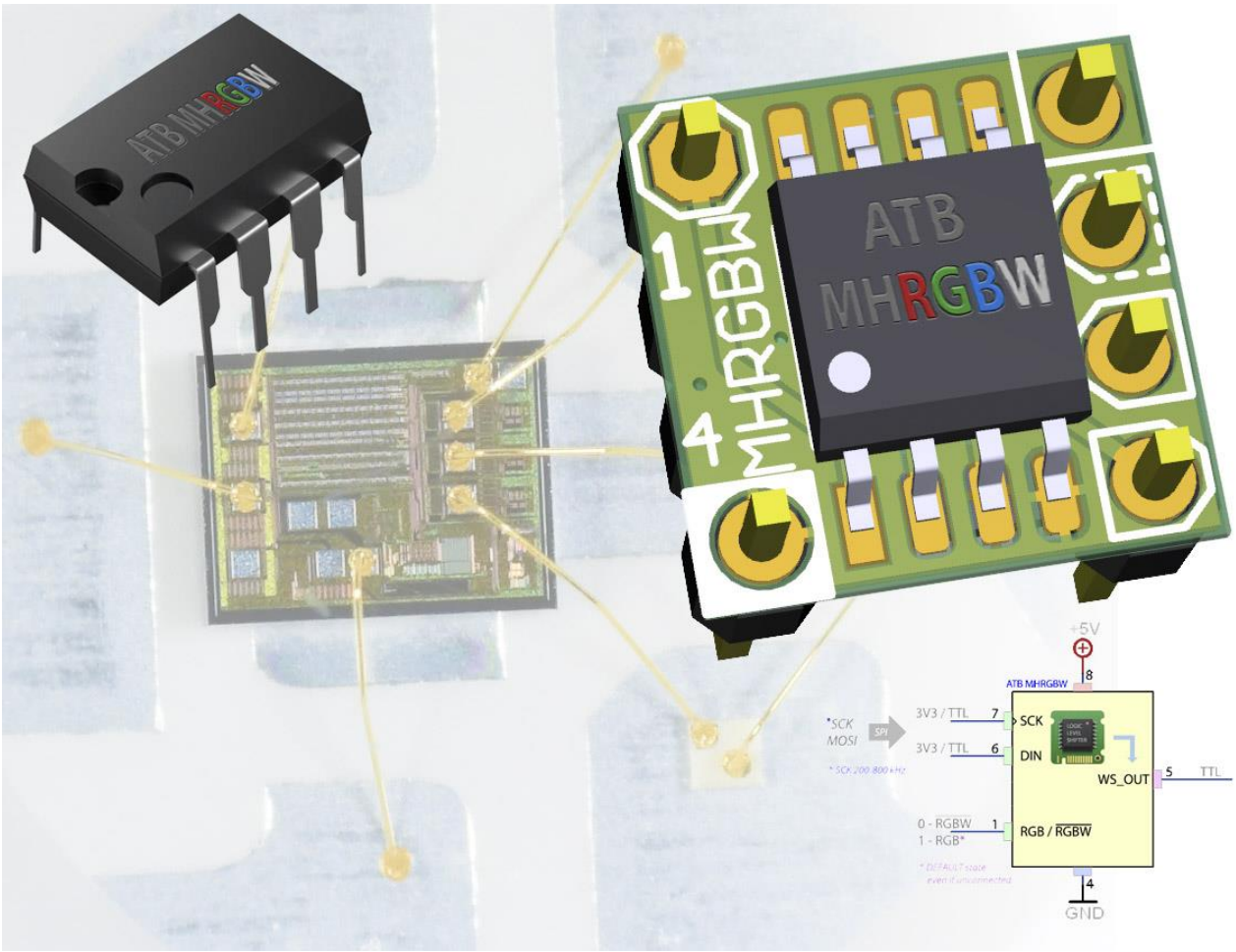
2023

PDF wersja 1.5

ATNEL

Mirosław Kardaś



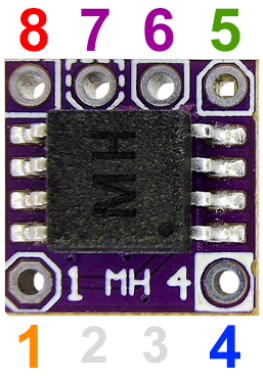
[INSTRUKCJA – MAGIC HERCULES]

Podstawowe informacje na temat Konwertera SPI do Magic LED. Opis wyprowadzeń, komunikacji i sposobów połączeń. Zasilanie.

Spis treści

Moduł Magic Hercules – wyprowadzenia / numeracja

Opis wyprowadzeń na podstawie zdjęcia fizycznego modułu Magic HERCULES. Widok od góry:



8 - Zasilanie +5 V

4 - Zasilanie GND

SPI – sygnały wejściowe

7 - SCK

6 - MOSI

5 - WS OUT – sygnał wyjściowy

1 - RGB/RGBW

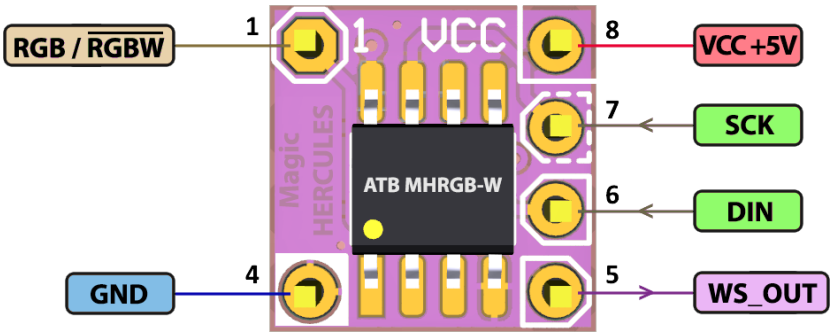
SCK MOSI WS OUT

Piny o numerach **2** i **3** nie zostały wyprowadzone na zewnątrz modułu MH. Lutując goldpiny należy wyjąć dwa środkowe piny, przylutować tylko dwa skrajne oznaczone numerami **1** i **4**.

Uwaga! W związku z tym, że na płytce PCB nie było miejsca dla wyraźnego opisu oznaczeń wprowadzono nowatorską metodę prostych symboli dookoła pinów sygnałowych, dzięki którym nawet bez zaglądania do tej dokumentacji widać który to jest sygnał wejściowy SCK, który wejściowy MOSI a który wyjściowy WS OUT.

SCK MOSI WS OUT

Poglądowy opis wyprowadzeń na modelu 3D. Widok od góry:



W celu zapewnienia poprawnej pracy moduł musi być bezwzględnie zasilany napięciem od minimum +4,7 V do maksimum +5,5 V. Moduł nigdy nie przechodzi do stanu uśpienia, w trakcie pracy, w zależności od transmisji SPI pobiera maksymalnie kilkanaście mA max.

Co to jest i do czego służy moduł Magic HERCULES

Moduł MH upraszcza programiście dowolnej rodziny mikrokontrolerów, do maksimum realizację programowego sterowania diodami adresowalnymi. Współpracuje z każdym rodzajem mikrokontrolera począwszy od 8051 i podobnych, poprzez rodziny **PIC8**, **PIC16**, **PIC32**. Mikrokontrolery 8-bitowe **AVR8** a także serie **ATxmega** w tym także najnowsze powstające modele **AVR8** i **AVR32**. Moduł jest polecany zdecydowanie dla mikrokontrolerów z serii **ARM/STM** ale także **ESP8266** oraz **ESP32** i wielu innych, które muszą być zasilane napięciem +3,3V. Dzięki modułowi MH nie trzeba korzystać z zewnętrznych układów scalonych do translacji poziomów napięć z +3,3V na TTL do diod LED. Moduł MH znacznie upraszcza wszelkie procedury związane z niskopoziomowym sterowaniem diod adresowalnych typu WS2812(B) i podobnych, szczególnie dla mikrokontrolerów 32 bitowych, dzięki czemu mogą one zużywać znacznie mniej własnych zasobów sprzętowych w porównaniu do popularnych i dostępnych w internecie rozwiązań pracujących bez udziału modułu MH. Nasz moduł daje większą swobodę programiście na skalowanie własnych projektów szczególnie gdy mowa o zastosowaniu systemów czasu rzeczywistego typu RTOS. Moduł umożliwia idealną współpracę również z takimi platformami jak **Raspbery PI** i podobne, które posiadają standardowo wyprowadzone magistrale SPI. W tych sytuacjach moduł MH jest wręcz idealnym i wymarzonym rozwiązaniem dla programisty używającego dowolnego języka programowania.

Moduł MH służy do bezpośredniej konwersji standardowego sygnału **SPI** *(tylko dwa sygnały **MOSI** i **SCK**)* na protokół do sterowania diodami adresowalnymi różnego rodzaju za pomocą jednej linii sygnałowej **WS OUT**.

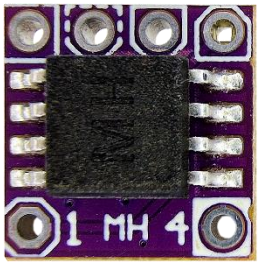
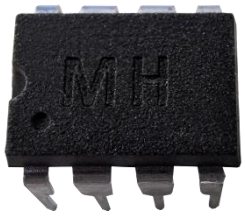
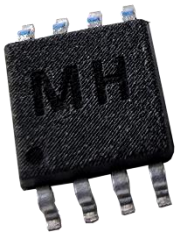
Częstotliwość sygnału wyjściowego do diod adresowalnych zależy bezpośrednio od częstotliwości sygnału SCK magistrali SPI. Pozwala to na diametralnie szeroki zakres sterowania w zależności od potrzeb programisty od ok **200-300 kHz** do maksymalnej prędkości równej **800 kHz** zgodnie ze standardem pracy diod adresowalnych. Dzięki takiemu podejściu można w dowolnych mikrokontrolerach wykorzystywać z pełnym powodzeniem nawet programową implementację SPI. Dzięki temu bez problemu można sterować nawet długimi łańcuchami LED z tak małych mikrokontrolerów 8-bitowych jak **ATtiny** i nie będzie to stanowiło dla programisty żadnego problemu. Nie będzie wymagane zastosowanie skomplikowanych wstawek asemblerowych. Oprogramowanie dla dowolnych rodzin mikrokontrolerów czy modułów w celu sterowania diodami adresowalnymi za pomocą modułu MH może być pisane począwszy od najbardziej popularnego **języka C**, poprzez **Arduino**, **Bascom**, **Python** *(dla Raspberry PI)* i innych.

Pełni on również bardzo istotną rolę związaną z konwersją poziomów napięć pomiędzy mikrokontrolerem zasilanym zapięciem +3,3 V a wyjściową linią sygnałową **WS OUT**, która bezwzględnie musi pracować w standardzie TTL. Wejścia sygnałowe **MOSI** i **SCK** tolerują zarówno stany **3V3** jak i **TTL**.

Ostatnią ważną funkcjonalnością modułu MH jest **możliwość pracy w trybie testera dla adresowalnych diod LED,** wyświetlaczy zbudowanych w oparciu o takie diody.

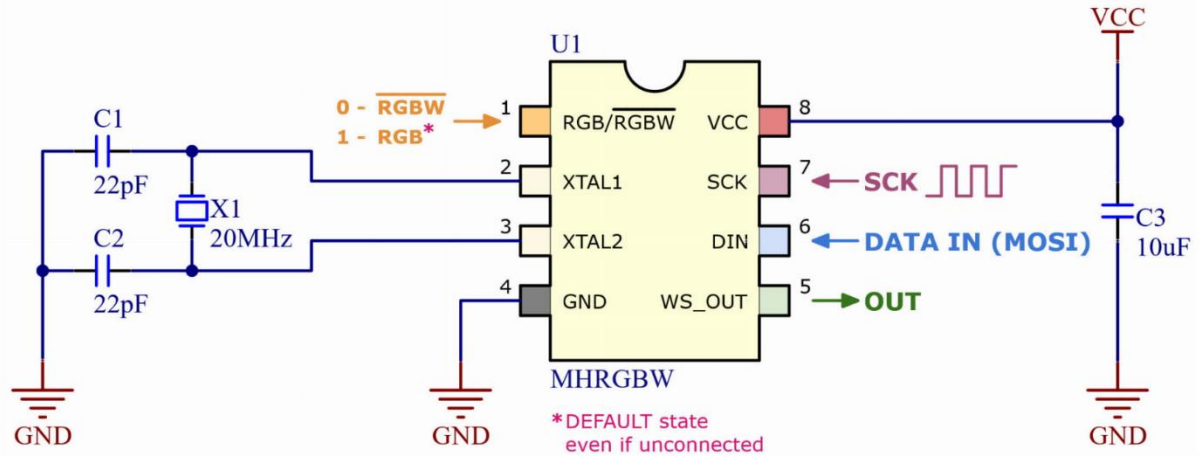
Rodzaje i obudowy MH

Magic Hercules można zakupić w trzech różnych postaciach:

Jako gotowy moduł	Układ scalony DIP8	Układ scalony SMD SO8
		

Gotowy moduł MH posiada wszystkie niezbędne elementy do pracy, przylutowane od spodu płytki PCB. Jest to subminiaturowy rezonator kwarcowy z wbudowanymi kondensatorami, oraz ważny kondensator 10 µF do filtracji zasilania. Nie są zatem wymagane żadne zewnętrzne elementy do jego prawidłowego uruchomienia. Gotowy moduł pozwala w możliwie najszybszy i najprostszy sposób zapoznać się z jego działaniem, bez konieczności lutowania zewnętrznych elementów, poza samymi goldpinami. Warto go zakupić na sam początek do testów.

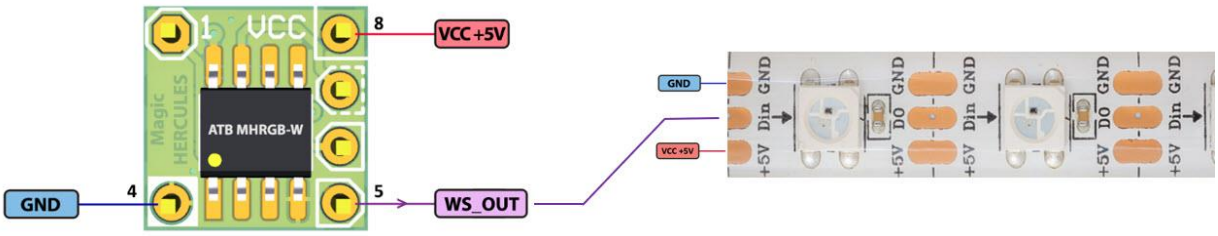
W przypadku układów **DIP8** i **SMD SO8** należy zastosować się do schematu połączeń poniżej aby prawidłowo uruchomić Magic Herculesa. Wymagany jest zewnętrzny rezonator kwarcowy o częstotliwości **20 MHz**, dwa kondensatory przy kwarcu, każdy o pojemności od 15 pF do 22 pF max. Do tego należy bezwzględnie dołożyć co najmniej jeden kondensator filtrujący jak najbliżej nóżek zasilania 4 i 8.



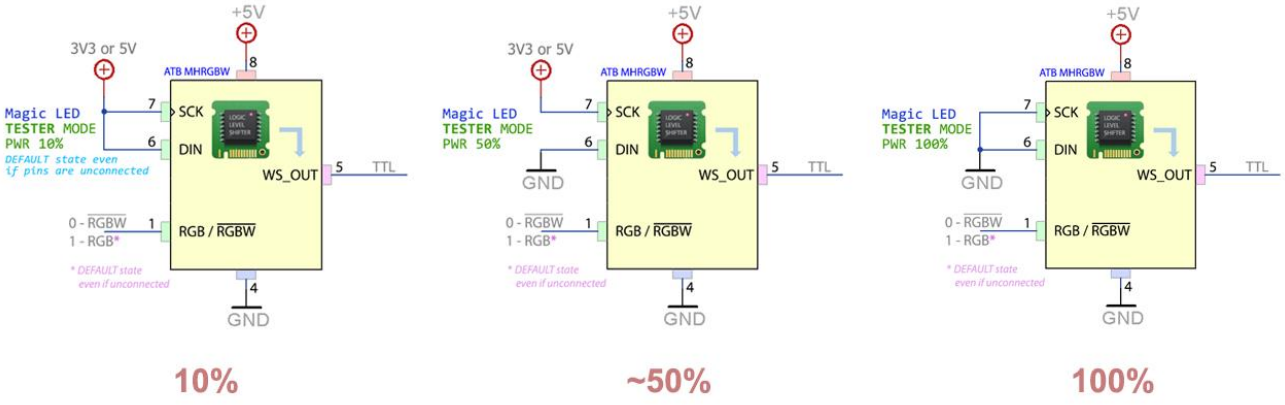
.

Moduł MH w trybie testera adresowalnych diod LED?

Do tego celu najlepiej nadaje się gotowy moduł MH, ponieważ wystarczy podłączyć do niego zasilanie +5V, GND i jego sygnał wejściowy do np. taśmy z adresowalnymi diodami LED lub wyświetlacza graficznego zbudowanego w oparciu o te diody. Jeśli wykorzystywane są diody zasilanie napięciem +5V to w tym przypadku samo podłączenie testera jest uproszczone do granic możliwości.



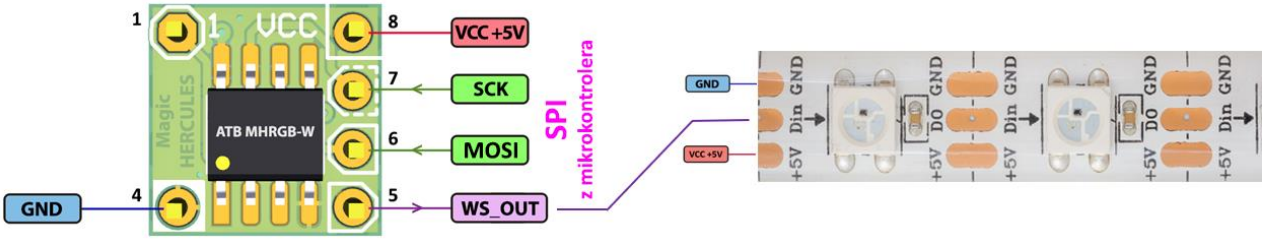
Wystarczy podłączyć jedno zasilanie +5V i GND do diod i modułu MH, a także podłączyć linię sygnału wyjściowego WS_OUT do wejścia DIN na taśmie led. Po włączeniu zasilania moduł MH bez sygnału SPI na wejściu zamienia się w tester i rozpoczyna wysyłanie sekwencji kolorowych wzorów na taśmę lub wyświetlacz LED. Domyślnie jeśli pin nr.1 (RGB/RGBW) jest do niczego nie podłączony to test rozpoczyna się w trybie RGB. Jeśli podłączone są diody RGBW to wtedy pin nr.1 należy zewrzeć do masy aby rozpoczęło się wysyłanie sekwencji kolorów testowych w trybie RGBW. Moduł MH może pracować w trzech trybach testu, generując sekwencje kolorowe z różnym natężeniem (mocą świecenia diod). Domyślnie gdy piny 6 i 7 są do niczego niepodłączone, wtedy emitowane są testy kolorów z najmniejszą mocą PWM 10%. W przypadku podłączania innych stanów do pinów nr. 6 i 7 można spowodować wysyłanie sekwencji kolorów z mocą PWM 50% oraz 100%.



Na rysunku powyżej przedstawiono zależność mocy świecenia diod w zależności od tego jak podłączone są piny 6 i 7 modułu MH, gdy nie jest do niego nadawany żaden sygnał po magistrali SPI.

Sterowanie adresowalnych diod LED z mikrokontrolera.

Wystarczy podłączyć piny nr 6 i 7 do magistrali SPI mikrokontrolera i rozpocząć transmisję danych do diod adresowalnych aby moduł MH automatycznie wyłączył swój tryb „testera” i przeszedł w stan konwertera SPI do NZR czyli do sygnału do sterownia diodami.



W dowolnym mikrokontrolerze można wykorzystać do tego celu zarówno wbudowany sprzętowy moduł SPI i jego linie SCK, **MOSI** (**DIN**) albo programową implementację obsługi SPI zrealizowaną na dowolnych pinach mikrokontrolera. W trakcie gdy moduł MH otrzymuje dane przez SPI, nie ma znaczenia jak podłączony jest pin nr.1 RGB/RGBW *(ma on znaczenie tylko gdy MH pracuje w roli testera)*.

Najprostsza procedura testowa na przykładzie AVR - soft SPI.

Poniżej przedstawiono najprostszą implementację programowej obsługi SPI, dzięki czemu kod ten można wykorzystać do zmiany kolorów na adresowalnych diodach LED podłączonych do modułu MH.

Uwaga! prezentowany kod można uruchomić na absolutnie dowolnym mikrokontrolerze AVR8 zarówno ATmega jak i małym ATtiny.

Definicje preprocesora dla pinów soft SPI w pliku **magic_hercules_drv.h**

```
/*
 * magic_hercules_drv.h
 *
 *  Created on: 3 mar 2023
 *      Author: Mirosław Kardaś
 */
#ifndef MAGIC_HERCULES_DRV_H_
#define MAGIC_HERCULES_DRV_H_

#define     LED_CNT           16          // liczba diod na taśmie/ekranie

/*````````````` MOSI pin ````````````````````````````````````````````````*/
#define MOSI            PB4 // dowolny pin na dowolnym porcie!
#define MOSI_DIR  DDRB
#define MOSI_PORT       PORTB

/*````````````` SCK pin ````````````````````````````````````````````````*/
#define SCK             PB3 // dowolny pin na dowolnym porcie!
#define SCK_DIR   DDRB
#define SCK_PORT  PORTB


/***** najpopularniejszy układ kolorów GRB *************/
typedef struct {
      uint8_t g;
      uint8_t r;
      uint8_t b;
} TRGB;


/*.......... preprocesor definitions ..............................*/
#define SCK_LO    SCK_PORT &= ~(1<<SCK)
#define SCK_HI    SCK_PORT |= (1<<SCK)

#define MOSI_LO   MOSI_PORT &= ~(1<<MOSI)
#define MOSI_HI   MOSI_PORT |= (1<<MOSI)


/* nagłówki funkcji */
extern void mk_hercules_init( uint32_t color );
extern void send_byte( uint8_t byte );


#endif /* MAGIC_HERCULES_DRV_H_ */
```

Najważniejsze funkcje w pliku **magic_hercules_drv.c**

```
/*
 * magic_hercules_drv.c
 *
 *  Created on: 3 mar 2023
 *      Author: Mirosław Kardaś
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stddef.h>

#include "magic_hercules_drv.h"

void mk_hercules_init( uint32_t color ) {
	MOSI_DIR |= (1 << MOSI);
	SCK_DIR  |= (1 << SCK);
	SCK_LO;     // after reset/restart SCK have to be in LOW state
	MOSI_HI;    // after reset/restart MOSI have to be in HIGH state
	delay_ms(10);     // oczekiwanie na ustalenie się poziomów na pinach
	TRGB rstRGB;
	rstRGB.r = color >> 16;
	rstRGB.g = color >> 8;
	rstRGB.b = color >> 0;
	size_t rstlen;
	uint8_t *wsk;
	uint8_t i;
/*-------- zalecana operacja po włączeniu zasilania uC i MH ------------*/
#ifdef MCUCSR
	#define MK_MCUSR  MCUCSR
#else
	#define MK_MCUSR  MCUSR
#endif
	uint8_t pwr_rst = MK_MCUSR & (1<<EXTRF);
	if( !pwr_rst ) {
		rstlen = 1024*3;
		while( rstlen-- ) send_byte(0);
		_delay_ms(10);
	}
/*---------------------------------------------------------------------*/
	rstlen = LED_CNT;
	while( rstlen-- ) {
		i = sizeof(TRGB);
		wsk = (uint8_t*)&rstRGB;
		while( i-- ) send_byte( *(wsk++) );
	}
}

void send_byte(uint8_t byte) {
	static uint8_t cnt;
	cnt = 0x80;
	while( cnt ) {
		if( byte & cnt ) MOSI_HI;
		else MOSI_LO;
		SCK_HI;
		cnt >>= 1;
		SCK_LO;
	}
	MOSI_HI;
}
```

Główny plik programu **main.c**

```
/*
 * main.c
 *
 * Created on: 3 mar 2023
 * Author: Mirosław Kardaś
 *
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stddef.h>

#include "magic_hercules_drv.h"

#define MY_COLOR 0x050005 // ustaw własny kolor RGB

int main(void) {

	/* dla pierwszych testów zmieniamy kolor w funkcji init() */
	mk_hercules_init( MY_COLOR); // wszystkie diody - fioletowe

	while (1) {

	}
}
```

Przypominam, że pełne kody źródłowe omawiane w filmach instruktażowych można bezpłatnie pobrać z linku poniże:

>>> Kliknij TUTAJ aby pobrać kody źródłowe <<<

https://atnel.pl/download/elektronika/MH/MH_Test_SOURCE.zip

Uwaga! Jeśli korzystasz z wyższego taktowania mikrokontrolera, np. zewnętrzny kwarc 16 MHz albo większy to funkcję send_byte() należy nieco rozbudować jak niżej:

```
void send_byte( uint8_t byte ) {

	static uint8_t cnt;
	cnt = 0x80;
	while( cnt ) {

		if( byte & cnt ) MOSI_HI;
		else MOSI_LO;

		SCK_HI;
		cnt >>= 1;

		#if F_CPU >= 16000000
		asm( "nop" );
		#endif
		#if F_CPU >= 18432000
		asm( "nop" );
		#endif
		#if F_CPU >= 20000000
		asm( "nop" );
		#endif
		#if F_CPU >= 24000000
		asm( "nop" );
		#endif

		SCK_LO;

		#if F_CPU >= 16000000
		asm( "nop" );
		#endif
		#if F_CPU >= 18432000
		asm( "nop" );
		#endif
		#if F_CPU >= 24000000
		asm( "nop" );
		#endif
	}
	MOSI_HI;
}
```

Rozbudowa funkcji send_byte() ma na celu dodanie za pomocą kompilacji warunkowej dodatkowych opóźnień w postaci rozkazów „nop” aby nie przekroczyć maksymalnej dopuszczalnej częstotliwości SCK większej niż 800 kHz.

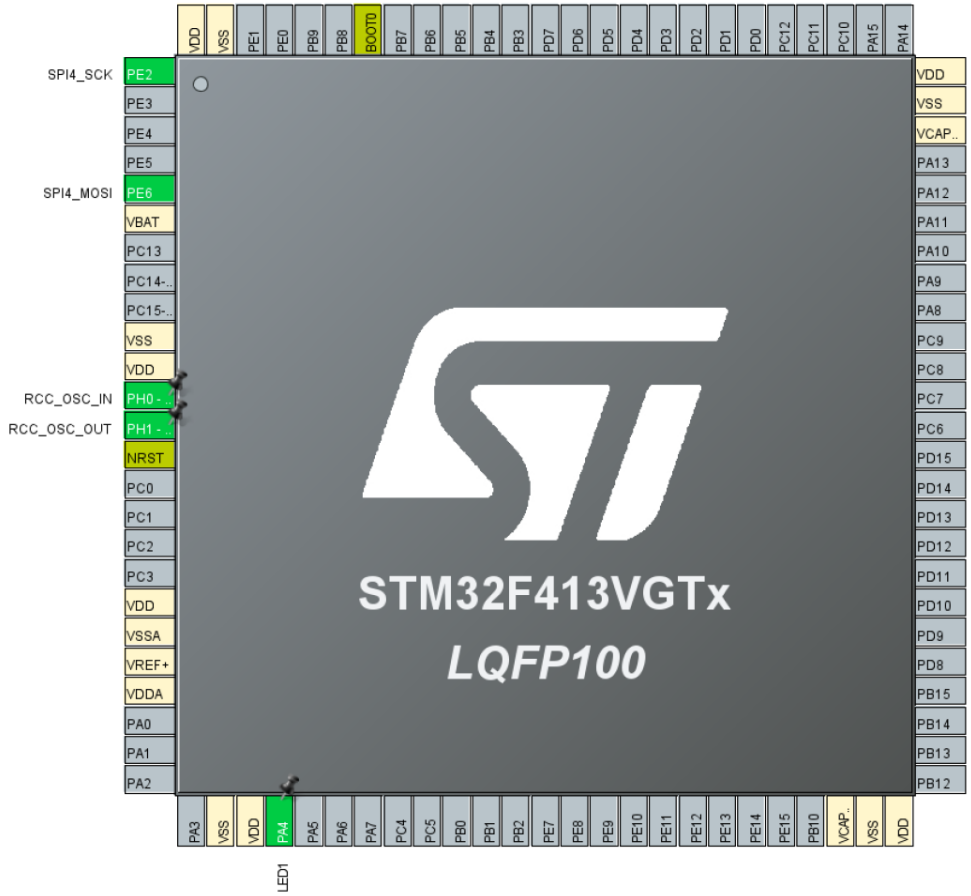
Uproszczony przykład kodu dla ARM/STM z użyciem DMA.

Oto jeden z najprostszych możliwych sposobów sterowania adresowalnymi diodami LED za pomocą modułu Magic HERCULES z poziomu mikrokontrolerów 32-bitowych STM, wykorzystujący popularne środowisko HAL a także sprzętowe moduły SPI *(hspi4)* i DMA. Sygnał zegarowy SPI można ustawić w CubeIDE idealnie na częstotliwość 800 kHz zapewniając tym samym maksymalną szybkość wyświetlania. Dzięki temu rozwiązaniu nie trzeba korzystać z żadnych innych peryferiów ani dodatkowego kodu obciążającego rdzeń procesora. Przykład został pomyślnie zrealizowany na mikrokontrolerze **STM32F413VGTx**.

```
/* RAM buffer for 1024 RGB W Magic LEDs */
char magic_buffer [ 4096];

/* main.c */
/* Initialize all configured peripherals */
	MX_GPIO_Init();
	MX_DMA_Init();
	MX_SPI4_Init();

	/* then somewhere in code */
	HAL_SPI_Transmit_DMA ( &hspi4, magic_buffer , 4096 );
```



.

INFORMACJE KONTAKTOWE

ATNEL Mirosław Kardaś

Adres:

ul. Kurza 24,

70 - 795 Szczecin

Telefon:

+48 91 4635 683

+48 501 008 523

Strona Internetowa:

www.atnel.pl

www.sklep.atnel.pl

www.youtube.com/mirekk36

e-mail:

biuro@atnel.pl

sklep@atnel.pl