Final Project Report: GestureSync - STM32-Controlled YouTube Player

Group 6

- 林方瀚 (T13901108) : implement python code to control YouTube
- 徐御宸 (B10901158) : implement code with gesture recognition on STM32

Motivation

GestureSync was conceived to offer a novel, hands-free way to control music playback. Traditional touch interfaces can be inconvenient or inaccessible in various scenarios, such as during cooking, driving, or exercising. Our aim was to provide a solution that leverages natural physical movements to enable intuitive navigation of music playlists. By recognizing gestures and mapping them to playback controls, we envisioned a system that is not only innovative but also accessible to users with diverse needs. This approach represents a step forward in combining convenience and technology in everyday interactions.

Methods

The project involved the design and implementation of a system that integrates gesture recognition with music playback controls on YouTube. We utilized both hardware and software components to achieve this functionality.

Hardware

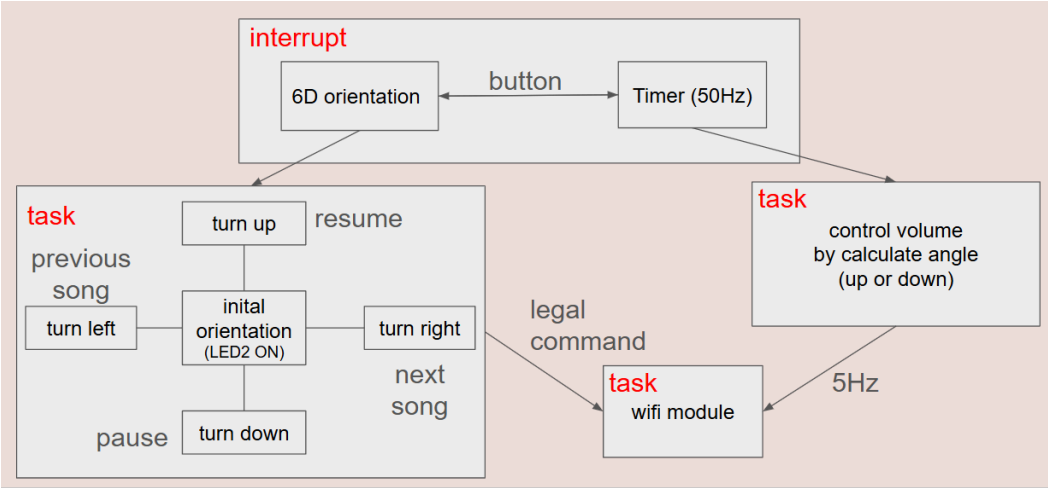
The core hardware setup included an STM32L4 series microcontroller, embedded with a LSM6DSL accelerometer and gyroscope sensor. LSM6DSL captured movement data, which was processed by the microcontroller to detect changes in orientation and calculate angles. A WiFi module was used for real-time communication with a computer system, enabling seamless command transmission.

The computer acted as a playback controller, equipped with a Python runtime and Google Chrome browser. Python scripts were employed to receive commands over a TCP socket and automate playback using Selenium WebDriver.

System Architecture

The system architecture was designed to ensure smooth interaction between hardware and software components. When the LSM6DSL detected gestures, the STM32 processed this data and generated corresponding playback commands. These commands were transmitted to the computer via a WiFi-based TCP socket. On the computer side, a Python script interpreted the commands and executed playback actions on YouTube, such as play/pause, volume control, and track navigation.

System Architecture in STM32 and gesture recognition



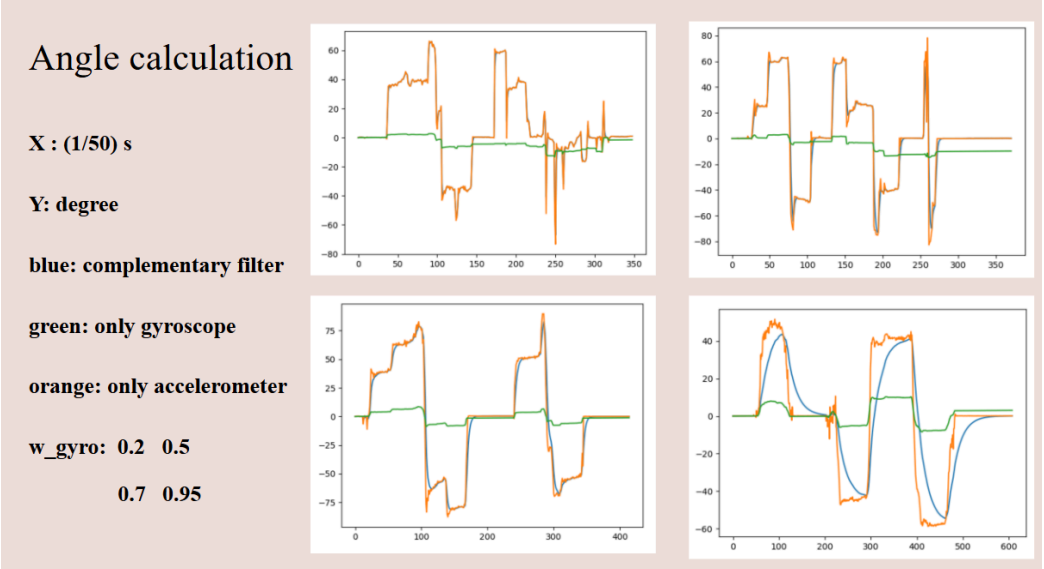
There are two different modes in STM32. Initial mode is simple command mode which enables the 6D orientation interrupt, another is volume control mode which turns on the timer and enables its interrupt. We can use the button to change the mode, but noticed that in simple command mode the button will function only when stm32 is in the initial orientation (the LED2 is on when stm32 is in the initial orientation).

In simple command mode, we use the 6D orientation detection which is provided by LSM6DSL. When the orientation changes (angle exceeds the threshold which we set 60°), the interrupt will be triggered and then enable the task to detect the new orientation and send the corresponding command (like resume, pause, previous or next song) by WIFI if the change is legal. We set two constraints to check legality, one STM32 should hold in the new orientation for a while (we set 0.3s), another is it should go back to initial orientation after sending the command. By two constraints, we can reduce the error rate on gesture recognition but increase the reaction time of the command.

In volume control mode, the timer is set to 50Hz and its interrupt will enable the task to calculate the angle, and then send angle by WIFI. Noticed that the wifi data communication rate is only 5Hz because the higher rate may cause WIFI disconnection more.

Angle calculation

Due to gravitational acceleration, we can use the accelerometer to calculate the precise angle when STM32 is stationary. Also when STM32 is moving, we can get an angle by calculating the integral of raw data from the gyroscope. Therefore to get the more precise angle, we combine the angles calculated from the accelerometer and gyroscope with complementary filter. That is, we calculate the weighted mean of two angles.The graph below is three angles under different weights on the gyroscope angles. If the weight is lower, the complementary angles fit more to the accelerometer angles and may be overfitting. If the weight is higher, the line of complementary angles is smoother but the delay to match the current angle increases.So we set the weight on the gyroscope angles to 0.7 .



Results

The GestureSync system successfully demonstrated its capability to recognize gestures and map them to music playback commands. Users could:

- Play or pause music with specific gestures.
- Skip to the next or previous track by tilting the device left or right.
- Adjust volume by tilting the device up or down, with changes based on the angle of tilt.

The complementary filter significantly enhanced data processing, enabling smoother outputs despite sensor noise. However, the project also revealed certain limitations. Errors were observed when debugging float values on the STM32, and high data communication rates occasionally caused WiFi disconnections. These issues were mitigated through careful interrupt prioritization and system constraints.

Future Improvements

Looking ahead, we plan to enhance the GestureSync system by integrating machine learning for more sophisticated gesture recognition. This would allow for a broader range of gestures and increased accuracy. We also aim to support multiple music platforms, such as Spotify and Apple Music, and implement adaptive gesture sensitivity to cater to individual user preferences. Additionally, we hope to explore more complex gesture mappings, including finer volume control and playlist search functionalities.

Resources

- **GitHub Repository**: ESLab Final Project
- **Demo Video Playlist**: YouTube

References

1. AI: How to Perform Motion Sensing on STM32L4 IoTnode. STM32 MCU Wiki. Link
2. LSM6DSL Datasheet and Application Notes. STMicroelectronics. Datasheet
3. Complementary Filters. Hunter Adams. Link