СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сейчас невозможно представить жизнь без Интернета и сервисов, которые по нему предоставляются пользователям: Spotify, YouTube, образовательные платформы и социальные сети. Все эти сервисы работают на каких-то мощных компьютерах, которые подключаются к глобальной сети и выдают туда свой функционал. Это возможно благодаря компьютерным сетям.

Компьютерная сеть (Computer Network) – это система компьютеров, связанных каналами передачи информации; программно-аппаратный комплекс, обеспечивающий автоматизированный обмен данными между компьютерами по каналам связи. Компьютеры и сервера соединяются с коммутаторами и маршрутизаторами посредством разнообразных линий связи, используя технологию Ethernet на канальном уровне, в своем большинстве.

Ethernet – это традиционная технология, используемая для подключения устройств в проводной локальной сети (LAN) или глобальной сети (WAN), позволяющая им обмениваться данными друг с другом через протокол (набор правил или общий сетевой язык). Ethernet описывает, как сетевые устройства форматируют и передают данные, чтобы другие устройства в той же локальной сети или сегменте сети могли идентифицировать, получать и обрабатывать информацию. Кабель Ethernet – это физическая замкнутая проводка, по которой проходят данные. Он популярен, потому что он обеспечивает хороший баланс между скоростью, стоимостью и простотой установки. Эти преимущества в сочетании с широким распространением на компьютерном рынке и возможностью поддержки практически всех популярных сетевых протоколов делают Ethernet идеальной сетевой технологией для большинства пользователей компьютеров сегодня.

По началу развития технологии Ethernet сети были относительно маленького размера, и в них использовалась топология сети общая шина. Среды передачи раньше строились на коаксиальных медных кабелях и коннекторах, поэтому выявление дефектов занимало много времени. Неисправности в канале такой сети приводили к неработоспособности всех коммуникаций между хостами и длительным работам по нахождению дефектов и обслуживанию.

С развитием технологии Ethernet пришла эра коммутируемой и маршрутизируемой сети на одноименных устройствах, а популярной и

активно используемой сетевой моделью стал стек TCP/IP, в основе канального уровня остался Ethernet. Линии связи стали связывать хостов и коммутаторы независимо от других абонентов через дуплексные каналы, а маршрутизаторы логически разбили локальные сети на различные сетевые сегменты, тем самым устранив домен коллизий и упростив задачу нахождения неисправностей в локальной сети для сисадминов. Теперь диагностика работоспособности, конфигурация и мониторинг сети занимают гораздо меньше времени и усилий. Для этого существует множество средств как программных (для мониторинга сети: WireShark и аналоги), так и реализованных в стеке TCP/IP (ICMP, SNMP, NTP, etc.).

Цель данного курсового проекта – это автономное устройство, которое «слушает» LAN-сеть, собирая информацию по возможно доступным хостам (IP адреса), использует возможности современных протоколов сетевого уровня для выявления их доступности (ICMP), замера метрик и вывода в удобочитаемом формате этих данных пользователю. Исходя из этого, выделим следующие функции данного устройства:

- Управление через пользовательский WebUI.
- Отображение в WebUI информации о составе сети, ее существующих хостах.
- Прослушивание сети, сбор информации о хостах из циркулирующих в сети пакетов.
- Обмен полученной информацией между микроконтроллерами.
- Функционал утилиты ping с выводом результата в пользовательский WebUI.

1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

1.1 Структура и архитектура вычислительной системы

В качестве устройства, с помощью которого реализовывался данный курсовой проект, мы выбрали плату Arduino Nano (рисунок 1).

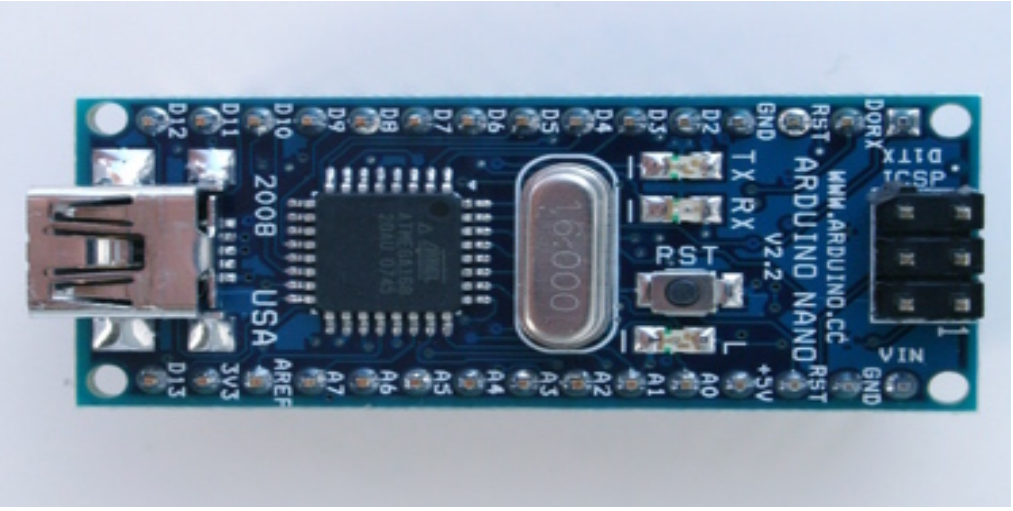

Рисунок 1 – Arduino Uno Nano [4]

Arduino Nano – это отладочная плата небольшого размера, которая входит в список лидеров по популярности среди программистов. Но несмотря на миниатюрный размер, она практически ничем не уступает Arduino Uno по функционалу и может использоваться в проектах, где габариты играют существенную роль.

Внимательно рассмотрев плату Arduino Nano можно заметить, что это полнофункциональное устройство на базе микроконтроллера ATmega328 (Arduino Nano 3.0), который адаптирован для использования с макетными платами [3].

Для экономии места разработчики расположили радиоэлементы по обе стороны платы. С лицевой стороны нанесена вся информативная шелкография, установлен микроконтроллер ATmega328, кварцевый резонатор, разъём Mini-USB, кнопка сброса и четыре индикаторных светодиода (TX, RX, PWR и L). Обратная сторона послужила основой для линейного стабилизатора напряжения 5V и преобразователя интерфейса FTDI USB. На рисунке 2 приведены обе стороны платы Arduino Uno Nano [3].

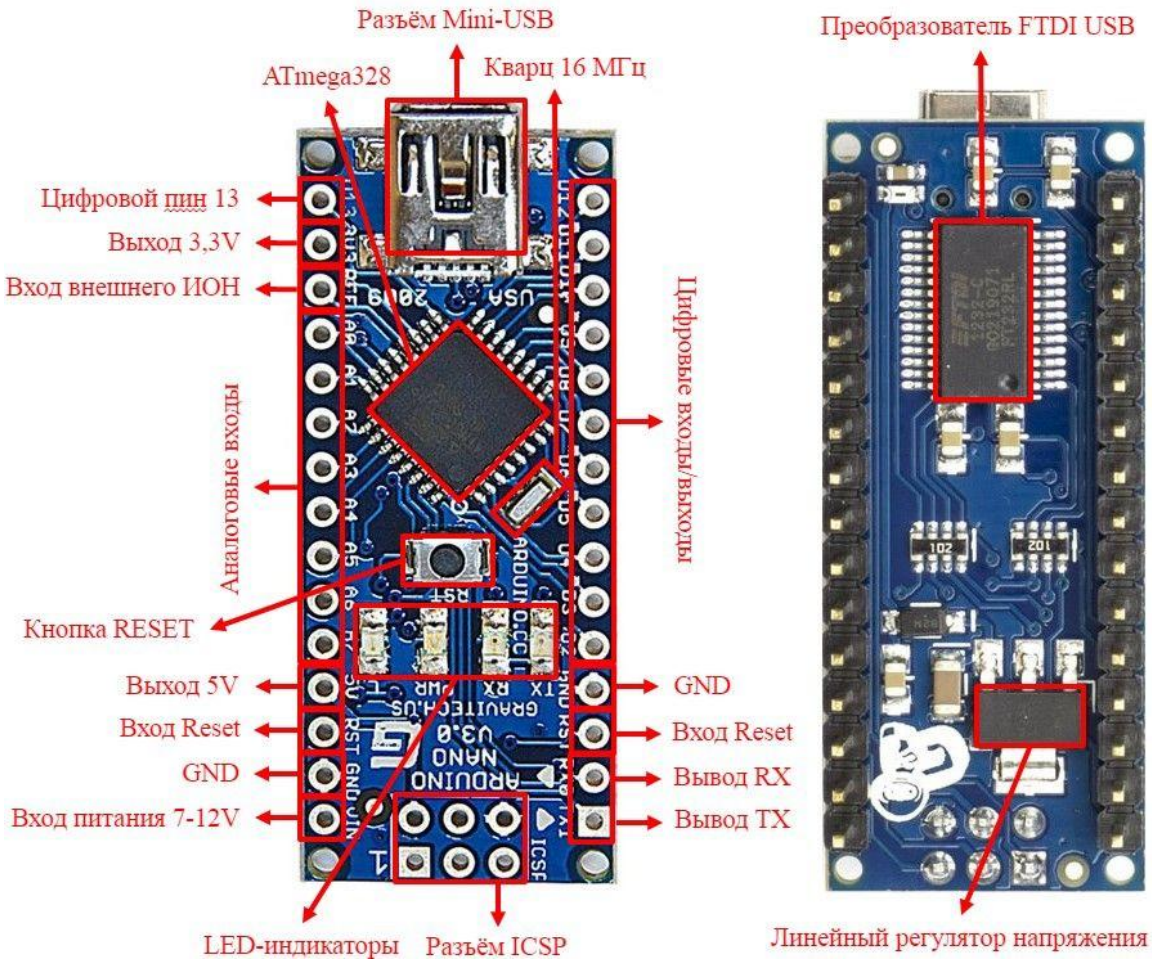

Рисунок 2 – Состав платы Arduino Uno Nano [4]

Для использования платы Arduino Nano, ее необходимо предварительно запитать одним из двух способов [4]:

– Через кабель Mini-B USB при подключении к компьютеру или другому источнику питания 5V.

– При помощи внешнего стабилизированного источника питания, напряжение которого должно лежать в диапазоне 6-20V (рекомендуется 7-12V). Данное напряжение подаётся непосредственно на вход VIN платы Arduino Nano.

Если одновременно подключить два источника питания, то плата выберет тот, потенциал которого будет выше. Независимо от способа подключения, вывод GND платы Arduino Nano является общим минусом [4].

Однако, напряжение на микросхему FTDI FT232RL подается только в случае питания Arduino Nano через USB. Поэтому при питании устройства от других внешних источников (не USB), выход 3.3В (формируемый

микросхемой FTDI) будет неактивен, в результате чего светодиоды RX и TX могут мерцать при наличии высокого уровня сигнала на выводах 0 и 1 [4].

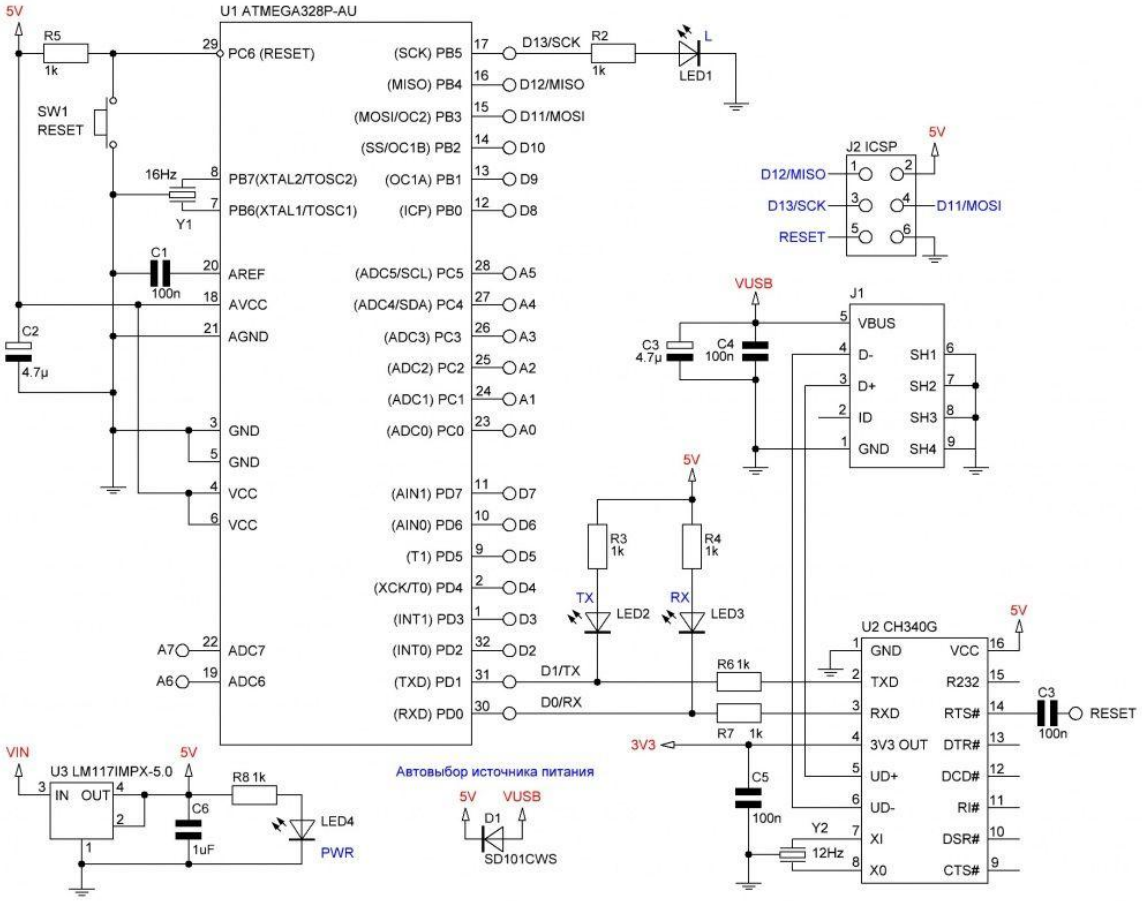

Рисунок 3 – Электрическая схема платы Arduino Uno Nano [4]

В данной плате предусмотрены 8 аналоговых входов, каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 различных значения). По умолчанию, измерение напряжения осуществляется относительно диапазона от 0 до 5 В. Тем не менее, верхнюю границу этого диапазона можно изменить, используя вывод AREF и функцию analogReference(). Но стоит знать, что любое превышение, несмотря на встроенные ограничительные диоды, может вывести микроконтроллер из строя. То же самое касается и нагрузочной способности порта. Максимальный ток, который может выдать один вывод, равен 40 мА, а суммарный ток всех выводов не должен превышать значение 200 мА. Для наглядности на рисунке 3 приведена электрическая схема платы Arduino Nano [4].

Память Arduino Nano можно функционально разделить на три разных

группы. Такая классификация обусловлена применением микроконтроллера линейки AVR, в котором подобная топология заложена производителем. Каждый подвид памяти выполняет свою функцию, что в комплексе обеспечивает полноценную жизнедеятельность микроконтроллера. Итак, Arduino Nano предоставляет следующие виды памяти [4]:

– Flash-память – служит для хранения программного кода в виде прошивки. Иногда используется также для хранения каких-либо константных данных с целью экономии места в оперативной памяти. Как пример – текстовые строки для меню и.т.п. Если Arduino Nano выполнена на базе микроконтроллера ATmega328, то объем flash-памяти будет равен 32 кБ. Следует учитывать, что 2 кБ будет занято программой-загрузчиком.

– EEPROM-память – это энергонезависимая память, значения которой не изменяются даже при отключении питания. Широко используется для сохранения последних настроек в проекте, которые по задумке подвержены периодическому отключению от источника электроэнергии. Объем этой памяти зависит от типа используемого микроконтроллера. В данном случае ATmega328 представляет 1 кБ EEPROM. В Arduino IDE есть специализированная библиотека, позволяющая удобно работать с энергонезависимой памятью. Подключить её можно следующей строкой: *#include<EEPROM.h>*

– SRAM-память – оперативная память, которая служит для хранения значений переменных в период выполнения программного кода и полностью обнуляется при отключении Arduino Nano от источника питания. Микроконтроллер ATmega328 имеет SRAM в объеме 2 кБ.

Arduino Nano предоставляет ряд возможностей для осуществления связи с компьютером, еще одним Arduino или другими микроконтроллерами. В ATmega168 и ATmega328 есть приемопередатчик UART, позволяющий осуществлять связь по последовательным интерфейсам посредством цифровых выводов 0 (RX) и 1 (TX). Микросхема FTDI FT232RL обеспечивает связь приемопередатчика с USB-портом компьютера, и при подключении к ПК позволяет Ардуино определяться как виртуальный COM-порт [4].

1.2 История, версии и достоинства

Выпущенный в 2005 году как скромный инструмент для студентов от Массимо Банзи при взаимодействии проектного института Иврея (IDII – Interaction Design Institute Ivrea), Arduino породил международную

революцию в электронике. Купить плату Arduino можно всего за невысокую стоимость или построить свою собственную с нуля: все аппаратные схемы и исходный код доступны бесплатно в рамках открытых лицензий. В результате, Ардуино стал самым выдающимся достижением своего времени в области открытого аппаратного обеспечения [12].

Маленькая плата теперь стала доступной для мастеров, любителей, студентов и тех, кто просто мечтает создать что-то свое. Более 250 000 плат Arduino были проданы по всему миру, и это не включает в себя множество клонов. «Он сделал возможным для людей, создать то, что они не могли создать ранее,» говорит Дэвид Меллис, который был студентом в IDII до того, как продолжил заниматься дипломом на MIT Media Lab (междисциплинарная исследовательская лаборатория в Массачусетском Технологическом Институте) и сегодня является ведущим разработчиком программного обеспечения Ардуино [12].

Ардуино возник в связи с задачей: как научить студентов создавать электронные проекты быстро. Это был 2002 год, и Банзи, архитектор программного обеспечения, был вызван IDII доцентом для поиска решения проблемы. Но, в связи с сокращением бюджета и ограниченного времени на использование помещений, его возможностей было мало. Как и многие его коллеги, Банзи опирался на BASIC Stamp, микроконтроллер, созданный в Калифорнии компанией Parallax, инженеры которой использовали его около десяти лет. Stamp был аккуратно расположен на маленькой плате, на ней также располагались блок питания, память, и порты ввода/вывода для подключения оборудования. Но у BASIC Stamp было две проблемы, Банзи обнаружил, что он не имеет достаточной вычислительной мощности для некоторых проектов его учеников, а также он был слишком дорогим, доска плюс основные части могли достигать достаточно высокой стоимости. Он также подыскивал что-то, что может работать на компьютерах Macintosh, которые были повсеместно среди дизайнеров IDII. Он решил, что может создать плату сам, которая подходила бы под его потребности [12].

У Банзи был коллега из Массачусетского технологического института, который разработал язык программирования, названный Processing. Processing стремительно набирал популярность, потому что это позволило даже неопытным программистам создавать сложные и красивые визуализации. Одна из причин его успеха была очень простая в использовании интегрированная среда разработки, другими словами IDE. Банзи поинтересовался, могут ли они создать подобные программные средства для программирования микроконтроллеров вместо графики на

экране. Студент Эрнандо Барраган, сделал первые шаги в этом направлении. Он разработал прототип платформу под названием Wiring (проводка), которая включала как IDE так и плату, готовую к использованию. Это был многообещающий проект, который продолжается и по сей день, но Банзи уже думает дальше: он хотел сделать платформу, которая была еще проще, дешевле и удобней в использовании [12].

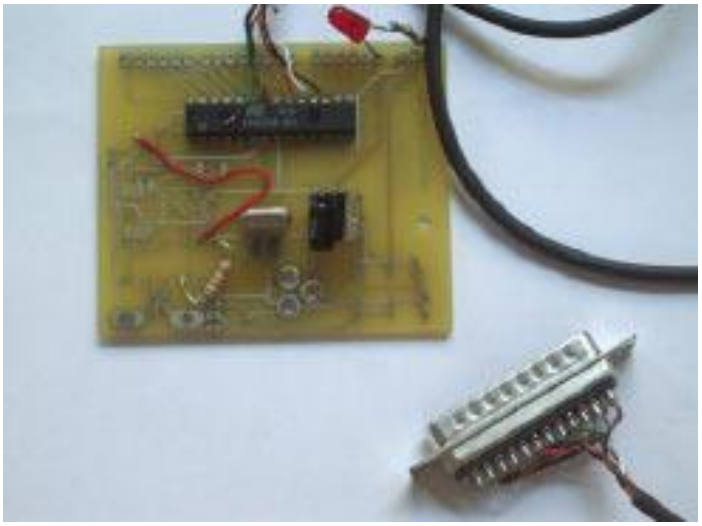

Рисунок 4 – Первая плата прототипа [11]

Первая плата – прототип, сделанный в 2005 году, имела простой дизайн, и она еще не называлась Arduino (рисунок 4). Массимо Банзи придумал имя год спустя.

Банзи и его сотрудники были сторонниками открытого кода в программном обеспечении. Поскольку цель в том, чтобы создать быструю и легкодоступную платформу, они чувствовали, что было бы лучше сделать проект открытым, насколько это возможно, а не держать его закрытым. Модель с открытым исходным кодом уже давно используется для стимулирования инноваций в программном обеспечении, но не аппаратного обеспечения. Чтобы она работала, они должны были найти соответствующее решение лицензирования, которое можно применить к их плате. После некоторого исследования, они поняли, что можно по-разному смотреть на проект. Они могли бы использовать лицензию Creative Commons, некоммерческой группы, чьи соглашения, как правило, используются для произведений культуры, таких как музыка и текст [12].

Команда создала продукт, состоящий из дешевых деталей, которые могут быть легко найдены, если люди захотели бы построить свои

собственные платы. Микроконтроллер они взяли ATmega328. Но главной целью было убедиться, что он будет, по сути, «включи и работай», что можно просто взять из коробки, подключить к компьютеру и сразу использовать. Такие платы, как BASIC Stamp требуют, чтобы разработчики раскошеливались на полдюжины других деталей, которые добавляются к общей стоимости. Но для них пользователь должен просто подключить кабель USB от платы к компьютеру – программирующему устройству [12].

Успех Arduino многим обязан существованию языка программирования Processing и платформы Wiring. Эти проекты дали Arduino один из его существенных преимуществ: очень удобную среду программирования. До Arduino программирование микроконтроллера несло определенные трудности в процессе обучения. С Arduino, даже те, кто не имел опыта в электронике, получили доступ к ранее непонятному миру электроники. Теперь, новички не должны много изучать, прежде чем они смогут построить рабочий прототип электронного устройства. Это большое продвижение, в то время, когда многие из самых популярных гаджетов работают по принципу «черных ящиков», которые закрыты и защищены патентами [12].

Команда старается идти в ногу со спросом на платы. Платформа Arduino больше не состоит из одного типа платы, теперь существует целое семейство плат. В дополнение к оригинальному проекту, названному Arduino Uno (рисунок 5), новые модели включают более мощную плату, названную Arduino Mega, компактную плату, с названием Arduino Nano (рисунок 1), водонепроницаемую плату LilyPad Arduino (рисунок 6), и недавно выпустила плату для подключения к сети, Arduino Ethernet.

Рисунок 5 – Плата Arduino Uno [11]

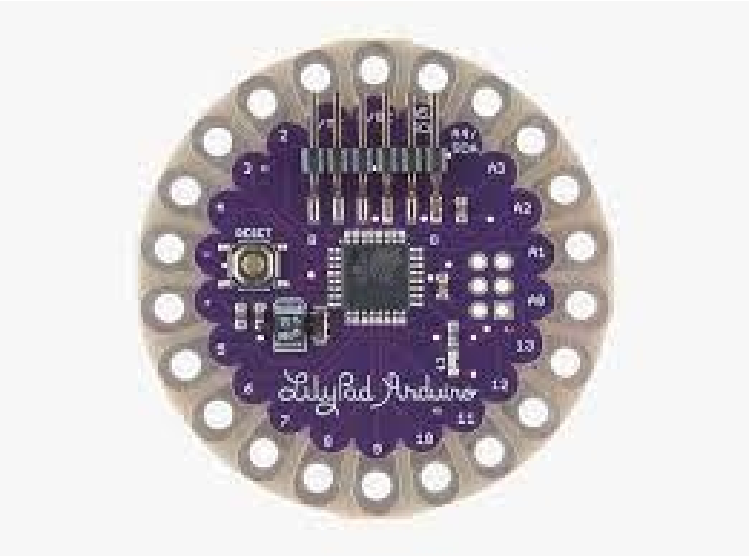

Рисунок 6 – Плата Arduino LilyPad [11]

1.3 Обоснование выбора платформы. Анализ выбранной вычислительной системы

Контроллеры Arduino Nano активно используются в самых разнообразных проектах. Использование миниатюрного контроллера позволяет создавать устройства в небольшом форм-факторе, что является важным для проектов в области автоматизации. Эта плата довольно компактная, удобная и обладает множеством преимуществ [13]:

– Кроссплатформенность. Программное обеспечение Arduino осуществляет работу на большинстве известных программ Windows, Macintosh OS X, Linux, являясь открытым приложением работающим на Java.

– Цена. Arduino Nano возможно купить менее чем за 30$.

– Простая среда программирования. Программная оболочка является достаточно простой в применении для новичков, но весьма гибкой для большинства продвинутых пользователей, чтобы оптимально быстро достичь нужного результата.

– Открытый исходный код. Язык может расширяется с помощью C++ библиотек, значительно более продвинутых, где специалисты могут самостоятельно создать свой собственный эксклюзивный инструментарий для Arduino на основе инновационного компилятора AVR C.

– Открытые спецификации и схемы оборудования. Arduino основан на микроконтроллерах ATmega168 и ATmega328. Схемы модулей публикуются под лицензией Creative Commons, из-за этого опытные

схемотехники могли создавать свои собственные версии модуля. Даже весьма неопытные пользователи смогут делать макетную версию данного модуля, чтобы понимать, каким же образом он осуществляет работу и экономит деньги.

Конечно же, присутствуют и минусы [13]:

– ПО. Не нативная программная оболочка, что может влечь за собой определенный трудночти.

– Частота. Достаточно низкая частота имеющегося процессора.

– Память. Малое количество «дисковой» флэш-памяти для создания программ.

Однако, для данного проекта выбранная плата, Arduino Nano, является если не лучшим, то точно одним из лучших вариантов, поскольку она не требует больших денежных затрат, доступна, имеет миниатюрный размер, представляет собой полнофункциональное законченное устройство, является универсальным и достаточно мощным инструментом при правильном использовании.

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для работы с Arduino Nano необходим компьютер, USB-data кабель и подходящая среда разработки.

ATmega328 в Arduino Nano выпускается с прошитым загрузчиком, позволяющим загружать в микроконтроллер новые программы без необходимости использования внешнего программатора. Взаимодействие с ним осуществляется по оригинальному протоколу STK500. Тем не менее, микроконтроллер можно прошить и через разъем для внутрисхемного программирования ICSP (In-Circuit Serial Programming), не обращая внимания на загрузчик [15].

Создание проекта на Arduino состоит из 3 главных этапов: написание кода, прототипирование (макетирование) и прошивка. Для того, чтоб написать код, а затем прошить плату, необходима среда разработки. Их есть немало, но для данного проекта использовалась оригинальная среда – Arduino IDE [16].

2.1 Среда разработки Arduino IDE

Arduino IDE – это интегрированная среда разработки для Windows, MacOS и Linux, разработанная на C и C++, предназначенная для создания и загрузки программ на Arduino-совместимые платы, а также на платы других производителей [17].

Простая и функциональная среда разработки для создания создания собственного ПО, которым управляются многочисленные устройства, собранные начинающими и опытными электронщиками. Соединение ПК с микроконтроллером реализовано через интерфейс USB. Код на языке C и C++ пишется в редакторе, в котором есть подсветка команд и спеллчекер.

Рассмотрев подробнее среду разработки Arduino IDE, можно заметить, что она включает в себя не только редактор, но еще и компилятор, работающий в паре с загрузчиком.

Среда разработки состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста (консоли), панели инструментов с кнопками часто используемых команд и нескольких меню (рисунок 7). Для загрузки программ и связи среда разработки подключается к аппаратной части Arduino.

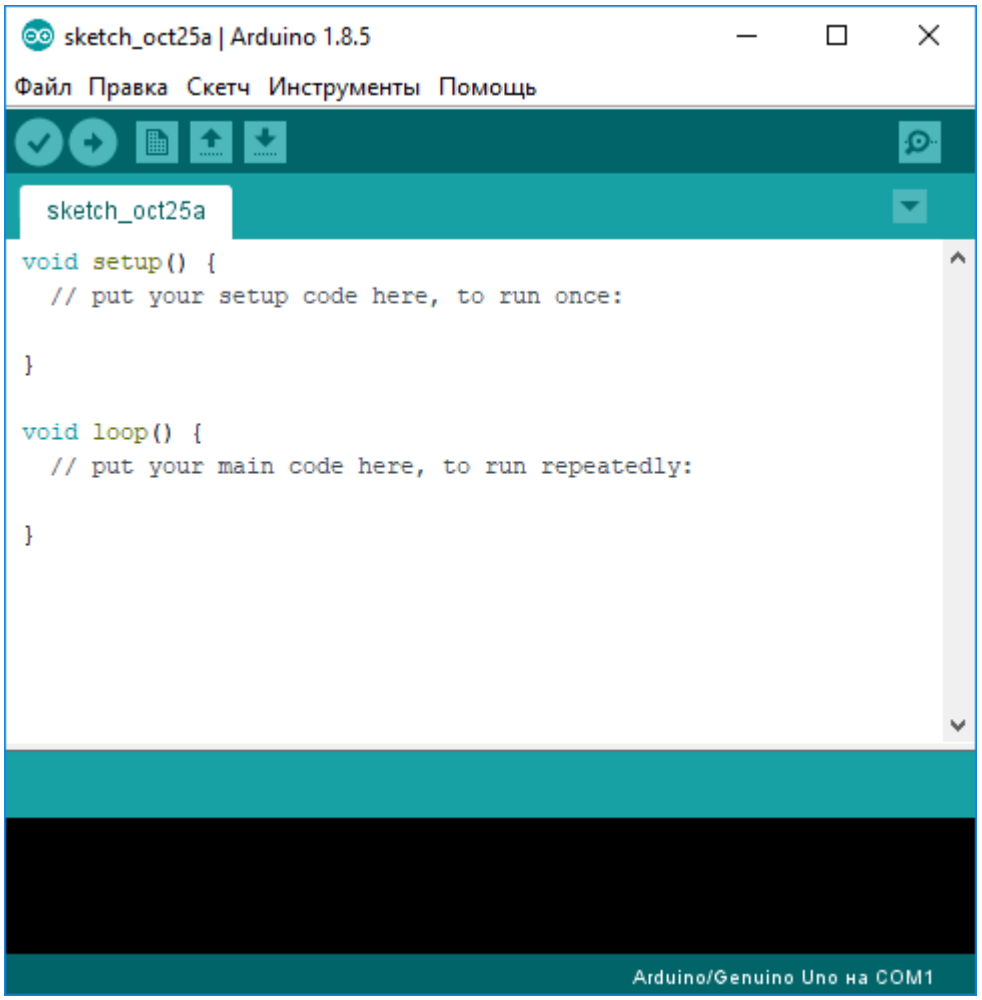

Рисунок 7 – Интерфейс среды разработки Arduino IDE

Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе, имеющем инструменты вырезки/вставки, поиска/замены текста. Во время сохранения и экспорта проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки. Окно вывода текста (консоль) показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины [17]:

– Verify/Compile: Проверка программного кода на ошибки, компиляция.

– Stop: Остановка мониторинга последовательной шины(Serial monitor) или затемнение других кнопок.

– New: Создание нового скетча.

– Open: Открытие меню доступа ко всем скетчам в блокноте. Открывается нажатием в текущем окне.

– Save: Сохранение скетча.

– Upload to I/O Board: Компилирует программный код и загружает его в устройство Arduino.

– Serial Monitor: Открытие мониторинга последовательной шины (Serial monitor).

Средой Arduino используется принцип блокнота: стандартное место для хранения программ (скетчей). Скетчи из блокнота открываются через меню File > Sketchbook (рисунок 8) или кнопкой Open на панели инструментов. При первом запуске программы Arduino автоматически создается директория для блокнота. Расположение блокнота меняется через диалоговое окно Preferences [17].

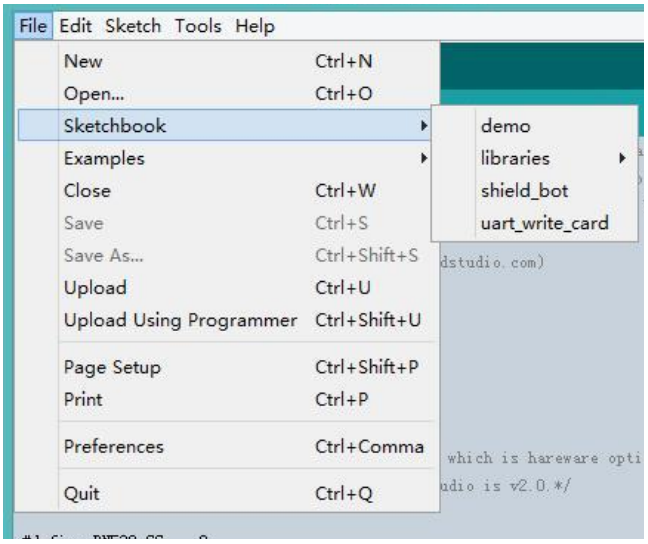

Рисунок 8 – Открытие скетча через меню Arduino IDE

Закладки, Файлы и Компиляция позволяют работать с несколькими файлами скетчей (каждый открывается в отдельной закладке). Файлы кода могут быть стандартными Arduino (без расширения), файлами C (расширение *.c), файлами C++ (*.cpp) или головными файлами (.h) [17].

Для того, чтобы загрузить скетч, необходимо задать необходимые параметры в меню Tools > Board и Tools > Port (рисунок 9). После выбора порта и платформы необходимо нажать кнопку загрузки на панели инструментов или выбрать пункт меню File > Upload to I/O Board. Современные платформы Arduino перезагружаются автоматически перед загрузкой. На старых платформах необходимо нажать кнопку перезагрузки.

На большинстве плат во время процесса будут мигать светодиоды RX и TX. Среда разработки Arduino выведет сообщение об окончании загрузки или об ошибках. При загрузке скетча используется Загрузчик (Bootloader) Arduino, небольшая программа, загружаемая в микроконтроллер на плате. Она позволяет загружать программный код без использования дополнительных аппаратных средств. Загрузчик (Bootloader) активен в течении нескольких секунд при перезагрузке платформы и при загрузке любого из скетчей в микроконтроллер. Работа Загрузчика (Bootloader) распознается по миганию светодиода (13 пин) (например, при перезагрузке платы) [17].

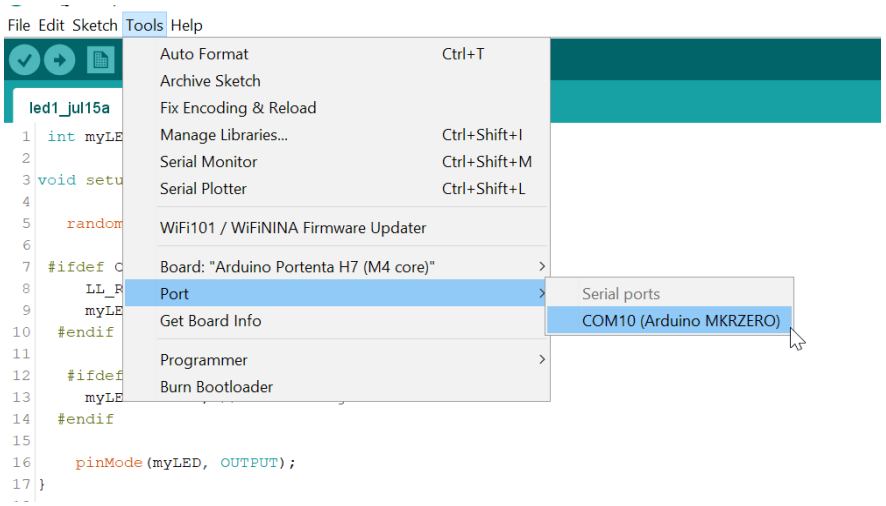

Рисунок 9 – Выбор порта в меню Arduino IDE

Также данная среда разработки позволяет использовать библиотеки, что дает дополнительную функциональность скетчам, например, при работе с аппаратной частью или при обработке данных. Для использования библиотеки необходимо выбрать меню Sketch > Import Library. Одна или несколько директив #include будут размещены в начале кода скетча с последующей компиляцией библиотек и вместе со скетчем. Загрузка библиотек требует дополнительного места в памяти Arduino. Неиспользуемые библиотеки можно удалить из скетча убрав директиву #include [17].

2.2 Анализ платформ Arduino IDE

Выбор платформы влияет на: параметры (например, скорость ЦП и скорость передачи данных), используемые при компиляции и загрузке

скетчей и на настройки записи загрузчика (Bootloader) микроконтроллера. Некоторые характеристики платформ различаются только по последнему параметру (загрузка Bootloader), таким образом, даже при удачной загрузке с соответствующим выбором может потребоваться проверка различия перед записью загрузчика (Bootloader) [17].

Arduino Nano с ATmega328.

Тактовая частота ATmega328 16 МГц с возможностью автоматической перезагрузки.

Arduino Nano с ATmega168.

Тактовая частота ATmega168 16 МГц с возможностью автоматической перезагрузки. Компиляция и загрузка соответствует старым версиям с ATmega168, но загрузка Bootloader имеет короткий таймаут (при перезагрузке светодиод пина 13 мигает один раз).

Arduino Mega.

Тактовая частота ATmega1280 16 МГц с возможностью автоматической перезагрузки.

Arduino Mini.

Соответствует Arduino NG или старым версиям с ATmega168 (напр.: тактовая частота ATmega168 16 МГц без возможности автоматической перезагрузки).

LilyPad Arduino с ATmega328.

Тактовая частота ATmega328 8 МГц (3.3 В) с возможность автоматической перезагрузки.

2.3 Обоснование выбора среды разработки.

Для разработки программной прошивки устройства используется среда Arduino IDE. Выбор данной программной среды в первую очередь обусловлен использованием платформы Arduino (версия Nano 3.0), для которой и разрабатывалась данная среда. Arduino IDE является свободно распространяемым ресурсом и имеет понятный и простой интерфейс.

Более того, данная среда разработки имеет немало преимуществ:

– Подсветка кода.

– Быстрая загрузка скетча в плату Arduino.

– Для того, чтобы обозначить определенный пин порта как вход/выход, достаточно написать функцию pinMode (Имя порта, OUTPUT/INPUT).

– Можно быстро присвоить номер порта к определённой переменной.

– Также можно достаточно быстро обозначить состояние порта (LOW или HIGH).

– Имеется множество полезных встроенных функций.

Имеются и недостатки, но количество преимуществ их перекрывает, поэтому, несмотря на их наличие, для разработки программного обеспечения была выбрана именно эта среда:

– Отсутствие CodeGuard, который помогает быстро записывать переменные, функции и т.д.

– Необходим определённый bootloader, который будет распознавать плату, подключенную через USB как COM-порт.

– Часть встроенных функций не всегда удобна в использовании, поскольку они используют некоторую аппаратную часть микроконтроллера.

– Отсутствие какого-либо виртуального симулятора.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Ethernet – технология локальных сетей, отвечающая за передачу данных по кабелю, доступную для устройств компьютерных и промышленных сетей. Данная технология очень полезна для пользователей и имеет множество преимуществ: она гибка, легко управляется, безопасна, стабильное подключение к интернету и отсутствие помех.

Очевидно, что Wi-Fi удобнее, чем проводные кабели Ethernet (рисунок 10), но Ethernet по-прежнему предлагает значительные преимущества. Вероятно, в ближайшее время никто не будет подключать к смартфону кабель Ethernet. Но обычно стоит проложить кабели Ethernet к важным устройствам, если есть такая возможность – игровым и мультимедийным ПК (или консолям), устройствам резервного копирования и телевизионным приставкам – лишь несколько примеров использования данной технологии в домашних условиях. Также Ethernet – это технология, с помощью которой информация передается между компьютерами, связанными в локальную сеть, поэтому используется не только для стабильного подключения к интернету. Данная технология используется в промышленности, офисах, сотовой связи, везде, где реализован обмен данными между машинами [19].

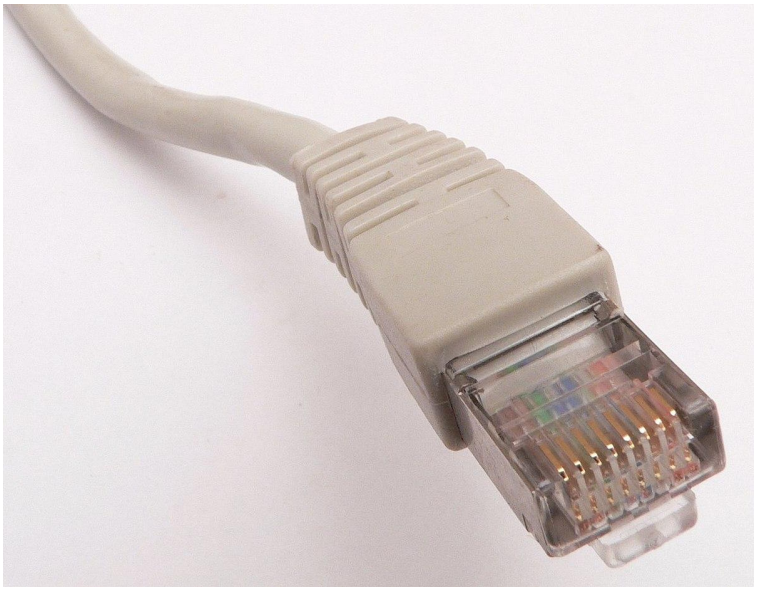

Рисунок 10 – Кабель Ethernet

Рассмотрев основные преимущества данной технологии, наверняка станет понятно, что Ethernet еще долгое время будет актуален [20]:

Безопасность.

Поскольку для проводного соединения требуются физические подключения, только те, у кого есть физический доступ к сети, могут получить из нее информацию. Шансы на то, что кто-то подключит свое оборудование и украдет информацию незаметно для всех, ничтожны, особенно при надлежащей безопасности.

Удобство использования.

Существуют кабели Ethernet разной длины и с поддержкой разной скорости. Таким образом, возможно выполнить соединение Ethernet практически на любое расстояние, которое нужно преодолеть.

Скорость подключения.

Ethernet относится к классу широкополосных (broadband) технологий. Он обеспечивает скорость передачи данных от 10 до 100 Мбит/с, причем симметрично – скорость не зависит от того, скачивается файл к себе на компьютер, или отправляете его. Обладатели различных DSL (ADSL, SDSL, VDSL и пр.) могут только мечтать о подобных скоростях. Высокая скорость позволяет комфортно работать с Web-сайтами, быстро перекачивать большие файлы и документы, работать с мультимедиа, полноценно использовать интерактивные приложения, поддерживать связь между несколькими филиалами так, как будто они расположены в соседних комнатах [21].

Однако, как и за любым другим способом подключения к сети, за кабельным подключением для избежания каких-либо угроз также необходимо следить: диагностировать работоспособность, выявлять дефекты, замерять метрики, прослушивать сети и т.д.

То есть данный курсовой проект может использоваться:

– Сетевыми администраторами для устранения неполадок сети.

– Инженерами по сетевой безопасности для проверки проблем безопасности.

– Для изучения внутренних сетевых протоколов.

Для данных целей уже существует набор анализаторов сетевых протоколов, которые позволяют захватывать и интерактивно просматривать трафик, работающий в компьютерной сети. Это де-факто стандарт во многих отраслях и учебных заведениях.

К примеру, рассмотрим несколько программ, выполняющих функции проверки сетевых соединений: WireShark, netcat, Intercepter-NG, tcpdump и т.д. Все они обладают необходимыми функциями проверки протоколов, их

записи и анализа, просмотра захваченных сетей, проверки доступных IP-сетевых адресов и многими другими.

Конечно, данный проект не имеет всего функционала данных программных обеспечений, но он все равно имеет ряд преимуществ:

– Компактность и мобильность устройства.

– Возможность использования без вспомогательных устройств, таких как персональный компьютер или ноутбук.

– Простота использования.

– Малая стоимость проекта, не требующая повторных затрат, таких как обновление подписки для продолжения использования.

– Возможность собственноручно изменять и расширять функционал в зависимости от ряда задач, для решений которых будет использоваться устройство.

4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

В рамках данного курсового проекта необходимо разработать автономное устройство проверки сетевых соединений. Были выбраны 2 микроконтроллера, которые выполняют 2 логически независимые задачи. Исходя из этого, были выделены следующие функции, которые должно выполнять устройство:

- Управление через пользовательский WebUI.
- Отображение в WebUI информации о составе сети, ее существующих хостах.
- Прослушивание сети, сбор информации о хостах из циркулирующих в сети пакетов.
- Обмен полученной информацией между микроконтроллерами.
- Функционал утилиты ping с выводом результата в пользовательский WebUI.

4.1 Определение компонентов структуры устройства

Компоненты структуры устройства выбираются исходя из выше описанных функций. Проанализировав выделенные функции, были определены следующие компоненты:

- Микроконтроллер – ключевой компонент всей схемы. Выполняет функцию обработки поступающей информации и выдает управляющие сигналы.
- Модуль питания – стабилизатор напряжения и источник питания схемы.
- Модуль управления – микроконтроллер, выступающий в роли веб-сервера.
- Ethernet-модуль – модуль, подключаемый к микроконтроллеру и выполняющий функции сетевого взаимодействия с подключаемой к нему сетью.

4.2 Взаимодействие компонентов устройства

Устройство, по старту своей работы, прослушивает сеть, получает данные о хостах и передает их модулю управления.

Модуль управления выводит информацию о хостах в веб-интерфейс пользователю и предлагает последнему пропинговать данных хостов. Хост вводит интересный ему IP-адрес устройство в поле ввода и нажимает «пинг». Далее модуль управления по интерфейсу UART передает данный IP микроконтроллеру.

Микроконтроллер, получив IP от модуля управления, начинает готовить ICMP пакет для необходимого хоста, отправляет и замеряет время до возвращения ответа от хоста.

Вытянув необходимые данные (или успешный echo reply, или какой-либо код ошибки) микроконтроллер передает результат модулю управления.

Модуль управления, получив ответ от микроконтроллера, выводит результат пользователю и ожидает следующих действий.

Модуль питания взаимодействует со всеми элементами схемы напрямую или через контроллер, благодаря ему осуществляется питание всех необходимых элемент.

4.3 Обоснование выбора микроконтроллера

В данной курсовом проекте в качестве контроллера могла быть использована любая плата из представленных в таблице 4.3.1, так как устройство не требует больших затрат в памяти и мощности для корректной работы. Контроллеры Raspberry PI 2 и OLIMEXINO-STM32 превосходят Arduino Nano по памяти и мощности, но требуют для работы больше входного напряжения и имеют большую стоимость. Также у Raspberry PI 2 нет аналоговых входов/выходов, которые могут пригодится при сборке проекта [22].

Таблица 4.3.1 – Сравнение микроконтроллеров

Параметры сравнения	Arduino NANO	Raspberry PI 2	OLIMEXINO STM32
Микроконтроллер	ATmega328	ARM Cortex-A7	stm32f103rbt6
Входное напряжение	6 – 20 В	6 – 28 В	9 – 30 В
Флэш-память	32 КБ	порт для microCD	128 Кб
ОЗУ	2 КБ	1024 Мб	20 Кб
Тактовая частота	16 МГц	900 МГц	72 МГц

Продолжение таблицы 4.3.1

Разрядность	8 бит	32 бит	12 бит
Цифровые входы/выходы	14 шт	26 шт	15 шт
Аналоговые входы/выходы	8 шт	0 шт	6 шт
Выходное напряжение	3.3В, 5 В	3.3В, 5 В	3.3В, 5 В
Рабочая температура	от -25 до +85 ºС	от -40 до +85 ºС	от -25 до +85 ºС
Встроенный видеочип	нету	есть	нету
Размеры	18.5 мм × 43 мм	85.6 мм × 56.5 мм	101.6 мм × 86 мм

Исходя из вышеперечисленного в данном проекте используется плата Arduino Nano, так как она полностью подходит под поставленные задачи, а также является более доступной.

4.4 Обоснование выбора Ethernet-контроллера

Рассмотрим сравнительные характеристики Ethernet-контроллеров, представленных в таблице 4.3.1, где показано, что все микроконтроллеры реализуют минимальный уровень MAC канального уровня сетевого взаимодействия. W5100 и PIC18F97J60 могут похвастаться реализованными в них стеком TCP/IP протоколов, но для нашего проекта далеко не все протоколы нужны, а цена этих микроконтроллеров ощутимо больше чем ENC28J60 [23].

На этом, окончательный выбор пал на данный Ethernet-контроллер.

Таблица 4.4.1 – Сравнение Ethernet-контроллеров

Параметры сравнения	ENC28J60	W5100	PIC18F97J60
MAC-уровень	Аппаратно вшит	Аппаратно вшит	Аппаратно вшит
TCP/IP стек	Нет	TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE	TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE

Продолжение таблицы 4.4.1

Интерфейс	SPI	SPI	SPI, UART
Встроенный LCD	Нет	Нет	Да
Флеш-память	Нет	Нет	128 Кб
Память буфера Tx-Rx	8 Кб	16 Кб	8 Кб
Размеры	48 мм × 18 мм	118.3 мм × 143.5 мм	73 мм × 54 мм
Цена	$3.07	$9.5	$43.22

4.5 Обоснование выбора устройства вывода

Тут безоговорочно выбор падает на NodeMCU v3 ESP8266, так как данный микроконтроллер обладает функциями, представленными в стандартах IEEE 802.11 b/g/n для Wi-Fi. Также этот микроконтроллер обладает достаточными объемами флеш-памяти и SRAM для реализации необходимого функционала. Но поскольку SRAM немного, а разметку HTML-страницы придется прописывать вручную, то будем хранить наши строки разметки во флеш-памяти.

4.6 Обоснование выбора стабилизатора напряжения

Будем использовать LM7805BT, так как он обладает необходимыми параметрами входного напряжения (от 6 до 24 В) и максимального выходного тока в 1 А.

5 АРХИТЕКТУРА РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

5.1 Требования к работе алгоритма

Разработанное микропроцессорное устройство собирает информацию следующего типа: IP-адреса хостов в локальной сети, время обращения ICMP-пакета с echo request/reply, MAC-адрес пропингованного хоста, вводимые IP-адреса пользователем.

Устройство работает следующим образом. При включении устройства, микроконтроллер инициализируется и ждет команд от модуля управления. При получении команды обновления таблицы доступных хостов, микроконтроллер переводит Ethernet-модуль в неразборчивый режим, при котором порт начинает принимать и обрабатывать пакеты, которые в поле DMAC не имеют MAC-адрес данного модуля. Основным критерием обработки пакетов является, чтобы в кадре Ethernet в поле пакета верхнего уровня был указан протокол IP. В таком режиме порт работает 2 секунды, просто записывая в оперативную память все IP и MAC адреса, которые смог разобрать. Потом эта таблица фильтруется от повторяющихся адресов и передается по UART-интерфейсу модулю управления. При получении команды пинга, микроконтроллер вытягивает из команды IP-адрес, который нужно пинговать. Далее микроконтроллер проверяет, находится ли он в одной подсети с хостом, если нет, то устанавливает IP-адрес интерфейса в одну подсеть. Далее, микроконтроллер собирает IP пакет с ICMP сообщением для данного адреса и запускает после отправки пакета таймер. После получения ICMP ответа от хоста с нужным IP-адресом, микроконтроллер вытягивает информацию из пакета, останавливает таймер и передает модулю управления результат пинга.

Модуль управления, при начальном запуске, инициализирует Wi-Fi модуль ssid-ом «NodeMCU» и паролем «12345678», а также назначает себе IP-адрес 192.168.2.1. и порт 80. Затем модуль ждет GET-запросов от клиента, которые обрабатываются модулем как команды микроконтроллеру. Модуль эти команды отправляет микроконтроллеру, а их результат получает и отправляет GET-ответом клиенту.

Пользователь может подключиться к точке доступа «NodeMCU» по паролю «12345678» и зайти на страницу веб-сервера по адресу https://192.168.2.1:80/, где увидит таблицу первичных адресов и поле для ввода. Перед этим нужно назначить IP-адрес в сети 192.168.2.0 на интерфейс устройства, с которого происходит подключение. Пользователь может ввести

интересное ему значение IP-адреса в поле для ввода и нажать кнопку «пинг». Также, пользователь может повторно запросить новую таблицу имеющихся адресов нажав на кнопку «обновить таблицу». Если нажата кнопка «пинг» и поле не пустое, то модуль управления передает этот адрес по UART микроконтроллеру.

Функциональная схема алгоритма представлена в приложении Б.

5.2 Блок-схема алгоритма

Блок-схема – это схематичное представление процесса, системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логических диаграмм. Блок-схема алгоритма программного обеспечения данного курсового проекта представлена в приложении В.

5.3 Исходный код для микроконтроллера

Задача микроконтроллера в данном устройстве – получать команды от модуля управления и соответствующе их обрабатывать. Для этого в главном цикле программы (функция *void loop();*) идет постоянная проверка наличия командных пакетов в UART.

Распознавание пакета-команды возложена на функцию void parsing(), которая анализирует информацию поступающую из UART. Признаком командного пакета является флаг начала «$» и флаг конца пакета «;». После флага начала обязательно идет символ-команда. Это могут быть:

– «P» – команда пинга, после нее идет IP-адрес хоста в десятичной форме (прим.: 192.168.1.1).

– «U» – команда обновления таблицы хостов, после нее идет сразу флаг конца пакета.

– «T» – команд отображения текущей таблицы хостов, после нее сразу идет флаг конца пакета. Эта команда использовалась для дебага и для клиента нет возможности ее использовать.

Поскольку информация записывается в UART как ASCII символы, то приходящие числовые значения приходится дополнительно обрабатывать.

Числовые данные, которые идут после команды (случай с командой «U»), заносятся функцией в глобальную переменную *byte buffer[4]*. Перед занесением, функция считывает байты-символы IP-адреса в String

string_convert, которая конвертирует наши значения из символов в числовое значение встроенным методом *toInt()*.

После получения пакета-команды и полезных данных оттуда, дальнейшая обработка переходит функциям *void doIpTable()* и *void doPing()*.

void doIpTable() состоит из функций *void listenNetwork()* и *void filterIpTable()*. *void listenNetwork()* переводит enc28j60 в неразборчивый режим на 2 сек и далее занимается вытягиванием IP и MAC адресов из IP-пакетов. Функции *ENC28J60::enablePromiscuous()* и *ENC28J60::disablePromiscuous()* отвечают за режим работы модуля.

void doPing() отвечает за формирование ICMP пакетов для пинга и замера времени обращения пакета. Эта информация собирается в *pingResult*.

5.4 Исходный код для модуля управления

Задача модуля управления – получать запросы от клиента и отправлять команды микроконтроллеру на выполнение. Команды передаются микроконтроллеру в таком же формате как представлено в п. 5.3.

Общение с клиентом происходит за счет реализованных асинхронных AJAX GET-запросов на клиентской стороне. По нажатию кнопок, происходит создание соответствующего обработчика события, который формирует GET-запрос.

Главные функции данного модуля :

– *void handle_OnConnect()* – вызывается при начальном подключении по адресу веб-сервера. Выдает клиенту index.html, который хранится в памяти NodeMCU.

– *void handle_updateIpTable()* – вызывается при GET-запросе с корнем «*/updateIpTable*». Обработка запроса на сбор информации по доступным хостам в сети. Отправляет микроконтроллеру пакет-команду «$U;». После отработки, получает от микроконтроллера IP-адреса хостов. Отправляет в GET-ответе эти адреса клиенту, отображение на странице происходит благодаря AJAX.

– *void handle_pingHost()* – вызывается при GET-запросе с корнем «*/pingHost?ip=<value>*», где <value> – значение IP-адреса подтянутое с поля ввода. NodeMCU формирует пакет-команду «$P<value>;» и отправляет микроконтроллеру Arduino Nano. После отработки, модуль управления получает от Arduino пакет-команду в формате «$P<times>;», где times – значения времени оборота ICMP-пакета 4-ех последовательных пингов. Время измеряется в миллисекундах.

Отправляет в GET-ответе эти значения, отображение на странице происходит благодаря скрипту на JavaScript.

– *void handle_NotFound()* – вызывается, когда приходит GET-запрос с неизвестным корнем. Выводит клиенту страницу с ошибкой (404 – страница не найдена).

Парсинг приходящих от микроконтроллера данных происходит с помощью функции *void parsing()*.

В главном цикле программы сервер постоянно ожидает запросов от клиента и обрабатывает их при их поступлении.

Исходный код микроконтроллера и модуля управления представлен в приложении А.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта было разработано автономное устройство проверки сетевых соединений. Отличительная особенность данного устройства заключается в том, что можно в дальнейшем расширять его функционал и разработать для него такие функции как, например, *traceroute*, *curl*, *nslookup*, разрешение DNS-адресов или автоконфигурацию по DHCP.

Недостатки устройства:

– NodeMCU в зависимости от текущей работы чипа ESP8266 может потреблять много энергии, что обещает недолгую автономную работу устройства.

– Достоинства устройства:

– Доступность элементной базы.

– Простота конструкции и относительная дешевизна проекта.

В дальнейшем возможно исправление представленных недочетов и внесение необходимых корректировок. С учетом того, что устройство контролируется программным обеспечением, его также можно использовать в качестве системы сбора и хранения сведений о подключаемых LAN с последующим их анализом, составлением ARP-таблиц, DNS-таблиц для работы с символьными адресами сайтов или хостов.

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Эндрю Таненбаум Архитектура компьютера. – 6-е изд. – СПб: Structured Computer Organization, 2013. – 467 с.

[2] Сара Л. Харрис, Дэвид Харрис Цифровая схемотехника: RISC-V. – ДМК Пресс, 2021. – 810 с.

[3] Блум Джереми Изучаем Arduino: инструменты и методы технического волшебства. – СПб: ДМК Пресс, 2017. – 336 с.

[4] Arduino Nano [Электронный ресурс]. – Режим доступа: http://arduino.ru/Hardware/ArduinoBoardNano – Дата доступа: 11.10.2022

[5] Документация Arduino [Электронный ресурс]. – Режим доступа: https://docs.arduino.cc/ – Дата доступа: 12.10.2021

[6] Вычислительные машины, системы и сети: дипломное проектирование (методическое пособие) [Электронный ресурс] : Минск БГУИР 2019. – Режим доступа: https://www.bsuir.by/m/12_100229_1_136308.pdf – Дата доступа: 12.10.2021

[7] Документация Arduino [Электронный ресурс]. – Режим доступа: https://docs.arduino.cc/ – Дата доступа: 18.10.2021

[8] Геддес, М. 25 крутых проектов с Arduino / М. Геддес ; [пер. с англ. М. А. Райтмана]. – Москва : Эксмо, 2019. – 272 с.

[9] Raspberry PI 2 Model B – второе поколение Raspberry Pi [Электронный ресурс]. – Режим доступа: https://micropi.ru/raspberry-pi-2-model-b-rpi-bcm2836-bcm2837/ – Дата доступа: 12.09.2021

[10] ATMEL – ATmega328P, 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash – DATASHEET, 294 pg.

[11] 3DIY [Электронный ресурс]. – Режим доступа: https://3d-diy.ru/wiki/arduino-platy/arduino-nano/. – Дата доступа: 01.10.2022.

[12] Arduino.UA [Электронный ресурс]. – Режим доступа: https://arduino.ua/art2-istoriya-sozdaniya-arduino. – Дата доступа: 10.11.2022.

[13] Arduino Master [Электронный ресурс]. – Режим доступа: https://arduinomaster.ru/platy-arduino/plata-arduino-nano/#i-2. – Дата доступа: 01.10.2022.

[14] Arduino-технология [Электронный ресурс]. – Режим доступа: https://arduino-tech.ru/sample-page/. – Дата доступа: 25.10.2022.

[15] Arduino+ [Электронный ресурс]. – Режим доступа: https://arduinoplus.ru/arduino-nano-dlya-nachinayuschih/#i. – Дата доступа: 10.11.2022.

[16] Амперка [Электронный ресурс]. – Режим доступа: https://amperka.ru/page/arduino-ide. – Дата доступа: 11.11.2022.

[17] Arduino IDE [Электронный ресурс]. – Режим доступа: https://arduino-ide.com. – Дата доступа: 15.11.2022.

[18] Arduino.ru [Электронный ресурс]. – Режим доступа: https://arduino.ru/Arduino_environment. – Дата доступа: 15.11.2022.

[19] Инструментальные системы [Электронный ресурс]. – Режим доступа: http://www.insys.ru/ethernet. – Дата доступа: 22.11.2022.

[20] The fast code [Электронный ресурс]. – Режим доступа: https://www.thefastcode.com/ru-rub/article/wi-fi-vs-ethernet-how-much-better-is-a-wired-connection. – Дата доступа: 06.12.2022.

[21] Composs [Электронный ресурс]. – Режим доступа: http://composs.ru/chto-takoe-ethernet/. – Дата доступа: 06.12.2022.

[22] PIC18F97J60 Family, DataSheet 64/80/100-Pin, High-Performance, 1-Mbit Flash Microcontrollers with Ethernet [Электронный ресурс]. Режим доступа – 1669403.pdf (yandex.by). – Дата доступа: 07.12.2022.

[23] Microchip – ENC28j60 – Stand-Alone Ethernet Controller with SPI Interface, DATASHEET, 96 pg.

ПРИЛОЖЕНИЕ А

(обязательное)
Листинг кода программы

<Код для Arduino>
main.cpp

```
#include <Arduino.h>
#include "../lib/EtherCard/src/EtherCard.h"

static uint32_t timer;
//mac-address
static const byte PROGMEM MAC[] = {
0x72,0x69,0x69,0x2D,0x30,0x31 };
//variables for IP
static const byte PROGMEM sizeOfHosts = 10;
static byte countOfFoundHosts = 0;
static struct host* tableOfHosts;
//const byte PROGMEM IPclassB  [IP_LEN] = {192, 168, 1, 0};
//const byte PROGMEM maskClassB[IP_LEN] = {255, 255, 255, 0};
//Buffer for ethernet-module
static const int PROGMEM BufferSize = 700;
byte Ethernet::buffer[BufferSize];
//For parsing data from UART
static const byte UARTsizeBuffer = IP_LEN;
static byte buffer [UARTsizeBuffer] = {0};
boolean recievedFlag;
boolean getStarted;
boolean commandFlag;
byte index;
String string_convert = "";
char command;
/******************
 * сообщения передаются между контроллерами в формате
{$<command> <values>;}
 * $ = начало пакета передачи, ; = конец передачи
 * <values> = переданное значение
 * <commands>:
 * M = сообщение - после него текст, <values> = 1 значение -
длина строки, далее строка
 * T = отобразить текущую таблицу IP, <values> отсутствует
 * U = обновить таблицу IP, <values> отсутствует
 * P = пинг хоста, <values> = IP написанный через пробел
```

```
 * <values> = передаются через пробел
 * ********************/
struct host{
    byte IP[IP_LEN] = {0};
    byte MAC[ETH_LEN] = {0};
};

void printUpdateIpTableMessage(){
    if (countOfFoundHosts == 0) {
        Serial.println("M No IPs Found;");
        return;
    }
    byte i = 0;
    Serial.print("$U");
    while (i < countOfFoundHosts) {
        for (uint8_t j = 0; j < IP_LEN; ++j) {
            Serial.print(tableOfHosts[i].IP[j], DEC);
            if (j < 3)
                Serial.print('.');
        }
        i++;
        if(i != countOfFoundHosts)
            Serial.print("&");
    }
    Serial.print(";");
}


void filterIPtable(){
    if(countOfFoundHosts > 1){
        for (byte i = 0; i < countOfFoundHosts; i++) { //
            for (byte j = i+1; j < countOfFoundHosts; j++) {
                if (memcmp(&tableOfHosts[i], &tableOfHosts[j],
sizeof(host)) == 0) {
                    for (byte k = j; k < countOfFoundHosts - 1;
k++) {
                        memcpy(&tableOfHosts[k], &tableOfHosts[k
+ 1], sizeof(host));
                    }
                    countOfFoundHosts--;
                }
            }
        }
    }
}
```

```
uint8_t listenNetwork(){
    ENC28J60::enablePromiscuous();       //enable  promiscuous
mode
    timer = millis();
    uint16_t recBytes = 0;
    byte i = 0;
    countOfFoundHosts = 0;
    while((millis() - timer ) <= 2000){     // for 2 sec
collecting IP addresses of hosts
        if( (recBytes = ENC28J60::packetReceive()) !=0){
            if(
                /* (ether.buffer[ETH_TYPE_H_P] ==
ETHTYPE_ARP_H_V && ether.buffer[ETH_TYPE_L_P]==ETHTYPE_ARP_L_V
&&
                 memcmp(ether.buffer + ETH_DST_MAC, ether.mymac,
ETH_LEN) )
                 ||*/
                    (ether.buffer[ETH_TYPE_H_P] ==
ETHTYPE_IP_H_V && ether.buffer[ETH_TYPE_L_P]==ETHTYPE_IP_L_V)
                    ) {
                EtherCard::copyIp(tableOfHosts[i].IP,
ether.buffer + IP_SRC_P);
                EtherCard::copyMac(tableOfHosts[i++].MAC,
ether.buffer + ETH_SRC_MAC);
                if (countOfFoundHosts != sizeOfHosts)
                    countOfFoundHosts+=1;
                if (i == sizeOfHosts) {
                    i = 0;
                    countOfFoundHosts = sizeOfHosts;
                }
            }
        }
    }
    ENC28J60::disablePromiscuous();
    return countOfFoundHosts;
}

void doIPTable(){
    listenNetwork();
    filterIPtable();
    printUpdateIpTableMessage();
}
```

```
static boolean is_lan(const uint8_t source[IP_LEN], const
uint8_t destination[IP_LEN]) {
    if(source[0] == 0 || destination[0] == 0) {
        return false;
    }
    for(int i = 0; i < IP_LEN; i++)
        if((source[i] & EtherCard::netmask[i]) !=
(destination[i] & EtherCard::netmask[i])) {
            return false;
        }
    return true;
}
uint8_t findHost (uint8_t* IP){
    for (int i = 0; i < sizeOfHosts; ++i) {
       if(memcmp(tableOfHosts[i].IP, IP, IP_LEN)==0)
           return i;
    }
    return 0;
}

void doPing(){
    word len = 0;
    host* tmp;
    byte index = 255;
    if((index=findHost(buffer))!= 255) {
        tmp = &tableOfHosts[index];
        if (!is_lan(ether.myip, tmp->IP)) {
            for (int i = 0; i < IP_LEN - 1; i++) {
                ether.myip[i] = (tmp->IP[i] &
EtherCard::netmask[i]);
            }
            ether.myip[3] = 4;
        }
        long ping_result[4];
        for(byte i = 0 ; i<4;i++) {
            timer = micros();
            ether.clientIcmpRequest(tmp->IP, tmp->MAC);
            while (!ether.packetLoopIcmpCheckReply(tmp->IP)) {
                len = ether.packetReceive();
                ether.packetLoop(len);
                if ((micros() - timer > 5000000)) {
                    Serial.print("$MTimeout - host
unreachable;");
                    ether.printIp(buffer);
                    ether.printIp(ether.myip);
```

```
                    break;
                }
            }
            ping_result[i] = micros() - timer;
        }
        Serial.print("$P");
        for(byte i = 0 ; i<4;i++) {
            Serial.print(ping_result[i] * 0.001, 3);
            if(i<3)
            Serial.print("&");
        }
        Serial.println(";");
    }

}

void parsing() {
    while (Serial.available() > 0) {
        char incomingByte = Serial.read();        // обязательно
ЧИТАЕМ входящий символ
        if (incomingByte == '$') {                // если это $
            getStarted = true;                      // поднимаем
флаг, что можно парсить
            index = 0;                           // сбрасываем
индекс
            commandFlag = true;
            string_convert = "";
            continue;                            // очищаем
строку
        }
        if (incomingByte == ';') {                // если таки
приняли ; - конец парсинга
            getStarted = false;                     // сброс
            recievedFlag = true;                    // флаг на
принятие
            return;
        }
        if (getStarted) {                         // если
приняли начальный символ (парсинг разрешён)
            if (incomingByte != '.' && incomingByte != ';') {
// если это не пробел И не конец
                if(commandFlag){
                    command = incomingByte;
                    commandFlag = false;
                    continue;
```

```
                }
                string_convert += incomingByte;       //
складываем в строку
            } else {                                  // если это
пробел или ; конец пакета
                buffer[index] = (byte) string_convert.toInt();
// преобразуем строку в int и кладём в массив
                string_convert = "";                  // очищаем
строку
                index++;                              //
переходим к парсингу следующего элемента массива
            }
        }

    }
}

static void gotPinged(byte* ptr){
}
/*********************************************SCRIPT**************
*********************************************/


void setup() {
    Serial.begin(9600);
    ether.begin(BufferSize, MAC, SS);    //init ether interface
    ether.parseIp(ether.netmask, "255.255.255.0");
    ether.parseIp(ether.myip, "192.168.1.102");
    for(byte i = 0; i < sizeOfHosts; i++){
        tableOfHosts = (host*)malloc(IP_LEN * sizeof(host));
    }
  ether.registerPingCallback(gotPinged);
}


void loop() {
   word len = ether.packetReceive(); // go receive new packets
   word pos = ether.packetLoop(len); // respond to incoming
pings
   parsing();
   byte i = 0;
   if(recievedFlag){
      // command = Serial.read();
       switch (command) {
           case 'U': doIPTable();  break;
```

```
            case 'P': doPing();  break;
            case 'T': while (i < countOfFoundHosts)
ether.printIp("IP: ", tableOfHosts[i++].IP); break;
            default: Serial.println("$M Unknown command;");
        }
        command = 0;
        recievedFlag=0;
    }
}
```

<Код для NodeMCU>

index.h

```
#ifndef ESP8266_INDEX_H
#define ESP8266_INDEX_H
String MAIN_page  = R"====(
<!DOCTYPE html> <html>
<head><meta name="viewport" content="width=device-width,
initial-scale=1.0, user-scalable=yes"
<title>The ESP8266 NodeMCU</title>
<style>
html {
font-family: Helvetica;
 display: inline-block;
 margin: 0px auto;
 text-align: center;}
body{
margin-top: 50px;}
h1 {
color: #444444;
margin: 50px auto 30px;}
h3 {
color: #444444;
margin-bottom: 50px;}
.button {
display: block;
width: 80px;
background-color: #1abc9c;
border: none;
color: white;
padding: 13px 30px;
text-decoration: none;
font-size: 25px;
margin: 0px auto 35px;
cursor: pointer;
border-radius: 4px;}
.button-on {
background-color: #1abc9c;}
.button-on:active {background-color: #16a085;}
.button-off {background-color: #34495e;}
.button-off:active {background-color: #2c3e50;}
p {font-size: 14px;color: #888;margin-bottom: 10px;}
.pingHost{
margin-bottom: 50px;
```

```
}
.pingResult{
margin-top: 50px;
margin-bottom: 50px;
}
</style>
</head>
<body>
<div id="pingHost" class="pingHost">
     <h1>Internet Connectivity Checker Web Page</h1>
     <h2>FORM TO PERFORM HOST PING</h2>
     Enter IP address you want to PING:<br/><br/>
     <input type="text" id="iptext" name="iptext"><br/><br/>
     <button type="button" id="buttonPing"
     class="button-on">PING HOST</button><br/>

     <div class="pingResult">
     Result of the ping (ms)</br><span id="pingResult"> * * *
     *</span>
     </div>
</div>

<div id="ipTablediv" class="h3">
     <h2>IP TABLE CONTENT</h2>
     <button id="buttonUpdate" type="button" class="button-on"
     >UPDATE IP TABLE</button><br/><br/>
     STATUS<span id="ipTableStatus"> - N/A</span><br/><br/>
     IP TABLE:<br/><br/><span id="ipTable">NO DATA (FIRST
     STARTUP)</span>
</div>

<script>

var buttonUpdate = document.getElementById("buttonUpdate");
buttonUpdate.addEventListener("error", transferFailed, false);
function transferFailed(evt) {
  alert("При загрузке файла произошла ошибка.");
}
buttonUpdate.addEventListener("click", function() {
  var command = "$U;";
  var value = "";
  var request = new XMLHttpRequest();
//request.timeout = 5000; // Set timeout to 4 seconds (4000
milliseconds)
request.ontimeout = function () { alert("Timed out!!!"); }
```

```
  request.open("GET", "updateIpTable", false);
  request.send();
  // если статус ответа 200 (OK) то
  if (request.readyState == 4 && request.status==200) {
          if(request.responseText != "" &&
          !request.responseText.includes("P")){
               value = request.responseText;
               value = value.replace("U", "");
               value = value.replace(";", "");
               value = value.replaceAll("&", "<br/>");
               document.getElementById("ipTable").innerHTML =
               value;
               document.getElementById("ipTableStatus").innerHTM
               L = " - UP TO DATE"
          }
     }
});



var buttonPing = document.getElementById("buttonPing");
buttonPing.addEventListener("error", transferFailed, false);
buttonPing.addEventListener("click", function() {
  var command = "$P";
  var value = "";
  var ip = document.getElementById("iptext").value;
  var request = new XMLHttpRequest();
//request.timeout = 5000; // Set timeout to 4 seconds (4000
milliseconds)
request.ontimeout = function () { alert("Timed out!!!"); }
  request.open("GET", "pingHost?ip="+ip+";", false);
  request.send();
  // если статус ответа 200 (OK) то

  if (request.readyState == 4 && request.status==200) {
          if(request.responseText != "" &&
          !request.responseText.includes("U")){
               value = request.responseText;
               value = value.replace("P", "");
               value = value.replace(";", "");
               value = value.replaceAll("&", "ms ");
            value+="ms";
               document.getElementById("pingResult").innerHTML =
               value;
          }
```

```
      }
});

</script>
</body>
</html>
)====";
#endif //ESP8266_INDEX_H
```

main.cpp

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WiFiServer.h>
#include "index.h"
/* Установите здесь свои SSID и пароль */
const char* ssid = "NodeMCU";       // SSID
const char* password = "12345678";  // пароль
/* UART settings */
boolean recievedFlag;
boolean getStarted;
boolean commandFlag;
String UART_receiver = "";
byte count;
char command;
long timer = 0;
/* Настройки IP адреса */
IPAddress local_ip(192,168,2,1);
IPAddress gateway(192,168,2,1);
IPAddress subnet(255,255,255,0);

ESP8266WebServer server(80);

void parsing() {
    timer = millis();
    recievedFlag = false;
    while (Serial.available() > 0 && !recievedFlag) {
        char incomingByte = Serial.read();        // обязательно
ЧИТАЕМ входящий символ
        if (incomingByte == '$') {                // если это $
            getStarted = true;                      // поднимаем
флаг, что можно парсить
            count = 0;                           // сбрасываем
индекс
            commandFlag = true;
            UART_receiver = "";
            continue;                            // очищаем
строку
        }
        if (incomingByte == ';') {                // если таки
приняли ; - конец парсинга
            getStarted = false;                     // сброс
            recievedFlag = true;                    // флаг на
принятие
```

```
        }
        if (getStarted) {                           // если
приняли начальный символ (парсинг разрешён)
            if (incomingByte != ';') {   // если это не пробел И
не конец
                UART_receiver += incomingByte;       //
складываем в строку
            }
        }
    }

    if(millis()-timer > 2000){
        UART_receiver = "$MRequest timeout";
    }
}


String SendHTML()
{
    String ptr = MAIN_page;
    return ptr;
}

void handle_OnConnect()
{
    server.send(200, "text/html", SendHTML());
}

void handle_updateIpTable()
{
    char* command = "$U;";
    Serial.print(command);
    //timer=millis();
   // while (millis() - timer < 2000);
    parsing();
    server.send(200, "text/html", UART_receiver);
    UART_receiver = "";
}

void handle_pingHost()
{
    String ip = server.arg("ip");
    String command ="";
    command = "$P" + ip + ";";
    Serial.print(command.c_str());
```

```
    parsing();
    server.send(200, "text/html", UART_receiver);
    UART_receiver = "";
}

void handle_NotFound()
{
    server.send(404, "text/plain", "Not found");
}



/*****************************SCRIPT*************************
**********************/
void setup()
{
    Serial.begin(9600);
    WiFi.persistent(false);
    WiFi.mode(WIFI_AP_STA);
    WiFi.softAP(ssid, password);
    WiFi.softAPConfig(local_ip, gateway, subnet);
    delay(100);

    server.on("/", handle_OnConnect);
    server.on("/updateIpTable", handle_updateIpTable);
    server.on("/pingHost", handle_pingHost);
    server.onNotFound(handle_NotFound);
    server.begin();

    Serial.println("HTTP server started");
}

void loop()
{
    server.handleClient();
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Функциональная схема алгоритма, реализующего программное средство

ПРИЛОЖЕНИЕ В

(обязательное)

Блок схема алгоритма, реализующего программное средство

ПРИЛОЖЕНИЕ Г

(обязательное)

Графический интерфейс пользователя

ПРИЛОЖЕНИЕ Д

(обязательное)

Ведомость