ABSTRACT

Today human-machine interaction is moving away from mouse and pen and is becoming pervasive and much more compatible with the physical world. With each passing day the gap between machines and humans is being reduced with the introduction of new technologies to ease the standard of living. The objective of designing this robot is simply to facilitate the humans in the future for security purposes. In the present scenario, there are many recent developments of robotics and communication on a large scale. The robot is in the form of a vehicle mounted with a smartphone. The movement of vehicle is controlled by microcontroller.

The main objective of the project was to command and control the Robot wirelessly by the GUI Application. The main task of this project was basically divided in two parts:

(1)to program the arduino microcontroller mounted on Robot interfaced which accepts input from the cell phone connected to another cell phone farther away and enables us to control the motion and other specifications of the robot.

(2)to program the GUI Application in android platform which would enable us to connect to robotic interface.

As a result, we achieved both wireless communication between the android app and the robotic interface, and serial communication between the DTMF module and the arduino board.

Contents

List of Figures

Figure Page No.

List of Codes

Code Page No.

CHAPTER 01

INTRODUCTION

Nowadays smart phones are becoming more powerful with reinforced processors, larger storage capacities, richer entertainment function and more communication methods. Many android based apps are being used for various purposes which includes data exchange, entertainment, surveillance, communication, education etc. Thank for these technology and other similar techniques, with dramatic increase in Smartphone users, smart phones have gradually turned into an all-purpose portable device and provided people for their daily use. In recent years, an open-source platform Android has been widely used in smart phones. Android has complete software package consisting of an operating system, middleware layer and core applications. Different from other existing platform like iOS (iPhone OS), it comes with software development kit (SDK), which provides essential tools and Application. Using a Smartphone as the "brain" of a robot is already an active research field with several open opportunities and promising possibilities. In this project we present a review of current robots controlled by mobile phone and discuss a closed loop control systems using audio channels of mobile devices, such as phones and tablet computers. In our work, move the robot upward, backward, left and right side by the android application.

Our robot in initial phase will be controlled using 2 mobiles as following:

First mobile should be connected to project. Settings: This phone should be in auto answer mode. This setting can be done in Headphone setting or Headset setting or in Enhancement settings.

Second mobile is used to give commands. Settings: Turn on Keypad tones in this mobile. These tones are DTMF tones. Please note: in some mobiles glossy or beep settings are there, make sure you select DTMF tones.

Then we have to dial a call from second mobile to first mobile. Call will be received automatically due to auto answer mode. We can press the keys once the call is received.

Then instead of the calling, an application will be used to directly control the robot. Android is a very familiar word in the world today. Millions of devices are running the Google Android OS and millions are being developed daily. Google has made the Android development platform open to everyone around the world, so there are millions of developers.

Purpose of our research is to provide simpler robot's hardware architecture but with powerful computational platforms and research and testDtmf connection infrastructure.

CHAPTER 2

ROBOTICS

Robotics[1] is the branch of mechanical engineering, electrical engineering and computer science that deals with the design, construction, operation, and application of robots, as well as computer systems for their control, sensory feedback, and information processing.

These technologies deal with automated machines that can take the place of humans in dangerous environments or manufacturing processes, or resemble humans in appearance, behaviour, and or cognition. Many of today's robots are inspired by nature, contributing to the field of bio-inspired robotics.

The concept of creating machines that can operate autonomously dates back to classical times, but research into the functionality and potential uses of robots[2] did not grow substantially until the 20th century. Throughout history, it has been frequently assumed that robots will one day be able to mimic human behaviour and manage tasks in a human-like fashion. Today, robotics is a rapidly growing field, as technological advances continue; researching, designing, and building new robots serve various practical purposes, whetherdomestically, commercially, or militarily. Many robots are built to do jobs that are hazardous to people such as defusing bombs, finding survivors in unstable ruins, and exploring mines and shipwrecks. Robotics is also used in STEM (Science, Technology, Engineering, and Mathematics) as a teaching aid.

There are many types of robots; they are used in many different environments and for many different uses, although being very diverse in application and form they all share three basic similarities when it comes to their construction:

1. Robots all have some kind of mechanical construction, a frame, form or shape designed to achieve a particular task. For example, a robot designed to travel across heavy dirt or mud, might use caterpillar tracks. The mechanical aspect is mostly the creator's solution to completing the assigned task and dealing with the physics of the environment around it.

2. Robots have electrical components which power and control the machinery. For example, the robot with caterpillar tracks would need some kind of power to move the tracker treads. That power comes in the form of electricity, which will have to travel through a wire and originate from a battery, a basic electrical circuit. Even petrol powered machines that get their power mainly from petrol still require an electric current to start the combustion process which is why most petrol powered machines like cars, have batteries. The electrical aspect of robots is used for movement (through motors), sensing (where electrical signals are used to measure things like heat, sound, position, and energy status) and operation (robots need some level of electrical energy supplied to their motors and sensors in order to activate and perform basic operations)

3. All robots contain some level of computer programming code. A program is how a robot decides when or how to do something. In the caterpillar track example, a robot that needs to move across a muddy road may have the correct mechanical construction, and receive the correct amount of power from its battery, but would not move anywhere without a program telling it to move. Programs are the core essence of a robot, it could have excellent mechanical and electrical construction, but if its program is poorly constructed its performance will be very poor (or it may not perform at all). There are three different types of robotic programs: remote control, artificial intelligence and hybrid. A robot with remote control programing has a pre-existing set of commands that it will only perform if and when it receives a signal from a control source, typically a human being with a remote control. It is perhaps more appropriate to view devices controlled primarily by human commands as falling in the discipline of automation rather than robotics. Robots that use artificial intelligence interact with their environment on their own without a control source, and can determine reactions to objects and problems they encounter using their pre-existing programming.

We develop an application controlled robot which can be controlled wirelessly through an android app and also via calling feature of any mobile. The technology used isbased on decoding the DTMF tones generated by a mobile.

2.1 Hardware Requirements

DTMF controlled robot[4] run by some commands that are send via mobile phone. We are here using DTMF function of mobile phone. One is user mobile phone that we will call 'remote phone' and second one that are connected with Robot's circuit using aux wire. This mobile phone we will call 'Receiver Phone'.

First we make a call by using remote phone to receiver phone and then attend the call by manually or automatic answer mode.

Apart from the 2 mobile phones for communication, other components required during the complete assembly are:

1. Arduino Uno Board
2. L298N Motor Driver
3. DTMF Decoder
4. Chassis
5. Breadboard
6. Wheels
7. Batteries
8. Mobile Phone
9. 3.5mm AUX Cable
10. Jumper Wires

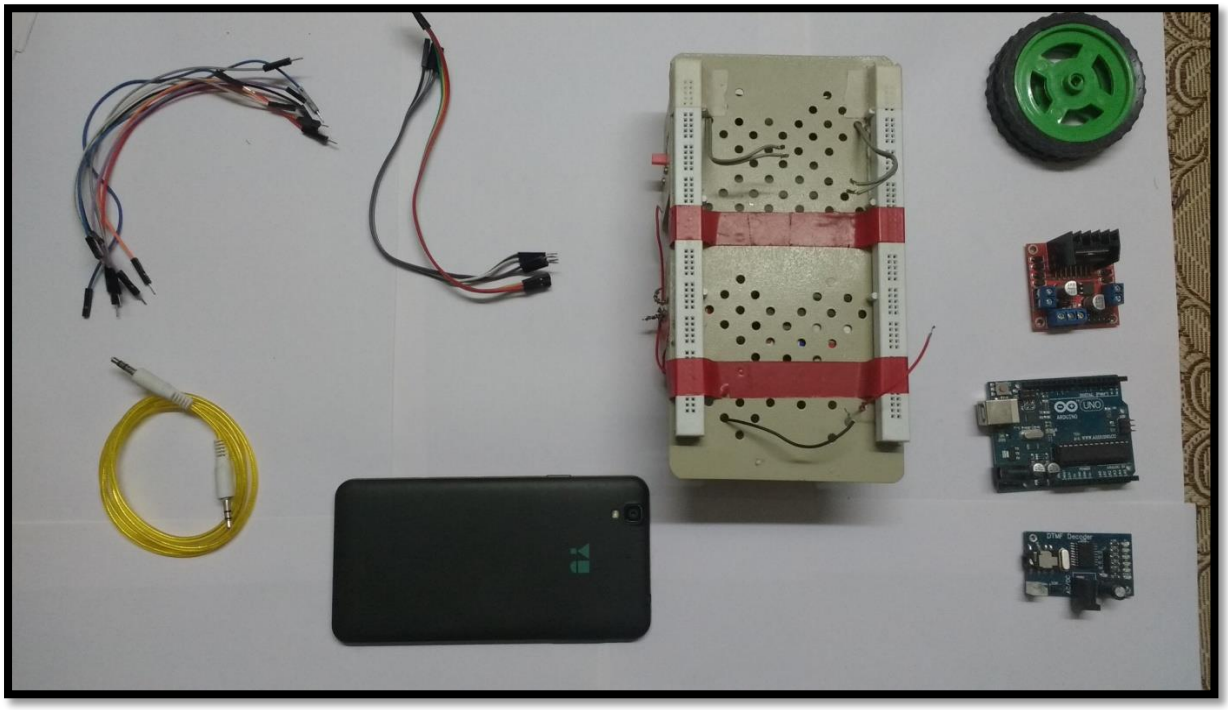


Figure 2.1 Components of Robot

The complete description of the above mentioned components will be provided later in the chapter.

2.2 COMPONENT DESCRIPTION

The following list shows robot control architecture

2.2.1 Arduino Uno R3

Arduino Uno Rev3 is a microcontroller board based on the ATmega328P, an 8-bit microcontroller with 32KB of Flash memory and 2KB of RAM. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

Arduino/Genuino Uno has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. The Uno board is the first in a series of USB boards and it is the reference model for the Arduino platform.

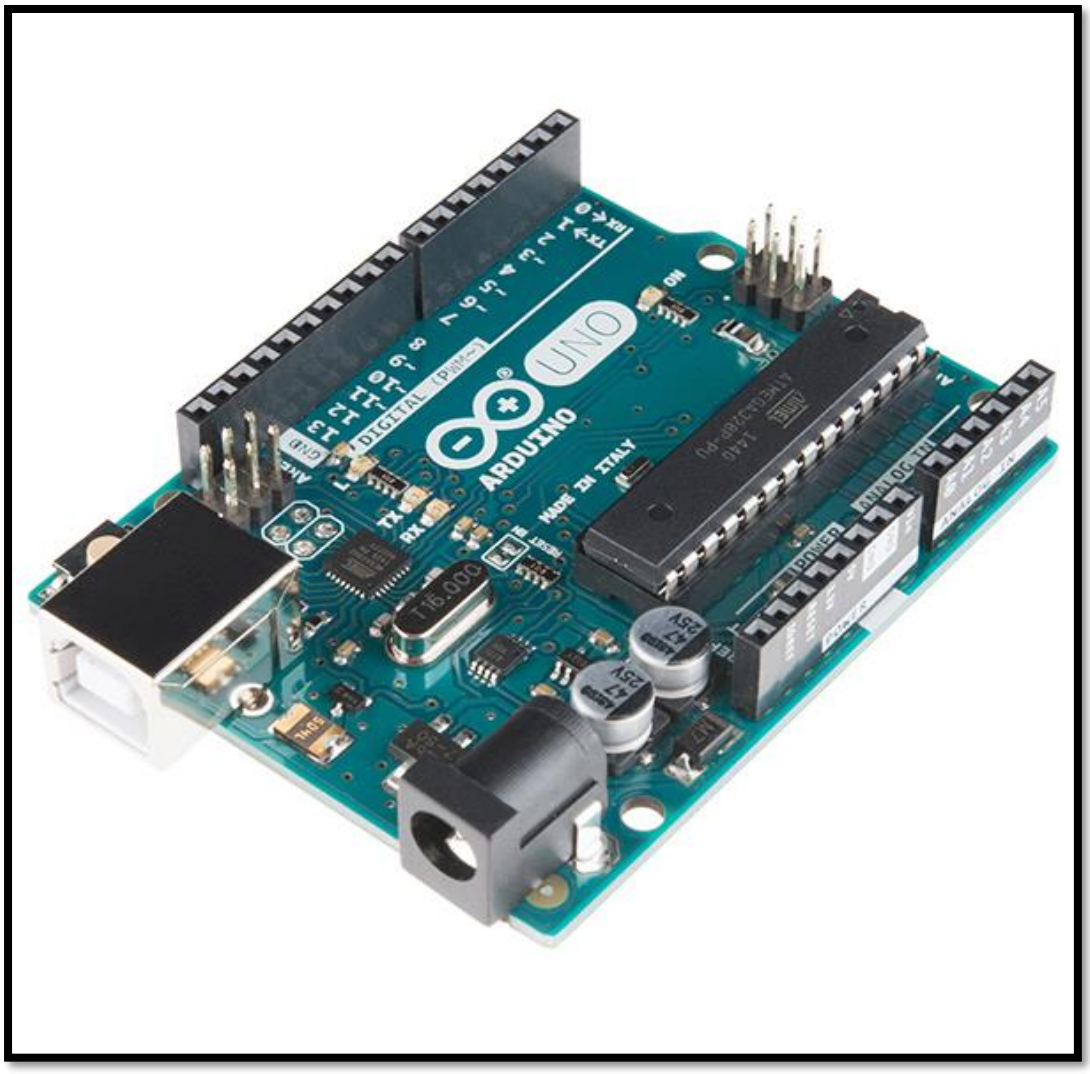


Figure 2.2 Arduino Uno R3

2.2.2 DTMF Decoder

This DTMF (Dual Tone Multi Frequency) decoder circuit identifies the dial tone from the telephone line and decodes the key pressed on the remote telephone. Here for the detection of DTMF signalling, we are using the IC MT8870DE which is a touch tone decoder IC. It decodes the input DTMF to 5 digital outputs. The M-8870 DTMF (Dual Tone Multi Frequency) decoder IC uses a digital counting technique to determine the frequencies of the limited tones and to verify that they correspond to standard DTMF frequencies. The DTMF tone is a form of one way communication between the dialler and the telephone exchange. The whole communication consists of the touch tone initiator and the tone decoder or detector. The decoded bits can be interfaced to a computer or microcontroller for further application.

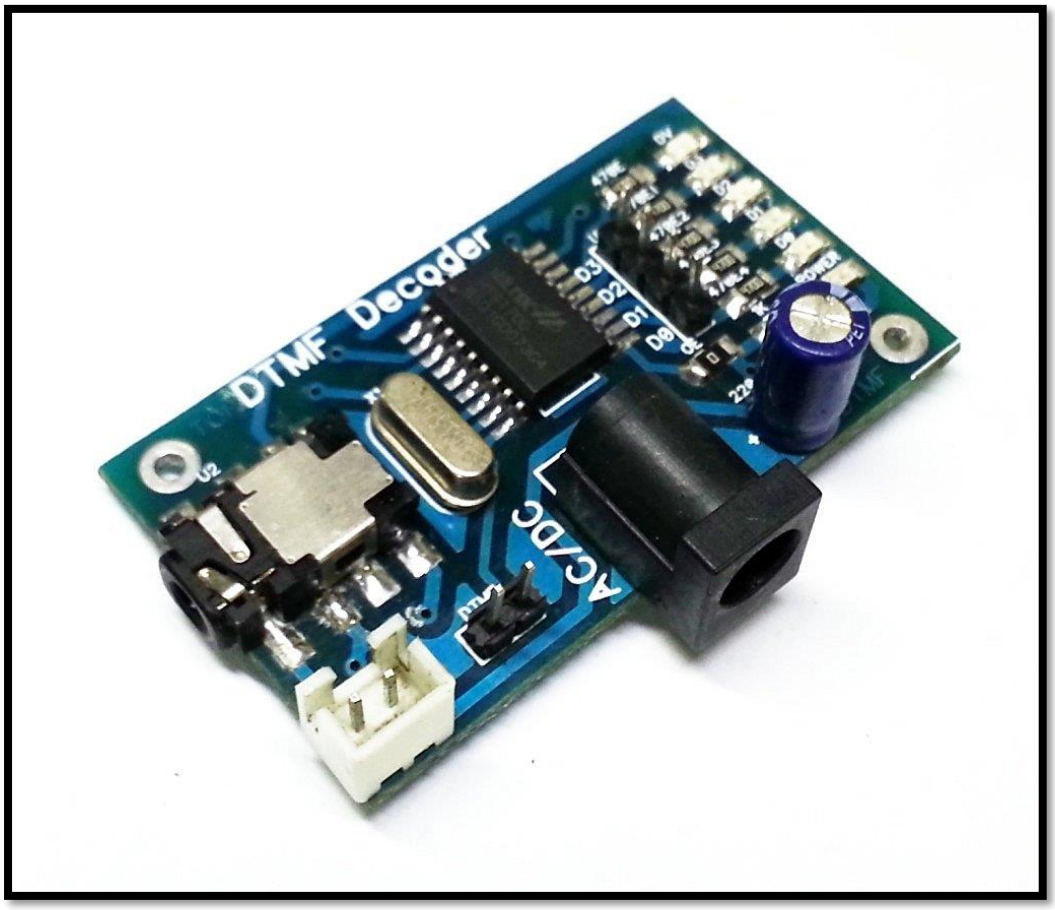


Figure 2.3 M-8870 DTMF

2.2.3 L298N Motor Controller

H-Bridges are typically used in controlling motors speed and direction, but can be used for other projects such as driving the brightness of certain lighting projects such as high powered LED arrays. An H-Bridge is a circuit that can drive a current in either polarity and be controlled by Pulse Width Modulation (PWM).

Pulse Width Modulation is a means in controlling the duration of an electronic pulse. In motors try to imagine the brush as a water wheel and electrons as the flowing droplets of water. The voltage would be the water flowing over the wheel at a constant rate, the more water flowing the higher the voltage. Motors are rated at certain voltages and can be damaged if the voltage is applied to heavily or if it is dropped quickly to slow the motor down. Thus PWM. Take the water wheel analogy and think of the water hitting it in pulses but at a constant flow. The longer the pulses the faster the wheel will turn, the shorter the pulses, the slower the water wheel will turn. Motors will last much longer and be more reliable if controlled through PWM.

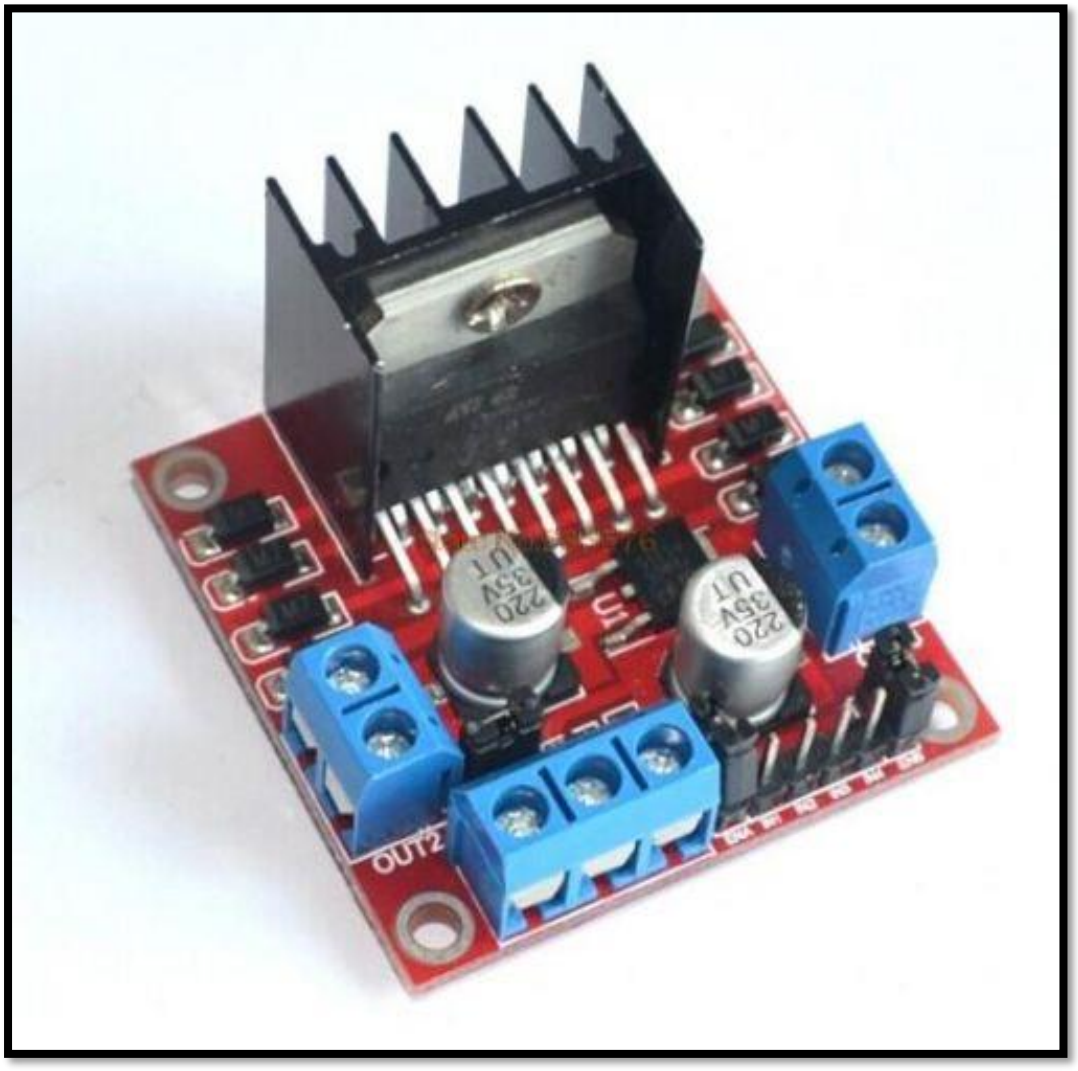


Figure 2.4 L298N Motor Controller

2.2.4 DC MOTOR

Almost every mechanical movement that we see around us is accomplished by an electric motor. Electric machines are means of converting energy. Motors take electrical energy and produce mechanical energy. Electric motor is used to power hundreds of devices we use in everyday life. An example of small motor applications includes motors used in automobiles, robot, hand power tools and food blenders

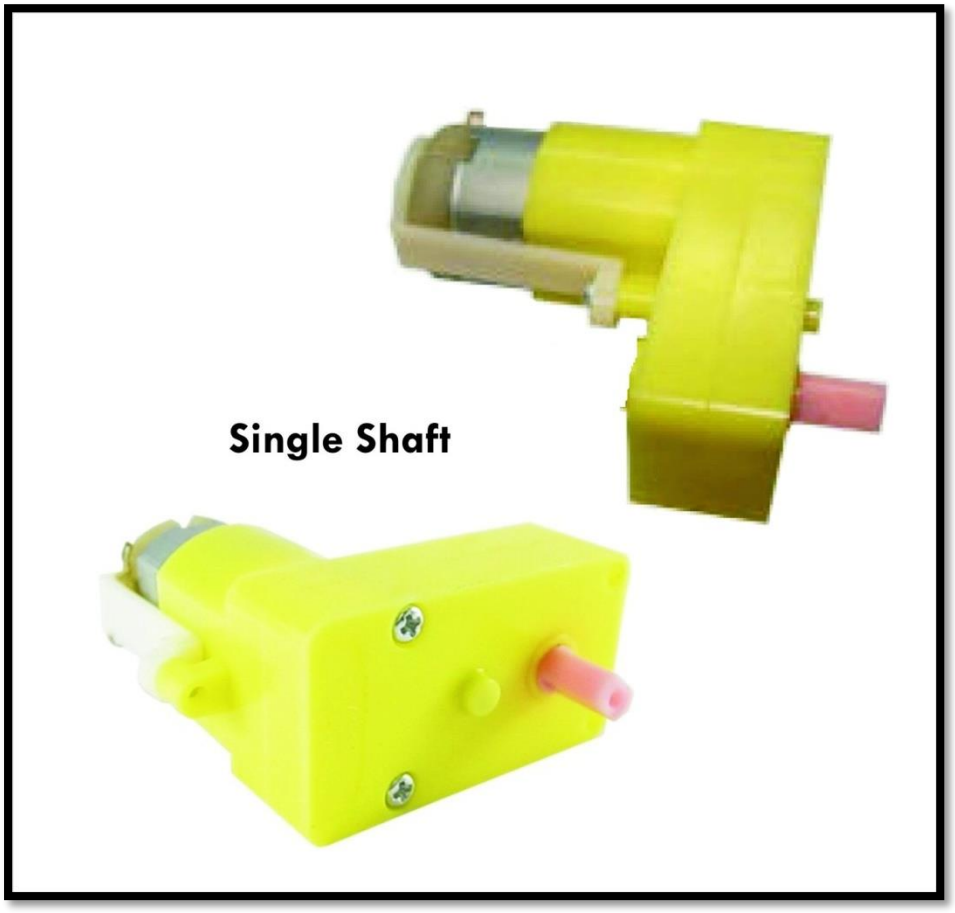


Figure 2.5 BO Motor

CHAPTER 03

BLOCK DIAGRAM

A smart phone Android operated robot[8]. Now here is a simple diagram to explain control of our robot using Dtmf and arduino microcontroller with our android Smartphone device. The controlling devices of the whole system are a microcontroller. Dtmf module, DC motors are interfaced to the microcontroller. The data receive by the Dtmf from android smart phone is fed as input to the controller. The controller acts accordingly on the DC motor of the robot. The robot in the project can be made to move in all the four directions using the android phone. The direction of the robot is indicators using LED indicators of the Robot system. In achieving the task the controller is loaded with program written using Embedded 'C' Languages.

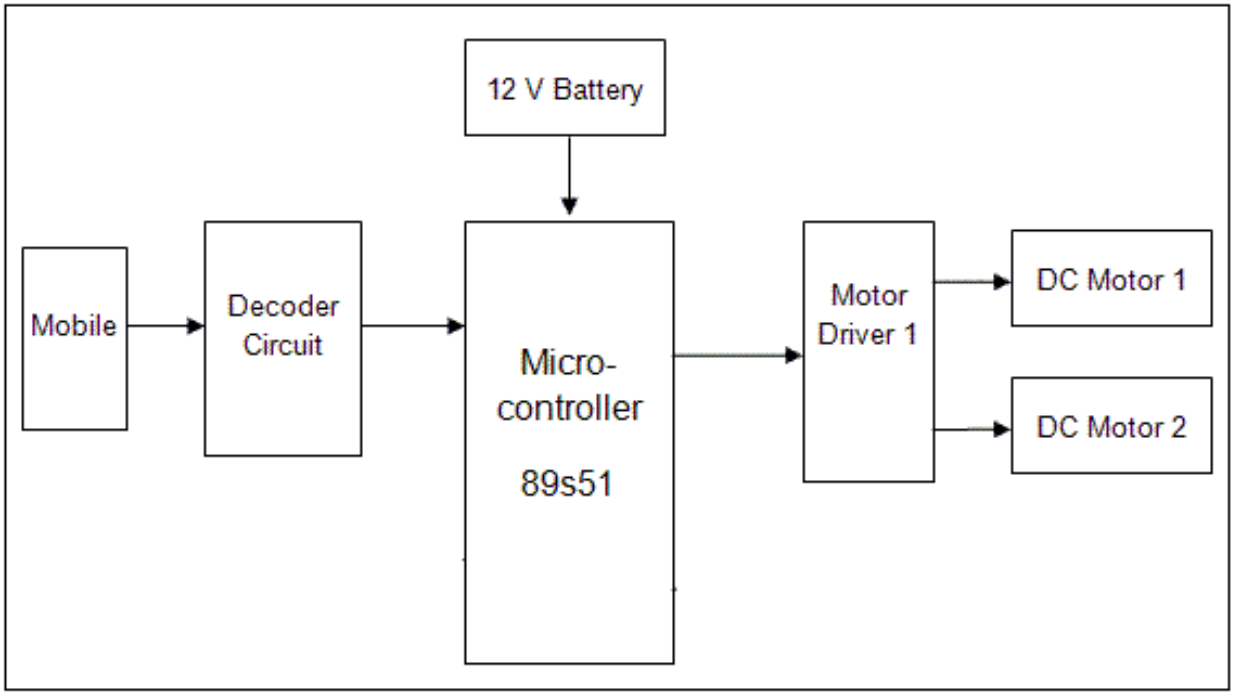


Figure 3.1 Block Diagram: To explain control of our robot

3.1 DESCRIPTION ABOUT BLOCK DIAGRAM

In this project the robot, is controlled by a mobile phone that makes call to the mobile phone attached to the robot. In the course of the call, if any button is pressed, a tone corresponding to the button pressed is heard at the other end of the call. This tone is called 'DUAL –TONE MULTIPLE-FREQUENCY' (DTMF) tone. The robot receives this DTMF tone with the help of phone stacked in the robot. The received tone is processed by the Arduino Uno microcontroller with the help of DTMF decoder MT8870. The decoder decodes

the DTMF tone in to its equivalent binary digit and this binary number is send to the microcontroller. The microcontroller is pre-programmed to take a decision for any give input and outputs its decision to motor drivers in order to drive the motors for forward or backward motion or a turn. The mobile that makes a call to the mobile phone stacked in the robot acts as a remote. So this simple robotic project does not require the construction of receiver and transmitter units. DTMF signalling is used for telephone signalling over the line in the voice frequency band to the call switching centre. The version of DTMF used for telephone dialling is known as 'Touch –Tone'. DTMF assigns a specific frequency (consisting of two separate tones) to each key s that it can easily be identified by the electronic circuit. The signal generated by the DTMF encoder is the direct algebraic submission, in real time of the amplitudes of two sine (cosine) waves of different frequencies, i.e., pressing '5' will send a tone made by adding 1336Hz and 770Hz to the other end of the mobile. The tones and assignments in a DTMF system shown below:

Frequencies	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Figure 3.2 DTMF system: Tones and frequency assignments

3.2 DTMF BASICS

DTMF, or tone dialling, is very commonly used. DTMF (Dual-tone Multi Frequency)[9] is a tone composed of two sine waves of given frequencies. Individual frequencies are chosen so that it is quite easy to design frequency filters, and so that they can easily pass through telephone lines (where the maximum guaranteed bandwidth extends from about 300 Hz to 3.5 kHz).

DTMF was not intended for data transfer; it is designed for control signals only. With standard decoders, it is possible to signal at a rate of about 10 "beeps" (=5 bytes) per second.

DTMF standards specify 50ms tone and 50ms space duration. For shorter lengths, synchronization and timing becomes very tricky.

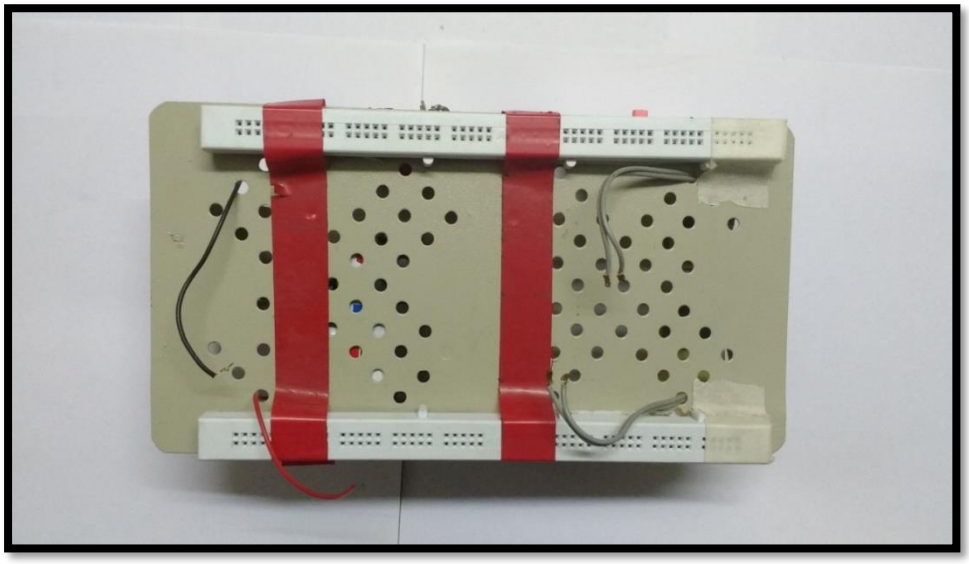
CHAPTER 04

CONSTRUCTING OUR BOT

Following steps were followed for the construction of the Land Rover Robot:

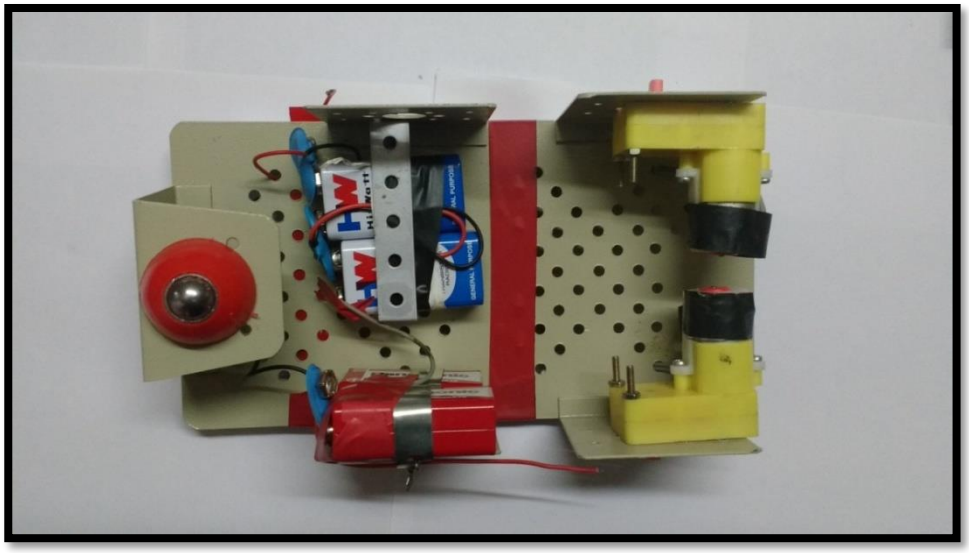
4.1 Preparation of initial frame

Attach breadboard strips on the top of the chassis for extending connections.



(a)

Also attach batteries and BO Motors at the bottom of chassis.



(b)

Figure 4.1 Initial Frame (a)Top of the chassis (b)Bottom of the chassis with all necessary connections

4.2 Connect the Motor Driver

Connect the motor driver to the BO Motors and attach it to the chassis of the Robot.

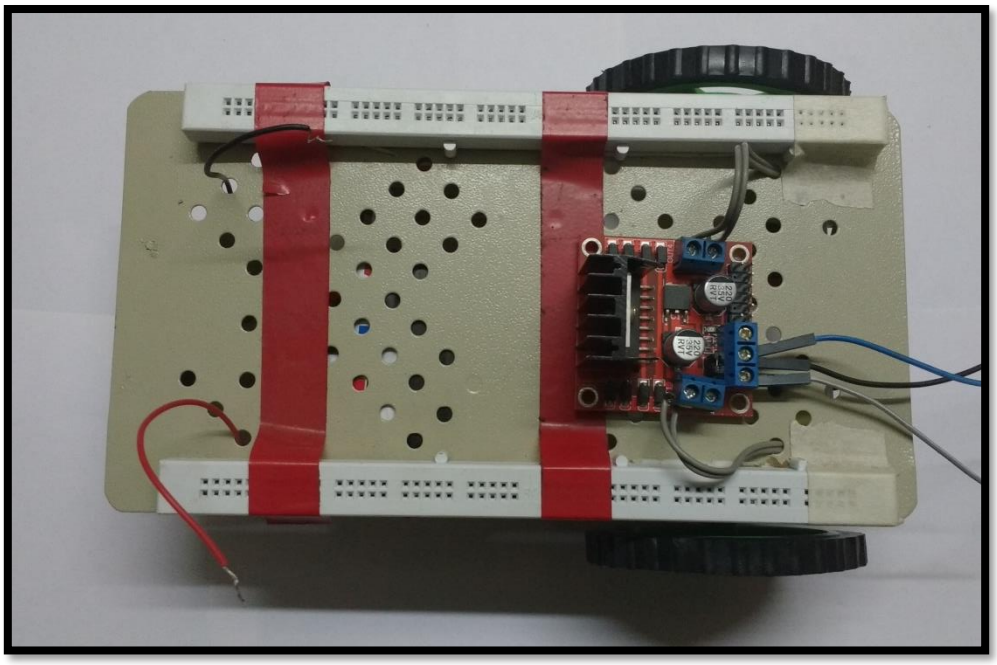


Figure 4.2 Connect the Motor Driver

4.3 Connect the Arduino Board

Make the proper connections using the jumper wires to connect the motor driver to the board. Further attachhe board to the chassis of the robot.

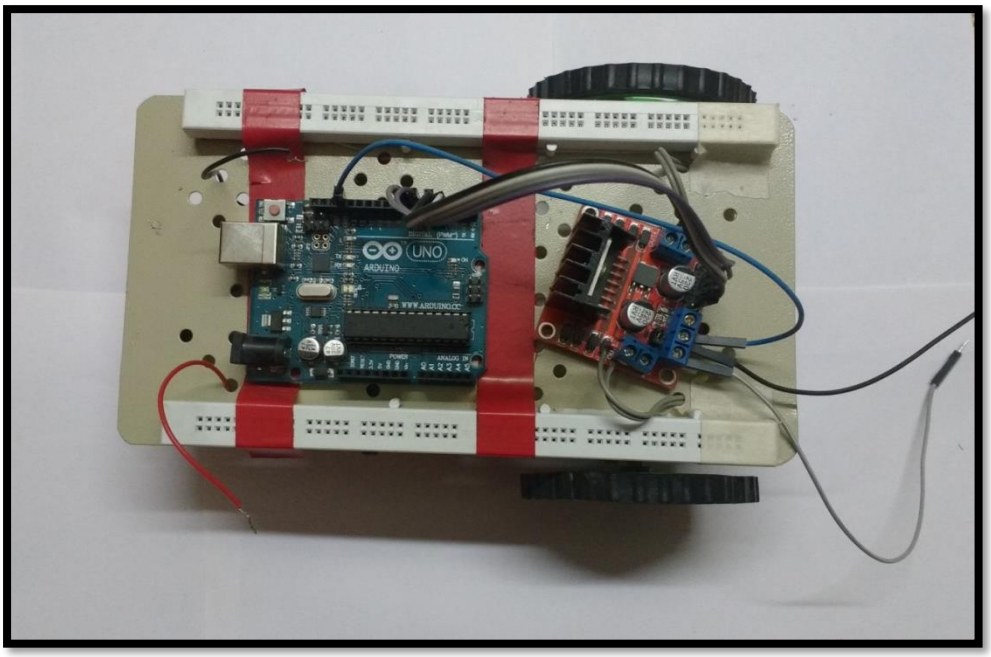


Figure 4.3 Connect the Arduino Board

4.4 Adding DTMF module to the Robot

Make the proper connections for the DTMF module and arduino board. Add the module on the chassis of the robot.

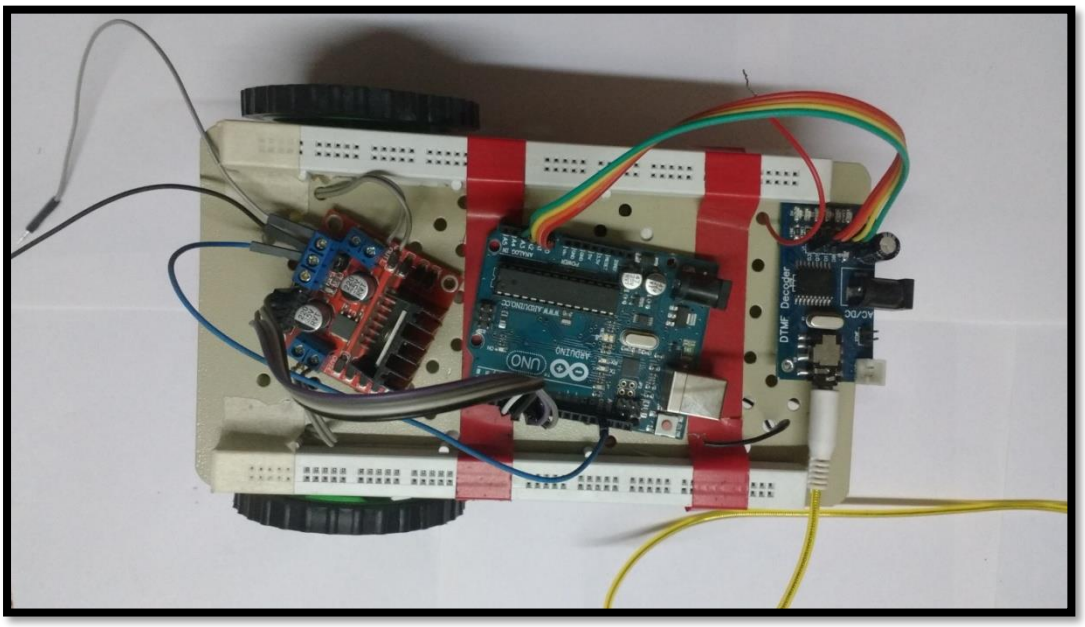


Figure 4.4 Connect DTMF module to the Robot

4.5 Connect the Mobile

Connect the mobile to the DTMF module using 3.5mm jack.

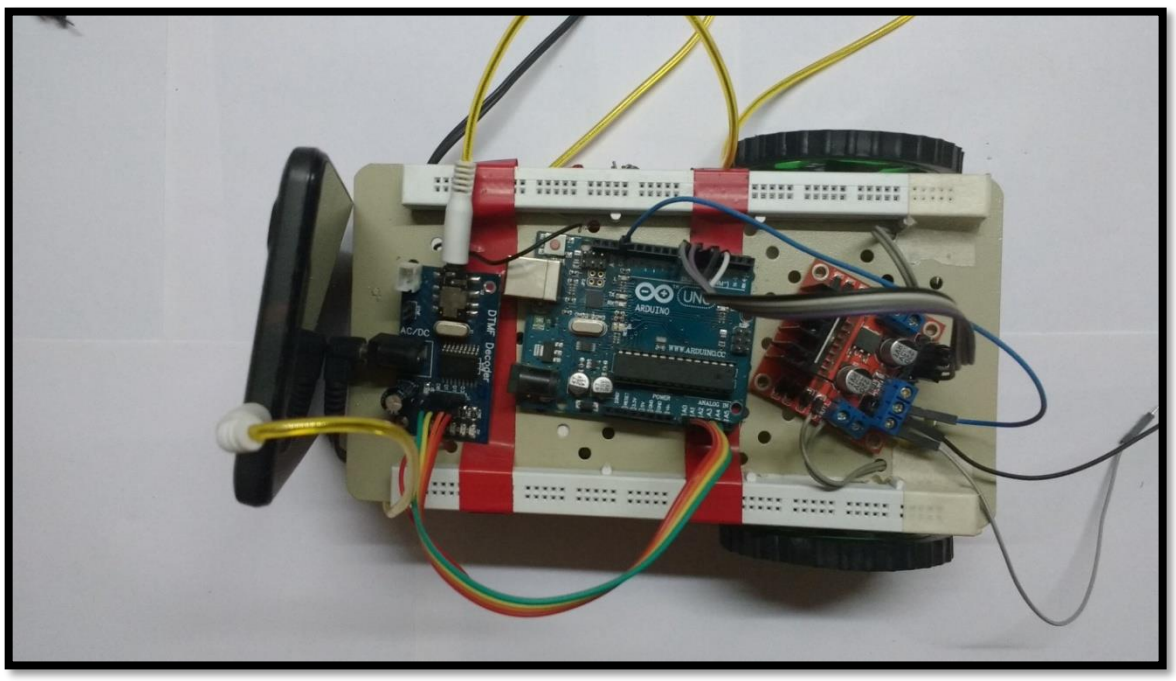


Figure 4.5 Connect the Mobile

CHAPTER 05

TESTING OF COMPONENTS

Testing of components requires an understanding of service conditions and mechanical testing and design. While there are many types of components tests for a multitude of products, this Chapter focus primarily on the basic principles for some common types of components used in our project. Using standard test methods provides for consistent test results. The mechanical evaluation of components[5] requires us to use many sources of information. It also requires an understanding of service conditions, design, and manufacturing variables. All these variables can make it difficult to validate components.

5.1 Arduino Board

After making the necessary connections between the Arduino board, Breadboard and the tester LEDs, upload the following code in the Arduino microcontroller.

Code to test the arduino board and the breadboard.

Code 5.1 Code to test Arduino Board

```
int red1pin=4;          //Defining the pins
int red2pin=7;
int red1time;
int red2time;
int red1num;
int red2num;

void setup() {
        // put your setup code here, to run once:
        pinMode(red1pin,OUTPUT);                //Setting the pins as OUTPUT
        pinMode(red2pin,OUTPUT);
        Serial.begin(9600);
        Serial.println("input red1 number: ");
        while(Serial.available()==0){}
        red1num=Serial.parseInt();
```

```
Serial.println("input red1 time: ");
        while(Serial.available()==0){}          //Waiting for input
        red1time=Serial.parseInt();

        Serial.println("input red2 number: ");
        while(Serial.available()==0){}                  //Waiting for input
        red2num=Serial.parseInt();

        Serial.println("input red2 time: ");
        while(Serial.available()==0){}                  //Waiting for input
        red2time=Serial.parseInt();
}

void loop() {
  // code here, to run repeatedly:
        for(inti=0;i<red1num;i++){                              //Lighting the LED1
                digitalWrite(red1pin,HIGH);
              delay(red1time);
              digitalWrite(red1pin,LOW);
              delay(red1time);
           }

        for(inti=0;i<red2num;i++){                              //Lighting the LED2
              digitalWrite(red2pin,HIGH);
              delay(red2time);
              digitalWrite(red2pin,LOW);
              delay(red2time);
        }
}
```

INPUT:

input red1 number: 5

input red1 time:1000

input red2 number:10

input red2 time:500

OUTPUT:

After execution of the above code and mentioned input, the first LED blinked 5 times at an interval of 1 second and the second LED blinked 10 times at an interval of 0.5seconds each.

CONCLUSION:

Hence it was clear that the board was working fine.

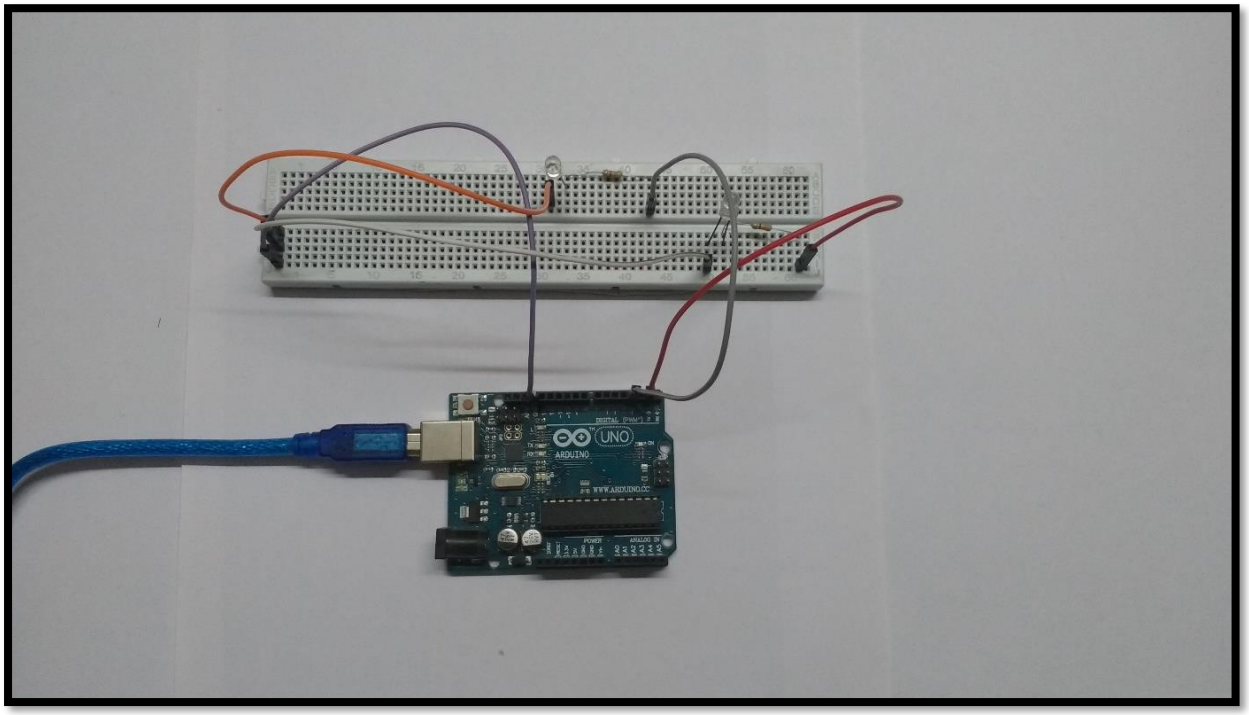


Figure 5.1 Connection to test Arduino Board

5.2 Motor Driver

After connecting the motor driver with the BO motors and making the precise connections between the microcontroller and motor drive, upload the following code in the aduinio microcontroller.

The code tests the functionality and compatibility of motors and motor driver with the microcontroller.

Code 5.2 Code to test Motor Driver

```
intmotorRB=12;        //Defining the motor pins.
intmotorRF=13;
intmotorLF=7;
intmotorLB=8;
intmotorRPWM=11;
intmotorLPWM=9;
charreadInput;



void setup() {
        Serial.begin(9600);
        pinMode(motorRF,OUTPUT);        // Setting the motor pins for output.
        pinMode(motorRB,OUTPUT);
        pinMode(motorLF,OUTPUT);
        pinMode(motorLB,OUTPUT);
        analogWrite(motorRPWM,255);
        analogWrite(motorLPWM,255);
}

void loop() {
        digitalWrite(motorRF,LOW);                //Instruction to Stop the Motors
        digitalWrite(motorRB,LOW);
        digitalWrite(motorLF,LOW);
        digitalWrite(motorLB,LOW);
```

```
        Serial.println("enter the direction of motion: ");
        while(Serial.available()==0){}
        readInput=Serial.read();

switch(readInput){

        case 'f' :      digitalWrite(motorRF,HIGH);     // Forward Instruction
                        digitalWrite(motorRB,LOW);
                        digitalWrite(motorLF,HIGH);
                        digitalWrite(motorLB,LOW);
                //analogWrite(motorRPWM,255);
                break;



case 'b' :

                digitalWrite(motorRF,LOW);      //Backward instruction
                digitalWrite(motorRB,HIGH);
                digitalWrite(motorLF,LOW);
                digitalWrite(motorLB,HIGH);
        break;
case 'l' :
                digitalWrite(motorRF,HIGH);     // Instruction to turn Left.
                digitalWrite(motorRB,LOW);
                digitalWrite(motorLF,LOW);
                digitalWrite(motorLB,LOW);
        break;
```

```
        case 'r' :      digitalWrite(motorRF,LOW);      //Instruction to turn Right.
                        digitalWrite(motorRB,LOW);
                        digitalWrite(motorLF,HIGH);
                        digitalWrite(motorLB,LOW);
                break;

        case 's' :      digitalWrite(motorRF,LOW);      //Instruction to Stop.
                        digitalWrite(motorRB,LOW);
                        digitalWrite(motorLF,LOW);
                        digitalWrite(motorLB,LOW);
                break;

        default:        Serial.println("wrong input. enterf,b,r,l or s");
                break;
  }
delay(5000);


}
```

INPUT:

enter the direction of motion: ***f***

OUTPUT:

Both the wheels started to rotate in forward direction.

CONCLUSION:

After executing the code, the motors and motor driver worked perfectly in synchronization with the arduino microcontroller.

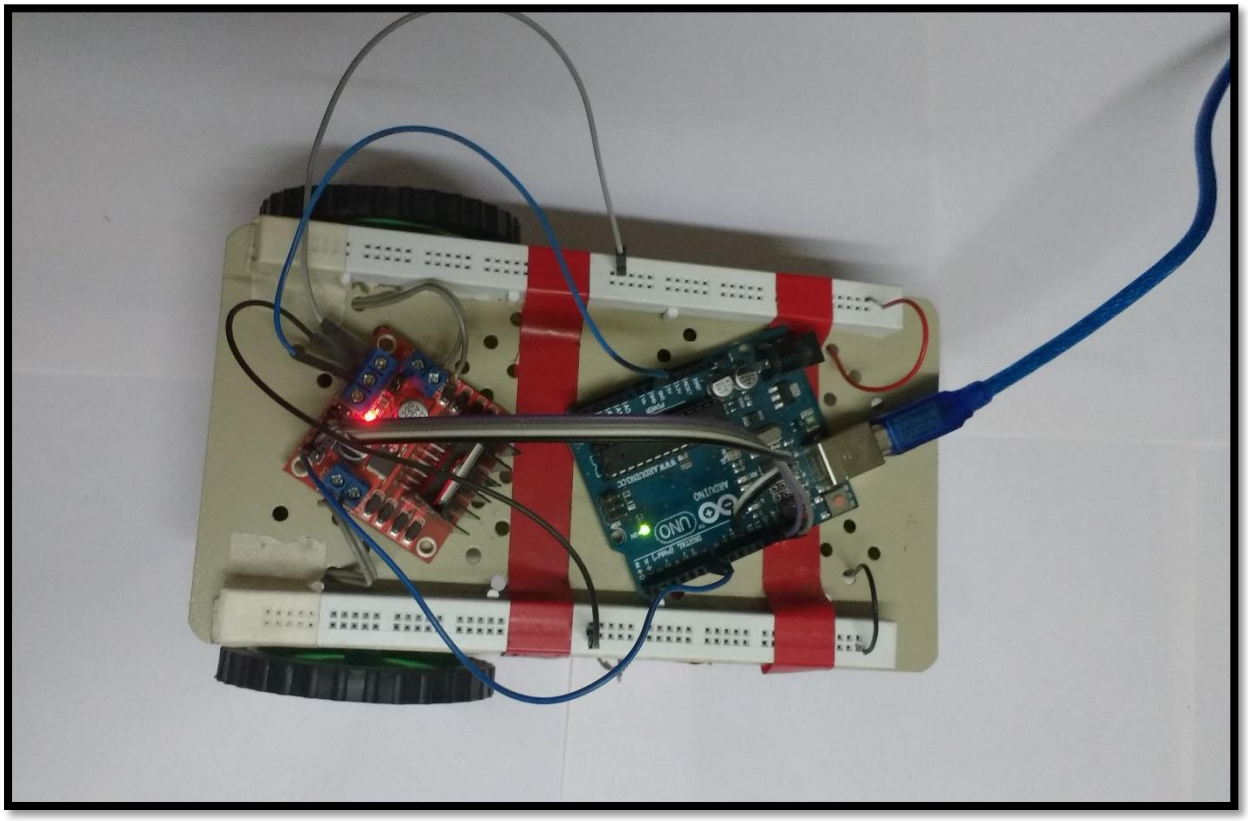


Figure 5.2 Connection to test Motor Driver

5.3DTMF decoder

Connect the DTMF decoder, Arduino board and the motor drivers precisely according to the pins mentioned in the code below. Upload the code in the microcontroller to test the working of the decoder.

Code 5.3 Code to test DTMF Decoder

```
constint inputPin1 = 2;  // Defining Digital Input Pins from DTMF Module
constint inputPin2 = 3;
constint inputPin3 = 4;
constint inputPin4 = 5;

int Sop1 = 0;     // Defining variable to store the status(HIGH/LOW) of above inputs.
int Sop2 = 0;
int Sop3 = 0;
int Sop4 = 0;
intoldCon = 0;  //  Variable to know what was the last button pressed.

void setup(){
pinMode(inputPin1, INPUT);  // Defining pins as input.
pinMode(inputPin2, INPUT);
pinMode(inputPin3, INPUT);
pinMode(inputPin4, INPUT);
Serial.begin(9600);  // Setup Serial Communication.

}

void loop(){
  Sop1 = digitalRead(inputPin1);  // Reading status of Input Pins. It can be LOW or HIGH
  Sop2 = digitalRead(inputPin2);
```

```
  Sop3 = digitalRead(inputPin3);
  Sop4 = digitalRead(inputPin4);


if(Sop4==LOW && Sop3==LOW && Sop2==LOW && Sop1==HIGH )   // Condition for Button 1. It is equal to Binary - 0001
  {
if (oldCon!=1){       // Here we are testing that what was the last pressed button,


Serial.println("1");


      // Your Action goes here.                    // You can add your set of action here.


   }
oldCon=1;
  }
else if(Sop4==LOW && Sop3==LOW && Sop2==HIGH && Sop1==LOW )
  // Condition for Button 2. It is equal to Binary - 0010
   {
if (oldCon!=2){
Serial.println("2");
        // Your Action goes here.
      }
oldCon=2;
   }
else if(Sop4==LOW && Sop3==LOW && Sop2==HIGH && Sop1==HIGH )  // Condition for Button 3. It is equal to Binary - 0011
   {
if (oldCon!=3){
Serial.println("3");
        // Your Action goes here.
```

```
    }
oldCon=3;
  }
else if(Sop4==LOW && Sop3==HIGH && Sop2==LOW && Sop1==LOW )  // Condition for Button 4. It is equal to Binary - 0100
  {
if (oldCon!=4){
Serial.println("4");
      // Your Action goes here.
    }
oldCon=4;
  }
else if(Sop4==LOW && Sop3==HIGH && Sop2==LOW && Sop1==HIGH )  // Condition for Button 5. It is equal to Binary - 0101
     {
if (oldCon!=5){
Serial.println("5");
      // Your Action goes here.
    }
oldCon=5;
  }


else if(Sop4==LOW && Sop3==HIGH && Sop2==HIGH && Sop1==LOW )  // Condition for Button 6. It is equal to Binary - 0110
  {
if (oldCon!=6){
Serial.println("6");
      // Your Action goes here.
    }
oldCon=6;
  }
```

```
else if(Sop4==LOW && Sop3==HIGH && Sop2==HIGH && Sop1==HIGH )  // Condition for Button 7. It is equal to Binary - 0111
   {
if (oldCon!=7){
Serial.println("7");
        // Your Action goes here.
      }
oldCon=7;
   }
else if(Sop4==HIGH && Sop3==LOW && Sop2==LOW && Sop1==LOW )   // Condition for Button 8. It is equal to Binary - 1000
   {
if (oldCon!=8){
Serial.println("8");
        // Your Action goes here.
      }
oldCon=8;
   }
else if(Sop4==HIGH && Sop3==LOW && Sop2==LOW && Sop1==HIGH )  // Condition for Button 9. It is equal to Binary - 1001
   {
if (oldCon!=9){
Serial.println("9");
        // Your Action goes here.
      }
oldCon=9;
   }
else if(Sop4==HIGH && Sop3==LOW && Sop2==HIGH && Sop1==LOW )  // Condition for Button 0 (10). It is equal to Binary - 1010
   {
if (oldCon!=10){
```

```
Serial.println("0");
        // Your Action goes here.
      }
oldCon=10;
   }
else if(Sop4==HIGH && Sop3==LOW && Sop2==HIGH && Sop1==HIGH )  // Condition for Button * (11). It is equal to Binary - 1011
   {
if (oldCon!=11){
Serial.println("*");
        // Your Action goes here.
      }
oldCon=11;
   }
else if(Sop4==HIGH && Sop3==HIGH && Sop2==LOW && Sop1==LOW )  // Condition for Button # (12). It is equal to Binary - 1100
   {
if (oldCon!=12){
Serial.println("#");
        // Your Action goes here.
      }
oldCon=12;
   }
else if(Sop4==HIGH && Sop3==HIGH && Sop2==LOW && Sop1==HIGH )  // Condition for Button A (13). It is equal to Binary - 1101
   {
if (oldCon!=13){
Serial.println("A");
        // Your Action goes here.
      }
oldCon=13;
```

```
  }
else if(Sop4==HIGH && Sop3==HIGH && Sop2==HIGH && Sop1==LOW )   // Condition for Button B (14). It is equal to Binary - 1110
  {
if (oldCon!=14){
Serial.println("B");
      // Your Action goes here.
    }
oldCon=14;
  }
else if(Sop4==HIGH && Sop3==HIGH && Sop2==HIGH && Sop1==HIGH )   // Condition for Button C (15). It is equal to Binary - 1111
  {
if (oldCon!=15){
Serial.println("C");
      // Your Action goes here.
    }
oldCon=15;
  }
else if(Sop4==LOW && Sop3==LOW && Sop2==LOW && Sop1==LOW )   // Condition for Button D (0). It is equal to Binary - 0000
  {
if (oldCon!=0){
Serial.println("D");
      // Your Action goes here.
    }
oldCon=0;
  }
delay(50);   // Debounce Delay.
}
```

INPUT:

(Generating DTMF tones through an android application)

On Pressing: 1

OUTPUT:

"**1**" got printed on Serial monitor.

CONCLUSION:

After execution of the above code, data is displayed on the serial monitor according to the DTMF tones sent over the telephonic signals.

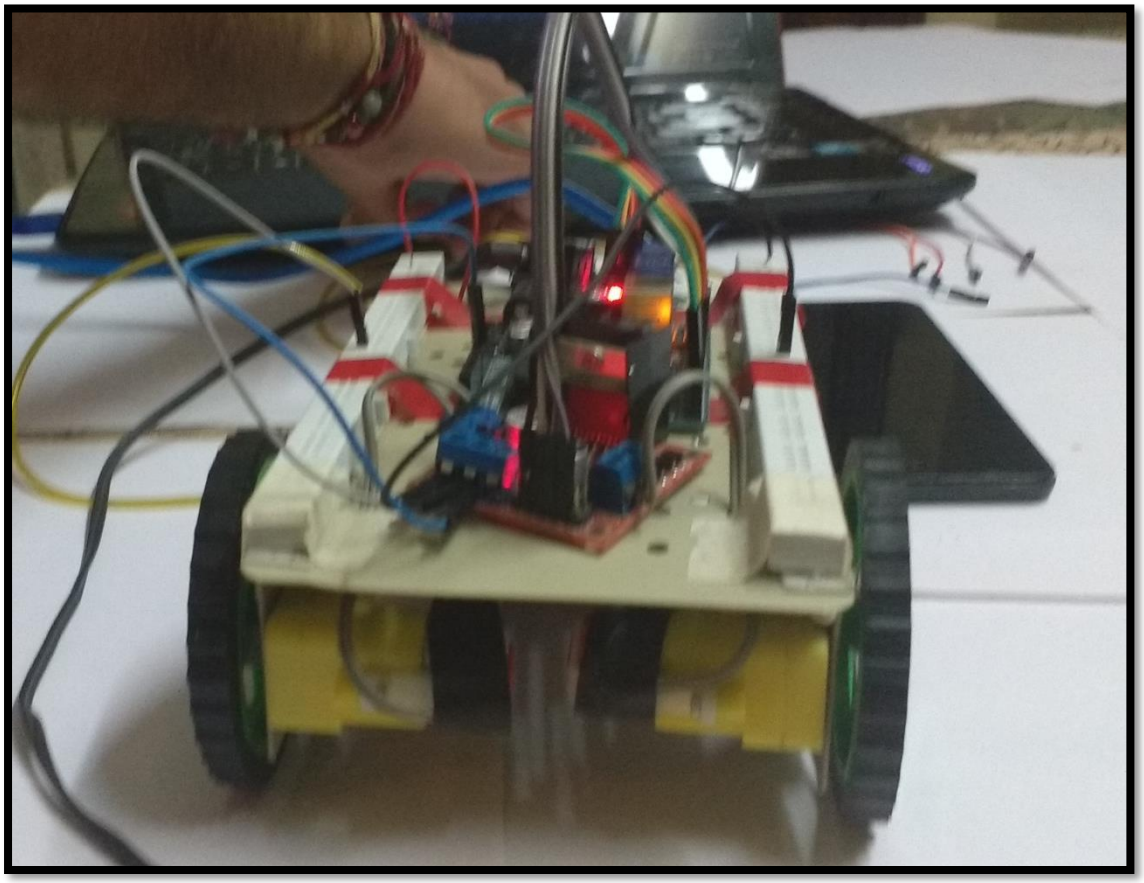


Figure 5.3 Connection to test DTMF Decoder

CHAPTER 06

IMPLEMENTATION

In order to control the robot, you need to make a call to the cell phone attached to the robot (through head phone) from any phone, which sends DTMF tunes on pressing the numeric buttons. The cell phone in the robot is kept in 'auto answer' mode. If the mobile does not have the auto answering facility, receive the call by 'ok' key on the vehicle connected mobile end then made it in hands-free mode. So after a ring, the cell phone accepts the call.

Now you may press any button on your mobile to perform actions as listed below:

The DTMF tones thus produced are received by the cell phone in the robot. These tones are fed to the circuit by the headset of the cell phone. The MT8870 decodes the received tone and sends the equipment binary number to the microcontroller, the robot starts moving. When you press key '2' (binary equivalent 00000010) on your mobile phone, the microcontroller outputs '10001001' binary equivalent. Port pins PD0, PD3 and Pd7 are high. The high output at PD7 of the microcontroller drivers the motor driver (L298N). Port pins PD0 and PD3 drive motors M1 and M2 in forward direction. Similarly, motors M1 and M2 move for left turn, right turn, backward motion and stop condition.

Final code to run the robot using the telephonic call

The code adds the functionality on the following dtmf tones for 3 seconds. Also, if connected to a serial port, it displays the input and running functionality:

On pressing "**1**": The robot turns **LEFT** and displays **1L** on serial port.

On pressing "**2**": The robot moves **FORWARD** and displays **2F** on serial port.

On pressing "**3**": The robot turns **RIGHT** and displays **3R** on serial port.

On pressing "**5**": The robot turns **STOP** and displays **5S** on serial port.

On pressing "**7**": The robot turns **LEFT BACK** and displays **7LB** on serial port.

On pressing "**9**": The robot turns **RIGHT BACK** and displays **9RB** on serial port.

On pressing "**#**": The robot moves **BACKWARD** and displays **#B** on serial port.

Code 6.1 Our Implementation

```
const int inputPin1 = 8;                    // Defining Digital Input Pins from DTMF Module
const int inputPin2 = 9;
const int inputPin3 = 10;
const int inputPin4 = 11;

const int motorRF=2;              //Output for motors
const int motorRB=3;
const int motorLF=4;
const int motorLB=5;
const int motorRPWM=6;
const int motorLPWM=9;

int Sop1 = 0;                               // Defining variable to store the status(HIGH/LOW)
of above inputs.
int Sop2 = 0;
int Sop3 = 0;
int Sop4 = 0;
int oldCon = 0;  //  Variable to know what was the last button pressed.

void setup(){
  pinMode(motorRF,OUTPUT);    //Defining pins for motor output
  pinMode(motorRB,OUTPUT);
  pinMode(motorLF,OUTPUT);
  pinMode(motorLB,OUTPUT);
  analogWrite(motorRPWM,255);
  analogWrite(motorLPWM,255);

  pinMode(inputPin1, INPUT);  // Defining pins as input.
```

```
  pinMode(inputPin2, INPUT);
  pinMode(inputPin3, INPUT);
  pinMode(inputPin4, INPUT);
  Serial.begin(9600);   // Setup Serial Communication.

}

void loop(){
  Sop1 = digitalRead(inputPin1);  // Reading status of Input Pins. It can be LOW or HIGH
  Sop2 = digitalRead(inputPin2);
  Sop3 = digitalRead(inputPin3);
  Sop4 = digitalRead(inputPin4);

       Serial.println("stop");    //Stop after 2 seconds
      digitalWrite(motorRF,LOW);
      digitalWrite(motorRB,LOW);
      digitalWrite(motorLF,LOW);
      digitalWrite(motorLB,LOW);


  if(Sop4==LOW && Sop3==LOW && Sop2==LOW && Sop1==HIGH )   // Condition for
Button 1. It is equal to Binary - 0001
  {
    if (oldCon!=1){                          // Here we are testing that what was the last
pressed button,

      Serial.println("1L");                    // Condition to move LEFT
      digitalWrite(motorRF,HIGH);
      digitalWrite(motorRB,LOW);
      digitalWrite(motorLF,LOW);
```

```
    digitalWrite(motorLB,LOW);

    delay(2000);

    // Your Action goes here.                    // You can add your set of action here.
  }

  oldCon=1;
 }
 else if(Sop4==LOW && Sop3==LOW && Sop2==HIGH && Sop1==LOW )  // Condition for Button 2. It is equal to Binary - 0010
  {
   if (oldCon!=2){
     Serial.println("2F");

     digitalWrite(motorRF,HIGH);                // Condition to move FORWARD
     digitalWrite(motorRB,LOW);
     digitalWrite(motorLF,HIGH);
     digitalWrite(motorLB,LOW);

     delay(2000);
     // Your Action goes here.
    }

   oldCon=2;
  }
 else if(Sop4==LOW && Sop3==LOW && Sop2==HIGH && Sop1==HIGH )  // Condition for Button 3. It is equal to Binary - 0011
  {
   if (oldCon!=3){
```

```
      Serial.println("3R");
      digitalWrite(motorRF,LOW);              // Condition to move RIGHT
      digitalWrite(motorRB,LOW);
      digitalWrite(motorLF,HIGH);
      digitalWrite(motorLB,LOW);

      delay(2000);
      // Your Action goes here.
    }

   oldCon=3;
  }

 else if(Sop4==LOW && Sop3==HIGH && Sop2==LOW && Sop1==HIGH )  //
Condition for Button 5. It is equal to Binary - 0101
     {
   if (oldCon!=5){
      Serial.println("5S");
      digitalWrite(motorRF,LOW);              // Condition to STOP
      digitalWrite(motorRB,LOW);
      digitalWrite(motorLF,LOW);
      digitalWrite(motorLB,LOW);
      // Your Action goes here.
    }
   oldCon=5;
  }

 else if(Sop4==LOW && Sop3==HIGH && Sop2==HIGH && Sop1==HIGH )  //
Condition for Button 7. It is equal to Binary - 0111
  {
```

```
    if (oldCon!=7){
      Serial.println("7Lb");
      digitalWrite(motorRF,LOW);          // Condition to move LEFT BACK
      digitalWrite(motorRB,HIGH);
      digitalWrite(motorLF,LOW);
      digitalWrite(motorLB,LOW);

     delay(2000);
     }
    oldCon=7;
  }

  else if(Sop4==HIGH && Sop3==LOW && Sop2==LOW && Sop1==HIGH )  // Condition for Button 9. It is equal to Binary - 1001
  {
    if (oldCon!=9){
      Serial.println("9");
      Serial.println("9Rb");
      digitalWrite(motorRF,LOW);          // Condition to move RIGHT BACK
      digitalWrite(motorRB,LOW);
      digitalWrite(motorLF,LOW);
      digitalWrite(motorLB,HIGH);

      delay(2000);
     }
    oldCon=9;
  }

  else if(Sop4==HIGH && Sop3==HIGH && Sop2==LOW && Sop1==LOW )  // Condition for Button # (12). It is equal to Binary - 1100
```

```
  {
    if (oldCon!=12){
      Serial.println("#B");

      digitalWrite(motorRF,LOW);          // Condition to move BACKWARD
      digitalWrite(motorRB,HIGH);
      digitalWrite(motorLF,LOW);
      digitalWrite(motorLB,HIGH);

      delay(2000);
      // Your Action goes here.
     }

    oldCon=12;
  }


delay(50);                          // Debounce Delay.

}
```

Result

After execution the robot did the specified functionality for 3 second and then stopped again and went into wait state for the next input.

CHAPTER 7

ANDROID

Android[3] is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

Android has the largest installed base of all operating systems (OS) of any kind. Android has been the bestselling OS on tablets since 2013, and on smartphones it is dominant by any metric.

Initially developed by Android, Inc., which Google bought in 2005,Android was unveiled in 2007 along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications ("apps") published – including many "business-class apps" that rival competing mobile platforms – and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform, which is comparable to Apple's iOS on 37% with both platforms far above others. In September 2015, Android had 1.4 billion monthly active devices.

Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. The success of Android has made it a target for patent (and copyright) litigation as part of the so-called "smartphone wars" between technology companies.

Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language that has complete access to the Android APIs. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support; the Go programming language is also supported since its version 1.4, which can also be used exclusively although with a

restricted set of Android APIs. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. Other development tools are available, including a native development kit (NDK) for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an framework based on Apache Cordova for porting Chrome HTML 5 web applications to Android, wrapped in a native application shell.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers; as of July 2013, there are more than one million applications available for Android in Play Store. As of July 2013, 50 billion applications have been installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill.

Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open source licenses.

We will be developing an application to control the robot while learning about Android Application Development and following a carefully planned procedure.

7.1Learning Phase – Android Application

7.1.1 Hello World APP

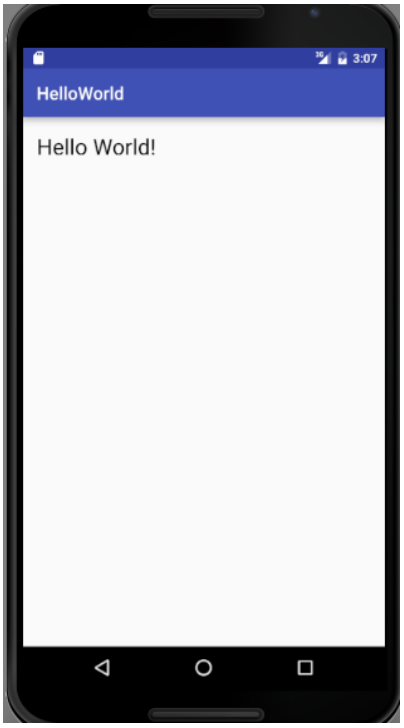


Figure 7.1 Hello World App Screen

7.1.2 A Simple Calculator

This app[6] illustrates four operations i.e.Plus(+),Multiply(*),Subtraction(-),Divide(/) which to be performed on two numbers. These numbers are taken as input in the text field. The result of the operation will be displayed in thetext field.

Components of app:

- Button: There are four buttons which perform +,-,* and /.On the click of any of them, performs the selected operation on the two numbers.
- EditText:There are two EditTextFileds in which a user can enter a number.
- TextView:The Result of the operation is displayed in this TextView.

Listeners Used:

- OnClickListener:This is only listener interface which is used for event handling of a button.

Method Used

- OnClick():This method is a call back method which is in OnClickListener interfaceand call when we click on button and here we can perform a required operation .

Figure 7.2 Calculator App Screen

7.1.3 A Puzzle Game App

This app is of great interest in which a user is given with a set of unordered numbers ranging from 1 to 8. A maximum number of moves are specified which is 50 in this game[7].Within this particular limit a user have to arrange the numbers in ascending order. When the limit of 50 is reached and user is unsuccessful in completing the game,a pop up is generated .In this pop up a user is provided with two options i.e. either a user can finish a game or restart it.

Functioning of RESET Button

On the click of this button a user can get set of random numbers ranging from 1 to 8 every time a user clicks on it.

Components used:

- Button: There is great use of button in this app.The coloured boxes in green and blue colour are buttons. There is also a reset button. On the click of Coloured Button,a number on that button can move in left,right,upward and downward direction only if the empty is in any of the directions.
- TextView:Moves are specified in this TextView.It get incremented every time we click on a coloured button .No increment are there in the moves if the number on the button does not moves.

Listeners Used:

- OnClickListener:This is only listener interface which is used for event handling of a button.

Method Used

- OnClick():This method is a call back method which is in OnClickListener interface and call when we click on button and here we can perform a required operation .

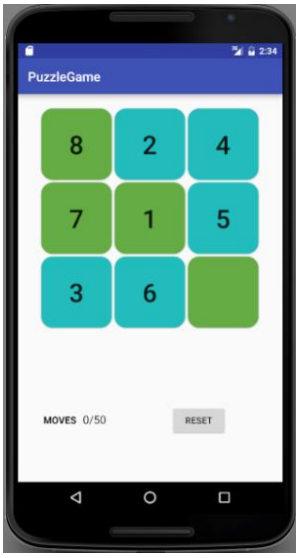


Figure 7.2 Puzzle App screen

7.2 Android Robo PGDAV App: Implementation

This app illustrates four functionig in the tab view ie Home , About , Bluetooth , Calling.The Home and About tabs gives us the describtion about the app.With the help of calling feature we can call anyone and bluetooth helps us to enable the bluetooth of our phone without going to the setting option of our phone.

TabLayout

The Four options ie Home,About,Calling and Bluetooth are in the TabLayout.By Clicking on any of them we can switch to different activites .

Classes Used in TabLayout:

- TabHost and TabHost.TabSpec :With these classes we can display what text we want to be displayed and also we can do some sort of editing.

SplashScreen

In this we use the concept of multithreading in which we start a thread containing an image and stop it for few seconds and then it disappears.

Permissions Used

We specify the permissions in the manifest file of the app.There are two main permissions used.These are:

- Bluetooth Permission:

<**uses-permission android:name="android.permission.BLUETOOTH"**/>
<**uses-permission android:name="android.permission.BLUETOOTH_ADMIN"**/>

- Calling permission:
<**uses-permission android:name="android.permission.CALL_PHONE"** />

7.2.1 Activity JAVA File

Code 7.1 Activity JAVA File

```
package com.example.hp.roboapp;

import android.Manifest;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.webkit.WebView;
import android.widget.TabHost;
import android.widget.TextView;
import android.widget.*;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.view.View.OnClickListener;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;

import android.graphics.Color;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.bluetooth.*;

import java.util.ArrayList;
import java.util.Set;
```

```
import android.net.Uri;


public class MainActivity extends AppCompatActivity {

    Button b1, b2, b3, b4;                  //buttons used with classes as Button

    BluetoothAdapter BA;                    //BlueToothAdapter class used for bluettoth
that allow device to wirelessly exchange                                    //data with
other bluetooth devices or communicate with bluetooth

    Set<BluetoothDevice> pairedDevices;

    ListView lv;                            //ListView Gives us the list of devices which can be
connected to our bluetooth

                                            //class used is ListView Class


   EditText edittext1;                      //Editext is a class and we are creating an instance of
this class

    Button button1;


    TabHost host;                           //It is a class that holds all the created activities
in the project like Home,About,Calling                                      //and Bluetooth
into a single frame Layout and developer can add multiple tab menu
                                //on application and each tab holds an activity showing their content


    TabHost.TabSpec spec;                   //It is an intent use to launch an activity as the
tab content


     View tabView;                          //


    @Override                               //onCreate() method is the first method called when
acitivity is first created

    protected void onCreate(Bundle savedInstanceState) {
```

```
   super.onCreate(savedInstanceState);                  //savedInstanceState saves the previous state of acitivity and pass as argumet in
         //constructor

      setContentView(R.layout.activity_main);       //It Binds the XML file to java file


      edittext1 = (EditText) findViewById(R.id.editText1); //findViewById() is a method with which we can access that view or widget into
//the java file and edittext1 is the unique id which we have created in XML file


     button1 = (Button) findViewById(R.id.button1);       //button1 is the unique id which we have created in XML file and we can access                                                                  //the button with the help of button1 object



      //bluetooth

      b1 = (Button) findViewById(R.id.button);

      b2 = (Button) findViewById(R.id.button2);

      b3 = (Button) findViewById(R.id.button3);

      b4 = (Button) findViewById(R.id.button4);


      BA = BluetoothAdapter.getDefaultAdapter();          //getDefaultAdapter() : this is your starting point for all Bluetooth actions.

      lv = (ListView) findViewById(R.id.listView);

      BA = BluetoothAdapter.getDefaultAdapter();

      //tabview

      host = (TabHost) findViewById(R.id.tabHost);

      host.setup();


      spec = host.newTabSpec("Home");

      spec.setContent(R.id.tab1);

      spec.setIndicator("Home");     //this is used for what we are specifying in the Tab

      host.addTab(spec);                  //This is used for adding the tab to the tabLayout
```

```
    TextView x = (TextView)
host.getTabWidget().getChildAt(0).findViewById(android.R.id.title);

                                //here we have specified some additional modifications ie how
beauiful the conntent appear in                                //TabLayout

    x.setTextSize(13);

    x.setTextColor(getResources().getColor(R.color.material_blue_grey_950));

    x.setSingleLine();



    tabView = host.getTabWidget().getChildTabViewAt(0);

    tabView.setPadding(-2, -2, -2, -2);



    spec = host.newTabSpec("Purpose");

    spec.setContent(R.id.tab2);

    spec.setIndicator("Purpose");

    host.addTab(spec);

    TextView x1 = (TextView)
host.getTabWidget().getChildAt(1).findViewById(android.R.id.title);

    x1.setTextSize(13);

    x1.setTextColor(getResources().getColor(R.color.material_blue_grey_950));

    x1.setSingleLine();

    tabView = host.getTabWidget().getChildTabViewAt(1);

    tabView.setPadding(-2, -2, -2, -2);



    spec = host.newTabSpec("Bluetooth");

    spec.setContent(R.id.tab3);

    spec.setIndicator("Bluetooth");

    host.addTab(spec);
```

```
    TextView x2 = (TextView)
host.getTabWidget().getChildAt(2).findViewById(android.R.id.title);
    x2.setTextSize(13);
    x2.setTextColor(getResources().getColor(R.color.material_blue_grey_950));
    x2.setSingleLine();
    tabView = host.getTabWidget().getChildTabViewAt(2);
    tabView.setPadding(-2, -2, -2, -2);



    spec = host.newTabSpec("Calling");
    spec.setContent(R.id.tab4);
    spec.setIndicator("Calling");
    host.addTab(spec);
    TextView x3 = (TextView)
host.getTabWidget().getChildAt(3).findViewById(android.R.id.title);
    x3.setTextSize(13);
    x3.setTextColor(getResources().getColor(R.color.material_blue_grey_950));
    x3.setSingleLine();
    tabView = host.getTabWidget().getChildTabViewAt(3);
    tabView.setPadding(-2, -2, -2, -2);



    b1.setOnClickListener(new View.OnClickListener() {



      @Override
      public void onClick(View v) {
        //Toast.makeText(getApplicationContext(),"H",Toast.LENGTH_LONG).show();
        if (!BA.isEnabled()) {
          //Toast.makeText(getApplicationContext(),"Turned
on",Toast.LENGTH_LONG).show();
```

```
            Intent turnOn = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);

                        //ACTION_REQUEST_ENABLEShow a system activity that allows
the user to turn on Bluetooth.


        startActivityForResult(turnOn, 0);


            Toast.makeText(getApplicationContext(), "Turned on",
Toast.LENGTH_LONG).show();

                        //Toast is a class which is used to dispaly a message for a short
duration

                        //.show() method helps us to dispaly the toast

                        //getApplicationContext() gives the refrence of current class

            } else {

            Toast.makeText(getApplicationContext(), "Already on",
Toast.LENGTH_LONG).show();


            }

        }

    });

                //setOnClickListener() method is used to register a button for event handling
ie Click on a button

    b2.setOnClickListener(new View.OnClickListener() {

        @Override

                //onClick is a callback method which is notified when we click on a button
and we can perform the required action              //here

        public void onClick(View v) {

            Intent getVisible = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

                //ACTION_REQUEST_DISCOVERABLE : Show a system activity that
requests discoverable mode. This activity will also              //request the user to turn
on  Bluetooth if it is not currently enabled.
```

```
        startActivityForResult(getVisible, 0);

                //Sometimes we want to get a result back from an activity when it ends so we
use startActivityForResult

        }


    });


    b3.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {


            pairedDevices = BA.getBondedDevices(); //Return the set of BluetoothDevice
objects that are bonded (paired) to the local
        //adapter.

            ArrayList list = new ArrayList();           //ArrayList is class that list all the
devices which are paired and ArrayList can support
        //daynamic arrays that can grow as needed


            for (BluetoothDevice bt : pairedDevices)

               list.add(bt.getName());

            Toast.makeText(getApplicationContext(), "Showing Paired Devices",
Toast.LENGTH_SHORT).show();


            final ArrayAdapter adapter = new ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_1, list);

            lv.setAdapter(adapter);

        }


    });


    b4.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {


            BA.disable();               //Turn off the local Bluetooth adapter—do not use
without explicit user action to turn off                                    //Bluetooth
            Toast.makeText(getApplicationContext(), "Turned off",
Toast.LENGTH_LONG).show();


        }
    });



    button1.setOnClickListener(new OnClickListener() {


        @Override
        public void onClick(View arg0) {
            String number = edittext1.getText().toString();
            Intent i = new Intent(Intent.ACTION_CALL);


            i.setData(Uri.parse("tel:" + number));


                //These are the set of Permission that we specify explicitly for calling
purposes
            if (ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                // TODO: Consider calling
                //    ActivityCompat#requestPermissions
                // here to request the missing permissions, and then overriding
                //   public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                //                          int[] grantResults)
```

```
            // to handle the case where the user grants the permission. See the
documentation
            // for ActivityCompat#requestPermissions for more details.


            return;
          }
          //
Toast.makeText(getApplicationContext(),number,Toast.LENGTH_LONG).show();
           Toast.makeText(getApplicationContext(), "Turned offee",
Toast.LENGTH_LONG).show();
           startActivity(i);     //used to start the activity
        //Temporary toast used here for testing purposes
        }
      });
  }
}
```

7.2.2 XML File

Code 7.2 XML File

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                    //xmlns:android keeps all the attributes used ie width , hieght into a
android namespace
   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"
                    //layout_width specifies the witdth of the component ie match_parent
specifies the width of component          //to be equal to the size of the parent ie
equal to the size of screen
   android:layout_height="match_parent"
                    //layout_height specifies the hieght of the component  ie
match_parent specifies the height of          //component to be
equal to the size of the parent ie equal to the size of screen
   android:orientation="vertical"
                    //orientation allign all the components vertically ie one below the
other
   tools:context="com.example.hp.roboapp.MainActivity">

   <TabHost
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
                    //warp_content specifies the height of component to be equal to it's
own size
      android:id="@+id/tabHost">
                    //id attribute is used to explicitly giving the id to the components ie in
this case the id of TabHost is          //tabHost

      <LinearLayout
         android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical">

<TabWidget
   android:id="@android:id/tabs"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

</TabWidget>

<FrameLayout
   android:id="@android:id/tabcontent"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <LinearLayout
      android:id="@+id/tab1"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">
      <TextView
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Application Controlled Land Rover Robot"
               //android:text is used to display what text we want on the textview to
be displayed
         android:textAlignment="center"
               //android:textAlignment="center" is used to align the text in center
with respect to screen
         android:textColor="#000000"
```

```
                    //android:textColor="#000000" is used to give the colors to text ie black in this case
                android:textSize="20dp"
                    //android:textSize="20dp" is used to display the larger ie the size of the text
                android:padding="10dp"
                    //android:padding="10dp":used to give the internal space in a textview


                />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Robot control using the wireless communication "
                android:textSize="15dp"
                android:textColor="#000000"
                android:padding="10dp"
                android:textAlignment="textStart"
                />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="20dp"
                android:text="This project outlines the strategy adopted for establishing wireless communication between a Robot and a control device. Our aim is to be able to command and control the Robot wirelessly by the GUI Application. "
                android:textColor="#000000"
                android:padding="10dp"
                android:id="@+id/webview1"
                />
            <TextView
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="The principle task of this project was to program the Arduino microcontroller interfaced to a DTMF controller module which would enable us to wirelessly control the Robot."
            android:textSize="20dp"
            android:textColor="#000000"
            android:padding="10dp"
                        //android:padding="10dp":used to give the internal space in a textview


            />


        </LinearLayout>


        <LinearLayout
           android:id="@+id/tab2"
           android:layout_width="match_parent"
           android:layout_height="match_parent"
           android:orientation="vertical">
           <TextView
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:padding="20dp"
              android:textColor="#000000"
              android:textSize="20dp"
              android:text="Purpose of our research is to provide simpler robot's hardware architecture but with powerful computational platforms so that robot's designer can focus on their research and tests instead of Dtmf connection infrastructure. This simple architecture is also useful for educational robotics, because students can build their own robots with low cost and use them as platform for experiments in several courses."
              />
```

```
        </LinearLayout>


        <RelativeLayout
           android:id="@+id/tab3"
           android:layout_width="match_parent"
           android:layout_height="match_parent"
           android:orientation="vertical">



           <Button
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Turn On"
              android:id="@+id/button"
              android:layout_marginLeft="10dp"
                     //android:layout_marginLeft="10dp" : is used to give the space
outside the button
              android:layout_marginTop="20dp"
              android:layout_alignParentStart="true" />


           <Button
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Get Visible"
              android:id="@+id/button2"
              android:layout_alignTop="@+id/button"
              android:layout_marginRight="10dp"
              android:layout_alignParentEnd="true" />


           <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="List Devices"
        android:id="@+id/button3"
        android:layout_below="@+id/button"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="10dp"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Turn OFF"
        android:id="@+id/button4"
        android:layout_alignTop="@+id/button3"
        android:layout_alignStart="@+id/button2"
        android:layout_marginRight="10dp"
        />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/button3" />
</RelativeLayout>

<LinearLayout
    android:id="@+id/tab4"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <EditText
           android:id="@+id/editText1"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_marginTop="25dp"
           android:hint="Enter Mobile Mumber"
            />

        <Button
           android:id="@+id/button1"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:text="Call"
           android:layout_margin="20dp"
           />




          </LinearLayout>
        </FrameLayout>
      </LinearLayout>
    </TabHost>
</LinearLayout>
```

7.2.3 Splash Screen JAVA File

Code 7.3 Splash Screen JAVA File

```
package com.example.hp.roboapp;


import android.app.Activity;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;


public class SplashScreen extends Activity {


   @Override
                //onCreate() method is the first method called when acitivity is first created
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_splash_screen);
                //savedInstanceState saves the previous state of acitivity and pass as argumet in                                          //constructor
      Thread timerThread = new Thread()  //Here we have created the instance of the thread
       {
         public void run(){              //thread should be in try catch block
            try{
               sleep(4000);              //We have created the thread and sleep it for the desired amount of time
            }catch(InterruptedException e){
               e.printStackTrace();
            }finally{
               Intent intent = new Intent(SplashScreen.this,MainActivity.class);
```

```
                    //when the finally statement will execute the MainActivity class will start explicitly
                        startActivity(intent);
                    }
                }
            };
            timerThread.start(); //It is a method by which a thread can start and after this  run() method is invoked
        }
            //another overrided method is onPause() which is called when activity is in background or not in focus
        @Override
        protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            finish();
        }
}
```

7.2.4 Splash Screen XML FILE

Code 7.4 Splash Screen XML FILE

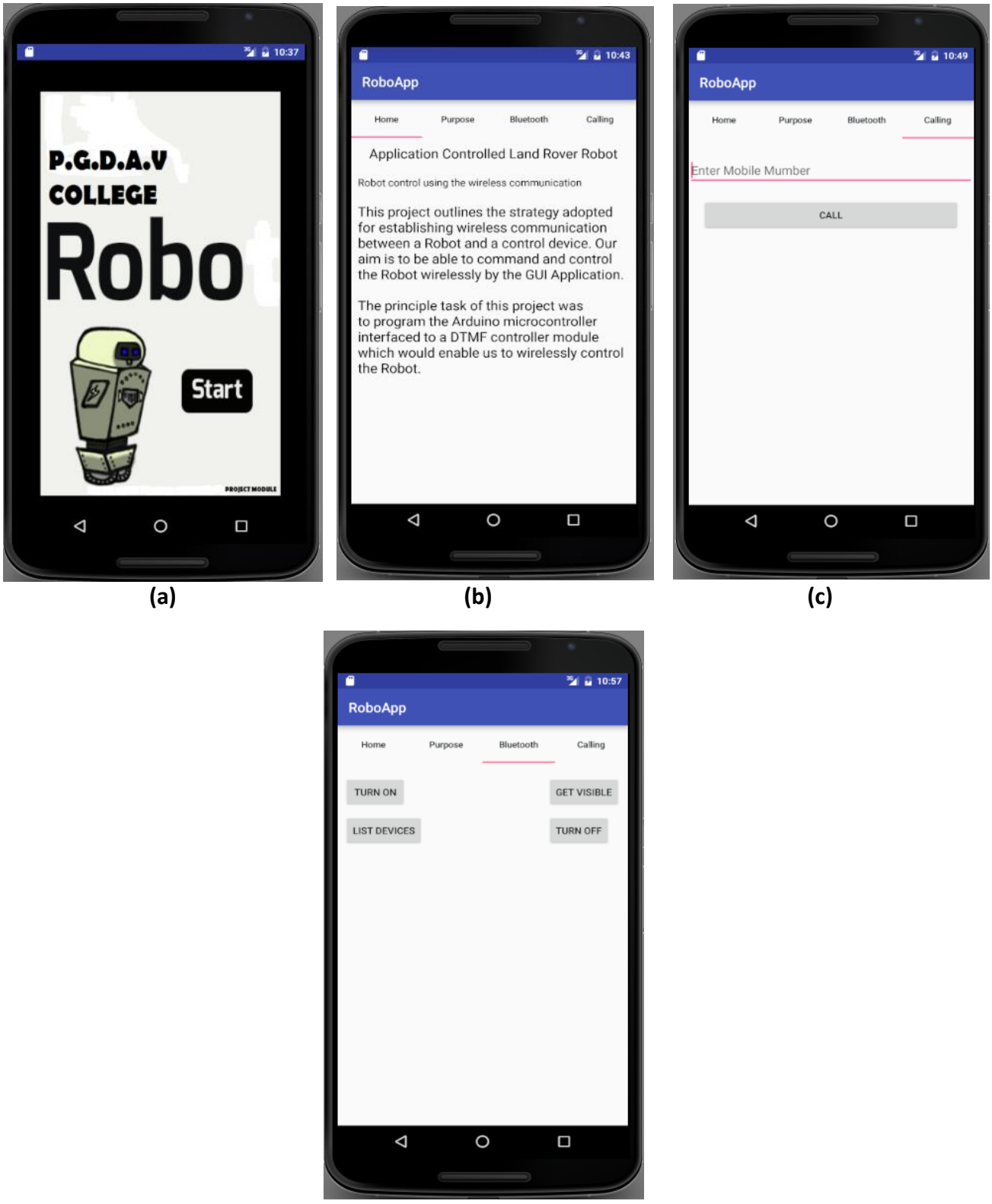
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"

tools:context="com.example.hp.roboapp.SplashScreen">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/robosplash">
//android:background="@drawable/robosplash" :is used to give the background to the linear layout

</LinearLayout>
</RelativeLayout>
```

7.2.5 Screen



(a) (b) (c)

(d)

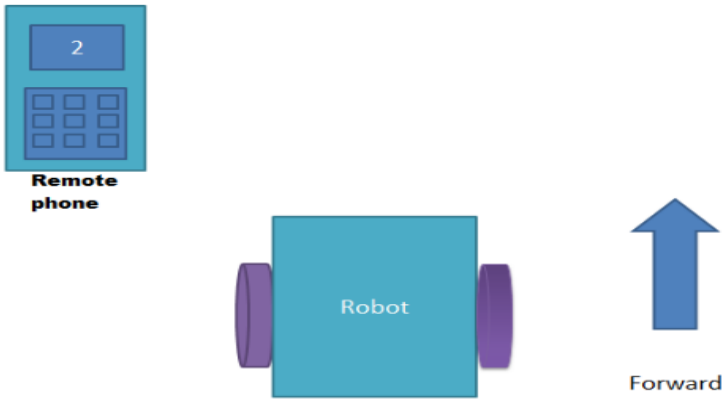
Figure 7.3 Robo App Screen

7.3 Working of Application

Robot is run by some commands that are send via mobile phone using the android app. We are here using DTMF function of mobile phone. One is user mobile phone that we will call 'Remote phone' and second one that are connected with Robot's circuit using aux wire. This mobile phone we will call 'Receiver Phone'.

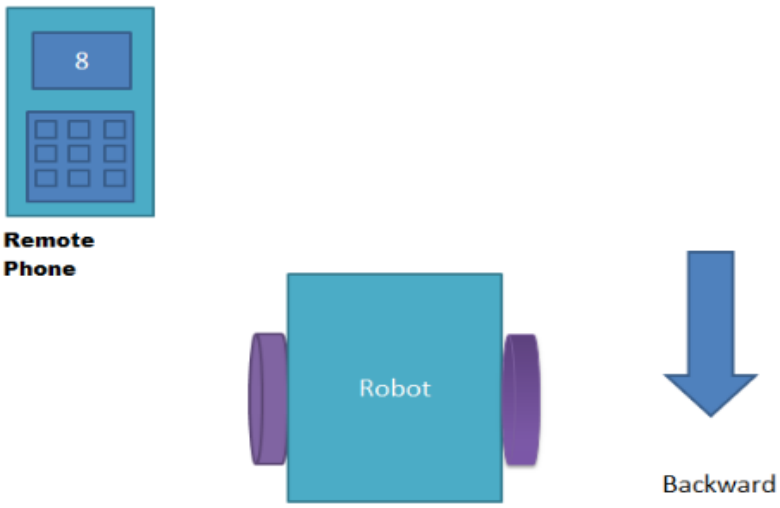
First we establish a connection between the two mobiles using the android app:

When we presses '2' by remote phone, robot start to moving forward.



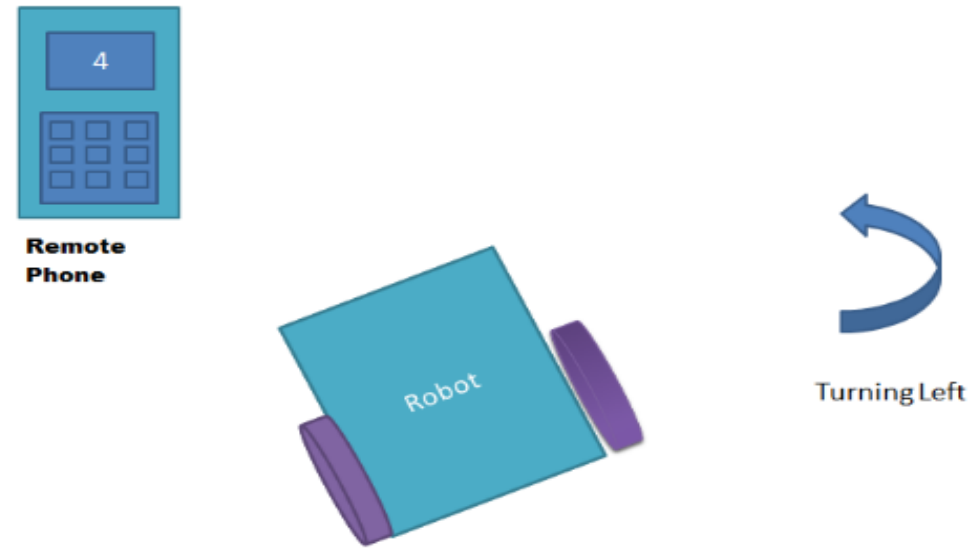
(a)

When we presses '8' by remote phone, robot change his state and start moving in backward direction.

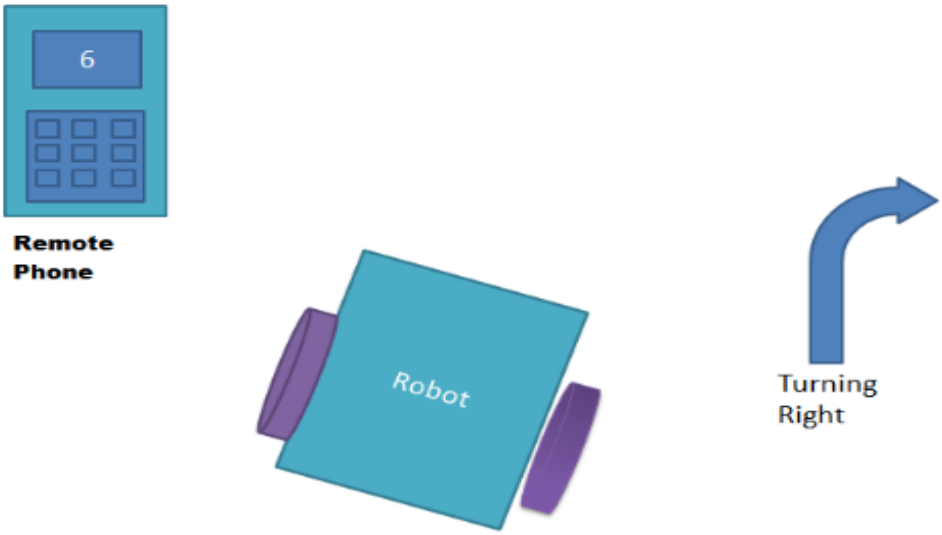


(b)

When we press ‘4’ or ‘6’, Robot get turn left or right respectively. On pressing 5, the movements of app stops.



(c)



(d)

Figure 8.1 Working of Robot (a)Moves forward on pressing 1

(b)Moves backward on pressing 8

(c)Moves left on pressing 4

(d)Moves right on pressing 6

CHAPTER 08
APPLICATIONS OF ROBOT

➢ Scientific

• Remote control vehicles have various scientific uses including hazardous environments, working in the deep ocean and space exploration. The majority of the probes to the other planets in our solar system have been remote control vehicles, although some of the more recent ones were partially autonomous. The sophistication of these devices has fuelled greater debate on the need for manned spaceflight and exploration.

 Military and Law Enforcement

• Military usage of remotely controlled military vehicles dates back to the first half of 20th century. Soviet Red Army used remotely controlled Teletanks during 1930s in the Winter War and early stage of World War II.

 Search and Rescue

• UAVs will likely play an increased role in search and rescue in the United States. This was demonstrated by the successful use of UAVs during the 2008 hurricanes that struck Louisiana and Texas.

 Recreation and Hobby

• See Radio-controlled model. Small scale remote control vehicles have long been popular among hobbyists. These remote controlled vehicles span a wide range in terms of price and sophistication. There are many types of radio controlled vehicles. These include on-road cars, off-road trucks, boats, airplanes, and even helicopters. The "robots" now popular in television shows such as Robot Wars, are a recent extension of this hobby (these vehicles do not meet the classical definition of a robot; they are remotely controlled by a human).

CHAPTER 09
FUTURE SCOPE

The knowledge is ever expanding and so are the problems which the mankind strive to solve. In this spirit, it is hoped that the current activity will lead to further enhancements.

➢ IR Sensors

• IR sensors can be used to automatically detect & avoid obstacles if the robot goes beyond line of sight. This avoids damage to the vehicle if we are manoeuvring it from a distant place.

➢ Password Protection

• Project can be modified in order to password protect the robot so that it can be operated only if correct password is entered. Either cell phone should be password protected or necessary modification should be made in the assembly language code. This introduces conditioned access & increases security to a great extent.

➢ Alarm Phone Dialler

• By replacing DTMF Decoder IC CM8870 by a 'DTMF Transceiver IC' CM8880, DTMF tones can be generated from the robot. So, a project called 'Alarm Phone Dialler' can be built which will generate necessary alarms for something that is desired to be monitored (usually by triggering a relay). For example, a high water alarm, low temperature alarm, opening of back window, garage door, etc.

• When the system is activated it will call a number of programmed numbers to let the user know the alarm has been activated. This would be great to get alerts of alarm conditions from home when user is at work.

CHAPTER 10

TROUBLESHOOTING

During our final year full time project we had to par many hurdles to complete our project we tackled it with the help of our mentor and with our knowledge with finally helped us to get the project done.

The different problems we faced were:-

 Since we lacked knowledge related to electronics, we had to study all about electronic components while choosing the components.

 While powering the DTMF decoder, we unintentionally gave large voltage to the decoder which lead to burning of chip. Hence we had t buy another DTMF Decoder.

 Due to improper connections the robot was operating in the reverse operations so we changed the wiring of it and it started operating.

 We were not able remotely to access the camera of the stacked smartphone. To overcome this problem we had to manipulate the software of the smartphone.

References/Bibliography

[1] Available: http://www.webopedia.com/TERM/R/robotics.html

[2]]Available: https://en.wikipedia.org/wiki/Robotics

[3] Available: https://en.wikipedia.org/wiki/Android_(operating_system)

[4] “Introduction to Robotics: Mechanics and Control” by John J. Craig

[5] “Probabilistic Robotics” by Sebastian Thrun, Wolfram Burgard, Dieter Fox

[6] “Android Programming: The Big Nerd Ranch Guide” (Big Nerd Ranch Guides) by Bill Philips & Brian Hardy

[7] “Beginning Android Games” by Mario Zechner

[8]"Robot control design based on smartphone" Control and Decision Conference (CCDC), 2013 25thChinese. IEEE, 2013.

[9]"Smartphone-controlled user calling system for a mobile robot" Robotics (ISR), 2013 44thInternational Symposium on. IEEE, 2013.