ИНФОРМАТИКА

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ТЕРМОГИГРОМЕТРА НА БАЗЕ ARDUINO

Методические указания к курсовой работе
для студентов бакалавриата направления 05.03.06

САНКТ-ПЕТЕРБУРГ
2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ТЕРМОГИГРОМЕТРА НА БАЗЕ ARDUINO

Методические указания к курсовой работе
для студентов бакалавриата направления 05.03.06

САНКТ-ПЕТЕРБУРГ
2023

УДК 004:007.52 (073)

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ТЕРМОГИГРОМЕТРА НА БАЗЕ ARDUINO: Методические указания к курсовой работе. Сост.: *О.В. Косарев, Е.Г. Дементьева, Т.В. Сарапулова*. СПб, 2023. 40 с.

Рассмотрен пример программирования работы термогигрометра на базе аппаратно-программного комплекса Arduino. Приведены общие сведения об Arduino UNO и программной среде Arduino IDE, необходимые для начала работы с комплексом. Показан порядок подключения и программирования работы датчика температуры DHT11, ридера карт памяти формата microSD и часов реального времени на базе микросхемы DS1302.

Методические указания предназначены для студентов направления 05.03.06 «Экология и природопользование».

Научный редактор зав. кафедрой ИСиВТ *Е.Б. Мазаков*

Рецензент к.т.н. *К.В. Столяров* (компания “Telum Inc.”)

ВВЕДЕНИЕ

В методических указаниях к курсовой работе описан процесс разработки и программирования прибора для измерения температуры и влажности атмосферного воздуха - термогигрометра. Физические параметры измеряются с помощью датчика DHT11. Управляет датчиком и записью данных на карту памяти аппаратно-программный комплекс Arduino. В методических указаниях описаны назначение и характеристики АПК Arduino, порядок сборки и программирования работы термогигрометра. В процессе выполнения курсовой работы студенту необходимо: подключить датчик DHT11 к Arduino, запрограммировать его работу на считывание физических параметров в течение определенного периода времени, запрограммировать запись полученных величин и меток даты и времени на карту памяти. Метки даты и времени получить с часов реального времени. Запись данных на карту выполнить через ридер карт памяти. Датчик, часы реального времени и ридер должны быть подключены к плате Arduino. По полученным данным построить графики в MS Excel и добавить полиномиальную линию тренда степени не ниже 3. Результаты работы оформить по ГОСТ 7.32. Программный код и блок-схему алгоритма работы термогигрометра вынести в приложение. Схему подключения выполнить в программе Fritzing и подтвердить фото своей собранной установки. Следует помнить, что все подключения проводов к плате необходимо выполнять после отключения питания! Все примеры приведены для операционной системы Windows 10.

1. ИСХОДНЫЕ ДАННЫЕ

Исходными данными для выполнения расчетов являются: физические величины, подлежащие регистрации (температура, влажность, атмосферное давление), период и интервал наблюдения (период наблюдения – не менее 24 часов, интервал наблюдения – по ГОСТ 8756-2005 [1]), форма представления результата (график изменения величин и линия тренда). Необходимое оборудование: аппаратно-программный комплекс Arduino, датчик температуры, влажности и атмосферного давления, модуль реального времени. Необходимое программное обеспечение: среда программирования Arduino IDE или любая другая, MS Excel [2] и MS Word любой версии.

2. ОБЩИЕ СВЕДЕНИЯ ОБ ARDUINO

Проект Arduino представляет собой совокупность аппаратного (электронная плата с микроконтроллером и интерфейсами, как правило это Arduino UNO) и программного обеспечения (среда программирования, например, Arduino IDE) для создания проектов в области робототехники, интернета вещей, умного дома. Проект открытый, бесплатный, предназначен для тех, кто не имеет познаний в электронике, компьютерах и программировании. Сайт проекта – www.arduino.cc. Проект развивается под этим названием с 2008 года. Поэтому сейчас существует бесчисленное множество клонов платы Arduino UNO, примеров программ (скетчей), датчиков, двигателей, корпусов и других запчастей. Стоимость самой платы Arduino UNO – 300-400 рублей, датчиков – от 50 рублей. Программное обеспечение Arduino IDE – бесплатное. Для сборки проекта не нужен паяльник – все соединения выполнены на разъемах. Таким образом, стоимость простого проекта не превышает стоимости обеда в Макдональдсе. Рассмотрим более подробно плату Arduino UNO и ее подключение к компьютеру.

Внешний вид платы Arduino UNO (один из возможных вариантов) показан на рисунке 1. Это рисунок из руководства «Быстрый старт. Первые шаги по освоению Arduino» с сайта Maxkit.ru, являющегося, в свою очередь, переводом с английского языка руководства «SIK GUIDE» с сайта www.sparkfun.com/SIK. Для изучения можно

использовать любую другую доступную книгу [3,4] и информацию из интернета [5]. На сайте проекта www.arduino.cc есть много полезной информации и форум разработчиков устройств.

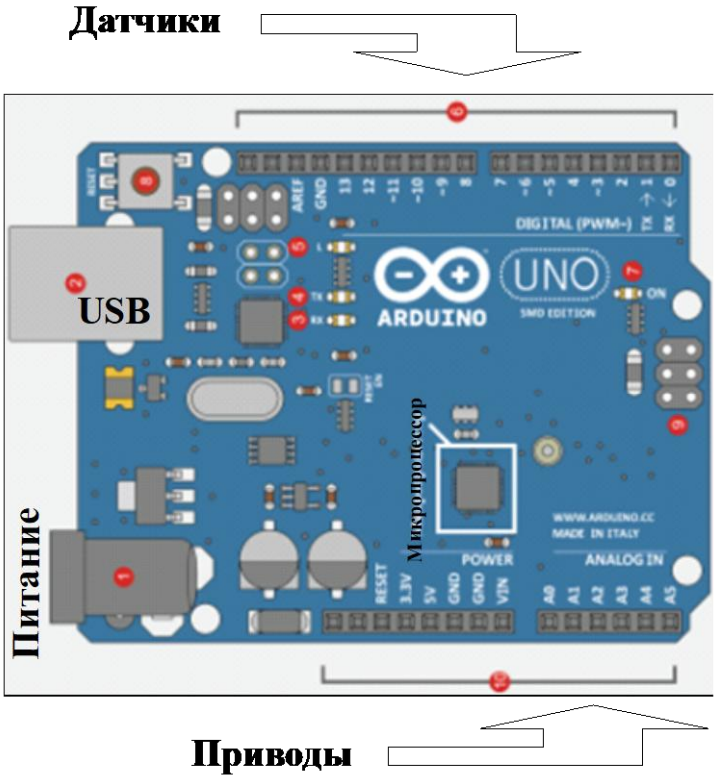


Рис. 1. Внешний вид платы Arduino Uno.

На рисунке 1 цифрами обозначены:

1 – разъем питания. Можно не использовать, если плата подключена к порту USB компьютера;

2 – разъем USB для подключения к компьютеру;

3, 4 – индикаторы, сигнализирующие о том, что плата принимает (3, Rx) или передает (4, Tx) данные;

5 – контрольный индикатор. Используется для контроля правильности работы загруженной программы. Мы будем мигать этим индикатором;
6 – разъемы с цифровыми портами ввода-вывода (DIGITAL, пронумерованы цифрами от 0 до 13). К этим портам будут подключаться цифровые датчики. Порты работают как на ввод информации, так и на вывод;
7 – индикатор питания. Если горит – значит, питание на плату подано. Этот индикатор может быть расположен в любом другом месте платы;
8 – кнопка сброса. Жмите её, если ничего не получается ☺;
9 – разъем для программирования платы (порт ICSP). В рамках курсовой работы этот разъем использоваться не будет;
10 – разъемы питания датчиков (POWER) и аналоговые порты ввода A0-A5 (ANALOG IN). К портам A0-A5 обычно подключаются разнообразные приводы (исполнительные механизмы) или на них подаются аналоговые сигналы.

Плата работает следующим образом. В память микропроцессора записывается программа. По алгоритму, заложенному в программе, опрашиваются подключенные к плате датчики или подаются сигналы управления на двигатели, светодиоды и другие устройства. Управляет процессом сбора информации с датчиков и выполнением промежуточных расчетов микропроцессор, расположенный на плате. Результаты выполнения программы сохраняются в памяти микропроцессора. На рисунке 1 микропроцессор находится примерно посередине платы. Это такая маленькая квадратная микросхема, обведенная белым прямоугольником. Микропроцессор может быть и в другом корпусе, например длинном прямоугольном. На работу это не влияет.

В платах Arduino UNO используется 8-битный RISC-микропроцессор ATmega328P фирмы Atmel. Микропроцессор построен по гарвардской архитектуре с раздельными памятью и шинами команд и данных. Обратите внимание, что она отличается от архитектуры фон Неймана (рисунок 2), используемой в классических компьютерах и ноутбуках. На рисунке 3 показана архитектура семейства микропроцессоров ATmegaXXXX. Сравните ее с архитек-

турой на рисунке 2 и найдите все элементы. Если плата построена на другом процессоре – ничего страшного, все будет работать.

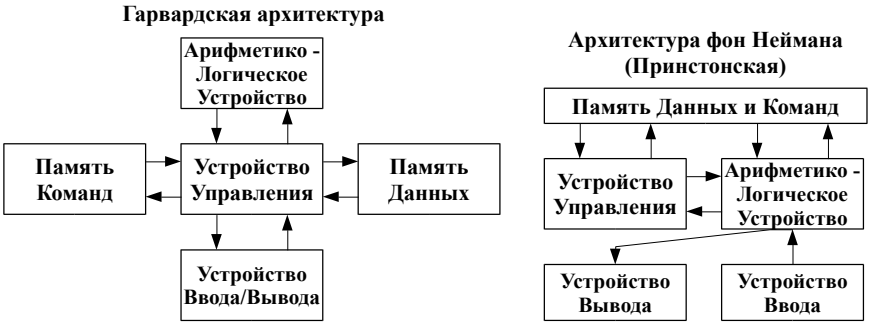


Рис. 2. Гарвардская и Принстонская архитектуры ЭВМ.

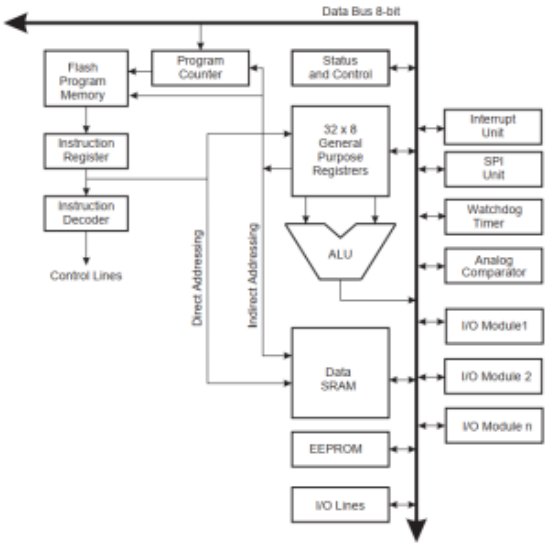


Рис. 3. Архитектура семейства микропроцессоров ATmegaXXXX.

Теперь рассмотрим, как подключить плату к компьютеру и загрузить программу в память микропроцессора.

3. ПРОГРАММИРОВАНИЕ ARDUINO

Для работы с Arduino на компьютер необходимо установить специальную программу. Существует несколько вариантов. На занятиях будет использоваться Arduino IDE (от англ. Integrated Development Environment – интегрированная среда разработки). Программу можно скачать с сайта по адресу https://www.arduino.cc/en/Main/Software. Доступны версии под операционные системы Windows, Linux и Mac OS. В пособии все примеры приведены для версии 1.8.10. Программа устанавливается обычным образом. Если права пользователя не позволяют установить программу, следует скачать автономную версию (Windows ZIP file for non admin install). После установки ее необходимо запустить, подключить плату Arduino к компьютеру по USB и указать в Arduino IDE к какому COM-порту подключена плата. Для связи платы с компьютером по USB на плате установлена специальная микросхема – преобразователь интерфейса USB в последовательный интерфейс UART. В недорогих платах это обычно микросхема CH340G или ATmega16U2, которая полностью эмулирует работу стандартного COM порта. Для корректного определения платы Arduino в операционной системе Windows необходимо установить драйвер этой микросхемы. Если плата определилась сразу после установки – ничего делать не нужно.

Окно установленной Arduino IDE выглядит, как показано на рисунке 4. Если плата подключена правильно, то можно прочитать информацию о ней по команде Инструменты\Получить_информацию_о_плате (рисунок 5).

Рассмотрим структуру программы. Программа для Arduino называется скетчем. Для записи скетчей практически не требуется никаких знаний о программировании. Тем не менее, будет полезно вспомнить три типа вычислительных процессов, определение типов переменных и функций, объекты, методы и свойства объектов.

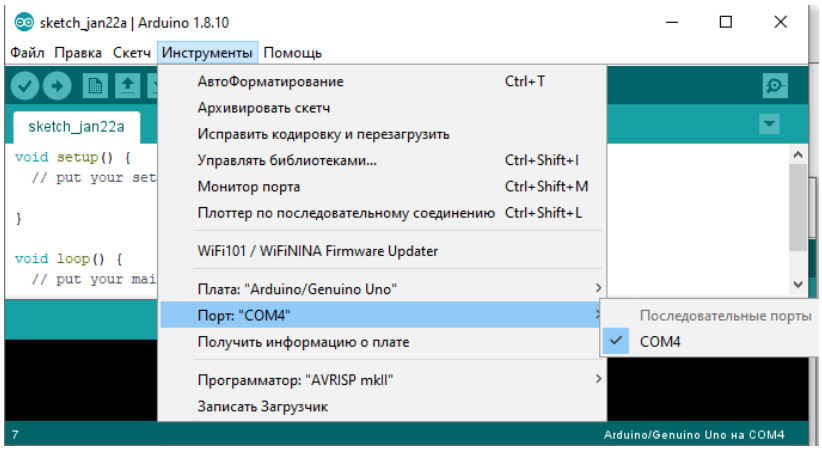


Рис. 4. Подключение платы в Arduino IDE.

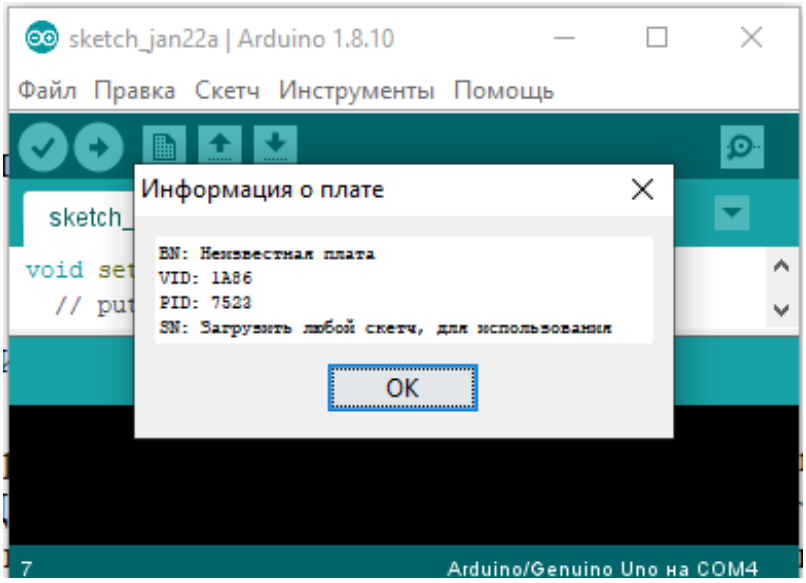


Рис. 5. Информация о плате в Arduino IDE.

Структура скетча для Arduino содержит:

1. Заголовок в виде комментария. Описано название и предназначение скетча. Может быть указана другая вспомогательная информация. **Это необязательная часть программы.**
// Так пишется однострочный комментарий
/ *А так пишется многострочный комментарий.*
Давайте помигаем светодиодом на плате. ****/***

2. Инструкция #define или константа const. Подключение библиотек #include. Инструкция #define используется для упрощения кода. Синтаксис: #define <что меняем> <на что меняем>. Очень похожа на переменную или константу. Замена происходит во всем коде перед компиляцией (преобразованием текста программы в двоичный код, вспоминаем принципы функционирования ЭВМ). Например, будем использовать для удобства название PIN_LED для обозначения порта, к которому подключен светодиод: #define PIN_LED 13. Теперь дальше в коде мы будем использовать имя PIN_LED, а вместо него будет подставляться число 13. Аналогичный результат модно получить при использовании константы: *const int PIN_LED = 13;*. Здесь «int» – тип константы PIN_LED, целочисленный тип. Аналогично подключаются специальные программы для работы с датчиками и приводами – библиотеки. Для подключения библиотеки используется инструкция #include. В этом же разделе при необходимости создаются Объекты подключаемых Классов (библиотек). Подключение библиотек и создание Объектов будет рассмотрено ниже. **Это необязательная часть программы.**

3. Функция void setup(). Функция называется «setup», слово «void» обозначает, что функция не возвращает никаких данных (в противном случае было бы необходимо вместо «void» написать тип возвращаемых данных, например «int» как при константе в п.2 выше). Все команды в этой функции (строки кода) должны находиться между фигурных скобок {}. Команды функции выполняются только один раз во время загрузки скетча. Это дает возможность пользователю поучаствовать в процессе загрузки. Здесь могут описываться

номера портов, устанавливаться скорость соединения с COM-портом, подключаться внешние программы (например NI LabVIEW) и др. Пример функции void setup() показан ниже. Здесь «pinMode» - функция определения режима работы цифрового порта; «PIN_LED» - константа номера порта; «OUTPUT» - режим работы порта, режим=ВЫХОД. **Это ОБЯЗАТЕЛЬНАЯ часть программы.**

```
// Программа мигания светодиодом
//#define PIN_LED 13 // Можно так задать порт
const int PIN_LED = 13; // А можно через константу

void setup() {
// инициализируем цифровой порт PIN_LED как выход.
  pinMode(PIN_LED, OUTPUT);
}
```

4. Функция void loop(). Функция называется «loop» (петля). Все команды, расположенные внутри этой функции, выполняются бесконечное количество раз, пока плата Arduino не будет выключена. Это основной блок программы, вся пользовательские алгоритмы заложены здесь. **Это ОБЯЗАТЕЛЬНАЯ часть программы.**

```
// Программа мигания светодиодом
//#define PIN_LED 13 // Можно так задать порт
const int PIN_LED = 13; // А можно через константу

void setup() {
// инициализируем цифровой порт PIN_LED как выход
  pinMode(PIN_LED, OUTPUT);
}

void loop() {
// зажигаем светодиод, HIGH – высокий уровень напряжения
  digitalWrite(PIN_LED, HIGH);
// ждем 1000 мс
  delay(1000);
// гасим светодиод
```

```
  digitalWrite(PIN_LED, LOW);
// ждем 1000 мс
  delay(1000);
// А дальше бесконечный повтор кода в скобках
}
```

5. Функции пользователя. В этом разделе можно написать код своей функции, обращение к которой будет в программе выше. Здесь же можно сделать вывод данных. **Это необязательная часть программы.**

Загрузим получившийся код в плату и помигаем светодиодом. Плата уже подключена, в окне редактора кода открыт скетч по умолчанию (рис.4). Сохраним наш скетч под любым именем, например, «Svetodiod», через меню «Сохранить как». Вместо пустого шаблона вставим код из пункта 4 и нажмем кнопку «Проверить» на панели инструментов (под кнопкой «Файл», круглая кнопка с пиктограммой галочки). Arduino IDE скомпилирует код и если все в порядке, выдаст сообщение «Компиляция завершена» (рисунок 6). Если в коде есть ошибки, то в нижней части редактора кода появятся сообщения об ошибках на английском языке оранжевым цветом. Если ошибок нет, то можно загружать код в плату. Для этого необходимо нажать кнопку «Загрузка» - вторая слева круглая кнопка с пиктограммой стрелки вправо (рисунок 7). После загрузки скетча светодиод на плате начнет равномерно мигать. Попробуйте изменить в коде одно из времен задержки и загрузить код заново. Что изменилось?

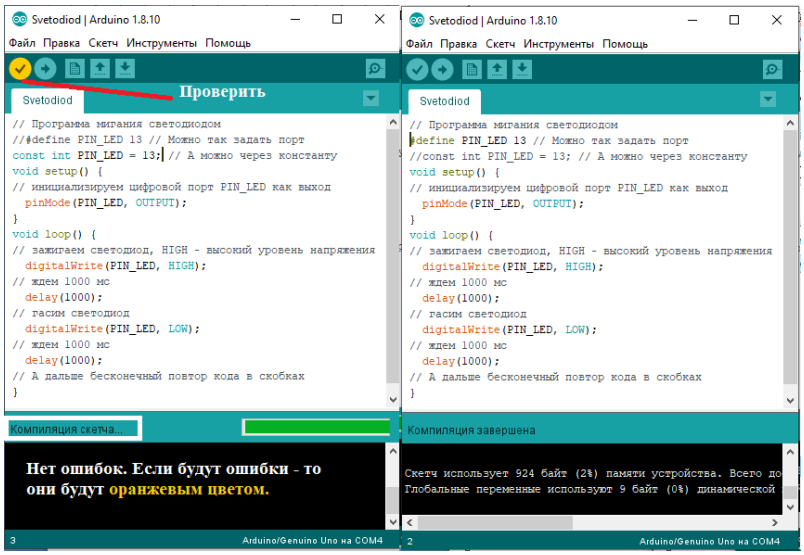


Рис. 6. Проверка (компиляция) скетча в Arduino IDE.

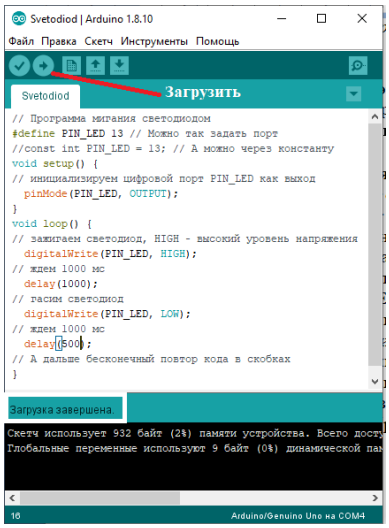


Рис. 7. Загрузка скетча в Arduino IDE.

4. ПРОГРАММИРОВАНИЕ ДАТЧИКА DHT11

Теперь подключим датчик температуры и считаем с него информацию. Будем использовать цифровой датчик DHT11, подключенный к цифровому порту 2. Для работы с датчиками, сервоприводами и другими устройствами, требуются специальные программы. Такие программы называются библиотеками (пункт 2 выше). Они не входят в состав пакета Arduino IDE и их нужно устанавливать дополнительно. Можно пользоваться любой библиотекой, поддерживающей DHT11. В этом пособии будет использоваться библиотека "DHT.h" b компании Adafruit. Библиотеки доступны по адресу https://github.com/adafruit/DHT-sensor-library и https://github.com/adafruit/Adafruit_Sensor. Необходимо скачать обе библиотеки (рисунок 8), распаковать архивы и загрузить папки с файлами в каталог *libraries* с установленной программой Arduino IDE: C:\Program Files (x86)\Arduino\libraries (путь до слова Arduino примерный, в учебных аудиториях он будет другой). Теперь подключим датчик температуры и считаем с него информацию. Будем использовать цифровой датчик DHT11, подключенный к цифровому порту 2. Датчик, выполненный на небольшой плате, имеет 3 вывода: + (плюс), out (выход датчика), - (минус, земля, GND). Эти выводы необходимо подключить проводами к контактам на плате 5V и GND с одной стороны, и вывод out к порту 2 с другой стороны платы (см. рис.1). Если перепутаете – ничего страшного, ничего не сгорит. Просто работать не будет.

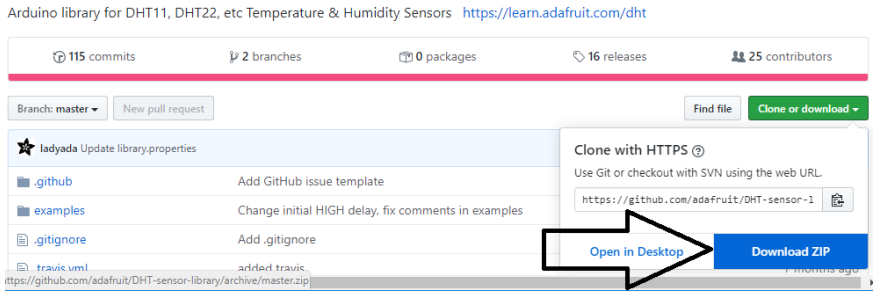


Рис. 8. Загрузка библиотеки с ресурса GitHub.com.

Создадим новый файл и сохраним его как TempHumid. При создании нового файла он сохраняется в выбранном пользователем месте (это не обязательно каталог, где установлена программа Arduino IDE) в папке с таким же названием. А сам файл имеет расширение *.ino – TempHumid.ino. Напишем код по структуре, описанной выше (5 разделов). Названия разделов в примере кода написаны для лучшего понимания материала, в своем коде их можно не писать.

Раздел 1 содержит только одну строку комментария. рекомендуется сюда добавить всю полезную информацию о скетче, авторе, дате, месте и цели создания. Во-первых, это поможет избежать путаницы при программировании. Во-вторых, это будет служить доказательством авторства кода. Вообще, весь код должен быть снабжен максимальным количеством комментариев. Это сэкономит очень много времени в будущем.

//Раздел 1
// Измерение температуры и влажности

В **разделе 2** подключим необходимую библиотеку: *#include <DHT.h>*. Папка с файлами библиотеки должна быть уже скачана и записана в нужное место! Переменной *DHTTYPE* присвоим значение *DHT11 (#define DHTTYPE DHT11),* а переменной *DHTPIN* присвоим значение *2 (#define DHTPIN 2).* Эти переменные будем использовать в качестве входных параметров вновь созданного Объекта *dht* Класса *DHT – DHT dht(DHTPIN, DHTTYPE);*. Здесь важно помнить следующее. Название Класса строго определено, и оно совпадает с названием папки и (файлов в ней) подключенной библиотеки. В данном случае в папке C:\Program Files (x86)\Arduino\libraries**DHT** находятся файлы **DHT.h** и **DHT.cpp**. Эти файлы и есть библиотека, которая была подключена командой *#include*. Название же Объекта может быть любым, на усмотрение автора кода. Желательно, чтобы название отражало цель создания этого Объекта. В качестве тренировки измените в своем коде название Объекта.

```
//Раздел 2
//Используется бибилиотека:
//DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
#include <DHT.h>
//Используется датчик DHT11
#define DHTTYPE DHT11 // Строка означает, что переменная
//DHTTYPE=DHT11
//Датчки подключим к цифровому порту 2
#define DHTPIN 2 //Строка означает, что переменная DHTPIN=2
//Инициализируем датчик DHT11
//Создадим Объект dht Класса DHT с параметрами DHTPIN и
//DHTTYPE
DHT dht(DHTPIN, DHTTYPE);
```

В **разделе 3** откроем последовательный порт на скорости 9600 бод - *Serial.begin(9600);*. Здесь *Serial* – Объект последовательного порта, а *begin(9600)* – метод объекта с параметром скорости «9600». В последовательный порт будем выдавать значения влажности и температуры с датчика (описано ниже). Посмотреть, что пришло в последовательный порт, можно с помощью **Монитора порта** на вкладке **Инструменты** Arduino IDE. После открытия порта выведем в него приветственное сообщение - *Serial.println("Начинаем измерять…").* Здесь *println* – Метод печати в порт, с переносом каретки в конце строки на новую строку. Аналогичным образом «открываем» датчик темепратуры - *dht.begin();*. Какими Свойствами и Методами обладает Объект *dht* Класса *DHT* можно посмотреть в файлах DHT.h и DHT.cpp. Файлы открываются в Блокноте.

```
//Раздел 3
void setup() {
  //Открываем последовательный порт на скорости 9600 бод
  Serial.begin(9600);
  //Приветствие перед началом работы датчика
  //Выводим текст в Монитор порта, меню Инструменты вверху
//Arduino IDE
  Serial.println("Начинаем измерять влажность и температуру!");
  //"Открываем" датчик измерения температуры и влажности
```

```
  dht.begin();
}
```

В **разделе 4** выполняется основная работа по чтению данных с датчика. Вначале делается пауза длительностью 2000 мс - *delay(2000);* Во время паузы микропроцессор не выполняет никаких действий. В первое выполнение цикла она конечно не нужна. Но так как цикл бесконечный, можно считать, что пауза делается в конце основного кода ☺. Далее, используя Метод *readHumidity* Объекта *dht* и Метод *readTemperature*, записываем в соответствующие переменные *h* и *t* типа данных «с плавающей запятой» значение влажности и температуры – *float h = dht.readHumidity();* и *float t = dht.readTemperature();*. Присвоение значений переменных выполнено одновременно с их объявлением и описанием типа данных. Если на выходе датчика нет информации, то в серийный порт будет выдан ошибка (конструкция *if*). Так будет продолжаться до тех пор, пока данные не появятся. После отладки всего кода попробуйте вытащить провод из гнезда порта 2 и посмотреть, что получится. Последняя часть кода – вывод значений влажности и температуры в последовательный порт с помощью Методов *print* и *println*.

```
//Раздел 4. Выполняется бесконечно
void loop() {
  //Делаем паузу между измерениями пару секунд, см. коммент.
  //ниже
  delay(2000);
  // Чтение температуры и влажности занимает примерно 250 мс!
  // Датчик очень медленный, может выдавать данные 2-х
  //секундной давности
  //Используем Метод readHumidity Объекта dht
  //Влажность записываем в переменную h типа float
  float h = dht.readHumidity();
  //Используем Метод readTemperature Объекта dht
  //Температуру в градусах Цельсия записываем в переменную h
  //типа float
  float t = dht.readTemperature();
  //Проверим данные на выходе датчика. Проверяем, пока данные не
  //появятся
```

```
  if (isnan(h) || isnan(t)) {
    //Нет данных? Выводим сообщение об ошибке в
  //последовательный порт
    Serial.println("Ошибка датчика!");
    //Возвращаемся к проверке данных на выходе датчика
    return;
  }
//Выводим данные в серийный порт
  Serial.print("Влажность: ");
  Serial.print(h);
  Serial.print("%  Температура: ");
  Serial.print(t);
  Serial.println("°C");
  }
```

В **разделе 5** в этом скетче ничего нет.

На рисунке 9 показан Монитор порта и данные с датчика. Попробуйте подержать датчик в руках, подышать на него, положите датчик под свет настольной лампы. Если все заработало, можно приступать к сохранению информации на внешний носитель.

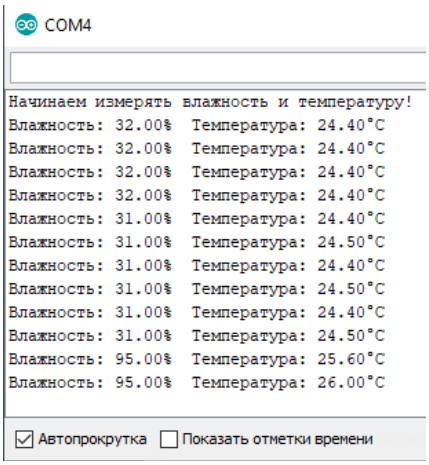


Рис. 9. Результат работы скетча TempHumid.ino.

5. ПРОГРАММИРОВАНИЕ РИДЕРА MICROSD КАРТ

Получить данные о влажности и температуре – это конечно хорошо. Но еще лучше – забрать их с собой и построить график. Например, в Excel. Сохранить результат измерений можно на карту памяти формата microSD. Для чтения карты памяти понадобится специальный ридер misroSD для Arduino. В этом разделе описано применение ридера HW-125. Ридер работает по протоколу SPI [6,7], поэтому для его работы требуются библиотеки <SPI.h> и <SD.h>. Эти библиотеки входят в дистрибутив Arduino IDE, поэтому необходимо их просто подключить. В меню Файл\Примеры\SD Arduino IDE есть пример чтения информации с карты – CardInfo. Необходимо запустить этот пример, подключить ридер с картой по инструкции в начале примера, и прочитать информацию о карте. На рисунке 10 показан заголовок примера с описанием подключения платы ридера к плате Arduino. Необходимо найти на плате ридера контакты: земля, питание, MOSI, MISO, CLK (или CSK), CS и соединить их проводами с цифровыми портами на плате Arduino. Первые два необходимо соединить со свободными разъемами GND и 3.3V. Какой контакт ридера к какому порту подключать показано на рисунке 10.

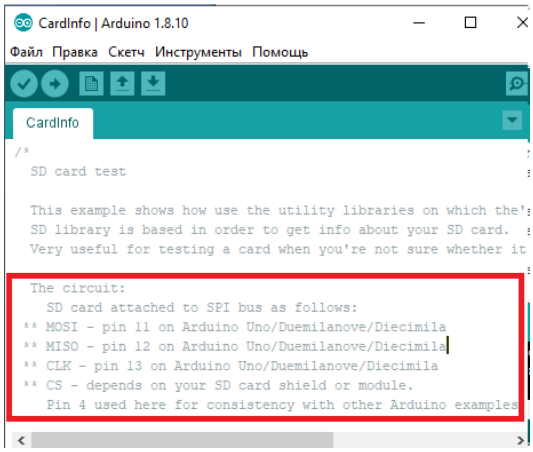


Рис. 10. Описание подключения ридера карт microSD.

Карту памяти предварительно необходимо отформатировать штатным образом в Проводнике Windows. Подключаем ридер с установленной картой памяти к Arduino и загружаем скетч CardInfo. **Перед извлечением и установкой карты памяти в ридер необходимо отключать питание от платы Arduino!** Если все выполнено правильно, в Мониторе порта увидим информацию о карте (рисунок 11). Теперь можно приступать к записи на карту влажности и температуры. В этом нам поможет другой пример из Файл\Примеры\SD, а именно Datalogger. Рассмотрим его.

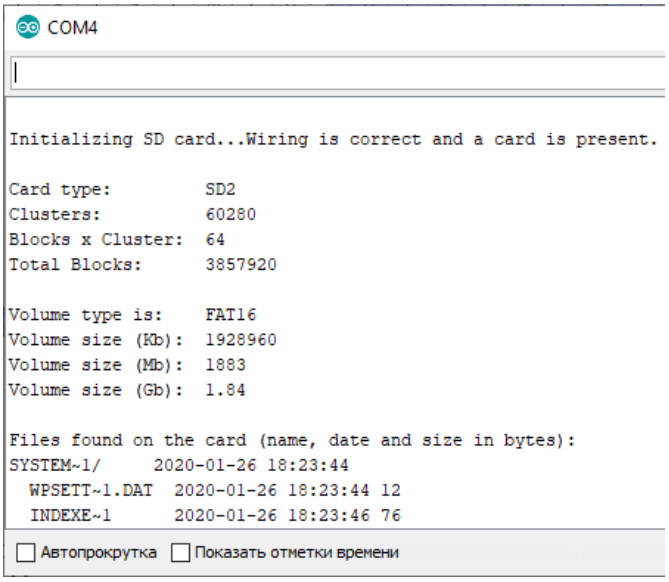


Рис. 11. Информация о карте из скетча CardInfo.

Скетч Datalogger.ino записывает данные с трех аналоговых входов в текстовый файл. Нам необходимо модифицировать наш скетч TempHumid.ino фрагментами кода записи данных на карту памяти из скетча Datalogger.ino. Для начала скопируем в наш скетч информацию о физическом подключении ридера и подключим необходимые библиотеки (рисунок 12).

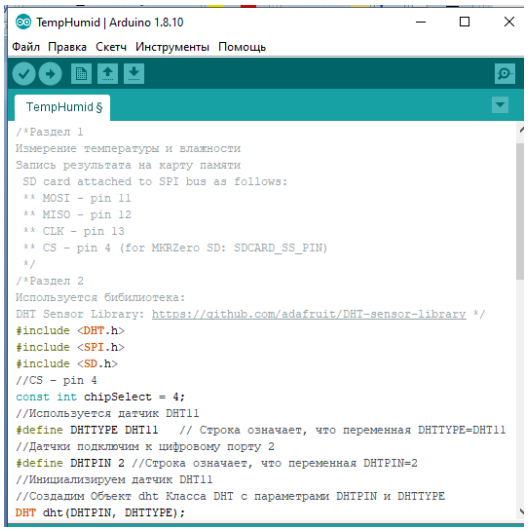


Рис. 12. Модификация разделов 1 и 2.

Теперь добавим в Раздел 3 (функция setup) код подключения карты памяти (рисунок 13).

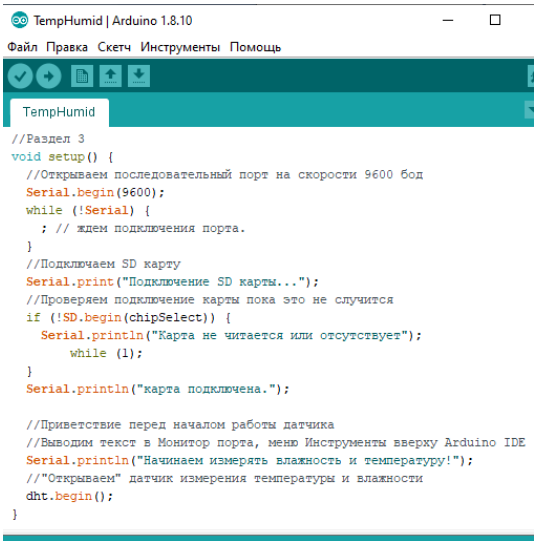


Рис. 12. Модификация функции void setup().

В раздел 4 внесем дополнения в самый конец кода, перед последеней закрывающей скобкой. Создадим переменную *dataString* типа *String* и сразу присвоим ей пустое значение: *String dataString = "";*. В эту переменную будем записывать через точку с запятой (требование формата SCV) значение влажности и температуры из переменных *h* и *t*. Для согласования типов данных преобразуем переменные из формата с плавающей запятой (float) в текстовый формат, сохраняя два знака после запятой. Для этого воспользуемся функцией *String()* (не путайте с типом данных String): *String(h,2), String(t,2).* Точку с запятой преобразовывать не нужно, просто возьмем ее в кавычки. Сложим последовательно текст влажности, точку с запятой и текст температуры, и присвоим это значение переменной *dataString*: *dataString=String(h,2)+";"+String(t,2);*. Далее создаем переменную *dataFile* класса (типа) *File* и присваиваем ей результат выполнения метода *open* объекта *SD* с параметрами *"datalog.csv"* и *FILE_WRITE*:
File dataFile = SD.open("datalog.csv", FILE_WRITE);. Попросту говоря, открываем для записи (или создаем и открываем для записи, если его не было) на карте памяти файл *"datalog.csv"* и присваиваем переменной *dataFile* это действие. Далее проверяем с помощью условия, удалось ли создать (открыть) файл. Если все хорошо, пишем в файл с новой строки значение переменной *dataString* и закрываем файл: *dataFile.println(dataString); dataFile.close();*. Ранее в эту переменную мы поместили влажность и температуру. Для проверки выводим в Монитор порта записанное значение: *Serial.println(dataString);*. После отладки кода эту строку можно закомментировать. Если с файлом что-то не то, пишем в Монитор порта сообщение об ошибке: *Serial.println("Ошибка при открытии datalog.csv");*. Весь код целиком показан на рисунке 13.

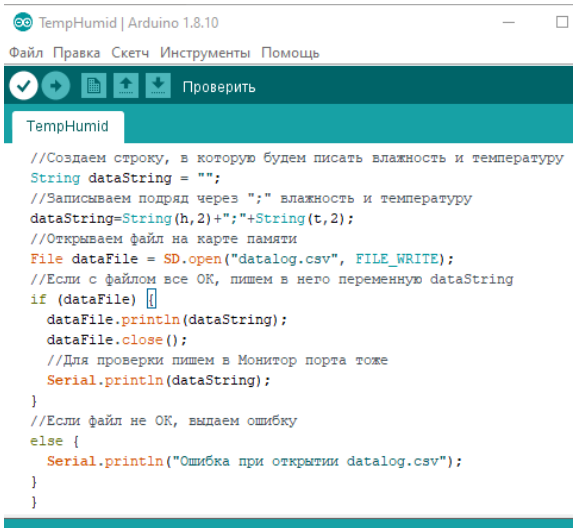


Рис. 13. Модификация функции void loop().

Если все сделано правильно, в Мониторе порта увидим соответствующие значения (рисунок 14).

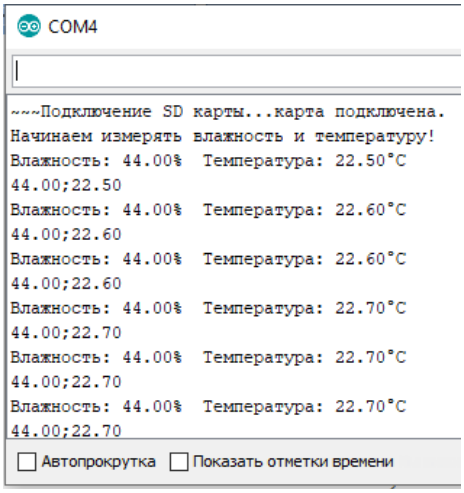


Рис. 14. Информация в Мониторе порта.

Подышите немного на датчик и выключайте плату. На карте памяти должен появиться файл DATALOG.CSV (рисунок 15). Обратите внимание на дату создания файла.

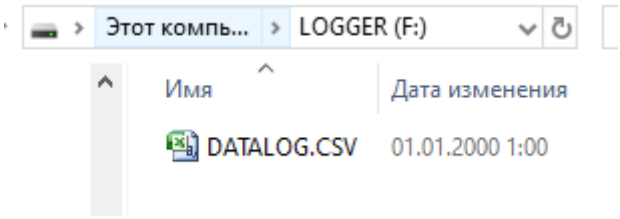


Рис. 15. Файл с данными на карте памяти.

Экспортируем данные из файла на лист Excel. Для этого выполним команду ***Данные\Из текста*** применительно к файлу на карте памяти (рисунок 16). Жмем ***Далее*** до второго шага.

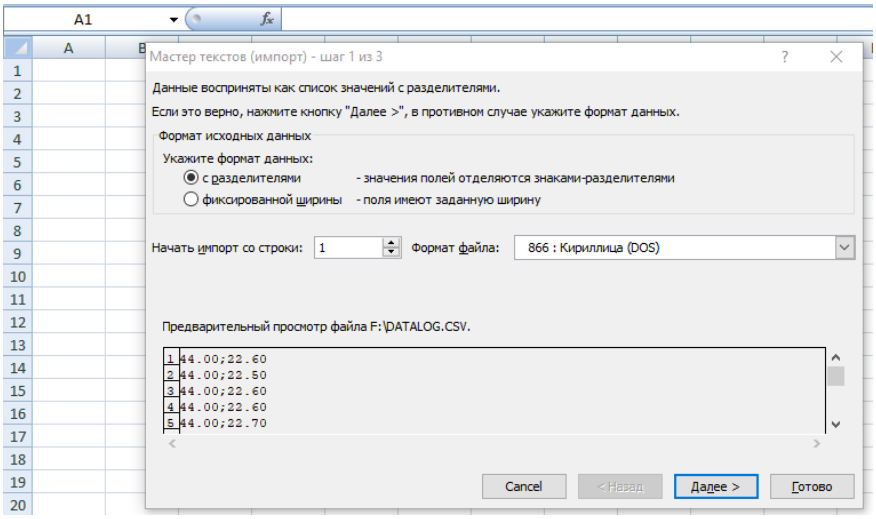


Рис. 16. Первый шаг импорта данных из файла.

На втором шагу указываем правильный символ-разделитель столбцов (точку с запятой, рисунок 17).

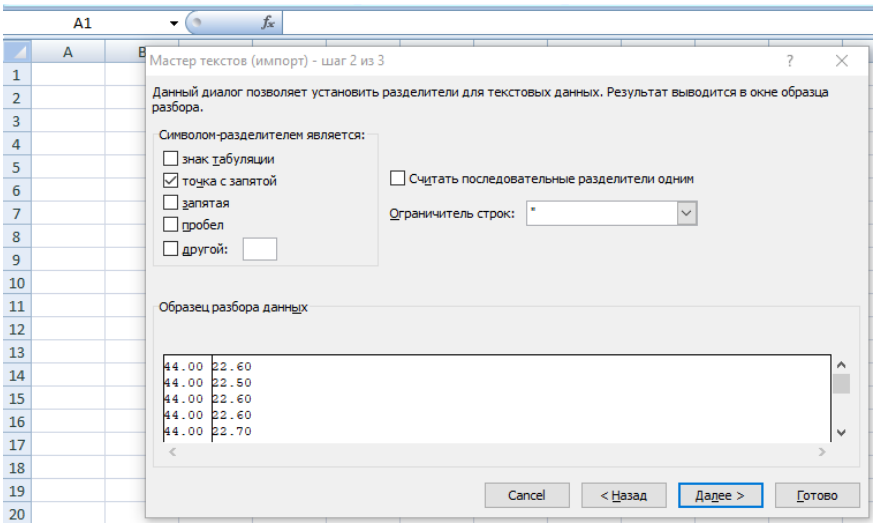


Рис. 17. Выбор символа-разделителя на втором шаге импорта данных из файла.

На третьем шаге жмем кнопку ***Подробнее*** и выбираем в качестве разделителя целой и дробной части **точку**. Выделяем второй столбец и делаем тоже самое (рисунок 18). Нажимаем ***Готово***, ***Ок*** и получаем два столбца данных. Теперь по этим данным можно строить график (рисунок 19). Конечно такое резкое изменение влажности не характерно для обычных условий. После отладки кода нужно установить интервал чтения параметров по [1].

На рисунке 15 видно, что с датой изменения информации на карте памяти что-то не то. Дело в том, что плата Arduino не содержит в своем составе часы реального времени. В следующем разделе описано, как подключить часы реального времени и добавить в файл с параметрами отметку времени.

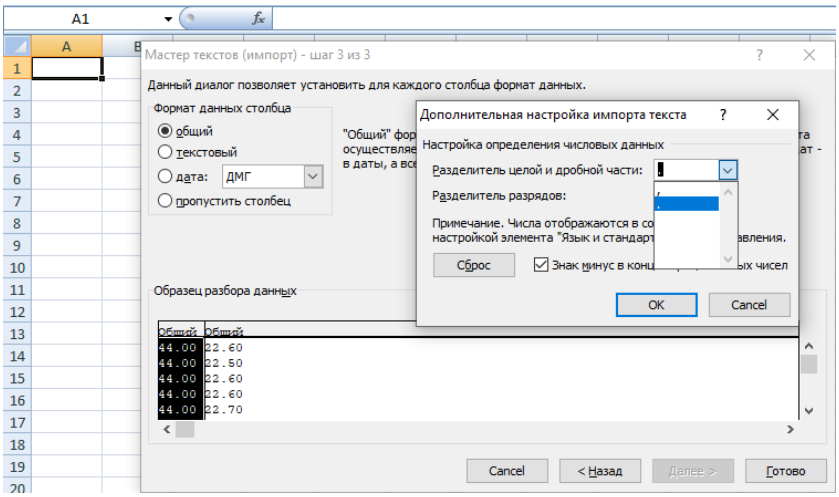


Рис. 18. Выбор разделителя целой и дробной части на третьем шаге импорта.

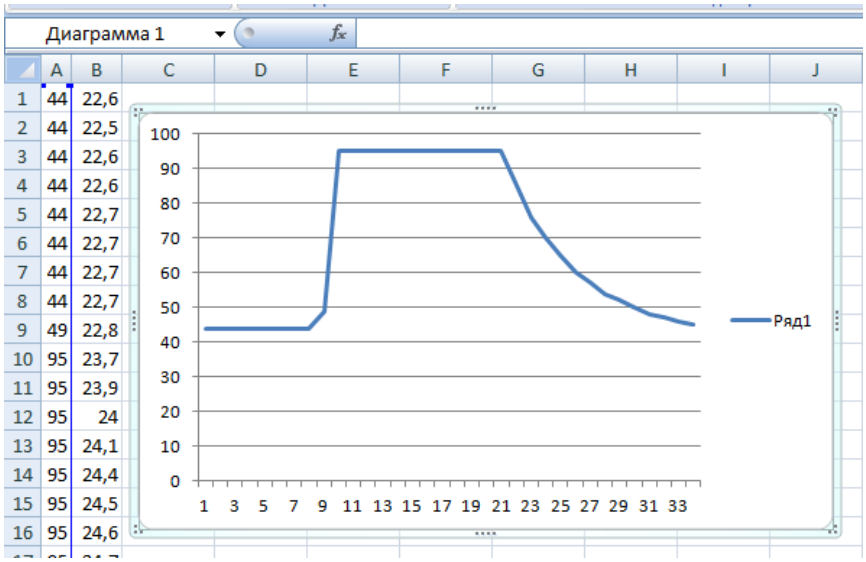


Рис. 19. График влажности по данным из файла на карте памяти.

5. ПРОГРАММИРОВАНИЕ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ

В этом пособии описано использование часов реального времени (Real Time Clock) на базе микросхемы DS1302. Собственно эта микросхема и есть часы. Микросхема установлена на плате с батарейкой и задающим кварцевым генератором с частотой 32.768 кГц. Эта информация важна для правильного выбора библиотеки RTC. Допустимо использовать любые другие доступные RTC. В этом случае код скетча необходимо будет соответствующим образом изменить.

Для работы с RTC необходимо установить любую подходящую библиотеку, например "RTClib.h", доступную по ссылке https://github.com/NeiroNx/RTCLib. Установка библиотеки аналогична установке "DHT.h" в разделе 4. После установки библиотеки в разделе с примерами появится папка RTCLib. В этой папке есть скетч ds1302.ino, который необходимо открыть. Фрагмент скетча показан на рисунке 20.

ds1302 §

```
#include "RTClib.h"

// Init rtc object
//   DS1302 rtc;
//   DS1302 rtc(ce_pin, sck_pin, io_pin);
//
// ce_pin  (RST): default 4
// sck_pin (CLK): default 5
// io_pin  (DAT): default 6
DS1302 rtc;
//DS1302 rtc(8, 6, 7);

void setup () {
 Serial.begin(9600);
 rtc.begin();

  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date
    rtc.adjust(DateTime(__DATE__, __TIME__));
  }
}
```

Рис. 20. Фрагмент скетча ds1302.ino из библиотеки RTCLib.

На основе этого скетча напишем вспомогательный скетч начальной установки в RTC актуальной даты и времени *DataTimeSetup.ino* и затем модифицируем скетч TempHumid.ino. Рассмотрим фрагмент скетча ds1302.ino на рисунке 20 подробно.

В первой строке подключается библиотека "RTClib.h". Затем в комментариях описано создание Объекта rtc и возможные варианты физического подключения платы RTC к плате Arduino. Плата RTC имеет пять выводов: земля, питание, RST, CLK, DAT. Первые два необходимо соединить со свободными разъемами GND и 5V на плате Arduino. Остальные три (информационные) по умолчанию подключаются к цифровым портам 4,5,6. Скетч предусматривает вариант альтернативного подключения к портам 8,6,7. Воспользуемся именно этим вариантом (в рассматриваемом скетче он закомментирован), т.к. порт 4 у нас занят ридером карт памяти. Далее в теле функции *setup()* идет инициализация последовательного порта на скорости 9600 бод и инициализация RTC. Здесь все аналогично инициализации датчика температуры и ридера карт памяти (разделы 4 и 5 пособия). Ниже в теле функции if проверяется подключение RTC и устанавливается дата и время. Функция установки даты и времени *rtc.adjust(DateTime(__DATE__, __TIME__));* записывает в микросхему DS1302 на плате RTC **текущую дату и время** загрузки скетча в плату Arduino. Эта функция должна выполниться один единственный раз. Именно ради нее и будет создан скетч начальной установки в RTC актуальной даты и времени DataTimeSetup.ino. Код скетча (на основе кода ds1302.ino) показан на рисунке 21. Функция loop() на рисунке 21 не показана ради экономии места, т.к. она пустая (функция есть, но не содержит код). Скетч DataTimeSetup.ino необходимо загрузить в плату Arduino один раз. Тем самым в плате RTC будет настроена правильная дата и правильное время. Больше к этому скетчу обращаться не нужно. Теперь проверим работу RTC с помощью скетча ds1302.ino. Внесем правки в скетч (изменим порты, отключим функцию *rtc.adjust*, функцию *loop()* оставим без изменения) и загрузим его в плату Arduino. Правки показаны на рисунке 22.

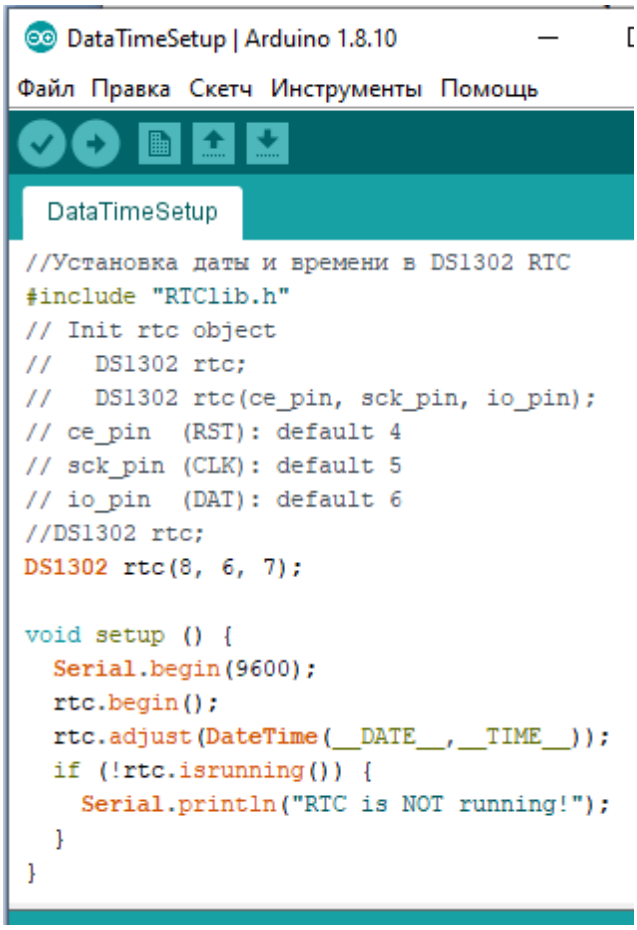


Рис. 21. Фрагмент скетча DataTimeSetup.ino.

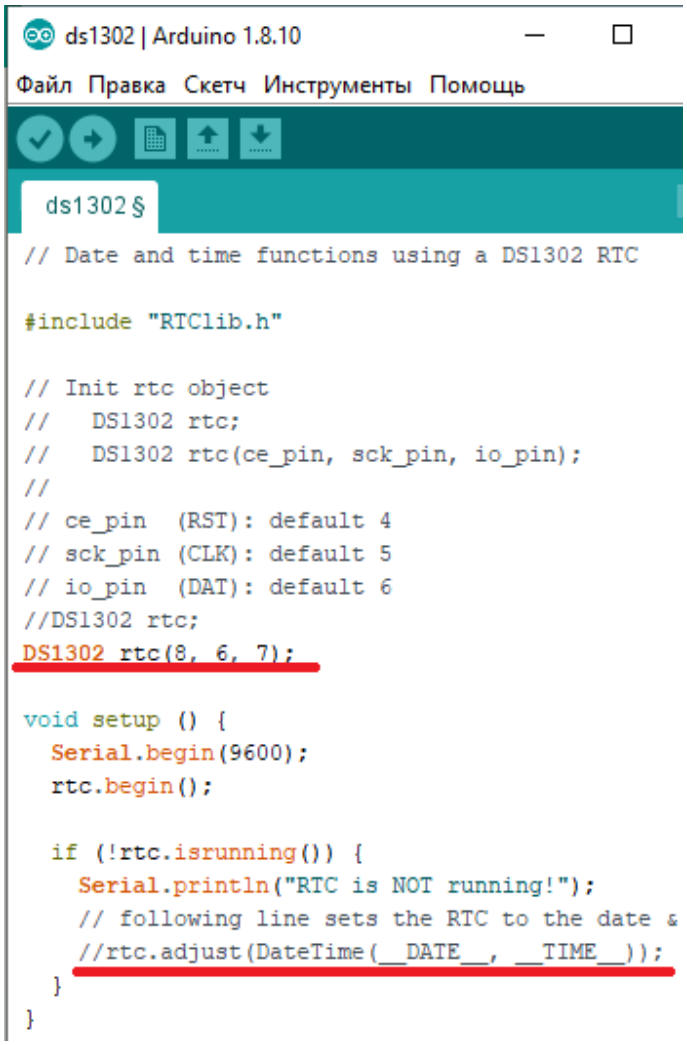


Рис. 22. Фрагмент скетча ds1302.ino.

Результат работы скетча в Мониторе порта показан на рисунке 23. Как видно на рисунке, плата RTC выдает актуальную дату и время. Попробуйте отключить плату Arduino от питания минут на

5-10. Затем снова подключите и откройте Монитор порта. В плате автоматически запустится скетч ds1302.ino, а в Мониторе порта опять будет актуальная дата. Таким образом, у нас появились актуальные метки даты и времени, и они не зависят от отключения питания! Добавим эти метки к нашему файлу на карте памяти.

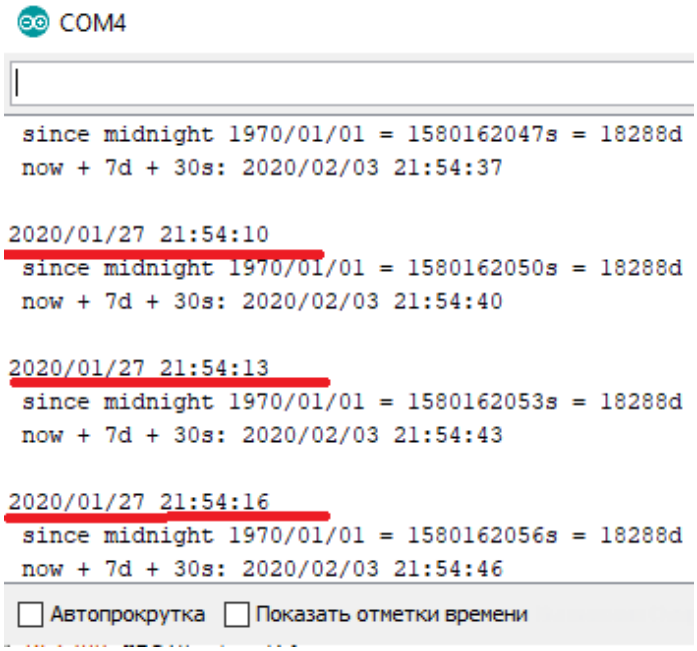


Рис. 23. Актуальная дата в Мониторе порта.

В раздел 1 скетча TempHumid.ino допишем описание подключения RTC к портам (рисунок 24).

Рис. 24. Описание подключения RTC.

В разделе 2 добавим ссылку на библиотеку RTC, подключим саму библиотеку *#include <RTClib.h>*, создадим Объект *rtc* Класса *DS1302* с параметрами *8,6,7* (порты, к которым подключен модуль RTC, рисунок 25).

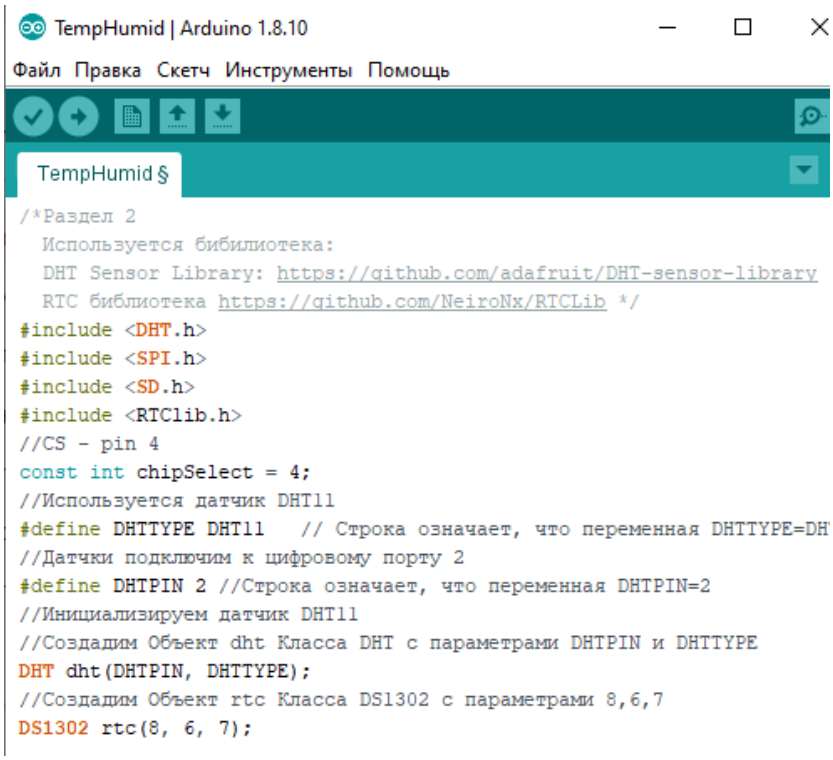


Рис. 25. Подключение библиотеки RTC.

В разделе 3 инициализируем модуль RTC командой *rtc.begin();* (рисунок 26).

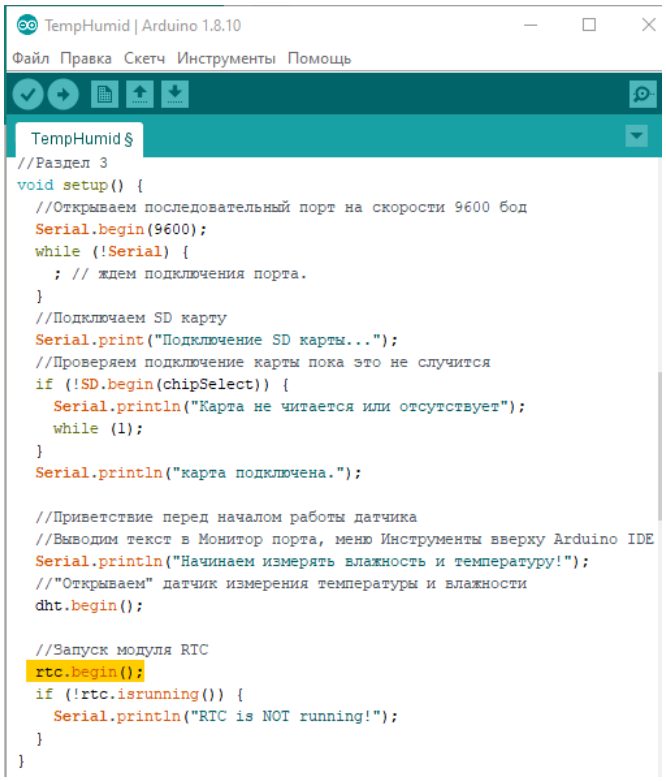


Рис. 26. Инициализация модуля RTC.

В разделе 4 объявим новую переменную *now* для хранения даты и времени *DateTime now = rtc.now();*, добавим вывод этой переменной в Монитор порта *Serial.println(now.toStr());* и в переменную dataString:
dataString = String(h, 2) + ";" + String(t, 2) + ";"+now.toStr();
(рисунок 27). Переменная dataString записывается на карту памяти, поэтому после выполнения скетча на карте памяти должны появиться дата и время. Проверим это, импортировав данные с карты памяти после выполнения скетча (как в разделе 4). **Перед извлечением и установкой карты памяти в ридер необходимо отключать питание от платы Arduino!** В противном случае карте может испор-

титься, и данные будут потеряны. На рисунке 28 показа Монитор порта при работе скетча TempHumid.ino. На рисунке видно, что отметки времени идут через 2 секунды, что соответствует задержке измерения, установленной в самом начале функции *loop()*.

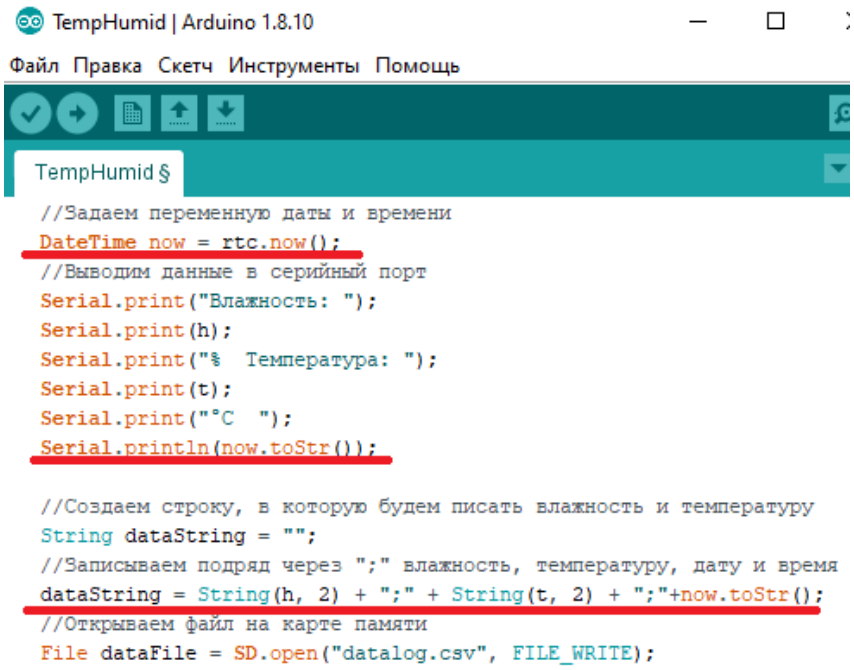


Рис. 27. Добавление в скетч даты и времени.

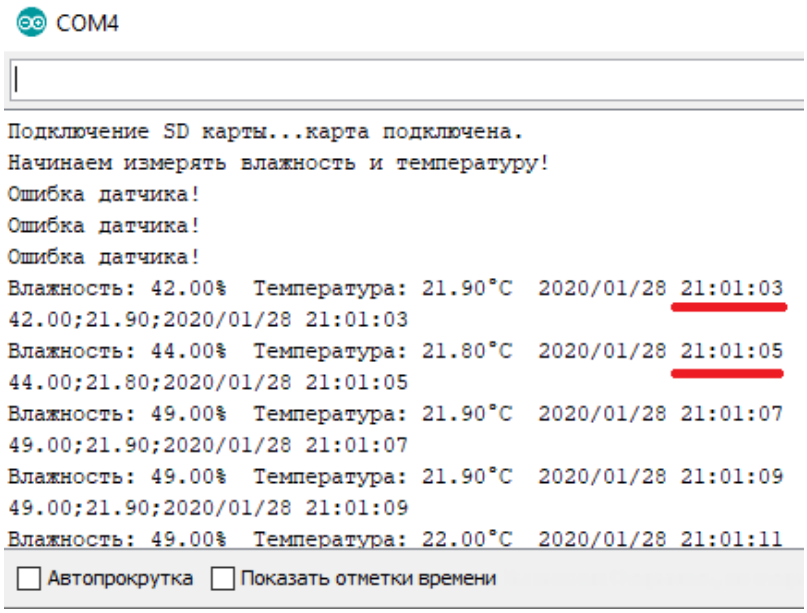
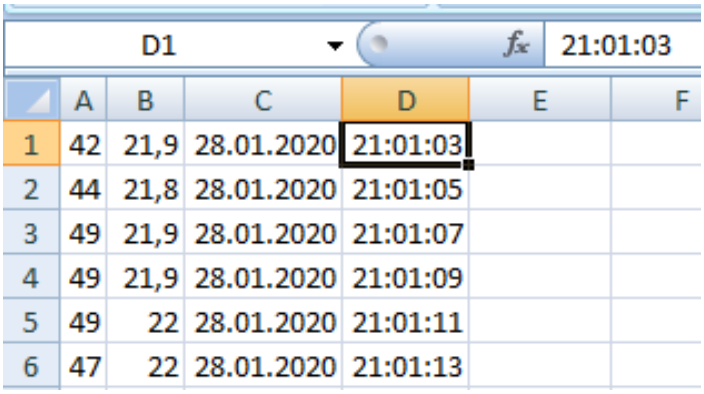


Рис. 28. Отметки даты и времени в Мониторе порта.

На рисунке 29 показаны данные, импортированные с карты памяти. Обратите внимание, что время записано в отдельной ячейке Excel. Для этого при импорте данных нужно указать второй тип разделителя столбцов – пробел.



D1 | 21:01:03

	A	B	C	D	E	F
1	42	21,9	28.01.2020	21:01:03		
2	44	21,8	28.01.2020	21:01:05		
3	49	21,9	28.01.2020	21:01:07		
4	49	21,9	28.01.2020	21:01:09		
5	49	22	28.01.2020	21:01:11		
6	47	22	28.01.2020	21:01:13		

Рис. 29. Данные с карты памяти импортированные в MS Excel.

ЗАКЛЮЧЕНИЕ

Курсовую работу необходимо выполнять по принципу «от простого – к сложному», двигаясь маленькими шажками к цели работы. Сначала необходимо добиться устойчивого выполнения кода примеров. Разобравшись в каждой строчке кода примера, переходить к его адаптации к заданию. Следует максимально полно использовать циклы и функции.

Оформление работы должно соответствовать государственным требованиям, предъявляемым к оформлению документации [7-10]. Внимательно прочитайте следующие строки и следуйте им при оформлении курсовой работы.

Содержание курсовой работы отражает ход мысли и способности студента. Оформление работы демонстрирует отношение студента к процессу обучения, преподавателю и лично к себе. Все разделы работы последовательно раскрывают этапы достижения цели. В работе содержится только информация, непосредственно связанная с этапами работы. Все иллюстрации раскрывают содержание выполненных действий. Текстовые пояснения помогают студенту отвечать на вопросы на защите, а не порождают новые уточняющие вопросы. Выводы по работе содержат личные впечатления студента и констатируют достижение цели работы, а не перечисляют очевидные факты. Переписывать теорию из учебника и интернета в курсовую работу не нужно. Из текста пояснительной записки должно быть видно, какие новые навыки приобретены студентом. Время, затраченное на выполнение курсовой работы, не является критерием оценки. Удачи!

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ Р ИСО 8756-2005. Качество воздуха. Обработка данных по температуре, давлению и влажности. – М.: Стандартинформ, 2007. – 8 с.

2. *Воробьева, Ф.И.* Информатика. MS Excel 2010 : учебное пособие / Ф.И. Воробьева, Е.С. Воробьев ; Министерство образования и науки России, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Казанский национальный исследовательский технологический университет». - Казань: Издательство КНИТУ, 2014. - 100 с.: ил. - ISBN 978-5-7882-1657-7; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428798 (дата обращения 26.02.2018).

3. *Блум Д.* Изучаем Arduino: инструменты и методы технического волшебства / Джереми Блум; [перевод с английского В. Петина]. - Санкт-Петербург: БХВ-Петербург, 2017. - 336 с.

4. *Монк С.* Программируем Arduino: профессиональная работа со скетчами / Саймон Монк; [пер. с англ. А. Киселев]. – Санкт-Петербург [и др.]: Питер, 2017. - 272 с.

5. Arduino сайт на русском для начинающих мастеров ардуино [Электронный ресурс]. – URL: https://arduinomaster.ru/#1508255432022-58d0f83f-1be8 (дата обращения 22.01.2020).

6. Библиотека SPI для Arduino [Электронный ресурс]. – URL: https://radioprog.ru/post/245 (дата обращения 22.01.2020).

7. SPI Arduino - подключение устройств к платам ардуино [Электронный ресурс]. – URL: https://arduinomaster.ru/datchiki-arduino/podklyuchenie-spi-arduino/ (дата обращения 22.01.2020).

8. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2018. – 33 с.

9. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М.: Стандартинформ, 2007. – 32 с.

10. ГОСТ Р 7.0.5-2008. Библиографическая ссылка. Общие требования и правила составления. . – М.: Стандартинформ, 2008. – 23 с.

СОДЕРЖАНИЕ

ИНФОРМАТИКА

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ТЕРМОГИГРОМЕТРА НА БАЗЕ ARDUINO

Методические указания к курсовой работе для студентов бакалавриата направления 05.03.06

Сост.: *О.В. Косарев, Е.Г. Дементьева, Т.В. Сарапулова*

Печатается с оригинал-макета, подготовленного кафедрой информатики и компьютерных технологий

Ответственный за выпуск *О.В. Косарев*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 06.02.2023. Формат 60×84/16.
Усл. печ. л. 2,3. Усл.кр.-отт. 2,3. Уч.-изд.л. 2,0. Тираж 50 экз. Заказ 75.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2