Tutorial

Overview: The basic aim of the project is to find an alternative to the usual RF circuits that are seen so frequently and replace them with cell-phone operated module. The advantages of a cell-phone operated bot are:-

1. A higher- in fact, limitless- range from which the bot can be controlled as compared to the RF circuits.
2. No interference with other devices as the communication takes place only between the caller and the cellphone connected to the bot.
3. An ability to incorporate 12 different kinds of commands(including the usual forward, backward, left, right and stop commands.) without making any additions to the remote i.e. by creating the appropriate code we can make our bot perform various functions using only our mobile phone.

Principle: A cellphone basically employs what is known as the DTMF (Dual Tone Multi Frequency) technology. Every button on the mobile phone is a superimposition of two waves of different frequencies. Thus every digit on the mobile phone is given its own identity.

	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

(The keys A,B,C and D are not usually found on mobile phones)

Components used:

IC1 -	MT8870 DTMF decoder
IC2 -	ATmega16 AVR microcontroller
IC3 -	L293D motor driver
IC4 -	74LS04 NOT gate
D1 -	1N4007 rectifier diode
R1, R2 -	100-kilo-ohm
R3 -	330-kilo-ohm
R4-R8 -	10-kilo-ohm
C1 -	0.47µF ceramic disk
C2, C3, C5, C6 -	22pF ceramic disk
C4 -	0.1µF ceramic disk
XTAL1 -	3.57MHz crystal
XTAL2 -	12MHz crystal
S1 -	Push-to-on switch
M1, M2 -	6V, 50-rpm geared DC motor
Batt. -	6V, 4.5Ah battery

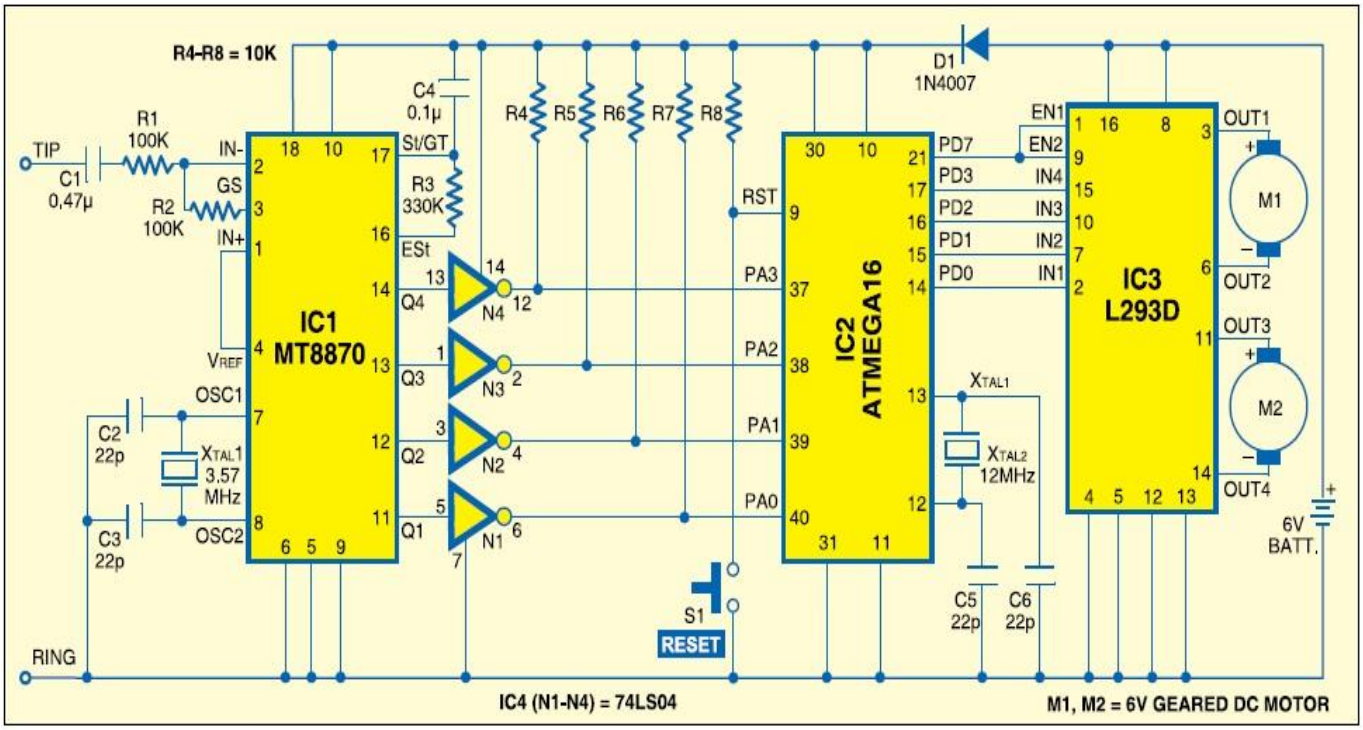
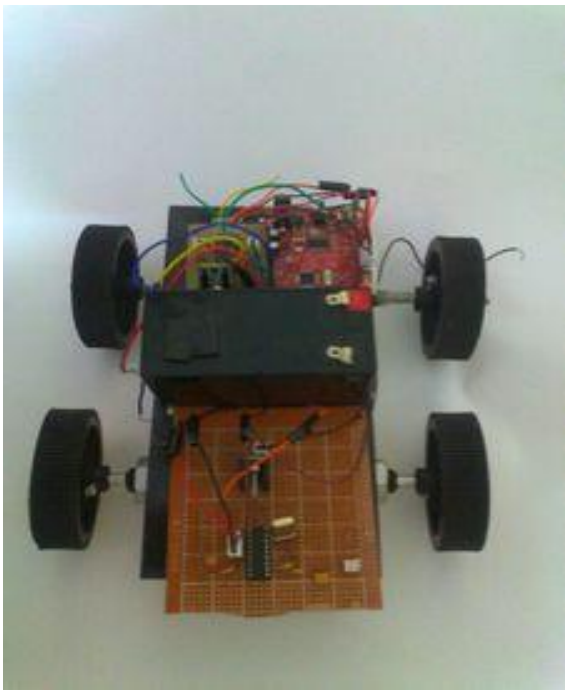


Fig. 2: Circuit diagram of microcontroller-based cellphone-operated land rover

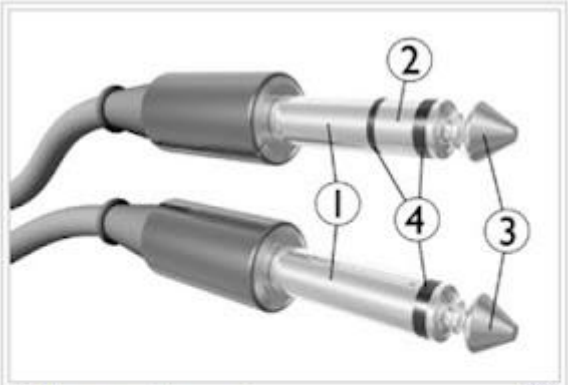
Construction and Execution: A mobile phone is attached to the circuit using a 3.5mm jack. A caller calls the mobile phone, which has to be accepted. Once the call is accepted, a caller can press any button his mobile phone. This is received on the other end by the cell phone and fed to the decoder circuit through the jack. The decoder circuit converts the number pressed into the corresponding binary equivalent through the 4 outputs. For example, if the caller presses the number '5', the pins 11 and 13 will be low but the pins 12 and 14 will be high, this corresponds to a 0101 configuration, which is the 4-bit binary equivalent of the decimal 5. This output is fed to the arduino, which is connected to the motor driver circuit. By using appropriate coding, the arduino can govern the motor driver circuit and thereby, drive the motors as required. The simple bot is operated with a differential driving mechanism; though it is possible to incorporate a number of mechanisms such as axial system as well as other functions such as a lifting mechanism quite easily.

Though the diagram here shows IC Atmega16 microcontroller, it is possible to use an arduino as well, which has an Atmgea 328 microcontroller.



Now, let's tackle the separate parts of the circuit sequentially.

Cellphone: Theoretically, any mobile phone can be connected to the circuit for the bot to work. However, all mobiles do not give a large enough output of the signal sent. Similarly, all calling mobiles do not give a large enough pulse width. We can check the magnitude of the signal received and pulse width by connecting the jack to the CRO. It is more pragmatic to have a mobile with the auto answer mode connected to the circuit as it saves the trouble of answer the call manually. We recommend the older models of Nokia such as Nokia 1100.



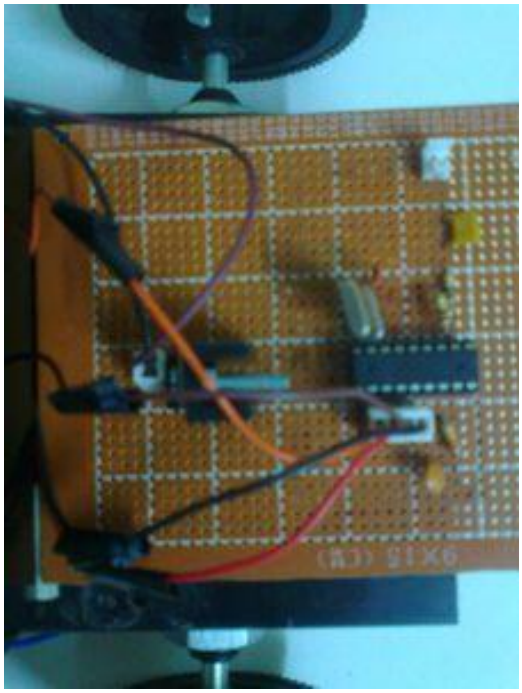
1. **Sleeve:** usually ground
2. **Ring:** Right-hand channel for stereo signals, negative polarity for balanced mono signals, power supply for power-requiring mono signal sources
3. **Tip**: Left-hand channel for stereo signals, positive polarity for balanced mono signals, signal line for unbalanced mono signals
4. **Insulating rings**

3.5mm jack: As shown in the figure, the jack consists of three parts-ring, ground and tip-and has three wires protruding from it each of which is connected to one of these three respectively. It should be ensured that the wires of tip and ground or tip and ring are not shorted by mistake.

Decoder Circuit: The most problem-prone area of the bot. A number of complications can arise in this region. Hence, it is necessary to have this circuit tested on a breadboard before proceeding with the soldering work on the pcb. Even then there are no guarantees. Both the ICs MT 8870 and HT 9170 can be tried and both are known to be untrustworthy. So make sure you buy a large number of them when you buy them. The same goes for crystals. The crystal has a quartz crystal inside which provides a clock pulse to the circuit. While testing the circuit on the breadboard, it should be ensured that the 22pf crystals are connected very close to the crystal and ground. As they have a very low capacitance, wiring capacitance can easily distort their value thereby making it imperative to ensure that the any wiring connections involving these capacitors are kept to a minimum. One should design the layout carefully before proceeding with the soldering.

Motor Driver Circuit: The usual motor driver circuit works perfectly and is usually not too erroneous. It's the easiest part of the circuit.

Rest of the bot: The rest of the bot's specifications such as the choice of motor, wheels, and chassis are entirely up to you. As mentioned before, you can also create a lifting mechanism on the bot or any other thing you can think of. It's quite easy to perform a large number of functions in this bot.



The pcb

Arduino and motor driver

Given below is the code used in the project.

CellPhone **Arduino Coding**

```
int value=0;
int input[]={2,3,4,5};
int output[]={8,9,10,11};

void setup()
{
  int thispin=0;
  for(thispin=0;thispin<4;thispin++)
  {
    pinMode (input[thispin],INPUT);
    pinMode (output[thispin],OUTPUT);
  }
}

void loop()
{
  value=digitalRead(2)+2*digitalRead(3)+4*digitalRead(4)+8*digitalRead(5);

  switch (value)
  {
    case 2:
      digitalWrite(8,HIGH);
      digitalWrite(9,LOW);
      digitalWrite(10,HIGH);
      digitalWrite(11,LOW);

      break;

     case 4:
      digitalWrite(8,LOW);
      digitalWrite(9,HIGH);
      digitalWrite(10,HIGH);
      digitalWrite(11,LOW);
      break;

     case 6:
      digitalWrite(8,HIGH);
      digitalWrite(9,LOW);
      digitalWrite(10,LOW);
      digitalWrite(11,HIGH);
      break;

     case 8:
      digitalWrite(8,LOW);
      digitalWrite(9,HIGH);
      digitalWrite(10,LOW);
      digitalWrite(11,HIGH);
      break;

     default:
      digitalWrite(8,LOW);
      digitalWrite(9,LOW);
      digitalWrite(10,LOW);
      digitalWrite(11,LOW);
      break;
  }
```

AS you can see, the code is fairly simple.