***Иванов Лев. С любовью и душой.***

***P.S. Надеюсь не закидаете палками. Если будут вопросы, пишите. Почта и ссылки на соцсети уже есть у вас на сайте)0***

**NodeMCU ESP8266 Мониторинг данных с датчика DHT11/DHT22 с помощью локального WEB-сервера**

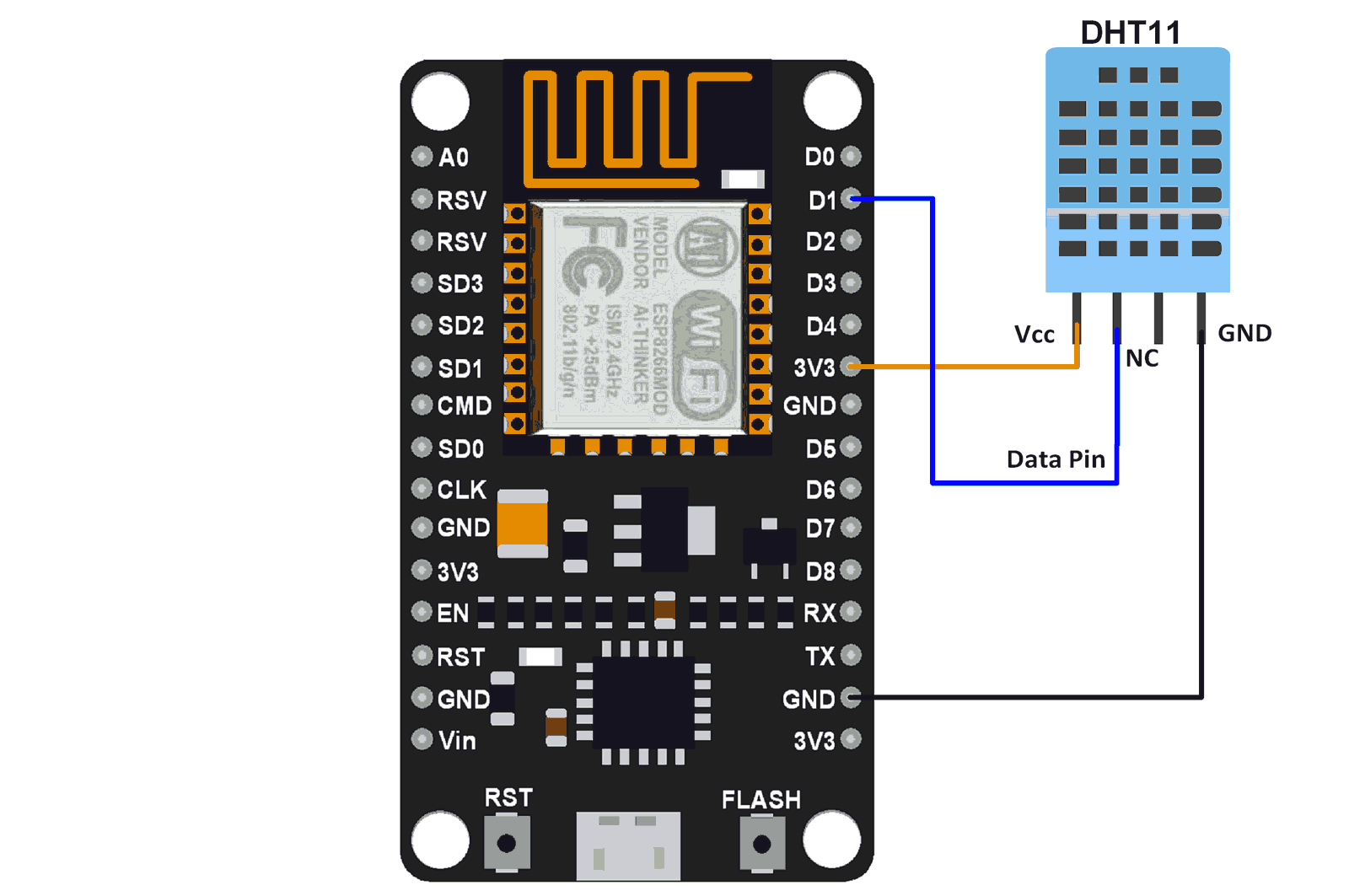
Автономный web-сервер с использование NodeMCU ESP8266, отображающий температуру и влажность с помощью датчика DHT11/DHT22 при помощи Arduino IDE. К веб серверу, который мы создадим, можно будет легко получить доступ с любого устройства через браузер в локальной сети.

В данном проекте используется асинхронный web-сервер, который автоматически обновляет температуру и влажность. Для обновления данных не нужно постоянно обновлять страницу. Для стилизации web-страницы используется собственный CSS.

**Компоненты:**

1. NodeMCU ESP8266;
2. Датчик температуры и влажности DHT11 или DHT22;
3. Макетная плата;
4. Провода.

**Принципиальная схема**

****

| **NodeMCU ESP8266** | **DHT** |
| --- | --- |
| D1 (GPIO5) | Data Pin |
| 3V3 | Vcc |
| GND | GND |

**Организация файлов**

Для создания web-сервера понадобятся два файла. Первый - скетч Arduino, второй - файл HTML разметки. Файл HTML должен быть сохранен в папке data текущего проекта Arduino.

Для этого создайте проект в среде разработки, создайте файл с index.html (для этого открываете текстовый документ и сохраняете его с расширением .html). Далее перейдите в Arduino IDE Скетч -> Добавить файл… и выберите свой только что сохраненный файл index.html.

Чтобы убедиться, что файл добавлен к проекту, Вы можете перейти в сам файл проекта и в нем должна появиться папка data, а в ней файл index.html.

**Создание файла HTML**

В файл index.html добавьте следующий код:

<!DOCTYPE HTML><html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>lioncloud.Weather</title>

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">

<script src="https://code.highcharts.com/highcharts.js"></script>

<style>

body {

height: 100px;

margin: 0 auto;

background: #111;

}

h2 {

font-family: Arial;

font-size: 2.5rem;

text-align: center;

}

p { font-size: 3.0rem; }

html {

font-family: Arial;

display: inline-block;

margin: 0px auto;

text-align: center;

}

p { font-size: 3.0rem; }

.units { font-size: 1.2rem; }

.dht-labels{

font-size: 1.5rem;

vertical-align: middle;

padding-bottom: 15px;

}

span{

color: white;

}

sup{

color: white;

}

table {

width: 100%;

border: 0;

}

td.header {

background-color: #111;

}

td.sidebar {

background-color: #111;

width: 30%;

vertical-align: top;

padding:5px;

}

td.sidebar>ul {

list-style: none;

padding:5px;

}

td.article {

background-color: #111;

height: 200px;

width: 70%;

vertical-align: top;

padding:5px;

}

td.footer {

background-color: #111;

text-align: center;

padding:5px;

}

</style>

</head>

<body>

<table border="0">

<tr>

<td colspan="2" class="header">

<h2>

<font face="trendy" style="color: #00add6;" size="7">

lioncloud.Weather

</font>

</h2>

</td>

</tr>

<tr>

<td class="sidebar">

<b face="trendy" style="color: #ffffff;">At the moment</b>

<p>

<i class="fas fa-thermometer-half" style="color:#059e8a;"></i>

<span class="dht-labels">Temperature</span>

<span id="temperature" style="color:#059e8a;"><output id="temp"></output></span>

<span class="units">&#8451;</span>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i>

<span class="dht-labels">Humidity</span>

<span id="humidity" style="color:#00add6;"><output id="hum"></output></span>

<span class="units">&#37;</span>

</p>

<script src="https://apps.elfsight.com/p/platform.js" defer></script>

<div class="elfsight-app-5b934c68-b5da-4f5c-b49a-d9c683d8fb35"></div>

</td>

<td class="article">

<script src="https://apps.elfsight.com/p/platform.js" defer></script>

<div class="elfsight-app-d6fca2b5-1e7d-40eb-a6c4-49e5b3d0e536"></div>

</td>

</tr>

<tr>

<td colspan="2" class="footer">

<div style="width: 1000px;height: 500px; display: inline-block;" id="chart-temp" class="container"></div>

</td>

</tr>

<tr>

<td colspan="2" class="footer">

<div style="width: 1000px;height: 500px; display: inline-block;" id="chart-hum" class="container"></div>

</td>

</tr>

</table>

</body>

<script>

var chartT = new Highcharts.Chart({

chart:{ renderTo : 'chart-temp' },

title: { text: 'Real-time temperature' },

series: [{

showInLegend: false,

data: []

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: { type: 'datetime',

dateTimeLabelFormats: { second: '%H:%M:%S' }

},

yAxis: {

title: { text: 'Temperature' }

},

credits: { enabled: false }

});

var chartH = new Highcharts.Chart({

chart:{ renderTo : 'chart-hum' },

title: { text: 'Real-time humidity' },

series: [{

showInLegend: false,

data: []

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: { type: 'datetime',

dateTimeLabelFormats: { second: '%H:%M:%S' }

},

yAxis: {

title: { text: 'Humidity' }

},

credits: { enabled: false }

});

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

var x = (new Date()).getTime(),

y = parseFloat(this.responseText);

var temp = this.responseText;

document.getElementById('temp').innerHTML = temp;

if(chartT.series[0].data.length > 1000) {

chartT.series[0].addPoint([x, y], true, true, true);

} else {

chartT.series[0].addPoint([x, y], true, false, true);

}

}

};

xhttp.open("GET", "/temperature", true);

xhttp.send();

}, 90000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

var x = (new Date()).getTime(),

y = parseFloat(this.responseText);

var hum = this.responseText;

document.getElementById('hum').innerHTML = hum;

if(chartH.series[0].data.length > 1000) {

chartH.series[0].addPoint([x, y], true, true, true);

} else {

chartH.series[0].addPoint([x, y], true, false, true);

}

}

};

xhttp.open("GET", "/humidity", true);

xhttp.send();

}, 90000 ) ;

</script>

</html>

В приведенном выше коде можно увидеть, что есть раздел <script></script>. С помощью него будет написан в программном коде JavaScript для красивой графики.

<script>

var chartT = new Highcharts.Chart({

chart:{ renderTo : 'chart-temp' },

title: { text: 'Real-time temperature' },

series: [{

showInLegend: false,

data: []

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: { type: 'datetime',

dateTimeLabelFormats: { second: '%H:%M:%S' }

},

yAxis: {

title: { text: 'Temperature' }

},

credits: { enabled: false }

});

var chartH = new Highcharts.Chart({

chart:{ renderTo : 'chart-hum' },

title: { text: 'Real-time humidity' },

series: [{

showInLegend: false,

data: []

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: { type: 'datetime',

dateTimeLabelFormats: { second: '%H:%M:%S' }

},

yAxis: {

title: { text: 'Humidity' }

},

credits: { enabled: false }

});

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

var x = (new Date()).getTime(),

y = parseFloat(this.responseText);

var temp = this.responseText;

document.getElementById('temp').innerHTML = temp;

if(chartT.series[0].data.length > 1000) {

chartT.series[0].addPoint([x, y], true, true, true);

} else {

chartT.series[0].addPoint([x, y], true, false, true);

}

}

};

xhttp.open("GET", "/temperature", true);

xhttp.send();

}, 90000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

var x = (new Date()).getTime(),

y = parseFloat(this.responseText);

var hum = this.responseText;

document.getElementById('hum').innerHTML = hum;

if(chartH.series[0].data.length > 1000) {

chartH.series[0].addPoint([x, y], true, true, true);

} else {

chartH.series[0].addPoint([x, y], true, false, true);

}

}

};

xhttp.open("GET", "/humidity", true);

xhttp.send();

}, 90000 ) ;

</script>

Новый объект Highcharts сформирован для отображения в теге <div> с идентификатором расстояния диаграммы в HTML-коде. Ось X определяется как время считывания данных, ось Y - значение считывания, которое представляет собой температуру и влажность. Затем функция setInterval будет отправлять данные сервера каждые 1.5 минуты (90000 мс.), обращаясь к конечной точке “/temperature” и “/humidity”, предоставленной сервером, и отправляя данные в объект графика.

Строки var temp = this.responseText и var hum = this.responseText записывают текущие показания датчика в переменные, для того, чтобы в дальнейшем вывести их на экран.

**Установка библиотек**

1. DHT для NodeMCU ESP8266

Для чтения данных с датчика DHT нужно воспользоваться библиотекой Adafruit Unified Sensor.

Чтобы установить эту библиотеку откройте Arduino IDE и перейдите в Скетч -> Подключить библиотеку -> Управлять библиотеками. Найдите библиотеку введя в поиске “DHT”. Установите ее.

Установите еще одну библиотеку введя в поле поиска Adafruit Unified Sensor.

После установки библиотек перезапустите Arduino IDE.

1. Установка библиотеки асинхронного сервера ESPAsyncWebServer

Чтобы установить библиотеку ESPAsyncWebServer перейдите по [ссылке](https://github.com/me-no-dev/ESPAsyncWebServer) и скачайте ее.

В папке “Загрузки” появится файл с расширением .zip. Перейдите в Arduino IDE -> Скетч -> Подключить библиотеку -> Добавить .ZIP библиотеку. Выберите свой скачанный файл и нажмите установить.

1. Установки библиотеки TCP ESPAsync

Библиотека ESPAsyncWebServer требует, чтобы библиотека TCP ESPAsync работала исправно. Установите библиотеку скачав файл по этой [ссылке](https://github.com/me-no-dev/ESPAsyncTCP). В папке “Загрузки” появится файл с расширением .zip. Перейдите в Arduino IDE -> Скетч -> Подключить библиотеку -> Добавить .ZIP библиотеку. Выберите свой скачанный файл и нажмите установить.

Перезапустите Arduino IDE.

**Код программы**

Так как в данном проекте используется плата NodeMCU ESP8266, необходимо установить менеджер плат ESP8266 в Arduino IDE. Для этого перейдите в Arduino IDE Файл -> Настройки и вставьте следующую ссылку <http://arduino.esp8266.com/stable/package_esp8266com_index.json> в строку “Дополнительные ссылки для Менеджера плат”. Перезапустите Arduino IDE.

// Import required libraries

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <Hash.h>

#include <ESPAsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <Adafruit\_Sensor.h>

#include <DHT.h>

const char\* ssid = "YOUR\_SSID";

const char\* password = "YOUR\_PASSWORD";

#define DHTPIN 5 // Digital pin D1 connected to the DHT sensor

// Uncomment the type of sensor in use:

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT22 // DHT 22 (AM2302)

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()

float t = 0.0, h = 0.0;

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

unsigned long previousMillis = 0; // will store last time DHT was updated

// Updates DHT readings every 10 seconds

const long interval = 10000;

void setup(){

// Serial port for debugging purposes

Serial.begin(115200);

dht.begin();

if (! SPIFFS.begin ()) {

Serial.println ("An Error has occurred while mounting SPIFFS");

return;

}

// Connect to Wi-Fi

WiFi.begin(ssid, password);

Serial.println("Connecting to WiFi");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println(".");

}

Serial.println(WiFi.localIP());

// Route for root / web page

server.on ("/", HTTP\_GET, [] (AsyncWebServerRequest \* request) {

request-> send (SPIFFS, "/index.html");

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", String(t).c\_str());

});

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", String(h).c\_str());

});

// Start server

server.begin();

}

void loop(){

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

// save the last time you updated the DHT values

previousMillis = currentMillis;

// Read temperature as Celsius (the default)

float newT = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

//float newT = dht.readTemperature(true);

// if temperature read failed, don't change t value

if (isnan(newT)) {

Serial.println("Failed to read from DHT sensor!");

}

else {

t = newT;

Serial.println(t);

}

// Read Humidity

float newH = dht.readHumidity();

// if humidity read failed, don't change h value

if (isnan(newH)) {

Serial.println("Failed to read from DHT sensor!");

}

else {

h = newH;

Serial.println(h);

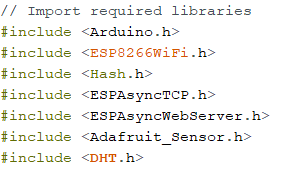
}

}

}

**Объяснение кода**

В приведенной программе включаются библиотеки.

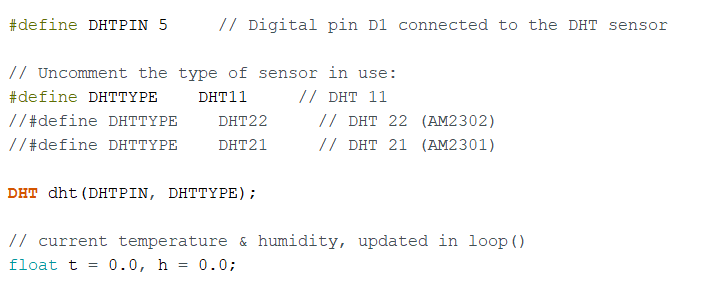


Далее нужно ввести имя своей точки доступа и пароль, для того, чтобы сервер смог подключиться к локальной сети.

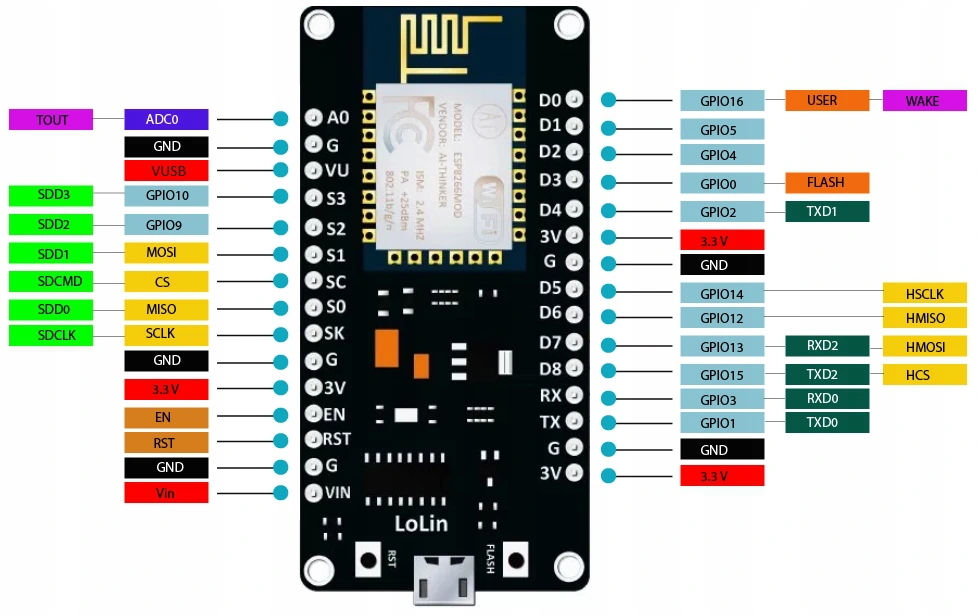


В переменную ssid введите имя своей точки доступа, в переменную password - пароль от нее.

Следующий блок, это настройка датчика температуры и влажности.



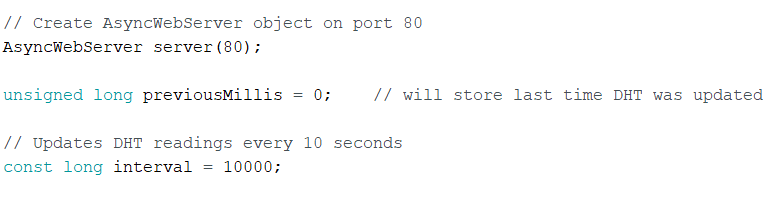
Задаем цифровой пин, на который подключен датчик к ESP8266. В данном примере датчик подключен к пину D1, если заглянуть в документацию контроллера можно определить, что это порт ввода/вывода GPIO5. Следовательно #define DHTPIN 5.



Определяем тип датчика DHT11/DHT22/DHT21, в зависимости от того, какой вы собираетесь использовать, раскомментируйте строку.

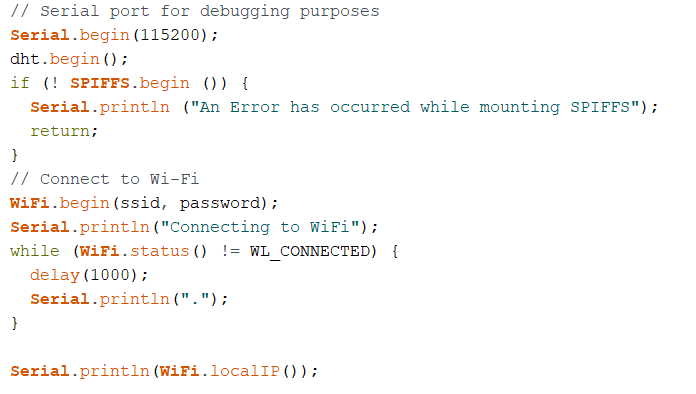
Создаем две переменные типа float для входных значений температуры (t) и влажности (h), приравниваем их к нулю.

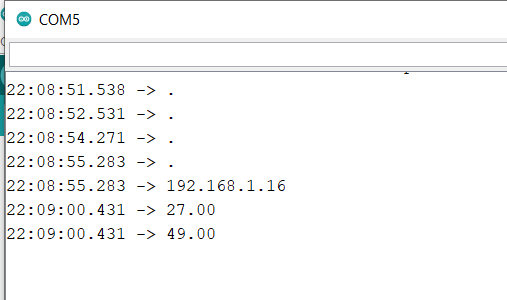
Создадим объект асинхронного сервера на порту 80. И несколько переменных для работы со временем.



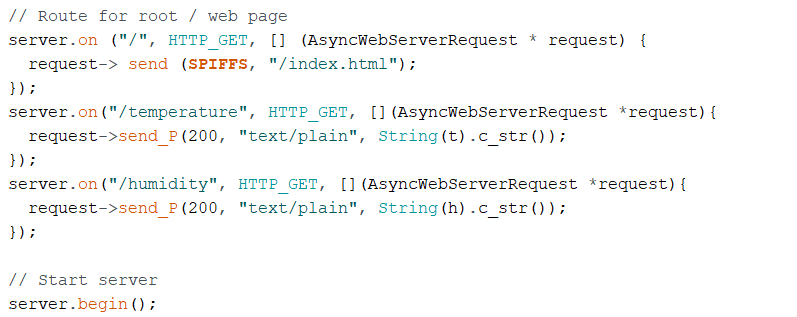
В функции setup настраиваем COM-порт на скорость 115200 бод. Инициализируем датчик температуры и влажности. Ставим проверку запуска SPIFFS, если не запустился возвращаем ошибку.

Следующим шагом инициализируем подключение к точке доступа WiFi. Пока идет подключение в монитор последовательного порта будут выводиться точки, после подключения на экране появится IP-адрес контроллера.

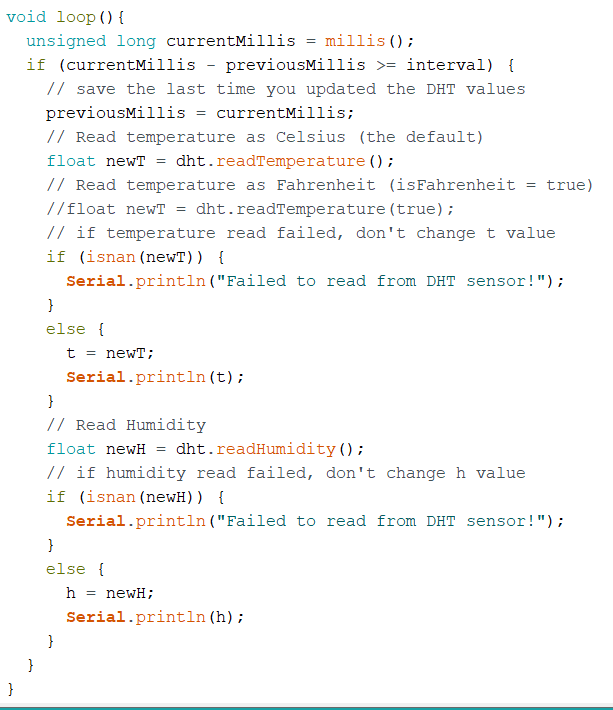




Определяем маршрут, обслуживаемый сервером. В данном проекте определены три маршрута. Один корневой и два маршрута с данными. Корневой маршрут вернет страницу index.html, а маршруты temperature и humidity вернут значения данных с датчика в формате Strings. Запускаем сервер.



В функции loop создаем беззнаковую переменную типа long и записываем в нее время, пройденное с момента запуска контроллера, в миллисекундах. Ставим условие, если текущее время минус время обновления датчика в последний раз больше заданного интервала, то приравниваем времени обновления данных с датчика к текущему времени. Создаем две переменных для новых значений датчика. Записываем показания датчика в переменные newT и newH. Если датчик не определился, выводим сообщение об ошибки, иначе записываем только что снятые показания датчика в глобальные переменные t и h.

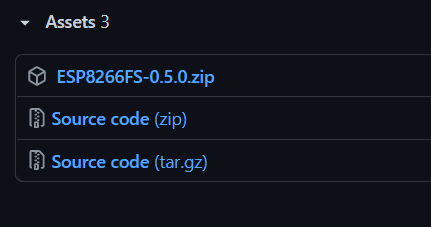


**Загрузка кода и HTML-файла**

1. Установка загрузчика файловой системы Arduino ESP8266

Установим плагин для Arduino IDE, который позволяет загружать файлы прямо в файловую систему ESP8266 из папки на вашем компьютере. Это упростит работу с файлами.

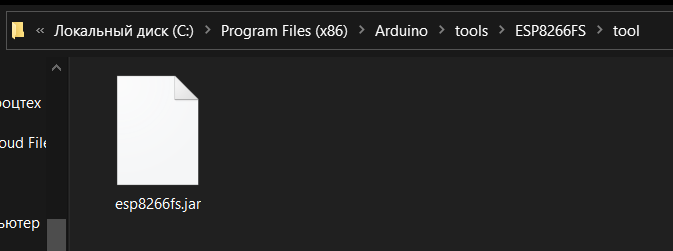
Перейдите на сайт <https://github.com/esp8266/arduino-esp8266fs-plugin/releases> и скачайте файл ESP8266FS-X.zip



Перейдите в каталог Arduino IDE (обычно он находится в папке документы) и откройте папку Tools.



Разархивируйте загруженную папку .zip в папку Tools . У вас должна быть аналогичная структура папок: C:\Program Files (x86)\Arduino\tools



Перезапустите Arduino IDE.

Чтобы проверить, успешно ли был установлен плагин, откройте Arduino IDE и выберите плату ESP8266. В меню «Инструменты» убедитесь, что у вас есть опция «ESP8266 Sketch Data Upload ».

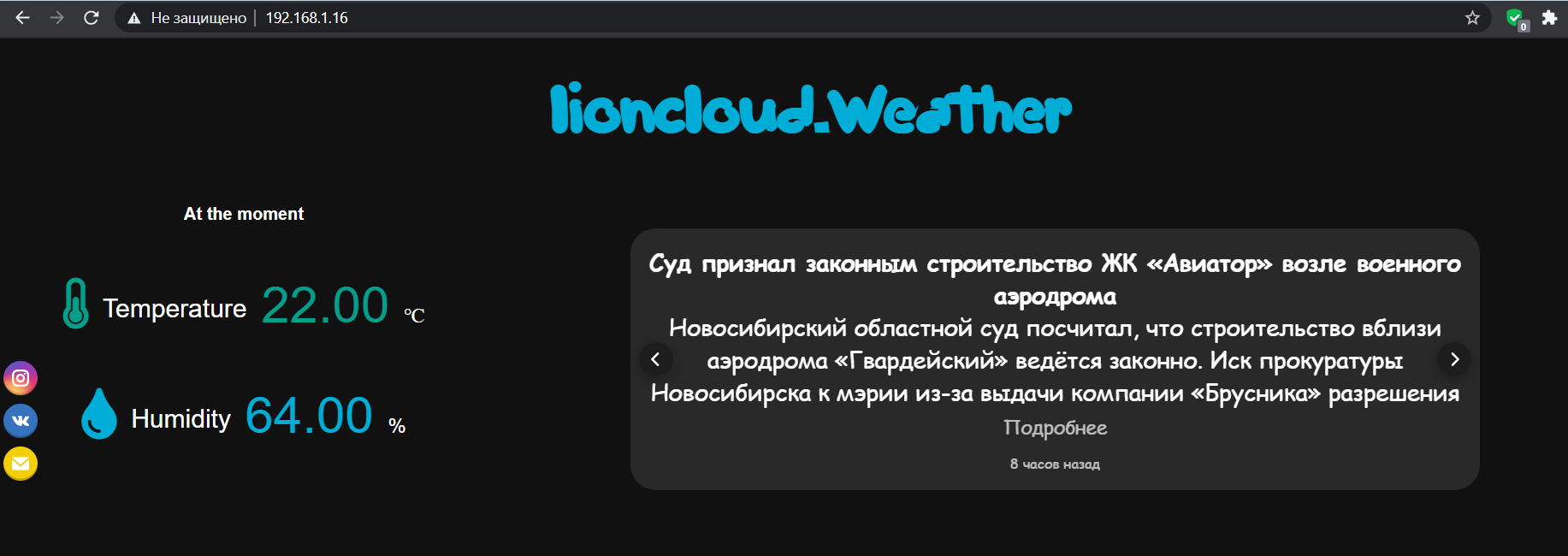
1. Теперь вам нужно загрузить HTML-файл в файловую систему ESP8266 . Перейдите в Инструменты -> ESP8266 Data Sketch Upload и дождитесь загрузки файлов.

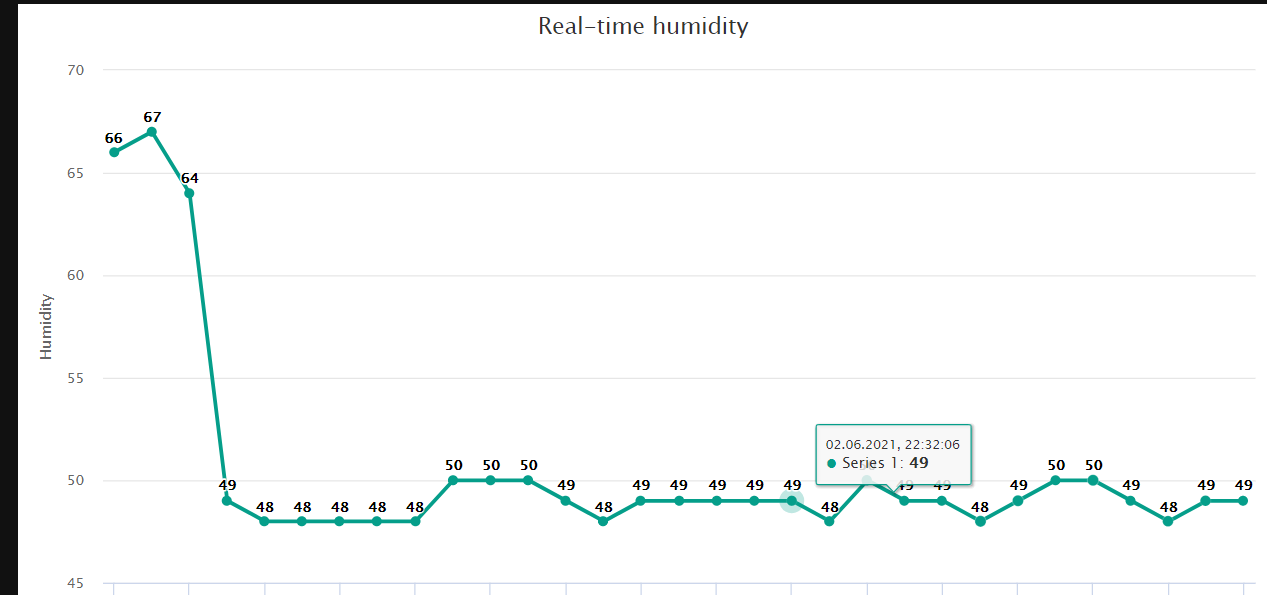
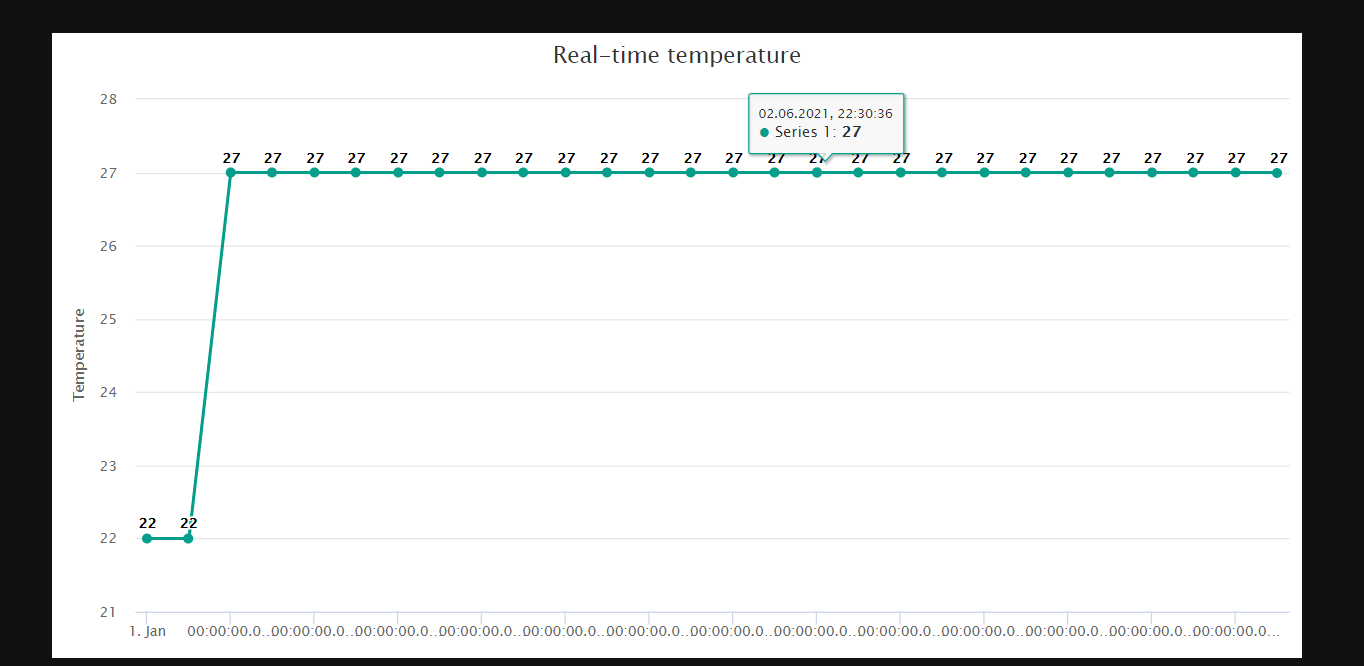
Затем загрузите код на свою плату NodeMCU. Убедитесь, что вы выбрали правильную плату и COM-порт. Кроме того, убедитесь, что вы ввели корректно свои учетные данные WiFi в код.

После успешной загрузки откройте Serial Monitor со скоростью 115200 бод. Выберите «EN/RST» на плате ESP8266. В последовательном порту появится IP-адрес контроллера.

1. Откройте браузер и введите IP-адрес ESP8266. Вы должны увидеть графики и текущие показания датчика. Первые показания появятся спустя 1.5 минуты. Новая точка данных добавляется каждые 1.5 минуты, всего 1000 точек. Пока вкладка вашего браузера открыта, данные постоянно отображаются на графиках.

Пример графиков температуры и влажности: вы можете выбрать каждую точку, чтобы увидеть точную временную метку.





***Если вы дошли до конца и у вас всё заработало, поздравляю, можете пойти отдохнуть...***