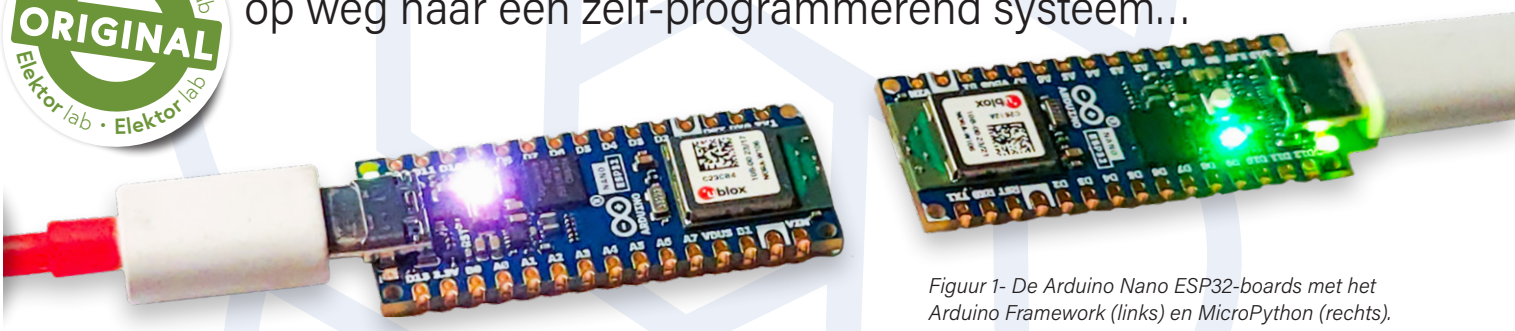
ESP32 en ChatGPT

op weg naar een zelf-programmerend systeem...



Figuur 1- De Arduino Nano ESP32-boards met het Arduino Framework (links) en MicroPython (rechts).

Saad Imtiaz (Elektor Lab)

We hebben de kracht van twee Arduino Nano ESP32-microcontrollers, draadloze communicatie en de ChatGPT API gecombineerd om een slim, interactief communicatiesysteem te maken. Eén van de boards is verbonden met ChatGPT. De antwoorden en de code die dit populaire AI-tool levert worden draadloos overgedragen naar het andere Nano-board. Zou dit gecombineerde systeem zichzelf kunnen programmeren met behulp van ChatGPT? Volg de stap-voor-stap instructies en leer hoe de boards communiceren en hoe de ChatGPT API werkt.

Espressif's ESP32-microcontroller is door zijn veelzijdigheid en robuustheid populair geworden voor toepassing in embedded systemen en het Internet of Things (IoT). Met zijn geïntegreerde WiFi-interface en rekenkracht opent hij een wereld van mogelijkheden voor het bouwen van slimme connected systemen. In dit artikel onderzoeken we een demonstratieproject dat de ChatGPT-API, een AI-taalmodel gemaakt door OpenAI, gebruikt om twee Arduino Nano ESP32-boards naar nieuwe hoogten te tillen.

Het doel van dit project is een naadloos communicatiesysteem tussen de Nano ESP32's te maken. De ene draait met de Arduino IDE en de andere met MicroPython. Door gebruik te maken van de ChatGPT-API, kunnen deze kaarten gezellige interactieve gesprekken aangaan. Stel je eens voor wat er allemaal mogelijk zou zijn als deze microcontrollers zelf op zoek gaan naar codefragmenten om tegemoet te komen aan jouw queries.

In dit artikel gaan we in op de technische aspecten van het opzetten van de hardware, het configureren van de software en het opbouwen van de communicatie tussen de Nano ESP32's. We onderzoeken de voordelen en beperkingen van zowel de Arduino IDE als het MicroPython-framework voor verschillende toepassingen. Ook geven we praktische codefragmenten, uitleg en inzichten om de interne werking van dit project te begrijpen.

Na het lezen van dit artikel zul je begrijpen hoe je een eigen Nano ESP32-communicatiesysteem kunt bouwen om de kracht van AI en IoT te combineren. Laten we ons samen in het avontuur van het bouwen van een bevattelijke, interactieve omgeving met Nano ESP32 en ChatGPT storten!

De hardware

Een aantal componenten is essentieel voor dit project. Om te beginnen kijken we naar de hardware-vereisten.

1. **Arduino Nano ESP32-modules (×2)**
   Het hart van ons project wordt gevormd door de ESP32-microcontroller. Voor de communicatie boards zijn gemakkelijk verkrijgbaar en bieden allerlei mogelijkheden, zoals WiFi, veel rekenkracht en GPIO-pinnen voor het aansluiten van externe componenten.
2. **USB-C-kabel (×2)**
   De USB-C-kabels dienen voor het voeden en programmeren van de Nano ESP32-boards. Deze kabels vormen de verbinding tussen de boards en onze computer voor de overdracht van programma's en data.
3. **WiFi-netwerk**
   Een stabiel WiFi-netwerk is essentieel voor de communicatie tussen de twee boards en de ChatGPT-API. Zorg voor een betrouwbaar WiFi-netwerk met internettoegang. Hiermee kan de eerste Nano ESP32, (we noemen hem "Nano ESP32-1") communiceren met de ChatGPT-API.

Nu we de noodzakelijke hardwarecomponenten hebben, komt de software en het programmeren van het project aan de beurt.

Software en programmeren

Voor het communicatiesysteem en het programmeren hebben we de volgende softwaretools nodig:

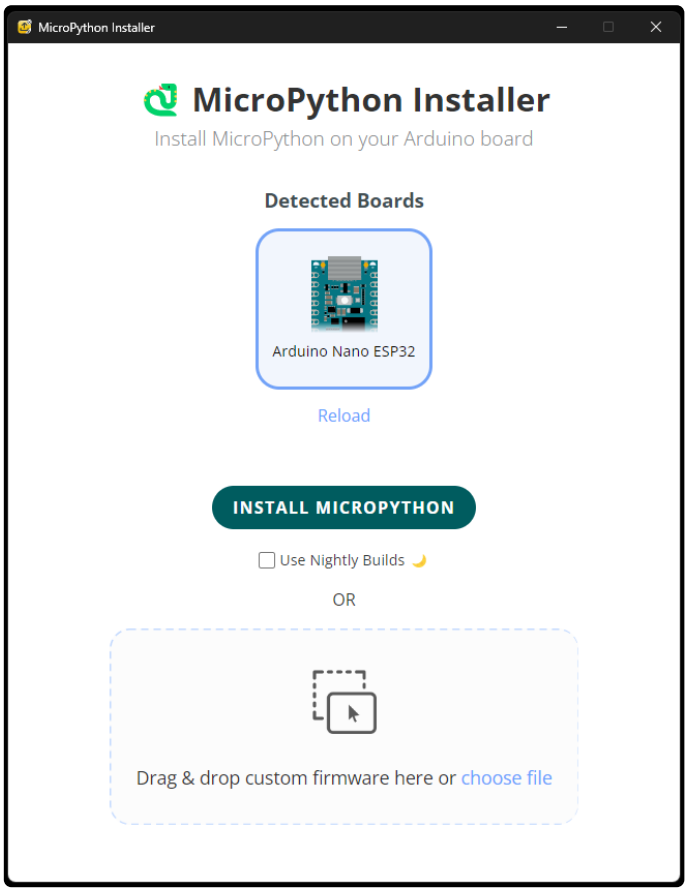
1. **Arduino IDE**
   De Arduino Integrated Development Environment (IDE) is de meest gebruikte IDE voor het programmeren van Arduino-microcontrollers. De nieuwe IDE 2.0 heeft een wat gebruikersvriendelijker interface, een vereenvoudigde programmeertaal en een grote bibliotheek van voorgebakken functies die het gemakkelijker maakt om met de Nano ESP32-boards te werken.
2. **MicroPython-firmware**
   MicroPython is een lichtgewicht versie van de Python-programmeertaal die is geoptimaliseerd voor microcontrollers. Deze levert een meer flexibele en interactieve programmeerervaring dan de Arduino IDE. Om de tweede Nano ESP32 (“ESP32-2”) te programmeren met MicroPython, moeten we de MicroPython-firmware erop installeren. In **figuur 1** zie je de beide boards die met elkaar verbonden zijn. Het linker board draait het Arduino-framework en het rechter draait MicroPython.
3. **Toegang tot de ChatGPT-API**
   Om de Nano ESP32-1 met de ChatGPT-API te verbinden, heb je toegang tot de API nodig. OpenAI biedt verschillende manieren om toegang te krijgen tot de ChatGPT-API met API-keys of tokens. Om een API-key te maken, ga je naar [1]. Maak daar een Open AI-account aan, ga dan naar [2] en klik op *Create New Secret Key*. Je kunt ook op het profielpictogram rechts bovenaan de pagina klikken en daar *View API Keys* kiezen in het menu. Als je een API-key hebt, bewaar die dan op een veilige plaats, want je kunt hem niet nog een keer kopiëren.
4. **Programmeren**
   We beginnen met de Nano ESP32-1 op de Arduino IDE. Voeg de voor onze code noodzakelijke bibliotheken toe. *WiFi.h* is nodig om de Nano ESP32 te verbinden met het WiFi-netwerk. *HTTPClient.h* wordt gebruikt voor het zenden van onze data (de ChatGPT-respons) naar de tweede Nano ESP32. En *ArduinoJson.h* is de bibliotheek om JSON te gebruiken op vrijwel elk board. Daarmee doen we de JSON-serialisatie en -deserialisatie, die we gebruiken voor de communicatie met de ChatGPT-API. De code voor de twee boards is ook beschikbaar op 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";
```

Nu voegen we het IP-adres van de tweede Nano ESP32 toe (dit IP-adres wordt getoond in de seriële monitor in de IDE als de tweede Nano ESP32 wordt verbonden met het WiFi-netwerk).

```
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
```

Er is een speciale functie geschreven voor het verbinden en communiceren met de ChatGPT-API (**listing 1**).
Nu gaan we naar de code van de tweede Nano ESP32. Voordat we beginnen met programmeren, bespreken we eerst hoe we MicroPython uitvoeren op de Nano ESP32: we gebruiken *Arduino Lab for MicroPython*. Eerst gaan we naar [4] en downloaden het Arduino MicroPython-firmwaretool om de Arduino Nano ESP32 te flashen (**figuur 2**). Sluit de Arduino Nano ESP32 gewoon aan op de computer. Als het board is gedetecteerd, klik je op *Install MicroPython* en na enkele seconden is het board geflasht met de nieuwste firmware en kun je MicroPython gaan gebruiken op de Nano ESP32.
Nu downloaden we de nieuwste versie van Arduino Lab for MicroPython IDE van de officiële website op [5]. Na het uitpakken kun je de software starten door *Arduino Lab voor Micropython.exe* te draaien. Op de tweede Nano ESP32 is de code even kort als eenvoudig. Eerst importeren we de bibliotheken. We gebruiken de bibliotheken *network* en *socket* om verbinding te maken met het WiFi-netwerk en onze serverpoort te maken voor het ontvangen van de code of antwoorden van de Nano ESP32-1. Verder zijn de bibliotheken *machine* en *time* nodig om de GPIO’s en de timer-functie voor de Nano ESP32.



Figuur 2. Het MicroPython Firmware-tool van Arduino Labs.

Listing 1. Arduino-code om requests naar ChatGPT te sturen.

```
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;
}
```

```
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])
```

Dan wordt een socket-server aangemaakt voor het ontvangen van data van Nano ESP32-1 en gaat de MCU in een while-lus waar hij wacht op binnenkomende antwoorden. Als code of een antwoord wordt ontvangen, voert de Nano ESP32 de code uit en dan gaat hij weer wachten op verdere instructies (zie **listing 2**).
In het kader **Arduino IDE, MicroPython – en andere** bespreken we de voor- en nadelen van de beide frameworks en om te begrijpen waarom het ene beter geschikt zou kunnen zijn dan het andere voor bepaalde toepassingen.

Communicatie tussen de Nano ESP32-boards

Laten we ons nu in de technische details storten, hoe de twee Nano ESP32-boards in dit project via WiFi met elkaar praten. We geven hier een gedetailleerde uitleg van het communicatieproces.

Nano ESP32-1 (Arduino IDE framework)
Nano ESP32-1, die op het Arduino IDE framework draait, maakt een WiFi-verbinding om te communiceren met externe services en apparaten. Hij maakt verbinding met een specifiek WiFi-netwerk met de gegeven parameters. Als de verbinding is gemaakt, wacht de Nano ESP32-1 op invoer van de gebruiker via de seriële monitor van de Arduino IDE.
Als de gebruiker *GPT* invoert en op Enter drukt, vraagt de Nano ESP32-1 hem om een boodschap in te tikken. Deze boodschap wordt als prompt van de Nano ESP32-1 naar de ChatGPT-API gestuurd. Dat gaat via de opgebouwde WiFi-verbinding, waarmee de Nano ESP32-1 kan communiceren met de externe API. In **figuur 3** zie je de verbindingen en communicatie tussen de twee boards en de ChatGPT-API.
De ChatGPT-API, een interface naar het AI-taalmodel, ontvangt de prompt van Nano ESP32-1. Met behulp van geavanceerde natuurlijke taalverwerkingstechnieken en machine learning-algoritmen analyseert de API de prompt om die te 'begrijpen' en een gevat antwoord of een codefragment te genereren. Het ChatGPT-model in de API

gebruikt zijn enorme hoeveelheid trainingsdata en taalherkenningsmogelijkheden om een relevante en nuttige uitvoer te genereren.
Nano ESP32-1 ontvangt het gegenereerde codefragement van de ChatGPT-API en slaat dat op. Dit codefragment is de AI-gegenereerde respons op de prompt van de gebruiker. Nano ESP32-1 geeft dan het ontvangen codefragment weer op de seriële monitor, zodat gebruikers de door de AI gegenereerde instructies kunnen beoordelen. In **figuur 4** zie je de seriële uitvoer van de beide boards, waarbij de Nano ESP32-1 de code naar Nano ESP32-2 stuurt na het ontvangen van de respons van ChatGPT.

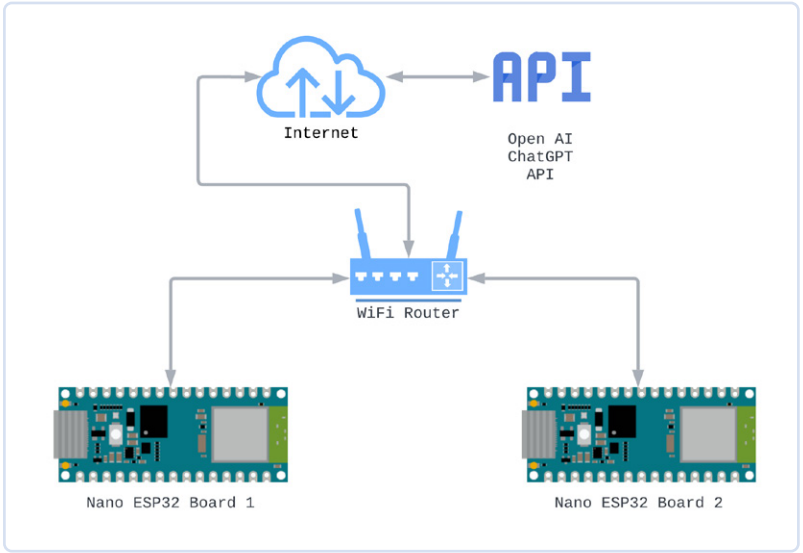
Nano ESP32-2 (MicroPython framework)
De Nano ESP32-2 wordt aangestuurd met MicroPython. Ook de Nano ESP32-2 maakt verbinding met het WiFi-netwerk met behulp van de gegeven parameters. Zo kan hij draadloos instructies ontvangen van de Nano ESP32-1.
Nano ESP32-2 zet een socket-verbinding op en luistert op een specifieke poort (typisch poort 80). Een socket is een software-eindpunt dat communicatie tussen twee apparaten over een netwerk mogelijk maakt. Door op een specifieke poort te luisteren, is Nano ESP32-2 klaar om binnenkomende data van Nano ESP32-1 te ontvangen.
Als Nano ESP32-1 de gegenereerde code wil verzenden, maakt hij een verbinding met Nano ESP32-2 via de socketverbinding. Dan wordt het codefragment naar Nano ESP32-2 verzonden, waar het wordt ontvangen en verwerkt.
Nano ESP32-2 verwerkt de ontvangen code door de instructies die erin staan uit te voeren. Deze instructies leggen specifieke taken vast die Nano ESP32-2 moet uitvoeren. Dat kan bijvoorbeeld de opdracht zijn om een LED voor een bepaalde tijd te laten knipperen, of een andere door de AI gegenereerde respons.
Na het uitvoeren van de instructies stuurt Nano ESP32-2 een antwoord terug naar Nano ESP32-1 via de opgebouwde socketverbinding. Dit geldt als een bevestiging dat de ontvangen code succesvol is uitgevoerd. Zo weet Nano ESP32-1 dat de gevraagde taak is uitgevoerd.
Met behulp van dit communicatieproces kunnen de twee Nano ESP32-boards effectief berichten uitwisselen en samenwerken. Nano ESP32-1 praat met de ChatGPT-API om gebruik te maken van de AI-mogelijkheden, terwijl Nano ESP32-2 functioneert als de ontvanger en uitvoerder van de door AI gegenereerde instructies.

Beperkingen van ChatGPT bij het programmeren

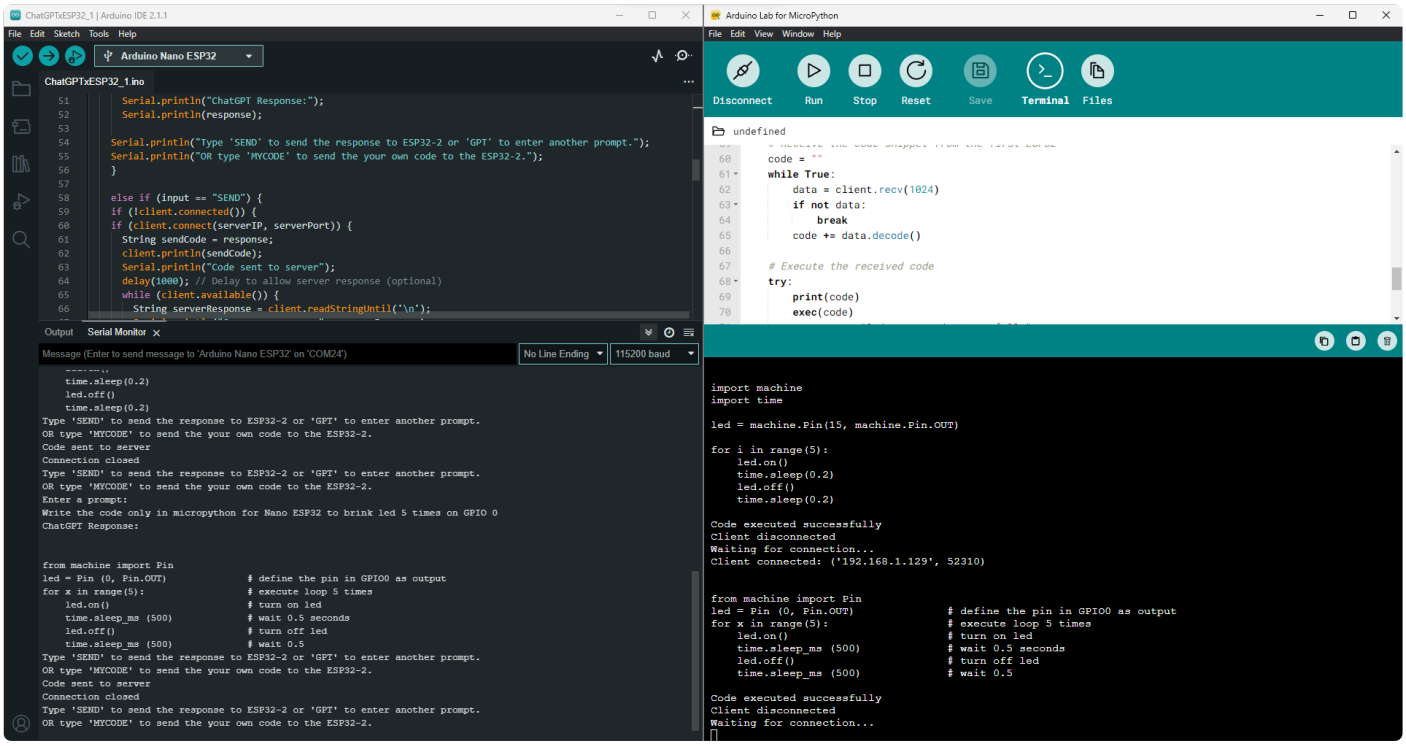
ChatGPT is een heel goed AI-taalmodel, maar het heeft wel beperkingen als er functionele code gegenereerd moet worden, vooral bij ingewikkelde programmeerscenario's. Bij het testen bleek dat ChatGPT in slechts 60% van de gevallen in staat was om functionele code voor een eenvoudige knipper-sketch te maken. Dat wijst erop dat het model niet altijd in staat is om nauwkeurige, werkende code te genereren.

Listing 2. Code ontvangen en uitvoeren op het tweede board (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")
```



Figuur 3. Communicatie tussen de twee Arduino Nano ESP32-boards en de ChatGPT-API.



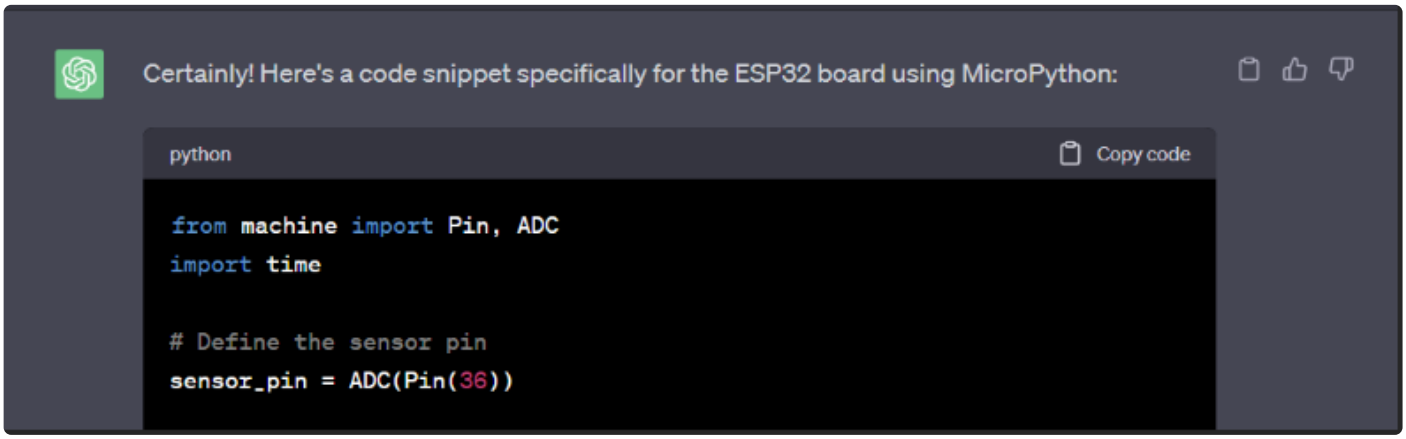
Figuur 4: Communicatie tussen de twee Nano ESP32's, met Arduino IDE (links) en met Arduino Lab for MicroPython (rechts).

Wanneer we ChatGPT om code vragen, moeten we nog met iets anders rekening houden: soms bevat de respons extra tekst voor en na de eigenlijke code, die problemen kan geven als we die één-op-één naar de tweede Nano ESP32 sturen. Die tekst heeft waarschijnlijk niet de juiste syntax en is dus niet uitvoerbaar. In **figuur 5** zie je de respons van ChatGPT toen hem gevraagd werd om code te maken voor het uitlezen van de sensorwaarde op pen 36 van de ESP32.

Als oplossing voor dit probleem geeft ons systeem gebruikers de mogelijkheid om de code zelf in te voeren en ChatGPT alleen te gebruiken als een referentie of gids bij het maken van de code. Dit geeft de gebruiker meer controle over de gegenereerde code en lost de layout- en syntaxproblemen op die ontstaan als we 100% zouden vertrouwen op de uitvoer van ChatGPT.

We moeten ons wel realiseren dat ChatGPT nuttige inzichten en suggesties kan leveren, maar dat er nog ruimte voor verbetering is bij het programmeren. Ontwikkelaars moeten voorzichtig zijn en niet puur vertrouwen op ChatGPT's antwoorden voor belangrijke of ingewikkelde programmeertaken. Gebruik van ChatGPT als referentie of voor code-optimalisatie kan het ontwikkelproces verbeteren, maar moet worden aangevuld met menselijke deskundigheid en nauwkeurige beoordeling van de code.

Als AI-taalmodellen zich blijven ontwikkelen, mogen we verwachten dat ze steeds beter worden in het genereren van betrouwbare code. Die vooruitgang zal nieuwe uitzichten openen voor het gebruik van AI bij het programmeren en nog meer samenwerking tussen mensen en machines mogelijk maken. Om optimaal te profiteren van ChatGPT,



Figuur 5. Respons van ChatGPT na een verzoek om een codefragment te schrijven.

Toepassingsscenario's
Het project van het integreren van Nano ESP32-kaarten met de ChatGPT-API opent een wereld van interessante toepassingen en use cases. Hierbij enkele opmerkelijke voorbeelden:

- **Smart Home Automation:** door gebruik te maken van de ChatGPT API kunnen gebruikers communiceren met hun smart home-systemen in natuurlijke taal. Ze kunnen de verlichting bedienen, de temperatuurinstellingen aanpassen of zelfs om aanbevelingen vragen over energiebesparende praktijken.
- **Snelle prototypes en ontwikkeling:** de combinatie van Nano ESP32-boards en door AI gegenereerde codefragmenten maakt snelle prototyping van IoT-projecten mogelijk. Gebruikers kunnen ideeën snel testen en itereren dankzij direct ontvangen codesuggesties voor verschillende functionaliteiten.
- **Educatief hulpmiddel:** het project biedt een waardevol educatief hulpmiddel voor wie leert te programmeren. Studenten kunnen deelnemen aan interactieve codeersessies met de AI, begeleiding krijgen en nieuwe programmeertechnieken leren.
- **Persoonlijke assistent en informatiebron:** de chat GPT-integratie kan worden gebruikt om een persoonlijke assistent-applicatie te ontwikkelen. Gebruikers kunnen vragen stellen, aanbevelingen zoeken of informatie verkrijgen over verschillende onderwerpen, allemaal mogelijk gemaakt door AI-gestuurde antwoorden.
- **Creatieve code-inspiratie:** het project dient als een bron van creatieve code-inspiratie. Het kan unieke en innovatieve ideeën genereren voor animaties, visualisaties of interactieve projecten, waardoor de creativiteit van ontwikkelaars wordt gestimuleerd.

moeten we het zien als een gereedschap en niet blindelings vertrouwen op zijn uitvoer. De combinatie van menselijke deskundigheid met de AI-gegenereerde antwoorden kan leiden tot nauwkeuriger en betrouwbaarder resultaten. Als we rekening houden met deze beperkingen, kunnen we ChatGPT effectief gebruiken en de potentiele risico's verminderen.

Alternatieve benaderingen

Naast de besproken aanpak in dit project zijn er alternatieve manieren en benaderingen om de Nano ESP32 te laten communiceren en te integreren met AI. We noemen hier een aantal verschillende manieren om dit project te implementeren op het Nano ESP32-platform:

1. **MQTT-Protocol**
   Het MQTT-protocol (Message Queuing Telemetry Transport) is een lichtgewicht en efficiënt protocol voor het uitwisselen van berichten dat veel wordt gebruikt in IoT-toepassingen. In plaats van WiFi- en socket-verbindingen kunnen Nano ESP32-boards met elkaar communiceren met behulp van het MQTT-protocol. MQTT-brokers kunnen het uitwisselen van berichten tussen de boards vergemakkelijken voor naadloze communicatie en AI-integratie.
2. **Direct Web Socket-verbinding**
   Een andere benadering is het opzetten van een directe Web Socket-verbinding tussen de Nano ESP32's. Web Socket is een communicatieprotocol dat tweerichting-communicatie over één enkele TCP-verbinding mogelijk maakt. Door Web Socket te implementeren op de Nano ESP32-boards, kunnen ze een permanente verbinding in twee richtingen opbouwen, waarmee real time-communicatie en AI-integratie mogelijk wordt.
3. **ESP-NOW-protocol**
   ESP-NOW is een communicatieprotocol dat specifiek is ontworpen voor low power-apparaten, waarmee snelle en betrouwbare dataoverdracht tussen Nano ESP32-boards mogelijk is. Dit protocol kan worden gebruikt om directe communicatie tussen de boards op te zetten zonder gebruik te maken van WiFi-verbindingen. Door de AI-mogelijkheden te integreren op één kaart en gebruik te maken van ESP-NOW voor de communicatie, kunnen real time-antwoorden en code efficiënt worden uitgewisseld.
4. **On-Device AI-integratie**
   Het is ook mogelijk om AI-modellen rechtstreeks op Nano ESP32-kaarten te draaien, in plaats van te vertrouwen op externe API's. Zo kunnen bijvoorbeeld met TensorFlow Lite AI-modellen lokaal op microcontrollers worden ingezet en uitgevoerd. Door een model te trainen of te tunen voor specifieke taken, zoals het genereren van code, kunnen de Nano ESP32-boards zelf AI-gestuurde antwoorden geven zonder externe verbindingen.

Deze alternatieve benaderingen hebben verschillende voor- en nadelen afhankelijk van de specifieke projectvereisten en -beperkingen. We moeten factoren als energieverbruik, latency, complexiteit en schaalbaarheid in overweging nemen om de beste aanpak te kiezen.
Door deze alternatieven te onderzoeken, kunnen ontwikkelaars de mogelijkheden van Nano ESP32's uitbreiden, AI integreren op verschillende niveaus en het communicatiesysteem aanpassen aan hun behoeften.
Het project van Nano ESP32-communicatie en AI-integratie kan op meerdere manieren worden aangepakt. Door alternatieven zoals MQTT, Web Socket, ESP-NOW, of On-Device AI-integratie te overwegen, kunnen ontwikkelaars het project aanpassen aan hun unieke eisen en profiteren van alle kracht van het Nano ESP32-platform. Dus laat je creativiteit de vrije loop, experimenteer met verschillende benaderingen en bouw innovatieve IoT-systemen die AI en Nano ESP32 op nieuwe manieren combineren.

Arduino IDE, MicroPython – en andere

Arduino IDE framework

De Arduino IDE is populair bij beginners en hobbyisten vanwege zijn eenvoud en bedieningsgemak. Hij biedt een beginnersvriendelijke programmeeromgeving met een vereenvoudigde versie van de programmeertaal C++. Hier enkele voordelen en nadelen van het gebruik van de Arduino IDE met de Nano ESP32:

Voordelen

- Gemakkelijk te leren: de Arduino IDE heeft geen steile leercurve, en is dus toegankelijk voor wie begint met programmeren of niet bekend is metmicrocontrollers.
- Grote community and ondersteunende bibliotheken: Arduino heeft een grote community van hobbyisten, uitgebreide online-hulpmiddelen en een grote verzameling van bibliotheken en voorbeelden, die het ontwikkelen vergemakkelijken.
- Snelle prototypes: de Arduino IDE maakt snelle prototypes mogelijk door zijn vereenvoudigde syntax en bibliotheek-ondersteuning, zodat snel resultaten zichtbaar worden.

Nadelen

- Beperkte taalmogelijkheden: de Arduino IDE heeft een vereenvoudigde versie van C++, waardoor er minder geavanceerde programmeermogelijkheden zijn dan in andere talen.
- Geheugen en prestaties: de Arduino IDE is misschien niet zo efficiënt qua geheugengebruik en prestaties vergeleken met andere frameworks.
- Minder flexibiliteit: de Arduino IDE legt bepaalde conventies en beperkingen op, die de mogelijkheden van de Nano ESP32 beperken.

MicroPython framework

MicroPython is een lichtgewicht versie van de Python-programmeertaal die is geoptimaliseerd voor microcontrollers. Het biedt een flexibele, interactieve programmeerervaring. Hier enkele voordelen en nadelen van het gebruik van MicroPython met de Nano ESP32:

Voordelen

- De taal Python: met MicroPython kunnen ontwikkelaars de krachtige taal Python gebruiken, waardoor gemakkelijker ingewikkelde code en algoritmen te schrijven zijn.
- Interactieve REPL: MicroPython heeft een Read-Eval-Print Loop-omgeving (REPL), waarmee ontwikkelaars rechtstreeks op het Nano ESP32-board de code interactief kunnen testen en ermee experimenteren.
- Efficiënt geheugengebruik: MicroPython is ontworpen om zuinig met geheugen om te gaan, wat het geschikt maakt voor apparaten met beperkte resources, zoals de Nano ESP32.

Nadelen

- Leercurve: MicroPython kan lastig zijn voor beginners die nog niet bekend zijn met de taal Python.
- Beperkte bibliotheek-ondersteuning: er komen steeds meer bibliotheken voor MicroPython beschikbaar, maar het aanbod is toch een stuk kleiner dan bij het Arduino-ecosysteem.
- Prestatie-compromis: MicroPython biedt veel flexibiliteit, maar het is een geïnterpreteerde taal. Daardoor worden programma's wat langzamer uitgevoerd dan bij gecompileerde talen zoals C++.

Conclusie: een boeiend snijvlak van AI en IoT

De integratie van Nano ESP32-boards met de ChatGPT-API vormt een fascinerend snijvlak tussen AI, IoT en programmeren. Door Nano ESP32-boards onderling te laten communiceren en te koppelen met een AI-taalmodel, ontsluiten we een wereld van mogelijkheden. Van slimme domotica tot rapid prototyping en educatieve gereedschappen, de toepassingen zijn eindeloos (zie het kader **Toepassingsscenario's**). We moeten wel steeds de beperkingen van ChatGPT in gedachten houden en voorzichtig zijn met het gebruikt van AI-gegenereerde antwoorden, zeker bij het programmeren en coderen. De combinatie van menselijke deskundigheid en beoordelingsvermogen met AI-gegenereerde inhoud geeft de beste resultaten.

Met dit project kunnen gebruikers hun creativiteit benutten, nieuwe uitzichten in de IoT-ontwikkeling ontdekken en hun coderingsvaardigheden vergroten. Dus pak je Nano ESP32's, duik in de wereld van AI-gestuurde interacties en benut de mogelijkheden van deze interessante integratie!

230485-03 (vertaling: Evelien Snel)

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Saad Imtiaz is een ontwikkelaar met ervaring in embedded systemen, mechatronische systemen en productontwikkeling. Hij heeft samengewerkt met meer dan 200 bedrijven, van startups tot globale ondernemingen, bij prototyping en ontwikkeling. Saad heeft ook gewerkt in de luchtvaartindustrie en heeft een technische startup geleid. Hij kwam in 2023 bij Elektor aan boord en stimuleert de projectontwikkeling in zowel software als hardware.

Andere frameworks

Eén van de opmerkelijke voordelen van de Nano ESP32 is zijn flexibiliteit: hij werkt met verschillende frameworks – naast Arduino IDE en MicroPython zijn dat onder meer PlatformIO, ESP-IDF, JavaScript (Node.js) en Lua. De Nano ESP32 is compatibel met verschillende andere frameworks, wat bijdraagt aan de veelzijdigheid. Door die flexibiliteit kunnen ontwikkelaars de ontwikkelomgeving kiezen die het best past bij hun projecteisen en hun kennis van programmeertalen.

Dit zijn een paar andere frameworks die je met de Nano ESP32 kunt gebruiken:

Mongoose OS: Mongoose OS is een open source-besturingssysteem voor microcontrollers, zoals de Nano ESP32. Het is een platform-onafhankelijke omgeving met ingebouwde cloud-verbinding en ondersteunt verschillende programmeertalen zoals JavaScript, C, en C++. Mongoose OS vereenvoudigt het ontwikkelproces met een gebruikersvriendelijke commandoregel-interface, rijke bibliotheken en mogelijkheden voor beheer op afstand.

Zephyr RTOS: Zephyr RTOS is een real time-besturingssysteem ontworpen voor systemen met beperkte resources, zoals de Nano ESP32. Het heeft een schaalbare, modulaire architectuur die het geschikt maakt voor projecten die multitasking, real time-verwerking en geavanceerd apparaatmanagement nodig hebben. Zephyr's ruime hardware-ondersteuning en uitgebreide verzameling van drivers en bibliotheken helpen ontwikkelaars om met gemak ingewikkelde IoT-toepassingen te bouwen.

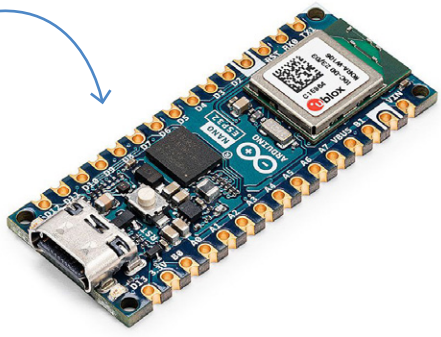
ESPHome: ESPHome is ontworpen voor de ESP32 en de ESP8266. Het is een veelzijdig IoT-framework met een modulaire architectuur, vergelijkbaar met Zephyr RTOS. Het biedt een brede hardware-ondersteuning, veel drivers en bibliotheken voor vereenvoudigde ontwikkeling voor systemen met beperkte resources. Dit platform ondersteunt real time-verwerking, multitasking en geavanceerd apparaatbeheer, wat het een uitstekende keus maakt voor domotica en IoT-projecten.

De keuze van het juiste framework wordt bepaald door de projecteisen, bekendheid met de programmeertaal, beschikbare bibliotheken, ondersteuning door een community en het gewenste niveau van besturing van de hardware en performance. Elk framework heeft zijn eigen voordelen en compromissen, zodat ontwikkelaars het meest geschikte kunnen kiezen voor hun specifieke toepassing.

De compatibiliteit van de Nano ESP32 met verschillende frameworks geeft ontwikkelaars de vrijheid om verschillende programmeertalen, ecosystemen en ontwikkelmethoden te onderzoeken. Met die flexibiliteit kunnen ze innovatieve IoT-oplossingen maken met bestaande tools en bibliotheken en de mogelijkheden van de Nano ESP32 efficiënt benutten.

Gerelateerde producten

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



WEBLINKS

[1] Open AI: https://platform.openai.com
[2] Open AI - API Keys: https://platform.openai.com/account/api-keys
[3] GitHub-repository van dit project: https://github.com/elektor-labs/ChatGPTxESP32
[4] Arduino Labs - MicroPython Installer: https://labs.arduino.cc/en/labs/micropython-installer
[5] Arduino Labs - MicroPython IDE: https://labs.arduino.cc/en/labs/micropython