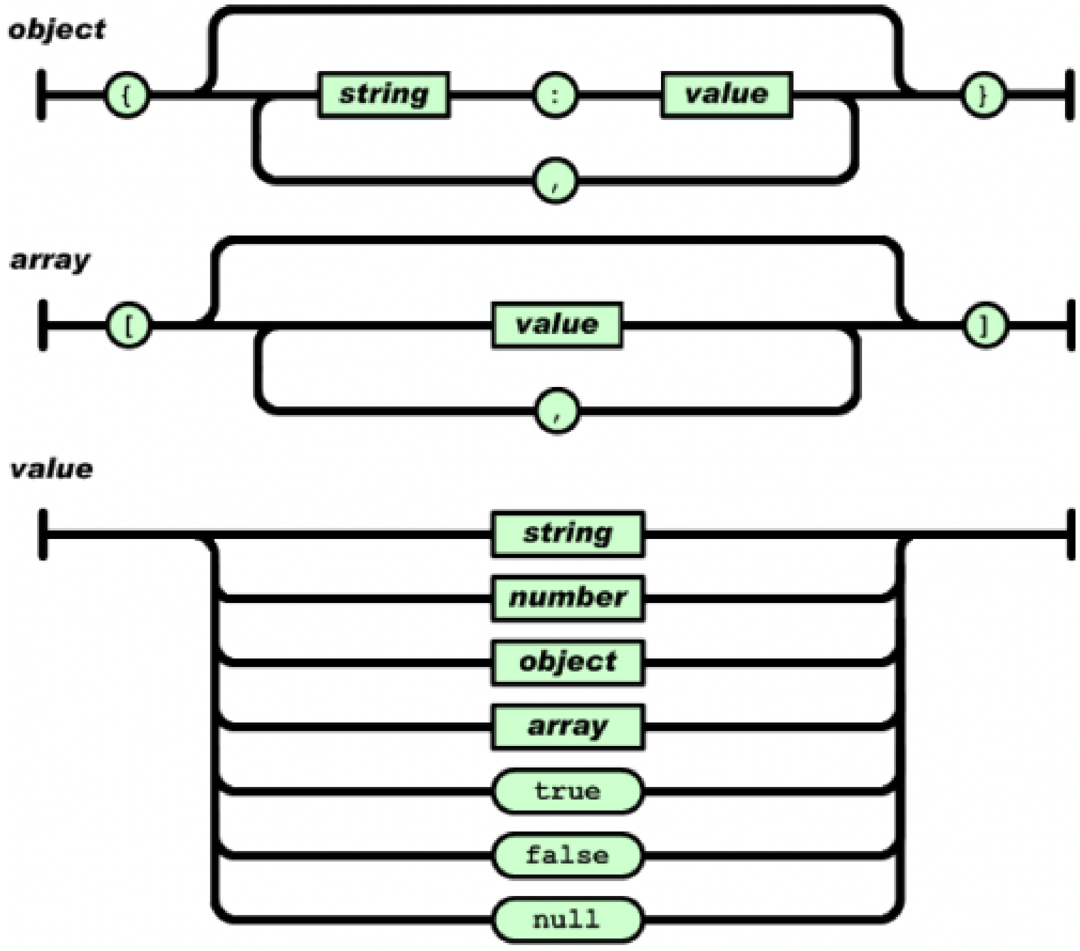
aJson – Handle JSON with Arduino

AUGUST 14, 2010
in INTERNET OF THINGS



Update: There has bee a major API change in Version 1.0 refer to the dedicated aJson page.

Exchanging data with other computers can be a daunting task with Arduino. No matter if you just want to pass some information to Processing, to a Web Service or something else – You always have to encode the data and decode the answer.

There always have been solutions like XML for structured data. But XML is hard to decode, complicated an takes up a lot of space. And then there is JSON. JSON is an easy and efficient way to transfer data. A complex data structure can for example represented as

```
{
"name": "Jack (\"Bee\") Nimble",
"format": {
"type": "rect",
"width": 1920,
"height": 1080,
"interlace": false,
"frame rate": 24
}
}
```

aJson is an Arduino library to enable JSON processing with Arduino. It easily enables you to decode, create, manipulate and encode JSON directly from and to data structures. By this you don't have to bother with data encoding and decoding – this will hand aJson for you. It is based on the cJSON implementation, reduced in size and removing one or two features.

Using aJson

To use aJson get it from the git repository or download the latest version. Unpack the zip file (or the git files) into an Folder 'aJson' in the Arduino library directory 'libraries\'. Due to the early stage of this library you can encounter problems. Simply submit a ticket and it will fix it as soon as possible. On github you will also find a simple Arduino sketch to test the library, or as a starting point.

To parse the above structure with aJson you simply convert it to a object tree:

```
aJsonObject* jsonObject = aJson.parse(json_
```

(assuming you got the JSON string in the variable json_string – as a char*). This is an object. We're in C. We don't have objects. But we do have structs. Therefore the objects are translated into structs, with all the drawbacks it brings. Now we can e.g. retrieve the value for name:

```
aJsonObject* name = aJson.getObjectItem(roo
```

The value of name can be retrieved via:

```
Serial.println(name->value.valuestring);
```

Note that the aJsonObject has a union 'value' which holds all possible value types as overlays – you can get only useful data for the type which you have at hand. You can get the type as

```
 name->type
```

which can be either aJson_False, aJson_True, aJson_NULL, aJson_Number, aJson_String, aJson_Array or aJson_Object. For aJson_Number you can use value.number.valueint or value.number.valuedouble, for aJson_String you can use value.valuestring, for True or False, you can use value.valuebool.

If you want to change the name, you can do so by

```
aJson.getObjectItem(jsonObject,"name")->valu
```

To render the object back to a string you can simply call

```
char *json_String=aJson.print(jsonObject);
```

Finished? Delete the root (this takes care of everything else).

```
aJson.delete(root);
```

This deletes the objects and all values referenced by it. So take care for your string, you directly assigned. They are deleted too. If you want to see how you'd build this struct in code?

```
aJsonObject *root,*fmt;
root=aJson.createObject();
aJson.addItemToObject(root, "name",
aJson.createString("Jack (\"Bee\") Nimble"));
aJson.addItemToObject(root, "format", fmt =
aJson.createObject());
aJson.addStringToObject(fmt,"type",
 "rect");
aJson.addNumberToObject(fmt,"width",
 1920);
aJson.addNumberToObject(fmt,"height",
 1080);
aJson.addFalseToObject (fmt,"interlace");
aJson.addNumberToObject(fmt,"frame rate",
 24);
```

The whole library (nicely provided by cJSON) is optimized for easy usage. You can create and modify the object as easy as possible.

Limitations of aJson

Unfortunately everything comes for a price and complex data structures and text processing is not really the primary domain for Arduino. So some simplifications had to be made:

aJson is not really suitable for ATmega168 based Arduinos – there is just not enough RAM – and the library itself eats up toe 80% of the flash memory.

aJson cannot handle Unicode.

Lists and Objects can only handle up to 256 entries.

All strings are case sensitive, this is no real problem for values. But if you want to find a name you have keep in mind that it works case sensitive.

A lot of these limitations will be gone in the future versions of aJson.

Further Development of aJson

Currently aJson is considered to be in a beta phase. It is usable, but it can get better. The most important task is to further reduce the memory footprint. Therefore the API may change a bit in next versions. When the library is better tested and more evolved it will be package as real Arduino library.

I hope that many people will use and test the library. So go ahead and download aJson!