Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет
Кафедра радиофизики

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
канд. физ.-мат. наук, доцент
_____ В.А. Мещеряков
подпись
«_20_» ____января____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА СПЕЦИАЛИСТА (ДИПЛОМНАЯ РАБОТА)

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМЫ СБОРА КЛИМАТИЧЕСКИХ ПАРАМЕТРОВ

по основной образовательной программе подготовки специалиста
по специальности 11.05.01 «Радиоэлектронные системы и комплексы»

Хлыстов Илья Сергеевич

Руководитель ВКР
кандидат физ.-мат. наук, зав. каф. ИТИДиС
_____С.Н. Торгаев
подпись
«_18_» ____января____ 2022 г.

Автор работы
студент группы № 07609
_____ И.С. Хлыстов
подпись
«_18_» ____января____ 2022 г.

Томск – 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Радиофизический факультет

УТВЕРЖДАЮ
Руководитель ООП
к.ф.-м.н., доцент
_____ В.А. Мещеряков
«_23_» ___декабря___ 2021 г.

ЗАДАНИЕ
на подготовку ВКР специалиста
студенту Хлыстов Илье Сергеевичу группы № 07609

1. Тема ВКР: Разработка программного обеспечения для системы сбора климатических параметров.

2. Срок сдачи студентом выполненной ВКР:
а) на кафедре 20.01.2022,
б) в ГЭК 02.02.2022.

3. Краткое содержание работы:

Изучение среды программирования Microsoft Visual Studio и принципов создание баз данных. Разработка двух баз данных и интерфейса приема-передачи данных на ПК. Проведение тестовых экспериментов по обмену данными между пользовательским интерфейсом (ПК) и модулями измерения температуры и влажности.

4. Календарный график выполнения ВКР:

а) оформление литературного обзора	23.12.2021–29.12.2021
б) разработка программного обеспечения на ПК	30.12.2021–14.01.2022
в) проведение тестовых экспериментов по обмену данными между программным обеспечением и модулями измерения температуры и влажности	15.01.2022–19.01.2022
г) оформление ВКР в соответствии с требованиями	17.01.2022–20.01-2022

5. Дата выдачи задания « 23 » декабря 2021 г.

Руководитель НИР –
кандидат физ.-мат. наук,
зав. кафедры ИТИДИС _____ Торгаев С.Н.

Задание принял к исполнению _____ Хлыстов И.С.

АННОТАЦИЯ

Отчет 40 с., 4 гл., 34 рис., 1 таблица, 10 источников.

VISUAL STUDIO, ESP8266, NODEMCUV3, DHT11, СБОР КЛИМАТИЧЕСКИЙ ДАННЫХ, СЕТЬ МИКРОКОНТРОЛЛЕРОВ.

Цель работы: Разработка программного обеспечения для системы сбора климатических параметров.

Методы: программирование в интегрированной среде разработки Microsoft Visual Studio, проектирование базы данных.

В ходе выполнения ВКР были получены следующие результаты:

- изучена среда программирования Microsoft Visual Studio;
- изучены принципы проектирования баз данных и система управления базами данных SQLite;
- спроектирована и создана база данных в системе управления базами данных SQLite;
- разработан интерфейс программного обеспечения в среде Microsoft Visual Studio, который организует взаимодействие с разработанной базой данных и микроконтроллером ESP8266;
- произведено исследование работы системы в разных климатических условиях;
- произведено исследование работы системы при выходе устройств из строя.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, СОКРАЩЕНИЙ, ТЕРМИНОВ

СанПиН – санитарные правила и нормативы;

МК – микроконтроллер;

ПК – персональный компьютер;

БД – база данных;

ПО – программное обеспечение;

ФЗ – функциональная зависимость;

СУБД – система управления базами данных;

SQL – язык структурированных запросов;

MAC-адрес – от англ. Media Access Control – контроль за доступом к среде, также Hardware Address – физический адрес;

UART – универсальный асинхронный приёмопередатчик;

USB – универсальный последовательный интерфейс;

COM-порт – последовательный порт.

ВВЕДЕНИЕ

Система менеджмента качества в Национальном исследовательском Томском государственном университете, опираясь на требования СанПиН регламентирует производить замеры основных климатических показателей в учебных и лабораторных помещениях минимум два раза в сутки. Основными климатическими показателями в учебных и лабораторных аудиториях являются температура и влажность, так как эти показатели могут достаточно сильно влиять на образовательный и исследовательский процесс. Для поддержания температуры и влажности на оптимальном уровне, зачастую, в учебных помещениях устанавливаются кондиционеры и подобные им устройства.

В данной работе будет рассмотрена разработка программного обеспечения для системы сбора климатических параметров в учебной аудитории. Программное обеспечение должно обладать следующими функциями:

– обмен информацией через COM-порт с устройством сбора климатических параметров;
– запись и отображение информации из базы данных;
– добавление и удаление устройств;
– настройка системы и начало измерения.

1 Теоретическая часть

1.1 Устройство сбора климатических параметров

В данной работе под устройством сбора климатических параметров понимается МК ESP8266 на отладочной плате NodeMcu v3 с подключенным к нему датчиком температуры и влажности DHT11. Также на каждом устройстве будет идентичная программа.

С помощью плата NodeMcu v3 на основе ESP8266 можно создавать различные проекты, а благодаря Wi-Fi модулю в ESP8266 возможно создание сети устройств, взаимодействующих между собой в локальной сети, либо же для передачи информации в интернет. Основные технические характеристики NodeMcu v3 ESP8266:

- процессор – 32-бит;
- Wi-Fi – 802.11 b/g/n;
- напряжение питания – 3,3 В;
- внешнее питание – 3.6 – 20 В;
- ток потребления: режим передачи данных – 200 мА, режим приёма данных – 60 мА;
- подсоединение к компьютеру – вход microUSB;
- имеет встроенную flash память 4 Мб;
- возможность обновления прошивки по Wi-Fi.

На рисунке 1 представлен отладочный макет NodeMcu v3 на ESP8266.

Рисунок 1 – Отладочный макет NodeMcu v3 на ESP8266

Датчик DHT11 – это цифровой датчик температуры и влажности, состоящий из термистора и емкостного датчика влажности. Отлично подходит для контроля влажности в помещении. Данный датчик представлен на рисунке 2.

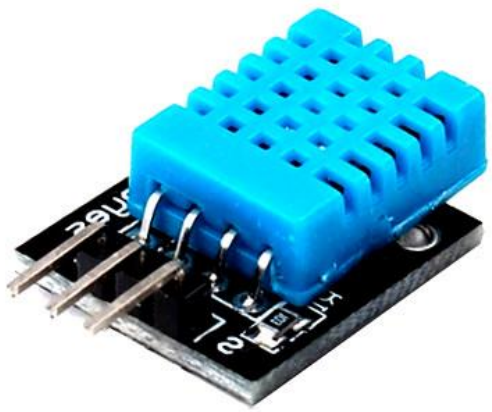


Рисунок 2 – Датчик температуры и влажности DHT11

Его выбрали исходя из того, что у него достаточно большой диапазон измерения температуры и влажности, а также он довольно дешёвый. В данной работе будем использовать датчик уже распаянный на плате вместе с подтягивающем резистором. Это уменьшит размер конечного устройства.

Основные характеристика датчика DHT11:

- питание: 3,5 – 5,5 В;
- ток питания: в режиме измерения 0.3 мА. В режиме ожидания 60 мкА;
- определение влажности 20 – 80 % с точностью 5 %;
- определение температуры 0–50 °С с точностью 2 %;
- частота опроса не более 1 Гц (не более одного раза в 1 сек.);
- размеры 15,5×12×5,5 мм.

1.2 Среда разработки Microsoft Visual Studio

Microsoft Visual Studio – это линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов [1]. Данные продукты позволяют разрабатывать

как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework и т.д.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии автодополнения, которая дописывает название функции при вводе начальных букв, и возможностью простейшего рефакторинга кода, где под рефакторингом понимается процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного, добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

Для данной работы используется версия Visual Studio Community, которая представляет из себя полнофункциональную, расширяемую и бесплатную интегрированную среду разработки для создания современных приложений Android, iOS и Windows, а также веб-приложений и облачных служб.

В Visual Studio Community для написания приложения с графическим интерфейсом будем использовать интерфейс программирования приложения Windows Forms, отвечающий за графический интерфейс пользователя.

Данный интерфейс позволяет упростить создание графической оболочки, за счёт того, что большинство необходимых функции таких как кнопки, поля для ввода информации, выпадающие списки и т.д. добавляются из панели элементов, при этом в коде автоматически описываются добавленные элементы [4].

1.3 Система управления базами данных SQLite

Для хранения и обработки полученных климатических параметров планируется использовать базу данных в системе управления базами данных SQLite.

База данных (БД) – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе [2]. База данных обычно управляется системой управления базами данных (СУБД) [3].

Современные БД являются реляционными, поэтому данные в них хранятся в виде таблиц. Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая строка таблицы представляет собой набор связанных значений, относящихся к одному объекту, или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.

В большинстве баз данных для определения данных и манипуляции с ними используется язык структурированных запросов (SQL). SQL – это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа. SQL был разработан в IBM в 1970-х годах.

SQLite – компактная встраиваемая СУБД, в которой данные представлены в виде кортежей (строк таблицы), сгруппированных в отношения (таблицы), как изображено на рисунке 14. Каждый кортеж представляется своими атрибутами. На рисунке 3 представлена графическая интерпретация отношения.

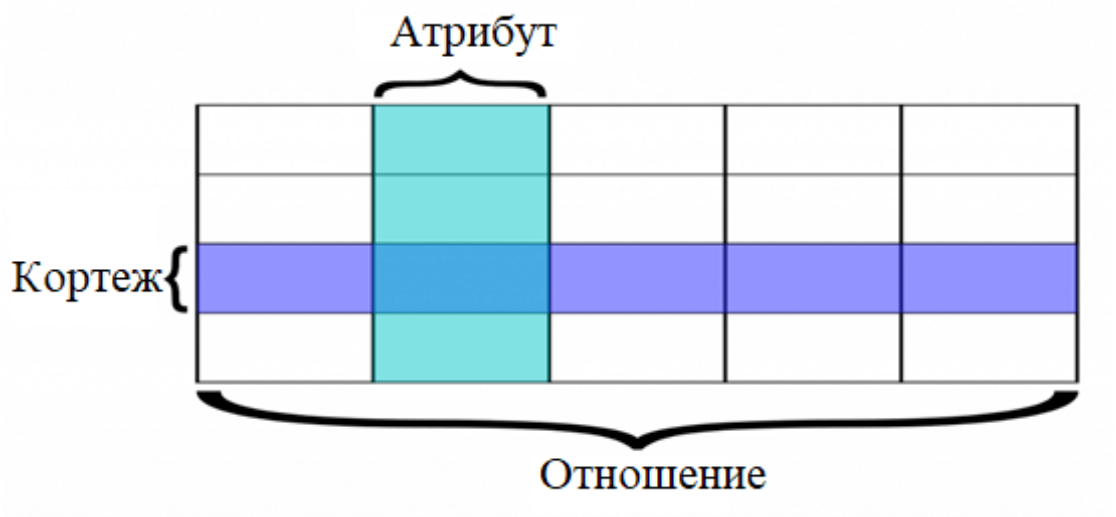


Рисунок 3 – Графическая интерпретация отношения

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. Т.е. код программы SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется.

1.4 Техника безопасности при работе с компьютером

В целях сохранения жизни и здоровья пользователя необходимо соблюдать действия, выполняемые на каждом этапе рабочего процесса.

Техника безопасности в начале работы на компьютере требует выполнения следующих операций:

– проверить исправность элементов электросистемы, обеспечивающей питание компьютера, включая электропроводку, выключатели, вилки и розетки, при помощи которых аппаратура подключается к сети;

– проконтролировать заземление компьютера;

– проверить его работоспособность.

При выполнении работ нужно соблюдать следующие правила:

– запрещается класть на корпус и дисплей компьютера посторонние предметы, прикасаться к элементам аппаратуры мокрыми руками, производить чистку корпуса оборудования, находящегося под напряжением, располагать технику близко к жилищно-коммунальным инженерным системам;

– в случае обнаружения неисправности компьютера немедленно прекратить работу и сообщить об этом непосредственному руководителю;

– эксплуатировать компьютер только с соблюдением инструкции, установленной производителем;

– избегать частого и необоснованного включения и выключения компьютера во время работы.

После завершения работы польз нужно выполнить следующие действия:

– выключить компьютер с использованием алгоритма, установленного производителем;

– обесточить периферийное оборудование;

– убедиться в отключении техники;

– выполнить очистку рабочих поверхностей влажной тканью.

Требования к расположению пользователя за компьютером нацелены на обеспечение его комфорта в течение всего времени работе на компьютере и отсутствие негативных следствий длительной работы. Они включают следующие правила:

– полная опора ступнями на пол при посадке;

– использование компьютерной мебели, отвечающей нормам СанПиН 2.2.2/2.4.1340-03;

– отказ от скрещивания конечностей, способного затруднить кровообращение;

– соблюдение расстояние до монитора компьютера не меньше 45 сантиметров;

– правильная установка освещения, которое не должно светить в глаза и оставлять блики на рабочем мониторе.

Нарушения техники безопасности при работе на компьютере способны вызвать стойкие расстройства здоровья, которые потом будет трудно ликвидировать.

2 Разработка системы и программного обеспечения

2.1 Принцип работы системы автоматизированного сбора климатических параметров

Система автоматизированного сбора климатических параметров в учебных помещениях состоит из N устройств. Каждое устройство представляет из себя МК ESP8266 с подключенным к нему датчиком температуры и влажности DHT11. Первое устройство будет идентично всем остальным, но в добавок ко всему будет связано со стационарным ПК, на который в конечном счёте будут приходить все собранные климатические данные. На рисунке 4 представлена структурная схема системы сбора климатических параметров.

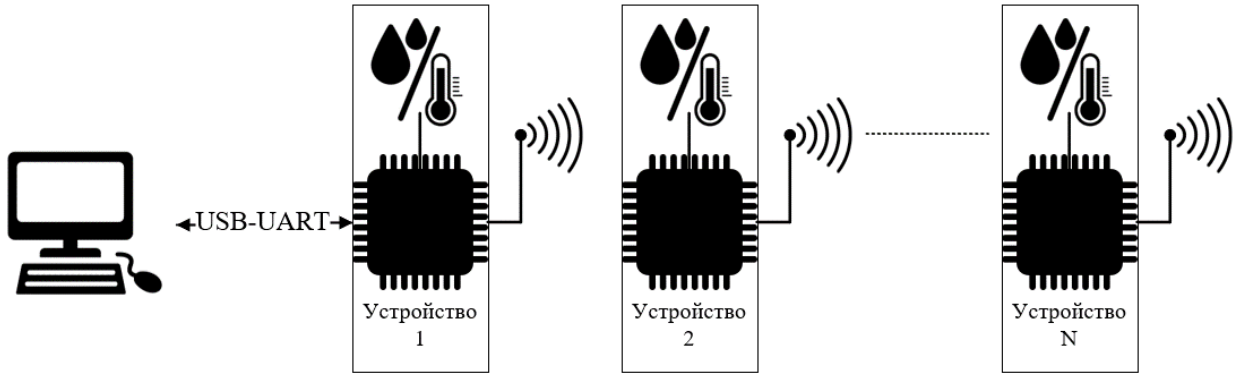


Рисунок 4 – Структурная схема системы сбора и обработки климатических параметров в учебных помещениях

На первом этапе на первое устройство с ПК подаётся сигнал начала сбора климатических параметров. Устройство формирует запрос и через Wi-Fi модуль передаёт его на следующее устройство. Передача запроса повторяется пока запрос не достигнет последнего устройства.

На втором этапе последнее устройство производит сбор климатических параметров с помощью датчика температуры и влажности DHT11, формирует эти данные в пакет и отправляет на предыдущее устройство. Приняв пакет данных, устройство производит сбор климатических параметров, добавляет эти данные в принятый пакет и производит передачу на очередное устройство.

Эта процедура повторяется до тех пор, пока данные не придут на первое устройство. Оно также проводит сбор климатических параметров и добавляет их к принятым данным. Завершается этап передачей данных на ПК.

2.2 Функциональные требования к программному обеспечению

Разрабатываемое программное обеспечение входит в состав системы, выполняющей функции сбора климатических параметров в учебных аудиториях. Программное обеспечение должно быть интуитивно понятным и обладать следующими функциями:

- отправка и получение запросов микроконтроллеру по COM-порту;
- сбор и систематизация информации, полученной от микроконтроллера;
- отправка полученной информации в базу данных;
- отображение информации из базы данных.

2.3 Структура пакета данных

Для взаимодействия ПО с системой сбора климатических параметров необходимы команды по котором МК будет понимать порядок своих действий.

Для настройки системы предусмотрен следующая команда:

1, mac1, mac2, …, macN;,

где 1 – команда для МК для настройки системы; mac1, mac2, …, macN – Mac-адреса в последовательности, в которой будет происходить обмен данными между устройствами; и символ «;» – который является символом конца строки. Под настройкой системы понимается запись Mac-адресов в память микроконтроллеров.

Для вывода в COM-порт всей таблице Mac-адресов реализована команда «2;». Следующая команда «3;». По ней МК начинает измерения температуры и влажности и передаёт эту команду последовательно на следующее устройство в соответствии с таблицей Mac-адрес. После этого в обратном направлении формируется пакет данных, в которые входят температуры и влажности с каждого устройства. Ответ от МК выглядит следующим образом:

«t1, h1, t2, h2, …, tn, hn;»,

где t – температура; h – влажность. После полученный пакет данных разбиваются на отдельные переменные в ПО.

Последняя команда «4;» реализована для добавления нового устройства в таблицу Mac-адресов. По этому запросу МК отправляет свой Mac-адрес в COM-порт

2.4 Расчёт стоимости устройства

Для расчета стоимости обратились к наиболее популярным магазинам как в Томске, так и в России. В таблице 1 приведены собранные данные по ценам в выбранных магазинах на момент декабря 2021 года.

Таблица 1. Сравнение цен в наиболее популярных магазинах.

Магазин / Товар	Chipdip, руб	Амперка, руб	Озон, руб	AliExpress, руб	Эльград, руб
DHT11	690	310	369	60	185
NodeMCUv3	1950	___	1250/3	250	___
Battery Shield	___	___	353	154	448
Модуль RGB-светодиода	150	60*	300	103	15*

* - цена за светодиод

Анализируя вышеуказанную таблицу, получаем, что стоимость компонентов, необходимых сбора для одного устройства варьируется от 570 до 3390 рублей. Важно заметить, что эта стоимость не учитывает траты на дополнительное коммутационное оборудование (наборы проводников), изготовление печатной платы, печать корпуса устройства и транспортные издержки.

3 Практическая реализация

3.1 Общий принцип создания программного обеспечения

Для хранения и систематизации климатических параметров в среде программирования Visual Studio была разработана программа (Приложение А). Обмен данными между ПК и МК происходит через COM-порт. Код программы написан на языке программирования C++ [5]. На рисунке 5 представлена рабочая область среды программирования Visual Studio.

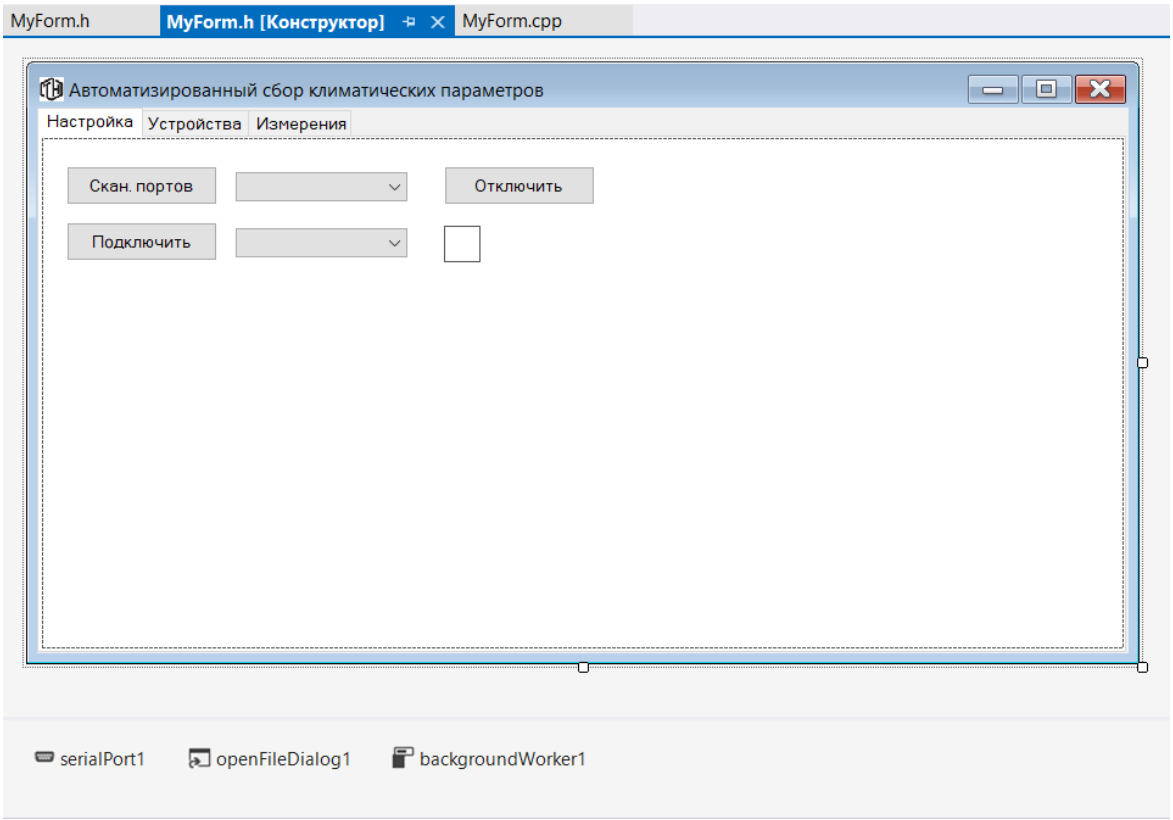


Рисунок 5 – Рабочая область Visual Studio

В Visual Studio для создания интерфейса программы используется 2 вкладки:

– Первая, MyForm.h [Конструктор] – для разработки графического интерфейса пользователя.
– Вторая, MyForm.h – для написания кода самой программы.

3.2 Настройка подключения микроконтроллера по COM-порту

Далее была разработана программа, реализующая подключение к МК через COM-порт (рисунок 6), с возможностью выбора скорости передачи, закрытия COM-порта, визуализацией подключения к МК, и полем для передачи и приёма информации от МК. А также, для работы с базой данных, разработанной в СУБД SQLite.

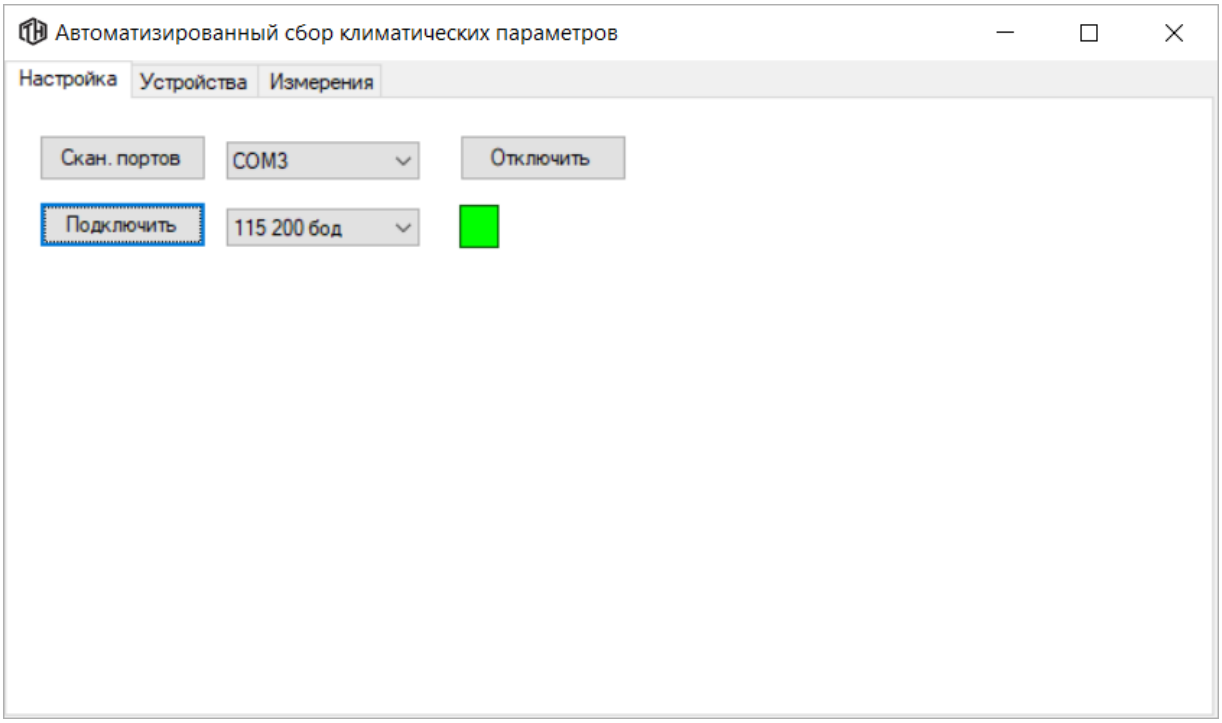


Рисунок 6 – Лицевая панель программы – вкладка «Настройка»

Данная программа работает в следующей последовательности. По нажатию кнопки «Сканирование портов» происходит сканирование всех активных портов в Windows и добавление их в выпадающий список, при этом автоматически выбирается первый доступный COM-порт в выпадающем списке и также автоматически выбирается скорость передачи 115200 бод. Автоматический выбор сделан для того, чтобы при попытке подключения программа не завершала работу с ошибкой.

По нажатию кнопки «Подключить» происходит проверка выбран ли COM-порт, в противном случае отобразится окно с ошибкой. Пример с ошибкой показан на рисунке 7.

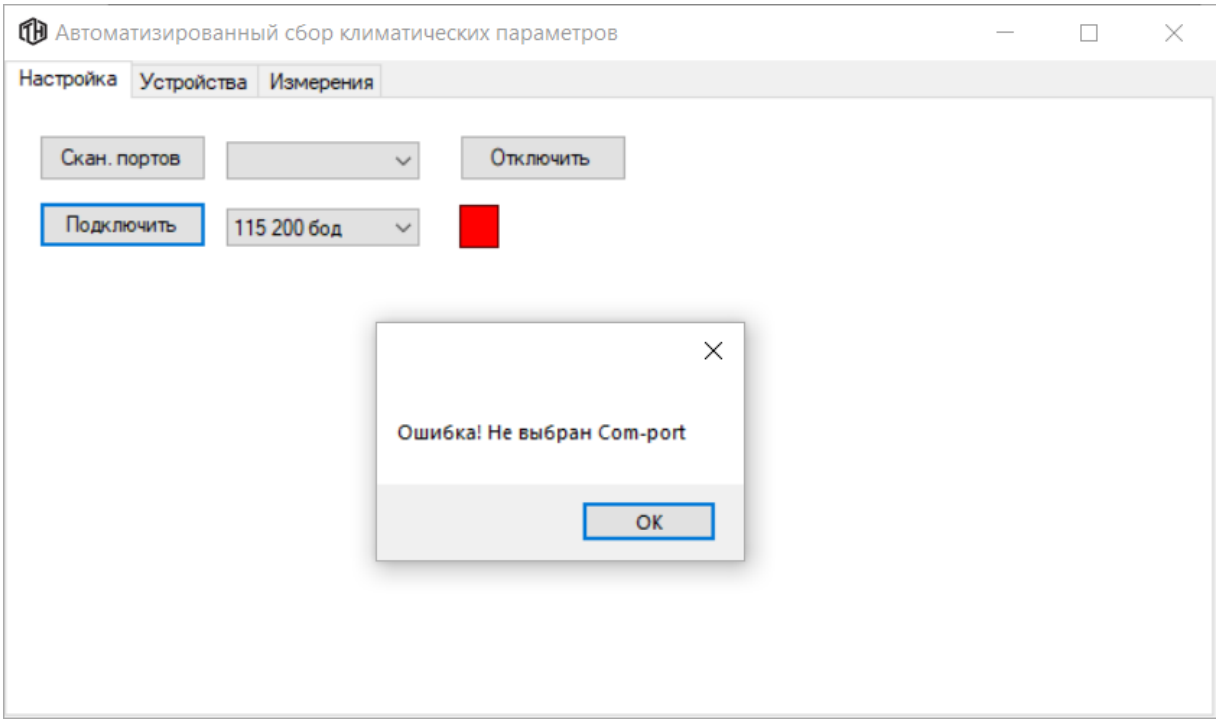


Рисунок 7 – Пример ошибочного подключения к МК

Также выполняется настройка соединения через COM-порт с заданной скоростью. И происходит попытка отрыть COM-порт. [6] В случае, если его удалось открыть, загорается зелёный светодиод, в противном случае загорается красный светодиод и создается окно, описывающее ошибку подключения.

По нажатию кнопки «Отключить» происходит закрытие COM-порта и загорается красный светодиод. [6]

3.3 Проектирование базы данных

Перед проектированием БД необходимо определить структуру данных для реализации в БД и соответствующую модель данных.

В базе данных планируется хранить информацию о устройствах (id_d (номер устройства), MAC-адрес и номер комнаты, в которой находится устройство) и результаты измерений (номер измерения, дата и время измерения и значения температуры и влажности). Т.е. в проектируемой базе данных будет две сущности, устройства и измерения, о которых необходимо хранить данные.

Сущности обладают свойствами, называемыми атрибутами. Например, сущность устройство обладает атрибутами id_d, MAC-адрес, номер комнаты.

Каждый атрибут принимает значение из некоторого множества значений, характерного для данного атрибута, называемого доменом. Например, атрибут номер комнаты принимает значение из множества чисел используемых для нумерации комнат.

Множество атрибутов (или один атрибут), уникально идентифицирующее объект в наборе объектов, называется ключом этого набора объектов. Например, для объекта измерения, ключом будет номер измерения.

Для данной работы была рассмотрена модель объектов-связей (ER-модель), которая представлена на рисунке 8. ER-модель является основным этапом при проектировании БД. Она является моделью представления БД на внешнем (пользовательском) уровне.

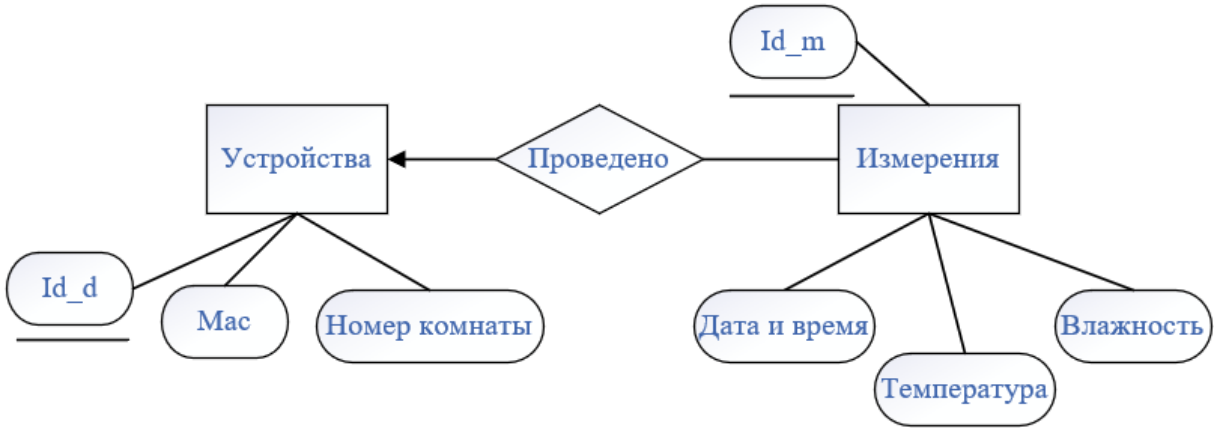


Рисунок 8 – ER-диаграмма

На данной диаграмме представлено 3 таблицы:

1. Устройства;
2. Измерения;
3. Проведено – связующая таблица.

У сущности устройства есть три объекта id_d, MAC-адрес и номер комнаты, при этом номер устройства (id_d) является ключевым объектом, что показано подчёркиванием на диаграмме. А у сущности измерения следующий

объекты: номер измерения (id_m), являющийся ключевым, дата и время измерения, измеренные температура и влажность.

В связи с тем, что связь сущностей, описывается как много измерения на одном устройстве, или же, что на одном устройстве можно провести много измерений, то таблицу связей можно упростить, при этом ER-диаграмму можно представить следующим образом, как представлено на рисунке 9.

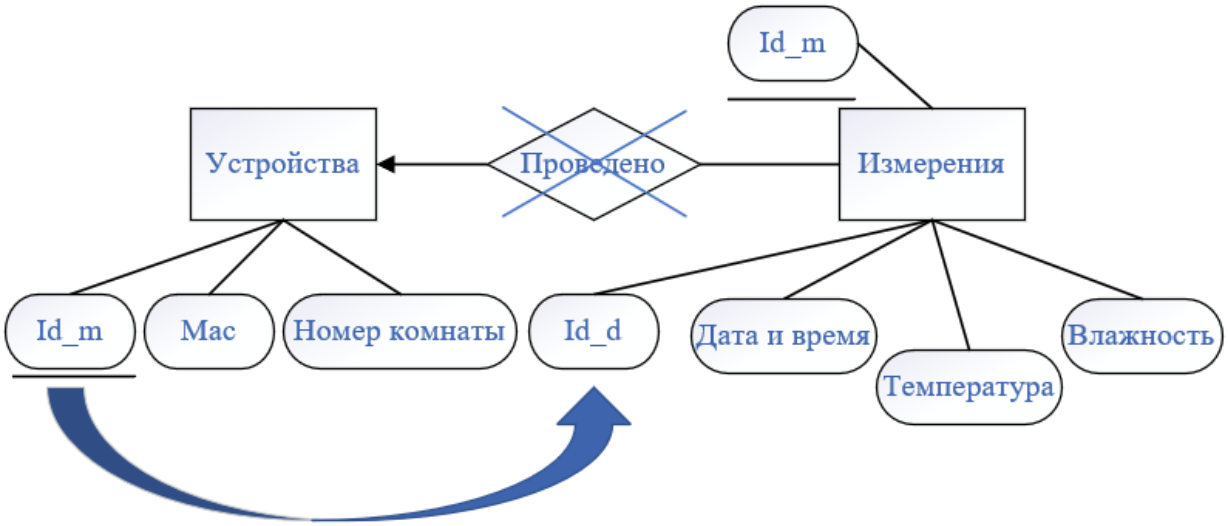


Рисунок 9 – ER-диаграмма с упрощением таблицы связей

После упрощения таблицы связей, чтобы связь сущностей осталась, нужно ключевой атрибут из сущности «Устройства» добавить в атрибуты, сущности «Измерения», что и представлено на рисунке 9.

Далее рассмотрим схему отношений для проектируемой базы данных. Она будет выглядеть следующем образом:

– Устройства (Id-d, Mac-адрес, номер комнаты);
– Измерения (Id-m, дата и время, температура, влажность, MAC-адрес).

Функциональная зависимость – это формальный способ задания ограничений целостности. Для данной схемы отношений, ФЗ следующая:

- {Id-d}→{Mac-адрес, номер комнаты};
- {Id-m}→{дата и время, температур, влажность, MAC-адрес }.

Полученные ФЗ имеют третью нормальную форму (3NF). Это значит, что всякий не ключевой атрибут функционально определяется только ключом (и никакими другими атрибутами), т.е. атрибуты независимы.

3.4 Создание базы данных в системе управления базами данных SQLite

СУБД SQLite – это библиотека на языке С, которая реализует небольшой, быстрый, автономный, высоконадежный, полнофункциональный механизм базы данных SQL.

Создание БД в SQLite сделано с помощью графического интерфейса, который в разы упрощает процесс разработки баз данных.

При создании БД в СУБД SQLite, будем отталкиваться от спроектированной БД в пункте 3.3. Сразу после создания БД, SQLite открывает окно с создание первой таблицы (сущности), которая представлена на рисунке 10.

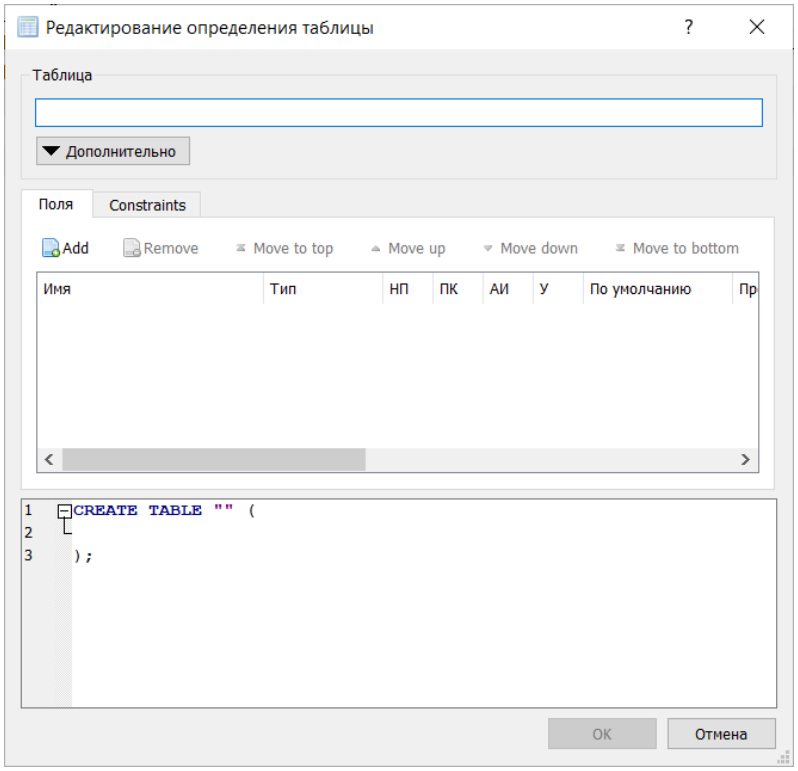


Рисунок 10 – Пример пустой таблицы в SQLite

После этого создадим таблицы, исходя из спроектированной базы данных. Созданные таблицы представлены на рисунке 11 и 12.

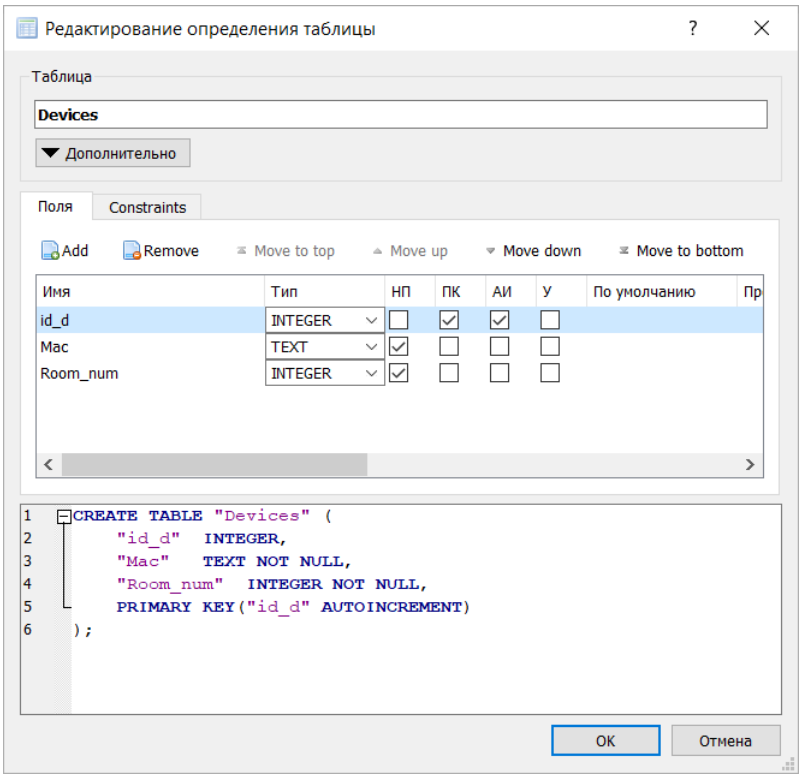


Рисунок 11 – Создание таблицы «Устройства»

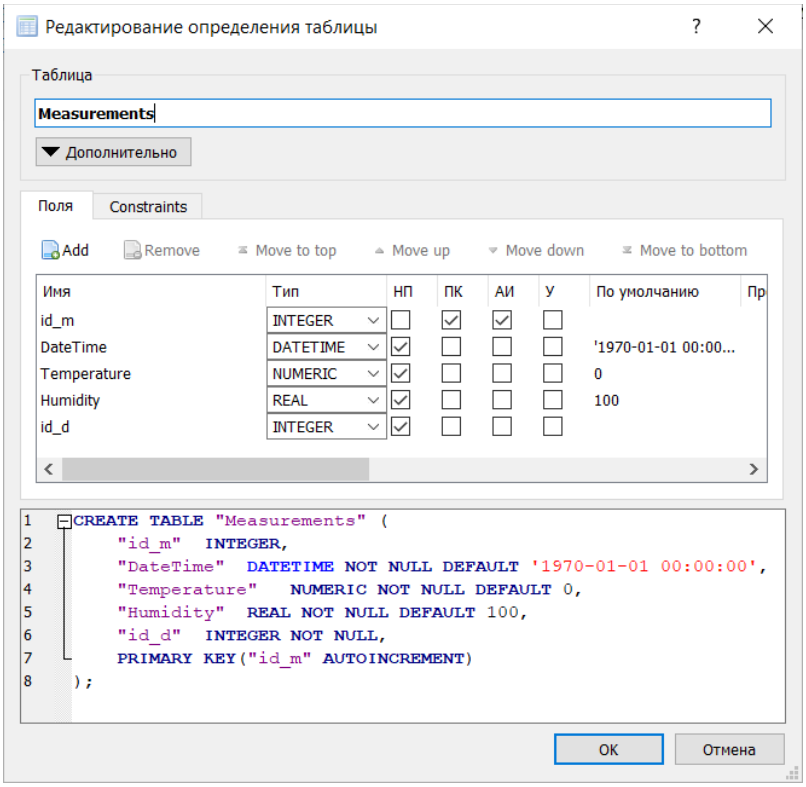


Рисунок 12 – Создание таблицы «Измерения»

Как видно из рисунков 11 и 12 SQLite автоматически генерирует команду на создание таблицы. Это видно в нижней части окна.

Изначально таблицы создаются пустыми, что продемонстрировано на рисунке 13.

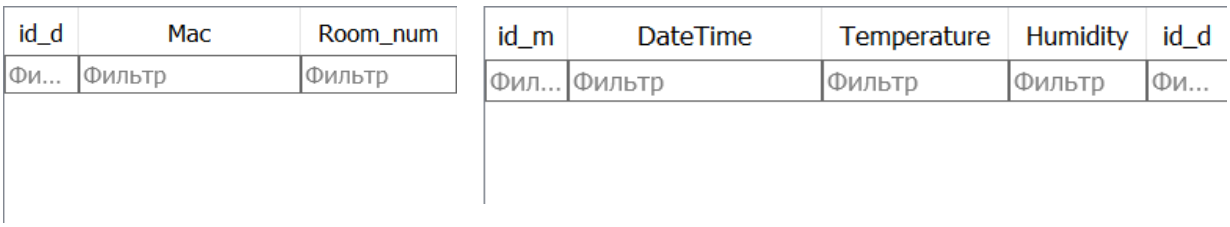


Рисунок 13 – Пустые таблицы «Устройства» и «Измерения»

После создания таблиц мы можем заполнить их вручную, либо как планируется в данной работе заполнить их из основной программы, к которой будет подключена данная база данных.

3.5 Реализация взаимодействия программного обеспечения с системой управления базами данных SQLite

Для работы с СУБД была сделана программа, а именно две вкладки, отвечающие за две таблицы – «Устройства» и «Измерения». Данные вкладки изображены на рисунке 14 и 15.

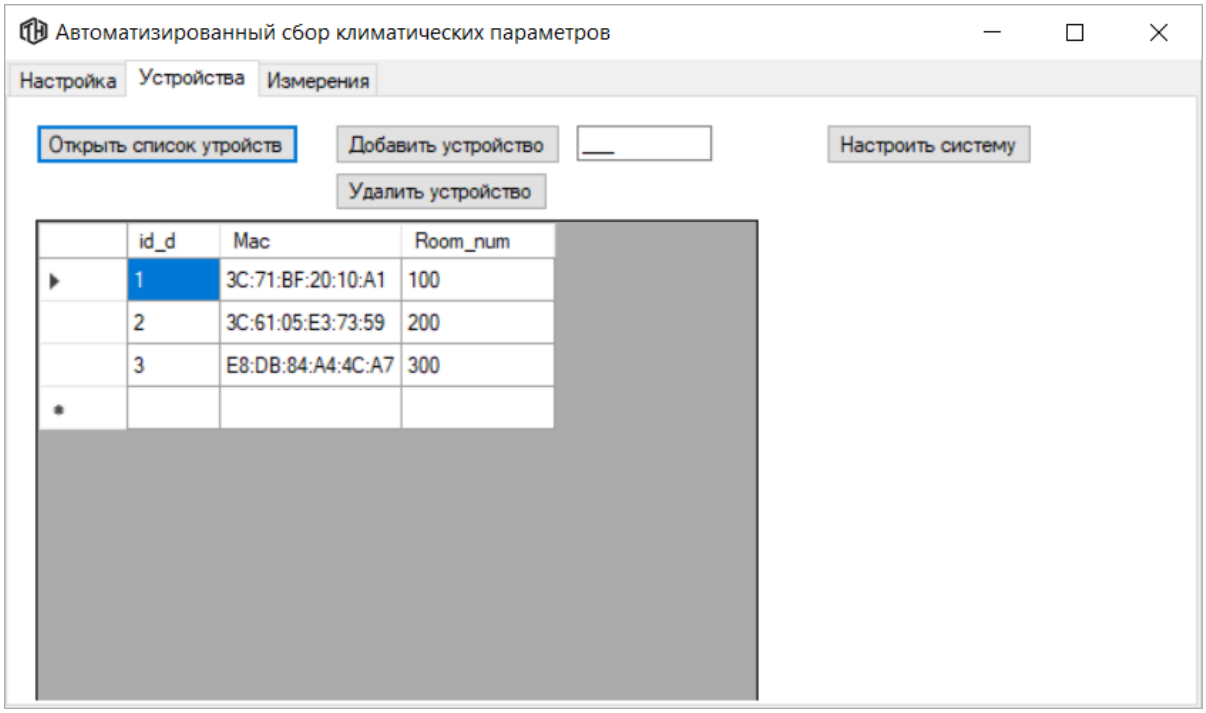


Рисунок 14 – Лицевая панель программы – вкладка «Устройства»

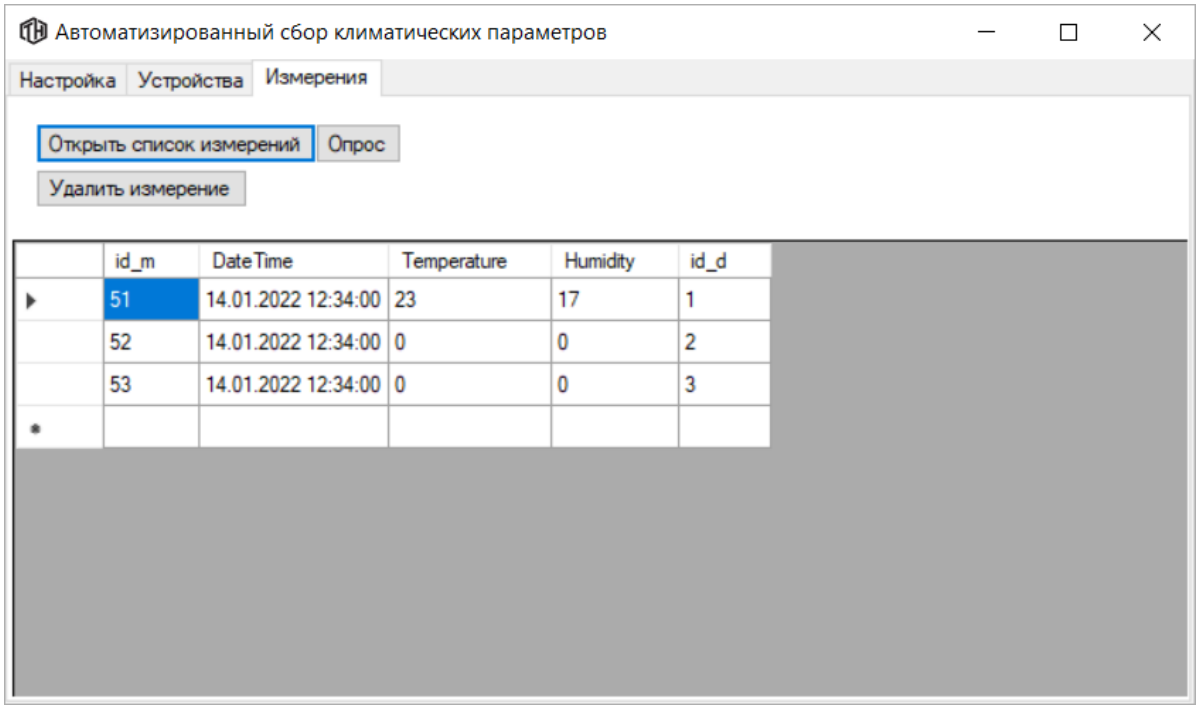


Рисунок 15 – Лицевая панель программы – вкладка «Измерения»

Программа на вкладке «Устройства» работает следующим образом. По нажатию кнопки открыть список устройств открывается стандартное диалоговое окно, предлагающее пользователю открыть файл с базой данных. Пример окна представлен на рисунке 16.

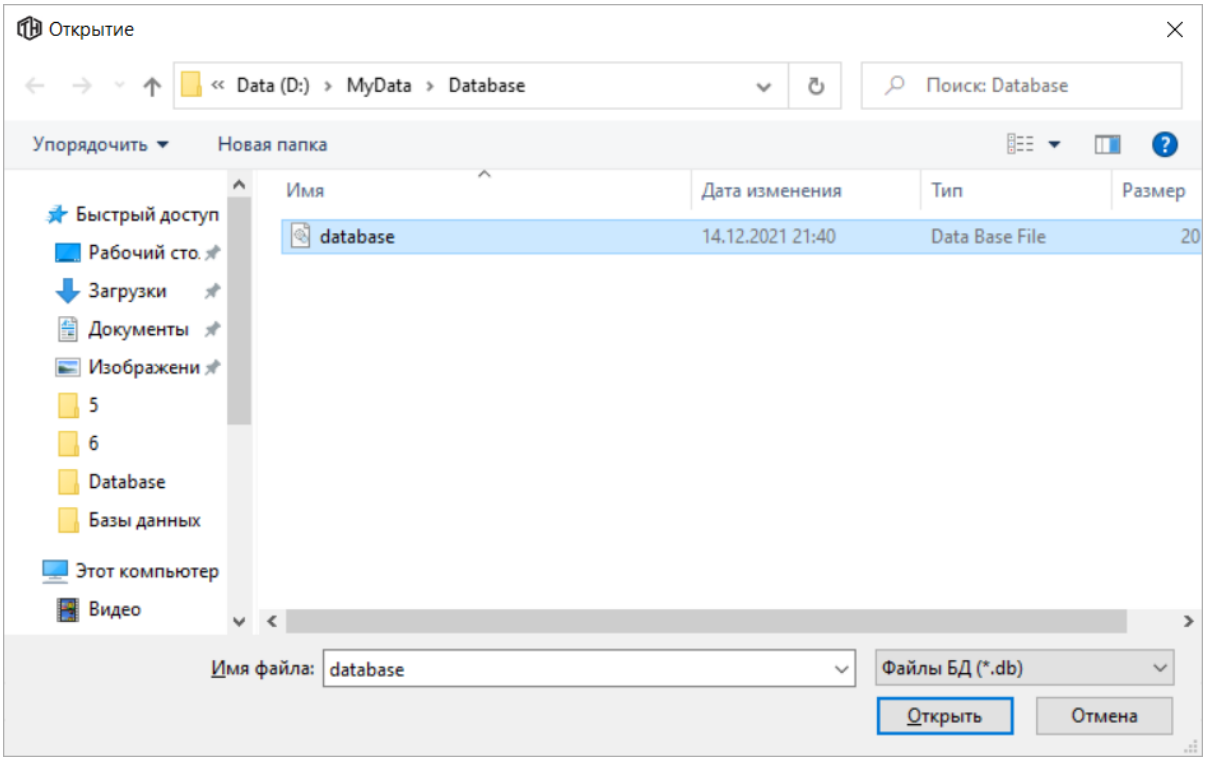


Рисунок 16 – Пример диалогового окна

После выбора файла, содержащего базу данных, программа создаёт запрос в СУБД SQLite на открытие данной БД и создаёт таблицу в программе на основе таблицы «Устройства» из БД.

Следующая кнопка «Добавить устройство» отправляет МК команду «4», по которой МК понимает, что ему нужно отправить в ответ свой Mac-адрес, далее, в соответствии с выбранным номером комнаты из поля для ввода текста, Mac-адрес и номер комнаты добавляются в таблицу «Устройства» и после повторно происходит ее отображение в программе в поле для таблиц. Пример добавления устройства представлен на рисунке 17.

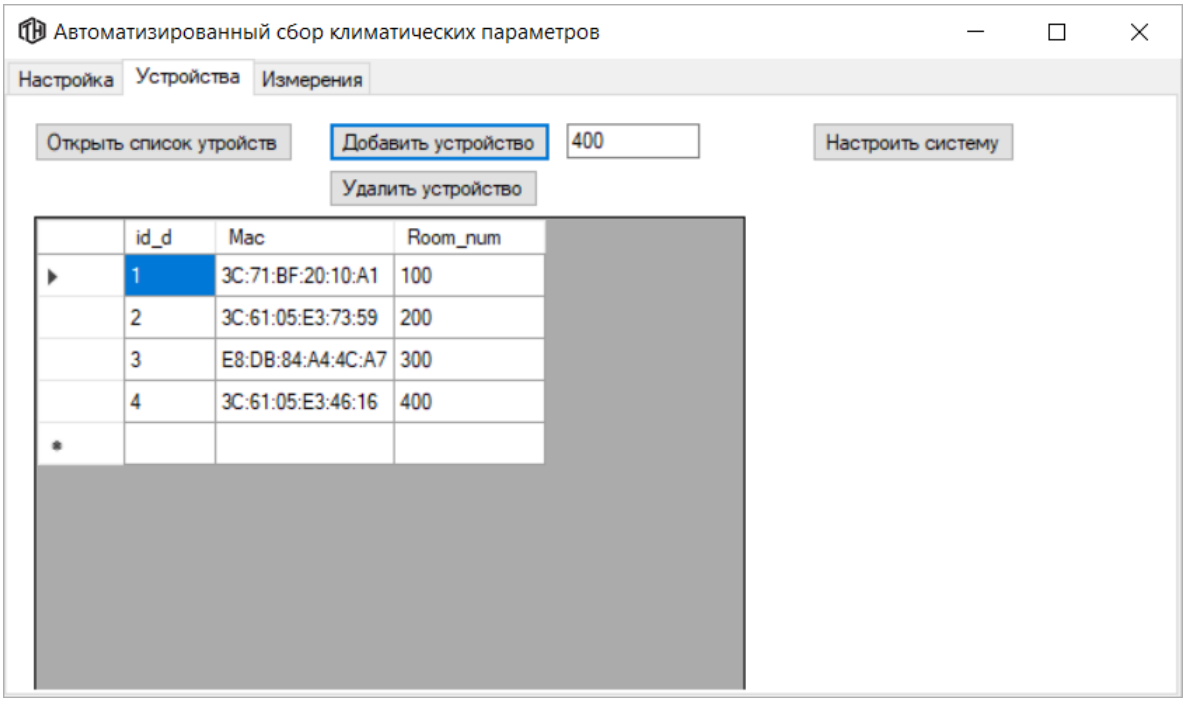


Рисунок 17 – Добавление нового устройства

Для удаления устройства из таблицы необходима выбрать нужное устройство в таблице и нажать кнопку «Удалить устройство». После нажатия кнопки происходит определение номера выделенной строки, затем определяется номер устройства (id_d) в этой строке и далее создаётся запрос в СУБД SQLite для удаления устройства из БД. Затем отображается обновленная таблица с активными устройствами. Удаление устройства изображено на рисунке 18.

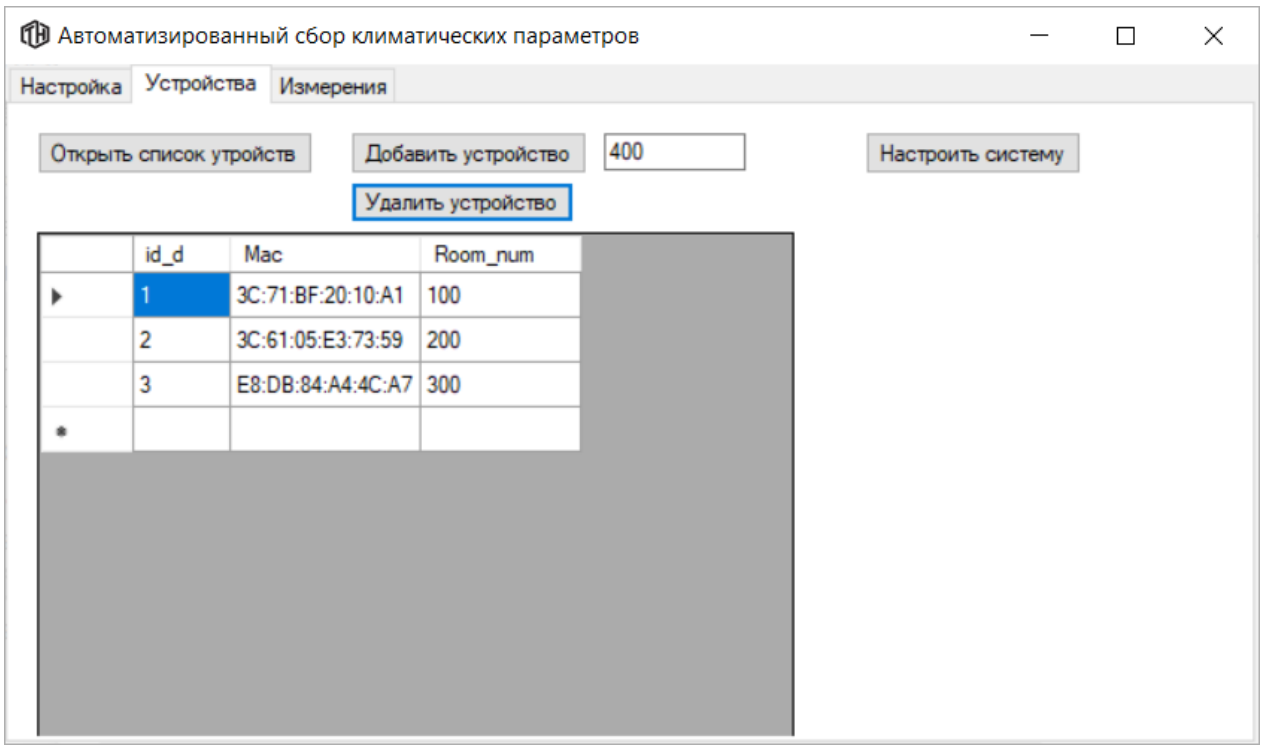


Рисунок 18 – Удаление устройства

Для того, чтобы МК понимал в каком порядке происходит передача данных между устройствами была реализована кнопка «Настроить систему». По её нажатию создаётся строка следующего вида:

«1, Mac1, Mac2, Mac3,…, MacN;»

где «1» – это номер команды для МК, по которой он понимает, что сейчас будет произведена настройка системы; «Mac1,…,MacN» – Mac-адреса в порядке нумерации комнат от 1 до N; «;» – символ конца передачи.

На вкладке «Измерения» были реализованы следующие функции:

- открытие списка измерений;
- удаление выбранного измерения;
- создание запроса на начало измерения температуры и влажности.

Кнопка для отображения таблицы с измерениями из БД «Открыть список измерений» работает по тому же алгоритму, что и кнопка «Открыть список устройств» с единственным отличием, что она создаёт запрос на отображение таблица измерений. Пример работы кнопки представлен на рисунке 19.

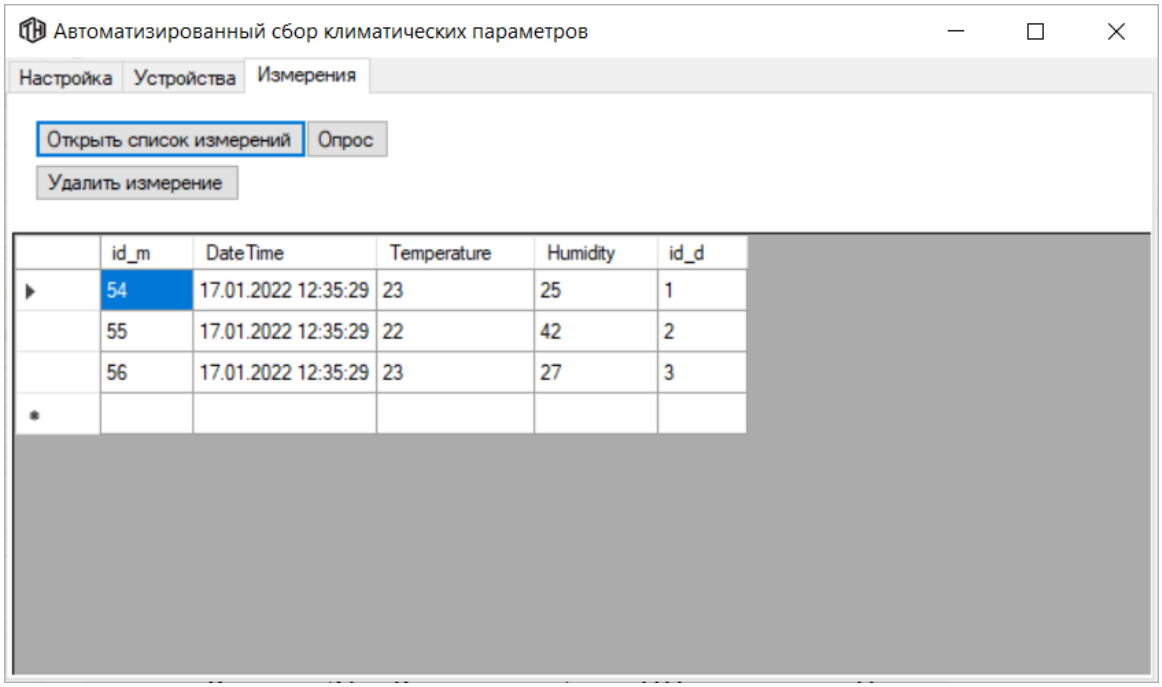


Рисунок 19 – Открытие списка измерений

Кнопка «Удалить измерение» работает аналогично кнопке «Удалить устройство», т.е. по её нажатию определяется номер выделенной строки и создаётся запрос в СУБД на удаление данной записи. Пример удаления представлен на рисунке 20.

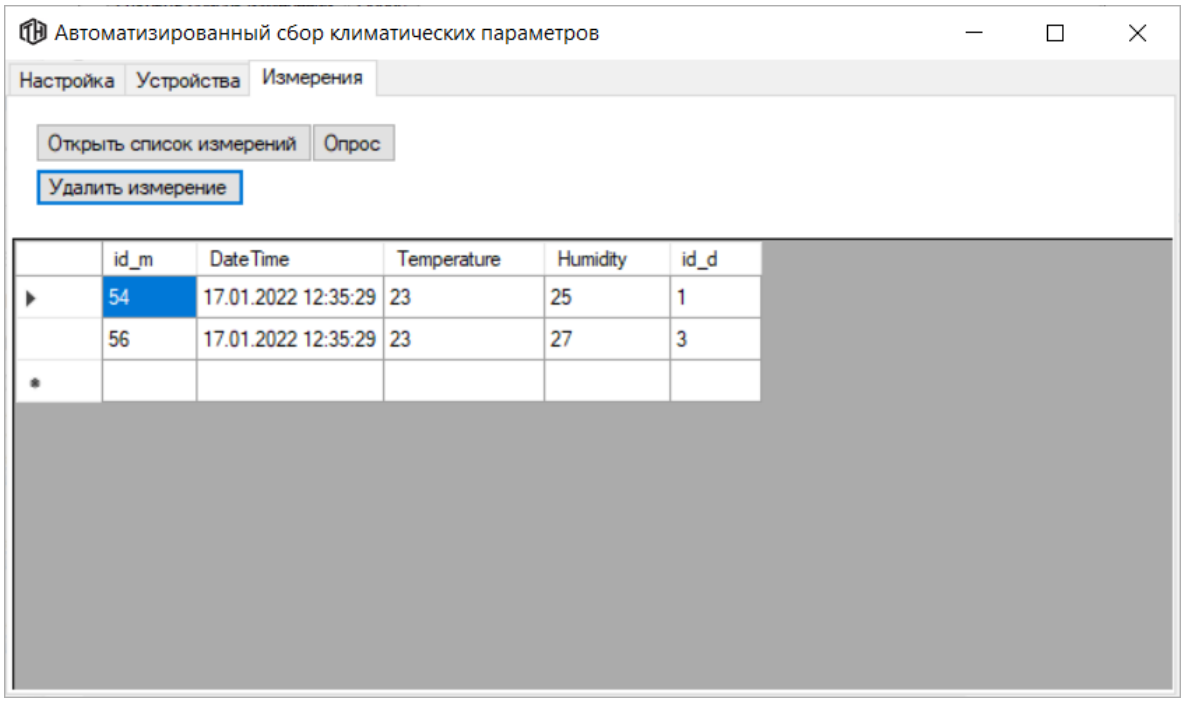


Рисунок 20 – Удаление измерения

На рисунке видно, что было удалено второе устройство, т.е. в таблице id_d = 2.

Кнопка «Опрос» работает в следующем порядке. По её нажатию создаются команда для МК «3», которая отправляется в COM-порт, по которой МК начинает измерение температуры и влажности. Одновременно с этим создаётся запрос в СУБД, в котором определяются номера устройств из таблицы «Устройства». После этого программа ждёт ответ от МК, в котором будут измеренные температуры и влажности. Данная посылка выглядит следующим образом:

$$T_1, H_1, T_2, H_2, \ldots, T_N, H_N,$$

где T_1, T_2, …, T_N – температуры измеренные на первом, втором, …, N-устройстве, а H_1, H_2, …, H_N – влажности измеренные на первом, втором, …, N-устройстве. Затем полученная посылка разбивается по устройствам и создаётся заброс в СУБД на запись. В запрос кроме измеренных параметров входит дата и время на момент измерения и номер устройства. Пример результата работы программы по нажатию кнопки «Опрос» представлен на рисунке 21.

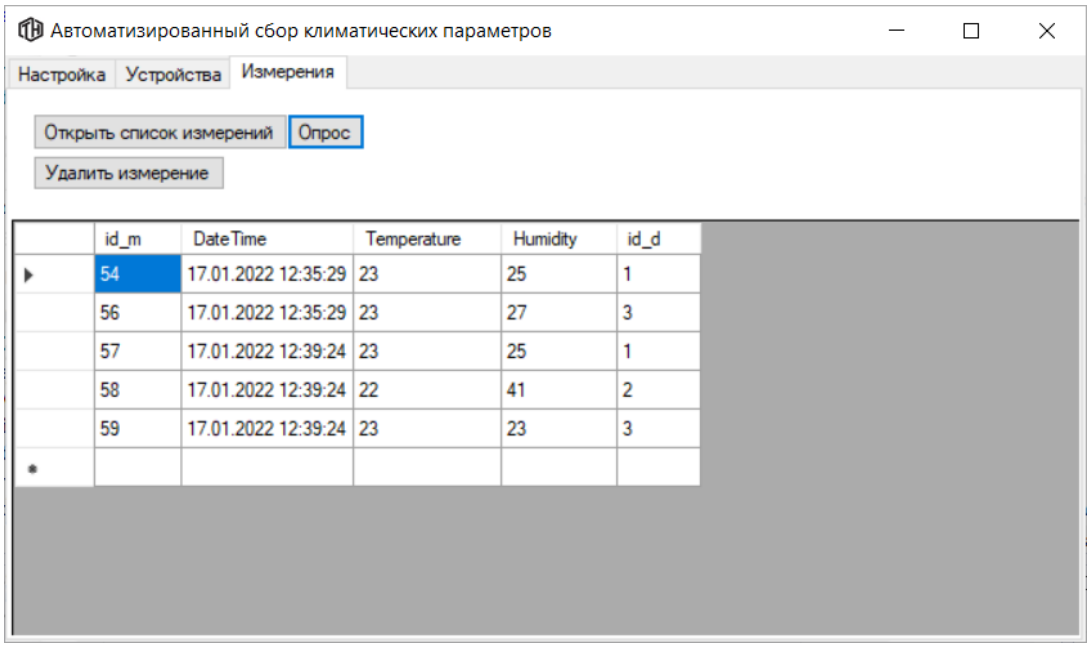


Рисунок 21 – Результат работы программы по нажатию кнопки «Опрос»

Как видно из рисунка 21 программа выполняет функцию записи в БД.

4 Экспериментальное исследование системы

Для эксперимента была собрана система сбора климатических параметров, принцип работы которой представлен на рисунке 4 в главе 2.1. На рисунке 22 представлено устройство сбора климатических параметров, расположенное в корпусе.

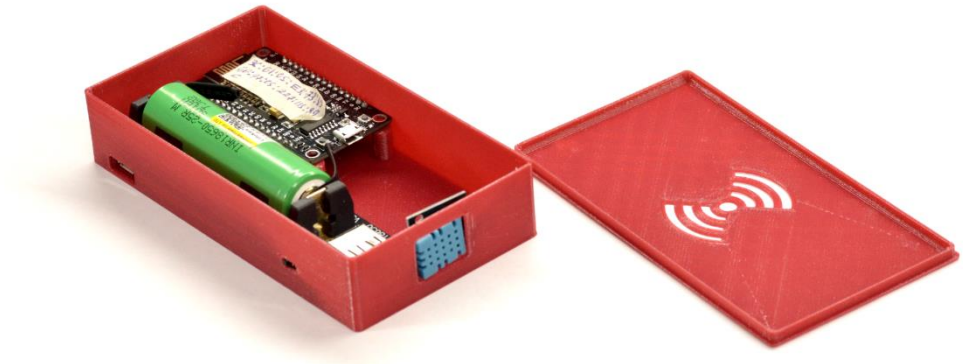


Рисунок 22 – Устройство сбора климатических параметров

У нас имеются три устройства сбора климатических параметров, одно из которых является “головным”, т.е. взаимодействующим с ПК на прямую. “Головное” устройство выполняет следующие функции:

- приём и оправку команд в COM-порт;
- оправку запроса на начало измерения в сеть устройств;
- формирование пакета измеренных температур и влажностей и отправка данного пакета на ПК.

4.1 Добавление нового устройства

Для добавления нового нужно подключить его к компьютеру, а в написанном программном обеспечении открыть нужный COM-порт и нажать кнопку «Добавить устройство». После этих действий ПО отправит запрос МК

«4;», по котором МК отправит свой Mac-адрес обратно в ПО. Пример добавления устройства приведен на рисунке 23.

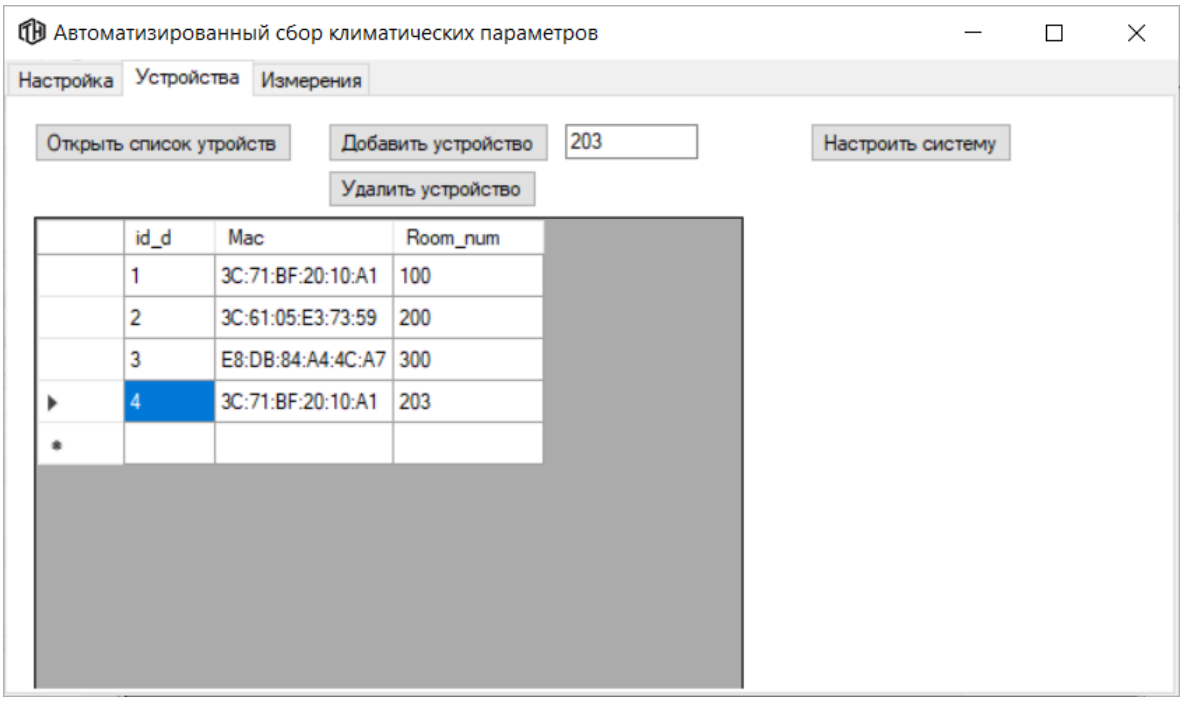


Рисунок 23 – Система сбора климатических параметров

Как видно из рисунка 23 было добавлено устройство со следующими атрибутами:

- номер устройства (id_d) – 4;
- Mac-адрес устройства – 3C:71:BF:20:10:A1;
- номер комнаты – 203.

4.2 Приём и отправка пакета данных

Для визуализации приёма и отправки пакета данных в код программного обеспечения добавили окно информирования, в котором отобразятся переменные, в которых записаны пакеты данных для приёма и отправки. Пример данных пакетов представлен на рисунке 24.

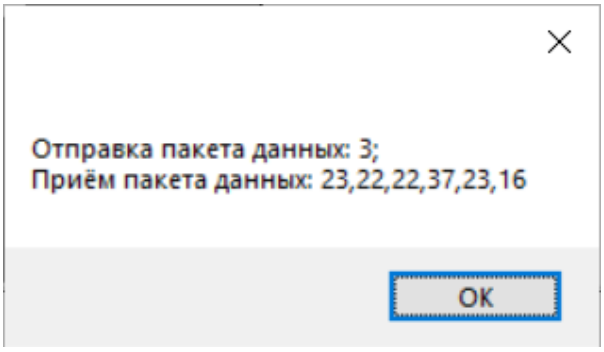


Рисунок 24 – Отправка и приём пакета данных

На рисунке 24 представлена команда микроконтроллеру на начало измерения и сформированный пакет данных, пришедший на ПК.

4.2.1 Описание эксперимента и результаты

Для проведения эксперимента имеются три устройства сбора климатических параметров, одно из них подключено к ПК для общения с программным обеспечением через COM-порт.

В ПО необходимо подключиться к нужному COM-порту, произвести настройку системы, т.е. отправить в МК порядок Mac-адресов по которому устройства поймут своё место в системе. После этого можно производить опрос системы сбора климатических параметров.

На рисунке 25 представлены результаты измерения температуры и влажности с трёх устройств.

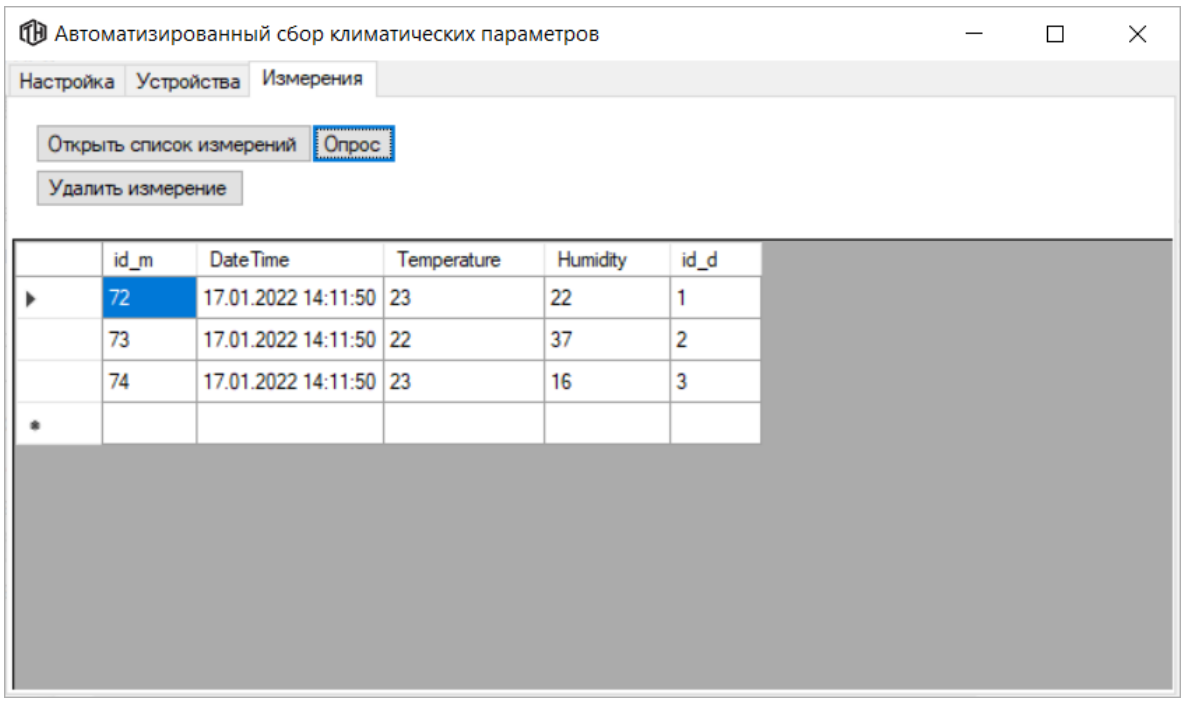


Рисунок 25 – Результат измерения температуры и влажности

Далее проверим, что произойдёт, если выключить последнее устройство. Результат отключения третьего устройства представлен на рисунке 26.

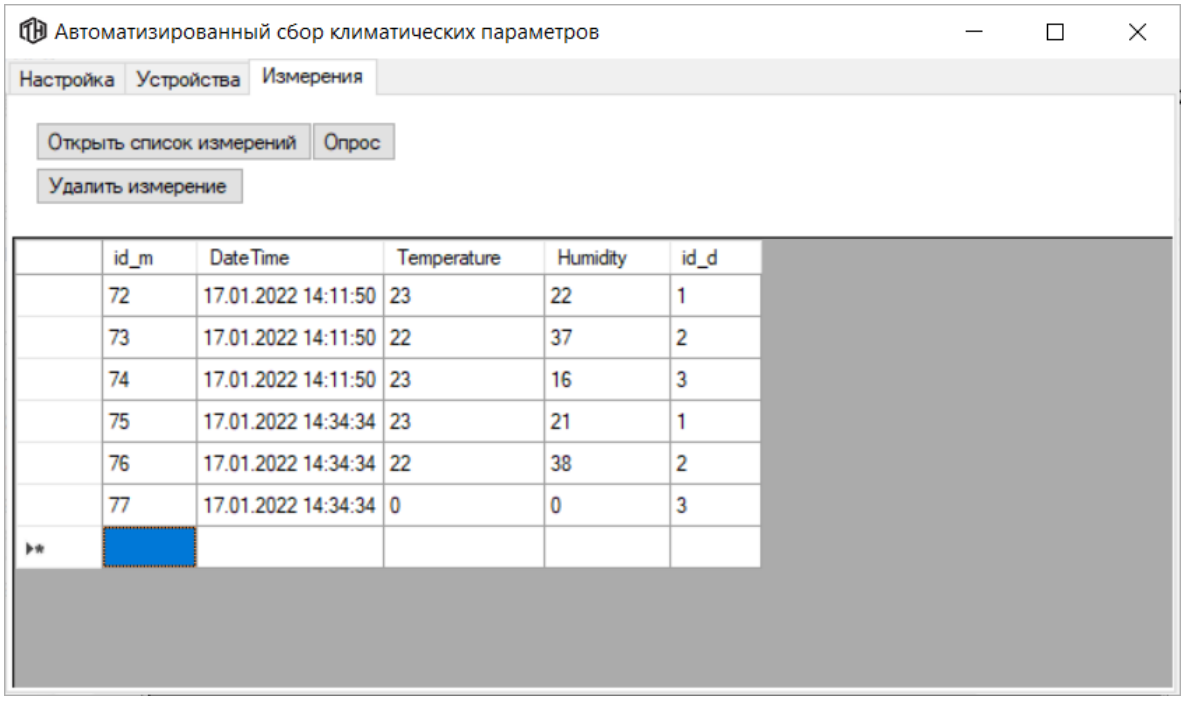
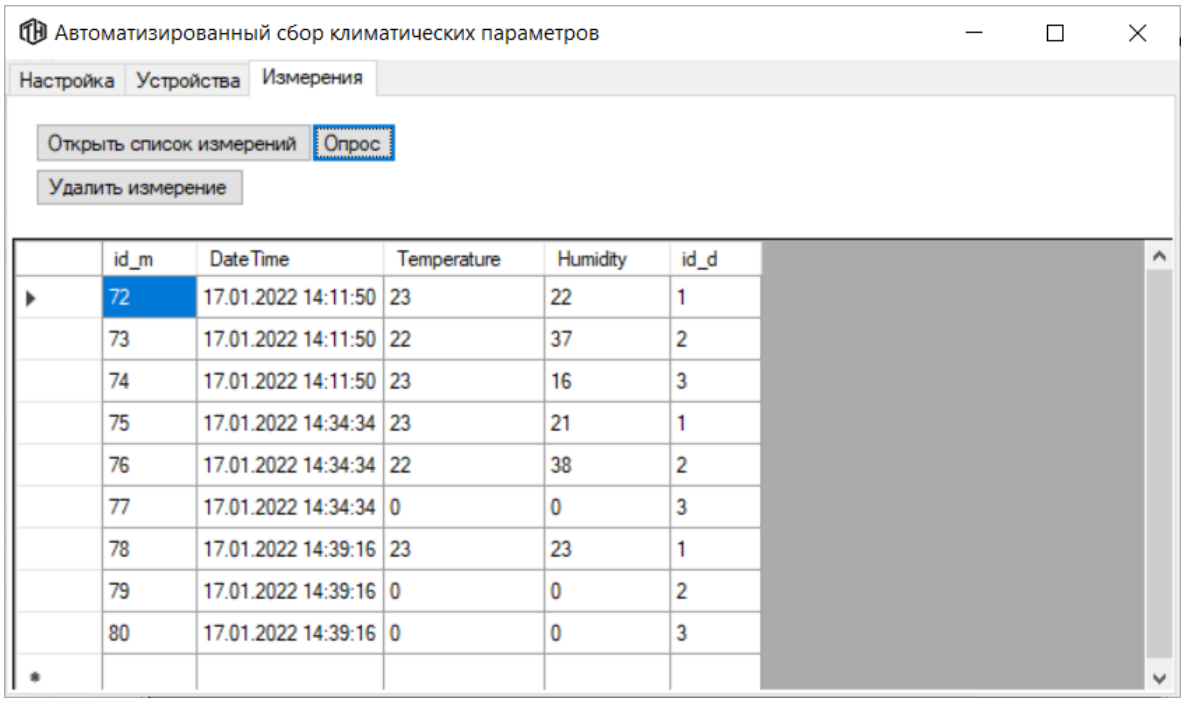


Рисунок 26 – Результат отключения третьего устройства

Из рисунка 26 видно, что при отключении устройства, температура и влажность равны нулю, т.е. по этим значениям можем понять, что устройство разрядилось.

Теперь проверим, что будет если отключить второе устройства при включенном третьем устройстве. Результат представлен на рисунке 27.



id_m	DateTime	Temperature	Humidity	id_d
72	17.01.2022 14:11:50	23	22	1
73	17.01.2022 14:11:50	22	37	2
74	17.01.2022 14:11:50	23	16	3
75	17.01.2022 14:34:34	23	21	1
76	17.01.2022 14:34:34	22	38	2
77	17.01.2022 14:34:34	0	0	3
78	17.01.2022 14:39:16	23	23	1
79	17.01.2022 14:39:16	0	0	2
80	17.01.2022 14:39:16	0	0	3

Рисунок 27 – Результат отключения второго устройства

Как видно из рисунка 27 при выходе из строя промежуточного устройства, все последующие устройства не будут опрошены.

4.3 Иследование предельной дальности связи

До проведения эксперимента разнесём устройства по комнатам как показано на рисунке 28.

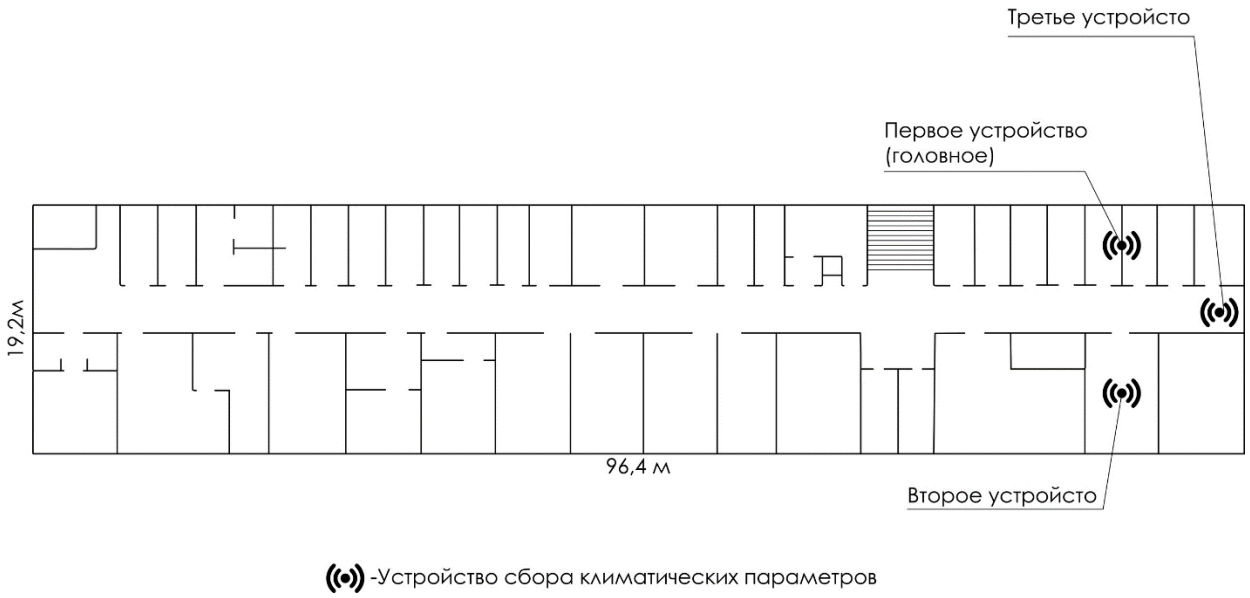
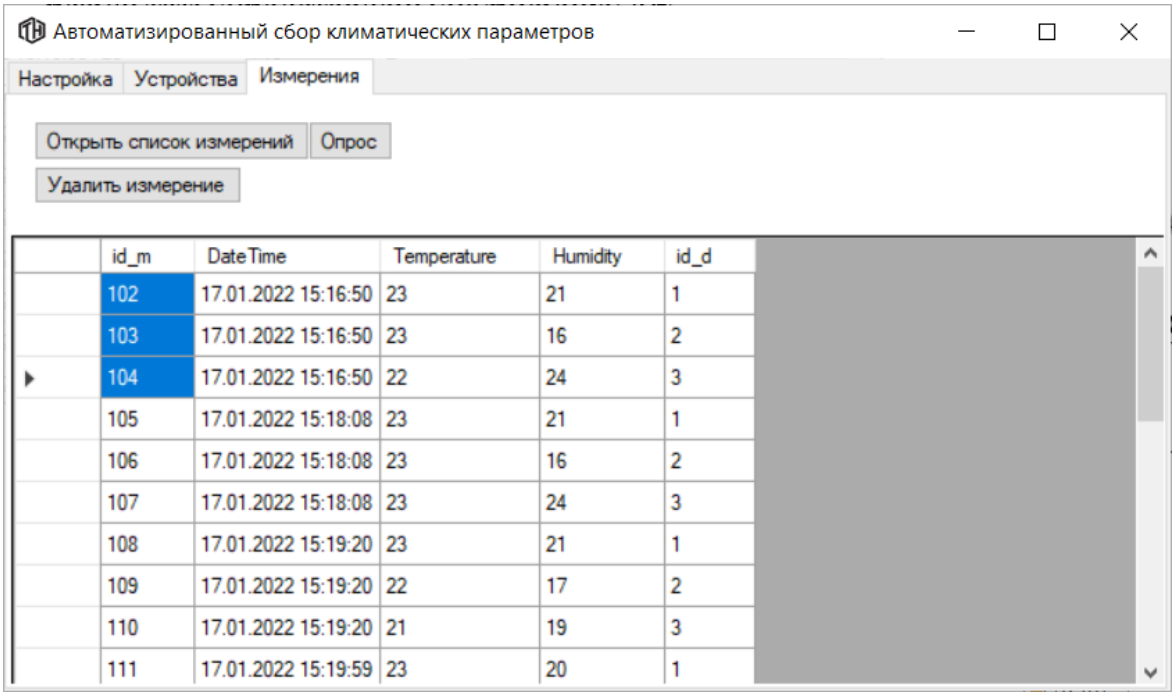


Рисунок 28 – Размещение устройств по комнатам

Затем проведем измерение температуры и влажности, полученные результаты представлены на рисунке 29.



id_m	DateTime	Temperature	Humidity	id_d
102	17.01.2022 15:16:50	23	21	1
103	17.01.2022 15:16:50	23	16	2
104	17.01.2022 15:16:50	22	24	3
105	17.01.2022 15:18:08	23	21	1
106	17.01.2022 15:18:08	23	16	2
107	17.01.2022 15:18:08	23	24	3
108	17.01.2022 15:19:20	23	21	1
109	17.01.2022 15:19:20	22	17	2
110	17.01.2022 15:19:20	21	19	3
111	17.01.2022 15:19:59	23	20	1

Рисунок 29 – Результат измерения температуры и влажности

На данном рисунке представлен результат измерения температуры и влажности в комнатных условиях, в соответствии с расположением устройств, представленном на рисунке 28.

Далее проведём эксперимент с увеличением расстояния между вторым и третьим устройствами, что показано на рисунке 30.

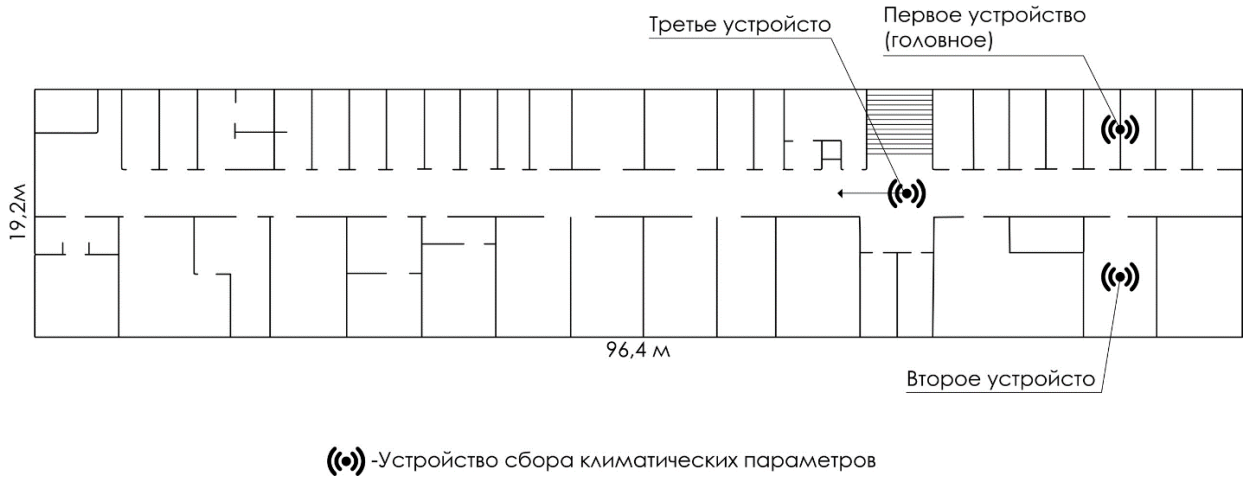
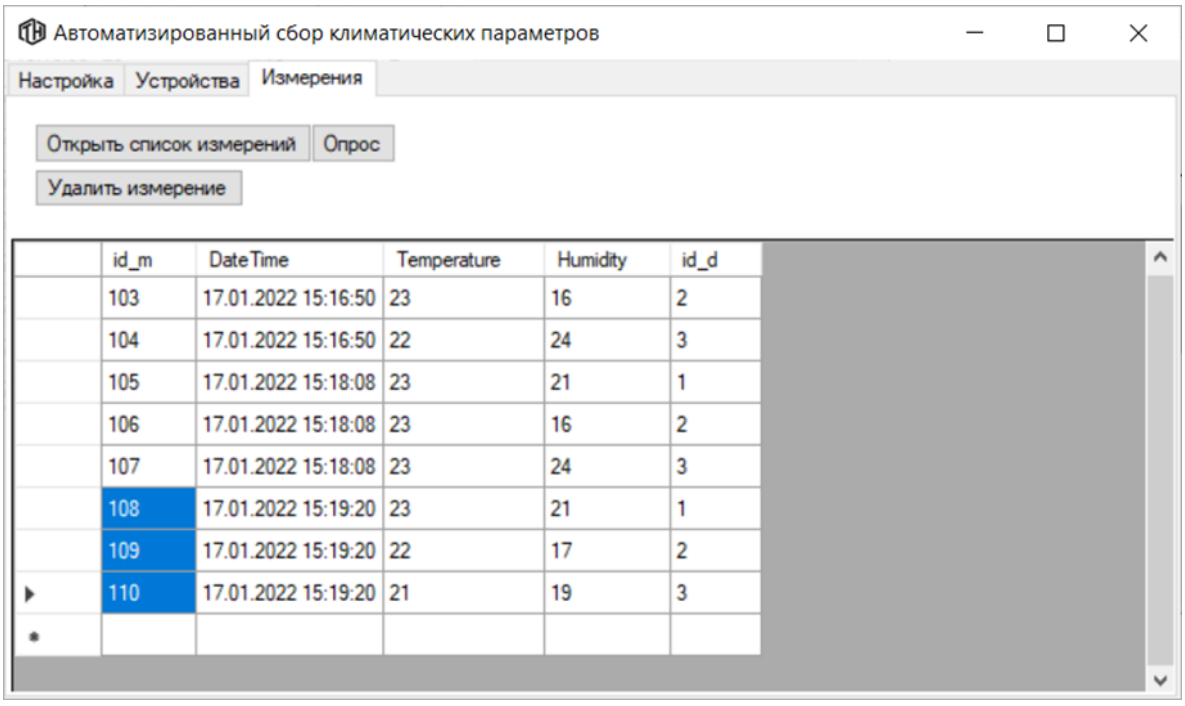


Рисунок 30 – Схема увеличения расстояния между устройствами

Результат измерения при расстоянии примерно равном $\Delta l \cong 20$ м между вторым и третьим устройствами представлен на рисунке 31.



id_m	DateTime	Temperature	Humidity	id_d
103	17.01.2022 15:16:50	23	16	2
104	17.01.2022 15:16:50	22	24	3
105	17.01.2022 15:18:08	23	21	1
106	17.01.2022 15:18:08	23	16	2
107	17.01.2022 15:18:08	23	24	3
108	17.01.2022 15:19:20	23	21	1
109	17.01.2022 15:19:20	22	17	2
110	17.01.2022 15:19:20	21	19	3

Рисунок 31 – Результат измерения температуры и влажности

Как видно из рисунка 31 между вторым и третьим устройствами происходит обмен данными.

Попробуем увеличить расстояние между устройствами до Δl≅40 м. Результат работы системы представлен на рисунке 32.

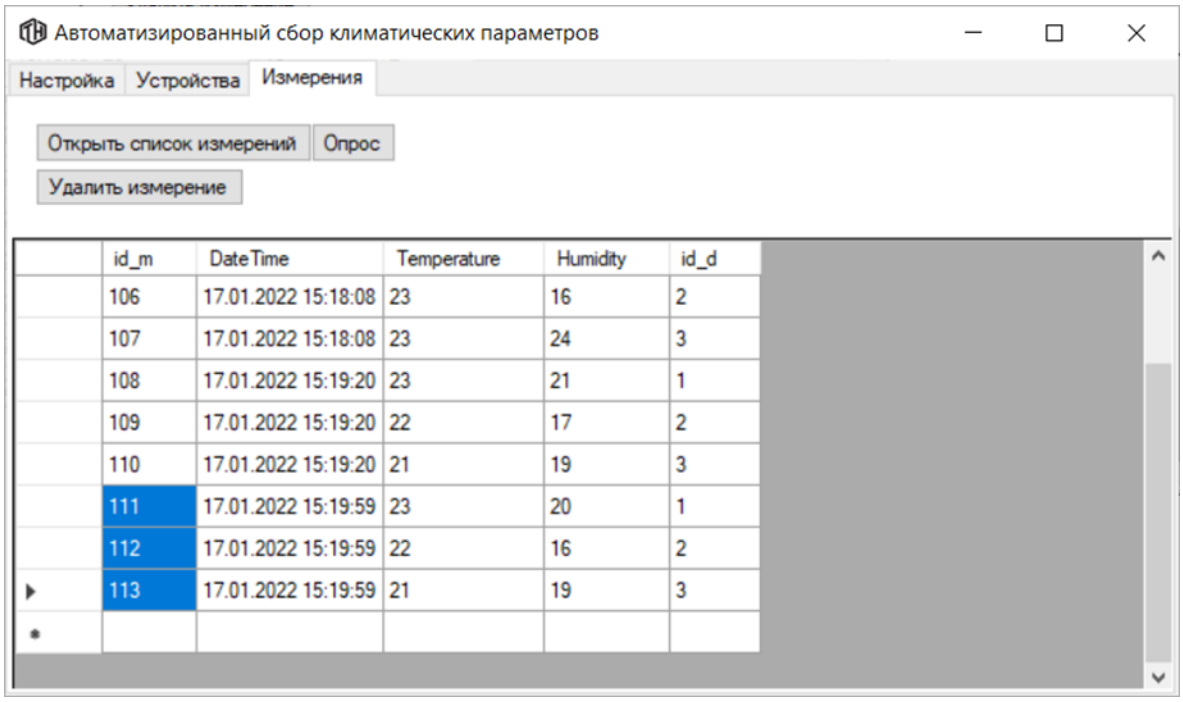


Рисунок 32 – Результат измерения температуры и влажности

Из рисунка видно, что связь поддерживается. Значит увеличиваем расстояние Δl≅50 м. результат измерения представлен на рисунке 33.

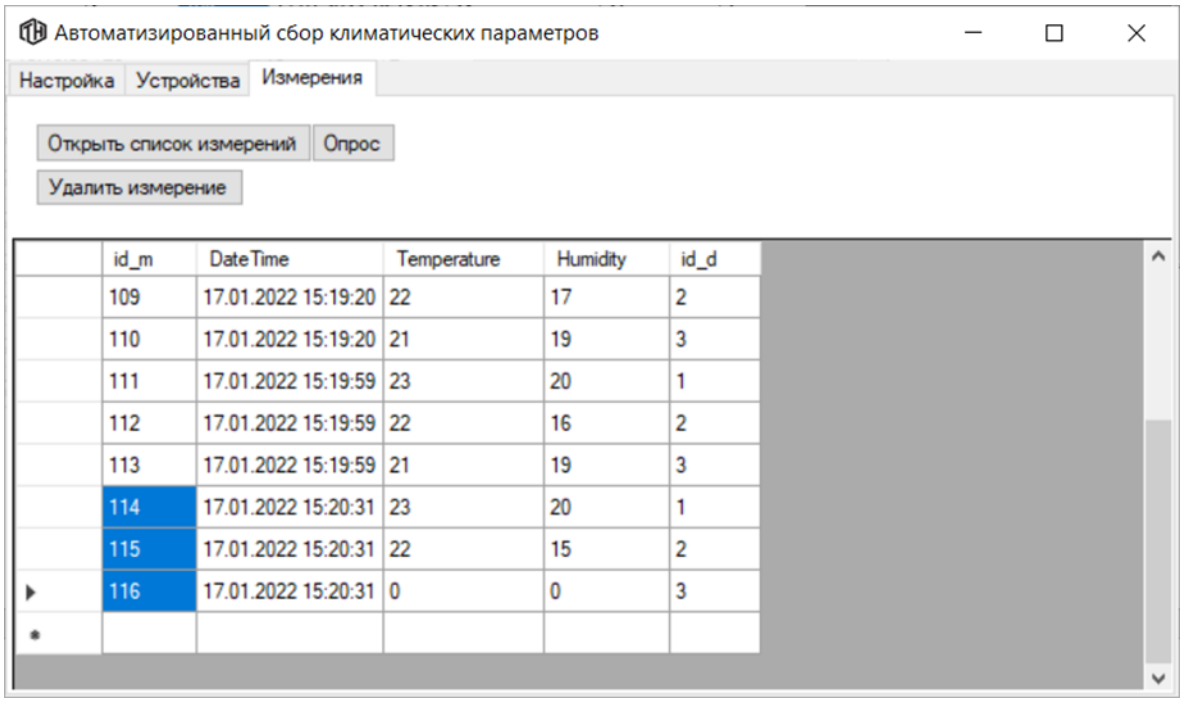


Рисунок 33 – Результат измерения температуры и влажности

На рисунке 33 видно, что температура и влажность с третьего устройства пришли равные нулю, а это значит, что его опросить не удалось из-за большой дальности между вторым и третьим устройствами. При этом можно сказать, что связь прекратилась примерно при расстоянии равном 50 м, а это результата вполне достаточно для нашей системы.

4.3.1 Эксперимент с изменением микроклимата

Для проверки работы системы при различных температурах и влажностях, был проведен эксперимент, где первое устройство находится в комнатных условиях, второе устройство расположили над стаканом с горячей водой, а третье – разместили рядом с открытым окном, при температуре воздуха за окном примерно равной 0 °C. Результаты данного эксперимента представлены на рисунке 34.

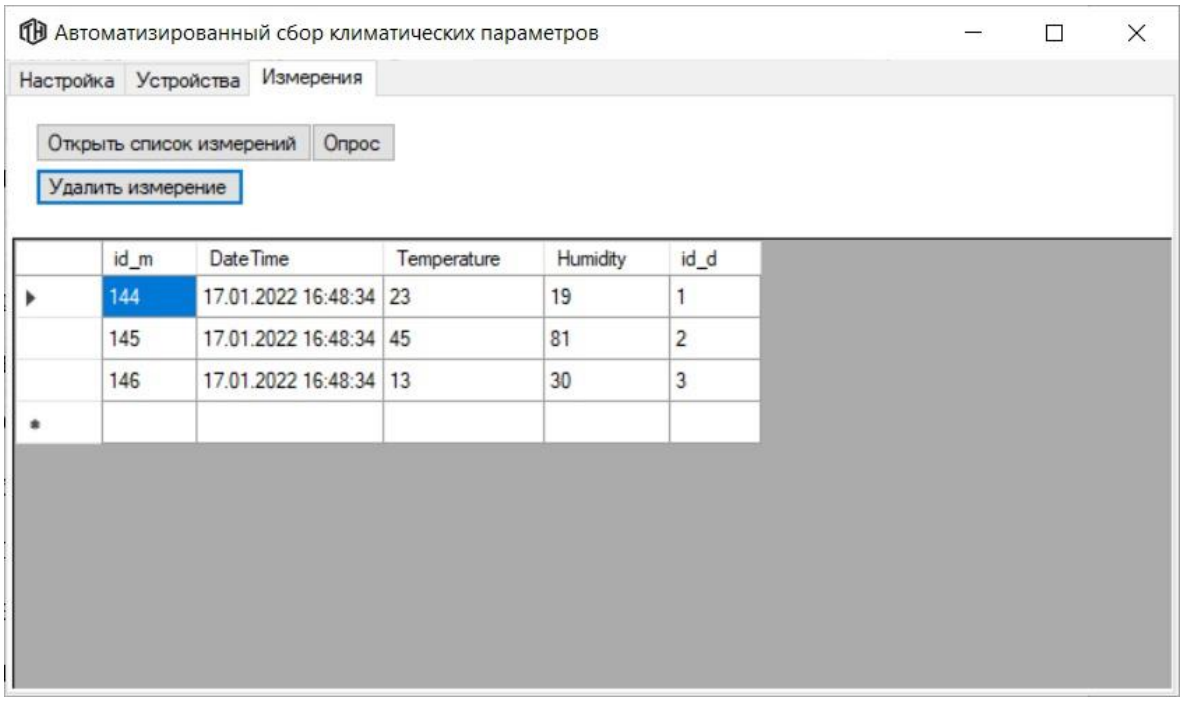


Рисунок 34 – Результат эксперимента с изменением микроклимата

Как видно из рисунка 34, на втором устройстве резко увеличилась температура и влажность, а на третьем уменьшилась температура, а влажность увеличилась.

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР были получены следующие результаты:

– изучена среда программирования Microsoft Visual Studio;

– изучены принципы проектирования баз данных и система управления базами данных SQLite;

– спроектирована и создана база данных в системе управления базами данных SQLite;

– разработан интерфейс программного обеспечения в среде Microsoft Visual Studio, который организует взаимодействие с разработанной базой данных и микроконтроллером ESP8266.

Кроме этого, был проведен ряд экспериментов, проверяющий работоспособность ПО с сетью микроконтроллеров, а именно было протестировано:

– добавление нового устройства;

– работа программы при отключении последнего либо промежуточного устройства;

– предельная дальность связи между устройствами, между которыми есть препятствия в виде стен из железобетона;

– работа системы в различных условиях микроклимата.

Также было проведено исследование по экономической целесообразности и техники безопасности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Visual Studio Community // Microsoft – Бесплатная полнофункциональная расширяемая среда IDE для создания современных приложений Android, iOS и Windows, а также веб-приложений и облачных служб. – 2022. – URL: https://visualstudio.microsoft.com/ru/vs/community/, Режим доступа: свободный (дата обращения 19.01.2022).

2. Дейт, К., Дж. // Д27 Введение в систему баз данных, 7-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1072 с.: ил. – Парал. тит. англ.

3. Ульман Дж. // Основы систем баз данных / Пер. с англ. М. Р. Когаловского и В. В. Когутовского; под ред. М. Р. Когаловского. – М.: Финансы и статистика, 1983. – 334 с., ил.

4. Документация по Visual Studio // Microsoft Docs. – 2022. – URL: https://docs.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022, Режим доступа: свободный (дата обращения 19.01.2022).

5. Microsoft Visual Studio // Википедия – свободная энциклопедия. – 2022. –URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_Community , Режим доступа: свободный (дата обращения 19.01.2022).

6. SerialPort Класс // Microsoft Docs: Средства разработчика, техническая документация и учебные ресурсы. – 2022. – URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-5.0, Режим доступа: свободный (дата обращения 19.01.2022).

7. An Introduction To The SQLite C/C++ Interface // SQLite – 2022. – URL: https://www.sqlite.org/cintro.html/, Режим доступа: свободный (дата обращения 19.01.2022).

8. Сравнение NodeMCU-совместимых плат с чипом ESP8266 // Microsin.net– 2022. – URL: http://microsin.net/programming/arm/esp8266-

nodemcu-development-boards.html/, Режим доступа: свободный (дата обращения 19.01.2022).

9. Датчики температуры и влажности DHT11 и DHT22 // MicroPi – Программирование микроконтроллеров, Banana Pi, Orange Pi, Raspberry Pi. – 2022. – URL: https://micro-pi.ru/dht11и-dht22-датчики-температуры-и-влажности/, Режим доступа: свободный (дата обращения 19.01.2022).

10. DateTime Структура // Microsoft – Техническая документация – 2022. – URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.datetime?view=net-6.0/, Режим доступа: свободный (дата обращения 19.01.2022).

ПРИЛОЖЕНИЕ А

Код программного обеспечения

```
#pragma once
#using <System.dll>
#using <system.drawing.dll>
#include <windows.h>
#include <stdio.h>
#include <cliext/vector>
#include <string>
#include <cstring>
#include <iostream>
#include <ctime>
namespace Test {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Collections::Generic;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO::Ports;
	using namespace System::Threading;
	using namespace System::Data::SQLite;
	using namespace cliext;
	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
		}
	private: System::Windows::Forms::Button^ button8;

	private: System::ComponentModel::BackgroundWorker^ backgroundWorker1;
	private: System::Windows::Forms::Button^ button9;
	private: System::Windows::Forms::Button^ button10;
	private: System::Windows::Forms::Button^ button11;
```

```
private: System::Windows::Forms::MaskedTextBox^ maskedTextBox1;
private: System::Windows::Forms::Button^ button12;
public:
SQLiteConnection^ db;
protected:
	/// <summary>
	/// Освободить все используемые ресурсы.
	/// </summary>
	~MyForm()
	{
		if (components)
		{
			delete components;
		}
		if (!db == 0)
		{
			db->Close();
		}

		serialPort1->Close();
	}
private: System::Windows::Forms::TabPage^ tabPage2;
protected:
private: System::Windows::Forms::TabControl^ tabControl1;
private: System::Windows::Forms::ComboBox^ comboBox1;
private: System::Windows::Forms::ComboBox^ comboBox2;
private: System::Windows::Forms::Button^ button3;
private: System::Windows::Forms::Button^ button2;
private: System::IO::Ports::SerialPort^ serialPort1;
private: System::Windows::Forms::Button^ button5;
private: System::Windows::Forms::TabPage^ tabPage3;
private: System::Windows::Forms::DataGridView^ dataGridView1;
private: System::Windows::Forms::TabPage^ tabPage1;
private: System::Windows::Forms::DataGridView^ dataGridView2;
private: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;
private: System::Windows::Forms::Button^ button7;
private: System::Windows::Forms::Button^ button6;
protected:
private: System::ComponentModel::IContainer^ components;
private:
	/// <summary>
```

```
        /// Обязательная переменная конструктора.
        /// </summary>
#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора – не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->components = (gcnew System::ComponentModel::Container());
            System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(MyForm::typeid));
            this->tabPage2 = (gcnew System::Windows::Forms::TabPage());
            this->button9 = (gcnew System::Windows::Forms::Button());
            this->button5 = (gcnew System::Windows::Forms::Button());
            this->button3 = (gcnew System::Windows::Forms::Button());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());
            this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
            this->tabControl1 = (gcnew System::Windows::Forms::TabControl());
            this->tabPage3 = (gcnew System::Windows::Forms::TabPage());
            this->maskedTextBox1 = (gcnew System::Windows::Forms::MaskedTextBox());
            this->button10 = (gcnew System::Windows::Forms::Button());
            this->button7 = (gcnew System::Windows::Forms::Button());
            this->button6 = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
            this->tabPage1 = (gcnew System::Windows::Forms::TabPage());
            this->button12 = (gcnew System::Windows::Forms::Button());
            this->button11 = (gcnew System::Windows::Forms::Button());
            this->button8 = (gcnew System::Windows::Forms::Button());
            this->dataGridView2 = (gcnew System::Windows::Forms::DataGridView());
            this->serialPort1 = (gcnew System::IO::Ports::SerialPort(this->components));
            this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());
            this->backgroundWorker1 = (gcnew
System::ComponentModel::BackgroundWorker());
            this->tabPage2->SuspendLayout();
            this->tabControl1->SuspendLayout();
            this->tabPage3->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
            this->tabPage1->SuspendLayout();
```

```
(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->BeginInit();
this->SuspendLayout();
//
// tabPage2
//
this->tabPage2->Controls->Add(this->button9);
this->tabPage2->Controls->Add(this->button5);
this->tabPage2->Controls->Add(this->button3);
this->tabPage2->Controls->Add(this->button2);
this->tabPage2->Controls->Add(this->comboBox2);
this->tabPage2->Controls->Add(this->comboBox1);
this->tabPage2->ForeColor = System::Drawing::SystemColors::ControlText;
this->tabPage2->Location = System::Drawing::Point(4, 25);
this->tabPage2->Name = L"tabPage2";
this->tabPage2->Padding = System::Windows::Forms::Padding(3);
this->tabPage2->Size = System::Drawing::Size(888, 419);
this->tabPage2->TabIndex = 1;
this->tabPage2->Text = L"Настройка";
this->tabPage2->UseVisualStyleBackColor = true;
//
// button9
//
this->button9->Location = System::Drawing::Point(330, 23);
this->button9->Name = L"button9";
this->button9->Size = System::Drawing::Size(124, 32);
this->button9->TabIndex = 11;
this->button9->Text = L"Отключить";
this->button9->UseVisualStyleBackColor = true;
this->button9->Click += gcnew System::EventHandler(this, &MyForm::button9_Click);
//
// button5
//
this->button5->BackColor = System::Drawing::Color::Transparent;
this->button5->Cursor = System::Windows::Forms::Cursors::Arrow;
this->button5->FlatAppearance->BorderSize = 0;
this->button5->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
this->button5->ForeColor = System::Drawing::SystemColors::ActiveCaptionText;
this->button5->Location = System::Drawing::Point(330, 72);
this->button5->Name = L"button5";
this->button5->Size = System::Drawing::Size(30, 30);
```

```
this->button5->TabIndex = 8;
this->button5->TabStop = false;
this->button5->UseVisualStyleBackColor = false;
//
// button3
//
this->button3->Location = System::Drawing::Point(20, 69);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(124, 32);
this->button3->TabIndex = 4;
this->button3->Text = L"Подключить";
this->button3->UseVisualStyleBackColor = true;
this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
//
// button2
//
this->button2->Location = System::Drawing::Point(20, 23);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(124, 32);
this->button2->TabIndex = 3;
this->button2->Text = L"Скан. портов";
this->button2->UseVisualStyleBackColor = true;
this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
//
// comboBox2
//
this->comboBox2->Cursor = System::Windows::Forms::Cursors::Hand;
this->comboBox2->DropDownStyle =
System::Windows::Forms::ComboBoxStyle::DropDownList;
this->comboBox2->FormattingEnabled = true;
this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(10) {
        L"300 бод", L"600 бод", L"1 200 бод", L"2 400 бод",
            L"4 800 бод", L"9 600 бод", L"19 200 бод", L"38 400 бод", L"57 600
бод", L"115 200 бод"
});
this->comboBox2->Location = System::Drawing::Point(159, 74);
this->comboBox2->Name = L"comboBox2";
this->comboBox2->Size = System::Drawing::Size(141, 24);
this->comboBox2->TabIndex = 2;
//
```

```
// comboBox1
//
this->comboBox1->Cursor = System::Windows::Forms::Cursors::Hand;
this->comboBox1->DropDownStyle =
System::Windows::Forms::ComboBoxStyle::DropDownList;
this->comboBox1->Location = System::Drawing::Point(159, 28);
this->comboBox1->Name = L"comboBox1";
this->comboBox1->Size = System::Drawing::Size(141, 24);
this->comboBox1->TabIndex = 9;
//
// tabControl1
//
this->tabControl1->Controls->Add(this->tabPage2);
this->tabControl1->Controls->Add(this->tabPage3);
this->tabControl1->Controls->Add(this->tabPage1);
this->tabControl1->Dock = System::Windows::Forms::DockStyle::Fill;
this->tabControl1->ImeMode = System::Windows::Forms::ImeMode::NoControl;
this->tabControl1->Location = System::Drawing::Point(0, 0);
this->tabControl1->Name = L"tabControl1";
this->tabControl1->SelectedIndex = 0;
this->tabControl1->Size = System::Drawing::Size(896, 448);
this->tabControl1->TabIndex = 5;
//
// tabPage3
//
this->tabPage3->Controls->Add(this->maskedTextBox1);
this->tabPage3->Controls->Add(this->button10);
this->tabPage3->Controls->Add(this->button7);
this->tabPage3->Controls->Add(this->button6);
this->tabPage3->Controls->Add(this->button1);
this->tabPage3->Controls->Add(this->dataGridView1);
this->tabPage3->Location = System::Drawing::Point(4, 25);
this->tabPage3->Name = L"tabPage3";
this->tabPage3->Size = System::Drawing::Size(888, 419);
this->tabPage3->TabIndex = 2;
this->tabPage3->Text = L"Устройства";
this->tabPage3->UseVisualStyleBackColor = true;
//
// maskedTextBox1
//
this->maskedTextBox1->Location = System::Drawing::Point(423, 19);
this->maskedTextBox1->Mask = L"990";
```

```
this->maskedTextBox1->Name = L"maskedTextBox1";
this->maskedTextBox1->Size = System::Drawing::Size(100, 22);
this->maskedTextBox1->TabIndex = 12;
//
// button10
//
this->button10->AutoSize = true;
this->button10->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
this->button10->Location = System::Drawing::Point(609, 17);
this->button10->Name = L"button10";
this->button10->Size = System::Drawing::Size(144, 26);
this->button10->TabIndex = 11;
this->button10->Text = L"Настроить систему";
this->button10->UseVisualStyleBackColor = true;
this->button10->Click += gcnew System::EventHandler(this,
&MyForm::button10_Click);
//
// button7
//
this->button7->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
this->button7->Location = System::Drawing::Point(241, 50);
this->button7->Name = L"button7";
this->button7->Size = System::Drawing::Size(160, 27);
this->button7->TabIndex = 3;
this->button7->Text = L"Удалить устройство";
this->button7->UseVisualStyleBackColor = true;
this->button7->Click += gcnew System::EventHandler(this,
&MyForm::button7_Click);
//
// button6
//
this->button6->AutoSize = true;
this->button6->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
this->button6->Location = System::Drawing::Point(241, 17);
this->button6->Name = L"button6";
this->button6->Size = System::Drawing::Size(159, 26);
this->button6->TabIndex = 2;
this->button6->Text = L"Добавить устройство";
this->button6->UseVisualStyleBackColor = true;
```

```
this->button6->Click += gcnew System::EventHandler(this,
&MyForm::button6_Click);
//
// button1
//
this->button1->AutoSize = true;
this->button1->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
this->button1->Location = System::Drawing::Point(17, 17);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(184, 26);
this->button1->TabIndex = 1;
this->button1->Text = L"Открыть список утройств";
this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
//
// dataGridView1
//
this->dataGridView1->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::AllCells;
this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
this->dataGridView1->Location = System::Drawing::Point(17, 84);
this->dataGridView1->Name = L"dataGridView1";
this->dataGridView1->RowHeadersWidth = 51;
this->dataGridView1->RowTemplate->Height = 24;
this->dataGridView1->Size = System::Drawing::Size(541, 339);
this->dataGridView1->TabIndex = 0;
//
// tabPage1
//
this->tabPage1->Controls->Add(this->button12);
this->tabPage1->Controls->Add(this->button11);
this->tabPage1->Controls->Add(this->button8);
this->tabPage1->Controls->Add(this->dataGridView2);
this->tabPage1->Location = System::Drawing::Point(4, 25);
this->tabPage1->Name = L"tabPage1";
this->tabPage1->Size = System::Drawing::Size(888, 419);
this->tabPage1->TabIndex = 3;
this->tabPage1->Text = L"Измерения";
this->tabPage1->UseVisualStyleBackColor = true;
```

```
            //
            // button12
            //
            this->button12->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
            this->button12->Location = System::Drawing::Point(17, 49);
            this->button12->Name = L"button12";
            this->button12->Size = System::Drawing::Size(160, 27);
            this->button12->TabIndex = 14;
            this->button12->Text = L"Удалить измерение";
            this->button12->UseVisualStyleBackColor = true;
            this->button12->Click += gcnew System::EventHandler(this,
&MyForm::button12_Click);
            //
            // button11
            //
            this->button11->AutoSize = true;
            this->button11->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
            this->button11->Location = System::Drawing::Point(228, 17);
            this->button11->Name = L"button11";
            this->button11->Size = System::Drawing::Size(58, 26);
            this->button11->TabIndex = 13;
            this->button11->Text = L"Опрос";
            this->button11->UseVisualStyleBackColor = true;
            this->button11->Click += gcnew System::EventHandler(this,
&MyForm::button11_Click);
            //
            // button8
            //
            this->button8->AutoSize = true;
            this->button8->AutoSizeMode =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
            this->button8->Location = System::Drawing::Point(17, 17);
            this->button8->Name = L"button8";
            this->button8->Size = System::Drawing::Size(196, 26);
            this->button8->TabIndex = 2;
            this->button8->Text = L"Открыть список измерений";
            this->button8->UseVisualStyleBackColor = true;
            this->button8->Click += gcnew System::EventHandler(this,
&MyForm::button8_Click);
            //
```

```
            // dataGridView2
            //
            this->dataGridView2->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::AllCells;
            this->dataGridView2->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
            this->dataGridView2->Location = System::Drawing::Point(0, 98);
            this->dataGridView2->Name = L"dataGridView2";
            this->dataGridView2->RowHeadersWidth = 51;
            this->dataGridView2->RowTemplate->Height = 24;
            this->dataGridView2->Size = System::Drawing::Size(888, 321);
            this->dataGridView2->TabIndex = 1;
            //
            // openFileDialog1
            //
            this->openFileDialog1->FileName = L"database.db";
            this->openFileDialog1->Filter = L"Файлы БД (*.db)|*.db";
            //
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(896, 448);
            this->Controls->Add(this->tabControl1);
            this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources-
>GetObject(L"$this.Icon")));
            this->Name = L"MyForm";
            this->Text = L"Автоматизированный сбор климатических параметров";
            this->tabPage2->ResumeLayout(false);
            this->tabControl1->ResumeLayout(false);
            this->tabPage3->ResumeLayout(false);
            this->tabPage3->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
            this->tabPage1->ResumeLayout(false);
            this->tabPage1->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView2))->EndInit();
            this->ResumeLayout(false);
        }
#pragma endregion
```

```
private: DataTable^ fillDataTable(String^ tablename) {

	DataTable^ table;
	try
	{
		SQLiteCommand^ cmdSelect = db->CreateCommand();
		//Обратите внимание, что SQL запрос оформляем как обычную строчку
		cmdSelect->CommandText = tablename;
			/*"SELECT * FROM Devices;";*/
		SQLiteDataReader^ reader = cmdSelect->ExecuteReader();
		//Переменные
		DataColumn^ column; //Столбец таблицы
		DataRow^ row; //Строка таблицы
		//Создаем таблицу данных
		table = gcnew DataTable();
		//Вектор названий столбцов
		vector<String^>^ nameColumns = gcnew vector<String^>();
		//Заполним данные о столбцах
		for (int i = 0; i < reader->FieldCount; i++) {
			nameColumns->push_back(reader->GetName(i));
			column = gcnew DataColumn(nameColumns->at(i), String::typeid);
			table->Columns->Add(column);
		}
		//Пробегаем по каждой записи
		while (reader->Read()) {
			//Заполняем строчку таблицы
			row = table->NewRow();
			//В каждой записи пробегаем по всем столбцам
			for (int i = 0; i < reader->FieldCount; i++) {
				//Добавляем значение столбца в row
				row[nameColumns->at(i)] = reader->GetValue(i)->ToString();
				reader->GetValue(i)->ToString();
			}
			table->Rows->Add(row);
		}
	}
	catch (Exception^ e)
	{
		MessageBox::Show("Error Executing SQL: " + e->ToString(), "Exception While Displaying
MyTable ...");
	}
	return table;
```

```
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
	cli::array<String^>^ serialPorts = nullptr;
	try
	{
		serialPorts = SerialPort::GetPortNames(); //Получение списка доступных COM-портов.
	}
	catch (Win32Exception^ ex)
	{
		MessageBox::Show("Error Win32Exception  " + ex->ToString());
	}
	comboBox1->Items->Clear();
	for each (String ^ port in serialPorts) //Отображение каждого порта в выпадающем списке.
	{
		comboBox1->Items->Add(port);
	}
	if (comboBox1->Items->Count > 0) comboBox1->SelectedIndex = 0;
	comboBox2->SelectedIndex = 9;
}
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
	if (comboBox1->SelectedIndex == -1) {
		button5->BackColor = System::Drawing::Color::Red;
		MessageBox::Show("Ошибка! Не выбран Com-port");
		return;
	}
	int BR;
	switch (comboBox2->SelectedIndex)
	{
	case 0:
		BR = 300; break;
	case 1:
		BR = 600; break;
	case 2:
		BR = 1200; break;
	case 3:
		BR = 2400; break;
	case 4:
		BR = 4800; break;
	case 5:
		BR = 9600; break;
	case 6:
		BR = 19200; break;
```

```
		case 7:
			BR = 38400; break;
		case 8:
			BR = 57600; break;
		case 9:
			BR = 115200; break;
		}
		serialPort1->PortName = comboBox1->SelectedItem->ToString();
		serialPort1->BaudRate = BR;
		serialPort1->Parity = Parity::None;
		serialPort1->DataBits = 8;
		serialPort1->StopBits = StopBits::One;

		serialPort1->ReadTimeout = 5000;
		serialPort1->WriteTimeout = 500;
		try {
			serialPort1->Open();
		}
		catch (System::Exception^ ex) {
			button5->BackColor = System::Drawing::Color::Red;
			MessageBox::Show("Unable to connect serial port: " + ex->Message);
			return;
		}
		button5->BackColor = System::Drawing::Color::Lime;
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
			String^ fileName = openFileDialog1->FileName;
			db = gcnew SQLiteConnection();
			try
			{
				db->ConnectionString = "Data Source=\"" + fileName + "\"";
				db->Open();
				DataTable^ table = fillDataTable("SELECT * FROM Devices;");
				dataGridView1->DataSource = table;
			}
			catch (Exception^ e)
			{
				MessageBox::Show("Error Working SQL: " + e->ToString(), "Exception ...");
			}
		}
}
```

```
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e) {
	try
	{
		String^ add = "4;";
		serialPort1->WriteLine(add);
		bool flag = true;
		String^ Mac_add;
		while (flag)
		{
			try
			{
			Mac_add = serialPort1->ReadLine();
			flag = false;
			}
			catch (TimeoutException^) {}
		}
		SQLiteCommand^ cmdInsertValue = db->CreateCommand();
		cmdInsertValue->CommandText = "INSERT INTO Devices (Mac, Room_num) VALUES('" +
Mac_add + "','" + maskedTextBox1->Text + "');";
		cmdInsertValue->ExecuteNonQuery();
		DataTable^ table = fillDataTable("SELECT * FROM Devices;");
		dataGridView1->DataSource = table;
	}
	catch (Exception^ e)
	{
		MessageBox::Show("Error Executing SQL: " + e->ToString(), "Exception ...");
	}
}
private: System::Void button7_Click(System::Object^ sender, System::EventArgs^ e) {
	//Номер выделенной строки
	int index = dataGridView1->CurrentCell->RowIndex;
	//Определим _id в выделенной строке
	String^ id_d = dataGridView1->Rows[index]->Cells["id_d"]->Value->ToString();
	try
	{
		SQLiteCommand^ cmdInsertValue = db->CreateCommand();
		cmdInsertValue->CommandText = "DELETE FROM Devices WHERE id_d = " + id_d + ";";
		cmdInsertValue->ExecuteNonQuery();
		DataTable^ table = fillDataTable("SELECT * FROM Devices;");
		dataGridView1->DataSource = table;
	}
	catch (Exception^ e)
```

```
		{
			MessageBox::Show("Error Executing SQL: " + e->ToString(), "Exception ...");
		}
	}
	private: System::Void button8_Click(System::Object^ sender, System::EventArgs^ e) {
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
			String^ fileName = openFileDialog1->FileName;
			db = gcnew SQLiteConnection();
			try
			{
				db->ConnectionString = "Data Source=\"" + fileName + "\"";
				db->Open();
				DataTable^ table = fillDataTable("SELECT * FROM Measurements;");
				dataGridView2->DataSource = table;
			}
			catch (Exception^ e)
			{
				MessageBox::Show("Error Working SQL: " + e->ToString(), "Exception ...");
			}
		}
	}
	private: System::Void button9_Click(System::Object^ sender, System::EventArgs^ e) {
		serialPort1->Close();
		button5->BackColor = System::Drawing::Color::Red;
	}
	private: System::Void button10_Click(System::Object^ sender, System::EventArgs^ e) {
		SQLiteCommand^ cmdSelect = db->CreateCommand();
		cmdSelect->CommandText = "SELECT Mac FROM Devices ORDER BY id_d";
		SQLiteDataReader^ reader = cmdSelect->ExecuteReader();
		String^ ssss;
		ssss = "1";
		if (reader->HasRows) {
			while (reader->Read()) {
				ssss += "," + reader->GetString(0);
			}
			ssss += ";";
			serialPort1->WriteLine(ssss);
			bool flag = true;
		}
	}
	private: System::Void button11_Click(System::Object^ sender, System::EventArgs^ e) {
		String^ opros = "3;";
```

```
serialPort1->WriteLine(opros);
bool flag = true;
String^ thdata;
SQLiteCommand^ cmdSelect = db->CreateCommand();
cmdSelect->CommandText = "SELECT id_d FROM Devices ORDER BY id_d";
SQLiteDataReader^ reader = cmdSelect->ExecuteReader();

List<long long>^ id_devices = gcnew List<long long>();
if (reader->HasRows) {
	while (reader->Read()) {
		id_devices->Add(reader->GetInt64(0));
	}
}
try
{
	thdata = serialPort1->ReadLine();
	thdata = thdata->Trim(String("\r\n;").ToCharArray());
	flag = false;
	array<String^>^ data = thdata->Split(String(",").ToCharArray());
	System::DateTime^ date_time = System::DateTime::Now;
	for (int i = 0; i < data->GetLength(0); i += 2)
	{
		auto provider = System::Globalization::CultureInfo::CreateSpecificCulture("en-US");
		double temperature = System::Convert::ToDouble(data[i], provider);
		double humidity = System::Convert::ToDouble(data[i+1], provider);
		SQLiteCommand^ cmdInsertValue = db->CreateCommand();
		cmdInsertValue->CommandText =
			"INSERT INTO Measurements (id_d, Temperature, Humidity, DateTime)
VALUES("
			+ id_devices[i/2] + ", '"
			+ temperature + "','"
			+ humidity + "', '"
			+ date_time->Year + "-" + (date_time->Month < 10 ? "0" : "") + date_time-
>Month + "-" + (date_time->Day < 10 ? "0" : "") + date_time->Day + " "
			+ (date_time->Hour < 10 ? "0" : "") + date_time->Hour
			+ ":" + (date_time->Minute < 10 ? "0" : "") + date_time->Minute
			+ ":" + (date_time->Second < 10 ? "0" : "") + date_time->Second
			+ "');";
		cmdInsertValue->ExecuteNonQuery();
	}
}
catch (TimeoutException^) {}
```

```
		DataTable^ table = fillDataTable("SELECT * FROM Measurements;");
		dataGridView2->DataSource = table;
}
private: System::Void button12_Click(System::Object^ sender, System::EventArgs^ e) {
		//Номер выделенной строки
		int index = dataGridView2->CurrentCell->RowIndex;
		//Определим _id в выделенной строке
		String^ id_d = dataGridView2->Rows[index]->Cells["id_d"]->Value->ToString();
		try
		{
			SQLiteCommand^ cmdInsertValue = db->CreateCommand();
			cmdInsertValue->CommandText = "DELETE FROM Measurements WHERE id_d = " + id_d +
";";
			cmdInsertValue->ExecuteNonQuery();
			DataTable^ table = fillDataTable("SELECT * FROM Measurements;");
			dataGridView2->DataSource = table;
		}
		catch (Exception^ e)
		{
			MessageBox::Show("Error Executing SQL: " + e->ToString(), "Exception ...");
		}
}
};
}
```

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: ilya.khlystov.tomsk.765@gmail.com / ID: 3948075
Проверяющий: (ilya.khlystov.tomsk.765@gmail.com / ID: 3948075)

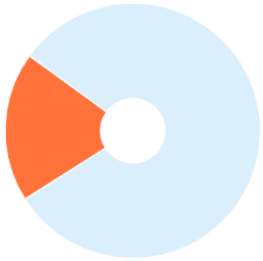
Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 25
Начало загрузки: 19.01.2022 13:13:26
Длительность загрузки: 00:00:01
Имя исходного файла: BKP.pdf
Название документа: BKP
Размер текста: 60 кБ
Символов в тексте: 60964
Слов в тексте: 7002
Число предложений: 258

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 19.01.2022 13:13:29
Длительность проверки: 00:00:06
Комментарии: не указано
Модули поиска: Интернет Free



ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
19,38%	0%	0%	80,62%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Доля в тексте	Источник	Актуален на	Модуль поиска	Блоков в отчете	Блоков в тексте
[01]	11,16%	15,84%	https://esu.citis.ru/ikrbs/BW5SQIYJVJ2DOU8VRSJZXLJS (1/2) https://esu.citis.ru	21 Мар 2018	Интернет Free	136	182
[02]	1,02%	13,45%	https://esu.citis.ru/ikrbs/EVWBUJYUUQLPIANOROKMD1CR (19/20) https://esu.citis.ru	21 Мар 2018	Интернет Free	13	167
[03]	4,38%	12,25%	Процедура цветовой калибровки для мультиспектральной системы медицинской диагностики http://library.eltech.ru	19 Сен 2019	Интернет Free	38	134

Еще источников: 7
Еще заимствований: 2,82%