OpenAI auf dem ESP32 verwenden

Hier bei Pollux Labs gibt es mittlerweile eine Vielzahl von Projekten, die OpenAI bzw. ChatGPT verwenden. Dabei kommt oft der Raspberry Pi und immer die Sprache Python zum Einsatz. Aber es geht auch anders: Wenn du in deinem nächsten Projekt ChatGPT oder auch DALL-E verwenden möchtest, kannst du hierfür auch OpenAI auf dem ESP32 verwenden.

In diesem Tutorial lernst du, wie du deinen ESP32 mit der API von OpenAI kommunizieren lässt, um dir Fragen beantworten und Bilder erstellen zu lassen. Du kommunizierst hierbei über den Seriellen Monitor – deine Fragen bzw. Anweisungen sendest du dabei als Nachricht an deinen ESP32 und erhältst die Antwort wie gewohnt im Textfeld des Monitors.

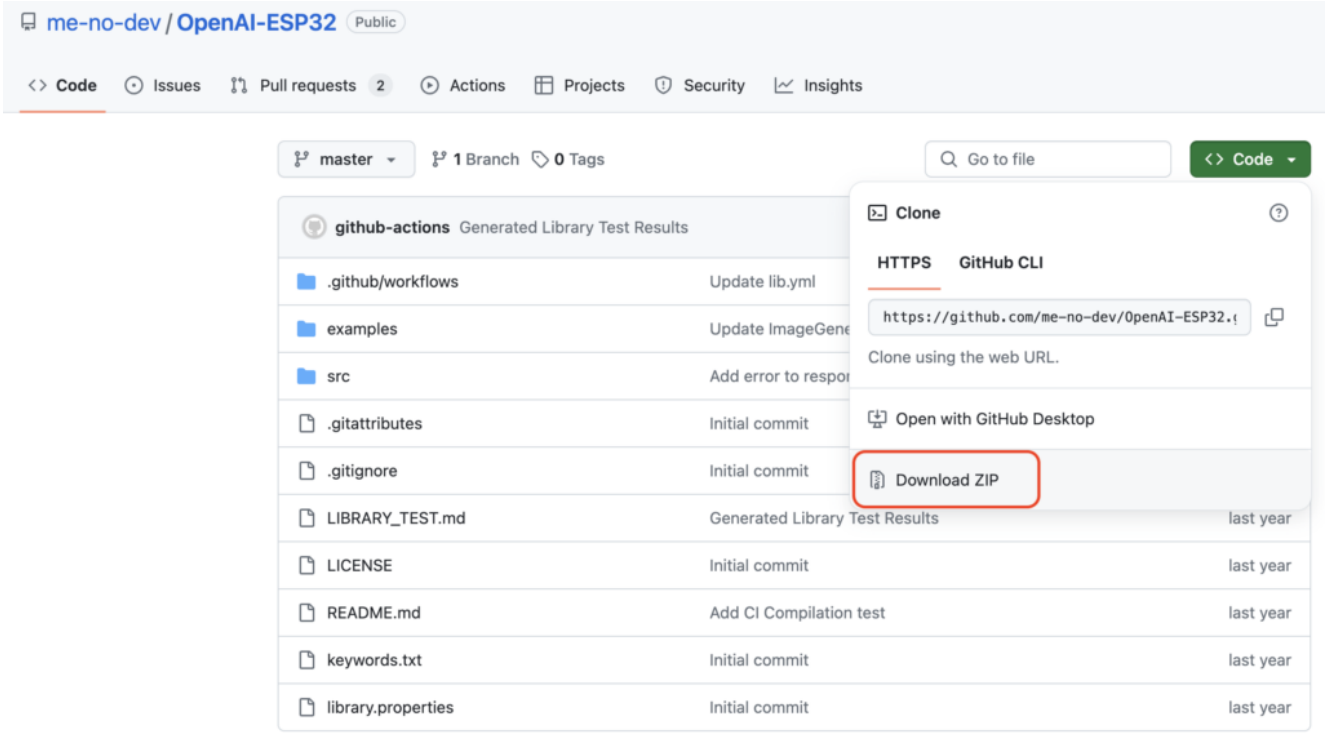
Für die folgende Anleitung benötigst du einen Account bei OpenAI und einen API-Key. Wie du beides erstellst, erfährst du zu Beginn dieses Tutorials.

Die Bibliothek, um OpenAI zu nutzen

Wie auch für Python gibt es für C++ eine Bibliothek, die du verwenden kannst, um ganz einfach auf die API von OpenAI zugreifen zu können.

Diese findest du aktuell jedoch nicht im Bibliotheksverwalter der Arduino IDE, sondern musst sie manuell herunterladen und in deinem Sketch verfügbar machen. Du findest sie auf der

GitHub-Seite des Entwicklers **Me No Dev**. Klicke dort oben auf den grünen Button **Code** und anschließend auf **Download ZIP**.



In deiner Arduino IDE kannst du diese Bibliothek dann ganz einfach über das Menü hinzufügen. Klicke hierfür auf **Sketch -> Bibliothek einbinden -> ZIP Bibliothek hinzufügen** und wähle dann die Datei, die du von GitHub heruntergeladen hast.

so verwendest du ChatGPT auf dem ESP32

Über den Menüpunkt **Datei -> Beispiele -> OpenAI-ESP32** kannst du die Beispiel-Sketches finden, die in der Bibliothek enthalten sind. Wähle den Sketch **ChatCompletion**, der sich nun in einem neuen Fenster öffnen sollte. Trage hier zunächst die Zugangsdaten zu deinem WLAN-Netzwerk sowie deinen API-Key von OpenAI ein:

```
const char* ssid = "your-SSID";
const char* password = "your-PASSWORD";
const char* api_key = "your-OPENAI_API_KEY";
```

Etwas weiter unten im Sketch findest du folgende Zeilen:

```
chat.setModel("gpt-3.5-turbo");     //Model to use for completion. Default is gpt-3.5-turbo
chat.setSystem("Code geek");        //Description of the required assistant
chat.setMaxTokens(1000);            //The maximum number of tokens to generate in the completion.
```

In der ersten Zeile kannst du das gewünschte Sprachmodell festlegen. Voreingestellt ist GPT-3.5-turbo, mit **gpt-4** kannst du jedoch auch das aktuellere Modell verwenden. Darunter kannst du die Rolle von ChatGPT einstellen. Im Beispiel ist das zunächst ein **Code geek**, aber wie wäre es z.B. mit einem Gärtner oder einem Designer? Deine Einstellung verhindert übrigens keine Antworten, die in einen anderen Wissenbereich fallen. Auch ein **Code geek** kann dir sagen, warum der Himmel blau ist.

In der nächsten Zeile kannst du einstellen, wie viele Token höchstens für die Antwort verwendet werden dürfen. Damit stellst du sicher, dass die Antworten von ChatGPT nicht zu kostspielig werden. Bei OpenAI findest du die aktuelle Preisliste. Aktuell (Mai 2024) liegst du mit 1000 Token und GPT-4 bei ungefähr 6 Cent.

In den Zeilen darunter findest du weitere Einstellungsmöglichkeiten, die du jedoch hier einmal außer Acht lassen kannst. Sobald du alles eingetragen hast, lade den Sketch auf deinen ESP32 hoch und öffne nach dem erfolgreichen Upload den Seriellen Monitor und stelle, falls nötig, eine Baudrate von 115200 ein.

Nachdem sich dein ESP32 mit deinem WLAN verbunden hat, siehst du die folgende Eingabeaufforderung:

```
09:57:48.525 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
09:57:48.525 -> configsip: 0, SPIWP:0xee
09:57:48.525 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
09:57:48.525 -> mode:DIO, clock div:1
09:57:48.525 -> load:0x3fff0030,len:1184
09:57:48.558 -> load:0x40078000,len:13260
09:57:48.558 -> load:0x40080400,len:3028
09:57:48.558 -> entry 0x400805e4
09:57:49.019 -> Connecting.....................................
09:57:52.736 -> You can now send chat message to OpenAI by typing in the Arduino IDE Serial Monitor.
09:57:52.736 -> Each line will be interpreted as one message and processed.
09:57:52.736 -> You can restart the conversation by typing "clear"
09:57:52.736 ->
```

Hier siehst du nun also ChatGPT, das auf deine Frage oder Anweisung wartet. Um mit ChatGPT zu kommunizieren, trage oben im Feld **Nachricht** deine Frage ein und sende sie mit **Enter** ab. Testweise habe ich gefragt, warum der Himmel eigentlich blau ist. Und hier die Antwort:

```
10:02:11.758 -> Warum ist der Himmel blau?
10:02:11.758 -> Processing...
10:02:16.011 -> Received message. Tokens: 175
10:02:16.011 -> Der Himmel erscheint blau, weil die Atmosphäre der Erde das Sonnenlicht in alle Richtungen streut. Blaues Licht hat eine kürzere
```

Deine Frage wird also an die API von OpenAI übermittelt, von ChatGPT beantwortet und die Antwort in deinem Seriellen Monitor ausgegeben. Über ihr findest du die Anzahl der Tokens, die die Antwort verbraucht hat.

Falls du die Antwort nicht nur im Seriellen Monitor ausgeben, sondern zum Beispiel auf einem Display anzeigen möchtest – du findest sie in der Variablen **response**, die an dieser Stelle im Code befüllt wird:

```
String response = result.getAt(i);
```

Bilder mit DALL-E auf dem ESP32

erzeugen

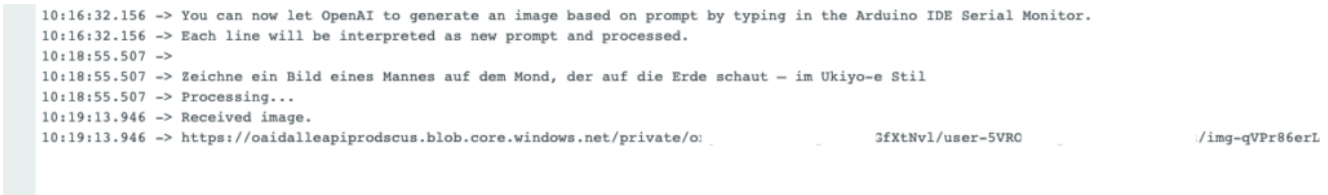
Neben dem Chat kannst du über die API von OpenAI auch DALL-E verwenden, um damit Bilder zeichnen zu lassen. Ein Tutorial, wie du mit Python und DALL-E Bilder erzeugst, findest du übrigens auch bei Pollux Labs.

Auch hierfür verwendest du einen Sketch, der schon als Beispiel mitgeliefert wird. Öffne hierfür wieder **Datei -> Beispiele -> OpenAI-ESP32** und wähle dann **ImageGeneration**. Trage als erstes wieder oben im Sketch deine Zugangsdaten ein. Eine weitere wichtige Stelle befindest sich etwas weiter unten:

```
imageGeneration.setSize(OPENAI_IMAGE_SIZE_256x256);
```

Dort kannst du die Größe des Bilds angeben. Voreingestellt sind 256×256 Pixel, du kannst jedoch auch 512×512 oder 1024×2024 wählen. Im Folgenden habe ich die größte Option gewählt. Laden nun diesen Sketch auf deinen ESP32 und öffne nach dem Upload (und einem evtl. notwendigen Reset des Microcontrollers) wieder deinen Seriellen Monitor. Du erhältst wieder die Aufforderung, einen Prompt zu senden.

Ich habe es einmal mit **Zeichne ein Bild eines Mannes auf dem Mond, der auf die Erde schaut – im Ukiyo-e Stil** versucht. Als Ergebnis erhältst du ein URL zurück, in der letzten Zeile:



Kopiere die gesamte URL aus dem Seriellen Monitor heraus und öffne sie in deinem Browser. Hier mein Ergebnis:

Das Ergebnis ist vielleicht nicht ganz die große Welle vor Kanagawa, aber okay, es war ja nur ein Versuch. Wie du in diesem Tutorial gesehen hast, ist die API von OpenAI also nicht nur Python und leistungsstärkeren Computern vorbehalten – auch mit einem ESP32 kannst du hier schon einiges erreichen.