Faza 3: głos asystenta

> Typ: procedura · Stan: zaliczona 2 października 2026 · Wymaga: zaliczona faza 2

Łączymy model z syntezą mowy: tekst z LLM trafia do TTS, a dźwięk do głośników. Zaczynamy od Pipera, który działa na procesorze obok modelu, a naturalny głos Chatterboxa testujemy osobno.

Skróty użyte na tej stronie (16)

Skrót	Po angielsku	Po polsku i co znaczy
AI	Artificial Intelligence	sztuczna inteligencja: programy, które naśladują część ludzkiego rozumowania i rozmawiają w języku naturalnym
ASCII	American Standard Code for Information Interchange	amerykański standardowy kod wymiany informacji: podstawowy zestaw znaków
CPU	Central Processing Unit	procesor centralny: główny procesor komputera
CUDA	Compute Unified Device Architecture	platforma firmy NVIDIA do obliczeń na jej kartach graficznych; od niej zależą sterownik i szybkość modeli
GB	Gigabyte	gigabajt: miliard bajtów, jednostka rozmiaru plików i pamięci
GPU	Graphics Processing Unit	procesor graficzny, czyli karta graficzna: liczy modele językowe wielokrotnie szybciej niż zwykły procesor
LLM	Large Language Model	duży model językowy: program przewidujący kolejne fragmenty tekstu, na którym opiera się czat AI
MIT	Massachusetts Institute of Technology; MIT License	w pierwszym znaczeniu amerykańska uczelnia techniczna, w której powstała ELIZA Weizenbauma; w drugim licencja MIT, bardzo liberalna licencja wolnego oprogramowania, która pozwala na dowolne użycie i zmiany kodu
MPL	Mozilla Public License	publiczna licencja Mozilli: otwarta licencja wolnego oprogramowania
PyPI	Python Package Index	indeks pakietów Pythona: publiczny rejestr bibliotek, z którego pobiera pakiety pip
Q5	5-bit quantization	kwantyzacja pięciobitowa: zapis parametrów modelu na około 5 bitach, większy plik i zwykle lepsza jakość niż Q4
RTF	Real-Time Factor	współczynnik czasu rzeczywistego: czas generowania podzielony przez czas trwania wypowiedzi; powyżej 1 oznacza wolniej niż na żywo

Skrót	Po angielsku	Po polsku i co znaczy
tok/s	tokens per second	tokenów na sekundę: miara szybkości, z jaką model generuje odpowiedź
TTS	Text-to-Speech	zamiana tekstu na mowę: program, który czyta tekst głosem
UTF	Unicode Transformation Format	format kodowania znaków Unicode, na przykład UTF-8
VRAM	Video Random Access Memory	pamięć karty graficznej: w niej musi zmieścić się model, żeby liczył się na karcie

□□ Leksykon: nazwy i pojęcia z tej strony (15)

Nazwa	Co to jest
awatar VRM	Postać 3D, którą widzimy podczas rozmowy; VRM to format pliku takiej postaci, z gotowymi wyrazami twarzy i ruchem ust.
Bielik	Polski model językowy rozwijany przez SpeakLeash i ACK Cyfronet. U nas Bielik-Minitron-7B w wersji v3 jest modelem rozmowy.
Chatterbox	Syntezator mowy o bardziej naturalnym głosie, działający na karcie graficznej. Zajmuje około 3 GB VRAM, więc na razie go odłożyliśmy.
czat	Rozmowa z modelem, w której on tylko odpowiada tekstem. Sam niczego nie uruchamia ani nie zmienia, dopóki nie damy mu narzędzi.
model językowy	Program wyuczony na dużych zbiorach tekstu, który dopisuje kolejne fragmenty tekstu pasujące do tego, co już jest. Rozmowa z nim to ciągłe dopisywanie, a nie szukanie faktów w bazie.
Modelfile	Plik Ollamy opisujący, jak zbudować własną odmianę modelu: model bazowy, prompt systemowy i parametry.
pip i PyPI	pip to instalator pakietów Pythona, a PyPI to publiczne składowisko tych pakietów.
Piper	Lokalny, szybki syntezator mowy. U nas działa na procesorze, więc nie zajmuje pamięci karty graficznej.
PowerShell	Powłoka i język skryptowy Windows. W nim wykonujemy polecenia z naszych stron dla Windowsa.
prompt i prompt systemowy	Prompt to tekst, którym dajemy modelowi polecenie. Prompt systemowy ustawia twórca aplikacji, a nie użytkownik, i opisuje rolę oraz zasady modelu; jest to prośba, a nie zabezpieczenie.
Python	Język programowania, w którym piszemy klienty, serwery MCP i skrypty.
STT i TTS	Dwie połowy rozmowy głosowej: STT zamienia mowę na tekst, TTS zamienia tekst na mowę.
test akceptacyjny	Sprawdzenie opisane z góry, które faza musi przejść, żeby uznać ją za skończoną.

Nazwa	Co to jest
token	Słowo o dwóch znaczeniach. W modelu to kawałek tekstu (słowo, jego część albo znak), na które model dzieli wszystko, co czyta i pisze; od liczby tokenów zależą szybkość i koszt. W dostępie to ciąg znaków, który zastępuje hasło i otwiera drzwi do usługi.
XTTS-v2 i RVC	Modele głosu: XTTS-v2 syntezuje mowę z klonowaniem głosu, RVC zmienia barwę istniejącego głosu. Oba wymagają karty graficznej.

Zanim zaczniemy

Wolnego VRAM przy załadowanym modelu jest mniej niż 1 GB. Dlatego pierwszy silnik TTS musi liczyć na procesorze, a każdy silnik na GPU wymaga osobnej decyzji o podziale karty.

Silnik	Polski	Jakość	Gdzie liczy	Licencja
Piper 1.8.0	tak (gosia, mc_speech, darkman, bass, mls_6892)	wyraźny, sztuczny	CPU	MIT
XTTS-v2 (coqui-tts 0.27.5)	tak	naturalny, klonowanie głosu	GPU	biblioteka MPL 2.0, wagi niekomercyjne
Chatterbox Multilingual 0.1.7	tak	naturalny, regulacja ekspresji	GPU	MIT

Krok 1: Python i Piper

Python instalujemy dopiero w tej fazie, bo dopiero tu jest potrzebny. Używamy `uv`, który trzyma interpreter i cache w `C:\AI\apps`, a każdy silnik dostaje osobne środowisko.

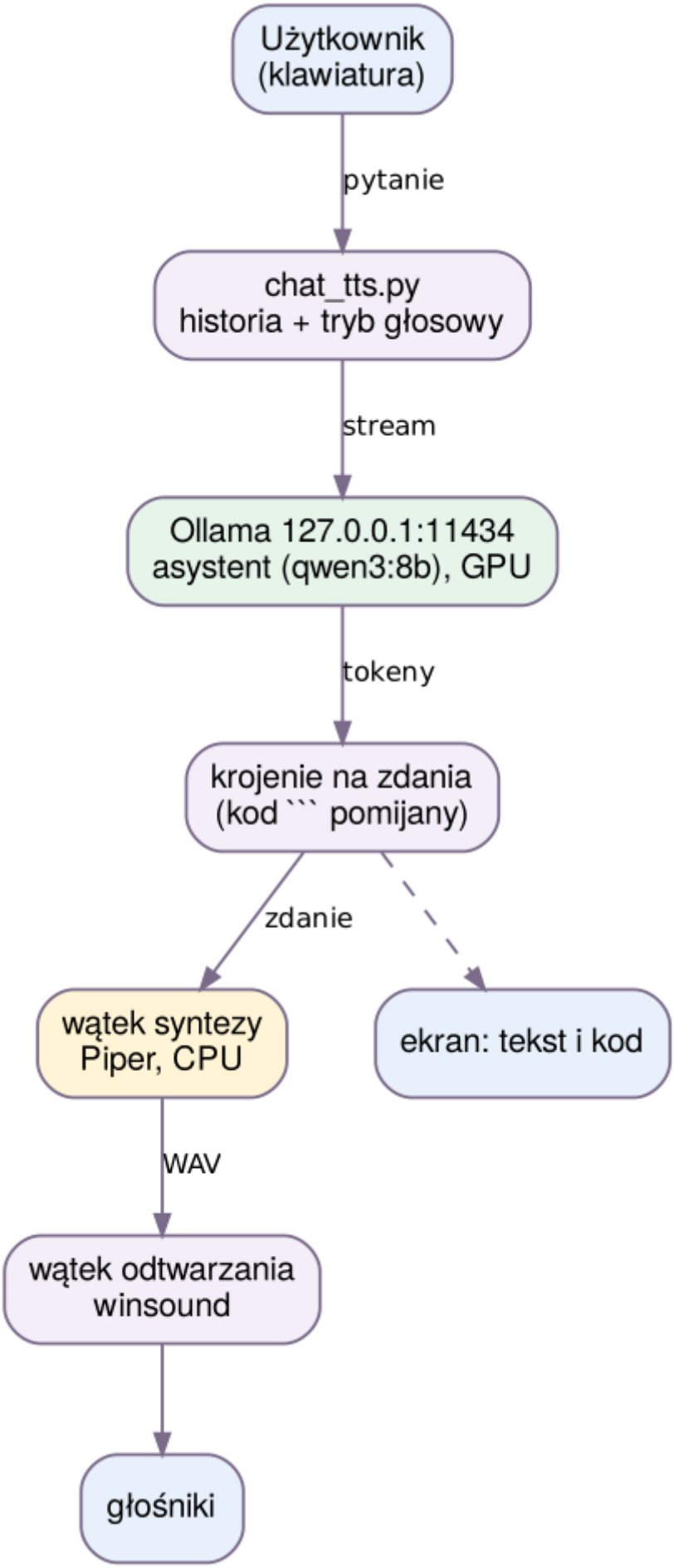
```
winget install --id astral-sh.uv --exact
[Environment]::SetEnvironmentVariable('UV_PYTHON_INSTALL_DIR', 'C:\AI\apps\python', 'User')
[Environment]::SetEnvironmentVariable('UV_CACHE_DIR', 'C:\AI\apps\uv-cache', 'User')
uv python install 3.12
uv venv C:\AI\tts\venv --python 3.12 --managed-python
uv pip install --python C:\AI\tts\venv\Scripts\python.exe piper-tts==1.8.0
& C:\AI\tts\venv\Scripts\python.exe -m piper.download_voices --download-dir C:\AI\models\piper
pl_PL-gosia-medium pl_PL-mc_speech-medium
```

Przykład pomiaru skryptem `C:\AI\tts\say.py`:

```
pl_PL-gosia-medium: ładowanie 4.29 s, synteza 0.80 s, nagranie 8.2 s, RTF 0.10
pl_PL-mc_speech-medium: ładowanie 4.38 s, synteza 0.84 s, nagranie 6.8 s, RTF 0.12
```

RTF to czas syntezy podzielony przez długość nagrania. Wartość 0,10 oznacza, że Piper mówi dziesięć razy szybciej, niż trwa sama wypowiedź.

Krok 2: rozmowa z głosem


Użytkownik
(klawiatura)
pytanie
chat_tts.py
historia + tryb głosowy
stream
Ollama 127.0.0.1:11434
asystent (qwen3:8b), GPU
tokeny
krojenie na zdania
(kod ``` pomijany)
zdanie
wątek syntezy
Piper, CPU
ekran: tekst i kod
WAV
wątek odtwarzania
winsound
głośniki

Skrypt `C:\AI\tts\chat_tts.py` wysyła pytanie do modelu `asystent` ze strumieniowaniem i kroi odpowiedź na zdania w trakcie generowania. Pierwsze zdanie jest czytane, zanim model skończy pisać. Synteza kolejnego zdania trwa w osobnym wątku, gdy poprzednie jest odtwarzane.

Kod w blokach jest pokazywany na ekranie, ale nie czytany. Skróty takie jak „np.” rozwijamy do pełnych słów. Aplikacja dopisuje do rozmowy jedno zdanie „trybu głosowego”: odpowiadaj krótko, zacznij od zwykłego zdania, a kod opisz słowami.

Przykład:

```
Ty> Cześć! Przedstaw się krótko.
AI> Cześć! Jestem AI, który pomaga w nauce programowania i rozwiązywaniu problemów.
[pierwszy token 0.26 s · pierwszy dźwięk 0.56 s · generowanie 0.9 s, 71 tok/s · koniec mowy
10.3 s]
```

Pierwszy dźwięk po 0,56 s od naciśnięcia Enter to dobry wynik dla rozmowy.

Krok 3: Chatterbox i dlaczego go odkładamy

Chatterbox ma osobne środowisko z Pythonem 3.11 i PyTorchem 2.6.0 z CUDA. Domyślny torch z PyPI na Windows nie obsługuje GPU, więc instalujemy go z indeksu `download.pytorch.org` przed samym Chatterboxem.

```
uv venv C:\AI\tts\venv-chatterbox --python 3.11 --managed-python
uv pip install --python C:\AI\tts\venv-chatterbox\Scripts\python.exe torch==2.6.0
torchaudio==2.6.0 --index-url https://download.pytorch.org/whl/cu124
uv pip install --python C:\AI\tts\venv-chatterbox\Scripts\python.exe chatterbox-tts==0.1.7
"setuptools<81"
```

Skrypt `C:\AI\tts\test_chatterbox.py` zwalnia model LLM z VRAM i mierzy sam TTS.

Przykład:

```
VRAM wolny przed załadowaniem: 6.94 GB z 8.00 GB
Model wczytany w 19.3 s
VRAM zajęty przez model: 3.00 GB
exaggeration 0.5: synteza 41.3 s, nagranie 7.1 s, RTF 5.81, szczyt VRAM 3.24 GB
exaggeration 0.8: synteza 16.7 s, nagranie 6.6 s, RTF 2.55, szczyt VRAM 3.28 GB
```

Wynik wyklucza Chatterboxa z rozmowy na żywo na tej karcie. Model 8B, Chatterbox i pulpit potrzebują razem około 10,5 GB przy 8 GB VRAM, a nawet sam Chatterbox mówi dwa i pół raza wolniej, niż trwa wypowiedź. Domyślny głos jest męski, bo barwę daje wyłącznie próbka referencyjna. Zostawiamy go do renderowania gotowych kwestii albo na mocniejszą kartę.

Krok 4: wyższy, młodszy ton bez dodatkowych bibliotek

Piper ma RTF 0,10, więc stać nas na obróbkę dźwięku. Syntezujemy wolniej (`length_scale` razy k) i zapisujemy nagranie z częstotliwością próbkowania razy k. Tempo wraca do normy, a ton rośnie o $12 \cdot \log_2(k)$ półtonów, bez żadnych obliczeń.

```
with wave.open(out, "wb") as w:
    w.setparams(params)
    w.setframerate(int(params.framerate * k))
    w.writeframes(frames)
```

Skrypt `C:\AI\tts\test_ton.py` odtwarza próbki od k = 1,00 do 1,25. Wybraliśmy głos gosia, k = 1,10 (o 1,7 półtonu wyżej) i tempo 0,95 (5 procent szybciej). To ustawienia domyślne w `chat_tts.py`, które nadpisujemy flagami `--ton` i `--tempo`.

Przesuwają się także formanty, czyli barwa głosu. Przy małym k daje to lżejszy, młodszy głos, a przy dużym efekt „chipmunka”.

Krok 5: ekspresja i komunikat o kodzie

Piper nie ma parametru emocji. Najbliżej są dwa parametry losowości modelu: `noise_scale` zmienia intonację, a `noise_w_scale` rytm długości głosek. Mnożymy domyślne wartości głosu przez 1,15 (flaga `--ekspresja`), co daje 0,767 i 0,92 zamiast 0,667 i 0,8.

Powyżej około 1,3 pojawiają się przeciągnięte albo połknięte głoski. Prawdziwe emocje zależne od treści dodamy w fazie awatara: aplikacja dostanie od modelu emocję i dobierze do niej ton, tempo i ekspresję.

Krótkie, osobne zdanie „Kod wyświetla się w oknie czatu.” brzmiało inaczej niż reszta, bo Piper nadaje melodię całemu zdaniu. Dlatego doklejamy komunikat do zdania przed kodem.

Przykład:

```
Oto przykładowa komenda, a kod wyświetla się w oknie czatu.
```

Krótkie polecenia w linii czytamy wprost („Użyj polecenia df -h.”), a dłuższe zastępujemy słowami „polecenie z ekranu”.

Test akceptacyjny

Przykład przy wyłączonej karcie sieciowej, z modelem Bielik-Minitron Q5:

```
Internet = False
[model asystent]
[ekspresja x1.15: noise_scale 0.767, noise_w 0.920]
Ty> Cześć! Opowiedz w dwóch zdaniach, co potrafisz.
AI> Jako lokalny asystent AI, mogę pomóc w wyjaśnianiu zagadnień technicznych (...)
    Nie mam jednak dostępu do narzędzi, internetu ani możliwości wykonywania poleceń poza tą
konwersacją.
[pierwszy token 0.86 s · pierwszy dźwięk 2.83 s · generowanie 2.8 s, 66 tok/s · koniec mowy
22.1 s]
```

Kryteria: odpowiedź słychać bez internetu, kod jest tylko na ekranie, model pamięta bieżącą rozmowę i nie zmyśla wcześniejszych.

Częste błędy

- **Wiadomość system w zapytaniu zastępuje prompt z Modelfile.** Aplikacja, która wysyła własną instrukcję (np. tryb głosowy), musi dołączyć do niej prompt bazowy. Inaczej model traci zasady i znów twierdzi, że „wykonuje polecenia”. `chat_tts.py` czyta `C:\AI\config\system-prompt.txt` i wysyła oba razem.
- **Odpowiedź z samego kodu brzmi jak cisza.** Gdy model odpowie wyłącznie blokiem kodu, nie ma czego czytać, a pomiar pokazuje `pierwszy dźwięk nan`. Pomaga reguła „zacznij od zwykłego zdania” i zabezpieczenie w aplikacji, które mówi „Kod jest na ekranie.”.
- **Polskie znaki przez potok.** Windows PowerShell 5.1 przekazuje tekst do programów przez potok w kodowaniu ASCII. Tekst do Pipera podajemy jako argument albo z pliku UTF-8.
- **`PerthImplicitWatermarker` jest `None`.** Pakiet znaku wodnego ukrywa błąd importu `pkg_resources`, a środowiska `uv` nie mają `setuptools`. Pomaga `uv pip install`

`"setuptools<81"`; prawdziwą przyczynę pokazuje bezpośredni import `perth.perth_net.perth_net_implicit.perth_watermarker`.

- **Torch bez CUDA.** Po instalacji Chatterboxa sprawdzamy `torch.__version__`; musi kończyć się na `+cu124`.

Ćwiczenia

1. Uruchom `chat_tts.py --ton 1.0` i `chat_tts.py --ton 1.2`, a potem porównaj, jak zmienia się wrażenie wieku głosu.
2. Zapytaj o komendę PowerShell i sprawdź, czy słychać zdanie wyjaśnienia, a polecenie jest tylko na ekranie.

Revision #11
Created 2026-10-01 21:35:28 UTC by Tomasz Konieczny
Updated 2026-10-20 09:05:54 UTC by Tomasz Konieczny