EE 186 Final Project

Demo video- https://drive.google.com/file/d/1CKohRoa478FF6vglMzryubWYNk usp=sharing

Github -

```
https://github.com/christopherlann/EE186-Final-Project
```

Timeline:

- **The final presentations will be on Thursday, December 4th, during class time from 12:00 to 1:20 PM**
- **The final project due date is still December 9th, and you will have until then to make any final edits.**

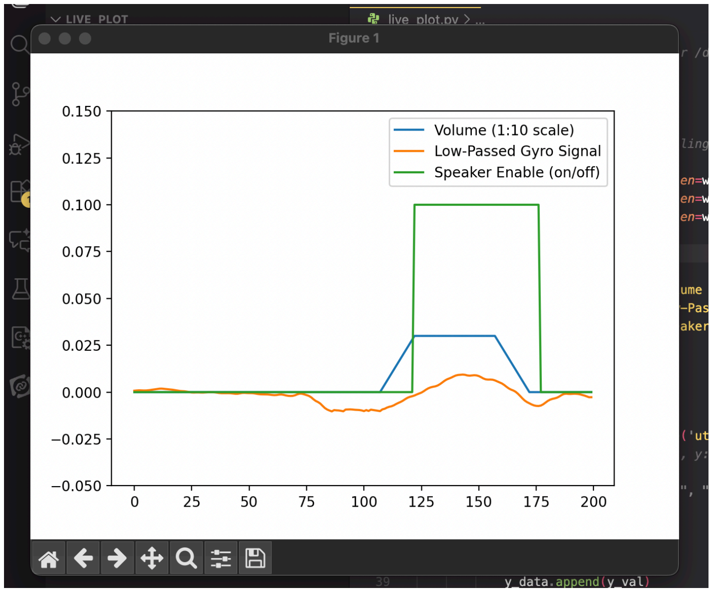
IMU - Ben

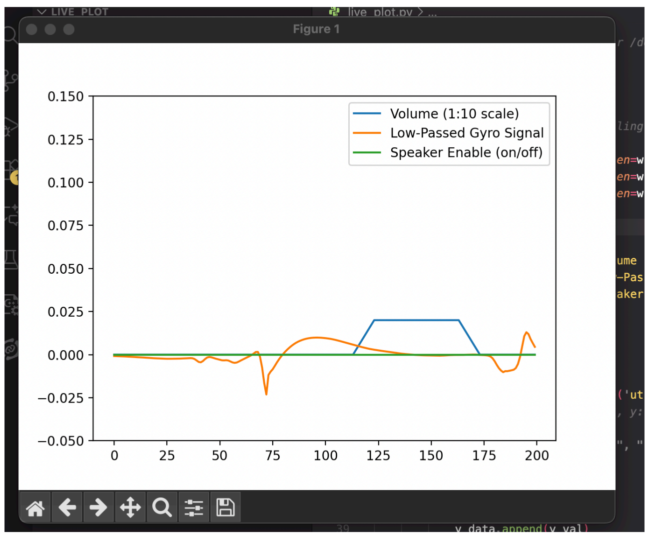
Development Process:

- Step #1: Set up data lines with two IMUs (I2C)
- Step #2: Convert acceleration values to angular velocity by taking the difference. Optionally taking the square root as well.
- Step #3: Create a layered filter to identify a steady turning signal and trigger the drifting sound effect

Filtering stack:

- calibrate DC component and remove from raw angular velocity signal
- offset angular velocity → 2nd order lowpass
- 1% leaky integrator inside the lowpass memory to account for sensor drift
- 10% and 50% leaky integrators above thresholds
- Trigger an initial guess at turning when a signal stays above a threshold for a certain number of samples
- Run a first order lowpass averaging window over these initial triggers to create a volume signal
- Trigger the speaker enable line when this volume signal goes above a threshold.





GPIO Map:

Device Pin	Board Pin
IMU #1 - SDA	PB9
IMU #1 - SCL	PB8
IMU #2 - SDA	PF0
IMU #3 - SCL	PF1

Peripherals

I2C	Configuration and data transfer from accelerometers

Custom Modules:

i2c_helper.c	Created wrapper functions for the STM32 i2c module that were tailored for my usage

acc_mag.c	Controller for accelerometer and gyroscope including a custom datatype.
acc_controller.c	Configuration of the sensors, calibration of the gyroscope calculation, and all of the filtering for identifying turning

Button/LEDs - Angel

Pins:

- Button - PB5
- LEDs - PA1(Green), PA2(Yellow), PA3(Red)

Development Process:

- Button:
  1. Set GPIO
     - Verified correct wiring(C → GND, NO → PB5)
  2. Set EXTI interrupt on rising/falling edge
  3. Set software debouncing
     - Added small debounce window using state machine
     - FIxed issues with mutliple interrupts triggering with one click
  4. Detect single clicks, double clicks, and long clicks to cycle through OLED display screens
     - Single Press
     - Double Press (Within 300ms)
     - Long Press (Within 800ms)

- LEDs:
  1. Set PA1, PA2, and PA3 as PWM outputs on TIM2
  2. Start PWM channels
  3. Adjust LED brightness using duty cycle

Peripherals:

- EXTI Interrupts
- TIM2 PWM

Software:

ButtonHandler():

- Classifies single, double, and long clicks based off of time

Set_LED_Level(Volume):

- Maps a level 0-100 for the PWM duty cycles of each LED

Hardware:

- PA1, PA2, PA3 → TIM2 PWM Channels → LEDs
- https://www.adafruit.com/product/560

Speaker - Luis

Pins:

DAC1 - PA4

Peripherals:

DAC, TIM, DMA

Development processes:

Flow for youtube → C array or Hex Dump

1. yt-dlp to get mp4 of youtube videos (https://github.com/yt-dlp/yt-dlp)
2. (If you dont have certain programs working) → use ffmpeg to convert mp4 to mp3
3. Use Audacity to convert mp3 into mono stereo wav. file
4. Wav to c array (https://github.com/folkien/wav2c)
    a. Or use xxd to get hex dump

Inspiration for flash embedded audio (https://www.youtube.com/watch?v=D2iXQy6DzbY)

- Saving Audio file .wav to STM32
    - Currently have a way to get youtube video → pcm16
    - Current method of saving: embed mem array into flash mem and ready from there

- Wire audio amplifier to DAC/Speaker
    - Test that Audio plays
    - Ensure audio snippet sounds clear
    - Ensure audio works with multiple peripherals

- Coordinate with Ben to determine when to play and how loud to play audio clip
    - Software: clamping DAC output for voltage
    - Trigger playing audio
    - Scale output based on volume configured by user

Software:

- InitSpeaker():
    - Initializes all data surronding the audio C arrays aka (sample count * pointer to memory where audio is stored)
- PlayDrift(index, volume):

- Loads the Audio array into buffer and loads it to DMA scaled by volume scalar

Hardware:

- Dac Output → PMA8302 audio amplifer → Speaker

Features:

- 3 Unique Drift sounds (~3-4 seconds each)

Display - Chris

- Display features:
  - OLED: https://www.adafruit.com/product/326?srsltid=AfmBOor3Y4gTTBQpBnzepaoWjhfxgAmhcPx05mmALNmAgTG7bI2Ymp6A
  - Mode 1: Displays acceleration data / peak acceleration
  - Mode 2: Displays drift sound selected
  - Mode 3: Displays volume bar and volume set indication

OLED Display Setup

This is for the SSD1306 display 4 wire SPI interface

Pin on OLED	Pin on STM32
Data	**D11 (MOSI)**
Clk	**D13 (SCK)**
DC	**D4**
Rst	**D2**
CS	**D7**
3Vo	*Leave unconnected*
VIN	**3V3**
GND	**GND**

- Using the following driver library: https://github.com/afiskon/stm32-ssd1306/tree/master

How to configure in STM32CubeIDE:

1. Make OLED folder in Drivers → Place files there and link in project settings
2. Configure defines for SPI in the ssd1306_conf_template.h file and rename as ssd1306_conf.h
3. In .ioc enable SPI 1
   a. Mode: Full-Duplex Master
   b. Data size: 8 bits
   c. First bit: MSB First
   d. CPOL: Low
   e. CPHA: 1 edge
   f. NSS: Software
4. Clock config:
   a. SPI1 is on the APB2 (PCLK2)

b. Configure for 1-2MHz using prescaler value

OLED Display modes logic

- Long press changes mode
- Short press cycles through submode functions

Software

oled_accel_mode()	Function to display accelerometer data
oled_speaker_mode()	Function to display and set speaker sound
oled_volume_mode()	Function to display the volume level
oled_volume_set()	Function to display that volume is set