Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Петрозаводский государственный университет (ПетрГУ)
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Промежуточный отчет о научно-исследовательской работе

Персонализированный анализ энергопотребления мобильных устройств

Выполнил:
студент 4 курса группы 22403 О. О. Савинов

_____
подпись

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

_____
подпись

Представлен на кафедру

« ____ » _____ 2017 г.

_____
подпись принявшего работу

Петрозаводск

2017

Содержание

Введение

За последние несколько лет произошел прорыв на рынке мобильных технологий. Очень часто появляются новые модели и версии мобильных устройств и, соответственно, приложений к ним. 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. Эффективность энергопотребления мобильных вычислительных систем, особенно при беспроводной передаче данных, участвующих в мобильных приложениях является одной из ва[illegible]ых задач. По сравнению с традиционными телефонными услугами, те возможности, что нам предоставляют современные устройства нельзя назвать «телефонными». 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, а, следовательно, возрастает актуальность проблемы эффективного мониторинга, такого важного ресурса, как заряд батареи. В связи с этим являются актуальными цели работы.

1) Совершенствование уже существующего прототипа системы мониторинга энергопотребления устройств.

2) Разработка мульти платформенного 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) Доработка системы отслеживание отклонений от нормы (оценок индивидуального шаблона потребления) и уведомление пользователя об этом.

4) Рарзаботка прототипа интерфейса, которое будет в виде всплывающих окон подсказывать, какова вероятность, оставщегося времени автономной работы устройства.

1 Изученные материалы

Для разработки прототипа приложения персонализированного анализа энергопотребления, требовалось изучить материалы указанные ниже.

Для того, чтобы считывать все необходимые данные для дальнейшего анализа, требовались специальные библиотеки C++ (пункт 1.2).

Чтобы анализировать считанные данные и строить по ним вероятностные модели, требуются некоторые знания в области теории вероятности (пункт 1.1).

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 (пункт 1.3).

Чтобы приложение выводило результаты собранных и обработанных данных и не загромаждало экран устройста пользвователя, был разработан прототип интерфейса всплывающих уведомлений посредством среды разработки Qt, в которой так же как и в C++, требовались специальные библиотеки (пункт 1.4).

Для построения нескольких оптимальных шаблонов для одного пользователя была осуществлена попытка применения проверки 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 не верные результаты. (пункт 1.5).

1.1 Статистический анализ экспериментальных данных

Вариационный ряд [1] – это отношение варианты(V) и частоты(P) (1).

$$M = \frac{V}{P} \tag{1}$$

Вариантой называют каждое уникальное значение в последовательности. А частотой называют количество повторений варианты. Статистическое определение вероятности применимы 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 Специальные библиотеки C++

Windows.h – это специальная библиотека с широким набором функций операционной системы. Является самым прямым способом взаимодействия приложений с 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).

1.3 Метод обнаружения нарушения процесса и выявление разладок

Контрольная карта (карта Шухарта) [3] это линейчатый график, построенный на основании данных измерений показателей процесса в различные периоды времени. Он позволяет отразить динамику изменений показателя и за счет этого контролировать процесс. Карты Шухарта включает с себя перечень установленных правил:

Рис. 1: Пример карты Шухарта

1) Выход точек за контрольные границы.

2) Серия – набор последовательных точек, находящихся по одну сторону от среднего уровня. Сигнализирующей 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.

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

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

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.

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

1.4 Классы QWidget и QDesktopWidget.

Классы QWidget и QDesktopWidget - требуются для создания объектов пользовательского интерфейса (Рис. 2). 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.

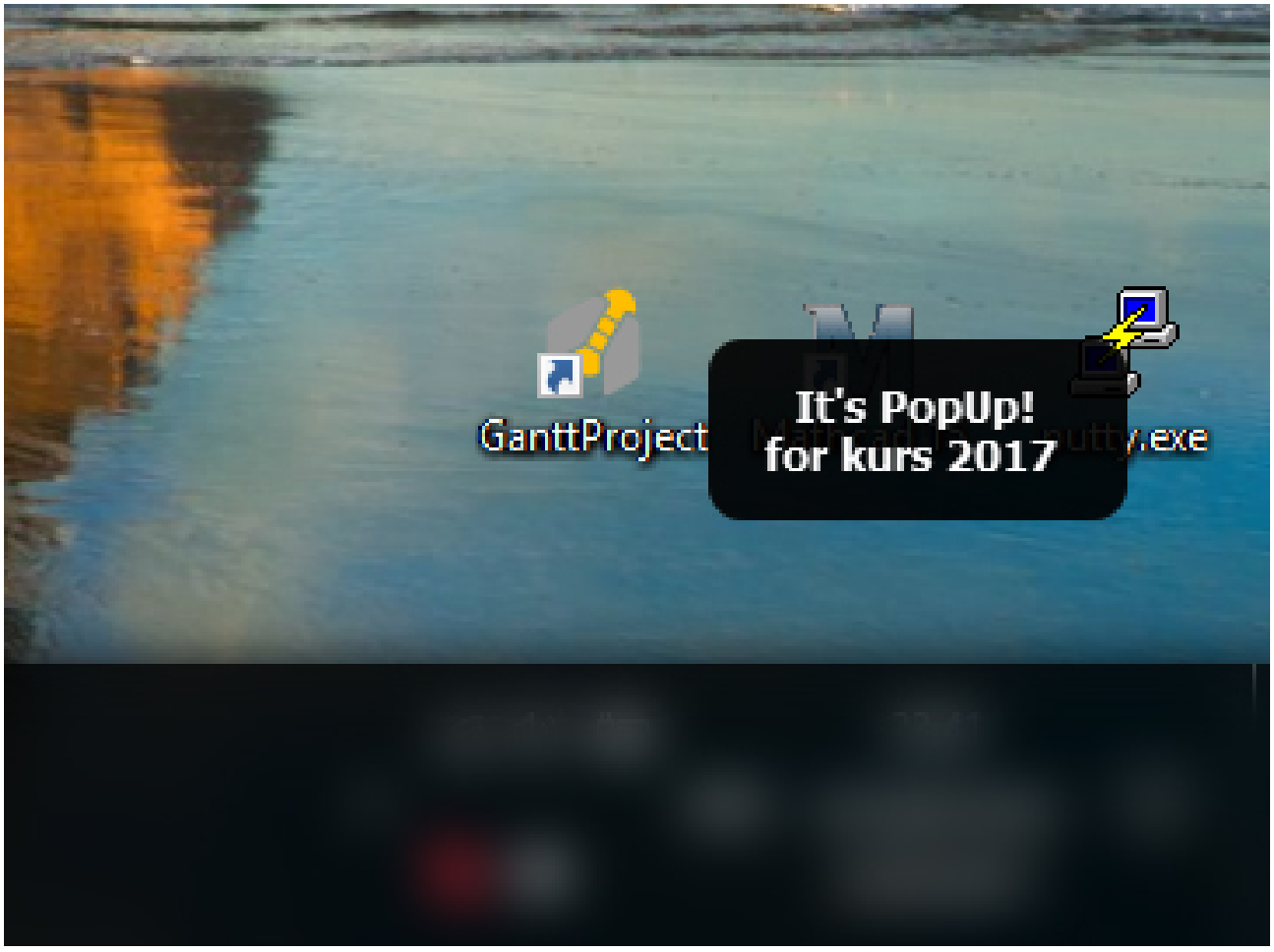

Рис. 2: Пример всплывающего уведомления.

1.5 Проверка на равномерность

Последовательность x_1,...,x_n проверяется на равномерность [5]. Множество значений полученной последовательности разбивается на k непересекающихся интервалов. Пусть n_i – количество элементов выборки, попавших в интервал i, а p_i – вероятность попадания в данный интервал. Статистика критерия определяется следующим образом:

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

Далее наблюдаемое значение статистики критерия сравнивается со всеми критериями которые уже были записаны ранее χ_m^2, где $m = \{1, ..., n-1\}$. Если $\chi_n^2 > \chi_m^2$, то создается новый оптимальный шаблон.

2 Постановка задачи

2.1 Дороботка алгоритма поиска разладок

1) Из-за сильного изменения энергопотребления, алгоритм 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:

1.1) Интенсивное использование (пользователь сильно нагружает систему и батарея быстро разряжается).

1.2) Умереное использование (пользователь в стабильном режиме использует устройство и равномерно и не очень быстро разряжается батарея).

2) Оптимизирован и усовершенствован код для поиска разладок.

Из-за внесенных правок в алгоритм описанные в пункте 1 были переделаны алгоритмы отвечающие за поиск разладок. Теперь они дают более точную оценку произошедшей разладки.

2.2 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 устройста было не равномерным, (система то нагружена, то нет) то алогоритм выдавал не верный результат, из-за чего пришлось отказаться от этой системы.

2.3 Интерфейс

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

2.4 Тестирование

Проводилось тестирование на личном ноутбуке, некоторые результаты тестов можно увидеть в приложении.

Архитектура разработанного прототипа приложения:

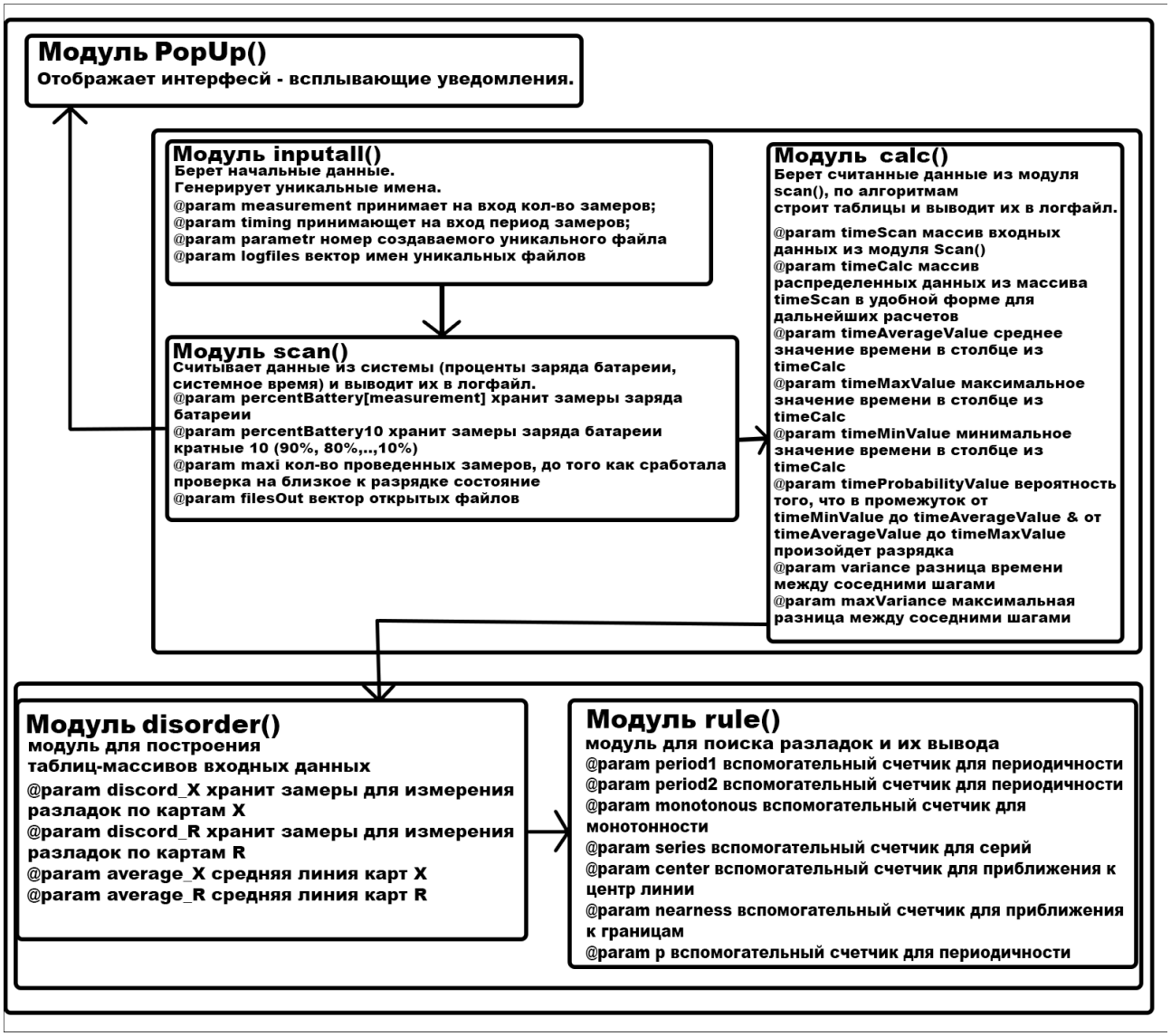

Рис. 3: Архитектура прототипа приложения персонализированого анализа энергопотребления мобильных устройств.

3 План дальнейшей работы

1. Расширять алгоритмическую базу для разладок. Существуют более современные, функциональные и точные алгоритмы нахождения нарушений, которые нужно реализовать для более точной работы приложения.

2. Создание кроссплатформенного приложения Разработаное приложение будет намного актуальнее для мобильных платформ, таких как: Android, IOS, WindowsPhone. Планируется внедрение утилиты в эти операционные системы.

3. Разработка всплывающих уведомлений, информирующих о том, что энергопотребление в момент использования, отличается от привычного.

4 Приложение

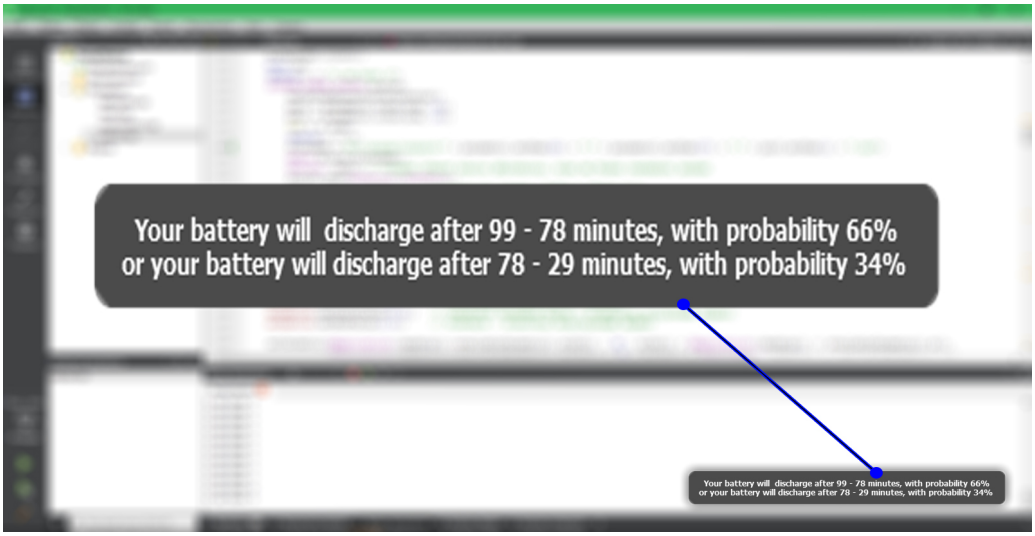

Рис. 4: Пример 1: Всплывающее уведомление, информирующее о том, за сколько минут разрядится батарея с данной вероятностью.

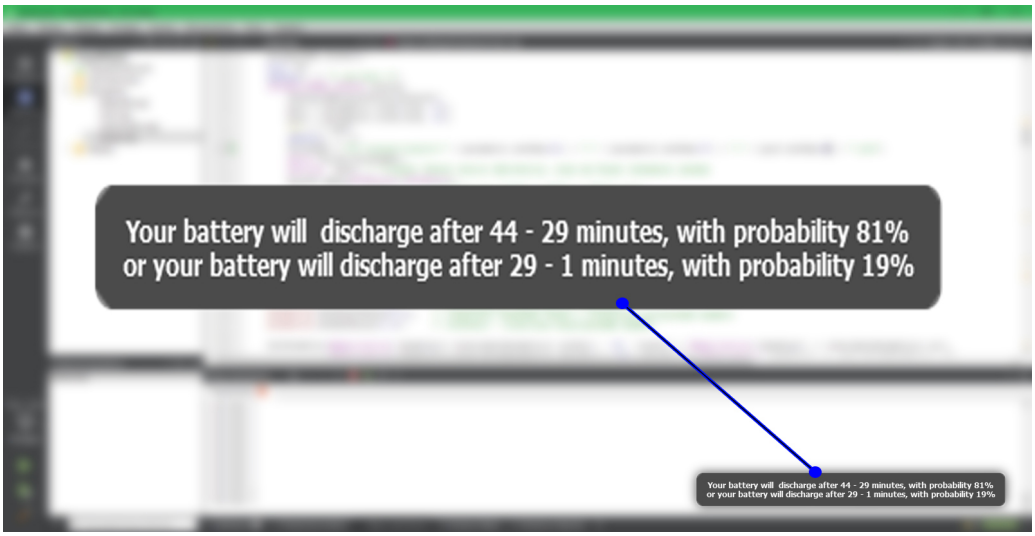

Рис. 5: Пример 2: Всплывающее уведомление, информирующее о том, за сколько минут разрядится батарея с данной вероятностью.

Пояснения к рисункам: уведомление показывается в нижнем правом углу экрана в момент, когда заряд батареи достигает определенного уровня, например, на Рис. 3 показано время и веротяность разрядки в момент 80%, а на Рис. 4 показано время и веротяность разрядки в момент 20%. Информация берется из заранее подготовленных лог-файлов, которые строятся по индивидуальной статистике, собираемой алгоритмом, разработанным в прошлом году.

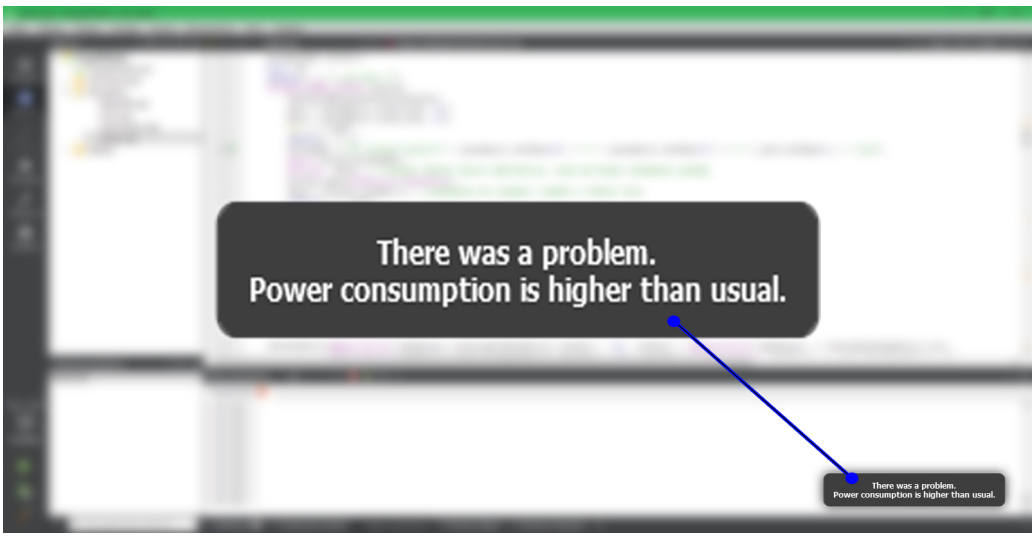

Рис. 6: Прототип всплывающего уведомления, информирующее о том, что энергопотребление выше, чем обычно.

```cpp
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; // Помещяем 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
        {
            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 пе
        {
            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. 7: Фрагмент кода считывания основных данных (Проценты заряда батареи и соответствуеющее им время). Фраг.1

```
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. 8: Пример выходного файла 1 из фрагмента кода (Фраг.1)

```
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. 9: Пример выходного файла 2 из фрагмента кода (Фраг.1)

```
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. 10: Фрагмент кода отвечающий за построение таблицы вероятности разряда. Фраг.2

```
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. 11: 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. 12: Фрагмент кода отвечающий за отслеживание отклонения от нормы. Фраг.3

```
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. 13: Пример выходного файла 4 из фрагмента кода (Фраг.3)

Список литературы

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.