Wyższa
Szkoła
Informatyki
i Umiejętności

WYŻSZA SZKOŁA INFORMATYKI I UMIEJĘTNOŚCI
Wydział Informatyki i Zarządzania
Kierunek: Informatyka

Marcin Maciaszczyk
nr albumu: 29572

Praca Magisterska
System oświetlenia skonstruowany za pomocą diod elektroluminescencyjnych oraz mikrokontrolera Arduino Uno

Praca napisana pod kierunkiem
dr inż. Grzegorz Zwoliński

Rok akademicki 2016/2017

Spis treści

1 Wstęp **3**
- 1.1 Uzasadnienie wyboru tematu . . . 3
- 1.2 Problematyka i zakres pracy . . . 6
- 1.3 Cele pracy . . . 6
- 1.4 Metoda badawcza . . . 7
  - 1.4.1 Studia literaturowe . . . 7
  - 1.4.2 Analiza istniejących systemów oświetlenia . . . 7
  - 1.4.3 Stworzenie własnego systemu oświetlenia . . . 8
  - 1.4.4 Analiza porównawcza oraz testy . . . 8
- 1.5 Przegląd literatury w dziedzinie . . . 8
  - 1.5.1 Tworzenie oprogramowania w języku Swift . . . 8
  - 1.5.2 Tworzenie oprogramowania przeznaczonego na mikrokontrolery Arduino . . . 9
- 1.6 Układ pracy . . . 9

2 Zagadnienia teoretyczne **11**

3 Analiza istniejących rozwiązań **12**
- 3.1 Kryteria analizy . . . 12
- 3.2 Porównanie istniejących rozwiązań . . . 13
  - 3.2.1 Technologia Philips Ambilight . . . 13
  - 3.2.2 System oświetlenia Lightpack . . . 15
  - 3.2.3 System oświetlenia Lightpack 2 . . . 17

4 System oświetlenia Lightning **19**
- 4.1 Analiza wymagań . . . 19
  - 4.1.1 Wymagania funkcjonalne . . . 19
  - 4.1.2 Wymagania niefunkcjonalne . . . 19
- 4.2 Komponenty systemu . . . 20

4.3 Moduły oprogramowania 21
4.3.1 Oprogramowanie mikrokontrolera 21
4.3.2 Aplikacja przeznaczona na komputer 22
4.4 Projekt 22
4.4.1 System kontroli wersji 22
4.4.2 Wykorzystane technologie 23
4.4.3 Diagram klas 24
4.4.4 Wzorce projektowe 24
4.4.5 Interfejs użytkownika 24
4.5 Implementacja 26
4.5.1 Oprogramowanie mikrokontrolera 26
4.5.2 Aplikacja przeznaczona na komputer 30
4.6 Podręcznik użytkownika 30
4.7 Przykładowa implementacja animacji 31
4.8 Analiza projektu 33
4.9 Perspektywy rozwoju projektu 34

5 Podsumowanie **36**
5.1 Dyskusja wyników 36
5.2 Perspektywy rozwoju pracy 36

Bibliografia **37**

Spis rysunków **40**

Spis tabel **41**

Spis listingów **42**

Rozdział 1

Wstęp

1.1 Uzasadnienie wyboru tematu

Wraz z upływem czasu postęp technologiczny ma wpływ na życia co raz szerszej rzeszy ludzi na całym świecie. Niezliczone ilości urządzeń zagościły na stałe w domach i mało kto wyobraża sobie bez nich swoje życie. Zaczynając od artykułów gospodarstwa domowego, a kończąc na elektronice użytkowej do której zaliczają się komputery, telewizory czy też smartfony[1]. Wszystkie te urządzenia mają na celu ułatwianie życia swoim użytkownikom.

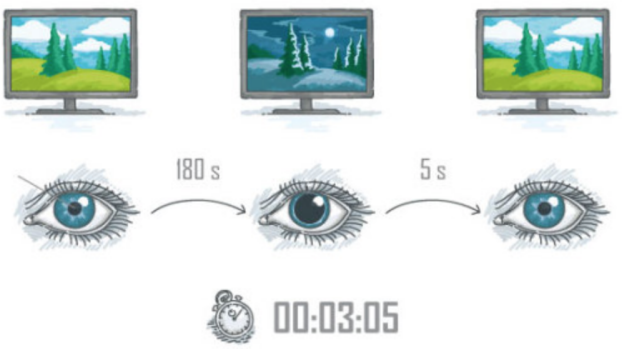
W parze z licznymi zaletami urządzeń elektronicznych idą jednak pewne wady. Jedną z istotniejszych jest wpływ czasu spędzanego przed różnego rodzaju wyświetlaczami na zdrowie. Badania przeprowadzone na bazie danych Nielsen Audience Measurement pokazują, że przeciętny Polak spędza dziennie średnio 4,5 godziny przed ekranem telewizora[2]. Nie oznacza to jednak, że przez cały ten czas ogląda on telewizję. Oglądanie filmów z dysku komputera, za pomocą serwisów VOD[3] czy granie na konsoli także są wliczone w ten czas. Gdyby jednak dodać do tego czas spędzony przed ekranem smartfona czy też komputera wynik byłby zapewne dwukrotnie większy.

Pogorszenie wzroku czy też wysychanie gałki ocznej są wymieniane jako najczęstsze skutki zbyt dużej ilości czasu spędzanego przed ekranem. Poza próbą jego ograniczenia, jedną z częstszych porad jest próba zmniejszenia kontrastu pomiędzy ekranem a jego otoczeniem.

[1] Przenośne urządzenia łączące w sobie zalety telefonów komórkowych oraz przenośnych komputerów (z ang. smartphone).

[2] Badania zostały przeprowadzone z uwzględnienem osób powyżej 4 roku życia w okresie od stycznia do czerwca 2015 roku [1].

[3] Wideo na życzenie (z ang. video on demand).



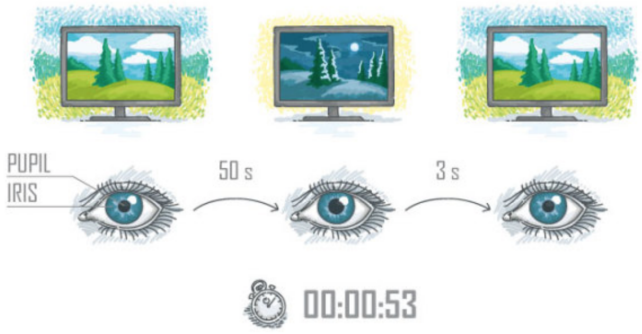
Rysunek 1.1: Średni czas dopasowania źrenicy do wyświetlanego obrazu [2]

W roku 2002 firma Philips opatentowała technologię Ambilight[4], która ma za zadanie rozszerzać obraz wyświetlany na ekranie na jego otoczenie za pomocą diod umieszczonych na tylnym panelu tworzonych telewizorów. Ambilight spotkało się z bardzo dobrym odbiorem społeczności, ponieważ poza imponującymi efektami wizualnymi zgodnie z badaniami przeprowadzonymi przez profesora Begemanna redukuje ono negatywny wpływ wyświetlaczy na oko użytkowników telewizorów [3].

Rysunek 1.2: Telewizor z technologią Philips Ambilight [4]

[4] Otaczające oświetlenie (z ang. ambient lighting).

Ambilight posiada jednak znaczącą wadę, jest ono przeznaczone tylko dla telewizorów marki Philips, a więc nie mogą z niej korzystać użytkownicy telewizorów innych marek jak i ekranów podłączonych do komputera.



Rysunek 1.3: Średni czas dopasowania źrenicy do wyświetlanego obrazu z otaczającym oświetleniem [2]

Autor niniejszej pracy jako cel postawił sobie złożenie i oprogramowanie systemu oświetlenia, który ma rozszerzać obraz widziany na ekranie na jego otoczenie w sposób podobny do Philips Ambilight. Poza zmniejszeniem kontrastu, a więc aspektem zdrowotnym, system ma także na celu zwiększyć wrażenia wizualne dostarczane przez oglądany obraz.

System oświetlenia składa się z taśmy diod elektroluminescencyjnych[5] podłączonych do mikrokontrolera Arduino Uno, który z kolei ma współpracować z komputerem z zainstalowanym systemem operacyjnym macOS. Oprogramowanie mikrokontrolera, którego zadaniem jest sterowanie diodami zostało przygotowane w języku Arduino, natomiast aplikacja kontrolująca cały system przeznaczona na komputer z systemem macOS została napisana w języku Swift. Wybór języków jest ściśle związany z koniecznością uzyskania jak najlepszej wydajności oraz użyciem najnowszych technologii.

Podobne systemy oświetlenia dostępne są już od pewnego czasu na rynku, jednak to właśnie nowoczesne technologie, prostota wykonania i niskie koszta powinny uczynić z Ligtning, bo taką nazwę otrzymał projekt, pełnowartościowego konkurenta.

[5] LED (z ang. light-emitting diode).

1.2 Problematyka i zakres pracy

Różnego rodzaju wyświetlacze towarzyszą w dniu codziennym większości ludzi. Co raz większa jest także popularność różnego rodzaju gadżetów elektronicznych. Niektórzy decydują się nawet na budowanie własnych narzędzi wykorzystując gotowe komponenty takie jak mikrokontrolery i inne urządzenia peryferyjne. Niniejsza praca porusza właśnie problematykę tworzenia oprogramowania przeznaczonego komputer jak i mikrokontroler, którego zadaniem będzie sterowanie taśmą diod elektroluminescencyjnych.

Zadanie to wymaga przejścia kilku kolejnych etapów, które składają się na zakres pracy:

1. Analiza istniejących rozwiązań realizujących podobne zadania.
2. Zebranie wymagań umożliwiających stworzenie konkurencyjnego systemu oświeletnia.
3. Zaprojektowanie systemu oświetlenia.
4. Skompletowanie odpowiednich komponentów i ich podłączenie.
5. Zaprojektowanie oprogramowania systemu oświetlenia przeznaczonego na mikrokontroler i komputer.
6. Implementacja oprogramowania.
7. Dokumentacja projektu.
8. Analiza stworzonego systemu oświetlenia.
9. Dyskusja osiągniętych wyników jak i perspektywy rozwoju projektu.

1.3 Cele pracy

Do najważniejszych celów niniejszej pracy dyplomowej należą:

- Analiza istniejących systemów oświetlenia o podobnym działaniu, a co za tym idzie dokładniejsze zapoznanie się z istniejącymi rozwiązaniami. Umożliwiło to sprecyzowanie wymagań oraz podjęcie kluczowych decyzji dotyczących wykorzystanych komponentów czy też technologii.

- Stworzenie własnego systemu oświetlenia umożliwiające dokładne zapoznanie się problematyką projektu, czyli między innymi:
  - Tworzeniem wielomodułowych systemów, a więc synchronizacji i komunikacji pomiędzy modułami.
  - Kompletowaniem złożonego z kilku komponentów systemu oświetlenia.
  - Optymalizacją obliczeń związanych z przechwytywaniem i obróbką obrazu.
  - Projektowaniem graficznego interfejsu użytkownika.
- Porównanie Lightning z analizowanymi wcześniej systemami mające na celu stwierdzić czy przyjęte założenia i wymagania okazały się trafne.

1.4 Metoda badawcza

1.4.1 Studia literaturowe

Większość źródeł odnoszących się do tematyki niniejszej pracy znajduje się w Internecie. Źródła te można podzielić na następujące kategorie:

- Tworzenie oprogramowania w języku Swift.
- Tworzenie oprogramowania przeznaczonego na mikrokontrolery Arduino.

Rozdział 1.5 przedstawia najważniejsze pozycje spośród z każdej wymienionych kategorii źródeł.

1.4.2 Analiza istniejących systemów oświetlenia

Studia literaturowe to jedna z ważniejszych metod badawczych, jednakże analiza istniejących systemów oświetlenia jest równie istotna. Umożliwia ona wczesne zlokalizowanie istniejących problemów związanych z wybraną tematyką jak, braków istniejących rozwiązań i sprecyzowanie potrzeb użytkowników.

1.4.3 Stworzenie własnego systemu oświetlenia

Etapem następnym po zapoznaniu się z istniejącą literaturą i przeprowadzeniu analizy istniejących systemów oświetlenia jest stworzenie właśnego systemu oświetlenia. Jest to ważna metoda badawcza, ponieważ umożliwia dogłębne zapoznanie się z problematyką pracy jak i przejście przez wszystkie etapy tworzenia własnego systemu wykorzystującego mikrokontroler Arduino współpracujący z komputerem. Wszystkie napotkane problemy muszą zostać rozwiązane w jak najbardziej optymalny sposób, prowadzi to więc to do precyzyjnego zbadania wybranej tematyki.

1.4.4 Analiza porównawcza oraz testy

Analiza stworzonego systemu oświetlenia i porównanie go z isniejącymi już rozwiązaniami ma za zadanie poprowadzić do jak najtrafniejszych wniosków. Celem tego etapu jest dowiedzenie się czy przyjęte założenia i wybrane rozwiązania są lepsze od tych które przyjęli autorzy istniejących już rozwiązań.

1.5 Przegląd literatury w dziedzinie

1.5.1 Tworzenie oprogramowania w języku Swift

Swift został opublikowany przez Apple w 2014 roku. Jest on następcą języka znanego jako Objective C, posiada on szereg usprawnień i nowych funkcjonalności mających za zadanie sprawić aby tworzenie oprogramowania na platforme Mac było jeszcze sprawniejsze. Oto lista wykorzystanych źródeł, które są związane z właśnie tym językiem:

- *The Swift Programming Language (Swift 3.1)* [5] – Najlepsze i najdokładniejsze źródło informacji dotyczących Swifta. Publikacja utworzona przez twórców języka, firmę Apple.
- *App Development with Swift* [6] – Kurs nauki programowania w języku Swift utworzony przez firmę Apple.
- Oficjalna strona internetowa otwartoźródłowego projektu Swift [7] – Źródła, przydatne projekty oraz przede wszystkim odnośniki do dokumentacji.
- Strona Apple Developer [8] – Obszerna baza gotowych przykładów, dokumentacja oraz pomoce dotyczące środowiska Xcode.

Fakt, że są to jedynie źródła internetowe związany jest głównie z niskim wiekiem języka a także jego ciągłym rozwojem.

1.5.2 Tworzenie oprogramowania przeznaczonego na mikrokontrolery Arduino

Arduino należy do jednego z najpopularniejszych producentów mikrokontrolerów, są one proste w obsłudze, posiadają dużą społeczność i są łatwo dostępne. W celu zapoznania się z zagadnieniami dotyczącymi tworzenia oprogramowania przeznaczonego na mikrokontrolery Arduino autor skorzystał z następujących źródeł:

- *Arduino w akcji* [9] – Książka poświęcona platformie Arduino. Zawiera informacje ułatwiające realizację projektów od najprostszych do tych bardziej wymagających wykorzystujących liczne operacje wejścia-wyjścia jak i komunikujących się z innym oprogramowaniem.
- *Arduino. Kurs video. Poziom pierwszy. Podstawowe techniki dla własnych projektów elektronicznych* [10] – Kurs wideo objaśniający podstawowe pojęcia elektroniki wymagane do tworzenia własnych systemów działających z Arduino. Zawiera dużo praktycznych informacji przedstawionych w łatwy do zrozumienia sposób.
- Oficjalna strona internetowa Arduino [11] – Specyfikacja techniczna wszystkich płytek jak i dokumentacja dotycząca języka, forum użytkowników oraz obszerna baza danych gotowych przykładów i materiałów instruktażowych.

1.6 Układ pracy

Temat niniejszej pracy to system oświetlenia skonstruowany za pomocą diod elektroluminescencyjnych oraz mikrokontrolera Arduino Uno. Jej głównym celem jest przeanalizowanie istniejących systemów oświetlenia oraz stworzenie własnego rozwiązania.

Praca rozpoczyna się od uzasadnienia wyboru tematu pracy, a także opisu celów pracy, jej problematyki, zakresu, wykorzystanych metod badawczych, przeglądu literatury oraz jej układu.

W kolejnym rozdziale zawarte zostały objaśnienia zagadnień teoretycznych powiązanych z tematyką pracy.

Następnie przeanalizowano istniejące już systemy oświetlenia biorąc pod uwagę wcześniej zdefiniowane kryteria.

Kolejny rozdział opisuje projekt systemu oświetlenia Lightning tworzonego w ramach części badawczej niniejszej pracy. W rozdziale tym opisane zostają wymagania postawione przed projektem, komponenty z jakich będzie się on składał,

tworzone moduły oprogramowania, diagram klas projektu, wykorzystane wzorce projektowe czy też sama implementacja. Rozdział kończy analiza wykonanego projektu i jego dokumentacja w postaci podręcznika użytkownika, opisu dodawania własnych animacji i perspektyw rozwoju projektu.

Pracę kończy podsumowanie, które zawiera dyskusję wyników oraz dalsze prespektywy jej rozwoju.

Rozdział 2

Zagadnienia teoretyczne

Rozdział 3

Analiza istniejących rozwiązań

3.1 Kryteria analizy

Tak jak to zostało wspomniane na wstępie projektowany system oświetlenia nie jest pierwszym w swoim rodzaju i ma on do czynienia ze sporą konkurencją w postaci istniejących już rozwiązań. W rozdziale tym zostanie przeprowadzona ich analiza, jednak przed przystąpieniem do niej konieczne jest zdefiniowanie kryteriów jakie zostaną wzięte pod uwagę. Kryteria te powinny mieć pokrycie z oczekiwaniami użytkowników, a Lightning powinno wyróżniać się co najmniej pod paroma względami. Po zapoznaniu się z opiniami użytkowników w Internecie pod uwagę zostały wzięte następujące kryteria:

- Koszt – Dla większości potencjalnych użytkowników może to być kluczowe kryterium wyboru, dlatego koszt zakupu gotowego systemu lub koszt skompletowania komponentów powinien być jak najniższy.
- Rodzaj systemu – Najwyżej cenione są systemy zintegrowane z wyświetlaczami, gotowe do uruchomienia od razu po zakupie. Im trudniejsze jest skompletowanie systemu oraz oprogramowanie tym mniej użytkowników z niego skorzysta.
- Płynność działania – Tylko systemy działające płynnie są w stanie zatrzymać przy sobie swoich użytkowników.
- Możliwość konfiguracji – Sprawia, że każdy może dopasować system do siebie, im większa tym lepiej.
- Interfejs użytkownika – Rzecz na którą wielu użytkowników zwraca uwagę na początku, im bardziej przejrzysty i łatwy z obsłudze tym lepiej.

- Dodatkowe możliwości – Bonusy takie jak tryb animacji czy wizualizacji odtwarzanej muzyki wpływają na powiększenie oceny końcowej.
- Wspierany sprzęt i oprogramowanie – Wsparcie dla różnego rodzaju platform i systemów operacyjnych zwiększa grono potencjalnych odbiorców.
- Licencja – Rodzaj licencji na jakiej udostępniane jest oprogramowanie ma znaczenie w licznych przypadkach. Większość użytkowników postawiona przed wyborem pomiędzy restrykcyjną licencją a oprogramowaniem otwartoźródłowym wybierze to drugie.

3.2 Porównanie istniejących rozwiązań

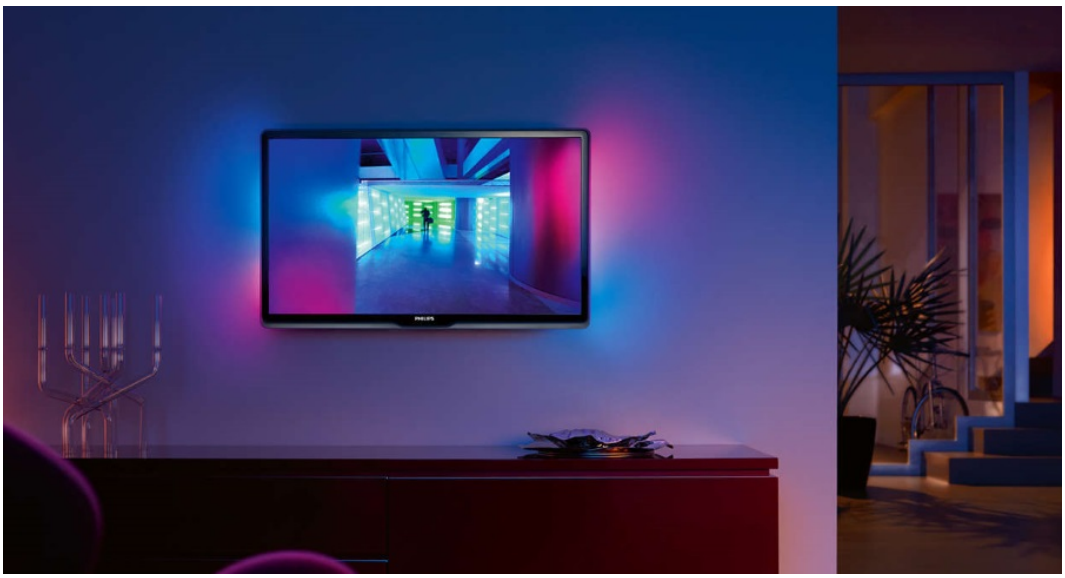
3.2.1 Technologia Philips Ambilight

Pierwsze miejsce w zestawieniu istniejących rozwiązań zajmuje technologia Ambilight opatentowana w 2002 roku przez firmę Philips. Pierwszy telewizor z Ambilight trafił do sklepów w 2004 roku. W chwili obecnej dostępnych jest dużo więcej modeli wyposażonych w Ambilight, który cieszy się sporą popularnością.

Rysunek 3.1: Logo Philips Ambilight [22]

Koszt	wysoki[1]
Rodzaj systemu	gotowy do użycia, zintegrowany
Płynność działania	bardzo wysoka
Możliwość konfiguracji	duża
Interfejs użytkownika	intuicyjny
Dodatkowe możliwości	obsługa dodatkowych lamp,
	aplikacja mobilna
Wspierany sprzęt i oprogramowanie	tylko telewizory marki Philips
Licencja	opatentowana technologia,
	brak dostępu do źródeł

Tablica 3.1: Analiza technologii Philips Ambilight



Rysunek 3.2: Telewizor z technologią Philips Ambilight [4]

Do największych zalet technologii Ambilight należy jej integracja z telewizorami Philips. Dzięki temu działa płynnie, ma duże możliwości konfiguracyjne, intuicyjny interfejs a także inne dodatkowe możliwości. Wielu użytkowników doceni

[1] W przypadku technologii Ambilight ciężko jednoznacznie określić jej dokładny koszt, ponieważ jest on wliczany do ceny telewizora. Warto jednak zauważyć, że cena telewizorów z technologią Ambilight przewyższa cene odpowiedników w nią nie wyposażonych. Ponadto dla użytkowników wyposażonych w telewizory innej marki koszt ten związany jest z kosztem zakupu nowego telewizora.

też brak konieczności podłączania dodatkowych urządzeń zewnętrznych. Część z nich uzna to za wadę, bo systemu oświetlenia nie da się przenieść do innego wyświetlacza.

Do wad Ambilight zaliczyć można wysoki koszt, wsparcie jedynie dla telewizorów marki Philips a także restrykcyjną licencję.

3.2.2 System oświetlenia Lightpack

Kolejne miejsce w zestawieniu zajmuje system oświetlenia Lightpack. Jest to projekt ufundowany przez społeczność za pomocą serwisu Kickstarter [23], gdzie zebrał ponad pół miliona dolarów.

Rysunek 3.3: Logo Lightpack [24]

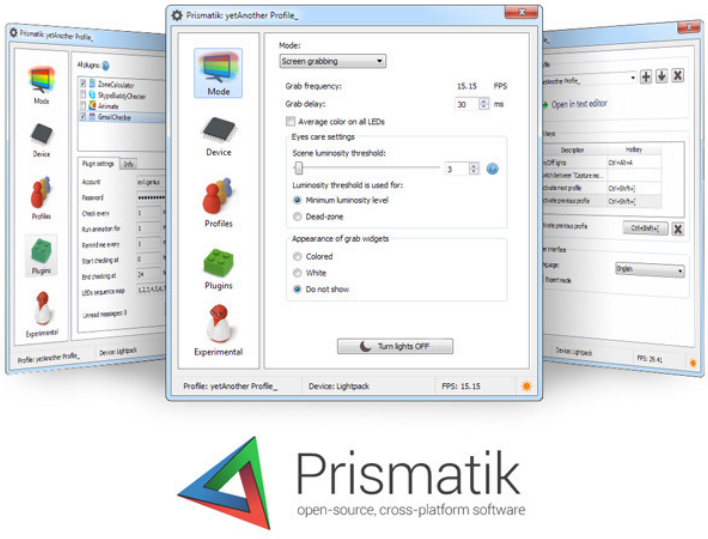
Koszt	od 119 dolarów, czyli około 482 zł
Rodzaj systemu	gotowy do użycia, zewnętrzny
Płynność działania	wysoka
Możliwość konfiguracji	duża
Interfejs użytkownika	intuicyjny
Dodatkowe możliwości	animacja przejść pomiędzy kolorami,
	pełna konfiguracja obszarów diod
Wspierany sprzęt i oprogramowanie	dowolny komputer i telewizor
Licencja	kod otwartoźródłowy

Tablica 3.2: Analiza systemu oświetlenia Lightpack

Nawiększą zaletą Lightpack jest multiplatformowość, działa on z każdym telewizorem oraz komputerami z systemami Windows, Linux jak i macOS. Posiada on także spore możliwości konfiguracyjne. Otwartoźródłowy kod również zalicza się do zalet, choć kod znajdujący się na repozytorium nie jest już uaktualniany od długiego czasu [25].

Minusem jest dość wysoka cena jak i konieczność podłączenia dodatkowego urządzenia do wyświetlacza. Diody montowane są za pomocą samoprzypelnych taśm co również nie musi spodobać się wszystkim użytkownikom.

Rysunek 3.4: Telewizor z systemem oświetlenia Lightpack [24]



Rysunek 3.5: Aplikacja sterująca systemu oświetlenia Lightpack [24]

3.2.3 System oświetlenia Lightpack 2

Lightpack 2 jest kontynuacją opisanego w rozdziale 3.2.2 projektu Lightpack. Wprowadza on jednak sporo zmian i podobnie jak swój poprzednik został ufundowany z wykorzystaniem serwisu Kickstarter [26], gdzie również zebrał ponad pół miliona dolarów.

Rysunek 3.6: Logo Lightpack 2 [27]

Koszt	od 239 dolarów, czyli około 968 zł
Rodzaj systemu	gotowy do użycia, zewnętrzny
Płynność działania	wysoka
Możliwość konfiguracji	bardzo duża
Interfejs użytkownika	intuicyjny
Dodatkowe możliwości	animacja przejść pomiędzy kolorami,
	pełna konfiguracja obszarów diod,
	aplikacja mobilna, dodatkowe lampy
Wspierany sprzęt i oprogramowanie	dowolny komputer i telewizor
Licencja	brak dostępu do źródeł,
	dostępne stare źródła aplikacji sterującej

Tablica 3.3: Analiza systemu oświetlenia Lightpack 2

System Lighpack 2 posiada zalety swojego poprzednika, jednak dodaje on zupełnie nowe możliwości. Ciekawym dodatkiem jest obsługa zewnętrznych lamp

rozmieszczonych w tym samym pomieszczeniu co wyświetlacz. Dodano aplikację mobilną. Najważniejszy jest jednak fakt, że nie wymaga on połączenia z komputerem. Tym razem podłączany on jest do dowolnego źródła sygnału HDMI, w tym konsoli czy dekodera telewizyjnego.

Dużą wadą systemu jest spora cena, która rośnie wraz z chęcią dokupienia dodatkowych lamp czy też większej ilościo diod. Ponadto na Lighpack 2 trzeba jeszcze poczekać, ponieważ jest on w ostatniej fazie produkcji i na razie możliwe jest jedynie składanie zamówień.

Rysunek 3.7: Telewizor z systemem oświetlenia Lightpack 2 [27]

Rozdział 4

System oświetlenia Lightning

4.1 Analiza wymagań

4.1.1 Wymagania funkcjonalne

W celu stworzenia jak najatrakcyjniejszego system oświetlenia podczas projektowania Lightning przyjęto poniższe wymagania funkcjonalne:

- System musi udostępniać tryb przechwytywania rozszerzający obraz wyświetlany na ekranie na jego otoczenie za pomocą diod elektroluminescencyjnych.
- System musi posiadać tryb animacji. Powinien być on łatwy do rozszerzenia o kolejne animacje.
- System musi być konfigurowalny z poziomu aplikacji sterującej. Konfiguracji podlegać musi co najmniej liczba i rozmieszczenie diod za ekranem, a także wykorzystywany port szeregowy.
- Projekt musi być odpowiednio udokumentowany. Dokumentacja powinna ułatwiać szybkie skonfigurowanie oraz uruchomienie systemu.

4.1.2 Wymagania niefunkcjonalne

Do wymagań niefunkcjonalnych postawionych przed projektowanym systemem należą:

- System musi działać płynnie, a więc liczba osiąganych klatek na sekundę powinna być jak największa nawet przy dużych rozdzielczościach przechwytywanego ekranu.

- Korzystanie z aplikacji sterującej powinno być jak najbardziej intuicyjne, a użytkownik powinien mieć dostęp do wskazówek dotyczących jej interfejsu.
- Oprogramowanie mikrokontrolera powinno oferować jak najprostszy interfejs i zawierać jak najmniej logiki sterującej.
- Pamięć na obu urządzeniach sterujących powinna być odpowiednio zarządzana, niedopuszczalne są wycieki pamięci czy też zapętlenia programu.

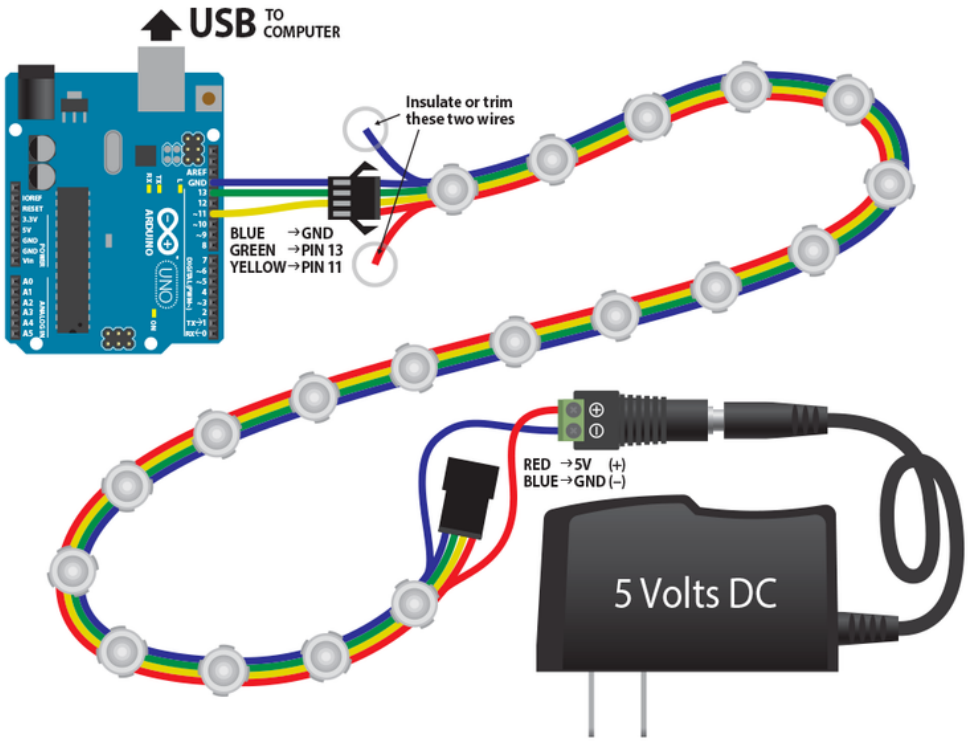
4.2 Komponenty systemu

Do konstrukcji systemu wykorzystane zostały następujące komponenty:

- Cyfrowo adresowany łańcuch składający się z 25 diod elektroluminescencyjnych o średnicy 12 mm ze sterownikiem WS2801. Dostępny w sklepie internetowym Botland w cenie 195 zł [12].
- Stabilizowany zasilacz sieciowy o napięciu wyjściowym 5 V. Dostępny w sklepie internetowym Botland w cenie 19,90 zł [13].
- Wtyk ze złączem pozawalającym połączyć łańcuch z zasilaczem. Dostępny w sklepie Botland w cenie 1,90 zł [14].
- Mikrokontroler Arduino Uno w wersji 3. Dostępny w sklepie Botland w cenie 95 zł [15].
- Przewód USB. Dostępny w sklepie Botland w cenie 4,90 zł [16].

Łańcuch z diodami został podłączony od jednej strony za pomocą wymienionego wcześniej wtyku z zasilaczem, z drugiej strony natomiast z mikrokontrolerem. Ten z kolei komunikuje się z komputerem z systemem macOS za pomocą przewodu USB. Dokładny schemat połączenia komponentów przedstawia rysunek 4.2.

Koszt związany ze skompletowaniem wszystkich komponentów wyniósł łącznie 316,70 zł. Cena ta mogłaby ulec znacznemu zmniejszeniu w przypadku użycia kompatybilnych zamienników dla najdroższych elementów, czyli dla łańcucha diod czy też Arduino Uno w najnowszej wersji. Ponadto komponenty te mogą zostać wykorzystane w innych systemach.



Rysunek 4.1: Schemat podłączenia komponentów systemu [17]

4.3 Moduły oprogramowania

4.3.1 Oprogramowanie mikrokontrolera

Głównym celem oprogramowania przeznaczonego na mikrokontroler jest zarządzanie diodami. Dzieje się to w oparciu o polecenia otrzymywane portem szeregowym od komputera, który steruje całym systemem. W związku z charakterem swojego zadania oprogramowanie musi działać jak najszybciej oraz posiadać jak najmniej logiki sterującej. Kontrola diod ogranicza się do przesyłania pakietów danych otrzymywanych z komputera na odpowiednie wyjścia do których diody są podłączone do mikrokontrolera.

Moduł ten został napisany w natywnym języku Arduino, który udostępnia wszystkie podstawowe funkcje umożliwiające stworzenie oprogramowania spełniającego postawione przed nim wymagania.

4.3.2 Aplikacja przeznaczona na komputer

W przeciwieństwie do swojego poprzednika moduł przeznaczony na komputer ma za zadanie sterowanie całym systemem. Odpowiada on bezpośrednio na żądania użytkownika wydawane za pomocą stworzonego interfejsu graficznego i komunikuje się z oprogramowaniem mikrokontrolera w celu przekazania instrukcji zapalenia pewnej konfiguracji diod. Komunikacja odbywa się nieprzerwanie, jednokierunkowo, a każdy pakiet jest poprzedzony nagłówkiem składającym się z 11 bajtów.

Aplikacja ta może działać w dwóch trybach:

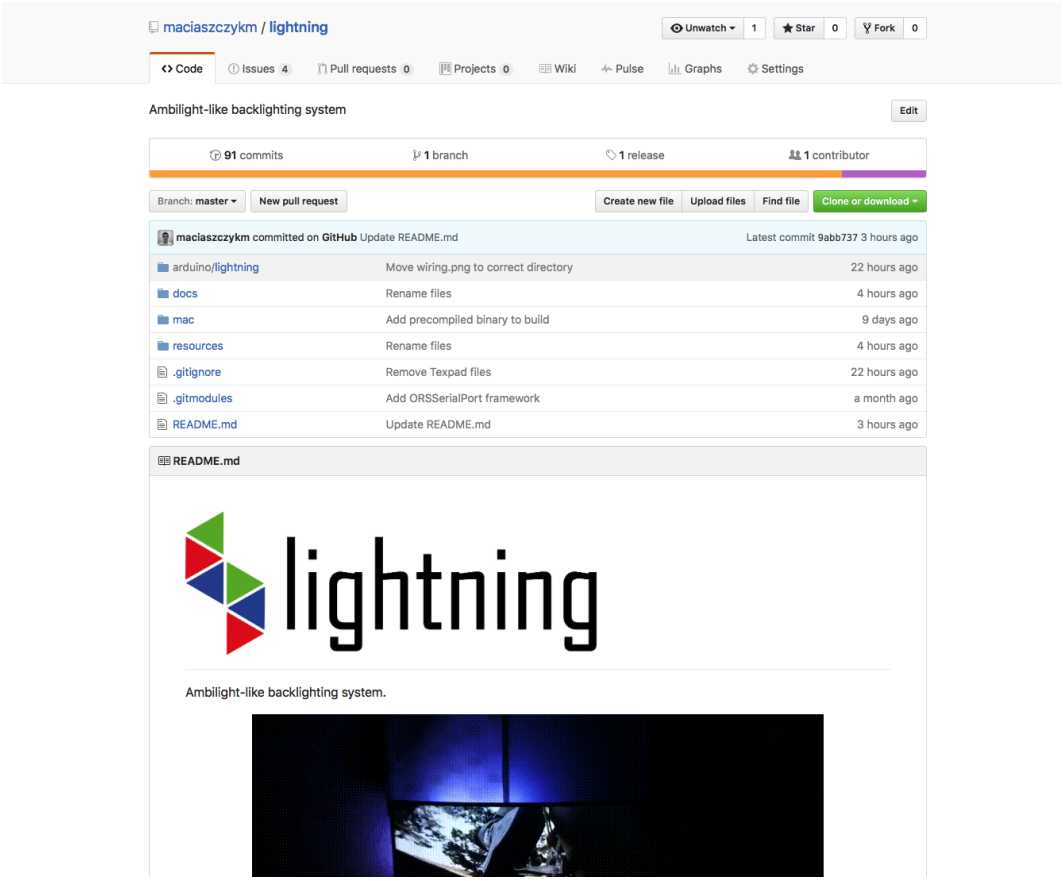
- Tryb przechwytywania w którym przechwytywany jest obraz wybranego wyświetlacza, a każda dioda świeci się w kolorze odpowiadającego jej koloru ekranu. W trybie tym użytkownik powinien mieć kontrolę nad jasnością diod a także możliwość wygładzania przejść aby uniknąć szybkich zmian kolorów.
- Tryb animacji w którym za pomocą diod system wyświetla przygotowane wcześniej animacje. Animacje zapisane są jako klasy w języku Swift. Użytkownik poza wyborem animacji ma wpływ na użyte w niej kolory oraz prędkość animacji.

Ponadto aplikacja posiada możliwość konfiguracji liczby i rozmieszczenia diod, a także wykorzystywanego portu szeregowego. Aplikacja została napisana w języku Swift, który jest przeznaczony dla komputerów z systemem macOS. Umożliwia on szybkie projektowanie interfejsów graficznych zgodnych z wyglądem systemu oraz przede wszystkim udostępnia on wszystkie podstawowe funkcje obiektowego języka programowania co jest istotne podczas projektowania aplikacji, której logika nie jest już tak prosta jak w przypadku oprogramowania mikrokontrolera.

4.4 Projekt

4.4.1 System kontroli wersji

Podczas pracy nad projektami programistycznymi często wymagana jest współpraca kilku programistów, cofanie pomyłkowo wprowadzonych zmian czy też dziennik zadań do wykonania. Wspomniane funkcjonalności udostępniają systemy kontroli wersji. Do najpopularniejszych należy Git, którego funkcjonalność darmowo udostępnia serwis GitHub, gdzie z kolei znajduje się repozytorium projektu [18].



Rysunek 4.2: Zrzut ekranu z repozytorium projektu

4.4.2 Wykorzystane technologie

Oprogramowanie mikrokontrolera zostało napisane w języku Arduino i to właśnie na tę platformę jest przeznaczone. Do jego napisania użyta została jedynie wbudowana biblioteka do obsługi portu szeregowego. Wykorzystano środowisko programistyczne Arduino [19].

Aplikacja sterująca została napisana w języku Swift, do stworzenia interfejsu graficznego wykorzystano framework[1] Cocoa. Ponadto wykorzystano bibliotekę ORSSerialPort ułatwiającą komunikację poprzez port szeregowy [20]. Wykorzystano środowisko programistyczne Xcode [21].

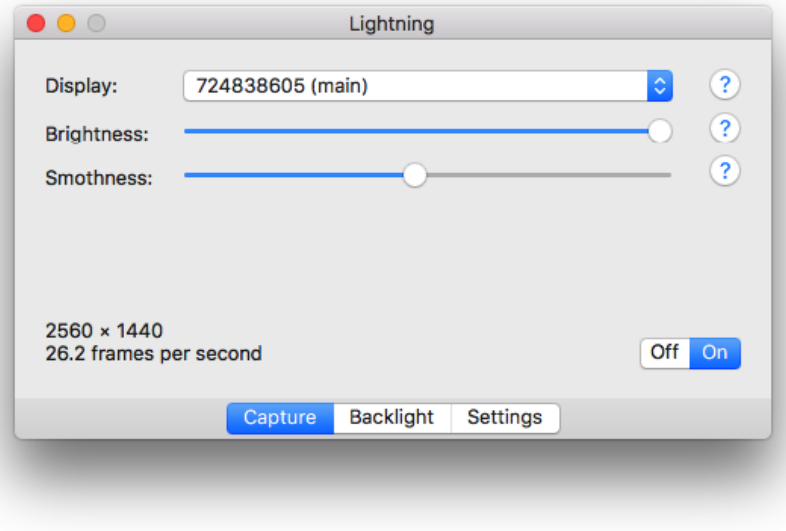
[1] Szkielet budowy aplikacji. Definiuje strukturę aplikacji oraz jej ogólny mechanizm działania, dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia do wykonywania określonych zadań.

4.4.3 Diagram klas

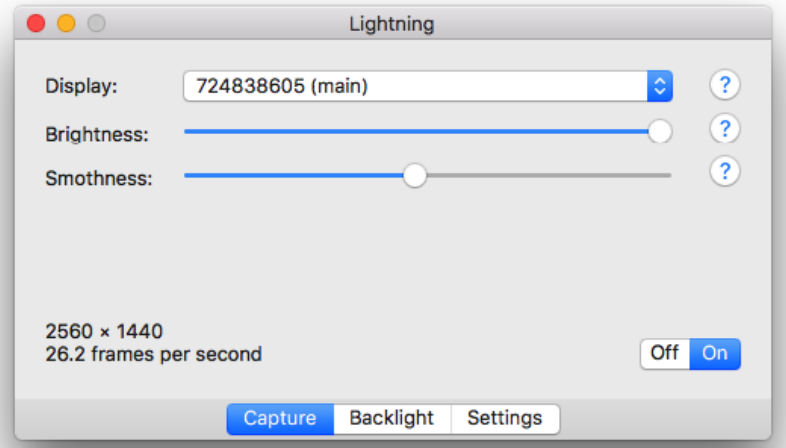
4.4.4 Wzorce projektowe

4.4.5 Interfejs użytkownika

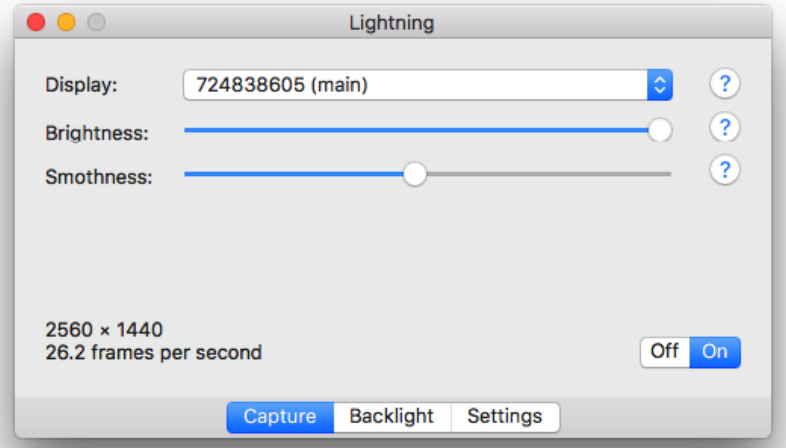
Interfejs, w tym przypadku graficzny, należy do elementów na które użytkownicy zwracają uwagę na samym początku, dlatego też powinien być on zaprojektowany z pomysłem i umożliwiać jak najprostsze poruszanie się po aplikacji. Aplikacja posiada dwa tryby działania oraz tryb konfiguracji, dlatego też logiczny jest podział na trzy widoki przedstawione poniżej:



Rysunek 4.3: Widok trybu przechwytywania



Rysunek 4.4: Widok trybu animacji



Rysunek 4.5: Widok konfiguracji

4.5 Implementacja

4.5.1 Oprogramowanie mikrokontrolera

Oprogramowanie mikrokontrolera napisane w natywnym języku Arduino mieści się w jednym pliku przedstawionym na listingu 4.5.1.

```
1  // Arduino "bridge" code between streamer device and WS2801-based digital RGB LED.
2  // Intended for use with Arduino Uno, but may work with other Arduino models as well.
3
4  #include <SPI.h>
5
6  // Lightning sends a packages with specific header. It consits of magic word, number of LEDs and checksum.
7  uint8_t magicWord[] = {'L','i','g', 'h', 't', 'n', 'i', 'n', 'g'}, leds, checksum;
8  #define MAGIC_WORD_SIZE  sizeof(magicWord)
9  #define HEADER_SIZE (MAGIC_WORD_SIZE + 2)
10
11 // Serial port bit rate.
12 #define PORT_SPEED 115200
13
14 // Port registers allow lower-level and faster manipulation of I/O pins of the microcontroller.
15 // Each port have three registers: DDR determining whether pin is input or output, PORT controlling if pin is in high or low state and PIN registering state of input pins set to input with pinMode().
16 // Lightning uses port B (digital pins from 8 to 13) for LED control.
17 // More information at https://www.arduino.cc/en/Reference/PortManipulation.
18 #define LED_DDR  DDRB
19 #define LED_PORT PORTB
20 #define LED_PIN  _BV(PORTB5)
21
```

```
22 // Lightning LEDs timeout, by default 5 seconds.
23 #define LED_TIMEOUT 5000
24
25 // Lightning main loop states.
26 #define HEADER_VALIDATION_STATE 0
27 #define DATA_BUFFERING_STATE   1
28 uint8_t state = HEADER_VALIDATION_STATE;
29
30 // Circular buffer for serial port data is 256 bytes
       long.
31 uint8_t buffer[256];
32
33 // Other global variables.
34 uint8_t indexIn = 0, indexOut = 0, spiFlag;
35 int16_t bytesBuffered = 0, index, value;
36 int32_t bytesRemaining;
37 unsigned long lastByteTime;
38
39 void setup() {
40   LED_DDR  |=  LED_PIN; // Make LED pin an output.
41   LED_PORT &= ~LED_PIN; // Turn off LEDs.
42
43   Serial.begin(PORT_SPEED);
44
45   SPI.begin();
46   SPI.setBitOrder(MSBFIRST);
47   SPI.setDataMode(SPI_MODE0);
48   SPI.setClockDivider(SPI_CLOCK_DIV16);
49
50   lastByteTime = millis();
51 }
52
53 void loop() {
54   if((bytesBuffered < 256) && ((value = Serial.read
       ()) >= 0)) { // Buffer byte if there are any on
       serial port.
55     buffer[indexIn++] = value;
56     bytesBuffered++;
57     lastByteTime = millis();
```

```
58    } else {
59      if((millis() - lastByteTime) > LED_TIMEOUT) {
60        for(index=0; index<32767; index++) {
61          for(SPDR=0; !(SPSR & _BV(SPIF)); ); // Turn
       off LEDs when no data recieved for more than
       LED_TIMEOUT.
62        }
63        delay(1);
64        lastByteTime = millis();
65      }
66    }
67
68    switch(state) {
69      case HEADER_VALIDATION_STATE:
70        if(bytesBuffered >= HEADER_SIZE) {
71          for(index=0; (index<MAGIC_WORD_SIZE) && (
       buffer[indexOut++] == magicWord[index++]);); //
       Move indexOut through buffer if it matches magic
       word.
72          if(index == MAGIC_WORD_SIZE) {
73            leds  = buffer[indexOut++];
74            checksum = buffer[indexOut++];
75            if(checksum == (leds ^ 0x13)) {
76              bytesRemaining = 3L * ((long)leds + 1L);
77              bytesBuffered -= 2;
78              spiFlag = 0; // Allows to skip check if
       first byte has been already buffered when skipping
        from HEADER_VALIDATION_STATE to
       DATA_BUFFERING_STATE.
79              state = DATA_BUFFERING_STATE;
80            } else {
81              indexOut  -= 2;
82            }
83          }
84          bytesBuffered -= index;
85        }
86        break;
87
88      case DATA_BUFFERING_STATE:
```

```
89        while(spiFlag && !(SPSR & _BV(SPIF))); // Wait
      for prior byte to buffer.
90        if(bytesRemaining > 0) {
91          if(bytesBuffered > 0) {
92            SPDR = buffer[indexOut++];
93            bytesBuffered--;
94            bytesRemaining--;
95            spiFlag = 1;
96          }
97        } else {
98          LED_PORT |= LED_PIN;
99          state = HEADER_VALIDATION_STATE;
100       }
101     }
102 }
```

Listing 4.1: Kod oprogramowania mikrokontrolera

Tak jak to zostało wcześniej wspomniane celem mikrokontrolera jest odbieranie sygnału portem szeregowym i zapalanie diod w odpowiedniej konfiguracji. Wszystko musi odbywać się możliwie szybko, tak aby diody swój kolor zmieniały płynnie i nie pojawiały się żadne opóźnienia.

Pierwszym przyjętym założeniem jest odpowiednie formatowanie pakietów danych, które będą wysyłane przez port szeregowy. Aby żadne dane nie zostały utracone każdy pakiet musi składać się z nagłówka i danych właściwych. Nagłówki składają się z 11 bajtów, długość całego pakietu jest natomiast uzależniona od ilości diod zdefiniowanej w nagłówku:

- Pierwsze 9 bajtów – Zakodowana nazwa projektu, a więc Łightning"jako tablica znaków.
- 10 bajt – Liczba diod wykorzystywana przez użytkownika.
- 11 bajt – Suma kontrolna nagłówka, czyli wartość 10 bajta XOR[2] 0x13. W przypadku niezgodności wartości, pakiet zostaje pominięty przez mikrokontroler.

[2] Alternatywa wykluczająca (z ang. exclusive or).

- Od 12 bajta – Zakodowane kolory diod, na które mikrokontroler musi je zaświecić. Na każdą diodę przypadają 3 bajty danych:
  - 1 bajt – Wartość koloru czerwonego w skali od 0 do 255.
  - 2 bajt – Wartość koloru zielonego w skali od 0 do 255.
  - 3 bajt – Wartość koloru niebieskiego w skali od 0 do 255.

Jeśli użytkownik korzysta z 25 diod, pakiety danych będą miały długość 86 bajtów, co jest małą liczbą biorąc pod uwagę zakładaną prędkość 115200 bitów na sekundę jeśli chodzi o transmisję portem szeregowym. Kluczem dla płynności działania aplikacji okazuje się więc odczyt i sprawne przekazanie instrukcji diodom.

4.5.2 Aplikacja przeznaczona na komputer

4.6 Podręcznik użytkownika

Odpowiedzi na poniższe pytania mają za zadanie przybliżyć potencjalnym użytkownikom czym jest Lightning, co oferuje, jak działa i jak mogą oni zacząć z niego korzystać. Jest to skrót informacji pojawiających się w niniejszej pracy.

Czym jest Lightning?

Lightning jest systemem oświetlenia posiadającym podobną funkcjonalność jak technologia Philips Ambilight, a więc rozszerzanie obrazu wyświetlanego na ekranie na jego otoczenie za pomocą diod elektroluminescencyjnych.

Jak działa Lightning?

Lightning składa się z kilku komponentów, poza komputerem podłączonym do wyświetlacza niezbędny jest mikrokontroler Arduino Uno oraz taśma diod elektroluminescencyjnych[3]. Działanie systemu opiera się na przechwytywaniu obrazu z komputera z systemem macOS, uśrednianiem kolorów występujących w sferach na które podzielony został ekran[4] i przesyłaniu informacji do mikrokontrolera. Jego zadaniem jest odbieranie informacji na temat kolorów diod i sterowanie diodami. System jest w pełni niezależny od wykorzystywanego wyświetlacza.

[3] Wszystkie komponenty systemu zostały wyszczególnione w rozdziale 4.2

[4] Każda ze sfer odpowiada jednej diodzie.

Ile diód można podłączyć do Lightning?

Do 255 diod, wystarczy to do oświetlenia powierzchni nawet za największym ekranem i zapewni uzyskanie wysokiej płynności działania nawet ze starszymi komputerami Mac.

Jak zacząć korzystać z Lightning?

Pierwszym etapem powinno być skompletowanie wszystkich komponentów systemu wymienionych w rozdziale 4.2. Następnie należy je ze sobą podłączyć tak jak przedstawia to rysunek 4.2. Ostatnim krokiem jest wgranie oprogramowania na Arduino – najłatwiej zrobić to za pomocą środowiska programistycznego Arduino, a także uruchomienie aplikacji sterującej na komputerze Mac.

Jak dodać własne animacje?

Proces dodawania własnych animacji został opisany w rozdziale 4.7.

4.7 Przykładowa implementacja animacji

Tryb animacji udostępnia kilka podstawowych animacji, które po uruchomieniu wyświetlane są za pomocą diod. W celu dodania nowej animacji wystarczy dodać do projektu nową klasę na wzór klasy Disco przedstawionej na listingu 4.7 oraz zarejestrować ją w klasie Animations przedstawionej na listingu 4.7

```
1  import Foundation
2
3  class Disco: LightController, Animation {
4
5      private var state = 0
6
7      func setup(colors: [Color]) {
8          for light in self.lights.lights {
9              light.color = colors[state]
10         }
11         self.proceed()
12     }
13
14     func run(colors: [Color]) {
```

```
15          for light in self.lights.lights {
16              light.color = colors[self.state]
17          }
18          self.proceed()
19          self.serialPort.send(lights: lights)
20      }
21
22      func proceed() {
23          if state < 3 {
24              self.state += 1
25          } else {
26              self.state = 0
27          }
28      }
29
30  }
```

Listing 4.2: Klasa implementująca animację

Każda implementacja animacji musi implementować protokół Animation co wiąże się z koniecznością implementacji następujących metod:

- setup(colors) – Metoda ta uruchamiana jest jednokrotnie przy starcie animacji. Jej zadaniem jest odpowiednie zainicjalizowanie wszystkich diod. Kolory wybrane przez użytkownika przekazane są w parametrze.
- run(colors) – Metoda ta uruchamiana jest przy każdym obiegu pętli w której odbywa się animacja. Jej głównym zadaniem jest przechodzenie pomiędzy kolejnymi stanami animacji. Kolory wybrane przez użytkownika przekazane są w parametrze.

Ponadto klasa animacji powinna roszerzać klasę LightController, która zawiera wszystkie podstawowe funkcje umożliwiające sterowanie diodami.

Animacja przedstawiona na listingu 4.7 inicjalizuje wszystkie diody na pierwszy kolor wybrany przez użytkownika, a potem w każdym następnym kroku zmienia kolor na kolejny i przesyła dane do mikrokontrolera.

```
1  import Foundation
```

```
2
3  class Animations {
4
5      // Each animation has to be registered here.
6      static var animations: [String: Animation] = ["
       Disco": Disco(), "Snake": Snake()]
7
8      static func getAnimationNames() -> [String] {
9          return Array(animations.keys)
10     }
11
12 }
```

Listing 4.3: Klasa służąca do rejestrowania animacji

Powyższy przykład obrazuje niski poziom skomplikowania implementacji nowych animacji.

4.8 Analiza projektu

W rozdziale 3.1 zostały zdefiniowane kryteria analizy której poddane zostały istniejące już systemy oświetlenia. Teraz biorąc pod uwagę te same kryteria zostanie przeanalizowany Lightning.

Rysunek 4.6: Logo Lightning

Koszt	316,70 zł[5]
Rodzaj systemu	do samodzielnego montażu,
	zewnętrzny
Płynność działania	bardzo wysoka
Możliwość konfiguracji	średnia
Interfejs użytkownika	intuicyjny
Dodatkowe możliwości	tryb animacji, duża rozszerzalność
Wspierany sprzęt i oprogramowanie	komputery z systemem macOS
Licencja	kod otwartoźródłowy

Tablica 4.1: Analiza systemu oświetlenia Lightning

Do niewątpliwych zalet Ligthning należy z pewnością niski koszt, otwartożródłowy kod i łatwa rozszerzalność systemu. Użytkownicy znający podstawy programowania mogą z łątwością dopasować projekt do swoich potrzeb, nie jest to jednak konieczne ponieważ większość opcji konfiguracyjnych jest już dostępna. Ciekawym dodatkiem jest tryb animacji. Animacje można dodawać według własnych upodobań. Dodatkowo system można podłączyć do dowolnego wyświetlacza połączonego z komputerem.

Podobnie jak w przypadku wszystkich innych analizowanych rozwiązań poza technologią Philips Ambilight Ligtning jest zewnętrznym systemem oświetlenia co wiąże się z koniecznością podłączenia dodatkowych urządzeń do wyświetlacza. Kolejną z wad jest przeznaczenie na komputery z systemem macOS, jest to związane z chęcią uzyskania jak najwyższej płynności działania.

4.9 Perspektywy rozwoju projektu

W celu dalszego rozwoju stworzonego systemu oświetlenia należy rozważyć wprowadzenie następujących usprawnień:

- Zaawansowana konfiguracja ułożenia diod – Obecnie diody rozmieszczane są równomiernie na górnej i bocznych krawędziach. Wprowadzenie trybu konfiguracji w którym istniałaby możliwość dopasowania każdego obszaru przechwytywania osobno byłoby znacznym usprawnieniem.

[5] Kwota ta mogłaby ulec znacznemu zmniejszeniu w przypadku wykorzystania dostępnych zamienników dla mikrokontrolera Arduino Uno a także taśmy diod elektroluminescencyjnych, tak jak zostało to opisane w rozdziale 4.2.

- Korekcja kolorów – Obecny stan systemu umożliwia kontrolę jasności diod oraz wygładzania przejść pomiędzy kolejnymi stanami. Wprowadzenie korekcji kolorów umożliwiłoby dopasowanie koloru diod do otoczenia, czyli na przykład czerwonej ściany za wyświetlaczem.
- Częstotliwość przechwytywania – Poza korekcją kolorów kolejnym wartościowym dodatkiem byłoby wprowadzenie ograniczenia częstotliwości przechwytywania.
- Poprawki graficzne szaty graficznej – Dodanie ikony dla aplikacji sterującej, użycie obrazka wyświetlacza trybie konfiguracji czy też wyświetlanie poglądu animacji.
- Automatyczne testy – Automatyczne budowanie i testowanie całej aplikacji przy każdej zmianie podniosłoby jakość projektu.

Rozdział 5

Podsumowanie

5.1 Dyskusja wyników

Przejście przez kolejne fazy tworzenia oprogramowania, od postawienia wymagań, przez zaprojektowanie, zaimplementowanie aż do stworzenia dokumentacji umożliwiło stworzenie systemu oświetlenia, który stanowi konkurencyjną alternatywę dla istniejących już rozwiązań. Lightning udostępnia w pełni funkcjonalny tryb przechwytywania ekranu, tryb animacji, posiada możliwości konfiguracyjne, działa płynnie a koszt jego konstrukcji jest najmniejszy spośród wszystkich analizowanych systemów oświetlenia.

Poza osiągnięciem rezultatu w postaci gotowego systemu oświetlenia, w niniejszej pracy autorowi udało się zgromadzić użyteczne informacje dotyczące tematyki tworzenia oprogramowania przeznaczonego na mikrokontrolery, komunikacji portem szeregowym czy też zagadnieniom dotyczącym efektywnego przechwytywania obrazu.

5.2 Perspektywy rozwoju pracy

W celu dalszego rozwoju niniejszej pracy należy rozważyć przede wszystkim dalsze prace nad stworzonym systemem oświetlenia. W rozdziale 4.9 przedstawione zostały liczne możliwości rozwoju projektu, ich realizacja byłaby z pewnością sporym krokiem w przód. Podjęcie się tego zadania oznaczałoby jednak trzymanie się obecnej tematyki pracy, a przecież wykorzystanie mikrokontrolerów podłączonych do komputera nie jest jedynym możliwym rozwiązaniem.

Odejście od obecnej tematyki pracy zostałoby zapewnione poprzez stworzenie oprogramowania przeznaczonego bezpośrednio dla telewizorów. Aplikacja steru-

jąca mogłaby wtedy zostać przeniesiona na urządzenia mobilne. W tym przypadku wymagane byłoby dokładne zapoznanie się z dostępnym sprzętem i istniejącymi ograniczeniami. Z pewnością jednak takie rozwiązanie spotkałoby się z szerokim gronem odbiorców.

Rozwiązaniem pośrednim byłoby przeniesienie oprogramowania przechytującego obraz do mikrokontrolera o większej mocy obliczeniowej, na przykład Raspberry Pi. W tym przypadku sterowanie diodami jak i przechytywanie obrazu mogłoby się odbywać bezpośrednio na nim. Mikrokontroler mógłby samodzielnie odtwarzać filmy bez konieczności podłączania kompuera.

Bibliografia

[1] http://www.wirtualnemedia.pl/artykul/coraz-dluzej-ogladamy-telewizje-najwiecej-czasu-przed-szklanym-ekranem-spedzaja--seniorzy-raport.
Data dostępu – 20.01.2017.

[2] https://www.indiegogo.com/projects/lightpack-2-light-orchestra-for-your-living-room-tv-videogames.
Data dostępu – 20.02.2017.

[3] http://www.embedded.com/print/4012996.
Data dostępu – 20.02.2017.

[4] http://www.komputerswiat.pl/media/2016/224/4688581/003-philips-ambilight-2-sides.jpg.
Data dostępu – 06.02.2017.

[5] https://swift.org/documentation/TheSwiftProgrammingLanguage-(Swift3.1).epub.
Data dostępu – 20.02.2017.

[6] https://itunes.apple.com/us/book/app-development-with-swift/-id1118575552?mt=11.
Data dostępu – 20.02.2017.

[7] https://swift.org.
Data dostępu – 20.02.2017.

[8] https://developer.apple.com/swift/.
Data dostępu – 20.02.2017.

[9] Evans, Mike, Noble, Joshua, Hochenbaum, Jordan. Arduino w akcji. Wyd. 1. Gliwice, 2014. ISBN 978-83-246-6356-9.

[10] http://videopoint.pl/kurs/arduino-kurs-video-poziom-pierwszy-podstawowe-techniki-dla-wlasnych-projektow-elektronicznych-pawel-matyszok,arduin.htm.
Data dostępu – 20.02.2017.

[11] https://www.arduino.cc.
Data dostępu – 20.02.2017.

[12] https://botland.com.pl/lancuchy-i-matryce-led/2443-lacuch-led-rgb-12-mm-ws2801-cyfrowy-adresowany-25-szt.html.
Data dostępu – 17.01.2017.

[13] https://botland.com.pl/zasilacze-sieciowe-5-v/1364-zasilacz-impulsowy-5v-25a-wtyk-dc-55-21-mm.html.
Data dostępu – 17.01.2017.

[14] https://botland.com.pl/szybkozlacza/1590-wtyk-dc-55-x-21-mm-z-szybkozlaczem.html.
Data dostępu – 17.01.2017.

[15] https://botland.com.pl/arduino-moduly-glowne/1060-arduino-uno-r3.html.
Data dostępu – 17.01.2017.

[16] https://botland.com.pl/przewody-usb-a-b-20/5313-przewod-usb-a-b-tracer-18m.html.
Data dostępu – 17.01.2017.

[17] https://cdn-learn.adafruit.com/assets/assets/000/001/484/-medium800/led_pixels_wiring-diagram.png?1396773276.
Data dostępu – 17.01.2017.

[18] https://github.com/maciaszczykm/lightning.
Data dostępu – 17.01.2017.

[19] https://www.arduino.cc/en/main/software.
Data dostępu – 18.01.2017.

[20] https://github.com/armadsen/ORSSerialPort.
Data dostępu – 18.01.2017.

[21] https://developer.apple.com/xcode/.
Data dostępu – 18.01.2017.

[22] http://www.cobra.fr/media/wysiwyg/logo-ambilight.jpg.
Data dostępu – 06.02.2017.

[23] https://www.kickstarter.com/projects/woodenshark/lightpack-ambient-backlight-for-your-displays.
Data dostępu – 09.02.2017.

[24] http://lightpack.tv/.
Data dostępu – 09.02.2017.

[25] https://github.com/Atarity/Lightpack.
Data dostępu – 09.02.2017.

[26] https://www.kickstarter.com/projects/woodenshark/lightpack-2-ultimate-light-orchestra-for-your-livi.
Data dostępu – 09.02.2017.

[27] http://lightpack.tv/promo/index.php.
Data dostępu – 09.02.2017.

Spis rysunków

1.1 Średni czas dopasowania źrenicy do wyświetlanego obrazu 4
1.2 Telewizor z technologią Philips Ambilight 4
1.3 Średni czas dopasowania źrenicy do wyświetlanego obrazu z otaczającym oświetleniem 5

3.1 Logo Philips Ambilight 13
3.2 Telewizor z technologią Philips Ambilight 14
3.3 Logo Lightpack 15
3.4 Telewizor z systemem oświetlenia Lightpack 16
3.5 Aplikacja sterująca systemu oświetlenia Lightpack 16
3.6 Logo Lightpack 2 17
3.7 Telewizor z systemem oświetlenia Lightpack 2 18

4.1 Schemat podłączenia komponentów systemu 21
4.2 Zrzut ekranu z repozytorium projektu 23
4.3 Widok trybu przechwytywania 24
4.4 Widok trybu animacji 25
4.5 Widok konfiguracji 25
4.6 Logo Lightning 33

Spis tablic

3.1 Analiza technologii Philips Ambilight 14
3.2 Analiza systemu oświetlenia Lightpack 15
3.3 Analiza systemu oświetlenia Lightpack 2 17

4.1 Analiza systemu oświetlenia Lightning 34

Spis listingów

4.1 Kod oprogramowania mikrokontrolera 26
4.2 Klasa implementująca animację 31
4.3 Klasa służąca do rejestrowania animacji 32