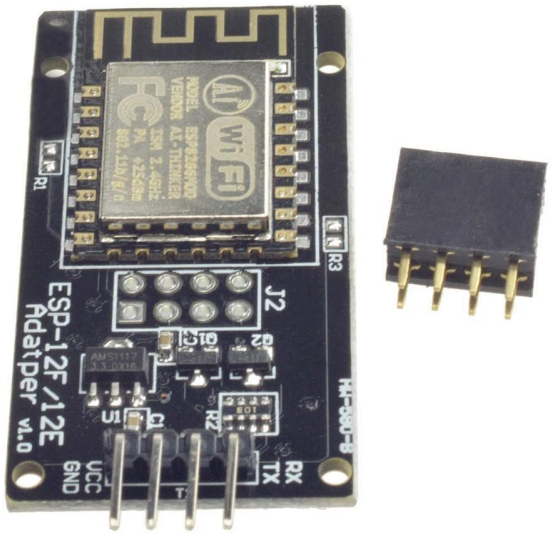
WiFi sériový modul ESP-12F

POPIS

WiFi modul je dodáván s již předinstalovaným firmwarem. Komunikace s modulem a jeho konfigurace probíhá pomocí AT příkazů zasílaných po sériové lince. Pro snadné připojení k platformě Arduino slouží komunikační rozhraní UART.
Modul je vhodný pro využití v IoT aplikacích (internet věcí).

- předinstalovaný firmware
- AT příkazy
- komunikace přes rozhraní UART
- čip ESP8266



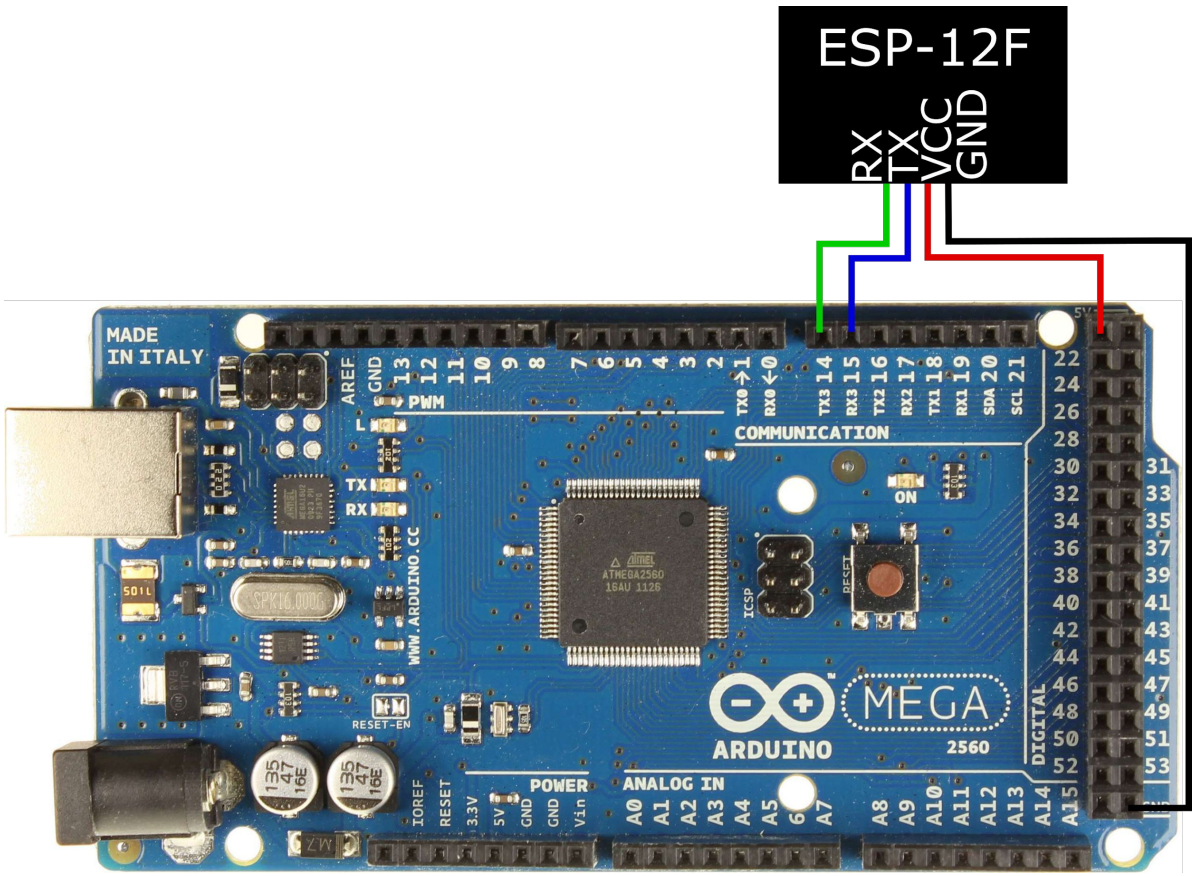
SPECIFIKACE

Hlavní čip	ESP8266	**Vysílací výkon (802.11b)**	20 dBm
Napájení	5 V	**Standardy**	802.11b/g/n
Napěťová úroveň logiky (RX, TX)	3,3 i 5 V	**Frekvenční pásmo**	2,4 GHz
Spotřeba	až 250 mW	**Programovací konektor**	ISP
Komunikace	UART	**Rozměry (mm)**	46 x 24 x 5
Nativní baudrate	115200	**Hmotnost**	6 g

ZAPOJENÍ

Upozornění: Pokud chce uživatel v aplikaci využívat sériovou linku pro komunikaci s PC, doporučujeme modul používat s deskami Arduino Mega nebo Due. Dané desky mají větší počet hardwarových sériových linek, ke kterým lze modul připojit. Díky tomu může být zachována komunikace s PC.

V případě použití Arduino Uno a Nano uživatel přijde o možnost komunikace přes sériový terminál s PC, jelikož je vhodné modul připojit na jedinou hardwarovou sériovou linku mikrokontroléru, která je připojena k USB TTL převodníku (sloužící pro komunikaci s PC). Není vhodné používat emulovanou sériovou linku, jelikož není navržena pro nativní baudrate 115200. Na modulu lze baudrate změnit na hodnotu 9600 (a v takovém případě by emulovaná linka pracovala správně), nicméně toto nastavení nelze uložit do flash paměti ESP a musí být nakonfigurováno po každém restartu WiFi modulu.



Základní AT příkazy

AT příkaz	Popis	Příklad
AT	Testovací příkaz	AT
AT+RST	Reset modulu	AT+RST
AT+RESTORE	Tovární nastavení	AT+RESTORE
AT+BAUD_CUR	Nastavení baudrate na 9600	AT+BAUD_CUR=9600,8,1,0,0
AT+CWMODE	AT+CIFSR	AT+CWMODE=1
AT+CWJAP	Připojení k WiFi síti	AT+CWJAP=”název sítě”,”heslo”
AT+CIFSR	Výpis přiřazené IP adresy	AT+CIFSR
AT+CIPMUX	Vícenásobné připojení klientů	AT+CIPMUX=1
AT+CIPSERVER	Vytvoření server na portu 80	AT+CIPSERVER=1,80
AT+CIPSEND	Posílání dat na server (25 bajtů)	AT+CIPSEND=0,25

Více informací o AT příkazech lze nalézt v originálním datasheetu výrobce.

UKÁZKA PROGRAMU

Upozornění: Jedná se o základní příklad serveru. Program je v této podobě kompatibilní jen s Arduino Mega a zapojení WiFi modulu koresponduje s obrázkem výše.

```
//code is for Arduino Mega only (HW serial)
const bool printReply = true;
const char line[] = "-----\n\r";
int loopCount = 0;
char html[50];
char command[20];
char reply[500];
char ipAddress [20];
char name[30];
int lenHtml = 0;
char temp[5];
void setup()
{
  Serial.begin(115200);
  Serial.println("Start\r\n\r\n");
  Serial3.begin(115200); // your ESP8266 module's baud rate might be different
  // reset the ESP8266
  Serial.println("reset the module");
  Serial3.print("AT+RST\r\n");
  getReply( 2000 );
  // configure as a station
  Serial.println("Change to station mode");
  Serial3.print("AT+CWMODE=1\r\n");
  getReply( 1500 );
  // connect to the network. Uses DHCP. ip will be assigned by the router.
  Serial.println("Connect to a network ");
  // Enter the SSID and password for your own network
  Serial3.print("AT+CWJAP=\"WiFi\",\"heslo\"\r\n");
  getReply( 6000 );
  // get ip address
  Serial.println("Get the ip address assigned ny the router");
  Serial3.print("AT+CIFSR\r\n");
  getReply( 1000 );
  // parse ip address.
  int len = strlen( reply );
  bool done = false;
  bool error = false;
  int pos = 0;
  while (!done) {
    if ( reply[pos] == 10) {
      done = true;
    }
    pos++;
```

```
    if (pos > len) {
      done = true;
      error = true;
    }
  }
  if (!error)
  {
    int buffpos = 0;
    done = false;
    while (!done)
    {
      if ( reply[pos] == 13 ) {
        done = true;
      }
      else {
        ipAddress[buffpos] = reply[pos];
        buffpos++;
        pos++;
      }
    }
    ipAddress[buffpos] = 0;
  }
  else {
    strcpy(ipAddress, "ERROR");
  }
  // configure for multiple connections
  Serial.println("Set for multiple connections");
  Serial3.print("AT+CIPMUX=1\r\n");
  getReply( 1500 );
  // start server on port 80
  Serial.println("Start the server");
  Serial3.print("AT+CIPSERVER=1,80\r\n");
  getReply( 1500 );
  Serial.println("");
  Serial.println("Waiting for page request");
  Serial.print("Connect to "); Serial.println(ipAddress);
  Serial.println("");
}
```

```
void loop()
{
  if (Serial3.available()) // check if the ESP8266 is sending data
  {
    // this is the +IPD reply - it is quite long.
    // normally you would not need to copy the whole message in to a variable you can copy up to
"HOST" only
    // or you can just search the data character by character as you read the serial port.
    getReply( 2000 );
    bool foundIPD = false;
    for (int i = 0; i < strlen(reply); i++)
    {
      if (  (reply[i] == 'I') && (reply[i + 1] == 'P') && (reply[i + 2] == 'D')   ) {
        foundIPD = true;
      }
    }
    if ( foundIPD  )
    {
      loopCount++;
      // Serial.print( "Have a request.  Loop = ");  Serial.println(loopCount); Serial.println("");
      // check to see if we have a name - look for name=
      bool haveName = false;
      int nameStartPos = 0;
      for (int i = 0; i < strlen(reply); i++)
      {
        if (!haveName) // just get the first occurrence of name
        {
          if (  (reply[i] == 'n') && (reply[i + 1] == 'a') && (reply[i + 2] == 'm') && (reply[i + 3] == 'e')  &&
(reply[i + 4] == '=') )
          {
            haveName = true;
            nameStartPos = i + 5;
          }
        }
      }

      // get the name - copy everything from nameStartPos to the first space character
      if (haveName)
      {
        int tempPos = 0;
        bool finishedCopying = false;
        for (int i = nameStartPos; i < strlen(reply); i++)
        {
          if ( (reply[i] == ' ') && !finishedCopying )  {
            finishedCopying = true;
          }
```

```
if ( !finishedCopying )                    {
          name[tempPos] = reply[i];
          tempPos++;
        }
      }
      name[tempPos] = 0;
    }
    if (haveName) {
      Serial.print( "name = ");
      Serial.println(name);
      Serial.println("");
    }
    else          {
      Serial.println( "no name entered");
      Serial.println("");
    }
    // start sending the HTML
    strcpy(html, "<html><head></head><body>");
    strcpy(command, "AT+CIPSEND=0,25\r\n");
    Serial3.print(command);
    getReply( 2000 );
    Serial3.print(html);
    getReply( 2000 );
    strcpy(html, "<h1>ESP8266 Webserver</h1>");
    strcpy(command, "AT+CIPSEND=0,26\r\n");
    Serial3.print(command);
    getReply( 2000 );
    Serial3.print(html);
    getReply( 2000 );
    strcpy(html, "<p>Served by Arduino and ESP8266</p>");
    strcpy(command, "AT+CIPSEND=0,36\r\n");
    Serial3.print(command);
    getReply( 2000 );
    Serial3.print(html);
    getReply( 2000 );
    strcpy(html, "<p>Request number ");
    itoa( loopCount, temp, 10);
    strcat(html, temp);
    strcat(html, "</p>");
    // need the length of html
    int lenHtml = strlen( html );
    strcpy(command, "AT+CIPSEND=0,");
    itoa( lenHtml, temp, 10);
    strcat(command, temp);
    strcat(command, "\r\n");
    Serial3.print(command);
```

```
getReply( 2000 );
      Serial3.print(html);
      getReply( 2000 );
      if (haveName)
      {
        // write name
        strcpy(html, "<p>Your name is "); strcat(html, name ); strcat(html, "</p>");
        // need the length of html for the cipsend command
        lenHtml = strlen( html );
        strcpy(command, "AT+CIPSEND=0,"); itoa( lenHtml, temp, 10); strcat(command, temp);
strcat(command, "\r\n");
        Serial3.print(command);
        getReply( 2000 );
        Serial3.print(html);
        getReply( 2000 );
      }
      strcpy(html, "<form action=\""); strcat(html, ipAddress); strcat(html, "\" method=\"GET\">");
strcat(command, "\r\n");
      lenHtml = strlen( html );
      itoa( lenHtml, temp, 10);
      strcpy(command, "AT+CIPSEND=0,");
      itoa( lenHtml, temp, 10);
      strcat(command, temp);
      strcat(command, "\r\n");
      Serial3.print(command);
      getReply( 2000 );
      Serial3.print(html);
      getReply( 2000 );
      strcpy(html, "</body></html>");
      strcpy(command, "AT+CIPSEND=0,14\r\n");
      Serial3.print(command);
      getReply( 2000 );
      Serial3.print(html);
      getReply( 2000 );
      // close the connection
      Serial3.print( "AT+CIPCLOSE=0\r\n" );
      getReply( 1500 );
      Serial.println("last getReply 1 ");
      getReply( 1500 );
      Serial.println("last getReply 2 ");
      getReply( 1500 );
    } // if(Serial3.find("+IPD"))
  } //if(Serial3.available())
  delay (100);
}
```

```
void getReply(int wait)
{
  int tempPos = 0;
  long int time = millis();
  while ( (time + wait) > millis())
  {
    while (Serial3.available())
    {
      char c = Serial3.read();
      if (tempPos < 500) {
        reply[tempPos] = c;
        tempPos++;
      }
    }
    reply[tempPos] = 0;
  }
  if (printReply) {
    Serial.println( reply );
    Serial.println(line);
  }
}
```