Real-time Audio Separation of Human Voices

Tainon Chen, Zhirui He, Nathan Hellstedt,
Haotian Qiao, Erik Sangeorzan, Yuxin Zhong

Presentation Overview

- Context and Other Approaches
- Methods
- System Overview
- Demonstration
- Encountered Challenges & Solutions
- Poster

Context: Audio Source Separation

- Goal: separate simultaneous human speech inputs into isolated output channels in real time
- If you’re in a noisy room, you can block out extraneous noise and focus on your conversation
  - Cocktail party problem
  - Difficult for computers to filter out unwanted noise
- Applicable to applications such as hearing aids, wireless communication
  - In general, an underdetermined problem
  - State of the art solutions are still being developed
- How can we isolate our desired signal?
  - Blind Source Separation (BSS)
- How can we do this in real-time?
  - Windowing & splicing with Independent Component Analysis (ICA)

Relevant Literature

- Nonlinear Blind Source Separation Using Vanishing Component Analysis
  - Uses MLP neural network to extract feature space
  - State of the art results for nonlinear audio inputs
  - Drawback: requires significant computing resources
- Blind spatial subtraction array (BSSA) that utilizes ICA as noise estimator
  - Post-filtering based on generalized spectral subtraction(SS)
  - Used to recover a single target source
  - Drawback: not suitable for real-time performance, recovers single target source
- Nonnegative Matrix Factorization (NMF)
  - Shift Invariant NMF (unsupervised, many instruments can be well approximated as shift-frequency invariant in a log scale representation, such as the constant-Q Transform)
  - NMF-based methods take advantage of harmonic structure of music
  - Drawback: heavy reliance on machine learning; better suited for music

ICA for Blind Source Separation (BSS)

- ICA: Independent Component Analysis
- Makes the following assumptions:
  - Sources are independent and non-Gaussian
  - Mixed signals are linear combinations of sources
  - Same number of receivers (mixed input) as sources (Figure 1)

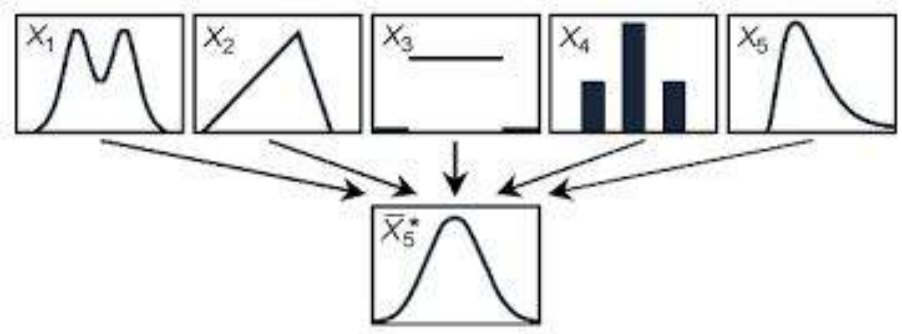


Figure 1: Diagram of mixed inputs as passed into ICA

ICA Methods (Negentropy) Detailed

- ICA is to solve linear combination (mixing) problems
- Preprocessing is performed on the mixed input signals
  - **Centralization**: make the averages zero
  - **Whitening**: use Principal Component Analysis (PCA) to transform the dataset and may reduce number of signals if need
- Perform set iterations of gradient ascent to maximize the non-Gaussianity of the signals, and obtain a transform matrix
  - Transform matrix maps the mixed signals to the original signals
- The demixing array will be passed to the next ICA as the beginning demixing array

This slide contains audio

ICA Results (Simulation):

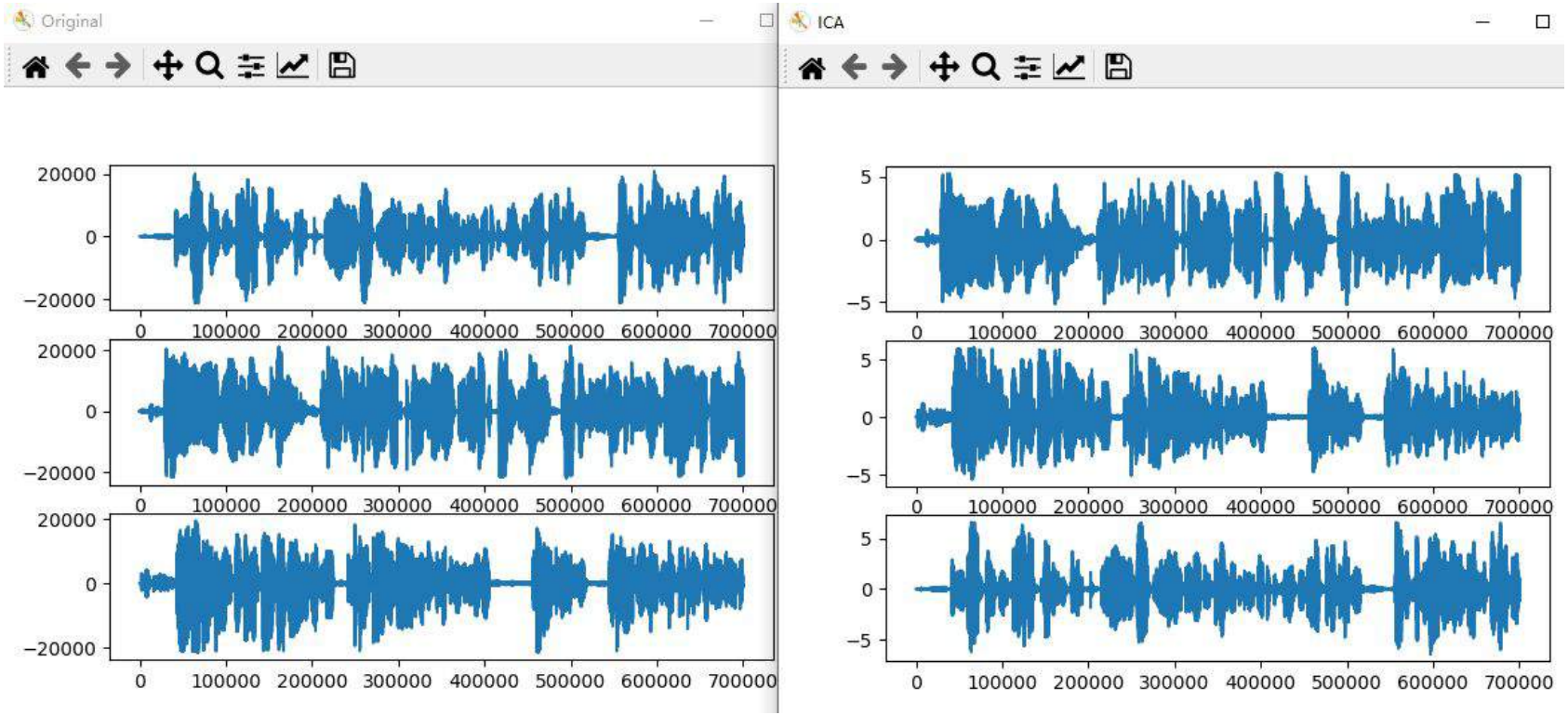


Figure 2: Waveforms of three input sources vs. ICA results

Software Layout

- ALSA buffer underrun leading to noisy output audio
- Problematic with usage of multithreading
- Modified code to separate each operation
- Use .npy files to transfer data between operations

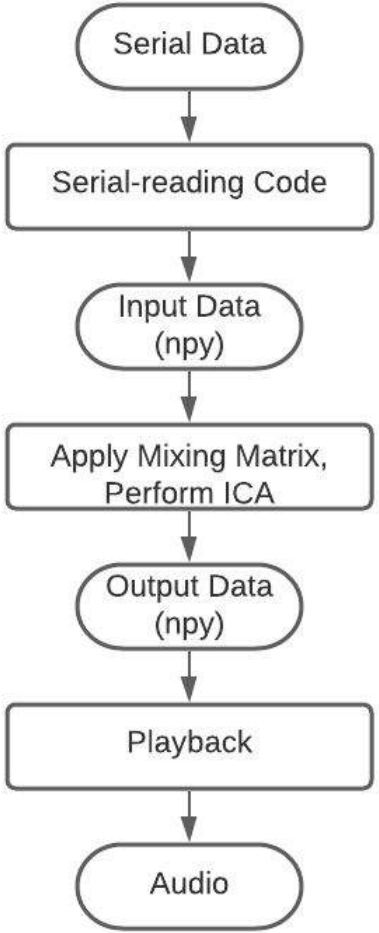


Figure 3: Raspberry Pi Software Layout

Demonstration (Preface)

- Upon execution, the code reads in audio data and begins source separation
- Each ICA output channel is plotted in real-time
- [Not included in demo]: The user may select a channel to play back (Figure 4)

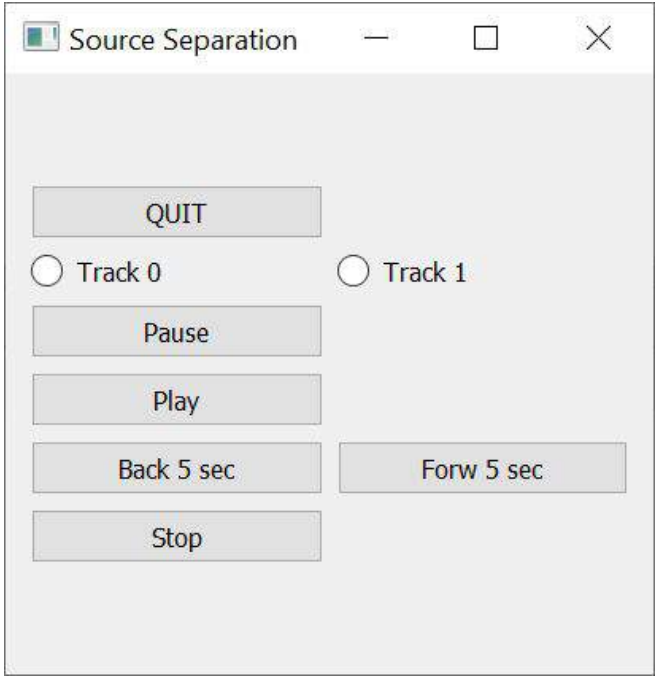
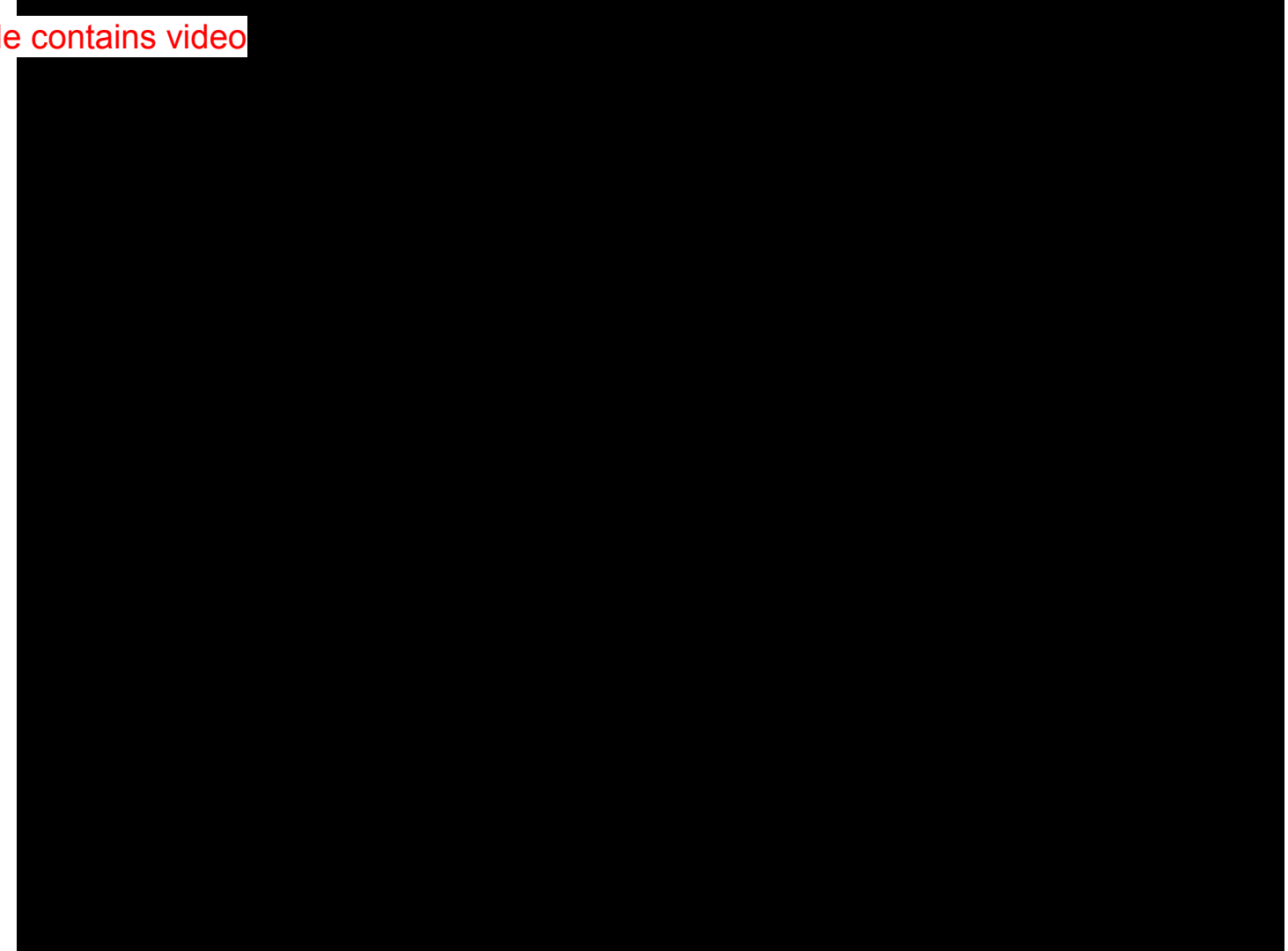


Figure 4: UI of the source separation program

This slide contains video



Encountered Challenges & Solutions (Hardware)

- Transferring serial audio data with sufficient speed
  - Lowpass Filter & 48kHz > downsample to 12 kHz and find a proper package size (384 B)
- Microphone incompatibility
  - Solder the mic input to the audio shield and use TRRS to TRS converter
- Receiving audio data from multiple Teensy boards in sync
  - Use a start pulse to engine all Teensys at the same time
  - Use multiprocessing method on Rpi side to receive at the same time
- Multiplexing
  - Connect each Teensy to a USB to TTL converter(CP2102) and use USB hub that handles it internally (Figure 5)

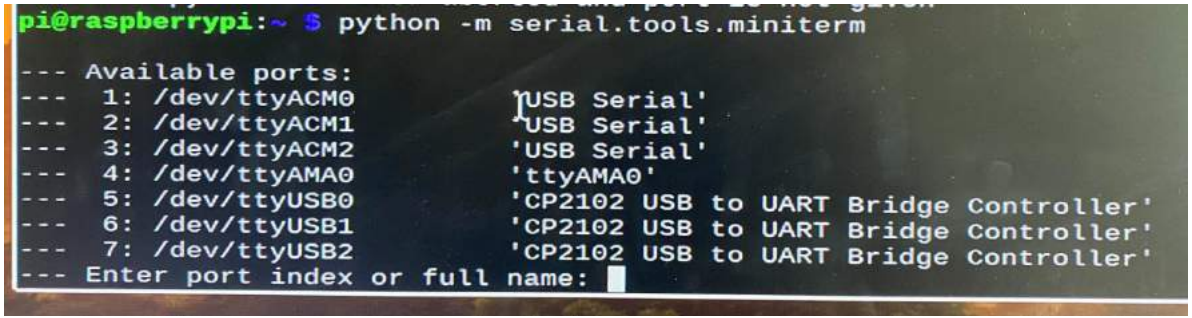


Figure 5: Terminal window showing separate input ports for each Teensy (USB Serial) and converter (CP2102)

This slide contains audio

Hardware Communication Result

fs=12 kHz, 16 bits per sample, the baud rate should be at least 12*16 kHz=192 kHz

500 kHz baud rate is chosen for serial data transmission

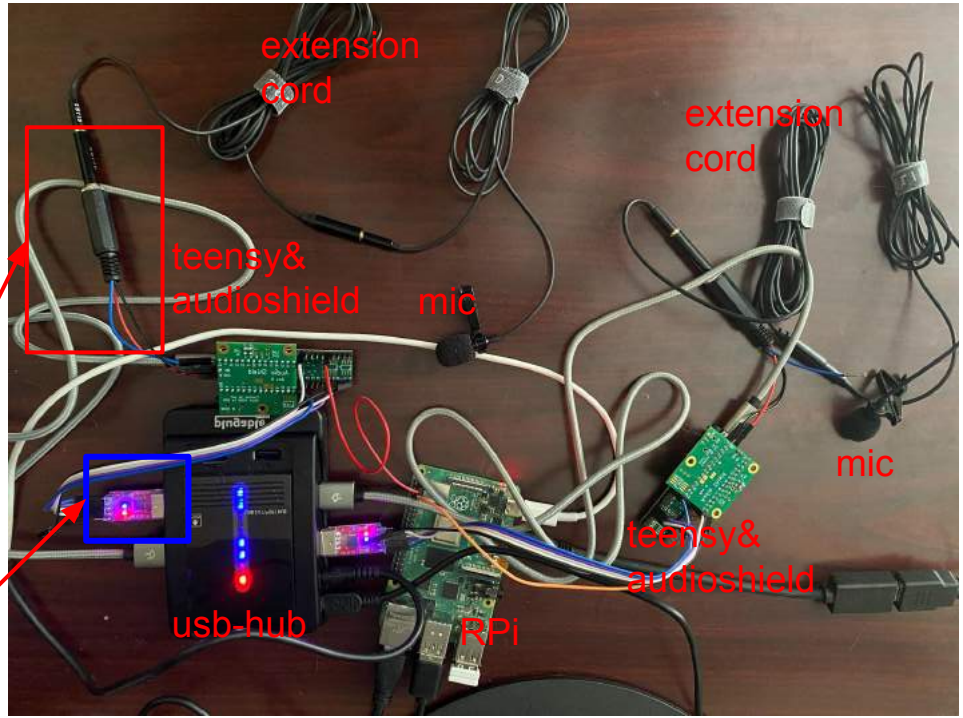


Figure 6: Photo of project hardware

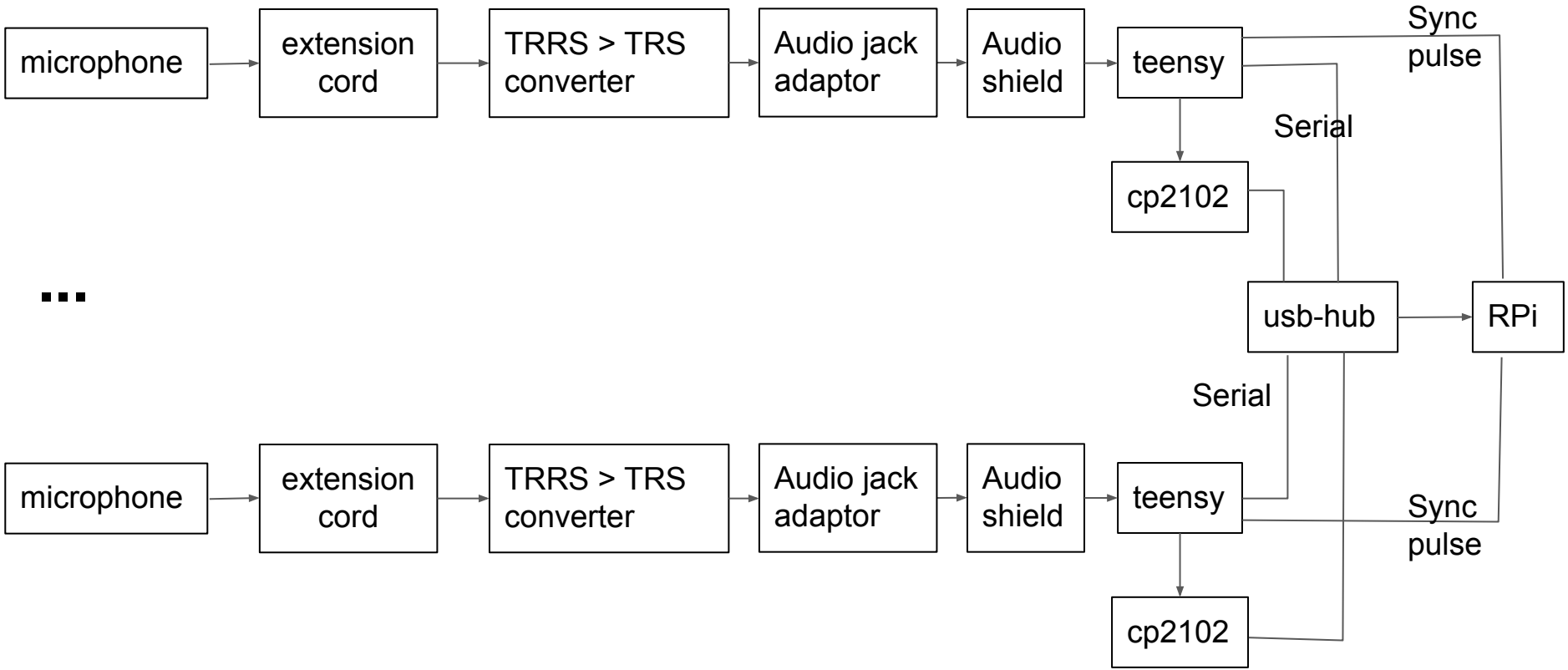


Figure 7: Diagram of System Hardware

This slide contains audio

What We Wish We Knew: ICA Limitations

- ICA makes many assumptions that rarely apply to real-world situations
  - ICA requires linear mixing but in reality, most recordings are nonlinear
  - ICA didn’t perform well on microphone recordings in our end-to-end system
    - Reverberation
    - Noise
    - Time delay
  - Current literature focuses on nonlinear, neural network-based solutions

Summary

- Transmit audio from multiple Teensys to Raspberry Pi at 12 KHz
  - Solved multiplexing issue
  - Little to no data loss
- ICA computation in real-time
  - Windowing and splicing
- Playback and plotting of output channels with minimal delay

Real-time Audio Separation of Human Voices

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY OF MICHIGAN

EECS 452: Digital Signal Processing Design Lab – Winter 2021
Tainon Chen, Zhirui He, Nathan Hellstedt, Haotian Qiao, Erik Sangeorzan, Yuxin Zhong

Introduction

Consider the cocktail party problem: humans can easily isolate a certain conversation in a noisy room, while computers cannot — we attempt to solve this problem, which is demonstrated in Figure 1. Our project aims to separate three individual human voices from mixed audio inputs. The hardware system we used in our project is shown in Figure 2.

Figure 1: Cocktail party problem

Independent Component Analysis (ICA)

Our project uses ICA to separate mixed audio into its individual sources. ICA uses the knowledge that the individual sources are non-gaussian and independent to isolate individual sources through the following algorithm:

1) Take N concurrent samples from each of the M microphones to form a MxN data array
2) Make the mean of each row in the array zero
3) Whiten the data to make the covariance between each array row zero
4) Iterate until a transform matrix that maximizes the non-gaussianity of each signal is found
5) Apply transform matrix to signal

System Architecture

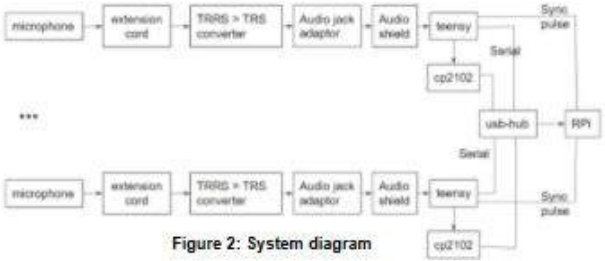


Figure 2: System diagram

Results / Evaluation

Our system was able to successfully transmit and separate linearly mixed audio, as seen in Figure 3. Audio recorded from the microphone was transmitted to the Raspberry Pi at 12 kHz. ICA performed well under ideal conditions, but when the mixed audio had nonlinearities, it didn't perform well. Our next step is to incorporate the handling of nonlinearity into our source separation algorithm.

ICA Simulation Results

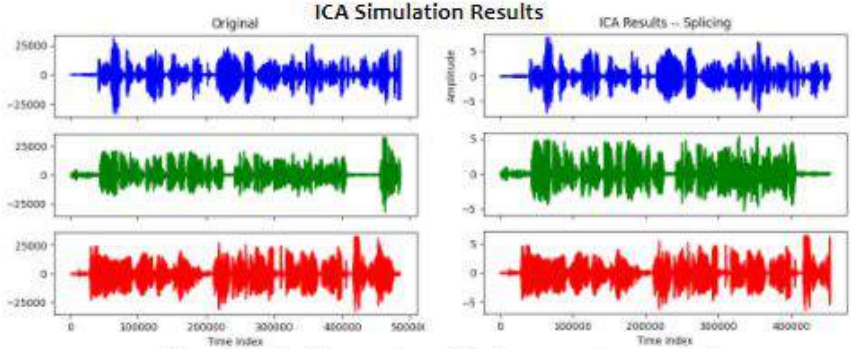


Figure 3: ICA simulation results — original vs. recovered source signals

Innovation

The main innovation of our project is performing source separation in real time. This was accomplished using our method of windowing and splicing to separate the signals. We also developed a user interface to playback and plot separated audio with minimal real-time delay.

Acknowledgements

We would like to thank Professor Shai Revzen, David Kucher, and Rishank Nair for their guidance and help on this project. Additionally, we recognize the ECE undergraduate program for providing funding for this project.