KAMAMI

KAmod ESP32 ETH POE (PL)

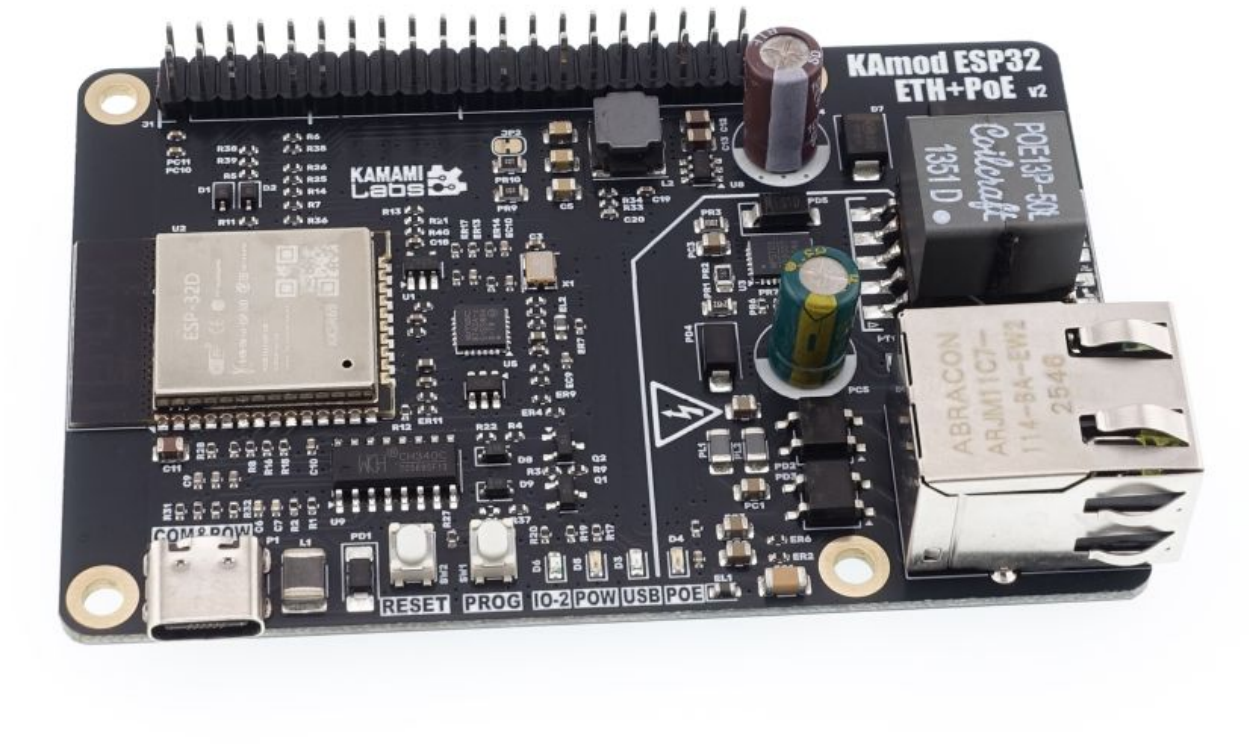
Rev. 20260404133337
Źródło: https://wiki.kamamilabs.com/index.php?title=KAmod_ESP32_ETH_POE_(PL)

Spis treści

Opis

KAmod ESP32 ETH POE - Płytka ewaluacyjna z modułem ESP32-WROOM połączonym z interfejsem Ethernet oraz układem zasilania PoE

Na płytce KAmod ESP32 ETH+POE znajduje się moduł ESP32-WROOM umożliwiający komunikację w sieci bezprzewodowej Wi-Fi 2,4 GHz, jednak został połączony z interfejsem przewodowym Ethernet z typowym złączem RJ45. Programowanie modułu ESP32 umożliwia konwerter USB-UART ze złączem USB-C. Płytkę uzupełnia układ zasilania PoE – Power over Ethernet, dzięki czemu zasilanie modułu może być dostarczane z instalacji internetowej. Konstrukcja płytki odpowiada SBC rodziny Raspberry Pi - ma wymiary 81x56 mm, a na charakterystycznym, 40-stykowym złączu zostały wyprowadzone wszystkie istotne porty I/O oraz napięcia zasilające 5 V oraz 3,3 V, które mogą zasilać dodatkowe komponenty dołączone do płytki.



Podstawowe cechy i parametry

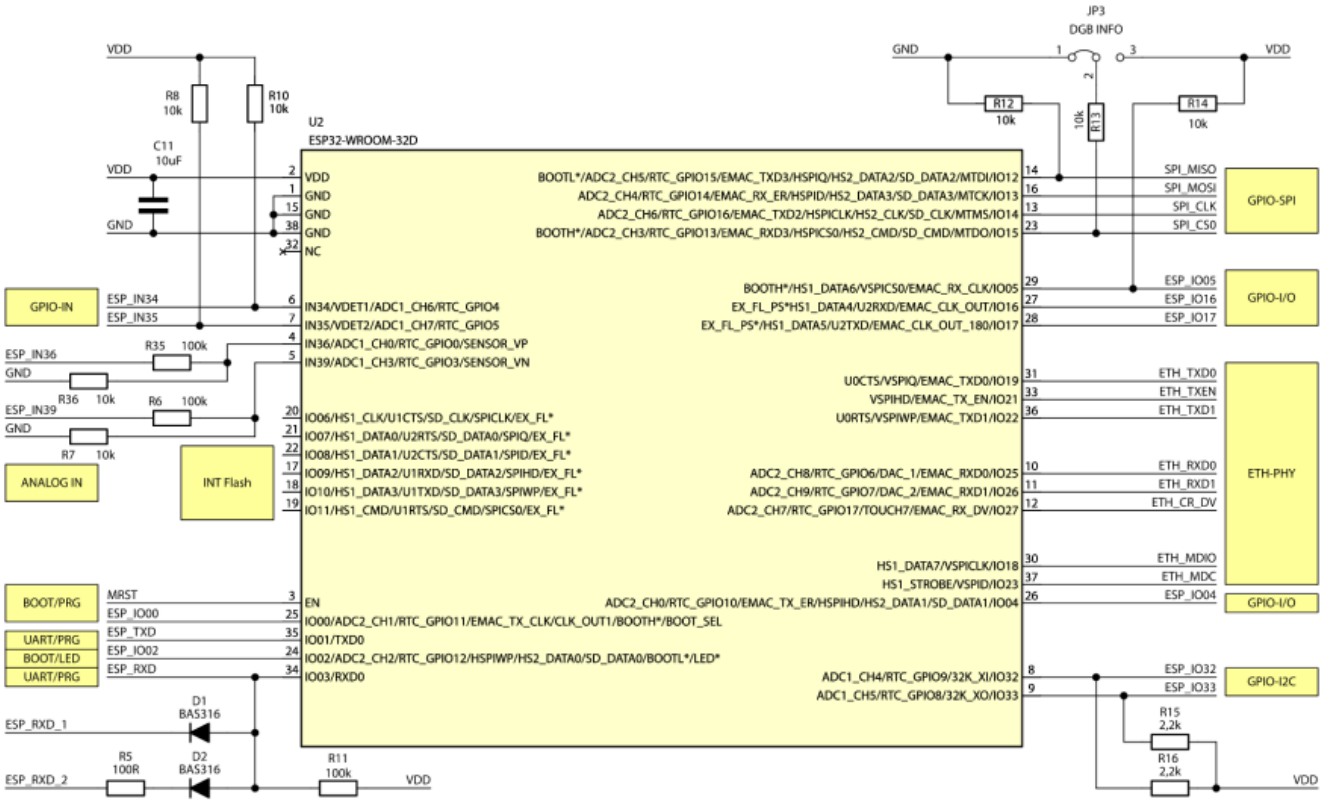
- Moduł ESP32-WROOM-32 - wydajny mikrokontroler z komunikacją Wi-Fi
- Mikroprocesor Xtensa Dual-Core 32-bit LX6, max 240 MHz
- Rozmiar pamięci: 520 kB SRAM, 4 MB SPI Flash
- Komunikacja Wi-Fi 2,4 GHz, IEEE 802.11 b/g/n oraz Bluetooth 4.2 LE
- Interfejs Ethernet na bazie układu LAN8742 (100/10 Mb; full/half duplex)
- Zintegrowany konwerter UART-USB ze złączem USB-C umożliwiający programowanie modułu ESP32
- Układ zasilania PoE, kompatybilny ze standardem IEEE 802.3af/at Class 0
- Dostarcza stabilizowanych napięć 5 V oraz 3,3 V o łącznym prądzie do 1 A
- Zabezpieczenie przepięciowe, przeciążeniowe oraz termiczne
- Na 40-stykowe złącze w standardzie Raspberry Pi zostały wyprowadzone wszystkie istotne porty I/O oraz napięcia zasilające
- Wymiary płytki: 85x56 mm, wysokość ok. 20 mm

Wyposażenie standardowe

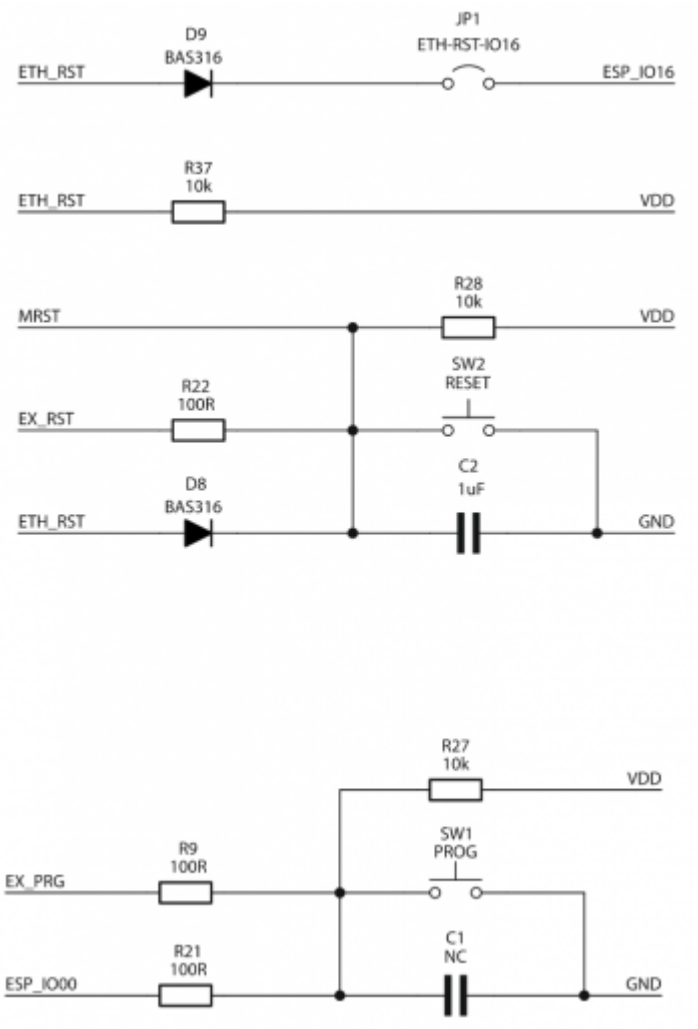
Kod	Opis
KAmodESP32 ETH+POE	Zmontowany i uruchomiony moduł

Schemat elektryczny

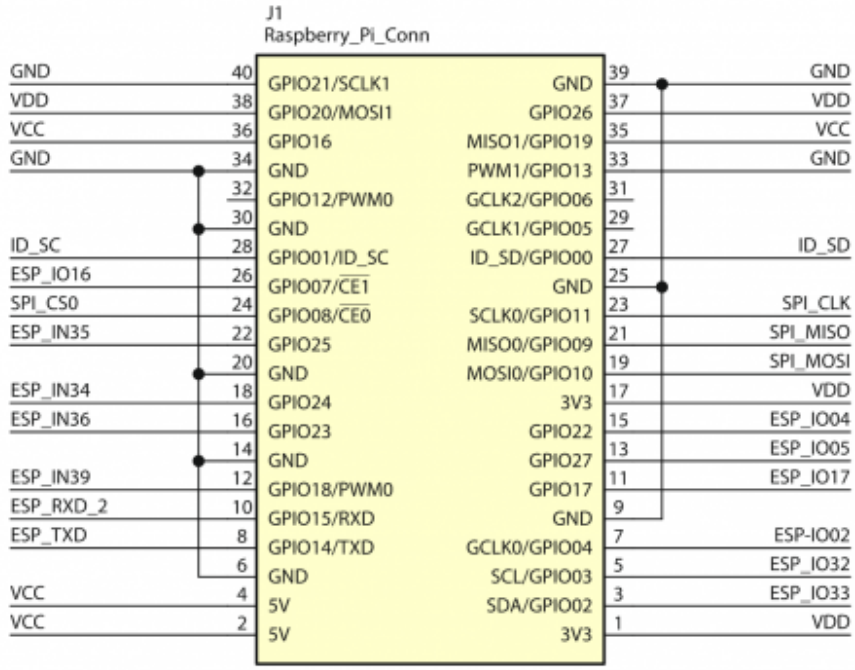
Moduł ESP32



Elementy odpowiedzialne za funkcje resetu i programowania

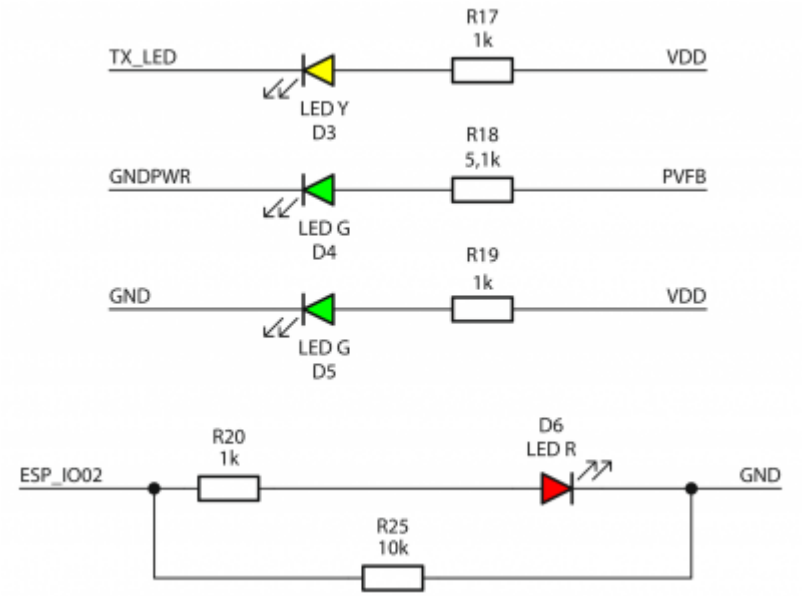


Złącze GPIO

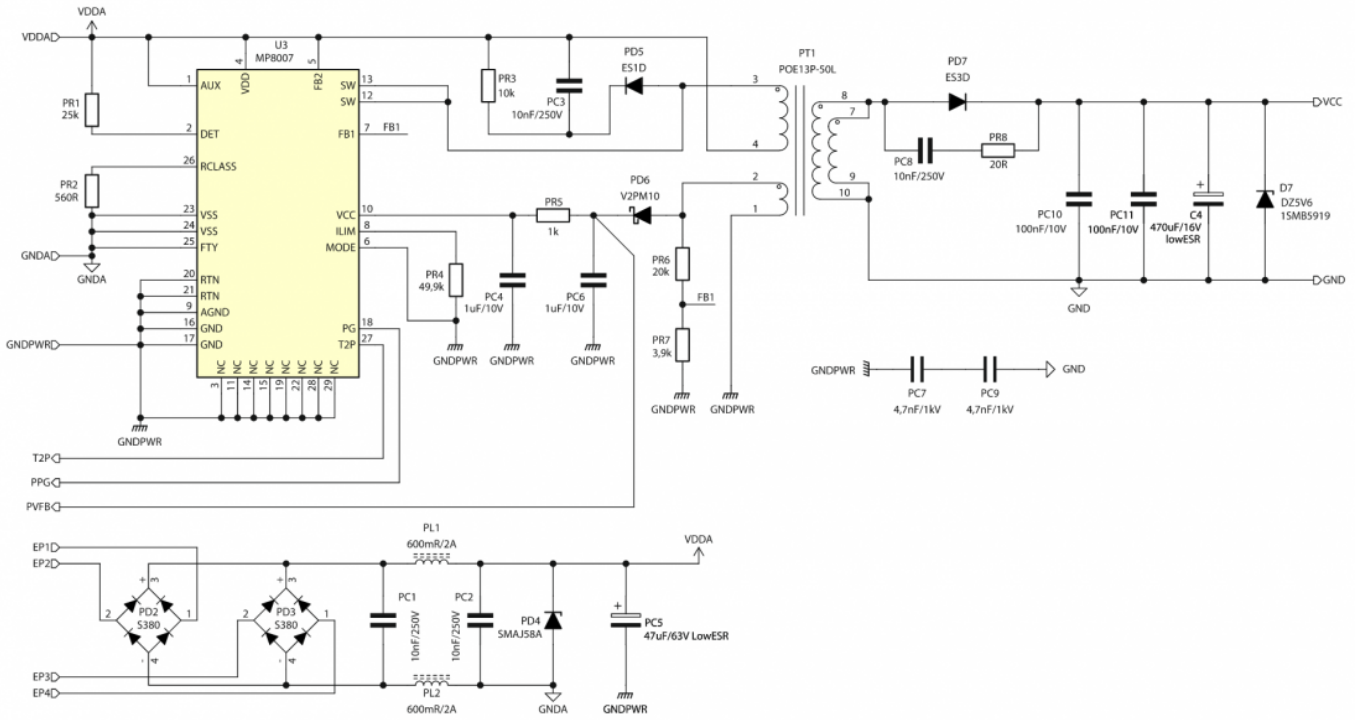

J1
Raspberry_Pi_Conn
GND 40
VDD 38
VCC 36
GND 34
32
30
ID_SC 28
ESP_IO16 26
SPI_CS0 24
ESP_IN35 22
20
ESP_IN34 18
ESP_IN36 16
14
ESP_IN39 12
ESP_RXD_2 10
ESP_TXD 8
6
VCC 4
VCC 2
GPIO21/SCLK1
GPIO20/MOSI1
GPIO16
GND
GPIO12/PWM0
GND
GPIO01/ID_SC
GPIO07/CE1
GPIO08/CE0
GPIO25
GND
GPIO24
GPIO23
GND
GPIO18/PWM0
GPIO15/RXD
GPIO14/TXD
GND
5V
5V
GND 39
GPIO26 37
MISO1/GPIO19 35
PWM1/GPIO13 33
GCLK2/GPIO06 31
GCLK1/GPIO05 29
ID_SD/GPIO00 27
GND 25
SCLK0/GPIO11 23
MISO0/GPIO09 21
MOSI0/GPIO10 19
3V3 17
GPIO22 15
GPIO27 13
GPIO17 11
GND 9
GCLK0/GPIO04 7
SCL/GPIO03 5
SDA/GPIO02 3
3V3 1
GND
VDD
VCC
GND
ID_SD
SPI_CLK
SPI_MISO
SPI_MOSI
VDD
ESP_IO04
ESP_IO05
ESP_IO17
ESP-IO02
ESP_IO32
ESP_IO33
VDD


R38
100R
ESP_IO32
ID_SC
R39
100R
ESP_IO33
ID_SD
R26
100R
ESP_IO02
ESP-IO02

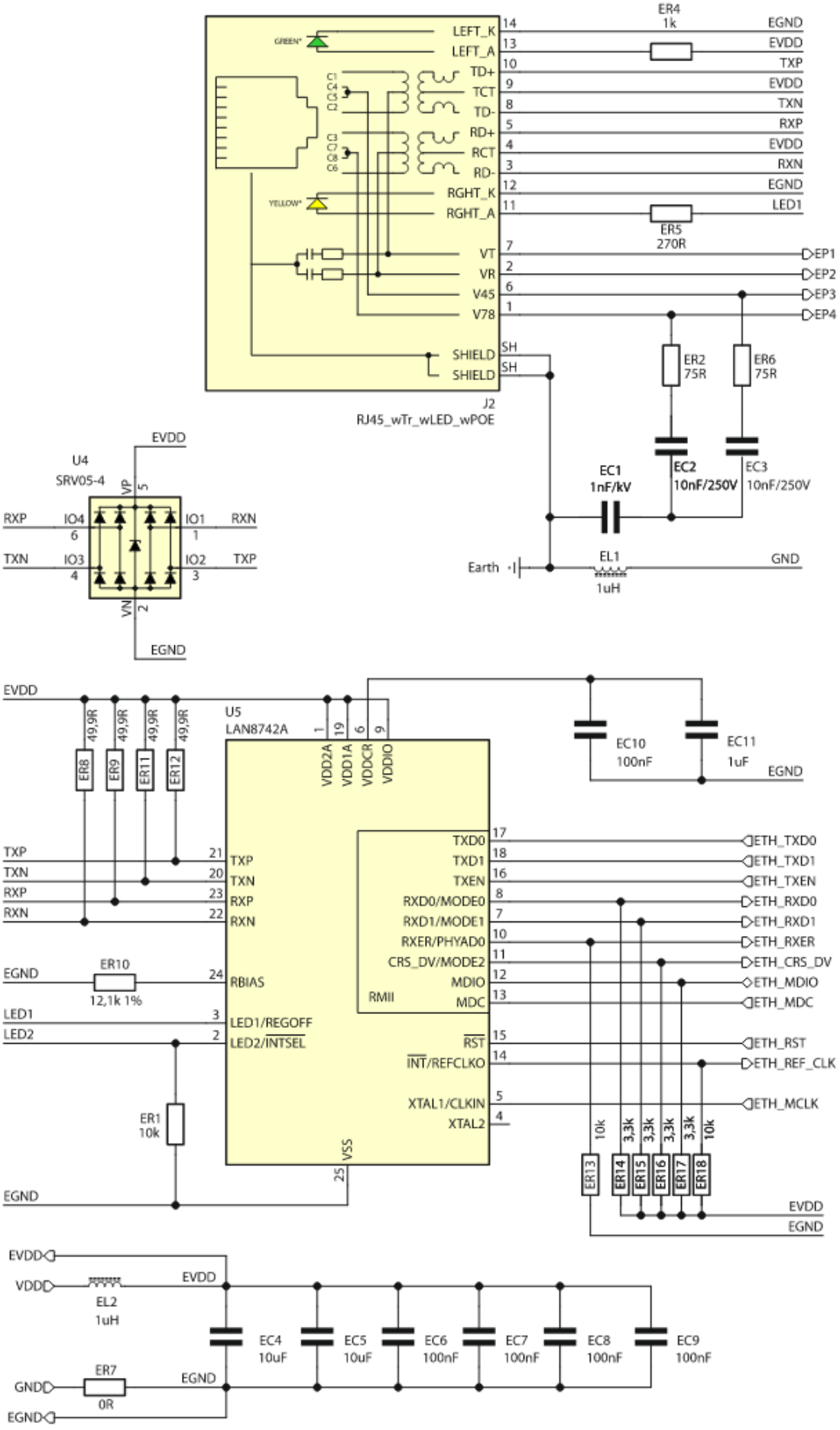
Diody sygnalizacyjne



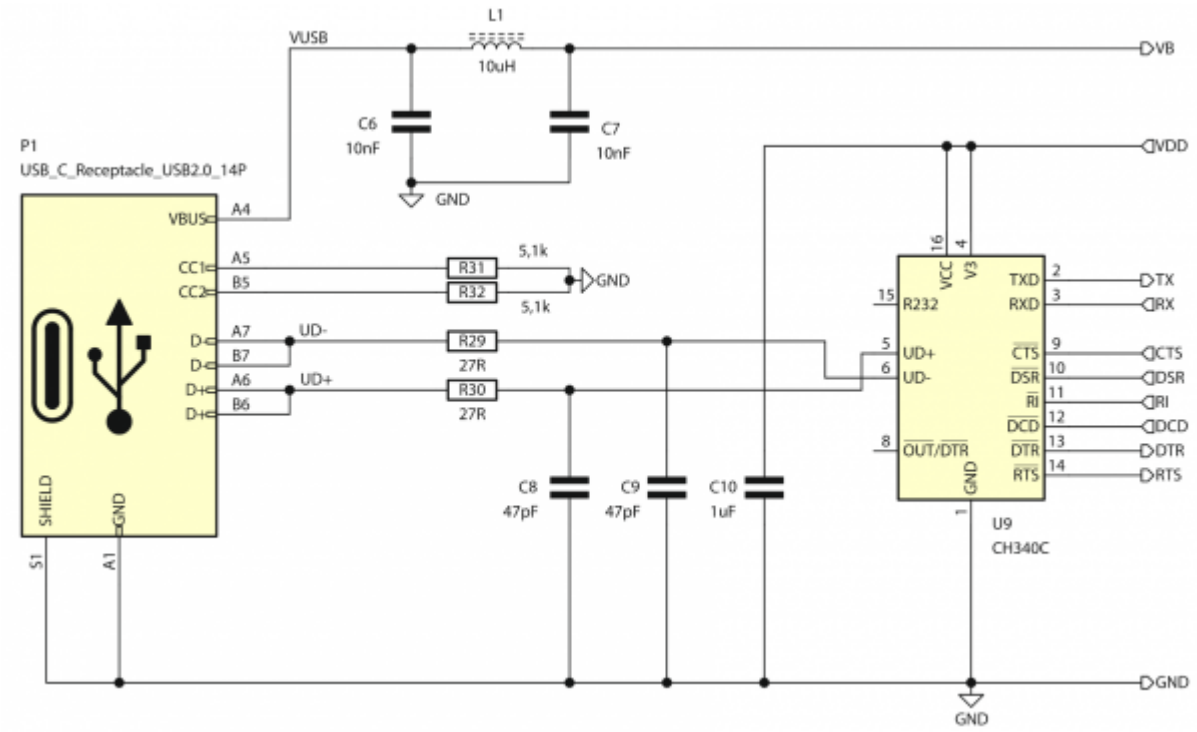
Interfejs Ethernet



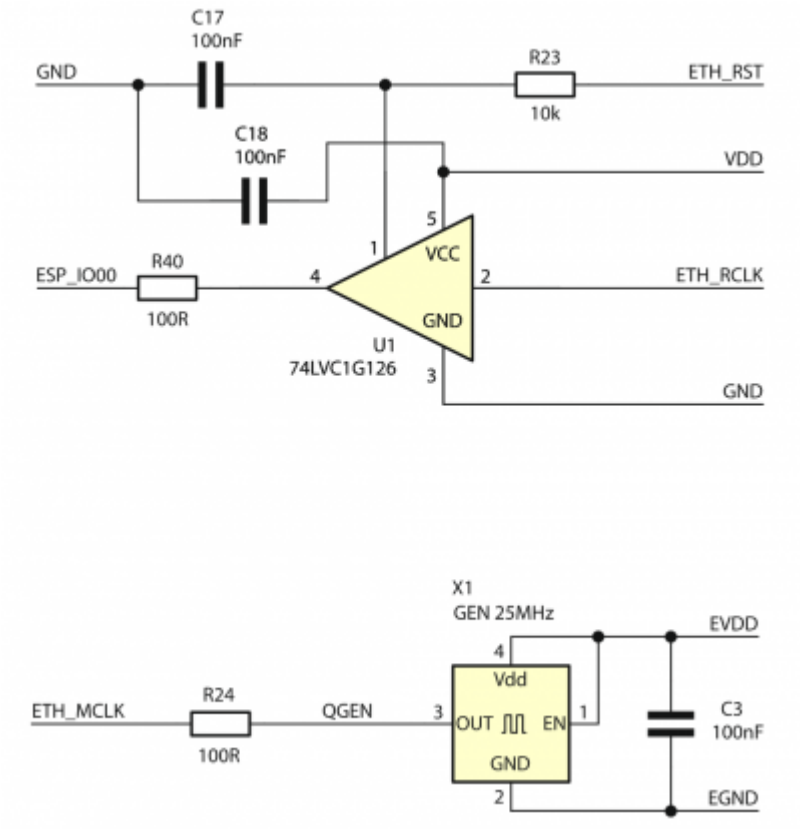
Zasilanie PoE


J2
RJ45_wTr_wLED_wPOE
LEFT_K
LEFT_A
TD+
TCT
TD-
RD+
RCT
RD-
RGHT_K
RGHT_A
VT
VR
V45
V78
SHIELD
SH
ER4
1k
ER5
270R
ER2
75R
ER6
75R
EC1
1nF/kV
EC2
10nF/250V
EC3
10nF/250V
EL1
1uH
Earth
GND
EVDD
EGND
EP1
EP2
EP3
EP4
U4
SRV05-4
RXP
RXN
TXN
TXP
U5
LAN8742A
ER8
ER9
ER11
ER12
49,9R
EC10
100nF
EC11
1uF
TXD0
TXD1
TXEN
RXD0/MODE0
RXD1/MODE1
RXER/PHYAD0
CRS_DV/MODE2
MDIO
MDC
RMII
RST
INT/REFCLKO
XTAL1/CLKIN
XTAL2
RBIAS
LED1/REGOFF
LED2/INTSEL
VSS
ER10
12,1k 1%
ER1
10k
ETH_TXD0
ETH_TXD1
ETH_TXEN
ETH_RXD0
ETH_RXD1
ETH_RXER
ETH_CRS_DV
ETH_MDIO
ETH_MDC
ETH_RST
ETH_REF_CLK
ETH_MCLK
ER13
ER14
ER15
ER16
ER17
ER18
10k
3,3k
EL2
1uH
VDD
EC4
10uF
EC5
10uF
EC6
100nF
EC7
100nF
EC8
100nF
EC9
100nF
ER7
0R

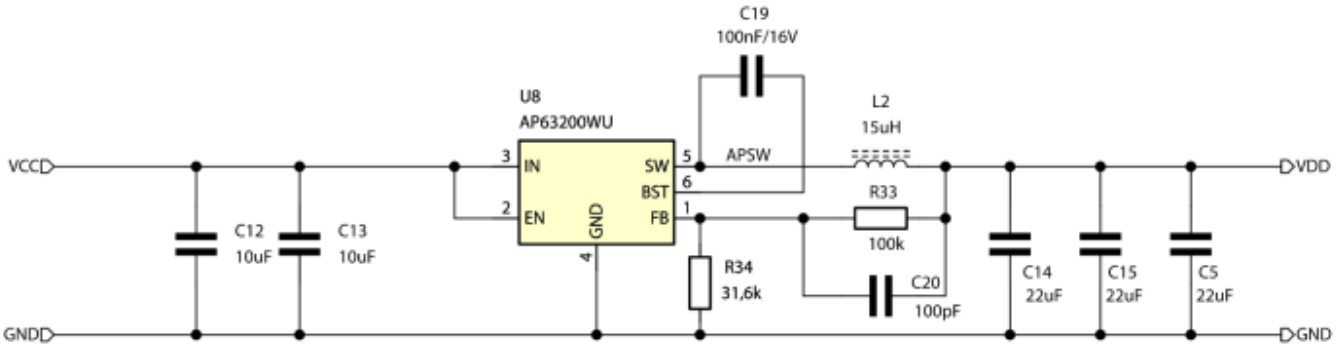
Interfejs USB-UART



Źródło sygnału taktującego



Blok zasilania o napięciu 3,3 V



Funkcje przypisane liniom GPIO

W tabeli opisano funkcje, które zostały przydzielone poszczególnym liniom GPIO modułu ESP 32 na płytce KAmod ESP32 ETH+POE. Niektóre wyprowadzenia mają dodatkowe właściwości, co również zostało dokładnie opisane.

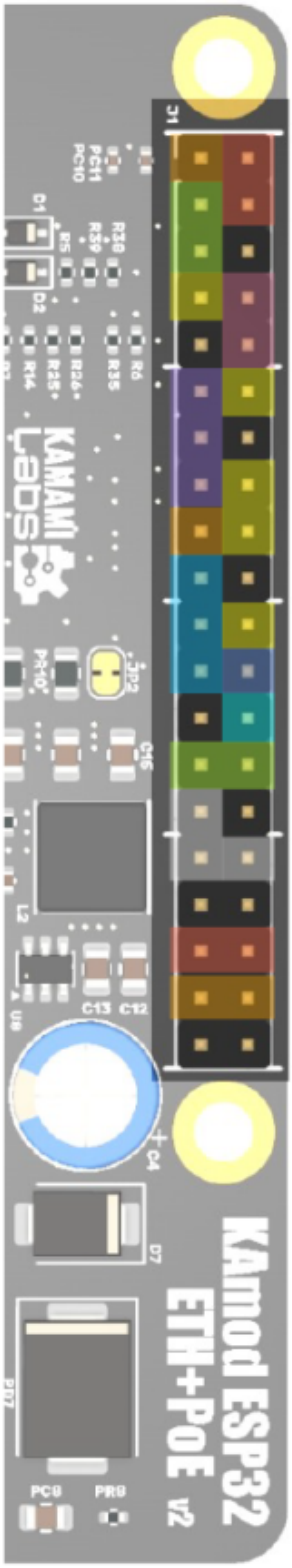
GPIO	Oznaczenie/funkcja	Opis	Numer pinu RPi Conn (40-pin)
00	BOOT SEL/RMII CLK	Boot Select - niski stan na tym wyprowadzeniu w chwili restartu modułu ESP32 uruchamia bootloader do programowania modułu ESP32 RMII CLK - w czasie normalnej pracy jest to wejście sygnału taktującego 50 MHz dla interfejsu RMII	-
01	UART TX	Wyjście szeregowego interfejsu UART, podłączone do konwertera UART-USB oraz do złącza RPi Conn J1-8	8
02	BOOT LOW/ LED D6	Wymagany jest stan niski na tym wejściu w chwili restartu modułu ESP32, dlatego zastosowano rezystor pull-down 10 k. W czasie normalnej pracy steruje diodą LED D6 - stan wysoki powoduje świecenie diody LED.	7 (pull-down)
03	UART RX	Wejście szeregowego interfejsu UART, podłączone jednocześnie do UART-USB oraz do złącza RPi Conn J1-10	10
04	GPIO 04	Uniwersalny port wej/wyj	15
05	GPIO 05	Wymagany jest stan wysokii na tym wejściu w chwili restartu modułu ESP32, dlatego zastosowano rezystor pull-up 10 k. Uniwersalny port wej/wyj	13 (pull-up)
12	SPI MISO/ GPIO 12	Wymagany jest stan niski na tym wejściu w chwili restartu modułu ESP32, dlatego zastosowano rezystor pull-up 10 k. Wejście szeregowego interfejsu HSPI	21 (pull-down)
13	SPI MOSI/ GPIO 13	Wyjście szeregowego interfejsu HSPI	19
14	SPI CLK/ GPIO 14	Wyjście taktujące szeregowego interfejsu HSPI	23
15	SPI CS/ GPIO 15	Wyjście aktywujące Chip Select szeregowego interfejsu HSPI. Na linii zastosowano rezystor pull-down 10 k.	24 (pull-down)
16	ETH RESET/ GPIO 16	Port GPIO16 został połączony z sygnałem zerującym Ethernet PHY – układ LAN8742. Stan niski na tym wyprowadzeniu blokuje działanie interfejsu Ethernet. Przecięcie zworki JP1 odcina GPIO16 od Ethernet PHY.	26
17	GPIO 17	Uniwersalny port wej/wyj	11
18	ETH MDIO	Sterowanie interfejsem Ethernet	-

19	ETH TXD0	Sterowanie interfejsem Ethernet	-
21	ETH TXEN	Sterowanie interfejsem Ethernet	-
22	ETH TXD1	Sterowanie interfejsem Ethernet	-
23	ETH MDC	Sterowanie interfejsem Ethernet	-
25	ETH RXD0	Sterowanie interfejsem Ethernet	-
26	ETH RXD1	Sterowanie interfejsem Ethernet	-
27	ETH CR DV	Sterowanie interfejsem Ethernet	-
32	I2C SCL/ GPIO 32	Wyprowadzenie przygotowane do pracy jako linia SCL interfejsu I2C - zastosowano rezystor podciągający pull-up 2,2 k	5, 28
33	I2C SDA/ GPIO 33	Wyprowadzenie przygotowane do pracy jako linia SDA interfejsu I2C - zastosowano rezystor podciągający pull-up 2,2 k	3, 27
34	INPUT 34/ ADC1 CH6	Wyprowadzenie może pełnić rolę wejścia cyfrowego lub wejścia analogowego. Zastosowano rezystor podciągający pull-up 10 k	18 (pull-up)
35	INPUT 35/ ADC1 CH7	Wyprowadzenie może pełnić rolę wejścia cyfrowego lub wejścia analogowego. Zastosowano rezystor podciągający pull-up 10 k	22 (pull-up)
36	ADC1 CH0	Wejście analogowe z dodatkowym dzielnikiem napięcia 100 k/10 k	16
39	ADC1 CH3	Wejście analogowe z dodatkowym dzielnikiem napięcia 100 k/10 k	12

Złącze RPi Conn w standardzie RPi

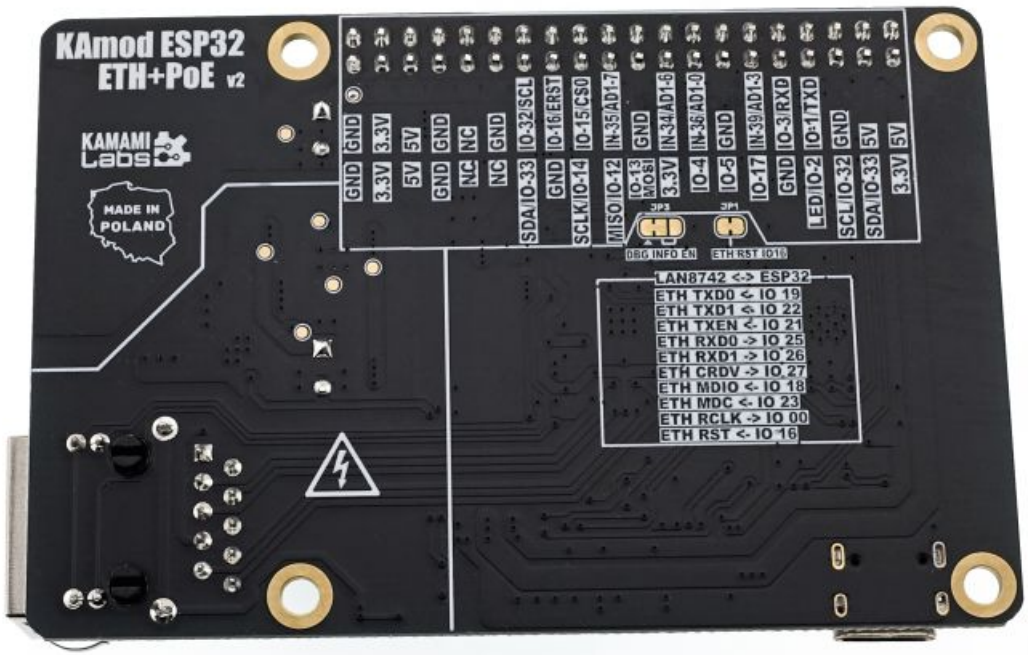
Złącze RPi Conn (J1) w standardzie Raspberry Pi zawiera 40 szpilek, do których doprowadzone są linie zasilania 5 V, 3,3 V, GND oraz niektóre wyprowadzenia GPIO modułu ESP32. Wyprowadzenia interfejsów UART (TXD, RXD), I2C (SDA, SCL) oraz SPI (MOSI, MISO, SCLK, CS0) zostały rozmieszczone tak, jak ma to miejsce w płytkach rodziny Raspberry Pi. Dokładny schemat wyprowadzeń oraz ich funkcje pokazuje rysunek:

Oznaczenie RPi	Uwagi	Funkcja	Pin
3,3 V		3,3 V	1
GPIO 02	I2C SDA	GPIO 33	3
GPIO 03	I2C SCL	GPIO 32	5
GPIO 04	LED/p-d	GPIO 02	7
GND		GND	9
GPIO 17		GPIO 17	11
GPIO 27	p-u	GPIO 05	13
GPIO 22		GPIO 04	15
3,3 V		3,3 V	17
GPIO 10	SPI MOSI	GPIO 13	19
GPIO 09	MISO/p-d	GPIO 12	21
GPIO 11	SPI CLK	GPIO 14	23
GND		GND	25
GPIO 00	I2C SDA	GPIO 33	27
GPIO 05			29
GPIO 06			31
GPIO 13		GND	33
GPIO 19		5 V	35
GPIO 26		3,3 V	37
GND		GND	39



Pin	Funkcja	Uwagi	Oznaczenie RPi
2	5 V		5 V
4	5 V		5 V
6	GND		GND
8	GPIO 01	TXD	GPIO 14
10	GPIO 03	RXD	GPIO 15
12	GPIO 39	ADC1-3	GPIO 18
14	GND		GND
16	GPIO 36	ADC1-0	GPIO 23
18	GPIO 34	IN/ADC1-6	GPIO 24
20	GND		GND
22	GPIO 35	IN/ADC1-7	GPIO 25
24	GPIO 15	SPI CS/p-d	GPIO 08
26	GPIO 16	ETH RST	GPIO 07
28	GPIO32	I2C SCL	GPIO 01
30	GND		GND
32			GPIO 12
34	GND		GND
36	5 V		GPIO 16
38	3,3 V		GPIO 20
40	GND		GPIO 21

Opis wyprowadzeń został również naniesiony na spodzie płytki KAmod ESP32 ETH+POE:

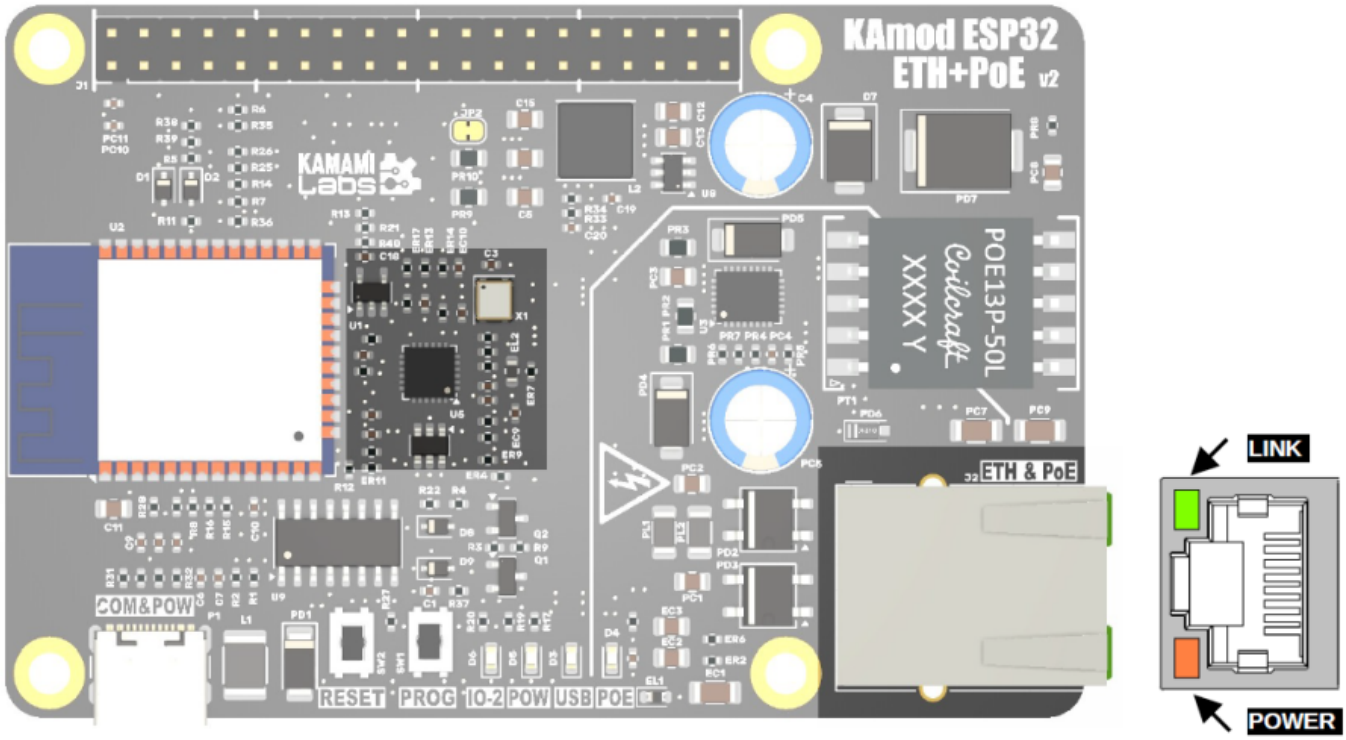


Interfejs Ethernet

Na płytce KAmod ESP32 ETH+POE został zaimplementowany przewodowy interfejs Ethernet z klasycznym gniazdem RJ45 (J2). W roli drivera (PHY) interfejsu Ethernet zastosowano układ **LAN8742**, który jest kompatybilny z **LAN8720** i jest wspierany w środowisku Arduino. Może działać z prędkościami 100 Mb lub 10 Mb w trybie Full-Duplex lub Half-Duplex.

Driver Ethernet jest połączony z modułem ESP32 poprzez interfejs RMII (Reduced media-independent interface). Przyporządkowanie sygnałów opisuje tabela:

Sygnał RMII	Kierunek	Wyprowadzenie modułu ESP32
TXD0	<-	GPIO19
TXD1	<-	GPIO22
TXEN	<-	GPIO21
RXD0	->	GPIO25
RXD1	->	GPIO26
CRS_DV	->	GPIO27
MDIO	<->	GPIO18
MDC	<-	GPIO23
REF_CLK	->	GPIO0
RESET	<-	GPIO16



Na złączu J2 znajdują się dwie diody sygnalizacyjne. Dioda po lewej stronie (POWER) sygnalizuje obecność głównego napięcia zasilającego - napięcia 3,3 V. Dioda po prawej stronie (LINK) miganiem sygnalizuje aktywność interfejsu Ethernet.

Zasilanie metodą PoE

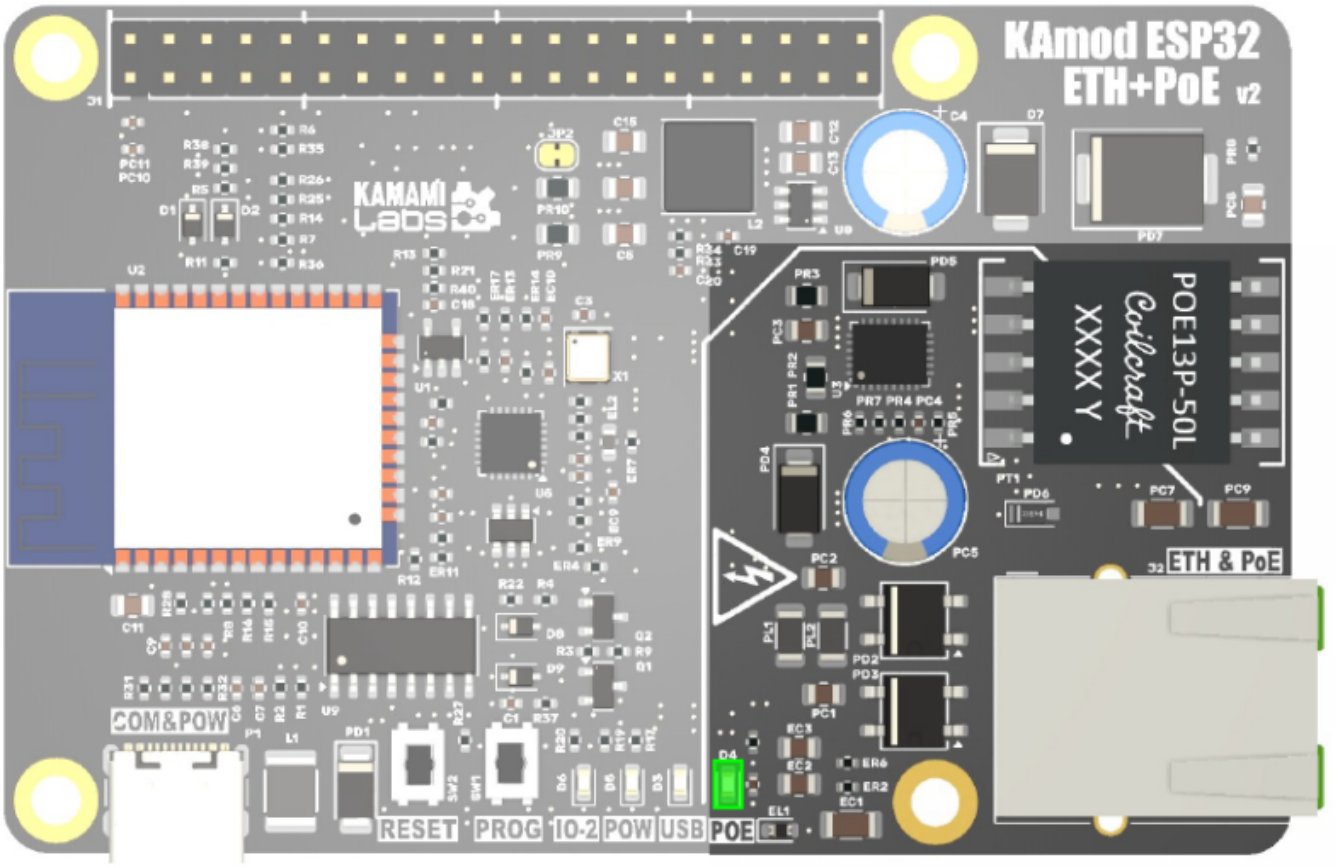
Poprzez złącze **ETH & PoE** (J2) może być dostarczane zasilanie do płytki ewaluacyjnej. Zastosowany kontroler zasilania PoE bazuje na układzie MP8007, który jest kompatybilny ze standardami **IEEE 802.3af** - Powered Devices Type-1 oraz **IEEE 802.3at** - Powered Devices Type-2. Blok zasilania PoE jest skonfigurowany do pracy w klasie 0 (Class 0), która definiuje pobór mocy urządzenia w zakresie 0,5...13 W.

Zasilanie metodą PoE jest możliwe tylko w kompatybilnej instalacji, zawierającej urządzenie PSE (Power Sourcing Equipments) spełniające standard IEEE 802.3af/at np. router PoE. Prawidłowe działanie bloku zasilania PoE jest sygnalizowane świeceniem diody POE (D4).

W czasie pracy bloku zasilania PoE, może być słyszalny szum lub cichy pisk – jest to naturalne zjawisko wywołane działaniem przetwornicy impulsowej (SMPS).

Przy właściwym zasilaniu płytki ewaluacyjnej KAmod ESP32 ETH+POE wytwarzane są napięcia stabilizowane 5 V oraz 3,3 V, dostępne na złączy szpilkowym J1. Mogą posłużyć do zasilania innych modułów dołączonych do płytki ewaluacyjnej, należy jednak pamiętać, że sumaryczny prąd nie powinien przekraczać 1 A. Świecenie diody D4 oznacza prawidłowe działanie modułu zasilania PoE.

W instalacji PoE występują napięcia o wartościach sięgających aż 60 V. Wszelkie czynności wykonywane w takich instalacjach z użyciem płytki ewaluacyjnej KAmod ESP32 ETH+POE należy wykonywać ze szczególną ostrożnością i z zachowaniem zasad bezpieczeństwa.



Interfejs USB

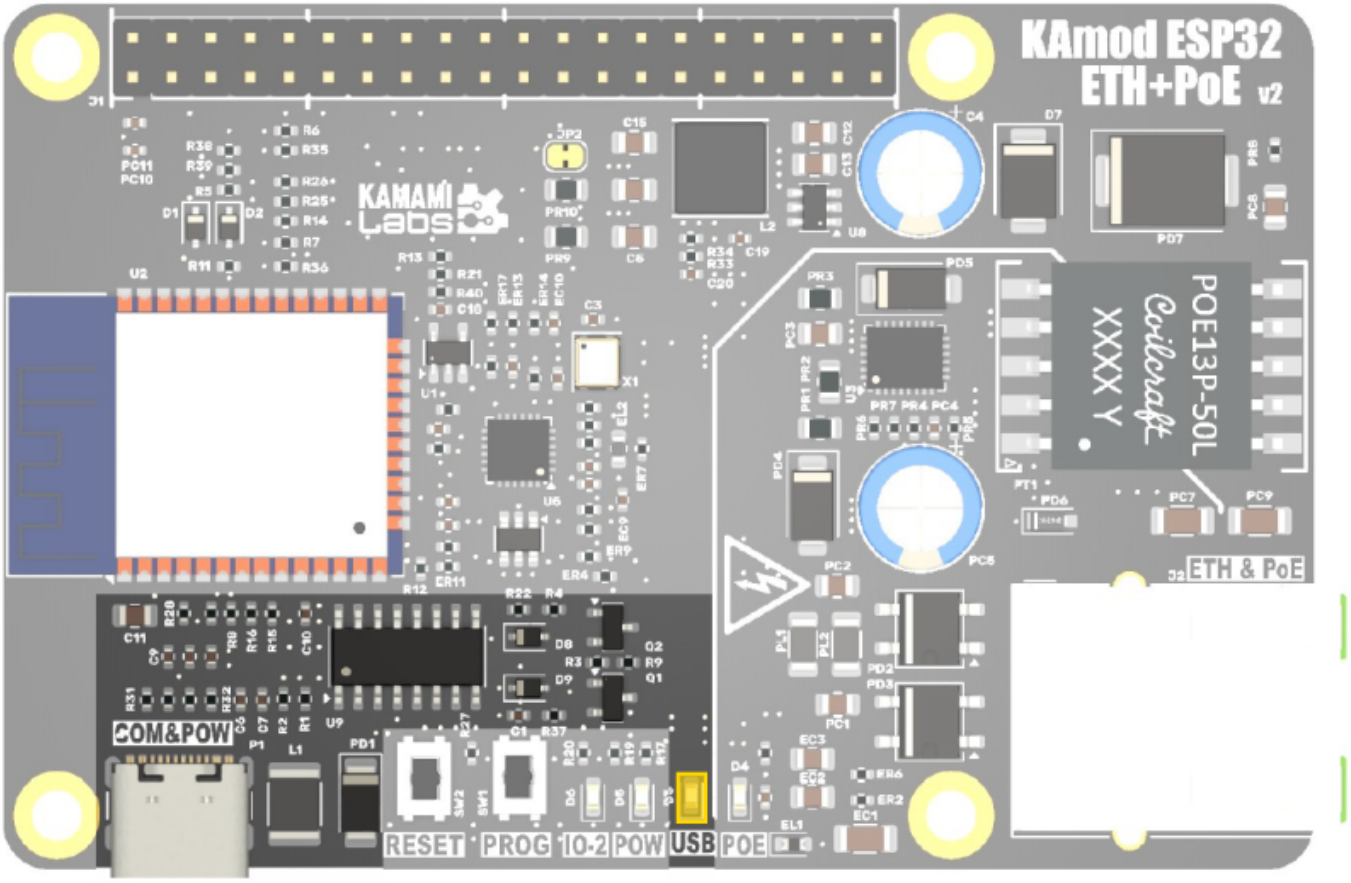
Złącze P1 typu USB-C jest połączone z kontrolerem typu CH340, który realizuje funkcje konwertera USB-UART. Interfejs UART może być używany w docelowej aplikacji, ale służy także do programowania modułu ESP32. Proces programowania może przebiegać całkowicie automatycznie, ponieważ kontroler CH340 steruje kluczowymi wyprowadzeniami modułu ESP32 (**GPIO0** – *Boot Select* oraz **EN** - *Chip Power-up*).

Połączenia sygnałów pomiędzy CH340 i ESP32 są następujące:

Sygnał kontrolera CH340	Wyprowadzenie modułu ESP32
TXD (wyjście danych)	GPIO03 (UART0 RXD)
RXD (wejście danych)	GPIO01 (UART0 TXD)
DTR (wyście kontroli transmisji)	EN (Chip Power-up)
RTS (wyjście kontroli transmisji)	GPIO0 (Boot Select)

Do linii TXD jest dołączona dioda led oznaczona USB (D3), która sygnalizuje odbieranie danych z interfejsu USB. W przypadku użycia w docelowej aplikacji konwertera USB-UART należy zadbać o to, aby linie DTR oraz RTS pozostały nieobsługiwane (*Handshaking: None*).

Złącze USB-C może służyć jako alternatywne wejście zasilania dla płytki KAmod ESP32 ETH+POE, jednak wtedy parametry obwodów zasilania nie będą spełnione. Napięcie na linii 5 V, będzie niższe i będzie wynosiło ok. 4,5 V; napięcie na linii 3,3 V nie powinno się zmienić; wydajność prądowa napięć 5 V oraz 3,3 V może być dużo niższa i będzie zależała od zastosowanego zasilania na złączu USB-C.



Przyciski oraz kontrolki sygnalizacyjne

Na płytce KAmod ESP32 ETH+POE znajdują się 4 diody LED, które sygnalizują działanie różnych komponentów – zgodnie z poniższą tabelą. Dioda D6 (LED IO-2) jest dołączona do wyprowadzenia GPIO2 modułu ESP32. Jej zaświecenie wymaga programowego ustawienia stanu wysokiego na wyprowadzeniu GPIO2.

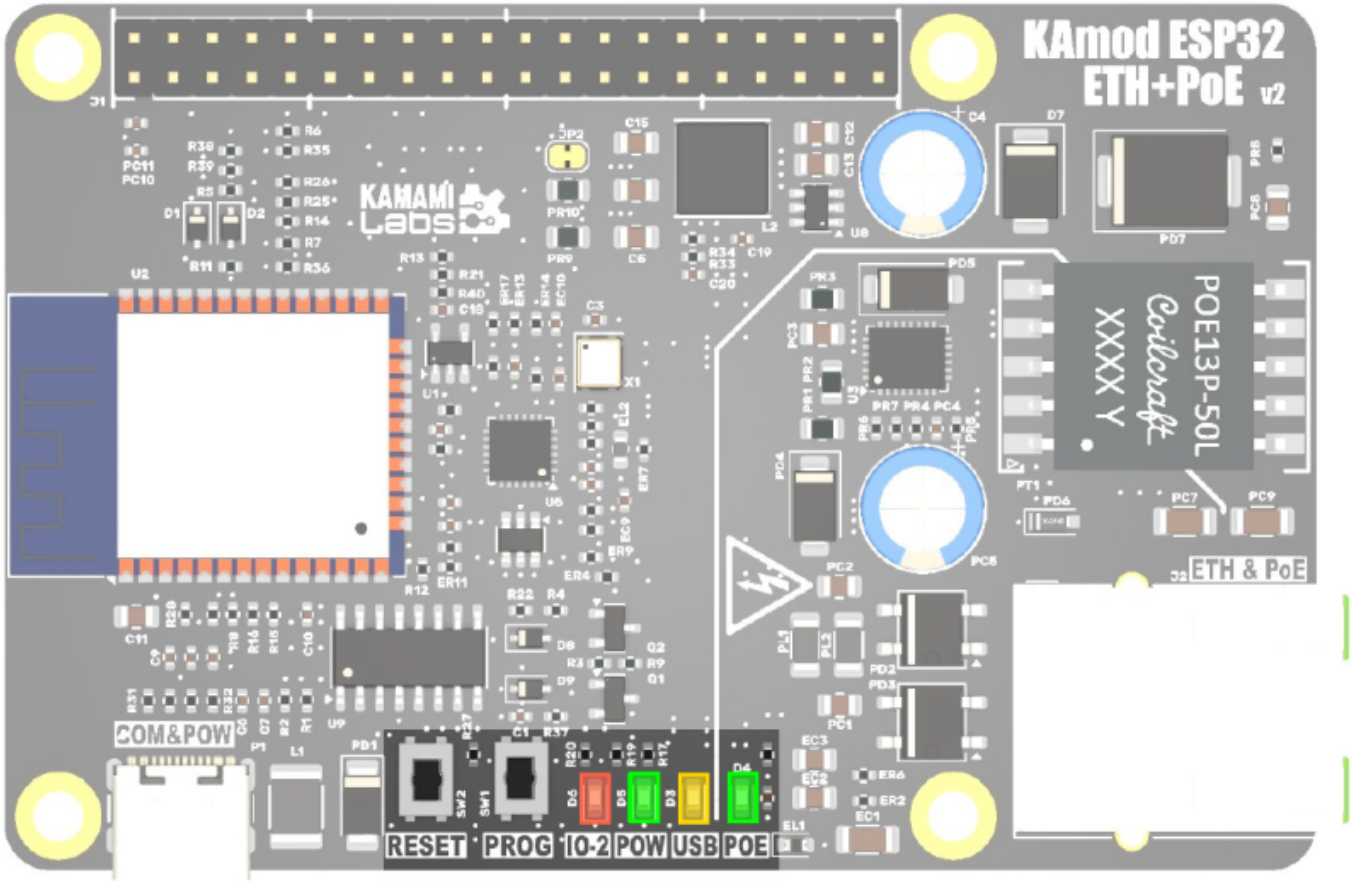
Komponent	Funkcja
D3 – **USB**	Miganie diody D3 oznacza przesyłanie danych z USB do modułu ESP32
D4 – **POE**	Świecenie diody D4 oznacza prawidłowe działanie modułu zasilania PoE
D5 – **POW**	Świecenie diody D5 oznacza obecność głównego napięcia zasilającego - napięcia 3,3 V
D6 – **IO-2**	Dioda D6 jest dołączona do wyprowadzenia GPIO2 modułu ESP32 i jej świecenie może być sterowane programowo

Przycisk RESET umożliwia wykonanie restartu modułu ESP32 oraz, jednocześnie kontrolera interfejsu Ethernet. Jest połączony z linią EN (*Chip Power-up*) modułu ESP32.

Przycisk PROG pozwala wprowadzić moduł ESP32 w tryb programowania. Należy wtedy nacisnąć przycisk RESET, następnie, trzymając wciśnięty RESET, przytrzymać przycisk PROG i wtedy zwolnić RESET, jednocześnie trzymając jeszcze przez chwilę wciśnięty PROG. Funkcjonalność ta może być przydatna, gdy z jakiegoś powodu tryb programowania nie będzie uruchamiany automatycznie poprzez konwerter USB-UART.

Komponent	Funkcja
Przycisk SW1 – **PROG**	• Uruchamia tryb programowania poprzez UART (tylko w momencie restartu modułu ESP32)

Przycisk SW2 – **RESET**	• Powoduje restart modułu ESP32 oraz kontrolera interfejsu Ethernet



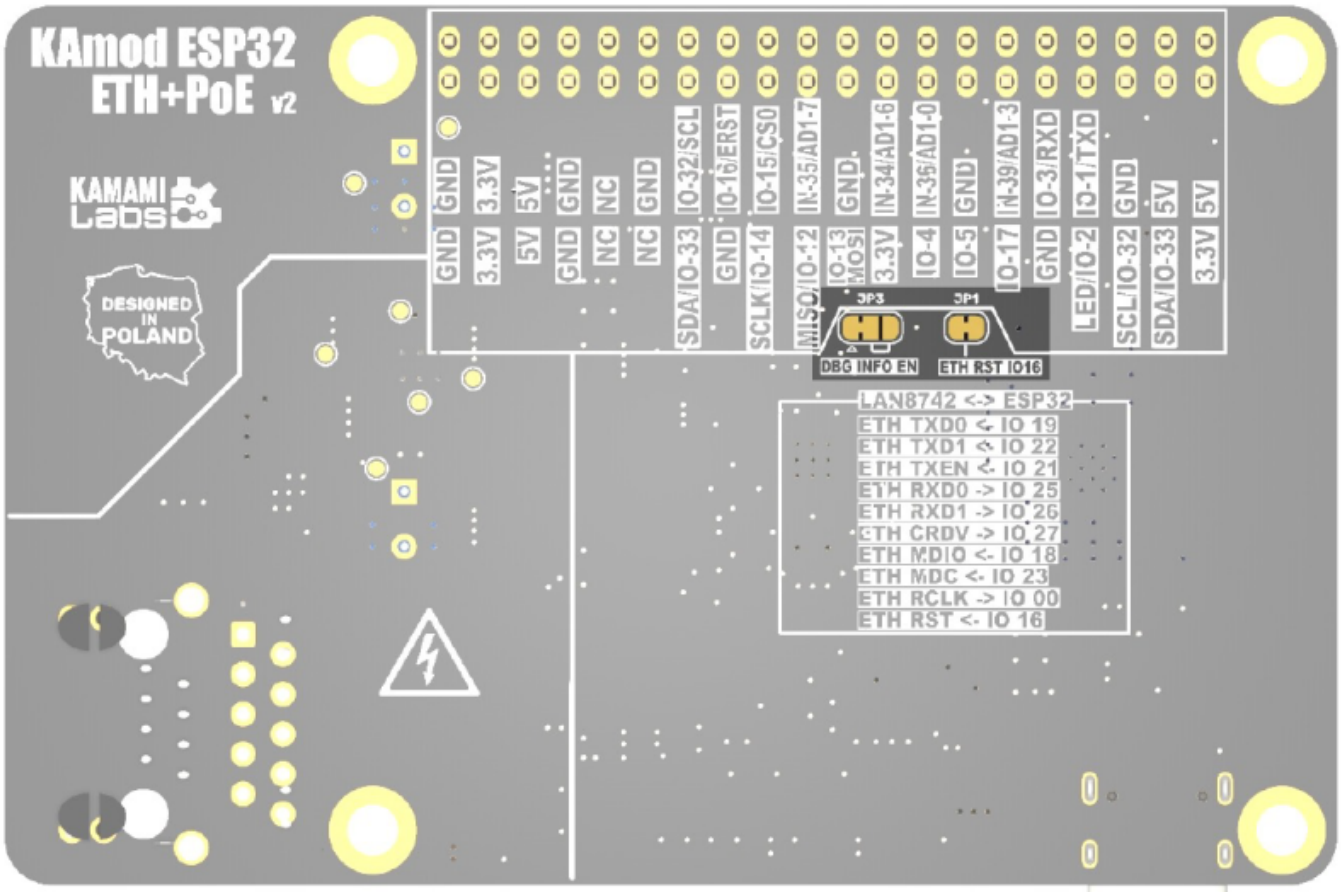
Funkcje zaawansowane

Zworki JP1 oraz JP3 znajdują się na dolnej stronie płytki ewaluacyjnej (bottom).

Komponent	Funkcja
JP1 – ETH RST IO16	• Zworka SMD, fabrycznie zwarta, stanowi połączenie portu GPIO16 z sygnałem zerowania drivera Ethernet
JP3 – DBG INFO EN	• Zworka SMD, pozwala włączyć wysyłanie komunikatów systemowych – Debugging Log, poprzez interfejs UART (USB)

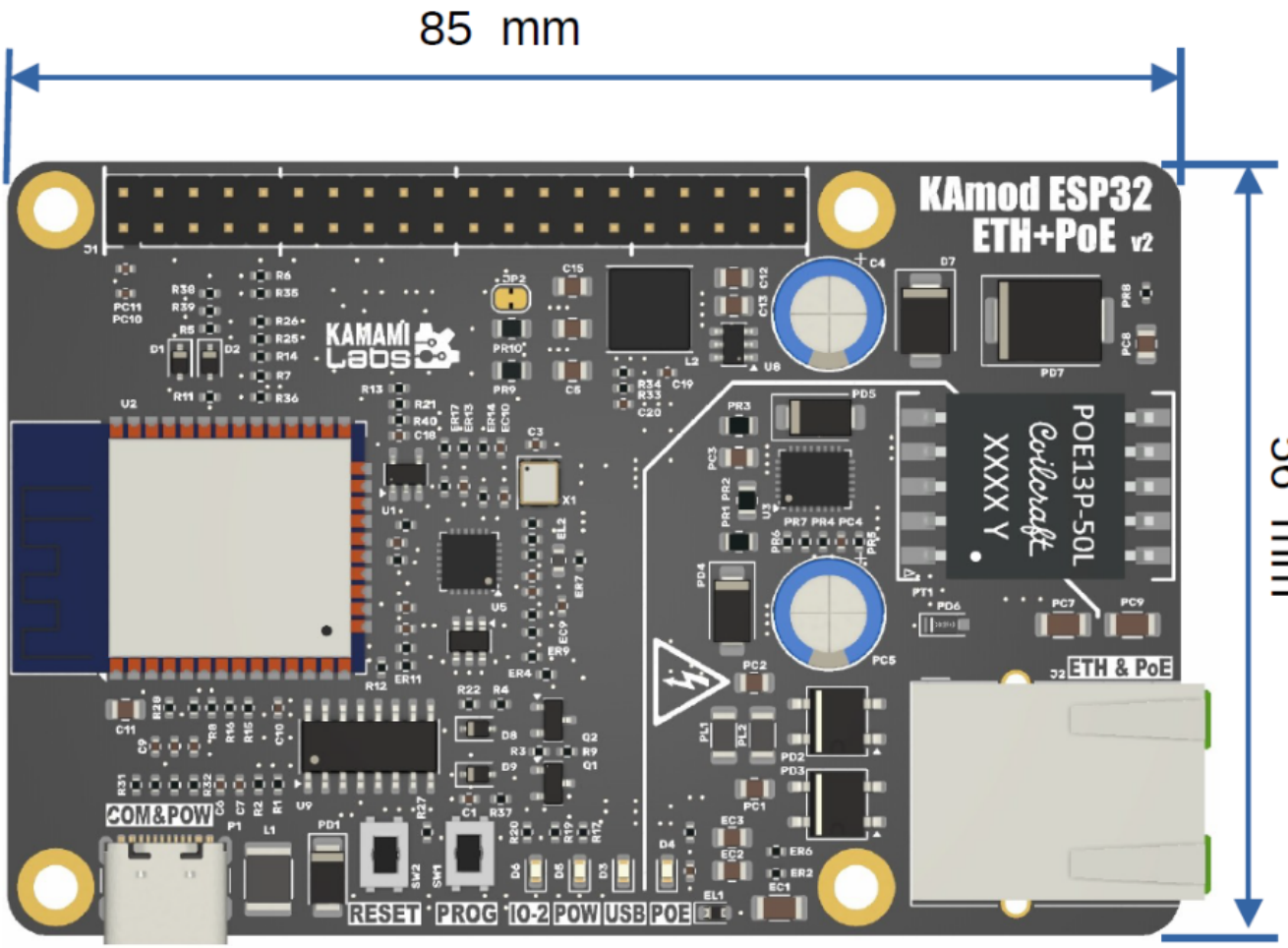
JP1 – ETH RST IO16 jest fabrycznie zwarta (ścieżka miedzi pomiędzy padami) i zapewnia połączenie pomiędzy portem GPIO16 modułu ESP32 i wejściem RESET drivera interfejsu Ethernet. Aby odłączyć port GPIO16 od sygnału zerowania drivera Ethernet należy przeciąć ostrym narzędziem powierzchnię płytki – tak jak wskazuje czerwona linia przy JP1 na poniższym rysunku. Ponowne połączenie sygnału zerowania jest możliwe poprzez naniesienie kropelki spoiwa lutowniczego, które połączy oba pady zworki JP1.

JP3 – DBG INFO EN jest fabrycznie połączona pomiędzy padem środkowym a padem numer 1 (masa zasilania) i powoduje wyciszenie komunikatów systemowych tzw. Debugging Log. Aby włączyć wysyłanie komunikatów systemowych należy przeciąć ostrym narzędziem powierzchnię płytki tak, jak wskazuje czerwona linia przy JP3 na poniższym rysunku oraz nanieść kroplę spoiwa lutowniczego, które połączy pady po przeciwnej stronie, pady 2-3 (do napięcia 3,3 V). Nie wolno połączyć padów 2-3, bez wcześniejszego rozdzielenia padów 1-2.



Wymiary

Wymiary płytki KAmod ESP32 ETH+POE to 85x56 mm. Wysokość maksymalna wynosi ok. 20 mm. Na płytce znajdują się 4 otwory montażowe o średnicy 3 mm rozmieszczone podobnie jak na płytkach z rodziny Raspberry Pi.


85 mm
56 mm
KAmod ESP32
ETH+PoE v2
KAMAMI Labs
POE13P-50L
Coilcraft
XXXX Y
ETH & PoE
COM&POW
RESET
PROG
IO-2
POW
USB
POE

Program testowy

Kod programu testowego znajduje się poniżej, można go skompilować w środowisku Arduino.

Zawsze aktualna, najnowsza wersja oprogramowania znajduje się w repozytorium na portalu GitHub

```
//Ino Board: esp32 -> ESP32-WROOM_DA-Module
#include <ETH.h>
#include <WiFi.h>
/*
   * ETH_CLOCK_GPIO0_IN   - default: external clock from crystal oscillator
   * ETH_CLOCK_GPIO0_OUT  - 50MHz clock from internal APLL on GPIO0
   * ETH_CLOCK_GPIO16_OUT - 50MHz clock from internal APLL on GPIO16
   * ETH_CLOCK_GPIO17_OUT - 50MHz clock from internal APLL inverted on GPIO17
*/
#ifdef ETH_CLK_MODE
  #undef ETH_CLK_MODE
#endif
#define ETH_CLK_MODE    ETH_CLOCK_GPIO0_IN
// Pin# of the enable signal for the external crystal oscillator (-1 to disable for internal
APLL source)
#define ETH_POWER_PIN   -1
// Type of the Ethernet PHY (LAN8720 or TLK110)
#define ETH_TYPE        ETH_PHY_LAN8720
// I²C-address of Ethernet PHY (0 or 1 for LAN8720, 31 for TLK110)
#define ETH_ADDR        0
// Pin# of the I²C clock signal for the Ethernet PHY
#define ETH_MDC_PIN     23
// Pin# of the I²C IO signal for the Ethernet PHY
#define ETH_MDIO_PIN    18
// Mode select PINs
#define ETH_RXD0_MODE0   25
#define ETH_RXD1_MODE1   26
#define ETH_CRS_MODE2    27

#define ETH_RESET       16

#define LED_PIN         2

static bool eth_connected = false;

WiFiServer server(80);

// Select the IP address according to your local network
IPAddress myIP(10, 1, 0, 182);
IPAddress myGW(10, 1, 0, 252);
IPAddress mySN(255, 255, 0, 0);
IPAddress myDNS(8, 8, 8, 8);

void myEvent(WiFiEvent_t event) {
  switch (event) {
    case ARDUINO_EVENT_ETH_START:
      Serial.println("ETH Started");
      ETH.setHostname("esp32-ethernet");
      break;
```

```
    case ARDUINO_EVENT_ETH_CONNECTED:
      Serial.println("ETH Connected");
      break;
    case ARDUINO_EVENT_ETH_GOT_IP:
      //Serial.println("ETH Got IP");
      //Serial.println(ETH);
      Serial.print("ETH MAC: ");
      Serial.print(ETH.macAddress());
      Serial.print(", IPv4: ");
      Serial.print(ETH.localIP());
      if (ETH.fullDuplex()) {
        Serial.print(", FULL_DUPLEX");
      }
      Serial.print(", ");
      Serial.print(ETH.linkSpeed());
      Serial.println("Mbps");
      eth_connected = true;
      break;
    case ARDUINO_EVENT_ETH_LOST_IP:
      Serial.println("ETH Lost IP");
      eth_connected = false;
      break;
    case ARDUINO_EVENT_ETH_DISCONNECTED:
      Serial.println("ETH Disconnected");
      eth_connected = false;
      break;
    case ARDUINO_EVENT_ETH_STOP:
      Serial.println("ETH Stopped");
      eth_connected = false;
      break;
    default:
      break;
  }
}

void setup() {
  //ETH Reset assert
  pinMode(ETH_RESET, OUTPUT);
  digitalWrite(ETH_RESET, LOW);
  pinMode(ETH_RXD0_MODE0, INPUT_PULLUP);
  pinMode(ETH_RXD1_MODE1, INPUT_PULLUP);
  pinMode(ETH_CRS_MODE2, INPUT_PULLUP);

  //Serial - start
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  for(int i=0; i<7; i++){
    digitalWrite(LED_PIN, HIGH);
    delay(200);
    digitalWrite(LED_PIN, LOW);
    delay(200);
  }
  //ETH Reset deassert
  digitalWrite(ETH_RESET, HIGH);
  delay(50);

  WiFi.onEvent(myEvent);
  ETH.setAutoNegotiation(false);
  ETH.setFullDuplex(true);
```

```
  ETH.setLinkSpeed(true);
  ETH.begin(ETH_TYPE, ETH_ADDR,
            ETH_MDC_PIN, ETH_MDIO_PIN,
            ETH_POWER_PIN, ETH_CLK_MODE);
  //ETH.config(myIP, myGW, mySN, myDNS);

  while (!ETH.connected()){}
  server.begin();
}

void loop() {
  // listen for incoming clients
  WiFiClient client = server.available();

  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("<H2>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=\"/SEND\">here</a> to write to Serial
(USB).<br></H2>");
            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 /SEND")) {
          Serial.println("\r\nHELLO - you send message via Serial(USB)");
        }
      }
    }

    delay(10);
    client.stop();
    Serial.println("********Client Disconnected********");
```

```
  }
}
```

Program testowy konfiguruje port GPIO2 jako wyjście sterujące diodą LED (D6) i sygnalizuje rozpoczęcie działania kilkoma mignięciami. Następnie uruchamia sprzętowy interfejs UART i konfiguruje go do pracy jako interfejs szeregowy połączony do konwertera UART-USB. Dzięki temu można monitorować działanie płytki w dowolnym programie typu terminal (Serial Monitor).

```
#define LED_PIN 2
pinMode(LED_PIN, OUTPUT);
for(int i=0; i<5; i++){
digitalWrite(LED_PIN, HIGH);
delay(200);
digitalWrite(LED_PIN, LOW);
delay(200);
}
//Serial - start
Serial.begin(115200);
```

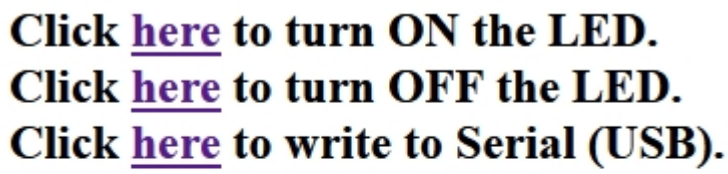
Przygotowanie do pracy drivera interfejsu Ethernet – układu LAN8742, wymaga dołączenia biblioteki ETH.h oraz zdefiniowania funkcji wyprowadzeń. Układ LAN8742 jest kompatybilny z układem LAN8720, który z kolei jest wspierany w środowisku Arduino.

```
#include <ETH.h>
/*
* ETH_CLOCK_GPIO0_IN - default: external clock from crystal oscillator
* ETH_CLOCK_GPIO0_OUT - 50MHz clock from internal APLL on GPIO0
* ETH_CLOCK_GPIO16_OUT - 50MHz clock from internal APLL on GPIO16
* ETH_CLOCK_GPIO17_OUT - 50MHz clock from internal APLL inverted on GPIO17
*/
#ifdef ETH_CLK_MODE
#undef ETH_CLK_MODE
#endif
#define ETH_CLK_MODE ETH_CLOCK_GPIO0_IN
// Pin# of the enable signal for the external crystal oscillator (-1 to disable for internal
APLL source)
#define ETH_POWER_PIN -1
// Type of the Ethernet PHY (LAN8720 or TLK110)
#define ETH_TYPE ETH_PHY_LAN8720
// I²C-address of Ethernet PHY (0 or 1 for LAN8720, 31 for TLK110)
#define ETH_ADDR 0
// Pin# of the I²C clock signal for the Ethernet PHY
#define ETH_MDC_PIN 23
// Pin# of the I²C IO signal for the Ethernet PHY
#define ETH_MDIO_PIN 18
#define ETH_RESET 16
//ETH Reset assert
pinMode(ETH_RESET, OUTPUT);
digitalWrite(ETH_RESET, LOW);
...
//ETH Reset deassert
digitalWrite(ETH_RESET, HIGH);
delay(200);
```

Teraz można uruchomić interfejs oraz serwer www:

```
WiFiServer server(80);
ETH.begin(ETH_TYPE, ETH_ADDR,
ETH_MDC_PIN, ETH_MDIO_PIN,
ETH_POWER_PIN, ETH_CLK_MODE);
...
while (!ETH.connected()){}
server.begin();
```

Po uruchomieniu programu testowego zostanie uruchomiony serwer www z bardzo prostą stroną internetową, która umożliwia sterowanie diodą LED D6 oraz wysłanie komunikatu poprzez port szeregowy:



Adres IP, który zostanie przydzielony serwerowi www w sieci LAN można odczytać z komunikatów wysyłanych przez port szeregowy:

```
ETH Started
ETH Connected
ETH MAC: 40:22:D8:6D:D8:D3, IPv4: 10.1.0.132, FULL_DUPLEX, 100Mbps
********New Client********
********Client Disconnected********
********New Client********

HELLO - you send message via Serial(USB)
********Client Disconnected********
```

Adres IP można również określić w programie, należy wtedy określić cztery parametry:

```
IPAddress myIP(10, 1, 0, 182);
IPAddress myGW(10, 1, 0, 252);
IPAddress mySN(255, 255, 0, 0);
IPAddress myDNS(8, 8, 8, 8);
```

a następnie wpisać polecenie:

```
ETH.config(myIP, myGW, mySN, myDNS);
```

przed linią:

```
server.begin();
```

Linki

- Karta katalogowa układu LAN8742
- Karta katalogowa układu MP8007
- Karta katalogowa układu ESP32
- Karta katalogowa układu CH340
- Program testowy Arduino

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