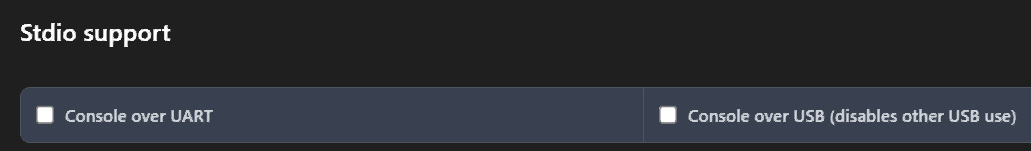
# Podstawy podstaw UART na Pico / BeamKit

Do wyboru są 2 urządzenia UART: 0 i 1, które można powiązać z różnymi pinami, ale umawiamy się że UART0 jest podłączony do GPIO0/1 a UART1 do GPIO4/5. Jeśli UART1 ma działać jako RS485, to trzeba zdefiniować i fizycznie podłączyć dodatkowy sygnał sterujący - u nas to jest GPIO3 połączone z DE/nRE. Jeśli nie wysyłasz akurat danych na magistralę 485 to ten sygnał musi mieć wartość 0 (jesteśmy wtedy w trybie odbioru i nie blokujemy magistrali).

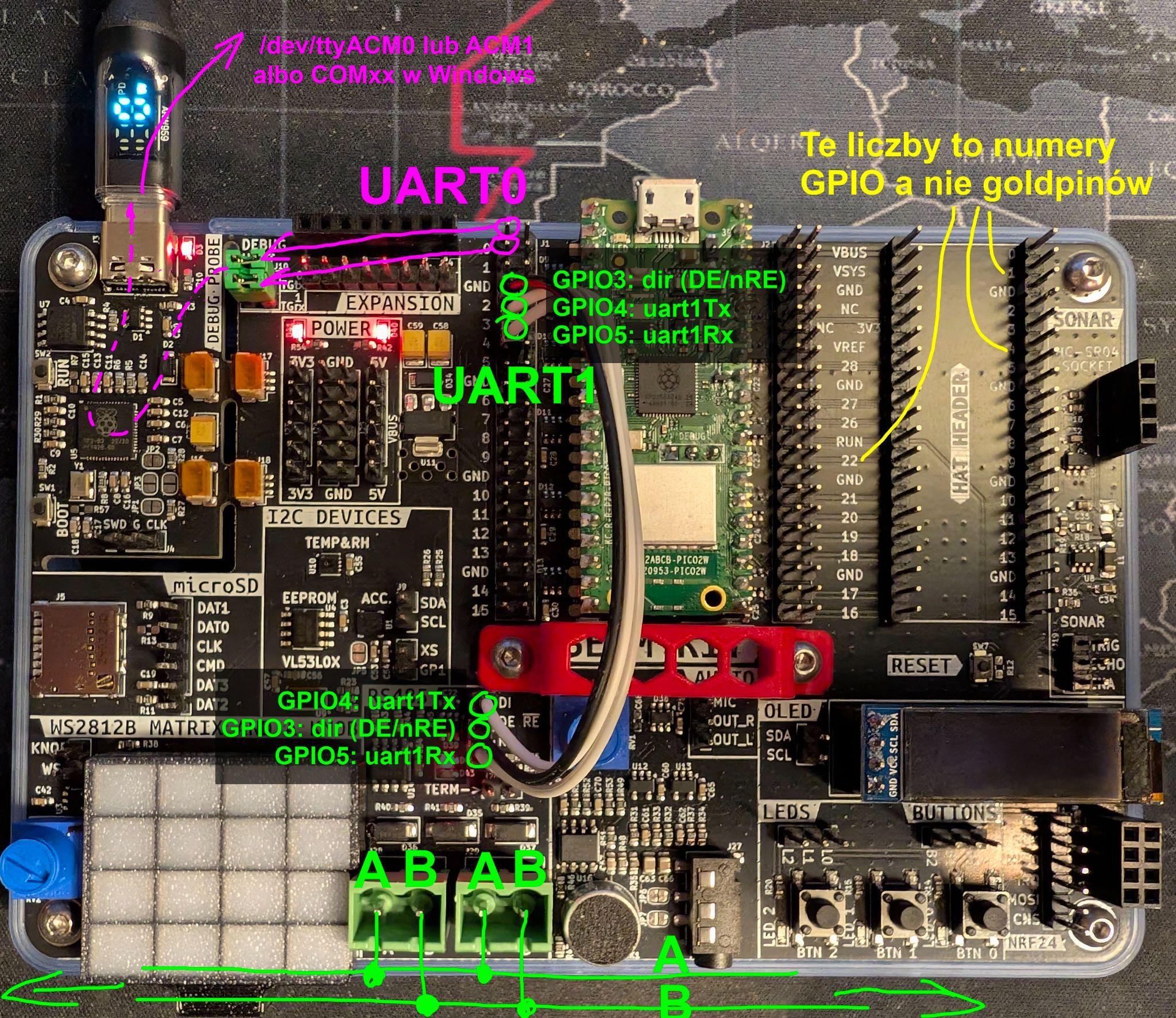
W trakcie zakładania projektu można powiązać stdio z UART0, z USB **albo z niczym**.



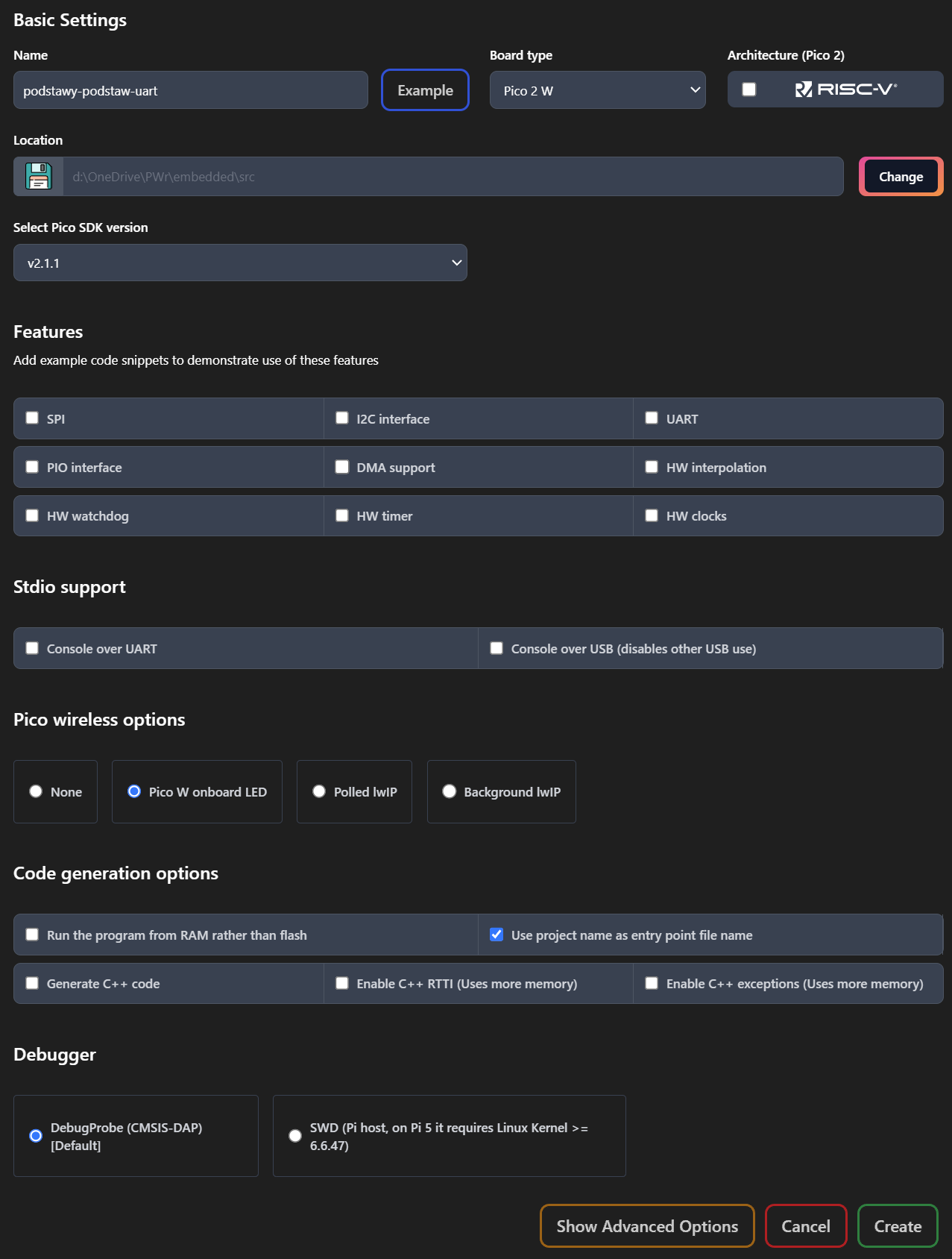
**Zalecenie: nie zaznaczaj Console over UART, pozwoli ci to na wygodną realizację zadania “serwer terminala znakowego z dekodowaniem komend” nawet bez dostępu do RS485.**

Magenta (wyjaśnienie dla facetów: różowy) to połączenia UART0[0,1] z konwerterem USB-serial w debugerze, nic nie musisz robić, te zworki są domyślnie zamontowane.

Limonkowy (wyjaśnienie dla facetów: żarówiasty zielony) to połączenia UART1[4,5+3dir] z konwerterem MAX485, które musisz wykonać 3 dodatkowymi przewodami, chyba że nie chcesz używać RS485 w projekcie.



**Jak założyć projekt**



#include <stdio.h>

#include <stdlib.h>

#include "pico/stdlib.h"

#include "pico/cyw43\_arch.h"

#include "hardware/uart.h"

void blink(int32\_t n) {

for(int32\_t i = 0; i<n; ++i) {

cyw43\_arch\_gpio\_put(CYW43\_WL\_GPIO\_LED\_PIN, 1);

sleep\_ms(100);

cyw43\_arch\_gpio\_put(CYW43\_WL\_GPIO\_LED\_PIN, 0);

sleep\_ms(500);

}

}

#define TERM\_UART\_ID uart0

#define TERM\_BAUD\_RATE 115200

#define TERM\_TX\_PIN 0

#define TERM\_RX\_PIN 1

void term\_send\_str(const char \*src) {

size\_t len = strlen(src);

uart\_write\_blocking(TERM\_UART\_ID, (const uint8\_t \*)src, len);

uart\_tx\_wait\_blocking(TERM\_UART\_ID); //wait for fifo

}

#define N\_TERM\_BUF 32 // Intentionally that short

char term\_buf[N\_TERM\_BUF];

uint32\_t term\_idx = 0;

uart\_inst\_t \*term\_uart;

void term\_process\_command(void);

// RX interrupt handler

void on\_uart\_rx() {

while (uart\_is\_readable(term\_uart)) {

uint8\_t ch = uart\_getc(term\_uart);

if(ch == '\r') {

term\_process\_command();

term\_idx = 0; // Prepare for another command reception

term\_buf[term\_idx] = '\0';

} else {

if(term\_idx<N\_TERM\_BUF-1) {

term\_buf[term\_idx++] = ch;

term\_buf[term\_idx] = '\0';

} else {

// buffer is full

}

}

}

}

// Set up TERM UART

void term\_init(uart\_inst\_t \*uart, uint baudrate, uint tx\_pin, uint rx\_pin) {

term\_uart = uart;

uart\_init(uart, baudrate);

gpio\_set\_function(tx\_pin, GPIO\_FUNC\_UART);

gpio\_set\_function(rx\_pin, GPIO\_FUNC\_UART);

uart\_set\_fifo\_enabled(uart, false); // Turn off FIFO's - we want to do this character by character

int UARTx\_IRQ = uart == uart0 ? UART0\_IRQ : UART1\_IRQ; // Select interrupt for the UART we are using

irq\_set\_exclusive\_handler(UARTx\_IRQ, on\_uart\_rx); // Set up and enable the interrupt handlers

irq\_set\_enabled(UARTx\_IRQ, true);

uart\_set\_irq\_enables(uart, true, false); // Now enable the UART to send interrupts - RX only

}

void term\_process\_command() {

char \*sp = strchr(term\_buf, ' ');

if (sp) {

\*sp++ = '\0';

int arg = atoi(sp); // add validation

term\_send\_str("\r\ncommand: >>> ");

term\_send\_str(term\_buf);

term\_send\_str(" <<<\r\nargs: >>> ");

term\_send\_str(sp);

term\_send\_str(" <<<\r\n");

if(0 == strcmp(term\_buf, "blink")) blink(arg);

}

}

int main() {

stdio\_init\_all();

if (cyw43\_arch\_init()) {

printf("Wi-Fi init failed\n");

return -1;

}

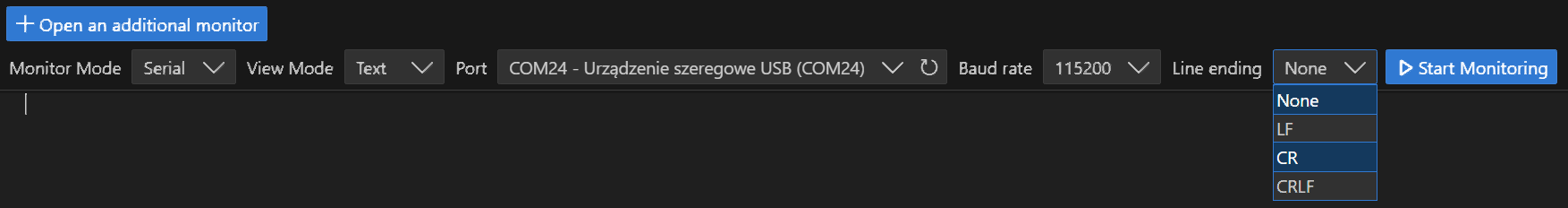
term\_init(TERM\_UART\_ID, TERM\_BAUD\_RATE, TERM\_TX\_PIN, TERM\_RX\_PIN);

while (true) {

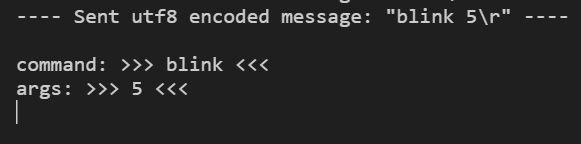
sleep\_ms(1000);

}

}



Co znaczy CR? jest o tym slajd na wykładzie, chodzi o \r lub 13 lub 0x0D co kto woli.



Nie uwierzycie, ale LED mrugnął 5 razy.

That's all folks!