GestureSync: STM32-Controlled YouTube Player

Group 6:

1. 林方瀚 (T13901108)
2. 徐御宸 (B10901158)

Table of Contents

1. Motivation
2. Hardware Components
3. System Architecture
4. Gesture Recognition
5. Future Improvement
6. Contribution and References

Motivation

1. Develop a hands-free music control system using gesture recognition
2. Enable intuitive music playlist navigation through physical movements

Hardware Components

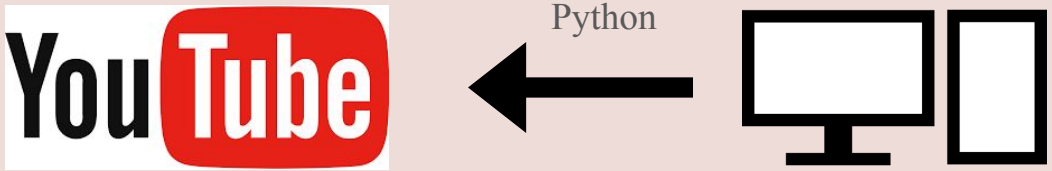
STM32 Board

- **Microcontroller**: STM32L4 Series
- **Key Components**:
  - Accelerometer (e.g., LSM6DSL)
  - Microcontroller
  - TCP/IP Network Interface (via Ethernet/WiFi module)

Computer

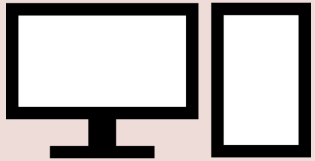
- Operating System: Windows/Linux/macOS
- Web Browser: Google Chrome
- Python 3.x Runtime

TCP Socket



System Architecture

TCP



Python

LSM6DSL

Gesture Recognition

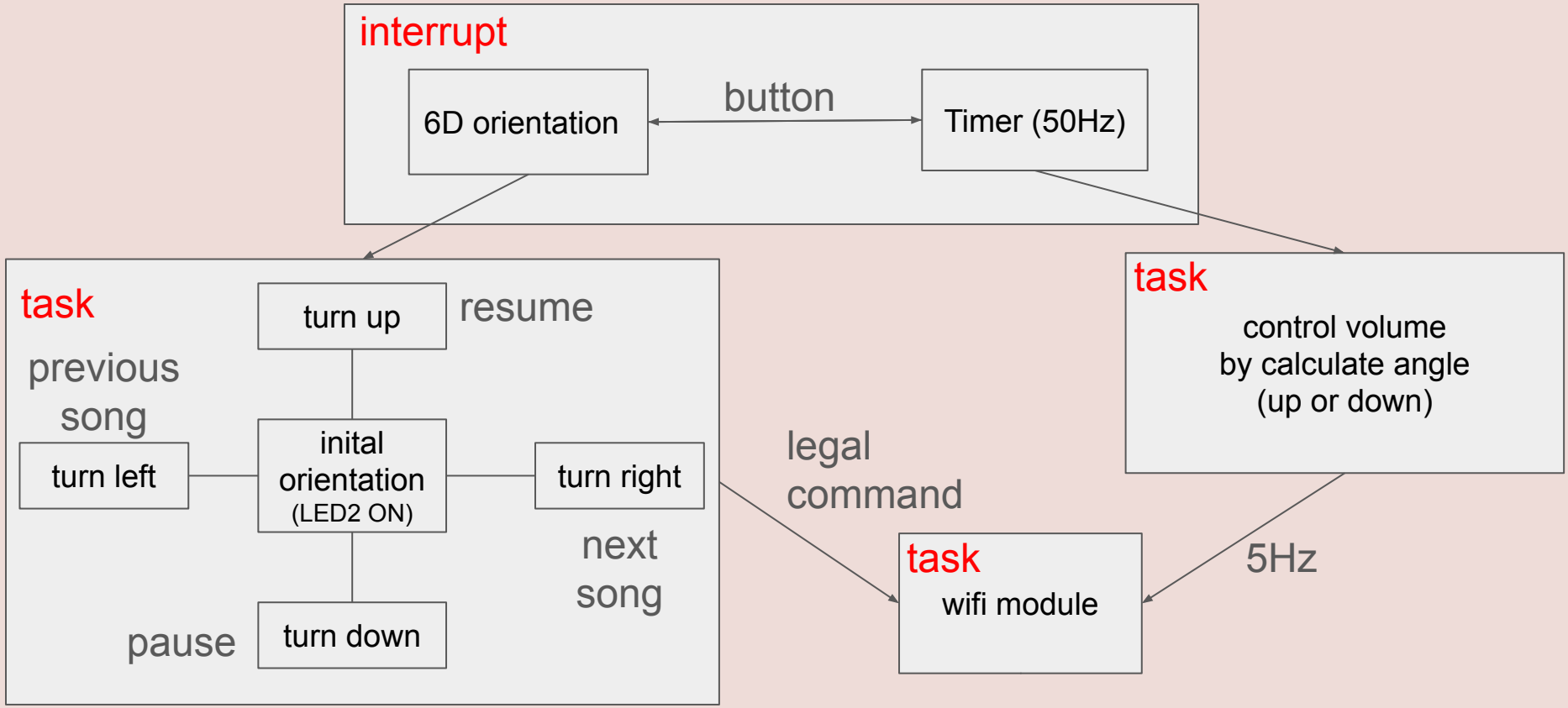
Python TCP Listener

Command Intepreter

Selenium WebDriver

Music Playback Control

System Architecture for STM32

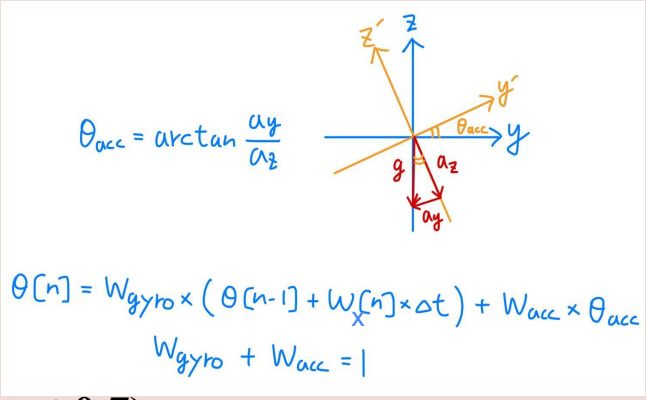


Gesture Recognition

1. **6D / 4D Orientation Detection (Interrupt)**
    - **Detect the change of orientation**
    - **Generate interrupt when degree exceed threshold (50/60/70/80)**
    - **Avoid false detection: 1.should hold in new orientation for a while (0.3s)**

      2.Return to initial orientation after change

Gesture Recognition



2. calculate angle from accelerometer and gyroscope

- **Need to consider the background value**
- **use complementary filter with proper w_gyro (we set 0.7)**
- **assume inital angle ~ 0 degrees (button will function only in initial orientation)**
- **observation :**

w_gyro ↑ ⇒ smooth↑ but delay↑

```
BSP_ACCELERO_AccGetXYZ(pDataXYZ);

BSP_GYRO_GetXYZ(pfDataXYZ);

float pos_or_neg = 0.0;
if(pDataXYZ[0] > 0) pos_or_neg = -1.0;
else pos_or_neg = 1.0;

float acc_angle;
if(pDataXYZ[2] == 0) acc_angle = pos_or_neg * 90.0;
else acc_angle = pos_or_neg*atan2f(abs(pDataXYZ[0]),abs(pDataXYZ[2]))*(180.0/PI);

angle = weight_gyo * ( angle + (1/50)*(pfDataXYZ[1]+gy_offset)/1000 ) + (1.0 - weight_gyo) * acc_angle;
```

Angle calculation

X : (1/50) s

Y: degree

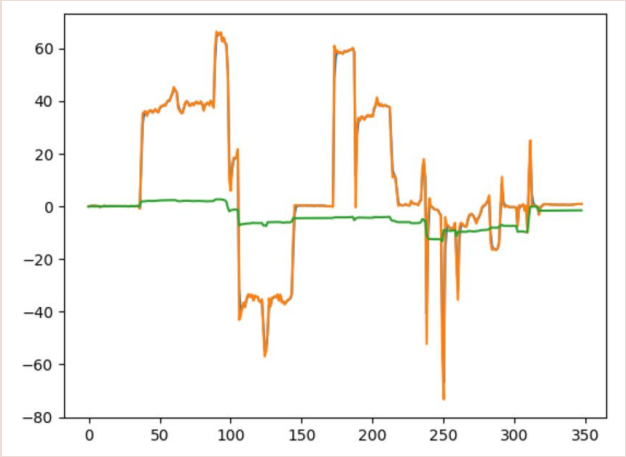
blue: complementary filter

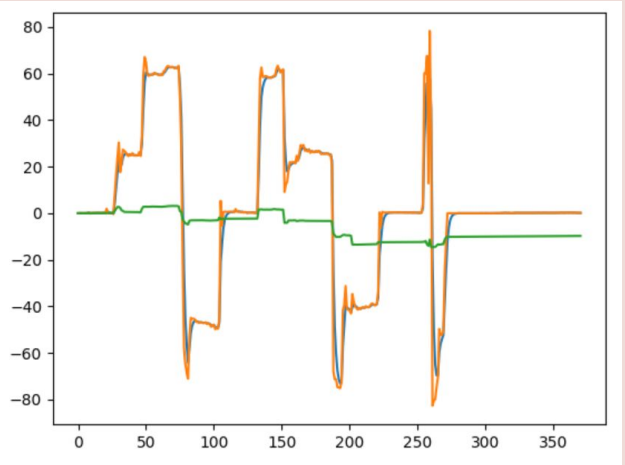
green: only gyroscope

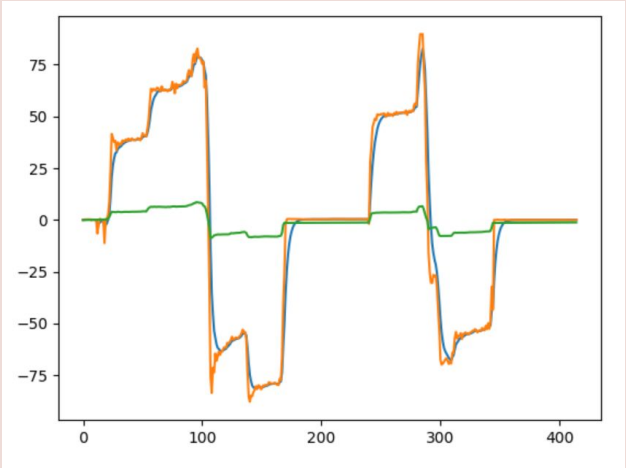
orange: only accelerometer

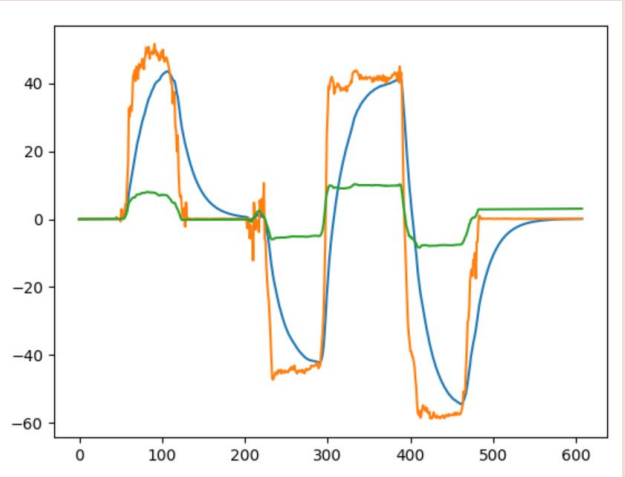
w_gyro: 0.2 0.5

0.7 0.95









Gesture Mapping and Data Decoding

```python
def process_gesture_command(self, command):
    """
    Map STM32 gestures to music control actions
    Supported Gestures:
    - 'Left': Previous Song
    - 'Right': Next Song
    - 'Down': Pause
    - 'Up': Play/Resume
    - 'volume_start': Begin Volume Control
    - 'volume_stop': End Volume Control
    """
```

```python
while True:
    # Receive data
    if self.volume_changing:
        data = client_socket.recv(1024)
        data_int = int.from_bytes(data[0:-1],byteorder= 'little', signed=True)
        if data_int <= 100 and data_int >= -100 :
            data = data_int
            print(data)
        else :
            data = data.decode("utf-8")
            data = data[0:-1]
            print(data)
    else:
        data = client_socket.recv(1024).decode("utf-8")
        data = data[0:-1]
        print(data)

    # Process received command
    self.process_gesture_command(data)
    msg_f = 'Completed'
    client_socket.send(msg_f.encode())
```

Relative Volume Control

New_volume = Old_Volume + Rotation*Sensitivity

```python
# Volume Control Attributes
self.volume_changing = False
self.volume_control_mode = None
self.initial_volume = 50  # Default initial volume
self.volume_sensitivity = 10/9  # abt 1% volume change per degree
```

```python
def adjust_relative_volume(self, rotation):
    try:
        # Ignore very small rotations to prevent noise
        if abs(rotation) < 0.1:
            return

        # Current volume retrieval
        current_volume = self.get_current_volume()

        # Volume change calculation
        volume_change = rotation * self.volume_sensitivity

        # Calculate new volume
        new_volume = max(0, min(100, current_volume + volume_change))

        # JavaScript to set volume
        volume_script = f"""
        var video = document.querySelector('video');
        if (video) {{
            video.volume = {new_volume / 100};
            return video.volume * 100;
        }}
        return null;
        """

        # Execute volume adjustment
        result = self.driver.execute_script(volume_script)

        if result is not None:
            actual_volume = round(result)
            change_direction = "increased" if volume_change > 0 else "decreased"
            self.update_status(f"Volume {change_direction} to {actual_volume}%")
        else:
            self.update_status("Failed to adjust volume")

    except Exception as e:
        self.update_status(f"Volume Adjustment Error: {e}")
```

Absolute Volume Control

New_volume = (Rotation / 90)*100

with 0 <= Rotation <= 90

```python
def map_rotation_to_volume(self, rotation):
    """
    Map rotation degree to absolute volume level
    Assumes rotation range of 0-90 degrees
    """
    # Linear mapping: 0 degrees = 0% volume, 90 degrees = 100% volume
    volume = (rotation / 90.0) * 100
    return max(0, min(100, volume))
```

```python
def set_absolute_volume(self, volume_level):
    try:
        # JavaScript to set volume
        volume_script = f"""
        var video = document.querySelector('video');
        if (video) {{
            video.volume = {volume_level / 100};
            return video.volume * 100;
        }}
        return null;
        """

        # Execute volume setting
        result = self.driver.execute_script(volume_script)

        if result is not None:
            actual_volume = round(result)
            self.update_status(f"Set Absolute Volume to {actual_volume}%")
        else:
            self.update_status("Failed to set volume")

    except Exception as e:
        self.update_status(f"Volume Setting Error: {e}")

def get_current_volume(self):
    try:
        volume_script = """
        var video = document.querySelector('video');
        return video ? video.volume * 100 : null;
        """

        current_volume = self.driver.execute_script(volume_script)

        return round(current_volume) if current_volume is not None else 50

    except Exception as e:
        self.update_status(f"Volume Retrieval Error: {e}")
        return 50  # Default safe volume
```

Observation and Discussion

- **Debug: Error occur often when using printf with float**
- **Debug: priority is important**
- **Use some constraint in gesture recognition will reduce the error rate but increase the reaction time of command**
- **High data communication rate with wifi may cause disconnection**

Future Improvements

1. **Machine Learning Gesture Recognition (Preplanned)**
2. **More Complex Gesture Mapping like Volumm Control**
3. **Support Multiple Music Platforms**
4. **Adaptive Gesture Sensitivity**

Contribution

林方瀚

1. Python Code and Implementation

徐御宸

1. Gesture Recognition
2. Code Debugging

Project link

1. github : https://github.com/Ken-Hsu-1/ESlab_final_project
2. demo video playlist (including absolute and relative volume control):

   https://youtube.com/playlist?list=PLpyxc1voi02qkzzVdoig6L7b6ILT59Pj0&feature=shared

References

1. *AI:How to perform motion sensing on STM32L4 IoTnode - stm32mcu*. (n.d.). https://wiki.st.com/stm32mcu/wiki/AI:How_to_perform_motion_sensing_on_STM32L4_IoTnode#Compile--download_and_run
2. STMicroelectronics. (2018). *LSM6DSL: always-on 3D accelerometer and 3D gyroscope* (pp. 1–3) [Application note]. https://www.st.com/resource/en/application_note/an5040-lsm6dsl-alwayson-3d-accelerometer-and-3d-gyroscope-stmicroelectronics.pdf
3. STMicroelectronics. (2017). *LSM6DSL Datasheet - Production Data* [Datasheet]. https://www.st.com/resource/en/datasheet/lsm6dsl.pdf
4. *Complementary_Filters*. (n.d.). https://vanhunteradams.com/Pico/ReactionWheel/Complementary_Filters.html