Documentation: VSCODE

Necessary requirements:

```
%pip install gradio #ALLOWS developers to quickly build UI’s
%pip install requests #Simplifies web communications
%pip install pyserial #communications with arduino
%pip install ipywidgets #Enables buttons and interactive elements
%pip install tqdm #Fast progress bars
```

SETUP:

- Code defines and uses Openrouter API, and creates last-version json.
- Code defines two global variables x and y.
- Code setups global cancelling event :to cancel buttons.

GPT BUILD INSTRUCTIONS: function that returns:

- Code defines API build instructions: including input expected, mandatory output format, and error handling, ex:"OUTPUT REQUIREMENT (MANDATORY):

Sending to OpenRouter:

- Sends input with API key and build instructions, with Max tokens set to 250.
    - Tokens are a fundamental unit of data, like a word.

Extracting text:

- Converts message into raw string in json format. JSON DUMPS { }.

Text strip:

- Uses .strip() to strip text and quotes to just get two numbers. Json loads it back to a python object like a list.

INSTANCE creation:

- Creates variables frequency and amplitude. Looks in dictionary and json objects, and assigns the json object in the dictionary key for key in ["frequency_hz", "frequency", "freq_hz", "freq", "x"]:

REGEX Heuristics:

- Uses pattern based making to strip text in case JSON did not return expected output, searching for key numbers in text strip.

- Performs basic validation making sure frequency and voltage is in reasonable values.

Set Variable:

- Complicated code that uses json and gets the frequency and amplitude from json files. Then uses command os.environ["FREQUENCY_HZ"] = str(x), to set global variables x and y with frequency and voltage.

<u>Generate and Parse:</u>

- This acts like the main hub running all functions such as generation, stripping, and updating of global variable.

List models:

- Custom functions for OpenRouter to request a JSON list of all models available for AI Generation.

LIST SERIAL PORTS:

- Lists all serial ports using pyserial and key function
    - ports = serial.tools.list_ports.comports()

SEND PARSED TO ARDUINO:

- Sends the parsed f and a, tries to get signal from arduino, if it receives “ARDUINO READY” from the arduino COM channel, it turns ready = True and sends the parsed variables.
- ACK symbolizes acknowledgement, NACK symbolizes the communication was not able to reach out to the arduino.

MANUAL PARSED:

- Accepts precision 6 decimal places for hertz and volts. Auto adjusts parsed variables f and a to be equal to volt and hertz inputted manually.
- Switches the parsed overlay block from generated to manual, with new text lines and a new json list.

EVERYTHING BELOW:

- UI Design using HTML and gradio.
- Defines all elements such as send, cancel, list ports, etc.. to relevant functions within the top of the code.

if __name__ == "__main__":

- Sets demolaunch to be true when the run feature is clicked. Running the whole code.

Complete Summary:

This document outlines a Python application that uses an AI language model (via OpenRouter API) to interpret natural language commands, convert them into structured control parameters, and send those instructions to an Arduino via serial communication. A user-friendly web interface is built with Gradio.

Core Functionality:
The system acts as an intelligent translator between a user's text input (e.g., "a slow, wobbly signal") and precise hardware control. It generates two numerical values—frequency (Hz) and amplitude/voltage (V)—which are sent to an Arduino to control a connected device, like a motor or signal generator.
Technical Process:

AI Interpretation: User text is sent to the OpenRouter API with strict instructions to output only a JSON object containing the two numbers.

Response Parsing: The AI's text response is rigorously cleaned, converted to JSON, and validated. Heuristics and regex patterns ensure the extraction of the two target numbers even if the output is malformed.

Parameter Assignment: The extracted numbers are mapped to frequency and amplitude variables, validated for reasonable ranges, and set as global environment variables.

Hardware Communication: The system checks for an available Arduino on a serial port. Upon receiving an "ARDUINO READY" signal, it transmits the parsed frequency and amplitude values. It listens for an acknowledgment (ACK/NACK) of the communication.

Manual Override: A manual input mode allows users to bypass the AI and directly enter precise frequency and voltage values.