KAMAMI

Kamod BluePill (PL)

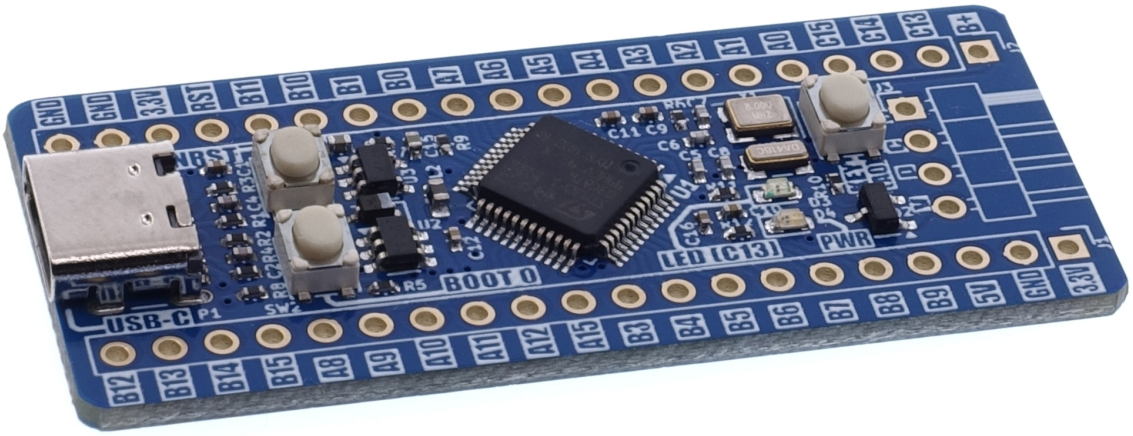
Rev. 20240916051059
Źródło: https://wiki.kamamilabs.com/index.php/Kamod_BluePill_(PL)

Spis treści

Opis

Kamod BluePill+
Płytka ewaluacyjna z mikrokontrolerem STM32F103C8T6, kompatybilna z BluePill

Płytka ewaluacyjna Kamod BluePill+ zawiera mikrokontroler STM32F103C8T6 oraz elementy niezbędne do jego uruchomienia i programowania. Płytka jest kompatybilna pod względem wyprowadzeń z projektem BluePill, ale ma szereg ulepszeń, m.in. nowy projekt PCB, złącze USB-C z zabezpieczeniem ESD, czy poprawiony obwód zasilania. Może być programowana z Arduino IDE, ponieważ w pamięci układu znajduje się odpowiedni bootloader.

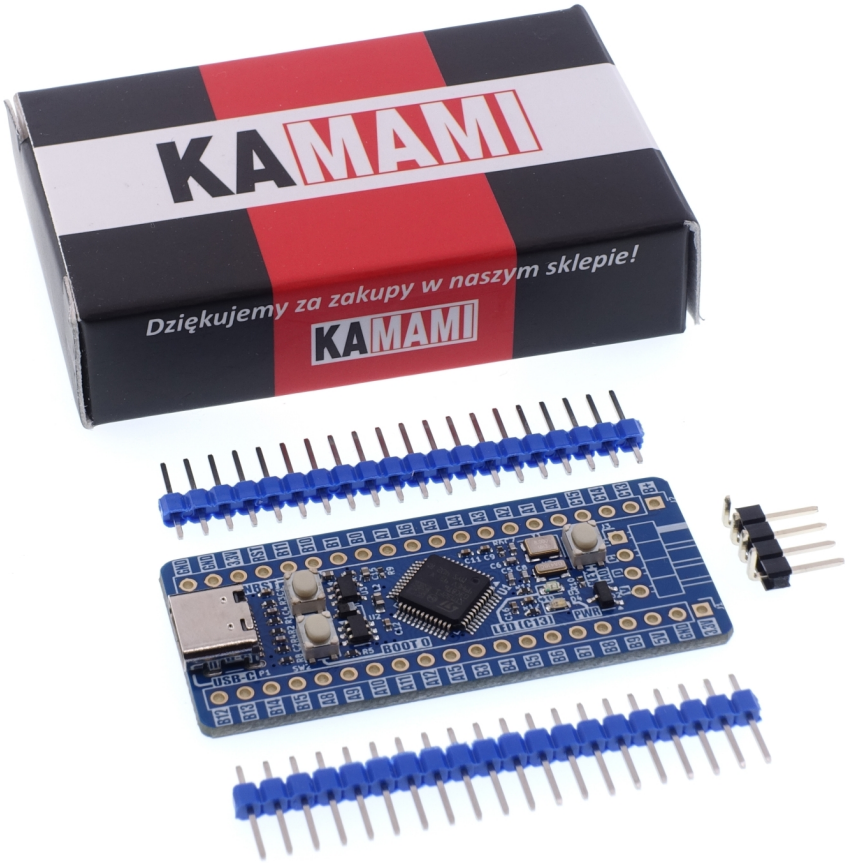


Podstawowe cechy i parametry

- mikrokontroler STM32F103C8T6: 128 kB Flash, 20 kB RAM, 72 MHz, 2 x ADC 12-bitowy, 4 timery, 2 x I2C, 2 x SPI, 3 x UART, CAN, USB, RTC
- złącze USB-C, które służy jako złącze zasilające, interfejs komunikacyjny USB oraz pozwala ma programowanie mikrokontrolera
- elementy filtrujące zakłócenia i przepięcia na liniach interfejsu USB
- 32 wyprowadzenia GPIO oraz linie zasilania 5 V i 3,3 V dostępne na standardowych złączach o rastrze 2,54 mm
- maksymalne obciążenie linii 5 V wynosi 500 mA, natomiast dla linii 3,3 V wynosi 200 mA
- precyzyjny rezonator taktujący mikrokontroler oraz niezależny rezonator dla modułu RTC
- możliwość dołączenia baterii podtrzymującej działanie modułu RTC
- złącze interfejsu programowania/debugowania SWD
- możliwość programowania poprzez STM32CubeIDE oraz Arduino IDE
- wymiary płytki: 53,5x23 mm, wysokość ok. 7 mm (bez wlutowanych szpilek goldpin)

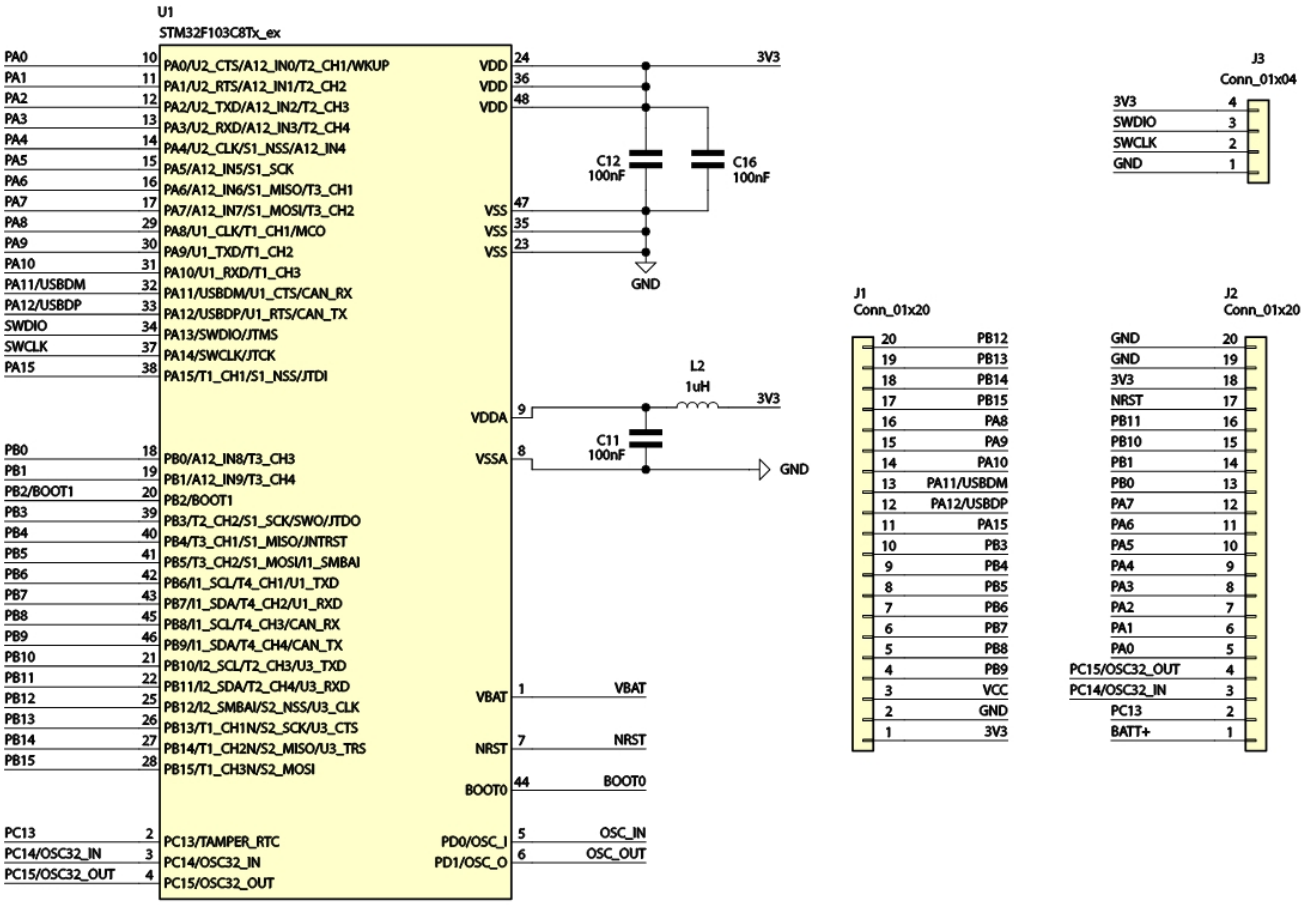
Wyposażenie standardowe

Kod	Opis
Kamod BluePill+	• Zmontowany i uruchomiony moduł, z wgranym bootloaderem • 2 x prosta listwa goldpin 20-pin raster 2,54 mm • 1 x kątowa listwa goldpin 4-pin raster 2,54 m

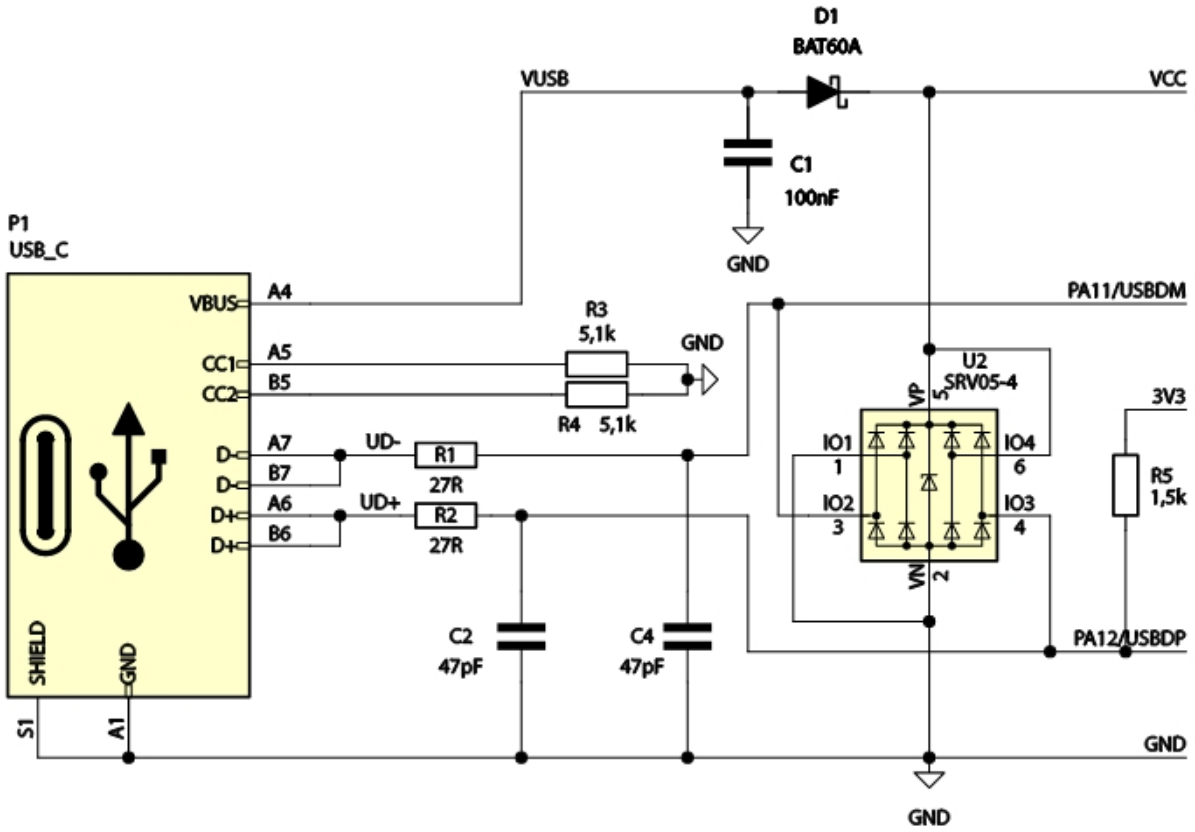


Schemat elektryczny

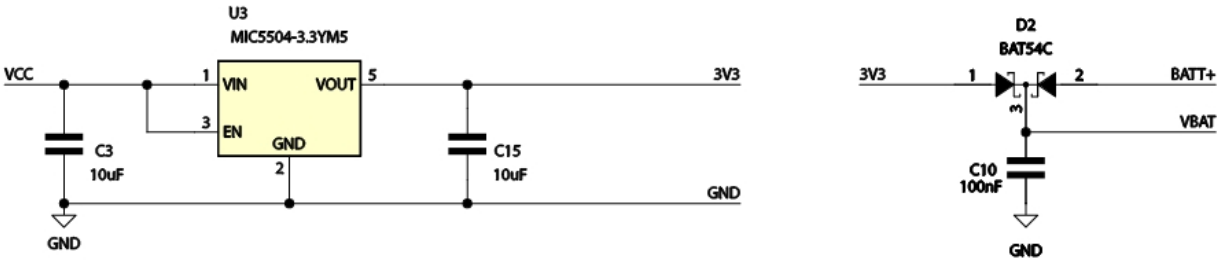
Mikrokontroler oraz złącza GPIO i SWD



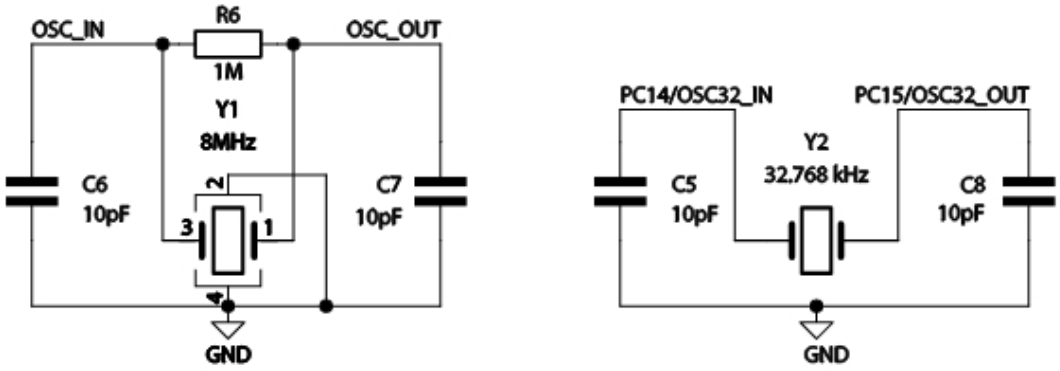
Interfejs USB



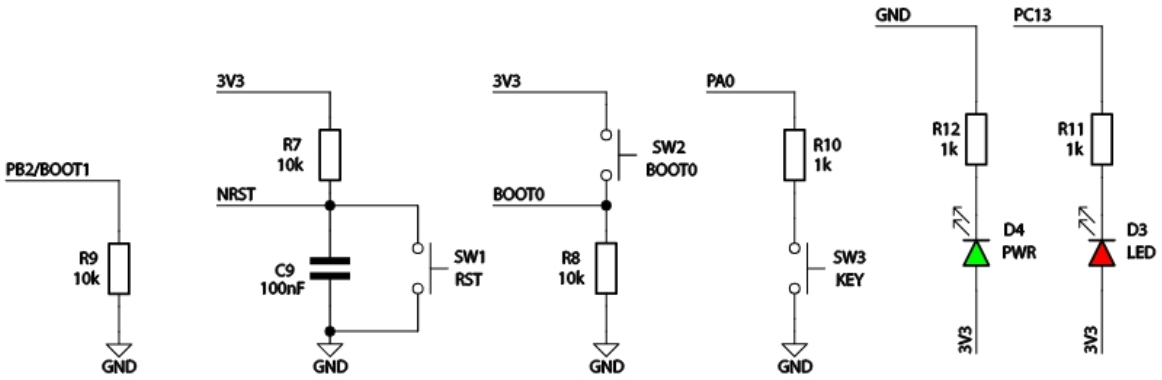
Obwód zasilania



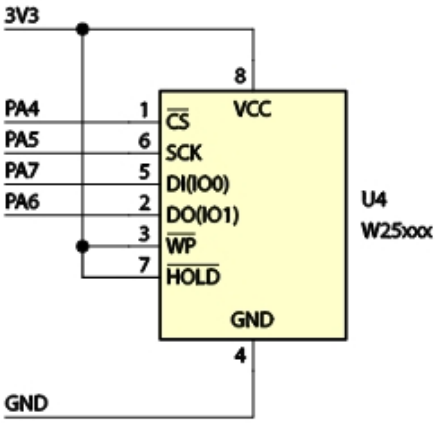
Taktowanie mikrokontrolera



Elementy dodatkowe



Zewnętrzna pamięć


3V3
8
VCC
PA4
1
CS
PA5
6
SCK
PA7
5
DI(IO0)
PA6
2
DO(IO1)
3
WP
7
HOLD
GND
4
GND
U4
W25xxx

Zasilanie

Złącze	Funkcja
USB-C J1, J2	• Doprowadza zasilanie 5 V do modułu • Pozwala doprowadzić zasilanie 5 V oraz udostępnia napięcie 3,3 V

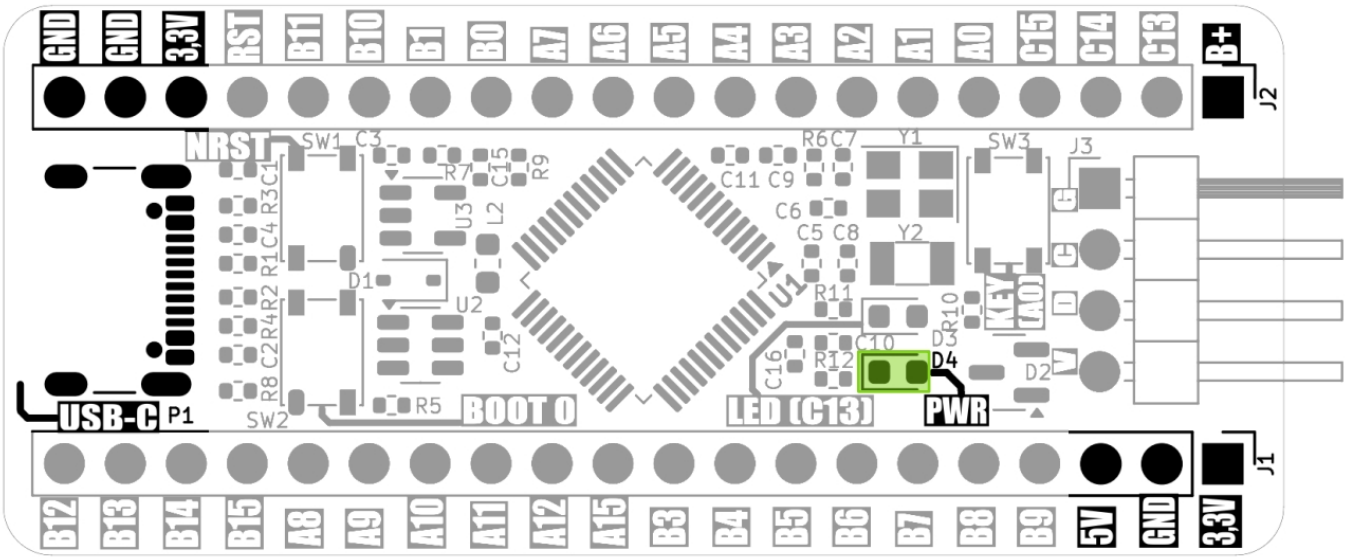
Płytka ewaluacyjna **Kamod BluePill+** może być zasilana na dwa sposoby:

- poprzez odpowiednie styki złączy J1 i J2,
- poprzez złącze USB-C.

Źródło zasilania o napięciu z zakresu 4,5...5,5 V i wydajności min. 100 mA należy dołączyć do styków oznaczonych ***5V*** (plus) oraz ***GND*** (minus) na złączach J1 i/lub J2. Wtedy na styku oznaczonym ***3,3V*** dostępne jest stabilizowane napięcie o wartości 3,3 V, które również zasila mikrokontroler. Obecność napięcia 3,3 V sygnalizuje świecenie diody LED oznaczonej ***PWR***.

Do złącza USB-C należy dołączyć standardowe źródło zasilania USB o wydajności min. 100 mA. Wtedy na styku 5V złącza J1 dostępne jest napięcie o wartości bliskiej 5 V (względem masy oznaczonej ***GND***). Niewielki spadek napięcia (ok 0,5 V) występuje na diodzie Schottky'ego, która umożliwia przepływ prądu w kierunku ze złącza USB-C do płytki, ale blokuje przepływ prądu w kierunku przeciwnym - do złącza USB-C. Dzięki temu można bezpiecznie dołączać zasilanie w różnych konfiguracjach - USB i/lub styki J1, J2.

Na złączu J2 znajduje się styk oznaczony ***B+***. Wraz z masą ***GND*** jest to wejście zasilania z baterii podtrzymującej działanie zegara RTC (zintegrowanego z mikrokontrolerem). Napięcie baterii podtrzymującej powinno zawierać się w przedziale 1,8...3,6 V. Dokładne informacje na temat działania modułu RTC można znaleźć w dokumentacji mikrokontrolera STM32F103C8T6.



Interfejs USB

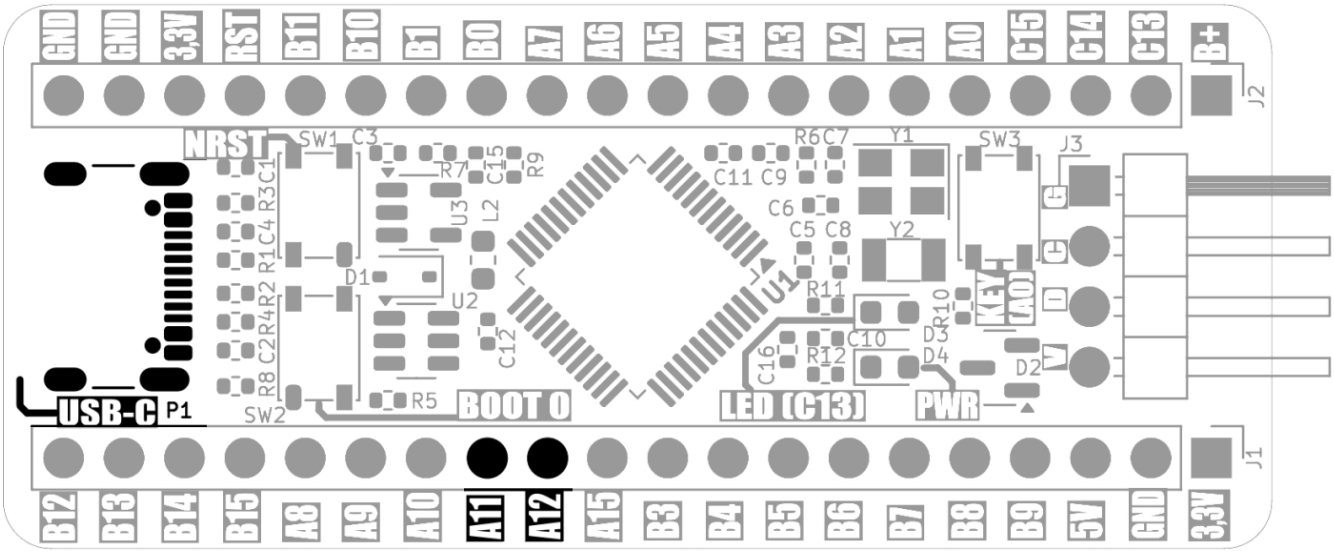
Złącze	Funkcja
USB-C	• Doprowadza zasilanie 5 V do modułu • Może realizować interfejs USB 2.0 FS Peripheral • Pozwala programować pamięć Flash mikrokontrolera

Złącze USB-C jest łatwym sposobem na doprowadzenie zasilania do płytki Kamod BluePill+. Ponadto mikrokontroler STM32F103C8T6 ma zintegrowany kontroler interfejsu USB 2.0 FS, który może służyć do komunikacji np. w trybie VCP (*Virtual COM Port*) lub CDC (*Communication Device Class*).

Płytka została skonfigurowana w taki sposób, że umożliwia działanie interfejsu w trybie Peripheral, nie jest przygotowana do pracy w trybie Host. Na płytce znajdują się również elementy filtrujące ewentualne zakłócenia i przepięcia na liniach interfejsu USB, które zapewniają jego stabilne działanie.

Dodatkową funkcjonalnością złącza USB-C jest możliwość programowania pamięci Flash mikrokontrolera. Wymagany jest do tego odpowiedni bootloader, który zostaje załadowany do mikrokontrolera modułu Kamod BluePill+ wraz z programem testowym.

Linie interfejsu USB (DP oraz DM) są jednocześnie portami GPIO PA12 oraz PA11 mikrokontrolera. Jeżeli korzystamy z USB to styki oznaczone *A12* i *A11* nie mogą pełnić żadnej innej funkcji – muszą pozostać niepodłączone.



Interfejs programowania/debugowania SWD

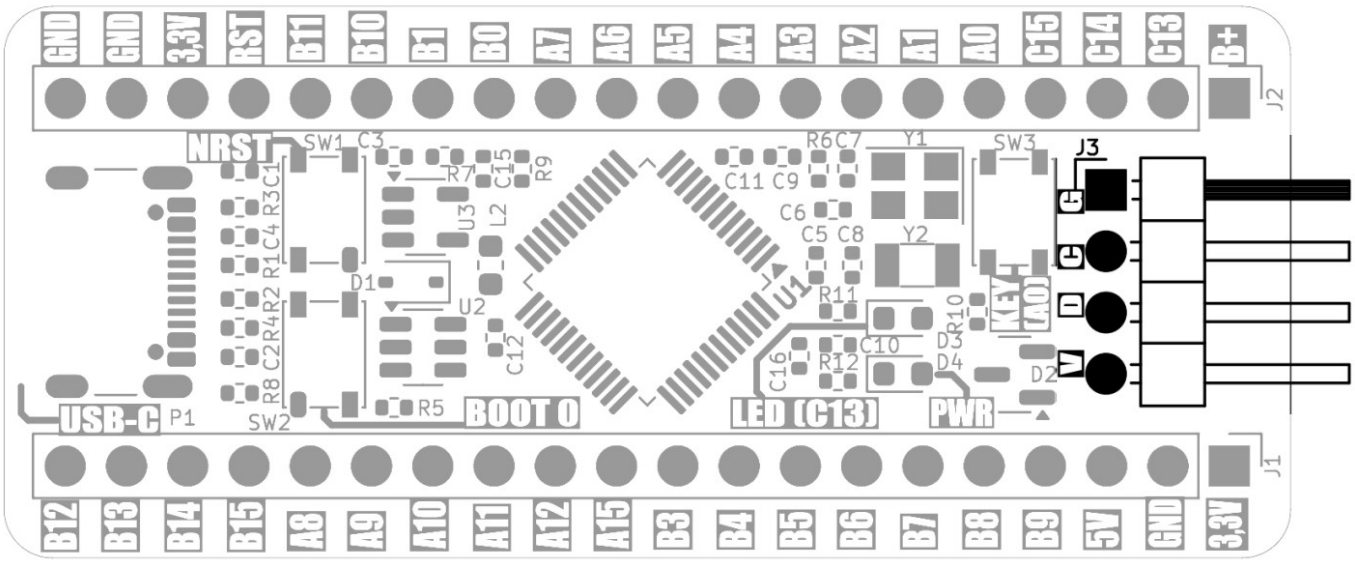
Złącze	Funkcja
J3	• Wyprowadzony interfejs SWD (*Single Wire Debug*) z sygnałami SWDIO oraz SWCLK

Interfejs SWD (*Single Wire Debug*) pozwala na programowanie pamięci Flash mikrokontrolera oraz śledzenie działania programu (debugowanie). Wymaga dołączenia zewnętrznego programatora/debuggera np. STLINK-V2 lub STLINK-V3MINIE.

Interfejs SWD został wyprowadzony na złącze szpilkowe J3. Sygnały zostały opisane w następujący sposób:

- G – masa układu,
- C – sygnał taktujący SWCLK,
- D – sygnał danych SWDIO,
- V – linia zasilania 3,3 V.

Sygnały należy połączyć z takimi samymi sygnałami na złączu programatora/debuggera. Czasami SWCLK jest oznaczony również jako TCK, natomiast SWDIO jest jednocześnie oznaczony jako TMS. Programator nie dostarcza zasilania do płytki Kamod BluePill+, zasilanie należy dołączyć do złącza USB-C lub styków J1/J2.



Programowanie mikrokontrolera

Komponent	Funkcja
SW1 – NRST	• Wymuszenie stanu wyzerowania mikrokontolera
SW2 – BOOT0	• Uruchomienie fabrycznego bootloadera mikrokontolera
J3 – SWD	• Interfejs programowania SWD (Single Wire Debug)
A9, A10 – UART	• Interfejs programowania UART (TXD, RXD)

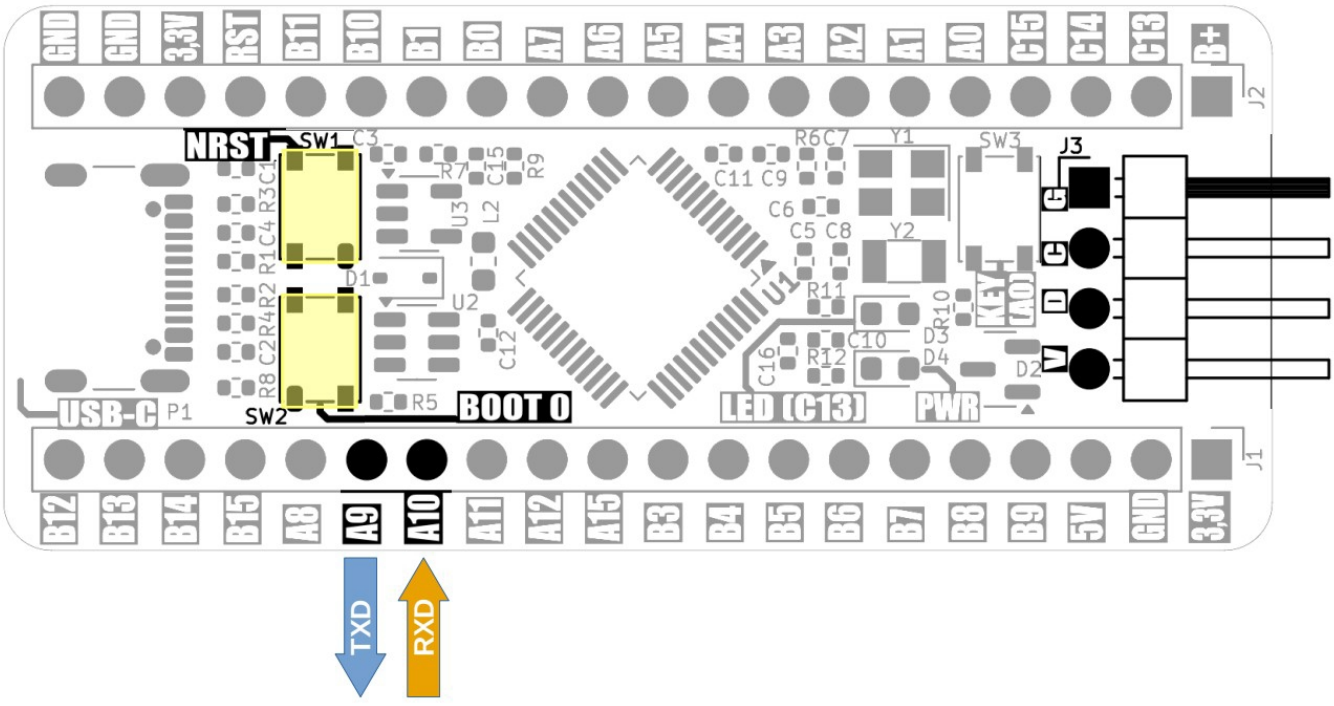
W specjalnym obszarze pamięci mikrokontrolera znajduje się oprogramowanie umożliwiające zaprogramowanie jego pamięci programu – jest to tzw. bootloader przygotowany przez producenta mikrokontrolera. Aby uruchomić bootloader należy wykonać następującą sekwencję czynności, przy podłączonym zasilaniu płytki:

1. Przycisnąć i trzymać przycisk NRTS
2. Przycisnąć i trzymać przycisk BOOT 0 trzymając wciśnięty przycisk NRST
3. Zwolnić przycisk NRTS, trzymając wciśnięty przycisk BOOT 0
4. Zwolnić przycisk BOOT 0

Uruchomienie bootloadera aktywuje dwa interfejsy programujące:

- SWD, który został opisany we wcześniejszym rozdziale i pozwala na programowanie mikrokontrolera z użyciem programatora/debuggera np. STLINK-V2 lub STLINK-V3MINIE.
- UART, który pozwala na użycie zwykłego konwertera USB-UART w roli programatora, np. KAmod USB-UART-mini. Należy wtedy podłączyć do styków A9 oraz A10 linie interfejsu UART, gdzie:
  - A9 – wyjście TXD mikrokontrolera należy podłączyć do wejścia RXD konwertera USB-UART,
  - A10 – wejście RXD mikrokontrolera należy podłączyć do wyjścia TXD konwertera USB-UART.

Teraz można programować mikrokontroler poprzez SWD lub UART za pomocą STM32CubeProgrammer, który jest dostępny na oficjalnej stronie STMicroelectronics: https://www.st.com/en/development-tools/stm32cubeprog.html



Płytka **Kamod BluePill+** umożliwia dodatkowo programowanie mikrokontrolera poprzez interfejs USB z użyciem Arduino IDE. Jest to możliwe dzięki temu, że wraz z programem testowym w pamięci programu umieszczany jest dodatkowy bootloader przeznaczony do komunikacji i programowania w środowisku Arduino. Należy wtedy:

- dodać następujący adres do menedżera płytek:http://dan.drown.org/stm32duino/package_STM32duino_index.json
- zainstalować pakiet STM32F1xx,
- wybrać płytkę Generic STM32F103C series,
- wybrać port komunikacyjny Maple Serial (COMxx).

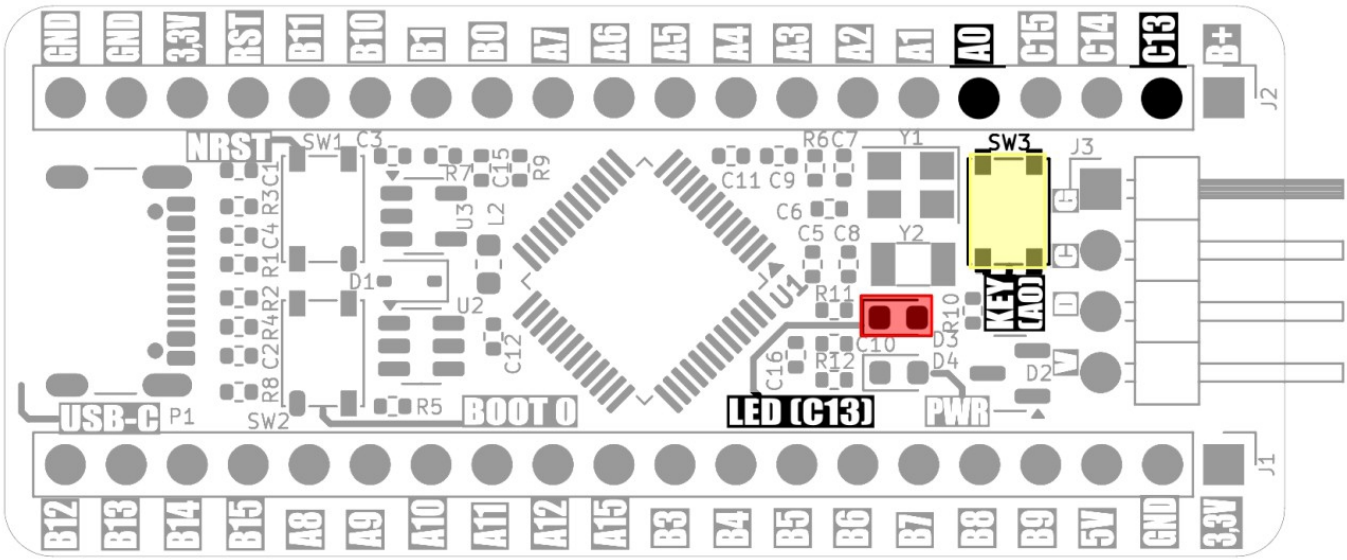
Dodatkowe elementy – dioda LED oraz przycisk

Komponent	Funkcja
D3 – LED (C13)	• Dioda LED dołączona do portu PC13, aktywna w stanie L
SW3 – KEY (A0)	• Mikroprzycisk dołączony do portu PA0, aktywny w stanie L

Płytka **Kamod BluePill+** zawiera elementy dodatkowe – diodę LED i mikroprzycisk, które mogą być użyte w docelowej aplikacji.

Dioda LED jest podłączona do portu PC13, jej zaświecenie następuje przy stanie niskim.

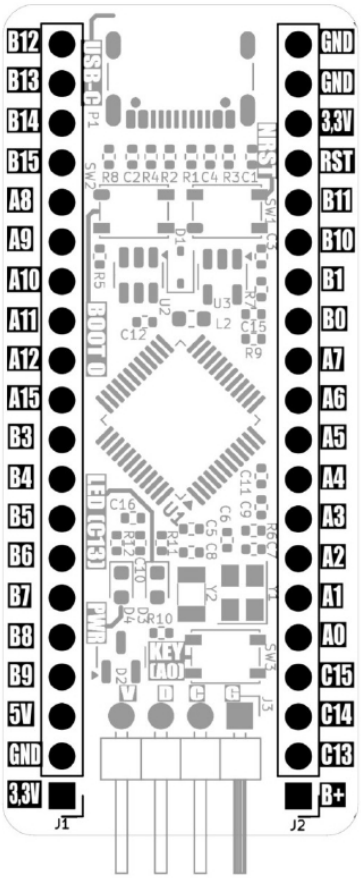
Mikroprzycisk jest dołączony do portu PA0, jego wciśnięcie wymusza stan niski na tym porcie. Przycisk jest dołączony poprzez rezystor 1k, zatem nie ma możliwości uszkodzenia portu PA0 skonfigurowanego jako wyjście.



Złącza GPIO

Złącze	Funkcja
J1, J2	• Złącza o 20 stykach z rastrem 2,54 mm, do których doprowadzone są porty GPIO mikrokontrolera oraz zasilanie 5 V i 3,3 V

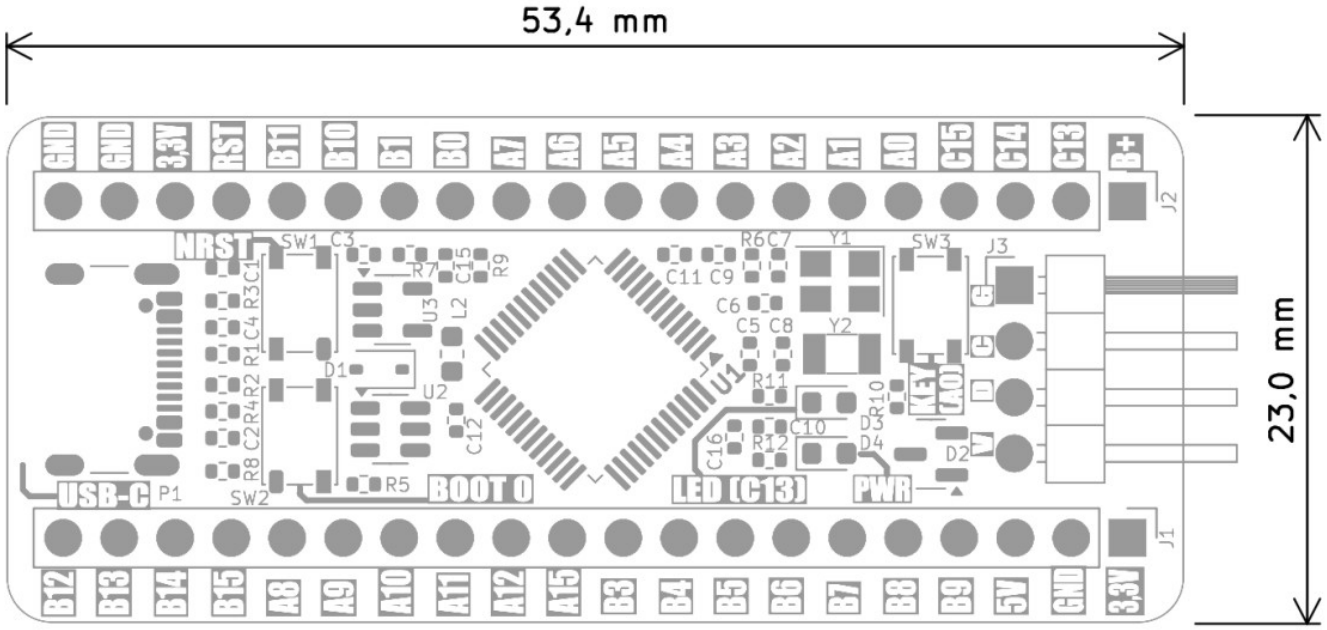
Złącza GPIO J1 i J2 zawierają po 20 szpilek, do których doprowadzone są linie zasilania 5 V, 3,3 V, masa GND oraz wyprowadzenia GPIO modułu mikrokontrolera. Dokładny opis wyprowadzeń oraz ich dodatkowe funkcje pokazuje rysunek i tabela poniżej:



J1		J2
PB12 (SPI2_NSS)		GND
PB13 (SPI2_SCK)		GND
PB14 (SPI2_MISO)		3,3 V
PB15 (SPI2_MOSI)		RESET
PA8 (TIMER1_CH1)		(UART3_RXD) PB11
PA9 (UART1_TXD)		(UART3_TXD) PB10
PA10 (UART1_RXD)		(ADC12_IN9) PB1
PA11 (USB_DM)		(ADC12_IN8) PB0
PA12 (USB_DP)		(ADC12_IN7) PA7
PA15 (SPI1_NSS)		(ADC12_IN6) PA6
PB3 (SPI1_SCK)		(ADC12_IN5) PA5
PB4 (SPI1_MISO)		(ADC12_IN4) PA4
PB5 (SPI1_MOSI)		(TIMER2_CH4) PA3
PB6 (I2C1_SCL)		(TIMER2_CH3) PA2
PB7 (I2C1_SDA)		(TIMER2_CH2) PA1
PB8 (CAN_RX)		(TIMER2_CH1) PA0
PB9 (CAN_TX)		(OSC_RTC_O) PC15
5 V		(OSC_RTC_I) PC14
GND		(LED_D3) PC13
3,3 V		RTC_BATTERY +

Wymiary

Wymiary płytki **Kamod BluePill+** to 53,5x23 mm, a wysokość wynosi ok. 7 mm (bez wlutowanych szpilek goldpin).



Program testowy

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

```
#include <RTClock.h>

#define LED_PIN   PC13

#define SW_PIN    PA0

int message_period = 0;

int config_state = 0;

int ports_state = 0;

tm_t tmt;


RTClock rtclock (RTCSEL_LSE); // initialise


void setup() {

 Serial.begin(115200);
 pinMode(LED_PIN, OUTPUT);
 pinMode(SW_PIN, INPUT_PULLUP);

}


// the loop function runs over and over again forever void loop() {

 //digitalWrite(LED_PIN, HIGH);
 if (digitalRead(SW_PIN) == HIGH){
   if ((ports_state&4) > 0){
     digitalWrite(LED_PIN, HIGH);
   } else {
     digitalWrite(LED_PIN, LOW);
   }

   if (config_state > 0){
     IO2input();
     config_state = 0;
   }

 } else {
   if ((ports_state&1) > 0){
     digitalWrite(LED_PIN, HIGH);
   } else {
     digitalWrite(LED_PIN, LOW);
   }

   if (config_state == 0){
     IO2output();
     config_state = 1;
   }

   if (config_state == 1){
     if ((ports_state&1) > 0){
```

```
        IO2test1();
      } else {
        IO2test2();
      }
    }
  }

  // check if plugged into a host
  if (message_period >= 11){
    message_period = 0;
    rtclock.getTime(tmt);
    Serial.print("KAmod BluePill+ RTC=");
    if (tmt.minute < 10) Serial.print("0");
    Serial.print(tmt.minute);
    Serial.print(":");
    if (tmt.second < 10) Serial.print("0");
    Serial.println(tmt.second);
  }

  delay(200);
  message_period++;
  ports_state++;

}

void IO2output(){

  pinMode(PA1, OUTPUT);
  pinMode(PA2, OUTPUT);
  pinMode(PA3, OUTPUT);
  pinMode(PA4, OUTPUT);
  pinMode(PA5, OUTPUT);
  pinMode(PA6, OUTPUT);
  pinMode(PA7, OUTPUT);
  pinMode(PA8, OUTPUT);
  pinMode(PA9, OUTPUT);
  pinMode(PA10, OUTPUT);
  pinMode(PA15, OUTPUT);

  pinMode(PB0, OUTPUT);
  pinMode(PB1, OUTPUT);
  pinMode(PB3, OUTPUT);
  pinMode(PB4, OUTPUT);
  pinMode(PB5, OUTPUT);
  pinMode(PB6, OUTPUT);
  pinMode(PB7, OUTPUT);
  pinMode(PB8, OUTPUT);
  pinMode(PB9, OUTPUT);
  pinMode(PB10, OUTPUT);
  pinMode(PB11, OUTPUT);
  pinMode(PB12, OUTPUT);
  pinMode(PB13, OUTPUT);
  pinMode(PB14, OUTPUT);
  pinMode(PB15, OUTPUT);

  //pinMode(PC13, OUTPUT);

}

void IO2input(){
```

```
 pinMode(PA1, INPUT);
 pinMode(PA2, INPUT);
 pinMode(PA3, INPUT);
 pinMode(PA4, INPUT);
 pinMode(PA5, INPUT);
 pinMode(PA6, INPUT);
 pinMode(PA7, INPUT);
 pinMode(PA8, INPUT);
 pinMode(PA9, INPUT);
 pinMode(PA10, INPUT);
 pinMode(PA15, INPUT);

 pinMode(PB0, INPUT);
 pinMode(PB1, INPUT);
 pinMode(PB3, INPUT);
 pinMode(PB4, INPUT);
 pinMode(PB5, INPUT);
 pinMode(PB6, INPUT);
 pinMode(PB7, INPUT);
 pinMode(PB8, INPUT);
 pinMode(PB9, INPUT);
 pinMode(PB10, INPUT);
 pinMode(PB11, INPUT);
 pinMode(PB12, INPUT);
 pinMode(PB13, INPUT);
 pinMode(PB14, INPUT);
 pinMode(PB15, INPUT);

 //pinMode(PC13, INPUT);

}

void IO2low(){

 digitalWrite(PA1, LOW);
 digitalWrite(PA2, LOW);
 digitalWrite(PA3, LOW);
 digitalWrite(PA4, LOW);
 digitalWrite(PA5, LOW);
 digitalWrite(PA6, LOW);
 digitalWrite(PA7, LOW);
 digitalWrite(PA8, LOW);
 digitalWrite(PA9, LOW);
 digitalWrite(PA10, LOW);
 digitalWrite(PA15, LOW);

 digitalWrite(PB0, LOW);
 digitalWrite(PB1, LOW);
 digitalWrite(PB3, LOW);
 digitalWrite(PB4, LOW);
 digitalWrite(PB5, LOW);
 digitalWrite(PB6, LOW);
 digitalWrite(PB7, LOW);
 digitalWrite(PB8, LOW);
 digitalWrite(PB9, LOW);
 digitalWrite(PB10, LOW);
 digitalWrite(PB11, LOW);
 digitalWrite(PB12, LOW);
 digitalWrite(PB13, LOW);
 digitalWrite(PB14, LOW);
 digitalWrite(PB15, LOW);
```

```
 //digitalWrite(PC13, LOW);

}

void IO2test1(){

 digitalWrite(PA1, HIGH);
   digitalWrite(PA2, LOW);
 digitalWrite(PA3, HIGH);
   digitalWrite(PA4, LOW);
 digitalWrite(PA5, HIGH);
   digitalWrite(PA6, LOW);
 digitalWrite(PA7, HIGH);
   digitalWrite(PA8, HIGH);
 digitalWrite(PA9, LOW);
   digitalWrite(PA10, HIGH);
 digitalWrite(PA15, HIGH);

 digitalWrite(PB0, LOW);
   digitalWrite(PB1, HIGH);
 digitalWrite(PB3, LOW);
   digitalWrite(PB4, HIGH);
 digitalWrite(PB5, LOW);
   digitalWrite(PB6, HIGH);
 digitalWrite(PB7, LOW);
   digitalWrite(PB8, HIGH);
 digitalWrite(PB9, LOW);
   digitalWrite(PB10, LOW);
 digitalWrite(PB11, HIGH);
   digitalWrite(PB12, HIGH);
 digitalWrite(PB13, LOW);
   digitalWrite(PB14, HIGH);
 digitalWrite(PB15, LOW);

 digitalWrite(PC13, HIGH);

}

void IO2test2(){

 digitalWrite(PA1, LOW);
   digitalWrite(PA2, HIGH);
 digitalWrite(PA3, LOW);
   digitalWrite(PA4, HIGH);
 digitalWrite(PA5, LOW);
   digitalWrite(PA6, HIGH);
 digitalWrite(PA7, LOW);
   digitalWrite(PA8, LOW);
 digitalWrite(PA9, HIGH);
   digitalWrite(PA10, LOW);
 digitalWrite(PA15, LOW);

 digitalWrite(PB0, HIGH);
   digitalWrite(PB1, LOW);
 digitalWrite(PB3, HIGH);
   digitalWrite(PB4, LOW);
 digitalWrite(PB5, HIGH);
   digitalWrite(PB6, LOW);
 digitalWrite(PB7, HIGH);
   digitalWrite(PB8, LOW);
 digitalWrite(PB9, HIGH);
```

```
  digitalWrite(PB10, HIGH);
 digitalWrite(PB11, LOW);
  digitalWrite(PB12, LOW);
 digitalWrite(PB13, HIGH);
  digitalWrite(PB14, LOW);
 digitalWrite(PB15, HIGH);

 digitalWrite(PC13, LOW);

}
```

Działanie programu polega na uruchomieniu portu szeregowego na bazie interfejsu USB oraz modułu zegara RTC i cyklicznym wysyłaniu informacji o czasie poprzez ten port szeregowy.

Działaniu programu towarzyszy powolne miganie diody LED D3. Gdy zostanie naciśnięty przycisk KEY (SW3) to na wszystkich portach naprzemiennie będzie zmieniał się stan wyjść (H-L, L-H) oraz dioda LED zacznie pulsować szybko.

Linki

- Karta katalogowa układu STM32F103C8T6
- Program testowy Arduino
- Projekt testowy STM32CUBE_IDE
- STLINK-V3MINIE - kompaktowy programator/debuger dla STM32
- STM32CubeProgrammer
- KAmod USB-UART-mini - Miniaturowy konwerter USB-UART

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