Санкт-Петербургский государственный университет

Математико-Механический факультет
Кафедра Системного Программирования

Группа 20Б.11-мм

Тарбеев Андрей Владимирович

Исследование энергопотребления модуля GPU и создание его модели энергопотребления в Navitas Framework

Отчёт по учебной практике
в форме «Решение»

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

Санкт-Петербург
2022

Оглавление

Введение

Мобильные устройства настолько сильно вошли в нашу жизнь, что сегодняшний день сложно вообразить без них. С каждым днём количество пользователей этих устройств растёт. В 2021 г. насчитывается около 6.3 млрд. владельцев смартфонов [23]. Дополнительно можно заметить, что лидирующей операционной системой для мобильных устройств является Android [22].

С ростом числа разрабатываемых мобильных приложений, использующих работу графического процессора, растёт и спрос на средства, позволяющие оценивать их влияние на энергопотребление. Поэтому вопрос об оценке энергопотребления данного модуля является актуальным, потому что от этого зависит то, насколько будут оптимизированы приложения, что в свою очередь влияет на продолжительность работы устройств.

Проект Navitas Framework [12], разработанный кафедрой Системного Программирования СПбГУ, является одним из существующих инструментов для оценки энергопотребления различных модулей устройства. В настоящий момент имеется поддержка оценки энергопотребления следующих модулей: CPU, дисплея, Wi-Fi и Bluetooth.

В данной работе предлагается рассмотреть подходы к извлечению информации о работе графического процессора, далее создать модель энергопотребления, а также предлагается дальнейшая реализация этой модели в Navitas Framework.

1. Постановка задачи

Целью работы является внедрение поддержки оценки энергопотребления графического процессора в Navitas Framework. Для её выполнения были поставлены следующие задачи.

1. Провести обзор существующих энергопрофилировщиков;
2. Провести обзор моделей энергопотребления графического процессора;
3. Рассмотреть подходы к извлечению информации о работе графического процессора;
4. Добавить поддержку модуля графического процессора в Navitas Framework.

2. Обзор предметной области

2.1. Обзор существующих энергопрофилировщиков

Обзор существующих инструментов для оценки энергопотребления был достаточно полно выполнен в работах прошлого года [31, 30]. В них были рассмотрены такие инструменты, как PowerTutor [26], Energy Profiler [2] и Battery Historian [4], а также выделены их недостатки.

После их анализа выяснилось, что ни один из этих инструментов не обладает возможностью измерять энергопотребление графического процессора.

Помимо указанных выше профилировщиков, существует инструмент Navitas Framework. Стоит заметить, что поддержка измерения энергопотребления графического процессора в нём также отсутствует, что и предлагается исправить в этой работе.

2.2. Способы получения данных об энергопотреблении

2.2.1. Внешнее измерительное устройство

Использование внешнего измерительного устройства позволяет напрямую получать данные об энергопотреблении, в этом случае наличие этого аппарата является обязательным. В этом и заключается недостаток этого способа, потому что нельзя гарантировать, что данное внешнее устройство будет в наличии у потенциального пользователя. Проект Navitas Framework предполагает использование данных, которые предоставляет само устройство без необходимости измерять энергопотребление с помощью внешних измерительных устройств.

2.2.2. Android Debug Bridge

Использование же adb [1] вполне уместно в силу наличия уже реализованных подходов в Navitas Framework, а также независимости от наличия того или иного реального дополнительного аппарата. К тому

же adb обладает набором уже готовых инструментов, с помощью которых можно извлечь ту или иную информацию об устройстве, а также при наличии соответствующих прав позволяет изучать директории с целью поиска и дальнейшего извлечения данных, которые могут быть полезными для расчёта энергопотребления.

2.3. Модели энергопотребления графического процессора

Поиск статей на тему создания моделей энергопотребления GPU осуществлялся с помощью сервиса Google Scholar [16]. Запросы для поиска были следующие: *mobile gpu power model*, *gpu power modeling for smartphones*. После отбора по релевантности детально были изучены девять статей. В основном предлагались подходы в виде реализации utilization-based модели, иногда с некоторыми доработками в виде добавления в модель дополнительных показателей (счётчиков контроля производительности).

2.3.1. Utilization-based модель

В работе C. Yoon, G. Ryu и H. Cha [5] была предложена модель энергопотребления GPU на основе данных о его использовании (utilization), а также проведено сравнение результатов созданной модели с результатами замеров внешнего измерительного устройства.

В качестве аппарата для экспериментов использовался Samsung Galaxy S3 GT-I9300 с Mali-400.

Индекс частоты	Частота (MHz)	Напряжение (V)	Пороги использования GPU	
			Верхний	Нижний
0	160	0.875	70	0
1	266	0.900	90	62
2	350	0.950	90	85
3	440	1.025	100	90

Рис. 1: Таблица соотношения частоты и напряжения Mali-400 [5]

Для анализа энергопотребления GPU было реализовано приложение, которое контролировало процент использования GPU с помощью изменения числа 3D кубов и скорости их вращения. Также были приняты меры по уменьшению влияния работы экрана на энергопотребление: на время эксперимента он выключался. Включался режим полёта. Замеры CPU и GPU делались раз в секунду. Для CPU использовалась модель, предложенная автором этой же статьи. В конце вычиталось энергопотребление CPU из общего энергопотребления. В итоге получался график зависимости utilization от энергопотребления.

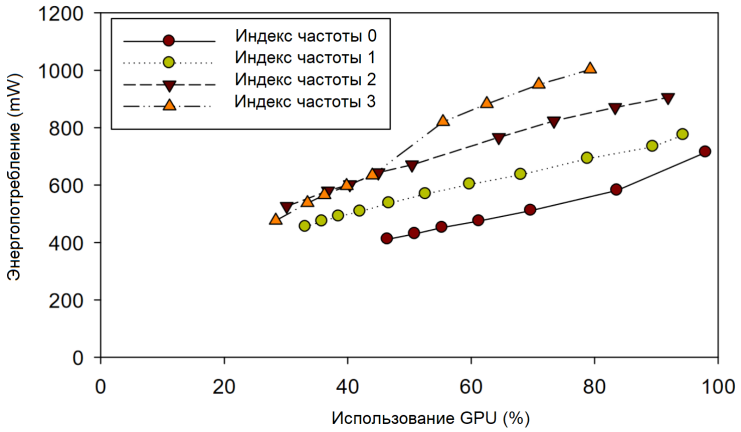

Рис. 2: Результат измерения энергопотребления GPU в зависимости от utilization [5]

Было установлено, что энергопотребление увеличивается пропорционально utilization на определенной частоте, поэтому была использована линейная регрессия для каждого из результатов частот из рис. 2. Опираясь на предыдущие замеры, была предложена следующая модель.

$$P_{GPU} = \beta^{GPU}_{freq_i} \times u_{GPU} \ + \beta^{GPU}_{base_i}, \quad i = 0, 1, 2, 3,$$

где i — это индекс рабочей частоты, $\beta^{GPU}_{freq_i}$ — градиент результата линейной регрессии для частоты i, u_{GPU} — использование GPU для i, $\beta^{GPU}_{base_i}$ — пересечение оси Oy линейной регрессии для частоты i.

Затем было произведено сравнение результатов с результатами внешнего измерителя (Monsoon Power Monitor [19]). Замеры делались один

раз в секунду. После применения модели энергопотребления GPU, константная ошибка оценки составила 45 mW в секунду в среднем.

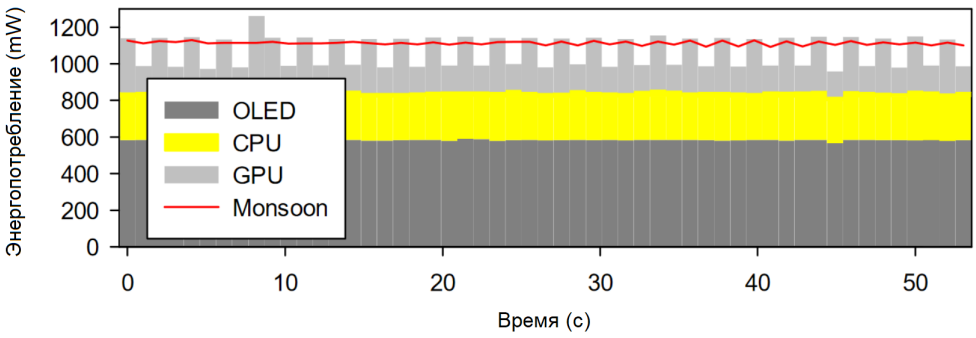

Рис. 3: Результат сравнения между оценкой с помощью модели и измерениями внешнего измерительного устройства [5]

2.3.2. Utilization-based модель с PMC

В работе T. Jin, S. He и Y. Liu [24] было предложено улучшение utilization-based модели с помощью добавления в модель счётчиков контроля производительности (performance monitoring counters).

На рис. 4 показано сравнение результатов utilization-based модели с результатами внешнего измерительного устройства Monsoon Power Monitor. По словам автора, ошибка оценки utilization-based модели может достигать 25%.

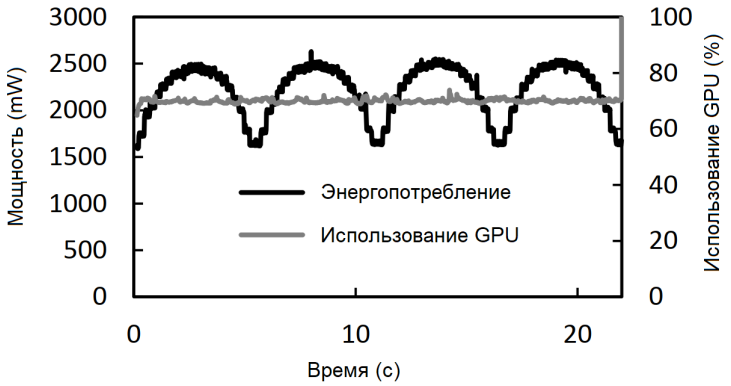

Рис. 4: Сравнение энергопотребления и использования GPU [24]

Предлагаемая модель выглядит следующим образом.

$$P = \alpha_1 \times U \ + \alpha_2 \times V \ + \alpha_3 \times T \ + b,$$

где U — это использование GPU, V — загрузка обработчика вершин, T — загрузка обработчика пикселей, α_1, α_2, α_3 — коэффициенты трех предикторов в линейной модели, b — базовое энергопотребление.

В результате сравнения с utilization-based моделью полученная модель имела среднюю ошибку 4,5% против 7,2%.

Однако наличие счётчиков контроля производительности может различаться от одной модели GPU к другой, что делает эту модель менее универсальной. Эту мысль подтверждает сам автор.

3. Извлечение информации о работе GPU

Все манипуляции осуществлялись на четырех аппаратах, которые были доступны в той или иной степени.

1. Samsung SM-J106F Galaxy J1 Mini Prime (Android 6.0.1, Spreadtrum SC9830, Mali-400 MP2, rooted) [9];
2. Fly IQ4514 Evo Tech 4 (Android 7.1.2, MediaTek MT6582M, Mali-400 MP2, rooted) [6];
3. Honor COL-L29 10 (Android 10, HiSilicon Kirin 970, Mali-G72 MP12, unrooted) [7];
4. Samsung SM-A305 Galaxy A30 (Android 11, Exynos 7904, Mali-G71 MP2, unrooted) [8].

Далее для поддержания структуры хода изложения исследования информация о проделанных действиях и полученных результатах будет представляться как нумерованный список, где каждому номеру соответствует модель аппарата из списка выше.

3.1. Использование dumpsys

Начать было решено с исследования лог-информации, которую предоставляет команда *dumpsys*. На всех устройствах были рассмотрены логи следующих опций на предмет наличия хотя бы какой-либо информации, связанной с работой графического процессора.

- *battery*;
- *batteryproperties*;
- *batterystats*;
- *cpuinfo*;
- *gpu*;

- *graphicsstats*;
- *hardware_properties*;
- *power*.

Вывод логов был преобразован в текстовый документ (например, на операционной системе Windows это было выполнено следующим образом: *adb shell dumpsys battery > battery_logs.txt*), который после и был предметом изучения.

После рассмотрения каждого из логов каждого из устройств, было установлено, что данных о работе GPU эти логи не содержат.

3.2. Использование bugreport

Следующим шагом было решено подтвердить тот факт, что отладочная информация не содержит сведений о работе GPU. Для каждого из устройств была выполнена следующая команда: *adb bugreport*. Результатом выполнения команды является ZIP-файл, содержащий в себе логи *dumpsys*, *dumpstate*, *logcat*.

Анализ этих файлов не привел к ожидаемым результатам: данных о работе графического процессора найдено не было.

3.3. Поиск непосредственно на устройстве

3.3.1. Использование power_profile.xml

power_profile.xml [3] представляет из себя файл с заданными значениями энергопотребления в миллиамперах (mA) различных модулей устройства. Эти значения устанавливаются производителем, однако это не является необходимым, и есть вероятность столкнуться с тем, что этот файл будет заполнен нулями, либо стандартными значениями [15].

Для начала стояла задача найти файл *power_profile.xml*. Он был найден в *framework-res.apk*, который на всех устройствах находился в */system/framework*. Для извлечения требуемого файла понадобилась

улитила Apktool [17], одной из возможностей которой является разборка [28] APK-файлов [27], которая в свою очередь осуществляется следующей командой: *apktool decode framework-res.apk*.

Результаты анализа следующие.

1. Файл оказался заполненным значениями, которые отличались от стандартных;
2. Файл оказался заполненным стандартными значениями;
3. Файл оказался заполненным стандартными значениями;
4. Файл оказался заполненным стандартными значениями.

Только один аппарат содержал в себе значения, установленные производителем. При дальнейшем изучении выяснилось, что данных об энергопотреблении графического процессора он не содержит. В подтверждение был найден источник [10], в котором также было установлено, что не все модели содержат информацию о данном модуле.

3.3.2. На устройствах с root

Первоначально было решено проанализировать все директории устройства с правами суперпользователя [29] на предмет содержания в них таких слов, как *gpu*, *mali* (в силу того, что аппараты с правами суперпользователя имели графический процессор Mali). Это было выполнено с помощью команды *adb shell -type f*.

После анализа путей и файлов, которые предположительно могли содержать требуемую информацию, внимание привлекли следующие файлы: *utilization_pp*, *utilization_gp*, *utilization_gp_pp*.

1. Для первого устройства они находились в *sys/kernel/debug/mali0*;
2. Для второго они находились в *sys/kernel/debug/mali*.

При первичном их осмотре они содержали в себе нули, поэтому не сразу стало ясно, что это именно те файлы, которые нужны. После

этого на аппарате было запущено приложение, использующее графический процессор. Вновь были проверены все файлы, и именно эти три файла теперь содержали числа, отличные от нуля, причём с каждой новой проверкой их содержимого значения менялись.

Следующим шагом было выяснить, какой смысл несут эти файлы. Из спецификации Mali [21] выяснилось, что *gp* и *pp* — это аббревиатуры, которые расшифровываются как *geometry processor* и *pixel processor* соответственно. Далее был выяснен смысл значений, представленных в этих файлах: пояснения были найдены в репозитории [11], то есть значение *0* соответствует отсутствию использования (utilization) графического процессора, а значение *256* — полному использованию.

Дальше были обнаружены следующие дополнительные сведения.

1. У первого устройства были обнаружены данные, которые находятся в *sys/devices/60000000.gpu/power*, содержащие информацию о статусе работы GPU (*active* или *suspended*) и времени работы (*runtime_ active_ time*) и времени простоя (*runtime_ suspended_ time*);
2. У второго устройства этих данных обнаружено не было.

Было установлено, что набор данных и их расположение на устройстве отличаются на каждом из устройств даже при условии наличия одинакового графического процессора.

3.3.3. На устройствах без root

После череды экспериментов было установлено, что данные, которые содержат информацию о работе графического процессора, недоступны, если на устройстве отсутствуют права суперпользователя.

3.4. Вывод

В результате проделанной работы можно подвести следующие итоги.

- Debug-информация не содержит никаких данных о рабочей частоте и энергопотреблении GPU;

- Файл *power_profile.xml* может содержать данные об энергопотреблении GPU, однако количество устройств, в которых производитель заменил стандартные значения на настоящие, небольшое;
- Для доступа к данным работы GPU нужны права суперпользователя;
- Данные, предоставляемые устройством, могут существовать на одном устройстве, но отсутствовать на другом.

4. Работа с Navitas Framework

4.1. Воспроизведение модели энергопотребления GPU

В качестве используемой модели энергопотребления графического процессора была выбрана utilization-based модель ввиду отсутствия на двух устройствах данных о счётчиках контроля производительности (performance monitoring counters).

4.1.1. Описание хода эксперимента

Для проведения замеров использовался мультиметр UNI-T UT803 [25]. В качестве генератора выступал QJE QJ3003H [20]. Устройство, с которым проводился эксперимент, было Fly [6]. Дополнительно был реализован скрипт, который с помощью adb обращался к устройству с целью извлечения информации о работе нескольких компонентов устройства [13]. Во время замеров на устройстве было запущено приложение Smash Hit [18], которое задействовало работу графического процессора, активирован режим полёта, установлена минимальная яркость экрана, выключен Wi-Fi и Bluetooth. Длительность замеров составляла около семи минут.

4.1.2. Сравнение значений энергопотребления внешнего измерителя и модели

По окончании эксперимента было посчитано общее энергопотребление устройства за время замеров. Для GPU использовалась модель, рассмотренная ранее [5]. Для CPU была использована следующая модель.

$$P_{CPU} = CPU_{utilization\,percent} \times CPU_{consumption\,on\,current\,frequency}$$

Для определения энергопотребления экрана было решено действовать следующим образом: произвести замер, используя приложение [14], которое в течение десяти секунд отображает на экране один цвет, далее

вычесть из полученного энергопотребления энергопотребление CPU, рассчитать среднее энергопотребление экрана в секунду.

Сложив энергопотребление GPU, CPU и дисплея, выяснилось, что разница с внешним измерителем составляет около десяти процентов, что будем считать приемлемым в рамках одного графического процессора, который совпадает с тем, что использовали авторы статьи [5], но разных устройств.

4.2. Интеграция в Navitas Framework

4.2.1. NaviProf

В рамках gradle-плагина NaviProf, который отвечает за получение информации об энергопотреблении от устройства и преобразование полученных данных в JSON файл, были добавлены команды для adb для извлечения информации об использовании графического процессора, а также запись этой информации в JSON файл вместе с уже поддерживаемыми модулями устройства.

4.2.2. NaviPlugin

В результате работы над NaviPlugin, который парсит JSON файл, полученный по окончании работы NaviProf, анализирует его содержимое и выдает результаты профилирования, был дописан парсер для работы с информацией от GPU модуля. Кроме этого была добавлена поддержка анализа результатов сбора информации о работе графического процессора с дальнейшим применением модели для вычисления приблизительного энергопотребления этого модуля.

4.2.3. Результат

Теперь в таблице с результатами профилирования присутствует столбец для GPU, который содержит результаты измерения его энергопотребления.

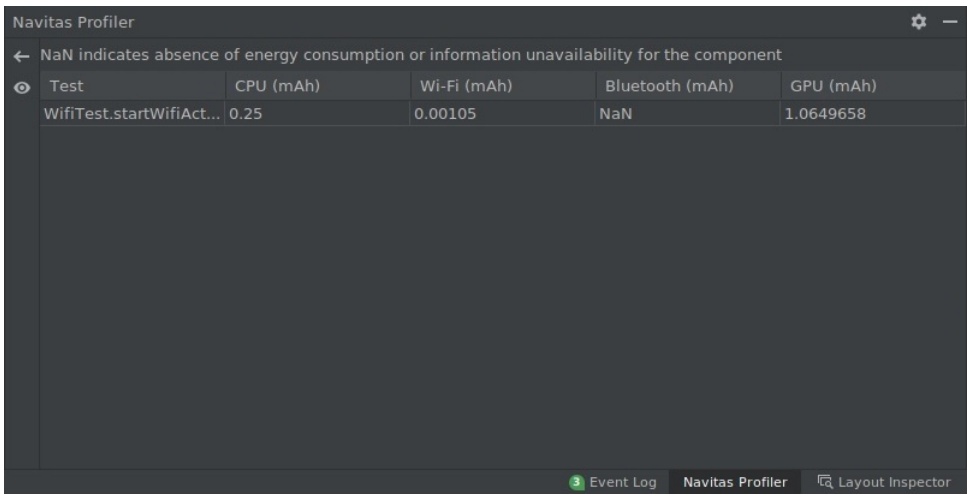

Рис. 5: Результат энергопрофилирования тестового приложения

4.3. Другие улучшения

В процессе интеграции модели оценки энергопотребления GPU в Navitas Framework было выяснено, что от устройства требовались логи работы всех ранее внедренных модулей, даже если эти логи отсутствовали, что приводило к ошибкам, которые препятствовали завершению профилирования. Устройство Fly [6] не отдавало логи работы Bluetooth, что и приводило к ранее описанной ситуации. Поэтому было решено сделать возможным продолжение профилирования, несмотря на отсутствие логов какого-либо из компонентов устройства.

Заключение

В результате работы над учебной практикой в течение осеннего и весеннего семестров были выполнены следующие задачи.

- Выполнен обзор существующих энергопрофилировщиков;
- Выполнен обзор моделей энергопотребления графического процессора;
- Рассмотрены подходы к извлечению информации о работе графического процессора;
- Добавлена поддержка модуля графического процессора в Navitas Framework;
- Была повышена стабильность работы плагина.

Список литературы

[1] Android. Android Debug Bridge (adb) // https://developer.android.com/studio/command-line/adb.

[2] Android. Inspect energy use with Energy Profiler // https://developer.android.com/studio/profile/energy-profiler.

[3] Android. Measuring Power Values // https://source.android.com/devices/tech/power/values.

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

[5] C. Yoon G. Ryu H. Cha. Utilization-based Power Modeling of Modern Mobile Application Processor. — 2013. — URL: https://mobed.yonsei.ac.kr/mobed_pages/pdf/mobed-tr-2013-01.pdf.

[6] Devicespecifications. Fly Evo Tech 4 - Технические характеристики // https://www.devicespecifications.com/ru/model/eddd306b.

[7] Devicespecifications. Huawei Honor 10 - Технические характеристики // https://www.devicespecifications.com/ru/model/2cbb49df.

[8] Devicespecifications. Samsung Galaxy A30 - Технические характеристики // https://www.devicespecifications.com/ru/model/cbf74f5d.

[9] Devicespecifications. Samsung Galaxy J1 mini Prime - Технические характеристики // https://www.devicespecifications.com/ru/model/f89c3f9d.

[10] Forum XDA. Other Power_profile.xml in comparison // https://forum.xda-developers.com/t/

haldis-benchmark-thread-for-governors-overclocking-testing-the-l
3376159/page-4#post-69099666.

[11] GitHub. Mali. mali_utgard.h // https://github.com/libcg/mali/blob/master/include/linux/mali/mali_utgard.h.

[12] GitHub. Navitas Framework // https://github.com/Stanislav-Sartasov/Navitas-Framework.

[13] Github. ADB Script for Research Project Spring 2022 // https://github.com/SofuriHafe/fly-iq4514-adb-info-grubber.

[14] Github. Energy-consumption-screen-test-app // https://github.com/artmotika/energy-consumption-screen-test-app.

[15] Google Git. Default power_profile.xml // https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/xml/power_profile.xml.

[16] Google Scholar // https://scholar.google.com/.

[17] Ibotpeaches. A tool for reverse engineering Android apk files // https://ibotpeaches.github.io/Apktool/.

[18] Mediocre. Smash Hit // http://www.smashhitgame.com/.

[19] Monsoon Solutions Inc. Power Monitor // https://web.archive.org/web/20150526072037/https://www.msoon.com/LabEquipment/PowerMonitor/.

[20] QJE QJ3005H 30V 5A REGULATED LINEAR BENCH POWER SUPPLY // https://www.triotest.com.au/electronic-and-rf/power-sources/qje-qj3005h-30v-5a-regulated-linear-bench-power-supply.

[21] SUNXI. Mali // https://linux-sunxi.org/Mali.

[22] Statcounter. Mobile Operating System Market Share Worldwide // https://gs.statcounter.com/os-market-share/mobile/worldwide.

[23] Statista. Number Of Smartphone Users From 2016 To 2021 // https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.

[24] T. Jin S. He Y. Liu. Towards Accurate GPU Power Modeling for Smartphones. — 2015. — URL: https://dl.acm.org/doi/abs/10.1145/2751496.2751502.

[25] UNI-T UT803 Multimeter // https://www.uni-trend.com/meters/.

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

[27] Wikipedia. Android application package // https://en.wikipedia.org/wiki/Android_application_package.

[28] Wikipedia. Reverse engineering // https://en.wikipedia.org/wiki/Reverse_engineering.

[29] Wikipedia. Superuser // https://en.wikipedia.org/wiki/Superuser.

[30] Кузнецов Илья. Исследование констант энергопотребления модулей Wi-Fi и Bluetooth и реализация модуля их определения в Navitas Framework. — 2021. — P. 6–9. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/spring-2021/uchebnye-praktiki/pi-2/Kuznetsov_Ilya-report.pdf.

[31] Мирошников Владислав. Исследование энергопотребления модулей Wi-Fi и Bluetooth и их интеграция в Navitas Framework. — 2021. — P. 6–9. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/spring-2021/uchebnye-praktiki/pi-2/Miroshnikov-report.pdf.