ECE 445

SENIOR DESIGN LABORATORY

PROJECT PROPOSAL

Hand Gesture Audio Effects System

Team No. 39

Sarthak Singh
(singh94@illinois.edu)
Zachary Baum
(zbaum2@illinois.edu)
Sergio Bernal
(sergiob2@illinois.edu)

TA: Zicheng Ma

February 7, 2024

1. Introduction

1.1 Problem

In audio production, both amateur and professional settings lack intuitive, hands-free control over audio effects. This limitation restricts the creativity and efficiency of users, particularly in live performance scenarios or in situations where physical interaction with equipment is challenging. Furthermore, those with certain disabilities may not always be able to physically interact with hardware to create audio effects.

1.2 Solution

A system which utilizes a camera and applies audio effects based on which gesture is being made in the view of the camera. It will make it so audio effects can be applied without needing to press physical buttons.

1.3 Visual Aid

On a high level, the final product would contain a camera, a speaker, and the electronics in between. The camera would be monitoring the hands of someone who is in view of the camera.

In this example let's say an effect “x” is triggered by a thumbs up. In a standby state, we will be playing some audio files out of the speaker with no effects.

Once the camera detects a thumbs up it will send the proper signal to our PCB which will then pass all the audio signals through a system which will apply the “x”. That effect will remain on until another gesture is shown in the camera & there will be a gesture for “normal”3

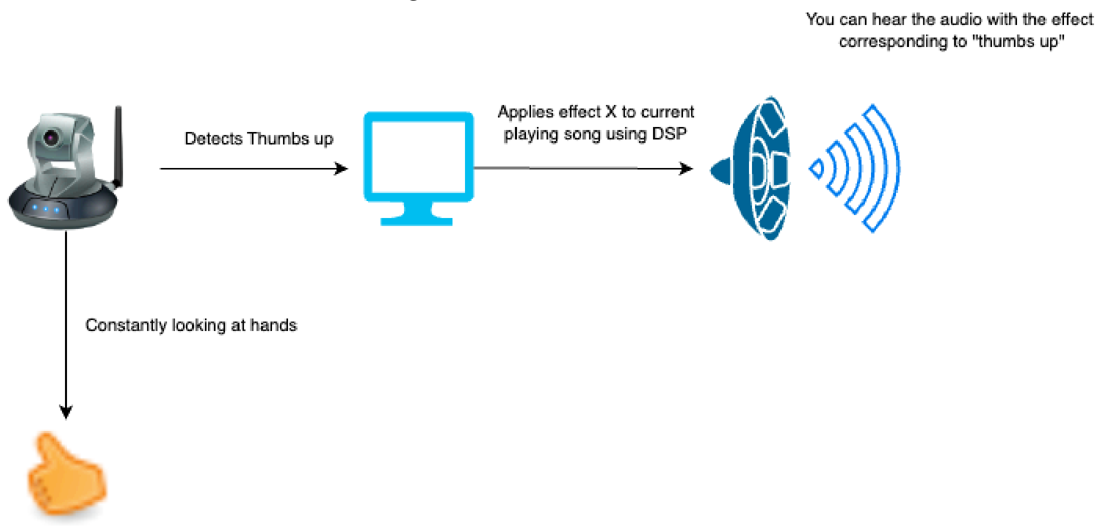
You can hear the audio with the effect corresponding to "thumbs up"
Detects Thumbs up
Applies effect X to current playing song using DSP
Constantly looking at hands

Figure 1. High-Level Overview of the Hands Gesture Audio Effects System

1.4 High level requirements

On a high level, here are requirements for this project to be considered successful:

1. We are able to detect hand gestures using a camera with 95% accuracy +-5% within 5 seconds of making the gesture. To be a bit more precise, from the time that a gesture is made from the time we detect it is done within 5 seconds. Furthermore, we want to make sure we can detect the “correct” gesture 90-100% of the time within those 5 seconds.
2. We are able to apply 5 digital effects to an audio signal and apply that effect to the external speaker within a 1 second of receiving a signal from the gesture detection subsystem with a tolerance of +-½ of a second.
3. We want to display the current effect being applied to the music on an external display with 99% accuracy with a tolerance of 2% in both directions. Regardless of what the hand gesture system is detecting, whichever effect is actually being applied and what is displayed on the screen needs to be the same with 99% accuracy.

2. Design

2.1 Block Diagram

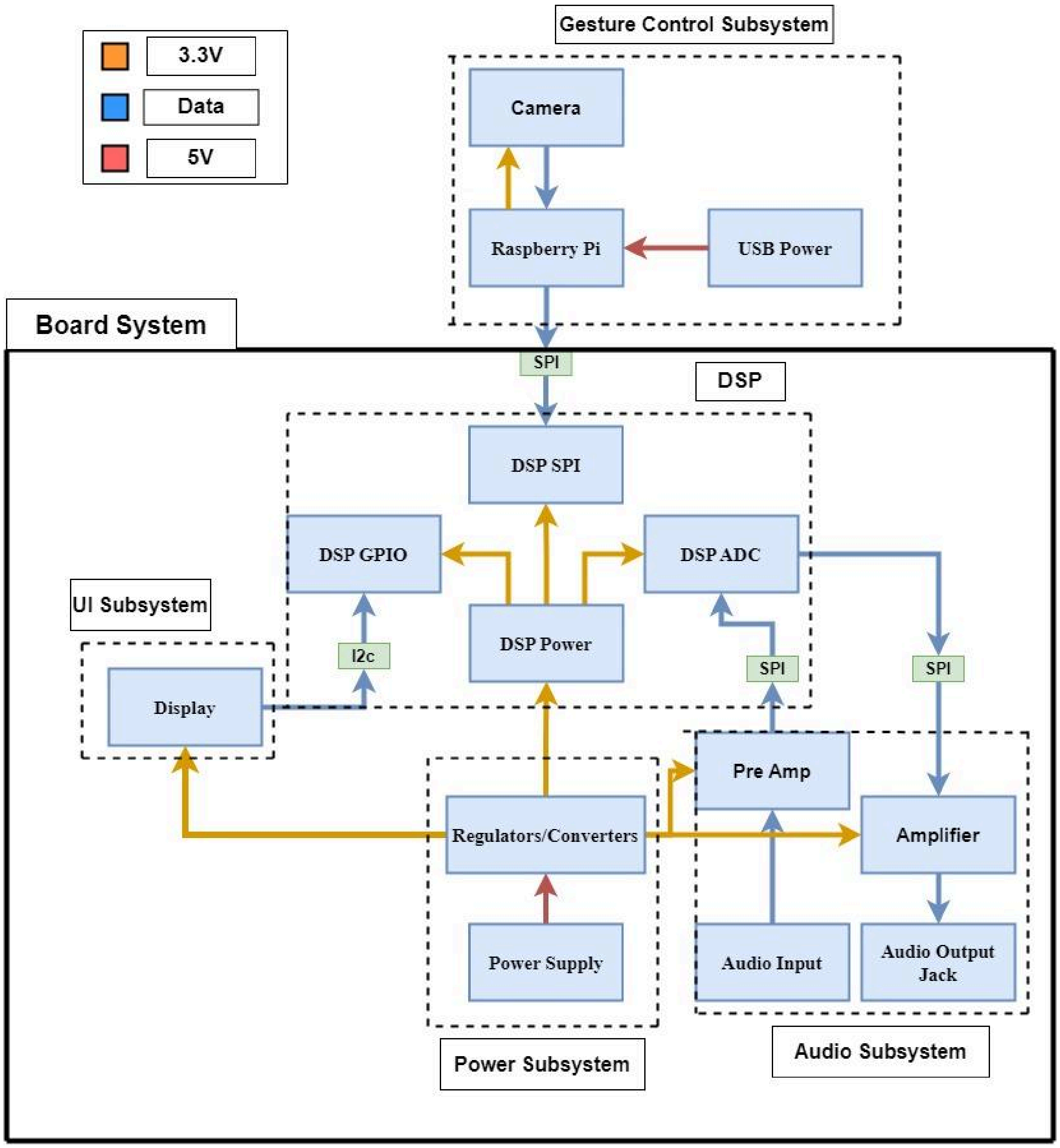

Figure 2. Block Diagram of the Hands Gesture Audio Effects System Design

2.2 Subsystem Overview

2.2.1 UI Subsystem

The UI (User Interface) Subsystem facilitates interaction between the user and the Digital Audio Effects Processing unit primarily through a display. This subsystem is essential for providing visual feedback to the user, showcasing current settings. The display communicates via I^2C with a STM32H743VIT6 microcontroller. This allows for dynamic updates on the screen based on system changes or user interactions via other input methods, such as the Gesture Control Subsystem. The display is a critical component for user feedback, ensuring users can navigate and control the device's functions effectively without physical buttons.

2.2.2 Gesture Control Subsystem

The Gesture Control Subsystem utilizes a camera to detect hand gestures which are translated into commands for the device. This subsystem consists of a imaging sensor (camera) connected to a Raspberry Pi via I^2C, which analyzes the captured video feed to recognize specific gestures. The Raspberry Pi serves as the gesture interpretation layer, converting recognized gestures into command signals to be sent to the STM32H743VIT6 microcontroller for digital audio effects processing. Also, the Raspberry Pi is powered through a USB separate from the power being supplied to the custom PCB. This ensures consistent operation without taxing the main device's Power Subsystem. The communication between the Raspberry Pi and STM32H743VIT6 microcontroller is established via SPI, enabling the transmission of gesture-based commands to alter device settings or control audio effects seamlessly.

2.2.3 Power Subsystem

The Power Subsystem provides essential power management and distribution to all components on the custom PCB. It is composed of a primary power source (5V wall power adapter) and various voltage regulators and converters to deliver specific voltage levels required by different parts of the device. This subsystem is responsible for ensuring that sensitive electronic components, including the processing units and the Audio Subsystem, receive clean and stable power. This stable power supply is crucial for the uninterrupted and noise-free operation of the device, affecting everything from digital processing to audio output quality. It interconnects with every other subsystem, directly influencing their performance by supplying the necessary electrical power.

2.2.4 Audio Subsystem

The Audio Subsystem's focus is on inputting and outputting audio signals. It encompasses audio inputs for receiving analog signals, which are then conditioned by pre-amplifiers if necessary, before being digitized by ADCs for processing. The DSP, a part of the microcontroller, applies desired audio effects, after which the signal is converted back to analog form via DACs for output. Amplifiers within this subsystem then boost the processed audio signals to drive output devices like speakers or headphones. This subsystem relies on the Power Subsystem for the energy needed to operate amplifiers and processing units.

2.3 Subsystem Requirements

2.3.1 User Interface Subsystem

1. The display must show the user what audio effect has been applied to the input audio signal, 0.1 seconds (at most) after the STM32H743VIT6 microcontroller communicates with it via I^2C that a unique hand gesture has been detected by the gesture detection algorithm.
2. The display must show at most a ten letter word that represents an audio effect.

2.3.2 Gesture Control Subsystem

1. The captured video coming from the imaging sensor (camera) to the Raspberry Pi via I^2C must be at least 20 frames per second.
2. The gesture detection algorithm must detect a hand gesture in less than 5 seconds.
3. Raspberry Pi must notify the STM32H743VIT6 microcontroller to apply an effect on the input audio signal, 0.2 seconds (at most) after a unique hand gesture corresponding to the audio effect has been detected by the gesture detection algorithm.

2.3.3 Power Subsystem

1. The buck regulator should provide 3.3 V ± 0.5% coming from a 5 V wall power adapter source to the necessary supply pins.
2. The 5 V wall power adapter should be able to provide at most 2 A of current. Although the pins of the microcontrollers only need milliamps of current, the external display output requires at least 1 A.

2.3.4 Audio Subsystem

1. The sound being outputted by the speaker must be less than 80 dB for safety purposes.
2. The effect applied to the input audio signal must be heard clearly such that there should not be any static sound unless it is a distortion audio effect.

2.4 Tolerance Analysis

In terms of design choices that could pose a risk to successful completion of the project, one important requirement of our project design is that our computer is powerful enough to process gestures within 5 seconds.

Computer vision is computationally intensive and although the tasks we need to do are not incredibly difficult, it will require a lot of resources to be able to execute within our defined tolerances. If we have a camera that is too high of quality, we may not fit our timing guarantees given that we are limited by the cpu/memory of the Raspberry Pi. Given that we do not have all the parts such as the camera that connects with the Raspberry Pi, we cannot actually benchmark if it can do the task that we want it to do. Instead we will try to estimate how long it should take to detect a hand gesture and match that against our requirement.

We plan to use OpenCV which is an imaging library. If we are able to download OpenCV on our Raspberry Pi, this ensures that we will be able to at least run some code. Upon searching the internet, I see examples of people running OpenCV on the Raspberry Pi 3 Model B. Furthermore, on the official OpenCV forum, there is a confirmation that it can run even on a Raspberry Pi 1 (see here). Given this, we can say that we will at least be able to detect hand signals given we can run this imaging library. Furthermore, the Raspberry Pi 3 used for this project is a lot more powerful than a Raspberry Pi 1.

There are a few things that factor into detecting hand gestures within a certain duration. First, we need to find the latency from when a camera sees something and it's shown inside of our microcontroller. The following formula is used to calculate that latency.

$Camera\ Latency\ =\ Sensor\ Latency\ +\ Processing\ Time\ +\ Transfer\ Time$

Sensor Latency: This is the time that it takes for the camera sensor to capture the image after receiving light. It typically depends on the camera's hardware and settings. An exact number was not found but the time was seen to be in the single digits of milliseconds from looking at similar cameras. We will approximate on the high end and say this takes 50ms which is way longer than it should ever take

Processing Time: After capturing the image, the camera's internal processing pipeline may apply various adjustments such as white balance, exposure correction, and noise reduction. Once again the Pi doesn't tell us this time but we can find a more high quality camera and use that as a high end. Most high end video cameras took a fraction of a second so lets say 250ms.

Transfer Time: Once the image is processed, it needs to be transferred from the camera to the device where you're viewing the feed. For wired connections, which is what we have, it is just the time to send the average size of an image over the wire. The ribbon cable we used can transfer data at the speed of 186 MPbs and we can assume we are shooting at 1080p (resolution of camera we plan to use). We will calculate the time to send a single frame.

$Total\ Data\ per\ Frame\text{:}\ 1920 \times 1080 \times 3\ bytes\ =\ 6,220,800\ bytes$ = 5.93 MB
$Total\ time\text{:}\ 5.93/186\ *\ 1000\ =\ 31.88\ ms$

In total we can say this will take $50ms\ +\ 250ms\ +\ 31.88ms\ =\ 331.88ms$

This is almost a third of a second and so we have more than 4 seconds to actually detect the gesture.

The Raspberry Pi 3B features a 1.2 GHz quad-core ARM Cortex-A53 processor and 1 GB of RAM. It's difficult to say exactly how much time it will take to do computer vision but a test program was ran on a macbook which processed a single frame. Also, it ran some analysis on it and took just about 100ms. It would need to take $(5000\ -\ 331.88)/100\ \ -\!>\ \ \sim 46x$ longer for the Pi to complete this task to fit into our timing constraint.

My computer had 4 cores at 3.2 GHz and 16Gb of ram. Assuming that memory and cpu capacity scales linearly (which it never does) we can say that the macbook is 20x more powerful that this Raspberry Pi. We could then say that it would take $100 * 20 = 2000ms$ or 2 seconds to process a frame on this Pi.

Putting this all together the super high end on how long this whole process can take seems to be about 2.3 seconds while our constraint is double that value.

High end was used for all estimates, and we predict that it would be way quicker than this in practice. However, for the case of a tolerance analysis, it is proven that we will be able to fit into our defined timing specification.

3. Ethics & Safety

We do not see any immediate ethical and safety implications of this project. However, for the sake of being comprehensive we will address some potential scenarios which could constitute. The first is that this project could be applied to help those with certain disabilities and could be even necessary for someone with a disability to DJ or work with audio effects. As a result, the price would be raised incredibly high to make it unaffordable for many(similar to how prescription drugs like insulin are prices) due to it being a necessity. We understand how unethical this would be and if we decided to sell this product we would ensure its pricing is fair to all those who need it.

Another could be an issue of privacy with the camera. Although we have no reason to record the camera other than a live feed, a malicious party could potentially try to use the camera to invade privacy. We will address this issue by not connecting any parts of this device to the internet meaning there is no chance of an “attack”, All camera processing will be local to the unit.