Zegar Nixie

Listingi do projektu dostępne są na stronie www.media.avt.pl oraz w artykule na stronie www.ep.com.pl

Historia lamp Nixie sięga początku lat 50. Seryjna produkcja zaczęła się w roku 1956 i trwała do lat 90. XX w. Niegdyś stosowane w urządzeniach pomiarowych czy wojskowych lub jako wskaźniki, dziś są poszukiwane głównie do wykonania zegara cyfrowego. I chociaż w ostatnich latach powstało wiele różnych zegarów, w których czas pokazany był właśnie na tym rodzaju wyświetlacza, autor prezentowanego projektu pokusił się o skonstruowanie wersji całkowicie zautomatyzowanej.

Konstrukcja urządzenia została ukierunkowana tak, aby zegar wyświetlał bardzo dokładny czas. Po włączeniu zasilania aktualny czas zostaje pobrany z układu DS3231, który jest podtrzymywany dodatkową baterią. Co dwanaście godzin czas zapisany w układzie RTC jest synchronizowany z czasem pobranym z serwera NTP, za co odpowiada moduł ESP01.

Budowa i działanie

Schemat ideowy ma budowę modułową i składa się z czterech zasadniczych części. Pierwsza część to wyświetlacz, który wyświetla godzinę i minuty na czterech lampach Nixie IN4 (**rysunek 1a**). Drugą stanowi układ przetwornicy typu boost, podwyższającej napięcie, zbudowanej na bardzo popularnym timerze NE555 (**rysunek 1b**). Przetwornica w dużej mierze bazuje na projekcie pod tą samą nazwą, opisanym w książce pt. „100 projektów na 555", autorstwa Krzysztofa Górskiego. Trzecia część to moduł Wi-Fi ESP826601 (**rysunek 1c**), zaprogramowany w ten sposób, że jeśli jest w stanie normalnej pracy, to co jedną sekundę wysyła ramkę danych, zawierającą aktualną datę, godzinę oraz dzień tygodnia. Dane pobierane są z serwera NTP. Wreszcie czwarta część, czyli płyta główna, do której moduły są podłączone (**rysunek 1d i 1e**). Na płycie umieszczony jest procesor ATmega128 z popularnej rodziny AVR, odpowiedzialny za sterowanie pozostałymi modułami. Część cyfrowa projektu zasilana jest napięciem 3,3 V, a zasilanie przechodzi przez zworkę JP1. Poza procesorem na płycie znajduje się układ zegara czasu rzeczywistego DS3231, tranzystory sterujące katodami lamp oraz transoptory sterujące anodami lamp.

Układ DS3231 to bardzo dokładny zegar czasu rzeczywistego, komunikujący

Tabela 1. Opis wyprowadzeń modułu ESP826601

Pin	Opis
Vcc	Zasilanie napięciem 3,3 V, pobór prądu do 300 mA
GND	Masa układu
Tx	Nadajnik interfejsu szeregowego UART
Rx	Odbiornik interfejsu szeregowego UART
RST	Reset układu aktywowanego stanem niskim. W czasie normalnej pracy należy podać stan wysoki 3,3 V
CH_PD	Power down – podanie stanu niskiego powoduje przejście w tryb uśpienia Do aktualizacji oprogramowania oraz komunikacji przez UART należy podać stan wysoki 3,3 V
GPIO0	Wyprowadzenie GPIO nr 0 Do aktualizacji oprogramowania należy podać stan niski
GPIO2	Wyprowadzenie GPIO nr 2

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5812

Podstawowe parametry:
- odmierzanie czasu za pomocą specjalizowanego układu DS3231,
- automatyczne synchronizowanie zegara z czasem pobranym z serwera NTP,
- automatyczne przełączanie pomiędzy czasem letnim i zimowym,
- wyświetlacz na lampach Nixie typu IN4.

Projekty pokrewne na www.media.avt.pl:

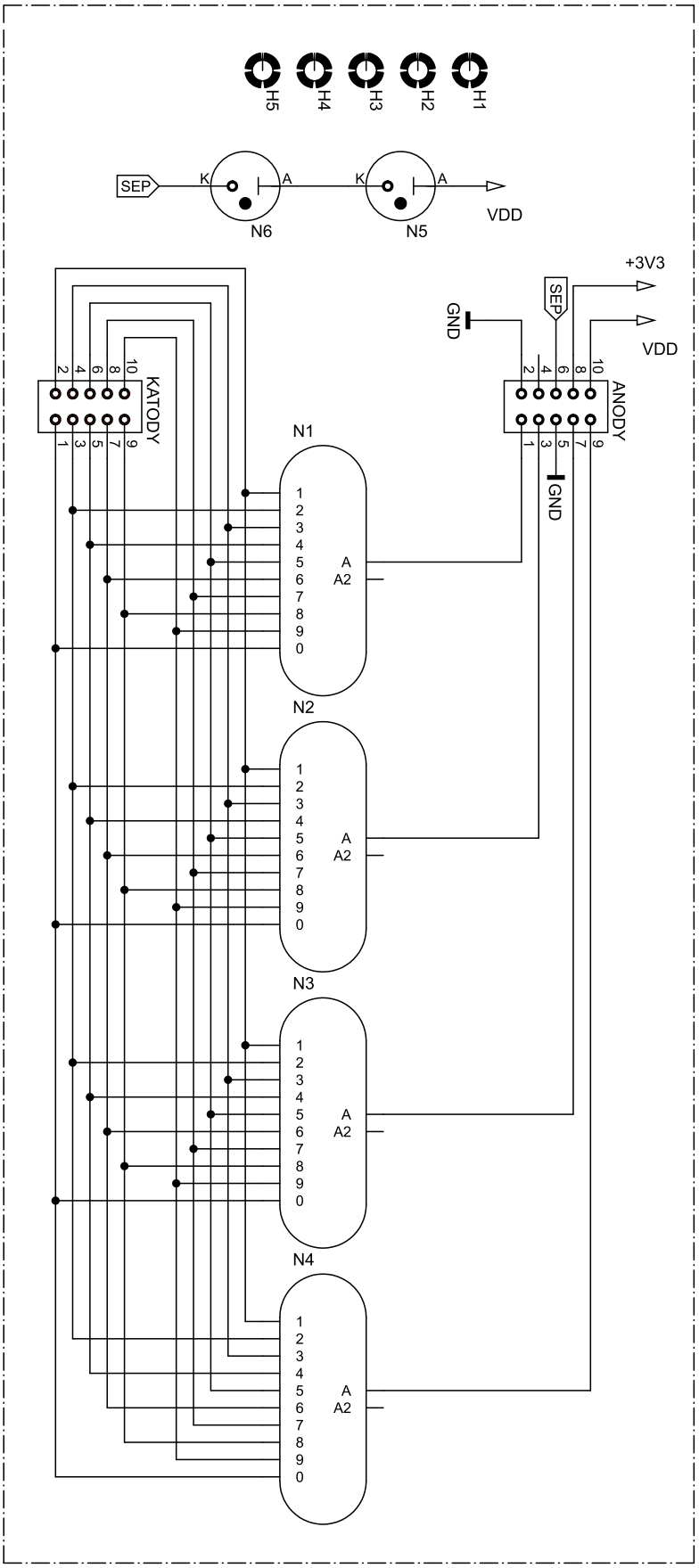
AVT-3226	Minizegar Nixie (EP 8/2018)
AVT-5582	Androidowy zegar Nixie (EP 5/2017, EP 8/2017)
AVT-3141	Zegar Nixie z sekundami (EdW 6/2016)
AVT-3097	Zegar Nixie (EdW 7/2014)
Projekt 210	Termometr pokojowy z lampami Nixie (EP 10/2013)
AVT-5390	Zegar w stylu retro z lampami Nixie typu Z570M/Z573M (EP 4/2013)
AVT-5145	Zegar retro na lampach Nixie (EP 9/2008)
AVT-521	Zegar Nixie dla oszczędnych (EP 8/2003)

* 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 http://sklep.avt.pl, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.

się z głównym procesorem poprzez magistralę I²C. W momencie braku zasilania (np. przerwa w dostawie energii elektrycznej) układ jest podtrzymywany poprzez zewnętrzną baterię lub kondensator o wysokiej pojemności. W przypadku zastosowania kondensatora dodatkowo należy przylutować diodę D2 oraz rezystor R13. Do układu może być podłączone jednocześnie tylko jedno źródło podtrzymujące napięcie. Stąd płytka zaprojektowana jest w taki sposób, że nie ma możliwości jednoczesnego przylutowania baterii i kondensatora.



Rysunek 1. Schemat elektryczny: a) moduł wyświetlacza

ESP826601 to moduł Wi-Fi działający w standardzie 802.11 b/g/n, na częstotliwości 2,4 GHz. Ma osiem wyprowadzeń w rastrze 2,54 mm oraz antenę umieszczoną na płycie PCB modułu. Jak pokazuje schemat, do wyprowadzeń modułu dołączone są rezystory R25...R28 umożliwiające wybranie odpowiedniego trybu pracy. Aby synchronizować czas co określone przedziały czasowe, należy przylutować tylko rezystory R27 oraz R28 (ustawienie domyślne). Jeżeli zamiast tych rezystorów zostaną zamontowane rezystory R25 i R26, moduł będzie na stałe połączony z siecią Wi-Fi. Nie ma konieczności montowania diody LED2. Warto nadmienić, że w czasie regularnej pracy pobór prądu przez ESP01 może wynosić nawet 300 mA. Procesor IC2 (ATmega128) komunikuje się z tym modułem poprzez interfejs UART, czyli sygnałami RX i TX (parametry komunikacji 115200, 8, n, 1). Opis wyprowadzeń modułu został pokazany w **tabeli 1** oraz na **rysunku 2**.

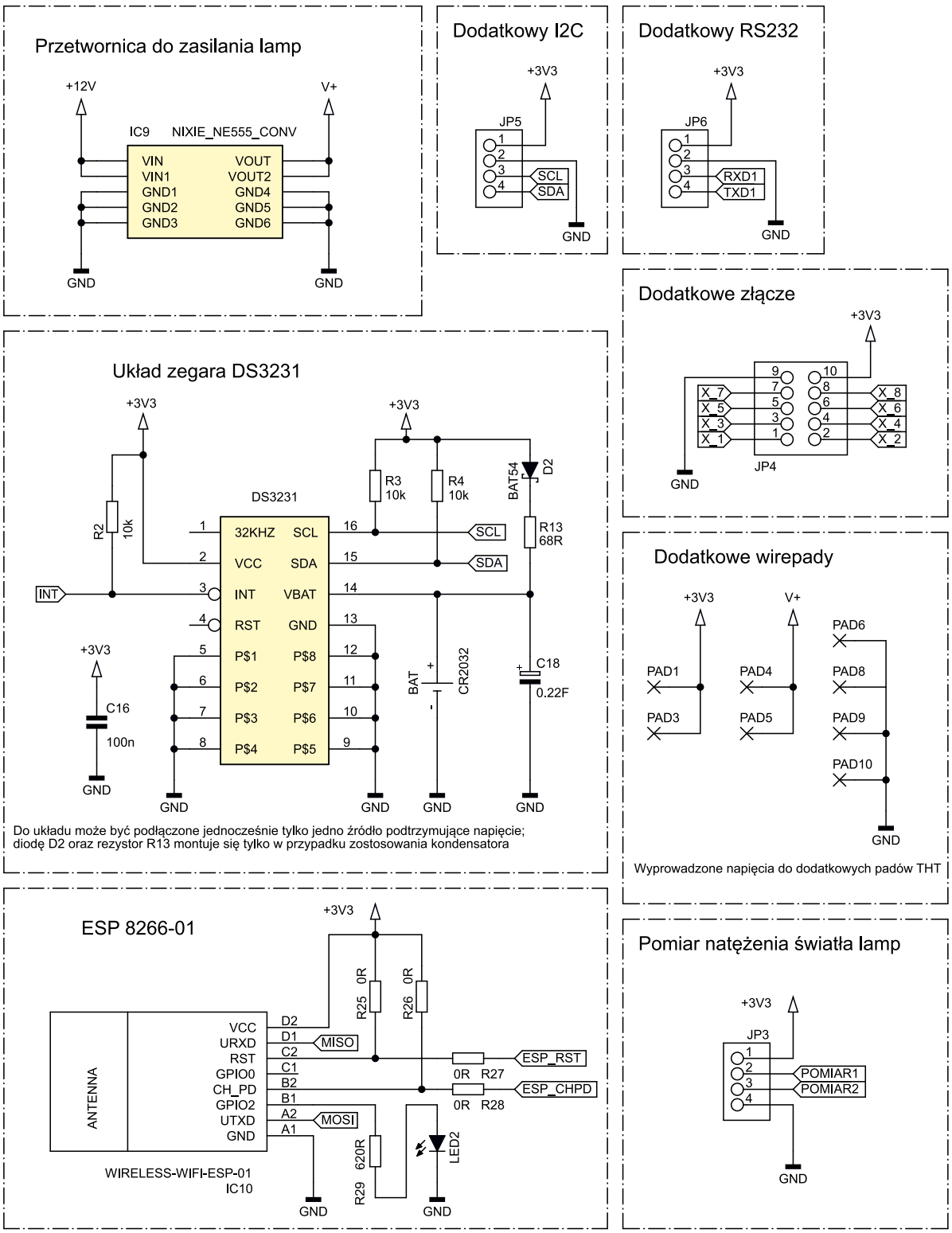
Przetwornica typu boost wytwarza wysokie napięcie odpowiednie dla anod lamp. Kluczowymi elementami są: tranzystor MOSFET, cewka o wartości 100 µH i prądzie co najmniej 1 A oraz dioda prostownicza. Działanie układu podwyższającego napięcie polega na cyklicznym załączaniu tranzystora. W momencie kiedy tranzystor jest włączony (**rysunek 3a**), na cewce L odkłada się napięcie stałe, uzyskane z napięcia wejściowego, a prąd płynący przez cewkę zaczyna narastać. Dioda jest spolaryzowana w kierunku zaporowym i nie przewodzi – jej anoda jest podłączona do masy układu. W chwili gdy tranzystor zostaje wyłączony (**rysunek 3b**), prąd cewki jest większy od zera. Strumień magnetyczny indukuje w cewce napięcie o takiej wartości, aby umożliwić dalszy przepływ prądu do obciążenia. Kiedy wartość prądu płynącego przez cewkę będzie równa zeru, dioda przestaje przewodzić, więc energia zgromadzona w cewce nie będzie przekazywana

do obciążenia. Ten stan potrwa do czasu ponownego załączenia tranzystora.

Oprogramowanie modułu ESP

Kod odpowiedzialny za pobieranie danych z serwera NTP (*Network Time Protocol*) i wysłanie ich na port szeregowy został pokazany na **listingu 1**. Program napisano w języku C, w środowisku Arduino. W dużej mierze przemawiała za tym łatwość w dostępie do bibliotek oraz szybki sposób przygotowania środowiska do programowania konkretnego modułu. Na samym początku należy podać nazwę sieci Wi-Fi oraz hasło do niej. W zależności od obecnie obowiązującego czasu (letni/zimowy) należy odkomentować odpowiednią linię ze zmienną *time_zone*. Wartość 3600 (czas zimowy) lub 7200 (czas letni) oznacza różnicę w milisekundach od czasu wzorcowego UTC. Kolejnym elementem kodu są deklaracje zmiennych przechowujących czas i datę. Są to zmienne typu string oraz int. Potem ustawiana jest odpowiednia prędkość interfejsu UART i połączenie z siecią Wi-Fi.

W pętli głównej programu już w pierwszych liniach kodu uzyskujemy interesujące nas informacje, tj. datę oraz godzinę. W tak podstawowej formie można by już to zostawić, ale chcąc w pełni zautomatyzować urządzenie, trzeba uwzględnić zmianę czasu z zimowego na letni w marcu i letniego na zimowy w październiku. Zmiana czasu odbywa się w niedzielę, więc trzeba uwzględnić także dzień tygodnia. Zostanie on obliczony, ale w pierwszej kolejności z odebranego łańcucha tekstowego należy wyselekcjonować interesujące nas



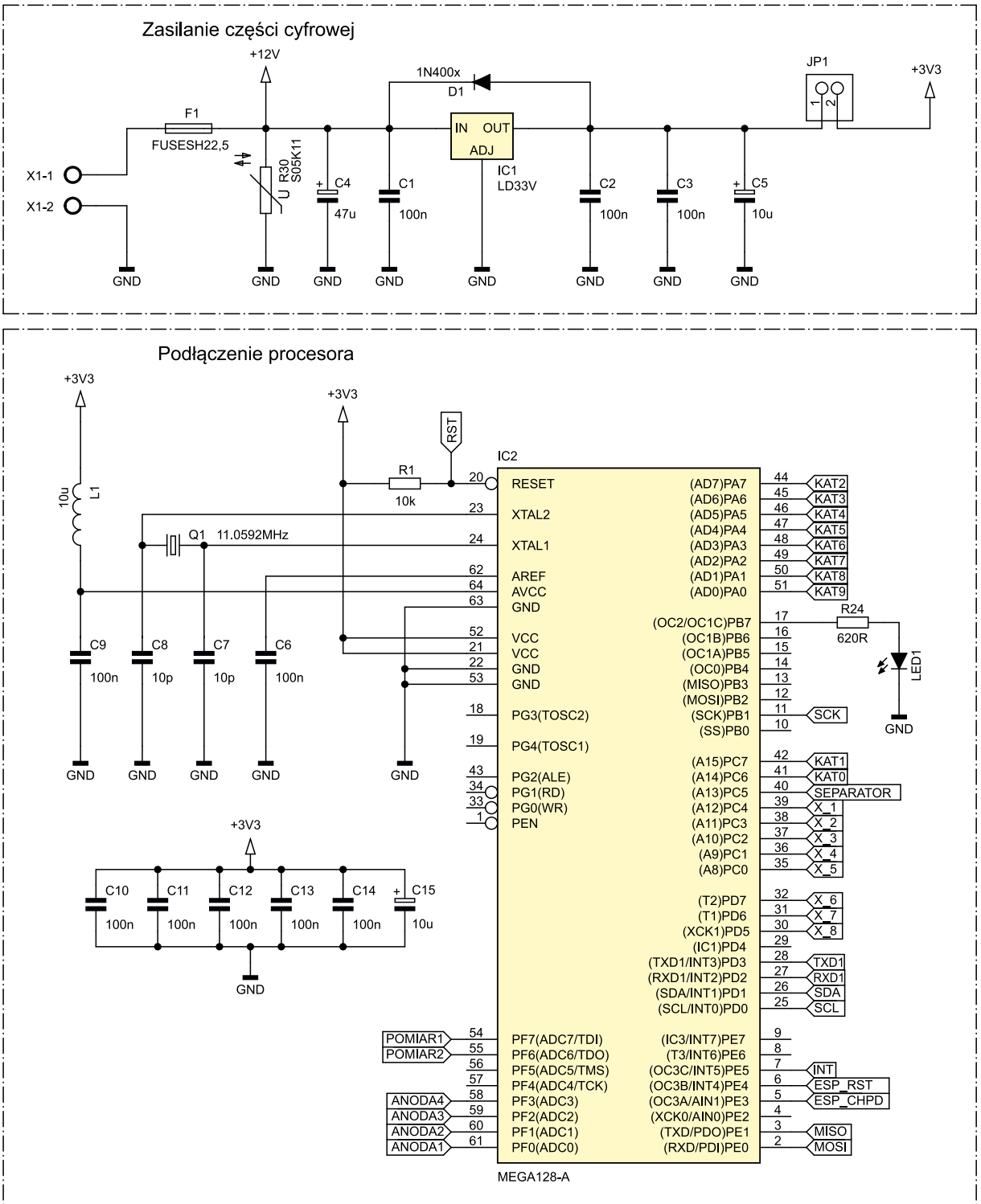
Rysunek 1. Schemat elektryczny cd.: c) moduł Wi-Fi ESP826601 oraz układ RTC

dane, czyli: dzień, miesiąc, rok oraz godzinę i zamienić je na typ całkowity int. Na podstawie dnia, miesiąca oraz roku, korzystając z tzw. algorytmu Zellera, można obliczyć numer dnia tygodnia. Po wywołaniu funkcji *zeller()* otrzymuje się dzień tygodnia w formie: 0 – niedziela, 1 – poniedziałek, …, 6 – sobota.

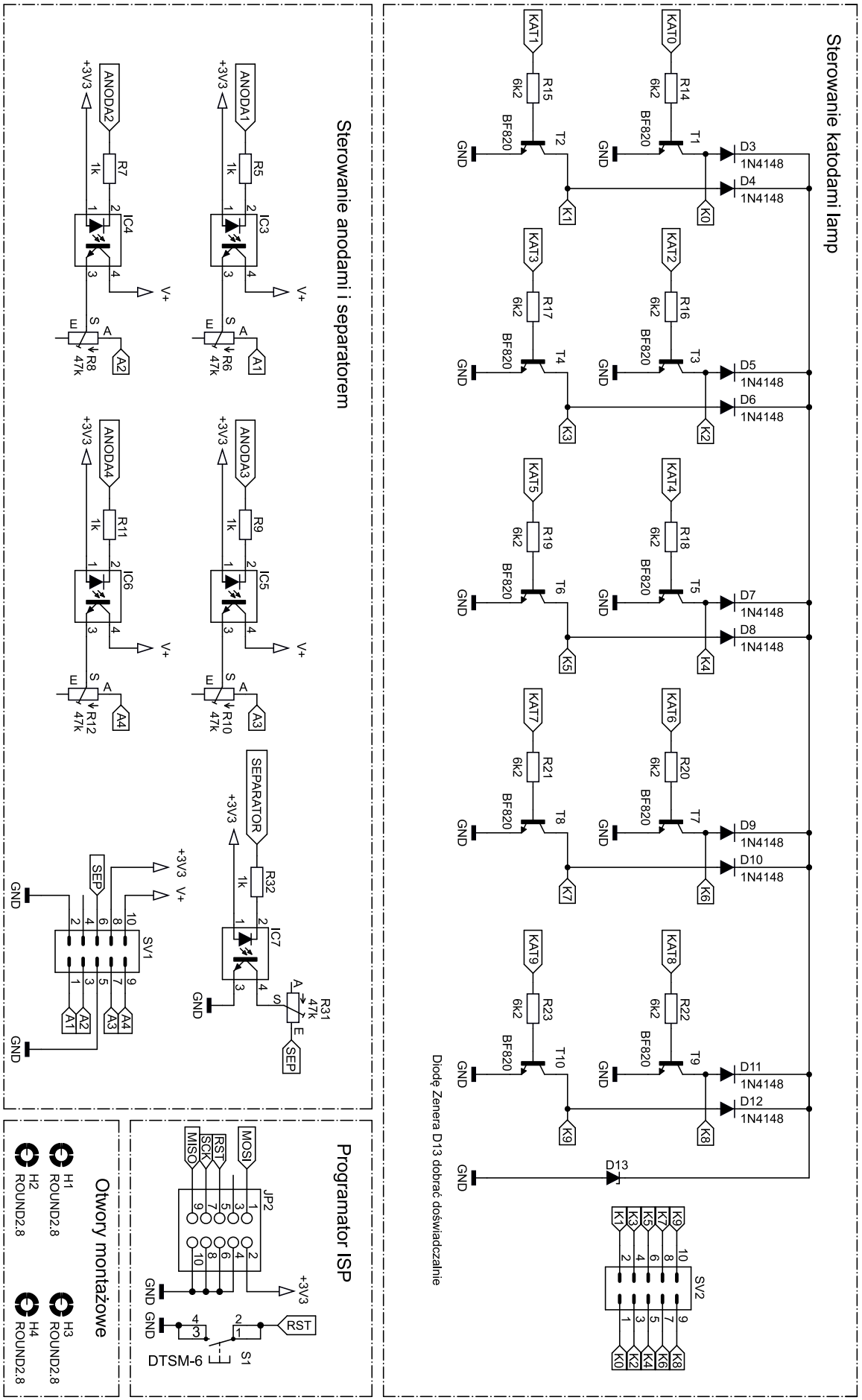
W dalszej części programu, w funkcji *ChangeTime()*, sprawdzana jest aktualna data wraz z dniem tygodnia. Na tej podstawie ustawia się obowiązującą strefę czasową. W miesiącach styczeń, luty, kwiecień… wrzesień oraz grudzień sprawa jest prosta. Nie ma wtedy zmiany czasu, dlatego strefa czasowa pozostaje ta sama. Sposób sprawdzania i zmiany czasu w marcu i październiku jest bardzo podobny, więc omówiona zostanie tylko zmiana czasu z zimowego na letni. Zmiana czasu nie odbywa się tego samego dnia co roku. Dla wyznaczenia właściwego dnia należy spełnić kilka warunków: miesiącem musi być marzec; dzień miesiąca musi być większy od 24; dniem tygodnia musi być niedziela, a godzina, w jakiej nastąpi zmiana, to 2.00. Wydaje się, że należy postawić warunek:

```
if(month==3 && day_of_month>24 &&
day_of_week==0 && hour>=2) time-
zone = 7200
```

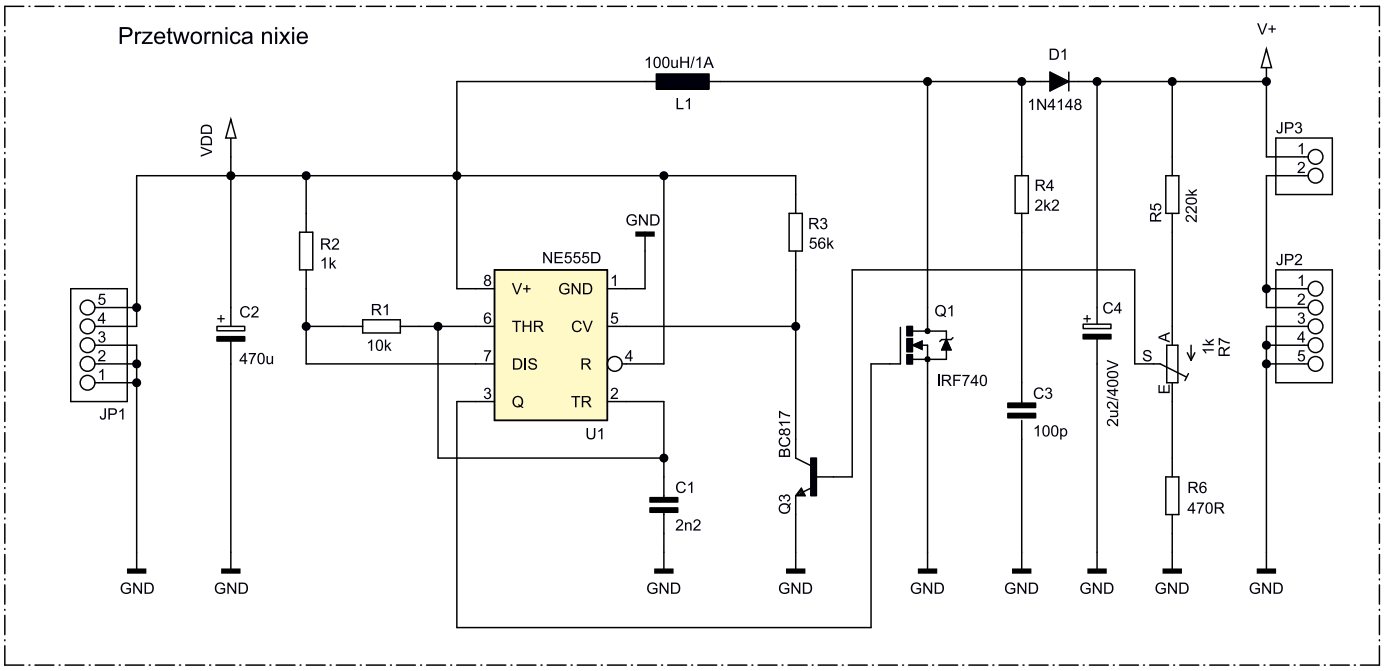
Nastąpiła zmiana czasu i wszystko jest w porządku. Tylko do rozważenia zostaje jedna rzecz: zakładając, że 20 marca uruchomiono moduł, zmiana czasu nastąpiła w niedzielę, 26 marca, z godziny 2.00 na 3.00, a 27 marca nastąpiła chwilowa przerwa w dostawie energii elektrycznej na jedną minutę i po jej



Rysunek 1. Schemat elektryczny cd.: d) moduł mikrokontrolera



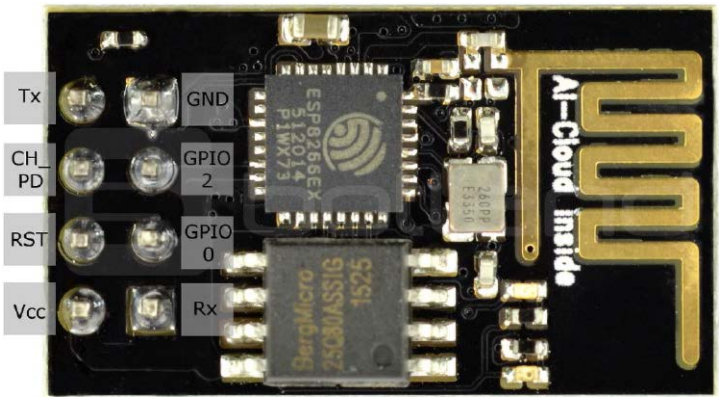
Rysunek 1. Schemat elektryczny cd.: e) moduł drivera wyświetlaczy



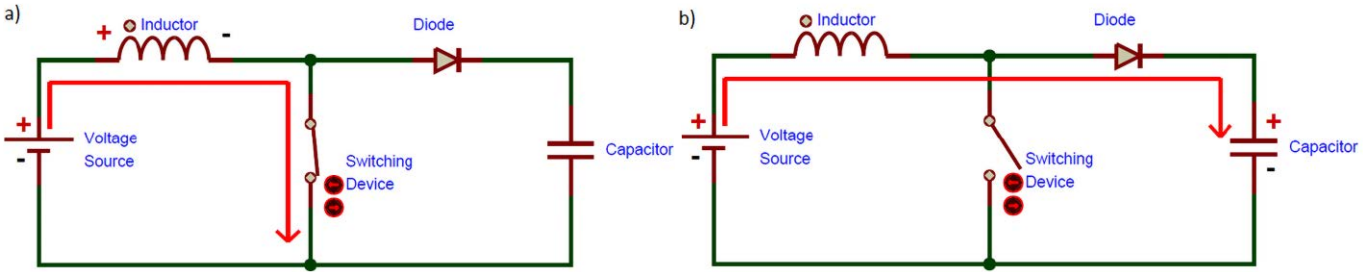
Rysunek 1. Schemat elektryczny cd.: b) moduł przetwornicy podwyższającej napięcie

upływie moduł się włącza. Co pojawi się na terminalu w takim przypadku? Po włączeniu, na terminalu, mimo obowiązującego już czasu letniego, pojawi się czas zimowy. Kiedy skończy się marzec i miną 4 dni, wszystko wróci do normy, czas zmieni się na właściwy.

W celu zapobieżenia tej sytuacji awaryjnej użyto pętli *do…while*. W pętli tej następuje sprawdzanie, czy nie nastąpiła przerwa w dostępie do Internetu, w dostawie energii lub inny przypadek. W pierwszej iteracji pętli sprawdza się, czy dzień miesiąca jest większy od 25 oraz czy nastąpił poniedziałek, po ostatniej niedzieli marca. W kolejnej iteracji sprawdza się,



Rysunek 2. Moduł ESP826601 z podpisanymi wyprowadzeniami



Rysunek 3. Schemat obrazujący zasadę działania przetwornicy boost: a) przy załączonym kluczu; b) przy otwartym kluczu

czy dzień miesiąca jest większy od 26 oraz czy nastąpił wtorek po ostatniej niedzieli marca. W kolejnych iteracjach pętli wykonywane jest analogicznie dalsze sprawdzanie, aż do końca miesiąca. Jeśli warunek, który jest sprawdzany w pętli, jest prawdziwy, to następuje zmiana czasu, czyli czas zmienia się z zimowego na letni. Wygląd odebranych ramek danych w terminalu pokazuje **rysunek 4**.

Oprogramowanie procesora ATmega128

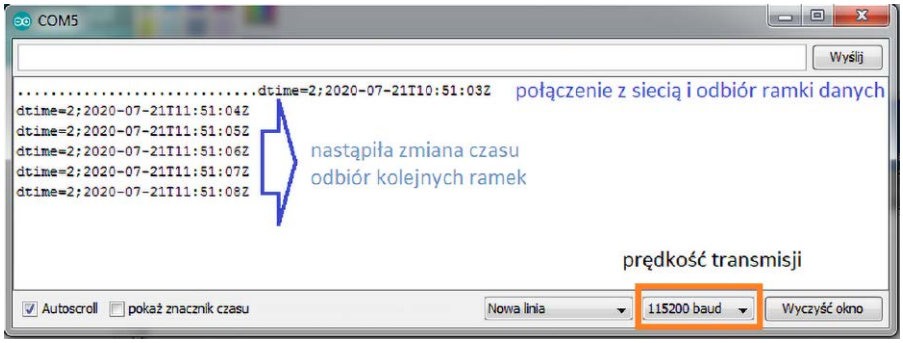
Mikrokontroler jest taktowany zewnętrznym rezonatorem 11,0592 MHz. Program sterujący jego pracą został napisany w języku C, w środowisku Eclipse. Ustawiając fusebity, oprócz wyboru taktowania procesora i wyłączenia interfejsu JTAG, należy także wyłączyć fusebit *m103c* (*m103=1*), ponieważ domyślnie jest kompatybilny z procesorem ATmega103. Działanie programu jest oparte na przerwaniach. Częstotliwość taktowania została tak dobrana, aby uzyskać małą ilość błędów transmisji interfejsu UART, przy prędkości dopasowanej do modułu Wi-Fi. Do obsługi interfejsów I²C (komunikacja z układem RTC DS3231) oraz UART (komunikacja z modułem Wi-Fi ESP8266-01) zastosowano biblioteki z książki Mirosława Kardasia pt. „Język C. Pasja programowania mikrokontrolerów 8-bitowych".

Funkcję konfigurującą układ DS3231 do pracy pokazuje **listing 2**. Po ustawieniu częstotliwości magistrali na 100 kHz należy wyzerować bit INTCN w rejestrze *Register Control* układu RTC. Ponieważ pozostałe bity tego rejestru także są równe zero, można wyzerować cały rejestr. Po wyzerowaniu rejestru na wyjściu INT/SQW pojawi się przebieg o częstotliwości 1 Hz, co zostało wykorzystane przy konfiguracji układu czasowego Timer0, który będzie zgłaszał przerwanie od układu RTC co jedną sekundę. Dane przedstawiające czas wpisuje się do układu poprzez wysłanie ich do odpowiednich rejestrów w kodzie BCD przez magistralę I²C.

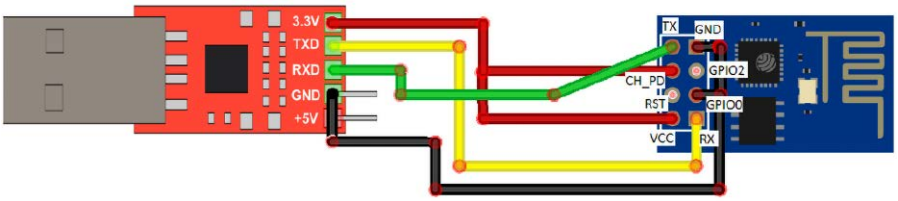
Biblioteka do obsługi UART została nieco zmodyfikowana w celu obsługi procesora ATmega128. Zmiany, jakie wykonano w pliku *mkuart.c*, to m.in. modyfikacja przerwania nadawczego i odbiorczego. Wszystkie modyfikacje zostały uwzględnione w kodach źródłowych dołączonych do materiałów dodatkowych do tego projektu. Funkcja inicjująca interfejs UART, gdzie ustawione są parametry ramki danych oraz włączony nadajnik i odbiornik poprzez ustawienie odpowiednich bitów w rejestrach mikrokontrolera, została pokazana na **listingu 3.** Samego modułu ESP nie trzeba konfigurować do pracy. Zostaje on wybudzony, gdy na jego wejściu CH_PD pojawi się stan wysoki. W przeciwnym razie układ jest w stanie uśpienia.

Kolejna funkcjonalność to konfiguracja i przygotowanie do pracy wyświetlacza lampowego. Wyświetlanie czasu na wyświetlaczu odbywa się poprzez multipleksowanie lamp z odpowiednią częstotliwością dobraną tak, aby pozbyć się efektu migotania i efektu duchów. Procedura obsługi przerwania od Timer2 korzysta z instrukcji *switch…case*, jej kod został pokazany na **listingu 4.** Anody lamp są cyklicznie załączane, a czas jest wyświetlany poprzez wysterowanie katod. Na samym początku, w pierwszym przerwaniu wszystkie anody są wyłączone. W kolejnym przerwaniu załącza się pierwszą anodę i wynik prezentowany jest na katodach. W następnym kroku pierwsza anoda jest wyłączona. Kiedy nadejdzie następne przerwanie, załączana jest druga anoda, a czas wyświetlany jest na katodach. Potem anodę się wyłącza i przy kolejnych lampach proces wygląda analogicznie do opisanego. Lampy więc załączane są de facto co drugie przerwanie.

Odbiór danych z modułu Wi-Fi odbywa się przez tzw. zdarzenia. Oznacza to, że przerwanie nastąpi w momencie nadejścia ramki danych. Ramka danych (zapisywana w zmiennej *buf*) ma postać taką jak na rysunku 4 i jest to zmienna typu string. W celu użycia tych danych należy przychodzący string rozbić na tzw. tokeny i wyodrębnić z niego zmienne, a następnie wysłać je do układu RTC. Odpowiedni kod zawarty



Rysunek 4. Odbierane ramki danych z modułu ESP826601, widok w oknie terminalu



Rysunek 5. Sposób podłączenia modułu ESP826601 na czas programowania

jest na listingu 1. Dla otrzymania dni tygodnia z ramki danych należy uzyskać string bez frazy *dtime=*. Wykonujemy to poprzez zastosowanie funkcji *strncasecmp()*. Otrzymujemy wtedy zmienną bez pierwszych sześciu znaków. Następnie wyodrębniamy dzień tygodnia do osobnej zmiennej (nadal typ string) poprzez użycie funkcji *strtok()*, która wyszukuje znak wpisany do drugiego argumentu funkcji, ze zmiennej podanej jako pierwszy argument, i w to miejsce wstawia zero (NULL). Po kolejnym wywołaniu funkcji, do zmiennej *esp_dayofweek* zapisuje się dzień tygodnia. Dalsze instrukcje pokazują wyodrębnienie zmiennych wyrażających godzinę, minutę i sekundę. Funkcja *atoi()* zamienia dane typu ASCII na INT. Uzyskane zmienne, jako typ liczbowy, zapisuje się w pamięci układu RTC.

Główna funkcja programu nie zawiera wielu linii kodu. Na samym początku ustawia się przedział czasu, co jaki ma być synchronizowany układ DS3231. Dokładnie co jedną sekundę, na podstawie przerwania od układu DS3231, odczytuje się dane i wyświetla na wyświetlaczu, za co odpowiada funkcja *show_time()*.

Montaż i uruchomienie

Schematy płytek drukowanych zostały pokazane na **rysunkach 6, 7, 8**. Lutowanie komponentów zaczyna się w sposób standardowy,

Wykaz elementów:

Płyta główna

Rezystory:
R1…R4: 10 kΩ SMD0603
R24, R29: 620 Ω SMD0603 (nie montować)
R13: 68 Ω SMD0603 (montować w przypadku montażu kondensatora C18)
R25, R26, R27, R28: 0 Ω SMD0603 (montować tylko R27 i R28)
R14…R23: 6,2 kΩ SMD0603
R5, R7, R9, R11, R32: 1 kΩ SMD0603
R6, R8, R10, R12, R31: 47 kΩ potencjometry

Kondensatory:
C1…C3, C6, C9…C14, C16: 100 nF SMD0603
C4: 47 µF/16 V
C5, C15: 10 µF/16 V
C7, C8: 10 pF SMD0603
C18: 0,22 F superkondensator (w przypadku rezygnacji z baterii)

Półprzewodniki:
LED1, LED2: SMD1206 zielona
D1, D3…D12: 1N4148 SMD
D2: BAT54 SOT23 (w przypadku montażu kondensatora C18)
D13: dioda Zenera 65 V typu C1702-15 65V
T1…T10: BF820 (SOT23)
IC1: LD1117V33 (TO220)
IC2: ATmega128 (TQFP64)
IC3…IC7 TLP627 (DIL04)
IC8: DS3231 (SO16W)
IC10: moduł Wi-Fi ESP-01

Pozostałe:
F1: 800mA SH22,5
R30: warystor
L1: 10 µH dławik L3230M
Q1: kwarc 11,0592 MHz SM49
S1: switch resetujący DTSM-6
JP1: goldpin 1×2
JP2: goldpin 2×5
JP3: goldpin 1×4
JP4: goldpin 2×5
JP5: goldpin 1×4
JP6: goldpin 1×4
SV1, SV2: goldpin 2×5

Przetwornica

Rezystory:
R1: 10 kΩ SMD0603
R2: 1 kΩ SMD0603
R3: 56 kΩ SMD0603
R4: 2,2 kΩ SMD0603
R5: 220 kΩ SMD0603
R6: 470 Ω SMD0603
R7: 1 kΩ potencjometr

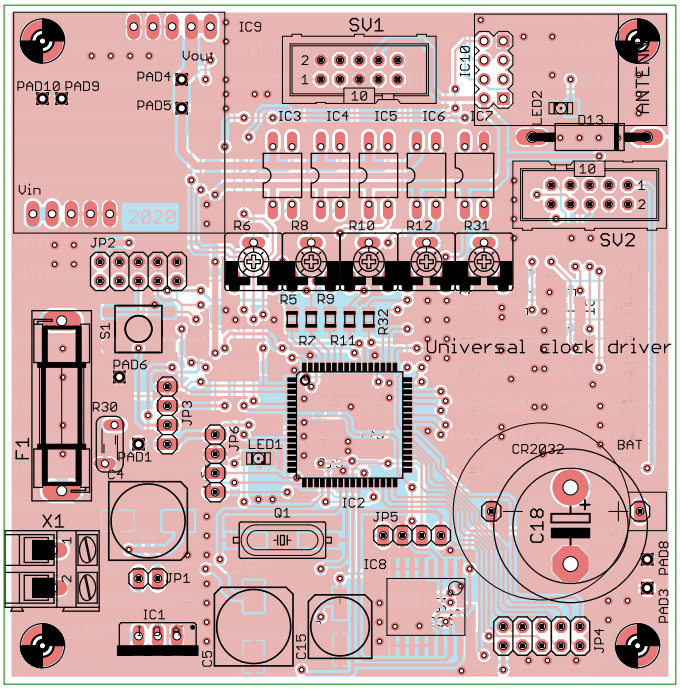
Kondensatory:
C1: 470 µF 5 mm THT
C2: 2,2 nF SMD0603
C3: 100 pF SMD0603
C4: 2,2 µF/400 V 5 mm THT

Półprzewodniki:
D1: 1N4148
Q1: IRF740 (TO220)
Q3: BC817 (TO92)
U1: NE555

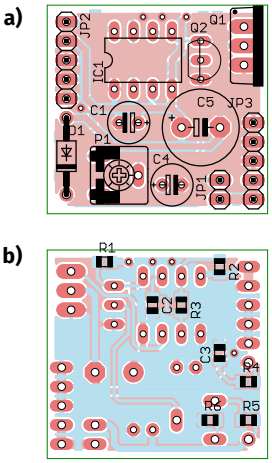
Pozostałe:
L1: 100 µH/1 A 10 mm THT co najmniej 1 A
JP1, JP2: goldpin 1×5
JP3: goldpin 1×2

Wyświetlacz

N1…N4: lampa Nixie IN-4 + podstawki pod lampy
N5, N6: separator INS-1
J1, J2: goldpin 2×5



Rysunek 6. Schemat płyty głównej zegara wraz z rozmieszczeniem elementów



Rysunek 8. Schemat płytki przetwornicy boost wraz z rozmieszczeniem elementów: a) strona TOP; b) strona BOTTOM

zaczynając od tych najmniejszych, czyli rezystorów i kondensatorów SMD. W następnej kolejności należy zamontować układy scalone (IC2, IC8), a na koniec potencjometry, transoptory i złącza THT. Mimo że na płytce jest wiele padów na komponenty, to nie ma konieczności montowania ich wszystkich. Elementy niezbędne do działania znajdują się w wykazie elementów.

Na płycie wyświetlacza, pod lampami, zastosowano piny montażowe, ale równie dobrze można zastosować piny z różnego rodzaju złączy, np. LPT DB25.

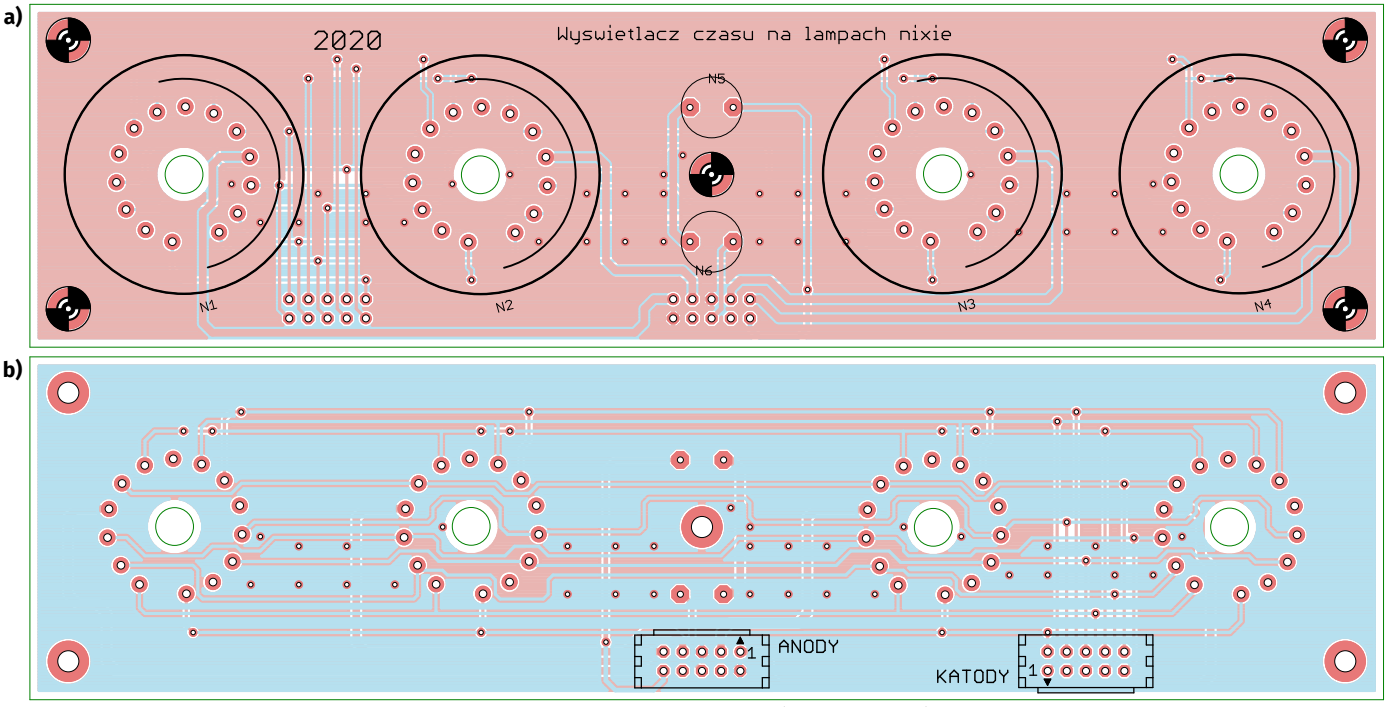
Szczególną ostrożność należy zachować przy lutowaniu elementów przetwornicy, a jej uruchamianie najlepiej przeprowadzić poza układem, ponieważ wyprostowane napięcie wyjściowe, w zależności od ustawień potencjometru, może wynieść nawet 300 V.

Przed programowaniem modułu ESP826601 należy zainstalować sterowniki USBUART oraz zmontować układ, umożliwiający zaprogramowanie modułu (**rysunek 5**), albo użyć gotowej przejściówki. Na czas programowania modułu trzeba zewrzeć wyjście GPIO0 z masą modułu. Poprawność wgrania programu można zobaczyć poprzez podgląd odbieranych ramek danych w dowolnym terminalu (np. terminal Arduino, PuTTY, itp.)

Kolejnym krokiem jest zaprogramowanie procesora AVR. Potrzebny będzie programator ISP, który należy podłączyć do złącza JP2. Do procesu programowania używa się tych samych wyjść procesora, których używa on przy komunikacji z modułem ESP01. Dlatego na czas programowania procesora moduł Wi-Fi powinien znajdować się poza płytką główną.

Nie ma konieczności lutowania diody LED1. Została ona umieszczona na płytce w celu sprawdzania odbierania danych z interfejsu I²C. Miga co jedną sekundę z każdym nadchodzącym przerwaniem.

Diody D3...D13 zostały zamontowane w celu sprzętowej eliminacji efektu duchów lamp. Dioda Zenera D13, w docelowym układzie, pracuje przy napięciu 65 V, ale w zależności od typu lamp można ją dobrać doświadczalnie. Przy pierwszym uruchomieniu pomocne będą zworki, przez które podaje się napięcia zasilające.



Rysunek 7. Schemat płytki wyświetlacza wraz z rozmieszczeniem elementów: a) strona TOP; b) strona BOTTOM

Fotografia 1. Zmontowane komponenty układu umieszczone w obudowie

Fotografia 2. Zmontowane komponenty układu, widok na płytkę wyświetlacza

Podczas uruchamiania prototypu zdarzało się, że po kilku godzinach działania występowało dziwne zjawisko – cyfra wyświetlająca na lampie godziny przeskakiwała godzinę do przodu lub do tyłu. Rozwiązaniem problemu było dolutowanie kondensatora ceramicznego przy wyprowadzeniach zasilających procesora. Początkowa wersja projektu nie zapewniała odpowiedniej filtracji zasilania. Wersja docelowa jest wolna od tej wady. Zmontowane płytki umieszczone w stylowej drewnianej obudowie pokazują **fotografie 1 i 2**.

Podsumowanie

Płytka główna zegara została zaprojektowana z myślą o różnych rozwiązaniach rozwojowych. Po zmianach w programie można zastosować inny typ wyświetlacza (TFT, OLED), łącząc go przez wyprowadzone piny. Dzięki zastosowaniu dołączanej przetwornicy można użyć wyświetlacza zasilanego z innego napięcia. Poprzez wyprowadzenia z przetwornika analogowo-cyfrowego, umieszczone na płycie, można mierzyć natężenie światła wyświetlacza i na tej podstawie programowo regulować jasność jego świecenia. Można także wykorzystać sam moduł Wi-Fi, wysyłający komplet danych z godziną, datą oraz dniem tygodnia do własnego projektu, który będzie potrzebował tych danych. Niekoniecznie musi to być zegar.

Mateusz Kuliński
mateuszkulinskio1@gmail.com