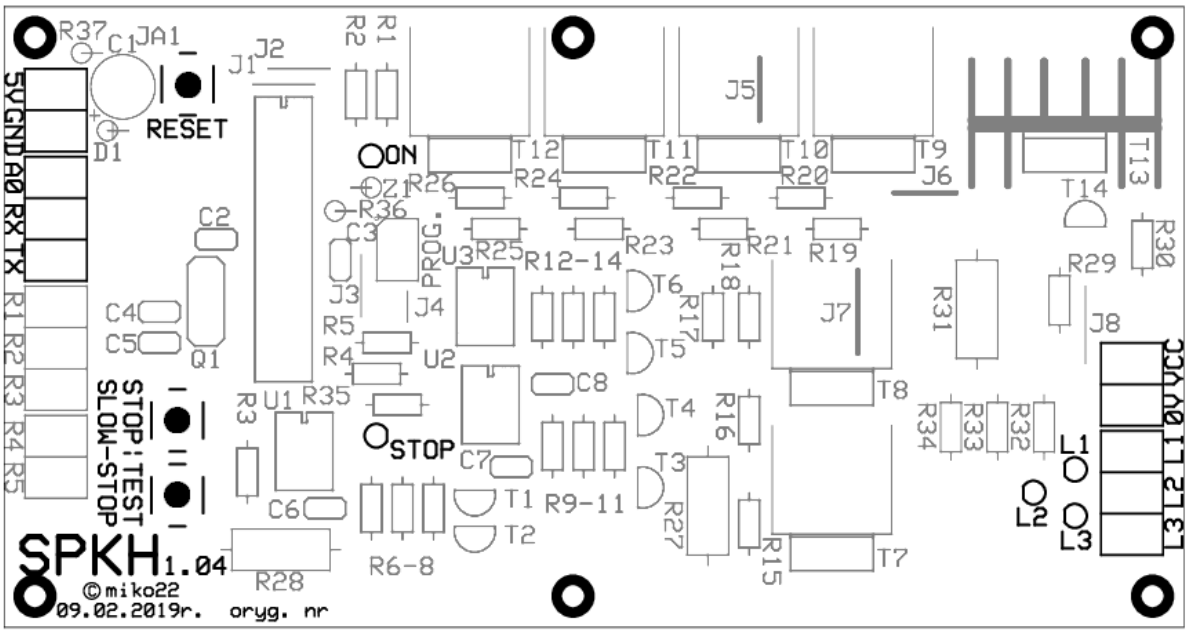
SPKH

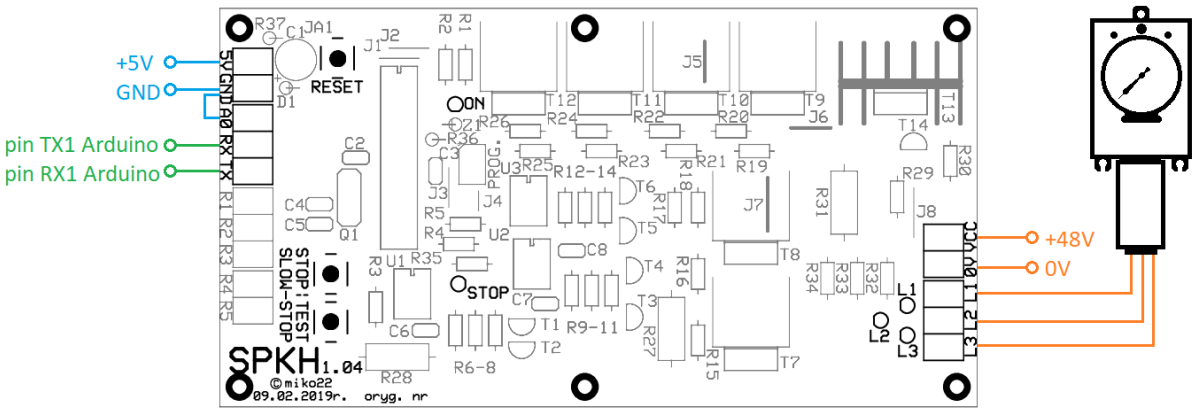
v1.04

(Sterownik Prędkościomierza Kolejowego Hasler)

Instrukcja obsługi



1. Schemat podłączenia do Arduino:



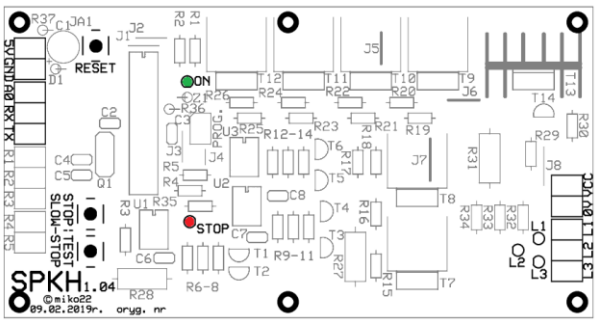
- 5V – zasilanie 5V z Arduino
- GND – masa z Arduino
- A0 – połączone mostkiem do GND
- RX – odbiór danych przez UART (57600 bps)
- TX – wysyłanie danych przez UART (57600 bps)
- VCC – zasilanie 48V DC min. 0,8A
- 0V – masa dla zasilania 48V
- L1 – 1. faza dla silnika prędkościomierza
- L2 – 2. faza dla silnika prędkościomierza
- L3 – 3. faza dla silnika prędkościomierza

Zasilanie 48V i poszczególne fazy dla silnika są odseparowane galwanicznie od zasilania 5V i danych odbieranych/wysyłanych przez UART.

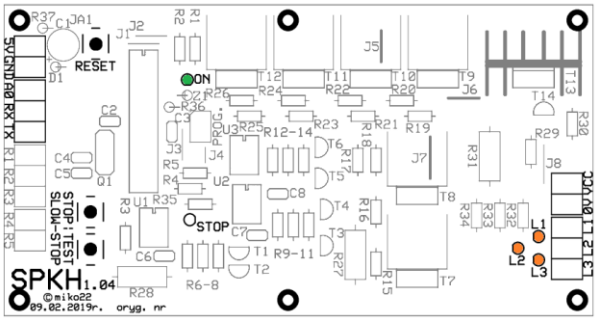
Wartość prędkości i inne komendy należy wysyłać do SPKH przez UART za pomocą funkcji *Serial1.print(liczba_calkowita)*, przy czym w przypadku komunikacji ciągłej (np. użytkowanie SPKH z symulatorem MaSzyna) zaleca się wysyłanie wartości prędkości co 100ms.
Dane wysyłane przez SPKH można odczytywać monitorem portu szeregowego Arduino IDE przez adapter USB-UART lub za pomocą funkcji *Serial1.read()* bezpośrednio w Arduino.

2. Podstawowe stany pracy:

Po podłączeniu zasilania 5V SPKH uruchamia się w stanie *zatrzymania*. Światłem ciągłym świeci dioda zielona (*ON*) i czerwona (*STOP*):

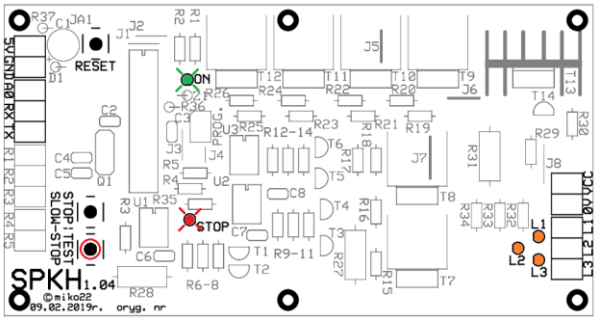


Po zadaniu prędkości przez UART SPKH przechodzi w stan *jazdy*. Dioda czerwona (*STOP*) gaśnie, a diody pomarańczowe (*L1*, *L2*, *L3*) wskazują obecność napięcia na poszczególnych fazach silnika:



Zadanie zerowej prędkości w stanie *jazdy* spowoduje przejście z powrotem do stanu *zatrzymania* i wyłączenie napięcia na fazach silnika.

Wciśnięcie i puszczenie przycisku *TEST* lub wysłanie przez UART komendy *9002* w stanie zatrzymania spowoduje wejście w stan *testu prawidłowości wskazań*. Dioda zielona (*ON*) i czerwona (*STOP*) zaczną mrugać naprzemiennie z częstotliwością 1Hz, a zadawana prędkość będzie automatycznie zwiększana o 5km/h co 20 sekund aż do końca zakresu skali prędkościomierza, a następnie analogicznie zmniejszana o 5km/h co 20 sekund aż do zera i ponownego wejścia w stan *zatrzymania*:



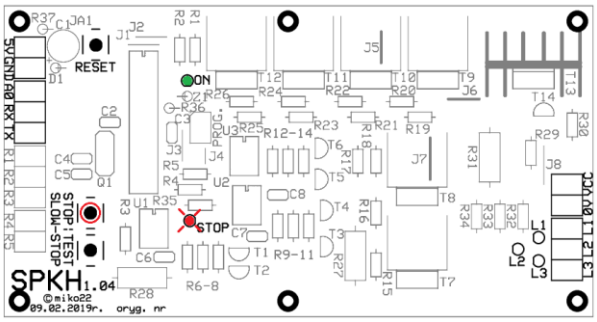
W trakcie *testu prawidłowości wskazań* diody pomarańczowe (*L1*, *L2*, *L3*) wskazują obecność napięcia na poszczególnych fazach silnika podobnie jak w stanie *jazdy*.

Do wywołania *testu prawidłowości wskazań* nie jest wymagane podłączenie SPKH do Arduino ani komputera – wystarczy zasilanie 5V ze złączem *A0* połączonym do złącza *GND* oraz zasilanie 48V.

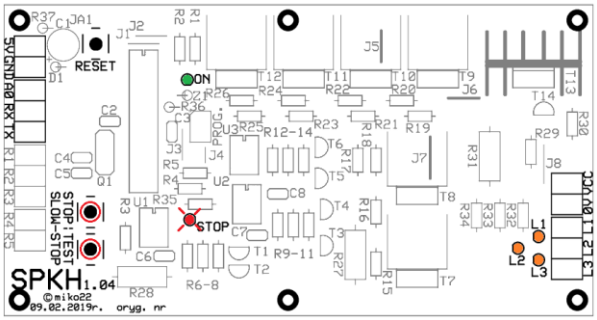
Zarówno w stanie *jazdy*, jak i w stanie *testu prawidłowości wskazań* można w każdej chwili wywołać *awaryjne zatrzymanie* lub *powolne zatrzymanie*:

- aby wywołać *awaryjne zatrzymanie*, należy wcisnąć i puścić przycisk *STOP* lub wysłać przez UART komendę *9000* – spowoduje to natychmiastowe wyłączenie zasilania na fazach silnika i przejście w stan *awaryjnego zatrzymania* sygnalizowane mruganiem diody czerwonej (*STOP*) z częstotliwością 4Hz i uniemożliwi jakiekolwiek inne działanie do czasu

przejścia w stan *zatrzymania* przez ponowne wciśnięcie i puszczenie przycisku *STOP* lub wysłanie przez UART komendy *9000*:



- aby wywołać *powolne zatrzymanie* (*SLOW-STOP*), należy jednocześnie wcisnąć i puścić przycisk *STOP* oraz *TEST* lub wysłać przez UART komendę *9001* – spowoduje to automatyczne płynne zmniejszanie zadawanej prędkości od aktualnej do zera w tempie 1km/h co 0,2 sekundy (5km/h co 1 sekundę) sygnalizowane mruganiem diody czerwonej (*STOP*) z częstotliwością 2Hz i uniemożliwi jakiekolwiek inne działanie z wyjątkiem przejścia w stan *awaryjnego zatrzymania*, które nastąpi także samoczynnie po zakończeniu procedury *powolnego zatrzymania*, w trakcie której diody pomarańczowe (*L1*, *L2*, *L3*) sygnalizują obecność napięcia na poszczególnych fazach silnika:



3. Inne funkcje wywoływane wyłącznie za pomocą komend:

- 9003 – wyłączenie pomijania zerowych wartości przy komunikacji z MaSzyną;
- 9004 – załączenie pomijania zerowych wartości przy komunikacji z MaSzyną – zerowa wartość prędkości musi być otrzymana co najmniej pięć razy z rzędu, aby SPKH przeszedł w stan zatrzymania (funkcja używana w celu eliminacji zakłóceń);
- 9005 – wyłączenie wymuszenia podwójnego odbioru takiej samej prędkości;
- 9006 – załączenie wymuszenia podwójnego odbioru takiej samej prędkości – wartość prędkości musi być otrzymana dwa razy z rzędu taka sama, aby SPKH zaktualizował prędkość zadawaną na wyjściu (funkcja używana w celu eliminacji zakłóceń);
- 9007 – załączenie zadawania prędkości w km/h razy 10 z dokładnością do 0,1 km/h – np. wysłanie do SPKH liczby 158 spowoduje zadanie prędkości 15,8km/h (funkcja dezaktywowana po każdorazowym wyłączeniu zasilania 5V SPKH);

- 9008 – zmiana generowanego przebiegu napięcia z sinusoidalnego na zbliżony do tego z rzeczywistej prądnicy prędkościomierza Hasler – niezalecane (funkcja dezaktywowana po każdorazowym wyłączeniu zasilania 5V SPKH);
- 9011 – odczyt numeru aktualnej konfiguracji tarczy i silnika prędkościomierza oraz naliczonego przebiegu kilometrów (np. przez adapter USB-UART i monitor portu szeregowego Arduino IDE lub za pomocą funkcji *Serial1.read()* bezpośrednio w Arduino) – aktualny przebieg kilometrów zapisywany jest automatycznie do pamięci SPKH przy każdym wyłączeniu zasilania 5V i odczytywany z pamięci przy ponownym podłączeniu zasilania 5V;
- 9099 – zapisanie aktualnej konfiguracji tarczy i silnika prędkościomierza do pamięci SPKH w celu samoczynnego ustawienia takiej konfiguracji przy następnym uruchomieniu SPKH;
- 9108 – ustawienie konfiguracji: tarcza 100km/h, silnik 800obr/min – konfiguracja nr 1;
- 9121 – ustawienie konfiguracji: tarcza 120km/h, silnik 1000obr/min – konfiguracja nr 2;
- 9128 – ustawienie konfiguracji: tarcza 120km/h, silnik 800obr/min – konfiguracja nr 3;
- 9158 – ustawienie konfiguracji: tarcza 150km/h, silnik 800obr/min – konfiguracja nr 4;
- 9188 – ustawienie konfiguracji: tarcza 180km/h, silnik 800obr/min – konfiguracja nr 5;
- 9999 – odczyt danych na temat SPKH: wersja, numer seryjny wersji, data wykonania, wykonawca i adres e-mail wykonawcy (np. przez adapter USB-UART i monitor portu szeregowego Arduino IDE).