W ofercie AVT*
AVT6016

Podstawowe parametry:
- liczba przegródek: 16,
- maksymalna liczba dziennych dawek: 5,
- źródło napięcia zasilającego: bateria alkaliczna typu AAA,
- prąd pobierany ze źródła zasilania (maksymalny/tryb uśpienia): 55 mA/80 µA (szczegóły w tekście artykułu).

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media
AVT5541 Dozownik detergentu, czyli czas na elektronizację WC (EP 6/2016)

* **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**.

Autor składa podziękowania firmie Artronic za dostarczenie wyświetlacza LCD zastosowanego w projekcie urządzenia e-Nurse.

e-Nurse
– elektroniczny dyspenser leków (1)

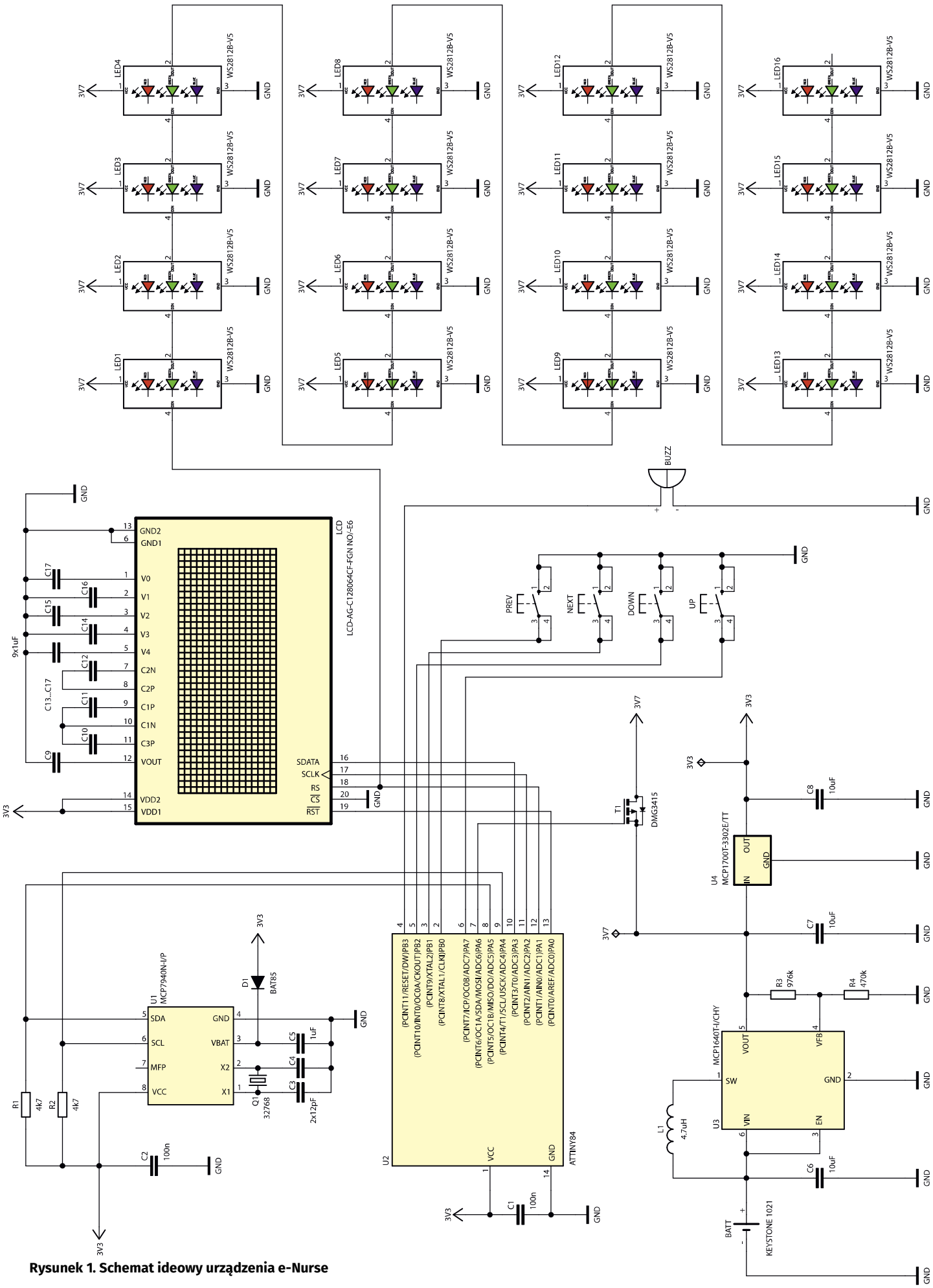
Pomysł na urządzenie tego typu narodził się… tak po prostu, zwyczajnie i nagle, jak wiele moich projektów. I dobrze się stało, że przyszedł mi do głowy akurat w tym czasie, gdy święta za pasem, gdyż, o czym się zaraz przekonacie, może on być idealnym i oryginalnym prezentem świątecznym dla naszych babć i dziadków, ale nie tylko. Otóż postanowiłem zaprojektować dyspenser leków podobny do tych plastikowych, dobrze znanych pudełeczek dostępnych w większości aptek lub drogerii, ale wyposażony w interfejs umożliwiający pełną konfigurację oraz automatyzację. Taki swego rodzaju przypominacz sprzętowy.

Przystępując do prac konstrukcyjnych, zdefiniowałem pewne, kluczowe założenia, które prezentują się następująco:
- Urządzenie będzie miało fizyczne „przegródki” (16 sztuk oznaczonych od „A” do „P”), w które będzie można włożyć przyjmowane leki (całą, dzienną dawkę);
- Przegródki, o których mowa powyżej, będą podświetlane w przypadku, gdy zachodzi konieczność przyjęcia stosownego leku;
- Kolor podświetlenia zależny będzie od liczby potencjalnie pominiętych dawek, a mianowicie: niebieski dla jednej dawki, zielony dla dwóch dawek i czerwony dla 3…5 dawek;
- Urządzenie będzie miało wyświetlacz graficzny o niskim poborze mocy, przy udziale którego (i współistniejących przycisków funkcyjnych) będzie można je skonfigurować;
- Konfiguracja, o której mowa, będzie niezależna dla każdej „przegródki” (oznaczonej od „A” do „P”), a obejmować będzie następujące ustawienia:
  - aktywność „przegródki” (gdyż można ją wstępnie skonfigurować, ale na obecną chwilę nie aktywować),
  - dzienną liczbę dawek leku (1…5),
  - godziny przyjmowania leku (dowolne z zakresu 0…23),
  - dni tygodnia przyjmowania leku,
- Konieczność przyjęcia leku sygnalizowana będzie dodatkowo przez wbudowany buzzer piezoelektryczny;
- Urządzenie standardowo pracować będzie w stanie uśpienia, ograniczając moc pobieraną ze źródła zasilania, zaś jego wybudzenie następować będzie na skutek użycia klawisza funkcyjnego (UP) lub poprzez nadejście zdarzenia konieczności przyjęcia leku;
- Urządzenie będzie automatycznie przechodzić w stan uśpienia po czasie 30 sekund bezczynności i nieobecności żadnych alarmów;
- Każde przyjęcie leku wymuszone pojawieniem się alarmu wymagać będzie potwierdzenia na ekranie głównym graficznego interfejsu użytkownika, które to potwierdzenie wyłączy alarm dźwiękowy oraz wygasi stosowną diodę LED podświetlenia przegródki (w przypadku pominięcia wielu dawek leku konieczne będzie potwierdzenie wszystkich zaległych alarmów);
- Urządzenie wyposażone będzie w zegar czasu rzeczywistego RTC;
- Wszystkie aktywne (czyli niepotwierdzone) alarmy będą automatycznie kasowane o północy. Do czasu skasowania alarmów urządzenie nie przejdzie w stan uśpienia;
- Do zasilania urządzenia przewidziano jedną baterię typu AAA.

To tyle, jeśli chodzi o założenia. Prawda, że zapowiada się ciekawie? A będzie jeszcze ciekawiej, gdyż do projektu urządzenia dołączony zostanie projekt obudowy dla „przegródek” wykonany w aplikacji do modelowania 3D a przeznaczony do wydrukowania na drukarce 3D. Zaczynajmy!

Budowa i działanie

Na **rysunku 1** pokazano schemat ideowy urządzenia e-Nurse. Jak widać, zaprojektowano bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) pod postacią układu ATtiny84



Rysunek 1. Schemat ideowy urządzenia e-Nurse

taktowany wewnętrznym oscylatorem o częstotliwości 8 MHz. Zastosowanie tak wysokiej, jak na tego rodzaju mikrokontroler, częstotliwości taktowania jest niejako w sprzeczności z potrzebą zachowania energooszczędności urządzenia, ale jest konieczne ze względu na fakt, że mikrokontroler nasz steruje pracą grupy 16 adresowalnych diod LED RGB, których specyfikacja interfejsu komunikacyjnego narzuca bardzo restrykcyjne wymogi czasowe. Niemniej jednak, i o czym wspomniałem wcześniej, urządzenie nasze standardowo przebywa w trybie uśpienia, przez co tak wysoka częstotliwość taktowania nie jest, aż tak „dokuczliwa".

Wróćmy zatem do schematu. Poza wspomnianymi diodami LED mikrokontroler steruje pracą zegara czasu rzeczywistego pod postacią układu MCP7940N produkcji firmy Microchip dokonuje tego dzięki programowej realizacji interfejsu I²C oraz realizuje obsługę graficznego, refleksyjnego wyświetlacza LCD o rozdzielczości 128×64 piksele wyposażonego w sterownik ekranu ST7565R firmy Sitronix, z którym komunikacja odbywa się dzięki programowej realizacji interfejsu SPI.

Uważny Czytelnik dostrzeże z pewnością, że interfejs komunikacyjny diod LED RGB podłączony został do jednego z wyprowadzeń interfejsu komunikacyjnego naszego wyświetlacza LCD (dokładnie do wyprowadzenia RS, czyli Data/Command), przez co sygnały sterujące pracą wyświetlacza mogłyby potencjalnie wpływać na stan diod LED i na odwrót. Generalnie mogłoby się tak dziać, ale interfejs diod celowo podłączono do wyprowadzenia RS, a nie żadnego innego wyprowadzenia w ramach interfejsu wyświetlacza LCD, gdyż przebiegi czasowe na nim występujące (chodzi głownie o czasy trwania stanu wysokiego) są na tyle powolne (co najmniej rząd wielkości większe), że nie zostaną rozpoznane przez logikę diody jako „ważne" dane, co najwyżej zostaną rozpoznane jako sygnał RESET, a to z kolei nie zaburza działania całego systemu.

Niemniej jednak zastanowicie się zapewne, dlaczego w ogóle powiązano te interfejsy? Odpowiedź jest bajecznie prosta… gdyż zabrakło wolnych wyprowadzeń mikrokontrolera. Oczywiście można by zastosować mikrokontroler o większej liczbie wyprowadzeń, ale po co, skoro problem można rozwiązać w ten prosty sposób?

Wróćmy zatem do samego wyświetlacza. Zastosowałem typ refleksyjny o bardzo dobrej widoczności w świetle słonecznym, lecz pozbawiony podświetlenia, gdyż zależało mi na oszczędności energii. Poza wspomnianymi peryferiami mikrokontroler nasz odpowiada za obsługę 4 przycisków funkcyjnych umownie oznaczonych jako UP, DOWN, NEXT i PREV, wykorzystując w tym celu przerwanie od porównania licznika Timer0 (wyzwalane 100 razy na sekundę), przez co możliwa stała się detekcja krótkiego i długiego naciśnięcia przycisku bez blokowania programu obsługi aplikacji, oraz odpowiada za obsługę wspomnianego wcześniej buzzera piezoelektrycznego.

Warto już w tej chwili pochylić się nad wspomnianym buzzerem. Uważny Czytelnik dostrzeże na pewno, że podłączono go do portu RESET mikrokontrolera i jak się zapewne domyślacie, nie bez powodu. A powód był znów tak samo trywialny, zwyczajnie nie miałem już wolnych portów, do których mógłbym podłączyć ten element, więc wybrałem ostatni, dostępny port…RESET.

Użycie wyprowadzenia RESET jako zwykłego portu I/O umożliwia nam stosowny bit konfiguracyjny dostępny w przestrzeni fuse-bitów (dokładnie bit RSTDISBL). Niestety ustawienie tegoż bitu (a dokładnie jego wyzerowanie) uniemożliwia dalsze programowanie mikrokontrolera z użyciem zwykłego programatora szeregowego, a jedyną możliwością w takim stanie rzeczy jest wykorzystanie programatora HV (wysokonapięciowego). Niesie to za sobą konsekwencję w postaci odpowiedniej kolejności programowania.

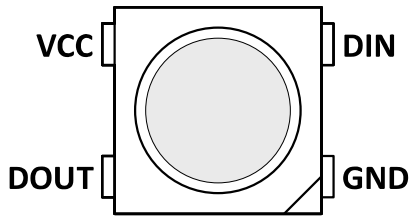
Otóż, i czego zapewne się domyślacie, w pierwszym kroku należy zaprogramować pamięć flash mikrokontrolera, a dopiero później ustawić bity konfiguracyjne. Brak ustawienia bitu RSTDISBL uniemożliwi obsługę buzzera piezoelektrycznego, co czasami może być wręcz pożądane (na przykład w przypadku, gdy nie planujemy używać tego rodzaju sygnalizacji). Z kolei, aby w ogóle móc zaprogramować mikrokontroler, konieczny jest demontaż buzzera (a przynajmniej odłączenie jego wyprowadzenia podłączonego do sygnału RESET), gdyż obecny na tej linii będzie zwierał sygnał RESET do masy zasilania, uniemożliwiając start programu obsługi aplikacji. Właśnie na ten problem natknąłem się, uruchamiając urządzenie i choć wydawał się oczywisty, to nie tak łatwo na początku było skojarzyć niezaprzeczalne fakty.

Wracając jeszcze do schematu urządzenia, wspomnieć należy o podtrzymaniu (choć niewielkim) zasilania zegara czasu rzeczywistego w postaci kondensatora C5 i diody D1, które zapewniają mu podtrzymanie zasilania na czas wymiany baterii zasilającej urządzenie.

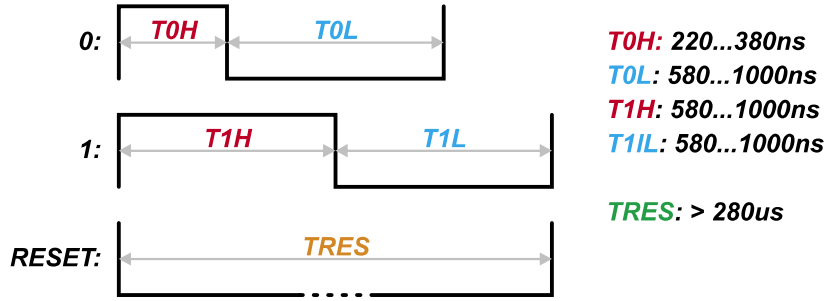
To byłoby tyle w kwestii sprzętu, ale poruszę jeszcze kwestię zastosowanych diod LED. Jako że założyłem, że każda z „przegródek" będzie podświetlana diodą LED, która może zmieniać kolor, konieczne było zastosowanie elementów typu RGB. W tym miejscu stanąłem przed wyzwaniem wyboru odpowiednich elementów wykonawczych, a więc sterownika diod LED, jak i samych diod. Jak wiemy, aby płynnie sterować kolorem diody LED typu RGB, należy zastosować 3-kanałowy sterownik PWM. Wynika z tego, że skoro przewiduję zastosowanie 16 elementów tego typu, to liczba niezbędnych kanałów wzrasta nam do 48. Trudno wyobrazić sobie mikrokontroler, który sprostałby tym wymaganiom, a jeszcze trudniej wyobrazić sobie projekt płytki drukowanej o niewielkiej wielkości, na której upakowalibyśmy tyle odrębnych ścieżek. Co oczywiste, można byłoby zastosować jakiegoś rodzaju sterowanie matrycowe, by ograniczyć liczbę koniecznych połączeń, ale biorąc pod uwagę liczbę wymaganych kanałów PWM i oczekiwaną rozdzielczość takiego sygnału (8 bitów), trudno mi sobie wyobrazić efektywne sterowanie bez użycia dość dużych prądów, by uzyskać wynikową jasność na akceptowalnym poziomie. Zresztą nawet w takim przypadku nie rozwiązujemy problemu ze skomplikowaniem rysunku obwodu drukowanego. Pat? Otóż nie.

Adresowalne diody LED RGB

Dość szybko zdałem sobie sprawę, że jedynym sensownym rozwiązaniem tego rodzaju problemu konstrukcyjnego będzie zastosowanie adresowalnych diod LED RGB, których konstrukcja pozwala na uniknięcie wszystkich wspomnianych problemów. Diody takie, oprócz wyprowadzeń zasilających, wyposażone są w jakiś szeregowy interfejs



Rysunek 2. Wygląd obudowy diody typu WS2812B-V5 z zaznaczeniem nazw wyprowadzeń



Rysunek 3. Przebiegi sygnałów interfejsu komunikacyjnego diody WS2812B-V5 w trakcie transmisji bitu logicznej „1", logicznego „0" i sygnału RESET

komunikacyjny, przy użyciu którego dokonujemy ustawień koloru jej świecenia. Interfejs, o którym mowa, zaimplementowany jest w taki sposób (zarówno w kwestii sprzętowej, jak i logicznej), że diody, takie połączone w łańcuchy, mogą być indywidualnie adresowane, a co za tym idzie, każda z nich ma niezależne sterowanie. Sam przebieg PWM niezbędny do regulacji koloru jej świecenia generowany jest sprzętowo dzięki sterownikowi zabudowanemu „na pokładzie" takiego elementu.

Przejdźmy zatem do konkretów. Pierwszą myślą, jaka przyszła mi w tym czasie do głowy, było zastosowanie bardzo popularnych i tanich elementów tego typu pod postacią diod z rodziny WS2811/WS2812. I właśnie tak zrobiłem, stosując diody typu WS2812B-V5 produkcji firmy Worldsemi, które mają tę dodatkową zaletę, że nie wymagają stosowania kondensatorów odsprzęgających zasilanie. Diody te produkowane są w niewielkich 4-wyprowadzeniowych obudowach SMD typu 5050 o wymiarach 5×5 mm, które idealnie wpisują się w założenia naszego projektu. Dioda tego rodzaju wyposażona jest w 4 wyprowadzenia:

- wyprowadzenia zasilające: GND i VCC,
- wejście asynchronicznego interfejsu komunikacyjnego DIN,
- wyjście asynchronicznego interfejsu komunikacyjnego DOUT.

Wygląd obudowy diody typu WS2812B-V5 z zaznaczeniem nazw wyprowadzeń pokazano na **rysunku 2**. Jak zapewne się domyślacie, diody typu WS2812B-V5 łączy się w łańcuchy, łącząc wyjścia interfejsu komunikacyjnego diody bieżącej z wejściami interfejsu komunikacyjnego diody kolejnej i tak dalej. Wejście interfejsu komunikacyjnego diody pierwszej, co oczywiste, łączy się z wyjściem tegoż interfejsu w mikrokontrolerze, zaś sama konstrukcja ramek danych i sposób działania sterownika „zabudowanego" w strukturze diody zapewnia odpowiednią synchronizację transmisji i niezbędne adresowanie.

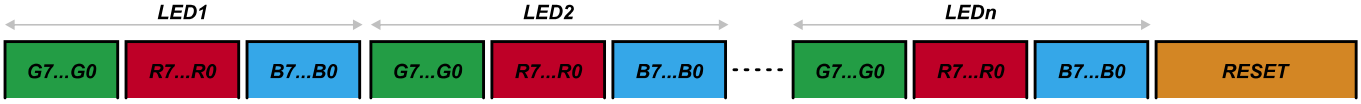
Zacznijmy więc od podstaw. Jak już wspomniałem, mamy do czynienia z interfejsem asynchronicznym, gdzie nie ma wyprowadzenia sygnału zegarowego, w związku z czym dane przesyłane przy jego użyciu muszą być w pewien sposób zakodowane, by możliwe stało się ich proste zdekodowanie i by były one odporne na zakłócenia i artefakty. W rozwiązaniu firmy Worldsemi zastosowano mechanizmy dobrze znane z interfejsów bezprzewodowej transmisji danych, stosowanych w torach podczerwieni, gdzie stany logiczne „1" i „0" zakodowane zostały długością impulsu. Dodatkowo wprowadzono tak zwany sygnał RESET (również zakodowany długością impulsu), który powoduje zresetowanie interfejsów komunikacyjnych sterowników diod LED i ich oczekiwanie na nowe dane. Na **rysunku 3**. pokazano przebiegi sygnałów interfejsu komunikacyjnego diody WS2812B-V5 w trakcie transmisji bitu logicznej „1", logicznego „0" i sygnału RESET. Co ważne, pojedyncze bity danych zgrupowane w bajty danych przesyłane są w kolejności od bitu najstarszego (MSB) do najmłodszego (LSB), a każda dioda LED w łańcuchu oczekuje na 3 bajty danych odpowiedzialnych za składowe jej koloru przesyłane w kolejności G, R, B.

W tym miejscu zapewne zadacie sobie pytanie, skąd każda z diod w łańcuchu „wie", które z przesyłanych danych użytecznych przeznaczone są właśnie dla niej, a nie dla innej? Zrealizowano to w bardzo prosty, acz skuteczny sposób. Każda z diod LED w łańcuchu po włączeniu zasilania (jak również po odebraniu sygnału RESET) oczekuje na 3 bajty danych przeznaczonych wyłącznie dla niej. Do tego czasu jej wyjściowy interfejs komunikacyjny (wyjście DOUT) jest „nieprzezroczysty" dla nadchodzących danych (a dokładnie rzecz biorąc, jest „przezroczysty" wyłącznie dla sygnału RESET). Po odebraniu wspomnianych 3 bajtów danych dioda ta staje się „przezroczysta" dla kolejnych nadchodzących danych, co znaczy ni mniej, ni więcej, że retransmituje je do kolejnych diod w łańcuchu (z małym opóźnieniem rzędu 300 ns). Biorąc pod uwagę, że dokładnie tak samo zachowuje się każda dioda w łańcuchu, dość szybko zdamy sobie sprawę, że kolejne dane użyteczne przesyłane przez tak skonstruowany interfejs komunikacyjny trafiają kolejno do następujących po sobie (w sensie elektrycznym) diod w łańcuchu.

Skuteczne i zarazem genialne w swojej prostocie, nieprawdaż? Niemniej jednak, co widać na **rysunku 4** prezentującym kompletną ramkę transmisji łańcucha diod tego typu, przyjęty sposób transmisji stanowiący podstawę „adresowania" poszczególnych diod LED w łańcuchu powoduje, że nie da się „zaadresować" (czyli przesłać do niej danych) na przykład diody czwartej w łańcuchu bez przesłania wcześniejszych (i zdawałoby się niepotrzebnych w tym momencie) danych dla diody trzeciej, drugiej i pierwszej. Mimo tego ograniczenia jest to rozwiązanie bardzo wygodne i chętnie stosowane przez producentów wszelkiego rodzaju peryferiów o takim przeznaczeniu.

Zasilanie i pobór prądu

I na tym zakończyłbym opis naszych ciekawych diod LED, gdyby nie jeden szczegół, który kompletnie demoluje założenia projektowe urządzenia, a wynika z niekompletności dokumentacji producenta (!). Otóż, jak pamiętacie, urządzenie e-Nurse standardowo, gdy nie ma żadnych aktywnych alarmów, przebywać będzie w trybie uśpienia o niskim poborze energii, podczas którego diody te pozostają wyłączone (wygaszone). Wszystko wydaje się OK, nieprawdaż? Gdyby właśnie nie ten „szczegół", którym producent peryferium specjalnie się „chwali" (w sumie nic dziwnego, gdyż nie jest to powód do dumy). Otóż dioda LED tego rodzaju, nawet jak jest wygaszona pobiera ze źródła zasilania bardzo duży (jak na jej stan pracy) prąd rzędu aż 0,7 mA, co przy 16 zastosowanych diodach daje nam aż 10 mA! I to w stanie czuwania!



Rysunek 4. Kompletna ramka transmisji łańcucha diod WS2812B-V5

Wykaz elementów:

Rezystory:
R1, R2: 4,7 kΩ (miniaturowy 1/8 W, raster 0,2")
R3: 976 kΩ 1% (SMD0805)
R4: 470 kΩ 1% (SMD0805)

Kondensatory: (ceramiczne)
C1, C2: 100 nF (THT, raster 0,1")
C3, C4: 12 pF (THT, raster 0,1")
C5: 1 µF (THT, raster 0,1")
C6...C8: 10 µF X7R (SMD0805)
C9...C17: 1 µF (THT, raster 0,1")

Półprzewodniki:
D1: BAT85 (DO34-7)
U1: MCP7940N-I/P (DIL8)
U2: ATtiny84 (DIL14)
U3: MCP1640T-I/CHY (SOT23-6)
U4: MCP1700T-3302E/TT (SOT23)
LED1...LED16: dioda adresowalna LED RGB typu WS2812B-V5 (SMD5050/PLCC4)
LCD: wyświetlacz graficzny typu LCD-AG-C128064CF-FGN NO/-E6 (rozdzielczość 128×64, sterownik ST7565R)

Pozostałe:
BUZZ: buzzer piezoelektryczny z wbudowanym generatorem (raster 0,3")
BATT: gniazdo baterii AAA typu KEYSTONE 1021
L1: dławik drutowy SMD 4.7 µH typu WLPN303015M4R7PB (SMD 3×3 mm)
Q1: rezonator kwarcowy zegarkowy 32768 Hz
PREV, NEXT, UP, DOWN: microswitch TACT (wysokość 13 mm)

Ustawienia fuse-bitów:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 1
CKOUT: 1
DWEN: 1
EESAVE: 0
RSTDISBL: 0(*)
(*): szczegóły w tekście artykułu

Tak duży prąd zniweczyłby korzyść z wprowadzania systemu mikroprocesorowego (i pozostałych peryferiów) w stan uśpienia, dla którego prąd spoczynkowy jest kilkaset razy mniejszy. Taki prąd dość szybko rozładowałby baterię AAA będącą źródłem napięcia zasilającego urządzenie. Jak poradzić sobie z takim nieoczekiwanym problemem? Postanowiłem wyłączać zasilanie diod LED w stanie uśpienia systemu za pomocą dodatkowego tranzystora MOSFET. Skuteczne a zarazem proste! Jedyny minus, że po włączeniu zasilania należy odczekać pewien (znowu brak o tym mowy w dokumentacji) czas niezbędny na „rozruch" interfejsu komunikacyjnego diody i dopiero po tym czasie można nawiązać z nią komunikację. Proste? Tyle, jeśli chodzi o szczegóły dotyczące naszych podstawowych elementów wykonawczych zaangażowanych w projekt urządzenia e-Nurse.

Kilka słów uwagi muszę poświęcić jeszcze sekcji zasilania. Jako że założyłem, że do zasilania urządzenia zastosowana będzie zwykła i tania bateria AAA o napięciu znamionowym 1,5 V (najlepiej alkaliczna z uwagi na jej pojemność), konieczne było użycie nowoczesnej przetwornicy step-up, która podwyższy napięcie ogniwa do minimum 3,7 V wymaganych dla diod LED i będzie odznaczać się niewielkim prądem spoczynkowym, by nie obciążać dodatkowo źródła zasilania w przypadku uśpienia systemu mikroprocesorowego. Co więcej, zastosowany rodzaj wyświetlacza wymaga z kolei napięcia zasilania 3,3 V, co spowodowało konieczność wykorzystania niewielkiego stabilizatora LDO obniżającego napięcie 3,7 V do poziomu 3,3 V cechującego się, jak poprzednio, niewielkim prądem spoczynkowym.

W związku z tym wybrano nowoczesne elementy SMD z oferty firmy Microchip, a mianowicie przetwornicę DC/DC typu MCP1640T-I/CHY (standardowy prąd spoczynkowy 19 µA i napięcie wejściowe od 0,65 V) oraz stabilizator LDO typu MCP1700T-3302E/TT (prąd spoczynkowy 1,6 µA). Bez problemu możemy też zastosować wersję stabilizatora o napięciu wyjściowym 3,0 V.

Na koniec opisu naszego urządzenia nie sposób nie poruszyć problematyki zużycia energii naszego systemu mikroprocesorowego, a co za tym idzie, żywotności zastosowanego źródła napięcia zasilania w postaci baterii AAA o przeciętnej pojemności w granicach 1300 mAh. Aby ocenić, jak długo urządzenie e-Nurse pracować będzie na pojedynczej baterii AAA, należy poczynić pewne założenia, jak również zastanowić się, z jakich etapów składa się cykl jego pracy i jakiej wielkości prądy pobiera wtedy ze źródła napięcia zasilającego.

Przystępując do obliczeń, założono, że urządzenie e-Nurse będzie alarmowało użytkownika o konieczności przyjęcia leku 5 razy na dobę, zaś czas tego alarmu, czyli okres od wystąpienia alarmu do skasowania go przez użytkownika, będzie każdorazowo wynosił 15 minut. Założono ponadto, że alarmy, o których mowa powyżej, wystąpią każdorazowo dla połowy dostępnych przegródek (slotów), co pociąga za sobą zapalenie 8 diod LED, zaś powrót urządzenia do stanu czuwania (po skasowaniu tychże alarmów) następował będzie automatycznie po upływie 30 sekund bezczynności po stronie użytkownika. W związku z założeniami poczynionymi powyżej wyodrębniono następujące stany pracy urządzenia i odpowiadające im prądy pobierane ze źródła napięcia zasilania:

- Stan alarmowania trwający 75 minut (5×15 minut), podczas którego diody LED zapalone są jedynie przez okres 1/7 długości tegoż stanu (gdyż takie jest wypełnienie sygnału sterującego ich miganiem) i pobierają wtedy ze źródła napięcia zasilania prąd o wartości 55 mA;
- Stan bezczynności systemu mikroprocesorowego, na który składa się 6/7 okresu stanu alarmowania (czyli około 64 minut, wtedy, gdy diody nie świecą) oraz 2,5 minuty bezczynności (5×30 sekund) przed przejściem systemu w stan uśpienia, podczas którego prąd pobierany ze źródła napięcia zasilania wynosi około 10,5 mA;
- Stan uśpienia, który trwa z dużym przybliżeniem 22,75 h/dobę i podczas którego pobierany jest prąd rzędu 80 µA (większość tego prądu to prąd spoczynkowy przetwornicy, stabilizatora oraz prąd pobierany przez logikę wyświetlacza LCD).

Przy założeniach jak wyżej otrzymano teoretyczny, 57-dniowy czas pracy na pojedynczej baterii AAA, co wydaje się wartością satysfakcjonującą, biorąc pod uwagę użyteczność urządzenia. Tyle w kwestii schematu, w związku z czym przejdźmy do zagadnień implementacyjnych, które ograniczają się do obsługi diod LED RGB.

Kolejna część opisu zostanie zaprezentowana za miesiąc. Omówimy wtedy działanie programu sterującego oraz montaż i obsługę urządzenia.

Robert Wołgajew, EP

W ofercie AVT*
AVT6016

Podstawowe parametry:
- liczba przegródek: 16,
- maksymalna liczba dziennych dawek: 5,
- źródło napięcia zasilającego: bateria alkaliczna typu AAA,
- prąd pobierany ze źródła zasilania (maksymalny/tryb uśpienia): 55 mA/80 µA (szczegóły w tekście artykułu).

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media
AVT5541 Dozownik detergentu, czyli czas na elektronizację WC (EP 6/2016)

* **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**.

Autor składa podziękowania firmie Artronic za dostarczenie wyświetlacza LCD zastosowanego w projekcie urządzenia e-Nurse.

e-Nurse
– elektroniczny dyspenser leków (2)

W poprzednim numerze EP zaprezentowaliśmy konstrukcję układu elektronicznego oraz szczegóły dotyczące sterowania i zasilania adresowalnych diod LED, stanowiących prosty i intuicyjny interfejs użytkownika dyspensera e-Nurse. Tym razem zajmiemy się architekturą programu sterującego oraz opiszemy sposób montażu i obsługi urządzenia. Dla Czytelników zainteresowanych odtworzeniem urządzenia dla siebie lub bliskiej osoby udostępniamy także szczegóły dotyczące drukowalnego panelu przegródek, opracowanego przez autora na potrzeby projektu.

Program sterujący

Zaczniemy od pliku nagłówkowego do obsługi diod LED RGB upraszczającego i porządkującego późniejszy kod, który to plik pokazano na **listingu 1**.

Dalej, na **listingu 2**, pokazano ciało funkcji odpowiedzialnej za przesłanie jednego bitu danych przy udziale wspomnianego wcześniej asynchronicznego interfejsu komunikacyjnego. Warto zauważyć, że stosowną funkcję zdefiniowano używając atrybutu *__always_inline__*, który instruuje kompilator by ZAWSZE wstawiał rozwinięcie funkcji w miejscu wywołania, nie zaś skok do jej ciała. Jest to o tyle istotne, że operujemy na dość rygorystycznych timingach rzędu setek nanosekund, gdzie, przy częstotliwości taktowania mikrokontrolera rzędu 8 MHz (okres 125 ns), niepotrzebne (z punktu widzenia programisty, nie kompilatora) wykonanie jakiegoś rozkazu asemblera może „rozłożyć" całą transmisję danych uniemożliwiając jakiekolwiek sterowanie wspomnianymi elementami.

Skoro mamy już funkcję umożliwiającą przesłanie jednego bitu informacji to pora na funkcję (także odpowiednio zadeklarowaną) umożliwiającą przesłanie kompletnego bajta danych, przy udziale wspomnianego wcześniej interfejsu danych, której to ciało pokazano na **listingu 3**.

Listing 1. Plik nagłówkowy modułu obsługi diod WS2812B-V5

```
#define LED_PORT PORTA
#define LED_DDR DDRA
#define LED_PIN_NR PA3
#define LED_SET LED_PORT |= (1<<LED_PIN_NR)
#define LED_RESET LED_PORT &= ~(1<<LED_PIN_NR)
#define LED_PIN_AS_OUTPUT LED_DDR |= (1<<LED_PIN_NR)
```

Dalej, by uprościć zapis danych do adresowalnych diod LED RGB, przewidziano funkcję pozwalającą na przesłanie kompletnej informacji o kolorze tejże diody LED, której to ciało pokazano na **listingu 4**.

Zapewne pamiętacie, że asynchroniczny interfejs komunikacyjny diod LED typu WS2812B-V5 przewiduje transmisję jeszcze jednego sygnału, a mianowicie sygnału RESET. Ciało funkcji odpowiedzialnej za transmisję sygnału RESET interfejsu diod WS2812B-V5 pokazano na **listingu 5**.

Normalnie w tym miejscu zakończyłbym opis obsługi naszych ciekawych diod LED RGB, ale nie tym razem. Prawdę mówiąc funkcje pokazane powyżej będą działać bez problemów wyłącznie na szybszych mikroprocesorach, dla których jeden cykl zegarowy jest najlepiej o rząd

Listing 2. Funkcja odpowiedzialna za przesłanie jednego bitu danych interfejsu diod WS2812B-V5

```
__inline__ void __attribute__((__always_inline__))
    ledSendBit(uint8_t Bit) {
    if(Bit) {
        LED_SET;
        _delay_us(0.65);
        LED_RESET;
        _delay_us(0.65);
    } else {
        LED_SET;
        _delay_us(0.25);
        LED_RESET;
        _delay_us(0.65);
    }
}
```

wielkości krótszy, niż timingi odpowiedzialne za wysłanie poszczególnych bitów koloru diod LED. W innym przypadku, czyli jak u nas, gdzie takt zegara trwa 125 ns (dla częstotliwości taktowania 8 MHz) wykonanie każdego, dodatkowego rozkazu (ja na przykład sprawdzenie wartości każdego bitu zmiennej przeznaczonej do wysłania) może dość skutecznie „rozłożyć" całą transmisję danych, gdyż nie zachowane są wówczas rygorystyczne timingi kodujące bity informacji.

Listing 3. Funkcja odpowiedzialna za przesłanie jednego bajta danych interfejsu diod WS2812B-V5

```
__inline__ void __attribute__((__always_inline__))
    ledSendByte(uint8_t Byte){
    ledSendBit(Byte & 0x80);
    ledSendBit(Byte & 0x40);
    ledSendBit(Byte & 0x20);
    ledSendBit(Byte & 0x10);
    ledSendBit(Byte & 0x08);
    ledSendBit(Byte & 0x04);
    ledSendBit(Byte & 0x02);
    ledSendBit(Byte & 0x01);
}
```

Listing 4. Funkcja pozwalającą na przesłanie kompletnej informacji o kolorze diody LED typu WS2812B-V5

```
__inline__ void __attribute__((__always_inline__))
    ledSendColor(uint8_t R, uint8_t G, uint8_t B){
    ledSendByte(G);
    ledSendByte(R);
    ledSendByte(B);
}
```

Listing 5. Funkcja odpowiedzialna za transmisję sygnału RESET interfejsu diod WS2812B-V5

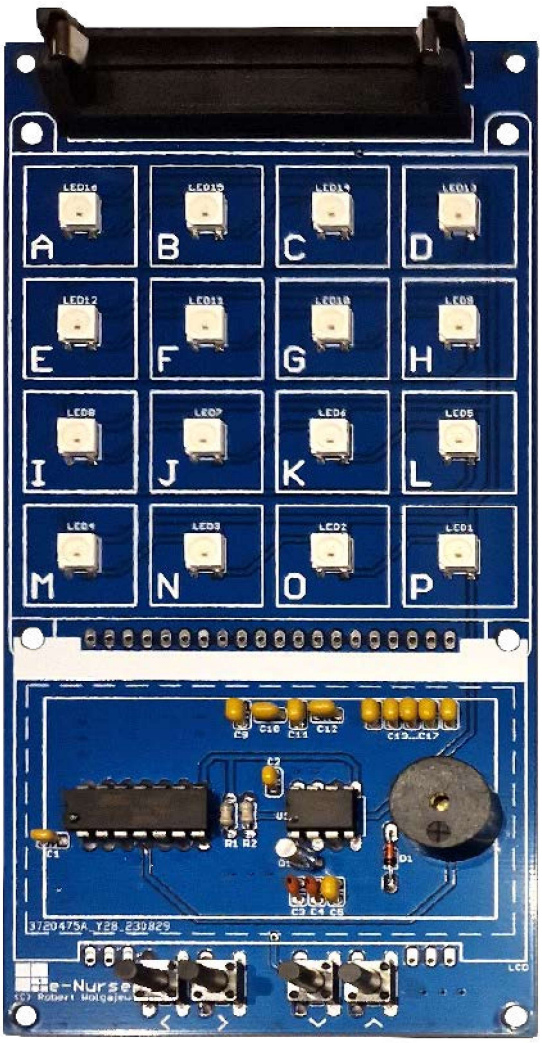
```
inline void ledReset(void){
    LED_RESET;
    _delay_us(300);
}
```

Listing 6. Uproszczone funkcje umożliwiające poprawną transmisję sygnałów sterujących interfejsu diod WS2812B-V5 dla mikrokontrolera taktowanego zegarem 8 MHz

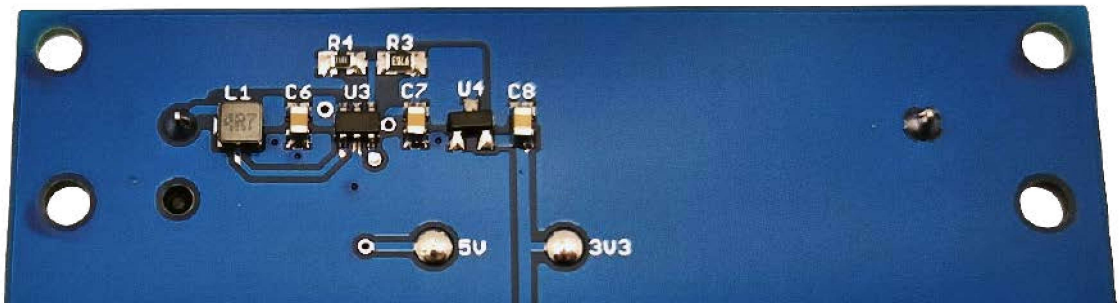
```
#define NOP() __asm__ __volatile__ („nop")

__inline__ void __attribute__((__always_inline__))
ledSendBit0(void) { //Dla zegara 8MHz (NOP = 125ns)
    LED_SET;
    NOP();
    LED_RESET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}

__inline__ void __attribute__((__always_inline__))
ledSendBit1(void) { //Dla zegara 8MHz (NOP = 125ns)
    LED_SET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    LED_RESET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}

__inline__ void __attribute__((__always_inline__))
ledRed(void) {
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();

    ledSendBit1();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();

    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
}
```

Jak sobie z tym poradzić? Dość prosto, choć nie nazbyt optymalnie. Ja utworzyłem osobne funkcje dla każdego rodzaju bitu (0 i 1) i każdego koloru, który zamierzałem przesyłać a wszelkie warunki zastąpiłem „gotowym" do wykonania kodem bez zbędnych (z punktu widzenia transmisji) rozkazów. Spowodowało to powstanie wielu funkcji o bardzo prostym (i w gruncie rzeczy podobnym) acz długim kodzie, który zachował niezbędne timingi. Te uproszczone funkcje, o których mowa powyżej, pokazałem na **listingu 6**, gdzie zamieszczono odrębne funkcje dla każdego z bitów informacji (0 i 1), jak i przykładową funkcję wysyłającą dane dla koloru czerwonego. Dzięki takiemu rozwiązaniu bez problemu możemy transmitować dane do naszych diod LED korzystając z mikrokontrolera taktowanego zegarem 8 MHz.

Oczywiście, i czego się zapewne domyślacie, w czasie takiej transmisji należy blokować możliwość obsługi przerwań systemowych. Można również poradzić sobie w prostszy sposób zwyczajnie zwiększając prędkość zegara taktującego mikrokontroler (najlepiej co najmniej dwukrotnie), lecz ja nie chciałem iść tą drogą, gdyż wydatnie zwiększa się wtedy zapotrzebowanie systemu mikroprocesorowego na energię. Inną sprawą jest to, że w przypadku naszego mikrokontrolera nie mam już wolnych portów I/O by podłączyć niezbędny rezonator kwarcowy, w związku z czym należałoby zastosować element o większej liczbie wyprowadzeń, co z kolei nie jest pożądane z uwagi na wymiary obwodu drukowanego. Kolejną możliwością jest użycie



Fotografia 1. Zmontowane urządzenie e-Nurse od strony warstwy TOP tuż przed przylutowaniem wyświetlacza LCD



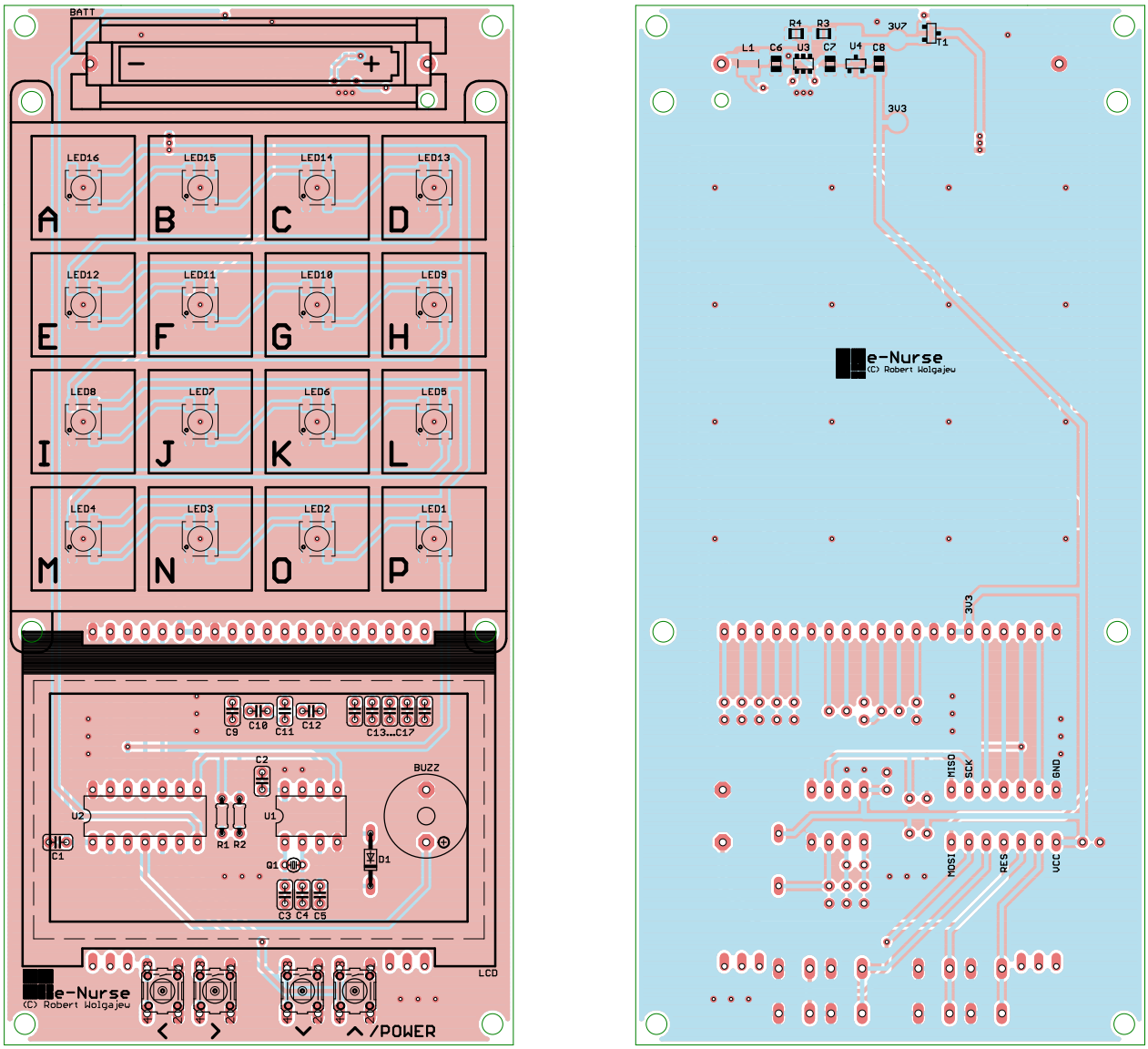
Fotografia 2. Zmontowane urządzenie e-Nurse od strony warstwy BOTTOM

jakiegoś sprzętowego, szeregowego medium transmisyjnego wbudowanego w mikrokontroler (USART lub SPI), ale i w takim przypadku nie uciekniemy od konieczności zwiększenia częstotliwości taktowania mikrokontrolera. Tyle w kwestiach implementacyjnych, w zawiązku z czym przejdźmy do zagadnień montażowych.

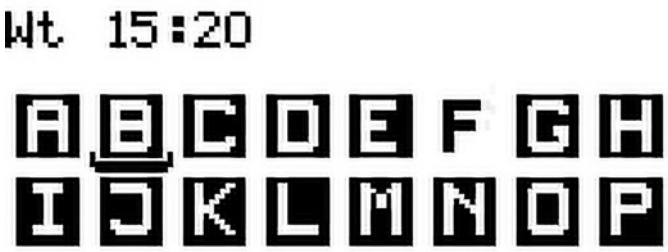
Montaż i uruchomienie

Schemat płytki PCB naszego urządzenia pokazano na **rysunku 5**. Jak widać zaprojektowano bardzo „zgrabną", dwustronną płytkę drukowaną ze zdecydowaną przewagą elementów THT montowanych po obu stronach laminatu (po stronie BOTTOM montowane są wyłącznie elementy SMD). Montaż urządzenia rozpoczynamy od warstwy BOTTOM, gdzie w pierwszej kolejności montujemy półprzewodniki, zaś w kolejnym kroku elementy bierne. W tym momencie przechodzimy na warstwę TOP, gdzie, podobnie jak poprzednio, w pierwszej kolejności montujemy diody LED RGB, następnie pozostałe elementy półprzewodnikowe (z wyłączeniem wyświetlacza) a na końcu elementy bierne (z wyłączeniem buzzer-a piezoelektrycznego) oraz elementy mechaniczne w postaci koszyczka baterii AAA czy przycisków funkcyjnych. Na koniec, do tak przygotowanego obwodu drukowanego, montujemy wyświetlacz LCD lutując jego wyprowadzenia w przeznaczone do tego celu punkty lutownicze, co zapewni mu zarówno niezbędne połączenia elektryczne, jak i pożądaną stabilizację mechaniczną.

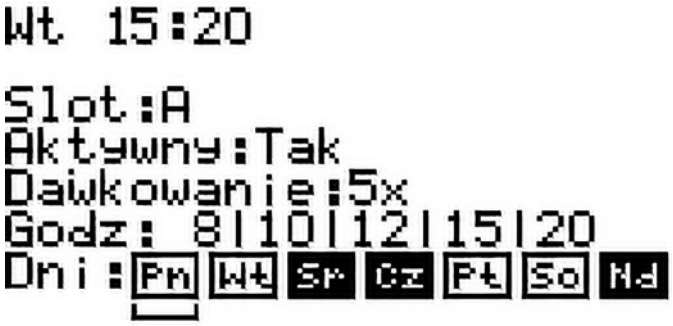
Na **fotografii 1** pokazano zmontowane urządzenie od strony warstwy TOP tuż przed przylutowaniem wyświetlacza LCD



Rysunek 5. Schemat montażowy urządzenia e-Nurse



Rysunek 6. Ekran Menu głównego urządzenia e-Nurse



Rysunek 7. Ekran Menu ustawień urządzenia e-Nurse

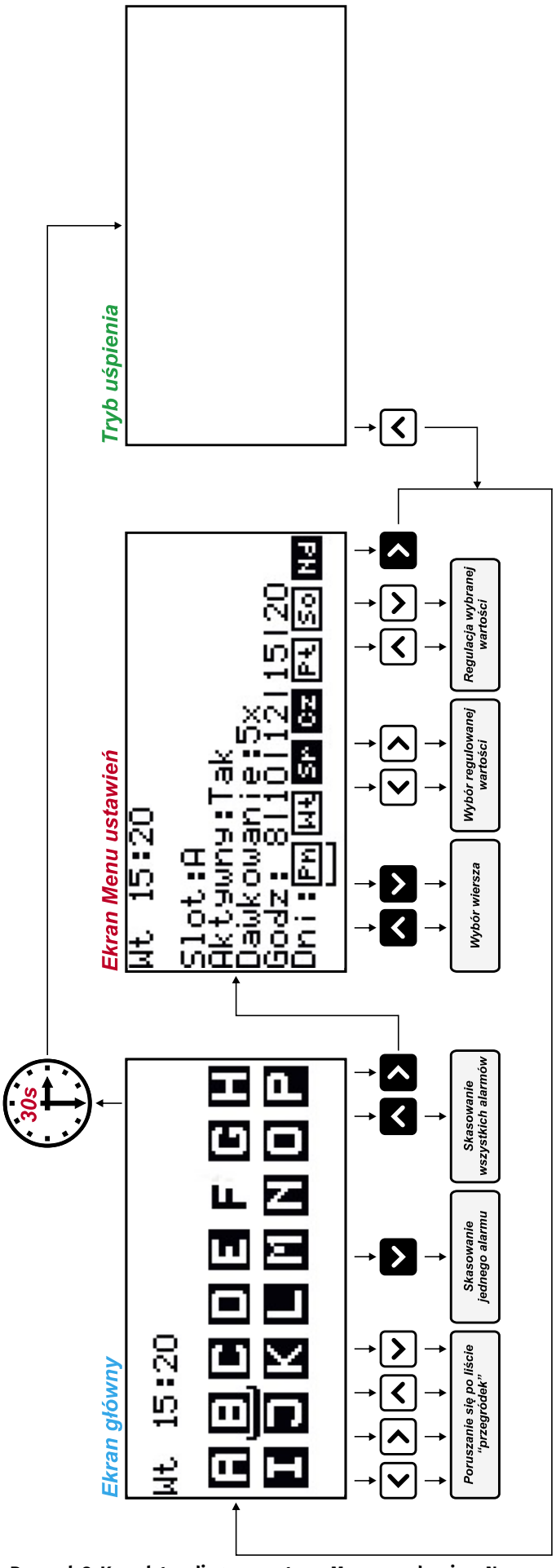
(wersja prototypowa, przed niewielkimi modyfikacjami), zaś na **fotografii 2** to samo urządzenie od strony warstwy BOTTOM (pokazano wyłącznie ten obszar obwodu drukowanego, gdzie montowane są elementy SMD i to przed niewielkimi modyfikacjami sekcji zasilania).

Obsługa

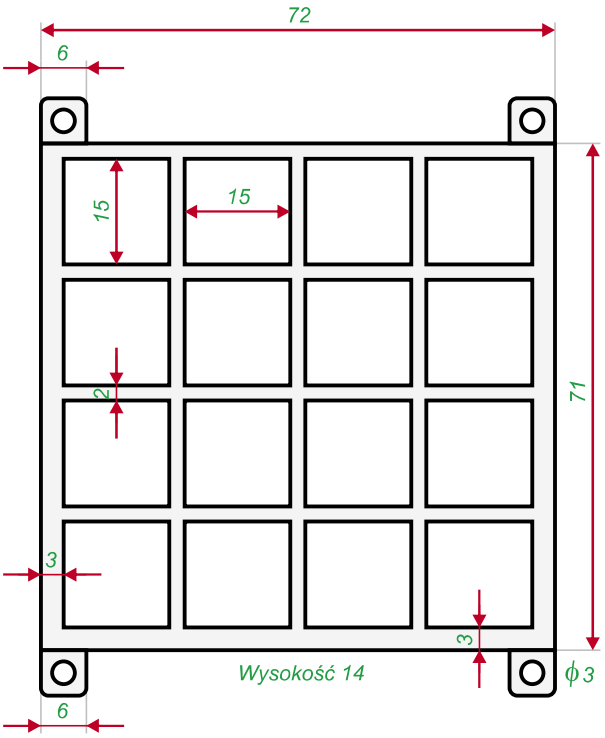
Projektując system Menu urządzenia e-Nurse oraz sposób jego obsługi przyjąłem, że ergonomia i prostota jego użytkowania jak i czytelność interfejsu użytkownika powinna być najważniejszym kryterium przy konstruowaniu stosownych procedur sterujących. Zgodnie z tymi podstawowymi założeniami, na płytce sterownika przewidziano wyłącznie 4 przyciski funkcyjne pozwalające na poruszanie się po dostępnych ekranach Menu. Mówiąc zupełnie ogólnikowo przyciski oznaczone jako NEXT, PREV (symbolizowane przez stosowne symbole strzałek) służą do zmiany typu edytowanego elementu, zaś przyciski oznaczone jako UP, DOWN (symbolizowane przez stosowne symbole strzałek) służą do zmiany wartości edytowanego elementu oraz poruszania się po kolejnych opcjach Menu urządzenia.

W ramach systemu Menu dostępne są 2 ekrany użytkownika: ekran Menu głównego oraz ekran Menu ustawień. Poniżej, na **rysunku 6**, pokazano ekran Menu głównego urządzenia e-Nurse, zaś na **rysunku 7** pokazano ekran Menu ustawień naszego urządzenia. Jak widać, w ramach Menu głównego dostępna jest informacja o bieżącej dacie i godzinie systemowej oraz informacja o aktywnych alarmach (aktywność alarmu pokazywana jest w postaci wypełnienia wybranego symbolu „przegródki"). Z tego poziomu dokonujemy potwierdzenia przyjęcia dawki leku, co skutkuje skasowaniem alarmu, posiłkując się przy tym przyciskami funkcyjnymi umożliwiającymi poruszanie się po liście ikonek alarmów (przesuwającymi podkreślenie konkretnego symbolu „przegródki") oraz ich kasowanie. Kasowanie może przebiegać w dwojaki sposób: długie przyciśnięcie przycisku DOWN kasuje jeden alarm dla danej „przegródki", zaś długie przyciśnięcie przycisku UP kasuje wszystkie alarmy dla tejże „przegródki". Dodatkowo, długie przyciśniecie przycisku NEXT wprowadza nas w tryb Menu ustawień.

Tryb Menu ustawień pozwala nam na dokonanie ustawień zarówno bieżącej godziny i dnia tygodnia, jak i skonfigurowanie ustawień dla każdej „przegródki" oznaczonej tu, jako „Slot". Ustawienia, o których mowa powyżej to: aktywność „przegródki" (gdyż można ją wstępnie skonfigurować, ale jeszcze nie aktywować), dzienną liczbę dawek



Rysunek 8. Kompletny diagram systemu Menu urządzenia e-Nurse wraz ze sposobem jego obsługi



Rysunek 9. Rysunek techniczny panelu „przegródek” (z zaznaczeniem ważniejszych wymiarów)

leku (1…5), godziny przyjmowania leku (dowolne z zakresu 0…23) oraz dni tygodnia przyjmowania leku. Poruszanie się pomiędzy elementami poddawanymi edycji, jak i ich edycję umożliwiają krótkie naciśnięcia przycisków funkcyjnych, zaś poruszanie się pomiędzy wierszami opcji Menu ustawień umożliwiają długie naciśnięcia przycisków UP i DOWN.

Wyjście z Menu ustawień i przejście do Menu głównego dokonywane jest poprzez długie naciśnięcie przycisku NEXT. Procesowi temu towarzyszy zapisanie bieżących ustawień urządzenia (ustawień „przegródek”) w nieulotnej pamięci EEPROM mikrokontrolera oraz aktualizacja czasu zegara RTC. Do czasu wyjścia z Menu ustawień niemożliwe jest automatyczne przejście urządzenia do stanu uśpienia w przypadku braku aktywnych alarmów. Z kolei, proces sprawdzania alarmów i sygnalizacja ich wystąpienia (przy pomocy

Rysunek 10. Model 3D panelu „przegródek”

wbudowanych diod LED RGB oraz buzzer-a) dokonywana jest również w czasie konfiguracji urządzenia (w Menu ustawień). Na **rysunku 8** pokazano kompletny diagram systemu Menu urządzenia e-Nurse wraz ze sposobem jego obsługi.

Obudowa

I na sam koniec obiecany „ekstras”. Gotowy panel obudowy „przegródek” wykonany w aplikacji do projektowania 3D przez mojego, zawsze niezawodnego, Kolegę Bartłomieja Wawrzyszko. Element ten zaprojektowany został w taki sposób by po jego przykręceniu do obwodu drukowanego (4 śruby ø3 mm maks.) stanowił kompletne rozwiązanie dozownika leków zbliżone swoim wyglądem do prostych kasetek na leki dostępnych w aptekach czy też drogeriach. Co więcej, w bocznych ściankach elementu (trzech z czterech) przygotowano specjalne wcięcia do wsunięcia w nie przeźroczystej pleksi o grubości do 1,5 mm i wymiarach 68×80 mm tak, by stanowiła ona pokrywę dozownika uniemożliwiającą wypadnięcie leków ze swoich „przegródek”. Prawda, że fajne?

Na **rysunku 9** pokazano rysunek techniczny panelu „przegródek” (z zaznaczeniem ważniejszych wymiarów), zaś na **rysunku 10** pokazano z kolei model 3D tegoż elementu. Plik produkcyjny do wykonania wyżej wymienionej obudowy udostępniony jest w materiałach dodatkowych do artykułu.

Robert Wołgajew, EP