라즈베리 파이를 통한 IOT 전등

최준영, 김요한, 박재영

(지도교수 : 이병국)

IOT light with Raspberry PI

Jun-Young Choi, Yo-Han Kim, Jae-Young Park

요약

초연결, 초지능, 초융합으로 대변되는 4차 산업혁명이 시작되면서 우리의 일상은 모든 면에서 서서히 하지만 아주 빠르게 변화하고 있다. 가정이나 직장에 배치된 기기를 쉽게 제어 가능하게 함으로써 보다 편리하고 유용한 효과를 볼 수 있다. 본 연구에서는 라즈베리파이와 여러 모듈을 연결하고 이를 제품으로 만들며 함께 애플리케이션을 개발하여 제품을 보다 빠르고 편리하게 조작하는 것을 목적을 두고 있다. 또한 기기와 스마트폰이 같은 네트워크에 연결되어있지 않더라도 외부에서 기기를 제어하도록 통신 환경을 구현하며 실시간 데이터 교환을 통해 애플리케이션에 스트리밍 서비스를 이용할 수 있도록 구성한다. 제품화를 위해 LED, 마이크 등 모듈들을 을 선정하고 사용자의 환경에 적합한 알고리즘을 개발하며 유연한 제품화와 동작을 설계한다.

Keyword: IoT, Rasberry Pi, WebRTC

I. 서 론

IoT 기술의 발전, 서비스형 인공지능(AIaaS)의 보편화로 인해 우리는 일상 속에서 많은 변화를 불러왔다.

자율주행 자동차, Chat gpt가 대표적인 사례이다. 특히나 Chat gpt와 같은 AIaaS의 등장은 기술적 장벽을 허무는 큰 역할을 하고 있다.

이러한 변화 속에서 우리는 기술적 장벽이 허물어지고 있다는 점에 주목하였다. 기술이 전문가만이 아닌 일반 사용자들이 쉽게 사용 가능하다면 꼭 완성된 제품이 아니더라도 이용자가 직접 설계하여 제품을 완성시킬 수 있게 되기 때문이다.

한 예시로 사용자가 제품 설계 과정을 직접 커스텀하여 제작하는 프로젝트가 있다. 바로 오픈 소스 하드웨어를 기반으로 개발하는 것을 목적으로 한 모듈형 스마트폰 프로젝트인 구글의 'ARA 프로젝트'이다.

당시 ARA폰은 사용자가 모듈화 된 스마트폰의 부품을 원하는 방향으로 선택하고 조립하여 쓸 수 있는 아키텍쳐 스마트폰이였다.

하지만 모종의 이유로 프로젝트가 무산되었고 ARA폰은 빛을 보지도 못 하고 그렇게 사라졌다.

당시 프로젝트 무산에 대해 많은 추측이 있었지만 하드웨어의 안정성 문제가 해결되지 못해 출시되지 못했다는 의견이 지배적이였다.

하지만 소프트웨어의 경우 소프트웨어의 안정성도 분명 중요하지만 하드웨어에 비하여

리스크가 크지 않다. 또한 어떠한 소프트웨어를 추가하느냐에 따라 개발환경과 제품에 내장되어 있는 보드의 성능이 중요하겠지만 일반적인 소프트웨어를 개발하고 적용 것에는 현재 이 시점 시중에 풀려있는 보드들의 성능으로도 충분하다.

II. 아두이노 기반 프로토타입

2-1 회로도 설계

IOT환경을 구축하기 위해 디바이스를 선정하기위해 먼저 아두이노를 선정하게 되었다. Neopixel LED를 제어하기 위해서 이와 호환되는 Arduino Uno R3보드를 사용하였고 LED네오픽셀은 Adafruit사의 Neopixel(10픽셀)을 사용하였다.

마이크 기능 구현을 위해 DM445 소리 감지 센서를 이용하였으며 무선 네트워크 연결하여 웹서버를 생성하기 위하여 ESP-01(wifi module)을 사용하였다.

와이파이 모듈을 사용하기 위해 본래 전압차(5V)로 인해 저항을 사용해야 하지만 회로도의 단순화를 위해서 ESP-1 어댑터를 추가 사용하여 회로도를 구성하였다.

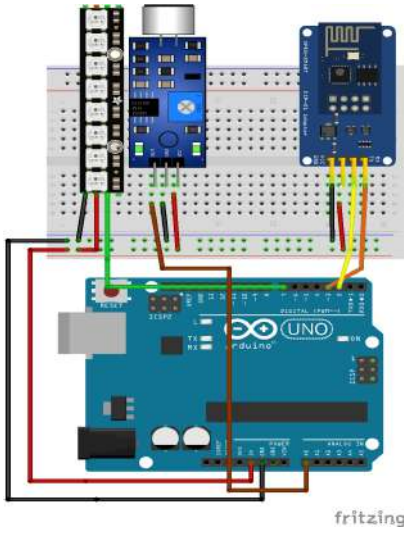


그림1 아두이노 회로도

그림1과 같이 모듈들은 5v에서 작동하며 Esp-01모듈은 시리얼통신을 위해 Rx, Tx 포트와 연결하고 dm445모듈은 아두이노에서 오디오 값을 아날로그 값으로 받아오기에 아날로그 A0에 연결해준다. Neopixel의 경우 하나의 디지털 단자로 모든 LED를 조작하기에 7번 디지털 핀에 연결해 주었다.

2-2 네트워크 연결

Esp-01 모듈을 사용하여 웹 서버 생성을 위하여 아두이노에서 제공 되는 Wifiesp 라이브러리를 사용하였다.

AT명령어와 시리얼 통신 방식으로 구현하였으며 미리 설정된 네트워크의 이름과 암호를 통해 해당 네트워크 에 접속 후 웹 서버가 생성된다.

```
void setup() {
  strip.begin();
  strip.setBrightness(200);
  r = 0;
  g = 0;
  b = 0;
  strip.show();
  Serial.begin(9600);               //통신속도
  esp01.begin(9600);                //와이파이 모듈 통신 속도(기본:115200=>9600)
  WiFi.init(&esp01);
  while (status != WL_CONNECTED) {  //wifi 연결 시도
    Serial.print(F("Attempting to connect to WPA SSID: "));
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);
  }
  server.begin();//서버 시작
}
```

그림2 아두이노 웹 서버

2-3 결과

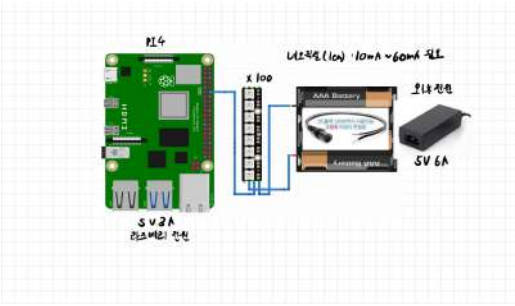
LED를 조작하기 위해 아두이노를 선정하였지만 실험 중 해당 코드에서 파라미터에 접근할 수 없으며 지정한 함수만을 실행 가능하였고 웹 서버를 갱신할 때마다 해당 페이지의 지속적인 지연이 반복되며 애플리케이션의 데이터교환이 제한되어 개발에 어려움

이 발생하였다. 이에 제품과 아두이노의 용도, 통신/네트워크, mic모듈의 부정확성의 이유로 해당 디바이스를 라즈베리 파이로 바꾸어 진행하였다.

III 라즈베리 파이 설계

3-1 라즈베리 파이 회로 설계

본 연구에서는 라즈베리 파이 4 Model B를, OS는 Raspbian OS (Linux기반)를 사용하였다. LED는 Adafruit사의 100픽셀 약 5M 길이의 Neopixel을 사용하였으며 마이크 기능과 스피커 기능 추가를 위해 USB 2.0 마이크와 스피커를 사용하였다. 전원 구성은 라즈베리 파이 보드의 권고 파워(5v 3a), Neopixel 한 픽셀을 기준으로 최대 밝기로 백색을 사용할 경우 최대 60ma가 필요하며 100개를 사용할 경우를 고려하였다. 마이크와 스피커를 추가 하였으며 또한 이후 추가적인 모듈의 사용이 있을 수도 있음을 고려하였고 결론적으로 라즈베리 파이 보드 및 LED 전력 사용량과 안정성을 고려하여 외부전력 사용을 위한 회로도 설계를 진행하였으며 외부전원은 전압 5V 전류 6A DC어댑터를 사용하였다.



. 그림3 라즈베리 회로도

3-2 모듈 구성 및 설정

네오픽셀은 5v, Gnd, D 단자로 이루어져 있다. 전원 연결 후 네오픽셀은 하나의 GPIO와 디지털 단자를 통해서 제어하며 이를 위해 10x10 픽셀 구성으로 납땜을 진행한다. 테스트를 위해 진행할 때 사용한 GPIO18은 PWM 프로토콜을 사용하며 이는 USB 마이크, 스피커의 통신과 중복이 된다. 이 때문에 네오픽셀의 제어가 불가능하며 네오픽셀의 통신방식을 GPIO10의 SPI 프로토콜로 변경하며 버퍼, 주파수를 재설정한다. 이후 마이크, 스피커 설정을 위해 alsa.conf의 pcm.hw를 마이크의 card번호와 일치함으로서 Default 값으로 설정하여 활성화를 시켜준다

IV 제품 제작

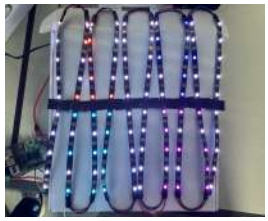


그림4 LED배치

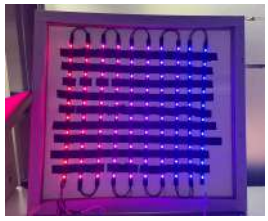


그림5 액자 제작

LED를 조명, 액자로 활용하기 위해 액자형으로 설계하였으며 Neopixel의 경우 하나의 디지털 단자로 조작하기에 그림과 같은 전선 구조를 선정하였다. 전원선과 라즈베리파이의 배치, 전선의 위치를 고려하여 제품 몸체는 목재를 사용하여 해당 제품을 제작하였다.

그림6 제품 제작

V 기능 설계

LED를 조작하기 위해 Adafruit에서 제공되는 라이브러리인 rpi_ws281x를 사용하며 이는 Neopoxel에 존재하는 ws2812b 칩을 제어할 수 있다. 하나의 GPIO로 모든 LED제어함에 있어 strip을 지정하여 해당 strip의 개수, HZ, DMA, INVERT를 설정 후 알고리즘에 적용시킨다.

5-1 조명 효과

조명 기능을 수행하기 위해 On/Off, Rainbow기능을 구현하였다. RGB값으로 색을 결정하며 Strip의 전체를 밝기와 함께 설정하여 On으로 설정하고 off 시 이를 모두 0으로 초기화 시킴으로서 해당 LED를 모두 종료시킨다. 또한 퍼포먼스적인 기능을 추가하기위해 해당 각 열에 10 픽셀의 그룹을 나누어 점차 색을 Wheel를 통해 계산 후 바뀌는 Rainbow 기능을 추가하였다.

5-2 오디오 스펙트럼 알고리즘

마이크를 통해 반응형 LED를 설계한다. 오디오 스펙트럼을 구현하기위해 소리의 요소 중 주파수, 데시벨을 필요로 한다. 이를 위해 파이썬 PyAudio 라이브러리를 사용하며 Mic로 설정한 INPUT_ FRAMES_PER_ BLOCK을 추출하여 이를 0에서 100단위로 임의로 조정하여 알고리즘에 적용할 수 있도록 설정한다. 이후 실행시 쓰레드를 통해 독립적으로 실행되며 오디오의 주파수와 데시벨을 계속 출력한다.

테스트 과정시 데시벨과 주파수 영역을 나누어 행(주파수), 열(데시벨)로 설정하여 구현을 시도하였으며 이 때 데시벨의 변화가 원할하지 못하였으며 오디오에 민감하게 반응하는 주파수만으로 알고리즘을 구현하였다.

오디오 스펙트럼의 Wave를 표현하기 위해 지속적으로 갱신되는 주파수 값을 실시간으로 LED로 표현하기 위해서 각 열마다의 10개의 픽셀을 그룹으로 설정하여 10개의 temp 리스트(스택)로 지정하였다. 이후 주파수를 frq변수로 설정하여 10개의 LED중 frq만큼의 크기만큼 1의 값을 append하게 설정하였다. 그리고 리스트에 저장된 1의 값의 수만큼 len값을 설정해 LED가 활성화되며 이때 가장 위쪽의 LED의 색을 따로 지정하여 가장 높은 소리의 위치를 보여준다. 이후 새로운 주파수 값이 temp[9]에 들어오기 전 왼쪽의 리스트로 그대로 값을 전달하며 이를 반복하여 LED의 스펙트럼을 표현한다. 이 때 리스트를 복사할 경우 LED의 구조상 가장 하단의 위치를 재설정할 필요가 있기에 하단 값을 계산하는 알고리즘을 추가하였고 함께 카피되기 전 서로의 len값의 비교하여 이전 값 보다 크기가 작을 경우 남은 LED가 꺼지는 기능을 구현하여 보다 자연스러운 입력, 복사, 초기화의 반복이 시행되도록 알고리즘을 구현하였다.

5-3 시계 알고리즘

시계를 표현하기 위해 시간, 분의 영역을 총 4개의 숫자로 나누어야하기에 10x10의 픽셀을 5x5 총 4구역을 나누어야한다. 이에 앞서 0에서 9까지의 숫자를 표현하기위한 numbers[]에 10개의 숫자를 5x5리스트에 0과 1로 설정하였고 LED를 설정하기 위한 temps(5x

5)를 만들고 이를 각 영역에 대입시킨다.

time을 통해 현재의 시간을 받아오며 n1부터 n4까지 시간과 분을 저장한다. 이후 각 4영역의 LED위치에 자동으로 대응시킬 수 있는 함수를 시행하며 temps에 순서대로 복사하며 각 시간과 분을 표현한다. 함께 무드등과 같이 표현하기위해 global 변수 brightness와 tos를 지정하여 밝기가 점차 켜지고 꺼지는 점등 효과를 표현하였다.

5-4 디지털 액자

제작한 10x10 픽셀을 통해 단순 전등의 역할 이 아닌 전시용으로서 활용할 수 있도록 디지털 액자로 사용하기 위한 알고리즘을 추가하였다. 먼저 Pillow 라이브러리를 통해 그림파일을 다루었다. 사용자가 원하는 그림파일을 업로드하여 이에 저장된 디렉터리에 접근하여 px로 불러와 new에 10x10크기로 저장한다.

이후 해당 위치의 픽셀의 rgb값을 10x10으로 저장한 후 각 LED위치를 계산하는 알고리즘을 통해 지정된 위치에 각 픽셀이 대응시킴으로써 해당 그림이 액자에 표현된다.

Ⅵ 통신/네트워크

연구에 앞서 Headless 라즈베리파이의 제어를 위해 React Native을 활용하여 Android 및 iOS 플랫폼에서 동작하는 모바일 애플리케이션을 개발하기로 결정하였다. 이를 통해 사용자는 애플리케이션을 통해 간편하게 Headless 환경의 라즈베리파이를 제어하고 모니터링할 수 있게 되어, 효율적이고 편리한 IoT 환경을 조성할 수 있을 것으로 기대된다.

6-1 통신 클라이언트 구축

Web Socket를 통한 SDP 및 ICE Candidate 교환과 P2P 연결을 위해 라즈베리파이는 python으로 개발되었으며, 모바일 애플리케이션은 Node.js의 React Native를 통해 개발이 되었다.

라즈베리파이는 python-socketio 라이브러리를 활용하여 websocket 클라이언트를 구축하였고, aiortc 라이브러리를 활용하여 WebRTC를 활용한 P2P연결을 지원하도록 개발 했으며, 모바일 애플리케이션은 Node.js의 React Native를 사용하며 socket.io-client 라이브러리를 활용하여 websocket 클라이언트를 구축하였고, react-native-webrtc 라이브러리를 활용하여 WebRTC를 활용해 P2P연결을 지원하도록 개발을 하였다.

6-2 WebRTC

모바일 애플리케이션 및 라즈베리파이에서 인자전달, 음성통화, 영상통화 등 여러 가지 형태로 일어날 수 있는 실시간 통신을 위해 WebRTC을 활용하기로 하였다. WebRTC에서 P2P(Peer-to-Peer) 연결을 설정하기 이전에는 네트워크의 복잡성과 보안 문제로 직접적인 피어간 연결설정이 어려웠지만 피어들간의 정보교환과 안전한 연결설정을 위해 중간에 시그널링 서버(Signaling Server)를 도입했다.

6-3 시그널링(Signaling) 서버

시그널링 서버는 Node.js을 사용했으며 So

cket.io를 활용하여 각 피어의 SDP(Session Description Protocol)를 교환하고 네트워크 환경 관리, NAT 관리, 보안 및 인증, 피어의 상태 및 가용성 관리 등을 담당하여 안정적이고 안전한 P2P 통신을 이루어 내었다.

6-4 통신 과정

1.room접속

라즈베리파이는 UUID(universally unique identifier)를 생성하고 이를 활용하여 Room을 생성한다. 모바일 어플리케이션 사용자는 블루투스 통신을 통해 라즈베리파이의 UUID를 알아내고 이를 활용하여 Socket.io서버에 생성된 라즈베리파이의 Room에 접근한다.

2.Offer와 Answer 교환

라즈베리파이측에서 Offer를 생성하고, React Native는 Answer를 생성한다. 이는 각 피어가 자신의 로컬 환경과 통신 설정에 대한 정보를 SDP 형식으로 제공한다.

3. ICE Candidate 교환

ICE Candidate는 P2P 네트워크에서 피어 간 연결을 설정하기 위한 프레임워크는 ICE (Interactive Connectivity Establishment)에서 사용되는 개념이며, 각 피어는 자신의 로컬 환경에서 ICE Candidate 정보를 생성 및 교환한다.

4. 피어 연결 설정

SDP 및 ICE Candidate 정보를 교환한 후, 각 피어는 이 정보를 사용하여 P2P 연결을 설정하고 두 피어는 서로의 정보를 활용하여 안정적이고 최적의 통신 경로를 찾아서 실시간 통신을 수행한다.

6-5 React Native의 WebRTC 데이터 채널에서 제어 알고리즘

```
const ReqDataChanel = (data) => {
  channel.send({
    id : Client.id,
    state : true,
    message : { 'handle' : data }
  })
}

// ...

<TouchableOpacity onPress={() => { ReqDataChanel('audio') }}>
  <Text>audio</Text>
</TouchableOpacity>
```

그림7 WebRTC 제어 코드

React Native에서 javascript로 작성된 P2P 연결 후 데이터 채널에서 메시지를 보내는 알고리즘이며 WebRTC 데이터 채널을 통해 라즈베리파이의 동작을 제어할 수 있는 인자를 전송할 수 있다.

6-6 라즈베리파이 GATT(Generic Attribute Profile)서버 구축

```
### Go 코드 ###
  scriptPath := "./script_name.sh"
  arg1 := SSID
  arg2 := PASSWORD

  cmd := exec.Command("sh", scriptPath, arg1, arg2)
```

그림8 GATT 서버

GATT는 BLE(Bluetooth Low Energy) 연결을 통해 프로필 및 데이터를 주고받기 위한 방법을 정의한다. 초기 라즈베리파이의 인터넷 설정을 위해 라즈베리파이에서 Go 언어와 빠른 구축을 위해 paypal/gatt 라이브러리를 사용하여 GATT 서버를 구축하였으며 GATT서버 에서 쉘 스크립트를 실행하여 Wi-Fi 연결 설정을 변경한다

VII 애플리케이션

7-1 애플케이션 UI/UX

앱의 네비게이션은 React Navigation 라이브러리를 적극적으로 활용하여 설계되었다. NavigationContainer로 애플리케이션의 네비게이션을 감싸며, StackNavigator를 사용하여 스택 네비게이터와 각 페이지를 정의했으며, BottomTabNavigator를 활용하여 탭 네비게이터를 생성하여 페이지 간의 간편한 이동 및 기능 사용을 지원하였다.

이 구조를 통해 사용자는 각 페이지로의 이동과 필요한 기능을 효과적으로 활용할 수 있을 것으로 기대된다.

그림9 연결

그림10 앱 제어

1. 메인 페이지 (Home)

기기 목록 표시. 기기 선택 시 팝업 애니메이션 효과 팝업 페이지에서 버튼을 통한 라즈베리파이 제어

2. 블루투스 검색 페이지 (Search)

블루투스 검색을 통한 기기 추가 기능. 라즈베리파이의 와이파이 설정 가능.

3. 스토리지 페이지 (Storage)

캡쳐한 이미지, 영상, 음성 확인 가능. 최신순으로 스크롤 가능한 디자인.

4. 설정 페이지 (Setting)

기기 정보 표시 (브랜드, OS, 기기 이름, 버전 등). 약관 및 개인정보 처리방침으로 이동하는 버튼 제공.

5. 상단 및 하단 부분

모든 페이지에 로고와 블루투스 버튼 배치. 하단에 퀵 메뉴 (메인, 블루투스, 스토리지, 설정) 제공.

6. 애니메이션 효과 팝업 애니메이션

밑에서 위로 올라오는 효과. TouchableOpacity를 사용한 버튼은 터치 시 살짝 연해지는 효과가 있다.

VIII. 시뮬레이션 및 결과

8-1 애플리케이션

블루투스를 통해 제품과 연결 후 기기에 대한 접속이 가능하다. 이후 메인 페이지를 통해 LED를 조작할 수 있으며 사진, 비디오를 전달 가능하다

8-2 알고리즘 테스트

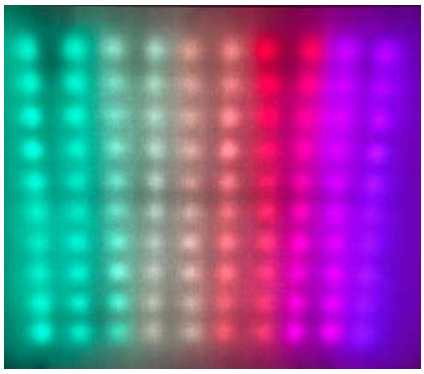


그림11 Rainbow

LED의 On/Off와 Rainbow효과를 테스트하였으며 다른 알고리즘 실행 시 충돌해 오류가 나는 경우가 발생하였으며 Rainbow를 오래 실행 시 색의 변화를 관

찰하였다.

*https://youtu.be/l6rCPJJNrhA

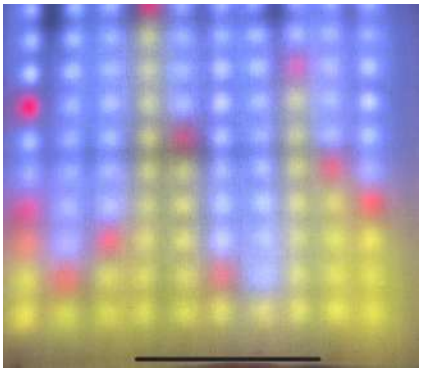


그림12 Audio Spectrum

연결된 마이크에 소리가 전달되면 해당 주파수의 범위만큼 LED가 켜지며 그 값을 받아오는 속도를 조절하여 흐름의 세기를 조정가능하다. 다만 함수의 속도가 너무 빠르게 설정될 경우 리스트 연산 과정에서 Overflow가 발생 되었으며 값을 다시 초기화함으로서 이를 해결하였다.

*https://youtu.be/w7xrlTHNYCQ

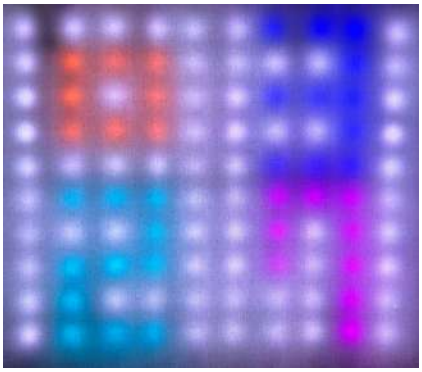


그림13 Clock

실시간 시간을 갱신하도록 설정하였으며 대기 모드에 적용시키는 만큼 전류 소모를 감소시키기 위해 밝기를 255에서 점차 줄이고 늘어나기를 반복하는 기능을 추가하였다.

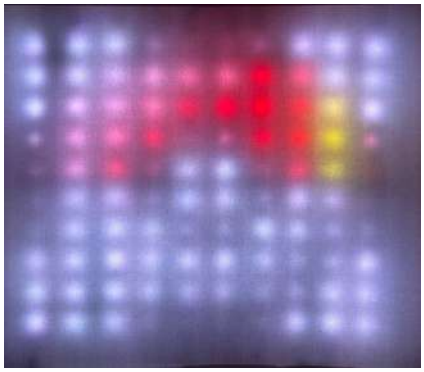


그림14 Picture

이미지의 처리 속도, RGB 값의 정확성을 관찰하였으며 검은색일 경우 LED의 색상이 (0,0,0)으로 설정되기에 결과물에서 관찰되기 어려웠다.

IX. 결 론

본 논문에서는 라즈베리파이를 통해 제품을 제작하고 이를 애플리케이션으로 제어해봄으로써 사물이 네트워크 환경에서 데이터를 교환하는 과정을 연구로 진행하였다. WebRTC와 시그널링 서버를 사용함으로서 사진, 오디오, 스트리밍을 전달가능하며 이를 사물에 적용해 사용자에게 필요한 서비스를 제공해 줄 수 있다. 또한 같은 네트워크 상에 접속돼있지 않더라도 인터넷이 가능한 환경이라면 언제든 애플리케이션을 통해 기기를 제어할 수 있도록 구현하였다.

제품 제작부터 통신, 애플리케이션 까지 직접 구현해 봄으로써 많은 지연, 오류 등의 어려움이 발생하였다. 라즈베리에서 작동하는 코드임에도 통신과정에서 주파수 값을 받아오는데 약 10초 이상이 걸린 사례도 있었으며, 애플리케이션으로 통해 제어할 경우 기존 코드와 겹쳐 두 LED모드가 동시에 실행되는 문제도 발생하였다.

연구를 진행하면서 이러한 오류들에 대응하기 위해 재설계를 반복하였으며 실제 사용자에 적합하고 필요한 서비스를 개발하였고 빠른 통신을 위해 P2P 통신 방식을 선정하고 확장성을 염두 하였다.

실제 제품 제작 시 LED의 확장이 가능하기에 얼마든지 확장이 가능하며 전류 용량과 LED(Neopixel)의 개수만을 고려하기에 비용적 측면에서도 절약 할 수 있다.

데이터 교환 측면에서도 데이터 처리가 불필요하며 이를 저장할 별도의 데이터 베이스가 없기에 경제성에도 강점으로 가지며 사용자에게는 더 빠른 성능을 제공할 수 있다.

References:

[1] 김정원. "라즈베리파이를 이용한 스마트 홈 프로토타입 구현" 한국전자통신학회 논문지 10, no.10 (2015) : 1139-1144.

[2] Jeremy, Garff . “Garff / Rpi_ws281x.” github, accessed October 10, 2023. https://github.com/jgarff/rpi_ws281x.

[3] The Pi Hut . “Using-Neopixels-with-the-Raspberry-Pi.” thepihut, last modified Nov 16, 2018, accessed November 3, 2023. https://thepihut.com/blogs/raspberry-pi-tutorials/using-neopixels-with-the-raspberry-pi.

[4] 최효현. “WebRTC를 이용한 화상회의 서비스 구현 Development of Video Conference Service Using WebRTC,” (2016) :