WYŻSZASZKOŁAINFORMATYKIIUMIEJĘTNOŚCI
WydziałInformatykiiZarządzania
Kierunek:Informatyka

MarcinMaciaszczyk
nralbumu:29572

PracaMagisterska
Systemoświetleniaskonstruowanyzapomocądiod elektroluminescencyjnychorazmikrokontroleraArduinoUno

Pracanapisanapodkierunkiem
drinż.GrzegorzZwoliński

Rokakademicki2016/2017

Spis treści

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).

Do celów niniejszej pracy należy złożenie i oprogramowanie systemu oświetlenia, który ma rozszerzać obraz widziany na ekranie na jego otoczenie. 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[4] 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.

1.2 Problematyka i zakres pracy

1.3 Cele pracy

1.4 Metoda badawcza

1.5 Przegląd literatury w dziedzinie

1.6 Układ pracy

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

Rozdział 2

Zagadnienia teoretyczne

Rozdział 3

Analiza istniejących rozwiązań

3.1 Kryteria analizy

3.2 Porównanie istniejących rozwiązań

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.

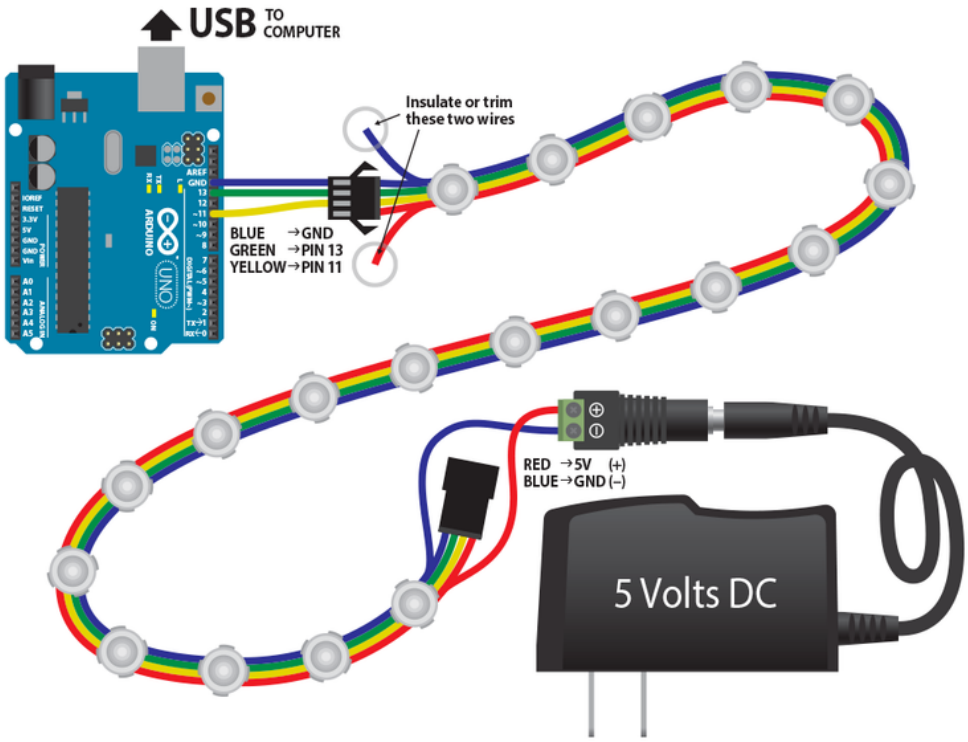
4.2 Komponenty systemu

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 [2].
- Stabilizowany zasilacz sieciowy o napięciu wyjściowym 5 V. Dostępny w sklepie internetowym Botland w cenie 19,90 zł [3].
- Wtyk ze złączem pozawalającym połączyć łańcuch z zasilaczem. Dostępny w sklepie Botland w cenie 1,90 zł [4].
- Mikrokontroler Arduino Uno w wersji 3. Dostępny w sklepie Botland w cenie 95 zł [5].
- Przewód USB. Dostępny w sklepie Botland w cenie 4,90 zł [6].

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.



Rysunek 4.1: Schemat podłączenia komponentów systemu [7]

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:

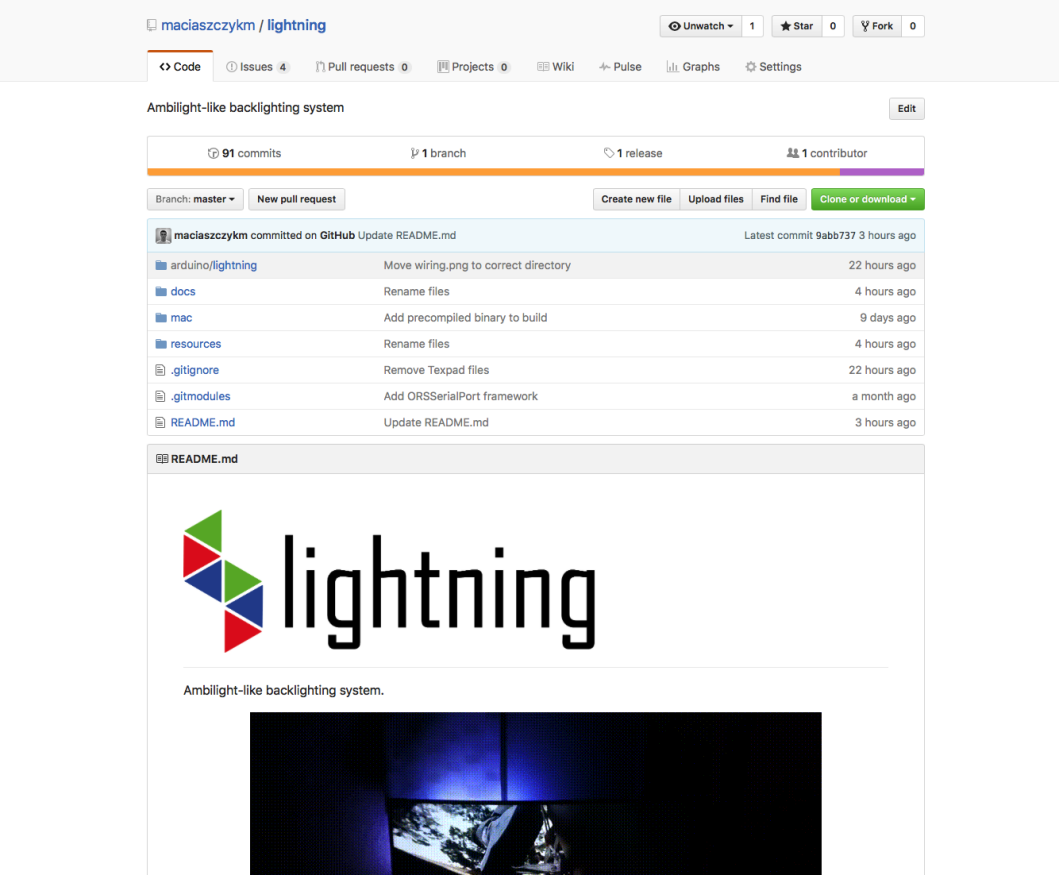
- Tryb przechwytywania w którym przechwytywany jest obraz wybranego wyświetlacza, a każda dioda świeci się w kolorze odpowiadającego jej koloru ekranu. W trybie tym użytkownik powinien mieć kontrolę nad jasnością diod a także możliwość wygładzania przejść aby uniknąć szybkich zmian kolorów.
- Tryb animacji 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 [8].



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 [9].

Aplikacja sterująca została napisana w języku Swift, do stworzenia interfejsu graficznego wykorzystano framework[1] Cocoa. Ponadto wykorzystano bibliotekę ORSSerialPort ułatwiającą komunikację poprzez port szeregowy [10]. Wykorzystano środowisko programistyczne Xcode [11].

[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

Rysunek 4.3: Logo projektu

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:

-

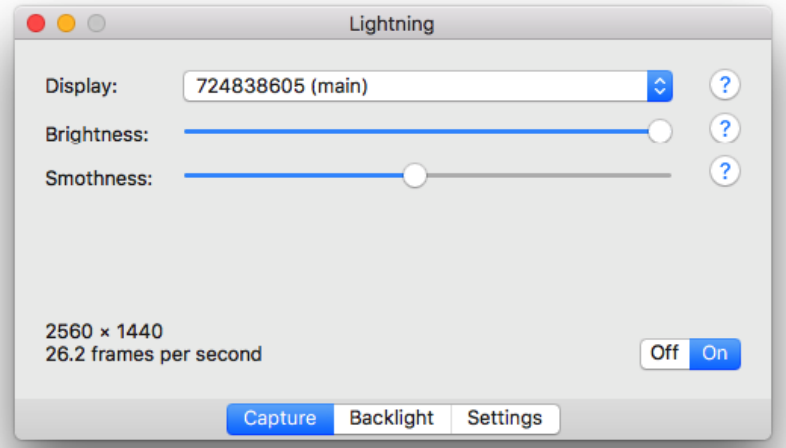
4.5 Implementacja

4.6 Podręcznik użytkownika

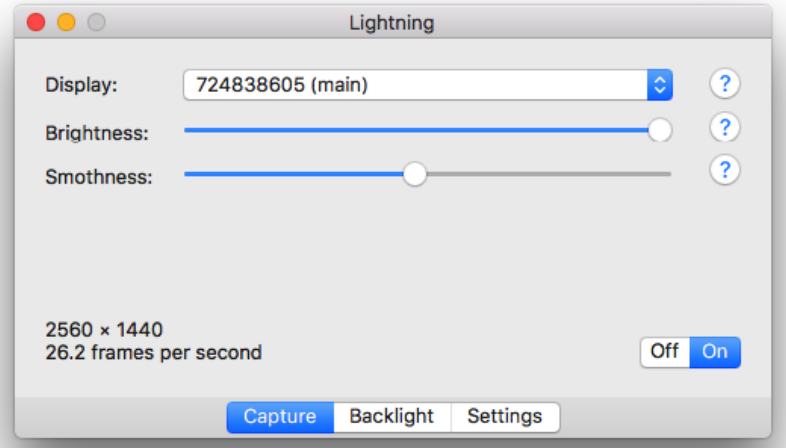
4.7 Przykładowa implementacja animacji

4.8 Analiza projektu

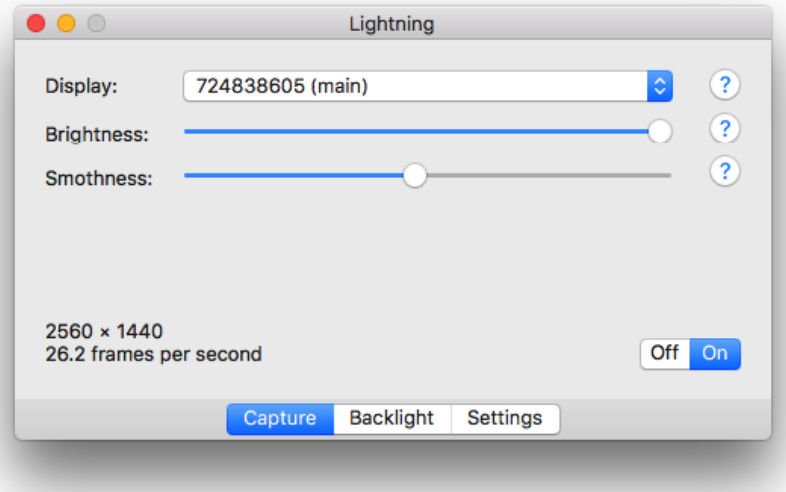
4.9 Perspektywy rozwoju projektu



Rysunek 4.4: Widok trybu przechwytywania



Rysunek 4.5: Widok trybu animacji



Rysunek 4.6: Widok konfiguracji

Rozdział 5

Podsumowanie

5.1 Dyskusja wyników

5.2 Perspektywy rozwoju pracy

Bibliografia

[1] http://www.wirtualnemedia.pl/artykul/coraz-dluzej-ogladamy-telewizje-najwiecej-czasu-przed-szklanym-ekranem-spedzaja--seniorzy-raport.
Data dostępu – 15.01.2017.

[2] 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.

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

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

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

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

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

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

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

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

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

Spis rysunków

Spis tablic

Spis listingów