KAMAMI

KAmodRPi Pico Prototyping Platform (PL)

KAMAMI

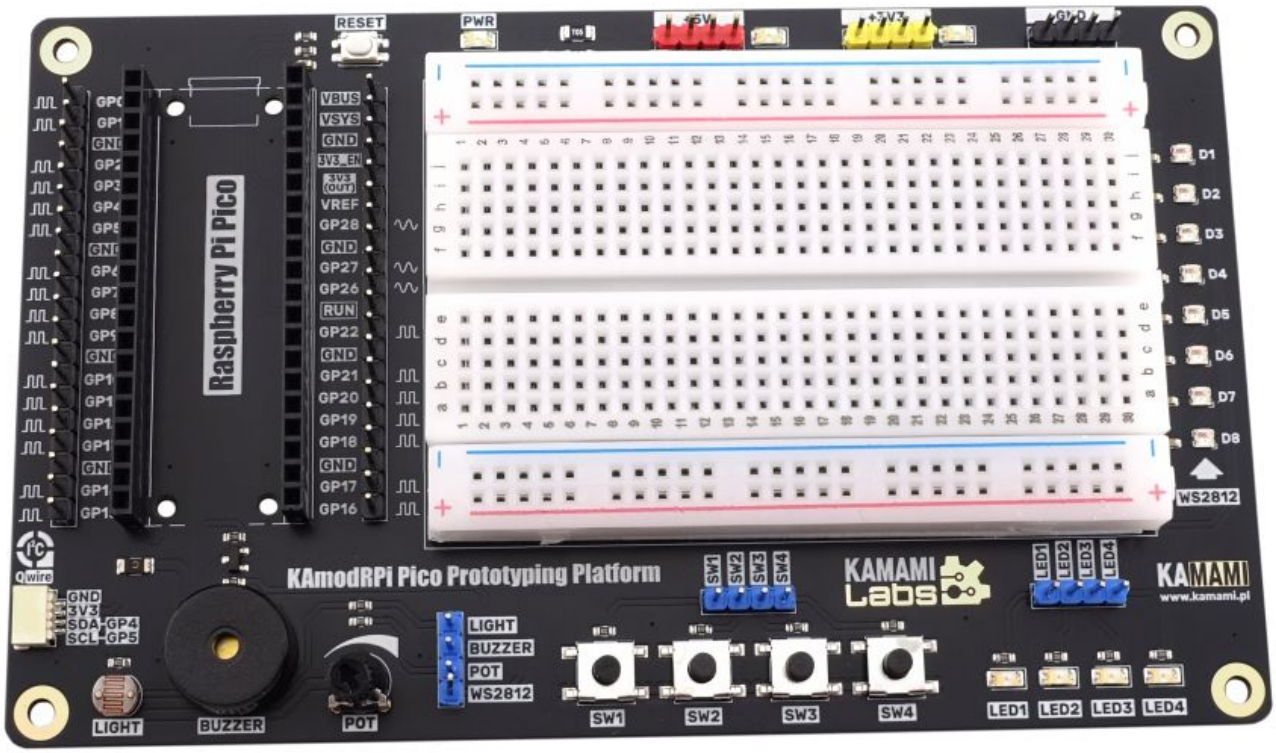
Rev. 202604192044l9
Źródło: https://wiki.kamamilabs.com/index.php?title=KAmodRPi_Pico_Prototyping_Platform_(PL)

Spis treści

Opis

KAmodRPi Pico Prototyping Platform - Uniwersalny System Prototypowy dla Raspberry Pi Pico

KAmodRPi Pico Prototyping Platform to wielofunkcyjna płyta rozszerzeń dla modułów z serii Raspberry Pi Pico, zaprojektowana w celu usprawnienia procesów prototypowania. Urządzenie integruje najczęściej używane komponenty elektroniczne z polem stykowym, tworząc uporządkowane środowisko pracy. Dzięki wyprowadzonym złączom GPIO i wbudowanym peryferiom, użytkownik może błyskawicznie budować i testować układy bez konieczności każdorazowego przygotowywania podstawowych elementów wejścia/wyjścia. Dzięki dedykowanemu przyciskowi resetu oraz złączu I²C Qwire, praca z nowoczesnymi czujnikami staje się płynna i intuicyjna. To dopracowany system prototypowy, który łączy bezpieczeństwo użytkowania, zapewnione przez inteligentne zasilanie, z pełną swobodą rozbudowy o zewnętrzne komponenty.



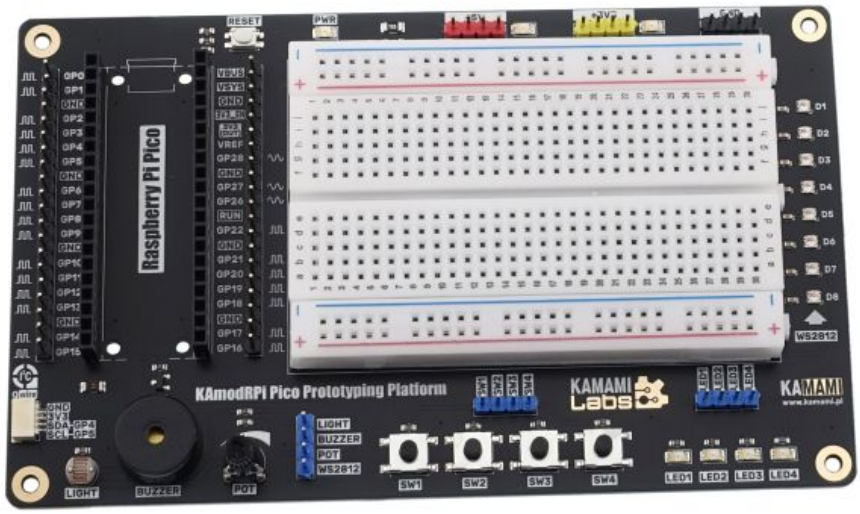
Podstawowe parametry

- Standardowe złącze żeńskie 2×20 pinów dla Raspberry Pi Pico, Pico W, Pico 2
- 20-pinowe złącza męskie z wyprowadzonymi i opisanymi wszystkimi sygnałami RPi Pico (raster 2,54 mm)
- Złącze I²C Qwire typu JST SH 4-pin 1 mm (kompatybilne z Qwiic / STEMMA QT)
- Prototypowa płytka stykowa 400 punktów
- 4 czerwone diody LED
- 8 adresowalnych diod LED WS2812
- 4 przyciski monostabilne
- Wbudowany potencjometr oraz fotorezystor do testów przetwornika ADC
- Zintegrowany przetwornik piezoelektryczny do sygnalizacji dźwiękowej
- Wbudowany przycisk RESET połączony z pinem RUN mikrokontrolera
- Możliwość odłączenia wbudowanych peryferiów za pomocą dołączonych przewodów, co pozwala na pełne wykorzystanie pinów GPIO w innych celach
- Diody LED dla każdej linii zasilającej (PWR, 5V, 3.3V)

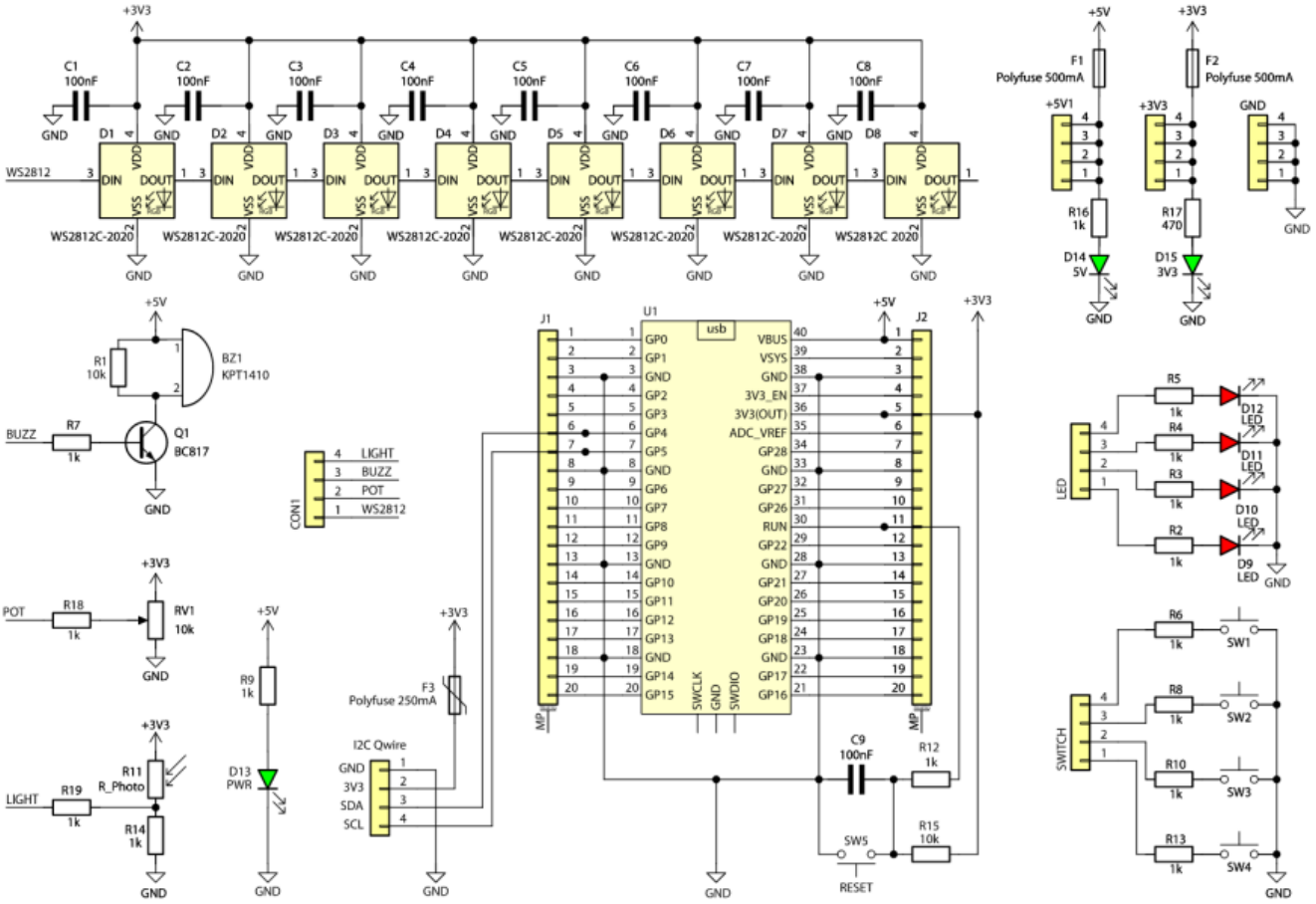
- Napięcie zasilania: 5V (z portu USB Raspberry Pi Pico).
- Zintegrowane bezpieczniki polimerowe na liniach zasilających wyprowadzonych na złącze Qwire oraz szpilki goldpin +5V i +3V3.
- Wymiary: 85 × 140 mm

Wyposażenie standardowe

Kod	Opis
KAmodRPi Pico Prototyping Platform	• Zmontowany i uruchomiony moduł • 6 x gumowa nóżka
Prototypowa płytka stykowa	• Prototypowa płytka stykowa 400 punktów o wymiarach 82x55x10 mm z taśmą klejącą dwustronną
Przewody połączeniowe	• Zestaw 40 szt. przewodów połączeniowych żeńsko-żeńskich, w różnych kolorach o długości 17 cm
Symetryczne końcówki złącz 2,54 mm	• Złącze goldpin 1×40, męsko-męskie, symetryczne idealne do tworzenia połączeń w płytkach stykowych.



Schemat elektryczny

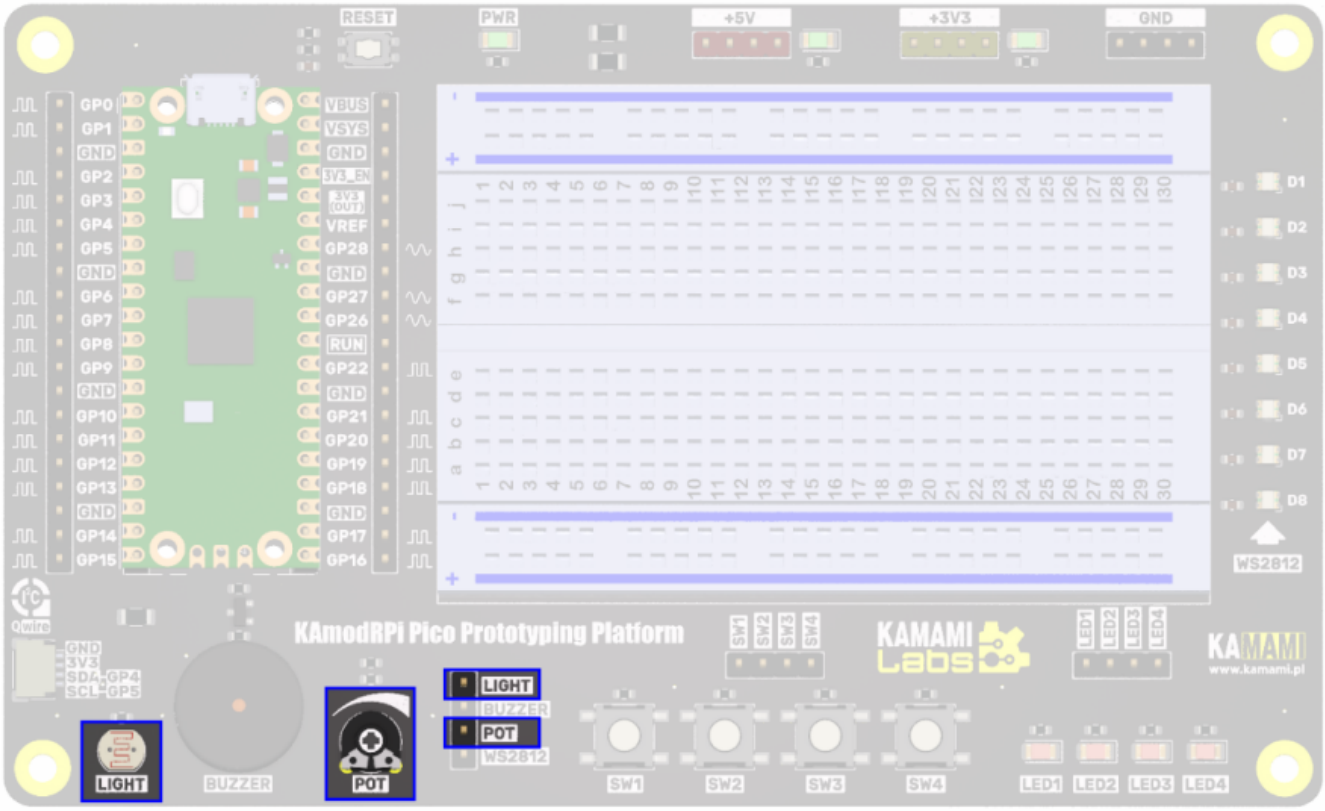


Potencjometr i Fotorezystor

Płytka posiada dwa zintegrowane źródła sygnałów analogowych, które pozwalają na naukę i testowanie przetworników ADC (Analog-to-Digital Converter).

- **Potencjometr (POT)**: Pozwala na ręczną regulację napięcia w zakresie od 0 do 3,3V. Idealny do symulowania odczytów z czujników (np. temperatury), regulacji jasności diod lub sterowania serwomechanizmami.
- **Fotorezystor (LIGHT)**: Zmienia swoją rezystancję w zależności od natężenia oświetlenia padającego na czujnik. Pozwala na tworzenie systemów automatycznego sterowania światłem lub detekcji dnia/nocy.

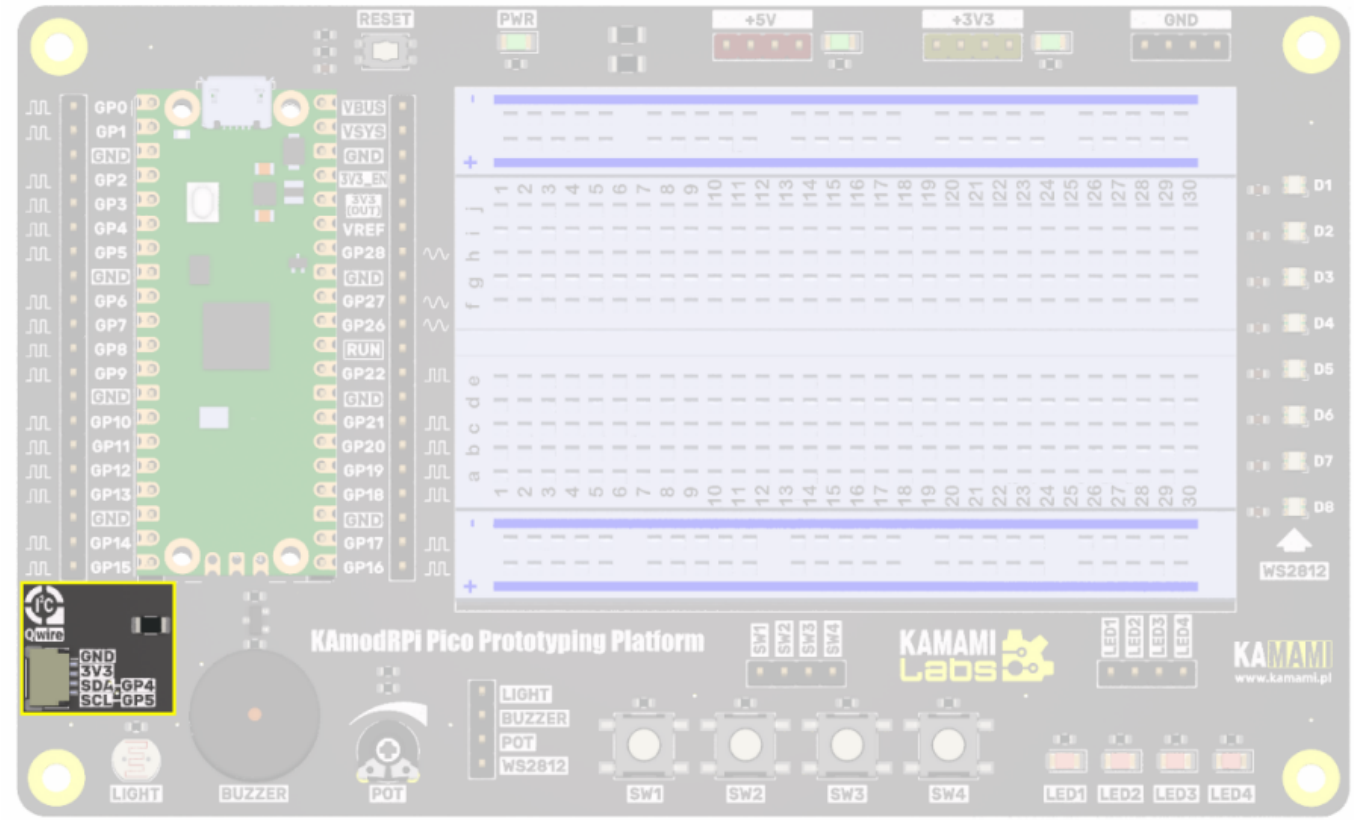
Uwaga: Sygnały te należy podłączyć do pinów Pico obsługujących funkcję ADC (standardowo GP26 i GP27) Piny ADC zostały wyróżnione na płytce symbolem fali ∿, co ułatwia ich szybką identyfikację.



Interfejs I²C

Interfejs I²C został wyprowadzony na złącze Qwire kompatybilne z Qwiic / STEMMA QT. Złącze Qwiic zostało zaprojektowane z myślą o szybkim prototypowaniu z wykorzystaniem zewnętrznych modułów. Dodatkowo linia zasilająca tego złącza posiada niezależny bezpiecznik polimerowy o progu zadziałania 250mA.

- Standard: Złącze typu JST SH 4-pin 1mm kompatybilne z ekosystemem Kamod Qwire, SparkFun Qwiic oraz Adafruit STEMMA QT.
- Komunikacja: Wykorzystuje magistralę I²C (linie SDA-GP4, SCL-GP5) oraz zasilanie 3,3V.
- Zaleta: Pozwala na łączenie kaskadowe (szeregowe) wielu czujników, wyświetlaczy OLED czy kontrolerów silników za pomocą jednego standardowego przewodu.

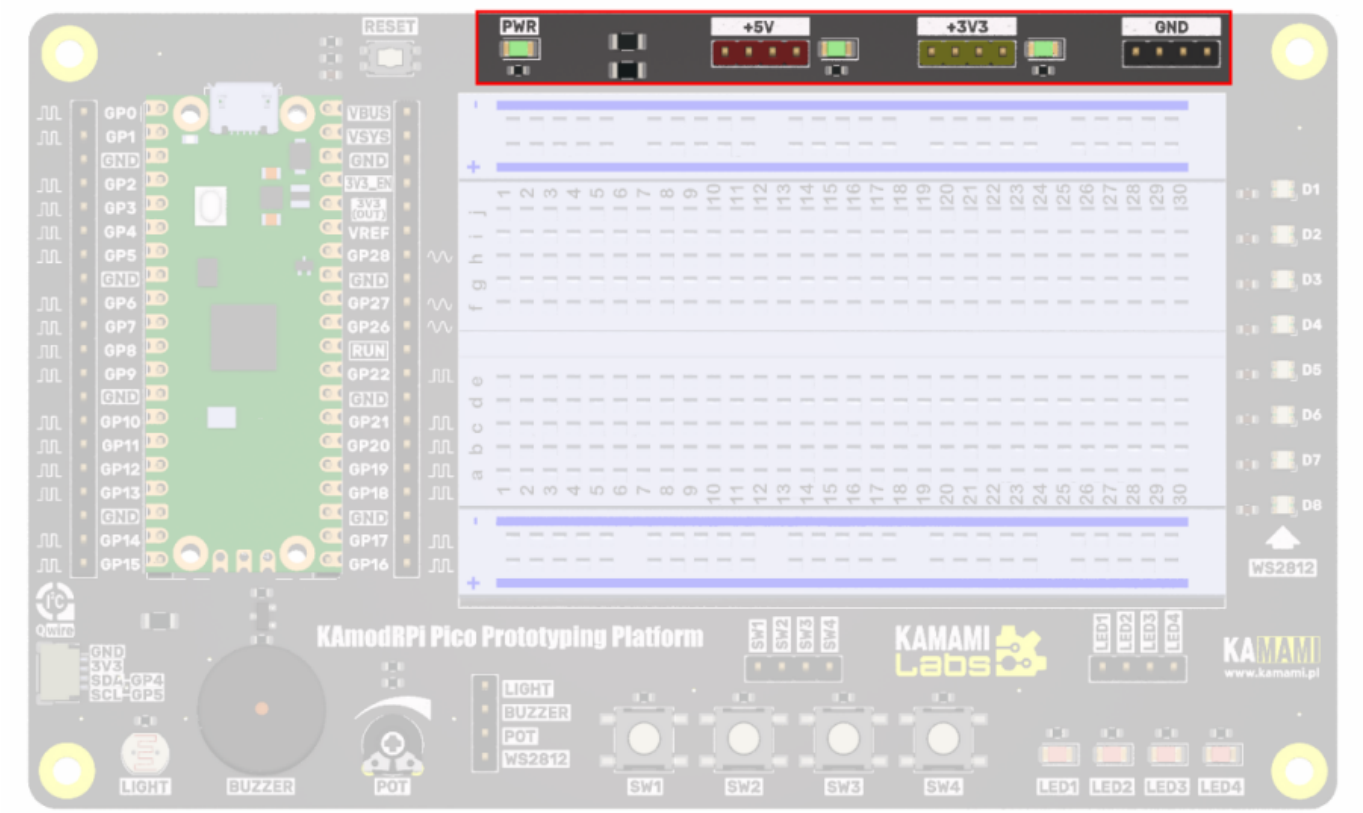


Zasilanie

KAmodRPi Pico Prototyping Platform jest zasilany napięciem 5V z portu USB Raspberry Pi Pico. Platforma została wyposażona w system monitorowania napięć oraz ochrony przed uszkodzeniami, co czyni ją bezpiecznym narzędziem do nauki i eksperymentów.

- **Wizualna kontrola napięć:** Trzy dedykowane diody LED informują o statusie kluczowych linii zasilających:
  - **PWR**: Sygnalizuje obecność napięcia bezpośrednio z portu USB (5V przed zabezpieczeniami).
  - **+5V**: Informuje o poprawnym zasilaniu głównej szyny 5V (5V za zabezpieczeniami).
  - **+3V3**: Potwierdza stabilną pracę regulatora napięcia mikrokontrolera (3V3 za zabezpieczeniami).
- **Ochrona bezpiecznikowa:** Linie zasilające wyprowadzone na szpilki goldpin +5V oraz +3V3 są chronione przez bezpieczniki polimerowe. W przypadku wystąpienia zwarcia lub przeciążenia na płytce stykowej, bezpiecznik automatycznie odcina dopływ prądu. Chronią one zarówno samą płytę bazową, jak i podłączony moduł Raspberry Pi Pico przed skutkami przypadkowych zwarć lub zbyt dużego poboru prądu w budowanym prototypie. Po usunięciu przyczyny awarii, element regeneruje się samoczynnie, przywracając zasilanie bez konieczności wymiany jakichkolwiek części.

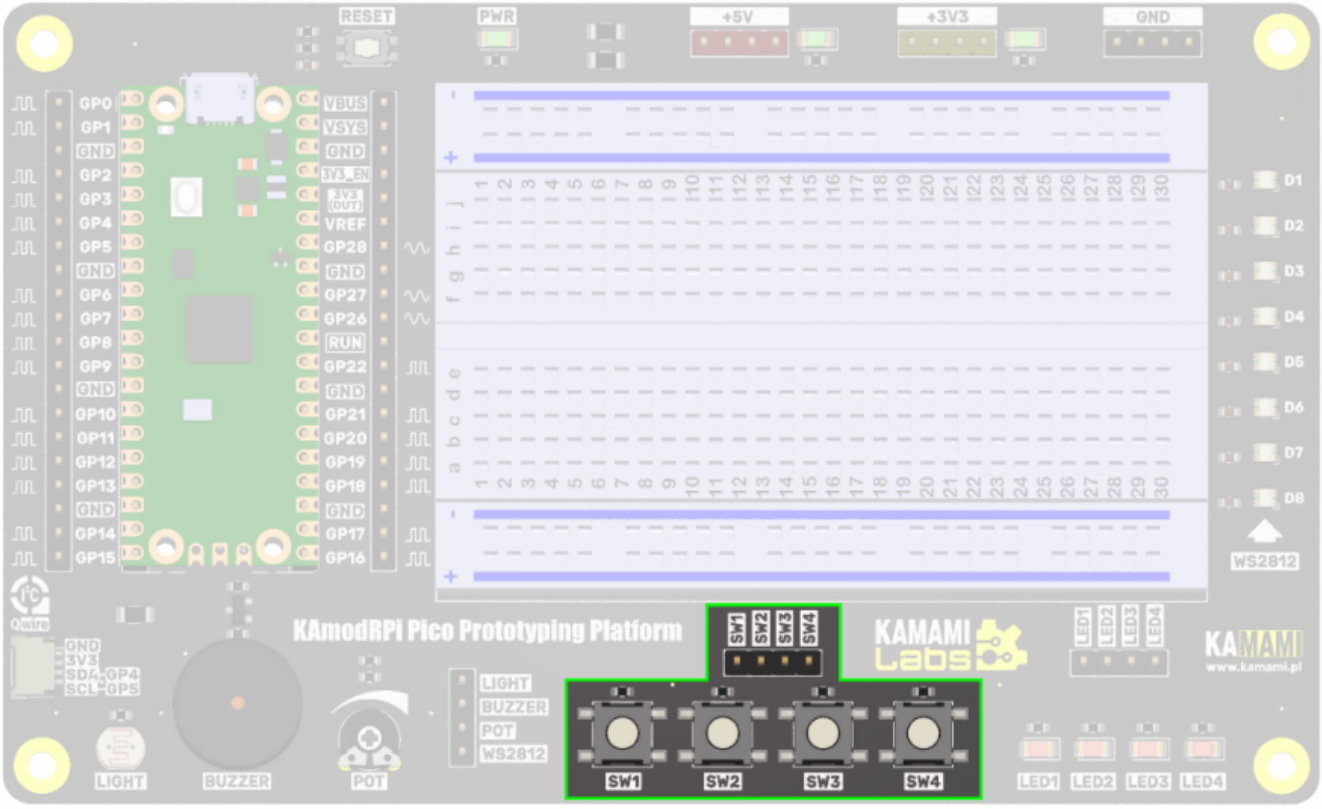
Ważne: Szyny zasilające na listwach goldpin posiadają zabezpieczenie do 500 mA. Jeśli Twój projekt wymaga większego prądu (np. wiele mocnych serwomechanizmów lub długie paski LED RGB), zalecamy zastosowanie zewnętrznego źródła zasilania, aby nie doprowadzić do zadziałania bezpieczników i tymczasowego odłączenia peryferiów.



Przyciski

Moduł KAmodRPi Pico Prototyping Platform wyposażony został w cztery przyciski monostabilne (SW1 – SW4) typu microswitch, przeznaczone do interakcji z użytkownikiem.

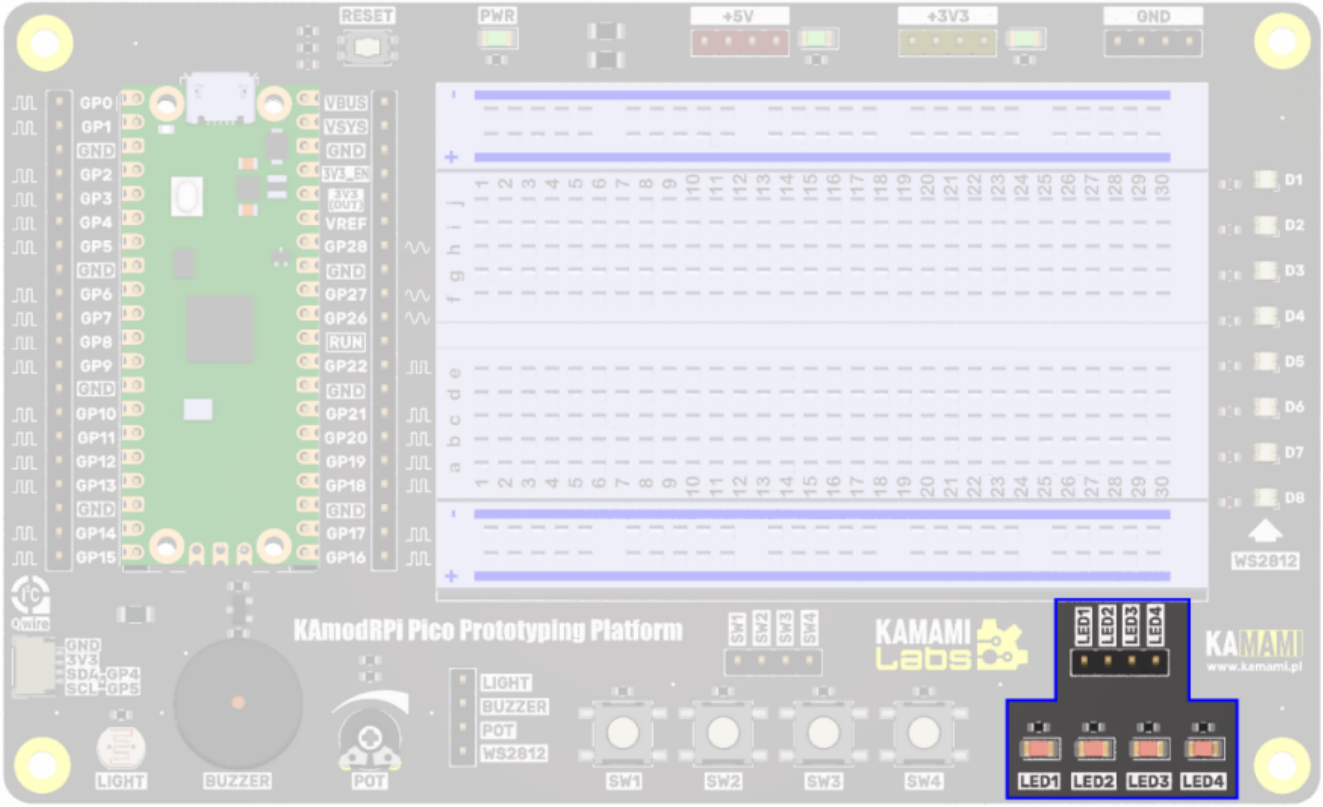
- Działanie: Przyciski łączą piny GPIO z masą (GND).
- Programowanie: W kodzie programu należy skonfigurować piny jako wejścia z włączonym wewnętrznym rezystorem podciągającym (Internal Pull-Up). Stan niski (0) oznacza naciśnięcie przycisku, a stan wysoki (1) jego spoczynek.
- Zastosowanie: Tworzenie interfejsów sterowania, przełączanie trybów pracy lub wyzwalanie zdarzeń.



Diody LED

Na płytce znajdują się cztery czerwone diody LED (LED1 – LED4), umieszczone obok przycisków, co ułatwia tworzenie intuicyjnych interfejsów wizualnych.

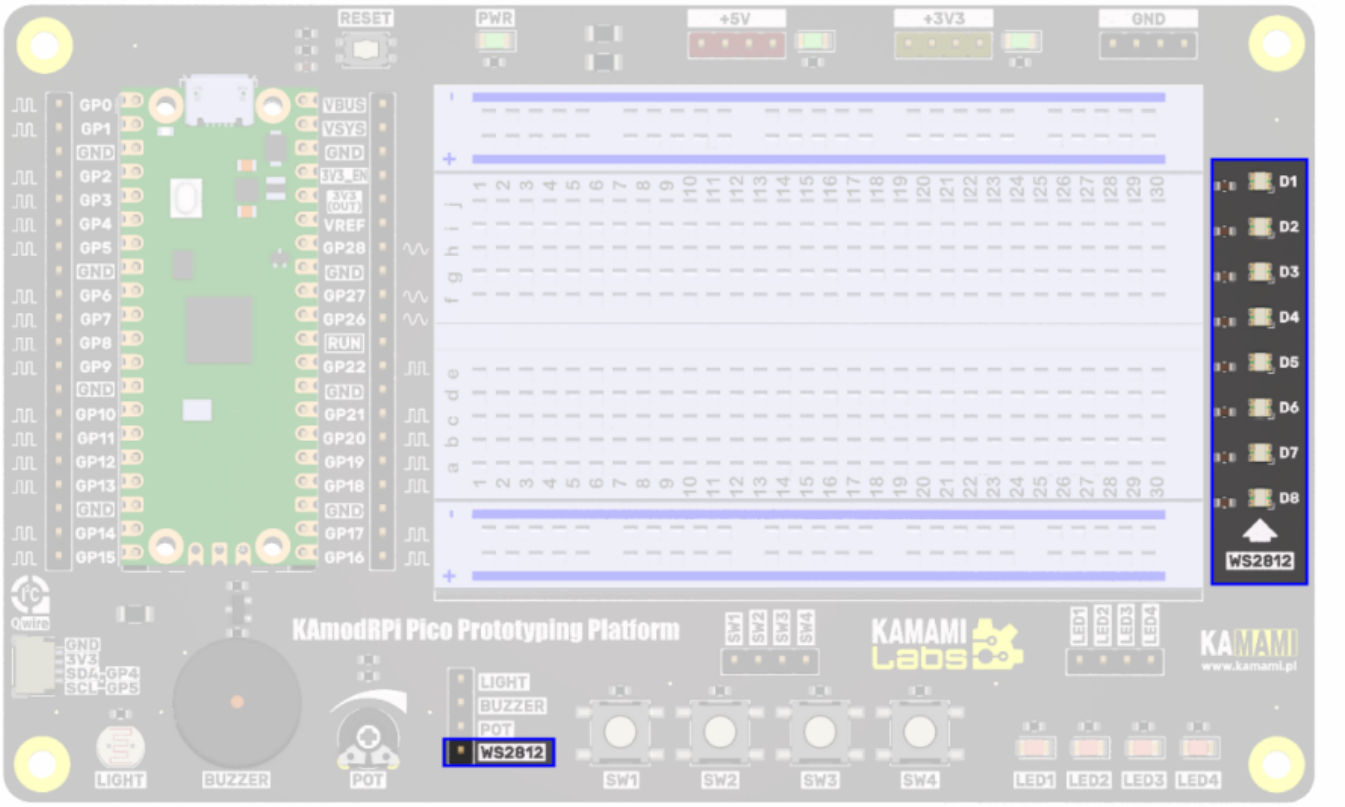
- Działanie: Diody są sterowane sygnałem cyfrowym. Załączenie diody następuje po podaniu stanu wysokiego (1) na odpowiedni pin GPIO.
- Konfiguracja: Każda dioda posiada zintegrowany rezystor ograniczający prąd, co pozwala na bezpośrednie połączenie z pinami mikrokontrolera bez obawy o ich uszkodzenie.
- Zastosowanie: Sygnalizacja stanów pracy, debugowanie kodu oraz informowanie o aktywności urządzenia.



Diody WS2812

Płytka wyposażona jest w pasek ośmiu adresowalnych diod LED WS2812 (standard Neopixel), które pozwalają na wyświetlanie dowolnych kolorów z palety RGB.

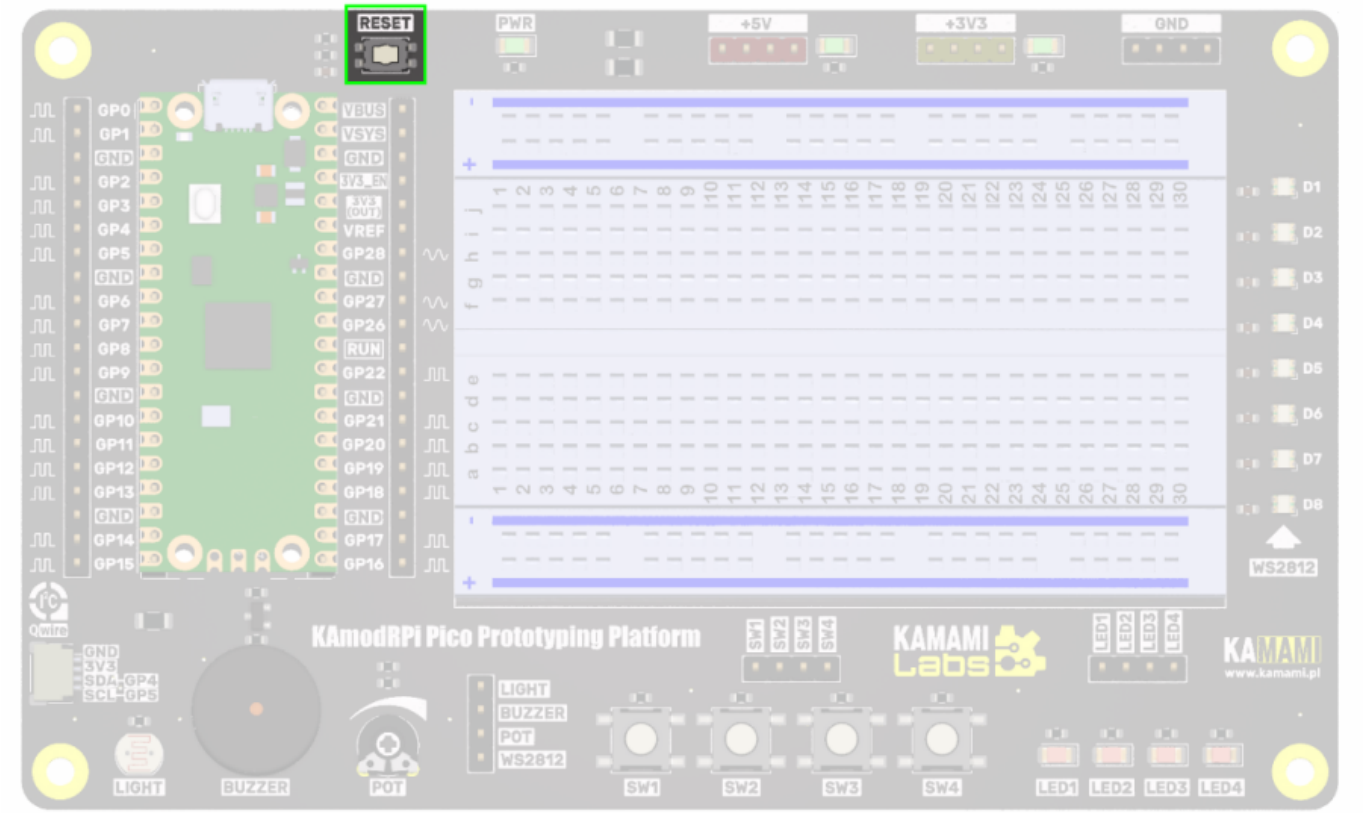
- Działanie: Wszystkie 8 diod sterowanych jest szeregowo za pomocą tylko jednego pinu GPIO. Każda dioda może świecić niezależnie innym kolorem i jasnością.
- Programowanie: Wymagana jest biblioteka obsługująca protokół jednoprzewodowy (np. Adafruit NeoPixel dla Arduino lub wbudowany moduł neopixel w MicroPythonie).
- Zastosowanie: Tworzenie efektownych iluminacji, wizualizacja danych z czujników (np. zmiana koloru w zależności od temperatury) lub sygnalizacja statusów systemu.



Przycisk RESET

Dedykowany przycisk RESET to udogodnienie, którego brakuje w standardowym module Raspberry Pi Pico.

- Działanie: Przycisk jest połączony na z pinem RUN mikrokontrolera oraz masą (GND). Jego naciśnięcie powoduje natychmiastowe przerwanie pracy procesora i ponowne uruchomienie programu.
- Zaleta: Eliminuje konieczność częstego odłączania i podłączania kabla USB w celu zrestartowania układu lub przejścia w tryb wgrywania oprogramowania (w połączeniu z przyciskiem BOOTSEL na module Pico).
- Bezpieczeństwo: Pozwala na szybkie zatrzymanie pracy urządzenia w przypadku błędnego działania prototypu.



Przykładowy program

Przykładowy program testowy został przygotowany zarówno w środowisku Arduino IDE jak i w MicroPythonie (uproszczona wersja), i służy do kompleksowego przetestowania wszystkich funkcji KAmodRPi Pico Prototyping Platform. Demonstruje on interakcję między czujnikami analogowymi a elementami wykonawczymi (LED, Buzzer, RGB).
Wymagania **Arduino**:

- Board Manager: `Raspberry Pi Pico/RP2040/RP2350`
- Biblioteki: `Adafruit NeoPixel`

Wymagania **MicroPython**:

- Skrypt wykorzystuje standardowe biblioteki MicroPythona (`machine, neopixel`)

Działanie programu:

1. Sekwencja startowa
   • Po uruchomieniu i inicjalizacji portu szeregowego, program wykonuje charakterystyczną sekwencję dźwiękową za pomocą buzzera, sygnalizując gotowość do pracy.
2. Inteligentne sterowanie diodami LED (Sekcja Cyfrowa)
   • Tryb automatyczny: Czerwone diody LED wykonują efekt płynnego "wędrowania" (knight rider). Prędkość animacji jest dynamicznie regulowana przez fotorezystor – im jaśniej, tym szybciej diody zmieniają stan.
   • Tryb ręczny: Po naciśnięciu dowolnego z 4 przycisków, animacja zostaje wstrzymana, a diody LED odpowiadają bezpośrednio stanom przycisków.
3. Interaktywny Buzzer
   • Każdy z czterech przycisków przypisany jest do innej częstotliwości dźwięku (od 500 Hz do 2000 Hz). Naciśnięcie przycisku generuje sygnał dźwiękowy o unikalnym tonie.
4. Pasek LED RGB (WS2812)
   • Program cyklicznie wypełnia pasek 8 diod kolorami: czerwonym, zielonym i niebieskim.

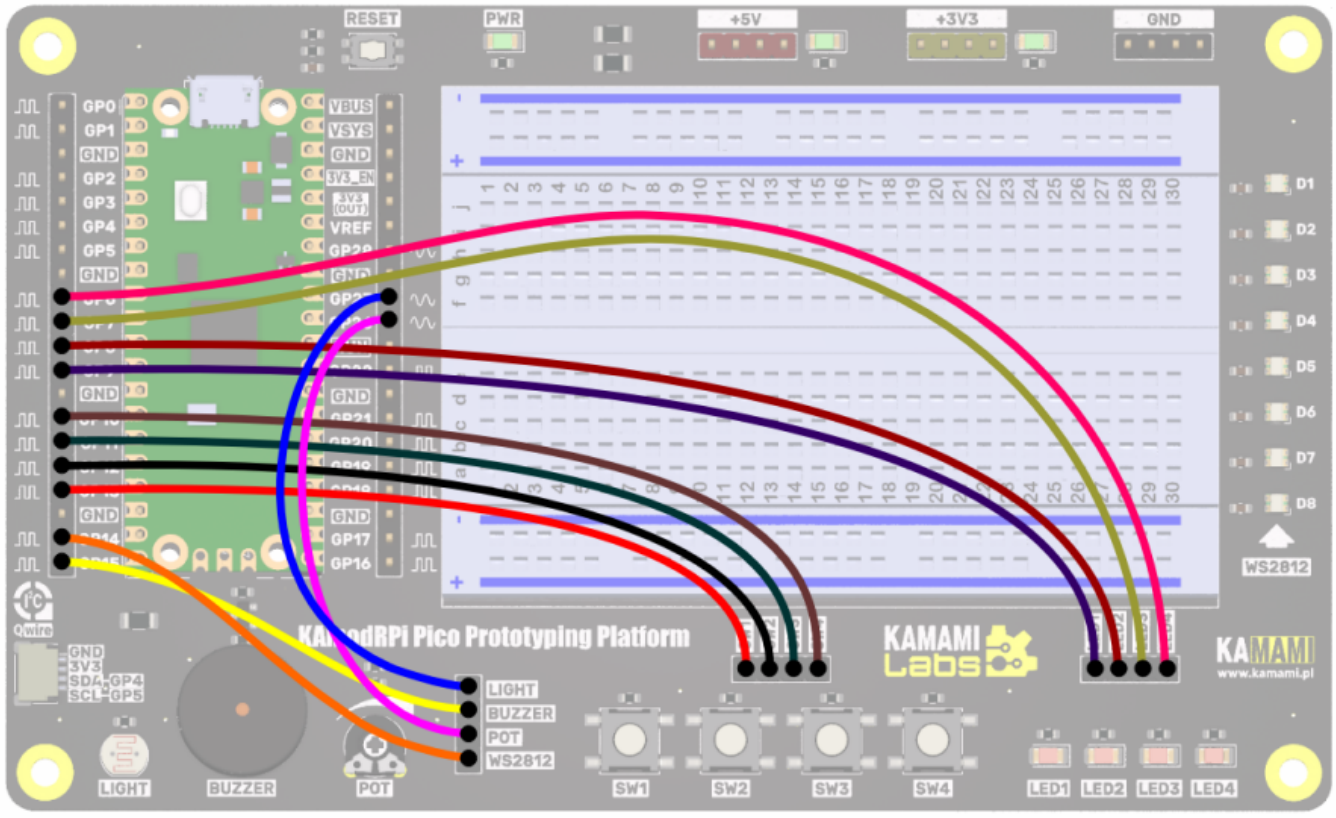
• Szybkość wypełniania paska i zmiany kolorów jest kontrolowana w czasie rzeczywistym za pomocą wbudowanego potencjometru.

5. Monitor Szeregowy

• Program wysyła powitanie `KAmodRPi Pico Prototyping Platform` na port szeregowy (115200 bps) przy każdym restarcie, ułatwiając weryfikację połączenia z komputerem.

5. Przypisanie sygnałów do GPIO

Komponent	Typ sygnału	GPIO
Przyciski (SW1-SW4)	Digital (PULL_UP)	GP10, GP11, GP12, GP13
Diody LED (LED1-LED4)	Digital (High = ON)	GP6, GP7, GP8, GP9
WS2812 (RGB)	One-Wire (Serial)	GP14
Buzzer	PWM	GP15
Potencjometr (POT)	Analog (ADC)	GP26
Fotorezystor (LIGHT)	Analog (ADC)	GP27
I²C (Qwire)	I²C	GP4 (SDA), GP5 (SCL), 3V3 zabezpieczone 250 mA



Program testowy w Arduino:

```
//board manager potrzebuje: "Raspberry Pi Pico/RP2040/RP2350" ->
https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.j
son
//potrzebna biblioteka: Adafruit NeoPixel

#include <Adafruit_NeoPixel.h>
//#include <Wire.h>

//--------------------------------
#define LED1    6
#define LED2    7
```

```
#define LED3    8
#define LED4    9
#define LED_ON      1
#define LED_OFF     0
#define LED_ALL     4
#define LED_DELAY   8

#define SW1     10
#define SW2     11
#define SW3     12
#define SW4     13
#define SW_ON       0
#define SW_OFF      1
#define SW_ALL      4

#define POT_PIN     26
#define LIGHT_PIN   27

#define WS2812_PIN  14
#define NUMPIXELS   8
#define BUZZ_PIN    15

//---------------------------------
int led_sw_man(int delay);
int ws_man(int delay);
void buzz_show(int d);

int buttons;
int adj;
int buzz_time;


Adafruit_NeoPixel pixels(NUMPIXELS, WS2812_PIN, NEO_GRB + NEO_KHZ800);

//---------------------------------
void setup() {
  Serial.begin(115200);
  delay(2000);
  Serial.println("KAmodRPi Pico Prototyping Platform");

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);

  pinMode(SW1, INPUT_PULLUP);
  pinMode(SW2, INPUT_PULLUP);
  pinMode(SW3, INPUT_PULLUP);
  pinMode(SW4, INPUT_PULLUP);

  pinMode(BUZZ_PIN, OUTPUT);

  pixels.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  pixels.show();
  buzz_show(200);
}

//---------------------------------
void loop() {
```

```
  adj = analogRead(LIGHT_PIN);
  buttons = led_sw_man(adj/16);

  if (buttons > 0) {
    buzz_time = 10;
    //digitalWrite(BUZZ_PIN, 1);
    if (buttons == 1) tone(BUZZ_PIN, 500);
    if (buttons == 2) tone(BUZZ_PIN, 1000);
    if (buttons == 4) tone(BUZZ_PIN, 1500);
    if (buttons == 8) tone(BUZZ_PIN, 2000);
  }
  if (buzz_time > 0){
      buzz_time--;
      //if (buzz_time == 0) digitalWrite(BUZZ_PIN, 0);
      if (buzz_time == 0) noTone(BUZZ_PIN);
  }

  adj = analogRead(POT_PIN);
  ws_man(adj/16);

  delay(10);
}

//--------------------------------
int ws_man(int delay){
  static int ws_i = 0;
  static int ws_led = 0;
  static int ws_color = 0;

  if (ws_i < delay){
    ws_i++;
  } else {
    ws_i=0;
    for (uint l = 0; l<NUMPIXELS; l++){
      if (ws_led >= l){
        if (ws_color == 0) pixels.setPixelColor(l, 255, 0, 0);
        if (ws_color == 1) pixels.setPixelColor(l, 0, 255, 0);
        if (ws_color == 2) pixels.setPixelColor(l, 0, 0, 255);
      }
    }
    pixels.show();

    ws_led++;
    if (ws_led >= NUMPIXELS) {
      ws_led = 0;
      ws_color++;
      if (ws_color > 2) ws_color = 0;
    }
  }
  return 1;
}

//--------------------------------
int led_sw_man(int delay){
  static int led_i = 0;
  static int led_pos = 0;
  static int led_dir = 0;
  static int led_map[LED_ALL] = {LED1, LED2, LED3, LED4};
  int push_map;
```

```
  static int sw_map[SW_ALL] = {SW1, SW2, SW3, SW4};

  push_map = 0;
  for (int i=0; i<SW_ALL; i++){
    if (digitalRead(sw_map[i]) == SW_ON){
      push_map |= (1<<i);
    }
  }

  if (push_map > 0){
    for (int i=0; i<SW_ALL; i++){
      if ((push_map & (1<<i)) > 0){
        digitalWrite(led_map[i], LED_ON);
      } else {
        digitalWrite(led_map[i], LED_OFF);
      }
    }

  //no any push
  } else {
    if (led_i < delay){
      led_i++;
    } else {
      led_i=0;
      if (led_pos >= 0 && led_pos < LED_ALL)
        digitalWrite(led_map[led_pos], LED_OFF);

      if (led_dir == 0){
        led_pos++;
        if (led_pos >= LED_ALL){
          led_pos = (LED_ALL - 2);
          led_dir = 1;
        }
      } else {
        led_pos--;
        if (led_pos < 0){
          led_pos = 1;
          led_dir = 0;
        }
      }

      if (led_pos >= 0 && led_pos < LED_ALL){
        digitalWrite(led_map[led_pos], LED_ON);
      }
    }
  }

  return push_map;
}

//--------------------------------
void buzz_show(int d){
  tone(BUZZ_PIN, 1000); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 1000); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 1000); delay(d/4);
```

```
  noTone(BUZZ_PIN); delay(d/4);
  tone(BUZZ_PIN, 1000); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 1200); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 1100); delay(d/4);
  noTone(BUZZ_PIN); delay(d/4);
  tone(BUZZ_PIN, 1100); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 1000); delay(d/4);
  noTone(BUZZ_PIN); delay(d/4);
  tone(BUZZ_PIN, 1000); delay(d);
  noTone(BUZZ_PIN); delay(d);

  tone(BUZZ_PIN, 900); delay(d/4);
  noTone(BUZZ_PIN); delay(d/4);
  tone(BUZZ_PIN, 1000); delay(d);
  noTone(BUZZ_PIN); delay(d);
}
```

Uproszczona wersja przykładowego programu testowego w MicroPythonie:

```
import machine
import utime
import neopixel

# --- Konfiguracja PINów ---
# Diody LED i Przyciski
led_pins = [6, 7, 8, 9]
leds = [machine.Pin(p, machine.Pin.OUT) for p in led_pins]

sw_pins = [10, 11, 12, 13]
buttons = [machine.Pin(p, machine.Pin.IN, machine.Pin.PULL_UP) for p in sw_pins]

# Analogowe (Potencjometr i Fotorezystor)
pot = machine.ADC(26)
ldr = machine.ADC(27)

# Buzzer i WS2812
buzzer = machine.PWM(machine.Pin(15))
pixels = neopixel.NeoPixel(machine.Pin(14), 8)

def play_tone(freq, duration):
    if freq > 0:
        buzzer.freq(freq)
        buzzer.duty_u16(32768) # 50% wypełnienia
    utime.sleep_ms(duration)
    buzzer.duty_u16(0) # Wyłącz dźwięk

print("RPi Pico Prototyping Platform - Start Test")
play_tone(1000, 100) # Sygnał startowy

while True:
```

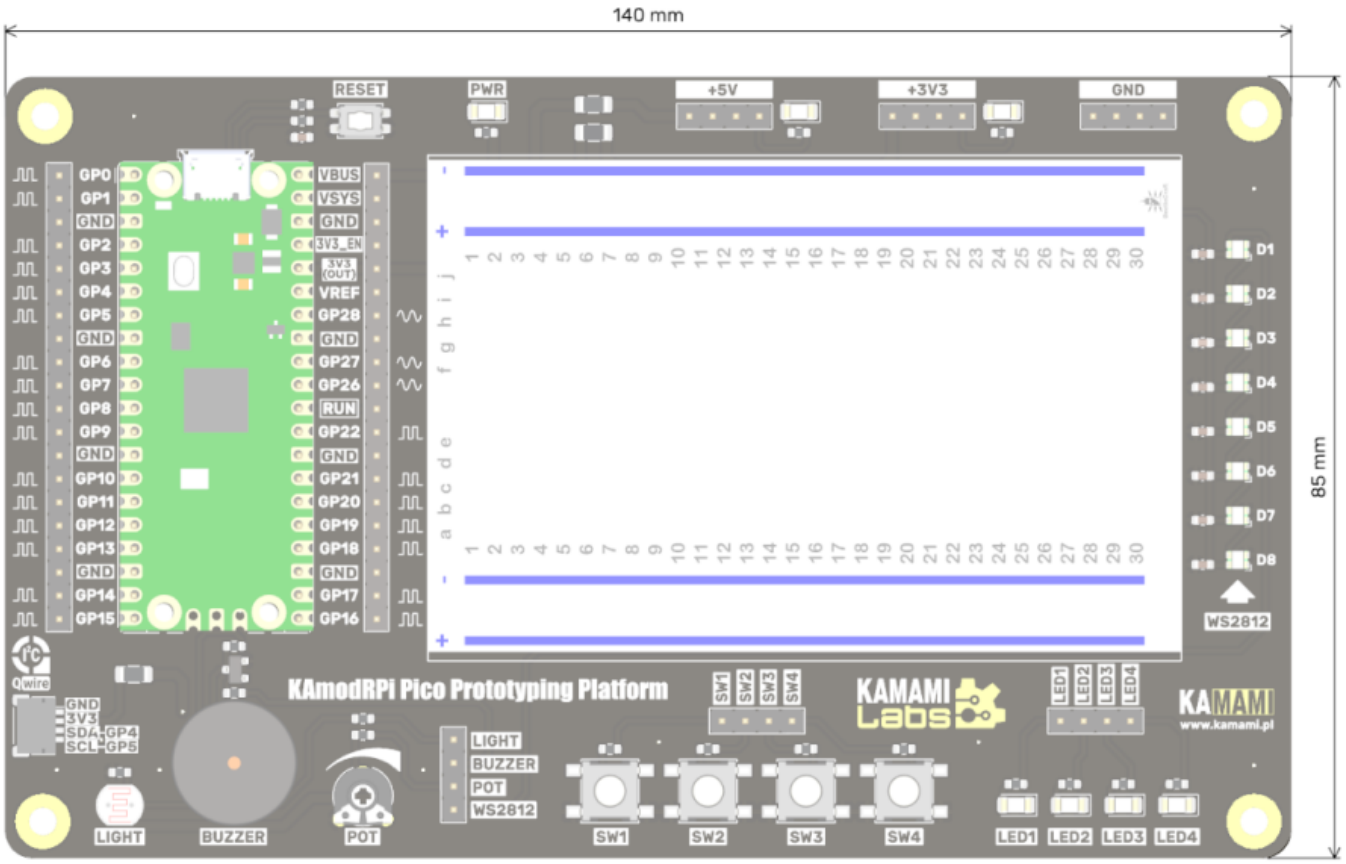
```
# 1. Odczyt fotorezystora i sterowanie diodami LED
light_val = ldr.read_u16()
# Im ciemniej, tym szybciej mrugają diody (prosta logika testowa)
speed = max(50, light_val // 100)
# 2. Obsługa przycisków i buzzera
for i in range(4):
    if buttons[i].value() == 0: # Przycisk wciśnięty
        leds[i].value(1)
        buzzer.freq(500 * (i + 1))
        buzzer.duty_u16(1000) # Cichy dźwięk przycisku
    else:
        leds[i].value(0)
if all(b.value() == 1 for b in buttons):
    buzzer.duty_u16(0)

# 3. Odczyt potencjometru i sterowanie RGB (WS2812)
pot_val = pot.read_u16()
brightness = pot_val // 256 # Skalowanie do 0-255
# Ustawienie wszystkich diod na kolor zależny od potencjometru
for i in range(8):
    pixels[i] = (brightness, 0, 255 - brightness)
pixels.write()

utime.sleep_ms(10)
```

Wymiary

Wymiary modułu KAmodRPi Pico Prototyping Platform to 85 x 140 mm. Na płytce znajdują się 4 otwory montażowe o średnicy 3 mm ułatwiające stabilne zamocowanie platformy w obudowie lub na stanowisku warsztatowym.



Linki

- Model CAD (STEP)

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