학습 결과보고서

프로그램명	IoT Tech. Upskilling 현장문제 중심 IoT-X 융합 메이커톤 프로그램				
학과	디스플레이반도체공학과	학번	202012471	학년	3학년
이름	정지우	연락처	010-8921-1379	이메일	jju0531@naver.com

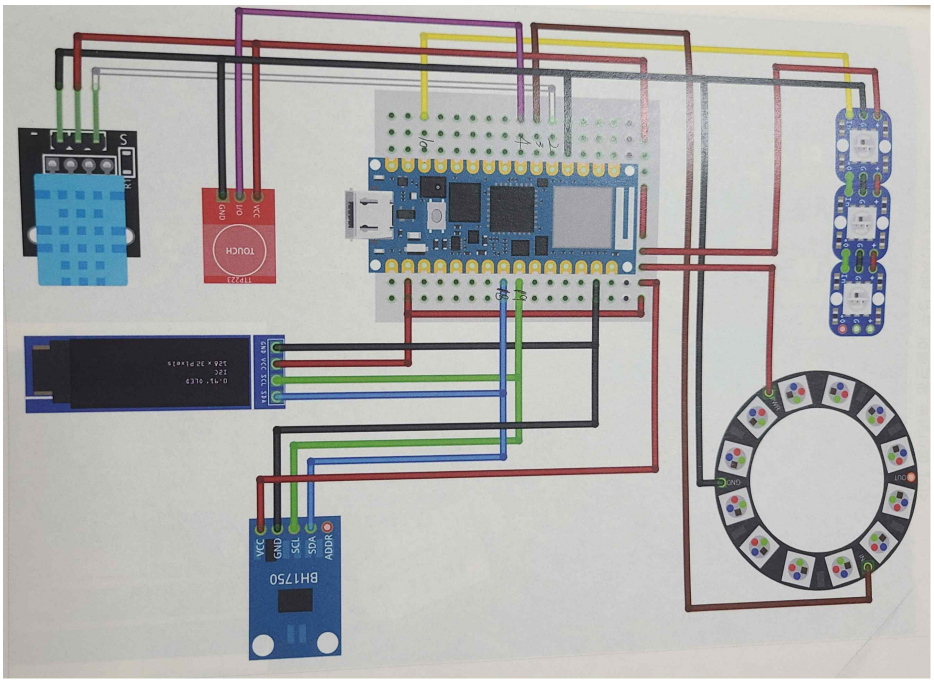
학습내용 및 과제결과

목차

- 아두이노
- 3D 모델링

[아두이노]

우선 회로는 다음 사진과 동일하게 구성하였습니다.

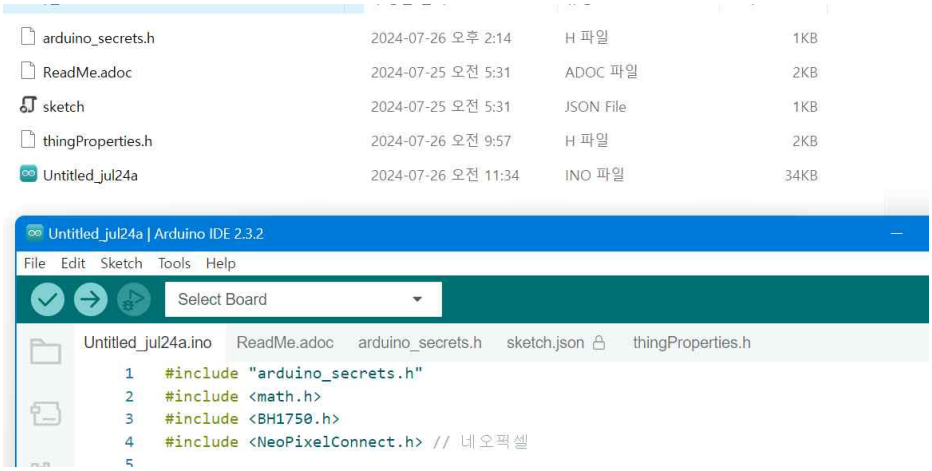


우선 Arduino Cloud에 들어가서 Device 등록후 Things를 통해 등록

Cloud Variables

	Name ↓	Last Value
☐	**fHumidity** `CloudTemperatureSensor fHumidity;`	62
☐	**fLightBrightness** `CloudTemperatureSensor fLightBrightness;`	244.167
☐	**fTemperature** `CloudTemperatureSensor fTemperature;`	26.7
☐	**Mode** `int mode;`	4
☐	**myCloudColoredLight** `CloudColoredLight myCloudColoredLight;`	{"bri":"0","hue":"0","sat":"0","swi":"false"}

이후 코딩은 zip파일을 다운받아서 오프라인 환경에서 진행하였습니다.



첫 번째,

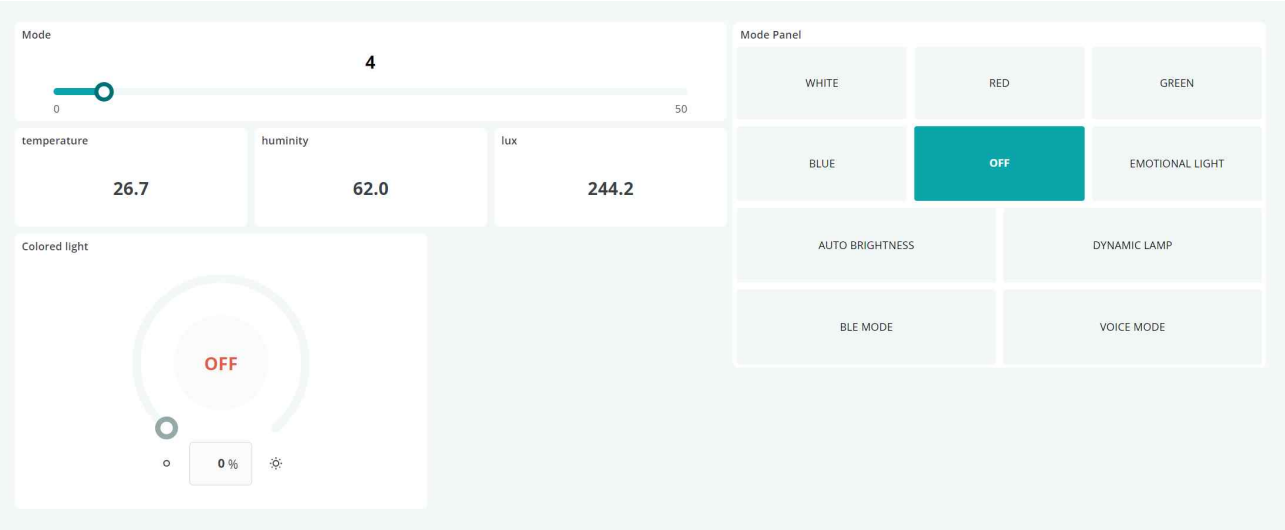
```
// 온도습도센서
#include <DHT.h>
#include <arduino-timer.h>

// 조도센서
#include <Wire.h>
 #include <NeoPixelConnect.h> // 네오픽셀
```

위의 라이브러리를 통해 온도,습도,조도를 측정될 수 있도록 구현하였고, 이에 따라 제공된 LED NeoPixel을 제어를 구현하였습니다.

```
void myDHT_SensorDAQ(void) {
  if(flgtemper==1)
  {
    fHumidity = myDHT.readHumidity();
  fTemperature = myDHT.readTemperature();
  unsigned char getTempData;


  #if(DEBUG_MODE_ENABLE)
    Serial.print("Temp.: "); Serial.print(fTemperature); Serial.print("[°C]");
    Serial.print("   Humidity: "); Serial.print(fHumidity); Serial.println("[%]");
  #endif
```

```
void myDxAmbientLightSensorGetData(void) {
  if(flglight==1)
  {
    if ((unsigned long)(millis() - seningPreviousMillis) >= sensingInterval) {
    seningPreviousMillis = millis();
    fLightBrightness = lightSensor.readLightLevel();
```

그리고 각 아두이노 클라우드에 등록해둔 Things의 변수로 보내어서 실시간으로 모니터링이 가능하도록 구현하였습니다.



그리고 또한 5번 모드에 온도습도에 따른 불쾌지수를 계산해서 불쾌지수에따른 LED를 제어하였고, 6번 모드에서 조도에 따라 조도가 너무 밝으면 LED밝기를 줄이고 반대의 경우에는 밝기를 상승시키는 모드를 구현하였습니다.

또한 delay를 사용하지 않고 mills를 사용하여 멈추지 않고 다른 작업을 가능하게 하였습니다.

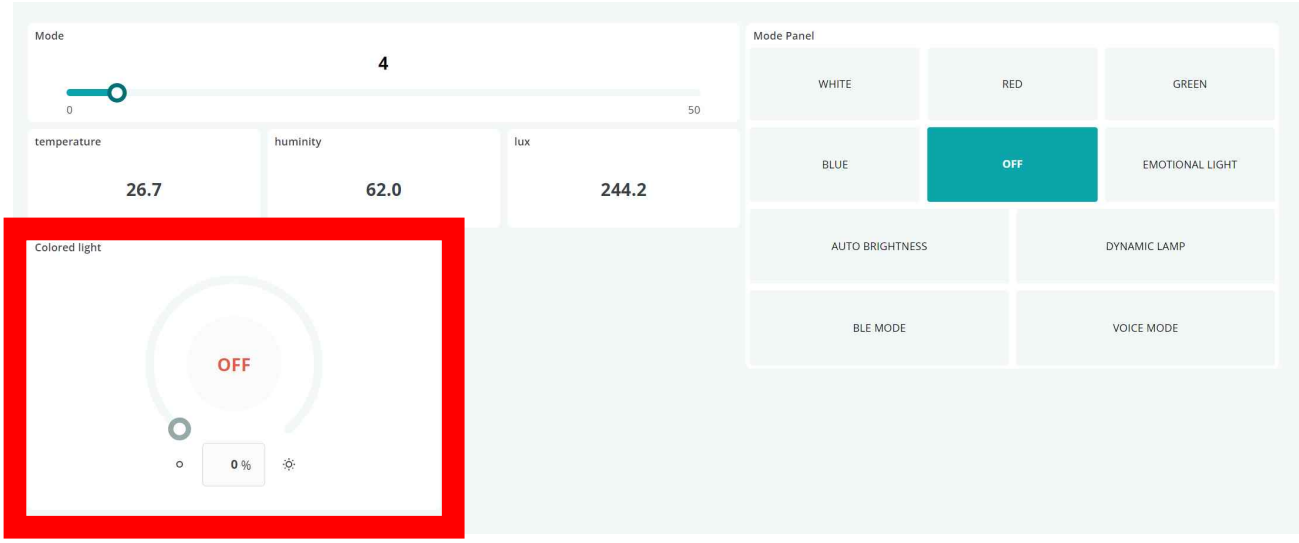
[모드5번]

```
Discomfortindex = 0.72*(temp+hum)+4.6;

if (Discomfortindex > 80) {
  // fTemperature가 32보다 큰 경우
  p_circle.neoPixelFill(154,231,197, true);
  delay(10);
  sendHSV_ColorDataToCloud(154,231,197);

}

else if (Discomfortindex >= 75 && Discomfortindex <= 80) {
  // fTemperature가 31~26 사이인 경우
  p_circle.neoPixelFill(117,213,106, true);
  delay(10);
  sendHSV_ColorDataToCloud(117,213,106);

} else if (Discomfortindex >= 68 && Discomfortindex < 75) {
  // fTemperature가 26~18 사이인 경우
  p_circle.neoPixelFill(255,163,44, true);
  delay(10);
  sendHSV_ColorDataToCloud(255,163,44);

} else {
  // fTemperature가 18~0 사이인 경우
  p_circle.neoPixelFill(233,0,3, true);
  delay(10);
  sendHSV_ColorDataToCloud(233,0,3);

}
```

[모드6번]

```
lux = lightSensor.readLightLevel();

if (lux > 450) {
  p_circle.neoPixelFill(50, 50, 50, true);
  delay(10);
  sendHSV_ColorDataToCloud(50, 50, 50);

} else if (lux >= 400 && lux <= 450) {
  p_circle.neoPixelFill(100, 100, 100, true);
  delay(10);
  sendHSV_ColorDataToCloud(100, 100, 100);

} else if (lux >= 300 && lux < 400) {
  p_circle.neoPixelFill(150, 150, 150, true);
  delay(10);
  sendHSV_ColorDataToCloud(150, 150, 150);

} else if (lux >= 100 && lux < 300) {
  p_circle.neoPixelFill(200, 200, 200, true);
  delay(10);
  sendHSV_ColorDataToCloud(200, 200, 200);

} else {
  p_circle.neoPixelFill(255, 255, 255, true);
  delay(10);
  sendHSV_ColorDataToCloud(255, 255, 255);

}
```

두 번째로, LED가 모드에 상관없이 빛이 바뀔때마다 이를 클라우드에 업데이트 하고 반대로 클라우드위젯에서 조절할때마다 LED가 바뀌도록 구현하였습니다.



여기서 주의해야 할점은 LED를 제어하기위해 정보를 보낼때는 RGB값으로 보내면되지만, 반대로 LED로부터 정보를 값을 얻어서 클라우드 위젯에 업데이트를 하기 위해서는 HSV색좌표계로 변환해서 값을 보내야한다는 점이다.

[값을 보낼 때]

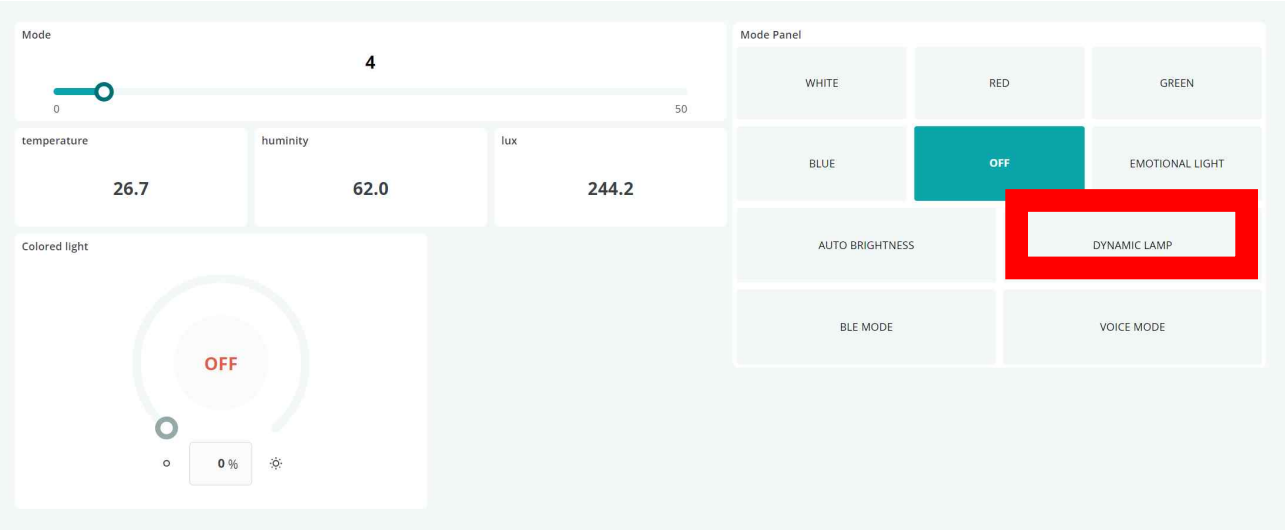
```
void onMyCloudColoredLightChange()  {
  // Add your code here to act upon MyCloudColoredLight change

  uint8_t r, g, b;
  if(myCloudColoredLight.getSwitch())
  {
    myCloudColoredLight.getValue().getRGB(r, g, b);
    p_circle.neoPixelFill(r, g, b, true);
    delay(10);
  }
  else
  {
    p_circle.neoPixelClear(true);
    delay(10);
  }
```

[값을 읽을 때]

```
void sendHSV_ColorDataToCloud(double r, double g, double b)
{
  rgb_to_hsv(r,g,b);

  myCloudColoredLight.setHue(hue);
  myCloudColoredLight.setSaturation(saturation);
  myCloudColoredLight.setBrightness(value);

  if (r == 0 && g == 0 && b == 0) {
  myCloudColoredLight.setSwitch(false);
} else {
  myCloudColoredLight.setSwitch(true);
}
}
```

세 번째로, 사전학습에서 미리 준비한 음성인식 기반의 ai 라이브러리를 활용하여 mode가 음성인식으로 선택되면, 음성인식을 하고 결과값에 따라 NeoPixel을 제어하는 조명장치를 구현하였습니다.

추가적으로 레코딩을 시작할 때, 작은 NeoPixel에 빨간불을 나타내게해서 녹음기능을 외부에서도 확인 할 수 있게 하였습니다.



[코드]

```
    ei_printf("Starting inferencing in 2 seconds...\n");

delay(2000);

p.neoPixelFill(255,0,0, true);
delay(10);

ei_printf("Recording...\n");

bool m = microphone_inference_record();
if (!m) {
    ei_printf("ERR: Failed to record audio...\n");
    return;
}

ei_printf("Recording done\n");
p.neoPixelFill(0,255,0, true);
delay(10);
p.neoPixelClear(true);
```

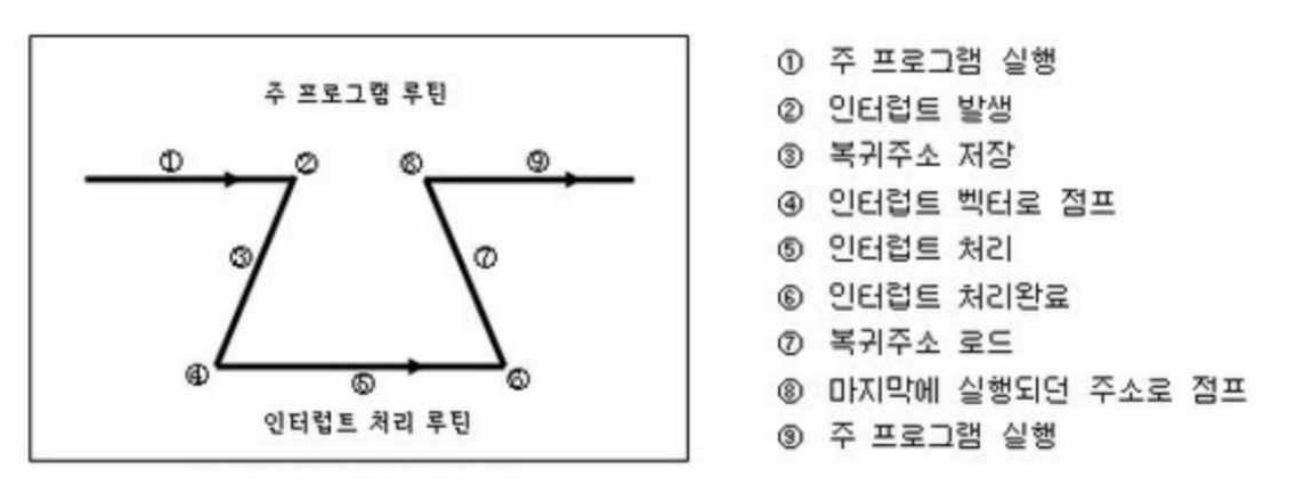
```
// print inference return code
ei_printf("run_classifier returned: %d\r\n", res);
print_inference_result(result);

if(result.classification[0].value>=0.5)
{
  p_circle.neoPixelFill(0,0,255, true);
  delay(10);
}
if(result.classification[1].value>=0.5)
{
  p_circle.neoPixelFill(0,255,0, true);
  delay(10);
}
if(result.classification[4].value>=0.5)
{
  p_circle.neoPixelFill(0,0,0, true);
  delay(10);
}
if(result.classification[5].value>=0.5)
{
  p_circle.neoPixelFill(255,255,255, true);
  delay(10);
}
if(result.classification[6].value>=0.5)
{
  p_circle.neoPixelFill(255,0,0, true);
  delay(10);
}

microphone_inference_end();
```

[주의사항]

음성인식을 시작할 때, 초기화를 하는 "microphone_inference_start()" 함수에서 malloc를 활용해서 메모리를 동적할당 해준다. 이때 끝나고 에러가 발생하지 않는 이상은 free를 통해 동적으로 설정한 메모리를 제거 해주지 않기 때문에 연속적으로 사용하면 메모리 누수가 발생할 수 있다.

그렇기 때문에 한번 음성인식을 하고나서 microphone_inference_end() 함수를 통해서 free 메모리를 해주어서 메모리 누수를 잡아야한다. 그렇지 않으면 아두이노는 메모리가 작기 때문에 재부팅 또는 망가질 우려가 있다.

네 번째로, 터치센서이다. 터치센서는 어떤 동작을 하더라도 인터럽트를 통해 인식이 되면 발생한 상황을 처리한 후 실행 중인 작업으로 복귀하도록 구성할 것이다.



우선 터치센서를 인식할 수 있는 핸들러함수를 main loop에 넣어서 인식이 되면 인터럽트를 실행하도록 구성한다.

```
void loop() {
  ArduinoCloud.update();
  // Your code here
  timer.tick(); // 온도 습도도
  myDxAmbientLightSensorGetData(); // 조도
  myBtnStatusCheckHandler(); //터치센서
  modedisplay();


}
```

[코드]

```
Serial.println("Touch Button pressed");
cntKeyInTimingCtrl++;
if(!flgOnOffstatus && (cntKeyInTimingCtrl>=KEY_IN_TIME_SHORT_CHECK_VALUE)){
  Serial.println("Mode 1:White Color");
  cntKeyInTimingCtrl=0;
  flgButtonDownStatus=true;
  flgOnOffstatus=true;
  seqSystemModeCtrl=5;
  digitalWrite(LED_BUILTIN,ON);
  p_circle.neoPixelFill(255,255,255,true);
  flgExternalInterruptDetected=false;
  sendHSV_ColorDataToCloud(255, 255, 255);


}
else if(flgOnOffstatus&& (cntKeyInTimingCtrl>=KEY_IN_TIME_LONG_CHECK_VALUE)){
  Serial.println("power off");
  cntKeyInTimingCtrl=0;
  flgButtonDownStatus=true;
  flgOnOffstatus=false;
  seqSystemModeCtrl=0;
  digitalWrite(LED_BUILTIN, OFF);
  p_circle.neoPixelFill(0,0,0,true);
  flgExternalInterruptDetected=false;
  sendHSV_ColorDataToCloud(0, 0, 0);


}
```

우선 첫 번째로 터치센서가 인식되면 2초 정도 인식되면 On을 의미하는 흰색을 NeoPixel로 표현하고, 5초 이상되면 Off를 의미하는 NeoPixel을 끄는 방식으로 구현하였다.

그리고 토글변수를 설정해서
만일 On상태에서 짧게 누르면, 이후로는 색을 바꿔주게 하는 코드를 구성하였다.

```
if(flgOnOffstatus){
  switch(seqSystemModeCtrl){
    case 5:
    if(!flgModeChange)seqSystemModeCtrl=10;
    break;
    case 10:
    Serial.println("MODE 2:RED COLOR");
    p_circle.neoPixelFill(255,0,0,true);
    flgModeChange=true;
    seqSystemModeCtrl=15;
    sendHSV_ColorDataToCloud(255, 0, 0);

    break;

    case 15:
    if(!flgModeChange)seqSystemModeCtrl=20;
    break;
    case 20:
    Serial.println("MODE 3:GREEN COLOR");
    p_circle.neoPixelFill(0,255,0,true);
    flgModeChange=true;
    seqSystemModeCtrl=25;
    sendHSV_ColorDataToCloud(0, 255, 0);

    break;

    case 25:
    if(!flgModeChange)seqSystemModeCtrl =30;
    break;
    case 30:
     Serial.println("MODE 4:BLUE COLOR");
     p_circle.neoPixelFill(0,0,255,true);
     flgModeChange=true;
     seqSystemModeCtrl=35;
    sendHSV_ColorDataToCloud(0, 0, 255);

     break;
```

case를 5~55까지 10의 간격으로 구성하였으며, 55이후에는 다시 5로 돌아오게 설정하였다.

다섯 번째로, OLED 디스플레이 장치를 통해 시작 부팅시, 팀명인 'Digital Image Processing'을 나타내었고, 이후에는 온도, 습도, 조도, 현재 모드를 나타내도록 코드를 구성하였습니다.

[코드]

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
static const unsigned char PROGMEM image_data_LOGOarray[] = {
  // Add Splash Screen LOGO ===> https://javl.github.io/image2cpp/
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4c, 0x00, 0xc0, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x0c, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xcc, 0xfc, 0xdf, 0x7e, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x4c, 0xcc, 0xcc, 0x06, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x4c, 0xf8, 0xcc, 0x66, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xcc, 0xf8, 0xcf, 0x6e, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x8c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
  0x00, 0xcd, 0x8c, 0x38, 0x3e, 0x38, 0x09, 0x8d, 0x1c, 0x0c, 0x38, 0x70, 0xe6, 0x4c, 0x1e, 0x00,
  0x00, 0xce, 0xf6, 0x46, 0x4c, 0xcc, 0x0c, 0xce, 0x73, 0x32, 0x46, 0xc3, 0x06, 0x76, 0x66, 0x00,
  0x00, 0xcc, 0x66, 0x3e, 0x6c, 0xfc, 0x0c, 0xcc, 0x63, 0x60, 0xfe, 0xf1, 0xc6, 0x66, 0x66, 0x00,
  0x00, 0xcc, 0x66, 0x66, 0xc0, 0xc0, 0x0c, 0xcc, 0x63, 0x30, 0xc0, 0x18, 0x76, 0x66, 0x60, 0x00,
  0x00, 0xcc, 0x66, 0x7e, 0x7c, 0x7c, 0x0f, 0xcc, 0x3e, 0x3e, 0x7c, 0xfb, 0xe6, 0x66, 0x7e, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xc6, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x7c, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

위 이미지처럼 원하는 이미지를 16진수로 바꿔주고,

```
void setup()  {
  Serial.begin(9600);
  Wire.begin(); delay(100);
  if(!myOledDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  myOledDisplay.clearDisplay(); myOledDisplay.drawBitmap(0, 0, image_data_LOGOarray, 128, 32, 1);  myOledDisplay.display();
  delay(2000);
}
```

위 코드와 같이 작성하여 디스플레이 시킬 수 있다.

[부팅시 팀명 디스플레이]

[온도, 습도, 조도, 모드 디스플레이]

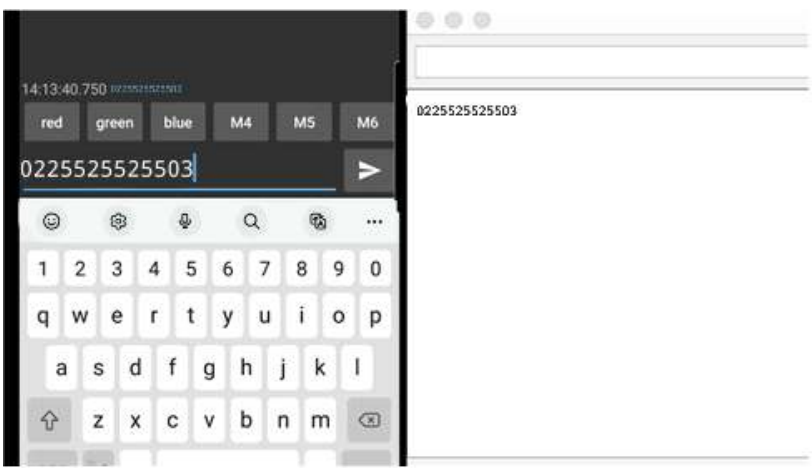
여섯 번째로, BLE 모드를 통해 블루트스 연결을 하고, 터미널 통신을 통해 13자리 숫자로된 정보를 전송하여 이를 아두이노에서 받아드리고 이에 따라 모드와 LED를 제어하는 코드를 구현하였습니다.

BLEService myService("0000ffe0-0000-1000-8000-00805f9b34fb");
BLECharacteristic myCharacteristic("0000ffe1-0000-1000-8000-00805f9b34fb", BLERead | BLEWrite | BLEWriteWithoutResponse | BLENotify, 32);

위 코드는 아두이노와 블루투스 모바일 기기간의 통신을 위해 필요한 통신 주소입니다.

```
void writtenEventCharacteristic(BLEDevice central, BLECharacteristic characteristic) {
  uint8_t value[32];
  int length;

  strMsgPacket ="";
  strMsgPacket.trim();

  length = myCharacteristic.valueLength();
  myCharacteristic.readValue((void *)value, length);

  if(length >= 13)  {
    for(int i=0; i<length; i++)  { strMsgPacket += (char)value[i];  }
    #if(DEBUG_MODE_ENABLE)
      Serial.print("Length: "); Serial.print(length);
      Serial.print("Data: "); Serial.println(strMsgPacket);
    #endif
```

다음으로 아두이노와 블루투스를 연결하고 통신을 통해 숫자로된 13자리의 정보를 받으면, 특정숫자에 따라 특정 행동을 취하도록 코드를 구성하여 제어하였다.



위 그림처럼 13자리 정보가 담긴 숫자를 전송하고 앞의 02는 모드를 의미하고, 뒤에 255, 255, 255는 RGB값을 나타내며 마지막 03은 마지막 코드로써 저는 저만의 정보를 인식하는 암호키처럼 사용하였습니다.

[주의사항]

이때, BLE모드와 와이파이를 통한 아두이노 클라우드 연동은 동시에 가능하지 않기 때문에, 토글변수를 통해서 Mode가 BLE로 설정됨과 동시에 와이파이 또는 아두이노 클라우드 변수들과 연동된 함수들은 다 off 시켜줄 필요가 있다. 또한 아두이노에 값을 계속해서 업데이트 하는 ArduinoCloud.update(); 함수는 와이파이를 연결을 해주는 함수가 있기 때문에 기존에 들어가 있는 함수라도 주의해서 사용하여야 한다.

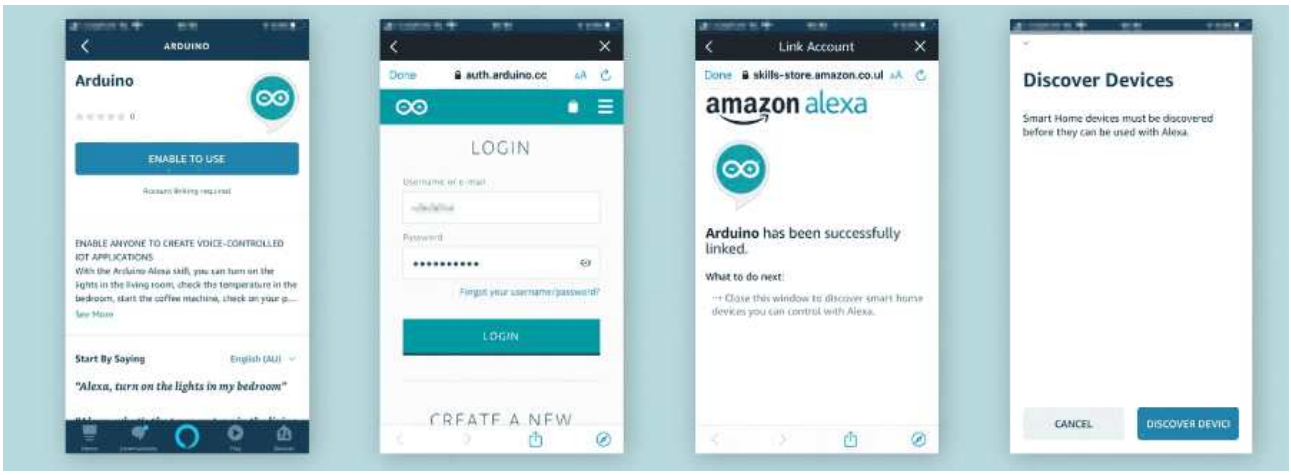
이유는 만일 BLE모드에서 특정함수에서 클라우드와 연동 되거나 와이파이와 관련된 함수를 작동시키면 두가지가 내부적으로 충돌해서 재부팅이 되기 때문이다.

[토글변수 off]

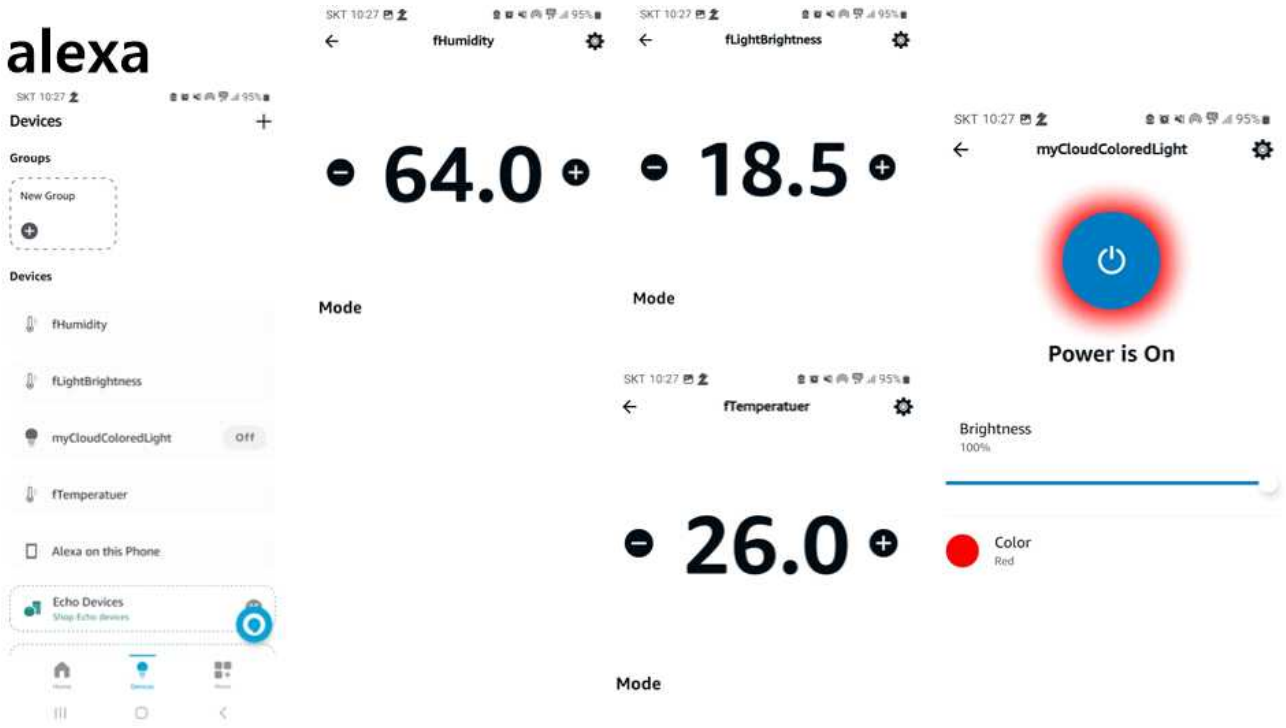
```
flglight=0;
flgtemper=0;
flgtouch=0;
```

일곱 번째로, Alexsa를 연동시켰다.

알렉사를 연동하기위해서는 간단하다. 우선 앱에서 Alexsa를 다운받고 실행시켜서 다음 그림과 같이 아두이노 클라우드의 계정과 동일한 구글계정으로 로그인하고



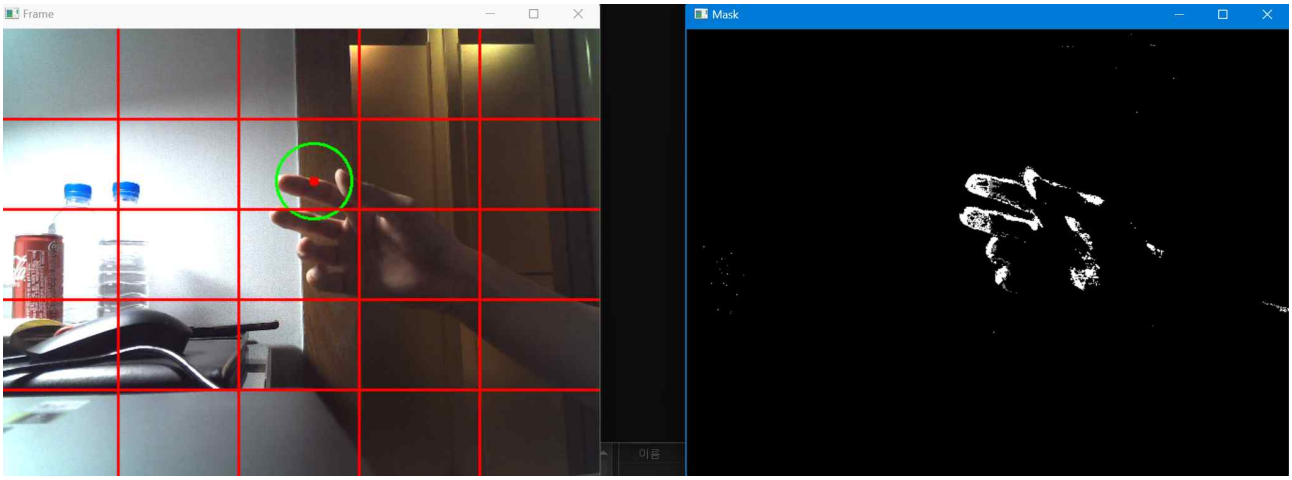
이후에 디바이스를 찾으면, 처음했던 Things에서 설정한 변수들 5개가 나온다. 이를 클릭하여 연동시켜주면 끝이다.



알렉사 화면을 축약해서 나타내었다.
좌측의 화면에서 변수를 클릭하여 연동하고 들어가서 각각의 센서값과 LED를 제어할 수 있게 된다.

마지막으로, 저희 조는 차별점으로 카메라를 통해 영상을 받아드리고 이를 영상처리 과정을 통해 정보를 추출해서 다시 IOT 사물인터넷로써 카메라와 아두이노간 공유기의 ip주소를 통해 웹서버를 만들어서 정보를 교환하여서 처리했습니다.

우선 첫 번째로



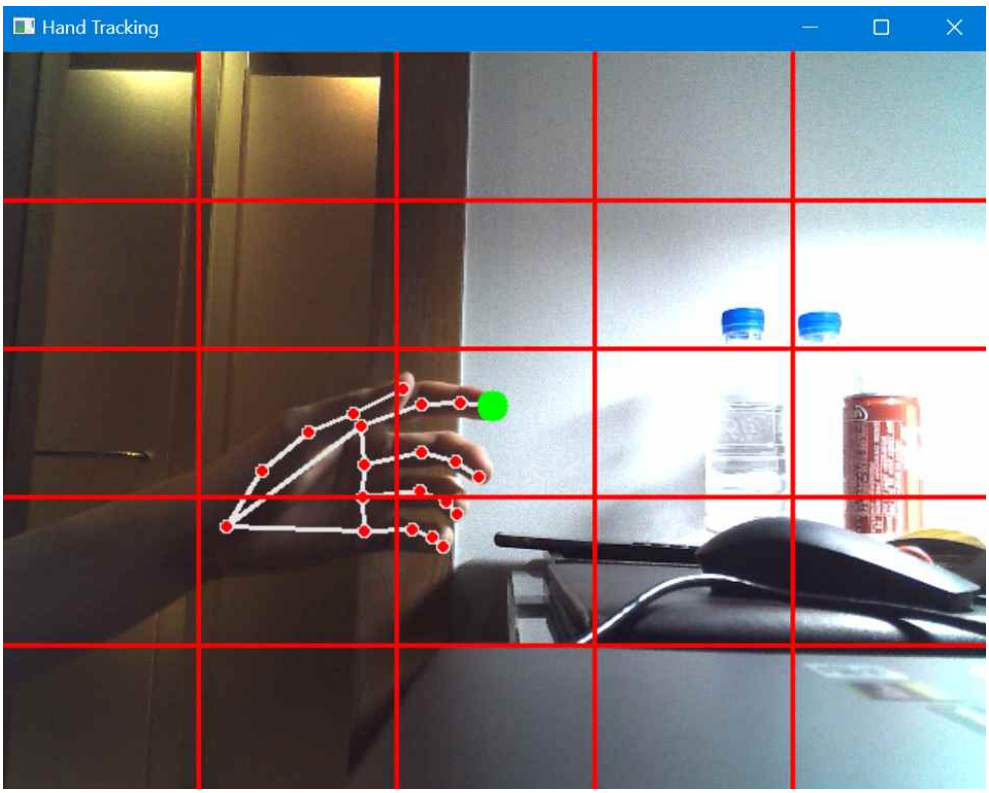
위 그림과 같이 영상을 받아드리면

● 움직임 차영상을 통해 배경을 제거하여 움직임을 검출

● HSV색좌표계를 활용하여 살색을 검출

두가지를 전처리로 시행하여 손이라는 객체의 정확도를 높이고,

이후, 손이미지를 학습한 인공지능을 결합시켜 정확도 높은 손을 검출하였습니다.



이렇게 하여 손이 위치하고 있는 좌표를 검출하였습니다.

다음으로 서브모터를 활용하여 x,y축으로 움직일 수 있는 팬틸트를 만들었습니다.

이후 정보를 카메라로부터 웹서버를 통해 아두이노로 받아드려서 정보를 받고 이를 각도로 변환하여 조명을 비춰주어서 넓은 작업대 환경에서 손의 위치에 적절하게 비춰주는 iot기반의 조명장치를 만들었습니다.

[코드]

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <Servo.h>
#include <NeoPixelConnect.h>
NeoPixelConnect p(4, 3, pio1, 1);
NeoPixelConnect p_(5, 3, pio1, 0);

// WiFi network information
char ssid[] = "kb_sc";         // your network SSID (name)
char pass[] = "kbsc4000";     // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;                  // your network key index number (needed only for WEP)

int status = WL_IDLE_STATUS;
WiFiServer server(80);
```

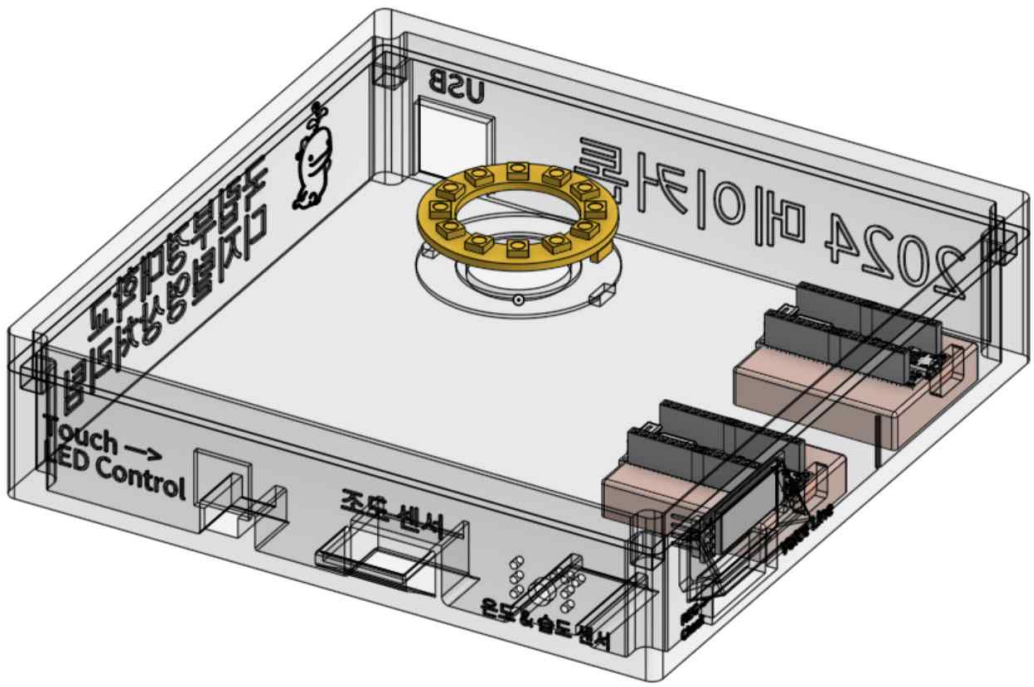
(정보를 받는 디폴트 & 웹서버 함수들 생략함)

```
// Servo control only if traking is 1
if (traking == 1 && currentLine.startsWith("GET /")) {
  int zone = currentLine.substring(5).toInt();
  handleServoPosition(zone);

}
```

이와 같이 traking토글 변수가 on이면 받아드린 정보를 통해 특정각도로 이동하게 코드를 구성하였습니다.

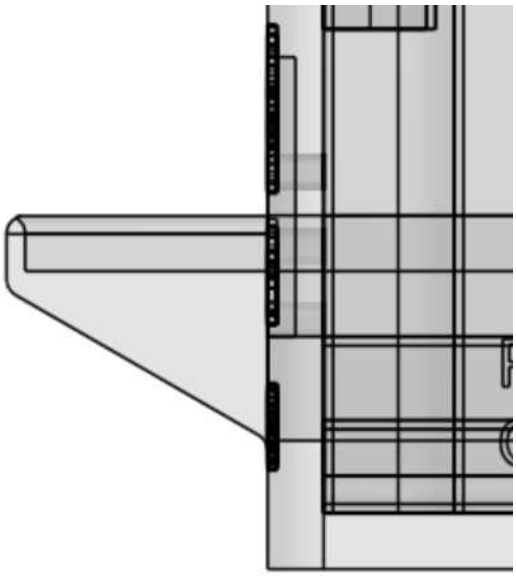
[3D 모델링]

우선 2개이상의 부품으로 구성하였고, 아랫 바텀케이스에 팀명과 부경대의 백경이를 글자와 이미지로 넣었습니다.

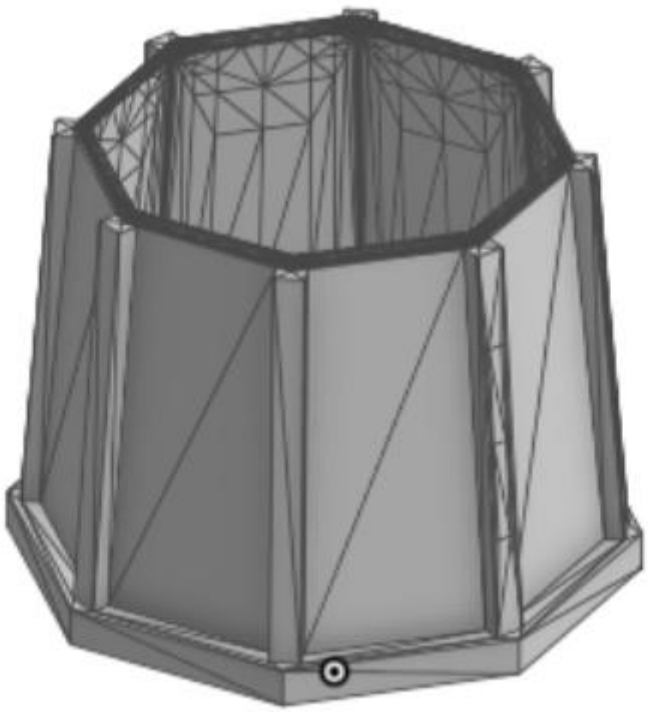


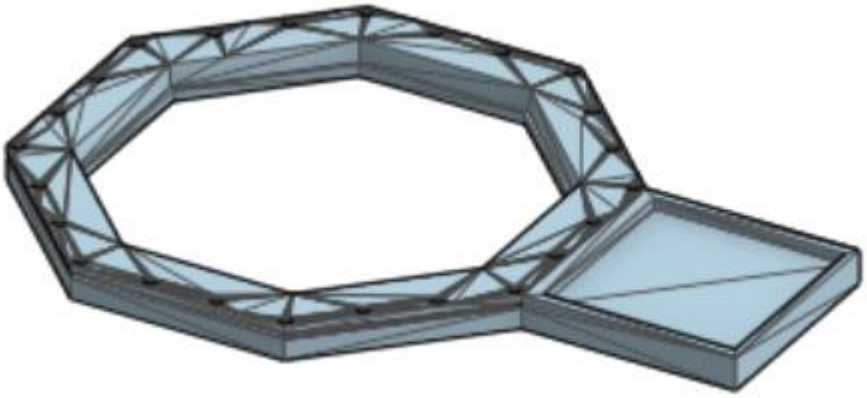
또한 케이스를 모델리하는데 각 구멍 또는 조립식으로 이루어지는 부분에는 공차를 주어서 실제 3D프린팅의 온도로 인한 팽창으로인해 발생하는 문제를 최소화 시키기위해 노력하였습니다.

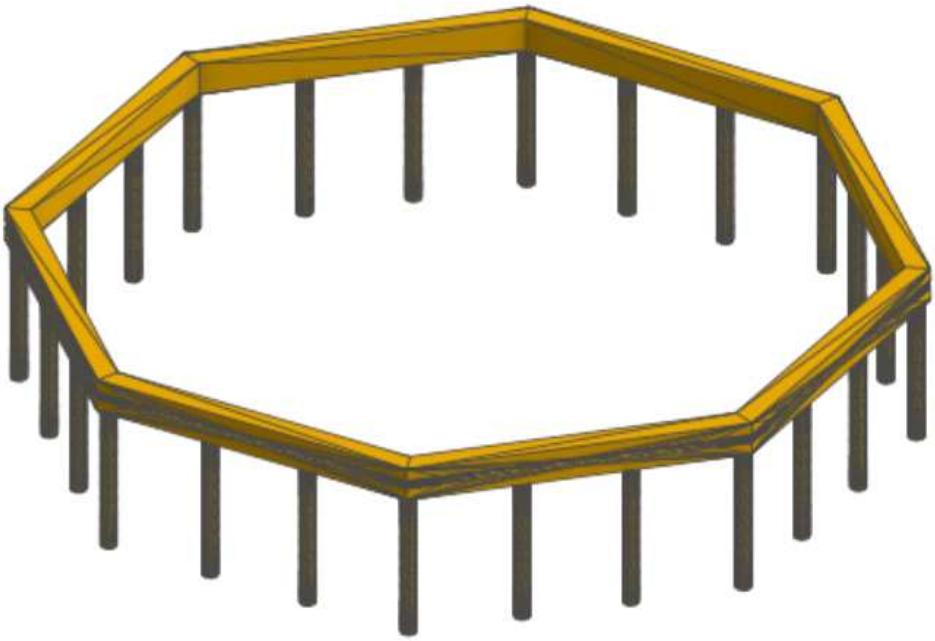
그리고 위의 사진에서 조도센서와 OLED를 받쳐주는 부분의 각도를 완만하게 모델링하여 서포트 없이도 출력물이 중력에 의해 무너져버리는걸 계산하였습니다.

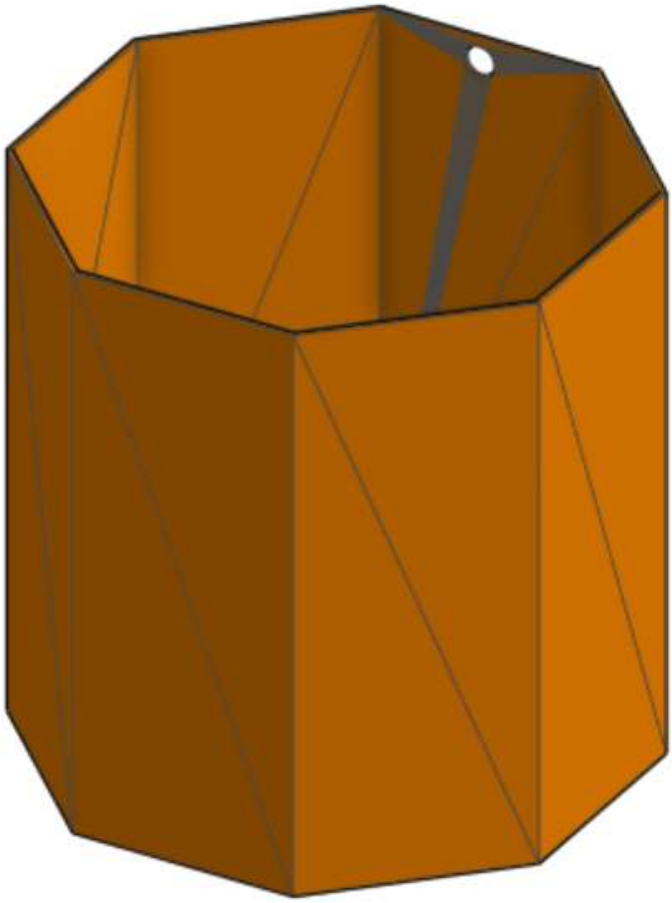


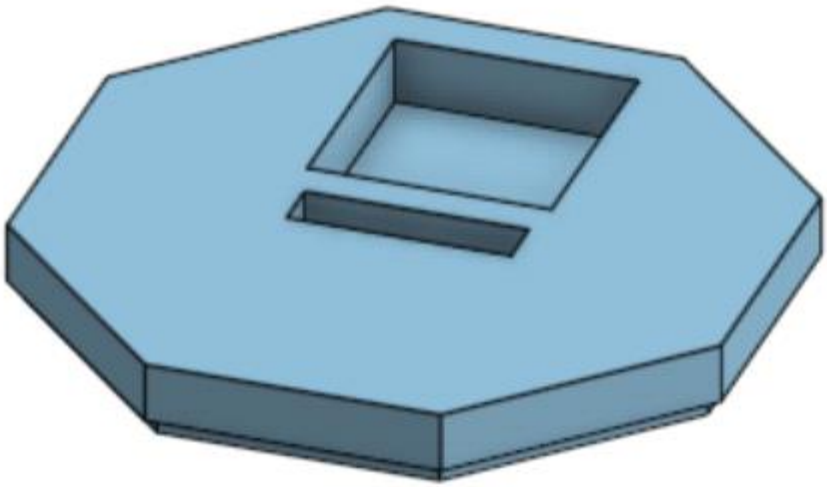
이후 다른 출력물을 보여드리겠습니다.

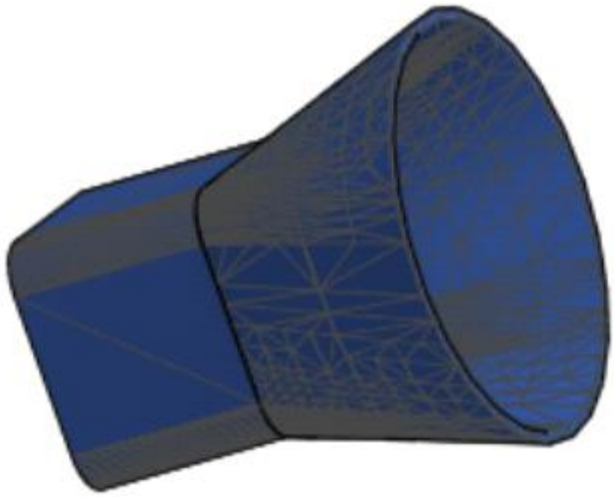




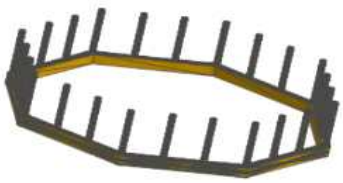








Multi Color



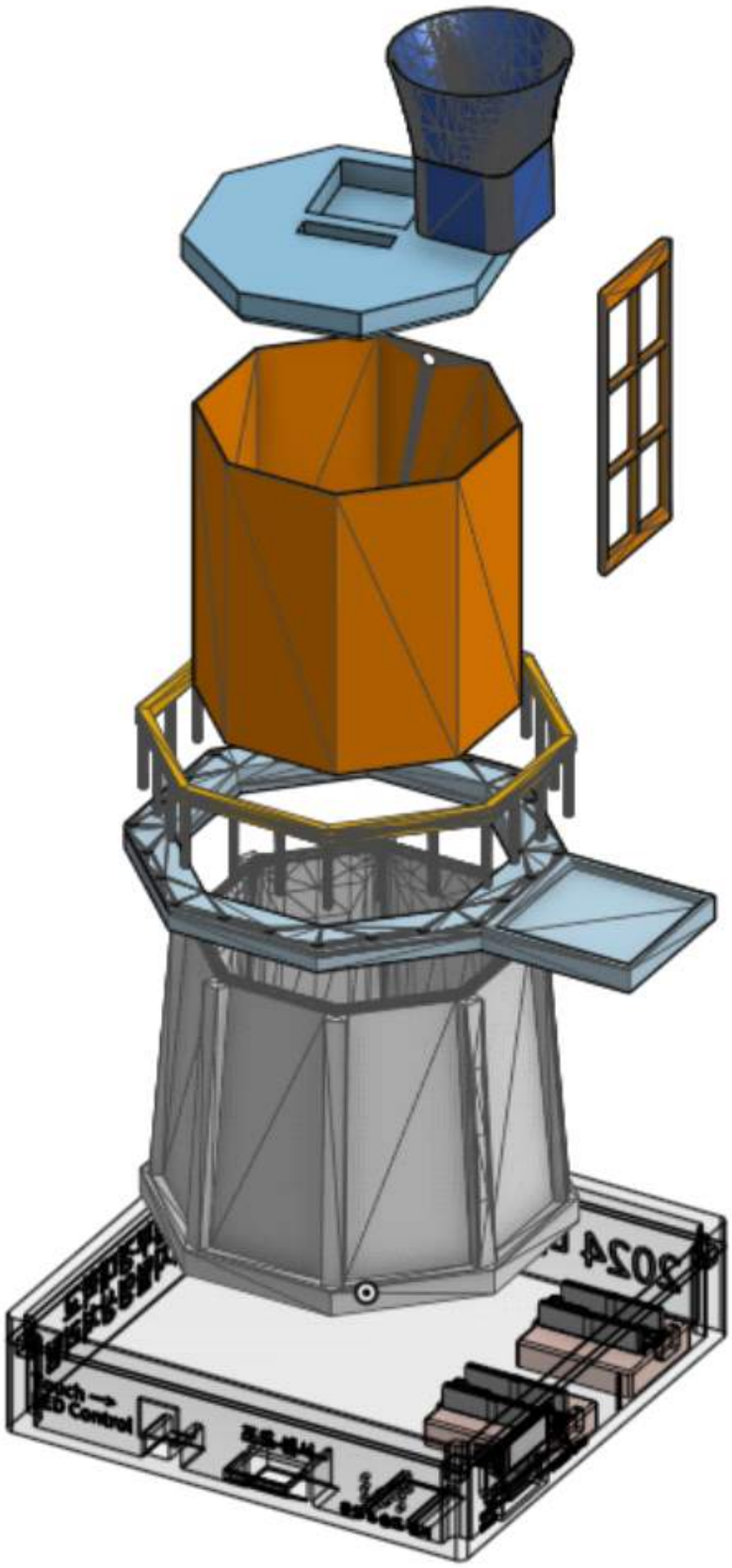
◆ Multi Color part 선정
→ Slicing parts 중 Mulit color 적용시 색의 표현이 가장 깔끔한 부분

→ 색의 변화 부분이 가장 잘 나뉘어 지는 부분

→ 지지대는 검정
연결하는 다리는 회색
연결하는 아래 파츠는 검정으로
뒤의 창문&램프 부분의 배색과 어울리도록 만듬

[어셈블링]

모든 출력물을 조립하면 다음과 같은 이미지를 가집니다.



[실제 완성물]

상기와 같이 특화교육 참가에 따른 학습결과보고서를 제출합니다.

2024년 08월 04일

국립부경대학교 공학교육혁신센터장귀하