Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

**Институт компьютерных наук и технологий**

**Кафедра Компьютерные интеллектуальные технологии**

**ОТЧЁТ**

**О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

**на тему: Проектирование высокоуровневой части IoT устройства для отслеживания изменений в сети Интернет**

направление подготовки: Прикладная информатика

Выполнил(а):

студент группы 33506/3

Заднепровский Андрей Владимирович

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Санкт-Петербург

2018

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc515388364)

[Актуальность сферы IoT 3](#_Toc515388365)

[Цели и задачи исследовательской работы 4](#_Toc515388366)

[1. Анализ актуальных IoT технологий 4](#_Toc515388367)

[1.1. Рассматриваемые устройства 4](#_Toc515388368)

[1.2. Критерии сравнения 5](#_Toc515388369)

[2. Проектирование IoT устройства 6](#_Toc515388370)

[2.1. Концепция модульности в IoT 6](#_Toc515388371)

[2.2. Задачи при проектировании хаба 6](#_Toc515388372)

[2.3. Проектирование низкоуровневого протокола 7](#_Toc515388373)

[3. Реализация IoT устройства 8](#_Toc515388374)

[3.1. Использование Arduino в качестве хаба 8](#_Toc515388375)

[3.2. Реализация протокола 8](#_Toc515388376)

[3.3. Реализация уведомлений о событиях 9](#_Toc515388377)

[ЗАКЛЮЧЕНИЕ 10](#_Toc515388378)

[СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 11](#_Toc515388379)

[ПРИЛОЖЕНИЯ 12](#_Toc515388380)

[Приложение 1. Реализация низкоуровневого протокола на с++ 12](#_Toc515388381)

[Приложение 2. Принципиальная схема 14](#_Toc515388382)

# ВВЕДЕНИЕ

## Актуальность сферы IoT

Интернет вещей (англ. Internet of Things — IoT) – это новый этап эволюционного развития Интернета, значительно расширяющий возможности сбора, анализа и распространения данных. Поскольку прогресс человеческого общества во многом зависит от превращения исходных данных в полезную информацию, и, в конечном итоге, в мудрость. В этом смысле Интернет вещей приобретает огромное значение.

Уже есть проекты, наглядно демонстрирующие его способность преодолеть разрыв между богатыми и бедными, предоставить мировые ресурсы тем, кто больше всего в них нуждается, и помочь нам лучше понять свою планету, чтобы научиться предупреждать проблемы [2]. Вместе с тем есть факторы, замедляющие развитие Интернета вещей. К ним относятся переход к протоколу IPv6, принятие единого набора общих стандартов и разработка источников питания для миллионов (и даже миллиардов) миниатюрных датчиков.

По оценкам компании Ericsson, в 2018 году число датчиков и устройств Internet of Things превысит количество мобильных телефонов и станет самой большой категорией подключенных устройств. Аналитики компании прогнозирует, что из приблизительно 28 млрд подключенных устройств по всему миру, к 2021 году, около 16 миллиардов будут связаны с IoT [1].

Сегодня Интернет вещей подходит к этапу, на котором разнородным сетям и множеству датчиков предстоит объединиться для взаимодействия под управлением единых стандартов. Эта цель требует от коммерческих организаций, государственных учреждений, стандартообразующих органов и учебных заведений общих усилий, направленных на достижение единой цели. Чтобы Интернет вещей обрел популярность у обычных пользователей, поставщики услуг и другие участники рынка должны разработать приложения, значительно повышающие качество жизни простых граждан.

## Цели и задачи исследовательской работы

В рамках моей исследовательской работы я намерен изучить области проектирования и разработки IoT устройств. В связи с чем я выделил следующие задачи:

* выявить наиболее оптимальные и применимые технологии и устройства, которые позволяют создавать функционально небольшие системы для решения узкого круга задач,
* спроектировать и реализовать IoT устройство на основе стека выявленных технологиях, которое сможет отслеживать изменения в сети Интернет и реагировать на них,
* проанализировать полученную информационную систему на предмет безопасности, отказоустойчивости и предложить дальнейшие варианты развития.

# 1. Анализ актуальных IoT технологий

Для того, чтобы понять, какая конфигурация IoT устройства будет приемлема для реализации поставленных целей, нужно ознакомиться с наиболее перспективными технологиями, определить критерии выбора и с помощью сравнительного анализа выбрать наиболее применимый по всем критериям вариант.

Основой любой информационной системы являются компьютеры и связи между ними. В моём случае IoT устройство также является полноценным компьютером небольшого размера со значительными ограничениями в производительности и вычислительной мощности, которых, тем не менее, достаточно для выполнения поставленных задач.

## 1.1. Рассматриваемые устройства

В рамках работы были рассмотрены следующие устройства:

1. Raspberry PI 3 B – новейшая модель небольшого портативного одноплатного компьютера с 1Гб ОЗУ, встроенным адаптером WiFi, 64-битным ARM-процессором Cortex-A53 с тактовой частотой 1,2 ГГц на однокристальном чипе Broadcom BCM2837, 40 контактами GPIO;
2. BeagleBone Black – бюджетный портативный комьютер с 512 Мб ОЗУ, [ARM-процессором Cortex-A8](https://www.ti.com/product/am3358) с тактовой частотой 1ГГц;
3. Radxa Rock2 Square – портативный компьютер с процессором ARM Cortex-A17 с тактовой частотой 1.8Ггц, ОЗУ можно выбрать: 1, 2 или 4 Гб;
4. NanoPC-T1 – портативный одноплатный компьютер на базе ARM-процессора Cortex-A9 Quad-core с тактовой частотой 1.5 Ггц. Имеет аналогичный Raspberry Pi характеристики, но также способен запускать ОС Android 5.0.

Существует множество других устройств, но эти четыре наиболее распространены и используются в сфере IoT.

## 1.2. Критерии сравнения

1. Цена. Стоимость устройства должна соответствовать целям и задачам, которые будут на него возложены;
2. Производительность. Устройство должно справляться с нагрузкой;
3. Доступность и документированность. Легче и надёжнее использовать проверенное людьми решение, которое используется в рабочих проектах, о котором собрана статистика использования;
4. Количество поддерживаемых типов интерфейсов ввода-вывода. Так как устройство должно функционировать в сети интернет, а также визуализировать входящие сообщения, оно должно обладать некоторым набором портов ввода-вывода.

Исходя из изложенных выше критериев, я остановился на портативном компьютере Raspberry Pi 3 B. Из прочих, оно выделяется невысокой ценой, а также поддержкой интерфейсов: GPIO (низкоуровневые порты ввода-вывода), HDMI, Ethernet, USB. Кроме того, для Raspberry Pi существует большой выбор операционных систем, оптимизированных специально под это устройство. Среди них как UNIX, так и Windows системы.

Наиболее оптимальным выбором операционной системы для меня стал дистрибутив Linux Arch для ARMv8 архитектуры. Среди кандидатов были Raspbian, CentOS Core, FreeBSD 11, Alpine Linux. К сожалению размеры Raspbian и CentOS были значительно больше остальных (3 Гб против 700Мб), FreeBSD 11 поддерживала не все порты ввода-вывода, а Alpine оказалась сложна для настройки и конфигурирования, так как требовала наличия монитора и подключения по HDMI. Дистрибутив Arch Linux славится своей универсальностью, надёжность и большим сообществом, поэтому идеально подходит в роли операционной системы для IoT устройства.

# 2. Проектирование IoT устройства

## 2.1. Концепция модульности в IoT

Концепция IoT состоит в организации множества независимых устройств в сеть, каждое которых выполняет свою конкретную функцию. Это позволяет легко подключать новые устройства к сети, удалять устройства из сети или заменять их на другие. Это похоже на организацию разработки ПО в виде отдельных модулей, каждый из которых отвечает за свою функцию и может быть использован в множестве программ. Для этого особое внимание должно быть уделено архитектуре устройств и интерфейсу их взаимодействия с другими устройствами.

В рамках данной работы проектируется устройство для отслеживания событий в Интернете, его функция заключается в отправке запросов на ресурсы в Интернете и при выполнении некоторого условия генерировать событие. Однако есть еще одна важная функция – это уведомлять об этих событиях, и в рамках модульного подхода должно быть второе устройство (хаб), единственной функцией которого должно быть уведомление о событиях, приходящих из разных источников.

## 2.2. Задачи при проектировании хаба

Исходя и вышеизложенного при разработке такого хаб-устройства нужно ответить на два вопроса:

* как устройство будет узнавать о событиях,
* как устройство будет уведомлять о событиях.

Для оповещения о событиях нашего хаба оптимальным вариантом будет разработать свой низкоуровневый протокол, который точно отвечает поставленной задаче.

В свою очередь хаб будет сигнализировать о пришедших событиях при помощи подключенных к нему светодиодов. Такой вариант хорош на начальном этапе, чтобы сосредоточится на разработке архитектуры и не вдаваться в детали взаимодействия с конкретными устройствами вывода. В дальнейшем можно будет подключить к нему динамик и монитор.

## 2.3. Проектирование низкоуровневого протокола

При разработке протокола следует придерживаться следующих рекомендаций:

* длина отдельных полей желательно делать кратной байту,
* общая длина сообщения должна быть кратна байту,
* должен быть заголовок и концевик протокольного кадра – некоторые константные двоичные слова, которые сигнализируют о начале и завершении полезных данных и отделяют одно сообщение от другого,
* желательно выделить поле для контрольной суммы, чтобы получатель мог удостоверится в целостности принятых данных.

Основываясь на этих правилах был разработан протокол, состоящий из следующих полей:

* Head – сигнализирует о начале полезных данных, 8 байт;
* Sender – код отправителя (источник) события, 8 байт;
* Type – идентификатор (тип) события, 4 байта;
* Value – значение (параметры) события, 12 байт;
* Check-sum – контрольная сумма кадра, 8 байт;
* Ending – концевик кадра, 8 байт.

Итого общий размер одного протокольного кадра равен 48 битам, то есть 6 байт.

# 3. Реализация IoT устройства

## 3.1. Использование Arduino в качестве хаба

В отличии от основного устройства – микрокомпьютера Rasberry Pi 3, который, в рамках решаемой задачи использует протоколы прикладного уровня, такие как IMAP или HTTP, от хаб-устройства этого не требуется. Поэтому для него рациональнее использовать более простую платформу – программируемый микроконтроллер вместо полноценного микрокомпьютера с ОС.

Основываясь на тех же критериях, обозначенных в пункте 1.2. и с учетом обозначенных изменений в требованиях к искомому устройству было решено использовать для реализации хаб-устройства микроконтроллер Arduino Uno, который обладает следующими преимуществами:

* низкая цена и большая популярность в своем сегменте рынка;
* огромное сообщество пользователей, много информации о успешно реализованных проектах на этой платформе;
* возможность программировать на языке высокого уровня с++, а не на уровне процессорных команд.

## 3.2. Реализация протокола

Прием сообщений по описанному в п. 2.3. протоколу был реализован в формате библиотеки для Arduino [3]. По факту библиотека для Arduino представляет собой обычный с++ класс, его код представлен в приложении 1. Он реализует следующие методы:

* available – возвращает true или false, есть или нет в буфере новое сообщение;
* get\_sender, get\_type, get\_value – возвращают соответствующую информацию о событии;
* next – вызывается когда текущее сообщение было обработано, позволяет приступить к анализу следующего протокольного кадра.

## 3.3. Реализация уведомлений о событиях

Как уже было сказано в п. 2.2. уведомление о событиях будет происходить миганием светодиодов. Так как в данном случае у нас есть только один источник событий (электронная почта) и один тип событий (новое сообщение) будет достаточно одного светодиода, он будет мигать, когда будет приходить новый валидный кадр через последовательный порт, соответствующий структуре протокола. Еще 6 светодиодов будут отвечать за отображение значения (Value), в нашем случае это будет число новых сообщений в электронном ящике, они будут непрерывно гореть и 1 светодиод будет соответствовать одному непрочитанному сообщению.

Принципиальная схема описанной конфигурации находится в приложении 2. Код программы, отвечающий за обработку полученных сообщений и соответствующие включение и выключение светодиодов достаточно прост – это бесконечный цикл на каждой итерации которого вызывается метод available описанный в п. 3.2. Если он возвращает true то происходит мигание первым светодиодом и обновление значений остальных соответствующие новому значению поля Value.

# ЗАКЛЮЧЕНИЕ

В данной работе мной совместно с Киселевым Валентином была исследована область проектирования и разработки IoT устройства и достигнуты следующие цели:

* был проанализирован рынок как микрокомпьютеров, поддерживающих установку полноценной ОС так и небольших программируемых микроконтроллеров;
* сделан выбор подходящих аппаратных платформ для реализации поставленных задач, а именно для отслеживания событий в сети Интернет и для уведомления об этих событиях;
* изучены технологии разработки программ, поддерживающиеся данными аппаратными платформами;
* разработано ПО решающее поставленные задачи для обеих аппаратных платформ с поддержкой масштабирования, что в дальнейшем позволит достаточно легко добавить новые источники и типы событий;

На основании проделанной работы можно сделать вывод, что …

# СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

1. Интернет вещей // WIKIPEDIA.ORG: Википедия: Свободная энциклопедия – URL: https://ru.wikipedia.org/wiki/Интернет\_вещей – (дата обращения 23.05.2018)
2. Дейв Эванс. Интернет вещей. Как изменится вся наша жизнь на очередном витке развития Всемирной сети. [Электронный документ] – URL: https://www.cisco.com/c/dam/global/ru\_ru/assets/executives/pdf/internet\_of\_things\_iot\_ibsg\_0411final.pdf – (дата обращения 23.05.2018)
3. Writing a Library for Arduino [Электронный документ] – URL: https://www.arduino.cc/en/Hacking/LibraryTutorial - (дата обращения 23.05.2018)

# ПРИЛОЖЕНИЯ

## Приложение 1. Реализация низкоуровневого протокола на с++

// header file

#include <Arduino.h>

class SerialCommunicator

{

public:

SerialCommunicator();

bool available();

int get\_sender();

int get\_type();

int get\_value();

void next();

void serialEvent();

private:

bool \_available;

byte \_message[4];

bool check\_sum();

};

// cpp file

SerialCommunicator::SerialCommunicator() {

\_available = false;

Serial.begin(9600);

}

bool SerialCommunicator::available() {

return \_available;

}

void SerialCommunicator::next() {

\_available = false;

}

int SerialCommunicator::get\_sender() {

if (\_available) return (int)\_message[0];

else return -1;

}

int SerialCommunicator::get\_type() {

if (\_available) {

int tmp = \_message[1];

return tmp >> 4;

}

else return -1;

}

int SerialCommunicator::get\_value() {

if (\_available) {

byte tmp = \_message[1];

bitClear(tmp, 0);

bitClear(tmp, 1);

bitClear(tmp, 2);

bitClear(tmp, 3);

return (int)\_message[2] + (tmp << 4);

}

else return -1;

}

void SerialCommunicator::serialEvent() {

while (!\_available && Serial.available()) {

// get new byte

byte tmp = Serial.read();

// if first byte = 10101010 and available >= 5 bites

while (tmp == 170 && Serial.available() >= 5) {

// get all new message

for (byte i = 0; i < 4; i++) {

\_message[i] = Serial.read();

}

// analize massage

if (Serial.read() == 255 && check\_sum()) {

\_available = true;

}

}

}

}

bool SerialCommunicator::check\_sum() {

// TODO: реализовать подсчет контрольной суммы

return true;

}

## Приложение 2. Принципиальная схема