

Projekt systemu wbudowanego



2019 r.

1. Wstęp

→ Tematyka projektu

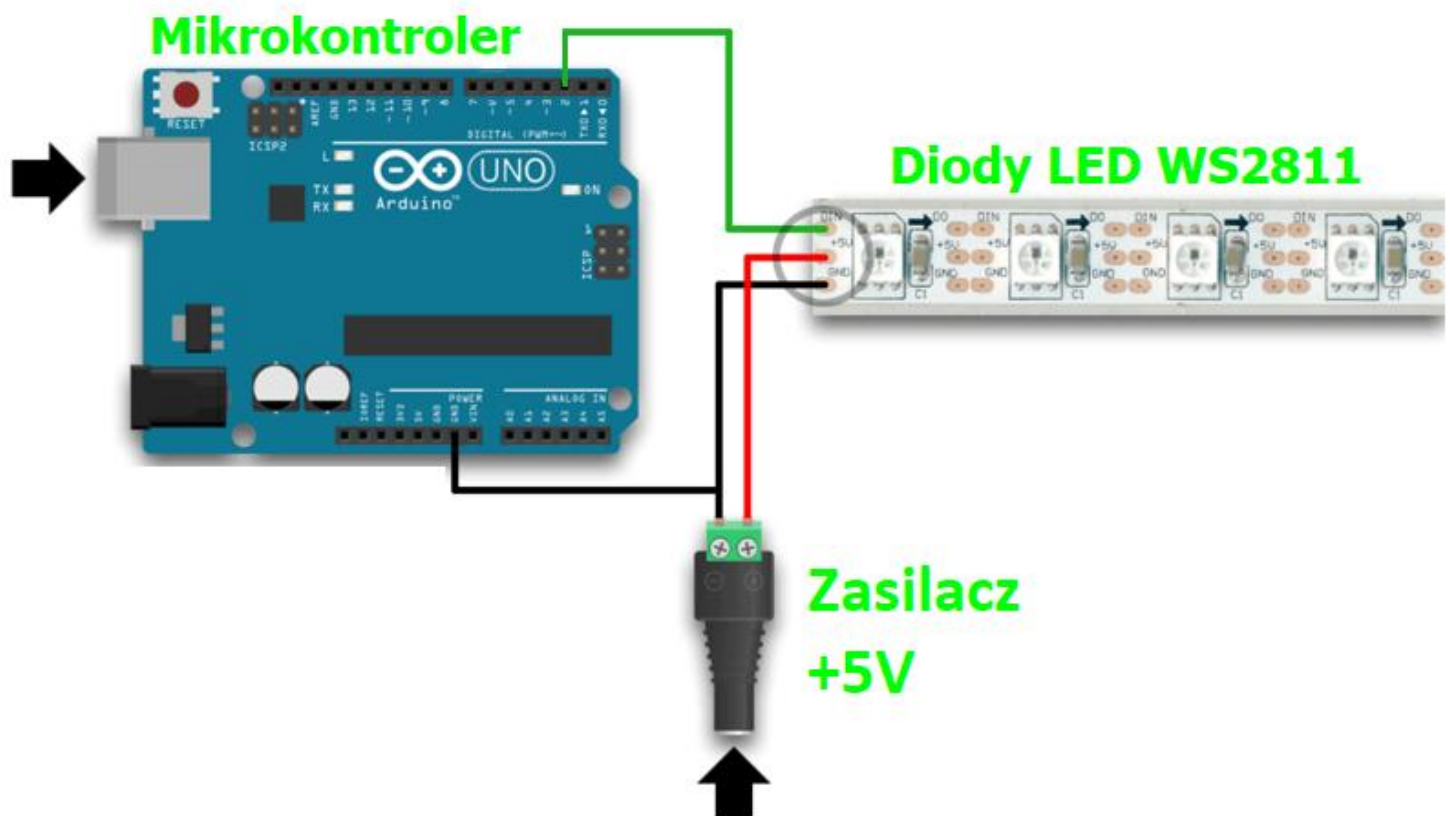
Celem mojego projektu było stworzenie sterownika opartego na Arduino Yun do obsługi taśmy LED na układzie WS2811.

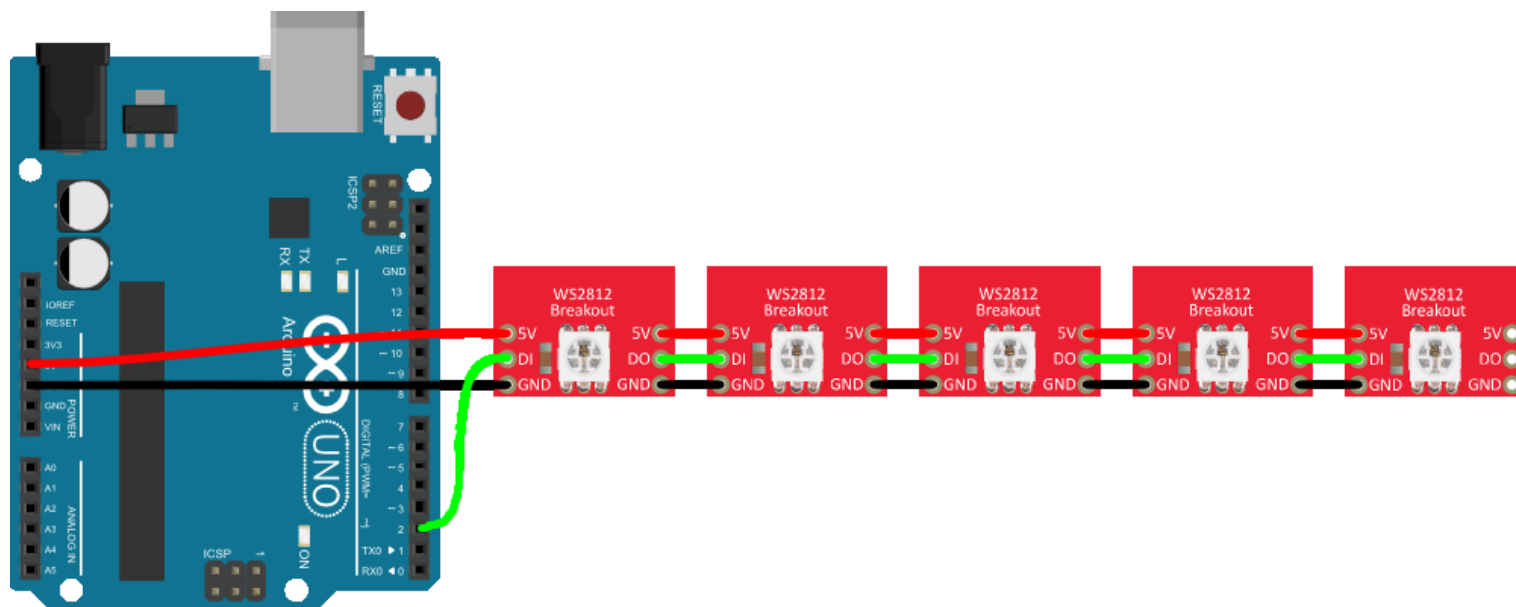
Do jego budowy wykorzystałam programowalny sterownik logiczny Arduino Yun, diody LED WS2811, zasilacz +5V oraz przewody połączeniowe. Program napisany został w języku C++ przy użyciu programu Arduino IDE.

→ Opis projektu

W ramach projektu stworzyłam sterownik, którego zadaniem jest kontrolowanie łańcucha układów WS2811 odpowiadających za świecenie diód. Sterownik umożliwia stworzenie efektów wizualnych, takich jak animacje i przejścia kolorów.

→ Schemat systemu wbudowanego





W projekcie sterujemy 150 diodami, używając tylko jednego pinu (wyjścia) mikrokontrolera.

2. Komponenty

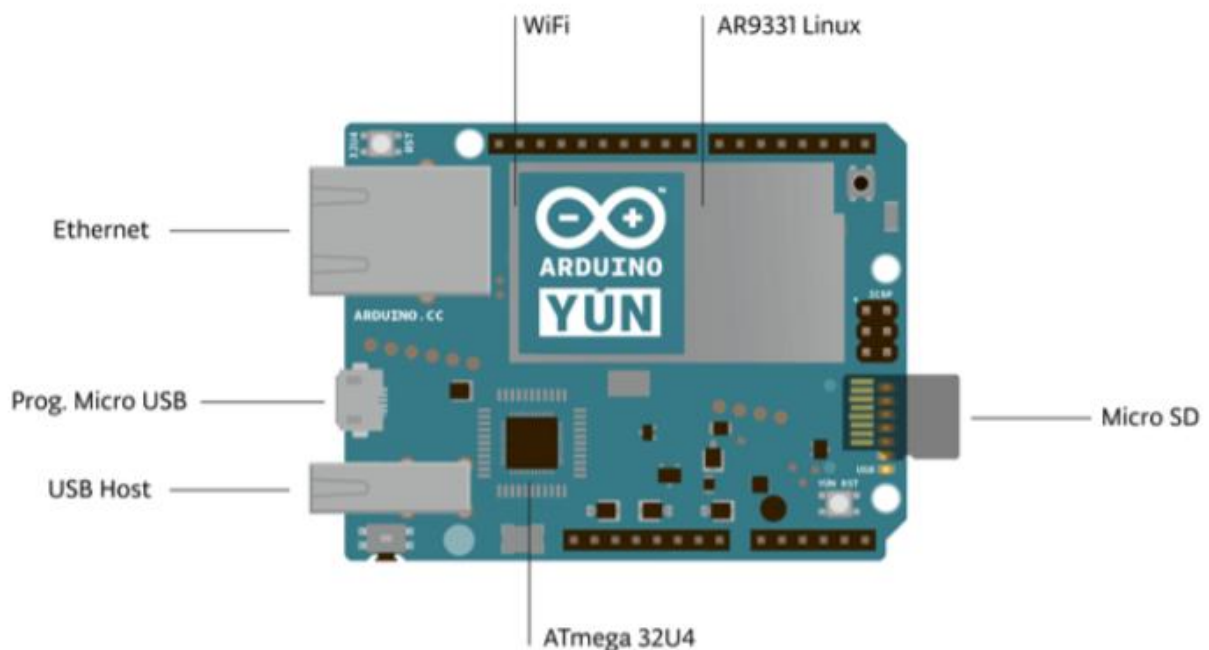
Do budowy tego systemu wbudowanego wykorzystałam programowalny sterownik logiczny Arduino Yun, diody LED WS2811, zasilacz +5V oraz przewody połączeniowe.

➔ Arduino Yun

Opis: Yun to połączenia Arduino Leonardo i kontrolera Atheros AR9331 z obsługą WiFi. Płytkę 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.

Specyfikacja

- Napięcie zasilania: 5 V
- Mikrokontroler AVR: 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 (kanały przetwornika A/C o rozdzielczości 10 bitów)
 - Interfejsy szeregowo: UART, SPI, I2C
 - Zewnętrzne przerwania
- Procesor Linux: Atheros AR9331
 - Architektura: MIPS 400 MHz
 - Napięcie zasilania: 3,3 V
 - Ethernet IEEE 802.3 10/100Mbit/s
 - WiFi IEEE 802.11b/g/n
 - Pamięć RAM: 64 MB DDR2
 - Pamięć Flash: 16 MB
- Gniazdo USB Typ-A 2.0
- Czytnik kart Micro-SD



➔ Diody LED WS2811

Opis: LED zawiera zintegrowany sterownik umożliwiający sterowanie przy pomocy jednej linii podłączonej do mikrokontrolera np. Arduino, STM32 lub Raspberry Pi. Ponieważ każda dioda posiada indywidualny adres, to na jednej magistrali można podłączyć wiele elementów.

Podłączenie: Dioda LED posiada cztery wyprowadzenia.

Pin	Opis
5V	Napięcie zasilania 5 V.
GND	Masa układu.
DIN	Wejście sterujące, które należy podłączyć do mikrokontrolera lub wcześniejszej diody w łańcuchu.
DOUT	Opcjonalne wyjście do podłączenia kolejnej diody.

Specyfikacja

- Napięcie zasilania: 5 V
- Pobór prądu I_f : do 50 mA
- Możliwość wyboru barwy z 24-bitowej palety
- Sterowana cyfrowo poprzez interfejs 1-wire z możliwością podłączenia wielu urządzeń na jednej linii
- Posiada indywidualny adres urządzenia

Protokół komunikacyjny

Diody WS2811 są sterowane przy pomocy prostego, szybkiego protokołu komunikacyjnego.

Barwa przesyłana jest w trzech liczbach RGB (czerwony-zielony-niebieski). Jasność każdej barwy jest zakodowana w serii przesyłanych 8-bitów, gdzie najbardziej znaczący jest przesyłany jako pierwszy. Oznacza to, że aby zakodować pełny kolor, należy przesłać 24 bity. Pierwsza sekwencja dotyczy diody umieszczonej najbliżej mikrokontrolera, a kolejna drugiej w łańcuchu itd.

Zastosowany protokół pozwala na szybką aktualizację stanów wielu diod LED w łańcuchu. Utworzona przez firmę Pololu biblioteka Arduino, pozwala zmienić stan 30 diod w ciągu 1,1 ms. Po ustawieniu odpowiedniej barwy, każda dioda będzie ją utrzymywać do momentu przesłania nowej sekwencji lub do wyłączenia zasilania.

3. Kod programu

Program napisany został w języku C++ przy użyciu programu Arduino IDE.

Poniżej przedstawiony został kod wraz z komentarzami:

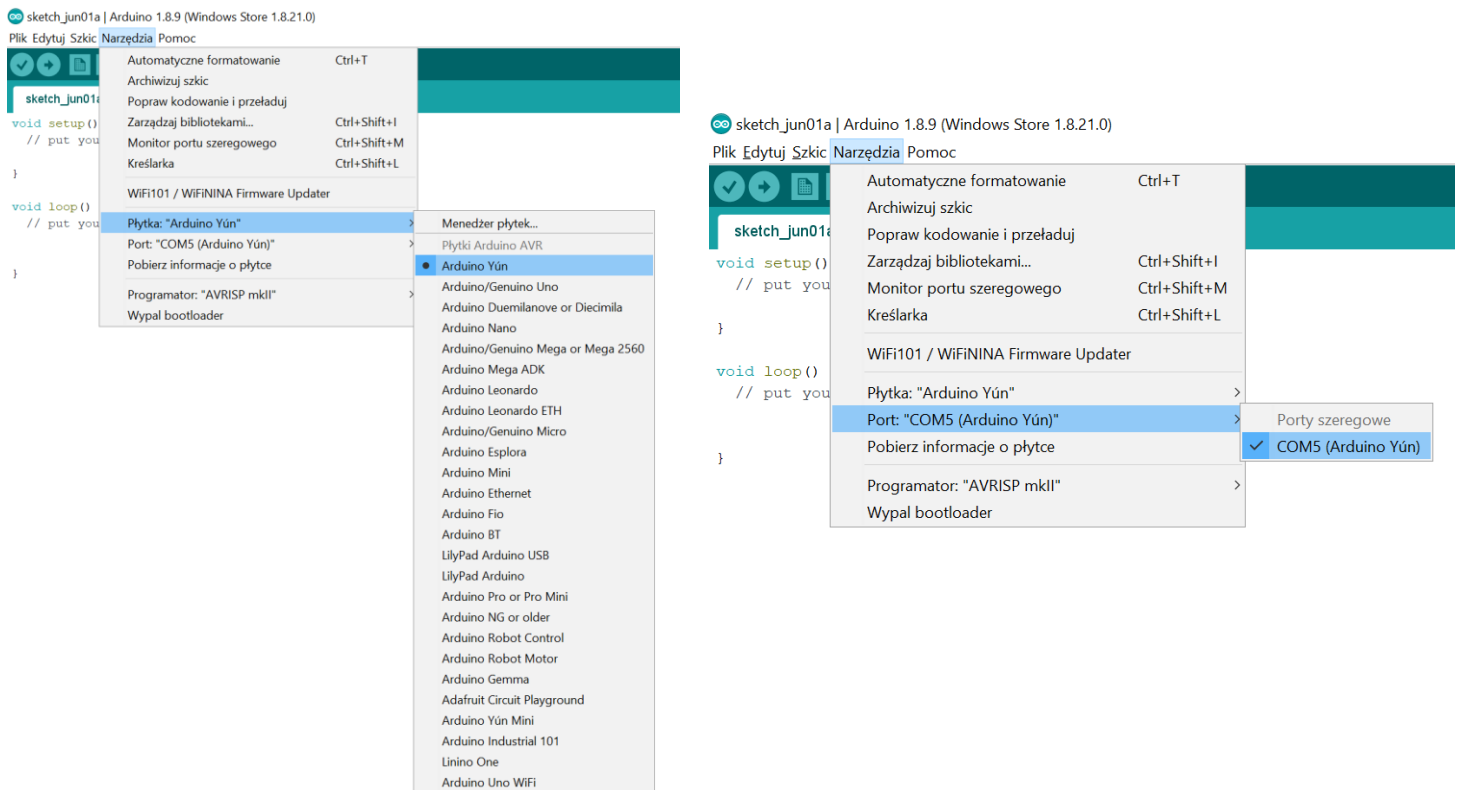
```
1
2  #include <NeoPixelBus.h>
3
4  const uint16_t PixelCount = 151; // stała z liczbą ilości diod w taśmie
5  const uint8_t PixelPin = 2; // wyjście, gdzie podłączamy pin transmisji danych
6
7  // element biblioteki dla prędkości 800 Kbps
8  NeoPixelBus<NeoGrbFeature, Neo800KbpsMethod> strip(PixelCount, PixelPin);
9
10 void setup()
11 {
12     // inicjalizacja biblioteki
13     strip.Begin();
14     strip.Show();
15 }
16
17
18 void loop()
19 {
20
21     // ustaw kolorowy
22     delay(1000); //poczekaj
23     for(int i=151 ;i>=7; i--)
24     {
25         //taśma randomowa
26         strip.SetPixelColor(i+6, RgbColor(random(5, 255), random(5, 255), random(5,
27             255)));
28
29         //przód który przewodzi zmieniając kolor diod
30         strip.SetPixelColor(i+5, RgbColor(4, 249, 247));
31         strip.SetPixelColor(i+4, RgbColor(4, 249, 247));
32         strip.SetPixelColor(i+3, RgbColor(4, 249, 247));
33         strip.SetPixelColor(i+2, RgbColor(4, 249, 247));
34         strip.SetPixelColor(i+1, RgbColor(4, 249, 247));
35
36         strip.Show();      delay(60); //show - pokaz
37     }
38     // ustaw niebieski
39     delay(900);
40     for(int i=0 ;i<600; i++)
41     {
42         //taśma różowa
43         strip.SetPixelColor(i-6, RgbColor(0, 98, 238)); //niebieski
44
45         //iskierki
46
47         strip.SetPixelColor(i-174, RgbColor(173, 0, 134)); //różowy
48         strip.SetPixelColor(i-175, RgbColor(173, 0, 134));
49         strip.SetPixelColor(i-176, RgbColor(0, 98, 238));
50
51         strip.SetPixelColor(i-204, RgbColor(173, 0, 134));
52         strip.SetPixelColor(i-205, RgbColor(173, 0, 134));
53         strip.SetPixelColor(i-206, RgbColor(0, 98, 238));
54
55         strip.SetPixelColor(i-234, RgbColor(173, 0, 134));
56         strip.SetPixelColor(i-235, RgbColor(173, 0, 134));
57         strip.SetPixelColor(i-236, RgbColor(0, 98, 238));
58
59         strip.SetPixelColor(i-384, RgbColor(173, 0, 134));
60         strip.SetPixelColor(i-385, RgbColor(173, 0, 134));
61         strip.SetPixelColor(i-386, RgbColor(0, 98, 238));
62
63         strip.SetPixelColor(i-414, RgbColor(173, 0, 134));
64         strip.SetPixelColor(i-415, RgbColor(173, 0, 134));
65         strip.SetPixelColor(i-416, RgbColor(0, 98, 238));
66     }
```

```

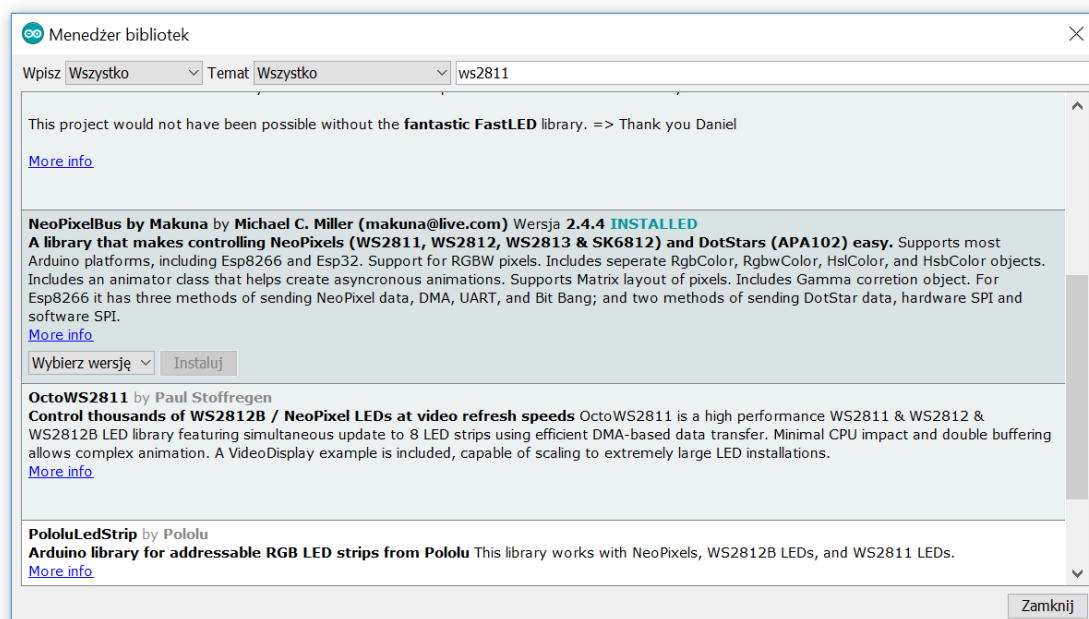
65     strip.SetPixelColor(i-444, RgbColor(173, 0, 134));
66     strip.SetPixelColor(i-445, RgbColor(173, 0, 134));
67     strip.SetPixelColor(i-446, RgbColor(0, 98, 238));
68
69     strip.Show();    delay(21);
70 }
71
72
73 // gaszenie diod
74 delay(500);
75 int zm = 0;
76 for(int i=151 ;i>75; i--)
77 {
78     strip.SetPixelColor(i, RgbColor(0, 0, 0)); //przod
79     strip.SetPixelColor(zm, RgbColor(0, 0, 0)); //tyl
80     zm++;
81     strip.Show();    delay(30);
82 }
83
84 // animacja
85
86 delay(800);
87 for(int j=0; j<5; j++){
88     int zm1=0;
89     int zm2=1;
90     for(int i=0 ;i<150; i++)
91     {
92         strip.SetPixelColor(zm1, RgbColor(173, 0, 134)); //rozowy
93         strip.SetPixelColor(zm2, RgbColor(0, 0, 0)); //zgaszony
94         zm1=zm1+2;
95         zm2=zm2+2;
96     }
97     strip.Show(); delay(500);
98
99     zm1=0;
100    zm2=1;
101
102    for(int i=0 ;i<150; i++)
103    {
104        strip.SetPixelColor(zm2, RgbColor(173, 0, 134)); //rozowy
105        strip.SetPixelColor(zm1, RgbColor(0, 0, 0)); //zgaszony
106        zm1=zm1+2;
107        zm2=zm2+2;
108    }
109    strip.Show(); delay(500);
110 }
111
112
113
114 // gaszenie diod
115
116 for(int i=151 ;i; i--)
117 {
118     strip.SetPixelColor(i, RgbColor(0, 0, 0)); //zgas
119 }
120
121 strip.Show();
122
123
124
125
126
127 }
128

```

W pierwszej kolejności musiałam ustawić odpowiednią płytkę oraz port:

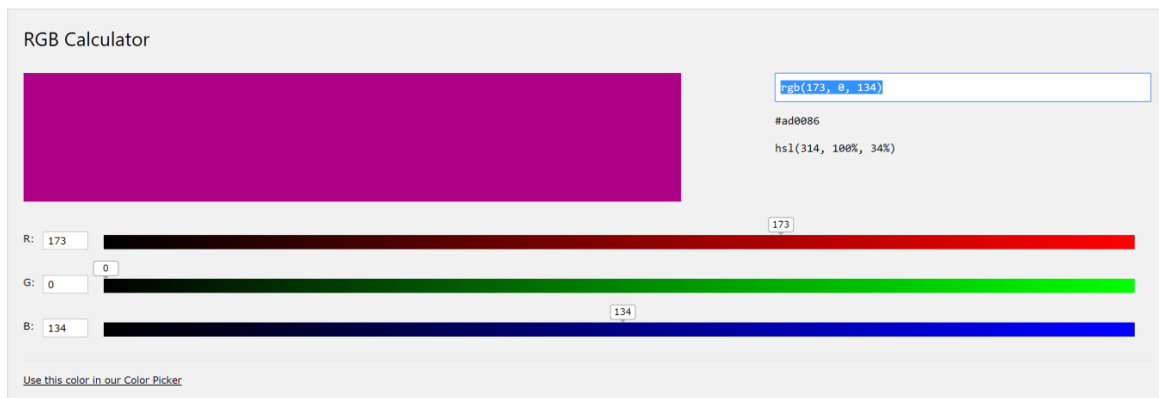


Dodałam również następującą bibliotekę - NeoPixelBus, niezbędną do komunikacji z diodami:



Do wybrania kolorów użyłam kalkulatora RGB dostępnego w Internecie:

RGB Calculator



rgb(173, 0, 134)

#ad0086

hsl(314, 100%, 34%)

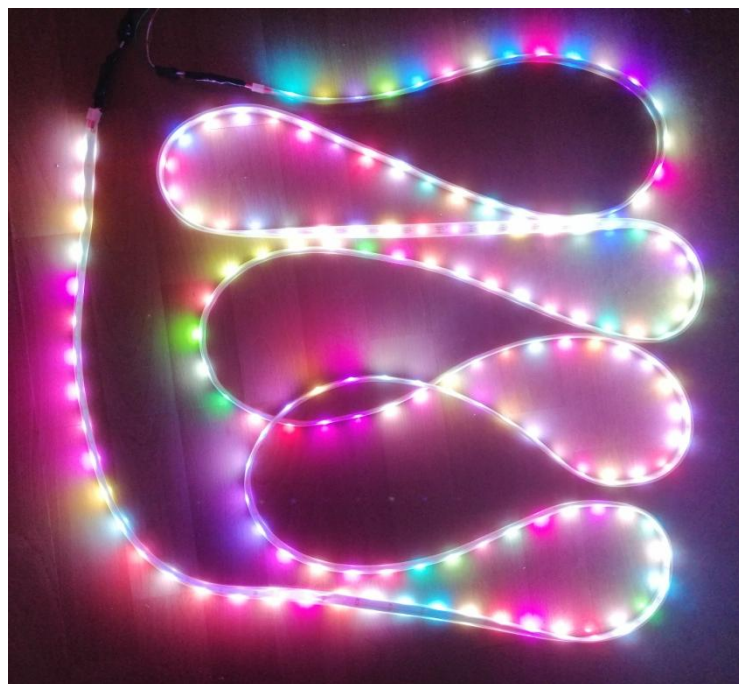
R: 173

G: 0

B: 134

[Use this color in our Color Picker](#)

4. Efekt końcowy



Efekt końcowy



5. Wnioski

Środowisko Arduino umożliwia szybkie i łatwe budowanie prototypów urządzeń elektronicznych. Doskonale sprawdza się przy testowaniu różnych rozwiązań bez potrzeby budowania specjalistycznych układów.