



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Прикладная информатика

## РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### К КУРСОВОЙ РАБОТЕ

*по дисциплине «Микропроцессорные системы»*

**НА ТЕМУ:**

**Фильтр сетевого трафика**

---

---

---

---

---

Студент

ИУ6-73Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Т. Р. Ярулин  
(И.О. Фамилия)

Руководитель

\_\_\_\_\_  
(Подпись, дата)

И. Б. Грамов  
(И.О. Фамилия)

2024 г.

## СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ.....	3
ВВЕДЕНИЕ .....	4
1 КОНСТРУКТОРСКАЯ ЧАСТЬ .....	5
1.1 Проектирование функциональной схемы .....	5
1.1.1 Базовое описание принципа работы устройства .....	5
1.1.2 Микроконтроллер ATmega164P .....	5
1.1.2.1 Используемые элементы.....	7
1.1.2.2 Распределение портов.....	8
1.1.2.3 Описание работы интерфейса I2C .....	8
1.1.2.4 Описание работы интерфейса UART .....	10
1.1.3 Внешние компоненты .....	13
1.1.3.1 Модуль HC-05 .....	13
1.1.3.2 Модуль DS3232.....	13
1.1.3.3 Устройство вывода звука MH-FMD .....	14
1.2 Построение функциональной схемы .....	14
1.3 Проектирование принципиальной схемы .....	14
1.3.1 Технические характеристики компонентов устройства .....	15
1.3.1.1 ATmega164P.....	15
1.3.1.2 HC-05 .....	15
1.3.1.3 DS3232.....	16
1.3.1.4 MH-FMD .....	16
1.3.2 Подключение компонентов схемы.....	17
1.3.3 Оценка потребляемой мощности.....	19
1.4 Построение принципиальной схемы .....	20
1.5 Алгоритм работы.....	20
2 КОНСТРУКТОРСКАЯ ЧАСТЬ .....	23
2.1 Настройка получение времени по DS3232.....	23
2.2 Настройка интерфейсов UART0 и UART1 .....	25
2.3 Отладка .....	29
2.4 Тестирование устройства.....	30
ЗАКЛЮЧЕНИЕ .....	36
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	37
ПРИЛОЖЕНИЕ А.....	39
ПРИЛОЖЕНИЕ Б.....	52
ПРИЛОЖЕНИЕ В.....	53

## **СПИСОК СОКРАЩЕНИЙ**

МК	Микроконтроллер
ПО	Программное обеспечение
ПЭВМ	Персональная электронно-вычислительная машина
UART	Universal Asynchronous Receiver-Transmitter

## **ВВЕДЕНИЕ**

В данной работе производится разработка микроконтроллерной системы для фильтрации сетевого трафика на основе семейства AVR. Целью является реализация системы, способной в реальном времени обрабатывать сетевой трафик, выявлять запрещенные слова и осуществлять соответствующую реакцию.

В современном мире, где обмен данными играет ключевую роль, вопросы безопасности информации становятся особенно актуальными. Микропроцессорные системы находят широкое применение в задачах фильтрации и анализа данных, обеспечивая высокую точность, оперативность и гибкость управления.

Фильтрация трафика в сетях является одной из важных задач для предотвращения распространения запрещенного контента, защиты информационных систем и обеспечения соблюдения нормативных требований. Такие системы применяются как в корпоративных сетях для контроля сотрудников, так и в домашних условиях для обеспечения безопасного интернет-доступа.

В рамках данного проекта реализуется фильтрация сетевого трафика путем поиска запрещенных слов. Система будет получать список таких слов от внешних устройств, включая персональный компьютер и мобильный телефон. При обнаружении запрещенных данных планируется подача сигнала на звуковое устройство, а также отправка информации о нарушении на компьютер.

Кроме того, предусмотрена еженедельная передача статистики фильтрации трафика, что позволяет анализировать эффективность системы и оценивать её производительность. Особенностью данного проекта является реализация алгоритмов фильтрации на микроконтроллере AVR, а также отладка системы в симуляторе или на аппаратном макете.

Проект охватывает разработку схемы устройства, алгоритмов, программы управления, а также оценку энергопотребления системы. Такое комплексное решение предоставляет надежный инструмент для обработки и анализа данных в сетях различного масштаба.

# **1 КОНСТРУКТОРСКАЯ ЧАСТЬ**

## **1.1 Проектирование функциональной схемы**

В этом разделе приведено функциональное описание работы системы и проектирование функциональной схемы.

### **1.1.1 Базовое описание принципа работы устройства**

Фильтр сетевого трафика – устройство безопасности, предназначенное для обработки передаваемых через него данных, на основе правил фильтрации, которые формирует оператор устройства.

В рамках данной работы разработано фильтр сетевого трафика со следующими функциональными возможностями:

- Обработка сетевого трафика, проходящего через микроконтроллер в режиме полного дуплекса;
- Фильтрация трафика по принципу поиска запрещенных слов;
- Передача списка запрещенных слов с ПЭВМ и мобильного телефона;
- Реакция системы при обнаружении запрещенных данных:
  - Подача сигнала на устройство вывода звука;
  - Отправка информации об инциденте на ПЭВМ.
- Передача на ПЭВМ статистики фильтрации трафика раз в неделю.

Для управления устройством, ПЭВМ оператора подключается по узлу UART. Оператор передает сообщения на мобильный телефон, настраивает правила фильтрации, просматривает еженедельную статистику срабатываний. Мобильный телефон пользователя подключается к фильтру сетевого трафика по протоколу Bluetooth. Пользователь отправляет сообщения и редактирует правила фильтрации.

Еженедельная отправка статистика осуществляется за счет синхронизации устройства с часами реального времени DS3232. Часы подключены к фильтру сетевого трафика по шине I2C.

### **1.1.2 Микроконтроллер ATmega164P**

Для выполнения поставленной задачи, в соответствии требованиям ТЗ, была выбрана микроконтроллерная интегральная схема ATmega164P от компании Microchip Technology.

Микроконтроллер ATmega164P представляет собой 8-битное устройство с архитектурой AVR, включающее процессорное ядро, а также богатый набор периферийных модулей. Это

позволяет эффективно решать задачи обработки данных и управления внешними устройствами. Основными критериями выбор ATmega164P явились:

- 1) Широкий набор периферийных устройств:
  - Два 8-битных и один 16-битный таймер/счетчик;
  - Два интерфейса UART: UART0 и UART1, интерфейсы SPI и I<sup>2</sup>C для работы с внешними устройствами, такими как ПК и мобильные телефоны.
- 2) Широкая доступность и документированность: ATmega164P широко используется в микроконтроллерных системах, что обеспечивает легкость разработки и отладки;
- 3) 32 универсальных программируемых ввода/вывода для подключения периферии.

Структурная схема микроконтроллера ATmega164P приведена на Рисунок 1

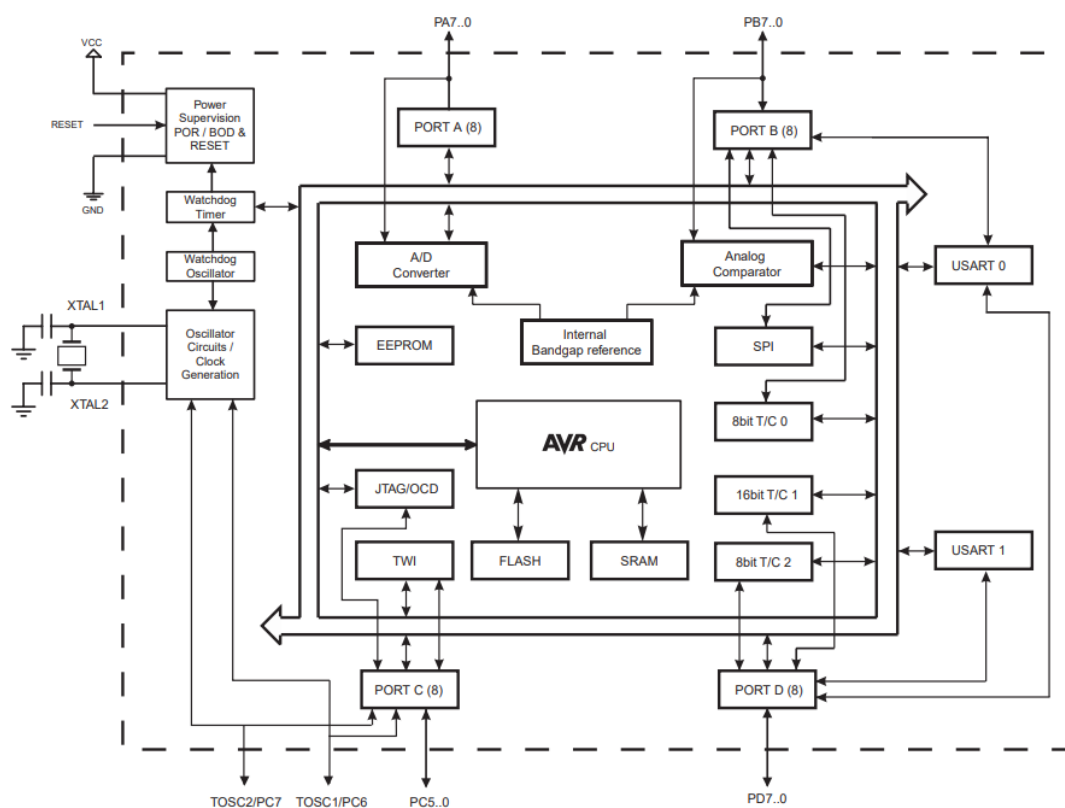


Рисунок 1 – Структурная схема ATmega164P

Наличие двух интерфейсов UART обеспечивает надежную работу устройства, позволяя разделить физическую среду передачи данных на два домена коллизий. Это позволяет взаимодействовать устройствам, подключенным к фильтру сетевого трафика, не прерывая работу друг друга.

Наличие шины I<sup>2</sup>C обеспечивает точную и надежную синхронизацию времени. В случае выключения устройства, отчет недели не будет нарушен.

### 1.1.2.1 Используемые элементы

Для разработки системы фильтрации сетевого трафика на основе микроконтроллера ATmega164P в проекте используются следующие элементы его архитектуры:

- Порты ввода/вывода (Порты C, D):

Эти порты используются для подключения внешних устройств, таких как, устройства вывода звука и модули связи (например, HC-05 для Bluetooth-соединения). Порты обеспечивают передачу данных между микроконтроллером и внешними модулями;

- SRAM (статическая память):

В этой памяти хранятся переменные, которые используются во время работы программы. Микроконтроллер ATmega164P имеет достаточный объем SRAM для хранения данных, связанных с фильтрацией сетевого трафика и управления внешними модулями;

- Регистры общего назначения:

Эти регистры используются для хранения операндов в процессе выполнения арифметико-логических операций и для хранения адресов ячеек памяти, что необходимо для обработки данных;

- АЛУ:

Блок процессора, который выполняет арифметические и логические операции над данными. Это необходимо для выполнения фильтрации трафика на основе поиска запрещенных слов;

- SREG (регистр состояния):

Содержит флаги состояния, которые показывают текущее состояние работы микроконтроллера, такие как флаг переноса, флаг переполнения, флаг нуля и другие. Эти флаги важны для контроля выполнения операций;

- Программный счетчик:

Используется для указания следующей команды, которая должна быть выполнена в программе. Он контролирует последовательность выполнения инструкций;

- Память Flash:

Память, в которой хранится загруженная программа. В этом блоке хранится весь код, отвечающий за фильтрацию сетевого трафика и взаимодействие с внешними устройствами;

- Таймеры/счетчики:

В проекте используется таймер для реализации периодической передачи статистики фильтрации или для работы с временными интервалами, связанными с обработкой данных;

- **Прерывания:**

Контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств, таких как таймеры или порты ввода/вывода. Прерывания используются для управления обработкой входящих данных и сигналов от внешних устройств;

- **Шина I2C:**

Порт C на ATmega164P используется для работы с I2C интерфейсом. I2C — это двухпроводной протокол для связи микроконтроллера с внешними устройствами;

- **Интерфейс UART:**

Порт D используется для реализации UART. UART используется для последовательной передачи данных между микроконтроллером и внешним устройством, таким как персональный компьютер, через последовательный порт.

### **1.1.2.2 Распределение портов**

МК ATmega164P содержит следующие порты: A, B, C, D. Разработанное устройство функционирует на основе портов C и D.

Порт C используется для подключения часов реального времени DS3232 по шине I2C, пины PC0 и PC1 соответствуют SCL, SDA указанной шины и соединены с пинами SCL, SDA DS3232.

Порт D используется для подключения ПЭВМ и HC-05 по UART0 и UART1 соответственно, а также для подключения устройства вывода звука. Пины PD0/RXD0 и PD1/TXD0 ATmega164P подключены к пинам TXD и RXD ПЭВМ. Пины PD2/RXD1 и PD3/TXD1 ATmega164P подключены к пинам TXD и RXD HC-05. Пин PD6 МК подключен к устройству вывода звука.

### **1.1.2.3 Описание работы интерфейса I2C**

Интерфейс I2C (Inter-Integrated Circuit) — это синхронный последовательный протокол связи, который используется для обмена данными между микроконтроллерами и периферийными устройствами. Основные принципы работы I2C:

1. **Архитектура "ведущий-ведомый"**

В I2C-шине используется два типа устройств: ведущие (master) и ведомые (slave). Ведущий управляет процессом передачи данных, генерирует тактовые импульсы и инициирует обмен данными с ведомыми;



## 2. Две линии связи

I2C использует две линии связи:

- SDA (Serial Data Line): передача данных;
- SCL (Serial Clock Line): синхронизация передачи данных.

Оба сигнала являются двунаправленными и подтягиваются к уровню питания через резисторы;

## 3. Адресация устройств

Каждое ведомое устройство имеет уникальный 7-битный адрес. Передача данных начинается с отправки стартового условия (START), за которым следует адрес устройства и бит, указывающий операцию чтения или записи;

## 4. Формат передачи данных

Передача данных организована в байтах. Каждый байт подтверждается битом ACK/NACK:

- ACK (Acknowledge): сигнал подтверждения успешного приема данных;
- NACK (Not Acknowledge): сигнал завершения передачи.

## 5. Работа с ведомыми устройствами

После адресации устройства ведущий может передать данные в ведомое или запросить данные от него. Завершение передачи данных сопровождается генерацией стоп-условия (STOP);

## 6. Синхронизация и управление

Все устройства синхронизируются по линии SCL, а управление состоянием шины (старт/стоп) осуществляется ведущим.

Структурная схема интерфейса I2C приведена на *Рисунок 2*.

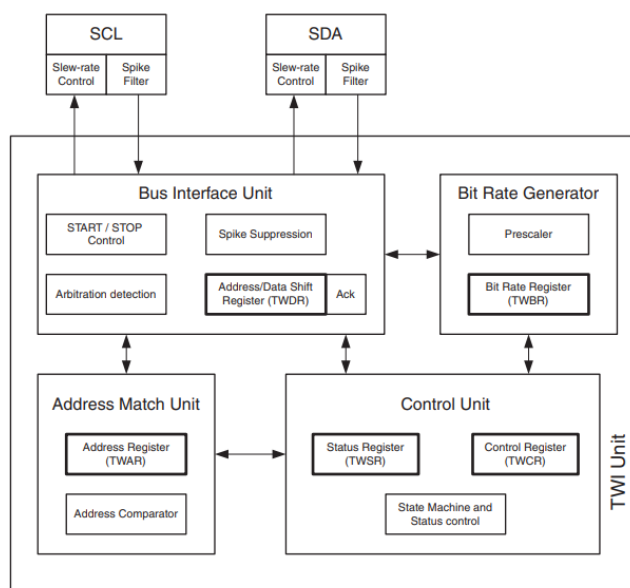


Рисунок 2 – Структурная схема интерфейса I2C

В фильтре сетевого трафика I2C используется для взаимодействия с модулем часов реального времени DS3232. Основные этапы:

1. Инициализация шины Подготовка микроконтроллера для работы с I2C, включая настройку скорости передачи данных;
2. Генерация стартового сигнала Начало передачи данных, которое устанавливает связь между ведущим устройством (микроконтроллером) и ведомым (DS3232);
3. Передача данных Ведущий отправляет адрес устройства, указывает требуемые регистры и передаёт необходимые данные, например, для установки времени;
4. Чтение данных Ведущий запрашивает и получает значения из регистров, например, текущие секунды, минуты и часы;
5. Завершение передачи Генерация стоп-сигнала для разрыва связи после завершения обмена.

Взаимодействие через I2C позволяет организовать надёжный и синхронизированный обмен данными с внешним модулем, обеспечивая как чтение текущего времени, так и его настройку.

#### **1.1.2.4 Описание работы интерфейса UART**

Интерфейс UART — это асинхронный последовательный протокол связи, используемый для обмена данными между микроконтроллерами, компьютерами и периферийными устройствами. Основные принципы работы UART:

1. Асинхронная передача данных

UART не использует тактовый сигнал для синхронизации передачи данных. Все устройства, участвующие в обмене, должны работать с одинаковой скоростью передачи данных (бит/сек). Это упрощает подключение и настройку, но требует точности в настройке скорости передачи на обоих концах канала связи;

2. Два направления связи

UART включает два направления связи:

- TX (Transmit) — линия передачи данных, по которой отправляются данные;
- RX (Receive) — линия приема данных, по которой принимаются данные.

Каждое устройство, подключенное к UART, имеет свои линии TX и RX, что позволяет двусторонний обмен данными;

3. Формат передачи данных

Передача данных организована в виде последовательности битов. Каждый передаваемый байт включает несколько частей:

- Стартовый бит: сигнализирует начало передачи;
- Данные: 5-9 бит данных (зависит от настроек);
- Бит четности (опционально): используется для проверки ошибок в передаваемых данных;
- Стопковые биты: один или два бита, обозначающие конец передачи данных;
- Скорость передачи данных (битрейт);
- Скорость передачи данных, или битрейт, должна быть одинаковой для всех устройств в системе. Обычно используются стандартные значения, такие как 9600, 115200 бит/с и другие.

#### 4. Проверка на ошибки

UART поддерживает базовую проверку ошибок с помощью бита четности (если используется). Также могут быть использованы другие методы, такие как контрольная сумма или механизмы управления потоком;

#### 5. Работа с данными

Передача данных осуществляется в виде байтов. Когда устройство передает данные, оно делит их на пакеты (обычно байты), которые посылаются по линии TX. При получении данных устройство использует линию RX для принятия информации.

Структурная схема интерфейса UART приведена на Рисунок 3.

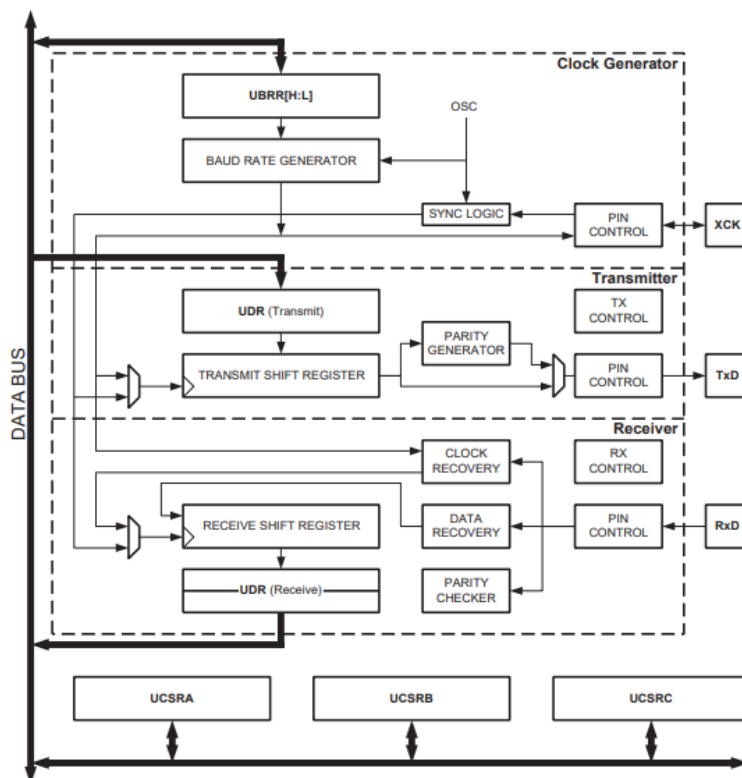


Рисунок 3 – Структурная схема интерфейса UART

В фильтре сетевого трафика UART используется для передачи данных между микроконтроллером и внешними устройствами: ПЭВМ и мобильный телефон. Основные этапы работы:

1. Инициализация UART Настройка скорости передачи данных, количества бит данных, стоп-битов и режима контроля четности. Эти параметры определяют протокол связи между микроконтроллером и внешним устройством;
2. Передача данных Микроконтроллер отправляет данные через UART, преобразуя их в последовательный поток битов. Данные передаются байт за байтом с использованием буфера передачи;
3. Прием данных Микроконтроллер принимает последовательный поток данных от внешнего устройства. Принятые данные преобразуются в байты и помещаются в буфер приема;
4. Обработка данных После приема данные могут быть проанализированы, обработаны или переданы в другие модули микроконтроллера для выполнения соответствующих задач. Например, данные могут использоваться для управления устройствами или записи в память;
5. Управление флагами передачи и приема Используются флаги состояния, чтобы отслеживать завершение передачи и готовность к приему новых данных;
6. Взаимодействие с периферийным устройством Через UART осуществляется обмен данными, необходимыми для настройки или мониторинга периферийного устройства. Например, могут передаваться команды для изменения настроек или запросы текущего состояния устройства.

Использование UART позволяет обеспечить надежную и эффективную связь между микроконтроллером и внешними устройствами, что важно для реализации функциональности устройства.

### 1.1.3 Внешние компоненты

Для реализации поставленной задачи были выбраны два внешних компонента: модуль Bluetooth HC-05 и модуль реального времени DS3232. Подключение описано в разделе 1.1.2.2.

#### 1.1.3.1 Модуль HC-05

Модуль HC-05 используется для беспроводной передачи данных между микроконтроллером и другими устройствами, такими как персональные компьютеры и мобильные телефоны. Этот модуль позволяет передавать список запрещенных слов с внешних устройств на микроконтроллер, используя стандартный Bluetooth-подключение. HC-05 поддерживает два режима работы: мастер и слейв, что позволяет ему как инициировать соединение, так и принимать запросы. Модуль предоставляет удобный интерфейс для обмена данными с микроконтроллером через последовательный порт (UART), что упрощает интеграцию в систему.

Структурная схема HC-05 приведена на Рисунок 4.

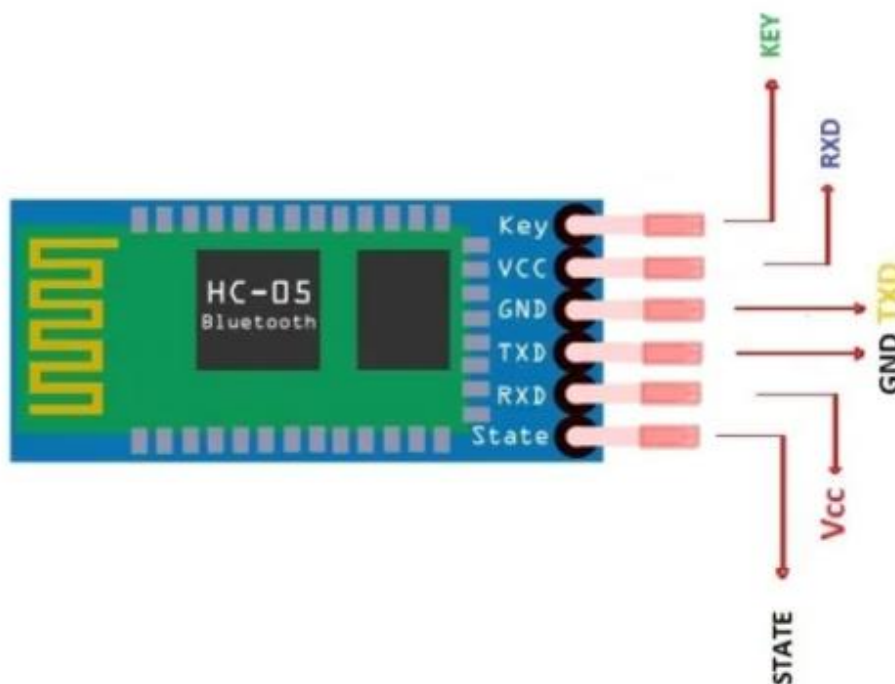


Рисунок 4 – Структурная схкма HC-05

#### 1.1.3.2 Модуль DS3232

Модуль DS3232 используется для отслеживания времени и синхронизации системы. Это высокоточный модуль реального времени, который используется для отслеживания интервала времени для передачи статистики фильтрации трафика каждую неделю. Модуль позволяет точно отслеживать текущее время и поддерживает работу с батарейным питанием, что

гарантирует сохранение времени даже при отключении внешнего питания. Это очень важно для системы, которая должна передавать отчеты и статистику по времени.

Структурная схема DS3232 изображена на Рисунок 5.

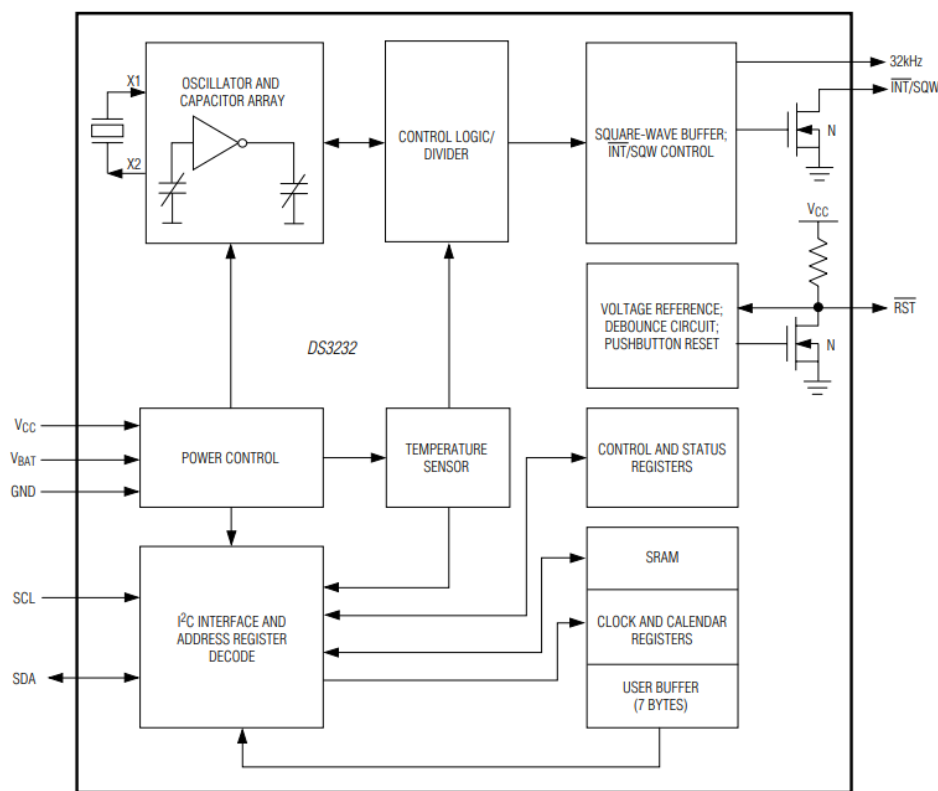


Рисунок 5 – Структурная схема DS3232

### 1.1.3.3 Устройство вывода звука МН-FMD

Устройство вывод звука МН-FMD используется устройством для индикации срабатывания запрещающего правила. Звуковой сигнал срабатывает, когда на пин Input приходит сигнал с пина PD6 МК ATmega164P.

## 1.2 Построение функциональной схемы

На основе раздела 1.1 была построена функциональная схема фильтра сетевого трафика. Указанная схема приведена в ПРИЛОЖЕНИЕ В настоящей расчетно-пояснительной записки.

## 1.3 Проектирование принципиальной схемы

На основе функциональной схемы проектировалась принципиальная схема. Принципиальная схема содержит всю информацию, необходимую для конструирования проектируемого устройства.

### 1.3.1 Технические характеристики компонентов устройства

В данном подразделе описаны основные технические характеристики компонентов фильтра сетевого трафика.

#### 1.3.1.1 ATmega164P

Микроконтроллер ATmega164P обладает следующими техническими характеристиками:

- Ядро: AVR, с поддержкой набора инструкций RISC;
- Тактовая частота: до 20 МГц;
- Память:
  - 16 КБ флэш-памяти для хранения программ;
  - 1 КБ SRAM для данных;
  - 512 байт EEPROM для энергонезависимого хранения данных.
- Порты ввода-вывода: A, B, C, D;
- Таймеры:
  - Три таймера/счетчика (2 x 8-битных, 1 x 16-битный);
- Интерфейсы:
  - I2C (TWI), UART, SPI;
- Питание:
  - Напряжение питания от 1.8 В до 5.5 В;
- АЦП: 10-разрядный модуль, 8 каналов;
- Особенности:
  - Поддержка режима Power-down для энергосбережения;
  - Возможность внешнего или внутреннего тактирования.

#### 1.3.1.2 HC-05

HC-05 обладает следующими техническими характеристиками:

- Протокол связи: Bluetooth 2.0 + EDR (Enhanced Data Rate);
- Напряжение питания: 3.3 В для ядра, допускает до 6 В на входе через регулятор;
- Интерфейсы: UART с поддержкой скоростей от 9600 до 1382400 бод;
- Мощность передатчика:
  - Класс 2 (до 4 дБм).
- Дальность связи: до 10 метров (в зависимости от условий);
- Рабочая температура: от -20°C до +75°C;
- Особенности:

- Поддержка режима команд (АТ-команды) для настройки параметров;
- Возможность работы как в режиме «ведущий» (master), так и «ведомый» (slave);
- Низкое энергопотребление.

#### **1.3.1.3 DS3232**

DS3232 обладает следующими техническими характеристиками:

- Точность:  $\pm 2$  ppm (секунды в месяц) при температуре от  $-40^{\circ}\text{C}$  до  $+85^{\circ}\text{C}$ ;
- Напряжение питания: от 2.3 В до 5.5 В;
- Встроенный кварц: 32.768 кГц, температурно-компенсированный;
- Энергонезависимая память:
  - 236 байт для пользовательских данных (SRAM).
- Функции:
  - Часы, календарь, будильник, таймер;
  - Поддержка форматов времени 12/24 часа.
- Особенности:
  - Встроенная батарейная поддержка с автоматическим переключением питания;
  - Поддержка температуры с встроенным термометром (вывод в виде цифрового значения).

#### **1.3.1.4 МН-FMD**

МН-FMD обладает следующими техническими характеристиками:

- Рабочее напряжение: 3.3–5 В;
- Размеры платы: около  $31 \times 13$  мм;
- Тип модуля: активный пьезоэлектрический зуммер;
- Принцип действия: модуль имеет встроенный генератор, который позволяет ему работать от постоянного тока, без необходимости использования внешних генераторов сигналов;
- Подключение: три вывода: VCC (питание), GND (земля), и I/O (управляющий сигнал от микроконтроллера).



### 1.3.2 Подключение компонентов схемы

В спроектированном устройстве, требуется согласование уровней сигналов между пинами PD3 и PD6 ATmega164P и пинами RXD HC-05, I/O MH-FMD соответственно, так как пины подключаемые компонентов работают по входному 3.3 В, а выходное напряжение на ATmega164P равно 5.5 В. Отсутствие резисторов для согласования уровней приведет к потере данных или к выходу из строя подключаемых компонентов.

Для согласования уровней сигналов используются два делителя напряжения с 5.5 В на 3.3 В. Делители подключаются между пинами PD3 и PD6 ATmega164P и пинами RXD HC-05, I/O MH-FMD соответственно. Номиналы резисторов  $R_1, R_5 = 1 \text{ кОм}$ ,  $R_2, R_6 = 2 \text{ кОм}$ . Подключение делителей приведено на Рисунок 6.

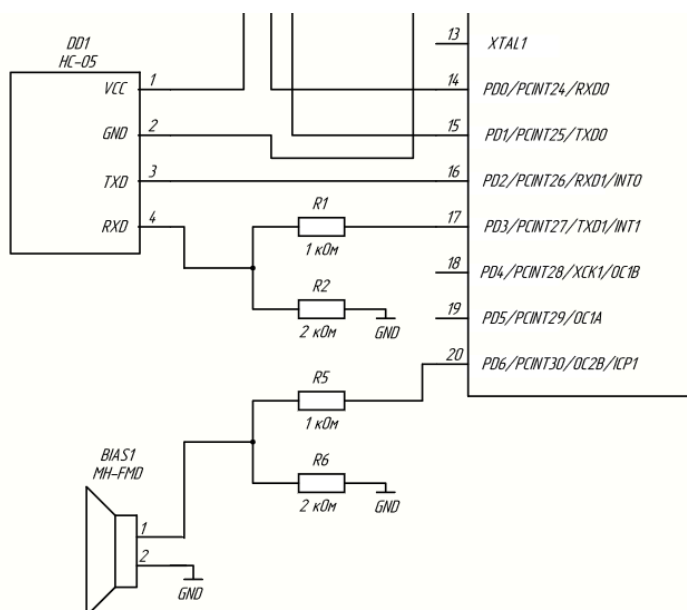


Рисунок 6 – Подключение делителей напряжения

Для обеспечения корректной работы модуля DS3232, к линиям данных SDA и SCL подключаются подтягивающие резисторы. Они подтягивают линию данных к высокому уровню напряжения (логическая "1") в отсутствии активного сигнала. Без них линии могут быть в "плавающем" состоянии, что вызовет ошибки в передаче данных. В соответствии с документацией на часы реального времени DS3232, выбраны резисторы номиналом 4.7 кОм.

Подключение подтягивающих резисторов приведено на Рисунок 7.

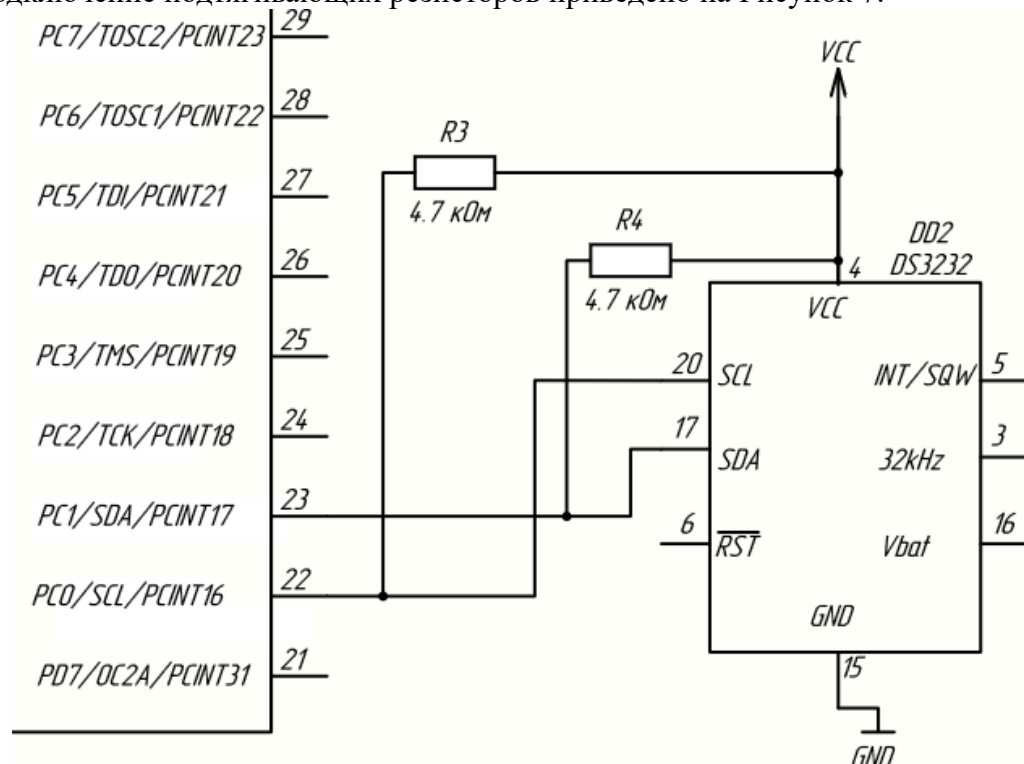


Рисунок 7 – Подключение подтягивающих резисторов

Подключение ПЭВМ к устройству происходит через разъем RS232. Соответствие пинов описано в пункте 1.1.2.2. Подключение разъема RS232 изображено на Рисунок 8.

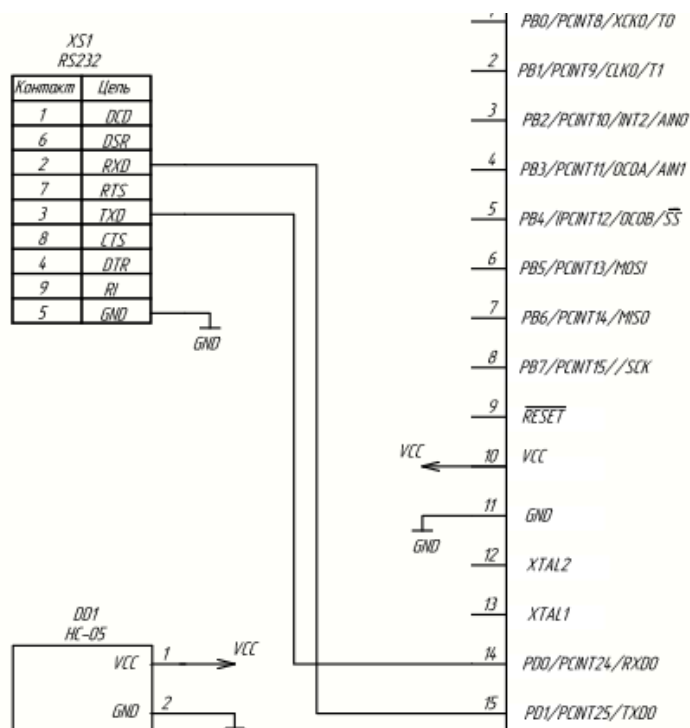


Рисунок 8 – Подключение RS232 к устройству

### 1.3.3 Оценка потребляемой мощности

Согласно документации на МК ATmega164P, при работе в активном режиме на частоте 1 МГц, напряжении питания 5 В, при работе двух UART и I2C интерфейсов, контроллер имеет следующий ток потребления:

$$I_{МК} = 8 \text{ мА}$$

$$I_{UART} = 1 \text{ мА}$$

$$I_{I2C} = 2 \text{ мА}$$

$$I_{\text{суммарная}} = 8 + 2 \cdot 1 + 2 = 12 \text{ мА}$$

Для Bluetooth-модуля HC-05 потребляемый ток зависит от его текущего режима: – в режиме поиска пары модуль потребляет 40 мА; – в режиме, когда пара создана, но устройство не подключено, модуль потребляет 8 мА; 45 – в режиме, когда установлено соединение с устройством, модуль потребляет 20 мА. В Разработанном устройстве используется последний режим, так как оно работает как ведомое. Имеем  $I_{HC-05} = 20 \text{ мА}$ .

Модуль часов реального времени DS3232 является устройством с крайне низким током потребления. Согласно документации, при напряжении 5 В ток потребления равен 0.2 мА, но так как модуль взаимодействует с I2C, то необходимо учитывать и его ток потребления. Имеем  $I_{DS3232} = 2.2 \text{ мА}$ .

RS3232 в активном режиме потребляет **0.2 мА**, MH-FMD аналогично. Итоговые результаты по токам потребления и мощности приведены в Таблица 1.

Таблица 1 – Значения токов и мощностей

Модуль	Ток потребления, мА	Напряжение питания, В	Потребляемая мощность, мВт $P = I \cdot U$
ATmega164P	12	5	60
HC-05	20		100
DS3232	2.2		11
RS3232	0.2		1
MH-FMD	0.2		1

Общая потребляемая мощность устройства равна сумме мощностей из Таблица 1. Тогда имеем:

$$P_{\text{сумм}} = 60 + 100 + 11 + 1 + 1 = 173 \text{ мВт}$$

## 1.4 Построение принципиальной схемы

На основе раздела 1.3 разработана принципиальная схема. Принципиальная схема содержит всю информацию, необходимую для конструирования проектируемого устройства. Указанная схема приведена в ПРИЛОЖЕНИЕ В настоящей расчетно-пояснительной записки.

## 1.5 Алгоритм работы

В данном разделе описывается алгоритм работы разработанного устройства. При подключении к фильтру сетевого трафика с ПЭВМ выводится приветственное сообщение, далее выводится приглашение командной строки. Устройство ждет ввода команд от оператора. При получении команды устройство анализирует ее на валидность. Если команда невалидна, то оператор оповещается об этом. В случае валидности команда обрабатывается. Каждую секунду фильтр сетевого трафика опрашивает по прерыванию TIMSK1 часы DS3232 на предмет истечения недели. Если неделя прошла, то статистика отправляется на ПЭВМ. Аналогично происходит взаимодействия пользователя с устройством по Bluetooth, за исключением отправки статистики. Параллельная работа двух устройств, реализована следующим образом: работа с ПЭВМ происходит в главном цикле программы, работа с мобильным устройством происходит в прерывании UART1 по получению данных.

Схемы алгоритмов основной программы и прерывания UART1 приведены на Рисунок 9.

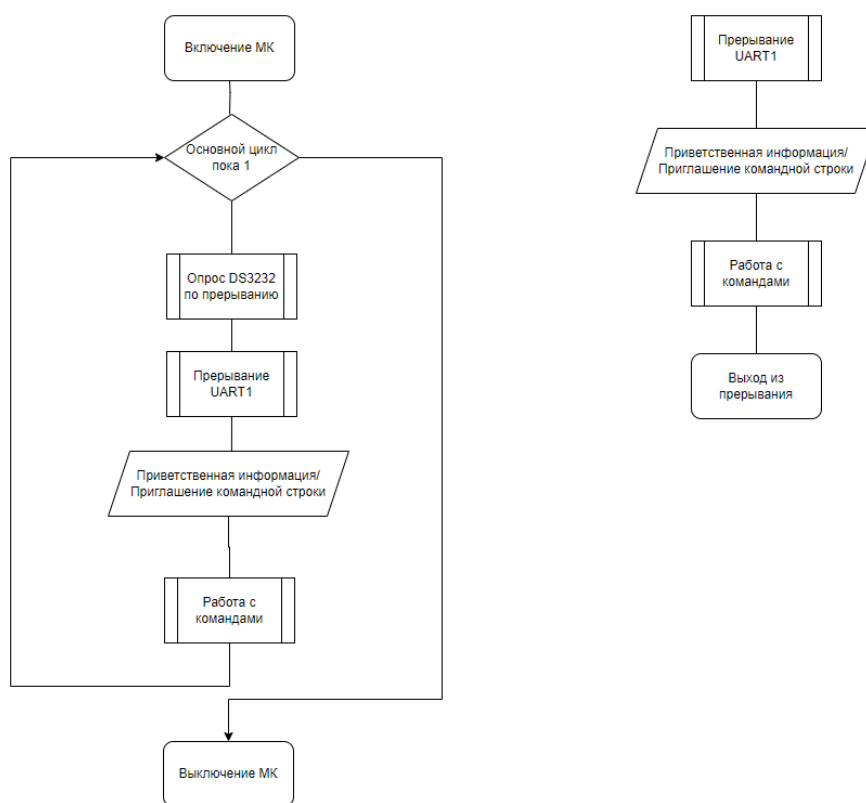


Рисунок 9 – Схемы алгоритмов работы основного цикла и прерывания UART1

Схема алгоритма прерывания для опроса DS3232 и приведена на Рисунок 11.

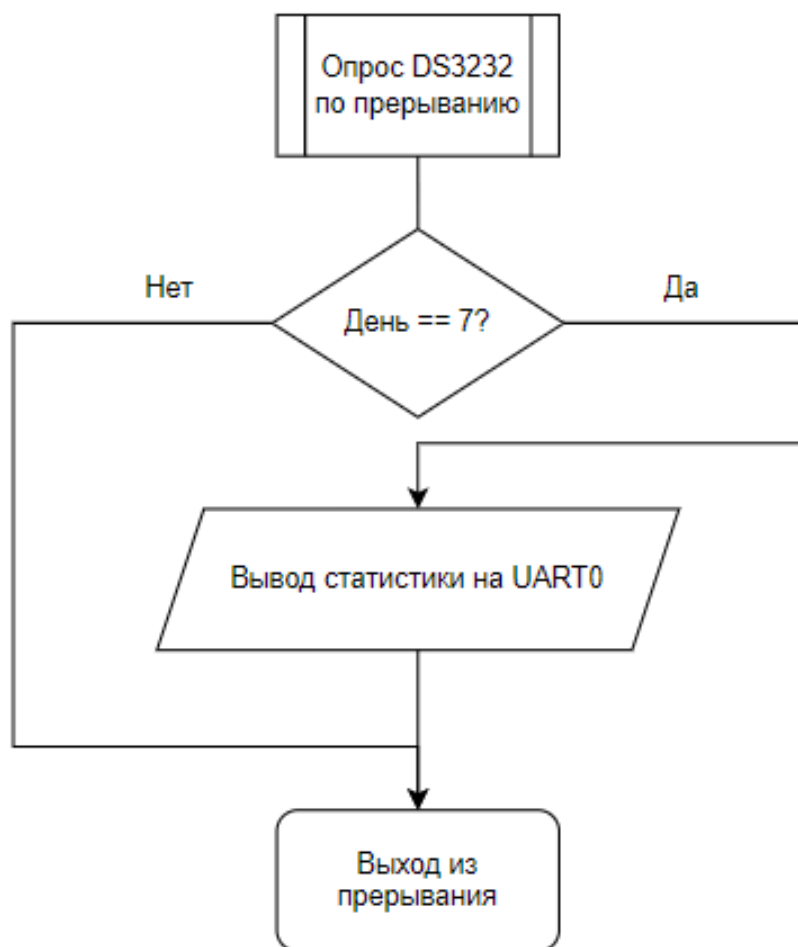


Рисунок 10 – Схема алгоритма прерывания для опроса DS3232

Схема алгоритма работы устройства с командами на Рисунок 11.

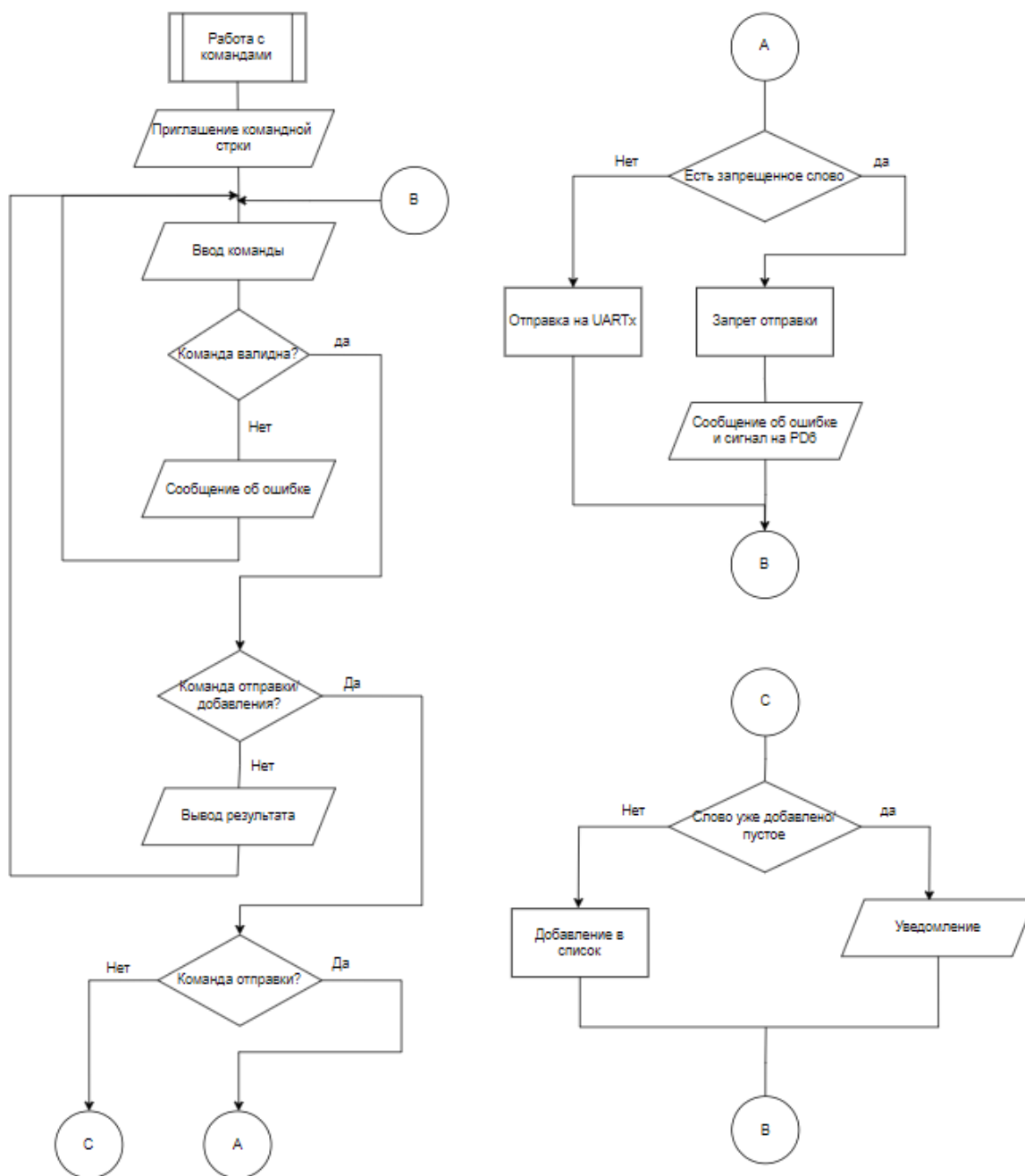


Рисунок 11 – Схема алгоритма работы с командами

На основе составленных схем алгоритмов написан программный код на языке программирования Си, который представлен в Приложении А (Исходный код программы микроконтроллера).

## 2 КОНСТРУКТОРСКАЯ ЧАСТЬ

Фильтр сетевого трафика реализован на языке программирования С. Для симуляции работы спроектированного устройства использовалось ПО Proteus. Написание программы и ее отладка осуществлялась в среде AVR Studio.

### 2.1 Настройка получение времени по DS3232

Как было описано в главе 1, взаимодействие МК и DS3232 осуществляется по интерфейсу I2C. Для обеспечения требования ТЗ к частоте отправки статистики используется прерывание TIMSK1 по переполнению. Каждую секунду прерывание срабатывает и опрашивает часы на предмет истечения недели.

Инициализация таймера происходит в пять этапов:

1. Установить в TCCR1B WGM12 = 1 – режим CTC для сброса таймера по совпадению;
2. Установить в TCCR1B значение делителя равным 1024 (CS10, CS12 = 1), так как частота = 1 МГц;
3. В OCR1A установить значение до которого будет производиться отчет, так как прерывание должно вызываться каждую секунду, значение равно 975. Приведу расчет для 1 секунды:  $OCR1A = (f_{clk} / (f_{target} * N)) - 1 = (1000 / (1 * 1024)) - 1 = 975$
4. Разрешить прерывания по совпадению: OCIE1A = 1;
5. Разрешить глобальные прерывания sei().

На приведены указанные регистры Рисунок 12.

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	OCIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 12 – Регистры 16-битного таймера

Далее был написан обработчик прерываний, инициализация таймера выполняется в основном цикле программы.

Листинг обработчика прерываний:

```
ISR(TIMER1_COMPA_vect) {  
    uint8_t day = DS3232_get_days(); // Получение дней из DS3232  
    if ((day % 7) == 0) {  
        show_stats(0); // Отправка статистики  
    }  
}
```

Параметры настройки I2C интерфейса:

1. Инициализация I2C (I2C\_init)

- TWBR (TWI Bit Rate Register) Устанавливается значение для формирования скорости передачи данных. Пример: TWBR = 0x48 для достижения 100 кГц при стандартном режиме.
- TWSR (TWI Status Register) Предделитель установлен в 1:  
TWBR = 0x00; // Биты TWPS0 и TWPS1 равны 0.
- TWCR (TWI Control Register) Включение интерфейса TWI:  
TWCR = (1 << TWEN);

2. Генерация стартового условия (I2C\_start)

- TWCR
- Генерация стартового сигнала:  
TWCR = (1 << TWSTA) | (1 << TWINT) | (1 << TWEN);

3. Отправка данных (I2C\_write)

- TWDR (TWI Data Register) Записывается байт данных (адрес устройства или значения):  
TWDR = data; // 'data' — отправляемый байт.
- TWCR  
Передача данных:  
TWCR = (1 << TWINT) | (1 << TWEN);

4. Чтение данных (I2C\_read)

- TWCR  
Считывание данных с подтверждением (ACK) или без подтверждения (NACK):
  - С подтверждением:  
TWCR = (1 << TWEA) | (1 << TWINT) | (1 << TWEN);
  - Без подтверждения:  
TWCR = (1 << TWINT) | (1 << TWEN);



## 5. Генерация стоп-условия (I2C\_stop)

- TWCR

Завершение передачи:

$$TWCR = (1 \ll TWSTO) | (1 \ll TWINT) | (1 \ll TWEN);$$

При взаимодействии с DS3232 через I2C происходят следующие операции:

- Установка регистров времени: Отправляются адреса регистров (0x00 - секунды, 0x01 - минуты, 0x02 - часы) и соответствующие данные для их записи.

Эти настройки регистров обеспечивают корректное взаимодействие через I2C в вашем проект

На Рисунок 13 изображен регистр – TWCR.

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	X	1	0	X	1	0	X

Рисунок 13 – Регистр TWCR

Работа настроенного взаимодействия приведена в разделе Тестирование устройства.

## 2.2 Настройка интерфейсов UART0 и UART1

ПЭВМ и мобильный телефон взаимодействуют с устройством по интерфейсам UART0 и UART 1 соответственно. Работа UART0 реализована в основном цикле программы, работа UART 1 реализована в прерывании, для обеспечения независимой работы подключенных устройств.

Настройка UART0:

1. Регистр UCSRB (Управление передачей/приемом):
  - RXEN0: Включает прием данных (установлено в 1);
  - TXEN0: Включает передачу данных (установлено в 1).
2. Регистр UCSRC (Настройка формата кадра):
  - URSEL: Выбор регистра UCSRC (установлено в 1);
  - UCSZ01 и UCSZ00: Установлены в 1 для конфигурации 8-битного формата данных.
3. Регистр UBRR0 (Скорость передачи):
  - Установлено значение для настройки скорости передачи в соответствии с выбранной частотой.

## Настройка UART1:

### 1. Регистр UCSRB (Управление передачей/приемом):

- RXEN1: Включает прием данных (установлено в 1);
- TXEN1: Включает передачу данных (установлено в 1);
- RXCIE1: Включаем прерывание по приему.

### 2. Регистр UCSRC (Настройка формата кадра):

- URSEL: Выбор регистра UCSRC (установлено в 1);
- UCSZ11 и UCSZ10: Установлены в 1 для конфигурации 8-битного формата данных;

### 3. Регистр UBRR1 (Скорость передачи):

- Установлено значение для настройки скорости передачи в соответствии с выбранной частотой.

Приведу расчет устанавливаемых в UBRR значений для частоты 1 МГц и скорости 1200

Бод:  $UBRR = (f_{clk} / (16 * BaudRate)) - 1 = (1000000 / (16 * 1200)) - 1 = 45$ .

На Рисунок 14 изображены настраиваемые регистры UART.

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEN	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8	
	—	—	—	—	UBRR[11:8]				UBRRHn
	UBRR[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Рисунок 14 – Настраиваемые регистры UART

## Листинг обработчика прерываний UART1

```
ISR(USART1_RX_vect) {  
    char buffer[BUFFER_SIZE];  
    UART_receive_string(buffer, BUFFER_SIZE, 1);  
    UART_send_string("USART1# ", 1);  
    to_lowercase(buffer);  
    trim(buffer);  
    if (strncmp(buffer, "send", 4) == 0) {  
        if (contains_forbidden_word(buffer + 5)) {  
            UART_send_string("Forbidden word found!\r\n", 1);  
            generate_sound();  
        } else {  
            UART_send_string(("r\nSender (USART1): "), 0);  
            UART_send_string(buffer + 5, 0);  
            UART_send_string("r\nUART0# ", 0);  
        }  
    } else if (strncmp(buffer, "add", 3) == 0) {  
        add_forbidden_word(buffer + 4, 1);  
    } else if (strncmp(buffer, "show", 4) == 0) {  
        show_forbidden_words(1);  
    } else if (strncmp(buffer, "delete", 6) == 0) {  
        delete_forbidden_words(1);  
    } else if (strncmp(buffer, "clear", 5) == 0) {  
        clear_screen(1); // Clear screen command  
    } else if (strncmp(buffer, "man", 3) == 0) {  
        show_man_page(1);  
    } else if (strncmp(buffer, "stat", 4) == 0) {  
        //show_stats(1); // Show stats  
    } else {  
        UART_send_string("Invalid command.\r\n", 1);  
    }  
}
```

Работа настроенного взаимодействия приведена в разделе Тестирование устройства.

Описание всех функций, касающихся UART:

1. `UART_init(unsigned int ubrr, unsigned char usart)`

- Описание: Инициализирует UART для заданного порта (USART0 или USART1) с указанной скоростью передачи данных (ubrr);
- Параметры:
  - ubrr — значение для настройки скорости передачи;
  - usart — выбор порта (0 для USART0, 1 для USART1).

2. `UART_send(unsigned char data, unsigned char usart)`

- Описание: Отправляет один байт данных через выбранный UART порт;
- Параметры:
  - data — данные для отправки;
  - usart — выбор порта (0 для USART0, 1 для USART1).

3. `UART_send_string(const char *str, unsigned char usart)`

- Описание: Отправляет строку через выбранный UART порт;
- Параметры:
  - str — строка для отправки;
  - usart — выбор порта (0 для USART0, 1 для USART1).

4. `UART_receive_string(char *buffer, unsigned int max_size, unsigned char usart)`

- Описание: Принимает строку через выбранный UART порт. Строка заканчивается при получении символа новой строки (\n) или при достижении максимального размера;
- Параметры:
  - buffer — указатель на буфер, в который будет записана принятая строка;
  - max\_size — максимальный размер буфера;
  - usart — выбор порта (0 для USART0, 1 для USART1).

5. `send_welcome_message(unsigned char usart)`

- Описание: Отправляет приветственное сообщение через выбранный UART порт;
- Параметры:

- o usart — выбор порта (0 для USART0, 1 для USART1).

## 6. clear\_screen(unsigned char usart)

- Описание: Очищает экран терминала, отправляя пустые строки через выбранный UART порт;
- Параметры:
  - o usart — выбор порта (0 для USART0, 1 для USART1).

Эти функции обрабатывают передачу и прием данных через UART, а также управление экраном и вывод приветственных и диагностических сообщений.

## 2.3 Отладка

Программа была отлажена в среде разработки AVR Studio, макет модели был собран в приложении Proteus 8. Proteus 8 предназначен для выполнения различных видов моделирования аналоговых и цифровых устройств.

Среда разработки AVR Studio позволяет проводить отладку проекта после программирования микроконтроллера. Отладка запускается с помощью кнопки, представленной на Рисунок – 15.

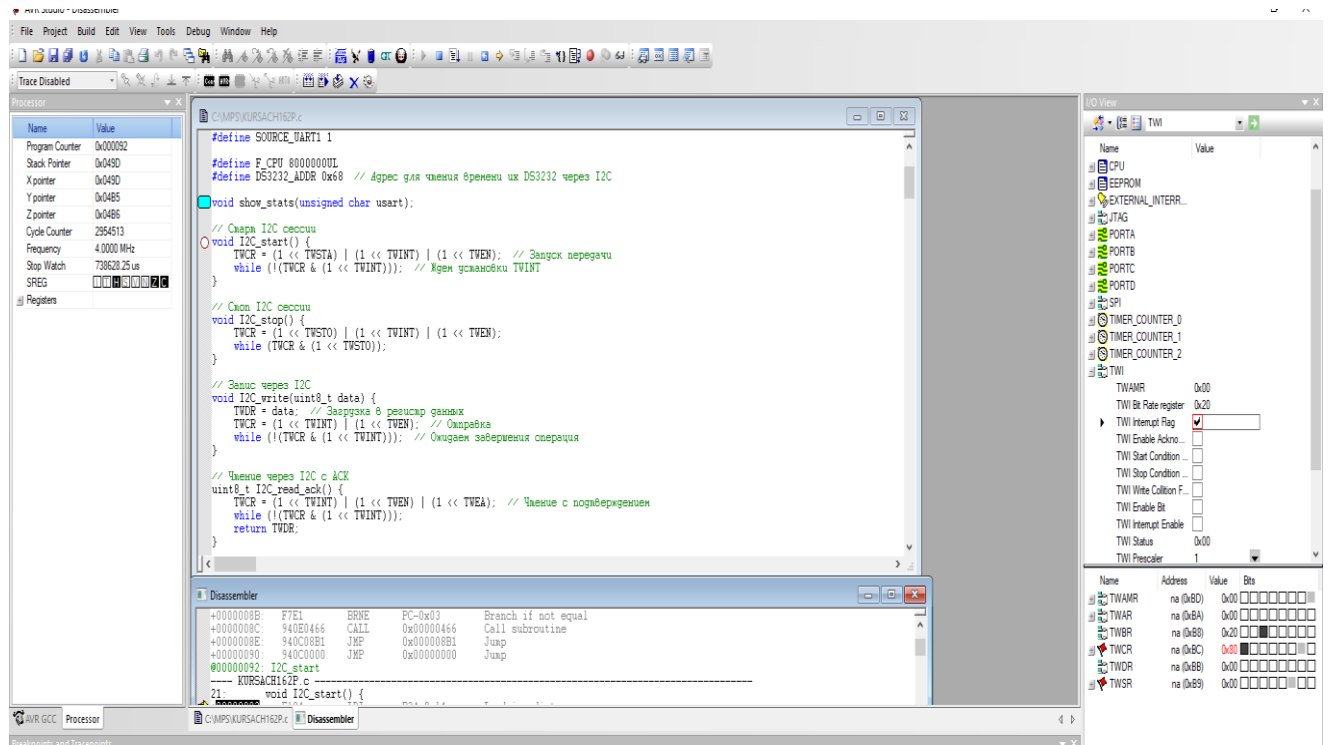


Рисунок – 15 Отладка кода в AVR Studio

## 2.4 Тестирование устройства

Выполнено тестирование по всем возможным сценариям его работы. Схема в Proteus приведена на Рисунок 16

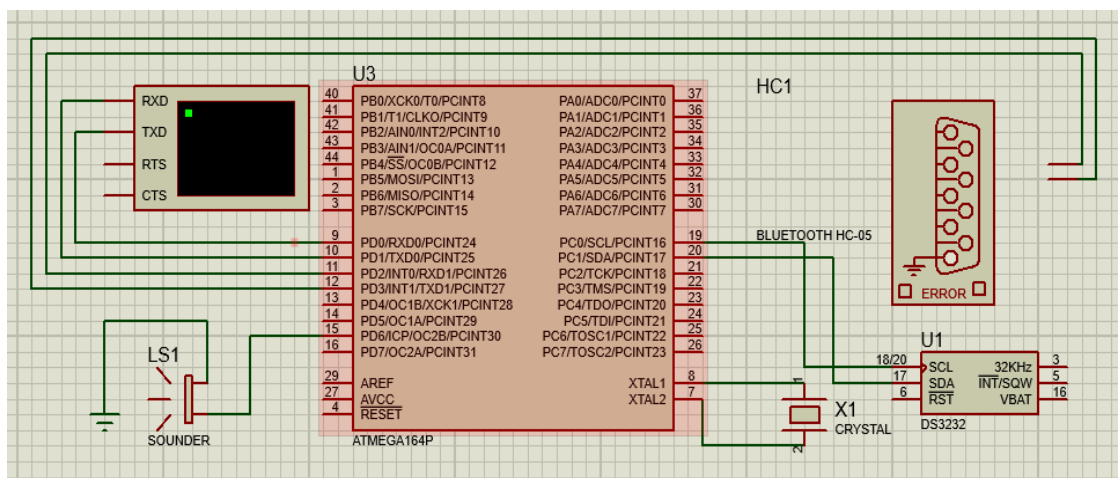


Рисунок 16 – Схема в Proteus

Устройство работает со следующими командами:

- Add – добавить слово в список запрещенных;
- Show – вывести список запрещенных слов;
- Send <string> – отправить строку;
- Delete – очистить список запрещенных слов;
- Clear – очистить окно ввода/вывода;
- Man – вывести описание команд.

При первом запуске ПЭВМ и мобильного телефона, фильтр сетевого трафика должен выводить приветственное сообщение и приглашение командной строки. На Рисунок 17 приведен результат тестирования на ПЭВМ.

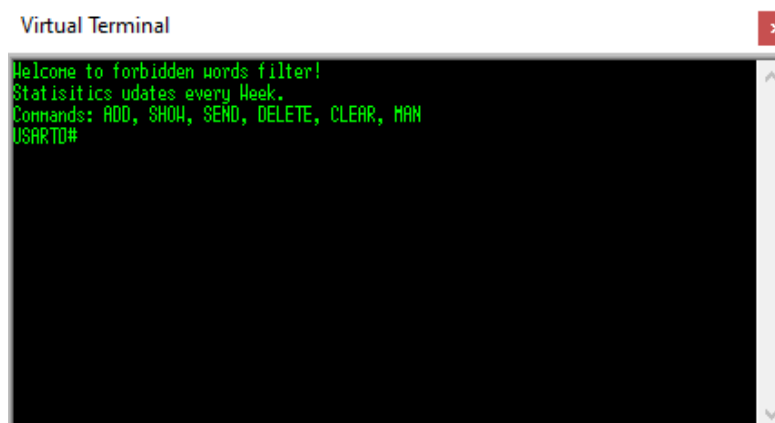


Рисунок 17 – Вывод при подключении с ПЭВМ

На Рисунок 18 приведен результат тестирования на мобильном телефоне.

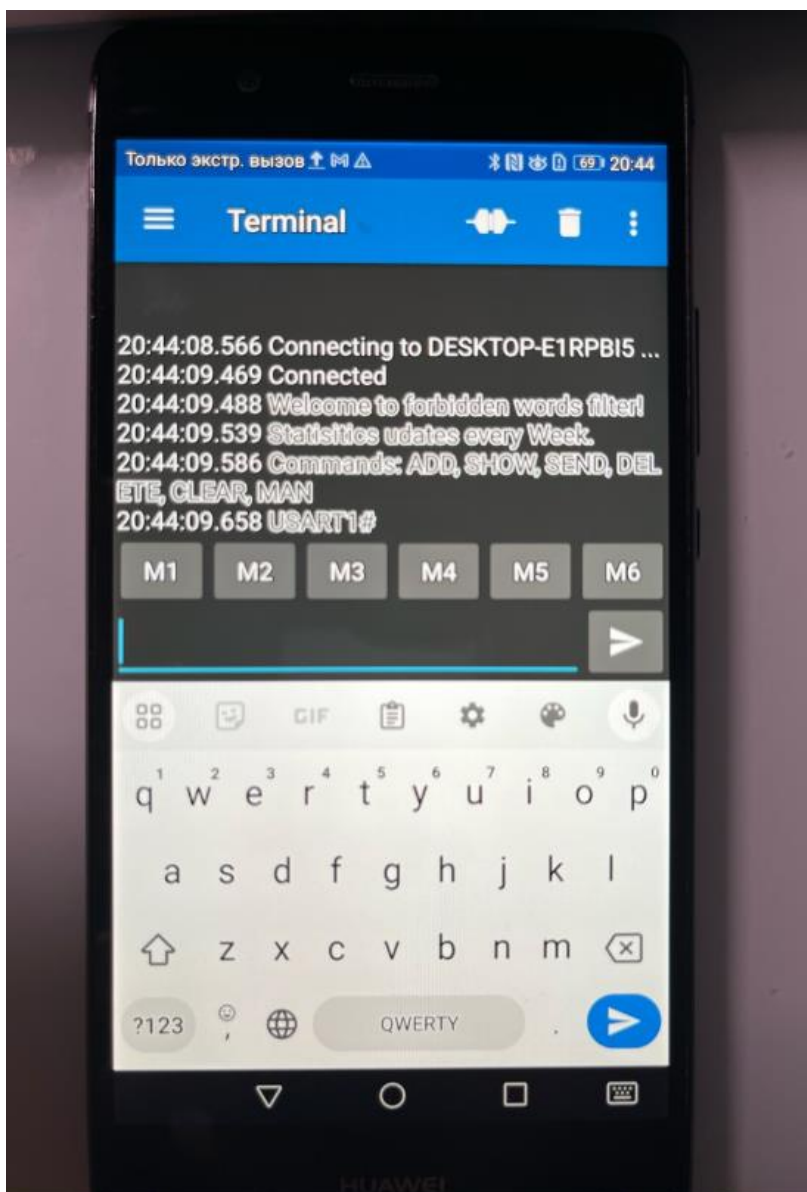
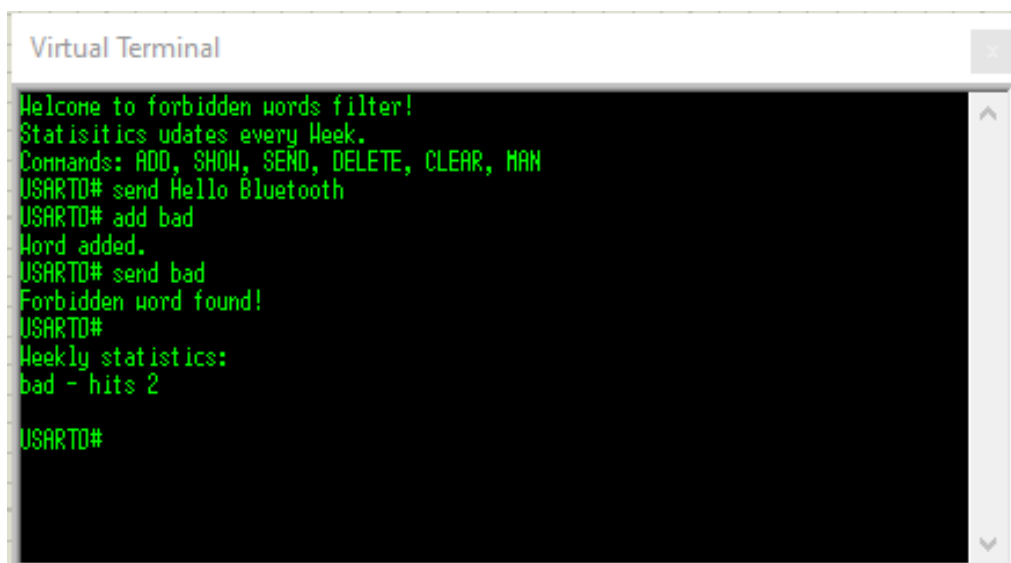


Рисунок 18 – Вывод при подключении с телефона

Далее необходимо протестировать функцию добавления запрещенных слов и отправку сообщений содержащих запрещенные слова и не содержащих запрещенные слова. Сначала с ПЭВМ отправляется строка “Hello Bluetooth!”, далее в список запрещенных слов добавляется слово “bad” и предпринимается попытка отправить строку, содержащую “bad” с обеих устройств. В результате счетчик должен иметь значение два, что должно отобразиться в присланной на ПЭВМ статистике.

На Рисунок 19 приведен результат теста на ПЭВМ. Видно, что строка с добавленным словом была заблокирована, в статистике отображается сработывание.



```
Virtual Terminal
Welcome to forbidden words filter!
Statistics updates every Week.
Commands: ADD, SHOW, SEND, DELETE, CLEAR, MAN
USART0# send Hello Bluetooth
USART0# add bad
Word added.
USART0# send bad
Forbidden word found!
USART0#
Weekly statistics:
bad - hits 2
USART0#
```

Рисунок 19 – Тестирование отправки на ПЭВМ

На приведен результат теста с телефона, видно, что сообщение, отправленное с ПЭВМ пришло, также видно сообщение о блокировке строки с запрещенным словом.

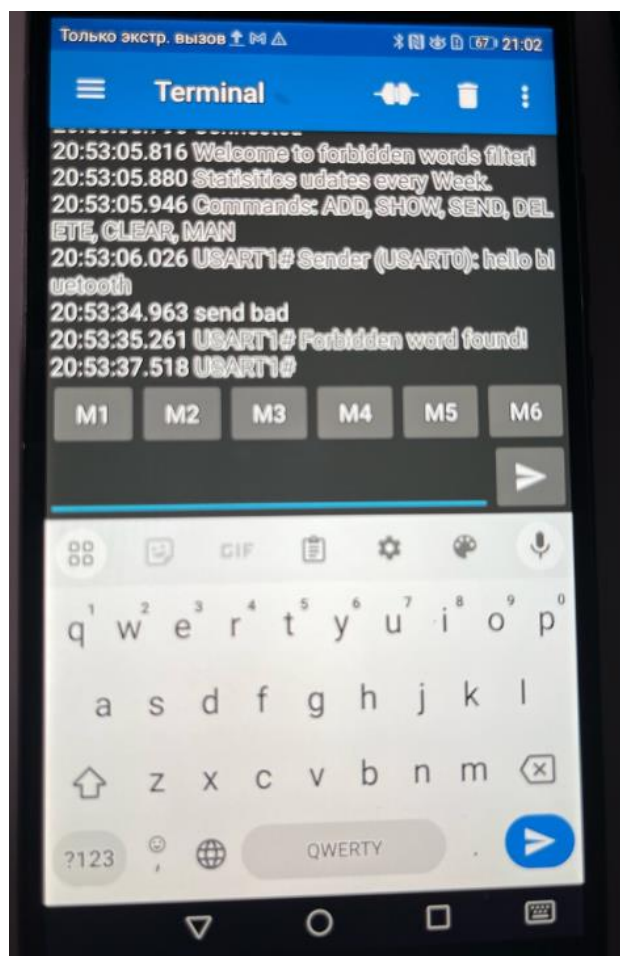


Рисунок 20 – Тестирование отправки на мобильном телефоне



Далее необходимо протестировать отправку с телефона на ПЭВМ, одно слово добавляется на телефоне, второе на ПЭВМ. Командой **show** отображается весь список.

Результат тестирования на ПЭВМ приведен на Рисунок 21. Видно, что пришло сообщение с телефона, а также в вводе **show** отображаются слова, добавленные с обоих устройств.

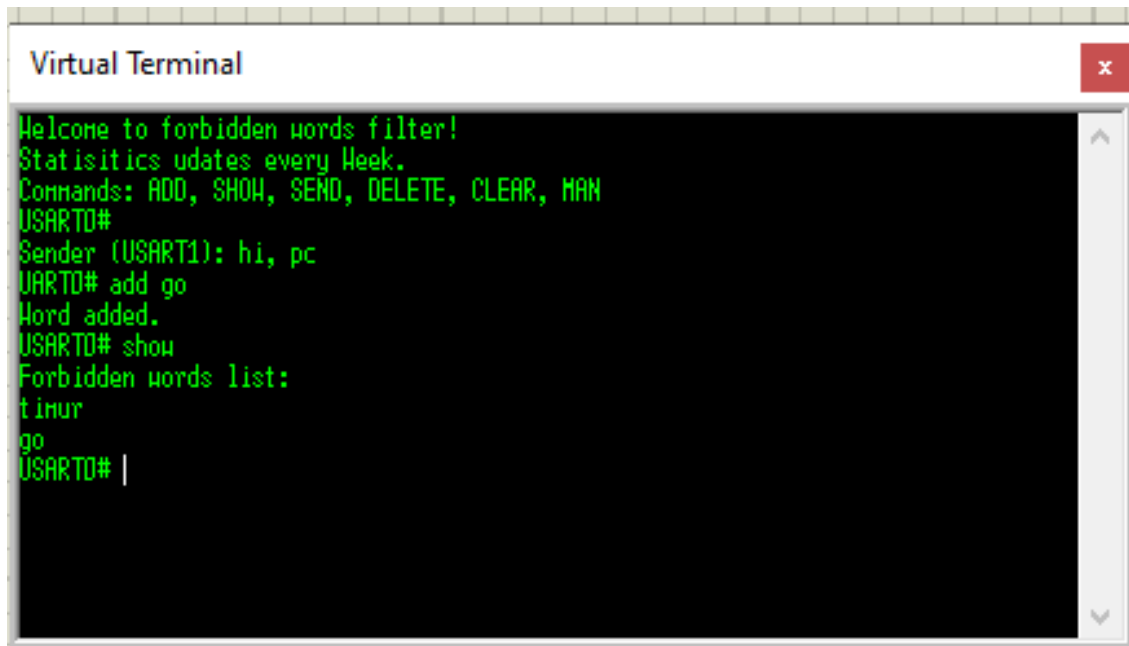


Рисунок 21 – Тестирование **show** на ПЭВМ

Результат тестирования на мобильном телефоне приведен на Рисунок 22. Все функции отрабатывают корректно.

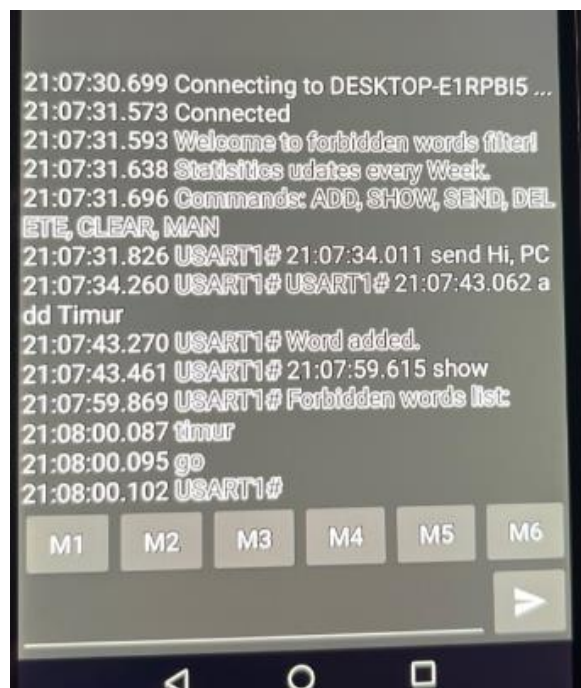
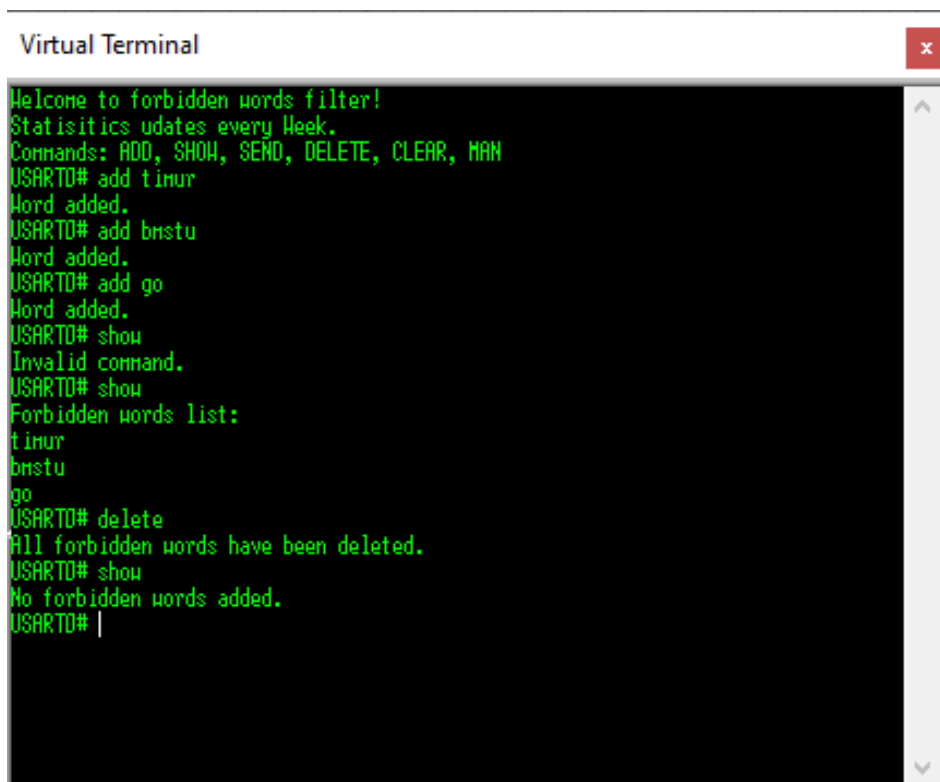


Рисунок 22 – Тестирование **show** на мобильном телефоне

Теперь необходимо протестировать очищение списка слов. Необходимо добавить несколько тестовых слов в список, отобразить их командой **show**, далее выполнить команду **delete**, а далее снова вызвать команду **show**. Тестирование выполняется на ПЭВМ, результат приведен на .

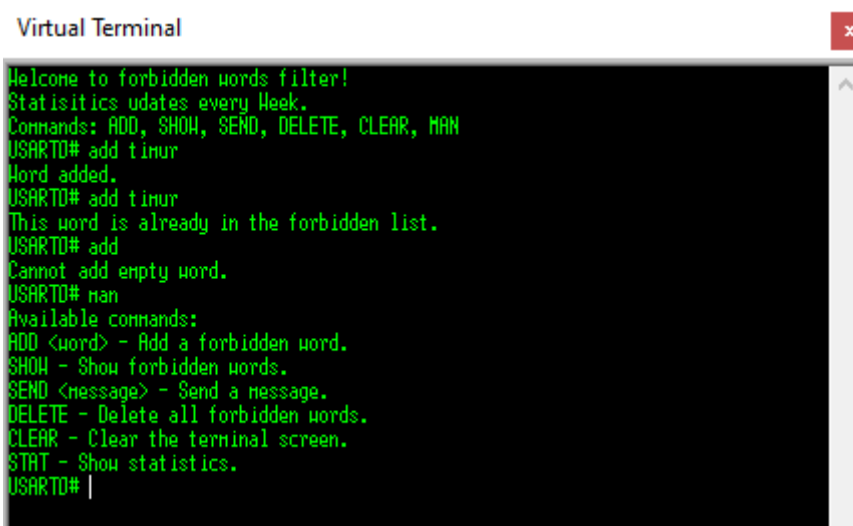
Результат тестирования приведен на Рисунок 23.



```
Virtual Terminal
Welcome to forbidden words filter!
Statistics updates every Week.
Commands: ADD, SHOW, SEND, DELETE, CLEAR, MAN
USARTO# add timur
Word added.
USARTO# add bmstu
Word added.
USARTO# add go
Word added.
USARTO# show
Invalid command.
USARTO# show
Forbidden words list:
timur
bmstu
go
USARTO# delete
All forbidden words have been deleted.
USARTO# show
No forbidden words added.
USARTO# |
```

Рисунок 23 – Тестирование delete

Последним тестом проверяется выполнение команды **man**, для вывода описания команд, а также запрет на добавление пустых и повторных слов. Результат приведен на Рисунок 24.



```
Virtual Terminal
Welcome to forbidden words filter!
Statistics updates every Week.
Commands: ADD, SHOW, SEND, DELETE, CLEAR, MAN
USARTO# add timur
Word added.
USARTO# add timur
This word is already in the forbidden list.
USARTO# add
Cannot add empty word.
USARTO# man
Available commands:
ADD <word> - Add a forbidden word.
SHOW - Show forbidden words.
SEND <message> - Send a message.
DELETE - Delete all forbidden words.
CLEAR - Clear the terminal screen.
STAT - Show statistics.
USARTO# |
```

Рисунок 24 – Проверка man

Вышеприведенные результаты тестирования показывают, что устройство функционирует правильно.

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы по дисциплине «Микропроцессорные системы» было спроектировано и реализовано устройство, выполняющее фильтрацию трафика. В процессе работы над курсовым проектом была проведена большая работа, в том числе исследовательская, которая позволила подробно описать все этапы разработки конечного устройства. Разработанное устройство полностью соответствует заданному техническому заданию. Результатом выполнения курсовой работы является полный комплект документации на объект разработки, состоящий из расчетно-пояснительной записки, электрической функциональной схемы, электрической принципиальной схемы, листинга программного кода и перечня элементов, использованных в разработанной системе.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Микроконтроллеры 8051, PIC, AVR и ARM: отличия и особенности [Электронный ресурс]. – Режим доступа: [http://digitrode.ru/computing-devices/mcu\\_cpu/1253-mikrokontrollery-8051-pic-avr-i-arm-otlichiya-i-osobennosti.html](http://digitrode.ru/computing-devices/mcu_cpu/1253-mikrokontrollery-8051-pic-avr-i-arm-otlichiya-i-osobennosti.html) (дата обращения: 13.09.2021)
2. ATmega8515 Datasheet [Электронный ресурс]. – Режим доступа: <https://static.chipdip.ru/lib/059/DOC000059786.pdf> (дата обращения: 15.09.2021)
3. ATmega8515, ATmega8515L 8-разрядный микроконтроллер с внутрисхемно программируемой флэш-памятью емкостью 8 кбайт [Электронный ресурс]. – Режим доступа: <http://www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr/atmega8515.htm> (дата обращения: 15.10.2021)
4. Устройство AVR микроконтроллера – Меандр – занимательная электроника [Электронный ресурс]. – Режим доступа: <https://meandr.org/archives/5146> (дата обращения: 09.10.2021)
5. MAX232x Dual EIA-232 Drivers/Receivers datasheet [Электронный ресурс]. – Режим доступа: <https://www.ti.com/lit/ds/symlink/max232.pdf> (дата обращения: 16.10.2021)
6. AVR. Учебный курс. Передача данных через UART [Электронный ресурс]. – Режим доступа: <http://easyelectronics.ru/avr-uchebnyj-kurs-peredacha-dannyx-cherez-uart.html> (дата обращения: 29.10.2021)
7. Подключение микроконтроллера. Ликбез. | Электроника для всех [Электронный ресурс]. – Режим доступа: <http://easyelectronics.ru/podklyuchenie-mikrokontrollera-likbez.html> (дата обращения: 27.09.2021)
8. ГОСТ 2.743-91 ЕСКД ОБОЗНАЧЕНИЯ БУКВЕННО-ЦИФРОВЫЕ В ЭЛЕКТРИЧЕСКИХ СХЕМАХ [Электронный ресурс]. – Режим доступа: <https://docs.cntd.ru/document/1200001985> (дата обращения: 05.10.2021)
9. ГОСТ 2.721-74 ЕСКД ОБОЗНАЧЕНИЯ УСЛОВНЫЕ ГРАФИЧЕСКИЕ В СХЕМАХ [Электронный ресурс]. – Режим доступа: <https://docs.cntd.ru/document/1200007058> (дата обращения: 05.10.2021)
10. Микроконтроллеры AVR: Параметры функции обработки прерываний ISR() в C [Электронный ресурс]. – Режим доступа: <http://avrprog.blogspot.com/2013/03/isrc.html> (дата обращения: 25.09.2021)
11. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих: учебное пособие. – 2-е изд., испр. и доп. – М.: Издательство: МГТУ им. Н.Э. Баумана, 2012. – 280с.

12. AVR. Учебный курс. Тракта́т о программато́рах | Электроника для всех [Электронный ресурс]. – Режим доступа: <http://easyelectronics.ru/avr-uchebnyj-kurs-traktat-o-programmatorax.html> (дата обращения: 15.11.2021)

13. Транзисторный ключ \* diodov.net | Электроника для всех [Электронный ресурс]. – Режим доступа: <https://diodov.net/tranzistornyj-klyuch/> (дата обращения: 20.11.2021)

## ПРИЛОЖЕНИЕ А

### Текст программы

```
#include <avr/io.h>

#include <util/delay.h>

#include <string.h>

#include <stdio.h>

#include <ctype.h>

#include <avr/pgmspace.h>

#include <avr/interrupt.h>

#define BUFFER_SIZE 64

#define MAX_FORBIDDEN_WORDS 10

#define WORD_SIZE 15

#define SOURCE_UART0 0

#define SOURCE_UART1 1

#define F_CPU 8000000UL

#define DS3232_ADDR 0x68 // Адрес для чтения времени их DS3232 через I2C

void show_stats(unsigned char usart);

// Старт I2C сессии

void I2C_start() {

    TWCR = (1 << TWSTA) | (1 << TWINT) | (1 << TWEN); // Запуск передачи

    while (!(TWCR & (1 << TWINT))); // Ждем установки TWINT

}

// Стоп I2C сессии

void I2C_stop() {

    TWCR = (1 << TWSTO) | (1 << TWINT) | (1 << TWEN);

    while (TWCR & (1 << TWSTO));

}
```

```

// Запис через I2C

void I2C_write(uint8_t data) {
    TWDR = data; // Загрузка в регистр данных
    TWCR = (1 << TWINT) | (1 << TWEN); // Отправка
    while (!(TWCR & (1 << TWINT))); // Ожидаем завершения операция
}

// Чтение через I2C с ACK

uint8_t I2C_read_ack() {
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA); // Чтение с подтверждением
    while (!(TWCR & (1 << TWINT)));
    return TWDR;
}

// Чтение через I2C без ACK

uint8_t I2C_read_nack() {
    TWCR = (1 << TWINT) | (1 << TWEN);
    while (!(TWCR & (1 << TWINT)));
    return TWDR;
}

// Инициализация I2C

void I2C_init() {
    TWBR = 32; // Настройка частоты
    TWSR = 0x00; // Настройка регистра состояний
}

// Чтение времени из DS3232

uint8_t DS3232_get_seconds() {
    I2C_start();
    I2C_write(DS3232_ADDR << 1);

```



```

I2C_write(0x00);

I2C_start();

I2C_write((DS3232_ADDR << 1) | 1);

uint8_t second = I2C_read_ack();

I2C_read_ack(); // минуты

I2C_read_ack(); // часы

I2C_read_nack(); // дни

I2C_stop();

return second; // нужны только секунды
}

char forbidden_words[MAX_FORBIDDEN_WORDS][WORD_SIZE];

unsigned char forbidden_word_count = 0;

unsigned int forbidden_word_hits[MAX_FORBIDDEN_WORDS] = {0};

// Зарезервированные команды

const char reserved_commands[][WORD_SIZE] PROGMEM = {

    "add", "show", "send", "delete", "clear", "man", "stat"

};

// Инициализация UARTx

void UART_init(unsigned int ubrr, unsigned char usart) {

    if (usart == 0) {

        UBRR0H = (unsigned char)(ubrr >> 8); // Скорость передачи

        UBRR0L = (unsigned char)ubrr;

        UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Включаем прием и передачу

        UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Размер данных 8 бит

    } else if (usart == 1) {

        UBRR1H = (unsigned char)(ubrr >> 8);

        UBRR1L = (unsigned char)ubrr;

```

```

    UCSR1B = (1 << RXEN1) | (1 << TXEN1) | (1 << RXCIE1); // Включаем прием и передачу
и прерывания по получению

    UCSR1C = (1 << UCSZ11) | (1 << UCSZ10);

}

}

// Отправка через UARTx

void UART_send(unsigned char data, unsigned char usart) {

    if (usart == 0) {

        while (!(UCSR0A & (1 << UDRE0)));

        UDR0 = data;

    } else if (usart == 1) {

        while (!(UCSR1A & (1 << UDRE1)));

        UDR1 = data;

    }

}

// Отправка через UARTx

void UART_send_string(const char *str, unsigned char usart) {

    while (*str) {

        UART_send(*str++, usart);

    }

}

// Получение байта через UARTx

void UART_receive_string(char *buffer, unsigned int max_size, unsigned char usart) {

    unsigned int i = 0;

    char c;

    while (i < max_size - 1) {

        if (usart == 0) {

```

```

    while (!(UCSR0A & (1 << RXC0)));

    c = UDR0;

} else if (usart == 1) {

    while (!(UCSR1A & (1 << RXC1)));

    c = UDR1;

}

if (c == '\n' || c == '\r') {

    buffer[i] = '\0';

    break;

}

buffer[i++] = c;

}

buffer[i] = '\0';

}

void to_lowercase(char *str) {

    while (*str) {

        *str = tolower(*str);

        str++;

    }

}

// Удаление пробелов

void trim(char *str) {

    while (isspace((unsigned char)*str)) {

        str++;

    }

    char *end = str + strlen(str) - 1;

    while (end > str && isspace((unsigned char)*end)) {

```

```

    end--;

}

*(end + 1) = '\0';
}

// Проверка строки
int contains_forbidden_word(char *str) {

    for (int i = 0; i < forbidden_word_count; i++) {

        if (strstr(str, forbidden_words[i]) != NULL) {

            forbidden_word_hits[i]++;

            return 1;

        }

    }

    return 0;

}

// Добавление запрещенного слова
void add_forbidden_word(char *word, unsigned char usart) {

    to_lowercase(word);

    trim(word);

    if (strlen(word) == 0) {

        UART_send_string("Cannot add empty word.\r\n", usart);

        return;

    }

    for (int i = 0; i < forbidden_word_count; i++) {

        if (strcmp(forbidden_words[i], word) == 0) {

            UART_send_string("This word is already in the forbidden list.\r\n", usart);

            return;

        }

    }

}

```

```

}

if (forbidden_word_count < MAX_FORBIDDEN_WORDS) {
    strncpy(forbidden_words[forbidden_word_count], word, WORD_SIZE - 1);
    forbidden_words[forbidden_word_count][WORD_SIZE - 1] = '\0';
    forbidden_word_hits[forbidden_word_count] = 0;
    forbidden_word_count++;
    UART_send_string("Word added.\r\n", usart);
} else {
    UART_send_string("Forbidden words list is full.\r\n", usart);
}
}

// Очищение списка
void delete_forbidden_words(unsigned char usart) {
    forbidden_word_count = 0;
    UART_send_string("All forbidden words have been deleted.\r\n", usart);
}

// Команда show
void show_forbidden_words(unsigned char usart) {
    if (forbidden_word_count == 0) {
        UART_send_string("No forbidden words added.\r\n", usart);
    } else {
        UART_send_string("Forbidden words list:\r\n", usart);
        for (int i = 0; i < forbidden_word_count; i++) {
            UART_send_string(forbidden_words[i], usart);
            UART_send_string("\r\n", usart);
        }
    }
}

```

```

    }

}

// Команда man
void show_man_page(unsigned char usart) {

    UART_send_string("Available commands:\r\n", usart);

    UART_send_string("ADD <word> - Add a forbidden word.\r\n", usart);

    UART_send_string("SHOW - Show forbidden words.\r\n", usart);

    UART_send_string("SEND <message> - Send a message.\r\n", usart);

    UART_send_string("DELETE - Delete all forbidden words.\r\n", usart);

    UART_send_string("CLEAR - Clear the terminal screen.\r\n", usart);

    UART_send_string("STAT - Show statistics.\r\n", usart);

}

// Генерация звука
void generate_sound() {

    for (int i = 0; i < 1000; i++) {

        PORTD |= (1 << PD6);

        _delay_ms(1);

        PORTD &= ~(1 << PD6);

        _delay_ms(1);

    }

}

// Отправка начального сообщения
void send_welcome_message(unsigned char usart) {

    UART_send_string("Welcome to forbidden words filter!\r\n", usart);

    UART_send_string("Statistics updates every Week.\r\n", usart);

    UART_send_string("Commands: ADD, SHOW, SEND, DELETE, CLEAR, MAN\r\n", usart);

}

```

```

// Отправка форматного сообщения

void send_message_with_label(const char *message, unsigned char usart, const char *label) {
    char formatted_message[BUFFER_SIZE];

    snprintf(formatted_message, sizeof(formatted_message), "%s: %s\r\n", label, message);

    UART_send_string(formatted_message, usart);
}

// Команда clear

void clear_screen(unsigned char usart) {
    for (int i = 0; i < 40; i++) {
        UART_send_string("\r\n", usart); // Send a space character to clear the screen
    }

    UART_send_string("\r\n", usart); // Optionally, print a newline after the space
}

// Статистика

void show_stats(unsigned char usart) {
    UART_send_string("\r\nWeekly statistics:\r\n", usart);

    for (int i = 0; i < forbidden_word_count; i++) {
        char stats[64];

        //snprintf(stats, sizeof(stats), "%s - %d hits\r\n", forbidden_words[i], forbidden_word_hits[i]);

        UART_send_string(forbidden_words[i], usart);

        UART_send_string(" - hits ",0);

        UART_send_string(itoa((forbidden_word_hits[i]),NULL,10), usart);

        UART_send_string("\r\n",0);
    }

    UART_send_string("\r\nUSART0# ", 0);
}

//Инициализация таймера

```

```

void Timer1_init() {

    // Устанавливаем режим CTC (Clear Timer on Compare Match) для таймера 1

    TCCR1B |= (1 << WGM12); // WGM12: выбираем режим CTC (Clear Timer on Compare Match),

        // когда таймер сбрасывается при совпадении с значением в OCR1A.

    // Устанавливаем предделитель на 1024 (CS12 = 1, CS10 = 1) для таймера 1

    TCCR1B |= (1 << CS12) | (1 << CS10); // CS12 и CS10 выбирают предделитель 1024.

        // Это значит, что таймер будет работать с тактовой частотой,
        деленной на 1024.

    // Устанавливаем значение для сравнения (OCR1A) — на 975, что соответствует 1 секунде

    OCR1A = 975; // OCR1A задает значение для сравнения. Когда таймер достигнет этого
    значения, он будет сброшен.

        // 975 соответствует времени 1 секунда при частоте с предделителем 1024.

    // Разрешаем прерывание по совпадению с OCR1A

    TIMSK1 |= (1 << OCIE1A); // OCIE1A: включаем прерывание по совпадению с OCR1A.

        // Это означает, что когда таймер достигнет значения в OCR1A (975), будет
        вызвано прерывание.

    // Разрешаем глобальные прерывания (включаем прерывания во всей программе)

    sei(); // Функция sei() включает глобальные прерывания, позволяя прерываниям работать.
}

//Обработка прерывания Таймера
ISR(TIMER1_COMPA_vect) {

    uint8_t second = DS3232_get_seconds();

    if ((second % 30) == 0) {

        show_stats(0);

    }

}

```



```

//Обработка прерывания USART1

ISR(USART1_RX_vect) {

    char buffer[BUFFER_SIZE];

    UART_receive_string(buffer, BUFFER_SIZE, 1);

    UART_send_string("USART1# ", 1);

    to_lowercase(buffer);

    trim(buffer);

    if (strcmp(buffer, "send", 4) == 0) {

        if (contains_forbidden_word(buffer + 5)) {

            UART_send_string("Forbidden word found!\r\n", 1);

            generate_sound();

        } else {

            UART_send_string("\r\nSender (USART1): "), 0);

            UART_send_string(buffer + 5, 0);

            UART_send_string("\r\nUART0# ", 0);

        }

    } else if (strcmp(buffer, "add", 3) == 0) {

        add_forbidden_word(buffer + 4, 1);

    } else if (strcmp(buffer, "show", 4) == 0) {

        show_forbidden_words(1);

    } else if (strcmp(buffer, "delete", 6) == 0) {

        delete_forbidden_words(1);

    } else if (strcmp(buffer, "clear", 5) == 0) {

        clear_screen(1); // Clear screen command

    } else if (strcmp(buffer, "man", 3) == 0) {

        show_man_page(1);

    } else if (strcmp(buffer, "stat", 4) == 0) {

```

```

        //show_stats(1); // Show stats

    } else {

        UART_send_string("Invalid command.\r\n", 1);

    }

}

// Main function

int main(void) {

    UART_init(51, 0); // Initialize USART0

    UART_init(51, 1); // Initialize USART1

    char buffer[BUFFER_SIZE];

    DDRD |= (1 << PD6); // Инициализация для звука

    send_welcome_message(0);

    send_welcome_message(1);

    UART_send_string("USART1# ", 1);

    Timer1_init();

    I2C_init();

    while (1) {

        UART_send_string("USART0# ", 0);

        UART_receive_string(buffer, BUFFER_SIZE, 0);

        to_lowercase(buffer);

        trim(buffer);

        if (strncmp(buffer, "send", 4) == 0) {

            if (contains_forbidden_word(buffer + 5)) {

                UART_send_string("Forbidden word found!\r\n", 0);

                generate_sound();
            }
        }
    }
}

```

```

    } else {

        send_message_with_label(buffer + 5, 1, "Sender (USART0)"); // на UART1

    }

} else if (strcmp(buffer, "add", 3) == 0) {

    add_forbidden_word(buffer + 4, 0);

} else if (strcmp(buffer, "show", 4) == 0) {

    show_forbidden_words(0);

} else if (strcmp(buffer, "delete", 6) == 0) {

    delete_forbidden_words(0);

} else if (strcmp(buffer, "clear", 5) == 0) {

    clear_screen(0);

} else if (strcmp(buffer, "man", 3) == 0) {

    show_man_page(0);

} else if (strcmp(buffer, "stat", 4) == 0) {

    //show_stats(0); // Show stats

} else {

    UART_send_string("Invalid command.\r\n", 0);

}

}

return 0;

}

```

## **ПРИЛОЖЕНИЕ Б**

Перечень элементов

На 1 листе

## **ПРИЛОЖЕНИЕ В**

Графическая часть

На 2 листах

Электрическая схема функциональная

Электрическая схема принципиальна