International Journal of Engineering & Technology, 7 (4.41) (2018) 105-109

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Voice Controlled Wheelchair

Shwetha V [1]*, Vaibhav Mani [2], Aditya Kumaran [3]

[1,2,3]Dept. of Electrical & Electronics Engineering, Manipal Institute of Technology
Manipal Academy of Higher Education, Manipal, Karnataka, India-576104
**Corresponding author E-mail:Shwetha.V@manipal.edu*

Abstract

The objective of this work is to facilitate the movement of people with physical disabilities rendering them incapable of independent movement and grants a degree of simplicity to those who lack the dexterity to control a joystick and had been achieved with the help of voice recognition model for the maneuvering of the wheelchair hence making it hands free and simplistic and can be added to existing wheel chair .The hardware is comprised of a wooden box designed in house open at one end, the wood used is plywood and the frame has a load withstanding capacity of 50 kg. The software implementation is the implementation of voice recognition on a Ubuntu distribution with the help of the CMU Sphinx Toolkit and the Pocket sphinx library. The coding for the voice recognition is done on a python platform using both Python 2.7 and 3.5 IDE as it is open source. The Arduino coding has been done in Ubuntu using the Arduino Software. In this work, two methods are used for voice recognition namely Key phrase Spotting and Sliding Match Filter and tested in different test environment. Using Keyphrase method gives a mean accuracy of up to 75% and maximum of 90% in noisy environment and tested in different environment along with fine tuning ,Successfully designed a voice controlled wheelchair with very quick response time along with voice recognition. Keyphrase spotting algorithm it is by far the more efficient system as compared to more robust Sliding match algorithm.

Keywords *Voice Recognition; API; Speech Recognition; CMU Sphinix; Audio length Keyphrase Spotting; Sliding Match filter; Unidirectional microphone..*

1. Introduction

Since the advent of the 21st century man has sought to develop technology which makes life easier, developments in this sphere has lead to great advancements in science and technology making life easier for everyone. Arguably the ones that should benefit the most is the physically challenged however, technology has become stagnant in this field after the invention the wheelchair. The joystick controlled version is the latest innovation but it fails to help those who lack the in motor skills to operate the joystick.

This work aims to assist these people and the social theme in this project grants it with great real world applicability, and served as the main source of inspiration for us. This project extends an existing voice controlled wheelchair project which was developed and was aimed at optimizing the performance and lowering the cost to make it a better solution to those who need it.

In this paper explains implementation of a voice recognition algorithm and lists out the basic commands that will move the wheelchair in four directions they have also included image processing for obstacle and edge detection and uses ultrasonic sensors to avoid these obstacles. Image processing has also been included to identify roadside signals to impart a sense of real world applicability to the design.

2. Research Method

2.1. Wheel Chair Design

The input of the top-level diagram includes five different voice commands telling the wheelchair where to go. These commands are given to the wheelchair via the connected microphone. This voice input is then searched for any one of the 5 different voice commands as specified using key phrase spotting by the python suite running on the laptop and if a keyword is recognized the necessary command is transmitted to the arduino via UART and based on the received input from the laptop the arduino will then give the required. Control signals to the motor driver to drive the wheel chair in the direction as specified in the voice command with minimalistic latency and low recognition error**.**

2.1.1 Hardware Design

The initial design incorporated a load bearing capacity 80kgs and was aimed to be a production quality wheel chair but the absence of aluminium rods required for the design forced us to look for alternative options. The chair on wooden box model was decided as a prototype as it clearly represented the functional attributes desired of this project and adequately represented by this design. The aluminium bracket used to mount the wheels comes from the previous rover of the team mars rover manipal which they graciously provided. The hardware is comprised of a wooden box designed in house open at one end, the wood used is plywood and the frame has a load withstanding capacity of 40kg shown in the Figure1. The motors are mounted using 2 aluminium brackets as shown in the Figure4 through the plywood box. The motors are 24 V 125 kg/cm and 250rpm high torque PMDC motors which are mounted through the bracket and through the wooden box using nuts and bolts. The wheels are low pressure balloon tires provided by wheels, these tires provide high traction even in sandy of wet terrain. They are mounted on the motors using triangular bearing blocks to ensure maximum contact between the wheels and the motor so as to avoid any play in the connection. A regular plastic chair is placed on the top of the frame to represent the functional objectives of the project

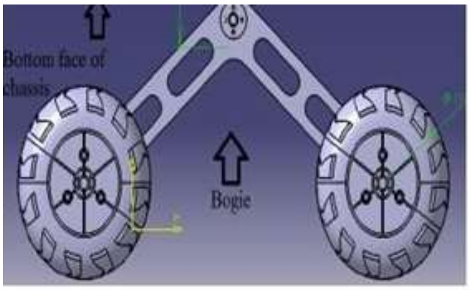


Fig1: Motor mounting bracket

Fig2: Plywood Frame

2.1.2 Electronic Design

The initial electronic raspberry pi, but when we tested it out we encountered with the compatibility of Sphinx with the raspberry pi and also ran into problems with the internet sharing with the raspberry pi and hence we decided to use a laptop computer in its place on the wheel chair. The interfacing of the motors with the motor driver saw its fair share of problems, the motor driver we were using earlier were made by Sabretooth with a peak current of 16A for 24V but the functionality of this motor driver caused a lot of problems. The motor driver requires a constant voltage of 2.5V to maintain the motors at zero, but the in our pursuit to do the same with the help of an arduino made it very difficult to provide the exact requisite voltage and would require the designing of a filter to smooth the output and maintain it at a constant 2.5V which was complicating the circuit and hence we switched to cytron motor driver which operate smoothly and have impeccable speed control and stability. The electronic section of the wheel chair boasts the seamless communication between the laptop and the Arduino which is interfaced with the motor through two motor driver ICs. A 24V 10000 mAh Lithium polymer battery is used to supply power to the motors and the motor driver circuits shown in the Figure3. The motor driver used are provided by its rated up to 48V and for peak currents of up to 12A. Laptop runs on the Ubuntu 16.04 Open Source Distribution.

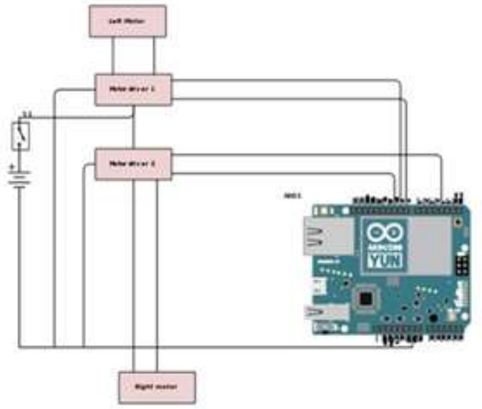


Fig3: Circuit Diagram

2.2 Software

Work involving voice recognition software that was tested and implemented to be used in the wheelchair for control. Even though the Google Speech Recognition software had more precision and robust and the CMUSphinx toolkit with its offline applicability and customization options was preferred.
The software implementation has been achieved with the implementation of voice recognition on a Ubuntu distribution with the help of the CMU Sphinx Toolkit and the Pocket sphinx library. The coding for the voice recognition is done on a python platform using both Python 2.7 and 3.5 IDE as it is open source. The arduino coding has been done in ubuntu using the Arduino Software as shown in the Figure4. Two methods have been used to achieve voice recognition namely Key phrase Spotting and Sliding Match Filter..

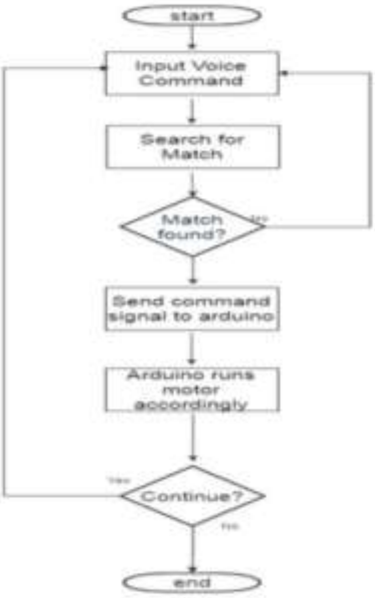


Fig4: Voice Detection Algorithm

2.2.1 Sliding Match Filter

Sliding Match Filter One of the most common solutions for voice recognition and under constant development is the Sliding Match Filter method which uses the CMU Sphinx Toolkit. In this algorithm, we first configured the decoder and then set the path for the language model to be used along with the link to the corresponding dictionary that it has to refer. The configured decoder is invoked and the microphone is set as the input source. Once this has been accomplished the thresholding is done where the background noise levels are recorded and used to specify a relative threshold which is used to detect the utterance of a keyword or phrase. A sliding match filter is designed and its parameters are set according to the length of the audio to be recorded and is configured to ”slide” throughout the audio stream looking for spikes in the recorded threshold. A stream of audio is recorded via the microphone when a spike over the threshold is detected by the match filter. This spike indicates the start of the command and the recording is stopped once the audio intensity falls below the threshold value. The silence width or the delay between the audio intensity falling below the threshold and the moment when the recording is stopped is specified at the start of the program. The recorded audio is decoded by the configured decoder by using the language model and dictionary given in the links. The values obtained by the decoder are compared with each and every word in the dictionary until a match is found and it is returned. The word returned is compared with the set keyphrases and then the wheelchair performs the given function if the corresponding keyword is detected as shown in Figure 5.

2.2.2 Key phrase Spotting

This method uses a keyword detection feature of the pocket sphinx library for voice recognition. This is achieved by running the pocket sphinx commands on the terminal window for a continuous stream of audio recorded via the microphone. It uses the same acoustic model as that in the sliding match filter method, but the search space is significantly reduced and hence the adjacency matrices to be compared is largely reduced. Therefore, the time taken for speech recognition is greatly improved. The terminal command used in this method has several arguments which needs to be set for the this method to work. The first deals with the audio length. The most often used argument is continuous which defines a continuous stream of audio which is infinitely long and this audio is searched for the keyphrase. The next argument specifies the source of audio and we have used the attribute inmic which sets the microphone as the audio source. The language model to be used is specified in the next argument and is the value to the -hmm field. A custom dictionary is created for this method which contains only the required keyphrases which is to be detected along with its pronunciations. The name of this file is saved with a .dic extension and serves as the value to the -dict field. This method requires the specifications of thresholds for detection of the keyphrases so as to compensate for the reduced dictionary size and account for misdetections. The thresholds are specified in a separate file which is saved with a .list extension and is passed as the value to the -kws field. The audio recorded is decoded by the default decoder and the adjacency matrices are compared with the words in the dictionary using a language model and if the values fall within the mentioned thresholds for each word, that word is printed as the detected keyphrase. Since, this method requires that the command be written in the terminal window in shell script it cannot be used as a python script directly. But we have researched a method which makes this possible using the OS library and command library in python running on Ubuntu. This method requires that an audio file of fixed length is recorded and then overwritten which is then used for the decoding process as the source of audio. Since this method directly uses the terminal window the delays in the voice recognition is almost completely eliminated flowchart as shown Figure 5.

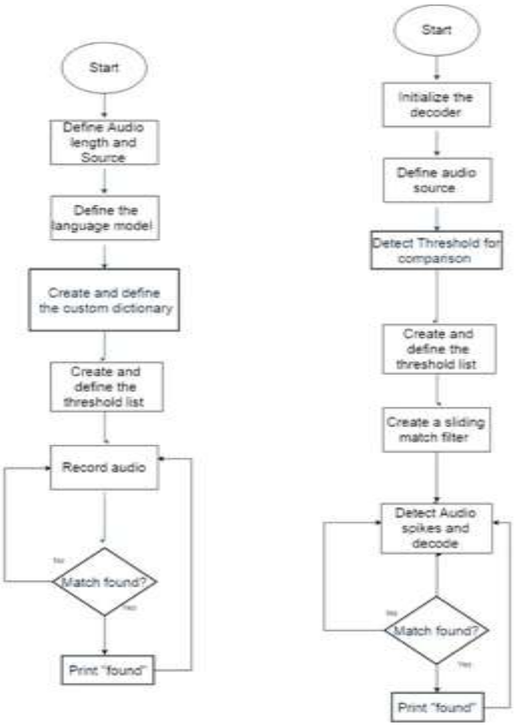
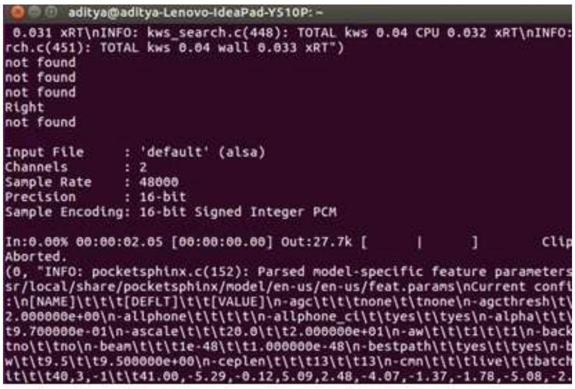

Start
Define Audio length and Source
Define the language model
Create and define the custom dictionary
Create and define the threshold list
Record audio
No
Match found?
Yes
Print "found"
Start
Initialize the decoder
Define audio source
Detect Threshold for comparison
Create and define the threshold list
Create a sliding match filter
Detect Audio spikes and decode
No
Match found?
Yes
Print "found"

Fig5: Key phrase Spotting & Sliding Match filter

3. Result

The results shown in the Figure 6 and Figure 7 demonstrate the usage and applicability of the voice recognition algorithm. . This section compares the algorithm used in this paper with the Siding match filter algorithm which is a popular algorithm for voice recognition. The detected set printed in the terminal window shows the detected output to the input to the comment Right rover moments are shown in the Table 1, the Sliding match filter output contains of multiple detections of the input, this is due to the huge latency and time delay involved in the Speech recognition process due to this we repeatedly supplied the input and hence resulted in multiple detections of the input. Whereas the key phrase spotting output shows only a single detection and this is due to the quick response of the system and the user has time to issue only a single command before the recognition output is shown and due to this there are even fewer chances of misinterpretation as the audio only contains a single phrase as opposed to the greater length in the audio file parsed by the sliding match filter.



```
 0.031 xRT\nINFO: kws_search.c(448): TOTAL kws 0.04 CPU 0.032 xRT\nINFO:
rch.c(451): TOTAL kws 0.04 wall 0.033 xRT")
not found
not found
not found
Right
not found

Input File     : 'default' (alsa)
Channels       : 2
Sample Rate    : 48000
Precision      : 16-bit
Sample Encoding: 16-bit Signed Integer PCM

In:0.00% 00:00:02.05 [00:00:00.00] Out:27.7k [      |      ]        Clip
Aborted.
(0, "INFO: pocketsphinx.c(152): Parsed model-specific feature parameters
sr/local/share/pocketsphinx/model/en-us/en-us/feat.params\nCurrent confi
:\n[NAME]\t\t\t[DEFLT]\t\t[VALUE]\n-agc\t\t\tnone\t\tnone\n-agcthresh\t\
2.000000e+00\n-allphone\t\t\t\t\n-allphone_ci\t\tyes\t\tyes\n-alpha\t\t\
t9.700000e-01\n-ascale\t\t\t20.0\t\t2.000000e+01\n-aw\t\t\t1\t\t1\n-back
tno\t\tno\n-beam\t\t\t1e-48\t\t1.000000e-48\n-bestpath\t\tyes\t\tyes\n-b
w\t\t9.5\t\t9.500000e+00\n-ceplen\t\t\t13\t\t13\n-cmn\t\t\tlive\t\tbatch
lt\t\t40,3,-1\t\t41.00,-5.29,-0.12,5.09,2.48,-4.07,-1.37,-1.78,-5.08,-2.
```

Fig 6: Keyphrase Spotting Output

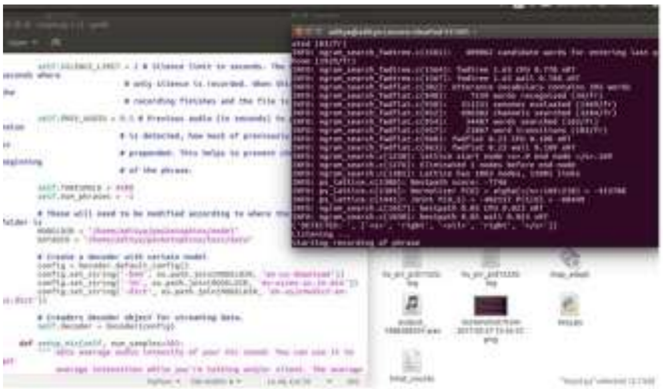
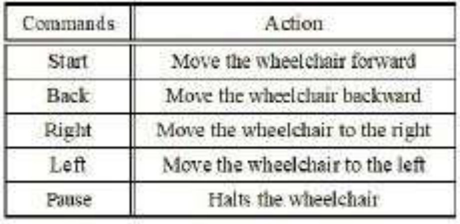


Fig 7: Sliding Match Filter output

The results obtained in the work clearly show the feasibility and applicability of this project in the real world and with minute improvements in the microphone and addition of few sensors for crash avoidance this product is ready to be hot fixed into any existing wheel chair design with relative simplicity. Since the Speech recognition is very quick real time latency is only limited to the length of the audio recorded for recognition which is fixed and easily increased or decreased in keyphrase spotting method to match one's needs. The only drawback of the keyphrase spotting model is its susceptibility to noise and this can be rectified by either adjusting the input sound volume in settings so as to reduce the volume of noise detected, but in this case one would have speak very loudly to make it work and this is not recommended. Another solution is the use of unidirectional microphones and place so that it records only the voice of the driver and no one else.Table1 explains for commands wheel chair movement.

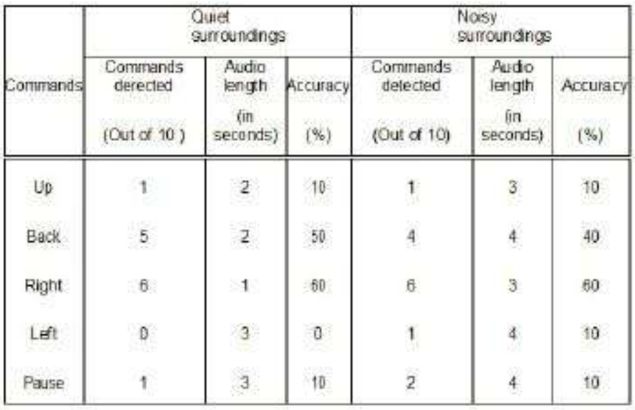
Table 1 : Commands for rover operation



Commands	Action
Start	Move the wheelchair forward
Back	Move the wheelchair backward
Right	Move the wheelchair to the right
Left	Move the wheelchair to the left
Pause	Halts the wheelchair

In Quiet surroundings as shown Table 2 the performance clearly favors the keyphrase spotting method and the difference in performance is huge. The keyphrase gives a mean accuracy of up to 75% which can be further improved by further fine tuning the thresholds specified for each keyphrase in the .list file before running the code. In this method the scope for misdetections are drastically reduced due to reduced search space for the decoder as the search space is limited to only the words specified in the. .dic file, all possible errors that may arise due to variation in pronunciations and accents can be easily corrected by simply setting the right thresholds for detection. Whereas, the Sliding Match Filter performs very poorly in this condition, and we encounter a lot of misdetections and errors mainly due to the lack of optimization of the language model, due to varying accents the variance matrices and adjacency lists constructed do not match the word given as input but instead in misinterpreted as another word hence, giving rise to a large number of errors in the results. It may be more accurate in the US where the language model was initially trained and constructed but in other countries or foreigners will be unable to use this method with success without training their own language model, or fine tuning the existing model both of which are long and arduous processes and require huge training sets. B. Noisy Surroundings In a Noisy environment the performance statistics shows a different story. Although pure performance wise the keyword spotting method is still favored, there is a clear decrease in performance when compared to the use of the same algorithm in the previous case. This is due to the fact that in the absence of a uni-directional microphone or a noise cancelling circuit this algorithm is prone to misdetections and errors in a noisy environment as the recording equipment used is the microphone from the pair of standard earphones the sound quality is also poor but when this is coupled with background noise it leads to erroneous results and misdetections. For demonstration purposes in this project this was handled by reducing the input sound volume of the microphone in the audio settings hence reducing all the background noise to very low volumes but when speaking directly into the microphone.

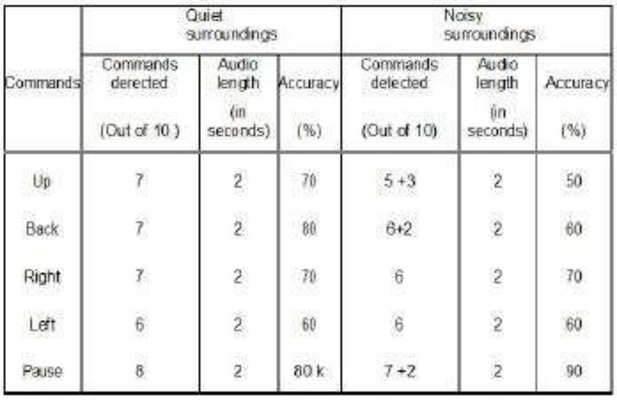
Table 2: Accuracy for sliding Match filter and Key Phrase spotting method in different environment condition



Commands	Quiet surroundings			Noisy surroundings		
	Commands derected (Out of 10)	Audio length (in seconds)	Accuracy (%)	Commands detected (Out of 10)	Audio length (in seconds)	Accuracy (%)
Up	1	2	10	1	3	10
Back	5	2	50	4	4	40
Right	6	1	60	6	3	60
Left	0	3	0	1	4	10
Pause	1	3	10	2	4	10

In a loud enough volume Table 3, the input command can be successfully isolated from the background noise hence slightly improving detection rates after accordingly adjusting the thresholds. In the table there is a different notation used in the commands detected column for this algorithm for example the Up command has the value 5+3, this notation has been used to show the number of times this word was detected even when it was not inputted. These conditions arise only in noisy environments where the background noise factors into the variance matrices created and due to the threshold widths the values sometimes fall into the thresholds for two different commands at the same time and this leads to the misinterpretation by the software followed by simultaneous detection of more than one key phrase.

Table 3: Accuracy for sliding Match filter and Key Phrase spotting method in different environment condition



Commands	Quiet surroundings			Noisy surroundings		
	Commands derected (Out of 10)	Audio length (in seconds)	Accuracy (%)	Commands detected (Out of 10)	Audio length (in seconds)	Accuracy (%)
Up	7	2	70	5 +3	2	50
Back	7	2	80	6+2	2	60
Right	7	2	70	6	2	70
Left	6	2	60	6	2	60
Pause	8	2	80 k	7 +2	2	90

However, for the sliding match filter even though the accuracy is low the variation from quiet from noisy environments is negligible as the algorithm first records the background sound and noise and sets that as a threshold for the capturing the utterance of the key phrase by the match filter. This means no matter how much noise is present the threshold is automatically set at the level of the noise, when a keyword is spoken into the microphone there is spike in the audio levels above the threshold as the input is given directly into the microphone and hence registers larger amplitudes than the noise

sliding match filter immediately detects this as an utterance of a keyword and hence the audio sent for decoding is only the spikes recorded , therefore the variation in results due to change in environment is not great and can be in most cases ignored. This shows that although this algorithm lacks the accuracy of Key phrase spotting algorithm it is much more robust.

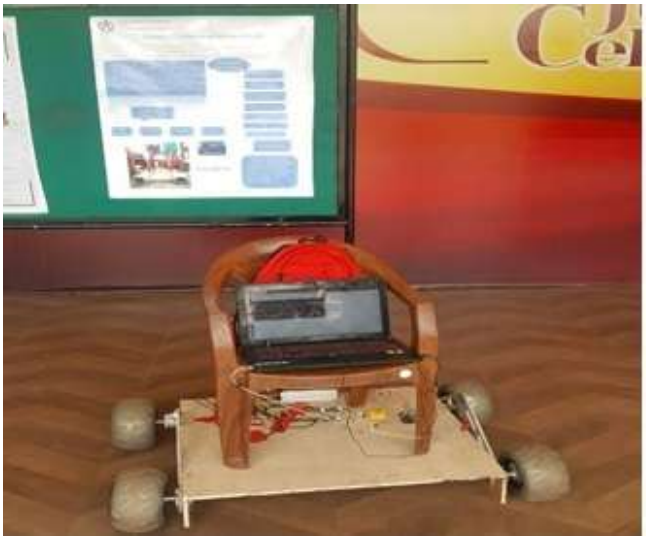


Fig 8: Voice Controlled Wheel chair

4. Conclusion

The goal of the project was to design a voice controlled wheelchair with very quick response time designed as shown in Figure 8. We have worked on the voice recognition and obtained satisfactory results. The difference between two main methodologies that can be used to implement voice recognition algorithms was mentioned.

This Work has a social cause and can serve to improve the transportation available to the paralyzed and the disabled. The Design is also made with a very rudimentary and cheap design that can be easily added to an existing wheelchair and can be commercially implemented. The limitations of this project are that the circuitry requires the addition of a noise cancellation filter or a unidirectional microphone and the threshold setting process needs to be made easy as it is now based on trial and error. For future improvements on this prototype model of the wheelchair, the model can be improved by adding ultrasonic sensors on the contour of the wheelchair so it can avoid obstacles, thereby making the wheelchair safe from collisions. Another improvement that could be implemented on the wheelchair is to include self-driving algorithms that help the wheelchair move independently on its own from one location to another with the help of cameras, sensors and GPS

References

[1] Yassine Rabhi ,Makrem Mrabet, Farhat Fnaiech, Philippe Gorce.” Intelligent joystick for controlling power wheelchair navigation” 3rd International Conference on Systems and Control on IEEE,2013

[2] Sobia, M. Carmel, V. Brindha, and A. Abudhahir. "Facial expression recognition using PCA based interface for wheelchair." Electronics and Communication Systems (ICECS), 2014 International Conference on. IEEE, 2014.

[3] Megalingam, Rajesh Kannan et al. "'Gest-BOT'-A Highly Convenient Locomotive Solution for the Elderly and Physically Challenged." Global Humanitarian Technology Conference (GHTC), 2012 IEEE. IEEE, 2012.

[4] Disability and rehabilitation-World Health Organization.

[5] http://www.who.int/disabilities/publications/technology/wheelchair guidelines/endate