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 *port*u. Zwracana wartość (0-1023) odpowiada napięciu 0-5V.

Zadanie 6.2

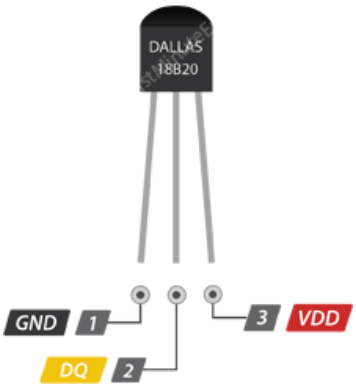
- Napisać program, który będzie wysyłał do komputera co 0,5 s napis ”*temperatura*”.
- Sprawdzić na schemacie (IDE) lub w symulatorze do którego wejścia podłączona jest środkowa noga potencjometru. Poinformować Arduino czy będzie to port wejściowy czy wyjściowy (patrz: poprzednia instrukcja *pinMode*).
- Zastąpić napis ”*temperatura*” funkcją `analogRead(OZNACZENIE_PORTU)`, gdzie OZNACZENIE_PORTU to 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.

Komunikacja z innymi urządzeniami

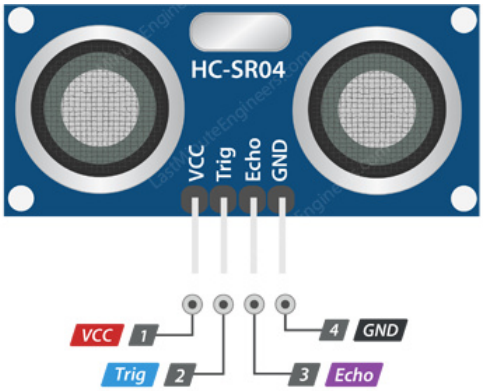
Port szeregowy nie jest jedynym standardem komunikacji pomiędzy urządzeniami. W zależności od potrzeb stosuje się różne rozwiązania – komunikację jednokierunkową lub dwukierunkową, szeregową lub równoległą, cyfrową lub analogową, na małe lub wielkie odległości itd.

Najpopularniejsze standardy i urządzenia posiadają swoje implementacje w środowisku Arduino – wystarczy więc wiedzieć jakiego urządzenia chcemy użyć i jak obsłużyć odpowiednie biblioteki.

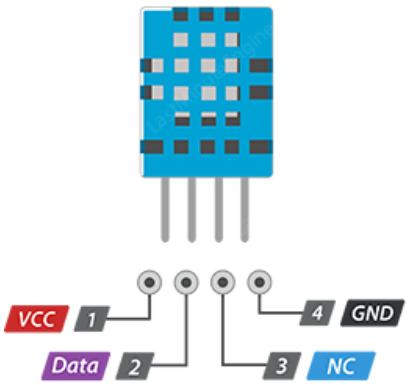
Przykładami popularnych urządzeń, do których biblioteki są dostępne są:



Termometr cyfrowy
DS18B20



Dalmierz ultradźwiękowy
HC-SR04



Termometr + wilgotnościomierz
DHT11

DS18B20 oraz DHT11 to urządzenia komunikujące się interfejsem OneWire stąd konieczność zainstalowania biblioteki OneWire oraz biblioteki odpowiedniej do urządzenia. Natomiast przykładowym urządzeniem, które nie posługuje się żadnym ze standardów, jest układ dalmierza ultradźwiękowego **HC-SR04**.

Po podaniu wysokiego stanu logicznego na wejście TRIG (o długości 10 µs) urządzenie wysyła falę dźwiękową a po jej *powrocie* podaje sygnał na wyjściu ECHO. Arduino mierzy czas pomiędzy jednym a drugim zdarzeniem, a czas ten (uwzględniając prędkość dźwięku – 340m/s) przelicza na odległość.

Zadanie 6.3 wersja na symulator

Niestety, obecna wersja symulatora nie pozwala na dodanie termometru ani higrometru. Natomiast można użyć dalmierza. Aby z niego skorzystać należy napisać własną funkcję zastępczą `measureDistanceCM` oraz użyć *Ultradźwiękowego czujnika odległości* (wersja niebieska, HC-SR04).

```
int triggerPin = 4;
int echoPin = 3;

void setup () {
    Serial.begin(9600);
    pinMode(triggerPin, OUTPUT);
    pinMode(echoPin, INPUT);
}

void loop () {
    Serial.println(measureDistanceCM());
    delay(500);
}

int measureDistanceCM() {
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    unsigned long durationMicroSec = pulseIn(echoPin, HIGH);
    double distanceCm = durationMicroSec / 2.0 * 0.0343;
    if (distanceCm == 0 || distanceCm > 400) {
        return -1.0 ;
    } else {
        return (int)distanceCm;
    }
}
```

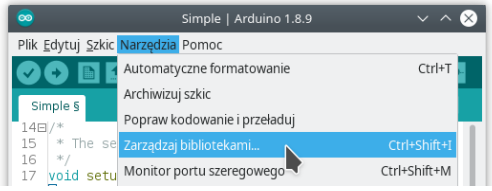
Polecenie:

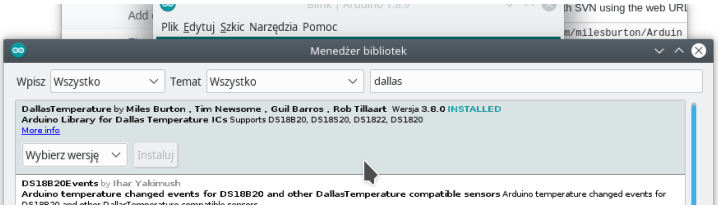
- przeanalizować powyższy program (wersja dla mniej ambitnych: do measureDistanceCM),
- w symulatorze podłączyć czujnik odległości wg schematu z poprzedniej instrukcji oraz skopiować powyższy program do *Kodu*,
- uruchomić symulację, i zweryfikować analizę. Po kliknięciu czujnika, istnieje możliwość *przestawienia* przedmiotu względem dalmierza.
- Zmodyfikować program tak, by wysyłał do komputera informację o aktualnej odległości oraz zapalał diodę gdy odległość będzie mniejsza niż 20 cm.

Zadanie 6.3 wersja na Arduino IDE

Do wykonania zadania potrzebne są dodatkowe biblioteki, należy je zainstalować (o ile jeszcze nie są zainstalowane) przy użyciu *Narzędzia → Menadżer bibliotek*, gdzie należy wyszukać odpowiednie biblioteki i je zainstalować:

OneWire, DallasTemperature, HCSR04** by **Martin Sosic** oraz **SimpleDHT** by **Winlin





a) Termometr cyfrowy

Przykładowym urządzeniem komunikującym się za pomocą protokołu 1-Wire jest termometr cyfrowy DS18B20. Z menu należy wybrać **Plik → Przykłady → DallasTemperature → Simple**. Przeanalizować program, następnie napisać taki, który będzie wysyłał do komputera informację o aktualnej temperaturze (co 5 s.) oraz zapalał diodę w przypadku przekroczenia temperatury 27 °C.

b) Dalmierz ultradźwiękowy

Z menu należy wybrać **Plik → Przykłady → HCSR04 → simple**. Przeanalizować program, a następnie napisać taki, który będzie wysyłał do komputera informację o aktualnej odległości oraz zapalał diodę gdy odległość będzie mniejsza niż 20 cm.

c) Czujnik temperatury i wilgotności

Przykładowym urządzeniem komunikującym się za pomocą protokołu 1-Wire jest termometr cyfrowy z higrometrem. Z menu należy wybrać **Plik → Przykłady → SimpleDHT → DHT11Default**. Przeanalizować program, a następnie napisać taki, który będzie wysyłał do komputera informację o aktualnej wilgotności oraz zapalał diodę gdy ta przekroczy 50% (lub inną zadaną).

Źródła:
Autodesk Eagle, Version 9.3.2, © 2019 Autodesk, Inc. All rights reserved
pl.wikipedia.org/wiki/Port_szeregowy
lastminuteengineers.com
Arduino 1.8.9

Patryk Król
v3.2