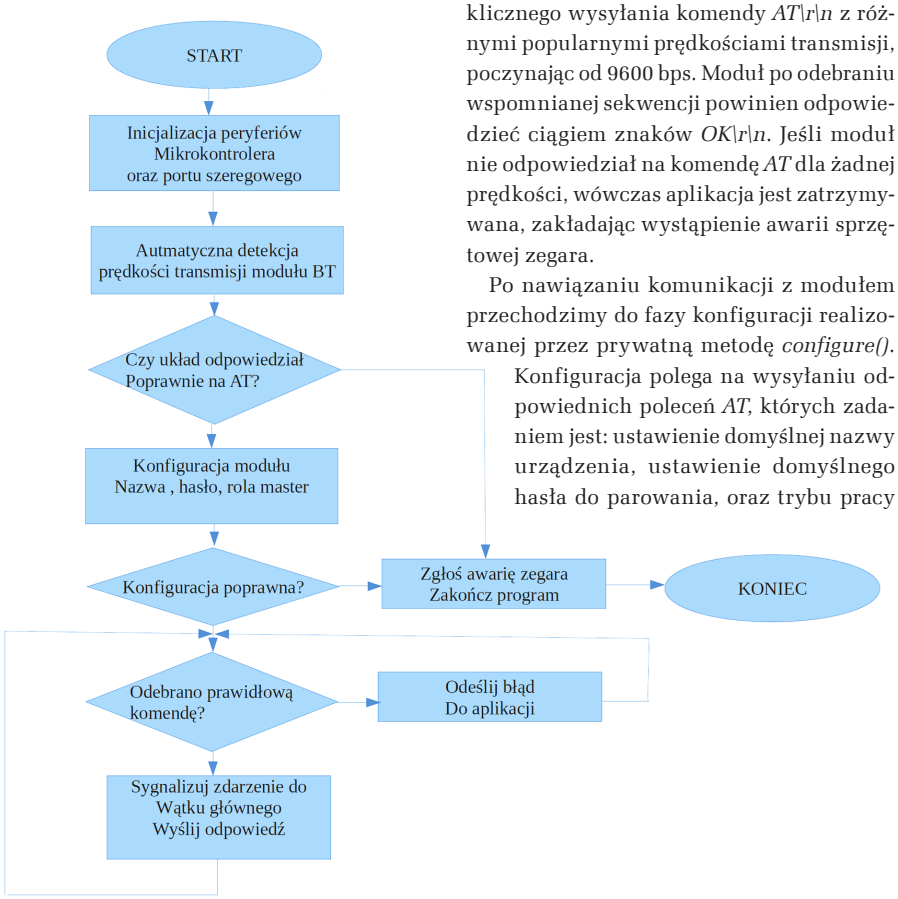
Androidowy zegar Nixie (2)

Na łamach czasopism elektronicznych oraz stronach internetowych opublikowano wiele projektów zegarków z lampami Nixie, w najróżniejszych konfiguracjach i opcjach. Były zarówno projekty oparte o popularne mikrokontrolery, były zegarki zbudowane na układach serii 74HC, a nawet zrealizowane na układach programowalnych. Jednak ten zegarek jest inny, ponieważ zastosowano w nim najnowsze zdobycze techniki, takie jak sensor laserowy lub sterowanie za pomocą smartfona. Uwaga: artykuł stanowi kontynuację projektu opisanego w EP 5/2017.

Za komunikację ze światem zewnętrznym odpowiada wątek serwisu Bluetooth, który zajmuje się zarządzaniem modułem oraz realizuje obsługę protokołu pomiędzy zegarem, a aplikacją zewnętrzną. Serwis Bluetooth zaimplementowany został w klasie *remote_cmd_serwer (pliki remote_cmd_server.cpp/hpp)* i działa według algorytmu przedstawionego na **rysunku 4**. Rozpoczyna pracę od inicjalizacji podstawowych układów peryferyjnych mikrokontrolera oraz konfiguracji portu szeregowego. Po skonfigurowaniu portu, mikrokontroler przystępuje do automatycznego wykrywania prędkości transmisji, jaką posługuje się moduł i próby nawiązania komunikacji, co jest realizowane przez prywatną metodę *detect_baudrate()*. Działanie tej metody polega na próbie cyklicznego wysyłania komendy *AT\r\n* z różnymi popularnymi prędkościami transmisji, poczynając od 9600 bps. Moduł po odebraniu wspomnianej sekwencji powinien odpowiedzieć ciągiem znaków *OK\r\n*. Jeśli moduł nie odpowiedział na komendę *AT* dla żadnej prędkości, wówczas aplikacja jest zatrzymywana, zakładając wystąpienie awarii sprzętowej zegara.

Po nawiązaniu komunikacji z modułem przechodzimy do fazy konfiguracji realizowanej przez prywatną metodę *configure()*. Konfiguracja polega na wysyłaniu odpowiednich poleceń *AT*, których zadaniem jest: ustawienie domyślnej nazwy urządzenia, ustawienie domyślnego hasła do parowania, oraz trybu pracy w roli podrzędnej slave. Tryb slave polega na tym, że moduł oczekuje na połączenie zgodne z protokołem RFCOMM. Po zakończeniu fazy konfiguracji serwis przechodzi



Rysunek 4. Algorytm działania serwisu obsługi Bluetooth

DODATKOWE MATERIAŁY NA FTP:
ftp://ep.com.pl
USER: 92822, PASS: 37euo8qf

W ofercie AVT*
AVT-5582

Podstawowe informacje:

Androidowy zegar Nixie ma następujące parametry:
- Wyświetlacz Nixie o wielkości cyfr 15,5 mm.
- Odchyłka autonomicznego pomiaru czasu max ±1 min/rok.
- Automatyczna zmiana czasu lato/zima.
- Wygodne sterowanie za pomocą interfejsu Bluetooth i aplikacji dla systemu Android dostępnej w sklepie Play.
- Podstawowa obsługa pozbawiona przycisków mechanicznych za pomocą zbliżeniowego czujnika laserowego.
- Funkcja alarmu w trybie tygodniowym.
- Automatyczne sterowanie poziomem jasności w zależności od natężenia oświetlenia, z możliwością ustawienia stałego poziomu podświetlania z aplikacji.
- Efekt płynnego przejścia pomiędzy cyframi.
- Możliwość zaprogramowania przedziału godzin, kiedy zegar ma być wyłączony np. w nocy, aby nie przeszkadzać oraz przedłużać żywotność lamp.
- Algorytm automatycznego odtruwania katod, przedłużający żywotność lamp.
- Przyjemny dla ucha sygnał alarmu.
- Zasilanie z zasilacza wtyczkowego 12 V, pobór mocy ok. 2 W (przy 50% jasności).

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-3141	Zegar Nixie z sekundami (EdW 6/2016)
AVT-3097	Zegar Nixie (EdW 7/2014)
AVT-5145	Zegar retro na lampach Nixie (EP 9/2008)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (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 dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają 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***

Tabela 1. Komendy sterujące zegarem

Pytanie	Odpowiedź	Opis
get time\r\n	OK <UTS>\r\n ERROR <description> <code>\r\n	Pobiera czas zegara, gdzie <UTS> to czas podany w formacie Unix Time Stamp.
set time <uts>\r\n	OK\r\n ERROR <description> <code>\r\n	Ustawia czas główny zegara na wartość wskazywaną przez <UTS>.
get alarm\r\n	OK <hh:mm> <al_flags>\r\n ERROR <description> <code>\r\n	Pobiera aktualny czas alarmu, gdzie: <hh:mm> to godzina i minuta alarmu, <al_flags> Maska bitowa dni tygodnia zwrócona w postaci liczby szesnastkowej oznaczająca odpowiednio 0x01: Poniedzialek 0x02: Wtorek itd. 0x00: alarm wyłączony.
set alarm <hh:mm><al_flags> \r\n	OK\r\n ERROR <description> <code>\r\n	Ustawia czas alarmu zgodnie z parametrami z poprzedniego punktu.
get brightness\r\n	OK <bsetmode> <currval>\r\n ERROR <description> <code>\r\n	Pobiera aktualny poziom jasności wyświetlacza, gdzie <bsetmode>: 1 – sterowanie jasnością automatyczne, <50,100> – ręcznie ustawiony poziom jasności. <currval> – Bieżąca jasność wyświetlacza w %
set brightness <value>\r\n	OK\r\n ERROR <description> <code>\r\n	Ustawia poziom jasności wyświetlacza, gdzie jako <value> przekazujemy wartość w zakresie 50-100 jeśli chcemy ustalić jasność wyświetlacza na stałe lub A, jeśli jasność wyświetlacza ma być kontrolowana przez czujnik światła.
get failures\r\n	OK <pcaddr> <fcount>\r\n	Funkcja diagnostyczna zwracająca aktualne informację o awariach oprogramowania zegara, gdzie <pcaddr> to adres PC, w którym aplikacja zawiodła, natomiast <fcount> oznacza totalną liczbę awarii.
get displayoff \r\n	OK <from> <to>\r\n ERROR <description> <code>\r\n	Zwraca dobowy przedział godzinowy, w którym wyświetlacz zostanie wygaszony, gdzie <from> to początkowa godzina wygaszenia wyświetlacza; <to> końcowa godzina wygaszenia wyświetlacza.
Set displayoff <from> <to>\r\n	OK\r\n ERROR <description> <code>\r\n	Ustawia dobowy przedział godzinowy, w którym wyświetlacz zostanie wygaszony.

do wykonania pętli głównej, w której oczekuje na przesłanie komendy przez urządzenie zewnętrzne. Zestaw rozkazów sterujących dla zegara oparty jest o proste komendy tekstowe zakończone znakami \r\n. Dzięki takiemu rozwiązaniu możemy sterować zegarem bez konieczności posiadania dedykowanej aplikacji za pomocą terminala dołączonego do wirtualnego portu szeregowego. Jest to istotne na etapie eksperymentowania i uruchamiania układu. Po odebraniu ciągu znaków na porcie szeregowym zakończonym znakami \r\n aplikacja przechodzi do parsowania odebranego ciągu tekstowego. Jeśli odebrana komenda jest prawidłowa wówczas następuje wysłanie zdarzenia *EV_QUERY* informującego wątek główny o odebraniu nowej komendy, a następnie serwer oczekuje na zdarzenie *EV_RESP*, które jest generowane przez aplikację główną po przetworzeniu komendy. Zdarzenie odpowiedzi może zawierać dodatkowe dane, które mają zostać przekazane do aplikacji zewnętrznej przez serwer komunikacyjny. Wszystkie komendy przesyłane są za pomocą znaków ASCII według schematu pytanie-odpowiedź, gdzie ogólny format rozkazu wygląda następująco:

Pytanie: get **lub** set <nazwa_komendy> <*opcjonalne argumenty*>\r\n

Odpowiedz: OK <*opcjonalne argumenty*>\r\n

ERROR <opis> <kod_numeryczny>\r\n

Komunikacja z zegarem polega na wysłaniu komendy odczytującej (**get**) lub ustawiającej (**set**) wybrany parametr zakończonej znakami \r\n , po czym należy czekać na odpowiedź, zawierającą opcjonalnie zwrócone argumenty, lub kod błędu wraz z opisem i kodem numerycznym. Zestaw wszystkich komend zegara przedstawiono w **tabeli 1**.

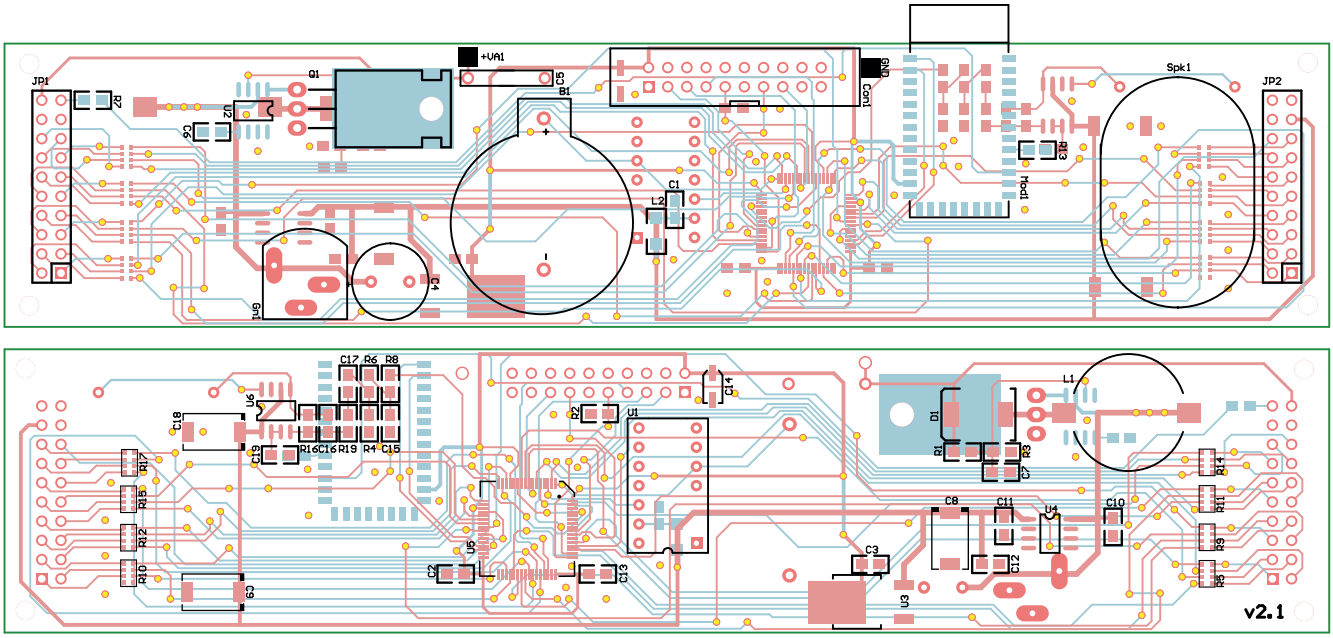
Przy omawianiu oprogramowania zegara warto również wspomnieć o sposobie prowadzenia czasu, który jest realizowany przez **UTS** *(Unix Time Stamp)*. **UTS** jest systemem reprezentacji czasu, który mierzy liczbę sekund jaka upłynęła od 1 stycznia 1970 roku **UTC**, czyli od chwili nazwanej epoką Unixa. Choć ten sposób może wydawać się dziwny, ma szereg zalet w stosunku do zwykłego zliczania czasu realizowanego przez typowe zegarki RTC w formacie **BCD**. Przede wszystkim takie operacje, jak: porównywanie, dodawanie, odejmowanie są bardzo łatwe i sprowadzają się do zwykłych operacji arytmetycznych na pojedynczych liczbach *int32_t* lub *int64_t*. Nie ma konieczności skomplikowanego przeliczania ile minut składa się na godzinę, ile dni na miesiąc itp. Po prostu zegar pracujący w formacie **UTS** zwiększa zawartość licznika co 1 sekundę. Podobnie sprawdzanie warunku wystąpienia alarmu sprowadza się do operacji zwykłego porównania liczb. Główna

Listing 5. Wybór strefy czasowej

```
void _tzset_unlocked_r( struct _reent* )
{
    constexpr auto sechour = 3600;
    __tzinfo_type *tz = __gettzinfo ();
    {
        tz->__tzrule[0].offset  = -1*sechour;
        tz->__tzrule[1].offset = tz->__tzrule[0].offset - 3600;

        tz->__tzrule[0].ch = 'm';
        tz->__tzrule[0].m = 3;
        tz->__tzrule[0].n = 5;
        tz->__tzrule[0].d = 0;
        tz->__tzrule[0].s = 2 * sechour;

        tz->__tzrule[1].ch = 'm';
        tz->__tzrule[1].m = 10;
        tz->__tzrule[1].n = 5;
        tz->__tzrule[1].d = 0;
        tz->__tzrule[1].s = 3 * sechour;
    }
    __tzcalc_limits (tz->__tzyear);
    _timezone = tz->__tzrule[0].offset;
    _daylight = tz->__tzrule[0].offset != tz->__tzrule[1].offset;
    }
}
```



Rysunek 5. Schemat montażowy zegara Nixie

trudność dla **UTS** sprowadza się do końcowego przeliczenia formatu na czas lokalny, zgodnie z lokalną strefą czasową. Pomimo pozornego skomplikowania zaletą takiej interpretacji czasu jest również automatyczne uwzględnienie czasu letniego i zimowego bez konieczności przestawiania zegara. Przeliczanie z formatu **UTS** na czas lokalny jest zadaniem dość skomplikowanym, jednak biblioteka standardowa *libc* kompilatora z *GCC* posiada standardowe funkcje ułatwiające to zadanie. Funkcja *localtime_r* służy do przeliczania czasu z formatu **UTS** na na czas lokalny, natomiast funkcja *mktime* przelicza czas lokalny na format **UTS**. Jedyną rzeczą jaką należy zrobić, to odpowiednio ustawić strefę czasową definiując funkcję **void _tzset_unlocked_r(struct _reent*).** Zadaniem tej funkcji jest wypełnienie struktury *__tzinfo_type* określającej strefę oraz godzinę zmiany czasu z letniej na zimowy i odwrotnie. W naszym przypadku zdecydowaliśmy się na ustawienie polskiej strefy czasowej *CEST*. Istnieje możliwość zdefiniowania dodatkowej funkcjonalności zegara obsługującej wiele stref czasowych, co pozostawiam jako zadanie domowe dla czytelników. Przykład ustawienia strefy czasowej dla polskiego czasu CEST wygląda następująco, jak na **listingu 5**.

Struktura *tzinfo_type* zawiera dwie struktury *__tzrule_type[0] __tzrule_type[1]* odpowiedzialne odpowiednio za czas letni, oraz za czas zimowy. Pole offset w strukturze *__tzrule_type* określa przesunięcie czasu w sekundach względem GMT, odpowiednio dla czasu letniego i zimowego. Pola *m/n/d* określają miesiąc , dzień tygodnia, oraz dzień miesiąca zmiany czasu z letniego na zimowy, lub zimowego na letni. Dodatnie powyższej funkcji spowoduje, że funkcje biblioteczne *localtime/mktime* będą prawidłowo przeliczały czas z formatu **UTS** na lokalny czas **CEST**. Pierwsze mikrokontrolery rodziny STM32 jako zegar **RTC** posiadały prosty licznik 32-bitowy zwiększający swoją zawartość co 1 sekundę, tak więc nadają się idealnie do automatycznego prowadzenia czasu w formacie UTS. Niestety w nowszych wersjach układów producent wycofał się z tego rozwiązania, wybierając gorsze rozwiązanie w postaci zegara RTC zliczającego w **BCD**.

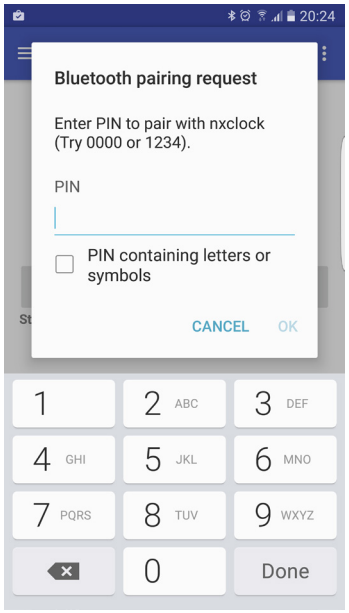
Aplikacja dla systemu ***Android*** służąca do komunikacji została przygotowane z wykorzystaniem oprogramowania *„Android Studio”*, i wykorzystuje profil *Bluetooth-SPP* do komunikacji z zegarem. Oprogramowanie zostało napisane zgodnie z wytycznymi *„Material Design”* i zawiera zestaw prostych i intuicyjnych formatek, które umożliwiają w wygodny sposób ustawienie wszystkich parametrów. Oprogramowanie działa w dwóch wątkach jeden wątek odpowiedzialny jest za realizację komunikacji *Bluetooth* natomiast drugi wątek realizuje operacje interfejsu użytkownika. Komunikacja po *Bluetooth* realizowana jest w klasie *BluetoothSPP (plik BluetoothSPP.java)* i sprowadza się do otwarcia domyślnego interfejsu Bluetooth, i próby nawiązania komunikacji, gdzie jako identyfikator protokołu SPP należy podać ciąg:

```
private final UUID UUID_SPP =
UUID.fromString(„00001101-0000-
1000-8000-00805F9B34FB”);
```

Niestety szczegółowe umówienie aplikacji Android wykracza poza łamy niniejszego artykułu, i zostanie pominięte. Zainteresowanych czytelników odsyłam do kodu źródłowego.

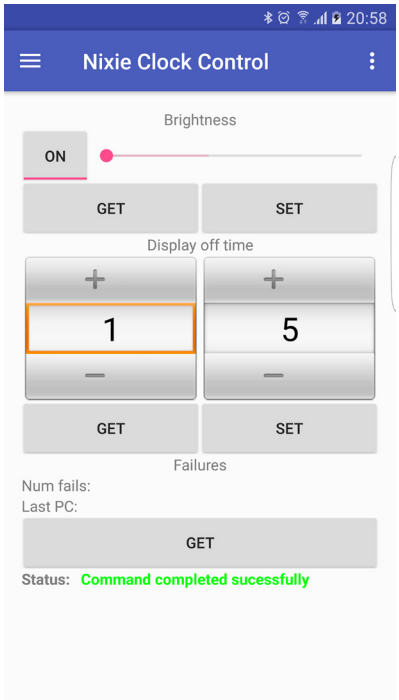
Montaż i uruchomienie

Shemat montażowy płytki sterownika zegara oraz wyświetlacza przedstawiono na **rysunku 5**. Całość wykonano z użyciem elementów SMD 0805, poza kondensatorami wysokonapięciowymi, tranzystorami mocy oraz podstawkami pod lampy. Montaż rozpoczynamy od płytki wyświetlacza. Najpierw montujemy rezystory, następnie tranzystory wysokonapięciowe *MMBTA42*, oraz czujnik oświetlenia *TSL238*. Na koniec montujemy gniazda do lamp NIXIE, oraz dwurzędowe goldpiny stanowiące interfejs połączeniowy z płytką sterownika. Po zmontowaniu płytki wyświetlacza przystępujemy do montażu sterownika. Najpierw montujemy dyskretne elementy SMD kondensatory, rezystory, a następnie przystępujemy montażu półprzewodnikowych elementów SMD. Kolejną czynnością jest montaż modułu Bluetooth HC-05, większych elementów przewlekanych, oraz złącz. Po zakończeniu montażu przystępujemy do uruchomienia płyty głównej, w tym celu do złącza *GN1* podłączamy

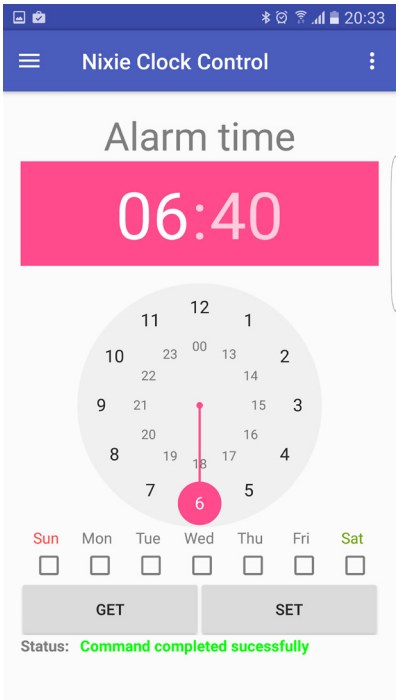


Rysunek 6. Okno parowania

Rysunek 7. Okno główne



Rysunek 8. Widok okna konfiguracyjnego



Rysunek 9. Widok okna alarmów

zasilacz laboratoryjny ustawiony na napięcie *12V* i ograniczeniu prądowym *200mA*. Należy tutaj zachować szczególną ostrożność ponieważ pracująca przetwornica dostarcza napięcia *165V*, co może stanowić zagrożenie. Po podłączeniu zasilania należy sprawdzić poprawność napięć zasilających: napięcie na wyjściu stabilizatora **U4** powinno mieć wartość *5V*, natomiast napięcie na wyjściu stabilizatora **U3** powinno być zbliżone do *3,3V*. Należy również sprawdzić poprawność działania przetwornicy dostarczającej napięcia dla lamp, gdzie wartość napięcia powinna być zbliżona do *165V*. Jeśli napięcie znacząco odbiega od wartości zadeklarowanej należy sprawdzić jakość cewki **L1**, oraz ewentualnie skorygować wartości rezystorów w dzielniku (**R1** oraz **R3**). Zaleca się użycie rezystorów o tolerancji 1%. Na zakończenie należy również sprawdzić ogólny pobór prądu przez układ, który przy niezaprogramowanym mikrokontrolerze i odłączony wyświetlaczu nie powinien przekraczać 30mA. Gdy już będziemy pewni co do prawidłowości napięć zasilających możemy przystąpić do wgrania oprogramowania do mikrokontrolera. W tym celu do złącza **CON1** należy podłączyć programator zgodny ze standardem *JTAG* np. *BF-30* czy *STLINK-v1*, a następnie za pomocą oprogramowania *openocd*, lub innego preferowanego zaprogramować mikrokontroler plikiem *btnxclock.hex*. Po zaprogramowaniu układu należy odłączyć zasilanie, dołączyć płytkę wyświetlacza, a następnie ponownie uruchomić zegar. W tym momencie na wyświetlaczu powinien pojawić się przez 10 sekund ciąg szybko zmieniających się cyfr. Zadaniem szybko zmieniającego się ciągu cyfr jest odtruwanie katod wyświetlacza, które również może być wykorzystane do kontroli stanu lamp. Należy zaobserwować czy na wyświetlaczu na pozycjach dziesiątek godzin pojawiają się na przemian cyfry 0,1,2 na pozycji jednostek godzin i minut cyfry 0-9, oraz na pozycji dziesiątek minut cyfry 0-5. Jeśli zaobserwujemy brak świecenia którejś z cyfr lub świecące naraz dwie cyfry należy odłączyć zasilanie i zweryfikować poprawność montażu, prawidłowe działanie tranzystorów MMBTA42 oraz sprawność lamp LC513. Jeśli wszystko działa poprawnie po wykonaniu sekwencji odtruwania na wyświetlaczu powinna pojawić się godzina **0:00** oraz migający dwukropek. Możemy również sprawdzić poprawność działania czujnika laserowego, oraz działanie układu audio. Przybliżając rękę do czujnika i zatrzymując w odległości większej niż 5 cm z głośnika powinniśmy usłyszeć sygnał dźwiękowy, a na wyświetlaczu powinna pojawić się data **1-01.**

Eksploatacja urządzenia

Zaprezentowany zegar do prawidłowej pracy wymaga smarthphona z systemem *Android*, gdyż bez dostępu do telefonu nie będziemy mieli możliwości ustawienia aktualnego czasu. Istnieje również możliwość napisania dedykowanej aplikacji dla innego systemu np. iOS, czy komputera PC ponieważ protokół jest stosunkowo prosty i publicznie dostępny. (Zachęcam do tego czytelników). W ostateczności możemy skorzystać terminala i wirtualnego portu szeregowego w systemie Windows czy Linux i wymagane komendy wpisać ręcznie. Aplikację dla systemu Android możemy zainstalować bezpośrednio ze sklepu Play korzystając z linku: https://play.google.com/store/apps/details?id=pl.boff.btc.nixieclockcontrol na komputerze PC, lub bezpośrednio z *„Play Store”* w telefonie wyszukując aplikację *„Nixie Clock Control”*. Po uruchomieniu aplikacji powinno pojawić się okno parowania z zegarem w oknie należy wpisać pin urządzenia **00972** (**rysunek 6**). Po pomyślnym sparowaniu urządzenia powinniśmy zostać przeniesieni na ekran główny, dotyczący aktualnego czasu. Aby poprawnie zsynchronizować czas zegara z naszym telefonem należy wcisnąć przycisk **SET** (**rysunek 7**).

Po ustawieniu czasu możemy również w zakładce *„Ustawienia”* skonfigurować jasność wyświetlacza, na automatyczną pochodzącą z czujnika światła wciskając przycisk ON, lub ręczną przesuwając suwak jasności na odpowiednią pozycję, a następnie wysłać ustawienia wciskając przycisk **SET** (**rysunek 8**). Możemy również ustawić przedział godzinowy w jakim wyświetlacz będzie wygaszony, co przedłuża żywotność lamp, lub jest przydatne jeśli zegar przeszkadza podczas snu. Ustawienie wartości 0-0 spowoduje, że wyświetlacz będzie zawsze aktywny.

Codzienna eksploatacja zegara sprowadzać się będzie do korzystania głównie z zakładki alarmów (**rysunek 9**), gdzie mamy możliwość zdefiniowania alarmu w cyklu tygodniowym. Co jakiś czas należy korygować ustawienia czasu zegara uruchamiając aplikację i wciskając na ekranie głównym przycisk **SET** jeśli stwierdzimy, że zegar spieszy lub spóźnia się. Z uwagi na precyzyjny generator kwarcowy nie powinno to mieć miejsca zbyt często. Aplikacja została zaprojektowana tak, aby zapewnić współpracę się z kilkoma zegarami jednocześnie. Wybór aktywnego zegara jest możliwy dzięki zakładce *„Paired devices”*.

Lucjan Bryndza, EP