**Android Control Robotic Car**

Thanh Tien Truong

Husnu Melih Erdogan

ECE 558: Embedded System Programming

Professor: Roy Kravizt

TA: Hiral Barot

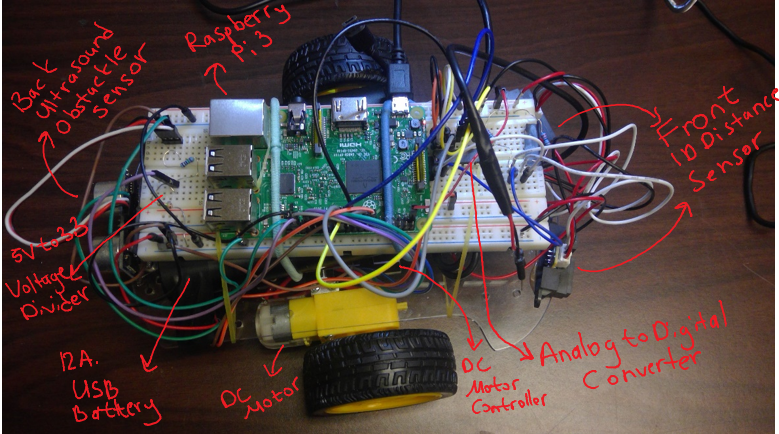
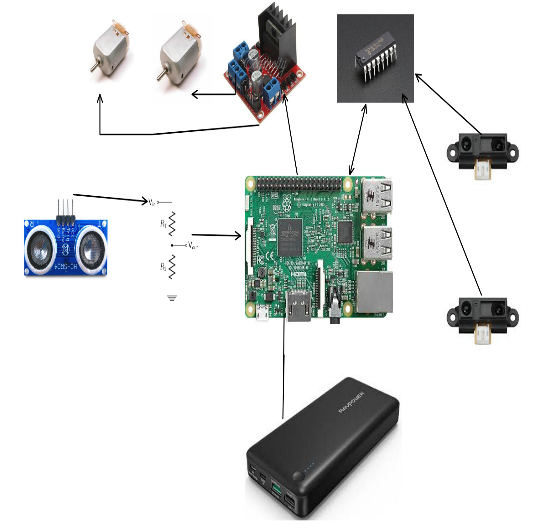
Date: December 9th, 2016



**Overview**

Android is a mobile operating system installed on mobile phones, tablets. It’s an open-source and it allows people to build, install and run their own Android applications. In ECE 558, we have learnt a lot about Android and have had some experience with building, installing and running Android application by doing projects. Since our interests are not only designing application but also learning more about how devices can communicate wirelessly, we want to do a project that mainly focus on Wi-Fi communication. Therefore, we decide to create an Android application that will Wi-Fi to communicate with a robotic car which is controlled by a Raspberry Pi 3 and Python scripts. We believe the project will hard enough to test our Android and Python skills, help us understand more about Wi-Fi communication and be an interesting project.

1. **Introduction**
   1. **Robotic Car Design**



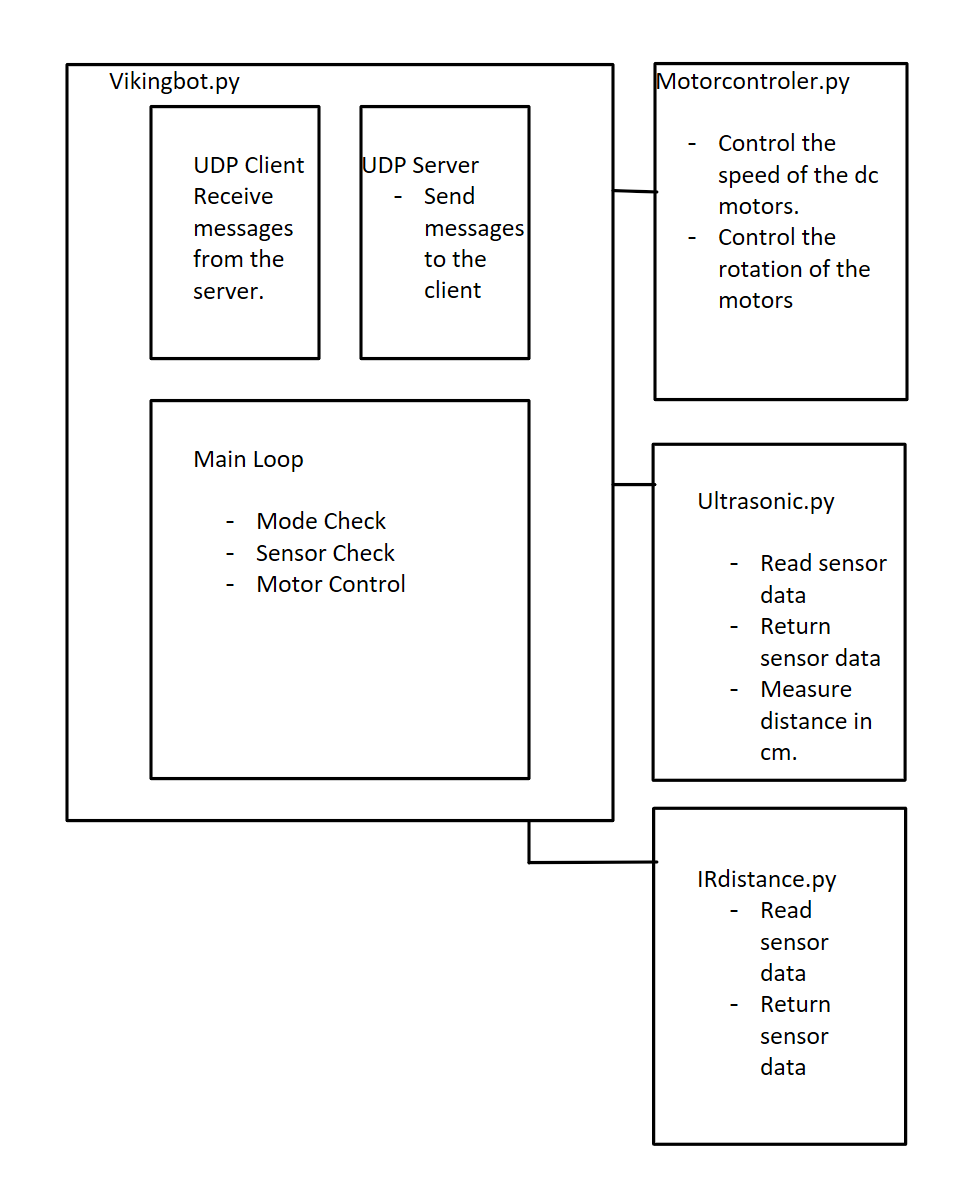
**Figure 1**: The design for the robotic car

You can find the top hardware design of the robot above. The robot consists of several different parts:

* 2 X DC motors
* 1 X Two channel DC motor driver
* 1 X ADC IC – MCP3000
* 2 X Sharp IR Distance Sensors – Sharp
* 1 X USB Battery Bank
* 1 X Ultrasonic Sensor
* 1 X 5 V to 3.3V voltage divider
* 1 X Raspberry Pi
* 2 X Micro USB Cables
* 20 X Jumper Wires
* 1 X bread board
* 1 X Plexiglass Robot Base

The Raspberry Pi reads the sensors and controls the speed and the rotation of the dc motors attached to the robot by running the Python scrips.

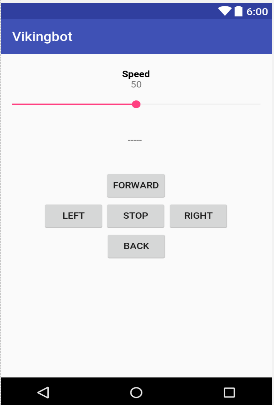
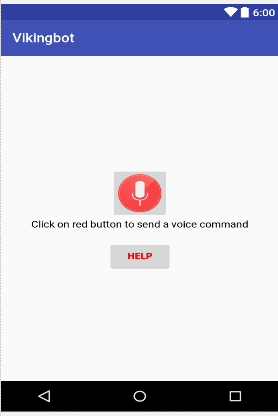
* 1. **Python Software Design**

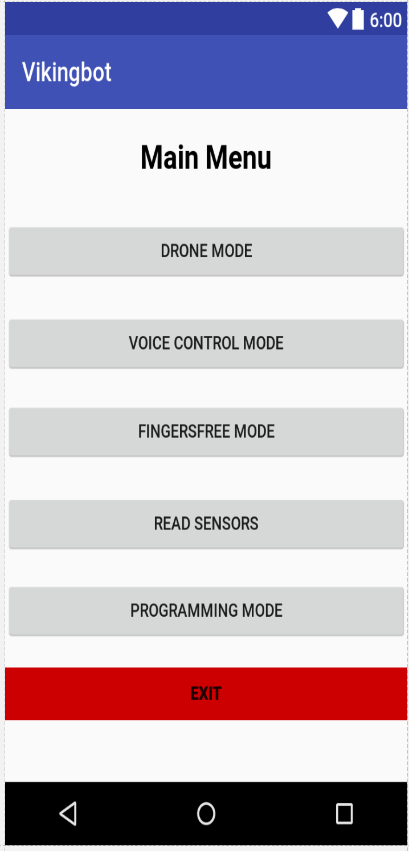


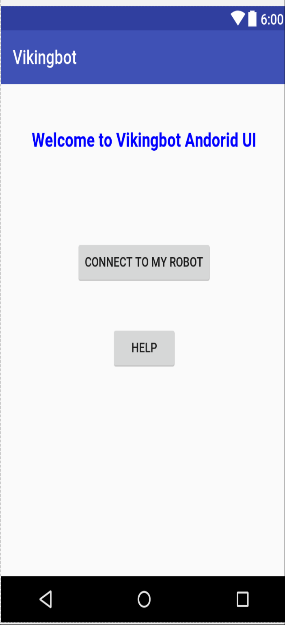
**Figure 2**: Python software design

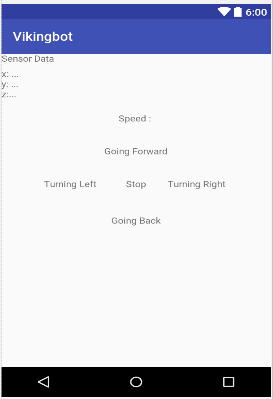
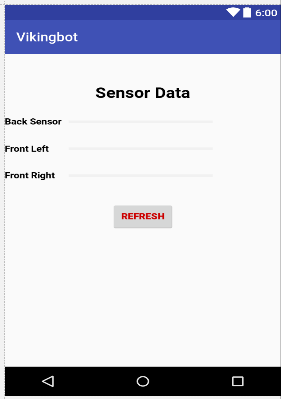
Vikingbot .py is the main python script. This script is in charge of the UDP communication, mode check, sensor data check, and motor control. It creates instances of ultrasonic, irdistance, and motorcontroller classes. Each class has several different modules to control the motors, read data from the ADC and ultrasonic sensor. It checks the messages that comes from the phone using the methods inside the other classes. It also sends messages back to the phone.

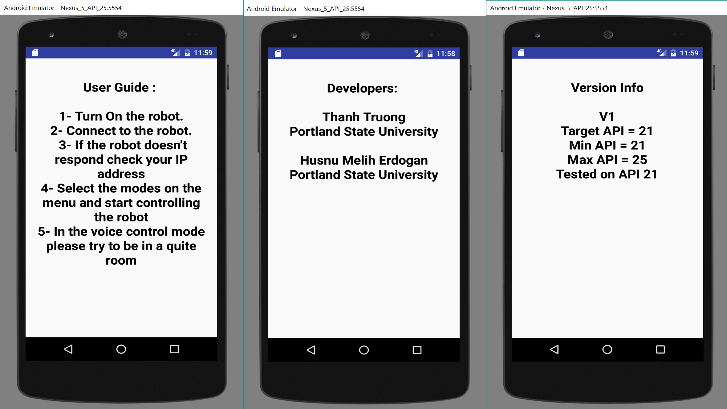
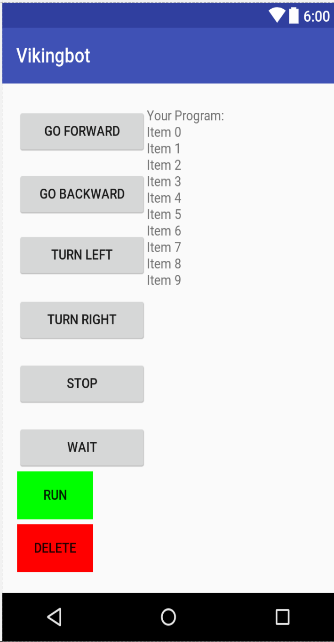
* 1. **Android Application Design**

****





****



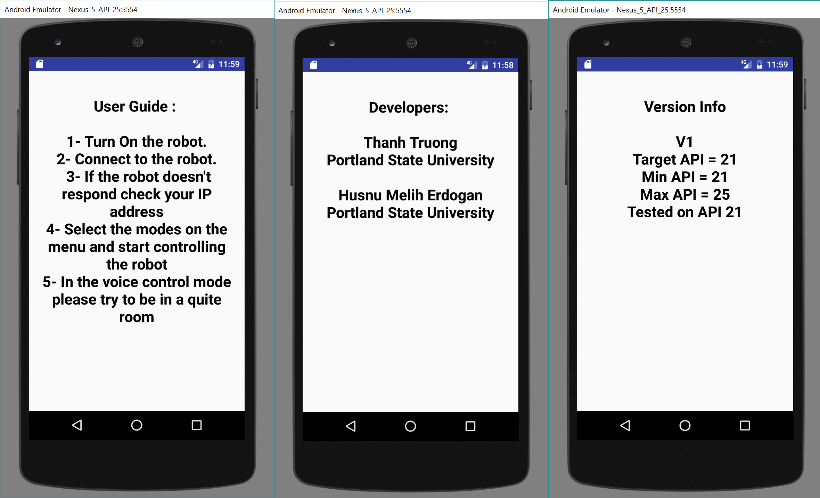
**Figure 3**: The design for the Android application

1. **Activities – Layouts and detailed information**
   1. **Welcome and Help**



Connecting to the robot automatically

Press

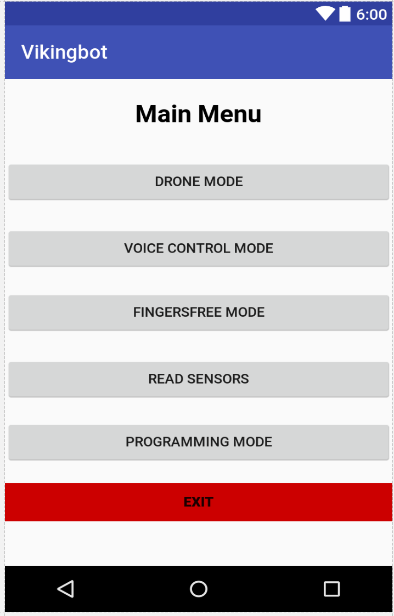
****

Press

**Figure 4**: Welcome screen and “Help” button

The welcome activity help users connect their phone to the robotic car. Currently, the IP address of the phone and the IP address of the Raspberry Pi 3 are hard coded into the program. It can cause some inconvenience since IP addresses can be different if we re-connect to the same routers or if we change to a different router. This can be improved in the future by letting users enter the two IP addresses and the application will try to pair them. If the application can pair them, it will move to the next activity. Otherwise, an error message will pop up and let users re-enter the IP addresses.

The “Help” button will allow users to see additional information of the application including information of the developers and the minimum and maximum API version needed to run the application.

* 1. **Main Menu**

Different modes to control the robotic car

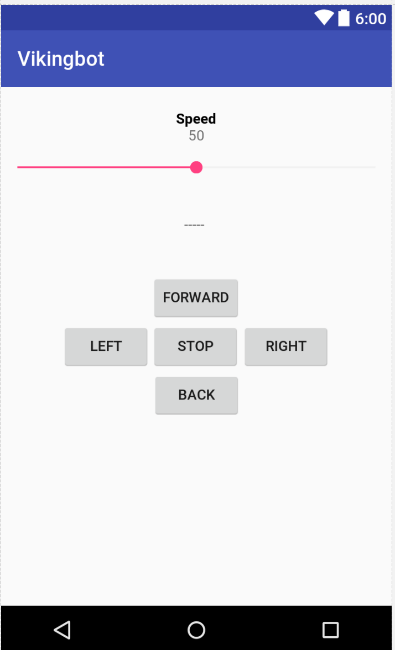
See what information the sensors provide

Go back to Welcome screen

**Figure 5**: Main menu screen

The main activity will allow users to select the mode they want to control the robotic car. There are four different modes total. The “Drone Mode” allows users to control the robotic car by pressing buttons. The “Voice Control Mode” allows users to control the robotic car by speaking. The “Fingers Free Mode” allows users to control the robotic car by using the accelerometer of the phone. The “Programming Mode” allows users to set up a list of actions for the robotic car to do.

* 1. **Drone Mode**

****

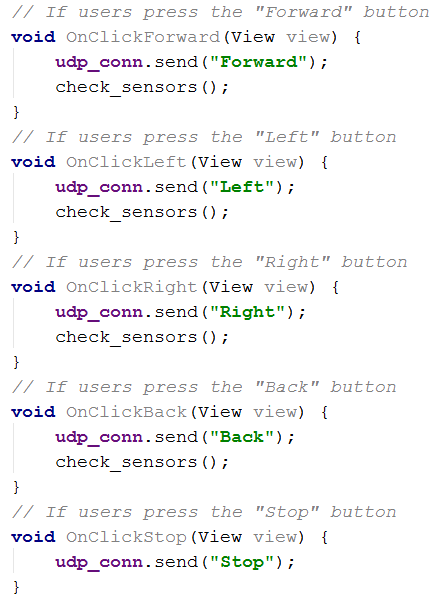
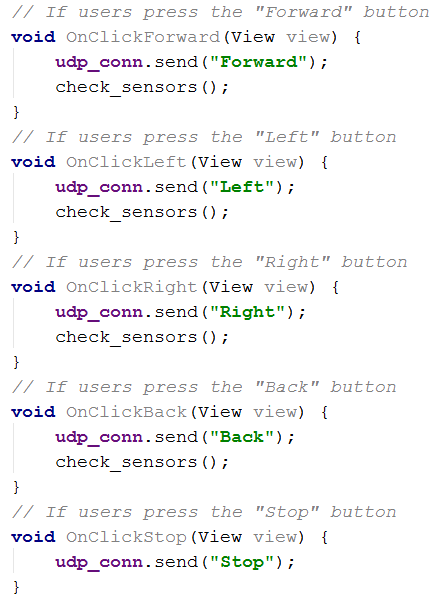
Seekbar to adjust the speed of the robotic car

A Textview to alert users if the robotic car detects an obstacle

Buttons to control the robotic car

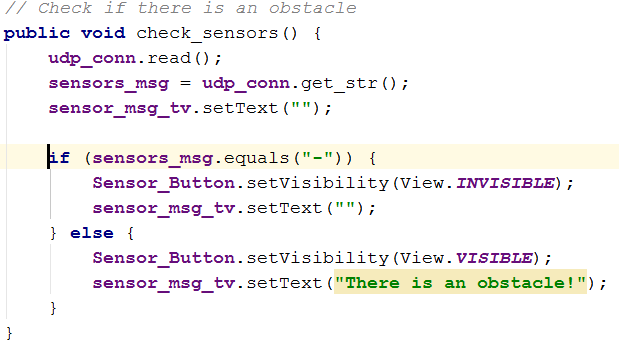
**Figure 6**: “Drone Mode” layout

In the “Drone Mode”, users can control the robotic car by pressing buttons. Each button will send a specific key word to the robotic car and the robotic car will behave differently depend upon the received key word.



**Figure 7**: “Drone Mode” screen

In addition, the application also ask for the information provided by the sensors on the robotic car. Based on that information, the application will set a “TextView” message and there will be a button for user to press if they want to see more detail about the location of the obstacle

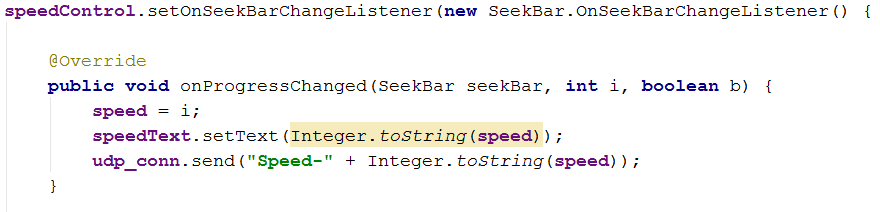


Set “TextView” and the button invisible if there is no obstacle

Set “TextView” message and make the button visible if there is an obstacle

**Figure 8**: check\_sensors method

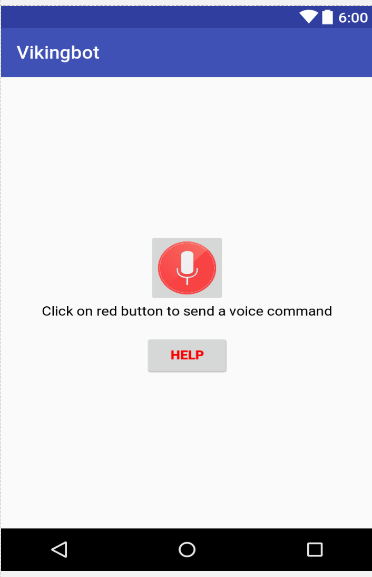
The Seekbar allows user to adjust the speed of the robotic car by dragging the thumb. A key word will be sent to the robotic car every time the Seekbar is changed.



A key word is sent

**Figure 9**: How to use SeekBar to change robotic car’s speed

* 1. **Voice Control Mode**



Start speaking (sending commands to the robotic car)

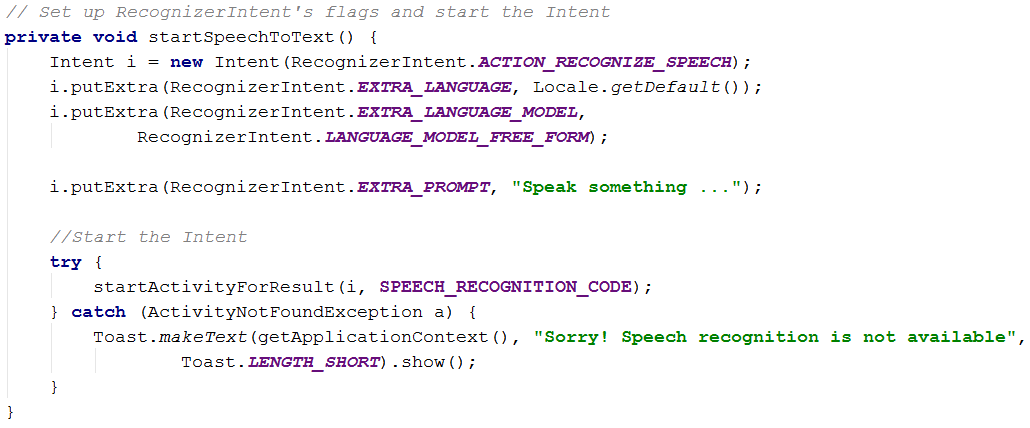
Showing list of possible commands

**Figure 10**: “Voice Control Mode” layout

Android comes with an inbuilt feature speech to text through which people can provide speech input to their app. In order to use the feature, the “RecognizerIntent” must be called. The “RecognizerIntent” allows people to set up necessary flags such as

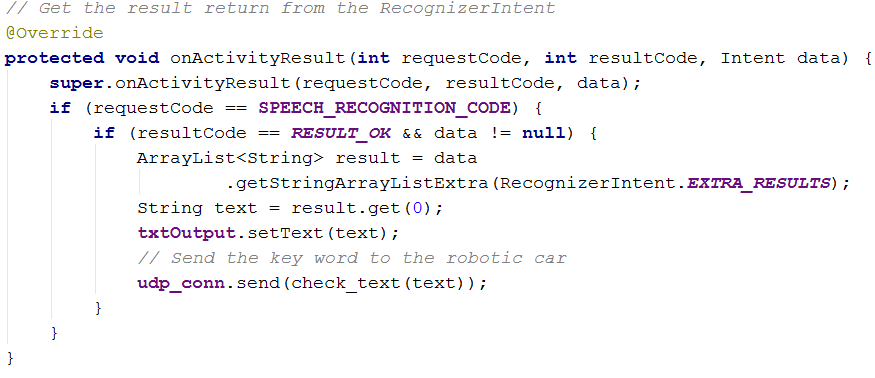
* ACTION\_RECOGNIZE\_SPEECH – Take user’s speech input and returns it to same activity
* LANGUAGE\_MODEL\_FREE\_FORM – Consider input in free form English
* EXTRA\_PROMPT – Text prompt to show to the user when asking them to speak

People can start the “RecognizerIntent” just like how they start other regular Intent.



Create a RecognizerIntent and set up its flags

Start the speech recognizer activity

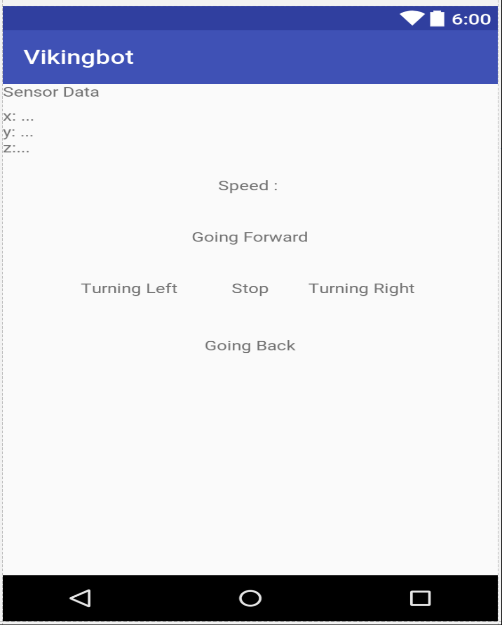


Get the return from the speech recognizer activity

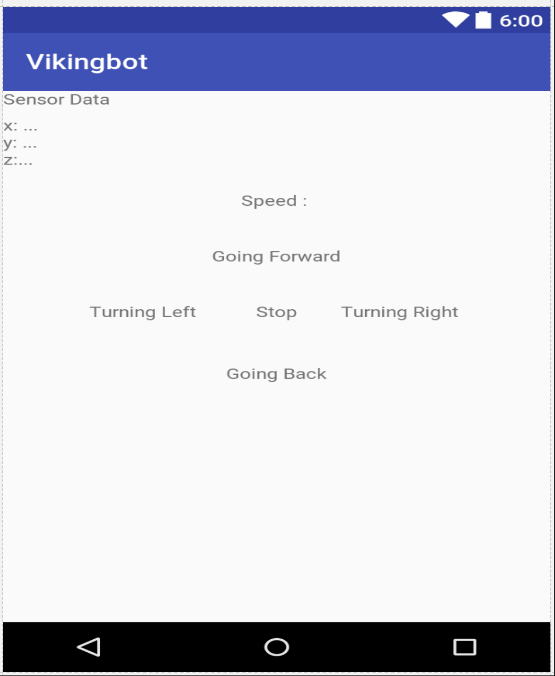
Send the command to the robotic car

**Figure 11**: How to use speech recognizer in Android

* 1. **Fingers Free Mode**



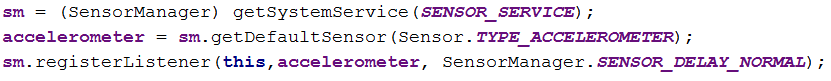
Current position of the phone



The word will turn to red color to indicate what state the robotic car currently in

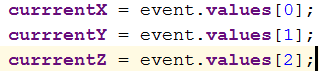
**Figure 12**: Fingers Free Mode layout

In “Fingers Free Mode”, users can control the robotic car by positioning the phone on their hand. The phone’s accelerometer will be used to control the robotic car.



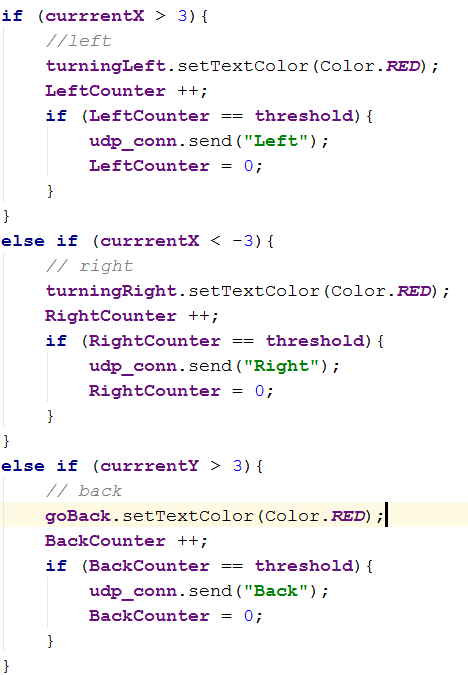
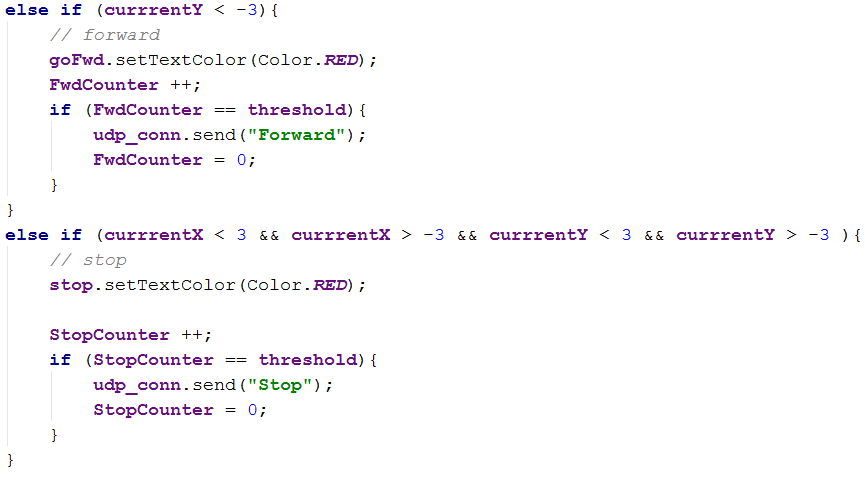
Initialize to use the accelerometer

**…**



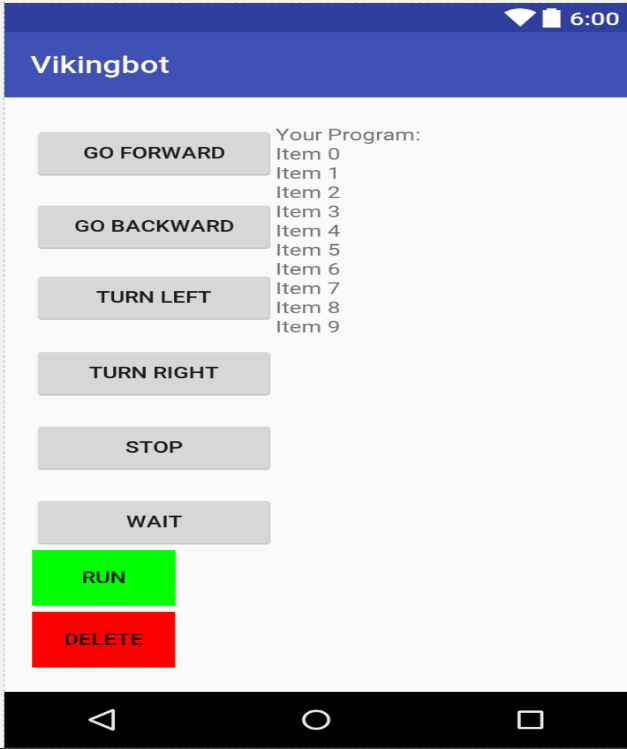
Get information from the accelerometer

**…**



**Figure 13**: How to use accelerometer to control robotic car

* 1. **Programing Mode**

****

RecyclerView for the list of added commands

Press to add the command that want the robotic car to perform

Delete the last command in the list

Send all the commands to the robotic car

**Figure 14**: Programing Mode layout

In the “Programming Mode”, users can set up a list of action that they want the robotic car to do, then press the “RUN” button to all the commands in the list to the robotic car. Users also can delete the last command in the list by pressing the “DELETE” button.



Each command in the list will be encoded an added into a string

Send the string to the robotic car

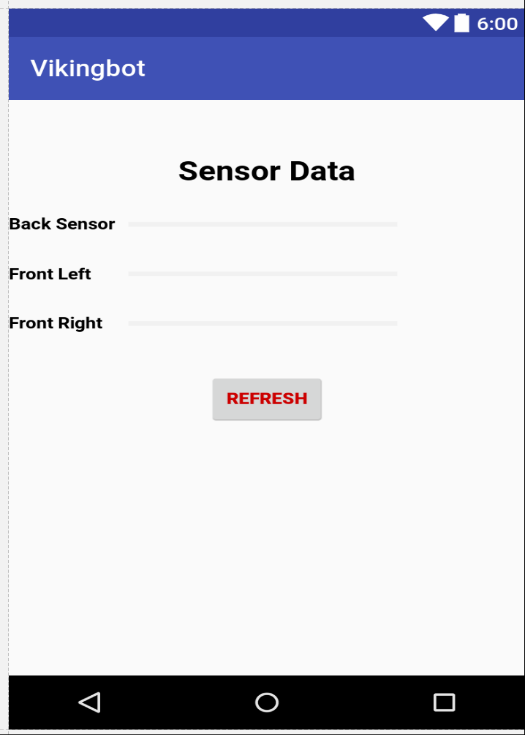
**Figure 15**: How commands are encoded and sent

Below is the table showing how each command is encoded

|  |  |
| --- | --- |
| **Command** | **Encode** |
| Forward | 1 |
| Left | 2 |
| Right | 3 |
| Back | 4 |
| Stop | 5 |
| Wait | 6 |
| Seconds to wait | Seconds + “\*” |
| **Example:**  Command: Forward, Left, Right, Back, Stop, Wait 2 seconds, Forward  Encode: 1 2 3 4 5 6 2\* 1  String sent: 1234562\*1 | |

**Table 1**: How commands are encoded

* 1. **Read Sensors**

****

Value get from three different sensors attached on the robotic car

Press to update to new value

**Figure 16**: “Read Sensor” layout

There are two ways to access to this screen, users can either press the button “READ SENSOR” in the main menu or press the button “Check Sensor” in the Drone Mode whenever an obstacle is detected. In the “Read Sensors” screen, users can see more detail about the how far the obstacle is and where it is from the robotic car based on the value given from three different sensors attached on the robotic car.

1. **Wi-Fi communication**

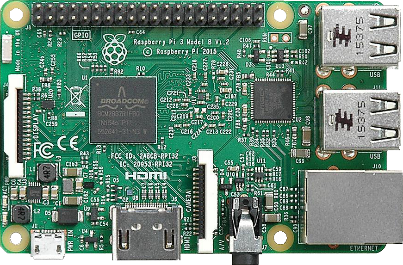
In order to have a reliable Wi-Fi connection, we built our own local network with a wireless rougher. Both the Raspberry Pi and the testing phone have to be on the same network, and both the robot and the phone have to be in the range of the network.



UDP Packages

UDP Packages





**Figure 17**: Hardware setup

There are threads to control how UDP packages are transferred and received

* 1. **How to transfer data**

In order to transfer data, we initially have to create a socket, then convert data into array of byte and send that array to the socket that we previously create.



Initialize port to use

Create a socket to send data

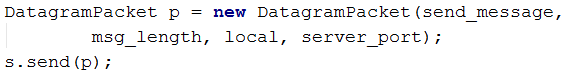
**…**



**…**



Get data’s length and convert data into Byte array

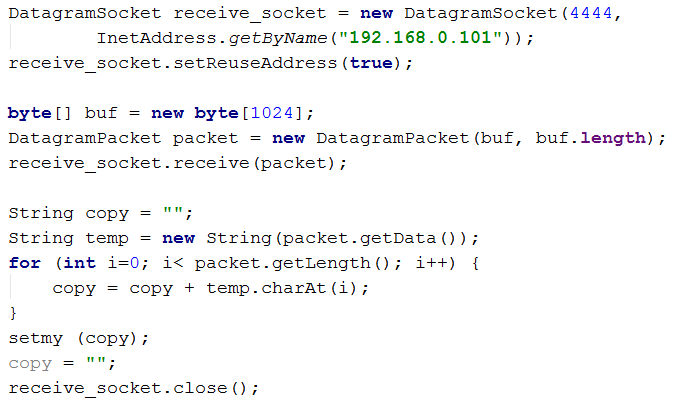


Create package containing the byte array and send it

**Figure 17**: How data is transferred from the application using UDP protocol

* 1. **How to receive data**

Since the robotic car will send the data in a form of byte array, in order to receive the data, we initially have to create a socket, byte array and package. The data will sent through the created socket and be store in a byte array inside the package.



Create a socket, byte array and package to receive data

Received data will be in the package

Get data from the package

**Figure 17**: How data is received from the application using UDP protocol

1. **Challenge**

* UDP communication: It was challenging to create a reliable UDP communication between Raspberry Pi and the Android phone due to lock of good documentation and tutorials. However, we were able to solve all the problems we have face.
* Building the robot: Putting the robot together took some time. Testing all the motions and sensors of the robot took more than our expectations.
* Testing and debugging the Android code: Although we didn’t have any big problems in our project, we had some small issues with the accelerometer, UDP communication and the speech recognition. Not all of them, but it was challenging to figure out a way to solve some of these issues.

1. **Future Improvements**

* Allow users to enter IP addresses and let the devices pair themselves
* Adding camera on the robotic car and adding streaming features or taking pictures features on the Android application (We will need to use TCP protocol this time)
* Improve the speed and accuracy of speech recognition

1. **Reference**

[**https://developer.android.com/reference/java/net/DatagramSocket.html**](https://developer.android.com/reference/java/net/DatagramSocket.html)

[**https://developer.android.com/reference/android/speech/RecognizerIntent.html**](https://developer.android.com/reference/android/speech/RecognizerIntent.html)

[**https://wiki.python.org/moin/UdpCommunication**](https://wiki.python.org/moin/UdpCommunication)