МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра _____

«Допущен к защите»
Заведующий кафедрой _____
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Сборка домашней метеостанции на базе платформы Arduino и разработка программного обеспечения для неё

Специальность Вычислительная техника и программное обеспечение

Выполнил (а) Алиев М.М. ВТ-12-4
(Фамилия и инициалы) группа

Научный руководитель Ельгибаева Р.Б. ст. преподаватель
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекишева А.Д.
(Фамилия и инициалы, ученая степень, звание)
_____ «05» 05 2016 г.
(подпись)

по безопасности жизнедеятельности:
к.х.н., проф. Мазалов У.Ф.
(Фамилия и инициалы, ученая степень, звание)
_____ «29» 04 2016 г.
(подпись)

по применению вычислительной техники:
ст. преподаватель Ельгибаева Р.Б.
(Фамилия и инициалы, ученая степень, звание)
_____ «27» 05 2016 г.
(подпись)

Нормоконтролер: ст. преподаватель Ельгибаева Р.Б.
(Фамилия и инициалы, ученая степень, звание)
_____ «27» 05 2016 г.
(подпись)

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20__ г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Аэрокосмических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Алиев Мир-Амаль Мухтарович
(фамилия, имя, отчество)

Тема проекта Сборка домашней метеостанции на базе платформы Arduino и разработка программного обеспечения для неё

утверждена приказом ректора № 148 от «19» октября 2015 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта

Разработать систему домашней метеостанции

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:

1. Разработать структуру домашней метеостанции на базе платы Arduino
2. Собрать домашнюю метеостанцию
3. Разработать программное обеспечение для операционной системы Windows, позволяющее считывать данные метеостанции, сохранять их в базу данных и визуализировать
4. Разработать базу данных для хранения информации, снятой с метеостанции.
5. Разработать мобильное приложение, позволяющее подключаться к метеостанции и считать показания датчиков

Перечень графического материала (с точным указанием обязательных чертежей)

8 рисунков в главе I
8 рисунков в главе II
1 рисунок в главе III
5 рисунков в главе IV
3 рисунка в главе V
12 рисунков в главе VI
2 рисунка в глав VII
6 рисунков в разделе БЖД
1 рисунок в разделе Экономика

Рекомендуемая основная литература

Улли Соммер – „Программирование микроконтроллерных плат Arduino / Freeduino"
Герберт Шилдт – „C# 4.0 Полное руководство"
Изик Бен-Ган – „Microsoft SQL Server 2008. Основы T-SQL"

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Мазалов	10.04-29.04	
Экономика	Бекишева А.И.	1.03-29.04	
Глава 1	Ельгибаева Р.Б	3.01-29.02.2016	
Главы 2 и 3	Ельгибаева Р.Б	28.02-20.03.2016	
Глава 4, 5	Ельгибаева Р.Б	5.03-6.05.2016	
Главы 6 и 7	Ельгибаева Р.Б	5.03-6.05.2016	
Нормоконтроль	Ельгибаева Р.Б	20.05.2016	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Глава 1	3.01-29.02.2016	
2	Глава 2	29.02 - 20.03.2016	
3	Глава 3	28.02-20.03.2016	
4	Глава 4	5.03.-6.05.2016	
5	Глава 5	5.03-6.05.2016	
6	Глава 6	5.03-6.05.2016	
7	Глава 7	5.03 - 6.05.2016	
8	БЖД	1.03 -7.05.2016	
9	Экономика	1.03-7.05.2016	

Дата выдачи задания «10» октября 2015 г.

Заведующий кафедрой _____ Курабаев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Ержибаева Р.Б.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Алуев М.М.
(подпись) (Фамилия и инициалы)

Аннотация

Данный дипломный проект посвящен сборке домашней метеостанции и разработке программного обеспечения для неё. Основным предназначением программного обеспечения является считывание данных с метеостанции, их обработка и хранение. В процессе работы использовались следующие технологии: Arduino IDE, Visual Studio (C#, Windows Form), MS SQL Server, MIT App Invertor. В разделе обеспечения безопасности жизнедеятельности проведен анализ рабочего места, условий труда, сделаны расчёты эвакуации из помещения при пожаре, а также разработана система молниезащиты для здания.

Аңдатпа

Берілген дипломдық жұмысы үй метрологиялық станцияны құрастыруға және оған бағдарламалық қамтамасыз етуді әзірлеуге арналған. Бағдарламалық қамтамасыз етудің негізгі мақсаты метрологиялық станциядан деректерді оқу, өңдеу және сақтау болып табылады. Жұмыс істеу кезінде келесі технологиялар: Arduino IDE, Visual Studio (C #, Windows Form), MS SQL Server, MIT App Invertor пайдаланады. Жұмыс орнында өмірі қауіпсіздігі талдау, еңбек жағдайларын, есептеулер өрт кезінде үй-жайларды эвакуациялау, сондай-ақ құрылыс үшін арналған найзағайдан қорғау жүйесі жасалады. Тіршілік әрекетінің қауіпсіздігін қамтамасыз етуді бөлімінде жұмыс орнының, еңбек жағдайларын талдау жасалған, өрт болған жағдайда бөлмеден эвакуацияның есептеулер жасалған, сондай-ақ ғимараттар үшін жайқорған жүйесі құрылған.

Annotation

This thesis project is dedicated to the assembly of a home weather station and software development for it. The main purpose of the software is to read the data from the weather station, handling and storage. The following technologies were used during operation: Arduino IDE, Visual Studio (C #, Windows Form), MS SQL Server, MIT App Invertor. In the life safety section analysis of the workplace and working conditions, calculated evacuation from premises in case of fire, as well as the designed lightning protection system for the building.

Содержание

Введение

На рынке сравнительно недавно стали появляться домашние метеостанции. Функционально домашняя метеостанция схожа с полноценной метеорологической станции, за исключением того, что она обрабатывает меньше данных поступающих от различных датчиков, которые могут быть расположении в помещении или же за его пределами. Обычно домашняя метеостанция показывает температуру внутри помещения, а также за его пределами, атмосферное давление, относительную влажность воздуха и исходя из полученных данных некоторые модели метеостанции вычисляют предположительный прогноз погоды на ближайшие сутки. [1]

Обычный человек практический каждый день выходит на улицу и обычно он полагается на данные полученные от городских метеостанций. Но погода может неожиданно измениться и жаркий солнечный день может смениться дождливым или пасмурным. Городские метеостанции обычно дают данные о прогнозе погоды вечером на кануне прогнозируемого дня. Если направление ветра изменится, то погода может резко поменяться. В этом случае можно использовать домашнюю метеостанцию. Получив данные о влажности воздуха и давлении можно точнее предположить прогноз погоды на ближайшие часы. В крупных городах данный вопрос стоит весьма остро. Это связано с тем, что в больших городах в различных частях города может быть различная погода. Одним из таких городов является Алматы. В нашем городе Алматы ситуация усугубляется наличием гор. В горной местности облака скапливаются намного быстрее и иногда не могут преодолеть горные вершины. В связи с этим в горах близ нашего города может быстро скопиться большое количество облаков и может неожиданно пойти дождь. В таком случае локальная метеостанция, расположенная неподалёку от горной местности может предупредить жителей города о предстоящем дожде. В таких случаях обычно в верхних частях города начинается дождь, в то время как в нижних частях ярко светит солнце и нет никаких намёков на дождливую погоду. Так же метеостанция будет полезна при планировании походов в горы или другие места для отдыха на природе. Метеостанцию можно установить в подобном месте для отдыха и подключить к глобальной сети, где она в режиме онлайн может предоставлять данные на специальные ресурсы. Люди, планирующие поход или отдых на природе, смогут зайти на ресурс и узнать метеоданные для необходимой им локации. Уже существуют специализированные сайты, где можно посмотреть данные, получаемые с домашних метеостанций различных городов мира. К сожалению, в нашем городе нет умельцев, которые собирают метеостанции и транслируют данные с них в интернет. [2]

Так же домашнюю метеостанцию можно использовать в помещении. В отличии от обычных термометров, домашняя метеостанция даст информацию об уровне влажности и давления. Для этого в домашних метеостанциях обычно устанавливаются датчики влажности или гигрометры, а также датчики

давления. Так же существуют модификации метеостанций имеющие датчики углекислого газа. Зная уровень углекислого газа можно предположить необходимость проветривания помещения. А в зависимости от уровня влажности можно включить или выключить домашний увлажнитель воздуха.

Так же существуют помещения, где установка кондиционеров запрещена. Примерами таких помещений являются больницы и детские сады. В таких помещениях необходимо регулировать температуру воздуха и влажность более безопасными способами, какими являются проветривание помещения или же установка небольших вентиляторов. Для более точного регулирования данных параметром можно использовать домашнюю метеостанцию.

Помимо самой метеостанции в данной дипломной работе так же было разработано программное обеспечение для операционной системы Windows и Android. С помощью данного программного обеспечения можно просматривать данные, выводить таблицы с данными, а также строить графики. В современном мире быстро развивается наука. Одной и существующих научных дисциплин является метеорология. Основной задачей метеоролога является изучение метеорологических данных в различных местностях. Обычно подобные данные измеряются специальными приборами, после чего записываются метеорологом в виде таблиц или каком-либо другом виде. Благодаря программному обеспечению данные записываются в базу данных автоматических. Метеорологу позже остаётся только считать их базы данных и произвести необходимые ему вычисления.

1 Обзор используемых технологий

В ходе разработки программного обеспечения для метеостанции использовались следующие технологии.

- Плата Arduino Uno;
- Arduino IDE - Среда разработки Arduino;
- Visual Studio (C#, Windows Form) - Создание программного обеспечения для операционной системы Windows, позволяющего считать данные с платформы Arduino, обработать их, записать в базу данных и построить графики;
- MS SQL Server – СУБД. Использовалась для хранения данных;
- MIT App Invertor - среда визуальной разработки android-приложений.

1.1 Arduino Uno

В качестве рабочего контроллера использовалась плата Arduino Uno. Arduino Uno является микроконтроллер платы основан на ATmega328 (техническое описание). Он имеет в наличии 14 цифровых ввода / вывода общего назначения (6 из которых могут быть использованы как выходы ШИМ), 6 аналоговых входов, 16 МГц керамические резонатор, соединение USB, разъем питания, программатор ICSP, и кнопку сброса. Он содержит все необходимых для поддержки микроконтроллера; необходимо просто подключить его к компьютеру с помощью кабеля USB, к адаптеру постоянного тока или батарее, чтобы начать работу. [3]

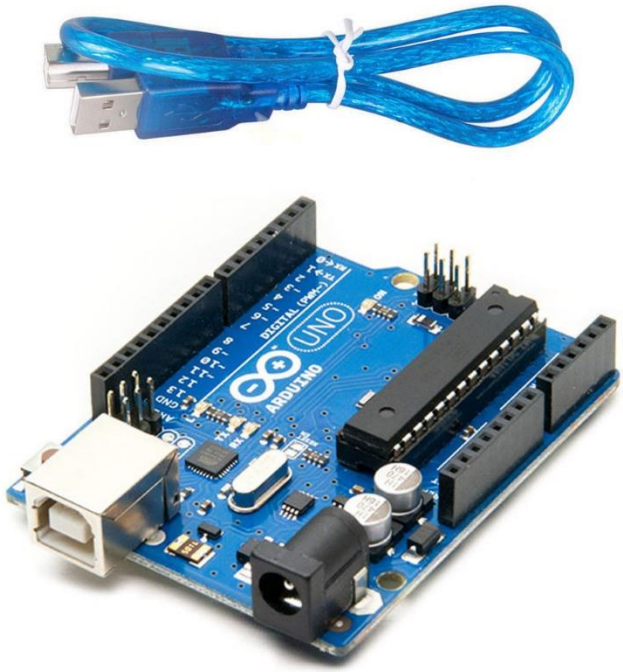

Рисунок 1 – Arduino Uno

Uno отличается от всех предшествующих её плат тем, что она не использует USB-Serial порт микросхемы драйвера FTDI. Вместо этого, она имеет Atmega16U2 (Atmega8U2 до версии R2) запрограммирован как USB-Serial конвертер.

Вторая версия платы Uno имеет резистор, перемещающий линию 8U2 HWB на землю, что упрощает установку платы в режим DFU.

Третья версия платы имеет следующие новые функции:

Новая распиновка: добавлены контакты SDA и SCL, которые находятся рядом с контактом AREF и два других новых контакта расположенные рядом с кнопкой сброса, в IOREF, которые позволяют дополнительным шилдам использовать напряжение с платы. В будущем, шилды будут совместимы как с платой, которые используют AVR, которые работают с напряжением 5В и с Arduino Due, которая работает под напряжением 3.3В. Второй особенностью является не подключенный PIN-код, который зарезервирован для будущих целей. Так же в третьей версии Arduino Uno микропроцессор 8U2 заменён на Atmega 16U2.

"Uno" итальянское название, данное в честь предстоящего выпуска Arduino 1.0. Uno и Arduino версии 1.0 являются эталонными версиями Arduino, которые ещё развиваются. Uno является последние версией Arduino в серии с USB подключением.[4]

Технические параметры платы Arduino Uno:

микроконтроллер ATmega328

Рабочее используемое напряжение 5V

Входное напряжение (рекомендуется) 7-12V

Входное напряжение (пределы) 6-20V

Цифровые креиакты14 (6 из которых обеспечивают выход ШИМ)

Входные аналоговые контакты 6

Ток на контактах I/O 40 мА

Постоянный ток для контактов 3.3В 50 мА

Флэш-память 32 Кб (ATmega328), из которых 0,5 КБ используется загрузчиком

SRAM 2 Кб (ATmega328)

EEPROM 1 КБ (ATmega328)

Тактовая частота 16 МГц [3]

Питание.

Arduino Uno может получать питание через подключение USB или от внешнего генератора питания. Сила источник выбирается автоматически. Внешнее (без USB) питание может исходить либо адаптера переменного тока в постоянный ток, либо от батареи. Адаптер может быть подключение помощью 2.1 мм штекером в гнездо питания платы. Провода от батареи могут быть вставлены в Gnd и Vin штекерные разъемы питания.

Плата может работать на внешнего источника питания от 6 до 20 вольт. Если поставляется менее чем 7В, то пятивольтовый переходник может поставить менее пяти вольт и плата может быть неустойчивым. При

использовании более чем 12В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон составляет от 7 до 12 вольт.

Имеются следующие выводы питания:

VIN. Входное напряжение к плате Arduino, когда она использует внешний источник питания (например, 5 вольт от подключения к USB или другого регулируемого источника питания). Вы можете настроить напряжение через этот контакт, или, если подачи напряжения происходит через специальный разъем питания, то обеспечить подачу питания через этот контакт.

5В. Этот контактный разъем выводит регулируемый 5V от регулятора на плате. Плата может питаться либо от гнезда DC питания (7 - 12), либо от разъема USB (5V), либо через VIN контакт платы (7-12V). Подача напряжения через 5В или 3.3В контакт обходит регулятор и может повредить плату. [4]

3.3 В. Подача 3,3 вольт генерируется регулятором на борту. Максимальный ток составляет 50 мА.

GND. Выводы заземления.

Память

ATmega328 имеет 32 КБ памяти (из них 0,5 КБ используется для загрузчика). Она также имеет 2 КБ для SRAM (статическое ОЗУ) и 1 КБ для EEPROM (который можно читать и записывать с помощью библиотеки EEPROM).

Ввод и вывод

Каждый из 14 цифровых выводов Uno может использоваться как вход или выход, используя функции pinMode(), digitalWrite() и digitalRead(). Они работают на под напряжением 5 вольт. Каждый контакт может обеспечить или получить максимум 40 мА и имеет внутренний подтягивающий резистор (отключен по умолчанию) 20-50 кОм. В

Кроме того, некоторые выводы имеют особые функции:

Сериальный контакты: 0 (RX) и 1 (TX). Используется для получения (RX) и передачи данных (TX) TTL. Эти контакты соединены с соответствующими выводами USB-TTL серийного чипа ATmega8U2.

Контакты внешнего прерывания: 2 и 3. Эти контакты могут быть сконфигурированы, чтобы вызвать прерывание при низких значениях или изменять значения.

ШИМ(Широтно-Импульсная Модуляция): 3, 5, 6, 9, 10 и 11. Обеспечить 8-битный выход ШИМ с функцией analogWrite().

SPI(последовательный периферийный интерфейс): 10 (CC), 11 (MOSI), 12 (MISO), 13 (SCK). Эти контакты поддерживают SPI связь с использованием библиотеки SPI.

LED: 13. Существует встроенный светодиод, подключенный к цифровому выводу 13. Если на этот контакт подаётся большое значение, то ветодиод горит, если небольшое значение, он выключен.

Uno имеет 6 аналоговых входов, помеченных от A0 до A5, каждый из которых обеспечивают 10 битов разрешения (т.е. 1024 различных значений). По умолчанию они измеряют от земли до 5 вольт, хотя это может изменить верхний конец их диапазона с использованием PIN-кода ARef и функции analogReference(). [3] Кроме того, некоторые контакты имеют специализированные функции:

TWI(I2C): A4 или SDA контактн и A5 или SCL контакт. Поддержка TWI связи с использованием библиотеки Wire.

Есть несколько других контактов на плате:

AREF. Опорное напряжение для аналоговых входов. Используется с помощью фукнции analogReference().

Сброс. Доведите эту строку до низкого значения для перезагрузки микроконтроллера. Обычно используется для добавления кнопки сброса на шилд, которые блокируют один на борту.

Связь

Arduino Uno имеет ряд устройств для осуществления связи с компьютером, другой платой Arduino, или другим микроконтроллером. ATmega328 обеспечивает последовательную связь, которая доступна на цифровых выводов 0 (RX) и 1 (TX). ATmega16U2 расположенный на плате обеспечивает сериальный обмен данными через USB и благодаря программному обеспечению появляется как виртуальный COM порт компьютере. Прошивка 16U2 использует стандартные драйверы USB COM и внешний драйвер не требуется. Тем не менее, на Windows для подключения необходим файл типа .inf. Программное обеспечение Arduino включает в себя серийный монитор, который позволяет получать от платы Arduino и отправлять на неё простые текстовые данные. Светодиоды RX и TX на плате будут мигать, когда данные передаются через чип и подключён через USB к компьютеру. [3]

Библиотека SoftwareSerial позволяет обеспечить связь с любым из цифровых выводов Uno через сериальный порт.

ATmega328 поддерживает интерфейсы I2C (TWI) и SPI связи. Программное обеспечение Arduino включает в себя библиотеку Wire для упрощения использования шины I2C. Для SPI связи используется библиотека SPI.

Программирование

Arduino Uno может быть запрограммирован с помощью программного обеспечения Arduino. Выберите "Arduino Uno из Инструменты> Совет меню (в зависимости от микроконтроллера на плате).

Микропроцессор ATmega328 для Arduino Uno поставляется с загрузчиком, который позволяет загружать новый код для него без использования внешних программаторов. Он связывается с помощью оригинального протокола. Вы также можете использовать загрузчик и запрограммировать микроконтроллер через ICSP (In Circuit Serial Programming).

Исходный код прошивки для ATmega16U2 (или 8U2 на плате версий 1 и 2) находится в открытом доступе. ATmega16U2 / 8U2 загружается с загрузчиком DFU, который может быть активирован с помощью: На платах 1й версии: подключением припоя перемычки на задней стороне платы и дальнейшей перезагрузкой 8U2; на платах 2й или более поздней версии: есть резистор, который тянет линию 8U2 / 16U2 HWB на землю, что позволяет легко установить микропроцессов в режим DFU. Затем вы можете использовать программное обеспечение FLIP компании Atmel (Windows) или DFU (Mac OS X и Linux), чтобы загрузить новую прошивку. Или вы можете использовать ISP с внешним программатором (для перезаписи DFU загрузчиком). [4]

Автоматическая (программная) перезагрузка

Вместо того, чтобы требовать физической нажатие кнопки сброса перед загрузкой, то Arduino Uno сконструирована таким образом, что может быть перезагружена при помощи программного обеспечения, работающего на подключенном компьютере. Программное обеспечение может управлять потоком линий (DTR) микропроцессора ATmega8U2/16U2, который подключается к линии сброса ATmega328 через 100 нФ конденсатор. Программное обеспечение Arduino использует эту возможность, чтобы позволить вам загружать код, просто нажав кнопку загрузки в среде Arduino. Это означает, что загрузчик может иметь более короткий тайм-аут, поскольку снижение DTR может быть хорошо скоординировано с началом загрузки.

Эта установка имеет и другие последствия. Когда Uno подключен к любому компьютеру под управлением Mac OS X или Linux, он сбрасывает каждый раз, когда установлено соединение к нему с помощью программы (через USB).

В то время как микропроцессор запрограммирован, чтобы игнорировать неверные данные (то есть ничего, кроме выгрузки нового кода), микропроцессор перехватит первые несколько байт данных, передаваемых на плаьу после подключения.

Uno содержит настройку, которую можно использовать чтобы отключить автоматическую перезагрузку. Подушечки по обе стороны от трассы могут быть спаяны вместе, чтобы снова включить его. Они промаркированы как "RESET-EN". Можно также отключить автоматический сброс путем подключения резистор 110 Ом от 5V к линии сброса.

Защита от перегрузки по току USB

Arduino Uno имеет самовосстанавливающийся предохранитель, который защищает порты USB вашего компьютера от перегрузки по току. Несмотря на то, что большинство компьютеров обеспечивают свою собственную внутреннюю защиту, предохранитель обеспечивает дополнительный уровень защиты. Если по порту USB применяется более чем 500 мА, то предохранитель автоматически разорвёт соединение до устранения короткого замыкания.

Ещё одним полезным свойством Arduino является возможность подключения плат расширения, так называемых shields или просто «шилдов». Это специальные дополнительные платы, которые ставятся подобно слоям бутерброда поверх Arduino, чтобы дать ему новые возможности. Так, например, существуют платы расширения для подключения к локальной сети и интернету (Ethernet Shield), для управления мощными моторами (Motor Shield), для получения координат и времени со спутников GPS (модуль GPS) и многие другие.

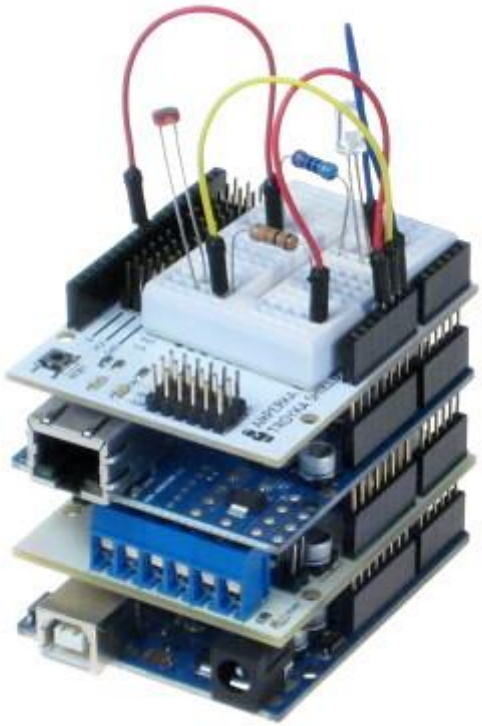

Рисунок 2 – Arduino с дополнительными шилдами

1.2 Arduino IDE

Для удобства работы с Arduino существует бесплатная официальная среда программирования «Arduino IDE», работающая под Windows, Mac OS и Linux. С помощью неё загрузка новой программы в Arduino становится делом одного клика, только лишь подключите плату к компьютеру через USB. Хотя для более пытливых умов возможна работа и через Visual Studio, Eclipse, другие IDE или командную строку. [3]

Полноценные устройства можно собирать, используя специальную макетную плату, перемычки и провода абсолютно без пайки. Конструирование ещё не было таким быстрым и простым.

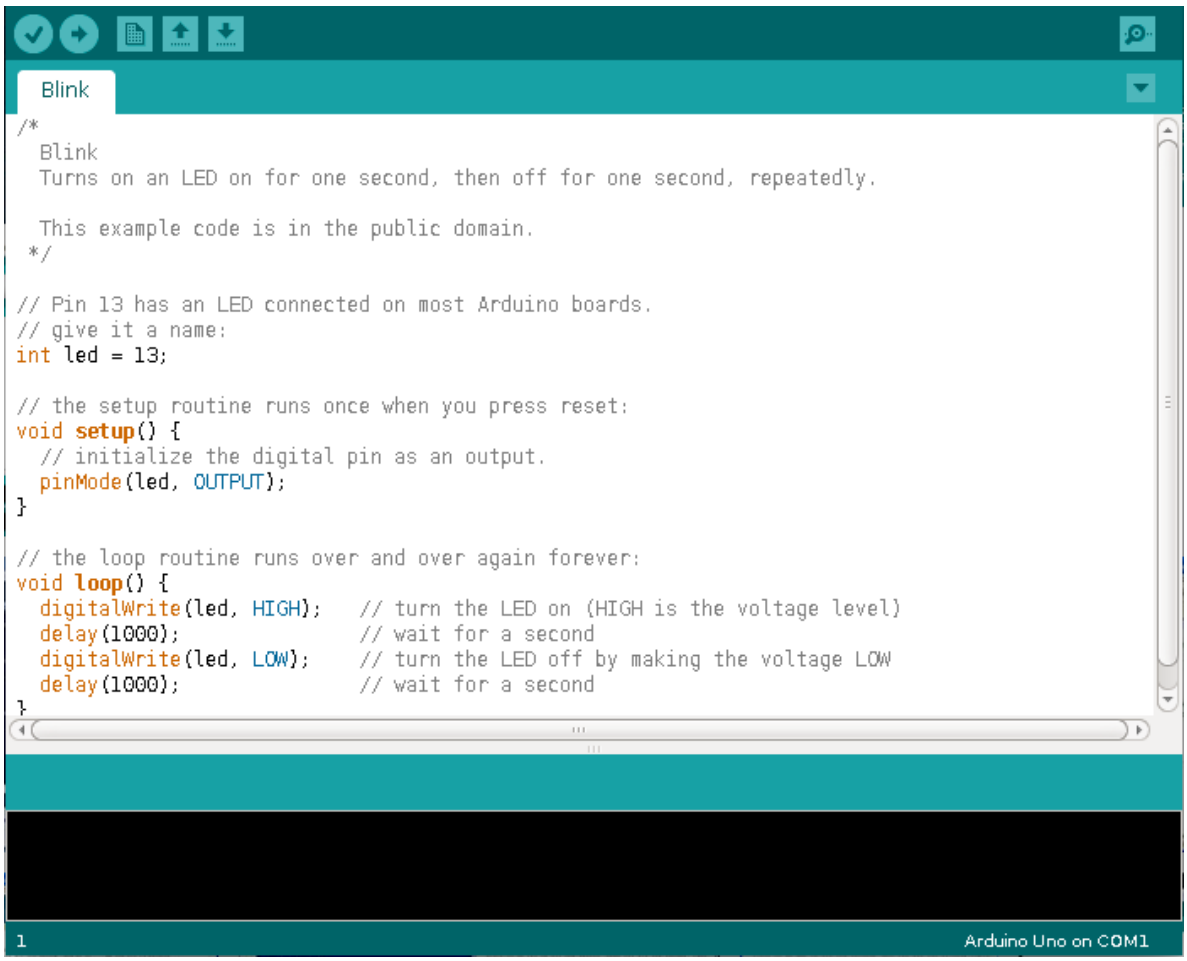

Рисунок 3 – Интерфейс Arduino IDE

Breadboard. При сборке метеостанции использовался беспаячный метод соединения деталей. Подобный метод сборки осуществляется с помощью макетной платы. Благодаря макетной плате к одному входу платформы Arduino можно подсоединить несколько различных устройств. Так же благодаря данному методу сборки можно без проблем заменить при необходимости какой-либо датчик или же подключить новые датчики.

1.3 Microsoft SQL Server

Microsoft SQL Server является реляционной системой управления базами данных, разработанной компанией Microsoft. В качестве сервера баз данных, это программный продукт с основной функцией хранения и извлечения данных в соответствии с запросом других программных приложений, которые могут работать либо на том же компьютере или на другом компьютере через сеть (в том числе в сети Интернет).

На рынке Microsoft по меньшей мере десяток различных выпусков Microsoft SQL Server, ориентированных на различную аудиторию и для рабочих нагрузок, начиная от небольших приложений одной машины до

крупных интернет-приложениям, с большим количеством одновременно работающих пользователей.

Microsoft делает SQL Server доступны в нескольких изданиях, с различными наборами функций и ориентации различных пользователей

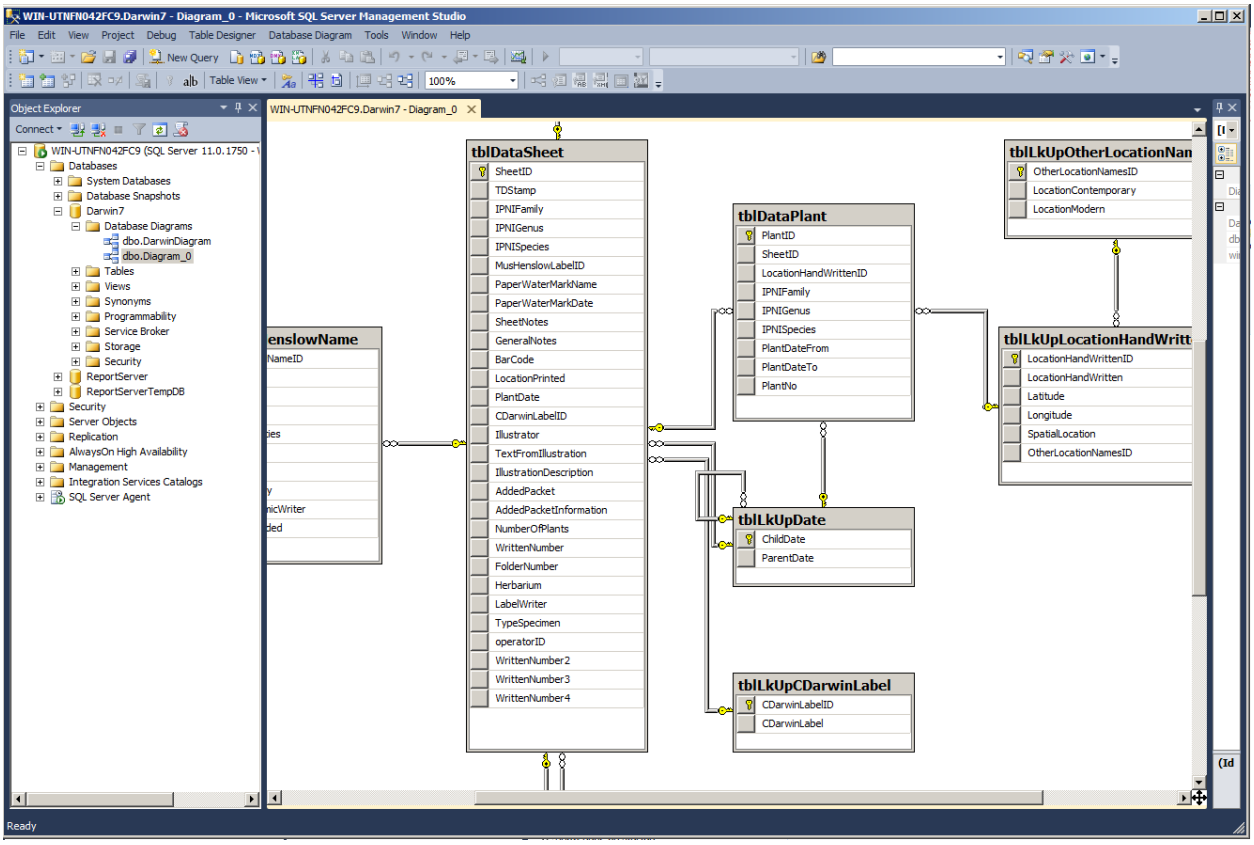

Рисунок 4 – Интерфейс MS SQL Server

Datacenter SQL Server 2008 R2 Datacenter является полнофункциональным выпуском SQL Server и предназначен для центров обработки данных, которые нуждаются в высоком уровне поддержки и масштабируемости приложений. Он поддерживает 256 логических процессоров и практически неограниченный объём памяти и поставляется с StreamInsight Premium Edition. Издание Datacenter было ликвидировано в SQL Server 2012; все его функции доступны в SQL Server 2012 Enterprise Edition. [5]

Enterprise. SQL Server Enterprise Edition включает в себя как ядро базы данных ядра и дополнительных услуг, с набором инструментов для создания и управления кластера SQL Server. Он может управлять базами данных размером до 524 петабайт и адресом до 2 терабайт и поддерживает 8 физических процессоров. SQL Server 2012 Enterprise Edition поддерживает 160 физических процессоров.

Standard SQL Server Standard Edition включает в себя ядро базы данных ядра, а также автономных служб. Она отличается от корпоративной версии тем, что она поддерживает меньшее количество активных экземпляров (количество узлов в кластере) и не включает в себя некоторые функции

высокой доступности, такие как добавление памяти (что позволяет памяти быть добавленой, пока сервер все еще работает), и параллельные индексы.

Web SQL Server Web Edition является низким TCO вариант для веб-хостинга.

Business Intelligence.

Введенный в SQL Server 2012 и сосредоточившая внимание на самообслуживании и корпоративной бизнес - аналитике. Она включает в себя возможности Standard Edition и инструментов Business Intelligence: PowerPivot, Power View, БИ семантической модели, Master Data Services, услуги Качество данных и xVelocity в памяти аналитики. [5]

Архитектура.

Слой протокола реализует внешний интерфейс для SQL Server. Все операции, которые могут быть вызваны на SQL Server передаются ему в формате, определенном Microsoft, называемым табличный поток данных (TDS). TDS является протоколом прикладного уровня, используемый для передачи данных между сервером базы данных и клиентом. Изначально разработанная в Sybase Inc, Sybase SQL Server реляционная база данных в 1984 году, а затем с помощью Microsoft в Microsoft SQL Server, TDS пакеты могут быть помещены в другие физические транспортные протоколы, включая TCP / IP, именованные каналами и общей памятью. Следовательно, доступ к SQL Server предоставляется с помощью этих протоколов. Кроме того, SQL Server API работает как веб–служба.

Хранилище данных представляет собой базу данных, которая представляет собой набор таблиц с типизированными столбцами. SQL Server поддерживает различные типы данных, в том числе первичных типов, таких как Integer, Float, Decimal, Char (в том числе символьных строк), Varchar (строки переменной длины символов), двоичный (для неструктурированных блоков данных), текст (для текстовых данных) и другие.

Microsoft SQL Server также позволяет создавать составные типы, определяемые пользователем (UDT), которые будут определены и использованы. Это также делает статистику сервера, доступной в виде виртуальных таблиц и представлений (так называемые динамические административные представления или DMVs). В дополнение к таблицам, база данных может также содержать другие объекты, в том числе взгляды, хранимые процедуры, индексы и ограничения, а также журнал транзакций. База данных SQL Server может содержать максимум 231 объектов, и может охватывать несколько файлов на уровне операционной системы с максимальным размером файла 260 байт (1 экзабайт). Данные в базе данных хранятся в первичных файлах данных с расширение mdf. Вторичные файлы данных, которые были определены с ndf расширением, используются, чтобы данные единой базы данных можно было распространяться на более чем один файл и, возможно, в более чем одной файловой системы. Файлы журнала идентифицируются с ldf расширением. [6]

Дисковое пространство выделяется в базу данных делится на последовательно пронумерованных страниц, каждая по 8 КБ. Страница является основной единицей ввода / вывода для операций SQL Server. Страница имеет 96-байтный заголовок, который хранит метаданные о странице, включая номер страницы, тип страницы, свободное пространство на странице и идентификатор объекта, которому они принадлежат. Тип страницы определяет данные, содержащиеся на странице: данные, хранящиеся в базе данных, индекса, карта распределения, которая содержит информацию о том, как страницы выделяются таблицы и индексы, карта изменения которая содержит информацию об изменениях, внесенных в другие страницы с момента последнего резервного копирования или протоколирования, или содержат большие типы данных, такие как изображение или текст. В то время как страница является основной единицей операции ввода/вывода, пространство на самом деле удалось с точки зрения степени, которая состоит из 8 страниц. Объект базы данных может либо охватывать все 8 страниц в степени ("равномерной степени") или доли протяженностью до более 7 объектов ("смешанной степени"). Строка в таблице базы данных не может занимать более одной страницы, поэтому ограничивается 8 Кбайт. Тем не менее, если данные превышает 8 КБ, и строка содержит VARCHAR или VARBINARY данные, данные в этих столбцах будут перемещены на новую страницу (или, возможно, последовательности страниц, называется Allocation блок) и указатели буду заменены на данные.

Для физического хранения таблицы, ее строки разделены на ряд разделов (пронумерованных от 1 до N). Размер раздела определяется пользователем. По умолчанию все строки находятся в одном разделе. Таблица разделена на несколько разделов, чтобы распространить базу данных по компьютерному кластеру. Строки в каждом разделе хранятся либо в B-дереве или куче структуры. Если таблица имеет связанный с ним, кластерный индекс, чтобы обеспечить быстрое извлечение строк, строки хранятся в порядке, в соответствии с их значениями индекса B-дерева, обеспечивающего индекс. Данные в листовом узле листьев и других узлов, хранящих значения индекса для данных листа достижимости из соответствующих узлов. Если индексы не кластерные, строки не сортируются в соответствии с индексными ключами. Индексированный вид имеет ту же структуру хранения как индексированные таблицы. Таблица без кластерного индекса хранится в неупорядоченной структуре кучи. Тем не менее, таблица может иметь не кластерных индексы, чтобы обеспечить быстрое извлечение строк. В некоторых ситуациях структура кучи имеет преимущества в производительности по сравнению с кластерной структурой. Обе кучи и B-деревья могут охватывать несколько единиц распределения. [5]

В SQL Server существуют буферы страниц в оперативной памяти для минимизации дискового ввода/вывода. Любая 8 KB страница может быть помещена в буфер в оперативной памяти, а также множество всех страниц, в настоящее время буферизованных называется кэшем буфера. Объем памяти,

доступной для SQL Server решает, сколько страниц будет сохраняться в памяти. Кэш - буфер управляется менеджером буфера. Чтение или запись на любой странице копирует данные в кэш буфера. Последующие операции чтения или записи перенаправляются на копии в памяти, а не на версии, оставшиеся на диске. Страница обновляется на диске буфером менеджера, только если в памяти кэш не ссылается на какое-то время. Во время написания страницы обратно на диск, используется асинхронный ввод/вывод при котором операция ввода/вывода выполняется в фоновом потоке, чтобы другие операции ждали операцию ввода/вывода для завершения. Каждая страница записывается вместе с её контрольной суммой в момент записи. При чтении страницы, её контрольная сумма вычисляется снова и согласуется с сохраненной версией, чтобы убедиться в не поврежденности страницы.

1.4 Visual Studio (C#, Windows Form)

Microsoft Visual Studio представляет собой интеграционное окружение для разработки (IDE) от компании Microsoft. Она используется для разработки программ для компьютеров для Microsoft Windows, а также веб-сайтов, веб-приложений и веб-сервисов. Visual Studio использует платформы разработки программного обеспечения Microsoft, такие как Windows API, Windows Forms и другие. Она может производить как машинный код, так и управляемый код.

Visual Studio включает в себя редактор кода, поддерживающий технологию IntelliSense (компонент завершения кода), а также рефакторинга кода. Интегрированный отладчик работает и как отладчик исходного уровня и как отладчик на уровне машины. Другие встроенные инструменты включают в себя конструктор форм для построения GUI приложений, веб-дизайнер, класс дизайнера и проектировщик схемы базы данных. Он принимает плагины, которые расширяют функциональность почти на каждом уровне, в том числе добавление поддержки для систем источника управления (например, Subversion) и добавление новых наборов инструментов, как редакторы и визуальные конструкторы для предметно-ориентированных языков или наборов инструментов для других аспектов жизненного цикла разработки программного обеспечения (как клиент Team Foundation Server: Team Explorer). [7]

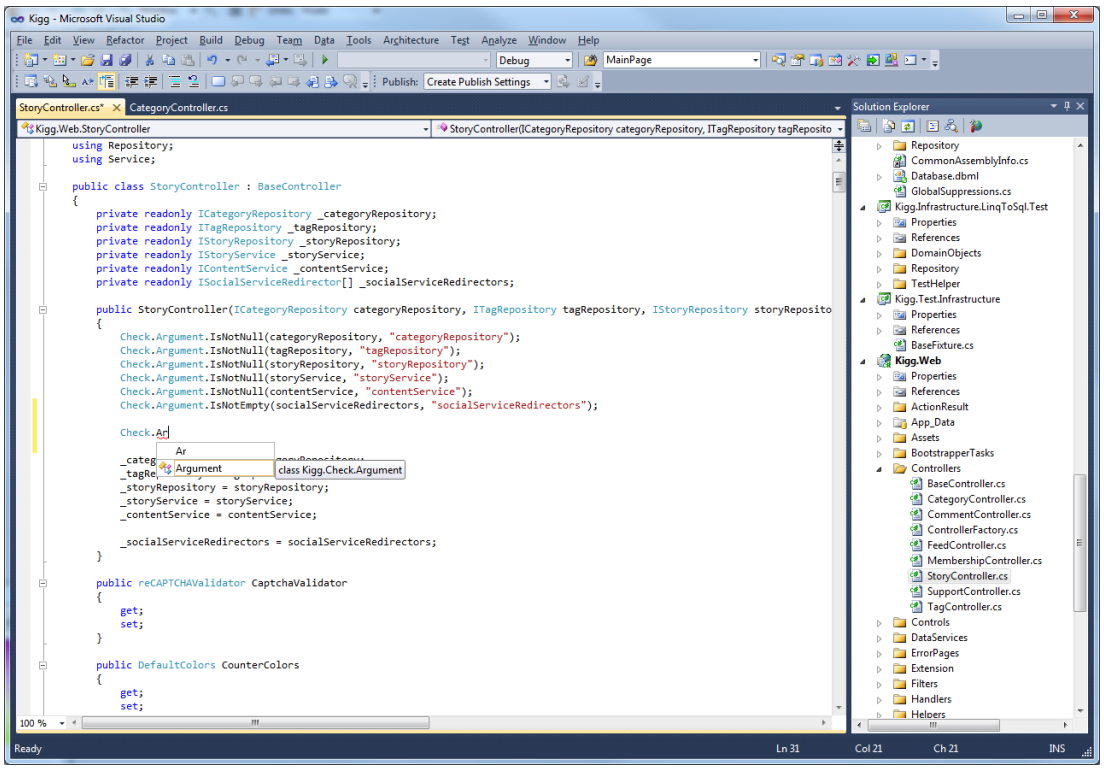

Рисунок 5 – Интерфейс MS Visual Studio

Visual Studio поддерживает различные языки программирования и имеет редактор кода и отладчик для поддержки (в разной степени) практически любого языка программирования. Встроенные языки включают C, C ++ и C ++ / CLI (с помощью Visual C ++), VB.NET (Visual Basic .NET), C # (Visual C #) и F # (как в Visual Studio 2010). Поддержка других языков, таких как Python, Ruby, Node.js и M среди прочего осуществляется с помощью языковых служб, установленных отдельно. Он также поддерживает XML / XSLT, HTML / XHTML, JavaScript и CSS. Java (и J #). [8]

До появления Visual Studio 2015 коммерческие версии Visual Studio были доступны бесплатно для студентов через программу DreamSpark Microsoft и только коммерческие версии поддерживали работу с плагинами. Начиная с Visual Studio 2015 Microsoft начала выпуск программного обеспечения, которое поддерживает плагины, без каких-либо затрат для всех пользователи.

Visual Studio не поддерживает ни одного языка программирования, решения или внутренние инструменты. Вместо этого, она позволяет плагину функционировать закодированным в виде VSPackage. При установке вся функциональность доступна в качестве сервиса. IDE предоставляет три услуги: SVsSolution, которая предоставляет возможность перечислить проекты и решения; SVsUIShell, которая обеспечивает оконную и UI функциональность (включая вкладки, панели инструментов и окна инструментов); и SVsShell, которая занимается регистрацией VSPackages. Кроме того, IDE также отвечает за координацию и создание благоприятной связи между различными службами. Все редакторы, дизайнеры, типы

проектов и другие инструменты реализованы в виде VSPackages. Visual Studio использует COM порт для доступа к VSPackages. Visual Studio SDK также включает в себя Managed Package Framework (MPF), который представляет собой набор управляемых оберток вокруг COM-интерфейсов, которые позволяют возможность работать пакетам, написанным на любом CLI совместимом языке. Однако MPF не предоставляет всю функциональность визуальных интерфейсов COM Visual Studio. Можно использовать специальные сервисы для создания других пакетов, которые добавляют функциональные возможности к среде разработки Visual Studio. [8]

Поддержка языков программирования добавляется с использованием специального VSPackage называемого лингвистической служба. Служба языка определяет различные интерфейсы, которые VSPackage может реализовать, чтобы добавить поддержку различных функций. Функциональности, которые могут быть добавлены таким образом, включают в себя подсветку синтаксиса, завершение выражений, согласование скобок, информацию о параметрах всплывающие подсказки, списки пользователей и маркеры ошибок. Если интерфейс реализован, то функциональные возможности будут доступны для языка программирования. Языковые услуги должны осуществляться на основе каждого языка программирования. Реализации могут повторно использовать код из синтаксического анализатора или компилятор для языка. Языковые услуги могут быть реализованы либо в машинный код, либо в управляемый код. Для машинного кода, может быть использован как родные COM интерфейсы или Babel Framework (часть Visual Studio SDK). Для управляемого кода, MPF включает в себя оболочки для написания служб управляемых языков.

Visual Studio не включает в себя поддержку управления версиями встроенную в систему, но она определяет два альтернативных пути для систем управления исходным кодом для интеграции с IDE. Источник управления VSPackage может предоставить свои собственные настройки пользовательского интерфейса. В отличие от этого, плагин управления источником с использованием MSSCCI (Microsoft Source Code Interface Control) предоставляет набор функций, которые используются для реализации различных функциональных возможностей для управления исходным кодом, с помощью стандартного пользовательского интерфейса Visual Studio. MSSCCI впервые был использован для интегрирования Visual SourceSafe с Visual Studio 6.0, но позже был добавлен в Visual Studio SDK. Visual Studio .NET 2002 используется MSSCCI 1.1 и Visual Studio .NET 2003 используется MSSCCI 1.2. Visual Studio 2005, 2008 и 2010 годов используют MSSCCI Версия 1.3, которая имеет поддержку для переименования и удаления, а также асинхронного открытия.

Visual Studio поддерживает запуск нескольких экземпляров окружающей среды (каждый со своим собственным набором VSPackages). Экземпляры используют разный реестр, чтобы сохранить их состояние конфигурации и различаются по их APPID (Application ID). Экземпляры

запускаются с помощью APPID конкретных EXE-файлов, которые выбирают AppId, устанавливает корневой улей и запускает IDE. VSPackages зарегистрированные с помощью одного AppId интегрированы с другими VSPackages для этого AppId. Различные издания продуктом Visual Studio создаются с использованием различных типов AppID. Издание продукта Visual Studio Express установлено со своими собственными типом AppID, но Standard, Professional и Team Suite имеют одни и те же AppId. Следовательно, можно установить Express Editions бок о бок с другими изданиями, в отличие от других изданий, которые обновляют ту же установку. Профессиональное издание включает в себя подмножество VSPackages в стандартном издании и люкс команда включает в себя подмножество VSPackages в обоих других изданиях. Система AppId начала использоваться в Visual Studio Shell в Visual Studio 2008.

Как и любая другая IDE, Visual Studio включает в себя редактор кода, поддерживающий подсветку синтаксиса и завершение кода с помощью IntelliSense для переменных, функций, методов, циклов и запросов LINQ. IntelliSense поддерживается для включенных языков, а также для XML и для каскадирования таблиц стилей и JavaScript при разработке веб-сайтов и веб-приложений. автозаполнение предложения появляются в немодальном списке над окном редактора кода, в непосредственной близости от курсора. В Visual Studio 2008 года, это автозаполнение может быть сделано временно полупрозрачным, чтобы увидеть код замаскированный по ним. Редактор кода используется для всех поддерживаемых языков.

Редактор кода Visual Studio также поддерживает настройки закладок в коде для быстрой навигации. Другие навигационные средства включают свертывание блоков кода и инкрементальный поиск, в дополнение к обычному поиску текста и регулярным выражениям поиска. Редактор кода также включает в себя нескольких элементов буфера обмена и список задач. Редактор кода поддерживает фрагменты кода, которые сохраняют шаблоны для повторяющегося кода, который может быть вставлен в код и настроен для рабочего проекта. Так же встроены инструменты управления фрагментами кода. Эти инструменты, появляются как всплывающие окна, которые могут автоматически скрываться, когда не используются. Редактор кода Visual Studio также поддерживает рефакторинг кода, включая параметр переназначения, переменные и методы переименования, извлечения интерфейса и инкапсуляции членов класса внутри свойств.

Visual Studio имеет особенность фоновой компиляции (также называемой инкрементный сборник). Когда код написан, Visual Studio компилирует его в фоновом режиме, чтобы обеспечить обратную связь о синтаксисе и компиляции ошибок, которые помечены красной волнистой линией. Предупреждения отмечены зеленым подчеркиванием. Фон компиляции не генерирует исполняемый код, так как он требует другой компилятор, в отличии от того, который используется для генерации

исполняемого кода. Фоновая компиляция была впервые введена с Microsoft Visual Basic, но теперь был используется для всех включенных языков. [3]

Кроме всего прочего Visual Studio имеет функцию построения диаграммы классов с учётом написанного кода. Это позволяет разобрать структуру программы и данных.

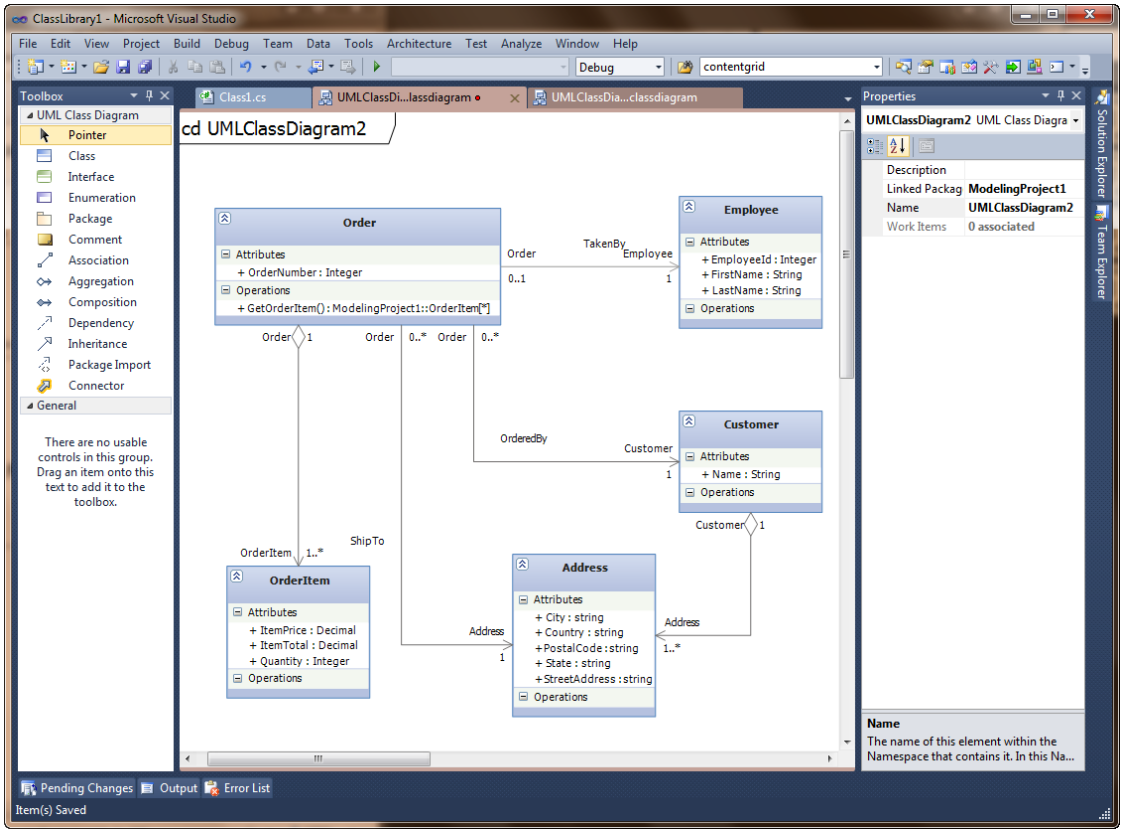

Рисунок 6 – Диаграмма классов в MS Visual Studio

1.5 MIT App Invertor

App Inventor для Android - веб-приложение с открытым исходным кодом, первоначально предоставленное компанией Google, и в настоящее время поддерживается сотрудниками Массачусетского технологического института (MIT). [9]

Это веб-приложение создавать приложения программные приложения для операционной системы Android. Оно использует графический интерфейс, очень похожий на пользовательский интерфейс таких программ как Scratch и the StarLogo TNG, который позволяет пользователям перетащить и установить визуальных объектов, чтобы создать приложение, которое может работать на Android устройствах. При создании App Inventor, компания Google множество своих исследований в образовательной вычислительной технике, а также

работы, проделанные в рамках Google при разработке интернет-сервисов и ресурсов.

App Inventor и проекты, на которых она основана используют конструкционный метод обучения теории, в котором подчеркивается, что программирование может быть средством для привлечения мощных идей посредством активного обучения. MIT App Invertor является частью продолжающегося развития в компьютерах и образовании, которое началось с работы Пейперт и логотипа группы MIT в 1960 - х годах , и также проявлялась в работе Митчела Резника касательно Lego Mindstorms и StarLogo. [9]

Создание приложения состоит из двух функций: создание интерфейса приложения при помощи специального конструктора и программирование событий для приложения при помощи разработки блочной структуры.

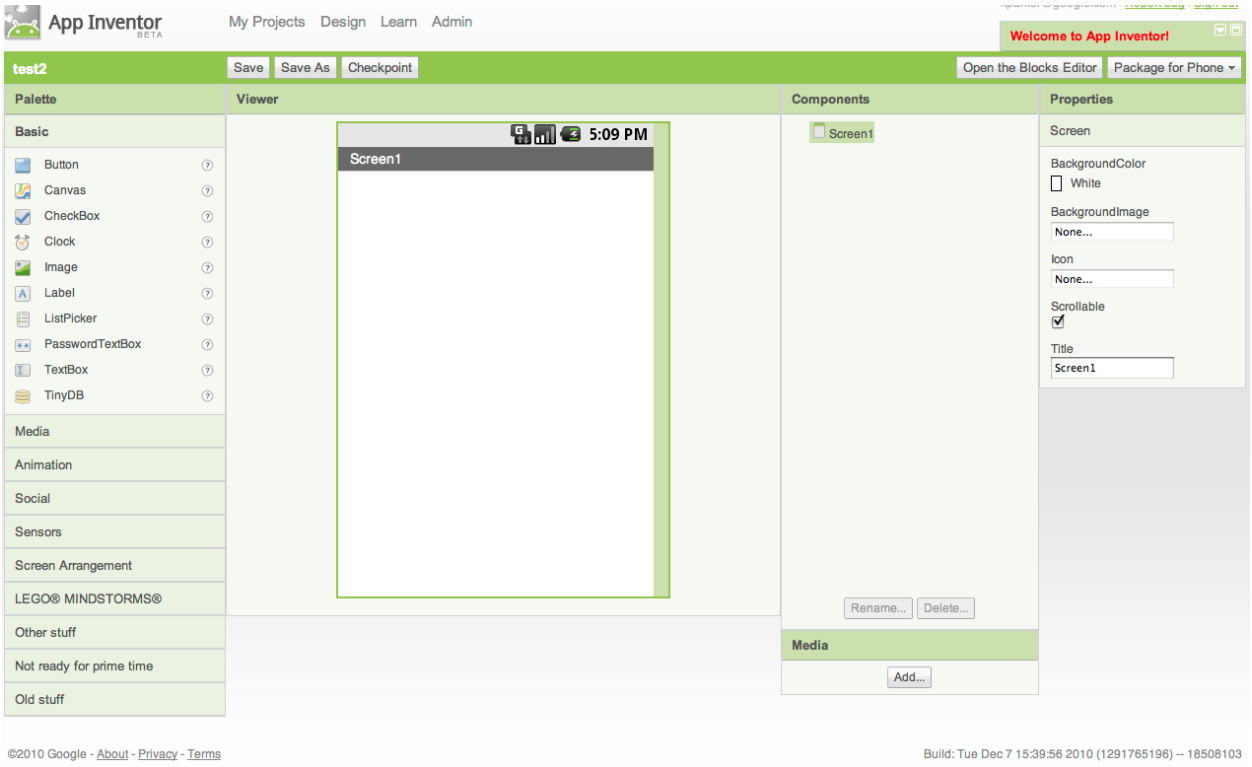

Рисунок 7 – Создание интерфейса для приложения

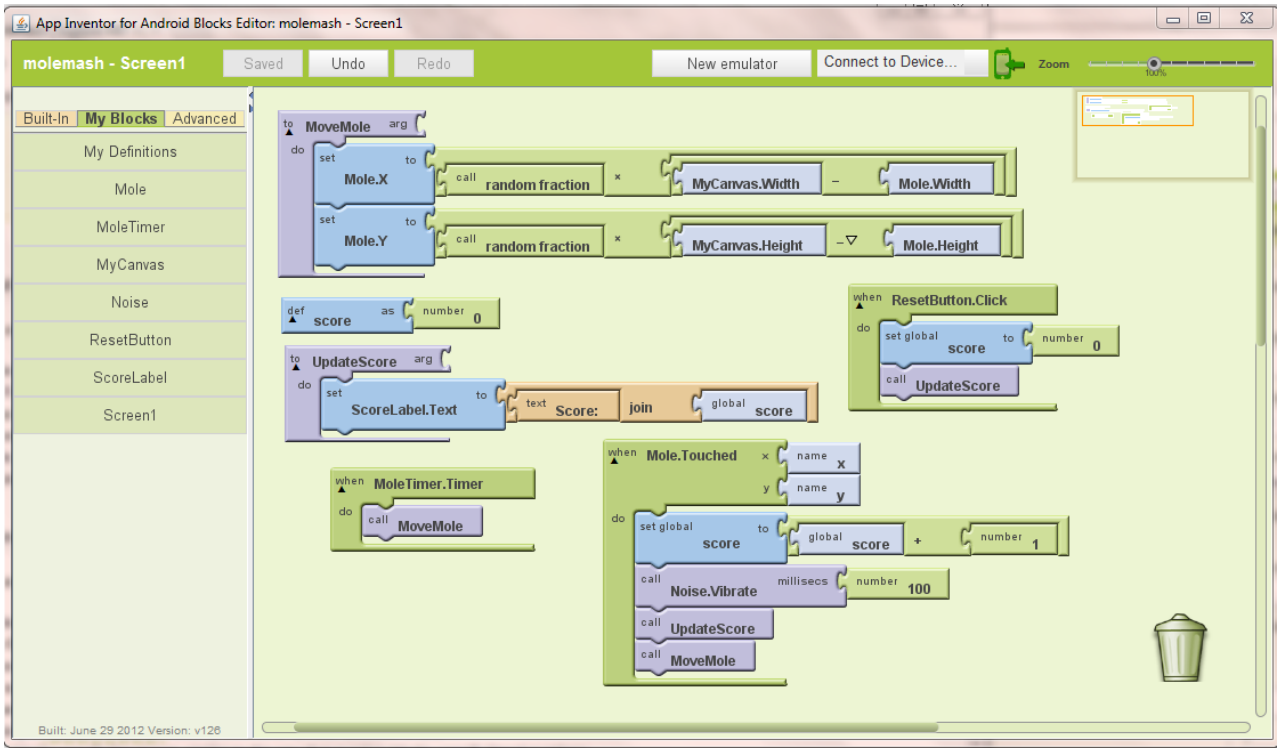

Рисунок 8 – Программирование событий

2 Проектирование системы

Для сборки метеостанции необходимые датчики, которые буду определять метеорологические данные. В данном случае считываются такие данные, как температура, давление воздуха, влажность воздуха, освещённость.

2.1 Структурная схема

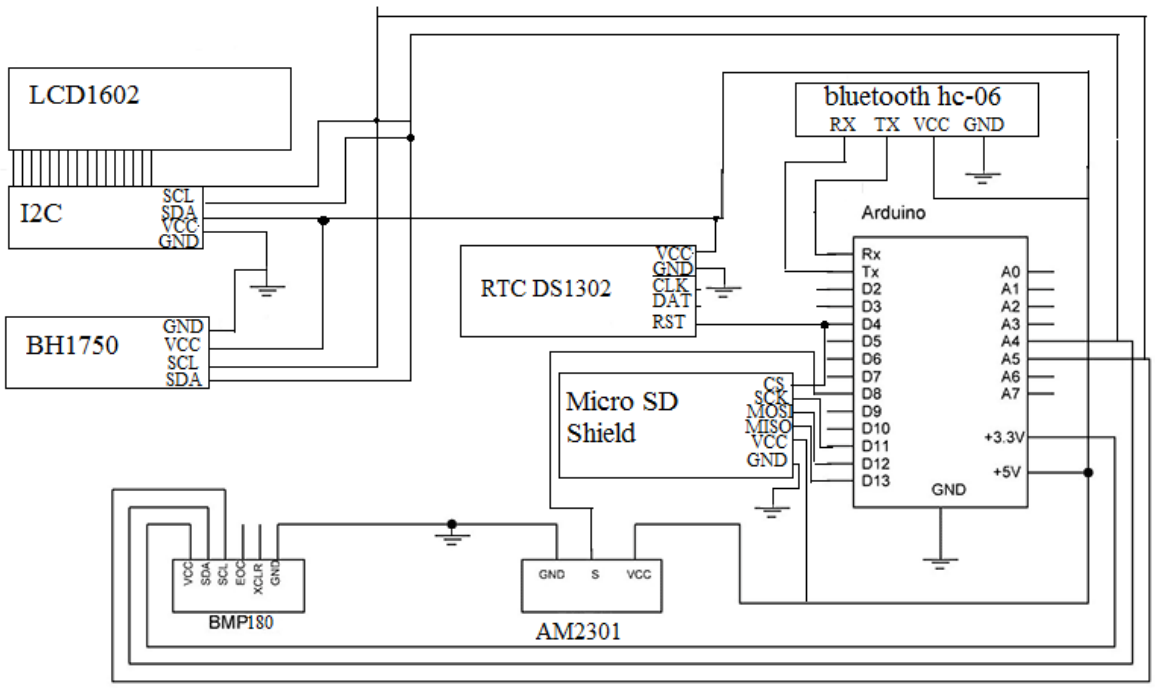

Рисунок 9 – Структурная схема метеостанции

В конечном итоге в состав метеостанции будут входить следующие устройства:

Arduino UNO;
AM2301 – датчик температуры и влажности;
BMP180 – датчик температуры и давления;
BH1750 – датчик освещённости;
LCD1602 – дисплей для отображения данных;
I2C модуль;
Micro SD Shield;
RTC1302 – модуль часов;
HC-06 – Bluetooth модуль.

2.2 AM2301 – датчик температуры и влажности

Датчик температуры и влажности AM2301 является аналогом датчика DHT21, но в отличии от него имеет более прочный корпус.

Рисунок 10 – AM2301 – датчик температуры и влажности

Датчик AM2301 имеет 3 контакта: VCC, GND, OUT. Контакт VCC (красный провод) подключается к питанию. Рабочее напряжение 3,5 - 5,5В. Контакт GND (черный провод) подключается к земле. Контакт OUT (желтый провод) подключается к цифровому контакту digital2. Для считывания данных используется специальная библиотека "DHT.h". После подключения библиотеки необходимо создать и инициализировать переменную типа dht. После инициализации переменной при помощи функций readHumidity (считать данные влажности) и readTemperature (считать значение температуры). Так же в библиотеке DHT.h имеются функции для получения значения температуры в фаренгейтах, а также для получения значения индекса тепла, которые используются зарубежными метеорологами. Диапазон измерения влажности датчика 0-100%. Диапазон измерения температуры -40 – 80 0С, но при слишком высоких или слишком низких температурах использование датчика нежелательно. В таких случаях вероятна погрешность получаемых данных. Погрешность измерения температуры датчика 0,5 0С. Погрешность измерения влажности ± 3%.

2.3 BMP180 - датчик температуры и давления

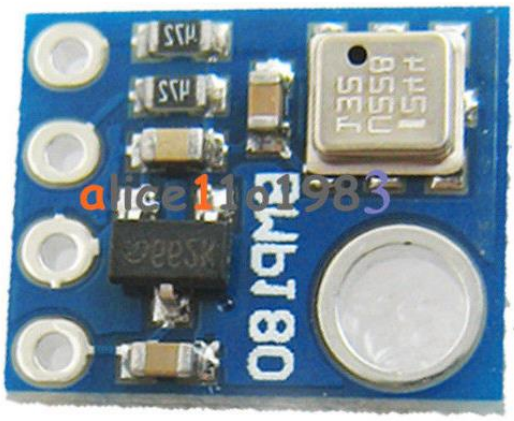

Рисунок 11 – Датчик температуры и давления BMP180

Датчик температуры и давления BMP180 является аналогом датчика BMP085. Датчик построен на модуле GY-65. Рабочее напряжение - 3...5 В. Датчик имеет интерфейс I2C, что весьма упрощает его подключения. В данной структурной схеме будут использоваться 4 контакта датчика BMP180: VCC, GND, SDA, SCL. VCC контакт, подключаемый к источнику питанию +3,3В. GND подключается к земле. Контакт SDA подключаем аналоговому пину 4 на Arduino Uno, который является приёмником SDA сигнала. Контакт SCL подключаем аналоговому пину 5 на Arduino Uno, который является приёмником SCL сигнала. Чтобы считать показания датчика необходимо подключить библиотеку Adafruit_BMP085. После подключения библиотеки можно считать данные с термометра и барометра с помощью функций bmp.readTemperature и bmp.readPressure.

2.4 BH1750 - датчик освещённости

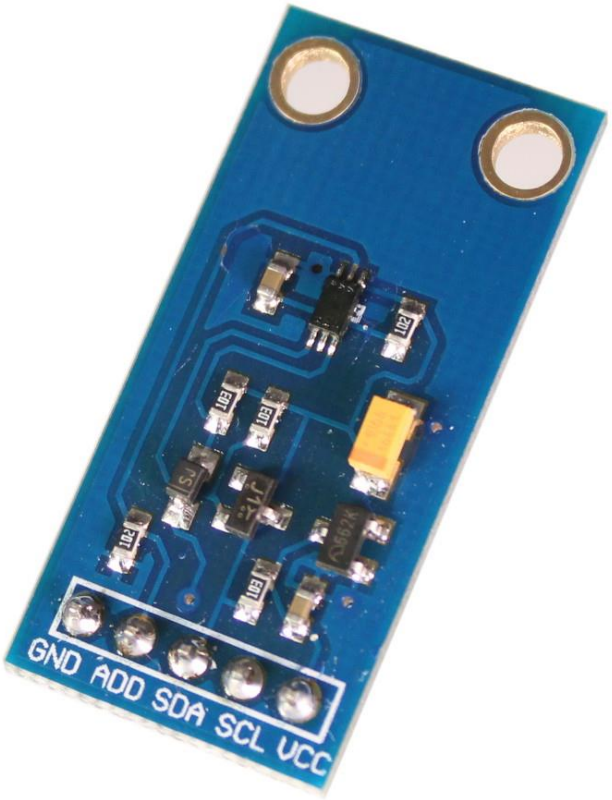

Рисунок 12 – Датчик освещённости BH1750

BH1750 является цифровым датчиком освещённости, основными элементами которого являются цифровой сенсор освещённости и интерфейс I2C шины. Датчик построен на модуле Gy-30. BH1750 4 вывода: VCC, GND, SDA, SCL. VCC контакт, подключаемый к источнику питанию +5В. GND подключается к земле. Контакты SDA и SCL подключаются к аналоговым контактам 4 и 5 соответственно. Для считывания данных с датчика используется библиотека BH1750.h. Считывание значения освещённости происходит с помощью функции readLightLevel в переменную типа uint16_t.

2.5 Модуль часов DS1302

Рисунок 13– Модуль часов DS1302

RTC DS1302 являются модулем часов реального времени. Он используется для записи время получения данных при записи их на SD карту. Данный модуль имеет 5 контактов. Контакт VCC подключается к источнику питания 5В. GND подключается к земле. CLK подключается к цифровому порту 2. DAT подключается к цифровому порту 3. RST подключается к цифровому порту 4. Порты для подключения CLR, DAT, RST можно изменить при желании изменив входные параметры в коде, заливаемом на плату Arduino. Для работы с модулем необходимо подключить библиотеку DS1302.h. Далее необходимо настроить время, дату и день недели с помощью функций rtc.setDOW, rtc.setTime, rtc.setDate. После установки данных необходимо вставить батарейку в модуль иначе данные сбросятся при отключении от Arduino. Рабочий модуль можно подключить к Arduino и считать дату и время с помощью функций rtc.getDate и rtc.getTime.

2.6 Жидкокристаллический дисплей LCD1602

Жидкокристаллический дисплей LCD1602 предназначен для отображения данных с датчиков в момент работы метеостанции. Символы на дисплей можно выводить в 2 строки по 16 символов в каждую строку. Дисплей имеет 16 контактов, которые необходимо подключить в плате Arduino Uno. Для более удобного подключения дисплея к плате используется I2C модуль. I2C является переходным звеном между дисплеем и платой Arduino. Модуль имеет 16 контактов, для подключения к дисплею и 4 контакта для подключения к Arduino (VCC, GND, SDA, SCL). VCC подключается к питанию 5 В. GND подключается к земле. SDA подключается к контакту A4. SCL подключается к контакту A5. Использование I2C модуля значительно упрощает подключение дисплея к плате.

Рисунок 14– Жидкокристаллический дисплей LCD1602

2.7 Модуль micro SD card

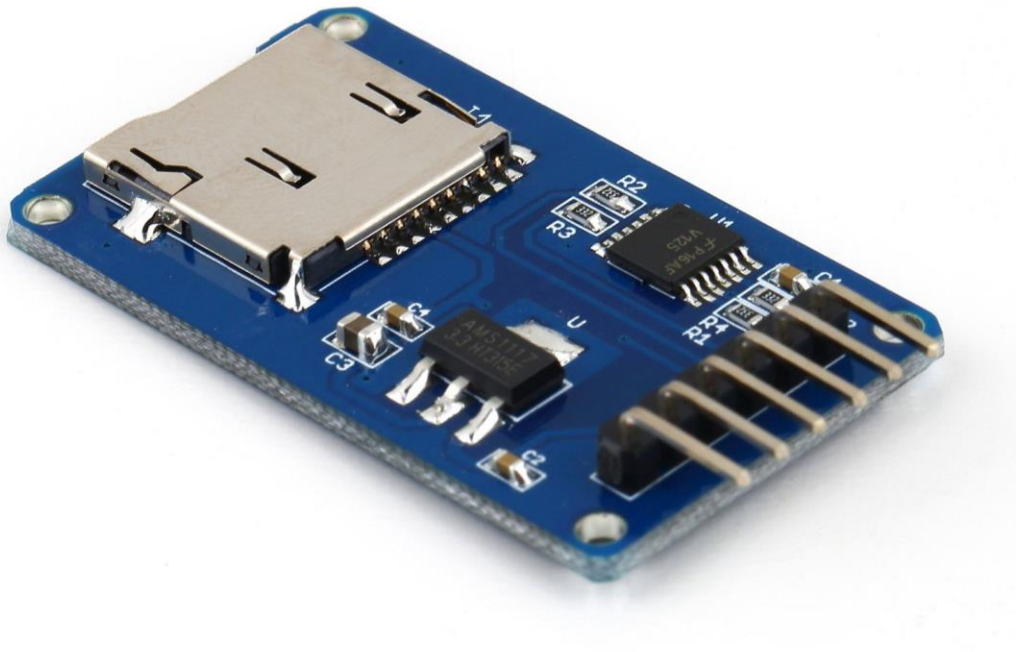

Рисунок 15– модуль micro SD card

Для хранения данных, полученных с датчиков при работе станции в автономном режиме, используется модуль micro SD card. Модуль имеет 6 контактов, которые необходимо подключить к нашей плате. Контакт с маркировкой 5V необходимо подключить к пину 5V на ардуино (пин питания). Контакт GND необходимо подключить к земле. Контакты CLK, DO, DI, CS подключаются к выводам D13, D12, D11, D10 на Ардуино соответственно. В качестве хранилища данных используется micro SD карта с 2 ГБ памяти. Перед началом использования micro SD карты её необходимо правильно отформатировать. Для более корректной записи и считывания данных желательно использовать такие форматы как FAT16 или FAT32.

2.8 Bluetooth-модуль HC-06

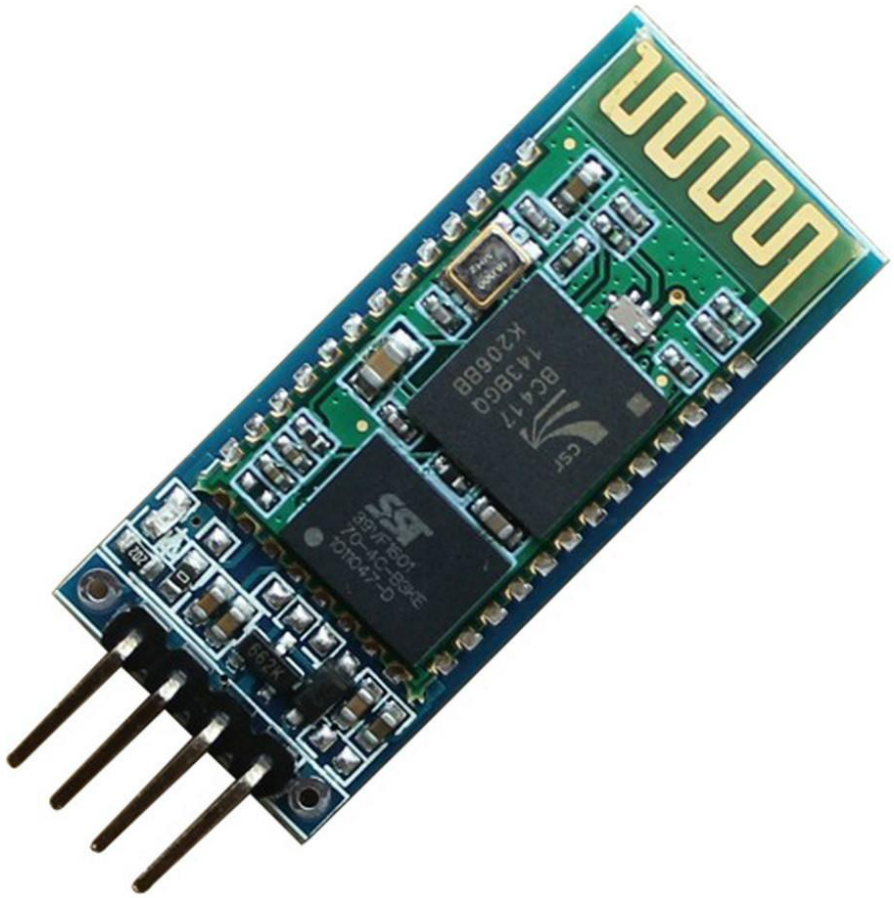

Рисунок 16– Bluetooth-модуль HC-06

Помимо всего прочего, так же имеется мобильное приложения для операционной системы Android, с помощью которого можно подключиться к нашей метеостанции и узнать показания подключённых датчиков. Модуль имеет 4 контакта: RXD, TXD, GND, VCC. VCC подключается к источнику питания 5В, в качестве которого в нашем случае выступает контакт Arduino с маркировкой 5V. Контакт RXD соответственно подключается к пину TX на плате Arduino, а контакт TXD к пину RX. После подключения Bluetooth-модуля к плате работать с ним можно как с сериальным портом. При работе с Bluetooth-модуля используются следующие команды: Serial.available, Serial.read, Serial.write. С помощью команды Serial.available определяется, когда Bluetooth-модуль подключился к мобильному устройству и получает какие-либо сигналы от него. С помощью функции Serial.read считывается команда или информация с мобильного устройства. Далее происходят преобразования данных и с помощью функции Serial.write на мобильное устройство передаются так же сигналы ли другие данные, в нашем случае показания датчиков.

3 Разработка системы

Система будет состоять из следующих компонентов: метеостанция, приложение для операционной системы Windows, приложение для операционной системы Android, база данные на основе MS Sql Server. Полная схема движения данных в системе изображена на рисунке 16.

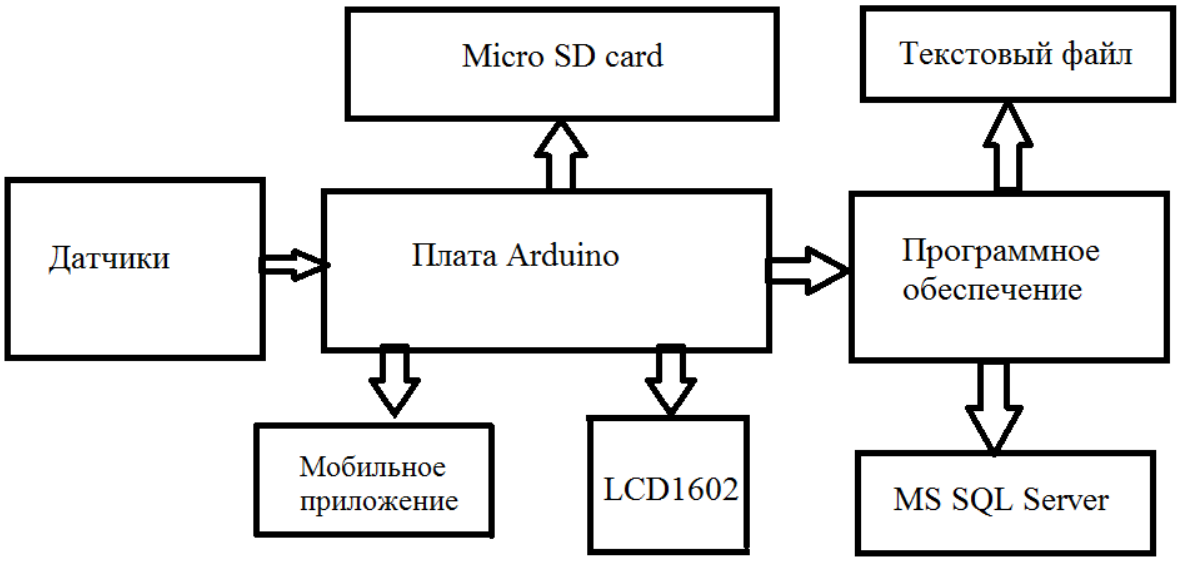

Рисунок 17– Схема движения данных

Данные в системе движутся следующим образом. Плата Arduino Uno считывает с датчиков AM2301, BMP180 и BH1750 значения температуры, давления, влажности и освещённости. Данные обрабатываются микропроцессором платы. С помощью специальных библиотек и функций данные преобразуются в необходимый вид. Обработка данных происходит в цикле. Считывание данных с датчиков и отправка их принимающим устройствам происходит каждые 20 секунд, но при желании этот параметр можно изменить. После обработки данных (например, после преобразования температуры из Цельсиев в Фаренгейты) плата Arduino Uno отправляет их на подключённые устройства. Таким образом, каждые 20 секунд данные записываются на micro SD карту. Для этого считываются текущее время и дата с модуля часов реального времени DS1302, далее эти данные вместе с данными, полученных с датчиков преобразуются в более удобный, для хранения в текстовом файле вид и сохраняются построчно, для удобного считывания, на внешнем накопителе, в качестве которого выступает micro SD карта, подключённая к плате через micro SD Shield. Помимо этого, данные так же отправляются на жидкокристаллический дисплей LED1602 через специальную I2C шину, предназначенную, для более удобного подключения дисплеев к плате Arduino. Данный дисплей помещает в себя 2 строки по 16 символов, в связи с этим приходится выводить данные используя обе строки дисплея. На первую строку выводятся данные с температурных датчиков,

расположенных на датчиках AM2301 и BMP180. На вторую строку выводятся данные с датчиков давления, влажности и освещённости.

В случае если метеостанция подключена к персональному компьютеру и на компьютере запущено программное обеспечение, то на сериальном порту метеостанции появляется информация, что подключено дополнительное устройство. В таком случае плата Arduino подаёт на сериальный порт данные, полученные с датчиков. Программа, получив данные с сериального порта, выводит их в специальные поля, расположенные на главной форме программы. При необходимости данные преобразуются в программном обеспечении. Получив данные, программа также запускает функции для сохранения информации в файл и в базу данных.

Функция для записи данных в базу данных MS Sql первоначально проверяет целостность полученных данных. Если все данные в порядке, то с помощью запроса Insert данные передаются в базу данных. Если какие-либо данные отсутствуют, например если один из датчиков отключён или вышел из строя, то функция автоматически заменяет отсутствующие значения на 0 и так же отправляет данные в базу данных. Если база данных отключена или же отсутствует на данном вычислительном устройстве, то программа при необходимости может вывести сообщение об ошибке.

Функция для записи данных в файл получает данные и построчно записывает их в специальный файл, необходимый для хранения данных и удобного получения их при попытке построить график в интерфейсе программного обеспечения.

При построении графиков данные считываются из файла, сохранённого при помощи программного обеспечения или файла, сохранённого на флэш накопителе. Структура данных в данных файлах идентична. Данные записаны построчно: дата и время, температура с датчика AM2301, температура с датчика BMP180, освещённость с датчика BH1750, влажность с датчика AM2301. При построении графиков, данные считываются тоже построчно и записываются в специальные массивы. После окончания считывания данные сопоставляются и строятся необходимые графики. На графиках можно проследить за изменением любого параметра с течением времени.

4 Разработка ПО для ПК

Программное обеспечение для персонального компьютера разрабатывалось на основе программной платформы .NET Framework 4.5. При разработке использовался программный продукт для разработки приложений Microsoft Visual Studio. В качестве интерфейса программирования приложения используется Windows Forms.

Программное обеспечение выполняет следующие функции:

-считывание данных с метеостанции;

-отображение данных в интерфейсе программы;

-сохранение данных в файл;

-сохранение данных в базу данных;

-построение графиков по каждому из параметров, используя данные, сохранённые на sd карту или же в файл на компьютере;

-отображение данных из базы данных в виде таблицы для большей наглядности и более удобного анализа.

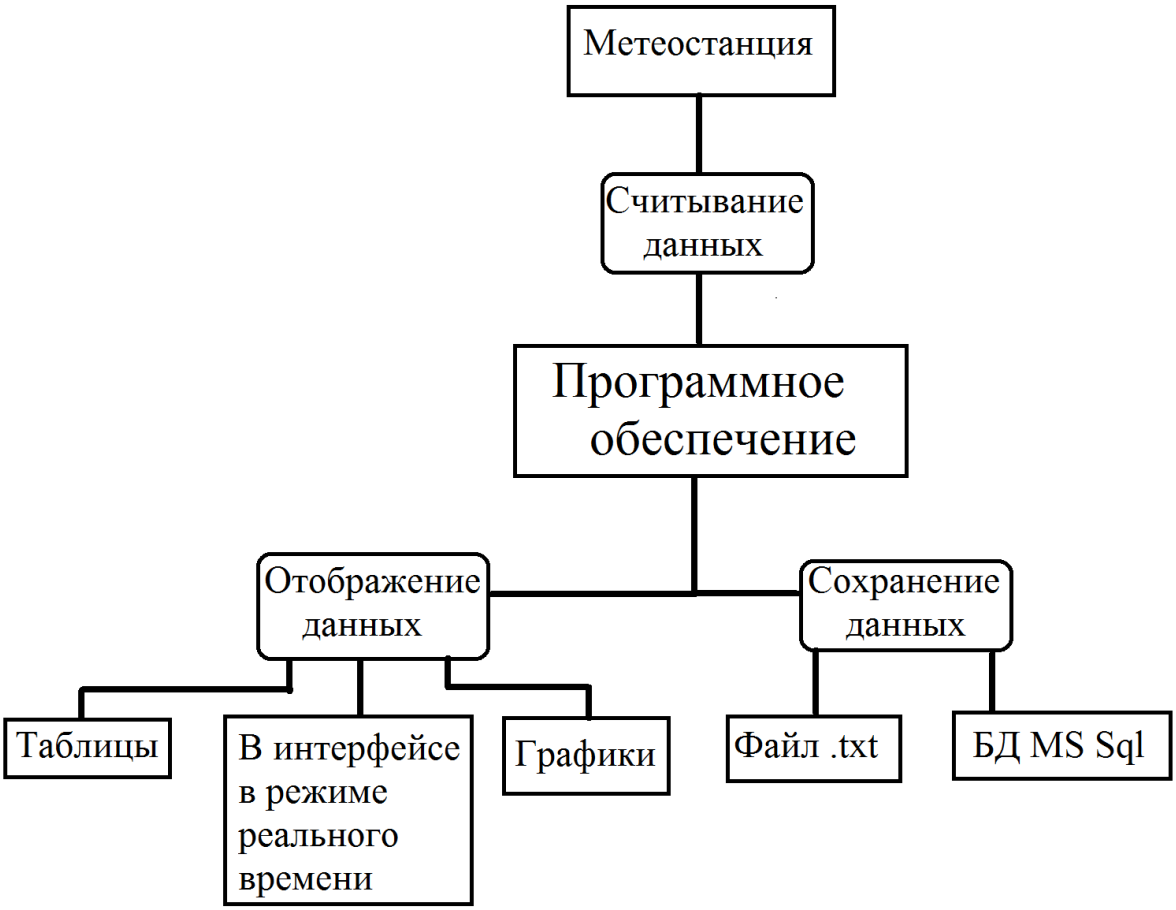

Рисунок 18– Схема основных функций программного обеспечения

Для полноценной работы программного обеспечения необходимо чтобы метеостанция была подключена к персональному компьютеру, на котором запускается программный продукт. Помимо этого, нужно учитывать, что метеостанция будет передавать данные через COM порт. Для обеспечения работоспособности данной функции необходимо установить специальные драйвера, позволяющие операционной системе Windows воспринимать плату Arduino Uno как устройство, которому можно передавать данные и от которого можно их принимать. Данные драйвера устанавливаются автоматически при установке Arduino IDE, или же их можно установить отдельно. После установки возможно понадобится перезагрузить компьютер. Чтобы убедиться в том, что драйвера установлены и работают стабильно можно провести некоторые манипуляции с платой Arduino, использую Arduino IDE для программирования платы. Помимо этого, можно проверить наличие COM порта. Для этого необходимо подключить плату к компьютеру, зайти в Панель управления, далее выбрать пункт Диспетчер устройств. В диспетчере устройств должно появиться новое устройство в разделе «Порты (COM и LPT).

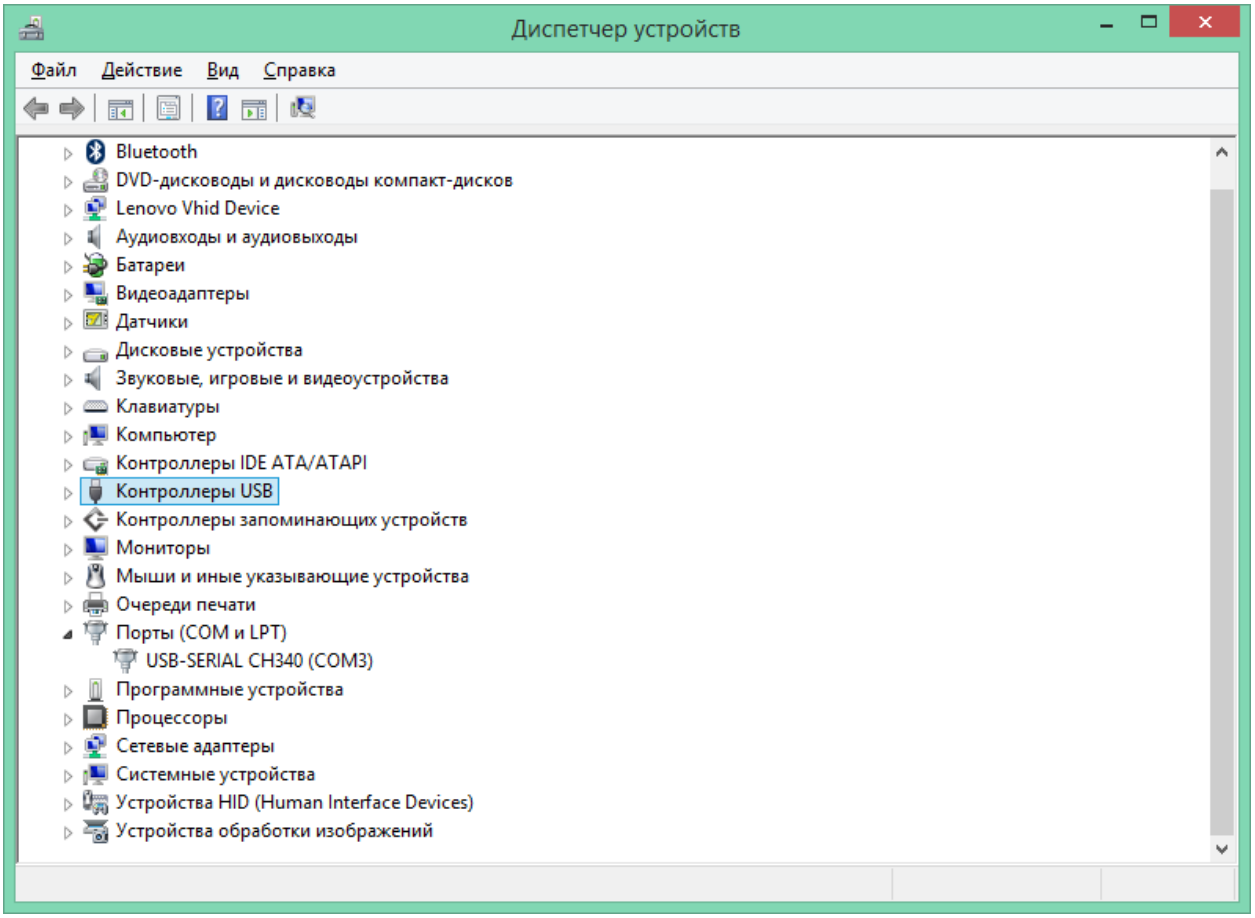

Рисунок 19 – Проверка подключения платы в диспетчере устройств

Убедившись, что плата подключена и всё работает корректно, можно спокойно запускать программу.

Запустив программу, пользователь попадает на основную форму программы. На данной форме отображаются данные со всех датчиков, а также имеются переходы на формы для построения графиков и таблиц.

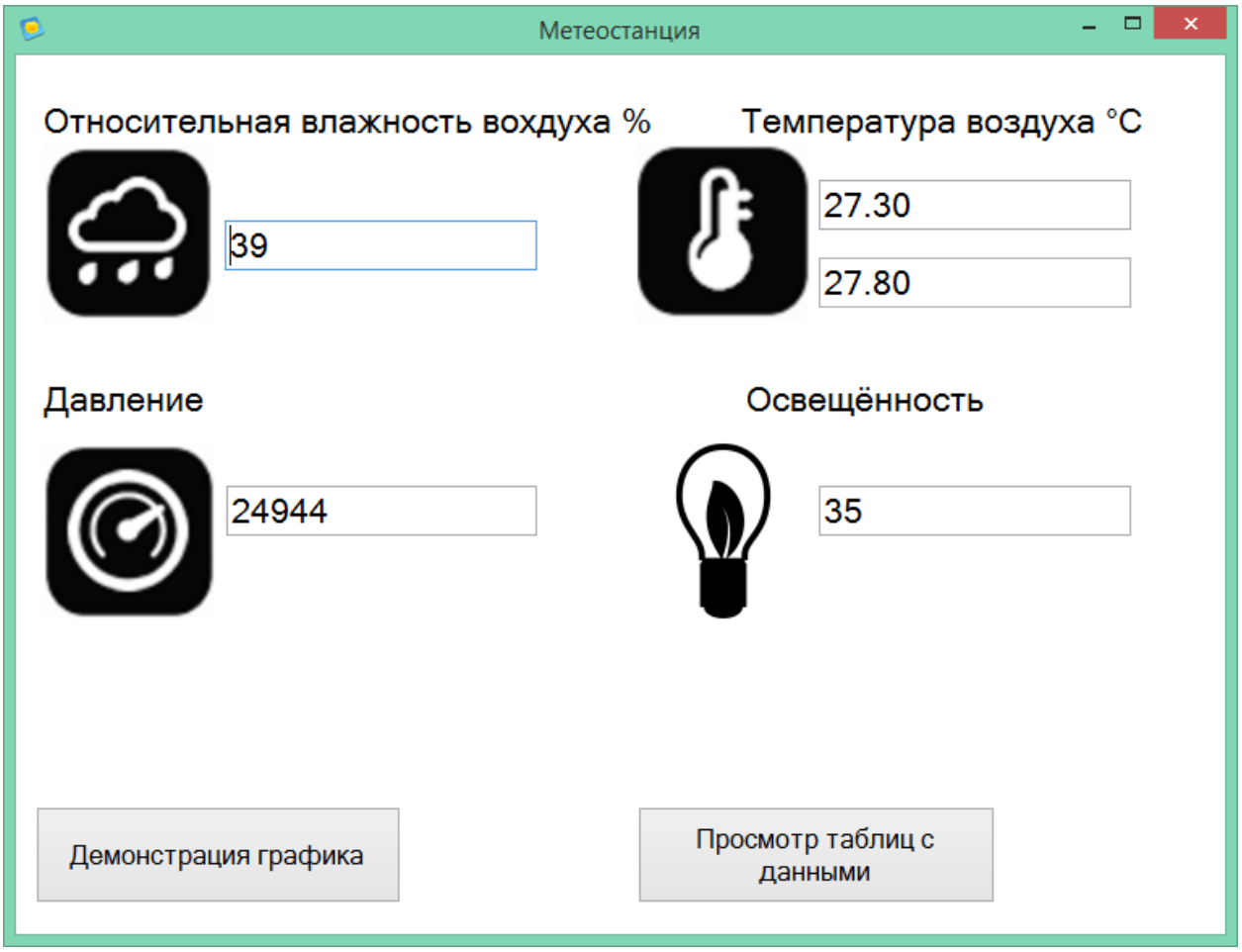

Рисунок 20 – Основная форма программы

Данные в полях обновляются автоматически. Отображение данных на прямую зависит от получения данных с метеостанции. При программировании метеостанции необходимо отправлять полученные данные на сериальный порт с помощью функции Serial.print(…). Данные передаются платой построчно и считываются программой также построчно. Передача данных происходит каждый 20 секунд. Данный промежуток программируется на плате Arduino Uno и при желании его можно изменить, перепрограммировав плату. При получении данных с платы, программа автоматически обновляет данные в полях.

После того, как данные в полях обновились, запускается функция для сохранения данных. Данная функция сохраняет все данные в текстовый файл, после чего данные так же сохраняются в базу данных MS Sql, которая расположена на персональном компьютере. Помимо данных со всех датчиков в файл и в базу данных так же сохраняется время проведения замера или же в данном случае время получения данных с метеостанции. В случае если какой-либо из датчиков отключён или неисправен, то плата Arduino возвращает

вместо значения параметра ноль. То есть на главной форме будет отображаться значение «0» и соответственно в файл и в базу данных так же будет записано значение «0».

На главной форма помимо прочего имеются также две кнопки: «Демонстрация графиков» и «Просмотр таблиц с данными». Данные кнопки позволяют перейти на соответствующие формы.

При нажатии на кнопку «Демонстрация графиков» откроется форма «График». Эта форма предназначена для построения графиков по имеющимся данным. Для построения графиков можно использовать файл под названием «Все данные», сохраняемый программой во время работы. Данный файл имеет расширение .txt и обычно сохраняется в папку с исходным кодом программы или же в папку, в которой расположен .exe файл программы. Так же для построения графиков можно использовать файл, сохранённый на SD карте с помощью шилда самой платой Arduino. Это осуществимо благодаря тому, что оба файла имеют одинаковую структуру.

Для выбора источника данных необходимо нажать на форме на кнопку «Выбрать файл». После этого откроется диалоговое окно с проводником. В проводнике необходимо открыть папку, содержащую файл и выбрать его. После выбора файл в левом верхнем углу формы появится раскрывающийся список, в котором необходимо выбрать нужный нам параметр. После выбора необходимого параметра построится график зависимости данного параметра от времени.

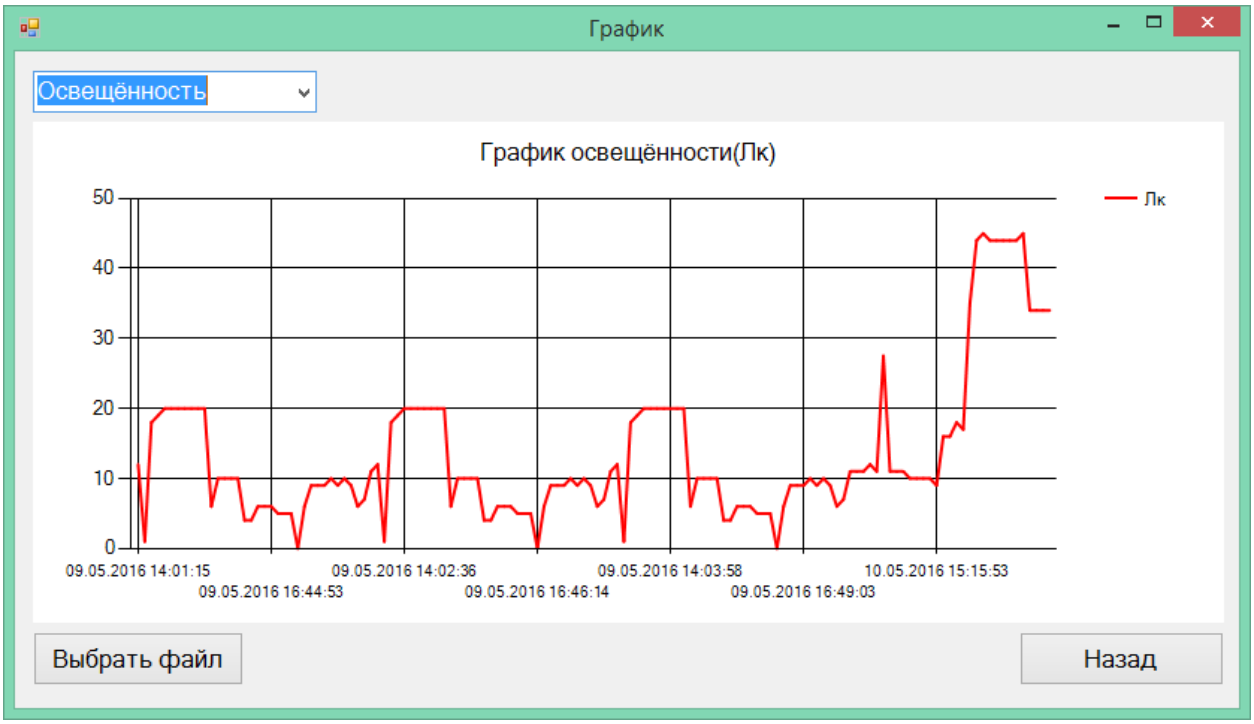

Рисунок 21 – Форма для построения трафиков

Для построения графика используется специальный элемент интерфейса Windows Form – chart. Данный элемент отображает данные в Windows Forms в

виде гистограммы или диаграммы. Для построения графика используются два массива. Каждому элементу одного массива соответствует ровно один элемент из другого массива. Нажав на данной форме на кнопку назад можно попасть на главную форму.

Помимо этого, есть ещё форма для построения таблиц. Для открытия данной формы на основной форме программы необходимо нажать на кнопку «Просмотр таблиц с данными».

На форме «таблица» отображаются данные, хранящиеся в базе данных MS Sql. Каждой записи соответствует определённая дата и время. При сохранении данных в базе данных программа использует системное время. При отображении данных в виде таблицы можно получить сразу все данные, сохранённые в базе данных или же можно выбрать определённую дату и получить данные, записанные в данный день. Для этого необходимо в окне календаря, расположенном в нижней части формы, выбрать необходимую дату и нажать на кнопку «Определённая дата».

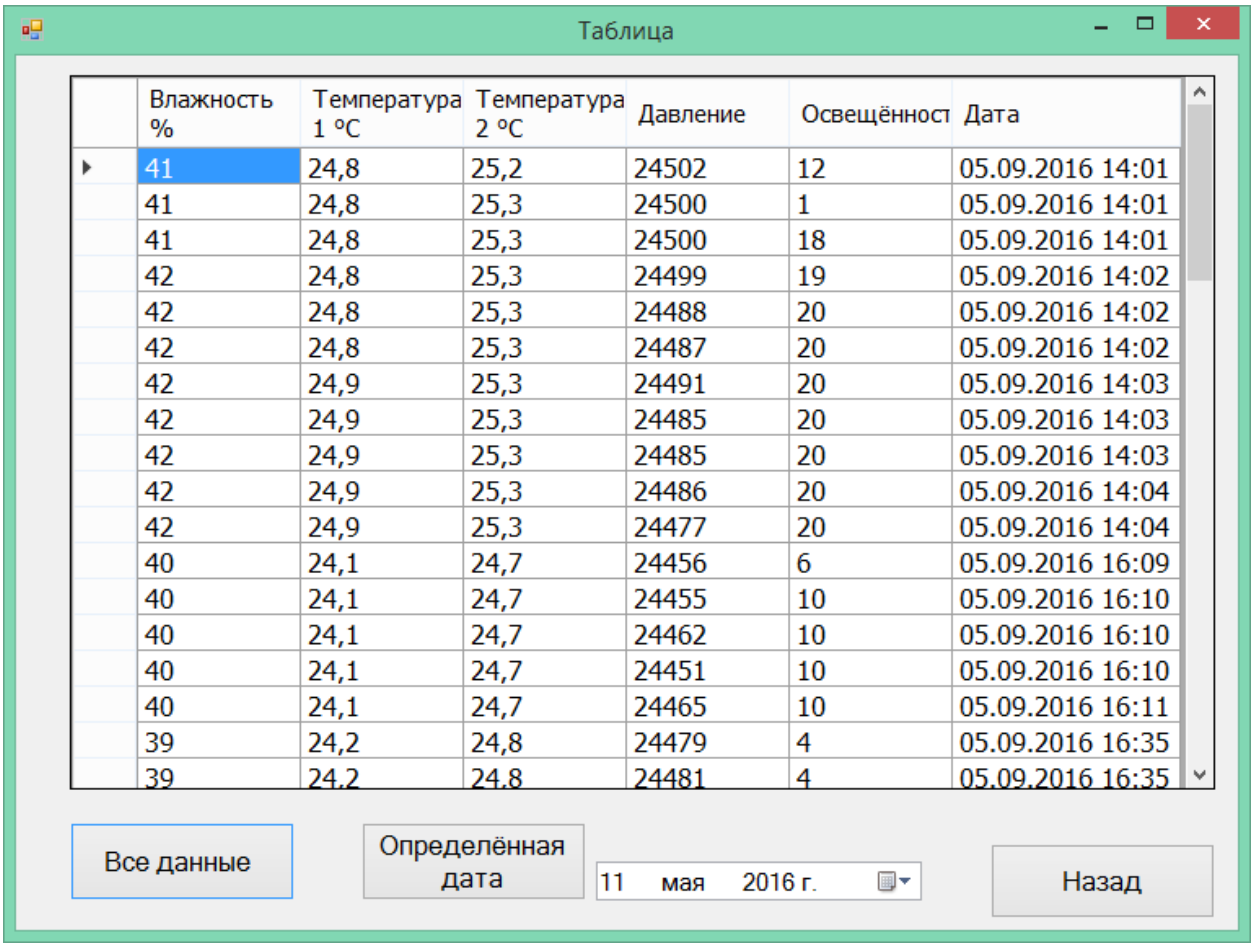

Рисунок 22 – Форма для построения таблиц

5 Сборка метеостанции

При сборке метеостанции использовался беспаячный метод соединения деталей. Подобный метод сборки осуществляется с помощью макетной платы. Благодаря макетной плате к одному входу платформы Arduino можно подсоединить несколько различных устройств. Так же благодаря данному методу сборки можно без проблем заменить при необходимости какой-либо датчик или же подключить новые датчики.

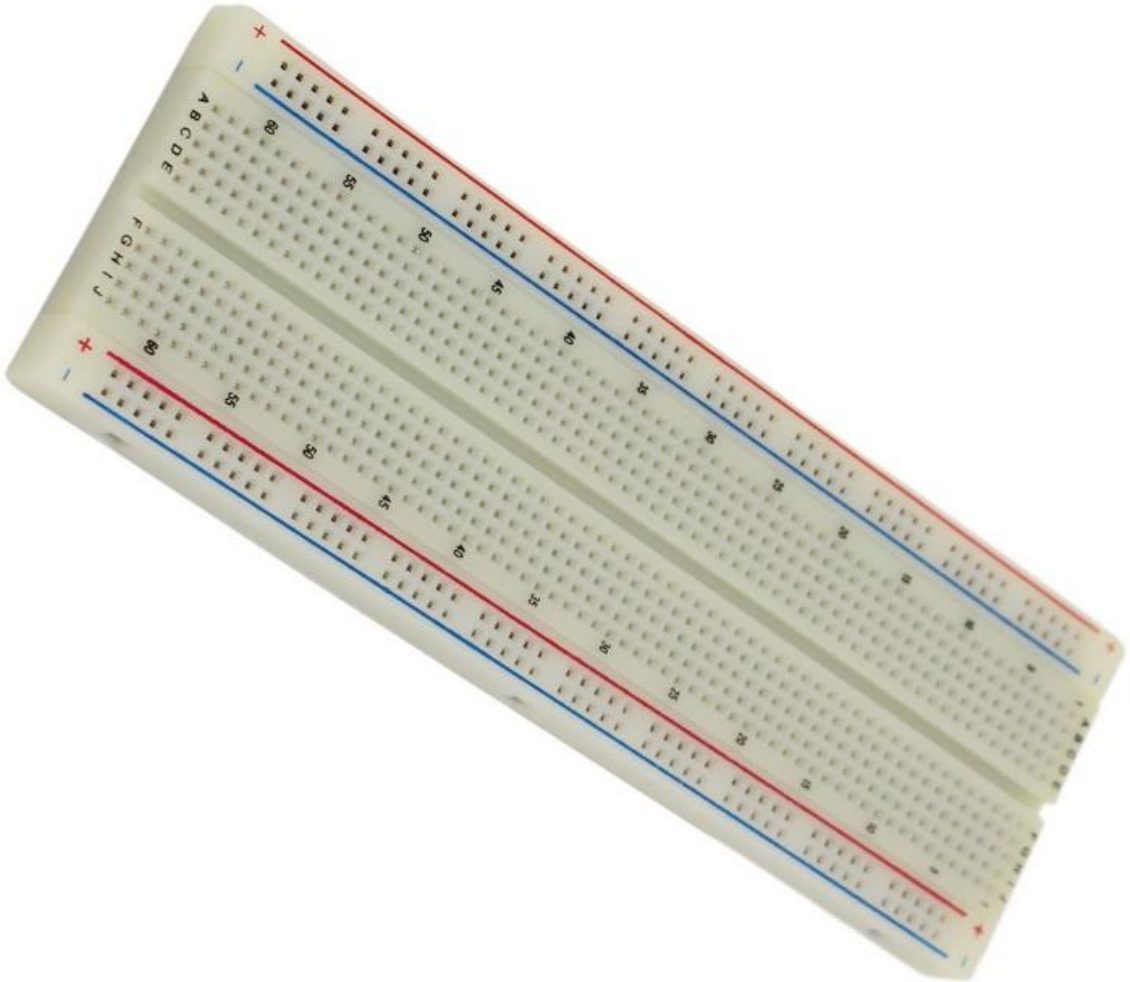

Рисунок 23 – Breadboard

Боковые рельсы соединены вдоль платы (красные и синие), а остальные- поперёк. Также в поперечных соединениях посередине платы можно увидеть разрыв. Такая конструкция необходима для того, чтобы можно было разместить более сложные устройства на плате, например, микросхему с двумя рядами ног. Боковые рельсы обычно служат для подачи питания и зачастую так и помечены, как «+» и «-».

Все приборы, из которых состоит прибор питаются двумя источниками. Первый источник подаёт напряжение 5В, а второй 3.3В. Питаются все

датчики и приборы через макетную плату. Благодаря тому, что на макете имеются две дорожки для распределения питания, с помощью неё можно записать все приборы. Так же имеется несколько приборов, которые подключаются к одним и тем же контактам. К этим приборам можно отнести датчики влажности, давления и температуры. Питание распределяется весьма равномерно. Дисплей LED1602 потребляет более всего электроэнергии. При обычном подключении дисплей занимает 16 портов на плате, при этом подключать его необходимо через специальный резистор, который не позволит спалить контакты, расположенные на дисплее. В данном случае для подключения дисплея используется всего лишь 4 контакта из них 2 необходимы для подключения к питанию и к земле. Благодаря I2C модулю можно регулировать подцветку экрана, например, если метеостанция будет использоваться в тёмном помещении или при очень ярком солнечном свете.

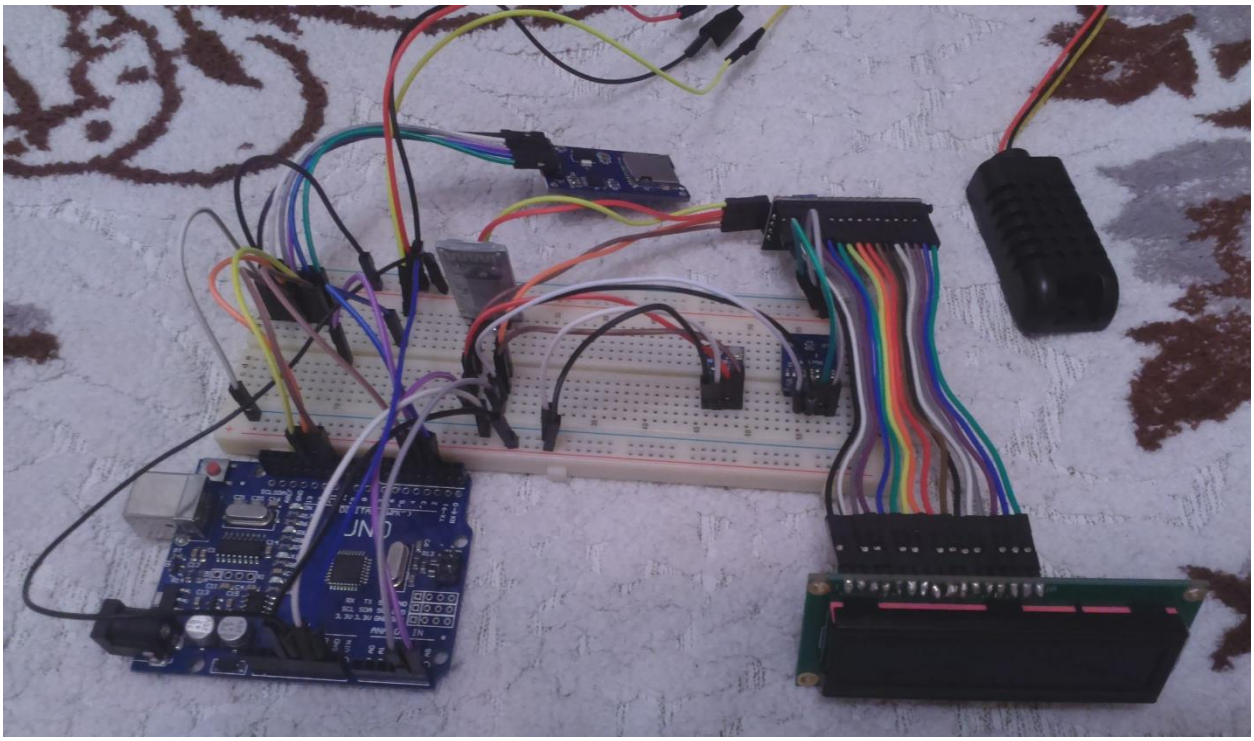

Рисунок 24 – Метеостанция без корпуса

При сборке метеостанции был использован пластиковый корпус. В корпусе были сделаны отверстия для контактов питания для платы Arduino и для подключения USB провода. Так же в корпусе сделано отверстие для дисплея (экрана, на котором будут отображаться данные с датчиков). Также в устройстве имеется датчик освещённости. Данный датчик нельзя располагать в закрытой коробке или корпусе. Для этого в корпусе были проделаны отверстия для вывода датчика за его пределы.

При необходимости крышку корпуса можно легко снять. В таком виде метеостанция может использоваться только в солнечную погоду, когда вероятность осадков очень мала. У данного использования метеостанции есть несколько плюсов. Например, при отсутствии крышки будет обеспечена

хорошая вентиляция. Это позволить воздуху в метеостанции соответствовать воздуху, расположенному за её пределами. Так же на дне метеостанции имеются небольшие отверстия. Первоначально они предназначались для установки ножек, но необходимости в установке ножек нет. Поэтому отверстия свободные и позволяют воздуху свободно циркулировать в метеостанции. Это позволяет датчикам получать более точные параметры воздуха на улице. Но несмотря на это, лучше использовать метеостанцию с закрытой крышкой. В этом случае устройство будет защищено от дождя и пыли, которая будет быстро накапливаться на приборах при открытой крышке. Так же метеостанцию можно использовать в помещении. В подобном случае можно подключить к ней датчик CO_2 – углекислого газа, который будет демонстрировать насыщенность воздуха чистым кислородом. Благодаря данному датчику можно узнать, когда необходимо проветрить помещение. Но в данный момент планируется использовать метеостанцию для получения метео-данных за пределами помещения, поэтому данный датчик не устанавливался.

Рисунок 25 – Метеостанция с корпусом

6 Разработка мобильного приложения

В данной разработке помимо всего прочего разрабатывалось мобильное приложение. Мобильное приложение разработано для операционной системы Android. Приложение разрабатывалось при помощи веб-приложение MIT App Invertor, предназначенным для разработки приложений под операционную систему Android.

Разработка приложения состоит из двух этапов: разработка интерфейса программы и программирование событий. Для разработки приложения первоначально нужно определить какие функции данное приложение будет выполнять. В данной работе мобильное приложение необходимо для прямого мониторинга данных с метеостанции. Другими словами, пользователь сможет узнать показания датчиков метеостанции, не подходя к ней. Данное приложение можно использовать в тех случаях, когда дисплей метеостанции недоступен, когда метеостанция не подключена к компьютеру или же на рабочей станции не установлено программное обеспечение для визуализации данных, когда метеостанция вне зоны доступа, то есть если метеостанция расположена на улице, а пользователь находится в помещении. Так же приложение будет использовать, когда пользователю будет элементарно лень подойти к метеостанции и узнать данные. Другими словами, приложение обеспечит удалённый доступ к получению данных с метеостанции.

6.1 Разработка интерфейса

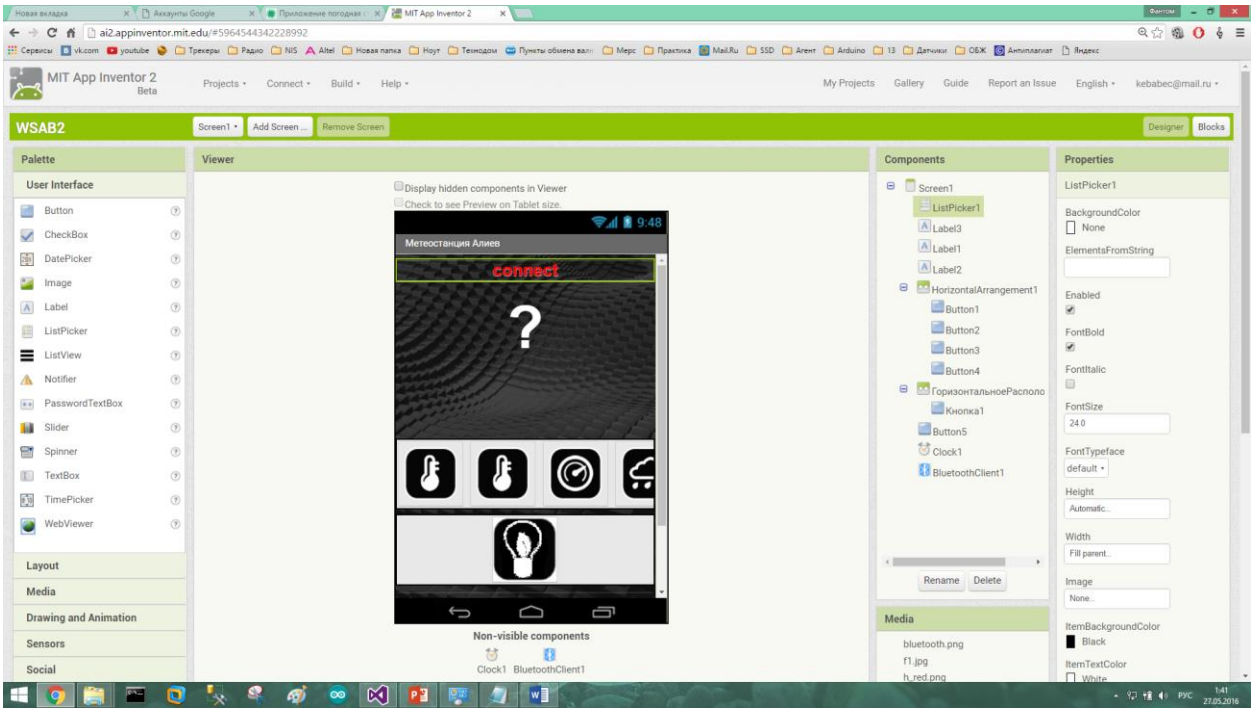

Рисунок 26 – Интерфейс приложения при разработке интерфейса.

При разработке интерфейса необходимо понять какие элементы интерфейса необходимы и какие функции они будут выполнять.

В нашем случае метеостанция необходима только для прямого отображения полученных данных. Другими словами, при получении данных они не будут преобразовываться или как-либо изменяться.

Первоначально при разработке интерфейса необходимо подобрать картинки, которые будут отображаться в качестве элементов интерфейса. Первоначально необходимо подобрать картинку, которая будет отображаться в списке приложений в качестве иконки нашего приложения. Для этого необходима картинка в формате PNG32 размером 128 на 128 пикселей. У нас основным устройством для получения данных является Bluetooth модуль. Поэтому в качестве иконки используется значок данного модуля.

Рисунок 27 – Иконка приложения

Далее необходимы несколько цветных картинок для выбора конкретного параметра. В приложении будут использоваться картинки 2 типов. Белого цвета, в случае если параметр не выбран и красного, если параметр выбран.

Рисунок 28 – Изображения кнопок для определённых параметров

Помимо данных картинок так же имеются картинка для главного фона приложения.

Так же необходимы элементы типа Label для отображения текста. Основным таким элементом является элемент Label1. В данном элементе интерфейса будут отображаться значения параметров при считывании данных. При запуске программы в данном окне будет отображаться знак вопроса, информирующий о том, что данные ещё не получены и необходимо подключиться к источнику информации. Так же данный знак будет

отображаться в тех случаях, когда по каким-то причинам не получилось считать значения определённых параметров с прибора для анализа метеоданных. В верхней части приложения расположено поле с его названием. Также в интерфейсе приложения используется стиль, который позволяет отображать параметры нашего устройства в верхней части дисплея. Например, в данной части дисплея отображаются заряд батареи, время, уровень сигнала и другое. Другими словами, приложение занимает не абсолютно всё место на дисплее.

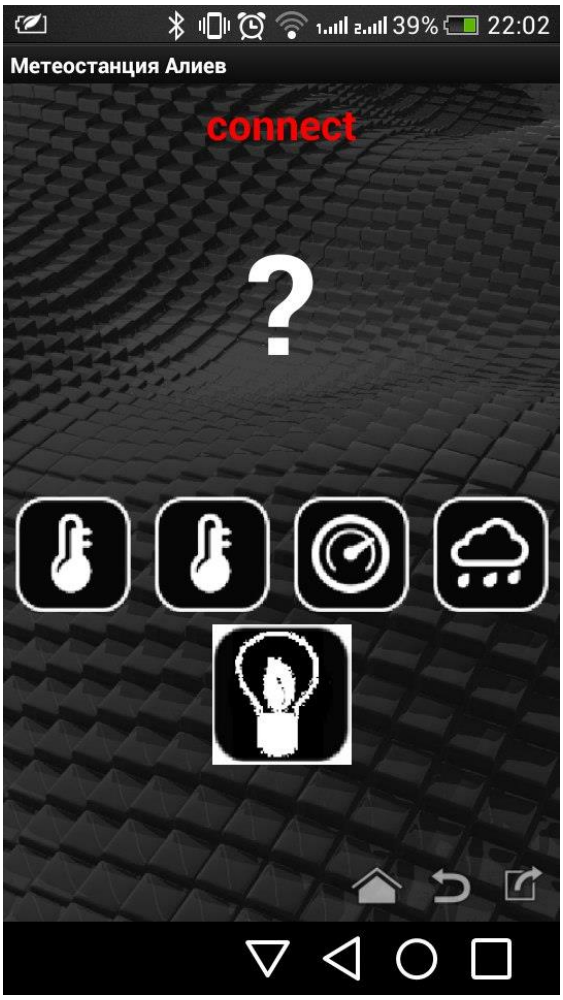

Рисунок 29 – Интерфейс приложения при запуске программы

В верхней части интерфейса расположен элемент типа ListPicker. Данный элемент необходим для подключения к определённому устройство при помощи Bluetooth модуля. Для подключения первоначально необходимо зайти в настройки мобильного устройства и включить Bluetooth, после чего произвести поиск устройств. При запуске приложения необходимо нажать на кнопку “Connect”. После этого откроется окошко для выбора устройства для подключения. В нашей метеостанции используется Bluetooth модуль HC-06. При выборе устройств необходимо выбрать устройство с названием HC-06. При выборе устройства рядом с наименованием пишется мак адрес устройства. В случае если поблизости будет работать ещё один модуль HC-06,

можно будет идентифицировать наше устройство именно по данному адресу. Но для этого первоначально необходимо запомнить его. Для удобства пользователей в нижней части дисплея располагаются системные кнопки операционной системы Android, такие как кнопки «Назад», «Свернуть всё», «Домой», «Развернуть панель настроек».

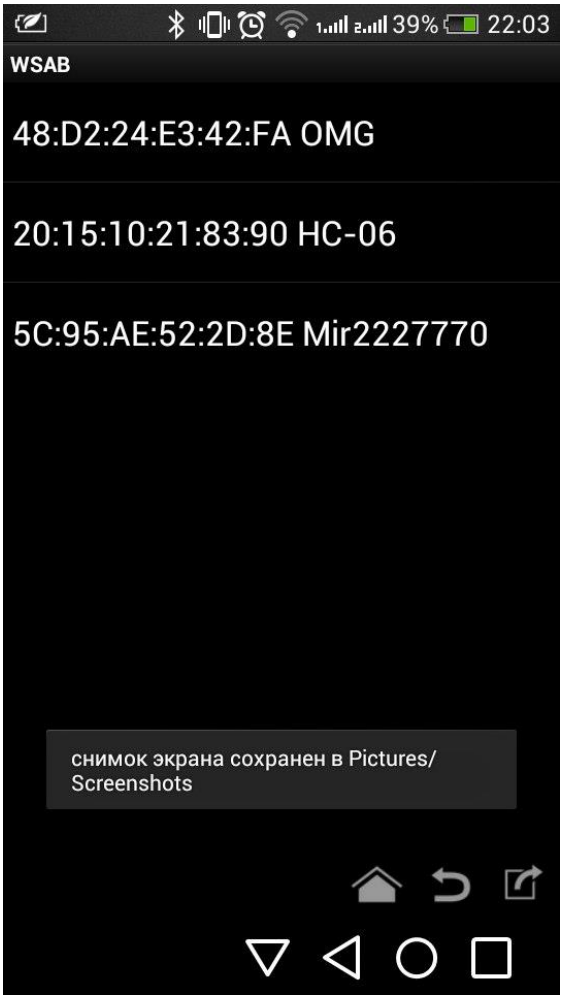

Рисунок 30 – Выбор устройства для подключения

После выбора устройства мобильное устройство попробует подключиться к источнику данных для приложения. В случае если подключение не состоялось, то приложение выведет ошибку «Нет соединения. Устройство включено?». В таком случае необходимо проверить состояние метеостанции. Если метеостанция включена необходимо проверить состояние Bluetooth модуля. Если модуль работает и не подключён к смартфону, то на нём должен мигать зелёный диод. Если же к метеостанции уже подключён другой смартфон и по Bluetooth модулю передаётся сигнал, то диод будет просто гореть зелёным цветом. Если же подключение прошло удачно, то в верхней части дисплея будет отображаться мак адрес Bluetooth модуля, расположенного на метеостанции. После этого необходимо выбрать параметр, значение которого необходимо узнать. После нажатия на кнопку она сменит свой цвет на красный. И на дисплее вместо вопросительного знака

будет отображаться цифра, соответствующая значению параметра, выбранного пользователем, которая будет получена с метеостанции.

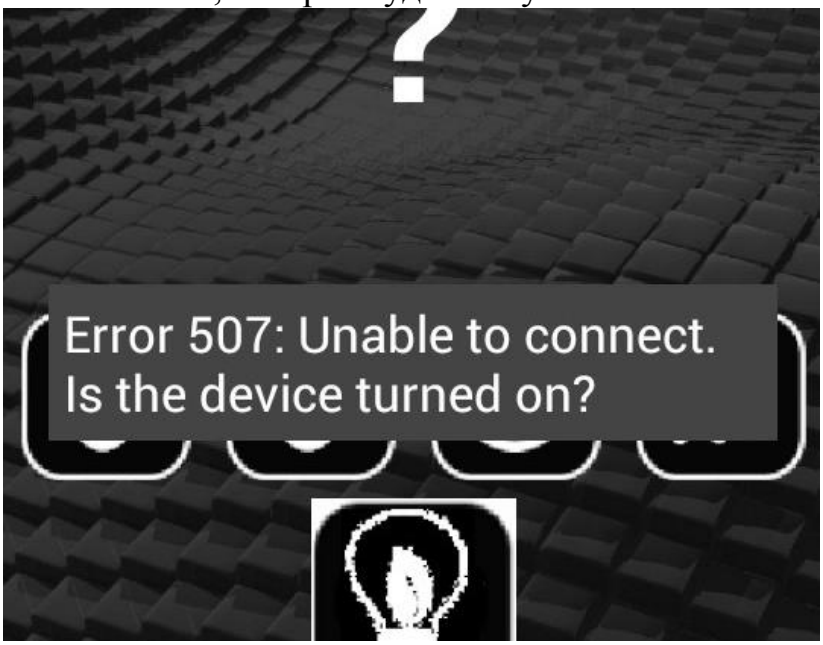

Рисунок 31– Сообщение об ошибке при подключении

Рисунок 32 – Интерфейс при удачном считывании данных

На этом разработка интерфейса мобильного приложения заканчивается и не обходимо приступить к программированию событий. Для этого в веб

приложении MIT App Invertor необходимо перейти в режим блоков. Для этого необходимо в верхней правой части приложения нажать на кнопку Blocks.

6.2 Программирование событий

Для начала необходимо запрограммировать момент нажатия на кнопку «Connect» и выбора устройства для подключения. Для этого необходимо запрограммировать событие нажатия на элемент ListPicker.

Рисунок 33 – Обработка настройки листпикера перед нажатием

В данном случае перед тем как произошло нажатие на элемент лиспикер в список его элементов добавляются все адреса и имена из списка блютуз модуля, установленного на смартфоне. Обычно подобные устройства добавляются в список автоматически при подключении к ним. Поэтому перед тем как подключаться к метеостанции необходимо проделать следующее. Зайти в настройки смартфона, выбрать настройки Bluetooth, включить устройство Bluetooth расположенное на смартфоне нажатием на кнопку On, после этого необходимо включить метеостанцию. На Bluetooth модуле метеостанции должен заморгать зелёный светодиод. Это означает, что модуль готов к передаче данных. После этого необходимо на смартфоне произвести поиск устройств. При поиске должно быть найдено устройство с именем, соответствующим модели Bluetooth модуля, установленного на метеостанции. Обычно это может быть HC-05 или HC-06. В данной метеостанции установлен модуль HC-06. Соответственно на смартфоне необходимо попробовать подключиться к данному устройству. Подключение возможно не произойдёт, но устройство сохранится в списке имён устройств к которым смартфон подключался и появится в списке, который вернёт функция BluetoothClient1-AddressesAndNames.

Далее необходимо запрограммировать процесс подключения к устройству. Для этого для элемента ListPicker1 необходимо обработать событие AfterPicking. В этом случае событии проверяется действительно ли смартфон смог подключиться к устройству, выбранному из списка. Если подключение произошло удачно, то происходим следующая цепочка событий. Цвет текста, отображаемого на элементе ListPicker1 меняется на синий и сам текст меняется на название выбранного элемента. Устройство автоматически отправляет на метеостанцию сигнал для считывания данных со второго температурного датчика. Для этого на метеостанцию отправляется текст «2». Сигнал отправляется на устройство BluetoothClient1, которым является Bluetooth модуль метеостанции, с помощью команды SendText. После этого

запускается модуль часов. Который позволяет обновлять данные каждые несколько секунд. После запуска часов меняются все картинки. Выбранный элемент получает кнопку красного цвета. Все остальные кнопки остаются белыми. Помимо этого, в специальном элементе, отображающем название считанного параметра устанавливается текст с названием параметра.

Если же попытка подключения оказалась неудачной, то в элементе ListPicker выводится текст Reconnect красного цвета. Также на дисплей выводится сообщение об ошибке.

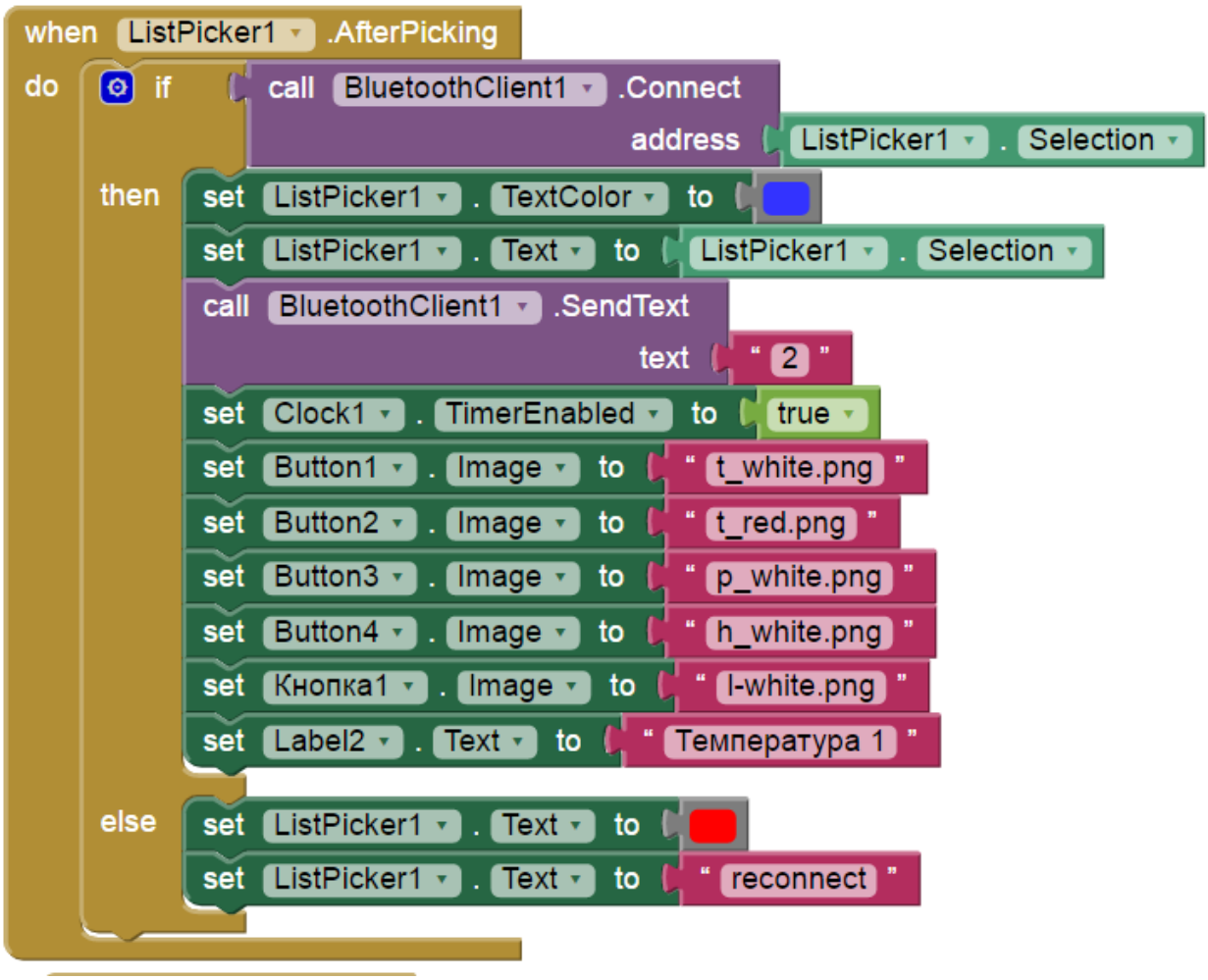

Рисунок 34 – Обработка настройки листпикера после нажатия

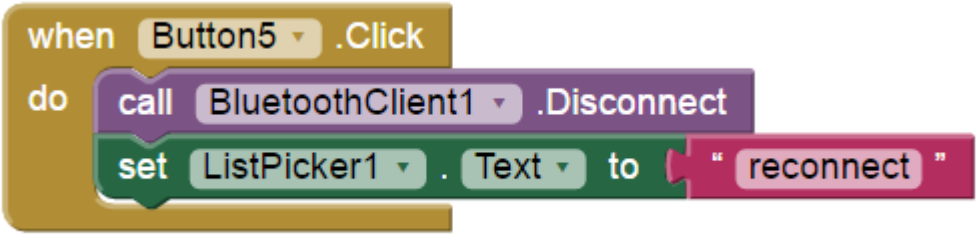

Рисунок 35 – Обработка кнопки для разрыва соединения

Для отключения смартфона от метеостанции существует специальная кнопка, которая вызывается событие Disconnect на элементе BluetoothClient. Данная кнопка необходима на тот случай, если пользователь желает отключиться от метеостанции или же подключиться к другому источнику информации.

Далее необходимо запрограммировать часы и событие нажатия на кнопки выбора параметра. На дисплее будут иметься пять таких кнопок: кнопка температуры с датчика AM2301, кнопка температуры с датчика BMP180, кнопка давления, кнопка влажности и кнопка освещённости.

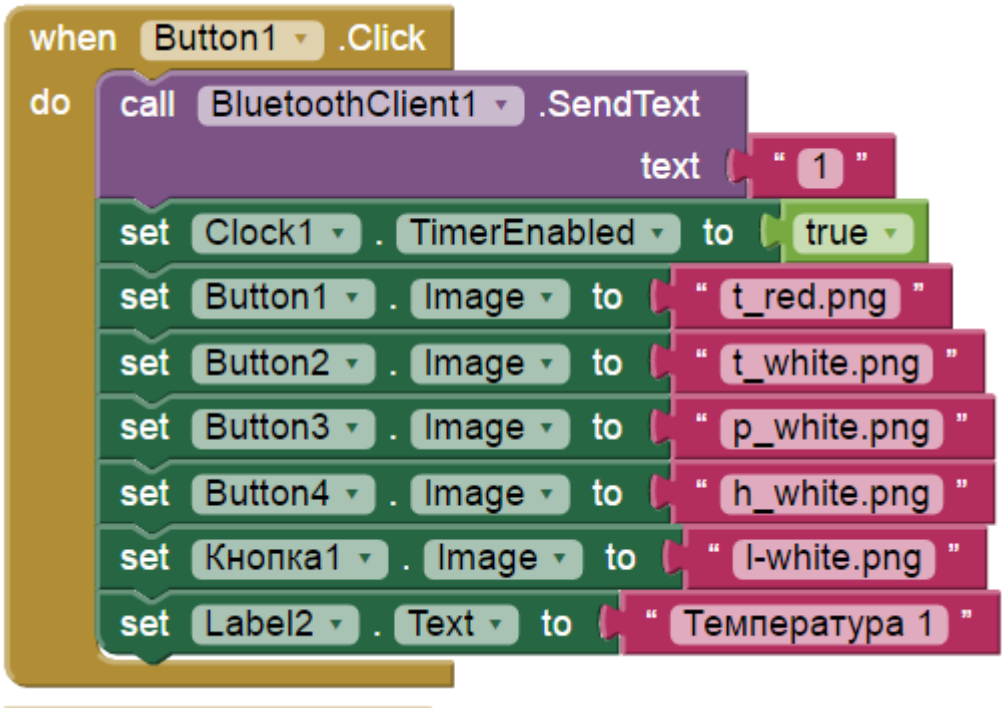

Рисунок 36 – Обработка события нажатия на кнопку

На рисунке 30 изображён модуль обработки события нажатия на кнопку Button1. Первым делом при нажатии на метеостанцию отправляется сигнал для получения данных о температуре. После этого запускается таймер. Все картинки, кроме картинки, соответствующей температуре, остаются белыми. На экран выводится текст с названием параметра «Температура1».

Теперь осталось только запрограммировать событие получения и отображения данных. В данной программе в качестве идентификатора выступает таймер. Он запускается только после того, как на метеостанцию отправлено сообщение с номером параметра. Если таймер запущен, то приложение ждёт ответных сообщений с текстом, содержащим показатели датчиков. Первоначально в цикле проверяется работает ли таймер. Если таймер работает, то проверяется действительно ли подключение к метеостанции. Если метеостанция подключена, то проверяется сигнал, поступающий от метеостанции. Если же сигнал не пустой, то в элемент Label1

выводится текст, поступивший с метеостанции. В данном случае этот текст соответствует показаниям датчиков, установленных на метеостанции.

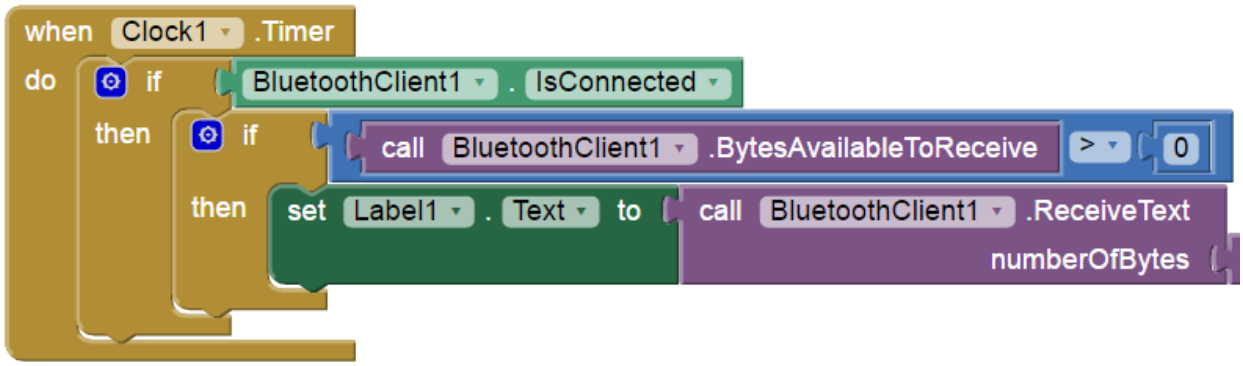

Рисунок 37 – Обработка получения данных

7 Разработка базы данных

В качестве системы управления базой данных использовался MS SQL Server. В данном проекте база данных использовалась для хранения данных, полученных программным обеспечением, расположенным на персональном компьютере, через COM порт, полученных с метеостанции.

Для этого использовалась специальная таблица, хранящая в себе 6 параметров. Можно было бы создать базу данных по сложнее, в которой каждому параметру соответствовала бы определённая таблица, но данное действие лишено какого-либо смысла.

В данную базу данных будут записываться следующие параметры: температура с датчика AM2301, соответствующая столбцу temperature1, влажность с датчика AM2301, соответствующая столбцу humidity, температура с датчика BMP180, соответствующая столбцу temperature2, давление с датчика BMP180, соответствующее столбцу pressure, освещённость с датчика BH1750, соответствующая столбцу light и системное время с датой, считываемое программой в момент записи из системных данный операционной системы Windows, соответствующее столбцу date_time.

Значение температуры может изменяться в точности до десятых и сотых частей градуса. Этот параметр зависит от точности измерения датчика. Вне зависимости от этого, данные упрощаются и сохраняются в базе данных в точности до десятичных долей. В связи с этим значения температур, полученных от датчиков температуры, записываются в базу данных с округлением до десятых долей и имеют тип данных float. Давление в данной метеостанции измеряется в паскалях, а не в миллиметрах ртутного столбца и не в атмосферном давлении. В связи с этим значение давления поступает в базу данных в точности до единиц.

Имя столбца	Тип данных	Разрешить значения NULL
humidity	int	☑
temperature1	float	☑
temperature2	float	☑
pressure	int	☑
light	int	☑
date_time	datetime	☑

Рисунок 38 – Структура основной таблицы

Значение влажности воздуха измеряется в процентах от 0 до 100%. Поэтому при измерении влажности датчик передаёт на плату Arduino целочисленные значения. В базу данных программа передаёт данные о влажности так же в целочисленном виде, поэтому при хранении значения влажности в базе данных используется тип данных int, который соответствует типа данных integer, используемому в C#.

Значения освещённости, получаемые с датчика BH1750, могут колебаться от 0 до 65000. Поэтому в базе данных используется тип данных int.

Для сохранения даты в базе данных программа считывает системные дату и время. Для таких данных в бд MS Sql имеется специальных типа данных datetime, в которых можно записать одновременно и дату, и время в точности до тысячных секунд.

Конечный вид таблицы, сохранённой в базе данных можно увидеть на рисунке 33.

humidity	temperature1	temperature2	pressure	light	date_time
41	24,8	25,2	24502	12	2016-09-05 14:01:15.000
41	24,8	25,3	24500	1	2016-09-05 14:01:35.000
41	24,8	25,3	24500	18	2016-09-05 14:01:56.000
42	24,8	25,3	24499	19	2016-09-05 14:02:16.000
42	24,8	25,3	24488	20	2016-09-05 14:02:36.000
42	24,8	25,3	24487	20	2016-09-05 14:02:57.000
42	24,9	25,3	24491	20	2016-09-05 14:03:17.000
42	24,9	25,3	24485	20	2016-09-05 14:03:37.000
42	24,9	25,3	24485	20	2016-09-05 14:03:58.000
42	24,9	25,3	24486	20	2016-09-05 14:04:18.000
42	24,9	25,3	24477	20	2016-09-05 14:04:38.000
40	24,1	24,7	24456	6	2016-09-05 16:09:45.000
40	24,1	24,7	24455	10	2016-09-05 16:10:05.000
40	24,1	24,7	24462	10	2016-09-05 16:10:26.000
40	24,1	24,7	24451	10	2016-09-05 16:10:46.000
40	24,1	24,7	24465	10	2016-09-05 16:11:06.000
39	24,2	24,8	24479	4	2016-09-05 16:35:04.000
39	24,2	24,8	24481	4	2016-09-05 16:35:25.000
40	24,2	24,8	24483	6	2016-09-05 16:35:45.000
39	24,1	24,8	24487	6	2016-09-05 16:44:53.000
39	24,2	24,8	24485	5	2016-09-05 16:45:13.000
39	24,2	24,8	24477	5	2016-09-05 16:45:33.000
40	24,2	24,8	24477	5	2016-09-05 16:45:54.000

Рисунок 39 – Таблица, содержащая данные

8 Безопасность жизнедеятельности

8.1 Пожароопасность

Защита против пожаров является таким средством, которое используется упрощения нежелательных последствий пожаров, несущих за собой какой-то вред. В эту систему включаются изучегие того, как поведёт себя пожар, пресечение и расследование пожара и связанных с ним чрезмерно нежелательными ситуациями, а также научные исследования и разрабатывающие, производства, тестирование и применение смягчающих систем. В структурах, будь то наземные, морские или даже корабли, владеющие и операторы несут ответственность за поддержание своих объектов в соответствии с дизайном, которой коренится в законах, в том числе местного строительного законодательства, такого как кодекс и пожарного кодекса, которые введены в действие с помощью органа, который препоручен это делать. Здания должны быть построены с полным учётом с версией строительного кодекса, который действует, когда заявление о выдаче разрешения на строительство производится. Строительные инспекторы проверяют на гармоничность строящегося здания со структурной схемой здания. После завершения постройки, здание должно поддерживаться в аналогии с действующим кодексом пожаробезопасности, который будет обеспечиваться соблюдение сотрудниками пожарной охраны местного пожарного департамента. В случае, когда вдруг возник пожар или неприятная ситуация, связанная с пожарами, пожарных следователей и другой персонал по предотвращению пожаров, тщательно опрашивается для исследования и выяснения повреждения от пожара. Уроки, извлеченные из пожаров, применяются при создании строительных норм и пожарных правил.

Пожароохранное мероприятие - это термин для всех мер, которые надо использовать за преждевременно, чтобы противодействовать возникновению и распространению пожаров структурными, технологическими и организационными мерами, а также уменьшить негативное действие пожаров на текущее окружение. Следовательно, противопожарная защита для профилактики делится на: структурная защита от пожара, технологическая система защиты от огня, организационная защита против огня.

В строительных нормах и правилах значение профилактической защиты от пожаров предназначена для защиты жизни, окружающей среды и общественной от воздействия огненного тепла и требуется в качестве предварительного условия для эффективной борьбы с очень опасными пожарами.

В дополнение нормам и правилам для построек, требования основаны в отношении защиты собственности в частных юридических соглашениях. Решающим требованием может быть наличие какой-то собственности у застрахованных людишек.

Разнообразие правовых норм, касающихся регулирования сферы ответственности предотвращения пожаров. В дополнение к основным социологическим, гуманитарным, политическим и экономическим требованиям основного закона и конституции, к правилам, которые используются в защите от пожара, особенно в законах защищённости от огня и строительных нормативах и правил страны, которая, в свою очередь, конкретизируются правила, директивы, различные постановления, технических регламентов и стандартов, рекомендаций и технических данных.

Таким образом при постройке любого здания строители и другие люди, которые участвуют в постройке сего здания используют некоторые правила и нормативы, которые помогают обеспечить построенное помещение всеми свойствами, которые помогут преждевременно устранить пожар или другие случаи, когда огонь может предоставить людям и хозяевам какие-то неприятности. Но помимо данных правил используются ещё другие различные приспособления, которые помогают избежать затрат на покрытие восстановления здания после того, как случился пожар.

Например, для того, чтобы вовремя узнать о пожаре и вовремя успеть потушить его устанавливаются специальные системы, которые сообщают различными способами о том, что что-то горит. Такие системы установлены в большей части зданий, в которых ведётся какое-либо производство и во всех помещениях, где есть большое количество компьютер и другой техники.

Подобные системы обычно состоят из нескольких компонентов. Чаще всего на потолке комнаты устанавливается специальный датчик дыма. Этот датчик улавливает молекулы дыма и передаёт информацию в процессор. Есть несколько видов таких датчиков. Некоторые из них начинают громко пищать, сообщая о том, что что-то горит. Другие же передают информацию в кабинет службы безопасности или же в специальные пожарные службы. Третьи же имеют свою систему для снабжения водой и автоматически начинают пытать тушить помещение жидкостью. У последних имеется определённый минус. Если же срабатывание будет ложным, то система для тушения может водой повредить бумаги или технику.

Рисунок 40 – Датчик дыма

Но часто бывают случаи, когда катастрофу избежать невозможно. Для таких случаев необходимо придумать план действий, который позволит уменьшить количество ущерба, который может нанести огонь при пожаре. Самым важным, что может пострадать при пожаре, является человеческая жизнь. Поэтому прежде всего в подобных ситуациях необходимо эвакуировать рабочий персонал из опасной зоны поражения. Для этого человек, отвечающий за технику безопасности в здании, изображает на стене план эвакуации из помещения. Подобный план должен присутствовать на каждом этаже здания.

Рисунок 41– Пример плана эвакуирования

8.2 Расчёт времени для эвакуации персонала из производственного здания

Учитывая определённые нормы, используемые при проектировании надо определить расчётное и необходимое время, которое будет затрачено на эвакуацию людей из сооружений производственного здания.

Исходные данные

Категория, которую присвоили данному производству – Д (пониженная пожароопасность)

Объём помещения – 148 $м^3$

Число людей на первом участке – 10 чел

Длина участка, м:	ширина участка, м
Первого $i_1 = 10$	первого $b_1 = 12$
Второго $i_2 = 8$	второго $b_2 = 2$
Третьего $i_3 = 5$	третьего $b_3 = 2$
Четвёртого $i_4 = 5$	четвёртого $b_4 = 2$

Учитывая все исходные данные надо построить схему эвакуации согласно СНиПу РК 2.02-05-2009 «ПОЖАРНАЯ БЕЗОПАСНОСТЬ ЗДАНИЙ И СООРУЖЕНИЙ»

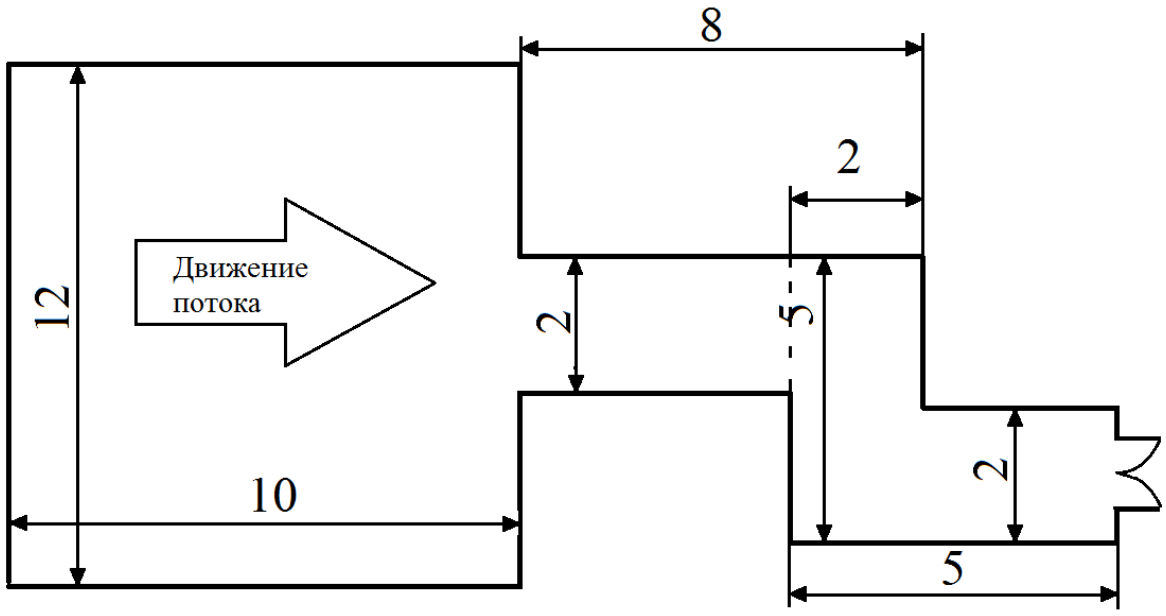

Рисунок 42– Расчётная схема эвакуации.

2.Среднюя площадь, которую занимает проекция человека в горизонтальном пространстве в зимней одежде надо принять равной 0,125 $м^2$

3.Ширину дверного проёма принимает 1,6 м

Определяем

А) плотность потока, состоящего из людей, на первом участке по формуле

$$D_1 = N_1 f/(i_1 b_1)$$

Подставим значения, получим

$D_1 = 10\times0,125/(10\times12) = 0,010417 \approx 0,01$ м^2/м^2

$D_2 = 10\times0,125/(8\times2) = 0,078 \approx 0,08$ м^2/м^2

$D_3 = 10\times0,125/(5\times2) = 0,125$ м^2/м^2

$D_4 = 10\times0,125/(5\times2) = 0,125$ м^2/м^2

Б)скорость, которую развивает поток из людей по пути горизонтального направления на первом участе v1=100 м/мин, а интенсивность движения людского потока q1=1 м/мин согласно СНиП РК 2.02-05-2009 (интерполируя значения таблицы1);

В) время движения людского потока на первом участке

$t_1 = l_1/v_1 = 12/100 = 0,12$ мин;

г) интенсивность движения людского потока q на втором, третьем и четвёртом участках пути

$q_i = q_{i-1} b_{i-1}/b_i$

Подставим числовые значения

$q_2 = 1\times12/2 = 6$ м/мин; $q_3 = 6\times2/2 = 6$ м/мин; $q_4 = 6\times2/2 = 6$ м/мин.

Соответственно скорости движения на подобных участках по данным сверочной таблицы $v_2 = 93$ м/мин, $v_3 = 93$ м/мин, $v_4 = 93$ м/мин.

Время движения:

$t_2 = l_2/v_2 = 8/93 = 0,086$ мин; $t_3 = l_3/v_3 = 5/93 = 0,054$ мин;

$t_4 = l_4/v_4 = 5/93 = 0,054$ мин;

д) расчётное время эвакуации людей как сумму времени движения по последовательным участкам и времени прохода дверного проема:

$$t_{расч} = t_1+t_2+t_3+t_4+t_{дв.пр.}$$

Согласно нормам проектирования при толщине стены менее 0,7 м длина пути в дверном проеме можно принимать равной нулю. Тогда $t_{дв.про}=0$. Следовательно $t_{расч} = t_1+t_2+t_3+t_4 = 0,12+0,086+0,054+0,054 = 0,314$ мин.

Е)необходимое время для эвакуации людей tнб в зависимости от степени огнестойкости сооружений, категории производства по взрывопожароопасности и объёма сооружений, которое для предоставленных условий $t_{нб} = 0,5$ мин

5.Расчётное количество времени для эвакуации людей их производственных сооружений здания $t_{расч} = 0,314$ мин, т.е. $t_{расч}<t_{нб}$. Условия обеспечения безопасности выполняются, следовательно, изменять схему эвакуации не требуется.

8.3 Молниезащита

Грозовые разряды содержат удивительные количества электрической энергии, которые изменяются от нескольких тысяч ампер до более чем 200

000 ампер - достаточно для питания половины миллиона лампочек в 100 ватт. Несмотря на это, разряд молнии имеет очень короткий срок существования (обычно 200 мкс), что является весьма реальной причиной повреждений и разрушений.

Последствия прямого удара очевидны и сразу бросаются в глаза - здания повреждены, деревья разлетаться, личные травмы и даже смерть. Однако вторичные эффекты молнии - кратковременность, высокие всплески напряжения, называемые переходным перенапряжением, бывают столь же катастрофическим, если визуально менее очевидно, то примером таких разрушений является повреждение электронных систем внутри здания. Мы постоянно встречаться с людьми, которые имеют структурную защиту от молний для их строительства, но понесли ущерб на - незащищенными - электронные системы изнутри. Проще говоря, структурная система защиты от молнии не может и не будет защищать электрические системы внутри здания от повреждения переходного перенапряжения. Надежная схема защиты от молнии должна охватывать как структурную защиту от воздействий молнии, так и переходных перенапряжений (электрических систем) защиты. Подобный вид защиты называется молниезащитой для производственных зданий. Одним из решений подобной проблемы является установка громоотводов.

Громоотвод представляет собой металлический стержень или металлический предмет, установленный на вершине возвышенности структуры, такие как здания, корабли или даже дерево, электрически соединенных с помощью проволоки или электрическим проводником для взаимодействия с землей или "землей" через электрод , спроектированный для защиты структуры в случае грозового удара. Если молния попадает в структуру, она будет предпочтительно ударять в стержень и проводиться на землю через провод, вместо того, чтобы проходить через структуры, где она могла бы вызвать пожары или вызвать поражения электрическим током. Молниеотводы также называются наконечниками, воздушными терминалами или устройствами для прекращения ударов молний.

В системе защиты от молнии, громоотвод является основным компонентом системы. Громоотводом требует подключения к земле, чтобы выполнять свою защитную функцию. Молниеотводы бывают разных форм, в том числе полые, твердый, заостренные, округлые, плоские полосы или даже щетинного типа. Главным атрибутом общим для всех молниеотводов является то, что все они сделаны из электропроводных материалов, таких как медь и алюминий. Медь и ее сплавы являются наиболее распространенными материалами, используемыми в молниезащиты.

Принцип громоотвода был впервые подробно распределён Бенджамином Франклином в Пенсильвании в 1749 году, который в последующие годы пытался развивать своё изобретение для бытового

применения, а также дальнейшего совершенствования в направлении надежной системы ближе к 1760 году.

С то время как здания становятся выше угроза от молний увеличивается. Молния может привести к повреждению конструкции из большинства материалов, таких как кирпичная кладка, дерево, сталь и бетон, так как огромные токи и напряжения, из которых состоит заряд молнии могут нагревать материалы до высокой температуры, в результате чего может произойти пожар.

Система защиты от молнии предназначены для защиты конструкции от повреждения вследствие ударов молнии, перехватывая такие удары и безопасно проводя их чрезвычайно большие токи на землю. Система защиты от молнии включает в себя сеть аэровокзалов, приклеенных проводников и заземляющих электродов, предназначенных для обеспечения пути низкого импеданса на землю для потенциальных ударов.

Системы защиты от молнии используются для предотвращения или уменьшения ущерба от удара молнии по несущим структурам. Системы защиты от молнии смягчают опасность возникновения пожара, который молния может создать при попадании в какой-либо материал. Система защиты от молнии обеспечивает путь низкого сопротивления для тока молнии, чтобы уменьшить тепловой эффект тока, протекающего через легковоспламеняющиеся конструкционные материалы. Если молния проходит через пористые и водонасыщенные материалы, эти материалы могут буквально взорваться, если их содержание воды мелькнула паром под воздействием тепла, полученного от высокого тока. Именно поэтому деревья часто разрушены ударами молнии.

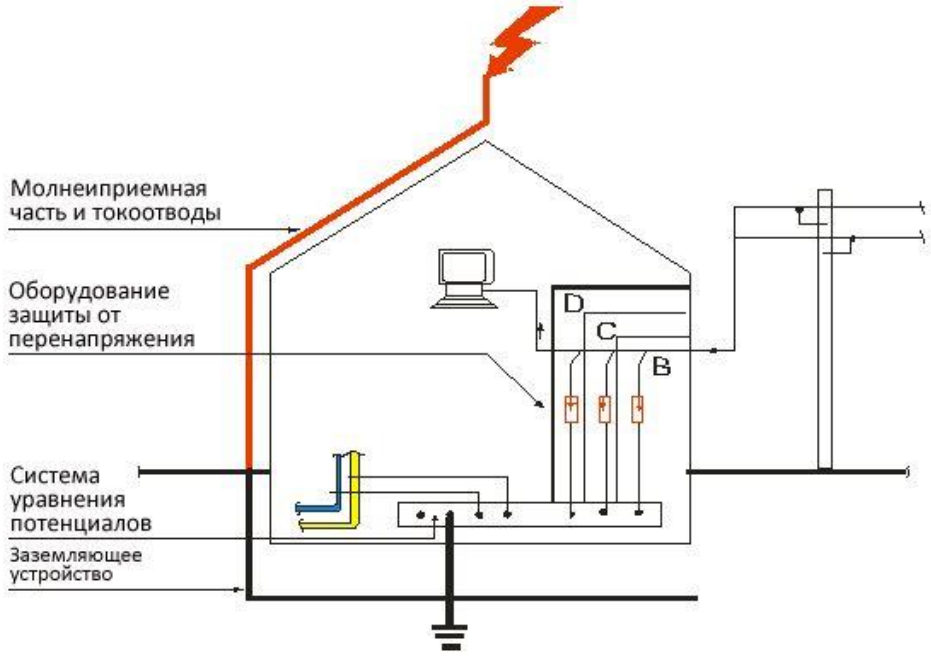

Рисунок 43– Структура системы молниезащиты.

Рисунок 44 – Структура тросовой молниезащиты

8.4 Расчёт молниезащиты производственного здания одиночным стержневым молниеотводом

Исходные данные.

Категория сооружения по молниезащите.

Размеры объекта, м:

Длина А = 30;

Ширина В = 25;

Высота Н = 20;

Интенсивность громовых действий nч=30…40 ч/год.

Решение. 1. Молниеотвод обязан быть установленным в центре крыши сооружения. Схема зоны защищенности сооружения одиночным стержнеобразным молниеотводом приведена на рисунке 45.

2.Определяем вероятную численность ударов молний в год в сооружение.

$$N = (A+6h_x)(B+6h_x)n\times10^{-6}, \qquad (8.1)$$

Где А, В- соответственно ширина и длина подзащитного сооружения, м;

h_x – наибольший уровень здания по вертикали, м;

n – среднее количество раскатов молнии в год на 1 км^2. Согласно Таблице8.2 в зависимости от интенсивности грозовых действий $n_ч$ = 30…40 ч/год принимаем n = 2,5.

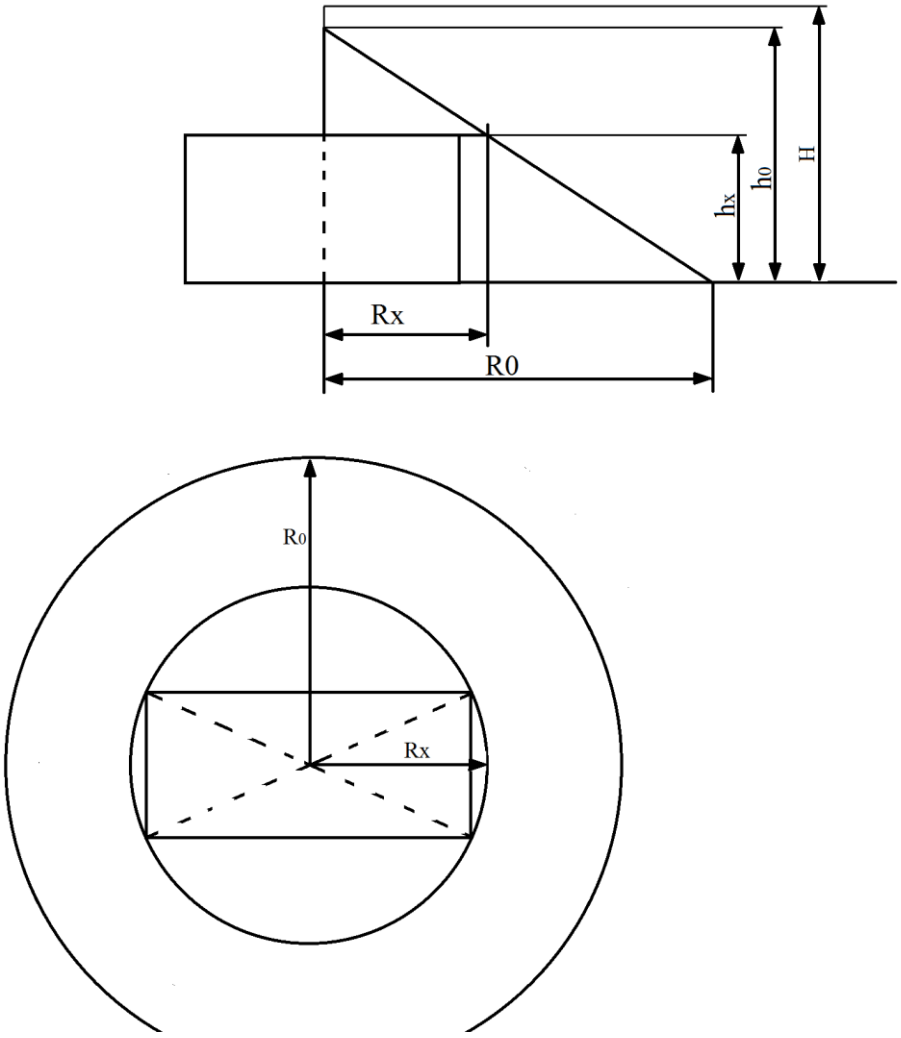

Рисунок 45 – Схема зоны защищенности сооружения одиночным стержнеобразным молниеотводом.

По схеме защиты горизонтальное сечение зоны защиты Rx представляет половину диагонали защищаемого здания.

$$R_x = \sqrt{(\frac{A}{2})^2 + (\frac{B}{2})^2}. \quad (8.2)$$

Поставив значения получим

$R_x = \sqrt{(\frac{30}{2})^2 + (\frac{25}{2})^2} = 19{,}5$ м. Принимаем 20 м.

Тогда $N = (30+6\times20)(25+6\times20)\times2{,}5\times10^{-6} = 0{,}054$

Согласно рекомендациям СН РК 2.04-29-2005 «Инструкция по устройству молниезащиты зданий и сооружений» при $0{,}1 < N \leq 2$ принимаем тип зоны защиты молниеотвода – Б.

Согласно СН РК 2.04-29-2005 минимальное расстояние между молниеотводом и объектом принимается равным нулю, т.к. допускается

установка молниеотвода на крыше зданий сооружений II и III категорий по молниезащите.

Основные параметры типа зоны защиты Б рассчитываются по формулам:

$h_0 = 0,92 \times h$;

$R_0 = 1,5 \times h$;

$R_x = 1,5 \times (h - h_x/0,92)$;

$h = (R_x + 1,63 h_x)/1,5$.

Согласно имеющимся данным последовательно вычисляем искомые параметры.

$h = (20 + 1,63 \times 20)/1,5 = 34,7$ м;

$h_0 = 0,94 \times 34,7 = 32,6$ м;

$R_0 = 1,5 \times 34,7 = 52$ м.

Таблица 8.1 – Таблица для расчёта времени эвакуации.

Плотность потока D, м2/м2	Горизонтальный путь		Дверной проём, интенсивностью q, м/мин	Лестница вниз		Лестница вниз	
	Скорость v, м/мин	Интенсивность q, м/мин		Скорость v, м/мин	Интенсивность q, м/мин	Скорость v, м/мин	Интенсивность q, м/мин
0,01	100	1	1	100	1	60	0,6
0,05	100	5	5	100	5	60	3
0,1	80	8	8,7	95	9,5	53	5,3
0,2	60	12	13,4	68	13,6	40	8
0,3	47	14,1	16,5	52	16,6	32	9,6
0,4	40	16	18,4	40	16	26	10,4
0,5	33	16,5	19,6	31	15,6	22	11
0,7	23	16,1	18,5	18	12,6	15	10,5
0,8	19	15,2	17,3	13	10,4	13	10,4
0,9 и более	15	13,5	8,5	8	7,2	11	9,9

Таблица 8.2 – Усреднённая численность попаданий молнии на участок площадью 1км^2 в зависимости от интенсивности громовой деятельности

Грозовая деятельность, ч в год	Среднее число поражений молнией
20-40	2,5
40-60	3,8
60-80	5
80-100	6,3
Более 100	7,5

9 Технико-экономическое обоснование проекта

Описание работы и обоснование необходимости.

Основной целью данного проекта является создание мобильной метеостанции, позволяющей определять погодные данные для небольших локаций, а также создание программного обеспечения для неё, с помощью которого можно побед считывать, анализировать и сохранять данные о погодных условиях. В различных районах крупных городов иногда погода отличается. Домашняя метеостанция позволить узнать точные данные о погодных условиях определённых точках или же в конкретном помещении. Данную технологию так же можно использовать для регулирования температуры, влажности, освещённости и давления в лабораториях или на рабочем месте.

9.1 Расчет трудоемкости работ и объема разработки программного обеспечения микропроцессора

В данной дипломной работе собирается метеостанция и разрабатывается программное обеспечение для неё. Вся работа состоит из следующих этапов:

- постановка задачи
- разработка алгоритмов проектирования
- работа над прибором
- работа над программой

Таблица 9.1.1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел.× ч.
1.постановка задачи	начальное теоретическое описание предстоящих работ, определение входных и выходных параметров при работе с прибором и программой	16
2.разработка алгоритмов проектирования	разработка алгоритма работы с устройством, разработка алгоритма работы программы	16
3.работы над прибором	определение необходимых элементов прибора для решения поставленных задач, сборка корпуса, тестирование и настройка датчиков	40 (из них 5 часов работы с паяльником)
4.работы над программой	Разработка программы для работы с прибором на основе разработанного алгоритма, подключение и решение проблем взаимодействий аппаратной части с программной, дальнейшее тестирование	152
ИТОГО трудоемкость выполнения работы		224

$$T_н = 16 + 16 + 40 + 152 = 224 \text{ чел.} \times \text{ч}$$

Следовательно, на всю работу уйдёт 224 часов работы. Учитывая то, что работник будет работать по 8 часов в день, посчитаем трудоёмкость разработки в днях.

$$T_д = T_н/8 = 224/8 = 28 \text{ чел.} \times \text{дней}$$

Время затраченное на полное исполнение поставленных задач считается следующим образом

$$Тр = T_д / (Ч_р \times Ф_{эф}) \quad (9.1)$$

где $T_д$ – трудовая ёмкость разработки в сутках;
$Ч_р$ – числа планируемых работничков;
$Ф_{эф}$ − эффектный фонд времени работы.

Эффектный фонд времени работы одного работничка ($Ф_{эф}$) рассчитывается по формуле

$$Ф_{эф} = Д_г − Д_п − Д_в − Д_о \quad (9.2)$$

где $Д_г$ − количество дней в выбранном году;
$Д_п$ − количество праздничных дней в выбранном году;
$Д_в$ − количество выходных дней в выбранном году;
$Д_о$ − количество дней отпуска.

Количество рабочих дней при пятидневной рабочей неделе за 2016 год равно 245 дням (разница между кол-вом дней году и праздничными днями с выходными).

$$Ф_{эф} = 245 – 24 = 221 \text{ дней}$$

Запланированное число работников $Ч_р = 1$, следовательно, время, которое необходимо затратить на разработку посчитается по формуле 3.5

$$Тр = 28/ (1 \times 221) = 0.1267 \text{ лет}$$

Итого на разработку данного программного продукта нам понадобится $365 \times 0.1267 = 46$ календарных дней.

Результаты всех расчетов, которые произведены и представлены в таблицу 9.1.2.

Таблица 9.1.2 − Объем и трудоемкость разработки программного обеспечения

Название	Условное обозначение	Значение
Трудовая емкость разработки	$Т_н$	224 чел.× ч
Трудовая емкость разработки в днях	$Т_д$	28
Эффектный фонд времени работы	Фэф	221 дня
Срок разработки проекта	Тр	0. 1267 лет
Численность исполнителей проекта	Чр	1 чел

9.2 Расчет затрат

Расчет абсолютно всех затрат, которые будут использованы на разработку проектного решения в виде информационных технологий ($С_{пі}$) осуществляется по формуле

$$С_{пі} = З_{фот} + З_{сзі} + М_і + Р_{сі} + Р_{мі} + Р_{нкі} + П_{зі} + Р_{ні} \quad (9.3)$$

где $З_{фот}$ – полный объём оплаты труда исполнителей, тенге;

$З_{сзі}$ – отчисления по социальному налогу, тенге;

$М_і$ – затраты на материалы, тенге;

$Р_{сі}$ – затраты на специальные программы, необходимые для разработки проектного решения, тенге;

$Р_{мі}$ – затраты, на использование технических средств, тенге;

$Р_{нкі}$ – затраты на различные командировки, тенге;

$П_{зі}$ – остальные затраты, тенге;

$Р_{ні}$ – накладные расходники, тенге.

Размер объёма оплаты трудов исполнителей ($З_{фот}$) рассчитывается по формуле

$$З_{фот} = З_о + З_{ді} \quad (9.4)$$

где $З_о$ – основная оплата трудов, тенге;

$З_{ді}$ – дополнительная оплата трудов, тенге.

Основная оплата трудов исполнителей на конкретное ПО рассчитывается по формуле

$$З_{оі} = ΣТ_{чі} \times Т_ч \times Ф_п \quad (9.5)$$

где n − количество исполнителей, занятых выполнением конкретного ПО;

$Т_{чi}$ − почасовая тарифное отчисление i-го исполнителя (тыс.тенге);

$Ф_{п}$ − плановый объём исполнительного времени исполнителя (дней);

$Т_{ч}$ − количество времени работы в день (час) (8 часов);

По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов-исполнителей, участвующих в разработке ПО, с определением образования, специальности, квалификации и должности представлено в таблице 9.1.3.

Таблица 9.1.3 − Задействованные в проекте работники

Исполнитель	Количество, человек	Заработная плата, тенге
Инженер-программист	1	120 000
Итого	1	120 000

Почасовая тарифная подставка рассчитывается путем деления месячной тарифной подставки на установленную при 40-часовой недельному объёму рабочего времени расчетную среднемесячную норму рабочего времени в часах ($Ф_{р}$) (22 дня – при пятидневной рабочей неделе)

$$ЧС = Т_{м} / Ф_{р} \quad (9.6)$$

$$Ф_{р} = 22 \times 8 = 176 \text{ часов}$$

где ЧС− часовая тарифная ставка (тыс.тенге);

$Т_{м}$ − месячная тарифная ставка (тыс.тенге).

По формуле 3.9 можно определить часовую тарифную ставку инженер-программиста

$$ЧС = \frac{120\,000}{176} = 682 \text{ тг/час}$$

Определив объём зарплаты исполнителя за один час работы, мы можем получить суммарную зарплату исполнителя, определяемую как сумма оплаты труда всех исполнителей, задействованных в работе. Для этого необходимо подсчитать зарплату на каждом этапе исполнения. Суммарная зарплата будет равна сумме зарплат на каждом из частей исполнения.

$$З_{о} = \sum_{i=1}^{n} З_{оi} = \sum_{i=1}^{n} ЧСi \times Т_{нi}$$

где $ЧС_i$– почасовая тарифицированная ставка исполнителя в данной части работы(тыс.тенге);

$Т_{нi}$ – трудовая емкость данной части работы(тыс.тенге).

С учётом того, что на всех частях исполнения выполняется один и тем же работником, то часовая ставка не изменяется для всех этапов. Вычислим зарплату исполнителя-работника для каждой части исполнения. Посчитаем основную оплату труда для каждого любой части исполнения:

$$З_{О1} = ЧС_1 \times Т_{Н1} = 682 \times 16 = 10\,912 \text{ тг.}$$

$$З_{О2} = ЧС_2 \times Т_{Н2} = 682 \times 16 = 10\,912 \text{ тг.}$$

$$З_{О3} = ЧС_3 \times Т_{Н3} = 682 \times 40 = 27\,280 \text{ тг.}$$

$$З_{О4} = ЧС_4 \times Т_{Н4} = 682 \times 152 = 103\,664 \text{тг.}$$

Тогда оплата труда за всю проделанную работу будет составлять:

$$З_{О1} = З_{О1} + З_{О2} + З_{О3} + З_{О4} = 10\,912 + 10\,912 + 27\,280 + 103\,664 = 152\,768 \text{ тг}$$

Дополнительная зарплата труда рассчитывается по формуле

$$З_д = З_о \times Н_д / 100 \qquad (9.7)$$

где $Н_д$ – коэффициент доп. оплаты труда. С учётом отсутствия доп. оплаты получаем

$$З_{фот} = З_{оi} = 152\,768 \text{ тг}$$

Социальный налог обязан составлять около 11% (ст. 358 п. 1 НК РК) от дохода исполнителя, и рассчитывается по формуле

$$З_{сзi} = (ФОТ - ПО) \times 11\% \qquad (9.8)$$

где ПО – пенсионные зачисления, которые составляют 10% от ФОТ и социальным налогом не облагаются

$$ПО = ФОТ \times 10\% \qquad (9.9)$$

$$ПО = 152\,768 \times 0.1 = 15\,276 \text{ тг}$$

Таким образом социальный налог составит

$$З_{сзi} = (152\,768 - 15\,276) \times 0.11 = 15\,124 \text{ тг}$$

Определим траты на материализованные ресурсы в таблице 9.1.4.

Таблица 9.1.4 − Траты на материализованные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Микроконтроллер Arduino Uno R3	шт	1	2300	2300
LCD1602 экран	шт	1	1200	1200
Датчик влажности и температуры DHT21	шт	1	2200	2200
Датчик атмосферного давления BMP180	шт	1	2300	2300
Монтажные провода (комплект)	шт	3	600	1800
SD карта	шт	1	1000	1000
Монтажная плата (400 выводов)	шт	1	800	800
Модуль микро SD	шт	1	800	800
Датчик CO для Arduino	шт	1	2100	2100
ИТОГО затраты на материальные ресурсы				14500

Расчитаем материальные траты на детали прибора.

M=2300+1200+2200+2300+1800+1000+800+800+2100=14500 тг.

При изготовлении метеостанции возникала необходимость использования электроприборов, отобразим используемые устройства в таблице 9.1.7, и рассчитаем траты на использование по формуле:

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц, \quad (9.10)$$

где M_i - паспортная мощь i-го электрооборудования, кВт;

K_i – уровень исполнения мощностей i-го электрооборудования (принимается K_i=0.7¸0.9);

T_i - время работы i-го оборудования за весь период исполнения, ч;

Ц - цена электроэнергии, тг/кВт×ч;

i - вид электрооборудования;

n - количество электрических устройств.

Персональный компьютер использовался в таких этапах разработки как постановка задачи, разработка алгоритмов проектирования, работы над программой. Посчитаем время работы персонального компьютера.

$T_к$=16+16+152=184 часов.

Электрический паяльник использовался на протяжении 5 часов в этапе работы над прибором.

Таблица 9.1.5 − Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки, ч	Цена электро энергии, $\frac{тг}{кВт \cdot ч}$	Сумма, тг
Персональный компьютер	0,45	0,7	184	23,12	1281,77
Электрический паяльник	0,04	0,7	5	23,12	3,24
ИТОГО затраты на электроэнергию					1285,01

Тогда на электричество при работе компьютера будет затрачено:

$$Э_к=0,45\times0,7\times184\times23,12=1281,77 \text{ тг}$$

На электричество при работе паяльника будет затрачено:

$$Э_п=0,4\times0,7\times5\times23,12=3,24 \text{ тг}$$

Общая сумма, затраченная на электричество:

$$Э= Э_к+ Э_п=1281,77+3,24=1285,01 \text{ тг.}$$

Рассчитаем амортизационную составляющую используемого оборудования в таблице 9.1.6, по формуле расчета амортизационной составляющей основных фондов (ОФ):

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{A\,i} \times Т_{НИРi}}{100 \times Т_{ЭФi}} \qquad (9.11)$$

где $Ф_i$ -ценность i-го ОФ, тг;

$Н_{Ai}$ - годовая норма амортизационной составляющей i-го ОФ, %;

$Т_{НИРi}$ - время исполнения i-го ОФ за всё время исполнения ПП, ч;

$Т_{Эфi}$ - эффективный размер времени работы i-го ОФ за год, ч/год;

i - вид ОФ;

n - количество ОФ.

Эффективный объём времени исполнения:

$$Э_{эф}= Ф_{эф}\times Т_{рд} \qquad (9.12)$$

где $Ф_{эф}$ - Эффективный объём времени исполнения;
$Т_{рд}$ – время рабочих суток.

$$Э_{эф}= 221\times8=1768 \text{ ч/год}$$

Расчет амортизационной части всех фондов для ПК:

$$З_{АМ\ ПК} = \frac{130000 \times 20 \times 184}{100 \times 1768} = 2705{,}9 тг$$

Расчет амортизационной части головных фондов для программных продуктов

$$З_{АМ\ WIN} = \frac{24000 \times 25 \times 184}{100 \times 1768} = 624{,}4 тг$$

$$З_{АМ\ MSO} = \frac{21000 \times 25 \times 184}{100 \times 1768} = 546{,}4 тг$$

Амортизационная часть головных фондов:

$$З_{АМ}=З_{АМ\ ПК} + З_{АМ\ WIN}+З_{АМ\ MSO} \quad (9.13)$$

$$З_{АМ}=2705{,}9 + 624{,}4 + 546{,}4 = 3876{,}7 \text{ тг}$$

Таблица 9.1.6 - Амортизационная часть головных фондов

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Персональный компьютер	130000	20	1768	184	2705,9
Microsoft Windows 7 Ultimate 32/64-bit	24000	25	1768	184	624,4
Microsoft Office 2010 Standard	21000	25	1768	184	546,4
ИТОГО амортизация основных фондов					3876,7

Проведём расчет вспомогательных трат, которые включат в себя стоимость арендных плат, коммунальной услуги, услуги коммуникации и интернет связи.

Согласно договору с арендодателем помещение прощадью 30 м^2 обходилось в 45 тысяч тенге в месяц. На работу было затрачено 224 чел. × час, с учётом вычетов проведённых ранее на исполнение программного продукта было использовано 46 календарных дней. С учётом этого комната была арендована на 2 месяца. Посчитаем затраты на арендодателя:

$$З_а = \frac{45\ 000}{1} \times 2 = 90\ 000 \text{ тг.}$$

Расчет стоимости услуги предоставления интернета зависит от времени его использования. С учётом того, что на исполнение потрачено 46 календарных дней интернет был оплачен на 2 месяца. При стоимости услуги интернет 4500 тг/мес, необходимая расчетная затрата на него ($З_и$) равна:

$$З_и = 4500 \times 2 = 9000 \text{ тг}$$

Итого вспомогательные затраты ($З_д$) равны:

$$З_д = З_а + З_и = 90000 + 9000 = 99000 \text{ тг}$$

Итак, определив составные части стоимости исполнения данного проекта, мы можем рассчитать суммарные траты на производство продукта, которые представляют собой сумму всех, выше рассчитанных величин. Определим общие траты в таблице 9.1.7.

Таблица 9.1.7 - Смета затрат на разработку продукта

Статьи затрат	Сумма, тг	Соотношение, %
1. Материальные затраты, в том числе:	-	5,51
- материалы	14500	5,06
- электроэнергия	1285,01	0,45
2. Затраты на оплату труда.	152 768	53,31
3. Отчисления на социальные нужды.	15 124	5,28
4. Амортизация основных фондов.	3876,7	1,35
5. Прочие затраты.	99000	34,55
ИТОГО	286553,71	100

Полные затраты:
$С_{пi}$=14500+1285,01+152768+15124+3876,7+99000=286553,71 тг.

Итогами всех вычислений является значение З = 286553,71 тг. Данная цифра состоит из всех основных потраченных ресурсов и услуг, используемых при выполнении данного продукта. Это позволит объективно оценить стоимость самого продукта при его дальнейшей реализации.

Как говорилось ранее, метеостанцию можно использовать не только для получения погодных данных вне помещения. Прибор так же можно использовать для обеспечения комфортных условий в кабинете или контроля

воздуха в лабораториях. В связи с этим прибор может оказаться конкурентоспособным на рынке. С учётом небольших затрат на компоненты можно наладить можно так же наладить массовое производство.

Стоимость производства метеостанции включает сумму полной себестоимости и сумму прибыли.

Сумма прибыли можно попробовать рассчитать по установленному уровню возможной рентабельности и составляет 15% от стоимости исполнения.

Полная стоимость исполнения работы (П) составила 262631 тг, тогда рассчитаем возможную получаемую прибыль:

$$П = 286553{,}71 \times 0{,}15 = 42983{,}05 \text{ тг.}$$

Стоимость произведения прибора и программного обеспечения (С), с учетом получения прибыли, составляет:

$$С = 286553{,}71 + 42983{,}05 = 329536{,}76 \text{ тг.}$$

При учете существующего НДС (12%), оно будет равен:

$$С_{НДС} = 329536{,}76 \times 0{,}12 = 39544{,}41 \text{ тг;}$$

Тогда стоимость прибора и программного обеспечения ($С_П$) для заказчиков составляет:

$$С_П = 329536{,}76 + 39544{,}41 = 369081{,}17 \text{ тг.}$$

Построим диаграмму, которая покажет отношение типов затрат к итоговой цене на рисунке 9.1.8.

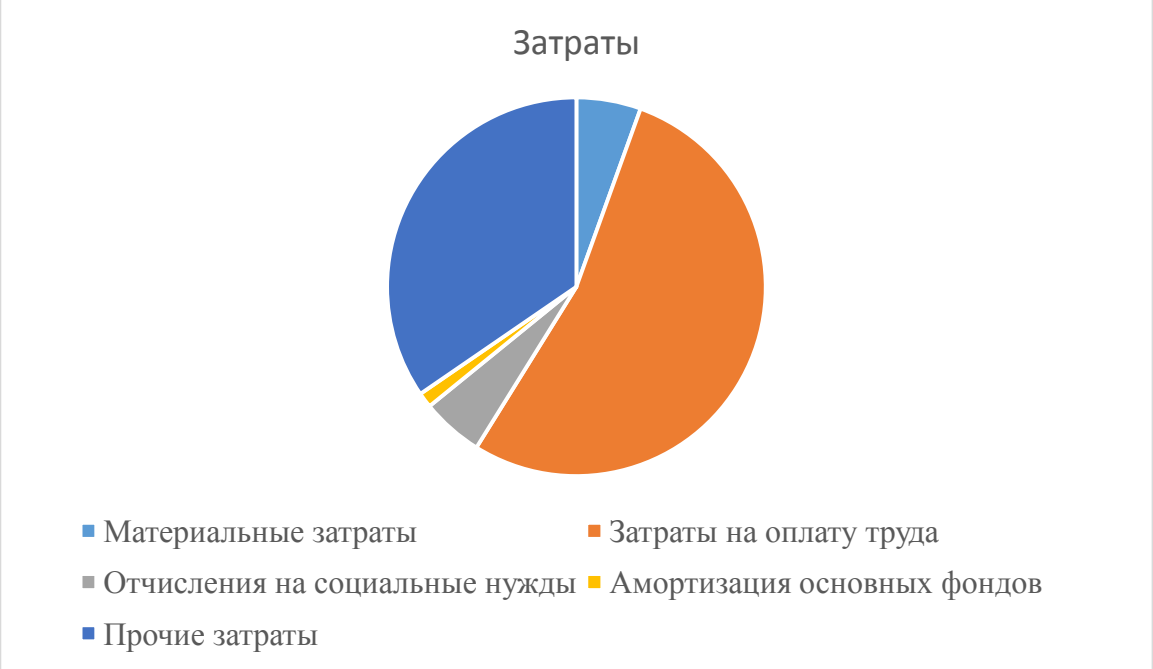

Рисунок 9.1.8 - Диаграмма затрат

9.3 Выводы

В данной работе были проделаны расчёты затрат на сборку домашней метеостанции и программного обеспечения для неё, а также была посчитана окончательная стоимость продукта для покупателя. Итогом расчётов являются цифры $С_{пі}$=286553,71 тг (затраты на разработку продукта) и $С_{П}$369081,17 тг (окончательная стоимость готового продукта).

На рынке Казахстана продаются домашние метеостанции, но они не имеют программного обеспечения, позволяющего хранить, отображать и обрабатывать полученные данные на компьютере. Разработанный продукт может заинтересовать человека, занимающегося исследованием погодных данных или же сборкой информации о погодных условиях определённой местности или помещения. Благодаря подобной уникальности продукта он может заинтересовать учёных и исследователей и окупиться.

С учётом небольших расходов на детали метеостанции (14500 тг) и наличия уже готового программного обеспечения есть возможность наладить производство метеостанций и продажу метеостанций вместе с готовым программным обеспечением. Это позволит значительно увеличить прибыль от реализации проекта.

Заключение

В ходе выполнения данной дипломной работы была собрана стабильно функционирующая метеостанция. Была разработана система для автоматической записи данных на SD карту во время работы метеостанции. Так же было разработано приложение для операционной системы Windows для просмотра данных с метеостанции и сохранения их в базу данных. Была разработана база данных, а также система для визуализации данных.

Было разработано мобильное приложение для операционной системы Android для удалённого получения данных с метеостанции.

При выполнении технико-экономического расчёта были произведены вычисления затрат на сборку метеостанции, создание программных продуктов и на оплату труда работников. Было выяснено наличие достаточной прибыли при налаживании производства метеостанций для продажи при наличии определённого спроса на них.

Список используемой литературы

1. Сайт https://www.gismeteo.kz/
2. Сайт https://pogoda.nur.kz/
3. Сайт https://www.arduino.cc/
4. Программирование микроконтроллерных плат Arduino/Freeduino / Улли Соммер - БХВ-Петербург, 2012.
5. Базы данных. Язык SQL / Ржеуцкая С.Ю - Вологда: ВоГТУ, 2010.
6. Microsoft SQL Server 2008. Основы T-SQL / Ицик Бен-Ган - БХВ-Петербург, 2009.
7. C# 4.0. Полное руководство/ Герберт Шилдт – Издательство Вильямс, 2015.
8. C# 4.0 и платформа .NET 4 для профессионалов / Кристиан Нагел, Билл Ивьен - Издательство Вильямс, 2011.
9. Сайт http://ai2.appinventor.mit.edu/
10. Информационная технология: Вопросы развития и применения. - Киев: Наук. думка, 1988.
11. Сайт https://www.visualstudio.com

Приложение А

```
#include <SPI.h>
#include <SD.h>
#include <DS1302.h>
#include "DHT.h"
#include <Wire.h>
#include <BH1750.h>
#include <Adafruit_BMP085.h>
#include <LiquidCrystal_I2C.h>
#define DHTPIN 2
#define DHTTYPE DHT21
Adafruit_BMP085 bmp;
BH1750 lightMeter;
DS1302 rtc(2, 3, 4);
LiquidCrystal_I2C lcd(0x27,16,2);
DHT dht(DHTPIN, DHTTYPE);
File myFile;
void setup() {
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  dht.begin();
  bmp.begin();
  lightMeter.begin();
  if (!SD.begin(4)) {
    return;
  }
}
void loop() {
  int h = dht.readHumidity();//переменная для влажности
  float t = dht.readTemperature();//переменная для температуры
  float t2 = bmp.readTemperature();
  int p = bmp.readPressure();
  uint16_t lux = lightMeter.readLightLevel();
  if (isnan(t) || isnan(h)) {//проверка (что мы получаем на запрос из датчика - цифры?)
  }
  else {
    Serial.print(h);//вывожу построчно - влажность
    Serial.print("\t\n");//сдвиг каретки и начало строки
    Serial.print(t);//температура
    Serial.print("\t\n");//сдвиг каретки и начало строки
    Serial.print(t2);//вывожу построчно - температура
    Serial.print("\t\n");
    Serial.print(p);//вывожу построчно - давление
    Serial.print("\t\n");
    Serial.print(lux);//освещённость
    Serial.print("\n");
  }
    lcd.clear();
```

```
    lcd.print("T=");
    lcd.print(t);
    lcd.print(" t=");
    lcd.print(t2);
    lcd.setCursor(0, 1);
    lcd.print("L=");
    lcd.print(lux);
    lcd.print(" H=");
    lcd.print(h);

    myFile = SD.open("Data.txt", FILE_WRITE);
    if (myFile) {
  myFile.println(rtc.getDOWStr());
  myFile.print(rtc.getDateStr());
  myFile.print(" ");
  myFile.println(rtc.getTimeStr());
  myFile.println(h);
  myFile.println(t);
  myFile.println(t2);
  myFile.println(p);
  myFile.println(lux);
    myFile.close();
    }
  delay(20000);
}
```

Приложение Б

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;
using System.Data.SqlClient;

namespace arduino
{
   public partial class Form1 : Form
   {
      SqlConnection con = new SqlConnection("Data Source=Azik;Initial Catalog=Meteo1;User ID=Ara; Password=Ara; integrated security=true");
      DataSet ds = new DataSet();
      SqlCommand myCommand;
      SqlDataReader myReader;
      String Orcon;
      DataTable dt;
      SqlCommand cmd;
      StreamReader streamReader;
      String Fname;

      SerialPort port1 = new System.IO.Ports.SerialPort("COM3", 9600, Parity.None, 8, StopBits.One);
      public Form1()
      {
         InitializeComponent();
         notifyIcon1.Visible = false;
         this.notifyIcon1.MouseClick += new MouseEventHandler(notifyIcon1_MouseClick);
         this.Resize += new System.EventHandler(this.Form1_Resize);
         try
         {
            port1.Open();
         }
         catch
         {
         }
         port1.DataReceived += serialPort1_DataReceived;
      }

      private void serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
```

```
        {
            string vlag = port1.ReadLine();
            string temp = port1.ReadLine();
            string temp2 = port1.ReadLine();
            string davl = port1.ReadLine();
            string svet = port1.ReadLine();
            this.BeginInvoke(new LineReceivedEvent(LineReceived), vlag, temp, temp2, davl, svet);
        }
        private delegate void LineReceivedEvent(string vlag, string temp, string temp2, string davl,
string svet);
        private void LineReceived(string vlag, string temp, string temp2, string davl, string svet)
        {
            textBox1.Text = vlag;
            textBox2.Text = temp;
            textBox3.Text = temp2;
            textBox4.Text = davl;
            textBox5.Text = svet;
            string path = "Все данные.txt";
            string date = DateTime.Now.ToString();
            string doweek = DateTime.Now.DayOfWeek.ToString();
            String X1 = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

            // Создание файла и запись в него
            using (StreamWriter sw = File.AppendText(path))
            {
                sw.WriteLine(doweek);
                sw.WriteLine(date);
                sw.WriteLine(vlag);
                sw.WriteLine(temp);
                sw.WriteLine(temp2);
                sw.WriteLine(davl);
                sw.WriteLine(svet);
            }
            con.Open();
            cmd = new SqlCommand("Insert into
AllValues(date_time,temperature1,temperature2,humidity,pressure,light) VALUES('" + X1 + "'," +
temp + "," + temp2 + "," + vlag + "," + davl + "," + svet + ")");
            cmd.Connection = con;
            try
            {
                cmd.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show(X1 + " "+temp + " "+temp2 + " "+vlag+" "+davl+" "+svet);
            }
            con.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
            Grafik form2 = new Grafik(this);
            form2.Show();
            this.Hide();
        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            if (this.WindowState == FormWindowState.Minimized)
            {
                this.Hide();
                notifyIcon1.Visible = true;
            }
        }

        private void notifyIcon1_MouseClick(object sender, MouseEventArgs e)
        {
            this.Show();
            this.WindowState = FormWindowState.Normal;
            notifyIcon1.Visible = false;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Table form3 = new Table(this);
            form3.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                Fname = openFileDialog1.FileName;
            }
        }
    }
}
```

Приложение В

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;

namespace arduino
{
   public partial class Grafik : Form
   {
      Form1 F1;
      OpenFileDialog openFileDialog1;
      StreamReader streamReader;
      String Fname;
      public Grafik(Form1 a)
      {
         F1=a;
         InitializeComponent();
      }

      private void button3_Click(object sender, EventArgs e)
      {
         if (openFileDialog2.ShowDialog() == DialogResult.OK)
         {
            Fname = openFileDialog2.FileName;
            comboBox1.Visible = true;

         }
      }

      private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
      {
         streamReader = new StreamReader(Fname);
         chart1.Series[0].Points.Clear();
         while (!streamReader.EndOfStream)
         {
            string doweek = streamReader.ReadLine();
            DateTime X = Convert.ToDateTime(streamReader.ReadLine());
            string Y = streamReader.ReadLine();
            string T1 = streamReader.ReadLine();
            string T2 = streamReader.ReadLine();
            string Z = streamReader.ReadLine();
```

```
            string R = streamReader.ReadLine();
            chart1.Series[0].Color = Color.Red;
            chart1.Series[0].BorderWidth = 2;
            switch (comboBox1.SelectedItem.ToString())
            {
               case "Влажность":
                  chart1.Titles["Title1"].Text = "График влажности воздуха(%)";
                  chart1.Series[0].Points.AddXY(Convert.ToString(X), Y);
                  chart1.Series[0].LegendText = "%";
                  break;
               case "Температура1":
                  chart1.Titles["Title1"].Text = "График температуры 1(°C)";
                  chart1.Series[0].Points.AddXY(Convert.ToString(X), T1);
                  chart1.Series[0].LegendText = "°C";
                  break;
               case "Температура2":
                  chart1.Titles["Title1"].Text = "График температуры 2(°C)";
                  chart1.Series[0].Points.AddXY(Convert.ToString(X), T2);
                  chart1.Series[0].LegendText = "°C";
                  break;
               case "Давление":
                  chart1.Titles["Title1"].Text = "График атмосферного давления(Па)";
                  chart1.Series[0].Points.AddXY(Convert.ToString(X), Z);
                  chart1.Series[0].LegendText = "Па";
                  break;
               case "Освещённость":
                  chart1.Titles["Title1"].Text = "График освещённости(Лк)";
                  chart1.Series[0].Points.AddXY(Convert.ToString(X), R);
                  chart1.Series[0].LegendText = "Лк";
                  break;

               }
            }
            streamReader.Close();
         }
         private void Grafik_FormClosing(object sender, FormClosingEventArgs e)
         {
            F1.Show();
         }
         private void button1_Click(object sender, EventArgs e)
         {
            this.Close();
         }
      }
   }
```

Приложение Г

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace arduino
{
   public partial class Table : Form
   {
      Form1 F1;
      SqlConnection con = new SqlConnection("Data Source=Azik;Initial Catalog=Meteo1;User ID=Ara; Password=Ara; integrated security=true");
      DataSet ds = new DataSet();
      SqlCommand myCommand;
      SqlDataReader myReader;
      DataTable dt;
      SqlDataAdapter oda;
      SqlCommand cmd;
      public Table(Form1 a)
      {
         F1 = a;
         InitializeComponent();
         this.dataGridView1.DefaultCellStyle.Font = new Font("Tahoma", 12);
         this.dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Tahoma", 11);

      }

      private void button1_Click(object sender, EventArgs e)
      {
         dataGridView1.DataSource = SelectTable();
         dataGridView1.Columns[5].Width = 140;
         dataGridView1.Columns[0].HeaderText = "Влажность %";
         dataGridView1.Columns[1].HeaderText = "Температура 1 °C";
         dataGridView1.Columns[2].HeaderText = "Температура 2 °C";
         dataGridView1.Columns[3].HeaderText = "Давление";
         dataGridView1.Columns[4].HeaderText = "Освещённость";
         dataGridView1.Columns[5].HeaderText = "Дата";
      }

      private void Table_FormClosing(object sender, FormClosingEventArgs e)
      {
         F1.Show();
```

```
        }


        public DataTable SelectTable()//Функция считывания таблицы и базы данных
        {
            con.Open();
            oda = new SqlDataAdapter("Select * from AllValues", con);
            dt = new DataTable();
            oda.Fill(dt);
            con.Close();
            return dt;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            con.Open();
            oda = new SqlDataAdapter("Select * from AllValues where CAST(date_time AS
DATE)='" + dateTimePicker1.Value.ToString("yyyy-dd-MM") + "' ", con);
            dt = new DataTable();
            try
            {
                oda.Fill(dt);
            }
            catch
            {
            }
            con.Close();
            dataGridView1.DataSource = dt;
            dataGridView1.Columns[5].Width = 140;
            dataGridView1.Columns[0].HeaderText = "Влажность %";
            dataGridView1.Columns[1].HeaderText = "Температура 1 °C";
            dataGridView1.Columns[2].HeaderText = "Температура 1 °C";
            dataGridView1.Columns[3].HeaderText = "Давление";
            dataGridView1.Columns[4].HeaderText = "Освещённость";
            dataGridView1.Columns[5].HeaderText = "Дата";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```