Optimizing the Control of a Wi-Fi based Teleoperated Mobile Wheelchair

China Project Center, E-term, 2014

A Major Qualifying Project (MQP) Report Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In Partial Fulfillment of the Requirements for the Degree of Bachelor of Science

Authors:

Ayesha Fathima, Robotics Engineering (atfathima@wpi.edu)

Xia Li, Computer Science (xli2@wpi.edu)

In partnership with Huazhong University of Science and Technology

Partners: Yuanjun Yao, Mechanical Engineering (317417329@qq.com)

Kun Xiong, Mechanical Engineering (1069487182@qq.com)

Liaison:

Huiping Zhang, DEPUSH Technologies Co. Ltd., Wuhan, China

Project Advisors:

Professor Yiming Rong, WPI Dept. of Mechanical Engineering

Professor Lingsong He, HUST Dept. of Mechanical Engineering

Professor Michael Ciaraldi, WPI Dept. of Computer Science

Abstract

The purpose of this project is to optimize an educational robotics platform built for students to be presented with a meaningful real-world problem: lack of control options available for people with different levels of disabilities to control a wheelchair. For this purpose, the Wi-Fi based wheelchair controlled by an Android application, is optimized. We established a robust wireless communication method and supplemented the multimodal control options with a speed regulation and GPS navigation system. The end-result is an expansible framework that can extend to other robots sold by the sponsoring company at DEPUSH Technology Co. Ltd., based in Wuhan, China.

Acknowledgement

We would like to thank Prof. Yiming Rong, Prof. Lingsong He and Prof. Michael Ciaraldi for their undying enthusiasm and support for this project. They have been excellent advisors to us and helped us through thick and thin during our project experience in China. Under their supervision, we were able to outperform during our presentation and sparked a lot of interest amongst other students. We are also grateful to the other students who attended the presentation and provided invaluable suggestions for future work on the Wi-Fi based wheelchair platform.

And last, but, not least, we would now like to extend our heartfelt gratitude to the project sponsor and liaison, Mr. Huiping Zhang at DEPUSH Technology Co. Ltd. in Wuhan, China, for giving us the wonderful opportunity to work on this project with the Huazhong University of Science and Technology (HUST) students. We are highly obliged for being shown around the company branch in Wuhan where we witnessed the incredible work being done by their engineering employees to bring innovative robotics kits to over 500 academic institutions across China.

Overall, it has been a pleasure working abroad and optimizing an innovative and cutting edge wheelchair-based project, as an educational platform for future use by university students. We will be looking forward to how our work on this project will take off when implemented and used in actual academic institutions across China.

Table of Contents

Table of Figures

List of Tables

Chapter 1: Introduction

In the “Robotic Revolution” era, many mechanical designs have been optimized to be more and more autonomous with the development of technology [1]. The mechanical design of a wheelchair is one such system that has been developing during this era. With the help of educational robotics, our sponsor at DEPUSH hopes to spread the knowledge of solving real world problems, such as teleoperating a wheelchair through various means of technology. Over the past year, an ongoing project on a Wi-Fi based wheelchair has been developed by our sponsor with the help of HUST students.

On one hand, wheelchairs are designed for people who are physically challenged and/or have disabilities that make it harder for them to walk. With the development of semi-autonomous wheelchairs, users can now choose from an array of multimodal control options available to control the wheelchair, depending on their level/type of disability. There are already several ways to control wheelchairs, from the traditionally self-propelled wheelchairs to the electrically powered wheelchairs. The electrically powered wheelchairs are the norm and requires users to control it with a joystick. The joystick may have additional controls to tailor to the user’s ability to access multiple control modes. Likewise, for users who are unable to use a joystick, various alternatives are available such as a sip-and-puff controllers, which works by sipping or blowing into a sensor that translates it into movements of the wheelchair [2]. Wheelchairs can also be controlled with voice recognition by speaking the commands out loud.

Our project, on the other hand, focuses on a transformed electrically powered wheelchair as an educational platform that can be controlled with an Android smartphone over a Wi-Fi connection. For this purpose, the student groups at Huazhong University of Science and Technology (HUST) transformed the original off-the-shelf electrically powered wheelchair that was controlled with a joystick. Even though such a wheelchair provided stable and safe transportation for the target users, it limited the user to a single control option.

Therefore, the students from HUST have been constantly working on transforming this wheelchair into a Wi-Fi based teleoperated mobile wheelchair. The students designed the wheelchair such that, it could be controlled by Android smartphones for development purposes as smartphone users are on the rise. Previously, students at HUST developed and integrated features like touch sensing, gravity sensing and online voice recognition and GPS tracking into an Android application. However, there were some problems and shortcomings in this previous design, such as poor Wi-Fi connectivity, lack of a speed regulation system, insufficient utilization of Android cell phone features and an inability to be expansible. This inability was caused by the wheelchair application either necessitating an internet connection for online features or failing to adapt to other robots.

Thus, *the goal of our project was to establish expansible control with an enhanced utilization of Android cell phone resources and improved wireless communication to control the wheelchair platform with an added speed regulation system*. To achieve this goal, we formulated 6 objectives. Firstly, we analyzed and modified the previous system architecture of the Wheelchair. Secondly, we upgraded the User Interface (UI) to be more intuitive and easy to use. Additionally, we also integrated offline features of voice recognition as well as a fully equipped offline GPS navigation system to route from one point to another. Thirdly, we enhanced the wireless communication with the Android phones by switching to another type of networking process. Next, we designed a motor controller circuit to establish a speed regulation system in the wheelchair architecture. Subsequently, we tested our new architectural capabilities on a simulator, named, “insta-car”, which made use of the same exact hardware circuit and software program as for the Wheelchair in order to demonstrate the expansible nature of our end-product. Lastly, we designed and manufactured a prototype of the new and improved Wi-Fi based wheelchair design.

Chapter 2: Literature Review

In this chapter, we discuss the background study required for the understanding of this project. Beginning with the sponsor company's background, the relationship between topics such as the impact of educational robotics in the contemporary world and human-robot interaction is explored. It is followed by an in-depth description of Android development for multimodal control of the wheelchair as well as wireless communication technologies. Furthermore, this chapter concludes with the previous project work performed on the wheelchair platform to get a better understanding of the project at hand.

Sponsor in the Spotlight

DEPUSH Technology Co., Ltd was founded in 2001 in ShenZhen, China. Its headquarters is located in Futian District, ShenZhen. It also has a branch company in Wuhan which is directly responsible for optimizing the Wi-Fi based mobile wheelchair. DEPUSH Technology emphasizes on bringing an innovative educational revolution in colleges and universities across China. DEPUSH dreams to become the first choice in the field of Chinese Engineering Quality Education. With the research of advanced engineering concepts in United States and HongKong, DEPUSH developed the project-oriented teaching method in order to further build relationships with universities and colleges in China [3, 4].

Not only is DEPUSH well-known for its iconic project-oriented teaching method, but also in providing both designs and consulting services to over 500 academies in China. DEPUSH has provided two main series of innovative educational products made by themselves, DRRob and DRLab, these two lines of products have been served in the Academies.

Products and Services

The products of DRRob consists of different kinds of robots which include humanoid-robots, six degree of freedom manipulators, Robot Sumo, Walking robot, baby-car, etc. The Figure 1 below shows the products that DEPUSH provides to the academies.

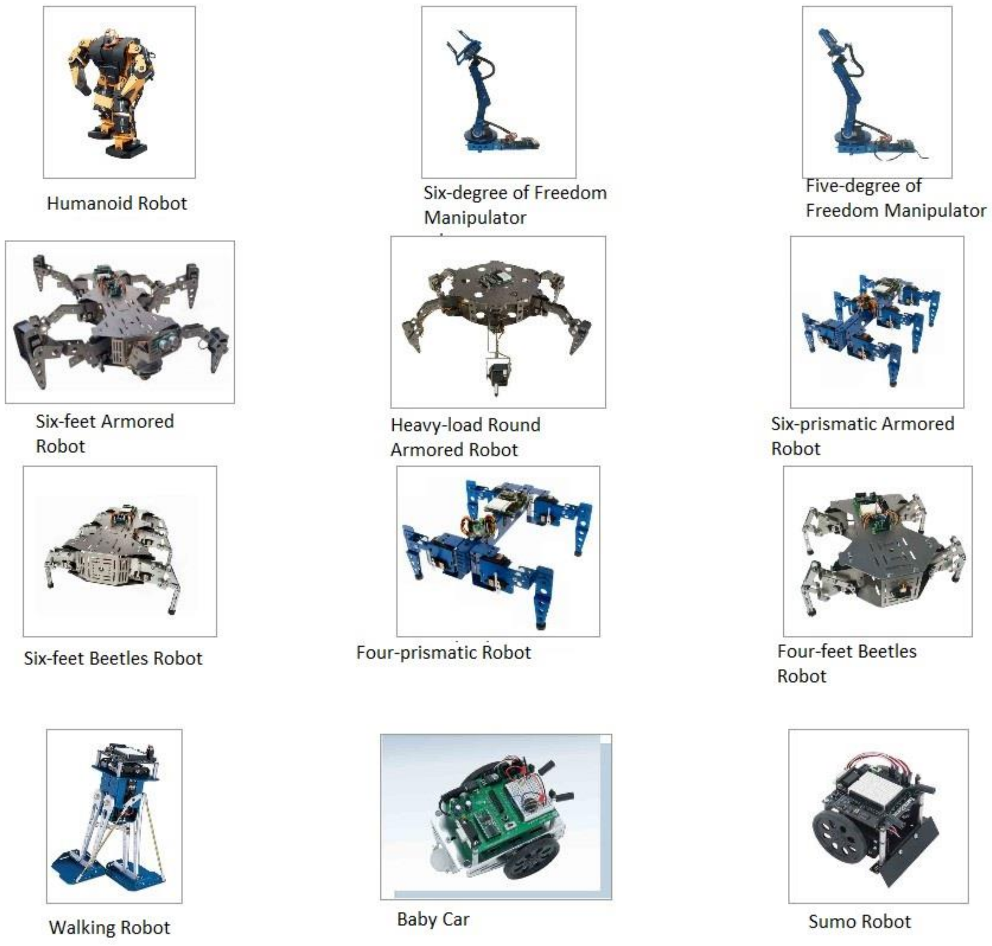


Figure 1: PRODUCTS MANUFACTURE BY DRROB [4]

The products of DRLab mainly serve in the laboratory such as the comprehensive experimental table, sensor test box, Multi-output power supply and control circuit experiment box. The DRLab mainly consists of fundamental instruments in the engineering educational field. The Figure 2 below shows some of the products of DRLab.

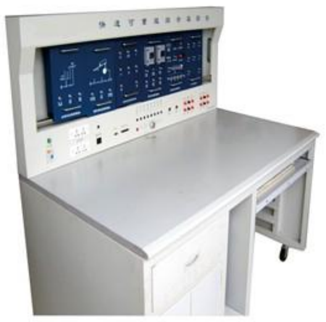
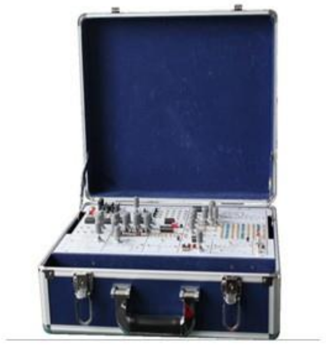
Sensor Test Box	 Comprehensive Experimental Table	 Control Experimental Box

Figure 2: PRODUCTS MANUFACTURED BY THE DRLAB

With the supplement of both experimental instruments and advanced educational ideas, DEPUSH has gained a good reputation among the high-tech enterprises. DEPUSH, as a company with the excellent qualifications, has a lead in advanced engineering education fields.

Under the revolution of robotics, we also join into this tide of the times seeking to upgrade the wheelchair as best as we can. DEPUSH wants to enable their products to be easily controlled remotely and expandable to different platforms. The robotic revolution brings the birth of several kinds of robots, wherein the educational robot occupies a crucial position nowadays.

Impact of Educational Robotics

The implications of the "Robotic Revolution", which is on its way, necessitates an in-depth study of the field of Robotics [1]. Since, learning has evolved from classrooms to distance learning and, now, interactive e-learning, Educational Robotics (ER) is the viable "next step" in the educational system. After virtual e-learning became an important part of the development of innovative educational practices, Robotics kits are now becoming commonplace in today's educational delivery structure [5, 6]. ER aims at such constructive learning uses and applications for integration into the education system.

This new type of educational system goes hand-in-hand with the different types of learning methods. Given that, Robotics has the potential to target each of these learning methods as:

1. An educational platform to help learn mechanical, electrical and computer science aspects of Robotics

A Robotic companion that can learn and/or teach a subject at “the same level” as the student

A Robotic teaching assistant that uses its interaction modules such as projection devices and verbal interactions to assist in teaching lessons to the students [5, 7].

For the purpose of our project, the first learning method mentioned above will be implemented by the Wi-Fi based wheelchair acting as the robotics education platform. Likewise, ER propagates meaningful and collaborative projects that students can reflect on, which compels students to explore the world of Robotics by identifying ideas and solving real-world problems [6]. The project at hand, sponsored by the DEPUSH Technology Co, Ltd. is trying to do just that by integrating a real-world problem of designing multimodal control interfaces for wheelchairs that can be easily controlled with a wide variety of Android smartphones.

Smartphones, especially in China, has seen an outburst in recent times, when China beat the United States as the world’s largest smartphone market, as per the report by Strategy Analytics [8]. According to the International Data Corp. (IDC), China accounted for 26.5% of all smartphone sales in 2012. The reason argues IDC, is the low-end and inexpensive models running Google Inc.’s Android Platform [8, 9]. Given that, China, thus holds a promising future for Android application development, fueling the project work on the Wi-Fi-based wheelchair, as a robotics education platform.

Furthermore, since, the awe-inspiring field of Robotics puts major engineering skills to use, it is beneficial for students to master the three engineering pillars of Mechanical Engineering, Electrical Engineering and Computer Science. Not to mention, the educational development of such a teleoperated wheelchair has a potential to serve a possible market pool for the disabled in the near future.

Thus, DEPUSH hopes to usher the era of Robotics as an educational platform for the wide array of colleges in China, especially with the growth of low-cost Android smartphone usage. The Wi-Fi based wheelchair is hence, a step toward achieving the mission of DEPUSH Ltd. to integrate innovative engineering solutions with the educational system in China. For this purpose, the new field of Human-Robot Interaction (HRI) can aid in the development of the Wi-Fi-based mobile wheelchair and its multimodal control interface.

Human-Robot Interaction

"The field of human–robot interaction (HRI) addresses the design, understanding, and evaluation of Robotic systems, which involve humans and robots interacting through communication" [10]. Similar to Human-Computer Interaction (HCI), HRI can help us understand that the Wi-Fi-based Wheelchair, as an educational platform requires a "man-machine" communication for users to understand the complexity of the multimodal control of the wheelchair. Moreover, the robot hardware needs to comply with the safety of the user using a safe robotic motion design [10-12]. A model of the Human-Robot Interaction at microsystems level can be found in the following Figure 3:

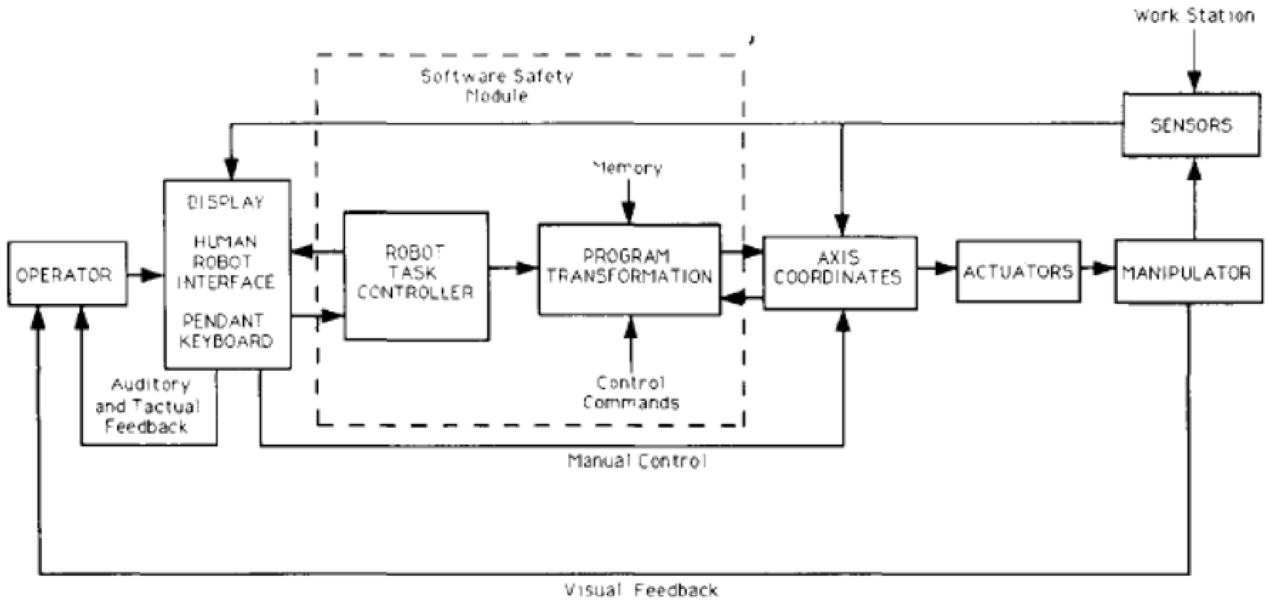


Figure 3: HRI MODEL[13]

For the wheelchair, the manipulator in the above diagram represents the wheels that are being actuated with the motors. However, the sensor loop back does not exist, due to lack of sensor usage and hence the current structure is open-loop (except for the visual feedback that the operator/user of the wheelchair receives from the motion of the wheelchair).

Given the different navigation control options established in the previous project work done on the wheelchair platform, the safety of the user and the ease of navigating the application itself, plays an important role in the optimization of the wheelchair. These factors need to be accommodated into both the physical hardware of the robot itself, and the software used to control the wheelchair.

"Robots must be able to adapt in real time their movements to the behavior of the humans" [12]. Thus, in the case of the wheelchair, a real-time reactional movement is necessitated when the user sends telecommands to the wheelchair platform. This reaction time is even more critical in emergency situations, such as potential crashes or falling off a cliff, where the wheelchair must come to a complete stop to avoid the situation. Since, the wheelchair makes use of telecommanding operations, a physical emergency switch may be necessitated as a hardware accommodation to achieve safe operations. This accommodation can also occur in the electrical and computer software facet of the communication protocol, which in the case of our project, is the Wi-Fi based wireless communication. In order to make it as fast as possible, the data exchange between the software modules needs to be relevant and concise to avoid the high overhead in processing time caused by junk data transfer. The importance of optimal response time is reaffirmed by Sidobre [12], where he emphasizes the significance of correct response in an acceptable time period.

Thus, the User Interface (UI), itself, that is provided for remote telecommanding of the wheelchair over Wi-Fi, also needs to be easily accessible for potentially handicapped users. So it follows that the Android application provides the platform to constitute a rich user experience that is not only efficiently designed, but, also easily expandable and maintained.

Android Application Development

Android applications are developed in the Java programming language environment to be able to run on Google Inc.'s Android platform that is based upon Linux [14]. Android provides API (Application Programming Interface) tools as a part of its SDK (Software Development Kit) in order for Android Developers to develop their own apps. These tools compile the Java code into an .apk file, an android package which is used by the android devices to install the app along with its resources.

Every android app requires an *AndroidManifest.xml* file, in which a set of properties for the particular app are defined. This file provides the basic information of your app to the Android system before the app itself can be run. There are four unique types of Android app components that act as building blocks for app development:

1. Activities: A screen with a UI (User Interface)
2. Services: Background component that can perform "long-running operations"
3. Content Providers: Shared data management component
4. Broadcast Receivers: Response system to "system-wide broadcast announcements" [15]

An Activity defines the user interface for interacting with the application. A given application may consist of multiple loosely bound activities, where one activity can start another. A set of reusable components are provided by the SDK, which in combination with the user-defined components can be glued together using *Intents*. Intents are objects used to communicate with the different app components. The communication can involve starting or stopping the app component (E.g. an Activity) with a given name [16]. These components and their intent filters are defined in the *AndroidManifest.xml* file along with the name of the android java package, application permissions, minimum level of Android API required by the application, amongst other properties of the specific application.

The major challenges with user interface design is the method of conveying all the information that the application provides. To do so, Clifton [17] argues that Android apps should start with a user goal in mind so as to design an intuitive application. This lets the user find the relevant information easily without any hassles. Using images and short text in dialogs helps convey meanings as quickly as possible. Animations are an integral part of the application, to convey to the user the real-time navigability of the app. An easy-to-use app ties in these visuals with a clear focus. The main features of the application should be easy to access and in case of user discrepancy, a fool proof way of *"undoing"* the changes is necessary [10].

Application Resources are the additional static content such as image files, layout, text strings, etc. that an application uses to enrich the user interface. These resources are organized in subdirectories in the Android Project's *res/* directory in the root. The name and type of resources are arranged in the following table:

Directory	Resource Type
`drawable/`	Bitmap files (`.png`, `.9.png`, `.jpg`, `.gif`) or XML files that are compiled into the following drawable resource subtypes: • Bitmap files • Nine-Patches (re-sizable bitmaps) • State lists • Shapes • Animation drawables • Other drawables See Drawable Resources.
`layout/`	XML files that define a user interface layout. See Layout Resource.
`menu/`	XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. See Menu Resource.
`raw/`	Arbitrary files to save in their raw form. To open these resources with a raw `InputStream`, call `Resources.openRawResource()` with the resource ID, which is `R.raw.`*`filename`*. However, if you need access to original file names and file hierarchy, you might consider saving some resources in the `assets/` directory (instead of `res/raw/`). Files in `assets/` are not given a resource ID, so you can read them only using `AssetManager`.

`values/`	XML files that contain simple values, such as strings, integers, and colors. Whereas XML resource files in other `res/` subdirectories define a single resource based on the XML filename, files in the `values/` directory describe multiple resources. For a file in this directory, each child of the `<resources>` element defines a single resource. For example, a `<string>` element creates an `R.string` resource and a `<color>` element creates an `R.color` resource. Because each resource is defined with its own XML element, you can name the file whatever you want and place different resource types in one file. However, for clarity, you might want to place unique resource types in different files. For example, here are some filename conventions for resources you can create in this directory: • arrays.xml for resource arrays (typed arrays). • colors.xml for color values • dimens.xml for dimension values. • strings.xml for string values. • styles.xml for styles. See String Resources, Style Resource, and More Resource Types.
`xml/`	Arbitrary XML files that can be read at runtime by calling `Resources.getXML()`. Various XML configuration files must be saved here, such as a searchable configuration.

Table 1: RESOURCE DIRECTORY STRUCTURE [18]

When the code is compiled a class for all the resources is formed, namely *R*. If a particular resource needs accessing, the subdirectory name mentioned in the table above is used as the resource type to access the elements inside it; for instance, R.drawable. [resource-name] would refer to the resource inside the drawable resource type. Similarly, Asset Manager is used to retrieve raw files, in the form of a stream of bytes, that the application comes bundled with. To access these raw files, AssetManager class is used to invoke the opening, reading and writing of the asset files. External libraries for third-party features can also be used in conjunction with the application files, to implement high level sub-functionalities in the application. This can be done using *.jar,* Java ARchives that act as an aggregation of previously written Java classes and other resources.

These third-party features can use the camera resources for image processing or GPS navigation for map views on the Android platform. Such features can then be integrated into the current Android application designed for the wheelchair to add on to its available features. However, it should be noted that the availability of wide coverage internet providers in China is deficient, making it harder for developing features for an app that requires an internet connection. Additionally, a lockdown on Google Inc.'s services, which is in place, hinders Android development in China. Finding a work-around to make use of the Google Inc.'s open source libraries for implementing such third party features is difficult, to say the least [19-21].

However, other third-party open source libraries have recently grown in numbers and are available for use by the general public for further development. Even though these upcoming software are open source for development purposes, the fact that they are very recent, implies that there is a lack of documentation. Moreover, the said software acquires a lot of memory or is computationally expensive, thereby adding to the non-feasibility of integrating such features. In any case, these libraries include, JavaCV, Nutiteq-3D, etc. and a further description about these topics are discussed in the sections below.

JavaCV

JavaCV is an open source Computer Vision library, which is an off-shoot of OpenCV that uses wrappers to provide an interface in Java for Android development [22]. This API provides an opportunity to perform high-level image processing, such as face/object detection, eye tracking, etc. The Wheelchair application can make use of the libraries provided in the JavaCV API to integrate the cameras on the Android device to add to the multimodal control of the Wheelchair. Path detection and eye tracking are two of the conceptual ideas for such multimodal integration into the current application that can be achieved using JavaCV.

Path Detection

For path detection, canny edge detection libraries provided by JavaCV can be used to filter the images taken by the cameras located on the Android device to isolate a clear path for the Wheelchair to follow. The algorithms provided by the libraries can perform the feature extractions to isolate the path, but, with high computational overheads [23, 24]. The resulting image with isolated clear paths can be seen in the following Figure 4:

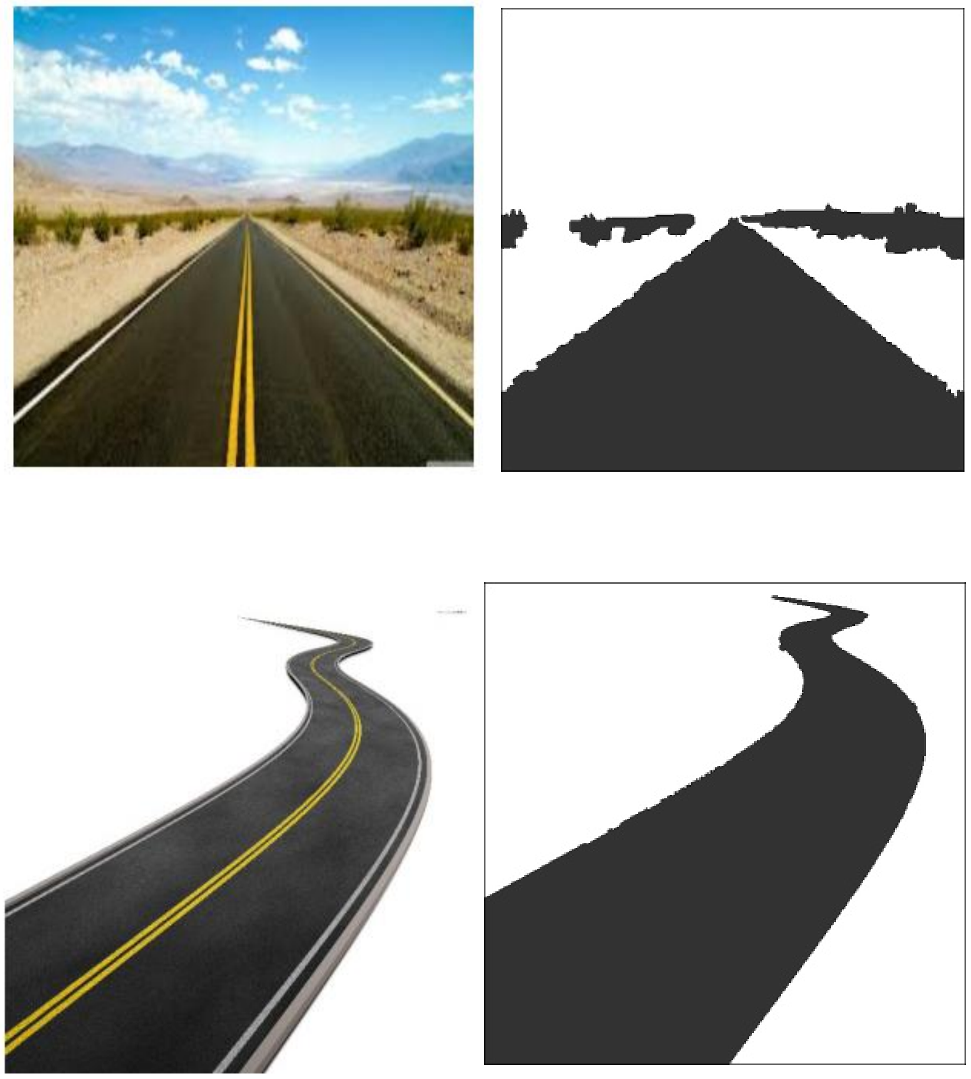


Figure 4: CLEAR PATH DETECTION RESULTS [24]

These results can then be manipulated by the Wheelchair application to make the wheelchair follow the path using telecommands for auto-navigation without user interruption. This can make the teleoperated wheelchair semi-autonomous and add to the multimodal control options.

Eye Tracking

Using the same OpenCV libraries, the user can use the front cameras to firstly, locate the face of the user and then detect the different parts of the eye, followed by isolating the pupils and following the gaze. It can make use of gradient algorithms to determine the center of the eyes [25, 26]. Following image displays the end result of the algorithm:

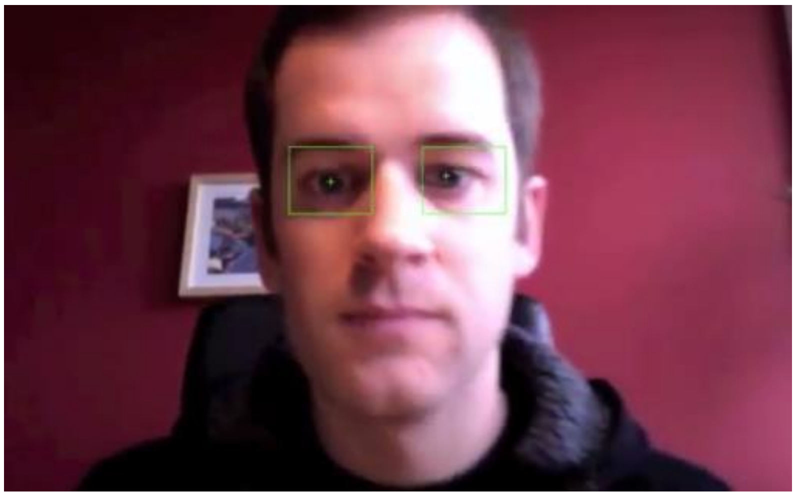


Figure 5: EYE BALL TRACKING [25]

The movement of the pupil can then be translated to the Wheelchair's movements such that when the eye gazes left, the wheelchair turns left and when the eye gazes right, the wheelchair turns right. A combination of blinks and gaze/eye tracking can be used to create a database of commands to control both the movements and the speed of the Wheelchair.

Overall, in order to implement both the eye tracking and path detection modules, third party APIs need to be used as the external .jar files in the application, thereby causing high memory and computational overheads for the application development.

GPS Navigation

As was discussed earlier, the need for offline computation is critical in this project and hence, map views and routing needs to occur offline. Also, Google Inc.'s map API cannot be used for offline map navigation, not only because Google Inc. is blocked in China, but, also because saving the Google Inc.'s map tiles offline is illegal. Hence, OpenStreetMap (OSM) can come to the rescue as it is open source and freely available to the general public. Additionally, the GraphHopper (GH) Route Planner, also open source, can be used to converts the open source map of China to be routable. It also acts as a road routing engine to navigate between any two points on the given map [27]. The following route has been generate by the graphhopper routing engine:

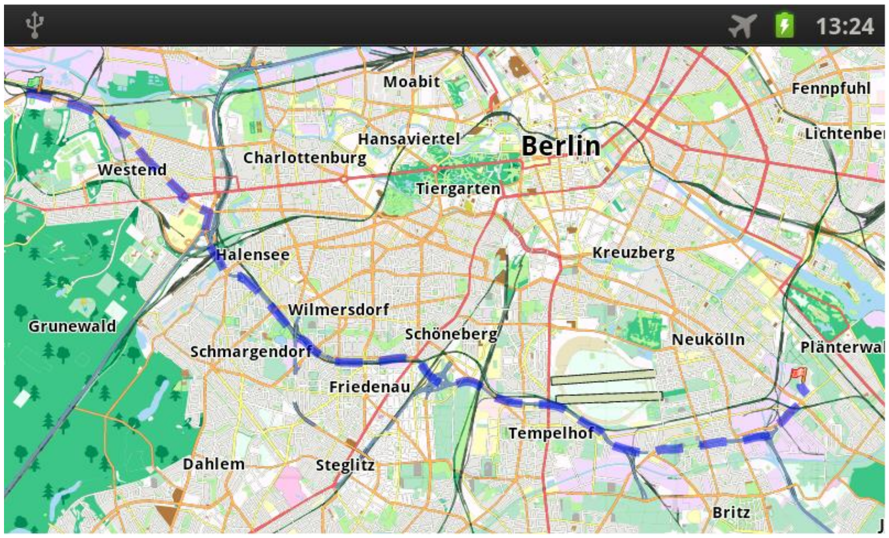


Figure 6: GRAPHHOPPER ROUTING ENGINE IN ACTION [27]

Once a route is found using the engine, the navigation instructions between the two points can then be translated to the wheelchair movements. The wheelchair platform can ultimately become totally autonomous with the help of an obstacle detection module and navigate using GPS from one point to another without human intervention.

Therefore, eye tracking, path detection and offline routing can together add to the multimodal control interface of the wheelchair. They can also be combined to take a step further in the autonomy of the Wheelchair. For instance, path detection and offline map navigation can together perform obstacle avoidance and auto navigation to the destination without human interference, thus, adding on to the autonomy of the Wheelchair.

In order to interface between the Android application and the wheelchair itself, there is a need for a wireless communication protocol to be established between the two modules.

Subsequently, we look at the realization of wireless communication between the hardware of the wheelchair and the Android software application.

Wireless Communication:

In order to reach the goal of expandable and robust communication between wheelchair and the Android smart phone, with a background study in wireless communications, several options for a new wireless communication was explored to solve the connection problem.

Bluetooth:

“Bluetooth is a wireless technology standard for exchanging data over short distances from fixed and mobile devices, and building personal area networks (PANs)” [28]. The only limitation of Bluetooth is that it is slower in transferring information than the Wi-Fi connection.

Wi-Fi:

Wi-Fi is “a local area wireless technology that allows an electronic device to exchange data or connect to the internet” [29]. It is usually taken as synonym for “WLAN”. We finally choose this option since the new Wi-Fi module, the default network type of ST-MW-09S is Software enabled Access Point (SoftAP) which provides both the security and efficiency of the wireless communication. Also, the new users do not need to change any settings in their cell phones. SoftAP can be understood as a "virtual router" which is the wireless network being set by the laptop or some mobile devices. “The advantage of SoftAP is the use of a regular cellphone, for example, with a client antenna and data connection as an Access Point to serve other wireless devices which do not have a data connection otherwise” [30].

Router:

“A router is a networking device, commonly specialized hardware that forwards data packets between computer networks” [7]. The router cannot be used in the wheelchair unless its hardware is modified to include serial ports to connect to the micro controller. And then implanting a OpenWrt system into the router to program it [31].

TCP/IP:

“TCP is the short form for Transmission Control Protocol. TCP is one of the core protocols of the Internet protocol suite(IP)”, and together they are called TCP/IP [32]. TCP is responsible for transmitting large chunks of data as packets until all the data has been sent to the target receiver and if any problems occur, TCP carries out re-sending the packets. The diagram in Figure 7 below shows how TCP transmits the data.

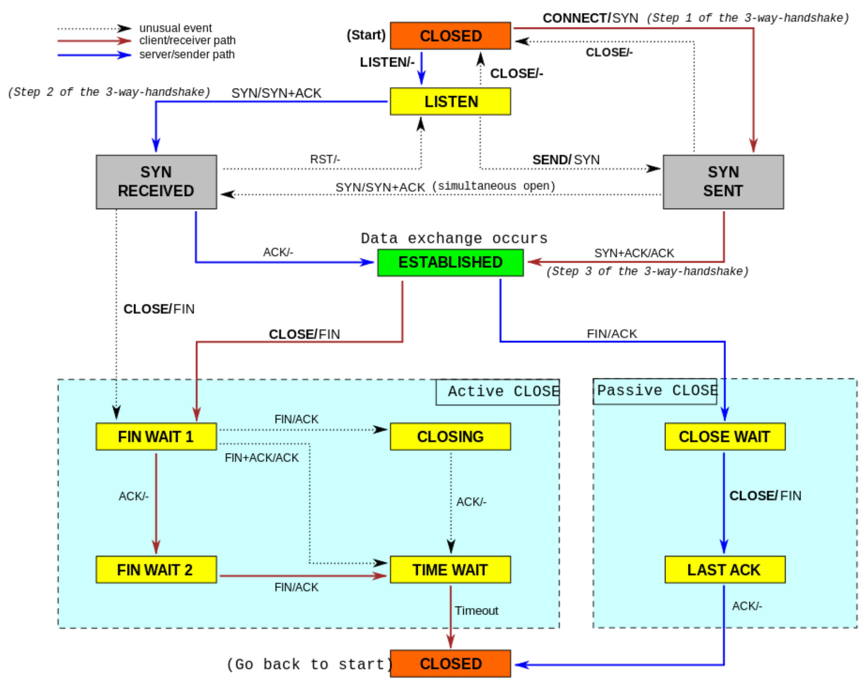


Figure 7: TCP SIGNAL TRANSMISSION [32]

Next, we look at the past project work done on the Wheelchair platform for telecommanding the Wheelchair using the different control options developed by the HUST students over the past year.

Previous Project Work

As we know, this project revolves around the Wi-Fi-based Electric Power Wheelchair, as an educational platform for DEPUSH Technology Co, Ltd. It was first developed by HUST students in a series of two projects over 2014. As seen in Figure 8, a normal electrically powered wheelchair: Model Wisking 1015, was transformed for this purpose.

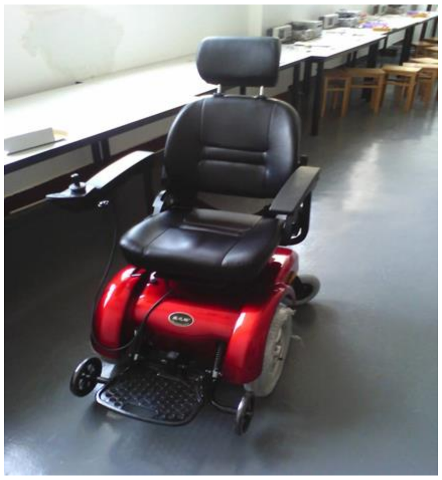
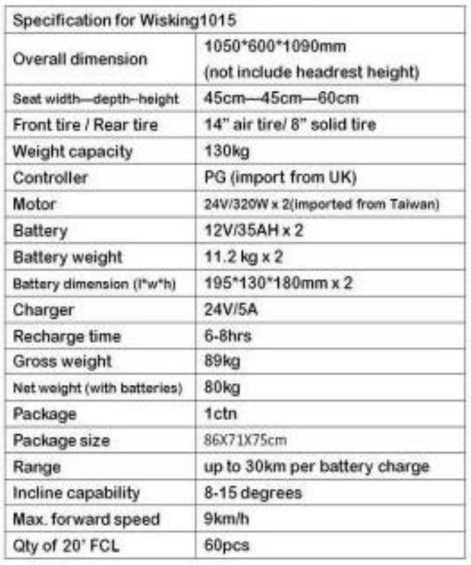


Figure 8: WISKING 1015 POWER WHEELCHAIR [31]

The Wisking 1015 shown above went through two iterations before we were given the responsibility to optimize it further. The original electrically powered wheelchair was driven by a P&G motor controller that was connected to two 24V DC motors. Each motor was powered up by two 12V batteries and drove one wheel each. The user could originally control the wheelchair by using a joystick for navigating the wheelchair. Some of the technical specifications of the original wheelchair can be seen in Figure 9:



Specification for Wisking1015	
Overall dimension	1050*600*1090mm (not include headrest height)
Seat width—depth--height	45cm—45cm—60cm
Front tire / Rear tire	14" air tire/ 8" solid tire
Weight capacity	130kg
Controller	PG (import from UK)
Motor	24V/320W x 2(imported from Taiwan)
Battery	12V/35AH x 2
Battery weight	11.2 kg x 2
Battery dimension (l*w*h)	195*130*180mm x 2
Charger	24V/5A
Recharge time	6-8hrs
Gross weight	89kg
Net weight (with batteries)	80kg
Package	1ctn
Package size	86X71X75cm
Range	up to 30km per battery charge
Incline capability	8-15 degrees
Max. forward speed	9km/h
Qty of 20' FCL	60pcs

Figure 9: TECHNICAL SPECIFICATIONS OF THE ELECTRIC WHEELCHAIR [33]

This framework was then intercepted to include the wireless communication and control module by replacing the joystick control. An open-ended Android Application was then developed to make use of the various cell phone features in order to distribute the workload of the wheelchair itself and transferring control over to the Android phone. Several navigation methods were implemented by the application to achieve control over the movement of the wheelchair. The main focus of the past project iterations was multimodal remote teleoperation/telecommanding of the wheelchair. Both hardware and software changes were made to achieve a multimodal control of the wheelchair and possibly more robots in the future with the same Android application.

Hardware Modifications

We know that the original architecture of the powered wheelchair revolved around the joystick for controlling the movements of the wheelchair as can be seen in the following Figure 10:

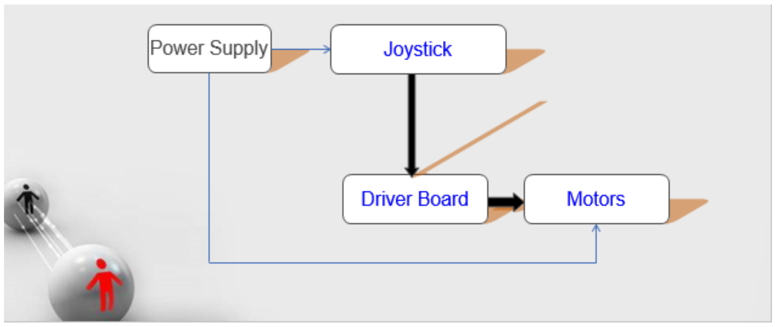


Figure 10: ORIGINAL ARCHITECTURE OF THE POWERED WHEELCHAIR

Here, the driver board in the above figure refers to the P&G motor controller that was supplied with the original wheelchair model. This architecture was then expanded by the previous groups to include the Wi-Fi-based control of the wheelchair with an Android smartphone.

For this purpose, a wireless module was added to the architecture by the previous groups to establish a Wi-Fi connection with the smartphone. So, the framework was intercepted to substitute the joystick control as it was no longer required. This can be seen in the following Figure 11:

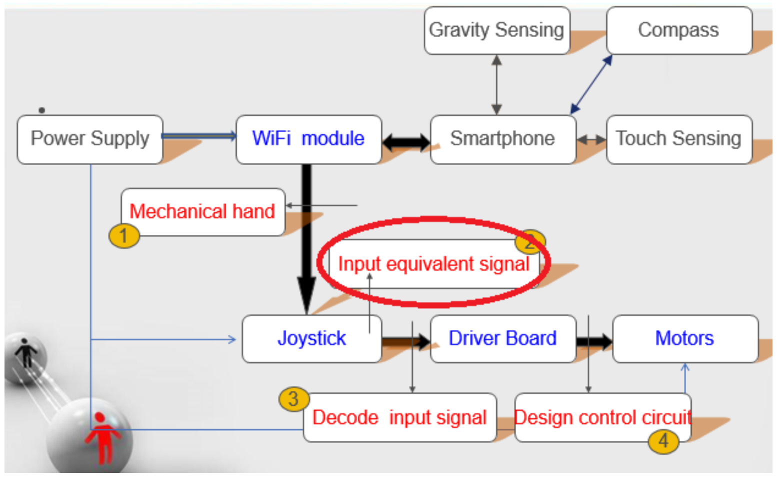


Figure 11: MODIFIED ARCHITECTURE OF THE POWERED WHEELCHAIR [31]

As seen in the figure above, this interception of the joystick control within the system architecture had choices varying from designing a mechanical hand, inputting an equivalent signal, decoding the input signal and/or designing a new control circuit; of which, inputting an equivalent signal to simulate the joystick control was selected by the previous groups. To input the equivalent signal, a relay circuit with optocouplers was used as seen in the following Figure 12 to simulate the joystick signals. These joystick signals were used to replicate the movements of the wheelchair controlled from the joystick.

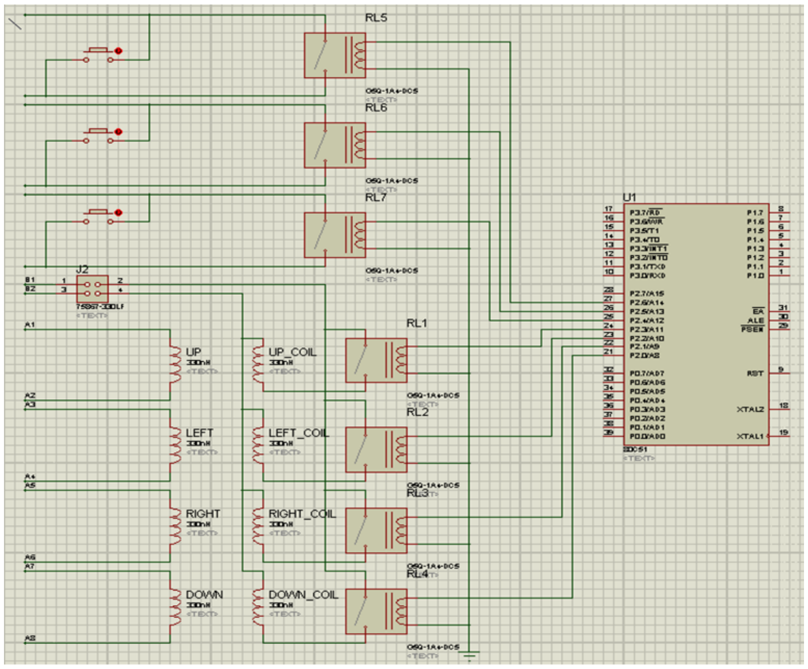


Figure 12: ELECTRIC RELAY CIRCUIT DIAGRAM [31]

The wireless communication between the Android phone and the wheelchair was realized by the previous groups using the ST MW 08S wireless module. This module was a serial communication module that created a wireless ad-hoc network. It was already packaged with an attached microcontroller, namely, the STC89S52 single chip seen in the following circuit diagram:

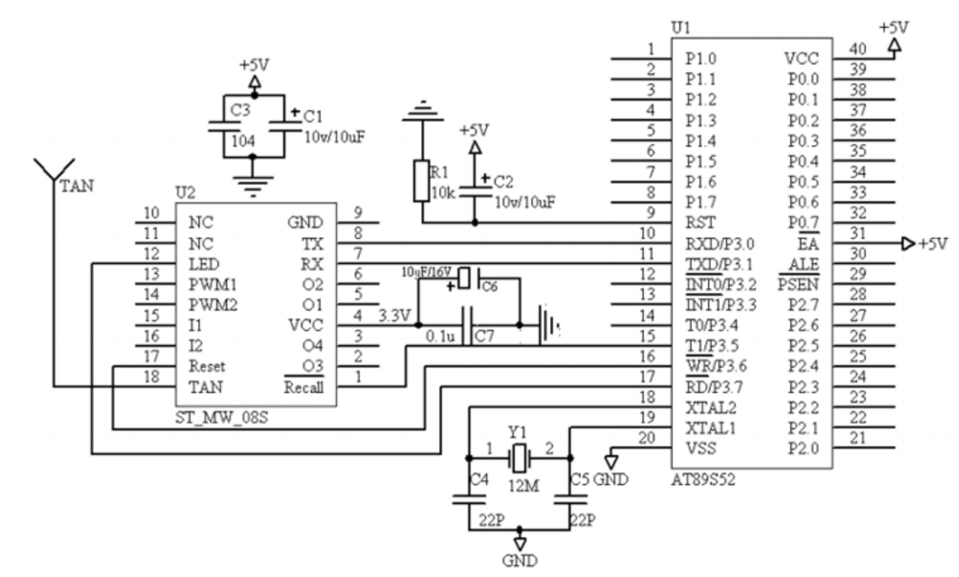


Figure 13: WI-FI MODULE CIRCUIT DIAGRAM [31]

Given that, the overall motherboard connections used by the previous groups can be noted in Figure 14 as follows:

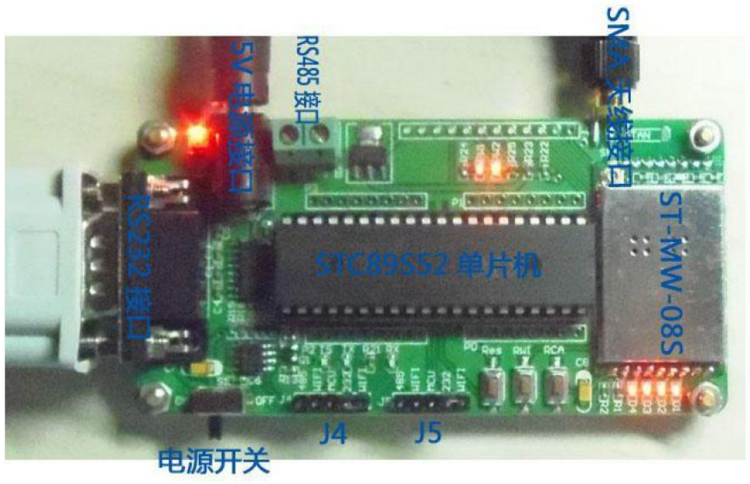


Figure 14: MOTHERBOARD USED BY PREVIOUS GROUPS [31]

The same single chip was used by the relay circuit such that the single chip was powered from the same battery source as the motors, i.e., 24V but reduced to 5V, once again using the electric relays, for the logic circuit. However, the different speed levels of the original wheelchair was not reached by this circuit as it lacked a bidirectional dual motor controller to handle the 24V, 2.5A dual DC motor requirements. Hence, the motors ran at full speed at all times and lacked a smooth deceleration when coming to a stop. Overall, an integration of a speed regulation system was necessitated to ensure the safe operation of the wheelchair.

Software Development

To realize the communication between the hardware and the software, an ad-hoc wireless connection was produced by the ST-MW-08S wireless module to connect to Android phones, and pass the commands to the microcontroller. Also, an Android application was developed in the Java programming language for user interface.

Wireless Communication

By default, Android phones do not natively support connections to ad-hoc networks unless the internal settings of the phone are changed. This requires rooting the device, however different devices require different changes to the settings for enabling ad-hoc connections, both of which are inconvenient in the long run.

In the old wheelchair, users have to do the Android rooting first in their smart phones and then install an application named ad-hoc Switcher in order to connect to the wheelchair’s Wi-Fi. "The Android rooting is the process of allowing users of smart phones and other devices running the Android mobile operating system to attain privileged control within Android’s sub-system" [34]. Rooting is for wheelchair users to be able to run the specialized application which is the ad-hoc Switcher that requires administrator-level permissions, otherwise inaccessible to a normal Android user.

This causes the limitations in expanding the application to more devices. "The ad-hoc Switcher is a patch for Android phones to detect the ad-hoc network since the original Android system does not support the wireless ad hoc network" [35].

"Wireless ad-hoc network is a decentralized type of wireless network and it often refers to a mode of operation of IEEE 802.11 wireless networks" [36]. And "IEEE 802.11 is a set of media access control and physical layer specifications for implementing WLAN computer communication in different frequency bands" [37]. It is usually assumed that the IEEE 802.11 and Wi-Fi are the same thing, but the difference between them is that IEEE 802.11 is the standard of WLAN and Wi-Fi is actually a brand of the Wi-Fi alliance. The standard and amendments provide the basis for wireless network products using the Wi-Fi brand.

The previous Wi-Fi module required that the Android phones install an application named ad-hoc switcher, but the installation process caused tremendous troubles for the new user, to avoid the unnecessary work, we decide to switch to a better option.

Android Application

Some of the smartphone's features such as touch sensing, gravity sensing, microphone and GPS (Global Positioning System) services were used to develop the Android application that can control the basic movements of the wheelchair using these various sensors.

The GUI (Graphical User Interface) developed by the past groups can be seen in the following Figure 15:

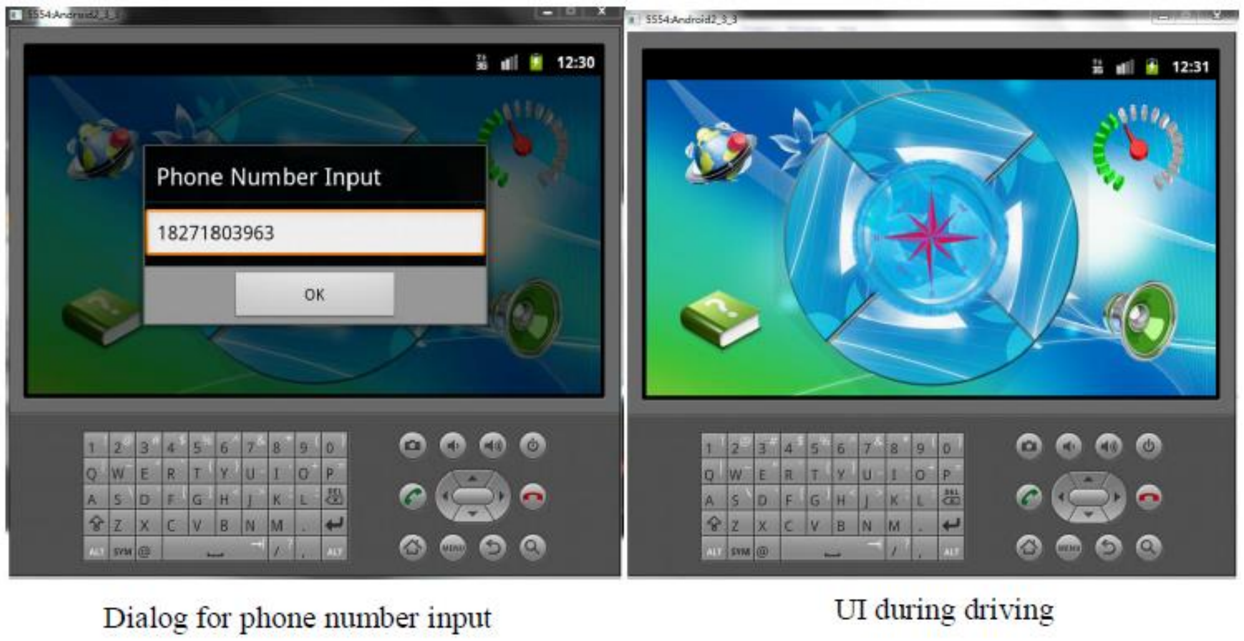


Figure 15: PREVIOUS GRAPHICAL USER INTERFACE (GUI) [31]

The interface included the movement commands for moving forward, backward, left and right. The acceleration and deceleration options were a part of the menu as was seen in the figure above. Even though the wheelchair did not reach variable speeds, a basic skeletal structure for integrating these in the future was set up by the previous groups. The speedometer was located in the upper right corner. A compass was also located in the center to determine the direction of movement at all times. The resulting multimodal control options for the wheelchair were as shown in the following table:

Variety of Multimodal Control Options
Touch Sensing
Gravity Sensing
GPS
Gesture Control
Speech Recognition

Table 2: LIST OF MULTIMODAL CONTROL OPTIONS

Touch sensing, on one hand, used the cellphone's touch screen feature to detect the command being pressed, which was then received by the Wi-Fi module and executed by the microcontroller. Gravity sensing, on the other hand, made use of the accelerometer sensors on board the Android cellphone to detect and translate the tilting of the phone to the correct command execution in order to move the Wheelchair. It also consisted of a graph being constantly animated to display the angle/time, command/time and distance/time of the cellphone as can be seen in Figure 16 below:

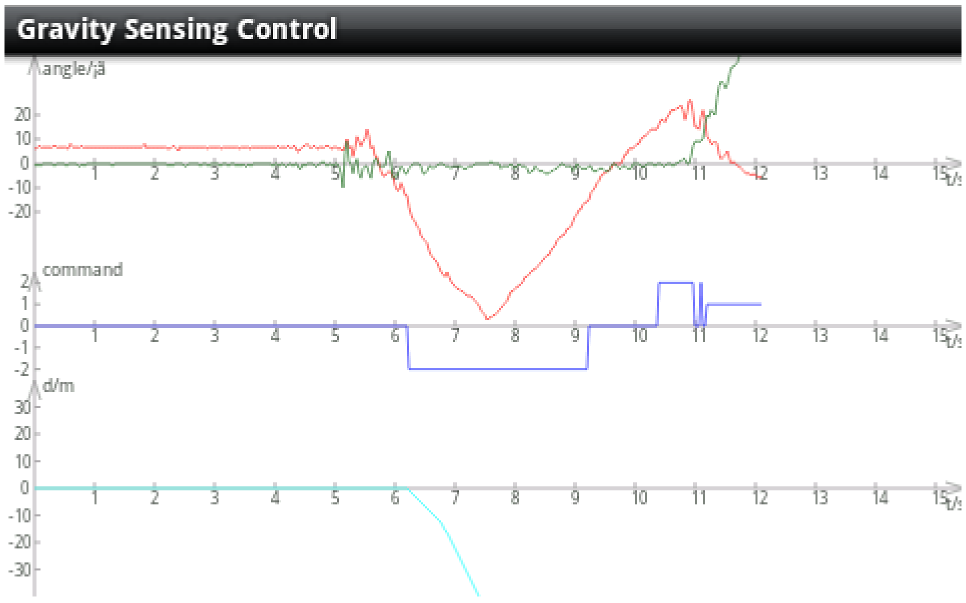


Figure 16: GRAVITY SENSING UI

The animation for the graph was not only computationally expensive but, also constantly crashed the application. In addition, these graphs were only approximations, as the accelerometer sensors cannot provide an exact distance and hence, is not reliable for such purposes.

The application also made use of Baidu maps for online routing and displaying the current location of the Android cellphone on an online Baidu map by reverse geocoding. This was done by obtaining the current location of the cellphone through the GPS on the phone, which was then translated to an address on the map. This functionality can be observed in the Figure 17 below:

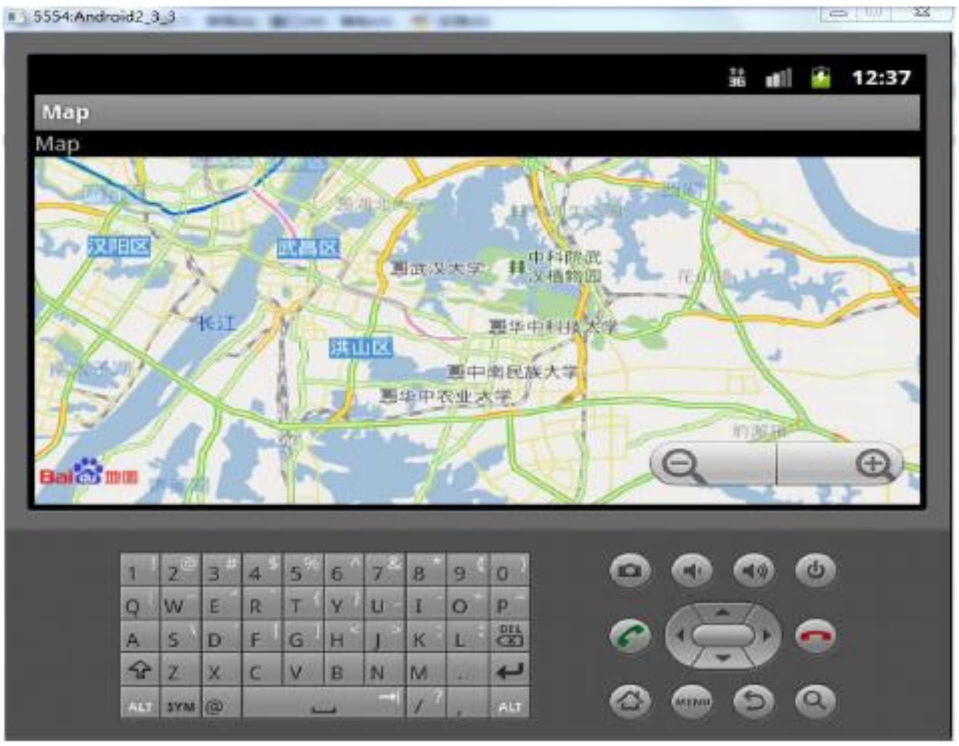


Figure 17: GPS WORKING ON PREVIOUS UI [31]

However, it must be noted that this feature lacked GPS-based navigation since users could not route between any two points of their choice. And since it was online, the user could no longer use the wheelchair's other control options until the Wi-Fi connectivity is switched back to use the Wi-Fi module used to connect to the wheelchair platform once again.

Moreover, the Gesture control simply made use of the swiping motion of the finger over the screen to detect the direction commands and executed the same. Likewise, the offline speech recognition module used PocketSphinx by Carnegie Melon University (CMU) to perform the speech to text algorithms [38]. The offline package in the app simply converted the speech to text and telecommanded the Wheelchair to follow that movement command. The UI that goes with this control method can be seen in Figure 18

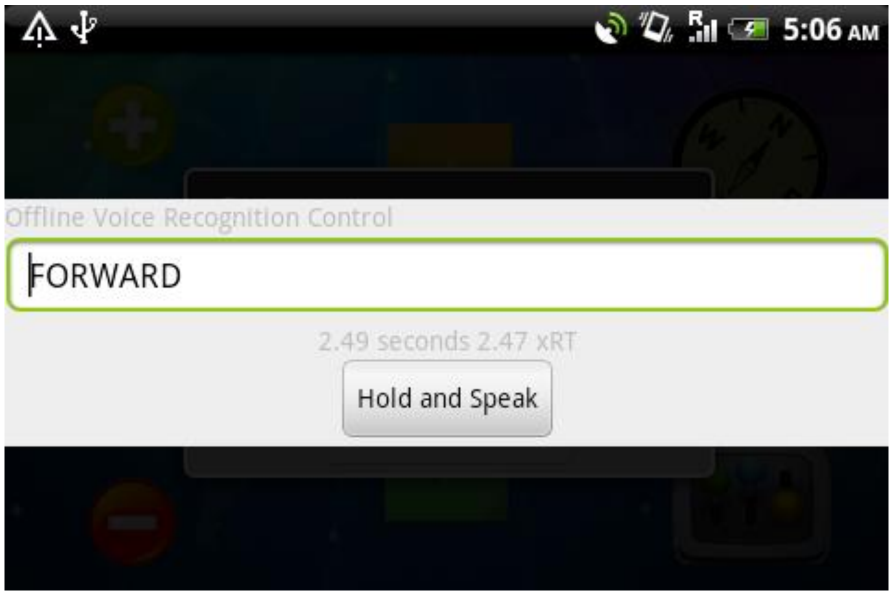


Figure 18: SPEECH RECOGNITION UI

The commands available inside the offline database of the voice recognition were as follows:

Offline Voice Recognition Database
Forward
Backward
Left
Right
Stop
Speed up
Speed down

Table 3: PREVIOUS OFFLINE VOICE RECOGNITION DATABASE

However, in order to recognize the above commands, certain voice packages had to be manually downloaded onto the device separately before it could be used in the application. Thus, the offline computation had its own trade-off and required to be automated.

The Wheelchair app also included an obstacle detection module which would send SMS (Short-Service Messages) in case of emergency, with the use of gravity sensing as a threshold.

So, when the phone is shaken, a dialog box appears to ask the user for confirmation to send the SMS. This SMS includes the location of the phone on the map that was extracted using the GPS. This threshold however, was very unreliable as every time the phone is shaken, even by mistake, the user is prompted to confirm if it is an emergency or not.

Summary

As an educational platform, the Wi-Fi based Wheelchair was to be teleoperated by users using hands-on multimodal control options. For the purpose of sticking to an Educational Robotics (ER) platform for the DEPUSH Technology Co, Ltd. expandability of the application was emphasized and major drawbacks of the previous wheelchair architecture were targeted. These included, the lack of a speed regulation system in the hardware as well as poor Wi-Fi connection and insufficient use of the Android device resources in the software. Thus, the study of Human-Robot Interaction (HRI) formed the basis for ensuring the safety of the hardware and optimization of the software associated with the wheelchair. Lastly, the android development study was used to leverage the safe telecommanding of the wheelchair using the multimodal control options with an efficient and user-friendly interface.

Chapter 3: Methodology

This chapter, details the problems laid out in the previous wheelchair platform, the goal formulated to tackle the problems and the methods we used to achieve our goal.

Problem Statement: The wheelchair architecture was non-expansible, lacked a speed regulation system, and exhibited poor Wi-Fi connection to Android phones and insufficient utilization of the Android resources on board.

In order to tackle the problems stated above, we formulated our goal as follows:

Goal: The goal of our project was to establish expansible control with enhanced utilization of Android cell phone resources and an improved wireless communication, to control the wheelchair platform with an added speed regulation system.

To achieve this goal, we developed the following objectives:

1. Analyze and modify the wheelchair's previous architecture
2. Upgrade the Android application GUI, automate offline voice control and integrate GPS navigation
3. Enhance the wireless communication to Android cellphones
4. Add the speed regulation system in the hardware circuit of the wheelchair
5. Simulate the developed features on a smaller platform
6. Design and manufacture a prototype of the improved wheelchair platform

Each of the above objectives will be detailed in the coming sections.

Objective 1: Analyzing and modifying the previous system architecture to determine sources of the problems and potential for improvement

The wheelchair's current architecture was analyzed to determine a set of problems or issues with the current wheelchair platform. For this purpose, we paid a visit to the sponsor to get a better understanding of the purpose of the Wi-Fi based Wheelchair project and how it adheres to the company mission. Following is the logo in the new location of the Depush Technology Co. Ltd.:

Figure 19: DEPUSH COMPANY LOGO

We also interviewed the previous groups to understand the nature of their work and obtain their opinions on the problems with the current platform and how they arose.

We found various issues with the current platform from the background study and interviews with the previous groups regarding the Wheelchair architecture. The following Table 4 lists these issues and their reasons based on the type of problem: hardware/software.

Issues/Problems	Type	Reason
Power drops in the circuit	**Hardware**	Same battery source for the motors and logic of the microcontroller
Wires cannot withstand high current and smoke all the time	**Hardware**	Usage of thin wires causing overheat
Lack of a speed regulation system	**Hardware**	Relay circuit already too complicated to add speed regulation system
Ad-hoc network unable to be detected by most Android phones	**Hardware**	Android phones do not support ad-hoc network connectivity by default
Manual installation for offline voice recognition	**Software**	Inability to download them automatically
Lack of offline GPS navigation	**Software**	Offline Baidu maps usage by developers is no longer permitted according to terms of service/usage
Gravity Sensing displays an approximation of distance	**Hardware/Software**	Lack of encoders/closed loop control for calculating position of the Wheelchair
Cameras not yet used	**Software**	No possible usage found
Emergency SMS threshold set to shaking the phone	**Hardware**	There are no other peripherals attached to the Wheelchair to detect obstacles
Non-intuitive application	**Software**	Scattered features in UI that is inconvenient for new users

Lack of user safety for emergency situations	**Hardware**	No circuit breaker switch or contactors for circuit protection

Table 4: OUTLINE OF ISSUES WITH PREVIOUS WHEELCHAIR PLATFORM

From the above table, it can be observed that the software components were either inaccessible, too tedious, inconvenient for the user or did not sufficiently use the resources on board the Android cell phones. As far as the hardware was concerned, the previous architecture of the Wheelchair lacked the capability for expanding the circuitry required to add more features to the Wheelchair. For instance, the relay circuit used to "input equivalent signals" in the previous architecture, was already too complicated to add more relays for speed regulation.

Therefore, we analyzed the overall architecture of the wheelchair platform in order to determine which component of the following architectural diagram, shown in Figure 20, needed to be targeted and/or modified to achieve our goal.

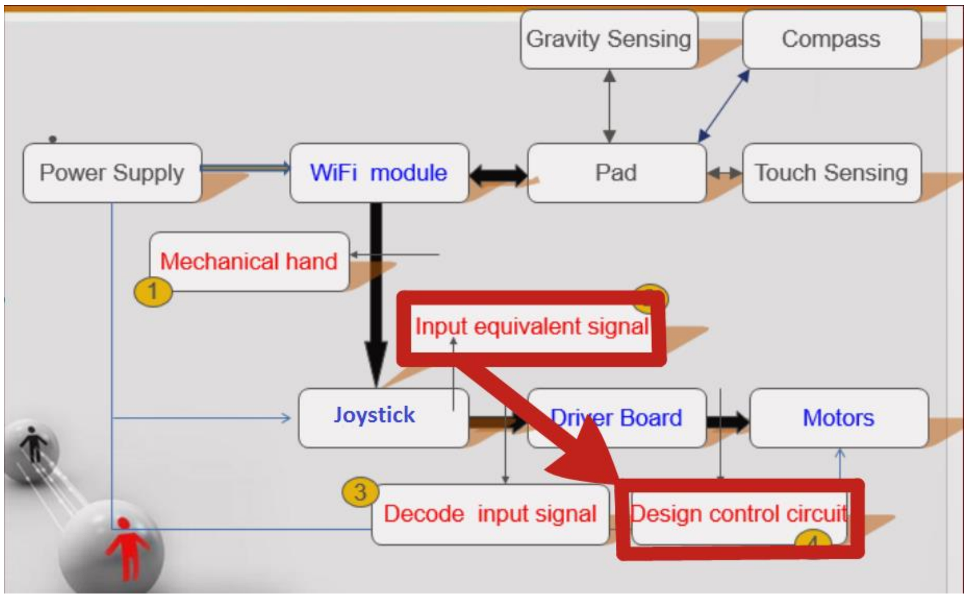


Figure 20: MODIFYING PREVIOUS ARCHITECTURE

Subsequently, we modified the above architecture by migrating from inputting an equivalent signal from the joystick, previously used in the past project work, to designing a new control circuit with a motor controller. Accordingly, the relay circuit for simulating the joystick signals was replaced with a circuit that used the motor controller to directly control both the speed and direction of the two DC motors. Likewise, the Wi-Fi module part of the architecture was also revised so as to connect to Android cellphones without any hassles in the form of manual installations of third-party software or internal changes to the settings of the phone. The application itself that was on the smartphone was also revised to be more intuitive in nature and use more features of the Android device.

In order to design the new control circuit and revise the Wi-Fi module component, certain pros and cons of the motor controller/microcontroller combination and wireless communication options were mapped out. Some of the factors that were used to base this comparison included feasibility in terms of cost, expansibility, computational overheads and safety.

Thereafter, the revised components of the architecture were combined to meet our goal in a "House of Quality" (HOQ) diagram. This diagram defines the relationship between the customer/sponsor needs and the product capabilities formed from the components of our Wheelchair architecture in a matrix form [39]. The following sections describe the methods used to finally replace and select the Wi-Fi module and motor controller components, respectively. These modifications to the architecture are followed by an explanation of a "divide and conquer" approach to the new architecture that was constructed thereafter.

Wi-Fi Module Replacement

The wireless communication protocol was revised by drawing certain parallels between the different available wireless communication options for controlling the Wheelchair with an Android device. The two communication options that we were faced with were: Wi-Fi and Bluetooth communication. The pros and cons for both the communication protocols were compiled to select the replacement of the Wi-Fi module by considering the above-mentioned factors.

From the analysis, it turned out that the Wireless communication was more advantageous for our project, and hence we replaced the current Wi-Fi module instead of switching to a Bluetooth module. The replacement was done based on the Android device connectivity, which was transformed from an ad hoc network to a SoftAP network. For this purpose, the Wi-Fi module, namely, ST-MW-09S was selected, as an upgraded version of ST-MW-08S, which was previously used in this project. This module can be observed in the following Figure 21:

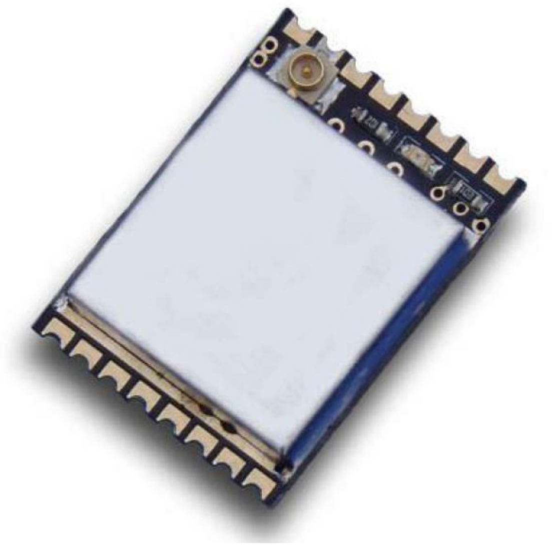


Figure 21: ST-MW-09S WI-FI MODULE

Also, an important feature to note is that, the Wi-Fi module was already equipped with a microcontroller, which was the same as before, namely, STC89C52. Further details regarding the new Wi-Fi module and its networking properties can be observed in Objective 2.

Motor Controller Selection

The choice of the motor controller was such that it needed to handle the voltage required for the DC motors, that is, 24V and the current drawn by the motors with load, which could possibly exceed 30A. Hence, a suitable motor controller had to be selected that would be safe to use at higher currents.

For this purpose, we contacted the original manufacturers of the Wheelchair motors, namely, Motion Technology Electric and Machinery Co., Ltd. We enquired for information regarding the DC motors themselves as well as the motor controller previously used for the original electrically powered wheelchair. The original DC motors manufactured by Motion Tech. can be seen in Figure 22 below:

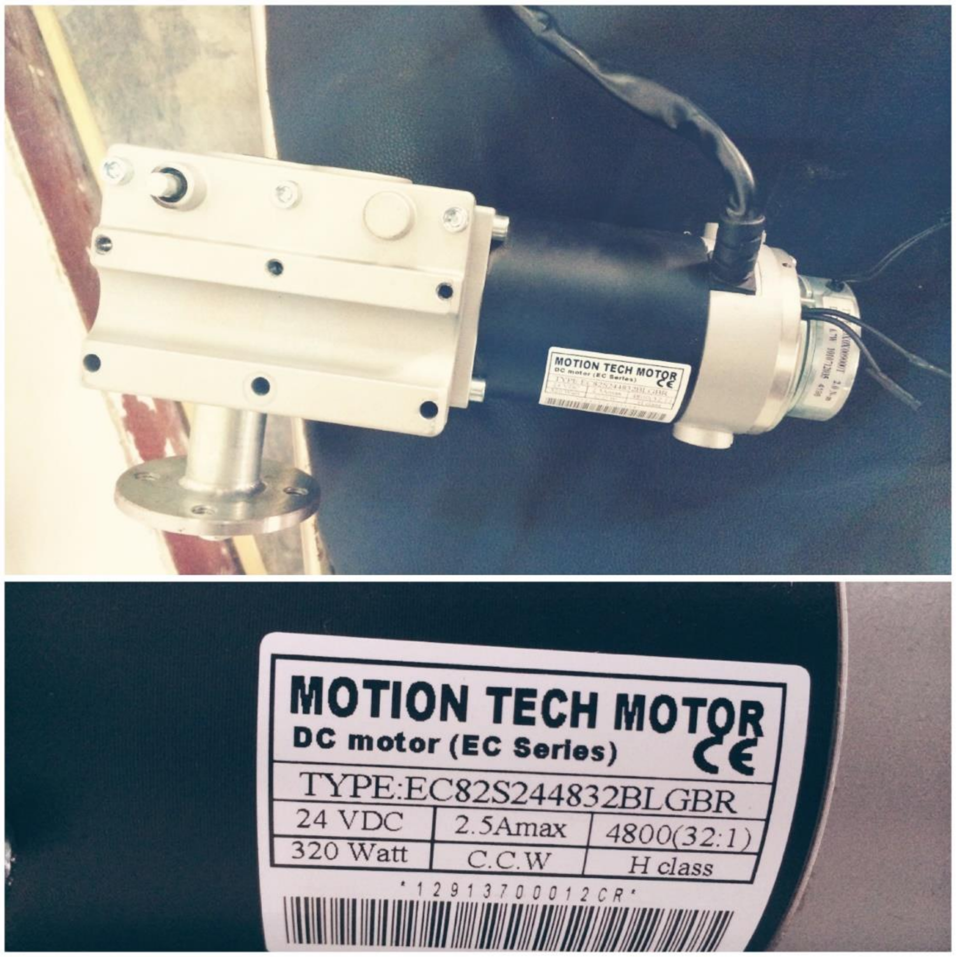


Figure 22: 24V MOTION TECH DC MOTORS

According to the datasheets received from the manufacturers, the DC motors were already equipped with a gear box, with a 1:32 gear ratio, a lever to relieve the motors for physical self-propelling of the wheelchair and a braking system. The braking system of the motors turned out to be irrelevant to our project as we decided to consolidate the total control of the wheelchair to the motor controller. This control included, smooth acceleration, deceleration and stopping of the wheelchair. Thus, we removed the brakes from both the motors and sealed the live wires emanating from the removal of the brakes. Instead, we used the software we designed to control the speed and stopping mechanism electrically through the motor controller.

After comparing the pros and cons of several motor controllers that were analyzed, we finally selected RoboClaw 2x60A dual motor controller that can handle the speed and bidirectional control of both the motors. Following is the image of the RoboClaw motor controller that was used for this purpose:

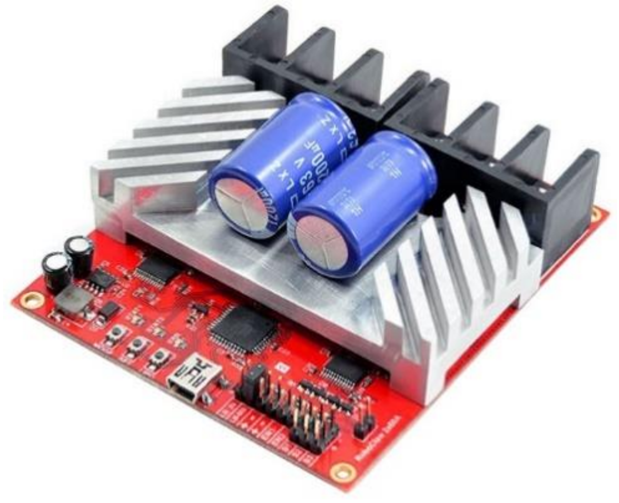


Figure 23: ROBOCLAW MOTOR CONTROLLER

The motor controller also had a built-in Encoder facility that could further be used for a closed-loop control of the Wheelchair. Encoders are used to count the number of revolutions of the motor spindle, which can then be used to calculate the number of revolutions of the wheel by using its diameter, which ultimately provides us with the distance traveled by the Wheelchair. Using encoders would enable us to know the position and thereby, the speed of the Wheelchair at any given time. However, we could not use the included encoder facility attached to the RoboClaw, since the motors we had could not be modified to physically add encoders. So, we contacted the motor manufacturing company, only to learn that such motors could be manufactured, but, would take quite some time for both manufacturing and shipping. So, we decided not to use the excellent encoder facility available on board, however, future application using RoboClaw can definitely implement it, if given enough time for shipment of the motors with encoders.

Further details about the specifications of the motor controller can be found in "Appendix A: RoboClaw Specifications". Objective 4 examines the use of the motor controller to add a speed regulation system in the wheelchair platform.

Overall Architectural Modification:

Comprehensively, the new architecture comprised of a motor controller circuit designed to directly control the DC motors instead of using relay circuits to simulate the joystick signals, which the previous architecture made use of. Furthermore, the Wi-Fi module replacement ensured that the Android cellphones could now connect seamlessly to the Wi-Fi network. The architecture was thus, modified to use revised components that includes a microcontroller, namely, STC89C52 connected to the Wi-Fi Module, ST-MW-09S and a RoboClaw 2x60A motor controller. The capabilities of these components were also drawn into a House of Quality structure to prioritize the methods to meet both our goal and the sponsor's expectations.

Thus, the above mentioned modifications to the Wheelchair architecture were put into place, which could further be divided into four segments/modules based on the communication between the different components of the architecture:

1. Android Device to the Multi-Modal Control Application
2. Wi-Fi Module to Android device
3. Wi-Fi Module to Motor Controller
4. Motor Controller to the Motors

This way of dividing the architecture based on the communication between the different components helped us "divide and conquer" the different aspects of the new architecture. These segments/modules can be visually represented as follows:

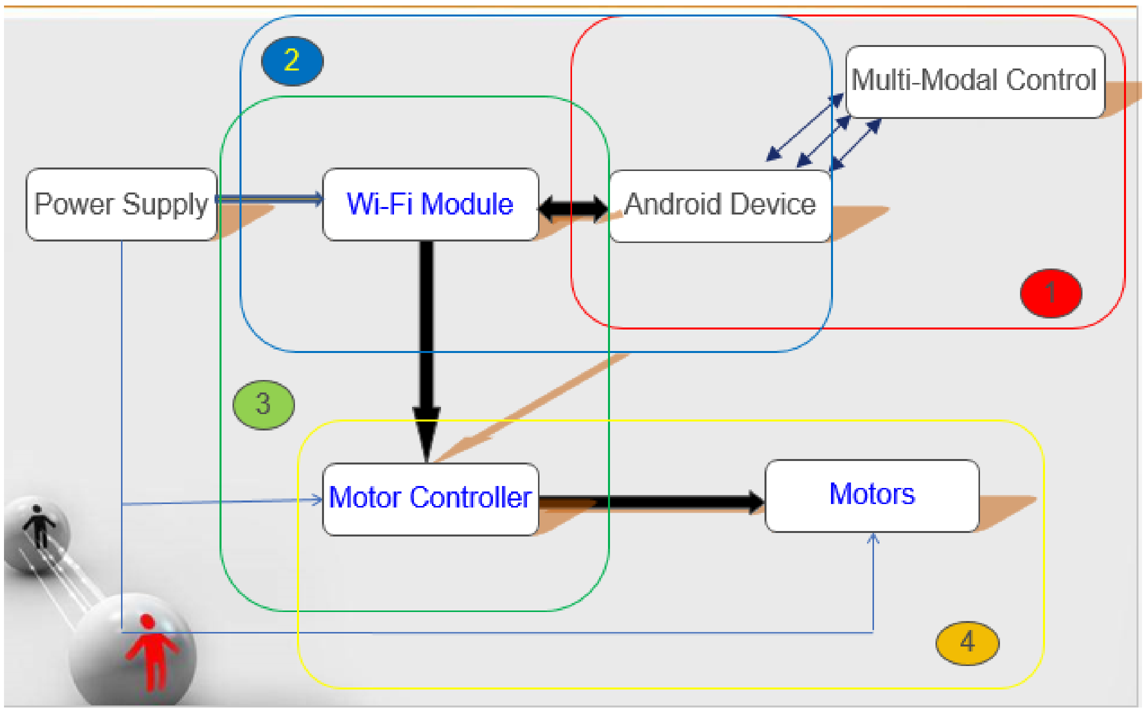


Figure 24: NEW ARCHITECTURE

Here, the Wi-Fi module acts as the means to communicate between the actual motor controller installed on the Wheelchair and the Android cellphone. While the first module will be discussed in the next objective: upgrading the UI and integrating offline voice control and GPS navigation, the second module, will be further detailed in objective 3: enhancing the wireless communication.

Likewise, the Motor Controller acts as the medium to control the motors directly via the Wi-Fi module commands it receives. So, it follows that the last two modules, numbered, 3 and 4 will be targeted in Objective 4: adding the speed regulation system.

Objective 2: Upgrading the Android application GUI and integrating offline voice control and GPS navigation

This section targets the first module of the new architecture namely, the multimodal control interface developed on the Android application. For this purpose, the Android application was upgraded to account for the changes in the architecture, such as the addition of a speed regulation system based on a motor controller circuit. Moreover, an effort was made to make the application more intuitive and user-friendly, and hence the Graphical User Interface (GUI) was also improved from the previous design.

Furthermore, the offline voice recognition control, already designed by the previous groups, was revised to switch from a manual download of the voice packages onto the Android device to an automatic download along with the application. Likewise, an additional flexibility was also provided to the users to be able to speak the same command in different ways. Besides, the GPS navigation was also modified to now be offline with the help of a third-party open source routable map of China.

Each of these changes made to the Android application is detailed in the following sections.

Upgrading the GUI

For enhancing the user experience when using the Android application, the previous design was scrutinized and several observations were made. Finally, a new design was prepared to target the shortcomings of the previous design. Following is a snapshot of the previous User Interface (UI):

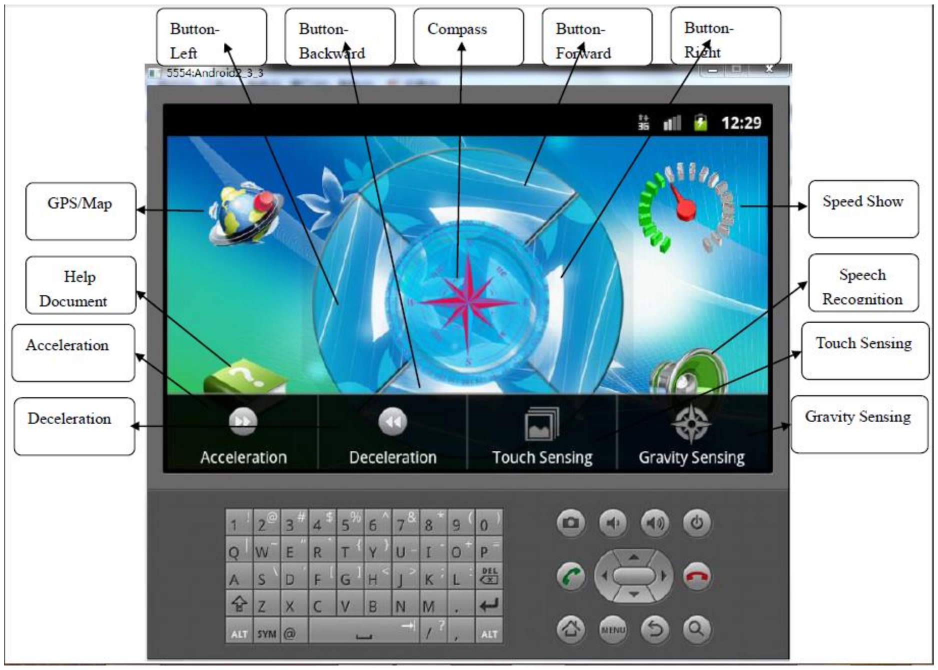


Figure 25: OLD GUI [31]

Taking the observations that were made on the old GUI into consideration, the improved Android application was made to target ease of accessibility and an efficient and intuitive design. The new GUI can be observed in Figure 26 below:

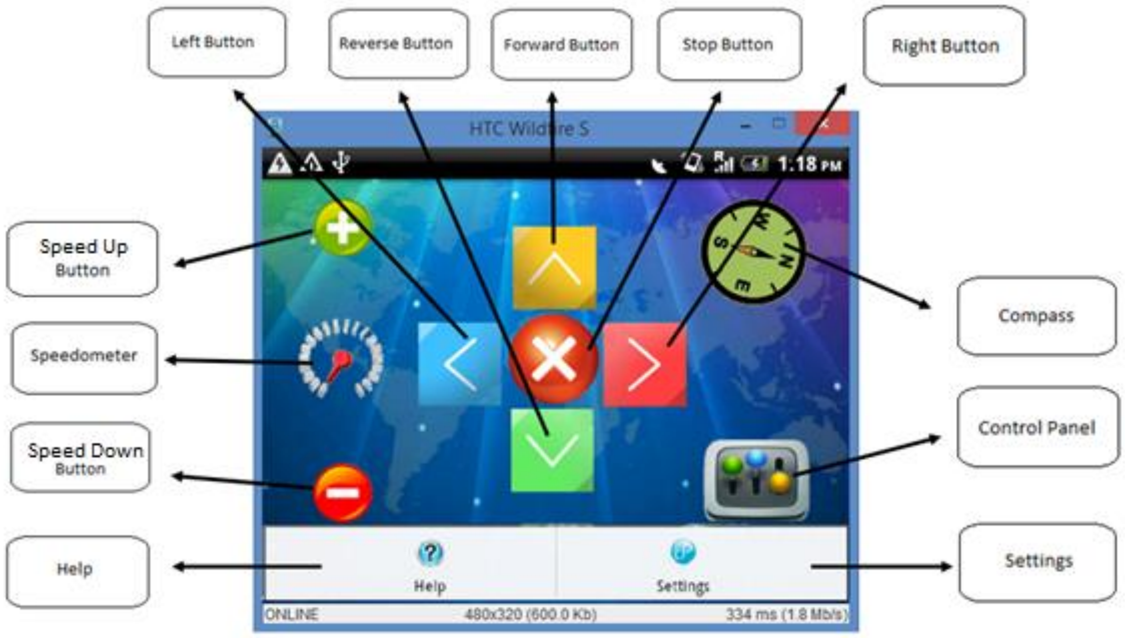


Figure 26: NEW GUI

Apart from adding a “Speed up” and “Speed down” buttons depicted by plus and minus signs, all the different multimodal control options were consolidated into a single control panel seen in the bottom right of the above figure. The speedometer was now made functional to display the 5 speed levels that was to be developed in Objective 4. The Compass in the top right was also modified to make use of a more real-time animation without any jitters, when the phone’s orientation is changed. Also, the options menu was altered to now constitute a “Help” and “Settings” options instead of the long list of “Acceleration”, “Deceleration”, “Touch sensing” and “Gravity Sensing” controls, which was hidden inside the options menu in the previous application.

Since the above modifications were made, the user can now simply use the control panel in the bottom right at any given point in order to switch to another control option. Additionally, the multimodal control option names were changed to sound more intuitive. Also, we made all of these multimodal control options of the Wheelchair appear right at the start of the application, instead of inconveniently navigating the Application to find a control option for the Wheelchair, as was done in the previous version of the Wheelchair Application. Following is a screenshot of the control options, which is displayed at the start of the Application as well as at any point when the Control Panel is clicked on while the application is still running:

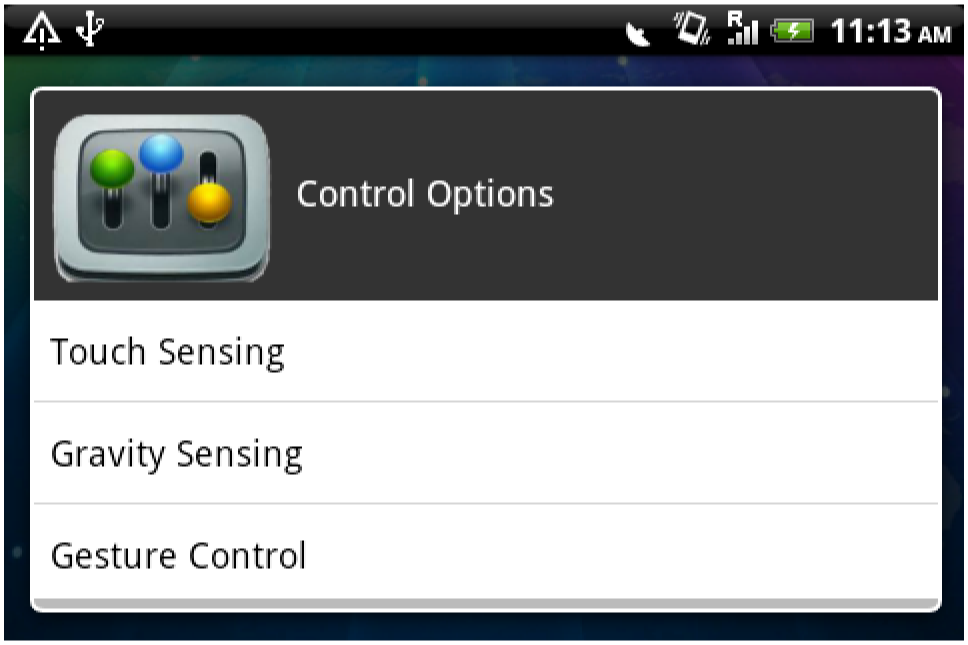


Figure 27: CONTROL OPTIONS IN UI

The user could scroll down to find the other control options, namely Voice Control and GPS Navigation. Other changes to the application involved remodeling the internal functionality of the voice control and GPS Navigation options.

The Voice Control option required manual installation of the voice recognition packages in the earlier version of the application, however, we made it easier for the user to use voice control by automatically downloading the required voice recognition packages when the application is started. Thus, the user could make full use of the Voice Control application feature without any hassles. Also, the offline database of the voice recognition control was modified to include both “slow down” and “speed down” commands to slow down the speed of the wheelchair to allow the user more flexibility when speaking the commands, especially to slow down in order to account for emergency cases.

Previously, the GPS navigation would require an online connection to the internet to display the map, which meant that the user cannot use the Wheelchair when connected to another Wi-Fi to use the internet facilities. Subsequently, we completely revamped the GPS navigation to make use of an offline map of China for navigating from one point to another.

Integrating Offline GPS Navigation

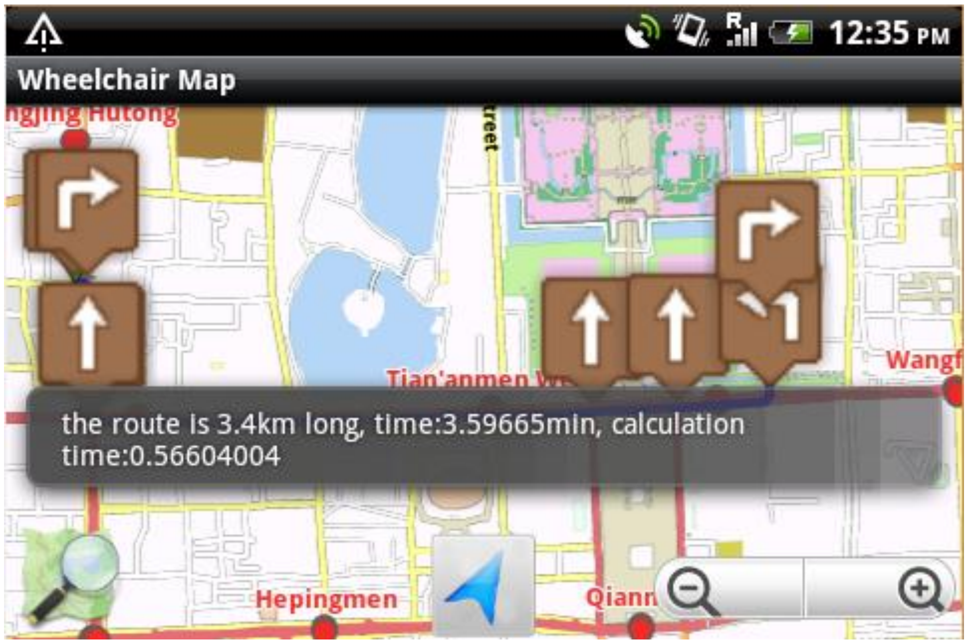
Earlier, the GPS navigation only worked online and made use of Baidu Maps, which was inconvenient as the user could no longer control the Wheelchair when in this mode. This was due to the fact that the user had to switch to another Wi-Fi network to gain internet connectivity, which implied that the user was no longer on the Wi-Fi module network that enables the control of the wheelchair. To solve this issue, we implemented an offline map navigation into the Android application, which made use of an offline routable map of China.

The offline map of China was not automatically downloaded onto the Android device, due to space constraints. Hence, for using the GPS navigation feature of the application, the user was required to download the freely available routable maps of their areas of interest onto their external Android device storage, that is, a Secure Digital (SD) card. Once installed, the user can navigate from one point on the map to another.

For the purpose of testing this feature, we used a routable map of China. These maps can be downloaded from Open Source Maps (OSM) for legal usage in software development. This map had to be downloaded onto the SD Card at location: "/Android/data/graphhopper/maps". This way, any number of maps can be loaded and used for navigation/routing.

For viewing, navigating and locating oneself on the map, we used a third-party Open Source Routing Engine which was packaged by another third-party open source SDK (Software Development Kit). These open source engines, namely, Graphhopper and Nutiteq Maps, helped in establishing a routing engine through an SDK that was used to display the maps, respectively. We also requested a license for using the engines and included it in the Application.

Once the open source files and libraries were included in the application package, the routing/navigation was automatically taken care of, by the GraphHopper routing engine when the user clicks on any two points on the map being displayed. A screenshot of the same can be seen as follows:



the route is 3.4km long, time:3.59665min, calculation time:0.56604004

Figure 28: UI DURING GPS NAVIGATION

The distance and time for travel is also calculated by the GraphHopper Engine, which is displayed after the route is calculated. We created an additional button in the bottom center on a separate layer on top of the map layer for users to determine their current location at any given point. This facility used the GPS location feature of the Android device to locate itself on the map by reverse geocoding from the latitude/longitude information provided by the GPS, similar to that of the previous GPS tracking. We used the SDK provided by Nutiteq for animating the current location on the map.

By doing the GPS navigation offline, the user can determine their current location on the map and plan their route accordingly. Subsequently, after noting the directions, the user can continue to navigate their way using one of the other control options, as they please, without having to switch back and forth between Wi-Fi network connections.

Objective 3: Enhancing the wireless communication to connect to a variety of cellphones without changing the internal settings of the phone

In this section, we focus on module 2 of the new architecture to communicate between the Android device and the Wi-Fi module. As discussed earlier in Objective 1, there existed a wireless communication problem between the cell phone and the previous Wi-Fi module, ST-MW-08S. The problem was that most Android devices could not detect the wireless signals sent from the Wi-Fi module without changing the internal settings of the device. This was because the previous module made use of ad hoc networks which by default, is inaccessible/unsupported by Android platforms, unless the device is rooted.

Thus, the problem stated above can be structure into be four different parts. We will introduce our solution to the problem, describe the new hardware circuit and finally explain the communication protocol and algorithm design.

The new wireless network section describes a new network type that was chosen. Then the hardware circuit explains how we test the new Wi-Fi module with different settings, communicating with different devices. Within “Programming the Communication Protocol” section, we show how the Wi-Fi module connects to the Android smart phone with the microcontroller. Lastly, the comparison of the current and previous algorithm design will be given to clearly show the difference and improvements.

Implementing New Wireless Network

The previous Wi-Fi module required that the Android cellphones installed an application named ad-hoc switcher, which changed the internal settings of the phone to allow connections to an ad hoc network. But, the installation process caused tremendous troubles for the new user. So, to avoid the unnecessary work, we decide to switch to a better option. In order to reach the goal of expandable and robust communication between the wheelchair and the Android smart phone with a background study in wireless communications, we considered and researched several options for a new wireless communication network option to solve the connection issues.

We selected the ST-MW-09S Wi-Fi module component to replace its old version, ST-MW-08S to fix the problem of using ad hoc networks. The most crucial upgrading from ST-MW-08S to ST-MW-09S is that the network mode changes from ad hoc to the SoftAP mode. The picture below shows how ST-MW-09S works.

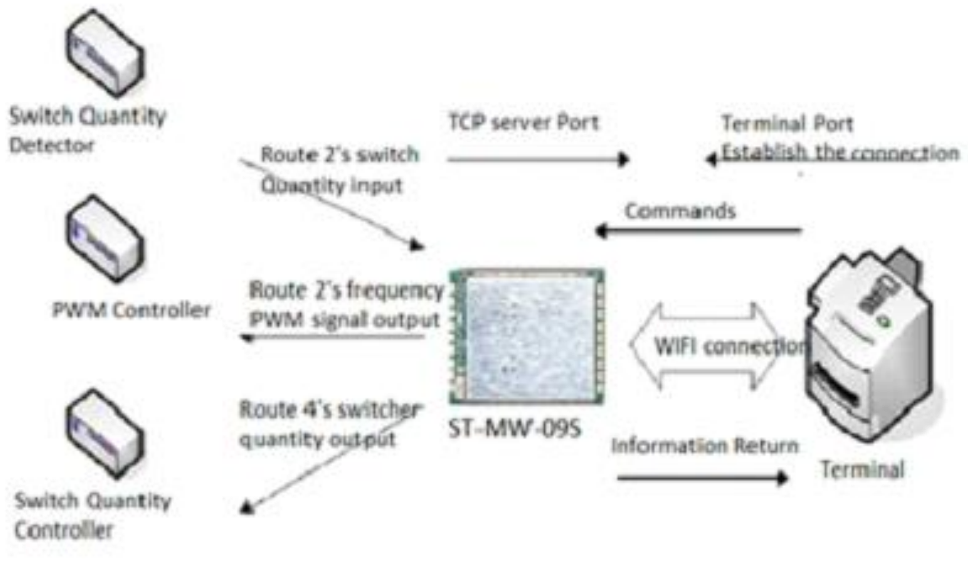

Switch Quantity Detector
TCP server Port
Terminal Port Establish the connection
Route 2's switch Quantity input
Commands
PWM Controller
Route 2's frequency PWM signal output
WIFI connection
Route 4's switcher quantity output
ST-MW-09S
Information Return
Terminal
Switch Quantity Controller

Figure 29: ST-MW-09S NETWORKING PROPERTIES

The working principle of ST-MW-09S is almost the same as its previous version, ST-MW-08S, other than the network type which is extremely critical to us since it support Android system naturally.

Hardware Circuit

The Wi-Fi module itself was packaged with the STC89C52 microcontroller with the following circuit diagram shown in Figure 30.

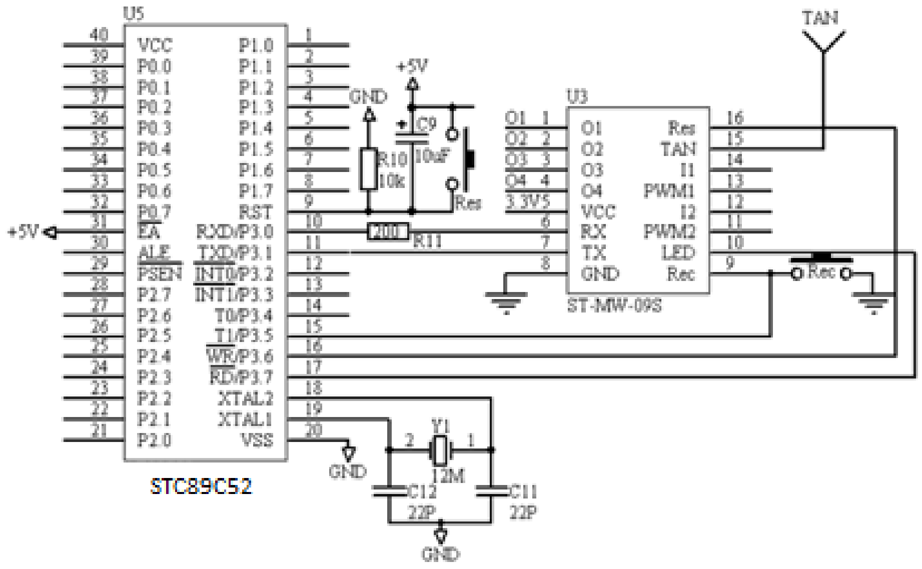


Figure 30: ST-MW-09S CIRCUIT DIAGRAM

This circuit shows the communication protocol of ST-MW-09S and STC89C52 is UART. And the data/signal from the Android phone received by ST-MW-09S is passed to STC89C52 through the UART communication protocol.

The entire motherboard with the STC89C52 microcontroller and the Wi-Fi module, namely, ST-MW-09S can be seen in the following Figure 31:

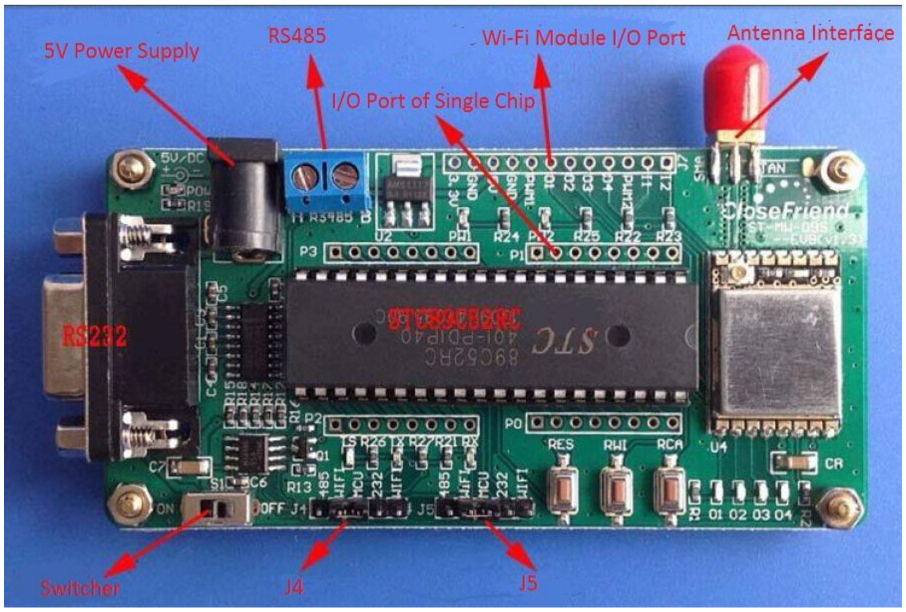


Figure 31: MOTHERBOARD WITH MICROCONTROLLER AND WI-FI MODULE

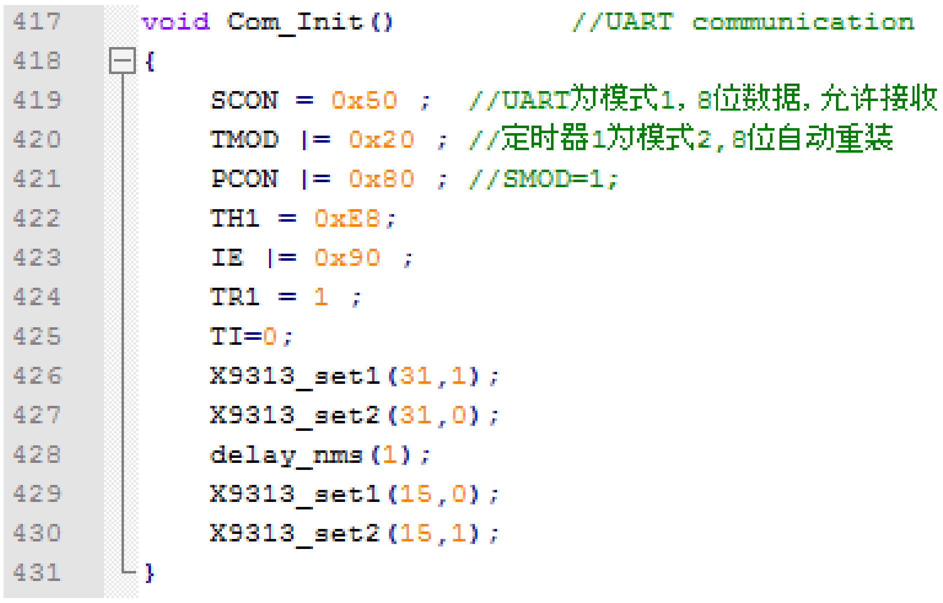
As seen in the Figure 26, there are three switches located in the bottom right side of the motherboard, which are RES, RWI and RCA. RES is a reset button used to reset the single chip, RWI resets the Wi-Fi module in real-time and RCA resets the Wi-Fi module to its default factory settings. Once this circuit is constructed, the Wi-Fi signals are transmitted and the data from the Android cell phones is received by the single chip, STC89C52. The jumpers J4 and J5, in the above figure, can be used to achieve four different means of communication between the single chip, the Wi-Fi module and the Android phone. We mainly used three of these communication ways:

1. The first way is to connect the third and fourth pin of jumpers, J4 and J5. This enables us to upload the program to the single chip from the computer.
2. The second way is to connect the second and third pin of J4 and J5 in order to establish a communication between the single chip and the Wi-Fi module. We could then use this method to control the wheelchair by Android phones.
3. The third way is to connect the fourth and fifth pin of J4 and J5 to establish communication between the Wi-Fi module and our computer for the purpose of debugging/testing the program on the Wi-Fi module. This selection was used to test the commands sent by the Android smart phones.

After doing the tests to see if the Wi-Fi works properly and the commands from cell phones can be received by the single chip, we control the wheelchair by the second way. As mentioned above, the communication protocol of STC89C52 and ST-MW-09S is the UART which is introduced in the next part.

Programming the Wi-Fi Communication Protocol

The Wi-Fi module uses UART (universal asynchronous receiver/transmitter) to communicate with the microcontroller, STC89C52. The code shown in Figure 32 below represents the initialization of the UART communication, "translating data between parallel and serial forms" [13].

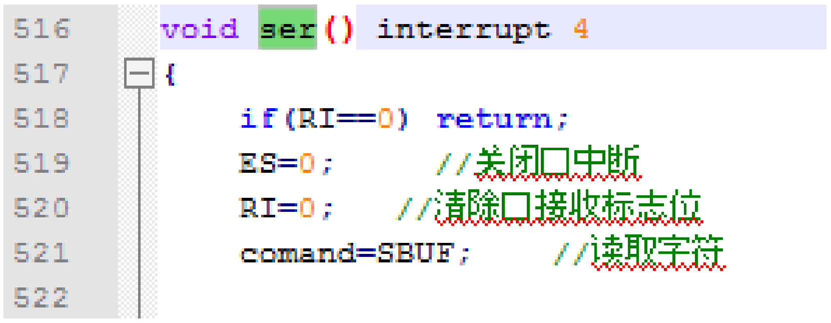


```
void Com_Init()            //UART communication
{
    SCON = 0x50 ;  //UART为模式1, 8位数据, 允许接收
    TMOD |= 0x20 ; //定时器1为模式2,8位自动重装
    PCON |= 0x80 ; //SMOD=1;
    TH1 = 0xE8;
    IE |= 0x90 ;
    TR1 = 1 ;
    TI=0;
    X9313_set1(31,1);
    X9313_set2(31,0);
    delay_nms(1);
    X9313_set1(15,0);
    X9313_set2(15,1);
}
```

Figure 32: UART COMMUNICATION INITIALIZATION

Here, the X9313_set methods are initialized for use in interfacing with the motor controller via digital potentiometers (refer to Objective 4 for more details).

The code shown in Figure 33 below provides the interrupt service routine (ISR), which is "a callback function in the microcontroller firmware", called every time that the Wi-Fi module receives any data [14]. Following is the code snippet of the ISR:

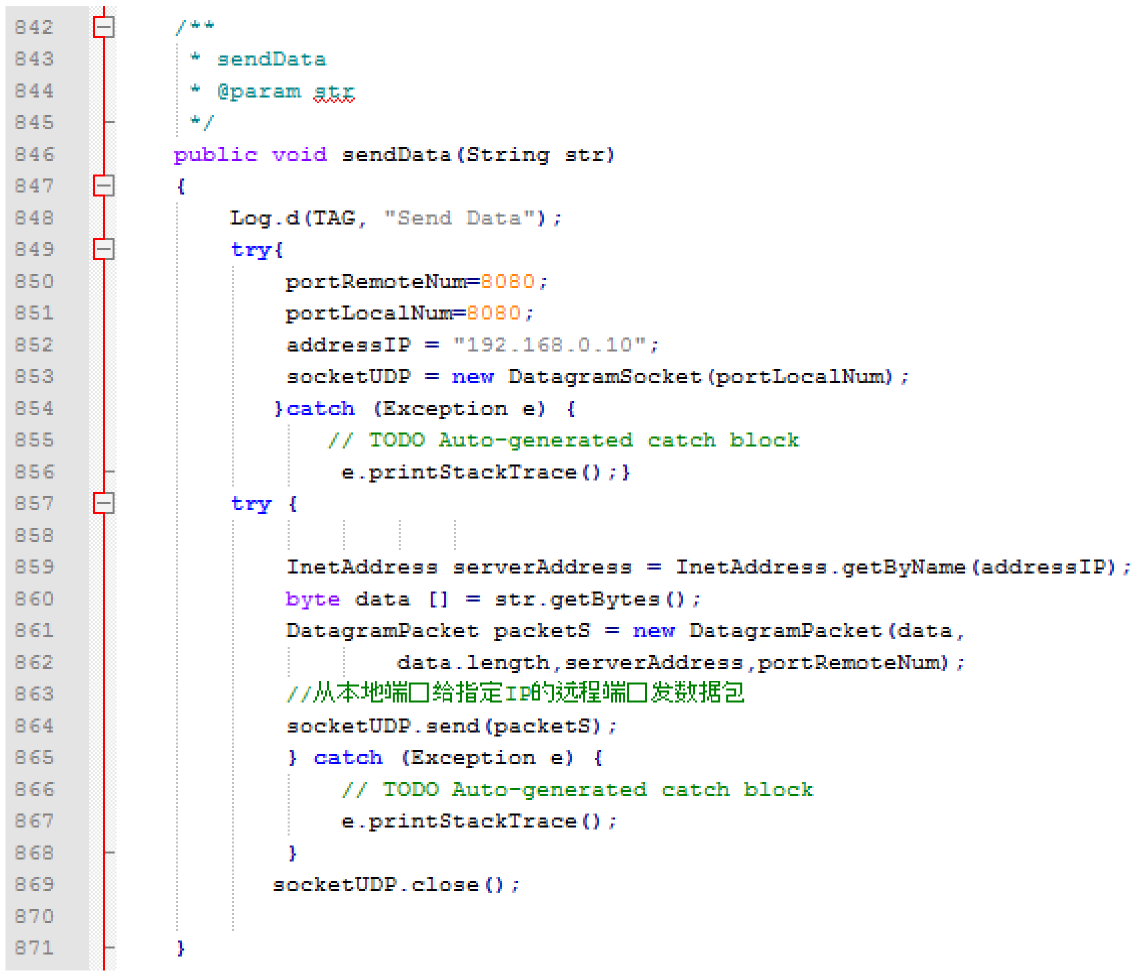


```
void ser() interrupt 4
{
    if(RI==0) return;
    ES=0;     //关闭口中断
    RI=0;   //清除口接收标志位
    comand=SBUF;    //读取字符
```

Figure 33: ISR FOR WI-FI COMMUNICATION

The ISR provides the function that the microcontroller, STC89C52 will clear and reset itself each time when it receives every single command, this makes every command to be independently received.

The code shown in the Figure 34 represents how Android phones detect and communicate with the Wi-Fi module by entering the exact port numbers and IP address. The port remote number, port local number and IP address are provided in the datasheet of the Wi-Fi module and can be set individually. Following is the code snippet of the same:



```
/**
 * sendData
 * @param str
 */
public void sendData(String str)
{
    Log.d(TAG, "Send Data");
    try{
        portRemoteNum=8080;
        portLocalNum=8080;
        addressIP = "192.168.0.10";
        socketUDP = new DatagramSocket(portLocalNum);
       }catch (Exception e) {
           // TODO Auto-generated catch block
            e.printStackTrace();}
    try {

        InetAddress serverAddress = InetAddress.getByName(addressIP);
        byte data [] = str.getBytes();
        DatagramPacket packetS = new DatagramPacket(data,
                data.length,serverAddress,portRemoteNum);
        //从本地端口给指定IP的远程端口发数据包
        socketUDP.send(packetS);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
       socketUDP.close();

}
```

Figure 34: CODE FOR SENDING COMMANDS FROM THE ANDROID CELLPHONE

The sendData () function initializes the port numbers and IP address as the mailing address. It is used to send data packages from the local port to the entered IP remote port to comply the wireless communication between Android phones and Wi-Fi module.

Algorithm Design

The flow charts of the previous and modified application are shown below in Figure 35 and Figure 36, respectively

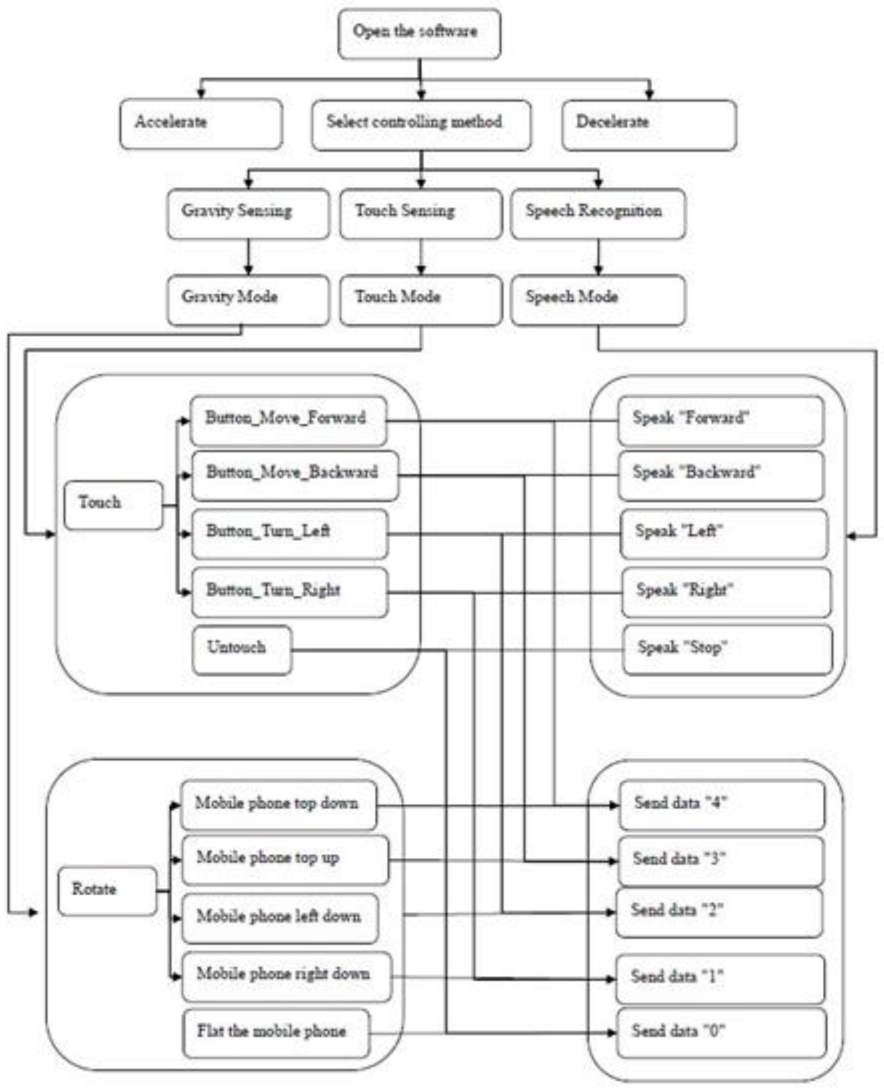


Figure 35: FLOWCHART OF PREVIOUS APPLICATION

The above flowchart describes the previous flow chart of the algorithm design, intuitively, users would have to open the application and navigate through it to find and select the control methods. The designs of acceleration and decelerations have not been complied yet by the previous group, however, we move them into the sub parts of different control methods. Each method has the same commands and each command eventually merges into the same signal.

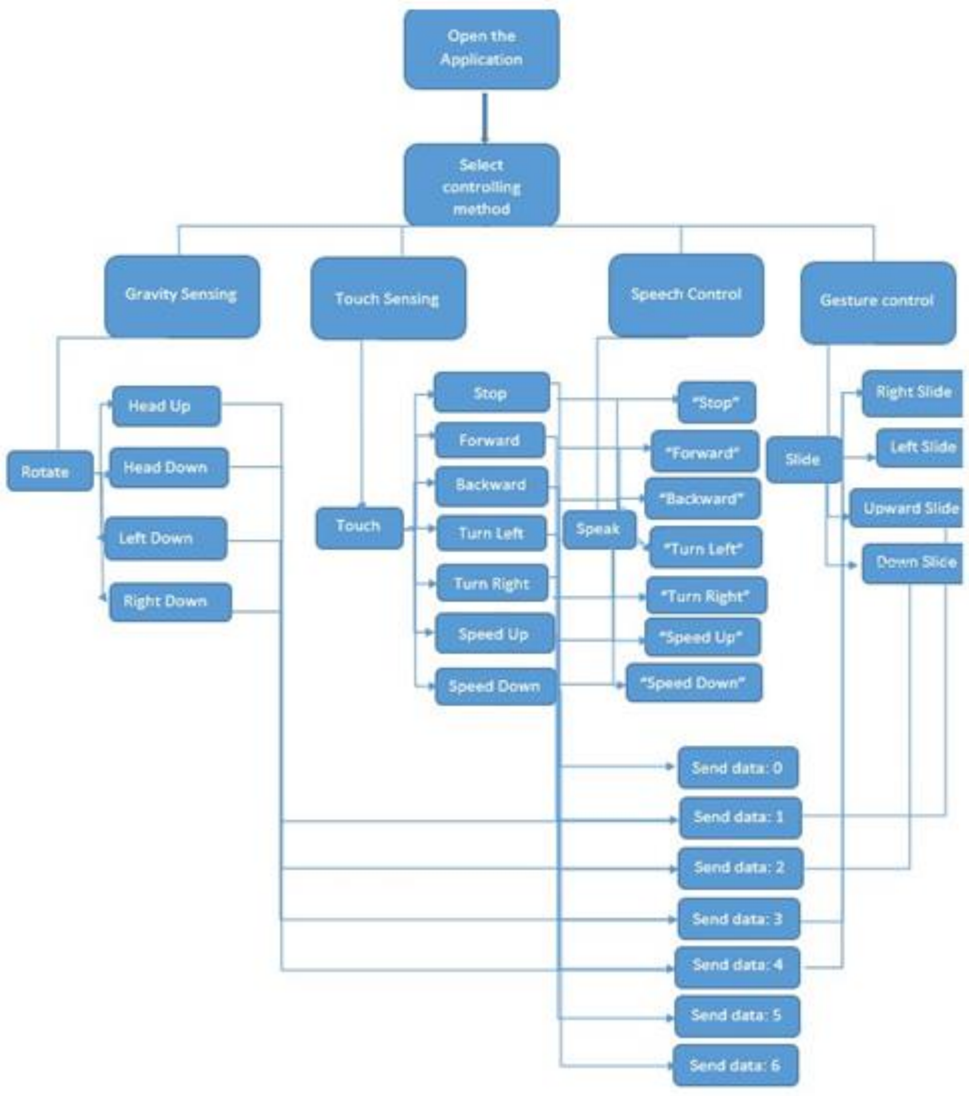


Figure 36: FLOWCHART OF MODIFIED APPLICATION

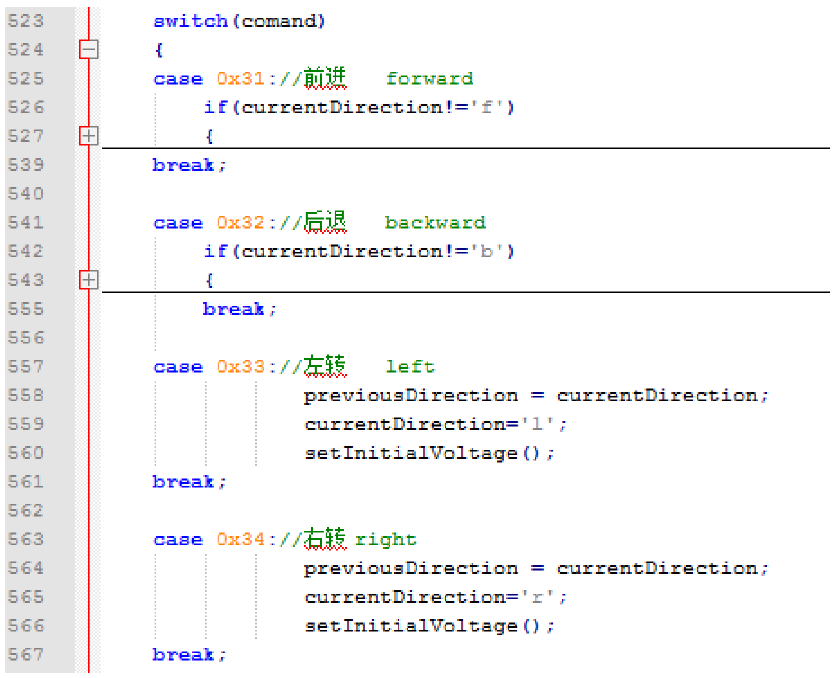
The hexadecimal code values that were sent over to the wireless module were modified to include "speed up" and "speed down" commands since we added the speed regulation system in the old architecture. From the current flow chart of the application, one can see that there are two more selections added to different control methods, "speed up" and "speed down", these two commands are represented by the characters "5" and "6" separately. The Table 5 below shows each of the hexadecimal code corresponding to the characters/commands.

Hexadecimal Code	Command
0x30	Stop
0x31	Forward
0x32	Backward
0x33	Turn left
0x34	Turn right
0x35	Speed up
0x36	Speed down

Table 5: HEXADECIMAL ASCII CODE CONVERSIONS

The hexadecimal ASCII code conversions of these characters can be mapped onto 0x30 to 0x36 that corresponds to the above-mentioned commands.

The code shown in Figure 37 below includes how wheelchair moves after receiving different commands. As mentioned above, there are several cases can be switched in our program. Following is the code snippet of different cases:



```
switch(comand)
{
case 0x31://前进   forward
    if(currentDirection!='f')
    {
break;

case 0x32://后退   backward
    if(currentDirection!='b')
    {
    break;

case 0x33://左转   left
            previousDirection = currentDirection;
            currentDirection='l';
            setInitialVoltage();
break;

case 0x34://右转 right
            previousDirection = currentDirection;
            currentDirection='r';
            setInitialVoltage();
break;
```

Figure 37: SWITCH CASES FOR COMMAND TRANSLATION

The case of moving forward which has the hexadecimal code as 0x31, the code will determine if the current direction of the wheelchair matches what the code needs and then switch to the required direction. The case of moving backward, turning left and right are also similar to the moving forward one. They all need to check the current direction then initial a new direction. For the case of speeding up and speeding down, the code will only adjust the speed itself of the wheelchair.

Objective 4: Adding the speed regulation system in the hardware circuit

To achieve this objective, we focused on handling the last two segments/modules of the new architecture, namely, 3 and 4, to constitute the communication between the wireless module and the motor controller as well as the motor controller and the motors themselves. The motor controller could reach 5 different speed levels after its implementation. However, only three of the control methods: touch sensing, voice control and gesture control can make use of the speed regulation system, other control methods, such as gravity sensing and GPS navigation cannot adjust the speed yet.

As discussed in the previous objective, the wireless communication architecture was modified to be replaced with the new Wi-Fi module device that connects to Android devices seamlessly. However, the communication between the Wi-Fi module and the motor controller for speed regulation was still lacking and hence needed to be added.

Hardware Circuit

Since, after much ado, RoboClaw 2x60A motor controller was selected, purchased and shipped to control the direction and speed of the two DC motors, it implies that the motor controller circuit was to be designed to communicate with the Wi-Fi module.

The main battery power supply and motor connections/terminals were available on board the RoboClaw, which were then connected to the two 12V batteries in series to produce 24V and the two 24V DC motors, respectively. The logic of the RoboClaw was supplied with a separate 5V from another battery source. Following circuit diagram shows the motor terminals and power supply connections to the RoboClaw.

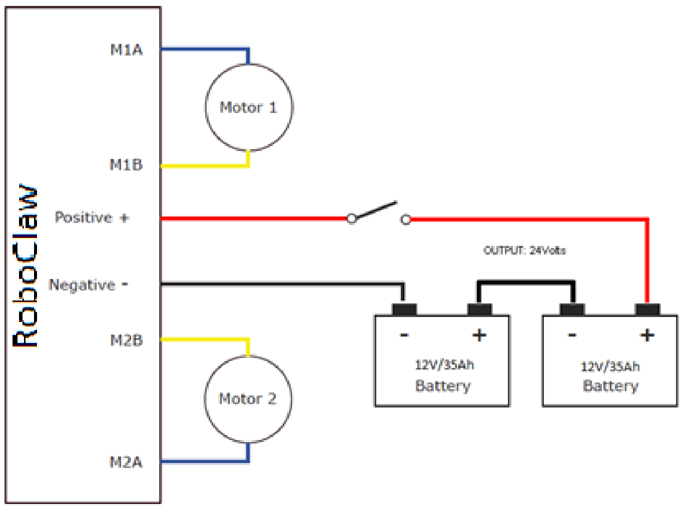


Figure 38: ROBOCLAW MAIN BATTERY POWER SUPPLY CONNECTIONS

As can be seen above, a circuit breaker switch that could handle the high voltage and current was installed on the batteries for emergency cases, in which the Wheelchair may not be able to be controlled with the Android phones for whatever reason and the only option is to cut-off the voltage to the batteries for the Wheelchair to stop. Besides, it also helps protect the motor controller circuit in case there is an overload or short circuit. Thicker wires were used to replace the old wires for all these connections as they have less resistance, thereby generating less heat when large current passes through them. The voltage drop across thicker wires is also smaller, thus, making it a safer option for the motor controller circuit that can possibly draw large amounts of current.

The Roboclaw motor controller had several modes of control ranging from simple serial (one-way), packet serial (two-way), RC (Radio Controlled), Analog and USB modes to control the motors. We could not use the serial mode due to lack of available UART ports on the microcontroller as it was already in use by the Wi-Fi module.

RoboClaw handled the direction and speed changes in a 0-2V range, where the different voltages in this range correspond to different speeds and directions of the motors: 0V corresponds to full reverse, 1V corresponds to stop and 2V corresponds to full forward. So, in order to establish communication between the Wi-Fi module and the motor controller, the analog mode of RoboClaw was used to supply voltage in the range of 0-2V.

The analog signals needed for RoboClaw in the analog mode could be generated as either PWM signals from a micro controller or as analog voltage from a potentiometer. Instead of generating PWM signals, which takes more computational time on the microcontroller and also since, we are aiming for a self-contained robotic application of the wheelchair, we decided to use digital potentiometers, namely the X9313, to generate the required analog signals for speed and direction control. The circuit design of the Wi-Fi module connected to the motor controller via the digital potentiometers can be seen as follows:

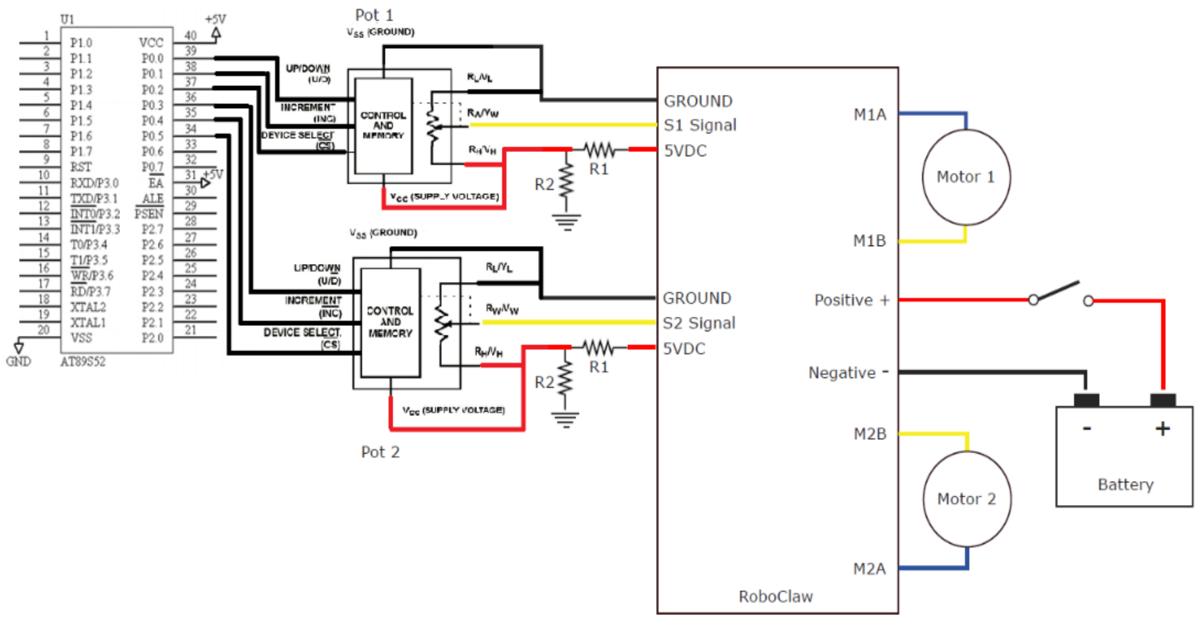


Figure 39: ROBOCLAW TO WI-FI MODULE CONNECTION VIA DIGITAL POTENTIOMETERS

The microcontroller in the above figure is already connected to the Wi-Fi module in the motherboard, which can be seen in the previous objective. So, the communication between the Wi-Fi module and the motor controller was thus based on the translation of the hexadecimal commands received by the Wi-Fi module to the digital potentiometer voltage values received at pins S1 and S2.

Since, the potentiometers wipers were limited to stay between 0-2V, the 5V DC power supply pins of the RoboClaw were connected to a resistor divider circuit with R1 = 3.3k and R2 = 2.2k to produce the 0-2V range power supply for the digital potentiometers.

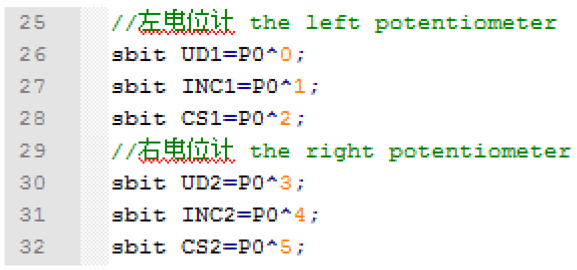
Once connected, the circuit involved setting the Analog mode on the RoboClaw using three push buttons on board, which blinked several LEDs to indicate the current mode of operation. The battery cut-off settings were also set in a similar fashion. Once set, the RoboClaw saves the settings into memory and can be plugged in to the power supply and used instantly thereafter.

The digital potentiometers used for the purpose of generating the 0-2V were to be controlled by the microcontroller that was already connected to the Wi-Fi module. The following section discusses this communication between the motor controller/Wi-Fi module and the X9313 digital potentiometers used to set the motor controller into action.

Interfacing between the Wi-Fi Module/Microcontroller and Motor Controller

The X9313 digital potentiometers were used for interfacing between the Wi-Fi module and the motor controller. It consisted of 32 wiper taps and was initialized and set inside the same program written for the Wi-Fi module communication discussed in the previous objective. "Appendix B: X9313 Digital Potentiometer Specifications" presents further technical specifications of the digital potentiometers.

The digital potentiometer pins used for this purpose: Chip Select, Increment and Up/Down counter seen in the Figure above were initialized in the program as follows:

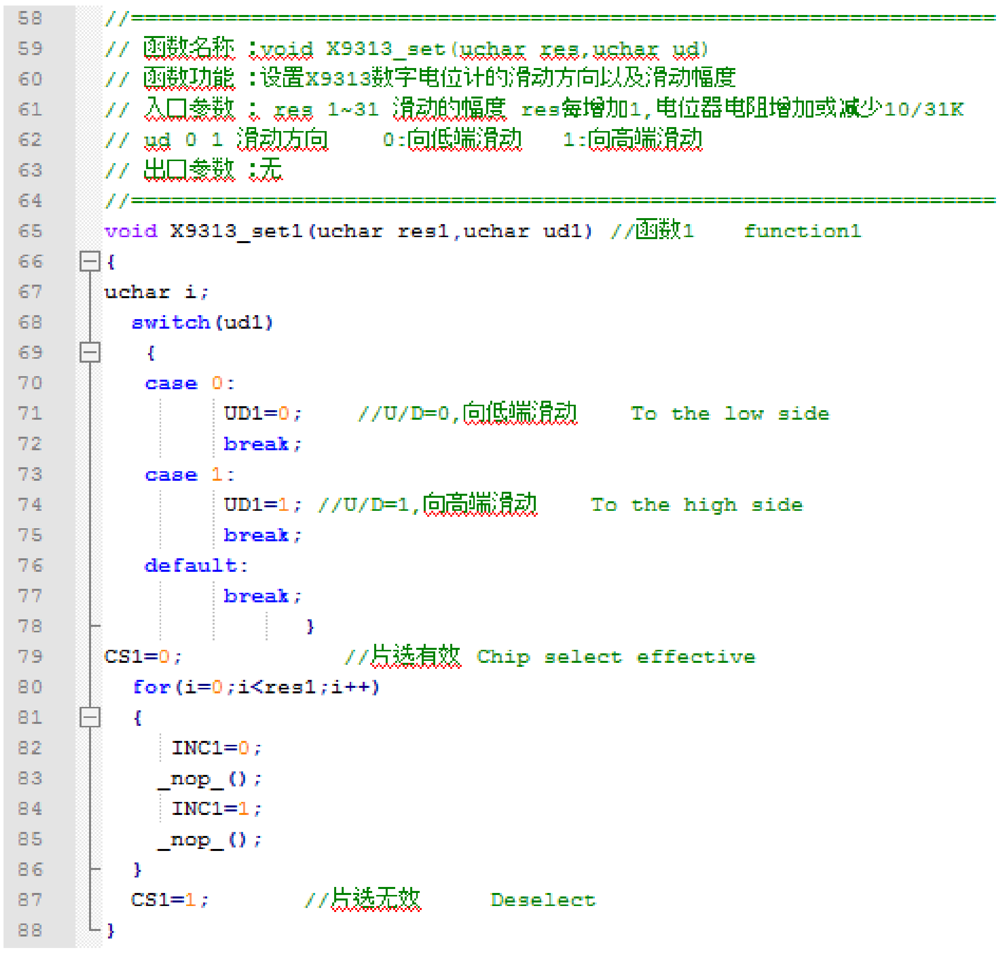


```
//左电位计 the left potentiometer
sbit UD1=P0^0;
sbit INC1=P0^1;
sbit CS1=P0^2;
//右电位计 the right potentiometer
sbit UD2=P0^3;
sbit INC2=P0^4;
sbit CS2=P0^5;
```

Figure 40: INITIALIZING THE DIGITAL POTENTIOMETER PINS

Once the potentiometer pins were initialized as seen above, the wipers were set to 1V inside the Com_init () method that was seen in the previous objective. This was done so that when the system was powered up, the motors are at rest, since 1V to the RoboClaw's S1 and S2 signal pins corresponds to stop.

The X9313_set1 () and X9313_set2 () methods were used to set the wipers of the left and right potentiometers, respectively. This method can be observed in the following code snippet:



```
//====================================================================
// 函数名称 :void X9313_set(uchar res,uchar ud)
// 函数功能 :设置X9313数字电位计的滑动方向以及滑动幅度
// 入口参数 : res 1~31 滑动的幅度 res每增加1,电位器电阻增加或减少10/31K
// ud 0 1 滑动方向     0:向低端滑动    1:向高端滑动
// 出口参数 :无
//====================================================================
void X9313_set1(uchar res1,uchar ud1) //函数1    function1
{
uchar i;
  switch(ud1)
   {
    case 0:
          UD1=0;    //U/D=0,向低端滑动    To the low side
          break;
    case 1:
          UD1=1; //U/D=1,向高端滑动    To the high side
          break;
    default:
          break;
                }
CS1=0;                //片选有效 Chip select effective
  for(i=0;i<res1;i++)
  {
     INC1=0;
    _nop_();
     INC1=1;
    _nop_();
  }
  CS1=1;          //片选无效     Deselect
}
```

Figure 41: SETTING X9313 IN THE PROGRAM

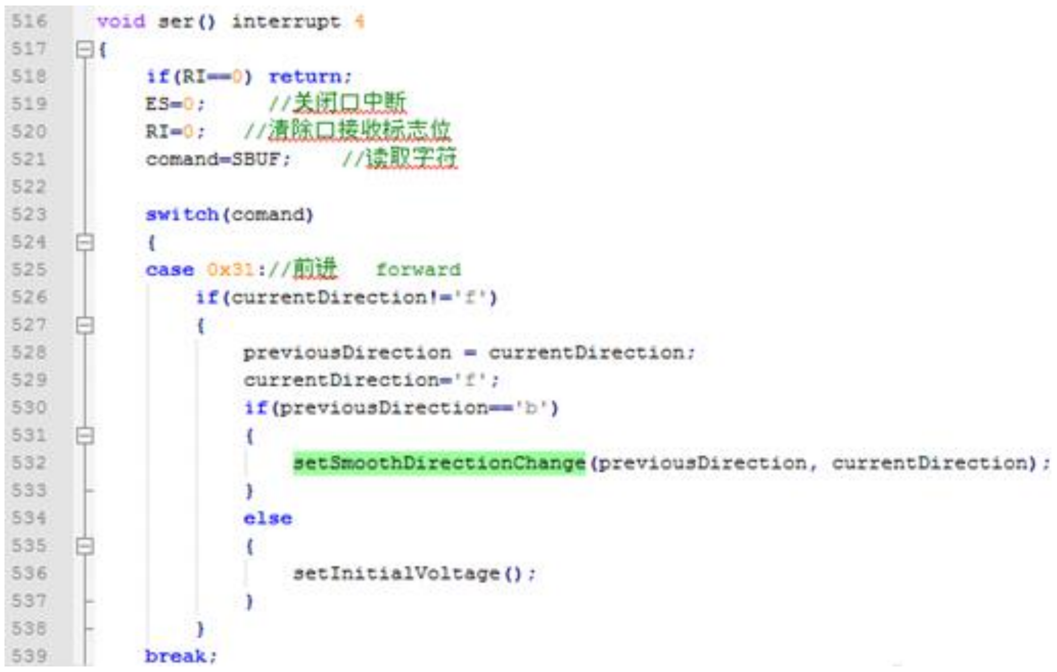
Here, one can observe that the wiper movement in terms of up/down was one of the parameters to the set () method and the other was the resolution of the wiper taps going from 0-31. The Chip Select pin was used when actually incrementing the wiper from the reference point.

Thus, the next step was to ensure that the potentiometer setting was smooth enough for safe operations so that there were no sudden movements in the Wheelchair when certain commands were sent. Especially, when accelerating/decelerating at a higher speed or making turns, the smoothing of the movement was even more necessitated. The next section describes our approach for smoothing the movements of the Wheelchair

Smoothing Acceleration/Deceleration and Turns

To avoid sudden jolts while driving the wheelchair, the left and right turn commands were set to be smooth, while the forward and backward commands were set to be continuous, unless accelerating. So, the wheelchair would only turn smoothly for 3 seconds and subsequently stop. When accelerating, the wheelchair would once again have to be smooth and not abrupt, as it could be dangerous for the user when going at higher speeds.

Since the change in speed and direction, both had to be smooth, certain functions in the Wi-Fi module program, were made use of to attain this smoothness. This was done by incrementing the voltage by 0.1V until the target voltage was reached. With 0.1V increments, the Wheelchair achieved 5 different speed levels at 0.2V increments between 0-1V and 1-2V, each going in a different direction, respectively. All of this was achieved inside the interrupt service routine (ISR) for the reception of the data over Wi-Fi. Following is a code snippet of the ISR:



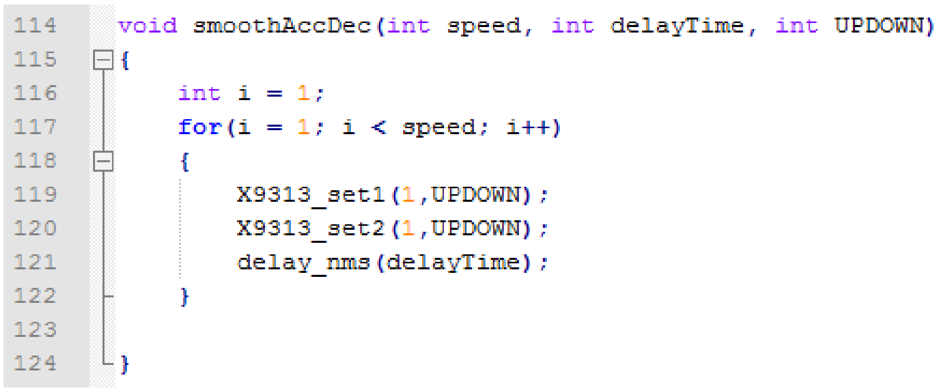
```
void ser() interrupt 4
{
    if(RI==0) return;
    ES=0;       //关闭口中断
    RI=0;    //清除口接收标志位
    comand=SBUF;     //读取字符

    switch(comand)
    {
    case 0x31://前进   forward
        if(currentDirection!='f')
        {
            previousDirection = currentDirection;
            currentDirection='f';
            if(previousDirection=='b')
            {
                setSmoothDirectionChange(previousDirection, currentDirection);
            }
            else
            {
                setInitialVoltage();
            }
        }
    break;
```

Figure 42: SMOOTH DIRECTION CHANGES IN ISR

The function prototypes to achieve the smooth changes in speed and direction are stated as follows:

1. void smoothAccDec(int increment, int delayTime, int UPDOWN)
2. void setStopVoltage()
3. void setSmoothDirectionChange(previousDirection, currentDirection)

The first method mentioned above smoothAccDec() either incremented or decremented the digital potentiometer wiper by 0.1V smoothly with a delay to match the acceleration/deceleration in speed requested by the user. Here's the function definition:



```
void smoothAccDec(int speed, int delayTime, int UPDOWN)
{
    int i = 1;
    for(i = 1; i < speed; i++)
    {
        X9313_set1(1,UPDOWN);
        X9313_set2(1,UPDOWN);
        delay_nms(delayTime);
    }

}
```

Figure 43: SMOOTH ACCELERATION/DECELERATION METHOD

Once the digital potentiometers were being set to these particular voltages, the motor controller automatically handled the speed and directional changes of the motors in a smooth fashion. Similarly, the setStopVoltage() and setSmoothDirectionChange() methods smoothly stopped and changed the direction of the Wheelchair, respectively, at any given time.

After implementing the speed regulation system and replacing the Wi-Fi module, we required a basic structure to test the capabilities of our design on a smaller platform for simulation purposes. Therefore, we used a new version of the “baby car” that was provided to us by DEPUSH, and named it “insta-car”. Using insta-car, we were able to simply plug in the same circuit as for the Wheelchair onto a smaller simulator platform for testing. This not only ensured that the circuit worked, but, also presented an opportunity to demonstrate the expansibility of the system architecture that we designed. Following is the explanation of the methods we used to simulate our design of the new Wheelchair platform onto a smaller platform, namely the insta-car.

Objective 5: Simulating the developed features on a smaller platform for testing and ensure expandability of the Android program

After finishing tests of all the developed features on the wheel chair, we planned to simulate more tests on the baby car to ensure expandability of the Android program. The baby car is named by DEPUSH and given to us for doing simulations before the wheelchair was assembled. But since the components are totally different from the wheelchair, we decided our mind to do simulations for the expandability then. The original baby car is shown in Figure 44.

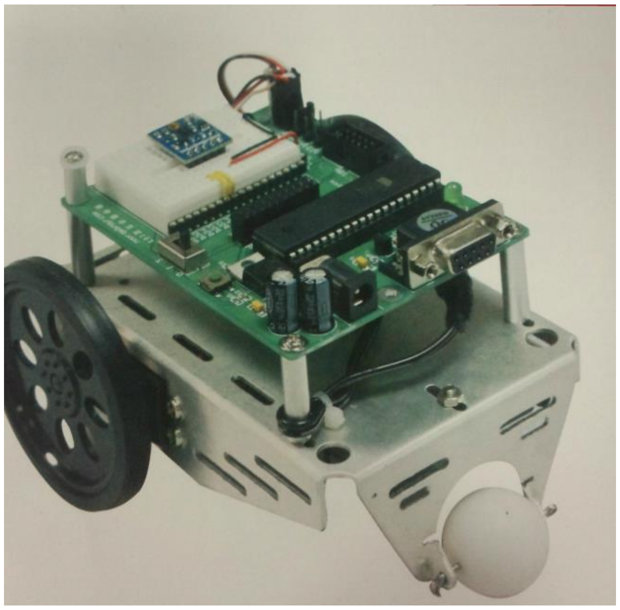


Figure 44: BABY CAR BY DEPUSH TECHNOLOGIES CO. LTD.

This is a small fundamental educational platform, the original version of it was not helpful to our project so we reassemble the baby car and renamed it "Insta-car".

We transplanted the entire system of wheelchair except the motor part to our new Insta-car. From the figure below, we have a comprehensive overlooking of the body part of the Insta-car. Though it is a pity that this is not the complete version of the Insta-car since the RoboClaw was moved before taking this picture.

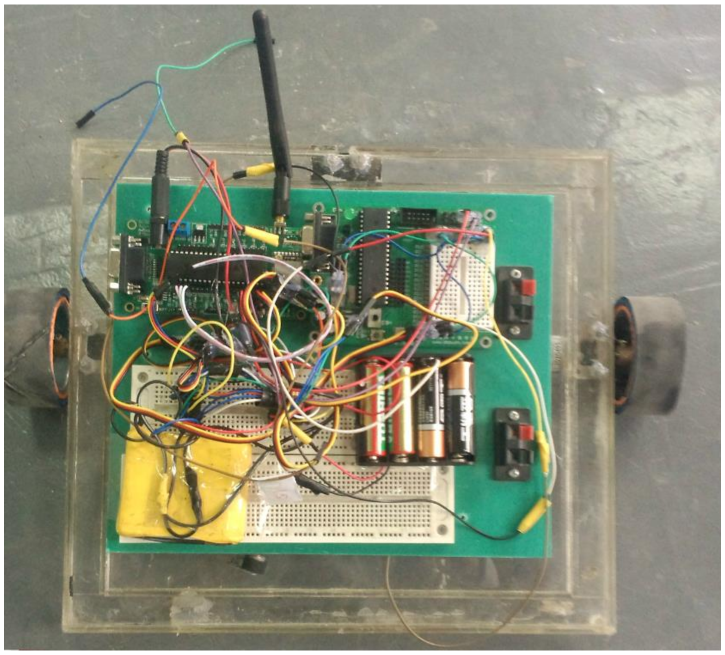


Figure 45: INSTA-CAR MODELED FOR SIMULATION

The top left side of the whole circuit board is transplanted from our wheelchair exactly the same. And we can also see a single chip in the top right side of the circuit board, but that circuit board was not used, however, that one served as the power module of the insta-car, which was removed from the baby car in order to solve the power supply problem.

Objective 6: Designing and manufacturing a prototype of the improved design

We finally designed the final prototype by implementing the above mentioned modifications to the hardware of the Wheelchair architecture and the software developments in the Android application used to control the Wheelchair. Hence, the overall architecture of the upgraded Wheelchair was developed as follows:

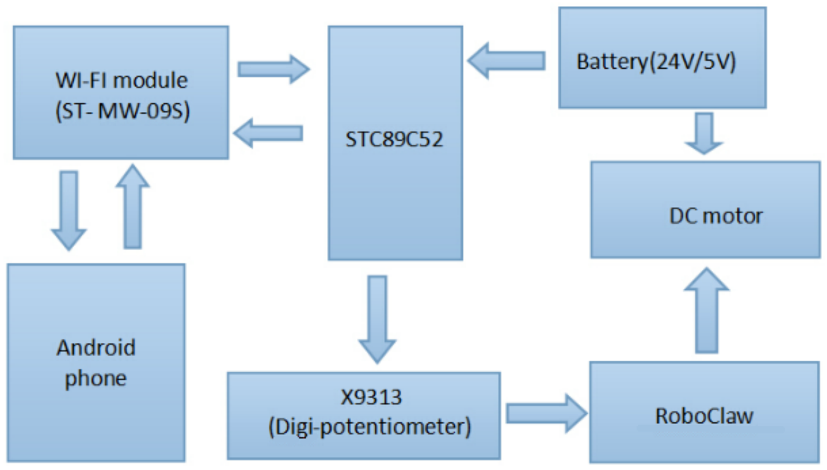


Figure 46: NEW OVERALL SYSTEM ARCHITECTURE

Finally, we designed the prototype to work based off of the HOQ, and compared our new design with the old one to gauge the improvement. The manufacturing process was not too complicated except building the hardware circuit itself and packaging the Android development of the application onto Github, for version control and future use of the product. The expansibility of the product was demonstrated using the insta-car developed for this purpose.

Overall, using the methods mentioned above we accomplished our objectives in a timely fashion to meet our final goal. The results and analysis of implementing these methods can be observed in the next chapter.

Chapter 4: Results and Analysis

In this chapter we will target the results from each objective and thoroughly analyze the same. Comprehensively, we hope that the results from this project can help future development, manufacturability and usage.

Visit to the Sponsor Company

During our visit to DEPUSH in Wuhan, the head of the company showed us several educational robots, one of which was a product developed by past WPI and HUST student group. This robot was equipped with a Kinect sensor and was programmed to detect and follow humans and recognize speech commands. Pictures of the visit can be seen as follows:

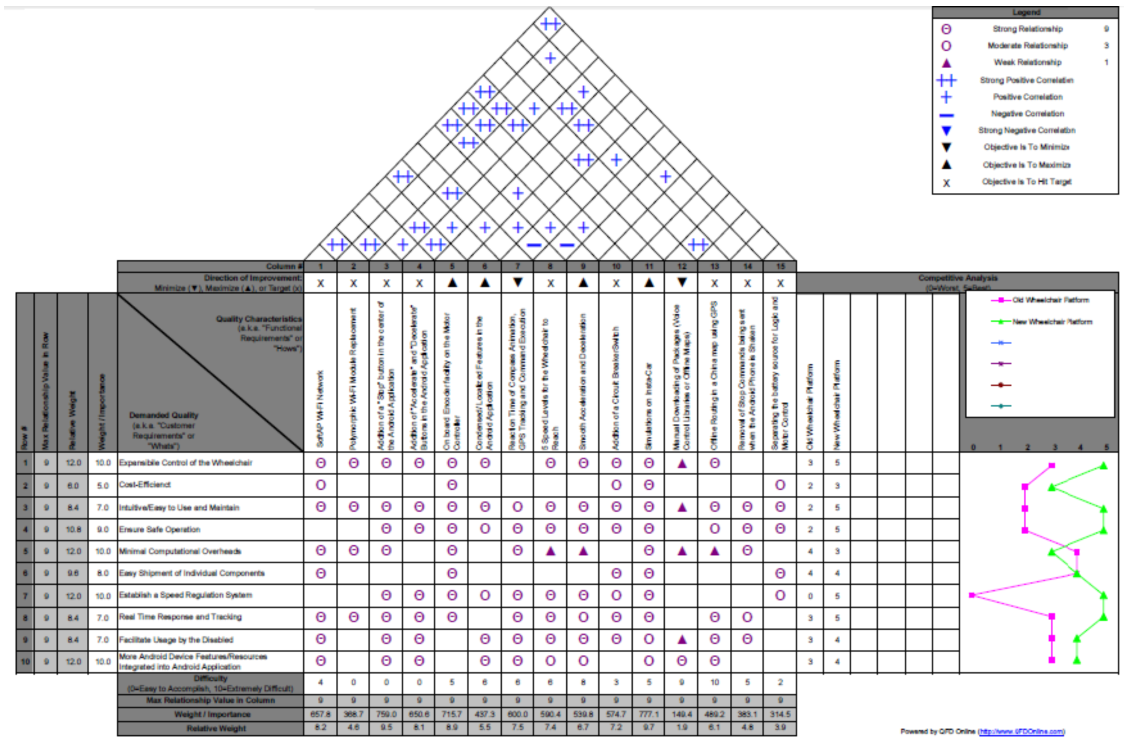
Figure 47: VISIT TO DEPUSH TECHNOLOGIES CO. LTD.

A baby car was requested and provided to us to perform the simulation of the complete product enhancement that we make to the Wheelchair platform, including the wireless communication, speed regulation and other control options implemented on the Android phone application.

From the visit, we learned that the main purpose of developing the Wheelchair platform was for usage in academic institutions for educational reasons. Thus, the focus of our project was more on the ease of future development and expansibility of use in other products manufactured by DEPUSH. This insight helped us design a better product that suits the needs of the sponsor company.

The House of Quality

Following the visit to the company and the interviews with the previous groups, we formulated the following House of Quality diagram:

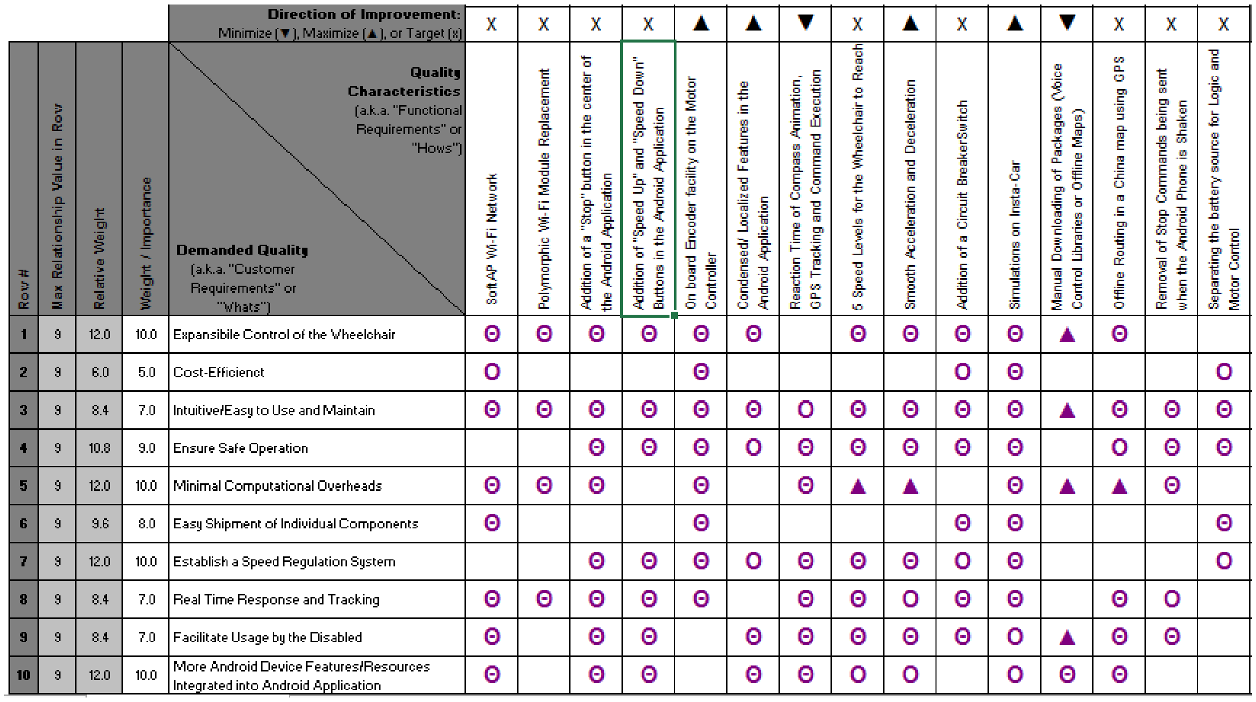


Legend
Θ Strong Relationship 9
O Moderate Relationship 3
▲ Weak Relationship 1
++ Strong Positive Correlation
+ Positive Correlation
− Negative Correlation
▼ Strong Negative Correlation
▼ Objective Is To Minimize
▲ Objective Is To Maximize
X Objective Is To Hit Target
Competitive Analysis (0=Worst, 5=Best)
Old Wheelchair Platform
New Wheelchair Platform
0 1 2 3 4 5
Powered by QFD Online (http://www.QFDOnline.com)

				Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	X	X	X	X	▲	▲	▼	X	▲	X	▲	▼	X	X	X		
Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") / Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	SoftAP Wi-Fi Network	Polymorphic Wi-Fi Module Replacement	Addition of a "Stop" button in the center of the Android Application	Addition of "Accelerate" and "Decelerate" Buttons in the Android Application	On board Encoder facility on the Motor Controller	Condensed/ Localized Features in the Android Application	Reaction Time of Compass Animation, GPS Tracking and Command Execution	5 Speed Levels for the Wheelchair to Reach	Smooth Acceleration and Deceleration	Addition of a Circuit Breaker/Switch	Simulations on Insta-Car	Manual Downloading of Packages (Voice Control Libraries or Offline Maps)	Offline Routing in a China map using GPS	Removal of Stop Commands being sent when the Android Phone is Shaken	Separating the battery source for Logic and Motor Control	Old Wheelchair Platform	New Wheelchair Platform
1	9	12.0	10.0	Expansible Control of the Wheelchair	Θ	Θ	Θ	Θ	Θ	Θ		Θ	Θ	Θ	Θ	▲	Θ			3	5
2	9	6.0	5.0	Cost-Efficienct	O				Θ					O	Θ				O	2	3
3	9	8.4	7.0	Intuitive/Easy to Use and Maintain	Θ	Θ	Θ	Θ	Θ	Θ	O	Θ	Θ	Θ	Θ	▲	Θ	Θ	Θ	2	5
4	9	10.8	9.0	Ensure Safe Operation			Θ	Θ	Θ	O	Θ	Θ	Θ	Θ	Θ		O	Θ	Θ	2	5
5	9	12.0	10.0	Minimal Computational Overheads	Θ	Θ	Θ		Θ		Θ	▲	▲		Θ	▲	▲	Θ		4	3
6	9	9.6	8.0	Easy Shipment of Individual Components	Θ				Θ					Θ	Θ				Θ	4	4
7	9	12.0	10.0	Establish a Speed Regulation System			Θ	Θ	Θ	O	Θ	Θ	Θ	O	Θ				O	0	5
8	9	8.4	7.0	Real Time Response and Tracking	Θ	Θ	Θ	Θ	Θ		Θ	Θ	O	Θ	Θ		Θ	O		3	5
9	9	8.4	7.0	Facilitate Usage by the Disabled	Θ		Θ	Θ		Θ	Θ	Θ	Θ	Θ	O	▲	Θ	Θ		3	4
10	9	12.0	10.0	More Android Device Features/Resources Integrated into Android Application	Θ		Θ	Θ		Θ	Θ	O	O		O	Θ	Θ			3	4
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	4	0	0	0	5	6	6	6	8	3	5	9	10	5	2		
				Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9		
				Weight / Importance	657.8	368.7	759.0	650.6	715.7	437.3	600.0	590.4	539.8	574.7	777.1	149.4	489.2	383.1	314.5		
				Relative Weight	8.2	4.6	9.5	8.1	8.9	5.5	7.5	7.4	6.7	7.2	9.7	1.9	6.1	4.8	3.9		

Figure 48: HOUSE OF QUALITY [40]

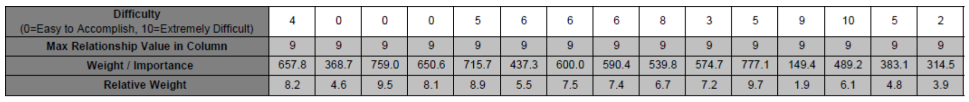
The demanded quality by the sponsor and its weight/importance was inserted into the rightmost columns. The Quality Characteristics that we developed through our project methodology with input from the previous groups were inserted into the uppermost rows of the house along with the direction of improvement. These can be seen as follows:



				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	X	X	X	X	▲	▲	▼	X	▲	X	▲	▼	X	X	X
Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	**Quality Characteristics** (a.k.a. "Functional Requirements" or "Hows") / **Demanded Quality** (a.k.a. "Customer Requirements" or "Whats")	SoftAP Wi-Fi Network	Polymorphic Wi-Fi Module Replacement	Addition of a "Stop" button in the center of the Android Application	Addition of "Speed Up" and "Speed Down" Buttons in the Android Application	On board Encoder facility on the Motor Controller	Condensed/ Localized Features in the Android Application	Reaction Time of Compass Animation, GPS Tracking and Command Execution	5 Speed Levels for the Wheelchair to Reach	Smooth Acceleration and Deceleration	Addition of a Circuit BreakerSwitch	Simulations on Insta-Car	Manual Downloading of Packages (Voice Control Libraries or Offline Maps)	Offline Routing in a China map using GPS	Removal of Stop Commands being sent when the Android Phone is Shaken	Separating the battery source for Logic and Motor Control
1	9	12.0	10.0	Expansibile Control of the Wheelchair	ϴ	ϴ	ϴ	ϴ	ϴ	ϴ		ϴ	ϴ	ϴ	ϴ	▲	ϴ		
2	9	6.0	5.0	Cost-Efficienct	O				ϴ					O	ϴ				O
3	9	8.4	7.0	Intuitive/Easy to Use and Maintain	ϴ	ϴ	ϴ	ϴ	ϴ	ϴ	O	ϴ	ϴ	ϴ	ϴ	▲	ϴ	ϴ	ϴ
4	9	10.8	9.0	Ensure Safe Operation			ϴ	ϴ	ϴ	O	ϴ	ϴ	ϴ	ϴ	ϴ		O	ϴ	ϴ
5	9	12.0	10.0	Minimal Computational Overheads	ϴ	ϴ	ϴ		ϴ		ϴ	▲	▲		ϴ	▲	▲	ϴ	
6	9	9.6	8.0	Easy Shipment of Individual Components	ϴ				ϴ					ϴ	ϴ				ϴ
7	9	12.0	10.0	Establish a Speed Regulation System			ϴ	ϴ	ϴ	O	ϴ	ϴ	ϴ	O	ϴ				O
8	9	8.4	7.0	Real Time Response and Tracking	ϴ	ϴ	ϴ	ϴ	ϴ		ϴ	ϴ	O	ϴ	ϴ		ϴ	O	
9	9	8.4	7.0	Facilitate Usage by the Disabled	ϴ		ϴ	ϴ		ϴ	ϴ	ϴ	ϴ	ϴ	O	▲	ϴ	ϴ	
10	9	12.0	10.0	More Android Device Features/Resources Integrated into Android Application	ϴ		ϴ	ϴ		ϴ	ϴ	O	O		O	ϴ	ϴ		

Figure 49: RANKING THE DEMANDED QUALITY VS. QUALITY CHARACTERISTICS

After rating each of the "Demanded vs. Quality Characteristics", based on their relationship, we obtained the following relative weights.



Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	4	0	0	0	5	6	6	6	8	3	5	9	10	5	2
Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Weight / Importance	657.8	368.7	759.0	650.6	715.7	437.3	600.0	590.4	539.8	574.7	777.1	149.4	489.2	383.1	314.5
Relative Weight	8.2	4.6	9.5	8.1	8.9	5.5	7.5	7.4	6.7	7.2	9.7	1.9	6.1	4.8	3.9

Figure 50: RELATIVE WEIGHTS CALCULATED FROM HOQ

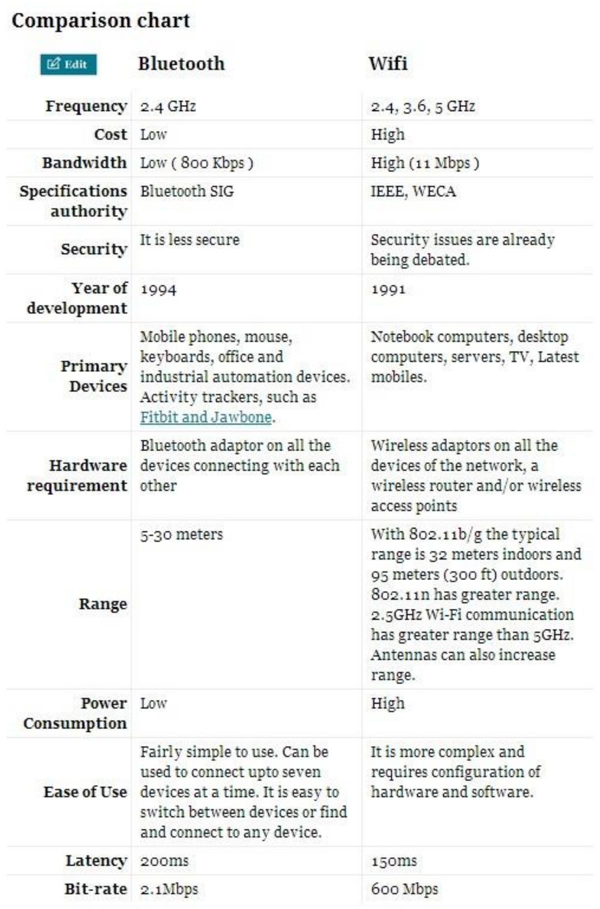
These weights implied that the Quality Characteristics should be ranked as follows:

1. Simulations on insta-car,
2. Addition of a "stop" button in the center of the application,
3. On board encoder facility on the motor controller,
4. SoftAP Wi-Fi network,
5. Addition of "Speed up" and "Speed down" buttons in the Android application,
6. Reaction time of compass animation, GPS tracking and command execution,
7. 5 speed levels for the Wheelchair to reach,
8. Addition of a circuit breaker switch,
9. Smooth Acceleration/Deceleration, amongst others

From the above rankings, one can observe that the simulations on insta-car was of utmost priority as it established the fact that our product is not only expansible but, also, safe with real-time response. Notice that according to the HOQ, the on board encoder facility on RoboClaw was suggested to be an excellent addition to the project for real time statistics to be recorded such as position and speed of the Wheelchair at any given time. However, it could not be implemented due to time constraints regarding manufacturing and shipping new motors with encoders attached to them. Thus, we focused on the other highest rankings, which resulted in selecting a Wi-Fi Module and Motor Controller based on their pros and cons in order to proceed further. However, other changes were based on upgrading the GUI to include the "stop", "speed up" and "speed down" buttons, integrating GPS tracking/navigation and compass animation in the application as well as implement a smooth acceleration/deceleration method.

The Comparisons/Pros and Cons for Wireless Communication and Motor+Micro Controller Options

As seen in Objective 1, the wireless communication options of Bluetooth and Wi-Fi were compared based on their feasibility in terms of cost, expansibility, computational overheads and safety. These comparisons can be seen in the table below:



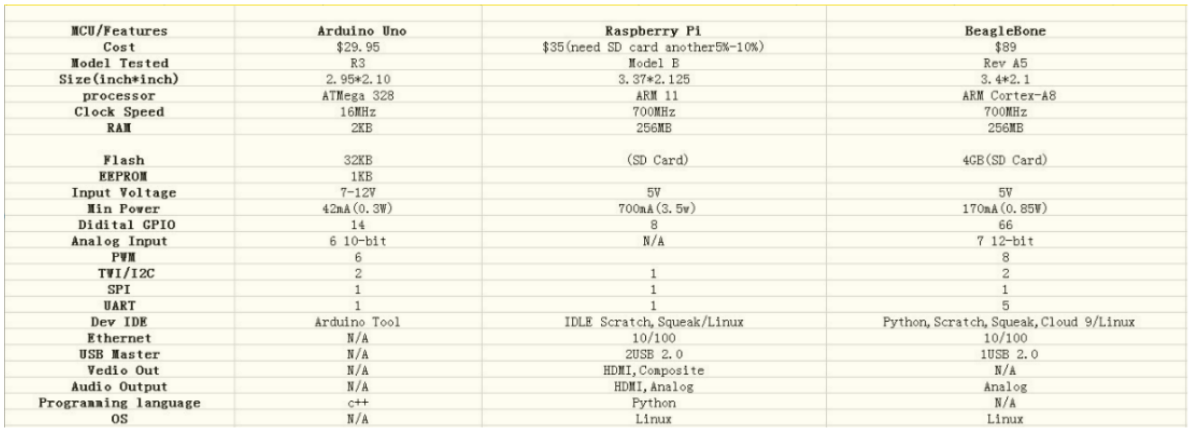
Comparison chart

	Bluetooth	Wifi
Frequency	2.4 GHz	2.4, 3.6, 5 GHz
Cost	Low	High
Bandwidth	Low (800 Kbps)	High (11 Mbps)
Specifications authority	Bluetooth SIG	IEEE, WECA
Security	It is less secure	Security issues are already being debated.
Year of development	1994	1991
Primary Devices	Mobile phones, mouse, keyboards, office and industrial automation devices. Activity trackers, such as Fitbit and Jawbone.	Notebook computers, desktop computers, servers, TV, Latest mobiles.
Hardware requirement	Bluetooth adaptor on all the devices connecting with each other	Wireless adaptors on all the devices of the network, a wireless router and/or wireless access points
Range	5-30 meters	With 802.11b/g the typical range is 32 meters indoors and 95 meters (300 ft) outdoors. 802.11n has greater range. 2.5GHz Wi-Fi communication has greater range than 5GHz. Antennas can also increase range.
Power Consumption	Low	High
Ease of Use	Fairly simple to use. Can be used to connect upto seven devices at a time. It is easy to switch between devices or find and connect to any device.	It is more complex and requires configuration of hardware and software.
Latency	200ms	150ms
Bit-rate	2.1Mbps	600 Mbps

Table 6: BLUETOOTH VS. WIFI COMPARISON [41]

Overall, Safety has the utmost priority, and since we cannot risk late response time when sending the command in cases of emergency, we chose to go with Wi-Fi communication, as its latency was much lower with a higher bitrate and bandwidth as compared to the Bluetooth communication. Another advantage was the range of communication which was also farther as compared to Bluetooth. Factors like ease of use did not matter to the end-user as it was meant for development purposes. Hence, overall, the Wi-Fi communication proved to be more advantageous and hence, we decided to replace the current Wi-Fi module with one that can use SoftAP network type. This was found to be a feature in the subsequent version of the old Wi-Fi Module, namely ST-MW-09S.

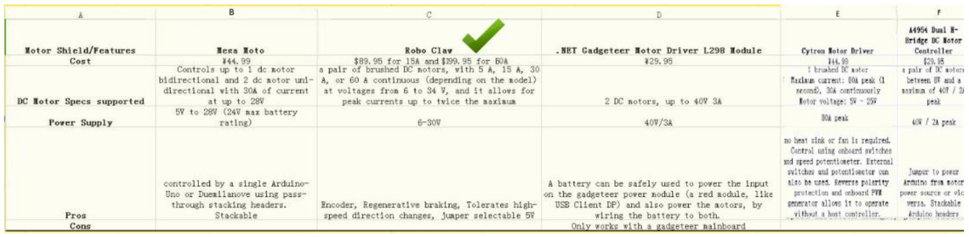
The microcontrollers were also compared, however, we did not choose any, because the ST-MW-09S Wi-Fi module was already equipped with a low-cost microcontroller (MCU), same as before, namely, STC89C52. In any case, following is the MCU Comparisons in a table format:



MCU/Features	Arduino Uno	Raspberry Pi	BeagleBone
Cost	$29.95	$35(need SD card another5%-10%)	$89
Model Tested	R3	Model B	Rev A5
Size(inch*inch)	2.95*2.10	3.37*2.125	3.4*2.1
processor	ATMega 328	ARM 11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(SD Card)
EEPROM	1KB		
Input Voltage	7-12V	5V	5V
Min Power	42mA(0.3W)	700mA(3.5w)	170mA(0.85W)
Didital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud 9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2USB 2.0	1USB 2.0
Vedio Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog
Programming language	c++	Python	N/A
OS	N/A	Linux	Linux

Table 7: MICROCONTROLLER COMPARISONS

For motor controller selection, at first, we looked at several of them, however we soon realized that the original motor controller used in the off-the-shelf electrically powered wheelchair was rated for 60A and hence, we quickly scratched off a lot of motor controllers that we were previously looking at. Following is the table describing the comparison between different motor controllers:



A	B	C	D	E	F
Motor Shield/Features	Mega Moto	Robo Claw ✓	.NET Gadgeteer Motor Driver L298 Module	Cytron Motor Driver	A4954 Dual H-Bridge DC Motor Controller
Cost	¥44.99	$89.95 for 15A and $199.95 for 60A	¥29.95	¥44.99	$29.95
DC Motor Specs supported	Controls up to 1 dc motor bidirectional and 2 dc motor uni-directional with 30A of current at up to 28V	a pair of brushed DC motors, with 5 A, 15 A, 30 A, or 60 A continuous (depending on the model) at voltages from 6 to 34 V, and it allows for peak currents up to twice the maximum	2 DC motors, up to 40V 3A	1 brushed DC motor Maximum current: 80A peak (1 second), 30A continuously Motor voltage: 5V - 25V	a pair of DC motors between 8V and a maximum of 40V / 2A peak
Power Supply	5V to 28V (24V max battery rating)	6-30V	40V/3A	80A peak	40V / 2A peak
Pros	controlled by a single Arduino-Uno or Duemilanove using pass-through stacking headers. Stackable	Encoder, Regenerative braking, Tolerates high-speed direction changes, jumper selectable 5V	A battery can be safely used to power the input on the gadgeteer power module (a red module, like USB Client DP) and also power the motors, by wiring the battery to both.	no heat sink or fan is required. Control using onboard switches and speed potentiometer. External switches and potentiometer can also be used. Reverse polarity protection and onboard PWM generator allows it to operate without a host controller.	Jumper to power Arduino from motor power source or vice versa. Stackable Arduino headers
Cons			Only works with a gadgeteer mainboard		

Figure 51: MOTOR CONTROLLER COMPARISONS

As safety was more important, we decided to purchase RoboClaw as it could handle up to 60A current as well as bi-directionally control two DC motors as compared to other motor controllers with low rating that could only handle 1 direction or 1 DC motor at a time, which implied that two of such controllers would have to be bought.

In the next section, we discuss our results from the upgraded GUI.

Upgraded GUI

As we scrutinized the previous GUI developed for the purpose of this project, we observed a lot of shortcomings in the user experience. Following is a side-by-side comparison of the old vs. new GUI:

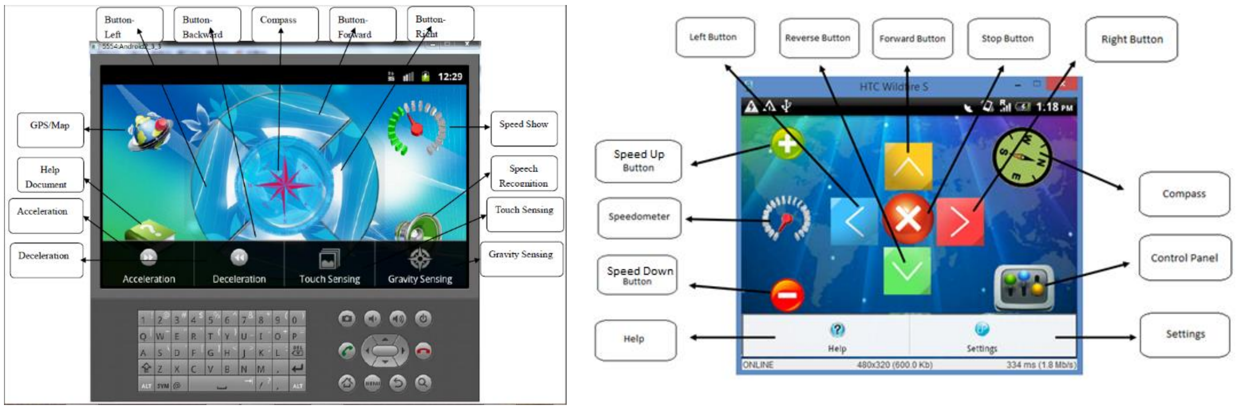


Figure 52: SIDE-BY-SIDE COMPARISON OF OLD VS. NEW GUI

In the old GUI, one can notice that the different multimodal control options were rather dispersed/de-centralized around the menu options and the main screen of the application with "Touch Sensing" and "Gravity Sensing" being hidden in the options menu while "GPS tracking" and "Voice Recognition" were found on the main screen of the application. Moreover, the acceleration/deceleration options were also hidden inside the menu options while the compass was hardly visible and would jitter from time to time. Also, the movement commands would not work until a control option was selected. Depending on the control option, the particular driving mode will then be set into action.

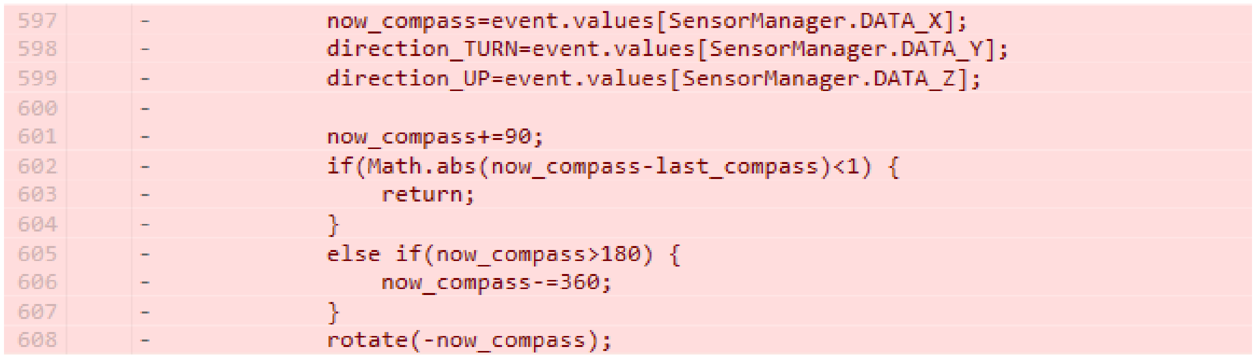
In the new GUI, all the control options were consolidated into the control panel on the bottom right. The "speed up" and "speed down" options, earlier incorrectly named as "acceleration" and "deceleration", were now situated on the main screen rather than the menu options to easily changing the speed. The incorrect naming of the multimodal control options were corrected and renamed as follows:

Modified Names of the Multimodal Control Options
Touch Sensing
Gravity Sensing
GPS Navigation
Gesture Control
Voice Control

Table 8: MODIFIED NAMES OF MULTIMODAL CONTROL OPTIONS

Also, the safety factor was taken into consideration and a stop button was physically added to the center of the application. The compass was now located at the top right corner for sensing the direction of movement at any given point of time. The animation of the compass was also changed and revised to be more real time.

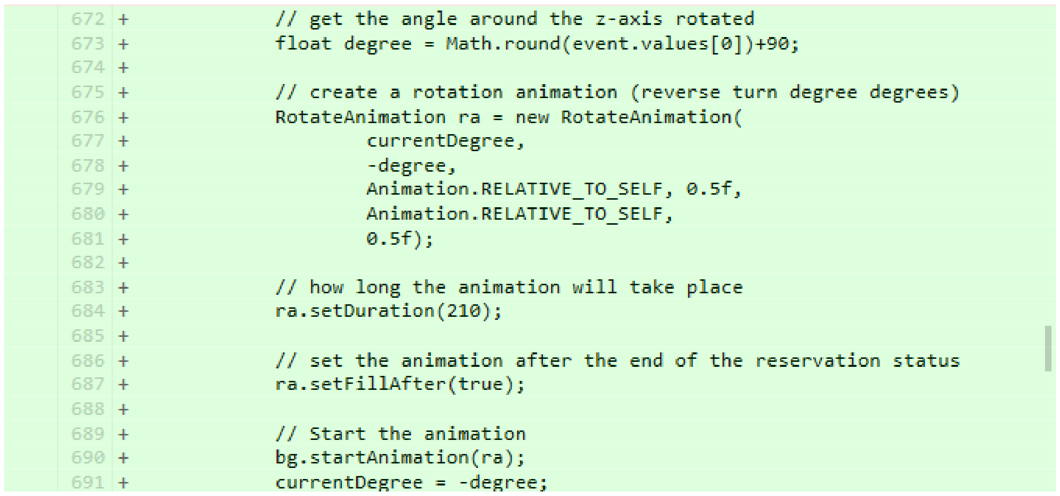
Following is a code snippet of the old animation code:



```
-              now_compass=event.values[SensorManager.DATA_X];
-              direction_TURN=event.values[SensorManager.DATA_Y];
-              direction_UP=event.values[SensorManager.DATA_Z];
-
-              now_compass+=90;
-              if(Math.abs(now_compass-last_compass)<1) {
-                  return;
-              }
-              else if(now_compass>180) {
-                  now_compass-=360;
-              }
-              rotate(-now_compass);
```

Figure 53: PREVIOUS COMPASS ANIMATION [41]

Here, the compass is only animated when a certain threshold was crossed. However, the revised animation of the compass can be seen in the following code snippet, which constantly animates the compass:



```
+           // get the angle around the z-axis rotated
+           float degree = Math.round(event.values[0])+90;
+
+           // create a rotation animation (reverse turn degree degrees)
+           RotateAnimation ra = new RotateAnimation(
+                   currentDegree,
+                   -degree,
+                   Animation.RELATIVE_TO_SELF, 0.5f,
+                   Animation.RELATIVE_TO_SELF,
+                   0.5f);
+
+           // how long the animation will take place
+           ra.setDuration(210);
+
+           // set the animation after the end of the reservation status
+           ra.setFillAfter(true);
+
+           // Start the animation
+           bg.startAnimation(ra);
+           currentDegree = -degree;
```

Figure 54: NEW COMPASS ANIMATION [41]

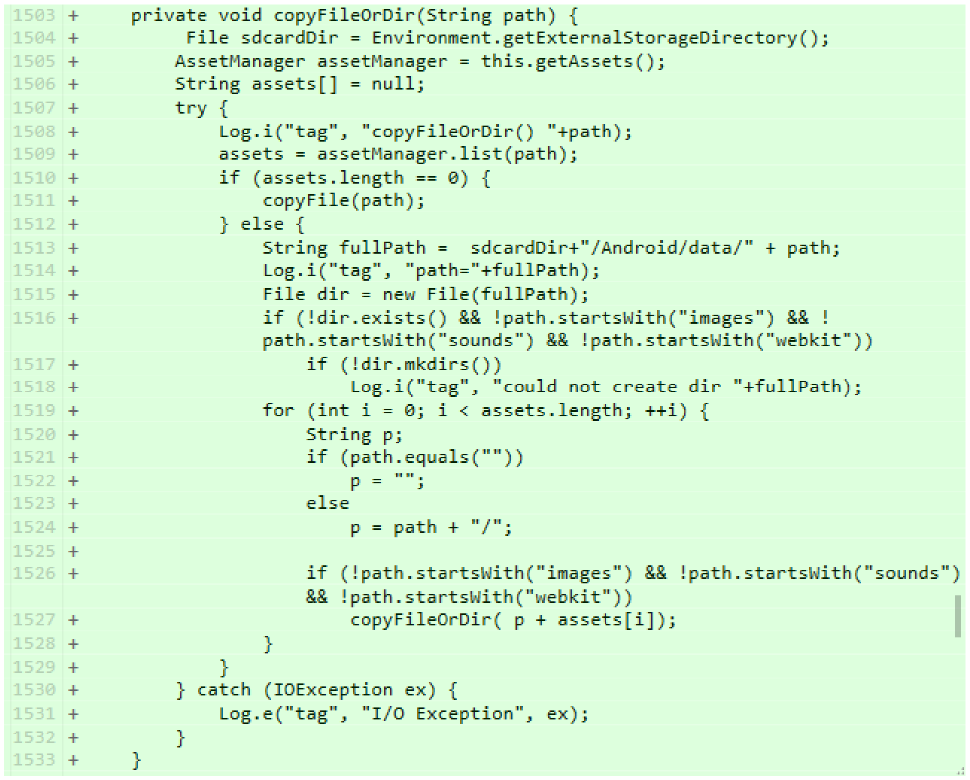
Here, a RotateAnimation() method handles the animation of the compass by keeping track of the currentDegree of rotation, thereby relatively changing the animation, hence getting rid of the jittery motion of the previous compass implementation.

The offline voice recognition database was also revised to include “Speed Down” as well as “Slow Down” commands to be recognized to allow users the flexibility to use either phrase to control the wheelchair. This can be observed in the new database given below:

New Offline Voice Recognition Database
Forward
Backward
Left
Right
Stop
Speed up
“Slow down”/“Speed down”

Figure 55: NEW OFFLINE VOICE RECOGNITION DATABASE

Also, the following method copyFileOrDir() was added to the Android application to download the voice packages required to run the offline voice recognition at the start of the application:



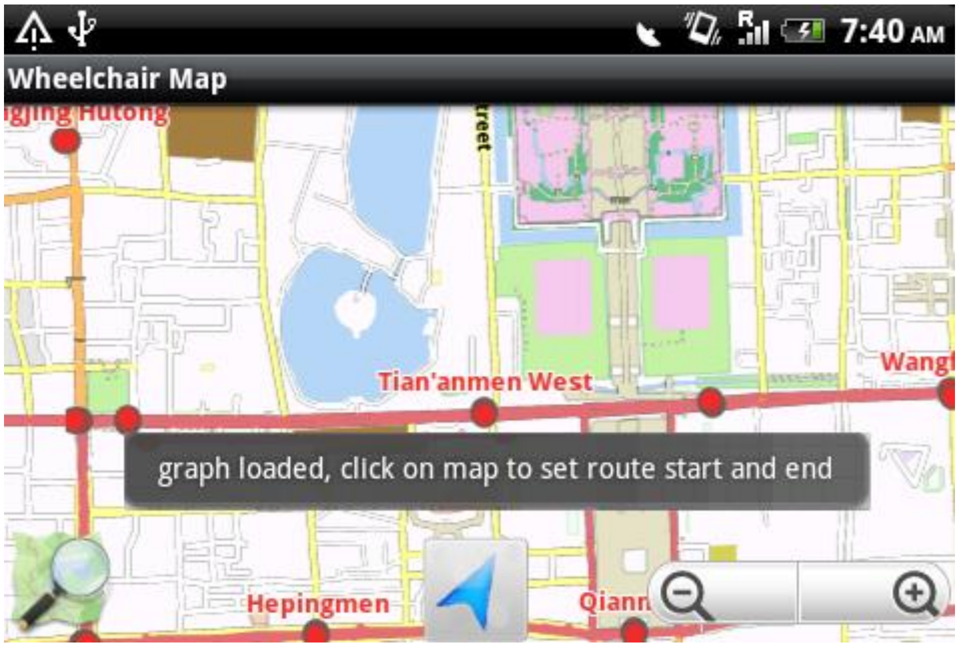
```
+    private void copyFileOrDir(String path) {
+         File sdcardDir = Environment.getExternalStorageDirectory();
+        AssetManager assetManager = this.getAssets();
+        String assets[] = null;
+        try {
+            Log.i("tag", "copyFileOrDir() "+path);
+            assets = assetManager.list(path);
+            if (assets.length == 0) {
+                copyFile(path);
+            } else {
+                String fullPath =  sdcardDir+"/Android/data/" + path;
+                Log.i("tag", "path="+fullPath);
+                File dir = new File(fullPath);
+                if (!dir.exists() && !path.startsWith("images") && !
                 path.startsWith("sounds") && !path.startsWith("webkit"))
+                    if (!dir.mkdirs())
+                        Log.i("tag", "could not create dir "+fullPath);
+                for (int i = 0; i < assets.length; ++i) {
+                    String p;
+                    if (path.equals(""))
+                        p = "";
+                    else
+                        p = path + "/";
+
+                    if (!path.startsWith("images") && !path.startsWith("sounds")
                     && !path.startsWith("webkit"))
+                        copyFileOrDir( p + assets[i]);
+                }
+            }
+        } catch (IOException ex) {
+            Log.e("tag", "I/O Exception", ex);
+        }
+    }
```

Figure 56: METHOD TO COPY FILES/FOLDERS FROM APPLICATION TO EXTERNAL STORAGE DEVICE

This ensured that the packages were downloaded automatically instead of necessitating the user to download them manually onto their SD Card. Thus, making it easier for the user to use the application in a “plug-and-play” manner.

GPS Navigation

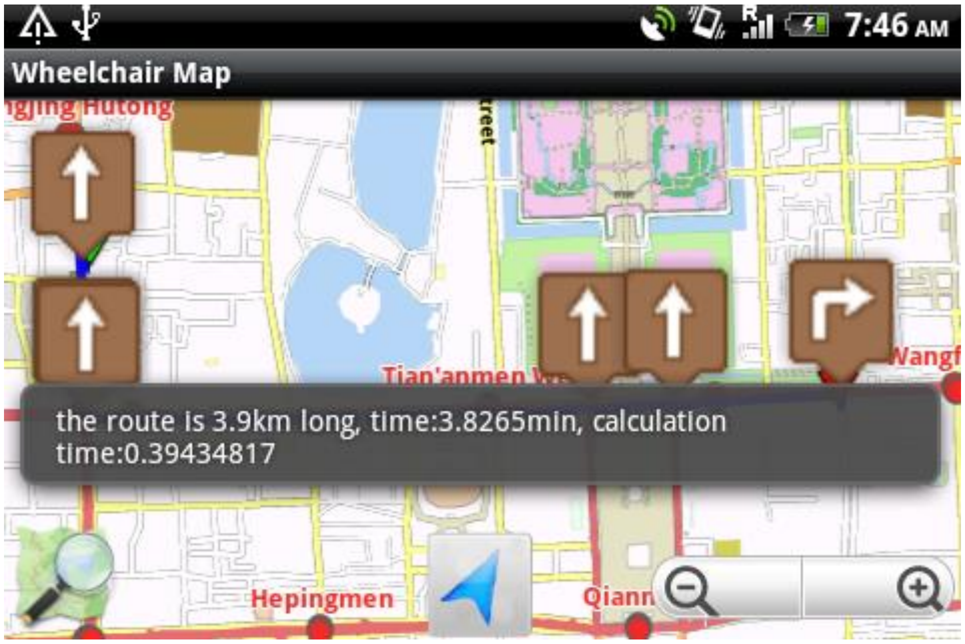
The GPS navigation, which was implemented using third-party software successfully loaded the maps from the external storage device, as seen below:



graph loaded, click on map to set route start and end

Figure 57: GRAPH SUCCESSFULLY LOADED

Once, loaded, the graph can be routed by clicking at any two points on the map; a result of this routing can be seen below:



the route is 3.9km long, time:3.8265min, calculation time:0.39434817

Figure 58: ROUTING AROUND TIAN'ANMEN SQUARE

It can be observed from the above Figure 58 that the routing from the green marker at the top left to the red marker in the center right is covered with the direction markers at every intersection. Thus, it can be deduced from the above figure, that the GraphHopper engine, used to generate this route, also returns the length of the route and the time it takes to get there. This information can also be used by a future developer to add more features to this type of navigation if needed. The blue navigation button in the bottom center is used to locate oneself on the map, using reverse geocoding just like the previous application.

Thus, the previous functionality of just GPS tracking the current location on an online map, has been transformed into a GPS navigation feature, which can not only show you the current location on an offline map, but, also help navigate from one point to another. The only shortcoming of this feature is that it is computationally more expensive and that the maps have to be downloaded manually onto the Android application.

New Wi-Fi Network

After we switched to a new version of the Wi-Fi module that used SoftAP network type, the power consumption reduced than the previous version and the transmission distance also increased than before. The key point is that the Android phones can connect to the Wi-Fi module directly. This improvement saves a lot of unnecessary processes for users to go through before controlling the wheelchair.

Following is the screenshot that shows the ST-MW-09S Wi-Fi connection being displayed in the Android cellphones without having to change the internal settings of the phone.

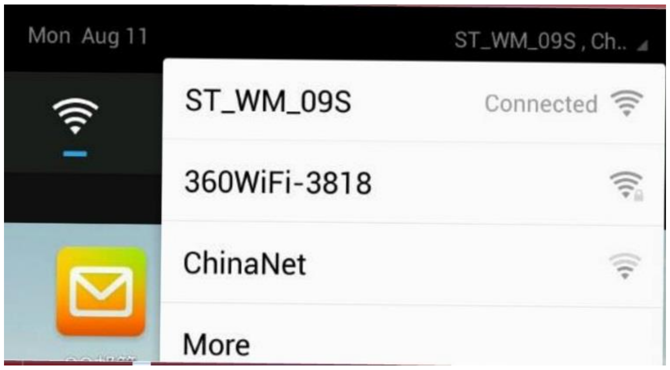

Mon Aug 11
ST_WM_09S , Ch..
ST_WM_09S Connected
360WiFi-3818
ChinaNet
More

Figure 59: ST-MW-09S AS SEEN IN THE AVAILABLE WI-FI CONNECTIONS

Speed Regulation System

The physical circuit of the speed regulation system which uses, RoboClaw as the motor controller and digital potentiometers to interface with the Wi-Fi module through the motor controller can be observed in the following image:

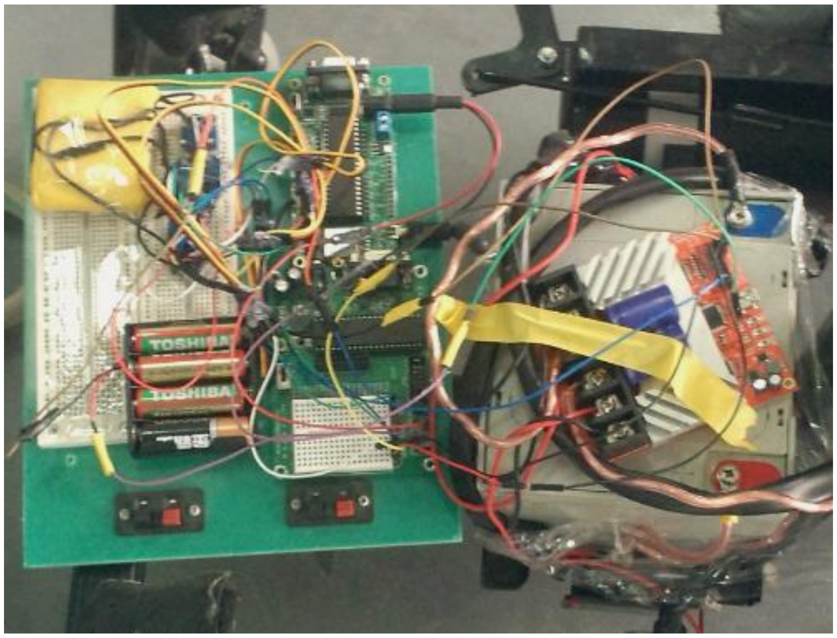


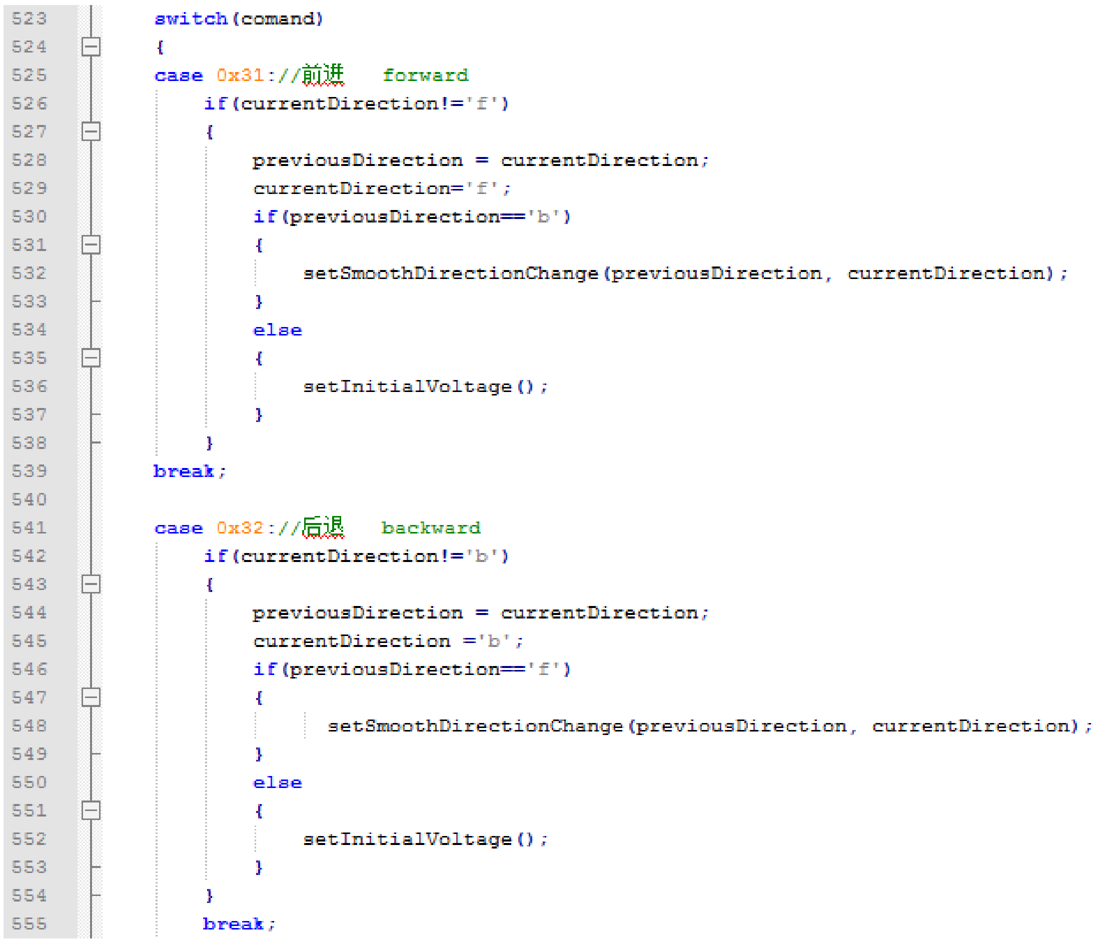
Figure 60: PHYSICAL MOTOR CONTROLLER CIRCUIT

There was a problem with the power supply, such that both the Wi-Fi module, STMW09S, and the motor controller, RoboClaw, were sensitive to fluctuations in the power supply. So, their logic and main motor power supply had to be kept separate. The logic voltage had to be 5.3 volts to be enough to power up the RoboClaw, so we used separate independent batteries for supplying to the RoboClaw which is the yellow wrap package shown in the Figure 60 above. Another set of separate batteries were introduced as the power supply for the logic circuits of the Wi-Fi module: STMW09S and the microcontroller: STC89C52.

Subsequently, the Wi-Fi module connected to the microcontroller would receive the direction and speed change commands from the Android device and the hexadecimal values corresponding to the command were then mapped to voltage values for the signals to the motor controller. Following which, the motor controller would receive these voltage signals, interpret the values for speed and direction changes and control the motors accordingly.

When turning left or right, the wheelchair would only turn for three seconds to ensure safety of the user on board and not making the user dizzy with complete turns. Also, the circuit breaker switch installed as a physical “stopping” mechanism came in handy when during testing, the user mounted off from the wheelchair, which caused a sudden acceleration due to loss of weight. In such a case, having a physical circuit breaker switch helps to avoid accidents.

Also, given that the circuit could reach 5 different speed levels, the need for smooth directional changes and smooth acceleration and deceleration was necessary. Hence, to be computationally inexpensive while writing the code, the condition of whether the smooth change was required or not was checked within the switch case as seen in the following code snippet:



```
switch(comand)
{
case 0x31://前进   forward
    if(currentDirection!='f')
    {
        previousDirection = currentDirection;
        currentDirection='f';
        if(previousDirection=='b')
        {
            setSmoothDirectionChange(previousDirection, currentDirection);
        }
        else
        {
            setInitialVoltage();
        }
    }
break;

case 0x32://后退   backward
    if(currentDirection!='b')
    {
        previousDirection = currentDirection;
        currentDirection ='b';
        if(previousDirection=='f')
        {
              setSmoothDirectionChange(previousDirection, currentDirection);
        }
        else
        {
            setInitialVoltage();
        }
    }
    break;
```

Figure 61: ENSURE SMOOTH CHANGE IS NECESSARY

Here, on line 530, one can notice that the previous direction is checked against the current direction to make sure that the smooth direction change only occurs if the previous direction was significant enough to use smooth direction change. Therefore, for cases where the user presses the “backward” or “forward” button in twice in a row, no action is taken, thereby making the program less computationally expensive.

Overall, the speed regulation system with its smooth acceleration can help users to ride over speed bumps in both directions (forward and backward) fairly easily. In Figure 62 below, one can observe the application performing the acceleration in both forward and backward directions to ride over the speed bumps.

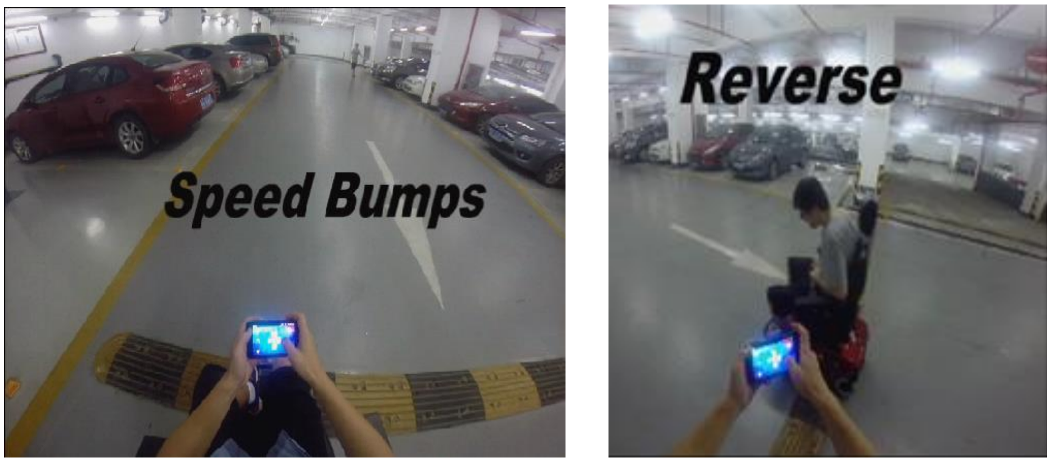


Figure 62: RIDING OVER SPEED BUMPS IN BOTH DIRECTIONS

The same applied to deceleration when trying to avoid an obstacle. Additionally, the real time reaction of the commands sent from the Android Application and executed at the RoboClaw was observed while testing the Wheelchair platform. Below is a snapshot of the real-time response of the application that occurred:

Figure 63: REAL TIME REACTION TO AVOID OBSTACLES

In order to avoid the obstacle, which, in this case, is the brown car to the left the commands were rapidly sent and the wheelchair was able to stop right in time before the collission.

Simulations on Insta-Car

We finally transplanted the control system of wheelchair to our own product, Insta-car. And we have done all the developed features on the Insta-car through our Android program. The only difference between the wheelchair and the Insta-car is the rating of the motors, however, they all used DC motors. This implies that any robot with two DC motors can be controlled with the same Android application as long as it uses the same motherboard circuit that was used for both the Wheelchair and the insta-car.

The Insta-car proves this expandability of our Android program and the motherboard circuit, providing a much more convenient way for students to develop more features on this smaller educational platform in the future.

Evaluation of the Improved Prototype Design

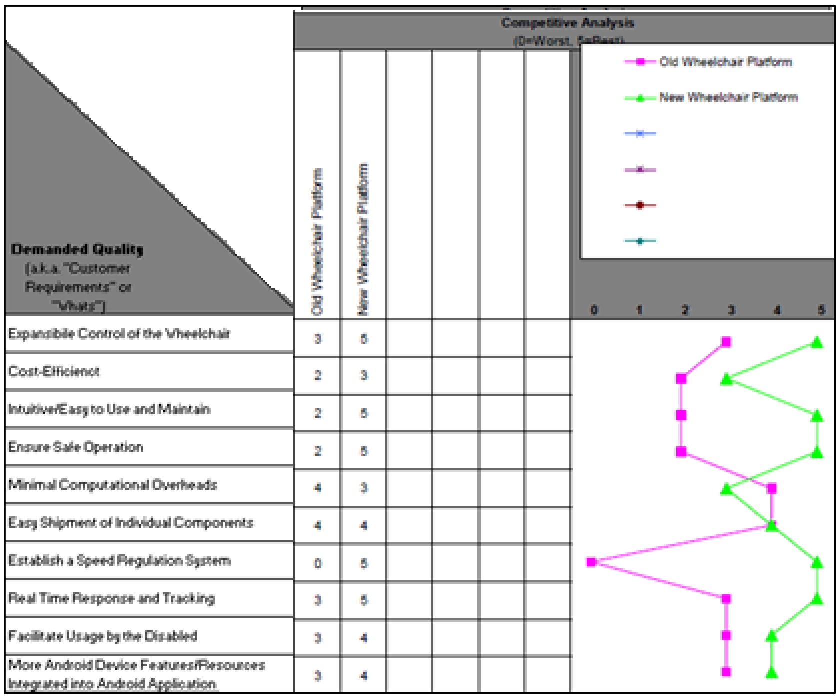
Following is the Table 4 from Objective 1 that described the issues and problems that the Wheelchair had except that the third column now explains what and how the problems were fixed if they were. If not, then possible solutions are noted down for future reference.

Issues/Problems	Type	Solution
Power drops in the circuit	**Hardware**	Separated the batteries for the logic and the motors
Wires cannot withstand high current and smoke all the time	**Hardware**	Used thicker wires
Lack of a speed regulation system	**Hardware**	Designed a motor controller circuit for bidirectional speed control of both the DC motors
Ad-hoc network unable to be detected by most Android phones	**Hardware**	Replaced Wi-Fi module to use SoftAP network
Manual installation for offline voice recognition	**Software**	The required packages are now automatically downloaded
Lack of offline GPS navigation	**Software**	Used third-party software, GraphHopper and Nutiteq to achieve Offline GPS Navigation
Gravity Sensing displays an approximation of distance	**Hardware/Software**	Not yet solved Possible Solution: Manufacture motors fitted with encoders
Cameras not yet used	**Software**	Not yet solved Possible Solution: Eye tracking/Path Detection

Emergency SMS threshold set to shaking the phone	**Hardware**	Not yet solved <u>Possible Solution</u>: Attach a cliff/bumper/infrared sensor
Non-intuitive application	**Software**	Consolidated/Localized the Application features for ease of use and intuitive navigation
Lack of user safety for emergency situations	**Hardware**	Added a circuit breaker switch to protect the circuit from overloads and for emergency usage

Figure 64: SOLUTIONS TO THE ISSUES FACED IN THE PREVIOUS WHEELCHAIR PLATFORM

Finally, the rightmost part of the HOQ, namely, "Competitive Analysis" included in the "House of Quality" section above, was used to model the comparison between the previous wheelchair architecture and the current wheelchair framework. Following is the fragment from the HOQ:



Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Old Wheelchair Platform	New Wheelchair Platform				
Expandibile Control of the Wheelchair	3	5				
Cost-Efficienct	2	3				
Intuitive/Easy to Use and Maintain	2	5				
Ensure Safe Operation	2	5				
Minimal Computational Overheads	4	3				
Easy Shipment of Individual Components	4	4				
Establish a Speed Regulation System	0	5				
Real Time Response and Tracking	3	5				
Facilitate Usage by the Disabled	3	4				
More Android Device Features/Resources Integrated into Android Application	3	4				

Figure 65: COMPETITIVE ANALYSIS

After self-scoring each wheelchair architecture by deduction from the results, we can observe that the new architecture scores higher than the previous architecture in various “Demanded Qualities” except in the quality of “minimal computational overheads”. This is due to the addition of the automatic download of offline voice recognition packages and the offline GPS navigation features, which, not only caused the computational overheads to increase but also the memory requirements to hike up. The fully functional application .apk file, along with the automatic downloads of the offline voice recognition package and the routing engine for GPS navigation, now takes up about 21.42MB of storage space. In any case, the new wheelchair architecture outperforms the old one, in terms of the other demanded qualities.

Chapter 5: Conclusion and Recommendations

In conclusion, the wheelchair platform was to be used in combination with a smartphone/tablet not only because mobile devices are becoming commonplace in the contemporary world, but, also to distribute the workload and expand the controllability of the wheelchair itself.

Ultimately, we expect to let users be able to adjust the speed of the wheelchair, while sitting on it, with the addition of a speed regulation system. This system was based on switching to a much more powerful motor controller than using the relay circuit design of the previous architecture. This decision was made after analyzing the pros and cons of each motor controller available in the market that suited the two of our 24V DC motor ratings. Finally, we were able to implement five speed gears on the wheelchair, such that one can speed up or down using the Android application. Thereby, smoothly increasing or decreasing the voltage of the motor, brings into effect, the smooth acceleration and deceleration during the movements.

Also we expect to let every new Android phone be able to connect to the wheelchair without changing its internal settings. Since, we upgraded the Wi-Fi module from ST-MW-08S to ST-MW-09S after we researched all the options for the wireless communication, we can now actually connect to every Android phone seamlessly and then directly control the wheelchair.

Fundamentally, the expansion of the circuitry and software application to the insta-car was just an icing on the cake. This implies that our design of the module can be applied to other Robotic products produced by the sponsor, which makes use of two brushed DC motors rated within the RoboClaw's capacity. Our design ensures that as long as the circuit that we built can be consolidated onto the body of such a product, it can be teleoperated with bidirectional control and speed regulation of both its motors.

Not to mention, the upgraded "look and feel" of the new GUI that is easy and intuitive to navigate and use. Despite the fact that additional Android features go hand in hand with memory and computationally intensive applications, there are various ways to further distribute the workload on the cellphone itself. We will be discussing these distributive ideas and more in the following section.

Future Work:

Advanced Multimodal Control Method

There are still different ways available to control the wheelchair, which can be adapted to this project. Users can possibly be able to control the wheelchair through more advanced eye tracking, path detection and sip-and-puff(SNP) control [2].

With the development of multi-modal control of a wheelchair, it is possible for us to add more functionalities/features to our Android application. Recently, WPI developed a semi-autonomous wheelchair with multi-modal control interface, with cliff detection and emotive control of the wheelchair [42]. This type of multimodal control can be extended to Android cell phones and their features/resources. One such example is shown in the Figure 66 below:

Figure 66: POSSIBLE CAMERA INTEGRATION

The eye-tracking function, as mentioned in Chapter 2: Literature Review, enables users to possibly control the wheelchair by tracking the eye ball movement, which can be further developed into a database of commands formed by a series of eye movements and blinking. For this purpose, the front cameras of the Android cellphones can be used to track the user's eye.

The back cameras can also be used to detect a clear path while navigating the wheelchair adding onto the autonomy of the wheelchair as a robotics educational platform.

Further Wi-Fi Optimization

There is also a possible way to use a third-party Wi-Fi server as a hub for even several wheelchairs and Android phones at the same time. The Figure 67 below shows the communication between the different components of the architecture with the help of a central hub:

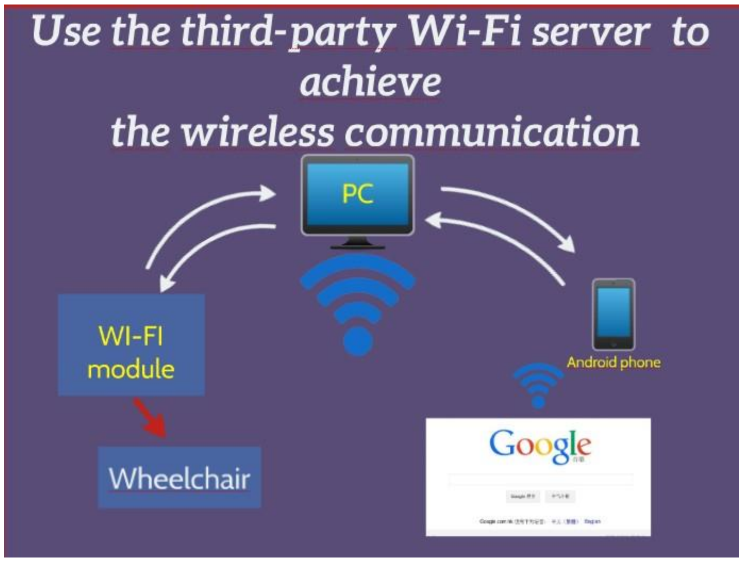

Use the third-party Wi-Fi server to achieve
the wireless communication
PC
WI-FI module
Wheelchair
Android phone
Google

Figure 67: POSSIBLE THIRD PARTY WI-FI SERVER HUB

There are several advantage of this new wireless communication method in that:

- The user can access the online features of the Android device when connected to a central hub,
- A server on this hub can share the load of the Android devices and make it less computationally and memory intensive

This uses the same principle as a router with an internet connection, but also enables control of the wheelchair at the same time. Furthermore, users can use their phones to browse internet.

There is always restrictions when the program can only have offline functionalities in the future, but with the internet connection, cloud services can also be used to further decrease the overheads associated with larger programs. There are several more applications that can make use of such architecture.

This has an interesting application in old-age homes, where several elderly people are restricted to use their wheelchairs, but, all of these wheelchair can be coordinated with the theme of multi-robot systems and controlled by a central server, such as a computer. For instance, commands can be sent to several wheelchairs at the same time to bring everyone together for dinner.

Adding Sensors to Detect Obstacles

As mentioned in the advanced multi-modal control method section above, the wheelchair can be interfaced with additional multimodal control options. Also, more sensors can be added to the wheelchair to improve the safety of the users.

The sip-and-puff controlling method is also considerable to be added to the Android program, the cell phone can be plugged in with assisted device as airflow sensor. Then users can control the wheelchair by blowing or sipping. Following is an infographic about the advantages of adding additional sensors:

Apply better threshold for sending emergency messages
SMS
STOP
Use Bumper Sensors and/or Infra-red sensors to detect obstacles

Figure 68: BETTER THRESHOLD FOR CRASH/OBSTACLE DETECTION

The bumper sensors and infra-red sensors are all possible additions to the wheelchair platform to detect obstacles in its path. Once the wheelchair reaches the cliff or an obstacles, the program will either stop the wheelchair or navigate around the obstacle automatically as well as notify the user about the situation. In case, the wheelchair does get into an accident, these sensors can be made to trigger an alarm and send SMS notification to their loved ones, along with the GPS location of the incident.

Use Encoder-Fitted Motors

Last, but, not least, the RoboClaw's excellent facility of encoder usability can be implemented in the future if customized encoder-fitted 24V DC motors can be ordered from the motor manufacturers. Such motors can be seen in the following infographic:

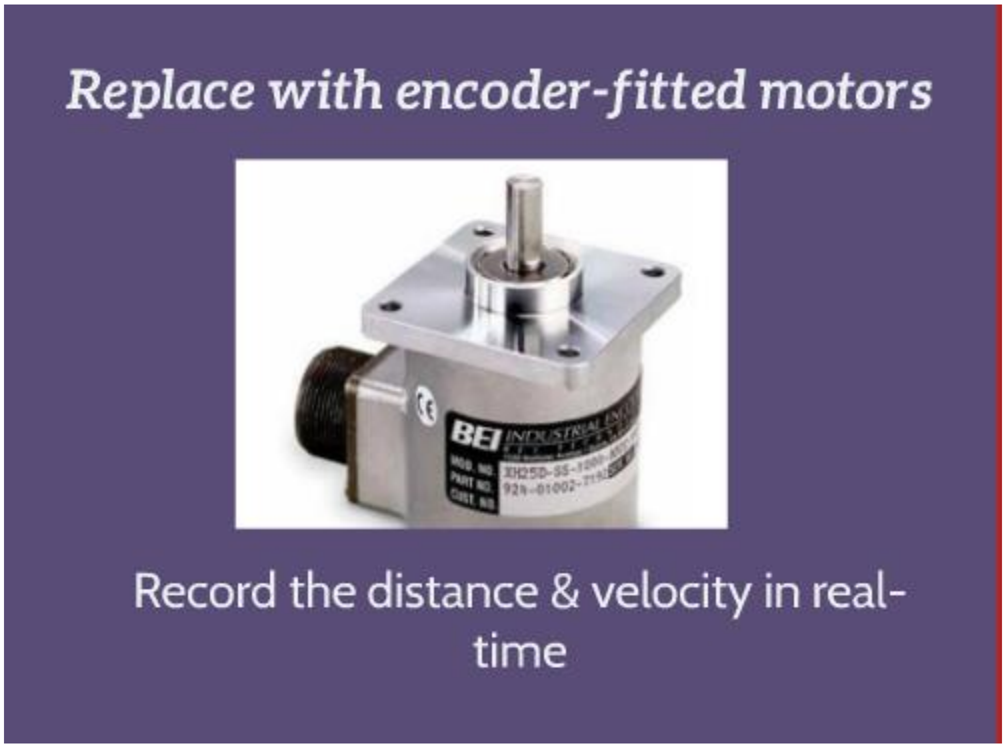

Replace with encoder-fitted motors
Record the distance & velocity in real-time

Figure 69: REPLACE WITH ENCODER-FITTED MOTORS

This would enable the real-time recording of data which is more accurate than the accelerometer sensors on board the Android smartphones. The distance and velocity calculated this way would establish a far greater accuracy in statistics of usage than without such facility. This would also result in a closed loop control of the wheelchair, thereby making the movements of the wheelchair more precise than before. This closed loop architecture can be seen in the following Figure 70:

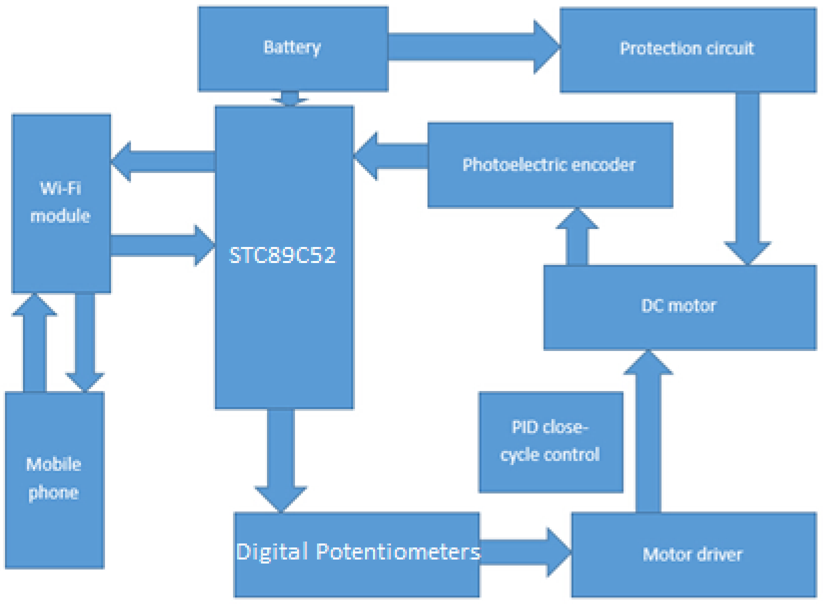


Figure 70: CLOSED-LOOP CONTROL OF THE WHEELCHAIR

Finally, the closed-loop control of the wheelchair depicted above can make use of Proportional-Integral-Derivate(PID) Control to correct its position through error calculation using the feedback from the encoder position.

Conclusively, this project has major implications in the field of Educational Robotics (ER), as it presents to students a real world meaningful problem of limited options available for the disabled to interface with their wheelchairs. ER, thus, forms the basis of developing practical engineering projects like the Wi-Fi based mobile teleoperated wheelchair that is controlled by Android smartphones. Ultimately, our sponsor at DEPUSH Technology Co. Ltd. can greatly benefit from our recommendations for the future work by bringing this wheelchair into the educational market for project research and further development of the platform.

Bibliography

[1] B. Michael, "Robotic Revolution," vol. 46, ed. New York: Media Source, 2000, p. 83.

[2] "HowStuffWorks "Shepherd Center Profile"," 2014.

[3] (2014). *深圳市德普施科技有限公司_百度百科*. Available: http://baike.baidu.com/view/6802453.htm?fr=aladdin

[4] 武汉网讯互联(www.027DNS.Net), "德普施科技， DRLab， DRRob， 机器人， 教育机器人， DRVI， 电工电子，传感器," 2014.

[5] I. Gaudiello and E. Zibetti, "La robotique éducationnelle : état des lieux et perspectives," *Psychologie Française,* vol. 58, pp. 17-40, 3// 2013.

[6] T. A. Mikropoulos and I. Bellou, "Educational robotics as mindtools," *Themes in Science and Technology Education,* vol. 6, pp. pp. 5-14, 2013.

[7] C. Ngome, "The Media in Education," ed: Edinburgh University Press, 2009.

[8] G. Sven, "China in Smartphone Lead," in *Wall Street Journal (Online) U6 - ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle=China+in+Smartphone+Lead&rft.jtitle=Wall+Street+Journal+%28Online%29&rft.au=Sven+Grundberg&rft.date=2011-11-24&rft.pub=Dow+Jones+%26+Company+Inc&rft.externalDocID=2518429851&paramdict=en-US U7 - Newspaper Article U8 - FETCH-proquest_dll_25184298511*, ed. New York, N.Y: Dow Jones & Company Inc, 2011.

[9] Z. Eric. (2012, Android Drives China Smartphone Adoption Past U.S. *(Journal, Electronic)*. Available: http://wpi.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwTZ09CgJBDEaDeAJBsdwLDEzmf2px8QB7gcwkKcX7V2ZFUNKmzvtekw9gqSVP5Sy7GogFhjlsEnH32jh9Hvr8GiL-rvl6goM8z7Ct9-32cN8yAPcywjjD6pSmUpGMcDxyYCxTfTLcc9OeKkVj2UiqMyOFZOZDRUn69EFGwQsczaflCotts3LASDtOLQEZkgdiCZ1j4UhveJ0vPA

[10] R. Murphy, T. Nomura, A. Billard, and J. Burke, "Human-Robot Interaction," *IEEE Robotics&Automation Magazine,* vol. 17, pp. 85-89, 2010.

[11] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury, "Playing it safe [human-friendly robots]," *IEEE Robotics & Automation Magazine,* vol. 11, p. 12, 2004.

[12] D. Sidobre, X. Broquère, J. Mainprice, E. Burattini, A. Finzi, S. Rossi*, et al.*, "Human–Robot Interaction." vol. 80, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 123-172.

[13] W. Karwowski and M. Rahimi, *Human-robot interaction*. London: CRC Press Inc, 1992.

[14] "Android smartphones rule in China," ed. Shanghai: SinoMedia, 2010.

[15] (2014). *Application Fundamentals | Android Developers*. Available: http://developer.android.com/guide/components/fundamentals.html

[16] (2014). *Activities | Android Developers*. Available: http://developer.android.com/guide/components/activities.html

[17] I. G. Clifton and O. Safari Books, *Android user interface design: turning ideas and sketches into beautifully designed apps*. Upper Saddle River, NJ: Addison-Wesley, 2013.

[18] (2014). *Providing Resources | Android Developers*. Available: http://developer.android.com/guide/topics/resources/providing-resources.html

[19] "CHINA Google to Partner With: FINAL Edition," in *The Washington Post*, ed. Washington, D.C: WP Company LLC d/b/a The Washington Post, 2007, p. D.5.

[20] "Google blocked in China," in *Investor's Business Daily*, ed. Los Angeles: Investor's Business Daily, Inc, 2012, p. A2.

[21] "Google blocked again in China," in *Progressive Digital Media Technology News U6 - ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle=Google+blocked+again+in+China&rft.jtitle=Progressive+Digital+Media+Technology+News&rft.date=2014-07-11&rft.pub=Progressive+Digital+Media&rft.externalDocID=3371579711&paramdict=en-US U7 - Newspaper Article U8 - FETCH-proquest_dll_33715797111*, ed. London: Progressive Digital Media, 2014.

[22] "bytedeco/javacv," 2014.

[23] (2014). *Vision-based clear path detection through the use of single in-vehicle camera*. Available: http://dl.acm.org/citation.cfm?id=2521012

[24] "FAST ROAD/PATH DETECTION USING OPENCV," 2014.

[25] T. Hume, "Simple, accurate eye center tracking in OpenCV - Tristan Hume," 2014.

[26] (2014). *TiBa11b.pdf*. Available: http://www.inb.uni-luebeck.de/publikationen/pdfs/TiBa11b.pdf

[27] "graphhopper/graphhopper," 2014.

[28] *Bluetooth*. Available: http://en.wikipedia.org/wiki/Bluetooth

[29] *Wi-Fi*. Available: http://en.wikipedia.org/wiki/Wi-Fi

[30] *SoftAP*. Available: http://en.wikipedia.org/wiki/SoftAP

[31] Q. S. Ge Fang, Youni Wu, "Wi-Fi based mobile chair," Aug.17 2013.

[32] *Transmission Control Protocol*. Available: http://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_over_wireless_networks

[33] (2014). *Wheelchair, View power wheelchair, wisking Product Details from Shanghai Wisking Electric Machine Co., Ltd. on Alibaba.com*. Available: http://wisking.en.alibaba.com/product/51470201-50016013/Wheelchair.html

[34] (2014). *Rooting (Android OS) - Wikipedia, the free encyclopedia*. Available: http://en.wikipedia.org/wiki/Rooting_(Android_OS)

[35] *Tutorial of installing ad hoc patch*. Available: http://www.wifigx.com/android_adhoc/

[36] *Wireless ad hoc network*. Available: http://en.wikipedia.org/wiki/Wireless_ad_hoc_network

[37] *IEEE 802.11*. Available: http://en.wikipedia.org/wiki/IEEE_802.11

[38] "cmusphinx/sphinx4," 2014.

[39] "The house of quality," Jan 29, 1998.

[40] "QFD Online," 2014.

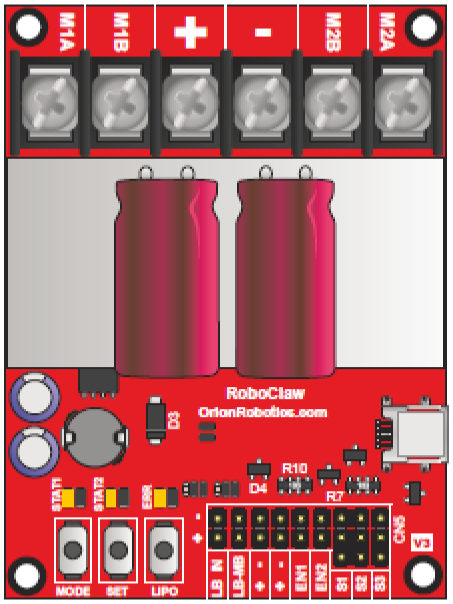
[41] "pelferina/wheelchair," 2014.

[42] (1/29/). *MCI_Wheelchair_Final_Report.pdf*. Available: http://www.wpi.edu/Pubs/E-project/Available/E-project-013014-131838/unrestricted/MCI_Wheelchair_Final_Report.pdf

Appendix A: RoboClaw Specifications

Key Specifications

- 2 Channels at 60Amp each, Peak 120Amp
- 3.3V Compliant Outputs
- 5V Tolerant Inputs
- Battery Elimination Circuit (BEC)
- Switching Mode BEC
- Hobby RC Radio Compatible
- Serial Modes
- TTL Input
- Analog Mode
- 2 Channel Quadrature Decoding
- Thermal Protection
- Lithium Cut Off
- Packet Serial with Error Detection
- High Speed Direction Switching
- Flip Over Switch
- Over Current Protection
- Regenerative Braking
- USB Capable(Optional)



Hardware Overview

A: Heat Sink

B: Power Stabilizers

C: Main Battery Input

D: Motor Channel 1

E: Motor Channel 2

F: BEC 3A Circuit

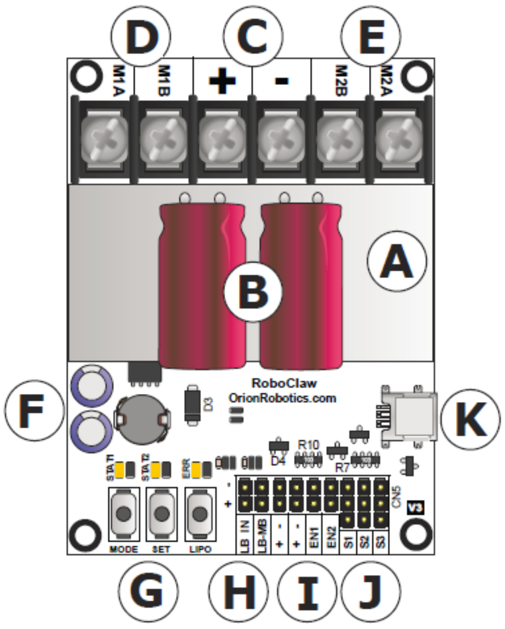
G: Setup Buttons

H: Logic Voltage Source Selection Header

I: Encoder Inputs

J: Controller Inputs

K: USB Connector - MiniB (Optional)

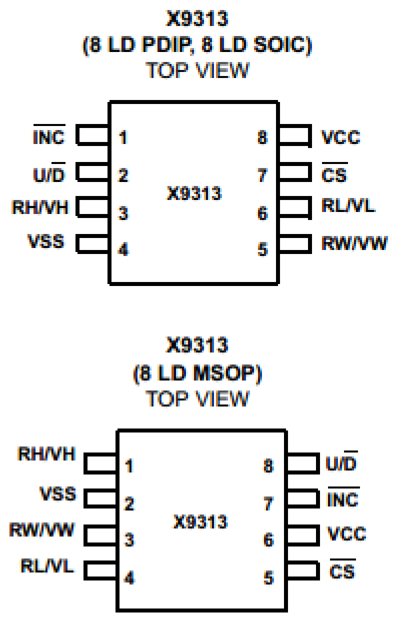


Appendix B: X9313 Digital Potentiometer Specifications

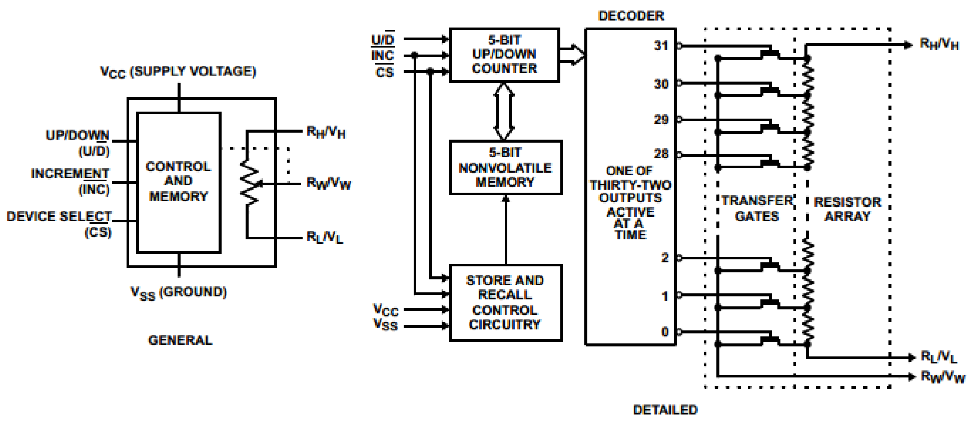
Key Features

- Solid-state potentiometer
- 3-wire serial interface
- 32 wiper tap points
  - Wiper position stored in nonvolatile memory and recalled on power-up
- 31 resistive elements
  - Temperature compensated
  - End-to-end resistance range ±20%
  - Terminal voltages, -VCC to +VCC
- Low power CMOS
  - VCC = 3V or 5V
  - Active current, 3mA max.
  - Standby current, 500µA max.
- High reliability
  - Endurance, 100,000 data changes per bit
  - Register data retention, 100 years
- RTOTAL values = 1kΩ, 10kΩ, 50kΩ
- Packages - 8 Ld SOIC, 8 Ld MSOP and 8 Ld PDIP
- Pb-free available (RoHS compliant)

Pinouts



Block Diagram


VCC (SUPPLY VOLTAGE)
UP/DOWN (U/D)
INCREMENT (INC)
DEVICE SELECT (CS)
CONTROL AND MEMORY
RH/VH
RW/VW
RL/VL
VSS (GROUND)
GENERAL
U/D
INC
CS
5-BIT UP/DOWN COUNTER
5-BIT NONVOLATILE MEMORY
STORE AND RECALL CONTROL CIRCUITRY
VCC
VSS
DECODER
31
30
29
28
ONE OF THIRTY-TWO OUTPUTS ACTIVE AT A TIME
2
1
0
TRANSFER GATES
RESISTOR ARRAY
RH/VH
RL/VL
RW/VW
DETAILED