Politechnika Świętokrzyska w Kielcach

Wydział Elektroniki, Automatyki i Informatyki

Projekt: **Programowanie Systemów Wbudowanych**

Grupa:	Temat:	Skład grupy:
1ID21A	**Zegar z wyświetlaczem siedmiosegmentowym**	**Michał Młodawski**
Rok studiów:		
4		

Zegar z wyświetlaczem siedmiosegmentowym – Pobieranie czasu z sytemu GPS

Opracowanie projektu

Czerwiec 2020

Wersja dokumentu 16062020-001PL

1. Opis projektu

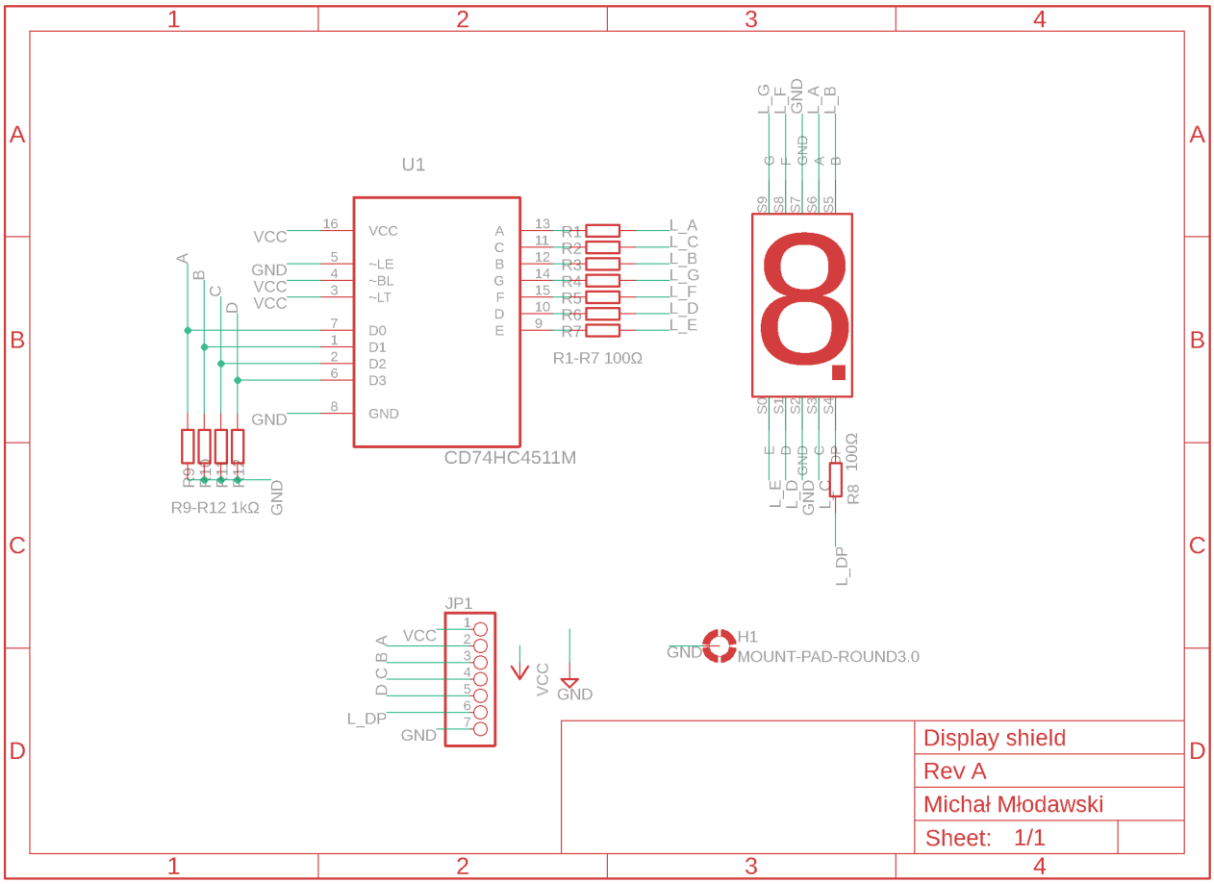
Projekt miał na celu stworzenie zegara opartego na wyświetlaczu siedmiosegmentowym umożliwiającym pobieranie daty i godziny przy użyciu sygnału GPS (ang. Global Positioning System), następnie zapisaniu daty i godziny do zegara czasu rzeczywistego RTC DS3231 i wyświetlenie ich następnie na wyświetlaczach siedmiosegmentowych.

2. Wykorzystane technologie

Jako główny mikrokontroler został wykorzystany mikrokontroler STM32F411RETx do pobierania czasu z systemu pozycjonowania GPS został użyty układ ublox neo-6m, natomiast do przechowywania czasu układ DS3231. Jako kontroler wyświetlaczy został wykorzystany układ scalony CD4511BE.

3. Opis wyświetlacza siedmiosegmentowy

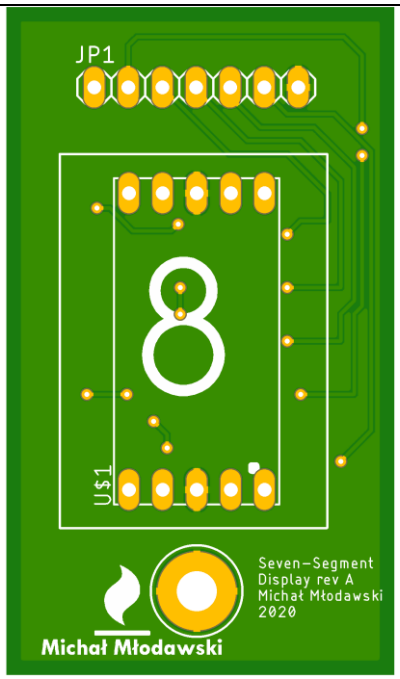
Wyświetlacz siedmiosegmentowy jest sterowany CD4511BE poprzez podanie kodu BCD na wejścia sterujące.



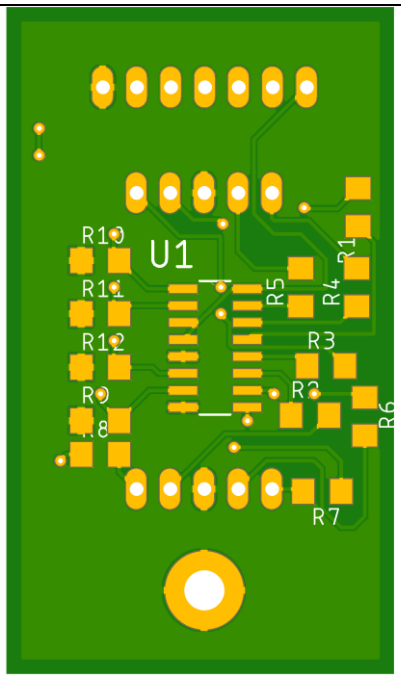
Rysunek 1 Schemat elektryczny przedstawiający sposób podłączenia układu scalonego do wyświetlacza. Źródło: Opracowanie własne

Rezystory R1-R7 służą do ograniczenia prądu anodowego, rezystory R9-R12 są to rezystory podciągające linie sterujące. Układ sterujący CD4511BE pracuje w logice od 3.3V do 18V, więc nie było potrzebny tworzyć żadnych kluczy tranzystorowych lub dzielników napięć.

Wizualizacja płyty PCB kontrolera wyświetlacza siedmiosegmentowego:



Rysunek 2 Strona wierzchnia PCB Źródło: Opracowanie własne



Rysunek 3 Strona tylna PCB Źródło: Opracowanie własne

Tablica prawdy dla kontrolera CD4511BE:

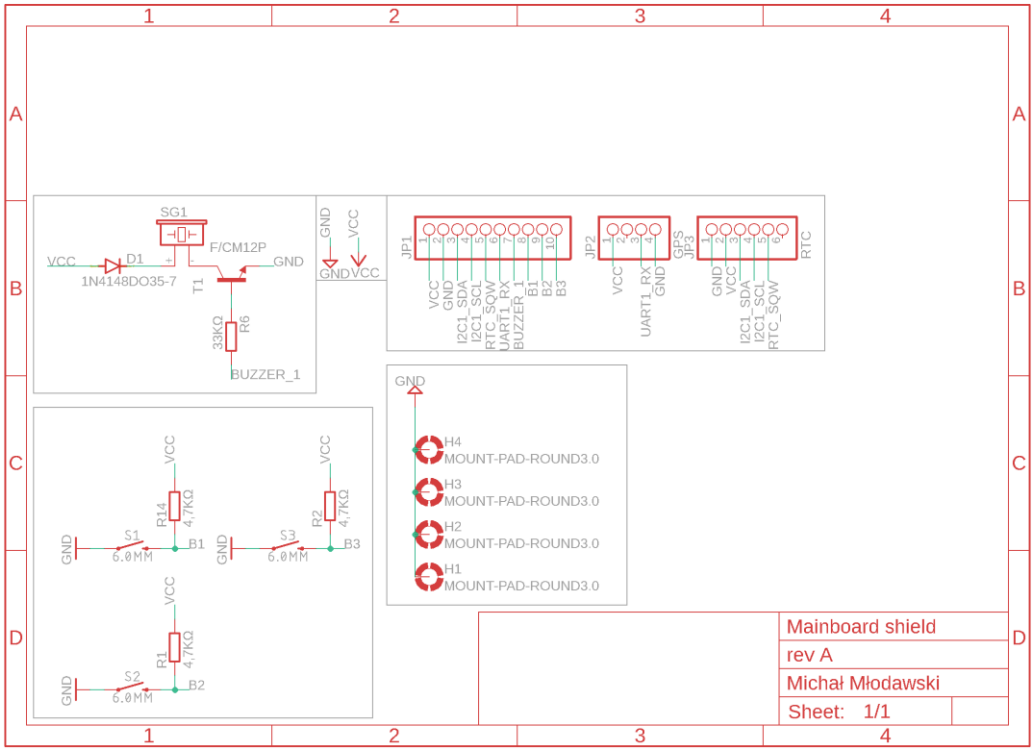
LE	$\overline{BI}$	$\overline{LT}$	D	C	B	A
X	X	0	X	X	X	X
X	0	1	X	X	X	X
0	1	1	0	0	0	0
0	1	1	0	0	0	1
0	1	1	0	0	1	0
0	1	1	0	0	1	1
0	1	1	0	1	0	0
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
0	1	1	1	0	0	1

Tablica prawdy jest to zbiór możliwych kombinacji stanów w celu osiągniecia określonych wyników. Natomiast kod BCD jest to sposób zapisu liczby polegający na zakodowaniu kolejnych cyfr dziesiętnych tej liczby w systemie dwójkowym, przy użyciu tylko czterech młodszych bitów.

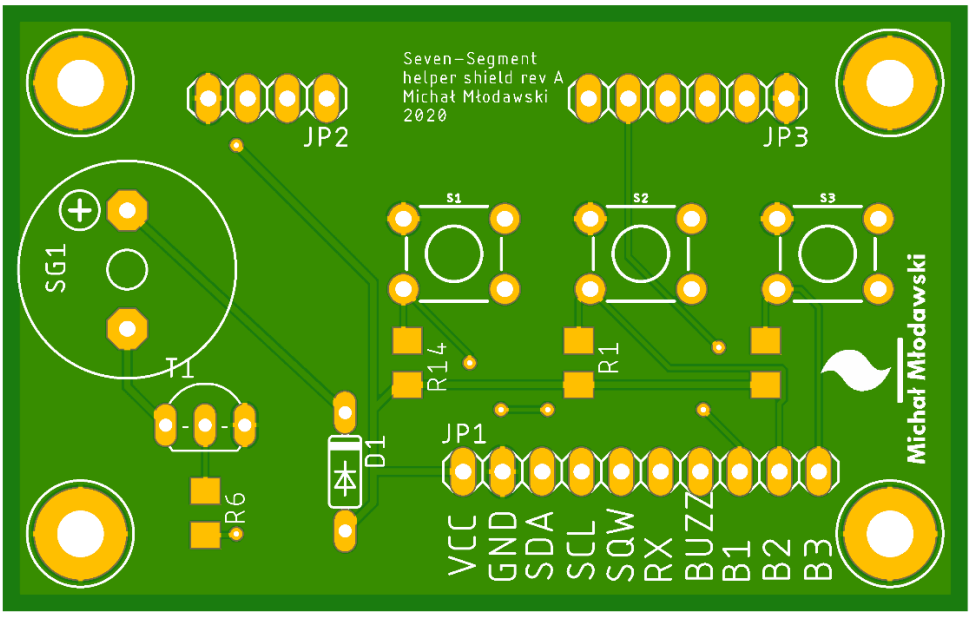
Tabela 1 Tablica prawdy dla CD4511BE Źródło: https://www.ti.com/lit/ds/symlink/cd4511b.pdf?ts=1592323109630

4. Opis kontrolera

Kontroler składa się z trzech przycisków w konfiguracji pull-up z podciągniętymi rezystorami do zasilania 3.3V. Zostały także dodane wyprowadzenia dla zegara czasu rzeczywistego, odbiornika GPS oraz buzzera kontrolowanego poprzez tranzystor NPN pracującego w formie klucza tranzystorowego, który z racji wykorzystania tranzystora bipolarnego ogranicza prąd pobierany przez buzzer. Dodatkowo dodano diodę 1N4148 w celu zapobiegnięcia zakłóceniom na linii zasilania. Niestety z racji zastosowania zwykłej diody powoduje ona spadek napięcia na buzzerze, ale nie wpływa to negatywnie na jego głośność.



Rysunek 4 Schemat elektryczny kontrolera Źródło: Opracowanie własne



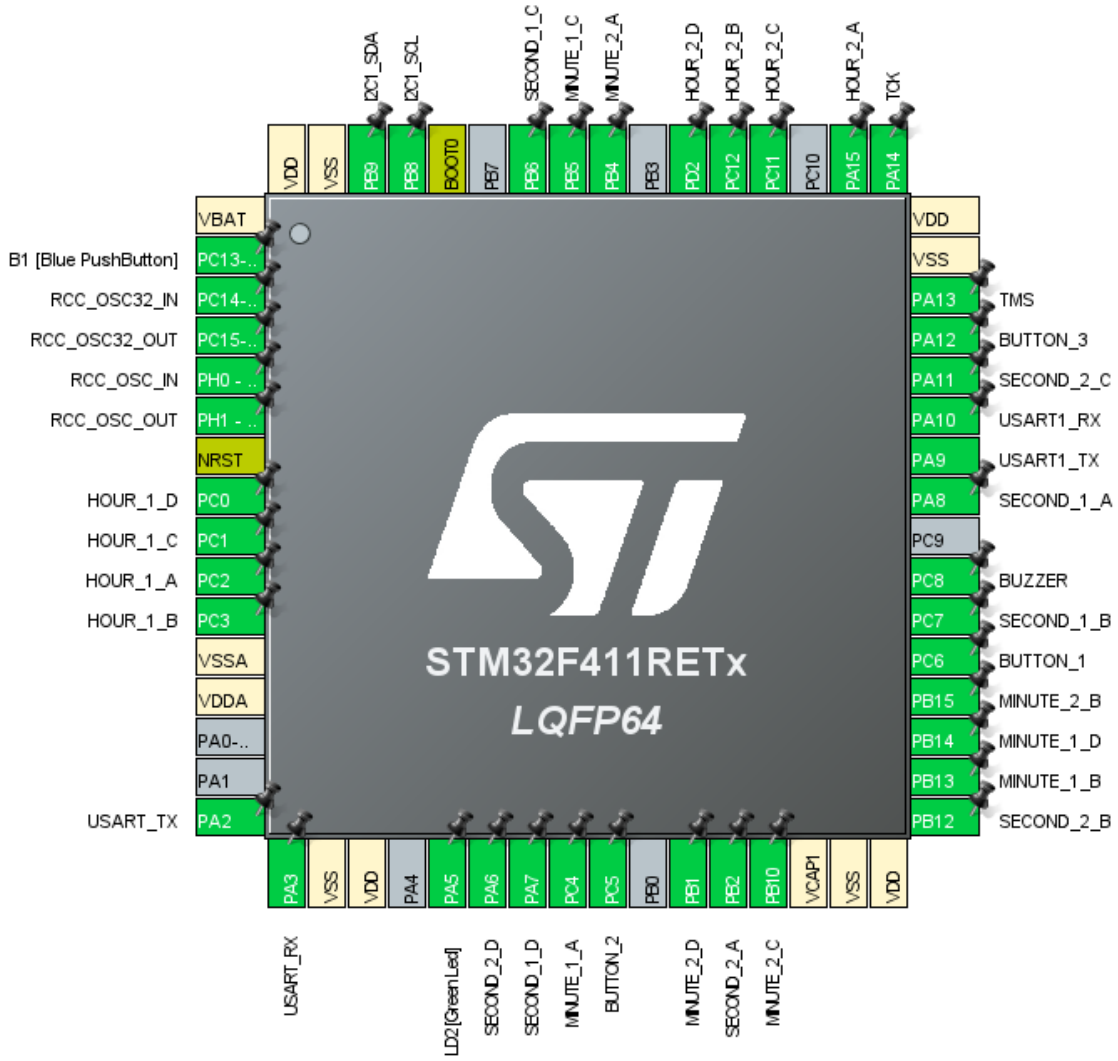
Rysunek 5 Strona wierzchnia PCB Źródło: Opracowanie własne

5. Omówienie warstwy sprzętowej

W tym rozdziale zostaną omówione wszystkie elementy sprzętowe systemu składającego się na gotowe rozwiązanie spełniające założenia projektowe.

5.1 Opis zastosowanych standardów komunikacyjnych

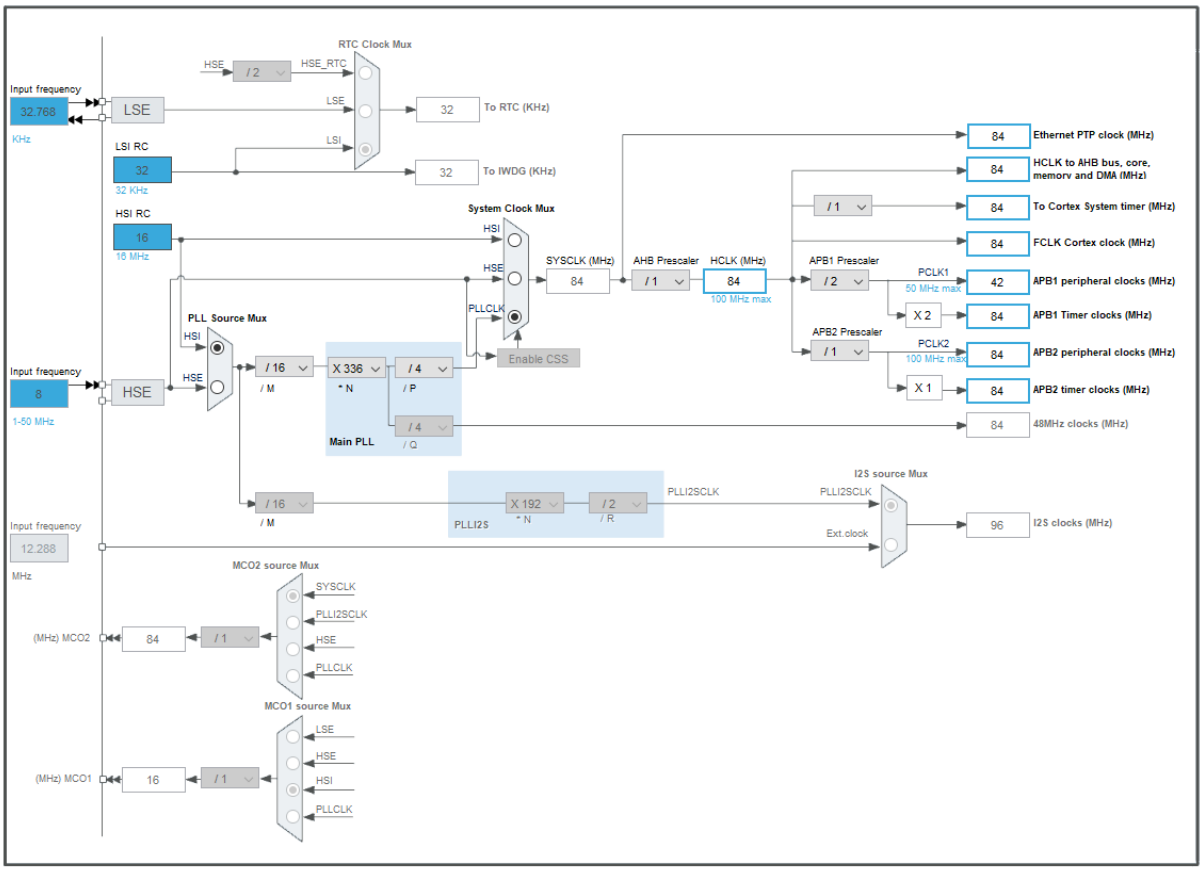
Sercem projektu został mikrokontroler z rodziny STM32, a dokładniej STM32F411RETx. Został on wyposażony w rdzeń ARM Cortex-M4 o maksymalnym taktowaniu wynoszącym 100MHz. Do pobierania danych z układu GPS został wykorzystany standard transmisji UART z prędkością wynoszącą 9600 bitów na sekundę z długością słowa wynoszącą 8 bitów. Aby zapewnić szybszą obsługę danych co jest kluczowe do pobierania czasu wykorzystano przerwania NVIC, które informują kontroler o określonym zdarzeniu. W celu pobierania i zapisywania danych do układu RTC DS3231 wykorzystano standard I2C z prędkością 400000Hz, podobnie jak w przypadku komunikacji UART tutaj także wykorzystano przerwania systemowe. Trzy przyciski zostały dodatkowo podciągnięte wewnętrznymi rezystorami Pull-up i przerwanie zostanie aktywowane gdy zbocze sygnału będzie opadające. Poniżej przedstawiam diagram wyprowadzeń dla mikrokontrolera. Jak można zauważyć nie został wykorzystany mechanizm multipleksowania. Co skutkuje zajęciem prawie wszystkich dostępnych wyjść.



Rysunek 6 Wyprowadzenia układu Źródło: Opracowanie własne

5.2 Taktowanie zegara STM32

Niestety mimo, że według specyfikacji mikrokontroler może pracować z częstotliwością do 100MHz taka częstotliwość pracy nie była możliwa do uzyskania z racji korzystania z dwóch różnych standardów komunikacji, które korzystają z dwóch różnych przebiegów czasowych. W tym celu zdecydowałem się użyć wewnętrznego rezonatora o taktowaniu 16MHz, a następnie poprzez pętlę PLCLK/HSI wygenerować zegar o częstotliwości 84MHz, który umożliwił poprawne działanie wszystkich układów.



Rysunek 7 Taktowanie procesora w programie STM32CubeIDE Źródło: Opracowanie własne

5.3 Opis układu neo-6m i technologii GPS

Do pobierania sygnału GPS (ang. Global Positioning System) został wykorzystany moduł neo-6m firmy ublox, który wspiera protokół UART i pracuje w logice 3.3V. Czas do nawiązania sygnału i synchronizowania się wynosi tylko 26 sekund od „zimnego startu", czyli stanu gdy moduł niesynchronizowań się od dłuższego czasu, a czas pobrania danych w przypadku „ciepłego startu" wynosi poniżej 1 sekundy.

Rysunek 8 Wygląd modułu GPS Źródło: https://www.u-blox.com/en/product/neo-6-series

Aby lepiej zrozumieć zasadę działania należałoby omówić jak działa system globalnej lokalizacji, jednakże celem projektu było pobranie czasu i daty, a nie lokalizacji, dlatego przytoczę tylko krótki fragment definicji:

> *Każdy satelita transmituje unikalny sygnał i parametry orbitalne, które pozwalają urządzeniom GPS dekodować i obliczać dokładną lokalizację satelity. [...] Dzięki pomiarom odległości z kilku kolejnych satelitów odbiornik może ustalić pozycję użytkownika i zanalizować ją.*

Źródło: https://bearpoint.pl/czym-jest-gps/ Dostęp: 16.06.2020 r.

Znacznie ciekawszym zagadnieniem z punktu widzenia projektu jest pobranie godziny i daty z użyciem standardu komunikacyjnego NMEA. System GPS w zależności od generacji potrafi odebrać od 11 do 58 różnych komunikatów. Dzięki standardu NMEA mamy dosyć dużą odporność na błędy, pojedyncza wiadomość zawsze będzie zawierać 82 znaki (puste informacje zostają zastąpione spacjami), komunikat zaczyna się od znaku $, a kończy \n\r. Dodatkowo każda wiadomość zawiera sumę kontrolną. W projekcie został wykorzystany komunikat $GPRMC (Recommended minimum specific GPS/Transit data) został on zaprezentowany poniżej:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

Opis poszczególnych parametrów:

- 123519 – Aktualność danych - 12:35:19 UTC,
- A – status (A – aktywny; V – nieaktywny),
- 4807.038,N – szerokość geograficzna (latitude) 48 deg 07.038' N,
- 01131.000,E – długość geograficzna (longitude) - 11 deg 31.000' E,
- 022.4 – prędkość obiektu (liczona w węzłach),
- 084.4 – kąt śledzenia/poruszania się obiektu (w stopniach) przydatny w celu określenia kierunku poruszania się obiektu, jeżeli urządzenie GPS nie jest wyposażone w kompas,
- 230394 – data (23 marca 1994),
- 003.1,W – odchylenie magnetyczne ziemi,
- *6A – suma kontrolna.

Źródło: http://www.systemy.ztt.edu.pl/pobierz/instrukcje/nmea.pdf Dostęp: 16.06.2020

Układ ublox neo-6m odbiera 19 głównych wiadomości dodatkowo posiada system ostrzegający o próbie zakłócenia sygnału.

5.4 Układ czasu rzeczywistego DS3231

W celu zarządzanie czasem rzeczywistym został wykorzystany układ DS3231 firmy Maxim Integrated pracujący w logice 3.3V zasilany napięciem 3.3V z podtrzymaniem bateryjnym CR2032, które przełącza się automatycznie w momencie utraty głównego zasilania (co pozwala na dłuższe „życie" baterii). Sam układ sterowany jest za pomocą standardu komunikacyjnego I2C pracującym z częstotliwością do 400 kHz. Komunikacja polega na zapisie i odczycie poszczególnych rejestrów znajdujących się pod określonymi adresami. Z racji na zjawisko dryfu rezonatorów kwarcowych w tym układzie został wykorzystany wewnętrzny rezonator 32.768kHz z kompensacjom temperaturową (TCXO)

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Tabela 2 Tablica adresów i rejestrów zegara RTC Źródło: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

6. Omówienie warstwy programowej

Dzięki rdzeniowy ARM CorteX możliwe jest programowanie w języku Asembler, C, C++. Projekt został napisany w języku C dzięki czemu wsad zajmuje mniej miejsca, wykorzystuje dwie biblioteki DS3231.h oraz gps_neo6.h, które zostały napisane w celu obsługi zegara RTC i GPS.

6.1 Biblioteka DS3231.h

Jest to biblioteka odpowiedzialna za komunikację z zegarem RTC. Głównie wykorzystuje metodę „HAL_I2C_Mem_Read" do odczytu danych i „HAL_I2C_Mem_Write" do zapisu danych. Dodatkowo zostały zainicjalizowane dwie struktury, która jedna przechowuje odczytane dane, a druga zapisane. Obie struktury zawierają wszystkie dane niezbędne do poprawnego funkcjonowania układu, czyli sekundę, minutę, godzinę, dzień, miesiąc, rok. Należy pamiętać, że dane są zapisane według strefy czasowej UTC.

```
typedef
struct
RTC_t {
            I2C_HandleTypeDef* huart_sds;
            uint8_t second;
            uint8_t minute;
            uint8_t hour;
            uint8_t day;
            uint8_t month;
            uint16_t year;
           } RTC_data;
```

```
typedef
struct
RTC_s {
            uint8_t second;
            uint8_t minute;
            uint8_t hour;
            uint8_t day;
            uint8_t month;
            uint16_t year;
           } RTC_setup;
```

Tabela 3 Struktury gromadzące dane.

6.2 Biblioteka gps_neo6.h

Biblioteka odpowiedzialna za odczyt danych z GPS. Dzięki znajomości maksymalnego rozmiaru wiadomości $GPRMC można było łatwo i szybko z pośród odebranych wiadomości znaleźć naszą i zdekodować używając metody stroke.

6.3 Główny program

Główny program zajmuje się zarządzaniem odczytami, a wyświetlaniem danych na wyświetlaczu siedmiosegmentowym. Zostały zdefiniowane niezbędne zmienne w celu obsługi przycisków, alarmu, a także trybu konfiguracyjnego. Kod BCD został na sztywno zaimplementowany do każdego z wyświetlaczy (numerowanych co 2, np. Hour_1, Hour_2 itp.). Aby móc zapewnić wyświetlenie liczby składającej się z dwóch cyfr został wykorzystany mechanizm modulo i dzielenia, przykładowy kod wyświetlający minuty został zaprezentowany poniżej:

```
void
DisplayMinute(int
digit) {
                        int part1 = digit / 10;
                        int part2 = digit % 10;
                        SprintMinute_1(part1);
                        SprintMinute_2(part2);
                    }
```

Za odczytywanie stanów przycisków i menu wyboru opcji konfiguracji została zrealizowana w przerwaniu HAL_GPIO_EXTI_Callback. Działa na zasadzie sprawdzenia stanu poszczególnych pinów, jeżeli pin odpowiedzialny za przycisk jeden zostanie wciśnięty zmienia się stan logiczny przerwanie systemowe to odnotowuje. Menu oferuje takie opcje jak:

- Zmiana strefy czasowej (do przodu lub do tyłu) – Aktywacja przyciskiem 1, zmiany strefy czasowej przyciskami 2 i 3. Wyjście z menu przyciskiem 1.
- Zapisanie alarmu (godzina i minuta)) – Aktywacja przyciskiem 2, zmiana godziny przyciskiem 1, zmiana minuty przyciskiem 2. Zapisanie alarmu ponownym przyciśnięciem przycisku 2.
- Wyświetlanie aktualnej daty – Aktywacja przyciskiem 3, wyjście z menu przyciskiem 3.

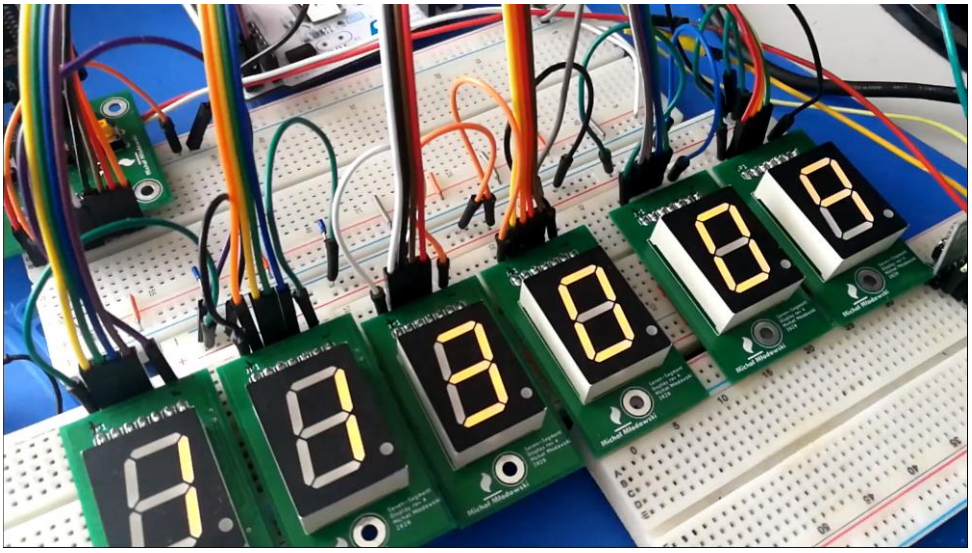
Program sygnalizuje pracę sygnalizacją dźwiękową, po tym sygnale zaczyna się działanie programu. Program pobiera datę z RTC i w tle oczekuje na nawiązanie połączenia przez system GPS, jednocześnie nie blokując działania programu. W momencie aktualizacji GPS magistrala UART wysyła przerwanie do procesora i dane zostają zapisane w jednym cyklu do układu RTC, synchronizacja z GPS sygnalizowana jest dźwiękiem. Co 100 milisekund następuje pobranie daty z RTC i odświeżenie wyświetlacza siedmiosegmentowego, co nie wypływa na opóźnienie zegara RTC ponieważ on pracuje samodzielnie.

GPS w momencie aktualizacji już nie aktualizuje ponieważ to by powodowało wyłącznie zajmowanie czasu procesora. Samo memu konfiguracji dzięki zastosowaniu przerwań systemowych także nie powoduje dryfu czasu tak samo jak aktywacja alarmu. Niestety projekt z racji ograniczonych przycisków wspiera wyłącznie jeden powtarzalny alarm (Powtarza się co określonej godzinie).

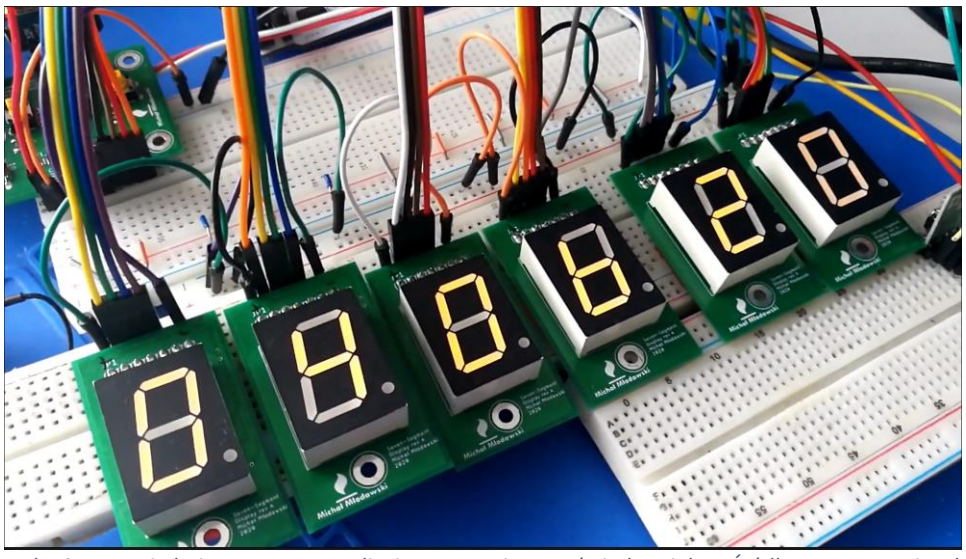
W projekcie przed możliwymi zakleszczeniami zastosowano prymitywne semafory np. aby buzzer nie wydawał dźwięków przez całą minutę po skończeniu metody zmienia stan zmiennej kontrolnej, podobnie jest z całym menu konfiguracyjnym i pobieraniem czasu z GPS.

7. Podsumowanie projektu

Projekt okazał się niezwykle interesujący, łącząc elementy sprzętowe i programowe można było osiągnąć kompaktowe rozwiązane dostosowane do potrzeb projektu. Dzięki zaprojektowaniu płytek drukowanych znacząco zmniejszono rozmiar prototypu oraz uniknięto przypadkowego połączenia złych elementów co mogłoby narazić na szkody finansowe i/lub zdrowotne. Chciałbym także podziękować dr hab. inż. Grzegorza Radomskiego za cenne wskazówki udzielone podczas prac nad projektem. Na koniec prezentuję gotowe rozwiązanie w formie zdjęć.



Rysunek 9 Zegar siedmiosegmentowy zdjęcie prezentujące wyświetlacz i godzinę Źródło: opracowanie własne



Rysunek 10 Zegar siedmiosegmentowy zdjęcie prezentujące wyświetlacz i datę Źródło: opracowanie własne