МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки/специальности)

Электроника и робототехника
(направленность (профиль)/специальности)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Метеостанция специального назначения

Обучающийся Д.С. Якушев
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук, доцент М.В. Позднов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант О.А. Головач
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Объем 94 с., 40 рис., 1 табл., 21 источник, 3 прил. Метеостанция специального назначения.

Объектом исследования является портативная метеостанция, предназначенная для использования ракетными войсками и артиллерией для организации метеорологической подготовки дивизиона (батареи).

Цель работы — разработать прототип устройства, способного снимать метеоданные и, экстраполируя их на основе программно прописанных таблиц значений, составлять приближенный бюллетень «Метеосредний».

Задачи работы заключались в разработке электронной, программной части устройства, корпуса, схемы электрической соединений. Также, задачами были разработка интуитивно понятного интерфейса, подбор электронных модулей и датчиков, отвечающих требованиям технического задания.

Работа состоит из четырех разделов, в которых решены упомянутые задачи.

В процессе разработки был создан прототип устройства, на котором производилась проверка всех модулей и датчиков, а также работа программы устройства.

Степень внедрения – устройство по разработанной документации является опытным образцом.

Областью применения данной системы является метеорологическая подготовка подразделений ракетных войск и артиллерии при выполнении огневых задач.

Применение данного устройства ускорит метеорологическую подготовку огневых подразделений в рамках выполнения правил стрельбы и управления огнем, а также повысит точность рассчитываемых поправок.

Abstract

The topic of the given graduation work is «A single-purpose weather station». The graduation work consists of 94 pages, including 40 figures, 1 table, 21 references, 3 appendices, and 6 graphical representations on A1-sized sheets.

The aim of the work is to develop the prototype of the equipment with the following functions: to measure the weather parameters, to extrapolate parameters for altitudes and to prepare the meteorological average. The necessary requirement is to create the interface with a keyboard and a display.

The object of the graduation work is a portable weather station meant to be used by an artillery unit for meteorological preparation of an artillery battalion (company). The main microcontroller unit of the equipment is ATmega328p on circuit board Arduino Nano.

The design process was divided into: developing hardware, and software, designing the body tools, assembling electronic hardware. Developing hardware includes selecting electronic parts (sensors, interface modules and other parts), and creating an electric schematic diagram.

Programming involves making software with C++ language for microcontroller ATmega328p. At the same stage the program code has been corrected. Developing all body parts includes designing in CAD and 3D printing.

Assembling electronic hardware is the process of mounting elements on the stripboard with soldering.

In conclusion, this portable weather equipment will be used in the artillery force of the Russian Federation to prepare artillery units, effectively and fast, for successful battle tasks.

Перечень сокращений и обозначений

РВиА – ракетные войска и артиллерия

ОП – огневая позиция

РЭБ – радиоэлектронная борьба

ПСиУО – правила стрельбы и управления огнем

ДМК – десантный метеорологический комплект

д. у. – деления угломера

б. д. у. – большие деления угломера

МК – микроконтроллер

СОБ – старший офицер батареи

PPR – количество импульсов за оборот

EEPROM – электрически стираемое перепрограммируемое постоянное запоминающее устройство

САПР - система автоматизированного проектирования

Содержание

Введение

Опыт специальной военной операции показывает, что сейчас боевые действия – это комплексная система с огромным количеством переменных, которые для успешного осуществления тактических действий надо учитывать.

В случае артиллерии основными тактическими нововведениями являются [5]: мобильность подразделений, тщательная маскировка ОП, относительная автономность отдельных тактических единиц, а также активное использование кочующих орудий. Подразделения все время должны быть готовы к быстрой передислокации, т.к. противник массово использует разведывательные дроны, средства спутниковой и радиолокационной разведки, а также средства радиоэлектронной борьбы. Таким образом, подразделения, только прибывшие на позицию и совершившие 1-2 выстрела, могут быть обнаружены противником. Учитывая наличие у вооруженных сил Украины, а также стран потенциального противника, готовых интегрированных систем разведки и управления огнем - в течение 2-3 минут по ОП может быть сначала открыт пристрелочный огонь, а затем через 30 секунд огонь на поражение, что потребует немедленной смены дислокации и очень быстрой подготовки новой ОП.

Для выполнения огневой задачи необходимо совершить многие подготовительные процедуры, одной из которых является метеорологическая подготовка. Учитывая особенности ведения современных боевых действий артиллерии, отмеченные выше, старшие офицеры батарей сами на месте доступными средствами получают данные о метеорологических условиях, чаще всего используя программы, установленные на смартфон, или свои портативные метеостанции.

Наиболее рациональным является использование именно портативных метеостанций, которые должны отвечать требованиям: малых габаритов, относительно высокой точности, быстрого приведения в рабочее положение.

1 Постановка задачи

1.1 Выявление проблемы

Подготовка стрельбы и управления огнем в дивизионе (батарее) проводится в целях непрерывного поддержания артиллерийских подразделений в постоянной готовности к выполнению огневых задач с высокой эффективностью. Одним из пунктов подготовки стрельбы и управления огнем является метеорологическая подготовка [6], которая включает в себя получение таких параметров, как:

- высота метеопоста над уровнем Балтийского моря, метры;
- температура воздуха, С°;
- атмосферное давление, мм рт. с.;
- скорость ветра, м/с;
- направление ветра, б. д. у.

Далее, все данные сводятся в приближенный бюллетень «Метеосредний» либо автоматическим образом, при помощи средств ведения огня на смартфонах, либо вручную при помощи специальных бланков (по нормативу не должно превышать 2 минут). Приближенный бюллетень «Метеосредний» составляет метеорологический пост дивизиона по результатам приземных метеорологических измерений с использованием комплектов метеоприборов. Срок годности приближенного бюллетеня «Метеосредний», составленного с помощью комплекта метеоприборов при определении установок способом полной (сокращенной) подготовки, равен 1 часу.

Таким образом, для эффективного выполнения огневых задач необходимо иметь комплект метеоприборов, обеспечивающий автоматическое снятие метеоданных и формирование приближенного бюллетеня «Метеосредний», а также функцию контроля времени годности бюллетеня.

Рассмотрим, какие средства для определения метеорологических условий используются в войсках РВиА на данный момент.

1.2 Поиск и анализ технических параметров аналогичных устройств

Задачей метеорологической подготовки является определение метеорологических условий, учитываемых при определении установок для стрельбы.

Основным принятым на вооружение средством метеорологической подготовки в войсках на данный момент является ДМК. Приведем его технические данные на основе технического описания от 1980 года:

а) мгновенная скорость ветра:

1) диапазон измерений от 1,5 до 40 м/с;

2) погрешность измерения не более $\pm(0{,}5+0{,}05V)$ м/с,

где V – измеренная скорость воздушного потока в м/с;

б) направление ветра:

1) диапазон измерений от 0 ° до 360 °,

2) погрешность измерения не более ± 10 °;

в) температура воздуха:

1) диапазон измерений от минус 55 ℃ до плюс 45 ℃;

2) погрешность измерения не более $\pm 0{,}8$℃;

г) атмосферное давление:

1) диапазон измерения от 560 до 800 мм рт. ст.,

2) погрешность измерений от $\pm 1{,}8$ до $\pm$ 5,4 мм рт. ст.

в зависимости от наземной температуры;

д) питание прибора:

1) серебряно-цинковые аккумуляторные батареи СЦ-25,

2) напряжение питания прибора $6 \pm 1{,}2$ В;

е) время разверстки ДМК одним человеком: 15 минут;

ж) масса всего комплекта не более 23 кг.

Изображение ДМК представлено на рисунке 1. Как можем видеть: в данном устройстве есть свое количество недостатков, ключевыми из которых являются: наличие аккумуляторных батарей, которые с течением времени выходят из строя; большие габариты; долгое время приведения в боевую готовность; незаменимость деталей (ДМК, которые производятся на данный момент имеют другие технические решения по сравнению с советскими, поэтому ремонт старых ДМК не представляется возможным новыми комплектующими).

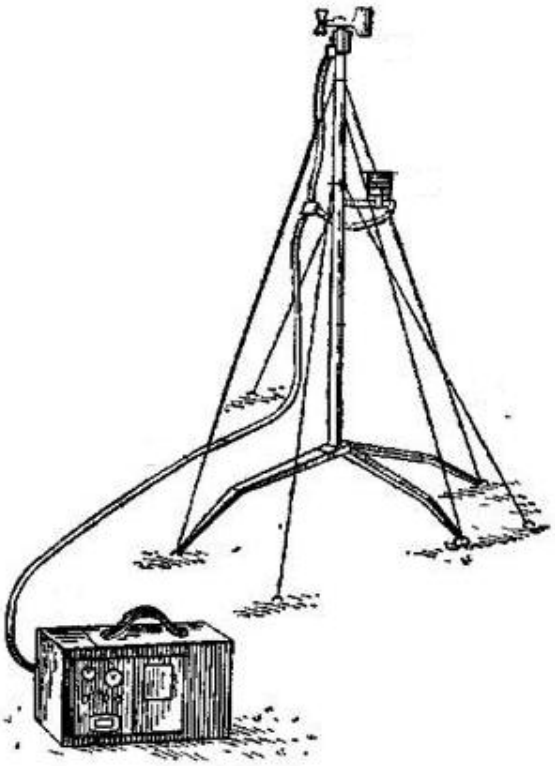


Рисунок 1 – ДМК в собранном состоянии

На данный момент особо актуальным является использование портативных метеостанций типа Kestrel, изображение одной из модели которых представлено на рисунок 2. Данные метеостанции предназначены в первую очередь для работы стрелков-снайперов, и поэтому данные средства снятия метеорологических данных являются, прежде всего, не специализированными под нужды артиллерии, а также чрезмерно дорогостоящими по причине того, что не являются отечественными и

содержат в себе дорогостоящие комплектующие и сложные программно-аппаратные решения. Но, тем не мене, данные средства получения метеоданных используются артиллеристами из-за отсутствия альтернатив.

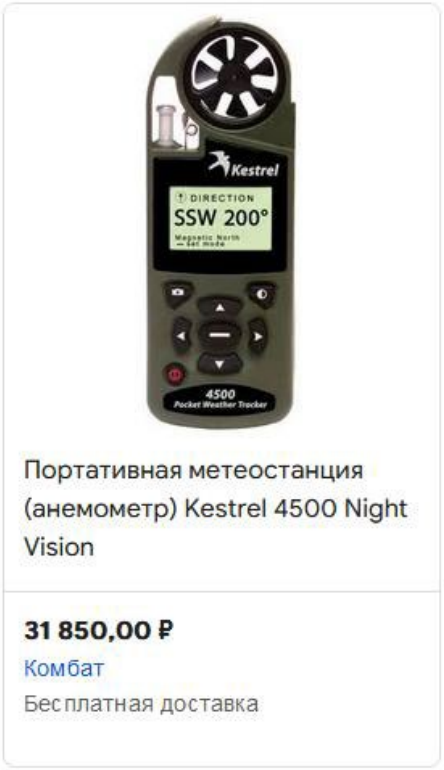


Рисунок 2 – Портативная метеостанция (анемометр) Kestrel 4500 NV

Приведем основные технические данные метеостанции Kestrel 4500 из документации [18]:

а) Мгновенная скорость ветра:

1) диапазон измерений от 0,4 до 60 м/с;

2) погрешность измерения не более ±3%;

б) Направление ветра:

1) диапазон измерений от 0° до 360°;

2) погрешность измерения не более ±5°;

3) цена деления 1°;

в) Температура воздуха:

-диапазон измерений от минус 45°С до плюс 125°С,

- погрешность измерения не более ±1°C;

г) Атмосферное давление:

- диапазон измерения от 10 до 1100 мм рт. ст.,
- погрешность измерений ±1,5 мм рт. ст.;

д) Питания прибора:

- 2 батарейки тип AAA;

е) Время приведения в готовое положение: не найдено;

ж) Масса метеостанции 102 г.

Однозначно можно сказать, что данная метеостанция является более подходящим для сегодняшних боевых действий решением по сравнению с ДМК, но цена и тот факт, что метеостанции Kestrel западного производства, в данном случае делают критичным их использование подразделениями в боевых действиях. Также отметим, что многие модели снабжаются GPS навигацией, что в свою очередь несет в себе ещё и потенциальную угрозу.

Стоит также упомянуть появившиеся относительно недавно программы для смартфонов, позволяющие посредством GPS и интернета получать метеоданные и сразу производить для соответствующей системы и используемого боеприпаса расчеты поправок в дальность и направление. Но, как было отмечено выше, в реалиях ведения РЭБ, а также с условием различного рельефа и лесистости, частым событием является отсутствие связи, что не позволяет использовать смартфон для получения метеоданных. Таким образом, в лучшем случае – подразделение останется без метеорологической подготовки; а в худшем, все же выйдя в сеть – будет обнаружено средствами разведки противника [10]. Причем точность получаемых метеоданных тоже может не соответствовать требованиям организации эффективного выполнения огневых задач.

Перейдем к рассмотрению предлагаемой структуры устройства и применяемой в работе аппаратной части.

1.3 Структура устройства

На рисунке 3 представлена структурная схема устройства.

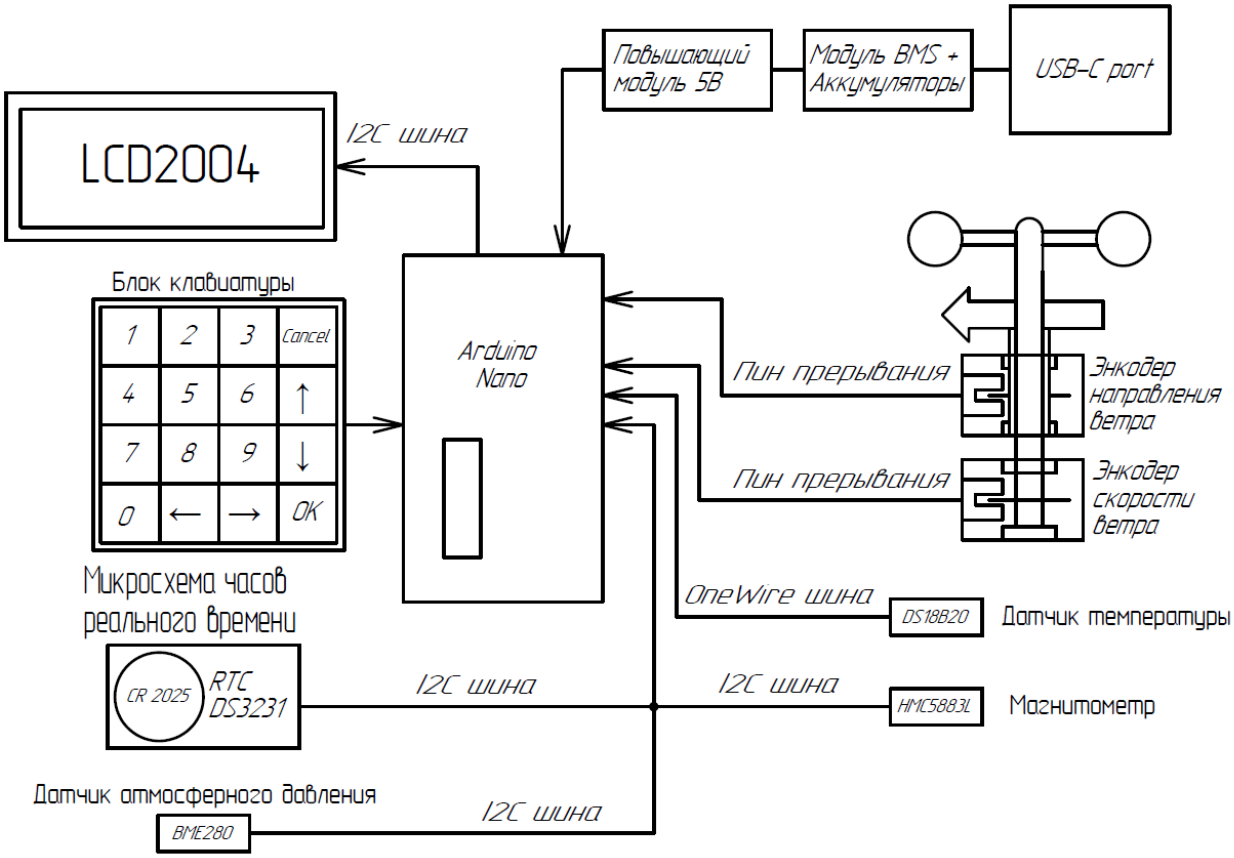


Рисунок 3 – Структурная схема устройства

На данной схеме можно видеть выбранные к использованию датчики и модули:

- плата Arduino Nano с МК ATmega328p,
- датчик атмосферного давления и температуры BME280,
- датчик температуры DS18B20,
- электронный компас GY-271 на микросхеме HMC5883L,
- энкодер E40H12-1200-3-N-5 с поддержкой Z сигнала и PPR равным 1200,
- энкодер серии LPD3806-600BM для снятия скорости ветра,
- микросхема часов реального времени RTC DS3231,
- дисплей LCD2004 I2C,

- модуль клавиатуры,
- модуль зарядки и защиты аккумуляторов на микросхеме TP4056 и DW01-P,
- повышающий до 5 В DC/DC преобразователь напряжения на микросхеме ME2108.

За основу для первой версии метеостанции было принято решение взять плату Arduino Nano, так как данная версия с МК ATmega328p является достаточно компактной и функциональной. Изображение данной платы представлено на рисунке 4. Также, у платы есть встроенный стабилизатор напряжения для внешнего питания, способный принять на вход до 12 В при 500 мА потребляемого платой тока – при проектировке схемы питания устройства может быть полезным использование данного стабилизатора. Не менее важным считаем наличие возможности использования протокола I2C на выводах 14 и 15 (A4 и A5 соответственно). В схеме и в программе для МК используется 2 внешних прерывания: одно для работы с энкодером направления ветра, и другое для снятия показаний для расчета скорости ветра.

Рисунок 4 – Используемая в устройстве отладочная плата Arduino Nano

При выборе микроконтроллера учитывались объемы необходимой памяти для программы и её выполнения. У ATmega328p SRAM память составляет 2 КБ (оперативная память), 32 КБ Flash (память, отвечающая за

хранение программы) и 1 КБ EEPROM-памяти (энергонезависимая перезаписываемая память, которая также используется в проекте) [13]. Отметим, что у данного микроконтроллера достаточно пинов для подключения всех используемых в проекте модулей и датчиков.

Выводы по первому разделу

Определив требования к устройству и основные компоненты, можем перейти непосредственно к описанию аппаратной части и последующему программированию устройства.

2 Конструкторское проектирование аппаратной части

Перейдем к рассмотрению датчиков и модулей, с указанием их эксплуатационных характеристик, точности снимаемых значений, а также используемых в программе библиотек.

2.1 Датчик температуры DS18B20

DS18B20 – цифровой датчик температуры, работающий по протоколу OneWire. Приведем его основные характеристики [16]:

- диапазон измерений от минус 55℃ до плюс 125 ℃;
- точность ±0,5 ℃;
- питание однополярное, в диапазоне от 3 до 5,5 В;
- разрешение датчика 9, 10, 11 или 12 бит (программно изначально 12 бит);
- период выдачи результата при 12 битном разрешении 750 мс;
- возможные корпус: TO-92, SOIC-8 или герметичный.

В нашем случае используется датчик DS18B20 в герметичном корпус, т.к. в устройстве конструктивно предполагается установка датчика в открытой части корпуса устройства (для повышения точности снимаемых данных), при этом герметичный корпус датчика позволит избежать воздействие влаги и других факторов. Схематическое изображение датчика и его распиновка представлена на рисунке 5.

Отметим, что информационный вывод датчика подтягивается к 5В питания через резистор номиналом 4,7 кОм.

Для взаимодействия с датчиком использовалась библиотека «microDS18B20», находящаяся в свободном доступе в интернете. Данная библиотека является облегченной версией официальной библиотеки «DallasTemperature.h». Опишем процесс получения температуры с датчика и основные методы библиотеки.

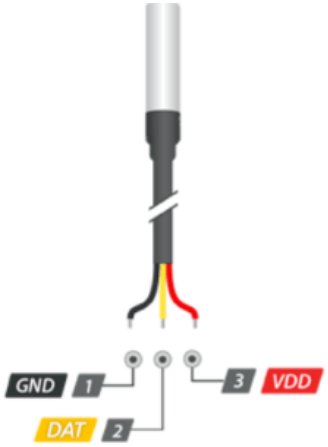


Рисунок 5 – Датчик температуры DS18B20 и его выводы

Для того, чтобы получить значение температуры с датчика необходимо сначала произвести запрос, и затем принять данные. Запрос делается методом «requestTemp()». После того, как датчик получил запрос, через 750 мс с 12 битным разрешением можно будет вернуть готовое значение с помощью метода «getTemp()» в типе данных float, или же через метод «getTempInt()» - вернется int. Если вызвать метод получения температуру до того, как температура будет измерена, датчик вернет предыдущее значение. Удобен также метод «readTemp()», позволяющий отдельно запросить температуру и определить корректность полученных данных, чтобы только после этого их прочитать и применить в программе. Также, это позволяет определить состояние подключения и исправность датчика.

2.2 Датчик влажности и атмосферного давления BME280

BME280 представляет собой комбинированное устройство, включающее в себя цифровой датчик влажности, давления и температуры. Приведем его основные характеристики, необходимых нам для разрабатываемого устройства [14]:

- диапазон рабочих температур составляет от минус 40℃ до плюс 85 ℃;
- диапазон измеряемого давления от 225 до 825 мм рт. ст.;
- точность измеряемого давления в среднем ±1 мм рт. ст.;

- диапазон измеряемых температур от минус 40℃ до 85 ℃;

- точность измерения температуры в среднем составляет ±1 ℃;

- питание однополярное, в диапазоне от 1,7 до 3,6 В;

- протокол обмена данными I2C.

Обратим внимание на питание: сам датчик поддерживает питание 3,3 В, но ATmega328p и плата Arduino рассчитаны на работу при 5В питания. В данном случае корректнее рассматривать не сам датчик, а плату, на которой он установлен (рисунок 6) со стабилизатором питания на 3,3 В. Таким образом, мы без опасений можем питать BME280 от цепи 5В.

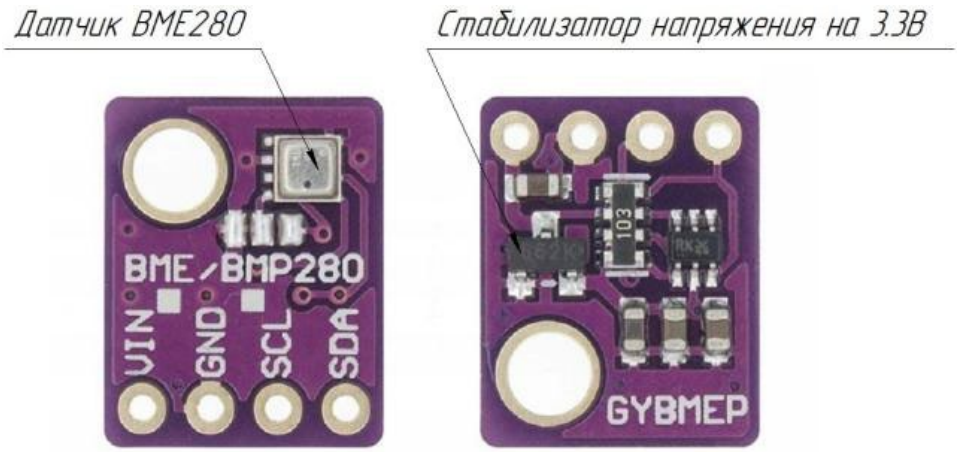


Рисунок 6 – Плата датчика BME280

Для работы с датчиком использовали находящуюся в свободном доступе библиотеку «GyverBME280». В библиотеке прописаны следующие методы взаимодействия с датчиком: метод «begin()» для инициализации датчика; метод «readTemperature()» для снятия значения температуры; метод «readPressure()» для снятия давления в Паскалях.

Снятие значения температуры с BME280 будет применяться в случае, если DS18B20 выйдет из строя, т.к. показатель точности у DS18B20 больше, при этом использование герметичного корпуса обеспечивает возможность, как писалось ранее, использования его в открытой части корпуса. Таким образом, измерение температуры с BME280 является резервным вариантом.

2.3 Электронный компас GY-271 на микросхеме HMC5883L

Для определения азимута магнитного ветра было принято решение использовать трёхосевой цифровой магниторезистивный компас GY-271 (рисунок 7) на микросхеме HMC5883L с 12-ти битным АЦП и интерфейсом I2C.

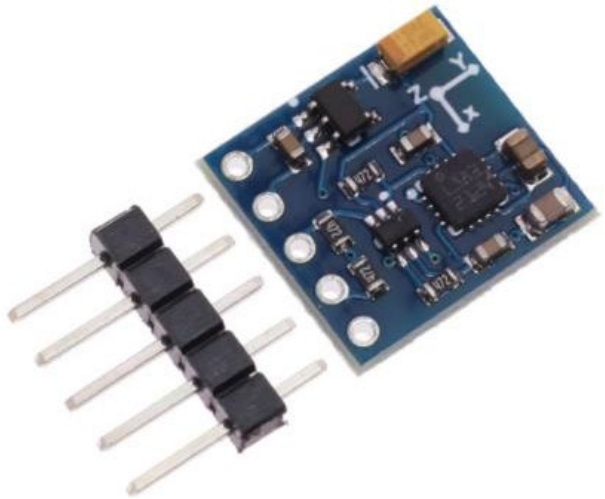


Рисунок 7 – Плата датчика GY-271

Приведем основные характеристики микросхемы HMC5883L, необходимых нам для разрабатываемого устройства [15]:

- диапазон рабочих температур от минус 40 до плюс 125 ℃;
- точность измерения азимута магнитного от 1° до 2°;
- питание однополярное, в диапазоне от 1,7 до 3,6 В;
- протокол обмена данными I2C.

Также, как и в случае с BME280, на плате GY-271 также имеется стабилизатор напряжения, поэтому микросхему HMC5883L можно беспрепятственно использовать в устройстве, подавая питание 5 В.

Для работы с датчиком использовалась библиотека «Wire.h», позволяющая работать по протоколу I2C, и библиотека «DFRobot_QMC5883.h», необходимая для работы с электронным компасом HMC5883L. Также, для внесения калибровочной матрицы в код программы

использовали программа «MagMaster», позволяющая рассчитать калибровочные данные для HMC5883L. Окно программы представлен на рисунке 8.

В библиотеке прописаны две ключевые функции: «getHeading()» - для получения значений с магнитометра; «transformation()» - полученные данные с магнитометра обрабатываются с учетом калибровочных данных.

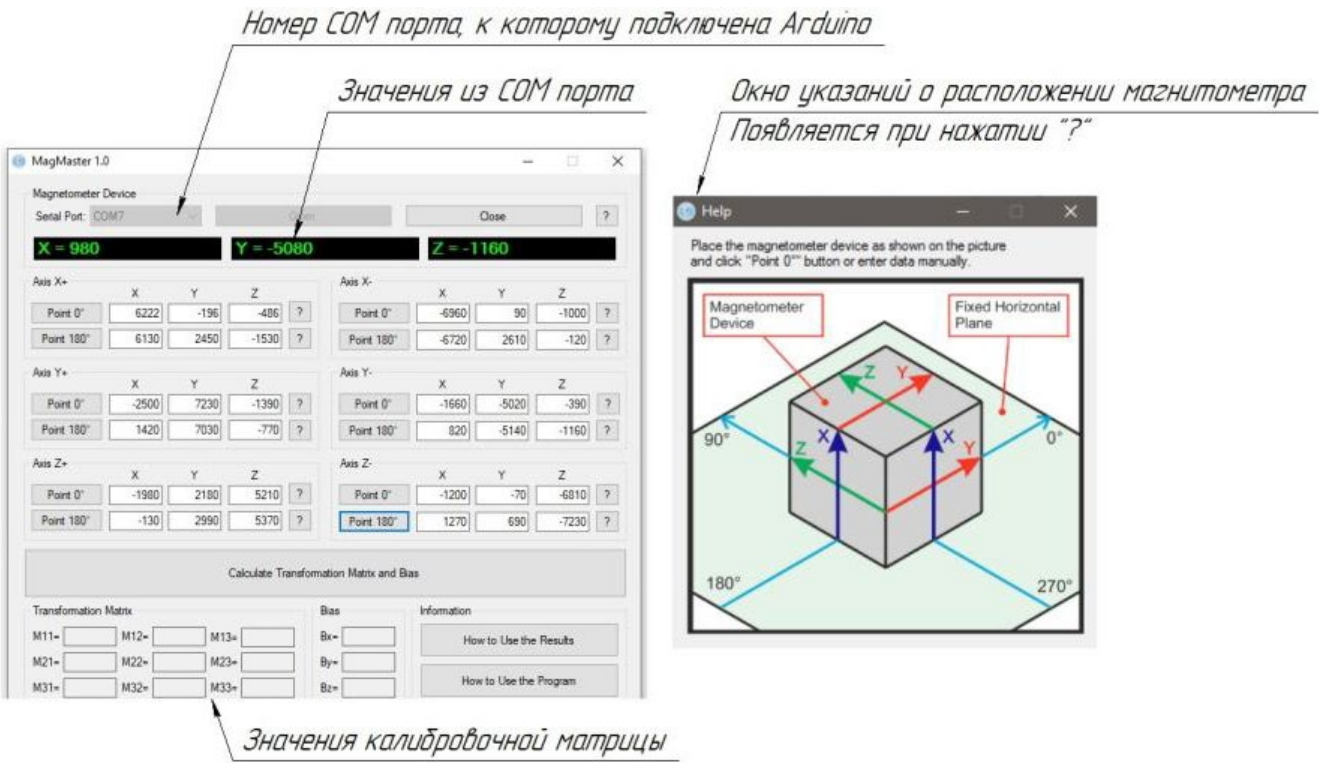


Рисунок 8 – Окно программы «MagMaster»

2.4 Энкодеры для измерения скорости и угла ветра

Для измерения характеристик ветра, а именно скорость и направления, было принято решение использовать оптические энкодеры.

Рассмотрим характеристики энкодера E40H12-1200-3-N-5 [11]:

- диапазон рабочих температур от минус 10 до плюс 70 ℃;
- PPR = 1200;
- выходные сигналы A, B, Z (рисунок 9);
- тип выхода NPN, открытый коллектор;

- питание однополярное от 4,75В до 5,25В;

- максимальный потребляемый ток: 80 мА;

- максимальный протекающий в цепи ток нагрузки: 30мА.

Наличие Z сигнала необходимо для привязки 0 отметки флюгера к корпусу метеостанции: при обороте вокруг своей оси и высоком уровне сигнала с вывода Z значение переменной счетчика будет сброшено программой на 0. При этом энкодер предполагается расположить в корпусе таким образом, чтобы он был со направлен с осью электронного компаса.

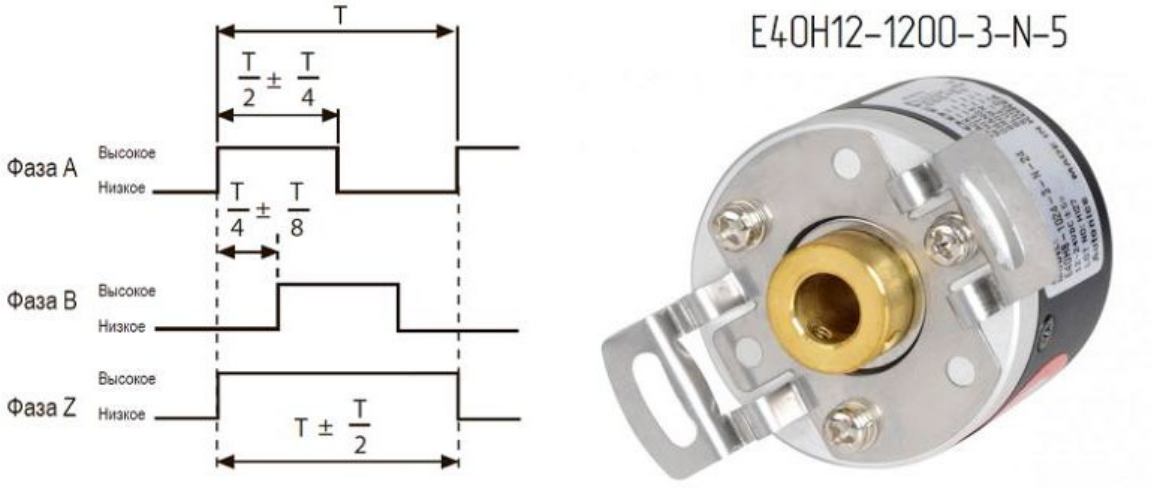


Рисунок 9 – Форма выходного сигнала и внешний вид энкодера E40H12-1200-3-N-5.

Если учитывать, что в артиллерии полная окружность 6000 тысячных, то не составит труда по формуле (1) подсчитать цену деления выбранного энкодера.

$$A_{\text{ц.д.энк}} = \frac{6000}{PPR} = \frac{6000}{1200} = 5 \text{ д.у} \quad (1)$$

где

$A_{\text{ц.д.энк}}$ – цена деления энкодера;

PPR – количество импульсов за оборот энкодера.

Примем точность измерения угла ветра цене деления ($A_{\text{ц.д.энк}} = \Delta A_{\text{энк.}}$). Перевод делений угломера в градусы осуществляется по формуле (2).

$$\alpha_{\text{град.}} = \alpha_{\text{д.у.}} \cdot 0{,}06 == 5 \cdot 0{,}06 = 0{,}3° \quad (2)$$

где

$\alpha_{\text{град.}}$ – угол в градусах;

$\alpha_{\text{д.у.}}$ – угол в делениях угломера.

Значение 5 д. у. в градусах будет равняться $0{,}3°$. В итоге на данном этапе мы можем посчитать и оценить точность измерения угла ветра, сложив точность электронного компаса и энкодера (формула (3)), а также сравнить с показателем ДМК и метеостанции Kestrel. Многое зависит от того, как программно организованно снятие данных и их обработка, но в данном случае мы приводим лишь аппаратную точность.

$$\Delta A_{\text{угл.ветр.}} = \Delta A_{\text{компаса}} + \Delta A_{\text{энк.}} = 2° + 0{,}3° = 2{,}3° \quad (3)$$

где

$\Delta A_{\text{ц.д.энк}}$ – точность энкодера;

$\Delta A_{\text{компаса}}$ – точность определения магнитного азимута HMC5883L;

$\Delta A_{\text{угл.ветр.}}$ – точность определение азимута магнитного ветра.

Точность снимаемых значений будет $\pm 2{,}3°$, что является более точным показателем по сравнению с ДМК и Kestrel.

Заметим, что диапазон рабочих температур в области ниже нуля ограничивается только -10℃, что является недостаточным для выполнения огневых задач в условии холодной зимы – данный энкодер был выбран по причине неимения другого аналога с более устойчивой к холоду системой. Но отметим, что принципы работы и обработки данных в любом случае аналогичны, поэтому для создания макетной версии достаточно имеющегося.

Обратим внимание на ток, протекающий в нагрузке: он не должен превышать 30 мА. Было принято решение произвести подтяжку всех выводов A, B и Z к плюсу питания резисторами 10 кОм, что соответствует протеканию тока 5 мА [1]. Схема выхода энкодера с открытым коллектором представлена на рисунке 10.

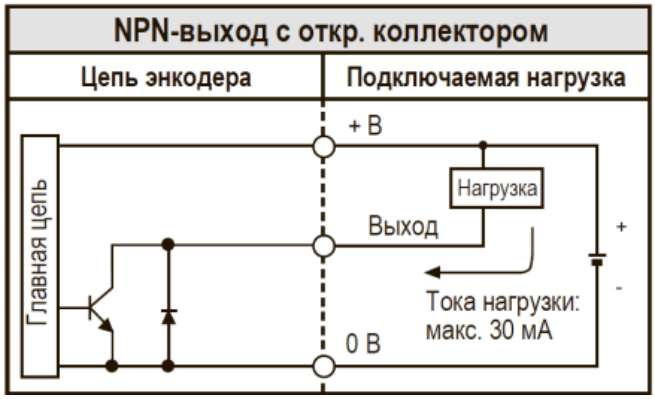


Рисунок 10 – Схема с открытым коллектором энкодера

Перейдем к рассмотрению энкодера, выполняющего функцию анемометра. Энкодер LPD3806-600BM для снятия скорости ветра имеет следующие характеристики [12]:

- диапазон рабочих температур от минус 10 ℃ до +70 ℃;
- PPR = 600;
- выходные сигналы A, B;
- тип выхода NPN, открытый коллектор;
- питание однополярное от 5 до 24 В;
- максимальный потребляемый ток 40 мА;
- максимальный протекающий в цепи ток нагрузки 30 мА.

Учитывая, что в документации не было написано о токе нагрузки, нами было принято решение выбрать подтягивающий резистор 10 кОм. В устройстве будет использоваться только A сигнал по спаду и по фронту, т.е. тем самым мы увеличиваем разрешение энкодера с 600 до 1200 PPR.

2.5 Дисплей LCD2004

Для отображения всей информации было принято решение использовать LCD дисплей с количеством ячеек 20 на 4 и с платой-переходником на базе PCF8574 [19] под шину I2C (рисунок 11).

Рисунок 11 – LCD2004 с платой-переходником под шину I2C

Для работы с дисплеем использовалась находящаяся в свободном доступе библиотека LiquidCrystal_I2C. В программе были задействованы следующие методы из библиотеки:

- print(data) – вывести (любой тип данных);
- setCursor(x, y) – курсор на (столбец, строка);
- clear()- очистить дисплей;
- home() – аналогично setCursor(0, 0);
- display() – включить отображение;
- blink() – мигать курсором на его текущей позиции;
- noBlink() – не мигать;
- cursor() – отобразить курсор;
- noCursor() – скрыть курсор;
- backlight() - включить подсветку;
- noBacklight() - выключить подсветку;
- createChar(uint8_t, uint8_t[]) - создать символ.

В перспективе развития проекта предусматривается переход на библиотеку «LCD_1602_RUS» для отображения русских слов, причем программа не претерпит кардинальных изменений, т.к. методы аналогичны, единственное будет необходимо переписать сам текст слов.

2.6 Модуль клавиатуры

Для взаимодействия с устройством было принято решение об установке клавиатуры 8 на 8 (рисунок 12), чтобы вводить различные данные, необходимые для расчета метеорологических данных.

Для взаимодействия с клавиатурой в программе применяется библиотека «Keypad.h». Из данной библиотеке задействуется конструктор при создании объекта, а также методы типа «get», для получения символа нажатой клавиши. Также, в данной библиотеке реализован случай залипания клавиши: если зажать одну клавишу и долго её держать, то символ вернется всего один раз.

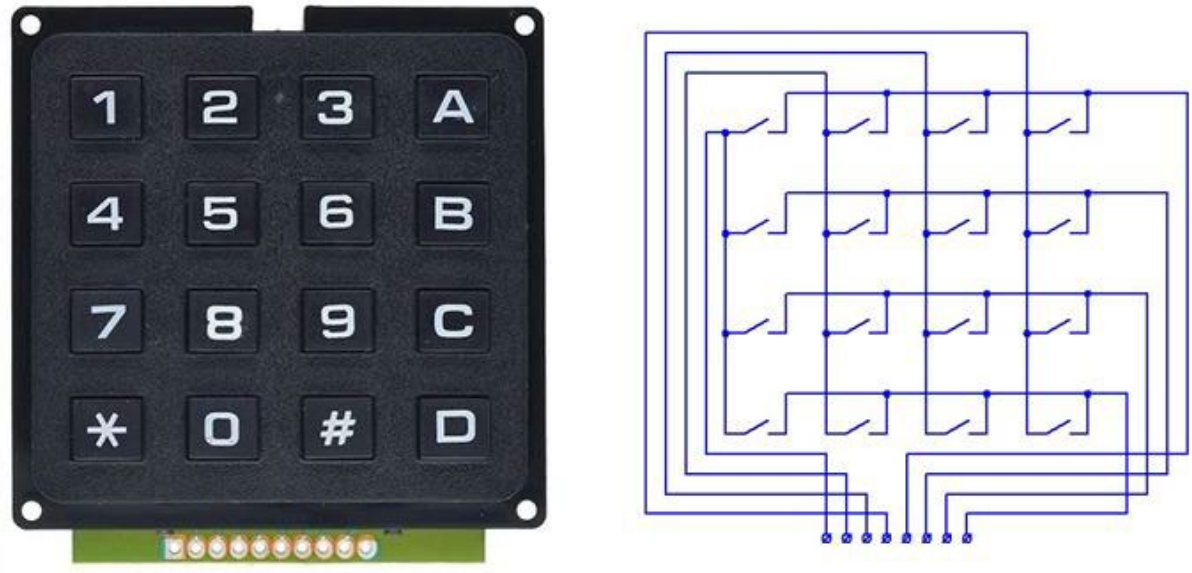


Рисунок 12 – Клавиатура 8 на 8 и её схема электрическая принципиальная

2.7 Модуль часов реального времени RTC DS3231

Одна из особенность артиллерийской метеостанции: приближенный бюллетень «Метеосредний» имеет срок годности 1 час, т.е. через час после того, как были получены метеоданные, СОБ не имеет права их использовать и «Метеосредний» необходимо обновить, причем дата и время отображаются в самом бюллетене в первой группе. В итоге, метеостанции нужен постоянный доступ к реальному времени. Микроконтроллер, используя собственный таймер, может сам производить подсчёт времени, но проблема заключается в необходимости постоянного ввода даты и времени при включении метеостанции, что будет занимать время и вносить свои неудобства – все сводится к установке отдельного модуля часов реального времени со своим независимым питанием.

Наш выбор в решении данного вопроса пал на RTC DS3231 – модуль часов реального времени, изображенный на рисунке 13. Необходимое напряжение питания от 2,3 В до 5.5 В, потребляемый ток при этом 180 мкА. Заметим, что у модуля имеется возможность считать часы, минут, секунды, число, месяц, год, день недели; при этом имеется встроенный календарь до 2100 года. Модуль работает по шине I2C [17].

Рисунок 13 – Модуль RTC DS3231

Ключевой особенностью является наличие энергонезависимости: модуль имеет свой автономный источник питания в виде батарейки CR2025, потребление тока от которой составляет 0.8 мкА.

Для работы с модулем часов было принято решение подключить свободно доступную в сети библиотеку «microDS3231». В библиотеки реализованы методы типа «get» и «set» для получения и установки времени соответственно.

2.8 Шина I2C

Протокол последовательной связи I2C – последовательная шина обмена данными между интегральными схемами.

В протоколе передача данных ведется по двум проводам: SDA (serial data line) и SCL (serial clock line). SDA используется для передачи данных, SCL для передачи сигнала синхронизации, генерируемого ведущим устройством. На плате Arduino Nano вывод SDA является пином A4, вывод SCL – A5.

Ключевые особенности протокола I2C [3]:

- скорость передачи порядка 100 кбит/с;
- реализована защита от помех, но отсутствует протокол обнаружения ошибок (что, например отличает данный протокол от UART);
- необходимо уделять особое внимание адресации устройств;
- используя две линии, можно подключать порядка 1000 устройств;
- протокол позволяет передавать данные на расстояния до 50 м.

Для работы с протоколом I2C используется библиотека «Wire.h» которая позволяет произвести инициализацию устройств с учетом их адресов, считать информацию или же отправить её конкретному устройству, находящемуся на шине.

2.9 Автономное питание устройства

Было принято решение в устройстве использовать автономный источник энергии Li-ion аккумулятор, ёмкостью 650 мАч. Обоснуем выбор данного вида аккумулирующего устройства.

Прежде всего li-ion аккумуляторы имеют компактные размеры и вместе с тем относительно большую ёмкость, при этом температурный диапазон работы в среднем составляет от минус 20℃ до плюс 60℃. Ёмкость 650 мАч была выбрана в соответствии с током потребления равным 110 мА, измеренным при работе всех элементов и модулей в штатном режиме. Измерение тока с использованием USB-тестера представлено на рисунке 14. Таким образом, схема при температуре 25℃ сможет непрерывно работать 5-6 часов.

Рисунок 14 – Измерение токового потребления собранной схемы

Зарядку аккумулятора предполагается производить от сети 5 В через разъём USB-C, доступно расположенного на корпусе. Питание с разъёма поступает на модуль зарядки и защиты аккумулятора, работающего на основе микросхемы TP4056 и DW01-P, расположенных на одной плате: данный модуль представлен на рисунке 14.

Рисунок 14 – Модуль зарядки и защиты аккумулятора

Изображение схемы модуля представлено на рисунке 15. На входе данного модуля возможно напряжение от 4 до 8 В, при этом ток зарядки аккумулятора ограничивается резистором R_3 на плате. С производства номинал резистора R_3= 1,2 кОм, что соответствует ограничению тока в 1А – для выбранного нами литий-ионного аккумулятора ёмкостью 650 мАч это небезопасно большое значение, поэтому в соответствии с таблицей 1 (взятая из документации [21]) нами был перепаян резистор R_3 на резистор номиналом 5,4 кОм, что соответствует ограничению тока в 250 мА (данное значение было взято как треть от ёмкости аккумулятора).

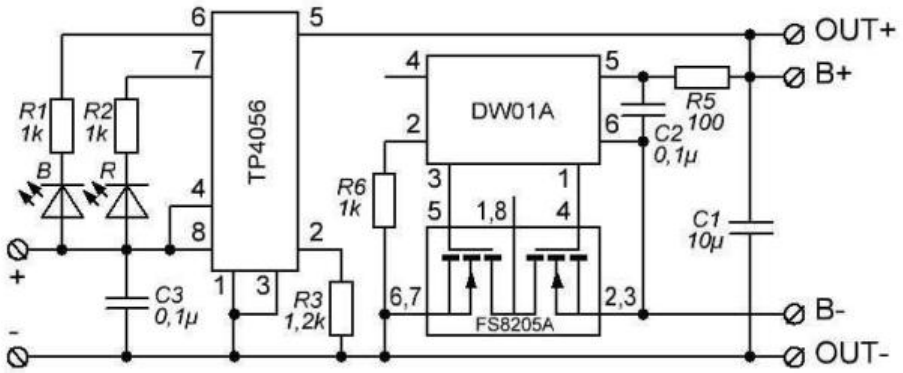


Рисунок 15 – Изображение схемы зарядки и защиты аккумулятора

Таблица 1 – Номиналы резистора R_3 и соответствующее значение тока нагрузки

R_3, кОм	I_{out}, мА
10	130
5	250
4	300
3	400
2	580
1,66	690
1,5	780
1,33	900
1,2	1000

Выводы «OUT» подключаются к повышающему модулю на микросхеме ME2108, представленного на рисунке 16.

Рисунок 16 – Изображение повышающего модуля на ME2108

Функция данного модуля состоит в том, чтобы повысить напряжение с 2,4 – 4,2 В до 5 В, при этом значение тока составляет порядка 400 мА [20] (запас в отношении потребления тока у разработанной нами схемы четырехкратный).

Таким образом, напряжение с аккумулятора, повышенное до 5 В, подается на микроконтроллер и происходит запитывание схемы.

Выводы по второму разделу

Мы привели характеристики всех выбранных модулей для создания прототипа метеостанции и далее, обозначив аппаратную часть, перешли к программированию Atmega328p.

3 Программирование устройства

Написание программы для устройство делалось посредством языка C++ в интегрированной среде разработки Arduino IDE, включающей в себя удобную среду написания кода, компилятор, компоновщик и программу для прошивки микроконтроллеров.

В программе устройства используется:

- определение препроцессора,
- сторонние библиотек,
- EEPROM и Flash память МК для хранения данных,
- глобальные функции и переменные,
- описание классов и объекты классов,
- функция входа в программу setup() и функция loop().

Итоговый код занимает 29504 байт памяти устройства (т.е. 96% доступной Flash памяти), при этом глобальные переменные занимают 1167 байт оперативной памяти устройства (56% доступной оперативной памяти).

Перейдем к рассмотрению кода программы.

3.1 Использование директивы препроцессора

Препроцессор запускается еще до того, как к работе подключится компилятор, на выходе которого имеется машинный код для последующей обработки компоновщиком. Препроцессор выполняет подстановку текста в зависимости от различных директив [8]. В нашей программе задействовались следующие ключевые слова для определения препроцессора: это «define» и «include». С помощью define мы объявляем текстовую замену, что удобно для подстановки каких-либо констант без необходимости создания переменной. С помощью include мы подключаем необходимые нам заголовочные файлы для работы с библиотеками. Часть кода с определение препроцессора представлено в листинге 1.

Листинг 1

```
#define DIAGNOSTIC_MOD 0
#define WE_HAVE_DS18B20 1
#define SAMPLE_INTERVAL_FOR_WINDSPEED_MILLISEC 1000
#define AMOUNT_OF_WINDSPEED_VALUES 3
#define SAMPLE_INTERVAL_FOR_WIND_ANGLE_MILLISEC 500
#define AMOUNT_OF_WIND_ANGLE_VALUES 3
#define TIMEOUT_FOR_ANEMOMETER 5000
#define TIMEOUT_FOR_NOTS 3000
#define SPEED_ENKODER_PPR 1200
#define RADIUS 0.1
#define WIND_KOEF 20
#define SPEED_ENCODER_PIN_A 2
#define DS18B20_SENSOR_PIN 6
#define DIRECTION_ENCODER_PIN_A 3
#define DIRECTION_ENCODER_PIN_B 4
#define DIRECTION_ENCODER_PIN_Z 5
#define DIRECTION_ENCODER_PPR 1200
#include “Wire.h”
```

Поясним некоторые замены в препроцессоре в листинге 1. DIAGNOSTIC_MOD 0 – данной заменой мы можем включить и выключить вывод значений, предусмотренных программой, в монитор порта. Реализовано это через ветвления, в которых прописаны взаимодействия с выводом в монитор порта. TIMEOUT_FOR_ANEMOMETER 5000 – задается константное значение ожидание работы анемометра. WIND_KOEF 20 – задается коэффициент для расчета скорости ветра. Отметим, что данная замена является калибровочной, т.е. в ходе валедации устройства можно изменять коэффициент для точного снятия скорости ветра. DS18B20_SENSOR_PIN 6 – прописываем этой заменой пин, к которому подключен датчик DS18B20. #include «Wire.h» – происходит подключение библиотеки для работы с протоколом I2C.

3.2 Функции прерывания

Рассмотрим функции, отвечающие за работу энкодеров. Функции прерывания срабатывают в любой момент, когда МК работает и выполняется программа – данный вид функций используется для обработки нерегулярных

событий в процессе работы основной программы [9]. Одно из ключевых применений – обработка работы энкодеров. Объявление функций прерывания представлено в листинге 2.

Листинг 2

```
volatile int windAngle;
const int increaseStep = (6000/DIRECTION_ENCODER_PPR);

void windAngleEncoder()
{
 if (digitalRead(DIRECTION_ENCODER_PIN_B))
 {
  windAngle-=increaseStep;
 }
 else
 {
  windAngle+=increaseStep;
 }

 if ((digitalRead(DIRECTION_ENCODER_PIN_Z))||(abs(windAngle)>=6000))
 {
  windAngle=0;
 }
}
volatile int counterForSpeed;

void windSpeedEncoder()
{
 ++counterForSpeed;
}
```

При объявлении глобальных переменных для работы функций прерывания использовалось ключевое слово volatile, которое связанно с особенностями работы компилятора – для работы с внешними прерываниями использование данного слова обязательно. Рассмотрим работу функций.

Первая функция прерывания срабатывает, когда сигнал с пина, к которому подключен вывод A сигнала энкодера направления ветра, сменится с низкого уровня на высокий (объявление прерывания будет в конструкторе класса направления ветра, поэтому подробнее будет рассмотрено далее). В теле функции происходит проверка пина, к которому подключен вывод энкодера с B сигналом (временная диаграмма сигналов представлена на

рисунке 8): если уровень высокий, то из переменной-счетчика windAngle вычитается шаг increaseStep, который связан с количеством PPR энкодера и полным оборотом окружности в артиллерии, равным 6000. Если же условие не выполняется, то происходит прибавление increaseStep к windAngle. Контроль нулевого значения происходит по пину, к которому подключен вывод с Z сигналом энкодера угла ветра – данный сигнал меняется с высокого на низкий при прохождении валом энкодера определенной точки, заранее определенной, поэтому энкодер в корпусе устройства располагается так, чтобы точка Z сигнала была соосна с 0 установкой флюгера. Таким образом, при срабатывании Z сигнала происходит обнуление переменной windAngle.

Следующая функция прерывания обслуживает работу энкодера анемометра и в ней только происходит инкремент переменой counterForSpeed, которая будет задействована в соответствующей функции расчета линейной скорости ветра.

3.3 Описание класса скорости ветра

Переходим к описанию классов [18]. Начнем с класса скорости ветра, описание которого представлено в листинге 3.

Листинг 3

```
class WindSpeed
{
  public:
  void CalculatWindSpeed();
  int GetSpeedValue();

  private:
  int speedValue;
};
WindSpeed WindSpd;
```

В классе имеется два модификатора доступа: public и private. В public секции у нас имеется метод CalculatWindSpeed(), который ничего не

возвращает, а также метод int GetSpeedValue(), который возвращает посчитанное значение скорости ветра. В private секции имеется только поле для хранения скорости ветра speedValue. После описания класса сразу создаем объект WindSpd, который дальше в коде и будем вызывать.

Остановимся подробнее на методе CalculatWindSpeed(), описанном вне класса (листинг 4).

Листинг 4

```
void WindSpeed::CalculatWindSpeed()
{
 windSpeedValueIsReady = false;

 attachInterrupt(digitalPinToInterrupt(SPEED_ENCODER_PIN_A), windSpeedEncoder,
       CHANGE);
 bool flagWatingForCounting= true;
 int sum=0;
 int valuesOfSpeed[AMOUNT_OF_WINDSPEED_VALUES];
 for (byte i = 0; i<AMOUNT_OF_WINDSPEED_VALUES;i++)
 {
  counterForSpeed=0;
  flagWatingForCounting= true;
  do
  {
   if (millis() - tmrForWindSpeed >= SAMPLE_INTERVAL_FOR_WINDSPEED_MILLISEC)
   {
    tmrForWindSpeed = millis();
       valuesOfSpeed[i]=round(float(((2*M_PI*(counterForSpeed/SPEED_ENKODER_PPR) *
       RADIUS)/(SAMPLE_INTERVAL_FOR_WINDSPEED_MILLISEC/1000))*WIND_K
OEF));
       flagWatingForCounting=false;
   };
  }
  while(flagWatingForCounting);
  sum+=valuesOfSpeed[i];
 };
 speedValue=round(float(sum/AMOUNT_OF_WINDSPEED_VALUES));
 detachInterrupt(digitalPinToInterrupt(SPEED_ENCODER_PIN_A));

 windSpeedValueIsReady = true;
}
```

В начале метода происходит подключение прерывания, которое по окончанию будет отключено – сделано это чтобы не загружать МК, т.к. прерывание необходимо только в рамках этого метода. Создается массив

снятых мгновенных скоростей ветра и переменная для их суммы для того, чтобы произвести расчет средней скорости, т.к. ветер обычно является порывистым, а нам необходимо именно среднее значение.

Далее переходим к циклу снятия мгновенных скоростей ветра. Здесь происходит задействование определения препроцессора: вместо AMOUNT_OF_WINDSPEED_VALUES для цикла будет подставлено число снимаемых значений, причем это же число и является размерностью массива мгновенных значений. Следующей строчкой мы обнуляем переменную-счетчик counterForSpeed и придаем значение логического нуля переменной flagWatingForCounting, и далее в цикле «do-whille» мы производим расчет мгновенной скорости ветра через количество времени, установленное в препроцессоре вместо SAMPLE_INTERVAL_FOR_WINDSPEED_MILLISEC. В программе расчет мгновенной линейной скорости ветра происходил по формуле (4), выведенной из связи линейной и угловой скорости.

$$v = w \cdot r = 2\pi\nu \cdot r = \frac{2\pi n}{t} \cdot r \cdot K_v \quad (4)$$

где

w – угловая частота, $\frac{\text{рад}}{\text{с}}$;

r – длина плеча анемометра, м;

ν – частота, Гц;

n – количество оборотов за время t;

t - время замера количества импульсов энкодера;

K_v – поправочный коэффициент.

После расчета i-ого значения мгновенной скорости опускаем флаг flagWatingForCounting, прибавляем полученное значение скорости в общую сумму и продолжаем снятие.

По итогу находим среднее арифметическое суммы всех полученных скоростей и заносим значение средней скорости в соответствующую переменную.

3.4 Описание класса направления ветра

Рассмотрим в листинге 5 описание класса, в котором прописаны поля и методы для определения и хранения данных о дирекционном угле ветра.

Листинг 5

```
class WindDirection
{
 public:
 WindDirection();

 void SetMagneticAdjustment(int gridMagneticAzimuthAdjustment);
 int GetMagneticAdjustment();

 int GetWindGridAngle();
 int ReadoutMagneticAzimuth();
 void CalculatWindGridAngle();

 void SetInfoUseMagnetometr(bool useMagneticAzimuth);
 bool GetInfoUseMagnetometr();

 private:
 int headingMil;
 int gridMagneticAzimuthAdjustment;
 int windGridAngle;
 bool useMagneticAzimuth;
};
WindDirection WindDir;
```

Рассмотрим находящийся в public секции конструктор класса WindDirection(), описание которого представлено в листинге 6. Конструктор класса вызывается перед созданием объекта класса и только один раз [4]. Конструктор класса в программе используется для подключения энкодера направления ветра (через функцию pinMode() подключаем выводы с B и Z

сигналами), а также, для объявления аппаратного прерывания по выводу энкодера с А сигналом.

Листинг 6

```
 WindDirection::WindDirection()
{
 scaler_flag = false;
 pinMode(DIRECTION_ENCODER_PIN_B, INPUT);
 pinMode(DIRECTION_ENCODER_PIN_Z, INPUT);
 attachInterrupt(digitalPinToInterrupt(DIRECTION_ENCODER_PIN_A), windAngleEncoder, RISING);
 gridMagneticAzimuthAdjustment=GetMagneticAdjustment();
}
```

Для объявления прерывания используется функция attachInterrupt(), в параметрах которой указываем: вывод, сигнал с которого будет триггерным; функцию, которая будет выполнятся при срабатывании прерывания; форма срабатывания (в нашем случае «RISING» указывает на то, что прерывание будет срабатывать только при смене уровня с низкого уровня на высокий). Также, в конструкторе задаются начальные значения для полей класса.

Перейдем к рассмотрению методов класса. Методы void SetMagneticAdjustment(int gridMagneticAzimuthAdjustment) и int GetMagneticAdjustment() прописаны для установки и получения данных поправки магнитного склонения из энергонезависимой памяти EEPROM – описание данных методов представлено в листинге 7. В методе set имеется ветвление с проверкой заносимого значения: поправка магнитного склонения должна быть от минус 6000 до плюс 6000 делений угломера (в программе все расчеты с углами ведутся в делениях угломера). Поэтому если пользователь ведет значение условно 8200, то программа не занесет это значение в нужную ячейку памяти. После проверки, используя стандартную библиотеку EEPROM.h, которая идёт в комплекте с ядром Arduino, производим запись параметра функции gridMagneticAzimuthAdjustment во второй байт памяти. Также, сразу заносим значение параметра в значение одноименного поля объекта, используя конструкцию «this->».

Листинг 7

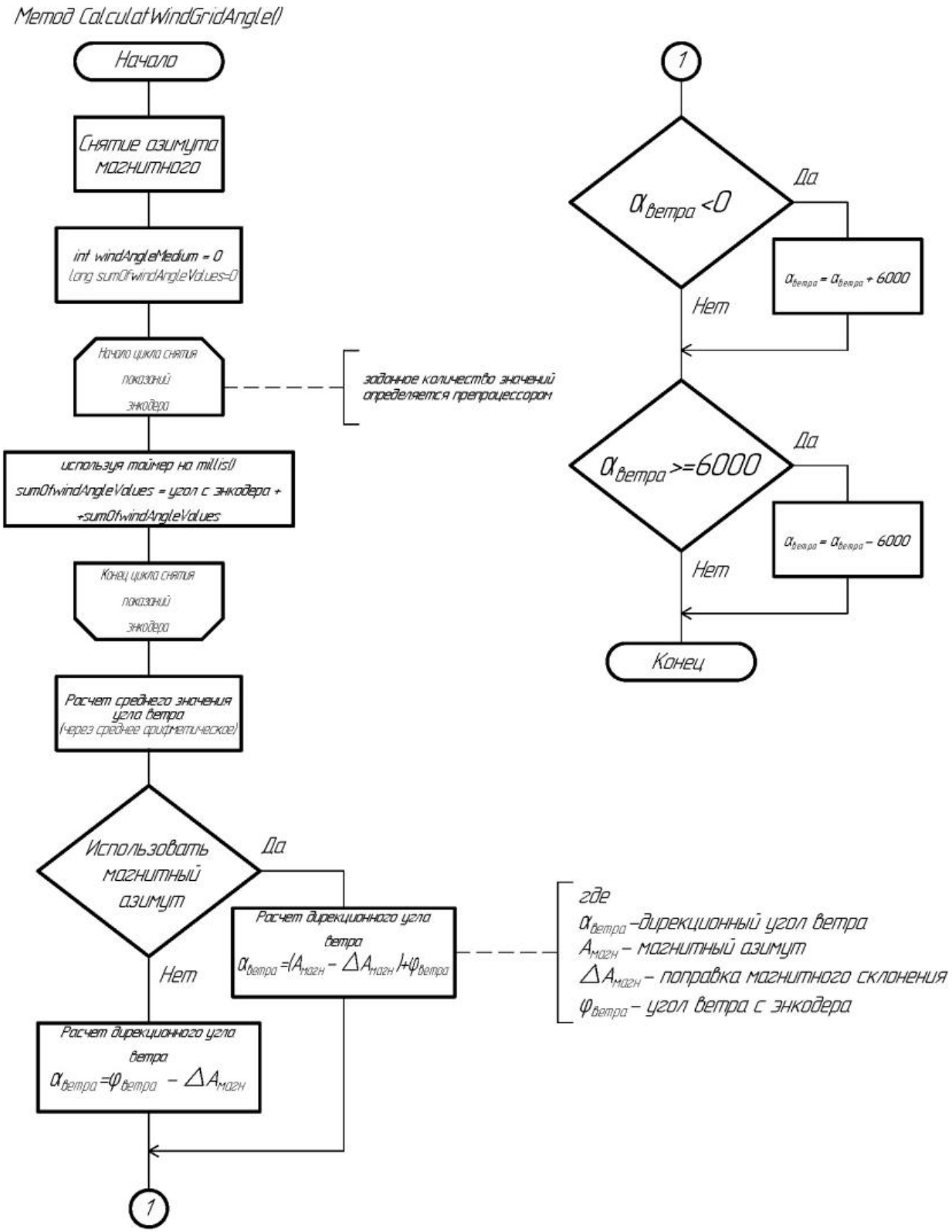
```
void WindDirection::SetMagneticAdjustment(int gridMagneticAzimuthAdjustment)
{
 if ((-6000<gridMagneticAzimuthAdjustment)&&(gridMagneticAzimuthAdjustment<6000))
 {
  EEPROM.put(2, gridMagneticAzimuthAdjustment);
  this-> gridMagneticAzimuthAdjustment= gridMagneticAzimuthAdjustment;
 }
}
int WindDirection:: GetMagneticAdjustment()
{
 EEPROM.get(2, this->gridMagneticAzimuthAdjustment);
 return gridMagneticAzimuthAdjustment;
}
```

В методе get тип возвращаемого значения int, и, используя EEPROM.get, считываем значение из второго байта и присваиваем его полю объекта, при этом возвращаем полученное значение.

На примере листинга 7 построены методы void SetInfoUseMagnetometr, и bool GetInfoUseMagnetometr() – использование энергонезависимой памяти приводит к сокращению времени приведения устройства в рабочее состояние снятия и расчета данных, т.к. все ранее введенные настройки сохраняются.

Перейдем к рассмотрению метода расчета дирекционного угла ветра void CalculatWindGridAngle(), блок схема-алгоритм которого изображена на рисунке 17.

Получение значения азимута магнитного включает в себя запрос данных по трем осям с электронного компаса, затем перерасчет данных с учетом калибровочной матрицы и далее расчет самого азимута магнитного в больших делениях угломера. Затем значение магнитного азимута передается в метод CalculatWindGridAngle().



Рисунке 17 – Блок схема-алгоритм метода CalculatWindGridAngle()

Далее в алгоритме предусмотрен цикл снятия определенного в препроцессоре количества значений угла ветра с энкодера флюгера через определенный в препроцессоре период времени – сделано это с целью

повышения точности снятия данных при порывистом ветре. Находится среднее значение угла ветра через среднее арифметическое нескольких измерений угла ветра.

Затем производим расчет дирекционного угла ветра, где из магнитного азимута вычитаем поправку магнитного склонения и прибавляем к полученному значению угол ветра.

После расчета дирекционного угла производится проверку на то, чтобы $\alpha_{\text{ветра}} \epsilon [0; 6000)$, используя ветвления.

3.5 Описание класса датчиков температуры и давления

Далее рассмотрим описание классов, связанных с измерением температуры и давления, представленное в листинге 8.

Листинг 8

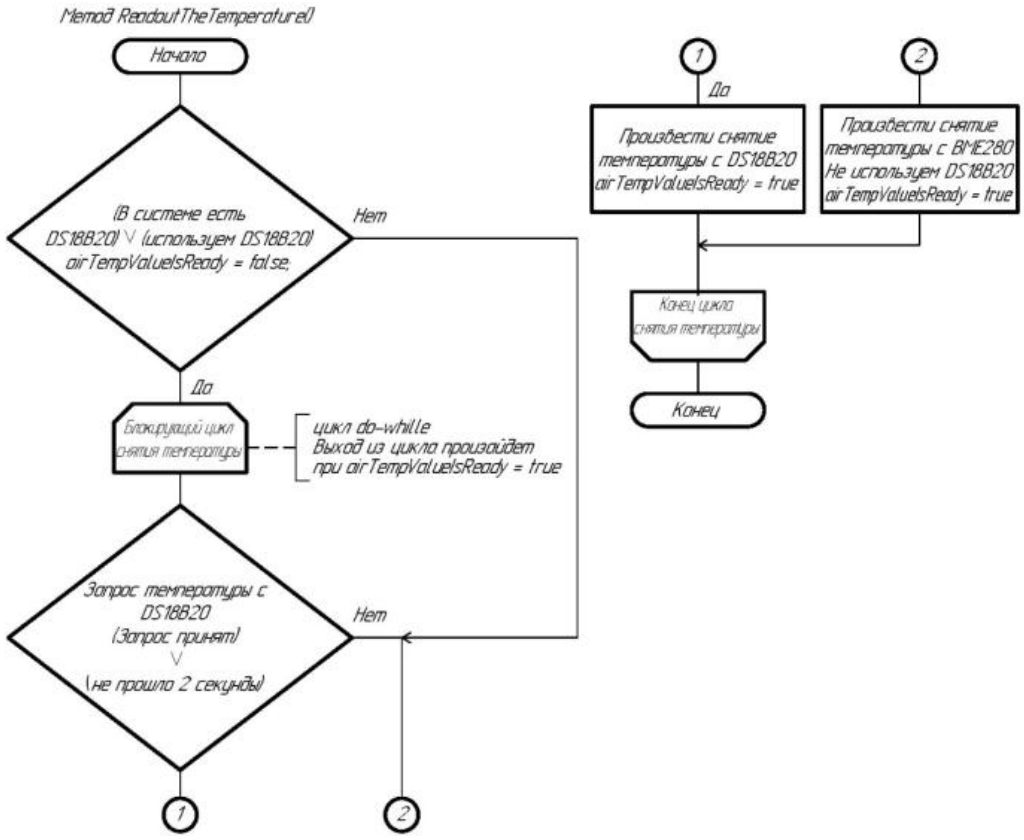
```
class Temperature
{
 public:
 Temperature();
 void ReadoutTheTemperature();
 int GetTemperture();

 private:
 int temperature;
 bool useDS18B20;
};
Temperature AirTemp;

class AirPressure
{
 public:
 void ReadoutThePressureBMP280();

 uint16_t GetPressure();

 private:
 uint16_t  pressure;
};
```

В классе температуры имеется свой конструктор, код которого представлен в листинге 9.

Листинг 9

```
Temperature::Temperature()
{
 if(WE_HAVE_DS18B20)
 {
  useDS18B20 = true;
 }
 else
 {
  useDS18B20=false;
 }
};
```

Предусмотрен вариант, что датчика температуры не будет в устройстве, и это задается в препроцессоре заменой WE_HAVE_DS18B20. Снятие температуры окружающей среды происходит либо с датчика DS18B20, либо с датчика атмосферного давления BME280. Рассмотрим алгоритм метода ReadoutTheTemperature(), представленный на рисунке 18.



Рисунке 18 – Блок схема-алгоритм метода ReadoutTheTemperature()

В программе также предусмотрен случай, если DS18B20 вышел из строя, и необходимо снятие температуры с BME280 (как описывалось в подразделе 2.2 у датчика BME280 также есть встроенный электронный термометр) – реализовано это через метод readTemp() библиотеки GyverBME280, который возвращает булево значение истины, если датчик получил запрос и готов отправить микроконтроллеру температуру. Заметим, что датчик посылает данные о готовности через 750 мс, таким образом, если через 750 мс после отправки запроса датчику не вернется значение истины в методе readTemp(), то можно сделать вывод, что датчик либо поврежден, либо произошел обрыв одного из поводов. Программно было прописано 2 секунды ожидания, и если за это время датчик DS18B20 не выходит на связь, то снятие температуры происходит с модуля BME280.

Для соблюдения принципа инкапсуляции в классе для атмосферного давления состоит из двух методов: запрос и снятие атмосферного давления с модуля BME280; получение в программе снятого значения.

3.6 Описание класса расчета метеосреднего

Приближенный бюллетень «Метеосредний» содержит средние значения метеорологических факторов в слоях до стандартной высоты 4000 м. Приближенный бюллетень «Метеосредний» передается в подразделения в виде цифровой радиограммы. Кроме того, в приближенном бюллетене не указываются средние отклонения плотности воздуха от табличных и высоты температурного и ветрового зондирования.

Для сокращения объема радиограммы с целью ускорения передачи бюллетеня используется специальный метеорологический код. Закодированные бюллетени состоят только из цифр. Цифры располагаются по группам таким образом, что значение каждой цифры определяется ее местом в группе и местом группы в бюллетене. Группа отделяется одна от другой знаком «тире» (раздел) [6].

В бюллетень записывают [6]:

- дату и время производства измерений метеорологическим постом;
- высоту метеорологического поста над уровнем моря (м);
- отклонение наземного давления атмосферы (мм рт. ст.) и отклонение наземной виртуальной температуры воздуха (℃) на уровне метеорологического поста;
- средние отклонения температуры воздуха, направление и скорость среднего ветра для стандартных слоев.

Исходными данными для составления приближенного бюллетеня «Метеосредний» являются:

- время производства метеорологических измерений;
- высота метеорологического поста.

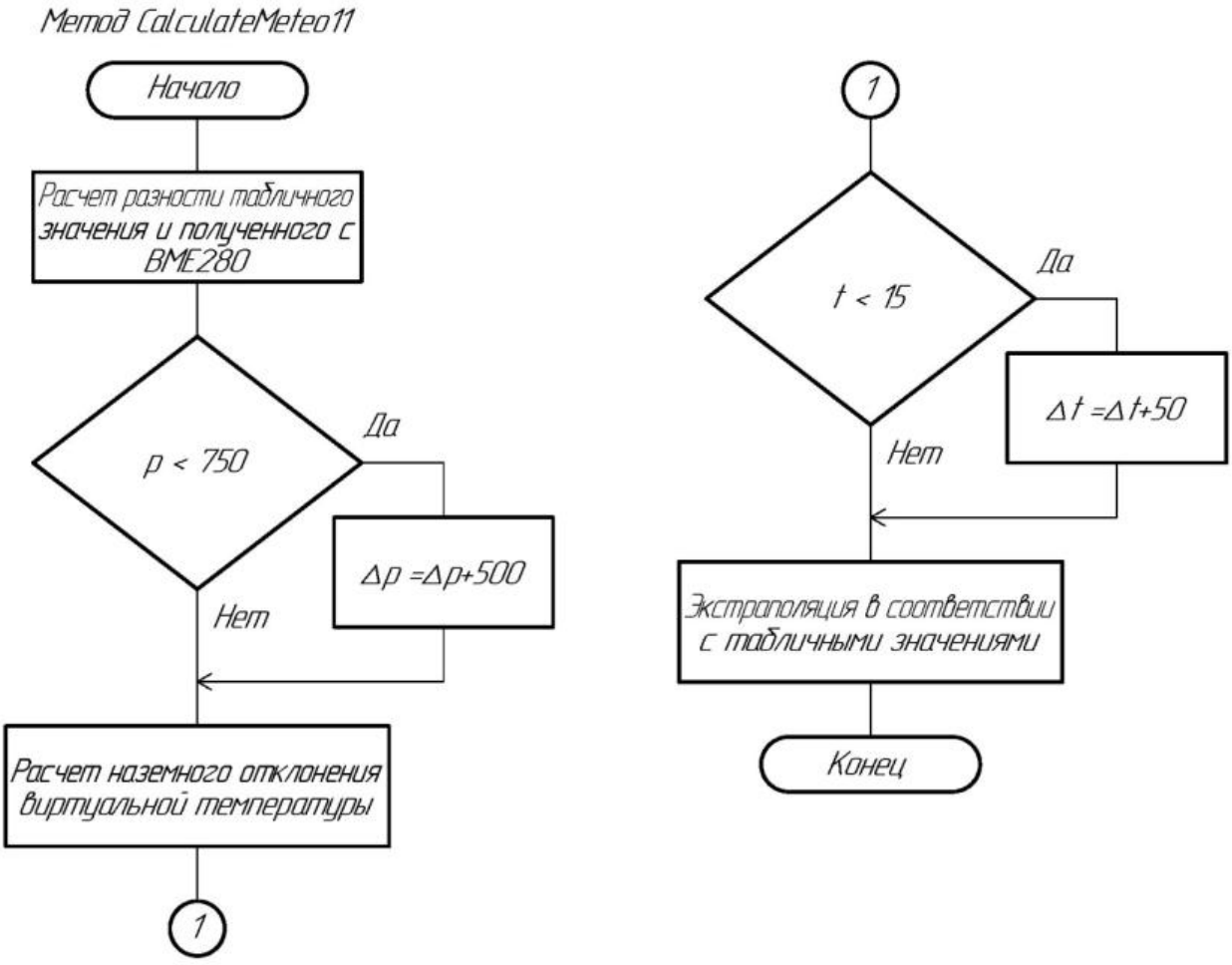
Пример приближенного бюллетеня «Метеосредний»:

«Метео 11 приближ. –08075–0120–50863–0203–623605–0403–613806–0802–614006–1202–604007–1602–594108–2002–584208–2401–584309–3001–574310–4001–564411–5000–554412–6000–544512–8000–544614–1051–534516–12–524618–14–514621–18–004723–22–014720–26–024718–30–034717–2627».

Расшифровка бюллетеня [6]:

- метеорологические данные сняты 8-го числа, в 7 ч 50 мин.
- высота расположения метеостанции над уровнем моря +120 м.
- наземное отклонение давления равно минус 8 мм рт. ст.
- наземное отклонение виртуальной температуры равно минус 13°С.
- в слое от поверхности земли до 200 м: среднее отклонение плотности воздуха +3%, среднее отклонение температуры воздуха минус 12°С, дирекционный угол направления среднего ветра 36-00, скорость среднего ветра 5 м/с.
- далее – по аналогии, данные для других стандартных высот бюллетеня.

С описанием класса метеосреднего можно ознакомится в приложении А в листинге всей программы. Рассмотрим основной метод класса void CalculateMeteo11(uint8_t dateOfMeteo, uint8_t hourOfMeteo, uint8_t wholeMinutesOfMeteo). В параметры класса входит дата, час и минуты составления метеосреднего. Рассмотрим на рисунке 19 блок схему-алгоритм метода расчета метеосреднего.



Рисунке 19 – Блок схема-алгоритм метода CalculateMeteo11

В основе блока с расчетом отклонения давления от табличного значения (табличной принята величина 750, именно от нее измеряют отклонение) лежит формула (5). Расчет отклонения виртуальной температуры основан на формуле (6). В бюллетени не передаются знаки «+» и «-», поэтому число отклонения со знаком плюс остается без изменений, а в случае отклонения с минусом к первой цифре прибавляют 5.

$$\Delta p = |750 - p| \quad (5)$$

где

Δp – отклонение значения атмосферного давления от табличной величины, мм. рт. ст.;

p – величина наземного атмосферного давления, мм. рт. ст.

$$\Delta t = |16 - t| \quad (6)$$

где

Δt – наземное отклонение виртуальной температуры, ℃;

t – величина наземной температуры, ℃.

Далее в программе происходит экстраполяция по высотам в соответствии с правилами стрельбы и управления огнем. Экстраполяция производится с использования таблиц, применяемых при составлении приближенного бюллетеня «Метеосредний». Пример такой таблицы представлен на рисунке 20 (часть данных скрыта). Заметим, что таблицы со значениями хранятся во Flash памяти микроконтроллера и обращение к данным происходит по ссылке.

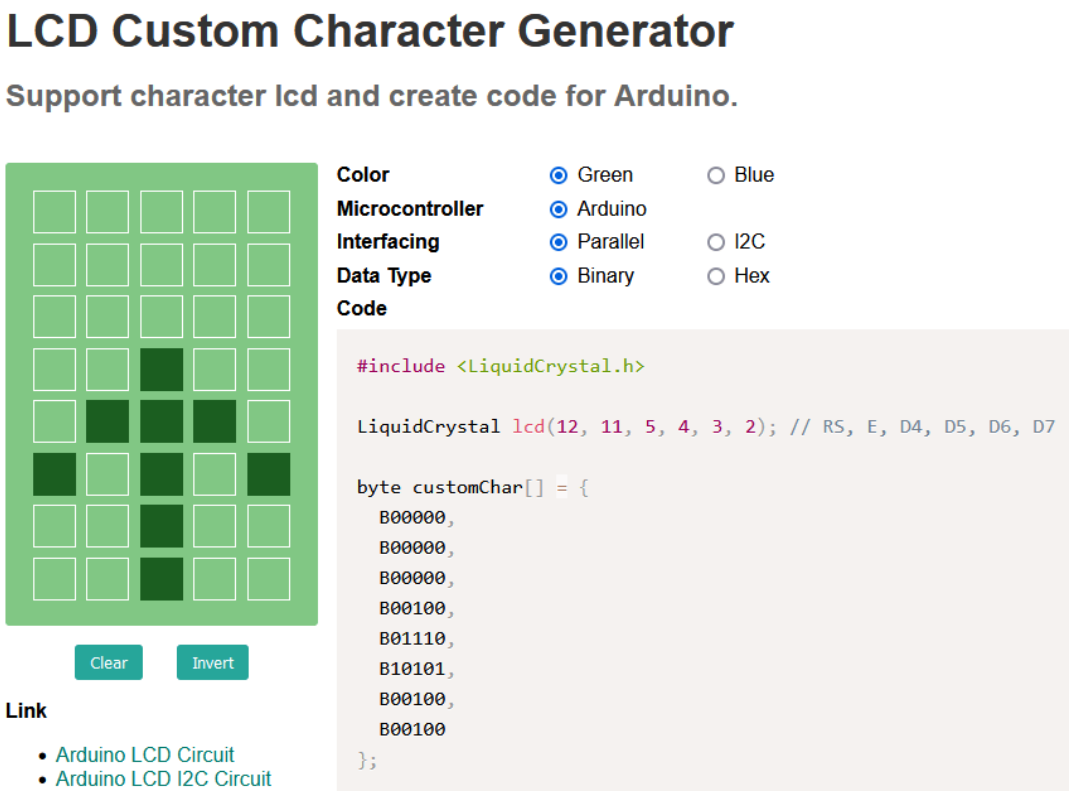
```
const PROGMEM uint8_t tempertureDeltaAtAltitudes[10][47]  =
{
        {1, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,   61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
    /*02*/{1,
    /*04*/{1,
    /*08*/{1,
    /*12*/{1,
    /*16*/{1,
    /*20*/{1,
    /*24*/{1,
    /*30*/{1,
    /*40*/{1,
};
```

Рисунке 20 – Объявление и инициализация массива данных для экстраполяции значений температуры для стандартных высот

3.7 Функция setup()

Рассмотрим функцию setup(), являющейся точкой входа в программу. В данной функции принято прописывать режимы пинов микроконтроллера, а также функции, которые должны сработать однократно в начале работы программы.

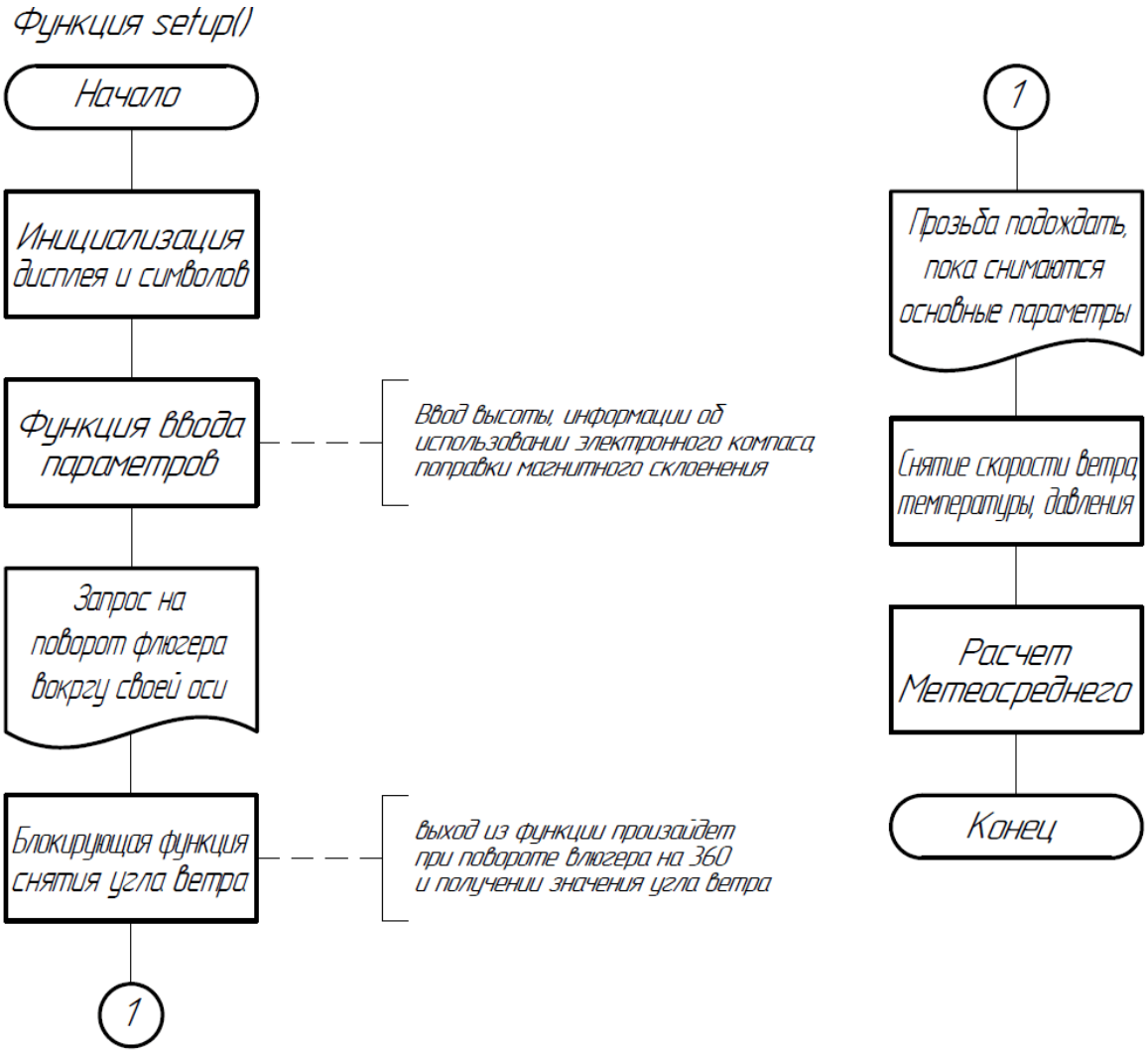
В рамках нашей программы объявления режимов работы пинов и придание начального значения переменным выведены в конструкторы классов. В функции setup() было принято решение произвести инициализацию LCD дисплея, также объявив 8 собственных символов. Пример создания символа и его представлено на рисунке 21.



Рисунке 21 – Создание символа стрелки для LCD модуля

Блок схема-алгоритм функции setup() представлена на рисунке 22. После инициализации дисплея и символов происходит процесс первоначального взаимодействия с пользователем: ввод необходимых данных для расчета

приближенного бюллетеня «Метеосредний» (или же пропуск ввода, т.к. параметры сохраняются в EEPROM, и пользователь перед изменением параметров видит раннее установленное значение), а также последующее снятие околоземных метеорологических параметров с последующей подготовкой приближенного бюллетеня.

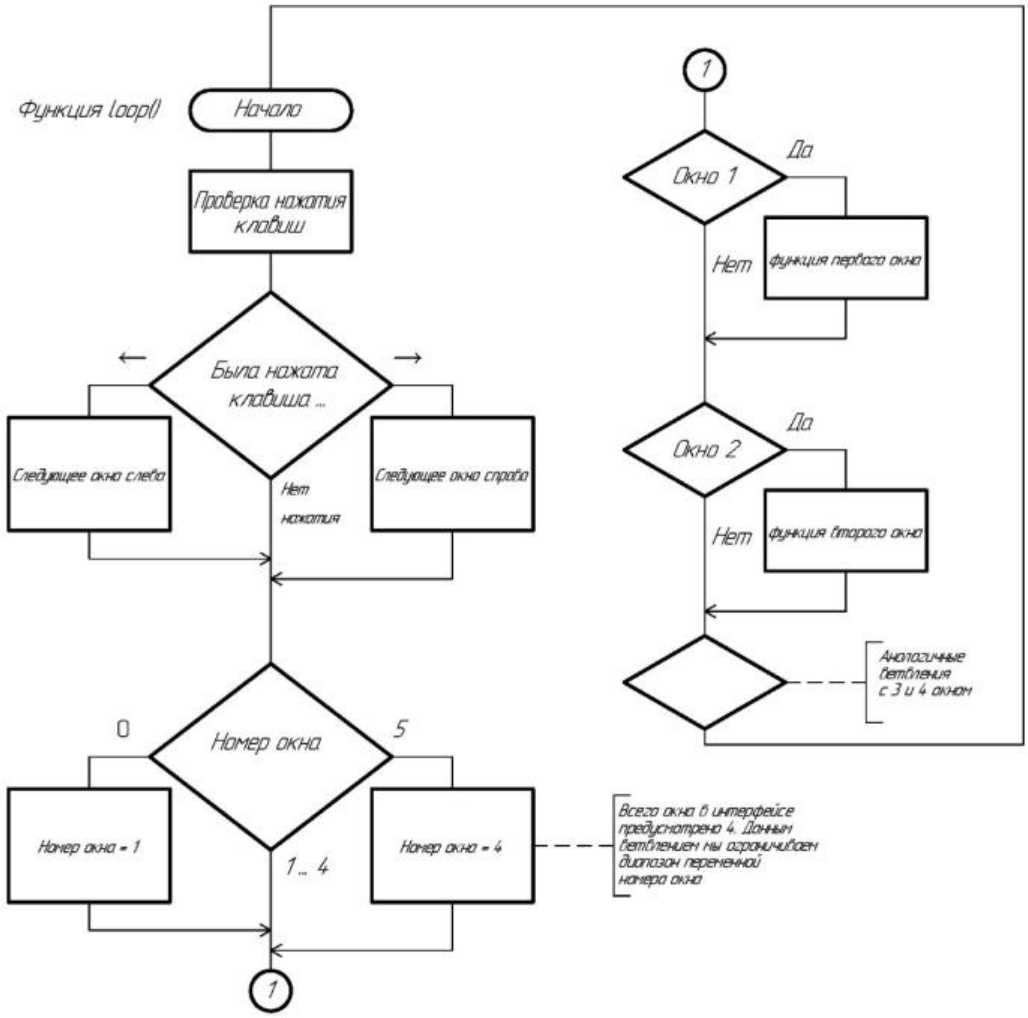


Рисунке 22 – Блок схема-алгоритм функции setup()

Таким образом функция setup() позволяет сразу же при запуске программы уточнить важные для дальнейшей работы данные, и затем произвести расчет всех данных.

3.8 Функция loop()

После функции setup() запускается цикл loop(), который продолжает свою работу в периода наличия питания у МК устройства. Блок схема-алгоритм представлена на рисунке 23.



Рисунке 23 – Блок схема-алгоритм функции loop()

В начале функции запускается функция проверки нажатой клавиши и ветвление, в котором происходит смена окна в соответствии с нажатой клавишей. Отвечающий за отображение в данный момент выбранного окна элемент является переменной, которая меняется инкрементом и декрементом:

отсюда следует, что необходимо произвести ограничение значения (в программе предусмотрены значения 1, 2, 3, 4 для соответствующих окон). Это ограничение реализовано через ветвление switch().

Далее по программе идут ветвления с выводом соответствующего окна в зависимости от значения переменной, изменяющейся в соответствии с нажимаемыми клавишами «влево» или «вправо» на клавиатуре – так реализована навигация по окнам.

Отображаемые окна рассмотрим в разделе 5.1, связанном с интерфейсом.

Выводы по третьему разделу

В данном разделе мы описали основные разделы программы, рассмотрев препроцессорную замену, подключение заголовочных файлов, функции прерывания, описание классов, а также основные функции setup() и loop(). С основной частью листинга программы можно ознакомится в приложении А. Код программы в приложении А не приводится полностью по причине потенциальной перспективности разработки в рамках военно-промышленного комплекса вооруженных сил российской федерации.

4 Технологическая часть

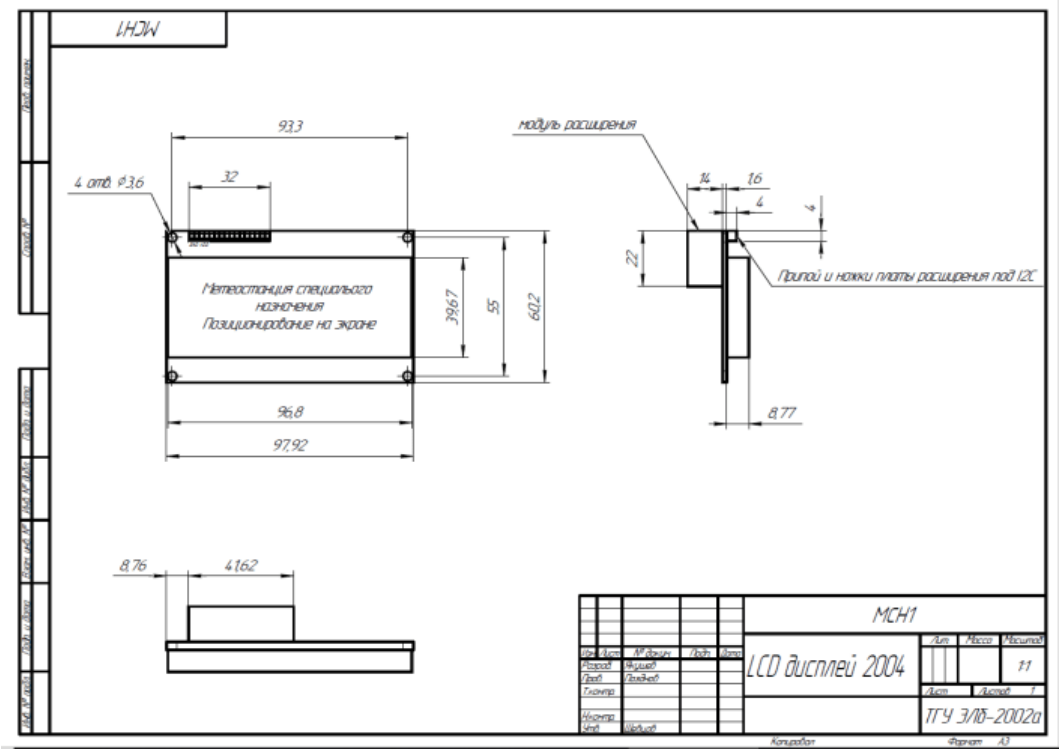
В технологической части рассмотрим процессы разработки корпуса, а также монтаж электронной части.

4.1 Проектирование корпуса

Проектирование корпуса происходило в САПР «КОМПАС 3D» версии 16 [2]. Процесс проектирования состоял из следующих этапов:

- измерение при помощи штангенциркуля всех модулей устройства;
- составление чертежей модулей;
- проектирование корпусных деталей в модуле «Деталь»;
- составление сборочной модели.

Чертежи в полной мере позволили произвести оценку габаритных размеров и подготовить посадочные места в корпусе устройства. Рассмотрим изображение чертежа LCD дисплея, изображенного на рисунке 23.



Рисунке 23 – Чертеж LCD модуля

На рисунке можем видеть основные габаритные размеры, а также технологические отверстия. Также, учитывалось место монтажа I2C переходника и сами габаритные размеры модуля. Подобный чертеж был сделан для клавиатуры, энкодеров.

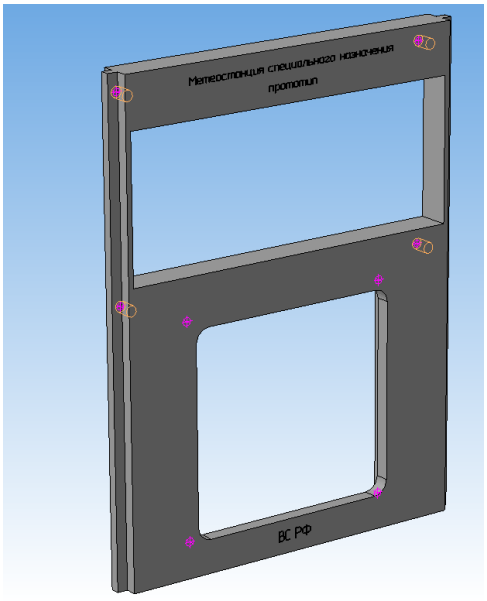
Условные изображения с размерами были сделаны для датчиков, для тумблера питания, а также для USB-C порта.

Следующим этапом являлось создание эскиза корпуса и последующее создание частей корпуса устройства.

Корпус был разделен на следующие элементы:

- лицевая часть,
- основание корпус,
- крышка корпуса,
- втулка флюгера,
- направляющая флюгер,
- вал анемометра,
- чаши анемометра.

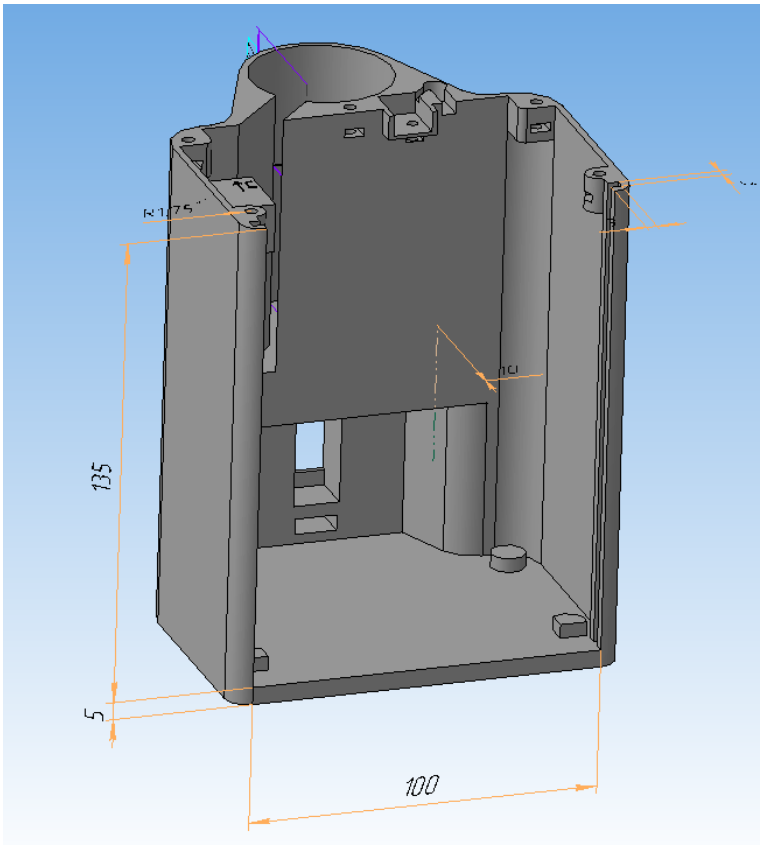
Модель лицевой части в окне программы «КОМПАС 3D» представлена на рисунке 24.



Рисунке 24 – Модель лицевой части в окне программы «КОМПАС 3D»

Лицевая честь включает в себя посадочные места под клавиатуру и под LCD дисплей. Также, на лицевой части имеются следующие надписи: название устройства («Метеостанция специального назначения»), функциональное предназначение («Прототип»), потенциальный заказчик («ВС РФ»).

Далее рассмотрим представленную на рисунке 25 модель основания корпуса – деталь, на которой размещены монтажная плата с МК, энкодеры, датчики, тумблер, аккумулятор и модуль зарядки. Минимальная толщина стенок данной детали составляет 2 мм. При создании корпуса учитывалось удобство хвата, а также практичность монтажа и демонтажа деталей (в том числе и в полевых условиях).

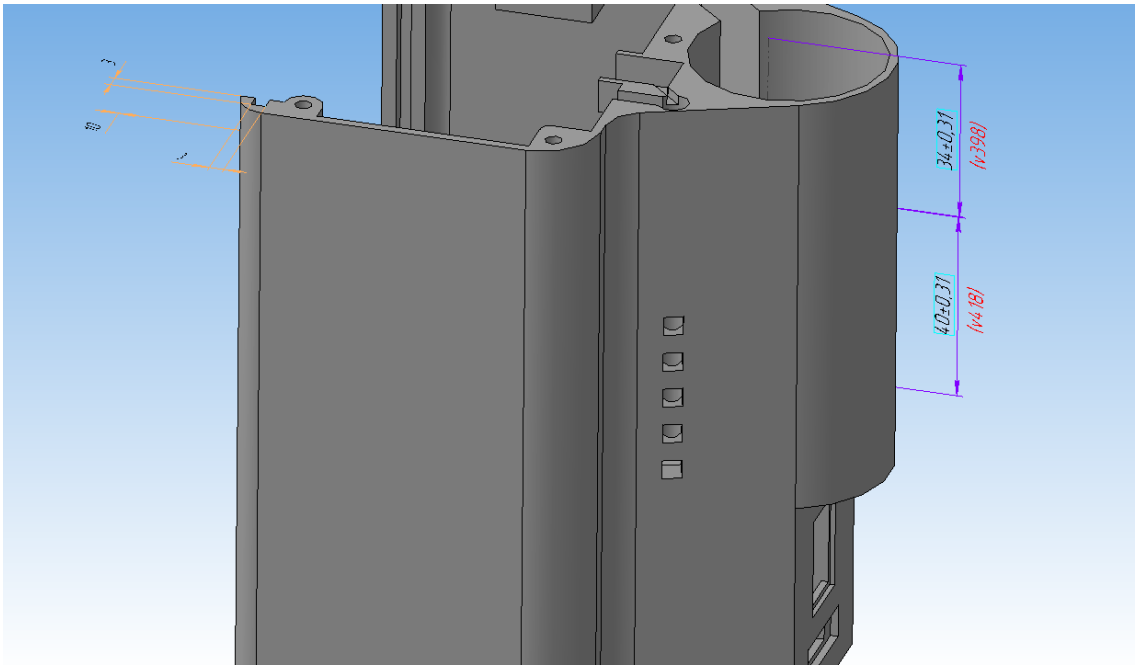


Рисунке 25 – Модель основания корпуса в окне программы «КОМПАС 3D»

Отметим особенности расположение датчиков. Электронный компас располагается на удалении от платы МК, цепи питания, информационных

цепей и цепи питания энкодеров, от элементов крепления корпуса и крышки: сделано это с целью уменьшения помех датчику. Также, на месте монтажа датчика расположена стрелка, указывающая направление нулевой отметки датчика.

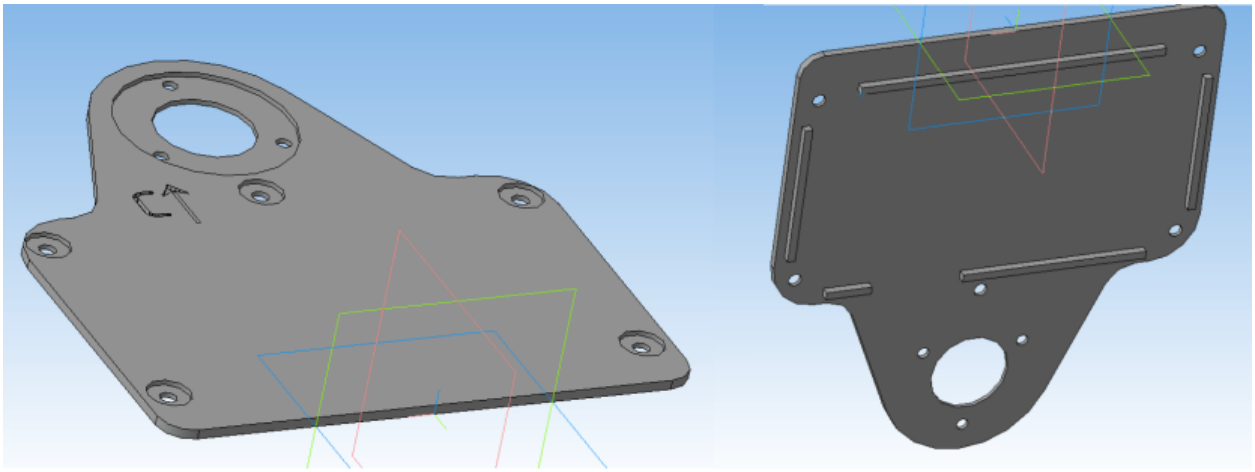
Датчик температуры располагается в корпусе таким образом, чтобы иметь доступ к окружающему воздуху для ускорения процесса приобретения датчиком температуры окружающей среды. Учитывалось устойчивость системы к попаданию влаги внутрь и предусматривались сливные наклоны, чтобы при дожде попавшая на датчик вода испарилась в окружающую среду без конденсации в корпусе. Отверстия под датчик температуры показаны на рисунке 26.



Рисунке 26 – Отверстия для работы датчика температуры DS18B20 в окне программы «КОМПАС 3D»

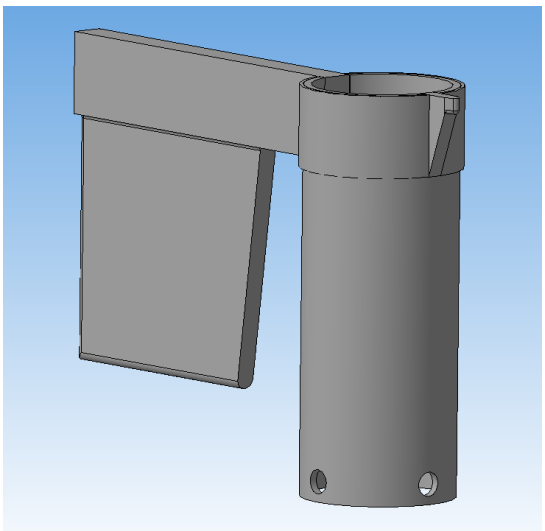
Рассмотрим деталь «крышка», модель которой можем наблюдать на рисунке 27. Крышка с основанием корпуса соединяется посредством винтов, гаек и шайб под М3. На крышке имеется 5 отверстий под винты с шайбами. Также, на крышке можем наблюдать 3 отверстия под энкодер флюгера: для

большей монолитности конструкции было принято решение использовать именно такой метод фиксации энкодера флюгера. Также важным является изображенная стрелка с буквой «С», означающей необходимую ориентировку устройства в пространстве относительно севера либо в случае внесения калибровок, либо в случае полевой работы в условиях магнитной аномалии. У крышки были предусмотрены пазы для плотной вставки корпус, и также для дополнительной пыле-влага защиты.



Рисунке 27 – Деталь «крышка» в окне программы «КОМПАС 3D»

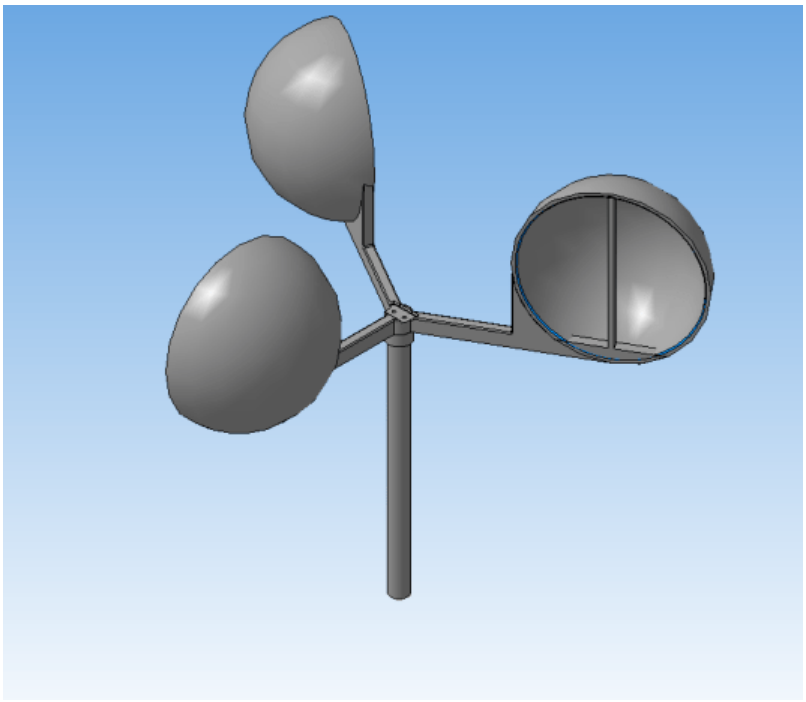
Далее перейдем к рассмотрению флюгера в сборе, модель которого приведено на рисунке 28.



Рисунке 28 – Деталь «флюгер в сборе» в окне программы «КОМПАС 3D»

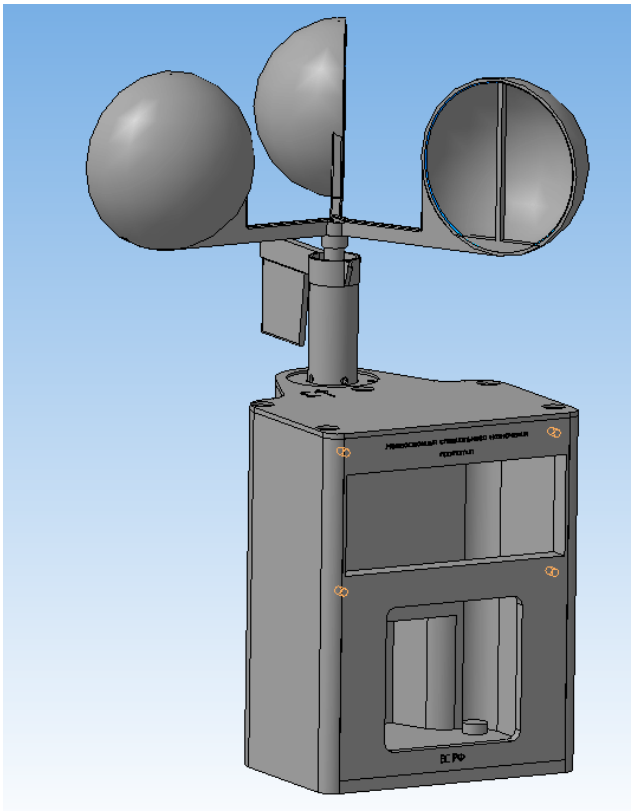
У флюгера имеется направляющая, которая снимается при транспортировке устройства, а также втулка, соединяющаяся с энкодером. У втулки предусмотрено два отверстия под крепление с энкодером под размер резьбы М3, причем отверстия сделаны таким образом, чтобы нулевая точка флюгера совпадала с нулевой точкой энкодера и корпуса (чтобы Z сигнал срабатывал именно при направлении флюгера на север при сориентированном на север корпусе устройства).

Рассмотрим на рисунке 29 модель анемометра в сборе, состоящую из чаш анемометра [7], а также вала, передающего от чаш крутящий момент на энкодер. Для укрепления конструкции было пронято решение сделать достаточно толстыми чаши и плечи анемометра. Соединение между чашами и валом происходит посредством двух саморезов.



Рисунке 29 – Анемометр в сборе в окне программы «КОМПАС 3D»

На рисунке 30 представлена модель всего устройства в сборе. Как можем видеть, все компоненты сходятся и нависших лишних деталей не имеется.



Рисунке 30 – Модель корпуса устройства в сборе в окне «КОМПАС 3D»

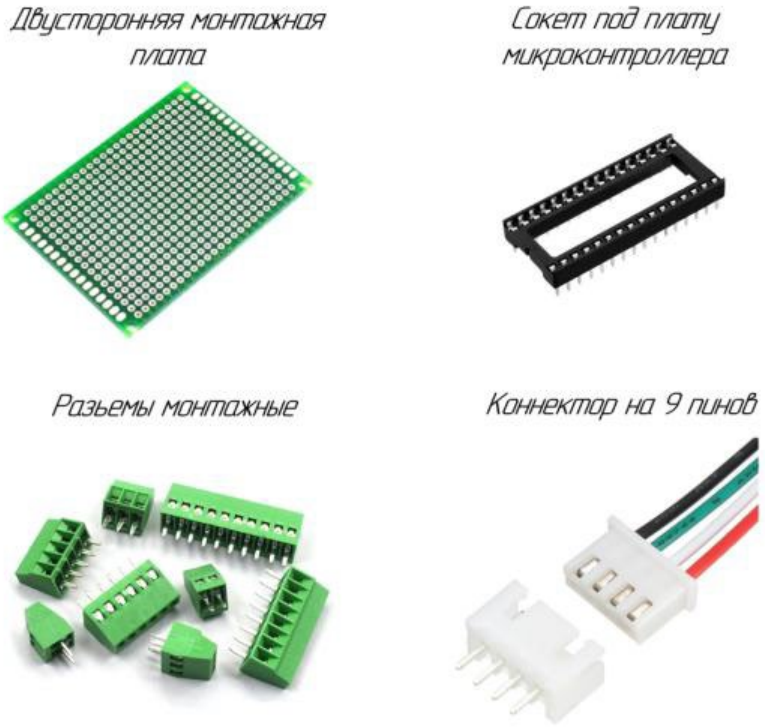
Модель спроектирована с опорой на особенности технологического процесса 3Д печати. С габаритными размеры можно ознакомится в приложении Б.

4.2 Монтажные работы

В данный раздел входят работы по монтажу компонентов на макетную плату, а также соединение модулей с МК.

Для основы электрической схемы была выбрана монтажная двусторонняя плата с предусмотренными отверстиями под крепление к корпусу. Также, для возможной быстрой замены платы микроконтроллера было принято решение использовать сокет, подходящий под типоразмер платы Arduino. Для реализации системы быстрой смены элементов нами было предусмотрена установка соединения с датчиками и другими модулями

посредством конвекторов и разъёмов. Все используемые для монтажа детали представлены на рисунке 31.



Рисунке 31 – Используемые в устройстве монтажные элементы

Соединениями на макетной плате служат медные многожильные провода относительно малого сечения, при этом для цепи питания используется цельный медный провод, являющийся шиной питания.

Монтаж происходил путем спайки элементов в соответствии схемой электрической-соединений, представленных в приложении В

Выводы по четвертому разделу

В данном разделе мы рассмотрели основные спроектированные модели корпуса, ознакомились с итоговой получившейся конструктивной сборкой и привели основную информацию о монтаже электрической части устройства.

5 Экспериментальное исследование системы

Затронем работу устройства при взаимодействии с пользователем, при снятии метеоданных, а также при выводе данных на экран.

5.1 Интерфейс устройства

Рассмотрим последовательность выводимых данных при запуске устройства. Прежде всего выводится запрос на ввод высоты. Выводимый текст представлен на рисунке 32.

Рисунке 32 – Выводимый на экран текст при запросе ввода высоты

Пользователь может либо, нажав клавишу «←», стереть высоту и ввести её заново (ограничение составляет четырехзначное число), либо же нажать кнопку «CANCEL» и оставить ранее введенное значение, занесенное в EEPROM. Заметим, что изначальное значение, выведенное на экран взято из EEPROM. Сохранение и изменение данных произойдёт только при нажатии клавиши «OK». Информация о высоте метеопоста известна старшему офицеру батареи на основе имеющейся карты, т.к. без данной информации невозможно топогеодезическая привязка батареи и в целом выполнение огневых задач.

В следующем окне программа запросит от пользователя выбрать: будет ли использоваться встроенный электронный компас для определения азимута

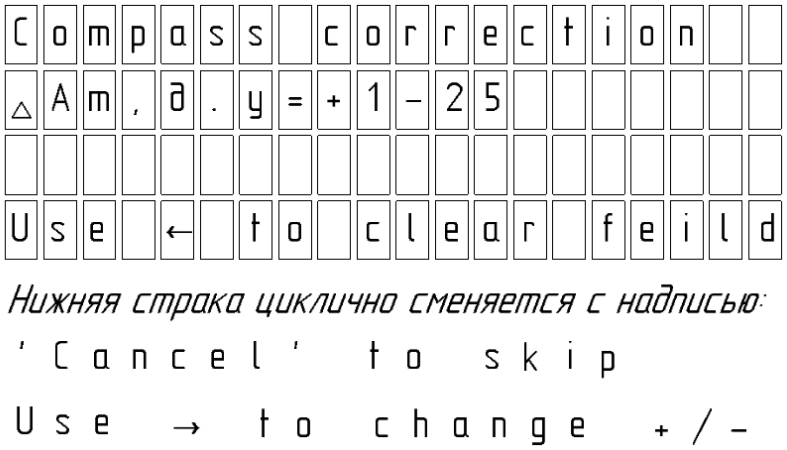
магнитного среднего для ветра, или нет (выбранный пункт также сохраняется EEPROM). Выводимый текст представлен на рисунке 33.

Рисунке 33 – Выводимый на экран текст при запросе использования встроенного электронного компаса

Данный запрос и в целом выбор работы реализован по двум причинам. Прежде всего возможна работа подразделений в зоне магнитных аномалий, что не позволит использовать электронный компас. Возможен вариант, что электронный компас вышел из строя, и его показания необходимо не учитывать. Для этого режима на крышке корпуса специально отмечено, как необходимо ориентировать устройство относительно севера (рисунок 26).

Включение компаса, в свою очередь, позволяет гораздо быстрее и точнее получать значение угла ветра. Относительно скорости снятия данных: нет необходимости определять самому на местности северное направление, устройство делает это автономно и причем с достаточно большой точностью, если сравнивать с ориентировкой вручную по заранее известному расположению севера.

Далее происходит запрос на ввод поправки магнитного склонения со своим знаком. Ранее вводимое значение берется из EEPROM. Выводимый текст представлен на рисунке 34.



Рисунке 34 – Выводимый на экран текст при запросе ввода поправки магнитного склонения для определения дирекционного угла ветра

Информация о поправке магнитного склонения также доступна старшему офицеру батареи, только в виде поправки буссоли (тоже неотъемлемая информация для топогеодезической привязки батареи) – в целом можно сказать, что данный параметр вводится один раз в определенной зоне боевых действий и больше не изменяется. Ввод величины происходит в делениях угломера, т.е. может быть от 0 до 5999 как с плюсом, так и с минусом.

Можем заметить, что во всех запросах нижняя строка является справочной: через каждые 2 секунды (задаваемый промежуток) текст, содержащий себе напоминание о функциональных клавишах в данном окне, сменяется.

Далее программа выводит текст, что необходимо провернуть флюгер и затем, после данного действия, выводится надпись, что производится снятие данные и необходимо подождать.

Затем нам открывается окно с измеренными данными – пример выводимых данных представлен на рисунке 35.

В данном окне отображены значения температуры, атмосферного давления, скорости ветра и дирекционного угла ветра в больших делениях угломера. Справа сверху расположена дата, также под датой высвечиваются два символа: один означает, что необходимо сориентировать устройство в

северное направление (при выключенном электронном компасе); и ещё один моргающий символ в виде буквы «М» означает, что метеосредний просрочен и необходимо его обновить. Переход между окнами осуществляется клавишами «←» и «→».

Рисунке 35 – Окно с измеренными значениями

Слева от первого открывшегося окна расположено окно меню, представленное на рисунке 36.

Рисунке 36 – Окно меню настроек устройства

В данном меню можно обновить метеосредний, изменить время обновления метеосреднего, а также изменить все те параметры, которые вводились в начале.

Справа от первого открывшегося окна расположены окна со значениями рассчитанного бюллетеня. Текст выводится в соответствии с нормами, принятыми в правилах стрельбы и управления огнем, т.е. в бюллетене присутствуют разделы и высоты. Вариант выводимого текста представлен на рисунке 37.

```
Meteo11 appr
16170-0250-50511-
02-111612-04-111714-
08-111815-12-111816-
```

Рисунке 37 – Окно с выводом приближенного бюллетень «Метеосредний»

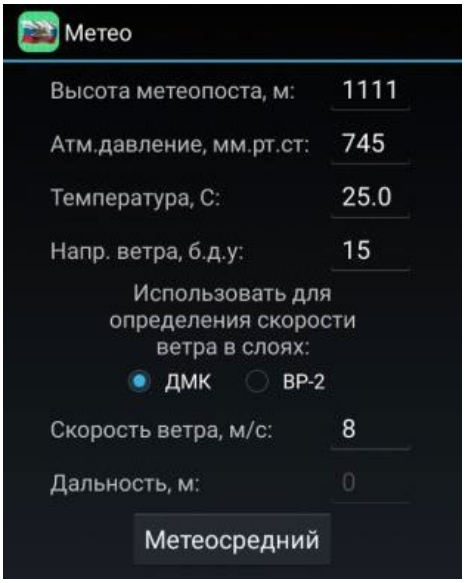
5.2 Проверка корректности выводимых данных и отладка программы

В данном подразделе произведём сравнение рассчитанного устройством приближенного бюллетеня «Метеосредний» с посчитанным бюллетенем в программе «АртГруппа».

На рисунке 38 представлены измеренные значения, а также значение бюллетеня, сформированные программой устройства.

Рисунке 38 – Измеренные устройством метеоусловия и часть сформированного на их основе приближенного бюллетеня «Метеосредний»

Открываем программу на смартфон «АртГруппа» и затем входим во вкладку «Метео», вводим снятые метеоданные, а также значение высоты, и далее нажимаем кнопку «Метеосредний». Проделанная процедура представлена на рисунке 39.



Рисунке 39 – Ввод данных в программу «АртГруппа» для расчета приближенного бюллетеня «Метеосредний»

В итоге мы получаем рассчитанный бюллетень, часть которого представлена на рисунке 40.

Рисунке 40 –Посчитанный бюллетень «Метеосредний» в программу «АртГруппа»

Как видим, устройство производит расчет бюллетеня верно.

Также, на этапе испытаний, программное обеспечение показало себя должным образом: неполадки и сбои не были замечены, программа работает исправна.

Для проверки истинности снимаемых метеопараметров (в том числе и для внесения калибровочных параметров) необходимо иметь соответствующее оборудование, и данная процедура уже имеет уровень валидации программного обеспечения и последующей полной сертификации устройства для использования в вооруженных силах, поэтому данный вопрос требует дополнительного исследования, невыполнимого в рамках данной работы при ограниченности ресурсов (поверенных приборов и времени).

Заключение

В рамках данной выпускной квалификационной работы было разработана портативная метеостанция, способная в короткие сроки и с достаточной точностью снимать метеоданные и рассчитывать на их основе приближенный бюллетень «Метеосредний», для обеспечения метеорологической подготовки подразделений РВиА и для успешного выполнения огневых задач.

В основе разработанного устройства лежит плата Arduino Nano с МК ATmega328p. Устройство в себе объединяет датчики, оптические энкодеры, а также модули ввода вывода.

Из особенностей устройства можно выделить наличие наглядного и интуитивно понятного интерфейса для взаимодействия с пользователем, компактный и в тоже время разборный корпус с электронной частью, предусматривающей работу и ремонт в полевых условиях. Корпусные детали разрабатывались под 3Д печать, что облегчает ремонт устройства.

В перспективах развития проекта можно выделить модернизацию электронной и программной части устройства для отправки метеоданных прямиком в программы расчета поправок для стрельбы на смартфон, с целью ещё большей экономии времени. Также, актуальным может быть модернизация корпуса устройства и придание большей жесткости корпусным деталям для повышения ударопрочности системы. Не последнее место в модернизации устройства также занимает повышение эргономики и разборности устройства.

Таким образом, данная разработка является одной из тех, которые позволят сохранить жизни военнослужащих вооруженных сил Росси в работе по защите интересов нашего государства.

Список используемых источников

1. Забродин Ю. С. Промышленная электроника. 2-е изд. М. : ООО ИД "Альянс", 2019. 419 с.

2. Зиновьв Д.В. Основы проектирования в КОМПАС-3D V16. 1-е изд. Студия Vertex, 2017. 327 с.

3. Катукия Г.Т., Дорофеев А.В., Линтварев Д.Э. АНАЛИЗ И СРАВНЕНИЕ ИНТЕРФЕЙСОВ ПЕРЕДАЧИ ДАННЫХ НА МИКРОКОНТРЕЛЛЕРАХ // Политехнический молодежный журнал. 2020. №07. С. 1-14.

4. Лафоре Р. Объектно-ориентированное программирование в С++. 4-е изд. СПБ. : Питер, 2022. 928 с.

5. Литвиненко В.И., Цеханович Д.Б. ТРЕБУЮТ РАЗРЕШЕНИЯ // АРМЕЙСКИЙ СБОРНИК. 2022. №9

6. Правила стрельбы и управления огнем артиллерии (ПСиУО-2020). Дивизион, батарея, взвод, орудие. Часть 1 : Введены в действие приказом главнокомандующего Сухопутными войсками от 24 декабря 2020 г. №322дсп – Москва : Военное издательство, 2020 г.

7. Протопопов Н.Г. Некоторые вопросы теории и расчета винтовых ветрочувствительных элементов // Метеорологические приборы и автоматизация метеорологических измерений. Ленинград : ГИДРОМЕТЕОРОЛОГИЧЕСКОЕ ИЗДАТЕЛЬСТВО, 1966. С. 3-32.

8. Рао Сиддхартха Освой самостоятельно С++ за 21 день. 7-е изд. М. : Вильямс, 2016 . 688 с.

9. Ревич Ю. В. Программирование микроконтроллеров AVR: от Arduino к ассемблеру. - СПБ.: БХВ-Петербург, 2020. - 488 с.

10. Сильников М.В., Баканеев С.А., Карпович А.В., Орлов С.А., Чернышев Ю.М. Курс артиллерии для оператора комплекса воздушной разведки. - СПБ. : Первый ИПХ, 2022. 364 с.

11. Энкодер E40HB12-1200-3-N-5 [Электронный ресурс] : РусАвтоматизация.

12. Энкодер LPD3806-600BM-G5-24C, [Электронный ресурс] : Roboparts URL: https://roboparts.ru/catalog/moduli_ustroystv/moduli_ustroystv_1/lpd3806-600bm-g5-24c/ (дата обращения: 27.04.2024).

13. ATmega328/P [Электронный ресурс] : DATASHEET COMPLET. URL: https://cdn.sparkfun.com/assets/c/a/8/e/4/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf (дата обращения: 27.04.2024).

14. BME280 Combined humidity and pressure sensor [Электронный ресурс] : Bosch Sensortec. URL: https://ecen499-nasa.github.io/WI_20/index_elements/datasheets/BST-BME280-DS002.pdf (дата обращения: 27.04.2024).

15. Digital Compass IC HMC5883L [Электронный ресурс] : Honeywell. URL: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf (дата обращения: 27.04.2024).

16. DS18B20 Programmable Digital Thermometer // Maxim Integrated [Электронный ресурс] URL: https://amperkot.ru/static/3236/uploads/datasheets/DS18B20%20datasheet.pdf (дата обращения: 27.04.2024).

17. Extremely Accurate I2C-Integrated RTC/TCXO/Crystal [Электронный ресурс] : maxim integrated. URL: https://www.farnell.com/datasheets/2345152.pdf (дата обращения: 27.04.2024).

18. Kestrel® 4500 POCKET WEATHER TRACKER [Электронный ресурс] : RICHARD PAUL RUSSELL. URL: https://docs.rs-online.com/e4c2/0900766b80c09496.pdf (дата обращения: 27.04.2024).

19. PCF8574 Remote 8-Bit I/O Expander for I2C Bus [Электронный ресурс] : Texas Instruments Incorporated. URL: https://www.elmatrix.ru/upload/docs/pcf8574.pdf (дата обращения: 27.04.2024).

20. PFM Step-up DC/DC Converter, ME2108 Series [Электронный ресурс] : MicrOne. URL: https://exotechnology.ru/pdf/ME2108A50M3G.pdf (дата обращения: 27.04.2024).

21. TP4056 1A Standalone Linear Li-lon Battery Charger with Thermal Regulation in SOP-8 [Электронный ресурс] : NanJing Top Power ASIC Corp. URL: https://robot-kit.ru/wa-data/public/blog/download/TP4056.pdf (дата обращения: 27.04.2024).

Приложение А

Код программы

```
#define DIAGNOSTIC_MOD 0
#define WE_HAVE_DS18B20 1
#define round(x)  ((x)>=0?(long)((x)+0.5):(long)((x)-0.5))
#define SAMPLE_INTERVAL_FOR_WINDSPEED_MILLISEC 1000
#define AMOUNT_OF_WINDSPEED_VALUES 3
#define SAMPLE_INTERVAL_FOR_WIND_ANGLE_MILLISEC 500
#define AMOUNT_OF_WIND_ANGLE_VALUES 3
#define TIMEOUT_FOR_ANEMOMETER 5000
#define TIMEOUT_FOR_NOTS 3000
#define SPEED_ENKODER_PPR 1200
#define RADIUS 0.1
#define WIND_KOEF 20
#define SPEED_ENCODER_PIN_A 2
#define DS18B20_SENSOR_PIN 6
#define DIRECTION_ENCODER_PIN_A 3
#define DIRECTION_ENCODER_PIN_B 4
#define DIRECTION_ENCODER_PIN_Z 5
#define DIRECTION_ENCODER_PPR 1200
#define VALUE_OF_ALTIDUDES 9
#define SAMPLE_INTERVAL_FOR_TEMP_PRES_FUNK_LOOP_MILLISEC 2000
#include "Wire.h"
#include <DFRobot_QMC5883.h>
DFRobot_QMC5883 compass;
#include <GyverBME280.h>
GyverBME280 bme;
#include <microDS18B20.h>
MicroDS18B20<DS18B20_SENSOR_PIN> DS18B20sensor;
#include <microDS3231.h>
MicroDS3231 rtc;
#include <EEPROM.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

uint32_t tmrForLoop;
uint32_t tmrForTemp;
uint32_t tmrForWindSpeed;
uint32_t tmrForWindAngle;
uint32_t tmrForLoopSec;
uint32_t tmrForMeteoRemind;
uint32_t tmrForMeteoRemindShowM;

byte Delta[] =
{
  0x00,
  0x00,
  0x00,
  0x00,
```

```
  0x00,
  0x04,
  0x0A,
  0x1F
};

byte M[] =
{
  0x00,
  0x00,
  0x00,
  0x11,
  0x1B,
  0x15,
  0x11,
  0x11
};

byte B[] =
{
  B00000,
  B00000,
  B00000,
  B11100,
  B10000,
  B11100,
  B10100,
  B11101
};

byte D[] =
{
  B00000,
  B00000,
  B00000,
  B01110,
  B01010,
  B01010,
  B11111,
  B10001
};

byte Y[] =
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x09,
  0x05,
```

```
  0x02,
  0x14
};

byte rifleUp[] =
{
  B00000,
  B00000,
  B00100,
  B01110,
  B10101,
  B00100,
  B00100,
  B00100
};

byte rifleDown[] =
{
  B00000,
  B00000,
  B00100,
  B00100,
  B00100,
  B10101,
  B01110,
  B00100
};

byte rotateToNorth[] =
{
  B10010,
  B11010,
  B10110,
  B10010,
  B00000,
  B00111,
  B00011,
  B00101
};

#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] =
{
  {'1','2','3','C'},
  {'4','5','6','U'},
  {'7','8','9','D'},
  {'O','0','L','R'}
```

```
};
byte rowPins[ROWS] = {7, 8, 9, 10}; //connect to the row pinouts of the keypad
byte colPins[COLS] =  {11, 12, 14, 15};//connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

char key;

char keyCheck()
{
  key=keypad.getKey();
  return key;
};

const PROGMEM uint8_t tempertureDeltaAtAltitudes[10][47]  =
{
        ***
};

//////////////////////////////////////////////////////////////велечина делты угла
const PROGMEM uint8_t windGridAgleDeltaAtAltitudes[9] =
{

***

};

//////////////////////////////////////////////////////////////велечина скорости ветра
const PROGMEM uint8_t windSpeedDeltaAtAltitudes[10][14] =
{

***

};


float scaler;
boolean scaler_flag;
float normal_vector_length;
float xv, yv, zv;
float calibrated_values[3];
float heading;

void GetHeading();
void Transformation(float uncalibrated_values[3]);
void vector_length_stabilasation();

volatile int windAngle;
const int increaseStep = (6000/DIRECTION_ENCODER_PPR);
```

```
void windAngleEncoder()
{
 if (digitalRead(DIRECTION_ENCODER_PIN_B))
 {
  windAngle-=increaseStep;
 }
 else
 {
  windAngle+=increaseStep;
 }

 if ((digitalRead(DIRECTION_ENCODER_PIN_Z))||(abs(windAngle)>=6000))
 {
  windAngle=0;
 }
}

volatile int counterForSpeed;

void windSpeedEncoder()
{
 ++counterForSpeed;
}

bool windSpeedValueIsReady = false;
bool windGridAngleIsReady = false;
bool airTempValueIsReady = false;
bool airPresValueIsReady = false;

////////////////////////////////////   Classes    ////////////////////////
class WindDirection
{
 public:
 WindDirection();

 void SetMagneticAdjustment(int gridMagneticAzimuthAdjustment);
 int GetMagneticAdjustment();

 int GetWindGridAngle();
 int GetHeadingMil();

 int ReadoutMagneticAzimuth();
 void CalculatWindGridAngle();

 void SetInfoUseMagnetometr(bool useMagneticAzimuth);
 bool GetInfoUseMagnetometr();

 private:
 int headingMil;
 int gridMagneticAzimuthAdjustment;
```

```
  int windGridAngle;
  bool useMagneticAzimuth;
  //bool CheckCompassWiring();
};
WindDirection WindDir;

class WindSpeed
{
  public:
  void CalculatWindSpeed();
  int GetSpeedValue();

  private:
  int speedValue;
};
WindSpeed WindSpd;

class Temperature
{
  public:
  Temperature();
  void ReadoutTheTemperature();
  int GetTemperture();


  private:
  int temperature;
  bool useDS18B20;
};
Temperature AirTemp;

class AirPressure
{
  public:
  void ReadoutThePressureBMP280();

  uint16_t GetPressure();

  private:
  uint16_t  pressure;
};
AirPressure AirPres;

class Meteo11BallparkTelegram
{
  public:
  void CalculateMeteo11(uint8_t dateOfMeteo, uint8_t hourOfMeteo, uint8_t
wholeMinutesOfMeteo);
```

```
  void SetAltitude(uint16_t altitude);

  uint8_t GetMinutesForUpdateMeteo();
  uint8_t GetHoursForUpdateMeteo();
  void SetMinutesForUpdateMeteo(uint8_t minutesForUpdate);
  void SetHoursForUpdateMeteo(uint8_t hoursForUpdate);

  void  SetInfoAboutAutoupdateMeteo(bool infoAboutAutoupdateMeteo);
  bool GetInfoAboutAutoupdateMeteo();
  uint8_t GetDateOfMeteo();
  uint8_t GetHourOfMeteo();
  uint8_t GetMinutesOfMeteo();

  uint16_t GetAltitude();

  uint16_t GetGroundPressureDifference();
  uint8_t GetGroundTempertureDifference();

  uint8_t GetAltitudeTempertureDifferencesValues(int i);
  uint8_t GetAltitudeWindAgleDifferencesValues(int i);
  uint8_t GetAltitudeWindSpeedDifferencesValues(int i);

  bool haveToUpdateMeteo;
  bool showHaveToUpdateMeteo;

  private:
  uint8_t altitudeTempertureDifferencesValues[VALUE_OF_ALTIDUDES];
  uint8_t altitudeWindAgleDifferencesValues[VALUE_OF_ALTIDUDES];
  uint8_t altitudeWindSpeedDifferencesValues[VALUE_OF_ALTIDUDES];

  uint8_t dateOfMeteo;
  uint8_t hourOfMeteo;
  uint8_t minutesOfMeteo;

  uint16_t altitude;
  uint16_t groundPressureDifference;
  uint8_t groundTempertureDifference;

  int8_t CheckForExtraAugend();
};
Meteo11BallparkTelegram Meteo;

void putInParameters();
void askingForInputTheAltitude();
void askingForInputTheMagneticAdjustment();\
```

```
void functionLcdForMeteoRemind(bool infoAboutAutoupdateMeteo, uint8_t hoursForUpdate,
uint8_t minutesForUpdate, uint8_t verticalCursorPosition);
void functionLcdForChangeTheTime(int8_t minutes, int8_t hours, int8_t date, int8_t month,
int16_t year);

////////////////////////////////////          MAIN       ////////////////////////
void setup()
{
 if (DIAGNOSTIC_MOD)
 {
  Serial.begin(9600);
 }
 Wire.begin();
 compass.begin();
 bme.begin(0x76);
 lcd.init();
 lcd.backlight();
 rtc.setTime(COMPILE_TIME); // установка на часах время и дату времени и даты
компеляции и загрузки кода//////////////////////////////////////////////////////////////
 lcd.home();

 lcd.createChar(0, Delta);
 lcd.createChar(1, M);
 lcd.createChar(2, B);
 lcd.createChar(3, D);
 lcd.createChar(4, Y);
 lcd.createChar(5, rifleUp);
 lcd.createChar(6, rifleDown);
 lcd.createChar(7, rotateToNorth);


 putInParameters();

 lcd.noBlink();


 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Turn fluger");

 do
 {
  if(digitalRead(DIRECTION_ENCODER_PIN_Z))
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Wait...");

   delay(TIMEOUT_FOR_ANEMOMETER);
   WindDir.CalculatWindGridAngle();
```

```
    }
   }
   while(!windGridAngleIsReady);


   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Getting wind speed.");
   lcd.setCursor(0, 1);
   lcd.print("Wait...");

   WindSpd.CalculatWindSpeed();

   AirTemp.ReadoutTheTemperature();
   AirPres.ReadoutThePressureBMP280();
   Meteo.CalculateMeteo11(rtc.getDate(), rtc.getHours(), rtc.getMinutes());

   lcd.clear();
}

uint8_t menuOption = 2;
uint8_t menuSelection=0;

void loop()
{
  keyCheck();


  switch(key)
  {
    case 'L':
    {
      lcd.clear();
      menuOption--;
      break;
    }
    case 'R':
    {
      lcd.clear();
      menuOption++;
      break;
    }
  }
  switch(menuOption)
  {
    case 0:
    {
      menuOption = 1;
      break;
    }
    case 5:
```

```
{
   menuOption = 4;
   break;
  }
 }

 if (menuOption==1)
 {
  int8_t cursorPos=0;
  int8_t menuScrolPosition=0;

  int8_t position1=0;
  int8_t position2=1;
  int8_t position3=2;
  int8_t position4=3;
  int8_t position5=50;
  int8_t position6=50;

  String stringForParametrs;
  stringForParametrs.reserve(5);
  String stringForMagneticAdjustmentLCD;
  stringForMagneticAdjustmentLCD.reserve(6);
  if((WindDir.GetMagneticAdjustment())>0)
  {
   stringForParametrs+="+";
    stringForParametrs+=WindDir.GetMagneticAdjustment();

   stringForMagneticAdjustmentLCD+="+";
   stringForMagneticAdjustmentLCD+=(WindDir.GetMagneticAdjustment()/100);
   stringForMagneticAdjustmentLCD+="-";
   stringForMagneticAdjustmentLCD+=(WindDir.GetMagneticAdjustment()%100);
  }

  if((WindDir.GetMagneticAdjustment())<=0)
  {
   stringForParametrs+="-";
   stringForParametrs+=abs(WindDir.GetMagneticAdjustment());
   stringForMagneticAdjustmentLCD+="-";
   stringForMagneticAdjustmentLCD+=abs((WindDir.GetMagneticAdjustment()/100));
   stringForMagneticAdjustmentLCD+="-";
   stringForMagneticAdjustmentLCD+=abs((WindDir.GetMagneticAdjustment()%100));
  }

  do
  {
   keyCheck();
   if(key=='R')
   {
    lcd.clear();
    menuOption++;
```

```
  break;
}

switch(key)
{
  case 'U':
  {
    lcd.setCursor(0, cursorPos);
    lcd.print(" ");
    cursorPos--;
    break;
  }
  case 'D':
  {
    lcd.setCursor(0, cursorPos);
    lcd.print(" ");
    cursorPos++;
    break;
  }
}

switch(cursorPos)
{
  case -1:
  {
    lcd.clear();
    if(menuScrolPosition==0)
    {
      cursorPos=3;
      menuScrolPosition=2;
    }
    else
    {
      menuScrolPosition--;
      menuScrolPosition=constrain(menuScrolPosition, 0, 2);
      cursorPos = 0;
    }
    break;
  }
  case 4:
  {
    lcd.clear();
    if(menuScrolPosition==2)
    {
      cursorPos=0;
      menuScrolPosition=0;
    }
    else
    {
      menuScrolPosition++;
```

```
    menuScrolPosition=constrain(menuScrolPosition, 0, 2);
    cursorPos = 3;
   }
   break;
  }
}

switch(menuScrolPosition)
{
 case 0 :
 {
  position1=0;
  position2=1;
  position3=2;
  position4=3;
  position5=50;
  position6=50;
  break;
 }
 case 1:
 {
  position1=50;
  position2=0;
  position3=1;
  position4=2;
  position5=3;
  position6=50;
  break;
 }
 case 2:
 {
  position1=50;
  position2=50;
  position3=0;
  position4=1;
  position5=2;
  position6=3;
  break;
 }
}
lcd.setCursor(0, cursorPos);
lcd.print(">");

if(position1<50)
{
 lcd.setCursor(1, position1);
 lcd.print("1)Update meteo");
}
if(position2<50)
{
```

```
  lcd.setCursor(1, position2);
  lcd.print("2)h="); lcd.print(Meteo.GetAltitude());
 }
 if(position3<50)
 {
  lcd.setCursor(1, position3);

lcd.print("3)");lcd.write(0);lcd.print("A");lcd.write(1);lcd.print(",");lcd.write(3);lcd.write(4);lcd.
print("="); lcd.setCursor(10, position3);
  lcd.print(stringForMagneticAdjustmentLCD);
 }

 if(position4<50)
 {
  lcd.setCursor(1, position4);
  lcd.print("4)Use compass: ");

  if(WindDir.GetInfoUseMagnetometr())
  {
   lcd.print("Yes");
  }
  else
  {
   lcd.print("No");
  }
 }

 if(position5<50)
 {
  lcd.setCursor(1, position5);
  lcd.print("5)Time/date");
 }

 if(position6<50)
 {
  lcd.setCursor(1, position6);
  lcd.print("6)Meteo remind");
 }

 if((cursorPos==position1)&&(key=='O')) //update meteo
 {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Wait...");

  WindDir.CalculatWindGridAngle();

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Getting wind speed.");
```

```
  lcd.setCursor(0, 1);
  lcd.print("Wait...");

  WindSpd.CalculatWindSpeed();

  AirTemp.ReadoutTheTemperature();
  AirPres.ReadoutThePressureBMP280();
  Meteo.CalculateMeteo11(rtc.getDate(), rtc.getHours(), rtc.getMinutes());
  lcd.clear();
  menuOption=3;
 }

 if((cursorPos==position2)&&(key=='O')) //changeAltitude
 {
  changeAltitude();
  lcd.clear();
 }

 if((cursorPos==position3)&&(key=='O')) //changeGridMagneticAzimuthAdjustment()
 {
  changeGridMagneticAzimuthAdjustment();
  if((WindDir.GetMagneticAdjustment())>0)
  {
   stringForParametrs="+";
   stringForParametrs+=WindDir.GetMagneticAdjustment();

   stringForMagneticAdjustmentLCD="+";
   stringForMagneticAdjustmentLCD+=(WindDir.GetMagneticAdjustment()/100);
   stringForMagneticAdjustmentLCD+="-";
   stringForMagneticAdjustmentLCD+=(WindDir.GetMagneticAdjustment()%100);
  }

  if((WindDir.GetMagneticAdjustment())<=0)
  {
   stringForParametrs="-";
   stringForParametrs+=abs(WindDir.GetMagneticAdjustment());

   stringForMagneticAdjustmentLCD="-";
   stringForMagneticAdjustmentLCD+=abs((WindDir.GetMagneticAdjustment()/100));
   stringForMagneticAdjustmentLCD+="-";
   stringForMagneticAdjustmentLCD+=abs((WindDir.GetMagneticAdjustment()%100));
  }
  lcd.clear();
 }

 if((cursorPos==position4)&&(key=='O')) //changeInfoAboutUsingCompas()
 {
  changeInfoAboutUsingCompas();
  lcd.clear();
 }
```

```
    if((cursorPos==position5)&&(key=='O'))//changeTheTime()
    {
     changeTheTime();
     lcd.clear();
    }

    if((cursorPos==position6)&&(key=='O')) //Meteo autoupdate
    {
     meteoRemind();
     lcd.clear();
    }
   }
   while(menuOption==1);
  }

  if (menuOption==2)
  {
   lcd.setCursor(0, 0);
   lcd.print("T="); lcd.print(AirTemp.GetTemperture());
   lcd.setCursor(0, 1);
   lcd.print("P="); lcd.print(AirPres.GetPressure());
   lcd.setCursor(0, 2);
   lcd.print("V="); lcd.print(WindSpd.GetSpeedValue());lcd.print(" m/s");
   lcd.setCursor(0, 3);
   lcd.write(224);lcd.print("w=");lcd.print((int)(WindDir.GetWindGridAngle()/100));lcd.print("
");lcd.write(2);lcd.write(3);lcd.write(4);

   lcd.setCursor(15, 0);
   if((rtc.getHours())<10)
   {
    lcd.print("0");
   }
   lcd.print(rtc.getHours());
   lcd.setCursor(18, 0);
   if((rtc.getMinutes())<10)
   {
    lcd.print("0");
   }

   lcd.print(rtc.getMinutes());

   lcd.setCursor(12, 1);
   if((rtc.getDate())<10)
   {
    lcd.print("0");
   }
   lcd.print(rtc.getDate());lcd.print(".");
   if((rtc.getMonth())<10)
   {
```

```
  lcd.print("0");
 }
 lcd.print(rtc.getMonth());lcd.print(".");
 if((rtc.getYear()-2000)<10)
 {
  lcd.print("0");
 }
 lcd.print(rtc.getYear()-2000);

 if (millis() - tmrForLoopSec >= 1000)
 {
  tmrForLoopSec = millis();
  lcd.setCursor(17, 0);
  if(rtc.getSeconds()&1)
  {
   lcd.print(" ");
  }
  else
  {
   lcd.print(":");
  }
 }

 if(!(WindDir.GetInfoUseMagnetometr()))
 {
  lcd.setCursor(17, 2);
  lcd.write(7);
 }
 else
 {
  lcd.setCursor(17, 2);
  lcd.print(" ");
 }
 if ((millis() - tmrForMeteoRemindShowM >= 1500)&&(Meteo.haveToUpdateMeteo))
 {
  tmrForMeteoRemindShowM = millis();
  lcd.setCursor(19, 2);
  if(Meteo.showHaveToUpdateMeteo)
  {
   Meteo.showHaveToUpdateMeteo=false;
   lcd.print("M");
  }
  else
  {
   Meteo.showHaveToUpdateMeteo=true;
   lcd.print(" ");
  }
 }
}
```

```
 if (menuOption==3)
 {

  lcd.setCursor(0, 0);
  lcd.print("Meteo11 appr");
  lcd.setCursor(0, 1);

  if((Meteo.GetDateOfMeteo())<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetDateOfMeteo());

  if((Meteo.GetHourOfMeteo())<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetHourOfMeteo());

  lcd.print(Meteo.GetMinutesOfMeteo()/10); lcd.print("-");

  if((Meteo.GetAltitude())<10)
  {
   lcd.print("000");
  }
  else
  {
   if((Meteo.GetAltitude())<100)
   {
    lcd.print("00");
   }else

   {
    if((Meteo.GetAltitude())<1000)
    {
     lcd.print("0");
    }
   }
  }
  lcd.print(Meteo.GetAltitude()); lcd.print("-");
  if(Meteo.GetGroundPressureDifference()<10)
  {
   lcd.print("00");
  }

if((Meteo.GetGroundPressureDifference()>10)&&(Meteo.GetGroundPressureDifference()<100)
)
  {
   lcd.print("0");
  }
```

```
lcd.print(Meteo.GetGroundPressureDifference());
if(Meteo.GetGroundTempertureDifference()<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetGroundTempertureDifference()); lcd.print("-");
lcd.setCursor(0, 2);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////02


lcd.print("02-");
if((Meteo.GetAltitudeTempertureDifferencesValues(0))<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(0));

if((Meteo.GetAltitudeWindAgleDifferencesValues(0))<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(0));

if((Meteo.GetAltitudeWindSpeedDifferencesValues(0)<10))
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(0));
lcd.print("-");

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////04
lcd.print("04-");
if(Meteo.GetAltitudeTempertureDifferencesValues(1)<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(2));

if(Meteo.GetAltitudeWindAgleDifferencesValues(1)<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(1));

if(Meteo.GetAltitudeWindSpeedDifferencesValues(1)<10)
{
  lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(1));
```

```
 lcd.print("-");

 lcd.setCursor(0, 3);
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////08
 lcd.print("08-");
 if(Meteo.GetAltitudeTempertureDifferencesValues(2)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(2));

 if(Meteo.GetAltitudeWindAgleDifferencesValues(2)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(2));

 if(Meteo.GetAltitudeWindSpeedDifferencesValues(2)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(2));
 lcd.print("-");
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////12
 lcd.print("12-");
 if(Meteo.GetAltitudeTempertureDifferencesValues(3)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(3));

 if(Meteo.GetAltitudeWindAgleDifferencesValues(3)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(3));

 if(Meteo.GetAltitudeWindSpeedDifferencesValues(3)<10)
 {
  lcd.print("0");
 }
 lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(3));
 lcd.print("-");
}

if (menuOption==4)
{
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////16
 lcd.setCursor(0, 0);
 lcd.print("16-");
```

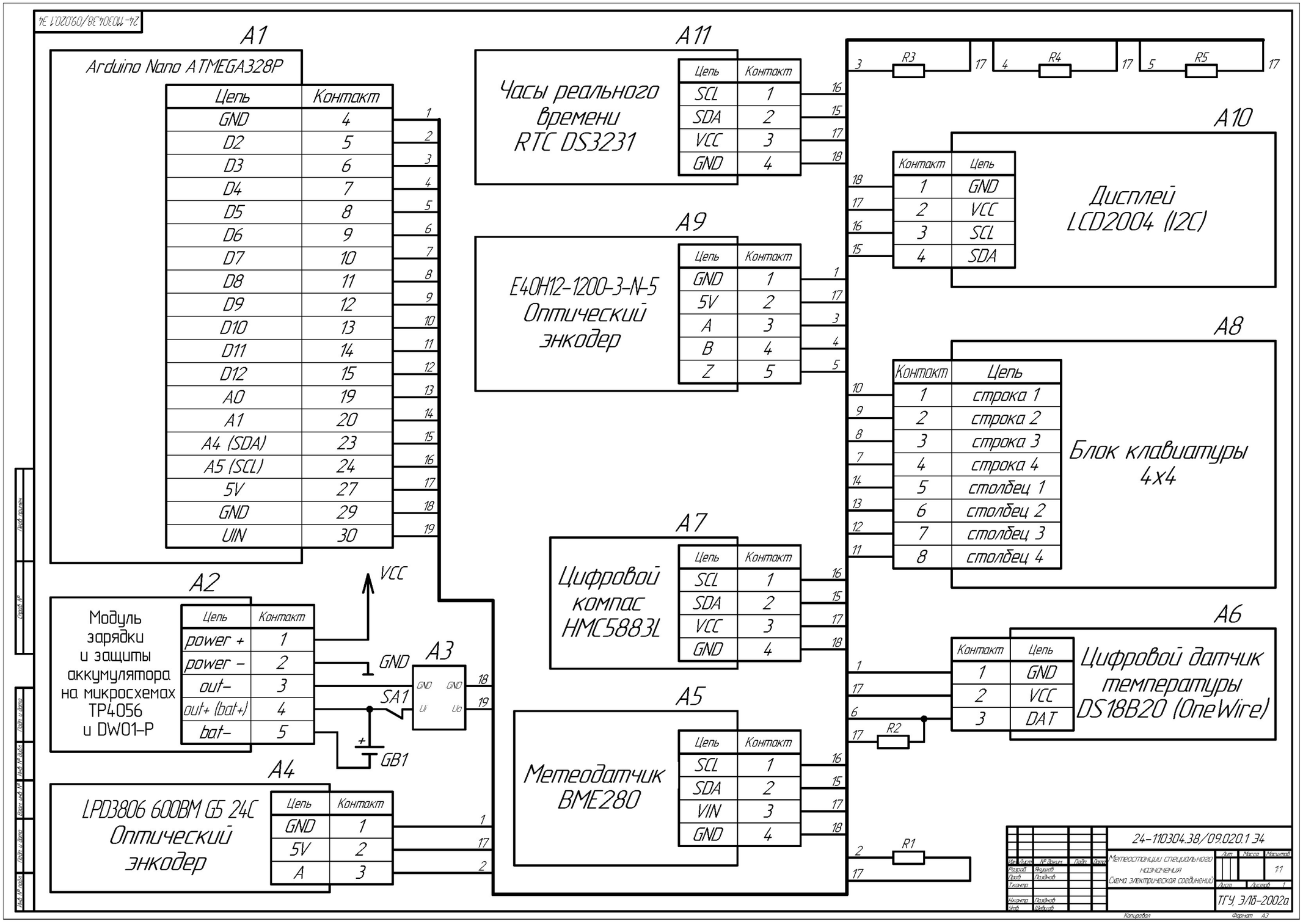
```
if(Meteo.GetAltitudeTempertureDifferencesValues(4)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(4));

if(Meteo.GetAltitudeWindAgleDifferencesValues(4)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(4));

if(Meteo.GetAltitudeWindSpeedDifferencesValues(4)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(4));
lcd.print("-");
////////////////////////////////////////////////////////////////////////////////////////////////////////////////20
lcd.print("20-");
if(Meteo.GetAltitudeTempertureDifferencesValues(5)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(5));

if(Meteo.GetAltitudeWindAgleDifferencesValues(5)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(5));

if(Meteo.GetAltitudeWindSpeedDifferencesValues(5)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(5));
lcd.print("-");
////////////////////////////////////////////////////////////////////////////////////////////////////////////////24
lcd.setCursor(0, 1);
lcd.print("24-");
if(Meteo.GetAltitudeTempertureDifferencesValues(6)<10)
{
 lcd.print("0");
}
lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(6));

if(Meteo.GetAltitudeWindAgleDifferencesValues(6)<10)
{
 lcd.print("0");
```

```
  }
  lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(6));

  if(Meteo.GetAltitudeWindSpeedDifferencesValues(6)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(6));
  lcd.print("-");
  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////30
  lcd.print("30-");
  if(Meteo.GetAltitudeTempertureDifferencesValues(7)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(7));

  if(Meteo.GetAltitudeWindAgleDifferencesValues(7)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(7));

  if(Meteo.GetAltitudeWindSpeedDifferencesValues(7)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(7));
  lcd.print("-");
  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////40
  lcd.setCursor(0, 2);
  lcd.print("40-");
  if(Meteo.GetAltitudeTempertureDifferencesValues(8)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeTempertureDifferencesValues(8));

  if(Meteo.GetAltitudeWindAgleDifferencesValues(8)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeWindAgleDifferencesValues(8));

  if(Meteo.GetAltitudeWindSpeedDifferencesValues(8)<10)
  {
   lcd.print("0");
  }
  lcd.print(Meteo.GetAltitudeWindSpeedDifferencesValues(8));
}
```

```
  if((millis() - tmrForMeteoRemind >=
(((Meteo.GetHoursForUpdateMeteo()*60)+Meteo.GetMinutesForUpdateMeteo())*60000))&&(
Meteo.GetInfoAboutAutoupdateMeteo()))
  {
    tmrForMeteoRemind=millis();
    askForUpdateMeteo();
  }
}
```

Приложение Б

Схема электрическая соединений



Поз обозначение	Наименование	Кол.	Примечание
	Модули		
A1	Палата Arduino Nano с	1	
	микроконтроллером ATMEGA328P		
A2	Модуль зарядки и защиты аккумулятора	1	
	на микросхемах TP4056 и DW01-p		
A3	Повышающий DC/DC 5В модулт ME2108	1	
A4	Оптический энкодер LPD3806 600BM G5 24C	1	
A5	Метеодатчик BME280	1	
A6	Цифровой датчик температуры DS18B20	1	
A7	Цифровой компас HMC5883L	1	
A8	Блок клавиатуры 4x4	1	
A9	Оптический энкодер E40H12-1200-3-N-5	1	
A10	Дисплей LCD2004 (I2C)	1	
A11	Часы реального времени RTC DS3231	1	
	Резисторы		
R1,	22388 10 кОм, 1 Вт	4	
R3, R4, R5			
R2	1122 4.7 кОм, 1 Вт	1	
	Коммутирующие элементы		
SA1	9000462985 Выключатель клавишный 250V 6A	1	
	Источники питания		
GB1	9000354355 Аккумулятор литий-полимерный	1	
	(Li-Pol) 700мАч		

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл

Изм.	Лист	№ докум	Подпись	Дата
Разраб.		Якушев		
Пров.		Позднов		
Рец.				
Н.контр		Позднов		
Утв.		Шевцов		

24-110304.38/09.020.1

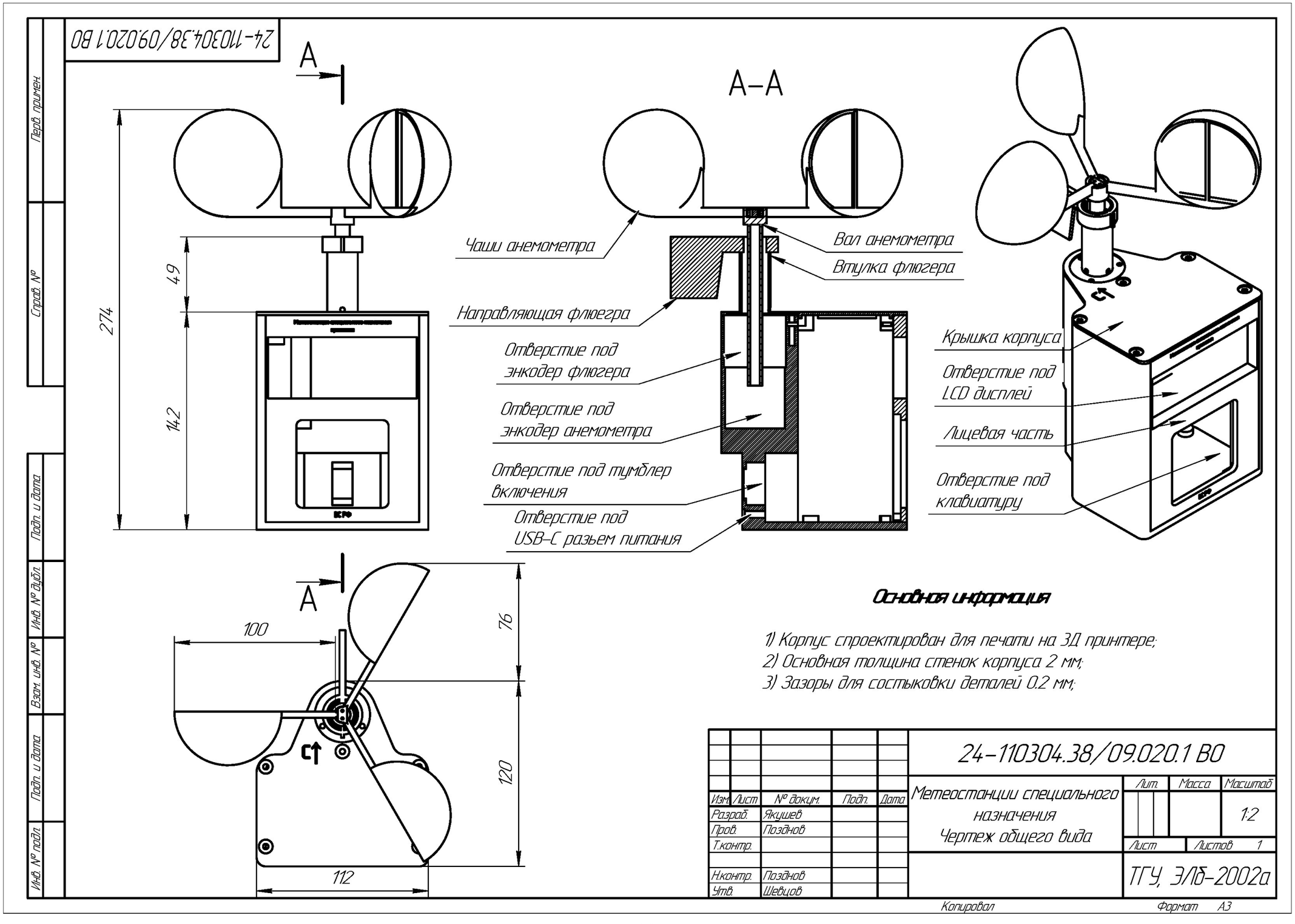
Метеостанции специального назначения
Схема электрическая соединений
Перечень элементов

Литера	Лист	Листов
		1

ТГУ ЭЛб-2002а

Приложение В

Чертеж общего вида



Основная информация

1) Корпус спроектирован для печати на 3Д принтере;
2) Основная толщина стенок корпуса 2 мм;
3) Зазоры для состыковки деталей 0.2 мм;

					24-110304.38/09.020.1 ВО	
Изм.	Лист	№ докум.	Подп.	Дата		
Разраб.		Якушев			Метеостанции специального назначения Чертеж общего вида	Лит. / Масса / Масштаб: 1:2
Пров.		Позднов				Лист / Листов: 1
Т.контр.						
Н.контр.		Позднов				ТГУ, ЭЛб-2002а
Утв.		Шевцов				

Копировал — Формат А3