Ein Newsticker per API Call & JSON

Du möchtest die neuesten Schlagzeilen direkt in deinem Seriellen Monitor lesen? Dann wird es Zeit für deinen eigenen Newsticker!

In diesem Projekt verbindest du deinen ESP32 mit dem Internet, rufst über einen API Call die aktuellen Nachrichten ab, parst JSON-Daten und gibst die Schlagzeilen in deinem Seriellen Monitor aus.

Wir konzentrieren uns in diesem Projekt zunächst auf die Software. Sobald dein Newsticker funktioniert, kannst du ihn um ein Display erweitern, um die Nachrichten dort anzuzeigen.

Hinweis: Für dieses Projekt benötigst du einen ESP32, den du schon für ein paar Euro kaufen kannst. Ein Board aus der Arduino- oder ESP8266-Familie hat für den Newsticker vermutlich zu wenig Arbeitsspeicher. Wenn du diesen Controller bisher noch nicht verwendet hast, wirf zunächst einen Blick in dieses Tutorial.

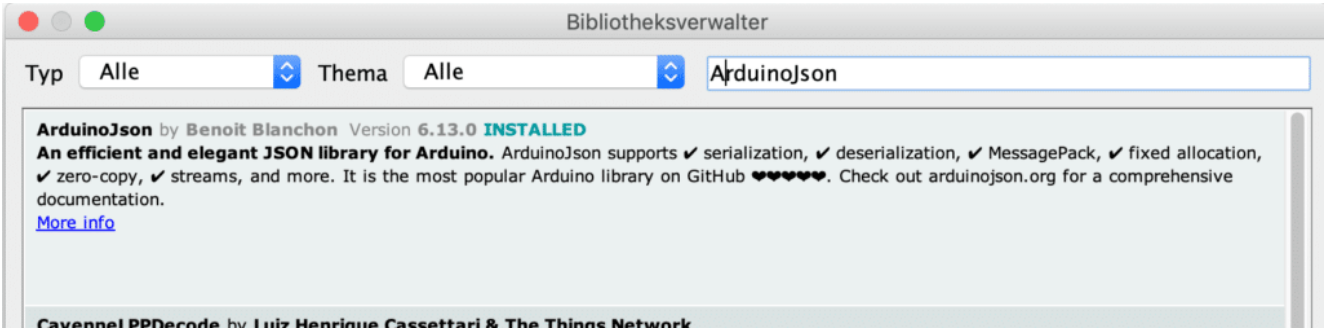
Der Code für deinen Newsticker

Als allererstes benötigst du drei Bibliotheken, die du wie gewohnt in deinem Sketch gleich zu Beginn lädst:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
```

Die ersten beiden Bibliotheken sollten in deiner Arduino IDE bereits vorinstalliert sein. Falls nicht, kannst du sie unter *Werkzeuge/Bibliotheken verwalten* hinzufügen.

Die Bibliothek **ArduinoJson.h** vom Entwickler Benoit Blanchon findest du in deinem Bibliotheksverwalter, indem du einfach nach „ArduinoJson“ suchst und installierst.



Bevor es richtig losgehen kann, trage als nächstes deine WLAN-Zugangsdaten in den Sketch ein:

```
const char* ssid = "Name deines WLAN-Netzwerks";
const char* password =  "Dein WLAN-Passwort";
```

Die Setup-Funktion

In der Funktion **void setup()** verwendest du Code, um deinen ESP32 mit dem Internet zu verbinden:

```
void setup() {

  Serial.begin(115200);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Ich verbinde mich mit dem Netzwerk...");
  }

  Serial.println("Ich bin mit dem Netzwerk verbunden!");
}
```

Hier startest du Verbindung zu deinem Seriellen Monitor und rufst die Funktion **WiFi.begin()** auf, die dich mit Hilfe deiner Zugangsdaten mit deinem WLAN-Netzwerk verbindet. Sobald die Verbindung steht, informiert dich dein Controller hierüber im Seriellen Monitor.

Die News abrufen, verarbeiten und ausgeben

Jetzt wird es spannend! In der Funktion **void loop()** rufst du alle 10 Sekunden eine Funktion auf, die die aktuellen Schlagzeilen von einer API als JSON abruft, sie verarbeitet und in deinem Seriellen Monitor ausgibt:

```
void loop() {
  apiCall();
  delay(10000);
}
```

Du kannst den Inhalt der Funktion **apiCall()** natürlich auch direkt in deinem Loop schreiben. Für mehr Übersichtlichkeit ist sie in diesem Projekt jedoch eigenständig angelegt.

Mit dem Internet verbinden und die JSON-Daten per API Call abrufen

Schauen wir uns diese Funktion also Schritt für Schritt an.

```
void apiCall(){

  if ((WiFi.status() == WL_CONNECTED)) {
```

Du definierst zunächst natürlich die Funktion selbst. Dann startest du mit einer If-Abfrage, um herauszufinden, ob dein ESP32 tatsächlich mit deinem WLAN-Netzwerk verbunden ist. Ist das der Fall, geht es weiter.

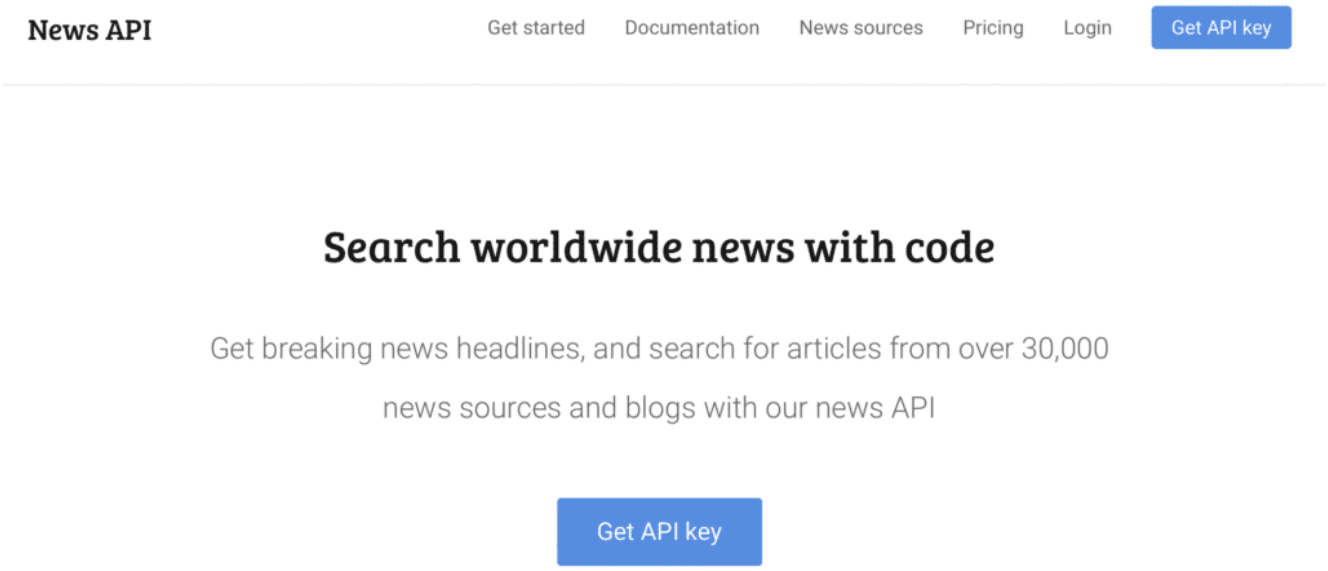
```
HTTPClient http;

http.begin("https://newsapi.org/v2/top-headlines?country=us&apiKey=API_KEY");
```

Zuerst erstellst du eine Instanz von **HTTPClient** mit dem Namen **http**. Diesen Namen verwendest du im folgenden Code als Präfix für alle Funktionen, die du verwendest.

Da wäre als erstes **http.begin()**. Als Parameter gibst du dieser Funktion die URL zur API mit, von der du die News abrufen möchtest. Und an dieser Stelle müssen wir einen kurzen Abstecher machen.

Beim Anbieter News API kannst du dich als Privatperson kostenlos registrieren, um dort aktuelle Schlagzeilen abzurufen. Hinweis: Leider hat News API den Service für internationale Medien eingestellt – es ist nur noch die Abfrage US-amerikanischer Schlagzeilen möglich. Falls du an deutschsprachigen Nachrichten interessiert bist, wirf einen Blick in dieses Raspberry Pi Projekt.



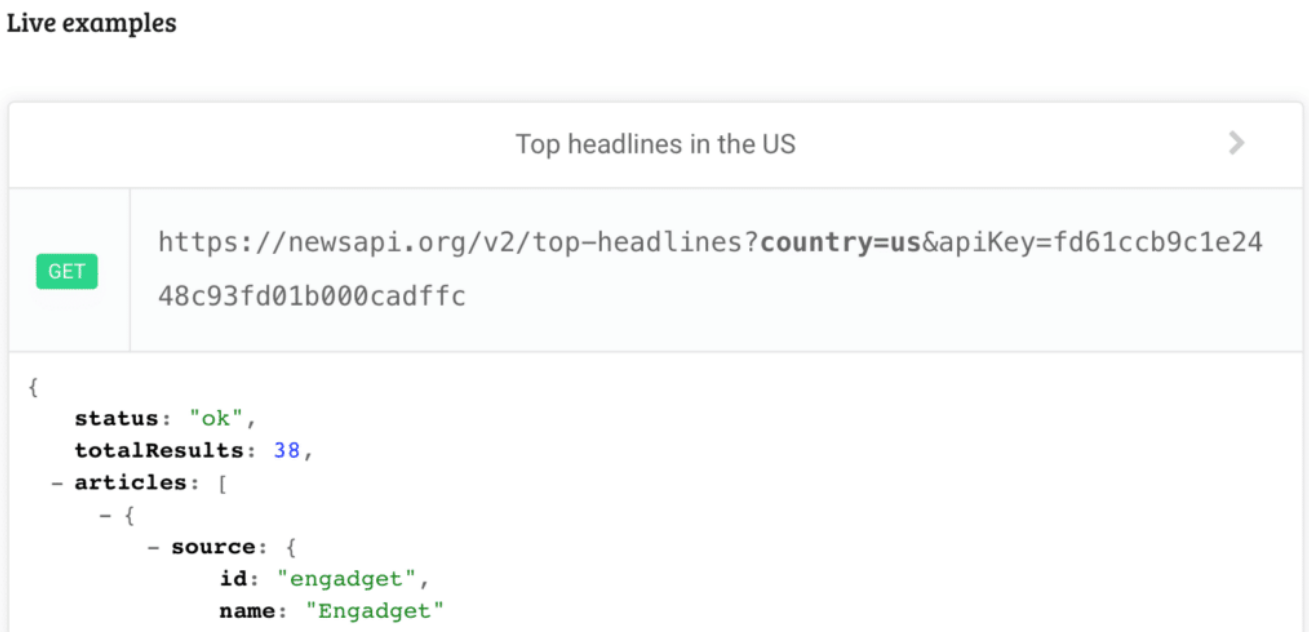
Die Registrierung geht schnell und unkompliziert. **Anschließend erhältst du deinen eigenen API Key, den du benötigst, um die Nachrichten abrufen zu dürfen.**

Wichtig: Auf newsapi.org findest du deinen API Key, indem du auf deine E-Mail-Adresse rechts oben klickst. Diesen Key musst du im Sketch in Zeile 14 an die URL anhängen, indem du **API_KEY** durch deinen Key ersetzt:

https://newsapi.org/v2/top-headlines?country=us&apiKey=API_KEY

Achte darauf, beim Einsetzen deines Keys keine anderen Zeichen zu überschreiben. Ohne API Key oder mit einer fehlerhaften URL funktioniert die Abfrage nicht.

Klicke danach auf den Menüpunkt **Get started** und anschließend im linken Menü auf **Top headlines**. Auf den Kacheln, die du jetzt siehst, findest du verschiedene Beispiele für API-Abfragen:



Neben dem grünen Label **GET** siehst du bereits die fertige URL für die Funktion **http.begin()** in deinem Sketch. Praktisch: Dein API Key ist auch schon integriert.

Klicke dich ein wenig durch das Angebot des Anbieters und probiere verschiedene URLs aus! Im Code oben ist eine URL für die Top-Schlagzeilen aus US-amerikanischen Medien integriert, aber du kannst diese nach Belieben austauschen. So kannst du zum Beispiel reine Business News abrufen oder auch nur Schlagzeilen zu einem bestimmten Suchbegriff.

Aber zurück zum Code: Um den API Call auszuführen, benötigst du zunächst die Funktion **http.GET()**. Diese erhält einen Status Code zurück, den du in einer Variable speicherst. Und weiter geht’s:

```
int httpCode = http.GET();

if (httpCode == 200) {
    Serial.println(httpCode);
```

Wenn du eine gültige Antwort vom News API Server bekommst, lautet der Status Code **200**. Nur dann kann der Sketch weitermachen. Auf Wikipedia erfährst du mehr über Status

Codes.

Bei einer erfolgreichen Abfrage (mit dem Status 200 also) erhältst du JSON-Daten, die die News sowie weitere Metadaten enthalten. Wie diese Daten in ihrer Rohform aussehen, kannst du herausfinden, indem du die URL aus deinem Code in einem Browser aufrufst.

Damit kannst du natürlich erst mal wenig anfangen, denn so die Nachrichten zu lesen, macht keinen Spaß. Also musst du diese JSON-Daten verarbeiten – bzw. **parsen**. Was das alles bedeutet? Auch beim Thema JSON und Parsing hilft dir Wikipedia weiter.

Zunächst speicherst du die Rohdaten in einer Variable des Typs **String**, um sie danach zu verarbeiten:

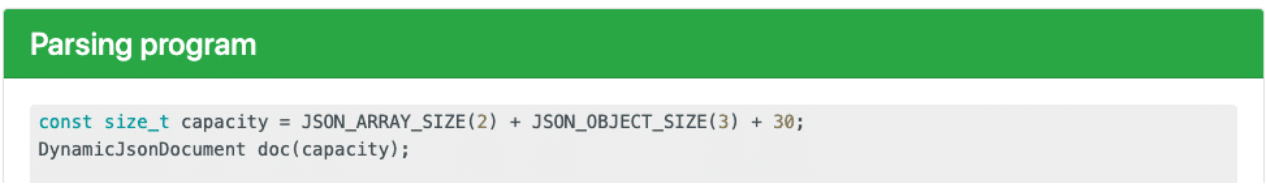
```
String payload = http.getString();
```

Die JSON-Daten parsen

Jetzt wird es ernst – und auch etwas knifflig. Um es nicht zu kompliziert zu machen, sollen vorerst einfache Erklärungen ausreichen. In der Dokumentation der Bibliothek ArduinoJson erfährst du alle Details, wenn du möchtest.

Also in aller Kürze: Damit dein ESP32 die Daten verarbeiten kann, müssen wir erst einmal den hierfür benötigten Speicherbedarf berechnen. Das machst du am einfachsten, indem du den offiziellen Assistenten der Bibliothek verwendest.

Rufe dafür die URL, die du weiter oben im Sketch eingetragen hast, in einem Browser auf. Markiere dann den gesamten Text, kopiere ihn und setze ihn in das linke Feld **Input** ein. Im Feld **Parsing program** darunter findest du anschließend die Zeile Code, die du für deinen Sketch brauchst:



In diesem Projekt ist das diese:

```
const size_t capacity = JSON_ARRAY_SIZE(20) + 20*JSON_OBJECT_SIZE(2) + JSON_OBJECT_SIZE(3) + 20*JSON_OBJECT_SIZE(8) + 19020;
```

Anschließend folgen einige Funktionen und Abfragen, um die JSON-Daten zu verarbeiten und diese Verarbeitung bei einem Fehler zu stoppen:

```
DynamicJsonDocument doc(capacity);

DeserializationError error = deserializeJson(doc, payload);
if (error) {
   Serial.print(F("deserializeJson() hat nicht funktioniert: "));
   Serial.println(error.c_str());
   return;
   }
```

Jetzt bist du fast am Ziel! Im Hintergrund hat dein Sketch bereits die JSON-Daten strukturiert und die verschiedenen Inhalte mehreren Konstanten (const) zugewiesen. Du musst dir nur noch die Rosinen rauspicken.

In diesem Projekt ist das der Titel der neuesten Nachricht. Du wählst zunächst das Array **articles** aus den Rohdaten aus. Hieraus wiederum den ersten, also neuesten Artikel **articles[0]** und von diesem wiederum den Titel **[„title“]**. Zuletzt gibst du diesen **title**, also deine Schlagzeile, im Seriellen Monitor aus:

```
JsonArray articles = doc["articles"];

JsonObject articles_0 = articles[0];

const char* articles_0_title = articles_0["title"];
```

```
Serial.println(articles_0_title);
}
```

Was jetzt noch fehlt, sind ein **else{}** bei einem Status Code, der nicht 200 entspricht – du erinnerst dich an diese Abfrage weiter oben? Und eine Funktion, mit der wir den API-Call wieder beenden:

```
else {
  Serial.println("Fehlerhafter HTTP-Request");
  }

http.end(); //Free the resources
}
}
```

Das war’s! In deinem Seriellen Monitor sollte jetzt alle 10 Sekunden die neueste Schlagzeile aus der von dir per URL definierten Nachrichtenquelle erscheinen. Hier nun der gesamte Sketch:

```
#include <ArduinoJson.h>
#include <WiFi.h>
#include <HTTPClient.h>

// WLAN-Daten
const char* ssid = "WLAN-NETZWERK";
const char* password =  "PASSWORT";

void apiCall(){
  if ((WiFi.status() == WL_CONNECTED)) {

    HTTPClient http;

http.begin("https://newsapi.org/v2/top-headlines?country=us&ap
iKey=API_KEY");
    int httpCode = http.GET();

    if (httpCode == 200) {
```

```
        Serial.println(httpCode);

        String payload = http.getString();
          const size_t capacity = JSON_ARRAY_SIZE(20) +
20*JSON_OBJECT_SIZE(2)   +   JSON_OBJECT_SIZE(3)   +
20*JSON_OBJECT_SIZE(8) + 19020;
        DynamicJsonDocument doc(capacity);

         DeserializationError error = deserializeJson(doc,
payload);
        if (error) {
             Serial.print(F("deserializeJson() hat nicht
funktioniert: "));
           Serial.println(error.c_str());
           return;
           }

        JsonArray articles = doc["articles"];

        JsonObject articles_0 = articles[0];

        const char* articles_0_title = articles_0["title"];

        Serial.println(articles_0_title);
        }

        else {
          Serial.println("Fehlerhafter HTTP-Request");
          }

        http.end();
        }
  }

void setup() {

  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
```

```
    Serial.println("Ich verbinde mich mit dem Netzwerk...");
  }

  Serial.println("Ich bin mit dem Netzwerk verbunden!");
}

void loop() {
  apiCall();
  delay(10000);
}
```

Funktioniert nicht?

Wenn in deinem Seriellen Monitor keine Nachrichten erscheinen, prüfe bitte folgende Fehlerquellen:

- Hast du in den Zeilen 6 und 7 deine WLAN-Daten eingetragen?
- Hast du in der URL in Zeile 14 deinen API Key von newsapi.org eingetragen?
- Sind alle verwenden Bibliotheken auf dem neuesten Stand?

Wie geht es weiter?

Mach deinen neuen Newsticker smart und baue noch einen Geräuschsensor dazu, um immer dann die News abzurufen, wenn du in die Hände klatschst. Lerne hier, wie du einen Geräuschsensor am Arduino (oder ESP32) verwendest.

Natürlich kannst du auch ein Display installieren und die Schlagzeilen darauf anzeigen lassen, damit du nicht auf deinen Seriellen Monitor angewiesen bist. Sicher fallen dir noch viele weitere Optimierungen ein!