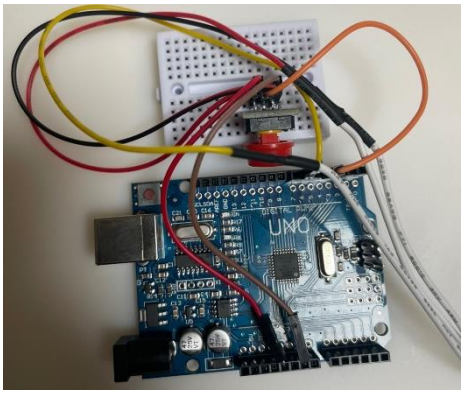
AI 스피커 만들기

우리는 현재 대인공지능의 시대를 맞이하고 있다. 과거 그 어느 때에도 인공지능이 우리의 삶에 이렇게 많은 영향을 끼친 적은 없었다. 나 또한 인공지능에 대해 굉장히 많은 관심을 가지고 있다. 나는 우리 주위에서 비교적 쉽게 인공지능을 사용할 수 있는 방법 중 하나인 AI 스피커를 만들어보고 싶었고, 직접 한번 만들어보려고 한다.

1. 준비과정

AI스피커를 만드는 과정은 크게 소프트웨어와 하드웨어 이 두가지로 나눌 수 있다. 우선 하드웨어 준비과정부터 살펴보자.

- 하드웨어



저렴한 방법으로 하드웨어를 구성할 수 있는 방법은 아두이노를 사용하는 것이다. 하지만 이 마저도 돈이 아까워 다른방법을 고민하던 찰나, 때마침 작년 동아리 활동으로 만들었던 작품에 사용된 아두이노가 집에 남아 있으니 이걸 사용해보도록 하자. 그리고 또 하나 필요한 것이 버튼이다. 버튼을 누르면 AI가 사용자의 말을 들을 준비가 되는 방식으로 작동이 되어야 한다. 때마침 남는 아두이노에 버튼 또한 남아있기에 나이스다. 이걸 그대로 사용하도록 하자. 그렇게 하드웨어를 구성할 부품을 모으고 있던 찰나 갑자기 문득 생각이 들었다.

이 액자.. 생각보다 이쁘다. 이렇게 열심히 만든 액자를 굳이 분해해야할까? 흔해 빠진 AI스피커 보다도 차라리 액자모양을 한 AI스피커가 인테리어적으로도 더 낫지 않을까? 나는 기존에 없던 새로운 AI스피커를 만들어보기로 했다. 그 이름은 바로 액자형 AI 스피커!!! 그렇게 나는 이 액자를 그대로 사용하기로 했다. 소리나 나야하는 스피커와 사용자의 말을 인식하는데 쓰일 마이크는 아두이노용으로 가지고 있는 것이 없기 때문에 컴퓨터 스피커와 마이크로 대체하면 하드웨어 준비는 끝이 나게 된다. 다음 장에서는 소프트웨어 준비 과정에 대해 소개하겠다.

- 소프트웨어

소프트웨어에서 가장 우선적으로 정해야 하는 것은 어떤 인공지능을 사용하느냐이다. 물론 처음부터 내가 수집한 데이터들로 학습을 진행한 인공지능 모델을 사용하여 AI 스피커를 만들면 정말 좋겠지만, 나는 아직 고등학생이고 그런 인공지능을 개발할 환경도 없는데다 비용또한 감당하지 못한다. 그래서 내가 선택한 것은 바로 대인공지능의 시대를 연 장본인 중 하나인 GPT의 힘을 빌리는 것이다. AI스피커 속 인공지능이 대화형 인공지능이 주를 이루는 만큼 OpenAI에서 제공하는 GPT-3.5-turbo라는 챗봇 모델 API를 사용하여 AI 스피커 속에 인공지능을 집어넣을 것이다. 또한 사람의 말을 인식하는 작업인 STT(Speech To Text)에는 구글에서 제공하는 Speech API를 사용할 수 있는 Python의 Speech Recognition이라는 라이브러리를 사용할 것이다. 인공지능이 답변한 문장을 소리 내어 읽을 수 있게 하는 TTS(Text To Speech)역시 gTTS라는 구글의 TTS API를 사용하여 구현할 것이다. 이렇게 기본적인 소프트웨어 구성을 정했다. 그렇다면 이제는 이들을 직접 코딩할 시간이다.

```
import speech_recognition as sr
import os
import openai
from gtts import gTTS
import playsound
import serial
from Arduino import Arduino

ser = serial.Serial(
    port='COM3',
    baudrate=9600,
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
) #아두이노 보드와 통신 설정

openai.api_key = "(블러처리 - API 키는 외부에 공개하면 위험합니다)" #GPT-3.5-turbo 를 사용하기 위한 API Key


def AI_speaker():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("무엇을 도와드릴까요?")
        tts = gTTS(text='무엇을 도와드릴까요?', lang='ko') #한국어로 '무엇을 도와드릴까요?'라고 말하는 TTS 작업
        Hello_file = "Hello.mp3"
        tts.save(Hello_file) #생성된 TTS 파일을 mp3 형태로 저장
        playsound.playsound(Hello_file) #저장한 mp3 출력하기
        os.remove(Hello_file)
        audio = r.listen(source)

    try:
        command = r.recognize_google(audio, language='ko-KR') #사용자의 말을 한국어로 인식하여 Command 라는 변수에 저장

        print("You said: " + command) #내가 한 말이 인식된 결과를 출력
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio") #제대로 인식이 되지 않았을 경우 오류가 발생했다고 출력
    except sr.RequestError as e:
```

```
        print("Could not request results from Google Speech Recognition service;
{0}".format(e)) #그 밖의 오류 출력

    messages = [
        {"role": "system", "content": str(command)},
    ]

    message = str(command)
    if message: #사용자가 한 말을 저장
        messages.append(
            {"role": "user", "content": message},
        )
        chat = openai.ChatCompletion.create(
            model="gpt-3.5-turbo", messages=messages #gpt-3.5-turbo 모델 불러오기
        ) #gpt 가 대답을 생성

    reply = chat.choices[0].message.content #생성한 대답을 reply 에 저장
    print(f"ChatGPT: {reply}") #생성된 대답 출력하기
    messages.append({"role": "assistant", "content": reply})
    reply_tts = gTTS(text=reply, lang='ko') #생성된 대답을 TTS 작업
    reply_file = "reply.mp3"
    reply_tts.save(reply_file) #작업된 것 mp3 파일로 저장
    playsound.playsound(reply_file) #생성된 mp3 파일 출력
    os.remove(reply_file)



while True:
    if ser.readable():
        res = ser.readline()
        print(res)
        ready = res.decode()[:len(res)-2] # bytes 를 decode 해서 str 로 변환, 자동으로 포함되어있는
개행을 제거하기 위해 [:len(res)-2] 를 추가함 : 즉, \r\n 을 제거함.

        print(ready) # 아두이노(PLC)에서 날라온 프로토콜.

        if ready == 'ready': #아두이노에서 준비완료 신호가 오면 AI_speaker 라는 위의 함수 실행
            AI_speaker()
        else:
            pass
```

이것이 AI스피커 속 인공지능이 실행되는 코드이다. 아래에는 아두이노에 들어가는 코드이다.

```
#include <Adafruit_NeoPixel.h>

const int buttonPin1 = 4; //  4 번핀 스위치 입력 테스트 (PLC 의 ready 신호)
#define LED_PIN 6     //네오픽셀에 신호를 줄 핀번호
#define LED_COUNT 60  //아두이노에 연결된 네오픽셀의 개수
int flag = 0;

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN);

void setup() {
```

```
  Serial.begin(9600);              //  시리얼 통신의 시작, 보레이트 입력
  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // 네오픽셀에 빛을 출력하기 위한 것인데 여기서는 모든 네오픽셀을
OFF 하기 위해서 사용한다.
  strip.setBrightness(50); // 네오픽셀의 밝기 설정(최대 255 까지 가능)
}

void rainbow(int wait) {
  for (long firstPixelHue = 0; firstPixelHue < 5 * 65536; firstPixelHue += 256) {
    for (int i = 0; i < strip.numPixels(); i++) {
      int pixelHue = firstPixelHue + (i * 65536L / strip.numPixels());
      strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(pixelHue)));
    }
    strip.show();
    delay(wait);
  }
}

void loop() {
  char c = Serial.read();           //  PC 로부터온 값을 읽음.
  int buttonValue1 = digitalRead(4); //  4 번 핀에서 읽음.

  if(buttonValue1 == HIGH )           //  스위치가 눌릴때 ready 신호를 보냄. (PLC -> PC)
  {
    flag = 1;
  }
  if(flag == 1)
  {
      Serial.println("ready");  //  준비가 되었다고 PC 에 신호를 보냄.
      rainbow(10);
      delay(500);
      flag = 0;
  }
}
```

이렇게 아두이노에서 버튼입력이 되면 첫번째 파이썬 코드에 신호를 주게 되고, 해당 신호를 받게 되면 챗봇 인공지능이 작동하는 원리이다. 물론 코드를 짜면서 모든게 순탄했던 것은 아니다. 또한 인공지능이 가끔씩 멍청한 소리를 하는 경우도 있었다. **아래에는 내가 개발과정 중 겪은 문제들이다.**

1. 최근 공개된 gpt-4.0과는 다르게 gpt-3.5-turbo는 학습 데이터의 시점도 2021년에 멈춰있는데다 가끔씩 거짓말도 한다. 프로그램을 테스트하던 도중 넌 누구야?라고 물어보니 '저는 Google의 인공지능 어시스턴트입니다.' 라는 어이없는 거짓말을 한 적도 있었다. 물론 gpt-4.0을 사용하면 해결 될 일이지만 gpt-4.0을 사용하기 위해서는 돈을 내야하기에 어쩔 수 없이 gpt-3.5-turbo로 만족해야했다.
2. 답변을 생성하고 그 생성된 답변을 소리로 출력하는데 시간이 오래걸린다. 이것 또한 gpt-3.5-turbo이 느리기 때문에 발생하는 일이다. 또한 생성된 답변을 TTS처리하는데도 꽤 시간이 걸린 것으로 예상된다. Mp3파일을 생성하고 이를 출력하는 과정을 좀 더 최적화 할 필요가 있다.
3. AI의 답변의 길이가 일관되지 못한다. 어떤 경우에는 답변이 너무 짧고, 어떤 경우에는 답변이 너무 길어 다음 질문을 하기까지 한참이 걸린다. 이 문제는 gpt-3.5-turbo에 max_token을 설정하면 해결할 수 있는 문제다.

프로젝트를 마무리하며…

이번 프로젝트에서는 어쩌면 다들 한번씩은 사용해 봤을 AI 스피커를 만들어보았다. 내 입맛대로 내가 원하는 대로 나만의 AI 스피커를 만드는 과정이 정말 재미있었다. 처음에는 많이 어려울 거라 생각했지만, 여러 API들의 도움 덕분에 비교적 간단하게 AI 스피커를 만들 수 있었다. 코딩을 하면서 오류가 발생하면 해당 오류에 대해 검색하고, 이를 통해 오류를 고쳐나가면서 프로그래밍 언어에 대한 공부 또한 가능했다. 내가 직접 만든 AI 스피커의 장점과 단점을 분석해보고 주변 사람들의 피드백도 수용하면서, 발견한 단점들을 어떻게 고쳐나가면 좋을지에 대해 다양한 시각에서 여러 번 생각해보았다. 이를 통해 내가 미처 생각하지 못했던 문제들도 발견할 수 있었고, 앞으로도 다양한 시각에서 문제들을 바라봐야겠다는 인식을 갖게 되었다.

https://youtu.be/1w_hDfeUkZQ 여기 링크에서 제가 만든 AI스피커를 시연하는 영상을 확인하실 수 있어요!