**Устройство измерения температуры на базе ESP32 и датчика DS18B20**

Устройство предназначено для измерения температуры воздуха и передачи данных через интерфейс Wi-Fi. Конструктивно устройство состоит из датчика DS18B20, контроллера на базе ESP32, литий-полимерного аккумулятора, разъема mic12-3 и корпуса.

**Датчик DS18B20** предназначен для измерения температуры. Датчик имеет влагозащищенный корпус, что позволяет использовать его в условиях дождя или же для измерения температуры жидкостей. Возможно также исполнение без герметичного корпуса.

**Характеристики:**

1. Питание: 3 … 5,5 В постоянного тока
2. Диапазон измеряемых температур: -55 … +125 °C от -10 до +85°C с точностью  ±0,5°C
3. Период измерений: 1 сек. между измерениями
4. Выходной сигнал: Цифровой

Датчик DS18B20 имеет 3 вывода. В скобках указаны контакты, которые были использованы для подключения датчика к плате контроллера.

**GND** – "Земля" (GND)

**Vdd** – Питание (3.3)

**Data** – вывод данных (14)

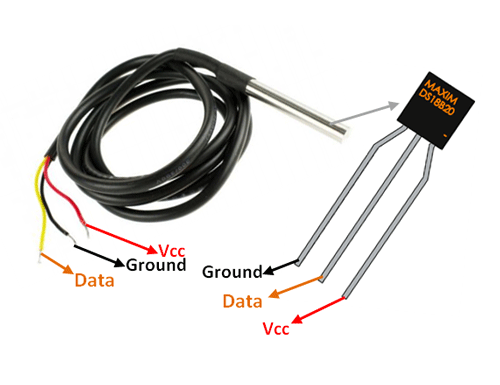
[](https://i2.wp.com/www.bizkit.ru/wp-content/uploads/2019/03/DS18B20-Pinout.png)

Рисунок . Датчик DS18B20

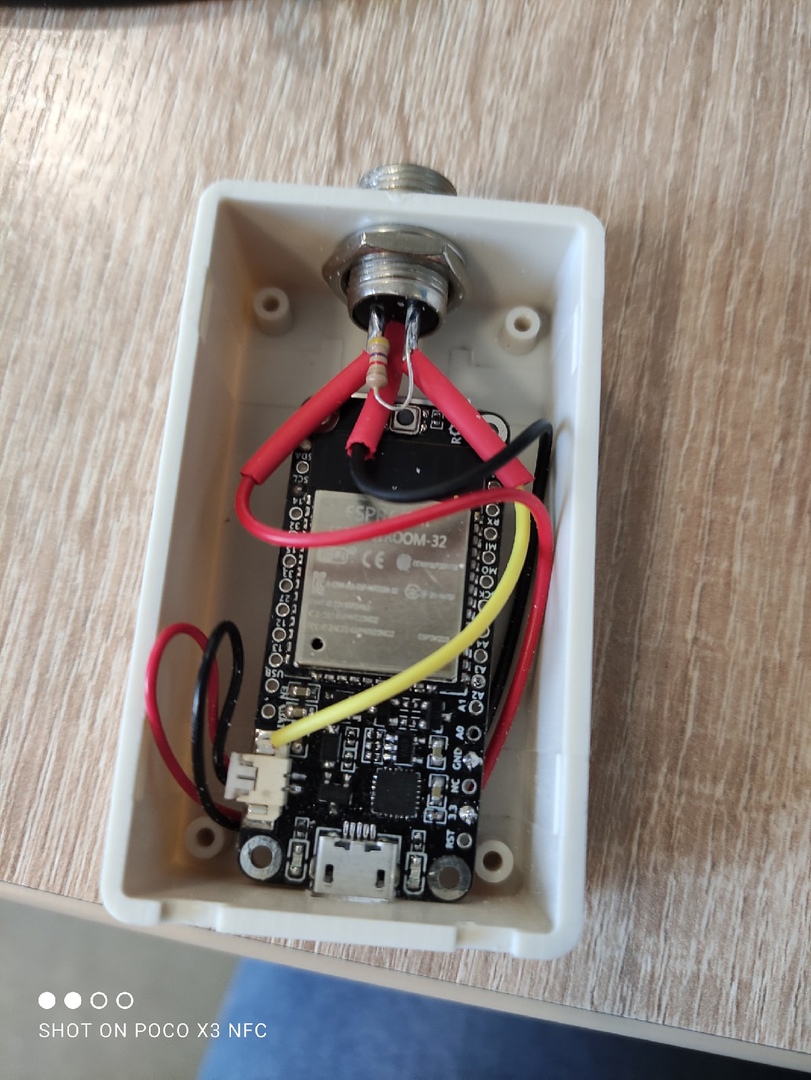


Рисунок . Плата контроллера в корпусе

**Подключение к Arduino:**

Для подключения датчика нам необходимы:

1. Плата с контроллером ESP32 или ESP8266 (В нашем случае используется плата Графит-32 на основе ESP-WROOM-32)

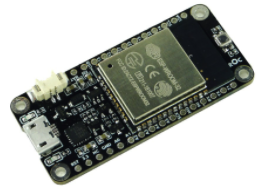


Рисунок . Плата Графит-32

1. Резистор 4,7 КOм
2. Компьютер с установленной средой Arduino IDE.
3. USB кабель для подключения Arduino к персональному компьютеру

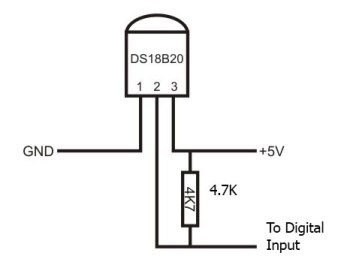


Рисунок . Принципиальная схема подключения датчика DS18B20

По условию технического задания необходимо измерить температуру, сформировать JSON-файл и отправить файл с информацией на сервер. Для начала необходимо подключить плату. Чтобы установить ESP32 в среду Arduino IDE, выполните следующее.

1) Откройте окно настроек в среде Arduino IDE. Выберите пункт меню «Файл > Настройки» («File > Preferences»).

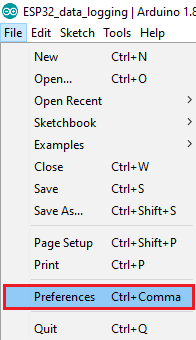


Рисунок . Меню file в среде Arduino IDE

2) В поле «Дополнительные ссылки для Менеджера плат» (Additional Boards Manager URLs) скопируйте адрес.Затем нажмите кнопку «OK».

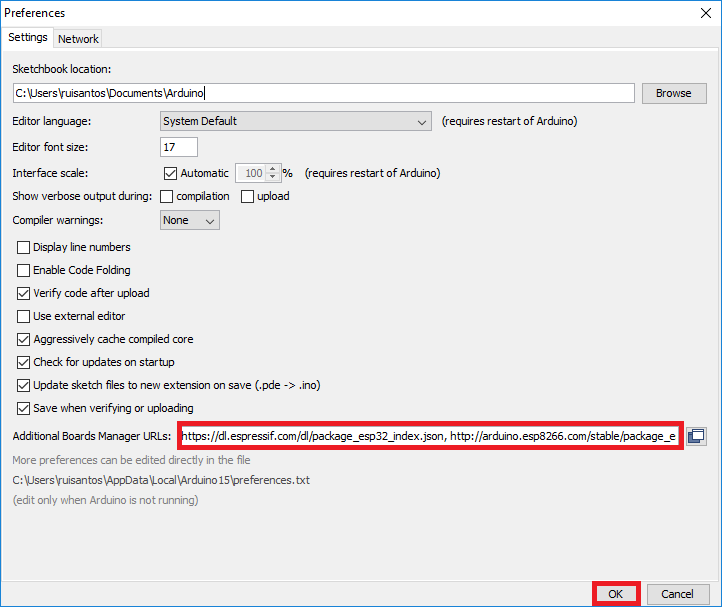


Рисунок . Меню настроек Arduino IDE

Примечание: если у вас уже есть ссылка для плат с чипом ESP8266, то вы можете разделить ссылки запятой следующим образом:

https://dl.espressif.com/dl/package\_esp32\_index.json, http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

3) Откройте менеджер плат. Нажмите «Инструменты > Плата > Менеджер плат» («Tools > Boards > Boards Manager»).

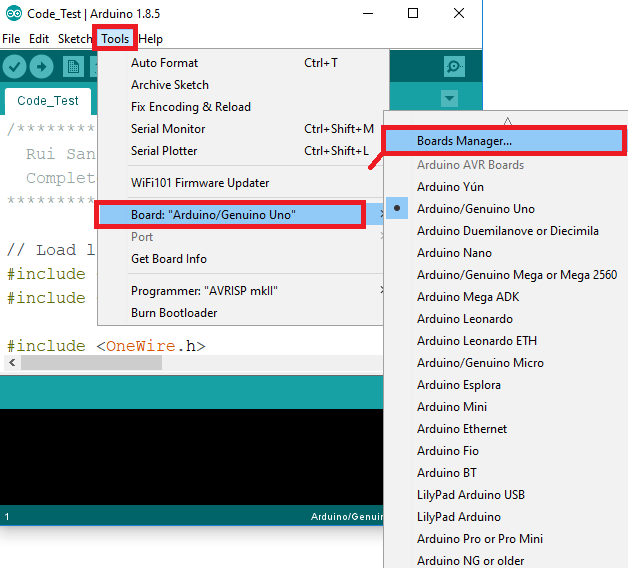


Рисунок . Выбор используемого контроллера в среде Arduino IDE

4) Введите в поиске «ESP32» и нажмите кнопку «Установка» (Install) для «ESP32 by Espressif Systems».

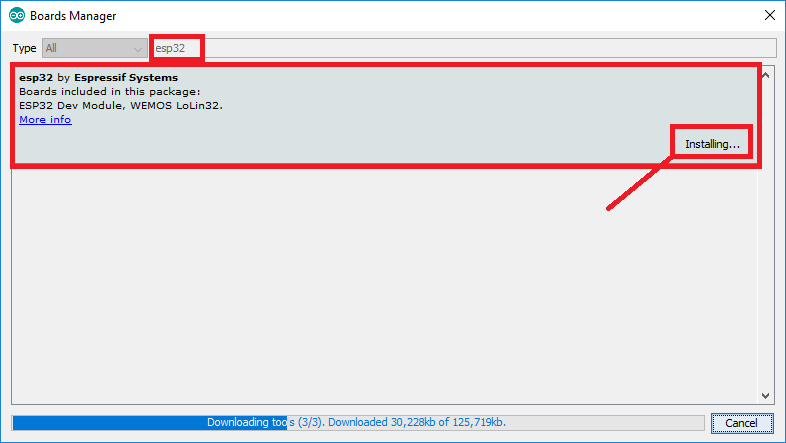


Рисунок . Установка дополнения в среде Arduino IDE

5) Готово. Дополнение должно установиться за несколько секунд.

**ПРОВЕРКА УСТАНОВКИ**

Подключите плату ESP32 к компьютеру. Затем выполните следующее:

1) Запустите среду Arduino IDE.

2) Пройдите в меню «Инструменты > Плата» («Tools > Board») и выберите свою плату (в нашем случае это DOIT ESP32 DEVKIT V1)

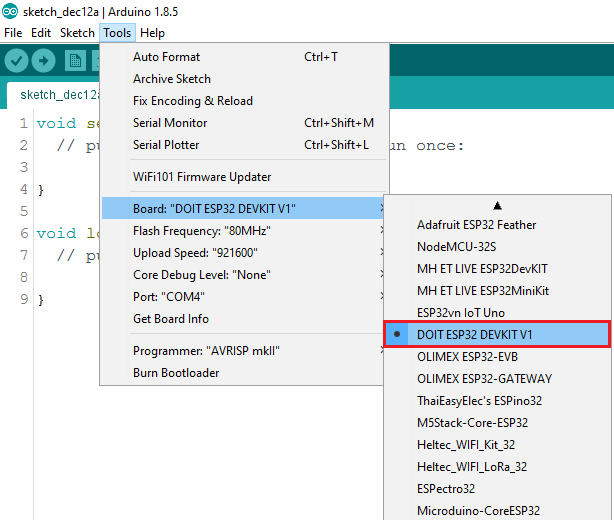


Рисунок .Выбор используемого контроллера в среде Arduino IDE

3) Выберите порт в меню «Порт» («Port») (если вы не видите COM-порт в среде Arduino IDE, то вам необходимо установить драйвер для адаптера CP210x USB to UART Bridge VCP).

4) Откройте следующий пример «Файл > Примеры > WiFi (ESP32) > WiFi Scan» («File > Examples > WiFi (ESP32) > WiFi Scan»).

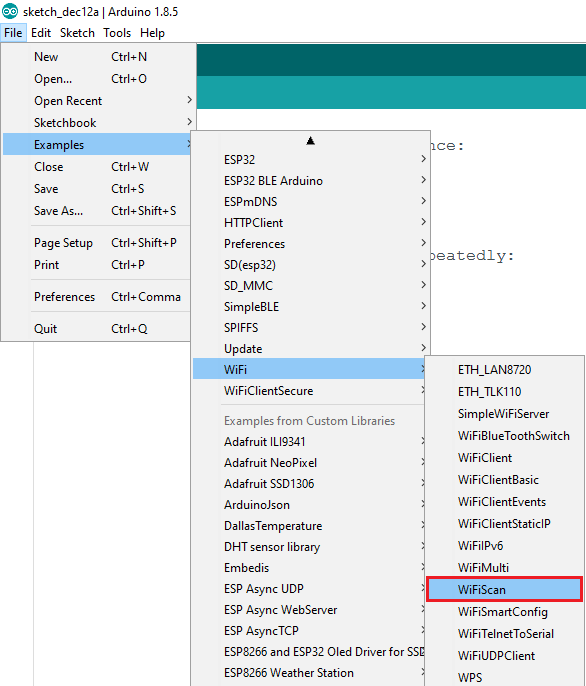


Рисунок . Выбор примера для проверки работоспособности контроллера

5) Откроется новый готовый пример.

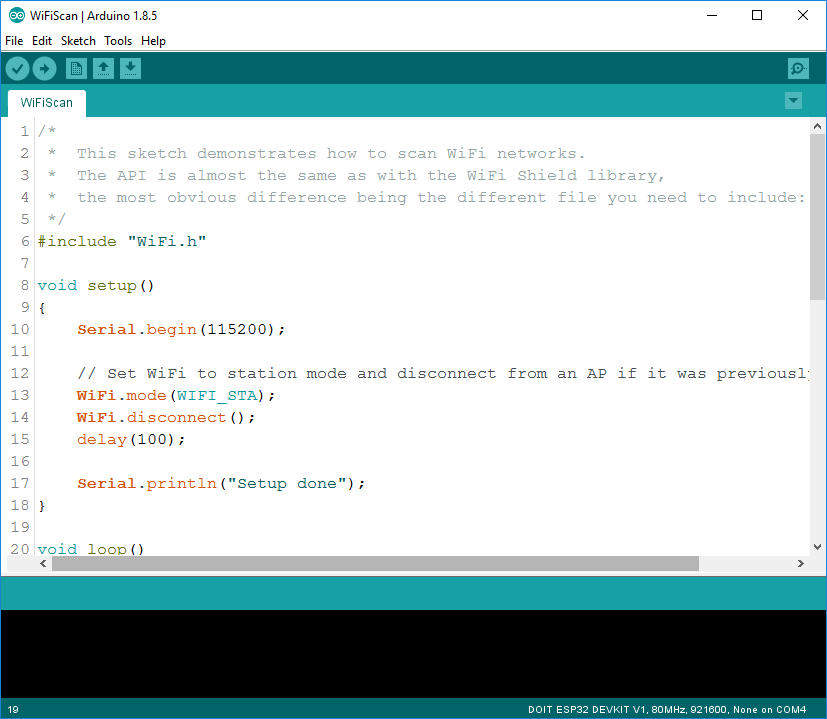


Рисунок . Тестовая программа для контрллера ESP32

6) Нажмите кнопку «Загрузка» («Upload») в среде Arduino IDE. Подождите несколько секунд, пока код компилируется и загружается в плату.

7) Если всё прошло, как ожидается, то вы должны увидеть сообщение «Done uploading» (Загрузка завершена).

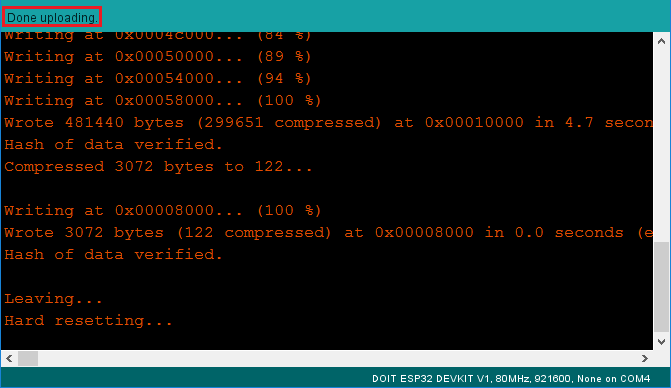


Рисунок . Вывод терминала при загрузке тестовой программы

8) Откройте окно последовательного COM-порта (Serial Monitor) среды Arduino IDE и настройте скорость передачи на 115 200 бод.

9) Нажмите кнопку «Enable» на плате с микроконтроллером ESP32, после этого вы должны увидеть сети, доступные для ESP32.

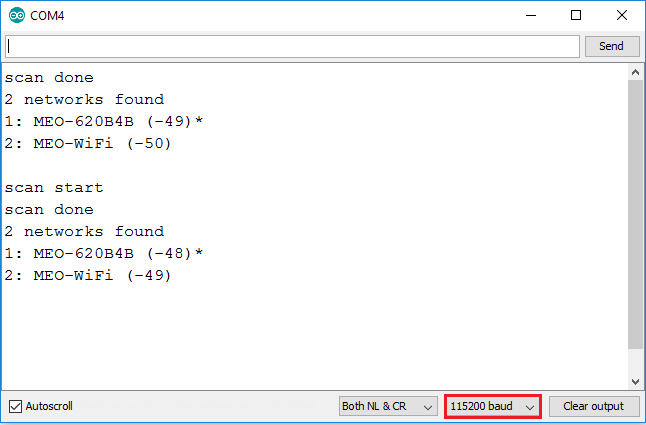


Рисунок . Сообщения, выведенные в монитор последовательного порта контроллера

# Исходный код программы для контроллера:

Для считывания показаний с датчика необходимо использовать библиотеки\*:

1. ArduinoJson.h
2. WiFi.h
3. HTTPClient.h
4. base64.h
5. OneWire.h
6. DallasTemperature.h

Для этого переходим в Инструменты Управлять библотеками

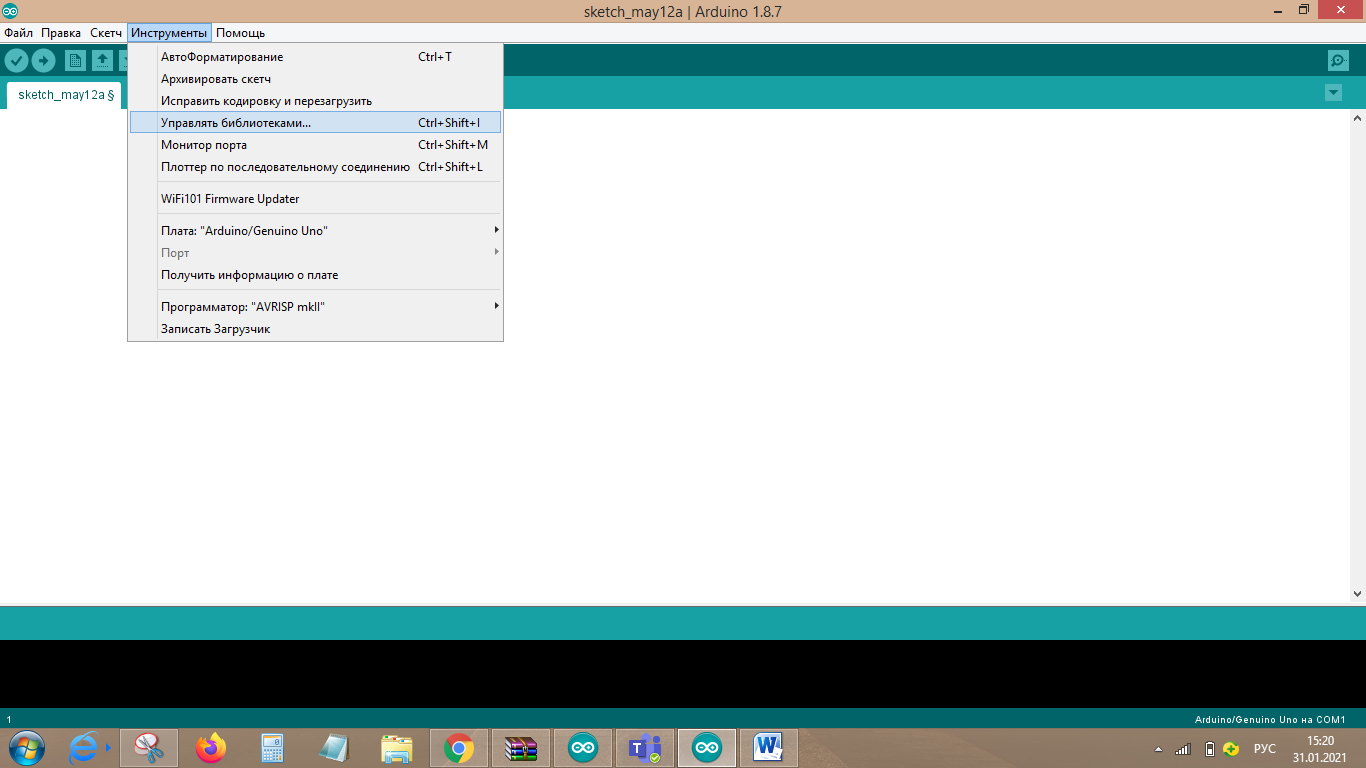


Рисунок . Включение библиотек в исходный код программы

Далее выбираем указанные выше библиотеки и устанавливаем

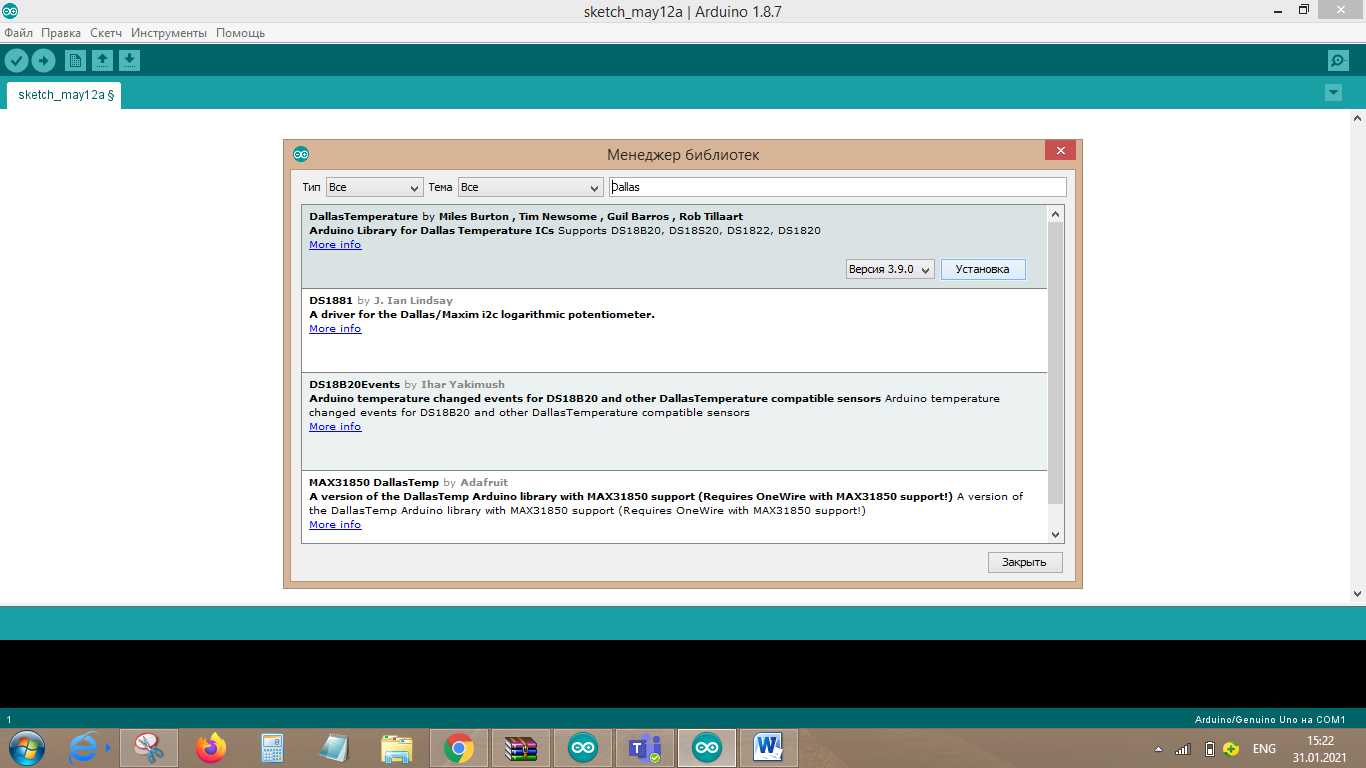


Рисунок . Установка библиотек в среде Arduino IDE

\*Библиотека Arduino – программный код, хранящийся не в скетче, а во внешних файлах, которые можно подключить к вашему проекту. Библиотека содержит в себе различные методы и структуры данных, которые нужны для упрощения работы с датчиками, индикаторами, модулями и другими электронными компонентами.

Ниже представлен пример кода измерения температуры, формирования Json-файла для передачи на сервер. Необходимым условием является прохождение аутентификации на сервере.

**Исходный код программы:**

[#include](https://vk.com/im?sel=84880029&st=%23include) <DallasTemperature.h>  
[#include](https://vk.com/im?sel=84880029&st=%23include) <OneWire.h>  
[#include](https://vk.com/im?sel=84880029&st=%23include) <ArduinoJson.h>  
[#include](https://vk.com/im?sel=84880029&st=%23include) <WiFi.h>  
[#include](https://vk.com/im?sel=84880029&st=%23include) <HTTPClient.h>  
[#include](https://vk.com/im?sel=84880029&st=%23include) <base64.h>  
  
String login = "HESUserOne";  
String authPass = "HESUserOne";  
String auth = base64::encode(login + ":" + authPass);  
const char\* ssid = "UrFU-Guest"; //имя точки  
const char\* password = ""; //пароль  
const int oneWireBus = 14; // GPIO куда подключен DS18B20  
OneWire oneWire(oneWireBus); // Объявление OneWire для связи с устройствами  
DallasTemperature sensors(&oneWire); // Пробрасываем OneWire к датчику  
String conn = "http://188.226.32.141/records";  
String con2 = "http://18.216.254.223/records";  
void setup() {  
Serial.begin(115200); //запускаем монитор порта  
sensors.begin(); //запускаем датчик  
delay(4000);  
WiFi.begin(ssid, password);  
  
while (WiFi.status() != WL\_CONNECTED) {  
delay(1000);  
Serial.println("Connecting to WiFi..");  
}  
  
Serial.println("Connected to the WiFi network");  
  
}  
  
void loop() {  
HTTPClient http;  
HTTPClient http2;  
String a = "";  
// put your main code here, to run repeatedly:  
sensors.requestTemperatures(); //обновляем температуру  
float temperatureC = sensors.getTempCByIndex(0); //запрашиваем и пишем в переменную  
Serial.print("Температура: ");  
Serial.println(temperatureC); //выводим в консоль  
DynamicJsonDocument doc(128);  
  
doc["latitude"] = 56.842135;  
doc["longitude"] = 60.651898;  
  
JsonObject values = doc.createNestedObject("values");  
values["temperature"] = temperatureC;  
values["pressure"] = 0;  
values["humidity"] = 0;  
doc["deviceName"] = "NMTM\_102701";  
doc["dateTime"] = "";  
serializeJson(doc, a);  
http.begin(conn);  
http.addHeader("Content-Type", "application/json");  
http.addHeader("Authorization: Basic " , auth);  
int httpCode = [http.POST](https://vk.com/away.php?to=http%3A%2F%2Fhttp.POST&cc_key=" \t "_blank)(a);  
String payload = http.getString(); //Get the response payload  
Serial.println(httpCode); //Print HTTP return code  
Serial.println(payload); //Print request response payload  
//Ждем 2 сек.  
delay (2000);  
//потом отключаемся  
http.end();  
delay(5000);  
http2.begin(con2);  
http2.addHeader("Content-Type", "application/json");  
http2.addHeader("Authorization: Basic " , auth);  
httpCode = [http2.POST](https://vk.com/away.php?to=http%3A%2F%2Fhttp2.POST&cc_key=)(a);  
payload = http2.getString(); //Get the response payload  
Serial.println(httpCode); //Print HTTP return code  
Serial.println(payload); //Print request response payload  
//Ждем 2 сек.  
delay (2000);  
//потом отключаемся  
http2.end();  
delay(5000);  
//delay(598000); //задержка в 598 секунд  
  
}

В результате мы получаем значение температуры, а также ответ от сервера, что аутентификация пройдена успешна.

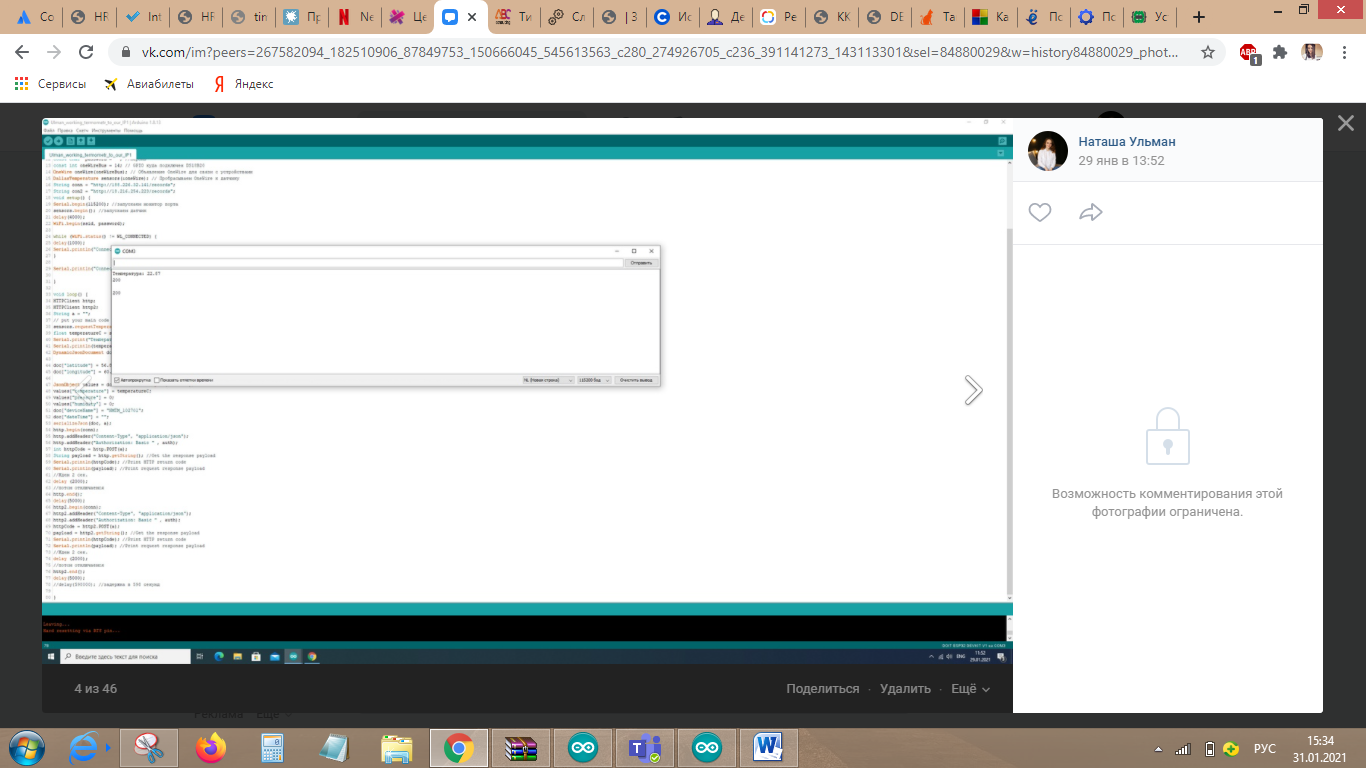


Рисунок . Вывод сообщения в последовательный порт при корректной работе контроллера