Laboratorium 2: Interfejsy komunikacyjne

Angelika Tefelska, Dariusz Tefelski

5 marca 2023

1 GPS

(a) Podłącz moduł GPS (WaveShare NEO-6M/7M) do Arduino Mega: VCC → 5V, GND → GND, TX → RX1, RX → TX1

(b) Poniżej umieszczono program, który w monitorze portu szeregowego wyświetli dane odebrane przez moduł GPS:

```
void setup()
{
   Serial.begin(9600);
   Serial.println("WaveShare Neo-6M/7M module test");
   Serial1.begin(9600);
}

void loop()
{
   if(Serial1.available()) Serial.write(Serial1.read());
}
```

Przepisz go i uruchom na płytce Arduino. Pamiętaj, że moduł GPS jest podłączony do Serial1 a wyświetlanie informacji będzie realizowane za pomocą obiektu Serial. Format zwracanych ramek jest dobrze opisany w poniższych tabelkach. Zapoznaj się z nim i postaraj się zrozumieć wyświetlane informacje. **UWAGA:** moduł GPS złapał sygnał z satelity w momencie gdy dioda czerwona miga.

$**GPGGA** - Global Positioning System Fix Data (*czas i pozycja*)

$GPGGA,211455.000,5022.3220,N,01850.9486,E,1,4,20.73,337.7,M,42.1,M,,*66		
Parametr	Wartość	Opis
Identyfikator	$GPGGA	
Czas UTC	211455.000	21:14:55.000 / hhmmss.sss
Szerokość geograficzna	5022.3220,N	50d 22.3220' N lub 50d 22' 19" N ddmm.mmmm, N - północ, S - południe
Długość geograficzna	01850.9486,E	18d 50.9486' E lub 18d 50' 57" E ddmm.mmmm, E - wschód, W - zachód
Jakość	1	0:błędna, 1:GPS fix, 2:DGPS fix
Liczba użytych seatelitów	4	Liczba widocznych satelitów
Względna dokładność pozycji poziomej	20.73	HDOP
Wysokość	337.7,M	336,7 metrów (3D fix)
Wysokość geoidy powyżej elipsoidy WGS84	42.1,M	42.1 metra
Czas od ostatniej aktualizacji DGPS	<brak>	
Stacja referencyjna DGPS	<brak>	
Suma kontrolna	*66	

$**GPGSA** - GPS DOP and active satellites (*tryb pracy odbiornika, aktywne satelity*)

$GPGSA,A,3,15,21,18,22,,,,,,,,,20.76,20.73,0.97*07		
Parametr	Wartość	Opis
Identyfikator	$GPGSA	
Tryb pracy	A	M - ręczny, A - automatyczny
Tryb fixa	3	1 - brak, 2 - 2D fix, 3 - 3D fix
Satelity biorące udział w obliczeniu pozycji	15,21,18,22,,,,,,,,,	Pola od 3 do 14
PDOP	20.76	Rozmycie pozycji
HDOP	20.73	Rozmycie poziome
VDOP	0.97	Rozmycie pionowe
Suma kontrolna	*07	

$**GPGSV** - GPS Satellites in view *(ilość widocznych satelit, ich numery identyfikacyjne, azymut)*

$GPGSV,3,1,12,27,61,297,,18,60,118,40,16,54,231,14,21,45,071,37*76
$GPGSV,3,2,12,22,44,174,32,26,34,197,15,39,31,171,26,19,30,296,*7D
$GPGSV,3,3,12,15,16,059,36,07,10,315,14,13,08,027,23,30,04,343,14*7C

Parametr	Wartość	Opis
Identyfikator	$GPGSV	
Ilość wiadomości	3	Ile jest dsotępnych wiadomości
Numer wiadomości	1	Powinny być zwrócone kolejno 1, 2 i 3
Całkowita liczba widocznych sateltów	12	
Identyfikator satelity	27	
Elewacja	61	Wysokość w stopniach
Azymut	297	Stopnie od prawdziwej północy
Siła sygnału	<brak>	00-99 dB, wartość pusta jeśli nie bierze udziału
Informacje o drugiej satelicie	18,60,118,40	Informacje takie same jak w polach 4 - 7
Informacje o trzeciej satelicie	16,54,231,14	Informacje takie same jak w polach 4 - 7
Informacje o czwartej satelicie	21,45,071,37	Informacje takie same jak w polach 4 - 7
Suma kontrolna	*76	

(c) Zmodyfikuj program z poprzedniego punktu tak aby w monitorze portu szeregowego były wyświetlone informacje zgodnie ze wzorem:

```
ime     Width   Length          Quality         Number of Satellites    Precision       Height
.3:57:10        5213.30084      02100.41699     1       10      0.86    135.0
.3:57:11        5213.30092      02100.41713     1       11      0.86    135.1
.3:57:12        5213.30093      02100.41722     1       11      0.86    134.9
.3:57:13        5213.30100      02100.41727     1       09      1.02    134.7
.3:57:14        5213.30113      02100.41737     1       09      1.02    134.5
.3:57:15        5213.30119      02100.41747     1       09      1.02    134.6
.3:57:16        5213.30125      02100.41760     1       09      1.02    134.6
.3:57:17        5213.30132      02100.41768     1       09      1.02    134.7
.3:57:18        5213.30135      02100.41774     1       10      1.02    134.7
.3:57:19        5213.30126      02100.41780     1       10      1.02    134.9
.3:57:20        5213.30113      02100.41793     1       11      0.83    135.1
.3:57:21        5213.30100      02100.41809     1       11      0.83    134.9
.3:57:22        5213.30093      02100.41831     1       11      0.77    135.0
```

W tym celu skorzystaj z dokumentacji do obiektu Serial dostępną na stronie arduino.cc oraz wybierz odpowiednią funkcję do odczytu danych.

Wskazówki:

- Funkcja *readStringUntil('znak')* wywołana na obiekcie *Serial* umożliwia odczytanie ciągu znaków do wybranego znaku np. przecinka, średnika itd. Funkcja zwraca *String*,

specjalny typ zdefiniowany dla Arduino. Więcej na ten temat: na stronie poświęconej *String*. Przykład użycia:

```
String element=Serial1.readStringUntil(',');
```

- Funkcja *c_str()* wywołana na zmiennej typu *String* zwraca łańcuch znaków w standardzie języka C.
- Funkcja *strstr(string1,string2)* sprawdza czy w *string1* znajduje się *string2*. Jeśli tak to zwraca wartość inną niż *NULL* a dokładniej adres pierwszego wystąpienia elementu *string2*. Przykład użycia:

```
if(strstr(element.c_str(),"GPGGA")!=NULL)
{
        //Wykryto "GPGGA" w element
}
```

- Funkcja atol(string) umożliwia zamianę string-a na int. Przykład użycia:

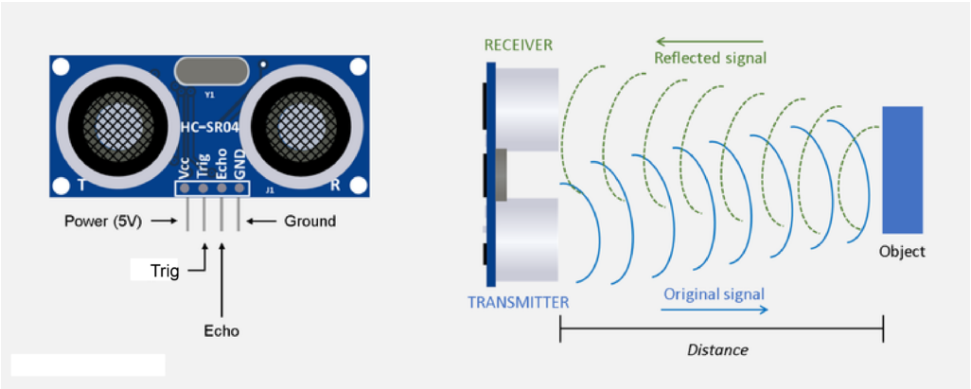
```
String timeval=Serial1.readStringUntil(',');
long int t=atol(timeval.c_str());
```

(d) Wejdź na stronę www.google.pl/maps i wpisz odczytane współrzędne zgodnie z konwencją: długość geograficzna, szerokość geograficzna np. 52 13.30113, 21 00.41737. Czy odczyt z GPS zgadza się z aktualną lokalizacją z dokładnością do 6m?

2 Czujnik parkowania

Zbuduj prototyp czujnika parkowania:

(a) Do pomiaru odległości wykorzystaj ultradźwiękowy czujnik odległości (HC-SR04). Podłącz go do arduino i spróbuj odczytać odległości na monitorze portu szeregowego. W tym celu stwórz funkcje *distance()*, która będzie odpowiednio konfigurowała czujnik w celu dokonania pomiaru i zwracała odległość w cm. Zasada działania czujnika HC-SR04 przedstawiono na rysunku 1.



Rysunek 1: Zasada działania czujnika odległości HC-SR04. Źródło obrazka: link.

Aby odczytać dane z ultradźwiękowego czujnika odległości należy:

- Skonfigurować pin podłączony do triggera jako wyjściowy.
- Skonfigurować pin podłączony do echo jako wejściowy.
- Ustawić sygnał HIGH na triggerze.
- Poczekać 10 μs.
- Ustawić sygnał LOW na triggerze.
- Odczytać czas pomiędzy dwoma wysokimi impulsami na wejściu echo.
- Podzielić otrzymany wynik przez 58 aby uzyskać odległość w cm. Dlaczego 58? Otóż fala dźwiękowa pokonuje podwójną odległość między czujnikiem a obiektem zatem:

$$2 \cdot s = v \cdot t \tag{1}$$

Prędkość dźwięku w powietrzu wynosi ok: 340 m/s czyli 0.034 $cm/\mu s$ zatem:

$$s = \frac{v}{2} \cdot t = \frac{0.034 cm/\mu s}{2} \cdot t[\mu s] = 0.017 \cdot t \approx \frac{t}{58} \tag{2}$$

Wskazówki:

- funkcja do wykonania opóźnienia w mikrosekundach to: **delayMicroseconds(czas)**,
- funkcja do mierzenia czasu między dwoma impulsami czy to wysokimi czy niskimi to: **pulseIn**(pin na którym ma być mierzony czas między impulsami, rodzaj impulsu między którym ma być czas mierzony czyli HIGH/LOW).

(b) Udoskonal swój układ podłączając buzzer BPT-14LF. Gdy przeszkoda znajdzie się w odległości 1m zasygnalizuj to sygnałem dźwiękowym z buzzera. Wraz z malejącą odległością od przeszkody zacznij coraz częściej generować dźwięk z buzzera.
Wskazówka: Jedno z możliwych rozwiązań to:

- Po odczytaniu odległości z ultradźwiękowego czujnika odległości można, najpierw włączyć buzzer wysyłając sygnał HIGH na dany pin.
- Potem wywołać funkcję do opóźnień z wybranym przez siebie czasem w ms.
- Potem wyłączyć buzzer wysyłając sygnał LOW na dany pin.
- Potem wywołać funkcję do opóźnień podając jako argument odczytany dystans np. delay(distance);

(c) Zmodyfikuj układ dołączając wyświetlacz LCD (rysunek 2). Wyświetl na nim informacje o odległości od przeszkody. W tym celu skorzystaj z biblioteki LiquidCrystal. Przykładowy program:

```
#include <LiquidCrystal.h>
#include <Wire.h>

#define RS 22
```

```
#define E 24
#define D4 26
#define D5 28
#define D6 30
#define D7 32

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);

void setup()
{
   lcd.begin(16,2);
   lcd.clear();
}

void loop()
{
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Hello!");
}
```

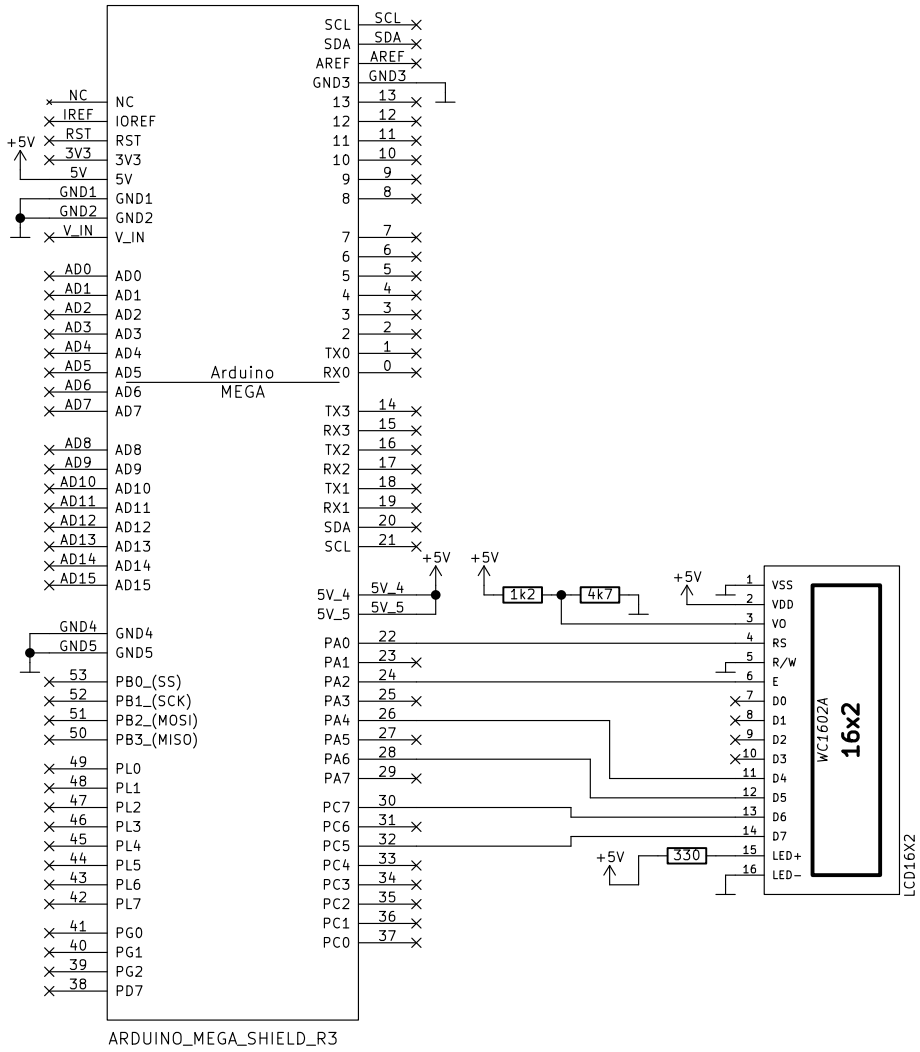
(d) Spróbuj podłączyć wyświetlacz LCD w inny sposób. W tym celu skorzystaj z konwertera I2C dla wyświetlacza LCD, który podłącz do arduino w następujący sposób: GND → GND, Vcc → 5V, SDA → SDA(pin 20), SCL → SCL (pin21). Patrz schemat na rysunku 3.

Do obsługi wyświetlacza LCD poprzez I2C skorzystaj z biblioteki **LiquidCrystal_I2C**, którą można pobrać bezpośrednio z poziomu Arduino IDE. W tym celu wybierz: *Narzędzia* → *Zarządzaj bibliotekiami...*. Następnie wpisz nazwę szukanej biblioteki, odnajdź właściwą z listy i zainstaluj.

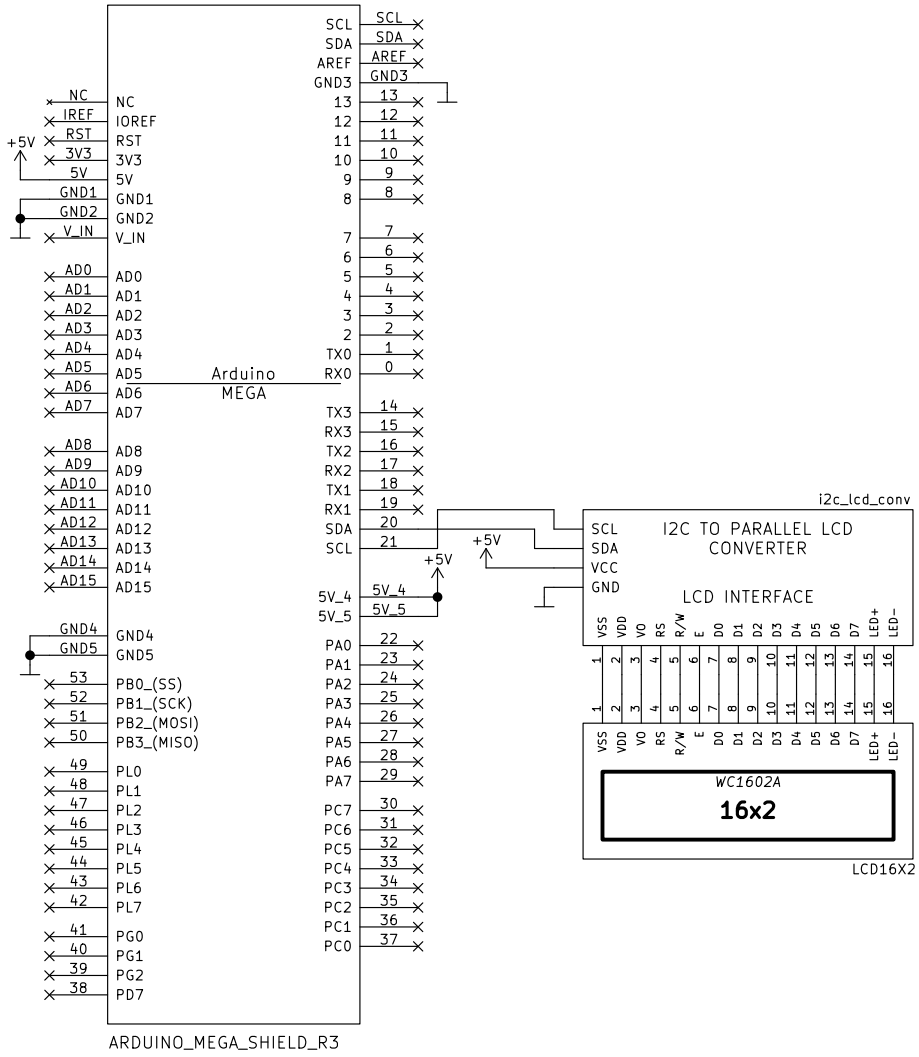
Przykładowy program:

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd2(0x3f, 16,4); //0x3f to adres
   wyswietlacza

void setup()
{
   lcd2.init();
   lcd2.backlight();
   lcd2.setCursor(0,0);;
}
```

```
void loop()
{
   lcd2.clear();
   lcd2.setCursor(0,0);
   lcd2.print("Hello!");
}
```



Rysunek 2: Podłączenie wyświetlacza LCD do Arduino. UWAGA: na schemacie jest mały błąd. Należy zamienić miejscami rezystory 1.2kΩ oraz 4.7kΩ. Alternatywna wersja podłączenia wyświetlacza LCD z wykorzystaniem potencjometru jest dostępna na slajdach z wykładu nr 2.



Rysunek 3: Podłączenie wyświetlacza LCD do Arduino poprzez konwerter I2C.