UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU
Wydział Fizyki, Astronomii
i Informatyki Stosowanej

Prototypowanie Urządzeń Elektronicznych z Wykorzystaniem Platformy Arduino

Podstawy Arduino i języka C/C++

dr inż. Rafał Szczepański
www.umk.pl/~szczepi
szczepi@umk.pl

24.10.2025

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Plan prezentacji

- Struktura programu Arduino
- Typy zmiennych
- Operatory, warunki i pętle
- Piny i podstawowe funkcje Arduino
- Moduły EduBox
- Przykłady
- Zadania

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

- setup() — inicjalizacja: piny, komunikacja, zmienne, czujniki.
- loop() — kod główny, który działa cały czas (odświeża stany, steruje, odczytuje czujniki).

```
void setup() {
  // Kod wykonywany raz po uruchomieniu
}

void loop() {
  // Kod wykonywany w pętli
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Funkcje obsługi pinów cyfrowych

Funkcja	Opis	Przykład
`pinMode(pin, tryb)`	Ustawia tryb pracy pinu: INPUT, OUTPUT, INPUT_PULLUP.	`pinMode(13, OUTPUT);`
`digitalWrite(pin, wartość)`	Ustawia stan pinu cyfrowego: HIGH (1) lub LOW (0).	`digitalWrite(13, HIGH);`
`digitalRead(pin)`	Odczytuje stan logiczny pinu: HIGH lub LOW.	`int stan = digitalRead(2);`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Funkcje wejść/wyjść analogowych

Funkcja	Opis	Przykład
`analogRead(pin)`	Odczytuje napięcie (0–1023).	`int val = analogRead(A0);`
`analogWrite(pin, wartość)`	Ustawia PWM (0–255), działa tylko na pinach PWM.	`analogWrite(9, 128);`
`map(x, in_min, in_max, out_min, out_max)`	Przekształca zakres wartości.	`int jasnosc = map(val, 0, 1023, 0, 255);`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Funkcje czasu

Funkcja	Opis	Przykład
`delay(ms)`	Wstrzymuje program na określoną liczbę milisekund (blokująco).	`delay(500);`
`delayMicroseconds(us)`	Wstrzymuje na mikrosekundy.	`delayMicroseconds(100);`
`millis()`	Zwraca czas (ms) od startu programu.	`if (millis() - t > 1000)`
`micros()`	Zwraca czas (µs) od startu programu.	`unsigned long t = micros();`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Funkcje matematyczne

Funkcja	Opis	Przykład
`min(a, b)`	Zwraca mniejszą wartość.	`int x = min(10, 5); // x = 5`
`max(a, b)`	Zwraca większą wartość.	`int x = max(10, 5); // x = 10`
`abs(x)`	Wartość bezwzględna.	`int x = abs(-7); // x = 7`
`constrain(x, a, b)`	Ogranicza wartość x do zakresu [a, b].	`val = constrain(val, 0, 255);`
`pow(x, y)`	Potęgowanie.	`pow(2, 3); // 8`
`sqrt(x)`	Pierwiastek kwadratowy.	`sqrt(9); // 3`
`round(x), ceil(x), floor(x)`	Zaokrąglanie (dla float).	`round(2.7); // 3`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Funkcje komunikacji szeregowej (Serial)

Funkcja	Opis	Przykład
`Serial.begin(baud)`	Uruchamia komunikację, np. 9600 bps.	`Serial.begin(9600);`
`Serial.print(dane)`	Wysyła dane bez nowej linii.	`Serial.print("Temp: ");`
`Serial.println(dane)`	Wysyła dane + nowa linia.	`Serial.println(25);`
`Serial.available()`	Liczba bajtów gotowych do odczytu.	`if (Serial.available())`
`Serial.read()`	Odczytuje 1 bajt.	`char c = Serial.read();`
`Serial.write(dane)`	Wysyła bajt (binarnie).	`Serial.write(65); // 'A'`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Struktura programu Arduino

Makra i stałe systemowe

Nazwa	Znaczenie
`HIGH, LOW`	stany logiczne 1 / 0
`INPUT, OUTPUT, INPUT_PULLUP`	tryby pinów
`LED_BUILTIN`	wbudowana dioda LED (zwykle pin 13)
`true, false`	wartości logiczne
`PI, EULER, DEG_TO_RAD, RAD_TO_DEG`	stałe matematyczne

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Typy zmiennych

Podstawowe

Typ	Zakres	Przykład
`int`	-32768 do 32767	`int liczba = 5;`
`float`	liczby zmiennoprzecinkowe	`float temp = 23.5;`
`bool`	true / false	`bool stan = true;`

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Typy zmiennych

Biblioteka <stdint.h>

Typ	Rozmiar	Zakres
`int8_t`	8 bitów	−128 do +127
`int16_t`	16 bitów	−32 768 do +32 767
`int32_t`	32 bity	−2 147 483 648 do +2 147 483 647
`uint8_t`	8 bitów	0 do 255
`uint16_t`	16 bitów	0 do 65 535
`uint32_t`	32 bity	0 do 4 294 967 295

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Operatory arytmetyczne

+ dodawanie, **-** odejmowanie, ***** mnożenie, **/** dzielenie, **%** modulo (reszta z dzielenia)

- Kolejność działań: ***, /, %** mają wyższy priorytet niż **+** i **-**. Używaj nawiasów **(...)** dla czytelności i pewności.

- Jeśli operujesz na liczbach całkowitych (int, int8_t, uint16_t itp.), to **/** wykonuje dzielenie całkowite (część ułamkowa jest odrzucana):

```
int a = 7 / 2;  // a = 3
```

- Jeżeli operujesz na liczbach zmiennoprzecinkowych (float):

```
float b = 7.0 / 2.0;  // b = 3.5
```

- **%** działa tylko na typach całkowitych. Dla **a % b** otrzymujesz resztę z dzielenia a przez b:

```
int r = 7 % 3;  // r = 1
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Operatory arytmetyczne

Pułapki i wskazówki

- **Dzielenie przez 0**: błąd/runtime (*undefined behavior* dla integerów). Zawsze sprawdzaj dzielnik.

- Przy **mieszaniu typów** (np. int i float) następuje konwersja — zwykle do typu „większego” (tzw. promocja).

- Dla typów ze znakiem **wynik % ma taki znak jaki ma dzielna** (w C99 reszta ma ten sam znak co dzielna). Zachowanie z ujemnymi operandami warto znać.

- **Przepełnienie** (ang. *overflow*) przy mnożeniu lub dodawaniu: jeśli suma/mnożenie przekracza zakres typu — wynik „zawija” — lepiej użyć większego typu.

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Operatory logiczne i bitowe

Logiczne: **&&** (AND), **||** (OR), **!** (NOT)

Bitowe: **&**, **|**, **^** (XOR), **~** (negacja bitowa), **<<** lub **>>** (przesunięcie)

```
if (a > 0 && b != 0) { ... }  // b nie jest sprawdzane, jeśli a>0 jest false
                              (tzw. short-circuit)
```

```
uint8_t x = 0b00001111;
x = x << 2;  // 0b00111100
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Instrukcja warunkowa:

if (warunek) { ... } else { ... }

```
if (warunek) {
  // wykona się jeśli warunek jest prawdziwy (non-zero)
} else {
  // wykona się w przeciwnym razie
}
```

UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU
Wydział Fizyki, Astronomii
i Informatyki Stosowanej

Instrukcja warunkowa:

if (warunek) { ... } else { ... }

```
int temp = analogRead(A0);

if (temp > 512) {
  digitalWrite(ledPin, HIGH);
} else {
  digitalWrite(ledPin, LOW);
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Instrukcja warunkowa:

Łańcuch warunków

```
if (x < 0) {
  // ...
} else if (x == 0) {
  // ...
} else {
  // x > 0
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Instrukcja warunkowa:

Operator warunkowy — skrócona forma:

```
(warunek) ? {co jeżeli prawda} : {co jeżeli fałsz}
```

```
digitalWrite(ledPin, (buttonState == HIGH) ? HIGH : LOW);
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Operatory arytmetyczne

Pułapki i wskazówki

• Warunek to wyrażenie całkowite: **if (a = 5)** - błąd logiczny: przypisanie zamiast porównania; w C/C++ ta konstrukcja jest dozwolona i zwróci 5 (czyli true).

• Dla bezpieczeństwa, przy porównaniach stałych można pisać **if (5 == a)** — wtedy kompilator złapie = jako błąd (5 = a jest nieprawidłowe).

•Używaj nawiasów, aby uniknąć niejasności przy złożonych warunkach: **if ((a > 0 && b < 10) || c == 3)**

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Pętle

for

```
for (inicjalizacja; warunek; iteracja) {
  // ciało pętli
}
```

Przykład:

```
for (int i = 0; i < 5; i++) {
  Serial.println(i);
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Pętle

while

```
while (warunek) {
  // wykona się dopóki warunek jest prawdziwy
}
```

Przykład:

```
int x = 0;
while (x < 10) {
  Serial.println(x);
  x++;
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Pętle

do-while

```
do {
  // wykona się dopóki warunek jest prawdziwy
} while (warunek);
```

Przykład:

```
int x = 0;
do {
  Serial.println(x);
  x++;
} while (x < 10);
```

UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU
Wydział Fizyki, Astronomii
i Informatyki Stosowanej

Cykliczne wywoływanie funkcjonalności

funkcja delay()

```
const uint32 interval = 500;

void loop() {
  // wykonaj akcję (np. przełącz LED)

  delay(interval);
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Cykliczne wywoływanie funkcjonalności

funkcja `millis()`

```
uint32_t previous = 0;
const uint32 interval = 500;

void loop() {
 uint32 now = millis();
  if (now - previous >= interval) {
    previous = now;
    // wykonaj akcję (np. przełącz LED)
  }
  // inne rzeczy można tu wykonywać bez blokowania
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Własne funkcje

```
typ_zwracany nazwa_funkcji (parametry) {
  // ciało funkcji – kod, który się wykona
  return wartość; // opcjonalnie (tylko jeśli typ ≠ void)
}
```

Przykład bez zwracania i pobierania wartości:

```
void MrugnijDioda(void) {
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Własne funkcje

```
typ_zwracany nazwa_funkcji (parametry) {
  // ciało funkcji – kod, który się wykona
  return wartość; // opcjonalnie (tylko jeśli typ ≠ void)
}
```

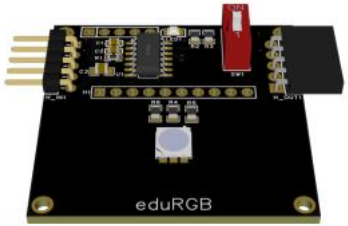
Przykład ze zwracaniem i pobieraniem wartości:

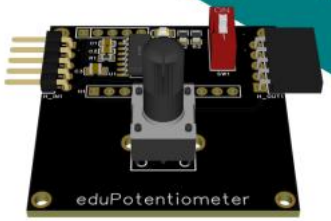
```
int dodaj(int a, int b) {
  int wynik = a + b;
  return wynik;  // zwracamy wartość
}
```

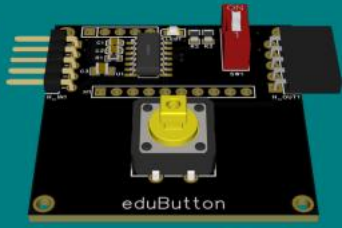
eduBox

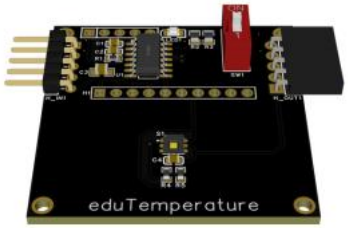
Pakiet podstawowy

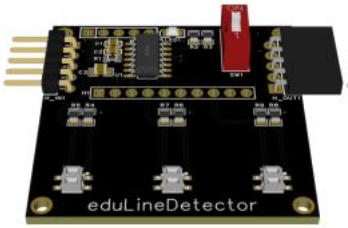
1. **eduRGB** - dioda LED
2. **eduPotentiometer** - potencjometr
3. **eduBuzzer** - brzęczyk
4. **eduButton** - przycisk
5. **eduTemperature** - czujnik temperatury i wilgotności
6. **eduLineDetector** - czujniki odbiciowe (czujniki linii)
7. **edu7Segment** - wyświetlacz siedmiosegmentowy dwucyfrowy
8. **eduColorSensor** - czujnik koloru
9. **eduDistance** - ultradźwiękowy czujnik odległości
10. **eduServo** - serwomechanizm 0-180 stopnii
11. **eduStepperMotor** - silnik krokowy

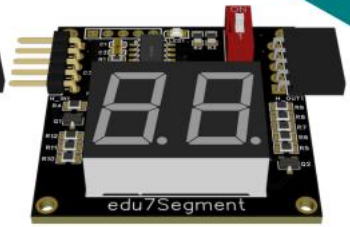


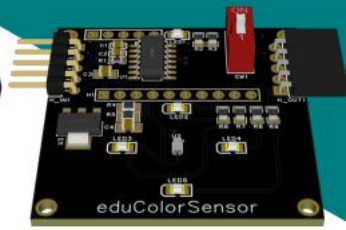


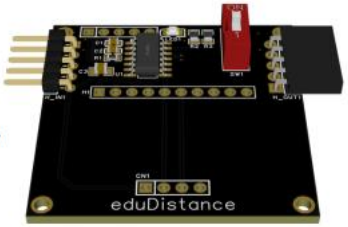


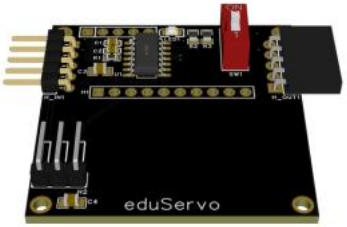


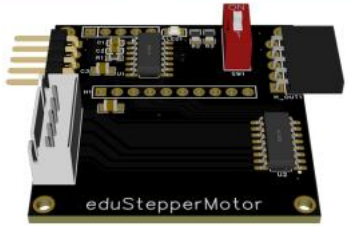








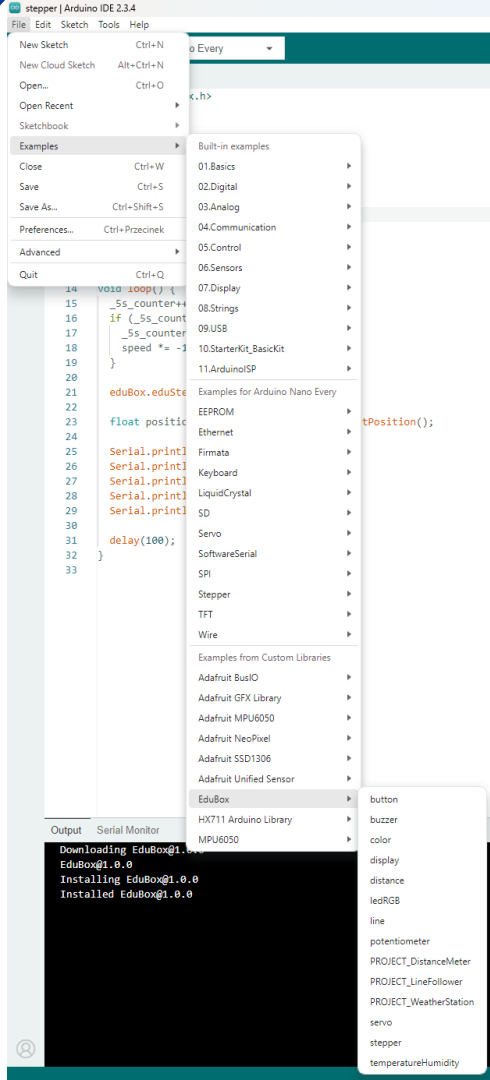




Moduły EduBox

eduRGB - Przykładowa aplikacja zmieniająca kolor diody LED

- Dioda sygnalizująca połączenie z modułem
- Możliwość przełączenia numeru modułu
- Gotowe biblioteki oraz przykłady



UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Moduły EduBox

eduRGB - Przykładowa aplikacja zmieniająca kolor diody LED

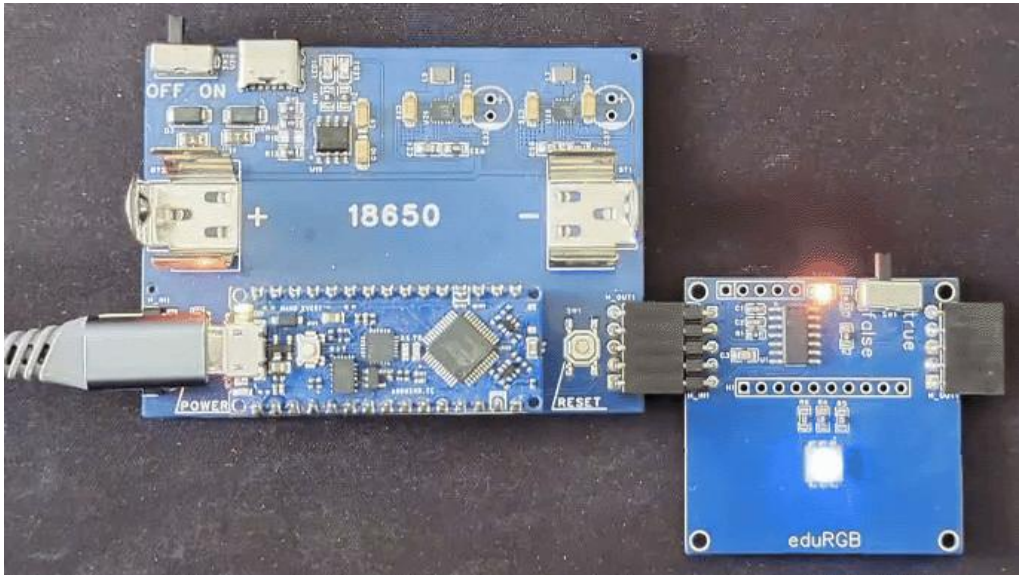
ledRGB | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Arduino Nano Every

ledRGB.ino

```
1   #include <EduBox.h>
2
3   EduBox eduBox;
4
5   void setup() {
6     eduBox.begin();
7   }
8
9   void loop() {
10    eduBox.eduRGB_SetValues(25, 0, 0);
11    delay(1000);
12    eduBox.eduRGB_SetValues(0, 25, 0);
13    delay(1000);
14    eduBox.eduRGB_SetValues(0, 0, 25);
15    delay(1000);
16  }
```



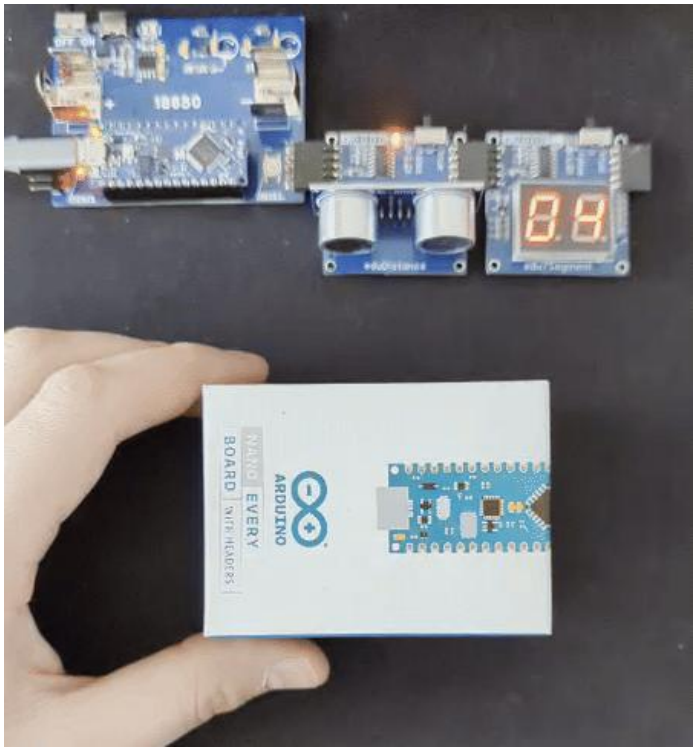
UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Moduły EduBox

eduDistance + edu7Segment - Przykładowa aplikacja mierząca odległość

- Moduły połączone szeregowo – kolejność jest bez znaczenia!

```
#include <EduBox.h>

EduBox eduBox;

void setup() {
  eduBox.begin();
}

void loop() {
  float distanceCentimeters = eduBox.eduDistance_GetDistanceCentimeters();

  eduBox.edu7Segment_DisplayNumber(distanceCentimeters);

  delay(250);
}
```



UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Moduły EduBox

eduTemperature + 2x edu7Segment - Przykładowa aplikacja stacji pogodowej

- Istotne jest przełączenie w jednym z wyświetlaczy przełącznika na ***true***!

```
#include <EduBox.h>

EduBox eduBox;

void setup() {
  eduBox.begin();
}

void loop() {
  float distanceCentimeters = eduBox.eduDistance_GetDistanceCentimeters();

  eduBox.edu7Segment_DisplayNumber(distanceCentimeters);

  delay(250);
}
```