Technology
Arts Sciences
TH Köln

Projektdokumentation

“BrushTalk”

Cosima Hiromi Zink, Mia Charlotte Henrichsmeyer, Ricardo Timm

Modul:	Internet of Things
Semester:	4. Semester
Abgabedatum:	31.08.2024
Betreuer:in:	Prof. Dr. Matthias Böhmer
	Vimal Darius Seetohul

Inhaltsverzeichnis

1. Einleitung **2**
2. Das Produkt **2**
3. Systemarchitektur und verwendete Technologien **3**
3.1 Systemarchitektur 3
3.2 Verwendete Hardware 3
4. Implementierung **5**
4.1. ESP32-Programm 5
4.2. Server Code 5
5. Teamarbeit und individuelle Beiträge **6**
6. Fazit, Ausblick und Reflexion **6**
6.1 Projektergebnisse 7
6.2 Herausforderungen und Lösungsansätze 7
6.3 Zukünftige Verbesserungen 7
6.4 Ausblick 8
7. Literaturverzeichnis **9**
8. Anhang **9**
Projektbilder 9

1. Einleitung

Wie können Freundschaften mit geografischer Distanz weiterhin eng verbunden bleiben? Diese Frage hat sich das Team des Verbundprojekts VREUNDE gestellt. Das Projekt baut auf dem Gedanken auf, dass Freunde der “große Verlierer der Corona-Pandemie”(Probst, 2023) sind.
Im Zuge unseres Moduls “Internet of Things” haben wir uns in kleineren Gruppen ebenfalls mit dieser Thematik beschäftigt.

Wir, vom Team BrushTalk, haben uns überlegt, dass es diesen Freunden meist an täglichem Austausch mangelt und oft kleine Dinge, die man gerne teilen würde, in der Distanz verloren gehen. Besonders bei Freunden, die einst zusammen gewohnt haben, zum Beispiel in einer Wohngemeinschaft, fällt durch eine räumliche Trennung sehr viel weg. Alltägliche Dinge, die man vielleicht einst gemeinsam gemacht hat, bleiben aus, Dinge wie zum Beispiel einen Einkauf im Supermarkt, ein gemeinsames Frühstück oder auch das Zähneputzen. Zu diesem Zweck haben wir das Produkt BrushTalk entwickelt.

2. Das Produkt

Durch das Produkt können Freunde für einander Sprachnachrichten aufnehmen, die sich der andere während des Zähneputzens anhören kann. Diese Nachrichten werden dann von dem Gerät an das des Freundes gesendet. Die Länge der Nachrichten beträgt immer drei Minuten, sodass es perfekt auf das Zähneputzen zugeschnitten ist.
Jedes Gerät verfügt über eine benutzerfreundliche Oberfläche, ein hochwertiges Mikrofon und einen Lautsprecher für das einfache Aufnehmen, Senden und Wiedergeben von Nachrichten. Zusätzlich ist das Gerät mit zwei LED-Buttons ausgestattet, die dem Nutzer das Aufnehmen und Abspielen von Nachrichten ermöglichen. Der rote LED-Button wird betätigt, wenn der Nutzer eine Aufnahme startet und leuchtet während der Dauer der Aufnahme. Der blaue LED-Button zeigt durch ein blinkendes Lichtsignal an, dass eine neue Aufnahme eingegangen ist. Außerdem leuchtet er, während eine Aufnahme abgespielt wird.

Das Produkt kann bequem mithilfe von Saugnäpfen am eigenen Badezimmerspiegel befestigt werden, sodass dem Nutzer in jedem Fall auffällt, wenn eine neue Nachricht eingegangen ist und diese direkt abgespielt werden kann.

Abbildung 1: BrushTalk Device. KI-generiert

BrushTalk soll die banale Aufgabe des Zähneputzens in eine Gelegenheit für Freunde verwandeln, engeren Kontakt zu halten und die tägliche Routine angenehmer gestalten.

3. Systemarchitektur und verwendete Technologien

3.1 Systemarchitektur

Das fertige Gerät besteht aus dem ESP32-Mikrocontroller und dem Webserver, welcher die gesendeten Audiodateien entgegennimmt, speichert und zum Download bereitstellt.
Dazu kommt noch die Hardware, die für das Aufnehmen und Abspielen der Aufnahmen sowie das Lichtsignal bei Eingang einer neuen Nachricht zuständig ist. Dafür muss jedes Gerät über ein Mikrofon, einen Lautsprecher, zwei LED-Buttons und eine SD-Karte mit passendem Modul verfügen.

3.2 Verwendete Hardware

Hardware

Komponente	Anzahl	Beschreibung
ESP32-Mikrocontroller	2	Hauptsteuergerät
Kondensatormikrofon	2	Zur Aufnahme von Audiosignalen (verstärkt durch den MAX9814 Mikrofonverstärker)

Lautsprecher	2	8 Ohm, 20 mm Radius, 0,5 Watt Lautsprecher zur Audioausgabe
Grove Red/Blue LED Button	2	LED-Taster zur Steuerung der Audioaufnahme/-wiedergabe, mit LED zur Anzeige des Zustandes
SD-Karte	2	Speichermedium 16 GB, zur lokalen Speicherung der aufgenommenen Audiodateien
SD-Karten-Modul	2	Schnittstelle zwischen ESP32 und Sd-Karte, ermöglicht das Lesen und Schreiben von Dateien

Die Aufnahme läuft über einen DEBO AMP MIC2 Mikrofonverstärker. Dieser ist für Sprachaufnahmen geeignet und bietet eine Verstärkungsregelung. Dieses Mikrofon haben wir über Reichelt gefunden und bestellt.

Für die Wiedergabe der Sprachaufnahmen verfügt das Gerät über einen Miniaturlautsprecher mit einem Widerstand von 8 Ohm, einem Durchmesser von 20 mm und einer Leistung von 0,5 Watt. Dieser kompakte Lautsprecher ist ausreichend für einen Prototypen für unseren Zweck. Man kann das Gesagte am anderen Ende verstehen, allerdings ist es sehr übersteuert und begleitet von Rauschen.

Audiodaten vom Mikrofon werden über GPIO 34 eingelesen, während die Ausgabe der Audiodaten an den Lautsprecher über GPIO 16 (Data Out) und GPIO 17 (Bit Clock) erfolgt.

Die Aufnahme- und Abspielbuttons werden durch Grove LED Buttons realisiert. Der rote Knopf steht für die Aufnahme und zeigt an, wenn eine Aufnahme aktiv ist. Der blaue Knopf steht für die Wiedergabe. Er blinkt, sobald eine neue Nachricht eingeht und er leuchtet stetig, während eine Aufnahme abgespielt wird. Diese Knöpfe sind einfach in der Handhabung und visuell leicht erkennbar, wodurch sie ideal für den Einsatz in diesem Projekt sind.

Außerdem muss das Gerät die Aufnahmen zwischenspeichern können, wenn die Aufnahmen vom Webserver heruntergeladen wurden. Dazu wird der Mikrocontroller noch an ein SD-Karten-Modul angeschlossen. In diesem Fall haben wir das DEBO MICROSD 2 Entwicklerboard verwendet. Dieses kann Mikro SD Karten lesen und beschreiben.

Bezogen auf die Softwaretechnologie verwenden wir die Arduino IDE zur Programmierung des ESP32. Dieser nutzt die I2S-Schnittstelle für die Audioverarbeitung.

Zudem implementieren wir den Node.js Server, welcher auf dem eigenen Rechner läuft und über Serveo erreichbar ist, zur Verwaltung der Dateiuploads und -downloads. Des Weiteren bietet dieser einen Endpunkt zum Überprüfen von Audiodateien.

4. Implementierung

4.1. ESP32-Programm

Das Programm auf dem ESP32 umfasst die folgenden Hauptfunktionen:

Der ESP32 verbindet sich automatisch mit einem vordefinierten WLAN-Netzwerk, um die Kommunikation mit dem Server zu ermöglichen. Dies stellt sicher, dass das Gerät jederzeit in der Lage ist, Daten zu senden und zu empfangen.
Über die I2S-Schnittstelle werden Audiodaten vom angeschlossenen Mikrofon erfasst. Diese Daten werden in Echtzeit verarbeitet und auf einer SD-Karte im WAV-Format gespeichert. Diese Datei wird später zum Upload auf den Server verwendet.
Die gespeicherte Audiodatei wird mittels einer HTTPS-Verbindung an einen zentralen Webserver übertragen. Hierbei werden gerätespezifische Informationen übermittelt, um die Datei im richtigen Verzeichnis auf dem Server abzulegen.
Der ESP32 prüft regelmäßig, ob neue Audiodateien auf dem Server verfügbar sind. Wird eine solche Datei gefunden, wird sie heruntergeladen und auf der SD-Karte gespeichert, um später wiedergegeben zu werden. Die Audiodatei wird nach dem Download ebenfalls vom Webserver gelöscht, um ein doppeltes Herunterladen zu vermeiden.
Nach dem Download wird die Audiodatei von der SD-Karte gelesen und über den Lautsprecher abgespielt. Dies geschieht ebenfalls über die I2S-Schnittstelle. Anschließend wird die Audiodatei von der SD-Karte gelöscht.

4.2. Server Code

Der Server, basierend auf Node.js, übernimmt die Verwaltung und Verarbeitung der Audiodateien und stellt folgende Funktionen bereit:

Der Server empfängt die vom ESP32 hochgeladenen Audiodateien und speichert sie in gerätespezifischen Verzeichnissen. Dabei wird sichergestellt, dass die Dateien korrekt zugeordnet werden und auf Anfrage verfügbar sind.

Auf Anforderung des ESP32 stellt der Server die gewünschten Audiodateien zur Verfügung. Nach erfolgreicher Übertragung wird die Datei vom Server gelöscht, um Speicherplatz zu sparen und sicherzustellen, dass nur aktuelle Dateien vorhanden sind. Deshalb ist auch keine mehrfache Wiedergabe der Nachrichten möglich.

Der Server bietet eine Funktion zur Überprüfung, ob neue Audiodateien in den gerätespezifischen Verzeichnissen vorhanden sind. Diese Überprüfung ermöglicht es dem ESP32, effizient festzustellen, ob neue Daten verfügbar sind, die heruntergeladen werden können.

5. Teamarbeit und individuelle Beiträge

Arbeitsmatrix

Aufgabenbereich	Ricardo T.	Mia H.	Cosima Z.
Hardware Integration	33%	33%	33%
ESP32-Programmierung	33%	33%	33%
Server-Programmierung	33%	33%	33%
Debugging und Testen	33%	33%	33%

Wir haben alle Aufgabenbereiche der Arbeitsmatrix gemeinsam bearbeitet, was sich in der gleichmäßigen Verteilung der Arbeit widerspiegelt. Sowohl bei der Verbindung und Konfiguration der Hardware Komponenten mit dem ESP32, der Implementierung aller wichtigen Bestandteile, der Entwicklung und Konfiguration des Servers, als auch bei der Fehleranalyse und -behebung haben wir jeweils zu gleichen Teilen beigetragen. Dies zeigt, dass die Arbeit im Team gleichmäßig verteilt war und jeder gleichermaßen Verantwortung in den verschiedenen Bereichen übernommen hat. Dadurch konnten wir die Aufgaben effizient angehen und sicherstellen, dass alle Aspekte des Projekts gemeinsam erarbeitet wurden.

6. Fazit, Ausblick und Reflexion

Durch den Einsatz von ESP32-Mikrocontrollern in Kombination mit der I2S-Schnittstelle sowie einer serverseitigen Infrastruktur mithilfe von Node.js konnten wir eine stabile und funktionale Lösung entwickeln, die es ermöglicht,

Audiodateien zwischen zwei Geräten auszutauschen, diese auf einem zentralen Server zu speichern und anschließend wieder abzuspielen.

6.1 Projektergebnisse

Ein voll funktionsfähiger Prototyp zur Audio Übertragung und -wiedergabe zwischen zwei ESP32-Geräten wurde erfolgreich implementiert. Der Prototyp nutzt WLAN für die Kommunikation und ermöglicht die Speicherung und Verwaltung von Audiodateien auf einem zentralen Webserver.
Im Verlauf des Projekts konnten tiefgehende Kenntnisse in mehreren technischen Bereichen gewonnen werden, darunter die I2S-Audiotechnologie, die serverseitige Dateiverwaltung und die Programmierung von Mikrocontrollern. Diese Erfahrungen bieten eine solide Grundlage für zukünftige Projekte in ähnlichen Bereichen.
Die erfolgreiche Integration von Hardware (Mikrofone, Lautsprecher, SD-Karten) und Software (Netzwerkkommunikation, Dateihandling) zeigt, dass ein solches System nicht nur theoretisch, sondern auch praktisch umsetzbar ist.

6.2 Herausforderungen und Lösungsansätze

Während der Projektumsetzung traten einige Herausforderungen auf, insbesondere in Bezug auf die Synchronisation von Audioaufnahmen, die Optimierung der Netzwerkkommunikation und die Handhabung von Dateiformaten. Beispielsweise wurde der erste Ansatz, die Dateien über MQTT zu senden, nach vielen Problemen mit dem Buffer und der Zusammensetzung der Chunks, verworfen. Bei unserem neuen Ansatz, die Audiodateien über einen Webserver zu senden, lief es deutlich reibungsloser, abgesehen von ein paar Problemen mit dem Herunterladen der Dateien. Durch mehrere Tests und Anpassungen konnten diese Herausforderungen jedoch erfolgreich gemeistert werden. Die kontinuierliche Fehlersuche war etwas nervenaufreibend, hat jedoch im Endeffekt zu einer qualitativen und stabilen Lösung geführt.

6.3 Zukünftige Verbesserungen

Obwohl die aktuelle Implementierung eine solide Basis bietet, besteht noch Potenzial zur Verbesserung der Audioqualität. Durch die Verwendung eines qualitativ hochwertigeren Mikrofons und Lautsprecher könnte man einen klareren Ton erreichen. Zukünftige Projekte könnten die Einführung erweiterter Audioverarbeitungsfunktionen wie beispielsweise Echtzeit-Equalizer oder Kompression beinhalten. Diese könnten auf dem Mikrocontroller oder serverseitig implementiert werden, um den Funktionsumfang des Systems zu erweitern.

Eine benutzerfreundlichere Oberfläche zur Verwaltung der Audiodateien und zur Steuerung der ESP32-Geräte könnte das System intuitiver machen und den Bedienkomfort erhöhen. Dies könnte durch die Entwicklung einer grafischen Benutzeroberfläche (GUI) erreicht werden, die es den Benutzern ermöglicht, Audiodateien einfach hochzuladen, herunterzuladen und zu verwalten. Außerdem benötigt das Gerät eine wasserfeste Hülle, damit es im Badezimmer angebracht werden kann und bei eventuellen Kontakt mit Wasser funktionsfähig bleibt.

Eine detailliertere und robustere Fehlerbehandlung wäre ein weiterer wichtiger Aspekt für die zukünftige Entwicklung, da wir zu dem jetzigen Zeitpunkt hauptsächlich Logs in den Konsolen nutzen. Dies könnte die Implementierung von Mechanismen zur Erkennung und Korrektur von Übertragungsfehlern, zur automatischen Wiederherstellung nach Verbindungsabbrüchen und zur präziseren Diagnose von Problemen beinhalten.

6.4 Ausblick

Dieses Projekt legt den Grundstein für weitere Entwicklungen im Bereich der drahtlosen Audioübertragung und -verarbeitung. In einem erweiterten Projekt könnte der Fokus auf der Skalierbarkeit des Systems liegen, um mehr als zwei Geräte zu integrieren oder eine größere Anzahl von Audiodateien gleichzeitig zu verwalten. Eine Integration in IoT-Umgebungen, wo solche Systeme in größeren Netzwerken zur Audioübertragung eingesetzt werden, ist ebenfalls denkbar.
Schließlich könnten zukünftige Iterationen dieses Projekts die Einbindung von Cloud-Diensten zur Speicherung und Verarbeitung der Audiodateien erforschen, was die Flexibilität und Funktionalität des Systems weiter erhöhen würde.

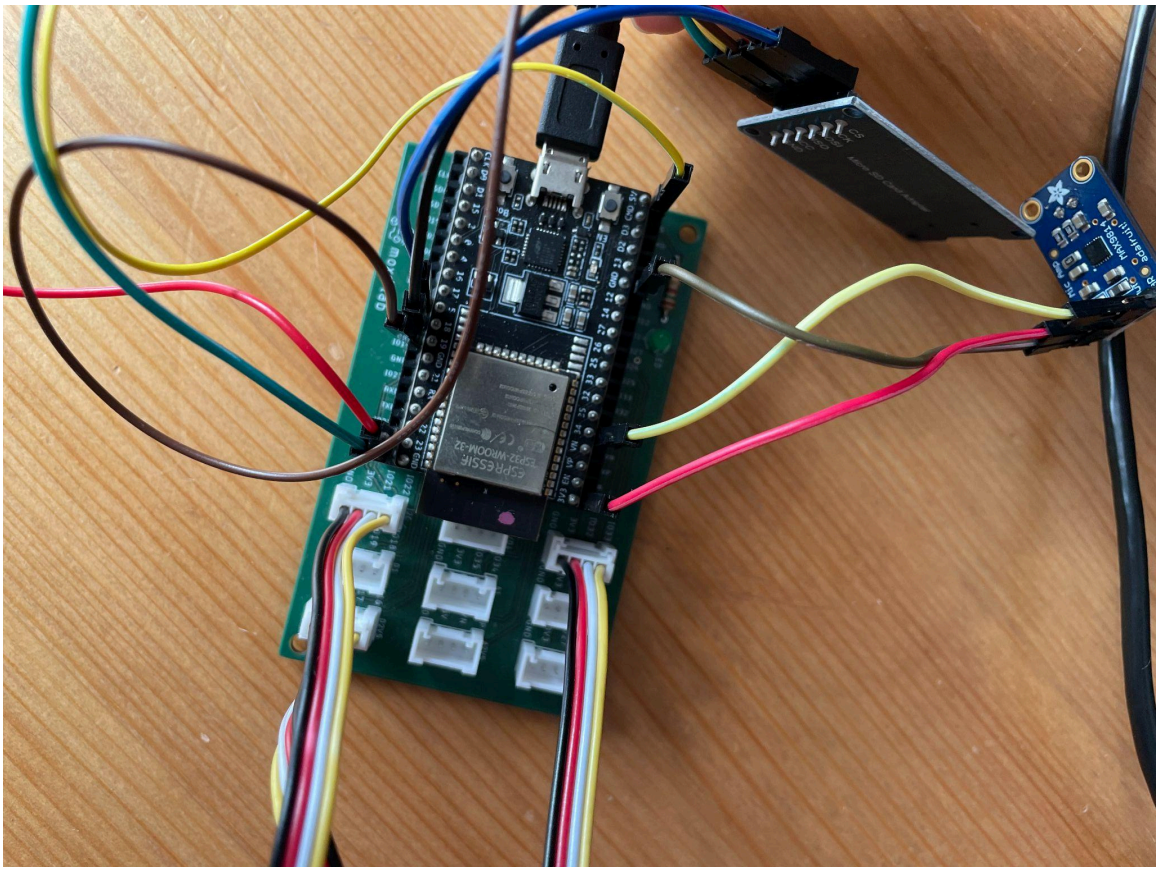
7. Literaturverzeichnis

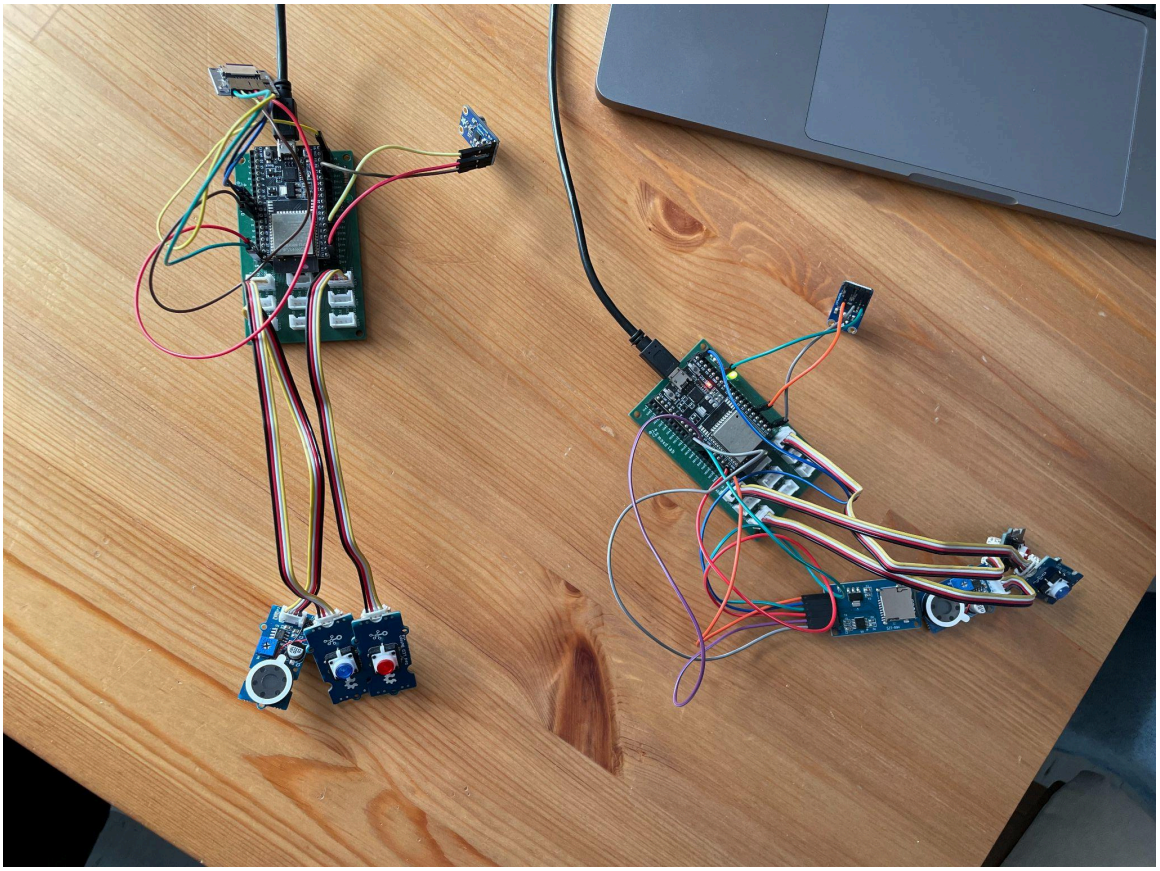
Probst, Monika (2023): VREUNDE bleiben, TH Köln Website, [online] https://www.th-koeln.de/hochschule/start-forschungsprojekt-vreunde_105385.php [abgerufen am 28.08.24].

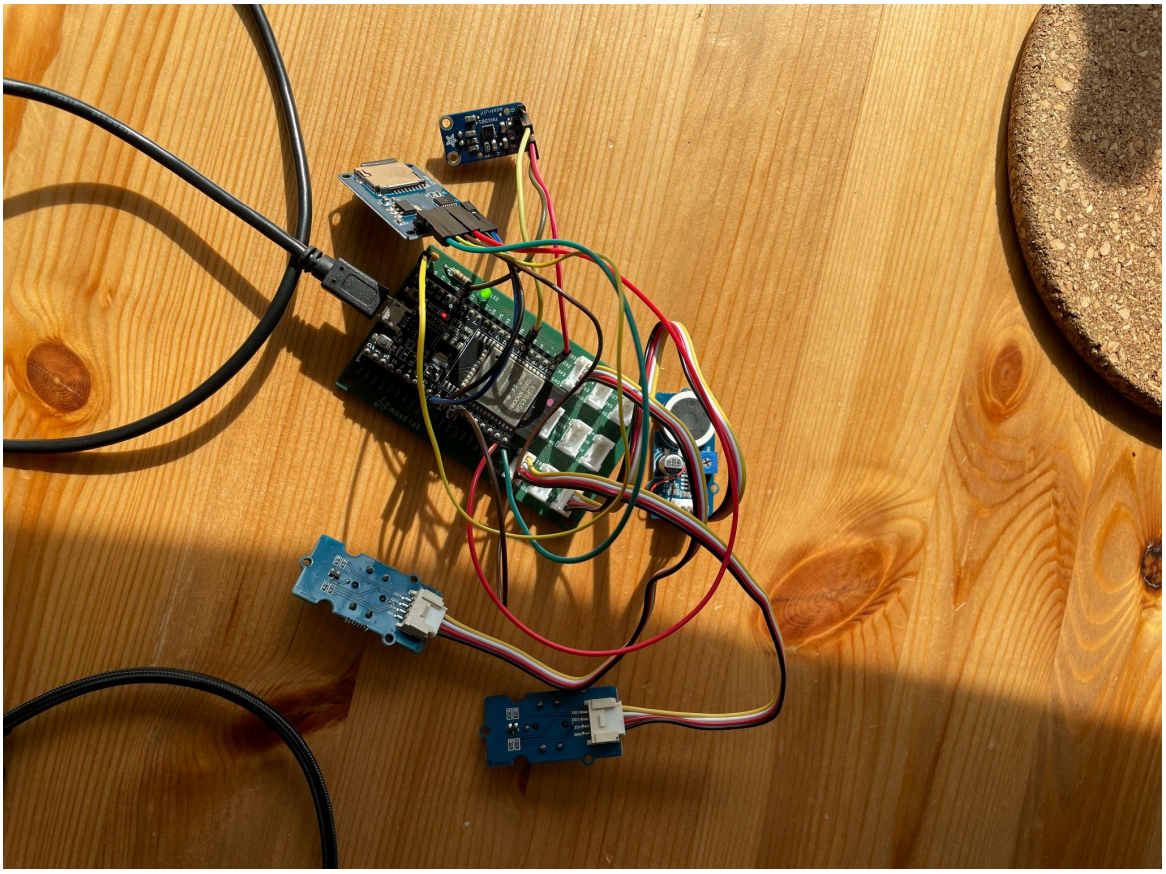
8. Anhang

GitHub-Repo: https://github.com/ricardotimmr/iot-brushtalkdevice
DIY Anleitung:
https://github.com/ricardotimmr/iot-brushtalkdevice/wiki/DIY-Anleitung-%E2%80%90-BrushTalk

Projektbilder

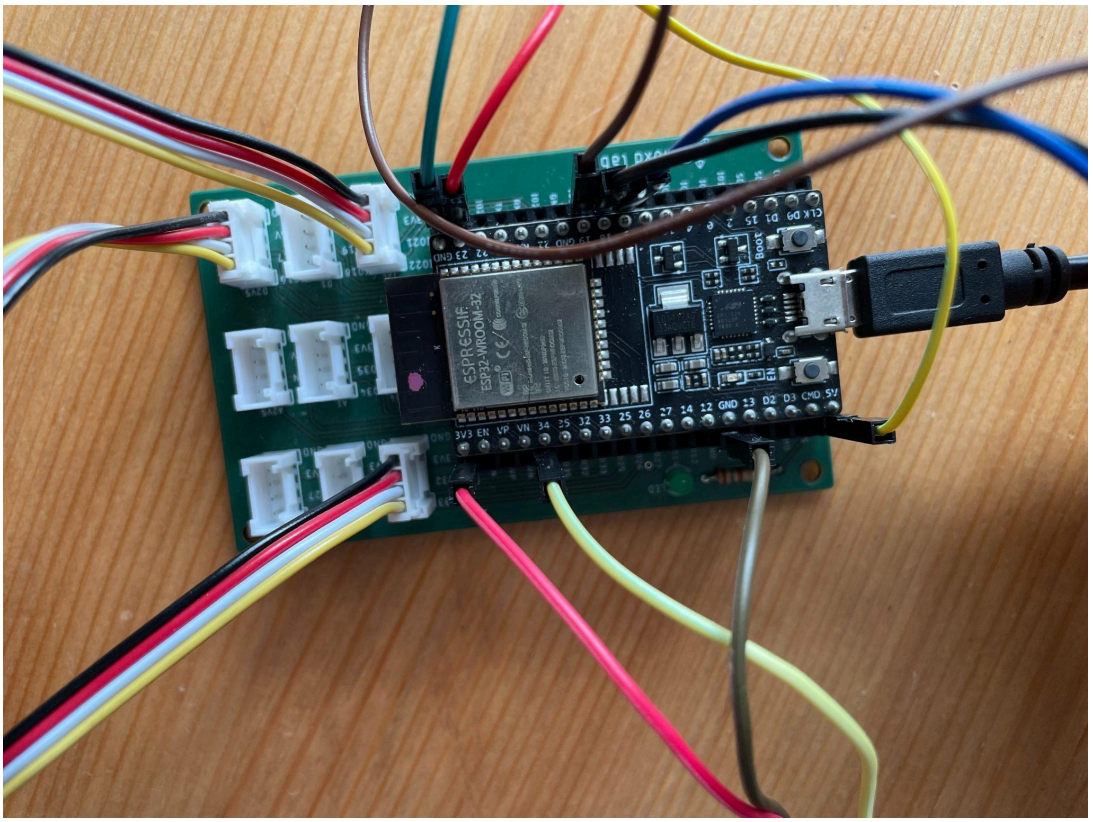


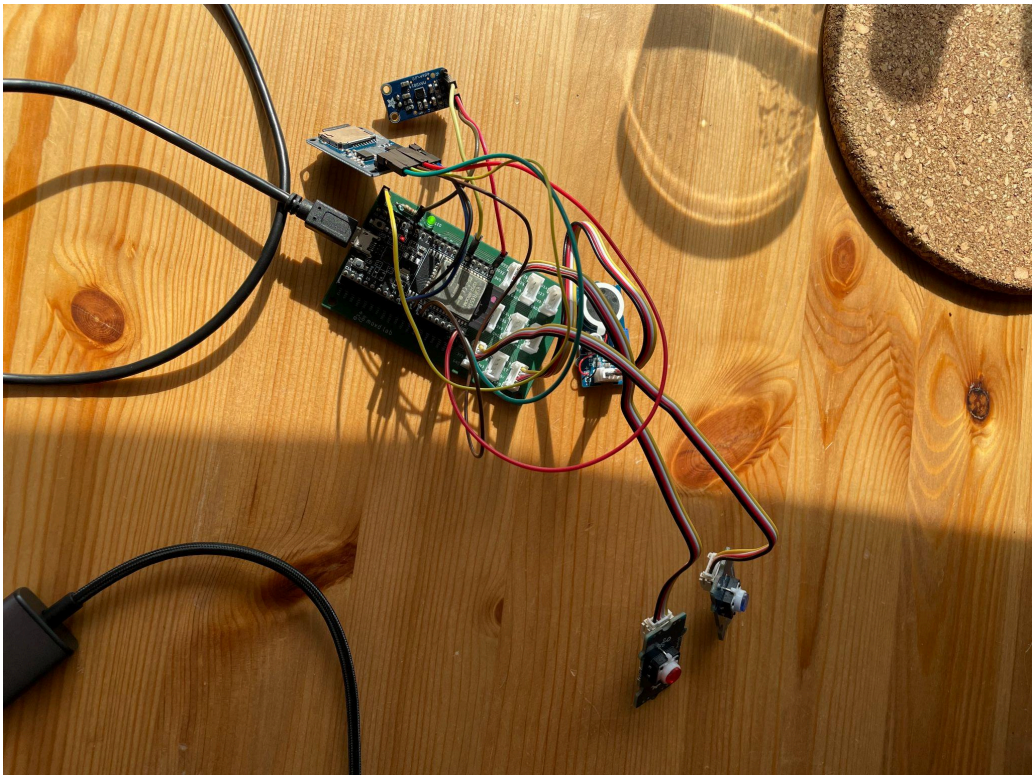


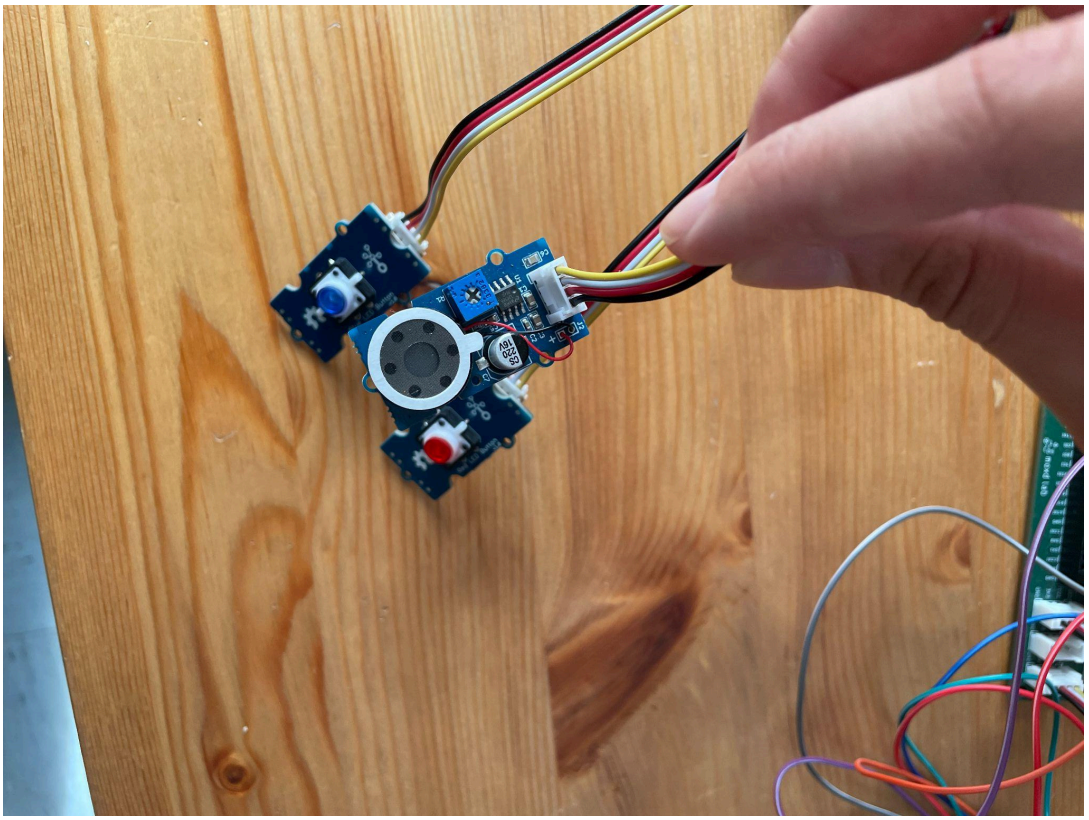


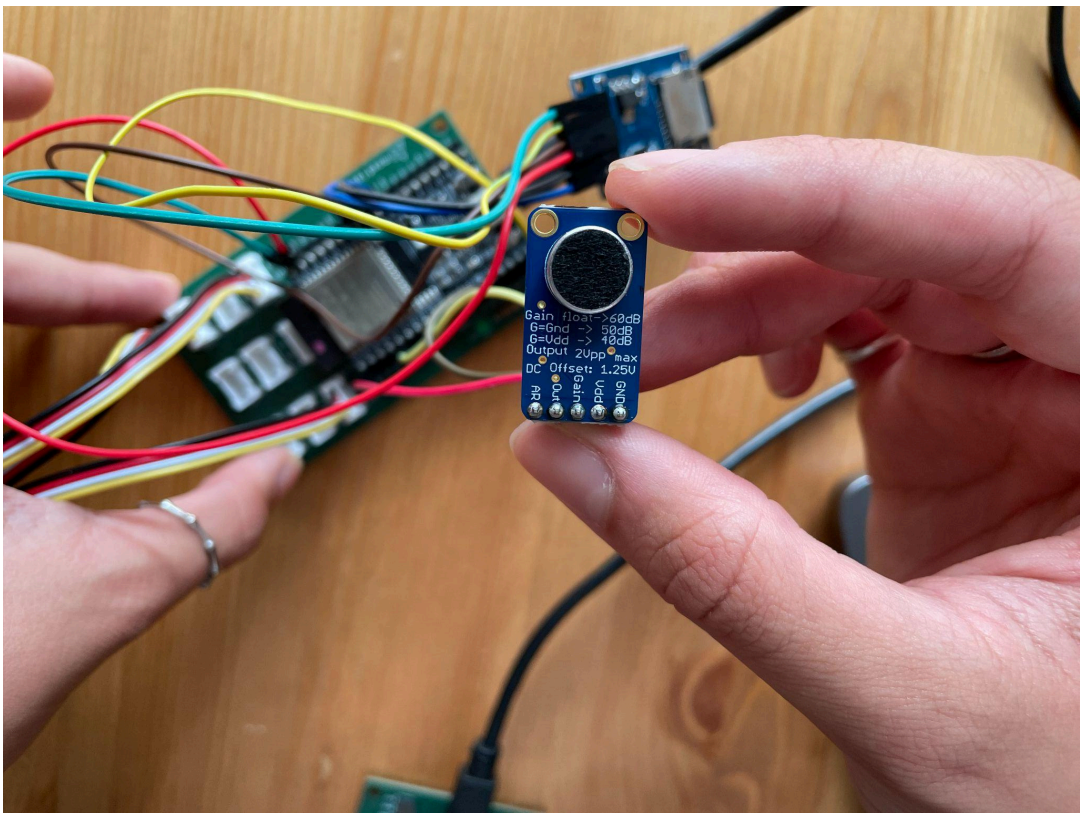
```
audio-server — node server.js — 115×34
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device2...
Available files: []
No new audio files found.
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device1...
Available files: []
No new audio files found.
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device2...
Available files: []
No new audio files found.
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device1...
Available files: []
No new audio files found.
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device2...
Available files: []
No new audio files found.
Received upload request
Headers: {
  host: '8ea3035bbf6b75532a241ec23480c4ee.serveo.net',
  'user-agent': 'ESP32HTTPClient',
  'content-length': '448512',
  'accept-encoding': 'identity;q=1,chunked;q=0.1,*;q=0',
  'content-type': 'audio/wav',
  'x-device-type': 'device2',
  'x-filename': 'uploaded_audio_device2.wav',
  'x-forwarded-for': '78.48.73.233',
  'x-forwarded-host': '8ea3035bbf6b75532a241ec23480c4ee.serveo.net',
  'x-forwarded-proto': 'https'
}
Content-Type: audio/wav
File uploaded successfully: /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device2/uploaded
_audio_device2.wav
Total bytes written: 448512
Checking for new audio files in /Users/ricardotimm/Desktop/Uni/Internet of Things/audio-server/uploads/device1...
Available files: []
```

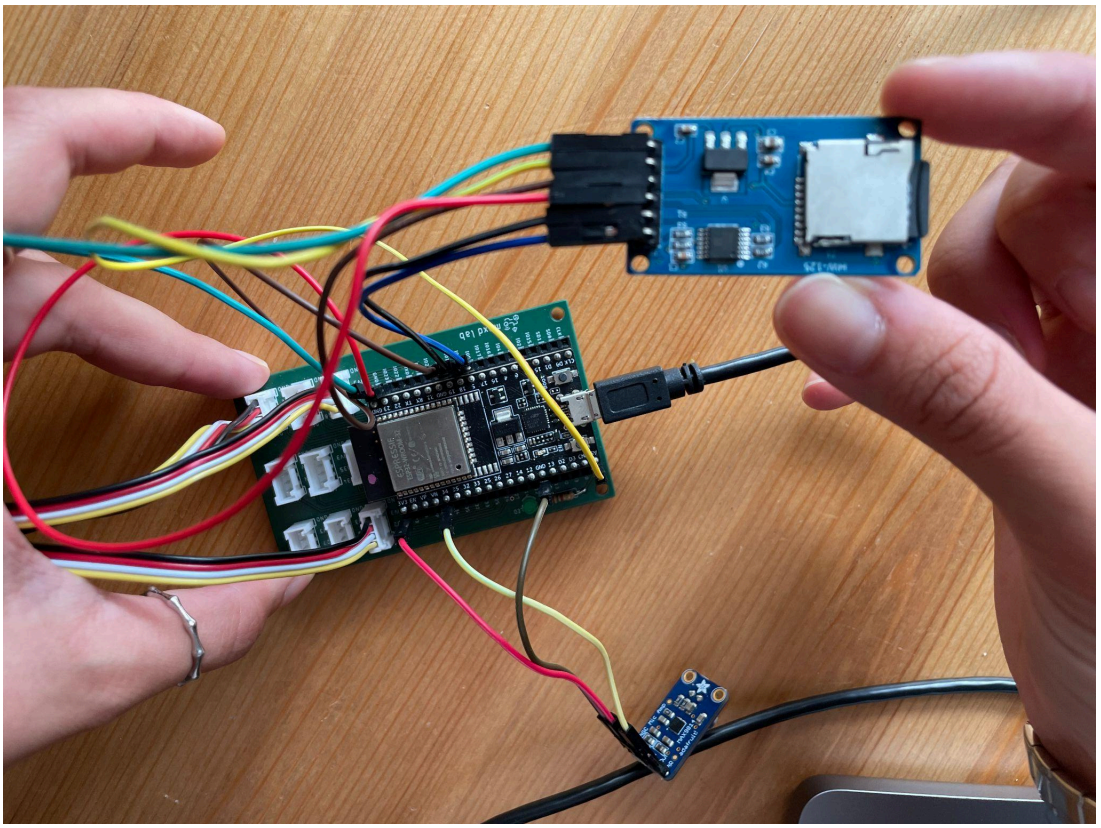
```
ricardotimm — ssh -R 80:localhost:80 serveo.net — 112×34
audio_device2.wav
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/download/device2/uploaded_
audio_device2.wav
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/upload
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/download/device2/uploaded_
audio_device2.wav
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/download/device2/uploaded_
audio_device2.wav
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device2
HTTP request from 78.48.73.233 to https://8ea3035bbf6b75532a241ec23480c4ee.serveo.net/check/device1
```

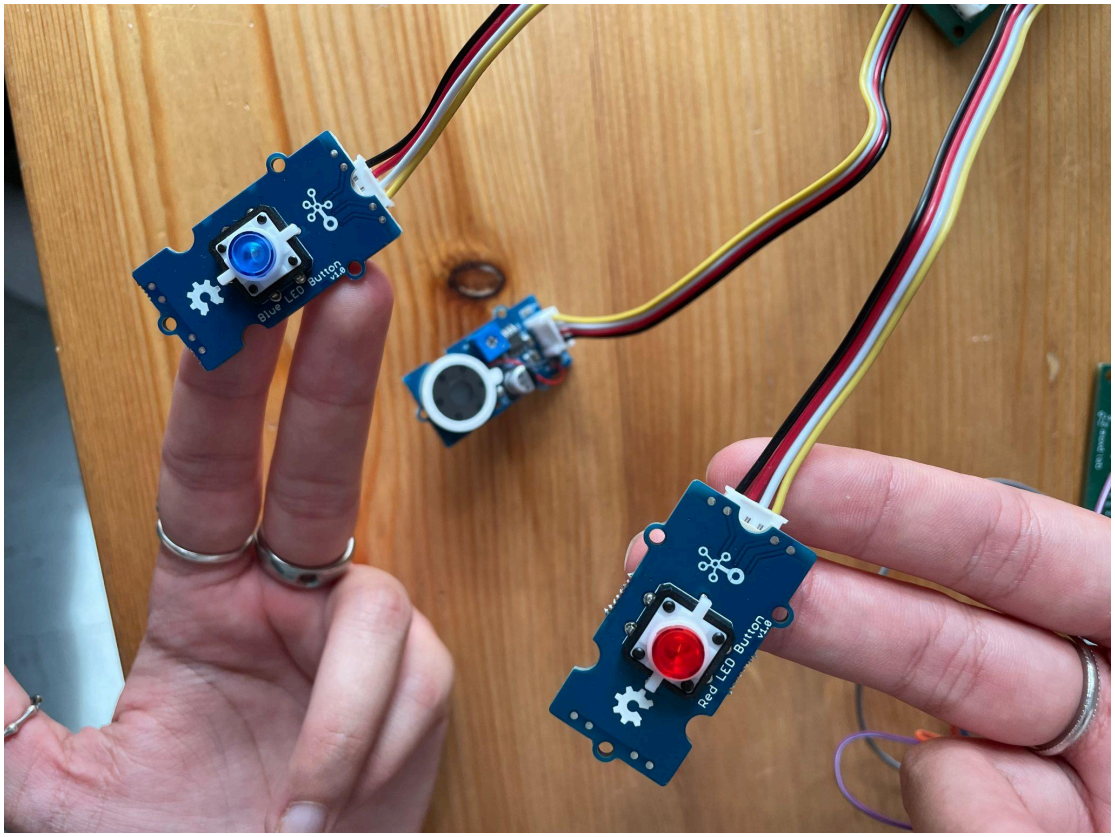







Gain float->60dB
G=Gnd -> 50dB
G=Vdd -> 40dB
Output 2Vpp max
DC Offset: 1.25V
AR
Out
Gain
Vdd
GND




Blue LED Button
v1.0
Red LED Button
v1.0