

ESP32로 만드는 IoT 월드 with 아두이노 IDE 2

프로젝트명 Ch12. JSON 인코딩과 디코딩

난이도 ★★☆☆☆



학습요약

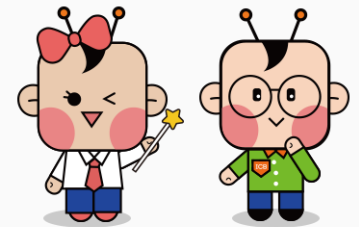
학습목표	Arduinojson라이브러리를 활용하는 법을 알아 보시다.
핵심 키워드	ESP32, IoT, 사물인터넷, JSON
준비물	ESP32
학습 시간	1시간
학습 난이도	★★☆☆☆

네트워크 전송시 사용하는

JSON데이터의 길이가 길어지거나 사용할 키값의 종류가 많아지면
코드가 복잡해지고 가독성이 떨어지게 될 것입니다.

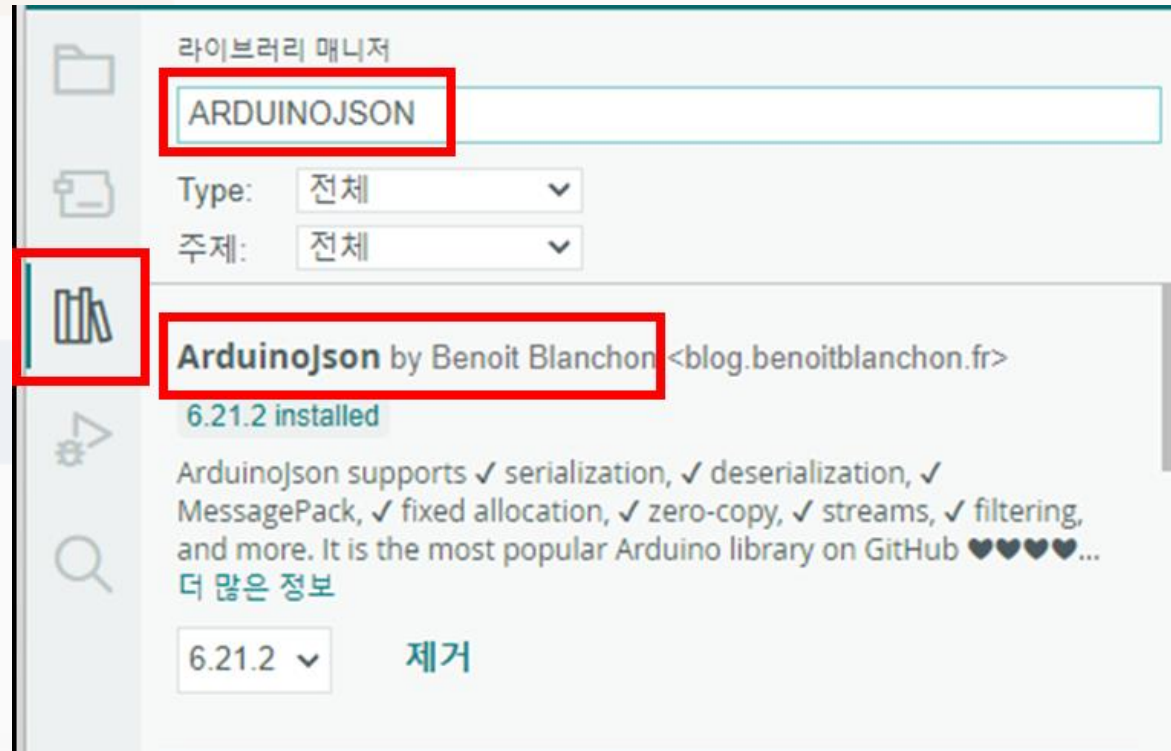
이런 경우 **ArduinoJson**라이브러리를 활용하면 좋습니다.

ArduinoJson라이브러리를 활용하는 방법을 알아보시다.



챕터 11의 코드 중,
Access token을 요청하는 코드를
Json Library를 활용하도록 수정해 봅시다.

Arduinjson by Benoit Blanchon을 찾아 설치합니다.



버퍼 사이즈 계산 Step1:

<https://arduinojson.org/v6/assistant/#/> 이동하여 Processor와 Mode, Input type 설정


1 Configuration 2 JSON 3 Size 4 Program

Step 1: Configuration

Processor

Mode

Input type
Uses the zero-copy mode: the JsonDocument stores pointer instead of copies of strings

Our sponsors


💡 This is the Assistant for ArduinoJson 6.21.2. Make sure the same version is installed on your computer.

Next: JSON

버퍼 사이즈 계산 Step2:
샘플 Json데이터 입력

1

2

3

4

ConfigurationJSONSizeProgram

Step 2: JSON

Examples: OpenWeatherMap, Reddit

Input

Enter here the JSON document you want your program to parse.

```
{
  "token_type": "bearer",
  "access_token": "c281d73b097",
  "expires_in": 43199,
  "refresh_token": "0a0c90af08f",
  "refresh_token_expires_in": 5184000,
  "scope": "account_email profile"
}
```

Input length: 193

Prettify

Previous

Next: Size

버퍼 사이즈 계산 Step3:
권장 버퍼 크기 확인

1

2

3

4

ConfigurationJSONSizeProgram

Step 3: Size

Data structures	96	Bytes needed to stores the JSON objects and arrays in memory i
Strings	0	Bytes needed to stores the strings in memory i
Total (minimum)	96	Minimum capacity for the <code>JsonDocument</code> .
Total (recommended)	96	Including some slack in case the strings change, and rounded to a power of two

► Tweaks (advanced users only)

Previous

Next: Program

12 : JSON코드를 저장할 버퍼를 선언해 줍니다.
17 : Access 토큰 갱신시 받는 JSON문자열 예시입니다.
27 : json 문자열을 디코딩합니다.
35, 36 : Key이름으로 해당 값을 얻을 수 있습니다.
37 : 문자열이 아닌 숫자, bool 형식으로 값을 얻을 수도 있습니다.

```
12 StaticJsonDocument<96> doc;  
13 // StaticJsonDocument<N>은 스택에 메모리를 할당합니다.  
14 // 힙에 할당하는 DynamicJsonDocument로 대체할 수 있습니다.  
15 // DynamicJsonDocument doc(200);  
16 // JSON 입력 문자열.  
17 char json[] = R"rawliteral({  
18     "token_type":"bearer",  
19     "access_token":"c281d73b097",  
20     "expires_in":43199,  
21     "refresh_token":"0a0c90af08f",  
22     "refresh_token_expires_in":5184000,  
23     "scope":"account_email profile"  
24 })rawliteral";  
25 Serial.println(json);  
26 // Deserialize the JSON document  
27 DeserializationError error = deserializeJson(doc, json);  
28 // Test if parsing succeeds.  
29 if (error) {  
30     Serial.print(F("deserializeJson() failed: "));  
31     Serial.println(error.f_str());  
32     return;  
33 }  
34 // Fetch values.  
35 const char* access_token = doc["access_token"];  
36 const char* refresh_token = doc["refresh_token"];  
37 long expires_in = doc["expires_in"];
```


실행 결과 :

JSON문자열의 데이터를 파싱하여 각각의 값으로 출력된 것을 확인할 수 있습니다

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM3')

새 줄

115200 baud

```
{
  "token_type": "bearer",
  "access_token": "c281d73b097",
  "expires_in": 43199,
  "refresh_token": "0a0c90af08f",
  "refresh_token_expires_in": 5184000,
  "scope": "account_email profile"
}
```

Access token: c281d73b097

Refresh token: 0a0c90af08f

Expire time : 43199

줄 44, 열 30 DOIT ESP32 DEVKIT V1 COM3 켜기 2

챕터 11의 코드 중, Access token을 요청하는 코드를
Json Library를 활용하도록 수정해 봅시다.

<https://arduinojson.org/v6/assistant/#/> 에서 필요한 버퍼 크기를 계산합니다.

The screenshot shows the 'Step 1: Configuration' page of the ArduinoJson Assistant. At the top, there is a progress bar with four steps: 1 Configuration (active), 2 JSON, 3 Size, and 4 Program. Below the progress bar, the 'Step 1: Configuration' section contains three dropdown menus: 'Processor' set to 'ESP32', 'Mode' set to 'Serialize', and 'Output type' set to 'char[N]'. These three dropdowns are highlighted with red rectangular boxes. To the right of these dropdowns is a section titled 'Our sponsors' featuring the 'PROGRAMMING ELECTRONICS ACADEMY' logo. At the bottom of the configuration section, there is a green box with a lightbulb icon and the text: 'This is the Assistant for ArduinoJson 6.21.2. Make sure the same version is installed on your computer.' A 'Next: JSON' button is located at the bottom right of the page.

25 ~ 31 : doc["키"] = "값" 쌍으로 데이터를 입력해 줍니다.

38 : doc변수를 줄바꿈이 없는 JSON문자열을 생성하여 jsonString에 대입하여 줍니다.

```
16 StaticJsonDocument<256> doc;
17
18 // StaticJsonObject는 스택에 메모리를 할당하고,
19 // 힙에 할당하는 DynamicJsonDocument로 대체할 수 있습니다.
20 //
21 // DynamicJsonDocument doc(200);
22
23 // 문서에 값 추가
24 doc["object_type"] = "text";
25 doc["text"] = "텍스트 영역입니다. 최대 " + String(200)+ "자 표시 가능합니다.";
26
27 doc["link"]["web_url"] = "https://developers.kakao.com";
28 doc["link"]["mobile_web_url"] = "https://developers.kakao.com";
29
30 doc["button_title"] = "바로 확인";
31
32 String jsonString ;
33 String jsonPrettyString ;
34
35 // 최소화된 JSON을 생성하여 직렬 포트로 보냅니다.
36 //
37 serializeJson(doc, jsonString);
38 Serial.println(jsonString);
```

실행 결과 :

주어진 data를 JSON문자열로 출력합니다. 줄바꿈 없는 양식이나 줄바꿈 있는 양식을 선택하여 출력할 수 있습니다.



The screenshot shows a serial monitor window titled "출력 시리얼 모니터 x". The message input field contains "Message (Enter to send message to 'DOIT ESP32 DEVKIT V". The settings are "No Line Ending" and "115200 baud". The output area displays a JSON object:

```
{
  "object_type": "text",
  "text": "텍스트 영역입니다. 최대 200자 표시 가능합니다.",
  "link": {
    "web_url": "https://developers.kakao.com",
    "mobile_web_url": "https://developers.kakao.com"
  },
  "button_title": "바로 확인"
}
```

The status bar at the bottom indicates "줄 20, 열 24 DOIT ESP32 DEVKIT V1 COM3 켜기 2".

카카오톡 서버에서 받은 JSON을
Decoding하여 Expire 값을 확인합니다.

```
79 // Deserialize the JSON document
80 DeserializationError error = deserializeJson(doc, payload);
81 long expire_time = doc["expires_in"];
82 Serial.println(expire_time);
```

토양 센서 값을 메세지 보내기 형식
에 맞게 JSON으로 인코딩하여 서버로
전송합니다.

```
122 doc["object_type"] = "text";
123 doc["text"] = "토양 센서 값 :" + String(sensorValue);
124 //doc["link"]["web_url"] = "https://www.naver.com";
125 JsonObject obj = doc.createNestedObject();
126 doc["link"] = obj;
127
128 String data;
129 // Generate the minified JSON and send it to the Serial port.
130 //
131 serializeJson(doc, data);
132 Serial.println(data);
133
134 http_code = http.POST("template_object=" + data);
```

서버에서 받은 JSON을 파싱하여
Access Token값을 알아 냅니다.
Refresh Token이 있는 경우 함께
갱신합니다.

```

178 // Deserialize the JSON document
179 DeserializationError error = deserializeJson(doc, response);
180
181 const char* atoken = doc["access_token"];
182 const char* rtoken = doc["refresh_token"];
183
184 Serial.print("Access token : ");
185 Serial.println(atoken);
186 Serial.print("New refresh token : ");
187 Serial.println(rtoken);
188
189 access_token = atoken;
190 //만료 1개월전부터 갱신되므로 data가 없을 수도 있음
191 if (rtoken != NULL) {
192     refresh_token = rtoken;
193 }
    
```



- 챗터 11과 동일하게 토양 센서 값이 일정한 시간 간격으로 카카오톡 메시지로 전송되는 것을 확인할 수 있습니다.



[저작권 안내]

*본 콘텐츠는 아이씨뱅크(ICBANQ)에 소유권이 있습니다. 소유권자의 허가를 받지 않고 무단으로 수정, 삭제, 배포, 상업적 사용을 할 수 없으며 위반시 민형사적 법적 처벌을 받을 수도 있습니다.