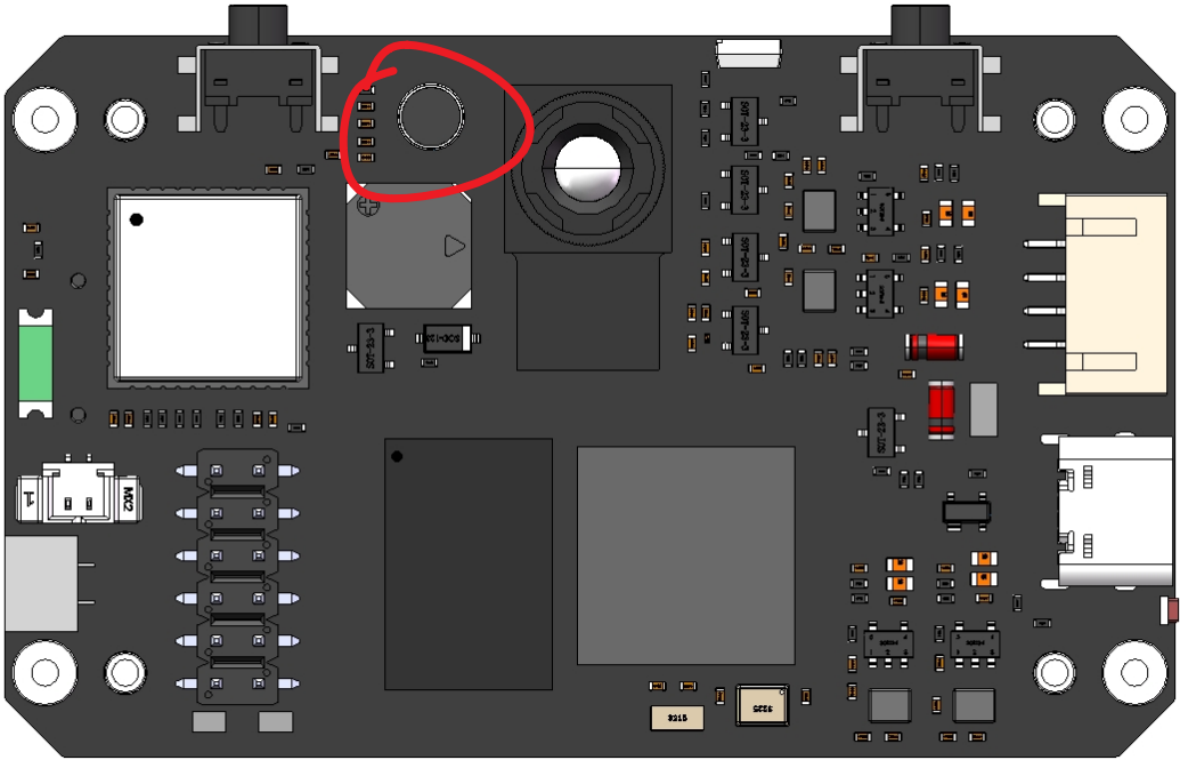
Audio Recording

Introduction to the results of routine experiments

In this section, we use the microphone on the K230 to record the sound.



The example code for this section is located in: [Source Code/10.Media/01.record_audio.py]

We use CanMV IDE to open the sample code and connect K230 to the computer via USB.

Click the Run button in the lower left corner of CanMV IDE, K230 will start recording 10 seconds of audio, and the recording results will be saved in

【/data/audio/audio.wav】(You can also modify it yourself)

Code Explanation

> In order to facilitate everyone to quickly call the audio recording function, we encapsulate the recording function into the AudioRecorder module

Complete code

```
import os
from media.media import *
from media.pyaudio import *
import media.wave as wave

class AudioRecorder:
    """
    音频录制类
    Audio recorder class
    """
    def __init__(self):
        """
        初始化音频参数
        Initialize audio parameters
        """
#         self.CHUNK = 44100 // 25    # 每个缓冲区的帧数 / Frames per buffer
        self.FORMAT = paInt16       # 采样格式为16位整型 / 16-bit integer sampling
format
        self.CHANNELS = 1           # 单声道 / Mono channel
        self.RATE = 44100          # 采样率44.1kHz / 44.1kHz sampling rate
#         self.RATE = 16000
        self.CHUNK = self.RATE // 25    # 每个缓冲区的帧数 / Frames per buffer
        self.frames = []            # 存储录音帧 / Store recorded frames
        self.is_recording = False   # 录音状态标志 / Recording status flag

        # 初始化PyAudio / Initialize PyAudio
        self.p = PyAudio()
        self.p.initialize(self.CHUNK)
        MediaManager.init()

    def exit_check(self):
        """
        检查是否有退出信号
        Check if there is an exit signal
        """
        try:
            os.exitpoint()
        except KeyboardInterrupt as e:
            print("user stop: ", e)
            return True
        return False

    def record(self, filename, duration, left_volume=85, right_volume=85,
ans=False):
        """
        录制音频
        Record audio

        参数 / Parameters:
            filename: 音频保存路径 / Path to save audio file
            duration: 录制时长(秒) / Recording duration in seconds
            left_volume: 左声道音量 / Left channel volume
            right_volume: 右声道音量 / Right channel volume
            ans: 是否启用音频3A功能：自动噪声抑制(ANS) / Open Ans or not
        """
```

```
        try:
            # 打开音频输入流 / Open audio input stream
            self.input_stream = self.p.open(
                format=self.FORMAT,
                channels=self.CHANNELS,
                rate=self.RATE,
                input=True,
                frames_per_buffer=self.CHUNK
            )

            # 设置音量 / Set volume
            self.input_stream.volume(LEFT, left_volume)
            self.input_stream.volume(RIGHT, right_volume)

            if(ans):
                self.input_stream.enable_audio3a(AUDIO_3A_ENABLE_ANS)

            print("start record...")
            self.frames = []
            self.is_recording = True

            # 开始录制 / Start recording
            for i in range(0, int(self.RATE / self.CHUNK * duration)):
                if not self.is_recording or self.exit_check():
                    break
                data = self.input_stream.read()
                self.frames.append(data)

            print("stop record...")

            # 保存为WAV文件 / Save as WAV file
            self._save_to_wav(filename)

        except BaseException as e:
            print(f"Exception {e}")
        finally:
            self.stop()

    def stop(self):
        """
        停止录音并清理资源
        Stop recording and cleanup resources
        """
        self.is_recording = False
        if hasattr(self, 'input_stream'):
            self.input_stream.stop_stream()
            self.input_stream.close()
        self.p.terminate()
        MediaManager.deinit()

    def _save_to_wav(self, filename):
        """
        保存录音为WAV文件
        Save recording as WAV file

        参数 / Parameters:
            filename: 保存路径 / Save path
        """
```

```
        wf = wave.open(filename, 'wb')
        wf.set_channels(self.CHANNELS)
        wf.set_sampwidth(self.p.get_sample_size(self.FORMAT))
        wf.set_framerate(self.RATE)
        wf.write_frames(b''.join(self.frames))
        wf.close()


# save image raw data, use 7yuv to preview

def ensure_dir(directory):
    """
    递归创建目录，适用于MicroPython环境
    """
    # 如果目录为空字符串或根目录，直接返回
    if not directory or directory == '/':
        return

    # 处理路径分隔符，确保使用标准格式
    directory = directory.rstrip('/')

    try:
        # 尝试获取目录状态，如果目录存在就直接返回
        print(os.stat(directory))
        print(f'目录已存在 The Folder already exists: {directory}')
        return
    except OSError:
        # 目录不存在，需要创建
        # 分割路径以获取父目录
        if '/' in directory:
            parent = directory[:directory.rindex('/')]
            if parent and parent != directory:  # 避免无限递归
                ensure_dir(parent)

        try:
            os.mkdir(directory)
            print(f'已创建目录 Created Folder: {directory}')
        except OSError as e:
            # 可能是并发创建导致的冲突，再次检查目录是否存在
            try:
                os.stat(directory)
                print(f'目录已被其他进程创建 The Directory has been created by
another process: {directory}')
            except:
                # 如果仍然不存在，则确实出错了
                print(f'创建目录时出错 An error occurred while creating the
directory: {e}')
    except Exception as e:
        print(f'处理目录时出错 An error occurred while processing the directory:
{e}')

# 使用示例 / Usage example
if __name__ == "__main__":
    os.exitpoint(os.EXITPOINT_ENABLE)
    save_path = "/data/audio/"
    # 目录不存在时创建 / create if not exist
    ensure_dir(save_path)
```

```
    recorder = AudioRecorder()
    print("音频录制开始 Recording start...")
    recorder.record(save_path + 'audio.wav', 10)
    print("音频录制完成 Recording complete...")
```

Code structure

Initialization section of the class:

- Set the audio parameters: 44.1kHz sampling rate, 16-bit sampling format, mono
- The buffer size is set to about 25 frames per second (44100/25)
- Initialize PyAudio and the media manager

Main methods:

- `exit_check()`: Detect exit signal and support user to interrupt recording
- `record()`: Core recording method
  - Open the audio input stream
  - Set the volume of left and right channels
  - Loop recording of audio data according to the set duration
  - Save recorded data as WAV file
- `stop()`: Stop recording and clean up resources
- `_save_to_wav()`: Save the recorded audio data as a WAV format file

Example usage:

- Create a recorder instance
- Record 10 seconds of audio
- Save to the specified path

This is an object-oriented audio recording implementation that provides complete recording process control, including:

- Recording parameter configuration
- Volume Control
- Exception handling
- Resource Management
- Save in WAV format