

| | | |
|------------------------------|--|--------------------------------------|
| | Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych | |
| Rok akademicki: | Rodzaj studiów*: SSI/NSI/NSM | Przedmiot (Języki Asemblerowe/SMiW): |
| 2018/2019 | NSI | SMiW Projekt |
| Imię: | Kordian | Prowadzący: OA/JP/KT/GD/BSz/GB |
| Nazwisko: | Kramek | GB |
| <i>Raport końcowy</i> | | |
| Temat projektu: | | |
| Lokalizator GPS | | |
| Data oddania: dd/mm/rrrr | | 08.09.2019 |

1. Temat projektu

Tematem projektu jest budowa Samochodowego Lokalizatora GPS. Tzn. urządzenia, które odczytuje dane z modułu GPS, następnie odczytane dane wyświetla na ekranie w formie zrozumiałej dla użytkownika oraz zapisuje historie wszystkich odczytanych informacji na karcie pamięci. Zasilanie układu dostarczane jest z instalacji elektrycznej samochodu.

2. Analiza zadania

Układ oparty jest na mikrokontrolerze Atmega328p. Przy wyborze mikrokontrolera sugerowałem się ilością posiadanej pamięci oraz jego dostępnością. Jako wyświetlacz pracuje wyświetlacz OLED o rozdzielczości 128x64 ze sterownikiem SSD1306. Komunikacja z mikrokontrolerem odbywa się przez interfejs I2C. Do określania położenia wykorzystuję moduł GPS NEO-6M z wbudowaną anteną. Moduł ten wybrałem ze względu na cenę, ponieważ lepsze alternatywy były dużo droższe. Komunikacja z mikrokontrolerem odbywa się poprzez interfejs szeregowy UART. Funkcje odpowiedzialne za obsługę przetwarzania zdań NMEA napisałem sam ze względu na to, że po załadowaniu gotowych bibliotek zabrakło mi pamięci. Czytnik kart został podłączony przez interfejs SPI. Wybierając taki sposób podłączenia, kierowałem się jednym z przykładów użycia biblioteki FatFS, którą wykorzystuję do obsługi karty microSD (przykład „generic”, zawiera również schemat podłączenia). Bibliotekę FatFS wybrałem ze względu na bezproblemowy zapis do pliku. Ponieważ alternatywna biblioteka PetitFS wymagała uprzedniego przygotowania, specjalnie spreparowanego pliku co było dość problematyczne przy testach. Programowanie mikrokontrolera odbywało się przez programator USBasp. Wszystkie elementy urządzenia pracują na napięciu 3.3V. Z założenia układ finalnie miał znaleźć się w samochodzie, w którym nie dość, że napięcie może być różne, ponieważ akumulator przeciętnej osobówki ma przeważnie około 12V, a np. samochodu ciężarowego 24V, to po odpaleniu napięcie wzrosnie z powodu działania alternatora i tak napięcie zasilania może wzrosnąć do około 14.5V dla osobówek oraz 28.8V dla samochodów ciężarowych. Z tego powodu zasilanie jest podłączone do modułu z przetwornicą impulsową LM2596, która po podaniu jej na wejście napięcia z zakresu od 3.3V do 40V przy odpowiednim ustaleniu potencjometru podaje na wyjście napięcie 3.3V którym zasilany jest układ urządzenia.

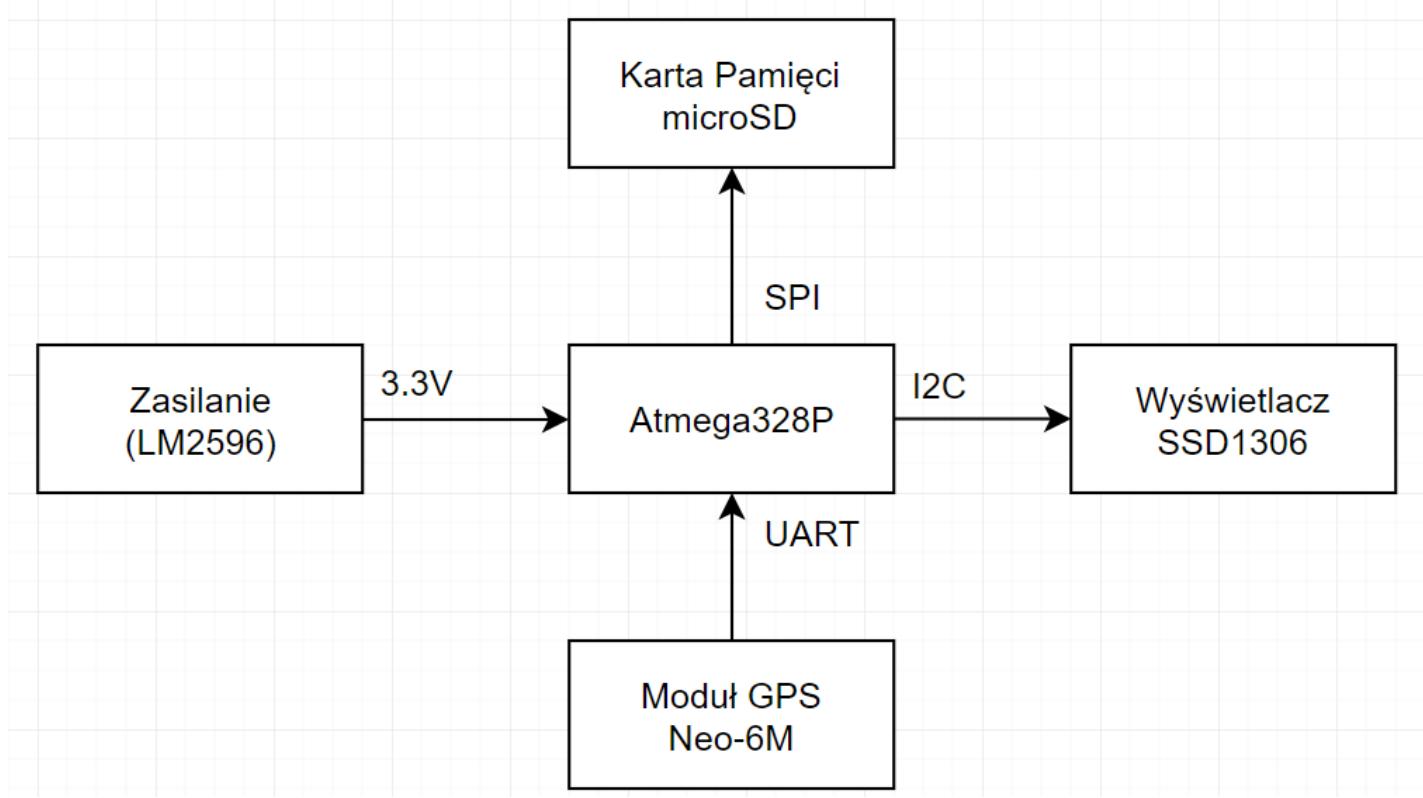
3. Specyfikacja wewnętrzna urządzenia

Układ składa się z kilku głównych elementów:

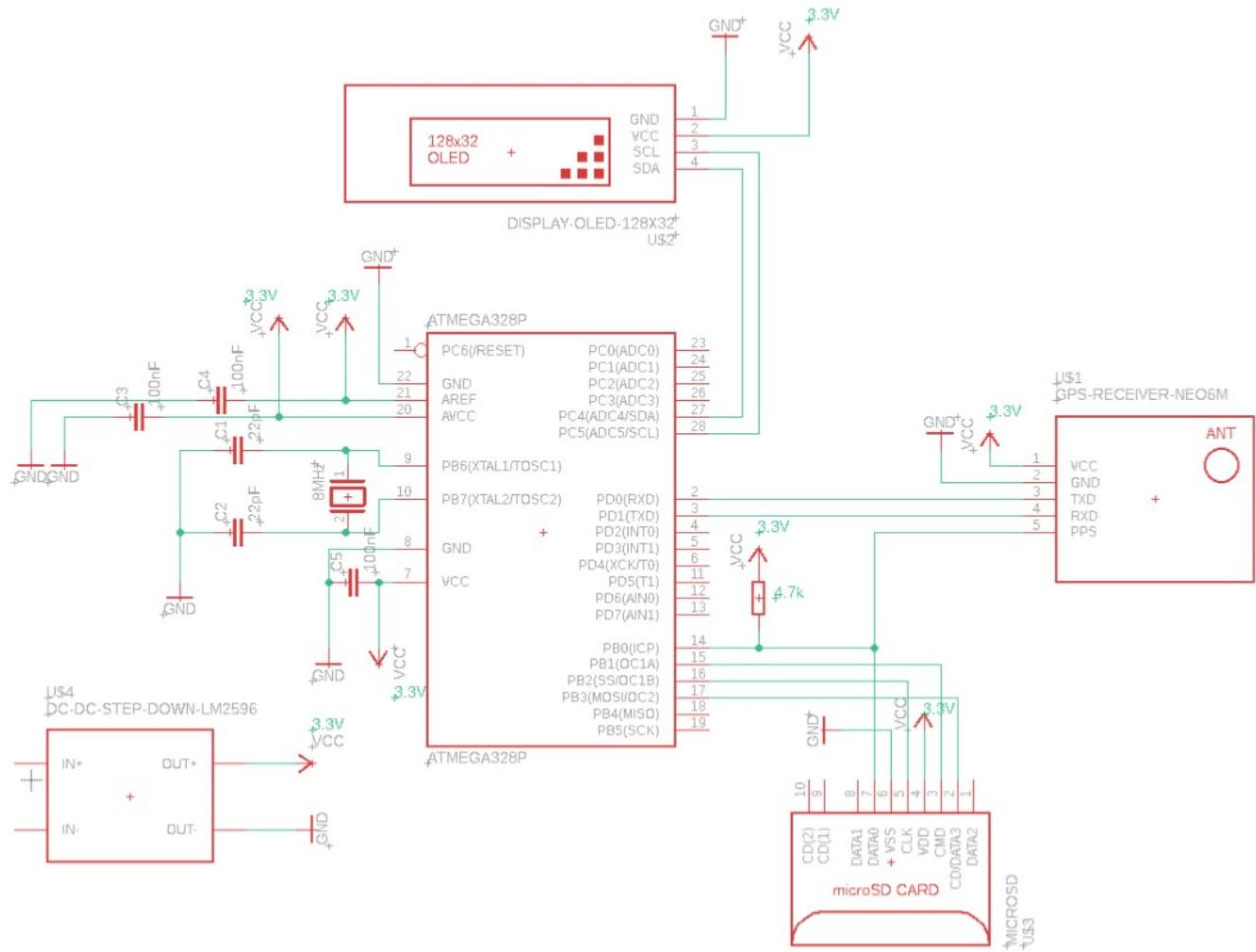
- Część odpowiedzialna za zasilanie
- Mikrokontroler
- Moduł GPS
- Czytnik kart MicroSD
- Wyświetlacz

3.1 Schematy

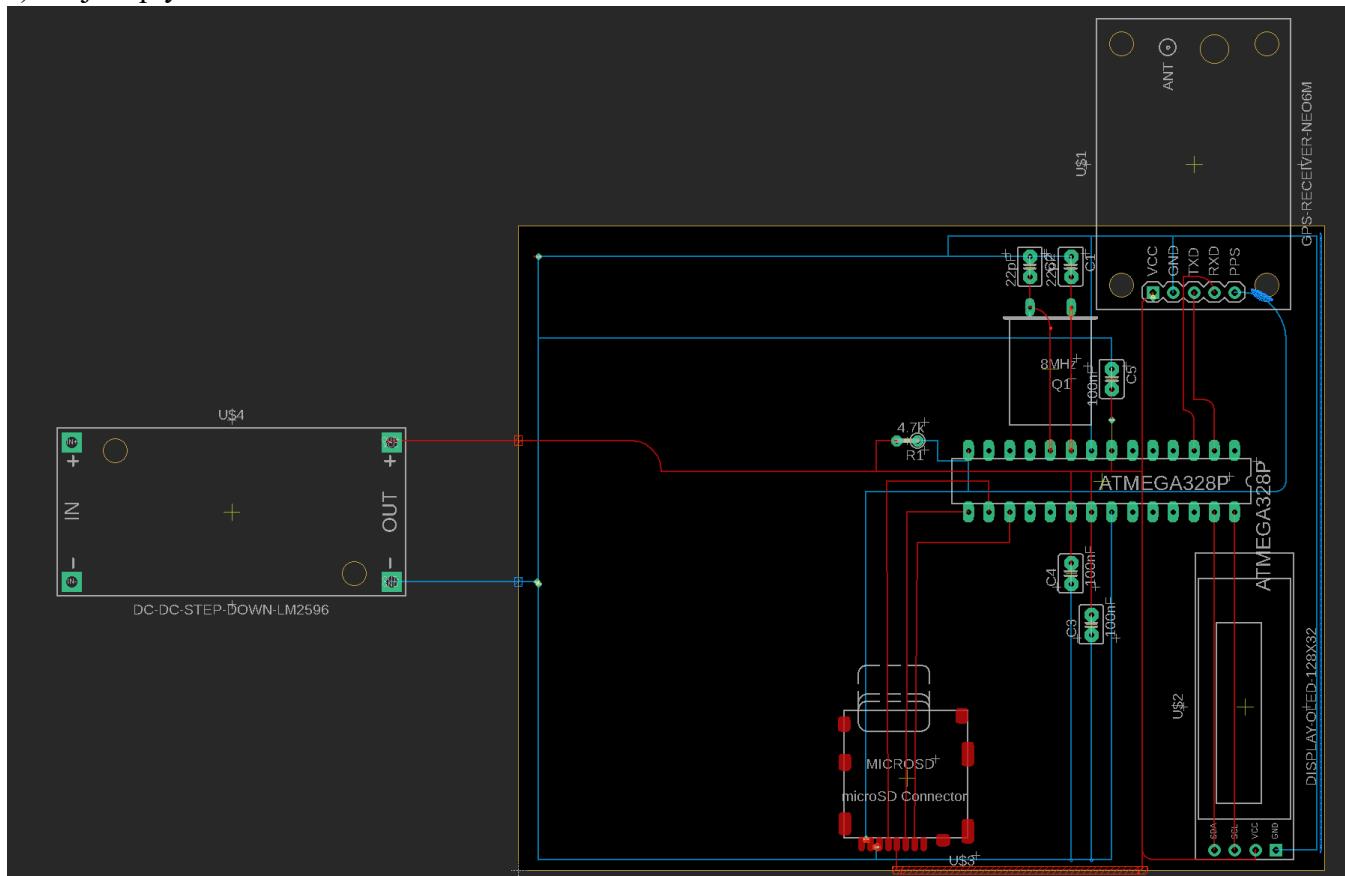
a) Schemat blokowy



b) Schemat ideowy



c) Projekt płytki



3.2 Lista elementów wykorzystanych w projekcie

1. Mikrokontroler Atmega328P
2. GPS Neo-6M
3. Wyświetlacz OLED 0,96' SSD1306
4. Slot microSD WaveShare
5. Przetwornica LM2596 DC-DC
6. Rezonator: kwarcowy; 8MHz
7. Rezystor 4.7 kΩ
8. Kondensatory 22pF (2 sztuki)
9. Kondensatory 100nF (3 sztuki)

3.3 Opis programu

Do tworzenia oprogramowania wykorzystałem środowisko Atmel Studio 7. Programowanie fusebitów odbywało się przez program AVRDUDESS 2.6 (Nakładkę graficzną na program AVRDUDE 6.3) Do obsługi karty microSD wykorzystałem bibliotekę FatFS.

Kod programu z komentarzami:

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <assert.h>
#include <math.h>

#include "ff.h"      // Dołączenie biblioteki FatFS
#include "lcd.h" // Dołączenie biblioteki do obsługi wyświetlacza

#define UART_BAUD 9600 // Standardowa prędkość transmisji modułu GPS
#define __UBRR (((F_CPU / (UART_BAUD * 16UL))) - 1)
#define GPS_BUFFOR_SIZE 80 // Rozmiar bufforu

char GPSBuffor[GPS_BUFFOR_SIZE] = {"\0"}; // Główny bufor zdań nmea
char lat[15] = {"\0"}; // Bufor szerokości geograficznej
char lon[15] = {"\0"}; // Bufor długości geograficznej
char latDir[2] = {"\0"}; // Bufor kierunku szerokości geograficznej
char lonDir[2] = {"\0"}; // Bufor kierunku długości geograficznej
char speed[15] = {"\0"}; // Bufor prędkości
float latf = 0; // Zmienna do przechowywania wartości zmienno przecinkowej szerokości geograficznej
float lonf = 0; // Zmienna do przechowywania wartości zmienno przecinkowej szerokości geograficznej

//Zmienne do obsługi karty pamięci
FATFS FatFs;
FIL Fil;

// Inicjalizacja Uarta
void USART_Init(void)
{
    UBRR0H = (uint8_t) (__UBRR >> 8);
    UBRR0L = (uint8_t) (__UBRR);
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);
    UCSR0C = (1<<UCSZ00) | (1<<UCSZ01); // Ustawienie ramki: 8bitów danych i 1 bit stopu
}

// Odbiór danych z Uarta
char USART_Receive(void)
{
    while (!(UCSR0A & (1<<RXC0)));
    return UDR0;
}

//Czyszczenie całego buffora dla zdań nmea odczytywanych z modułu GPS
void clearBuffor()
{
    int i, j;

    for(i = 0; i < GPS_BUFFOR_SIZE-1; ++i)
    {
        GPSBuffor[i] = "\0";
    }

    latf = 0;
    lonf = 0;
}
```

```

// Pobieranie/budowanie zdań nmea (Zapełnianie buffora danymi)
int getNmea()
{
    clearBuffor();          // Czyszczenie buffora z ewentualnych pozostałości
    char buffor = "";        // Buffor znaku odbieranego z uarta
    int nmeaReady = 0;        // Flaga informująca czy ramka nmea jest poprawna
    int i = 0;

    // Główna pętla budowania zdań nmea
    while(1)
    {
        buffor = USART_Receive(); // Odbieranie znaku z modułu GPSA

        //0x0D cr
        //0x0A nl
        //Sprawdzenie czy odebrany znak jest jednym ze znaków kończących
        if(buffor == (char)0x0D || buffor == (char)0x0A || i > GPS_BUFFOR_SIZE - 1 ) {

            //0x24 = $
            //0x47 = G
            //0x50 = P
            // Sprawdzenie czy pierwsze 3 znaki znajdujące się w buforze są znakami od których musi zaczynać się
            // każde zdanie nmea ($GP
            if(GPSBuffor[0] == (char)0x24 &&
                GPSBuffor[1] == (char)0x47 &&
                GPSBuffor[2] == (char)0x50)
            {
                //Jeśli tak ustawia flagę poprawności na 1 i przerywa główną pętlę odbioru danych z GPSA
                nmeaReady = 1;
                break;
            } else {
                //Jeśli zdanie jest nie poprawne czyści buffor i również przerywa
                działanie pętli pobierającej dane
                clearBuffor();
                break;
            }
        }

        //0x24 = $
        if(buffor == (char)0x24 && (GPSBuffor[0] == (char)0x24 || GPSBuffor[1] == (char)0x24))
        { //Jeśli odebrany znak to $ (znak początku dla zdania nmea)
            // Czyści bufor i przerywa pętle ponieważ oznacza to że ciąg znaków jest niepoprawny
            clearBuffor();
            break;
        }

        //0x24 = $
        //Jeśli pierwszy znak jest poprawny lub jeśli bufor jest już poprawnie zainicjalizowany
        if(buffor == (char)0x24 || GPSBuffor[0] == (char)0x24) {
            //Dodaje znak do bufora oraz inkrementuje zmenną i
            GPSBuffor[i] = buffor;
            i = i+1;
        }
    }

    //Jeśli pętla została przerwana zwracamy flagę informującą czy w bufforze znajduje się poprawne
    //zdanie nmea
    return nmeaReady;
}

//Zapis na kartę pamięci
void saveBufforToMmc()
{
    UINT bw1, bw2;
    f_mount(&FatFs, "", 0);
    // Otwiera plik w trybie do zapisu i dopisywania
    if (f_open(&Fil, "GPS.txt", FA_WRITE | FA_OPEN_APPEND) == FR_OK) {
        f_write(&Fil, GPSBuffor, GPS_BUFFOR_SIZE, &bw1); // Dopisuje zawartość bufora
}

```

```

        f_write(&Fil, "\r\n", 2, &bw2); // Dopusuje znaki nowej linii i powrotu karetki
        f_close(&Fil); // Zamknie plik
    }

    // Wyświetlanie na ekranie
    void showOnDisplay()
    {
        //Buffory znakowe
        char latStr[30];
        char lonStr[30];
        char speedStr[30];

        // Konwersja odebranych ciągów znaków na floaty
        latf = atof(lat);
        lonf = atof(lon);
        float speedkmh = atof(speed);
        int speedkmhint = speedkmh*1.852; // Przeliczenie węzłów na km/h
        speedkmhint = (speedkmhint < 10) ? 0 : speedkmhint; // Zaokrąglenie prędkości

        if(latf>0 && lonf>0) { // Jeśli współrzędne są różne od 0 0 (Z założenia urządzenie jest
            stosowane w samochodzie a punkt 0 0 występuje na oceanie atlantyckim)

            //Przeliczanie współrzędnych otrzymanych z GPSa na format który przyjmuje Google Maps
            int latdegrees = latf / 100;
            latf = latf - (latdegrees*100);
            float latminutes = latf / 60;
            unsigned long latmin = (unsigned long)(latminutes * 100000);

            //Przeliczanie współrzędnych otrzymanych z GPSa na format który przyjmuje Google Maps
            int londegrees = lonf / 100;
            lonf = lonf - (londegrees*100);
            float lonminutes = lonf / 60;
            unsigned long lonmin = (unsigned long)(lonminutes * 1000000);

            //Budowanie ciągów znaków do wyświetlenia
            sprintf(latStr, "Lat: %d.%ld %s", latdegrees, latmin, latDir);
            sprintf(lonStr, "Lon: %d.%ld %s", londegrees, lonmin, lonDir);
            sprintf(speedStr, "Speed: %d", speedkmhint);

            // Wyświetlanie przetworzonych danych na wyświetlacz
            lcd_clrscr();
            lcd_puts(latStr);
            lcd_gotoxy(0,2);
            lcd_puts(lonStr);
            lcd_gotoxy(0,4);
            lcd_puts(speedStr);

            _delay_ms(5000);
        }
    }

    // Parsowanie buffora
    void parseBuffor()
    {
        char buffor[GPS_BUFFOR_SIZE]; // Lokalny bufor nmea
        strcpy(buffor, GPSBuffor); // Kopiowanie globalnego buffora nmea do lokalnej zmiennej

        char* Message_ID = strtok(buffor, ","); // Pobieranie ID wiadomości (Klucza określającego typ
        ramki np $GPRMC, $GPGLL)

        // Jeśli ramka jest typu "$GPRMC"
        if(strcmp(Message_ID, "$GPRMC") == 0) {
            // $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
            char* Time = strtok(NULL, ","); // Pobierz czas
            char* Data_Valid = strtok(NULL, ","); // Pobierz flagę poprawności

```

```

        strcpy(lat, strtok(NULL, ","));
        // Pobierz szerokość geograficzną
        strcpy(latDir, strtok(NULL, ","));
        // Pobierz kierunek szerokości
        strcpy(lon, strtok(NULL, ","));
        // Pobierz długość geograficzną
        strcpy(lonDir, strtok(NULL, ","));
        // Pobierz kierunek długości
        strcpy(speed, strtok(NULL, ","));
        // Pobierz prędkość
        char* COG = strtok(NULL, ",");
        char* Date = strtok(NULL, ",");
        char* Magnetic_Variation = strtok(NULL, ",");
        char* M_E_W = strtok(NULL, ",");
        char* Positioning_Mode = strtok(NULL, ",");

        showOnDisplay(); // Wyświetl dane na wyświetlaczu
    }

}

int main(void)
{
    int flag = 0; // Flaga informująca o poprawności ramki nmea

    USART_Init(); // Inicjalizacja Uarta
    lcd_init(LCD_DISP_ON); // Inicjalizacja wyświetlacza

    lcd_clrscr(); // Czyszczenie wyświetlacza
    lcd_gotoxy(5,2); // Przesunięcie kurSORA do pozycji 5, 2
    lcd_puts("Starting..."); // Wypisanie komunikatu o starcie urządzenia

    //Główna pętla programu
    while(1)
    {
        flag = getNmea(); // Pobieranie zdania nmea

        if(flag == 1) { // Jeśli zdanie jest poprawne
            saveBufforToMmc(); // Zapisz buffor na kartę pamięci
            parseBuffor(); // Przeparsuj buffor + wyświetlanie
        }
        clearBuffor(); // Czyszczenie buffora
    }

    return 0;
}

```

Program działa w nieskończonej pętli:

1. Odczyt lokalizacji

Podstawową funkcjonalnością układu jest odczytywanie współrzędnych geograficznych, w których się znajduję. Odbywa się to poprzez moduł GPS (Neo-6M), który komunikuje się z mikroprocesorem poprzez interfejs UART. Konfiguracja komunikacji pomiędzy tymi elementami układu jest następująca:

Częstotliwość: 8MHz

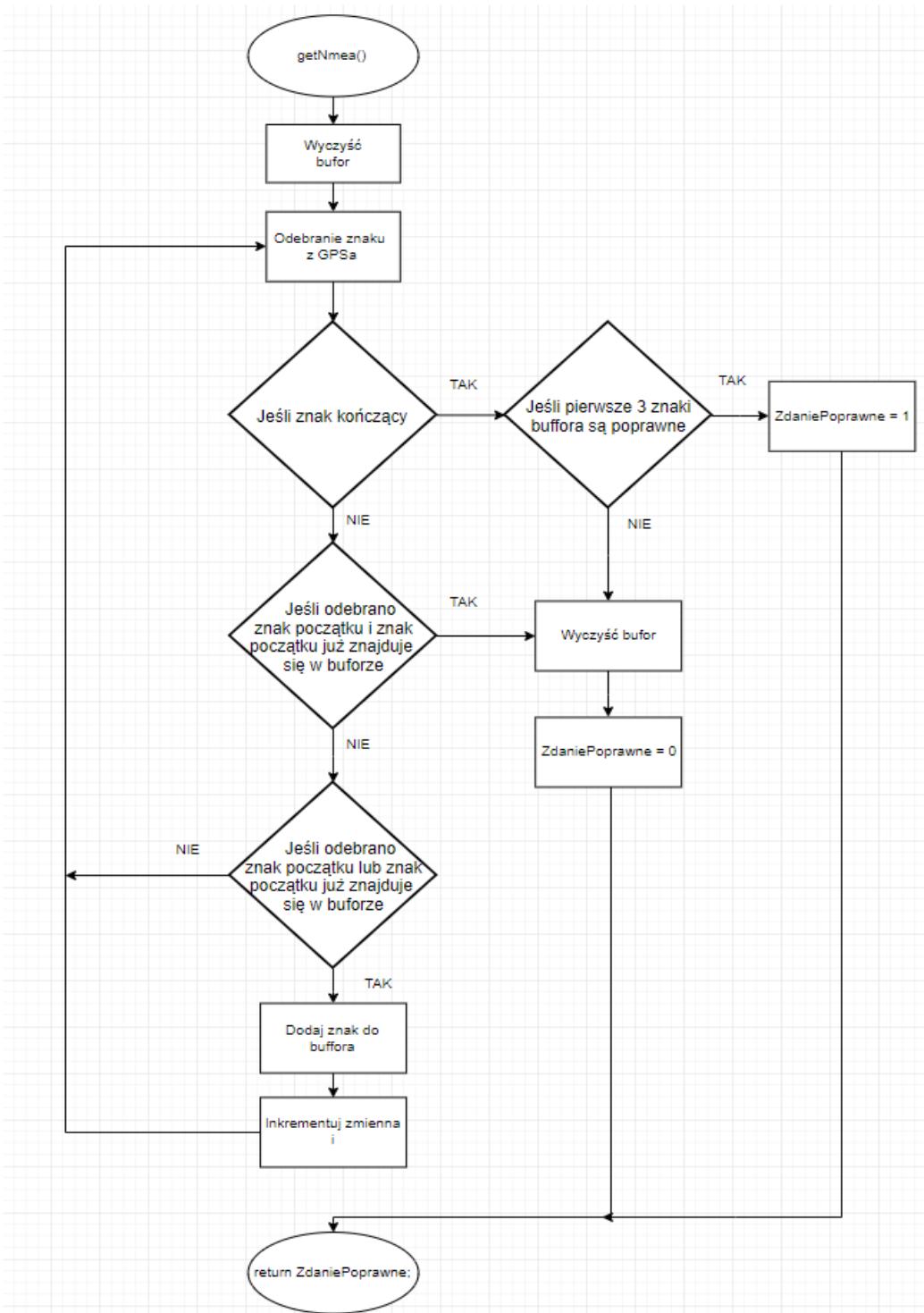
Baud rate: 9600 (Domyślna częstotliwość modułu GPS, przy której poziom błędów wynosi 0.2%)

Dane: 8 bitów danych, 1bit stopu

Za inicjalizacje komunikacji pomiędzy mikrokontrolerem a GPSem odpowiada funkcja USART_Init(), funkcja USART_Receive() natomiast pozwala na odczyt znaku, który wysyła moduł GPS.

Zdania NMEA to format danych, które wysyła moduł GPS. Zaczynają się one zawsze od znaków „\$GP” a kończą na znakach nowej linii oraz powrotu karetki. Każde zdanie NMEA może składać się maksymalnie z 80 znaków. Typ zdań, które obsługuje mój program to RMC zawiera ono między innymi informacje o współrzędnych geograficznych oraz prędkości. Przykładowe zdanie RMC wygląda następująco:
\$GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62

Główna funkcja zapełniania bufora zdaniami NMEA to `getNmea()`, której działanie opisuje poniższy schemat blokowy:



2. Konwersja współrzędnych

Współrzędne w zdaniach NMEA mają format DDMM.MM... Format ten jest nie specjalnie wykorzystywany w aplikacjach więc aby odczytane dane móc wprowadzić do np. Google Map należy je przeliczyć do formatu DD.DDDDD. Za konwersje odpowiada następujący fragment kodu:

```

int latdegrees = latf / 100;           // Wyciągamy DD
latf = latf - (latdegrees*100);        // Wyciągamy 0.MM...
float latminutes = latf / 60;          // Zamieniamy 0.MMM... na 0.DDD...
unsigned long latmin = (unsigned long)(latminutes * 100000); // I 0.DDD... na DDDDD
...
sprintf(latStr, "Lat: %d.%ld %s", latdegrees, latmin, latDir);

```

3. Konwersja prędkości

Prędkość, która znajduje się w zdaniach NMEA wyrażona jest za pomocą węzłów więc również została przeliczona na kilometry na godzinę w następujący sposób:

```
int speedkmhint = speedkmh*1.852; // Przeliczenie węzłów na km/h  
speedkmhint = (speedkmhint < 10) ? 0 : speedkmhint; // Zaokrąglenie prędkości
```

Prędkość została zaokrąglona, tak by pokazywać wartości większe niż 10km/h, ponieważ przy słabszym zasięgu, stojąc w miejscu, urządzenie pokazywało prędkość rzędu kilku kilometrów na godzinę.

4. Specyfikacja zewnętrzna układu

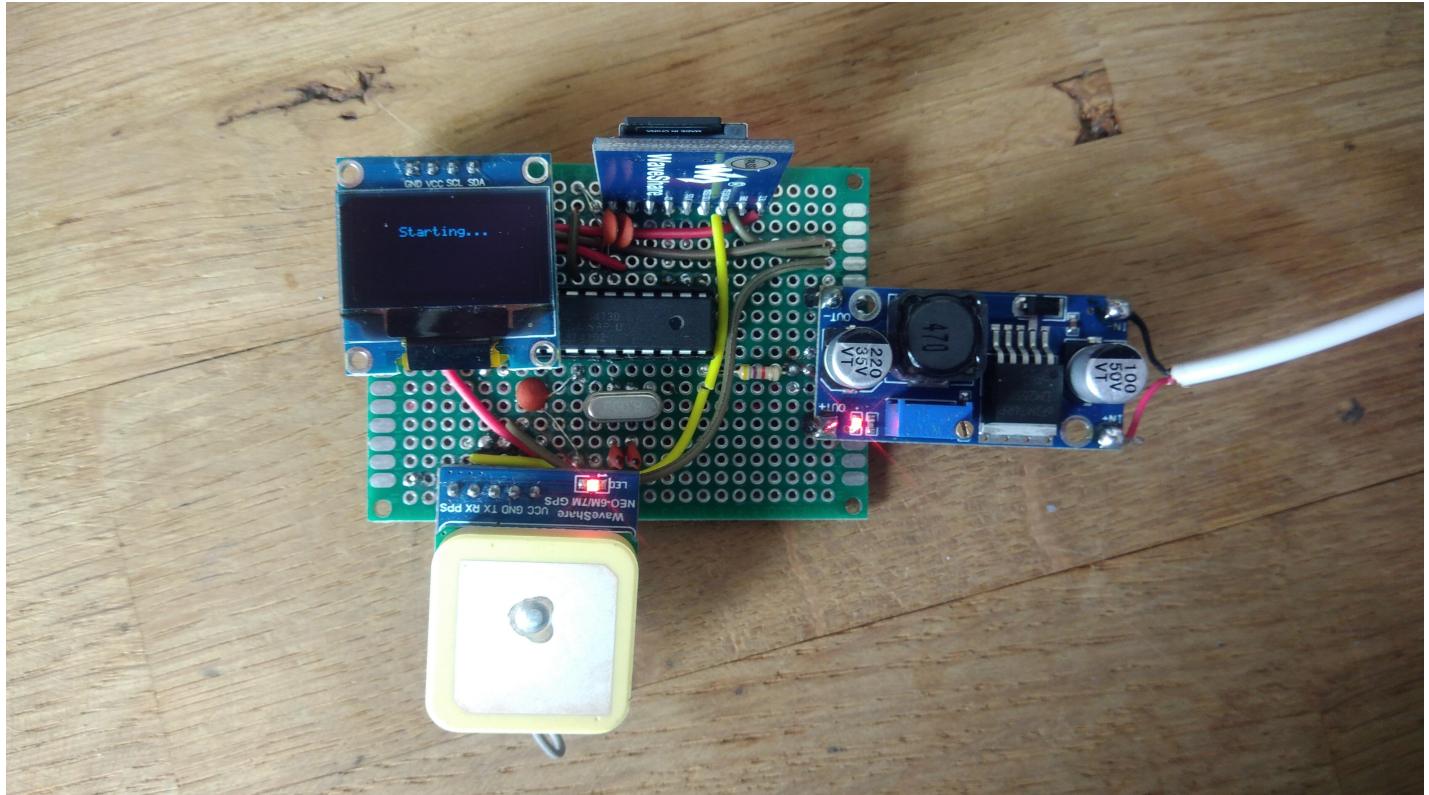
Elementem Sterującym układem jest mikrokontroler Atmega328P

Elementy wykonawcze układu to:

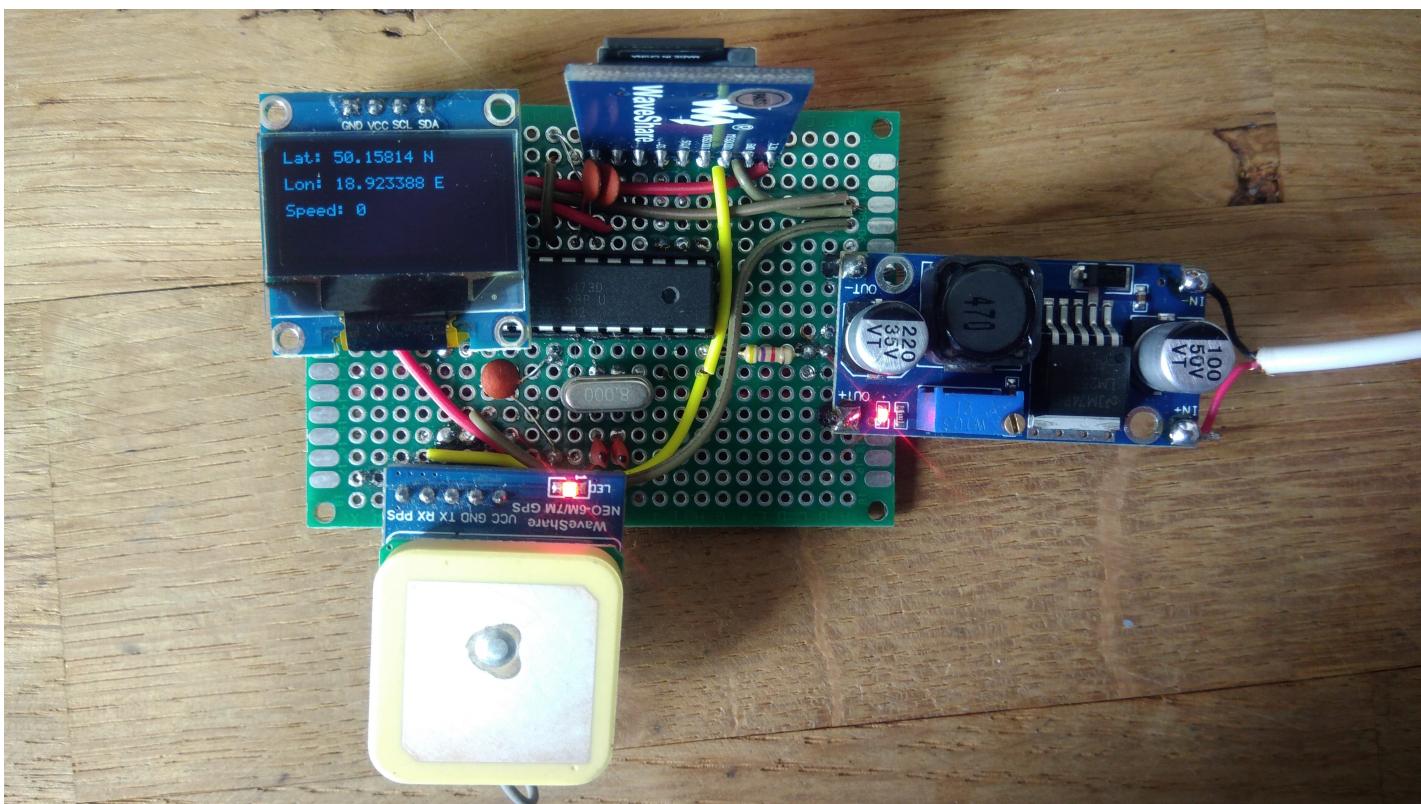
- Mikrokontroler Atmega328P
- GPS Neo-6M
- Wyświetlacz OLED 0,96' SSD1306
- Slot microSD WaveShare

4.1 Instrukcja obsługi urządzenia

Po podłączeniu urządzenia do zasilania na ekranie pokazuje się ekran powitalny.



Następnie rozpoczynainicjalizacja modułu GPS, który po złapaniu sygnału zaczyna przekazywać do układu zdania NMEA, które są zapisywane na kartę pamięci oraz po odpowiedniewybörce wyświetlane na wyświetlaczu jak na poniższym zdjęciu:



Na karcie pamięci zapisywana jest również historia w postaci zapisanych zdań NMEA

GPS.TXT - Notepad

```
File Edit Format View Help
$GPGLL,5012.56052,N,01858.42129,E,060939.00,A,A*66
$GPRMC,060940.00,A,5012.56068,N,01858.42362,E,4.931,85.15,040919,,,A*58
$GPGLL,5012.56239,N,01858.42880,E,060945.00,A,A*68
$GPRMC,060946.00,A,5012.56230,N,01858.42922,E,0.184,,040919,,,A*7A
$GPRMC,060952.00,A,5012.56178,N,01858.43122,E,0.056,,040919,,,A*77
$GPGSV,3,3,10,29,59,070,44,31,32,229,28*78
$GPRMC,060958.00,A,5012.56424,N,01858.44359,E,10.463,73.35,040919,,,A*67
$GPGLL,5012.56792,N,01858.47668,E,061003.00,A,A*6B
$GPRMC,061004.00,A,5012.56923,N,01858.48330,E,15.847,73.35,040919,,,A*60
$GPGLL,5012.57347,N,01858.51261,E,061009.00,A,A*66
$GPRMC,061010.00,A,5012.57364,N,01858.51725,E,8.969,77.36,040919,,,A*53
$GPGLL,5012.57734,N,01858.54331,E,061015.00,A,A*6A
$GPRMC,061016.00,A,5012.57808,N,01858.54938,E,14.204,78.63,040919,,,A*61
$GPGLL,5012.58078,N,01858.57697,E,061021.00,A,A*67
$GPRMC,061022.00,A,5012.58129,N,01858.58243,E,12.231,78.19,040919,,,A*65
$GPGLL,5012.58543,N,01858.60685,E,061027.00,A,A*6B
$GPRMC,061028.00,A,5012.58593,N,01858.61184,E,11.643,78.72,040919,,,A*67
$GPGLL,5012.59039,N,01858.64292,E,061033.00,A,A*61
$GPRMC,061034.00,A,5012.59101,N,01858.64726,E,9.616,79.33,040919,,,A*52
$GPGLL,5012.59431,N,01858.67118,E,061039.00,A,A*65
$GPRMC,061040.00,A,5012.59521,N,01858.67542,E,9.812,77.13,040919,,,A*52
$GPGLL,5012.61139,N,01858.67814,E,061045.00,A,A*6D
$GPRMC,061046.00,A,5012.61606,N,01858.67658,E,17.588,346.95,040919,,,A*5F
```

5. Montaż układu

Pierwsza wersja układu została zbudowana na płytce stykowej. Układ w tej wersji zasilany był modułem zasilającym do phytów stykowych MB102 – 3,3V. Wersja finalna układu została zmontowana na płytce uniwersalnej. Układ montowany był przy pomocy lutownicy kolbowej. Mikrokontroler został zamontowany na podstawce a wrażliwe elementy takie jak wyświetlacz czy moduł GPS. Dzięki temu zmniejszyłem ryzyko ich uszkodzenia podczas montażu.

6. Wnioski

Wykonane urządzenie to mój pierwszy tego typu projekt, przy którym wiele się nauczyłem. Nie obyło się bez problemów i zmian w schematach, ponieważ dobrane elementy okazywały się ze sobą nie współpracować.

Pierwszym problemem, jaki napotkałem przy implementacji zapisu na kartę pamięci. Okazało się, że karty pamięci, które posiadałem w domu (8GB oraz 6GB) nie działają pomimo ustawienia biblioteki flagi odpowiedzialnej za zapis na karty powyżej 2GB. Po zakupie kart 512MB układ zaczął działać.

W kolejnym kroku wziąłem się za implementację obsługi wyświetlacza z interfejsem SPI. Niestety po próbie połączenia uprzednio przygotowanego programu obsługującego zapis na kartę pamięci okazało się, że moduły te ze sobą nie współpracują. Po wymianie wyświetlacza na taki, z którym komunikacja odbywała się przez interfejs I2C. Pomimo wielu zmian bibliotek i odpowiedniej ich konfiguracji wyświetlacz nie wyświetlał tylko połowę tekstu:



Dopiero po zakupie tego samego wyświetlacza w wersji 128x64 i ponownym przekonfigurowaniu biblioteki na odpowiednie wymiary, układ zaczął wyświetlać poprawny tekst.

Ostatnim krokiem było podpięcie modułu GPS, tu obyło się bez problemów technicznych. Problemem okazał się natomiast finalny rozmiar programu który po załadowaniu bibliotek do obsługi karty pamięci, wyświetlacza oraz parsowania zdań NMEA ważył o wiele za dużo. Odchudzanie projektu zacząłem od próby zmiany biblioteki FatFS na PetitFS która miała być dużo lżejsza. Niestety okazało się że lepiej radzi sobie ona z odczytem, a w przypadku zapisu musiałem przygotowywać specjalnie spreparowany plik tekstowy co było dość uciążliwe i nie spodobało mi się to rozwiązanie. Następnie spróbowałem odchudzić bibliotekę do parsowania zdań NMEA ale nie przynosiło to oczekiwanych rezultatów więc finalnie napisałem swoją wersję obsługi tego typu danych.

6.1 Testy

Po zmontowaniu układu zamontowałem go do samochodu i przeprowadziłem testy w postaci przebycia pewnej trasy, a następnie sprawdziłem na Google Maps czy pozycja wyświetlana przez urządzenie faktycznie pokrywa się z moją aktualną pozycją.

7. Wnioski końcowe

Zrealizowany przeze mnie projekt pt. „Budowa Lokalizatora GPS” spełnia wszystkie założenia, które zostały postawione na początki. Urządzenie pozwala odczytać swoją lokalizację oraz zapisuje na karcie pamięci historie odwiedzonych miejsc. Lokalizacja, którą pokazuje uwędzenie jest zadziwiająco dokładna jak na fakt, że wykorzystany moduł był jednym z najtańszych dostępnych na rynku.

8. Literatura

1. Mirosław Kardaś „Mikrokontrolery AVR. Język C - podstawy programowania.”
2. http://elm-chan.org/fsw/ff/00index_e.html
3. <https://github.com/Sylaina/oled-display>