Плата D1 UNO R3 на основе ESP-8266 EX

Руководство

Содержание

1 О плате D1

- Основан на ESP-8266EX.
- Совместим с Arduino, программируется через Arduino IDE.
- 11 цифровых входов/выходов (I/O).
- 1 аналоговый вход (ADC) с диапазоном входного напряжения 0–3,3 В.
- Поддержка OTA-обновления по беспроводной сети.
- Встроенный импульсный источник питания 5 В 1 А (максимальное входное напряжение 24 В).
- Поддержка интерфейсов: SPI, I2C, 1-Wire и др.

Технические характеристики:

- Процессор: ESP-8266EX
- Рабочее напряжение: 3,3 В
- Входное напряжение: 7–24 В
- Количество I/O: 11
- Напряжение аналогового входа (AD): 0–3,3 В
- Flash-память: 4 МБ
- SRAM: 32 КБ
- DRAM: 80 КБ
- Тактовая частота: 80 МГц / 160 МГц
- Сеть: 802.11 b/g/n

Описание выводов:

Вывод	Внутренний вывод IC	Описание
D0 (RX)	GPIO3	Прием данных UART
D1 (TX)	GPIO1	Передача данных UART
D2	GPIO16	Ввод/вывод (I/O), не поддерживает прерывания, PWM, I2C и 1-Wire
D3/SCL/D15	GPIO5	Ввод/вывод (I/O), по умолчанию SCL для I2C
D4/SDA/D14	GPIO4	Ввод/вывод (I/O), по умолчанию SDA для I2C
D5/SCK/D13	GPIO14	Ввод/вывод (I/O), тактовый сигнал SPI
D6/MISO/D12	GPIO12	Ввод/вывод (I/O), MISO для SPI
D7/MOSI/D11	GPIO13	Ввод/вывод (I/O), MOSI для SPI
D8	GPIO0	Ввод/вывод (I/O), подтянут к питанию, низкий уровень для режима FLASH
D9/TX1	GPIO2	Ввод/вывод (I/O), подтянут к питанию
D10/SS	GPIO15	Ввод/вывод (I/O), подтянут к земле, выбор кристалла (SS) для SPI
A0	ADC	Аналоговый вход, диапазон 0–3,3 В

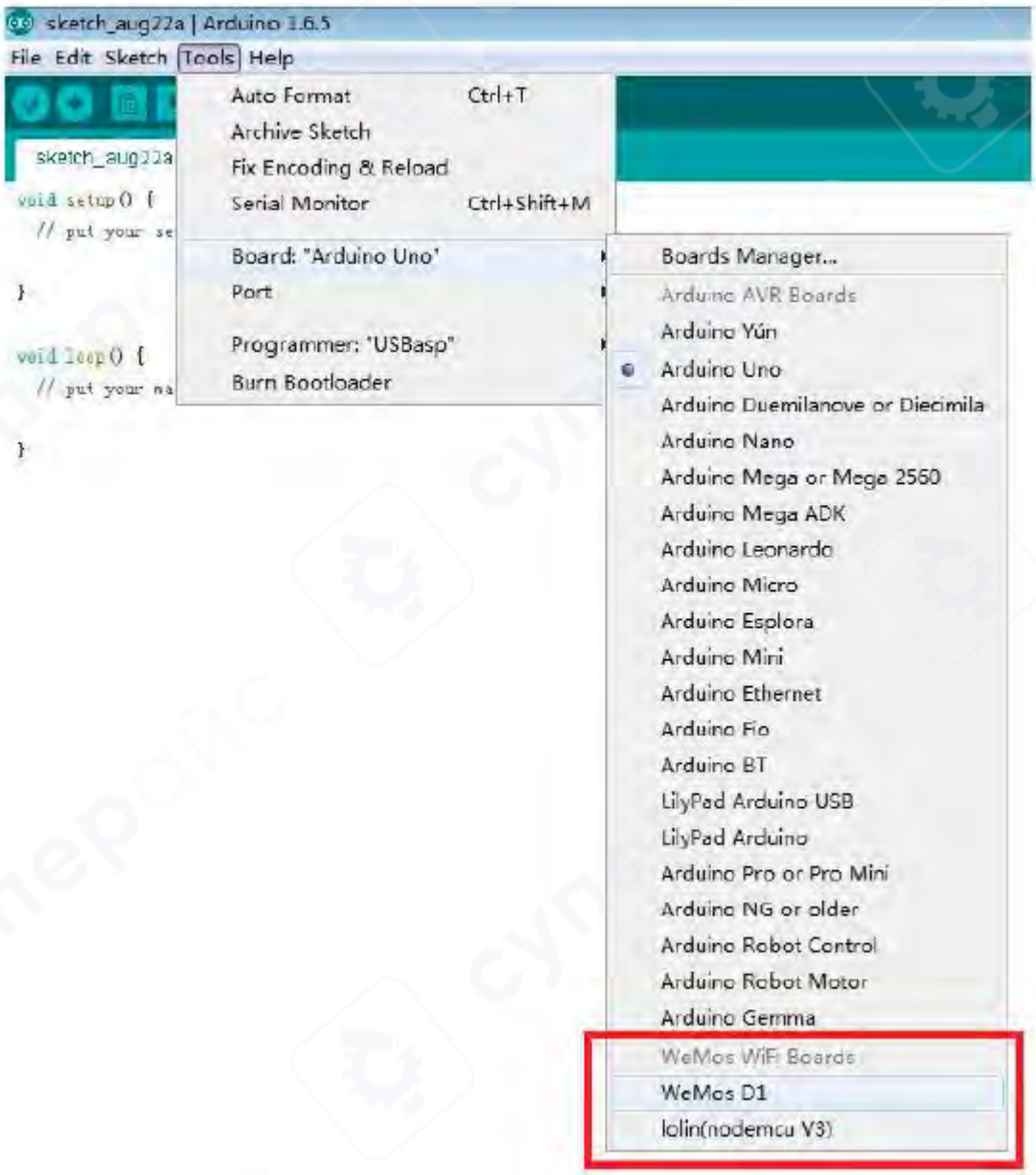
Примечания:

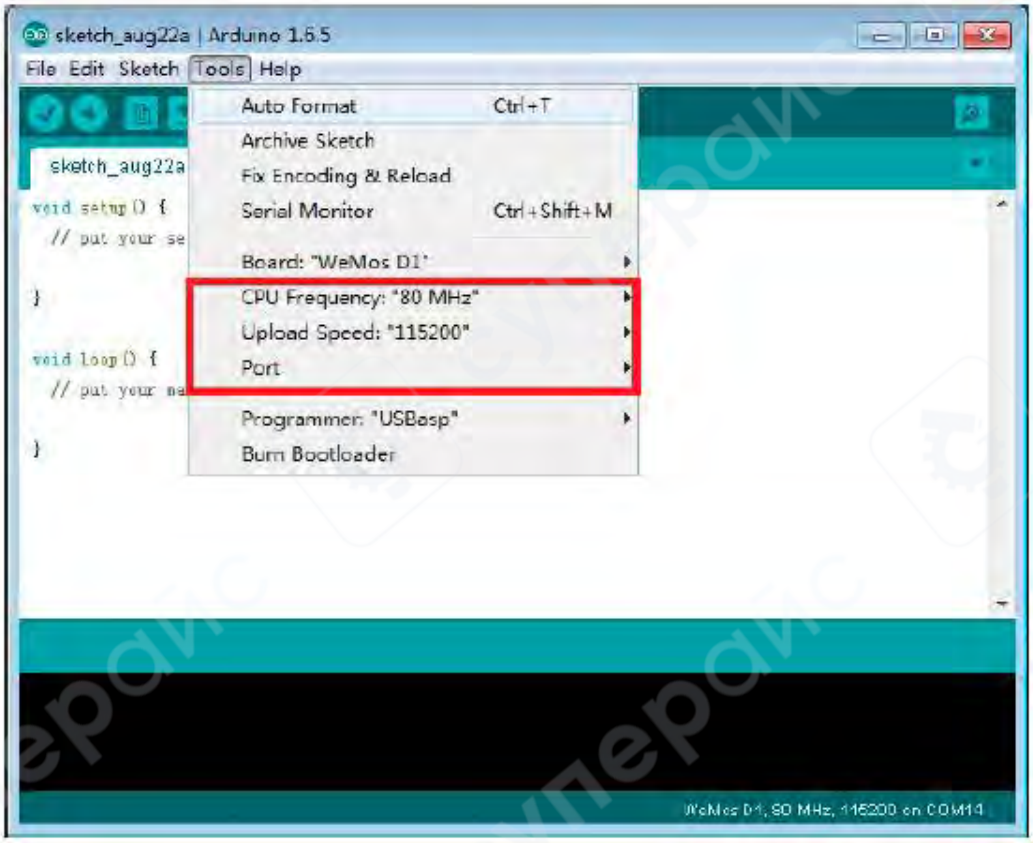
- Все выводы I/O работают на уровне 3,3 В, но могут кратковременно выдерживать 5 В.
- Все выводы I/O, кроме D2, поддерживают прерывания, PWM, I2C и 1-Wire.

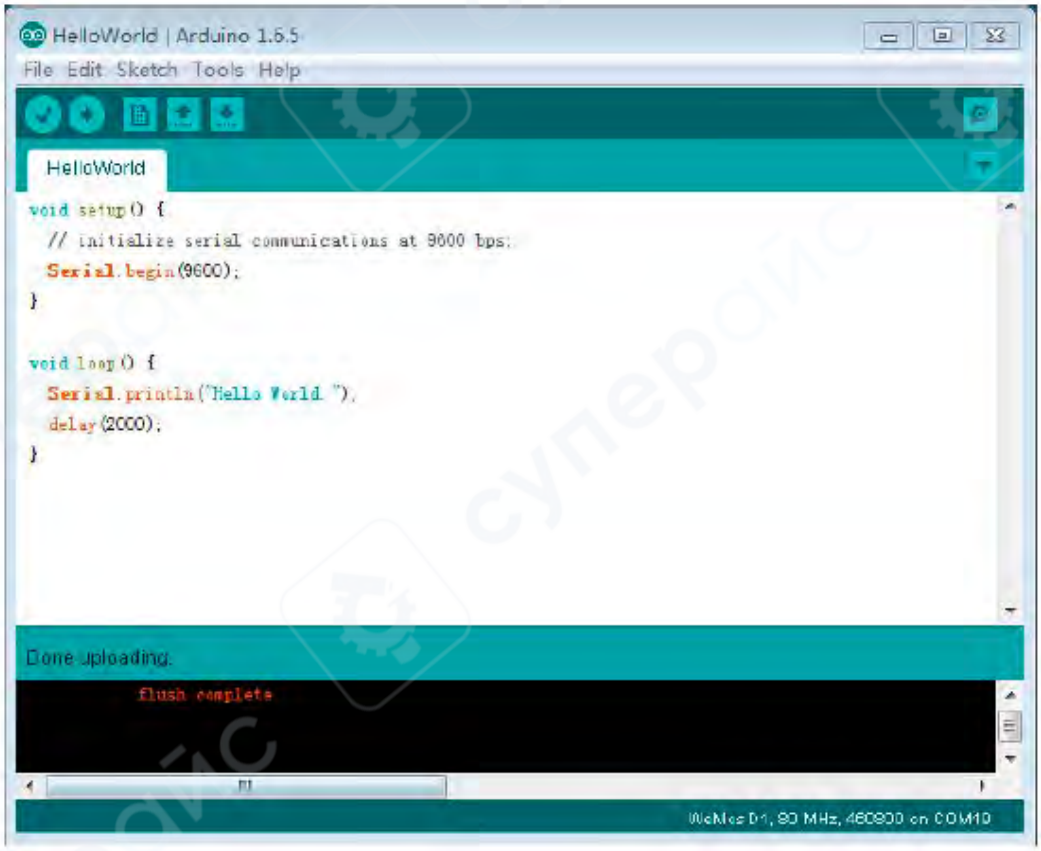
Дополнительная информация:

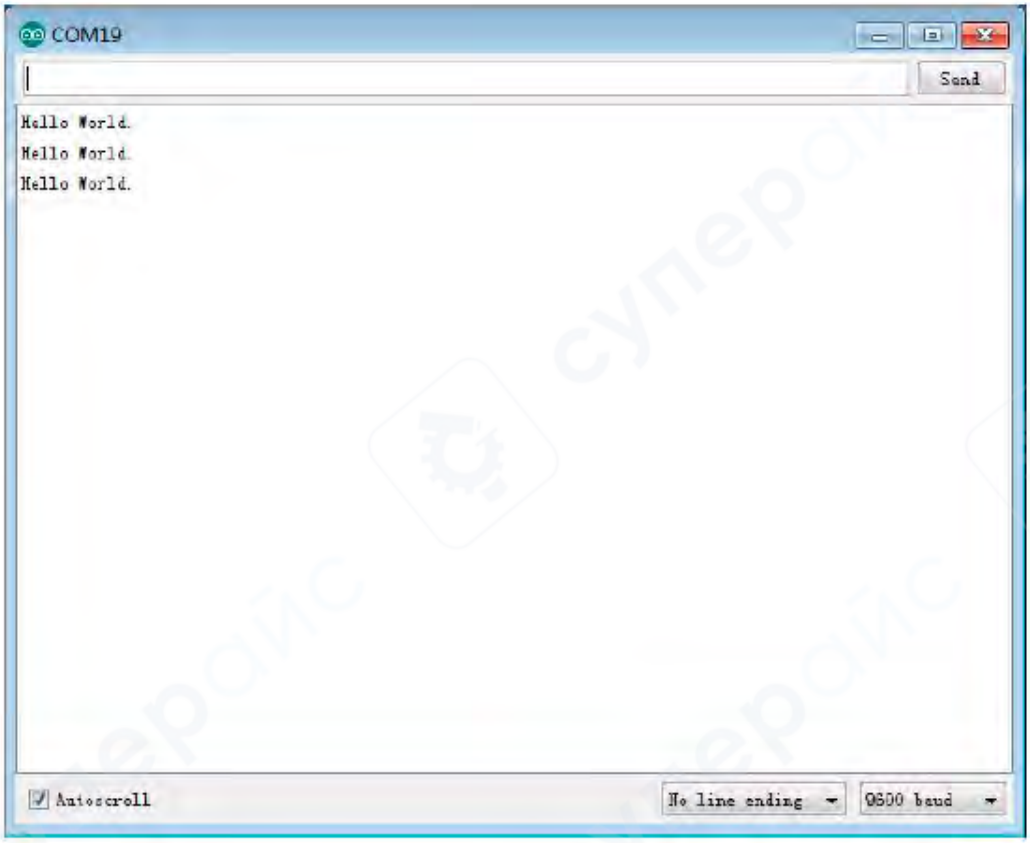
- Данное устройство является экспериментальным компонентом для разработки и не является готовым к использованию RF-модулем. Оно не относится к оборудованию, регулируемому телекоммуникационными стандартами.
- WeMos D1 WiFi Arduino UNO на базе ESP8266 поддерживает прямое программирование через Arduino IDE.

После установки пакета оборудования можно программировать через Arduino IDE, как и с Arduino UNO.



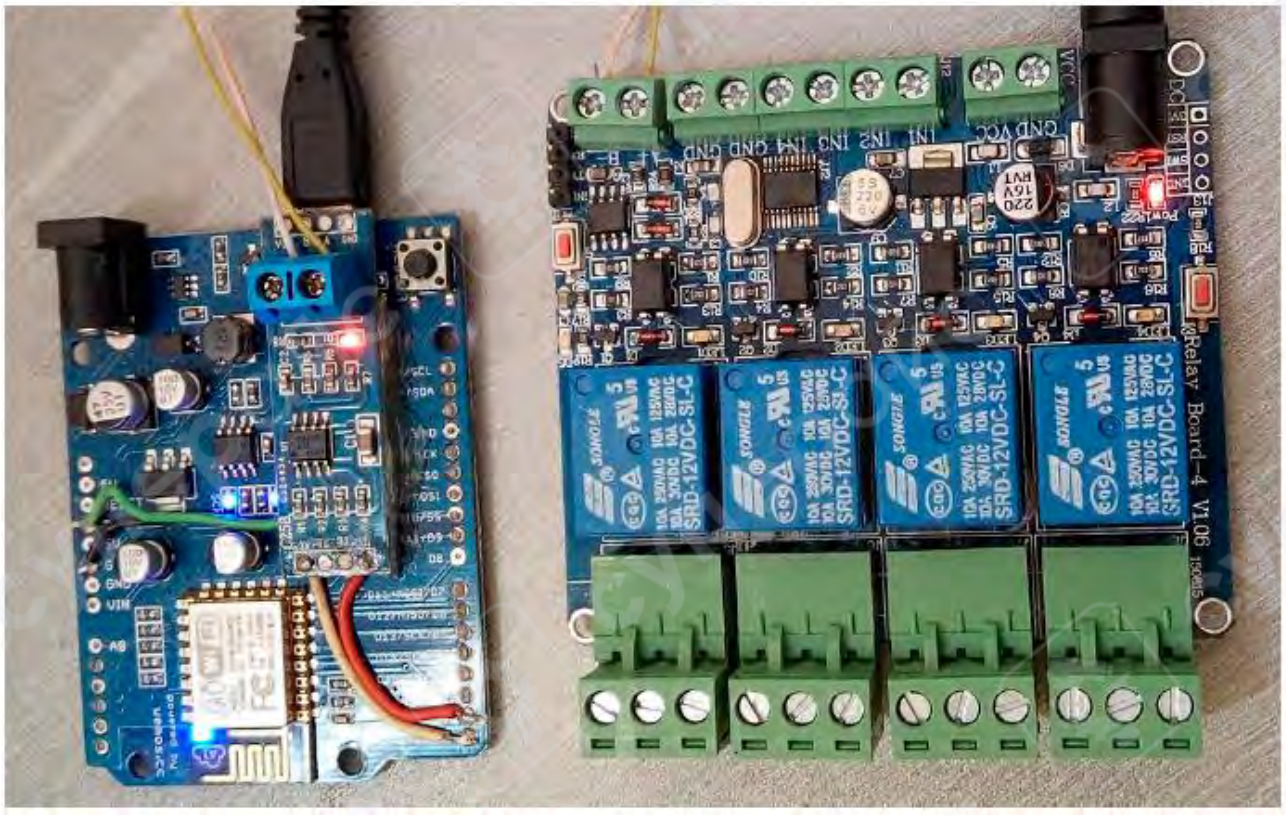

sketch_aug22a | Arduino 1.6.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "WeMos D1"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Port
Programmer: "USBasp"
Burn Bootloader
sketch_aug22a
void setup() {
// put your se
}
void loop() {
// put your ma
}
WeMos D1, 80 MHz, 115200 on COM14


HelloWorld | Arduino 1.6.5
File Edit Sketch Tools Help
HelloWorld
void setup() {
// initialize serial communications at 9600 bps:
Serial.begin(9600);
}
void loop() {
Serial.println("Hello World ");
delay(2000);
}
Done uploading.
flush complete
WeMos D1, 80 MHz, 460800 on COM10


COM19
Send
Hello World.
Hello World.
Hello World.
Autoscroll
No line ending
9600 baud

2 Пример реализации системы умного дома на базе D1 WiFi

Это очень доступный по цене пример реализации системы умного дома с беспроводным управлением. Используя программирование на Arduino, можно легко создать беспроводной шлюз. С помощью модуля TTL-to-RS-485 подключен 4-канальный релейный модуль с интерфейсом RS-485, что позволило успешно собрать простую систему управления умным домом.

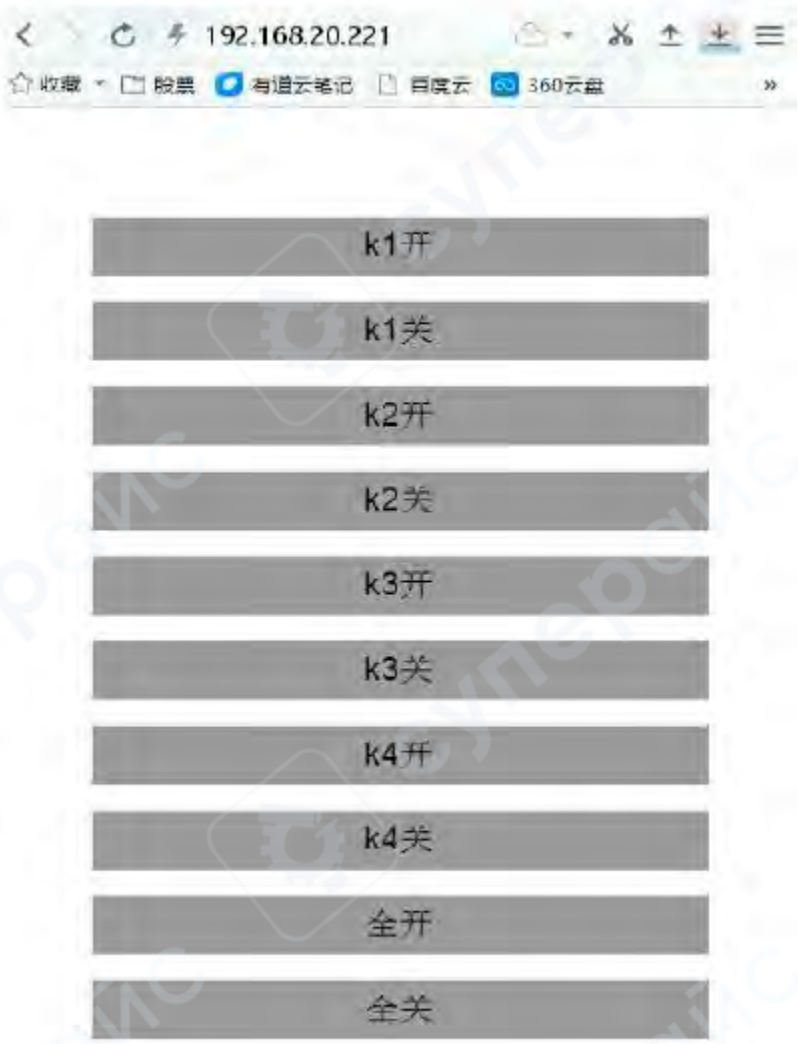


Плата D1 WiFi имеет 11 цифровых входов/выходов (I/O) и 1 аналоговый вход (ADC), а также 4 МБ Flash, 32 КБ SRAM и 80 КБ DRAM, что открывает большие возможности для расширения функциональности.

В данном примере плата D1 WiFi настроена в режиме WebServer, выступая в роли беспроводного шлюза. Для управления используются готовые релейные модули с интерфейсом RS-485, что обеспечивает безопасность и надежность системы. Интерфейс управления доступен напрямую через браузер на устройствах Android, PC, iOS (телефоны, планшеты или другие умные устройства).

IP-адрес платы:

После тестирования отображение на всех устройствах (терминалах) показало высокую степень согласованности.



Описание кода:

1. **Библиотеки и настройки:**
   - Используются библиотеки ESP8266WiFi, ESP8266WebServer, EEPROM и Ticker для работы с Wi-Fi, веб-сервером, энергонезависимой памятью и таймерами.
   - Настройки Wi-Fi (SSID, пароль, IP-адрес, шлюз и маска подсети) задаются вручную.
2. **Команды для управления реле:**
   - Массивы openc и closec содержат команды для включения и выключения реле (в формате Modbus RTU).
   - Каждая команда соответствует определенному реле (1–4) или всем реле одновременно.
3. **Функции управления:**

- handleRoot() — отображает веб-страницу с кнопками для управления реле.
- op1() – op10() — функции для обработки команд включения/выключения реле.
- opoper() — отправляет команды через последовательный интерфейс (UART) и обновляет состояние реле.

4. **Настройка оборудования:**
   - В setup() инициализируются пины, настраивается Wi-Fi (режим точки доступа или клиента), запускается веб-сервер.
   - Используется EEPROM для сохранения состояния реле при отключении питания.
5. **Основной цикл:**
   - В loop() обрабатываются запросы к веб-серверу и поддерживается соединение с клиентом.

Ключевые моменты:

- **Управление через веб-интерфейс:**
  - Плата работает как веб-сервер, позволяя управлять реле через браузер на любом устройстве (ПК, смартфон, планшет).
  - Веб-страница содержит кнопки для включения/выключения каждого реле, а также кнопки "Все включить" и "Все выключить".
- **Подключение реле:**
  - Реле подключаются через интерфейс RS-485, что обеспечивает надежное управление на расстоянии.
  - Команды отправляются через последовательный интерфейс (UART).
- **Защита от сбоев:**
  - Состояние реле сохраняется в EEPROM, что позволяет восстановить их состояние после перезагрузки.

Пример использования:

1. Подключите плату D1 WiFi к сети Wi-Fi.
2. Откройте браузер и введите IP-адрес платы.
3. На веб-странице нажмите кнопки для управления реле (например, "k1 включить" или "Все выключить").
4. Плата отправит команды через RS-485, и реле выполнят соответствующие действия.

Рекомендации:

- Убедитесь, что настройки Wi-Fi (SSID, пароль, IP-адрес) соответствуют вашей сети.
- Проверьте правильность подключения реле и RS-485 модуля.
- Для расширения функциональности можно добавить дополнительные реле или датчики.

Исходный код:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <EEPROM.h>
#include <Ticker.h>
      Ticker tickerflash;

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p);for(;
i<sizeof(p); i++) EEPROM.write(address+i,pp);}
#define EEPROM_read(address, p) {int i = 0; byte *pp =
(byte*)&(p);for(; i<sizeof(p); i++) pp=EEPROM.read(address+i);}
/* Set these to your desired credentials. */
static char Apid[9] = "NETGEAR";// В зависимости от настроек вашего роутера
static char softAPID[] = "KYSMART";
static char ApPass[10] = "zjky61448";// В зависимости от настроек вашего
роутера
byte APip[] = { 192, 168, 20, 221 };// В зависимости от настроек вашего роутера
byte APGateWay[] = { 192, 168, 20, 254 };// В зависимости от настроек вашего
роутера
byte APSubNet[] = { 255, 255, 255, 0 };
unsigned char openc[5][8] = {
{ 0x01, 0x06, 0x00, 0x01, 0x01, 0x01, 0x18, 0x5a},//1 Реле №X включено
{ 0x01, 0x06, 0x00, 0x01, 0x02, 0x01, 0x18, 0xaa},//2 Реле №X включено
{ 0x01, 0x06, 0x00, 0x01, 0x03, 0x01, 0x19, 0x3a},//3 Реле №X включено
{ 0x01, 0x06, 0x00, 0x01, 0x04, 0x01, 0x1b, 0x0a},//4 Реле №X включено
{ 0x01, 0x06, 0x00, 0x01, 0xff, 0xff, 0xd9, 0xba}//全亮
};
unsigned char closec[5][8] = {
{ 0x01, 0x06, 0x00, 0x01, 0x01, 0x00, 0xd9, 0x9a},//1 Реле №X выключено
{ 0x01, 0x06, 0x00, 0x01, 0x02, 0x00, 0xd9, 0x6a},//2 Реле №X выключено
{ 0x01, 0x06, 0x00, 0x01, 0x03, 0x00, 0xd8, 0xfa},//3 Реле №X выключено
{ 0x01, 0x06, 0x00, 0x01, 0x04, 0x00, 0xda, 0xca},//4 Реле №X выключено
{ 0x01, 0x06, 0x00, 0x01, 0x00, 0x00, 0xd8, 0x0a}// Все выключено
};
byte TSwitch[] = {
0, 0, 0, 0
};
byte StatSave[] = {
0xff, 0xff, 0xff, 0xff
};// Защита от потери питания
```

```
byte Switchnum = 10;//13
const byte SwitchIO[] = {
D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D14, D15
};// Список переключателей
byte flashLed = D13; // Индикатор с плавающей яркостью
int ledState = LOW;
char funcstr[800];
     byte aptype = 0;// Режим: 1 AP 0 CLIENT
const char pageS[] PROGMEM = "<meta name=\"viewport\"
content=\"width=device-width,initial-scale=1.0\">\r\n<meta
http-equiv=\"Content-Type\" content=\"text/html;charset=utf-8\">\r\n"
"<style
type=\"text/css\">"
"*
{margin:3;padding:3;}"
"input{width:90%;hei
ght:40px;font-size:20px;background:#999;}"
"</style>"
"<center><h2> Добро пожаловать!
Используйте умный дом от Keyou  <br></h2><h3>"
"%s"
"<br><a
href=\"http://www.it15168.com\">ZJKEYOU SMART HOME</a></center></h3>";
const char LineS[] PROGMEM = "%s<br><br><a href=\"/\">返回</a><br>";
// Create an instance of the server
ESP8266WebServer server(80);
int RevB(int v, byte b) // Инвертировать значение бита под номером B.
{
return (v ^= 1 << b); // Инвертировать бит на позиции POS числа (Number).
}
void SaveSta()
{
EEPROM.begin(512);
EEPROM_write(30, StatSave);
EEPROM.commit();
EEPROM.end();
}
void setSta(byte Aswitch, byte Sta )
{
```

```
byte i = Aswitch / 8;
byte j = Aswitch % 8;
bitWrite(TSwitch[3 - i], j, Sta);
}
void flash() {
digitalWrite(flashLed, ledState);
ledState = !ledState ;
     }
void handleRoot() {
sprintf_P(funcstr, pageS, "<form action=/op1><input type=submit
value=k1 开></form>"
"<form action=/op2><input type=submit value=k1
Выключить ></form>"
"<form action=/op3><input type=submit value=k2
开></form>"
"<form action=/op4><input type=submit value=k2
Выключить ></form>"
"<form action=/op5><input type=submit value=k3
开></form>"
"<form action=/op6><input type=submit value=k3
Выключить ></form>"
"<form action=/op7><input type=submit value=k4
开></form>"
"<form action=/op8><input type=submit value=k4
Выключить ></form>"
"<form action=/op9><input type=submit value=全
开></form>"
"<form action=/op10><input type=submit value=全
Выключить ></form>");
server.send(200, "text/html", funcstr);
}
void op1() {
opoper(0, 1, "k1  Уже включено ");
}
void op2() {
opoper(0, 0, "k1 Уже выключено ");
}
void op3() {
opoper(1, 1, "k2  Уже включено ");
```

```
}
void op4() {
opoper(1, 0, "k2 Уже выключено ");
}
void op5() {
opoper(2, 1, "k3  Уже включено ");
     }
void op6() {
opoper(2, 0, "k3 Уже выключено ");
}
void op7() {
opoper(3, 1, "k4 Уже включено ");
}
void op8() {
opoper(3, 0, "k4 Уже выключено ");
}
void op9() {
opoper(4, 1, " Все уже включено ");
}
void op10() {
opoper(4, 0, " Все уже выключено ");
}
void opoper(byte port, byte oper, char *str)
{
char funcstr1[100];
if (oper == 1) {
Serial.write(openc[port], 8);
}
else
{
Serial.write(closec[port], 8);
}
if (port < 4) {
digitalWrite(SwitchIO[port], oper);
setSta(port, oper);
}
funcstr[0] = 0;
sprintf_P(funcstr1, LineS, str);
sprintf_P(funcstr, pageS, funcstr1);
```

```
server.send(200, "text/html", funcstr);
}
void handleNotFound() {
String message = "File Not Found\n\n";
message += "URI: ";
     message += server.uri();
message += "\nMethod: ";
message += (server.method() == HTTP_GET) ? "GET" : "POST";
message += "\nArguments: ";
message += server.args();
message += "\n";
for (uint8_t i = 0; i < server.args(); i++) {
message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
}
server.send(404, "text/plain", message);
}
void setup(void) {
Serial.begin(9600);
pinMode(flashLed, OUTPUT);
for (int i = 0; i < Switchnum / 8; i++)
for (int j = 0; j < 8; j++) {
pinMode(SwitchIO[i * 8 + j], OUTPUT);
digitalWrite(SwitchIO[i * 8 + j], (bitRead(TSwitch[3 - i], j) >
0) && (bitRead(StatSave[3 - i], j) > 0));
}
Serial.print("Conn to:");
Serial.println(Apid);
if (aptype == 1) {
WiFi.softAP(softAPID, ApPass);
IPAddress myIP = WiFi.softAPIP();
Serial.print("AP IP address:");
Serial.println(myIP);
}
else
{
WiFi.config(APip, APGateWay, APSubNet);
WiFi.begin(Apid, ApPass);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
```

```
Serial.print(".");
}
Serial.println("server started:");
Serial.println(WiFi.localIP());
}
server.on("/", handleRoot);
server.on("/op1", op1);
server.on("/op2", op2);
     server.on("/op3", op3);
server.on("/op4", op4);
server.on("/op5", op5);
server.on("/op6", op6);
server.on("/op7", op7);
server.on("/op8", op8);
server.on("/op9", op9);
server.on("/op10", op10);
server.onNotFound(handleNotFound);
server.begin();
tickerflash.attach_ms(800, flash);
}
void loop(void) {
server.handleClient();
WiFiClient client = server.client();
client.flush();
     }
```