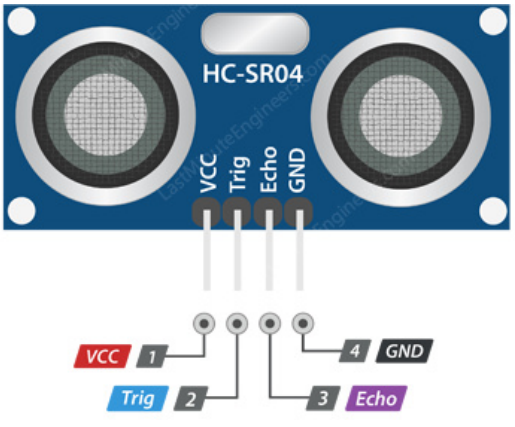
# 7. Arduino

# Komunikacja z innymi urządzeniami

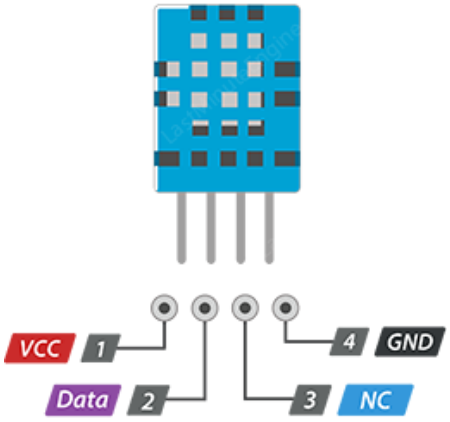
Port szeregowy nie jest jedynym standardem komunikacji pomiędzy urządzeniami. W zależności od potrzeb stosuje się różne rozwiązania – komunikację jednokierunkową lub dwukierunkową, szeregową lub równoległą, cyfrową lub analogową, na małe lub wielkie odległości itd.

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:




**Dalmierz ultradźwiękowy**
HC-SR04




**Termometr + wilgotnościomierz**
DHT11 (niebieski) lub DHT22 (biały)

DHT11 oraz DHT22 to urządzenia komunikujące się interfejsem OneWire – stąd konieczność zainstalowania biblioteki OneWire oraz biblioteki SimpleDHT. Przykładowym urządzeniem, które nie posługuje się żadnym ze standardów, jest układ dalmierza ultradźwiękowego **HC-SR04**.

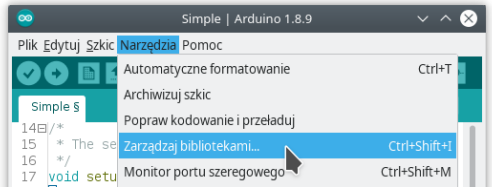
Po podaniu wysokiego stanu logicznego na wejście TRIG (o długości 10 µs) urządzenie wysyła falę dźwiękową a po jej *powrocie* podaje sygnał na wyjściu ECHO. Arduino mierzy czas pomiędzy jednym a drugim zdarzeniem, a czas ten (uwzględniając prędkość dźwięku – 340m/s) przelicza na odległość.

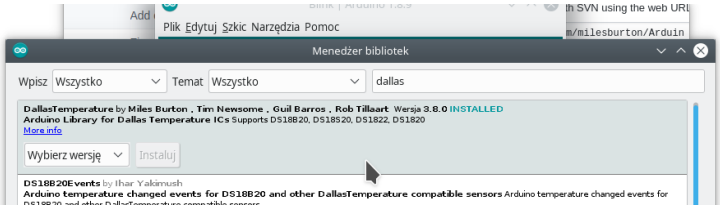
**Zadania na symulator odpowiadają zadaniom na IDE w sytuacji, gdy sprzęt fizyczny (arduino wraz z układem) jest niedostępne.**

## Zadanie 7.1 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:

***DHTNEW** by **Rob Tillart** oraz **SimpleDHT** by **Winlin***




**O podłączenie poniższych urządzeń należy zawsze poprosić prowadzącego.**

### a) Dalmierz ultradźwiękowy

- Z menu należy wybrać **Plik → Przykłady → WTD → HCSR04**. Przeanalizować program.
- Na podstawie przykładu napisać program, który będzie wysyłał do komputera informację o aktualnej odległości oraz zapalał diodę gdy odległość będzie mniejsza niż 20 cm.
- Zmodyfikować program w taki sposób, by ilość zapalonych diod odpowiadała zmierzonej odległości (np.: 10 cm – pierwsza dioda, 20 cm – druga dioda itd.).

### b) Czujnik temperatury i wilgotności

- Z menu należy wybrać: **Plik → Przykłady → WTD → DHT**. Przeanalizować program.
- Napisać taki program, który będzie wysyłał do komputera informację o aktualnej wilgotności oraz zapalał diodę gdy ta przekroczy 50% (lub inną zadaną).
- Rozbudować program tak, aby po przekroczeniu zadanej temperatury (np. 27 st. C.) zapalała się inna dioda.
- Rozbudować program tak, żeby w przypadku przekroczenia temperatury ORAZ wilgotności zapalały się wszystkie diody.
- Dodatkowe (trudne): zmodyfikować zadanie tak, by po przekroczeniu odpowiednie diody migały.

**Zadanie 7.1 wersja na symulator** (tinkercad.com, gdy niedostępne IDE!)

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.

Serwomechanizm – zamknięty układ sterowania (układ automatyki, układ regulacji) ze sprzężeniem zwrotnym, w którym sygnałem wejściowym jest jakaś dana, taka jak położenie, prędkość czy przyspieszenie. Często jest nim przesunięcie.


MicroServo to serwomechanizm często wykorzystywany w modelarstwie i różnych urządzeniach DIY. Wymaga ono jedynie informacji w jakim położeniu ma się ustawić. Po otrzymaniu takiej informacji serwomechanizm w sposób automatyczny ustawia się w zadane położenie i kontroluje je w sposób ciągły.

**7.2. Polecenie:**

a) **Arduino IDE:** W Arduino wybrać **Przykałdy** → **WTD** → **Serwo**. Przeanalizować program i zweryfikować jego działanie.

**Symulator:** Skopiować do symulatora kod z przykładu (link*), oraz podłączyć *Mikroserwo* (niebieskie) do zasilania oraz *sygnał* do pinu D4. Przeanalizować program i zweryfikować jego działanie.

b) Zmodyfikować program w taki sposób, by początkowa wartość `potVal` wynosiła 90 stopni. Wykorzystać dwa przyciski w taki sposób, by jeden zmniejszał wartość `potVal` o jeden, a drugi ją zwiększał. Kąt ustawienia serwonapędu powinien odpowiadać wartości `potVal`.

c) Rozbudować program o przycisk ustawiający serwonapęd w pozycję początkową.

*https://raw.githubusercontent.com/PMKrol/WTDAutomatyka/main/snap/arduino/current/Arduino/libraries/WTD/examples/Serwo/Serwo.ino

Źródła:
pl.wikipedia.org/wiki/Serwomechanizm
lastminuteengineers.com
Arduino 1.8.9