

**КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**  
**ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИ-**  
**ОННЫХ ТЕХНОЛОГИЙ**

*Кафедра системного анализа и информационных технологий*

**К.С. Николаев**

**ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕ-**  
**РОВ ESP8266 И ARDUINO NANO/UNO/MEGA В**  
**СРЕДЕ ARDUINO IDE**

**Казань – 2022**

**УДК 004.021**  
**ББК ББК ШЗ(4)**

*Принято на заседании кафедры системного анализа и информационных технологий Казанского (Приволжского) федерального университета  
Протокол № 7 от 13 апреля 2022 года*

**Николаев К.С.**

**Программирование микроконтроллеров ESP8266 и Arduino Nano/Uno/Mega в среде Arduino IDE / К.С. Николаев. – Казань: Казан. ун-т, 2022. – 15 с.**

В данном пособии представлен набор практических работ, направленных на изучение основных понятий и способов программирования микроконтроллеров Arduino Nano/Uno/Mega, а также микроконтроллеров ESP8266 (более современная версия контроллера с меньшим количеством входов/выходов, но с большим количеством оперативной и flash-памяти и подключением к беспроводной Wi-Fi сети).

Пособие адресовано прежде всего студентам специальностей «Фундаментальная информатика и информационные технологии», «Прикладная информатика», а также техническим специальностям Института физики.

© Николаев К.С., 2022  
© Казанский университет, 2022

## ВВЕДЕНИЕ

В наше время микроконтроллеры применяются в качестве основных или дополнительных модулей для управления логикой работы различных технических устройств (от радиоуправляемых машин до космических кораблей).

Начиная с 2005 года, микроконтроллеры стали доступны для широкого круга энтузиастов, в связи с выходом первых версий микроконтроллера Arduino [1]. Низкая цена, программирование на языке C++ [2] и удобная обвязка входов/выходов контроллера позволила Arduino стать образцом для подражания среди массово доступных микроконтроллеров.

За годы существования торговой марки Arduino было выпущено множество версий микроконтроллеров, отличающихся размерами, количествами входов/выходов, количеством доступной памяти и, соответственно, сферами применения.

Помимо Arduino, в 2014 году огромную популярность набрал микроконтроллер ESP8266[3] от Espressif Systems. Преимуществом этого контроллера является увеличенная оперативная память (128 Кбайт), флеш-память (4 Мбайт), встроенный Wi-Fi модуль. Недостатками контроллера является отсутствие документации на внутреннюю периферию микроконтроллера<sup>1</sup>.

В данном пособии предлагается формулировка и реализация нескольких проектов в среде Arduino IDE, направленных на выполнение на контроллерах Arduino и ESP8266. Проекты имеют бытовую направленность, и могут реализованы читателями на собственных микроконтроллерах. При необходимости производился уход от профессиональной терминологии для упрощения понимания.

---

<sup>1</sup> <https://ru.wikipedia.org/wiki/ESP8266>

В лабораторной работе 1 студенты познакомятся с основной терминологией микроконтроллеров, и научатся выполнять базовые операции на микроконтроллере – чтение напряжения с контактов микроконтроллера, подача напряжения. Также студенты узнают базовые методы языка программирования C++, используемого при написании кода контроллера.

В лабораторной работе 2 студенты познакомятся с возможностью подключения сторонних библиотек в среду разработки на примере библиотеки ArduinoJson, научатся отправлять и принимать данные через серийный порт в формате json. [5]

В лабораторной работе 3 студенты познакомятся с новым видом микроконтроллеров – ESP8266, и создадут свой веб-сервер, принимающий команды на управление подключенной периферией из любого веб-браузера.

## **СОКРАЩЕНИЯ И ТЕРМИНЫ**

Прежде чем приступать к описанию лабораторных работ, введем определения и сокращения, которые будут использоваться далее по тексту:

1. МК -сокр. микроконтроллер
2. Пин - контакт МК, с помощью которого он может отправлять или считывать данные.
3. ШИМ - широтно-импульсная модуляция (см. лабораторную работу 1) [4]
4. LED - светодиод
5. Потенциометр – электрическое сопротивление с переменным номиналом.
6. Парсинг – выделение данных из строкового представления в определенном формате.

## ЛАБОРАТОРНАЯ РАБОТА 1. БАЗОВЫЕ ОПЕРАЦИИ НА МИКРОКОНТРОЛЛЕРЕ ARDUINO

Работа с любым МК представляет из себя ни что иное, как чтение данных с входных пинов, выполнение определенных действий с оперативной памятью компьютера (если таковая имеется) и вывод определенного напряжения на выходные пины.

К примеру, если Вы захотите отрегулировать яркость лампочки накаливания с помощью микроконтроллера, то самый простой способ – это использовать связку «МК – потенциометр». На рисунке 1 приведена схема подключения.

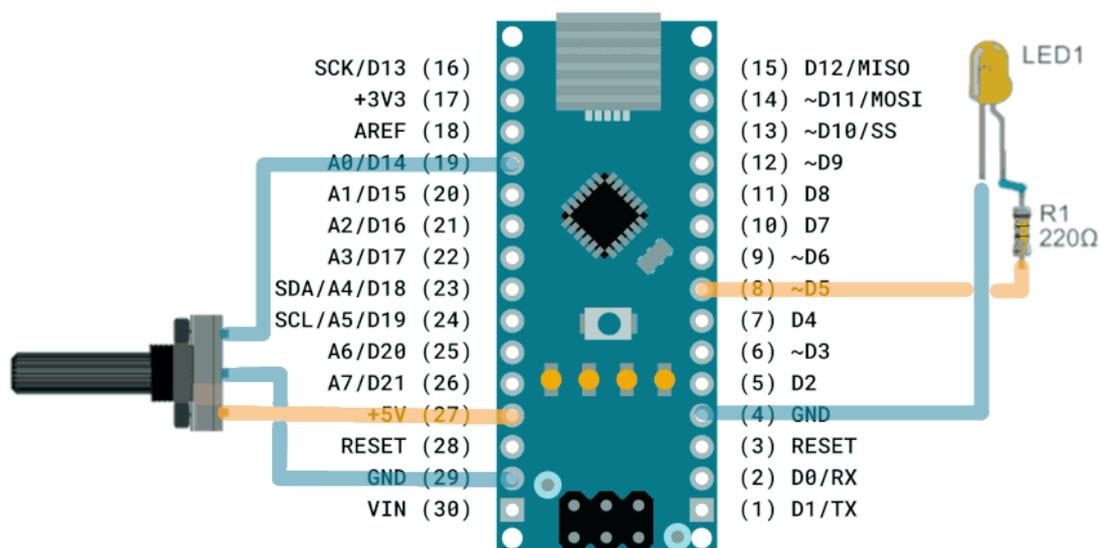


Рисунок 1. Схема подключения светодиода и потенциометра к Arduino Nano

Напряжение, подающееся на пин 5V, проходит через потенциометр, понижается по закону Ома, и отправляется на пин A0 (входной пин для чтения точного значения напряжения от 0 до 5 вольт). Это напряжение необходимо считать в коде микроконтроллера, и отправить соответствующее значение на пин D6.

Далее приведен код МК, реализующий данную задачу.

```

//Метод setup выполняется один раз при подаче питания на МК или при перезагрузке МК
void setup() {
  //Устанавливаем пин D5 в режим "выход"
  pinMode(5, OUTPUT)
}

// Метод loop повторяется в течение всего времени работы МК
void loop() {
  //Читаем значение с пина A0 (он всегда находится в режиме входа)
  int sensorValue = analogRead(A0);
  //Конвертируем значение из диапазона [0,1023] в диапазон [0,255]
  float voltage = sensorValue / 4;
  analogWrite(5, voltage);
}

```

### Рисунок 2. Код МК, управляющий яркостью светодиода

Стоит заметить, что подача напряжения на пин со светодиодом осуществляется с помощью ШИМ.

ШИМ – это подход к понижению напряжения, использующий

Еще одним способом взаимодействия МК с внешним миром, помимо подачи и чтения напряжения с пинов, является серийный порт. Серийный порт позволяет отправлять массивы байтов с помощью 4 пинов, зарезервированных для этой цели, либо через USB-подключение (что гораздо более удобно).

В Arduino IDE есть встроенный монитор порта, который позволяет считывать данные, отправленные МК, и отправлять команды в ответ. Дополним код с предыдущего рисунка так, чтобы вольтаж отправлялся не только на LED, но и в серийный порт.

Для этого необходимо инициализировать серийный порт в методе setup(), установив при этом частоту опроса порта. Такое же значение должно быть установлено в настройках монитора серийного порта или любого другого устройства, пытающего прочитать данные от МК. (Рисунок 3)

```

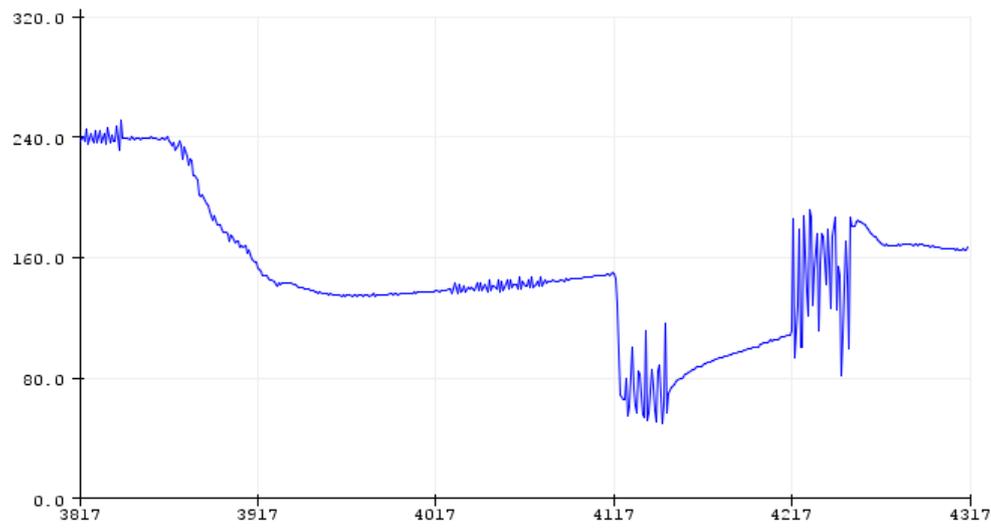
//Метод setup выполняется один раз при подаче питания на МК или при перезагрузке МК
void setup() {
  //Устанавливаем пин D5 в режим "выход"
  pinMode(5, OUTPUT);
  //Инициализируем серийный порт с частотой 9600 единиц информации в секунду
  Serial.begin(9600);
}

// Метод loop повторяется в течение всего времени работы МК
void loop() {
  //Читаем значение с пина A0 (он всегда находится в режиме входа)
  int sensorValue = analogRead(A0);
  //Конвертируем значение из диапазона [0,1023] в диапазон [0,255]
  float voltage = sensorValue / 4;
  analogWrite(5, voltage);
  //Отправляем данные по серийному порту
  Serial.print(voltage);
}

```

**Рисунок 3. Код МК, отправляющий напряжение с потенциометра в серийный порт**

На заметку: если запустить код на МК, к которому не подключен потенциометр, то в мониторе порта будут выводиться случайные данные, так как пин A0 будет снимать электромагнитные наводки окружающей среды (Рисунок 4).



**Рисунок 4. Визуализация электромагнитных наводок на контакте A0**

Подключите, запустите, проверьте, что всё работает.

## ЛАБОРАТОРНАЯ РАБОТА 2. РАБОТА С БИБЛИОТЕКАМИ В ARDUINO IDE

Еще одним огромным плюсом Arduino является наличие большого числа встроенных и сторонних библиотек, упрощающих обработку данных или содержащих набор удобных методов для работы с различными устройствами.

В этой лабораторной работе предлагается отправить с МК и принять в МК строку в формате json, используя библиотеку ArduinoJson. В качестве отправляемых данных будут использоваться данные с того же пина A0 (условно случайные данные). Отправка будет производиться пачками по 10 пар, содержащих количество секунд, прошедших с запуска микроконтроллера, и показание с пина A0.

Установка библиотеки происходит с помощью менеджера библиотек (запускается с помощью сочетания клавиш Ctrl+Shift+I). Нужная библиотека приведена на рисунке 5.

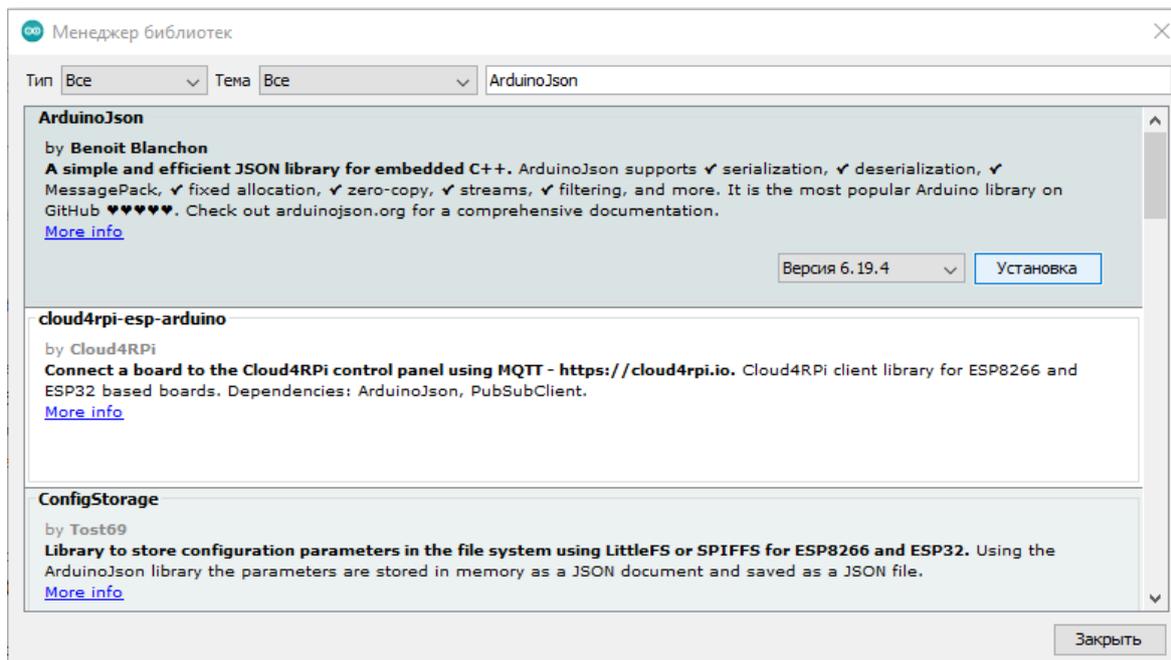


Рисунок 5. Установка библиотеки ArduinoJson в Arduino IDE

Полный код с комментариями приведен на рисунке 6. Обратите внимание, что при добавлении пары значений в json-документ производится вызов

конструктора String. Если передать значение миллисекунд в ключ без приведения к строке, то библиотека решит, что мы передаем ей индекс записи в документе, а не название ключа.

Кроме того, между замерами напряжения добавлена пауза в 50 миллисекунд, чтобы значения функции millis() отличались для разных измерений. В остальном передача данных в формате json очень проста. При необходимости, на официальном сайте<sup>2</sup> доступна подробная документация библиотеки.

```
//подключаем библиотеку
#include <ArduinoJson.h>
//Создаем объект для работы с json
DynamicJsonDocument doc(1024);
//Метод setup выполняется один раз при подаче питания на МК или при перезагрузке МК
void setup() {
  //Инициализируем серийный порт с частотой 9600 единиц информации в секунду
  Serial.begin(9600);
}

// Метод loop повторяется в течение всего времени работы МК
void loop() {

  for (int i =0;i<10;i++)
  {
    //с помощью встроенной функции millis() получаем текущее время МК
    long seconds = millis();

    //Читаем значение с пина A0 (он всегда находится в режиме входа)
    int sensorValue = analogRead(A0);

    //добавляем в документ json пару значений
    doc[String(seconds)] = sensorValue;

    //ждем 50 мс, чтобы у ключей были разные значения
    delay(50);
  }
  //после добавления 10 пар чисел, отправляем данные в серийный порт.
  serializeJson(doc, Serial);
  Serial.println();
  Serial.println();
  //очищаем документ json для следующей итерации
  doc.clear();
}
```

### Рисунок 6. Код МК, отправляющий по серийному порту данные о вольтаже в формате json

Перейдем к следующей части работы, а именно приеме и парсинге json-строки на МК от внешнего устройства. Строку заранее подготовим на рабочем компь-

---

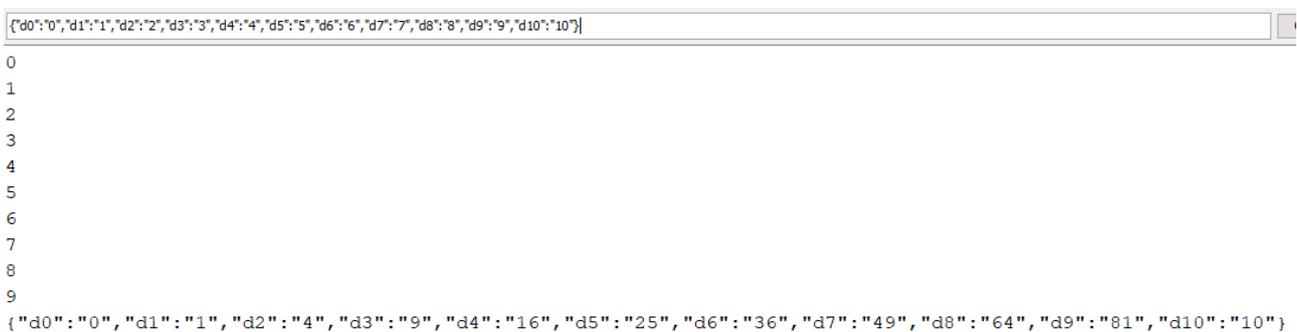
<sup>2</sup> <https://arduinojson.org/>

ютере и отправим через монитор серийного порта. Кроме того, при успешном получении данных мы отправим ответ в виде квадратов полученных значений. Код приведен на рисунке 7.

```
void loop() {  
  
    String s = Serial.readString();  
    DeserializationError error = deserializeJson(doc, s); //десериализуем json  
    if (!error)  
    {  
        for (int i = 0; i < 10; i++) //предполагаем, что мы знаем количество полученных данных  
        {  
            String key = "d" + String(i); //формируем ключ  
            int val = atoi(doc[key]); //получаем значение из json и конвертируем в int  
            doc[key] = String(val * val); //возводим в квадрат и пишем обратно в документ  
        }  
        serializeJson(doc, Serial);  
        Serial.println();  
        Serial.println();  
        //очищаем документ json для следующей итерации  
        doc.clear();  
    }  
}
```

**Рисунок 7. Код МК, принимающий по серийному порту данные в формате json и отправляющий ответ**

В качестве тестовой строки мы будем использовать строку, содержащую 4 пары ключ-значение {"d0": "0", "d1": "1", "d2": "2", "d3": "3", "d4": "4", "d5": "5", "d6": "6", "d7": "7", "d8": "8", "d9": "9", "d10": "10"}. Результат, принимаемый от микроконтроллера, должен быть равен {"d0": "0", "d1": "1", "d2": "4", "d3": "9", "d4": "16", "d5": "25", "d6": "36", "d7": "49", "d8": "64", "d9": "81", "d10": "10"}. На рисунке 8 приведен вывод монитора серийного порта, подтверждающий корректную работу микроконтроллера.



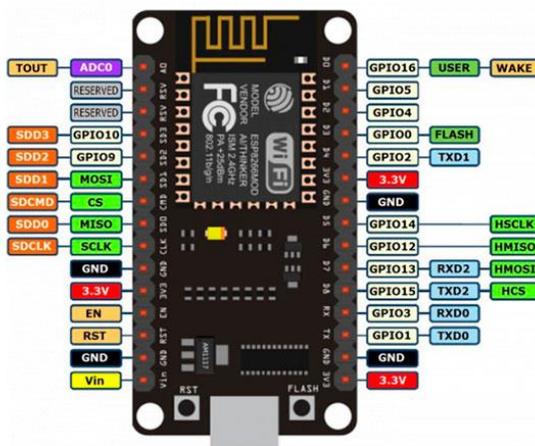
The screenshot shows a serial monitor window with a text input field at the top containing the JSON string: {"d0": "0", "d1": "1", "d2": "2", "d3": "3", "d4": "4", "d5": "5", "d6": "6", "d7": "7", "d8": "8", "d9": "9", "d10": "10"}. Below the input field, a list of numbers from 0 to 9 is shown, representing the indices of the data elements. At the bottom, the received JSON output is displayed: {"d0": "0", "d1": "1", "d2": "4", "d3": "9", "d4": "16", "d5": "25", "d6": "36", "d7": "49", "d8": "64", "d9": "81", "d10": "10"}.

**Рисунок 8. Отправленные и полученные данные**

## ЛАБОРАТОРНАЯ РАБОТА 3. ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ ESP8266

Перейдем к контроллерам ESP8266, представляющих большой интерес в бытовом плане. Их можно использовать в качестве устройств умного дома, так как подключение к интернету через беспроводное соединение, а также возможность развернуть простой веб-сервер прямо на МК позволяет отправлять команды и данные дистанционно и с помощью автоматизированных программ.

На рисунке 9 приведен внешний вид самой полной обвязки ESP8266, а также распиновка контактов.



**Рисунок 9. Внешний вид и распиновка ESP8266 (изображение получено из <https://clck.ru/h3Gmi>)**

Для работы с ESP8266 в Arduino IDE необходимо произвести дополнительную настройку среды разработки<sup>3</sup>.

В данной лабораторной работе будет реализован простой веб-сервер, выполняющий команды при переходе по определенным страницам в пределах сервера. К МК подключен светодиод по схеме, приведенной на рисунке 1 (на

---

<sup>3</sup> [https://wiki.iarduino.ru/page/WEMOS\\_start/](https://wiki.iarduino.ru/page/WEMOS_start/)

ESP8266 так же присутствует контакт GND, светодиод так же подключен к контакту D5).

Например, IP-адрес МК в локальной сети – 192.168.0.121. При переходе по адресу 192.168.0.121/ledon светодиод должен включиться. При переходе по адресу 192.168.0.121/ledoff светодиод должен выключиться. Также реализуем метод, плавно изменяющий яркость светодиода до заданного процента яркости. Процент яркости передадим в параметрах GET-запроса. Например, при переходе по адресу 192.168.0.121/set?value=25 яркость светодиода плавно изменится от текущего до 25% от максимальной яркости.

На рисунке 10 приведен метод setup(), содержащий настройки подключения к сети, создание и настройку веб-сервера.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

/* Тут необходимо указать название и пароль вашей домашней сети*/
const char* ssid = "Trenzalore";
const char* password = "65544105";
ESP8266WebServer server(80); //создание веб-сервера
uint8_t LED1pin = D5; //LED подключен к 5 пину
byte LED1status = 0; //стартовое состояние - выкл.
//Переменные для плавного изменения яркости
int start_val = 0;
int aim_val = 0;
int bright_step = 0;
bool changing = false;
void setup()
{
  pinMode(LED1pin, OUTPUT);
  Serial.begin(500000);
  WiFi.begin(ssid, password); //настраиваем подключение
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(1000);
    Serial.print(".");
  }
  Serial.println(WiFi.localIP());
  //WiFi.softAPConfig(local_ip, gateway, subnet);
  delay(100);
  //задаем методы, выполняемые при переходе по различным адресам.
  server.on("/", handle_OnConnect);
  server.on("/ledon", handle_ledon);
  server.on("/ledloff", handle_ledloff);
  server.on("/set", handle_set);
  server.begin(); //запуск сервера
}
```

**Рисунок 10. Стартовая настройка веб-сервера**

Метод `server.on` устанавливает соответствие между посещенным адресом и методом, который должен при этом запуститься.

Прежде, чем перейти к описанию методов, выполняемых при переходе на страницы, необходимо привести вспомогательный метод, формирующий html-разметку запрашиваемой страницы. В данном случае на странице будет отображаться текущее состояние светодиода (оно передается в параметры методу `SendHTML`). Код приведен на рисунке 11.

```
String SendHTML(uint8_t led1stat)
{
    String ptr = "<!DOCTYPE html> <html>\n";
    ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
    ptr += "<title>LED Control</title>\n";
    ptr += "</head>\n";
    ptr += "<body>\n";
    ptr += "<h1>ESP8266 Web Server</h1>\n";
    ptr += "<h3>Using Access Point(AP) Mode</h3>\n";
    if (led1stat)
        ptr += "<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</a>\n";
    else
        ptr += "<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">ON</a>\n";
    ptr += "</body>\n";
    ptr += "</html>\n";
    return ptr;
}
```

### Рисунок 11. Вспомогательный метод для формирования HTML-разметки

Методы `handle_led1on` и `handle_led1off` практически одинаковые, просто устанавливают состояние светодиода во включенное и выключенное соответственно. Код у них следующий:

```
{
    LED1status = HIGH; //для led1on
    LED1status = LOW; //для led1off
    server.send(200, "text/html", SendHTML(true));
}
```

Наиболее интересным в этой лабораторной работе является код плавного изменения яркости светодиода. Идея метода состоит в следующем. В отдельной переменной (`start_val`) храним текущее значение яркости. При вызове метода `handle_set` в переменную `aim_val` записываем целевое значение яркости. Вычисляем значение шага яркости, вычитая текущее значение яркости из целевого и

деля на некоторое  $N$ , от которого будет зависеть плавность изменения. Устанавливаем значение логической переменной `changing` в `True`. Изменения яркости будут происходить только тогда, когда эта переменная равна `True`. В методе `loop` проверяем, чему равна переменная `changing`. Если она равна `True`, прибавляем к `start_val` значение шага. Как только значение `start_val` выходит за пределы `aim_val`, устанавливаем `changing` в `False` (рисунки 12, 13).

```
void handle_set()//выполняется при переходе по адресу /set
{
  changing = true;
  aim_val = 255 * server.arg(0).toInt() / 100; //получаем значение первого параметра из запроса
  bright_step = (aim_val - start_val) / 20; //вычисляем значение шага
  server.send(200, "text/html", SendHTML(true));
}
```

**Рисунок 12. Настройка и запуск плавного изменения яркости**

```
void loop()
{
  server.handleClient(); //обрабатываем запросы от клиента
  if (changing)
  {
    start_val += bright_step;
    delay(20);
    if (bright_step > 0)//если увеличиваем яркость
    {
      if (start_val >= aim_val)
        changing = false;
    }
    else //если уменьшаем яркость
    {
      if (start_val <= aim_val)
        changing = false;
    }
  }
  Serial.println(start_val);
  if (LED1status)
    analogWrite(LED1pin, start_val);
  else
    analogWrite(LED1pin, 0);
}
```

**Рисунок 13. Плавное изменение яркости**

## СПИСОК ЛИТЕРАТУРЫ

1. <https://www.arduino.cc/> [Электронный ресурс]. URL: <https://www.arduino.cc/> (дата обращения: 16.05.2022).
2. [cppreference.com](https://en.cppreference.com/) [Электронный ресурс]. URL: <https://en.cppreference.com/w/> (дата обращения: 16.05.2022).
3. Espressif [Электронный ресурс]. URL: <https://www.espressif.com/en/products/socs/esp8266>
4. *Clark L.* Pulse Width Modulation [Электронный ресурс] // Controlling LEDs with Arduino : [сайт]. [2019]. URL: [https://link.springer.com/video/segment/10.1007/978-1-4842-4883-6\\_3](https://link.springer.com/video/segment/10.1007/978-1-4842-4883-6_3) (дата обращения: 16.05.2022)
5. [json.org](https://www.json.org/) [Электронный ресурс]. URL: <https://www.json.org/>