



KAmod ESP32 ETH POE (PL)



Rev. 20260127092338

Źródło: [https://wiki.kamamilabs.com/index.php?title=KAmod_ESP32_ETH_POE_\(PL\)](https://wiki.kamamilabs.com/index.php?title=KAmod_ESP32_ETH_POE_(PL))

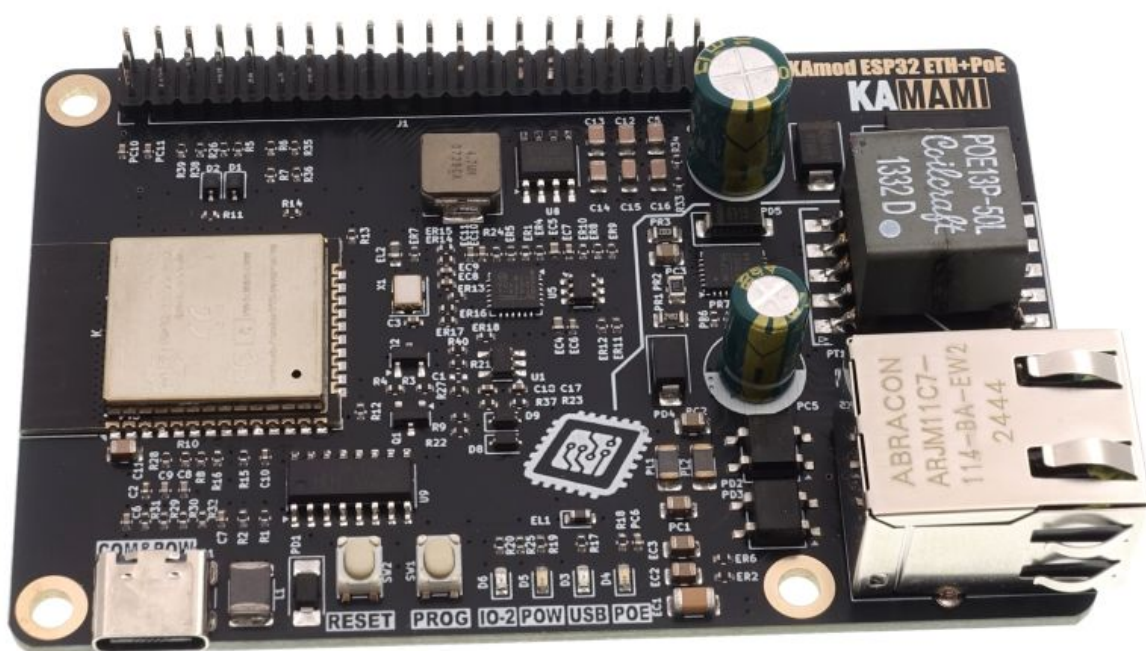
Spis treści

Opis	1
Podstawowe cechy i parametry	2
W wyposażenie standardowe	3
Schemat elektryczny	4
Interfejs Ethernet	10
Zasilanie metodą PoE	11
Interfejs USB	12
Przyciski resetowania i programowania	13
Kontrolki sygnalizacyjne	14
Złącze GPIO w standardzie RPi	15
Funkcje zaawansowane	18
Wymiary	19
Program testowy	20
Linki	26

Opis

KAmoD ESP32 ETH POE - Płytki 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

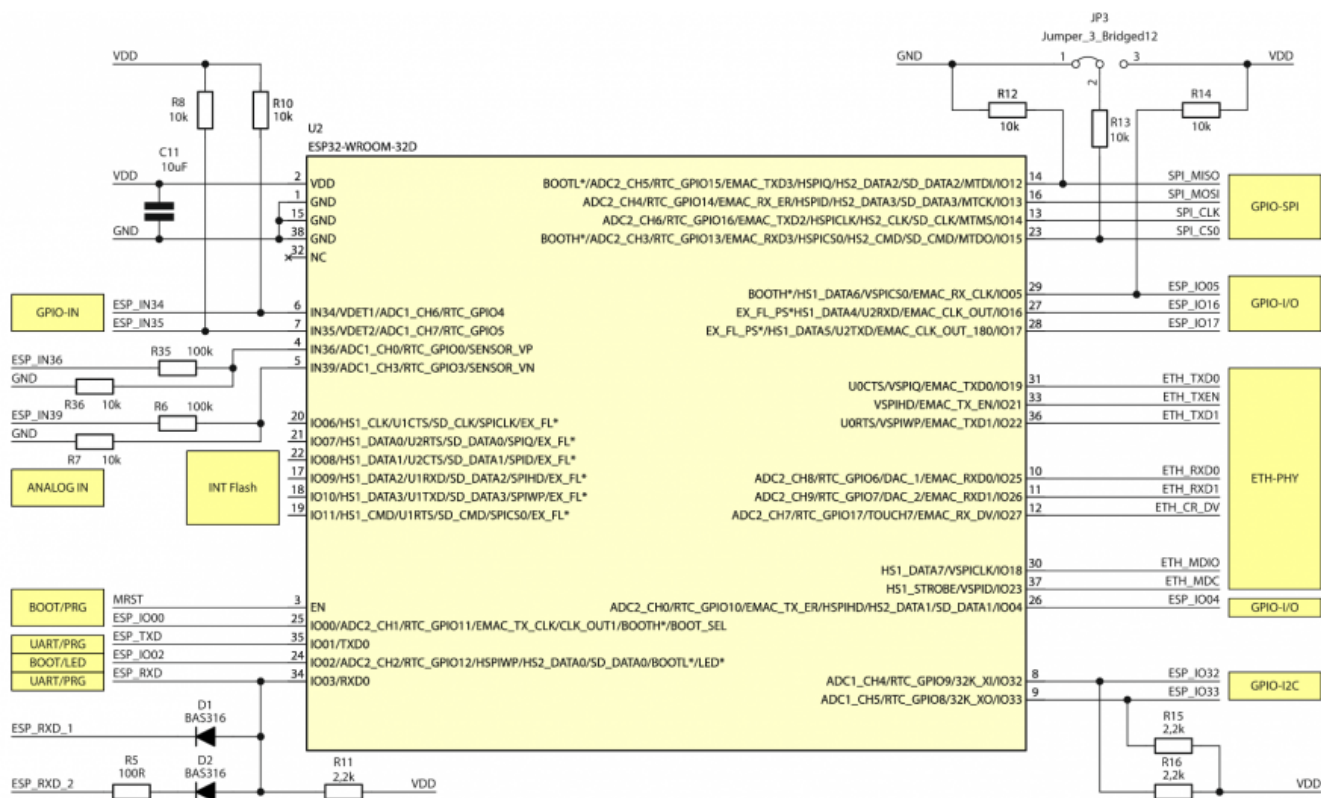
- Moduł ESP32-WROOM umożliwiający komunikację w sieci Wi-Fi w paśmie 2,4 GHz
- Interfejs Ethernet na bazie układu LAN8742 (100/10 M; full/half duplex)
- Zintegrowany konwerter UART-USB ze złączem USB-C umożliwiający programowanie układu ESP32
- Układ zasilania PoE, kompatybilny ze standardem IEEE 802.3af/at Class 0
- Dostarcza stabilizowanych napięć 5 V (+/-10%) oraz 3,3 V (+/-5%) o łącznym prądzie do 1,5 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

Wypożyczenie standardowe

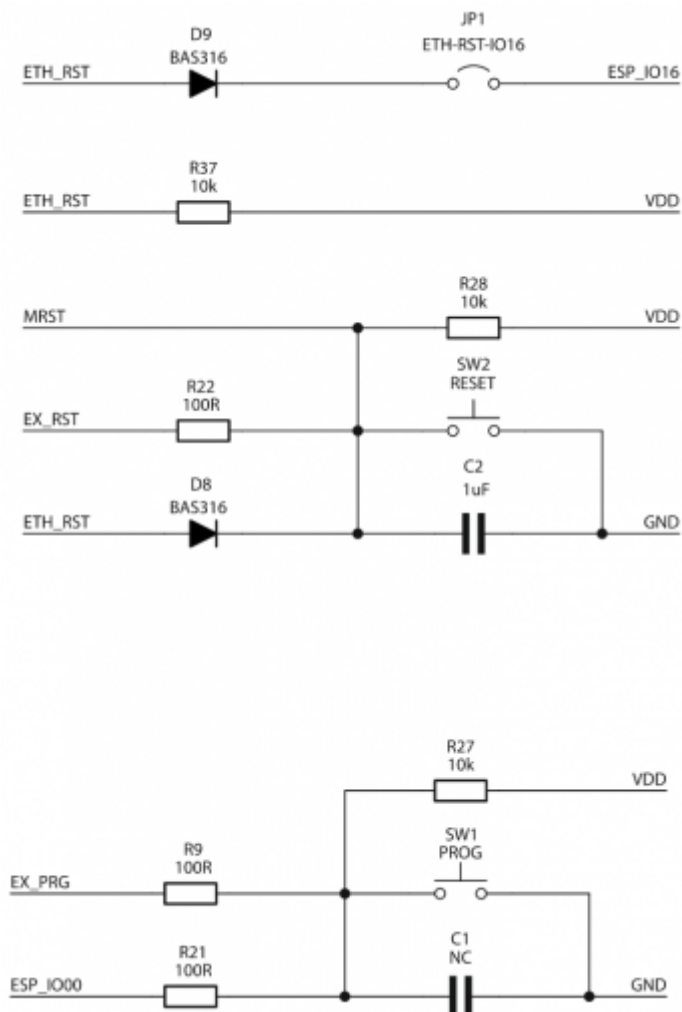
Kod	Opis
KAmoESP32 ETH+POE	Zmontowany i uruchomiony moduł

Schemat elektryczny

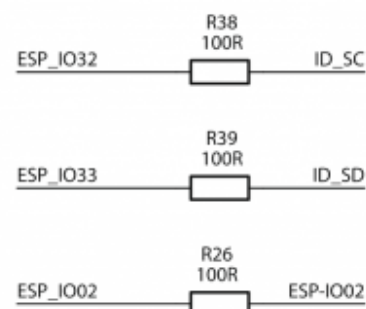
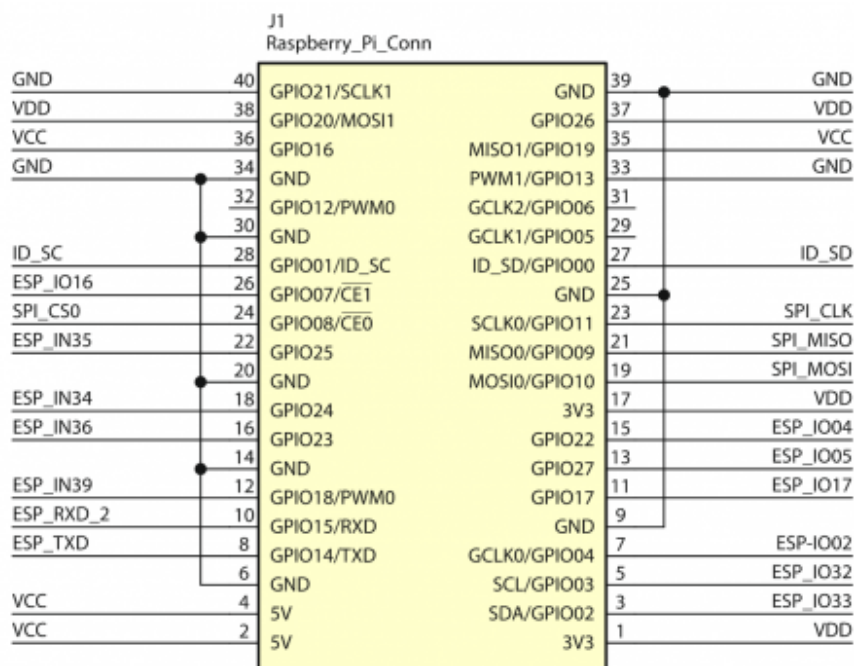
Moduł ESP32



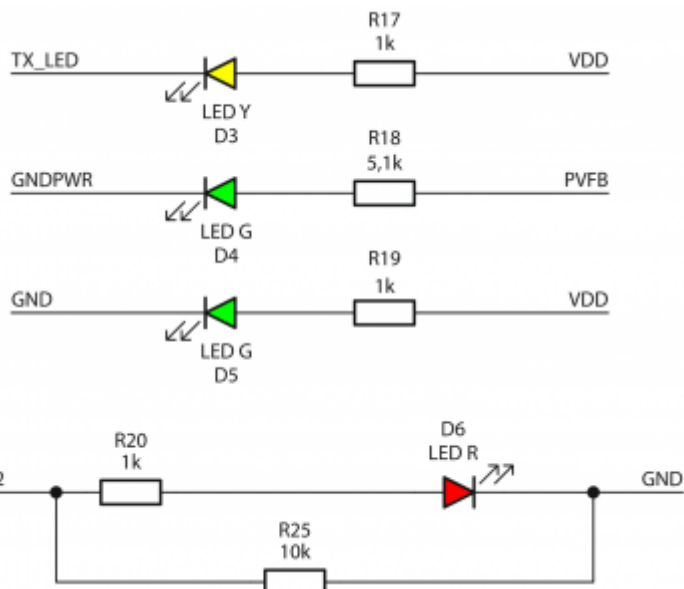
Elementy odpowiedzialne za funkcje resetu i programowania



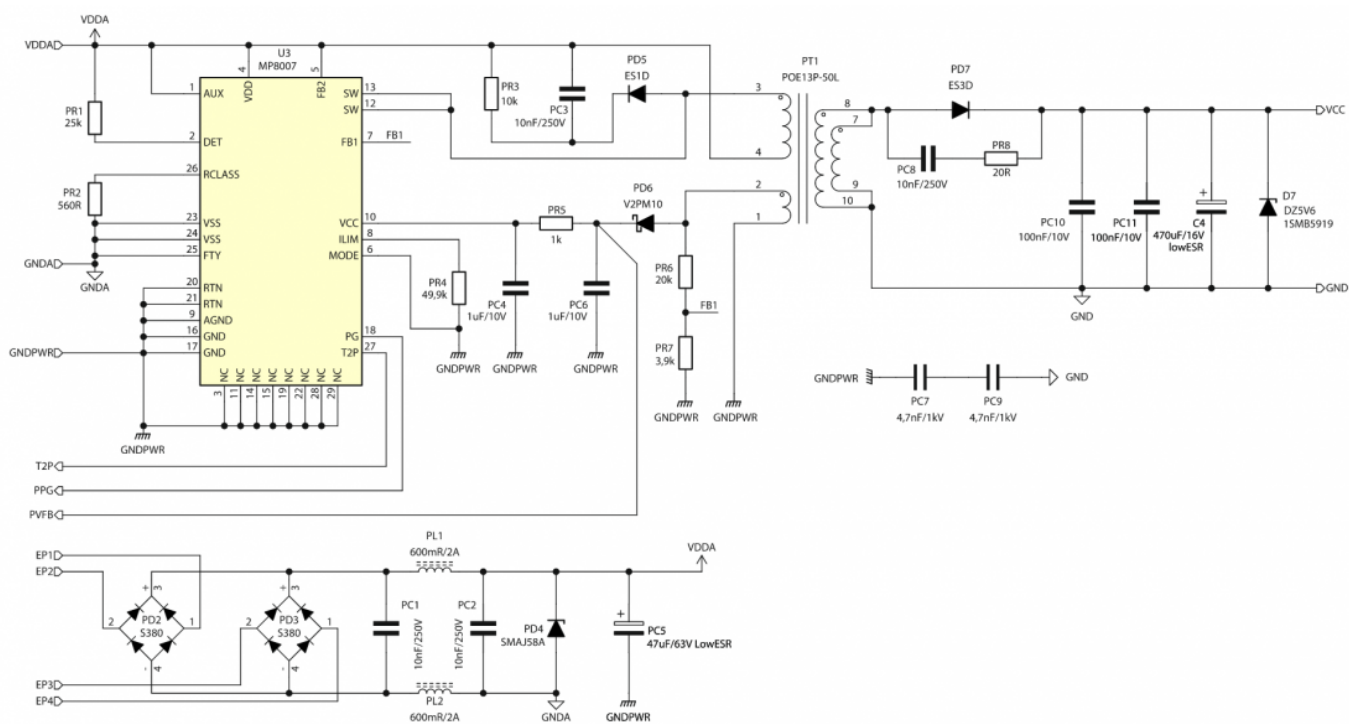
Złącze GPIO



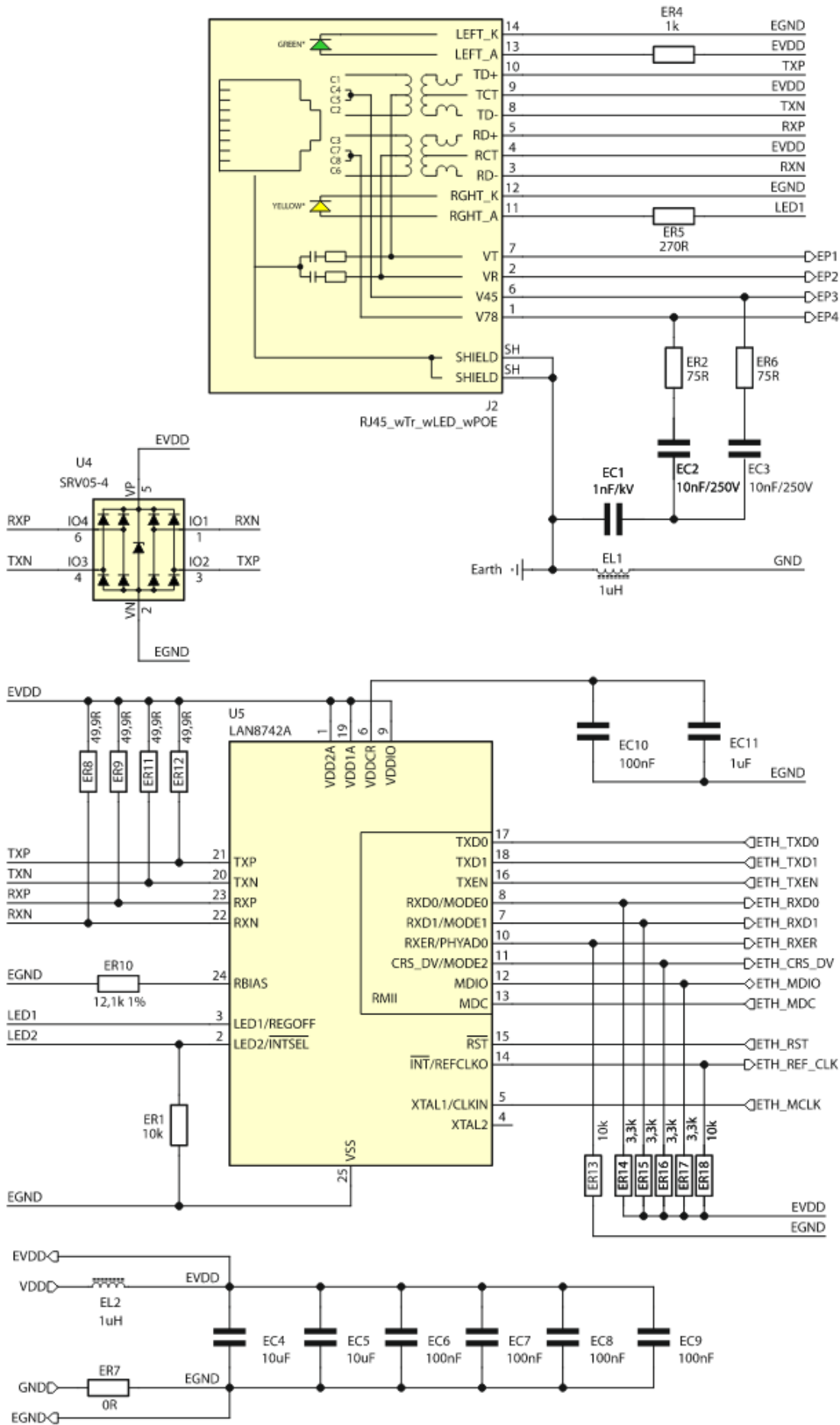
Diody sygnalizacyjne

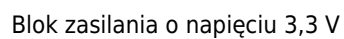
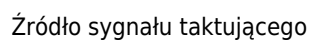


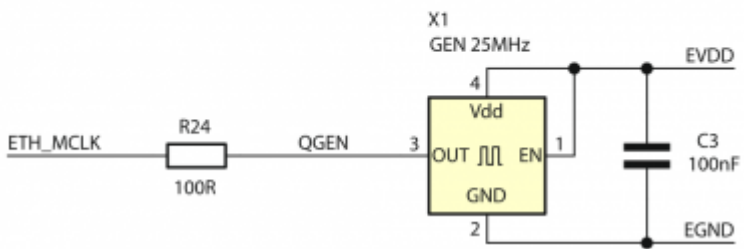
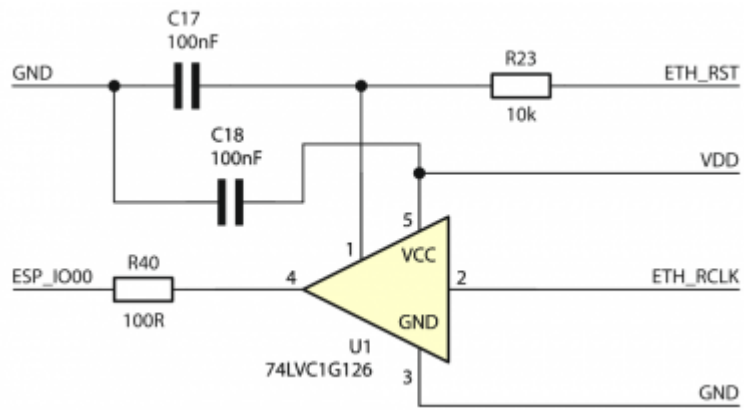
Interfejs Ethernet



Zasilanie PoE







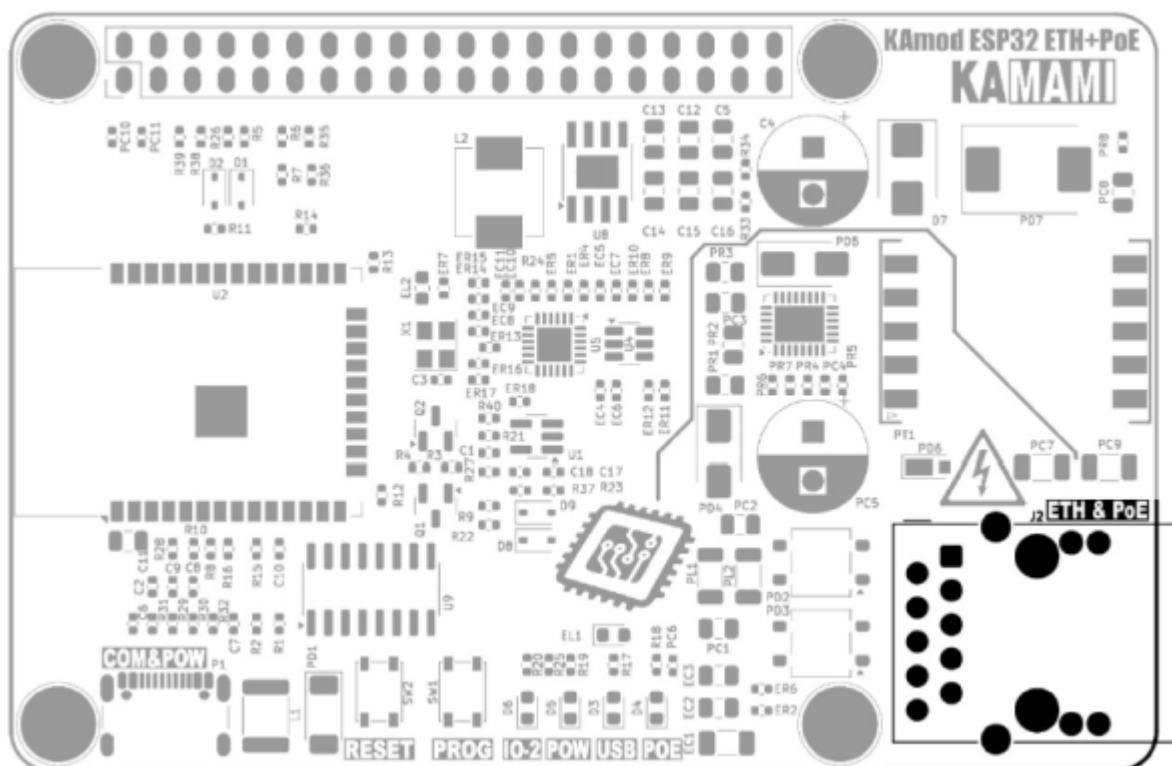
Interfejs Ethernet

Złącze	Funkcja
J2 - ETH & PoE (RJ45)	• Umożliwia połączenie kablowe z siecią internetową

Na płytce KAmód 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



Zasilanie metodą PoE

Złącze	Funkcja
J2 - ETH & PoE (RJ45)	• Odbiera zasilanie z instalacji PoE

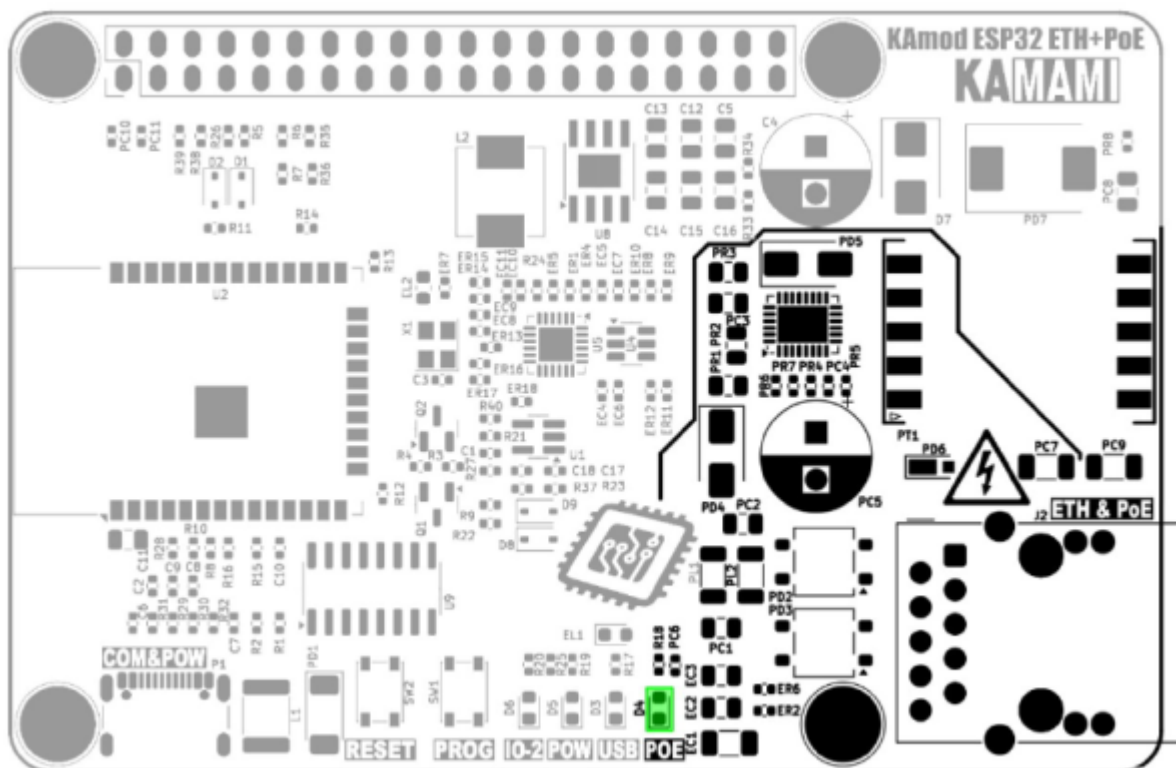
Poprzez złącze **ETH & PoE** (J2) może być dostarczane zasilanie do płytki ewaluacyjnej. Zastosowany kontroler zasilania PoE bazuje na układzie MP8007, który jest kompatybilny ze standardami **IEEE 802.3af** - Powered Devices Type-1 oraz **IEEE 802.3at** - Powered Devices Type-2. Blok zasilania PoE jest 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).
Dźwięk jest najbardziej intensywny, gdy obciążenie bloku zasilania PoE jest niewielkie, natomiast dźwięk cichnie, gdy obciążenie się zwiększa.**

Przy właściwym zasilaniu płytki ewaluacyjnej KAmoD ESP32 ETH+POE wytwarzane są napięcia stabilizowane ok. 5 V oraz 3,3 V, dostępne na złączu szpilkowym J1. Mogą posłużyć do zasilania innych modułów dołączonych do płytki ewaluacyjnej, należy jednak pamiętać, aby sumaryczny prąd nie powinien przekraczać 1,5 A.

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	Funkcja
P1 - COM&POW (USB-C)	- Realizuje funkcję konwertera USB-UART - Umożliwia programowanie modułu ESP32 - Jest alternatywnym wejściem zasilania

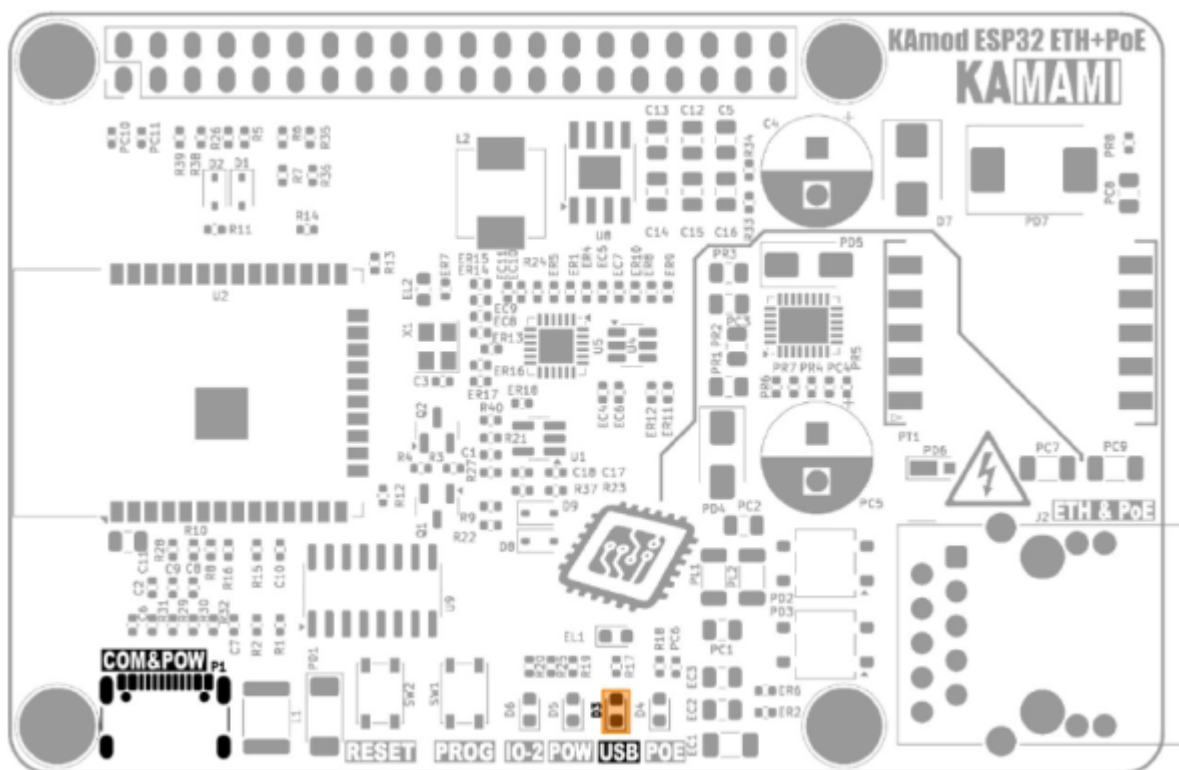
Złącze P1 typu USB-C jest połączone z kontrolerem typu CH340, który realizuje funkcje konwertera USB-UART. Interfejs UART może być używany w docelowej aplikacji, ale służy także do programowania modułu ESP32. Proces programowania może przebiegać całkowicie automatycznie, ponieważ kontroler CH340 steruje 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 (wyjście 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 KAmód 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 zmieniać; wydajność prądowa napięć 5 V oraz 3,3 V może być dużo niższa i będzie zależała od zastosowanego zasilania na złączu USB-C.

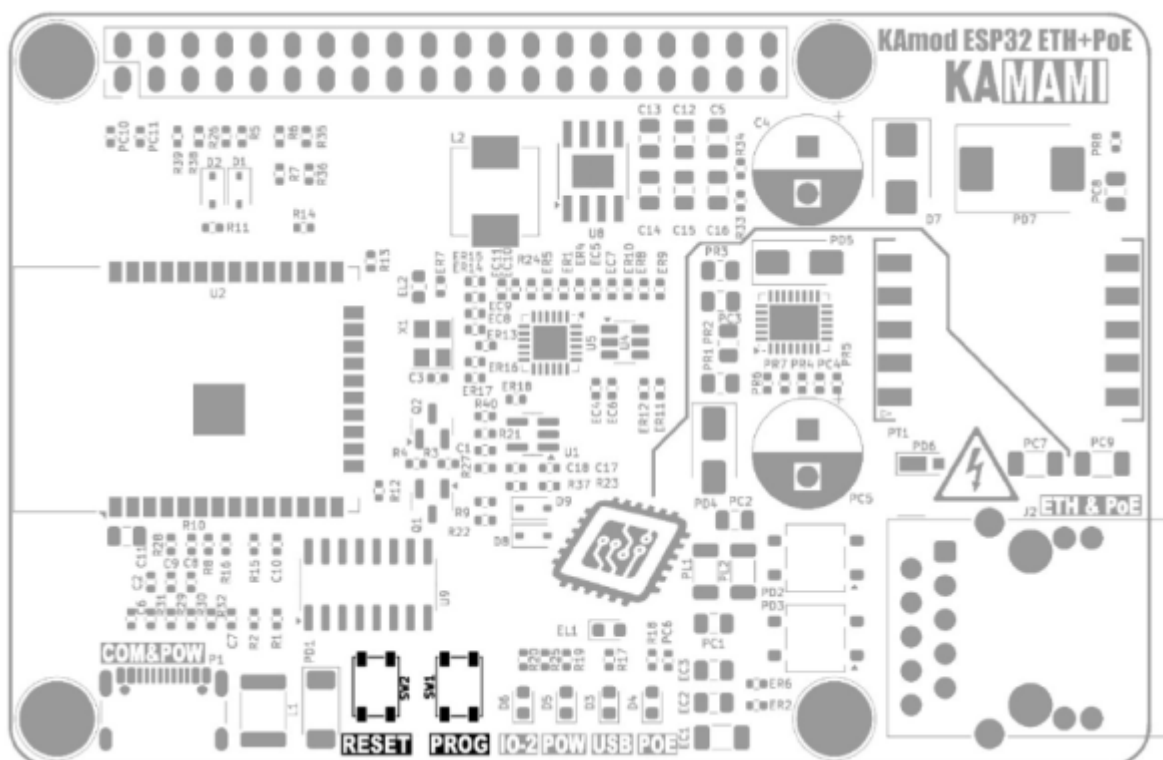


Przyciski resetowania i programowania

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

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.



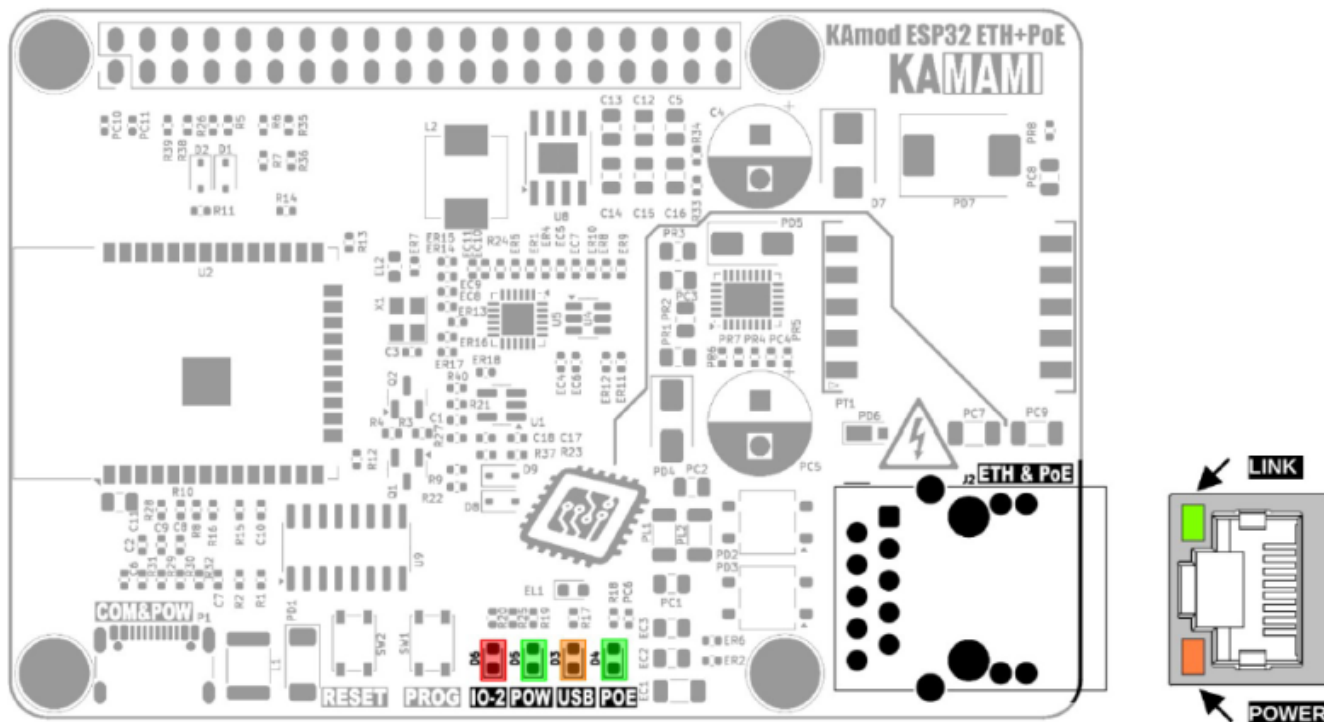
Kontrolki sygnalizacyjne

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

Na płytce KAmód ESP32 ETH+POE znajdują się 4 diody LED, które sygnalizują działanie różnych komponentów – zgodnie z powyższą tabelą.

Dioda D6 (LED IO-2) jest dołączona do wyprowadzenia GPIO2 modułu ESP32. Jej zaświecenie wymaga programowego ustawienia stanu wysokiego na wyprowadzeniu GPIO2.

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

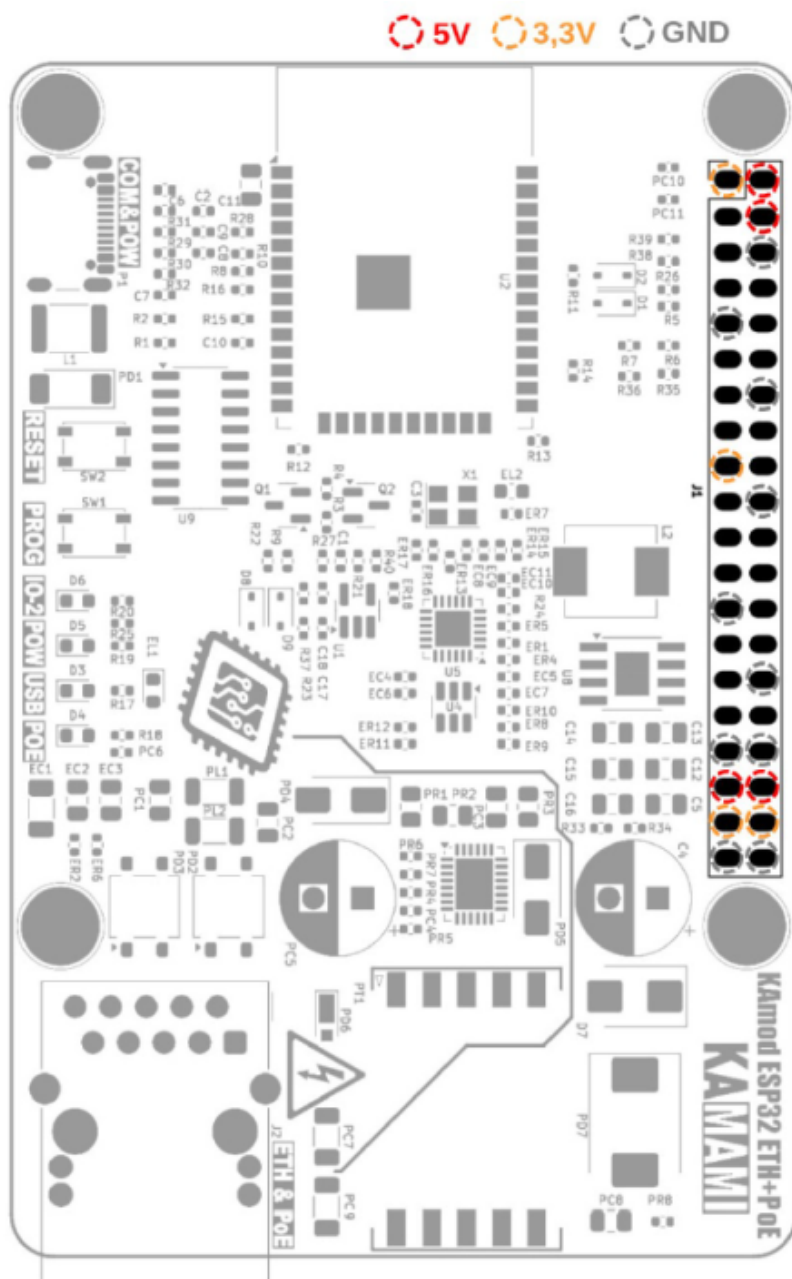


Złącze GPIO w standardzie RPi

Złącze	Funkcja
J1 - Goldpin 2x20	- Złącze w standardzie Raspberry Pi - Wyprowadzone niektóre porty GPIO modułu ESP32 - Wyprowadzone linie zasilania 5 V, 3,3 V, GND

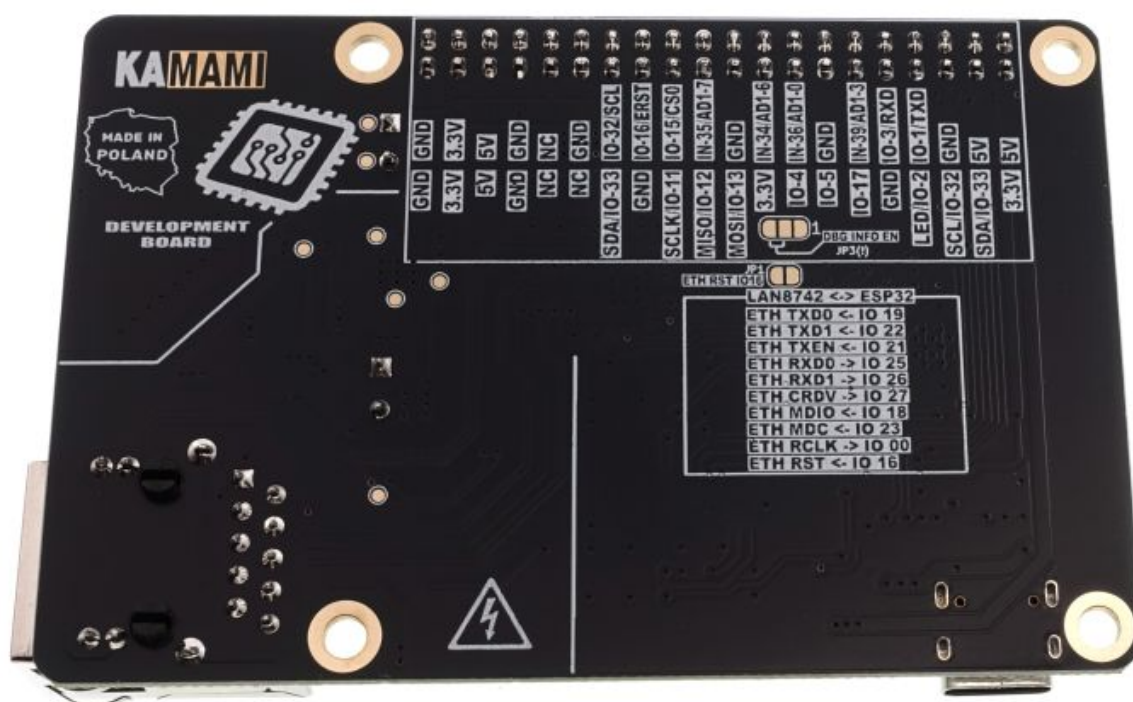
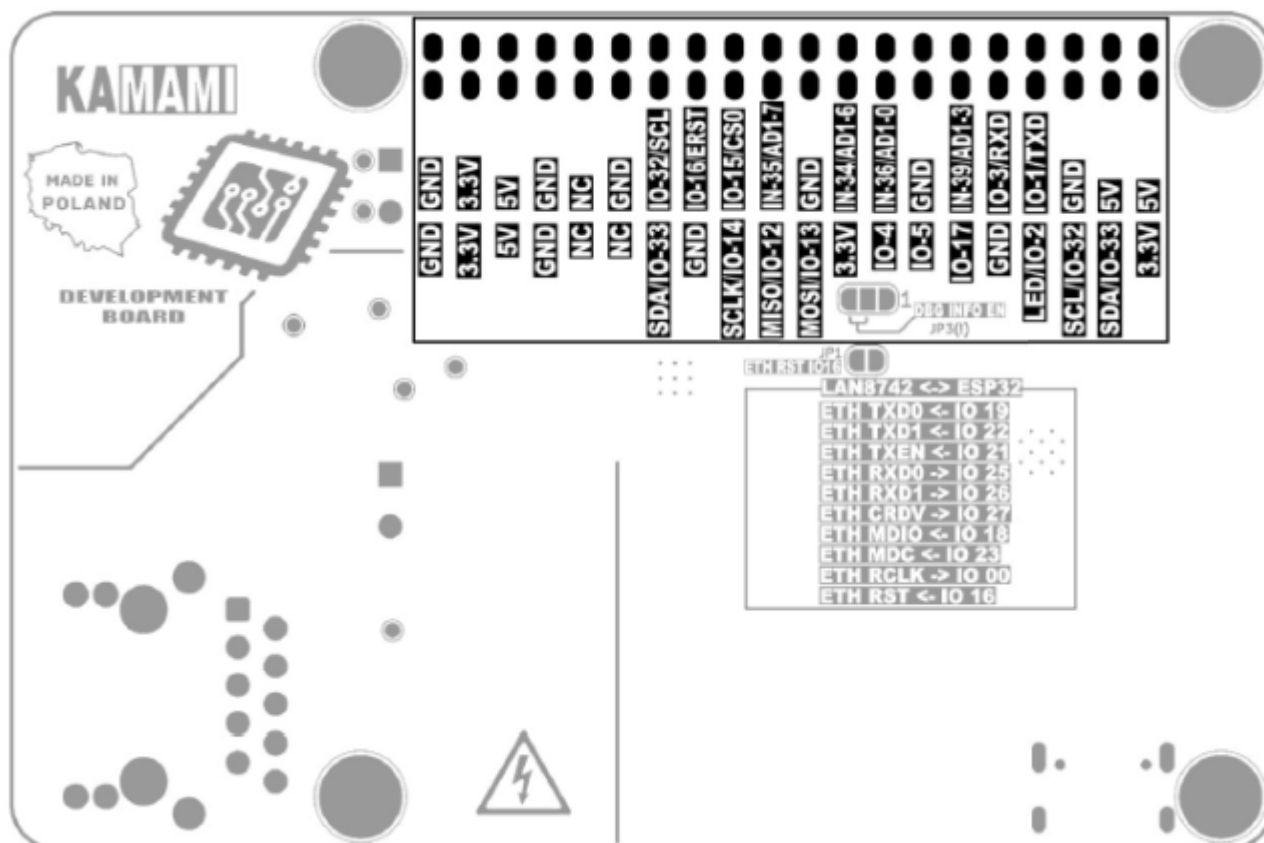
Złącze GPIO (J1) w standardzie Raspberry Pi zawiera 40 szpilek, do których doprowadzone są linie zasilania 5 V, 3,3 V, GND oraz niektóre wyprowadzenia GPIO modułu ESP32. Wyprowadzenia interfejsów UART (TXD, RXD), I2C (SDA, SCL) oraz SPI (MOSI, MISO, SCLK, CS0) zostały rozmieszczone tak, jak ma to miejsce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

Dokładny opis wyprowadzeń oraz ich funkcje pokazuje rysunek i tabela poniżej:



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 (ETH RST)
GPIO33 (SDA)	GPIO32 (SCL)
NC (niepodłączony)	GND
NC (niepodłączony)	NC (niepodłączony)
GPIO18	GND
5 V	5 V
3,3 V	3,3 V
GND	GND

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



Uwagi dotyczące sygnałów wyprowadzonych na złącze GPIO

- Porty **GPIO 34, 35, 36 i 39** modułu ESP32 mogą pracować wyłącznie jako wejścia cyfrowe lub analogowe –

zostały one oznaczone symbolem IN.

- Porty **GPIO 34 i 35** zostały wyposażone w rezystory podciągające pull-up 10k.
- Porty **GPIO 36 i 39** zostały wyposażone w dzielniki napięcia (100k/10k), dzięki czemu można do nich dołączyć napięcie o maksymalnej wartości 35 V.
- Porty **GPIO 32 i 33** zostały dostosowane do funkcjonalności magistrali I2C i zawierają rezystory podciągające pull-up 2,2k.
- Porty **GPIO 1** oraz **GPIO 3** pełnią funkcję interfejsu UART i zostały połączone do modułu konwertera USB-UART oraz równolegle do złącza GPIO J1. Interfejs UART wysyła/odczytuje dane do/z złącza GPIO J2 oraz konwertera USB-UART jednocześnie.
- Porty **GPIO5**, **GPIO12** oraz **GPIO15** konfiguruje pewne parametry modułu ESP32. Ich stan jest odczytywany w momencie uruchomienia (restartu) modułu ESP32, dlatego zostały wyposażone w rezystory podciągające: GPIO5 – pull-up, GPIO12 – pull-down, GPIO15 – pull-down. Należy zadbać o to, aby poziom logiczny na tych wyprowadzeniach w momencie uruchomienia (restartu), odpowiadał stanowi wymuszonemu rezystorami pull-up/down.
- Port **GPIO16** został połączony z sygnałem zerującym driver Ethernet – układ LAN8742. Stan niski na tym wyprowadzeniu blokuje działanie interfejsu Ethernet.

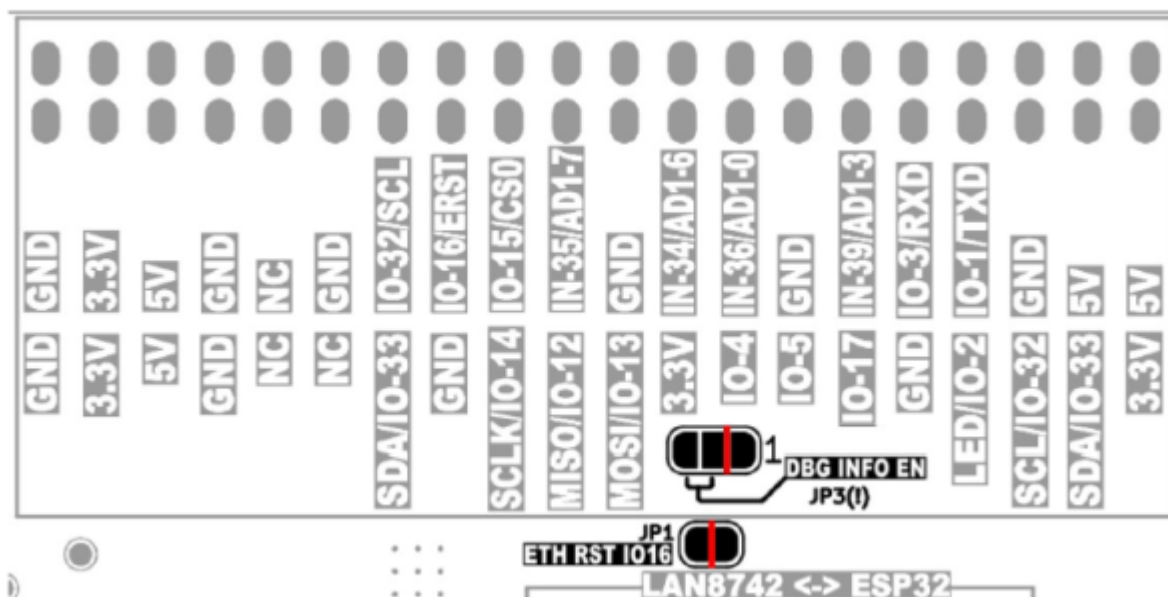
Funkcje zaawansowane

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)

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

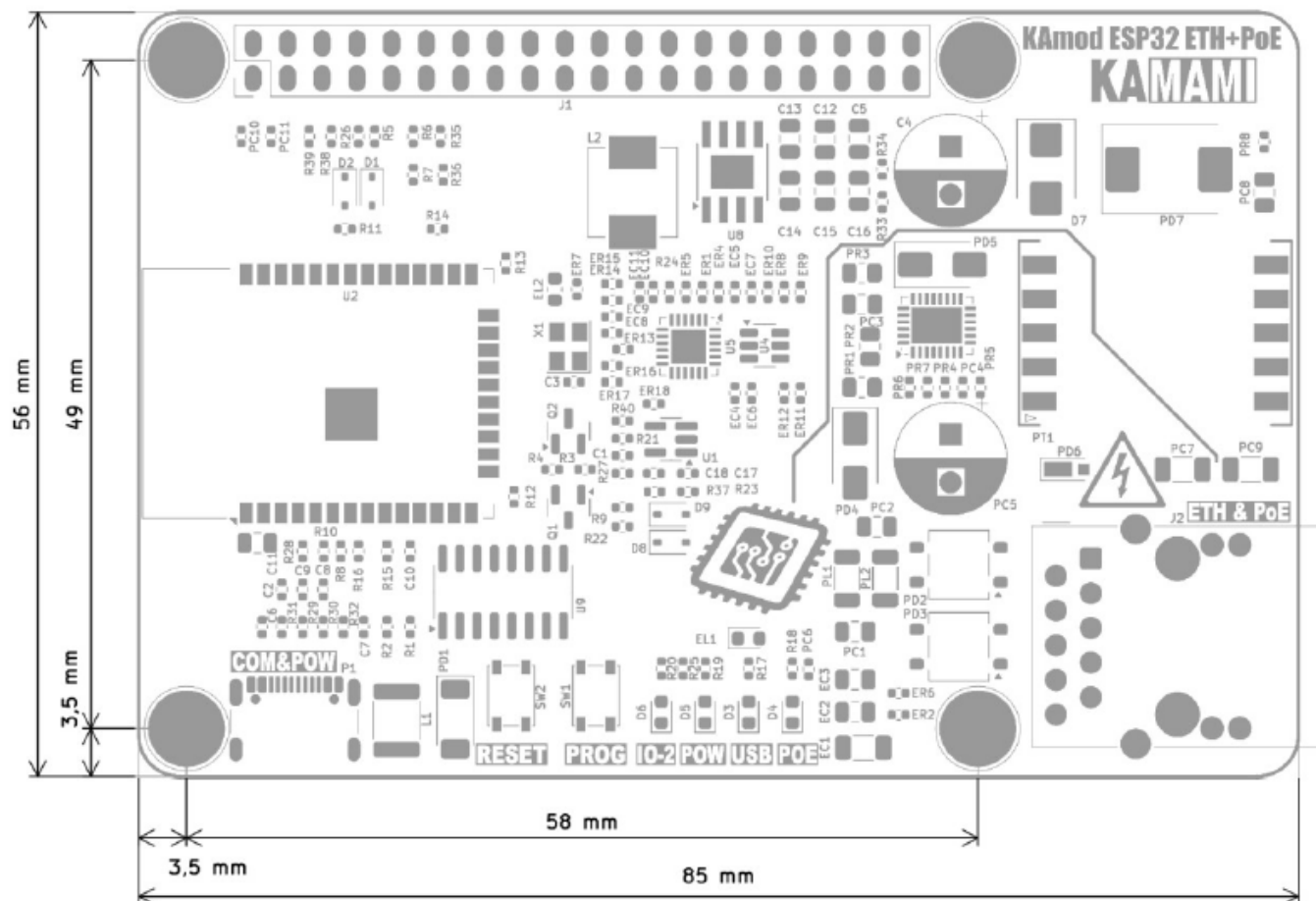
JP1 - ETH RST IO16 jest fabrycznie zwarta (ścieżka miedzi pomiędzy padami) i zapewnia połączenie pomiędzy portem GPIO16 modułu ESP32 i wejściem RESET drivera interfejsu Ethernet. Aby odłączyć port GPIO16 od sygnału zerowania drivera Ethernet należy przeciąć ostrym narzędziem powierzchnię płytki – tak 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 wysyłanie 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.



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_CRD_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_CRIS_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:

Click [here](#) to turn ON the LED.
Click [here](#) to turn OFF the LED.
Click [here](#) to write to Serial (USB).

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:

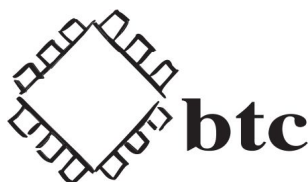
```
ETH.config(myIP, myGW, mySN, myDNS);
```

przed linią:

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

Linki

- [Karta katalogowa układu LAN8742](#)
- [Karta katalogowa układu MP8007](#)
- [Karta katalogowa układu ESP32](#)
- [Karta katalogowa układu CH340](#)
- [Karta katalogowa układu ST1S10](#)
- [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>

Zastrzegamy prawo do wprowadzania zmian bez uprzedzenia.

Oferowane przez nas płytki drukowane mogą się różnić od prezentowanej w dokumentacji, przy czym zmianom nie ulegają jej właściwości użytkowe.

BTC Korporacja gwarantuje zgodność produktu ze specyfikacją.

BTC Korporacja nie ponosi odpowiedzialności za jakiegokolwiek szkody powstałe bezpośrednio lub pośrednio w wyniku użycia lub nieprawidłowego działania produktu.

BTC Korporacja zastrzega sobie prawo do modyfikacji niniejszej dokumentacji bez uprzedzenia.