Damian Tracz

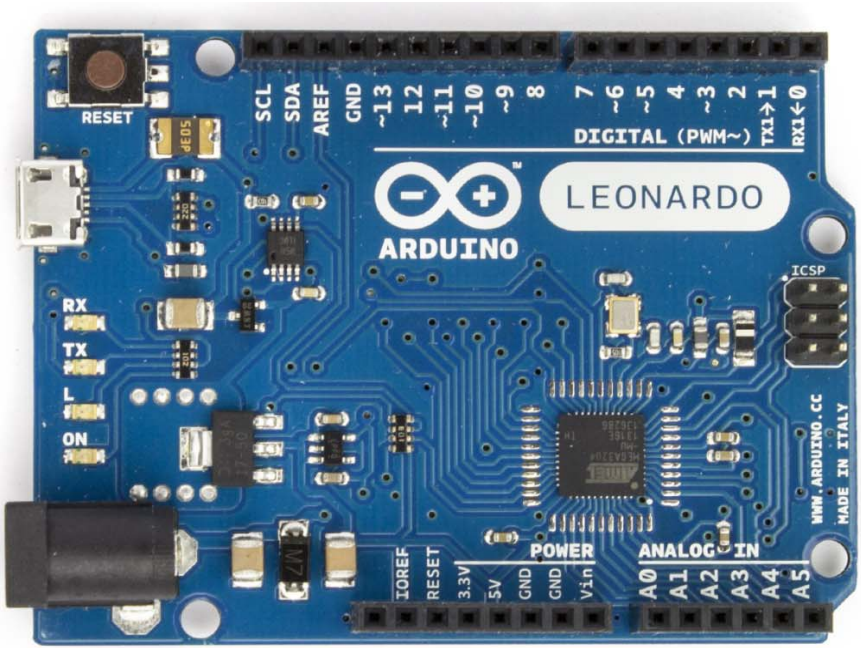
Arduino. Sterownik diod LED.

1. Cel ćwiczenia:

Głównym celem ćwiczenia jest zapoznanie się z możliwościami platformy Arduino oraz stworzenie sterownika 64 diod LED.

2. Wprowadzenie:

Arduino – platforma programistyczna dla systemów wbudowanych oparta na prostym projekcie Open Hardware (dostępne są szczegółowe schematy wykonania płytki) przeznaczonym dla mikrokontrolerów montowanych w pojedynczym obwodzie drukowanym (płytka z połączeniami elektrycznymi, przeznaczone do montowania podzespołów elektronicznych), z wbudowaną obsługą wejścia/wyjścia oraz standaryzowanym językiem programowania. Język programowania Arduino jest oparty na środowisku Wiring i zasadniczo na języku C/C++. Wiring jest środowiskiem OpenSource, które ma ułatwić połączenie elektroniki i programistyki oraz ułatwić projektowanie fizycznych układów.



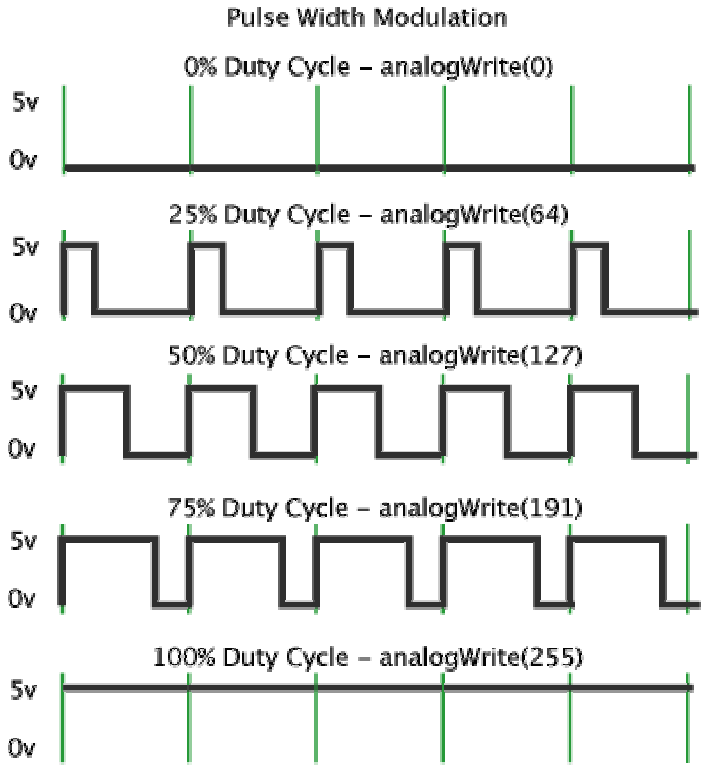
Rys.1. Widok wykorzystanego w ramach ćwiczenia jednego z modeli Arduino (Leonardo)

Arduino Leonardo (Rys.1) jest jedną z najpopularniejszych wersji z serii. Płytka zawiera mikrokontroler ATmega32u4 wyposażony w 20 cyfrowych wejść/wyjść, z czego 7 można wykorzystać jako wyjścia PWM(np. do sterowania silnikami) oraz 12 jako wejścia analogowe. Układ taktowany jest sygnałem zegarowym o częstotliwości 16 MHz, posiada 32 kB pamięci programu Flash oraz 2,5 kB pamięci operacyjnej SRAM.

Specyfikacja płytki:

- Napięcie zasilania: od 7 V do 12 V
- Mikrokontroler: ATmega32u4
  - Maksymalna częstotliwość zegara: 16 MHz
  - Pamięć SRAM: 2,5 kB
  - Pamięć Flash: 32 kB (4 kB zarezerwowane dla bootloadera)
  - Pamięć EEPROM: 1 kB
- Porty I/O: 20
- Wyjścia PWM: 7
- Ilość wejść analogowych: 12
- Interfejsy szeregowe: UART, SPI, I2C, USB
- Zewnętrzne przerwania
- Podłączona dioda LED na pinie 13
- Gniazdo microUSB do programowania
- Złącze DC 5,5 x 2,1 mm do zasilania

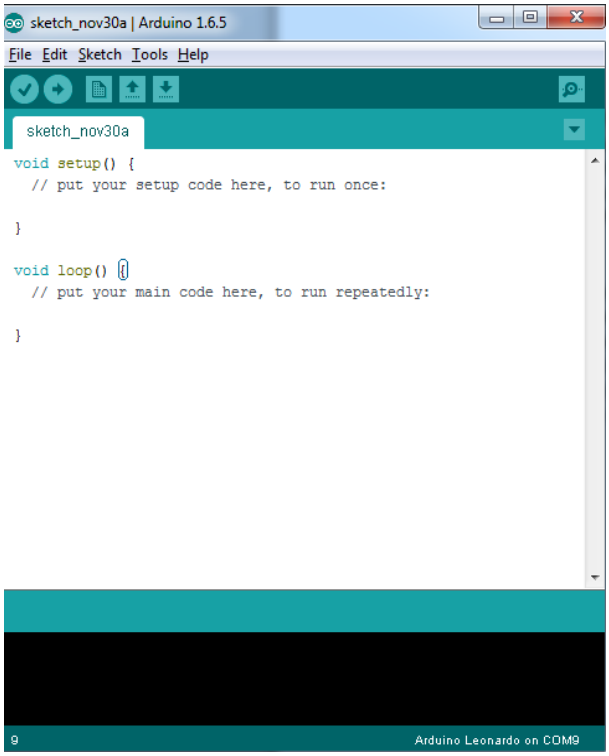
Wyjścia PWM oznaczone są symbolem ~. Symbol ten znajduję się na pinach 3, 5, 6, 9, 10, 11 i 13. PWM to skrót od angielskich słów „Pulse Width Modulation" co oznacza „Modulacja Szerokości Impulsu". Działanie PWM polega na dostarczeniu mniejszej ilości energii elektrycznej do urządzenia w przeciągu jednostki czasu. Skutkiem tego są efekty takie jak regulacja szybkości silnika lub jasności diody LED. Przykład działania jest przedstawiony na rysunku 2.



Rys.2. PWM – zasada działania

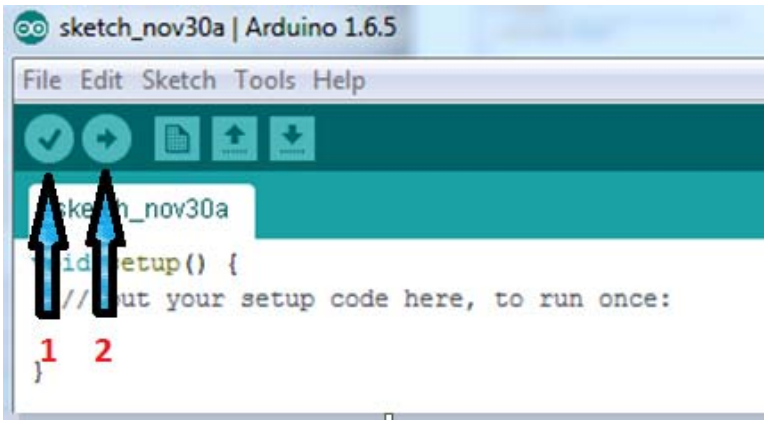
Środowisko Arduino:

Po pierwszym uruchomieniu Arduino IDE pokazuje się okno główne przedstawione na Rys.3. Jak widać środowisko wymaga definicji dwóch funkcji: **void setup()** oraz **void loop()**.



Rys.3. Okno główne środowiska Arduino

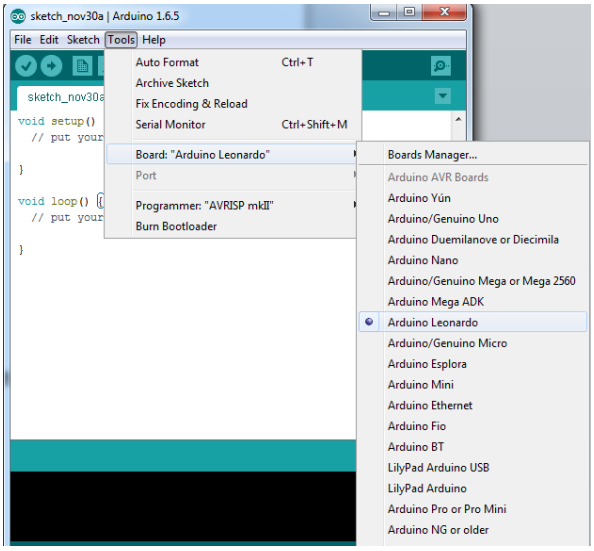
Funkcja setup() to funkcja, która wywoływana jest automatycznie po włączeniu zasilania lub naciśnięciu przycisku RESET. Funkcja loop() wywoływana jest bezpośrednio po funkcji setup() w nieskończonej pętli i podobnie jak ww. funkcja nie zwraca żadnej wartości oraz nie wymaga żadnych parametrów..



Rys.4. Weryfikacja kodu(1). Załadowanie do Arduino(2).

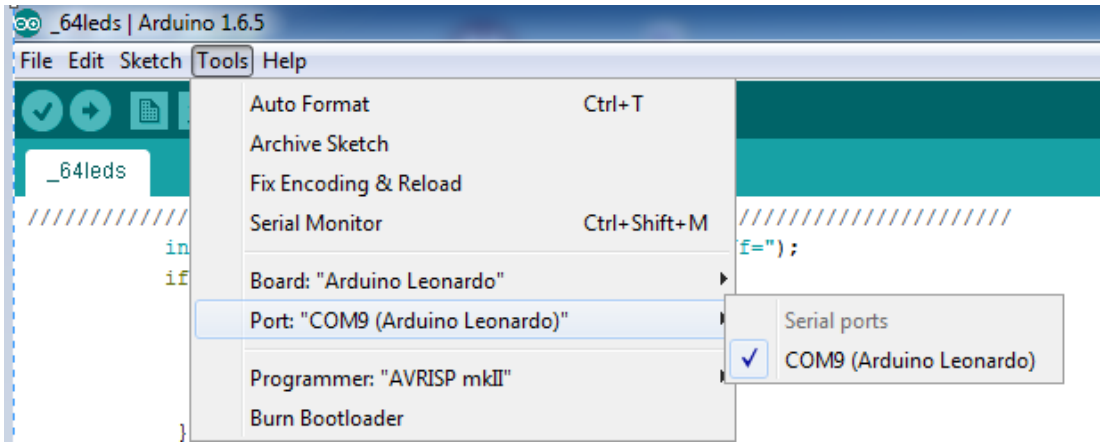
Do załadowania programu do Arduino i zweryfikowania poprawności kodu służy przycisk numer 2 (Rys.4). Przycisk numer 1 (Rys.4) weryfikuje tylko kod.

Przed wgraniem programu do Arduino należy wybrać typ płytki Arduino jaką posiadamy (Rys.5) W opisywanym tu ćwiczeniu wybieramy płytke Arduino Leonardo.



Rys.5. Wybranie odpowiedniej płytki(w ćwiczeniu jest użyta płytka Arduino Leonardo)

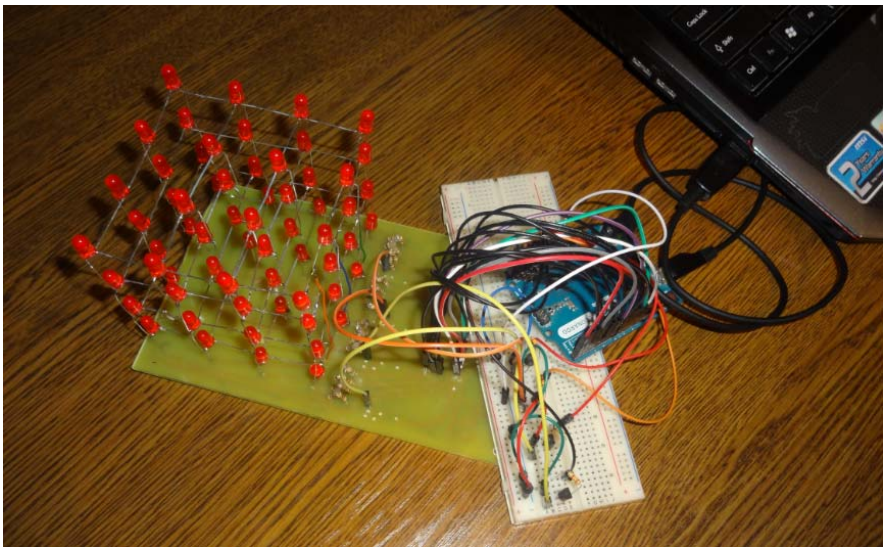
Kolejnym krokiem jest wybór odpowiedniego portu (Rys.6)



Rys.6. Wybranie odpowiedniego portu COM.

3. Opis techniczny i zasada działania sterownika.

Ćwiczenie realizowane jest z wykorzystaniem gotowego układu 64 diod LED połączonych ze sobą wraz z układem sterującym na płytce PCB, która jest podłączana do modułu mikrokontrolera Arduino. Program obsługujący układ sterownika tworzony jest w środowisku Arduino IDE. Moduł Arduino komunikuje się z komputerem za pomocą portu szeregowego. W Arduino IDE należy wybrać „Serial Monitor" by wydawać polecenia np. Serial.println(). Podłączenie elementów (plytka, Arduino, komputer) znajduję się na zdjęciu numer 7.

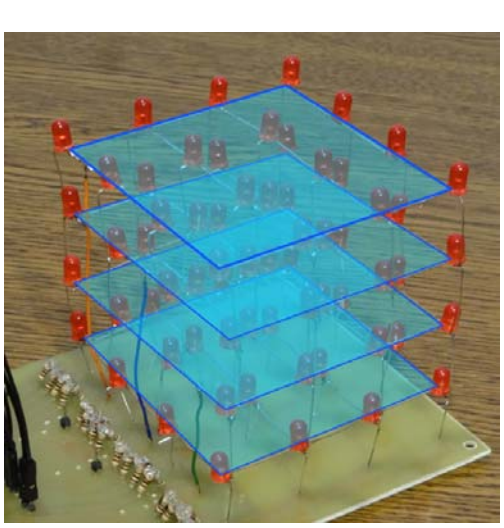
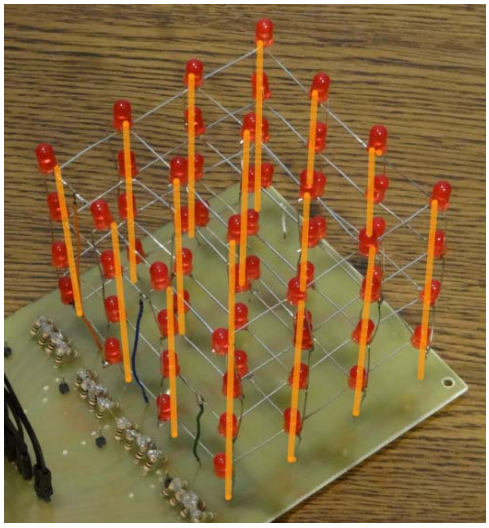
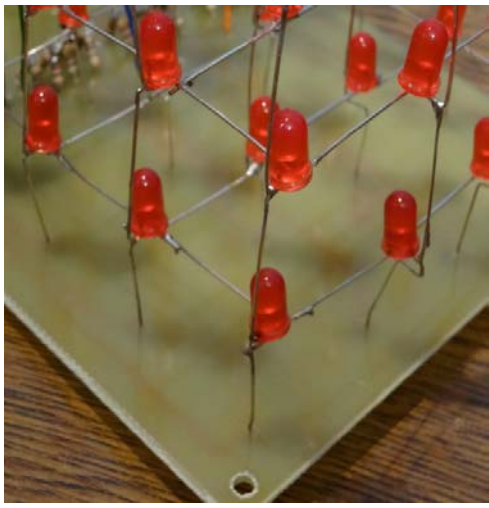


Rys.7.Podłączenie płytki z układem, Arduino i komputera.

3.1 Elementy użyte do wykonania konstrukcji z diod LED wraz z układem sterującym :

- 1x Arduino
- 4x tranzystor NPN BC338
- 4x rezystor 2.4kΩ
- 20x rezystor 147Ω
- 64x dioda LED 5mm czerwona dyfuzyjna
- płytka PCB

3.2 Sposób połączenia diod w konstrukcji kostki:

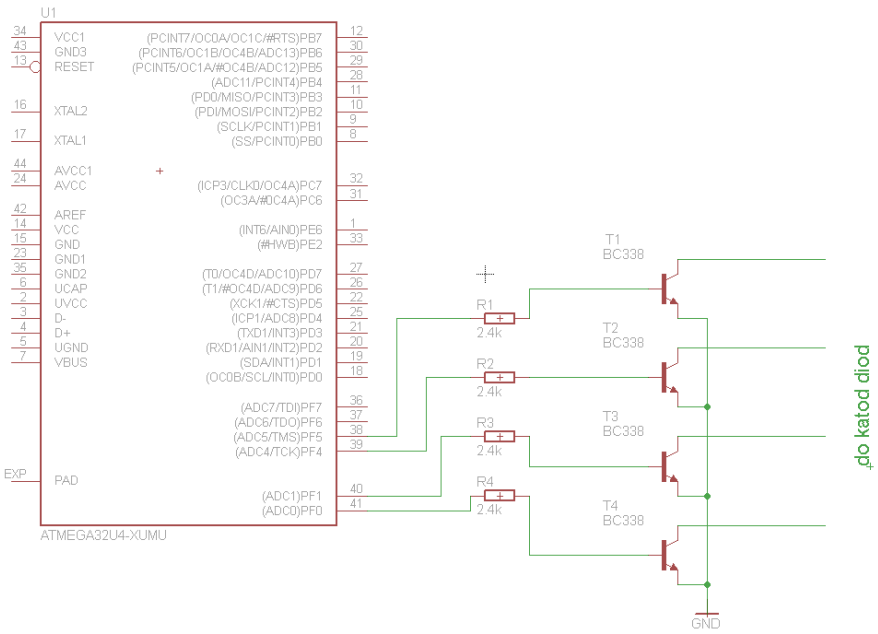
Rys.8. Ilustracja konstrukcji 64 połączonych diod LED. Układ warstw a) Układ kolumn b) oraz połączenie katod i anod diod LED (Pionowo anody, poziomo katody) c)

Kostka led składa się z 16 kolumn (rys.12a) i 4 warstw (rys. 12b). Widoczna na zdjęciu 8 konstrukcja została zrealizowana poprzez połączenie diod które ilustrują schematy ideowe przedstawione na rysunkach 9, 10, 11, 12, 13. Katody każdej z nóżek w warstwie (poziomo) (rys.8a) są połączone ze sobą. Wszystkie anody w kolumnie (pionowo) (rys.8b) również są połączone ze sobą. Na rysunku 8c pokazano z bliska jak podłączone są diody.

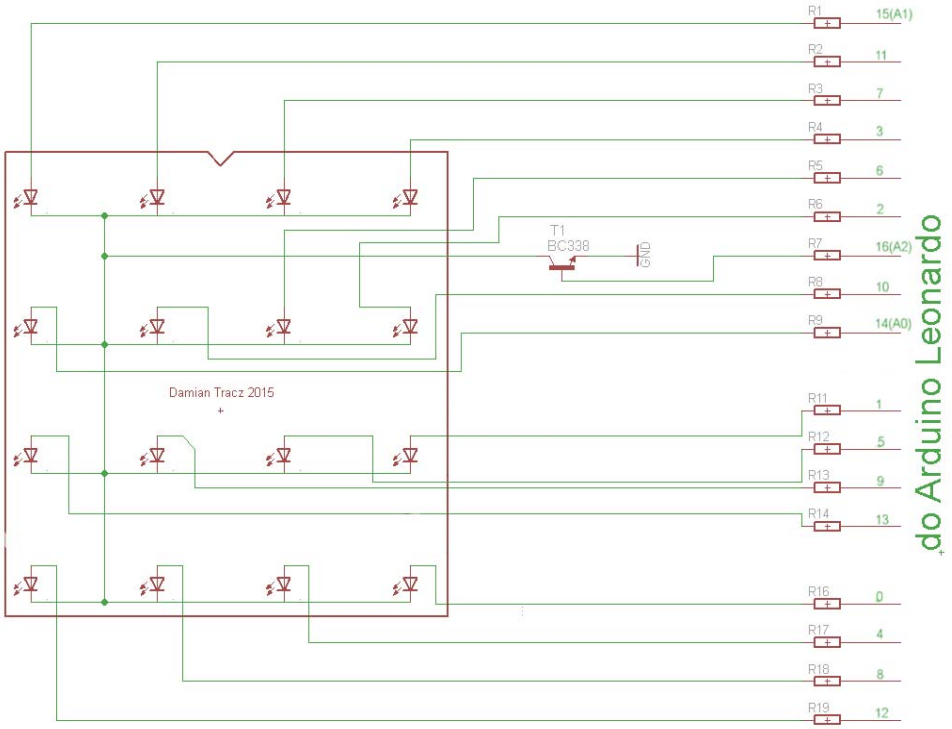
3.3 Układ sterujący włączaniem i wyłączeniem diod LED

Każda z 16 kolumn jest podłączona do płytki Arduino osobnym przewodem do pinów: 0, 1, 2, 3,4 ,5, 6, 7,8,9,10,11,12,13,14,15 poprzez rezystor 147Ω. (Rys.10, 11, 12, 13) .

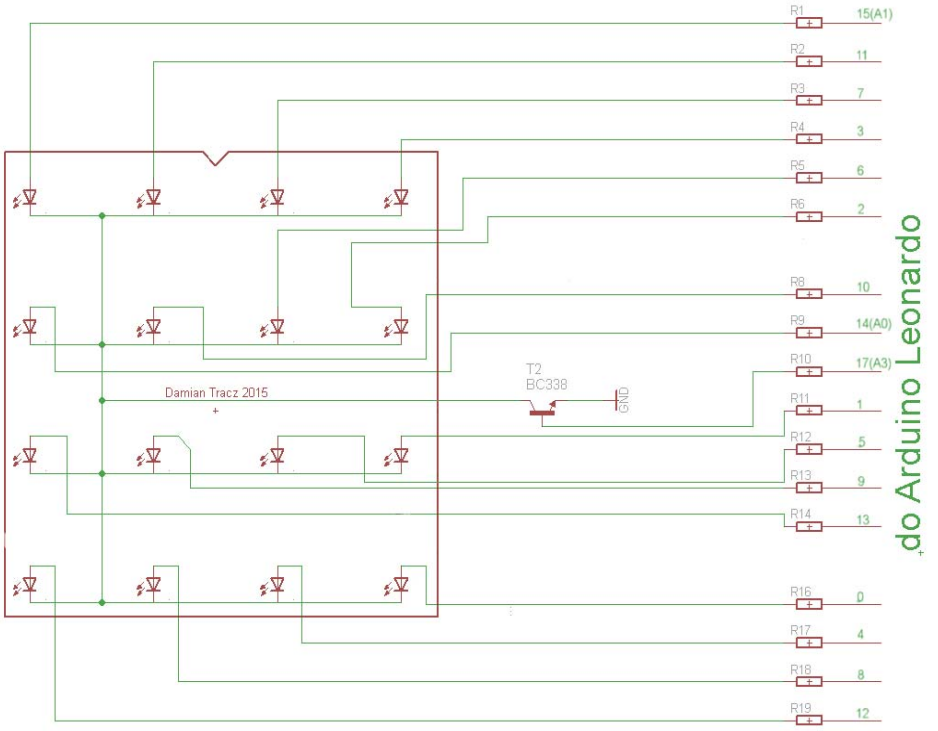
Każda z czterech warstw jest połączoną do kolektora tranzystora NPN który umożliwia włączenie lub wyłączenie warstwy. Emiter tranzystora jest połączony do GND, baza do jednego z pinów w Arduino: 16(A2), 17(A3) ,18(A4) ,19(A5) poprzez rezystor 2.4kΩ (Rys.9) .Tranzystory na rysunkach 10, 11, 12, 13 odpowiadają poszczególnym tranzystorom na rysunku 8. Np. tranzystor T1 na rysunku 8 odpowiada tranzystorowi T1 na rysunku 10. Tranzystory umożliwiają nam łatwe włączanie i wyłączanie całej warstwy. Przy mniejszej ilości diod LED można zrezygnować z tranzystorów. 64 diody pobiera dużo prądu dlatego zastosowane są tranzystory aby zabezpieczyć mikrokontroler przed spaleniem.



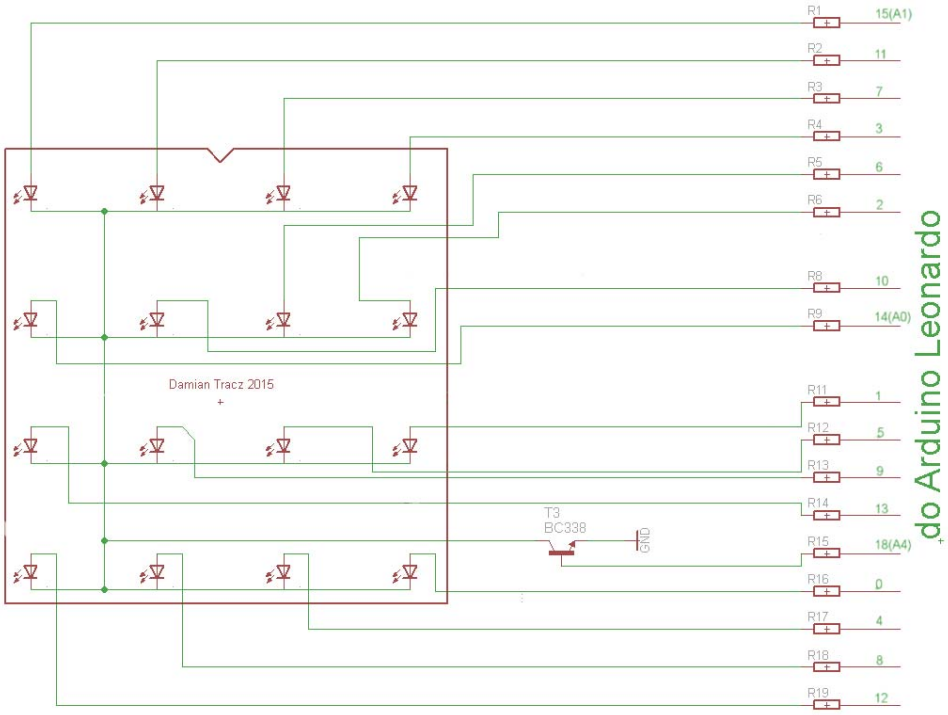
Rys.9. Schemat ideowy układu sterowania włączaniem diod led.



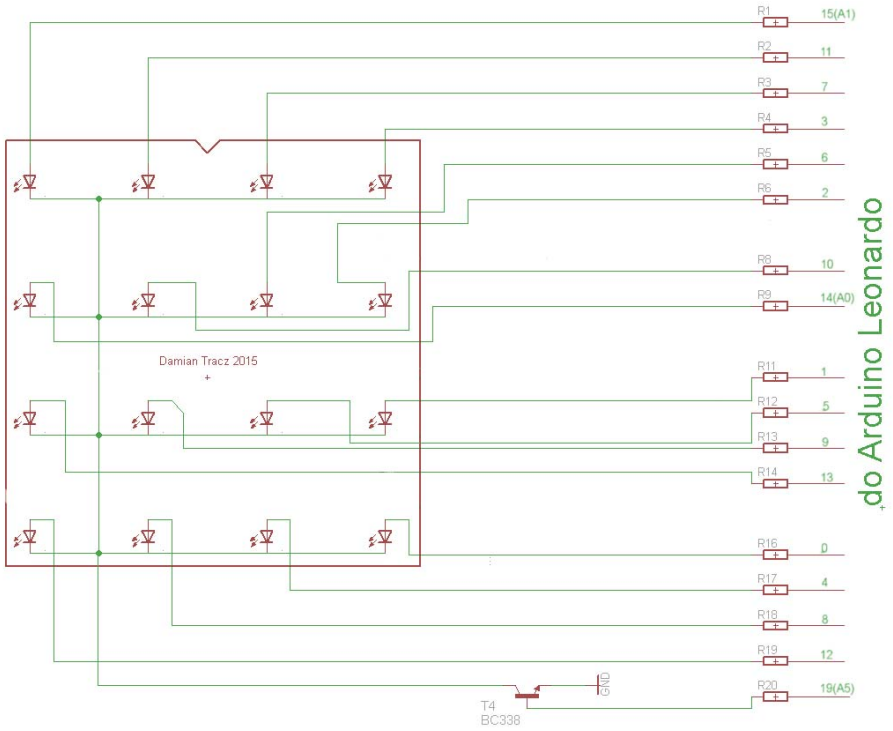
Rys.10. Schemat ideowy podłączenia kolumn i warstwy pierwszej



Rys.11. Schemat ideowy podłączenia kolumn i warstwy drugiej

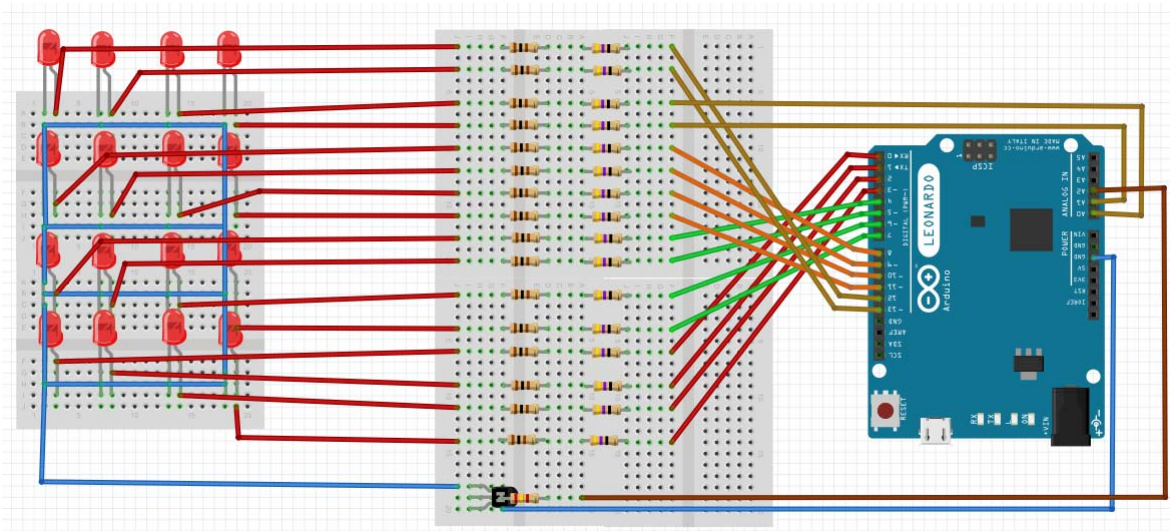


Rys.12. Schemat ideowy podłączenia kolumn i warstwy trzeciej



Rys.13. Schemat ideowy podłączenia kolumn i warstwy czwartej

Poniższy schemat (Rys.14) odpowiada schematowi ideowemu który jest przedstawiony na rysunku 10. Jest tutaj pokazane podłączenie warstwy pierwszej do mikrokontrolera Arduino.



Rys.14. Schemat montażowy płytki Arduino dla warstwy pierwszej kostki LED

3.4 Przykładowe sposoby sterowania kostką LED

Przykład1:
Aby zaświecić wszystkie diody które znajdują się na warstwie 3, trzeba ustawić stan wysoki na pinie 18(A4) oraz ustawić stan wysoki na pinach 0, 1, 2, 3,4 ,5, 6, 7,8,9,10,11,12,13,14,15.

4. Programowanie sterownika

Na samym początku programu dołączone są 2 pliki:

```
#include <LED_Effects.h>
#include <LED_Basic.h>
```

W plikach znajdują się klasy i metody pomocne do sterowania diodami LED.

Tablice **int LEDPin[] ,int layer[]** przechowują numery pinów do jakich podłączona jest kostka led.
Zmienna **String command** przechowuje łańcuch znaków wprowadzonych prze użytkownika przekazanego przez port szeregowy z komputera do Arduino. W dalszej części kodu zostały utworzone dwa obiekty:

```
LED_Basic led_basic(LEDPin, layer);
LED_Effects led_effects(LEDPin, layer);
```

Jako parametry przyjmują tablice z numerami pinów.

W funkcji **setup()** ustawiono szybkość transmisji danych oraz piny Arduino jako wyjście.

```
Serial.begin(9600);

for (int pin=0; pin<=16; pin++)
{
 pinMode( LEDPin[pin], OUTPUT );
}

for(int i = 0; i<4; i++)
{
 pinMode(layer[i], OUTPUT);
}
```

W funkcji **loop()** sprawdzana jest jaka komenda została wpisana przez użytkownika i wywoływana jest odpowiednia akcja. Np. po wpisaniu komendy „blinkAll” w SerialMonitor zostanie wywołana metoda klasy LED_Effects która będzie mrugać wszystkimi diodami LED w odstępach co 50 ms.

```
led_effects.blinkAll(50);
```

Funkcja **returnNumber(String command, String commandSerial)** została specjalnie napisana aby ułatwić wprowadzanie komend z parametrami np. **ledOn=10**. Funkcja zwraca numer który pojawił się po znaku „=”.

Poniżej przedstawiony jest poglądowy kod który przybliży zasadę działania sterownika. Zadaniem poniższego programu jest włączenie i wyłączenie diody LED w odstępach co 1 sekunda.

```
void setup() {
  // ustawienie pinu 13 jako wyjście
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   //włącza diode LED
  delay(1000);              //odczekanie jednej sekundy
  digitalWrite(13, LOW);    //wyłączenie diody LED
  delay(1000);              //odczekanie jednej sekundy
}
```

4. 1 Opis funkcji użytych w kodzie sterownika:

Serial.begin(speed, config) – otwiera port szeregowy. Pierwszy parametr „speed" ustawia prędkość w bitach na sekundę. Najczęściej stosowaną prędkością jest 9600 bitów na sekundę. Dostępne są również inne wartości: 300, 600, 1200, 2400, 4800, 14400, 19200, 28800, 38400, 57600 lub 115200. Drugi opcjonalny argument konfiguruje parametry takie jak bit stopu i bit kontroli parzystości. Standardowo jest 8 bitów danych, bez bitu parzystości, jeden bit stopu.

```
Serial.begin(9600);
```

Serial.available() – zwraca liczbę bajtów(znaków) dostępnych do odczytu z portu szeregowego.

```
if (Serial.available() > 0) {
        // odczytaj przychodzący bajtów:
 }
```

Serial.readString() – czyta znaki z buforu portu szeregowego do łańcucha znaków.

```
String command = Serial.readString();
```

Serial.print() – Wyświetla dane odebrane z portu szeregowego(Arduino) jako tekst ASCII.

```
Serial.print("Arduino");
```

pinMode(pin, mode) - konfiguruje określony pin do zachowywania się jako wejście lub wyjście.

```
pinMode(13, INPUT);       // ustawia cyfrowy pin 13 jako wejście
pinMode(13, OUTPUT);   //ustawia cyfrowy pin 13 jako wyjście
```

delay(ms) – zatrzymuje wykonywane programu na określony czas (w milisekundach).

delay(1000); //zatrzymanie programu na jedną sekundę

digitalWrite(pin, value) – ustawia stań wysoki(HIGH) lub niski(LOW) na podanym pinie. HIGH oraz LOW są to stałe w języku Arduino. Przyjmują następujące wartości 1 dla HIGH i 0 dla LOW.

digitalWrite(1, HIGH); //ustawia stan wysoki dla pinu 1
digitalWrite(1, LOW); //ustawia stan niski dla pinu 1

5. Przebieg ćwiczenia:

1.Skopiuj 2 katalogi z folderu _64leds/Libraries do katalogu D:\Program Files (x86)\Arduino\libraries

W katalogu D:\Program Files (x86)\Arduino\libraries\LED_Basic znajdują się pliki LED_Basic.cpp oraz LED_Basic.

2. Dodaj metodę do klasy LED_Basic która:
   - włączy wszystkie 64 diody led.
     - turnOnAll()
   - wyłączy wszystkie 64 diody led
     - turnOffAll()
   - włączy jedną kolumnę
     - turnOnColumn(int column)
   - wyłączy jedną kolumnę
     - turnOffColumn(int column)
   - zaświecić tylko jedną diodę led podaną przez użytkownika
     - turnOnOneLed(int led)
   - włączy jedna warstwę (zaświeci wszystkie diody w poziomie na wybranej warstwie)
     - turnOnOneLayer(int layer)
   - wyłączy jedną warstwę
     - turnOffOneLayer(int layer)

3. W programie głównym dodaj odpowiednie biblioteki. Stwórz obiekt klasy LED_Basic.
4. Wywołaj metody klasy LED_Basic w odpowiednich miejscach programu.

6. Literatura:

https://pl.wikipedia.org/wiki/Arduino
https://www.arduino.cc/en/Main/ArduinoBoardLeonardo
http://botland.com.pl/arduino-moduly-glowne/1213-arduino-leonardo.html
https://pl.wikipedia.org/wiki/Obw%C3%B3d_drukowany
http://starter-kit.nettigo.pl/2012/01/co-to-jest-pwm/
https://www.arduino.cc/en/Tutorial/PWM
http://www.plociennik.info/index.php/podstawy-jezyka
https://www.arduino.cc/en/Reference/Serial
https://www.arduino.cc/en/Reference/PinMode
https://www.arduino.cc/en/Reference/Delay
https://www.arduino.cc/en/Reference/DigitalWrite