KAMAMI

KAmod ESP32 POW RS485 EANT

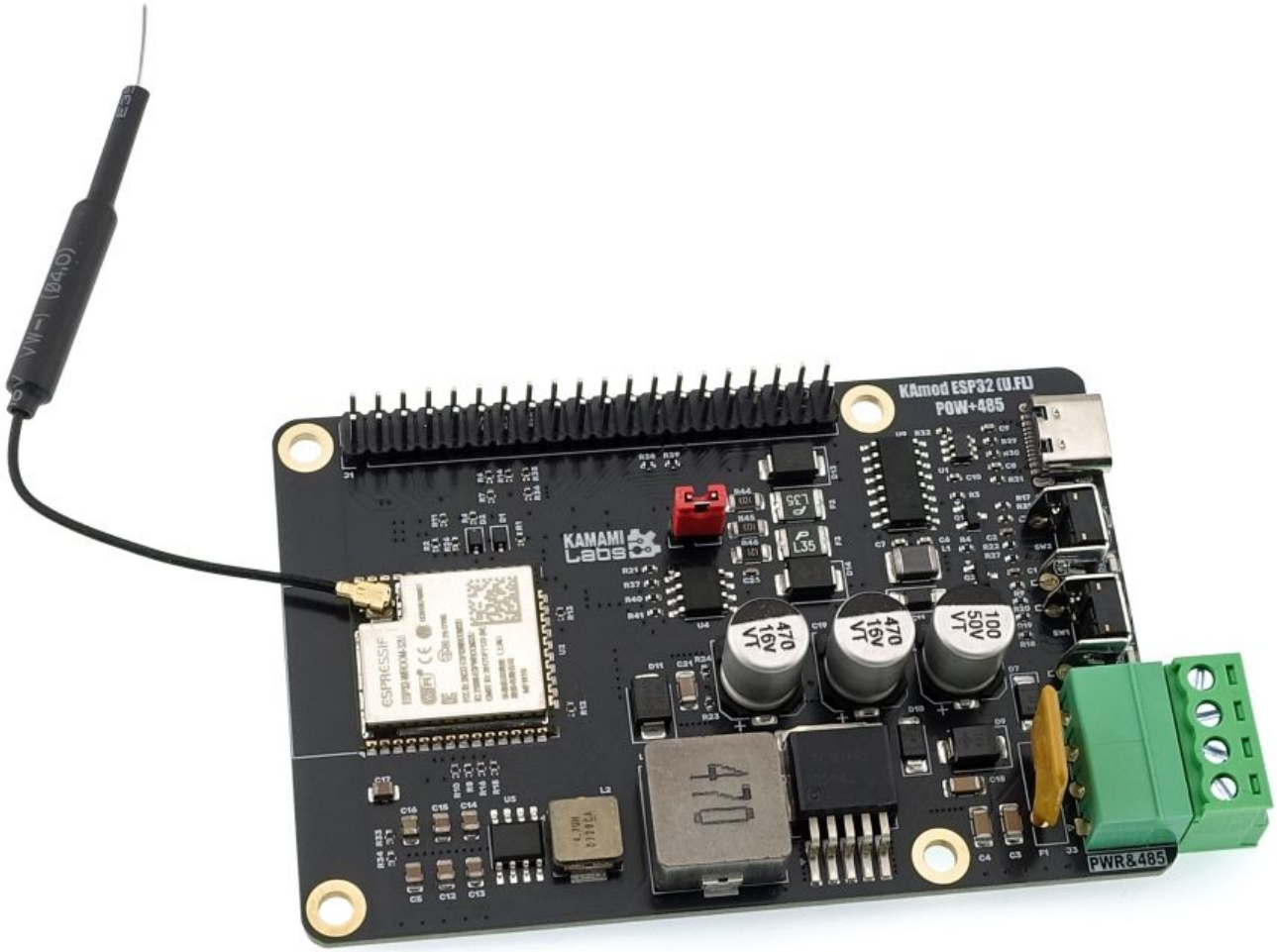
Rev. 20260404125131
Źródło: https://wiki.kamamilabs.com/index.php?title=KAmod_ESP32_POW_RS485_EANT

Table of contents

Description

KAmod ESP32 POW RS485 EANT (U.FL) - Evaluation board with ESP32-WROOM-32U Wi-Fi module, RS485 interface, and high-efficiency power supply system.

The KAmod ESP32 POW RS485 EANT board features the ESP32-WROOM-32U module, which enables communication in 2.4 GHz Wi-Fi wireless networks and is equipped with a U.FL connector for an external antenna. The board includes a complete USB-UART converter with a USB-C connector, allowing for convenient programming of the ESP32 module. Another feature is the RS485 interface block, commonly used in home and industrial automation, which implements communication via 2-wire twisted pair, e.g., in the MODBUS standard. The board is complemented by a high-efficiency power supply system that operates with an input voltage range of 8 to 32 V, providing stabilized 5 V and 3.3 V voltages with significant current capacity. The board design matches the Raspberry Pi family SBCs - it measures 85x56 mm, and all essential I/O ports and power supply voltages are broken out on the characteristic 40-pin connector.



Basic Parameters

- ESP32-WROOM-32U module for 2.4 GHz Wi-Fi communication, equipped with U.FL (IPX) antenna connector
- Integrated UART-USB converter with USB-C connector for ESP32 programming
- RS485 interface for Half-duplex communication with a maximum speed of 1 Mbps
- Maximum number of devices connected to the RS485 bus: 64
- Adapted for supply voltage in the range of 8...32 V
- Provides stabilized 5.1 V voltage with continuous current up to 3 A and short-term up to 5 A
- Provides stabilized 3.3 V voltage with continuous current up to 1 A and short-term up to 2 A
- Overvoltage, overload, and thermal protection
- All essential I/O ports and power supply voltages are broken out to a 40-pin connector in the Raspberry Pi standard

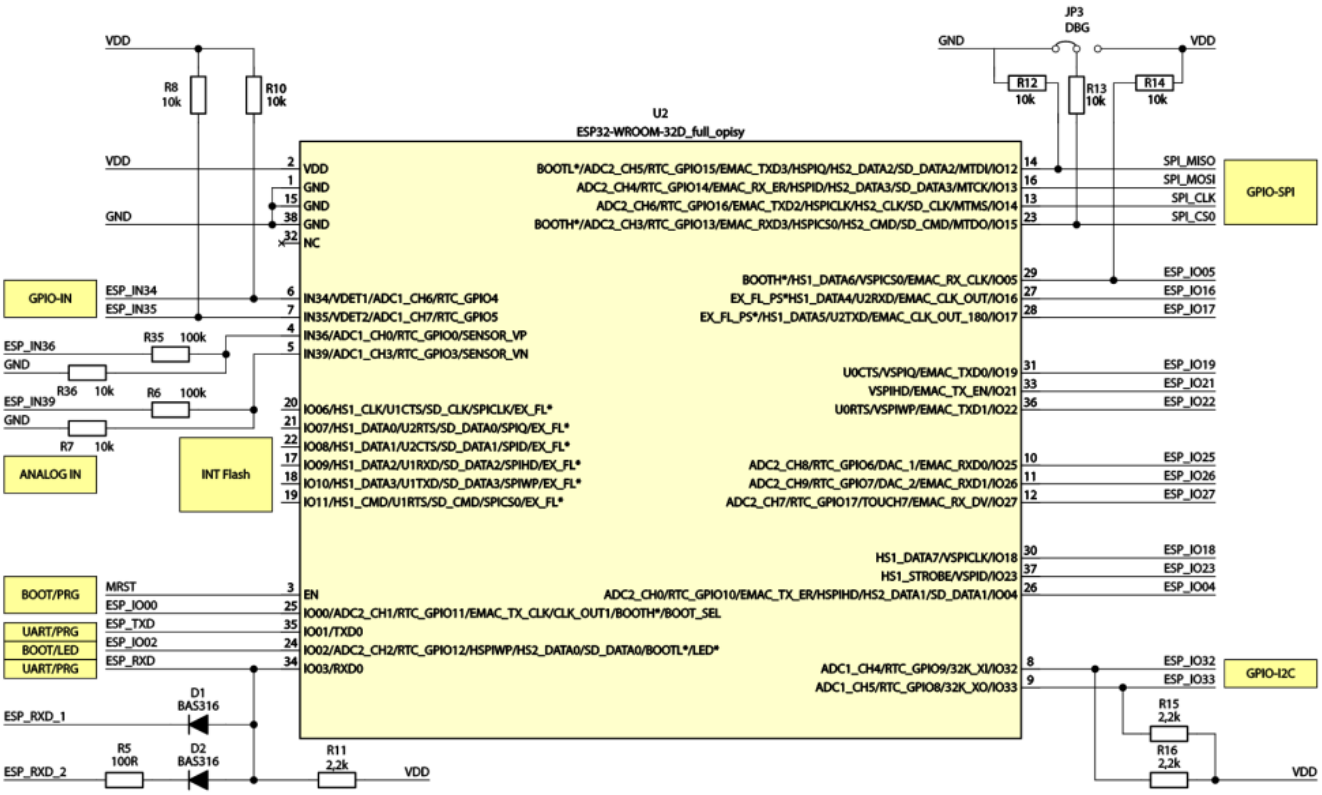
- Board dimensions: 85x56 mm, height approx. 20 mm

Standard Equipment

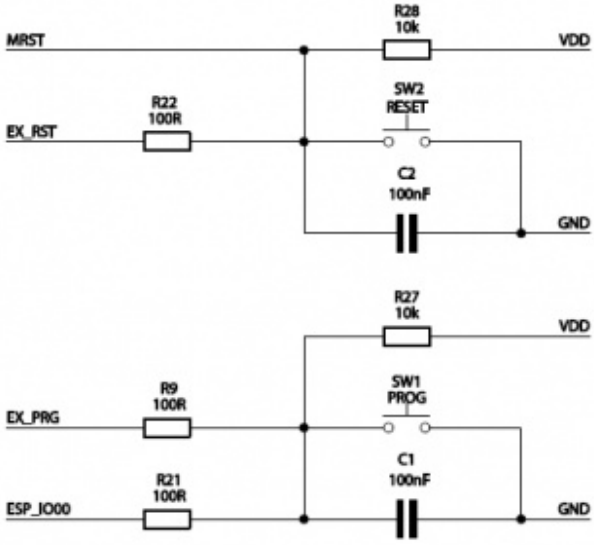
Code	Description
KAmod ESP32 POW RS485 EANT (U.FL)	Assembled and tested module
Antenna	Wi-Fi antenna with U.FL connector

Electrical Schematic

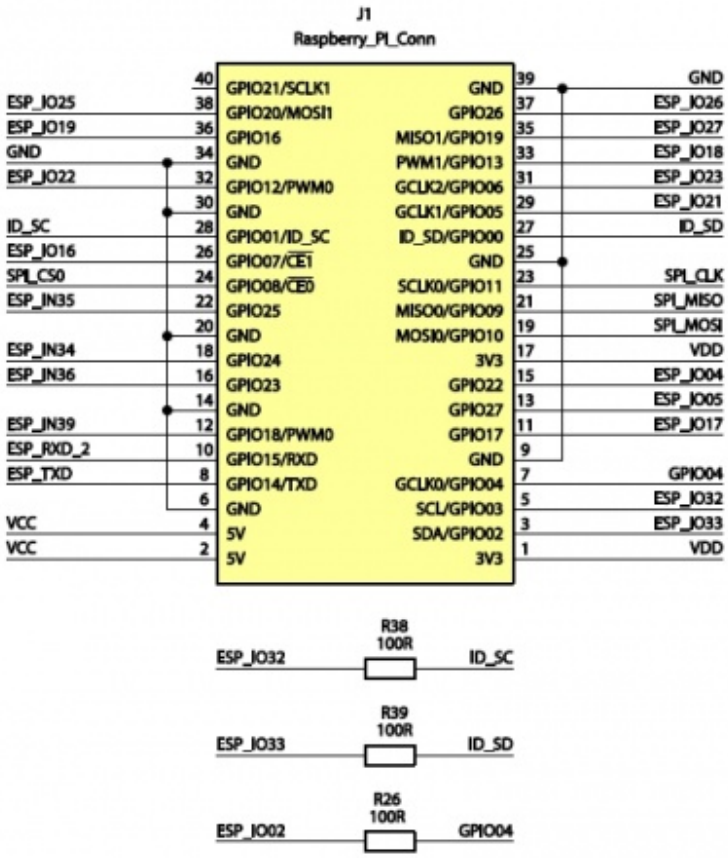
ESP32 Module



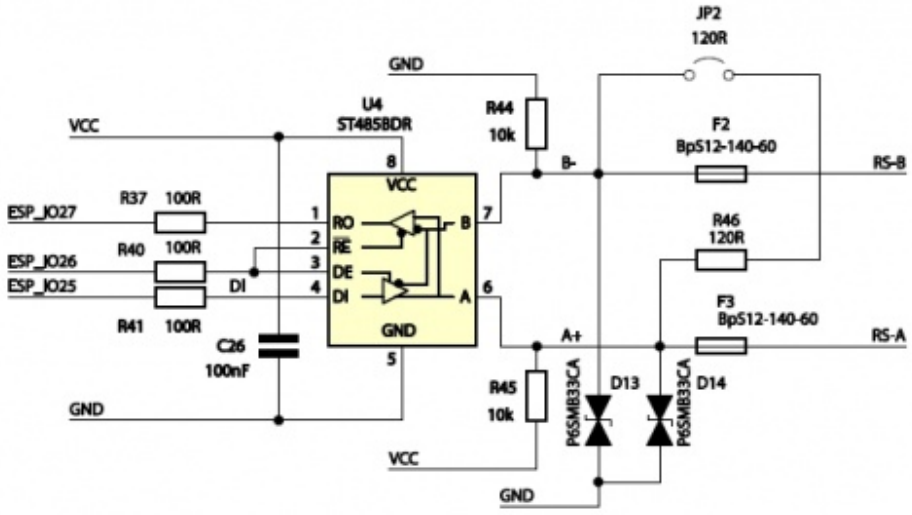
Reset and programming components



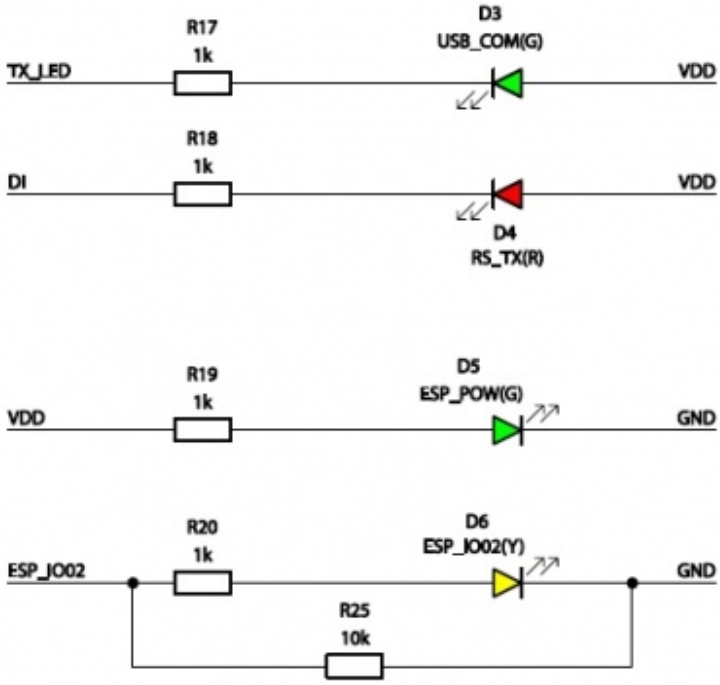
GPIO Connector



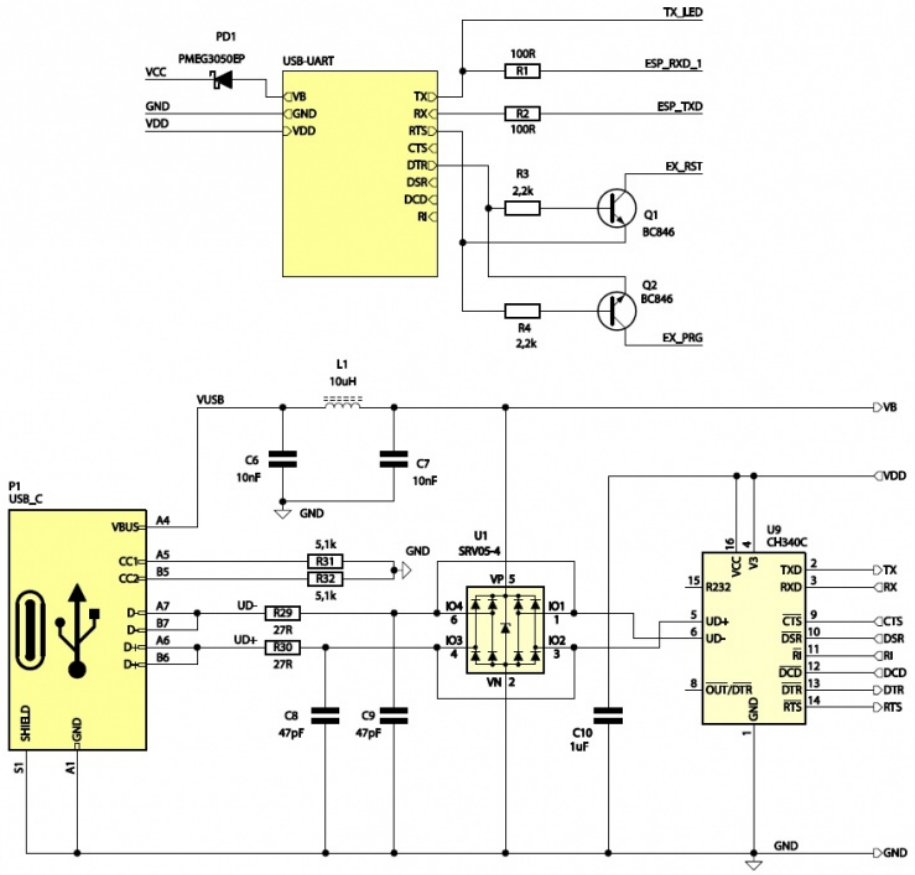
RS485 Interface



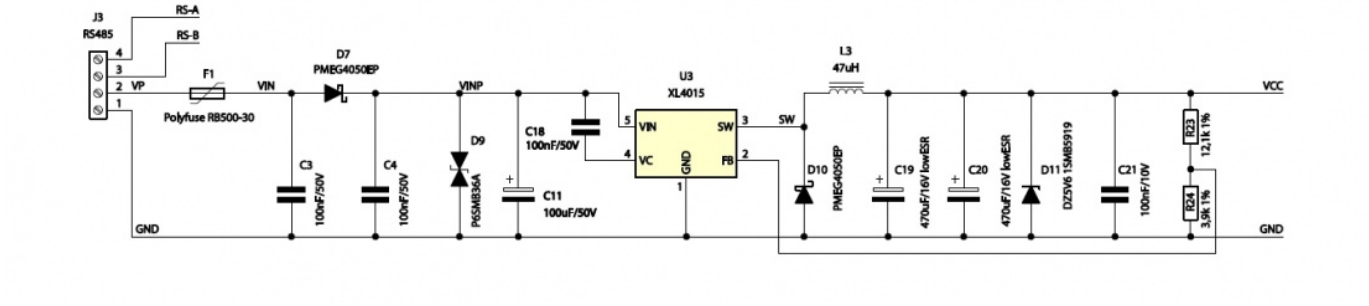
Signaling LEDs



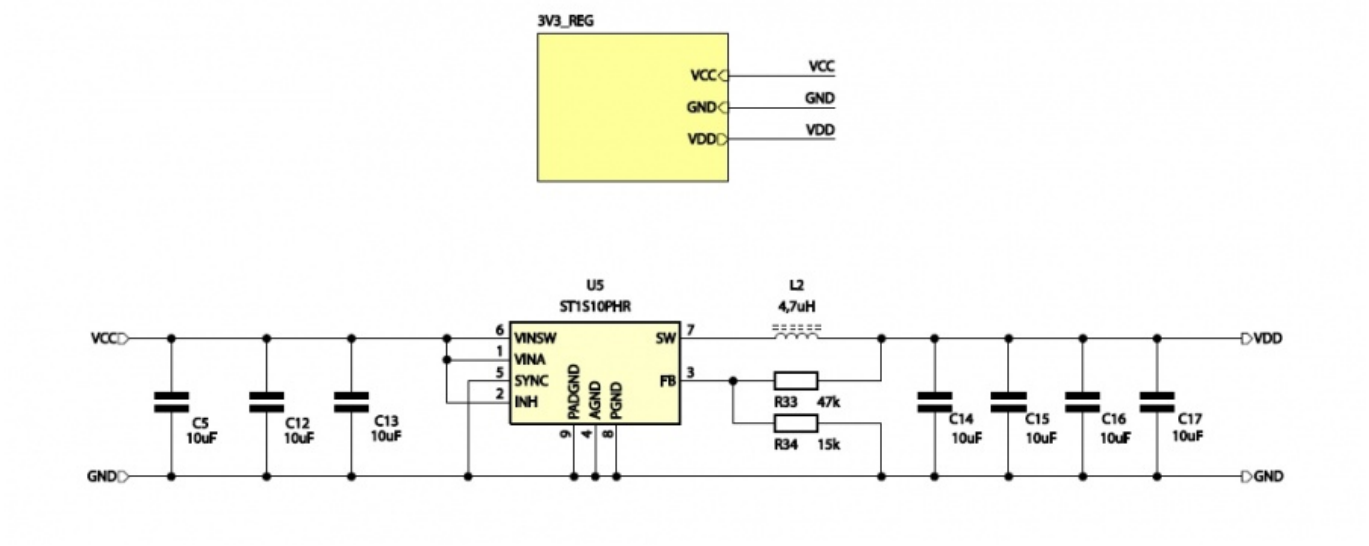
USB-UART Interface



5 V Power Supply Block



3.3 V Power Supply Block



Power Connector

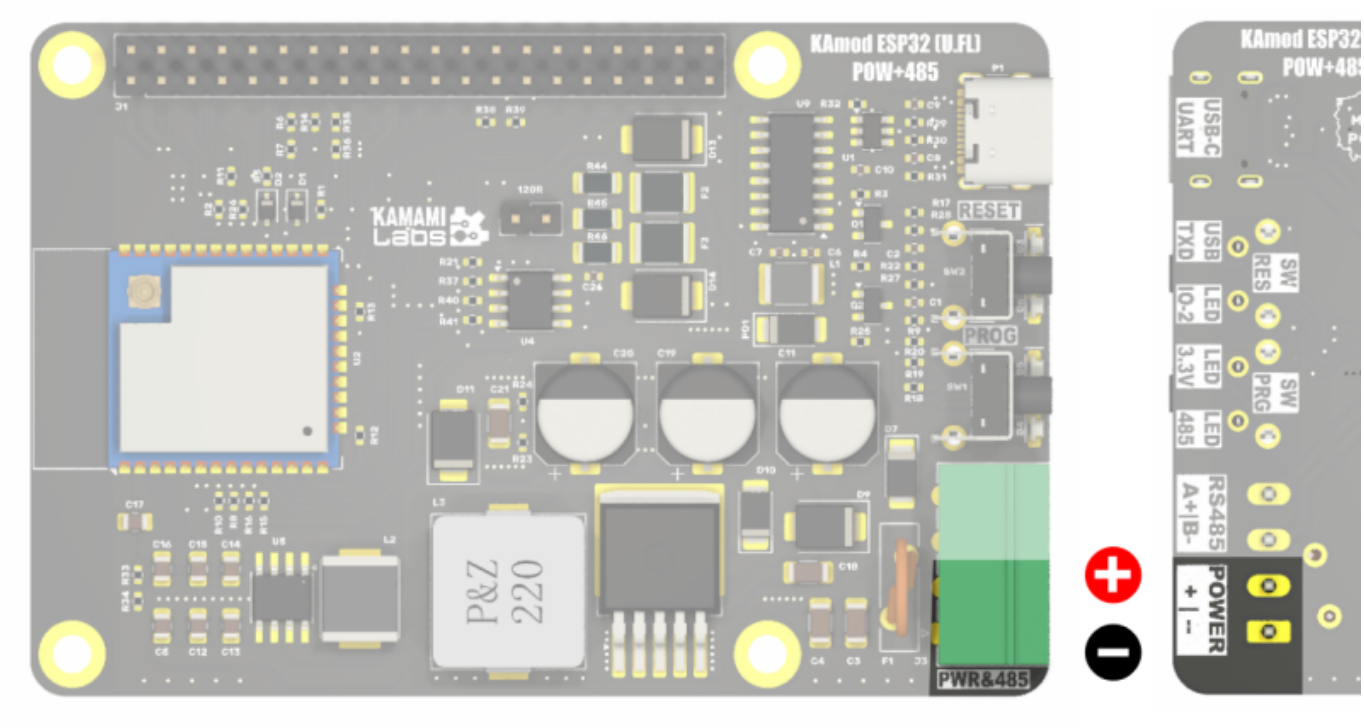
Connector	Function
POW&485 Phoenix MC3.81 mm	• Supplies power to the module • RS485 interface connector

The **POW&485** (J3) connector allows for connecting a direct current (DC) voltage in the range of 8...32 V, from which 5.1 V is generated to power all board components. When connecting voltage to the POW&485 connector, pay attention to its correct polarity. The marking on the bottom of the board: **POWER --|+** indicates the correct polarity:

- **--** - pin No. 1 is ground, negative power pole (GND),
- **+** - pin No. 2 is the positive power pole input.

Connecting power with parameters sufficient to obtain 3.3 V will be signaled by the D3 LED. Do not connect a voltage exceeding 34 V. The module includes overvoltage protection that will disconnect the power at a voltage higher than approx. 34 V.

The connected power source should have adequate power. For the module to operate with all parameters maintained (5.1 V / 5 A), the power of the source should not be less than 30 W. Connecting a lower power source will result in a lower maximum current value in the 5 V and 3.3 V circuits.



RS485 Interface

Connector	Function
POW&485 Phoenix MC 3.81 mm	• Supplies power to the module • RS485 interface connector

The KAmod ESP32 POW RS485 EANT board implements a complete RS485 interface circuit along with overvoltage protection elements. The ST485 chip is used as a driver, allowing the connection of up to 64 devices, operating in half-duplex mode, and implementing communication at a maximum speed of 1 Mbps. Data transmission to the bus is signaled by the flashing of the D4 LED.

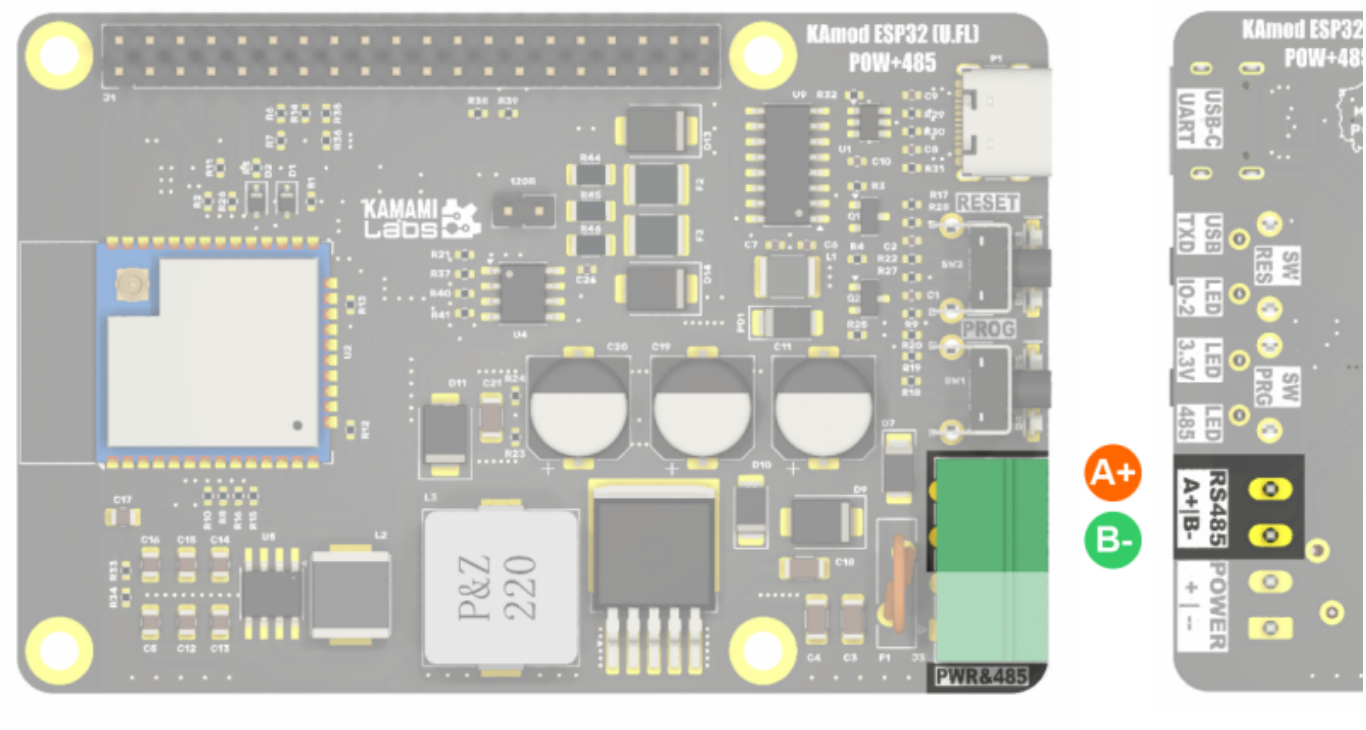
The connection of the RS485 interface to the ESP32 module is as follows:

RS485 Signal	ESP32 Module Pin
RO (read from RS485 bus)	GPIO27
DI (send to RS485 bus)	GPIO25
DE/RE (communication direction control; H – transmit / L - read)	GPIO26

The transmission medium is a 2-wire twisted pair, which should be connected to the **POW&485** (J3) connector:

- **B-**, pin No. 3, negative pole of the RS485 bus,
- **A+**, pin No. 4, positive pole of the RS485 bus.

Connection of all devices on the bus to a common ground/earth must also be ensured. The RS485 bus specification requires all connections to form a line without branches or loops, and 120 Ω resistors (bus terminators) must be connected at its ends. The KAmod ESP32 POW RS485 EANT board has a suitable resistor that can be connected to the bus by placing a jumper on the pins marked **120R**.



USB Interface

Connector	Function
P1 USB-C	• Implements USB-UART converter function • Enables programming of the ESP32 module • Acts as an alternative power input

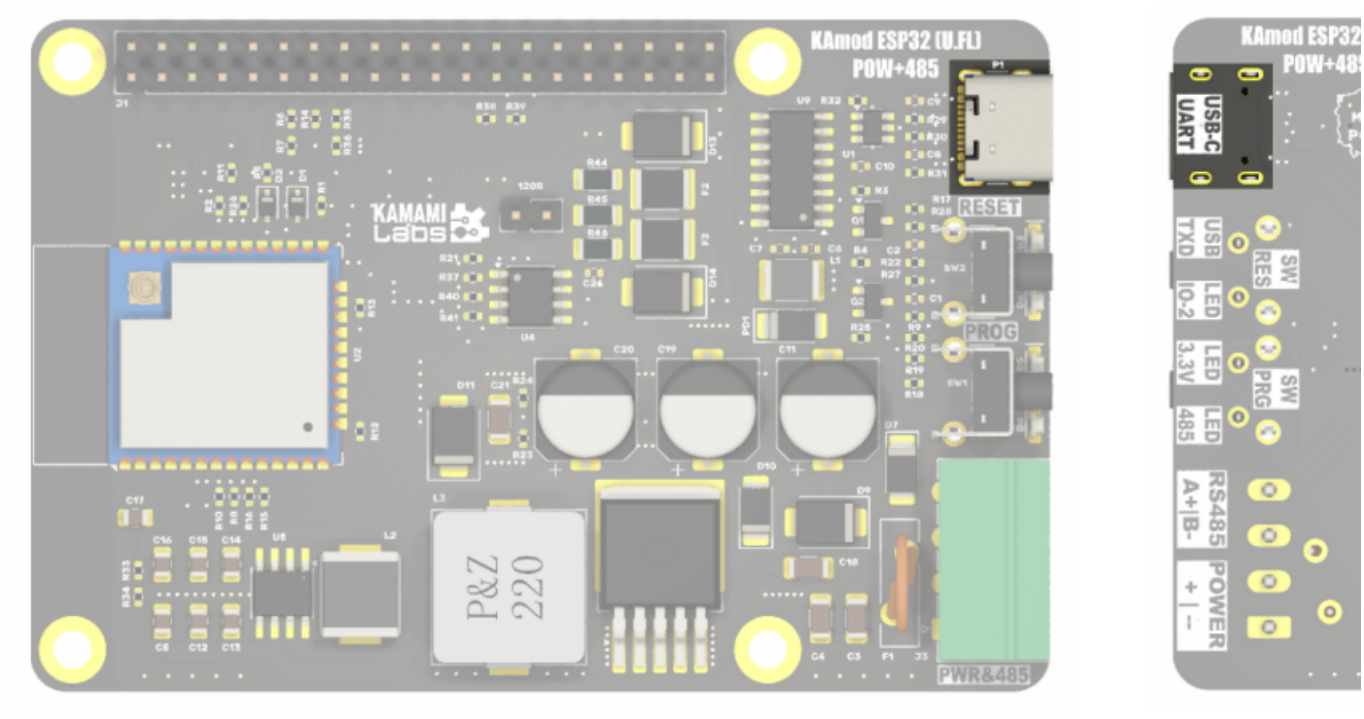
The USB-C type P1 connector is connected to a CH340 controller, which implements USB-UART converter functions. The UART interface can be used in the target application, but it also serves for programming the ESP32 module. The programming process can be fully automatic, as the CH340 controller controls the key pins of the ESP32 module (**GPIO0** – *Boot Select* and **EN** - *Chip Power-up*).

Signal connections between CH340 and ESP32 are as follows:

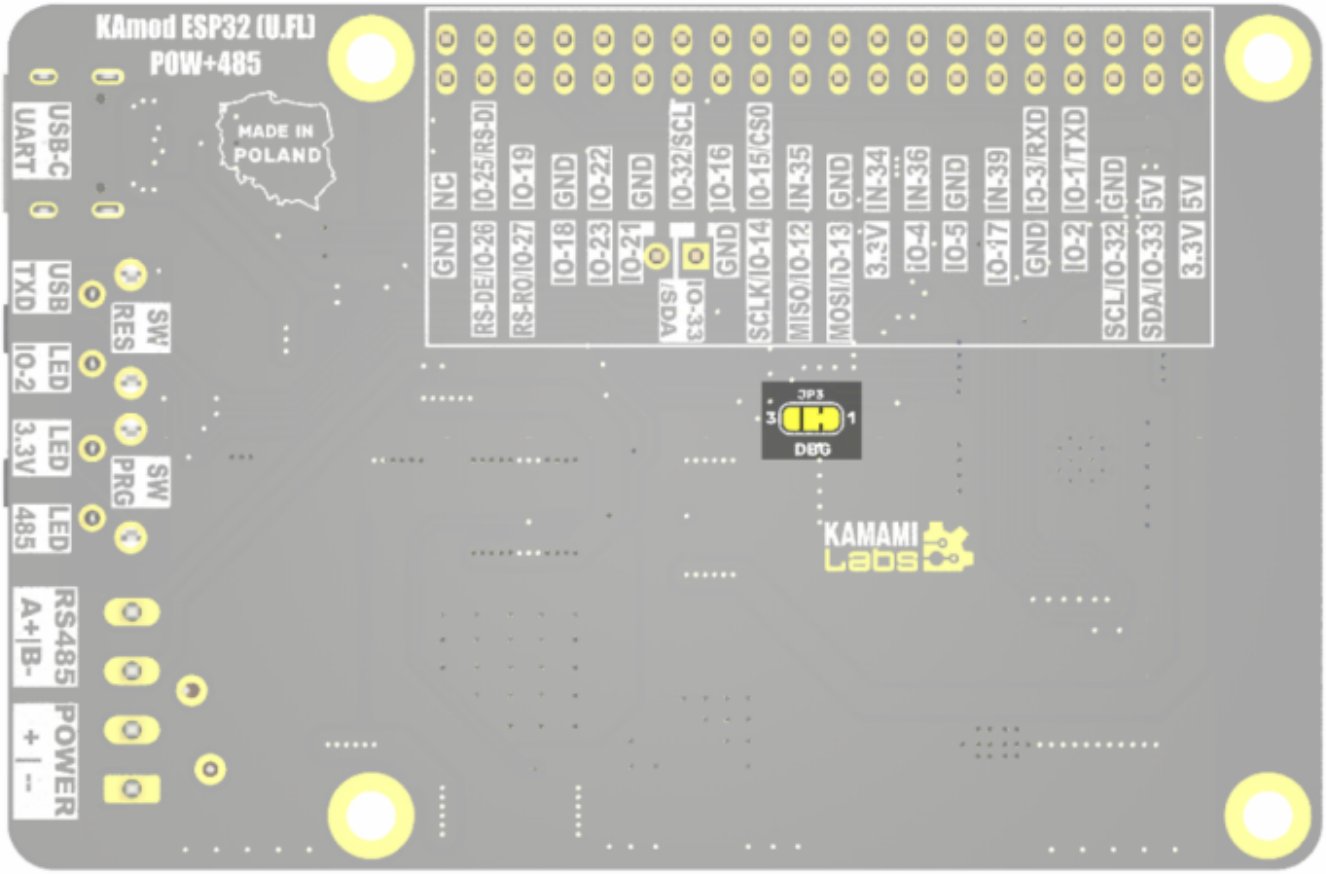
CH340 Controller Signal	ESP32 Module Pin
TXD (data output)	GPIO03 (UART0 RXD)
RXD (data input)	GPIO01 (UART0 TXD)
DTR (transmission control output)	EN (Chip Power-up)
RTS (transmission control output)	GPIO0 (Boot Select)

An LED (D3) is connected to the TXD line, signaling data reception from the USB interface. When using the USB-UART converter in the target application, ensure that the DTR and RTS lines remain unhandled (*Handshaking: None*).

The USB-C connector can serve as an alternative power input for the KAmod ESP32 POW RS485 board, however, the power circuit parameters will not be met in this case. The voltage on the 5 V line will be lower (approx. 4.5 V); the voltage on the 3.3 V line should not change; the current capacity of the 5 V and 3.3 V voltages will be much lower and will depend on the power supply used on the USB-C connector.



On the bottom side of the evaluation board, there is an SMD jumper marked as JP3 – DGB. It is connected to the GPIO15 pin of the ESP32 module (via a 10k resistor). By default, it is connected to pad number 1 (GND) and mutes system messages (Debugging Log). To enable system messages, cut the board surface between pads 1-2 of JP3 with a sharp tool and apply a drop of solder to connect the pads on the opposite side - pads 2-3 (to 3.3 V). Do not connect pads 2-3 without first separating pads 1-2.

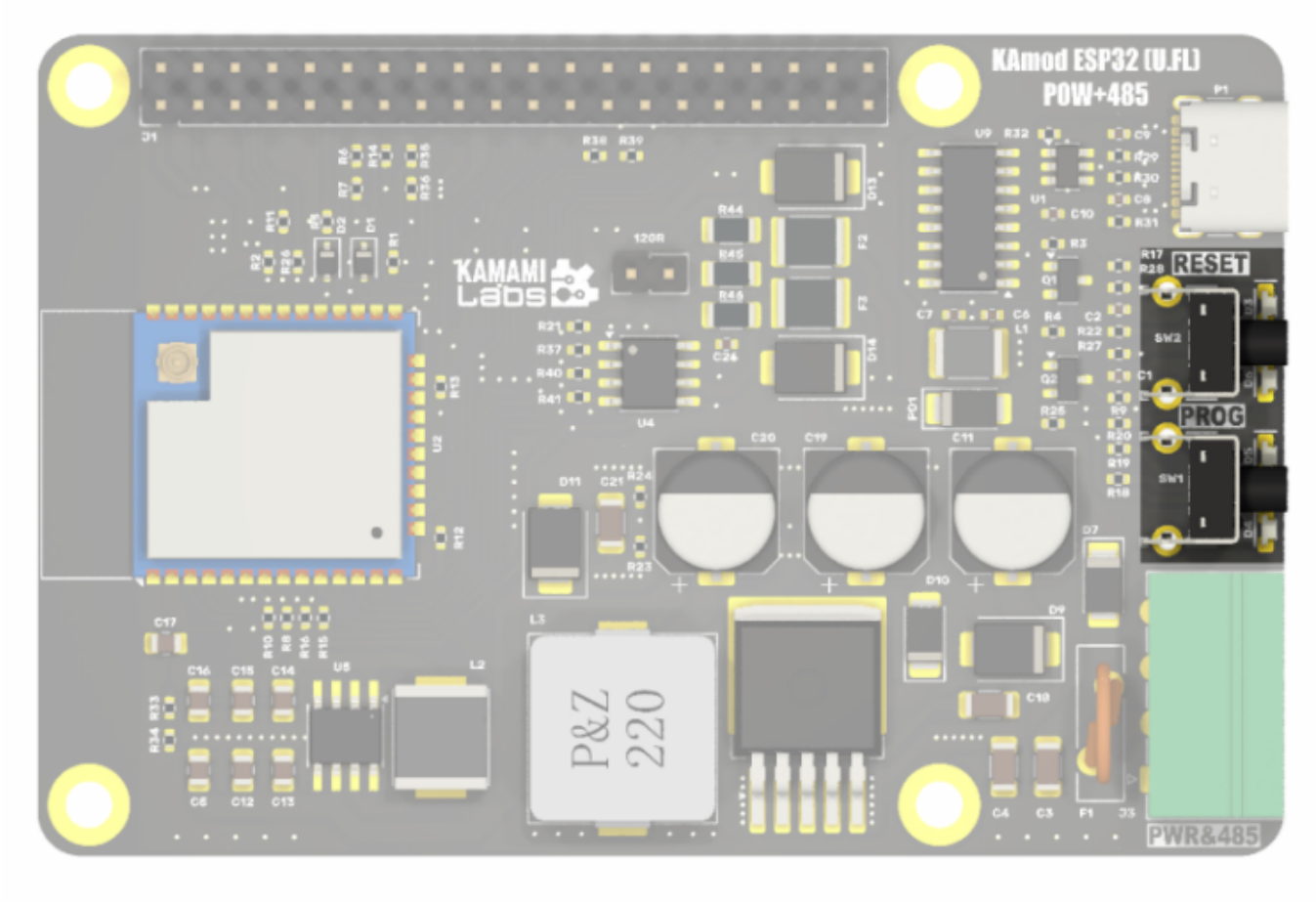


Reset and Programming Buttons

Component	Function
Button SW1 – **PROG**	Initiates UART programming mode (only during ESP32 module restart)
Button SW2 – **RESET**	Restarts the ESP32 module

The RESET button allows for restarting the ESP32 module. It is connected to the EN (*Chip Power-up*) line of the ESP32 module.

The PROG button allows entering the ESP32 module into programming mode. To do this, press the RESET button, then while holding RESET, hold the PROG button, and then release RESET while still holding PROG for a moment. This functionality may be useful if for some reason the programming mode is not initiated automatically via the USB-UART converter.

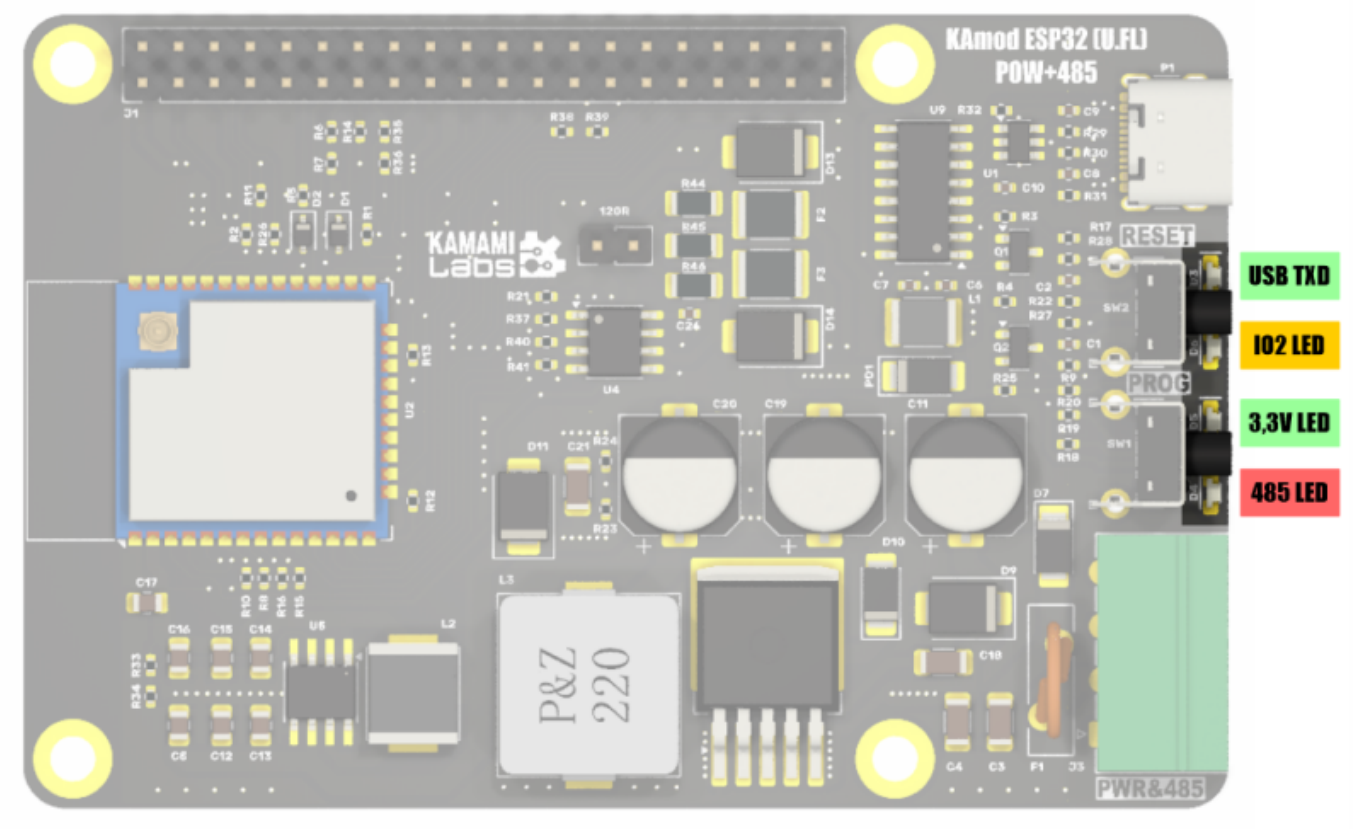


Signaling Indicators

Component	Function
D3- USB TXD	Flashing D3 indicates data transfer from USB to the ESP32 module
D4- LED 485	Flashing D4 indicates data transmission to the RS485 bus
D5- LED 3.3V	Glowing D5 indicates the operation of the power supply circuit and the presence of 3.3 V voltage
D6- LED IO-2	D6 is connected to the GPIO2 pin of the ESP32 module and its illumination can be triggered programmatically

The KAmod ESP32 POW RS485 board features 4 LEDs that signal the operation of various components as per the table above.

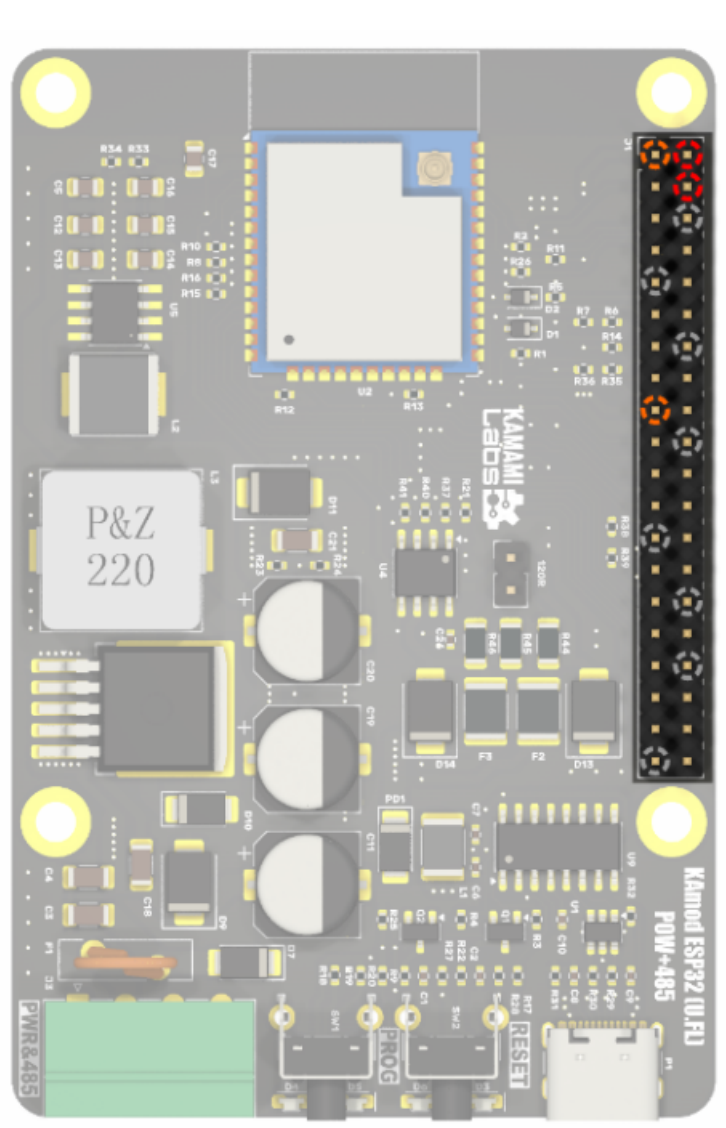
The D6 LED (LED IO-2) is connected to the GPIO2 pin of the ESP32 module. Lighting it up requires setting a high state on the GPIO2 pin in the software.



GPIO Connector in RPi Standard

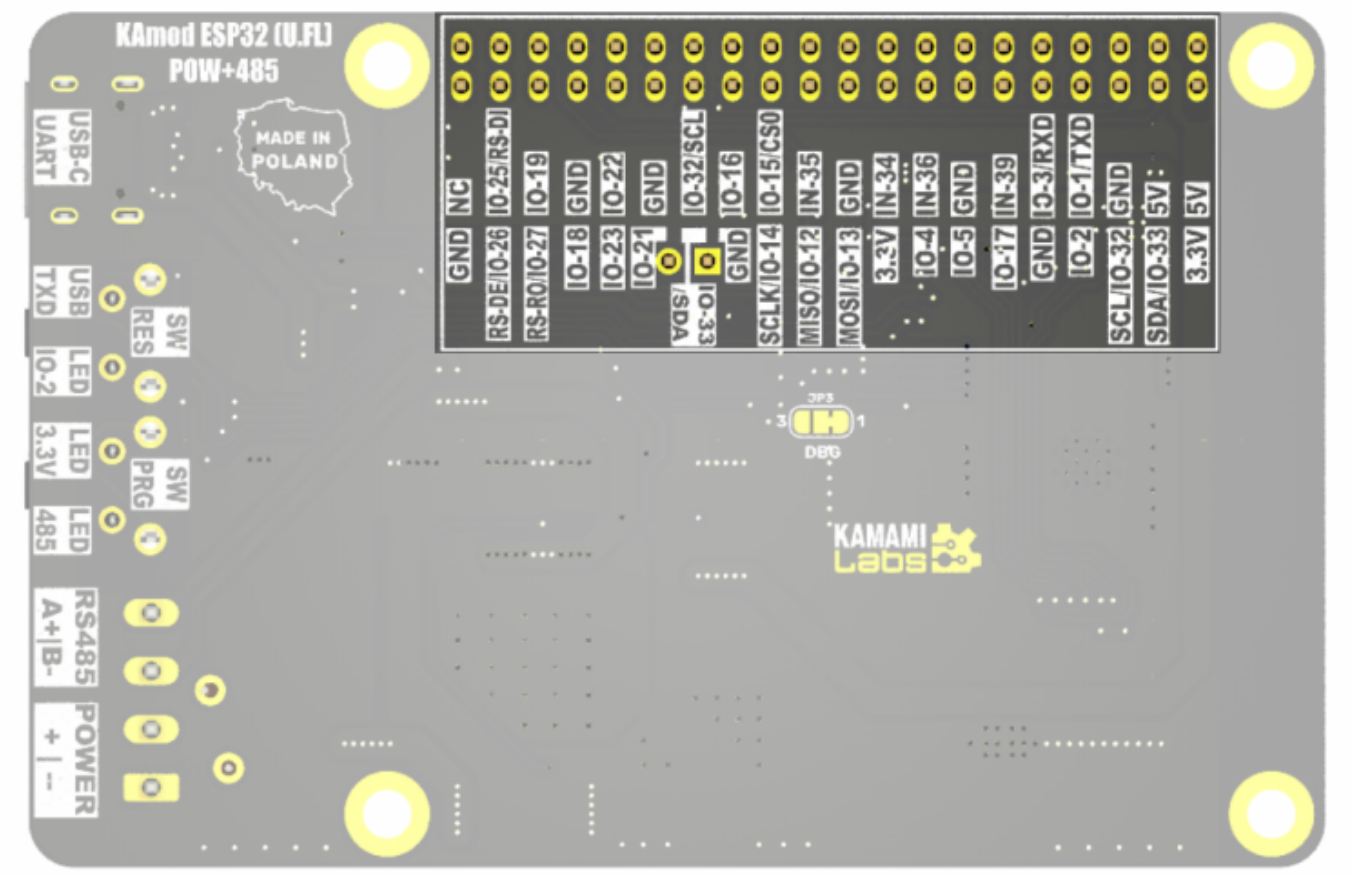
The GPIO connector (J1) in the Raspberry Pi standard contains 40 pins, to which 5 V, 3.3 V, GND power lines, and ESP32 GPIO pins are routed. The UART (TXD, RXD), I2C (SDA, SCL), and SPI (MOSI, MISO, SCLK, CS0) interface pins are arranged just like in the Raspberry Pi family boards.

A detailed description of the pins and their functions is shown in the figure and table below:



J1	
3,3 V	5 V
GPIO33 (SDA)	5 V
GPIO32 (SCL)	GND
GPIO02 (LED D6)	GPIO1 (TXD)
GND	GPIO3 (RXD)
GPIO17	GPIO39 (IN)
GPIO5	GND
GPIO4	GPIO36 (IN)
3,3 V	GPIO34 (IN)
GPIO13 (MOSI)	GND
GPIO12 (MISO)	GPIO35 (IN)
GPIO14 (SCLK)	GPIO15 (SPI CS0)
GND	GPIO16
GPIO33 (SDA)	GPIO32 (SCL)
GPIO21	GND
GPIO23	GPIO22
GPIO18	GND
GPIO27 (RS485 RO)	GPIO19
GPIO26 (RS485 DE)	GPIO25 (RS485 DI)
GND	NC (niepodłączony)

The pin description is also marked on the bottom of the KAmod ESP32 POW RS485 board:

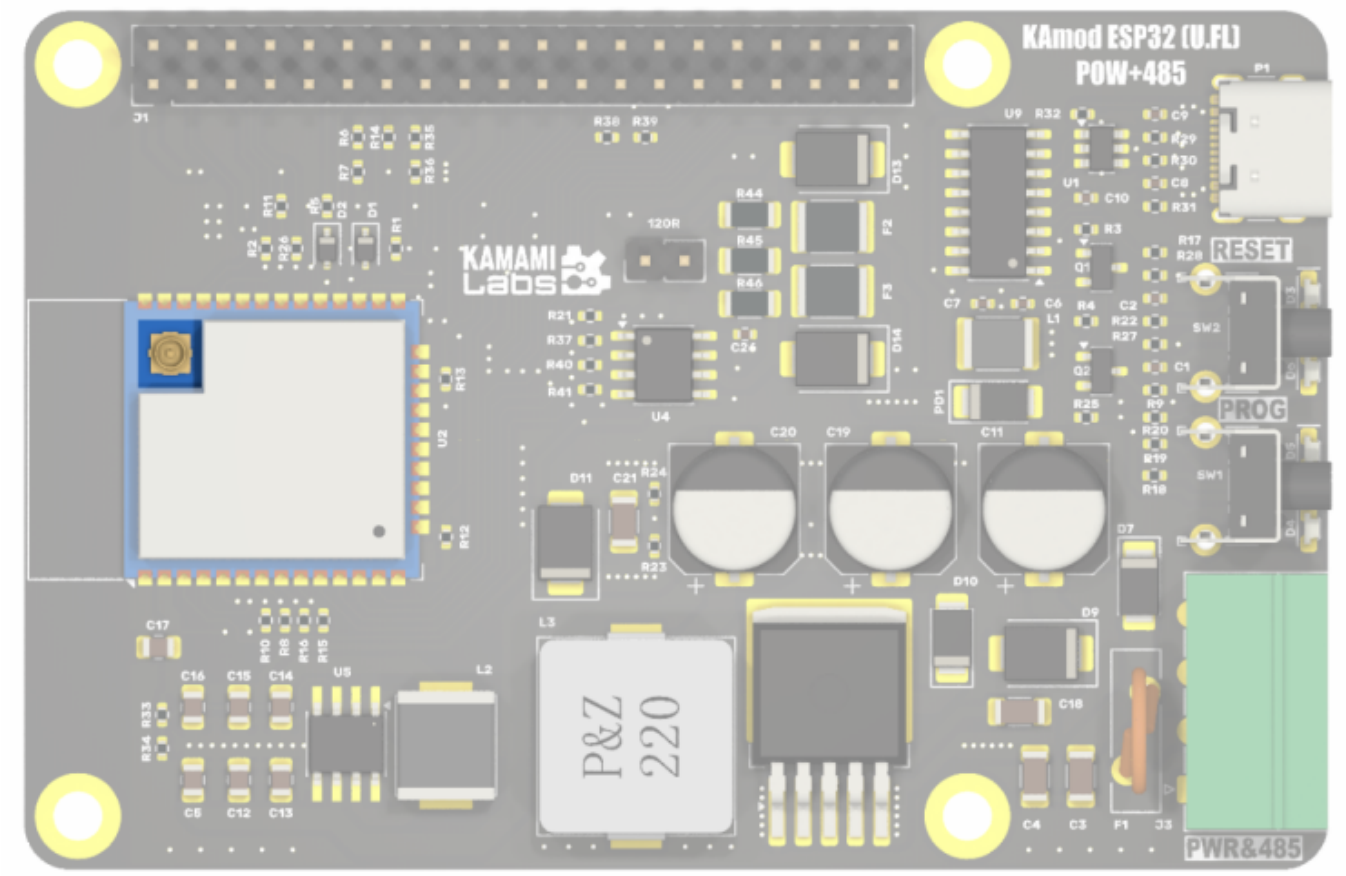


Notes on signals routed to the GPIO connector:

- GPIO ports 34, 35, 36, and 39 of the ESP32 module can operate only as digital or analog inputs – they are marked with the IN symbol.
- GPIO ports 34 and 35 are equipped with 10k pull-up resistors.
- GPIO ports 36 and 39 are equipped with voltage dividers (100k/10k), allowing connection of voltage up to a maximum value of 35 V.
- GPIO ports 32 and 33 (corrected from 32 and 32) are adapted for I2C bus functionality and include 2.2k pull-up resistors.
- GPIO1 and GPIO3 act as the UART interface and are connected to the USB-UART converter module and in parallel to the J2 GPIO connector. The UART interface sends/reads data to/from the J2 GPIO connector and the USB-UART converter simultaneously.

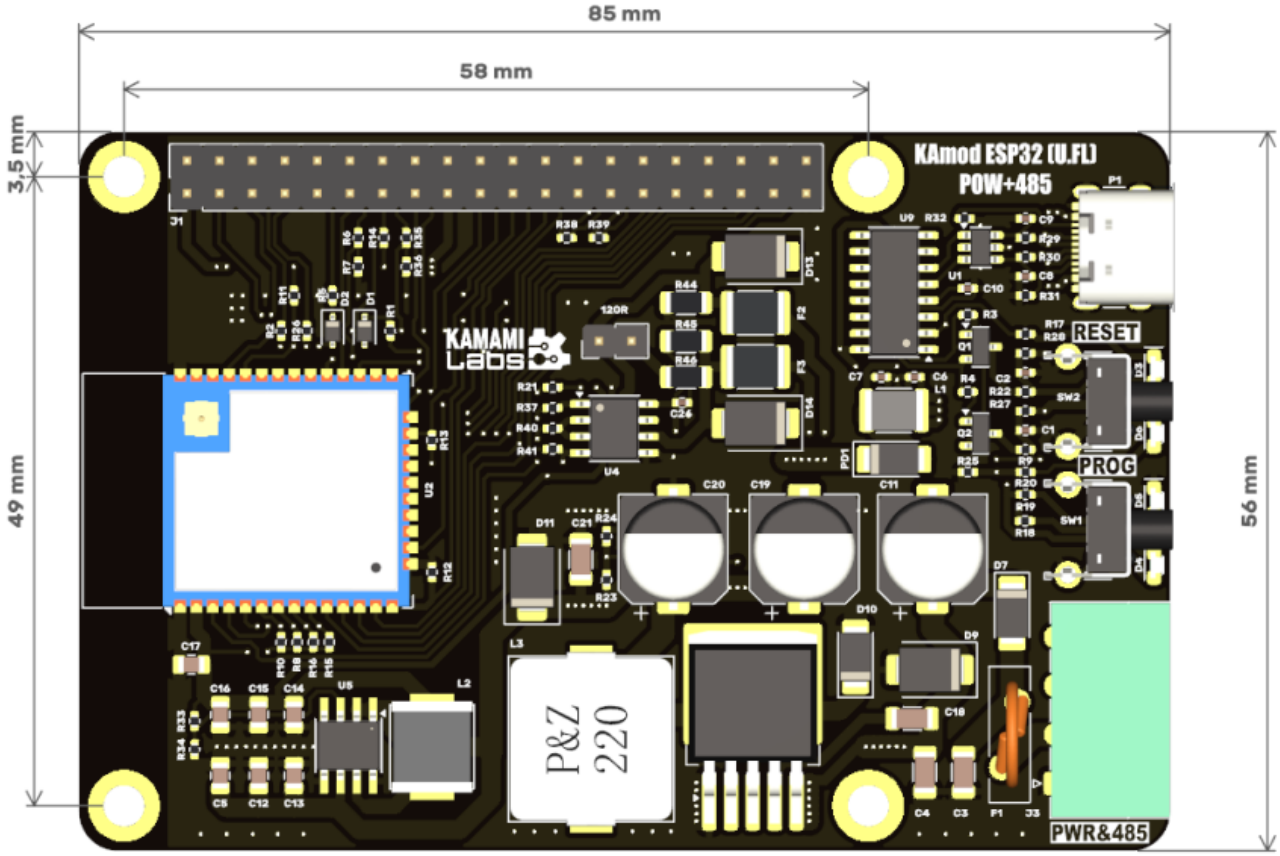
Antenna Connector

On the ESP32-WROOM-32U module board, there is a U.FL (IPX) type connector, to which a Wi-Fi antenna intended for the 2.4 GHz band should be connected.



Dimensions

The dimensions of the KAmod ESP32 POW RS485 EANT board are 85x56 mm. The maximum height is approx. 20 mm. There are 4 mounting holes on the board with a diameter of 3 mm, arranged similarly to Raspberry Pi family boards.



Test Program

The test program code is below; it can be compiled in the Arduino environment.

```
#include <WiFi.h>
#include <NetworkClient.h>
#include <WiFiAP.h>
#include <HardwareSerial.h>

#define LED_PIN 2

//Server
const char *ssid = "KAmodESP32";
const char *password = "12345678";
NetworkServer server(80);

IPAddress AP_LOCAL_IP(192, 168, 1, 1);
IPAddress AP_GATEWAY_IP(192, 168, 1, 254);
IPAddress AP_NETWORK_MASK(255, 255, 255, 0);

//RS485
#define RS485_RXD 27 //RO PIN
#define RS485_TXD 25 //DI PIN
#define RS485_RE_DE 26 //RE & DE PIN
#define RX_BUFF_SIZE 64
```

```
HardwareSerial RS485Port(2);

void setup() {
  //Serial
  Serial.begin(115200);
  Serial.println("Hello. Configuring access point...");

  //Server
  WiFi.softAPConfig(AP_LOCAL_IP, AP_GATEWAY_IP, AP_NETWORK_MASK);
  if (!WiFi.softAP(ssid, password)) {
    log_e("Soft AP creation failed.");
    while (1);
  }
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  Serial.print("AP Mac Address: ");
  Serial.println(WiFi.softAPmacAddress());
  server.begin();

  //RS485
  //pinMode(RS485_RE_DE, OUTPUT);
  //digitalWrite(RS485_RE_DE, LOW);
  RS485Port.setPins(RS485_RXD, RS485_TXD, RS485_RE_DE);
  RS485Port.setHwFlowCtrlMode(UART_HW_FLOWCTRL_CTS);
  RS485Port.setMode(UART_MODE_RS485_HALF_DUPLEX);
  RS485Port.begin(115200, SERIAL_8N1, RS485_RXD, RS485_TXD);
  RS485Port.println("RS485 Hello");

  //LED
  pinMode(LED_PIN, OUTPUT);
  for(int i=0; i<10; i++){
    digitalWrite(LED_PIN, HIGH);
    delay(200);
    digitalWrite(LED_PIN, LOW);
    delay(200);
  }
}
void loop() {
 NetworkClient client = server.accept();

 if (client) {
  Serial.println("********New Client********");

  String currentLine = "";
  while (client.connected()) {
   if (client.available()) {
    char c = client.read();
    Serial.write(c);
    if (c == '\n') {

     if (currentLine.length() == 0) {
      client.println("HTTP/1.1 200 OK");
      client.println("Content-type:text/html");
      client.println();
      client.print("Click <a href=\"/H\">here</a> to turn ON the LED.<br>");
      client.print("Click <a href=\"/L\">here</a> to turn OFF the LED.<br>");
      client.print("Click <a href=\"/RS485\">here</a> to write to RS485.<br>");
      client.println();
```

```
      break;
     } else {
      currentLine = "";
     }
    } else if (c != '\r') {
     currentLine += c;
    }

    //if (currentLine.endsWith("GET /H")) {
    if (currentLine.indexOf("GET /H") >= 0) {
     digitalWrite(LED_PIN, HIGH);
    }
    //if (currentLine.endsWith("GET /L")) {
    if (currentLine.indexOf("GET /L") >= 0) {
     digitalWrite(LED_PIN, LOW);
    }
    if (currentLine.endsWith("GET /RS485")) {
     RS485Port.println("You send message via RS485");
    }
   }
  }
  client.stop();
  Serial.println("********Client Disconnected********");
 } else {
  RS485Port.println("Tick...");
  delay(1000);
 }
}
```

The test program configures the hardware UART 2 interface to work as an RS485 interface with hardware transmission direction control:

```
#define RS485_RXD 27 //RO PIN
#define RS485_TXD 25 //DI PIN
#define RS485_RE_DE 26 //RE & DE PIN
#define RX_BUFF_SIZE 64
HardwareSerial RS485Port(2);
...
RS485Port.setPins(RS485_RXD, RS485_TXD, RS485_RE_DE);
RS485Port.setHwFlowCtrlMode(UART_HW_FLOWCTRL_CTS);
RS485Port.setMode(UART_MODE_RS485_HALF_DUPLEX);
RS485Port.begin(115200, SERIAL_8N1, RS485_RXD, RS485_TXD);
RS485Port.println("RS485 Hello");
```

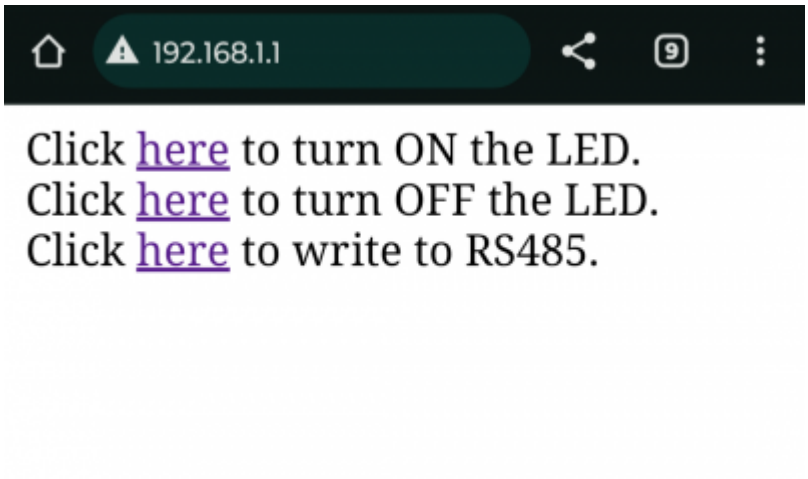
Furthermore, the test program starts the ESP32 module in Access Point mode and acts as a web server:

```
const char *ssid = "KAmodESP32";
const char *password = "12345678";
NetworkServer server(80);
IPAddress AP_LOCAL_IP(192, 168, 1, 1);
IPAddress AP_GATEWAY_IP(192, 168, 1, 254);
IPAddress AP_NETWORK_MASK(255, 255, 255, 0);

WiFi.softAPConfig(AP_LOCAL_IP, AP_GATEWAY_IP, AP_NETWORK_MASK);
```

```
if (!WiFi.softAP(ssid, password)) {
  log_e("Soft AP creation failed.");
  while (1);
}
IPAddress myIP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(myIP);
Serial.print("AP Mac Address: ");
Serial.println(WiFi.softAPmacAddress());
server.begin();
```

After running the test program, a Wi-Fi network named (SSID): *KAmodESP32* with password: 12345678 will be created. Connect to this network using a smartphone and enter the IP address in a web browser: 192.168.1.1. A very simple website will be displayed, allowing you to control the D6 LED or send a simple message via the RS485 interface:



Links

- Sample project
- CAD Model (STEP)
- XL4015 datasheet
- CH340 datasheet
- ESP32 datasheet
- STS10 datasheet

BTC Korporacja
05-120 Legionowo
ul. Lwowska 5
tel.: (22) 767-36-20
faks: (22) 767-36-33
e-mail:
sprzedaz@kamami.pl
https://kamami.pl