W ofercie AVT*

AVT5903

Podstawowe parametry:

- steruje wyświetlaczem matrycowym zawierającym 256 diod LED typu WS2812,
- wyświetla czas, temperaturę oraz ciśnienie,
- wbudowany budzik, który alarmuje dźwiękiem piania koguta.

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

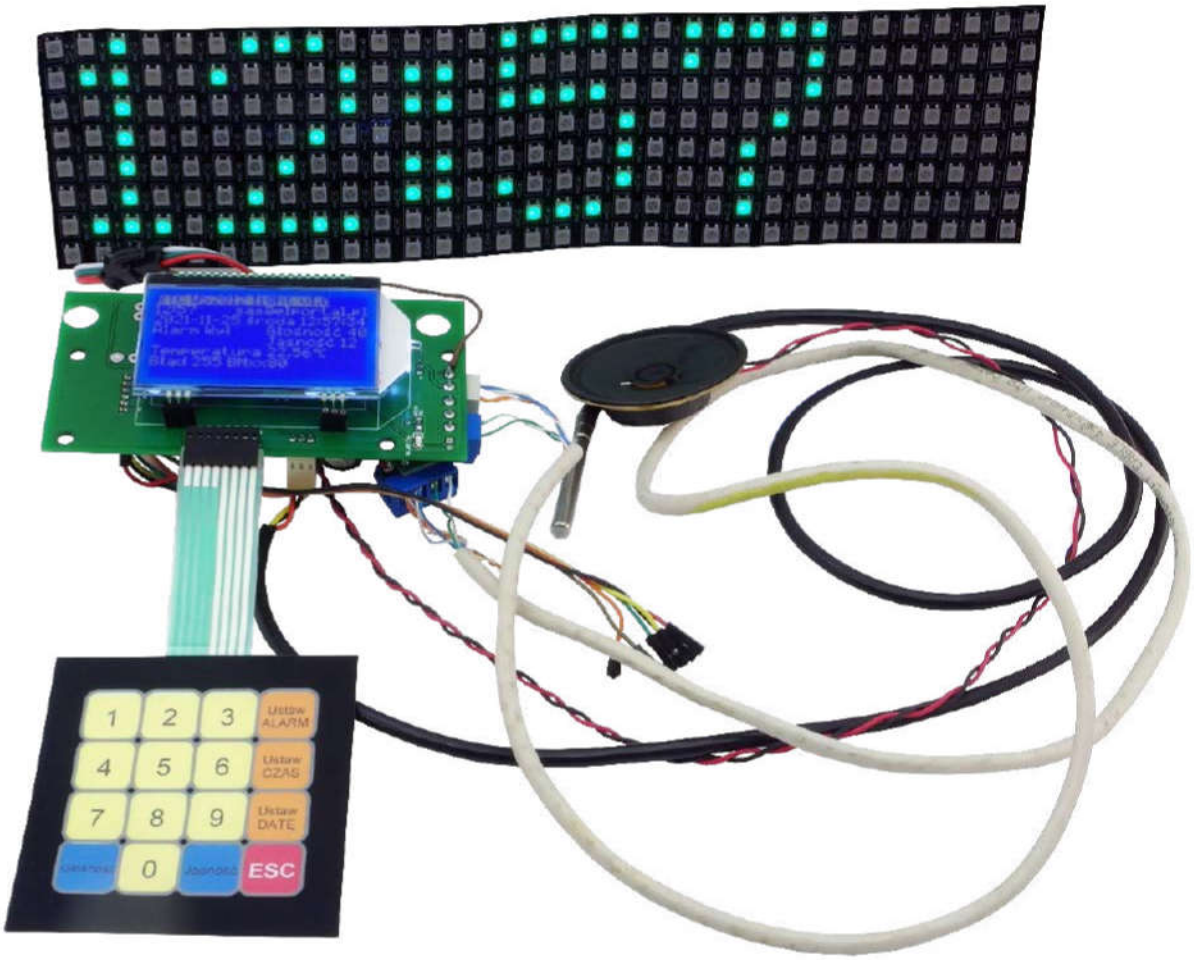
AVT5846	Flames (EP 3/2021)
-	Modułowa matryca LED (EP 4/2019)
AVT5606	Magic Matrix (EP 10–11/2017)
AVT3184	Zegar/termometr z wyświetlaczem matrycowym (EdW 11/2017)
AVT5561	Efektowny sterownik oświetlenia (EP 12/2016)
AVT2784	Zegar matrycowy (EdW 4/2006)

* 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.



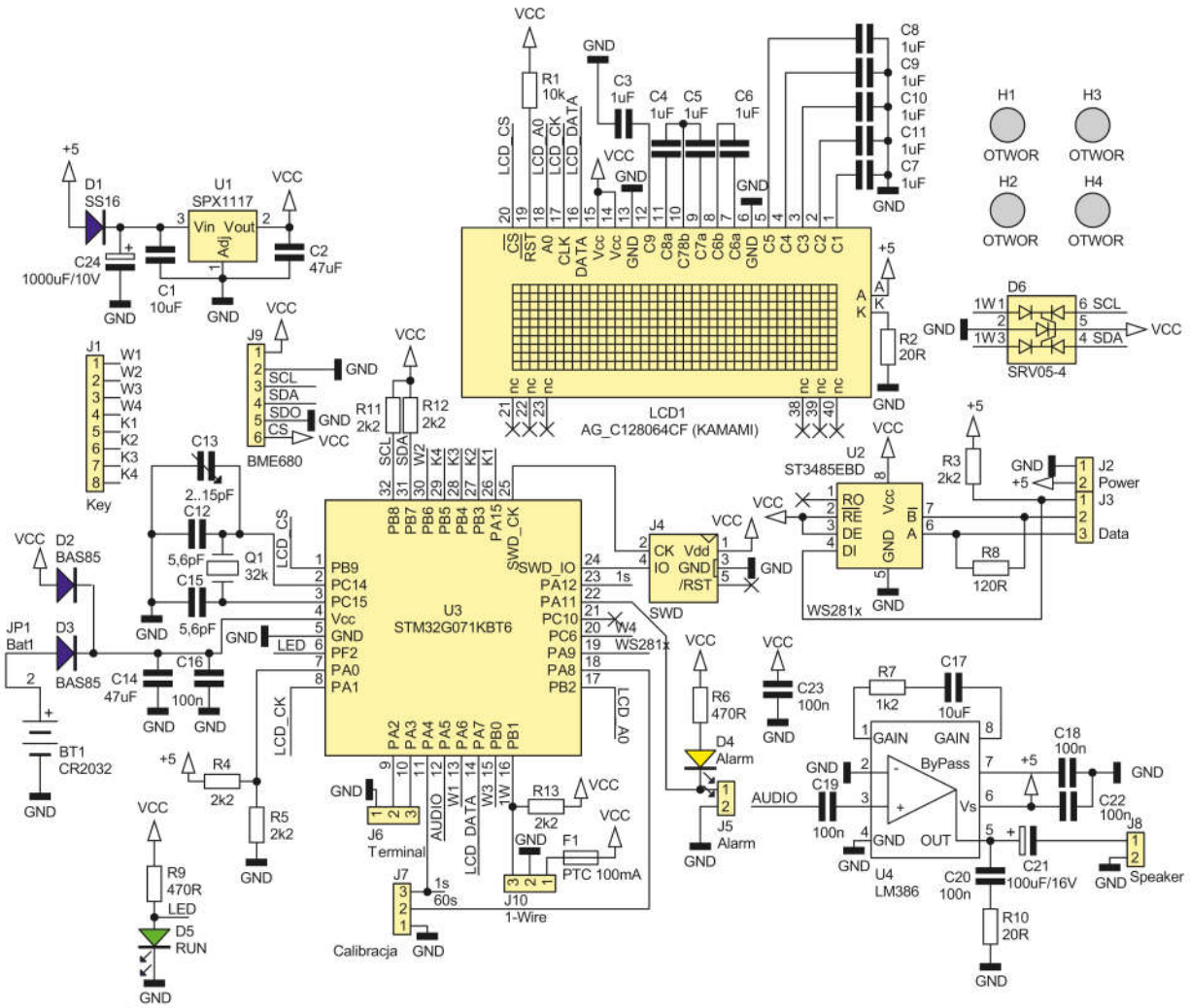
Zegar z wyświetlaczem matrycowym na diodach WS2812

Diody WS2812 zdobyły dużą popularność za sprawą sporych możliwości, prostego interfejsu i niskiej ceny. Zaprezentowany w artykule zegar steruje wyświetlaczem matrycowym zawierającym 256 diod LED tego typu, co umożliwia wyświetlenie 5...6 znaków czcionki o wielkości 5×8 w dowolnym kolorze. Poza wyświetlaniem czasu, zegar pokazuje także temperaturę oraz ciśnienie, a wbudowany budzik alarmuje dźwiękiem… piania koguta.

Konstrukcje podobne do opisanej w tym artykule z pewnością już powstały z zastosowaniem popularnej platformy Arduino. Jednak, aby zrealizować wszystkie funkcjonalności, Arduino wymagałoby kilku modułów rozszerzających, zawierających m.in. układ zegara RTC oraz odtwarzacz plików audio. Podwyższa to cenę urządzenia i zmniejsza jego niezawodność. W zaprezentowanym zegarze zastosowano wysokowydajny mikrokontroler typu STM32G071K z rdzeniem ARM Cortex M0+. Przy cenie ok. 11 zł, poza 128 kB pamięci FLASH i 36 kB pamięci RAM, ma on liczne peryferia w tym RTC, UART (bardzo usprawniający obsługę LED WS2812), SPI (niezbędny do sterowania wyświetlaczem LCD), 12-bit DAC i co ważne – kontroler DMA, który odciąża CPU od czasochłonnych transferów danych.

Budowa i działanie

Schemat zegara został pokazany na **rysunku 1**. Zasilanie układu jest doprowadzane do złącza J2

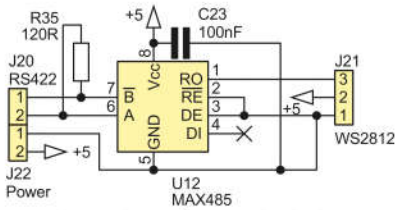


Rysunek 1. Schemat ideowy zegara

i poprzez diodę D1 trafia do wejścia stabilizatora napięcia U1. Mikrokontroler U3 jest zasilany napięciem 3,3 V z wyjścia tego stabilizatora poprzez dodatkową diodę D2. Jest ona niezbędna, ponieważ separuje główne zasilanie 3,3 V od awaryjnego zasilania mikrokontrolera z baterii. Dużo lepszym rozwiązaniem niż połączone D2 i D3 byłby specjalizowany przełącznik zasilania, ale biorąc pod uwagę fakt, że przerwy w zasilaniu będą dla tego układu rzadkie i z baterii zegar będzie korzystał sporadycznie, można pogodzić się z niedoskonałością takiego rozwiązania. Zamiast baterii lepszym rozwiązaniem wydaje się zastosowanie superkondensatora, który bez problemu podtrzyma pracę zegara przez kilka dni.

Wyświetlacz graficzny zastosowany w urządzeniu – LCD1, jest sterowany interfejsem SPI. Sygnał audio z przetwornika DAC jest przekazywany do wzmacniacza mocy U4. Regulacja poziomu dźwięku jest realizowana cyfrowo, dlatego brak jest potencjometru regulacji głośności. Sygnał dla matrycy LED dostarczany jest na wyprowadzenie 1. złącza J3. Rezystor R6 ustala poziom wysoki jako 5 V, co gwarantuje poprawną pracę WS2812 (przy napięciu 5 V wymagają minimum 3,3 V w stanie wysokim). Podciąganie do 5 V jest możliwe, ponieważ wyjście UART pracuje w trybie OD (Open Drain), a GPIO akceptuje napięcie do wartości 5,5 V. Dodatkowo, po konwersji do poziomów RS485/422, sygnał dla WS2812 jest doprowadzony do pinów 2 i 3 złącza J3. Umożliwia to niezawodną transmisję na duże odległości, dzięki czemu układ zegara może być znacznie oddalony od wyświetlacza. Więcej o sposobie sterowania WS2812 można przeczytać w artykule „Sterowanie diod WS2812 poprzez DMX" opublikowanym w EP 4/2021.

Klawiatura matrycowa 4×4 jest podłączona bezpośrednio do portów mikrokontrolera przez złącze J1. Termometr DS18B20 podłączono do złącza J10. Dioda D6 zabezpiecza port mikrokontrolera przed ładunkami elektrostatycznymi podobnie jak sygnały magistrali I²C potrzebne do komunikacji z czujnikiem ciśnienia. Złącze J4 umożliwia zaprogramowanie mikrokontrolera. Opcjonalnie można skorzystać z bootloadera za pośrednictwem interfejsu UART wyprowadzonego na złącze J6.



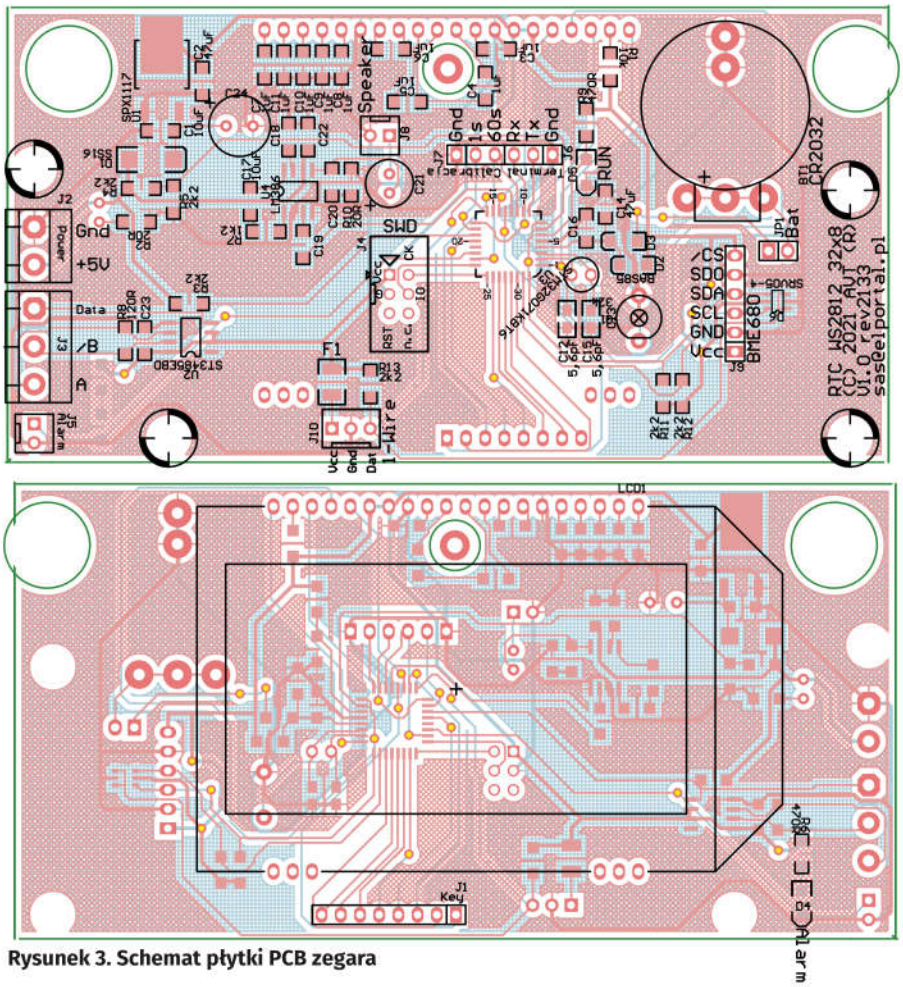
Rysunek 2. Schemat ideowy dodatkowego interfejsu RS485

W przypadku wykorzystania do sterowania WS281x sygnału o poziomach RS485/422 wymagany jest prosty układ konwertera na układzie MAX485 pokazany na **rysunku 2**. Płytka PCB konwertera powstała na potrzeby innego projektu, lecz okazała się optymalna do tego zastosowania.

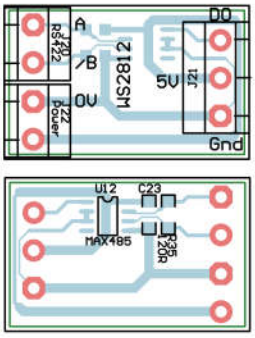
Wyjaśnienia może wymagać rola rezystorów R4, R5. Służą one do wykrycia zaniku zasilania. Jest to konieczne, aby uśpić mikrokontroler, ponieważ nie ma on osobnego wyprowadzenia do zasilania RTC, jak ma to miejsce w mikrokontrolerach o większych obudowach. Gdyby mikrokontroler nie został uśpiony, pobierałby prąd około 50 mA, co szybko wyczerpałoby baterię, o ile w ogóle pozwoliłaby ona na pracę mikrokontrolera, ponieważ CR2032 ma niewielką obciążalność (ma dużą rezystancję wewnętrzną).

Montaż i uruchomienie

Schemat płytki zegara został pokazany na **rysunku 3**, a dodatkowej płytki z układem dla interfejsu RS485 na **rysunku 4**. Montaż jest typowy i nie wymaga omawiania. Płytka zegara dostosowana jest do umieszczenia w obudowie K-M60.



Rysunek 3. Schemat płytki PCB zegara



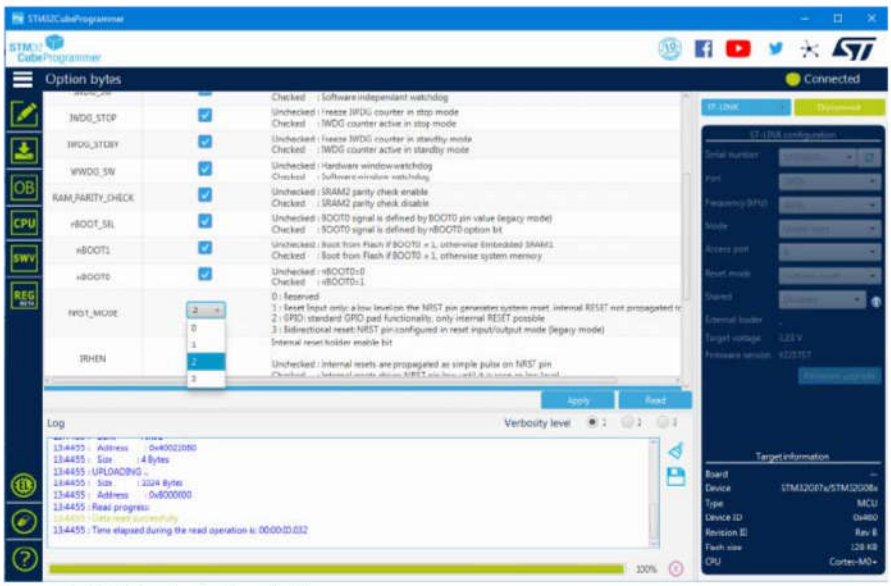
Rysunek 4. Schemat płytki PCB dodatkowego interfejsu RS485

Fotografia 1. Wygląd ekranu głównego

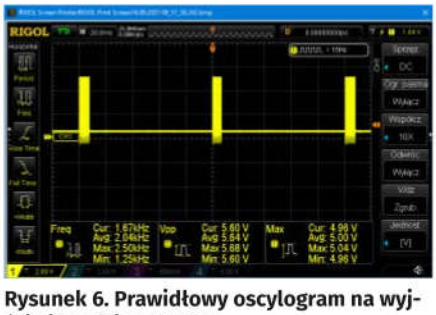
Przed załadowaniem programu do mikrokontrolera warto sprawdzić napięcie zasilania 3,3 V. Po załadowaniu programu należy ustawić bit konfiguracyjny *NRST_MODE* na wartość 2 – tak jak pokazano na **rysunku 5**. Bez tego dioda D5 – *Run*, nie będzie migała. Jeśli program pracuje poprawnie, na wyświetlaczu LCD pojawi się ekran powitalny, a po chwili ekran główny (**fotografia 1**). Dodatkowo na matrycy LED pokaże się godzina, wyświetlana naprzemiennie z datą, temperaturą i ciśnieniem. Jeśli termometr lub czujnik temperatury nie jest podłączony, ich pomiary są pomijane na LED, a na LCD wyświetlany jest komunikat o błędzie.

Przy zastosowaniu sygnałów RS485/422 do połączenia konwertera z wyświetlaczem trzeba mieć na uwadze to, aby przewody sygnałowe były wykonane w postaci skrętki. Może do tego posłużyć np. kabel UTP. W razie wątpliwości, czy przewody sygnałowe zostały podłączone prawidłowo, można sygnał sprawdzić oscyloskopem na płytce układu ze schematu 2. Na wyjściu MAX485 powinny być paczki impulsów o długości niecałych 8 ms w odstępach 100 milisekund. Pomiędzy paczkami powinien występować poziom niski (**rysunek 6**). Jeśli przerwy pomiędzy paczkami mają poziom H, oznacza to, że przewody A i /B są zamienione miejscami. Aby oscylogram był stabilny, warto skorzystać z zaawansowanego wyzwalania długością impulsu.

Przed zakończeniem opisu budowy zegara warto na chwilę zatrzymać się przy obwodzie zasilania awaryjnego. Pierwotnie źródłem tego zasilania miała być bateria CR2032. Jednak lepszym rozwiązaniem wydaje się zastosowanie superkondensatora. Cena takiego rozwiązania jest porównywalna do ceny akumulatora, ale trwałość jest zdecydowanie większa. Proponuję więc, zamiast baterii, zastosować prosty układ pokazany na **rysunku 7**. Świadomie zrezygnowano ze specjalizowanego układu ładowania i kontroli superkondensatora, co pozwala zmontować układ „na pająka" lub na płytce uniwersalnej. Konieczne jest zastosowanie dwóch kondensatorów, ponieważ ich maksymalne napięcie pracy wynosi 2,5...2,7 V. Nie należy stosować



Rysunek 5. Ustawienia fuse bitów



Rysunek 6. Prawidłowy oscylogram na wyjściu interfejsu RS485

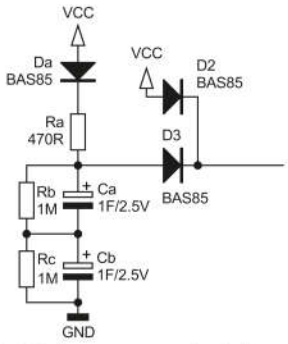
WYKAZ ELEMENTÓW, które możesz zamówić w sklepie AVT na stronie sklep.avt.pl lub bezpośrednio (ul. Leszczynowa 11, 03-197 Warszawa, tel. 48222578451, e-mail: handlowy@avt.pl):

Rezystory: (SMD1206)
R1: 10 kΩ
R2, R10: 20 Ω
R7 1,2 kΩ
R3, R4, R5, R11, R12, R13: 2,2 kΩ
R8: 120 Ω
R6, R9: 470 Ω

Kondensatory:
C1, C17: 10 µF SMD1206
C2, C14: 47 µF SMD1206
C3....C11: 1 µF SMD1206
C13: trymer 2...25 pF
C12, C15: 5,6 pF SMD0805
C16, C18, C19, C22, C23, C20: 100 nF SMD1206
C21: 100 µF/16 V elektrolityczny
C24: 1000 µF/10 V elektrolityczny

Pozostałe:
Q1: rezonator kwarcowy 32,768 Hz zegarkowy
F1: bezpiecznik PTC 100 mA SMD1210
BT1: CR2032
J1: goldpin kątowy 1×8
J2: złącze TB-5.0-PP-2P + TB-5.0-PIN24
J3: złącze TB-5.0-PP-3P + TB-5.0-PIN24
J4: złącze T821-1-06-S1
J5: złącze NS25-W2K
J6: złącze NS25-G3
J7: goldpin prosty 1×3
J8: złącze NS25-W2P
J9: gniazdo goldpin 1×6
J10: złącze NS25-W3
JP1: goldpin 1×2 + jumper
Klawiatura: klawiatura typu QW-02 (np.: https://bit.ly/3l8fmD9)

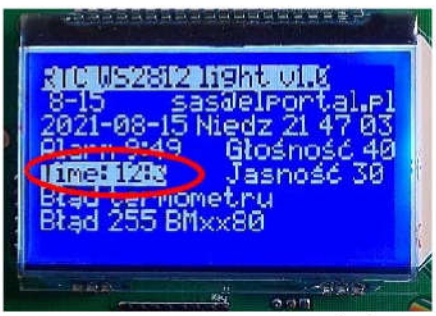
Półprzewodniki:
D1: SS16 (SMD)
D2, D3: BAS85 (SMD)
D4: dioda LED żółta (SMD1206)
D5: dioda LED zielona (SMD1206)
D6: SRV05-4 (SOT-23-6)
U1: SPX1117 (SOT-223)
U2: ST3485EBD (SO-8)
U3: STM32G071KBT6 (TQFP32)
U4: LM386 (SO-8)
LCD1: wyświetlacz AG_C128064CF (np. https://bit.ly/3HQEV5l)



Rysunek 7. Schemat zastąpienia baterii superkondensatorami

kondensatorów na 5 V, ponieważ charakteryzują się one dużą rezystancją wewnętrzną i nie do końca sprawdzą się w tym zastosowaniu. Superkondensatory (Ca i Cb) mają małą rezystancję wewnętrzną, dlatego rezystor Ra ogranicza ich prąd ładowania. Lepszym rozwiązaniem byłoby źródło prądowe, które szybciej naładowałoby kondensatory, ale w przypadku zegara nie ma to dużego znaczenia, a rozwiązanie z rezystorem jest proste, tanie i niezawodne. Dioda Da zabezpiecza przed przepływem prądu z kondensatorów do obwodów innych niż mikrokontroler. Rezystory Rb i Rc są prostym balancerem, zabezpieczającym superkondensatory przed uszkodzeniem przy zbyt wysokim napięciu powstałym przez nierównomierny rozpływ prądu ładowania. Oczywiście elementy związane z superkondensatorami włączamy zamiast baterii CR2032.

Na koniec warto napisać kilka słów o klawiaturze. Typowe klawiatury z szesnastoma przyciskami mają najczęściej klawisze oznaczone cyframi 0...9, literami A...D oraz znaki # i *. Wykonanie klawiatury na zamówienie to duży koszt, zwłaszcza przy niewielkich ilościach.



Fotografia 2. Wygląd ekranu ustawiania czasu

Na szczęście dostępne są klawiatury, do których można wsunąć dowolny wydruk. Taka klawiatura jest dostępna między innymi w TME: https://bit.ly/3l8fmD9. Wzór rysunku klawiatury znajduje się w materiałach dodatkowych, zarówno w postaci pliku PDF, jak i edytowalnej.

Obsługa zegara

Pierwszą czynnością, jaką należy wykonać po włączeniu zasilania, jest ustawienie zegara. W tym celu należy nacisnąć przycisk *Ustaw CZAS*, po czym wprowadzić godzinę w formacie gg:mm:ss z poprzedzającymi zerami. Wprowadzany czas jest widoczny na wyświetlaczu LCD (**fotografia 2**). Wprowadzanego czasu nie trzeba akceptować żadnym przyciskiem, zostanie on wprowadzony do RTC po wpisaniu jedności sekund. Wprowadzanie można anulować przyciskiem *ESC*. W przypadku złego formatu czasu wyświetlany jest komunikat błędu.

Ustawianie daty realizuje się podobnie jak ustawianie czasu, z tym że należy użyć przycisku *Ustaw DATĘ*, którą wprowadza się w formacie rrrr-mm-dd-t, gdzie t to dzień tygodnia w zakresie 1...7 (1 – poniedziałek). Godzinę alarmu ustawia się po naciśnięciu klawisza *Ustaw ALARM* w formacie gg:mm. Ustawienie alarmu na 00:00 wyłączy go – nie można ustawić alarmu na północ, ale można minutę wcześniej albo później. Po załączeniu alarmu zostanie wygenerowany dźwięk piejącego koguta (https://bit.ly/3l8hM4A). Dodatkowo wyjście PA11 jest zwierane do masy, co powoduje zaświecenie diody D4 – *Alarm*. Maksymalny czas generowania dźwięku jest ograniczony do trzech minut. Można go skrócić, naciskając przycisk *ESC*, co także zdezaktywuje wyjście *Alarm*.

Przycisk *Jasność* pozwala na ustawienie jasności wyświetlacza LED w zakresie 1...99. Głośność alarmu i dźwięków generowanych podczas naciskania klawiszy regulujemy w zakresie 0...99 po naciśnięciu przycisku *Głośność*. Nie zaleca się przekraczania wartości 40...50, bo wzmacniacz będzie przesterowany. Można co prawda uniknąć przesterowania wzmacniacza, ale wymaga to zwiększenia napięcia zasilania, to z kolei wymaga modyfikacji wartości R3 lub podłączenia go do 3,3 V. Alternatywą jest usunięcie R3 i przełączenie wyjścia UART z trybu OD (Open Drain) na PP (PushPull), ale to wymaga modyfikacji oprogramowania (zainteresowanych proszę o kontakt via e-mail). Przy zwiększeniu napięcia zasilającego należy mieć na uwadze moc wydzielaną w stabilizatorze U1. W przypadku wykorzystania zewnętrznego układu wzmacniacza sygnał można pobrać z C19. Po dostosowaniu go do poziomu wymaganego przez wzmacniacz regulacja głośności będzie działała poprawnie w zakresie 1...99.

Program sterujący

Program, jak praktycznie wszystkie pisane przeze mnie, bazuje na maszynie stanów – nie czeka w pętli na zdarzenie, tylko realizuje przerwania oraz DMA. Dzięki temu nie ma sytuacji, że program czeka na np. naciśnięcie przycisku przez użytkownika, przez co nie realizuje innych zadań, np. odświeżania zawartości wyświetlacza. Daje to wrażenie pracy wielozadaniowej, dlatego należy zadbać, aby poszczególne zadania nie wykonywały się zbyt długo. Gdy o to nie zadbany, program może mieć „czkawkę". Takich problemów nie mają systemy czasu rzeczywistego – RTOS, ale powodują inne, przez co napisanie dobrego, dużego programu dla RTOS nie jest łatwe. Osobiście sięgam po RTOS, gdy muszę równocześnie realizować czasochłonne zadania, które ciężko zrealizować tylko na przerwaniach czy przez maszynę stanu, jak równoczesna obsługa systemu plików i innego czasochłonnego zadania. Jest to możliwe bez RTOS, ale wymagałoby przepisania kodu systemu plików, co jest zadaniem niebanalnym.

W przypadku zegara najdłuższe zadanie to obsługa LCD, która nie została napisana z wykorzystaniem przerwań i DMA, przez co zajmuje kilkanaście milisekund. Można by pokusić się o modyfikację procedur transferu do LCD, ale w wypadku zegara nie ma to większego sensu. Wielu początkujących programistów, zwłaszcza na platformy Arduino, może zastanawiać się, jak zrealizowano odczyt temperatury tak, aby nie zawieszać głównej pętli programu na około 700 ms. Jest to możliwe, realizując obsługę termometru z użyciem przerwań, ale w przypadku termometru skorzystano z prostszego rozwiązania, zamiast realizować odczyt w tradycyjny sposób. Cały proces przebiega tak:

1. Uruchomienie konwersji;
2. Odczekanie czasu konwersji (dla 12-bitowej rozdzielczości jest to ok. 700 ms);
3. Odczyt temperatury.

Postąpiono inaczej:

1. Odczyt temperatury (pierwszy jest błędny, dlatego jest ignorowany);
2. Uruchomienie konwersji.

Listing 1. Kod procedury regulacji głośności wykonywanej w przerwaniu

```
void irqDac(){
    uint16_t const ofs = 128;
    uint32_t const size = kogut4b_8b_wav_size;
    uint16_t static dana;

    if( pSample++ >=  size-ofs) pSample = 0;
    dana =  kogut4b_8b_wav[ofs+pSample];

    dana *= volume;
    dana >>= 8;          // dzielenie przez 256
    HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_2, DAC_ALIGN_8B_R, dana );
}
```

Po tych operacjach mikrokontroler realizuje inne zadania i dopiero gdy timer odliczy czas potrzebny na konwersję, realizowany będzie odczyt i uruchomienie kolejnej konwersji. Dzięki temu operacje na 1-Wire trwają maksymalnie 4 ms (odczyt danych z termometru razem z CRC). Wartość CRC jest konieczna do wyeliminowania wszelkich błędów w działaniu i uszkodzeń w obwodzie czujnika temperatury.

Zegar nie ma potencjometru regulacji głośności, czy to tradycyjnego, czy też elektronicznego. Regulację głośności zrealizowano w dosyć prosty, cyfrowy sposób. Każdą próbkę dźwięku mnoży się przez wartość głośności w zakresie 1...255, po czym wynik należy podzielić przez 256. Aby uniknąć niepotrzebnej i czasochłonnej operacji dzielenia, wynik mnożenia przez 1...255 jest przesuwany w prawo o 8 bitów, co odpowiada dzieleniu przez 256. Kod regulacji głośności wykonywany w przerwaniu został pokazany na **listingu 1**. Należy też mieć na uwadze fakt, że cyfrowa regulacja, przy małych poziomach sygnału, zmniejsza rozdzielczość próbek dźwięku, a co za tym idzie jego jakość. Jednak zegarek to nie sprzęt Hi-Fi i nie stanowi to problemu.

W mikrokontrolerze na dźwięk przeznaczone jest blisko 70 kB (program zajmuje blisko 60 kB), co pozwala na odtworzenie niecałych dziewięciu sekund nagrania (8 bitów, 8 kHz). W razie konieczności można zwiększyć czas nagrania dwukrotnie, przy okazji zwiększając rozdzielczość do 12-bit. Jest to możliwe po zastosowaniu kompresji ADPCM. Taka kompresja została użyta w AVT5841 (dekoder DCC trakcji i dźwięku do modeli kolejowych) opublikowany w EP 2 i 3/2021.

Jeszcze kilka słów o obsłudze czujnika ciśnienia. Kod jego obsługi został przeniesiony z Arduino i okazało się (jak zwykle), że ma poważne błędy i niedociągnięcia. Nadużywanie operacji zmiennoprzecinkowych pominę, bo to już norma w bibliotekach dla Arduino, ale skupię się na innym problemie. W przypadku, gdy brakuje czujnika ciśnienia lub termometru, zegar pomija wyświetlanie tych parametrów na wyświetlaczu. Niestety okazało się, że w przypadku braku czujnika ciśnienia program zawiesza się w trakcie inicjalizacji czujnika i następuje restart CPU przez układ watchdog. Po analizie kodu funkcji *BMP280_Init* w oczy rzuca się taki oto fragment programu:

```
while(BMP280_Read8(BMP280_
CHIPID) != 0x58);
```

Jak widać, to wieczna pętla, dopóki czujnik nie zwróci poprawnej wartości.

W przypadku zainteresowania zaawansowaną wersją zegara (https://bit.ly/3nREdgg) proszę o kontakt via e-mail.

Sas
sas@elportal.pl

- Opis sterownika diod WS2812 poprzez DMX opublikowanego w EP 4/2021 jest dostępny na: https://bit.ly/3r6efHO
- Opis dekodera DCC opublikowanego w EP 2/2021 i EP 3/2021 jest dostępny na: https://bit.ly/32mtbqP i na: https://bit.ly/3DVO0Hz

REKLAMA