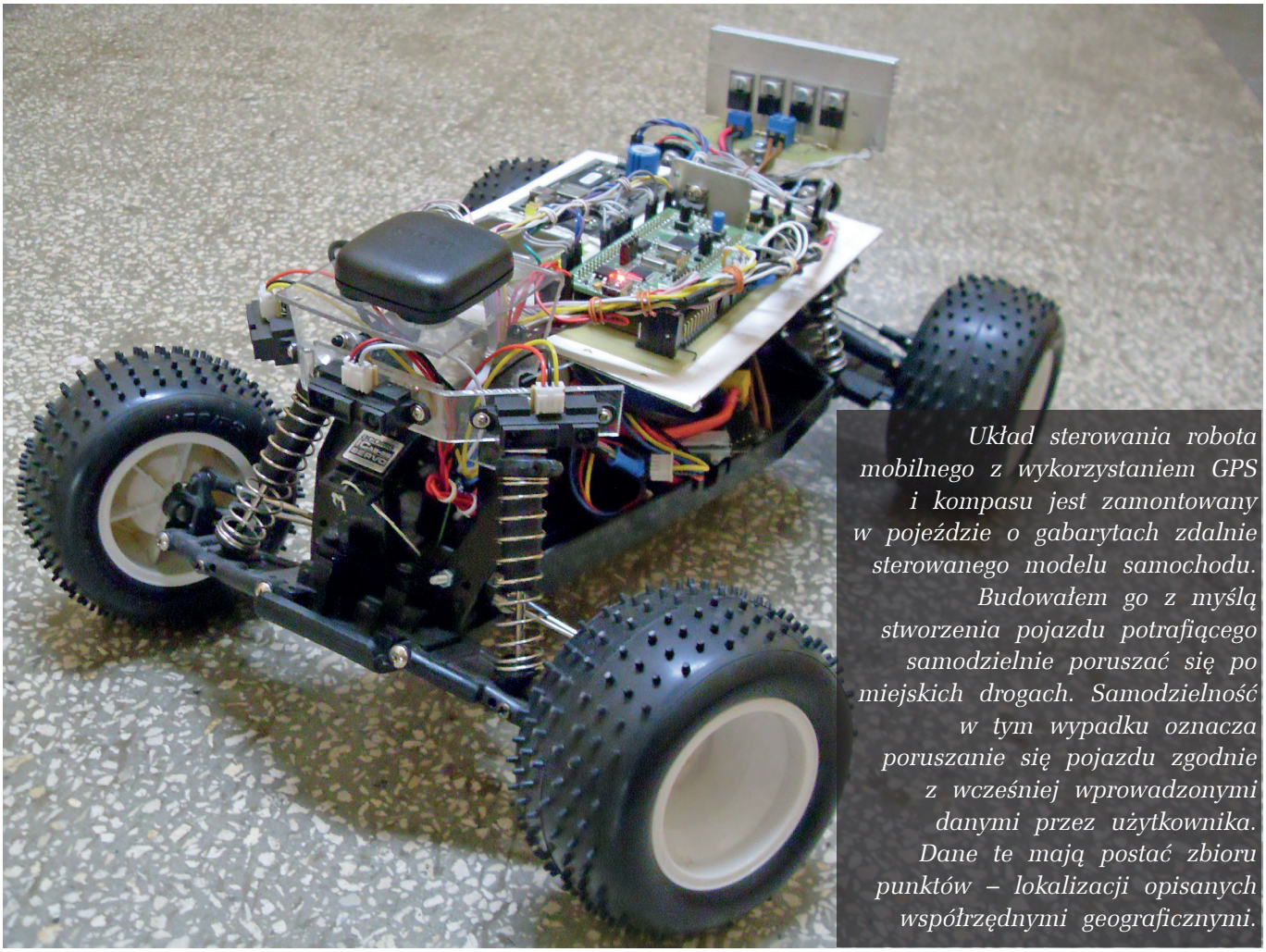
Knight Rider

Układ sterowania robota mobilnego z wykorzystaniem GPS i kompasu

Projekt 217



Układ sterowania robota mobilnego z wykorzystaniem GPS i kompasu jest zamontowany w pojeździe o gabarytach zdalnie sterowanego modelu samochodu. Budowałem go z myślą stworzenia pojazdu potrafiącego samodzielnie poruszać się po miejskich drogach. Samodzielność w tym wypadku oznacza poruszanie się pojazdu zgodnie z wcześniej wprowadzonymi danymi przez użytkownika. Dane te mają postać zbioru punktów – lokalizacji opisanych współrzędnymi geograficznymi.

Oto założenia jakie przyjąłem na początku budowy pojazdu:

- Budowa mechaniczna pozwalająca na poruszanie się po jezdni.
- Pojazd ma być zwrotny, potrafi pokonać przeszkody o wysokości około 3 cm.
- Autonomiczny – zasilanie bateryjne, czujniki przeszkód, logika omijania przeszkód.

Aby osiągnięcie celu było mierzalne, weryfikacja działania pojazdu będzie polegała na samodzielnym przejechaniu przez robota ustalonej wcześniej, prostej trasy złożonej z czterech liniowych odcinków łączących wskazane wierzchołki prostokąta.

Budowa pojazdu

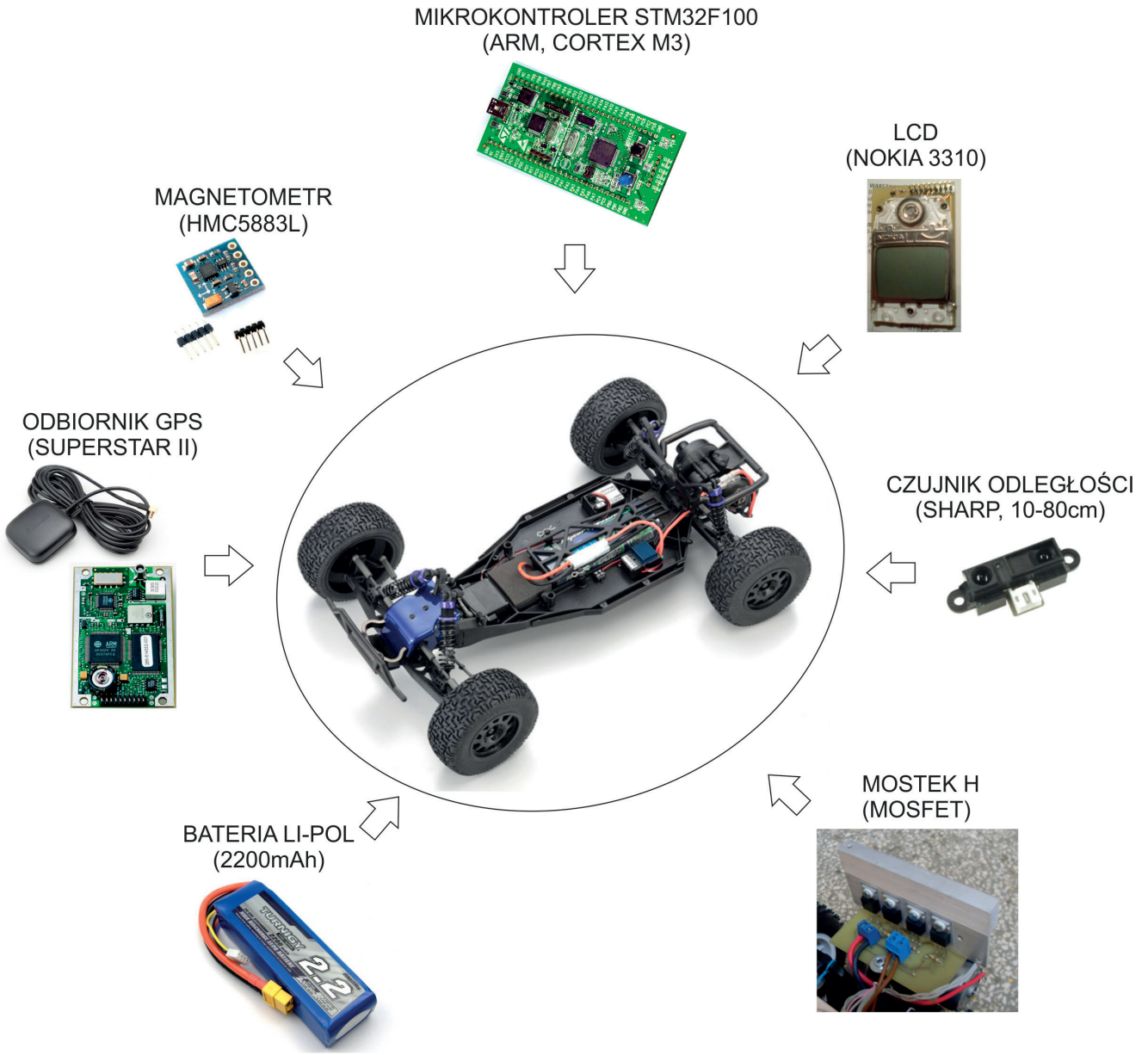
W projekcie wykorzystałem podwozie od używanego zdalnie sterowanego pojazdu RC, w którego skład wchodzą:

- cztery koła, każde amortyzowane niezależnie,
- silnik prądu stałego, szczotkowy, o mocy około 150 W z przekładnią,
- układ skręcania kół napędzany serwomechanizmem modelarskim.

Wszystkie czujniki, bateria, sterownik i elektronika sterująca znajduje się na podwoziu pojazdu, co schematycznie pokazano na **rysunku 1**.

Zasilanie silnika i układu sterującego zrealizowałem stosując nowoczesną baterię LiPo składającą się z dwóch ogniw o pojemności 2200 mAh i napięciu 3,7 V połączonych szeregowo. W efekcie uzyskuje się napięcie 7,4 V i pojemność 2200 mAh. Taki pakiet pozwala na około 20…30 minutową jazdę.

Sterowanie pojazdu oraz obsługa czujników są realizowane przez jednostkę centralną, którą stanowi mikrokontroler STM32F100RB. Dla uproszczenia części elektronicznej zastosowałem moduł ewaluacyjny STM32DISCOVERY (**fotografia 2**). Ten moduł, oprócz wspomnianego mikrokontrolera, ma wbudowany programator ST LINK komunikujący się z komputerem PC poprzez port USB. Programator wykorzystuje interfejs SWD (dwie linie sygnałowe).



Rysunek 1. Schemat blokowy pojazdu

Płytka STM32DISCOVERY została osadzona w zaprojektowanej i wykonanej przeze mnie płytce bazowej, rozszerzającej liczbę wyprowadzeń związanych z zasilaniem (do zasilania wielu czujników). Na płytce rozszerzającej znajduje się również liniowy stabilizator napięcia, prosta klawiatura 4-przyciskowa, 7 diod LED oraz brzęczyk.

Program sterujący napisałem w języku C, zgodnie z algorytmem pokazanym schematycznie na **rysunku 3**. Konfiguracja układów peryferyjnych mikrokontrolera jest zrealizowana poprzez bezpośrednie zmiany wartości odpowiednich rejestrów przypisanych do danego układu peryferyjnego, nie stosowałem biblioteki dostarczonej przez firmę STMicroelectronics.

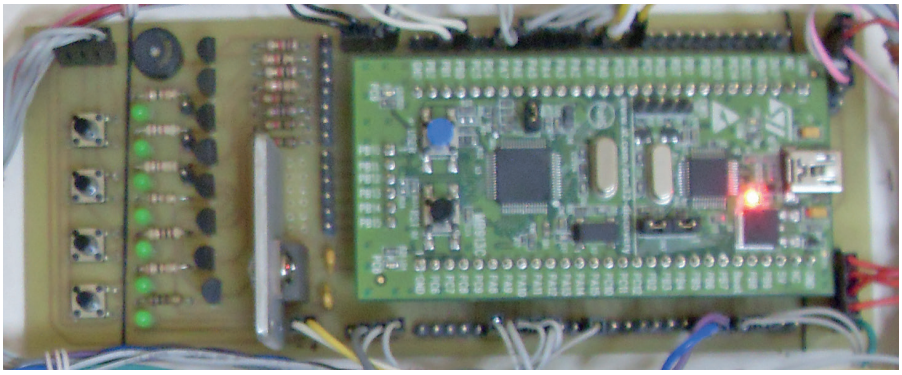
Odbiornik GPS

W projekcie użyłem modułu Superstar II kanadyjskiej firmy NovTel (**fotografia 4**). Jest to 12-kanałowy odbiornik GPS. Po dołączeniu zasilania (3,3 V) automatycznie wyszukuje i utrzymuje połączenie z widocznymi satelitami. Gdy odpowiednia liczba satelitów znajduje się w zasięgu i moduł prawidłowo odbiera sygnał, wysyła do użytkownika informację o pozycji w trójwymiarowej przestrzeni oraz wartość prędkości, z jaką się porusza.

W moim projekcie użyłem podstawowej aplikacji modułu, dołączając jedynie napięcie zasilające, port szeregowy COM1 oraz zasilanie anteny aktywnej. Zastosowany odbiornik GPS wymaga anteny zewnętrznej. Moduł ma dla niej gniazdo MCX. Zastosowałem antenę aktywną zewnętrzną Garmin GA 25MCX.

Używany w projekcie moduł GPS ma dwa porty szeregowe – COM1 oraz COM2. Port COM2 służy do wymiany danych związanych z DGPS – nie jest używany w tym projekcie. Port COM1 to interfejs szeregowy UART służący do transmisji danych o poziomach logicznych 0…3,3 V. Właśnie przez niego odbiornik komunikuje się dwukierunkowo z urządzeniem zewnętrznym. Format oraz rodzaj danych można wybrać na etapie konfigurowania.

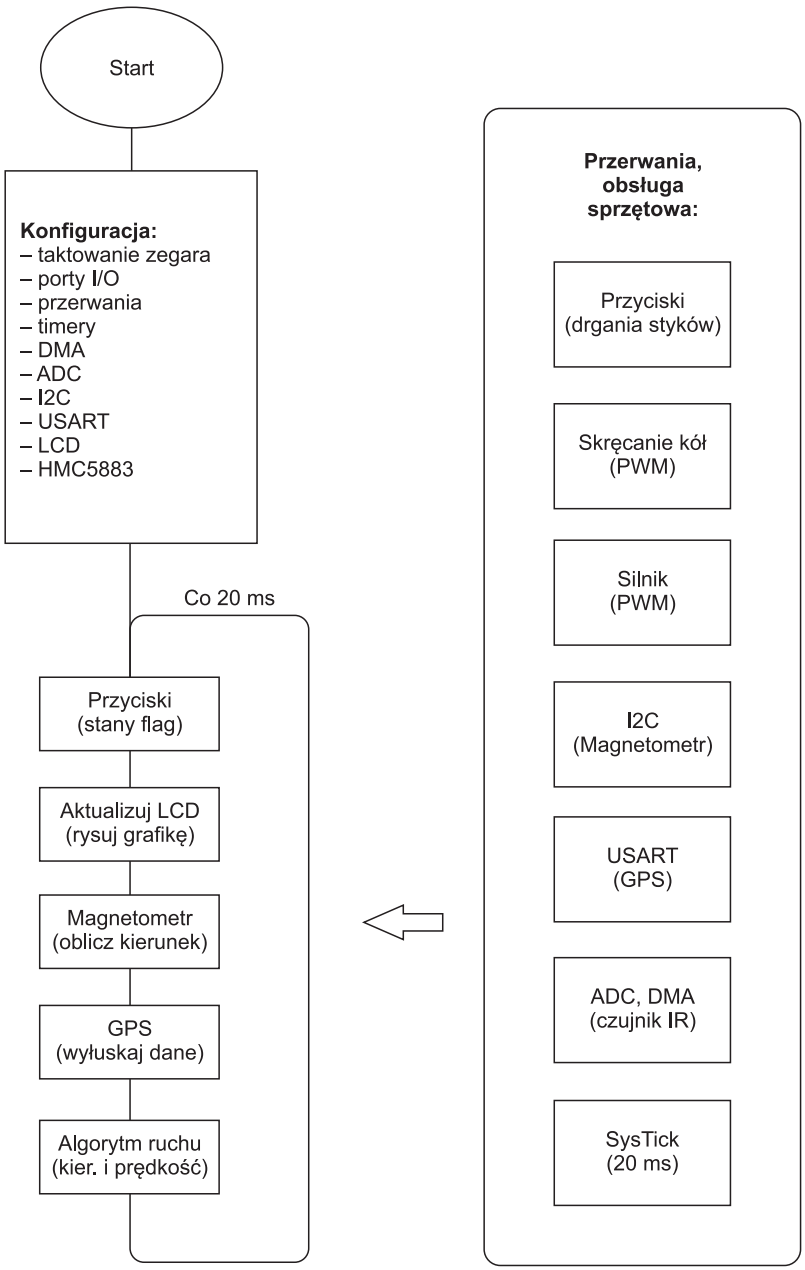
Już przy pierwszym uruchomieniu moduł Superstar automatycznie wyszukuje



Fotografia 2. Wygląd płytki sterujacej

satelity i przesyła komunikaty zawierające koordynaty GPS. Jednak, aby móc w pełni wykorzystać możliwości modułu, warto skorzystać z dedykowanego dla tego odbiornika programu dla komputera PC poprawiającego czytelność odbieranych danych oraz umożliwiającego konfigurowanie modułu.

W celu komunikacji między odbiornikiem a komputerem PC zastosowałem konwerter USB/UART. Ponieważ moduł GPS pracuje zasilany napięciem 3,3 V, wybrałem przejściówkę z układem scalonym FT232RL. Wspomniany przeze mnie program współpracujący z modułem GPS – Superstar 2 – jest udostępniony przez producenta za darmo i nosi nazwę StarView. Odbiornik GPS ma możliwość nadawania wiadomości w formacie binarnym lub zgodnie z protokołem NMEA0183. Aby zmienić metodę wysyłania danych przez odbiornik GPS, wystarczy wywołać odpowiednią pozycję z menu programu, w tym wypadku *Configure COM1 Port Mode*. Po wcześniejszym połączeniu odbiornika z komputerem, po zaakceptowaniu zmiany ustawień, aplikacja wyśle do odbiornika odpowiednią komendę odpowiedzialną za ustawienie konfiguracji. Dane te zostaną przesłane w formacie binarnym, po czym odbiornik natychmiast zmieni format wysyłanej wiadomości z binarnego na NMEA. Powtórne przesłanie wiadomości w formacie binarnym okaże się nieskuteczne, ponieważ odbiornik komunikuje się już wyłącznie w trybie NMEA. Zmiany zostają zachowane trwale, przerwa w zasilaniu nie zeruje konfiguracji.



Rysunek 3. Algorytm funkcjonowania programu

Program StarView (**rysunek 5**) korzysta ze wszystkich możliwości odbiornika Superstar 2. Oprócz podstawowej konfiguracji przedstawionej powyżej możemy skonfigurować różnicowy GPS (DGPS), odczytywać położenie widocznych satelitów. Program pozwala na graficzne przedstawienie odczytanych przez odbiornik punktów w przestrzeni, może się to okazać bardzo pomocne przy badaniu dokładności odbiornika GPS w terenie.

Przedstawione szerokie możliwości programu StarView bazują na oprogramowaniu znajdującym się w odbiorniku GPS. Moduł rozpoznaje blisko 35 komend binarnych, przy czym wiele z nich jednorazowo zmienia całą grupę ustawień. Oprócz rozpoznawania wejściowych instrukcji moduł potrafi generować 24 wiadomości wyjściowe, zależnie od aktualnej konfiguracji. Komendę konfigurującą port COM1 pokazano na **rysunku 6**. Po odebraniu komendy, odbiornik przechodzi w tryb nadawania wiadomości w formacie protokołu NMEA. Taki komunikat jest generowany z częstotliwością 1 Hz.

Przykład wiadomości otrzymywanej po przejściu na tryb NMEA po połączeniu z satelitami, można go podglądać np. w terminalu portu szeregowego, gdy odbiornik podłączony jest do komputera. Przykładowy komunikat może wyglądać następująco: *$GPGGA,012338.61,5619.2837,N,17235.8964,E,1,05,2.3,34.2,M,-17.5,M.* Informacje zawarte w tej wiadomości:

- godzina: 01:23:38.61,
- szerokość geograficzna: 56°19,2837' N,
- długość geograficzna: 172°35,8964' E,
- jakość: jest sygnał,
- liczba satelitów: 5,
- wysokość n.p.m.: 34,2 m.

Moduł GPS wysyła łańcuch znaków, z którego program sterujący robota odczy-

Listing 1. Funkcja obliczająca kąt przemieszczania się pojazdu

```
int DestAngle(void)
{
    float dx, dy, pi, AngleDegrees;
    pi = 3.14159265359;
    dy = DestLatitude() -  GpsLatitude();
    dx = cos((pi/180)*GpsLatitude())*(DestLongitude()-GpsLongitude());
    AngleDegrees = atan2(dx, dy)*(180/pi);
    return AngleDegrees;
}
```

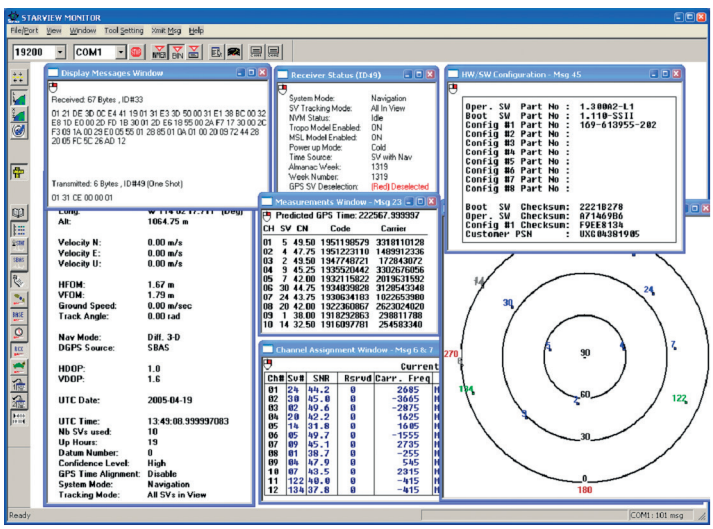
Fotografia 4. Odbiornik SuperStar II

tuje długość i szerokość geograficzną aktualnego położenia. Współrzędne geograficzne mierzone stopniach, minutach i sekundach kątowych są nieprzyjazne dla programisty korzystającego z decymalnego bądź heksadecymalnego systemu liczbowego. Do obliczeń wykorzystałem format, który przedstawia współrzędne geograficzne w zapisie decymalnym (dziesiętnym). Na przykład, zamiast 41°24'12,1674''; 2°10'26,508''używam 41,40338 oraz 2,17403. Mapy Google i odbiornik GPS generują współrzędne geograficzne wskazanego punktu w formacie dziesiętnym, dzięki czemu nie ma potrzeby konwertowania współrzędnych geograficznych przed ich wprowadzeniem do programu bądź konwertowania podczas działania programu. Punkt aktualny (A) oraz punkt docelowy (B) tworzą prostą o pewnym kącie nachylenia (α) względem południka, jak pokazano na **rysunku 7**. Właśnie pod tym kątem pojazd powinien przemieszczać się aż do osiągnięcia punktu zadanego. Ponieważ odległości pokonywane przez pojazd są nieznaczne w stosunku do powierzchni Ziemi, zastosowałem upraszczające założenie mówiące, że powierzchnia Ziemi jest płaszczyzną. Takie założenie nie wprowadzi znaczących błędów a pozwoli zredukować ilość wykonywanych przez mikrokontroler obliczeń. Funkcję obliczającą kąt, pod którym powinien podążać zamieszczono na **listingu 1**. Funkcje *GpsLatitude() oraz GpsLongitude()* zwracają aktualne położenie, natomiast *DestLongitude()* oraz *DestLatitude()* zwracają współrzędne geograficzne punktu docelowego. Wynikiem jest zmienna całkowita wyrażona w stopniach, której wartość mieści się w przedziale (0, 360).

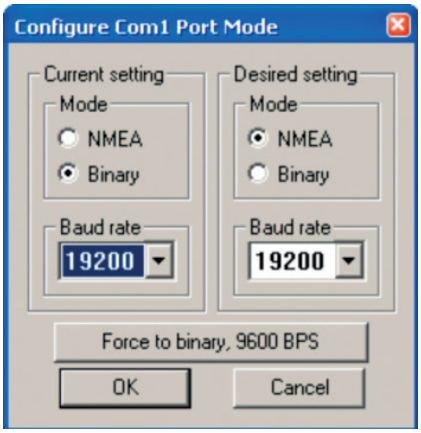
Na **rysunku 8** przedstawiłem graficznie rozrzut, z którym są odczytywane koordynaty GPS, moduł odbiornika przez cały okres testu (100 sekund) znajdował się w tym samym miejscu. Punkt koloru czerwonego jest wartością średnią wszystkich stu odczytów (punkty koloru niebieskiego).

Kompas

Znając koordynaty GPS przemieszczającego się pojazdu możliwe jest wyznaczenie aktualnej orientacji względem głównych kierunków świata. Gdy jednak pojazd został dopiero uruchomiony bądź przejechał krótki dystans opieranie się na współrzędnych geograficznych nie pozwoli mi wyznaczyć



Rysunek 5. Ekran programu StarView



Rysunek 6. Parametry COM1

poprawnej orientacji względem biegunów geograficznych. Dlatego kluczowym elementem systemu obok modułu GPS jest czujnik pola magnetycznego pozwalający wyznaczyć aktualny kierunek ustawienia pojazdu.

Czujnik HMC5883L firmy Honeywell to montowany powierzchniowo moduł z interfejsem I²C, zaprojektowany do mierzenia pól magnetycznych o słabym natężeniu. Zawiera 12-bitowy przetwornik A/C pozwalający na uzyskanie dokładności rzędu 2 stopni podczas pracy jako kompas. Ponieważ układ jest montowany powierzchniowo i ma obudowę stosunkowo trudną do przylutowania w warunkach domowych, postanowiłem zakupić płytkę drukowaną z już przylutowanym układem i wymaganymi elementami zewnętrznymi.

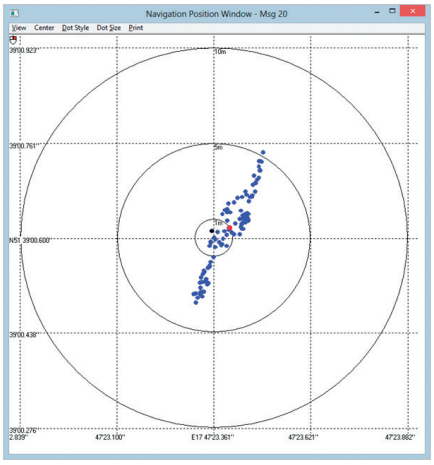
Czujnik pola magnetycznego przesyła do mikrokontrolera trzy wartości zmierzonego pola magnetycznego wyrażanego w Gaussach, odpowiednio dla trzech prostopadłych wzajemnie kierunków – osi X, Y oraz Z. Wielkość zmierzoną w osi Z ignoruję, gdyż jest prostopadła do powierzchni Ziemi a zatem wartości pola magnetycznego w tym kierunku są pomijalne. Do wyznaczenia kierunku świata, nie będą mnie interesowały konkretne wartości, ale stosunek wartości dla osi X i Y, który występuje w funkcji arcustangens. Do prawidłowego działania funkcji niezbędne są zmienne *xmin*, *xmax*, *ymin* oraz *ymax*. Określają one zakres wartości wektorów X, Y, które są odbierane z czujnika. Wyznaczyłem je na podstawie danych odczytanych z magnetometru obracanego wokół pionowej osi, gdy leżał się na płaskiej powierzchni. Podczas wyznaczania kierunku istotnym elementem okazuje się wartość deklinacji magnetycznej w danym miejscu na Ziemi i czasie, którą kompensujemy otrzymany wstępnie pomiar. W Polsce wynosi ona około 4 stopnie w kierunku wschodnim. Na **listingu 2** zamieszczono kod funkcji odpowiedzialnej za wyznaczanie kierunku.

Czujnik HMC5882 komunikuje się z mikrokontrolerem zarządzającym pojazdem poprzez interfejs I²C, częstotliwość na linii SCLK wynosi 100 kHz. Magnetometr może

Listing 2. Funkcja służąca do wyznaczania kierunku

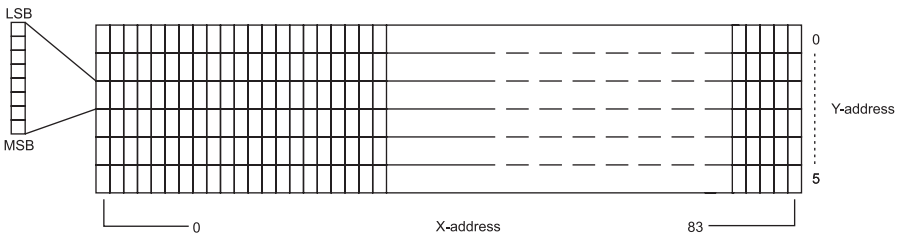
```
uint16_t HMC5883_Read(void)
{
    uint8_t address[1], data[6];
    signed short data_fine[2], offsetx, offsety, x, y;
    static signed short xmin=-1 ,xmax=543,ymin=-95,ymax=369;
    float heading, scaledx, scaledy;
    uint16_t heading_degrees;

    address[0] = 0x03;
    i2c_write(0x3C, address, 1); // Set pointer position to 0
    i2c_read(0x3D, data, 6);       // Read 6 bytes of magnetic field data
    data_fine[0] = (signed short) (data[0] << 8) | data[1]; // X
    data_fine[1] = (signed short) (data[4] << 8) | data[5];  // Y (Z axis has beed ignored)
    x = data_fine[0];
    y = data_fine[1];
    offsetx=fabs((xmin+xmax)*0.5); // Set offset of coordinate system
    if (fabs(xmin)<xmax) offsetx=-offsetx;
    offsety=fabs((ymin+ymax)*0.5);
    if (fabs(ymin)<ymax) offsety=-offsety;
    scaledx=(x+offsetx)*0.73; //0.73 because of 0.88 gauss gain
    scaledy=(y+offsety)*0.73;
    heading = atan2(scaledy, scaledx);
    heading += 0.08145; // declination angle in Lodz
    if(heading < 0) heading += 6.2831853;  //2 pi
    if(heading > 6.2831853) heading -= 6.2831853;
    heading_degrees = heading * 180/3.1415926; // Change from radians to degrees
    heading_degrees += 180;
    heading_degrees = heading_degrees%360;
    return heading_degrees; // Returns value between 0 - 360
}
```

Rysunek 7. Punkt aktualny (A) oraz punkt docelowy (B) tworzą prostą o pewnym kącie nachylenia (α) względem południka



Rysunek 8. Rozrzut, z którym są odczytywane koordynaty GPS

pracować w dwóch trybach pomiarowych (cyklicznym, pojedynczym) lub spoczynkowym (oszczędzania energii). Do aplikacji wybrałem tryb pracy cykliczny, z pomiarami wykonywanymi z częstotliwością 15 Hz. Tryb wybiera się odpowiednio ustawiając rejestr MR (Mode Register). Rejestr CRA pozwala na ustalenie, ile próbek ma brać udział w uśrednianiu pomiaru, decyduje również o częstotliwości przesyłania danych pomiarowych wyjściowych. Rejestr CRB definiuje wzmocnienie czujnika. Możliwe jest wybranie jednego z ośmiu poziomów wzmocnienia. W trybie ciągłym, po dokonaniu pomiaru, wyprowadzenie RDY czujnika jest ustawiane, co sygnalizuje gotowość do odbioru danych. Wówczas mikrokontroler wysyła zapytanie do czujnika o przesłanie wyników pomiarów, po którym czujnik zwraca 6 bajtów danych. Następnie wyprowadzenie RDY jest zerowane i rozpoczyna się kolejny pomiar.

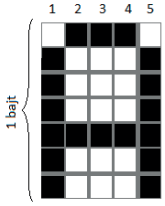


Rysunek 9. Organizacja pamięci PCD8544

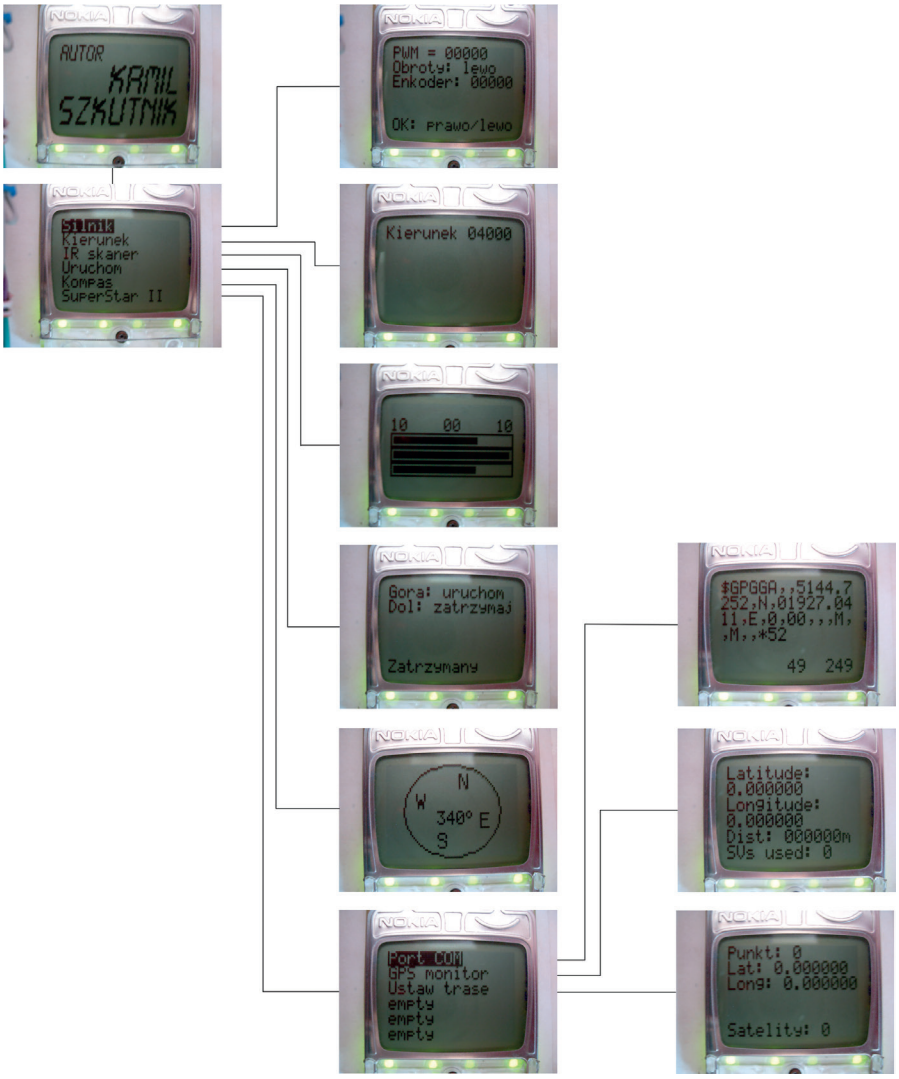
Wyświetlacz LCD

Zastosowałem wyświetlacz monochromatyczny LCD od telefonu Nokia 3310. Ma on rozdzielczość 84×48 pikseli (wymiary czynnego pola ekranu 30 mm×20 mm). Wyświetlacz ma sterownik PCD8544 z interfejsem szeregowym SPI. Dostępny jest wyłącznie tryb graficzny, ale można łatwo programowo emulować tryb tekstowy. Przykładowo, można pokazać sześć linii tekstu, każda po 14 znaków wielkości 6×8 pikseli. Ze względu na delikatną budowę wyświetlacza, postanowiłem pozostawić go w oryginalnej ramce, jedynie odpowiednio przycinając ją celem usunięcia części przeznaczonej dla klawiatury telefonu.

Pierwszym rozwiązaniem, które zastosowałem do połączenia wyświetlacza z mi-



Rysunek 10. Wygląd litery „A"



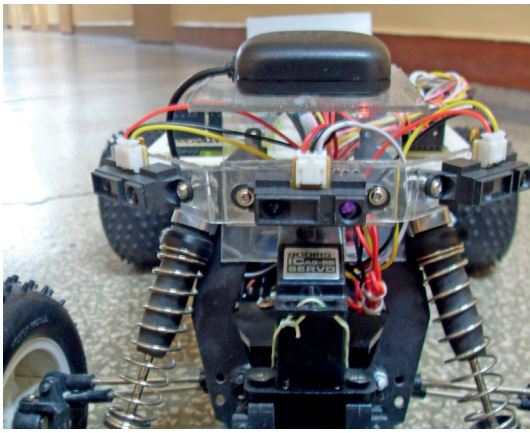
Rysunek 11. Organizacja menu użytkownika

Listing 3. Zapis 8 bitów do pamięci wyświetlacza

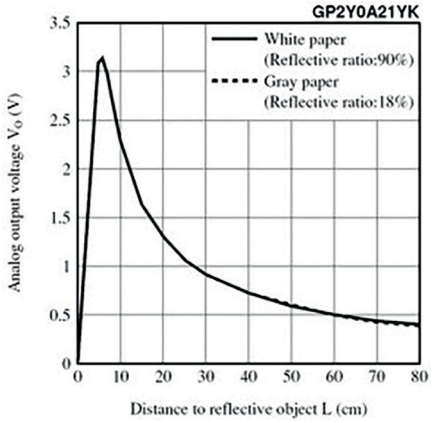
```
//zapisanie danej do sterownika LCD
void WriteDataToLcd(unsigned char data) {
    ClrBit(SCE);
    SetBit(DC); //rejestr danych
    WriteSPI(data);
    SetBit(SCE);
}

//programowa realizacja SPI - LCD
void WriteSPI(unsigned char data) {
    char i;
    ClrBit(CLK);
    for (i = 0; i < 8; i++) {
        if ((data & 0x80) == 0x80) SetBit(DATA);
        else ClrBit(DATA);
        SetBit(CLK);
        ClrBit(CLK);
        data <<= 1;
    }
}
```

Listing 4. Konfigurowanie sterownika wyświetlacza LCD

```
//inicjalizacja sterownika LCD
void LcdInit(void) {
    SetBit(RES); //wylacz reset
    WriteCmd(0x21); //komendy rozszerzone
    WriteCmd(0x05); //komenda ustawiająca tryb pracy zgodny PCD8544
    WriteCmd(0xbe); //ustawienie Vop
    WriteCmd(0x06); //korekcja temperatury dla PCD8544
    WriteCmd(0x14); //współczynnik multipleksowania
    WriteCmd(0x20); //komendy standardowe - adresowanie poziome
    WriteCmd(0x01);
    WriteCmd(0x0c); //tryb wyswietlania Standard Mode
    WriteCmd(0x40); //zerowanie licznika wierszy
    WriteCmd(0x80); //zerowanie licznika kolum
    ClrDisp();
}
```



Fotografia 12. Czujniki odległości

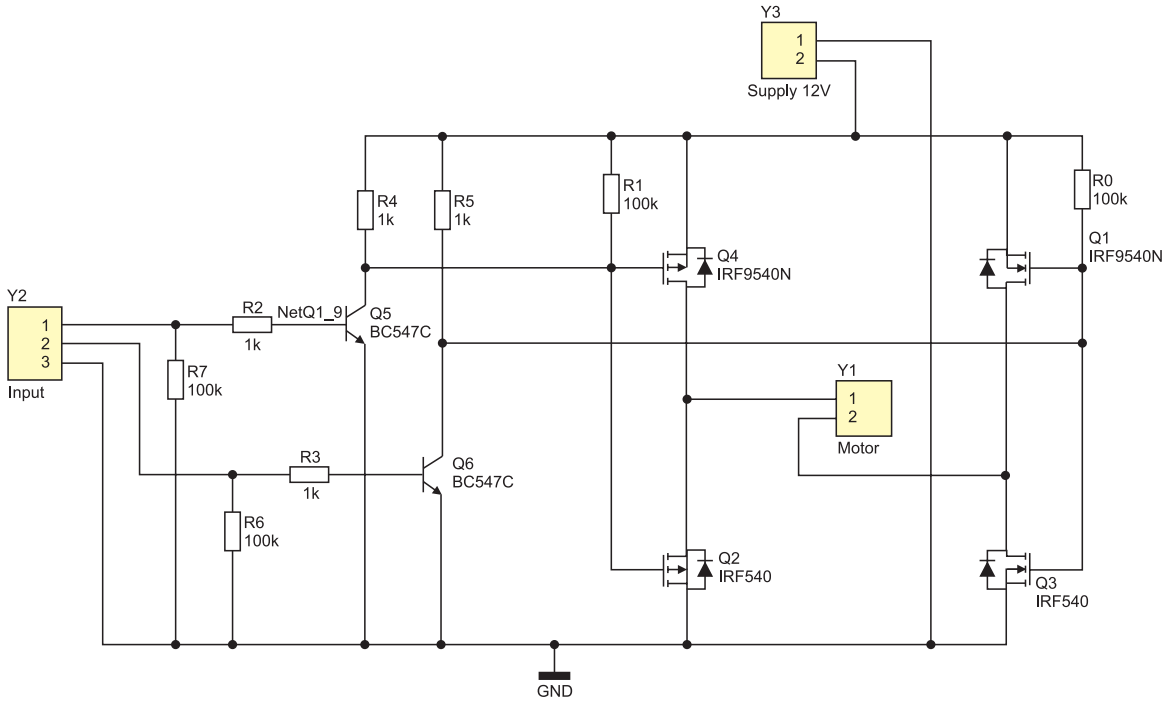


Rysunek 13. Charakterystyka czujników odległości

krokontrolerem, było przylutowanie przewodów do odpowiednich styków wyświetlacza. Rozwiązanie to okazało się nieskuteczne, ponieważ wyświetlacz początkowo nie wykazywał żadnych znaków otrzymania danych. Jednak, gdy palcem przycisnąłem wyprowadzenia do obudowy wyświetlacza, okazało się, że wyświetlacz pracuje prawidłowo. W związku z powyższym, montaż powinien odbyć się na płytce PCB, tak jak zostało to wykonane oryginalnie w telefonie NOKIA 3310.

Oprócz wykonanej płytki PCB dla wyświetlacza, zrealizowałem również podświetlenie wyświetlacza, które działa na podobnej zasadzie jak miało to miejsce w telefonie. Podświetlenie wyświetlacza można wykonać używając płytki dwustronnej. Ponieważ dysponuję metodą pozwalającą na wytrawienie płytki jednostronnej, zastosowałem w projekcie dwie płytki drukowane połączone ze sobą na obu bokach po jednym przewodzie. Oba te przewody tworzą mechaniczne połączenie oraz doprowadzają zasilanie diod LED.

Jak wspomniano, wyświetlacz od Nokii 3310 ma kontroler PCD8544 z interfejsem szeregowym SPI. Jego interfejs ma cztery wejścia: SDIN – szeregowe wejście danych, SCLK – wejście sygnału zegarowego taktującego dane na linii, /SCE – wejście aktywujące interfejs szeregowy, D/C – wejście wyboru



Rysunek 14. Schemat ideowy układu zasilania silnika

rodzaju danych (wyświetlane lub sterujące). Komunikacja przebiega tylko w jednym kierunku – od mikrokontrolera do wyświetlacza. Zależnie od stanu linii D/C, bajty danych wysyłane do wyświetlacza, mogą być interpretowane przez kontroler jako komendy do wykonania, albo dane zapisywane do pamięci RAM obrazu. Poziom wysoki na linii D/C sygnalizuje daną, a niski komendę.

Komunikację rozpoczyna się od ustawienia linii /SCE w stan niski, co aktywuje interfejs SPI. Jeśli wysyłany bajt jest komendą, linię D/C ustawia się w stan niski, a jeśli zwykłą daną – w stan wysoki. Następnie, linią SDIN, szeregowo (bit po bicie) przesyła się 8 bitów danej, zaczynając od bitu najbardziej znaczącego. Transmisja szeregowa jednego bitu przebiega w następujący sposób: najpierw zeruje się lub ustawia, zależnie od wartości przesyłanego bitu, linię danych SDIN. Następnie, na linii SCLK podaje się impuls: 0-1-0. Zmiana na linii /SCE poziomu niskiego na wysoki sygnalizuje zakończenie transmisji [3]. Funkcja zamieszczona na **listingu 3** realizuje wysłanie 1-bajtowej liczby do pamięci wyświetlacza.

Wbudowana w sterownik pamięć RAM jest zapisywana po ustawieniu linii D/C. Matryca wyświetlacza może wyświetlić 48 linii. Każda linia ma 84 piksele. Pamięć jest zorganizowana w 6 banków po 84 bajty każdy (**rysunek 9**). Taka organizacja wymusza logiczny podział pola wyświetlacza na 6 wierszy po 84 kolumny. Wiersze są numerowane 0…5, a kolumny 0…83. Przy wpisywaniu danej do pamięci najpierw określamy numer banku a następnie numer bajtu w wierszu. Zapisanie całego banku odpowiada wyświetleniu jednego wiersza na wyświetlaczu. W trybie tekstowym, odpowiada to wyświetleniu pojedynczej linijki tekstu. Każdy bajt pamięci RAM jest wyświetlany

Listing 5. Funkcja wyświetlająca znak na ekranie LCD

```
void SendChar(unsigned char NrChar) {
    uint8_t k;
    for (k = 0; k < 5; k++) WriteData(Font[NrChar][k]);
        WriteData(0X00); //odstep miedzy znakami
}
```

jako pionowy pasek składający się z 8 pikseli. Najwyżej położony piksel w pasku odpowiada najmniej znaczącemu bitowi, a najniżej – najbardziej znaczącemu. W standardowym trybie wyświetlania, ustawienie bitu w bajcie pamięci odpowiada zaświeceniu, a wyzerowanie zgaszeniu piksela [4]. Sterownik PCD8544 należy odpowiednio skonfigurować, ustawiając współczynnik kompensacji temperaturowej, napięcie wewnętrznej przetwornicy (Vop) oraz inne. Na **listingu 5** zamieszczono listę komend poprawnie inicjalizujących wyświetlacz.

Gdy mamy gotową funkcję wysyłania bajtu do wyświetlacza, możemy tworzyć struktury programowe umożliwiające wyświetlanie pojedynczych pikseli, obrazków, figur geometrycznych oraz tworzyć złożone animacje. Dalej przedstawię rozwiązanie wyświetlenia znaków alfanumerycznych oraz rysowanie linii prostej. Opisywany przeze mnie wyświetlacz nie ma zaimplementowanego sprzętowego trybu tekstowego, dlatego wyświetlanie tekstu zrealizowałem programowo. Polega to na utworzeniu struktury programowej, która jako argument pobiera ciąg znaków, a efektem jej działania jest wysłanie do wyświetlacza grafik, które przedstawiają pobrany napis. Liczby całkowite z przedziału 0…128 mogą być interpretowane przez mikrokontroler jako znaki kodu ASCII (*American Standard Code for Information Interchange*). Dla wybranych znaków umieściłem w tablicy dwuwymiarowej ich przedstawienie w formie graficznej – jest to w moim przypadku 90 znaków. Pozostałe 38 znaków nie ma praktycznego zastosowania w mojej aplikacji, są to znaki specjalne takie jak tabulacja. Dla przykładu przeanalizujmy sposób wyświetlania litery „A" pokazanej na **rysunku 10**. Jej definicja wygląda następująco: { 0x7E, 0x11, 0x11, 0x11, 0x7E }. Pozycja składa się z pięciu liczb 8 bitowych. Reprezentują one pięć kolumn na wyświetlaczu LCD. Funkcję wyświetlająca znak na ekranie LCD pokazano na **listingu 6**.

Interfejs użytkownika robota na LCD

Aby obserwować stan robota i aktualne wskazania czujników stworzyłem interfejs użytkownika z wielopoziomowym menu stwarzającym nieograniczone możliwości prezentacji danych i komunikacji między użytkownikiem a pojazdem (**rysunek 11**). Oprócz wyświetlania informacji, interfejs służy również do wymuszenia zmiany kluczowych zmiennych wewnętrznych (prędkość, kierunek skrętu kół), zadawania trasy przejazdu i uruchamiania pojazdu.

Czujnik odległości

W celu skutecznego omijania przeszkód, pojazd posiada czujniki odległości, dzięki którym jest w stanie wykryć przeszkodę oraz ją prawidłowo zlokalizować. Rozpoznawanie przez pojazd otoczenia zrealizowałem poprzez zastosowanie trzech czujników odległości Sharp GP2Y0A21YK0F znajdujących się z przodu pojazdu (**fotografia 12**). Czujnik pozwala na wykrywanie obiektów w odległości od 10 do 80 cm. Wyjściem jest sygnał analogowy, któ-

Listing 6. Procedura omijająca przeszkody

```
/* Main move algorithm */
void MoveAlgorithm(void) {
    //if obstacle in sight, but far away
    if ((IR_3way(0) >= 40) & (IR_3way(1) >= 40) & (IR_3way(2) >= 40)) {
        //when the obstacle is nearer the right side
        if (IR_3way(0) < IR_3way(2)) {
            Direction_write(2800); //turn left
            Motor_set(14000, 0); //medium speed, forward
        }
        //when the obstacle is nearer the left side
        if (IR_3way(2) < IR_3way(0)) {
            Direction_write(5200); //turn right
            Motor_set(14000, 0); //medium speed, forward
        }
        //when the obstacle is in the front of the vehicle
        if (IR_3way(0) == IR_3way(2)) {
            Direction_write(4000); //center
            Motor_set(14000, 0); //medium speed, forward
        }
    } else { //if the obstacle is in sight and close
        //when the obstacle is in the front of the vehicle and close
        if ((IR_3way(1) < 40) {
            Direction_write(5100); //turn right
            Motor_set(14000, 1); //medium speed, backward
        }
        //when the obstacle is nearer the left side
        if (IR_3way(0) < IR_3way(2)) {
            Direction_write(5200); //turn right
            Motor_set(14000, 1); //medium speed, backward
        }
        //whaen the obstacle is nearer the right side
        if (IR_3way(2) < IR_3way(0)) {
            Direction_write(2800); //turn left
            Motor_set(14000, 1); //medium speed, backward
        }
    }
}
```

rego wartość zależna jest od odległości pomiędzy wykrytym obiektem a sensorem. Im obiekt znajduje się bliżej, tym napięcie na wyjściu jest wyższe. Pojazd odpowiednio steruje układem kierowniczym oraz napędowym w taki sposób, aby w linii prostej dotrzeć do następnego punktu pośredniego lub docelowego. Gdy czujniki odległości rozpoznają przeszkodę w pobliżu, algorytm zbliżania się do punktu pośredniego lub docelowego jest przerywany i sterowanie przejmuję funkcja odpowiedzialna na ominięcie przeszkody. Algorytm omijania przeszkód ma za zadanie tak wysterować układ kierowniczy i napędowy pojazdu, aby ten bezpiecznie oddalił się od wykrytej przeszkody. Na **listingu 6** zamieszczono kod funkcji odpowiedzialnej za zmianę zachowania pojazdu w zależności od miejsca przeszkody względem robota.

Ponieważ charakterystyka wyjściowa czujnika jest nieliniowa (**rysunek 13**), w celu wyznaczenia odległości od czujnika zastosowałem linearyzację jego charakterystyki wyjściowej. Zrealizowałem ją programowo. Pierwszym krokiem było wykonanie serii pomiarów, od 10 cm do 80 cm co 10 cm. Otrzymane napięcia dla każdej z odległości umieściłem w programie w postaci tablicy danych. Napięcie odczytane z przetwornika A/C jest porównywane z danymi zawartymi w tablicy. Gdy wartość odczytanego napięcia zawiera się w przedziale dwóch sąsiednich komórek tablicy, pobierana jest odpowiadająca tym komórkom odległość i zwracana przez funkcję zawierającą opisany algorytm. W wyniku działania funkcji otrzymujemy wartości odległości z dokładnością 10 cm.

Sterowanie silnikiem głównym i napędem skrętu

Sterowanie silnika głównego robota mobilnego zostało przeze mnie zrealizowane poprzez zastosowanie układu typu mostek H. Zastosowałem tranzystory MOSFET, dwa z kanałem typu P, dwa kolejne z kanałem typu N, zgodnie ze schematem na **rysunku 14**.

Tranzystory mocy są sterowane pośrednio poprzez sygnał PWM mikrokontrolera. Elementami pośredniczącymi są dwa tranzystory bipolarne. Zastosowałem je aby zwiększyć napięcie sterujące tranzystorami mocy. Dzięki zastosowaniu dodatkowych tranzystorów napięcie na bramce w momencie odblokowania tranzystora wynosi 7,4 V zamiast 3,3 V, którym dysponuje mikrokontroler. Mostek został zaprojektowany dla prądu ciągłego do 23 A przy napięciu zasilania 7,4 V.

Sterowanie serwonapędem układu kierowniczego zrealizowałem poprzez zastosowanie sygnału okresowego o zmiennej szerokości impulsów (PWM). Aby kontrolować wychylenia serwomechanizmu mikrokontroler generuje przebieg prostokątny o częstotliwości 50 Hz, którego stan wysoki w każdym okresie trwa od 1 ms do 2 ms. Dzięki zmianie czasu trwania stanu wysokiego w każdym okresie, możliwe jest odpowiednie ustawianie wychylenia serwomechanizmu w zakresie 90 stopni.

Test

Po wykonaniu wielu prób w terenie, wprowadzeniu kolejnych poprawek programu, jak i samej konstrukcji mechanicznej pojazdu, mogę określić zbudowany przeze mnie pojazd jako w pełni funkcjonalny. Wykonałem test pojazdu w środowisku do jakiego został zaprojektowany, odbył się on na parkingu Pasażu Łódzkiego przy ulicy Aleja Jana Pawła II w Łodzi.

Test polegał na przejechaniu trasy w kształcie prostokąta (o wymiarach: 14×23 metry) określonego przez cztery punkty (**rysunek 15**). Punkty te zostały wprowadzone ustawiając pojazd w kolejnych lokalizacjach B, C, D, A i zapisując aktualne położenie GPS korzystając z wbudowanego interfejsu użytkownika. Po takim zaprogramowaniu trasy pojazd ustawiłem w punkcie A, z którego wystartował w stronę punktu B, następnie C, D, kończąc wrócił do punktu A i tam się zgodnie z planem zatrzymał.

Niebieska linia łączy zaprogramowane w pojeździe punkty za pomocą prostych odcinków, natomiast kolorem żółtym została zaznaczona trasa pokonana przez robota. Ścieżkę zrealizowaną przez pojazd narysowałem na podstawie nagrania wideo.

W celu dokonania analizy dokładności z jaką pojazd pokonał trasę, narysowałem na powyższym zdjęciu siatkę o boku 1 m. Największa odchyłka ścieżki pojazdu od trasy zadanej znajduje się pomiędzy punktami A i B, wynosi 4 m.

Kamil Szkutnik
www.kamilszkutnik.pl

Rysunek 15. Trasa testowa