µSpeech

4.1.1

Generated by Doxygen 1.8.5

Fri Dec 27 2013 10:03:56

Contents

1 uSpeech library 1

2 Class Index 3

2.1 Class List . . . 3

3 Class Documentation 5

3.1 signal Class Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Constructor & Destructor Documentation . . . 6

3.1.2.1 signal . . . 6

3.1.3 Member Function Documentation . . . 6

3.1.3.1 calibrate . . . 6

3.1.3.2 complexity . . . 6

3.1.3.3 getPhoneme . . . 6

3.1.3.4 maxPower . . . 6

3.1.3.5 power . . . 6

3.1.3.6 sample . . . 6

3.1.4 Member Data Documentation . . . 6

3.1.4.1 aconstant . . . 6

3.1.4.2 amplificationFactor . . . 7

3.1.4.3 arr . . . 7

3.1.4.4 econstant . . . 7

3.1.4.5 f_enabled . . . 7

3.1.4.6 fconstant . . . 7

3.1.4.7 minVolume . . . 7

3.1.4.8 shconstant . . . 7

3.1.4.9 vconstant . . . 7

3.2 syllable Class Reference . . . 8

3.2.1 Detailed Description . . . 8

Index 9

Chapter 1

uSpeech library

The uSpeech library provides an interface for voice recognition using the Arduino. It currently produces phonemes, often the library will produce junk phonemes. Please bare with it for the time being. A noise removal function is underway.

Minimum Requirements

The library is quite intensive on the processor. Each sample collection takes about 3.2 milliseconds so pay close attention to the time. The library has been tested on the Arduino Uno (ATMega32). Each signal object uses up 160bytes. No real time scheduler should be used with this.

Features

- Letter based recognition
- Small memory footprint
- Arduino Compatible
- No training required
- Fixed point arithmetic (not anymore)
- 30% - 40% accuracy if based on phonemes, up to 80% if based on words.
- Plugs directly into an `analogRead()` port

Documentation

Head over to the `wiki` and you will find most of the documentation required.

Algorithm

The library utilizes a special algorithm to enable speech detection. First the complexity of the signal is determined by taking the absolute derivative of the signal multiplying it by a fixed point saclar and then dividing it by the absolute integral of the signal. Consonants (other than R,L,N and M) have a value above 40 and vowels have a value below 40. Consonants, they can be divided into frictaves and plosives. Plosives are like p or b whereas frictaves are like s or z. Generally each band of the complexity coeficient (abs derivative over abs integral) can be matched to a small set of frictaves and plosives. The signal determines if it is a plosive or a frictave by watching the length of the utterance (plosives occur over short periods while frictaves over long). Finally the most appropriate character is chosen.

- Return to main page

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

signal 5
syllable 8

Chapter 3

Class Documentation

3.1 signal Class Reference

Public Member Functions

- signal (int port)
- void sample ()
- unsigned int maxPower ()
- unsigned int power ()
- unsigned int complexity (int power)
- int **snr** (int power)
- void calibrate ()
- char getPhoneme ()
- int **goertzel** (int freq)

Public Attributes

- int arr [32]
- int **avgPower**
- int **testCoeff**
- int minVolume
- int fconstant
- int econstant
- int aconstant
- int vconstant
- int shconstant
- bool f_enabled
- int amplificationFactor
- int **micPower**

3.1.1 Detailed Description

Definition at line 23 of file uspeech.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 signal::signal (int *port*)

Constructor

Definition at line 5 of file signal.cpp.

3.1.3 Member Function Documentation

3.1.3.1 void signal::calibrate ( )

Calibration of background based on averaging

Definition at line 17 of file signal.cpp.

3.1.3.2 unsigned int signal::complexity (int *power*)

power/abs(sum of derivative)

Definition at line 51 of file signal.cpp.

3.1.3.3 char signal::getPhoneme ( )

The recognizer function: takes 1-4ms to execute

Definition at line 5 of file phoneme.cpp.

3.1.3.4 unsigned int signal::maxPower ( )

Point of maximum amplitude

Definition at line 66 of file signal.cpp.

3.1.3.5 unsigned int signal::power ( )

An estimate of background noise

Definition at line 38 of file signal.cpp.

3.1.3.6 void signal::sample ( )

Sampling of the sound: Based on storing values minus average background noise

Definition at line 25 of file signal.cpp.

3.1.4 Member Data Documentation

3.1.4.1 int signal::aconstant

This is the threshold for /a/ /o/ /r/ /l/, configure it yourself

Definition at line 31 of file uspeech.h.

3.1.4.2 int signal::amplificationFactor

Amplification factor: Adjust as you need

Definition at line 35 of file uspeech.h.

3.1.4.3 int signal::arr[32]

This is the audio buffer

Definition at line 25 of file uspeech.h.

3.1.4.4 int signal::econstant

This is the threshold for /ee/, /i/, configure it yourself

Definition at line 30 of file uspeech.h.

3.1.4.5 bool signal::f_enabled

Set this to false if you do not want to detect /f/s

Definition at line 34 of file uspeech.h.

3.1.4.6 int signal::fconstant

This is the threshold for /f/, configure it yourself

Definition at line 29 of file uspeech.h.

3.1.4.7 int signal::minVolume

This is the highest audio power that should be considered ready

Definition at line 28 of file uspeech.h.

3.1.4.8 int signal::shconstant

This is the threshold for /sh/ /ch/, above this everything else is regarded as /s/

Definition at line 33 of file uspeech.h.

3.1.4.9 int signal::vconstant

This is the threshold for /z/ /v/ /w/, configure it yourself

Definition at line 32 of file uspeech.h.

The documentation for this class was generated from the following files:

- uspeech.h
- phoneme.cpp
- signal.cpp

3.2 syllable Class Reference

Public Member Functions

- void **classify** (char c)
- int * **tointptr** ()
- void **debugPrint** ()

Public Attributes

- int **f**
- int **e**
- int **o**
- int **s**
- int **h**
- int **v**

3.2.1 Detailed Description

Definition at line 58 of file uspeech.h.

The documentation for this class was generated from the following files:

- uspeech.h
- vocab.cpp

Index

aconstant
    signal, 6
amplificationFactor
    signal, 6
arr
    signal, 7

calibrate
    signal, 6
complexity
    signal, 6

econstant
    signal, 7

f_enabled
    signal, 7
fconstant
    signal, 7

getPhoneme
    signal, 6

maxPower
    signal, 6
minVolume
    signal, 7

power
    signal, 6

sample
    signal, 6
shconstant
    signal, 7
signal, 5
    aconstant, 6
    amplificationFactor, 6
    arr, 7
    calibrate, 6
    complexity, 6
    econstant, 7
    f_enabled, 7
    fconstant, 7
    getPhoneme, 6
    maxPower, 6
    minVolume, 7
    power, 6
    sample, 6
    shconstant, 7
    signal, 6
    vconstant, 7
syllable, 8

vconstant
    signal, 7