Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Петрозаводский государственный университет»
Институт математики и информационных технологий
кафедра информатики и информационного обеспечения

(подпись соискателя)

Савинов Олег Олегович

Магистерская дисертация

Персонализированный анализ энергопотребления мобильных устройств

Направление 01.04.02 — Прикладная математика и информатика

Научный руководитель:
к.т.н., доцент О. Ю. Богоявленская

_____

(подпись руководителя)

Петрозаводск — 2020

Оглавление

Введение

За последние несколько лет произошел прорыв на рынке мобильных технологий. В настоящее время тысячи новых приложений ежедневно публикуются в интернет магазинах, увеличивая спрос на новые и более мощные устройства. При этом изменилось само использование и назначение устройств, теперь мы ставим иные, более сложные задачи для которых требуется больше энергии. Обеспечение эффективности энергопотребления мобильных вычислительных систем, особенно при беспроводной передаче данных, участвующих в мобильных приложениях - является одной из важных задач. ~~Выполнение~~ современного мобильного приложения требует намного больше вычислительных и сетевых ресурсов и, следовательно, ~~потребляется гораздо больше энергии~~. Однако технологии ~~развития~~ батареи развиваются не так стремительно, как мобильные компьютерные технологии и не в состоянии удовлетворить растущий спрос на энергию. Это привело к значительному снижению времени работы аккумулятора, из-за чего возрастает актуальность проблемы эффективного мониторинга, такого важного ресурса, как заряд батареи.
В связи с этим являются актуальными цели работы:

1. Разработка прототипа системы мониторинга энергопотребления устройств.

2. Разработка мультиплатформенного приложения, предсказывающего на основе индивидуальных данных каждого пользователя, время разряда батареи ноутбука/смартфона.

В рамках работы планируется решение следующих задач:

1. Развитие методов анализа и прогнозирования разрядки устройств.
2. Совершенствование вычислений на основе статистических методов времени разрядки батареи.
3. Разработка системы отслеживание отклонений от нормы (оценок индивидуального шаблона потребления) и информирование пользователя об отклонении от нормы
4. Разработка прототипа интерфейса, выводящего на основе расчетов алгоритма, сообщения в виде всплывающего уведомления, о вероятном времени разрядки батареи.
5. Разработка прототипа интерфейса, выводящего всплывающее сообщение, о произошедшем отклонении от привычного режима энергопотребления, на основе расчетов алгоритма.
6. Разработка прототипа интерфейса, строящего графическое представление прогноза разрядки батареи.
7. Разработка прототипа приложения под OS windows и OS Android.

Глава 1

Использованные методы и технологии

Для разработки прототипа приложения персонализированного анализа энергопотребления, ~~требовалось изучить нижеперечисленные материалы.~~

Статистический анализ экспериментальных данных требуется для того, чтобы анализировать считанные данные и строить по ним вероятностные модели, требуются некоторые знания в области теории вероятности (Глава 1, пункт 1.1).

Марковские цепи требуются для более углубленного проведения анализа эксперементальных данных, используется аппарат теории Марковских цепей (Глава 1, пункт 1.2). Они используются для построения прогноза разрядки батареи на основе матрицы переходов полученной в процессе считывания информации с помощью специальных библиотек Windows и построения вариационных рядов.

Бинарное возведение в степень нужно для поиска вектора переходных вероятностей необходимо автоматизировать возведение матрицы в степень. С этой целью используется метод бинарного возведения в степень (Глава 1, пункт 1.3).

Проверка на равномерность используется для построения

нескольких оптимальных шаблонов для одного пользователя, использовалась проверка распределения на равномерность. В случае если шаблон отклонялся от статистическго критерия, то создавался бы еще один оптимальный шаблон (или сверялся с уже созданными). Эта идея работала для равномерно распределенных последовательностей. Но проведя тесты на разных входных данных стало понятно, что при различном энергопотреблении в рамках одной сессии, алгоритм проверки дает неверные результаты (Глава 1, пункт 1.4).

Метод обнаружения нарушения процесса и выявление разладок требуется так как прототип приложения расчитан на персональные экспериментальные данные и строит шаблон энергопотребления, то требуется следить за отклонениями от шаблона, для этого используется обнаружение разладок (Глава 1, пункт 1.5).

Специальные библиотеки C++ нужны для того, чтобы считывать все необходимые данные для дальнейшего анализа, требовались специальные библиотеки C++ (Глава 1, пункт 1.6).

Классы QWidget и QDesktopWidget требуются, чтобы приложение выводило результаты собранных и обработанных данных и не загромаждало экран устройства пользователя, был разработан прототип интерфейса всплывающих уведомлений посредством среды разработки Qt, в которой так же как и в C++, требовались специальные библиотеки (Глава 1, пункт 1.7).
Классы QWidget и QDesktopWidget требуются, чтобы приложение выводило результаты собранных и обработанных данных и не загромаждало экран устройства пользователя, был разработан прототип интерфейса всплывающих уведомлений посредством среды разработки Qt, в которой так же как и в C++, требовались специальные библиотеки (Глава 1, пункт 1.7).

Специальные библиотеки и классы Android Studio требу-

ются, чтобы можно было собирать статистику на android устройстве, а также строить графики и выводить нотификации(Глава 1, пункт 1.8, пункт 1.9).

1.1 Статистический анализ экспериментальных данных

Вариантой называют каждое уникальное значение в последовательности. Частотой называют количество повторений варианты.
Вариационный ряд [1] – это отношение варианты(V) и частоты(P) (1.1).

$$M = \frac{V}{P} \tag{1.1}$$

$$\sum P_i = n \tag{1.2}$$

Где n - это общее число наблюдений.
Из формулы (1.1) видно, что вариационный ряд — это статистический ряд, показывающий распределение изучаемого явления по величине какого-либо количественного признака.
Виды вариационных рядов:

1. Простой - в таком ряду каждая варианта встречается только один раз. В этом случае все частоты равны единице
2. Взвешенный - в таком ряду одна или несколько вариант встречаются неоднократно.

Вариационный ряд изображается в виде таблицы, графика или гистограммы.

x_i	15	18	19	20	21	22	23	24	25	26
m_i	2	4	2	4	3	1	5	4	3	2

Рис. 1.1: Табличное представление вариационного ряда

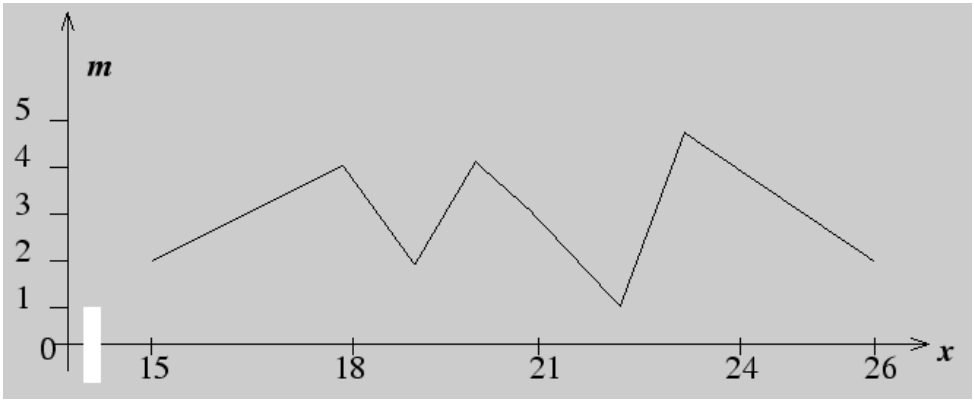


Рис. 1.2: Графическое представление вариационного ряда

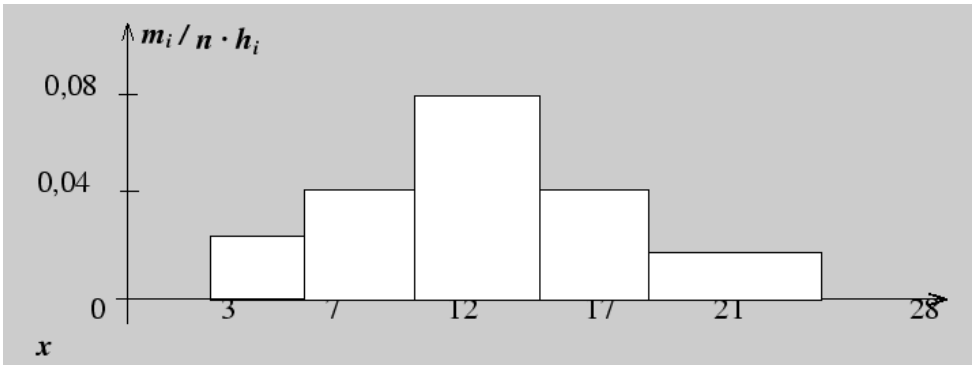


Рис. 1.3: Гистограммное представление вариационного ряда

Статистическое определение вероятности применимо не к любым данным. Они должны соответствовать некоторым свойствам:

А) Рассматриваемые данные должны быть исходами именного того теста, который может быть воспроизведен неограниченное число раз и при тех же условиях.

Б) Число тестов должно быть как можно больше. Если число тестов велико, то мы можем говорить не только о вероятности появления события, но и об относительной частоте. Вариационный ряд будет использоваться для вычисления вероятности разрядки батареи.

1.2 Марковские цепи

"Цепью Маркова называют такую последовательность случайных событий, в которой вероятность каждого события зависит только от состояния, в котором процесс находится в текущий момент и не зависит от более ранних состояний"[7].

$$P(X_n = j | X_0 = k_0, ..., X_{n-1} = i) = P(X_n = j | X_{n-1} = i) = p_{ij}^{(n)} \quad (1.3)$$

Марковская цепь называется однородной, если

$$p_{ij}^{(n)} = p_{ij} \quad (1.4)$$

Марковскую цепь представляют в виде графа переходов. Вершины графа соответствуют состояниям цепи, а дуги - это переходы между вершинами.

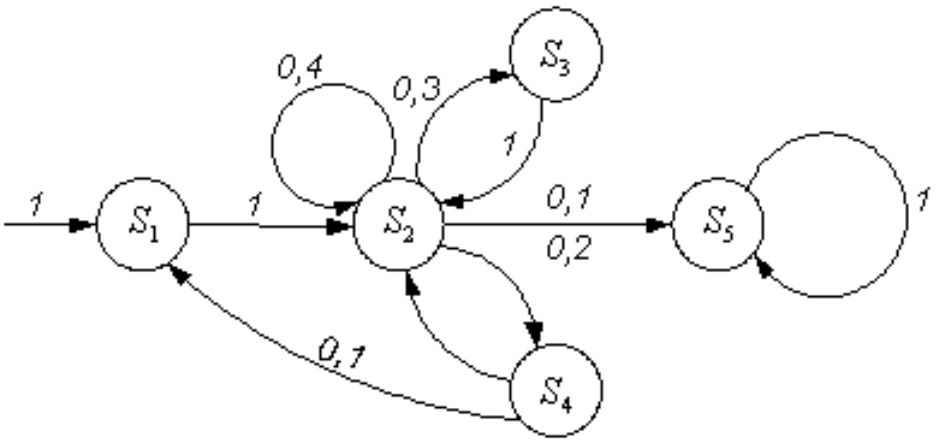


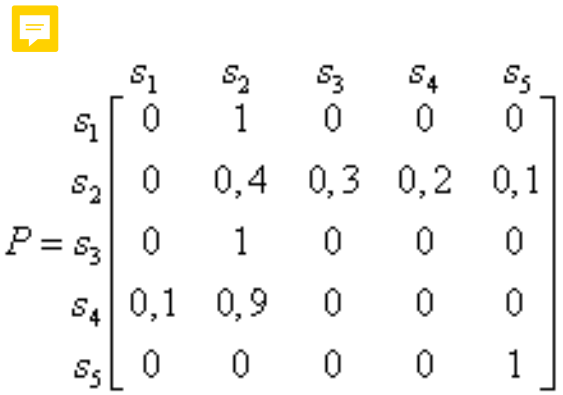
Рис. 1.4: Граф

Конечная дискретная цепь определяется:

1. Множество состояний S = {S_1,...S_n} - переход из одного состояния в другое.

2. Вектора начальных вероятностей q^0 = {q^0(1),...,q^0(n)} - определяющий вероятности того, что в момент t = 0 процесс находился в состоянии S_i.

3. Матрица переходных вероятностей P $= \{P_{ij}\}$ - характеризующая вероятности перехода процесса из текущего состояния S_i в состояние S_j.



$$P = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{array} \begin{array}{c} \begin{array}{ccccc} s_1 & s_2 & s_3 & s_4 & s_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0,4 & 0,3 & 0,2 & 0,1 \\ 0 & 1 & 0 & 0 & 0 \\ 0,1 & 0,9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Рис. 1.5: Пример матрицы переходных вероятностей

$$\sum_{j=1}^{n} P_{ij} = 1 \tag{1.5}$$

С помощью вектора начальных вероятностей и матрицы переходных вероятностей можно вычислить вектор q^n - вектор вероятностей, что процесс окажется в состоянии j в момент времени t = n с помощью формулы

$$q^n = q^0 * P^n \tag{1.6}$$

1.3 Бинарное возведение в степень

"Бинарное (двоичное) возведение в степень — это приём, позволяющий возводить любое число в n-ую степень за $O(\log n)$ умножений (вместо n умножений при обычном подходе)"[8].
Алгоритм применим к любым ассоциативным операциям. Операция называется ассоциативной, если для любых a, b, c выполняется

$$(a * b) * c = a * (b * c) \tag{1.7}$$

Сам алгоритм представляет собой снижение порядка степени при помощи ассоциативного свойства.
Для любого числа a и четного числа n, выполняется следующее тождество:

$$a^n = (a^{n/2})^2 = a^{n/2} * a^{n/2} \tag{1.8}$$

Если степень n не четна, то мы переходим к n - 1 степени, которая четна

$$a^n = (a^{n-1}) * a \tag{1.9}$$

и в конечном итоге, для не четной n степени это будет выглядеть так:

$$a^n = (a^{(n-1)/2}) * a * (a^{(n-1)/2}) * a \tag{1.10}$$

1.4 Проверка на равномерность

Последовательность x_1,...,x_n проверяется на равномерность [5]. Множество значений полученной последовательности разбивается на k непересекающихся интервалов. Критерий Пирсона (критерий χ^2), основанный на группированных данных. Весь отрезок значений [0, 1] делится на k интервалов равной длины. Обозначим через n_i число вариант (элементов выборки), попавших в i-ый интервал, $i = 1, ..., k$. Очевидно, что $n_1 + n_2 + ... + n_k = n$. Поскольку идет сравнение с равномерным распределением, теоретическая вероятность попадания чисел в i-ый интервал $p_i = 1/k$. Статистика критерия вычисляется по

следующей формуле

$$\chi_n^2 = \sum_{n=1}^{k} \frac{(n_i - np_i)^2}{np_i} \tag{1.11}$$

Если $\chi_n^2 > \chi_m^2$, то создается новый оптимальный шаблон.

1.5 Метод обнаружения нарушения процесса и выявление разладок

Контрольная карта (карта Шухарта) [3] это график, построенный на основании данных измерений показателей процесса в различные периоды времени. Он позволяет отразить динамику изменений показателя и за счет этого контролировать процесс.

Рис. 1.6: Пример карты Шухарта

Карты Шухарта включает с себя перечень установленных ГОСТом [2] правил:

1. Выход точек за контрольные границы.

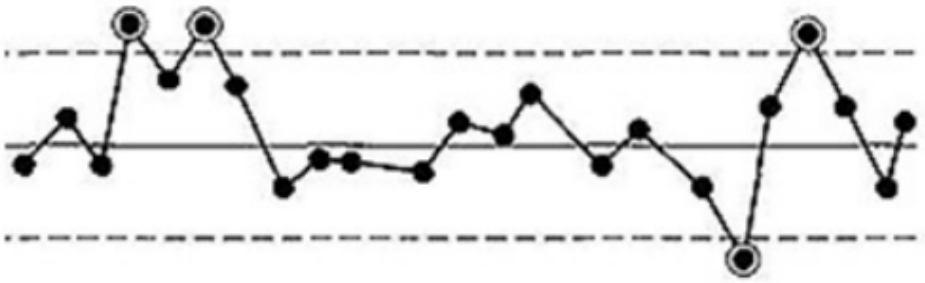


Рис. 1.7: Выход точек за контрольные границы

2. Серия – набор последовательных точек, находящихся по одну сторону от среднего уровня. Сигнализирующий о нарушении рассматривается серия длиной от 7 точек. Также настораживающими считаются ситуации, когда по одну сторону от среднего уровня оказываются 10 из 11, 12 из 14 или 16 из 20 точек.

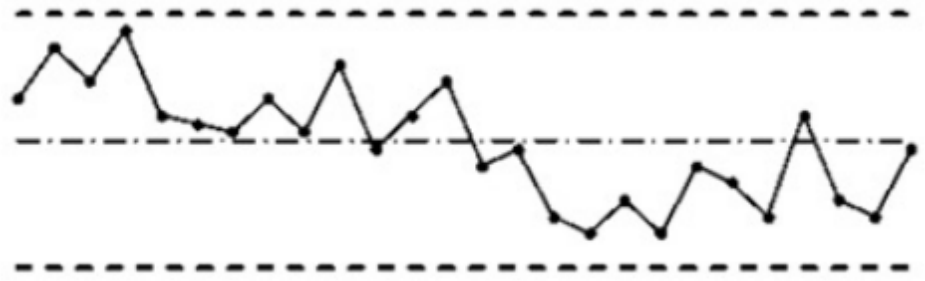


Рис. 1.8: Серия

3. Тренд (дрейф) - набор точек, образующий непрерывно повышающуюся или понижающуюся структуру.

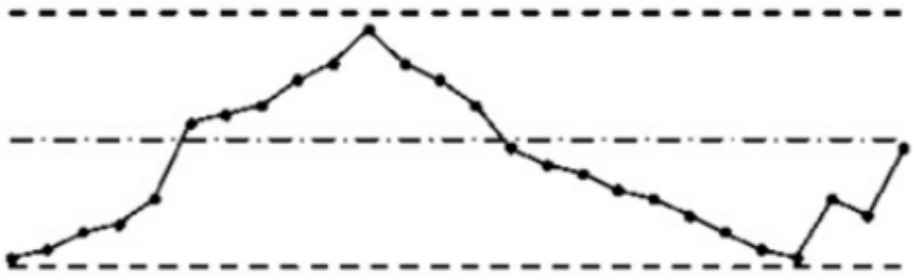


Рис. 1.9: Тренд

4. Приближение к контрольным пределам: ненормальным считается случай, когда две из трех последовательных точек оказываются за двухсигмовыми границами.

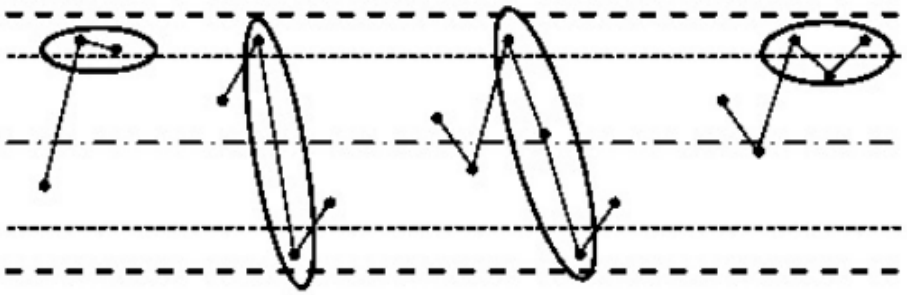


Рис. 1.10: Приближение к контрольным пределам

5. Приближение к центральной линии. Ситуация, когда большинство точек концентрируется в полосе между 1,5-сигмовыми пределами, указывает, скорее всего, на неподходящее разбиение данных на подгруппы.

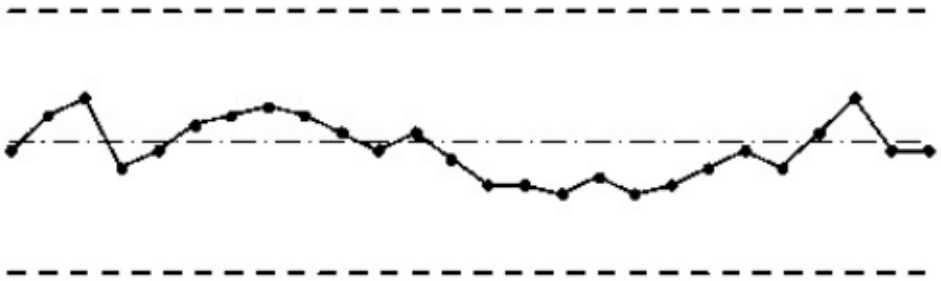


Рис. 1.11: Приближение к центральной линии

6. Периодичность. Ненормальной также считается состояние, когда через примерно равные интервалы времени чередуются спады и подъемы.

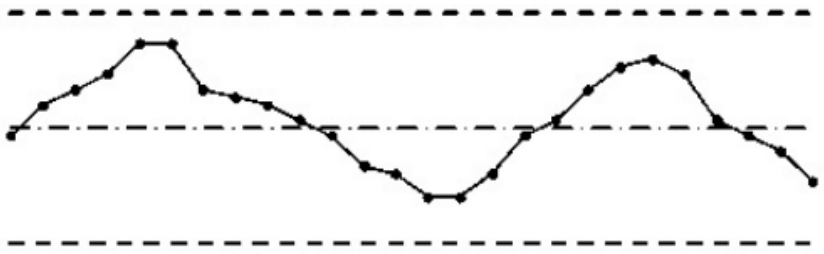


Рис. 1.12: Периодичность

1.6 Специальные библиотеки C++

Windows.h – это специальная библиотека с широким набором функций операционной системы. Является самым прямым способом взаимодействия приложений с ОС Windows. Она включает в себя такие функции как: работа с графическим интерфейсом, файлами, сетью, звуком и другие, а также нужная нам функция - GetSystemPowerStatus. Была изучена часть библиотеки windows.h для разработанного прототипа приложения.
Функция GetSystemPowerStatus(&status) – возвращает данные в переменную status, которая ссылается на структуру SYSTEM POWER STATUS [4].

В этой структуре содержаться такие поля как:
ACLineStatus; - Подключение к сети переменного тока
BatteryFlag; - Состояние батареи (уровень заряда и прочее)
BatteryLifePercent; - Оставшийся ресурс батареи в процентах
BatteryLifeTime; - Оставшееся время работы батареи (в сек.)
BatteryFullLifeTime; - Полное время работы батареи (в сек.)
Прототип извлекает нужные нам данные из структуры обращаясь к ней
-status.BatteryLifePercent, предварительно проверив флаг BatteryFlag с числом 128 (признак отсутствия батареи).

1.7 Классы QWidget и QDesktopWidget

Классы QWidget и QDesktopWidget - требуются для создания объектов пользовательского интерфейса (Рис. 1.13). Они имеют множество функций и флагов, нужных для создания уведомлений, например:

1. adjustSize() - эта функция подгоняет размеры окна уведомления под содержимое.
2. setAttribute(key, value) - эта функция изменяет атрибуты окна.
3. setStyleSheet() - эта функция изменяет стиль компонентов.
4. setWindowFlags(flags) - задает флаги виджета.
5. Tool - флаг сообщает, что окно является панелью инструментов (Отменяем показ в качестве отдельного окна).
6. WindowStaysOnTopHint - флаг сообщает, что уведомление будет поверх всех окон.

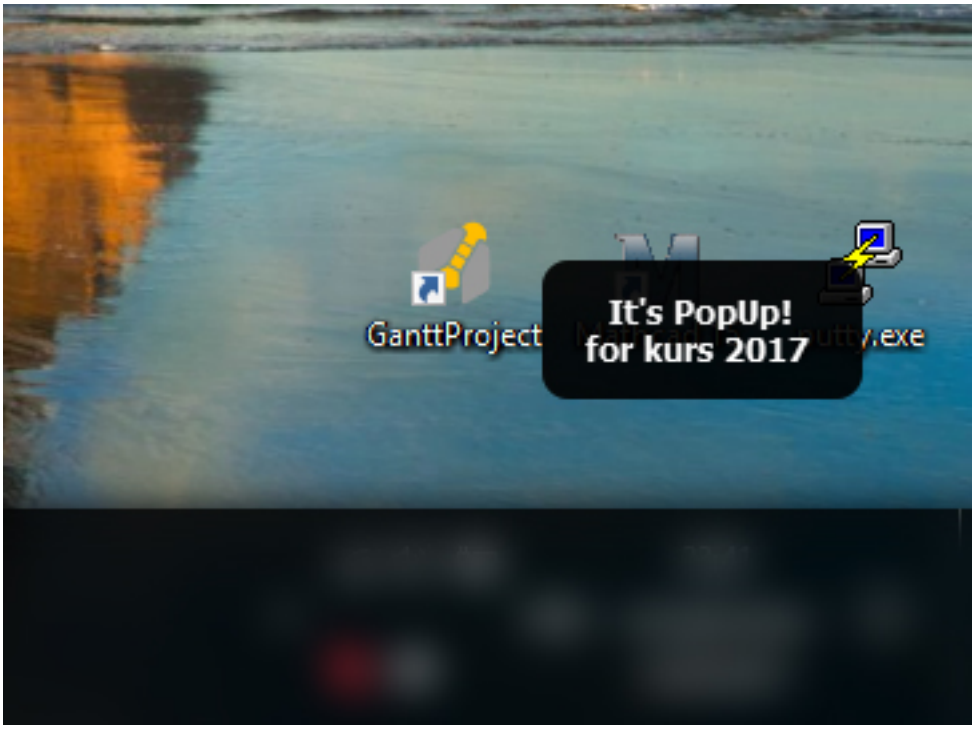


Рис. 1.13: Пример всплывающего уведомления.

1.8 Класс BatteryManager в Android Studio

Класс BatteryManager позволяет работать с методами для запроса свойств и состояния заряда батареи. Он имеет множество методов, например:

1. ACTION_CHARGING - показывает, что батарея начала заряжаться или устройство включено в сеть.
2. ACTION_DISCHARGING - показывает, что батарея начала разряжаться, можно использовать для оптимизации приложений, исключая лишние активности в памяти устройства
3. EXTRA_STATUS - возвращает в каком состоянии сейчас батарея (заряжается, разряжается).
4. EXTRA_LEVEL - возвращает текущий уровень заряда батареии.

1.9 Класс NotificationManager в Android Studio

Класс NotificationManager позволяет работать с методами, для извещения пользователя, об изменениях в фоновых процессах, посредством уведомлений. Уведомления можно гибко настраивать посредствам разных методов. Задавать время прихода, задавать приоритет уведомления или можно сделать так, чтобы уведомление всегда висело в панели задач и обновлялось время от времени. Примеры методов:

1. cancel(int id)/cancelAll() - скрывает предыдущее уведомление от этого же приложения. По определенному идентификатору или сразу все.

2. createNotificationChannel(NotificationChannel channel) - создает канал для уведомлений, на который они потом приходят.

3. notify(String tag, int id, Notification notification) - отправляет уведомление, которе будет отображаться в панели уведомлений.

4. areNotificationsEnabled() - проверяет включены ли уведомления на устройстве.

5. shouldHideSilentStatusBarIcons() - проверяет приходят ли уведомления в тихом режиме.

6. IMPORTANCE_HIGH - задает высший приоритет для нотификации (можно изменить HIGH - на любой другой приоритет).

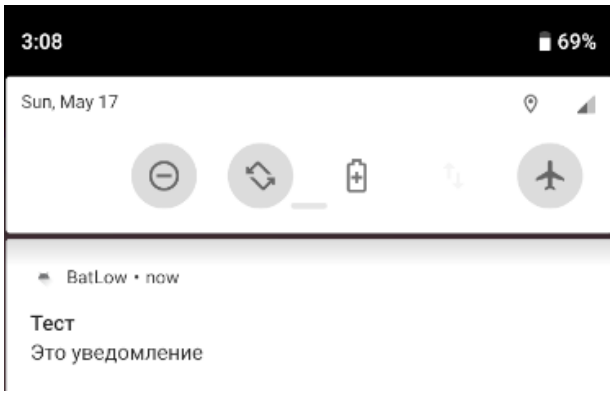


Рис. 1.14: Пример уведомления на устройстве.

Глава 2

Постановка задачи

2.1 Требования к прототипу

Для работы с прототипом приложения:

1. Требуется скачать exe/apk файл на устройство, которое поддерживает прототип.
2. Далее его нужно запустить на устройстве отключенном от сети питания.

2.2 Общие требование к работе приложения

1. Приложение в фоновом режиме должно начать собирать данные о состоянии заряда батерии и времени, когда был осуществлен замер.
2. Замеры производятся с постоянной периодичностью.
3. Все замеры структурировано логируются в журнал, который представляет из себя набор файлов.
4. Замеры передаются в алгоритмы обработки.
5. Должна начаться обработка информации и анализ данных.
   5.1. Считается вероятность разрядки батареи.
   5.2. С определенной периодичностью выводится сообщение с прогнозом.

5.3. Идет поиск отклонений разрядки батареи в реальном времени.

5.4. Если произошло более 5 разладок, выводится сообщение с предупреждением об отклонении энергопотребления от нормы.

6. Постройка графиков с визуальным представлением прогноза о разрядке батареи.

2.3 Требования к интерфейсу

2.3.1 Всплывающие уведомления на ноутбуке

1. Появляется при каждом уменьшении заряда батареи на 10%.
2. Не должны быть навязчивыми.
3. Не должны перекрывать основные элементы рабочего стола.
4. Разборчивый шрифт и цвет текста.
5. Уведомление не должно висеть слишком долго на экране.
6. Содержание уведомления должно быть лаконичным и понятным для любого пользователя.

2.3.2 Пользовательский интерфейс на ноутбуке

1. Отдельное окно, открывающееся при нажатии на иконку приложения.
2. Содержит график с визуальным представлением прогноза о разрядке батареи, который будет понятен любому пользователю.
3. Содержит кнопки выбора режимы работы с устройством (например режимы: автоматический, игры, работа, интернет).

2.3.3 Уведомления на смартфоне/планшете на базе android

1. Появляется первый раз при запуске приложения, а далее обновляется при каждом уменьшении заряда батареи на 10%.
2. Разборчивый шрифт и цвет текста.
3. Уведомление должно все время висеть в панели уведомлений - его нельзя закрыть. Для того, чтобы пользователю не нужно было каждый раз разворачивать приложение.
4. Содержание уведомления должно быть лаконичным и понятным для любого пользователя.

2.3.4 Пользовательский интерфейс на смартфоне/планшете на базе android

1. Вся информация распологается на главном экране.
2. Содержит график с визуальным представлением прогноза о разрядке батареи, который будет понятен любому пользователю.
3. Содержит кнопку сброса всех данных.

2.4 Требования к алгоритмам

1. Все входные данные должны браться из системы.
2. Алгоритмы должны обрабатывать и выводить данные автоматически, не требуя от пользователя никаких дополнительных действий, за исключением того, что пользователь может выбрать режим работы за устройством.

2.5 Перечень алгоритмов

1. Алгоритм создания логфайлов с уникальными именами для ведения журнала.

2. Алгоритм сбора информации из системы.

3. Алгоритм начальной обработки считанных данных и их структуризация для дальнейших углубленных расчетов и прогнозов.

4. Алгоритм углубленной обработки информации и построения прогнозов на основе аппарата теории марковских цепей.

5. Алгоритм структуризации данных полученных после обработки, для дальнейших поисков разладок.

6. Алгоритм для поиска разладок.

2.6 Перечень интерфейсов

2.6.1 Перечень интерфейсов на ноутбуке

1. Интерфейс выводящий всплывающие уведомления с прогнозом о разрядке.

2. Интерфейс всплывающих уведомлений с произошедшей разладкой.

3. Интерфейс строящий график наглядно показывающий прогноз разрядки и позволяющий выбрать режим использования устройства.

2.6.2 Перечень интерфейсов на смартфоне/планшете на базе android

1. Интерфейс уведомления с прогнозом о разрядке.

2. Интерфейс уведомлений с произошедшей разладкой.

3. Интерфейс строящий график наглядно показывающий прогноз разрядки и позволяющий выбрать режим использования устройства.

4. Поясняющий текст для прогноза и отклонений от нормы.

Глава 3

Реализация

3.1 Разработка алгоритма начального анализа эксперементальных данных

1. Журнал

   При каждом запуске, программа создает файлов(лог) с уникальным названием. Это сделано для того, чтобы сохранить результаты каждого запуска приложения, так как для индивидуального прогноза требуется не менее 10 тестов.

2. Считывание

   Программа считывает состояние батареи в процентах. Программно задается частота замеров (через какое время будет проводиться замер).

3. Прогноз

   Программа строит вариационные ряды и вычисляет вероятность разрядки на основе полученных данных за несколько тестов. Чем больше тестов будет проведено, тем точнее будет прогноз.

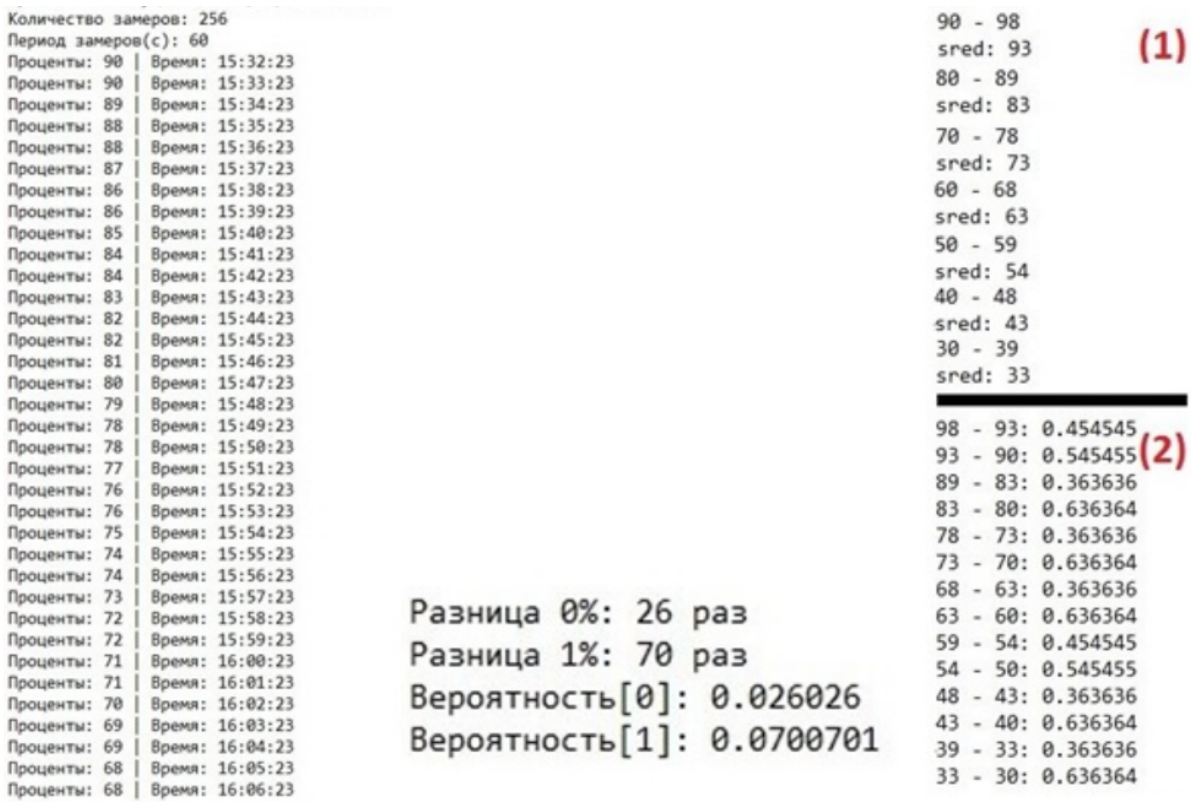


Рис. 3.1: Примеры выходных файлов: Табл.1, табл.2, табл.3

Пояснения к рисункам: В левой части (Рис. 3.1) изображена таблица 1. В левом столбце указаны проценты заряда батареи. В правом столбце указано текущее на момент замера процентов время. На средней части (Рис. 3.1) изображены частоты, построенные на основе данных взятых из таблицы 1. В нем показано: в первых двух строках (их может быть больше, в зависимости от разницы разрядки, количества замеров и периода замеров) указана разница между каждым шагом из таблицы. Следующие строки показывают вероятность того, на сколько будут с каждым замером меняться проценты. В правой части (Рис. 3.1) изображен вариационный ряд, построенный на основе данных взятых из таблицы 1. В нем показано: в первой части таблицы min – max значения времени разрядки от 90% до 30% / от 80% до 30%... Затем это время разделено на интервалы и построен вариационный ряд - с какой вероятность батарея разрядится от max до min времени. Было проделано 10 тестов, для того, чтобы построить вариационный ряд на (Рис. 3.1)

3.2 Разработка алгоритма углубленного анализа эксперементальных данных

Первые 10 замеров будут обрабытываться с помощью алгоритма начального анализа, пока не наберется достаточное количество данных. Весь анализ и построение прогноза строится на основе обычного вариационного ряда. В данном алгоритме прогноз строится с помощью Цепей Маркова.

Построена матрица перехода P_{ij} на основе вероятностей разряда за определенный промежуток времени замера.

```
Разница0 13
Разница1 49
Разница2 6
Разница3 1
Разница4 0
Разница5 1

Вероятность[0] 0.183099
Вероятность[1] 0.690141
Вероятность[2] 0.084507
Вероятность[3] 0.0140845
Вероятность[4] 0
Вероятность[5] 0.0140845
```

Рис. 3.2: Данные для построения матрицы переходов

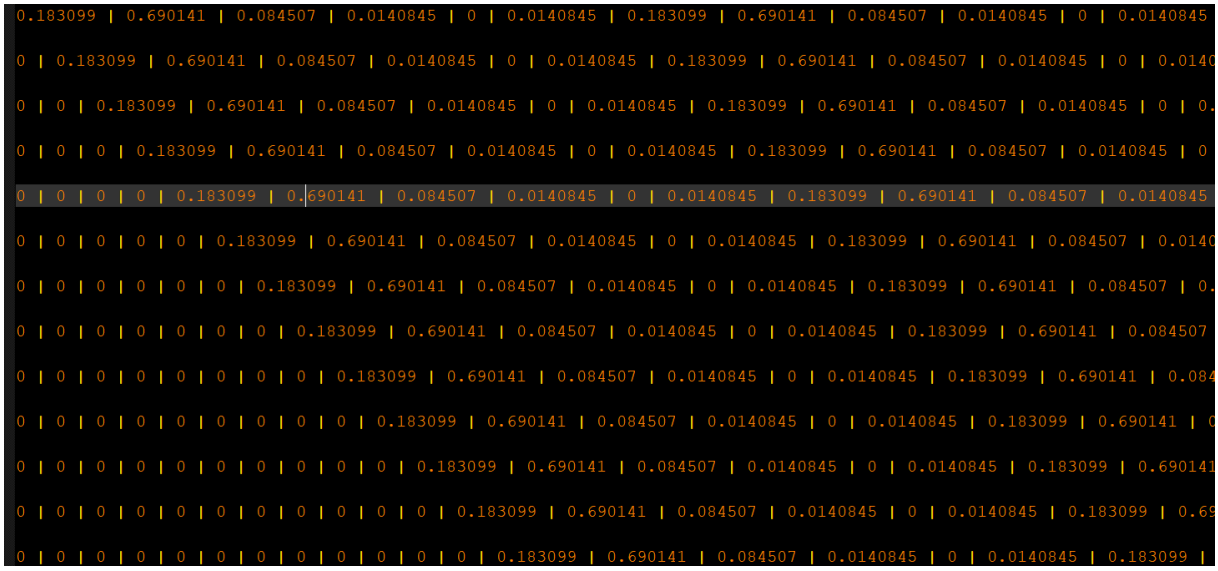


Рис. 3.3: Фрагмент матрицы перехода P_{ij}

За вектор начальных вероятностей берется столбец с вероятностями на (Рис. 3.2) т.е. в данном случае $q^0 = \{0.183099, 0.690141, 0.084507, 0.0140845, 0, 0.0140845\}$.
С помощью формулы (1.6) находим вероятности за какое время и на сколько разрядится устройство в момент t = n, выбирая максимальную вероятность из q^n - это и будет вероятность разрядки за некоторое время.

3.3 Разработка алгоритма поиска разладок

1. Обнаружение нарушения
   Программа обнаруживает расхождения с нормой. У каждого пользователя будет построена своя индивидуальная норма. Например, если в какой-то момент времени, батарея разрядилась быстрее, чем раньше, то это будет сообщено владельцу устройства.

2. Из-за сильного изменения шаблона энергопотребления, если в алгоритм подавался один и тот же шаблон, то он не верно выявлял и выводил разладки. Для этого была разработана защита от ложных срабатываний алгоритма разладок.
   Идея заключается в том, что запоминаются несколько индивидуальных шаблонов, в данный момент это:

   2.1. Автоматически - используется стандартный шаблон энергопотребления, прогноз строится на основании всех замеров, вне зависимости от того, какое было энергопотребление. В этом режиме сообщение о разладках выводится пользователю, если произошло 15 отклонений от нормы. Это сделанно для того, чтобы компенсировать большую разницу в рамках всех шаблонов.

   2.2. Игры - пользователь сильно нагружает систему играми и батарея быстро разряжается. Прогноз строится на основании замеров, которые проводились, когда пользователь выбирал

шаблон для игр.

2.3. Работа - пользователь средне нагружает систему программами для работы (офисные программы, среды разработки и т.п.). Батарея разряжается в среднем темпе. Прогноз строится на основании замеров, которые проводились, когда пользователь выбирал шаблон для работы.

2.4. Интернет - пользователь в стабильном режиме использует устройство для интернета. Батарея равномерно и не очень быстро разряжается. Прогноз строится на основании замеров, которые проводились, когда пользователь выбирал шаблон для интернета.

```
90 99 98 95 98 90 89 99 90 96 105 90 106 80 96 93 96 93 90 91 92
93 94 95 96 97 98 99 100 99 95 96 96 96 96 96 96 96 96 96
Норма: 94 Верхняя граница: 104 Нижняя граница: 84
Карты X

Произошла разладка - переодичность 99 - 8
Произошла разладка - выход за контрольные точки 105 - 11
Произошла разладка - выход за контрольные точки 106 - 13
Произошла разладка - приближение к контрольным границам 106 - 13
Произошла разладка - выход за контрольные точки 80 - 14
Произошла разладка - приближение к контрольным границам 80 - 14
Произошла разладка - приближение к контрольным границам 96 - 15
Произошла разладка - переодичность 93 - 16
Произошла разладка - монотонность 7 - 98 - 27
Произошла разладка - монотонность 8 - 99 - 28
Произошла разладка - монотонность 9 - 100 - 29
Произошла разладка - монотонность 8 - 99 - 30
Произошла разладка - монотонность 7 - 95 - 31
Произошла разладка - монотонность 8 - 96 - 32
Произошла разладка - монотонность 8 - 96 - 33
Произошла разладка - монотонность 8 - 96 - 34
Произошла разладка - серия 10 - 96 - 35
Произошла разладка - монотонность 8 - 96 - 35
Произошла разладка - серия 11 - 96 - 36
Произошла разладка - монотонность 8 - 96 - 36
Произошла разладка - серия 12 - 96 - 37
Произошла разладка - монотонность 8 - 96 - 37
Произошла разладка - серия 13 - 96 - 38
Произошла разладка - монотонность 8 - 96 - 38
Произошла разладка - серия 14 - 96 - 39
Произошла разладка - монотонность 8 - 96 - 39
```

Рис. 3.4: Пример выходного файла разладок карты X

```
Карты R
-9 1 3 -3 8 1 -10 9 -6 -9 15 -16 26 -16 3 -3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 4 -1 0 0 0 0 0 0 0 0
Норма: -1 Верхняя граница: 9 Нижняя граница: -11

Произошла разладка - приближение к контрольным границам -10 - 7
Произошла разладка - переодичность 9 - 8
Произошла разладка - приближение к контрольным границам 9 - 8
Произошла разладка - приближение к контрольным границам -6 - 9
Произошла разладка - приближение к контрольным границам -9 - 10
Произошла разладка - выход за контрольные точки 15 - 11
Произошла разладка - приближение к контрольным границам 15 - 11
Произошла разладка - выход за контрольные точки -16 - 12
Произошла разладка - приближение к контрольным границам -16 - 12
Произошла разладка - выход за контрольные точки 26 - 13
Произошла разладка - приближение к контрольным границам 26 - 13
Произошла разладка - выход за контрольные точки -16 - 14
Произошла разладка - приближение к контрольным границам -16 - 14
Произошла разладка - приближение к контрольным границам 3 - 15
Произошла разладка - переодичность 4 - 30
Произошла разладка - серия 10 - 0 - 37
Произошла разладка - серия 11 - 0 - 38
Произошла разладка - серия 12 - 0 - 39
```

Рис. 3.5: Пример выходного файла разладок карты R

На рисунках выше, изображены примеры сообщений, если произошло нарушение процесса. Построенная карта X (Рис. 3.4), основана на текущих значениях, в данном случае время разрядки от начала теста, до полной разрядки. Построенная карта R(Рис. 3.5) основана на размахах(разницы между минимальным и максимальным значениями) для каждого периода.

3.4 Разработка интерфейса

3.4.1 Интерфейсы на ноутбуке

1. Разработка прототипа интерфейса всплывающих уведомлений, которое информирует пользователя о вероятности времени разрядки батареи, с текущего значения заряда, до нуля.

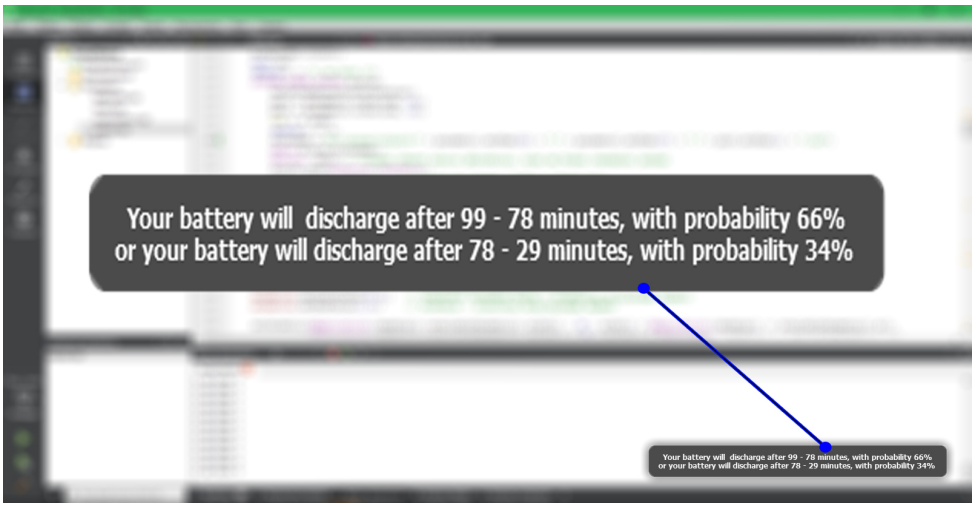


Рис. 3.6: Пример 1: Всплывающее уведомление, информирующее о том, за сколько минут разрядится батарея с данной вероятностью.

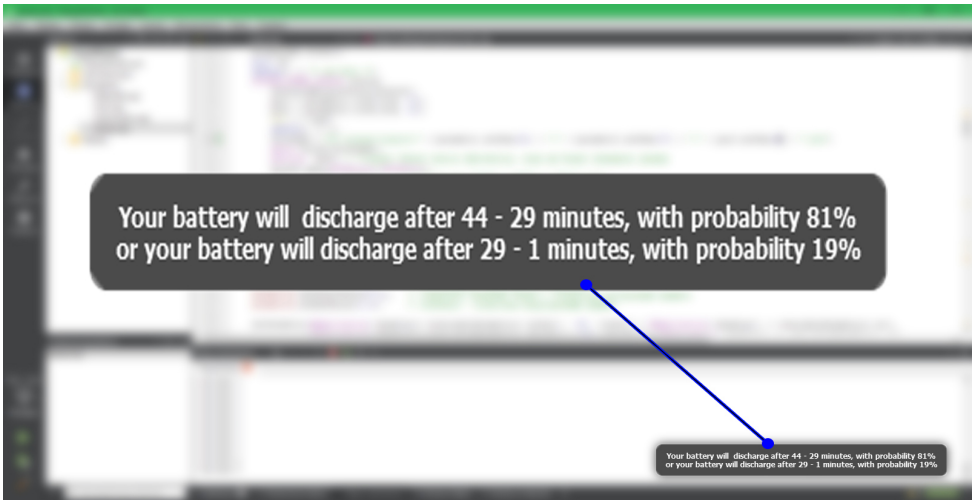


Рис. 3.7: Пример 2: Всплывающее уведомление, информирующее о том, за сколько минут разрядится батарея с данной вероятностью.

Пояснения к рисункам: уведомление показывается в нижнем правом углу экрана в момент, когда заряд батареи достигает определенного уровня, например, на (Рис. 3.6) показано время и веротяность разрядки в момент 80%, а на (Рис. 3.7) показано время и веротяность разрядки в момент 20%. Информация берется из заранее подготовленных логфайлов, которые строятся по индивидуальной статистике, собираемой и обрабатываемой алгоритмами описанными выше

2. Разработка прототипа интерфейса всплывающих уведомлений, которое информирует пользователя о произошедшей разладке (пользователь использует устройство не в том режиме, что обычно).

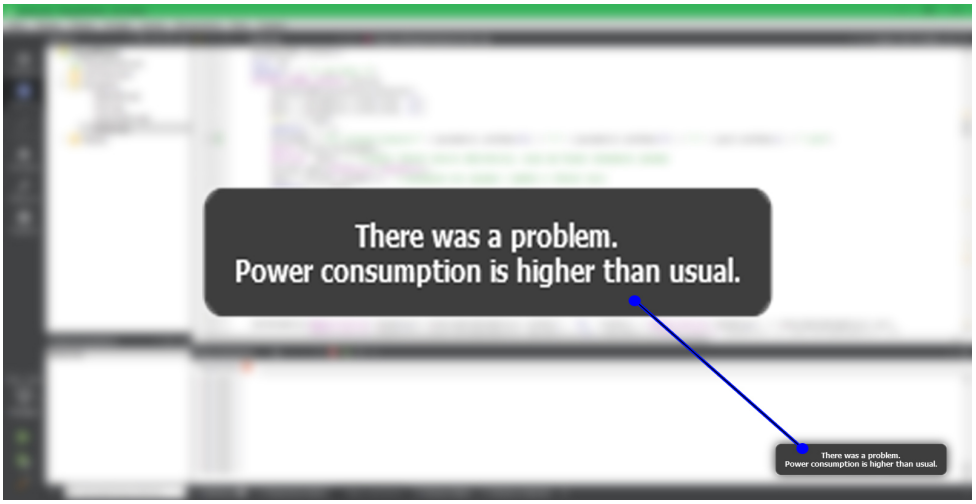


Рис. 3.8: Прототип всплывающего уведомления, информирующее о том, что энергопотребление выше, чем обычно.

3. Разработка интерфейса, строящего графическое представление прогноза разрядки батареи.

   3.1. График показывающий вероятности разрядки за некоторое время.

   3.2. Кнопки с выбором интенсивности энергопотребления (автоматический, игры, работа, интернет).

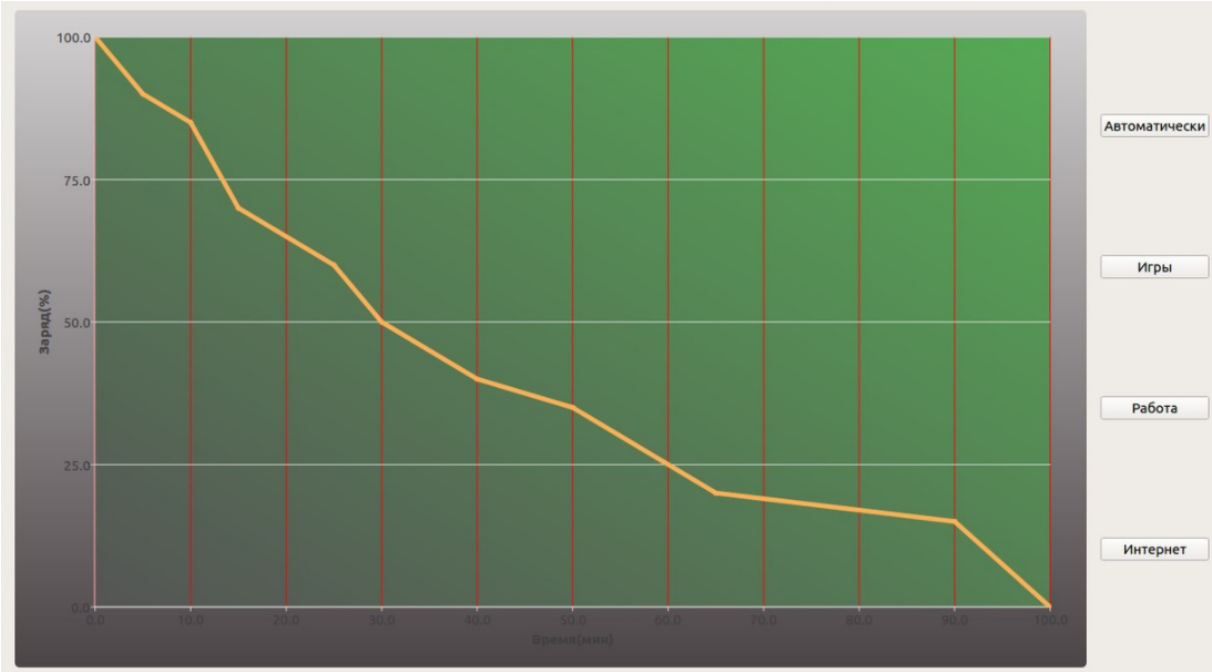


Рис. 3.9: Пример интерфейса для "автоматической"интенсивности энергопотребления

3.4.2 Интерфейсы на смартфоне/планшете

1. Разработка прототипа интерфейса на смартфоне/планшете. Интерфейс состоит из:

   1.1. Кнопка сброса всей статистики.

   1.2. Блок текста, который выводиться, если произошло отклонение от нормы.

   1.3. График, на котором изображен прогноз времени разрядки устройства от ста процентов до нуля.

   1.4. Блок текста, который показывает сколько осталось времени, от текущего количества заряда батареи.

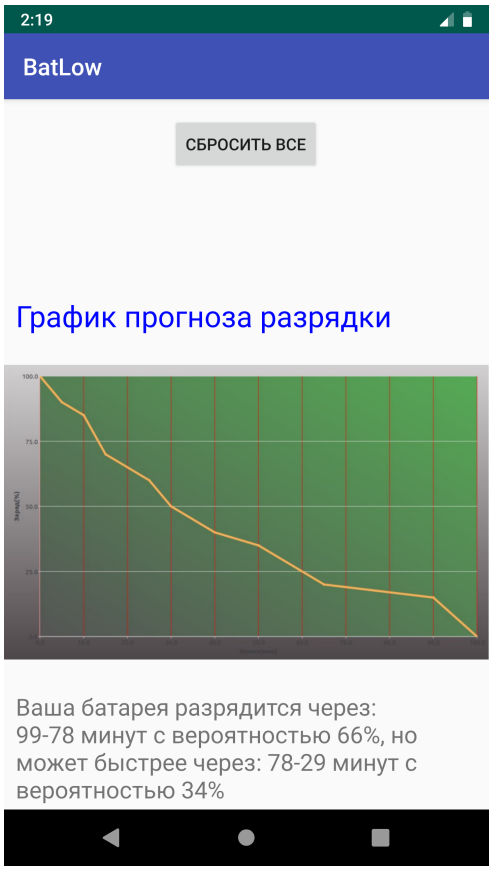


Рис. 3.10: Пример 1: Интерфейс на устройстве (смартфон), без сообщения об отклонении от нормы.

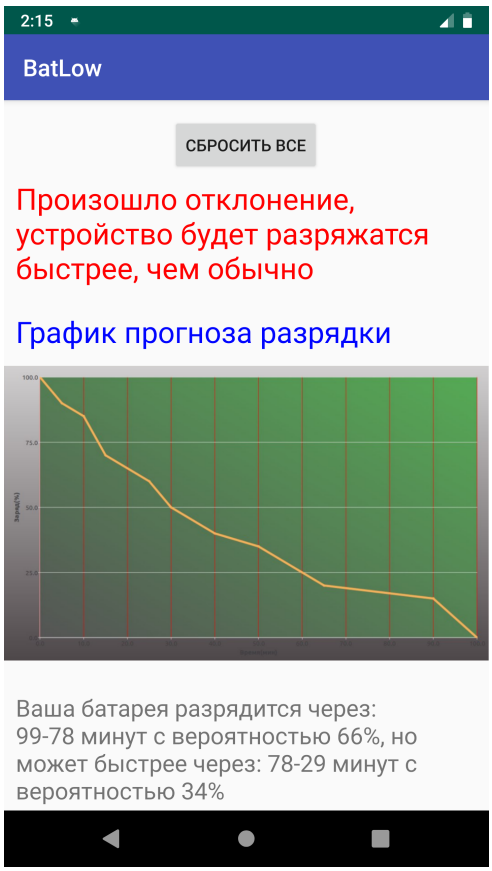


Рис. 3.11: Пример 2: Интерфейс на устройстве (смартфон), с сообщением об отклонении от нормы.

Пояснение к рисункам: На рисунке (Рис. 3.10) изображается пользовательский интерфейс на смартфоне. Элементы которые изображены:

1. Сверху кнопка сброса всей статистики. При нажатии на нее вся статистика очищается, все файлы с логированием становятся пустыми. Это нужно, например, при смене батареи, когда старая статистика становится неактуальной или если пользователь видит, что прогнозы стали сильно отклоняться от действительности.

2. График, на котором изображен прогноз времени разрядки устройства от ста процентов до нуля. График разбит на шаги в 10 процентов по которым строиться кривая, обозначающая время.

3. Ниже расположен блок текста, который показывает сколько осталось времени, от текущего количества заряда батареи. На данном

изображении говориться о том, что устройство будет разряжено за 99-78 минут, от текущего уровня зарядки, до нуля. Также представлен альтервантивный вариант, говорящий о том, что если использовать устройство интенсивннее, то и устройство разрядится быстрее.

На рисунке (Рис. 3.11) изображено то же самое, что и на (Рис. 3.10), но добавился блок текста, говорящий о том, что энергопотребление стало интенсивнее, чем обычно и что устройство может разряжаться быстрее.

1. Разработка уведомлений на смартфоне/планшете.

    1.1. Уведомление с прогнозом постоянно находится в панели уведомлений, для того, чтобы пользователю не нужно было каждый раз разворачивать приложение.

    1.2. Такое уведомление каждый раз обновляется, при снижении заряда батареии на 10 процентов и выводит новые данные с прогнозом.

    1.3. Уведомление состоит из:

        i. Заголовок - короткое сообщение о том, какого рода уведомление пришло - Статус разрядки

        ii. Ниже разположен блок текста, который развернуто говорит о том, сколько времени осталось до разрядки.

    1.4. Уведомление об отклонении от нормы приходит только при фактичесеском отклонении. Если энергопотребление выше, чем обычно, то уведомлнение будет содержать информацию об этом. Если ниже, то будет говориться о том, что энергопотребление ниже.

    1.5. Уведомление состоит из:

        i. Заголовок - короткое сообщение о том, какого рода уведомление пришло - отклонение от нормы энергопотребления

        ii. Ниже разположен блок текста, который говорит о том, что энергопотребление отличается от типичного.

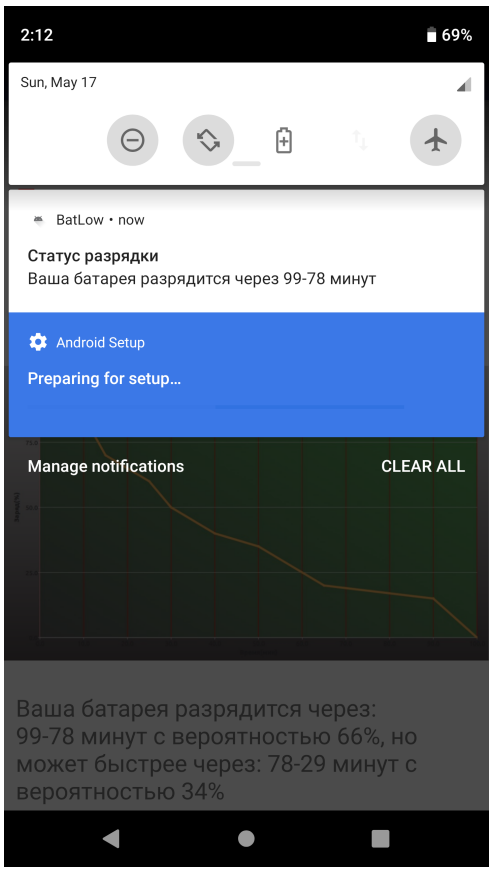


Рис. 3.12: Пример 1: Нотификация сообщающая о том, через сколько времени разрядится устройство.

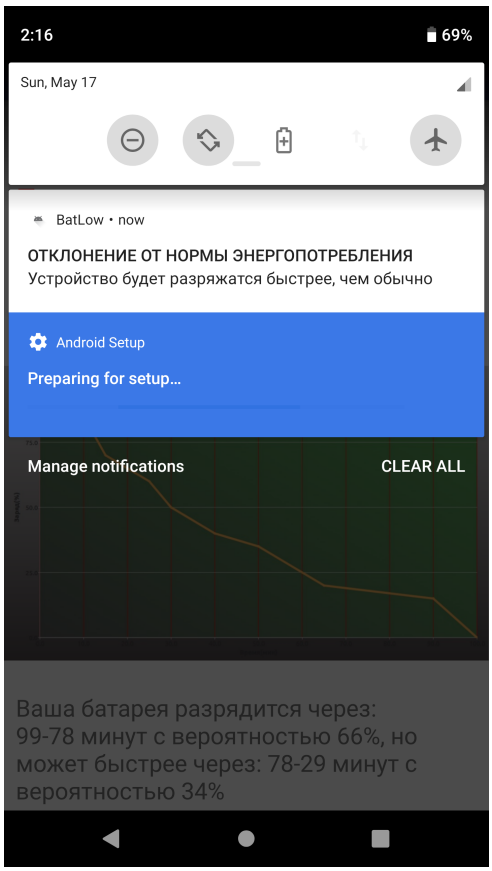


Рис. 3.13: Пример 2: Нотификация сообщающая о том, что энергопотребление не типично и устройство может разрядиться быстрее

Пояснение к рисункам: На рисунке (Рис. 3.12) изображается уведомление, коворящее пользователю о том, что батарея разрядится за 99-78 минут. На рисунке (Рис. 3.13) изображено пришедее уведомление, в котором говориться о том, что устройство будет разряжаться быстрее, чем обычно.

3.5 Сложности

Попытка использования проверки на равномерность не увенчалась успехом. На некоторых данных алгоритм показывал верный результат, (например, при использовании устройства в одном и том же режиме на протяжении всей сессии, прототип приложения ведет замеры и получается равномерное распределение). Но если в рамках одной сессии использование устройства было не равномерным, (система то нагружена, то нет) то алгоритм выдавал не верный результат, из-за чего пришлось отказаться от этой системы.

3.6 Тестирование и эксперименты

Проводилось тестирование на личном ноутбуке ASUS - K750J. Также проводилось тестирование на смартфоне LG nexus 5x. Все результаты тестов можно увидеть выше и в Приложении.

Архитектура разработанного прототипа приложения:

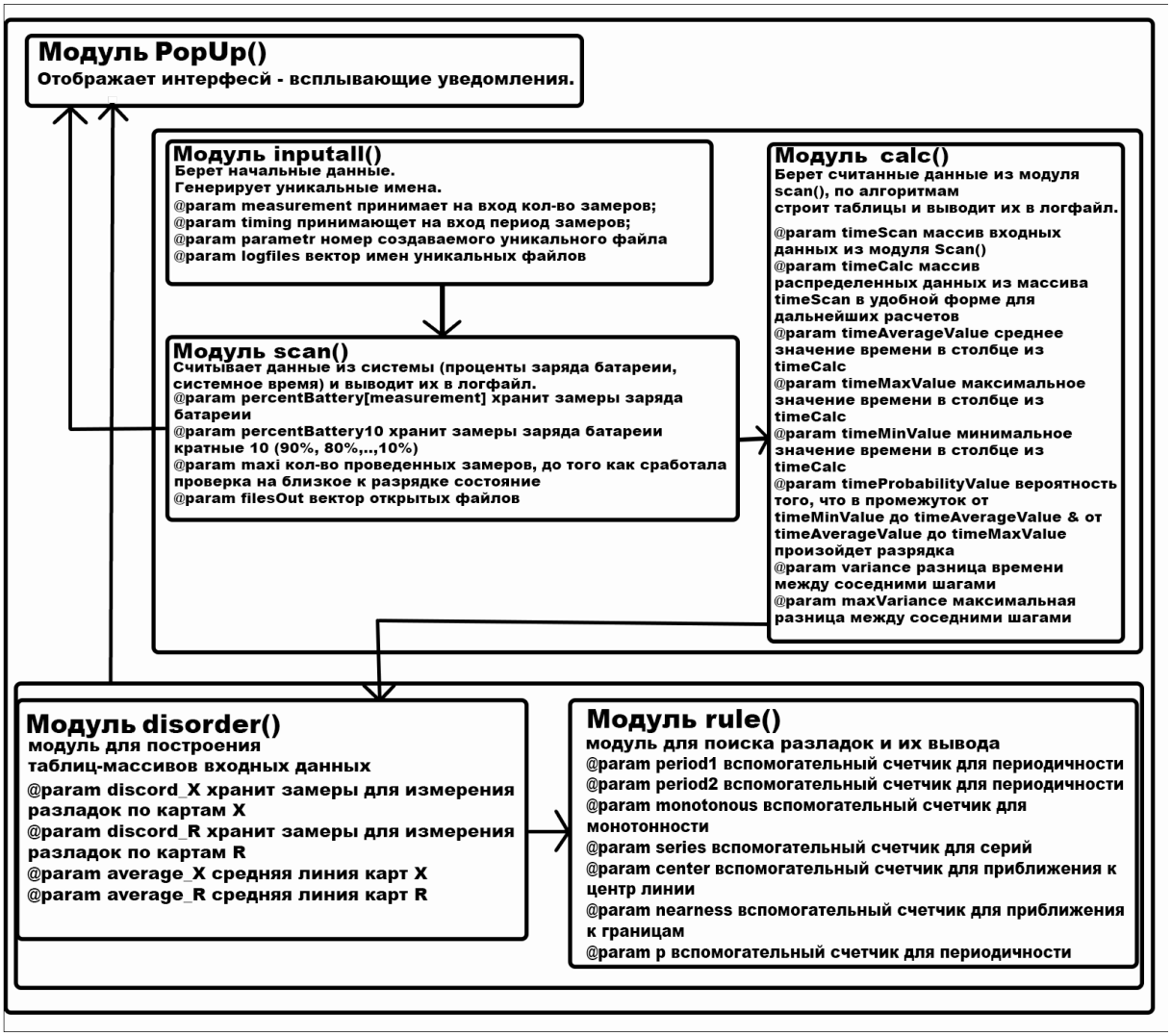


Рис. 3.14: Архитектура прототипа приложения персонализированного анализа энергопотребления мобильных устройств.

Краткое описание работы модулей:

- Основные модули, которые использовались в прототипе для ноутбуков:
  - Модуль inputall() - этот модуль берет начальные данные, такие как период и количество замеров. Также в этом модуле создаются файлы с уникальными именами под журнал.
  - Модуль scan() - этот модуль отвечает за сбор всех данных, считывает заряд батареи в процентах и соответствующее этому состоянию время. Также в этом модуле определяется с ка-

кой интенсивностью используется устройство в данной итерации.

– Модуль calc() (calcEasy()) - модуль отвечающий за расчет первых 10 прогнозов на основе построения вариационного ряда. Берет входные данные считанные в модуле scan() и строит таблицы нужные для использования в дальнейших прогнозах. На выходе записывает все данные в логфайлы, которые будут использованы в модулях calcHard() и disorder(). Также первые 10 прогнозов появляющиеся с помощью модуля PopUp().
– Модуль disorder() - модуль предназначен для структурирования данных, по которым будут искаться разладки.
– Модуль rule() - в этом модуле происходит расчет и логирование найденных отклонений от нормы (разладок). На выходе данные передаются в модуль PopUp() при условии, что произошло более 5 разладок, так как это будет означать, что произошла смена схемы энергопотребления.
– Модуль PopUp() - модуль отвечающий за вывод реализованных всплывающих уведомлений, таких как произошедшие разладки, прогноз времени разрядки батареи.
– Модуль gui() - модуль предназначенный для отображения графического интерфейса и отображения графика с прогнозом разрядки батареи (модуль не изабражен на архитектуре, так как не интегрирован в прототип, а работает автономно).
– Модуль calcHard() - модуль отвечающий за более глубокий и точный расчет прогнозов. На основе аппарата теории Марковских цепей строится прогноз с вероятностью разрядки за некоторое время. На выходе данные записываются в логфайл, на основе которого будет строится и выводится график в модуле gui() (модуль не изабражен на архитектуре, так как не интегрирован в прототип, а работает автономно).

- Дополнительные модули разработанные в рамках работы над мобильным прототипом приложения:
  - notificationBuilder() - модуль предназначенный для отправки уведомлений в нужный момент.
  - createChannelNotiff() - модуль предназначенный для создания каналов для уведомлений. Разныне каналы нужны для того, чтобы задать приоритет нотификациям. У уведомлений сообщающих об отклонении от нормы приоритет выше, чем у уведомлений сообщающих о времени разрядки.
  - fileDirectory() - модуль отвечающий за операции всей файловой системы, создания лог-файлов, хранения и упорядочивания в ней информации.
  - calcGraph() - модуль отвечает за создание данных для графика с прогнозом разрядки.
  - drawUI() - модуль отвечающий за создание всего пользовательского интерфейса, трисовку кнопок, текста и графика.

Глава 4

Результаты

4.1 Результаты

1. Разработаны алгоритмы прогнозирования
   1.1. Алгоритм начального анализа эксперементальных данных
   1.2. Алгоритм углубленного анализа эксперементальных данных
2. Разработан алгоритм поиска разладок
3. Разработан прототип интерфейса для ноутбуков
   3.1. Прототип интерфейса всплывающих уведомлений, которое информирует пользователя о вероятности времени разрядки батареи, с текущего значения заряда, до нуля.
   3.2. Прототип интерфейса всплывающих уведомлений,которое информирует пользователя о произошедшей разладке (пользователь использует устройство не в том режиме, что обычно).
   3.3. Интерфейс, строящий графическое представление прогноза разрядки батареи, с возможностью выбора шаблона энергопотребления.
4. Разработан прототип интерфейса для смартфонов/планшетов на базе OS android

4.1. Уведомления, которое информирует пользователя о вероятности времени разрядки батареи, с текущего значения заряда, до нуля.

4.2. Уведомления,которое информирует пользователя о произошедшей разладке (пользователь использует устройство не в том режиме, что обычно).

4.3. Интерфейс, строящий графическое представление прогноза разрядки батареи, а также текстовые пояснения. С возможностью сброса всей статистики.

4.2 Возможные улучшения

1. Тестирование на большем количестве устройств. Это нужно для более глубокой оценки точности алгоритмов и дальнейшей их модернизации. Трудно проводить большой объем различных тестов для покрытия всех возможных шаблонов из-за того, что каждый тест очень времязатратный.

2. Расширять алгоритмическую базу для разладок. На основе большого количества тестов можно выявить дополнительные и более функциональные алгоритмы нахождения нарушений, которые нужно реализовать для более точной работы приложения.

3. Создание кроссплатформенного приложения. Разработанное приложение будет также актуально для мобильной платформы IOS.

4. Проработка пользовательского интерфейса на мобильном устройстве. Сейчас интерфейс далек от идеала, его можно улучшить.

Приложение

```
SYSTEM_POWER_STATUS status;
    GetSystemPowerStatus(&status);
    etalon = status.BatteryLifePercent;
    for (int i = 0; i < measurement; i++) // Считывает данные о заряде батареии
    {
        time_t t = time(NULL);
        tm* aTm2 = localtime(&t);
        GetSystemPowerStatus(&status); // Возвращает данные о заряде батарее в переменную status
        percentBattery[i] = status.BatteryLifePercent; // Помещаем в массив
        filesOut[0].width(2);
        filesOut[0] << "Проценты: " << percentBattery[i] << " | Время: " << aTm2->tm_hour
                    << ":" << aTm2->tm_min << ":" << aTm2->tm_sec << endl;
                        cerr << etalon << endl;
        if((etalon >= percentBattery[i] + 10) && (percentBattery[i] != percentBattery[i-1])) // С шагом в 10% берем замеры и сколько прошло времени с момента начала тес
        {
            start_time[j] =  clock();
            percentBattery10[j] = percentBattery[i];
            j++;
            etalon = percentBattery[i];
        }
        if(percentBattery[i] < 10 || i == measurement - 1 ) // Если батарея почти разряжена или кол-во замеров подошло к концу производим последние расчеты перед переда
        {
            unsigned int end_time = clock();
            for (int k = 0; k < j; k++){
                search_time[k] = end_time - start_time[k];

                filesOut[1] << "Время прошло до разрядки с "<< percentBattery10[k] << " до "
                            << percentBattery[i] << "(м): " << (float(search_time[k]) / CLOCKS_PER_SEC) / 60 << endl;

                ofstream fileOutTime("out1.txt", ios::app); // Выводим данные, которые будет использоать функция разладок
                fileOutTime << endl << (search_time[k] / CLOCKS_PER_SEC) / 60;
            }
                filesOut[0] << "Проценты: " << percentBattery[i] << " | Время: " << aTm2->tm_hour
                            << aTm2->tm_min << aTm2->tm_sec << endl;
                maxi = i;
                i = measurement;
        }
    Sleep(timing);
```

Рис. 4.1: Фрагмент кода считывания основных данных (Проценты заряда батареи и соответствующее им время).

```
Приложение запущено: 2015/11/17 22:13:53
Количество замеров: 999
Период замеров(с): 60
Проценты: 96 | Время: 22:13:53
Проценты: 96 | Время: 22:14:53
Проценты: 95 | Время: 22:15:53
Проценты: 94 | Время: 22:16:53
Проценты: 93 | Время: 22:17:53
Проценты: 93 | Время: 22:18:53
Проценты: 92 | Время: 22:19:53
Проценты: 91 | Время: 22:20:53
Проценты: 90 | Время: 22:21:53
Проценты: 89 | Время: 22:22:53
Проценты: 88 | Время: 22:23:53
Проценты: 87 | Время: 22:24:53
Проценты: 86 | Время: 22:25:53
Проценты: 85 | Время: 22:26:53
Проценты: 85 | Время: 22:27:53
Проценты: 84 | Время: 22:28:53
Проценты: 83 | Время: 22:29:53
Проценты: 82 | Время: 22:30:53
Проценты: 81 | Время: 22:31:53
Проценты: 80 | Время: 22:32:53
```

Рис. 4.2: Пример выходного файла 1 из фрагмента кода

```
Время прошло до разрядки с 80 до 19(м): 59
Время прошло до разрядки с 70 до 19(м): 41
Время прошло до разрядки с 60 до 19(м): 44
Время прошло до разрядки с 50 до 19(м): 36
Время прошло до разрядки с 40 до 19(м): 28
Время прошло до разрядки с 30 до 19(м): 16
Время прошло до разрядки с 20 до 19(м): 1
```

Рис. 4.3: Пример выходного файла 2 из фрагмента кода

```
for (int i = 0; i < 7; i++){ // Поиск среднего значения в ряде и разбиения на интервалы
    for(int j = 0; j < l; j++){
        if(timeMinValue[i] > timeCalc[j][i]){
            timeMinValue[i] = timeCalc[j][i];
        }
        if(timeMaxValue[i] < timeCalc[j][i]){
            timeMaxValue[i] = timeCalc[j][i];
        }
        helpMass[i] += timeCalc[j][i];
    }
    timeAverageValue[i] = helpMass[i] / l;
    filesOut << timeMinValue[i] << " - " << timeMaxValue[i] << endl;
    filesOut <<"timeAverageValue: "<< timeAverageValue[i] << endl;
    filesOut << endl;
}
for(int i = 0; i < maxi - 1; i++){ // Построение таблицы вероятности интервалов времени разрядки
    variance = percentBatteryCalc[i] - percentBatteryCalc[i + 1];
    probabilityVariance[variance] = probabilityVariance[variance] + 1;

    if (maxVariance < variance){
        maxVariance = variance;
    }
}
    ofstream filesOut2(files[2]);
    for(int i = 0; i < maxVariance + 1; i++){
        filesOut2 << "Разница " << i << ": " << probabilityVariance[i] << endl;
    }
    for(int i = 0; i < maxVariance + 1; i++){
        double lh = float(probabilityVariance[i]) / maxi;
        filesOut2 << "Вероятность["<< i <<"]: "<< lh << endl;
    }
```

Рис. 4.4: Фрагмент кода отвечающий за построение таблицы вероятности разряда.

```
Разница 0: 74
Разница 1: 57
Разница 2: 1
Разница 3: 0
Разница 4: 1
Вероятность[0]: 0.552239
Вероятность[1]: 0.425373
Вероятность[2]: 0.00746269
Вероятность[3]: 0
Вероятность[4]: 0.00746269
```

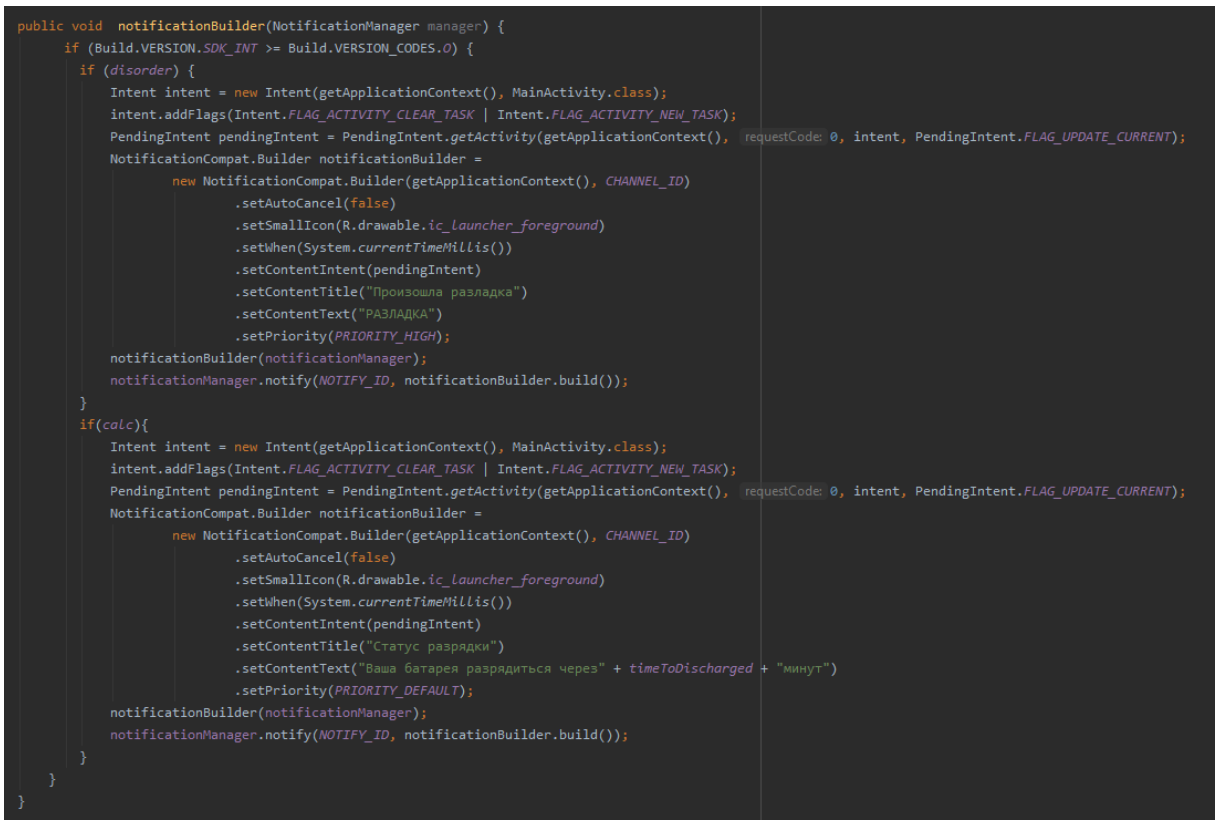
Рис. 4.5: Пример выходного файла 3 из фрагмента кода (Фраг.2)

```
if ((discord[j] > average + 10) || (discord[j] < average - 10)){
    cout << "Произошла разладка - выход за контрольные точки " << discord[j] << " - " << j+1 << endl;
}
if (discord[j] > average){series++;}
else if (discord[j] < average) {series--;}

if((series > 9) || (series < -9)){
    cout << "Произошла разладка - серия " << series << " - " << discord[j] << " - " << j+1 << endl;
}
if (discord[j] > discord[j-1]){monotonous++; p++; period1++;}
else if (discord[j] < discord[j-1]){monotonous--; p++; period2++;}

if((monotonous > 6) || (monotonous < -6)){
    cout << "Произошла разладка - монотонность " << monotonous << " - " << discord[j] << " - " << j+1<< endl;
}
if(monotonous == 0 && p > 6 && period1 == period2){
    cout << "Произошла разладка - переодичность " << discord[j] << " - " << j+1 << period1 << "| " << period2 << endl;
    period1 = 0; period2 = 0; p = 0;
}
if(j >= 2 && j <= 36){
for (int zz = j; zz < j + 3; zz++){
    if (discord[zz-2] > average + 7 || discord[zz-2] < average - 7 ){
        nearness++;
        cout << "discord_X[zz] " << discord[zz-2] <<" nearness " << nearness  << zz << endl;
    }
    }
    if(nearness > 1 ){cout << "Произошла разладка - приближение к контрольным границам " << discord[j] << " - " << j+1<< endl;}
    }
nearness = 0;

if ((discord[j] <= average + 3 && discord[j] >= average) || (discord[j] >= average - 3 && discord[j] <= average)) { center++;} else {center--;}
    if(center > 11)cout << "Произошла разладка приближение к центр линии " << discord[j] << " - " << j+1<< endl;
```

Рис. 4.6: Фрагмент кода отвечающий за отслеживание отклонения от нормы.

```
99 88 77 66 55 44 33 91 81 71 61 51 41 37 95 83 71 69 54 42 37 59 44 36 28 16 1
average_X: 70

Произошла разладка - выход за контрольные точки 99 - 1
Произошла разладка - выход за контрольные точки 88 - 2
Произошла разладка - приближение к контрольным границам 77 - 3
Произошла разладка - выход за контрольные точки 55 - 5
Произошла разладка - выход за контрольные точки 44 - 6
Произошла разладка - приближение к контрольным границам 44 - 6
Произошла разладка - выход за контрольные точки 33 - 7
Произошла разладка - монотонность 33 - 7
Произошла разладка - приближение к контрольным границам 33 - 7
Произошла разладка - выход за контрольные точки 91 - 8
Произошла разладка - приближение к контрольным границам 91 - 8
Произошла разладка - выход за контрольные точки 81 - 9
```

Рис. 4.7: Пример выходного файла 4 из фрагмента кода



```
public void  notificationBuilder(NotificationManager manager) {
      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        if (disorder) {
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),  requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
            NotificationCompat.Builder notificationBuilder =
                    new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID)
                            .setAutoCancel(false)
                            .setSmallIcon(R.drawable.ic_launcher_foreground)
                            .setWhen(System.currentTimeMillis())
                            .setContentIntent(pendingIntent)
                            .setContentTitle("Произошла разладка")
                            .setContentText("РАЗЛАДКА")
                            .setPriority(PRIORITY_HIGH);
            notificationBuilder(notificationManager);
            notificationManager.notify(NOTIFY_ID, notificationBuilder.build());
        }
        if(calc){
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),  requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
            NotificationCompat.Builder notificationBuilder =
                    new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID)
                            .setAutoCancel(false)
                            .setSmallIcon(R.drawable.ic_launcher_foreground)
                            .setWhen(System.currentTimeMillis())
                            .setContentIntent(pendingIntent)
                            .setContentTitle("Статус разрядки")
                            .setContentText("Ваша батарея разрядиться через" + timeToDischarged + "минут")
                            .setPriority(PRIORITY_DEFAULT);
            notificationBuilder(notificationManager);
            notificationManager.notify(NOTIFY_ID, notificationBuilder.build());
        }
    }
}
```

Рис. 4.8: Фрагмент кода отвечающий за отправку уведомлений на смартфоне в зависимости от ситуации.

Литература

1. vvsu.ru/ebook [Электронный ресурс]: Сайт цифровых учебно-методических материалов ВГУЭС – Электрон. дан. – [Владивостокск], cop. 2005- URL: abc.vvsu.ru/Books/statistika_up/page0002.asp

2. ГОСТ Р 50779.42-99. Статистические методы. Контрольные карты Шухарта. - М.: Издательство стандартов, 1999. - 36 с

3. Кравцов Ю.А. модели, алгоритмы и программы обнаружения нарушений при многомерном статистическом контроле процесса: канд тех наук: 2015 / В.Н. Клячкин; Федер. гос. бюджет. образоват. учреждение высш. проф. образования "Ульяновский гос ун-т Ульяновск 2015. - 143с. : карты шухарта : с. 15-23.

4. evileg.com [Электронный ресурс]: Сайт для программистов Qt - Электрон. дан. - [Москва], cop. EVILEG 2017 - URL: evileg.com/ru/knowledge/qt

5. Р. С. Некрасова, О. В. Лукашенко, И. В. Пешкова Моделирование случайных величин Учебно-методическое пособие для студентов математического факультета
   "ПетрГУ Петрозаводск 2013. - 13с. : Проверка на равномерность: с. 5.

6. Кобзарь А. И. Прикладная математическая статистика / А. И. Кобзарь. – М.: Физматлит, 2006. – 816 с Оценка параметров нормального распределения: с. 98.
   Критерии согласия для равномерного распределения: с. 319.

7. rain.ifmo.ru [Электронный ресурс]: санкт-петербургский государственный университет информационных технологий, механики и оптики факультет информационных технологий и программирования кафедра компьютерных технологий дискретная математика: алгоритмы Электрон. дан. - [Санкт-Петербург], cop. rain.ifmo 2018 - URL: rain.ifmo.ru/cat/view.php/theory/processes-automata/markov-2008

8. e-maxx.ru [Электронный ресурс]: Сайт с алгоритмами и языка C++ - Электрон. дан. - [Москва], cop. Public Domain 2018 - URL: e-maxx.ru/algo/binary_pow