SDP Team 23 CDR Presentation

Phillip Ashe, Liam Shea,
Jason Nguyen, Paul Mahoney

Speaker Verification System (SVS)

Phillip Ashe-CSE

Database development and access code

Liam Shea - EE

Feature extraction and analysis

Paul Mahoney-CSE

Feature extraction and analysis

Jason Nguyen - EE

Feature extraction and analysis

Hardware implementation

The Problem

Security, security, security…

- Always a need for novel forms of user verification
- Traditional methods (text passwords) are easily forgotten and can be guessed

Privacy

- Some users may find other forms of biometric verification intrusive

Accessibility

- Traditional modes of input (e.g. keypads) may not be usable by everyone

The Solution

Voice Biometrics

- Usable by those with motor impairments
- Only need to remember a phrase instead of string of characters
- Voice is unique to each user
- Not easily duplicated as in the case of fingerprint

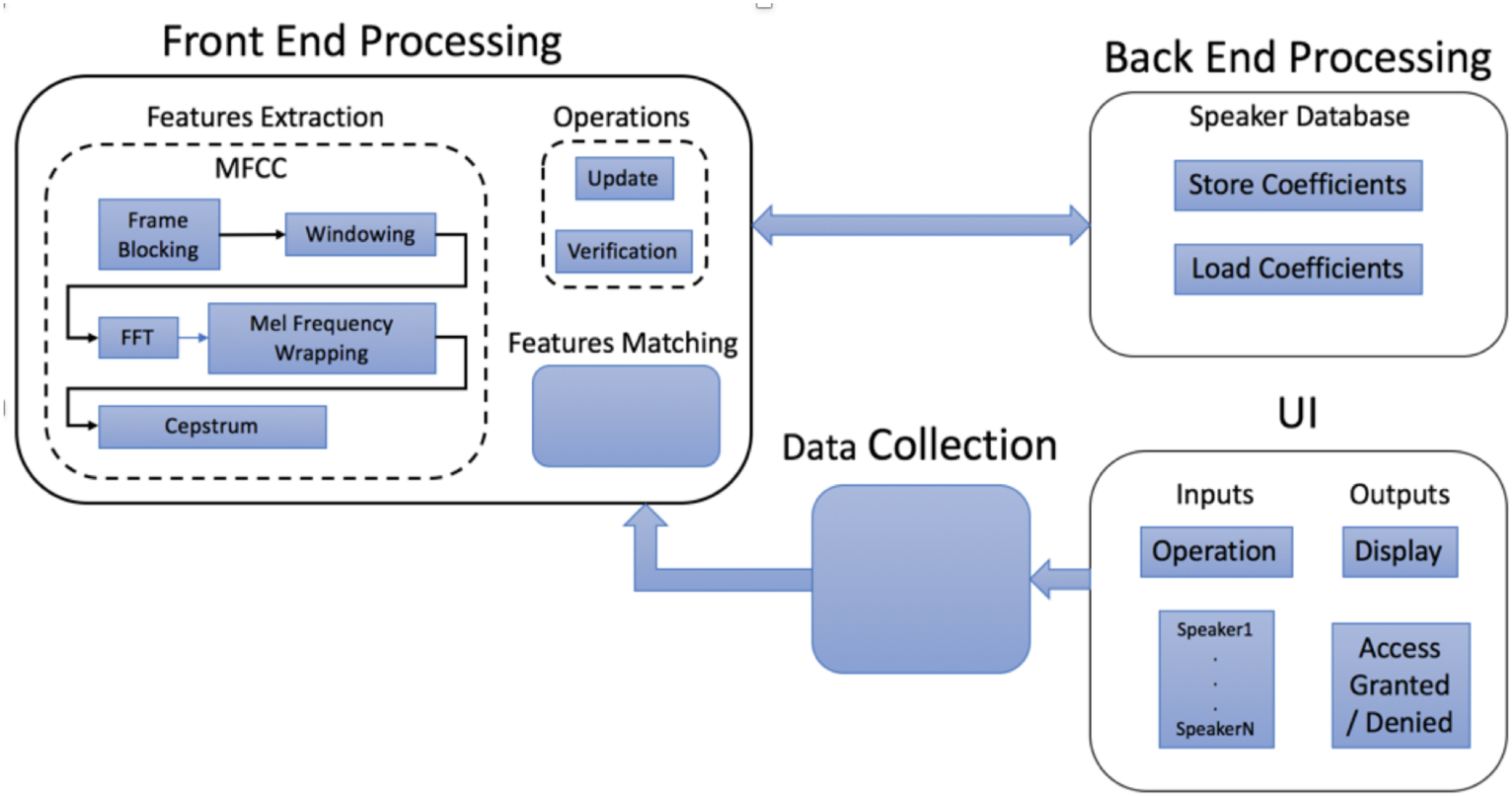
System Description

A “phrase dependent” speaker verification system

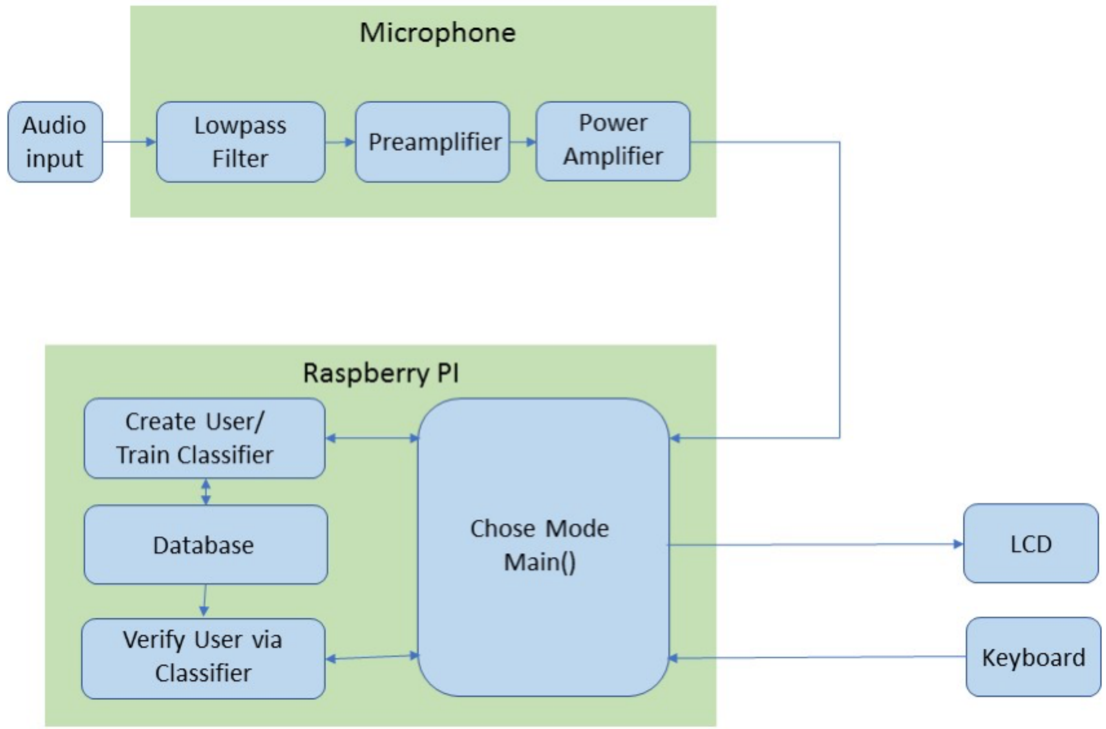
Two modes of operation:

- **Training mode**: Create a new user profile and train system to recognize them
  - New user provides many (e.g. n = 50) utterances of the same phrase by speaking into microphone
    - Relevant features are extracted from each utterance
    - Features are saved off and used to train machine learning model
- **Matching mode**: Attempt to verify identity of a speaker
  - User claims be a user already stored in the system
  - Purported user provides an experimental utterance by speaking into microphone

Previous block diagram



Current Block Diagram



System Requirements

System shall record voice in single-channel format at sampling rate of 44.1KHz

System shall maintain a database of voice samples from different users

System shall have an interface to provide verification feedback

Reliability:

System shall verify the identity of a speaker with >90% confidence

Recording:

System shall function in an environment with up to a mild amount of ambient noise

Proposed CDR Deliverables

1. Implementation of feature extraction algorithm on Raspberry Pi
   - Changed as per Prof. Hollot’s recommendation
2. Implementation of training algorithm
3. Database implementation/integration
4. UI implementation/integration
5. Custom hardware design

Subsystem: Feature Extraction/Classifier Training

Old method: MFCC

- Computationally expensive, complicated to implement
- Much more data than necessary

New method:

- Eliminate silent regions by throwing out all samples with amplitude below some threshold
- Transform time-domain input signal into spectrogram
- Extract highest-power frequency component in each frame (currently 5 frames per sample)
- Create feature vector for each voice sample
- Use voice samples in supervised training of Support Vector Machine classifier

Subsystem: Feature Extraction/Classifier Training

To use the system a user must

- say scripted input with ideally little variation between recordings
- record large amount of samples to train classifier

Subsystem: Feature Extraction/Classifier Training

How does a Support Vector Machine (SVM) work?

- Input a large amount of data that is manually classified into two classes, positive and negative.
- SVM then tries to make a boundary separating the data classified as positive from the data classified as negative.
- Accuracy is partially determined by percentage of misclassified data.
- Allowing for no misclassified data may lead to overtraining
- Classifies new samples by determining what side of boundary sample falls on

Subsystem: Database

Each database entry should contain:

1. Speaker ID
    a. Uniquely identifies each speaker (i.e a username)
2. Feature vector(s)
    a. The "Kelly Coefficients" obtained from feature extraction subsystem

Speaker ID	KCs
'paul'	[...]

Subsystem: User Interface

The means through which the user interacts with the system:

1. Prompts user to select operation mode (‘new user’ or ‘existing user’)
    a. If ‘new user’: prompt user to enter their voice training samples
    b. If ‘existing user’: prompt user to enter their experimental voice sample
2. Provide user with visual feedback

Proposed Subsystem: Microphone

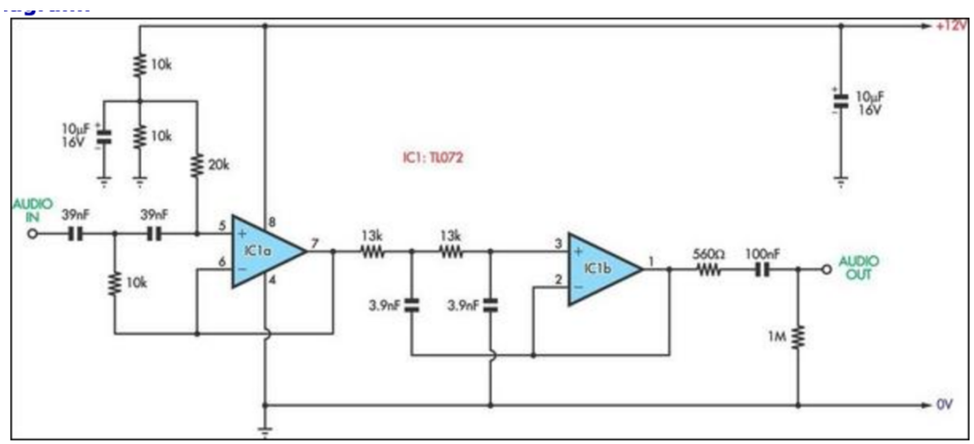
Voice Recording Subsystem:

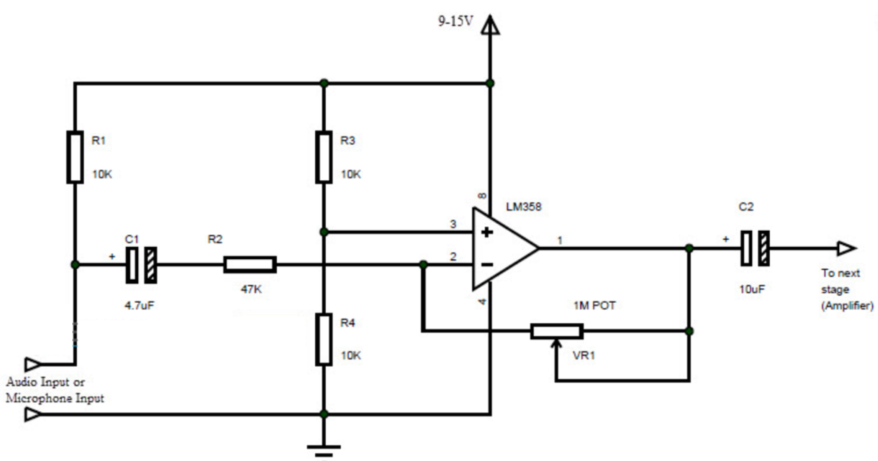
- Electret condenser microphone chip record the voice samples
- Frequencíe above the desired human voice's is eliminated with a pass filter
- Preamplifer circuit prepares the collected signal to be fed into a power amplifier
- Power amplifier amplifies the signal before it is fed into the signal processing component

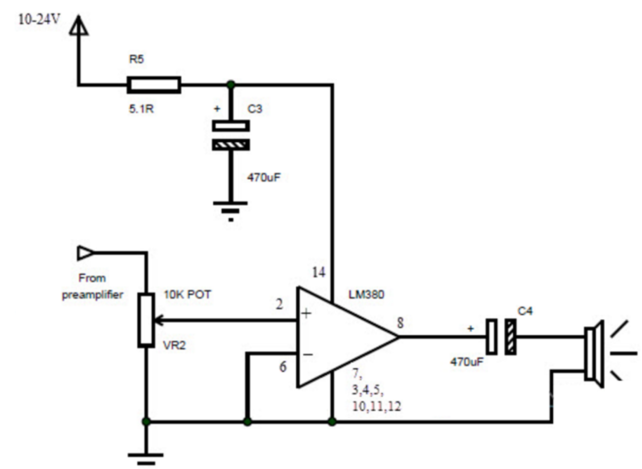
Subsystem’s Output:

- The result will be the speech signal with less noise

Microphone Schematics







Proposed FDR Deliverables

1. Implementation of feature extraction/classifier training on Raspberry Pi

2. Implementation of feature matching algorithm on Raspberry Pi

3. Build a proper database

4. Implementation of basic GUI

5. Custom built microphone on PCB

Role Assignment for FDR

Paul Mahoney: Implementation of Feature Extraction

Liam Shea: Implementation of SVM Training and Classifying

Phillip Ashe:Database implementation and management

Jason Nguyen: Noise cancelling microphone and filter design

Cost Estimation

RaspberryPi 3 - Model B (1 unit) : $35 element14

Raspberry Pi Touch screen display (1 unit) : $67 amazon

Microphone chip (TBD) : $0.93/unit digikey

Improving Usability/Reliability

Many parameters which can be tweaked to improve reliability:

- Spectrogram frame size/number of frames
- Number of training samples per user

Demo

1. Positively identifying Phil

2. Negatively identifying Paul, Liam, Jason

3. Training for a new user (any volunteers?)