Санкт-Петербургский
Государственный Университет

Математико-Механический факультет
Кафедра Системного Программирования

Мирошников Владислав Игоревич

Исследование энергопотребления модулей Wi-Fi и Bluetooth и их интеграция в Navitas Framework

Отчёт по учебной практике

Научный руководитель:
ст. преп. С. Ю. Сартасов

Санкт-Петербург
2021

Содержание

Введение

Наше время невозможно представить без мобильных устройств. Ввиду высокой портативности и универсальности, смартфоны, планшеты и умные часы всё активнее распространяются в мире. В 2020 г. уже насчитывается более 3.5 млрд. пользователей смартфонов, а в 2021 г. ожидается, что эта цифра превысит 3.8 млрд. человек [1]. На сегодняшний день наиболее распространенной операционной системой для мобильных устройств является Android [2].

Около 2 млрд. пользователей смартфонов, что составляет половину от общего числа, ежедневно используют Wi-Fi для выхода в сеть Интернет, скачивания и выгрузки данных [3], а также многие пользователи мобильных устройств активно используют Bluetooth для беспроводного прослушивания аудио, передачи данных между устройствами [4]. В связи с этим вопрос об оценке и оптимизации энергопотребления данных модулей устройств является актуальным, ведь от этого показателя напрямую зависит время их работы. Многие компании, такие как Google, Bluetooth SIG и другие, работают над снижением энергопотребления данных модулей. Так, к 2025 г. ожидается, что 90% всех устройств, поддерживающих Bluetooth, будут использовать технологию Bluetooth Low Energy [4].

Одним из существующих проектов по оценке энергопотребления различных модулей устройств на базе Android является Navitas Framework, разработанный студентами кафедры Системного Программирования СПбГУ [5]. На данный момент Navitas Framework поддерживает оценку энергопотребления CPU и дисплея, проводит анализ собранных данных и визуализирует результаты энергопрофилирования.

Помимо уже имеющейся функциональности, предлагается расширить возможности профилирования путем сбора и анализа информации об энергопотреблении модулей Wi-Fi и Bluetooth, а также их интеграции в Navitas Framework. В данной работе рассматриваются различные способы получения энергопотребления данных модулей, а также предлагается реализация одного из них в рамках проекта Navitas Framework.

1. Цели и задачи

Целью данной работы является добавление поддержки модулей Wi-Fi и Bluetooth в энергопрофилировщик Navitas Framework. Для достижения цели были поставлены следующие задачи.

1. Провести обзор существующих энергопрофилировщиков.
2. Провести обзор методов получения информации об энергопотреблении и выбрать метод для внедрения в Navitas Framework.
3. Проанализировать результаты, полученные выбранным методом, и оценить их корректность.
4. Реализовать выбранный метод и интегрировать его в Navitas Framework.
5. Провести работу над улучшением существующей функциональности Navitas Framework, повышением стабильности и увеличением числа поддерживаемых устройств.

2. Термины и определения

- **Power Profile** (**power_profile.xml**) — XML-файл с константными значениями энергопотребления компонентов конкретного смартфона, предоставляющиеся производителями в определенной директории устройства. В противном случае, содержимое файла инициализируется значениями по умолчанию от Google [6]. В профиле мощность указывается в миллиамперах (mA) потребляемого тока при номинальном напряжении.

- **Android Debug Bridge** (**adb**) — универсальный инструмент командной строки, позволяющий взаимодействовать с мобильным устройством и предоставляющий доступ к оболочке UNIX, которую можно использовать для запуска различных команд на устройстве [7].

- **UID приложения** — уникальный идентификатор, который платформа Android назначает каждому приложению. Android использует UID для создания изолированной программной среды приложения на уровне ядра [8].

- **Права суперпользователя (root-права)** — специальный аккаунт и группа пользователей в UNIX-подобных системах, владелец которого имеет право на выполнение всех без исключения операций.

3. Обзор предметной области

3.1. Обзор энергопрофилировщиков

3.1.1. PowerTutor

PowerTutor — отдельное мобильное приложение, показывающее информацию о текущем потреблении энергии смартфоном [9]. Для работы с модулем Wi-Fi оно использует библиотеку `android.net.wifi.WifiManager` [10]. Однако информации об энергопотреблении модуля Bluetooth данное приложение не получает.

PowerTutor для каждого компонента сопоставляет переменную статуса, которая характеризует возможные состояния модулей устройств. Если сложить их и умножить на определенные коэффициенты, полученные в результате запуска тестовых сценариев на данном устройстве, можно получить приблизительный расход энергии.

Технические ограничения: для устройств под управлением Android 7.0 (Android API Level 24) и выше доступ к файлам, откуда достается информация о потреблении энергии модулем Wi-Fi [11], невозможен без получения прав суперпользователя, что значительно усложняет такой способ получения значений энергопотребления и делает его пригодным для применения только на заранее подготовленных устройствах.

HTC Dream Power Model

Model	$(\beta_{uh} \times freq_h + \beta_{ul} \times freq_l) \times util + \beta_{CPU} \times CPU\_on + \beta_{br} \times brightness + \beta_{Gon} \times GPS\_on + \beta_{Gsl} \times GPS\_sl + \beta_{Wi\text{-}Fi\_l} \times Wi\text{-}Fi_l + \beta_{Wi\text{-}Fi\_h} \times Wi\text{-}Fi_h + \beta_{3G\_idle} \times 3G_{idle} + \beta_{3G\_FACH} \times 3G_{FACH} + \beta_{3G\_DCH} \times 3G_{DCH}$						
Category	System variable	Range	Power coefficient	Category	System variable	Range	Power coefficient
CPU	util	1–100	β_{uh}: 4.34 β_{ul}: 3.42	LCD	brightness	0–255	β_{br}: 2.40
	$freq_l$,$freq_h$	0,1	n.a.	GPS	GPS_on	0,1	β_{Gon}: 429.55
	CPU_on	0,1	β_{CPU}: 121.46		GPS_sl	0,1	β_{Gsl}: 173.55
Wi-Fi	npackets, R_{data}	0–∞	n.a.	Cellular	data_rate	0–∞	n.a.
	$R_{channel}$	1–54	β_{cr}		downlink_queue	0–∞	n.a.
	$Wi\text{-}Fi_l$	0,1	$\beta_{Wi\text{-}Fi\_l}$: 20		uplink_queue	0–∞	n.a.
	$Wi\text{-}Fi_h$	0,1	$\beta_{Wi\text{-}Fi\_h}$: Equation 1		$3G_{idle}$	0,1	β_{3G_idle}: 10
Audio	Audio_on	0,1	β_{audio}: 384.62		$3G_{FACH}$	0,1	β_{3G_FACH}: 401
					$3G_{DCH}$	0,1	β_{3G_DCH}: 570

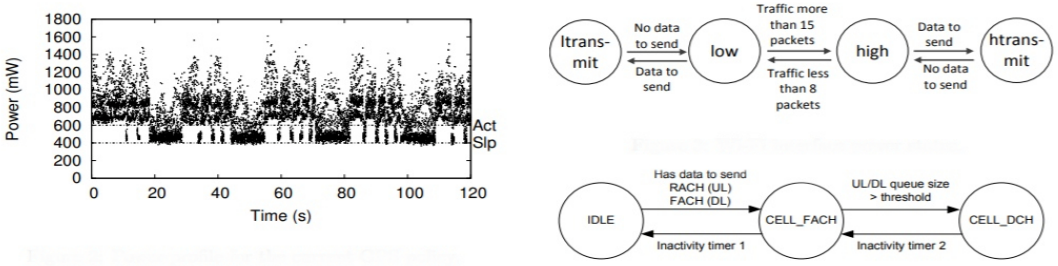

Рис. 1: Модель энергопотребления устройства в PowerTutor [12]

3.1.2. Energy Profiler

Energy Profiler — встроенный в Android Studio плагин-профилировщик.

Energy Profiler контролирует использование CPU, мобильной сети, Wi-Fi, Bluetooth, датчика GPS и отображает визуализацию того, сколько энергии потребляет каждый из этих компонентов [13]. Energy Profiler также показывает наличие системных событий (wakelocks, alarms, jobs и location requests), которые могут повлиять на потребление энергии.

Energy Profiler не измеряет потребление энергии напрямую, а использует модель, которая его оценивает для каждого ресурса на устройстве. Информация о потреблении тока в реальном времени сопоставляется с программным счётчиком.

Технические ограничения: Android 8.0 (Android API Level 26) и выше.

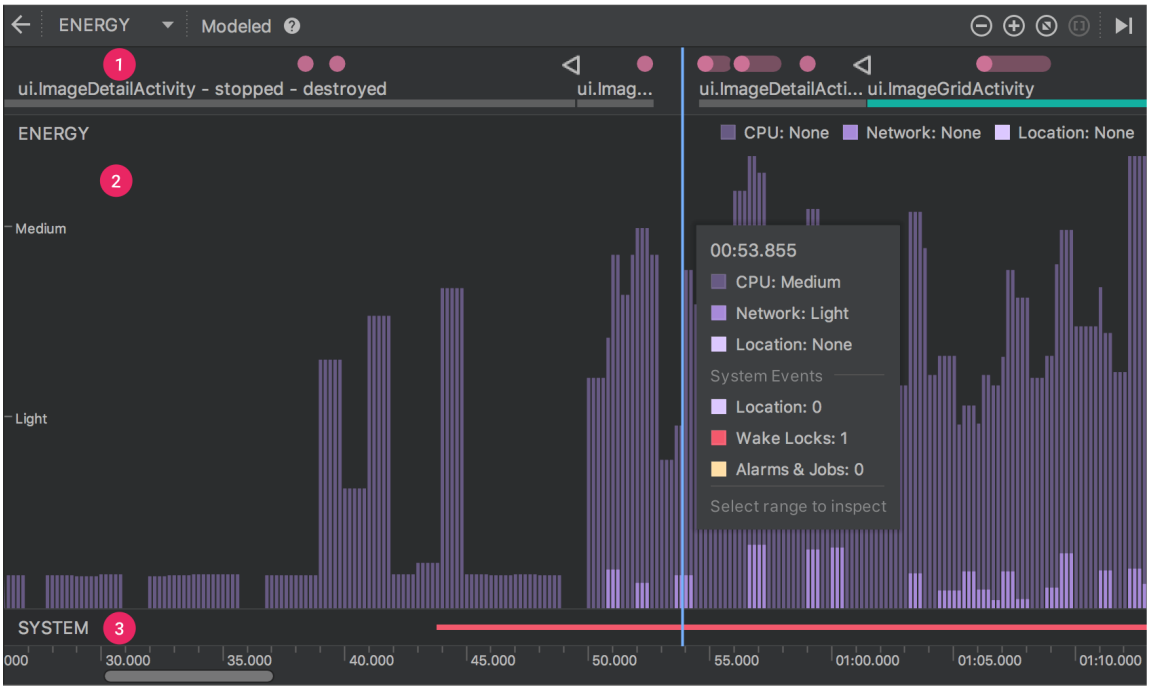

Рис. 2: Визуализация энергопотребления в Energy Profiler

3.1.3. Battery Historian

Battery Historian — инструмент компании Google, разработанный для получения информации о разряде батареи устройства с течением времени [14].

На общесистемном уровне инструмент визуализирует связанные с питанием события из системных журналов в формате HTML. На уровне конкретного приложения инструмент предоставляет различные данные, которые могут помочь определить поведение приложения, расходующего заряд батареи.

Стоит отметить, что профилировщик собирает данные, непосредственно используя команду `adb shell dumpsys batterystats`, при этом используя для запуска Docker и .zip-отчёт, который получается командой `adb bugreport [path/] bugreport.zip`.

Диаграмма, отображающая выводимую Battery Historian информацию, разделена по категориям, в которых отображается информация об активности соответствующего модуля в течение определенного времени, как показано на абсциссе диаграммы. На ординате диаграммы каждая строка показывает сегмент цветной полосы во времени, когда компонент системы активен и, таким образом, потреблял ток от батареи [15].

Однако диаграмма не показывает, сколько энергии было использовано компонентом, а только то, в течение какого времени приложение было активным.

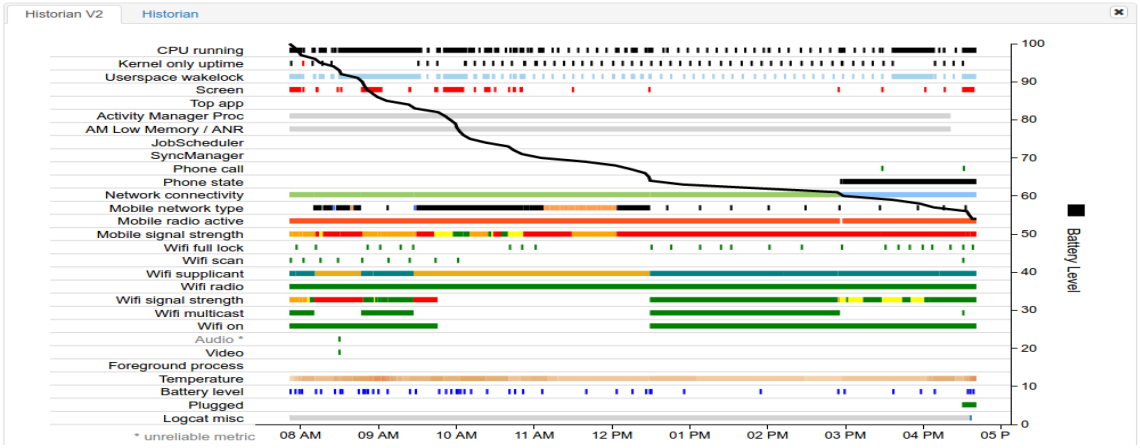

Рис. 3: Отображение в Battery Historian общесистемных событий, влияющих на энергопотребление

Можно также просмотреть дополнительную информацию из соответствующего файла `batterystats.txt` в разделе статистики **System Stats** под диаграммой, показанной в Battery Historian. Именно из этого файла, который был

получен командой `adb shell dumpsys batterystats`, Battery Historian берёт и анализирует всю информацию.

Стоит заметить, что данные в нём отсчитываются с последнего сброса статистики энергопотребления в данном файле.

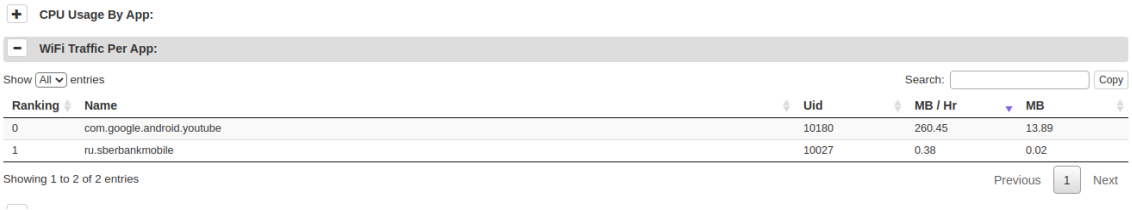

Рис. 4: Информация об использовании трафика приложениями

Таким образом, здесь выводится более подробная информация о модуле Wi-Fi, в которой имеется:

- UID приложения, использующего или когда-либо использовавшего данный модуль;
- скорость потребления в мегабайт/час для каждого приложения;
- количество мегабайт, потребленных тем или иным приложением.

Однако информации, напрямую отражающей потребление энергии каким-либо модулем или приложением, взаимодействующим с этим модулем, здесь нет.

Необходимо отметить, что, несмотря на наличие информации в Battery Historian о времени работы модуля Bluetooth в различных состояниях, какой-либо информации о данном модуле в разделе **System Stats** нет.

Технические ограничения: Android 5.0 (Android API Level 21) и выше.

3.2. Профили питания Android

В соответствии с документацией платформы Android [16], информация об использовании батареи получается из статистики использования батареи и значений профиля мощности [17].

Платформа Android автоматически определяет статистику использования батареи, отслеживая, как долго компоненты устройства находятся в разных состояниях. Когда компоненты (набор микросхем Wi-Fi, мобильная связь, Bluetooth, GPS, дисплей, CPU) меняют состояния (вкл. / выкл., режим ожидания / полная мощность, низкая / высокая яркость и другие), управляющая служба сообщает об этом службе фреймворка BatteryStats. BatteryStats собирает информацию с течением времени и сохраняет её для использования при перезагрузках. Служба не отслеживает потребление тока батареей напрямую, а вместо этого собирает информацию о времени, которую можно использовать для приблизительного расчета потребления батареи различными компонентами.

Во избежание потери статистики использования в случае выключения устройства (то есть батарея достигла нулевой оставшейся емкости), фреймворк отображает статистику примерно каждые 30 минут.

В соответствующем Power Profile файле устройства находятся следующие параметры Wi-Fi:

- `wifi.on` — данное значение показывает мощность, потребляемую, когда модуль Wi-Fi включен, но не осуществляет передачу или прием данных. По рекомендациям Google, данное значение можно измерить разницей между текущим потреблением энергии в состоянии приостановки (сна) системы с включенным и отключенным Wi-Fi;

- `wifi.scan` — данное значение показывает мощность, потребляемую во время сканирования точек доступа при помощи Wi-Fi;

- `wifi.active` — данное значение показывает мощность, используемую при передаче или приеме данных по Wi-Fi.

Также в этом файле есть информация о параметрах модуля Bluetooth:

- `bluetooth.on` — данное значение показывает мощность, когда модуль Bluetooth включен и находится в режиме поиска доступных устройств, но при этом не подключен к какому-либо устройству;

- `bluetooth.active` — данное значение показывает мощность, когда происходит передача данных по Bluetooth.

3.3. Методы получения информации об энергопотреблении

Для поиска существующих алгоритмов использовался сервис «Google Scholar». Были произведены следующие запросы: «Android energy measurement», «Getting information about energy consumption of Android modules», «Measuring the energy consumption of Wi-Fi and Bluetooth modules in Android», «Methods to derive energy profiles for Android Platform». По каждому из запросов были изучены первые 2-3 страницы выдачи поисковой системы. Все соответствующие статьи были рассмотрены на предмет соответствия данной предметной области в рамках системного обзора литературы. В результате были отобраны следующие статьи, вынесенные в отдельную таблицу [18].

Из рассмотренных статей можно выделить следующие методы, с помощью которых можно получить информацию об энергопотреблении:

- **внешнее измерительное устройство** — используется в широком спектре работ в данной предметной области. К примеру, в работе Андрея Саксонова «Method to Derive Energy Profiles for Android Platform» [19] для измерения энергопотребления использовался амперметр Yoctopuce Yocto-Amp USB Electrical Sensor [20], который можно запрограммировать на исполнение определённого тестового сценария. Также можно использовать иные амперметры или более функциональные мультиметры;

- **построение модели энергопотребления устройства с помощью устаревших системных вызовов** — по аналогии с API PowerTutor [11], информацию об энергопотреблении модулей в различных состояниях можно получить из соответствующих файлов. После чего, путём сопоставления измеренной потребляемой мощности с этими состояниями, строится модель энергопотребления устройства, что подробно описано в статье [12]. Как уже утверждалось ранее в разделе 3.1.1. при рассмотрении энергопрофилировщика PowerTutor, для устройств под управлением Android 7.0 (Android API Level 24) и выше доступ к файлам, откуда достается информация о потреблении энергии модулем Wi-Fi, невозможен без получения прав суперпользователя, что является важным недостатком данного метода;

- **Android Debug Bridge (adb)** — API **adb** предоставляет множество команд для получения подробной информации об устройстве, среди которых есть и те, что выдают информацию о модулях Wi-Fi и Bluetooth.

При изучении этих трех методов было принято решение исключить из рассмотрения метод **построения модели энергопотребления устройства при помощи устаревших системных вызовов** по следующей ключевой причине: проект Navitas Framework не требует обязательного наличия на устройствах прав суперпользователя, а также ориентирован на устройства под управлением Android 7.0 и выше. Это фактически означает, что внедрение данного метода получения информации об энергопотреблении в Navitas Framework приведёт к изменению его требований к устройствам в сторону наличия прав суперпользователя, что повлечёт уменьшение числа поддерживаемых устройств.

При рассмотрении метода с применением **внешнего измерительного устройства**, был замечен следующий определяющий недостаток: использование внешнего измерительного прибора для получения данных об энергопотреблении требует непосредственного физического вмешательства в работу устройства. Во-первых, это бы накладывало со стороны Navitas Framework требование в виде наличия подходящих специализированных устройств. Во-вторых, многие современные устройства не позволяют получить доступ к батарее без дополнительных, зачастую требующих специальной подготовки, действий.

Однако стоит отметить, что такой метод позволяет экспериментально получить текущие значения энергопотребления устройства. За счёт этой важной особенности, подход с применением внешнего измерительного устройства в дальнейшем возможно будет использоваться для проверки корректности данных, получаемых выбранным в соответствии с разделом 1. методом для внедрения в Navitas Framework.

Таким образом, было принято решение взять за основу метод с использованием **Android Debug Bridge**, по нескольким причинам.

1. Некоторые команды **adb** уже используются в Navitas Framework для получения информации об энергопотреблении других компонентов устройств.

2. Применение такого метода не влечёт изменения требований к устройствам со стороны Navitas Framework.

3. Данный метод позволяет достичь цели, описанной в разделе 1., поскольку поддерживает получение информации о модулях Wi-Fi и Bluetooth.

4. Исследование Android Debug Bridge (adb)

4.1. Получение информации об энергопотреблении

В данном разделе будут рассмотрены команды **adb** и информация, которую они аккумулируют и отображают.

1. `adb shell dumpsys wifi`

   Данная команда позволяет получить расширенную информацию об анализе сети Wi-Fi. Она выдает историю подключений устройства к Wi-Fi за всё время, а также информацию, которая недоступна некоторым приложениям, например, когда устройство в последний раз подключалось или отключалось от Wi-Fi. Главное, на что стоит обратить внимание, — информационный модуль **Wi-Fi Power Metrics**, в поле **Energy consumed by wifi** которого и выводится информация об энергопотреблении в **mAh**.

   По результатам вывода данной команды было принято решение провести её тестирование на двух устройствах для исследования поведения в зависимости от влияния внешних факторов. В данном эксперименте не уточняются значения, отображаемые данной командой, а исследуется их поведение в зависимости от влияния внешних факторов.

   (a) **Samsung Galaxy S9+ (Android API Level 28)**

   Первое измерение потребления:

   Energy consumed by wifi (mAh) = 12.363808.

   Затем телефон использовался с включенным модулем Wi-Fi в режиме воспроизведения аудиофайлов онлайн в течение 30 сек. Зафиксируем новый результат:

   Energy consumed by wifi (mAh) = 12.471022.

   Затем возник вопрос, с какого конкретно момента идёт отсчёт энергопотребления. Телефон был отключен от сети Wi-Fi, подключен снова, и после повторного воспроизведения аудиофайлов в течение 15 сек. Получен следующий результат:

   Energy consumed by wifi (mAh) = 12.557738.

   Нетрудно заметить, что значение не было сброшено. Следовательно, данное значение измеряется не с момента включения модуля Wi-Fi/подключения к сети Wi-Fi.

Затем телефон был подключен к другой сети Wi-Fi, снова проведено активное использование модуля. Результат потребления в **mAh** вырос, но отсчет был аналогично предыдущему продолжен от уже имеющегося значения. Следовательно, данное значение измеряется не для конкретной сети Wi-Fi.

Затем телефон был перезагружен и было аналогично проведено использование модуля Wi-Fi, однако результат также немного увеличился, но так и не был посчитан от нуля. Следовательно, данное значение измеряется не с момента включения устройства.

Таким образом, возникла гипотеза: учёт энергопотребления обнуляется с момента зарядки устройства до 100%.

Затем устройство было заряжено до 100% и был проведен аналогичный эксперимент с вызовом данной команды. В данном поле был обнаружен следующий результат:

Energy consumed by wifi (mAh) = 0.101896.

Таким образом, энергопотребление сбросилось. Следовательно, гипотеза подтверждена.

После чего был выполнен поиск команды для **adb**, при помощи которой можно сбрасывать статистику энергопотребления. Его результатом стала команда, устанавливающая новую точку отсчёта **RESET**, — `adb shell dumpsys batterystats -reset`.

(b) **Samsung Galaxy S5 (Android API Level 23)**

При запуске данной команды поля **Wi-Fi Power Metrics** не было обнаружено, что говорит о том, что в данной версии API отображение данной информации не поддерживается.

2. `adb shell dumpsys bluetooth_manager`

   Данная команда была также протестирована на двух устройствах: **Samsung Galaxy S9+ (Android API Level 28)** и **Samsung Galaxy S5 (Android API Level 23)**. На обоих устройствах команда выдает информацию о состоянии модуля, имени устройства, о процессах, использующих модуль Bluetooth, но информации об энергопотреблении обнаружено не было.

3. `adb shell dumpsys batterystats`

   Данная команда выводит информацию об энергопотреблении модулей в контексте каждого процесса. В том числе, присутствует информация о том,

сколько потребил энергии соответствующий модуль, в течение какого времени он работал, а также некоторые другие данные с отчетом от точки **RESET**.

Внимание стоит обратить на модуль **Statistics since last charge**. В нем содержится информация об энергии, потраченной во время работы Wi-Fi. Стоит отметить, что значение **Wi-Fi Battery drain** совпадает со значением **Energy consumed by wifi (mAh)** из команды `adb shell dumpsys wifi`. Соответственно, все результаты эксперимента, описанного в 1 пункте, применимы и к данной команде.

На тестовом устройстве **Samsung Galaxy S5 (Android API Level 23)** был получен следующий результат: значение поля **Wifi Battery drain** здесь всегда нулевое, что в ещё раз подтверждает, что в данной версии API вывод данной информации не поддерживается.

Ключевой особенностью данной команды является то, что на обоих телефонах выводится информация об энергопотреблении модуля Bluetooth, в отличие от результатов команды `adb shell dumpsys bluetooth_manager`, описанной во 2 пункте.

Следующий важный раздел, который выводится данной командой, — **Estimated power use (mAh)**. В нём выводится широкий перечень значений энергопотребления, начиная от дисплея и заканчивая информацией о каждом отдельном процессе с точностью до того, сколько в контексте его работы потребляет CPU, Sensor и многие другие модули, включая Wi-Fi и Bluetooth. Все данные здесь также отсчитываются от точки **RESET**.

Всю информацию, получаемую данной командой, можно преобразовать в текстовый файл, используя команду `adb shell dumpsys batterystats > [path/]batterystats.txt`, который необходим для дальнейшего анализа результатов данной команды и преобразования их в **JSON-файл**, использующийся в Navitas Plugin [21] для визуализации получаемых данных.

4.2. Анализ данных энергопотребления Wi-Fi и Bluetooth

По результатам, полученными в разделе 4.1., было принято решение провести анализ данных энергопотребления модулей Wi-Fi и Bluetooth, полученных при помощи команды `adb shell dumpsys batterystats`.

Для достижения этой цели, были поставлены следующие промежуточные задачи.

1. Проверить «чистоту» нулевого энергопотребления, показывающую, потребляют ли данные модули какую-либо энергию в выключенном состоянии.

2. Провести тестирование энергопотребления модулей Wi-Fi и Bluetooth в разных режимах работы командой `adb shell dumpsys batterystats` на протяжении фиксированного временного промежутка с различной частотой дискретизации.

3. По полученным данным для каждой частоты дискретизации построить соответствующие линии регрессии, выражающие наилучшее предсказание зависимой переменной (Y) по независимым переменным (X), а также доверительные интервалы, показывающие оценку с заданной надёжностью 95%.

4.2.1. Ход экспериментальной проверки «чистоты» нулевого энергопотребления

Для выполнения данной задачи был написан соответствующий bash-скрипт [24], отключающий модули Wi-Fi и Bluetooth, сбрасывающий информацию батареи, затем ожидающий в течение 30 секунд и, наконец, собирающий информацию об энергопотреблении. Далее анализируются результаты, полученные командой `adb shell dumpsys batterystats` на двух тестируемых устройствах.

(a) **Samsung Galaxy S5 (Android API Level 23)**

На данном устройстве в полях **Wi-Fi power drain** и **Bluetooth power drain** мы видим ноль:

WiFi Battery drain: 0 mAh,

Bluetooth Battery drain: 0 mAh.

(b) **Samsung Galaxy S9+ (Android API Level 28)**

На данном устройстве поля **Wi-Fi Battery drain** и **Bluetooth Battery drain** вовсе отсутствуют после сброса статистики. Так, данные поля не появляются до тех пор, пока не появится потребление Wi-Fi или Bluetooth, что было проверено их последующим включением. Однако в данном тесте эти модули отключены, поэтому они так и не появились. Отсюда получается следующий вывод: отсутствие данных полей до начала потребления энергии модулями эквивалентно нулю.

Необходимо отметить, что среди полученных результатов присутствует информационный модуль **Estimated power use (mAh)**. Как уже рассказывалось в разделе 4.1., в нём отображаются значения энергопотребления Wi-Fi и Bluetooth в контексте каждого процесса совместно со множеством других компонентов устройства. Значения в полях здесь, соответствующих интересующим нас модулям, полностью соответствуют значениям в **Wi-Fi Battery drain** и **Bluetooth Battery drain**.

На обоих тестируемых устройствах в данном разделе вывода команды `adb shell dumpsys batterystats` отсутствует информация о потреблении обоих анализируемых модулей. То есть текущая информация содержит только энергопотребление не интересующих нас компонентов, что еще раз подтверждает, что в этом случае она равна нулю:

Wifi: 0.000645 (cpu=0.000645),

Bluetooth: 0.000967 (cpu=0.000967).

С включенными модулями Wi-Fi и Bluetooth данные поля будут выглядеть примерно следующим образом:

Wifi: 0.00228 (cpu=0.000645 wifi=0.00164), где значение только интересующего нас модуля **wifi=0.00164**,

Bluetooth: 0.001947 (cpu=0.000967 bt=0.00098), где значение только интересующего нас модуля **bt=0.00098**.

4.2.2. Ход экспериментальной проверки энергопотребления в разных режимах работы на протяжении фиксированного временного промежутка с различной частотой дискретизации

Для выполнения данной задачи был написан соответствующий bash-скрипт [24]. Все тестовые запуски проводились на устройстве **Samsung Galaxy S9+ (Android API Level 28)**. В эксперименте была обеспечена следующая изоляция для модулей Wi-Fi и Bluetooth.

1. Перевод всех приложений, кроме непосредственно задействованных в эксперименте, в режим (глубокого) сна, запрещающего любую фоновую дея-

тельность и отправку уведомлений. Ввиду отсутствия команд, позволяющих осуществить это без прав суперпользователя, подготовка выполнялась вручную.

2. Включение авиарежима, который отключает мобильную связь. Данное действие производится вручную, так как команду, не требующую прав суперпользователя для текущего уровня API, найти не удалось; а также до начала начала тестирования, так как его включение приводит к отключению тестируемых модулей Wi-Fi и Bluetooth.

3. Отключение зарядки устройства с помощью команды `adb shell dumpsys battery unplug`.

4. Включение только одного модуля, тестируемого в текущий момент, для снижения погрешности в замерах энергопотребления, связанной с влиянием компонентов друг на друга. Данное действие совершается при помощи команды `adb shell svc [module] enable`.

5. Отключение экрана устройства с помощью команды `adb shell input keyevent 26`.

Эксперимент проходит по следующей схеме: результаты измерений фиксируются в каждый момент времени, соответствующий текущей частоте дискретизации, и выгружаются в отдельный текстовый файл при помощи команды `adb shell dumpsys batterystats`. Полученный файл разбирается таким образом, что из его полей **Wi-Fi Battery drain** и **Bluetooth Battery drain** извлекается только значение энергопотребления для тестируемого в данный момент компонента. В выходной файл эксперимента записывается пара **(время замера, значение энергопотребления данного модуля)**. Таким образом проводится многократное независимое тестирование модулей Wi-Fi и Bluetooth при разных частотах дискретизации в течение **60 секунд**.

В приведённой ниже таблице показано, с какими частотами дискретизации было проведено тестирование обоих анализируемых модулей, а также число замеров, произведённых на соответствующей частоте за указанное время.

Таблица 1: Частоты дискретизации и количество замеров за 60 секунд

Частота дискретизации (Гц)	**Количество замеров (шт.)**
4	240
2	120
1.33	80
1	60
0.66	40
0.5	30
0.4	24
0.33	20
0.28	17
0.25	15
0.22	13
0.2	12
0.16	10
0.14	8
0.12	7
0.11	6
0.1	6

1. **Тестирование модуля Wi-Fi в режиме** `wifi.active`

   На устройстве был включён модуль Wi-Fi, оно было подключено к сети, а затем было проведено тестирование энергопотребления в режиме проигрывания трансляции на YouTube с разрешением 1080p и включённым экраном, к чему пришлось прибегнуть для поддержания воспроизведения видео. Никакой дополнительной задержки перед началом тестирования выдерживать нет необходимости, так как сканирование доступных сетей Wi-Fi происходит до подключения к ним, и поэтому не могло отразиться в данном эксперименте.

   По результатам тестирования на примере частот дискретизации 4 Гц, 2 Гц, 1 Гц, 0.5 Гц и 0.1 Гц были построены следующие графики с применением регрессионного анализа, а также доверительные интервалы к ним. Для частот, не попавших в данное перечисление, было выполнено аналогичное построение. Полученные таким образом регрессионные прямые позволяют определить функции, минимизирующие возникающие погрешности, а доверительные интервалы обеспечивают некоторые допустимые «разбросы»

значений.

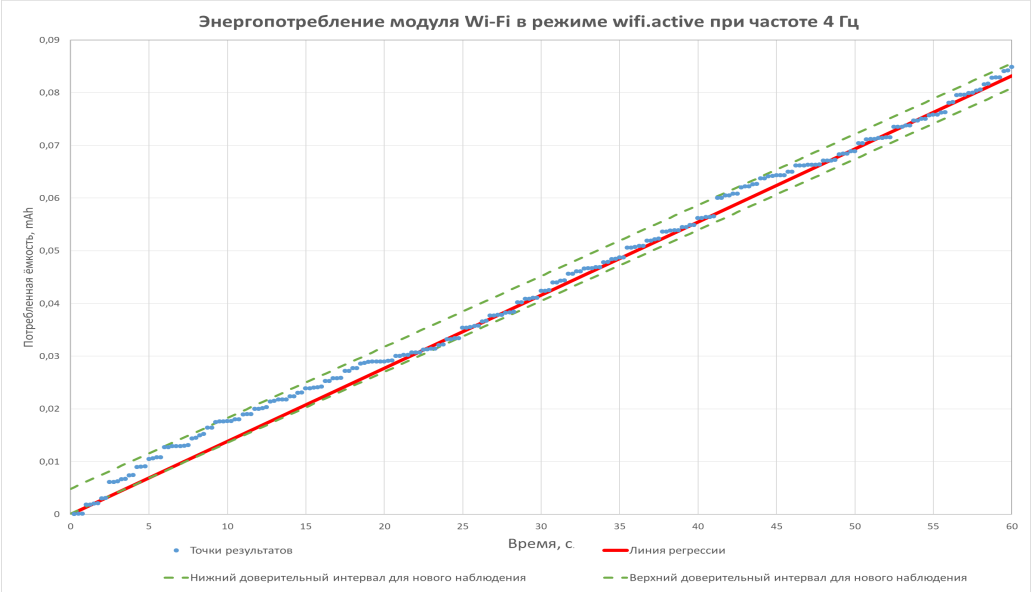

Рис. 5: Информация об энергопотреблении модуля Wi-Fi в режиме `wifi.active` при частоте 4 Гц

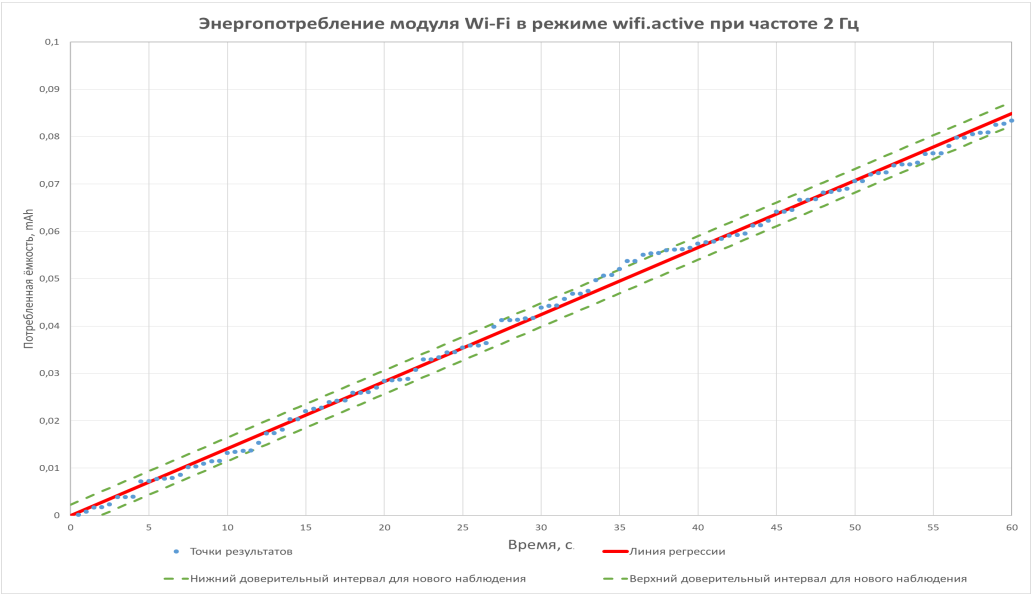

Рис. 6: Информация об энергопотреблении модуля Wi-Fi в режиме `wifi.active` при частоте 2 Гц

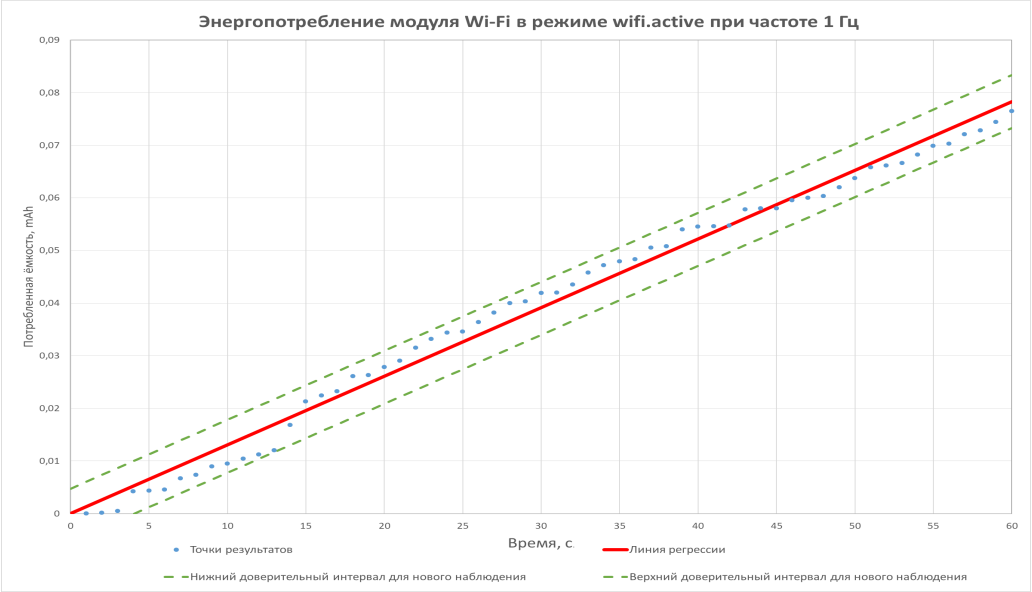

Рис. 7: Информация об энергопотреблении модуля Wi-Fi в режиме `wifi.active` при частоте 1 Гц

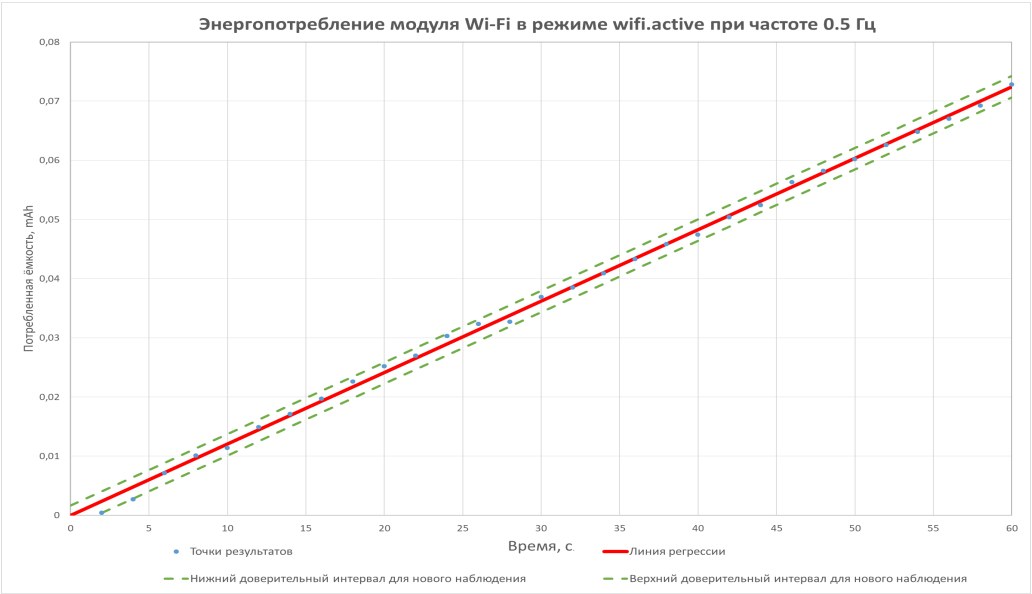

Рис. 8: Информация об энергопотреблении модуля Wi-Fi в режиме `wifi.active` при частоте 0.5 Гц

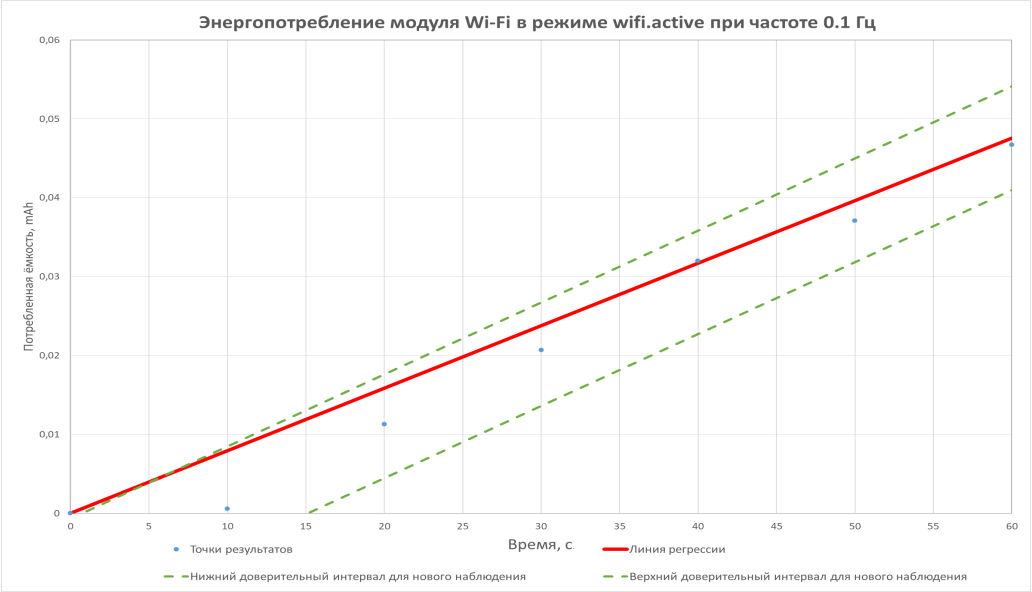

Рис. 9: Информация об энергопотреблении модуля Wi-Fi в режиме `wifi.active` при частоте 0.1 Гц

По построенным графикам и выходным данным эксперимента было обнаружено, что с уменьшением частоты дискретизации, то есть при уменьшении числа совершённых замеров за 60 секунд, энергопотребление в каждой полученной паре значений снижается. Например, на графике с частотой дискретизации 4 Гц можно увидеть, что энергопотребление выше, нежели с частотой дискретизации 0.1 Гц.

Важным замечанием является, что для каждой конкретной частоты дискретизации энергопотребление растёт достаточно линейно, именно поэтому, если объединить графики в один и соединить все точки, полученные в ходе измерений с конкретной частотой, то он будет выглядеть как множество расходящихся прямых.

Вероятно, это связано с тем, что получение информации об энергопотреблении средствами **adb** также затрачивает некоторую энергию, ведь **adb** использует для работы сервер, управляющий агрегацией и обменом данными [7].

2. **Тестирование модуля Wi-Fi в режиме `wifi.on`**

   На устройстве был включён модуль Wi-Fi, оно не было подключено ни к одной сети, причём, в отличие от предыдущего эксперимента, был выдержан

временной интервал в 30 секунд для того, чтобы пропустить сканирование, начинающееся автоматически после включения данного компонента.

По результатам тестирования для каждой частоты дискретизации были построены графики с применением регрессионного анализа, а также доверительные интервалы к ним. Для краткости и из-за аналогичности пункту 1, приведены они для текущего состояния не будут.

По выходным данным эксперимента было обнаружено, что потребление модуля Wi-Fi в режиме `wifi.on` в несколько раз меньше потребления в режиме `wifi.active`, что соответствует действительности, и всё также наблюдается линейный рост полученных данных энергопотребления.

Стоит учесть, что, по аналогии с предыдущим экспериментом, здесь наблюдается схожая тенденция: с уменьшением частоты дискретизации энергопотребление в каждой полученной паре значений снижается.

3. **Тестирование модуля Wi-Fi в режиме** `wifi.scan`

   Для данного тестового сценария была найдена команда, запускающая сканирование сетей Wi-Fi, `adb shell su 0 service call wifi 11`, однако для запуска она требует прав суперпользователя, которых нет на тестируемом устройстве. Поэтому тестирование данного режима не проводилось.

4. **Тестирование модуля Bluetooth в режиме** `bluetooth.on`

   На устройстве был включён модуль Bluetooth, оно не было подключено ни к одной точке, причём был выдержан временной интервал в 30 секунд для того, чтобы пропустить сканирование, начинающееся автоматически после включения данного компонента.

   По результатам тестирования для каждой частоты дискретизации были построены графики с применением регрессионного анализа, а также доверительные интервалы к ним. Для краткости и из-за аналогичности пункту 1, приведены они для текущего состояния не будут.

   По построенным графикам тоже было замечено, что энергопотребление с уменьшением частоты дискретизации также снижается. Стоит отметить, что для каждой конкретной частоты дискретизации энергопотребление тоже растёт линейно.

5. **Тестирование модуля Bluetooth в режиме** `bluetooth.active`

На устройстве был включён модуль Bluetooth, оно было подключено к беспроводной портативной аудиосистеме в режиме воспроизведения аудиофайлов, загруженных предварительно. Временной промежуток в 30 секунд, по аналогии с предыдущим экспериментом, выдерживать нет необходимости, так как сканирование доступных точек Bluetooth происходит до подключения к ним, и поэтому не могло отразиться в данном эксперименте.

По результатам тестирования для каждой частоты дискретизации были построены графики с применением регрессионного анализа, а также доверительные интервалы к ним. Для краткости и из-за аналогичности пункту 1, приведены они для текущего состояния не будут.

По построенным графикам также обнаружено, что значения энергопотребления при больших частотах дискретизации находятся выше аналогичных при более низкой частоте по тем же причинам. Все прямые, как и прежде, растут линейно и по значениям превосходят аналогичные из `bluetooth.on`, что соответствует действительности.

В связи с обнаруженной связью между уменьшением частоты дискретизации и снижением энергопотребления в каждой полученной паре для всех протестированных состояний, была поставлена задача провести исследование с целью поиска зависимости накладных расходов **adb** от частоты дискретизации. К примеру, показательной будет зависимость коэффициента наклона прямых соответствующих линий регрессий от частоты. Таким образом, будет предпринята попытка получить чистое значение потребления модулей Wi-Fi и Bluetooth. Данная задача рассматривается в работе Кузнецова И.А. «Исследование констант энергопотребления модулей Wi-Fi и Bluetooth и реализация модуля их определения в Navitas Framework».

5. Интеграция модулей Wi-Fi и Bluetooth в Navitas Framework

В рамках задачи интеграции модулей Wi-Fi и Bluetooth с учетом архитектуры Navitas Framework было выделено три основные подзадачи.

1. Добавить функциональность связанную с получением от **adb** информации об энергопотреблении в профилировщик NaviProf.

2. Добавить функциональность в плагин к Android Studio — Navitas Plugin для анализа и отображения полученных данных.

3. Реализовать экспериментальные классы в тестовом приложении NaviTest для определения энергопотребления соответствующих модулей на произвольном устройстве.

5.1. NaviProf

NaviProf — это gradle-плагин [22], отвечающий за профилирование, то есть получение информации об энергопотреблении устройства, а также инструментовку тестов и преобразование полученных данных в JSON файл.

В рамках модуля NaviProf была реализована описанная ниже логика.

1. Добавлено конфигурирование и подготовка тестируемого устройства перед запуском тестов, а именно: отключение питания батареи и сброс её статистики.

2. С помощью инструмента **adb** и команд, описанных в разделе 4.1., реализовано получение информации об энергопотреблении модулей Wi-Fi и Bluetooth, а также запись полученных данных в промежуточные текстовые файлы.

3. С учетом новой информации об энергопотреблении был переработан парсер текстовых файлов с возможностью последующего расширения другими компонентами.

4. При помощи изменения структуры класса `JSONGenerator` в него были добавлены модули Wi-Fi и Bluetooth, увеличена его масштабируемость. Также был расширен спектр поддерживаемых устройств: теперь имеется возможность профилировать устройства не только под управлением Android 7.0 — 11, но и некоторые с более ранней версией.

5. С учетом новой информации об энергопотреблении была изменена структура JSON файла.

5.2. Navitas Plugin

Navitas Plugin — плагин к Android Studio [21], отвечающий за конфигурирование тестов, запускающихся на устройстве, профилирование устройства с помощью NaviProf, парсинг полученных данных, представленных в формате JSON, анализ собранных данных и визуализацию результатов профилирования.

В рамках работы над Navitas Plugin была реализована приведённая ниже функциональность.

1. В соответствии с изменениями структуры JSON файла был изменен и его парсинг.

2. Была доработана возможность загружать Power Profile конкретного устройства, а не использовать только лишь Power Profile, предоставляемый в Navitas Framework по умолчанию.

3. Также был изменен Power Profile по умолчанию, добавлены начальные значения для режимов wifi.on, wifi.scan, wifi.active, bluetooth.on, bluetooth.active.

4. Была принята новая единица измерения энергопотребления. Так, миллиджоули (**mJ**) были заменены на миллиампер-часы (**mAh**), чтобы соответствовать привычным единицам измерения энергопотребления устройств. К тому же, **mAh** легко переводить в **mA**, требуемые для Power Profile.

5. Был изменен класс `ProfilingResultAnalyzer`, отвечающий за анализ собранных данных. Таким образом, теперь информация об энергопотреблении инкапсулируется в общий класс, состоящий из следующих компонентов: CPU, Wi-Fi и Bluetooth. Данное решение позволит масштабировать Navitas Framework и на другие компоненты, которым только предстоит обрести поддержку в плагине.

6. Расширенная новыми данными о Wi-Fi и Bluetooth итоговая информация об энергопотреблении была визуализирована. Для этого была изменена соответствующая front-end логика в Navitas Framework, написанная с использованием библиотеки Swing.

5.3. NaviTests

Корректность работы плагина проверяется в специально созданном Android-приложении [23], в рамках которого для тестирования модулей Wi-Fi и Bluetooth были написаны соответствующие тесты.

1. Тест модуля Wi-Fi в режиме `wifi.active`
   В отдельном созданном окне с помощью API YouTube происходит проигрывание прямой трансляции в течение 30 секунд.

2. Тест модуля Bluetooth в режиме `bluetooth.scan`
   Происходит сканирование доступных устройств с заранее выставленными настройками сканирования. Данное действие производится в течение 30 секунд.

Пример результатов работы тестов:

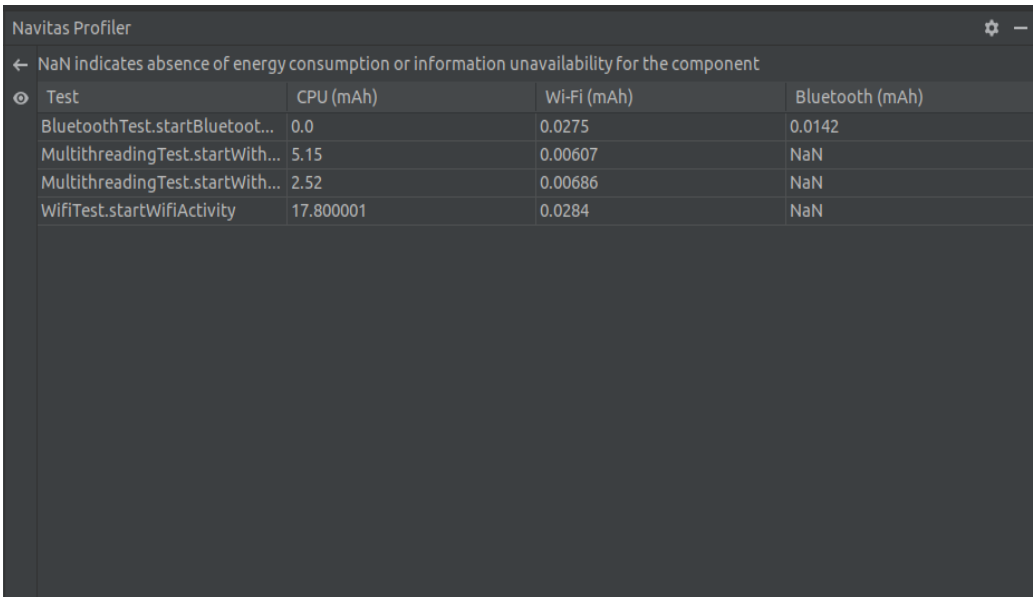

Рис. 20: Визуализация результатов тестов

В обобщенном режиме визуализации отображаются названия выполненных тестов, а также их энергопотребление по модулям устройства. Информация выдается в миллиампер-часах (**mAh**). В случае отсутствия потребления модулем или если информация о нём не поддерживается на тестируемом устройстве, то выдаётся **NaN**.

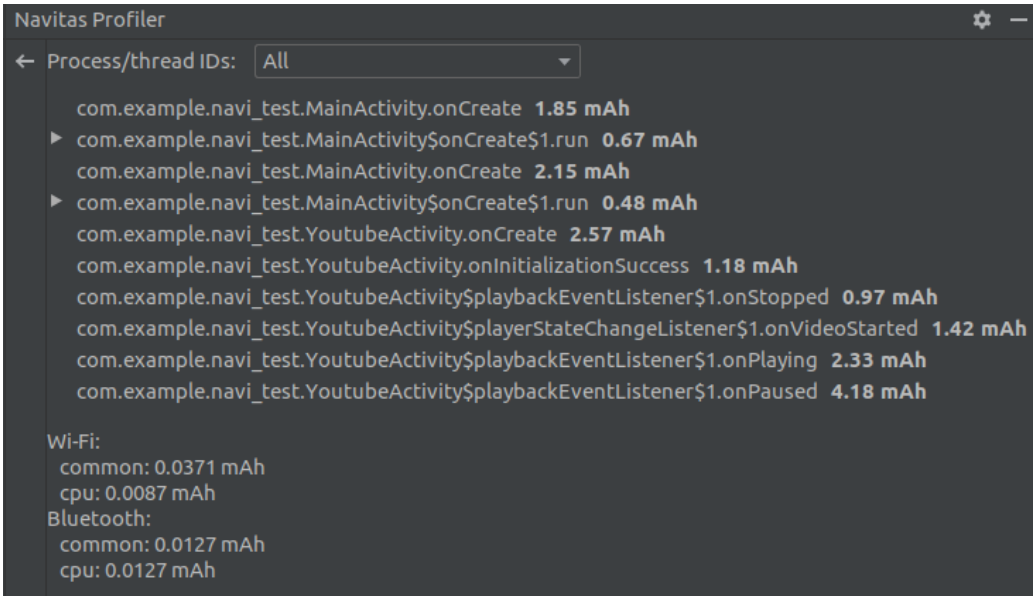

Рис. 21: Детальная визуализация результатов тестов

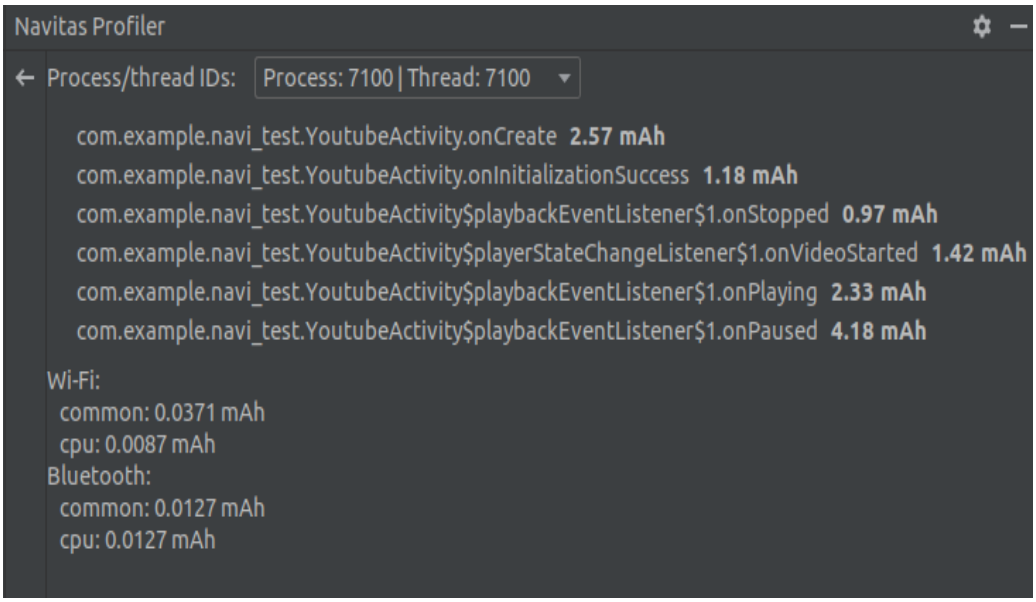

Рис. 22: Детальная визуализация результатов тестов на конкретном потоке

В детальной визуализации результатов отображено энергопотребление CPU по тестируемым методам с разделением на процессы и потоки, а также выводится детальная информация по энергопотреблению модулей Wi-Fi и Bluetooth о том, какое было общее потребление этого процесса на устройстве, сколько из

этого приходится на «чистое» потребление компонента, а сколько — на модули, так или иначе участвовавшие в его работе.

5.4. Другие улучшения и исправления

- В рамках работы над Navitas Framework и в качестве помощи с защитой статьи по данному плагину на конференции `SEIM-2021` был реализован набор тестовых сценариев.

    1. Написан тест для оценки вариантов кода, в котором на предмет энергопотребления сравниваются два типа: отдельно реализованный класс с созданием объекта этого класса и анонимный класс, повторяющий реализацию первого. Navitas Framework показал, что первый сценарий использует больше энергии, нежели второй. Это связано с тем, что при использовании анонимного класса происходят оптимизации `ahead-of-time` компилятора `ART`, которые положительно сказываются на энергопотреблении.
    2. Также был создан тест, показывающий, сколько энергии потребляет код, работающий в отдельном потоке и создающий массив, заполняя его случайными числами, и затем записывающий его в файл. Для сравнения существует и второй вариант данного эксперимента: после заполнения массив ещё сортируется. Navitas Framework показал, что вариант с сортировкой использует больше энергии, что соответствует действительности: при сортировке большого массива чисел растет процессорное время, число инструкций и, как следствие, общее энергопотребление.

- Увеличино количество поддерживаемых устройств за счёт рассмотрения большего числа вариаций вывода информации об энергопотребления средствами **adb**, так как основные ошибки возникали именно из-за их несоответсвия требуемой в Navitas Framework структуре. Таким образом, минимальная поддерживаемая версия Android в Navitas Framework расширилась до 5.1 (Android API Level 22). Однако рекомендуемая версия осталась без изменений — 7.0 и выше (Android API Level 24+).

- Было произведено улучшение распределения процессоров по кластерам, вызванное тем, что на некоторых Power Profile устройств предоставляется некорректная информация о нём. Такое несоответствие ранее приводило к аварийному завершению программы. Теперь присутствует возможность

использовать больше процессоров, чем обозначено в кластерах, тогда как обратная функциональность присутствовала и раньше. Также это открывает возможности для профилирования устройств с использованием заведомо не принадлежащих им Power Profile. Таким образом, выбор верного Power Profile для тестирования и достижения наилучшей точности — ответственность пользователя, однако планируется поиск и реализация более продвинутого решения.

- Текстовые файлы и JSON, содержащие логи тестирования устройства, сделаны доступными пользователю и не удаляются после окончания экспериментов.

6. Результаты

Благодаря работе над учебной практикой были достигнуты следующие результаты.

1. Проведен обзор энергопрофилировщиков.
2. Проведен обзор методов получения информации об энергопотреблении, а также выбран метод для исследования и последующего внедрения в Navitas Framework.
3. Проведен анализ результатов, получаемых выбранным методом.
4. Внедрен метод получения информации об энергопотреблении средствами **adb** в Navitas Framework.
5. Проведена работа над улучшением существующей функциональности Navitas Framework, повышением стабильности и увеличением числа поддерживаемых устройств.

Благодаря полученным результатам, виден дальнейший вектор работы над проектом Navitas Framework со следующей поставленной задачей:

1. Добавить валидацию используемого для тестирования Power Profile по CPU.

Список литературы

[1] Number of smartphone users worldwide from 2016 to 2021:
https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Mobile Operating System Market Share Worldwide Oct 2019 - Oct 2020:
https://gs.statcounter.com/os-market-share/mobile/worldwide

[3] Estimating the number of Wi-Fi users via smartphones Jan 2019:
https://www.bluetooth.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf

[4] Total Annual Bluetooth Device Shipments in 2019:
https://www.bluetooth.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf

[5] Estimating the number of Wi-Fi users via smartphones Jan 2019:
https://www.bluetooth.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf

[6] Default Power Profile by Google:
https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/xml/power_profile.xml

[7] Android Debug Bridge Developers Documentation:
https://developer.android.com/studio/command-line/adb

[8] Application Sandbox in Android:
https://source.android.com/security/app-sandbox?hl=en

[9] PowerTutor - A Power Monitor for Android-Based Mobile Platforms:
http://ziyang.eecs.umich.edu/projects/powertutor/index.html

[10] Android Developers Documentation about Wi-Fi Manager:
https://developer.android.com/reference/android/net/wifi/WifiManager

[11] Wi-Fi module implementation in PowerTutor:
https://github.com/msg555/PowerTutor/blob/master/src/edu/umich/PowerTutor/components/Wifi.java

[12] Robert P. Dick, Lide Zhang, Birjodh Tiwana and Zhiyun Qian. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105-114, october 2010.
https://dl.acm.org/doi/abs/10.1145/1878961.1878982

[13] Android Developers Documentation about Energy Profiler:
https://developer.android.com/studio/profile/energy-profiler

[14] Profile battery usage with Batterystats and Battery Historian:
https://developer.android.com/topic/performance/power/setup-battery-historian

[15] Analyze power use with Battery Historian:
https://developer.android.com/topic/performance/power/battery-historian

[16] Power Profiles for Android:
https://source.android.com/devices/tech/power

[17] Power value settings:
https://source.android.com/devices/tech/power/values#values

[18] Useful links for System Literature Review:
https://docs.google.com/spreadsheets/d/17SRvTyuGqcMrsQe856b6pDOe-9Ynszg8KhhAeGrcvvM/edit#gid=0

[19] Andrey Saksonov. Method to Derive Energy Profiles for Android Platform, Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany, 2014.
http://www.se.uni-oldenburg.de/documents/saksonov-MA2014.pdf

[20] Yocto-Amp ammeter:
http://www.yoctopuce.com/EN/products/usb-electrical-sensors/yocto-amp

[21] Navitas Plugin implementation:
https://github.com/Stanislav-Sartasov/Navitas-Framework/tree/master/Navitas-Plugin

[22] NaviProf plugin implementation:
https://github.com/Stanislav-Sartasov/Navitas-Framework/tree/master/NaviProf

[23] NaviTests implementation:
https://github.com/Stanislav-Sartasov/Navitas-Framework/tree/master/NaviTests

[24] Bash-scripts used for Wi-Fi and Bluetooth power consumption research:
https://github.com/Stanislav-Sartasov/Navitas-Framework/tree/wifi_bluetooth_research