Medical Assistive Robot

Ashraful Ashiq,[1,*] Jayanta Dey,[3] Ajoad Hasan,[4] Tanzim Mahrin,[5] Abdullah Ibn Mahmud,[6]and Niloy Acharjee[7]
EEE,BUET
Bangladesh
*asrafulashiq@gmail.com

***Abstract*—For the last decade, human assistive robots have been playing vital role in technologically advanced countries like in household chores, industries, hospitals etc. Often people fall into critical situations where they need emergency help. Especially this holds true most in case of medically challenged persons, like those who are in bedrest or the disabled. There may not be other persons available around to look after them. These are the extreme cases when there arises the need of robots to help the persons in distress. They are called medical assistive robots. They must be handy and user-friendly, and needless to say, of low cost. People from all ages, jobs or economic health should be able to use it. We ventured to build a medical assistive robot that meets all the above-mentioned requirements. It will help people to bring necessary objects from a certain place to the user after being trained, ready to be used both in voice-command control and direct keyboard control via bluetooth.**

Keywords : Handshaking ,Voice activity Detection(VAD), Mel Frequency Cepstral Coefficients (MFCCs) , Dynamic Time Warping (DTW).

I. Introduction

Robotics is the newest field of applied technology and needless to say the most popular one nowadays. From industry to household, military to medical facility, everywhere the works are being dominated by robots or automation. This project deals with the aspect of medical assistance of automation, aiming to assist the sick and the disabled. Being user friendly and voice controlled, it can be used in large scale in hospitals or other medical facilities and also in home.

The goal of our project was to help the medically challenged persons; we engaged to determine how to help them using our robotic resources in the simplest possible and economic way. We thus settled for voice command-controlled automaton. We have come to build a robot which takes instructions from the patient to fetch any particular object, which can be medicine or food or any other light but necessary objects in a certain room. What actually takes place here is that, the robot is trained using some pre-recorded voice commands, which are taken from different users to ensure the process of identifying a certain command. The commands are processed through MATLAB using MFCC based Dynamic time warping. There are two modes of the robot, i.e. Training and Testing. In the training phase, the robot is trained to reach the destination of an object in the room, and then it will return to the origin point in the shortest possible path. Several paths can be trained for different objects. In the testing phase, the user will instruct the robot to bring an object which was already trained in the previous phase, and then the robot will follow that path to the destination, grab it using a gripping mechanical hand and return to the initial position near the user. Also, beside voice-command, it can also be trained using simply MATLAB Graphical User Interface. The instructions are sent using Bluetooth from laptop.

II. Structure

The robot resembles a 7x9 rectangular structure with a height of 6. We used white cardboard to make the structure light.

We used two DC motors, with the rating of 12 V, 41 rpm. They are placed in the rear side of the robot which are attached to the rectangular structure. Two plastic and rubber-bodied wheels are attached to the rotor shafts of the motors. There is no wheel in the front part, rather a caster ball to allow free movement of the robot.

A servo motor [55gm weight, Torque 13kg-cm at 4.8V, Operating Voltage 4.8-7.2V, Connecting wire: Heavy Duty, 300mm] was used to move the gripper hand which is in the front of the body. The mechanical structure of the hand was first designed in AUTOCAD and then was implemented to plastic to shape the hand/gripper. The servo motor is connected to only one gear, which is mechanically connected to another one, which move mutually. Each gear is a part of a hand. On the board, there is a PCB set upon a raised level of three inches using four screws. And the electrical wires are sustained there. Below the PCB is an ARDUINO MEGA, attached to it; meanwhile, above, there is an ARDUINO NANO.

III. Electrical Circuitry

The total circuit is completed in just one shield like PCB. The PCB contains all the standalone features to satisfy the electrical requirements of the whole system. The PCB contains the connections for

i. Two Arduino Board one Arduino Mega 2560 and one Arduino Nano.
ii. Two 7805 IC
iii. One Metal gear servo motor
iv. One 12V battery
v. One buck module
vi. One Bluetooth module
vii. One l293d IC
viii. Two +12V DC motor
ix. Four slide switches
x. One proximity sensor
xi. Four LEDs

A. The Peripherals:

1) Arduino Boards: : Arduino Mega 2560 is the master processor. This is used for the core programming which operates the whole system. It contains the code for the two phase code- the test phase and the train phase.

Arduino Nano is the slave processor. This is used only for the

serial data reception. When it detects any data in its input, it delivers the data to the Arduino Mega 2560 through single handshaking input and output mechanism. The handshaking is done through two pins, (lets assume, strobe pin and acknowledge pin) in both of the Arduino boards which are shorted together.
Both the board operates at +5V and the default 16 MHz clock frequency.

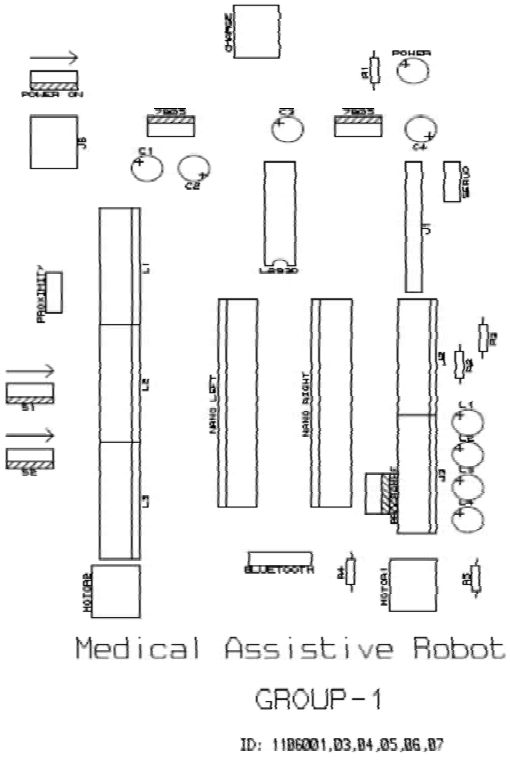

Fig. 1. PCB top silk layout

2) 7805 ICs: : Two 7805 IC sere used to step down the supply +12V to +5V, as all the logic ICs and Arduino boards need constant +5V supply for their operation. We used a 100uF capacitor in the input and a 10uF capacitor in the output for each IC. The reason behind using two ICs instead of one is that the servo motor draws excessive current, typically 0.4 Amp. Normally 7805s rated current is 1 Amp, but in order to avoid the overheating of the IC, we prefer to operate it on half of its operating current. Thats why we used a separate 7805 IC to operate the servo motor so that the other ICs operations are not hampered.

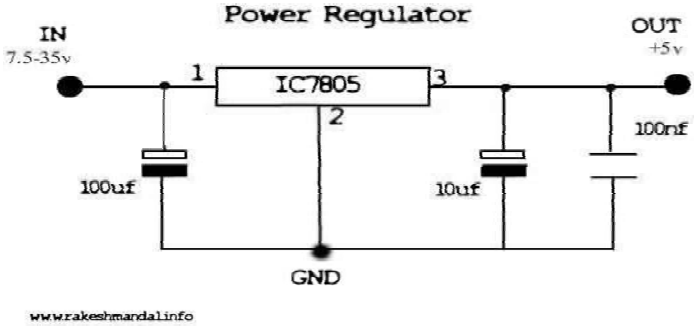

Fig. 2. 7805 connection diagram

3) Servo Motor: : One half rotational servo motor MG996 is used for Robotic hand operation. The motor is operated at constant +5V. At +5V, the motor 0.17 sec/60 degrees and the approximate torque we get is 14 kg-cm, which is more than enough to grab a normal sized object, like medicine box, glass water or food bowl. In our code, we positioned the servo motor in between 10 and 70 degrees provided this is the sufficient range to grab and open our object depending on its size.

4) Battery and power socket:: We bought three +3.75V batteries (manufactured by ANIK Telecom), soldered them together to connect them in series so that we get around +11.8 volts at the output port. Each battery is capable of supplying 3500 milliamps, which was sufficient for the whole system. During test runs, we observed that the battery would nominally last around 3-4 hours with one full charge.
The PCB also contains an on board DC socket to charge up the battery, when the charge depleted below +5V. This greatly eases the procedure, otherwise the battery is needed to disconnect from PCB, insert in breadboard and charge up.

5) Buck module : For our system, we need to supply a constant voltage during its runtime. During test and train phase, the robot needs to precisely turn 90 degrees left or right; even an 89 or 91 degree rotation would accumulate into a large offset. As we implemented the motors constant angle rotation through an open loop scheme by using time domain delay instruction, its precision greatly depends on the motors input supply voltage. But as our supply voltage was rechargeable, with time, the charge would deplete and the rotation angle would decrease which is not expected. Thats why we used a buck module in between the supply battery and the input port. This module effectively steps down the voltage in expense of higher input current. Before the final run, in the calibration period, we rotated the potentiometer knob in the buck module to set the voltage at constant +10V.

6) Bluetooth module : We used a HC-06 Bluetooth module for Serial data transfer in slave mode. We used it only in Rx mode.

That means the Bluetooth module only receives data from PC interface, not the other way around. HC-06 operates on +5V. The module is to be paired on MATLAB interface before every run.

7) l293d IC:: One L293d IC is used for the DC motor driving. The IC is built-in with two H-bridge driver circuits to drive two DC motors simultaneously. The advantage of using this IC is that it can run the motor in both directions: clockwise and anti-clockwise through simple PWM command in Arduino code. Also it can extract sufficient current from supply input for driving the motors.

8) DC motors : Two +12V DC motor were used for the movement of the robot through wheels. The rated current was 0.38 amps for each motor, which the system was capable to supply. Rated torque at +12V is 1.5 kg-cm. As our total structure is approximately 500 grams (We actually measured), both motors can smoothly move at rated speed without any jerking.

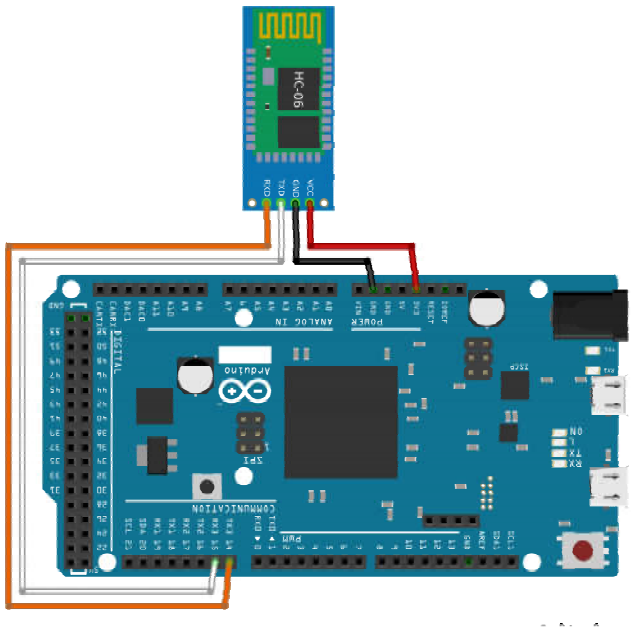

Fig. 3. HC-06 connection diagram

9) Slide switches:: One slide switch is for the power on/off option. Another one is provided to connect/disconnect the Arduino Mega 2560 Rx pin and Arduino Nano Tx pin. Because it is highly recommended that while uploading the code to either of the board, the Tx or Rx pin remains open. But during runtime, the Arduino Mega 2560 Rx pin and Arduino Nano Tx pin need to be connected for one-way data transfer. The two other slide switches are to select either train mode or test mode.

10) Proximity sensor: : In our PCB, we provided a port for proximity sensor for obstacle detection and avoiding. The plan was such that during Test mode, if it detected any obstacle on its pre-trained path, it would intelligently avoid the obstacle by diverting into another equivalent path. The sensor we tried to use is Grove- 80 cm Infrared proximity sensor. It operates on +5V and draws very negligible current, around 30 mA. Unfortunately, we could integrate the sensor in the system at the last moment for lack of time.

11) LEDs: : They are for the debugging purpose of the Arduino Mega 2560 code.

IV. Voice Processing:

Both voice training and voice detection are possible in our matlab app. In the training mode, different voice samples of a word are recorded by audiorecorder tools of matlab. This recorded voice is labeled, then preprocessed. The preprocessing stage starts with Voice Activity Detection algorithm, which removes noisy and inactive signal from voice, then MFCC features are calculated from the modified voice signal. The features are saved in a folder for comparison. In detection mode, a voice sample is recorded, then vad is applied and MFCC features are calculated. The similarities between these features and the features of recorded voice signal are calculated - from which the best matched voice is decided.

A. *Voice Activity Detection(VAD):*

Voice Activity Detection is a very popular tool to detect silent part of a speech signal. Many algorithms have been developed in this respect. The main difference between those is the features used. In the project, 2 or 3 sec of voice signal was divided in different frames. Each frame was of 10 ms voice signal. No window function was applied on the frames. Three features were used per frame. They are - short-term energy, spectral flatness, and dominant frequency component. Energy is the most common features in voice activity detection. But it is not enough for voice detection, because this feature loses its efficiency in lower SNR condition. The second feature is Spectral Flatness Measure(SFM). It detects the noisiness of a signal. The equation for this is :

$$\text{SFM}_{dB} = 10log(\frac{G_m}{A_m})$$

Here G_m and A_m arithmetic and geometric means of speech spectrum respectively. The third feature is dominant frequency component of the speech frame. The values of different features in active and inactive segments are given below :

TABLE I
INACTIVE SEGMENT FEATURES

Energy	Dominant	Flatness
0.0593	0.0938	-inf
0.1631	0.1514	-1.7063
0.3214	0.1851	-2.4147
0.4845	0.2814	-2.6867
0.8602	0.4009	-2.6232
0.8553	0.3858	-3.1424

By observing the features in different active and inactive segment, we have chosen the threshold values as energy $>$ 5, flatness $>$ -1, dominant $>$ 0.6.

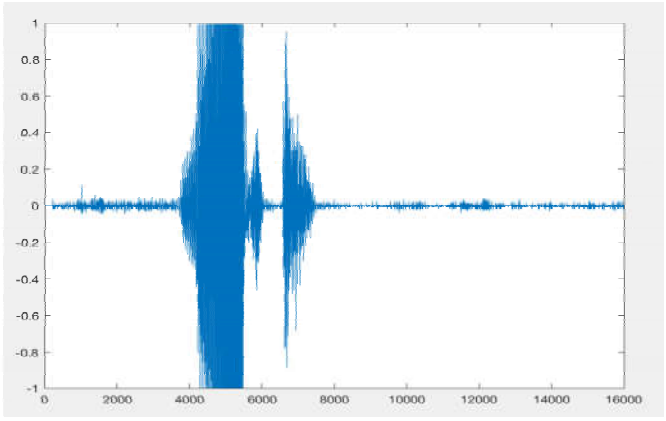

Fig. 4. Voice signal before VAD is applied

B. *MFCC*

Mel Frequency Cepstral Coefficents (MFCCs) are a feature widely used in automatic speech and speaker recognition. The speech signal is first preemphasised using a first order FIR filter with preemphasis coefficient.

TABLE II
ACTIVE SEGMENT FEATURES

Energy	Dominant	Flatness
14.1493	1.7758	-5.7462
33.7600	3.2180	-5.5772
73.0898	4.0163	-7.1328
87.6643	4.1686	-6.5090
104.6514	5.0374	-2.8602
153.0714 v5.5791	-3.2466	

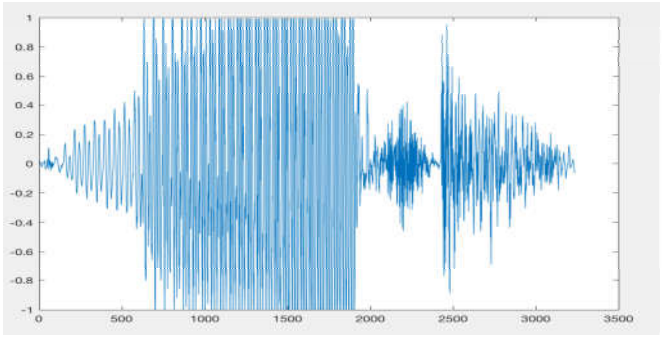

Fig. 5. Voice signal after VAD is applied

The preemphasised speech signal is subjected to the short-time Fourier transform analysis with frame durations of TW (ms), frame shifts of TS (ms) . This is followed by magnitude spectrum computation followed by filterbank design with M triangular filters uniformly spaced on the mel scale between lower and upper frequency limits given in R (Hz). The formula for converting from frequency to Mel scale is:

$$M(f) = 1125\ln(1+\frac{f}{700})$$

The filterbank is applied to the magnitude spectrum values to produce filterbank energies. Log-compressed FBEs are then decorrelated using the discrete cosine transform to produce cepstral coefficients. Final step applies sinusoidal lifter to produce MFCCs.

C. Dynamic Time Warping (DTW)

Dynamic time warping (DTW) is a time series alignment algorithm developed originally for speech recognition. It aims at aligning two sequences of feature vectors by warping the time axis iteratively until an optimal match (according to a suitable metrics) between the two sequences is found. Two sequences can be arranged on the sides of a grid, with one on the top and the other up the left hand side. Both sequences start on the bottom left of the grid. Inside each cell a distance measure can be placed, comparing the corresponding elements of the two sequences. To find the best match or alignment between these two sequences one need to find a path through the grid which minimizes the total distance between them. The procedure for computing this overall distance involves finding all possible routes through the grid and for each one computes the overall distance. The overall distance is the minimum of the sum of the distances between the individual elements on the path divided by the sum of the weighting function.

In our algorithm, we have modified the DTW algorithm. We have divided our testing data in small windows of 25 ms, then we extracted 13 MFCC features from each window. The first MFCC feature denotes the intensity of the voice. As we do not want our algorithm to be intensity dependent, we ignored the first feature, and constructed a feature vector of 2-13 MFCC features.Number of feature in MFCC depends on the duration of voice activity. A word slowly uttered may result in larger number of MFCC features than the same word uttered fast. So, we applied DTW algorithm on the feature vectors of training and testing data, which calculates the distance based cost of the two feature vectors.

TABLE III
DTW-BASED DISTANCE SCORE

train \ test	water	stop	medicine	left	Food
Water	803.44	926.64	987.4	868.08	848.36
Stop	961.05	442.63	876.43	606.53	685.34
Medicine	1218.1	1278.3	861.51	989.84	1117.8
Left	1135.5	1227.9	1010.0	920.82	1036.0
food	1101.6	1144.2	1014.5	1029.0	899.07

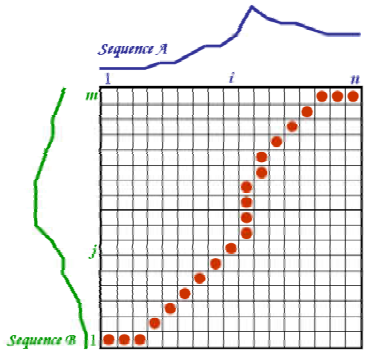

Fig. 6. DTW grid

There were different versions of each word in our testing dataset (only 6 samples each!) . We have calculated the dtw cost for each sample and then calculated the average cost. The word which gives the lowest cost is the matched word.

From our experiment, we have seen that our algorithm matched all the voices perfectly.

V. Arduino Coding:

Two different Arduino codes were written, compiled and uploaded to two different Arduino processors

A. Arduino nano

We used this dedicated processor to search continuously for Serial data from the Bluetooth module. The reason behind is that if the master processor is assigned to do this task, it cant run any other statement in the code. Our first plan was to utilize the dedicated interrupt pin in Arduino board through AttachInterrupt and DetachInterrupt function. But Arduino cant transfer serial data in interrupt vector. Thats where Arduino Nano cames into play. It implements single handshaking input and output mechanism to send data to Arduino Mega 2560 when any is available.

B. Arduino Mega 2560:

We used this processor for DC motor control operation, hand control operation and serial data reception. We used two Arduino header files in this code, EEprom.h and servo.h.

1) Serial Data Transfer:: In both Arduino code, in setup function, the baud rate is set to 9600. In its continual search for Serial data,if the Arduino Nano detects any data, it immediately sets the strobe pin and sends the data to output buffer. Seeing this pin high, the Arduino Mega 2560 sets the acknowledge pin and receives the data from the input buffer. While the acknowledge pin remains high, the Arduino Nano waits. After completing the reception, the Arduino Mega

2560 resets the acknowledge pin. Seeing this pin low, The Arduino Nano restes the strobe pin and resumes its quest for further data.

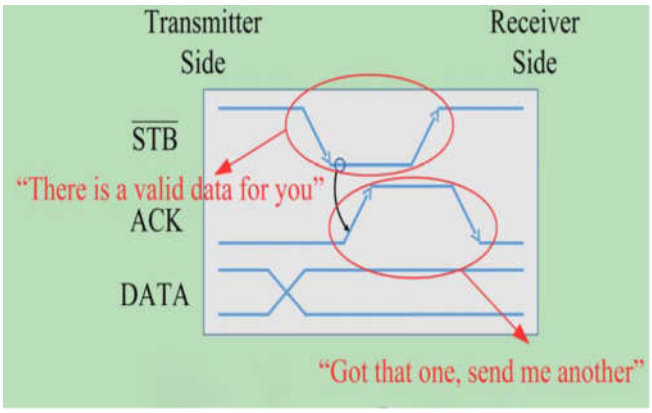

Fig. 7. Handshaking mechanism

2) DC motor control: We controlled the DC motor position through open loop. We imagined the whole plane upon which the robot would move as a rectangular grid. A position in the plane is defined by certain X coordinate and Y coordinate value.

We defined the initial value of X coordinate as 0. As the EEPROM cannot save any negative number (its data range is from 0 to 255), the initial value of Y coordinate was defined 128.

We also defined a certain resolution for forward command. During the train mode, if one forward command is given, the robot will step forward in the initial direction that precise amount and the coordinate value (x or y) will change by one. In the left direction, the y coordinate decreases by one and in the right direction increases by one.

In the loop function, the code will first look into whether the train or test mode is activated. If the train mode is on, the motor waits for the forward/backward/left/right command as one step at a time and changes the coordinate accordingly. When it will reach the target position of a desired object, it will save the X and Y coordinate in the EEPROM against that object number and return to the initial position in the shortest path by remembering the path. In the same way, the robot can be trained for several objects. When the test mode is activated, after given the object number command, the robot immediately reads the EEEPROM X and Y coordinate values and follows the shortest path to reach the object. This is done in a FOR loop in which the main objective of the code is to change the X coordinate value to 0 and the Y coordinate value to 128.

The direction change operation is done in left and right function. When either of this function is called, the appropriate PWM pins are set for changing directions of motor and then the delay function is called. After much calibration, we found that at +10V supply voltage, if the argument of the delay function is passed 1320, the robot would turn precisely left or right, according to the initial PWM directions.

VI. GUI Interface:

We have developed a MATLAB GUI Interface for the whole task, from voice training and detection to controlling the robot. First the robot has to be paired with MATLAB through bluetooth connection.

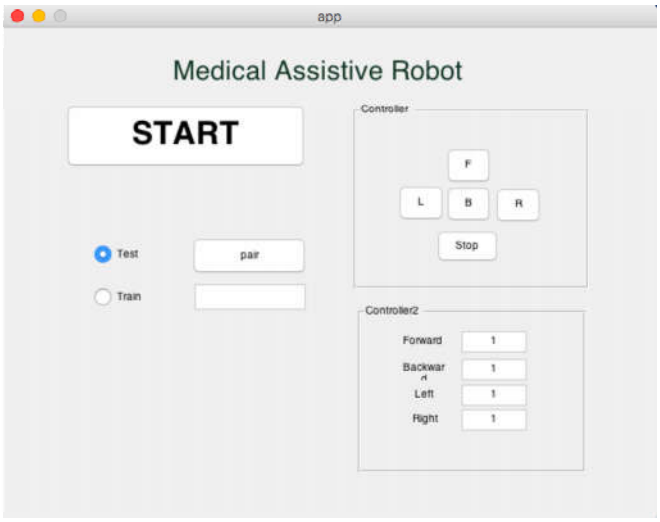

Fig. 8. GUI interface

VII. Costs:

TABLE IV
PCB

Name	Size	Price
PCB double layer (with green masking):	6.2 3.6	650.00

TABLE V
Soldering

Name	Pieces	Price
Rung	1	150.00
Rasin	*1*	50.00

TABLE VI
PCB Mountings

Name	Pieces	Price
Capacitors	4	15.00
Resistors	5	10.00
L293D IC	1	84.00
Green connectors	3	18.00
7805 IC	2	14.00
LEDs	4	4.00
Serial rails (long)	2	44.00
Serial rails (short)	2	20.00
Slide switches	4	16.00
DC socket	1	8.00

TABLE VII
PCB Peripherals

Name	Pieces	Price
DC motors	2	1800.00
Servo motor	1	630.00
Arduino Nano Board	1	642.00
Arduino Mega 2560 Board (Original)	1	4590.00
Bluetooth Module	1	560.00
Buck module	1	250.00
+12V battery	1	960.00

TABLE VIII
STRUCTURE-HAND

Name	Pieces	Price
Raw materials- Acrylic Sheet	1	200.00
Sheet laser printing		500
Screws and bolts	20	50.00
Gripper(Foam and glue)		200.00

TABLE IX
STRUCTURE-BODY

Name	Pieces	Price
Plastic board	1	120.00
Wheels	2	1000.00
Caster ball	1	95.00
Nails and bolts	10	30.00

VIII. FUTURE DEVELOPEMENT:

In our PCB, there are pins for further circuitry. For example, adding digital compass in the PCB would increase the accuracy of the movement to a great extent. Also, digital compass would have allowed the robot to rotate in different angle , i.e. 30 degree or 15 degree movement. There is also option for proximity sensor, which would have been used in obstacle detection.

IX. CONCLUSIONS

We have developed the robot in such a way that it can be trained to meet specific needs in a particular clinical environment and respond to the voice of a patient. We have used MFCC based Dynamic Time Warping voice detection system that can detect different voices perfectly, even in a low SNR environment. The connectivity between the laptop and the Arduino was based on Bluetooth connection, which also works fine in mid-range distance. The rpm of the robot was chosen so that it can cover the distance in a cabin quite satisfactorily. It can be trained to carry light object like food, medicine, water bottle etc. and it can be easily modified to carry heavier objects within the existing system. Finally, this robot can be manufactured commercially to give service in clinics and hospitals.

REFERENCES

[1] M. H. Moattar and M. M. Homayounpour, "A Simple But Efficient Real-time Voice Activity Detection Algorithm," in *EUSIPCO*, 2009.

[2] Cory Myers and Aaron E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition."