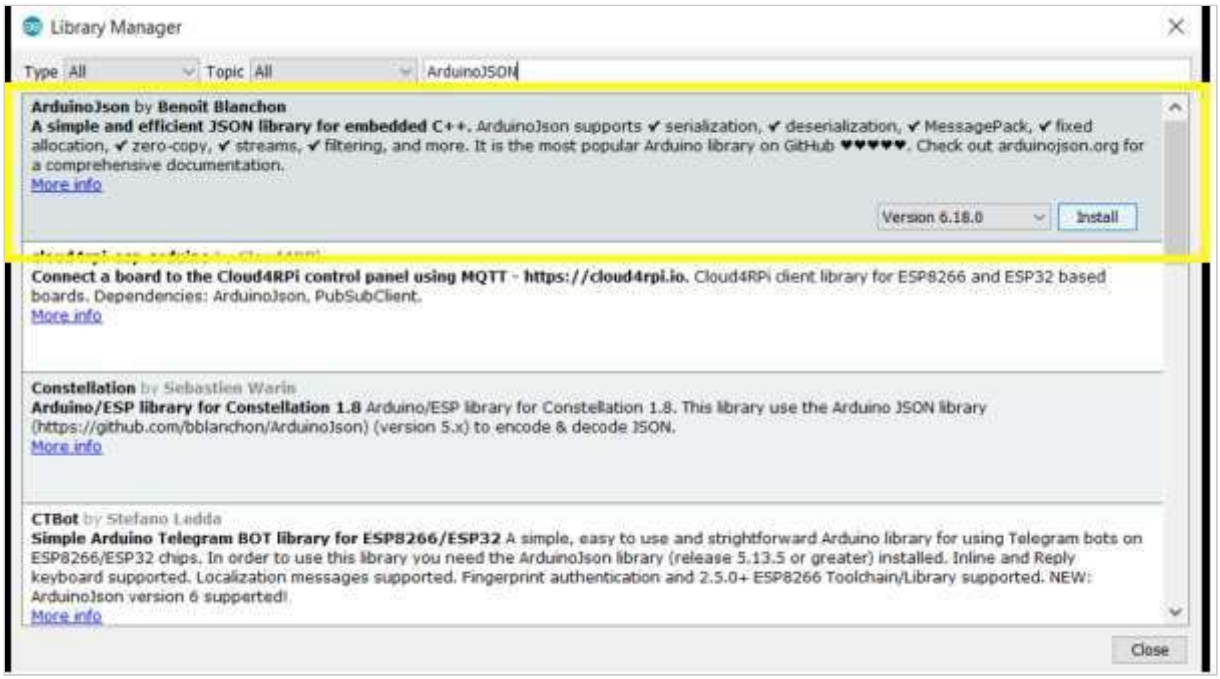
ArduinoJSON: Serialize and Deserialize

The **ArduinoJSON** library, as the name suggests, helps you work with JSON objects on Arduino. In order to install it, go to the Library Manager, and search for **ArduinoJSON**. Install the library by Benoit Blanchon.



This is one of the very heavily documented libraries. In fact, it has its own website: https://arduinojson.org/ . You can find answers to several questions on this website.

In this article, we will look at Serialization (generating a JSON document), and deserialization (parsing a JSON document) using this library.

Serialization

Let's start with Serialization. It is pretty straightforward. And if you've worked with python, the code will look all the more familiar.

Once you download the **ArduinoJSON** library, go to - **File→Examples→ArduinoJSON**

The example we should look at, for Serialization, is the **JsonGeneratorExample**. You will notice that the code is heavily commented, and you are encouraged to go through the comments in the example.

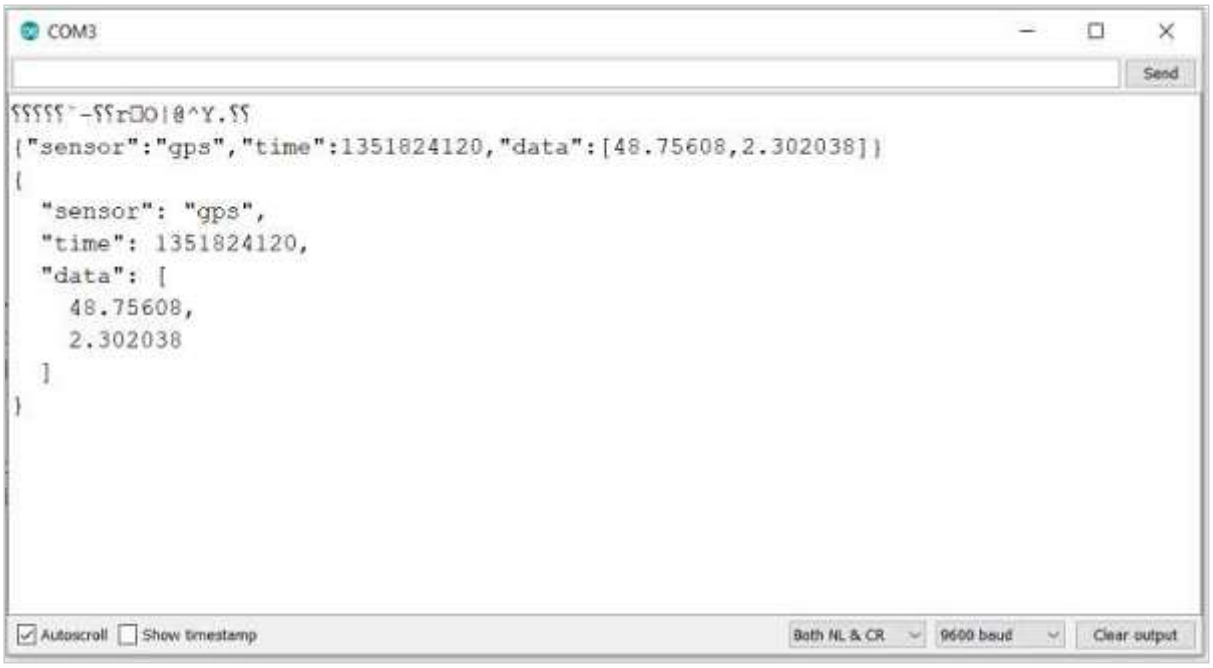
Without the comments, this is what the code looks like −

```
#include <ArduinoJson.h>

void setup() {
```

```
   Serial.begin(9600);
   while (!Serial) continue;
   StaticJsonDocument<200> doc;
   doc["sensor"] = "gps";
   doc["time"] = 1351824120;

   JsonArray data = doc.createNestedArray("data");
   data.add(48.756080);
   data.add(2.302038);

   serializeJson(doc, Serial);
   Serial.println();

   serializeJsonPretty(doc, Serial);
}
   void loop() {
}
```

Output

The code is, more or less, self-explanatory. The Serial Monitor output is shown below −



A couple of things to note −

- Adding key-value pairs to the JSON has this syntax − **doc_name[key] = value** The **JsonArray** type is used for adding array as a value for any key. The above code explains it well (for key "data")

- **serializeJSON** takes in two arguments: the JSON doc, and the destination buffer. Therefore, instead of directly printing to Serial, you could have used −

```
serializeJson(doc, serializedJSON);
Serial.println(serializedJSON);
```

And, the output would have been the same

Deserialization

Deserialization is the opposite of serialization. You have a JSON document (it could have come from any source, Serial, webserver, etc.) and you wish to parse it. Go to - **File** → **Examples** → **ArduinoJSON** and open the **JsonParserExample**. This again, is a heavily commented code, and you are encouraged to go through the comment in the example.

Example

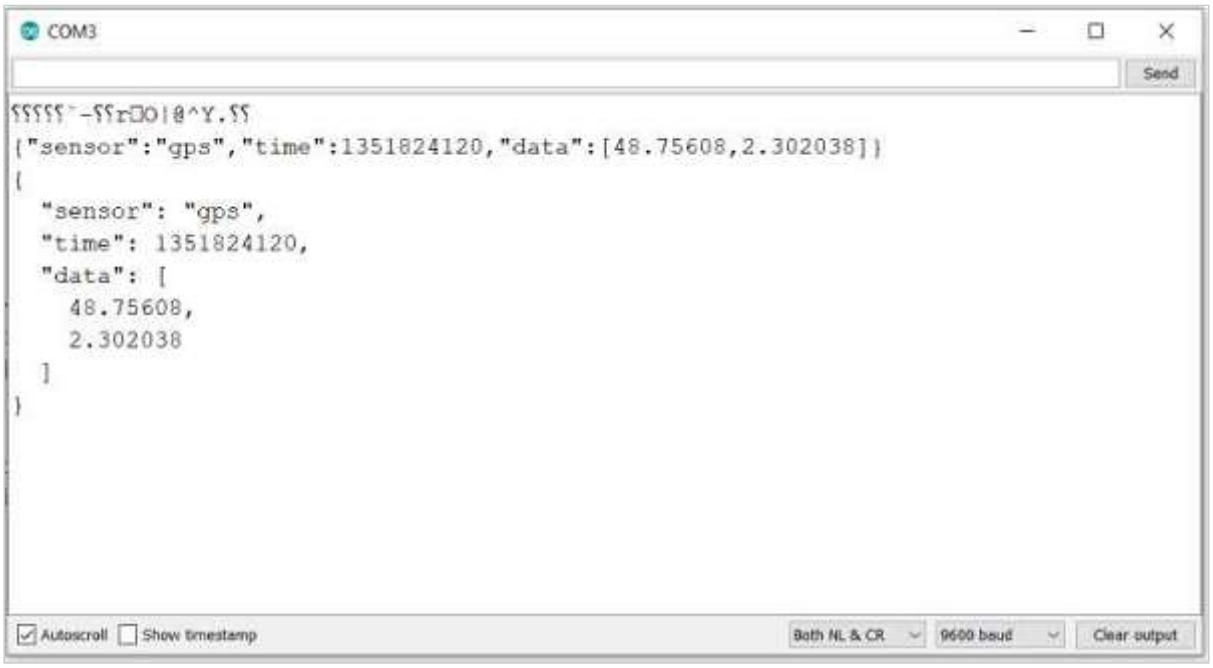
Without the comments, the code is −

```
#include <ArduinoJson.h>

void setup() {
   Serial.begin(9600);
   while (!Serial) continue;
   StaticJsonDocument<200> doc;

   char json[] = "{\"sensor\":\"gps\",\"time\":1351824120,\"data\":[48.756080,2.30203

   DeserializationError error = deserializeJson(doc, json);

   if (error) {
      Serial.print(F("deserializeJson() failed: "));
      Serial.println(error.f_str());
      return;
   }

   const char* sensor = doc["sensor"];
   long time = doc["time"];
   double latitude = doc["data"][0];
   double longitude = doc["data"][1];

   Serial.println(sensor);
   Serial.println(time);
   Serial.println(latitude, 6);
```

```
   Serial.println(longitude, 6);
}

void loop() {
}
```

Output

The Serial Monitor output is −



As you can see, the deserialization syntax is **deserializeJson(doc, json)** and it returns an error if there is any. It takes in two arguments: the JSON Document in which to store the deserialized output, and the buffer containing the JSON contents.

Once you have the deserialized content, extracting the content from it has the following syntax −

```
data_type var_name = doc[key]
```

You are encouraged to go through the other **ArduinoJSON** examples as well.