



**UNIWERSYTET OPOLSKI
WYDZIAŁ MATEMATYKI, FIZYKI I INFORMATYKI
Instytut Matematyki I Informatyki**

PRACA INŻYNIERSKA

Dawid Bugajski

**STEROWANY MIKROKONTROLEREM WYSWIETLACZ LED -
PROJEKT I IMPLEMENTACJA**

**MICROCONTROLLER BASED LED DISPLAY -
DESIGN AND IMPLEMENTATION**

Praca wykonana w Zakładzie Informatyki pod kierunkiem
dra Jacka Iwańskiego

Opole 2017

Spis treści

Streszczenie.....	3
Abstract.....	3
1. Wprowadzenie.....	4
2. Charakterystyka projektu.....	6
2.1. Cel projektu.....	6
2.2. Technologia wytworzenia.....	6
2.2.1 Języki programowania.....	6
2.2.2 Środowisko deweloperskie.....	7
2.2.2.1 Konfiguracja środowiska programistycznego Arduino IDE.....	7
2.2.2.2 Konfiguracja środowiska programistycznego Java.....	8
3. Przykłady istniejących rozwiązań.....	9
4. Wymagania.....	11
4.1. Wymagania niefunkcjonalne.....	11
4.2. Wymagania funkcjonalne.....	11
5. Projekt sprzętu.....	12
5.1. Wybór komponentów elektronicznych.....	12
5.2. Projekt i wykonanie modułu wyświetlacza.....	14
5.3. Projekt i wykonanie modułu napędowego wyświetlacza.....	16
6. Projekt oprogramowania.....	18
6.1. Model opracowanego systemu informatycznego.....	18
6.2. Oprogramowanie mikrokontrolera wyświetlacza.....	20
6.3. Oprogramowanie kostki NXT.....	22
6.4. Oprogramowanie PC.....	23
7. Dokumentacja użytkownika.....	24
7.1. Obsługa oprogramowania PC.....	24
7.2. Obsługa oprogramowania Arduino.....	24
8. Podsumowanie.....	26
9. Bibliografia.....	27
9.1. Zasoby internetowe.....	27
9.2. Spis rysunków.....	27
Zawartość płyty CD.....	28

Streszczenie

Sterowany mikrokontrolerem wyświetlacz LED – projekt i implementacja

Tematem pracy inżynierskiej jest projekt i implementacja urządzenia służącego do wyświetlania obrazów za pomocą pojedynczego paska diod LED wykorzystując zjawisko bezwładności oka ludzkiego. Praca zawiera projekt sprzętu oraz projekt oprogramowania służącego do projektowania oraz wyświetlania zaprojektowanych obrazów. W projekcie sprzętu zawarto opis użytych komponentów, schematy urządzenia i sposób działania. W części projektu oprogramowania dokonano specyfikacji wymagań, przedstawiono model opracowanego systemu i ważne elementy oprogramowania.

Słowa kluczowe: systemy wbudowane, mikroprocesory, systemy hybrydowe, trójwymiarowe wyświetlacze

Abstract

Microcontroller based LED display – project and implementation

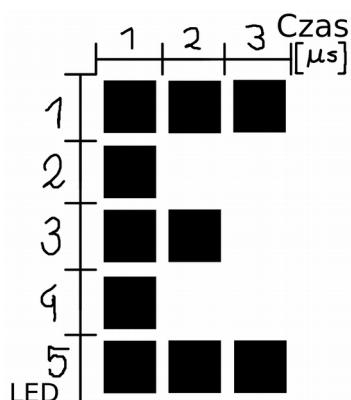
The subject of engineering work is the design and implementation of the device used to display images using single LED strip based on the phenomenon of inertia of human eye. The thesis contains a design of the device and the project of the software which serves in order to design and display those pictures. The design of the device presents the description of the components used, schemes of the device and the way in which it works. The part of the project of software presents specifications of the requirements. It also depicts the model of the software and its important elements.

Keywords: integrated circuits, microprocessors, hybrid systems, three-dimensional displays

1. Wprowadzenie

Dzięki współczesnym technologiom istnieje wiele sposobów przekazywania informacji. Jednym ze sposobów przekazywania informacji są wyświetlacze obrazów oparte na diodach LED. Często wykorzystywaną techniką w tego typu urządzeniach jest multipleksowanie obrazu. Polega ono na tym, że cały obraz nie jest wyświetlany natychmiast, lecz częściowo, co pozwala na transmisję z użyciem jednej szyny transmisyjnej.

Ciekawą odmianą multipleksowania jest wykorzystanie właściwości ludzkiego oka polegającej na tym, że wykazuje ono pewną bezwładność. Jeśli w danej przestrzeni i określonej chwili czasu zadbane o to, żeby w tym samym momencie została odpowiednio wyświetlona ta sama sekwencja diod, możemy uzyskać dwuwymiarowy obraz, co pokazano na rysunku 1.



Rys. 1. Dwuwymiarowy obraz uzyskany za pomocą ustalonych sekwencji diod w określonych jednostkach czasu.

Sposób ten znany i wykorzystywany jest w technikach video, telewizjach oraz przy multipleksowanych wyświetlaczach LED. Zaletą takiego wyświetlacza jest znaczna oszczędność liczby diod potrzebnych do wyświetlenia obrazu, natomiast wadą jest konieczność zaprojektowania urządzenia służącego do przesuwania paska diod.

W mojej pracy chciałbym zająć się odmianą, gdzie zamiast kilku multipleksowanych pasków znajduje się jeden, umieszczony na silniczku,

którego stan jest odpowiednio ustawiany w określonych momentach obrotowych, co daje możliwość wyświetlenia całego obrazu.

Praca została podzielona na pięć rozdziałów.

W pierwszym rozdziale przedstawiono charakterystykę projektu, technologie wytworzenia.

Rozdział drugi mówi o zastosowanych językach programowania oraz istniejących przykładach rozwiązań.

Rozdział trzeci skupia się na wymaganiach zgodnie z zasadami inżynierii oprogramowania.

W rozdziale czwartym przedstawiono projekt sprzętu i oprogramowania.

Pracę kończy rozdział poświęcony dokumentacji użytkownika.

2. Charakterystyka projektu

2.1. Cel projektu

Celem projektu jest stworzenie urządzenia umożliwiającego wyświetlanie prostych obrazów z wykorzystaniem diod LED. Technika umożliwiająca tego typu wyświetlanie nosi nazwę „Persistence Of Vision”, czyli utrzymanie wizji.

2.2. Technologia wytworzenia

2.2.1 Języki programowania

Oprogramowanie składa się z dwóch części. Część pierwsza została umieszczona bezpośrednio na mikrokontrolerze. Druga natomiast znajduje się na komputerze wspierającym oprogramowanie Java.

Oprogramowanie mikrokontrolera wyświetlacza zostało napisane w języku C.

Aplikacja służąca do projektowania obrazów została napisana z wykorzystaniem języka programowania Java. Takie rozwiązanie umożliwia użytkowanie aplikacji na każdym systemie operacyjnym wspierającym uruchamianie programów z pomocą maszyny wirtualnej JVM.

Jedynym minusem jest konieczność uruchomienia maszyny wirtualnej JVM przy każdym włączeniu programu, co zwiększa ilość zużytej przez program pamięci RAM z kilkunastu do kilkudziesięciu megabajtów.

Znaczącym plusem natomiast jest możliwość uruchomienia tego samego pliku wykonywalnego aplikacji zarówno na systemie operacyjnym Windows jak i Linux czy też MacOSX.

2.2.2 Środowisko deweloperskie

2.2.2.1 Konfiguracja środowiska programistycznego Arduino IDE.

Program umieszczony bezpośrednio na mikrokontrolerze wyświetlacza został napisany, skompilowany oraz wgrany za pomocą oprogramowania Arduino IDE w wersji 1.8.2.

Arduino IDE standardowo zaopatrzone jest w bibliotekę **Wire**, która została wykorzystana do komunikacji między poszczególnymi komponentami wyświetlacza. Jedyna rzecz, którą należy zrobić po zainstalowaniu, to skonfigurowanie programu pod obsługę mikrokontrolerów Atmega8.

W tym celu po wyłączeniu programu należy udać się do ustawień programu (z poziomu menu „**Plik – Preferencje**” lub kombinacja klawiszy „**Ctrl+,**”), gdzie w polu „*Dodatkowe adresy URL do menedżera płytka*” należy umieścić ten adres:

[**https://raw.githubusercontent.com/sleemanj/optiboot/master/dists/packa
ge_gogo_diy_atmega8_series_index.json**](https://raw.githubusercontent.com/sleemanj/optiboot/master/dists/package_gogo_diy_atmega8_series_index.json)

a następnie kliknąć „OK”. Teraz tylko wystarczy wybrać opcję z menu „**Narzędzia → Płytki → Menedżer płytka...**”. W okienku, które się pojawi można znaleźć konfiguracje wielu urządzeń. Nas interesuje jedynie biblioteka **DIY ATmega**, którą można znaleźć poprzez wpisanie *atmega* w polu wyszukiwania. Po zainstalowaniu powinniśmy mieć możliwość wybrania płytki **Atmega8/A** z poziomu menu „**Narzędzia → Płytki**”.

2.2.2.2 Konfiguracja środowiska programistycznego Java.

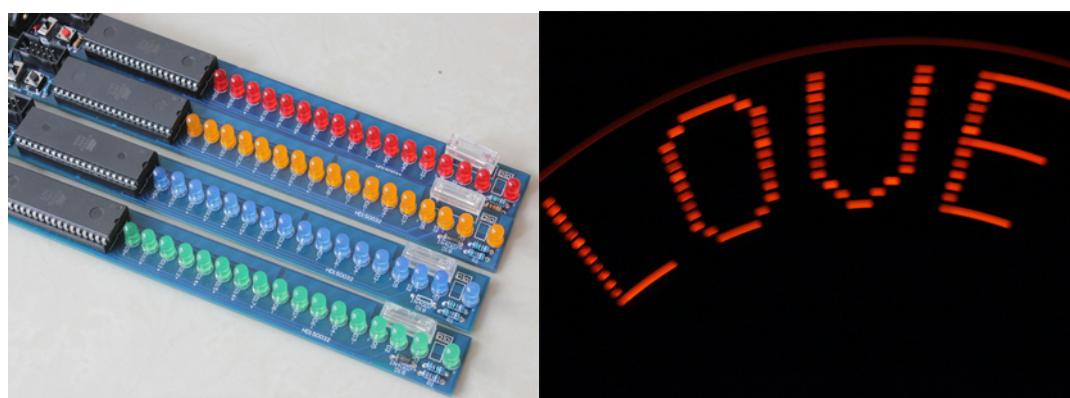
Aplikacja służąca do projektowania wyświetlnych animacji została stworzona w programie IntelliJ Idea Community Edition. W tym celu, oprócz instalacji wyżej wymienionego oprogramowania, konieczna była również instalacja środowiska Java Runtime Environment (JRE), bez której uruchomienie gotowego, skompilowanego kodu Java jest niemożliwe oraz Java Development Kit (JDK), bez którego stworzenie programu w języku Java byłoby znacznie trudniejsze.

Po instalacji oprogramowania IntelliJ Idea Community Edition można otworzyć projekt programu i podjąć się jego edycji, gdyż niewymagane są żadne dodatkowe wtyczki ani specjalne konfiguracje.

3. Przykłady istniejących rozwiązań

1. Shaking LED Stick

Jest to urządzenie działające na zasadzie POV, wymagające przy tym od użytkownika wkładu własnej siły fizycznej. Osoba trzymająca urządzenie w ręce robi zamach co pozwala na uzyskanie efektu jak to pokazano na rys. 2.



Rys. 2. Urządzenie oraz efekt jego działania

2. Rotational POV Ball Display

Urządzenie zostało stworzone przy pomocy diod RGB, co z kolei pozwoliło na wyświetlanie różnych kolorowych obrazów. Diody zamocowane są na półokrągłej płytce, co pozwala osiągnąć efekt sfery pokrytej obrazem. Całość umieszczona jest pod kulistą kopułą co zmniejsza ryzyko niebezpieczeństwa dla urządzenia jak i jego otoczenia. Taki wyświetlacz można zamówić w sklepach internetowych, takich jak na przykład AliExpress. Efekt działania urządzenia został przedstawiony na rys. 3.



Rys. 3. Urządzenie oraz efekt jego działania

3. Propeller Clock

Urządzenie to zostało stworzone przez jednego z użytkowników forum internetowego związanego z tematyką elektroniki. Płytki umieszczona jest wewnątrz przezroczystej obudowy, co zmniejsza ryzyko niebezpieczeństwa dokładnie tak samo, jak zostało to opisane we wcześniejszym przykładzie. Redukuje to również ilość hałasu wytwarzanego przez silniczek, który w tym przypadku został wymontowany z dysku twardego.



Rys. 4. Urządzenie oraz efekt jego działania

4. Wymagania

4.1. Wymagania niefunkcjonalne

- a) Program powinien uruchamiać się na systemie operacyjnym obsługującym maszynę wirtualną Java oraz udostępniającym interfejs graficzny.
- b) Oprogramowanie powinno posiadać czytelny i łatwy w obsłudze interfejs.
- c) Obsługa programu powinna być intuicyjna i niewymagająca specjalnego przeszkolenia.
- d) Projektowanie wyświetlanego obrazu nie powinno sprawiać użytkownikowi problemów.
- e) Tworzenie animacji nie powinno wymagać od użytkownika umiejętności programowania.
- f) Program powinien być niezawodny.

4.2. Wymagania funkcjonalne

- a) Program powinien generować kod wyświetlacza, który można bezpośrednio wgrać na mikrokontroler.
- b) Urządzenie powinno prawidłowo wyświetlać zaprojektowany obraz lub tekst.
- c) Różne rodzaje animacji wyświetlacza:
 - I. Przewijany tekst
 - II. Statyczny obraz

5. Projekt sprzętu

5.1. Wybór komponentów elektronicznych

- a) **Atmega8** - mikrokontroler z rodziny Atmel posiadający pamięć wbudowaną 8KB. Działa z częstotliwością zegara 16MHz. Zawiera pamięć SRAM o pojemności 1KB, co w zupełności wystarcza urządzeniom niewymagającym dużej mocy obliczeniowej, a którym zależy na oszczędności energii.



Rys. 5. Atmega8

Zdecydowałem się na użycie tego mikrokontrolera, gdyż akurat jestem w jego posiadaniu. Jego niskie zużycie energii gwarantuje długą pracę całego urządzenia.

- b) **TCRT5000** - odblaskowy czujnik optyczny z wyjściem tranzystorowym. Działa na zasadzie emitowania podczerwieni, która odbita od powierzchni wyłapywana jest przez czujnik optyczny, co pozwala na określenie tego jak daleko znajduje się obiekt od tej powierzchni.



Rys. 6. TCRT5000

Wybrałem go ze względu na łatwy sposób montażu oraz ograniczenia nałożone przez wyświetlacz. Otóż wyświetlacz składa się z dwóch części: pierwsza z nich odpowiedzialna jest za położenie drugiej części. Obie działają niezależnie od siebie, co znaczy, że nie są ze sobą połączone żadnymi przewodami. Toteż w celu wykrycia momentu pełnego obrotu drugiej części potrzebny był czujnik niewymagający przewodów. Czujnik optyczny TCRT5000 okazał się rozwiązaniem łatwym w implementacji, gdyż do wykrycia pełnego obrotu wystarczyło zamontować białą kartkę w odpowiednim miejscu tak, żeby czujnik dokładnie w tym momencie osiągnął najwyższą wartość napięcia, czyli odebrał najwięcej odbitych przez kartkę fal podczerwonych. Dzięki temu rozwiązaniu nie musiałem przejmować się obliczeniami dotyczącymi aktualnego momentu obrotowego wyświetlacza.

c) PCF8574



Rys. 7. PCF8574 – zdalny 8-bitowy ekspander wejścia/wyjścia dla szyny I²C

Ze względu na ograniczoną ilość pinów wyjść cyfrowych mikrokontrolera zostałem zmuszony do zastosowania dwóch ekspanderów wyjścia/wyjścia, pozwalających na podłączenie większej ilości diod nie tracąc przy tym na wydajności całego urządzenia.

Ekspayer 8-bitowy polega na tym, że przekazywana do niego jest liczba 8-bitowa (stąd też jego nazwa), odpowiadająca za określenie, które wyjścia otrzymują napięcie, a które nie. Ekspayer posiada 8 takich wyjść, co przekłada się na $2^8 = 256$ możliwych różnych kombinacji.

W moim projekcie pozwoliło to na podłączenie do wyświetlacza łącznie czternaście diod, zamiast zaledwie sześciu.

d) Diody elektroluminescencyjne



Rys. 8. Dioda elektroluminescencyjna

Zastosowane w projekcie diody, to diody elektroluminescencyjne, czyli takie, które emitują światło. Zaliczane są one do półprzewodnikowych przyrządów optoelektronicznych, emitujących promieniowanie w zakresie światła widzialnego, podczerwieni i ultrafioletu.

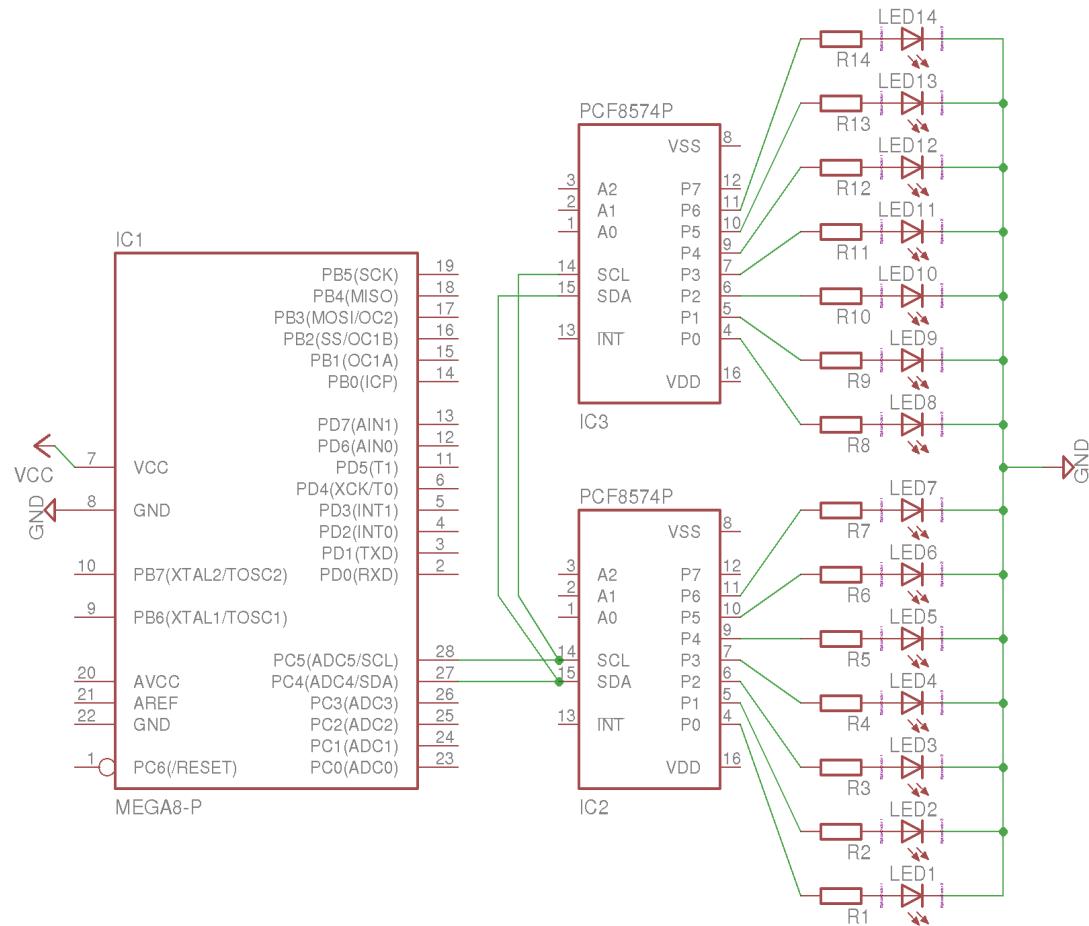
e) Akumulator niklowo-metalowo-wodorkowy

Za zasilanie wyświetlacza posłużyły cztery akumulatory niklowo-metalowo-wodorkowe o pojemności 2700 MAh. Taki rodzaj zasilania zapewnia wyświetlaczowi wiele godzin nieustannej pracy.

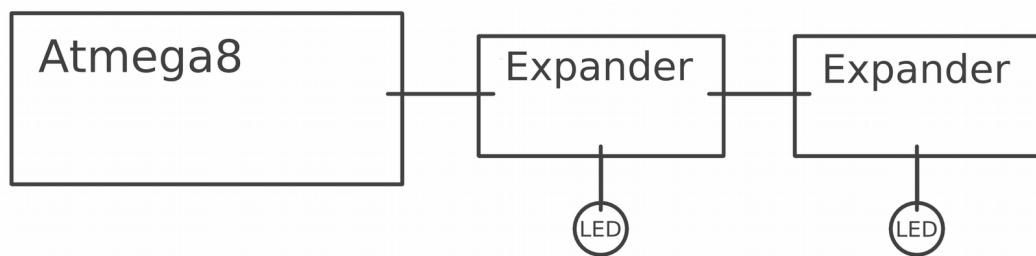


Rys. 9. Akumulator niklowo-metalowo-wodorkowy

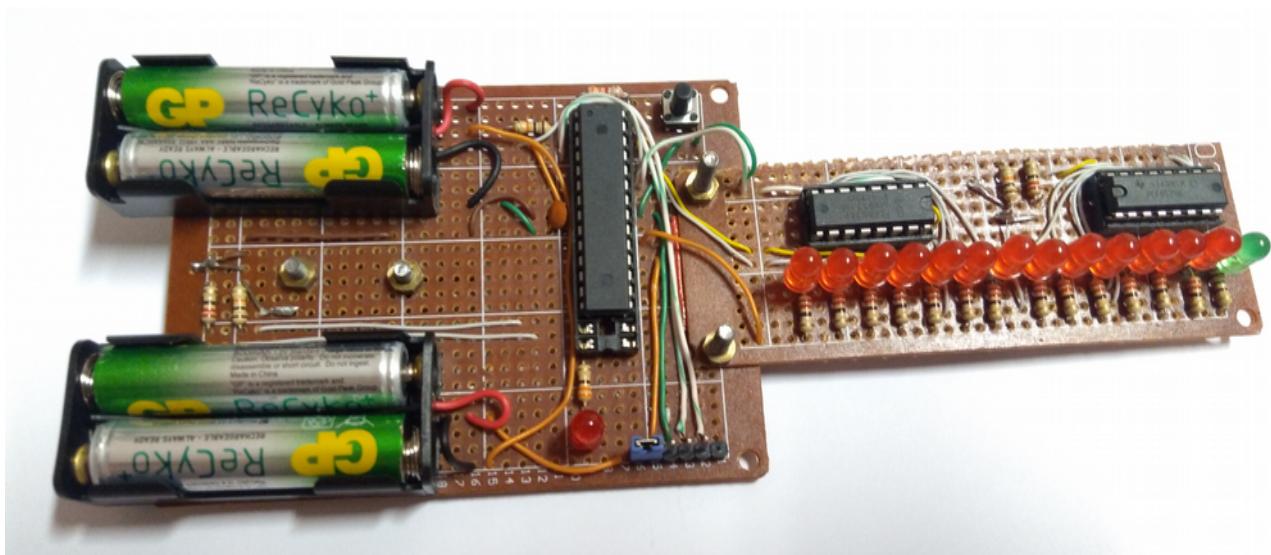
5.2. Projekt i wykonanie modułu wyświetlacza



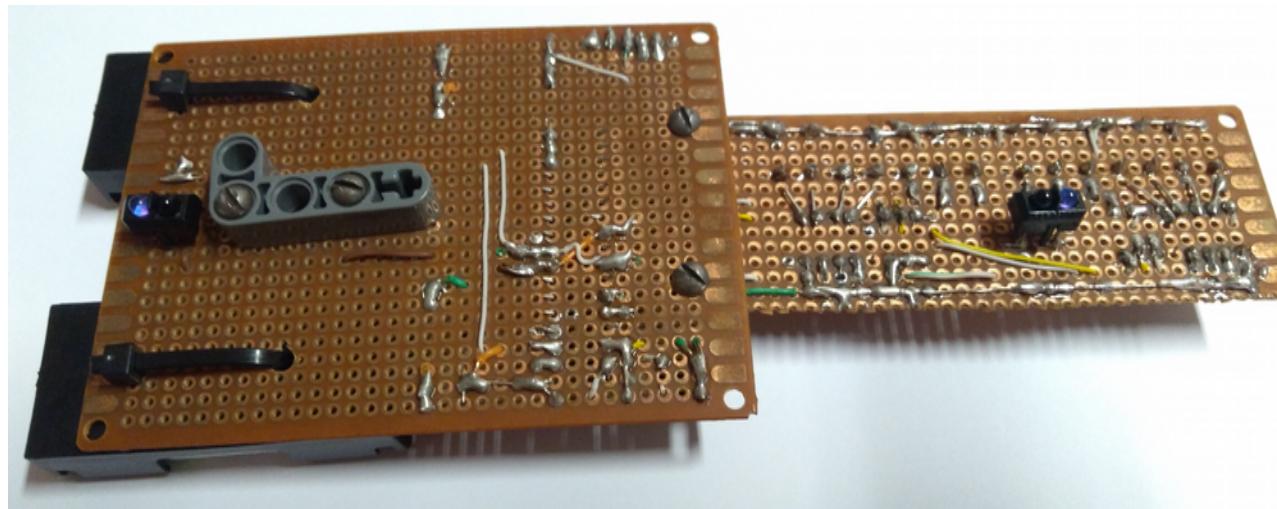
Rys. 10. Schemat ideowy modułu wyświetlacza



Rys. 11. Schemat blokowy modułu wyświetlacza

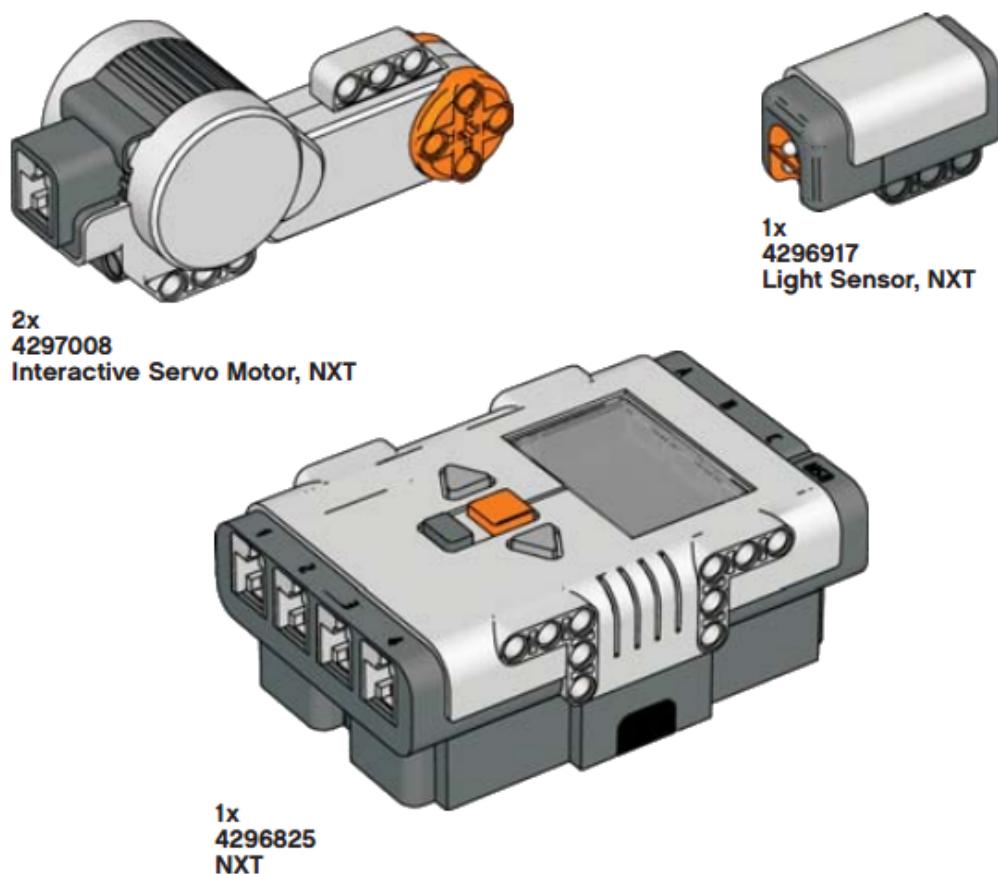


Rys. 12. Zdjęcie gotowej płytki (góra)

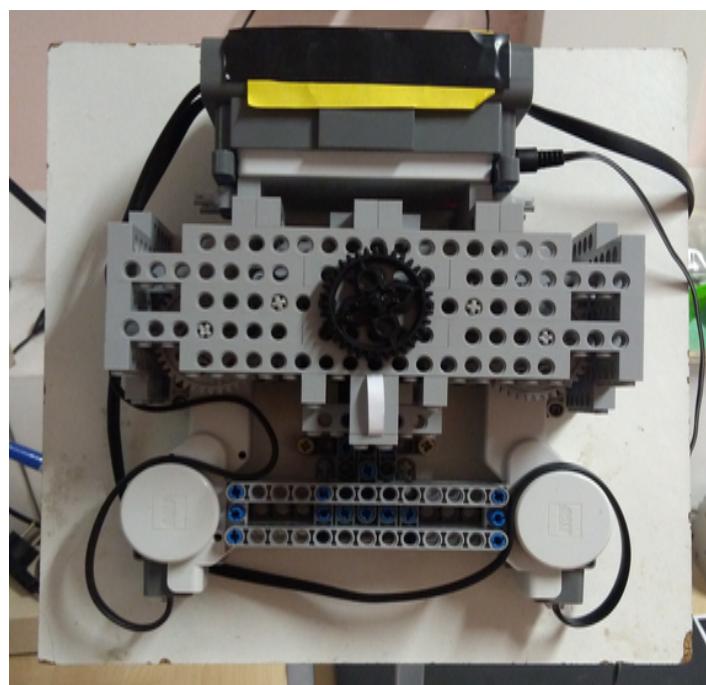


Rys. 13. Zdjęcie gotowej płytki (spód)

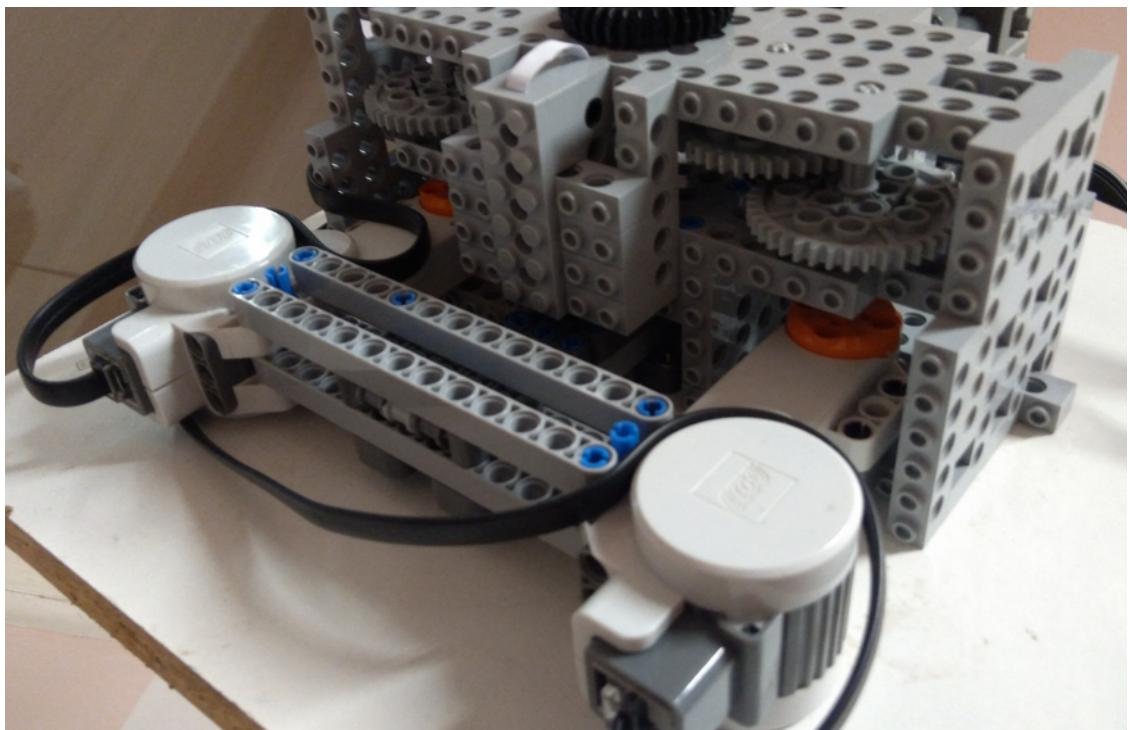
5.3. Projekt i wykonanie modułu napędowego wyświetlacza



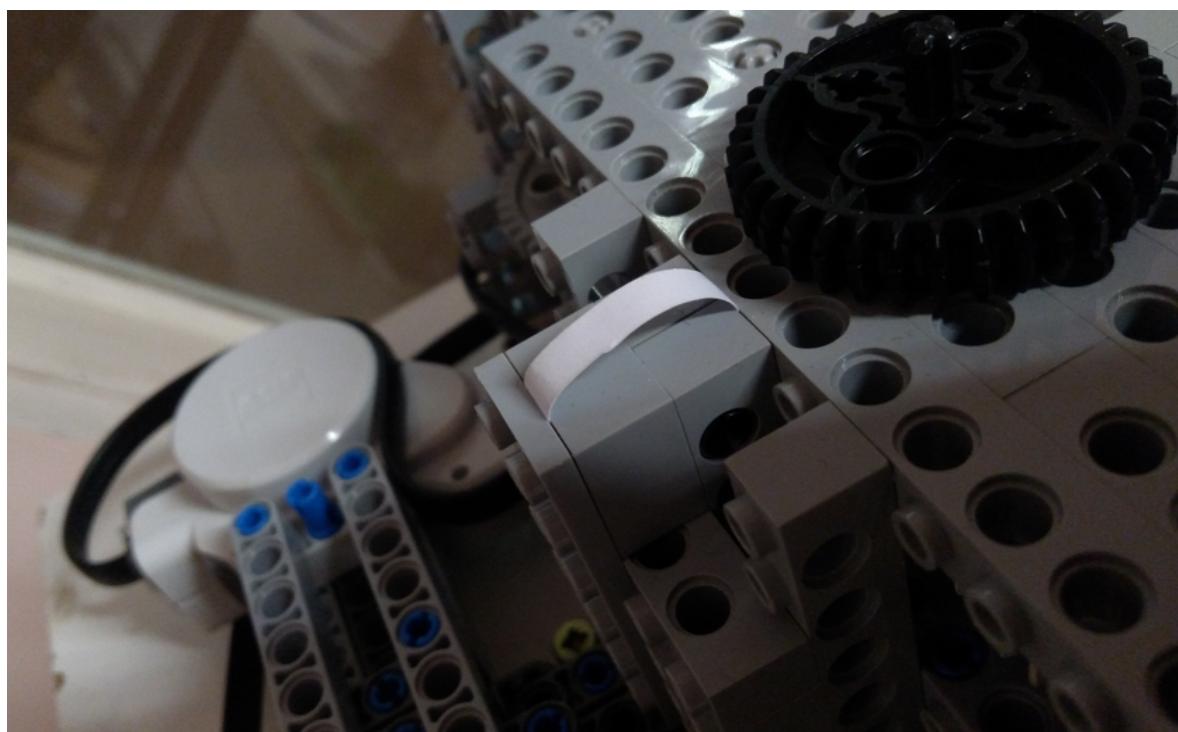
Rys. 14. Wykorzystane moduły zestawu LEGO Mindstorms NXT



Rys. 15. Zdjęcie gotowego modułu silnikowego



Rys. 16. Przekładnie wewnętrz silnika



Rys. 17. Punkt zerowy wyświetlacza

6. Projekt oprogramowania

6.1. Model opracowanego systemu informatycznego

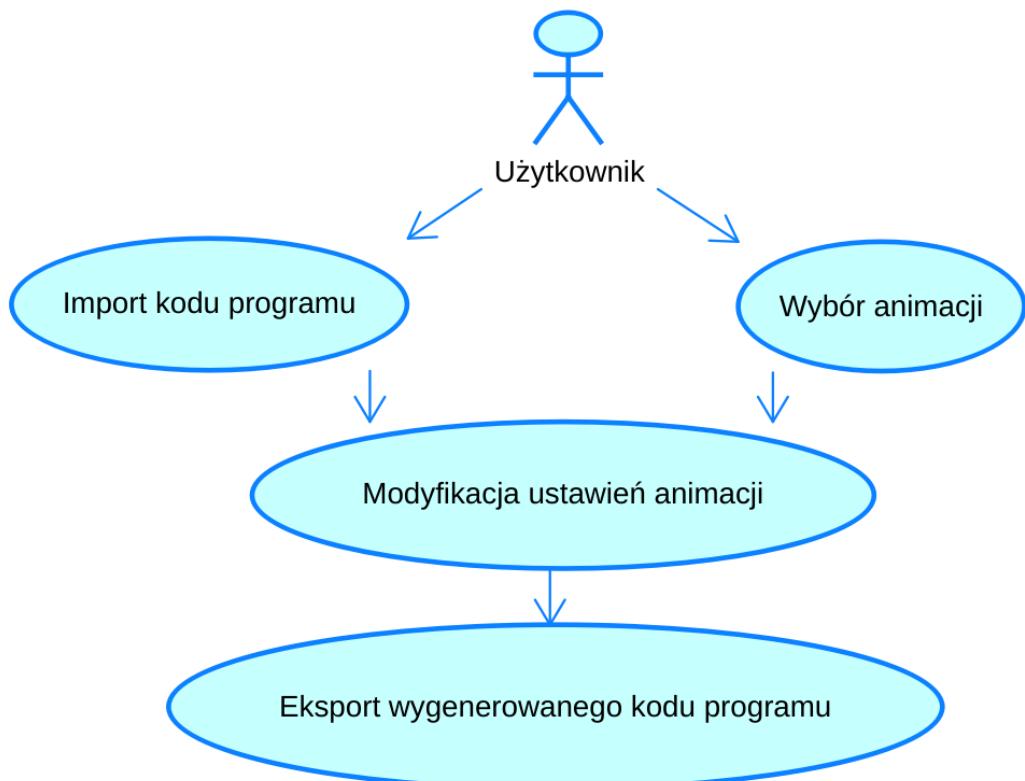
a) Diagram aktywności

Rys. 18. Diagram aktywności

b) Diagram sekwencji

Rys. 19. Diagram sekwencji

c) Diagram przypadków użycia



Rys. 20. Diagram przypadków użycia

6.2. Oprogramowanie mikrokontrolera wyświetlacza

Oprogramowanie mikrokontrolera składa się z dwóch części. Część pierwsza odpowiedzialna jest za obliczanie aktualnej pozycji wyświetlacza oraz obsługę wszystkich komponentów, czyli przesyłanie odpowiednich danych do ekspanderów skutkujących ustawieniem stanów poszczególnych ledów. Druga część odpowiedzialna jest za obsługę wyświetlanej animacji.

```
IR = analogRead(IR_PIN_1);
if (IR_countdown > 0)
    IR_countdown--;
else {
    if (IR > IR_THRESHOLD)      Rys. 21.
        doReset = true;
    else if (doReset) {          Fragment kodu
        doReset = false;         wyliczającego pozycję
        currentStep = 0;          wyświetlacza
        IR_countdown = 10;
    }
}
```

Rys. 21.
Fragment kodu
wyliczającego pozycję
wyświetlacza

Najważniejszym fragmentem kodu mikrokontrolera jest wyzerowanie w odpowiednim momencie aktualnej pozycji wyświetlacza. Z każdym przeskokiem programu kod animacji zmienia wartość zmiennej *currentStep*, którą należy wyzerować w momencie osiągnięcia punktu zero. Na rys.17 przedstawiono fragment kodu, który wychwytuje moment, w którym wyświetlacz osiąga wspomniany punkt zero, a następnie zmienia wartość *currentStep* na 0.

Drugim równie istotnym elementem oprogramowania mikrokontrolera jest ustawienie stanu ledów. Jako, że przesyłamy naraz wszystkie stany ledów do ekspandera, to wcześniej aby ustawić każdy z osobna, muszą te stany być przechowane w programie. Aby to osiągnąć stworzyłem klasę **Display**, która zawiera zarówno aktualne stany ledów, jak również funkcje, które ustalają stany poszczególnych ledów oraz wysyłają te dane do ekspanderów.

```
enum LedState { OFF = 0, ON = 1 };
class Display {
    private:
        static byte expander[];
        static byte LEDs[];
    public:
        static void Update();
        static void SetLEDs(byte expander, byte LEDs);
        static void TurnLEDs(byte expander, byte LEDs, LedState state);
};
```

Rys. 22. Klasa wyświetlacza ułatwiająca ustawianie stanów ledów

- **expander** - tablica zawierająca adresy poszczególnych ekspanderów
- **LEDs** – tablica zawierająca aktualne stany ledów na każdym z ekspanderów
- **Update()** - funkcja wysyłająca aktualne stany ledów na każdy z ekspanderów
- **SetLeds(expander,leds)** – ustawia stany wszystkich ledów na danym ekspanderze
- **TurnLeds(expander,leds,state)** – ustawia stan podanych ledów na danym ekspanderze. Na przykład: Jednocześnie pierwszy, drugi i piąty, nie zmieniając przy tym stanów reszty ledów

W części drugiej oprogramowania kod animacji odpowiada za ustawianie stanów oraz naliczanie wartości zmiennej currentStep. Na rys. 23. przedstawiono fragment przykładowej animacji, jaką w tym przypadku jest przewijany tekst.

```
iText = (currentStep - (MAX_STEPS + textLenWidth - position));
iChar = iText % 6;
iText /= 6;

Display::SetLEDS(0, 0);
Display::SetLEDS(1, 0);

if (iText >= 0 && iText < textLen && iChar >= 0 && iChar < 5)
    if (textToDisplay[iText] != 32) {
        Display::SetLEDS(0, FONT[ FONT_MAPPING[textToDisplay[iText]] ][iChar][0]);
        Display::SetLEDS(1, FONT[ FONT_MAPPING[textToDisplay[iText]] ][iChar][1]);
    }

if (ANIMATION_OUTLINE)
    Display::TurnLEDS(0, 8, ON);

if ((positionCountdown = (positionCountdown + 1) % ANIMATION_SPEED) == 0)
    position = (position + 1) % (MAX_STEPS + textLenWidth * 2);

Display::Update();
if ((currentStep += 1) >= MAX_STEPS) {
    Display::SetLEDS(0, 0);
    Display::SetLEDS(1, 0);
    Display::Update();
}
```

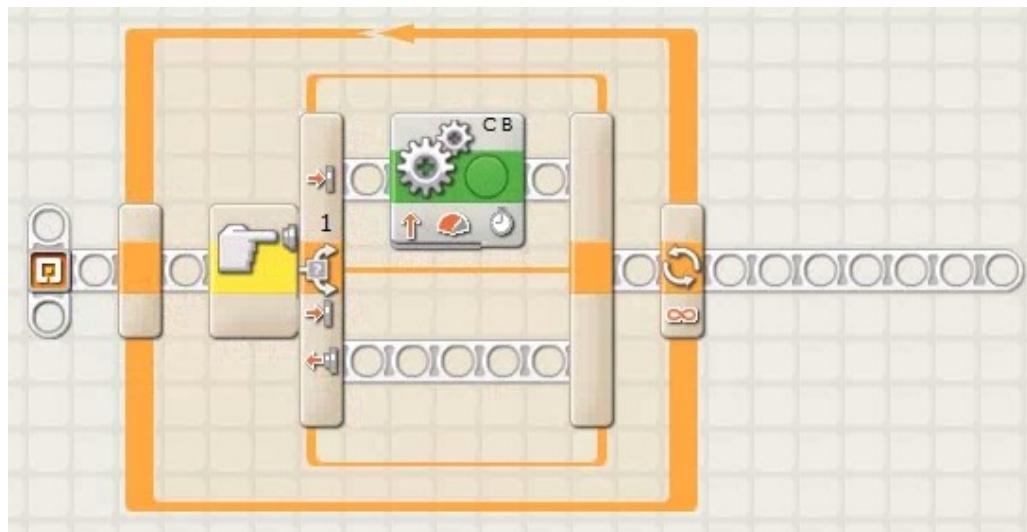
Rys. 23. Animacja przewijanego tekstu

6.3. Oprogramowanie kostki NXT

Program sterujący mocą silnika został stworzony w oprogramowaniu LEGO Mindstorms NXT. Wygląd programu został przedstawiony na rys. 24.

Komponenty, których użyto do stworzenia wspomnianego programu, to:

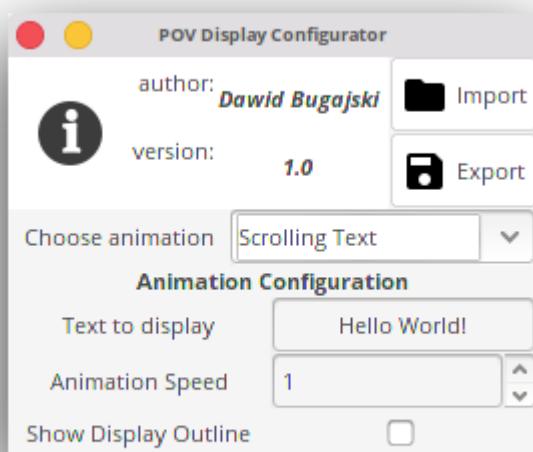
- **motor** – pozwalający na synchroniczne sterowanie pracą dwóch silników napędzających zębatki, które z kolei poruszają płytka wyświetlacza.
- **sensor** – z ustawieniem na czujnik światła pozwala na uruchomienie silniczków tylko wtedy, gdy czujnik światła wykrył ruch.
- **loop** – dzięki temu komponentowi program na kostce wykonuje się bez końca. Dzięki temu można włączyć urządzenie na dłuższy czas, wymagając jedynie by użytkownik od czasu do czasu przybliżył się do czujnika światła umieszczonego w odpowiednim miejscu, który z kolei włączy silniczki napędzające.



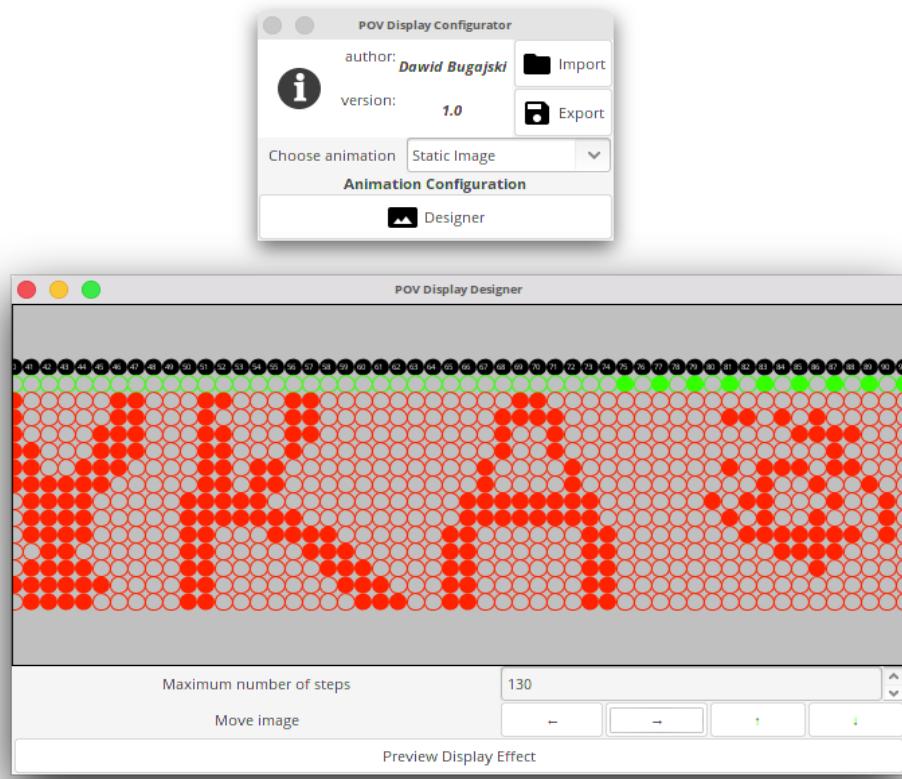
Rys. 24. Program LEGO do sterowania silnikiem.

6.4. Oprogramowanie PC

Oprogramowanie PC pozwala użytkownikowi na stworzenie animacji nie wymagając przy tym od niego umiejętności programowania w języku C. Program posiada interfejs graficzny, który jest w całości intuicyjny. Stworzona animacja eksportowana jest do gotowego kodu programu, który następnie można wgrać na mikrokontroler.



Rys. 25. Główne okno programu



Rys. 26. Okno projektowania

7. Dokumentacja użytkownika

7.1. Obsługa oprogramowania PC.

1. Po uruchomieniu programu możemy wybrać interesującą nas animację lub otworzyć poprzednio zapisany projekt.
2. Program następnie wyświetla dodatkowe opcje konfiguracji w zależności od wybranej animacji.
 - a) **Scrolling Text** – to animacja tworząca efekt przewijanego tekstu powszechnie znanego z reklam przemysłowych opartych o technologię diod LED. Możemy skonfigurować wartość tekstu, który zostanie wyświetlony, szybkość przewijania tekstu oraz czy w trakcie wyświetlania tekstu ma zostać wyświetlone obramowanie wyświetlacza.
 - b) **Static Image** – po wybraniu tej opcji mamy dostęp do okna projektowania, w którym możemy ręcznie ustawić stany poszczególnych ledów na wybranych pozycjach przejścia wyświetlacza.
3. Po odpowiednim skonfigurowaniu animacji możemy ją wyeksportować do kodu arduino, który z kolei możemy wgrać bezpośrednio na mikrokontroler.

7.2. Obsługa oprogramowania Arduino.

Po wygenerowaniu za pomocą oprogramowania *POV Display Configurator* kodu mikrokontrolera, musi on zostać wgrany na urządzenie.

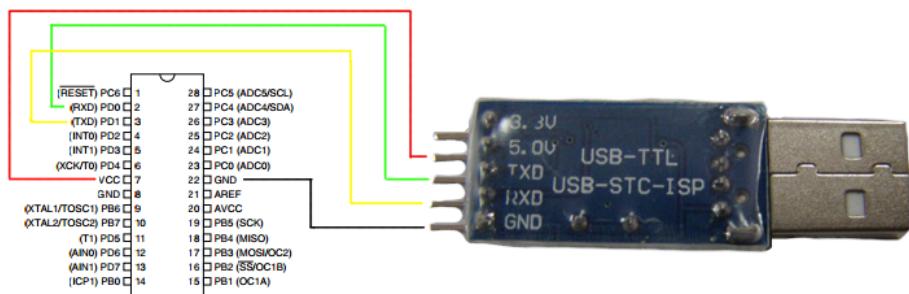
W tym celu należy:

1. Otworzyć zapisany plik z kodem mikrokontrolera w programie Arduino IDE.
2. Z pozycji menu **Narzędzia** wybrać opcję:
 - a) *Płytki* → Atmega8/A
 - b) *Wersja procesora* → Atmega8
 - c) *Użyj Bootloader'a* → Tak
 - d) *Prędkość procesora* → 8MHz Internal Oscillator
 - e) *Programator* → Arduino as ISP

3. Kolejnym krokiem jest wgranie programu. Aby to zrobić należy:
- Podłączyć programator **PL2303** według zamieszczonego poniżej schematu:



Rys. 27. Programator PL2303 USB to TTL-UART Converter



Rys. 28. Schemat podłączenia programatora oraz płytka wyświetlacza

- Nacisnąć w programie Arduino przycisk „Wgraj”, który skompiluje i wgra za pomocą programatora skompilowany kod na mikrokontroler.
- Jedyną rzeczą jaką osoba wgrywająca program musi dopilnować, jest odliczenie około 5 sekund od momentu wcisnięcia przycisku wgrywania programu, a następnie przyciśnięcie na pół sekundy przycisk RESET na płytce wyświetlacza.



Rys. 29.
Przycisk RESET na
płytcie wyświetlacza

8. Podsumowanie

Celem mojej pracy było stworzenie oraz oprogramowanie wyświetlacza LED sterowanego mikrokontrolerem wykorzystując zjawisko *Persistence Of Vision*, a także stworzenie aplikacji umożliwiającej projektowanie wyświetlanych na urządzeniu animacji osobom, które nie posiadają wiedzy na temat programowania mikrokontrolerów. Projekt zakończył się sukcesem, gdyż wszelkie założenia dotyczące zarówno wyświetlacza, jak i aplikacji zostały spełnione.

Aplikacja została przetestowana na komputerach z systemami Linux, Windows 7 oraz 10, a także OS X 10.10 Yosemite. Praca z programem nie przysparzała w żadnym z wymienionych systemów jakichkolwiek problemów. Wygenerowany przez aplikację kod, również był prawidłowy, co pozwoliło na wgranie go na mikrokontroler, nie wymagając przy tym wprowadzania żadnych zmian w wygenerowanym kodzie.

Największymi problemami jakie napotkałem podczas realizacji projektu było odpowiednie wyważenie płytki wyświetlacza, zbudowanie sprawnego silnika obracającego płytka oraz znalezienie uniwersalnego sposobu na generowanie kodu w zależności od wybranej animacji.

Aplikację w przyszłości można rozbudować o import obrazów, różne rodzaje animacji (na przykład kilku-klatkowe nagranie wideo) oraz uzupełnienie i ulepszenie czcionki wyświetlanej w przypadku animacji przewijanego tekstu. Można również dodać do urządzenia możliwość bezprzewodowej komunikacji, co znacznie ułatwiłoby modyfikowanie wyświetlanych animacji.

9. Bibliografia

9.1. Zasoby internetowe

- [1] <https://www.aliexpress.com/item/POV-display-shaking-LED-stick-diy-Electronic-diy-kits-soldering-kits/32706118452.html>
- [2] <https://www.aliexpress.com/item/Rotational-POV-diamete-200-mm-120-LEDS-full-color-POV-animation-POV-DIY-programming-rotary-screen/32804494575.html>
- [3] <http://www.elektroda.pl/rtvforum/topic944484.html>
- [4] http://www.atmel.com/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf
- [5] <http://www.vishay.com/docs/83760/tcrt5000.pdf>
- [6] https://www.nxp.com/documents/data_sheet/PCF8574.pdf
- [7] <https://potentiallabs.com/cart/image/cache/catalog/Arduino/10%20mm%20red%20led-600x315.jpg>
- [8] <http://img.nokaut.pl/p-56-dc-56dc7cd29e802004ed22e173a46074ba500x500/akumulatory-nimh-aa-gp-batteries-12-v-2600-mah-1-szt-oferta-35fdbd6a6cf63f41e.jpg>
- [9] http://www.depts.ttu.edu/coe/stem/gear/ev3/documents/Base-Set_Parts-List.pdf
- [10] https://d3s5r33r268y59.cloudfront.net/88221/products/thumbs/2014-04-30T14:13:03.647Z-PL2303.jpg.2560x2560_q85.jpg

9.2. Spis rysunków

Rys. 1. Schemat sekwencji diod.....	4
Rys. 2. Shaking LED Stick. Źródło: [1].....	9
Rys. 3. Rotational POV Ball Display. Źródło: [2].....	9
Rys. 4. Propeller Clock. Źródło: [3].....	10
Rys. 5. Atmega8. Źródło: [4].....	12
Rys. 6. TCRT5000. Źródło: [5].....	12
Rys. 7. PCF8574. Źródło: [6].....	13
Rys. 8. Dioda elektroluminescencyjna. Źródło: [7].....	13
Rys. 9. Akumulator niklowo-metalowo-wodorkowy. Źródło: [8].....	13
Rys. 10. Schemat ideowy modułu wyświetlacza.....	14
Rys. 11. Schemat blokowy modułu wyświetlacza.....	14
Rys. 12. Płytnica wyświetlacza, góra.....	15
Rys. 13. Płytnica wyświetlacza, spód.....	15
Rys. 14. Wykorzystane moduły zestawu LEGO Mindstorms NXT. Źródło: [9].....	16
Rys. 15. Moduł silnikowy.....	16
Rys. 16. Przekładnie wewnętrz silnika.....	17
Rys. 17. Punkt zerowy wyświetlacza.....	17
Rys. 18. Diagram aktywności.....	18
Rys. 19. Diagram sekwencji.....	18
Rys. 20. Diagram przypadków użycia.....	19
Rys. 21. Fragment kodu wyświetlacza.....	20
Rys. 22. Klasa wyświetlacza.....	20
Rys. 23. Kod animacji przewijanego tekstu.....	21
Rys. 24. Program LEGO do sterowania silnikiem.....	22
Rys. 25. POV Display Configurator – okno główne programu.....	23
Rys. 26. POV Display Configurator – okno projektowania.....	23
Rys. 27. Programator PL2303. Źródło: [11].....	25
Rys. 28. Schemat podłączenia programatora oraz płytka wyświetlacza.....	25
Rys. 29. Przycisk RESET na płytce wyświetlacza.....	25

Zawartość płyty CD

Na płycie CD dołączonej do pracy znajduje się:

- kod źródłowy aplikacji
- przykładowy kod źródłowy animacji przewijanego tekstu
- praca inżynierska w formacie PDF