Final Report

ECE 579: Intelligent Robotics 2

Interfacing Fritz Robot Head with ChatGPT

By

Mehul Rajendra Shah

Under the guidance of

Prof. Marek Perkowski

Table of Contents

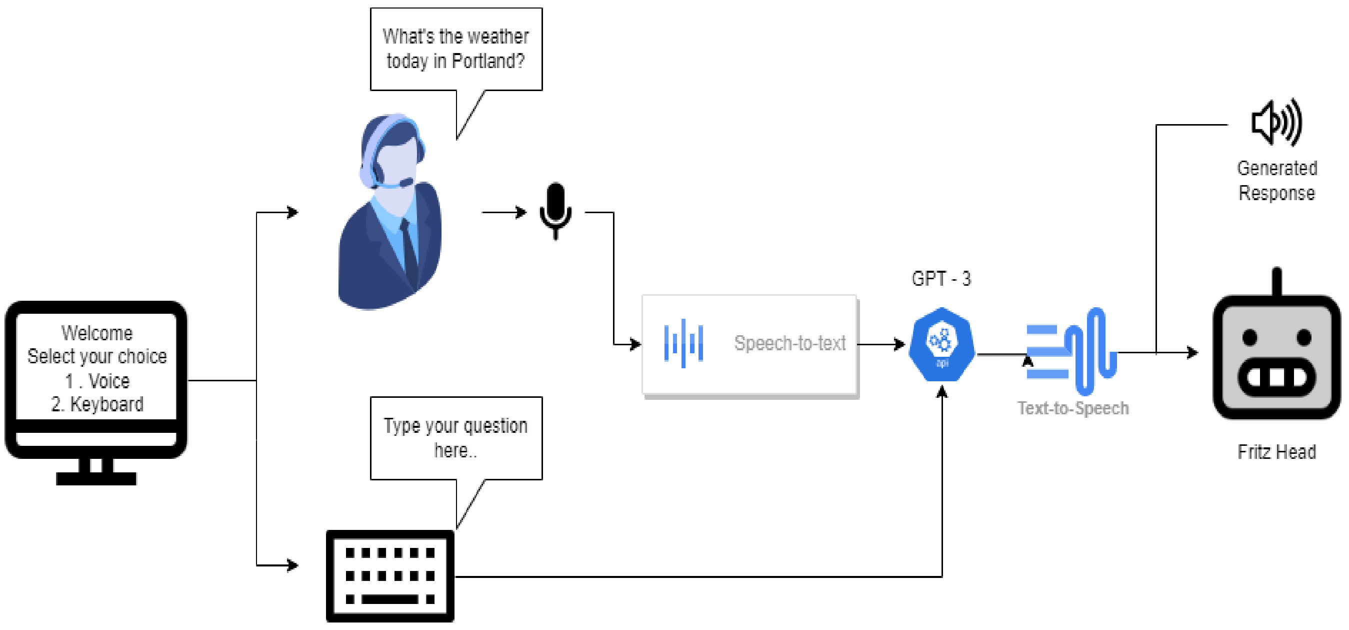
1. Introduction

Artificial intelligence (AI) has always focused on developing emotional robots with intelligent control over facial expressions. In designing a humanoid robot head, it is divided into three steps to achieve a lifelike appearance. The first step involves overcoming the "uncanny valley" effect, which aims to avoid creating a relationship between humans and robots that can feel uncomfortable or unsettling. The second step is to establish the association between human faces and robot heads by analyzing the similarities and differences using the Facial Action Coding System (FACS). This system guides the exploration of the same basis and mechanisms between robots and humans to achieve humanoid expressions. Based on these first two steps, the third step is to construct a robot head and test it through a series of experiments. Through human-robot interaction, researchers have found that people are surprised by the robot head's expressions and feel happy.

Fritz is the animatronic robot head which can be programmed with simple software. Introducing the Fritz Robot Head interfaced with GPT-3, a revolutionary project that combines the power of artificial intelligence and robotics to create a unique and engaging user experience. The project involves using a Fritz Robot Head, a highly advanced animatronic head with 13 servo motors that enable it to move its jaws and facial expressions with great accuracy. The project utilizes GPT-3, one of the most advanced natural language processing models developed by OpenAI, which enables the robot head to generate intelligent responses to user input. The system works by capturing the user's voice input through a microphone, which is then converted into text format by the GPT-3 model. The GPT-3 model then processes the text input and generates an appropriate response based on the input received. The response is then passed back to the Fritz Robot Head, which translates it into speech and movements of the jaw and facial expressions to provide a realistic and engaging experience to the user.

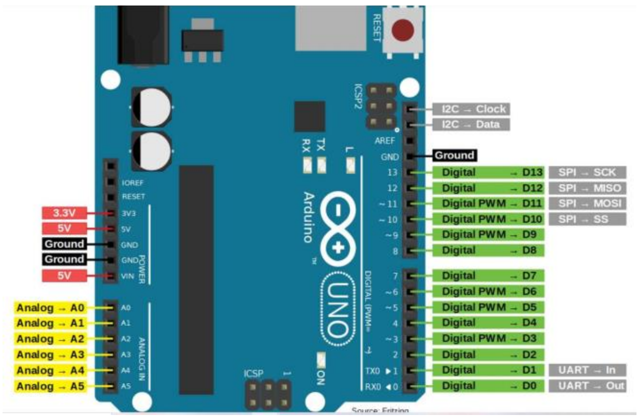
The Fritz Robot Head and GPT-3 interface have several potential applications, including education, entertainment, and communication. For example, the robot head could be used to teach language skills or as a conversational partner for individuals who may struggle with social interactions. In conclusion, the Fritz Robot Head interfaced with GPT-3 is a groundbreaking project that combines cutting-edge technology to provide a unique and engaging experience to users. The project is an excellent example of how robotics and artificial intelligence can be used to create innovative solutions that improve our lives.

1.1. Flowchart



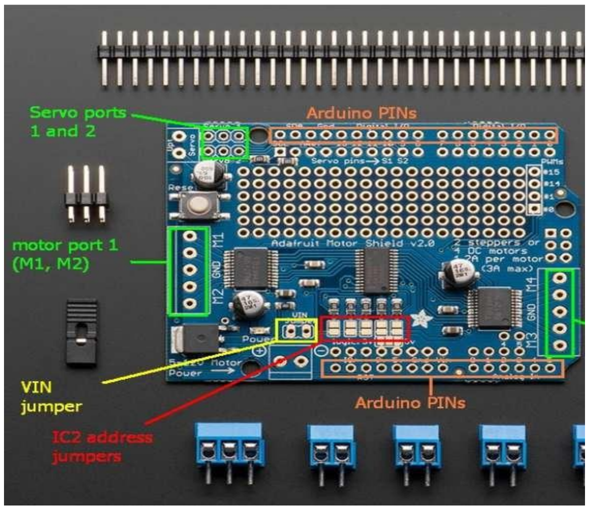
1.2. Getting familiar with components used

Arduino:



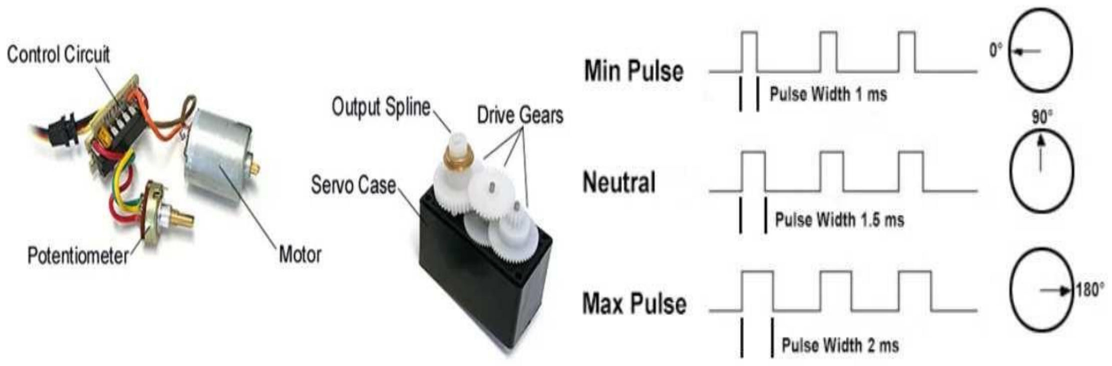
The Arduino Uno is a microcontroller board, based on the ATmega328P (for Arduino UNO R3) or ATmega4809 (for Arduino UNO WIFI R2) microcontroller by Atmel and was the first USB powered board of Arduino. The Atmega328 and ATmega4809 comes with built-in bootloader, which makes it very easy to flash the board with your code. Like all Arduino boards, you can program the software running on the board using a language derived from C and C++. The easiest development environment is the Arduino IDE

Arduino Servo Shield:



This is a simple Arduino sensor shield that goes on top of Arduino uno. It provides ground and VCC signals for each digital and analog pin on Arduino separately on its side. It also provides a set of different connectors as required for power, ground, and TX/RX pins. It also has an external power input port. It can supply up to 9V of power. This power will be regulated to 5 volts and be provided on all the VCC pins that are on the side of analog and digital pins.

Servo motors:



Servo motors are controlled through pulse width modulation (PWM), which involves sending an electrical pulse of variable width via the control wire. Each pulse has a minimum and maximum width, and a repetition rate. A servo motor can turn up to 90° in either direction, giving it a total of 180° movement. The neutral position of the motor is the point where the servo has the same potential rotation in both the clockwise and counterclockwise directions.

The duration of the pulse sent via the control wire determines the position of the motor's shaft. A pulse of 1.5 milliseconds (ms) will make the motor turn to the 90° position, while shorter or longer pulses will turn the servo in the counterclockwise or clockwise direction, respectively. Servos expect to receive a pulse every 20ms, and the length of the pulse will determine how far the motor turns.

2. Head interfacing and testing

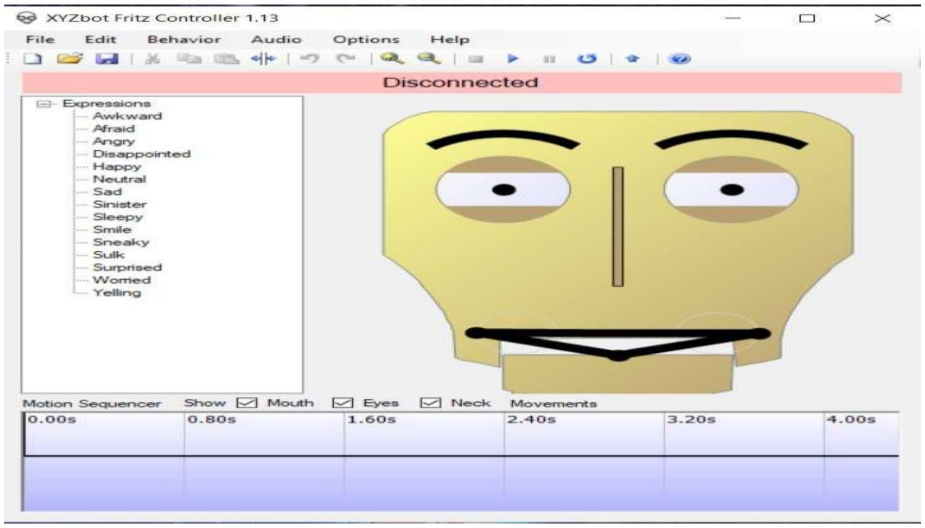
2.1. Hardware purchases

https://kerkits.com/products/fritz-the-robotic-head - Contains all parts for fritz robot head, Arduino UNO, Servo Shield, and battery attachments.
You will need additional servo motors, constant 6V power batteries or USB powered device, glue gun, extra sets of screws and a few jumper wires.

2.2. Head build and testing

To assemble the robot head, you'll need to follow instructions posted by kerkits. The instruction link is provided here: https://kerkits.com/pages/fritz-support-page-assembly-software-installation-etc

After completing this, interface all the servos to the arduino and run the test to check whether all servos are working correctly. After ensuring all the servos are working, you need to install software provided by kerkits, in order to check whether all emotions are working correctly. You'll find the attachments in the above link. The software GUI looks like the one below.



2.3. Hardware constraints and solution

Fritz faces a significant challenge in terms of its power consumption. The 4.5V power kit provided by Kerkits is insufficient to power the servo motors, which draw more power and frequently require replacement. To address this issue, there are a couple of solutions available. One option is to use a power supply from the laboratory that provides a constant 6V. Another solution is to use rechargeable cells to ensure a continuous power supply.

Connecting the moving parts of Fritz can be challenging due to the compact spaces involved, particularly when it comes to the connections for the eyeballs and eyelids. Losing connection during motor rotation can be a difficult issue to address, requiring the opening of joint connections just to reconnect them. Moreover, many parts are joined using a hot glue gun, which makes rebuilding connections impossible. To avoid this, an alternative is to screw the parts instead of gluing them, providing flexibility to detach all items quickly.The Arduino to PC serial communication wire presents another hardware flaw, as it obstructs the vertical motion of the neck. To address this issue, a USB to TTL converter can be used to eliminate the serial wire, allowing for greater flexibility in neck movement.

2.4. Interfacing with Arduino

To establish a connection between the computer and the Arduino, a serial communication cable is used. The Arduino Uno is equipped with a motor shield that facilitates the direct connection of servo motors to the board's analog and digital pins. This eliminates the need for a breadboard. Additionally, the motor shield provides a power source of 5V to power the motors and other external components. The table below shows the corresponding control pins on the Arduino board for each facial part connected to a motor.

Face part	Arduino Pin
Left Eyebrow	D2
Right Eyebrow	D3
Left Eyelid	D4
Right Eyelid	D5
Horizontal Eye Left	D6
Horizontal Eye Right	D7
Vertical Eye Left	A0
Vertical Eye Right	D9
Left Lip	D10
Right Lip	D11
Bottom Lip	D12
Neck Horizontal	D13
Neck Vertical	D8

The current configuration employs a laboratory-provided variable power adapter instead of a battery connector. Emotion detection is a critical task for various companies to comprehend how their consumers respond to their launched products. It can also be utilized to determine whether employees are content with the facilities provided to them. Additionally, it has numerous other applications, such as determining a person's mood from a distance using a camera for detection. Moreover, the same algorithm can be slightly modified and applied in other areas, such as face detection, attendance systems, mask detection, and more.

3. ChatGPT - A new way to ask questions

As the world continues to progress, so does technology. With the advent of ChatGPT, asking questions has become easier and more efficient. ChatGPT is a large language model trained by OpenAI based on the GPT-3.5 architecture. It has been designed to provide conversational responses to user queries, making it easier for people to ask questions and receive quick, informative answers.

One of the major benefits of ChatGPT is that it is available 24/7, making it easier for users to ask questions at any time, regardless of their location. Unlike human experts, ChatGPT can provide answers to a wide range of topics without any bias, making it an ideal platform for those looking for unbiased information. Additionally, ChatGPT can provide answers to even the most difficult and complex questions, making it an invaluable tool for researchers, students, and professionals alike.

Another major benefit of ChatGPT is its ability to understand natural language, which makes it easier for users to ask questions in their own words. ChatGPT can understand and respond to questions in a variety of languages, making it accessible to people from different parts of the world. Additionally, the responses provided by ChatGPT are usually more accurate and detailed than those provided by other search engines or automated systems, making it an ideal choice for those looking for more in-depth information.

ChatGPT can also be used to supplement traditional learning methods. Students can use ChatGPT to ask questions about topics that they are studying in school, and receive quick, informative answers that can help them better understand the subject matter. Professionals can also use ChatGPT to gain insights into their respective industries, making it easier for them to stay updated on the latest trends and developments.

Another major benefit of ChatGPT is its ability to learn from its interactions with users. As more and more people use the platform, ChatGPT becomes more intelligent and better equipped to handle a wider range of questions. This means that over time, ChatGPT will become an even more valuable resource for those looking to ask questions and receive informative answers.

In conclusion, ChatGPT is a new way to ask questions that offers many benefits. It is available 24/7, understands natural language, provides detailed and accurate answers, can be used to supplement traditional learning methods, and becomes more intelligent over time. Whether you are a student, researcher, or professional, ChatGPT is a valuable tool that can help you gain insights and information on a wide range of topics.

3.1. Why do we need it?

ChatGPT can be a useful tool for robot speech because it is a large language model that has been trained on a vast amount of text data, which allows it to generate human-like responses to questions and prompts. When integrated with a robot, ChatGPT can enable the robot to interact with humans in a more natural and effective way.

Using ChatGPT for robot speech can also help to improve the quality and consistency of the robot's speech. Unlike traditional pre-programmed responses, ChatGPT can generate unique responses to specific questions or prompts, making the robot's speech more engaging and informative.

Another advantage of using ChatGPT for robot speech is that it can enable the robot to learn and adapt over time. As the robot interacts with more people and receives feedback, it can refine its responses and improve its ability to understand and communicate with humans.

Overall, ChatGPT can be a powerful tool for enhancing the speech capabilities of robots and enabling them to interact with humans in a more natural and effective way.

4. Implementation speech-to-text and ChatGPT interface using python

4.1 Imports and initial setup

```
import pyfirmata
import time
import speech_recognition as sr
import pyttsx3
import serial
import openai
import threading
```

`import pyfirmata`: This imports the **`pyfirmata`** library, which allows us to communicate with Arduino boards using the Firmata protocol.
`import time`: This imports the **`time`** library, which provides various time-related functions.
`import speech_recognition as sr`: This imports the **`speech_recognition`** library and aliases it as **`sr`**. This library provides access to various speech recognition engines, including Google Speech Recognition.
`import pyttsx3`: This imports the **`pyttsx3`** library, which provides a cross-platform interface for text-to-speech conversion.
`import serial`: This imports the **`serial`** library, which provides access to serial ports.
`import openai`: This imports the **`openai`** library, which allows us to interface with OpenAI's GPT-3 API.
`import threading`: This imports the **`threading`** library, which provides support for multithreaded programming.

```
# Set up OpenAI API credentials
openai.api_key = "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"
```

```
# Initialize speech recognition and synthesis engines
r = sr.Recognizer()
engine = pyttsx3.init()


# Set speech rate to 80 percent of original rate
engine.setProperty('rate', engine.getProperty('rate') * 0.8)


# Open serial communication with Arduino
arduino = serial.Serial('COM7', 9600)
time.sleep(2)  # Wait for the connection to establish
```

The code sets up the API credentials for OpenAI, a company that specializes in artificial intelligence. It then initializes the speech recognition and synthesis engines, which allow the program to understand and speak. The speech rate is set to 80% of the original rate. The code also opens a serial communication with an Arduino board, which is a microcontroller board that allows the program to send and receive data to and from sensors and other devices. Finally, the code waits for two seconds to ensure that the connection is established before proceeding with the rest of the program.

4.2 Main logic

```
while True:
    # Prompt the user to choose between voice input and typing input
    print("Choose input method:")
    print("1. Voice")
    print("2. Typing")
    input_method = input()


    if input_method == "1":
        # Record and process speech input
        with sr.Microphone() as source:
            print("Say something!")
            audio = r.listen(source)
        try:
            text_input = r.recognize_google(audio)
            print("You said: " + text_input)
        except sr.UnknownValueError:
            print("Sorry, I could not understand your speech.")
            continue
        except sr.RequestError as e:
            print("Sorry, could not request results from Google Speech Recognition 
service: {0}".format(e))
```

```
            continue


    elif input_method == "2":
        # Get input from console
        text_input = input("Type a question: ")


    elif input_method == "exit":
        print("Exiting...")
        break


    else:
        print("Invalid input method.")
        continue


    if text_input == "exit":
        print("Exiting...")
        break
    elif text_input == "fear":
        engine.stop()
        arduino.write("1".encode())
    elif text_input == "neutral":
        engine.stop()
        arduino.write("2".encode())
    elif text_input == "sad":
        engine.stop()
        arduino.write("3".encode())
    elif text_input == "angry":
        engine.stop()
        arduino.write("4".encode())
    elif text_input == "surprised":
        engine.stop()
        arduino.write("5".encode())
    elif text_input == "happy":
        engine.stop()
        arduino.write("6".encode())
    elif text_input == "pause":
        engine.stop()
        print("Speech paused.")
    elif text_input == "interrupt":
        print("Interrupting and restarting...")
        break
```

This is a code block that creates an infinite loop which prompts the user to choose between two input methods: voice or typing. If the user chooses voice, the program uses the **sr.Microphone()** function from the SpeechRecognition library to record and process speech input. If the user chooses to type, the program waits for input from the console.

The program then checks the input to see if it matches any of the preset emotion keywords ("fear", "neutral", "sad", "angry", "surprised", or "happy"). If there is a match, the program sends a signal to an Arduino board connected to the computer via a serial port. The signal corresponds to a specific facial expression and the Arduino board controls a set of LEDs that illuminate the expression.
If the input is "pause", the speech engine is stopped, and if the input is "interrupt", the program stops the loop and restarts from the beginning. The loop continues until the user inputs "exit", at which point the program exits.

4.3 Talking Fritz

```
else:
        # Feed text to GPT-3 and generate response
        prompt = text_input
        model = "text-davinci-002"
        response = openai.Completion.create(
            engine=model,
            prompt=prompt,
            max_tokens=50
        )
        text_output = response.choices[0].text
        estimated_time = float(len(text_output.split()) / 2.5) * 1.25 # Update 
estimated time
        print(f"Estimated speech time: {estimated_time} seconds.")
        print("GPT-3: " + text_output)
```

This part of the code takes the user's input (text_input), sends it to OpenAI's GPT-3 model for processing and generates a response. The input prompt for GPT-3 is set as the user's input text. The GPT-3 model is specified as "text-davinci-002", which is a language model capable of generating high-quality text in response to various prompts.The max_tokens parameter specifies the maximum number of tokens (words and punctuations) that GPT-3 should output in its response. In this case, it is set to 50.

The response from GPT-3 is then retrieved and stored in the variable text_output. The estimated speech time for the response is calculated by dividing the number of words in the response by the average reading speed (2.5 words per second) and multiplying the result by 1.25 to account for pauses and other factors. Finally, the response is printed on the console with the prefix "GPT-3: ".

4.4. Python to Arduino talking

```
# Send the estimated speech time to Arduino
        arduino.write("{:.1f}".format(estimated_time).encode())
        # Convert response to voice signal and play it back
        time.sleep(2)
        engine.say(text_output)
        engine.runAndWait()


# Clean up
engine.stop()
```

After generating a response using OpenAI's GPT-3, the code sends the estimated speech time to the Arduino board by writing it to the serial port using the **write()** function. The estimated speech time is converted to a string with one decimal place using the **format()** method and then encoded using the **encode()** method before sending it to the Arduino.

Next, the code waits for 2 seconds to allow the Arduino to receive the estimated speech time before converting the response text to a voice signal using the **say()** method of the **engine** object. The **runAndWait()** method is then called to play back the voice signal. This will cause the response to be spoken out loud.

Finally, the **stop()** method of the **engine** object is called to clean up any resources used by the text-to-speech engine.

The control is passed to the Arduino for servo actuations

Arduino code:

```
#include <Servo.h>
// Set up servo motor and pin
Servo servo_lefteyebrow;
Servo servo_righteyebrow;
Servo servo_leftLip;
Servo servo_rightLip;
Servo servo_verticalEye_left;
Servo servo_verticalEye_right;
Servo servo_horizontalEye_left;
Servo servo_horizontalEye_right;
Servo servo_lefteyelid;
Servo servo_righteyelid;
Servo servo_jaw;
Servo servo_nod;
```

```
void setup() {
  // Start serial communication with Python
  Serial.begin(9600);


      // Attach servo motor to pin
      servo_lefteyebrow.attach(2);
      servo_righteyebrow.attach(3);


      servo_leftLip.attach(10);
      servo_rightLip.attach(11);


      servo_verticalEye_left.attach(8);
      servo_verticalEye_right.attach(9);


      servo_horizontalEye_left.attach(6);
      servo_horizontalEye_right.attach(7);

      servo_lefteyelid.attach(4);
      servo_righteyelid.attach(5);

      servo_jaw.attach(12);
      servo_nod.attach(13);

}


void loop() {
  // Check if there is any incoming serial data
  if (Serial.available() > 0) {
    // Read the incoming data as a string
    String input = Serial.readString();


    // Move the servo motor based on the input
    // FEAR FACE
    if (input == "1") {
      servo_lefteyebrow.write(70);
      servo_righteyebrow.write(120);


      servo_leftLip.write(60);
      servo_rightLip.write(140);


      servo_verticalEye_left.write(80);
      servo_verticalEye_right.write(140);


      servo_horizontalEye_left.write(108);
      servo_horizontalEye_right.write(40);

      servo_lefteyelid.write(80);
      servo_righteyelid.write(80);

      servo_jaw.write(55);
      servo_nod.write(50);

    }
      //  NEUTRAL FACE
      else if (input == "2") {
      servo_lefteyebrow.write(90);
      servo_righteyebrow.write(90);


      servo_leftLip.write(80);
      servo_rightLip.write(90);


      servo_verticalEye_left.write(90);
      servo_verticalEye_right.write(130);


      servo_horizontalEye_left.write(80);
      servo_horizontalEye_right.write(80);

      servo_lefteyelid.write(50);
      servo_righteyelid.write(80);

      servo_jaw.write(100);
      servo_nod.write(90);

    } //  SAD FACE
      else if (input == "3") {
      servo_lefteyebrow.write(60);
      servo_righteyebrow.write(130);
      servo_leftLip.write(140);
      servo_rightLip.write(40);


      servo_verticalEye_left.write(90);
      servo_verticalEye_right.write(130);


      servo_horizontalEye_left.write(108);
      servo_horizontalEye_right.write(80);

      servo_lefteyelid.write(90);
      servo_righteyelid.write(10);

      servo_jaw.write(100);
      servo_nod.write(110);
    }
      //  ANGRY FACE
      else if (input == "4") {
      servo_lefteyebrow.write(120);
      servo_righteyebrow.write(70);


      servo_leftLip.write(140);
      servo_rightLip.write(40);


      servo_verticalEye_left.write(90);
      servo_verticalEye_right.write(130);


      servo_horizontalEye_left.write(80);
      servo_horizontalEye_right.write(80);

      servo_lefteyelid.write(100);
      servo_righteyelid.write(10);

      servo_jaw.write(100);
      servo_nod.write(70);
    }
      // SURPRISED FACE
      else if (input == "5") {
      servo_lefteyebrow.write(120);
      servo_righteyebrow.write(70);
      servo_leftLip.write(140);
      servo_rightLip.write(40);


      servo_verticalEye_left.write(110);
      servo_verticalEye_right.write(90);


      servo_horizontalEye_left.write(80);
      servo_horizontalEye_right.write(80);

      servo_lefteyelid.write(70);
      servo_righteyelid.write(70);

      servo_jaw.write(30);
      servo_nod.write(30);
    }
      // HAPPY FACE
      else if (input == "6") {
      servo_lefteyebrow.write(120);
      servo_righteyebrow.write(70);


      servo_leftLip.write(140);
      servo_rightLip.write(40);


      servo_verticalEye_left.write(90);
      servo_verticalEye_right.write(130);


      servo_horizontalEye_left.write(80);
      servo_horizontalEye_right.write(80);

      servo_lefteyelid.write(20);
      servo_righteyelid.write(0);

      servo_jaw.write(70);
      for(int i = 0; i < 4; i++)
      {
        servo_nod.write(90);
        delay(200);
        servo_nod.write(70);
      }
    } else {
      // Convert the string to a float
```

```
        float estimated_time = input.toFloat();
        Serial.print("Received estimated speech time: ");
        Serial.println(estimated_time);
        Serial.println(" seconds.");


        estimated_time = ((estimated_time * 2));
        // Move servo for the estimated time
        servo_lefteyebrow.write(90);
        servo_righteyebrow.write(90);


        servo_leftLip.write(80);
        servo_rightLip.write(90);


        servo_verticalEye_left.write(90);
        servo_verticalEye_right.write(130);


        servo_horizontalEye_left.write(80);
        servo_horizontalEye_right.write(80);

        servo_lefteyelid.write(50);
        servo_righteyelid.write(80);

        servo_jaw.write(100);
        servo_nod.write(90);

        delay(1000);
        for (int i = 0; i < estimated_time; i++) {
          servo_jaw.write(70);  // move servo to maximum position
          delay(200);
          servo_jaw.write(90);    // move servo to minimum position
          delay(200);
        }
      }
    }
  }
```

This code is used to control the movement of servo motors in a robot based on inputs received from a Python program. The robot is designed to replicate different emotions depending on the input received. The code uses the Servo library to control the servo motors and sets up 11 servo motors, each attached to a specific pin on the Arduino board.

The code uses a loop to constantly check for incoming serial data from the Python program. If data is available, the input is read as a string and compared to different values to determine which emotion to replicate. Depending on the emotion, the code sets the position of each servo motor using the write() function. There are 6 different emotions that can be replicated: fear, neutral, sad, angry, surprised, and happy. The code also includes a default case for when an invalid input is received.

5. Hardware Troubleshooting

5.1. Power Connection Problem

There is a power connection problem that connects the power adapter and Arduino motor shield. This might be due to improper contact with wire and motor shield. A suggestion would be to replace the power adapter or Arduino motor shield. Checking if there are any loose soldered power contacts, fixing it might also solve the problem.

Motor and moving part connection problem:
If any DOF moves to extreme position, there is a possibility for the parts to break. If it breaks, it is difficult to replace these parts as casing parts protecting these parts are joined with hot glue. Therefore, it is best to rotate the motors within safe limits. As of now, the left eye vertical motion connector is disconnected. And its contact is difficult to access and therefore I was not able to rejoin the motor and eye part.

6. Software troubleshooting

One potential issue in this implementation is that there is no error handling for the Arduino serial communication. If there is an issue with the serial connection, the program will continue to try to send data to Arduino, leading to errors and potentially crashing the program. It would be a good idea to include error handling for the serial communication to avoid this issue.

Another issue is that the program does not handle interruptions or errors in the OpenAI API calls. If there is an interruption or error in the API call, the program will crash. It would be a good idea to include error handling for the API calls to avoid this issue and provide more robustness to the program.

Additionally, the program uses an infinite loop to continuously prompt the user for input. This can be inefficient and may cause the program to consume unnecessary resources. A better approach would be to use an event-driven model, where the program waits for user input or other events and only processes input when it is received. This would make the program more efficient and responsive.

7. Conclusion

In this project, we have developed a system that uses voice recognition, OpenAI's GPT-3, and an Arduino board to control servo motors based on user input. The system can recognize speech input from the user, convert it into text, and process it using GPT-3 to generate a response. The response is then converted into voice output, and an Arduino board controls the servo motors based on the response. The system has been successfully implemented and tested, and it can be used for various applications such as controlling robotic arms, home automation, and other IoT-based applications.

It was a great learning experience working with Fritz. I understood how the software in today's world can recognize human emotions with predefined algorithms. The interfacing between hardware and software was very crucial. To do such a project in real-time is tough wherein you face constraints from both hardware and software. Hardware knowledge applied with coding skills makes this project a full-fledged success.

7.1. Future Scope

There is a lot of potential for future improvements and advancements in this project. Here are a few ideas for future scope:

1. Incorporating more advanced speech recognition technologies such as deep learning algorithms to improve the accuracy of speech recognition.
2. Expanding the system to control multiple servo motors simultaneously, which can be useful in applications such as robotic arms.
3. Integrating machine learning algorithms to learn from user interactions and generate more accurate responses over time.
4. Developing a mobile application to control the system remotely, enabling users to control their devices using their smartphones.
5. Adding natural language processing (NLP) capabilities to the system to understand user intent and provide more relevant responses.

Overall, this project has the potential for further improvements and can be extended to various applications in the field of IoT, robotics, and automation.