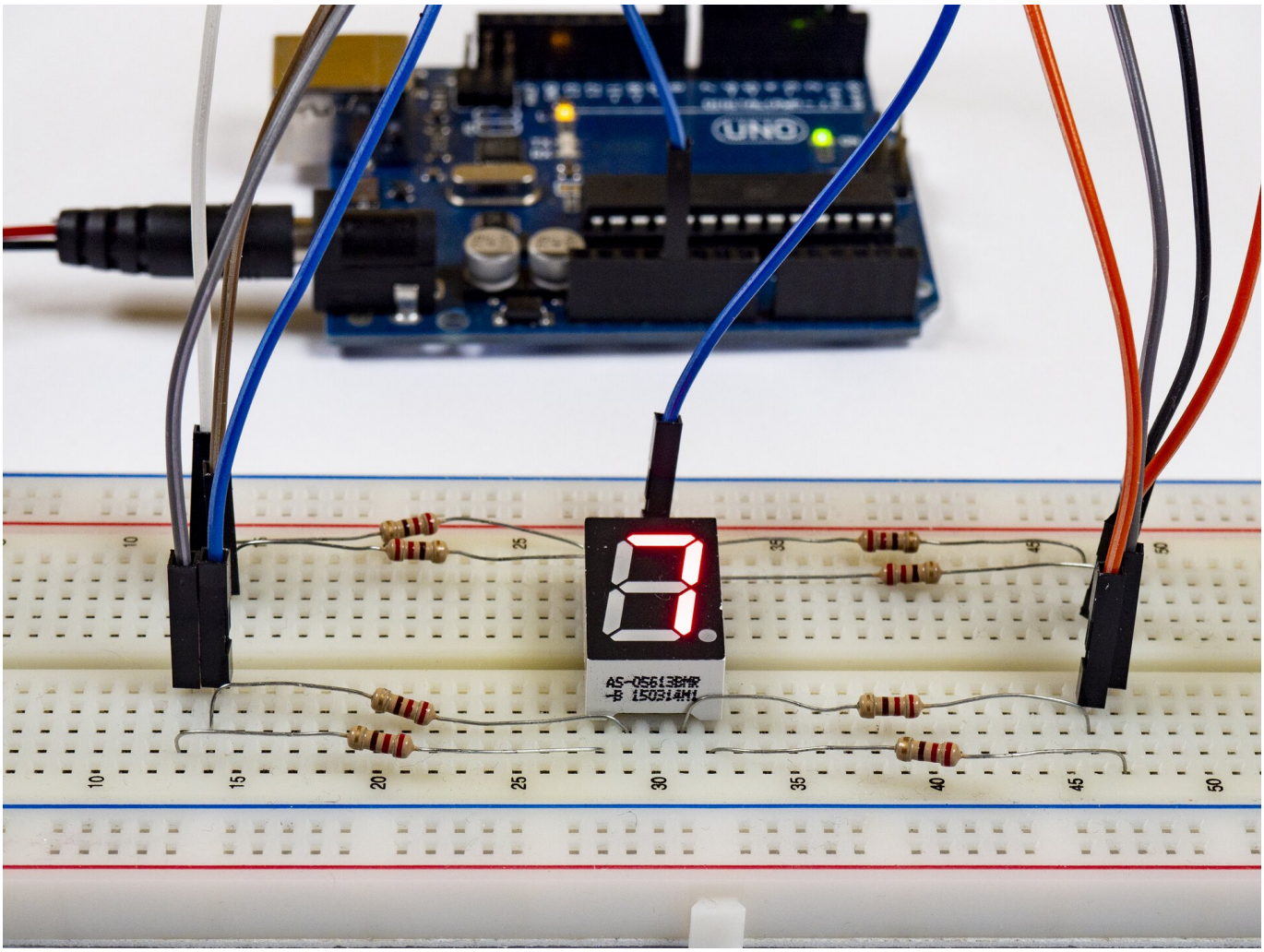
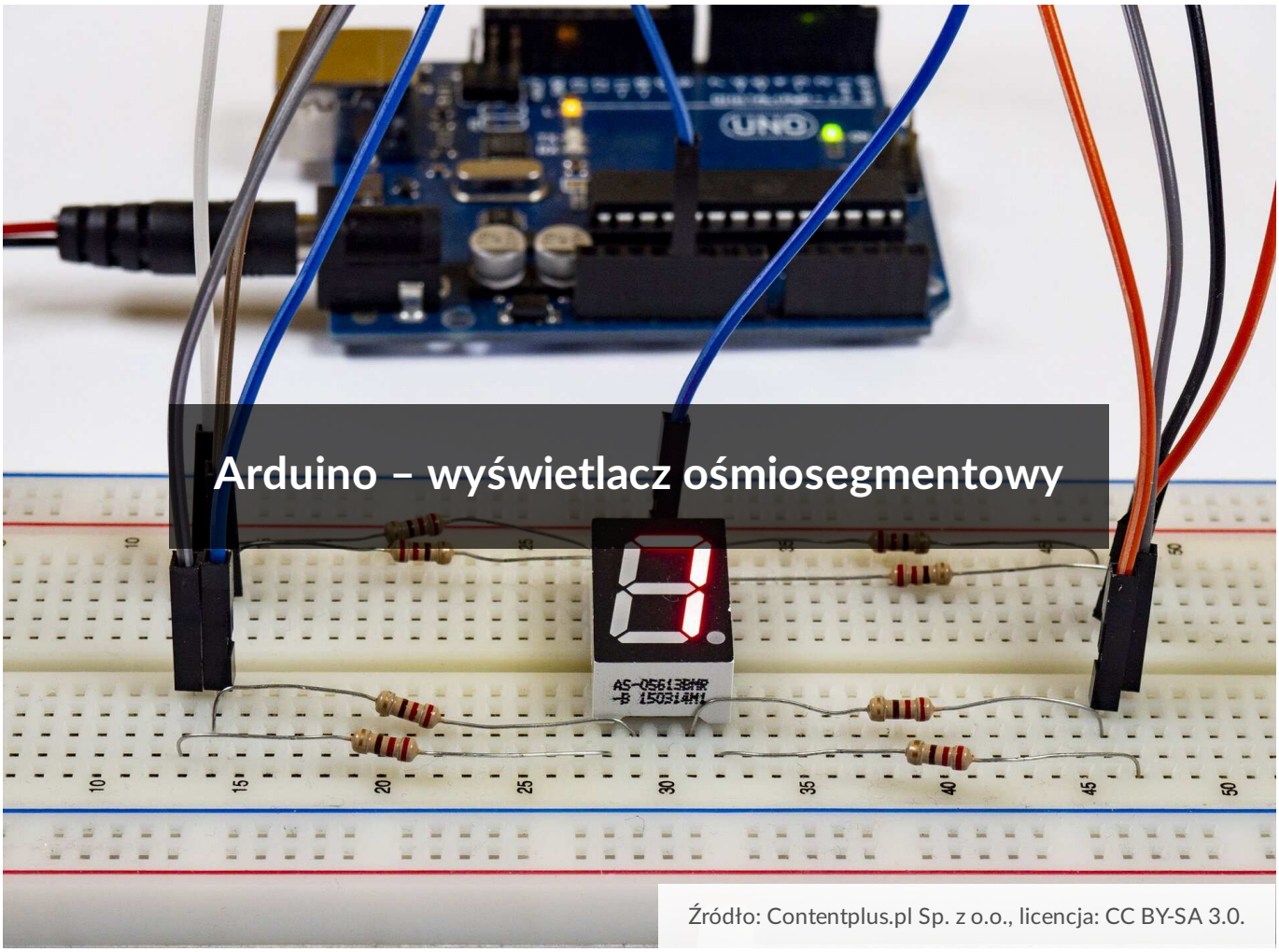



# Arduino – wyświetlacz ośmiosegmentowy




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Przy budowie układów elektronicznych często zachodzi potrzeba wyświetlenia jakiejś informacji zwrotnej. Zazwyczaj stosujemy do tego celu wyświetlacze LCD, na przykład 2x16 (dwa wiersze po szesnaście znaków). Nie zawsze jednak musimy wyświetlać dużą ilość informacji – czasem wystarczą dosłownie jedna czy dwie cyfry. Albo cztery – jak choćby w prostym zegarku elektronicznym. Do tego celu znakomicie nadają się wyświetlacze 8-segmentowe.

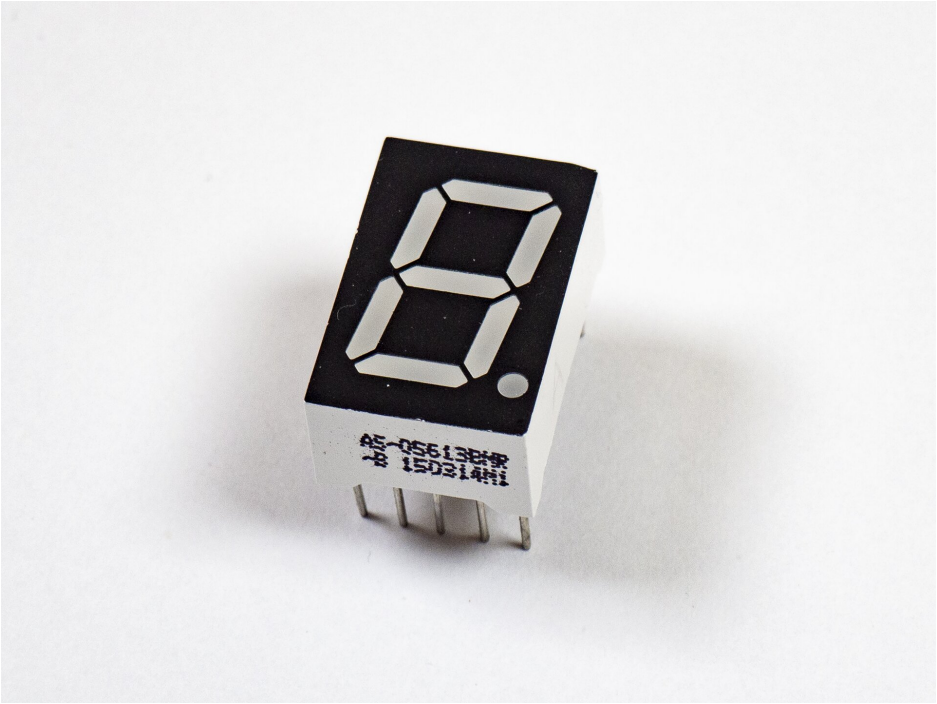
**Twoje cele**

- Poznasz budowę 8-segmentowego wyświetlacza LED.
- Dowiesz się, jak podłączyć wyświetlacz 8-segmentowy do płytki Arduino.
- Zapoznasz się z możliwościami generowania liczb prawdziwie losowych.

# Przeczytaj

## Wyświetlacz 8-segmentowy

Wyświetlacz 8-segmentowy (czy też 7-segmentowy, bez znaku kropki) to tak naprawdę ułożone w specyficzny sposób diody LED. Do wyświetlania danej cyfry służą odpowiednio włączane segmenty (czyli właśnie diody) wyświetlacza. Na zdjęciach poniżej przedstawione są dwa przykładowe wyświetlacze 8-segmentowe – jednocyfrowy oraz czterocyfrowy.



Wyświetlacz jednocyfrowy

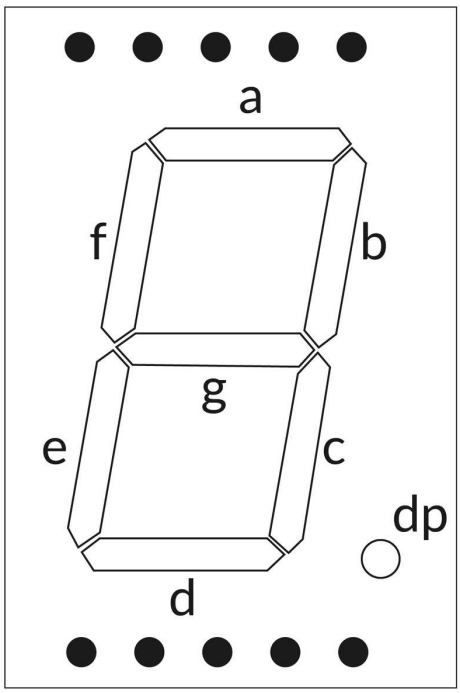
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.


Wyświetlacz czterocyfrowy

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

## Budowa wyświetlacza

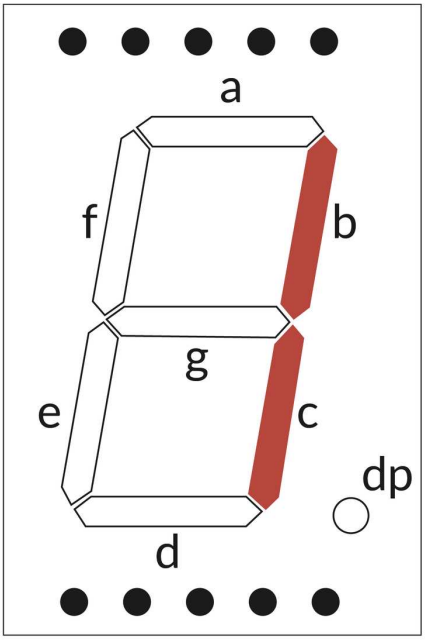
Aby można było łatwo zidentyfikować poszczególne segmenty tworzące cyfrę, są one opisane literami: a, b, c, d, e, f, g, natomiast kropka – p lub dp.



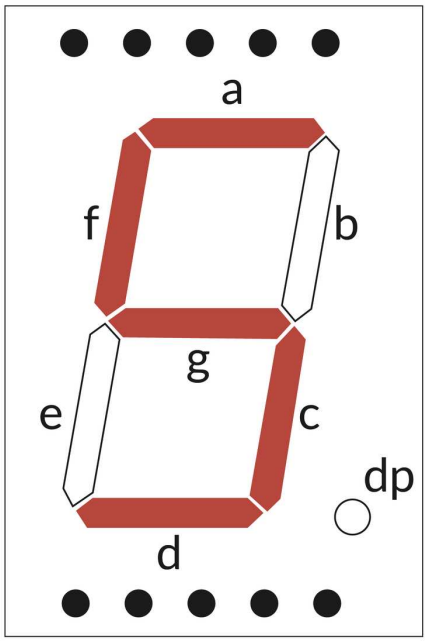

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Przykładowo, jeżeli chcemy wyświetlić cyfrę 1, włączamy segmenty b i c. Dla cyfry 5 włączamy segmenty a, c, d, f, g.

wyświetlenie cyfry 1



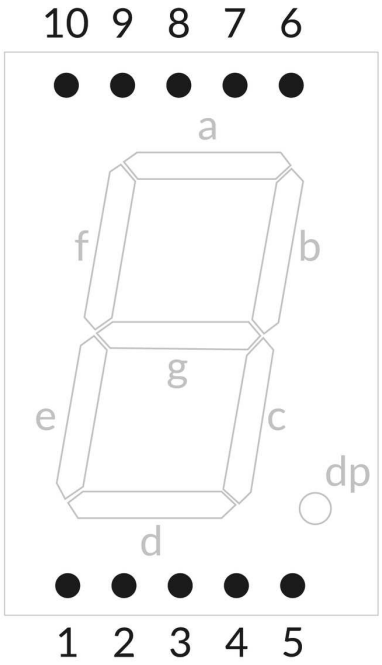

wyświetlenie cyfry 5




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W przypadku wyświetlaczy jednocyfrowych mamy do czynienia z dziesięcioma wyprowadzeniami (spodnia część wyświetlacza). Zazwyczaj pin numer 1 znajduje się z lewej

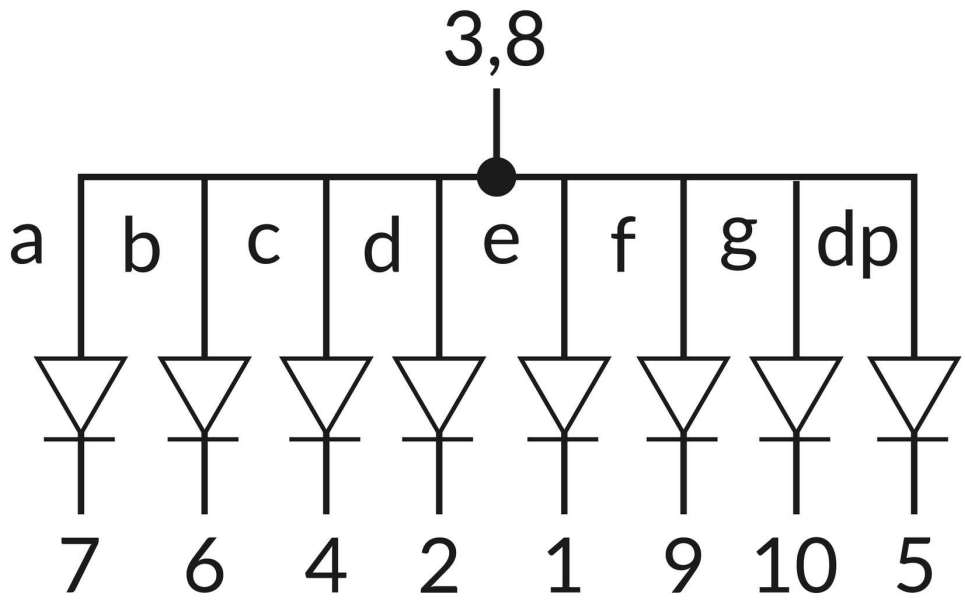
strony u dołu (patrząc od przodu wyświetlacza), a ostatni pin numer 10 umieszczony jest u góry, również po lewej stronie.




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Aby ograniczyć liczbę pinów, wyświetlacze 8-segmentowe zbudowane są podobnie jak diody RGB. Mówimy zatem o wyświetlaczu ze wspólną anodą lub wspólną katodą.
W zależności od tego, z jakiego typu wyświetlaczem mamy do czynienia, zastosujemy odpowiednie podłączenia.

Przedstawiony tutaj wyświetlacz 8-segmentowy to wyświetlacz ze wspólną anodą. Jeżeli nie mamy dostępu do karty katalogowej danego wyświetlacza, możemy przyjąć, że większość wyświetlaczy jednocyfrowych ze wspólną anodą będzie miała taki oto schemat wyprowadzeń:



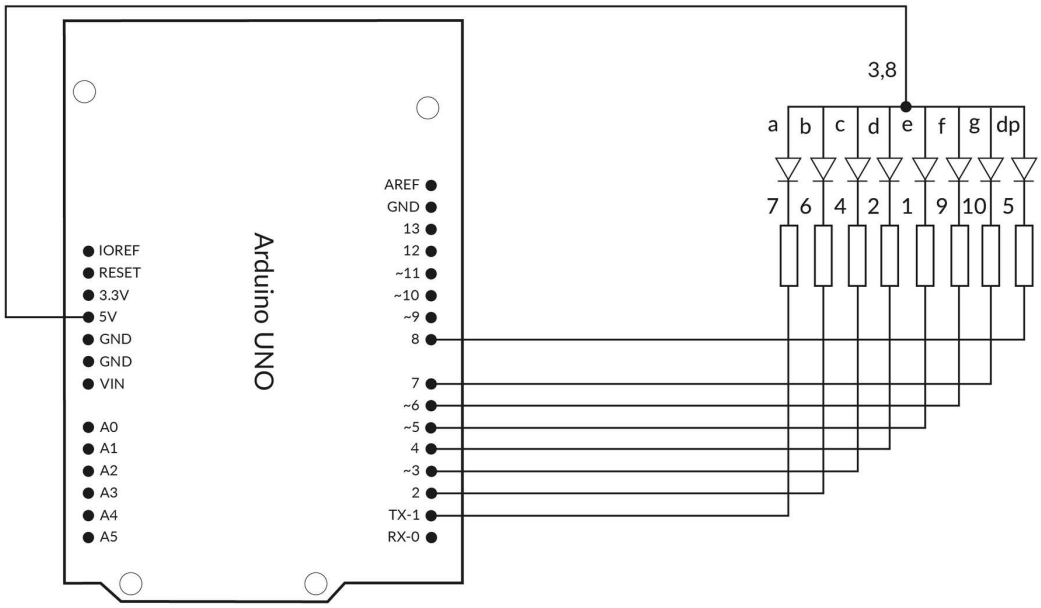
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Ze schematu możemy odczytać, że aby włączyć na przykład segment **a**, podłączamy plus zasilania do pinu numer 3 (anoda), zaś minus do pinu numer 7 (katoda).

## Podłączenie wyświetlacza do Arduino

Wiedząc, z którego typu wyświetlaczem mamy do czynienia (wspólna anoda lub katoda), możemy narysować schemat połączeń.

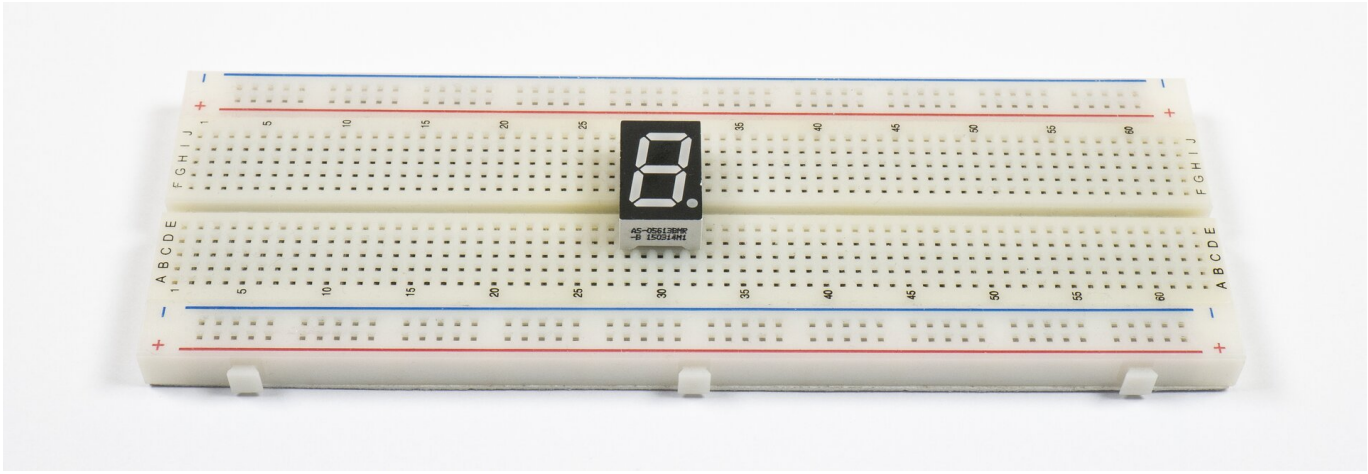
> **Ważne!**
>
> Pamiętaj, że diody LED w wyświetlaczu muszą być podłączone do Arduino poprzez rezystory ograniczające prąd. W przeciwnym przypadku dioda (segment) ulegnie zniszczeniu. Zastosowany tutaj rezystor ma wartość 220 Ω, ale można też użyć większych, na przykład 1 kΩ.




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

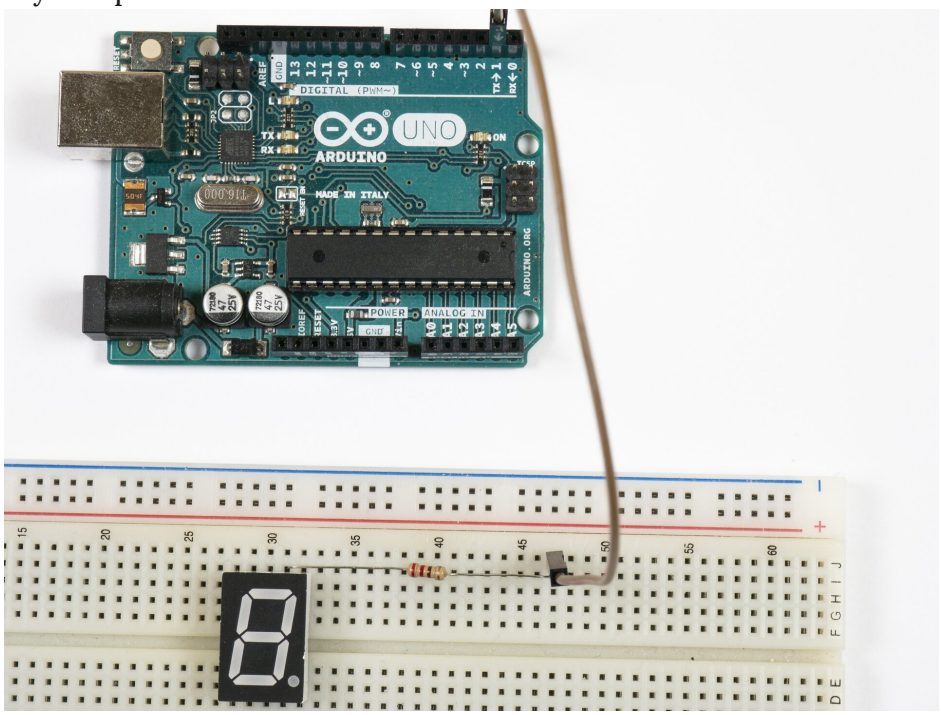
W celu podłączenia wyświetlacza do Arduino musimy wykonać następujące kroki:

1. Najpierw wpinamy wyświetlacz w płytkę stykową tak, aby piny górne i piny dolne były wpięte w oddzielne połówki płytki.



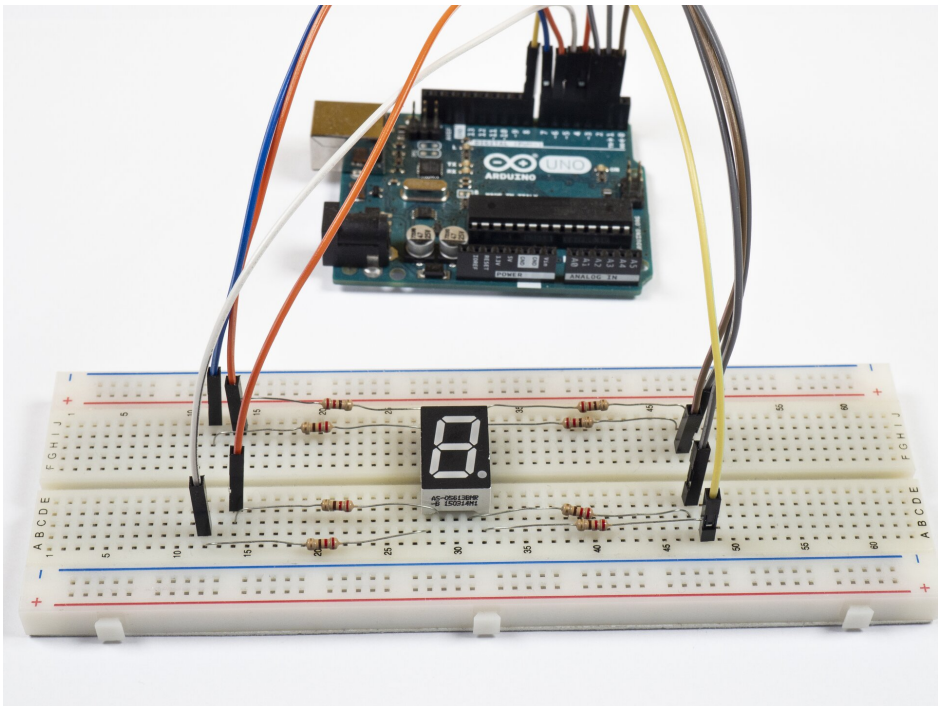
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

2. Teraz, zgodnie ze schematem, do pinu numer 7 wyświetlacza (segment a) podpinamy poprzez rezystor pin numer 1 w Arduino.



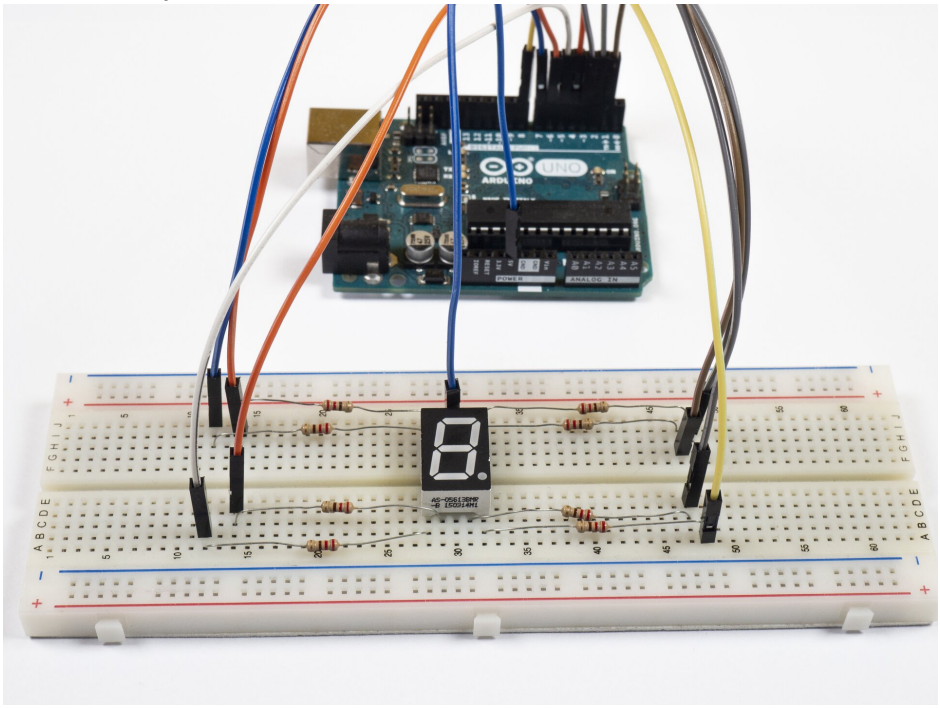

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

3. Analogicznie postępujemy z kolejnymi wyprowadzeniami:
    - segment b (pin 6) – rezystor – pin 2 Arduino,
    - segment c (pin 4) – rezystor – pin 3 Arduino,
    - segment d (pin 2) – rezystor – pin 4 Arduino,
    - segment e (pin 1) – rezystor – pin 5 Arduino,
    - segment f (pin 9) – rezystor – pin 6 Arduino,
    - segment g (pin 10) – rezystor – pin 7 Arduino,
    - segment dp (pin 5) – rezystor – pin 8 Arduino.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

4. Na koniec podpinamy anodę wyświetlacza do pinu 5 V w Arduino. Warto tutaj zaznaczyć, że zamiast podpinać na stałe zasilanie 5 V, możemy użyć dowolnego pinu cyfrowego do zasilenia wyświetlacza. Takie rozwiązanie ułatwi na przykład miganie daną cyfrą, gdyż będziemy sterować jednym pinem zasilającym, a nie każdym segmentem z osobna. Należy tutaj pamiętać o wydajności prądowej pinu cyfrowego i użyć na przykład rezystorów 1 kΩ, aby ograniczyć prąd w przypadku cyfr, które będą wyświetlane przy użyciu dużej ilości zapalonych segmentów. Jednak najlepszym rozwiązaniem będzie podłączenie anody poprzez tranzystor i zasilanie wyświetlacza z innego źródła napięcia.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

## Obsługa wyświetlacza w Arduino IDE

Teraz możemy przejść do zbudowania układu, który będzie bazą działania naszego wyświetlacza. W pierwszej kolejności otwieramy nowy szkic i przed funkcją `setup()` definiujemy nazwy pinów. Dzięki temu łatwiej nam będzie odnaleźć się w programie i sterować wyświetlaczem.

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8
```

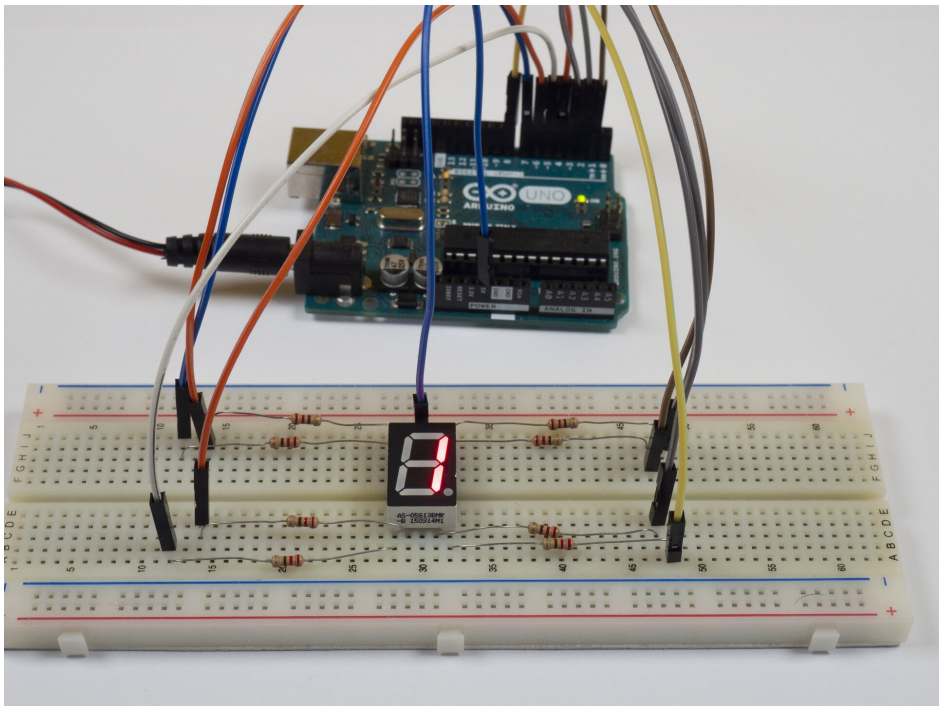
Pin numer 1 to pin o nazwie `segment_A`, pin numer 2 to pin o nazwie `segment_B` itd.

Następnie w funkcji `setup()` należy określić typ pinów, czyli zdecydować, czy będą to piny wejściowe, czy piny wyjściowe. Oczywiście ustawiamy je na piny wyjściowe.

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8

void setup() {
    pinMode(segment_A, OUTPUT);
    pinMode(segment_B, OUTPUT);
    pinMode(segment_C, OUTPUT);
    pinMode(segment_D, OUTPUT);
    pinMode(segment_E, OUTPUT);
    pinMode(segment_F, OUTPUT);
    pinMode(segment_G, OUTPUT);
    pinMode(segment_DP, OUTPUT);
}
```

Teraz, wiedząc, że segmenty wyświetlacza są po prostu diodami LED, możemy włączyć każdy z nich za pomocą funkcji `digitalWrite()`. Pamiętajmy, że korzystamy z wyświetlacza ze wspólną anodą, dlatego aby załączyć dany segment, musimy ustawić stan niski na pinie. Spróbujmy wyświetlić cyfrę 1. W pętli `loop` ustawiamy odpowiednie stany dla każdego segmentu. Aby wyświetlić cyfrę 1 musimy włączyć segmenty b i c, a resztę segmentów wyłączyć.

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8

void setup() {
    pinMode(segment_A, OUTPUT);
    pinMode(segment_B, OUTPUT);
    pinMode(segment_C, OUTPUT);
    pinMode(segment_D, OUTPUT);
    pinMode(segment_E, OUTPUT);
    pinMode(segment_F, OUTPUT);
    pinMode(segment_G, OUTPUT);
    pinMode(segment_DP, OUTPUT);
}

void loop() {
    digitalWrite(segment_A, HIGH);
    digitalWrite(segment_B, LOW);
    digitalWrite(segment_C, LOW);
    digitalWrite(segment_D, HIGH);
    digitalWrite(segment_E, HIGH);
    digitalWrite(segment_F, HIGH);
    digitalWrite(segment_G, HIGH);
    digitalWrite(segment_DP, HIGH);
}
```



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Analogicznie, chcąc wyświetlić na przykład cyfrę 5, włączamy odpowiednie segmenty:

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8

void setup() {
    pinMode(segment_A, OUTPUT);
    pinMode(segment_B, OUTPUT);
    pinMode(segment_C, OUTPUT);
    pinMode(segment_D, OUTPUT);
    pinMode(segment_E, OUTPUT);
    pinMode(segment_F, OUTPUT);
    pinMode(segment_G, OUTPUT);
    pinMode(segment_DP, OUTPUT);
}

void loop() {
    digitalWrite(segment_A, LOW);
    digitalWrite(segment_B, HIGH);
```

```
    digitalWrite(segment_C, LOW);
    digitalWrite(segment_D, LOW);
    digitalWrite(segment_E, HIGH);
    digitalWrite(segment_F, LOW);
    digitalWrite(segment_G, LOW);
    digitalWrite(segment_DP, HIGH);
}
```

## Prosty licznik

Wykorzystując zbudowany układ, stwórzmy prosty licznik odliczający cyfry od 0 do 9.

Początek szkicu będzie identyczny z powyższym:

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8

void setup() {
    pinMode(segment_A, OUTPUT);
    pinMode(segment_B, OUTPUT);
    pinMode(segment_C, OUTPUT);
    pinMode(segment_D, OUTPUT);
    pinMode(segment_E, OUTPUT);
    pinMode(segment_F, OUTPUT);
    pinMode(segment_G, OUTPUT);
    pinMode(segment_DP, OUTPUT);
}
```

Następnie użyjemy pętli `for` w formie licznika odliczającego liczby od 0 do 9 oraz funkcji `switch-case`, która wyświetli nam odpowiednie cyfry.

```
#define segment_A 1
#define segment_B 2
#define segment_C 3
```

```
#define segment_D 4
#define segment_E 5
#define segment_F 6
#define segment_G 7
#define segment_DP 8

void setup() {
    pinMode(segment_A, OUTPUT);
    pinMode(segment_B, OUTPUT);
    pinMode(segment_C, OUTPUT);
    pinMode(segment_D, OUTPUT);
    pinMode(segment_E, OUTPUT);
    pinMode(segment_F, OUTPUT);
    pinMode(segment_G, OUTPUT);
    pinMode(segment_DP, OUTPUT);
}

void loop() {
    int i = 0;

    for (i = 0; i < 10; i++) {
        switch (i) {
            case 0:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, LOW);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, HIGH);
                digitalWrite(segment_DP, HIGH);
                break;
            case 1:
                digitalWrite(segment_A, HIGH);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, HIGH);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, HIGH);
                digitalWrite(segment_G, HIGH);
                digitalWrite(segment_DP, HIGH);
                break;
```

```
            case 2:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, HIGH);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, LOW);
                digitalWrite(segment_F, HIGH);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 3:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, HIGH);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 4:
                digitalWrite(segment_A, HIGH);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, HIGH);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 5:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, HIGH);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 6:
                digitalWrite(segment_A, LOW);
```

```
                digitalWrite(segment_B, HIGH);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, LOW);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 7:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, HIGH);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, HIGH);
                digitalWrite(segment_G, HIGH);
                digitalWrite(segment_DP, HIGH);
                break;
            case 8:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, LOW);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
            case 9:
                digitalWrite(segment_A, LOW);
                digitalWrite(segment_B, LOW);
                digitalWrite(segment_C, LOW);
                digitalWrite(segment_D, LOW);
                digitalWrite(segment_E, HIGH);
                digitalWrite(segment_F, LOW);
                digitalWrite(segment_G, LOW);
                digitalWrite(segment_DP, HIGH);
                break;
        }

        delay(1000); // Poczekaj sekundę
    }
```

```
130 }
```

Po włączeniu układu zmienna `i` będzie miała wartość 0, a za sprawą instrukcji `switch-case` zostaną włączone odpowiednie segmenty, które utworzą na wyświetlaczu cyfrę 0. Następnie program odczeka sekundę, po czym zmienna `i` przyjmie wartość 1, a instrukcja `switch-case` zadba o włączenie odpowiednich segmentów, aby wyświetlić cyfrę 1.

Po wgraniu szkicu do płytki Arduino efekt działania licznika powinien wyglądać następująco:


Film dostępny pod adresem /preview/resource/R1SOPIrzoJcS2

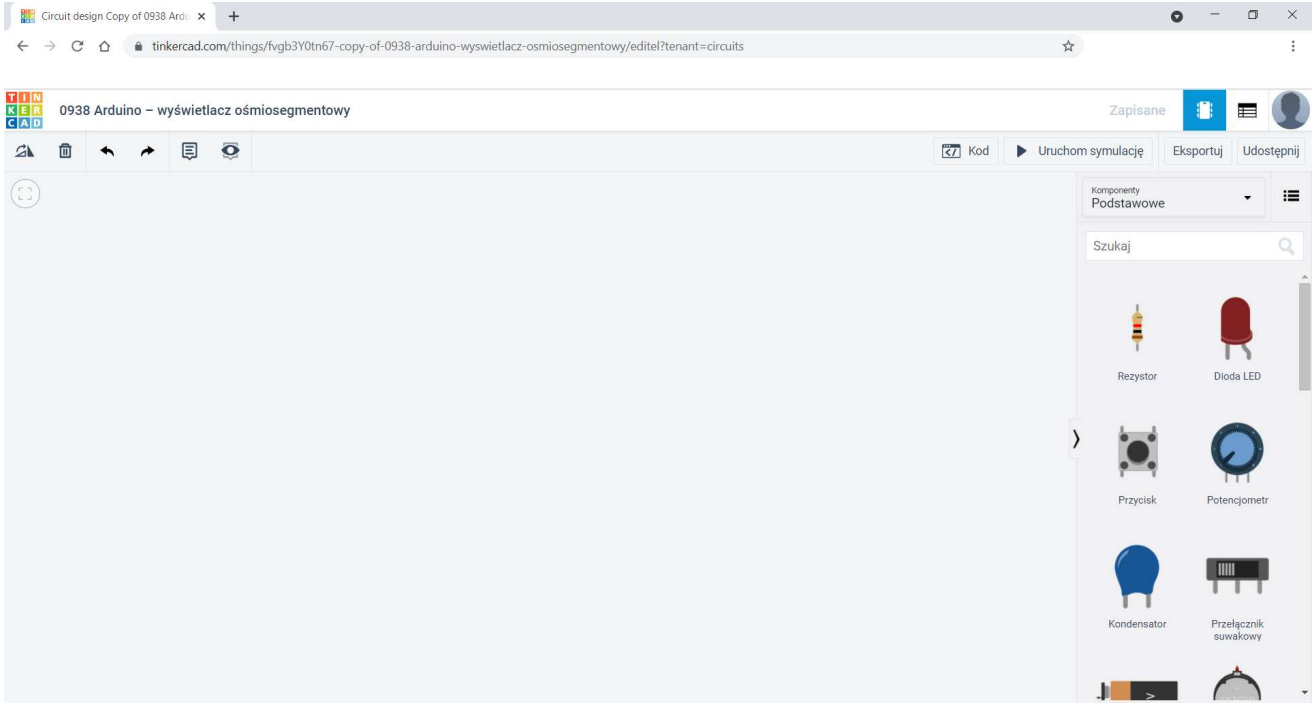


Film przedstawiający licznik wyświetlający wartości od 0 do 9

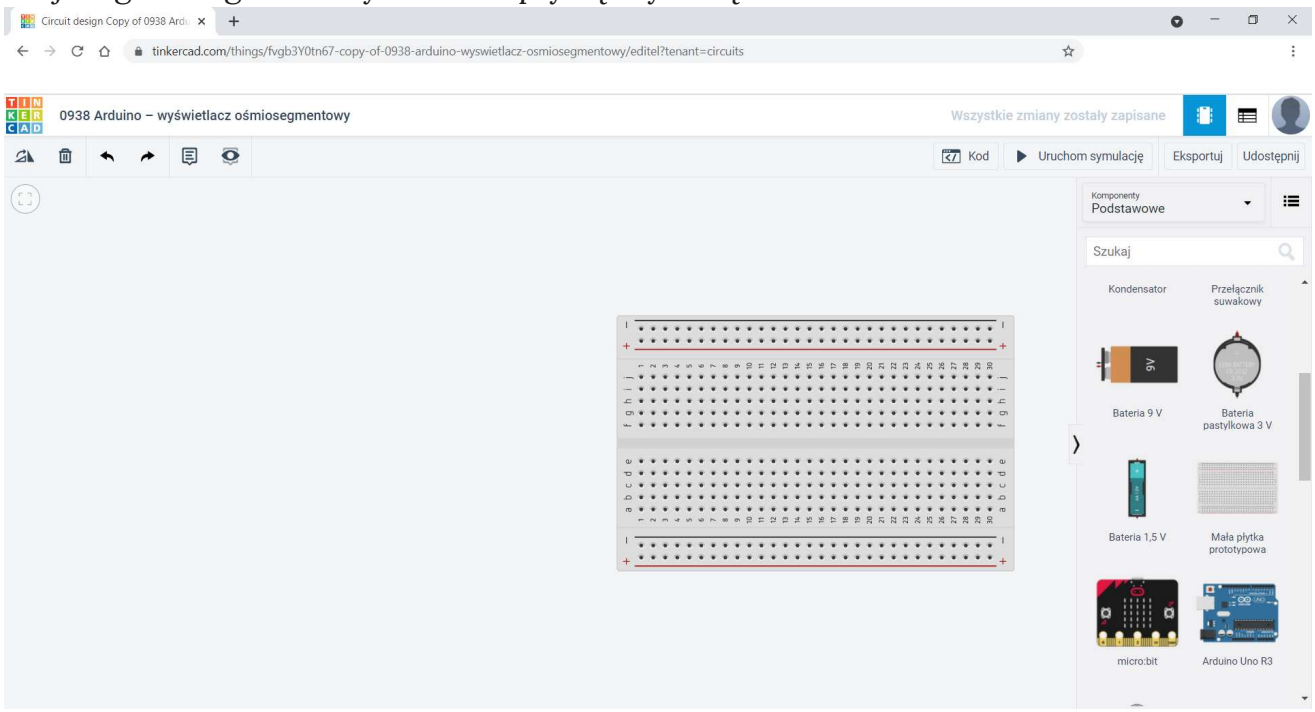
---

## Tinkercad

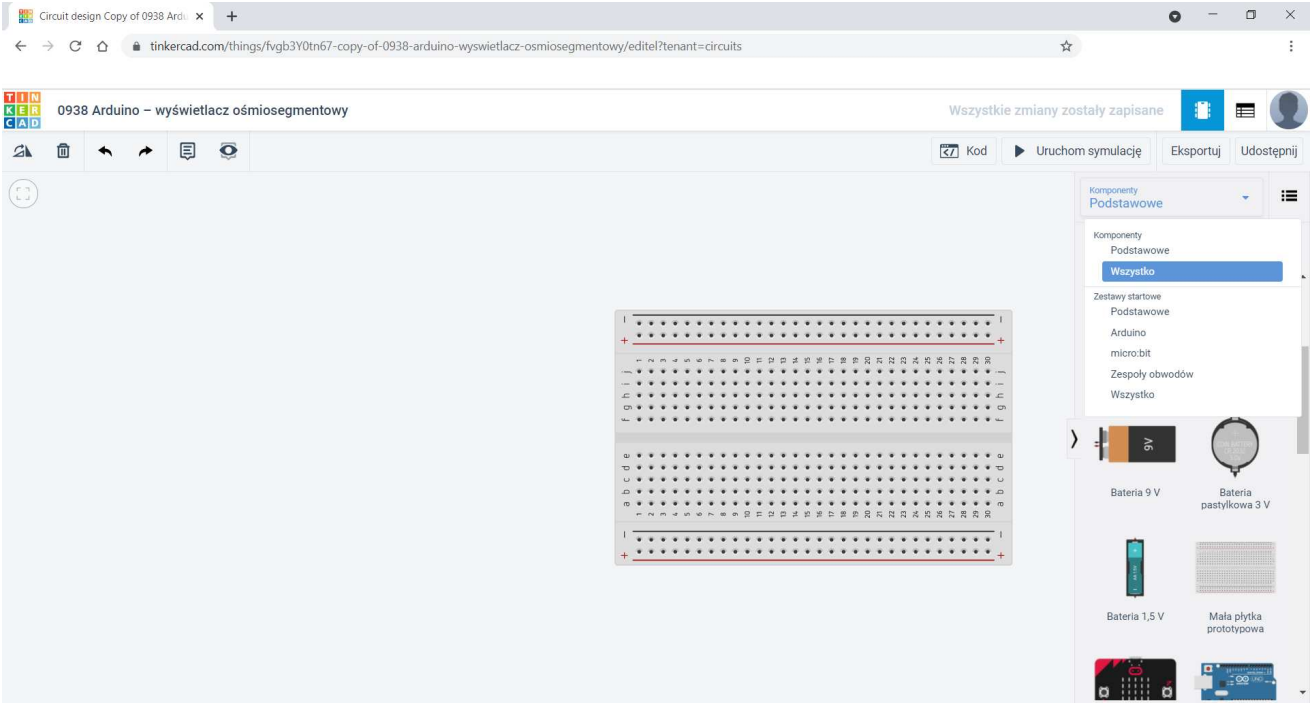
Zróbmy symulację układu z wyświetlaczem ośmiosegmentowym, za pomocą środowiska `Tinkercad`.



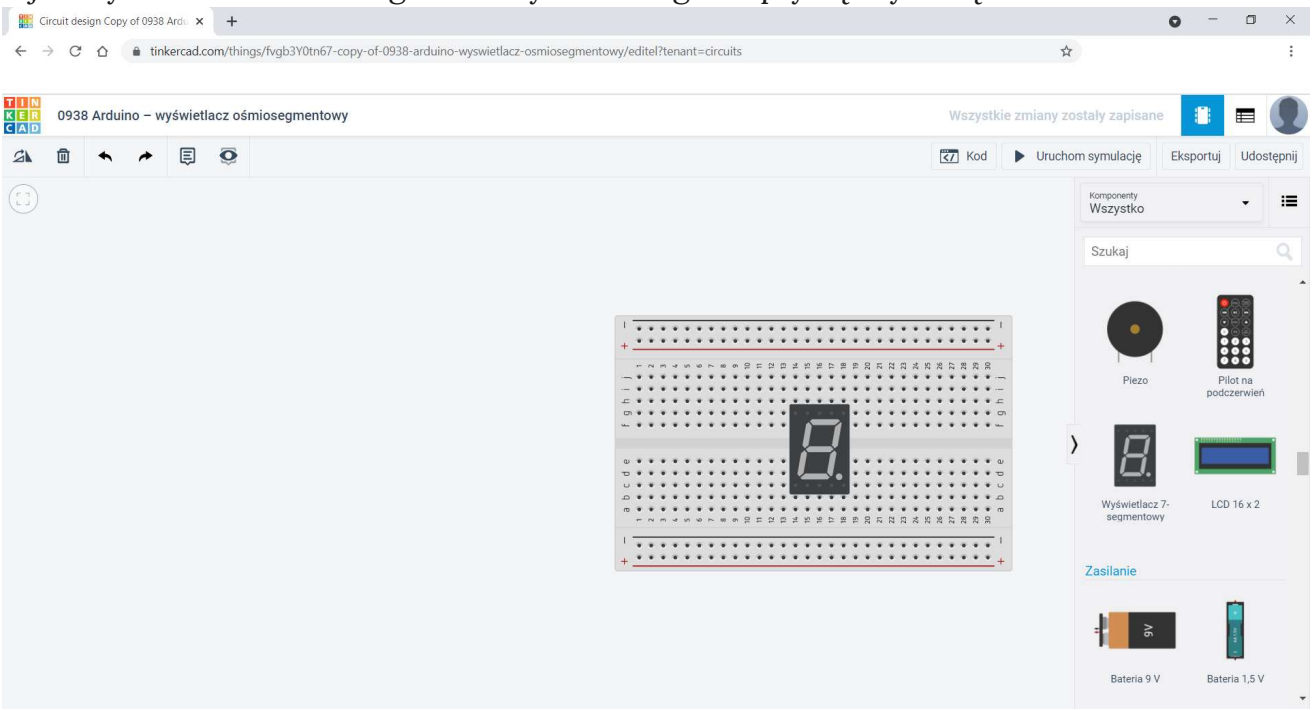

Dodaj do głównego okna symulatora płytkę stykową.



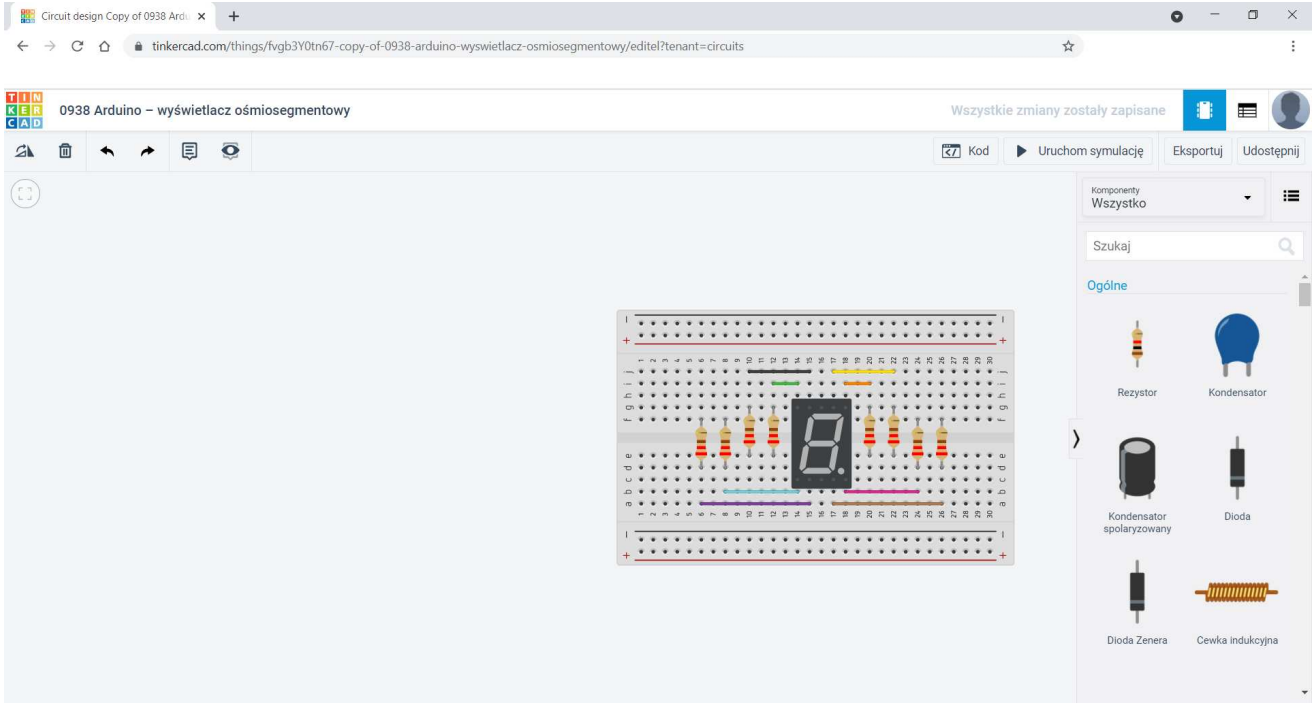

Z listy rozwijanej `Komponenty` wybierz `Wszystko`.



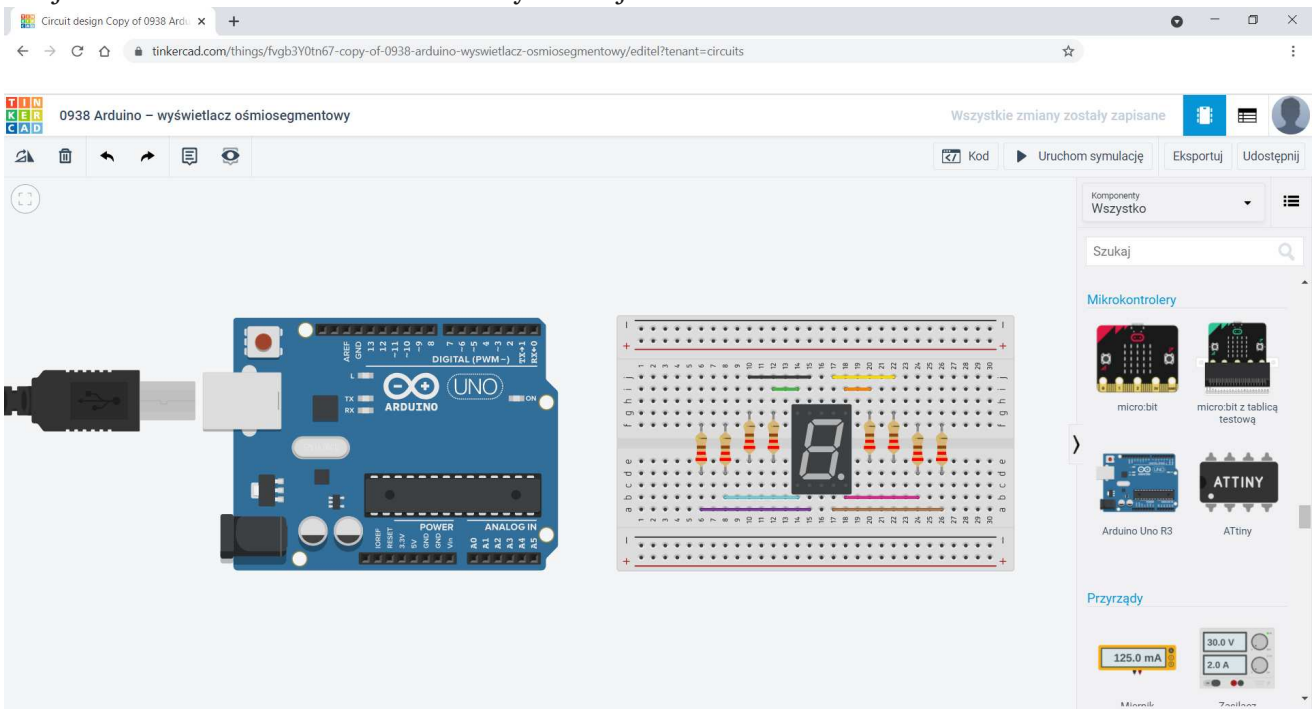

Znajdź wyświetlacz ośmiosegmentowy i wstaw go na płytkę stykową.



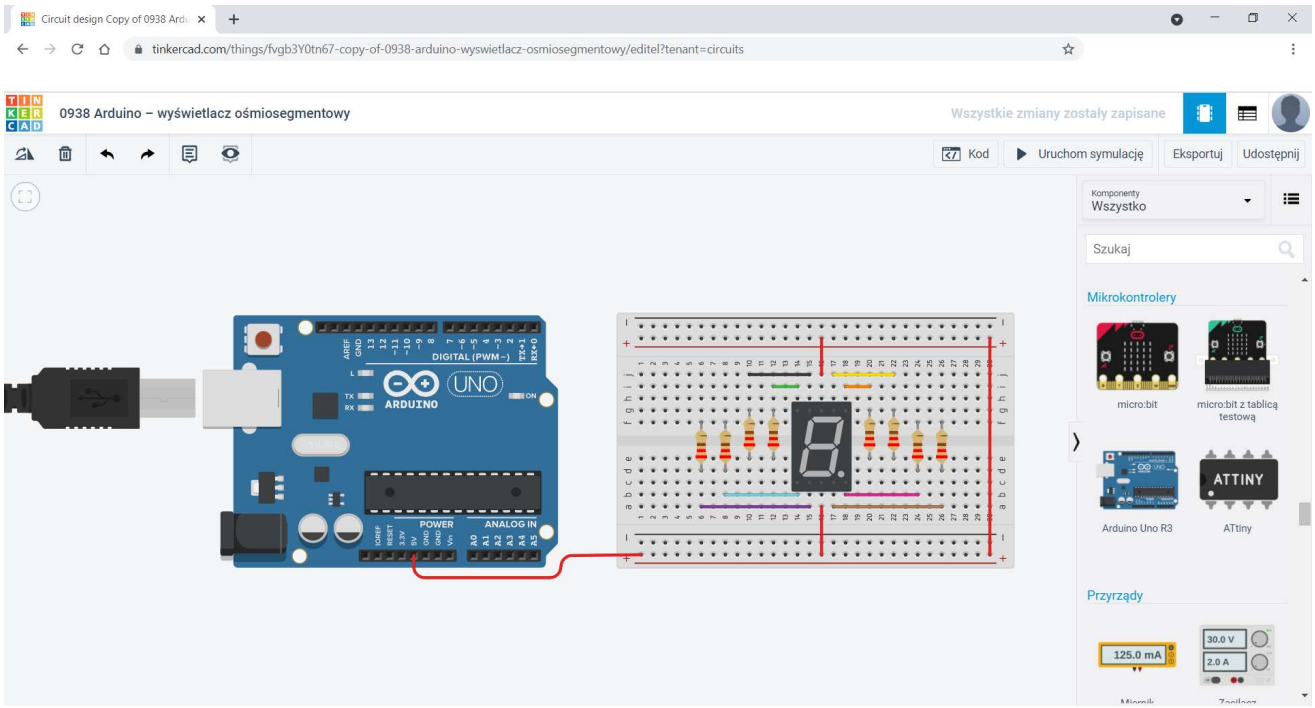

Do każdej katody diody znajdującej się w wyświetlaczu dodaj rezystory o wartości `220 Ω`.



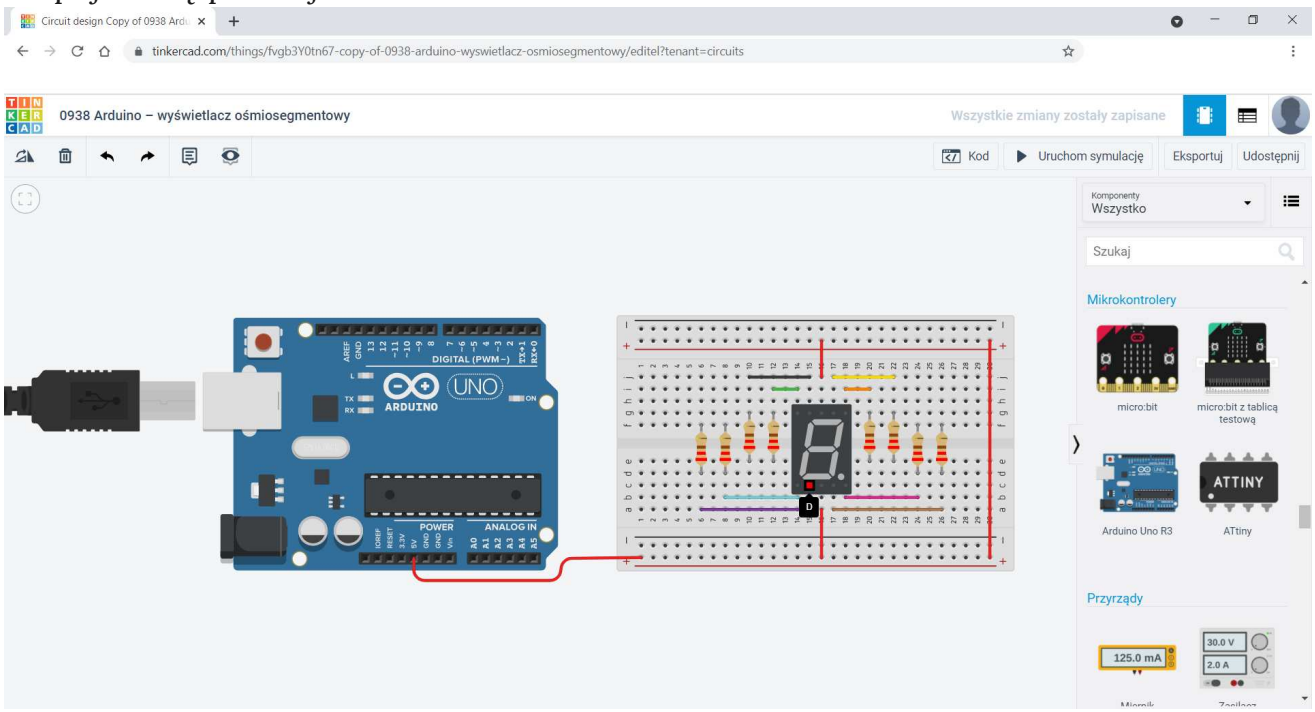

Dodaj moduł Arduino Uno do okna symulacji.




Pod dodatnią szynę płytki podepnij piny 3 oraz 8 wyświetlacza oraz pin 5V Arduino. Nie zapomnij o tym by obie szyny połączyć razem.



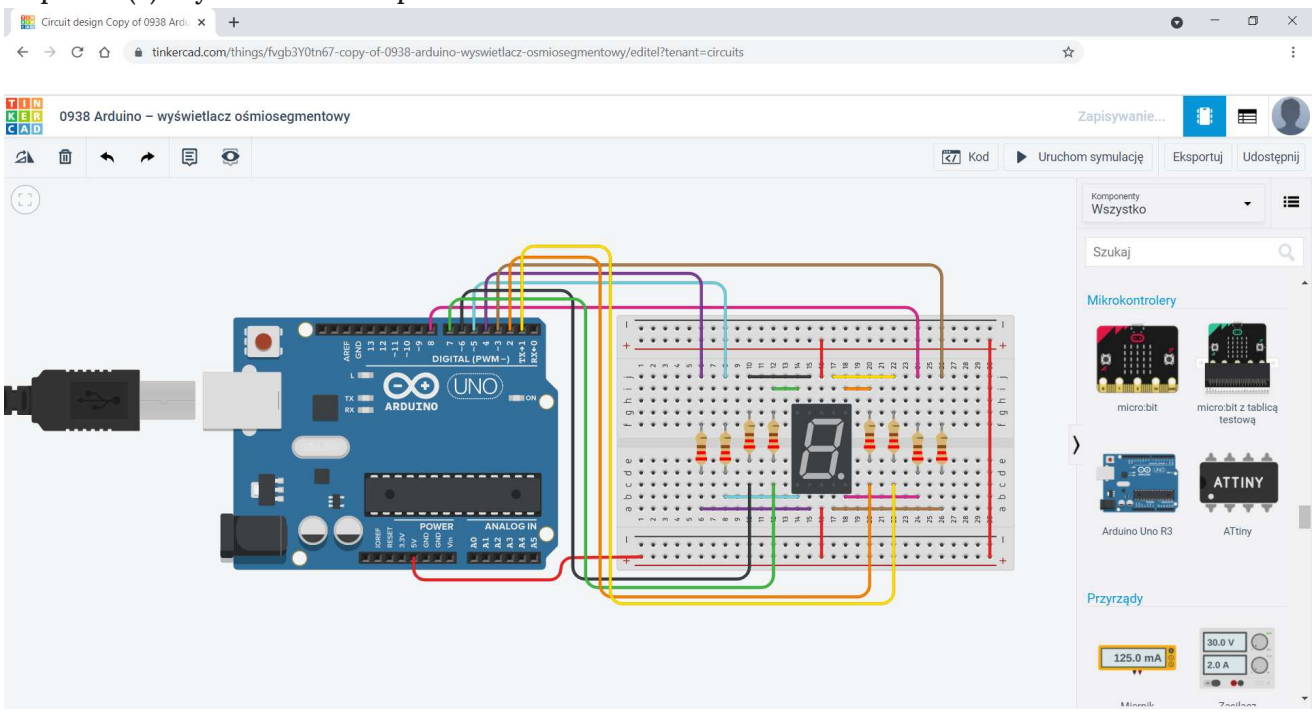

Teraz musimy podpiąć segmenty wyświetlacza do Arduino. Jeśli nie wiesz jaki pin wyświetlacza odpowiada jakiemu segmentowi, najedź kursorem na pin a informacja z nazwą pinu pojawi się poniżej kursora.




Podepnij segmenty wyświetlacza do Arduino Uno:

- pin DP (5) wyświetlacza z pinem D8 Arduino,
- pin G (10) wyświetlacza z pinem D7 Arduino,
- pin F (9) wyświetlacza z pinem D6 Arduino,
- pin E (1) wyświetlacza z pinem D5 Arduino,
- pin D (2) wyświetlacza z pinem D4 Arduino,
- pin C (4) wyświetlacza z pinem D3 Arduino,

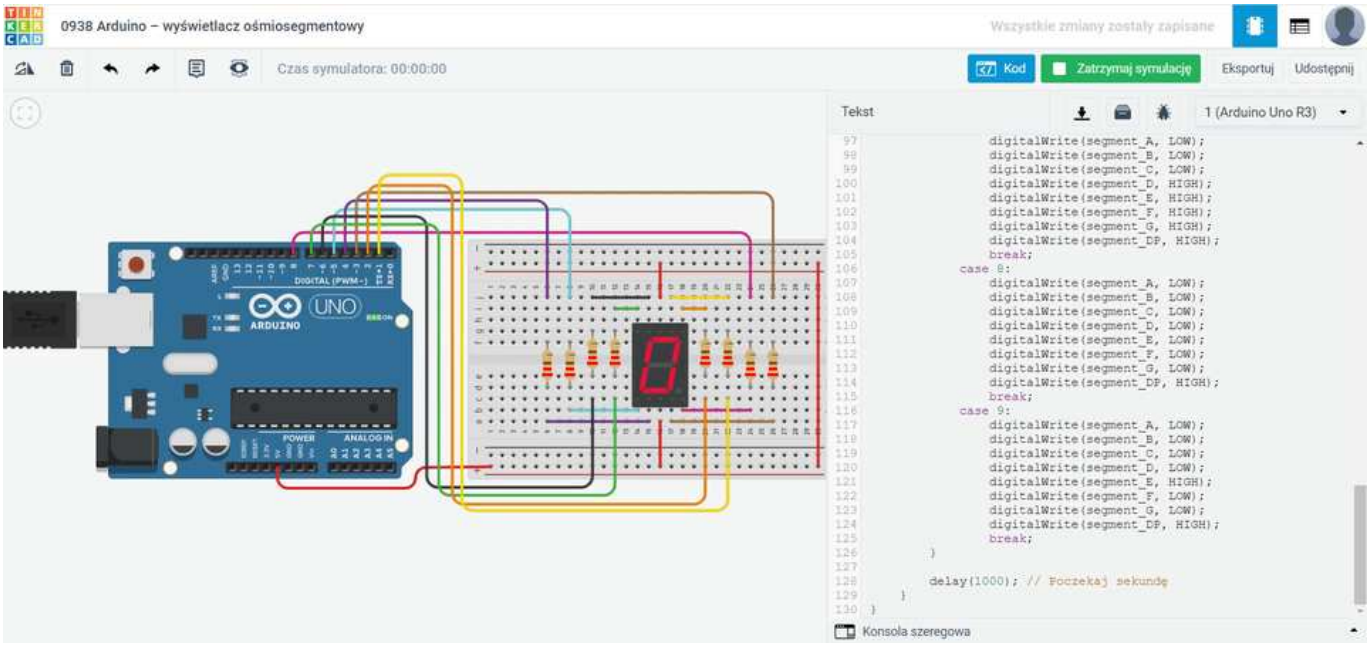
- pin B (6) wyświetlacza z pinem D2 Arduino,
- pin A (7) wyświetlacza z pinem D1 Arduino.




Przejdźmy teraz do pisania kodu. W tym celu klikamy na przycisk `Kod`, z listy rozwijanej wybieramy `Tekst`, potwierdzamy chęć zmiany widoku i czyścimy zawartość.

Przepisz kod licznika, dopasuj widok okna i uruchom symulację.

Film z działania programu.




Film dostępny pod adresem /preview/resource/RhMBmz9Lqxp0D

Film przestawiający działanie programu.

# Słownik

> **wyświetlacz 8-segmentowy**
>
> wyświetlacz, który składa się z ośmiu segmentów w postaci diod LED wyświetlających cyfry dziesiętne

# Prezentacja multimedialna

**Polecenie 1**

Zapoznaj się ze sposobem generowania liczb prawdziwie losowych, a następnie wykonaj ćwiczenie.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Do generowania liczb pseudolosowych w środowisku Arduino użyjemy funkcji `random`.

```
random();
```

Funkcja ta może przyjąć dwa argumenty odpowiadające za przedział, z którego będzie generowana liczba.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

W zależności od potrzeb możemy użyć zapisu z jednym argumentem, np:

```
random(100);
```

W tym momencie funkcja zwróci wartość z zakresu 0-99.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Czasami zachodzi potrzeba ograniczenia zakresu – w takiej sytuacji dodajemy drugi argument:

```
random(50, 100);
```

Teraz funkcja zwróci wartość z zakresu 50-99.

4

Materiał audio dostępny pod adresem:

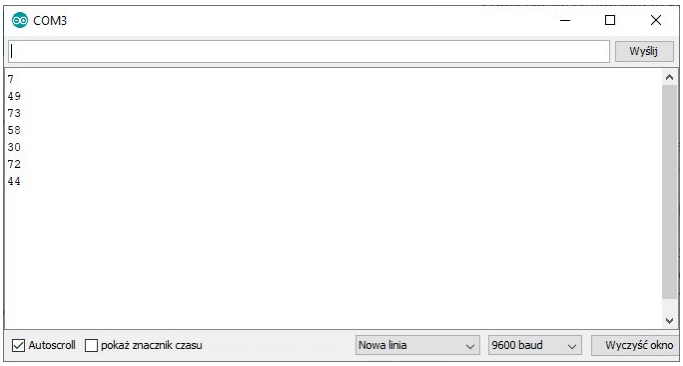
https://zpe.gov.pl/b/Pfe4QOIz4

Sprawdźmy, jak funkcja działa w praktyce. W środowisku Arduino IDE utwórzmy nowy szkic.

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

void loop() {

Serial.println(random(100)
);
    delay(1000);
}
```

Jest to prosty kod, którego zadaniem jest wyświetlanie w monitorze portu szeregowego losowej liczby z zakresu 0-99.

5




Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Efektem działania kodu jest ciąg losowych liczb z zakresu 0-99.

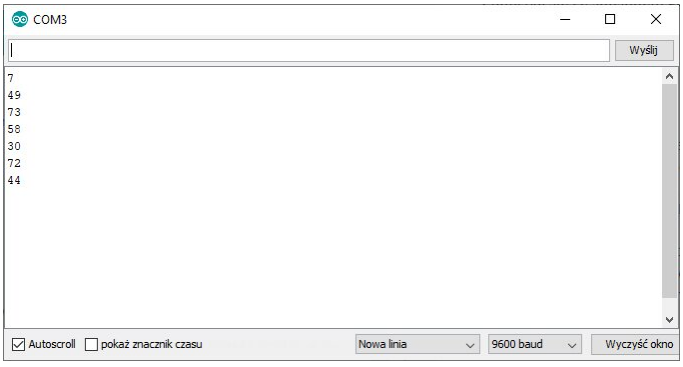
6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Niestety, losowość tych liczb jest pseudolosowa, ponieważ mikrokontroler działa bardzo precyzyjnie i tak naprawdę nie losuje tych liczb, tylko wykonuje określone operacje na poprzednio wylosowanych liczbach.

7




Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Uruchommy program kilka razy i sprawdźmy, co się wydarzy. Tak jak mogliśmy się spodziewać, kilkukrotne uruchamianie tego samego kodu daje nam dokładnie ten sam ciąg liczb pseudolosowych.

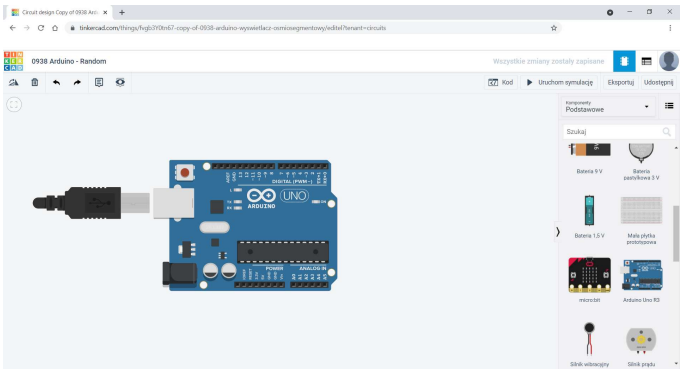
8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Dzieje się tak, ponieważ „losowanie” zaczyna się zawsze od tej samej liczby. Na szczęście jest na to sposób – musimy zastosować funkcję `randomSeed()`, która będzie odpowiedzialna za pierwszą liczbę, czyli za tzw. ziarno/zarodek.

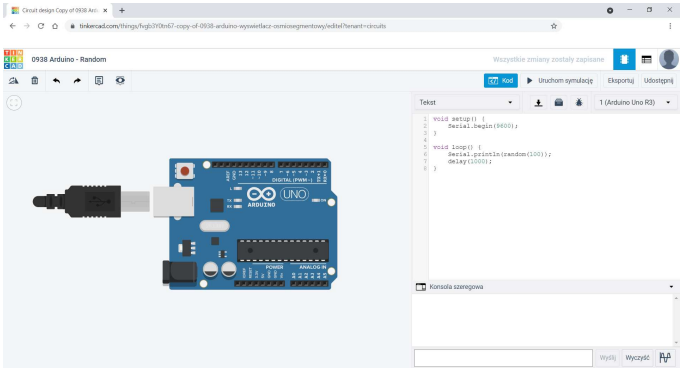
9




Zróbmy symulację za pomocą środowiska Tinkercad.

Dodaj do głównego okna moduł Arduino Uno.

10




Przejdźmy teraz do pisania kodu. W tym celu klikamy na przycisk Kod, z listy rozwijanej wybieramy Tekst, potwierdzamy chęć zmiany widoku i czyścimy zawartość.

Przepisz kod, uruchom Konsolę szeregową oraz symulację.

11

Film z działania programu.

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Funkcja `randomSeed()` przyjmuje jeden parametr, czyli właśnie tę naszą liczbę początkową. Przy czym, jeżeli wpiszemy jako argument tej funkcji stałą liczbę, np: 10, to przy kolejnych uruchomieniach programu znów otrzymamy te same ciągi liczb.

```
1 void setup() {
2     Serial.begin(9600);
3     randomSeed(10);
4 }
5
6 void loop() {
```

```
7
   Serial.println(random(100)
   );
8      delay(1000);
9  }
```

13

Nie musimy robić nowego podłączenia.

Wystarczy zmodyfikować kod i uruchomić symulację.

Film z działania programu.

14

Materiał audio dostępny pod adresem:

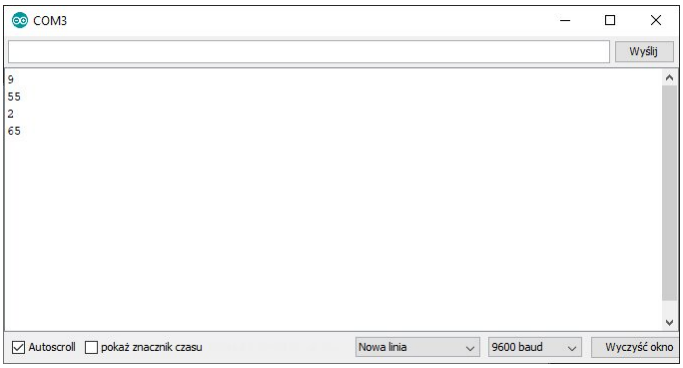
https://zpe.gov.pl/b/Pfe4QOIz4

I tutaj powstaje problem, ponieważ żeby wylosować liczby prawdziwie losowe, musimy podać na wstępie liczbę prawdziwie losową. Na szczęście w Arduino możemy w łatwy sposób pozyskać taką losową liczbę, której możemy użyć jako zarodka. Wykorzystamy do tego jeden z nieużywanych pinów analogowych. Wystarczy, że argumentem funkcji `randomSeed()` będzie wartość przechwycona z pinu analogowego. Dzieje się tak, ponieważ nieużywany pin analogowy zbiera „śmieci” z otoczenia i odczytana wartość praktycznie nigdy nie będzie równa 0, tylko ta liczba będzie się losowo zmieniać.

```
1  void setup() {
2      Serial.begin(9600);
3
   randomSeed(analogRead(0));
4  }
5
```

```
6 void loop() {
7
  Serial.println(random(100)
  );
8     delay(1000);
9 }
```

15




Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Pfe4QOIz4

Teraz po uruchomieniu programu za każdym razem będziemy otrzymywać inny ciąg liczb prawdziwie losowych.

16

W tym przypadku również nie musimy modyfikować układu, wystarczy zmienić kod.

Film z działania programu.



**Polecenie 2**

Zbuduj cyfrową kostkę do gry.

# Sprawdź się

Pokaż ćwiczenia:

### Ćwiczenie 1

Wskaż, ile segmentów trzeba włączyć w wyświetlaczu 8-segmentowym, aby wyświetlić cyfrę 2.

- ◯ 6
- ◯ 5
- ◯ 4

### Ćwiczenie 2

Dokończ zdanie:
**Segmenty na wyświetlaczu 8–segmentowym są opisane za pomocą...**

- ◯ cyfr arabskich
- ◯ cyfr rzymskich
- ◯ liter

### Ćwiczenie 3

Korzystając z zaproponowanych poniżej elementów, uzupełnij kod tak, aby pin `segment_A` był pinem wyjściowym.

```
pinMode(        ,        );
```

`HIGH` `INPUT` `segment_A` `OUTPUT` `LOW`

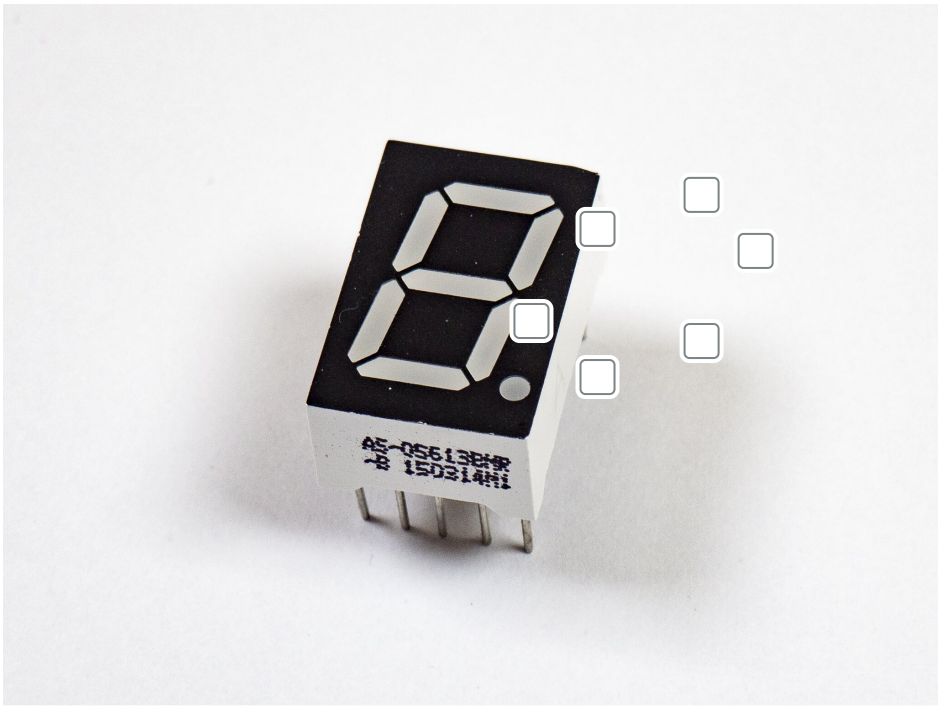
### Ćwiczenie 4

Wskaż, co wspólnego mogą mieć różne wyświetlacze 8-segmentowe.

- [ ] nóżkę odpowiedzialną za wyświetlenie kropki i segmentu d
- [ ] katodę
- [ ] nóżkę odpowiedzialną za wyświetlenie kropki i segmentu a
- [ ] anodę

### Ćwiczenie 5

Wskaż segment a na wyświetlaczu.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

### Ćwiczenie 6

Korzystając z zaproponowanych poniżej elementów, uzupełnij funkcję `random()` tak, aby losowane liczby należały do przedziału 20-100.

```
(random(      ,      ));
```

`102` `99` `21` `100` `20` `101`

### Ćwiczenie 7

Dokończ zdanie:

**W przypadku dużej liczby warunków zamiast instrukcji warunkowej `if` możemy użyć...**

- ◯ funkcji `random()`
- ◯ instrukcji `switch-case`
- ◯ pętli `for`

### Ćwiczenie 8

Wskaż, której funkcji należy użyć, aby ustawić tzw. ziarno w przypadku funkcji `random()`.

- ◯ `delay(1000);`
- ◯ `Serial.begin(9600);`
- ◯ `Serial.println(random(100));`
- ◯ `randomSeed();`

# Dla nauczyciela

**Autor:** Dawid Mazur
**Przedmiot:** Informatyka

**Temat: Arduino – wyświetlacz ośmiosegmentowy**

**Grupa docelowa:**

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum

**Podstawa programowa:**

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym

struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:

a) tworzy i edytuje dwuwymiarowe oraz trójwymiarowe wizualizacje i animacje, stosuje właściwe formaty plików graficznych,

b) uczestniczy w opracowaniu dokumentacji projektu zespołowego, pracując przy tym w odpowiednim środowisku,

IV. Rozwijanie kompetencji społecznych.

Zakres podstawowy. Uczeń:

1) aktywnie uczestniczy w realizacji projektów informatycznych rozwiązujących problemy z różnych dziedzin, przyjmuje przy tym różne role w zespole realizującym projekt i prezentuje efekty wspólnej pracy;

6) poszerza i uzupełnia swoją wiedzę korzystając z zasobów udostępnionych na platformach do e-nauczania.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) przy realizacji zespołowego projektu programistycznego posługuje się środowiskiem przeznaczonym do współpracy i realizacji projektów zespołowych, w tym środowiskiem w chmurze; współtworzy zasoby udostępniane na platformach do e-nauczania;

V. Przestrzeganie prawa i zasad bezpieczeństwa.

Zakres podstawowy. Uczeń:

1) postępuje zgodnie z zasadami netykiety oraz regulacjami prawnymi dotyczącymi: ochrony danych osobowych, ochrony informacji oraz prawa autorskiego i ochrony własności intelektualnej w dostępie do informacji; jest świadomy konsekwencji łamania tych zasad;

2) respektuje obowiązujące prawo i normy etyczne dotyczące korzystania i rozpowszechniania oprogramowania komputerowego, aplikacji cudzych i własnych

oraz dokumentów elektronicznych;

3) stosuje dobre praktyki w zakresie ochrony informacji wrażliwych (np. hasła, pin), danych i bezpieczeństwa systemu operacyjnego, objaśnia rolę szyfrowania informacji;

**Kształtowane kompetencje kluczowe:**

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

**Cele operacyjne (językiem ucznia):**

- Poznasz budowę 8-segmentowego wyświetlacza LED.
- Dowiesz się, jak podłączyć wyświetlacz 8-segmentowy do płytki Arduino.
- Zapoznasz się z możliwościami generowania liczb prawdziwie losowych.

**Strategie nauczania:**

- konstruktywizm;
- konektywizm.

**Metody i techniki nauczania:**

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

**Formy pracy:**

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

**Środki dydaktyczne:**

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

**Przebieg lekcji**

**Przed lekcją:**

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Arduino – wyświetlacz ośmiosegmentowy". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

**Faza wstępna:**

1. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

**Faza realizacyjna:**

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna", czyta treść polecenia nr 1 „Zapoznaj się ze sposobem generowania liczb prawdziwie losowych, a następnie zbuduj cyfrową kostkę do gry." i omawia kolejne kroki rozwiązania.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1-8. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

**Faza podsumowująca:**

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

**Praca domowa:**

1. Uczniowie opracowują FAQ (minimum 3 pytania i odpowiedzi) do tematu lekcji („Arduino – wyświetlacz ośmiosegmentowy").

**Materiały pomocnicze:**

- Oficjalna dokumentacja techniczna dla projektu Arduino.

**Wskazówki metodyczne:**

- Nauczyciel może wykorzystać multimedium w sekcji „Prezentacja multimedialna" do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Arduino – wyświetlacz ośmiosegmentowy".