Meeting Cube

Embedded System Final Project - Group 5

B10901009 鄭煜儒 B10901098 蔡承恩 B10901173劉宏彧

GitHub Link

Youtube Link

I. Motivation & Introduction

In contemporary large-scale meetings or conferences, managing participant attendance, voting results, and speaking requests poses significant challenges for the moderator. Typically, these tasks require multiple devices or cumbersome processes, making it inefficient and time-consuming. To address these issues, we have developed a Smart Meeting Cube, a multifunctional device designed to streamline these activities for both the participants and the moderator.

The Smart Meeting Cube enables participants to perform key actions such as voting and requesting to speak by simply turning the cube to different orientations. Additionally, the cube incorporates facial recognition technology to verify the identity of each participant. The gathered information is then transmitted to the main system controlled by the moderator, integrating multiple functions into a single, user-friendly device. This innovation aims to enhance the efficiency and effectiveness of meeting management, facilitating a smoother and more organized process.

II. Function Overview

As outlined in the introduction, our system comprises two primary processes: one for the moderator and another for the participants.

1. **Moderator's Process:**

   The moderator's interface is a command-line application that facilitates various meeting management tasks. This interface allows the moderator to:

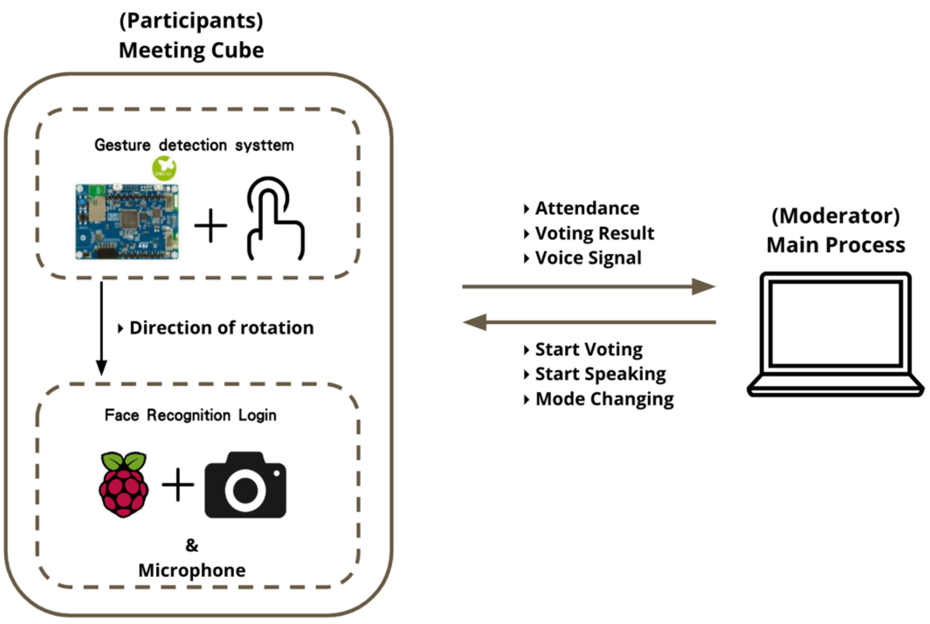
- Check participant attendance in real-time.
- Manage voting topics and view the voting results.
- Approve speaking requests from participants and control speaking permissions.

2. Participant's Meeting Cube:

The Smart Meeting Cube is an interactive device that participants use to perform key actions by rotating the cube to specific orientations:

- **Face Recognition for Attendance:** Participants rotate the cube to initiate face recognition, which verifies their identity and marks their attendance.
- **Voting:** During the voting session, participants can rotate the cube to the right or left to cast their votes.
- **Request to Speak:** Participants can rotate the cube to the front to request speaking permission. Once the moderator grants permission, the participant can speak into the cube, and the audio signal is transmitted to the moderator and broadcast through the sound system.

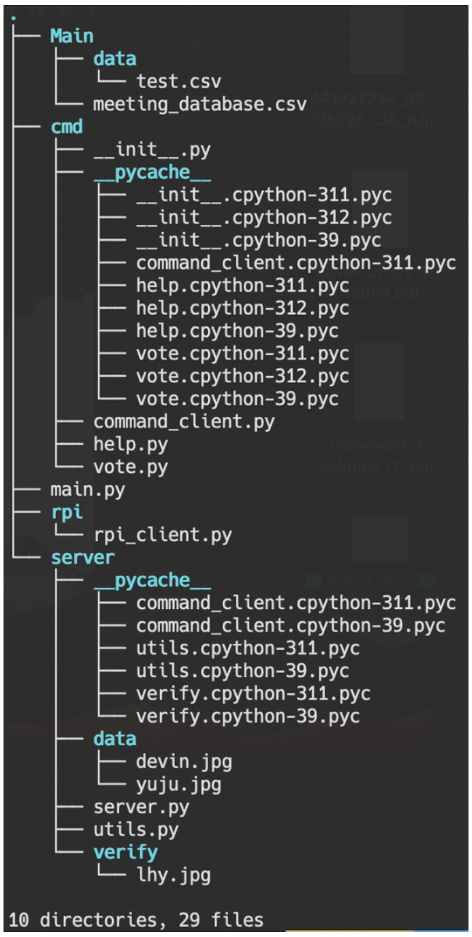
III. Flow Chart



We use rpi as a client to the server, it integrates messages from stm32 and rpi itself then sends them to the server. We use sockets to transmit messages to the server.

IV. Methods & Techniques

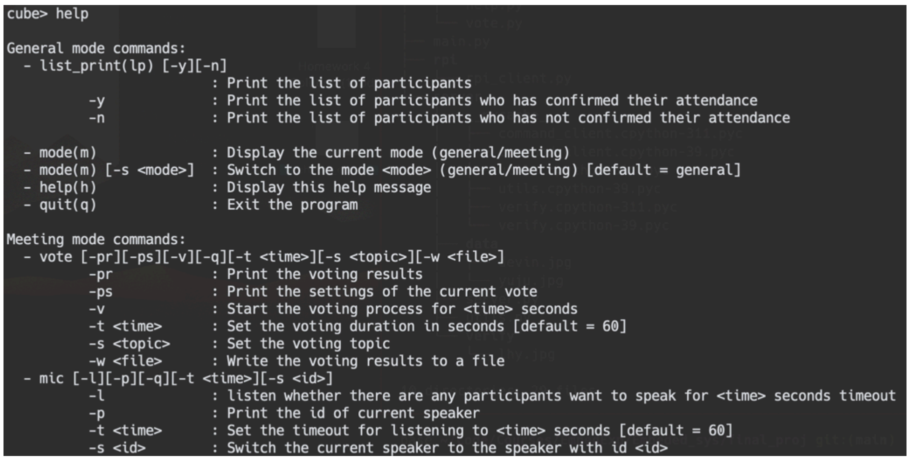
1. Code Structure



```
.
├── Main
│   ├── data
│   │   └── test.csv
│   └── meeting_database.csv
├── cmd
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-311.pyc
│   │   ├── __init__.cpython-312.pyc
│   │   ├── __init__.cpython-39.pyc
│   │   ├── command_client.cpython-311.pyc
│   │   ├── help.cpython-311.pyc
│   │   ├── help.cpython-312.pyc
│   │   ├── help.cpython-39.pyc
│   │   ├── vote.cpython-311.pyc
│   │   ├── vote.cpython-312.pyc
│   │   └── vote.cpython-39.pyc
│   ├── command_client.py
│   ├── help.py
│   └── vote.py
├── main.py
├── rpi
│   └── rpi_client.py
└── server
    ├── __pycache__
    │   ├── command_client.cpython-311.pyc
    │   ├── command_client.cpython-39.pyc
    │   ├── utils.cpython-311.pyc
    │   ├── utils.cpython-39.pyc
    │   ├── verify.cpython-311.pyc
    │   └── verify.cpython-39.pyc
    ├── data
    │   ├── devin.jpg
    │   └── yuju.jpg
    ├── server.py
    ├── utils.py
    └── verify
        └── lhy.jpg

10 directories, 29 files
```

2. Main Process for Moderator
   a. Command Line Interface



```
cube> help

General mode commands:
  - list_print(lp) [-y][-n]
                        : Print the list of participants
        -y              : Print the list of participants who has confirmed their attendance
        -n              : Print the list of participants who has not confirmed their attendance

  - mode(m)             : Display the current mode (general/meeting)
  - mode(m) [-s <mode>] : Switch to the mode <mode> (general/meeting) [default = general]
  - help(h)             : Display this help message
  - quit(q)             : Exit the program

Meeting mode commands:
  - vote [-pr][-ps][-v][-q][-t <time>][-s <topic>][-w <file>]
        -pr             : Print the voting results
        -ps             : Print the settings of the current vote
        -v              : Start the voting process for <time> seconds
        -t <time>       : Set the voting duration in seconds [default = 60]
        -s <topic>      : Set the voting topic
        -w <file>       : Write the voting results to a file
  - mic [-l][-p][-q][-t <time>][-s <id>]
        -l              : listen whether there are any participants want to speak for <time> seconds timeout
        -p              : Print the id of current speaker
        -t <time>       : Set the timeout for listening to <time> seconds [default = 60]
        -s <id>         : Switch the current speaker to the speaker with id <id>
```

We create a command line interface cube> such that moderators can interact with these commands. The function detail can be checked with command $help as the figure shown above.

Some important commands:

- **list_print**

  Moderator can check the .csv file that contains the name, attendance, voting result of each participant.

- **mode -s**
  Switch the mode between meeting/general.
- **vote [-tags]**
  Set the topic of the voting / Set the time for voting / Start voting, etc.
- **mic [-tags]**
  Set the time for speaking / Start receiving requests, etc.

b. Server
The server opens two ports separately for rpi and command interface, therefore we can gracefully handle different connections. Command interface control the state of meeting and server deal with different messages sent by rpi depends on the state, for example, when rpi ask for signin during meeting, the server marks the attendant as “arrive late”.

3. Functions in Meeting Cube

a. Cube orientation detection

- Detection
  We use STM32 to detect the orientation of the cube. In practice, we detect the gyro acceleration of the cube, which provides us with the moving direction of the cube.
- Communication
  After getting the spinning direction of the cube, we use WiFi sockets to send this message to RPI. RPI transform the message into corresponding instruction such as “Sign in”, “Speaking” and “Voting”

b. Face Recognition
The pipeline consists of two phases.

- Phase 1**:**
  We first take photos for attenders, which will be sent to the central controller for further processing.
- Phase 2**:**

After the central controller receives face images. These images are used to distill the characteristic vector of the faces. The vectors will compare with all the characteristic vectors in our database, resulting in final results.

c. Microphone

We use the pyaudio package sending audio to the server to allow real-time speakers. The audio is also transmitted through a web socket. When the buffer is filled, we send it to the server, and to accomplish low latency and clear audio, we modify the buffer size to achieve the best user experience.

d. Server/Client in RPI

In RPI, we listen to messages transmitted from stm32 to do the following operation, when stm32 detects the signin gesture, we then take the picture of the attendant and send the image to the server to verify. Similarlly, when asking for speaking, we send the request to the server. Lastly, detect and send corresponding voting messages.

V. Result

影片: https://www.youtube.com/watch?v=XhlPUvcbSuw

The demo video linked above demonstrates the entire process in a real-world meeting. The steps are as follows:

1. **Moderator Checks Attendance:**
   - The moderator reviews the attendance list to check which participants are expected for the meeting.
2. **Participants' Arrival and Face Recognition:**
   - Participants arrive at the meeting and use the Smart Meeting Cube for face recognition to mark their attendance.
3. **Moderator Verifies Attendance:**
   - The moderator checks the updated list and confirms that participants have attended the meeting.
4. **Setting Up Voting:**
   - The moderator sets the voting topic and defines the voting duration.
   - The voting session is started, allowing participants to cast their votes.
5. **Participants Cast Votes:**
   - Participants rotate the cube to indicate their vote (e.g., rotate right for "yes" and left for "no").

6. **Moderator Reviews Voting Results:**
   - The moderator checks the voting results after the session concludes.
7. **Handling Speaking Requests:**
   - The moderator sets the time for listening to speaking requests from participants.
8. **Participants Request to Speak:**
   - Participants rotate the cube forward to request permission to speak.
   - The moderator grants permission to speak.
9. **Speaking and Broadcasting:**
   - Participants speak into the cube.
   - The sound is broadcast from the moderator's computer to the meeting audience.

VI. Future Work

1. Beautiful GUI for moderator, instead of the current command line interface.
2. Monitor or other device for the participants to receive feedback from the Meeting Cube.
3. Include error handling in our process, equipping Meeting Cube with a fool proof mechanism.

VII. Reference

face recognition : https://github.com/ageitgey/face_recognition

Pi Camera : https://blog.wuct.me/raspberry-pi-100abbe7a1fd

pyaudio: https://ithelp.ithome.com.tw/m/articles/10291543