*µ*Speech Developer's Guide

µSpeech provides a simple algorithm for the arduino to perform speech recognition. µSpeech is designed to work with a small vocabulary set. The minimal requirements are 0.5KB of ram and 5KB of programmable flash memory. µSpeech requires no training to recognize words.

Installation
Just like any other library µSpeech is meant to be installed in your arduino libraries folder under a directory named uSpeech. Now comes the tricky part. Wire up a condenser mic with a pre-amp and hook a microphone onto analog pin A0 and create a new sketch in the arduino IDE. Copy and paste the following in:

```
#include <uspeech.h>
signal voice(A0);
void setup(){
      voice.calibrate();
      Serial.begin(9600);
}
void loop(){

      Serial.println(voice.power());

}
```

Upload the code to your arduino/AVR board then open the serial monitor. You will see a stream of numbers, this is the loudness of your system. Now keep quiet. Then after 1 second stop autoscrolling. Select the most common value and copy it. In my case it was 92. Open your arduino library directory find the source code of µSpeech. Open `uspeech.h` in a text editor. At the top you will see the following:

```
#include "Arduino.h"
#include <math.h>
#define SILENCE 92
```

Change the value of SILENCE to your value. Now you are set to get your first voice recognition project going.

Hardware
µSpeech supports any form of analog speech input. Typically one would use a condenser microphone with a pre-amp made using a single transistor. PC microphones can also be used and so should a laser microphone (if you can build one...).

Crafting the vocabulary
"Vocabulary" is literally what it means. It is a set of commands that your device will respond to. So say we have a car powered by an arduino compatible processor and board, we want it to be able to left, right and stop. These commands need to be converted into a set of strings which make sense to the µcontroller's umatch() function. Each letter or sound needs to be converted according to the following table.

Letter	Phonetic
a	a
e	e
i	i
o	o
r	r,l,m,n
v	v, w
p	p, b
g	g
z	z, f, ch, j, tr, th
k	k,c,qu
s	s, sh

So left gets translated to rez and right to ri. Now the current recognizer (v0.0.2), is unable to handle diphthongs and will likely change. ‘R’ acts like a vowel making ri a diphthong. We can eliminate the ‘r’ from both the word leaving us with ‘ez’ and ‘i’. ‘Stop’ become ‘Sop’ as t is not part of the phonetic list in µSpeech. Ideally commands should be long and of varied length to achieve greater

Creating our recognizer

Now that you have crafted your vocabulary of two words all you need to do is collect the phoneme. Then check if the phoneme is silent, if so the recognizer will return ‘h’. Otherwise call the de-noise function. If it is silent we call the string matcher which will attempt to match the words to our predefined command set. Once matched the string will be reset to “” and the corresponding function is called. Now to code. Let’s include µSpeech and create some global variables and initiate our recognizer.

```
#include <uspeech.h>
signal voice(A0);
String collvoice;
```

We now need to initiate our setup function. In this function we calibrate the system to the microphone. For debugging reasons we also initiate a serial connection to help with debugging.

```
void setup(){
     voice.calibrate();
     Serial.begin(9600);
}
```

Next its time we write our left, right and stop functions. For now I'm assuming you've not built a real car so we will create a virtual one which sends commands to the serial monitor.

```
void left(){
      voice.calibrate();
      Serial.println("left");
}
void right(){
      voice.calibrate();
      Serial.println("right");
}
void stop(){
      voice.calibrate();
      Serial.println("stop");
}
```

Now that you have created these command functions its time to actually perform the speech recognition. We begin by calling the phoneme recognizer and storing the phoneme.

```
void loop(){
      char phoneme = voice.getPhoneme();
```

Next we check for silence, if there is no silence, de-noise the signal otherwise process.

```
      if( phoneme != 'h'){
            collvoice = denoise(phoneme,collvoice);
      }
      else {
            int i[3],j,min,x;
            i[0] = umatch(collvoice,"sop"); //stop
            i[1] = umatch(collvoice,"ez"); //left
            i[2] = umatch(collvoice,"i"); //right
            //find the lowest number
            while(j<0){
                  if(i[j]<min){
                        x = j;
                        min = i[j];
                  }
                  j++;
            }
            if(x == 0){
                  stop();
            }
            if(x == 1){
                  left();
            }
            if(x == 2){
                  right();
            }
      }
}
```

We are done, compile away!