```
import os
import subprocess
import sys

import requests
from dotenv import load_dotenv

from swarm_models.base_llm import BaseLLM

try:
    import wave
except ImportError as error:
    print(f"Import Error: {error} - Please install pyaudio")
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pyaudio"]
    )


# Load .env file
load_dotenv()


# OpenAI API Key env
def openai_api_key_env():
    openai_api_key = os.getenv("OPENAI_API_KEY")
    return openai_api_key
```

```
class OpenAITTS(BaseLLM):
    """OpenAI TTS model

    Attributes:
        model_name (str): _description_
        proxy_url (str): _description_
        openai_api_key (str): _description_
        voice (str): _description_
        chunk_size (_type_): _description_

    Methods:
        run: _description_


    Examples:
    >>> from swarm_models.openai_tts import OpenAITTS
    >>> tts = OpenAITTS(
    ...     model_name = "tts-1-1106",
    ...     proxy_url = "https://api.openai.com/v1/audio/speech",
    ...     openai_api_key = openai_api_key_env,
    ...     voice = "onyx",
    ... )
    >>> tts.run("Hello world")
```

```
"""

def __init__(
    self,
    model_name: str = "tts-1-1106",
    proxy_url: str = "https://api.openai.com/v1/audio/speech",
    openai_api_key: str = openai_api_key_env,
    voice: str = "onyx",
    chunk_size=1024 * 1024,
    autosave: bool = False,
    saved_filepath: str = None,
    *args,
    **kwargs,
):
    super().__init__()
    self.model_name = model_name
    self.proxy_url = proxy_url
    self.openai_api_key = openai_api_key
    self.voice = voice
    self.chunk_size = chunk_size
    self.autosave = autosave
    self.saved_filepath = saved_filepath

    self.saved_filepath = "runs/tts_speech.wav"

def run(self, task: str, *args, **kwargs):
```

```
    """Run the tts model

    Args:
        task (str): _description_

    Returns:
        _type_: _description_
    """
    response = requests.post(
        self.proxy_url,
        headers={
            "Authorization": f"Bearer {self.openai_api_key}",
        },
        json={
            "model": self.model_name,
            "input": task,
            "voice": self.voice,
        },
    )

    audio = b""
    for chunk in response.iter_content(chunk_size=1024 * 1024):
        audio += chunk
    return audio

def run_and_save(self, task: str = None, *args, **kwargs):
```

"""Run the TTS model and save the output to a file.

Args:

    task (str): The text to be converted to speech.

    filename (str): The path to the file where the speech will be saved.

Returns:

    bytes: The speech data.

"""

Run the TTS model.

speech_data = self.run(task)

Save the speech data to a file.

with wave.open(self.saved_filepath, "wb") as file:

    file.setnchannels(1)

    file.setsampwidth(2)

    file.setframerate(22050)

    file.writeframes(speech_data)

return speech_data