

Report

**Smart Wheelchair**

Supun Rathnayake

10018712

Mr. Dileepa Jayamanne

Submitted in partial fulfillment

Of the requirements of the

Bsc (Hons) in Software Engineering

2020.07.26

Smart wheel chair

Introduction

Nowadays in hospitals they are using most powerful and high technologies. Because only that place can save human lives therefore they have high risk and everything should happen in perfect manner. So we already know they are using many different technologies to different diseases. As an example assume to identify a cancer they are using many technologies like computerized tomography (CT) scan, bone scan, magnetic resonance imaging (MRI), positron emission tomography (PET) scan. Likewise they have more technologies to identify more diseases. Other than those things they have technologies for their equipment as well. As an example they have beds can control using remotes.

Smart Wheel seat with mechanical control devices designed to move with the help of user commands. This reduces human effort and the ability to drive wheelchairs. In addition it also provides an opportunity for people with visual impairments to move from one place to another. A wheelchair is also provided with an obstacle detection system that reduces the risk of collisions while traveling.

Smart Wheelchair has gained a lot of interest in recent times. These devices are especially useful for transporting from one place to another. These devices can also be used in nursing homes where elderly people have difficulty walking. Devices serve as a blessing to those who have lost their mobility.

Various types of smart wheelchairs have been developed in the past but new generations of wheelchairs are being developed and used with artificial intelligence which is why it leaves little room for the user of the wheelchair user. The project also aims to build a similar wheelchair that will have some kind of ingenuity and thus help the user on his journey.

So I supposed to develop a smart wheel chair to travel between wards. Actually nowadays we can see many smart wheel chairs in hospitals. But the thing is those smart chairs working under human control. That’s mean those chairs cannot control it’s self alone. So need a person to guide route. Sometimes it’s under control remote or driving panel. So this is good for normal patients who cannot walk alone. But the patient should able to control driving panel. Otherwise no way to drive wheel chair alone. So assume there is a blind patient then the patient cannot control the wheel chair alone. That’s why the patient able to handle the driving panel but he or she cannot identify the route. So I suppose this smart wheel chair especially for disabled patients. In this idea I suppose to use (HMI) human machine interface technology and robotics. So when I explained how to work this, the patient has only site on the wheelchair and give voice commands (go forward, turn left, stop and etc.) to the specific microphone. Then the voice assistant will able to control wheelchair without any human effort.

Methodology

As I mentioned I’m try to develop this control with minimum human effort. So there is 3 major parts. Those are voice assistant part for give voice commands, HMI part for active voice assistant and Arduino part for control the chair. So let’s see what the purposes of these parts are and how to connect these parts together.

* Voice assistant

This is like a guider. In mobile devices we have Siri, Bixby, Google assistant and etc. So these bots are able to help us at every time. So in this case I’m going to use a voice assistant for control the wheelchair. That’s mean the patient has sit on the wheelchair and connect with voice assistant. So when the user give some voice commands to the voice assistant then it will control the wheelchair without any human effort.

So I’m going to develop an android application to take voice commands. Simply there is an API as google API. So using this API we can develop an application for obtain voice commands. As of my opinion I’m going to develop an application like speech to text. Then the text will convert to bytes and sent to the Arduino board via Bluetooth connection.

Let’s see how I build android application for obtain voice commands.

First of all, we need to create a new Android Studio project and in the manifest file add the following user-permissions.

|  |
| --- |
| <?xml version="1.0"encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.texttospeech"> <uses-permission android:name="android.permission.RECORD\_AUDIO"/> <uses-permission android:name="android.permission.INTERNET"/> |

After that we need to create an activity\_main.xml file. And basically we are adding EditText and ImageView as the UI part. Let’s see how to develop the UI part.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent" android:layout\_height="match\_parent" tools:context=".MainActivity"> <RelativeLayout android:layout\_width="match\_parent" android:layout\_centerInParent="true" android:layout\_marginLeft="10dp" android:layout\_marginRight="10dp" android:layout\_height="wrap\_content"> <EditText android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_toLeftOf="@id/button" android:layout\_marginRight="15dp" android:padding="10dp" android:hint="Tap to Speak" android:id="@+id/text" android:layout\_centerInParent="true" /> <ImageView android:layout\_width="40dp" android:layout\_height="40dp" android:backgroundTint="#F9F8FA" android:paddingRight="10dp" android:src="@drawable/ic\_mic\_black\_off" android:layout\_alignParentRight="true" android:id="@+id/button"/> </RelativeLayout>  </RelativeLayout> |

So after we created the XML file, we need to create the related java file. Usually we named it as MainActivity.java. In the java file first of all we need to check permissions. Here we have the coding part for check permissions.

|  |
| --- |
| if(ContextCompat.checkSelfPermission(this,Manifest.permission.RECORD\_AUDIO) != PackageManager.PERMISSION\_GRANTED){ checkPermission(); } |

If the permission is not granted then we will call the checkPermission method.

|  |
| --- |
| private void checkPermission() { if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) { ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.RECORD\_AUDIO},RecordAudioRequestCode); } } |

Now here comes the important part first which will initialize the SpeecRecognizer object and then create the intent for recognizing the speech.

|  |
| --- |
| speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);  final Intent speechRecognizerIntent = new Intent(RecognizerIntent.ACTION\_RECOGNIZE\_SPEECH); speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE\_MODEL,RecognizerIntent.LANGUAGE\_MODEL\_FREE\_FORM); speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE, Locale.getDefault()); |

As you can see we have added some extras, let’s see what those things are.

The constant **ACTION\_RECOGNIZE\_SPEECH** starts an activity that will prompt the user for speech and send it through a speech recognizer.

**EXTRA\_LANGUAGE\_MODEL**: Informs the recognizer which speech model to prefer when performing ACTION\_RECOGNIZE\_SPEECH.

**LANGUAGE\_MODEL\_FREE\_FORM**: Use a language model based on free-form speech recognition.

**EXTRA\_LANGUAGE**: Optional IETF language tag (as defined by BCP 47), for example, “en-US”.

Now we will set a speechRecognitionListener to our speechRecognizer object using the setRecognitionListener() method.

You can see after setting the listener we get several methods to implement. We will go the onResults method and add the following code.

|  |
| --- |
| @Override public void onResults(Bundle bundle) { micButton.setImageResource(R.drawable.ic\_mic\_black\_off); ArrayList<String> data = bundle.getStringArrayList(SpeechRecognizer.RESULTS\_RECOGNITION); editText.setText(data.get(0)); } |

In the onBeginningOfSpeeh() method we will add the following code to tell the user that his voice is being recognized.

|  |
| --- |
| @Override public void onBeginningOfSpeech() { editText.setText("Listening..."); } |

Now, lets set up the imageView. We will add a touchListener to the image view to know when the user has pressed the image.

|  |
| --- |
| micButton.setOnTouchListener(new View.OnTouchListener() {  @Override  public boolean onTouch(View view, MotionEvent motionEvent) {  if (motionEvent.getAction() == MotionEvent.ACTION\_UP){  speechRecognizer.stopListening();  }  if (motionEvent.getAction() == MotionEvent.ACTION\_DOWN){  micButton.setImageResource(R.drawable.ic\_mic\_black\_24dp);  speechRecognizer.startListening(speechRecognizerIntent);  }  return false;  }  }); |

When the user taps the imageView the listener starts listening and the imageView source image is also changed to update the user that his voice is being listened to.

This is the full source code.

|  |
| --- |
| package com.example.texttospeech;  import androidx.annotation.NonNull; import androidx.appcompat.app.AppCompatActivity; import androidx.core.app.ActivityCompat; import androidx.core.content.ContextCompat;  import android.Manifest; import android.content.Intent; import android.content.pm.PackageManager; import android.os.Build; import android.os.Bundle; import android.speech.RecognitionListener; import android.speech.RecognizerIntent; import android.speech.SpeechRecognizer; import android.view.MotionEvent; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast;  import java.util.ArrayList; import java.util.Locale;  public class MainActivity extends AppCompatActivity { public static final Integer RecordAudioRequestCode = 1; private SpeechRecognizer speechRecognizer; private EditText editText; private ImageView micButton;  @Override protected void onCreate(final Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main); if(ContextCompat.checkSelfPermission(this,Manifest.permission.RECORD\_AUDIO) != PackageManager.PERMISSION\_GRANTED){ checkPermission(); }  editText = findViewById(R.id.text); micButton = findViewById(R.id.button); speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);  final Intent speechRecognizerIntent = new Intent(RecognizerIntent.ACTION\_RECOGNIZE\_SPEECH); speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE\_MODEL,RecognizerIntent.LANGUAGE\_MODEL\_FREE\_FORM); speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA\_LANGUAGE, Locale.getDefault());  speechRecognizer.setRecognitionListener(new RecognitionListener() { @Override public void onReadyForSpeech(Bundle bundle) {  }  @Override public void onBeginningOfSpeech() { editText.setText(""); editText.setHint("Listening..."); }  @Override public void onRmsChanged(float v) {  }  @Override public void onBufferReceived(byte[] bytes) {  }  @Override public void onEndOfSpeech() {  }  @Override public void onError(int i) {  }  @Override public void onResults(Bundle bundle) { micButton.setImageResource(R.drawable.ic\_mic\_black\_off); ArrayList<String> data = bundle.getStringArrayList(SpeechRecognizer.RESULTS\_RECOGNITION); editText.setText(data.get(0)); }  @Override public void onPartialResults(Bundle bundle) {  }  @Override public void onEvent(int i, Bundle bundle) {  } });  micButton.setOnTouchListener(new View.OnTouchListener() { @Override public boolean onTouch(View view, MotionEvent motionEvent) { if (motionEvent.getAction() == MotionEvent.ACTION\_UP){ speechRecognizer.stopListening(); } if (motionEvent.getAction() == MotionEvent.ACTION\_DOWN){ micButton.setImageResource(R.drawable.ic\_mic\_black\_24dp); speechRecognizer.startListening(speechRecognizerIntent); } return false; } });   }  @Override protected void onDestroy() { super.onDestroy(); speechRecognizer.destroy(); }  private void checkPermission() { if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) { ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.RECORD\_AUDIO},RecordAudioRequestCode); } }  @Override public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) { super.onRequestPermissionsResult(requestCode, permissions, grantResults); if (requestCode == RecordAudioRequestCode && grantResults.length > 0 ){ if(grantResults[0] == PackageManager.PERMISSION\_GRANTED) Toast.makeText(this,"Permission Granted",Toast.LENGTH\_SHORT).show(); } } } |

* HMI dash board

I supposed to use HMI (Human Machine Interface) to control wheelchair. Because directly we can’t control live system using voice assistant. So for that we can use HMI. Using HMI we can control any live system using monitor dash board or any visible interface. So in this case I’m supposed to connect voice assistant and Arduino part as well.

In the case my HMI interface will be a mobile phone display. So there is no any button to control my live system. Only thing is we have a button to put voice commands. Simply what we have to do is tap the mic button and input the voice commands. Then it will transfer to the Arduino board via Bluetooth connection.

Basically what I do is, created String variables for relevant key words. Then take the voice command from user and turn it into the text. Then I assigned passwords for each and every relevant key word. After that I will compare the voice commands and crated key words. If it is matched, I’ll send the password to the Arduino board via a Bluetooth connection. Let’s see how does the Bluetooth connection build and work it.

When we talk about Bluetooth connection, there is few main occasions. As I mentioned to connect android application and Arduino board I used HC-05 Bluetooth module. First of all need to connect android application and Bluetooth module. For that I had to use UUID. The UUID stands for Universally Unique Identifier. UUID is a simple 128 bit digit which uniquely distributed across the world.

Bluetooth sends data over the air and all nearby devices can access it. Suppose, for example, that you have to send important files via Bluetooth and that all nearby devices can access them at a distance. So when pairing with other devices, they simply share the UUID number and pair before sharing the files. If you send any file and your device encrypt that file with the appropriate UUID device then share it over the network. Now all Bluetooth devices in that range can access the encryption file but require the correct UUID number. Therefore only the right UUID devices have file encryption access and some will deny incorrect UUID cause.

This is the code for build the connectivity between android app and HC-05 module.

|  |
| --- |
| private UUID mDeviceUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); |

Then I defined variable for each and every voice command and assigned passwords for those variables. That’s why, when we communicated via a Bluetooth we need to convert commands in to Bytes and send it. Using small passwords we can easily convert it into Bytes.

|  |
| --- |
| final static String forward="1";  final static String right="2";  final static String left="3";  final static String back="4";  final static String stop="5"; |

Basically I’m using an equaling method to identify voice commands. So as I mentioned I’ll take voice commands through Google API. Then I’ll compare those things with pre-defined variables. Then I’ll convert the relevant password and send it to the Arduino board. When we communicate using Bluetooth, there is main two parts like send data and receive data. To send data we need use method getOutputStream() and to receive data we need to use getInputStream() method.

|  |
| --- |
| if(data.get(0).equals("move forward")){  try {  Toast.makeText(getApplicationContext(),"Successfully recognized", Toast.LENGTH\_LONG).show();  mBTSocket.getOutputStream().write(forward.getBytes());  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  } |

* Arduino Part

After user gave the commands, the commands will be convert in to bytes and sent to the Arduino board via Bluetooth connection. In this case I’m supposed to use HC-06 Bluetooth module to communicate between Arduino board and mobile application. HC-06 Bluetooth module only act as slave part. Actually I also need only slave device. Because I supposed sent commands to the Arduino board and no need return any value to the start point. Also there is an Arduino Uno board to control wheelchair as well.

Let’s see about Arduino coding part. in Arduino Uno we have only one Serial port which is available on pin number 0 and pin number 1. As I always say never use these pins with any Serial communication supported devices. The Arduino’s default Serial Port should only be used for debugging purposes. You can always define other Serial Ports using the SoftwareSerial library. So that’s the reason I added the softwareSerial.h header file.

|  |
| --- |
| #include <SoftwareSerial.h> |

I defined a Serial Port with the name Blue on pin number 2 and pin number 3 of the Arduino. The pin number is the Rx while pin number 3 is the Tx.

|  |
| --- |
| SoftwareSerial Blue(2, 3); |

On the third line, I defined a variable data of the type long int. This variable will be used to store the number which is send from the android cell phone.

|  |
| --- |
| long int data; |

Wheels are connected with pin numbers 7,8,12 and 13. For that I defind four variables using data type integer.

|  |
| --- |
| int wheel1 = 7;  int wheel2 = 8;  int wheel3 = 12;  int wheel4 = 13; |

Then I defined five variables password1, password2, password3, password4 and password5 of the type long integer. The password1 is used to move forward while password2 is used to come back the wheelchair also password3 is used to turn right, password4 is used to turn left and password5 is used to stop the wheelchair.

Basically I will pass the secret key from my android app and it will save on those variables for checking purpose.

|  |
| --- |
| long int password1 = 1;  long int password2 = 2;  long int password3 = 3;  long int password4 = 4;  long int password5 = 5; |

To activate the serial communication I used the Serial.begin() function while 9600 is the baud rate and similarly for the Bluetooth module.  then starts the void loop function.

|  |
| --- |
| while(Blue.available()==0) ; |

this line means that if the Bluetooth module has not received any data from the android cell phone then simply wait here.

|  |
| --- |
| if(Blue.available()>0) |

this condition means if the Bluetooth module has received data from the android cell phone then store the received number in variable data and then using the if conditions the number stored in variable data is compared with the password1, password2, password3, password4 and password5. If the number is equal to the password1 then the wheelchair will move forward and if the number stored in data is equal to password2 then the wheelchair will come back like wise if the number is equal to password3 then the wheelchair will turn left, number is equals to password4 then wheelchair will turn left and if the number is equal to password5 then the wheelchair will stop. So that’s all about the Arduino’s programming.

This is the completed source code for Arduino part.

|  |
| --- |
| #include <SoftwareSerial.h>  SoftwareSerial Blue(2, 3);  long int data;    int LED = 13; // Led connected  long int password1 = 92;// light on  long int password2 = 79; // light off    char state = 0;      void setup()  {    pinMode(LED, OUTPUT);  digitalWrite(LED, LOW);  Serial.begin(9600);  Blue.begin(9600);    }    void loop()  {      while(Blue.available()==0) ;    if(Blue.available()>0)  {  data = Blue.parseInt();    }  delay(400);  //Serial.print(data);    if (data == password1)  {      digitalWrite(LED,HIGH);    Serial.println("LED ON ");       }       if( data == password2)     {         digitalWrite(LED,LOW);    Serial.println("LED OFF");     }    } |

Non-technical functionalities

* Comfortability

This is especially true for different types of patients. We should therefore be concerned about their comfort or harm. When we talk about comfortability there are several key points. Let's start by talking about posture.

The width of the seats is very important. If the wheelchair is too small, it will injure the patient's hips and press them against the sides of the seat. It can irritate the patient's skin or cause skin damage. It also makes transfers more difficult if it is difficult to put the patient inside and get them out of your seat.

Seat length is important, too. If the seat is too short, your knees will hang on the edge, unsupported. If it is too long, your back will not reach the back of the wheelchair, and your spine will not be supported. A comfortable wheelchair supports your knees and back, keeping the patient in good condition.

The movements and speed are really important, in the case movements are should be really smooth. That’s why the patient should not face and difficulty while movements. Also the speed also should be same as movements. If it is accelerate suddenly, it can be an effect to the patient. So the increment of speedo meter should be very smooth.

* Durability

The wheelchair can be use long time without any repairs. Because if it is happened any damages and repairs after very few rides, it’s not possible. So it can be use more time without any problem. Especially smart wheelchairs, that’s why compare with normal wheelchair the building cost is expensive for smart wheelchair. Because it’s using high technology, electronic parts and software combination as well. So the cost of these things are very high. If we need to repair wheelchair within close periods it’s will be a huge effect to financial side.

* Reliability

Reliability is an important fact. Because this is based on patients life. So the strengthens, movements, speed and other things can be reliable. As an example if the movements are do not work at real time it will be a huge problem. And the speed also should work at real time.

Most probably this is for single use. That’s mean there is no one to control wheelchair apart from the patient. So the each and every function should be more reliable.

* Low maintenance cost

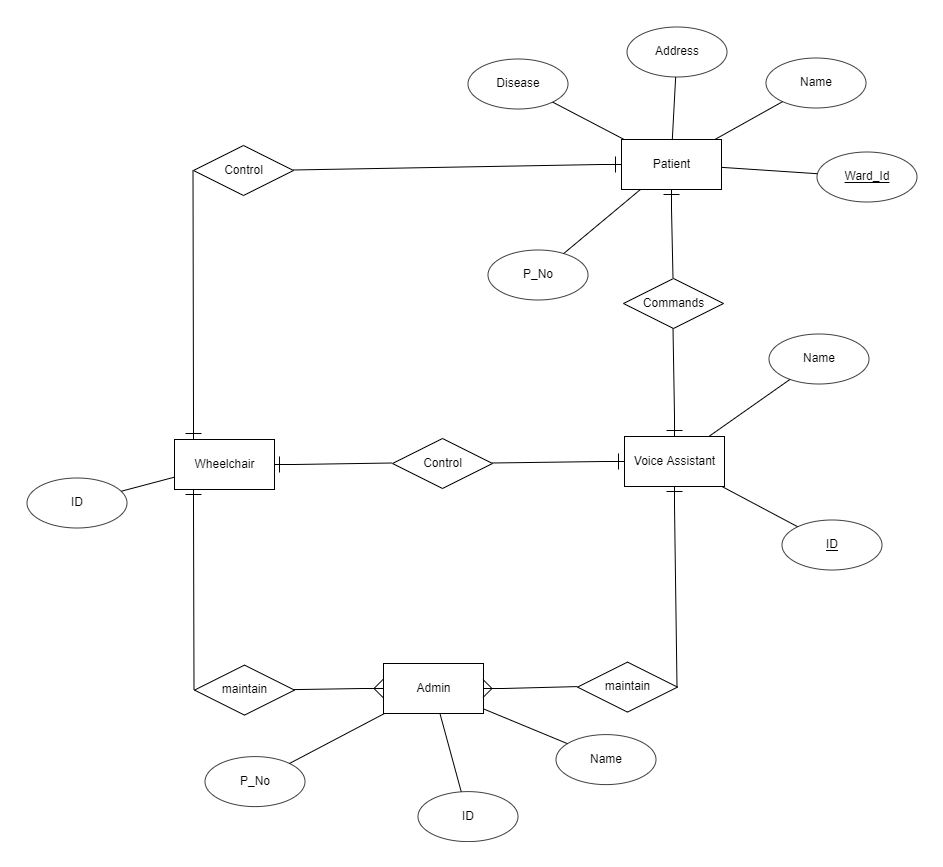
So after use this randomly it should check and do the maintenance. Then those can be cover under low cost. As an example we can check randomly about Arduino parts and refresh. Otherwise the staff have to spend more cost for maintenance. As an example if any damage to mortars, we need to replace those things using high cost. Therefore the best thing is have a random checking and fill up small needs.

* Accuracy

Accuracy is more important. Because this is used in real world. So the decisions are more important and any time user abled to give commands. Thing is each and every commands should work on time otherwise user will be idle. As an example if user asked to turn left it’s should able to turn left immediately. Also for other commands as well. Apart from that the speed control also more important. Any time user should can be control the speed as he/she wants.

Diagrams

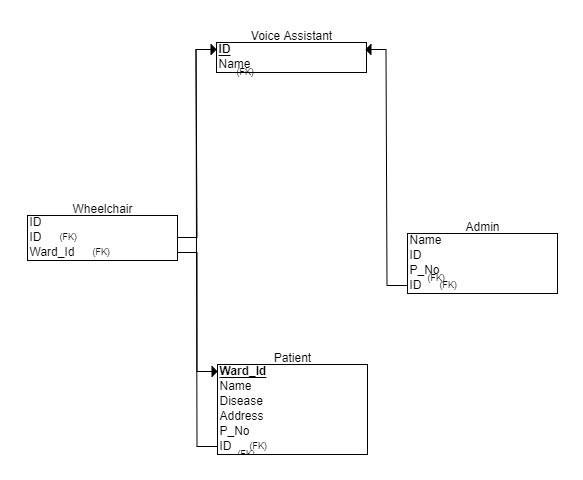
* ER diagram



This is the ER diagram related to my project. So when we talk about ER diagram there is four major entities like patient, wheelchair, voice assistant and admin. Let’s go through each and every entity. First we will consider about patient. So patient has many attributes like name, address, ward\_No, P\_No and etc. When we talk about the facilities of the patient, he or she should able to control the wheelchair using voice commands (through the voice assistant) or manually. Next we will consider about the wheelchair. So the wheelchair has only attribute as an ID. Because there can be several wheelchairs and only can identify using ID.

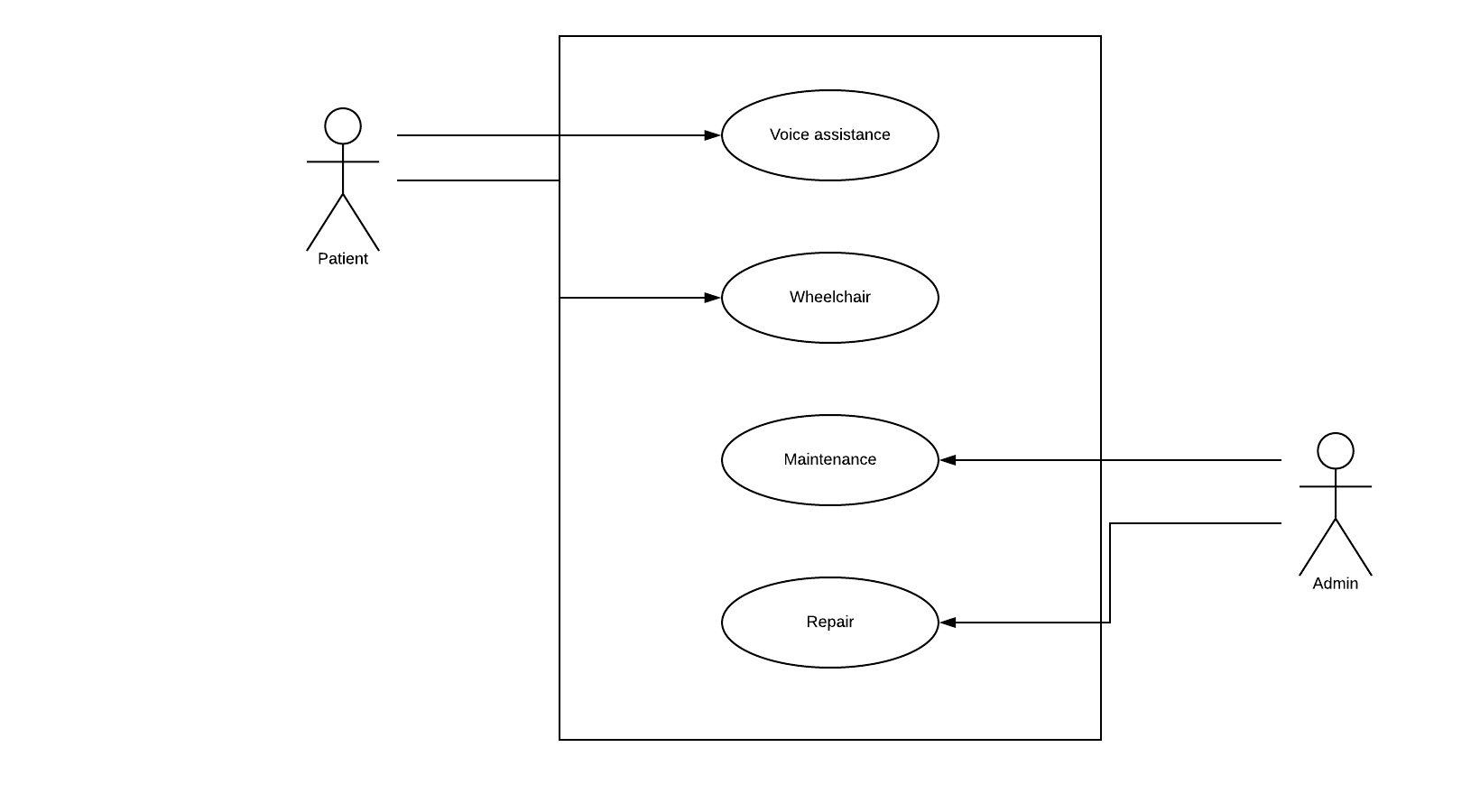
Apart from that next entity is voice assistant. When we talk about voice assistant it has name and ID as main attributes. So when the patient give the voice commands voice assistant will control the wheelchair under the voice commands. Last entity is admin. When we talk about admin it’s contains name, P\_No and ID. So admin can maintain the wheelchair and voice assistant as the user requirements or if any damange.

* Relational schema



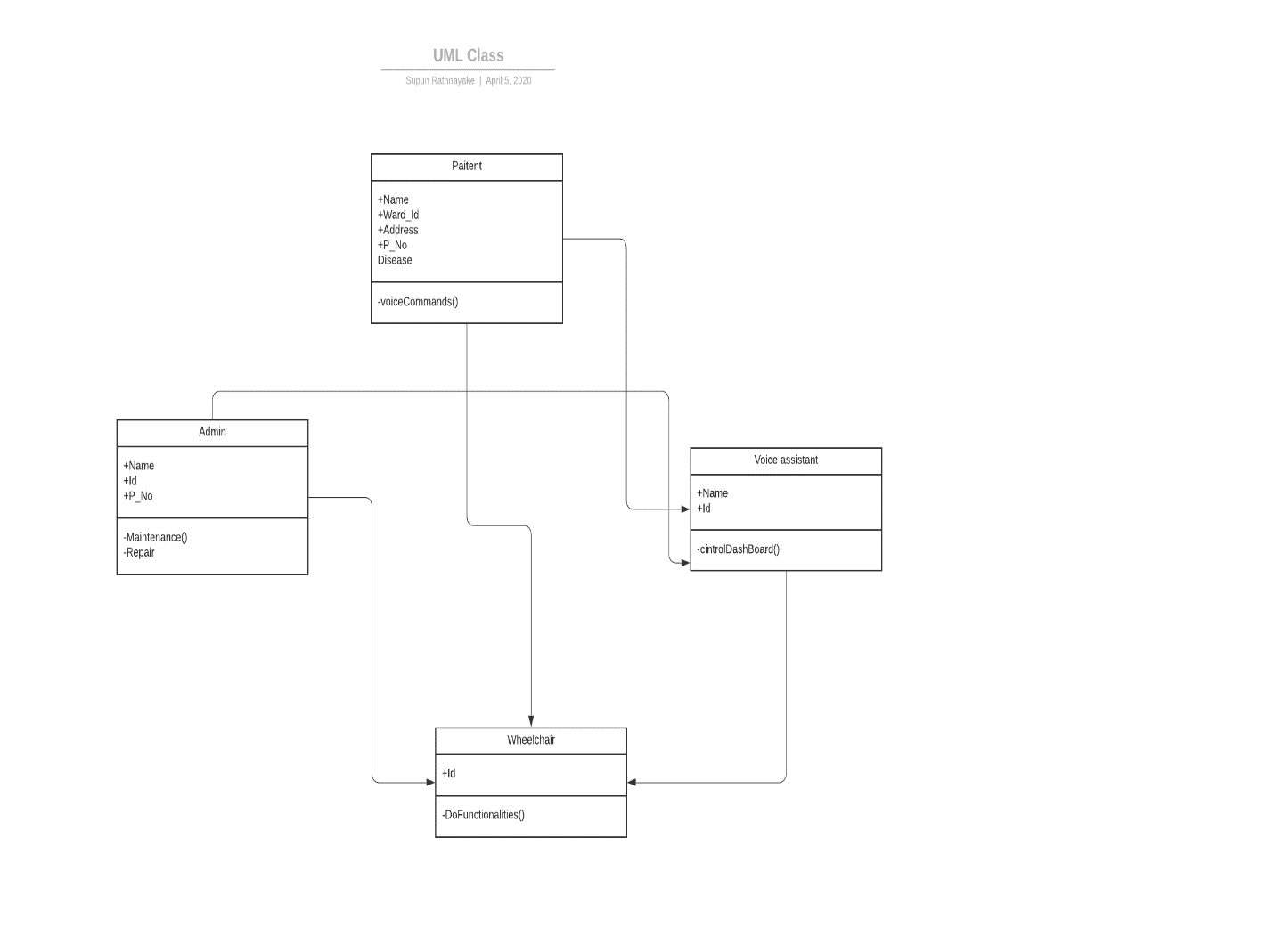
This is the relational schema related to above ER diagram. When we talk about relational schema each and every entity including primary key. Apart from voice assistant other are including foreign keys as well.

* Use case diagram



This is the UML use case diagram according to smart wheelchair. In the case there is four main use cases as voice assistant, wheelchair, maintain and repair. Likewise including two main actors as patient and admin. So the patient can access voice assistant and wheelchair manually. Otherwise the admin team able to do maintenance and repairs.

* Class diagram



This is the class diagram related to smart wheelchair.

Results

Literature Review

As I mentioned I supposed to develop a smart wheelchair using HMI as my final year project. So before I choose this topic I searched about HMI projects. At that moment I saw some projects those are similar to my project. Then I go through those projects and checked how there are completed and what are the technologies include in those.

[1] A wheelchair controlled with voice commands and assisted by a reactive Fuzzy\_ Logic controller.

This is describes new results with a Reactive Shared-Control system that enables a semi-autonomous navigation of a wheelchair in unknown and dynamic environments. The purpose of the reactive shared controller is to assist wheelchair users providing an easier and safer navigation. It is designed as a fuzzy-logic controller and follows a behavior-based architecture. The implemented behaviors are three: intelligent obstacle avoidance, collision detection and contour following. Intelligent obstacle avoidance blends user commands, from voice or joystick, with an obstacle avoidance behavior. Therefore, the user and the vehicle share the control of the wheelchair.

[2] Fuzzy logic control and HMI interfaces of an intelligent wheelchair system.

In this article an obstacle avoidance system and a Fuzzy Logic Controller (FLC), together with a distributed system of embedded microcontrollers, are presented. The Fuzzy logic algorithm is a powerful solution among Soft computing techniques, which deals with certain control problems especially when the situation is undefined and ambiguous. One of the main problems faced by conventional control systems is the inability to operate when information is incomplete and imprecise. As the complexity of the system increases, a traditional mathematical model will be difficult to implement. The paper detailed the FLC algorithm controlling the Wheelchair system. The controller enables the wheelchair movement by avoiding and encountered obstacles. The wheelchair can make brake in dangerous environment by the infrared and ultrasonic sensors. We focus the study on non-conventional control algorithm implemented using a PC interfaced with an embedded microcontroller platform. Voice and face recognition algorithm have been also tested on the system to facilitate the control of the wheelchair for disabled person. Simulation and experimental results are carried out on a wheelchair platform

[3] Wheeled robot control based on based recognition using the kinect sensor.

Human Machine Interaction (HMI) plays a very important role in intelligent service robot research. Traditional HMI methods such as keyboard and mouse cannot satisfy the high demands in some environments. To solve this problem, many researchers pay attention to vision-based gesture recognition research recently. This paper proposes a simple method to control the movement of a robot based on Kinect which provides skeleton data with low computation, acceptable performance and financial cost. The method can recognize eleven defined gestures by using the coordinates of joints, which are obtained from the skeleton model provided by the Kinect SDK. A Khepera III robot is used as a prototype control object, to verify the effectiveness of the proposed method. The experimental results show that the success rate of gesture recognition is over 96%. The proposed method is robust to work in real-time.

[4] HMI and driver distraction.

Incorporating advanced technology and connectivity in motor vehicles has the ability to make travel more entertaining and specifically far safer – but these very features may contribute to driver distraction, negating all the safety aspects. Accurate statistics in regard to distracted driving are notoriously hard to sample as most surveys rely on self-reporting for their data. Estimates of distraction as a cause or contributing factor in accidents range between a low percentages to as much as 70% of accidents. Statistics indicate that the impact of distracted driving now leads to higher fatalities. According to the National Highway Transportation Safety Administration, the number of distracted driving accidents decreased from 1,481,000 in 2004 to 1,100,000 in 2008, but the percentage of distracted driving accidents that resulted in fatalities increased from 29.8% to 48.5% during that same time period. This data suggests that when an accident involves distracted driving today it’s more likely to result in a fatality than it was in prior years.

References

[1] Gabriel Pires & Urbano Nunes, "A Wheelchair Steered through Voice Commands and Assisted by a Reactive Fuzzy-Logic Controller",July 2002,Polo II, University of Coimbra, 3030, Coimbra, Portugal,Journal of Intelligent and Robotic Systems volume 34, pages301–314(2002).

[2] Mohamed Slim Masmoudi, Imen Klabi, Mohamed Masmoudi, "FUZZY LOGIC CONTROL AND HMI INTERFACES OF AN INTELLIGENT WHEELCHAIR SYSTEM ", 2014, International Conference on Control, Engineering & Information Technology (CEIT’14), 4, pp.253-260.

[3] Yali Wang, Guangming Song, Guifang Qiao, Ying Zhang, Jun Zhang, Weiguo Wang, "Wheeled robot control based on gesture recognition using the Kinect sensor",17 March 2014,Shenzhen, China.

[4] Mark Bowler, "HMI and Driver Distraction",2012, D-10117 Berlin, Germany.