

# Voice-enabled Internet of Things

Advanced Topics in Internet of Things

Presented by **Mohammad Mofrad**

University of Pittsburgh

April 26, 2018



# Motivations

- IoT devices are all around your home
  - *Smart speakers*: Amazon Echo & Google Home
  - *Heating & cooling*: Nest Learning Thermostat & Ecobee4
  - *Smart locks & doors*: August Smart Lock & SkyBell HD
  - *Smart lightening*: Philips Hue & Lix Color
  - *Cleaning*: iRobot Roomba, Neato Botvac
  - *Smart shades*: Lutron's Serena Shades
  - *Security cameras*: Amazon Cloud Cam & Nest Protect



- Ways to communicate with IoT devices:



- *Graphical User Interface (GUI)*:

- Pushing buttons
- Clicking on icons

- *Speech Interfaces*:



- Just talking to the device which is more intuitive and efficient
- Speech processing and natural language processing empowers these interfaces

# Motivations (Continued)

- **Limitations** of the current smart home IoT devices (e.g. a smart speaker)

1. Most of smart home devices are not customizable and customers cannot extend them to have their customized voice commands or tune their accuracy
2. Smart home speakers cannot handle complex scenarios such as:
  1. They fail processing combined commands separated by “and”.
  2. They fail processing two concurrent commands



- **Contributions** of this project is two folds:

1. Building a customizable voice-enabled IoT device
2. Proposing a solution for handling two concurrent voice commands to a voice-enabled IoT device

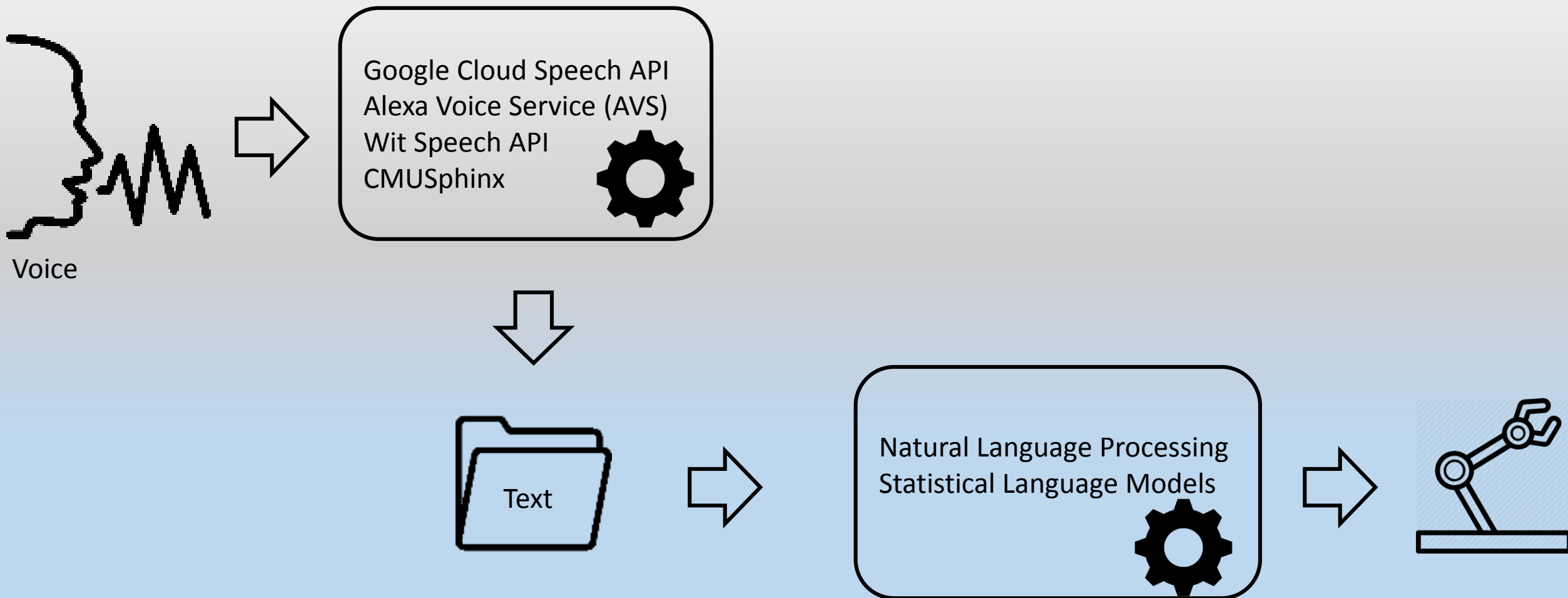


# A Customizable Voice-enabled IoT Device



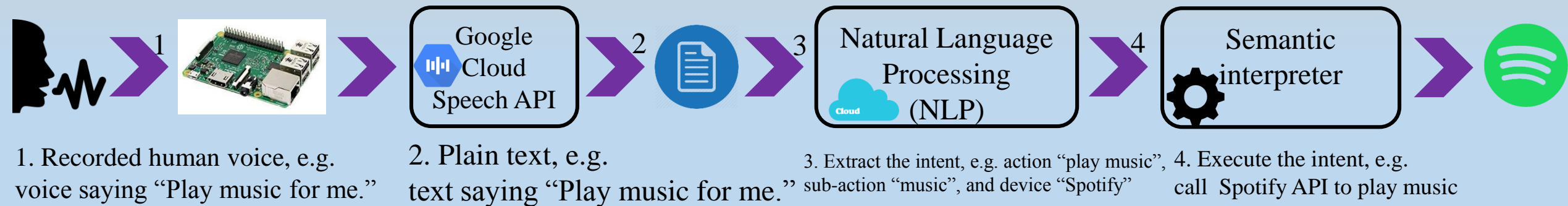
Blind Source Separation for Voice-enabled IoT

# How Speech recognition works for IoT



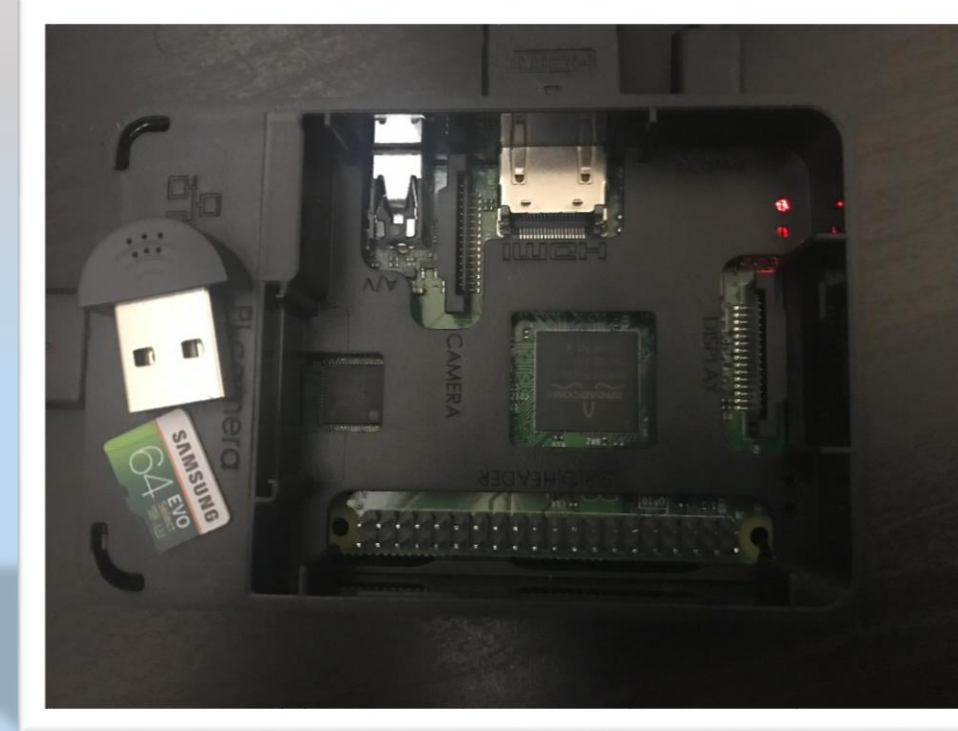
# Proposed Customizable Voice-enabled IoT Device

- The proposed model consists of the following components:
  1. We use a small USB microphone that captures the incoming voice
  2. We use Google Cloud Speech API which is free for developers
  3. We use a simple Natural Language Processing (NLP) model to create a language model for playing music
  4. We use the inferred intent on the target device to play/pause the music



# Introducing Voice-enabled IoT Device

- The total cost of building the prototype is **\$68.42** and consists of the following two building blocks:
- The hardware components of the customized IoT device:
  - Raspberry Pi 3 Model B Motherboard, \$35.80
    - Quad core Cortex A53 @ 1.2GHz
    - 1GB SDRAM
    - Wireless 802.11
    - Bluetooth 4.0
  - Kinobo USB 2.0 Mini Microphone, \$4.65
  - Samsung 64GB Micro SD Card, \$19.99
  - Raspberry Pi Case, \$7.98

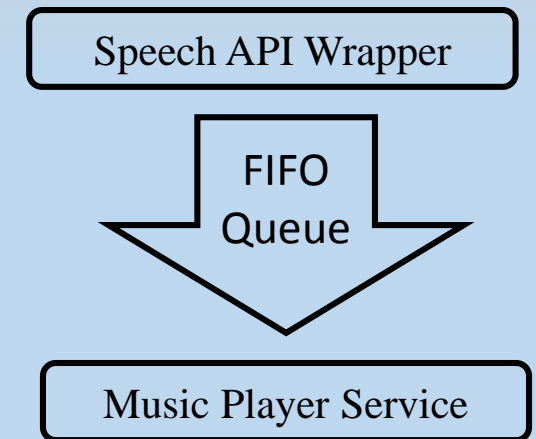


# Introducing Voice-enabled IoT Device

- The total cost of building the prototype is **\$68.42** and consists of the following two building blocks:
  - Other hardware which we did not pay for and you can easily find one around your place:
    - A 2.5A power adaptor (mobile adaptor)
    - A monitor and HDMI cable
    - A USB keyboard and mouse
    - Wired/Bluetooth speaker




- The software components of the prototype are:
  - Google Cloud Speech API which is free to use
  - Raspbian OS
  - Python 3.5





# Demo of Voice-enabled IoT Device



## Customizable Speech Recognition for Internet of Things

MohammadHasanzadeh Mofrad<sup>1</sup>, Omid Kashef<sup>2</sup>, and Daniel Mosse<sup>1</sup>


*Department of Computer Science, School of Computing and Information, University of Pittsburgh*  
*Intelligent Systems Program, School of Computing and Information, University of Pittsburgh*

### Motivations

- Internet of Things (IoT) devices are all around your place:
  - Smart Speakers: Amazon Echo & Google Home
  - Heating & Cooling: Nest Learning Thermostat & Ecobee
  - Smart Locks & Doors: August Smart Lock & SkyBell HD
  - Smart Lighting: Philips Hue & Lifi Color
  - Cleaning: iRobot Roomba, Neato Botvac
  - Smart Shades: Lutron's Serena Shades
  - Security Cameras: Amazon Cloud Cam & Nest Protect
- Ways to communicate with IoT devices:
  - Graphical User Interface (GUI)
    - Pushing buttons
    - Clicking on icons
  - Speech Interfaces
    - Just talking to the device which is more intuitive and efficient
    - Speech processing and natural language processing empowers these interfaces
- Current smart home devices (e.g. speakers) limitations:
  - They are not customizable and customers cannot extend them to have their customized voice commands
  - They are designed to serve the company's specific purposes such as making it easy to buy products in Amazon Echo which contributes to Amazon's profit
- The contribution of this work is to add this level of customization by:
  - Grabbing a Raspberry Pi as a cheap and programmable IoT infrastructure
  - Using free or open source speech recognition APIs like Alexa Voice Service (AVS) or Google Cloud Speech API for converting speech to text
  - Writing customized acoustic and language models on top of these APIs to gain more control over the IoT devices

### Customizable Speech Recognition Model


- The proposed model consists of the following components:
  - Recorder component (Acoustic Model)**
    - We use a small USB microphone that captures the incoming voice
  - Speech to text component (Acoustic Model)**
    - We use Alexa Voice Service (AVS) which is free for developers
  - Text to Intent component (Language Model)**
    - We use Natural Language Processing (NLP) to create a language model that returns the intent of the extracted sentence.
  - Intent to action component (Language Model)**
    - Execute the inferred intent on the target device



### Acoustic Model (Recorder and Speech to Text Components)

- The Recorder component has two parts:
  - A USB microphone mounted on the Raspberry Pi and records the voice
  - A program written in Python which send the recorded voice to AVS and store the transcribed text
- The Speech to text component:
  - We use the AVS as the choice of speech to text component because it is already trained with tons of voice data by Amazon
  - We use the AVS device SDK as a tool to transmit the recorded voice by the recorder component
  - The text to speech processing part is done in Amazon's Cloud and not the Raspberry Pi
  - The recorded voice is discarded after being transcribed to save space and memory

### Language Model (Text to Intent and Intent to Action Components)


- Noisy Channel Model
  - Find intended sentence, given a sentence where the words are scrambled
  - E.g. "I've ordered [two/too/to] of it."
  - Approximation
 
$$P(\text{sentence} | \text{acoustic}) = \frac{P(\text{acoustic} | \text{sentence}) P(\text{sentence})}{\text{Likelihood} \quad \text{Prior}}$$
- Likelihood
 

"I've ordered [two/too/to] of it."
- Prior
 

I've ordered [two/too/to] of it  $\frac{\text{Language Model}}{P(w_1) \dots P(w_n) \dots P(w_{n+1})}$  I've ordered **two** of it
- Text to Intent component: extract the intent from the constructed sentence
- Intent to action component: execute the inferred intent on the device

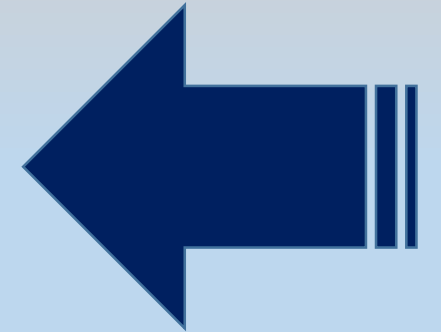
### Customized prototype

- The core hardware components of the proposed customized IoT prototype costs us \$65.4 and consists of the following building blocks:
  - Raspberry Pi 3 Model B Microcomputer - Quad core Cortex A53 @ 1.2GHz, 1GB DDR4 RAM, 16GB eMMC, and Bluetooth LE - \$35.00
  - Elektor USB2.0 Mini Microphone - \$10.00
  - Samsung 8 EVO M.2 SSD Card - \$19.99
  - Raspberry Pi Case - \$10.41
- Other hardware which we did not pay for and you can easily find one around your place:
  - A 2.5A power adapter (the one made at night)
  - A monitor and HDMI cable
  - A USB keyboard and mouse
  - Wireless Bluetooth speaker
- The software components of the prototype are:
  - Language model developed using example sentences
  - Alexa Voice Service (AVS) API to talk to free to use
  - Raspbian OS
  - Python 3.5



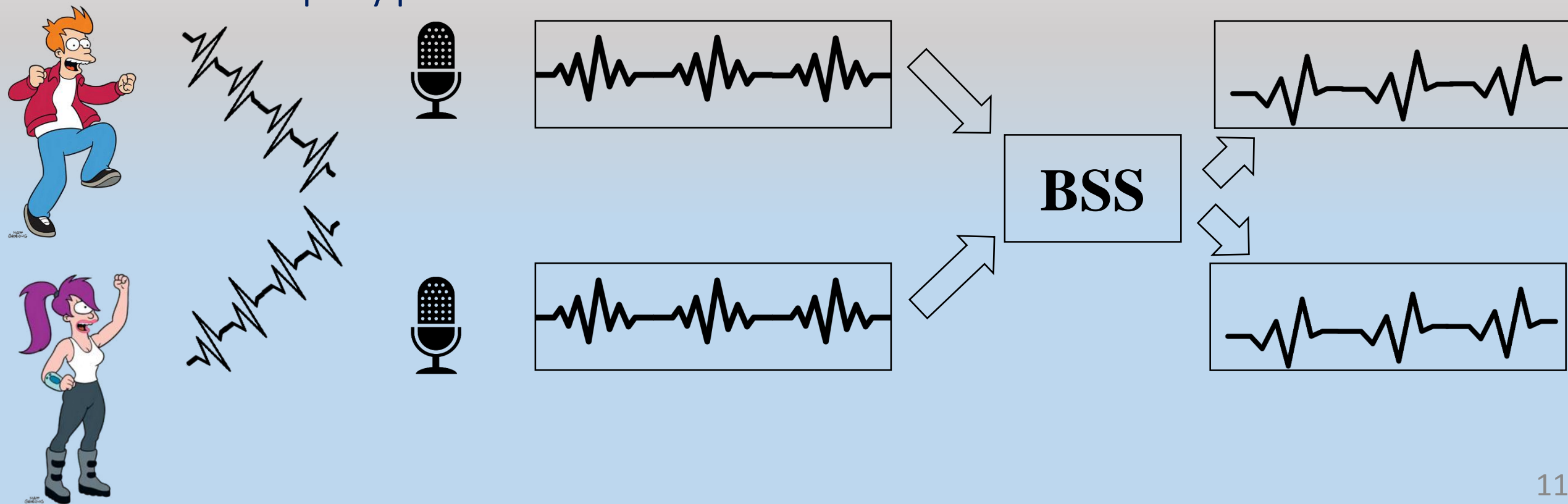
A Customizable Voice-enabled IoT Device

# Blind Source Separation for Voice-enabled IoT



# Blind Source Separation (BSS)

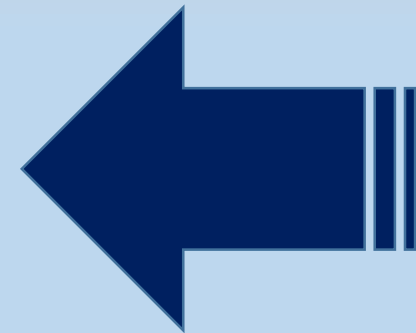
- Blind Source Separation is the separation of a set of source signals from a set of mixed signals without knowing about the mixing process of source signals.
  - Cocktail party effect
  - Cocktail party problem



A Customizable Voice-enabled IoT Device

# Blind Source Separation for Voice-enabled IoT

Reconstruction Independent  
Component Analysis (RICA)



# Reconstruction Independent Component Analysis (RICA)

- Independent Component Analysis (ICA) is a method for separating a multivariate signal into its component with the assumption of:
  - Subcomponents are non-Gaussian
  - Subcomponents are statistically independent
- ICA is a special case of blind source separation
- RICA algorithm is based on minimizing an objective function.
  - Source model:  $x = \mu + As$  where
    - $x_{p \times 1}$  mixed signals
    - $\mu_{p \times 1}$  offset values
    - $A_{p \times q}$  mixing matrix i.e.  $p = q$
    - $s_{q \times 1}$  original signal
  - Source signal can be recovered by
    - $s = A^{-1}(x - \mu)$
  - Without knowing  $A$  and  $\mu$ , given observations  $x_1, x_2, \dots$ , RICA extracts the original signal  $s_1, s_2, \dots$

# Reconstruction Independent Component Analysis (RICA) (continued)

- RICA uses

- A data matrix  $X$

$$X_{n \times p} = [x_1^T, x_2^T, \dots, x_n^T]^T \text{ i.e. } x_i \text{ } p \times 1$$

- Each row represents an observation and each column represents a measurement

- An initial random weight matrix  $W$

$$W_{p \times q} = [w_1, w_2, \dots, w_q] \text{ i.e. } w_i \text{ } p \times 1$$

- And optimizes

$$\sum g(XW) \text{ where } g = \frac{1}{2} \log(\cosh(2x))$$

- The objective function  $h$  results in a minimal matrix  $W$  that transforms data  $X$  to  $XW$

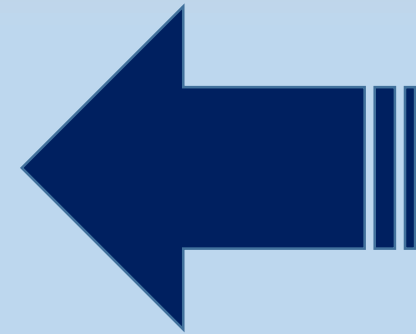
$$h = \lambda/n \sum_{i=1}^n ||WW^T x_i - x_i||_2^2 + 1/n \sum_{i=1}^n \sum_{j=1}^q \sigma_j g(w_j^T x_i)$$

A Customizable Voice-enabled IoT Device

# Blind Source Separation for Voice-enabled IoT

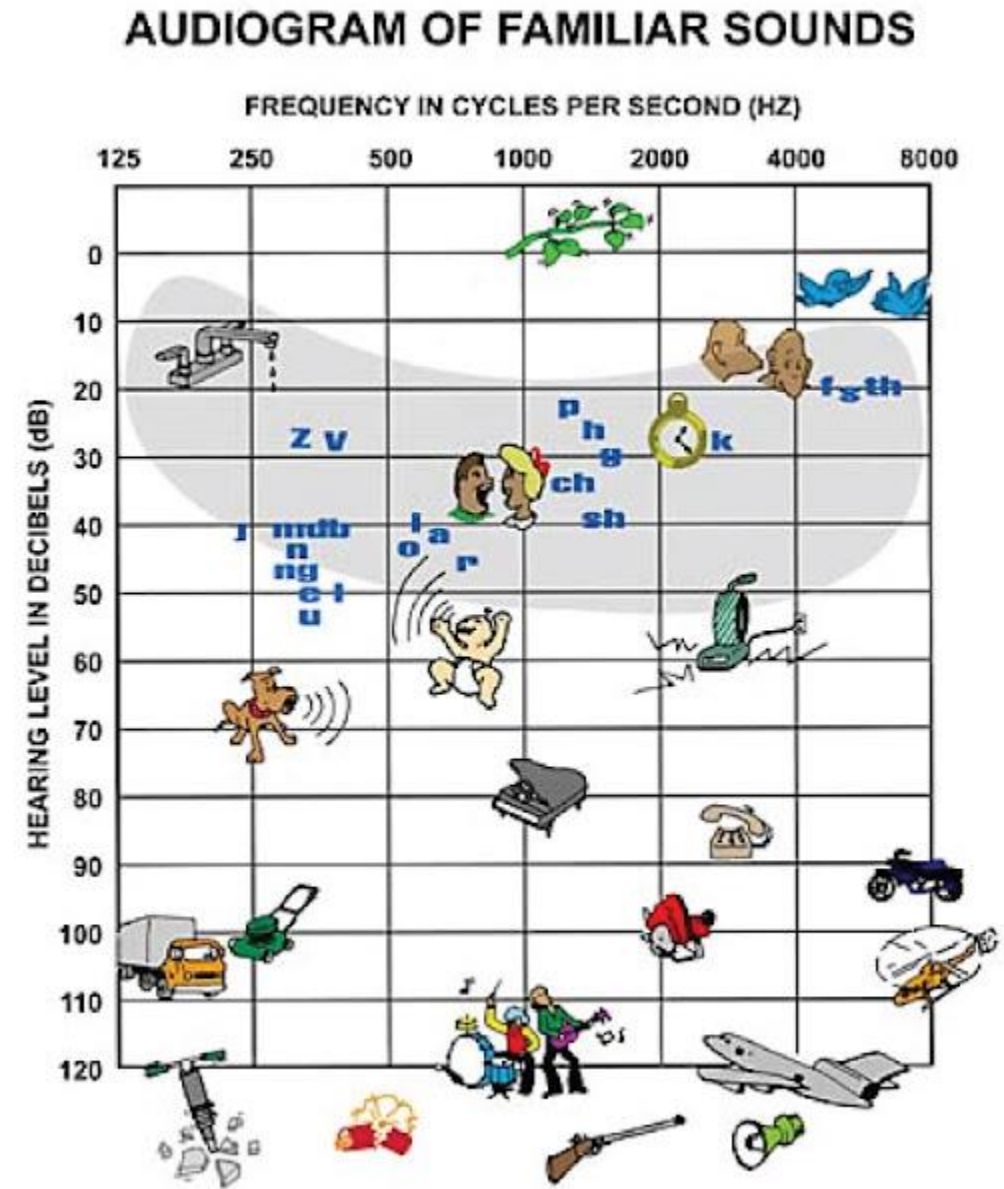
Reconstruction Independent Component Analysis (RICA)

Crossed Lowpass Filter



# Lowpass Filter

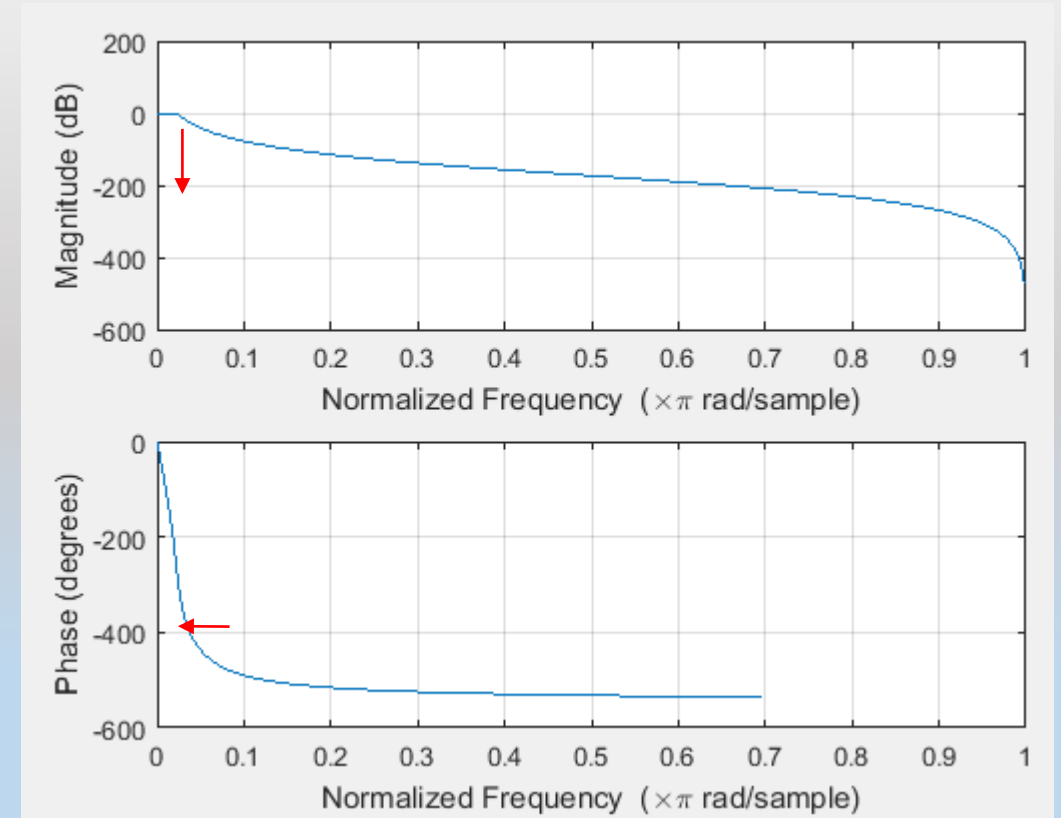
- Lowpass filter
  - Allows signals below a cutoff frequency (passband)
  - Attenuates signals above the cutoff frequency (stopband)
- By removing some frequencies, the lowpass filter smooths the signal.
- Hearing frequency range
  - The frequency of human voice is between 100 Hz and 8 KHz
  - The fundamental frequency is 100 Hz to 900Hz



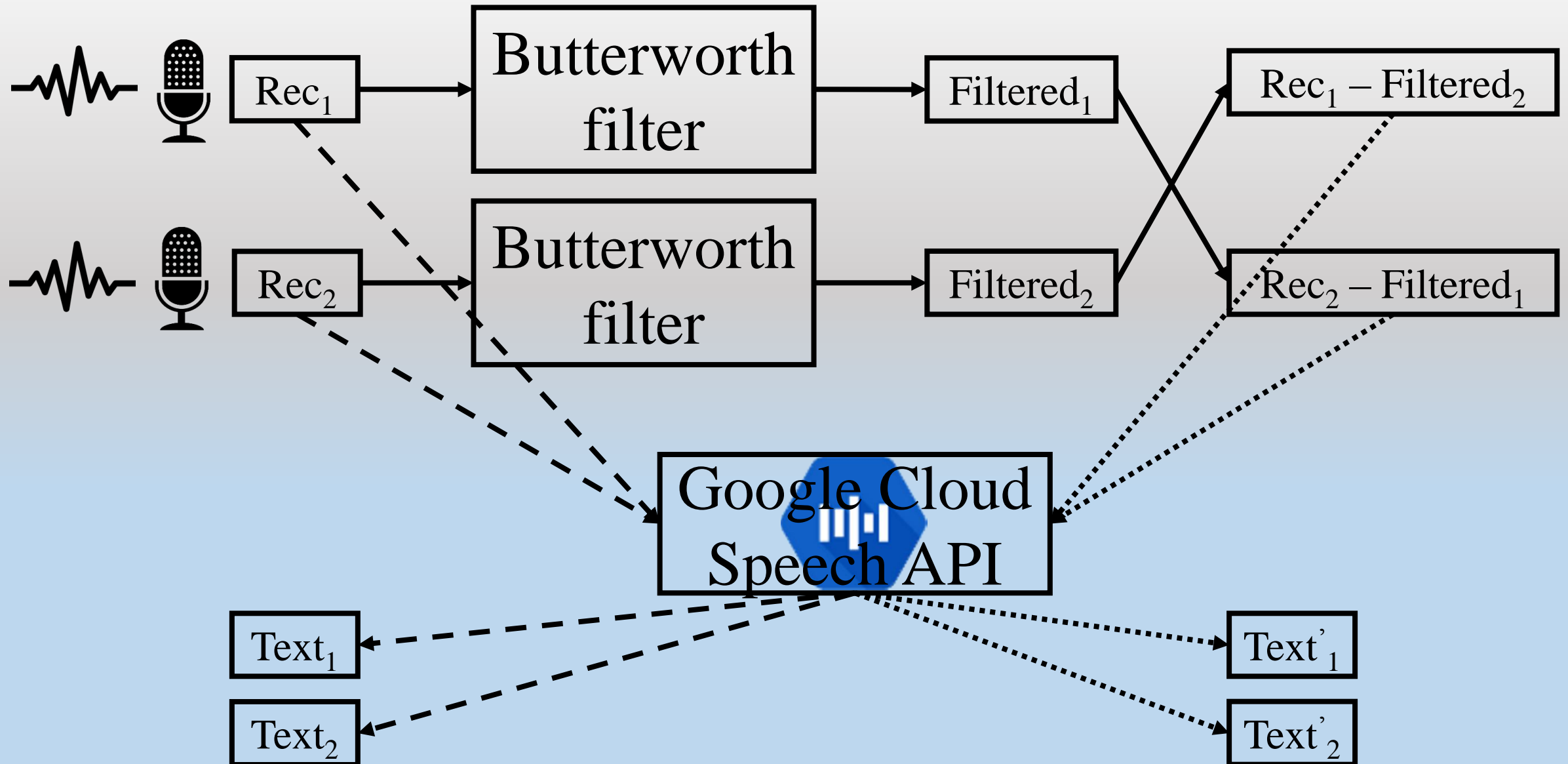


# Lowpass Filter Design

- Filter design: Butterworth filter
  - Butterworth filter completely reject the unwanted frequencies
  - Butterworth filter has uniform sensitivity for the wanted frequencies
- Filter details
  - The sample rate  $f_s$  is 44100 Hz
  - The cutoff frequency  $f_c$  is 500
  - $f_c / (f_s / 2) = 0.023 \pi$  rad/sample



# Crossed Lowpass Filter



# Transcription Accuracy

- Accuracy of transcription is calculated using

- Accuracy = 1 - Word Error Accuracy (WER) i.e. WER

$$\text{WER} = (S + D + I) / (S + D + C) \text{ i.e.}$$

S = #Substitutions

D = #Deletions

I = #Insertions

C = #Corrects

N = #Words = S + D + C

- Models

1. Baseline
2. RICA
3. Crossed Lowpass

- Recordings are annotated in a supervised manner and the ground-truth data is used to calculate WER for each model.

# Results

## Dataset 1:

30 different sentences

*Pairs of sentences are recorded in the proximity of the microphone*

Algorithm	Baseline	RICA	Crossed Lowpass
Mic <sub>1</sub>	0.96	0.91	0.99
Mic <sub>2</sub>	0.95	0.35	0.96
Total	0.95	0.63	0.97

## Dataset 2:

44 different sentences

*Pairs of sentences are recorded far from the proximity of the microphone*

Algorithm	Baseline	RICA	Crossed Lowpass
Mic <sub>1</sub>	0.96	0.95	0.98
Mic <sub>2</sub>	0.44	0.18	0.47
Total	0.70	0.56	0.73

# Conclusion

- The contributions of this project is two folds
  1. Building a voice-enabled IoT device comparable with smart speakers
    - Cost effective
    - Customizable
  2. Designing filters to improve the recognition power of existing speech-to-text APIs such as Google Cloud Speech API
    - *Adding a wrapper atop of Google Cloud Speech API which enables the API to process two concurrent commands which is not supported by the smart speakers right now.*
    - By enhancing the quality of the input speech signal , we were able to improve the performance of Google Cloud Speech API by 3%.

# Questions

