МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

# ИНТЕГРАЦИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ В КЛИЕНТ-СЕРВЕРНУЮ ТЕХНОЛОГИЮ, НА ПРИМЕРЕ ARDUINO

## АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 02.04.03 — Математическое обеспечение и администрирование
информационных систем
факультета КНиИТ
Черенкова Дмитрия Игоревича

Научный руководитель
к. ф.-м. н., доцент _____ В. М. Соловьев

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

# ВВЕДЕНИЕ

С каждым годом человек все больше пытается облегчить свою жизнь, придумывая различные устройства или средства их автоматизации. Одно из средств автоматизации - это построение умных микропроцессорных устройств. Такие устройства могут автономно выполнять заранее запрограммируемые действия. Например, поддерживать определенную температуру в доме или же фиксировать телеметрию гоночного трека. Таких устройств бесконечно много, взять хотя бы, мелкую или же крупную бытовую технику. Все вышеперечисленное требует близости нахождения потребителя. Ни один чайник не начнет нагреваться, пока мы его не включим. Управлять различным оборудованием удобнее на расстоянии, в этом нам поможет клиент-серверная технология. С её помощью по сети интернет можно передавать команды на исполняемое устройство и получать какие-либо ответы на эти команды. Совместив простые устройства с домашней сетью или же сетью интернет можно отслеживать состояние этих устройств. Управление ими осуществляется даже с мобильного телефона.

Целью магистрской работы является: изучение теоритических основ автоматизированных систем, построение архитектуры будущей системы и реализация шаблона много модульной микропроцессорной системы.

# 1 Архитектура клиент-сервер

Архитектура клиент-сервер – это сетевая архитектура, в которой взаимодействуют устройства называемые клиентами и серверами. Данная архитектура может использоваться как для физических устройств, так и для программного обеспечения зависимо от распределения логических компонентов приложения между клиентом и сервером.

В архитектуру клиент-сервер входят следующие основные компоненты:

- сервер баз данных отвечает за хранение, доступ, защиту и резервное копирование данных;
- сервер приложений - это устройство, выполняющее определенные бизнес-правила;
- клиент предоставляет интерфейс пользователя;
- сеть и коммуникационное ПО – это всевозможное оборудование, каналы для передачи данных и ПО используемое для осуществления передачи запросов и ответов от клиента к серверу и обратно через сетевые протоколы.

Характеристики архитектуры клиент-сервер:

- клиентские и серверные машины нуждаются в различном количестве аппаратных и программных ресурсов.
- клиентские и серверные машины могут принадлежать разным поставщикам.
- горизонтальная масштабируемость (увеличение числа клиентских компьютеров) и вертикальная масштабируемость (миграция на более мощный сервер или на многосерверное решение)
- клиентское или серверное приложение взаимодействует непосредственно с протоколом транспортного уровня для установления связи и передачи или приема информации.
- затем транспортный протокол использует протоколы более низкого уровня для отправки или приема отдельных сообщений. Таким образом, компьютеру необходим полный стек протоколов для запуска либо клиента, либо сервера.
- один компьютер серверного класса может предлагать несколько услуг одновременно;
- для каждой службы требуется отдельная серверная программа.

## 2 REST или что такое RESTful

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами.

Как и любой другой архитектурный стиль, REST также имеет собственные 6 направляющих ограничений, которые должны быть удовлетворены, если интерфейс должен называться RESTful.

REST API должен вводиться без предварительного знания, выходящего за пределы первоначального URI (закладки) и набора стандартизированных типов носителей, подходящих для предполагаемой аудитории (т.е., как ожидается, понятных любому клиенту, который может использовать API). С этого момента все переходы состояний приложения должны управляться выбором клиентом предоставляемых сервером вариантов, которые присутствуют в принятых представлениях или подразумеваются манипулированием пользователем этими представлениями. Переходы могут быть определены (или ограничены) знаниями клиента о типах медиа и механизмах связи с ресурсами, оба из которых могут быть улучшены "на лету"(например, код по запросу). Отказ подразумевает, что внеполосная информация является движущей силой взаимодействия вместо гипертекста.

Так как этот принцип основан на взаимодействии через HTTP протокол, при отправке ответа с результатом важно указать код ответа. Коды стандартные HTTP протокола, помогают более обширно следить за состоянием ресурса, за доступом к ресурсу при их правильном использовании.

REST предоставляет набор архитектурных ограничений, которые при применении в целом подчеркивают масштабируемость взаимодействия компонентов, универсальность интерфейсов, независимое развертывание компонентов и промежуточных компонентов для уменьшения задержек взаимодействия, обеспечения безопасности и инкапсуляции старых систем.

Все эти принципы помогают RESTful-приложениям быть простыми, легкими и быстрыми.

## 3 Arduino совместимые контроллеры

Встроенная электроника находится в бесчисленном количестве устройств и приборов, в основе управления - микроконтроллеры. Микроконтроллер представляет собой компактную интегральную схему, которая управляет конкретной работой во встроенной системе. Было время, когда для проектирования, прототипирования и тестирования встраиваемых систем были доступны только микроконтроллеры марок BASIC с простым интерпретатором.

Arduino - компания с открытым исходным кодом, которая разрабатывает и производит одноплатные микроконтроллеры и комплекты микроконтроллеров. Продукция лицензирована под GPL или LGPL. Это проект с открытым исходным кодом, который получает взносы от мирового сообщества пользователей. Компания предоставляет свои платы микроконтроллеров в предварительно собранном виде и в виде комплектов.

ESP8266 - это система на чипе (SoC), производимая китайской компанией Espressif. Состоит из 32-разрядного микроконтроллера Tensilica L106 (MCU) и приемопередатчика Wi-Fi. Она имеет 11 контактов GPIO * (контактов ввода/вывода общего назначения), а также аналоговый вход. Это значит, что программировать его можно, как любой обычный Arduino или другой микроконтроллер. А кроме того, вы получаете связь Wi-Fi, поэтому можете использовать ее для подключения к сети Wi-Fi, подключения к Интернету, размещения веб-сервера с реальными веб-страницами, пусть смартфон подключится к ней и т.д.

WeMos - китайская компания, которая производит Arduino-совместимые микроконтроллеры. Из основных плюсов - размер и набор возможностей этих плат.

Микроконтроллер D1 - намного мощнее Arduino Uno В D1 WeMos используется микроконтроллер ESP 8266, который в 2 раз быстрее Uno, имеет 160Kbs Ram по сравнению с 2K Uno и в 100 раз больше флэш-памяти.

Платы WeMos, позволяют управлять различными модулями вместо Arduino, но в отличии от большинства плат Arduino, у плат WeMos больший объем памяти программ и памяти ОЗУ, она построена на базе 32 разрядного микроконтроллера с большей тактовой частотой и оснащена встроенным WiFi модулем, который можно настроить как клиент (STA), точка доступа (AP), или клиент+точка доступа (STA+AP). [11]

# 4 Практическая реализация многомодульной системы

## 4.1 Архитектура

При выборе архитектуры было решено реализовать модель взаимодействия ведущий-ведомый.

«Ведущий — ведомый» — модель взаимодействия в вычислительных комплексах, телекоммуникационных и информационных системах, в которой одно главное устройство (ведущее устройство) или процесс осуществляет однонаправленное управление подчинённым (ведомым) устройством или процессом или их группой. В некоторых системах ведущий (мастер) выбирается из группы подходящих для этого устройств, остальные устройства в группе продолжают работать в режиме ведомых. Например, при репликации баз данных ведущий узел базы данных (мастер-реплика) устанавливается в качестве авторитетного источника данных, а дополнительные, ведомые реплики синхронизируются с ним.

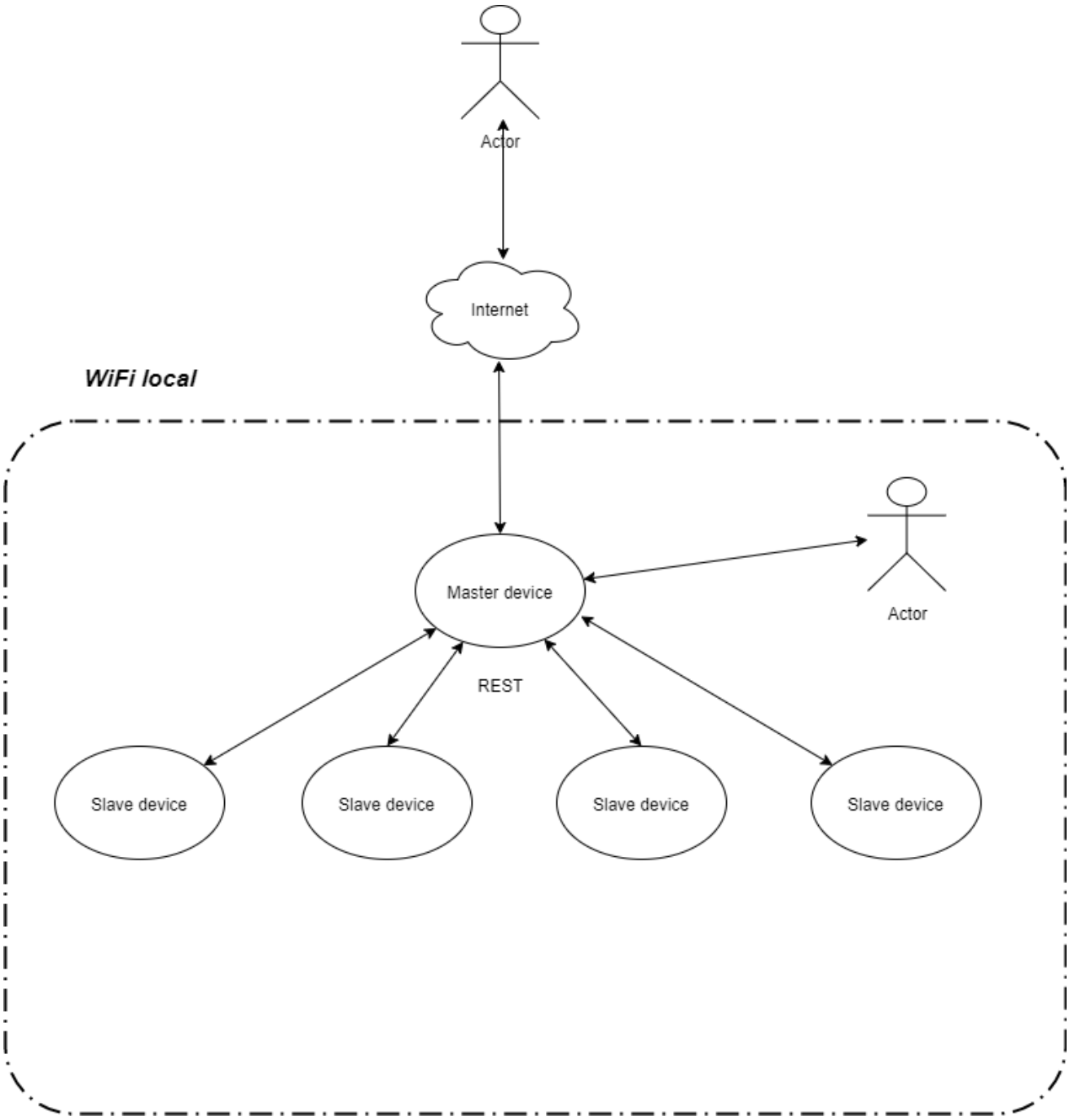


Рисунок 4.1 – Архитектура

### 4.2 Ведущий модуль

В роли ведущего модуля (master device) используем микроконтроллер Wemos. Все коммуникации между модулей реализованы по сети WiFi. Всвязи с этим было обработано два случая:

1. Есть сеть WiFi и мы можем к ней подключится. При программировании микроконтроллера достаточно правильно указать параметры подключения к сети, а именно название сети и пароль.
2. Либо есть сеть WiFi и ее нельзя использовать или же данной сети нет. Микроконтроллер сам может огранизовать сеть WiFi. Так же для настройки необходимо задать в коде программы желаемое название сети, ее пароль и установить в переменну, что требуется создать сеть.

Так как система многомодульная, оставался вопрос, как сделать поиск устройст в сети. Было реализовано два варианта:

— По текщей сети с помощью бибилиотеки Ping;
— Поиск устройств через специальный end point.

Для удобства работы был реализован Web-интерфейс. Одностраничный пользовательский интерфейс, в котором мы видим все подключеные устройсва к сети и их текущение состояние. Каждые 10минут автоматически этот список обновляется, на случай подлючения нового устройства - оно сразу будет готово к удаленному управлению. Наиболее часто используемое действие это включаение и выключание устройства. Для управления используются слайдеры. Так же по ним можно отслеживать состояние. При возникновении каких либо ошибок у система выводит текст ошибки на экран. Возможные типы ошибок:

— Устройство отключили, а список еще не успел обновится, при попытке изменить состояние устройства - будет выведена ошибка о том, что данное устройство не найдено в сети.
— При неполадке на самом устройстве так же выводится ошибка о том, что с устройством есть связь, но оно технически не исправно.

### 4.3 Ведомые модули

В качестве ведомых модулей (slave device) так же используем Wemos.

Как говорилось ранее Wemos обладает свойством обновления по воздуху (over-the-air OTA). Обновление OTA (Over the Air) - это процесс загрузки

нового микропрограммного обеспечения в ESP8266 модуль с использованием WiFi-соединения, а не последовательного соединения. Этот тип функций чрезвычайно полезен в случае отсутствия физического доступа к модулю ESP.

Загрузка нового эскиза по беспроводной сети из Arduino IDE предназначена для следующих типичных сценариев:

— Во время разработки микропрограммного обеспечения - в качестве более быстрой альтернативы загрузке нового эскиза по последовательному
— Для обновления микропрограммного обеспечения нескольких ESP в сети

Как раз второй случай идеально подходит под обновление такой системы, не важно где у тебя будет микроконтроллер, его можно будет обновить, модернизировать, добавив новый функционал.

В качесве базового примера slave модуля был реализован шаблон On/Off потому, что на его основе можно сделать любой часто использующийся функционал.

У микроконтроллера есть две точки доступа REST. Одна реализует смену действия, в зависимости от входных параметров POST запроса. К данному микрокотроллеру можно полключить реле, например для управления бытовыми проборами дома. Другая отвечат за получение метаинформации о устройстве, используемой при автоматическом поиске утройств на master контроллере.

Так же есть возможность указания врени задержки включения/выключения устройста, в интерфейсе это представленно числовым полем. Функционал с таймером реализован в модуле устройства, на которое направлен таймер.

Систему можно назвать отказоустойчивой, потому что она модульная, при отказе ведомых модулей не нарушается целостность системы.

# ЗАКЛЮЧЕНИЕ

При написании данной работы была изучена соответсвующая литература, с помощью которой удалось выполнить поставленную задачу.

Была разработана архитектура системы и программы для микроконтроллеров. Был реализован Web-интерфейс для ведущего модуля, реализован шаблон ведущего и ведомого модуля, реализован функционал автоматического опрделения ведомых модулей в сети, а так же простая аутентификация. В качестве соединения была выбрана сеть Wi-Fi, как более современная и часто используемая. К ней может подлючиться любое нынешнее устройсво: от часов до промышленного вычислительного коплекса.

Цель магистреской работы была достигнута в полном обьеме.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1 Дэвид Васкевич. Стратегии клиент/сервер. Диалектика, Киев. 1996 .

2 Макконнелл С. Совершенный код. 2017 .

3 Клиент-сервер шаг — за — шагом. [Электронный ресурс]: (на 18 мая 2020 года) URL: https://habr.com/post/330676/

4 Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API [Электронный ресурс]: (на 18 мая 2020 года) URL: https://tproger.ru/articles/web-api/

5 Рой Филдинг. Диссертация [Электронный ресурс]: (на 18 мая 2020 года) URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

6 Jennifer Williams. Practical Internet Of Things using Wemos D1 and it's shields: IoT Real Time example with Esp8266 Microcontroller. 2017.

7 Антошина И.В., Котов Ю.Т. Микропроцессоры и микропроцессорные системы. Москва. 2005.

8 Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino. БХВ-Петербург. 2012.

9 Петин В. Проекты с использованием контроллера Arduino. БХВ-Петербург. 2015.

10 Делаем сенсоры. Проекты сенсорных устройств на базе Arduino и Raspberry Pi. Вильямс. 2015.

11 Manoj Thakur. Zero to Hero ESP8266. 2016.

12 Джереми Блум. Изучаем Arduino. Инструменты и методы технического волшебства. БХВ-Петербург. 2018.

13 Виктор Петин. Arduino и Raspberry Pi в проектах Internet of Things. БХВ-Петербург. 2016.

14 Карл Вигерс, Джой Битти. Разработка требований к программному обеспечению. БХВ-Петербург. 2016.

15 Перри Ли. Архитектура интернета вещей. ДМК Пресс. 2018.

16 Мартин Роберт. Чистая архитектура. Искусство разработки программного обеспечения. Питер. 2018.

17 Обзор способов и протоколов аутентификации в веб-приложениях [Электронный ресурс]: (на 18 мая 2020 года) URL: https://habr.com/ru/company/dataart/blog/262817/

18 Ричард Э. Смит. Аутентификация: от паролей до открытых ключей. Вильямс. 2002.

19 М. Н. Молдабаева. Автоматизация технологических процессов и производств. Инфра-Инженерия. 2019.

20 А. М. Водовозов. Элементы систем автоматики. Академия. 2006.