6. Arduino - Komunikacja

Arduino ma możliwość komunikowania się z różnymi innymi urządzeniami. Podstawową komunikacją jest komunikacja z komputerem przez port szeregowy. Pozwala to na odczytywanie danych za jego pośrednictwem, ich wysyłanie, sterowanie podzespołami itd. Najczęściej komunikacje przez port szeregowy wykorzystuje się do „sprawdzania” co się dzieje w Arduino.

Komunikacja przez port szeregowy

Port szeregowy – port komputerowy, przez który dane są przekazywane w formie jednego ciągu bitów. Port ten jest zwykle zaopatrzony w specjalny układ (tak zwany uniwersalny asynchroniczny nadajnik-odbiornik), który tłumaczy ciągi bitów na bajty i na odwrót. Innymi słowy, jest to urządzenie pozwalające na wysyłanie oraz odbieranie ciągu bajtów (liter, cyfr i innych znaków). Port szeregowy arduino jest połączony z konwerterem USB, co umożliwia bezpośrednią komunikację przez USB z komputerem osobistym.

```
Serial.begin(9600);                 //uruchamia komunikację szeregową,
                                    // komenda ta powinna się znajdować
                                    // w setup().

Serial.println("ciąg znaków");      //wysyła ciąg znaków zakończony nową
                                    // linią przez port szeregowy

Serial.print("ciąg znaków");        //wysyła ciąg znaków niezakończony
                                    // nową linią przez port szeregowy
```

Z przyczyn technicznych, aby Arduino mogło wysłać do komputera, należy mu to „zapowiedzieć”, dodając do funkcji setup linijkę `Serial.begin(9600);`. Potem, w dowolnym miejscu programu, można wydać polecenie *wyślij wiadomość o treści*, przy użyciu np. `Serial.print("tresc");`. Uwaga!, komunikacja szeregowa pozwala wyłącznie na przesyłanie jedynie podstawowych znaków – do których nie należą polskie znaki.

Uwaga!

Przed wysłaniem odpowiedzi należy je sprawdzić w symulatorze lub w układzie.

Zadanie 6.1

a) Przeanalizować poniższy kod programu. Jaka wiadomość zostanie wysłana do komputera jeśli wciśniemy przycisk podłączony do portu 5. Co jeśli go puścimy? Co się będzie działo, jeśli na długo wciśniemy przycisk? Przetłumaczyć program na pseudokod.

```
int przyciskA = 5;

void setup() {
  Serial.begin(9600);
  pinMode(przyciskA, INPUT);
}

void loop() {
  if(digitalRead(przyciskA)){
    Serial.println("Przycisk A jest wcisniety.");
  }else{
    Serial.println("Przycisk A nie jest wcisniety.");
  }
  delay(100);
}
```

IDE: Wybrać Przykłady → WTD → Komunikacja_1 lub skopiować program jako nowy szkic, następnie wgrać go na Arduino. Uruchomić komunikację szeregową: w menu Narzędzia → Monitor portu szeregowego.

Symulator: skopiować kod do symulatora. Pod okienkiem *Kod* rozwinąć *Konsola szeregowa* i uruchomić symulację.

Czy zachowuje się zgodnie z przewidywaniami?

b) Zmodyfikować powyższy lub napisać nowy program wypisujący odpowiedni tekst, gdy wciśnięte zostaną odpowiednie dwa oraz trzy przyciski. Dodać informację o zwolnieniu przycisków.

Odczyt analogowy

Część wejść Arduino, poza poznanymi już funkcjami cyfrowymi posiada *zdolność* odczytu wartości analogowych. Wejścia te oznaczone są zazwyczaj jako A0-A7. Aby odczytać wartość *analogową* na wejściu, należy wykorzystać odpowiednią funkcję:

```
analogRead(port);     //odczytuje wartość analogową z portu. Zwracana
                      // wartość (0-1023) odpowiada napięciu 0-5V.
```

Zadanie 6.2

- Napisać program, który będzie wysyłał do komputera co 0,5 s napis ”*temperatura*”.
- Sprawdzić na schemacie (IDE) lub w symulatorze do którego wejścia podłączona jest środkowa noga potencjometru. Poinformować Arduino czy będzie to port wejściowy czy wyjściowy (patrz: poprzednia instrukcja *pinMode*).
- Zastąpić napis ”*temperatura*” funkcją `analogRead(OZNACZENIE_PORTU)`, gdzie OZNACZENIE_PORTU to numer pinu potencjometru (np. A0, A1, A7).
- Przyjąć, że potencjometr jest urządzeniem analogowym (liniowym), natomiast funkcja `analogRead()` zwraca: 0 w temperaturze 0 °C, 1023 w temperaturze 40 °C. Jak przeliczyć wartości od 0-1023 na temperaturę? Napisać program, wyświetlający na komputerze *aktualną temperaturę*.
- Zmodyfikować program tak, aby zapalał diodę po przekroczeniu temperatury 25 °C.

Arduino potrafi również odczytywać informacje (pojedyncze znaki) wysyłane z komputera do niego. Służy do tego komenda `Serial.read()`. Tak jak i `Serial.write()` tak i read rozumie ograniczony zestaw znaków (patrz ASCII).

```
Serial.available();   //sprawdza ile znaków czeka na interpretację
Serial.read();        //odczytuje jeden znak z wysłanego do arduino ciągu
```

Zadanie 6.3

```
int znak = 0;   //Miejsce gdzie zapamiętywany będzie odebrany znak.
int dioda1 = 8;

void setup() {
  Serial.begin(9600); //Uruchomienie portu szeregowego
                      // z prędkością 9600 bitów na sekundę.

  pinMode(dioda1, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {      //Jeżeli otrzymany nowy znak...

    znak = Serial.read();            //Zapamiętaj jaki znak został odczytany
                                     // ale tylko jeden - pozostałe 'czekają'.

    if(znak == '1'){                         //Uwaga!
      digitalWrite(dioda1, HIGH);            // Pojedyncze znaki MUSZĄ być pomiędzy apostro-
      Serial.println("Zapalam diodę 1.");    // fami: "1" to co innego niż '1'.
    }
  }
}
```

IDE: Wybrać Przykłady → WTD → Komunikacja_2 lub skopiować program jako nowy szkic, następnie wgrać go na Arduino. Uruchomić Monitor portu szeregowego.

Symulator: skopiować kod do symulatora. Pod okienkiem *Kod* rozwinąć *Konsola szeregowa* i uruchomić symulację.

Czy zachowuje się zgodnie z przewidywaniami?

a) Zmodyfikować program tak, by można było zapalić pozostałe diody.

b) Zmodyfikować program tak, by dodatkowo można było zgasić wszystkie diody – komendą oraz przyciskiem.

c) **Dodatkowe.** Wprowadzając zmienne zapamiętujące *ostatnią wartość każdej diody*, zmodyfikować program tak, by wpisanie np. *1* powodowało *zmianę stanu* diody (dla przykładu, wysłanie dwóch jedynek spowoduje zapalenie, a następnie zgaszenie diody 1).

Źródła:
pl.wikipedia.org/wiki/Port_szeregowy
Arduino 1.8.9
arduino.cc/reference

Patryk Król
v3.4