Software Controlled Sound Sources

Team 26 - Ben Sisserman, Micki Rentauskas, and Won Woo Lyu
ECE 445 Project Proposal - Fall 2020
TA: AJ Schroeder

1 Introduction

1.1 Objective

The development of hearing aids requires the replication of real-world environments and sounds in order to test and improve hardware and software. This requires a variety of different types of sounds from different distances and angles. Commonly, speakers are used with pre-recorded sounds that must be originally recorded in anechoic chambers, which can be expensive. Speakers also direct their audio output, unlike their physical counterparts. A bell would create a sound wave that would go in all directions and reflect off of the walls of the room creating a new sound that a speaker would not be able to mimic. In order to provide more immersive demos and a larger variety of sounds for the development of hearing aids, one must build physical systems that create natural sounds rather than recordings. In addition these devices should be controlled by software for repeatable and consistent use during trials.

1.2 Background

Ryan Corey, who is a Postdoc in UIUC, does research on audio-processing for noise cancelling hearing aids and uses speakers placed throughout a room to emulate various sound sources for testing. He has requested that we create software controlled real sound sources for his lab. In his lab, most of the sounds are outputted from a series of speakers, and sounds are produced by using specialized software. The below picture is from the article[1] of his team working on Cooperative Listening Devices, and they used 12 speakers to test Cooperative Listening Devices. These speakers can be replaced by our project, which will produce various real sounds while not using a speaker.

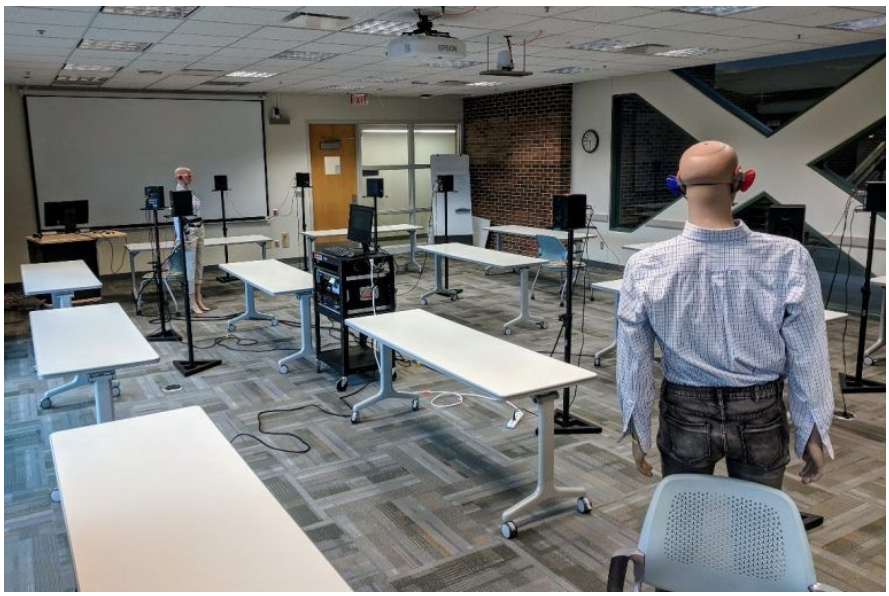
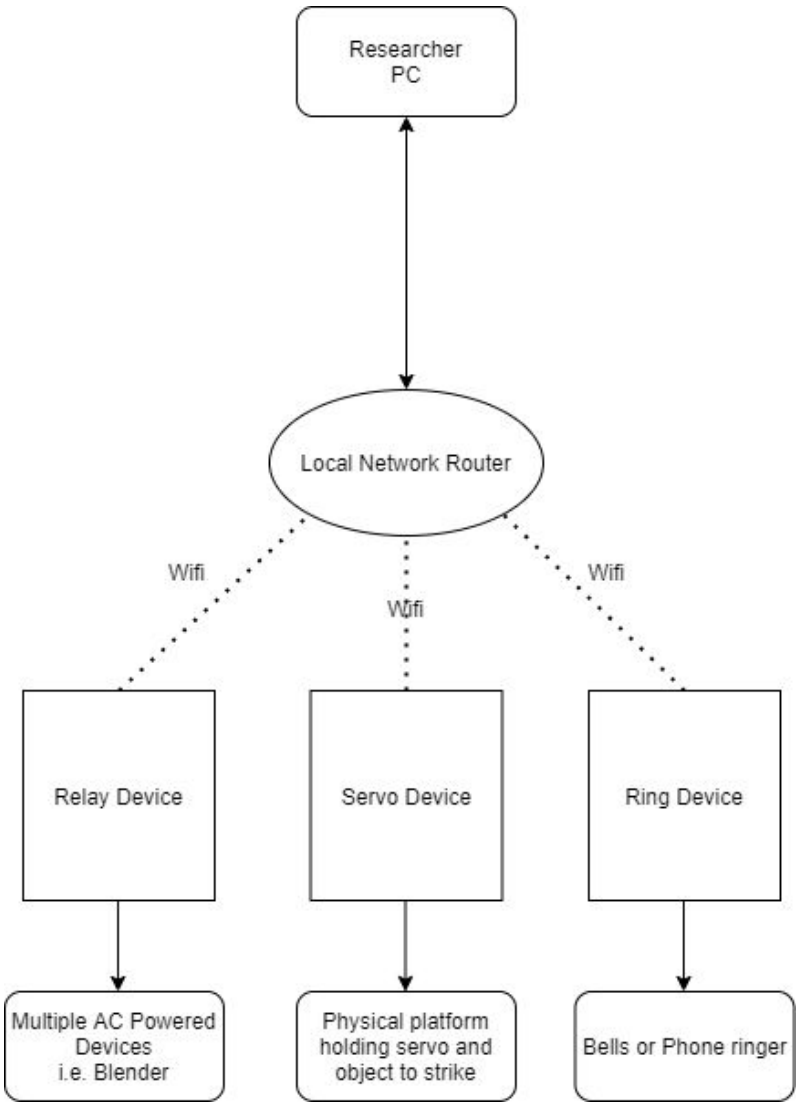


Figure 1. The conference room used for the massive distributed array dataset

1.3 Physical Design Overview



1.4 High-level Requirement list

- At least 3 different types of sounds must be generated by the devices, which are consistent and repeatable with exception to real-world variations, i.e. air pressure/temperature.
- Latency between software commands and sound generation must be consistent on the same device. Latency itself is not a concern, but latency should not vary by more than 5ms.
- Devices must function with at least 6 ft of distance between devices for simulating real-world environments and circumstances.

2 Design

Our design will trigger the sound devices from commands sent via a Python module that will be running on the test PC. Our Python module will initialize all SCSS on the network by listening for broadcasts from the devices to the local router. The devices will broadcast their device type (relay, servo, or ringer) and ID (MAC address) when turned on, and will continue broadcasting until they are powered off or initialized by the control unit. The devices will receive and decode commands sent from a router communicating with the PC using an ESP32 microcontroller. The boards will be powered by a 5V power source and each PCB type has its own mechanism for sound generation. The relay mechanism contains relays which will toggle wall power to various “switch type” electronic devices like a vacuum or blender. The servo mechanism will use a PWM from the microcontroller to control the servo. The ringing mechanism will use digital output to activate a ringing circuit to activate the sound device extracted from a vintage wall phone.

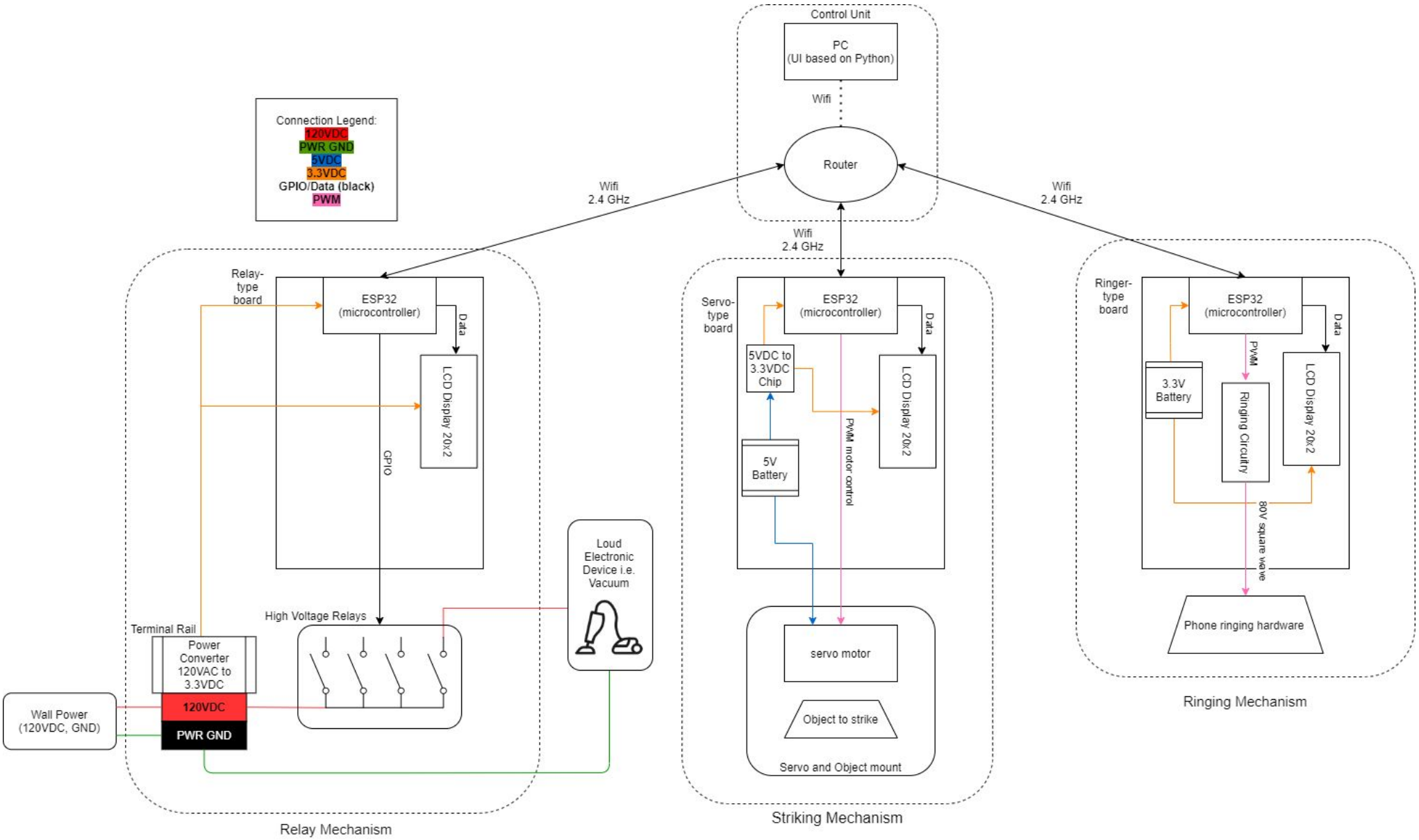


Figure 2. Block Diagram

2.1 Control Unit

The control unit will consist of the user's PC and local router. The PC will communicate commands using a Python API that the user can integrate into their current Python scripts. The API should be limited to **1** Python module for simplicity of integration. The local router will send commands to the boards via Wifi using TCP communication, which will be set up separately for each board, due to its superior reliability.

Requirement #1: The Python module must be able to distinguish between the different boards and have error checking for invalid commands to the boards. For example the servo board will not take commands for duration like the relay board would, and assertions must be made for the user to tell which commands are valid for each device.
Requirement #2: Another requirement for this system is to ensure in software that all messages are consistent in their timing. Since we are sending messages through Wifi, there is no guarantee that latency will not vary. A software solution from the side of the control unit, in communication with the boards, must be implemented. Perhaps the message could contain timing of command execution, and the boards will purposefully delay upon receiving messages until a set latency is reached.
Requirement #3: Control Unit must be OS agnostic. The user PC should not be limited by the system.

2.2 Relay-type Board

The relay board will connect the microcontroller with relays to turn standard electronic devices plugged into wall power on and off. It will do so for varying amounts of time depending on received commands. The board will supply power to the microcontroller via an onboard battery and protect the microcontroller from the high-voltage of the relays. The microcontroller will communicate with the Control Unit using Wifi.

Requirement #1: Complete power circuit for 4 different "switch" type devices with <10ms latency from GPIO in to switch action. Should be able to tolerate rated currents for devices, up to 12A (15A is the typical circuit breaker level, so these devices individually won't pull higher than that)

2.2.1 120VAC to 3.3VDC Power Converter

Terminal rail mounted power converter.

Requirement #1: Capable of supplying at least 1A for powering ESP32 and LCD display.

2.2.2 ESP32

Microcontroller must be able to receive signals over Wifi and decode. It should also be able to receive commands via a wired USB connection. Decoded signals control the relays and ringer boards via GPIO for the decoded duration, and the servo board takes PWM.

Requirement #1: Receive and decode commands over 2.4GHz Wifi.
Requirement #2: Receive and decode commands via USB.
Requirement #3: Output 3.3VDC GPIO for given duration.

2.2.3 LCD Display

LCD display will report the status of the device to the user Indicating when it is broadcasting its identification on the network, successfully initialization of Control Unit, and when battery is low. Would display which relay is on and for how long. Size of display is 20x2.

Requirement #1: Must be able to receive 3.3V signals from GPIO
Requirement #2: Must be powered at 3.3V similar to the microcontroller
Requirement #3 : Must be large enough to display strings of at least 20 characters

2.2.4 Relay Board

An off the shelf commercial relay board will be used to ensure that high-voltage of the outlet does not contact our PCB and damage/burn components. This board will have four relays which can toggle four different sound-producing objects.

Requirement #1: Toggle positive line voltage for four separate devices.
Requirement #2: Tolerate up to 12A for each relay at 120VAC.
Requirement #3: <10ms latency between received GPIO and switch action.

2.2.5 Terminal Rail

A terminal rail will be used in order to safely organize and connect positive and negative wall voltages as well as a 120VAC to 3.3VDC converter. See example of terminal rail here (https://m.media-amazon.com/images/I/61LYVf5zH7L._AC_SS350_.jpg)

Requirement #1: Hold terminal blocks for 120VDC and PWR GND and provide isolation from each other.
Requirement #2: Support up to 48A of current (12A per switch-type device).

2.3 Servo-type Board

The Servo board will use an ESP32 microcontroller to receive commands via Wifi, power one ESP32 microcontroller and one 4.8V servo motor. This block will create the “striking” sound by mounting an object to strike and the servo close together, and controlling the servo with the microcontroller to strike when the microcontroller receives a command.

2.3.1 5V Battery

The battery must be able to supply 5VDC to the servo motor and 3.3VDC +/- 0.3VDC to the microcontroller.

Requirement #1: Must provide at least 1.5A for three hours of frequent use of both the microcontroller and the servo motor (once or twice a minute).

2.3.2 Step Down Chip

Off the shelf chip which converts a 5VDC battery to voltage usable for microcontroller and LCD display.

Requirement #1: Must be able to step down 5VDC input to 3.3V +/- 0.3VDC output

2.3.3 ESP32

Microcontroller must be able to receive signals over Wifi and decode. It should also be able to receive commands via a wired USB connection. The microcontroller will output a PWM to control the angle and speed of the servo motor.

Requirement #1: Must be able to output 3.3VDC PWM for necessary duration to the servo motor striking object.
Requirement #2: Must be able to receive and decode commands over 2.4GHz Wifi
Requirement #3: Receive and decode commands via USB.
Requirement #5: Must be able to decode messages into commands and execute with a consistent latency similar to the control unit.

2.3.4 LCD Display

LCD display will report the status of the device to the user Indicating when it is broadcasting its identification on the network, successfully initialization of Control Unit, and when battery is low. Would display when sending signal to the servo motor.

Requirement #1: Must be able to receive 3.3V signals from GPIO
Requirement #2: Must be powered at 3.3V similar to the microcontroller
Requirement #3 : Must be large enough to display strings of at least 20 characters

2.3.5 Servo motor

Servo motor will be equipped with a drumstick to hit objects. Powered by a 5V battery on the board and controlled by a PWM signal from the ESP32.

Requirement #1: Take 3.3VDC PWM from ESP32 to control the motor.
Requirement #2: Provide 1.51 kg.cm torque to strike objects.

2.3.6 Servo Mount

Servo must be securely attached to a mount that can hold flat items to strike with a drumstick that will stay in place for at least one dozen strikes before servo or item are displaced.

Requirement #1: Able to hold servo and object to strike for at least 12 strikes before manual realignment is required.

2.4 Ringing-type Board

The Ringing block will take commands from the Python API for turning on and off and for duration. The microcontroller will decode the command received via the Wifi adapter and signal using GPIO to turn on the ringing circuitry for that duration.

2.4.1 3.3V battery

Battery must supply 3.3VDC +/- 0.3VDC and 1A for powering the microcontroller and LCD display..

Requirement #1: Output 3.3VDC +/- 0.3VDC.
Requirement #2: Provide at least 1A at rated voltage.

2.4.2 ESP32

Microcontroller must be able to take and decode commands for enabling ringing and its duration.

Requirement #1: Must be able to output 3.3VDC PWM for decoded duration of sound.
Requirement #2: Must be able to receive and decode commands over 2.4GHz Wifi
Requirement #3: Receive and decode commands via USB.
Requirement #5: Must be able to decode messages into commands and execute with a consistent latency similar to the control unit.

2.4.3 LCD Display

LCD display will report the status of the device to the user Indicating when it is broadcasting its identification on the network, successfully initialization of Control Unit, and when battery is low. Would report time remaining for the board to actively ring.

Requirement #1: Must be able to receive 3.3V signals from GPIO
Requirement #2: Must be powered at 3.3V similar to the microcontroller
Requirement #3 : Must be large enough to display strings of at least 20 characters

2.4.4 Ringing circuit driver

Once we acquire a vintage phone, we need to reverse engineer it to see precisely what kind of control circuit is required, but typically the circuit consists of an H-bridge at 55VAC.

Requirement #1: Take 3.3V PWM from ESP32 and use it to toggle 55VAC across the H-bridge

2.4.5 Ringing Mechanism

Must be securely mounted onto board and be powered by the same power supply. Should be able to ring for at least 30 seconds continuously.

2.5 Risk Analysis

The main risk of the control unit is our method of communication. We need to allow wireless capabilities while also ensuring consistency typically found in wired systems. Furthermore, if several boards are going to run simultaneously, signals from the control unit need to be consistent

The main risk in a relay-type board is the mix of high-voltage AC and low-voltage DC. Since we are connecting to power outlets, we must also make sure that all parts of the relay and terminal rail are covered for user safety. Additionally, since the physical sound devices will be moved often, we should be sure that connections are reliable.

Possible risk in a servo-type board is that the mount will not be physically good enough, which is a mechanical failure that can make the block obsolete.

The risk in a ringing-type board is that we are still taking apart an old phone to figure out exactly how the mechanism works and how it can be used by the microcontroller. The requirements of the board may change if the mechanism is not compatible with the current design.

3 Ethics and Safety

Since the relay-type board will be connected to the wall power, we have to care about any components that are connected to this board since there is a possibility of getting an electric shock. In order to avoid this, we are planning to use a terminal block (aka terminal rail) to avoid any kinds of poor connected wires. Using it will lead to a convenient and safer way to distribute power from a single input source of the wall power to multiple outputs. We are responsible for our design and safety, and this safety concern is an implementation of the IEEE Code of Ethics Section I.1, “disclose promptly factors that might endanger the public or the environment” [2]. We are also considering ordering a commercial relay board to separate between sensitive components like the microcontroller and the AC power.

References

[1] Corey, Ryan M., and Ryan M. Corey. “Cooperative Listening Devices.” *Innovation in Augmented Listening Technology*, 16 Dec. 2019, publish.illinois.edu/augmentedlistening/cooperative-listening-devices/.

[2] “IEEE Code of Ethics.” *IEEE*, www.ieee.org/about/corporate/governance/p7-8.html.