IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)
e-ISSN: 2278-1684,p-ISSN: 2320-334X, Volume 13, Issue 5 Ver. II (Sep. - Oct. 2016), PP 01-05
www.iosrjournals.org

Mobile Source In Automobiles

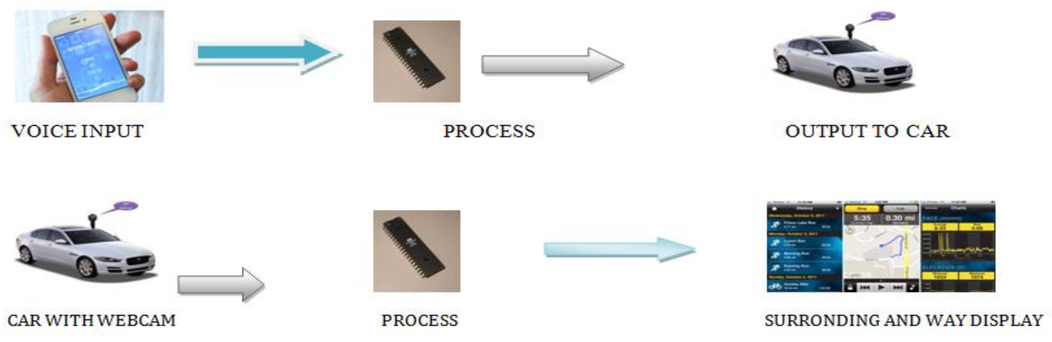
R. Arivazhagan,
Assistant Professor, Department of Mechanical Engineering, Kongunadu College of Engineering and Technology,Thottiyam,Trichy,Tamilnadu,India.
G.Anban,
Department of Mechanical Engineering, Kongunadu College of Engineering and Technology, Thottiyam, Trichy,Tamilnadu, India.

Abstract: *In the emerging automobile world our concept goes to play a vital role by the way of Using* ***MOBILE SOURCE*** *under,cloud computing voice recognition in automotive models for both commercial and secure surveillance process. The system works on the basics of recognition of voice as input and process in vehicles as output. Input is given to vehicles by using the mobile phones or PC or any web connecting systems. It maintains privacy between voice commander and receiver. Receiver has 360 degree webcam thus helps us to know the surrounding of our receiver which is the vehicle. By know the way, we can access the vehicle and get it to our spot. For far distance we can lock our location using GPS (automatic driving route). If any problem held in the way of passing, like diversion to another way like that, (temporary changing ways) then from that spot it makes call to us, then we access the vehicle spot using voice commanding or gestures to another way and lock the GPS (automatic driving route).After that it will arrive to our spot automatically by using sensors as well as using vehicle intelligence. The most use of this project is to get our vehicle (car) from parking to our doorstep.*

I. Introduction

Think about a creating a car which would be controlled by your voice. By giving a command, the car would drive you to your destination. The voice recognition algorithm we used could be applied to daily life; for example it would be most helpful to disabled people to perform their daily work. We created a speech controlled car using various electrical and mechanical domains such as digital signal processing, analog circuit design, and interfacing the car with the Mega32.

Block Diagram:



Background math:

Speech Analysis:

In order to analyze speech, we needed to look at the frequency content of the detected word. To do this we used several 4th order Chebyshev band pass filters. To create 4th order filters, we cascaded two second order filters using the following "Direct Form II Transposed" implementation of a difference equations.

$$y_1(n) = b_{11}x_1 + b_{12}x_1(n-1) + b_{13}x_1(n-2) - a_{11}y_1(n-1) - a_{12}y_1(n-2)$$

$$y_2(n) = b_{21}x_2 + b_{22}x_2(n-1) + b_{23}x_2(n-2) - a_{21}y_2(n-1) - a_{22}y_2(n-2)$$

$$y_{Out}(n) = gy_2(n)$$

Where the coefficient a's and b's were obtained through Mat lab using the following commands. [B,A] = cheby2(2,40,[Freq1, Freq2]);(Where 2 defines a 4th order filter, 40 defines the stop band ripple in decibels, and Freq1 and Freq2 are the normalized cut off frequencies). [sos2, g2] = tf2sos (B2, A2,'up','inf');

Fingerprint Calculation:

Due to the limited memory space on the Mega32, we needed a way to encode the relevant information of the spoken word. The relevant information for each word was encoded in a "fingerprint". To compare fingerprints we used the Euclidean distance formula between sampled word fingerprint and the stored fingerprints to find correct word.

Euclidean distance formula is: $d = \sqrt{\sum_{1}^{n} (p_i - q_i)^2}$

P = ($p_1, p_2, \ldots, p_n$) and Q = ($q_1, q_2, \ldots, q_n$)

Where P is a dictionary fingerprint and Q is the sampled word fingerprint and p i and q i are the data points that make up the fingerprint. To see if two words are the same we compute the Euclidean distance between them and the words with the minimum distance are considered to be the same. The formula above requires squaring the difference between the two points, but since we are using fixed point arithmetic, we found that squaring the difference produced too large of a number causing our variables to overflow. Thus we implemented a "pseudo Euclidean distance calculation" by moving the sum out of the square root reducing the equation to

D = $\sum_{1}^{n} |p_i - q_i|$

PWM duty cycle calculation:

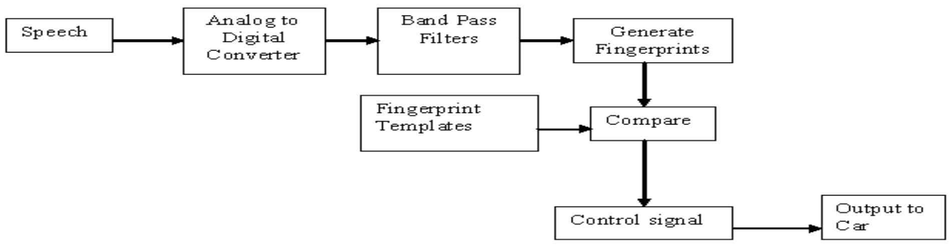
The motors in the car were measured to have a 50 Hz PWM frequency and movement was controlled by varying the duty cycle from 5% to 10%. To generate the PWM signals we used timer/counter1 in phase correct mode. The top value of timer/counter 1 was set to be 20000 and using a /8 prescalar the PWM signal was set to have a frequency of 50Hz =16MHz/(8*2*20000). To calculate the duty cycle the following equation was used OCR1x = (20000 - 40000*duty cycle). Where OCR1x is the value in the output compare register 1 A or B.

Hardware/Software tradeoffs:

The signal coming out of the microphone needed to be amplified. We had two different versions of operational amplifier, LMC 711 and LM 358. The LMC711 has a slew rate of 0.015 V/µs, on the other hand LM 358 has 0.3V/µs. The LM358 has a better slew rate and it gave us better response to input signal, so we used it when we designed our amplification circuit. The signal processing of speech requires lot of computations, which implies we need fast processor, but we had to operate at 16 MHz. In order to minimize the number of cycles we used filtering the audio signal we had to write most of the code in assembly. We wrote all of 10 digital filters in assembly which made them very efficient and significantly improved our performance over a C code implementation.

II. Software/Hardware Design

1) Software Description:



Overview

The Basic algorithm of our code was to check the ADC input at a rate of 4 KHz. If the value of the ADC is greater than the threshold value it is interpreted as the beginning of a half a second long word. The sample word passes through 8 band pass filters and is converted into a fingerprint. The words to be matched are stored as fingerprints in a dictionary so that sampled word fingerprints can be compared against them later. Once a fingerprint is generated from a sample word it is compared against the dictionary fingerprints and using the modified Euclidean distance calculation finds the fingerprint in the dictionary that is the closest match. Based on the word that matched the best the program sends a PWM signal to the car to perform basic operations like left, right, go, stop, or reverse.

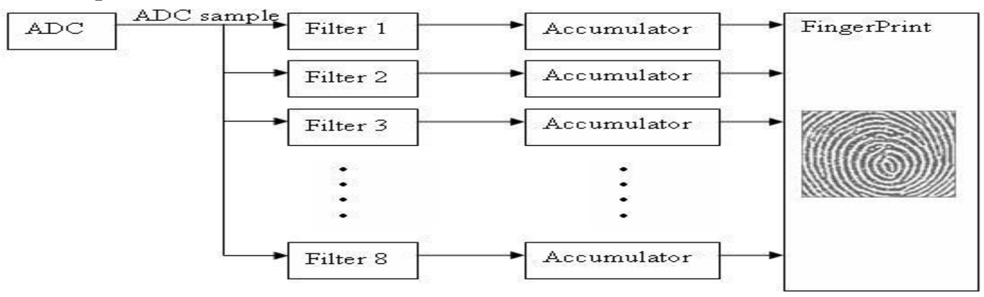
Initial-Threshold Calculation:

At start up as part of the initialization the program reads the ADC input using timercounter0 and accumulates its value 256 times. By interpreting the read in ADC value as a number between 1 to 1/256, in fixed point, and accumulating 256 times. The average value of ADC was calculated without doing a multiply or divide. Three average values are taken each with a 16.4msec delay between the samples. After receiving three average values, the threshold value is to be four times the value of the median number. The threshold value is useful to detect when a word has been spoken or not.

Fingerprint Generation:

The program considers a word detected if a sample value from the ADC is greater than the threshold value. Every sample of ADC is typecast to an integer and stored in a dummy variable Ain. The Ain value passes through 8 4th order Chebyshev band pass filters with a 40 dB stop band for 2000 samples (half a second) once a word has been detected. When a filter is used its output is squared and that value is accumulated with the previous squares of the filter output. After 125 samples the accumulated value is stored as a data point in the fingerprint of that word. The accumulator is then cleared and the process is begun again. After 2000 samples 16 points have been generated from each filter, thus every sampled word is divided up into 16 parts.
Our code is based around using 10 filters and since each one output 16 data points every fingerprint is made up of 160 data points.

Filter Implementation*:*



We chose a 4th order Chebyshev filter with 40 dB stop band since it had very sharp transitions after the cut-off frequency. We designed 10 filters a low pass with a cut-off of 200 Hz, a high pass with a cut-off of 1.8 KHz, and eight band passes that each had a 200 Hz bandwidth and were evenly distributed from 200Hz to 1.8 KHz. Thus we had band pass filters that went from 200-400 Hz, 400-600, 600 – 800 and so on all the way to the filter that covered 1.6 Khz – 1.8 Khz. We designed our filters in this way because we felt that most of the important frequency content in words was within the first 2 KHz since this usually contains the first and second speech formants, (resonant frequencies). This also allowed us to sample at 4 KHz and gave us almost enough time to implement 10 filters. We thought we needed ten filters each with approximately a 200 Hz bandwidth so that we would have enough frequency resolution to properly identify words. Originally we had 5 filters that spanned from 0 – 4 KHz and were sampling at 8 KHz, but this scheme did not produce consistent word recognition.

***Fingerprint Comparison*:**

Once the fingerprints are created and stored in the dictionary when a word was spoken, it was compared against the dictionary fingerprints. In order to do the comparison, we called a lookup() function. The lookup() function did a pseudo Euclidean distance formula by calculating the sum of the absolute value of the difference between each sample finger print a finger print from the dictionary. The dictionary has multiple words in it and the lookup went through all of them and picked the word with the smallest calculated number.
We had originally used the square of the correct Euclidean distance calculation, $d = \Sigma(pi - qi) 2$. The words we finally used in our dictionary were Let's Go, (sound of a finger snapping), daiya [right – in Hindi], rukh [stop – in Hindi], peiche [back – in Hindi]. We had originally used English words, go, left, right, stop, and back, but many of these words seemed to be very similar in frequency as far as our algorithm was concerned. We then went to vowels and had better success, but we still wanted to use words that were directions and so we went to Hindi The set of words that we used were mostly orthogonal, but in Hindi left is baiya, which sound very similar to daiya and so that could not be used. We had previously had success with snapping so we used that for left.

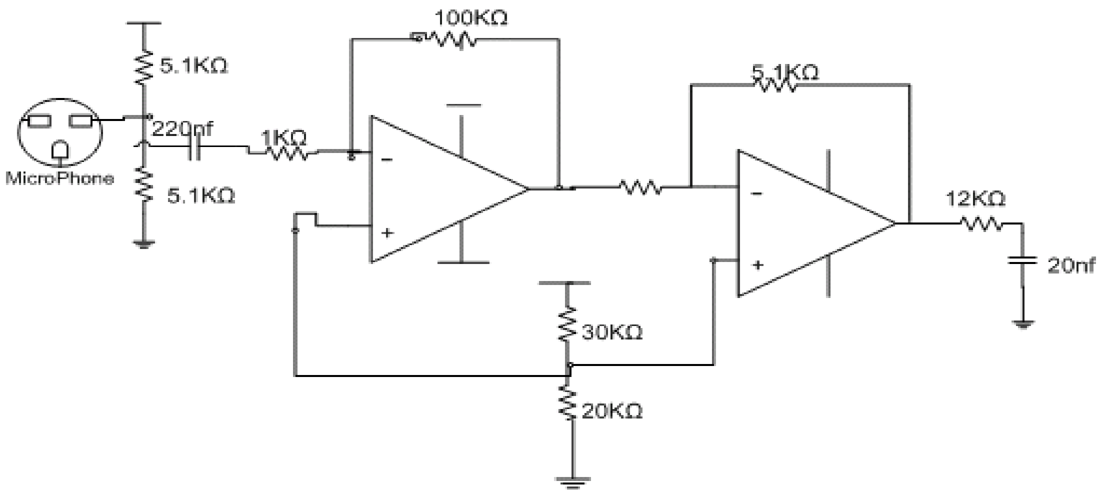
PWM signal to move car*:*

Once a word is recognized, its time to perform an action based on the recognized word. To perform an action we generated a PWM signal using timercounter1. Control of the PWM signal generation is done by the car control () function. For our car, we needed to generate two different PWM signals, one for moving the car front/back and another one to steer left or right. We also need to send a default PWM signal to pause a car. We chose timercounter1 because it has two different compare registers, OCR1A and OCR1B and can output two unique PWM signals. We used Phase correct mode to generate PWM signals because it is is glitch free, which is better for the motor.

To find out a frequency and a duty cycles at which car turns forward/backward and left/right, we attached an oscilloscope probe to a car's receiver. We sent different signals to the receiver using the car's remote control an d measured the frequency and duty cycle for different motions. From the measurements, we found that car PWM frequency was 50Hz (period of 20ms) and had the following properties.

Mode of Operation	Duty Cycle	Motor Type
Forward	10%	Electric
Reverse	5%	Electric
Right	5%	Servo
Left	10%	Servo
Stand by	7.5%	Both

Hard ware details:



The microphone that we used has a built in amplifier but after checking through oscilloscope, the signal strength was pretty low. We built the above amplifier signal to improve the peak to peak audio signal. The op. Amps are in negative feedback loops and are inverting and thereby have a gain of $\frac{-R_f}{R_{in}}$ So the total gain of the circuit is $\frac{-100K}{1K} * \frac{-5.1K}{1K} = 500$, with a dc bias of 2V given by the resistor divider circuit at the positive terminal of the op amp.

For live telecast

we use skype app for this process using webcam and transferring signal. In practical, we made a programme for our process and we can run vehicles privacy.

III. Project Results

Since we had to pass the ADC output through all of the filters faster than our sample time; the time it took do all the filter calculations was very important. We were able to run through 9 filters in under 4000 cycles, which is the amount of cycles available when sampling from the ADC at 4 KHz. The fingerprint comparison function did not have a speed requirement and so the cycle time for that was unimportant. The program was able to recognize five words, but sometimes it would become confused and match the incorrect word if the word that was spoken varied too much from the word stored in the dictionary. As a rough estimate the program recognized the correct word about 70% of the time a valid word was spoken. The program achieved success using voice, and with sufficient practice a person could say the same word with a small enough variation for the program to recognize the spoken word most of the time. For the general person though the recognition program would have a much lower percentage of success. Also the words in the dictionary are words spoken by only one person. If someone else said the same words it is unlikely the program would recognize the correct word most of the time, if at all. For safety an testing we made sure the PWM signals sent to the car were as close to neutral as possible, while still letting the move go forward and backward.

IV. Conclusion

At the beginning of our project, we set a goal to recognize five words, at the end of project we got five words to be recognized. However our five words needed to be orthogonal to each other because our filters were not giving a high enough resolution and inaccuracy in fingerprint calculations due to using fix point arithmetic made the lookup function to be error prone. As a result, we had to pick various different words that sound apart. If we had to do this again instead of trying to use the Euclidean distance formula to match words we would like to try do perform a correlation of the two fingerprints. A correlation is less sensitive to amplitude differences and is a better way of identifying patterns between two objects. If we had faster process chip, we could modified our algorithm to add more filters, perform Fourier transform, or floating point arithmetic in order to improve our results.