Zespół Szkół Technicznych im. J. i J. Śniadeckich w Grudziądzu

Pracownia Automatyki i Robotyki (s.48)

Instrukcja Laboratoryjna:

Wprowadzenie do Arduino

I. Czym jest Arduino…

Arduino – platforma programistyczna dla systemów wbudowanych oparta na prostym projekcie Open Hardware przeznaczonym dla mikrokontrolerów montowanych w pojedynczym obwodzie drukowanym, z wbudowaną obsługą wejścia/wyjścia oraz standaryzowanym językiem programowania

Czym właściwie jest **Arduino**? Spróbuję opisać to **najprostszymi słowami**:
Arduino to płytka z mikrokontrolerem oraz całą dodatkową elektroniką, która pomaga/ułatwia programowanie i wykorzystywanie Uc w najróżniejszych projektach. O tym, czym jest mikrokontroler pisałem już wcześniej (link do artykułu: „Mikrokontroler…”). Sercem układu jest mikrokontroler AVR firmy **Atmel** (konkretnie ATmega8, ATmega168, ATmega328, oraz ATmega1280 i ATmega 2560). Większość płytek zawiera 5V ***regulator*** napięcia, 16 MHz ***rezonator*** kwarcowy, ***piny*** wejścia/wyjścia oraz dodatkowe elementy usprawniające pracę (np. piny PWM). Mikrokontroler w platformie jest wstępnie zaprogramowany z wykorzystaniem programu rozruchowego, co upraszcza przesyłanie programu do pamięci flash (jest to „dysk twardy” naszego mikrokontrolera) układu, w porównaniu do innych urządzeń, gdzie najczęściej potrzebny jest zewnętrzny programator.
Najłatwiej zaprogramować Arduino z pomocą specjalnie do tego przygotowanego programu: **Arduino IDE**. Jest to wieloplatformowa aplikacja napisana w języku Java. Środowisko jest zaprojektowane w taki sposób, aby było przyjazne dla hobbystów i osób niezajmujących się tworzeniem oprogramowania na co dzień. IDE zawiera edytor kodu z takimi funkcjami jak podświetlanie składni czy automatyczne wcięcia w kodzie, oraz pozwala na kompilację i upload programu do płyty Arduino. Zazwyczaj nie ma potrzeby dodatkowej edycji plików Makefile (jest to plik, dzięki któremu jesteśmy w stanie wgrać program do mikrokontrolera – na razie nie musicie nic o nim wiedzieć) lub uruchamiania programów z linii poleceń, co znacznie ułatwia pracę.
Oficjalnie Arduino wypuściło **kilka wersji** swojej platformy, tak aby dobrać odpowiednią, w sam raz dla siebie. Poniżej prezentuję listę dostępnych modeli:

1. Serial Arduino, programowany przez interfejs szeregowy DB9 i korzystający z układu ATmega8
2. Arduino Extreme, programowany przez interfejs USB i zaopatrzony w układ ATmega8
3. Arduino Mini, miniaturowa wersja Arduino wykorzystująca montowany powierzchniowo układ ATmega168 lub Atmega328
4. Arduino Nano, jeszcze mniejsza wersja Arduino, zasilana przez USB zaopatrzona w montowany powierzchniowo układ ATmega168
5. Arduino LilyPad, minimalistyczny projekt wykorzystujący montowany powierzchniowo układ ATmega168
6. Arduino NG, programowany przez interfejs USB i korzystający z układu ATmega8
7. Arduino NG plus, programowany za pośrednictwem interfejsu USB i zaopatrzony w układ ATmega168
8. Arduino BT, programowana bezprzewodowo przez interfejs Bluetooth korzystająca z układu ATmega168
9. Arduino Diecimila, z interfejsem USB i układem Atmega168 w obudowie DIL28
10. Arduino Duemilanove („2009″), wykorzystujący Atmega168 (Atmega328 w nowszych wersjach) zasilany z zewnętrznego zasilacza lub przez USB

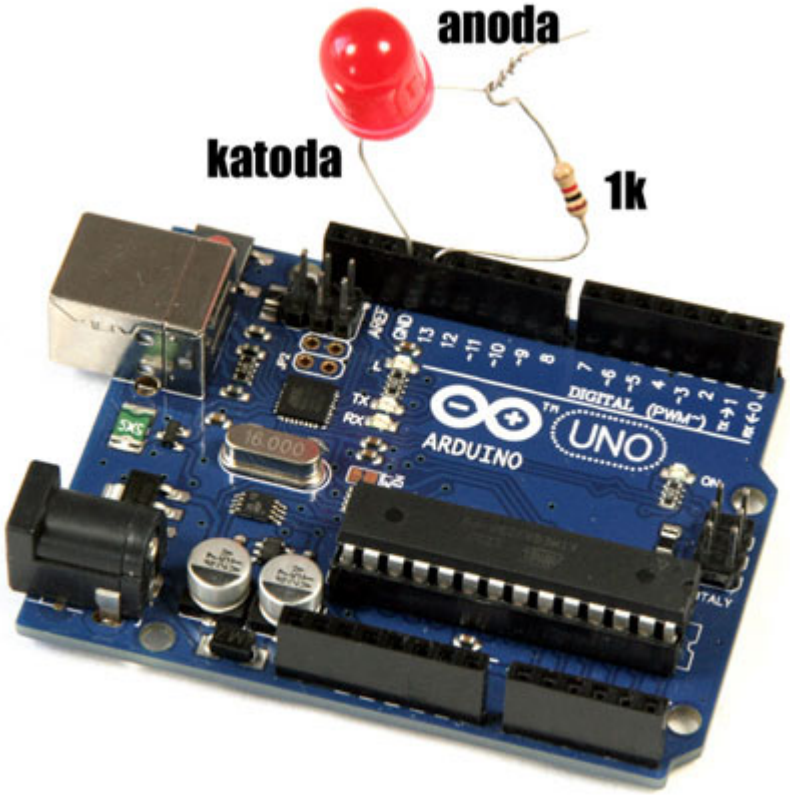
11. Arduino Mega, zaopatrzony w montowany powierzchniowo układ ATmega1280, dzięki czemu posiada więcej pamięci i pinów wejścia/wyjścia.
12. Arduino Uno, wersja z programowalnym interfejsem USB i układem Atmega328
13. Arduino Leonardo, wersja z układem ATmega32u4, który jest sercem Arduino i odpowiada za komunikację USB
14. Arduino Ethernet, wersja z układem ATmega328, interfejsem Ethernet zamiast USB oraz czytnikiem kart microSD
15. Arduino Yún, wersja z układem ATmega32u4 oraz wbudowanym układem WiFi
16. Arduino Esplora, wersja z układem Atmega32U4, płytka zaprojektowana jako baza pod kontroler do konsoli
17. Arduino Robot, wersja zbudowana z dwóch okrągłych płyt, każda wyposażona w procesor ATmega32u4

Za najpopularniejsze wersje uznałbym Arduino: **UNO, Leonardo, Mega, Nano, Mini** – ja bym je zaklasyfikował do tych najbardziej znanych.
Jednak Oryginalne płytki to nie wszystko, internet „rozrywa" masa klonów, czyli układów identycznych bądź podobnych do oficjalnej wersji. Główną różnicą jest cena; Cena oryginału: ~150zł Cena klonu: ~50zł … No i podzespoły. Zazwyczaj do produkcji tanich klonów stosuje się części gorszej jakości.

II. Pierwszy prosty program na Arduino…

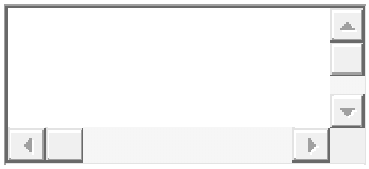
Dołączenie diody LED do mikrokontrolera i jej wysterowanie to zwykle jedno z pierwszych zadań stawianych przed adeptami sztuki programowania mikrokontrolerów. I słusznie, gdyż okiełznanie potencjału drzemiącego we współczesnych mikrokontrolerach powinno się zaczynać od jak najprostszych wyzwań.

W artykule Język programowania Arduino – jedyny słuszny wybór przedstawiono najczęściej przytaczany przykład pierwszego programu mikrokontrolerowego, czyli sterowanie diodą LED dołączoną do linii we/wy mikrokontrolera. Układ testowy dla tego przykładu pokazano na **fotografii 1**. Na **listingu 1** znajduje się program powodujący co 1 sekundę świecenie i gaszenie diody LED dołączonej do linii 13.



Fot. 1. Dołączenie diody LED do zestawu Arduino Uno

Listing 1. Program migania diodą LED dla układu pokazanego na fot. 1



```
const int LED = 13;         // dioda LED jest dołączona do linii 13

void setup()                // funkcja konfigurująca
{
  pinMode(LED, OUTPUT);     // ustawienie linii LED jako wyjściowej
}

void loop()                 // pętla nieskończona
{
  digitalWrite(LED, HIGH);  // ustawienie bitu
  delay(1000);              // opóźnienie 1 s (1000 ms)
  digitalWrite(LED, LOW);   // wyzerowanie bitu
  delay(1000);              // opóźnienie 1 s (1000 ms)
}
```

Teraz postawimy sobie odrobinę ambitniejsze zadanie – do mikrokontrolera zostanie dołączonych 8 diod LED i uzyskamy na nich efekt przesuwającego się punktu świetlnego. Schemat elektryczny naszej platformy sprzętowej pokazano na **rysunku 2**, natomiast na **rysunku 3** pokazano sposób wykonania połączeń. Rysunki, czyli uproszczoną dokumentację techniczną projektu, wykonano z wykorzystaniem programu fritzing. Zmontowany układ pokazano na **fotografii 4**. Podczas montażu wykorzystano:

– Arduino Uno zestaw z mikrokontrolerem ATmega328 (rodzina AVR firmy Atmel)

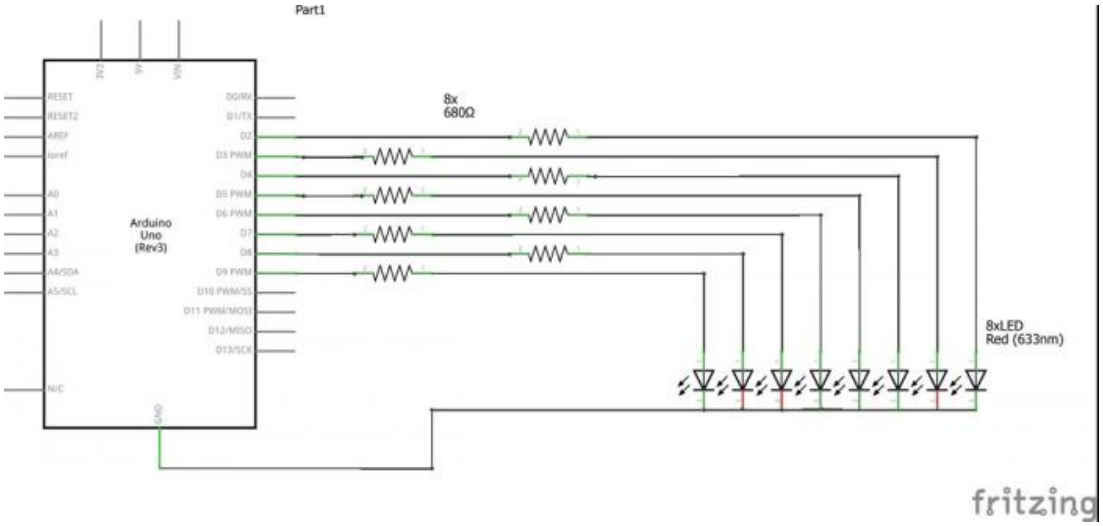
– przewód USB do programowania i zasilania Arduino Uno (np. CAB_USB_AB)

– płytkę stykową (np. PPS0400)

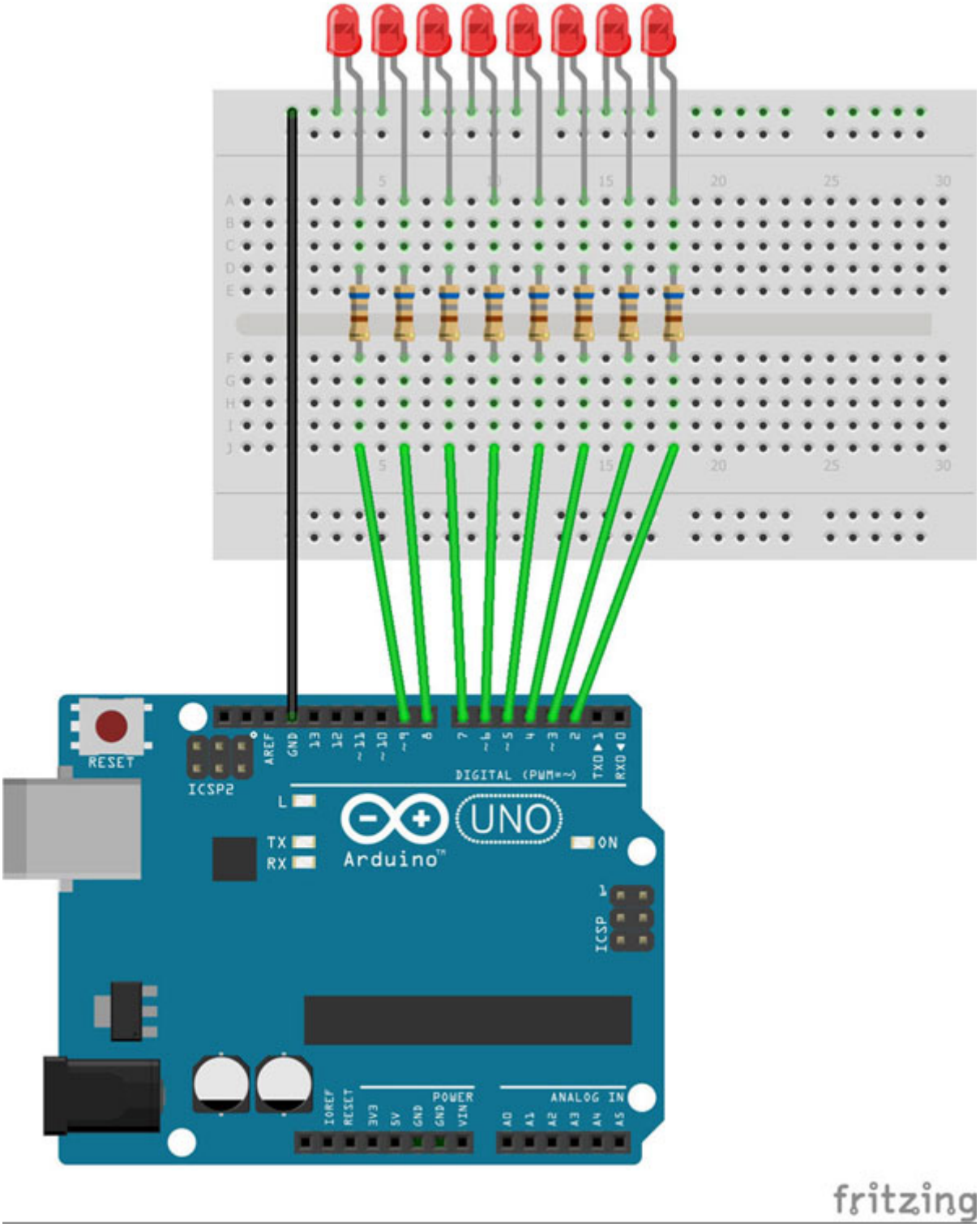
– 8 diod LED (np. LED-AL-30R-D00300-60)

– 8 rezystorów 680 Ohm

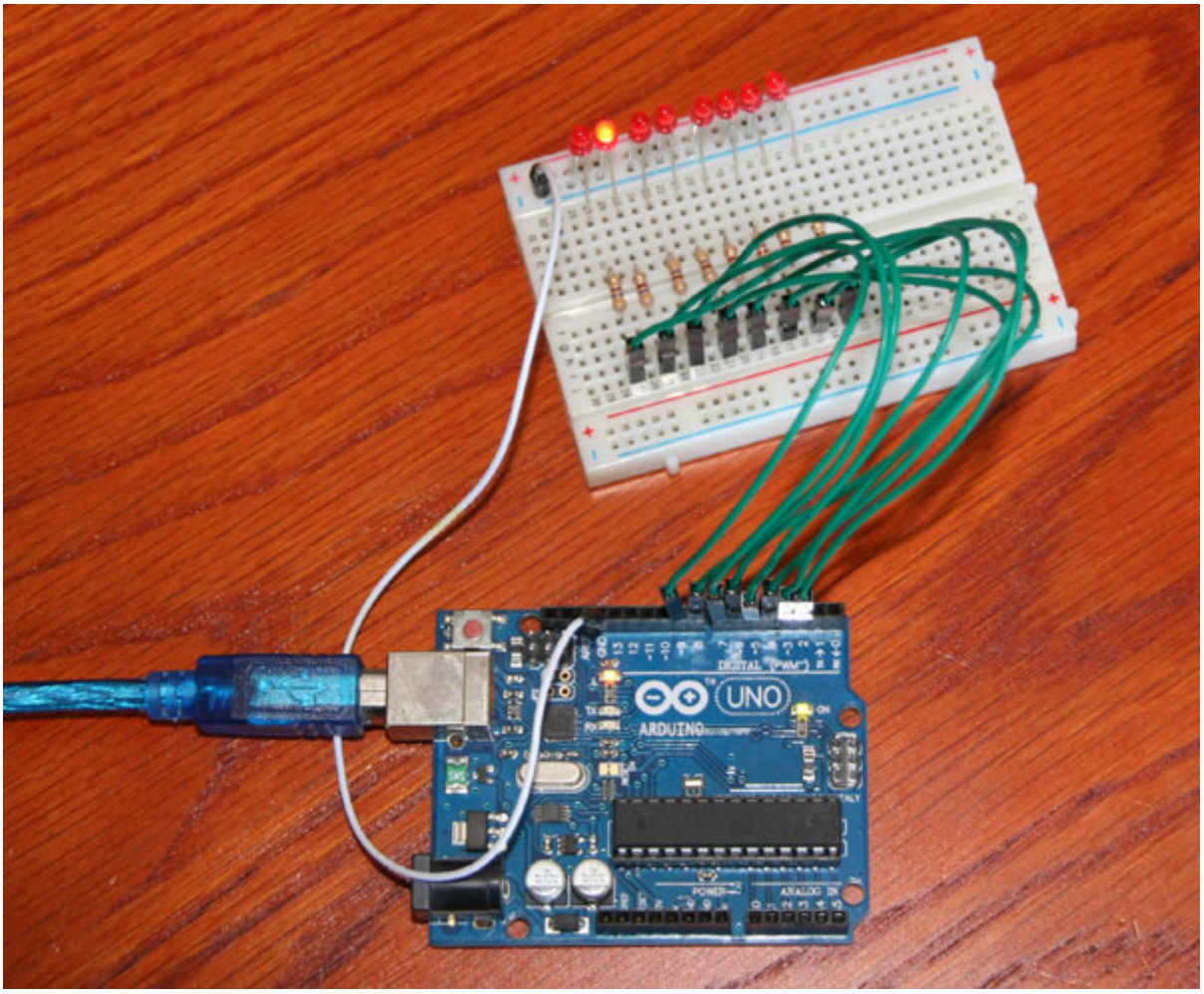
– 9 przewodów z zestawu CAB_M-M(65-Rainbow)



Rys. 2. Dołączenie 8 diod LED do Arduino Uno – schemat elektryczny

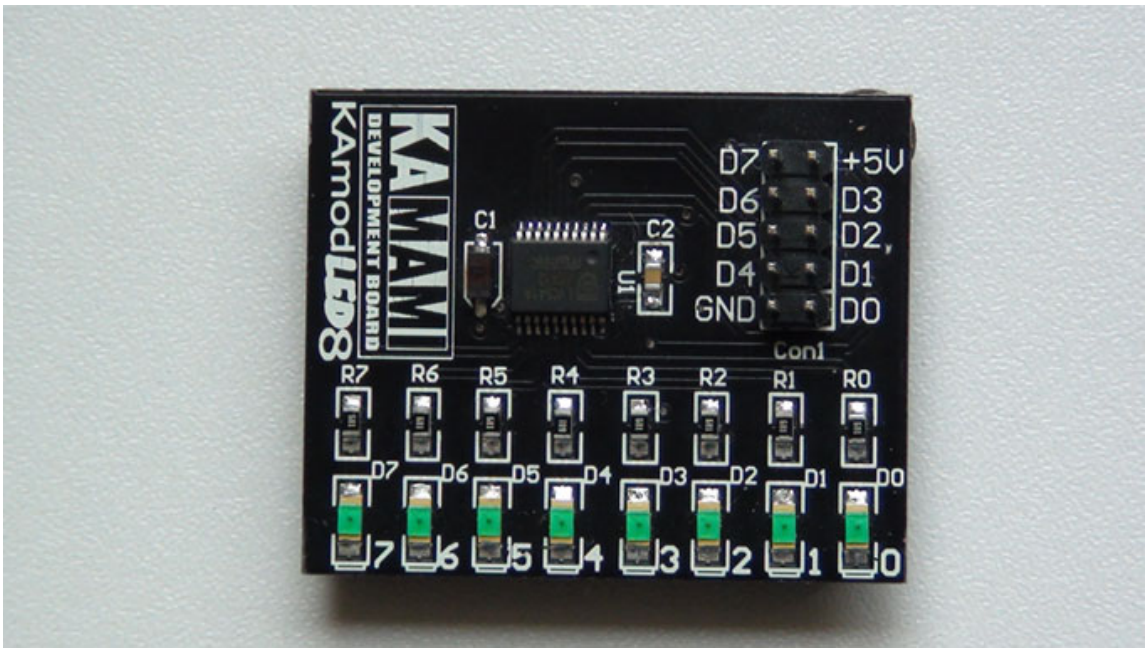


Rys. 3. Dołączenie 8 diod LED do Arduino Uno – wykonanie połączeń



Fot. 4. Widok zmontowanego układu

Układ pokazany na rysunku 2 można zbudować również odrobinę łatwiej (ale i drożej) z wykorzystaniem modułu KAmodLED8 pokazanego na **fotografii 5**.



Fot. 5. W realizacji projektu można również zastosować moduł KAmodLED8

Po zmontowaniu układu zgodnie z rysunkiem 3 należy go dołączyć do komputera osobistego kablem USB i można rozpocząć eksperymenty z programem. Na **listingu 2** pokazano program pozwalający uzyskać efekt przesuwającego się punktu świetlnego.

Listing 2. Program sterujący linijką diod LED dla układu pokazanego na fot. 4 (prog_01.ino)

```
  }

 }
```

```
1  #define LED_POCZ 2
2  #define LED_KON 9
3
4  void setup()
5  {
6    int i;
7    //linie 2...9 jako wyjścia
8    for (i = LED_POCZ; i &lt;= LED_KON; i++)
9      pinMode(i, OUTPUT);
10 }
11
12 void loop()
13 {
14   int i;
15   //przesuwający się punkt
16   for (i = LED_POCZ; i &lt;= LED_KON; i++)
17   {
18     digitalWrite( i, HIGH );
19     delay(200);
20     digitalWrite( i, LOW );
21   }
22
23 }
```

W pierwszych dwóch liniach programu, wykorzystując dyrektywę preprocesora #define, zdefiniowano dwie stałe: LED_POCZ i LED_KON o wartościach odpowiednio 2 i 9. Diody LED są dołączone do linii we/wy Arduino Uno o numerach: 2, 3, 4, 5, 6, 7, 8 i 9. W funkcji setup() linie te są ustawiane w tryb wyjściowy za pomocą funkcji:

```
1  pinMode(i, OUTPUT);
```

Zdefiniowane stałe LED_POCZ i LED_KON odpowiadają pierwszej i ostatniej wykorzystywanej linii we/wy. W programie dostęp do tych linii jest realizowany za pomocą pętli for, która dla zwiększenia czytelności programu wykorzystuje te stałe:

```
1  for (i = LED_POCZ; i &lt;= LED_KON; i++)
2    pinMode(i, OUTPUT);
```

Po ustawieniu wymaganej konfiguracji linii we/wy pożądany efekt, czyli przesuwający się punkt świetlny, uzyskuje się w pętli:

```
1  for (i = LED_POCZ; i &lt;= LED_KON; i++)
2  {
3    digitalWrite( i, HIGH );
4    delay(200);
5    digitalWrite( i, LOW );
6  }
```

W każdej iteracji powyższej pętli jest zapalana kolejna dioda LED. Każda dioda świeci się przez 200 ms. Zmieniając wartość funkcji delay() można łatwo zmienić szybkość przesuwania się punktu świetlnego.

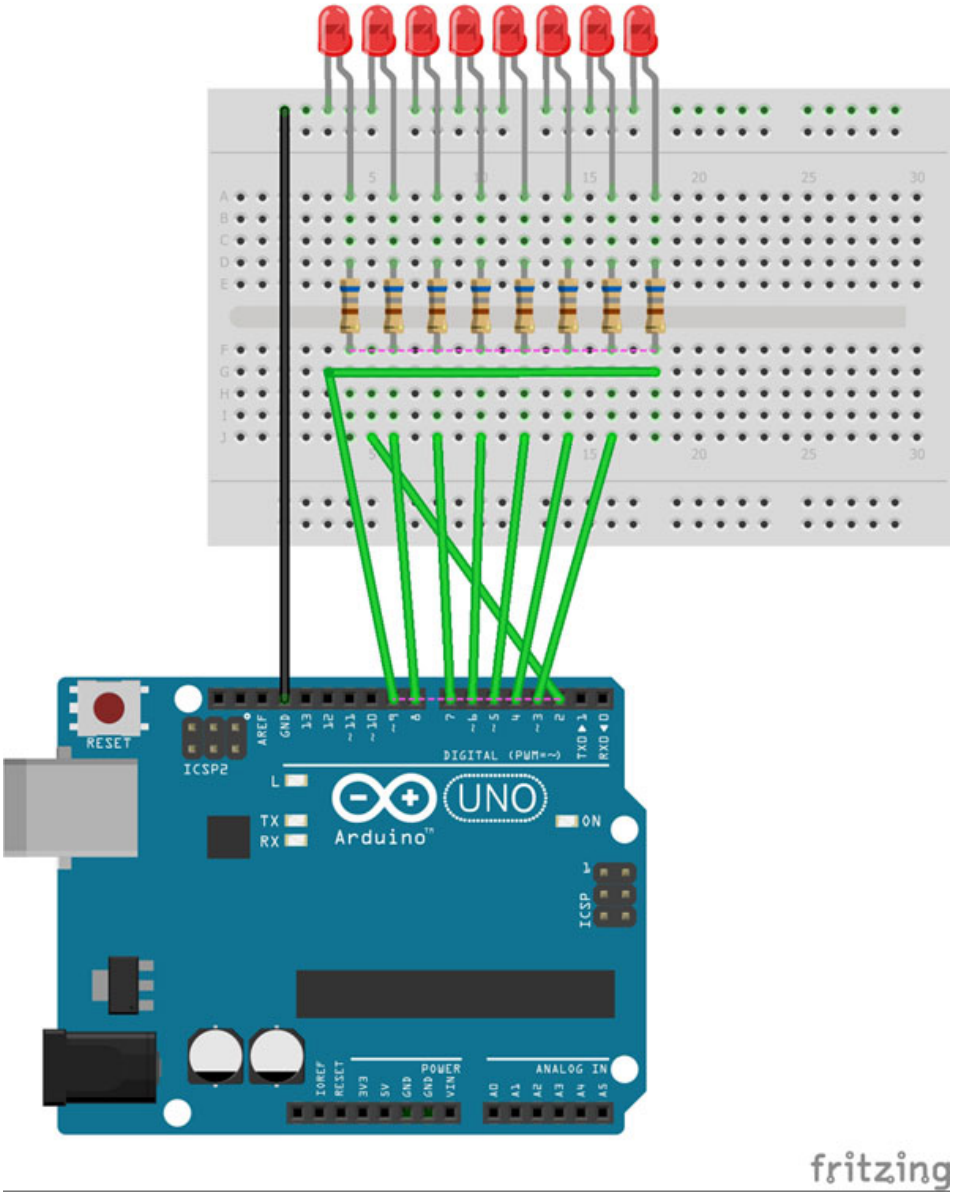
W tym miejscu warto zauważyć, że zmiana linii we/wy, do których są dołączone diody LED, w zasadzie demoluje nasz program. Niestety czasami projektując płytkę drukowaną okazuje się, że zmiana kolejności dołączenia tych diod, skutkowałaby znacznie łatwiejszą w wykonaniu płytką drukowaną. Jest łatwy sposób zabezpieczenia się przed tego typu niedogodnościami. Wystarczy do „adresowania” diod LED wykorzystać odpowiednio przygotowaną tablicę, tak jak to pokazano na **listingu 3**. Tablica leds[] zawiera numery linii we/wy, do których są dołączone diody LED. W przypadku układu pokazanego na rysunku 3 tablica leds[0] powinna wyglądać następująco:

```
int leds[] = {2, 3, 4, 5, 6, 7, 8, 9};
```

W programie odwołanie do pierwszej diody LED w linijce świetlnej odbywa się poprzez wywołanie pierwszego elementu tablicy, czyli leds[0], co zostanie w trakcie pracy programu zastąpione przez odwołanie do linii we/wy o numerze 2.

Jeżeli z jakichś powodów okaże się, że diody należy dołączyć w inny sposób, np. zamieniając diodę pierwszą z ostatnią (rysunek 6), to wystarczy zmienić zawartość tablicy leds[] bez modyfikowania reszty programu:

```
int leds[] = {9, 3, 4, 5, 6, 7, 8, 2};
```



Rys. 6. Widok połączeń Dołączenie 8 diod LED do Arduino Uno – wykonanie połączeń

Listing 3. Program sterujący linijką diod LED dla układu pokazanego na fot. 4 (prog_02.ino)

```
int leds[] = {2, 3, 4, 5, 6, 7, 8, 9};
#define LEDS_SIZE 8

void setup()
{
  int i;
  //linie z tablicy leds jako wyjścia
  for (i = 0; i &lt; LEDS_SIZE; i++)
    pinMode( leds[i], OUTPUT );
}

void loop()
{
  int i;
  //przesuwający się punkt
  for (i = 0; i &lt; LEDS_SIZE; i++)
  {
    digitalWrite( leds[i], HIGH );
    delay( 200 );
    digitalWrite( leds[i], LOW );
  }

}
```