Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Кафедра «Телекоммуникационные системы»
Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»

ДОПУЩЕН К ЗАЩИТЕ
Зав. кафедрой
к.т.н., профессор Байкенов А.С.
(ученая степень, звание, ФИО)

_____
(подпись)
«____»_____ 2015 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

на тему: «Разработка платформы для беспроводной сенсорной сети»

Магистрант:	Досатаев А.Р		МРЭн-13
	(Ф.И.О.)	(подпись)	(группа)
Руководитель:	ph.D, профессор		Чайко Е.В.
	(ученая степень, звание)	(подпись)	(Ф.И.О.)
Рецензент:	к.т.н., декан «АиТ» КазАТК		Бахтиярова Е.А.
	(ученая степень, звание)	(подпись)	(Ф.И.О.)
Консультант по ВТ:	к.т.н., ст. преподаватель		Ефремова Ю.И.
	(ученая степень, звание)	(подпись)	(Ф.И.О.)
Нормоконтроль:	ст. преподаватель		Демидова Г.Д.
	(ученая степень, звание)	(подпись)	(Ф.И.О.)

Алматы, 2015

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Факультет: «Радиотехники и связи»
Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»
Кафедра: «Телекоммуникационные системы»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Досатаеву А.Р.
(фамилия, имя, отчество)

Тема диссертации: «Разработка платформы для создания беспроводной сенсорной сети»
Утверждена: Приказом № 139 от «31» октября 2013 г.
Срок сдачи законченной диссертации: «24» июня 2014
Цель эксперимента: проверка работоспособности платформы для создания беспроводной сенсорной сети с визуализацией данных

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:
1. Анализ рынка микроконтроллеров для узлов беспроводной сенсорной сети
2. Анализ рынка трансиверов для узла-шлюза беспроводной сенсорной сети
3. Анализ веб-серверов для приема, сохранения и визуализации принятых данных
4. Разработка программного обеспечения для приема, сохранения и визуализации данных, полученных от беспроводной сенсорной сети

Перечень графического материала (с точным указанием обязательных чертежей)
1. Рисунок 3 – Архитектура узла беспроводной сенсорной сети
2. Рисунок 12 – Использование памяти различных Web-серверов
3. Рисунок 17 – Схема организации платформы
4. Рисунок 18 – Типовая схема включения датчика вибраций

Рекомендуемая основная литература:
1. W. Dargie и C. Poellabauer, Fundamentals of wireless sensor networks: theory and practice, Singapore, 2010

ГРАФИК
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Обзор литературы	Апрель 2014	
Детальное изучение существующих МК и трансиверов	Ноябрь 2014	
Анализ и выбор микроконтроллера	Январь 2015	
Анализ и выбор трансивера шлюза	Февраль 2015	
Анализ и выбор веб-сервера	Март 2015	
Разработка ПО для визуализации данных с сенсоров (парсинг данных)	Май 2015	
Разработка ПО для узла-шлюза, для взаимодействия с модулем ESP8266	Май 2015	

Дата выдачи задания: февраль 2014

Заведующий кафедрой: _____ (подпись) Байкенов А.С. (Ф.И.О.)

Руководитель: _____ (подпись) Чайко Е.В. (Ф.И.О.)

Магистрант: _____ (подпись) Досатаев А.Р. (Ф.И.О.)

Аңдатпа

Осы диссертациялық жұмыста сымсыз сенсорлық желісі үшін платформаны жобалау карастырылады. Ол құрылғылық және бағдарламалық жасақтаманың талдауы мен таңдауы және сенсордан алынған деректерді көзкөрімді өткізу үшін бағдарламалық жасақтаманың әзірлеуі енгізілді.

Аннотация

В работе рассматривается проектирование платформы для беспроводной сенсорной сети, который включает в себя анализ и выбор аппаратного и программного обеспечения, а также разработку программного обеспечения для визуализации данных, полученных с сенсоров.

Abstract

This paper considers design of a platform for wireless sensor network, which includes the analysis and selection of hardware and software, as well as the development of software for the visualization of data from sensors.

Содержание

Введение

В настоящее время повсеместно развивается концепция IoT (Internet of things), которая включает в себя такие технологии как:

– RFID;
– WSN;
– M2M.

Технология беспроводных сенсорных сетей подразумевает под собой самоорганизующуюся сеть с узлами с крайне низким энергопотреблением, которая способна передавать данные с сенсоров на оконечное устройство.

Основные элементы беспроводной сенсорной сети:

– Сенсор;
– Узел;
– Шлюз.

Шлюз в данном контексте понимается как преобразователь из одного формата в другой, т.е. он занимается переводом информации из сети ZigBee в сеть Wi-Fi.

Также в сети могут участвовать дополнительные элементы, такие как оконечные устройства, которыми можно управлять в зависимости от значений сенсоров.

В работе будет использоваться понятие платформа для беспроводной сенсорной сети, под которым необходимо понимать следующее – аппаратная и программная часть беспроводной сенсорной сети. Причем аппаратная часть включает в себя проектирование узлов и шлюзов, а конкретно, анализ рынка, выбор компонентов, разводка платы. Программная же часть включает в себя анализ рынка ОС для узлов, выбор конкретной ОС, разработка софта для обработки полученных данных с узлов.

Таким образом, в данной работе будет рассмотрена разработка платформы для беспроводной сенсорной сети, которая будет включать в себя следующие этапы:

– анализ рынка микроконтроллеров для узлов беспроводной сенсорной сети;
– анализ рынка трансиверов для шлюза беспроводной сенсорной сети;
– анализ рынка ОС для узлов беспроводной сенсорной сети;
– выбор микроконтроллера для узлов беспроводной сенсорной сети;
– выбор трансивера для шлюза беспроводной сенсорной сети;
– выбор ОС для узлов беспроводной сенсорной сети;
– анализ рынка веб-серверов;
– выбор веб-сервера для реализации сохранения данных с сенсоров и их визуализации;
– разработка ПО для сохранения данных с БД и дальнейшей их визуализации.

Таким образом, в работе будет приведено полное поэтапное описание всех шагов разработки платформы для беспроводной сенсорной сети с визуализацией данных сенсоров.

1 Беспроводные сенсорные сети

1.1 История беспроводных сенсорных сетей

Как и во многих других технологиях, военные были движущей силой развития беспроводных сенсорных сетей. Например, в 1978 году, агентство по перспективным оборонным научно-исследовательский проектам (DARPA) организовало семинар по сетям распределенных датчиков (DAR 1978). На семинаре были рассмотрены вопросы сенсорных сетей, такие как сетевые технологии, методы обработки сигнала и распределенные алгоритмы. DARPA также работала над программой распределенных сенсорных сетей (DSN) в начале 1980-х.

В сотрудничестве с Центром Rockwell, Калифорнийский университет в Лос-Анджелесе предложил концепцию беспроводных интегрированных сетевых датчиков или WINS (Pottie 2001). Одним из результатов проекта WINS был маломощный интегрированный микросенсор беспроводной связи (LWIM), выпускаемый в 1996 году. Эта интеллектуальная система зондирования была основана на чипе CMOS, интегрируя нескольких датчиков, интерфейсные схемы, цепи цифровой обработки сигналов, беспроводную связь, и микроконтроллер в одной микросхеме. Проект умная пыль в Университете Калифорнии в Беркли сосредоточены на проектировании чрезвычайно маленьких сенсорных узлов называемых пылинками. Целью этого проекта было показать, что сенсорная система может быть интегрирована в крошечное устройство, возможно с размером зерна песка или даже частицы пыли. Проект PicoRadio центра беспроводных научных исследований в Беркли (BWRC) специализируется на разработке маломощных сенсорных устройств, чья потребляемая мощность настолько мала, что они могут работать от источников энергии окружающей среды, таких как солнечная энергия или энергии вибрации. MIT μAMPS также фокусируется на малой мощности аппаратных средств и программные компоненты для сенсорных узлов, в том числе использование микроконтроллеров, способных реструктурировать алгоритмы обработки данных для уменьшения требования к питанию на уровне программного обеспечения.

В то время как предыдущие усилия в основном движимые академическими институтами, за последнее десятилетие появились ряд коммерческих компаний. Например: Crossbow, Sensoria, Worldsens, Dust Networks, и Ember Corporation. Эти компании предоставляют возможность приобрести сенсорные устройства готовые к развертыванию в различных сценариях применений наряду с различными инструментами управления для

программирования, технического обслуживания и визуализации данных датчика [1].

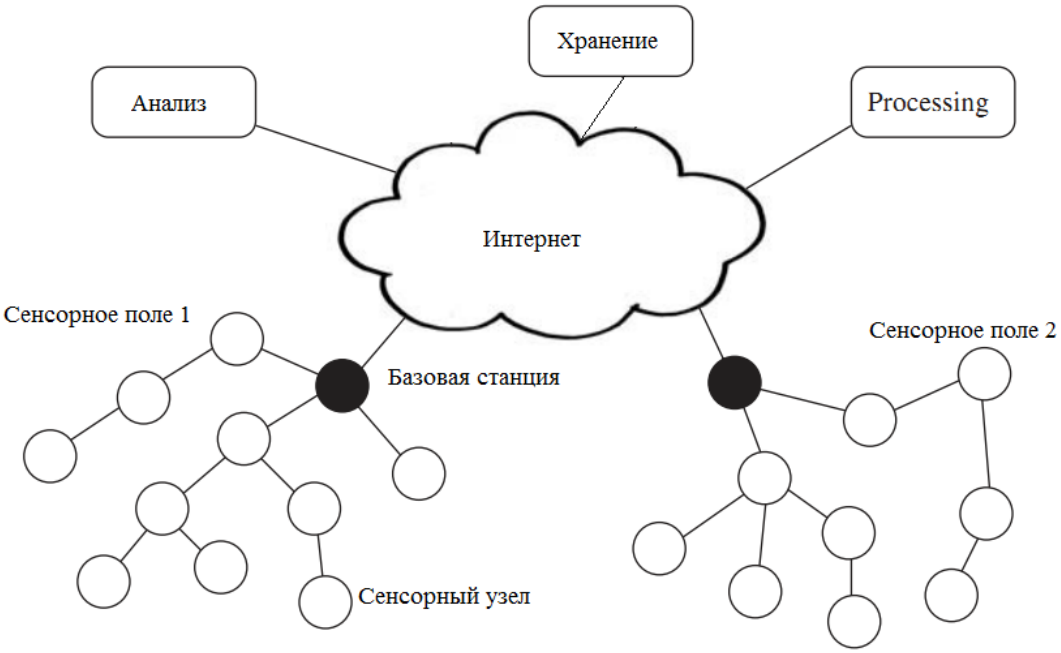


Рисунок 1 – Беспроводная сенсорная сеть

Беспроводная сенсорная сеть (WSN) – это сеть пространственно распределенных автономных узлов с датчиками для контроля различных физических или экологических условий, таких как температура, влажность, давление, и т.д., а также для совместной передачи своих данных к главной станции. Более современные сети являются двунаправленными, то есть также позволяют управлять деятельностью датчиками [2].

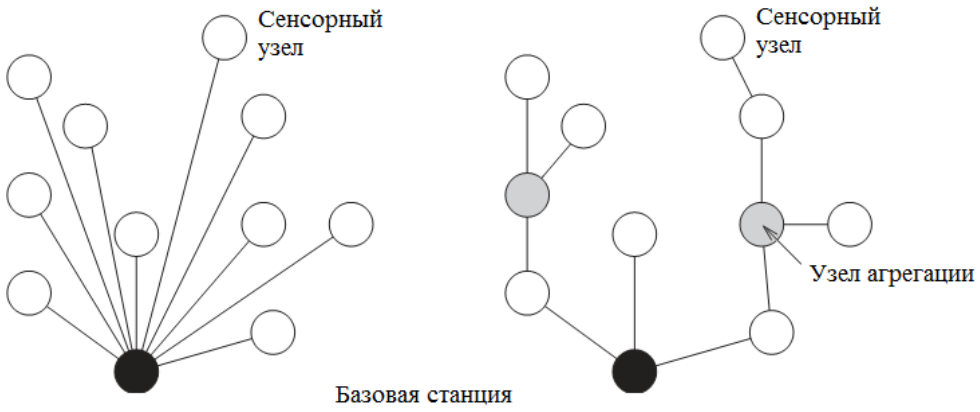


Рисунок 2 – Режимы передачи напрямую к базовой станции и через агрегационный узел

2 Выбор микроконтроллера и беспроводной части

2.1 Архитектура узла

Любой узел в беспроводной сенсорной сети должен включать в себя следующие подсистемы:

- процессорную подсистему;
- коммуникационную подсистему;
- сенсорную подсистему.

В данном случае каждая подсистема в равной степени важна.

Рисунок 3 – Архитектура узла беспроводной сенсорной сети

В настоящее время возможны следующие варианты построения узлов:

- в разных корпусах, то есть коммуникационная часть отдельно, процессорная отдельно;
- в одном корпусе, то есть и коммуникационная часть и процессорная в одном корпусе.

Примерами узлов в разных корпусах будут являться сочетания следующих микроконтроллеров: TI MSP 430, AVR ATMEGA, STM32 и прочих со следующих микросхемами приемо-передатчиков Zig-Bee: TI CC2420, TC 2520, Microchip MRF24j40 и прочими.

Примерами же узлов с процессорной и коммуникационной частью в одном корпусе являются: AVR ATMEGA128RFA1, STM32W108xx, TI CC2530 и прочие.

В следующей главе будет проведен анализ выбора конкретного железа.

2.2 Анализ “hardware” для узла беспроводной сенсорной сети

Как уже было сказано выше, возможно два варианта построения узла беспроводной сенсорной сети:

- в разных корпусах;
- в одном корпусе.

В своей работе я буду рассматривать устройства в одном корпусе. Причины на это в следующем:

− высокая оптимизация продукта;

− стоимость двух корпусов с доставкой выше, чем одного монолитного;

− нет необходимости использовать дополнительные интерфейсы для связи с радио модулем.

Для выбора был использован сайт фирмы DigiKey, поставщика электронных компонентов [3].

Таким образом, решено рассмотреть следующие виды микроконтроллеров:

− Freescale electronics MC1322;

− ST microelectronics STM32W108xx;

− Atmel ATMEGA128RFA1;

− TI CC2530.

Основными критериями для выбора было следующее:

– доступность микроконтроллера;

– невысокая цена, менее 15$ за штуку;

– флэш-память не менее 128 кб;

– наличие трансивера;

– высокая чувствительность.

Как видно из таблицы А1, наиболее широким диапазоном значений выходной мощности обладает STM32W108, также и наиболее высокая чувствительность у этого же решения. Самая минимальная мощность потребления ядра в активном режиме у CC2530 из-за отличия в архитектуре. По потреблению при приеме/передаче безусловным лидером является Atmel Atmega128RFA1. По сумме же характеристик наиболее совершенным микроконтроллером из приведенных является STM32W108, в том числе из-за низкой цены.

Таким образом, решено использовать микроконтроллер STM32W108 от фирмы ST Microelectronics.

2.3 Проектирование схемы и печатной платы узла беспроводной сенсорной сети

На рисунке Б1 [4] приведена принципиальная схема подключения микроконтроллера STM32W. С помощью программы EagleCAD [5] будет спроектирована схема и печатная плата для дальнейшего травления и сборки готового узла.

На рисунке приведена печатная плата, полученная после трассировки в программе EagleCAD.

2.4 Получение готовой платы узла беспроводной сенсорной сети

2.4.1 Лазерно-утюжная технология

Наиболее популярным способом изготовления печатной платы в домашних условиях является ЛУТ, который подразумевает под собой использование лазерного принтера, фотобумаги и утюга.

Для начала необходима предварительная обработка меди с помощью кусочка наждачной бумаги наименьшей зернистости. Обрабатывать желательно круговыми движениями, очищая от окислов.

На следующем этапе необходимо обработать медную фольгу растворителем 646 или ацетоном.

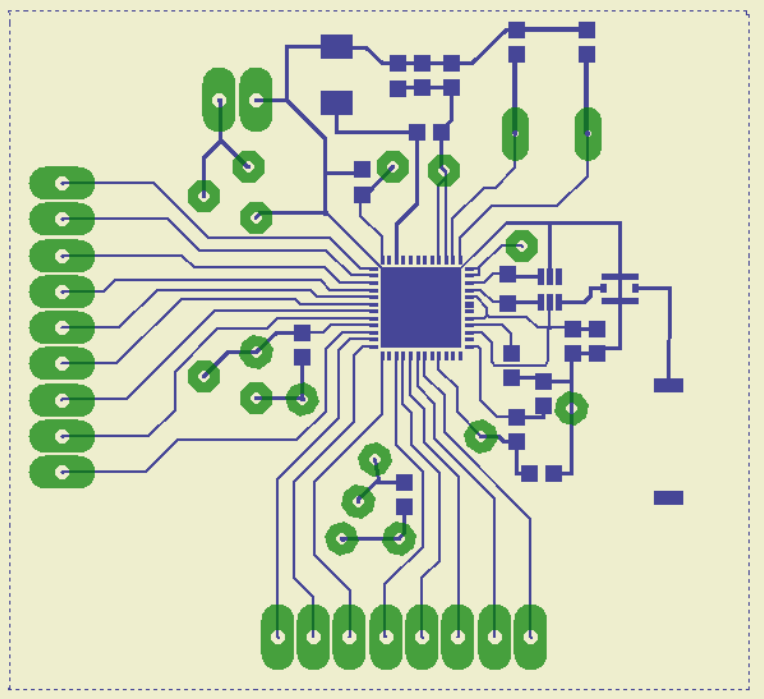


Рисунок 4 – Печатная плата узла на базе SoC STM32W

Теперь необходимо распечатать полученный рисунок печатной платы на лазерном принтере с максимальным качеством изображения. Готовое изображение предназначенное для принтера должно быть отображено зеркально и приведено на рисунке .

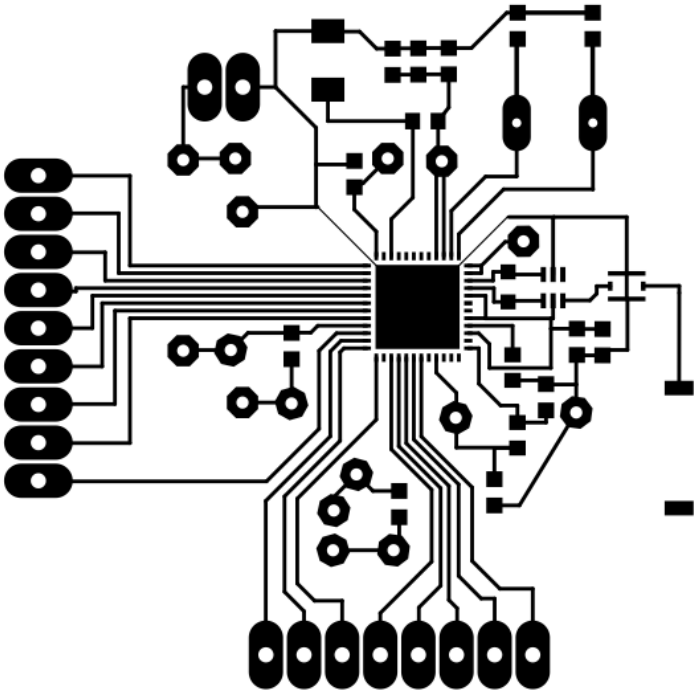


Рисунок 5 – Изображение для распечатки на фотобумаге, увеличенное в 3 раза

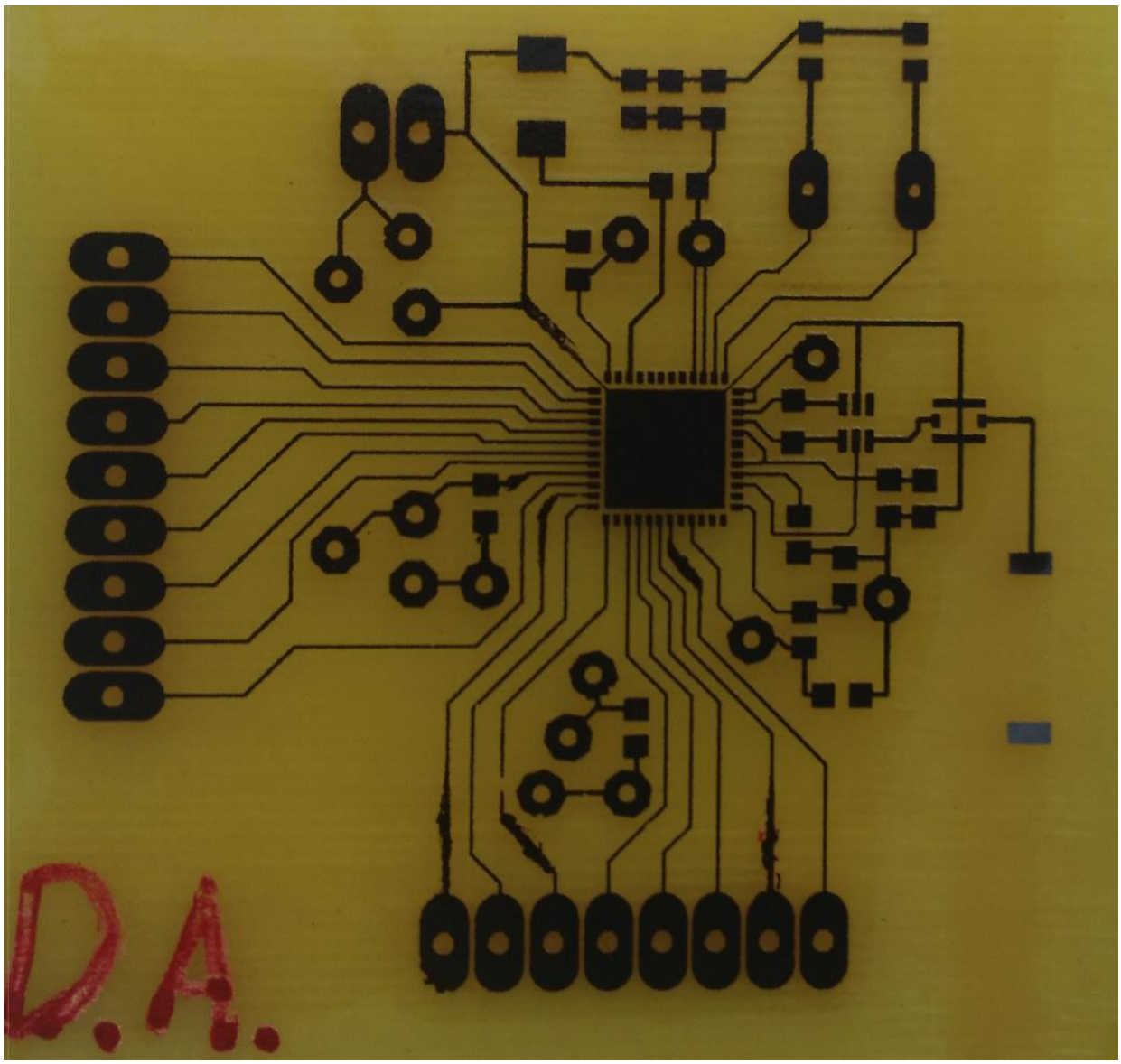


Рисунок 6 – Протравленная печатная плата после перевода тонера с фотобумаги на медную фольгу

На рисунке показана протравленная печатная плата с перенесенным тонером на медную фольгу, а на рисунке , протравленная и очищенная от тонера.

Для травления использовался следующий раствор [6]:

- 100 мл 3% перекиси водорода;
- 30 грамм лимонной кислоты;
- 5 грамм поваренной соли.

Скорость травления в данном растворе весьма высока, для платы размером 6х6 см, весь процесс занял 23 минуты.

Для очистки медной фольги от тонера после травления необходимо использовать растворитель 646 или ацетон с ватными дисками. Таким образом, на этом этапе получаем готовую для пайки печатную плату [7].

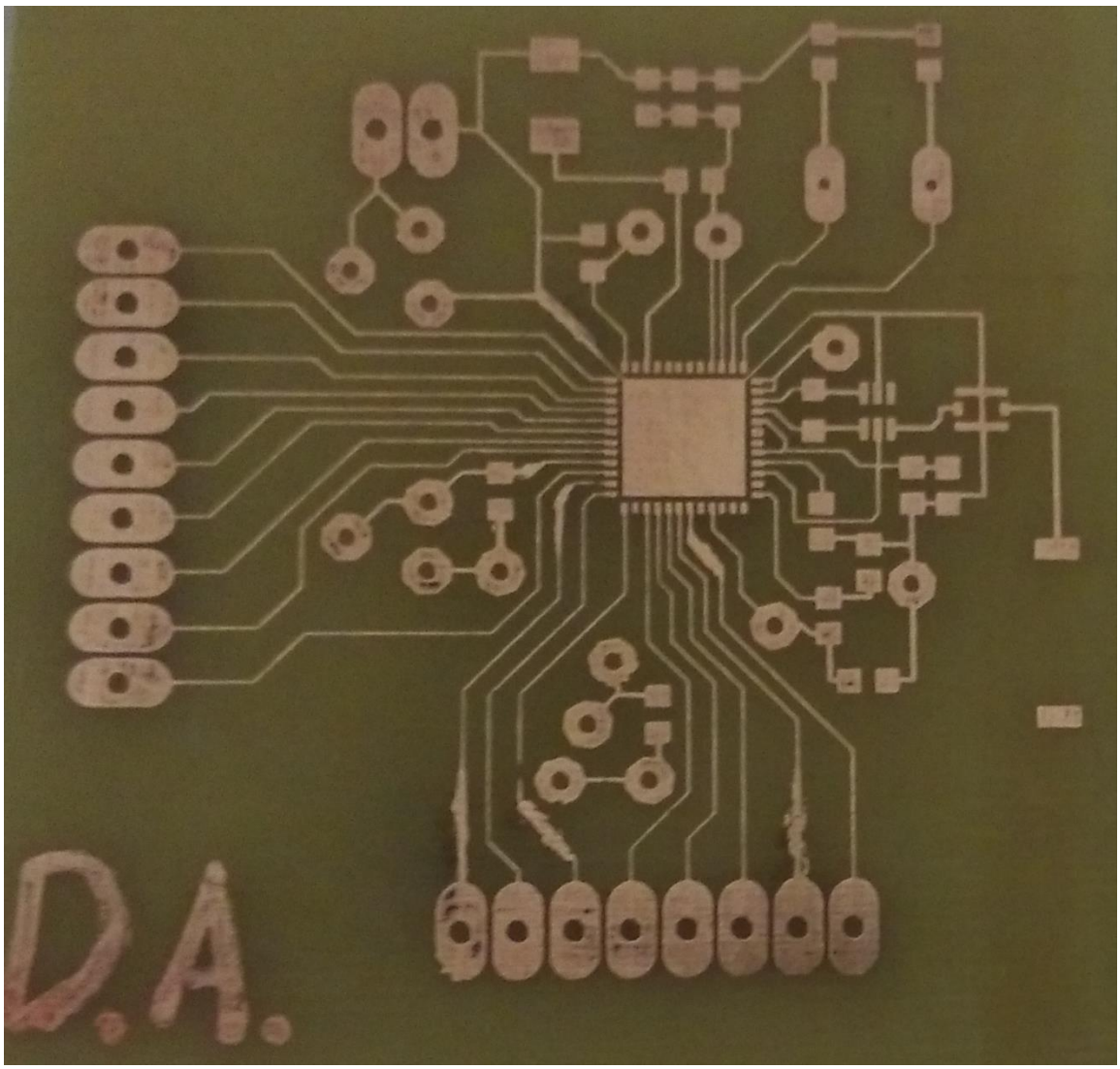


Рисунок 7 – Очищенная от тонера, печатная плата после травления

2.5 Коммуникационная часть для узла-шлюза

2.5.1 Анализ “hardware” для узла-шлюза беспроводной сенсорной сети

Для проведения анализа был использован сайт фирмы DigiKey [8], поставщика электронных компонентов. Были использованы следующие критерии в определении модулей:

- широкая доступность модуля;
- невысокая цена, менее 20$ за штуку;
- наличие интерфейсов UART, SPI.

Таким образом были выбраны три модуля:

- ESP8266;
- CC3200;
- CC3100.

Таблица 1 – Сравнительная таблица модулей Wi-Fi - UART

Наименование параметра	ESP8266 [9]	CC3200 [10]	CC3100 [11]
Максимальная скорость передачи данных, мбит/с	72,4	54	54
Потребление при передаче на скорости 1 Мбит/с, мА	215	272	272
Потребление при передаче на скорости 54 Мбит/с, мА	145	223	223
Потребление при приеме на скорости 1 Мбит/с, мА	60	53	53
Потребление при приеме на скорости 54 Мбит/с, мА	60	53	53
Чувствительность на приеме на скорости 1 Мбит/с (CCK), dBm	-98	-94,7	-94,7
Чувствительность на приеме на скорости 11 Мбит/с (CCK), dBm	-91,0	-87,0	-87,0
Чувствительность на приеме на скорости 6 Мбит/с (OFDM), dBm	-93,0	-89,0	-89,0
Чувствительность на приеме на скорости 54 Мбит/с (OFDM), dBm	-75,0	-73,0	-73,0
Мощность передачи на скорость 1 Мбит/с, dBm	19,5	17	17
Мощность передачи на скорость 11 Мбит/с, dBm	18,5	17,25	17,25
Мощность передачи на скорость 54 Мбит/с, dBm	16	13,5	13,5
Цена, долларов	2,5 [12]	14,65 [13]	-

Как видно из таблицы 1, лучшим выбором является модуль ESP8266, так как он по всем параметрам превосходит аналог от Texas Instruments, при этом стоимость в разы ниже. Конечно, для каких-то применений модуль от TI окажется лучше, но для нашего случая выбором является ESP8266.

2.5.2 Описание модуля ESP8266

ESP8266 предназначен для использования различных устройствах, например: беспроводных сенсорах, носимой электронике, умных розетках и так далее. Таким образом, можно сказать, что модуль ESP8266 будет являться важной составляющей «Интернета вещей».

Предусмотрено два варианта использования чипа:

1) в качестве моста UART-WIFI, когда модуль ESP8266 не использует встроенные интерфейсы, а лишь подключается к другому микроконтроллеру и

управляется AT-командами через UART, тем самым обеспечивая мост между уже готовым решением и инфраструктурой Wi-Fi;

2) в качестве самостоятельного мозга системы со своими портами ввода-вывода, то есть вся логика обрабатывается непосредственно модулем ESP8266.

Довольно интересная возможность использования модуля ESP8266 в качестве мозга всей системы. В этом случае в одном корпусе получается устройство со всевозможными интерфейсами, в том числе и Wi-Fi для взаимодействия с внешним миром. ESP8266 имеет все необходимые интерфейсы для работы с периферией, включая SPI, I2C и порты ввода-вывода [14].

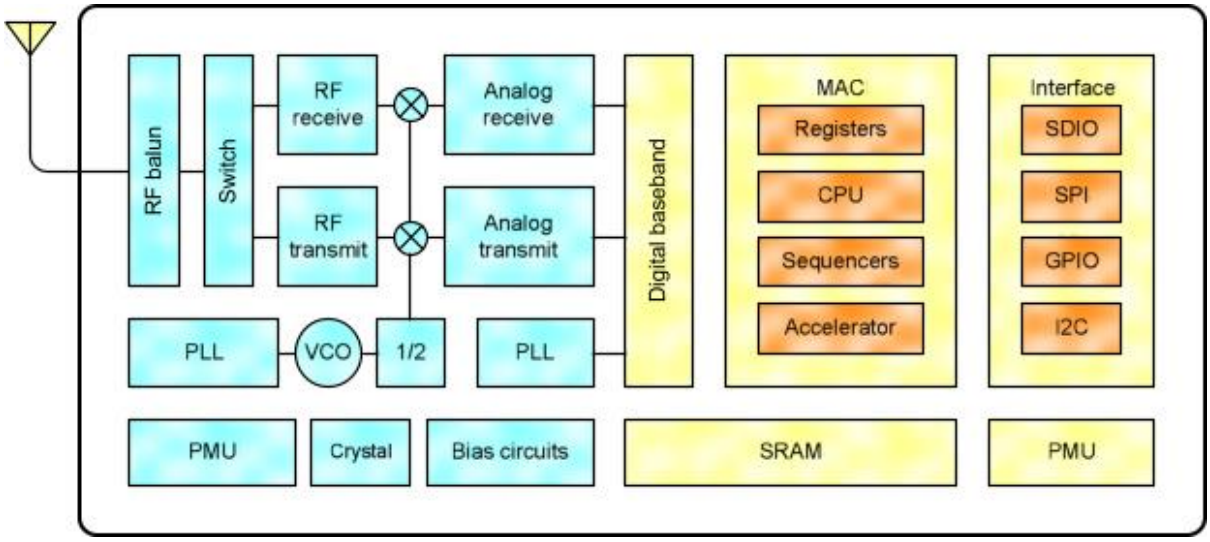


Рисунок 8 – Архитектура чипа ESP8266

3 Выбор ОС

3.1 Анализ рынка ОС для беспроводных сенсорных сетей

В настоящее время из программного обеспечения для беспроводных сенсорных сетей можно выделить следующие ОС:

- TinyOS;
- Contiki OS;
- RIOT.

Эти операционные системы развиваются, по сей день. Для того, чтобы сделать выбор, необходимо свести данные вместе. В таблице приведены характеристики ОС для беспроводных сенсорных сетей.

Таблица 2 – Сравнение характеристик различных ОС [15]

OS	Мин RAM	Мин ROM	Поддержка C	Поддержка C++	Многопоточность	Модульность	Реальное время
Contiki	< 2kB	< 30kB	Частично	Нет	Частично	Частично	Частично
Tiny OS	< 1kB	< 4KB	Нет	Нет	Частично	Нет	Нет
Linux	~ 1MB	~ 1MB	Полностью	Полностью	Полностью	Частично	Частично

OS	Мин RAM	Мин ROM	Поддержка C	Поддержка C++	Многопоточность	Модульность	Реальное время
RIOT	~ 1.5kB	~ 5kB	Полностью	Полностью	Полностью	Полностью	Полностью

Самая развитая ОС из приведенных в таблице – Linux. В то же время она требует порядка 1 Мб Rom, что является недопустимым в случае использования микроконтроллеров. Такой объем памяти практически не встречается среди доступных микроконтроллеров. Таким образом, выбираем из оставшихся ОС. Следующей ОС по функционалу является RIOT, но в ней отсутствует поддержка микроконтроллера STM32W, в то время как у Contiki OS таковая имеется. Поэтому, не смотря на более широкий функционал операционной системы RIOT, выбор сделан в пользу Contiki OS. Вариант с Tiny OS не рассматривается по причине крайне скудного базового функционала,

Таким образом минимально нам необходимо 30 Кб памяти постоянного запоминающего устройства и 2 Кб памяти оперативного запоминающего устройства

3.2 История создания и развития рассмотренных ОС

3.2.1 TinyOS

TinyOS это операционная система с открытым исходным кодом предназначен для маломощных беспроводных устройств, таких датчиком сетей, сетей распределенных вычислений, личных сетей, интеллектуальных зданий и умных счетчиков. TinyOS предоставляет полезные программные абстракции аппаратного обеспечения базового устройства: например, TinyOS может представить чип флэш-памяти, которая имеет блоки и секторы с определенными свойствами стирания/записи, как простой абстракции цикличного журнала. Предоставление полезных, хорошо продуманных и тщательно протестированных программных абстракций значительно упрощает работу приложений и систем разработчиков [16].

3.2.2 RIOT

История проекта RIOT:

− 2008 год. Корни проекта. Основой для RIOT был FeuerWare, операционная система для беспроводных сенсорных сетей. Это было частью проекта FeuerWhere где пожарники должны быть под контролем. Основные цели проекта были надежность и гарантии реального времени.

− 2010 год. Соответствие Интернету. Для повышения модульности и включают в себя новые IETF протоколы, μkleos была отпочкована от оригинального репозитория FeuerWare. Поддержка 6LoWPAN, RPL, и TCP была интегрирована в течение следующих лет.

− 2013 год. RIOT становится общедоступным. RIOT является прямым наследником μkleos. Произошел ребрендинг, чтобы избежать проблем с

правописанием и произнесением имени операционной системы. Это поспособствовало большему сообществу [17].

3.2.3 Contiki OS

Contiki появилась из желания Адама Дункеля для подключения различных вещей к Интернету. В 2001 году был разработан стек с открытым исходным кодом uIP, который быстро распространился по всему миру встроенной техники. Но uIP не просто используется в глубоко встраиваемых системах, он также дает доступ к Интернету различным вещам. В 2003 году Contiki последовали его примеру.

В 2004 году концепция protothreads, которая в настоящее время составляет основу процессов Contiki, был введен в Contiki. Ранние версии Cooja и Rime стека были добавлены с Contiki 2.0 в 2007 году. Профилирование мощности было разработано в 2007 году. Instant Contiki и файловая система Coffee были внедрены в начале 2008 года Позже в 2008 году, Cisco внес полностью сертифицированный IPv6 стек в Contiki.

В 2009 и 2010 годах, было добавлено множество новых платформ в Contiki и были разработаны новые механизмы с низким энергопотреблением. В 2011 году добавлено два важных механизма: ContikiRPL, для маршрутизации IPv6, и ContikiMAC для спящих маршрутизаторов.
В 2012 году была основана компания Thingsquare, чтобы перенести Contiki в облака [18].

3.3 Основные характеристики Contiki OS

Основные функции операционной системы Contiki OS:

− широкие сетевые возможности, включая стек протоколов TCP/IP, поддержку IPv6;
− многопоточность;
− динамическая подгрузка модулей;
− возможность перераспределения памяти;
− поддержка множества аппартных платформ;
− встроенный симулятор Cooja;
− встроенная система компиляторов для различных платформ [19].

Все перечисленные выше функции и инструменты значительно облегчают вхождение в разработку приложений для беспроводных сенсорных сетей.

В ОС Contiki OS имеются порты для следующих микроконтроллеров [20]:

− TI CC2538
− TI MSP430x
− Atmel AVR
− Freescale MC1322x
− ST STM32w
− TI MSP430

- Atmel Atmega128 RFA1
- Microchip pic32mx795f512l
- TI CC2530
- RC2300/RC2301

Как видно из вышеприведенного списка, все популярные на сегодняшний день микроконтроллеры для беспроводных сенсорных сетей поддерживаются Contiki OS, что говорит о современности этой ОС.

4 Сборка и написание ПО

4.1 Установка Contiki OS

Для установки Contiki OS необходима среда на основе Unix подобной ОС. Поэтому была выбрана операционная система Linux на основе дистрибутива Debian.

После установки и настройки самой ОС Linux необходимо ввести следующие команды для установки дополнительных пакетов:

sudo apt-get install build-essential binutils-msp430 gcc-arm-none-eabi gcc-msp430 msp430-libc binutils-avr gcc-avr gdb-avr avr-libc avrdude openjdk-7-jdk openjdk-7-jre ant libncurses5-dev doxygen git

Таким образом, в системе появятся компиляторы для различных платформ, система контроля версий git, окружение java для запуска эмулятора cooja из состава contiki-os, doxygen для чтения документации.

Теперь необходимо перейти в домашнюю папку и ввести следующую команду:

git clone git://github.com/contiki-os/contiki.git contiki

Тем самым будет скопирована последняя версия исходников contiki-os.

Таким образом, была проведена предварительная подготовка к работе с contiki-os.

Для тестирования возможностей Contiki OS, будем использовать симулятор Cooja из комплекта поставки. Cooja поддерживает несколько аппаратных платформ:

- Zolertia Z1 (TI MSP430+CC2420);
- Wismote (TI MSP430+CC2520);
- Micaz (Atmel AVR+CC2420);
- TMote Sky (TI MSP430+CC2420)
- Esb (TI MSP430+RFM TR1001);
- Native (Java module).

Соберем тестовую схему на базе узлов на платформе TMote Sky, приведенную на рисунке . В качестве прошивки будем использовать файлы из стандартных примеров:

- Udp-client.c;
- Udp-server.c.

Таким образом, мы организовываем передачу данных с датчиков на сервер. То есть скомпилированный код файла udp-client.c заливаем в модули

со второго по десятый, а код файла udp-server в узел под номером один. Исходный код файлов udp-client.c и udp-server.c приведен соответственно в приложениях Г и Д.

Для того, чтобы выполнить компиляцию, необходимо перейти в папку с исходными файлами следующей командой:

cd contiki/examples/udp-ipv6

Далее выбираем нашу платформу и выполняем компиляцию с помощью следующих команд:

make TARGET=sky udp-server

make TARGET=sky udp-client

Теперь мы можем использовать полученные скомпилированные файлы для симуляции в Cooja [21].

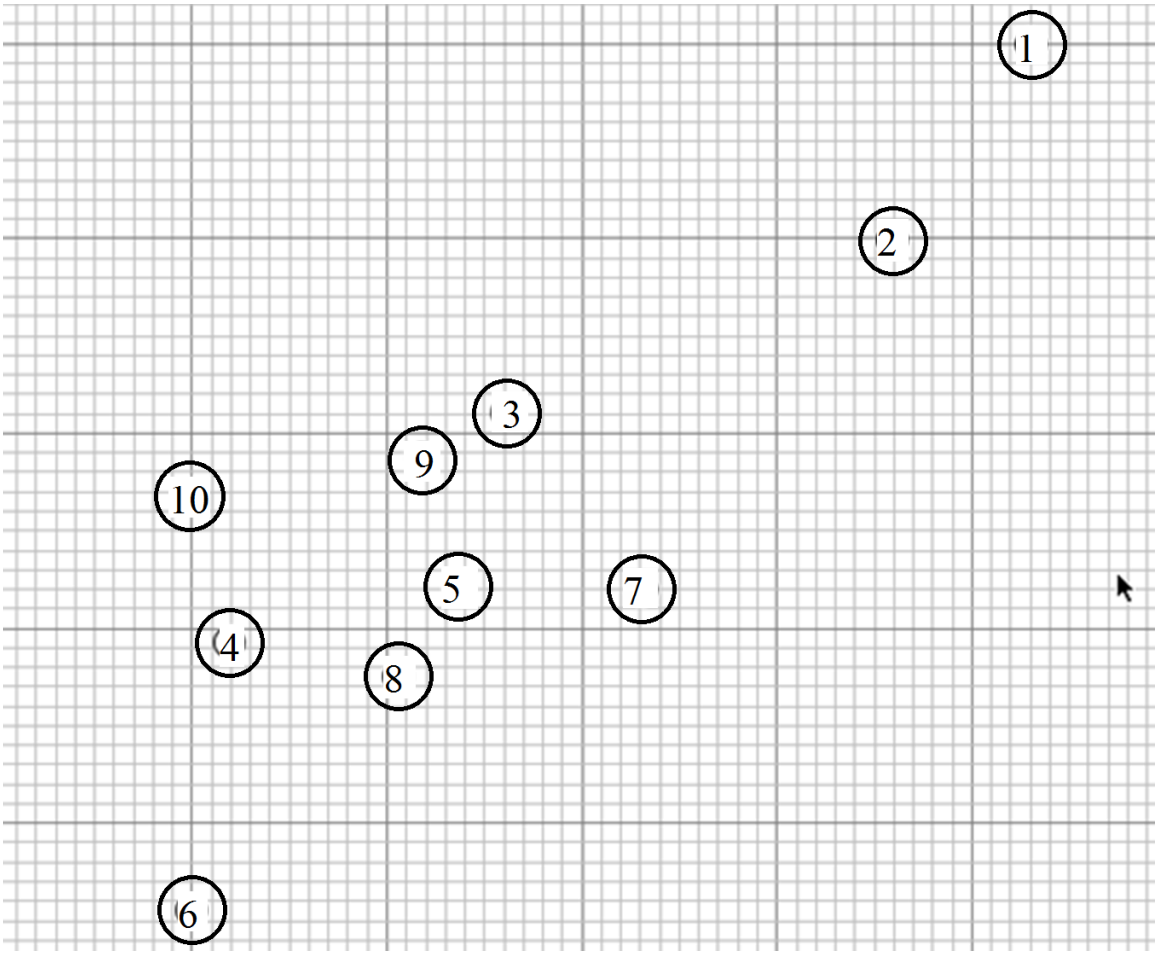


Рисунок 9 – Скриншот окна симулятора Cooja с тестовой схемой для проверки работы симулятора

На рисунке 10 приведен скриншот окна симулятора Cooja с запущенной симуляцией передачи пакетов от узлов 2-10 на узел-сервер.

На рисунке 11 приведена диаграмма вероятности приема в зависимости от расположения узлов. В данном случае напрямую с узла 4 напрямую на узел 1 нельзя передать данные, так как вероятность приема составляет 0%, в тоже время возможна передача через соседние узлы.

Таким образом, была проверена работа симулятора Cooja на реальном примере.

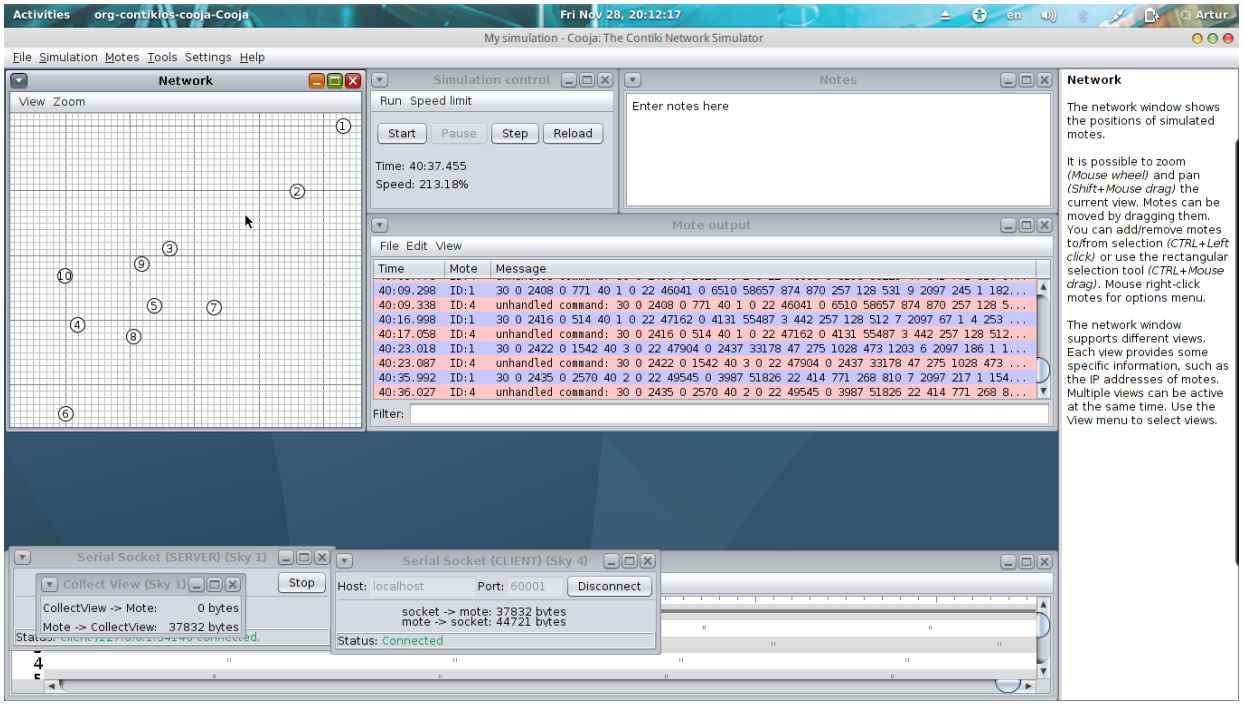


Рисунок 10 – Скриншот окна симулятора Cooja для платформы Tmote Sky

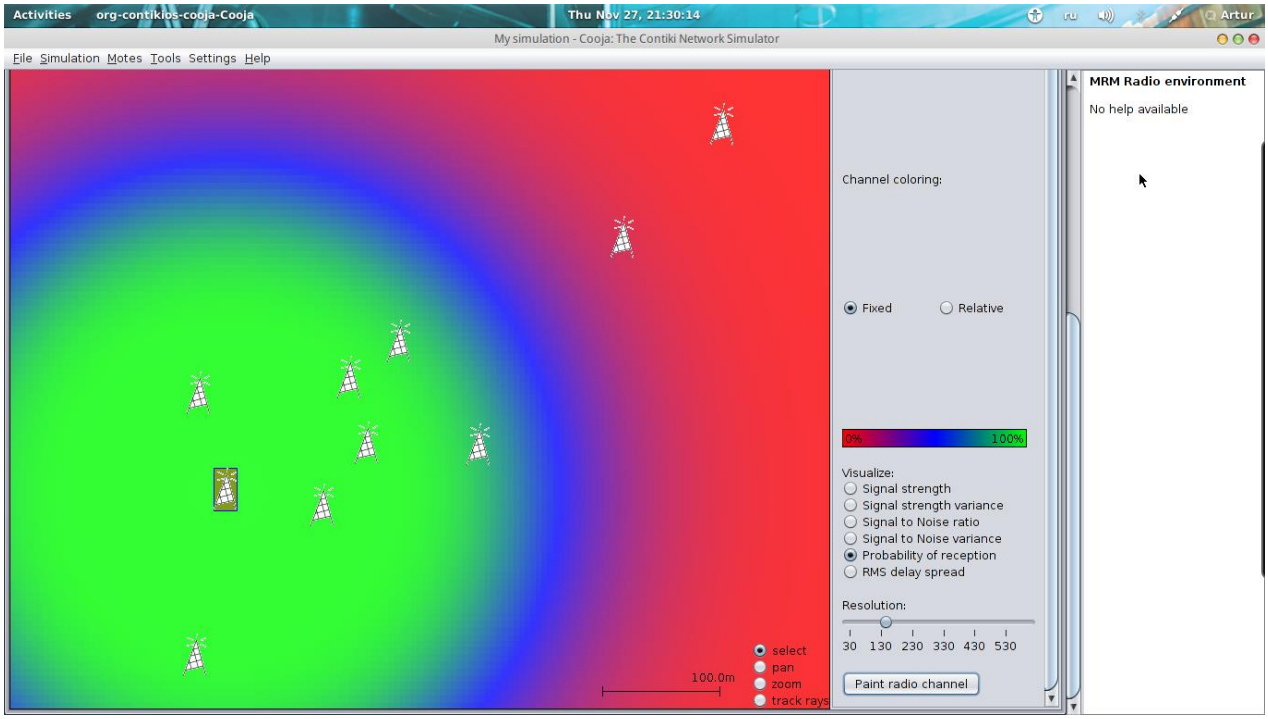


Рисунок 11 – Вероятность приема при отправке с 4 узла

5 Разработка ПО для визуализации данных

5.1 Выбор серверной платформы

В настоящее время наибольшую долю рынка занимают следующие Web-серверы [22]:

– Apache (39.26%);
– Microsoft IIS (28.88%);
– Nginx (14.42%).

Таким образом, наилучшим выбор для самостоятельного развертывания является один из этих трех продуктов. Так как платформа Microsoft IIS является не самой удобной для такого языка как PHP, то выбор будет сделан между Apache и Nginx. Основными характеристикой Web-сервера является [23]:

– потребление памяти при определенном количестве запросов;
– максимальное количество запросов в секунде при одновременном подключении.

На рисунке 12 приведен график использования памяти при различном количестве соединений для трех Web-серверов. Чем меньше значение тем лучше.

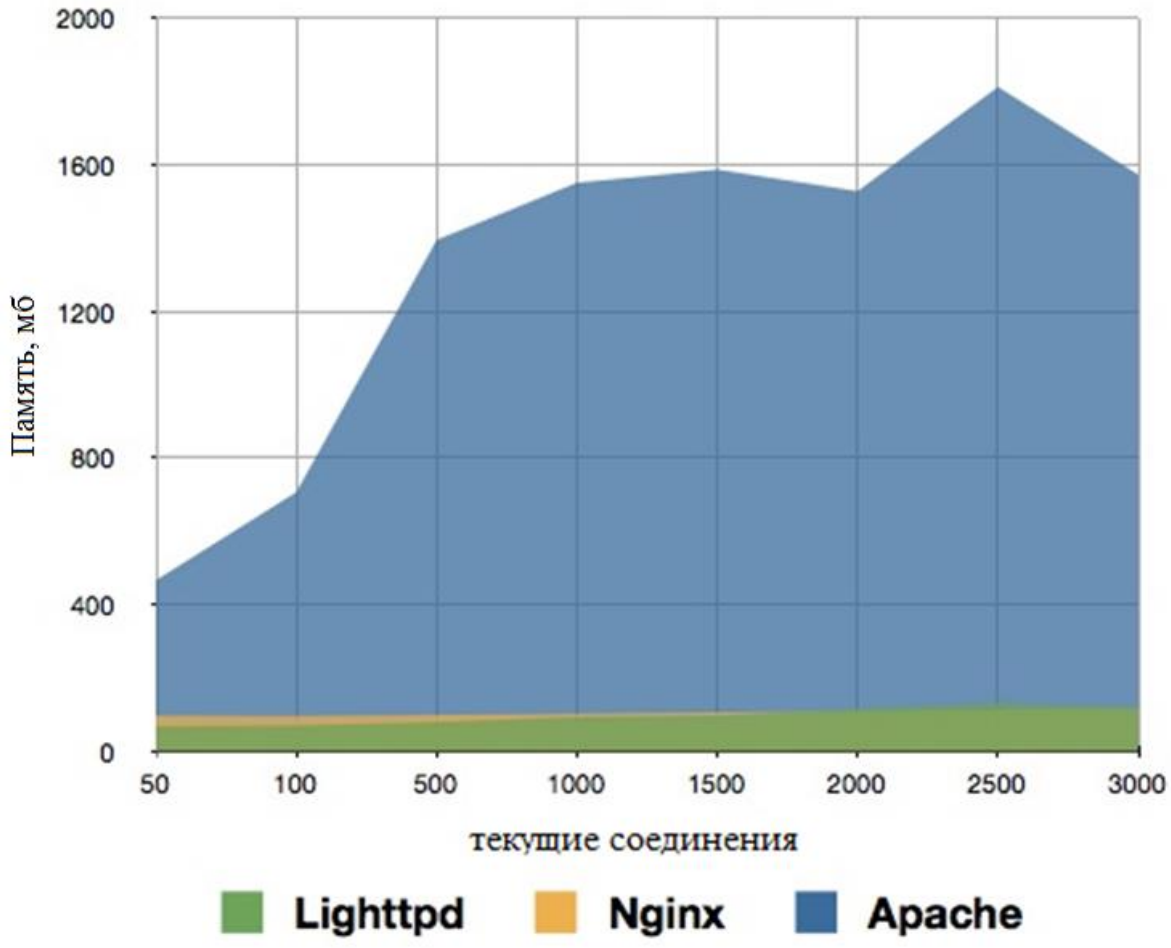


Рисунок 12 – Использование памяти различных Web-серверов

На рисунке Рисунок 13 можно видеть, что наибольшее количество одновременных запросов выдерживает Nginx, а наихудшим является Apache.

В обоих случаях наилучшие результаты были показаны Web-сервером Nginx, причиной этому является его новизна, а также желание разработчиков оптимизировать Web-сервер под большие нагрузки. Таким образом, наилучшие результаты были показаны именно Web-сервером Nginx, поэтому будем использовать именно его.

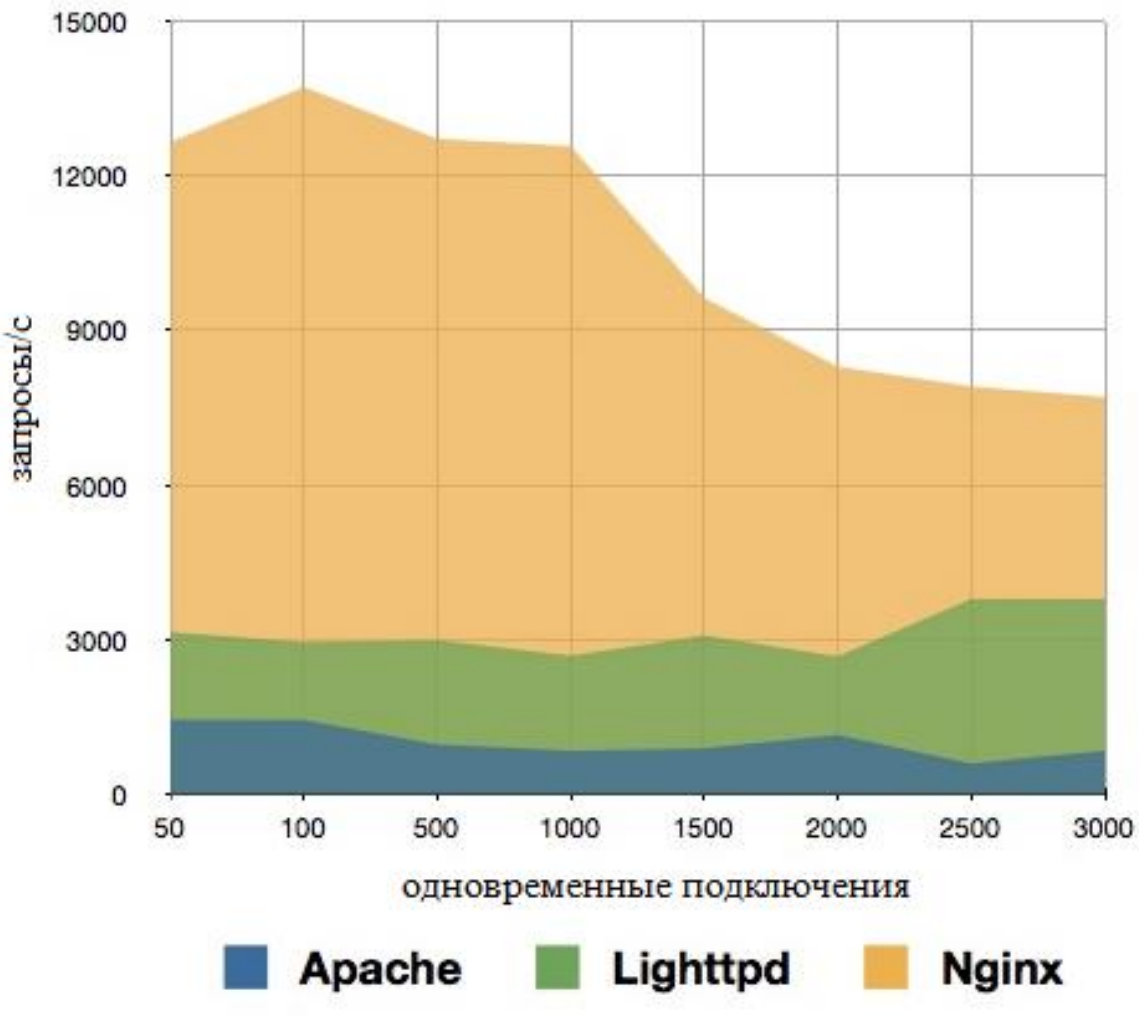


Рисунок 13 – Максимальное количество запросов, которые может отработать Web-сервер

5.2 Установка и настройка

В качестве Web-сервера для Windows был взят продукт Winginx [24], он содержит в себе следующие модули:

– Nginx;
– PHP;

– MySQL.

Таким образом, установив Winginx мы получаем готовый Web-сервер для развертывания своих Web-страниц, которые могут использовать серверные языки программирования и базы данных.

В работе не будет описана установка, а только лишь конфигурация Web-сервера для того, чтобы с ним можно было работать из локальной сети. Основным являются следующие строчки, полная же конфигурация приведена в приложении Е:

listen 80 default_server;

server_name _;

В данном случае, мы заставляем Web-сервер слушать 80 порт от любого сервера. Переменная default_server подразумевает вышесказанное. Имя сервера присваиваем несуществующему значению [25].

5.3 ПО для визуализации данных

Для визуализации данных была использована библиотека Highchart. В приложении И приведен код класса для работы с БД MySQL, написанный на языке PHP. В приложении К приведен код скрипта, написанного на языке PHP для записи в БД, данных полученных посредством POST запроса от модуля ESP8266. В приложении Л приведен код скрипта, написанного на PHP для выборки всех данных из таблицы в БД MySQL и дальнейшей визуализации средствами JavaScript с помощью библиотеки Highchart.

На рисунке 14 приведена подробная схема организации работы серверной части и взаимодействия с клиентами.

На первом этапе модуль ESP8266, который находится на узле-шлюзе посылает POST запрос, который минимально должен содержать следующую информацию:

- адрес отправки данных;
- версию HTTP проктола;
- тип передаваемых данных;
- размер передаваемых данных;
- передаваемые данные.

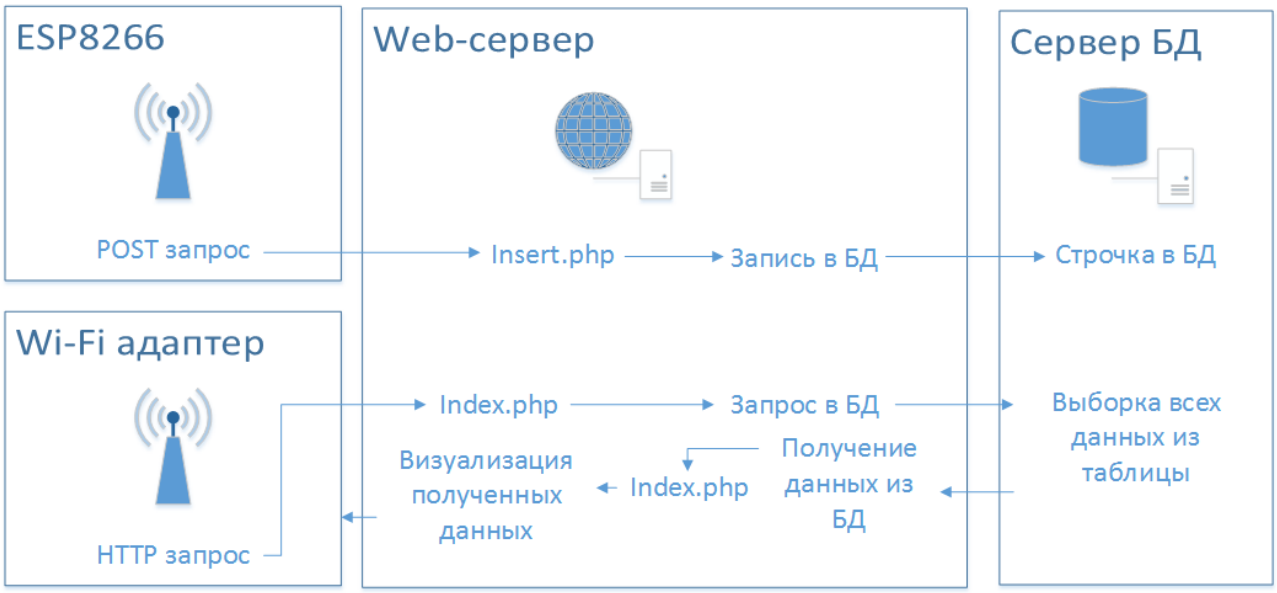


Рисунок 14 – Структура работы скриптов для записи, чтения и визуализации данных, полученных от шлюза

POST запрос в нашем случае будет выглядеть следующим образом:

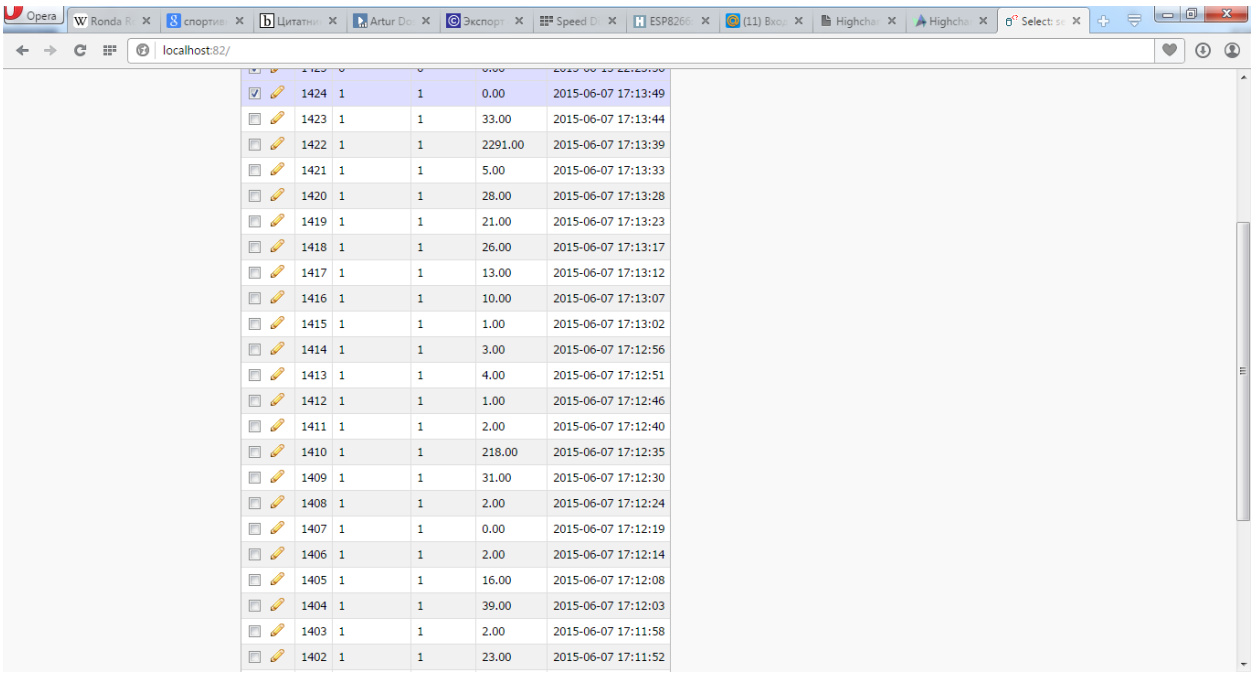
POST http://192.168.1.100/insert.php HTTP/1.0
Content-Type:application/x-www-form-urlencoded
Content-Length:35
sensnodeid=1&senstype=1&sensvalue=1

Таким образом, мы осуществляем отправку данных скрипту insert.php, находящемуся по адресу 192.168.1.100, используем протокол HTTP версии 1.0, тип данных – application/x-www-form-urlencoded, который используется при отправке из HTML форм. Длина информации составит 35 символов или 35 байт, и сама информация «sensnodeid=1&senstype=1&sensvalue=1», которая будет обработана на серверной стороне и разбита на части в зависимости от присвоенных переменных.

На рисунке 16 приведен пример веб-страницы с данными от датчика вибраций за короткий промежуток времени. Библиотека Highchart автоматически подбирает временной интервал на основании полученных данных. На рисунке 15 приведена структура таблицы из базы данных, куда сохраняются следующие данные:

- ID ячейки таблицы;
- Номер узла;
- Тип датчика;
- Значение датчика;
- Текущее время.

Для работы с базами данных на PHP была использована программа Adminer [26].



1424	1	1	0.00	2015-06-07 17:13:49
1423	1	1	33.00	2015-06-07 17:13:44
1422	1	1	2291.00	2015-06-07 17:13:39
1421	1	1	5.00	2015-06-07 17:13:33
1420	1	1	28.00	2015-06-07 17:13:28
1419	1	1	21.00	2015-06-07 17:13:23
1418	1	1	26.00	2015-06-07 17:13:17
1417	1	1	13.00	2015-06-07 17:13:12
1416	1	1	10.00	2015-06-07 17:13:07
1415	1	1	1.00	2015-06-07 17:13:02
1414	1	1	3.00	2015-06-07 17:12:56
1413	1	1	4.00	2015-06-07 17:12:51
1412	1	1	1.00	2015-06-07 17:12:46
1411	1	1	2.00	2015-06-07 17:12:40
1410	1	1	218.00	2015-06-07 17:12:35
1409	1	1	31.00	2015-06-07 17:12:30
1408	1	1	2.00	2015-06-07 17:12:24
1407	1	1	0.00	2015-06-07 17:12:19
1406	1	1	2.00	2015-06-07 17:12:14
1405	1	1	16.00	2015-06-07 17:12:08
1404	1	1	39.00	2015-06-07 17:12:03
1403	1	1	2.00	2015-06-07 17:11:58
1402	1	1	23.00	2015-06-07 17:11:52

Рисунок 15 – Структура таблицы из базы данных, куда сохраняются данные с датчиков

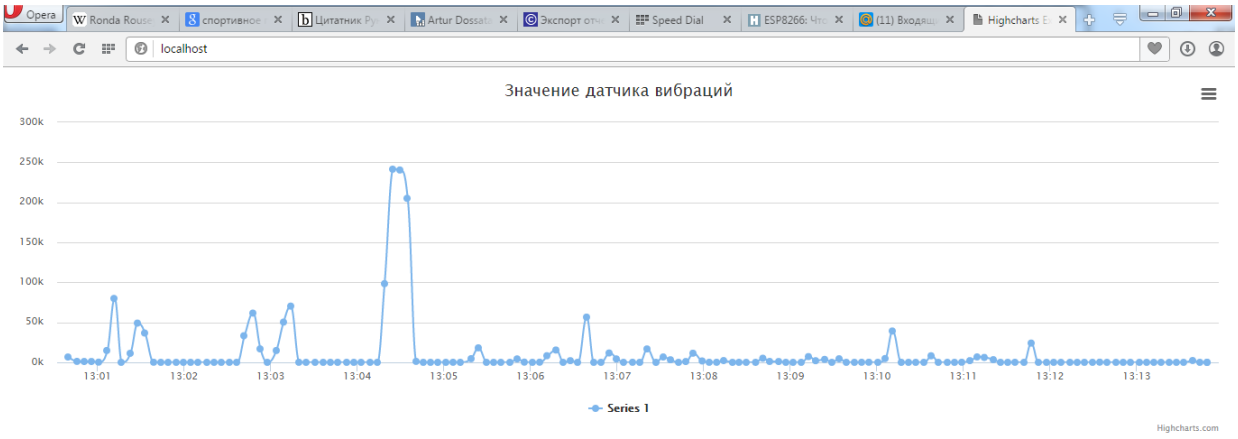


Рисунок 16 – Веб-страница с визуализацией данных, полученных от датчика

5.4 Тестирование разработанного ПО для визуализации данных с датчиков

Для тестирования была собрана схема, приведенная на рисунке 17. Датчик вибраций был подключен по схеме, приведенной на рисунке 18, исходя из схемы чем большее сопротивление у датчика, тем выше напряжение подается на АЦП микроконтроллера. В тоже время сопротивление датчика меняется в зависимости от уровня вибраций, чем выше вибрации тем ниже сопротивление, соответственно и ниже показания на выходе АЦП.

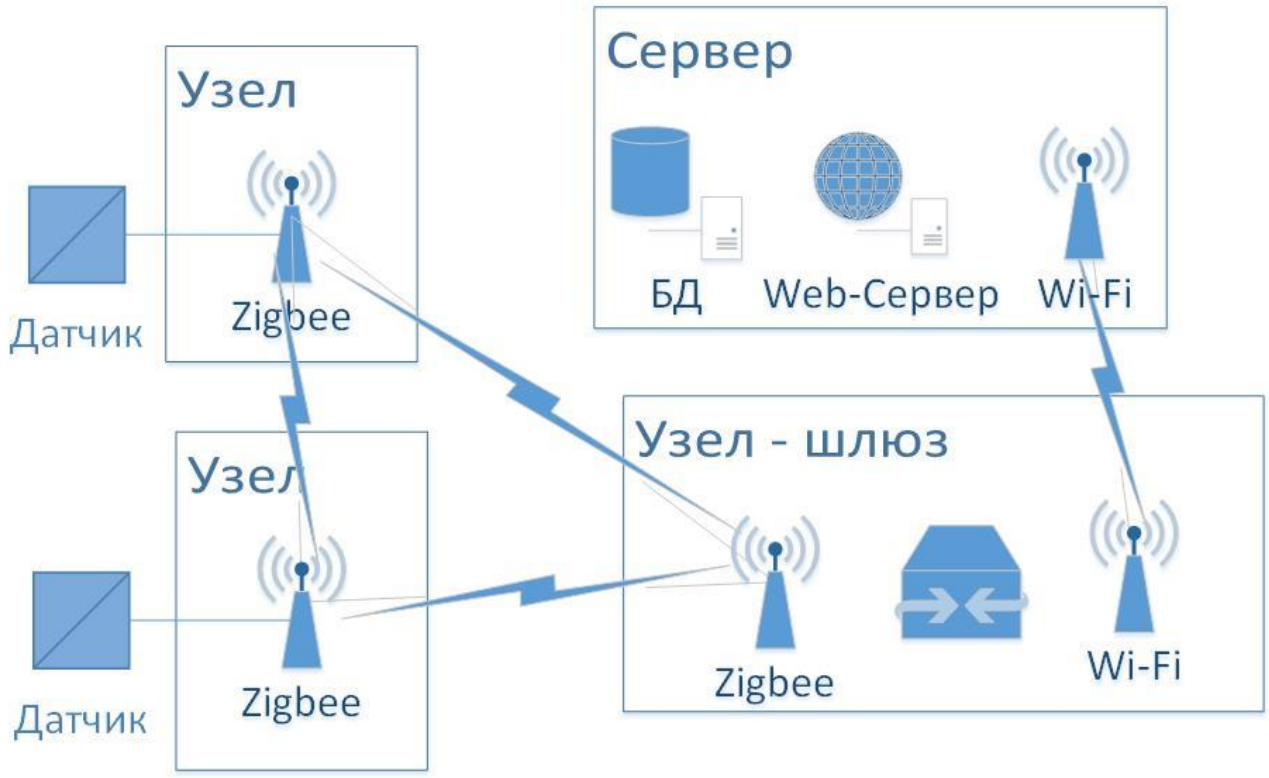


Рисунок 17 – Схема организации платформы

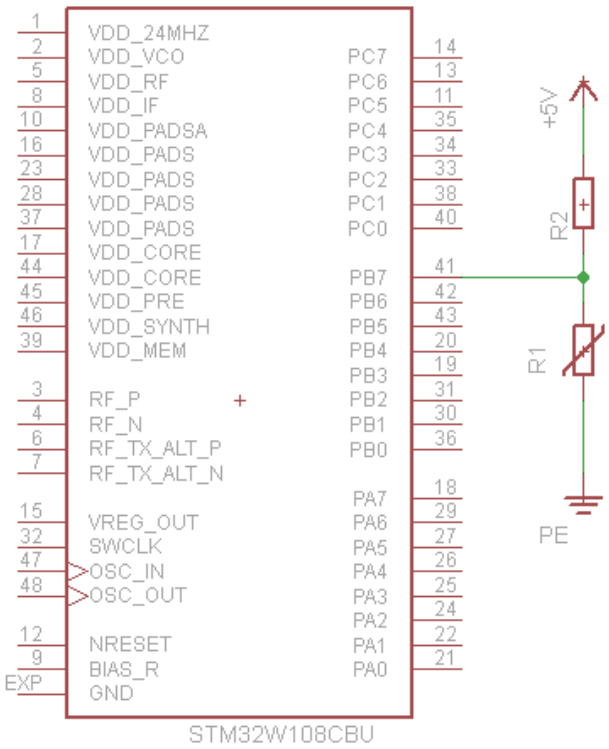


Рисунок 18 – Типовая схема включения датчика вибраций

Так как узел на базе микроконтроллера STM32W не был собран, то был использован модуль Arduino, с загруженным скетчем, который приведен в приложении М. Скетч взаимодействует с модулем ESP8266 по протоколу SPI,

отправляя POST запросы с данными датчика. Данные с датчика вибраций снимаются при помощи встроенного в микроконтроллером АЦП. Таким образом, мы можем протестировать работу ПО для визуализации данных и в целом работу платформы для беспроводной сенсорной сети.

6 Экономический расчет

В этой главе будет приведен экономический расчет производства платформы для беспроводной сенсорной сети.

В таблице приведена стоимость компонентов для производства узлов беспроводной сенсорной сети.

Таблица 3 – Стоимость компонентов для узла-шлюза беспроводной сенсорной сети

Наименование элемента	Цена
STM32W108	$7.07
ESP8266	$4
Антенна AN0835-245	$1.66
Резисторы, конденсаторы и прочие детали	$0.2
Печатная плата	$0.05
Итого	$12.98

Таким образом, итоговая стоимость узла-шлюза составляет $12.98, в то время как цена узла составит $8.98.

Средняя цена платформы будет зависеть от примененных в ней датчиков, базовая же стоимость составит $21.96, из двух узлов без системы питания.

Заключение

Итогом работы явилась разработанная платформа для беспроводной сенсорной сети с визуализацией данных, в основе которой находится узел на базе микроконтроллера STM32W108CB, шлюз с трансивером ESP8266 и серверной частью на основе веб-сервера NGINX.

Одной из главных целей, поставленных в работе была доступность продукта. Итоговая стоимость продукта составит $12.98 для узла-шлюза и $8.98 для оконечных узлов.

В работе был проведен подробный анализ выбора конкретного веб-сервера. В ходе работы был проведен маркетинговый анализ рынка микроконтроллеров для беспроводных сенсорных сетей, рынка трансиверов Wi-Fi – UART, для взаимодействия микроконтроллера с внешним миром. Также был проведен анализ операционных систем для беспроводных сенсорных сетей.

В тоже время было разработано программное обеспечение для визуализации данных, полученных с датчиков. Было использовано решение в виде веб-сервера с использованием баз данных. Таким образом, информация с датчиков на сенсорных узлах сначала передается узлу-шлюзу, а уже впоследствии предается на сервер в локальной сети с помощью Wi-Fi – UART трансивера. Полученные данные записываются в базу данных на сервере, а любой пользователь, подключившийся к локальной сети может визуально просмотреть эти данные.

Список сокращений

RFID	Radio Frequency Identification
WSN	Wireless Sensors Network
M2M	Machines-to-Machines
UDP	User Datagram Protocol
UART	Universal asynchronous receiver/transmitter
TCP	Transmission Control Protocol
SPI	Serial Peripheral Interface
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
ROM	Read-only memory
RAM	Random-access memory
OS	Operating system
OFDM	Orthogonal frequency-division multiplexing
MAC	Media access control
IP	Internet Protocol
ID	Identification
HTTP	Hyper text type protocol
HTML	Hyper text Markup language
DB	Database
CMOS	Complementary metal–oxide–semiconducto
AP	Access Point
АЦП	Аналогово-цифровой преобразователь
БД	База данных
ВТ	Вычислительная техника
ЛУТ	Лазерно-утюжная технология
МК	Микроконтроллер
ОЗУ	Оперативное запоминающее устройство
ОС	Операционная система
ПЗУ	Постоянно запоминающее устройство
ПО	Программное обеспечение

Список литературы

1. Dargie, W и Poellabauer, C. *Fundamentals of wireless sensor networks: theory and practice.* Singapore : б.н., 2010.
2. Wireless sensor network. *Wikipedia.* [В Интернете] [Цитировано: 02 Август 2014 г.] http://en.wikipedia.org/wiki/Wireless_sensor_network.
3. DigiKey.com. [В Интернете] http://digikey.com.
4. PCB Design Guidelines. [В Интернете] Май 2015 г. [Цитировано: 24 Апрель 2015 г.] http://www.st.com/web/en/resource/technical/document/application_note/CD00270908.pdf.
5. Официальный сайт программы EagleCad. [В Интернете] [Цитировано: 25 Май 2015 г.] http://www.cadsoftusa.com/.
6. Обзор растворов для травления меди. [В Интернете] [Цитировано: 31 Май 2015 г.] http://radiokot.ru/lab/hardwork/62/.
7. Создание печатной платы методом ЛУТ. [В Интернете] [Цитировано: 29 Май 2015 г.] http://easyelectronics.ru/sozdanie-pechatnoj-platy-metodom-lazernogo-utyuga.html.
8. Перечень модулей Wi-Fi - UART на сайте digikey. [В Интернете] [Цитировано: 28 Май 2015 г.] http://www.digikey.com/product-search/en?FV=.
9. ESP8266 Datasheet. [В Интернете] [Цитировано: 29 Март 2015 г.] http://www.adafruit.com/datasheets/ESP8266_Specifications_English.pdf.
10. Datasheet CC3200. [В Интернете] [Цитировано: 29 Март 2015 г.] http://www.ti.com/lit/ds/symlink/cc3200mod.pdf.
11. Datasheet CC3100. [В Интернете] [Цитировано: 29 Март 2015 г.] http://www.ti.com/lit/ds/symlink/cc3100mod.pdf.
12. Цена на модуль ESP8266. [В Интернете] [Цитировано: 02 Июнь 2015 г.] http://ru.aliexpress.com/item/Free-shipping-ESP8266-serial-WIFI-wireless-module-wireless-transceiver/32341788594.html.
13. Цена на модуль CC3200. [В Интернете] [Цитировано: 02 Июнь 2015 г.] http://ru.aliexpress.com/item/1PCS-IV-CC3200MOD-CC3200-Wifi-Module-CC3200-LAUNCHXL-CC3200R1M2RGC/32311425905.html.
14. Описание модуля ESP8266. [В Интернете] [Цитировано: 16 Май 2015 г.] http://habrahabr.ru/company/coolrf/blog/238443/.
15. Сравнение ОС для беспроводных сенсорных сетей. [В Интернете] [Цитировано: 14 Февраль 2015 г.] http://www.riot-os.org.
16. История создания TinyOS. [В Интернете] [Цитировано: 07 Ноябрь 2014 г.] http://tinyos.stanford.edu/tinyos-wiki/index.php/FAQ.
17. About. *RIOT OS.* [В Интернете] [Цитировано: 08 май 2015 г.] http://riot-os.org/#about.
18. The history of contiki. *Contiki OS.* [В Интернете] [Цитировано: 15 Апрель 2015 г.] http://www.contiki-os.org/community.html.
19. Документация к Contiki OS. [В Интернете] [Цитировано: 07 Апрель 2015 г.] http://contiki.sourceforge.net/docs/2.6/index.html.

20. Платформы, поддерживаемы Contiki OS. [В Интернете] [Цитировано: 06 Июнь 2015 г.] http://www.contiki-os.org/hardware.html.
21. Компиляция примеров в Contiki OS. [В Интернете] [Цитировано: 03 Июнь 2015 г.] http://www.contiki-os.org/start.html.
22. May 2015 Web server survey. [В Интернете] [Цитировано: 01 Июнь 2015 г.] http://news.netcraft.com/archives/category/web-server-survey/.
23. Web Server Performance Comparison. [В Интернете] [Цитировано: 28 Апрель 2015 г.] http://wiki.dreamhost.com/Web_Server_Performance_Comparison.
24. Официальная страница продукта Winginx. [В Интернете] [Цитировано: 04 Июнь 2015 г.] http://winginx.com/en/download.
25. Настройки веб-сервера Nginx. [В Интернете] [Цитировано: 23 Май 2015 г.] http://nginx.org/en/docs/http/server_names.html.
26. СУБД Adminer. [В Интернете] [Цитировано: 02 Июнь 2015 г.] http://www.adminer.org.
27. Datasheet of STM32W108. [В Интернете] http://www.st.com/web/en/resource/technical/document/datasheet/CD00248316.pdf.
28. Atmega128RFA1 Price. [В Интернете] [Цитировано: 11 Май 2015 г.] http://www.digikey.com/product-detail/en/ATMEGA128RFA1-ZU/ATMEGA128RFA1-ZU-ND/2208800.
29. Datasheet of AtmegaRFA1. [В Интернете] [Цитировано: 29 Март 2015 г.] http://www.atmel.com/Images/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf.
30. Datasheet of CC2530. [В Интернете] [Цитировано: 29 Март 2015 г.] http://www.ti.com/lit/ds/symlink/cc2530.pdf.
31. Datasheet of MC1323. [В Интернете] [Цитировано: 29 Март 2015 г.] http://cache.freescale.com/files/rf_if/doc/data_sheet/MC13237.pdf.
32. Price TI CC2530. [В Интернете] [Цитировано: 01 Июнь 2015 г.] http://www.digikey.com/product-detail/en/CC2530F128RHAR/296-38898-1-ND/5143005.
33. Freescale MC1323 Price. [В Интернете] [Цитировано: 01 Июнь 2015 г.] http://www.digikey.com/product-detail/en/MC13237CHT/MC13237CHT-ND/4234642.
34. Price of STM32W. [В Интернете] [Цитировано: 01 Июнь 2015 г.]

Приложение А

Таблица А1 – Сравнительная таблица основных микроконтроллеров [31] [29] [30] [27]

Наименование параметра	Freescale MC1323	Atmel Atmega128RFA1	ST STM32W108	TI CC2530
Выходная мощность, дБм	-30 – +2	-17 – +3.5	-55 - +5	0 – +4.5
Чувствительность, дБм	-94	-100	-100	-97
Архитектура процессора	ARM7 32bit (RISC)	8-bit (RISC)	ARM-M3 32bit (RISC)	8051 8-bit (CISC)
Частота процессора, МГц	24	16	24	-
Мощность потребления при приеме, мА	26,6	12,5	22	24
Мощность потребления при передаче, мА	34,2	14,5	26	29
Мощность потребления ядра в активном режиме, мА	4,7	4,1	7,5	0,2
Потребление в режиме сна, мкА	0,56	0,25	0,4	0,4
Рабочее напряжение, В	1,8 – 3,6	1,8 – 3,6	2,1 – 3,6	2 – 3,6
Разрешение АЦП, бит	12	10	12	12
АЦП, каналов	8	8	6	8
UART, каналов	2	2	2	2
Порты ввода-вывода, штук	32	38	24	21
Флэщ-память, Кбайт	128	128	128	128
Статическая ОЗУ, Кбайт	8	16	8	8
ПЗУ, Кбайт	80	4	-	-

Сравнительная таблица основных микроконтроллеров

Приложение Б

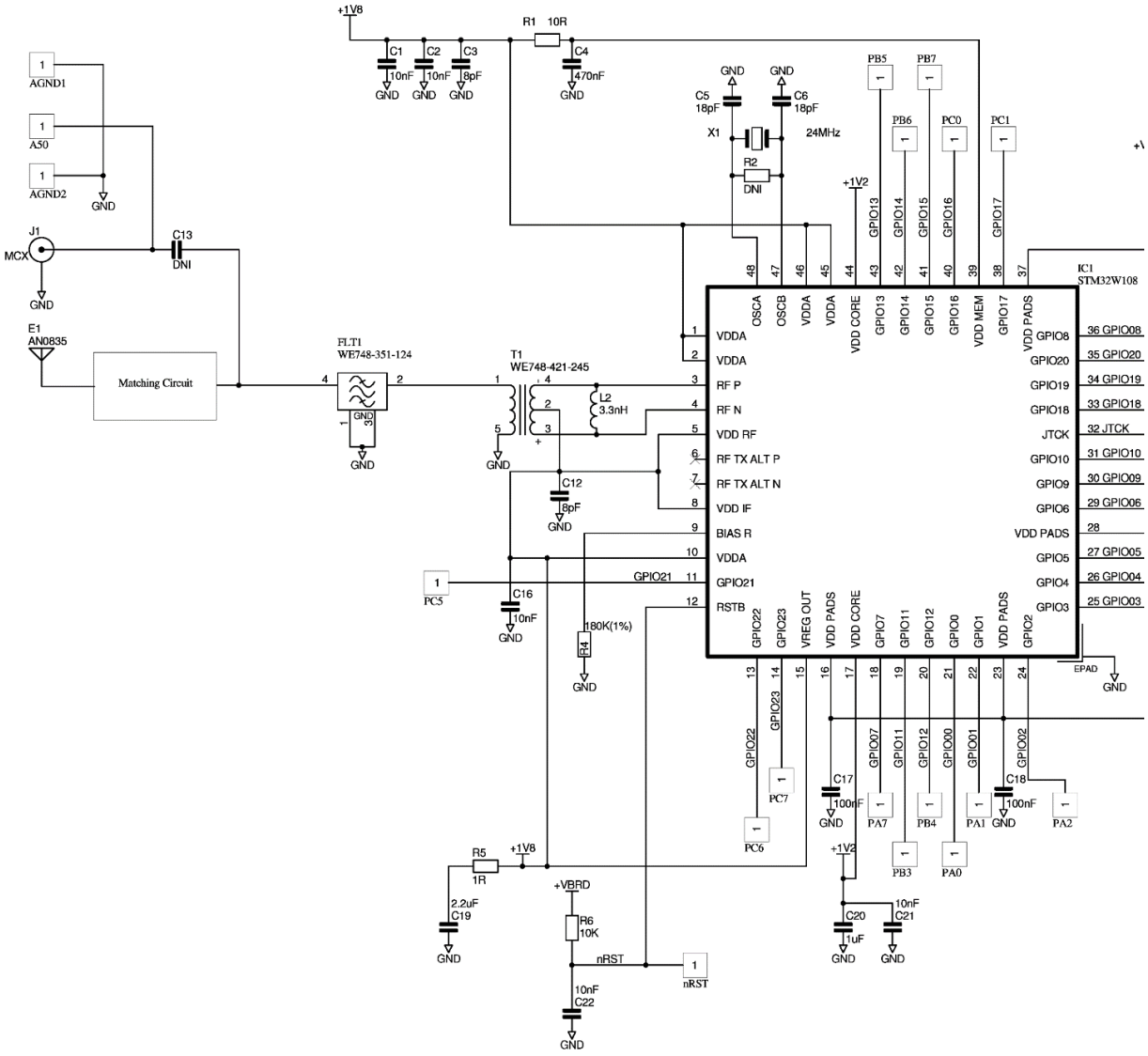


Рисунок Б1 – Принципиальная схема узла на базе STM32W108

Принципиальная схема узла на базе микроконтроллера STM32W108CB

Приложение В

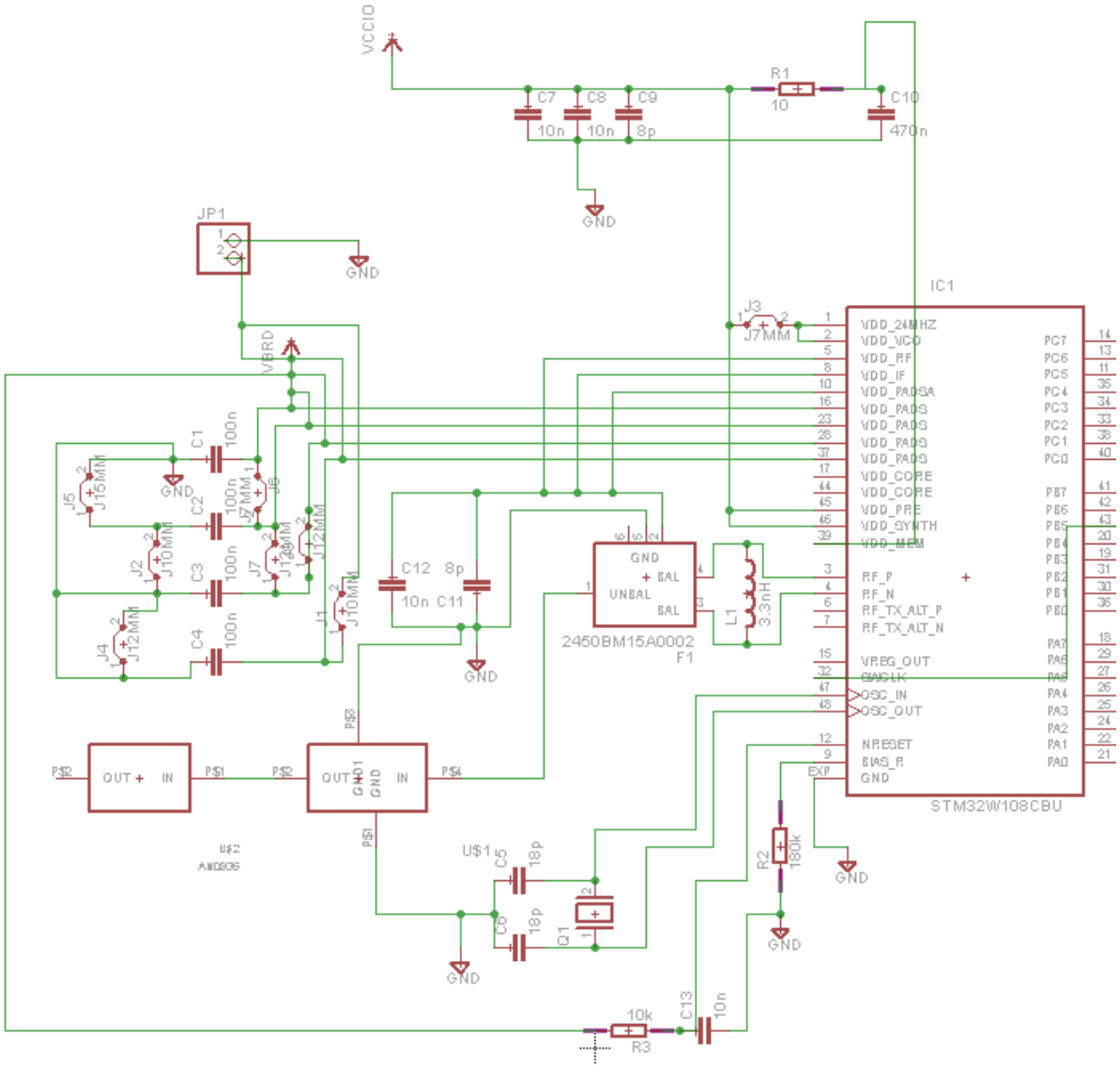


Рисунок В1 – Схема узла в программе EagleCAD версии 7.1

Схема узла в программе EagleCAD версии 7.1

Приложение Г
Исходный код программы UDP-client.c

```
#include “contiki.h”
#include “lib/random.h”
#include “sys/ctimer.h”
#include “net/uip.h”
#include “net/uip-ds6.h”
#include “net/uip-udp-packet.h”
#include “sys/ctimer.h”
#ifdef WITH_COMPOWER
#include “powertrace.h”
#endif
#include <stdio.h>
#include <string.h>

#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT 5678

#define UDP_EXAMPLE_ID  190

#define DEBUG DEBUG_PRINT
#include “net/uip-debug.h”

#ifndef PERIOD
#define PERIOD 60
#endif

#define START_INTERVAL          (15 * CLOCK_SECOND)
#define SEND_INTERVAL           (PERIOD * CLOCK_SECOND)
#define SEND_TIME           (random_rand() % (SEND_INTERVAL))
#define MAX_PAYLOAD_LEN             30

static struct uip_udp_conn *client_conn;
static uip_ipaddr_t server_ipaddr;

/*---------------------------------------------------------------------------*/
PROCESS(udp_client_process, “UDP client process”);
AUTOSTART_PROCESSES(&udp_client_process);
/*---------------------------------------------------------------------------*/
static void
tcpip_handler(void)
{
```

```
 char *str;

  if(uip_newdata()) {
    str = uip_appdata;
    str[uip_datalen()] = '\0';
    printf("DATA recv '%s'\n", str);
  }
}
/*---------------------------------------------------------------------------*/
static void
send_packet(void *ptr)
{
  static int seq_id;
  char buf[MAX_PAYLOAD_LEN];

  seq_id++;
  PRINTF("DATA send to %d 'Hello %d'\n",
         server_ipaddr.u8[sizeof(server_ipaddr.u8) – 1], seq_id);
  41print(buf, "Hello %d from the client", seq_id);
  uip_udp_packet_sendto(client_conn, buf, strlen(buf),
                        &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}
/*---------------------------------------------------------------------------*/
static void
print_local_addresses(void)
{
  int i;
  uint8_t state;

  PRINTF("Client Ipv6 addresses: ");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
      PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
      PRINTF("\n");
      /* hack to make address "final" */
      if (state == ADDR_TENTATIVE) {
         uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
      }
    }
  }
```

```
}
/*---------------------------------------------------------------------------*/
static void
set_global_address(void)
{
  uip_ipaddr_t ipaddr;

  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
  uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);

/* The choice of server address determines its 6LoPAN header compression.
 * (Our address will be compressed Mode 3 since it is derived from our link-local address)
 * Obviously the choice made here must also be selected in udp-server.c.
 *
 * For correct Wireshark decoding using a sniffer, add the /64 prefix to the 6LowPAN protocol preferences,
 * e.g. set Context 0 to aaaa::.  At present Wireshark copies Context/128 and then overwrites it.
 * (Setting Context 0 to aaaa::1111:2222:3333:4444 will report a 16 bit compressed address of aaaa::1111:22ff:fe33:xxxx)
 *
 * Note the IPCMV6 checksum verification depends on the correct uncompressed addresses.
 */

#if 0
/* Mode 1 – 64 bits inline */
   uip_ip6addr(&server_ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 1);
#elif 1
/* Mode 2 – 16 bits inline */
  uip_ip6addr(&server_ipaddr, 0xaaaa, 0, 0, 0, 0, 0x00ff, 0xfe00, 1);
#else
/* Mode 3 – derived from server link-local (MAC) address */
  uip_ip6addr(&server_ipaddr, 0xaaaa, 0, 0, 0, 0x0250, 0xc2ff, 0xfea8, 0xcd1a);
//redbee-econotag
#endif
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(udp_client_process, ev, data)
{
```

```
  static struct etimer periodic;
  static struct ctimer backoff_timer;
#if WITH_COMPOWER
  static int print = 0;
#endif

  PROCESS_BEGIN();

  PROCESS_PAUSE();

  set_global_address();

  PRINTF("UDP client process started\n");

  print_local_addresses();

  /* new connection with remote host */
  client_conn = udp_new(NULL, UIP_HTONS(UDP_SERVER_PORT), NULL);
  if(client_conn == NULL) {
    PRINTF("No UDP connection available, exiting the process!\n");
    PROCESS_EXIT();
  }
  udp_bind(client_conn, UIP_HTONS(UDP_CLIENT_PORT));

  PRINTF("Created a connection with the server ");
  PRINT6ADDR(&client_conn->ripaddr);
  PRINTF(" local/remote port %u/%u\n",
        UIP_HTONS(client_conn->lport), UIP_HTONS(client_conn->rport));

#if WITH_COMPOWER
  powertrace_sniff(POWERTRACE_ON);
#endif

  etimer_set(&periodic, SEND_INTERVAL);
  while(1) {
    PROCESS_YIELD();
    if(ev == tcpip_event) {
      tcpip_handler();
    }

    if(etimer_expired(&periodic)) {
      etimer_reset(&periodic);
```

```
      ctimer_set(&backoff_timer, SEND_TIME, send_packet, NULL);

#if WITH_COMPOWER
          if (print == 0) {
          powertrace_print("#P");
      }
      if (++print == 3) {
          print = 0;
      }
#endif

    }
  }

  PROCESS_END();
}
```

Приложение Д
Исходный код программы UDP-server.c

```
#include "contiki.h"
#include "contiki-lib.h"
#include "contiki-net.h"
#include "net/uip.h"
#include "net/rpl/rpl.h"

#include "net/netstack.h"
#include "dev/button-sensor.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define DEBUG DEBUG_PRINT
#include "net/uip-debug.h"

#define UIP_IP_BUF   ((struct uip_ip_hdr *)&uip_buf[UIP_LLH_LEN])

#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT5678

#define UDP_EXAMPLE_ID  190

static struct uip_udp_conn *server_conn;

PROCESS(udp_server_process, "UDP server process");
AUTOSTART_PROCESSES(&udp_server_process);
/*---------------------------------------------------------------------------*/
static void
tcpip_handler(void)
{
  char *appdata;

  if(uip_newdata()) {
    appdata = (char *)uip_appdata;
    appdata[uip_datalen()] = 0;
    PRINTF("DATA recv '%s' from ", appdata);
    PRINTF("%d",
          UIP_IP_BUF->srcipaddr.u8[sizeof(UIP_IP_BUF->srcipaddr.u8) - 1]);
    PRINTF("\n");
```

```
#if SERVER_REPLY
    PRINTF("DATA sending reply\n");
    uip_ipaddr_copy(&server_conn->ripaddr, &UIP_IP_BUF->srcipaddr);
    uip_udp_packet_send(server_conn, "Reply", sizeof("Reply"));
    uip_create_unspecified(&server_conn->ripaddr);
#endif
  }
}
/*---------------------------------------------------------------------------*/
static void
print_local_addresses(void)
{
  int i;
  uint8_t state;

  PRINTF("Server IPv6 addresses: ");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(state == ADDR_TENTATIVE || state == ADDR_PREFERRED) {
      PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
      PRINTF("\n");
      /* hack to make address "final" */
      if (state == ADDR_TENTATIVE) {
         uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
      }
    }
  }
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(udp_server_process, ev, data)
{
  uip_ipaddr_t ipaddr;
  struct uip_ds6_addr *root_if;

  PROCESS_BEGIN();

  PROCESS_PAUSE();

  SENSORS_ACTIVATE(button_sensor);

  PRINTF("UDP server started\n");
```

```
#if UIP_CONF_ROUTER
/* The choice of server address determines its 6LoPAN header compression.
 * Obviously the choice made here must also be selected in udp-client.c.
 *
 * For correct Wireshark decoding using a sniffer, add the /64 prefix to the 6LowPAN protocol preferences,
 * e.g. set Context 0 to aaaa::.  At present Wireshark copies Context/128 and then overwrites it.
 * (Setting Context 0 to aaaa::1111:2222:3333:4444 will report a 16 bit compressed address of aaaa::1111:22ff:fe33:xxxx)
 * Note Wireshark's IPCMV6 checksum verification depends on the correct uncompressed addresses.
 */

#if 0
/* Mode 1 - 64 bits inline */
   uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 1);
#elif 1
/* Mode 2 - 16 bits inline */
  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0x00ff, 0xfe00, 1);
#else
/* Mode 3 - derived from link local (MAC) address */
  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
#endif

  uip_ds6_addr_add(&ipaddr, 0, ADDR_MANUAL);
  root_if = uip_ds6_addr_lookup(&ipaddr);
  if(root_if != NULL) {
    rpl_dag_t *dag;
    dag = rpl_set_root(RPL_DEFAULT_INSTANCE,(uip_ip6addr_t *)&ipaddr);
    uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
    rpl_set_prefix(dag, &ipaddr, 64);
    PRINTF("created a new RPL dag\n");
  } else {
    PRINTF("failed to create a new RPL DAG\n");
  }
#endif /* UIP_CONF_ROUTER */

  print_local_addresses();

  /* The data sink runs with a 100% duty cycle in order to ensure high
```

```
   packet reception rates. */
  NETSTACK_MAC.off(1);

  server_conn = udp_new(NULL, UIP_HTONS(UDP_CLIENT_PORT), NULL);
  if(server_conn == NULL) {
    PRINTF("No UDP connection available, exiting the process!\n");
    PROCESS_EXIT();
  }
  udp_bind(server_conn, UIP_HTONS(UDP_SERVER_PORT));

  PRINTF("Created a server connection with remote address ");
  PRINT6ADDR(&server_conn->ripaddr);
  PRINTF(" local/remote port %u/%u\n", UIP_HTONS(server_conn->lport),
        UIP_HTONS(server_conn->rport));

  while(1) {
    PROCESS_YIELD();
    if(ev == tcpip_event) {
      tcpip_handler();
    } else if (ev == sensors_event && data == &button_sensor) {
      PRINTF("Initiaing global repair\n");
      rpl_repair_root(RPL_DEFAULT_INSTANCE);
    }
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------*/
```

Приложение Е
Конфигурация сервера, для доступа к веб-странице из локальной сети

```
server {
        listen 80 default_server;
        server_name _;
        root home/localhost/public_html;

        index index.php index.html;

        log_not_found off;
        access_log logs/maxsite.local-access.log;

        charset utf-8;

        location ~ /\. { allow all; }
        location = /favicon.ico { }
        location = /robots.txt { }

        location /
        {
        try_files $uri $uri/ /index.php?q=$uri&$args;
        }

        location ~ \.php$ {
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root/$fastcgi_script_name;
        include fastcgi_params;
}       }
```

Приложение Ж
Список команд модуля ESP8266:

AT+RST – перезагрузка модуля;
AT+CWJAP="Alma_TV","" – подключение к точке доступа с SSID:Alma_TV;
AT+CIFSR – вывести полученный IP-адрес;
AT+CWMODE=1 – режим работы модуля ESP8266 (1-STA, 2-AP, 3-Both)
AT+CIPMUX=1 – режим одновременной работы;
AT+CIPSTART=4,"TCP","192.168.1.10x",80 – начать TCP сессию
AT+CIPSEND=4,xxx – отправить xxx байт

Как отправить POST запрос на сервер:

POST http://192.168.1.100/insert.php HTTP/1.0 (one enter \r\n)
Content-Type:application/x-www-form-urlencoded (one enter \r\n)
Content-Length:16 (two enter \r\n\r\n)
user_name=21389219 (two enter \r\n\r\n)

Приложение И
Скрипт для взаимодействия с БД

```
<?php
   class mysql_db{

//+=====================================================+
      function sql_connect($sqlserver, $sqluser, $sqlpassword, $database){
         $this->connect_id = mysql_connect($sqlserver, $sqluser, $sqlpassword);
         if($this->connect_id){
            if (mysql_select_db($database)){
               return $this->connect_id;
            }else{
               return $this->error();
            }
         }else{
            return $this->error();
         }
      }

//+=====================================================+
      function error(){
         if(mysql_error() != ''){
            echo '<b>MySQL Error</b>: '.mysql_error().'<br/>';
         }
      }

//+=====================================================+
      function query($query){
         if ($query != NULL){
            $this->query_result = mysql_query($query, $this->connect_id);
            if(!$this->query_result){
               return $this->error();
            }else{
               return $this->query_result;
            }
         }else{
            return '<b>MySQL Error</b>: Empty Query!';
         }
      }

//+=====================================================+
      function get_num_rows($query_id = ""){
```

```
    if($query_id == NULL){
            $return = mysql_num_rows($this->query_result);
        }else{
            $return = mysql_num_rows($query_id);
        }
        if(!$return){
            $this->error();
        }else{
            return $return;
        }
    }

//+=========================================================+
    function fetch_row($query_id = ""){
        if($query_id == NULL){
            $return = mysql_fetch_array($this->query_result);
        }else{
            $return = mysql_fetch_array($query_id);
        }
        if(!$return){
            $this->error();
        }else{
            return $return;
        }
    }

//+=========================================================+
    function get_affected_rows($query_id = ""){
        if($query_id == NULL){
            $return = mysql_affected_rows($this->query_result);
        }else{
            $return = mysql_affected_rows($query_id);
        }
        if(!$return){
            $this->error();
        }else{
            return $return;
        }
    }

//+=========================================================+
    function sql_close(){
```

Окончание приложения И

```
        if($this->connect_id){
            return mysql_close($this->connect_id);
        }
    }

//+=========================================================+
  }

?>
```

Приложение К
Скрипт для сохранения данных, полученных от сенсора в БД

```
<?php
include 'mysql_class.php';
?>
<?php
$sensnode=$_POST['sensnodeid'];
$senstype=$_POST["senstype"];
$sensvalue=$_POST["sensvalue"];


        $DB = new mysql_db();
   $DB->sql_connect('localhost', 'root', '', 'wsn_db');
   $DB->query("INSERT INTO `sensors_values` (`sensnodeid`, `senstype`,
`sensvalue`, `datetime`)
VALUES ('$sensnode', '$senstype', '$sensvalue', now());");
   $DB->sql_close();

?>
```

Приложение Л
PHP скрипт для визуализации данных с выборкой из БД

```
<?php
include 'mysql_class.php';
?>
<?php

$DB = new mysql_db();
$DB->sql_connect('localhost', 'root', '', 'wsn_db');
$result=$DB->query("SELECT `sensvalue`, `datetime` FROM `sensors_values`
ORDER BY `datetime` LIMIT 5000");
?>

<!DOCTYPE HTML>
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
                <meta http-equiv="refresh" content="10">
                <title>Highcharts Example</title>

                <script type="text/javascript" src="js/jquery.min.js"></script>
                <style type="text/css">
${demo.css}
                </style>
                <script type="text/javascript">
                var time=Date.now();
$(function () {
   $('#container').highcharts({
      chart: {
         type: 'spline'
      },
      title: {
         text: 'Значение датчика вибраций'
      },
      xAxis: {
         type: 'datetime',
                        floor: time-24*3600*1000
                        },
      yAxis: {
         title: {
            text: 'Показания датчика'
```

```
            },
            min: 0
        },
        tooltip: {
        },

        plotOptions: {
            spline: {
                marker: {
                    enabled: true
                }
            }
        },

        series: [{
            // Define the data points. All series have a dummy year
            // of 1970/71 in order to be compared on the same x axis. Note
            // that in JavaScript, months start at 0 for January, 1 for February etc.
            data: [
<?php
while($row=$DB->fetch_row($result))
{
echo '[';
$date=new DateTime($row[1]);
print($date->format('U')*1000);
echo ',';
print($row[0]);
echo '],';
echo "\n";
}
echo '['; echo $date->format('U')*1000; echo ',0]'; echo "\n";
?>

            ]
        }]
    });
});
                </script>
        </head>
        <body>
<script src="js/highcharts.js"></script>
<script src="js/modules/exporting.js"></script>
```

Окончание приложения И

```
<div id="container" style="min-width: 310px; height: 400px; margin: 0
auto"></div>

        </body>
</html>


<?php
$DB->sql_close();
?>

<?php
$DB->sql_close();
?>
```

Приложение М
Скетч Arduino Mega256 для работы с модулем ESP8266 и отправки значений с сенсора с помощью АЦП

```
unsigned long sensvalue=0;
unsigned long time=0;
unsigned long time2=0;

#include "ESP8266.h"
#include "string.h"
#define SSID        "WSN_Test"
#define PASSWORD    ""
#define HOST_NAME   "192.168.1.100"
#define HOST_PORT   80

ESP8266 wifi(Serial1);

void setup(void)
{
   Serial.begin(115200);
   Serial.print("setup begin\r\n");

   Serial.print("FW Version: ");
   Serial.println(wifi.getVersion().c_str());


   if (wifi.setOprToStationSoftAP()) {
      Serial.print("to station + softap ok\r\n");
   } else {
      Serial.print("to station + softap err\r\n");
   }

   if (wifi.joinAP(SSID, PASSWORD)) {
      Serial.print("Join AP success\r\n");
      Serial.print("IP: ");
      Serial.println(wifi.getLocalIP().c_str());
   } else {
      Serial.print("Join AP failure\r\n");
   }

   if (wifi.enableMUX()) {
      Serial.print("multiple ok\r\n");
   } else {
      Serial.print("multiple err\r\n");
```

```
    }

    Serial.print("setup end\r\n");
    time=millis();
    time2=millis();
}

void loop(void)
{
    if(millis()-time2>20)
    {
    sensvalue+=analogRead(A9);
    time2=millis();
    }

    if(millis()-time>5000)
    {
    uint8_t buffer[128] = {0};
    static uint8_t mux_id = 4;

    if (wifi.createTCP(mux_id, HOST_NAME, 80)) {
        Serial.print("create tcp ");
        Serial.print(mux_id);
        Serial.println(" ok");
    } else {
        Serial.print("create tcp ");
        Serial.print(mux_id);
        Serial.println(" err");
    }
    String aaa=String(sensvalue);
    char *hello = "POST / HTTP/1.0\r\nContent-Type:application/x-www-form-
urlencoded\r\nContent-Length:40\r\n\r\nsensnodeid=1&senstype=1&sensvalue=
\r\n\r\n";
    (*(hello+120))=aaa[0];
    (*(hello+121))=aaa[1];
    (*(hello+122))=aaa[2];
    (*(hello+123))=aaa[3];
    (*(hello+124))=aaa[4];
    (*(hello+125))=aaa[5];
    if (wifi.send(mux_id, (const uint8_t*)hello, strlen(hello))) {
        Serial.println("send ok");
    } else {
```

```
    Serial.println("send err");
  }

  uint32_t len = wifi.recv(mux_id, buffer, sizeof(buffer), 10000);
  if (len > 0) {
    Serial.print("Received:[");
    for(uint32_t i = 0; i < len; i++) {
      Serial.print((char)buffer[i]);
    }
    Serial.print("]\r\n");
  }

  if (wifi.releaseTCP(mux_id)) {
    Serial.print("release tcp ");
    Serial.print(mux_id);
    Serial.println(" ok");
  } else {
    Serial.print("release tcp ");
    Serial.print(mux_id);
    Serial.println(" err");
  }
 mux_id++;
 if(mux_id>4)
 mux_id=0;
time=millis();
sensvalue=0;
}


}
```