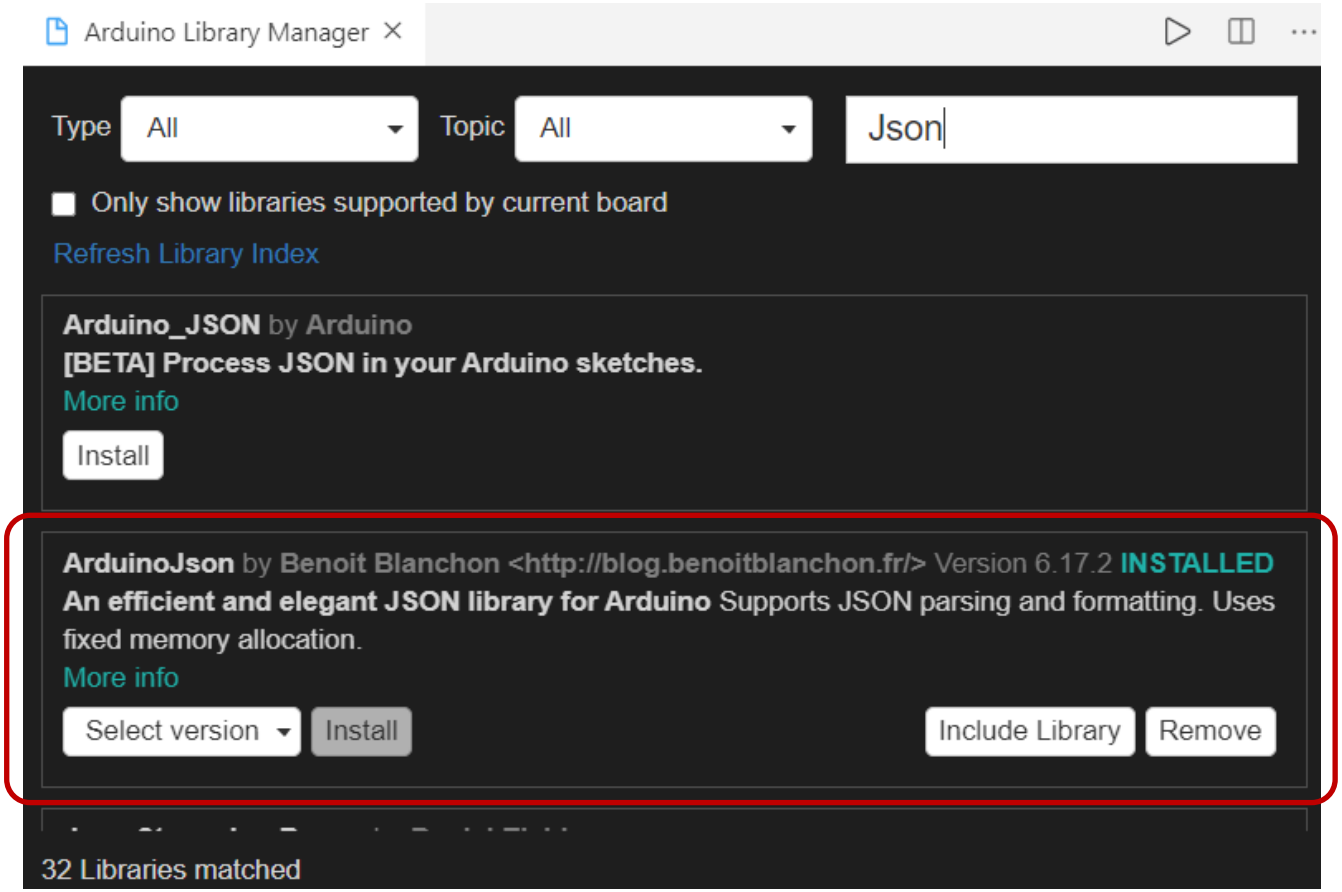
ArduinoJson

ArduinoJson

❖ 설치

- ○ Arduino Library Manager
  - ▪ Json 검색



ArduinoJson

❖ 주요 객체

- JsonDocument
  - DynamicJsonDocument doc(capacity);
    - 버퍼를 Heap에 배치
    - 속도는 늦으나 큰 용량 커버
  - StaticJsonDocument<capacity> doc;
    - 버퍼를 Stack에 배치
    - 속도는 빠르나 용량에 제한

❖ 참고문헌

- https://arduinojson.org/

ArduinoJson

❖ 역직렬화(Deserializing)

- JSON 문자열 -> 객체

```
String buf = "{\"sensor\": \"temp\", \"value\": 20.5}";

StaticJsonDocument<256> doc;

auto error = deserializeJson(doc, buf);
if (error) {
    Serial.print("deserializeJson() failed with code ");
    Serial.println(error.c_str());
    return;
}

const char* sensor = doc["sensor"];
const char* value =  doc["value"];
```

ArduinoJson

❖ 역직렬화(Deserializing)

- 타입 캐스팅

```
auto name = doc["name"].as<char*>();
auto stars = doc["stargazers"]["totalCount"].as<long>();
auto issues = doc["issues"]["totalCount"].as<int>();
```

ArduinoJson

❖ 직렬화(Serializing)

- 객체 -> JSON 문자열

```
StaticJsonDocument<256> doc;


doc["value"] = 42;
doc["lat"] = 48.748010;
doc["lon"] = 2.293491;
```

ArduinoJson

❖ 직렬화(Serializing)

- 객체 -> JSON 문자열, 배열

```
StaticJsonDocument<256> doc;

doc.add(1);
doc.add(2);

또는
doc[0] = 1;
doc[1] = 2;
```

ArduinoJson

❖ 직렬화(Serializing)

- 객체 -> JSON 문자열, 내장 객체

```
StaticJsonDocument<256> doc;

JsonObject obj1 = doc.createNestedObject();
obj1["key"] = "a1";
obj1["value"] = analogRead(A1);

JsonObject obj2 = doc.createNestedObject();
obj2["key"] = "a2";
obj2["value"] = analogRead(A2);
```

ArduinoJson

❖ 직렬화(Serializing)

- 객체 -> JSON 문자열, 내장 객체 배열

```
StaticJsonDocument<256> doc;

doc[0]["key"] = "a1";
doc[0]["value"] = analogRead(A1);

doc[1]["key"] = "a2";
doc[1]["value"] = analogRead(A2);
```

ArduinoJson

❖ 직렬화(Serializing)

- 객체 -> JSON 문자열

```
char output[256];

serializeJson(doc, output);
```