W ofercie AVT*

AVT5945

Podstawowe parametry:

- służy do pomiaru liczby obrotów związanego z nim elementu mechanicznego,
- obsługuje sensor w postaci enkodera mechanicznego lub w postaci dwóch transoptorów szczelinowych i odpowiednio przygotowanej tarczy obrotowej,
- zakres pomiarowy wynosi 0...9999 obrotów,
- napięcie zasilania wynosi 4...9 V,
- prąd obciążenia to 7 mA (bez uwzględnienia prądu obciążenia podłączonych transoptorów).

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

AVT1824	Programowany licznik zdarzeń (EP 8/2014)
AVT1750	Licznik impulsów (EP 8/2013)
AVT3188	Licznik impulsów (zdarzeń)
AVT1810	Uniwersalny licznik z LCD

* Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania! Podstawową wersją zestawu jest wersja [B] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [B] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu. Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [C] – zmontowany, uruchomiony i przetestowany zestaw [B] (elementy wlutowane w płytkę PCB)
- wersja [A] – płytka drukowana bez elementów i dokumentacji

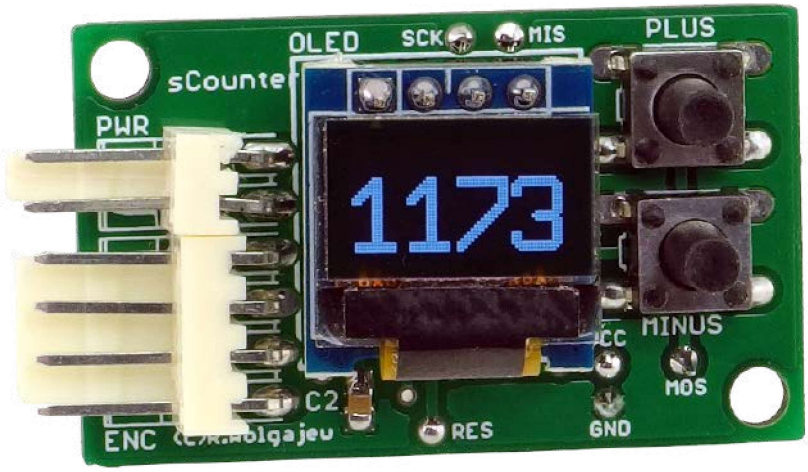
Kity, w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [A+] – płytka drukowana [A] + zaprogramowany układ [UK] i dokumentacja
- wersja [UK] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! – http://sklep.avt.pl.

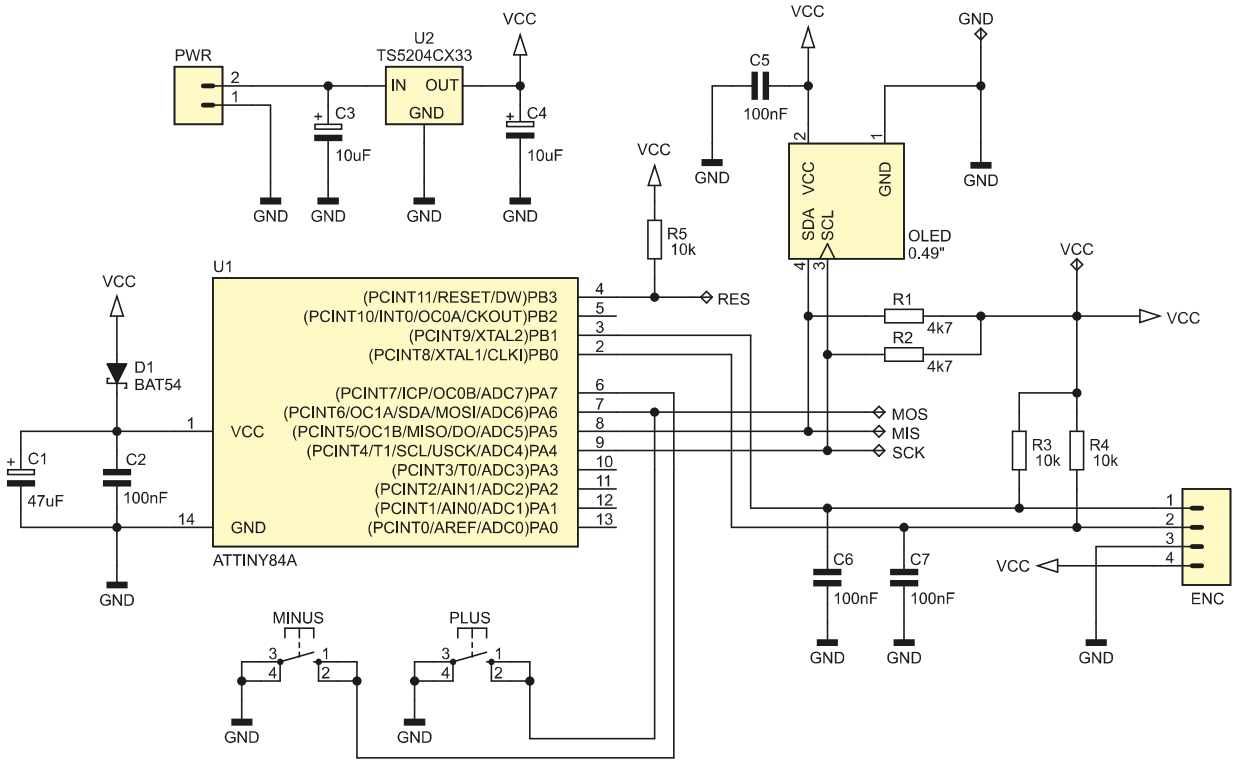
W przypadku braku dostępności na stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt Via e-mail: kity@avt.pl.

sCounter

Historia powstania tego projektu jest dość osobliwa. Otóż, jakiś czas temu napisał do mnie jeden z Czytelników naszego miesięcznika, który jest jednocześnie sympatykiem starej, dobrej techniki audio. Poprosił o pomoc w znalezieniu rozwiązania problemu konstrukcyjnego, jakim było zastąpienie mechanicznego licznika obrotów w starym magnetofonie szpulowym licznikiem elektronicznym. Specjalnie na potrzeby takiego zastosowania powstał zaprezentowany układ.



Wygląda na to, że dawne magnetofony szpulowe, a zwłaszcza kasetowe przechodzą w tej chwili renesans. Widać to dość dobrze po liczbie ofert na popularnych portalach aukcyjnych. Cena takich urządzeń w ostatnim roku znacznie wzrosła. Okazuje się, że jest naprawdę dość spora grupa sympatyków tego rodzaju sprzętu, z których część poddaje go modernizacjom wymuszonym



Rysunek 1. Schemat ideowy licznika sCounter

Listing 1. Plik nagłówkowy programowej implementacji magistrali I²C

```
#define I²C_BUS_FREQ 100000

#define SDA_PORT_REG PORTA
#define SDA_PIN_REG PINA
#define SDA_DDR_REG DDRA
#define SDA_PORT_NR PA5

#define SCL_PORT_REG PORTA
#define SCL_PIN_REG PINA
#define SCL_DDR_REG DDRA
#define SCL_PORT_NR PA4

#define SDA_AS_OUTPUT SDA_DDR_REG |= (1<<SDA_PORT_NR)
#define SDA_AS_INPUT SDA_DDR_REG &= ~(1<<SDA_PORT_NR)
#define SDA_HIGH SDA_PORT_REG |= (1<<SDA_PORT_NR)
#define SDA_LOW SDA_PORT_REG &= ~(1<<SDA_PORT_NR)
#define SDA_IS_LOW (!(SDA_PIN_REG & (1<<SDA_PORT_NR)))
#define SCL_AS_OUTPUT SCL_DDR_REG |= (1<<SCL_PORT_NR)
#define SCL_AS_INPUT SCL_DDR_REG &= ~(1<<SCL_PORT_NR)
#define SCL_HIGH SCL_PORT_REG |= (1<<SCL_PORT_NR)
#define SCL_LOW SCL_PORT_REG &= ~(1<<SCL_PORT_NR)
#define SCL_IS_LOW (!(SCL_PIN_REG & (1<<SCL_PORT_NR)))

#define WAITSTATE (500000UL/I²C_BUS_FREQ)

#define NACK 0
#define ACK 1
```

Listing 2. Funkcja inicjująca porty magistrali I²C

```
void i2cInit(void){
    //Porty SDA i SCL, jako wyjściowe
    //ze stanem wysokim
    SDA_HIGH;
    SCL_HIGH;
    SDA_AS_OUTPUT;
    SCL_AS_OUTPUT;
}
```

Listing 3. Funkcja odpowiedzialna za wygenerowanie sygnału Start magistrali I²C.

```
void i2cStart(void){
    //Wysyłamy sygnał START:
    //SDA z 1 na 0, w czasie gdy SCL=1
    SDA_LOW;
    _delay_us(WAITSTATE);
    //SCL z 1 na 0 - przygotowanie do transmisji
    SCL_LOW;
    _delay_us(WAITSTATE);
}
```

Listing 4. Funkcja odpowiedzialna za wygenerowanie sygnału Stop magistrali I²C

```
void i2cStop(void){
    SDA_LOW;
    SCL_HIGH;
    _delay_us(WAITSTATE);
    SDA_HIGH;
    _delay_us(WAITSTATE);
}
```

Listing 5. Funkcja odpowiedzialna za wysłanie bajta przy użyciu magistrali I²C

```
uint8_t i2cWriteByte(uint8_t Byte){
    uint8_t Ack;

    for(uint8_t i=0; i<8; ++i){
        if(Byte & 0x80) SDA_HIGH; else SDA_LOW;
        Byte <<= 1;
        SCL_HIGH;  //Slave zatrzaskuje bieżący bit
        SCL_AS_INPUT;
        //Sprawdzamy, czy Slave
        // nie przedłuża sygnału zegarowego
        while(SCL_IS_LOW);
        SCL_AS_OUTPUT;
        _delay_us(WAITSTATE);
        SCL_LOW;
        _delay_us(WAITSTATE);
    }

    //Odczytujemy bit potwierdzenia z układu Slave
    SDA_HIGH;
    SDA_AS_INPUT;
    SCL_HIGH;
    SCL_AS_INPUT;
    //Sprawdzamy, czy Slave
    // nie przedłuża sygnału zegarowego
    while(SCL_IS_LOW);
    SCL_AS_OUTPUT;
    _delay_us(WAITSTATE);
    //Odczytujemy bit potwierdzenia układu Slave
    Ack = (SDA_PIN_REG & (1<<SDA_PORT_NR))>>SDA_PORT_NR;
    SCL_LOW;
    _delay_us(WAITSTATE);
    SDA_AS_OUTPUT;

    return Ack;
}
```

niesprawnością lub chęcią ulepszenia urządzenia. Zachęcony tym wyzwaniem postanowiłem zbudować bardzo proste i zarazem niewielkie urządzenie, które będzie można zaadaptować do wspomnianych celów, a jednocześnie będzie na tyle uniwersalne, że może znaleźć zastosowanie w innych urządzeniach, jak dla przykładu nawijarka cewek, zliczanie osób wchodzących/wychodzących z pomieszczenia, zliczanie przedmiotów na taśmie transmisyjnej i tym podobne. Mowa o uniwersalnym liczniku obrotów nazwanym przeze mnie sCounter.

Budowa i działanie

Schemat ideowy układu został pokazany na **rysunku 1**. Zbudowano bardzo prosty, wręcz banalny, system mikroprocesorowy, który jest sterowany niewielkim mikrokontrolerem firmy Microchip (dawniej Atmel) o oznaczeniu ATtiny84A. Zawiera on wewnętrzny oscylator RC o częstotliwości 1 MHz, który w tej aplikacji jest źródłem taktowania. Mikrokontroler jest odpowiedzialny za realizację całej, założonej funkcjonalności – realizuje 3 podstawowe zadania:

- obsługuje niewielki wyświetlacz OLED o organizacji 64×32 piksele i przekątnej 0,49 cala, sterowany dzięki programowej realizacji interfejsu I²C;
- obsługuje 2 przyciski funkcyjne PLUS i MINUS, wykorzystując do tego wbudowany układ czasowo-licznikowy Timer1 – dzięki czemu obsługa tych przycisków nie wstrzymuje programu obsługi aplikacji i umożliwia rozróżnienie krótkiego i długiego przyciśnięcia;
- obsługuje sensor licznika obrotów w postaci enkodera mechanicznego lub dwóch transoptorów szczelinowych i odpowiednio przygotowanej tarczy obrotowej.

Użycie tak niewielkiego wyświetlacza OLED wynikało z potrzeby minimalizacji wymiarów zewnętrznych licznika, bo o czym wspomniano na wstępie, ma on docelowo zastępować rozwiązania mechaniczne. Oczywiście, by zliczanie obrotów stało się w ogóle możliwe, użytkownik musi przygotować odpowiednie rozwiązanie w zakresie mechaniki, co sprowadza się do tego, że wałek napędowy, którego obroty chcemy zliczać, należy sprzęgnąć z enkoderem mechanicznym lub, co zdecydowanie bardziej polecane (z uwagi na bezawaryjność i trwałość konstrukcji), należy wyposażyć w tarczę obrotową ze szczelinami i dwa czujniki szczelinowe umieszczone jeden obok drugiego, które umożliwią zarówno zliczanie obrotów, jak i rozróżnienie kierunku obrotów.

Uważny Czytelnik zapewne zastanowi się, dlaczego zastosowano programową realizację interfejsu I²C, skoro mikrokontroler ATtiny84A wyposażono w uniwersalny interfejs szeregowy USI (*Universal Serial Interface*). Niestety sprzęg USI nie zapewnia całkowicie sprzętowego wsparcia medium transmisyjnego I²C, przez co, w moim przekonaniu, programowa implementacja jest w pełni uzasadniona i na dodatek prostsza. Po drugie, w przypadku programowego wsparcia interfejsu I²C do wyboru mamy dowolne piny mikrokontrolera, co ułatwia zaprojektowanie obwodu drukowanego urządzenia. Z tych właśnie powodów zdecydowałem się zaimplementować sprzęg I²C na drodze czysto programowej. Inną sprawą jest fakt, że tego typu implementacja jest bardzo prosta, gdyż wspomniane medium transmisyjne jest dobrze udokumentowane i proste w swoich założeniach. Co istotne, nie będę w tym miejscu opisywał wszystkich, szczegółowych cech standardu, gdyż łatwo znaleźć stosowne informacje w sieci, a skupię się jedynie na mojej autorskiej implementacji programowej.

Na początek plik nagłówkowy implementacji definiujący wszystkie niezbędne cechy sprzętowe, którego zawartość pokazano na **listingu 1**. Na **listingu 2** pokazano z kolei prostą funkcję inicjującą porty magistrali I²C, której zadaniem jest ustawienie stanów spoczynkowych. Dalej, dwie podstawowe funkcje standardu I²C generujące sygnał Start i Stop, których kod został pokazany na **listingu 3** i **listingu 4**. Dalej, na **listingu 5** i **listingu 6** pokazano kolejne dwie proste funkcje umożliwiające wysłanie i odebranie bajta na magistrali I²C. Funkcja wysyłająca, jako rezultat swojego działania, zwraca wartość

bitu potwierdzenia po stronie odbiornika danych (w tym przypadku układu Slave), zaś funkcja pozwalająca na odbiór bajta przyjmuje jako argument wywołania wartość bitu potwierdzenia po stronie odbiornika danych (w tym przypadku układu Master). Co ciekawe, obie funkcje obsługują mechanizm *clock stretching*, czyli możliwość chwilowego wstrzymywania komunikacji przez układ Slave w czasie, gdy nie jest on w stanie przesyłać kolejnych danych. Tyle w kwestiach implementacyjnych. Prawda że proste? Wróćmy zatem do naszego urządzenia, jakim jest projekt sCounter.

Uważny Czytelnik zauważy pewną nietypową implementację w zakresie zasilania samego kontrolera, która angażuje diodę Schottky'ego D1 oraz kondensator elektrolityczny C1. Czemu ma służyć takie rozwiązanie? Jak wiemy, nasz licznik ma zastępować mechaniczne liczniki tego typu, w związku z czym musi on pamiętać wskazania po zaniku zasilania. Jak tego dokonać? Najprościej jest dokonywać każdorazowego zapisu zmiany jego wartości do pamięci EEPROM mikrokontrolera, jednak tego typu rozwiązanie ma poważną wadę. Producent mikrokontrolera deklaruje, że trwałość pamięci EEPROM wynosi w dużym przybliżeniu 100 tysięcy cykli zapisu, z czego wynika, że dość szybko wyczerpalibyśmy dostępne możliwości. Jak rozwiązałem ten temat? Zdecydowałem się na zapis wartości licznika przy zaniku zasilania, stąd niezbędny układ separacji i podtrzymania zasilania mikrokontrolera w postaci wspomnianych dwóch elementów. W celu detekcji momentu wyłączenia zasilania zastosowano wbudowany w mikrokontroler przetwornik ADC pracujący w trybie Free Running, monitorujący kilka tysięcy razy na sekundę napięcie zasilające mikrokontroler (po zaniku zasilania dioda D1 zapewnia separację zasilania mikrokontrolera od reszty urządzenia, a kondensator C1 zapewnia odpowiedni czas podtrzymania zasilania).

Co ciekawe, na pierwszy rzut oka nie wydaje się, by nasz sterownik w jakikolwiek sposób używał przetwornika ADC, gdyż żaden z kanałów wejściowych nie jest prze niego używany w tym celu. To prawda, patrząc na schemat układu i nie mając do dyspozycji listingu programu, można by wysnuć taki wniosek. Jest jednak zgoła inaczej. Nasz przetwornik ADC mierzy specjalne, wewnętrzne napięcie V_{BG}=1,1 V (wartość dla mikrokontrolera ATtiny84A) dzięki temu, że wewnętrzny, analogowy multiplekser przetwornika może zostać właśnie w ten sposób ustawiony. Napięciem odniesienia jest z kolei napięcie zasilające mikrokontroler, czyli napięcie dostarczane na wyprowadzenie VCC. Spadek tego napięcia, podczas wyłączania zasilania, powoduje wzrost wartości wyniku przetwarzania według wzoru jak niżej (korzystamy z 8-bitowej rozdzielczości przetwornika):

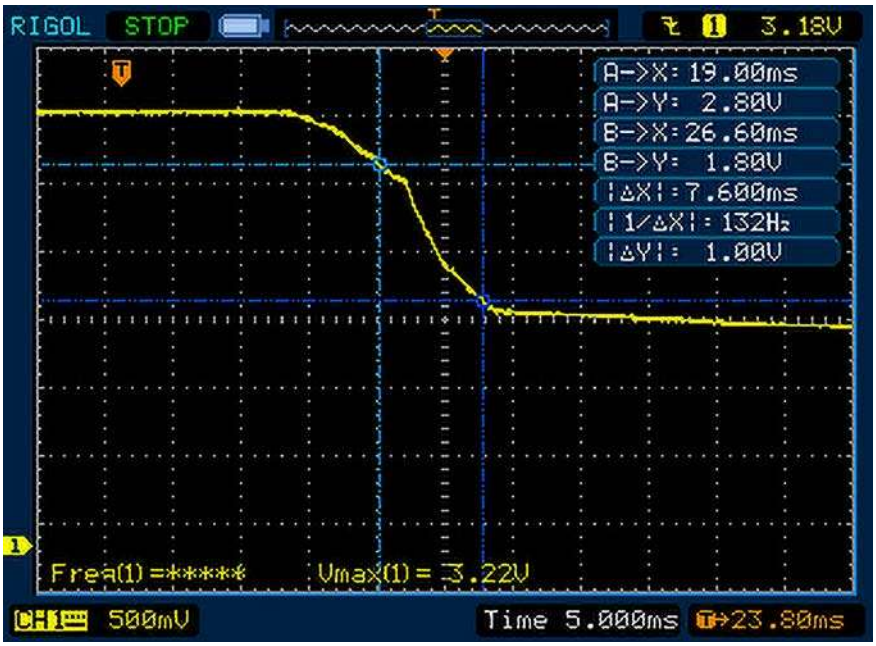
$V_{ADC}=(V_{BG}\times 256)/VCC$

Procedura obsługi przerwania przetwornika ADC sprawdza każdorazowo, czy nie został przekroczony zdefiniowany wcześniej próg obliczeniowy, a jeśli ma to miejsce, to inicjuje proces zapisywania wartości licznika do wbudowanej pamięci EEPROM, po czym czeka, aż napięcie zasilania spadnie do poziomu resetowania mikrokontrolera, które dokonywane jest przez uruchomiony wcześniej układ BOD (typowo przy wartości 1,8 V). Wspomniany próg zadziałania mechanizmu zapisu ustawiono na wartość 2,8 V, co oznacza, że czas opadania napięcia zasilającego od wartości 2,8 V do wartości 1,8 V jest czasem, w którym mikrokontroler musi przeprowadzić zapis wartości licznika – wyłącznie 4 bajty danych.

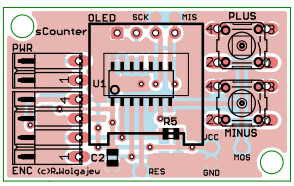
Listing 6. Funkcja odpowiedzialna za odbiór bajta przy użyciu magistrali I²C.

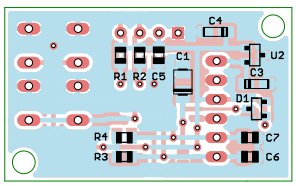
```
uint8_t i2cReadByte(uint8_t Ack){
    uint8_t Byte = 0;

    SDA_AS_INPUT;
    SDA_HIGH;

    for(uint8_t i=0; i<8; ++i){
        Byte <<= 1;
        SCL_HIGH;
        SCL_AS_INPUT;
        //Sprawdzamy, czy Slave
        //nie przedłuża sygnału zegarowego
        while(SCL_IS_LOW);
        SCL_AS_OUTPUT;
        _delay_us(WAITSTATE);
        //Odczytujemy bieżący bit wystawiony
        //przez układ Slave
        if(!SDA_IS_LOW) Byte++;
        SCL_LOW;
        _delay_us(WAITSTATE);
    }

    //Transmitujemy 9-ty bit Ack
    SDA_AS_OUTPUT;
    if(Ack) SDA_LOW; else SDA_HIGH;
    //Slave zatrzaskuje bit potwierdzenia
    SCL_HIGH;
    SCL_AS_INPUT;
    //Sprawdzamy, czy Slave
    //nie przedłuża sygnału zegarowego
    while(SCL_IS_LOW);
    SCL_AS_OUTPUT;
    _delay_us(WAITSTATE);
    SCL_LOW;
    _delay_us(WAITSTATE);

    return Byte;
}
```

Jak pokazały testy praktyczne, zastosowanie wspomnianego wcześniej rozwiązania sprzętowego (dioda D1 i kondensator C1) i mechanizmów programowych zapewnia 100% skuteczność zapisu danych z bardzo dużym marginesem czasowym. Na **rysunku 2** pokazano wykres zależności napięcia

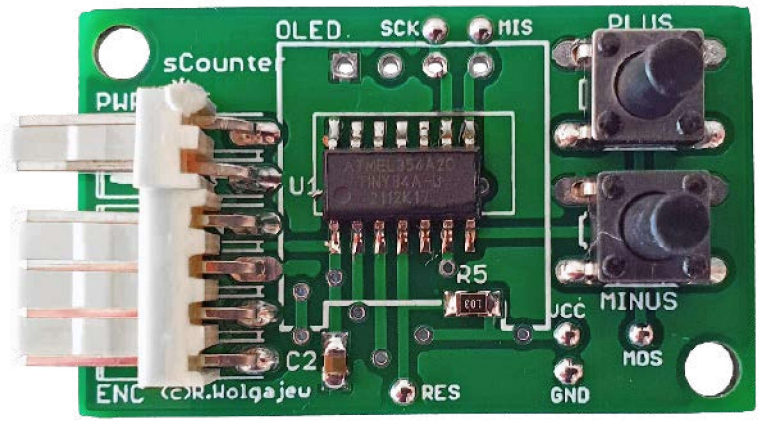


Rysunek 2. Wykres zależności napięcia zasilania mikrokontrolera w funkcji czasu podczas wyłączania urządzenia

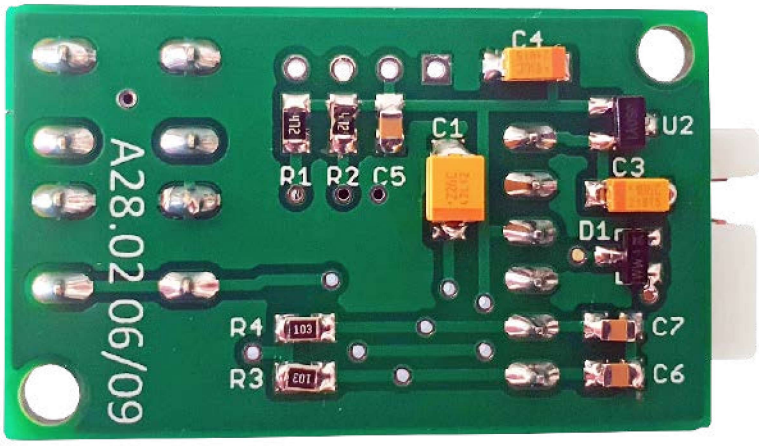




Rysunek 3. Schemat montażowy licznika sCounter



Fotografia 1. Zmontowane urządzenie sCounter (od strony TOP) tuż przed przylutowaniem wyświetlacza OLED



Fotografia 2. Zmontowane urządzenie sCounter od strony BOTTOM

zasilania mikrokontrolera w funkcji czasu podczas wyłączania urządzenia. Jak widać, dla prototypowego kondensatora C1 o wartości 22 µF (docelowo zmienionego na 47 µF) osiągnięto dostępny czas zapisu na poziomie 7,6 ms, co prawie dwukrotnie przekracza niezbędny czas na zapis pamięci EEPROM mikrokontrolera (4 ms). To tyle, jeśli chodzi o szczegóły natury programistycznej, przejdźmy zatem do schematu montażowego.

Montaż i uruchomienie

Schemat montażowy został pokazany na **rysunku 3**. Zaprojektowano bardzo małą dwustronną płytkę drukowaną z przewagą elementów do montażu powierzchniowego SMD montowanych po obu stronach obwodu drukowanego. Montaż urządzenia rozpoczynamy od warstwy BOTTOM, gdzie przylutowujemy wszystkie elementy bierne oraz półprzewodniki. Następnie przechodzimy na warstwę TOP, gdzie w pierwszej kolejności przylutowujemy mikrokontroler, następnie elementy bierne, dalej wyświetlacz OLED, posiłkując się złączem goldpin, a na samym końcu montujemy złącza ENC i PWR oraz przyciski funkcyjne PLUS i MINUS.

Poprawnie zmontowany układ nie wymaga żadnej regulacji i powinien działać po włączeniu zasilania. Na **fotografii 1** pokazano zmontowane urządzenie (od strony TOP) przed przylutowaniem wyświetlacza OLED, zaś na **fotografii 2** pokazano to samo urządzenie od strony BOTTOM. Warto również wspomnieć, że w przypadku użycia transoptorów szczelinowych jako czujników obrotu nie należy montować elementów R3, R4 oraz C6 i C7. Są one niezbędne wyłącznie w przypadku enkodera mechanicznego.

Ostatnim etapem uruchamiania urządzenia jest podłączenie do złącza ENC odpowiedniego sensora. W przypadku enkodera mechanicznego korzystamy z 3 wyprowadzeń złącza ENC: sygnałowych oznaczonych jako 1 i 2 oraz masy oznaczonej numerem 3. W przypadku czujników szczelinowych

Ustawienia Fusebitów:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1
EESAVE: 0
BODLEVEL2...0: 110

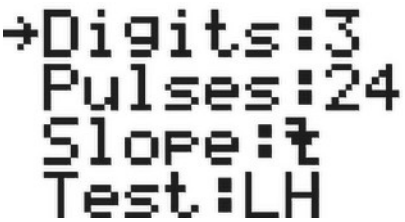
korzystamy dodatkowo z wyprowadzenia numer 4, które dostarcza napięcia zasilania tym czujnikom. Oczywiście wyjście każdego z czujników podłączamy do innego złącza sygnałowego (1 lub 2). Na koniec ustawiamy parametry konfiguracyjne, korzystając z menu urządzenia.

Obsługa urządzenia

Układ standardowo służy do pomiaru liczby obrotów związanego z nim elementu mechanicznego (za pomocą odpowiedniego sensora), w związku z czym podstawowym trybem pracy urządzenia jest wyświetlanie liczby obrotów. Niemniej jednak z uwagi na to, że jest to rozwiązanie dość uniwersalne, przewidziano stosowne menu konfiguracyjne, za pomocą którego ustawiamy parametry sprzętowe podłączonego czujnika, jak też inne wartości konfiguracyjne. Menu konfiguracyjne urządzenia sCounter pokazano na **rysunku 4**. Jak widać, ustawiamy następujące parametry:

- Digits – liczba cyfr, jakie wyświetla licznik obrotów (3 lub 4),
- Pulses – liczba impulsów podłączonego czujnika na jeden obrót wału,
- Slope – rodzaj zbocza, które generuje zliczanie impulsów (rosnące, opadające).

Dodatkowa pozycja tego menu niepodlegająca regulacji to pozycja Test. Umożliwia ona poprawne, wzajemne ustawienie podłączonych czujników szczelinowych, tak aby każdy z nich w danym położeniu wystawiał na wyjście inny stan logiczny. Oczekiwane wartości to: LH lub HL. Jest to konieczne dla poprawnego zliczania i detekcji kierunku obrotów przez 2 czujniki szczelinowe położone



Rysunek 4. Menu konfiguracyjne urządzenia sCounter

WYKAZ ELEMENTÓW, które możesz zamówić w sklepie AVT na stronie sklep.avt.pl lub bezpośrednio (ul. Leszczynowa 11, 03-197 Warszawa, tel. 48222578451, e-mail: handlowy@avt.pl):

Rezystory: (SMD0805)
R1, R2: 4,7 kΩ
R3...R5: 10 kΩ

Kondensatory:
C1: tantalowy 47 µF/10 V (SMD B/3528)
C2, C5...C7: 100 nF ceramiczny X7R (SMD0805)
C3, C4: 10 µF/10 V tantalowy (SMD A/3216)

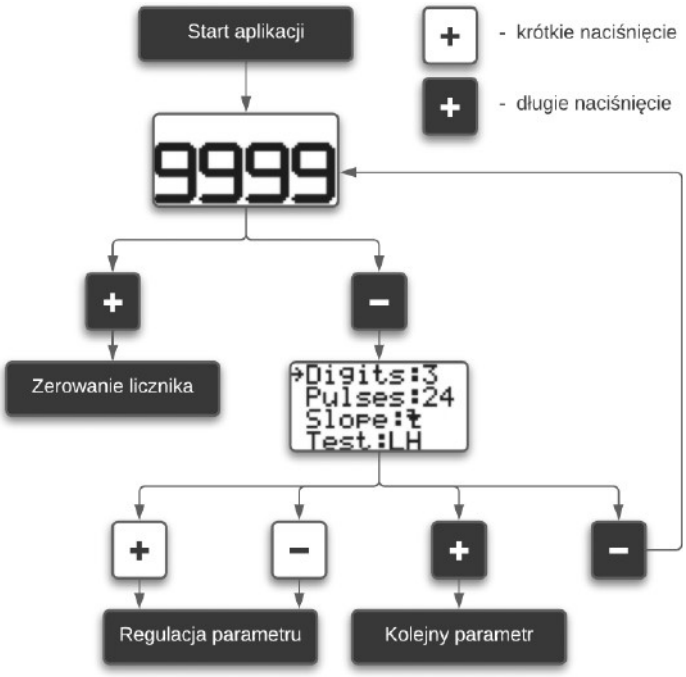
Półprzewodniki:
U1: ATtiny84A (SOIC14)
U2: TS5204CX33 (SOT23)
D1: BAT54 (SOT23)
OLED: wyświetlacz OLED 64×32 px, 0,49", I²C, 15×16 mm

Pozostałe:
PWR: gniazdo męskie kątowe 2 piny (NSL25-2W)
ENC: gniazdo męskie kątowe 4 piny (NSL25-4W)
MINUS, PLUS: microswitch TACT wysokość 6 mm

obok siebie i obsługiwane przez to samo koło szczelinowe. Niezbędna może okazać się modyfikacja standardowego koła szczelinowego i jego odpowiednie położenie w stosunku do czujników szczelinowych, co należy dobrać na drodze eksperymentalnej. Sposób obsługi urządzenia sCounter i znaczenie poszczególnych przycisków funkcyjnych pokazano na **rysunku 5**.

Wyjście z systemu menu powoduje zapamiętanie wszystkich parametrów konfiguracyjnych w nieulotnej pamięci EEPROM mikrokontrolera. Warto również podkreślić, że podczas normalnej pracy licznika wprowadziłem bardzo efektowny efekt animacji przypominający swoim działaniem pracę starych liczników mechanicznych, gdzie zmianie znaku towarzyszyło przesunięcie się jednego znaku w górę lub w dół i wskoczenie na jego miejsce znaku kolejnego (w przypadku liczników były to oczywiście cyfry). Muszę przyznać, że wygląda to nadspodziewanie efektownie.

Robert Wołgajew, EP



Rysunek 5. Sposób obsługi urządzenia sCounter