# IoT클라우드 플랫폼 기말 프로젝트 보고서

1595016김현우

1594023유병찬

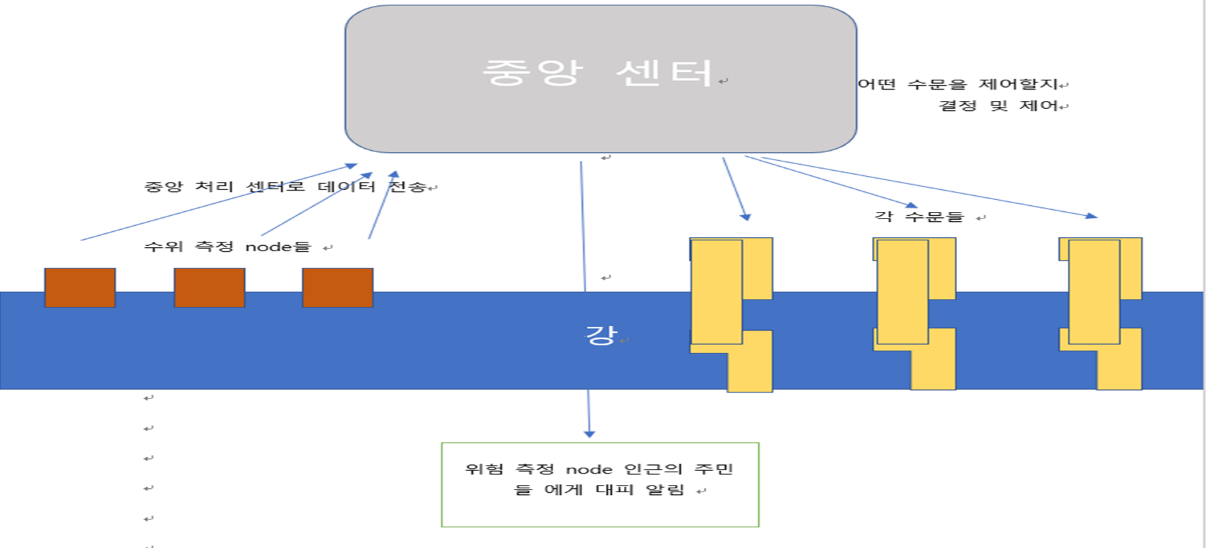
구성

1. IoT 서비스 개요
2. IoT 서비스 구조도
3. IoT 서비스 주요 기능
4. 소스 코드
5. 실행 화면 및 동영상
6. **IoT 서비스 개요**

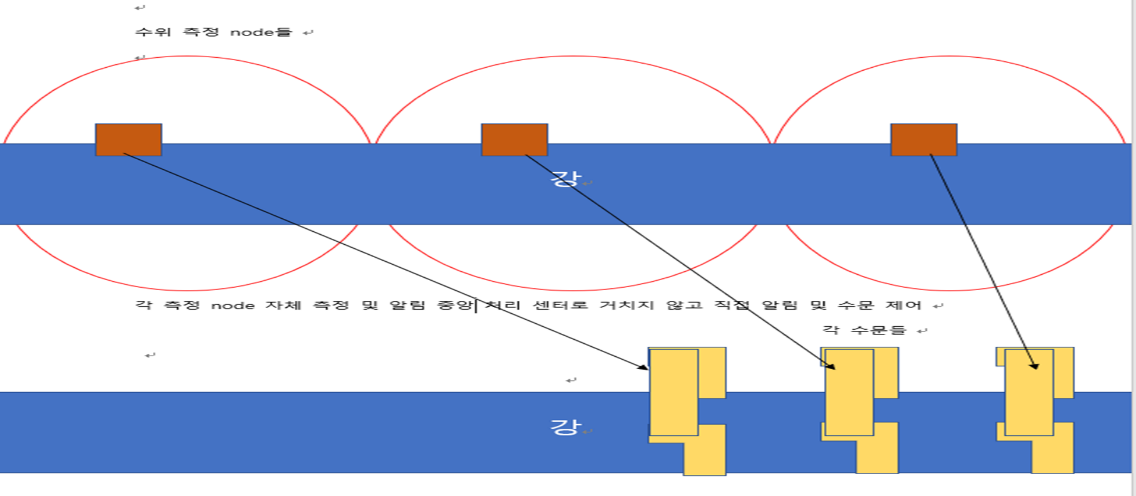
올 여름과 가을 남부지방 특히 경남지방에 자주 침수피해에 관한 뉴스를 접하였고 왜 홍수나 침수 대비 뉴스와 대피문자가 있음에도 피해를 보는 것 인지 특히 하천 주변 전통시장의 피해에 관하여 왜 미리 1층 상인들은 가전제품들을 미리 위층으로 옮기지않았는지 의문이 들었고 침수대피 및 알림 서비스를 이번 iot클라우드 플랫폼을 통해 구현할 수 있을 것 같아 이 서비스를 준비해 보았다.

1. **Iot 서비스 구조도**

기존 댐 및 수문제어와 대피 알림서비스가 이런 구조라면



이번에 우리팀은 아래와 같은 구조를 표현 해보고 싶었다.



인근 마을당 측정node가 직접 측정과 알림을 하며 대피수준의 수위측정시 각 node가 맡은 인근 주민에게 문자알림 서비스를 제공한다. 또한 각node가 맡은 수문/댐을 자동으로 제어한다.

1. **IoT 서비스 주요 기능**

Step 1. 수위 측정 노드는 2개의 수위센서를 평균을 낸 값을 측정한다.

Step 2. 측정된 값은 AWS 플랫폼으로 전송되고 그 값을 확인할 수 있다.

Step 3. 그 값을 토대로 ‘안전’, ‘경고’, ‘위험’, ‘대피’ 이벤트를 만든다.

Step 4. 각 이벤트의 구성은 일정 수위 이상일 경우 Green/Yellow/Red LED중 하나가 점등하게 되며 수위값과 이벤트의 상황을 보고 웹에서 UP, DOWN 버튼을 통해 수문을 제어한다.

Step5. 이 이벤트와 수위측정데이터에 관한 모니터링은 web을 통해 구현하였다.

1. **소스 코드**

* 1. 아두이노 mkr wifi 1010 수위측정 노드 소스코드
* 2. 아두이노 mkr wifi 1010 수문제어 노드 소스코드
* 3. 웹 구현 소스 코드
* **1. 아두이노 mkr wifi 1010 수위측정 노드 소스코드**

#include <ArduinoBearSSL.h>

#include <ArduinoECCX08.h>

#include <ArduinoMqttClient.h>

#include <WiFiNINA.h> // change to #include <WiFi101.h> for MKR1000

#include "arduino\_secrets.h"

#define LED\_1\_PIN 5 //RED

#define LED\_2\_PIN 4 //YELLOW

#define LED\_3\_PIN 3 //GREEN

#include <ArduinoJson.h>

#include "Led.h"

/////// Enter your sensitive data in arduino\_secrets.h

const char ssid[] = SECRET\_SSID;

const char pass[] = SECRET\_PASS;

const char broker[] = SECRET\_BROKER;

const char\* certificate = SECRET\_CERTIFICATE;

WiFiClient wifiClient; // Used for the TCP socket connection

BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrates with ECC508

MqttClient mqttClient(sslClient);

unsigned long lastMillis = 0;

Led led1(LED\_1\_PIN);

Led led2(LED\_2\_PIN);

Led led3(LED\_3\_PIN);

void setup() {

Serial.begin(115200);

while (!Serial);

if (!ECCX08.begin()) {

Serial.println("No ECCX08 present!");

while (1);

}

// Set a callback to get the current time

// used to validate the servers certificate

ArduinoBearSSL.onGetTime(getTime);

sslClient.setEccSlot(0, certificate);

}

void loop() {

if (WiFi.status() != WL\_CONNECTED) {

connectWiFi();

}

if (!mqttClient.connected()) {

// MQTT client is disconnected, connect

connectMQTT();

}

void connectMQTT() {

Serial.print("Attempting to MQTT broker: ");

Serial.print(broker);

Serial.println(" ");

while (!mqttClient.connect(broker, 8883)) {

// failed, retry

Serial.print(".");

delay(5000);

}

Serial.println();

Serial.println("You're connected to the MQTT broker");

Serial.println();

// subscribe to a topic

mqttClient.subscribe("$aws/things/MyMKRWiFi1010/shadow/update/delta");

}

// poll for new MQTT messages and send keep alives

mqttClient.poll();

// publish a message roughly every 5 seconds.

if (millis() - lastMillis > 10000) {

lastMillis = millis();

char payload[512];

getDeviceStatus(payload);

sendMessage(payload);

}

}

unsigned long getTime() {

// get the current time from the WiFi module

return WiFi.getTime();

}

void connectWiFi() {

Serial.print("Attempting to connect to SSID: ");

Serial.print(ssid);

Serial.print(" ");

while (WiFi.begin(ssid, pass) != WL\_CONNECTED) {

// failed, retry

Serial.print(".");

delay(5000);

}

Serial.println();

Serial.println("You're connected to the network");

Serial.println();

}

void sendMessage(char\* payload) {

char TOPIC\_NAME[]= "$aws/things/MyMKRWiFi1010/shadow/update";

Serial.print("Publishing send message:");

Serial.println(payload);

mqttClient.beginMessage(TOPIC\_NAME);

mqttClient.print(payload);

mqttClient.endMessage();

}

void onMessageReceived(int messageSize) {

// we received a message, print out the topic and contents

Serial.print("Received a message with topic '");

Serial.print(mqttClient.messageTopic());

Serial.print("', length ");

Serial.print(messageSize);

Serial.println(" bytes:");

// store the message received to the buffer

char buffer[512] ;

int count=0;

while (mqttClient.available()) {

buffer[count++] = (char)mqttClient.read();

}

buffer[count]='\0'; // 버퍼의 마지막에 null 캐릭터 삽입

Serial.println(buffer);

Serial.println();

void connectMQTT() {

Serial.print("Attempting to MQTT broker: ");

Serial.print(broker);

Serial.println(" ");

while (!mqttClient.connect(broker, 8883)) {

// failed, retry

Serial.print(".");

delay(5000);

}

Serial.println();

Serial.println("You're connected to the MQTT broker");

Serial.println();

// subscribe to a topic

mqttClient.subscribe("$aws/things/MyMKRWiFi1010/shadow/update/delta");

}

// Read led status

const char\* led = (led1.getState() == LED\_ON)? "ON" : "OFF";

const char\* ledtwo = (led2.getState() == LED\_ON)? "ON" : "OFF";

const char\* ledthree = (led3.getState() == LED\_ON)? "ON" : "OFF";

// make payload for the device update topic ($aws/things/MyMKRWiFi1010/shadow/update)

sprintf(payload,"{\"state\":{\"reported\":{\"WaterLevel\":\"%0.2f\",\"LED\":\"%s\",\"LED2\":\"%s\",\"LED3\":\"%s\"}}}", WaterLevel, led, ledtwo, ledthree);

}

if (strcmp(ledthree,"ON")==0) {

led3.on();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","ON");

sendMessage(payload);

} else if (strcmp(ledthree,"OFF")==0) {

led3.off();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","OFF");

sendMessage(payload);

}

}

DynamicJsonDocument doc(1024);

deserializeJson(doc, buffer);

JsonObject root = doc.as<JsonObject>();

JsonObject state = root["state"];

const char\* led = state["LED"];

const char\* ledtwo = state["LED2"];

const char\* ledthree = state["LED3"];

Serial.println(led);

Serial.println(ledtwo);

Serial.println(ledthree);

char payload[512];

if (strcmp(led,"ON")==0) {

led1.on();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","ON");

sendMessage(payload);

} else if (strcmp(led,"OFF")==0) {

led1.off();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","OFF");

sendMessage(payload);

}

if (strcmp(ledtwo,"ON")==0) {

led2.on(); sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","ON");

sendMessage(payload);

} else if (strcmp(ledtwo,"OFF")==0) {

led2.off(); sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","OFF");

sendMessage(payload);

}

* . **아두이노 mkr wifi 1010 수위측정 노드 소스코드(Led.cpp)**

**소스코드(Led.cpp)**

if (strcmp(ledthree,"ON")==0) {

led3.on();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","ON");

sendMessage(payload);

} else if (strcmp(ledthree,"OFF")==0) {

led3.off();

sprintf(payload,"{\"state\":{\"reported\":{\"LED\":\"%s\"}}}","OFF");

sendMessage(payload);

}

}

**소스코드(Led.h)**

#include <Arduino.h>

#define LED\_OFF 0

#define LED\_ON 1

class Led {

private:

int pin;

byte state;

public:

Led(int pin);

void init();

void on();

void off();

byte getState();

};

**소스코드(arduino\_secrets.h)**

// Fill in your WiFi networks SSID and password

#define SECRET\_SSID "\*\*\*\*\*\*"

#define SECRET\_PASS "00000000"

// Fill in the hostname of your AWS IoT broker

#define SECRET\_BROKER "\*\*\*\*\*\*\*\*\*"

// Fill in the boards public certificate

const char SECRET\_CERTIFICATE[] = R"(

-----BEGIN CERTIFICATE-----

MIICiTCCAXGgAwIBAgIVALZCZYu7v3233gQT71ZlygMHlqVzMA0GCSqGSIb3DQEB

\*\*\*\*

PE8QXoAlSak2KTcwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCB4AwDQYJKoZI

hvcNAQELBQADggEBAHo2rZRaRz07A/XX+kcTenK5FyRsihsZxl68dDH0GAeNcWHZ

lahQzQ0Wxvn9a5dEcF6c9qEB3NmKiwn9Vc8NekJ8Igv4t+96Jrf8bxGW8okRD97I

kE4B8rBIbfNU/igNoaRS8WMgKb1zdV4iCmZ+OTxl0j+gP6G57ILPdspbD1bwUMAQ

UDlPZTgE9sWBdCAfj9LLACADhpVxn1/tR09W6wcl/buUzVhopcGTC4EvZYG7xZwW

46nDiljW1RcKGL0+vbA/ar1d8KlDtgUAPMuhRqIend6YhwhITQiyoNApkPcrq5QK

hRDKJijjh9YCeZQ1/HvlM5NY+XrOVw7Xri6Nprw=

-----END CERTIFICATE-----

)";

* . **아두이노 mkr wifi 1010 수문제어 노드 소스코드**

#include <ArduinoBearSSL.h>

#include <ArduinoECCX08.h>

#include <ArduinoMqttClient.h>

#include <WiFiNINA.h> // change to #include <WiFi101.h> for MKR1000

#include "arduino\_secrets.h"

#define Motor\_1\_PIN 6 //motor

#define Motor\_2\_PIN 7 //motora

#include <ArduinoJson.h>

#include "Motor.h"

/////// Enter your sensitive data in arduino\_secrets.h

const char ssid[] = SECRET\_SSID;

const char pass[] = SECRET\_PASS;

const char broker[] = SECRET\_BROKER;

const char\* certificate = SECRET\_CERTIFICATE;

WiFiClient wifiClient; // Used for the TCP socket connection

BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrates with ECC508

MqttClient mqttClient(sslClient);

unsigned long lastMillis = 0;

Motor motor1(Motor\_1\_PIN);

Motor motor2(Motor\_2\_PIN);

void setup() {

Serial.begin(115200);

while (!Serial);

if (!ECCX08.begin()) {

Serial.println("No ECCX08 present!");

while (1);

}

ArduinoBearSSL.onGetTime(getTime);

sslClient.setEccSlot(0, certificate);

mqttClient.onMessage(onMessageReceived);

}

void connectWiFi() {

Serial.print("Attempting to connect to SSID: ");

Serial.print(ssid);

Serial.print(" ");

while (WiFi.begin(ssid, pass) != WL\_CONNECTED) {

// failed, retry

Serial.print(".");

delay(5000);

}

Serial.println();

Serial.println("You're connected to the network");

Serial.println();

}

void connectMQTT() {

Serial.print("Attempting to MQTT broker: ");

Serial.print(broker);

Serial.println(" ");

while (!mqttClient.connect(broker, 8883)) {

// failed, retry

Serial.print(".");

delay(5000);

}

void loop() {

if (WiFi.status() != WL\_CONNECTED) {

connectWiFi();

}

if (!mqttClient.connected()) {

// MQTT client is disconnected, connect

connectMQTT();

}

// poll for new MQTT messages and send keep alives

mqttClient.poll();

// publish a message roughly every 5 seconds.

if (millis() - lastMillis > 5000) {

lastMillis = millis();

char payload[512];

getDeviceStatus(payload);

sendMessage(payload);

}

}

unsigned long getTime() {

// get the current time from the WiFi module

return WiFi.getTime();

}

Serial.println();

Serial.println("You're connected to the MQTT broker");

Serial.println();

// subscribe to a topic

mqttClient.subscribe("$aws/things/MotorMKR/shadow/update/delta");

}

void getDeviceStatus(char\* payload) {

// Read temperature as Celsius (the default)

//float t = dht.readTemperature();

// Read led status

const char\* motor = (motor1.getState() == Motor\_UP)? "UP" : "DOWN";

const char\* motora = (motor2.getState() == Motor\_UP)? "UP" : "DOWN";

// make payload for the device update topic ($aws/things/MyMKRWiFi1010/shadow/update)

sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\",\"Motora\":\"%s\"}}}", motor, motora);

}

void sendMessage(char\* payload) {

char TOPIC\_NAME[]= "$aws/things/MotorMKR/shadow/update";

Serial.print("Publishing send message:");

Serial.println(payload);

mqttClient.beginMessage(TOPIC\_NAME);

mqttClient.print(payload);

mqttClient.endMessage();

}

void onMessageReceived(int messageSize) {

// we received a message, print out the topic and contents

Serial.print("Received a message with topic '");

Serial.print(mqttClient.messageTopic());

Serial.print("', length ");

Serial.print(messageSize);

Serial.println(" bytes:");

// store the message received to the buffer

char buffer[512] ;

int count=0;

while (mqttClient.available()) {

buffer[count++] = (char)mqttClient.read();

}

buffer[count]='\0'; // 버퍼의 마지막에 null 캐릭터 삽입

Serial.println(buffer);

Serial.println();

if (strcmp(motor,"UP")==0) {

motor1.up();

sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\"}}}","UP");

sendMessage(payload);

} else if (strcmp(motor,"DOWN")==0) {

motor1.down();

sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\"}}}","DOWN");

sendMessage(payload);

}

if (strcmp(motora,"UP")==0) {

motor2.up();

sprintf(payload,"{\"state\":{\"reported\":{\"Motora\":\"%s\"}}}","UP");

sendMessage(payload);

} else if (strcmp(motora,"DOWN")==0) {

motor2.down();

sprintf(payload,"{\"state\":{\"reported\":{\"Motora\":\"%s\"}}}","DOWN");

sendMessage(payload);

}

}

DynamicJsonDocument doc(1024);

deserializeJson(doc, buffer);

JsonObject root = doc.as<JsonObject>();

JsonObject state = root["state"];

const char\* motor = state["Motor"];

const char\* motora = state["Motora"];

Serial.print(motor);

Serial.println(motora);

char payload[512];

* **. 웹 구현 소스 코드**

<html>

<head>

* **아두이노 mkr wifi 1010 수문제어 노드 소스코드**

**(Motor.h)**

#include <Arduino.h>

#include <Servo.h>

#define Motor\_DOWN 0

#define Motor\_UP 1

class Motor {

private:

int pin;

byte state;

public:

Motor(int pin);

void init();

void up();

void down();

byte getState();

};

**(Motor.cpp)**

#include "Motor.h"

Servo servo;

Motor::Motor(int pin) {

// Use 'this->' to make the difference between the

// 'pin' attribute of the class and the

// local variable 'pin' created from the parameter.

this->pin = pin;

init();

}

void Motor::init() {

servo.attach(pin);

// Always try to avoid duplicate code.

// Instead of writing digitalWrite(pin, LOW) here,

// call the function off() which already does that

down();

state = Motor\_DOWN;

}

void Motor::up() {

servo.write(170);

state = Motor\_UP;

}

void Motor::down() {

servo.write(10);

state = Motor\_DOWN;

}

byte Motor::getState() {

return state;

}

**(arduino\_secrets.h)**

// Fill in your WiFi networks SSID and password

#define SECRET\_SSID "\*\*\*\*\*\*"

#define SECRET\_PASS "00000000"

// Fill in the hostname of your AWS IoT broker

#define SECRET\_BROKER "a27fcaeo1326fu-ats.iot.ap-northeast-2.amazonaws.com"

// Fill in the boards public certificate

const char SECRET\_CERTIFICATE[] = R"(

-----BEGIN CERTIFICATE-----

MIIChDCCAWygAwIBAgIVAIB7idzg6cbUtW3GVRtxxm3/MjRWMA0GCSqGSIb3DQEB

CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t

IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0xOTEyMTQwNjM0

MjlaFw00OTEyMzEyMzU5NTlaMBMxETAPBgNVBAMTCE1vdG9yTUtSMFkwEwYHKoZI

zj0CAQYIKoZIzj0DAQcDQgAEBI30gcxU305a0s1odmHGqNE8P2KMFucrCHG4KuuC

UWeNw0KLQWyrwGru4x4+MYS7s6bz60IA1CWpORZmWC6t6qNgMF4wHwYDVR0jBBgw

FoAUf0KldSOwq8wUuv7sK+jcl1z+A7UwHQYDVR0OBBYEFKk2zOeyPuzqv93YNrLS

/xdddcPbMAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMA0GCSqGSIb3DQEB

CwUAA4IBAQCeKXCpsjMv6dsV+JF3TnfFwB3GybDFDHWWiq7sBkI1EV3LMbIuSXRC

EpcbLcM/6jLN9Msg0Kkhv1De0Gdxyx0KUt//Bhcdjwn223VUYOhrXTDwSDuFiFoS

CDhtS+nmLTqRA7/6xTmsviVtCt93WbVxeehgQ4OjUgB4yPESySBC1+i7cvF2gMDk

g8WNpm09k6SBcvOlCO1vVuSYiGByRBqJIQ+5l08jHz4aa1JEDbjeuChbac00PUqL

wxOjia2UyaHSKNj3KtbQEon8lLzkVkbMRRyLXXwJy5MKtRW5q8ysp+bugxiXsFSa

Nov5HA+S62gsr/dDJUAorQspi69TU0y7

-----END CERTIFICATE-----

)";

* **웹 사이트 만들기**

**List\_devices.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>AWS Open API Sample</title>

<!-- JQuery 라이브러리 설정 -->

<script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>

<!-- 디바이스 조회 자바스크립트 로딩-->

<script src="list\_devices.js"></script>

<script src="controller\_devices.js"></script>

</head>

<body>

<div style="width:30%;float: left;">

<h3>My AWS API</h3>

<h4> 나의 디바이스 데이터 조회

<input type="button" value="조회" onclick="Start();" />

</h4>

<table id="mytable">

<thead style="background-color:grey">

<th>수 위 </th>

<th>RED </th>

<th>YELLOW </th>

<th>GREEN </th>

<th>발 생 시 간 </th>

</thead>

<tbody> </tbody>

</table>

</div>

<div style="width:70%;float: right;">

<h3>MY AWS Device Control</h3>

<h4> 나의 디바이스 제어

<input type="button" value="UP" onclick="UP();" />

<input type="button" value="DOWN" onclick="DOWN();" />

<div id="state" style="font-size: xx-large;">No Data</div>

</h4>

</div>

</body>

</html>

‘

**List\_devices.js**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>AWS Open API Sample</title>

<!-- JQuery 라이브러리 설정 -->

<script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>

<!-- 디바이스 조회 자바스크립트 로딩-->

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script src="controller\_devices.js"></script>

<script type="text/javascript">// API 시작

var aar = [];

var bar = [];

var i=0;

function Start() {

invokeAPI();

invokeAPI2();

emptyTable();

}

var invokeAPI = function() {

// 디바이스 조회 URI

// prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함

var API\_URI = 'https://gy0bfye8v3.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MyMKRWiFi1010/log?from=2019-11-29%2000:00:00&to=2019-12-13%2023:09:36';

$.ajax(API\_URI, {

method: 'GET',

contentType: "application/json",

success: function (data, status, xhr) {

var result = JSON.parse(data);

setDataList(result.data); // 성공시, 데이터 출력을 위한 함수 호출

setData(result.data);

//setData2(result.data);

console.log(data);

},

error: function(xhr,status,e){

// document.getElementById("result").innerHTML="Error";

alert("error");

}

});

};

// 테이블 데이터 삭제

var emptyTable = function() {

$( '#mytable > tbody').empty();

}

// 데이터 출력을 위한 함수

var setDataList = function(data){

$( '#mytable > tbody').empty();

data.forEach(function(v){

var tr1 = document.createElement("tr");

var td1 = document.createElement("td");

var td2 = document.createElement("td");

var td3 = document.createElement("td");

var td4 = document.createElement("td");

var td5 = document.createElement("td");

td1.innerText = v.WaterLevel;

td2.innerText = v.ledRed;

td3.innerText = v.ledYellow;

td4.innerText = v.ledGreen;

td5.innerText = v.timestamp;

if(v.ledRed == 'ON'){

td2.style.color="red"

}

if(v.ledYellow == 'ON'){

td3.style.color="#fff000"

}

if(v.ledGreen == 'ON'){

td4.style.color="green"

}

tr1.appendChild(td1);

tr1.appendChild(td2);

tr1.appendChild(td3);

tr1.appendChild(td4);

tr1.appendChild(td5);

$("table").append(tr1);

})

if(data.length>0){

// 디바이스 목록 조회결과가 있는 경우 데이터가 없습니다 메시지 삭제

} else if (data.length ==0) {

document.getElementById("result").innerHTML="No Data";

}

}

var setData = function(data){

data.forEach(function(v){

aar.push(v.timestamp);

bar.push(v.WaterLevel);

i++;

})

for(var j = 0 ; j < i ; j++){

bar[j] \*= 1;

}

}

setTimeout(function() {

drawChart2();

}, 5000);

google.charts.load('current', {'packages':['corechart']});

google.charts.setOnLoadCallback(drawChart2);

function drawChart2() {

var data = new google.visualization.arrayToDataTable([

['시간', '수심', ''],

[aar[i-10], bar[i-10],0],

[aar[i-9], bar[i-9],0],

[aar[i-8], bar[i-8],0],

[aar[i-7], bar[i-7],0],

[aar[i-6], bar[i-6],0],

[aar[i-5], bar[i-5],0],

[aar[i-4], bar[i-4],0],

[aar[i-3], bar[i-3],0],

[aar[i-2], bar[i-2],0],

[aar[i-1], bar[i-1],0],

]);

var options = {

title: '강 수심',

curveType: 'function',

legend: { position: 'bottom' }

};

var chart2 = new google.visualization.LineChart(document.getElementById('curve\_chart'));

chart2.draw(data, options);

}

</script>

</head>

<body>

<div style="width:30%;float: left;">

<h3>My AWS API</h3>

<h4> 나의 디바이스 데이터 조회

<input type="button" value="조회" onclick="Start();" />

</h4>

<table id="mytable">

<thead style="background-color:grey">

<th>수 위 </th>

<th>RED </th>

<th>YELLOW </th>

<th>GREEN </th>

<th>발 생 시 간 </th>

</thead>

<tbody> </tbody>

</table>

</div>

<div style="width:70%;float: right;">

<h3>MY AWS Device Control</h3>

<h4> 나의 디바이스 제어

<input type="button" value="UP" onclick="UP();" />

<input type="button" value="DOWN" onclick="DOWN();" />

<div id="state" style="font-size: xx-large;">No Data</div>

<div id="curve\_chart" style="width: 900px; height: 400px"></div>

</div>

</h4>

</div>

</body>

</html>

**Controller\_devices.js**

// API 시작

function UP() {

invokeAPI2();

invokeAPImotor();

}

function DOWN() {

invokeAPI3();

invokeAPImotor();

}

var invokeAPI2 = function(){

// 디바이스 조회 URI

// prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함

var API\_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';

var param = {

"tags" : [

{

"tagName": "Motor",

"tagValue": "DOWN"

},

{

"tagName": "Motora",

"tagValue": "UP"

}

]

};

$.ajax(API\_URI, {

method: 'PUT',

contentType: "application/json",

data : JSON.stringify(param),

success: function (data, status, xhr) {

console.log(data);

},

error: function(xhr,status,e){

// document.getElementById("result").innerHTML="Error";

alert("error");

}

});

};

var invokeAPI3 = function(){

// 디바이스 조회 URI

// prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함

var API\_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';

var param = {

"tags" : [

{

"tagName": "Motor",

"tagValue": "UP"

},

{

"tagName": "Motora",

"tagValue": "DOWN"

}

]

};

$.ajax(API\_URI, {

method: 'PUT',

contentType: "application/json",

data : JSON.stringify(param),

success: function (data, status, xhr) {

console.log(data);

},

error: function(xhr,status,e){

// document.getElementById("result").innerHTML="Error";

alert("error");

}

});

};

var invokeAPImotor = function() {

// 디바이스 조회 URI

// prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함

var API\_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';

$.ajax(API\_URI, {

method: 'GET',

contentType: "application/json",

success: function (data, status, xhr) {

var result = JSON.parse(data);

printData(result); // 성공시, 데이터 출력을 위한 함수 호출

console.log("data="+data);

},

error: function(xhr,status,e){

alert("error");

}

});

};

var printData = function(result){

if(result.state.reported.Motor == "UP" && result.state.reported.Motora == "DOWN"){

document.getElementById("state").innerHTML = " UP"

}

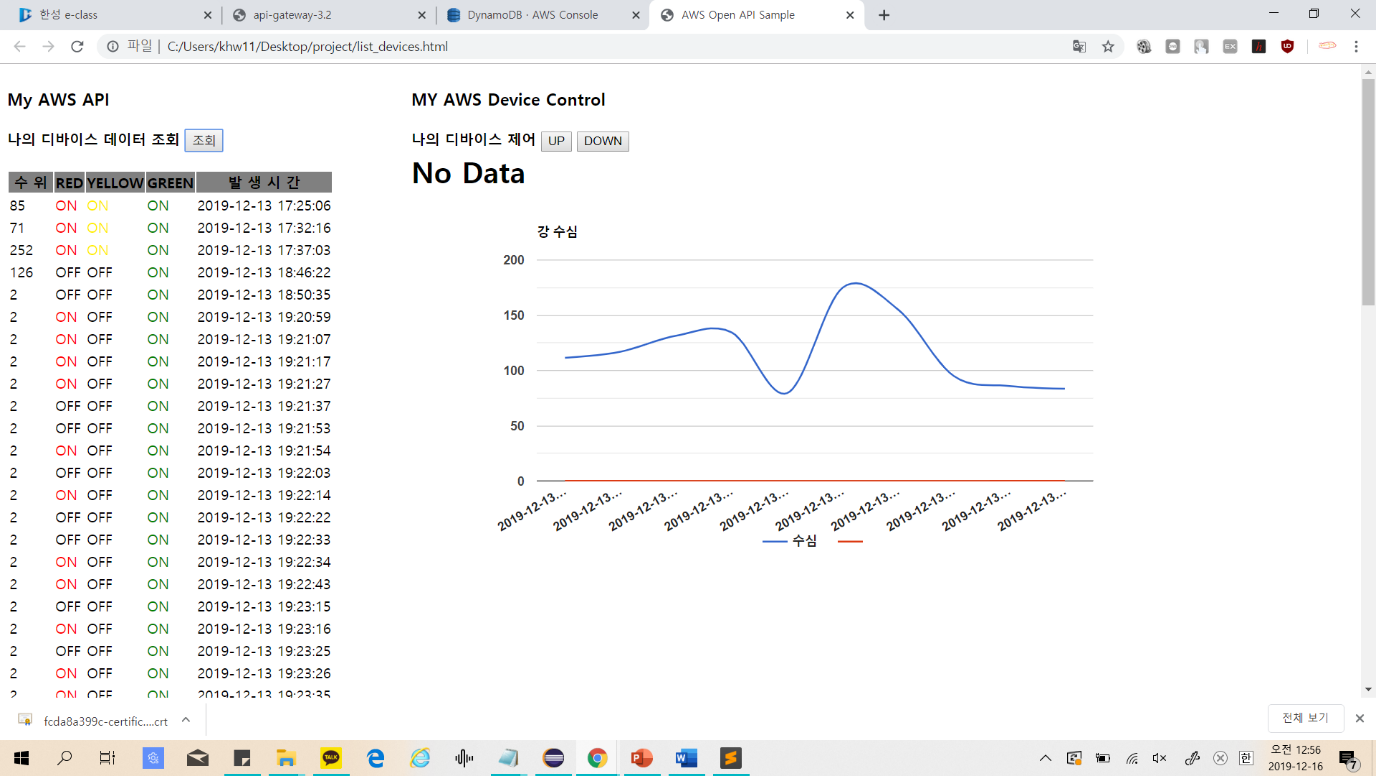
if(result.state.reported.Motor == "DOWN" && result.state.reported.Motora == "UP"){

document.getElementById("state").innerHTML = " DOWN"

}

}

1. **실행 화면 및 동영상**



<https://www.youtube.com/watch?v=pWMmUJvj6_E>