import pyaudio  
import queue  
from aip import AipSpeech  
import serial  
import modbus\_tk.defines as cst  
import modbus\_tk.modbus\_rtu as modbus\_rtu  
  
APP\_ID = '62991368'  
API\_KEY = 'VOlhsGpzWgK6UR78uaooix1z'  
SECRET\_KEY = 'QrlRKmd2iw8n8Pg4svTHSTRvJVqEgLHQ'  
  
FORMAT = pyaudio.paInt16 # 录音的格式  
CHANNELS = 1 # 音频录制的声道数 1单声道 2立体声  
RATE = 16000 # 音频采样率即一秒钟采样多少次  
CHUNK = 1024 # 定义每次读取音频数据的帧数  
RECORD\_SECONDS = 5 # 定义音频录制的持续时间 单位是秒  
RELAY\_PORT = 'COM7' # 指定串行端口的名称  
RELAY\_RESPONSE\_TIMEOUT = 5.0 # 设置Modbus RTU响应超过时间 如果继电器模块超过5s没有响应就会视为超时  
commands\_queue = queue.Queue() # 初始化一个线程安全的队列 用于在音频识别线程和命令执行线程之间传递识别出的命令  
is\_program\_running = True # 新增一个全局变量控制程序运行状态  
  
client = AipSpeech(APP\_ID, API\_KEY, SECRET\_KEY) # 创建一个百度语音识别服务的客户端实例  
  
  
def connect\_relay(port): # 接收参数port 参数表示连接的串行窗口  
 try: # 启用try模块 如果后面代码执行发生异常则会执行except块中的代码  
 master = modbus\_rtu.RtuMaster(serial.Serial(port=port, baudrate=9600, bytesize=8,  
 parity='E', stopbits=1))  
 master.set\_timeout(RELAY\_RESPONSE\_TIMEOUT)  
 return 1, master  
 except Exception as exc:  
 print("Error opening serial port: ", str(exc))  
 return -1, None  
  
  
def switch(master, action): # 控制继电器开关  
 try:  
 if "开" in action.lower():  
 master.execute(2, cst.WRITE\_SINGLE\_COIL, 0, output\_value=True)  
 elif "关" in action.lower():  
 master.execute(2, cst.WRITE\_SINGLE\_COIL, 0, output\_value=False)  
 return 1  
 except Exception as exc:  
 print(str(exc))  
 return -1  
  
  
def record\_audio(): # 录制音频  
 p = pyaudio.PyAudio()  
 stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True, frames\_per\_buffer=CHUNK)  
 print("开始录音，请说话...")  
 frames = [stream.read(CHUNK) for \_ in range(int(RATE / CHUNK \* RECORD\_SECONDS))]  
 print("录音结束")  
 stream.stop\_stream() # 调用了流的stop\_stream方法停止音频流  
 stream.close() # 关闭流，释放与流相关的资源  
 p.terminate() # 终止pyaudio会话  
 return b''.join(frames)  
  
  
def recognize\_audio(audio\_data): # 识别音频数据的语音并转换成文本  
 result = client.asr(audio\_data, 'pcm', 16000, {'dev\_pid': 1537}) # 调用asr方法进行语音识别 dev对应于一个特定的语音识别模型 1537指某个普通话模型  
 if 'result' in result and len(result['result']) > 0:  
 command = result['result'][0]  
 print(f"识别结果: {command}")  
 return command  
 else:  
 print("未能识别语音")  
 return ""  
  
  
def recognize\_realtime():  
 global is\_program\_running # 声明修改全局变量  
 while is\_program\_running: # 修改循环条件  
 audio\_data = record\_audio()  
 command = recognize\_audio(audio\_data)  
 if command:  
 if "退出程序" in command: # 检测到退出命令  
 print("接收到退出指令，程序将关闭。")  
 is\_program\_running = False # 修改状态，通知其他线程退出  
 break # 退出当前线程的循环  
 if "打开" in command or "关闭" in command or "风扇" in command:  
 commands\_queue.put(command)  
  
  
def execute\_commands():  
 global is\_program\_running # 也许并不需要在这个函数中修改全局变量，但需要检查它的状态  
 \_, relay\_master = connect\_relay(RELAY\_PORT)  
 while is\_program\_running: # 修改循环条件  
 if not commands\_queue.empty():  
 command = commands\_queue.get()  
 if command and ("打开" in command or "关闭" in command or "风扇" in command):  
 switch(relay\_master, command)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import threading  
  
 t1 = threading.Thread(target=recognize\_realtime)  
 t1.start()  
 t2 = threading.Thread(target=execute\_commands)  
 t2.start()