ESP32 et ChatGPT

vers un système d'autoprogrammation...

Figure 1. Les cartes Arduino Nano ESP32, avec les environnements Arduino (à gauche) et MicroPython (à droite).

Saad Imtiaz (Elektor Lab)

Nous avons combiné la puissance de deux microcontrôleurs Arduino Nano ESP32, la communication sans fil et l'API ChatGPT pour créer un système de communication intelligent et interactif. L'une des cartes est connectée à ChatGPT ; les réponses et le code à renvoyer de cet outil d'intelligence artificielle populaire seront transférés sans fil à l'autre carte Nano. Ce système à deux cartes peut-il se programmer lui-même avec ChatGPT ? Suivez les instructions pas à pas et découvrez comment les cartes communiquent et comment fonctionne l'API ChatGPT.

Dans le domaine des systèmes embarqués et de l'Internet des objets (IoT), le microcontrôleur ESP32 d'Espressif a gagné en popularité grâce à sa polyvalence et à ses capacités robustes. Grâce à sa connectivité Wifi intégrée et à ses puissantes capacités de traitement, il ouvre un éventail de possibilités pour la construction d'appareils intelligents et connectés. Dans cet article, nous examinons un projet de démonstration qui utilise l'API ChatGPT, un modèle de langage d'IA créé par OpenAI, pour atteindre de nouveaux sommets avec deux cartes Arduino Nano ESP32.

L'objectif de ce projet est de créer un système de communication transparent entre les cartes Nano ESP32, l'une fonctionnant avec l'EDI Arduino et l'autre avec l'environnement MicroPython. En exploitant l'API ChatGPT, nous permettons à ces cartes de s'engager dans des conversations engageantes et interactives. Imaginez les possibilités d'envoyer à vos microcontrôleurs des extraits de code ou de recevoir des solutions créatives à vos questions.

Tout au long de cet article, nous aborderons les aspects techniques de la configuration du matériel et du logiciel et de l'établissement de la communication entre les cartes Nano ESP32. Nous explorerons les avantages et les limites de chaque *framework* (EDI Arduino et MicroPython) en mettant en lumière leurs caractéristiques uniques et leurs cas d'utilisation. En outre, nous fournirons des extraits de code pratiques, des explications et des idées pour vous aider à comprendre le fonctionnement interne de ce projet.

À la fin de cet article, vous comprendrez clairement comment construire votre propre système de communication basé sur le Nano ESP32, en exploitant la puissance de l'IA et de l'IdO. Alors, plongeons dans cette aventure passionnante et commençons à créer un environnement intelligent et interactif avec le Nano ESP32 et ChatGPT !

Matériel

Pour commencer ce projet, nous aurons besoin de quelques composants essentiels. Examinons de plus près le matériel exigé :

1. **Modules Arduino Nano ESP32 (×2)**
   Le microcontrôleur ESP32 est au cœur de notre projet. Pour la communication, nous aurons besoin de deux cartes Arduino Nano ESP32. Ces cartes sont très répandues et offrent de nombreuses fonctionnalités telles que la connectivité Wifi, une grande puissance de traitement et des broches GPIO pour l'interfaçage avec des composants externes.
2. **Câbles USB Type-C (×2)**
   Les câbles USB Type-C sont nécessaires pour alimenter et programmer les cartes Nano ESP32. Ces câbles nous permettent d'établir une connexion entre les cartes et notre ordinateur, facilitant ainsi la programmation et le transfert de données.
3. **Réseau Wifi**
   Un réseau Wifi stable est essentiel pour établir la communication entre les deux cartes et l'API ChatGPT. Assurez-vous d'avoir accès à un réseau Wifi fiable avec une connectivité Internet.

Cela permettra à la première carte Nano ESP32 (appelée ici « Nano ESP32-1 ») de se connecter à l'API ChatGPT et d'échanger des messages.

Maintenant que nous connaissons les composants nécessaires, passons aux aspects logiciels et de programmation du projet.

Logiciel et programmation

Pour mettre en place le système de communication et programmer les cartes, nous aurons besoin des outils logiciels suivants :

1. **EDI Arduino**
   L'environnement de développement intégré (EDI) Arduino est le plus largement utilisé pour la programmation des microcontrôleurs basés sur le firmware Arduino. Désormais, avec sa nouvelle version 2.0, il offre une interface plus conviviale, un langage de programmation simplifié et une bibliothèque étendue de fonctions préconstruites qui facilitent le travail avec les cartes Nano ESP32.
2. **Firmware MicroPython**
   MicroPython est une version légère du langage de programmation Python optimisée pour les microcontrôleurs. Il offre une expérience de programmation plus flexible et interactive par rapport à l'EDI Arduino. Pour programmer la deuxième Nano ESP32 (ESP32-2) en MicroPython, nous devrons installer le micrologiciel MicroPython à bord. Dans la **figure 1,** vous pouvez voir les deux cartes connectées. La carte de gauche fonctionne avec le *framework* Arduino et celle de droite fonctionne avec MicroPython.
3. **Accès à l'API ChatGPT**
   Pour connecter la carte Nano ESP32-1 à l'API ChatGPT, vous devez avoir accès à l'API. Pour créer votre clé API, vous devrez vous rendre sur [1], créer un compte Open AI, puis vous rendre sur [2] et cliquer sur *Create New Secret Key* ; vous pouvez également le faire en cliquant sur l'icône de profil dans le coin supérieur droit de la page Web et en sélectionnant *View API Keys* dans le menu déroulant. Une fois que vous avez créé la clé API, sauvegardez-la dans un endroit sûr car vous ne pourrez pas la recopier.
4. **Programmation**
   Nous allons commencer avec le Nano ESP32-1 fonctionnant avec l'EDI Arduino. Tout d'abord, nous ajoutons les bibliothèques appropriées à notre code. La bibliothèque *WiFi.h* est nécessaire pour que nous puissions ajouter la fonction de connexion du Nano ESP32 au réseau Wifi. Ensuite, *HTTPClient.h* est utilisé pour envoyer nos données, c'est-à-dire la réponse de ChatGPT à la seconde carte Nano ESP32. Enfin, nous utilisons *ArduinoJson.h* (la bibliothèque permettant d'utiliser JSON sur pratiquement toutes les cartes existantes). Elle nous permet la sérialisation et la désérialisation JSON, que nous utiliserons pour accéder à l'interface API ChatGPT, envoyer des requêtes et recevoir des réponses. Les codes des deux cartes sont également disponibles sur GitHub [3].

```
// Load Wi-Fi library
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

// Replace with your network credentials
const char* ssid = "WIFI_SSID";
const char* password = "WIFI_PWD";
//chatgpt api key
const char* apiKey =
  "sk-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
```

Nous allons maintenant ajouter l'adresse IP de la deuxième carte Nano ESP32 (cette adresse IP sera affichée dans le moniteur série de l'EDI, lorsque la deuxième carte Nano ESP32 sera connectée au réseau Wifi).

```
const char* serverIP = "192.168.1.82";
  // IP address of the second Nano ESP32
const uint16_t serverPort = 80;
  // Port number on the second Nano ESP32
```

Une fonction a été créée spécifiquement pour se connecter à l'API ChatGPT et communiquer avec elle, voir **listage 1**.
Passons maintenant au code de la deuxième carte Nano ESP32. Avant de commencer la programmation, expliquons comment exécuter MicroPython sur le Nano ESP32 : nous utiliserons *The Arduino Lab*

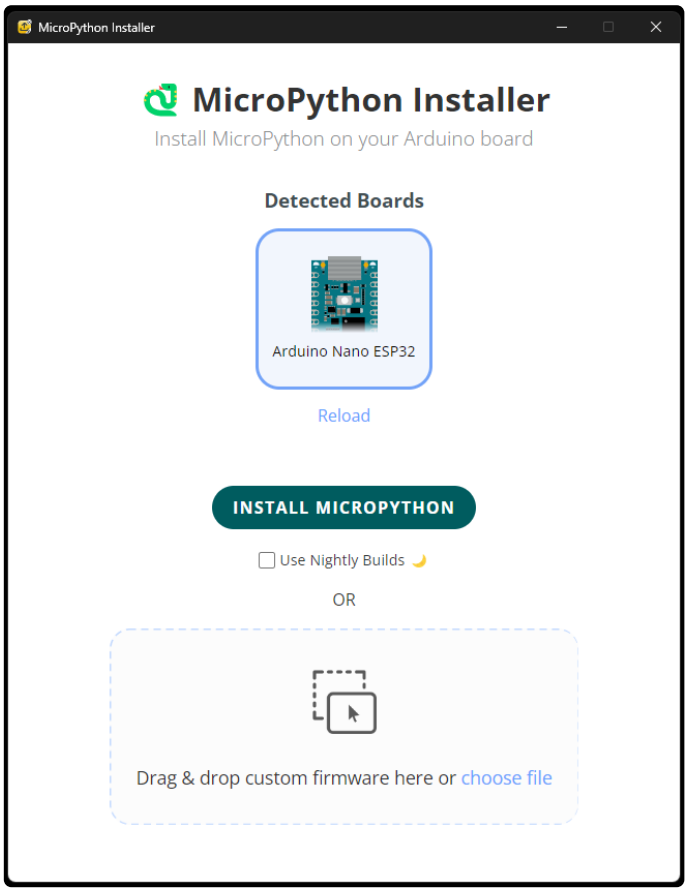


Figure 2. Outil MicroPython Firmware par Arduino Labs.

Listage1. Code Arduino pour envoyer des requêtes à ChatGPT.

```
String sendChatGPTRequest(String prompt) {
 String received_response= "";
 String apiUrl = "https://api.openai.com/v1/completions";
 String payload = "{\"prompt\":\"" + prompt +
                  "\",\"max_tokens\":100, \"model\": \"text-davinci-003\"}";
 HTTPClient http;
 http.begin(apiUrl);
 http.addHeader("Content-Type", "application/json");
 http.addHeader("Authorization", "Bearer " + String(apiKey));

 int httpResponseCode = http.POST(payload);
 if (httpResponseCode == 200) {
 String response = http.getString();

 // Parse JSON response
 DynamicJsonDocument jsonDoc(1024);
 deserializeJson(jsonDoc, response);
 String outputText = jsonDoc["choices"][0]["text"];
 received_response = outputText;
 //Serial.println(outputText);
 } else {
 Serial.printf("Error %i \n", httpResponseCode);
 }
 return received_response;
}
```

pour MicroPython. Pour commencer, nous allons d'abord nous rendre sur [4] et télécharger l'outil *Arduino MicroPython firmware* pour flasher l'Arduino Nano ESP32, comme le montre la **figure 2** ci-dessous.
Connectez votre Arduino Nano ESP32 à votre ordinateur, une fois la carte détectée, cliquez sur *Install Micropython* et après quelques secondes, votre carte sera flashée avec le dernier firmware, et vous serez capable d'utiliser MicroPython sur le Nano ESP32.
Nous allons maintenant télécharger *Arduino Labs for MicroPython IDE* depuis son site officiel [5] et télécharger la dernière version. Après avoir extrait le fichier, lancez le logiciel en exécutant le fichier *Arduino Labs for Micropython.exe*.
Sur la deuxième carte Nano ESP32, le code est court et simple ; nous allons d'abord importer les bibliothèques. Nous utiliserons les bibliothèques *network* et *socket* pour nous connecter au réseau Wifi et pour créer notre port serveur où nous pourrons recevoir le code ou les réponses du Nano ESP32-1. De plus, les bibliothèques *machine* et *time* sont nécessaires pour utiliser les GPIO et la fonction *timer* du Nano ESP32.

```
import network
import socket
import machine
import time

# Wi-Fi credentials
wifi_ssid = "WIFI_SSID"
wifi_password = "WIFI_PWD"

# Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Wait until connected to Wi-Fi
while not wifi.isconnected():
 pass
# Print the Wi-Fi connection details
print("Connected to Wi-Fi")
print("IP Address:", wifi.ifconfig()[0])
```

Ensuite, un serveur socket est créé pour recevoir les données du Nano ESP32-1, puis le microcontrôleur entre dans une boucle `while` où il attend de recevoir d'éventuelles réponses ; lorsqu'un code ou une réponse est reçu, le Nano ESP32 exécute le code puis recommence à attendre d'autres instructions (voir **listage 2**).
Dans l'encadré **EDI Arduino, MicroPython et plus encore**, nous discutons des avantages et des inconvénients des deux environnements et comprenons pourquoi l'un peut être plus approprié que l'autre selon le scénario du cas d'utilisation.

Communication entre les cartes Nano ESP32

Entrons maintenant dans les détails techniques de la communication entre les deux cartes Nano ESP3 via Wifi dans le cadre de ce projet. Cette section vise à fournir une explication plus détaillée du processus de communication pour une meilleure compréhension.

Nano ESP32-1 (EDI Arduino)

Le Nano ESP32-1, qui fonctionne avec l'EDI Arduino, établit une connexion Wifi pour communiquer avec des services et des appareils externes. Il se connecte à un réseau Wifi spécifique avec des informations d'identification fournies. Une fois connecté, le Nano ESP32-1 attend une entrée de l'utilisateur sur le moniteur série de l'EDI Arduino.

Lorsque l'utilisateur tape GPT et appuie sur Entrée, le Nano ESP32-1 l'invite à saisir un message. Le message de l'utilisateur est ensuite envoyé par le Nano ESP32-1 à l'API ChatGPT pour traitement. Cette transmission s'effectue via la connexion Wifi établie, ce qui permet au Nano ESP32-1 de communiquer avec l'API externe. Dans la **figure 3**, vous pouvez voir l'ensemble de la communication et de la boucle de connexion entre les deux cartes et l'API ChatGPT.

L'API ChatGPT, une interface avec le modèle de langage d'IA, reçoit le message du Nano ESP32-1. À l'aide de techniques avancées de traitement du langage naturel et d'algorithmes d'apprentissage automatique, l'API analyse le message pour en comprendre le contexte et générer une réponse intelligente ou un extrait de code. Le modèle ChatGPT au sein de l'API exploite ses vastes données d'apprentissage et ses capacités de compréhension du langage pour fournir une réponse pertinente et captivante. Nano ESP32-1 reçoit l'extrait de code généré par l'API ChatGPT et le stocke en mémoire. Cet extrait de code représente la réponse générée par l'IA à la requête de l'utilisateur. Nano ESP32-1 affiche ensuite l'extrait de code reçu sur le moniteur série, ce qui permet aux utilisateurs de revoir les instructions générées par l'IA. Dans la **figure 4**, vous pouvez voir la sortie du moniteur série des deux cartes, où le Nano ESP32-1 envoie le code au Nano ESP32-2 après avoir reçu la réponse de ChatGPT.

Nano ESP32-2 (Framework MicroPython)

D'autre part, le Nano ESP32-2 fonctionne avec le *framework* MicroPython. Tout comme le Nano ESP32-1, le Nano ESP32-2 établit une connexion Wif sur le même réseau avec des informations d'identification fournies. Cela lui permet de recevoir des instructions sans fil de la part de Nano ESP32-1.

Nano ESP32-2 établit une connexion socket et écoute sur un port spécifique, typiquement le port 80. Un socket est un point d'extrémité logiciel qui permet la communication entre deux appareils sur un réseau. En écoutant sur un port spécifique, le Nano ESP32-2 est prêt à recevoir des données entrantes du Nano ESP32-1.

Lorsque le Nano ESP32-1 souhaite envoyer l'extrait de code généré, il établit une connexion avec le Nano ESP32-2 via la connexion socket. L'extrait de code est alors transmis au Nano ESP32-2, où il est reçu et traité.

Nano ESP32-2 traite l'extrait de code reçu et en extrait les instructions. Ces instructions définissent des tâches spécifiques à exécuter par le Nano ESP32-2. Par exemple, l'extrait de code peut demander au Nano ESP32-2 de faire

Listage 2. Recevoir et exécuter du code sur la deuxième carte (Python).

```
# Create a socket server
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('', 80))
server.listen(1)

# Accept and handle incoming connections
while True:
 print("Waiting for connection...")
 client, addr = server.accept()
 print("Client connected:", addr)

 # Receive the code snippet from the first ESP32
 code = ""
 while True:
 data = client.recv(1024)
 if not data:
 break
 code += data.decode()

 # Execute the received code
 try:
 exec(code)
 response = "Code executed successfully"
 print (response)
 except Exception as e:
 response = "Error executing code: " + str(e)

 # Send the response back to the first ESP32
 client.sendall(response.encode())
 # Close the connection
 client.close()
 print("Client disconnected")
```

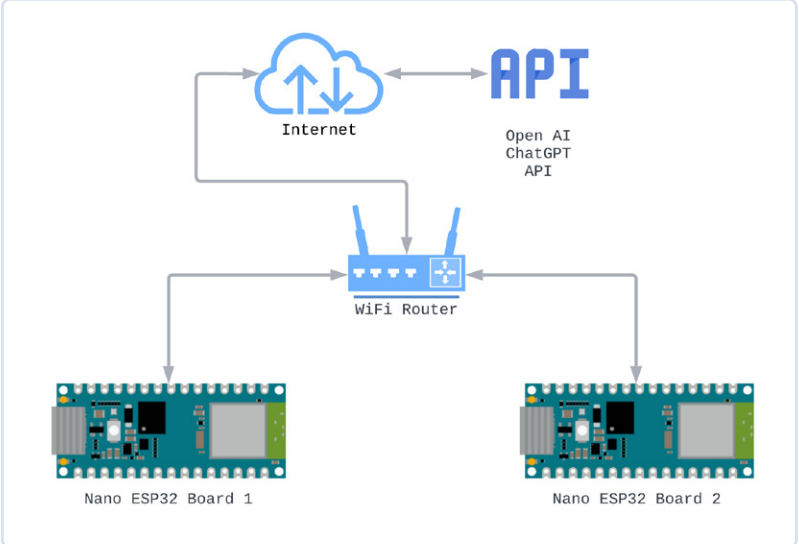


Figure 3. Communication entre les deux cartes Arduino Nano ESP32 et l'API ChatGPT.

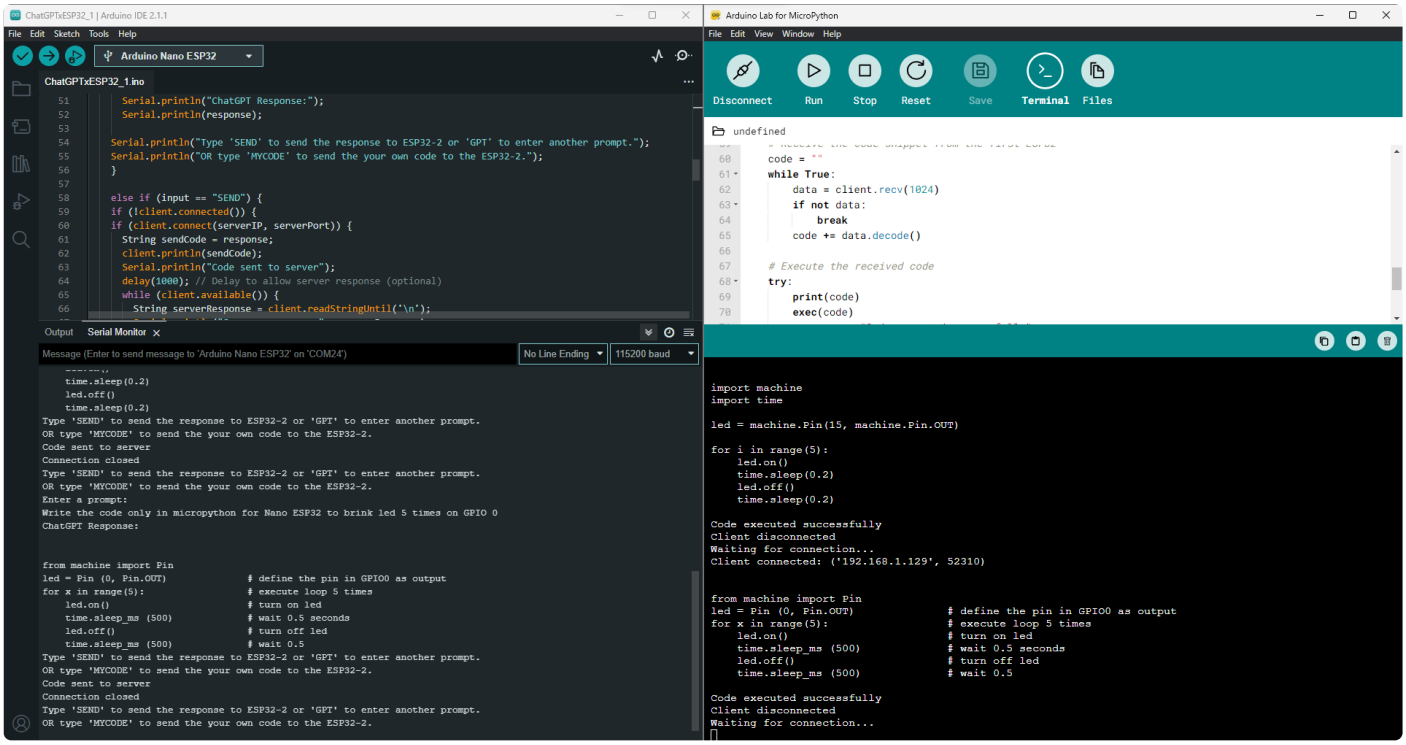


Figure 4. Communication entre les deux Nano ESP32, EDI Arduino (à gauche) et Arduino Lab pour MicroPython (à droite).

clignoter une LED pendant une certaine durée ou d'effectuer d'autres actions en fonction de la réponse générée par l'IA.

Après avoir exécuté les instructions, le Nano ESP32-2 envoie une réponse au Nano ESP32-1 via la connexion socket établie. Cette réponse sert à confirmer que l'extrait de code reçu a été exécuté avec succès. Elle garantit que le Nano ESP32-1 est au courant de l'achèvement de la tâche demandée.

En suivant ce processus de communication, les deux cartes Nano ESP32 peuvent échanger des messages et communiquer efficacement. Le Nano ESP32-1 interagit avec l'API ChatGPT, en tirant parti de ses capacités d'IA, tandis que le Nano ESP32-2 agit en tant que destinataire et exécuteur des instructions générées par l'IA.

Limites de la programmation avec ChatGPT

Bien que ChatGPT soit un modèle de langage d'IA remarquable, il présente certaines limites lorsqu'il s'agit de générer du code fonctionnel, en particulier dans des scénarios de programmation plus complexes. Lors des tests, on a constaté que ChatGPT n'était en mesure de fournir un code fonctionnel pour un simple croquis de clignotement que dans 60 % des cas. Cela indique que la capacité du modèle à générer des extraits de code précis et fonctionnels n'est pas toujours garantie.

Un problème se pose lorsque l'on demande des extraits de code à ChatGPT. Parfois, la réponse inclut du texte explicatif avant et après le code, ce qui peut poser des problèmes lorsque l'on tente d'envoyer l'intégralité de la réponse à la seconde carte Nano ESP32. Ce texte peut manquer de formatage ou de syntaxe, ce qui le rend incompatible lors d'une exécution directe. Dans la **figure 5**, vous pouvez voir la réponse de ChatGPT après lui avoir demandé de fournir un code pour vérifier la valeur du capteur sur la broche 36 de l'ESP32.

Pour surmonter cette limite , nous avons intégré une fonction supplémentaire dans le code, qui permet aux utilisateurs de saisir eux-mêmes le code et d'utiliser ChatGPT comme référence ou guide pour optimiser et réduire le nombre de lignes de code. Cette approche permet de mieux contrôler le code généré et de résoudre les problèmes de formatage et de syntaxe rencontrés lorsque l'on se fie uniquement à la réponse de ChatGPT.

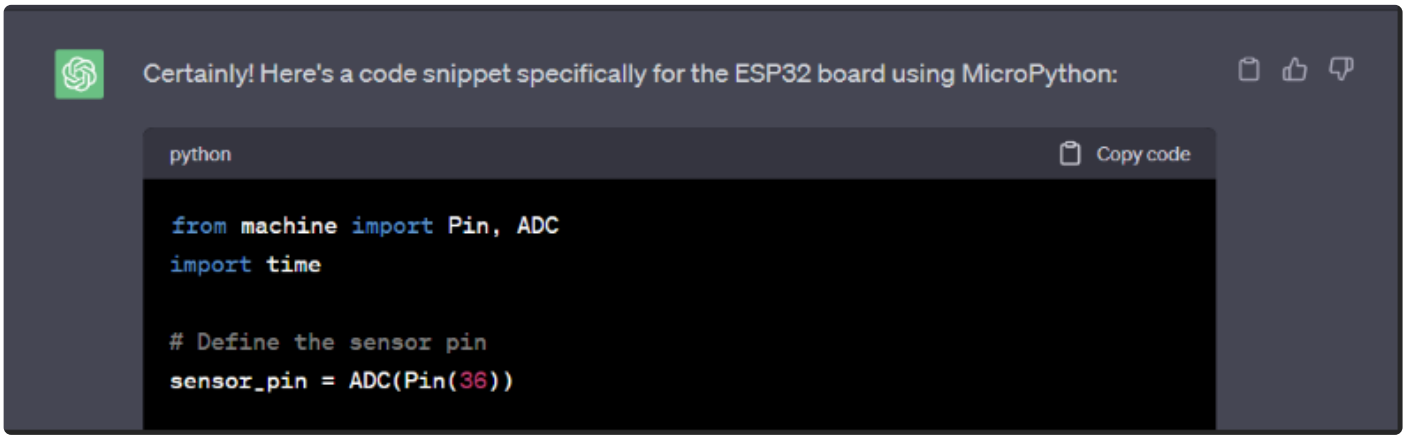


Figure 5. Réponse de ChatGPT après l'avoir invité à écrire un extrait de code.

Applications et cas d'utilisation

Le projet d'intégration des cartes Nano ESP32 avec l'API ChatGPT permet de réaliser toute une série d'applications et de cas d'utilisation intéressants. Voici quelques exemples notables :

- Domotique intelligente : en exploitant l'API ChatGPT, les utilisateurs peuvent interagir avec leurs systèmes domotiques intelligents en utilisant des messages en langage naturel : ils peuvent contrôler les lumières, ajuster les paramètres de température ou même demander des conseils sur les pratiques d'économie d'énergie.
- Prototypage et développement rapide : la combinaison des cartes Nano ESP32 et des extraits de code générés par l'IA permet un prototypage rapide des projets IdO. Les utilisateurs peuvent rapidement tester et développer leurs idées en recevant des suggestions de code instantanées pour différentes fonctions.
- Outil pédagogique : le projet constitue un outil pédagogique précieux pour les débutants en programmation. Les étudiants peuvent participer à des conversations interactives sur la programmation avec l'IA, recevoir des conseils et apprendre de nouvelles techniques.
- Assistant personnel et recherche d'informations : l'intégration de ChatGPT peut être utilisée pour développer une application d'assistant personnel. Les utilisateurs peuvent poser des questions, demander des recommandations ou obtenir des informations sur divers sujets, le tout grâce à des réponses pilotées par l'IA.
- Inspiration en matière de codage : le projet sert de source d'inspiration pour le codage créatif. Il peut générer des idées uniques et innovantes pour des animations, des visualisations ou des projets interactifs, en stimulant la créativité des développeurs.

Il est important de souligner que si ChatGPT peut fournir des informations et des suggestions précieuses, il peut encore s'améliorer dans le domaine de la programmation. L'utilisation de ChatGPT comme référence ou pour l'optimisation du code peut améliorer le de développement, mais elle doit être complétée par une expertise humaine et une révision approfondie du code.

À mesure que les modèles de langage de l'IA continuent d'évoluer, nous pouvons nous attendre à des améliorations dans leur capacité à générer un code plus précis et plus fiable. Ces progrès ouvriront de nouveaux horizons pour utiliser l'IA dans la programmation et permettront une collaboration encore plus étroite entre les humains et les machines. Pour tirer le meilleur parti de ChatGPT, il est important de l'utiliser comme un outil et de ne pas se fier uniquement à ses résultats. La combinaison de l'expertise et du jugement humains avec les réponses générées par l'IA peut conduire à des résultats plus précis et plus fiables. En comprenant et en tenant compte de ces limites, les utilisateurs peuvent utiliser efficacement ChatGPT tout en atténuant les risques et les défis potentiels.

Explorer d'autres approches

En plus du projet décrit précédemment, il existe d'autres méthodes et approches qui pourraient être utilisées pour réaliser la communication Nano ESP32 et l'intégration de l'IA. Explorons les différentes façons dont ce projet peut être implémenté sur la plateforme Nano ESP32 :

1. **Protocole MQTT**
   Le protocole MQTT (*Message Queuing Telemetry Transport*) est un protocole de messagerie léger et efficace couramment utilisé dans les applications IdO. Au lieu de recourir au Wifi et aux connexions socket, les cartes Nano ESP32 peuvent communiquer entre elles grâce au protocole MQTT. Il est possible de configurer des broker MQTT pour faciliter la communication entre les cartes, ce qui permet un échange de données transparent et une intégration de l'IA.
2. **Connexion directe par Web Socket**
   Une autre approche consiste à établir une connexion Web Socket directe entre les cartes Nano ESP32. Web Socket est un protocole de communication qui permet une communication en duplex intégral sur une seule connexion TCP. En implémentant Web Socket sur les cartes Nano ESP32, il est possible d'établir une connexion continue et bidirectionnelle, ce qui permet une communication en temps réel et une intégration de l'IA.
3. **Protocole ESP-NOW**
   ESP-NOW est un protocole de communication spécialement conçu pour les appareils à faible consommation, qui permet une transmission rapide et fiable des données entre les cartes Nano ESP32. Grâce à ce protocole, il est possible d'établir une communication directe entre les cartes, sans avoir recours à des connexions Wifi. En intégrant des capacités d'IA sur une carte et en utilisant ESP-NOW pour la communication, les réponses en temps réel et les extraits de code peuvent être échangés efficacement.
4. **Intégration de l'IA sur l'appareil**
   Au lieu de dépendre d'API externes, il est possible de déployer des modèles d'IA directement sur les cartes Nano ESP32. TensorFlow Lite, par exemple, permet de déployer et d'exécuter localement des modèles d'IA sur des microcontrôleurs. En entraînant ou en optimisant un modèle pour des tâches spécifiques, telles que la génération d'extraits de code, les cartes Nano ESP32 peuvent fournir des réponses pilotées par l'IA sans nécessiter de connexions externes.

Ces méthodes alternatives offrent des avantages et des considérations différents en fonction des exigences et des contraintes spécifiques du projet. Il est important de prendre en compte des facteurs tels que la consommation d'énergie, la latence, la complexité et l'évolutivité lors du choix de l'approche la plus appropriée.

En explorant ces méthodes alternatives, les développeurs peuvent étendre les capacités des cartes Nano ESP32, intégrer l'IA à différents niveaux et adapter le système de communication à leurs propres besoins. Le projet de communication Nano ESP32 et d'intégration de l'IA n'est pas limité à une seule approche. En envisageant des méthodes alternatives telles que MQTT, Web Socket, ESP-NOW ou l'intégration de l'IA sur l'appareil, les développeurs peuvent personnaliser le projet pour répondre à leurs exigences uniques et tirer parti de tout le potentiel de la plate-forme Nano ESP32. Alors, libérez votre créativité, expérimentez différentes méthodes et construisez des systèmes IdO innovants qui combinent l'IA et le Nano ESP32.

EDI Arduino, MicroPython, et plus encore

Arduino IDE Framework

L'EDI Arduino est un choix populaire pour les débutants et les amateurs en raison de sa simplicité et de sa facilité d'utilisation. Il fournit un environnement de programmation convivial avec une version simplifiée du langage de programmation C++. Voici quelques avantages et inconvénients de l'utilisation de l'EDI Arduino avec la Nano ESP32 :

Avantages

- Facile à apprendre : l'EDI Arduino offre une facilité d'apprentissage, ce qui le rend accessible aux débutants en programmation ou en microcontrôleurs.
- Grande communauté et support de bibliothèque : Arduino a une vaste communauté de passionnés, des ressources en ligne étendues et une large gamme de bibliothèques et d'exemples disponibles, ce qui simplifie le développement.
- Prototypage rapide : l'EDI Arduino permet un prototypage rapide grâce à sa syntaxe simplifiée et à la prise en charge des bibliothèques, ce qui accélère les cycles de développement.

Inconvénients

- Langage limité : l'EDI Arduino dispose d'une version simplifiée de C++, ce qui peut limiter l'utilisation de fonctions de programmation avancées par rapport à d'autres langages.
- Mémoire et performance : l'EDI Arduino est parfois moins performant en termes d'utilisation de la mémoire et de performances que d'autres frameworks.
- Moins de flexibilité : l'EDI Arduino impose certaines conventions et limitations qui peuvent restreindre le plein potentiel du Nano ESP32.

Framework MicroPython

MicroPython est une implémentation légère du langage de programmation Python conçue pour les microcontrôleurs. Il offre une expérience de programmation plus flexible et interactive. Voici quelques avantages et inconvénients de l'utilisation du framework MicroPython avec le Nano ESP32 :

Avantages

- Langage Python : MicroPython permet aux développeurs d'exploiter le langage Python puissant, ce qui facilite l'écriture de codes et d'algorithmes complexes.
- REPL interactif : MicroPython fournit un environnement Read-Eval-Print Loop (REPL), permettant aux développeurs de tester et d'expérimenter le code de manière interactive directement sur la carte Nano ESP32.
- Utilisation efficace de la mémoire : MicroPython est conçu pour utiliser efficacement la mémoire, ce qui le rend adapté aux appareils à ressources limitées tels que le Nano ESP32.

Inconvénients

- Courbe d'apprentissage : MicroPython peut être plus difficile à apprendre pour les débutants qui ne sont pas familiers avec le langage Python.
- Support de bibliothèques limité : bien que MicroPython dispose d'une collection en constante évolution de bibliothèques, il peut avoir moins d'options que l'écosystème étendu de bibliothèques Arduino.
- Compromis de performance : bien que MicroPython offre une grande flexibilité, la nature interprétée du langage peut entraîner une exécution légèrement plus lente que les langages compilés tels que C++.

Le croisement fascinant de l'IA et de l'IdO

L'intégration des cartes Nano ESP32 avec l'API ChatGPT représente un croisement fascinant entre l'IA, l'IdO et la programmation. En permettant aux cartes Nano ESP32 de communiquer et d'interagir avec un modèle de langage d'IA, nous créons un éventail de possibilités. De la domotique au prototypage rapide et aux outils éducatifs, les applications sont vastes (voir l'encadré **Cas d'utilisation**).

Toutefois, il est essentiel de garder à l'esprit les limites de ChatGPT et de toujours rester prudent lorsque l'on travaille avec des réponses générées par l'IA, en particulier dans les scénarios de programmation et de codage. La combinaison de l'expertise et du jugement humains avec le contenu généré par l'IA produira de meilleurs résultats.

Avec ce projet, les utilisateurs peuvent libérer leur créativité, explorer de nouveaux horizons dans le développement IdO et améliorer leurs compétences en programmation. Alors, prenez vos cartes Nano ESP32, plongez dans le monde des interactions alimentées par l'IA et profitez du potentiel illimité de cette intégration passionnante !

230485-04

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Saad Imtiaz est un ingénieur expérimenté dans les systèmes embarqués, les systèmes mécatroniques et le développement de produits. Il a collaboré avec plus de 200 entreprises, allant des startups aux entreprises mondiales, sur le prototypage et le développement de produits. Saad a également travaillé dans l'industrie aéronautique et a dirigé une startup technologique. Il a récemment rejoint Elektor en 2023 et participe au développement de projets logiciels et matériels.

Autres environnements

L'un des avantages notables du Nano ESP32 est sa flexibilité à fonctionner avec différents frameworks, outre l'EDI Arduino, MicroPython, il y a PlatformIO, ESP-IDF, JavaScript (Node.js), et les frameworks Lua. Le Nano ESP32 est compatible avec plusieurs autres frameworks, ce qui accroît encore sa polyvalence. Cette flexibilité permet aux développeurs de choisir le framework qui correspond le mieux aux exigences de leur projet et à leur familiarité avec les langages de programmation. Nous allons explorer quelques autres frameworks qui peuvent être utilisés avec le Nano ESP32 et leurs avantages :

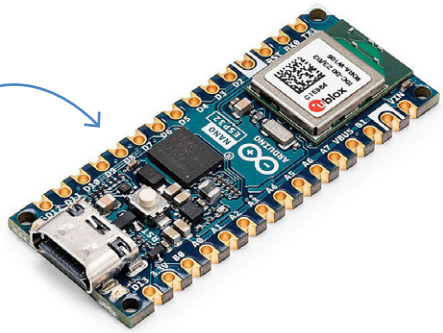
Mongoose OS : Mongoose OS est un système d'exploitation open-source pour microcontrôleurs, comme le Nano ESP32. Il fournit un environnement indépendant de la plateforme avec une connectivité cloud intégrée et prend en charge de nombreux langages de programmation tels que JavaScript, C et C++. Mongoose OS simplifie le processus de développement en offrant une interface de ligne de commande facile à utiliser, de riches bibliothèques et des capacités de contrôle à distance des appareils.

Zephyr RTOS : Zephyr RTOS est un système d'exploitation en temps réel conçu pour les systèmes à ressources limitées, y compris le Nano ESP32. Il offre une architecture évolutive et modulaire, ce qui le rend adapté aux projets qui nécessitent du multitâche, du traitement en temps réel et une gestion avancée des périphériques. Le large support matériel de Zephyr et son ensemble étendu de pilotes et de bibliothèques permettent aux développeurs de créer facilement des applications IdO complexes.

ESPHome : ESPHome, conçu pour les appareils ESP32 et ESP8266, est un framework IdO polyvalent doté d'une architecture modulaire proche du RTOS Zephyr. Il offre un large support matériel, un ensemble de pilotes et de bibliothèques pour simplifier le développement de systèmes à ressources limitées. Ce cadre facilite le traitement en temps réel, le multitâche et la gestion avancée des périphériques, ce qui en fait un excellent choix pour les projets IdO de domotique.

Le choix du bon framework dépend de facteurs tels que les exigences du projet, la familiarité avec le langage de programmation, les bibliothèques disponibles et le soutien de la communauté, ainsi que le niveau de contrôle souhaité sur le matériel et les performances. Chaque framework présente ses avantages et ses inconvénients, ce qui permet aux développeurs de choisir celui qui convient le mieux à leur cas d'utilisation spécifique.

Grâce à la compatibilité du Nano ESP32 avec différents framework, les développeurs ont la liberté d'explorer différents langages de programmation, écosystèmes et approches de développement. Cette flexibilité leur permet de créer des solutions IdO innovantes, d'exploiter les outils et les bibliothèques existants et d'utiliser efficacement les capacités du Nano ESP32.



Produits

- **Arduino Nano ESP32**
  www.elektor.fr/20562
- **Arduino Nano ESP32 avec connecteurs www.elektor.fr/20529**
- **Dogan Ibrahim and Ahmet Ibrahim,** *The Official ESP32 Book* **(E-book) (Elektor 2017)**
  www.elektor.fr/18330
- **Günter Spanner,** *MicroPython for Microcontrollers* **(Elektor 2021)**
  www.elektor.fr/19736

LIENS

[1] Open AI : https://platform.openai.com
[2] Open AI - API Keys : https://platform.openai.com/account/api-keys
[3] Dépôt GitHub de ce projet : https://github.com/elektor-labs/ChatGPTxESP32
[4] Arduino Labs - MicroPython Installer : https://labs.arduino.cc/en/labs/micropython-installer
[5] Arduino Labs - EDI MicroPython : https://labs.arduino.cc/en/labs/micropython