

Podstawy Systemów Wbudowanych

Laboratorium

Laboratorium 2: Interfejsy komunikacyjne — UART i I²C

Autorzy: Angelika Tefelska, Dariusz Tefelski

Spis treści

1	Wprowadzenie	1
2	Wyposażenie stanowiska	1
3	Zadanie 1: Komunikacja szeregową UART	2
3.1	Parametry transmisji UART	2
3.2	Serial Monitor — komunikacja z komputerem	3
3.3	Sprzętowy UART i moduł GPS	4
3.4	Co wysyła GPS? — komunikaty NMEA	5
4	Zadanie 2: Wyświetlacz LCD i magistrala I²C	7
4.1	Klasyczne podłączenie wyświetlacza LCD 16x2	7
4.2	Dlaczego warto zastosować I ² C?	9
4.3	Dlaczego potrzebne są rezystory podciągające?	10
4.4	Adresowanie i potwierdzenie ACK	10
4.5	Wyświetlacz LCD jako urządzenie I2C	11
5	Zadanie końcowe: Połączenie dwóch sposobów komunikacji	13
6	Podsumowanie	14

1 Wprowadzenie

Cel ćwiczenia



Celem ćwiczenia jest poznanie dwóch często wykorzystywanych sposobów komunikacji mikrokontrolera z innymi urządzeniami: UART oraz I²C. W pierwszej części wykorzystamy sprzętowy port UART do odbioru danych z modułu GPS, a w drugiej poznamy magistralę I²C na przykładzie wyświetlacza LCD.

W systemie wbudowanym mikrokontroler rzadko pracuje całkowicie samodzielnie. Czujniki, wyświetlacze i inne układy peryferyjne muszą wymieniać z mikrokontrolerem informacje. Nie każde urządzenie komunikuje się jednak w taki sam sposób. W tym laboratorium poznamy dwa różne rozwiązania: prostą komunikację punkt–punkt UART oraz magistralę I²C, do której można podłączyć wiele urządzeń.

UART i I²C — dwie różne idee



	UART	I ² C
Typ transmisji	asynchroniczna	synchroniczna
Linie	TX, RX	SDA, SCL
Zegar	brak osobnej linii	linia SCL
Połączenie	zwykle punkt–punkt	wiele urządzeń na wspólnej magistrali
Identyfikacja urządzenia	nie jest potrzebny adres	urządzenia mają adresy

W UART nadajnik i odbiornik nie przesyłają wspólnego sygnału zegarowego, dlatego muszą być skonfigurowane z takimi samymi parametrami transmisji. W I²C urządzenie sterujące generuje sygnał zegarowy na linii SCL, który synchronizuje prace czujników z mikrokontrolerem.

2 Wyposażenie stanowiska

Do wykonania ćwiczenia potrzebne są:

- Arduino UNO R4 WiFi,
- przewód USB,
- moduł GPS WaveShare NEO-6M/7M,
- wyświetlacz LCD 16x2,
- konwerter I²C do wyświetlacza LCD,
- potencjometr 10 kΩ do regulacji kontrastu LCD,

- przycisk,
- płytki stykowa,
- przewody połączeniowe.

Co należy zapamiętać?



Wszystkie połączenia wykonuj przy odłączonym zasilaniu Arduino. Urządzenia komunikujące się przewodowo muszą mieć wspólny potencjał odniesienia, dlatego ich masy należy połączyć. Przed podłączeniem modułu sprawdź oznaczenia jego wyprowadzeń.

3 Zadanie 1: Komunikacja szeregową UART

UART (*Universal Asynchronous Receiver/Transmitter*) jest interfejsem szeregowym, w którym dane przesyłane są kolejnymi bitami bez osobnej linii zegarowej. Podstawowe linie to:

- TX (*transmit*) — linia, którą urządzenie wysyła dane,
- RX (*receive*) — linia, którą urządzenie odbiera dane.

Jeżeli dwa urządzenia mają komunikować się dwukierunkowo, linie należy skrzyżować: TX pierwszego urządzenia łączy się z RX drugiego, a RX pierwszego z TX drugiego. Konieczne jest także połączenie mas obu urządzeń.

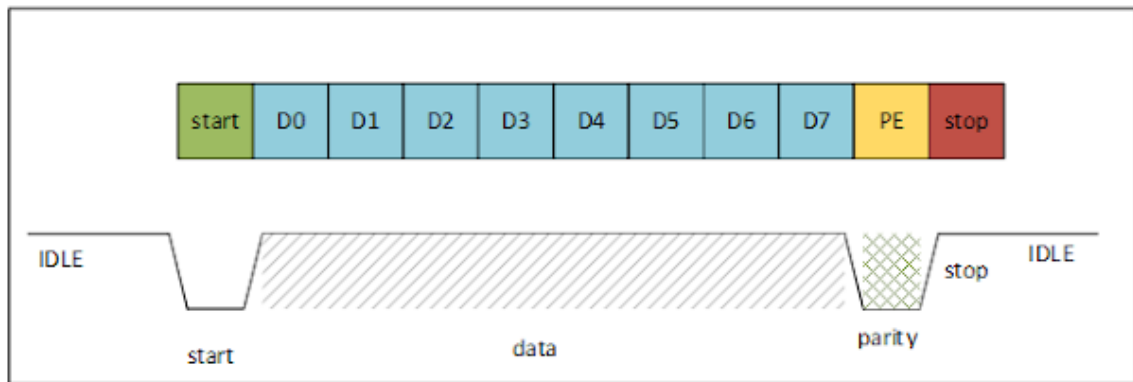
3.1 Parametry transmisji UART

Ponieważ UART nie ma wspólnej linii zegara, oba urządzenia muszą wcześniej uzgodnić sposób przesyłania danych. Jednym z najważniejszych parametrów jest szybkość transmisji, określana zwykle jako *baud rate*. Popularne wartości to np. 9600 lub 115200 bodów.

Typowa konfiguracja zapisywana jako 8N1 oznacza:

- 8 bitów danych,
- brak bitu parzystości (*None*),
- 1 bit stopu.

Każdy przesyłany znak jest poprzedzony bitem startu, a zakończony bitem lub bitami stopu. Odbiornik wykorzystuje ustaloną szybkość transmisji, aby wiedzieć, w jakich chwilach odczytać kolejne bity.



Rysunek 1: Format ramki UART. Źródło obrazka: <https://rtdigitaldesign.blogspot.com/2016/06/universal-asynchronous.html>.

Co się stanie przy złej szybkości transmisji?



Jeżeli nadajnik i odbiornik pracują z różnymi szybkościami, odbiornik próbuje odczytywać bity w niewłaściwych chwilach. W monitorze portu szeregowego może to wyglądać jak przypadkowe, niezrozumiałe znaki.

3.2 Serial Monitor — komunikacja z komputerem

Obiekt `Serial` pozwala wymieniać dane pomiędzy programem na Arduino a komputerem przez połączenie USB. W tym przypadku nie musimy samodzielnie podłączać linii TX i RX — połączenie jest realizowane przez płytke.

Zadanie 1.1 — Odczyt danych z monitora portu szeregowego



Uruchom poniższy program:

```

1 void setup() {
2     Serial.begin(115200);
3     Serial.println("PSW - Laboratorium 2");
4     Serial.println("Wpisz dowolny znak:");
5 }
6
7 void loop() {
8     if (Serial.available() > 0) {
9         char znak = Serial.read();
10
11         Serial.print("Odebrano znak: ");
12         Serial.println(znak);
13     }
14 }
```



Otwórz *Serial Monitor* z menu edytora IDE, ustaw szybkość transmisji na 115200 i wyślij kilka pojedynczych znaków. Następnie wyślij całe słowo. Zwróć uwagę, że program odczytuje dane po jednym znaku. Sprawdź również, czy po wysłaniu tekstu pojawia się dodatkowy znak wynikający z ustawionego w *Serial Monitor* zakończenia linii.

Jak odczytywany jest strumień danych?



- `Serial.available()` zwraca liczbę bajtów oczekujących w buforze odbiorczym,
- `Serial.read()` pobiera jeden bajt z bufora,
- `Serial.print()` i `Serial.println()` wysyłają dane w kierunku komputera.

Jeżeli wpiszesz kilka znaków, trafiają one do Arduino jako kolejne bajty. Program może je przetwarzać pojedynczo albo gromadzić, aby utworzyć całą wiadomość.

3.3 Sprzętowy UART i moduł GPS

Arduino UNO R4 WiFi udostępnia sprzętowy port UART na pinach:

Funkcja UART	Pin Arduino
RX	D0
TX	D1

W programie port ten obsługiwany jest przez obiekt `Serial1`. Jest on niezależny od obiektu `Serial`, którego używamy do komunikacji z komputerem.

W tym ćwiczeniu moduł GPS będzie przede wszystkim **źródłem danych**. Arduino ma odbierać komunikaty GPS i przekazywać je do komputera. Do tego wystarczy połączenie linii TX modułu GPS z linią RX Arduino.

Zadanie 1.2 — Odbiór surowych danych z GPS



Odłącz Arduino od USB i podłącz moduł GPS:

GPS NEO-6M/7M	Arduino UNO R4 WiFi
VCC	5V
GND	GND
TX	D0/RX
RX	nie podłączaj

Po sprawdzeniu połączeń ponownie podłącz Arduino i uruchom program:

```

1 void setup() {
2     Serial.begin(115200); // komunikacja z komputerem
3     Serial1.begin(9600);  // komunikacja z GPS
4     Serial.println("Oczekiwanie na dane GPS...");

```



```
6 }  
7  
7 void loop() {  
8     if (Serial1.available() > 0) {  
9         char znak = Serial1.read();  
10        Serial.write(znak);  
11    }  
12 }
```

Otwórz *Serial Monitor*. Powinny pojawić się kolejne linie tekstu wysyłane przez moduł GPS.

Zaobserwuj:

- jak często pojawiają się komunikaty,
- od jakich znaków rozpoczynają się poszczególne linie,
- czy wszystkie linie mają taki sam format,
- co zmienia się po uzyskaniu przez moduł informacji o położeniu.

3.4 Co wysyła GPS? — komunikaty NMEA

Popularne moduły GPS przesyłają informacje w postaci tekstowych komunikatów zgodnych ze standardem NMEA. Poszczególne pola są oddzielone przecinkami. W zależności od konfiguracji odbiornika możesz spotkać m.in. komunikaty rozpoczynające się od \$GPGGA, \$GNGGA, \$GPRMC lub \$GNRMC.

W komunikatach można znaleźć m.in.:

- czas UTC,
- szerokość i długość geograficzną,
- informację, czy odbiornik posiada poprawne wyznaczenie pozycji (*fix*),
- liczbę wykorzystywanych satelitów,
- wysokość lub prędkość — zależnie od rodzaju komunikatu.

Najważniejsza ramka NMEA została przedstawiona na rysunku 2.

\$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 satelitów	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	

Rysunek 2: Ramka GPGGA.

Zadanie 1.3 — Odczyt pozycji z GPS



Stwórz program, który rozpoznaje ramkę GPGGA, odczytuje jej kolejne pola i wyświetla w monitorze portu szeregowego co najmniej: czas UTC, szerokość geograficzną, długość geograficzną, jakość pozycji, liczbę satelitów oraz wysokość. Wynik przedstaw w czytelnej postaci podobnej do tabeli 1.

W tym celu skorzystaj z dokumentacji obiektu `Serial` dostępnej na stronie <https://www.arduino.cc/reference/en/language/functions/communication/serial/> oraz dobierz odpowiednie funkcje do odczytu strumienia danych.

Wskazówki:

- Funkcja `readStringUntil('znak')` wywołana na obiekcie `Serial1` umożliwia odczytanie ciągu znaków do wskazanego separatora, np. przecinka. Funkcja zwraca obiekt typu `String`. Przykład:

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



- Funkcja `c_str()` wywołana dla zmiennej typu `String` zwraca łańcuch znaków w formacie używanym przez standardowe funkcje języka C.
- Funkcja `strstr(string1, string2)` pozwala sprawdzić, czy w pierwszym łańcuchu znajduje się drugi. Można jej użyć do rozpoznania początku ramki GPGGA:

```
1 if (strstr(element.c_str(), "GPGGA") != NULL) {  
2     // wykryto początek ramki GPGGA  
3 }
```

- Funkcje takie jak `atol()` lub `atof()` pozwalają zamienić tekst na wartość liczbową. Przykład:

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

Uwaga na format współrzędnych. W ramach NMEA współrzędne nie są zapisane bezpośrednio w stopniach dziesiętnych. Szerokość ma postać `ddmm.mmmm`, a długość `dddmm.mmmm`. Przykładowo:

$$52^{\circ}13.300084' = 52 + \frac{13.300084}{60} \approx 52.221668^{\circ}.$$

Analogicznie przelicz długość geograficzną. Uwzględnij również oznaczenia półkul: N/S oraz E/W; dla S i W współrzędna w zapisie dziesiętnym powinna mieć znak ujemny.

Po przeliczeniu współrzędnych wpisz je do mapy w kolejności **szerokość, długość**, np. 52.221668, 21.006950. Porównaj wskazane miejsce z położeniem laboratorium. Nie oczekuj dokładności kilku metrów — szczególnie wewnątrz budynku liczba widocznych satelitów i jakość wyznaczenia pozycji mogą być ograniczone.

Czas	Szerokość	Długość	Jakość	Satelity	HDOP	Wysokość
12:57:10	5213.300084	02100.41699	1	10	0.86	135.0

Tablica 1: Przykład danych odczytanych z ramki GPGGA przed przeliczeniem współrzędnych do stopni dziesiętnych.

4 Zadanie 2: Wyświetlacz LCD i magistrala I²C

4.1 Klasyczne podłączenie wyświetlacza LCD 16x2

Typowy wyświetlacz alfanumeryczny LCD 16x2 z kontrolerem zgodnym z HD44780 może być sterowany bezpośrednio z mikrokontrolera. W trybie 4-bitowym do przesyłania danych wykorzystujemy cztery linie danych D4--D7 oraz dwie linie sterujące: RS i E. Oznacza to zajęcie łącznie sześciu linii GPIO.

Pin RW wyświetlacza podłączamy w tym ćwiczeniu do masy, ponieważ będziemy jedynie zapisywać dane do LCD. Pin VO służy do regulacji kontrastu i należy połączyć go ze środkowym wyprowadzeniem potencjometru 10 kΩ, którego skrajne wyprowadzenia są podłączone do 5 V i GND.

LCD 16x2	Arduino UNO R4 WiFi
VSS	GND
VDD	5V
VO	środkowe wyprowadzenie potencjometru 10 kΩ
RS	D7
RW	GND
E	D8
D0-D3	zostaw niepodłączone
D4	D9
D5	D10
D6	D11
D7	D12
LED+	podłącz przez rezystor 330Ω do 5V
LED-	GND

Kontrast a podświetlenie



Jeżeli na LCD nic nie widać, nie oznacza to od razu błędu programu. Najpierw wyreguluj potencjometrem napięcie na pinie VO. Podświetlenie LCD jest osobnym obwodem.

Zadanie 2.1 — LCD podłączony równolegle



Zmontuj układ zgodnie z tabelą, a następnie uruchom program:

```

1 #include <LiquidCrystal.h>
2
3 LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
4
5 void setup() {
6     lcd.begin(16, 2);
7
8     lcd.setCursor(0, 0);
9     lcd.print("PSW - Lab 2");
10
11    lcd.setCursor(0, 1);
12    lcd.print("LCD rownolegle");
13 }
14

```



```
void loop() {  
}
```

Wyreguluj kontrast potencjometrem i sprawdź działanie wyświetlacza. Następnie zmień tekst tak, aby w pierwszej linii pojawił się numer stanowiska, a w drugiej dowolny krótki komunikat.

Na koniec policz, ile **linii sygnałowych GPIO** mikrokontrolera jest zajętych przez wyświetlacz w tej konfiguracji. Zapamiętaj tę liczbę — za chwilę porównamy ją z połączeniem przez I²C.

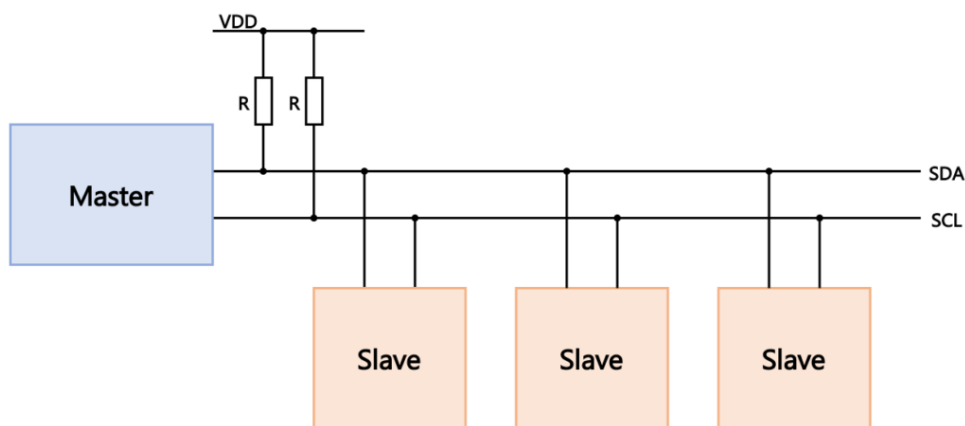
4.2 Dlaczego warto zastosować I²C?

W poprzednim zadaniu jeden wyświetlacz zajął sześć linii GPIO. W bardziej rozbudowanym systemie wbudowanym wyprowadzenia mikrokontrolera są cennym zasobem: mogą być potrzebne dla czujników, przycisków, elementów wykonawczych i innych urządzeń. Konwerter I²C pozwala sterować tym samym wyświetlaczem za pomocą tylko dwóch wspólnych linii magistrali.

I²C (*Inter-Integrated Circuit*) jest synchroniczną magistralą szeregową. Do komunikacji wykorzystywane są dwie linie:

- SDA — linia danych,
- SCL — linia zegara.

W przeciwieństwie do typowego połączenia UART, na tej samej parze przewodów może pracować wiele urządzeń (patrz rysunek 3). Urządzenie sterujące (np. mikrokontroler) inicjuje transmisję, a urządzenia peryferyjne są rozpoznawane za pomocą adresów.



Rysunek 3: Schemat podłączenia urządzeń do magistrali I²C. Urządzenie *master* to np. mikrokontroler a urządzenia *slave* to np. czujniki.

Na Arduino UNO R4 WiFi wykorzystamy:

Linia I ² C	Pin Arduino
SDA	A4 / SDA
SCL	A5 / SCL

4.3 Dlaczego potrzebne są rezystory podciągające?

Linie SDA i SCL nie działają jak zwykłe wyjścia cyfrowe, które aktywnie ustawiają zarówno stan niski, jak i wysoki. Urządzenia dołączone do magistrali mogą przede wszystkim **wymuszać stan niski**. Gdy żadne urządzenie nie wymusza stanu niskiego, rezystor podciągający ustala na linii stan wysoki.

Dzięki temu kilka urządzeń może bezpiecznie współdzielić tę samą linię: żadne z nich nie próbuje jednocześnie wymusić stanu wysokiego, gdy inne wymusza stan niski.

Pull-up w I²C a INPUT_PULLUP



W Laboratorium 1 rezystor podciągający wykorzystaliśmy przy przycisku, aby wejście miało określony stan logiczny, gdy przycisk jest zwolniony. W I²C idea jest podobna: rezystor zapewnia stan wysoki wtedy, gdy żadne urządzenie nie ściąga linii do masy. Różni się natomiast sposób wykorzystania linii i wymagania elektryczne magistrali.

W używanym konwerterze LCD rezystory podciągające są zazwyczaj już umieszczone na module, dlatego w tym ćwiczeniu nie dokładamy ich osobno.

4.4 Adresowanie i potwierdzenie ACK

Każde urządzenie peryferyjne na magistrali I²C posiada adres. Urządzenie sterujące rozpoczyna transmisję od wskazania adresu, a urządzenie o tym adresie może potwierdzić swoją obecność sygnałem ACK (*acknowledge*).

Dzięki temu można przygotować program, który sprawdzi kolejne adresy i wykryje urządzenia podłączone do magistrali.

Zadanie 2.2 — Skaner I²C



Odłącz zasilanie i podłącz wyświetlacz LCD z konwerterem I²C:

LCD I ² C	Arduino UNO R4 WiFi
VCC	5V
GND	GND
SDA	SDA / A4
SCL	SCL / A5

Wgraj poniższy program:

```

1 #include <Wire.h>
2
3 void setup() {
4     Serial.begin(115200);

```



```

6   Wire.begin();
7
8   Serial.println("Skanowanie magistrali I2C...");
9
10  for (byte address = 1; address < 127; address++) {
11      Wire.beginTransmission(address);
12      byte error = Wire.endTransmission();
13
14      if (error == 0) {
15          Serial.print("Znaleziono urządzenie: 0x");
16          if (address < 16) Serial.print("0");
17          Serial.println(address, HEX);
18      }
19  }
20
21  void loop() {
22  }

```

Otwórz monitor portu szeregowego i zanotuj znaleziony adres. Następnie odłącz LCD, ponownie uruchom skaner i porównaj wyniki.

Co robi skaner?



Funkcja `Wire.beginTransmission(address)` rozpoczyna przygotowanie transmisji do urządzenia o wskazanym adresie. `Wire.endTransmission()` kończy transmisję i zwraca kod błędu. Wartość 0 oznacza, że urządzenie odpowiedziało potwierdzeniem ACK.

Skaner nie wie, *jakim* urządzeniem jest znaleziony element. Potrafi jedynie stwierdzić, że pod danym adresem urządzenie odpowiada.

4.5 Wyświetlacz LCD jako urządzenie I2C

Klasyczny wyświetlacz LCD 16x2 wymaga kilku linii sterujących. Zastosowany na stanowisku konwerter I²C pozwala sterować nim poprzez magistralę SDA/SCL. Biblioteka `LiquidCrystal_I2C` realizuje cały protokół wysyłania danych do wyświetlacza LCD.

Zadanie 2.3 — Pierwszy tekst na LCD przez I²C



Do obsługi wyświetlacza wykorzystaj bibliotekę `LiquidCrystal_I2C`, którą można pobrać bezpośrednio z poziomu Arduino IDE. W tym celu wybierz: *Narzędzia* → *Zarządzaj bibliotekami.....* Następnie wpisz nazwę szukanej biblioteki, odnajdź właściwą z listy i zainstaluj.

W poniższym programie zastąp adres 0x27 adresem znalezionym przez skaner:



```
1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3
4 LiquidCrystal_I2C lcd(0x27, 16, 2);
5
6 void setup() {
7     lcd.init();
8     lcd.backlight();
9
10    lcd.setCursor(0, 0);
11    lcd.print("PSW - Lab 2");
12
13    lcd.setCursor(0, 1);
14    lcd.print("Stanowisko: 1");
15 }
16
17 void loop() {
18 }
```

Uruchom program i sprawdź działanie wyświetlacza. Zmień numer stanowiska na właściwy. Następnie zmodyfikuj tekst tak, aby wykorzystać obie linie wyświetlacza do przedstawienia własnego krótkiego komunikatu.

Porównaj to połączenie z zadaniem 2.1. Ile linii sygnałowych Arduino zajmuje teraz LCD? Zwróć uwagę, że linie SDA i SCL mogą być jednocześnie współdzielone z innymi urządzeniami I²C.

Pozycja kursora



Wywołanie `lcd.setCursor(kolumna, wiersz)` ustawia miejsce rozpoczęcia kolejnego zapisu. Dla wyświetlacza 16x2 kolumny numerowane są od 0 do 15, a wiersze od 0 do 1.

Zadanie 2.4 — Licznik naciśnięć na LCD



Podłącz przycisk do pinu D2 w taki sam sposób jak w Laboratorium 1 i skonfiguruj wejście z wykorzystaniem wewnętrznego rezystora podciągającego:

```
1 const int BUTTON_PIN = 2;
2
3 void setup() {
4     pinMode(BUTTON_PIN, INPUT_PULLUP);
5     // inicjalizacja LCD
6 }
```

Napisz program, który zlicza **kolejne naciśnięcia** przycisku i wyświetla ich



liczbę na LCD, np.:

Licznik:

17

Zastosuj poznaną w Laboratorium 1 programową eliminację drgań styków, aby pojedyncze fizyczne naciśnięcie nie zwiększało licznika kilka razy.

Aktualizacja tekstu na LCD



Jeżeli wcześniej na wyświetlaczu znajdowała się liczba o większej liczbie cyfr, samo wypisanie krótszej liczby może pozostawić na ekranie stare znaki. Możesz przed ponownym zapisem wyczyścić odpowiedni fragment ekranu lub nadpisać pozostałe pozycje spacjami.

5 Zadanie końcowe: Połączenie dwóch sposobów komunikacji

W ostatnim zadaniu wykorzystamy dwa interfejsy w jednym programie. Komputer będzie wysyłał tekst do Arduino przez połączenie szeregowe, a Arduino przekaże informację do wyświetlacza korzystającego z I²C.

Zadanie 3.1 — Serial Monitor → Arduino → LCD



Napisz program spełniający następującą specyfikację:

1. użytkownik wpisuje krótką wiadomość w *Serial Monitor*,
2. Arduino odbiera wiadomość przez **Serial**,
3. na pierwszej linii LCD wyświetlany jest napis `Odebrano:`,
4. na drugiej linii LCD wyświetlana jest odebrana wiadomość,
5. wysłanie kolejnej wiadomości zastępuje poprzednią zawartość drugiej linii.

Ustaw w *Serial Monitor* zakończenie linii na *Newline*. Do odczytu całej linii możesz wykorzystać:

```
1 if (Serial.available() > 0) {  
2     String message = Serial.readStringUntil('\n');  
3     message.trim();  
4  
5     // wyświetlenie message na LCD  
6 }
```

Pamiętaj, że LCD ma tylko 16 znaków w jednym wierszu. Zdecyduj, co program ma zrobić, gdy użytkownik wpisze dłuższą wiadomość. Najprostsze rozwiązanie to wyświetlenie tylko pierwszych 16 znaków.

Co dzieje się w zadaniu końcowym?



W programie występują dwa niezależne kanały komunikacji:

komputer → USB/Serial → Arduino → I²C → LCD.

Arduino odbiera dane jednym interfejsem, interpretuje je, a następnie przekazuje wynik do urządzenia korzystającego z innego interfejsu. Taki sposób łączenia różnych urządzeń i protokołów jest bardzo częsty w praktycznych systemach wbudowanych.

6 Podsumowanie

Po tych zajęciach powinieneś umieć



- wyjaśnić, czym różni się transmisja asynchroniczna UART od synchronicznej I²C,
- wskazać linie TX i RX oraz poprawnie połączyć nadajnik z odbiornikiem,
- wyjaśnić znaczenie szybkości transmisji UART,
- odczytać bajty za pomocą `Serial1.available()` i `Serial1.read()`,
- opisać, czym są komunikaty NMEA,
- wskazać linie SDA i SCL magistrali I²C,
- wyjaśnić rolę rezystorów podciągających w I²C,
- wyjaśnić, do czego służy adres urządzenia I²C,
- znaleźć urządzenie za pomocą skanera I²C,
- obsłużyć LCD zarówno przez połączenie równoległe, jak i przez konwerter I²C oraz porównać liczbę zajętych linii GPIO,
- wykorzystać bibliotekę do obsługi LCD,
- odebrać informację jednym interfejsem i przekazać ją do urządzenia korzystającego z innego interfejsu.

Co należy zapamiętać?



Przed opuszczeniem stanowiska odłącz zasilanie Arduino i modułów, uporządkuj przewody oraz pozostaw stanowisko w stanie umożliwiającym rozpoczęcie pracy następnej grupie.