Programowanie C (CP1S01005)

Politechnika Białostocka - Wydział Elektryczny

Cyfryzacja przemysłu, sem. I, studia stacjonarne I stopnia

Rok akademicki 2024/2025

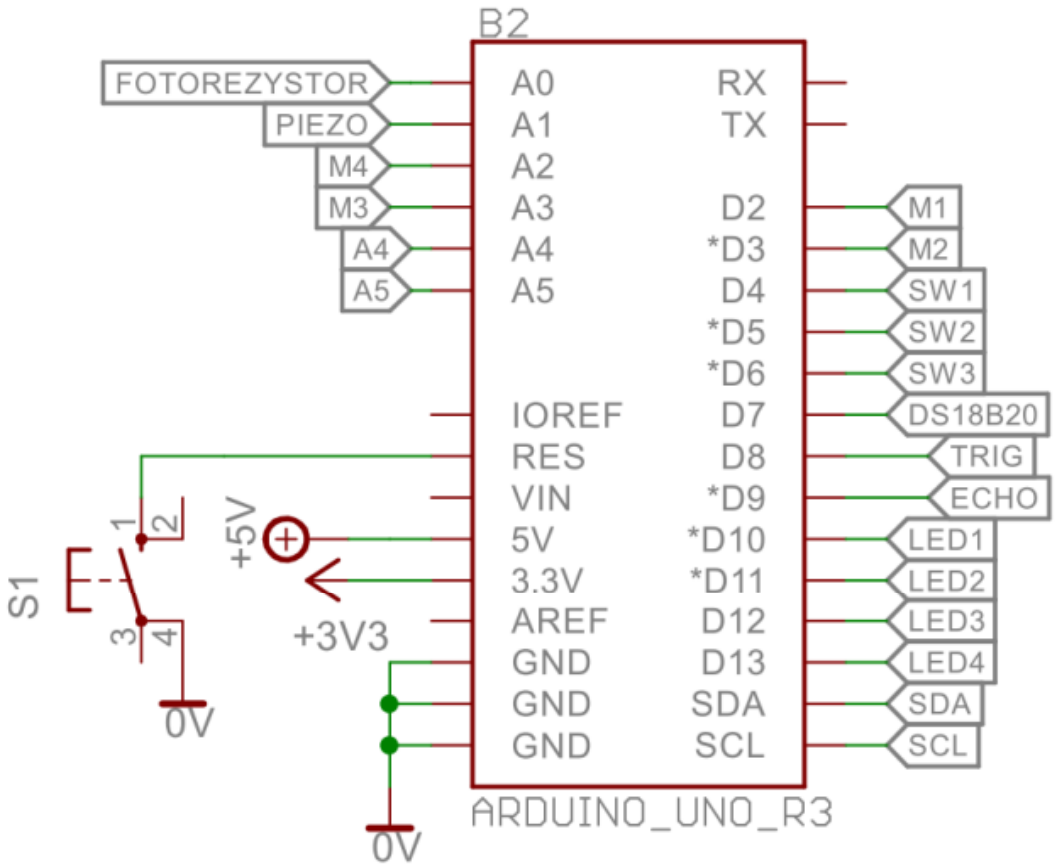
Wykład nr 6 (12.12.2024)

dr inż. Jarosław Forenc

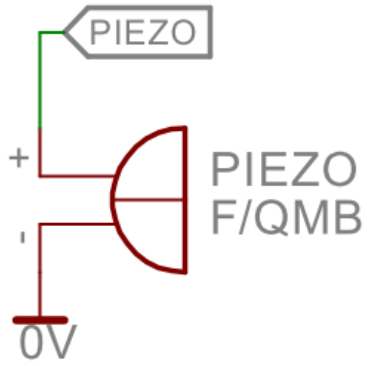
Plan wykładu nr 6

- Arduino
  - buzzer (brzęczyk piezoelektryczny)
  - fotorezystor
  - ultradźwiękowy czujnik odległości
  - czujnik natężenia światła
- Język C
  - tablice jednowymiarowe (wektory) - operacje
  - tablice dwuwymiarowe (macierze)
  - tablice wielowymiarowe
  - tablice o zmiennym rozmiarze (VLA)

Arduino - buzzer (brzęczyk piezoelektryczny)



Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
buzzera

Arduino - funkcja tone()

```
tone(pin, frequency, duration)
```

- Funkcja używana do generowania sygnału dźwiękowego o określonej częstotliwości na wskazanym pinie
- pin - numer pinu, na którym ma być generowany sygnał
- frequency - częstotliwość sygnału w hercach (Hz)
- duration - czas trwania sygnału w milisekundach (parametr opcjonalny), jeśli nie zostanie podany, to sygnał będzie generowany bez ograniczenia czasowego, aż do wywołania funkcji noTone()
- Tylko jeden sygnał dźwiękowy może być generowany w danym momencie
- Generuje falę kwadratową o określonej częstotliwości (i 50% współczynniku wypełnienia) na podanym pinie

Arduino - funkcja noTone()

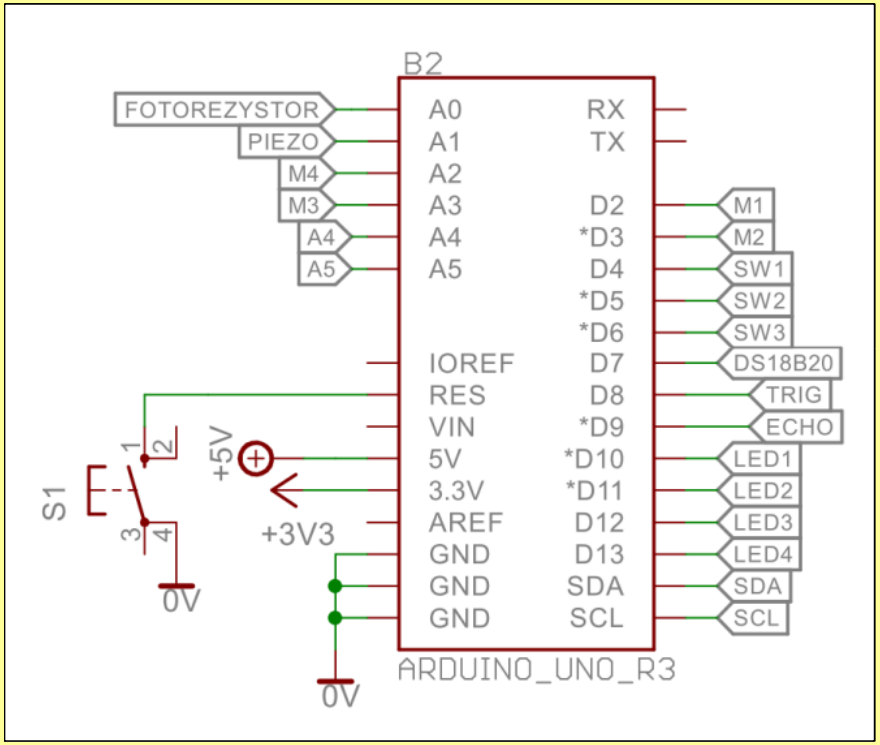
```
noTone(pin)
```

- Zatrzymuje sygnał dźwiękowy generowany przez funkcję tone() na określonym pinie
- pin - numer pinu, na którym ma być zatrzymany sygnał dźwiękowy
- Jeśli funkcja tone() nie była wcześniej wywołana dla podanego pinu, to noTone() nic nie robi

Arduino - buzzer (brzęczyk piezoelektryczny)

Program obsługujący buzzer

```
#include <Arduino.h>

#define BUZZER_PIN A1
#define SW1_PIN 4

void setup()
{
  pinMode(SW1_PIN,INPUT);
}
```

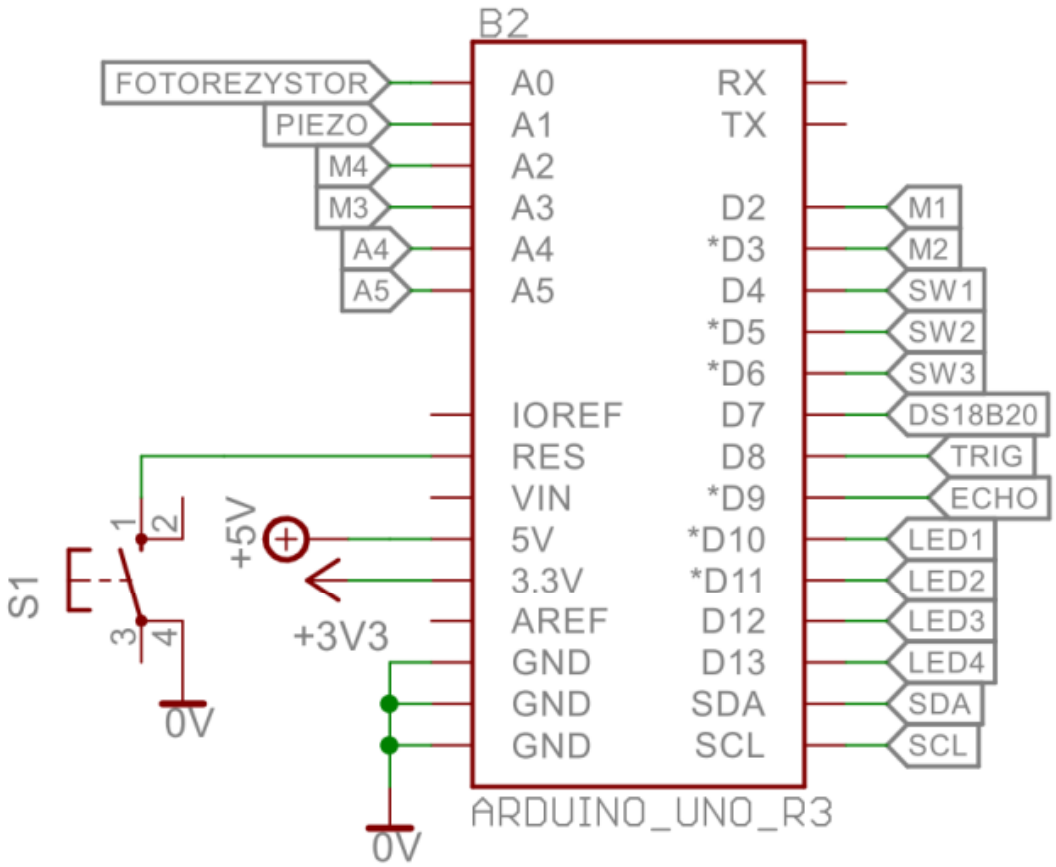


Arduino - buzzer (brzęczyk piezoelektryczny)

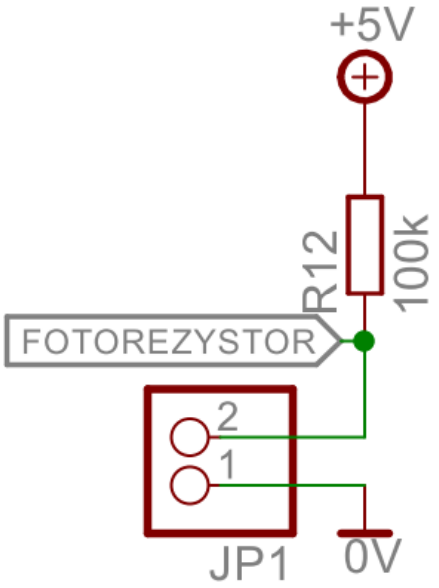
Program obsługujący buzzer

```
void loop()
{
  if(digitalRead(SW1_PIN) == LOW)
  {
    delay(200);
    tone(BUZZER_PIN,1000);
    delay(200);
    tone(BUZZER_PIN,500);
    delay(200);
    tone(BUZZER_PIN,750);
    delay(200);
    noTone(BUZZER_PIN);
  }
}
```

Arduino - fotorezystor



Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
fotorezystora

Arduino - fotorezystor

- Fotorezystor - element półprzewodnikowy, którego rezystancja zmienia się w zależności od natężenia padającego na niego światła
- Wraz ze wzrostem natężenia oświetlenia rezystancja fotorezystora maleje
- Fotorezystor jest podłączony do analogowego wyjścia A0
- Arduino posiada 10-bitowy przetwornik analogowo-cyfrowy
- Napięcie w zakresie od 0 do 5 V doprowadzone do wejścia analogowego Arduino jest przetwarzane przez przetwornik na liczbę całkowitą z przedziału od 0 do 1023
- Liczbę tę można odczytać, wywołując funkcję analogRead()
- Do wyświetlenia jej wartości wykorzystane zostało okno monitora portu szeregowego

Arduino - funkcja analogRead()

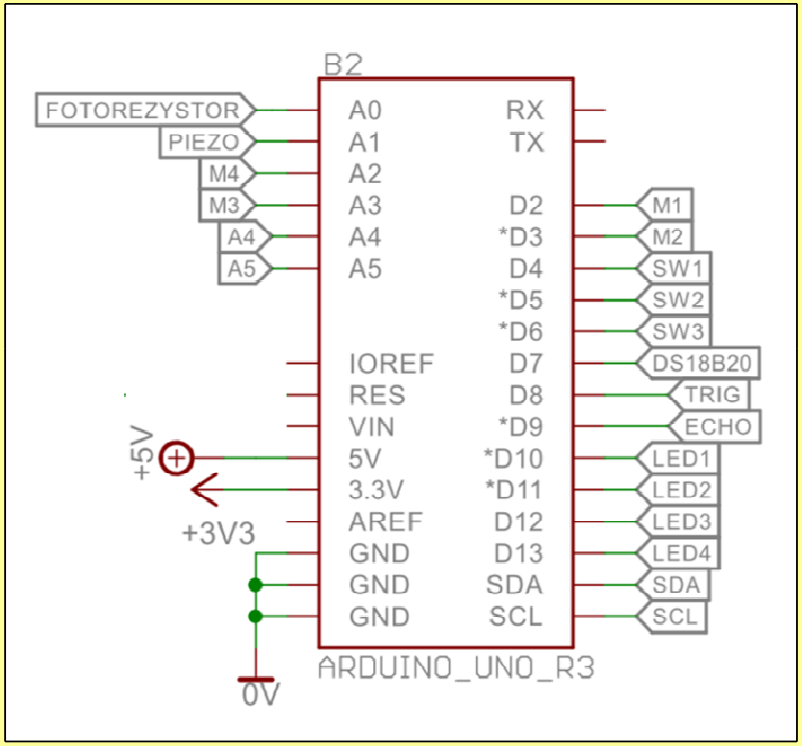
```
analogRead(pin)
```

- Odczytuje wartość analogową z jednego z pinów analogowych (A0-A5)
- pin - numer pinu analogowego, z którego odczytujemy wartość
- Zwraca wartość typu int z przedziału od 0 do 1023
- Zwrócona wartość jest proporcjonalna do zmierzonego napięcia
- Wartość 0 odpowiada 0 V, a 1023 odpowiada 5 V

Arduino - fotorezystor

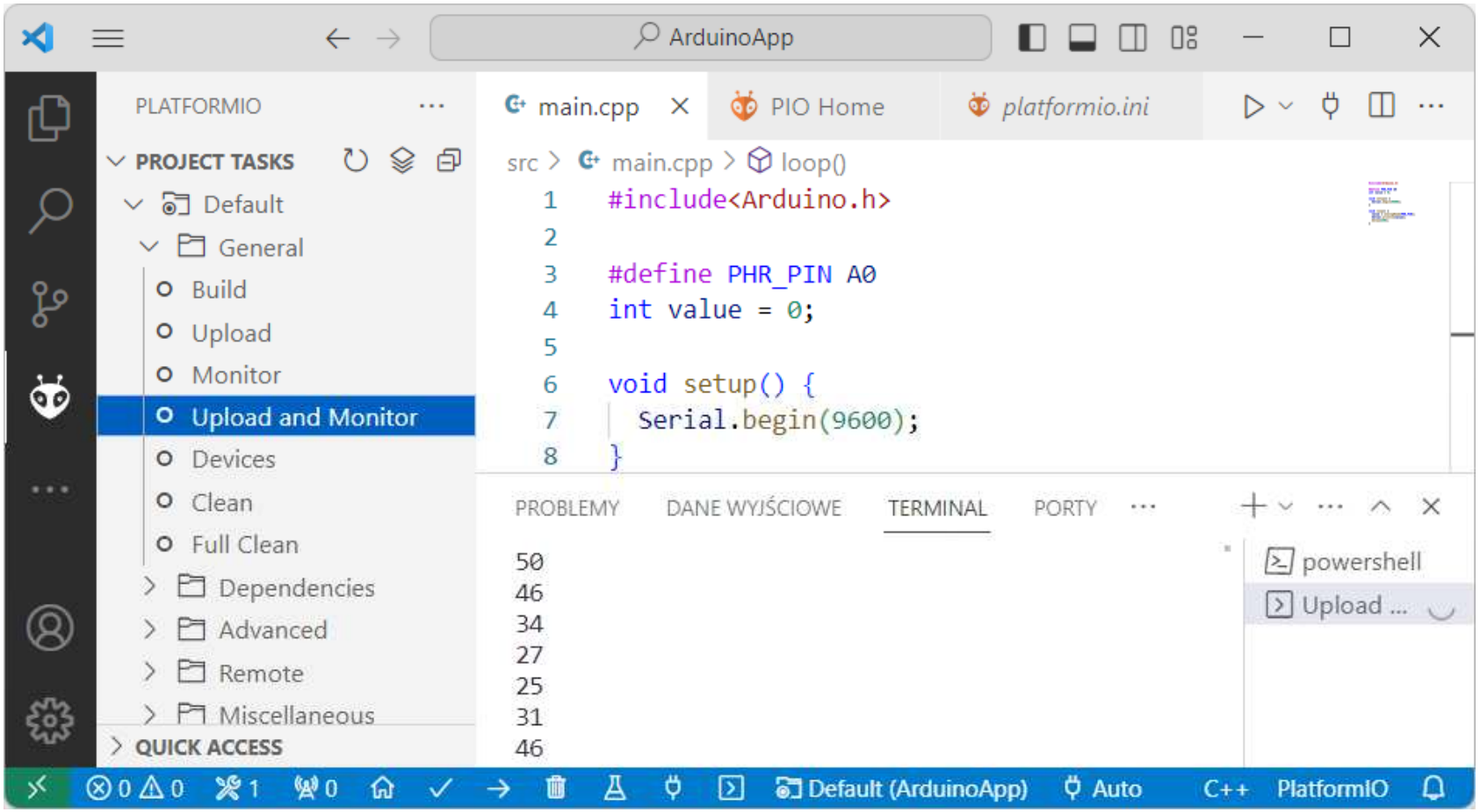
Program obsługujący fotorezystor

```
#include<Arduino.h>

#define PHR_PIN A0
int value = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  value = analogRead(PHR_PIN);
  Serial.println(value);
  delay(250);
}
```

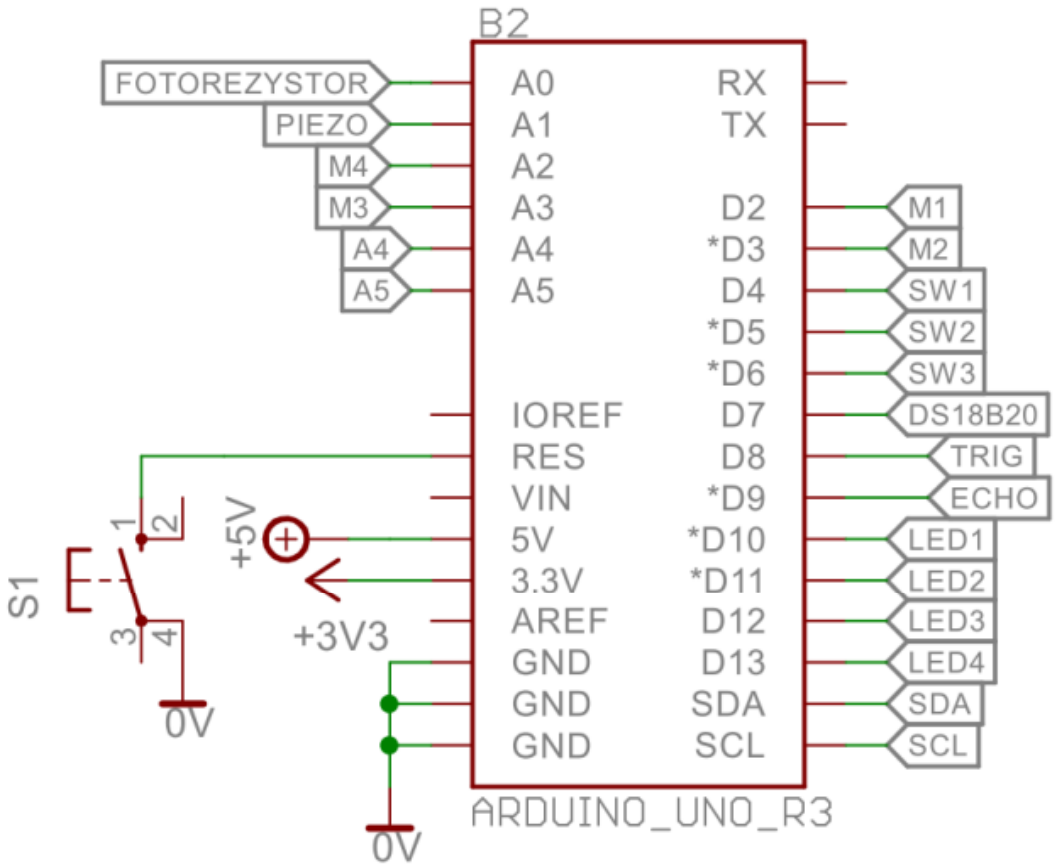


Arduino - fotorezystor

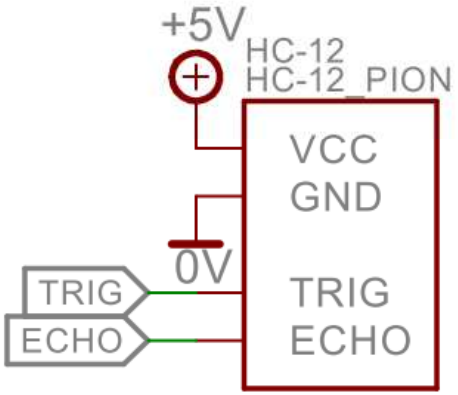
- Uruchomienie programu: PlatformIO → PROJECT TASK → Default → General → Upload and Monitor



Arduino - czujnik odległości (HC-SR04)

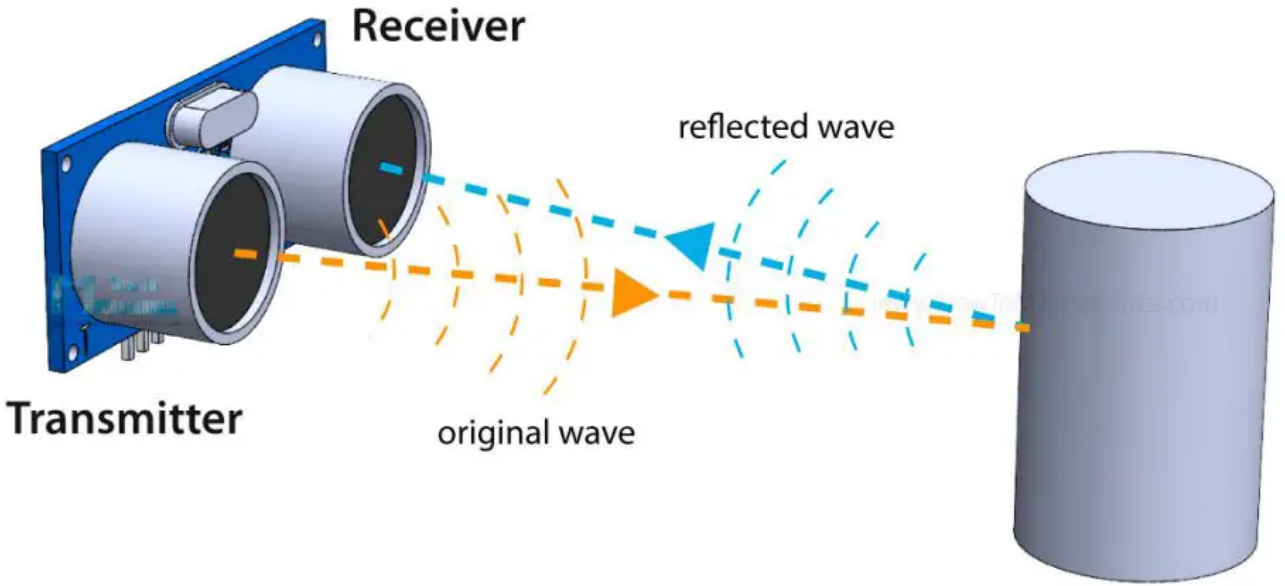


Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
czujnika HC-SR04

Arduino - czujnik odległości (HC-SR04)

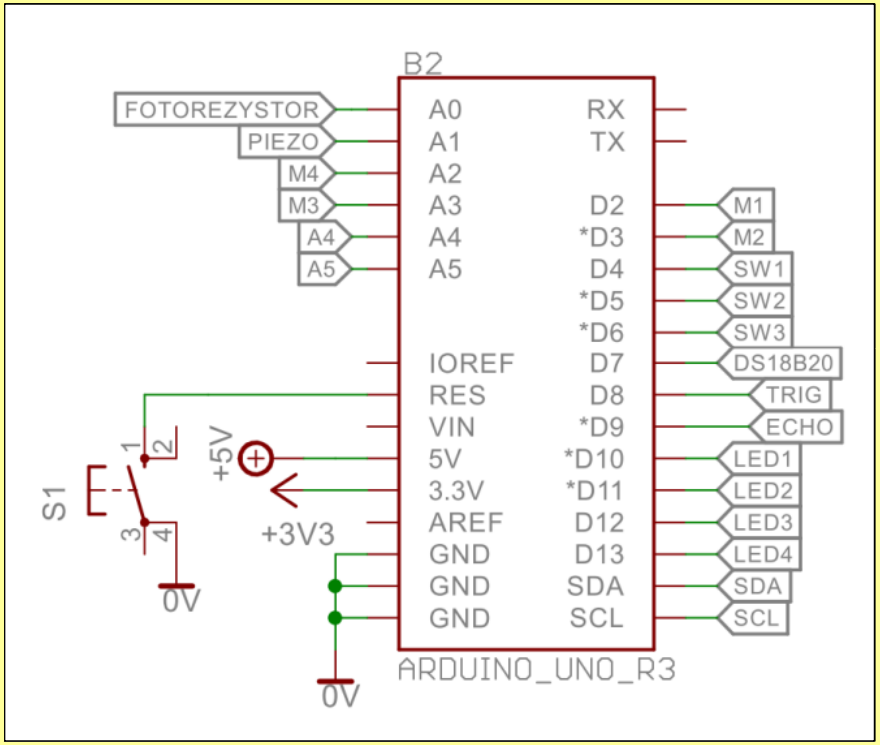


- Czujnik składa się z nadajnika i odbiornika, układu sterowania oraz generatora sygnału pomiarowego
- Nadajnik wysyła sygnał ultradźwiękowy, który odbija się od obiektu i wraca do odbiornika
- Odległość jest obliczana na podstawie różnicy czasu, który upływa od chwili wysłania sygnału pomiarowego, do chwili jego odbioru

Arduino - czujnik odległości (HC-SR04)

Czujnik odległości

```
#include<Arduino.h>

#define TRIG_PIN 8
#define ECHO_PIN 9

long duration;
int distance;

void setup()
{
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}
```



Arduino - czujnik odległości (HC-SR04)

Czujnik odległości

```
void loop()
{
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Odlegosc: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(250);
}
```

Arduino - czujnik odległości (HC-SR04)

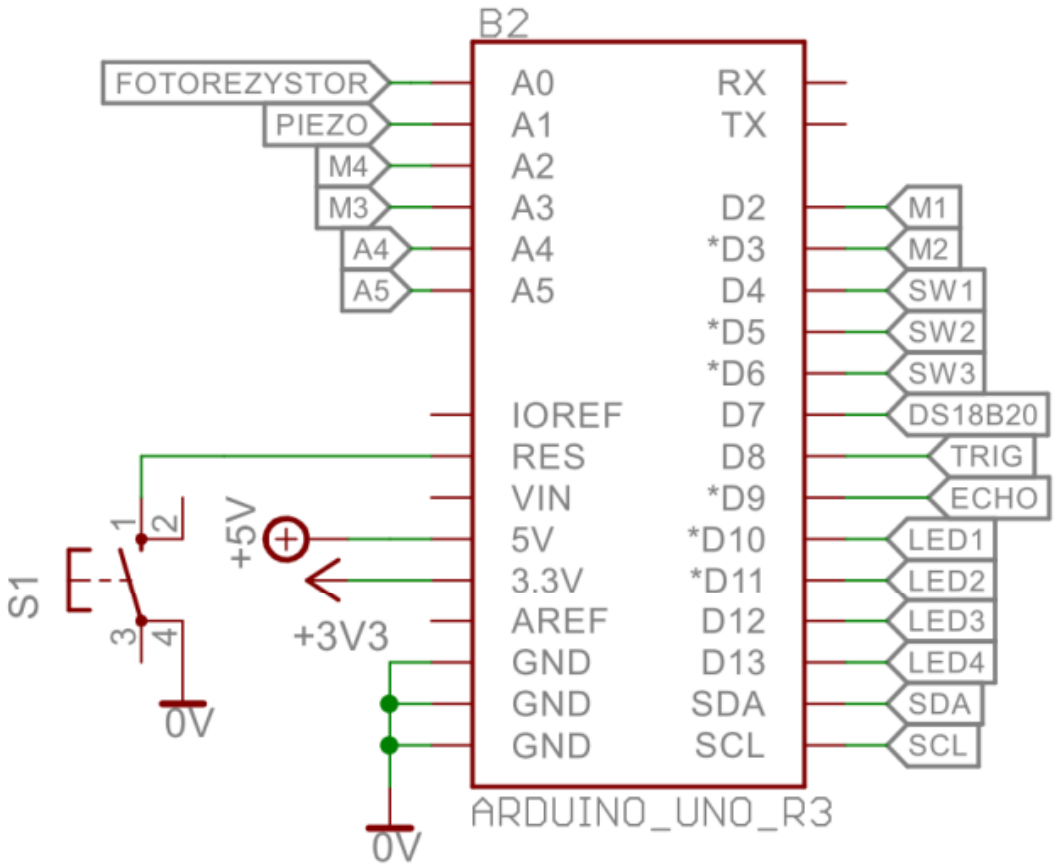
- Zasada pomiaru:
  - po podaniu na wejście Trig stanu wysokiego o czasie trwania 10 µs, wbudowany generator wysyła na nadajnik czujnika sygnał ultradźwiękowy o częstotliwości 40 kHz przez czas 200 µs
  - po zakończeniu wysyłania tego sygnału pin Echo zmienia stan na wysoki, rozpoczynając pomiar
  - wyemitowana fala ultradźwiękowa jest wysyłana i gdy napotka na drodze obiekt, zostaje odbita w stronę odbiornika
  - zmierzony czas trwania stanu wysokiego na pinie Echo (funkcja pulseIn()) jest proporcjonalny do odległości czujnika od przeszkody
  - odległość d (w cm) jest obliczana na podstawie wzoru:

$$d = \frac{t\,[\mu s] \cdot v\,\left[\frac{cm}{\mu s}\right]}{2}$$

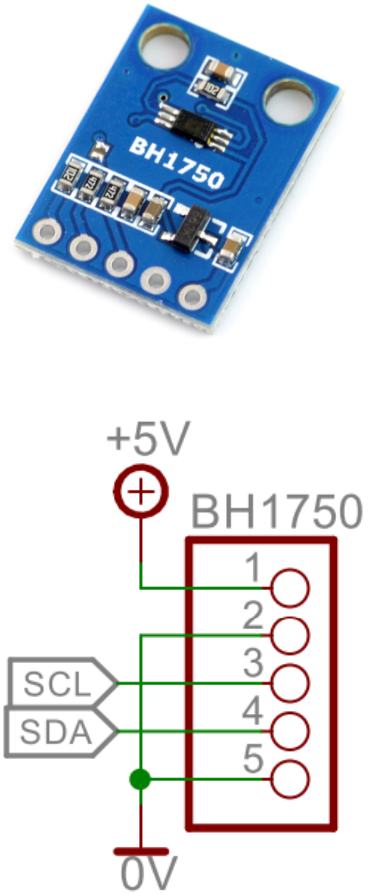
t - zmierzony czas w µs

v - prędkość dźwięku w powietrzu (v = 340 m/s = 0,034 cm/µs)

Arduino - czujnik natężenia światła (BH1750)



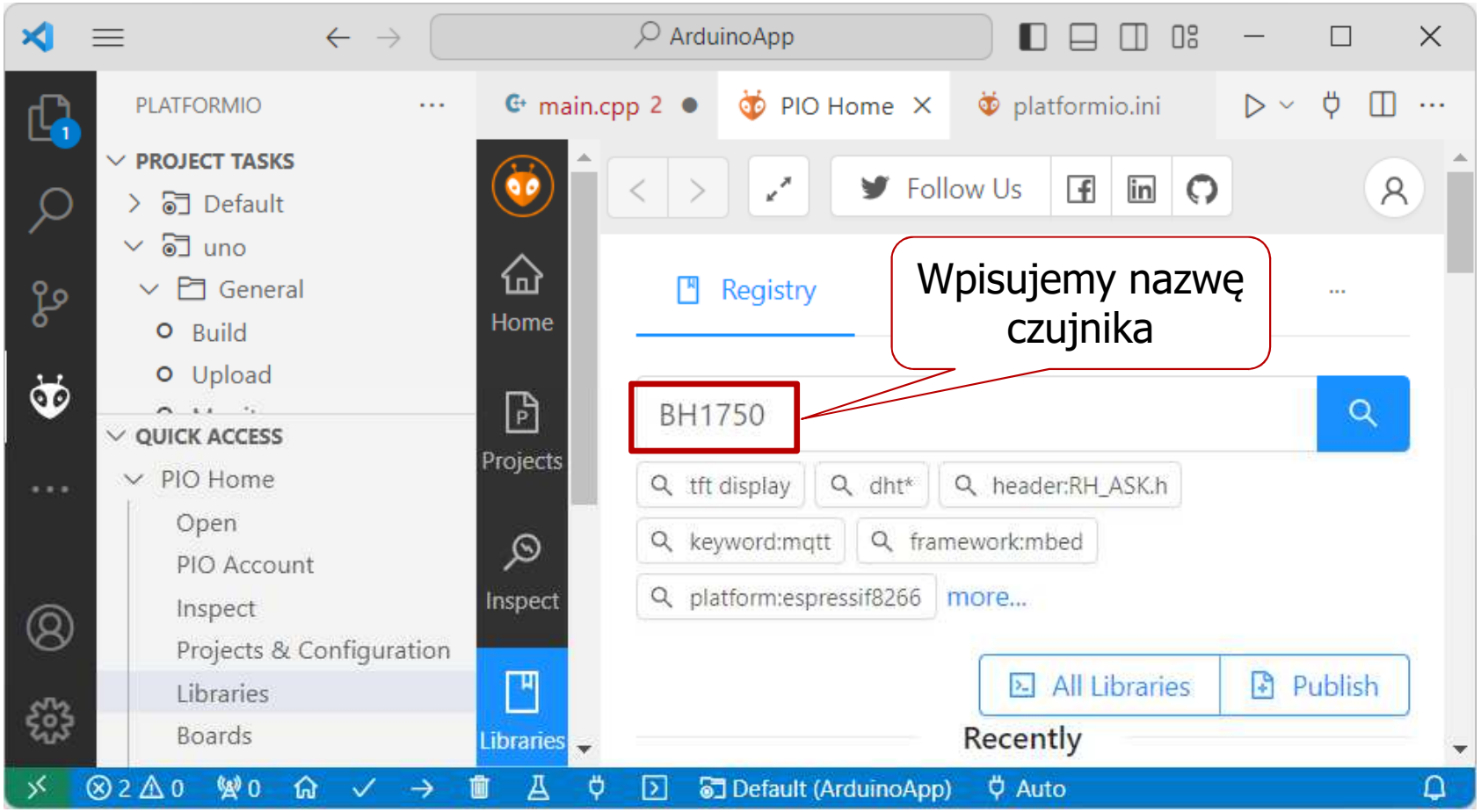
Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
czujnika BH1750

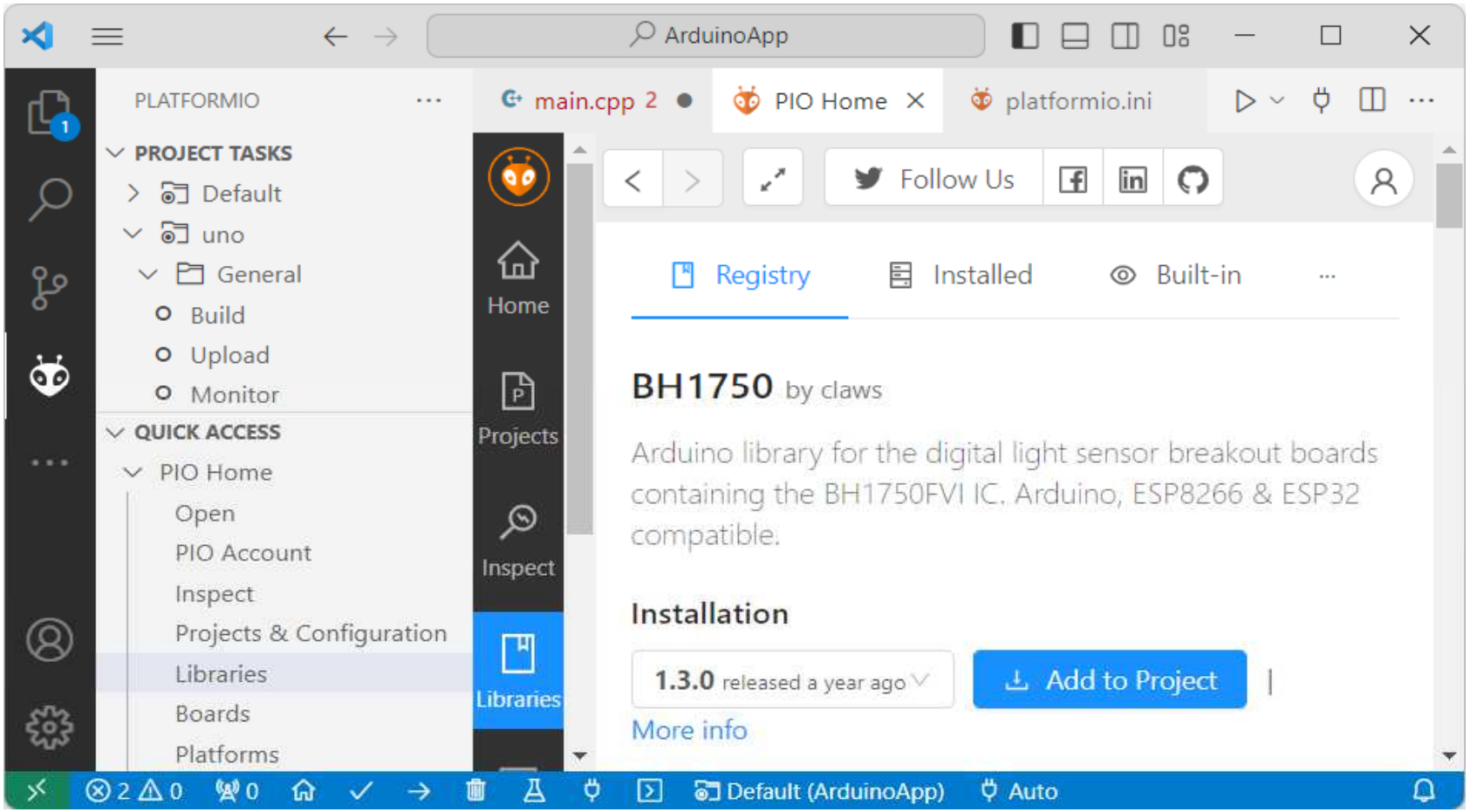
Arduino - czujnik natężenia światła (BH1750)

- Zastosowanie czujnika BH1750 wymaga zainstalowania biblioteki:
  PlatformIO → QUICK ACCESS → PIO Home → Libraries



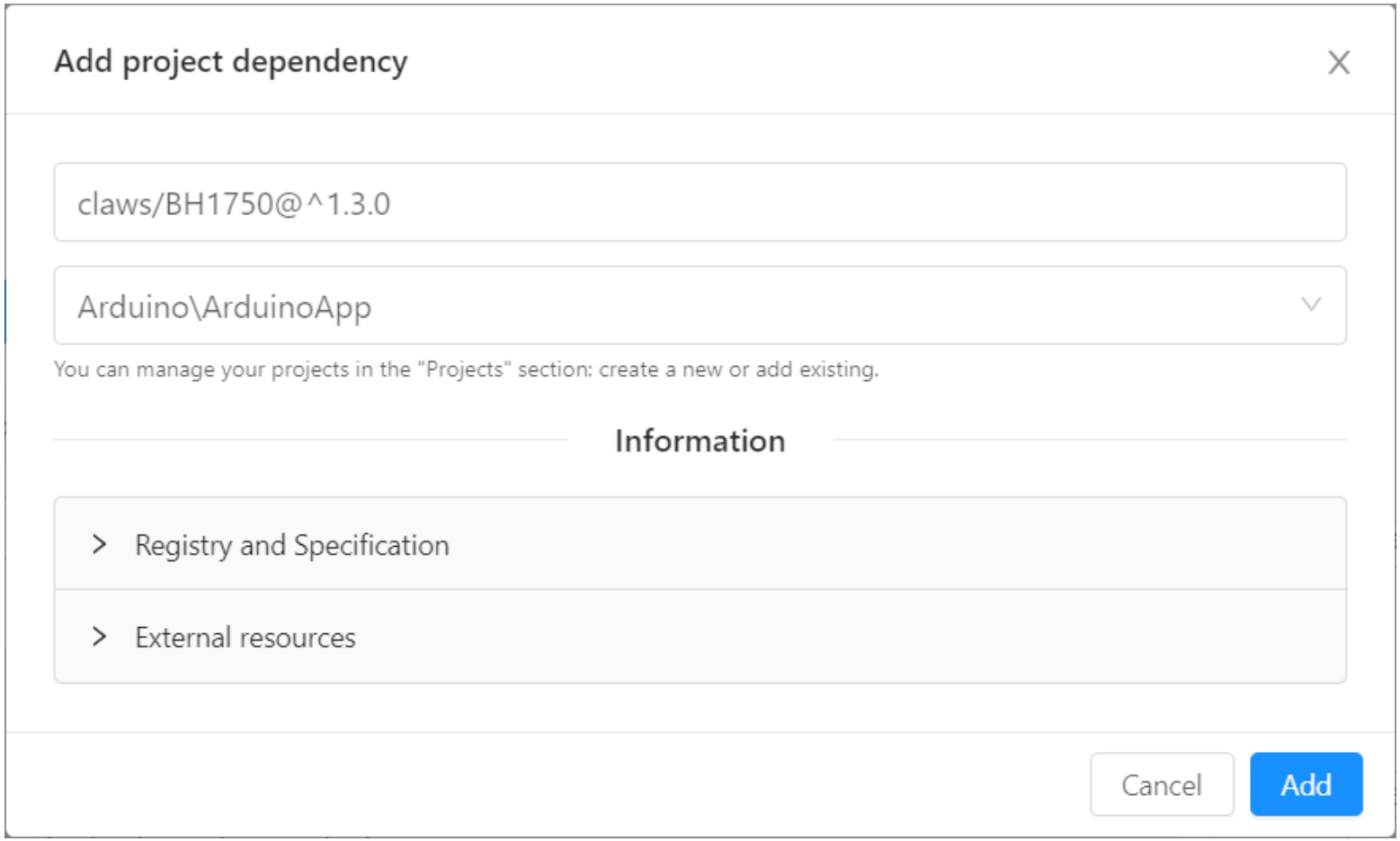
Arduino - czujnik natężenia światła (BH1750)

- Wybieramy bibliotekę o nazwie BH1750 by claws i instalujemy ją, klikając przycisk Add to Project



Arduino - czujnik natężenia światła (BH1750)

- Wybieramy projekt, do którego ma być dodana biblioteka BH1750 by claws



Arduino - czujnik natężenia światła (BH1750)

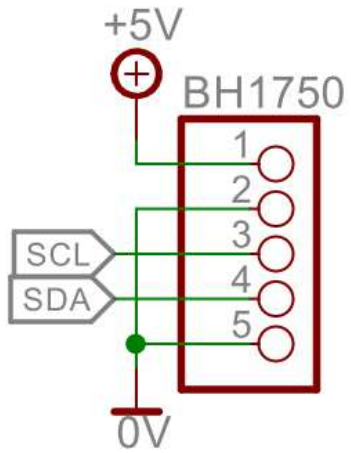
- W pliku konfiguracyjnym platformio.ini zostanie dodany wiersz:

```
lib_deps = claws/BH1750@^1.3.0
```

- W pliku .cpp z kodem źródłowym zostaną dodane dwie biblioteki

```
#include <BH1750.h>
#include <Wire.h>
```

- Czujnik komunikuje się z modułem Arduino za pomocą interfejsu I2C, wykorzystując dwie linie:
  - danych - SDA (Serial Data Line)
  - zegarową - SCL (Serial Clock Line)



Arduino - czujnik natężenia światła (BH1750)

Czujnik natężenia światła

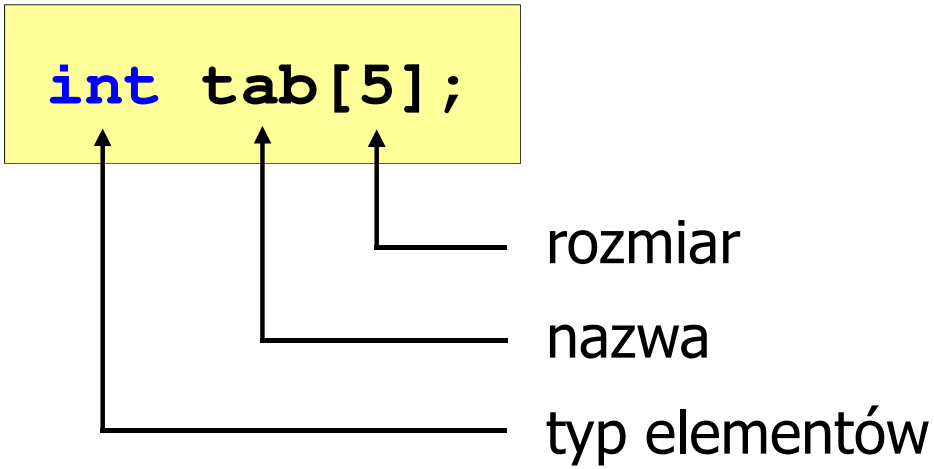
```
#include <Arduino.h>
#include <BH1750.h>
#include <Wire.h>

BH1750 LightMeter(0x23);

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  LightMeter.begin();
}
```

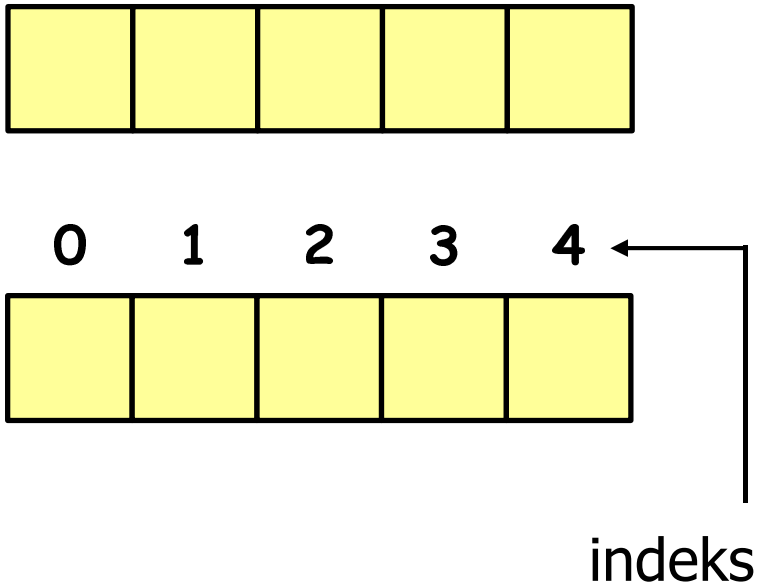
Arduino - czujnik natężenia światła (BH1750)

Czujnik natężenia światła

```
void loop()
{
  float lux = LightMeter.readLightLevel();
  Serial.print("Natezenie swiatla: ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(250);
}
```

Język C - deklaracja tablicy jednowymiarowej





- rozmiar tablicy to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

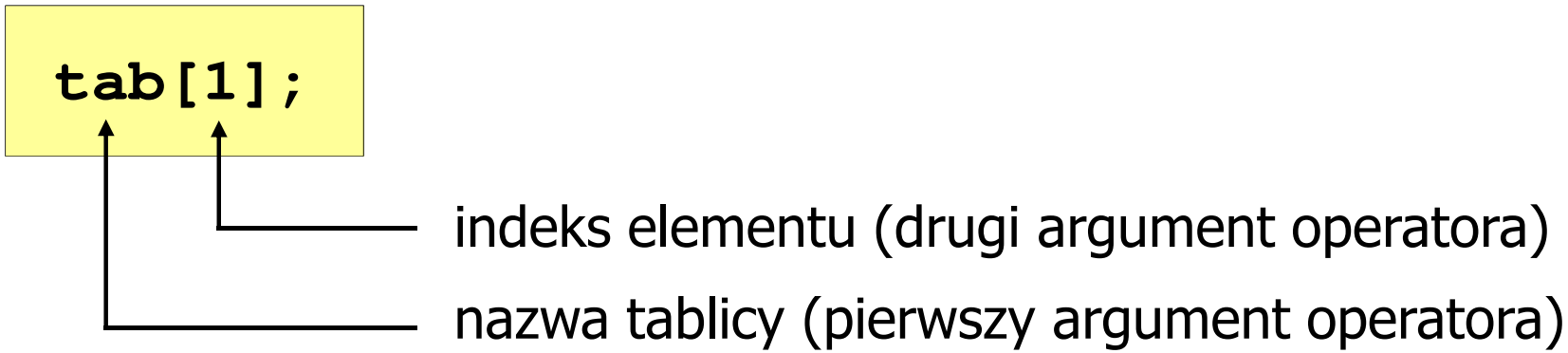
```
int tab[5];
```

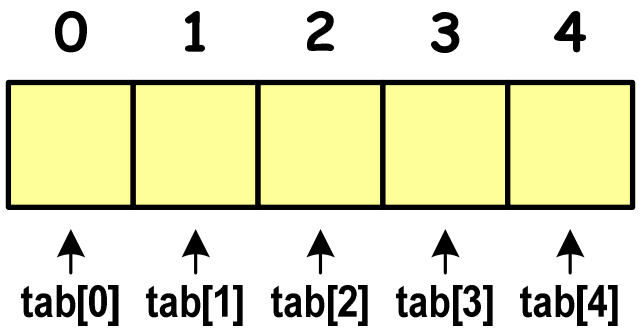
```
int tab[N];
```

```
int tab[n];
```

Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania





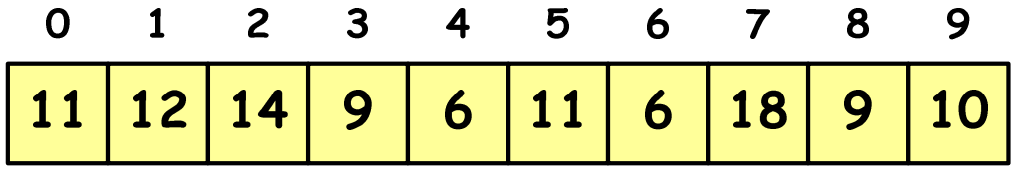
- indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np. i*j+5

Język C - generator liczb pseudolosowych

- rand() - zwraca liczbę pseudolosową - zakres: 0 ... RAND_MAX (0 ... 32767)
- srand() - inicjalizuje generator liczb pseudolosowych
- Plik nagłówkowy: stdlib.h (time.h)

```
int x, y, z;
srand((unsigned int) time(NULL));
x = rand();                  // zakres <0,32767>
y = rand() % 100;            // zakres <0,99>
z = rand() % (b-a+1)+a;      // zakres <a,b>
```

Język C - operacje na wektorze

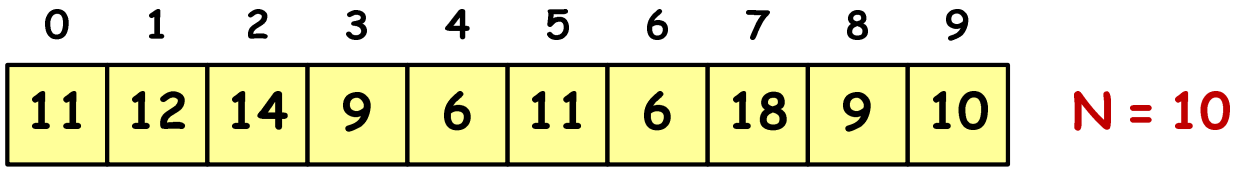



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main(void)
{
     int tab[N], i;

     /* generowanie elementów tablicy */

     srand((unsigned int) time(NULL));

     for (i=0; i<N; i++)
          tab[i] = rand() % 20;
```

Język C - operacje na wektorze

```
/* wyświetlenie elementów tablicy */

printf("Elementy tablicy:\n");
for (i=0; i<N; i++)
    printf("%d  ",tab[i]);
printf("\n");
```

```
Elementy tablicy:
11  12  14  9  6  11  6  18  9  10
```



Język C - operacje na wektorze

```
/* wyświetlenie elementów w odwrotnej kolejności */

printf("Elementy w odwrotnej kolejnosci:\n");
for (i=N-1; i>=0; i--)
    printf("%d  ",tab[i]);
printf("\n");
```

```
Elementy w odwrotnej kolejnosci:
10  9  18  6  11  6  9  14  12  11
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* wyszukanie elementu o najmniejszej wartości */

int min;

min = tab[0];
for (i=1; i<N; i++)
    if (tab[i]<min)
        min = tab[i];
printf("Wartosc elementu najmniejszego: %d\n",min);
```

```
Wartosc elementu najmniejszego: 6
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* indeksy elementów o najmniejszej wartości */

printf("Indeksy elementu najmniejszego: ");
for (i=0; i<N; i++)
    if (tab[i]==min)
        printf("%d  ",i);
printf("\n");
```

```
Indeksy elementu najmniejszego: 4  6
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* suma i średnia arytmetyczna elementów tablicy */

int suma = 0;
float srednia;

for (i=0; i<N; i++)
    suma = suma + tab[i];
srednia = (float) suma/N;
printf("Suma: %d, srednia: %g\n",suma,srednia);
```

```
Suma: 106, srednia: 10.6
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* liczba parzystych elementów tablicy  */

int ile = 0;

for (i=0; i<N; i++)
    if (tab[i]%2==0)
        ile++;
printf("Liczba parzystych elementow: %d\n",ile);
```

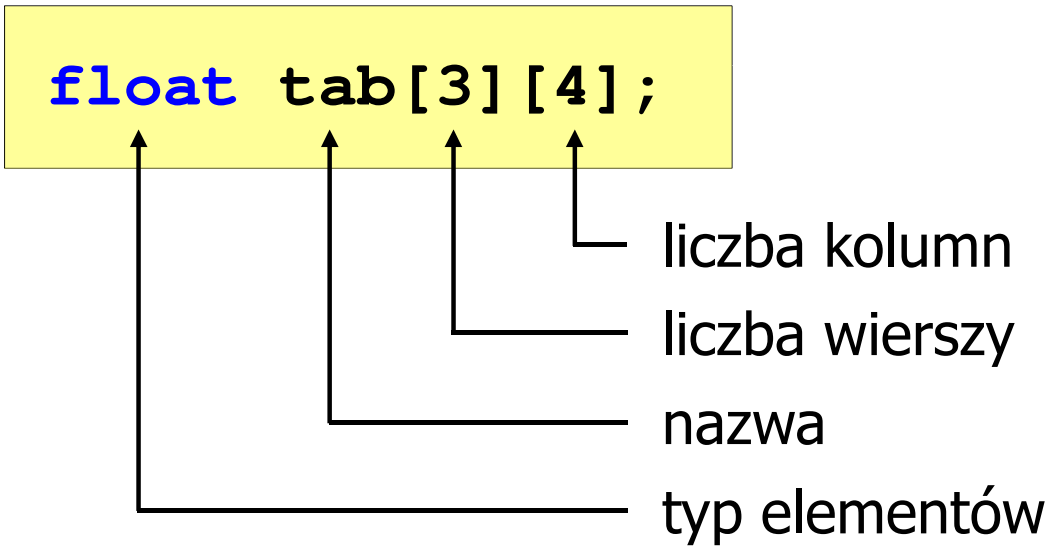
```
Liczba parzystych elementow: 6
```

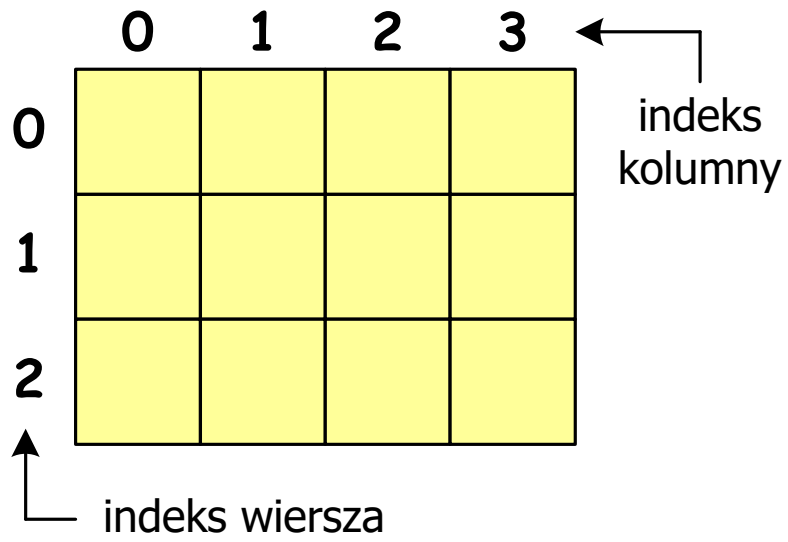
0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - deklaracja tablica dwuwymiarowej

```
float tab[3][4];
```





- Rozmiar tablicy (liczb wierszy i kolumn) to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

Język C - odwołania do elementów macierzy

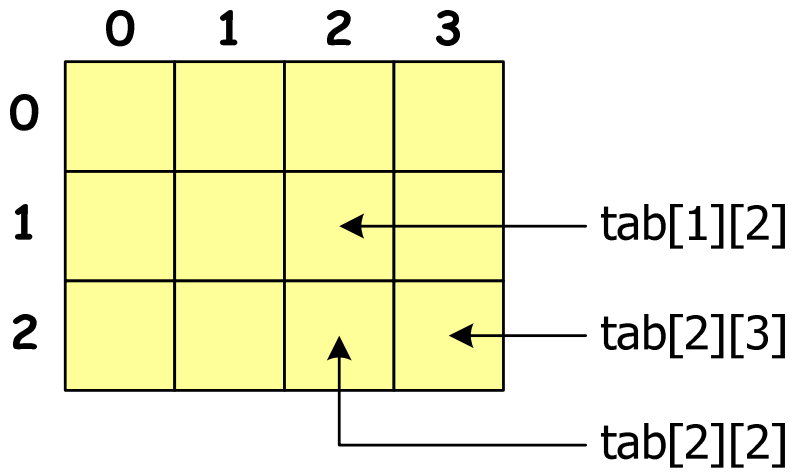
```
tab[1][2];
```

[ ] - dwuargumentowy operator indeksowania

indeks (numer) kolumny

indeks (numer) wiersza

nazwa tablicy



- Indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np. i*j+5
- Brak sprawdzania poprawności indeksów!

Język C - inicjalizacja elementów macierzy

```
int T[2][3] = {{1,2,3},{4,5,6}};
```

	0	1	2
0	1	2	3
1	4	5	6

```
int T[2][3] = {1,2,3,4,5,6};
```

	0	1	2
0	1	2	3
1	4	5	6

```
int T[2][3] = {1,2,3,4};
```

	0	1	2
0	1	2	3
1	4	0	0

```
int T[2][3] = {{1},{4,5}};
```

	0	1	2
0	1	0	0
1	4	5	0

Język C - inicjalizacja elementów macierzy

```
int T[2][3] = {0};
```

```
int T[2][3] = {};
```

wyzerowanie elementów macierzy

	0	1	2
0	0	0	0
1	0	0	0

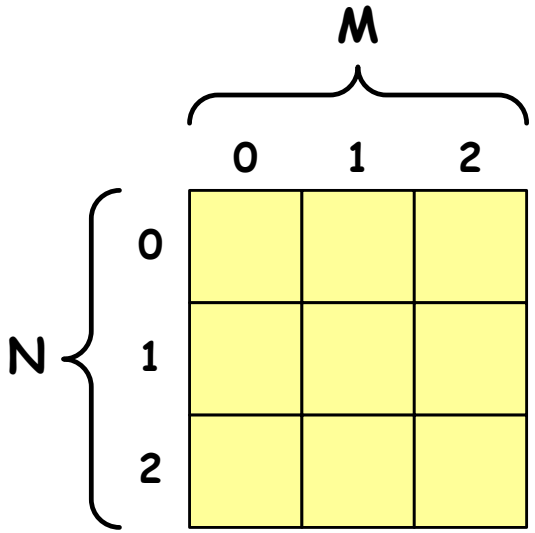
```
int T[][3] = {{1,2,3},{4,5,6}};
```

pominięcie liczby wierszy

	0	1	2
0	1	2	3
1	4	5	6

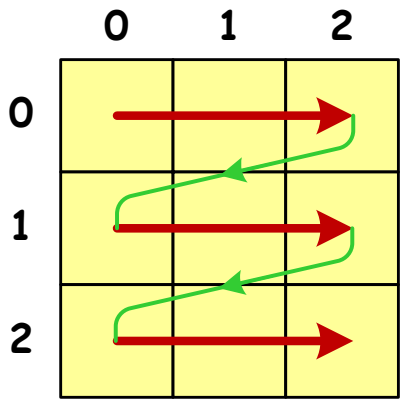
Język C - operacje na macierzy

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 3      /* liczba wierszy */
#define M 3      /* liczba kolumn */

int main(void)
{
    int  tab[N][M];
    int  i, j;
```

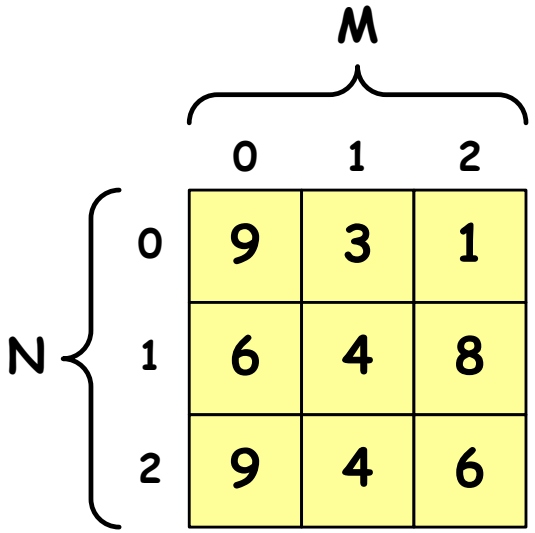


Język C - operacje na macierzy

```
/* generowanie pseudolosowe elementów macierzy */

srand((unsigned int) time(NULL));

for (i=0; i<N; i++)
    for (j=0; j<M; j++)
        tab[i][j] = rand() % 10;
```



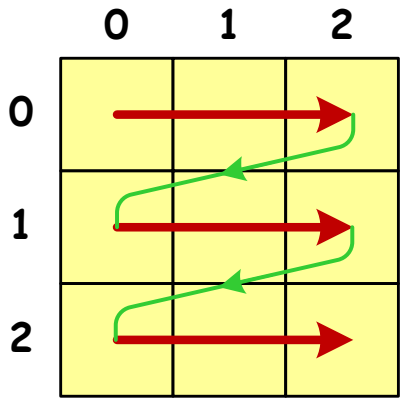
kolejność zapisywania wartości elementów macierzy



Język C - operacje na macierzy

```
/* wyświetlenie elementów macierzy */

for (i=0; i<N; i++)
{
    for (j=0; j<M; j++)
        printf("%3d",tab[i][j]);
    printf("\n");
}
```

```
  9   3   1
  6   4   8
  9   4   6
```

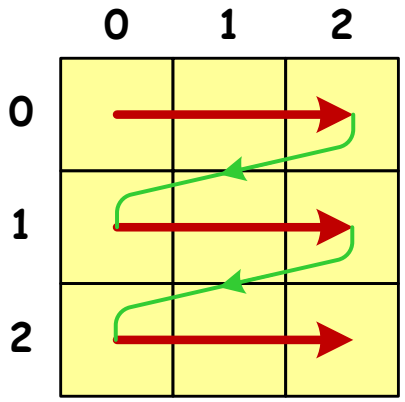


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Język C - operacje na macierzy

```
/* poszukiwanie elementu o wartości minimalnej */

int min = tab[0][0];
for (i=0; i<N; i++)
    for (j=0; j<M; j++)
        if (tab[i][j] < min)
            min = tab[i][j];
printf("Wartosc min: %d\n",min);
```

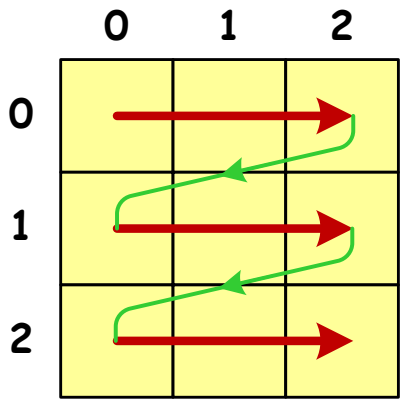
```
Wartosc min: 1
```



	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Język C - operacje na macierzy

```
/* suma i średnia arytmetyczna elementów */

int suma = 0;
for (i=0; i<N; i++)
    for (j=0; j<M; j++)
        suma = suma + tab[i][j];
float srednia = (float) suma/(N*M);
printf("Suma:     %d\n",suma);
printf("Srednia: %f\n\n",srednia);
```

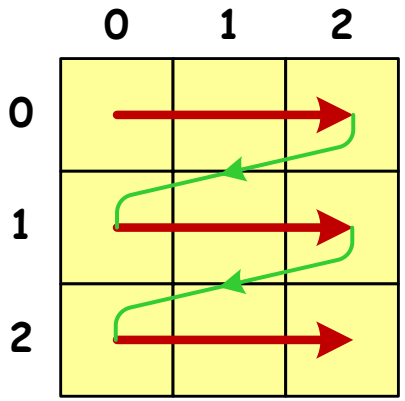


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

```
Suma:     50
Srednia: 5.555555
```

Język C - operacje na macierzy

```
/* sumy elementów w poszczególnych wierszach */

for (i=0; i<N; i++)
{
     suma = 0;
     for (j=0; j<M; j++)
          suma = suma + tab[i][j];
     printf("Suma wiersza %d = %d\n",i,suma);
}
```

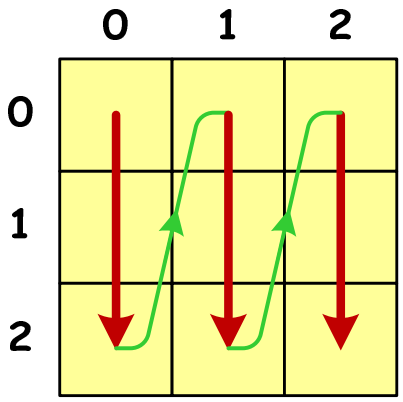


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

```
Suma wiersza 0 = 13
Suma wiersza 1 = 18
Suma wiersza 2 = 19
```

Język C - operacje na macierzy

```
/* sumy elementów w poszczególnych kolumnach */

for (j=0; j<M; j++)
{
     suma = 0;
     for (i=0; i<N; i++)
          suma = suma + tab[i][j];
     printf("Suma kolumny %d = %d\n",j,suma);
}
```



	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

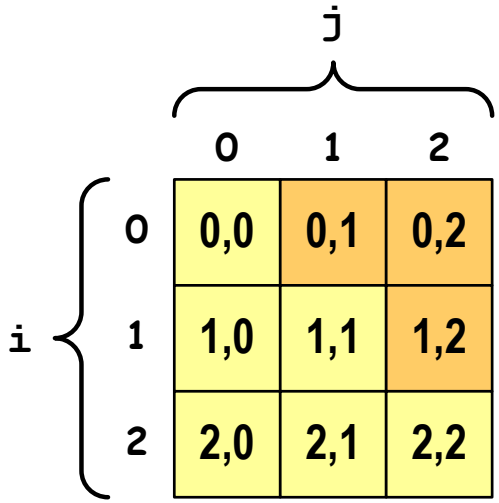
```
Suma kolumny 0 = 24
Suma kolumny 1 = 11
Suma kolumny 2 = 15
```

Język C - operacje na macierzy

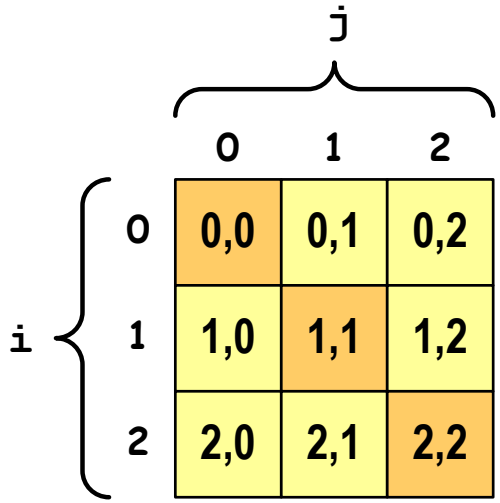
```
/* sumy elementów nad, na i poniżej przekątnej */

suma = suma1 = suma2 = 0;
for (i=0; i<N; i++)
    for (j=0; j<M; j++)
    {
        if (i < j) suma1+=tab[i][j];   /* nad */
        if (i > j) suma2+=tab[i][j];   /* pod */
        if (i == j) suma+=tab[i][j];   /* na  */
    }

printf("Suma nad: %d\n",suma1);
printf("Suma na:  %d\n",suma);
printf("Suma pod: %d\n",suma2);
```

```
Suma nad: 12
Suma na:  19
Suma pod: 19
```

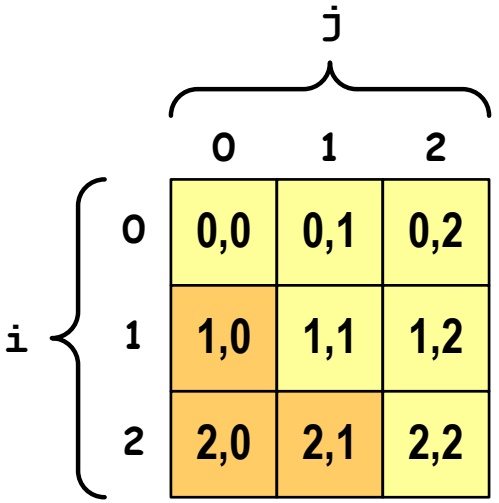
Język C - operacje na macierzy



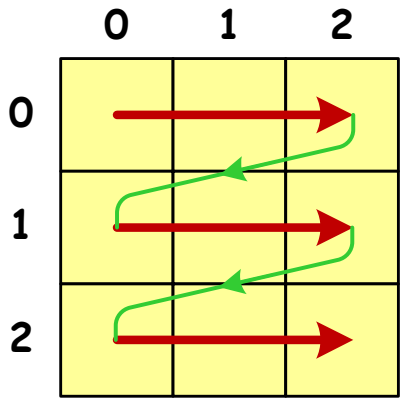
`i < j`

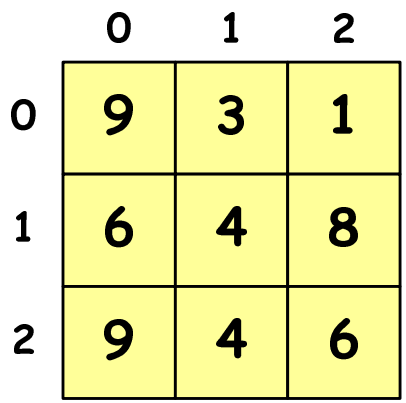


`i = j`



`i > j`






```
Suma nad: 12
Suma na:  19
Suma pod: 19
```

Język C - tablice wielowymiarowe

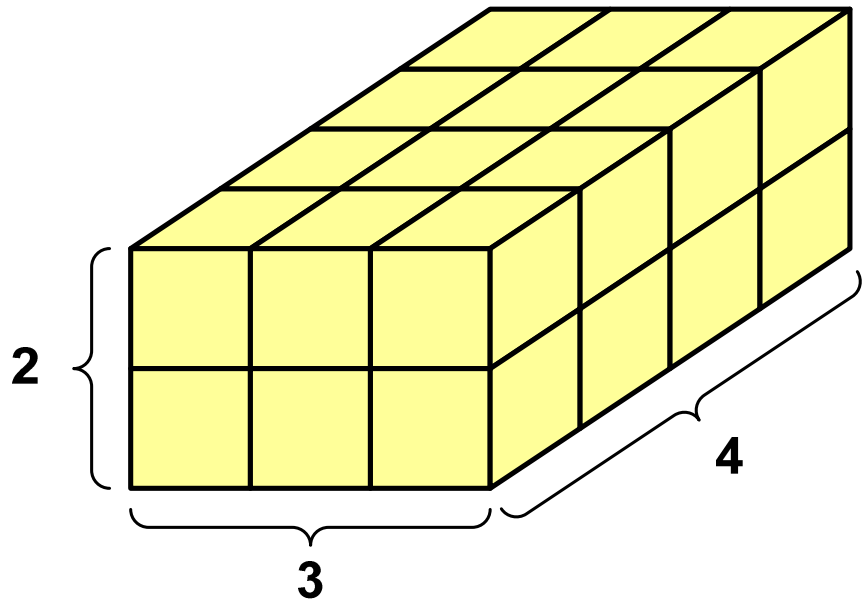
- Deklaracja tablicy wielowymiarowej

```
typ nazwa[wymiar_1][wymiar_2]…[wymiar_N]
```

- Deklaracja tablicy trójwymiarowej

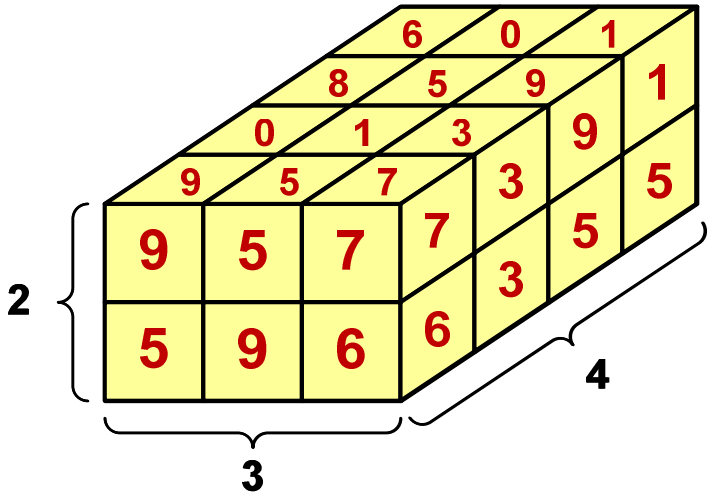
```
int tab[4][2][3];
```

- Inicjalizacja i odwoływanie się do elementów są analogiczne jak w przypadku macierzy



Język C - tablice wielowymiarowe

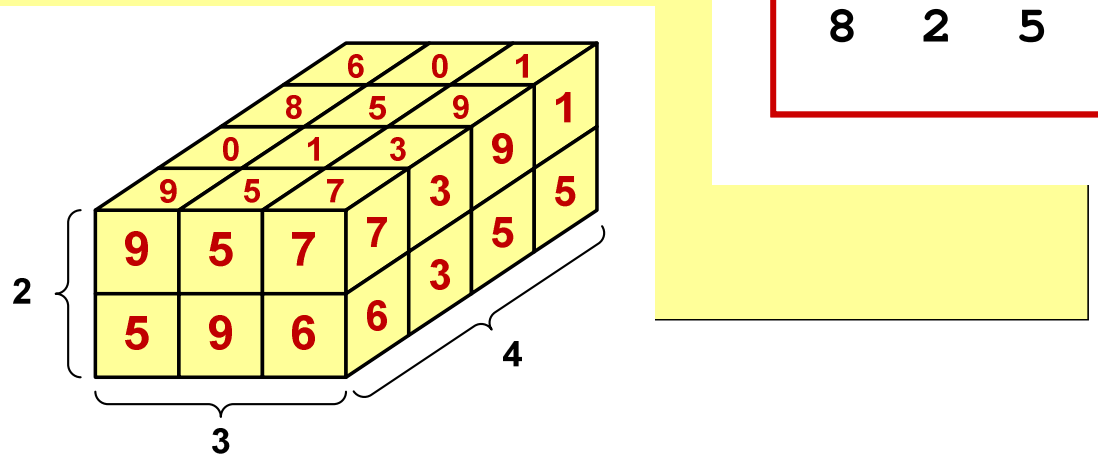
```
#include <stdio.h>

#define X 3
#define Y 2
#define Z 4

int main(void)
{
    int x, y, z;
    int tab[Z][Y][X] = {{{9,5,7},{5,9,6}},
                        {{0,1,3},{7,4,3}},
                        {{8,5,9},{1,3,5}},
                        {{6,0,1},{8,2,5}}};
```



Język C - tablice wielowymiarowe

```
    for(z=0; z<Z; z++)
    {
        for(y=0; y<Y; y++)
        {
            for(x=0; x<X; x++)
                printf("%3d",tab[z][y][x]);
            printf("\n");
        }
        printf("\n");
    }

    return 0;
}
```

```
  9   5   7
  5   9   6

  0   1   3
  7   4   3

  8   5   9
  1   3   5

  6   0   1
  8   2   5
```



Język C - tablice o zmiennym rozmiarze (VLA)

- VLA (ang. variable length array) - tablice, których rozmiar określany jest na etapie wykonywania programu (np. jako rozmiar może wystąpić nazwa zmiennej)

```
int n;
n = 10;
int T[n];
```

```
int n;
scanf("%d",&n);
int T[n];
```

- Rozmiar tablicy, a standardy języka C:
  - do standardu C99 rozmiar tablicy musiał być stałym wyrażeniem całkowitym (stała liczbowa: 5, #define N 5, const int n = 5;)
  - w standardzie C99 wprowadzono tablice o zmiennym rozmiarze
  - w standardzie C11 tablice o zmiennym rozmiarze określane są jako opcjonalne dla implementacji

Język C - tablice VLA (Visual Studio 2008 / 2019)

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar wektora: ");
    scanf("%d",&n);

    float T[n];

    for (i=0; i<n; i++)
        T[i] = sqrt((float)i);

    for (i=0; i<n; i++)
        printf("T[%d] = %f\n",i,T[i]);

    return 0;
}
```

Język C - tablice VLA (Visual Studio 2008 / 2019)

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar we
    scanf("%d",&n);

    float T[n];

    for (i=0; i<n; i
        T[i] = sqrt((fl

    for (i=0; i<n;
        printf("T[

    return 0;
}
```

error C2057: expected constant expression
error C2466: cannot allocate an array of constant size 0
error C2133: 'T' : unknown size

error C2057: oczekiwano stałego wyrażenia
error C2466: nie można przydzielić tablicy stałego rozmiaru 0
error C2133: "T": nieznany rozmiar

Język C - tablice VLA (Dev-C++, Code::Blocks)

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar wektora: ");
    scanf("%d",&n);

    float T[n];

    for (i=0; i<n; i++)
        T[i] = sqrt((float)i);

    for (i=0; i<n; i++)
        printf("T[%d] = %f\n",i,T[i]);

    return 0;
}
```

```
Rozmiar wektora: 8
T[0] = 0.000000
T[1] = 1.000000
T[2] = 1.414214
T[3] = 1.732051
T[4] = 2.000000
T[5] = 2.236068
T[6] = 2.449490
T[7] = 2.645751
```

Język C - tablice VLA

- Tablica VLA może być także tablicą dwu- lub wielowymiarową

```
int n = 5, m = 6;
int T1[n][m], T2[n][m][n];
```

- Nie można modyfikować rozmiaru tablic VLA po deklaracji
- Tablice VLA nie mogą być inicjalizowane podczas deklaracji
  - błędy i ostrzeżenia w Code::Blocks

```
error: variable-sized object may not be initialized
warning: excess elements in array initializer
warning: (near initialization for 'T')
```

  - w Dev-C++ inicjalizacja jest dopuszczalna!

Koniec wykładu nr 6

Dziękuję za uwagę!