Countdown zum nächsten SpaceX Raketenstart

Wichtiger Hinweis: Leider funktioniert dieses Projekt nicht mehr, da die hierfür benötigte API leider eingestellt wurde. Es ist trotzdem weiterhin online, denn vielleicht bringt es dich beim Durchlesen auf eine andere Idee. Falls du eine andere Möglichkeit kennst, den nächsten Start einer Rakete zu ermitteln, schreibe gerne an info@polluxlabs.net

Wenn du an einen Raketenstart denkst, kommt dir bestimmt vieles in den Sinn. Sicherlich auch der obligatorische Countdown. Wie wäre es, wenn du daheim die Sekunden zum nächsten Launch einer Rakete von SpaceX herunterzählst?

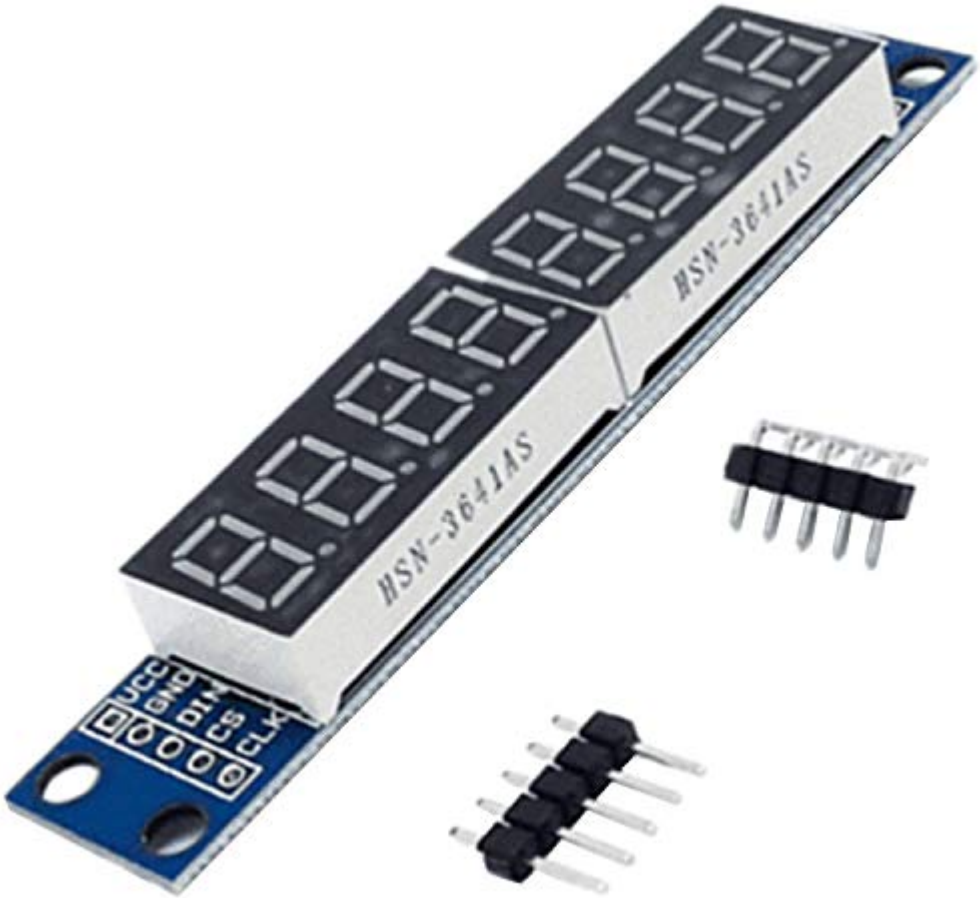
Elon Musks Weltraumunternehmen betreibt eine API, von der du unter anderem die Daten des nächsten Starts beziehen kannst. Hierfür benötigst du nur einen ESP8266. Den Countdown kannst du stilecht auf einem 7-Segment Display darstellen. Und für die Beleuchtung beim Lift-off eignet sich ein NeoPixel LED-Ring.

Für dieses Projekt benötigst du (Mengen s. Beschreibung):

[AZDelivery NodeMCU Amica Modul V2 ESP8266 ESP-12F WiFi - Node MCU ESP 8266 WiFi Development Board mit CP2102 kompatibel mit Arduino - inklusive Installationsanleitung als E-Book]

 Maße (LxBxH): 48 x 26 x 13 mm

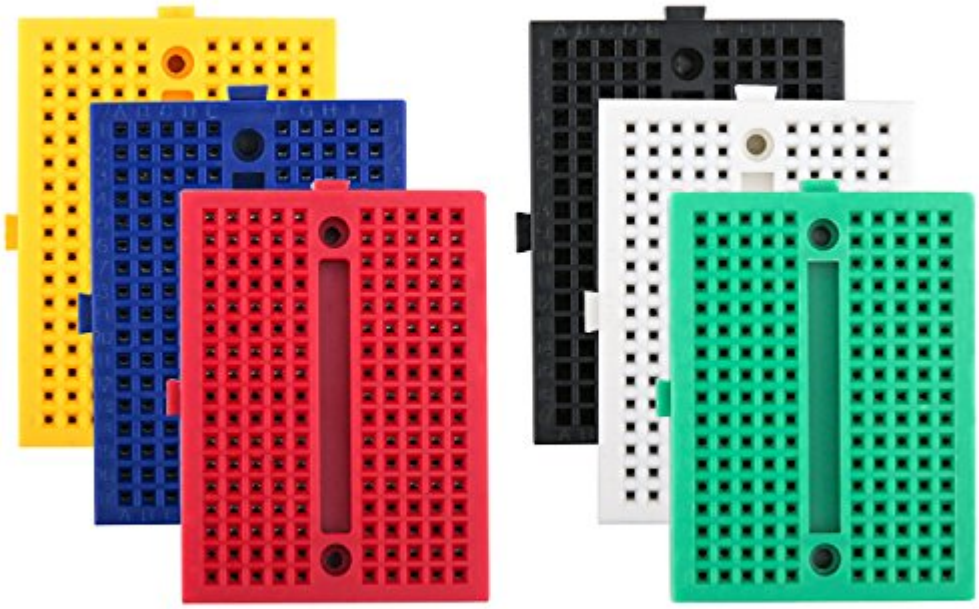
8,49 €



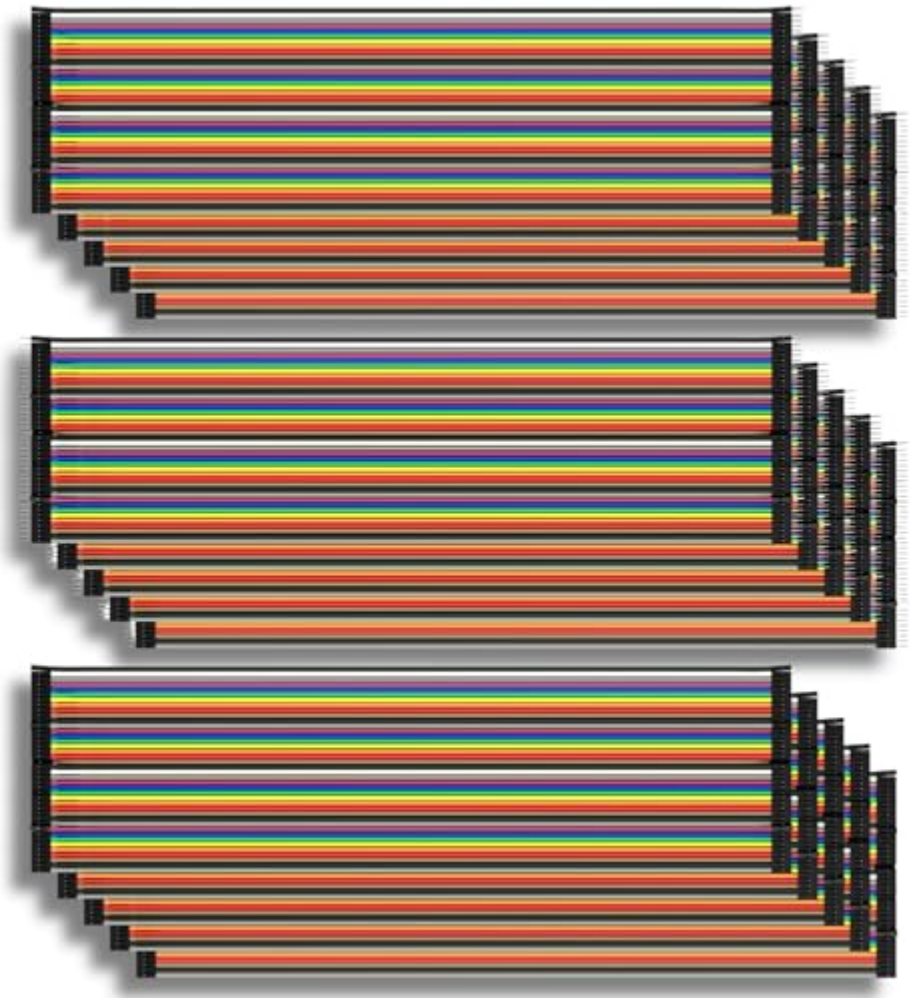
[AZDelivery 1 x MAX7219 LED Modul TM1637 8 Bit 7-Segmentanzeige LED Display kompatibel mit Arduino und Raspberry Pi inklusive E-Book!]

6,99 €

[AZDelivery 5 x 5V RGB LED Ring Kompitabel mit WS2812B 12-Bit 38mm kompatibel mit Arduino inklusive E-Book!]
16,99 €
Angebot



[ELEGOO Steckbrett 6er Set 170 Tie Points Mini Breadboard Kit für Arduino]
Diese Teile sind gut um ganz kleine Schaltungen oder nur einen Chip zu verdraten.; 6...
6,79 €



AZDelivery Jumper Wire Kabel 3 x 40 STK. je 20 cm M2M/ F2M / F2F kompatibel mit Arduino und Raspberry Pi Breadboard inklusive E-Book!
6,99 €

Noch ein Hinweis vorab: Genau auf die Sekunde wirst den Lift-off der Rakete leider nicht treffen. Das liegt weniger an deinem Projekt, sondern daran, dass die meisten Raketen nicht ganz pünktlich sind. Zumindest haben wir bei unseren Tests beobachtet, dass der tatsächliche Start von der Angabe in der API immer um **ein paar Sekunden** abweicht.

Grundlegende Tutorials

In diesem Projekt kommt einiges zum Einsatz, für das du auf Pollux Labs passende Tutorials findest:

- ESP8266 und D1 Mini mit der Arduino IDE programmieren

- Einen ESP8266 oder ESP32 mit dem Internet verbinden
- JSON abrufen & dekodieren mit ArduinoJson und APIs und Daten
- Die aktuelle Uhrzeit mit einem ESP8266 abfragen
- Eine 7-Segment-Anzeige am Arduino anschließen und verwenden
- NeoPixel RGB LED Ring anschließen & verwenden
- Ein einfacher Arduino-Countdown mit Retro-Optik

Schaue zunächst in die oben genannten Tutorials, wenn du ein paar der Themen noch nicht kennst oder erfahren möchtest, wie du ein 7-Segment Display oder den NeoPixel LED-Ring anschließt.

Im Folgenden konzentrieren wir uns auf den Start der nächsten Rakete von SpaceX – bzw. auf die Daten, die du hierfür benötigst.

Die SpaceX API

SpaceX betreibt eine sehr große Datenbank, aus der du dich kostenlos bedienen darfst. Hier findest du beispielsweise Daten zu Raketen, Modulen und Startrampen – aber natürlich auch zu vergangenen und kommenden Raketenstarts. Du findest sogar die aktuelle Position des Tesla Roadsters, der in den Orbit geschossen wurde.

Foto: SpaceX

All diese Daten kannst du über eine API abrufen und weiterverarbeiten – und zwar auch mit deinem ESP8266. Für dieses Projekt benötigst du also die Daten zum nächsten Raketenstart. Diese findest du unter folgender Adresse:

https://api.spacexdata.com/v4/launches/next

Wenn du diese URL in deinem Browser aufrufst, findest eine ganze Menge Daten im JSON-Format. Unter anderem auch den Zeitpunkt des nächsten SpaceX-Starts als Unixzeit. In diesem Fall der 24.3.2021 um 9:58 MEZ.

"date_unix":1616576280

Diese URL verwendest du in deinem Sketch, um mit Hilfe der aktuellen Uhrzeit die Dauer bis zum nächsten Start – und somit den Countdown – zu berechnen.

Die Daten abrufen

Du findest den gesamten Sketch am Ende des Projekts. An dieser Stelle steigen wir direkt beim Abruf der Daten von der SpaceX API ein.

Hierfür hinterlegst du zunächst zu Beginn des Sketchs die URL der API und den Port, der für die sichere Übertragung via HTTPS zuständig ist: 443.

```
const char* host = "https://api.spacexdata.com/v4/launches/next";
const int httpsPort = 443;
```

Als nächstes kommt der Loop. Hier erstellst du zunächst die Instanz **http** von **HTTPClient** und verbindest dich mit der API:

```
HTTPClient http;
client.connect(host, httpsPort);
http.begin(client, host);
```

Danach prüfst du, ob die Verbindung steht und speicherst die JSON-Daten, die du von der API erhältst, in der Variablen **payload**. Anschließend dekodierst du sie und „ziehst“ dir die oben genannte Zeitangabe als Unixtime. Diese Zeitangabe speicherst du in der Variablen **launchTime**.

```
if (http.GET() == HTTP_CODE_OK) {
  String payload = http.getString();
  DynamicJsonDocument doc(3072);
  deserializeJson(doc, payload);

  launchTime = doc["date_unix"]; //Startzeit speichern
```

Als nächstes berechnest du den Countdown. Hierfür benötigst du natürlich auch die aktuelle Uhrzeit. Diese hast du weiter oben

im Sketch schon ermittelt und in der Variablen **currentTime** gespeichert (Schaue hierfür in den vollständigen Sketch).

```
countdown = launchTime - currentTime;
```

Den SpaceX Countdown anzeigen

Jetzt lässt du deinen Countdown laufen, bis die Startzeit erreicht wurde:

```
while (countdown != 0) {
```

Du zeigst zunächst die Sekunden, die es noch bis zum Start dauert, auf dem Display an. Hier kommt eine Funktion ins Spiel, die du im vollständigen Sketch findest:

```
drawDigits(countdown);
```

Danach ziehst du vom Wert in der Variablen **countdown** eine Sekunde ab und wartest mit deinem **delay** auch genau diese eine Sekunde, bevor du die nächste Zahl anzeigst.

```
countdown -= 1;
delay(1000);
```

Wenn der While-Loop ausgelaufen ist, die Variable **countdown** also eine Null enthält, löschst du das Display, startest dafür den LED-Ring und lässt ihn z.B. eine halbe Stunde angeschaltet.

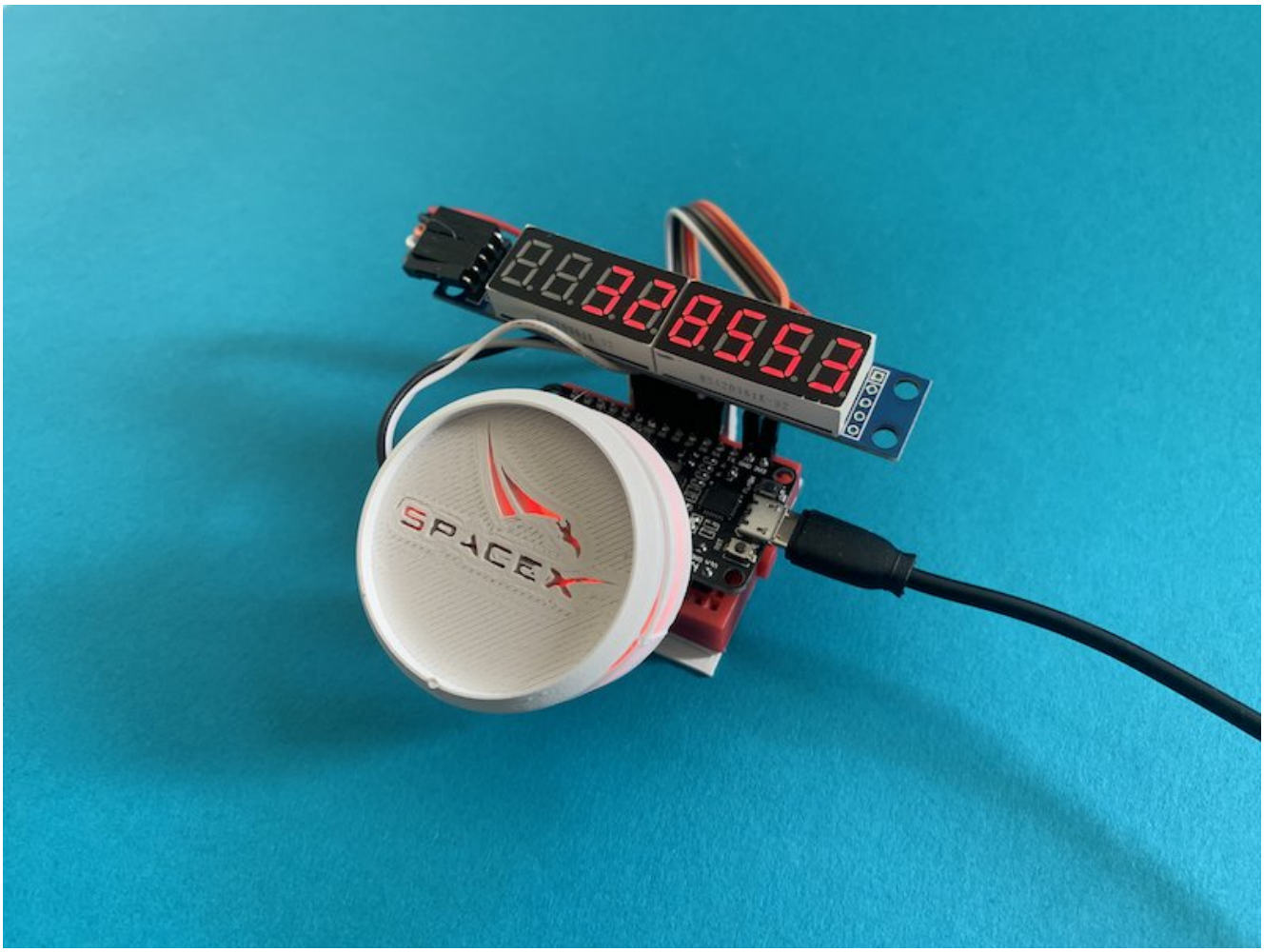
```
lc.clearDisplay(0);

for (int i = 0; i < 12; i++) {
  pixels.setPixelColor(i, pixels.Color(255, 0, 0));
```

```
  pixels.show();
  delay(1);
}

delay(1800000); //30 Minuten
```

In diesem Beispiel leuchten alles LEDs auf dem Ring in einem satten Rot auf. Du kannst dir aber natürlich auch etwas raffinierteres überlegen!



Ein möglicher Aufbau für das Projekt

Und das war es auch schon! **Zuletzt noch ein Hinweis:** Wenn dein Delay am Ende des Sketchs ausgelaufen ist, folgt die nächste Abfrage der SpaceX API. Möglicherweise liegen dann aber noch keine neuen Daten vor. Bis das soweit ist, kann es mitunter

einige Stunden oder auch Tage dauern – du könntest deinen ESP8266 also auch erst einmal gar keine neue Abfrage stellen lassen.

Sketch als .txt anschauen

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>

WiFiClientSecure client;

//NTP
#include <NTPClient.h>
#include <WiFiUdp.h>
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

//SpaceX API URL und Port
const char* host = "https://api.spacexdata.com/v4/launches/next";
const int httpsPort = 443;

//WLAN Zugangsdaten
const char* ssid = "NETWORK";
const char* password =  "PASSWORD";

//Variablen
long launchTime;
long currentTime;
long countdown;

//7-Segment-Display
#include <LedControl.h>
LedControl lc = LedControl(12, 15, 13, 1); //12=D6=DIN, 15=D8=CLK, 13=D7=CS
const int NUM_DIGITS = 8;

//NeoPixel LED-Ring
```

```
#include <Adafruit_NeoPixel.h>

int leds = 12; //Anzahl der LEDs
int ledPin = 14; //14=D5, Pin, an dem der NeoPixel
angeschlossen ist

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(leds, ledPin,
NEO_GRB + NEO_KHZ800);

void getCurrentTime() {

  timeClient.update();
  currentTime = timeClient.getEpochTime();
  Serial.print("Current Time= ");
  Serial.println(currentTime);
}

void drawDigits(int num) {
  for (int i = 0; i < NUM_DIGITS ; i++) {
    lc.setDigit(0, i, num % 10, false);
    num /= 10;
    if (num == 0)
      return;
  }
}

void setup() {
  Serial.begin(115200);
  client.setInsecure();

  //Einstellungen des 7-Segment-Displays
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);

  //Einstellungen des NeoPixels
  pinMode (ledPin, OUTPUT);
  pixels.begin();
  pixels.setBrightness(255); //Helligkeit: 0 (aus) - 255

  for (int i = 0; i < 12; i++) {
```

```
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
    pixels.show();
    delay(1);
  }

  WiFi.begin(ssid, password); //Mit dem WLAN verbinden

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  delay(1000);
  Serial.println("Hello, world!");
}

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

    getCurrentTime();

    HTTPClient http;
    client.connect(host, httpsPort);
    http.begin(client, host);

    if (http.GET() == HTTP_CODE_OK) {
      String payload = http.getString();
      DynamicJsonDocument doc(3072);
      deserializeJson(doc, payload);

      launchTime = doc["date_unix"]; //Startzeit speichern
      Serial.print("Launch Time: ");
      Serial.println(launchTime);

       countdown = launchTime - currentTime; //Countdown =
Startzeit - aktuelle Zeit
    } else {
      Serial.print("Error on HTTP request (SpaceX): ");
      Serial.println(http.GET()); //Error Code
    }
  }
```

```
   while (countdown != 0) { //Solange die Startzeit nicht
erreicht wurde
    Serial.print("Time Until Launch: ");
    Serial.println(countdown);
    drawDigits(countdown);
    countdown -= 1;
    delay(1000);
  }

  Serial.println("LAUNCH!"); // Startzeit = aktuelle Zeit
  lc.clearDisplay(0);

  for (int i = 0; i < 12; i++) {
    pixels.setPixelColor(i, pixels.Color(255, 255, 0));
    pixels.show();
    delay(1);
  }

  delay(1800000);
}
```