Search Free

arduino

Posts Wiki

Posted by u/frankcohen 1 day ago

JSON Procedural Scripting using ArduinoJSON and ESP32

I am engineering a watch to play home videos of my children on my wrist - my Reflections project. I need a procedural scripting language to identify the order of the videos to play. I encode the show in a JSON structure. I use ArduinoJSON 6, Arduino IDE 1.8.13, ESP32, and an SD card. This is my developer's journal of the problems that I encountered and the work-arounds I used.

Here is the JSON structure:

```
{
        "ReflectionsShow": {
                "title": "Candle blowing",
                "version": "1",
                "events": {
                        "1": {
                                "type": "event",
                                "name": "First try",
                                "trigger": "OnStart",
                                "comment": "Plays a sequence once upon startup",
                                "sequence": [
                                        {
                                                "playvideo": "Candle.mpg",
                                                "playaudio": "Tron.m4a"
                                        },
                                        {
                                                "playvideo": "Hurry.mpg",
                                                "playaudio": "Hurry.m4a"
                                        }
                                ]
                        },
                        "2": {
                                "type": "event",
                                "name": "Left button pressed",
                                "trigger": "LeftButtonPress",
                                "comment": "Plays when left button pressed",
                                "sequence": [
                                        {
                                                "playvideo": "Candle.mpg",
                                                "playaudio": "Tron.m4a"
                                        },
```

Search Free

```
                                                playaudio :  Hurry.m4a
                                        }
                                    ]
                                }
                            }
                        }
                }
```

The **sequence** key contains video/audio file values. I need a way to iterate through an unknown set of nested keys, hopefully using something as simple as this:

```
{
  JsonObject seqs = doc["ReflectionsShow"]["events"]["event"]["sequence"];
  for (auto sequence : seqs){
    if ( strcmp( sequence, "PlayAtOnce" )
    {
      //
    }

    if ( strcmp( sequence, "PlayAtEnd" )
    {
      //
    }
}
```

I like ArduinoJson 6 a lot. It has great documentation, an Assistant Utility to validate and generate C++ code based on your JSON structure, and Benoit Blanchon (the principal maintainer) is wonderfully responsive. Benoit seems fond of Wandbox. It is a Web based C++ composition and execution utility for quickly debugging code.

Here is the code to parse the above JSON structure:

```
  Serial.println("ReflectionsOS JSON Experiments");

  if ( !SD.begin(SD_CS_PIN, SPI, 80000000) ) /* SPI bus mode */
  {
    Serial.println("SD init failed");
  }

  File file = SD.open( "/tempo/StartupShow/candle.ref" );

  DynamicJsonDocument doc(1000);
  auto error = deserializeJson(doc, file);
  if (error) {
      Serial.print(F("deserializeJson() failed with code "));
      Serial.println(error.c_str());
      return;
```

Search Free

```
JsonVariant title = doc.getMember( "ReflectionsShow" ).getMember( "title" );
if ( title.isNull() )
{
  Serial.println( "Title not found");
}
else
{
  Serial.print( "Title: " );
  Serial.println( title.as<const char*>() );
}

JsonObject events = doc["ReflectionsShow"]["events"];
if ( events.isNull() )
{
  Serial.println( "Events not found");
}

for (auto eventseq : events)
{
    String eventskey = eventseq.key().c_str();
    String eventsvalue = eventseq.value().as<const char*>();  //ver 6.18

    Serial.print( "Event " );
    Serial.println( eventskey );

    JsonObject event = doc["ReflectionsShow"]["events"][ eventskey ];

    String eventname = event["name"].as<char*>();

    Serial.print( "name " );
    Serial.println( eventname );

    JsonArray sequence = doc["ReflectionsShow"]["events"][eventskey]["sequence"];
    for (JsonObject step : sequence) {
      const char* video = step["playvideo"];
      const char* audio = step["playaudio"];

      Serial.println( "--" );
      Serial.print( "video " );
      Serial.println( video );
      Serial.print( "audio " );
      Serial.println( audio );
    }
}

Serial.println("done");
```

Search Free

Running this code gives me this output:

```
ReflectionsOS JSON Experiments
Title: Candle blowing
Event 1
name First try
--
video Candle.mpg
audio Tron.m4a
--
video Hurry.mpg
audio Hurry.m4a
Event 2
name Left button pressed
--
video Candle.mpg
audio Tron.m4a
--
video Hurry.mpg
audio Hurry.m4a
done
```

I made a bunch of mistakes:

1. Objects vs Arrays. I defined the sequence as a nested set of objects. Benoit told me "the order of the members in a JSON object is not guaranteed; it turns out that the current version of ArduinoJson returns them in the same order, but a future version could (and most likely will) return them in alphabetical order." I changed the definition to an array using [and] characters.
2. Duplicate Keys. My original JSON structure used duplicate keys. For example {"play":"myfile1", "play":"myfile2"}. While JSON Lint will not validate duplicate keys, there is no restriction against duplicate keys in the JSON standard. What's worse is ArduinoJson will return only the last of the duplicate keys. The bigger problem is most JSON parsers use key dictionaries in their underlying implementation. Key dictionaries don't guarantee preserving the order of the stored keys. Benoit recommends using JSON arrays and that worked fine for me.
3. I thought using the ArduinoJson doc[] (or doc.getMember(“”) technique would return a JsonObject, that’s why I added .as<JsonArray>() to the end. This is not needed. By casting it I get a null value for array.
4. I thought the JsonArray initializer only accepted DynamicJsonDocument or StaticJsonDocument. It works fine with JsonObjects.
5. I didn’t expect ArduinoJson’s initializers to work so well with various different object types. That’s really cool. For example, it didn’t occur to me that I could do further value filtering when the value is an object: for (JsonObject step : sequence) { const char* video = step["playvideo"]; ...
6. I thought using a JsonArray would lose the context of the key I was iterating through. For example, when I’m in this loop:

Search Free

```
JsonArray array = doc.as<JsonArray>();
for(JsonVariant v : array) {
Serial.println(v.as<int>());
}
```

I thought I would not be able to use v as a JsonObject to do further processing on that key and its nested values below. I was wrong and it works fine.

Conclusion

In my experience ArduinoJson is excellent: documentation, the object implementation, and the Assistant are wonderful. I had my own misunderstandings (above) that kept me from being successful. With great support from Benoit I was able to succeed.

My watch project is called Reflections and is hosted here.

2 Comments Share Edit Post ••• 90% Upvoted

Comment as frankcohen

Markdown Mode Comment

SORT BY BEST

NoBulletsLeft 1 day ago

This is really cool. I recently had a project where I needed to sequence a bunch of stepper motors, so I took it as an opportunity to start working on a reusable sequence engine. Mine is done in code with the sequence being an array of command/parameter structs, but what you have here is really nice and I should really consider rewriting mine in a similar fashion since it makes it much easier for a non-technical person to specify the behavior. I'm also a big fan of data-driven designs.

2 Reply Give Award Share Report Save

frankcohen 1 day ago

Thanks for the compliment, I'm glad it has an application on your project. Please post your work to help out everyone else.

1 Reply


Search
Free