого был установлен слушатель событий для Spinner, который реагирует на выбор пользователя. В случае, если выбранный язык отличается от текущего, происходит изменение локали приложения с помощью метода LocaleHelper.setLocale(), который переключает язык на английский или русский. После смены языка пользователю отображается уведомление о необходимости перезапуска приложения, чтобы изменения вступили в силу.

Также была реализована логика для автоматической установки текущего языка при запуске экрана. В зависимости от установленной локали, Spinner автоматически отображает соответствующий выбранный язык. Такой подход обеспечивает гибкость в управлении локализацией и улучшает пользовательский опыт, позволяя легко переключаться между языками в приложении.

****

Рисунок 11. Фрагмент кода обработки изменения языка интерфейса

# Заключение

В ходе практики я получил ценный опыт работы с готовым приложением, что позволило мне глубже погрузиться в особенности разработки для Android и улучшить навыки использования языка программирования Kotlin. Одной из ключевых задач было изменение протокола работы весов, что требовало глубокого понимания работы с данными и взаимодействия с аппаратным обеспечением. Этот процесс включал в себя работу с байтовыми массивами и их преобразование, что позволило мне улучшить навыки работы с низкоуровневыми данными. Кроме того, я внедрил функционал, позволяющий пользователю изменять язык интерфейса. Это улучшение повысило гибкость приложения и обеспечило лучшую доступность для пользователей, говорящих на разных языках. Опыт, приобретённый в ходе выполнения этих задач, значительно углубил мои знания в области разработки Android-приложений и работы с Kotlin, а также позволил развить навыки работы с протоколами и интерфейсами, что является важным аспектом в разработке современных программных решений.

# Список использованных источников

1. <https://www.incotex.ru/about/>
2. <https://developer.android.com/develop/>