


Assignment 24: Support multiple languages - 10422030

Questions:

A universal translator is a device that can translate between multiple languages, allowing folks who speak different languages to be able to communicate. Use what you have learned over the past few lessons to build a universal translator using 2 IoT devices.

If you do not have 2 devices, follow the steps in the previous few lessons to set up a virtual IoT device as one of the IoT devices.

You should configure one device for one language, and one for another. Each device should accept speech, convert it to text, send it to the other device via IoT Hub and a Functions app, then translate it and play the translated speech.

 Tip: When sending the speech from one device to another, send the language it is in as well, making it easier to translate. You could even have each device register using IoT Hub and a Functions app first, passing the language they support to be stored in Azure Storage. You could then use a Functions app to do the translations, sending the translated text to the IoT device.

Answer:

- In this assignment, I will simulate the universal translator on the local computer. Because I can't access the Azure and also don't have enough two IoT devices to test so that simulation is the best choice.

1. First, we create the [device1.py](#) to simulate the first device.

- This device is used to Translate English to vietnamese.

```
~/Desktop/Assignment 24/universal_translator/device1.py - Mousepad
File Edit Search View Document Help
from googletrans import Translator

# English to vietnam

def device1_translate():
    # read input from user
    with open("device1_input.txt", "r", encoding="utf-8") as f:
        text = f.read()
    print(f"Device 1 talk (English): {text}")

    # Dịch sang tiếng Việt
    translator = Translator()
    translated = translator.translate(text, src="en", dest="vi").text
    print(f"Translated to Vietnamese: {translated}")

    # write and send to device2
    with open("device1_to_device2.txt", "w", encoding="utf-8") as f:
        f.write(translated)

device1_translate()
```

2. Then, we also create the [device2.py](#)

- This [device2.py](#) can be used to translate the Vietnamese to English.

```
~/Desktop/Assignment 24/universal_translator/device2.py - Mousepad
File Edit Search View Document Help
from googletrans import Translator

# vietnam to English

def device2_translate():
    # read file from device1
    with open("device1_to_device2.txt", "r", encoding="utf-8") as f:
        received_text = f.read()
    print(f"Device2 recieved (Vietnamese): {received_text}")

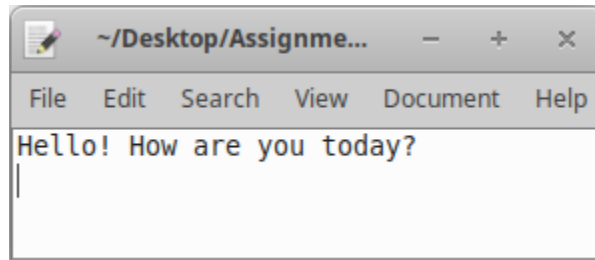
    # translate to English (reponse)
    translator = Translator()
    translated = translator.translate(received_text, src="vi", dest="en").text
    print(f"Translate to English (reponse): {translated}")

    # write the file
    with open("device2_response.txt", "w", encoding="utf-8") as f:
        f.write(translated)

device2_translate()
```

3. We create the simulated input from device 1.

- To run the simulation, we need the input, so that we must create the input file and start at device 1.

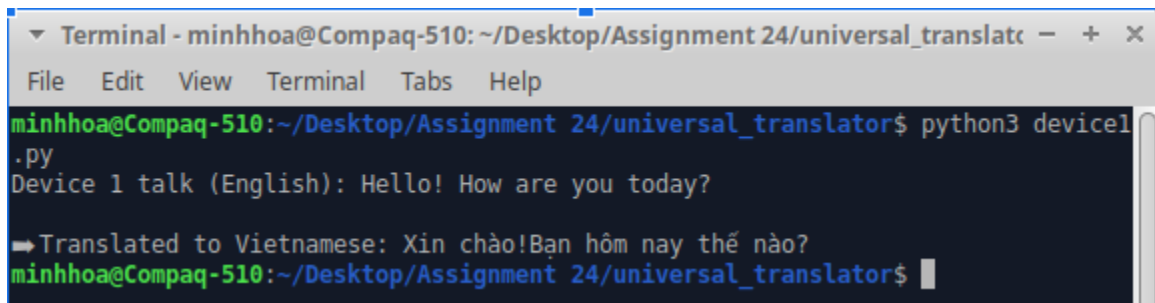


4. Run the test.

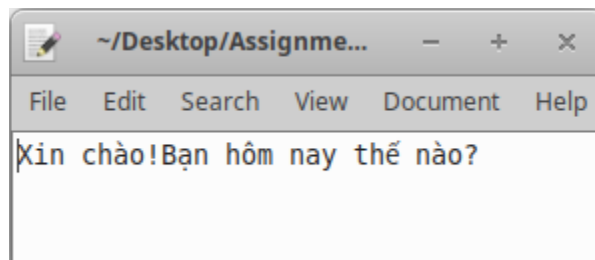
- To run the simulation, first we need to install the library:

pip install googletrans==4.0.0-rc1

- The run the first [device1.py](#)

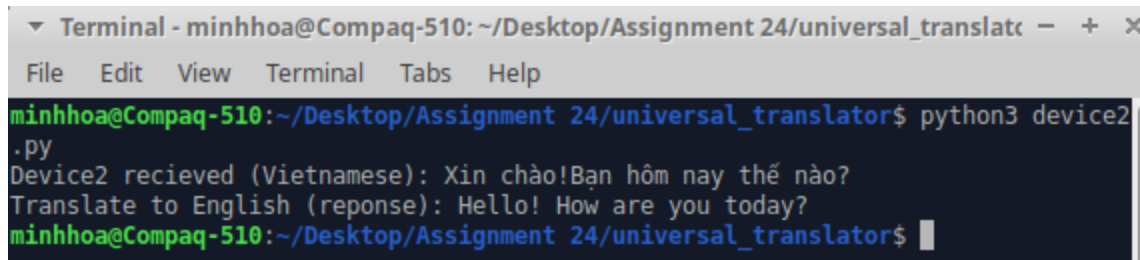


- In the device1_to_device2.txt, we can see that.



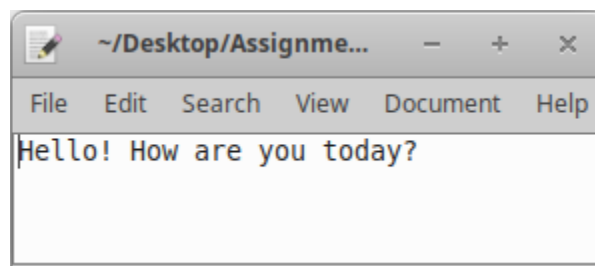
- > It is similar to the code on the terminal.

- Then, we run the second [device2.py](#)

A terminal window titled "Terminal - minhhoa@Compaq-510: ~/Desktop/Assignment 24/universal_translatc" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "minhhhoa@Compaq-510:~/Desktop/Assignment 24/universal_translator\$". The user enters "python3 device2.py". The output shows "Device2 recieved (Vietnamese): Xin chào!Bạn hôm nay thế nào?" followed by "Translate to English (reponse): Hello! How are you today?". The prompt returns to "minhhhoa@Compaq-510:~/Desktop/Assignment 24/universal_translator\$".

```
minhhhoa@Compaq-510:~/Desktop/Assignment 24/universal_translator$ python3 device2.py
Device2 recieved (Vietnamese): Xin chào!Bạn hôm nay thế nào?
Translate to English (reponse): Hello! How are you today?
minhhhoa@Compaq-510:~/Desktop/Assignment 24/universal_translator$
```

-> The result is that, when we call the [device1.py](#), we will receive the device1_to_device2.txt file that means English is translated to Vietnamese. When we run the [device2.py](#), we will have device2_response.txt and this is the translation again to English.



- This is similar to the two device translations and received through the IoT Hub.

5. Conclusion

In this Assignment, we successfully simulated a universal translator system using two virtual IoT devices on a local machine without relying on Azure or physical hardware. Each device was emulated using Python scripts: one acted as a source that captured English speech in the form of text, translated it into Vietnamese, and sent the result via a shared file; the other device received the translated message and translated it back into English.

The system demonstrates the core concept of an IoT-based universal translator, including speech-to-text, translation, and communication between devices.

Although real-time audio and cloud-based services like Azure IoT Hub were not used due to platform limitations, the simulation accurately reflects the data flow and logic expected in a full implementation. This approach provides a strong foundation for future integration with speech recognition, text-to-speech, and real IoT hardware.