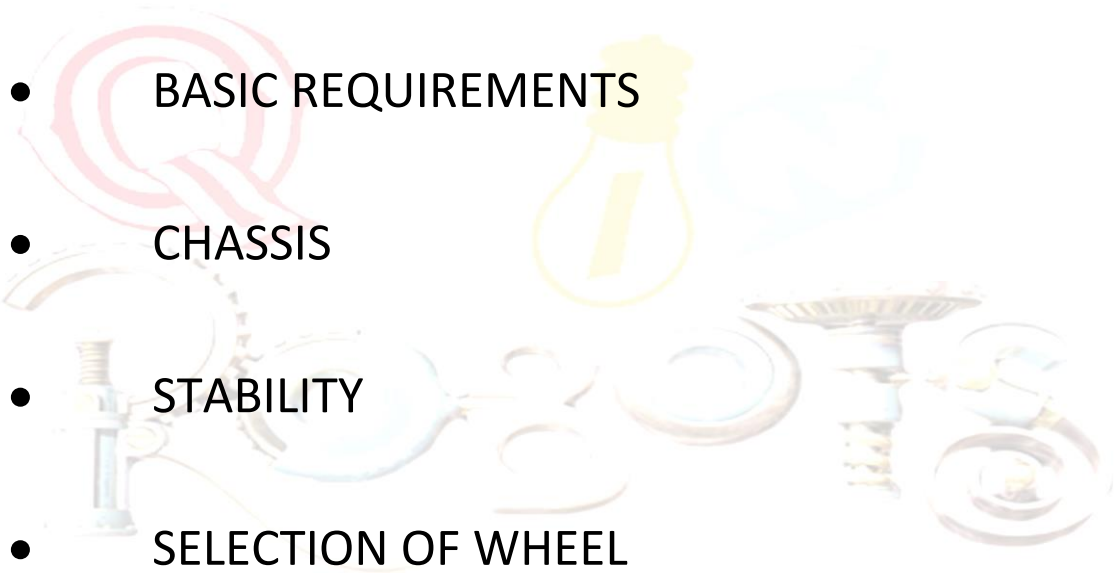


SMT. KASHIBAI NAVALE COLLEGE OF ENGINEERING



MECHANICAL SECTION

CONTENTS

- 
- BASIC REQUIREMENTS
 - CHASSIS
 - STABILITY
 - SELECTION OF WHEEL
 - TORQUE & RPM
 - MOTOR MOUNTING
 - OTHER TIPS

BASIC REQUIREMENTS

These are some of the most fundamental materials which are definitely needed before anyone begins with robot making:

- [Hack saw/ blade](#)
- [Screw drivers](#)
- [Spanner](#)
- [Driller \(little bit expensive, if you don't wish to buy you can always go to carpenter\)](#)
- [Base material](#)
- [Wheels + Castors](#)
- [Motors dc geared \(as per your need\)](#)
- [Nails, Screws n nuts](#)
- [Clamps](#)

CHASSIS

The chassis of a robot is the basic framed structure to which everything else is attached. It is probably the largest part of a robot, so make sure it is made of a light weight rigid material. Thus it is a part on which other mountings such as batteries, wheels, mechanism is done.

Properties of Material:

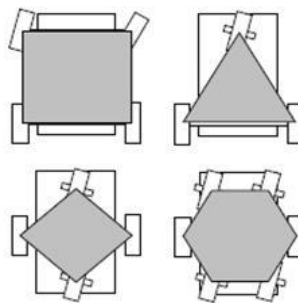
- Strength to carry weight
- Should be light
- Can be easily worked (drilled, cut)
- Easily available

Materials which you can use for building bOts:

1. Balsa wood: light weight, good choice for flying bots.
2. Aluminium: light and strong.
3. Acrylic sheet: cheaper than aluminium, easy to work.
4. MDF (Med. Density fibre) or any other easily available material.

STABILITY

The centre of gravity (cog) of robot should lie inside the polygon formed by the

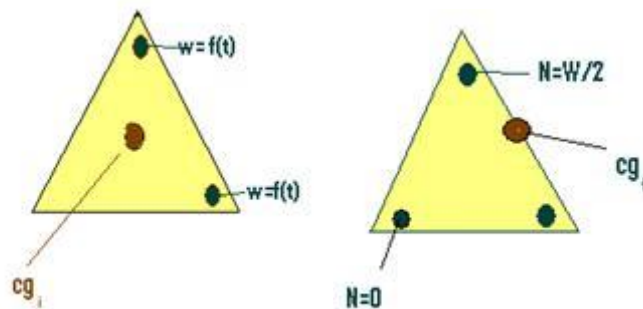


safety region for cog

wheels.

Let's take an example of a three wheeler to understand this concept. Let the polygon formed by the wheels be an equilateral. (REMEMBER the condition to topple is that the normal about any one point of contact become ZERO!!) Let

initially the cog is located at the centroid, if we start increasing weight equally on the two adjacent corners of the polygon, the normal on the third corner become zero.



- The cog should be as low as possible for better stability while accelerating/decelerating.
- Moving the cog can allow the bot to move across wider gaps, climb steeper slopes and get over or onto higher steps.

SELECTION OF WHEELS

Wheel diameter:

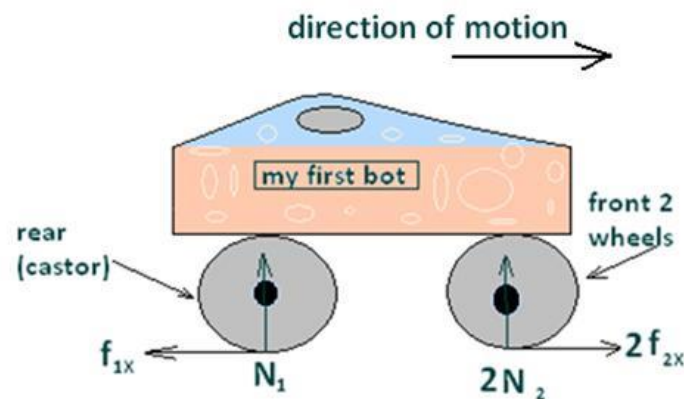
While buying (or making) wheels, motor consideration must be taken into an account. For a start, there is **torque** and **velocity**. Large diameter wheels give your robot low torque but high velocity. So if you already have a very strong motor, then you can use wheels with larger diameters. **Servo's** already have good torque, so you should use larger diameter wheels. But if your motor is weak (such as if it does not have any gearing), you want to use a much smaller diameter wheel. This will make a robot slower, but at least it has enough torque to go up a hill!

The texture of your wheel is very terrain dependent. If your wheel is **too smooth** then it will not have much friction. This is a serious issue with **omni-wheels**. So it is advisory to have a normal wheel with good friction characteristics.

TORQUE & RPM

- No of wheels = 3 with differential drive
- No of motors = 2
The bOt is front wheel driven and has a rear castor wheel.
- Coefficient of friction = 0.6 and radius of wheel = 3cm
- Weight of bot ($W=mg= 20N$) ($f=$ friction, $N=$ normal)

When the bOt is accelerating in the forward direction, the friction acts in the forward direction on the front wheels and on the rear castor wheel the friction acts in opposite direction (see figure). Assuming that we have an **impending motion** i.e. the torque is just sufficient to start the bOt.



Following are the equations of motion:-

$$\begin{aligned} N_1 + 2N_2 &= W \\ f_{2x} * r - \tau &= 0 \\ f_{2x} &= \mu N \end{aligned}$$

Since there is an impending motion hence

For a motor the impending torque should be able to overcome the static friction torque. After this, the torque required for rotating the wheels of bOt reduces and the rpm of motor will increase, hence the speed of bOt.

$$N = W/3 \text{ (considering that weight is equally distributed)}$$

$$\begin{aligned} \mu * W/3 * r - \tau &= 0 \\ 0.6 * (20/3) * 0.03 - \tau &= 0 \\ \tau &= 0.12N\cdot m \text{ or } 1.2kg\cdot cm \end{aligned}$$

Calculating the rpm of motor:

Speed of bot= rpm of motor* radius (while moving forward)

For e.g.

$$\begin{aligned} 0.5(\text{m/s}) &= \omega * 0.03(\text{m}) \\ \omega &= 17 (\text{rad/s}) = 17 * 60 / 2\pi (\text{rpm}) = 160 (\text{rpm}) \end{aligned}$$

Still rpm upto 300 is comfortable or rather controllable. If you feel that the rpm of your bot is difficult to handle then you should choose a lower voltage rating.

MOTOR MOUNTING

To mount any type of motor to your chassis you will need to use an **L shaped bracket**. For a DC motor, all you need to do is take a sheet of aluminum, drill two holes in two of the corners, drill two more holes on the other half to match the motor screw holes, then bend the entire piece in a 90 degree angle. This particular image I had found a U shaped piece of aluminum in some bin, I cut it to size, and just drilled the appropriate holes to attach it to my white HDPE chassis.



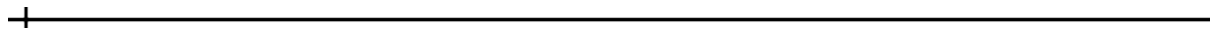
OTHER TIPS

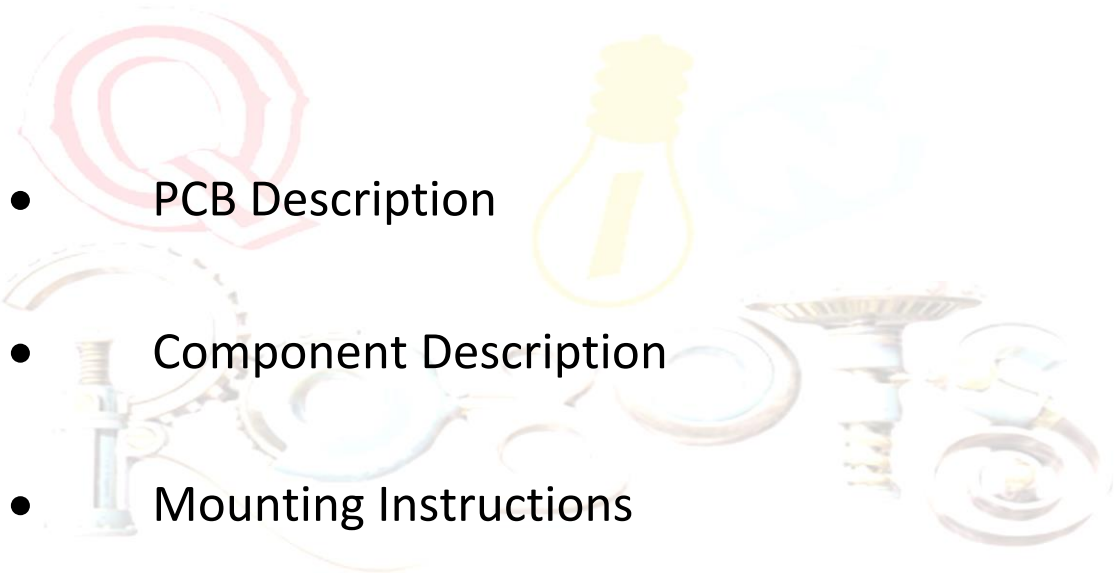
- **Use fewer & simpler parts.** Don't use unnecessary or over complicated features, or designs you do not have the tools to make or are really hard to make.
- **Use off-the-shelf parts.** When you purchase a part, it costs money. However there is a good chance that the off-the-shelf part is better than anything you can design and build yourself.
- **Do not use more than 2 or 3 different screw types.** If you can make your entire robot out of only the very common **4-40** screws, you are on the right track.
- Most importantly, remember to **KISS**. Keep it simple, silly!! 😊

+

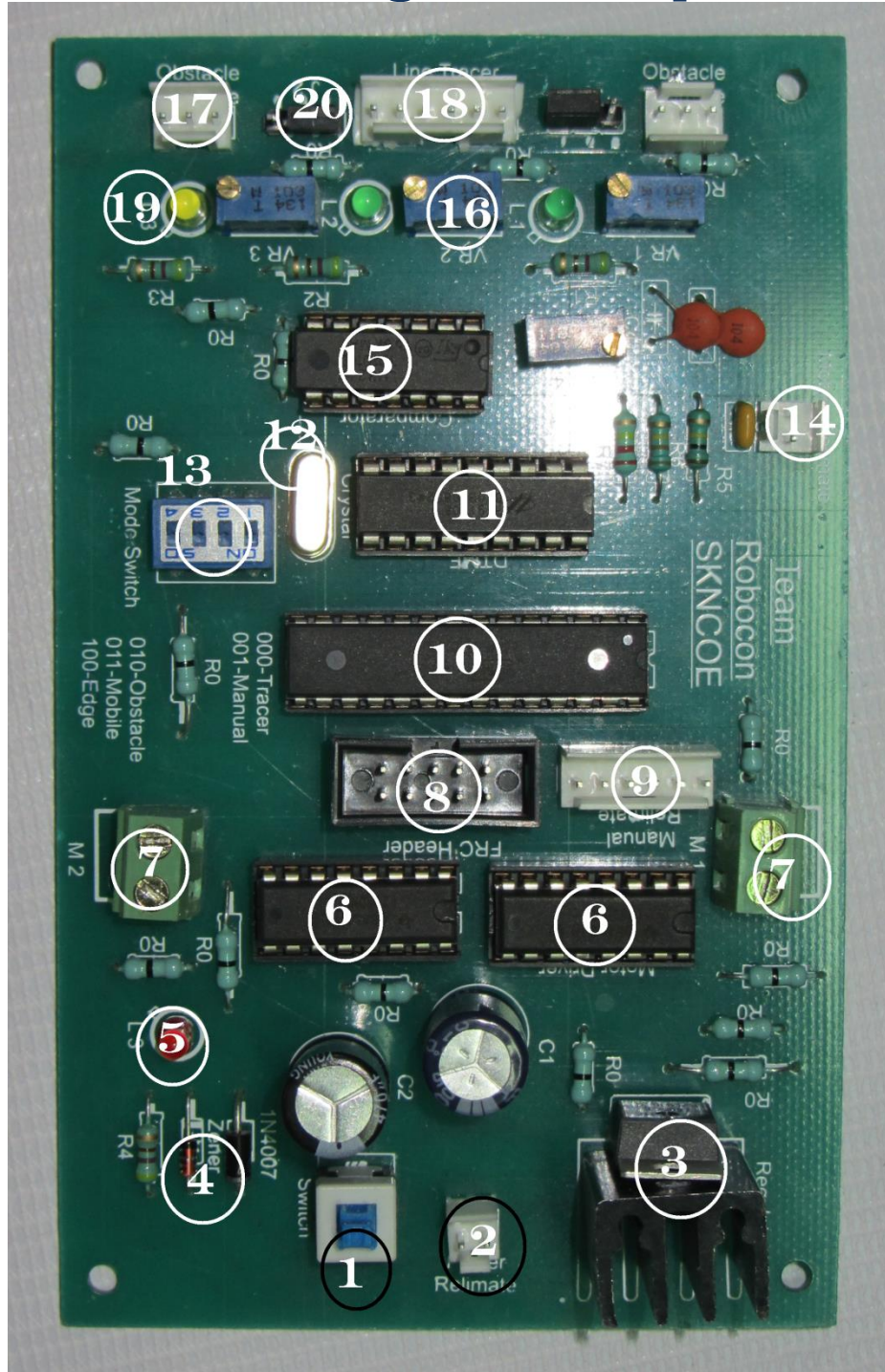
ELECTRONICS SECTION

CONTENTS



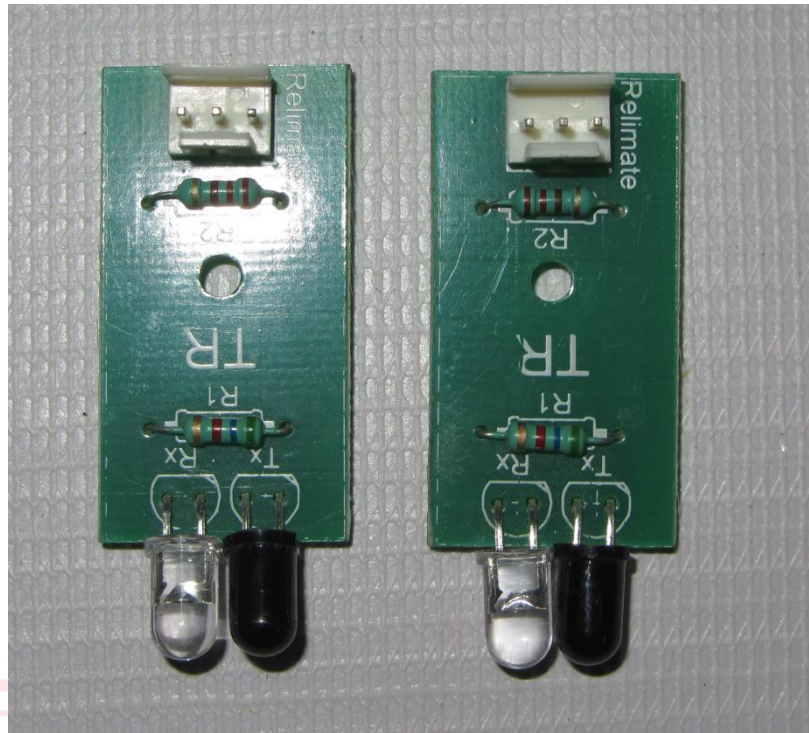
- 
- PCB Description
 - Component Description
 - Mounting Instructions
 - Do's And Don'ts

PCB : Locating the Components

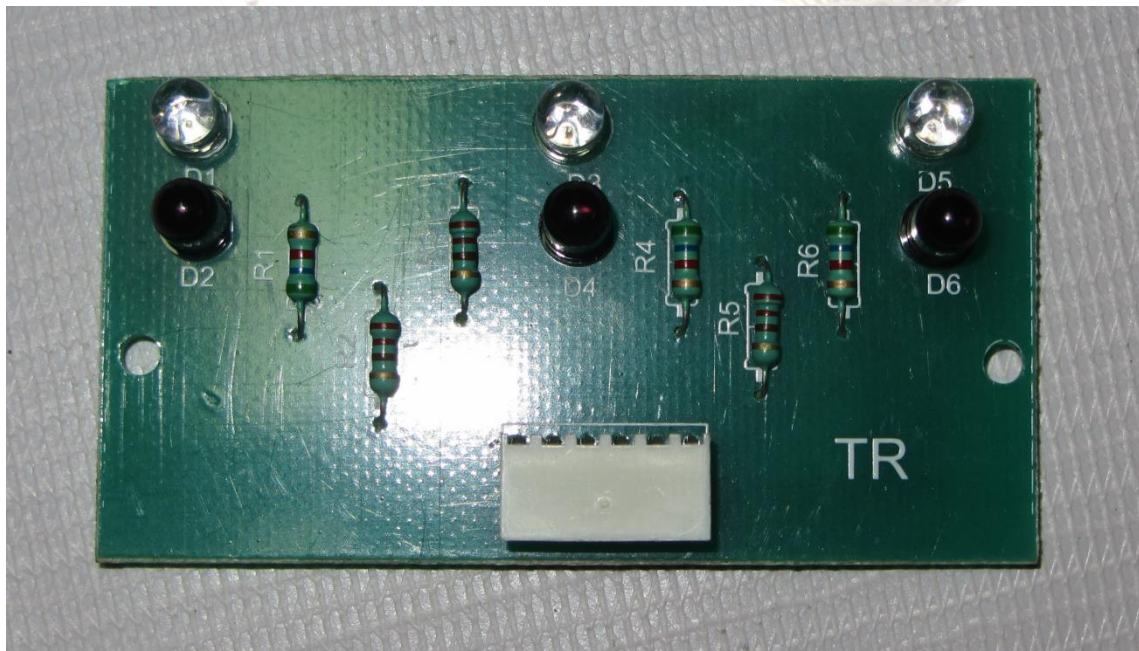


1. On/Off Switch
2. Power Jack
3. Voltage Regulator
4. Protection Diode
5. Power LED
6. Motor Drivers
7. Motor Connectors
8. FRC male Header for ISP
9. Relimate Connector for Manual control
10. Atmega8 Microcontroller
11. HT7290B DTMF Decoder
12. Crystal Oscillator
13. DIP switch
14. Mobile input
15. Comparator IC LM324
16. Potentiometer
17. (A/B) Single sensor inputs
18. (A/B) 3 sensor input
19. Reference Sensor output LED
20. Jumper

Note: While mounting the PCB onboard, try making it firm. Loose mountings may lead the falling of PCB from the bot which may lead to malfunction of the robot.



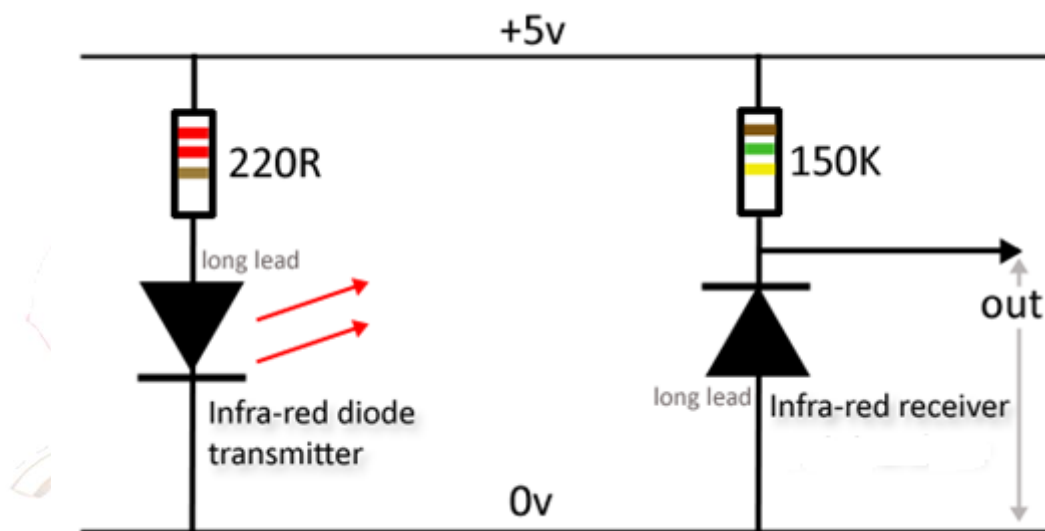
2 obstacle detector sensor module



Line tracer/Edge Detector Sensor Module

How it works: The Sensor Module

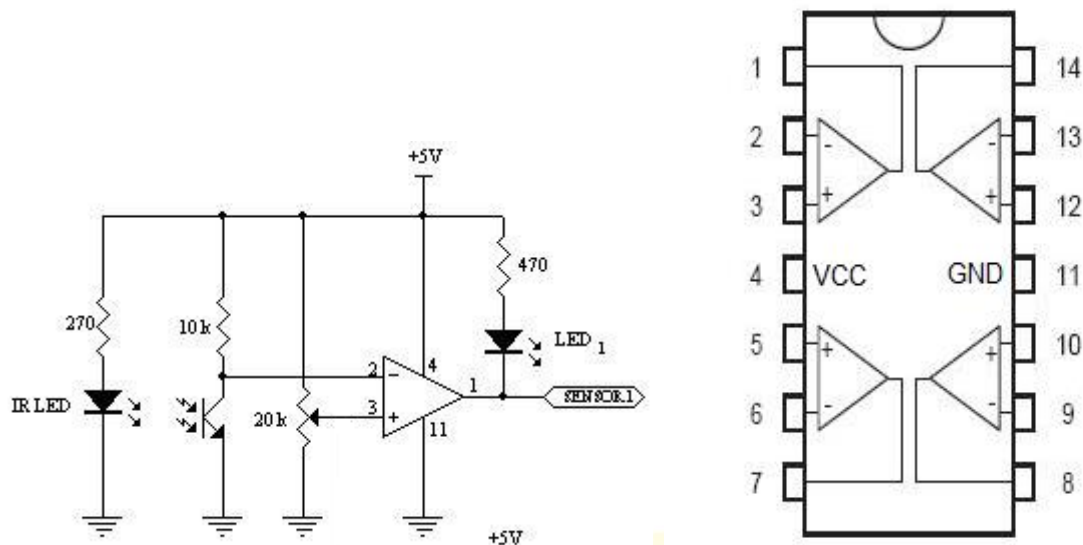
We are providing a sensor module which has 3 pairs of IR Transmitter-Receiver and 2 modules with one pair each. These sensors can be utilized and operated to receive a non electrical data from the environment and to convert it into an electrical signal. This signal can thus further processed by the brain of our robot. So this is how it works:



Above diagram shows the schematic of one part of sensor module. Here the IR LED is connected forward biased and the IR Receiver is connected as reverse biased. As per the intensity of light falling on it, it gives changes in the output.

When certain changes in the position of a robot and thus there are the changes in the environment around our robot, the changed conditions gives changes in the output. Like during obstacle detection mode, a considerable change in the output occurs when there is an obstacle in front of it, than when it is free from obstacles. Similarly for line tracing mode, a light reflecting back from the surfaces of different colours, the sensor gives different outputs. All these properties of sensors are thus used and given to the comparator.

How it Works: The Comparator



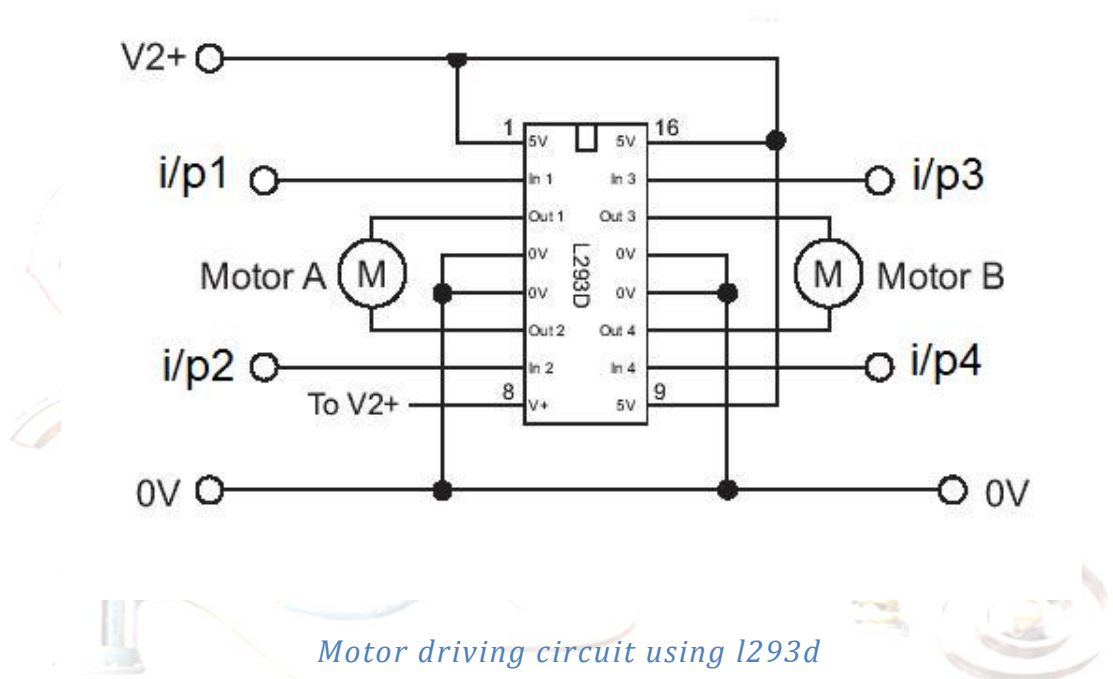
Sensors cannot itself provide an output, which can solely be used as a reference to the controlling.

Also every IR Receiver provides a different output even for the same conditions and is Analog in nature. Thus a fundamental signal conditioner is necessary to turn it into identifiable. This job is done using a comparator IC LM324. IC LM324 has 4 Comparators in it, so it can compare 4 sensor outputs at a time.

As shown above, one input of the comparator has given the output of a sensor and another one is given to the variable pot connected to the supply voltage, which will provide the reference comparator input. So the output of the sensor is compared with the reference voltage and the respective HIGH or LOW output is given out.

How it Works: The Motor Driver

While driving the motor, motors draw large currents from the source, which is in amperes. Thus connecting these loads directly to the control circuitry is totally impractical. Thus to control the motors from the robot brain, driving loads are needed to be isolated from it, also the high current to drive the motors has to be provided from some other source. These needs can be fulfilled by a DRIVER CIRCUITRY.



In our robot, we will be using l293d ICs for driving the motors. The schematic diagram of a common motor driver circuit is like above. Here i/p1 and 2 pins are for motor 1 and for another motor it is input 3 and 4. And each motor drives as follows

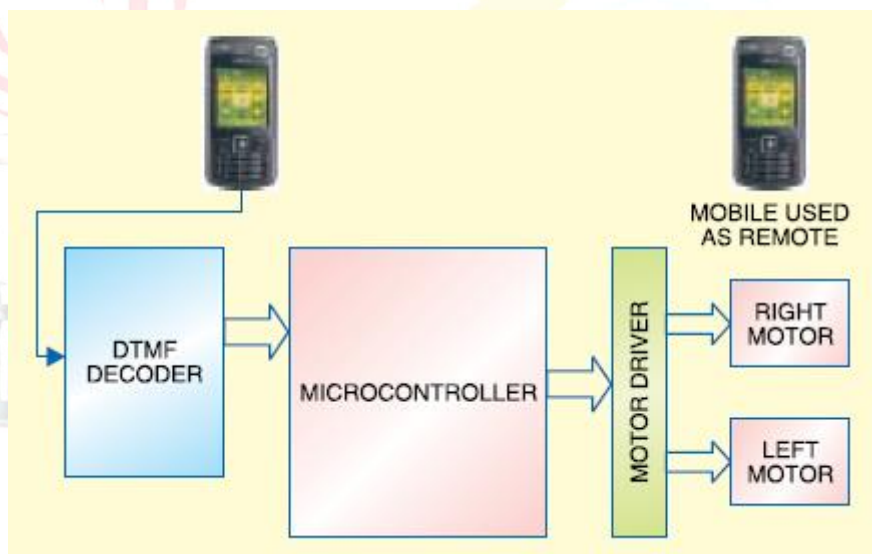
- When both i/p's are high – Motor Halts
- When both i/p's are Low – Motor Halts
- When i/p 1 is high and i/p 2 is low – Motor moves in forward direction
- When i/p 1 is Low and i/p 2 is High – Motor moves in Backward direction

In our robot, we are using one Motor driver IC for each motor by shorting out the terminals of both terminals of IC's. this helps to provide high load current to the motors without the IC getting damaged

How it Works: The Mobile Control

To control the robot using a Cell phone, we are using the DTMF (Dual tone Multiple Frequencies) tones to do so. These are

the irritating tones we hear in a call when some keys are pressed mistakenly. Each keys on dial pad produces different frequencies and are distinct from each of them



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 control corresponding to the button pressed is heard at the other end of the call. This tone is called dual tone multi frequency tone (DTMF) IC HT9170B receives this DTMF tone with the help of phone stacked in the robot

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

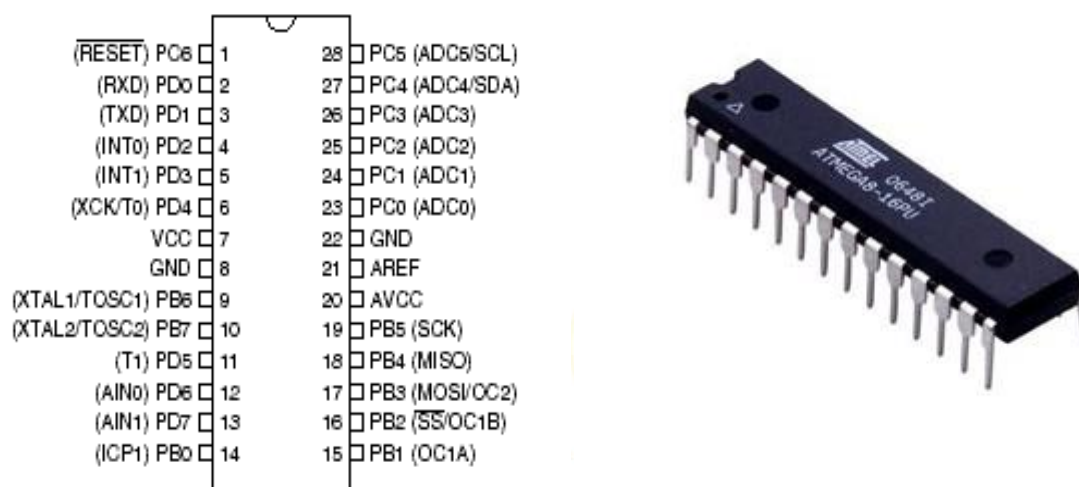
The mobile that makes a call to the mobile phone stacked in the robot acts as a remote. So this simple robotic project does not require the construction of receiver and transmitter units.

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 5will send a tone made by adding 1336hz and 770hz to the other end of the mobile

How it Works: The Microcontroller

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.



Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes.

The Microcontroller we will be using is atmega8. It is a 28 pin Microcontroller in DIP package it has following key parameters

Key Parameters

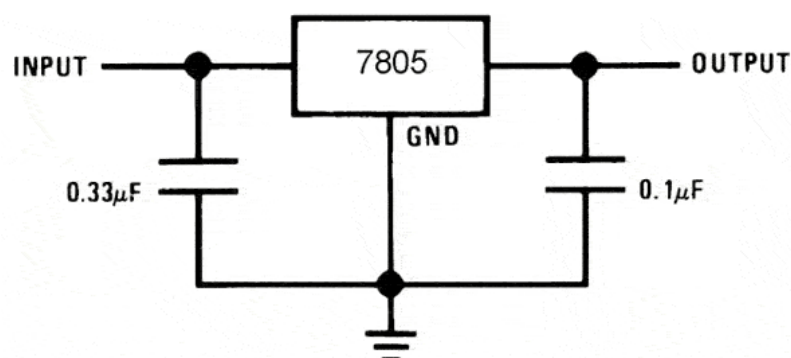
Parameter	Value
Flash (Kbytes):	8 Kbytes
Pin Count:	32
Max. Operating Frequency:	16 MHz
CPU:	8-bit AVR
Max I/O Pins:	23
Ext Interrupts:	2

The atmega8 IC has given the inputs from all the sensor modules, also from the cell phone and the manual joystick and the mode switch input is also given to it. Motor drivers are also connected to the same IC. So as per the mode selected and as per the program burned inside the flash memory of atmega8 the operation of the robot is done.

Miscellaneous Components:

- **Voltage Regulator**

A fixed voltage regulator , IC7805 will regulate any unstable voltage in the range of 7 Volts to 30 Volts to a constant regulated voltage of 5V. It provides both Line and load regulation to the circuit. Also it has over voltage protection and thermal shutdown in it.



- **Motor Connector**

These High Current carrying screw connectors are used to connect the motors to the circuit. It makes it convenient to attach the motor to the circuit



- **Crystal Oscillator**

Decoder HT9170B requires an external clock signal of 3.57Mhz. This clock signal is provided by a crystal clock oscillator.



- **FRC Header (Male)**

A FRC male connector is present in the side of Microcontroller. It is used to connect with the In system programming programmer to control the microcontroller.



- **Battery**



We will be using a Lead acid Battery of 12V , 1.2AH supply rating, a lead acid battery requires less time to charge and also lasts long. Also it has longer lifetime than other battery types. It is cheaper than Li-Ion battery and also easily available.

- **DC Motor**

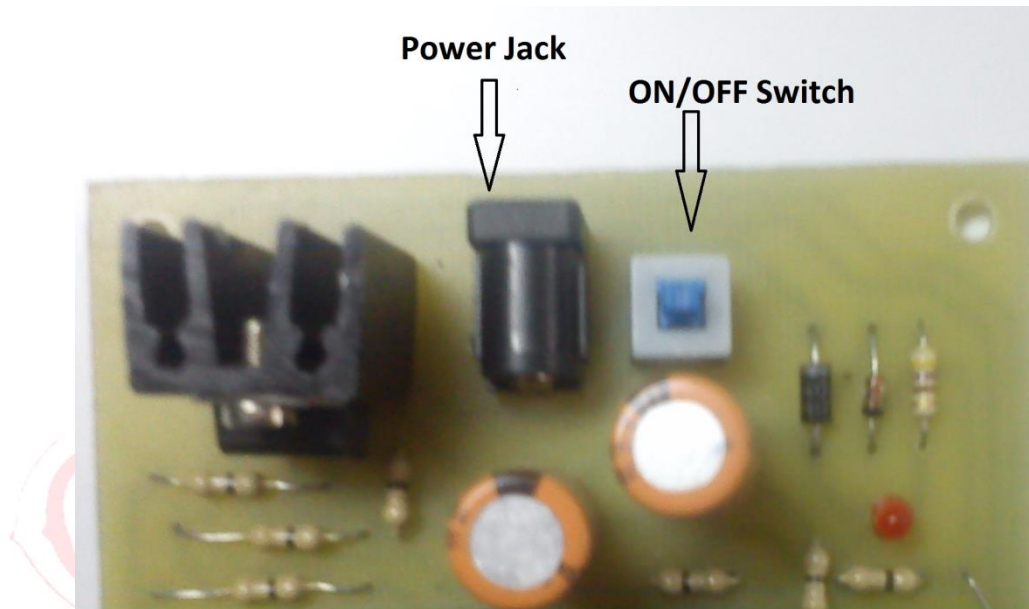


The working principle behind any DC motor is the attraction and repulsion of magnets. The simplest motors use electromagnets on a shaft, with permanent magnets in the case of the motor that attract and repel the electromagnets. The reason for using electromagnets is so that it is possible to flip their magnetic field (their north and south poles).

So the electromagnet is attracted to one of the permanent magnets. As soon as it reaches the permanent magnet, it's north and south poles flip so that it is repelled from that magnet and attracted to the other permanent magnet.

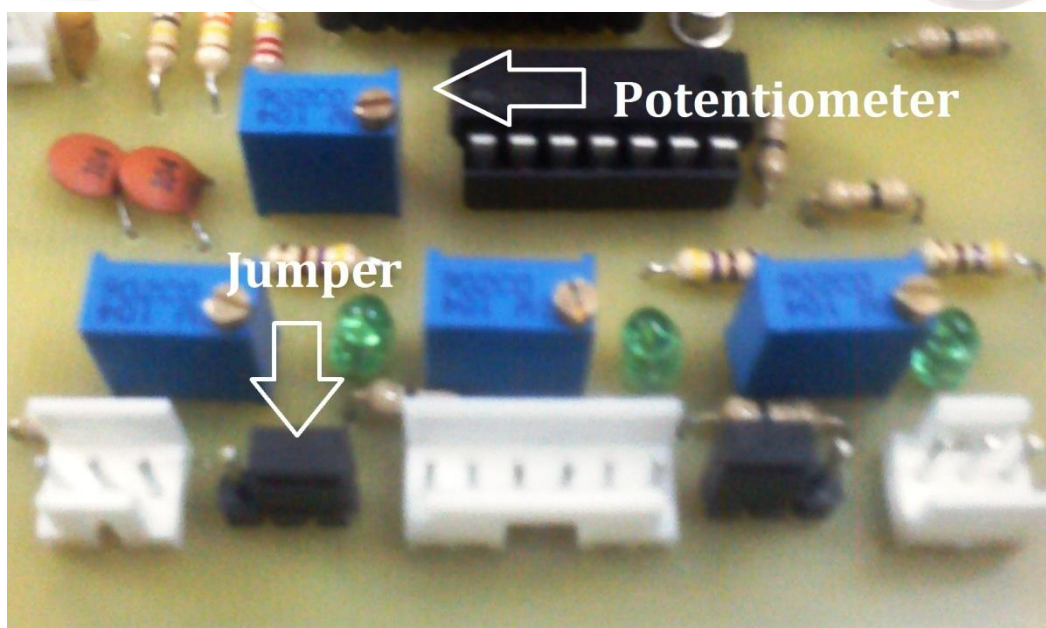
Mounting Instructions:

- **Connecting the Battery**



The provided Battery is already connected with a power plug to avoid the problems. To start the bot you need to plug the power plug in the power socket. But before plugging it you should consider checking the on/off switch if it is off or not as plugging the battery in the on state may give rise to the surge current damaging the circuitry

- **Calibrating the Sensors**



- **Calibrating the Sensors**

The sensor modules provided are not most of the time pre calibrated too the comparator and to make the robot operate work properly their good calibration is necessary and it can be done using the potentiometers shown in above diagrams. Adjusting the reference voltage to the comparator is done by varying the potentiometers.

- **Jumper settings**

Jumpers are used to provide the connectivity between two points. 3 pin buck strip above is used to do the switching between obstacle avoider mode and the line tracer/Edge detector mode. Jumpers towards the inner side sets the bot in Line tracing mode and on outer sides it is adjusted to obstacle avoider mode

- **Connecting the Motors**



Green connectors as shown above are provided one side of each motor driver. First you have to make some space in them by unscrewing it and then by putting the wire coming from the motors, the screws have to be tightened. While connecting, consider checking the polarity of the motor as it may lead to the motor moving in reverse direction. Also check if the screws are tightened or not as during the movement of the robot it may lead to disconnection of motors making the robot stop.

- **Programming the Robot.**



The robot provided is pre-programmed with the default program. But if you consider writing your own program to make some changes in its functionality or simply for testing we have provided a male FRC header for an ISP programmer. The male header is provided on the side of the Microcontroller to program the robot you have to just get your compiled hex file, put it into your burner software, connect the programmer to your Computer and another end to the Robot and hit the burner On.

- **Modes of working (Switching the modes)**

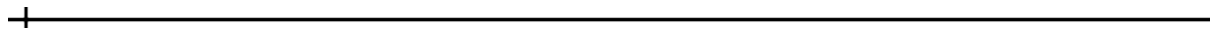
When the switches are in on state consider them in state 1 and the off state as state 0. And we have 3 switches, adjusting which will give us 8 different modes, 5 of them can be used for switching between 5 different modes of our robot.

Refer the chart below to adjust the desired mode of operation

State of Switch	Mode of Operation
000	Manual
001	Line Tracer
010	Obstacle Avoider
011	Cell phone Controlled
100	Edge Detector

Computer SECTION

CONTENTS



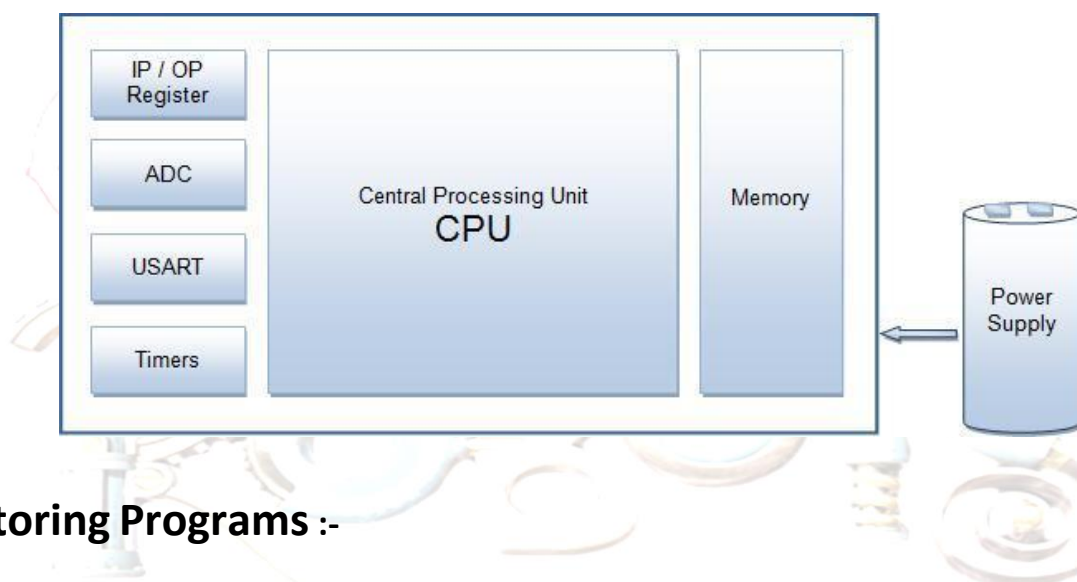
- Programming a Microcontroller
- Storing Programs
- Working with AVR Studio
- Code Explanation

Programming a Microcontroller

Nearly all factory-fresh microcontrollers arrive without any program in them whatsoever. The configurable pins are usually designed to start as inputs at power-up. Without any output pins, the chip does nothing but listen to the inputs and constantly reset itself as it encounters improper (random garbage) instructions.

It's up to you, the developer, to give each microcontroller its purpose.

Block diagram of a microcontroller :-



Storing Programs :-

Most modern microcontrollers contain some form of internal non-volatile memory for storing programs. The program remains inside of the chip even when power is turned off or the chip is removed from the circuit. So, once you've programmed a microcontroller, you can move it from circuit to circuit, power it up, and have the program start running immediately. This really provides the sense of having created a custom chip.

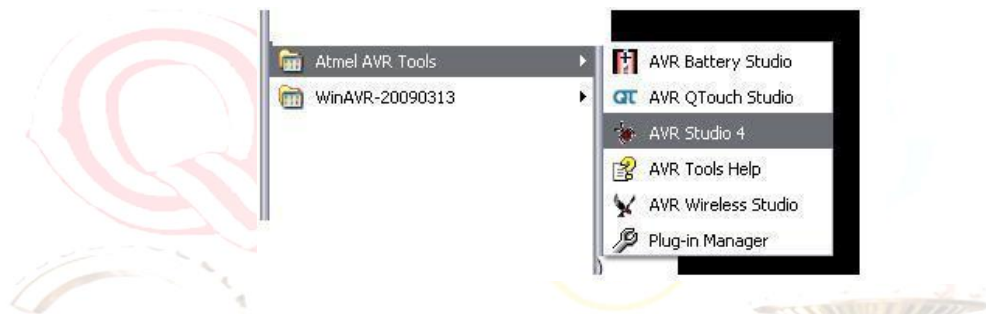
The two most popular reprogrammable non-volatile memory technologies in use today in microcontrollers are electrically erasable programmable read-only memory (EEPROM) and Flash. These allow the chips to be programmed and reprogrammed over and over again, between ten thousand and a million times or more. The program or other memory contents can be read an infinite number of times and remains intact for decades.

We are going to use Atmel's **Atmega8** microcontroller which has **8KBytes In-System Programmable Flash**.

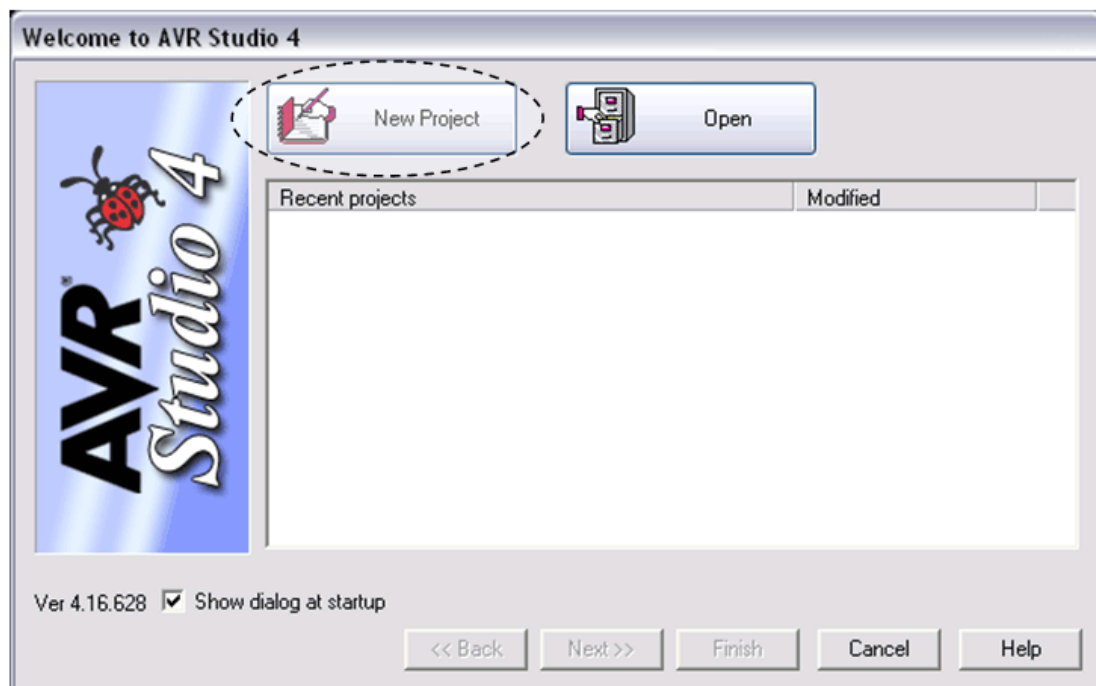
Working with AVR Studio

AVR studio is an Integrated Development Environment (IDE) by ATMEL for developing applications based on 8-bit AVR microcontroller. Prior to installation of AVR Studio you have to install the compiler WinAVR. This will allow AVR Studio to detect the compiler.

Step 1:

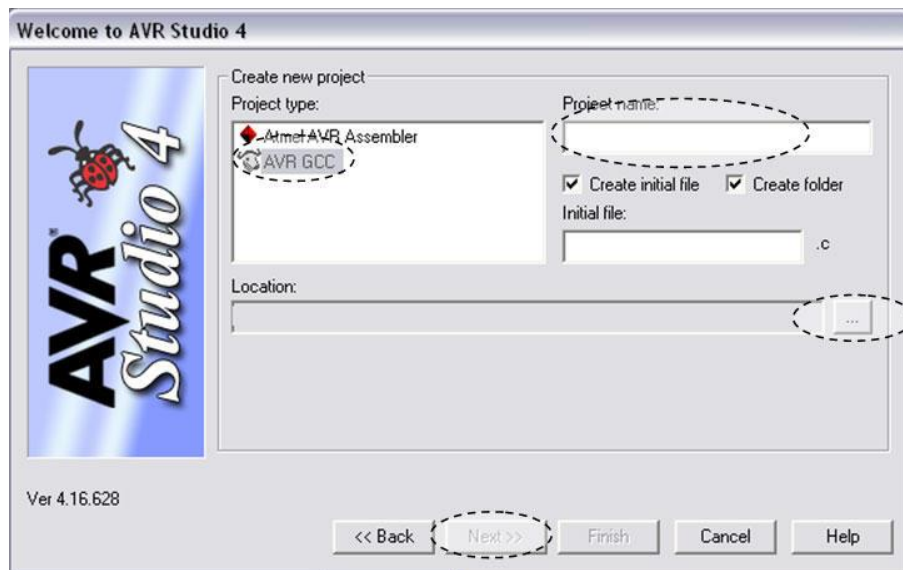


Step 2:



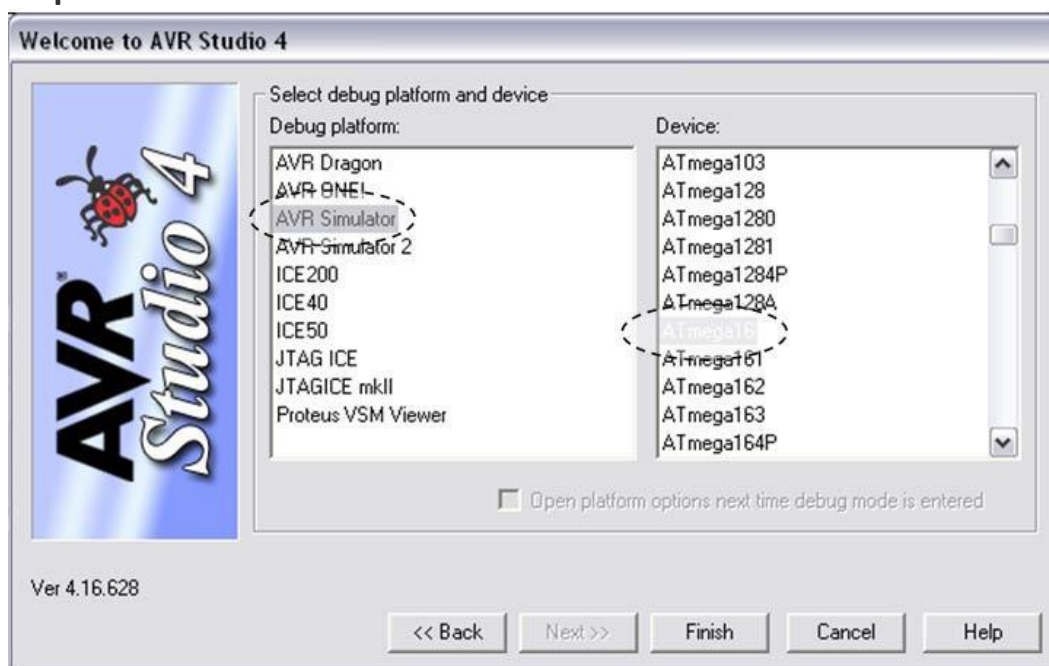
Click on new project.

Step 3:



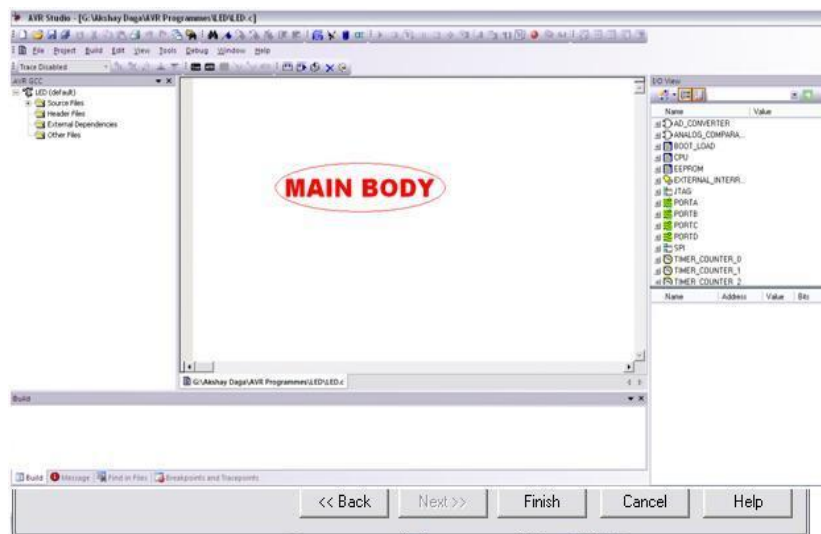
Click on AVR GCC. Write the project name. Select your project location. Click on Next>>

Step 4:



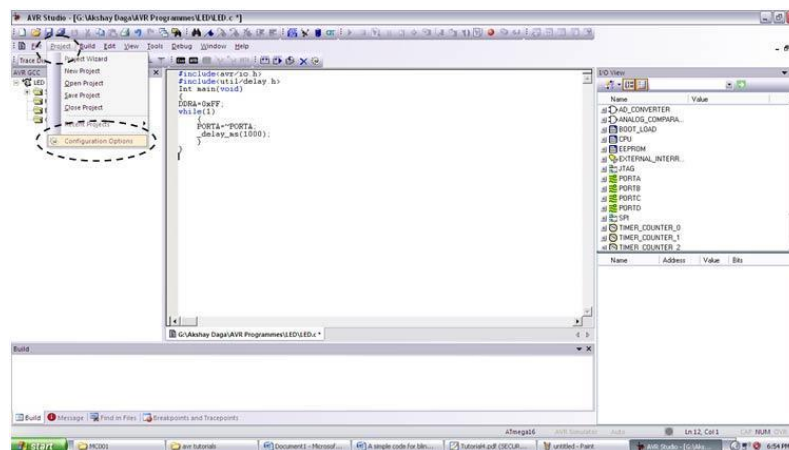
Click on AVR Simulator in left block and then select your controller (e.g.: [ATmega8](#)). Click on finish button.

Step5:



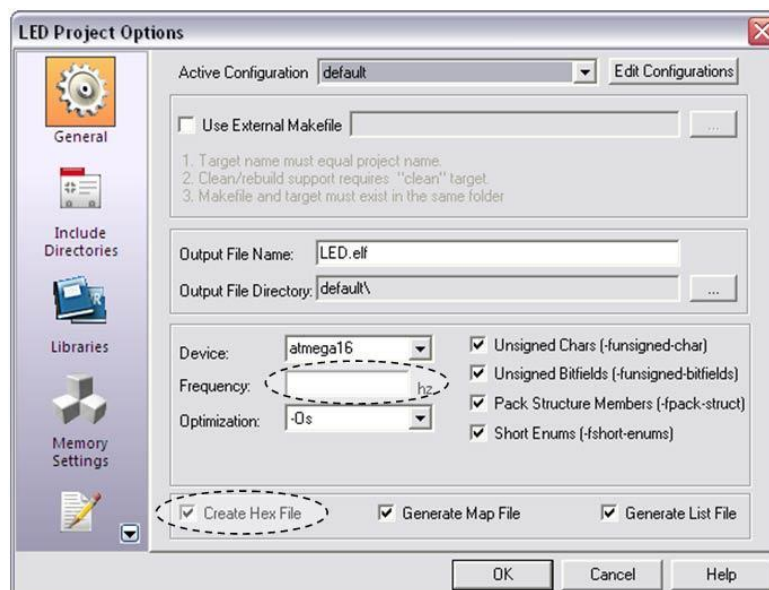
Write the code in main body area. Save the project file.

Step6:



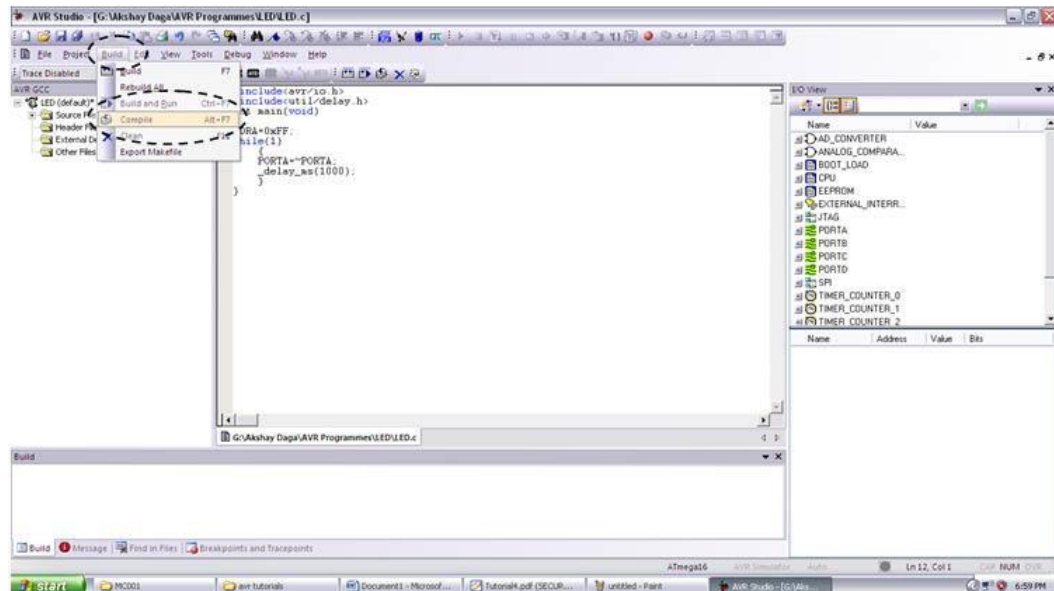
Go to PROJECT -> Configuration Options

Step 7:

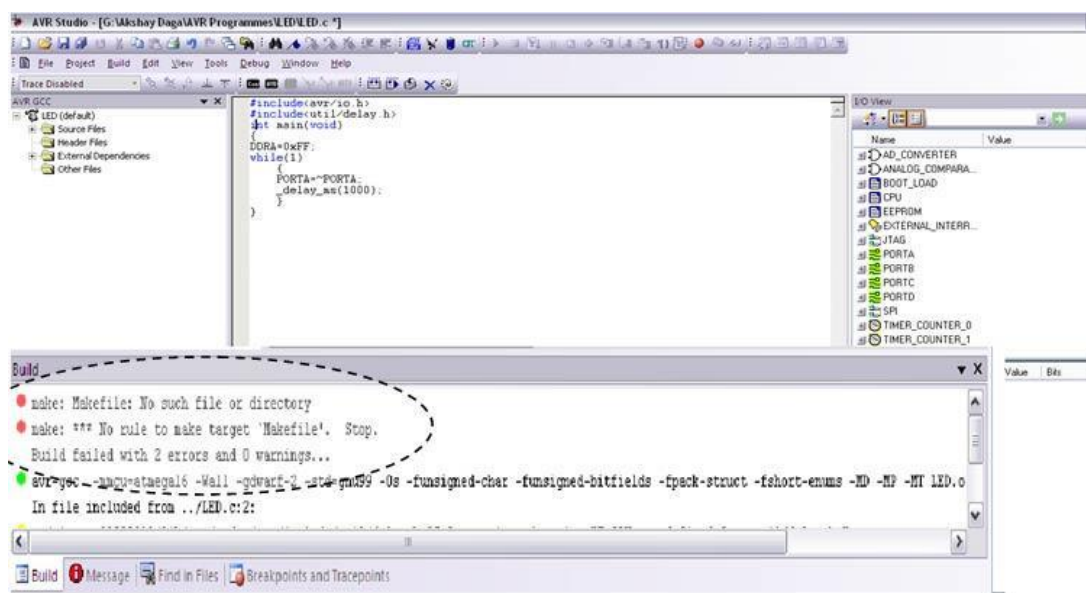


Write the crystal frequency if you are using external crystal.
Check the checkbox corresponding to Create Hex File and then click on OK.
Save the project again.

Step 8:

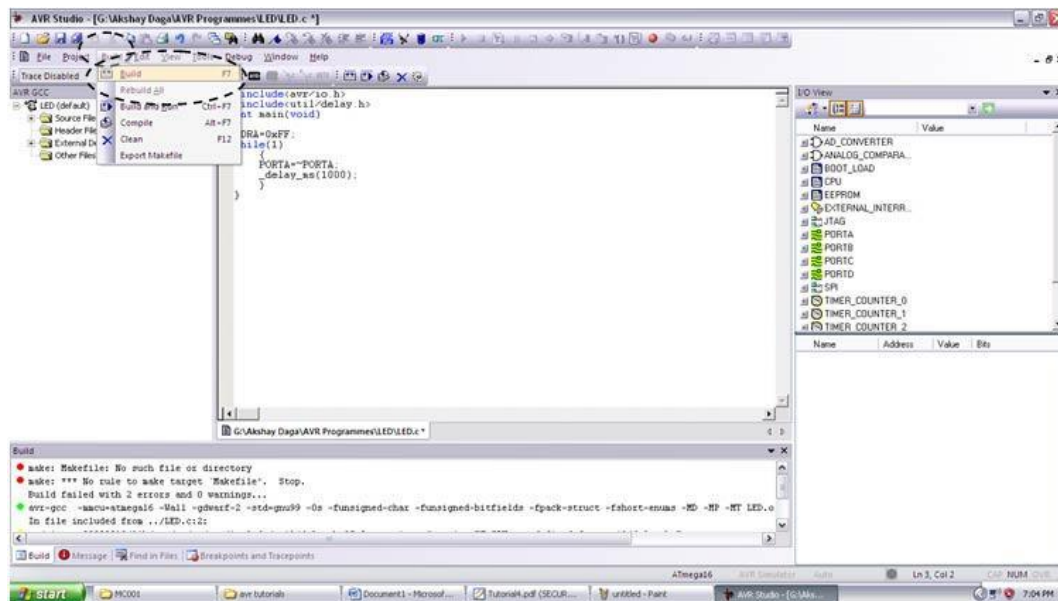


Go to BUILD -> Compile. This will compile your code and generate error if any.



For the first time it will generate two errors, ignore them.

Step 9:



Again go to BUILD and click on Build. This will generate hex file of the code. Use that Hex file to burn your microcontroller.

Where you will find Hex file?

Just go to the location which you selected at the starting. Open that folder you will find one more folder named Default. This is the default location of where the hex file is generated.

While working in real time if you want to change the code, make changes and build the file again. This will automatically update the previous hex file.

Code Functionalities:

- **Motion Modules**

```
void fwd(int speed)
{
    stage=0;
    SETBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    SETBIT(PORTB,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    delay_ns(speed);
    CLEARBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    CLEARBIT(PORTB,BIT(1));

    delay_ns(200-speed);
}
```

```
void back(int speed)
{
    stage=0;
    SETBIT(PORTC,BIT(1));
    CLEARBIT(PORTC,BIT(0));
    SETBIT(PORTB,BIT(2));
    CLEARBIT(PORTB,BIT(1));
    delay_ns(speed);
    CLEARBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    CLEARBIT(PORTB,BIT(1));
    delay_ns(200-speed);
}
```



```

void left(int speed)
{
    stage=1;
    SETBIT(PORTB,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    CLEARBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    delay_ns(speed);
    CLEARBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    CLEARBIT(PORTB,BIT(1));
    delay_ns(200-speed);
}

```

```

void right(int speed)
{
    stage=2;
    SETBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    CLEARBIT(PORTB,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    delay_ns(speed);
    CLEARBIT(PORTC,BIT(0));
    CLEARBIT(PORTC,BIT(1));
    CLEARBIT(PORTB,BIT(2));
    CLEARBIT(PORTB,BIT(1));
    delay_ns(200-speed);
}

```

PortC,Bit0 and Bit1 and PortB Bit1 and Bit2, are inputs to motor drivers which decides the direction of the motion of Motors. It provides the PWM to the motors. Value of speed decides the duty cycle of the PWM generated. As the modules are in infinite loops, it will produce continuous PWM to the motors. All above four Modules, using combinations of Bit Values will generate 4 different directions of motors , that are Forward, Reverse, Left and

- **Obstacle Avider Module**

```

void obstacle()
{
    long unsigned int i,j;
    fwd(80);
    if(CHECKBIT(PIND,BIT(6)) ||
    CHECKBIT(PIND,BIT(5)))
    {
        stop();
        for(i=10000;i>0;i--)
        {
            delay_ns(25);
        }

        for(i=1000;i>0;i--)
        {
            back(120);
            delay_ns(7);
        }
    }
}

```

```

        for(i=1000;i>0;i--)
        {
            right(100);
            delay_ns(5);
        }
    }
}

```

```

/*
For Obstacle Avider, Sensor output is
given to PortD bit5 and Bit6 and if at
any of these pins a high input is
obtained the previously set forward
motion will be stopped and after
certain time delay robot will move
backwards and then after moving right,
if the obstacle is out of its way the
robot will be set again in the forward
motion
*/

```


- **Line Tracer Module**

```
void trace_line(void)
{
    if(!(CHECKBIT(PIND,BIT(6))))
        //right
        {
            if(!(CHECKBIT(PIND,BIT(7))))
                //center
                {
                    if(!(CHECKBIT(PIND,BIT(5))))
                        //left
                        {
                            {
                                fwd(80);
                            }
                            else
                            {
                                right(100);
                            }
                        }
                    else
                    {
                        if(!(CHECKBIT(PIND,BIT(5))))
                        {
                            left(120);
                        }
                        else
                        {
                            right(120);
                        }
                    }
                }
            else
            {
                if(!(CHECKBIT(PIND,BIT(7))))
                {
                    if(!(CHECKBIT(PIND,BIT(5))))
                    {
                        left(100);
                    }
                    else
                    {
                        fwd(150);
                    }
                }
            }
        }
}
```

```
else
{
    if(!(CHECKBIT(PIND,BIT(5))))
    {
        left(120);
    }
    else
    {
        if(stage==1)
        {
            right(100);
        }
        else
        {
            left(100);
        }
    }
}
```

/* Conditional If Else Statements are used to check the position of the robot. Input is obtained at Pin5,6,and 7 of portD. These 3 pins will decide the position of the robot. It thus with respect to the position with reference to the line is deciding the direction of motors using the Above speed modules */

- **DTMF Mobile Control**

```
void mobile()
{
    dtmf=PIND;
    dtmf=dtmf & 0x1b;
    if(dtmf==0x02)
    {
        fwd(150);
    }
    if(dtmf==0x10)
    {
        back(150);
    }
    if(dtmf==0x08)
    {
        left(120);
    }
    if(dtmf==0x0A)
    {
        right(120);
    }
    if(dtmf==0x09)
    {
        stop();
    }
}
```

```
/*
```

Decoded output from the decoder IC arrives at PortD of microcontroller.

The dual tone frequencies are thus converted and got into the digital output at these pins.

For each key keystroke of 2,4,6,8 and 5 a different digital output is obtained.

These inputs are converted

And using 'if else' conditional statement the respected motion of motor is decided and produced.

```
*/
```

- **Manual Bot Control Module**

For manual robot mode an input from an external Joystick is given to the microcontroller and as per the input from it, the motion of robot is decided. Joystick has 2 DPDT switches thus it has 4 possible combinations which are both forward , both backward and 2 alternate combinations and 4 single of each. These in combine as generates 8 different inputs and hence the microcontroller processes into 8 different possibilities of motion using the motion modules described above.

```
void trace_line(void)
```

```
{    if(!(CHECKBIT(PIND,BIT(6))))
    //right
    {
        if(!(CHECKBIT(PIND,BIT(7))))
        //center
        {
            if(!(CHECKBIT(PIND,BIT(5))))
            //left
            {
                fwd(80);
            }
            else
            {
                right(100);
            }
        }
        else
        {
            if(!(CHECKBIT(PIND,BIT(5))))
            {
                left(120);
            }
            else
            {
                right(120);
            }
        }
    }
}
```

```
else
```

```
{
    if(!(CHECKBIT(PIND,BIT(7))))
    {
        if(!(CHECKBIT(PIND,BIT(5))))
        {
            left(100);
        }
        else
        {
            fwd(150);
        }
    }
    else
    {
        if(!(CHECKBIT(PIND,BIT(5))))
        {
            left(120);
        }
        else
        {
            if(stage==1)
            {
                right(100);
            }
            else
            {
                left(100);
            }
        }
    }
}
}
```

Do's:-

- Refer the datasheets for full details about components
- Keep the supply voltage between 8V-18V
- Wrap insulation tape on open wire.
- Hold the PCB from back side. Holding it from front side may break the sensor PCB
- Handle IC's Carefully. The pins may break easily
- Use a 14v ADAPTER to charge the battery

Don'ts:-n

- Short the two terminals of battery, it may damage the battery as well as motherboard
- Leave battery in discharged condition. It reduces the battery life
- Increase the supply more than 18V
- Keep the motherboard in moist , wet or electrostatic environment. It may damage the IC.
- Obstruct the motion of motor while in motion. Motor windings may get burnt
- Touch the heat sink directly, it may burn your skin
- Operate the mother board without the heat sink on 7805
- Overcharge the battery
- Touch the IR transmitter in operation.

Checklist for items in Kit :-

- ✓ 2 Motors
- ✓ 2 Wheels
- ✓ 2 Clamps
- ✓ 1 Chassis
- ✓ 1 Sensor board
- ✓ 2 Obstacle Detecting sensor Modules
- ✓ 1 Motherboard including all IC's
- ✓ 12V battery
- ✓ On/off Switch
- ✓ 2 DPDT switches
- ✓ 1 Remote Box
- ✓ 1.5M rainbow Wire
- ✓ 1 Caster Wheel
- ✓ 6 Pin relimate connector wire
- ✓ 2 3-pin relimate connector wire
- ✓ Six 2-Cm Bolts
- ✓ Five 2-cm Bolts
- ✓ Three 4-CM bolts
- ✓ 35 M3 Nuts
- ✓ 3.5MM Jack with wire
- ✓ 1 double headed screw Drive
- ✓ 1 Insulation tape
- ✓ 1 Cutter
- ✓ And Lots of surprises in-between

Reference Links:

- [http:// www.societyofrobots.com](http://www.societyofrobots.com)
- <http://www.electronicsteacher.com/>
- [http:// extremeelectronics.co.in](http://extremeelectronics.co.in)
- <http://avrfreaks.com>
- <http://www.efymag.com>
- <http://www.robotics.org>
- <http://robots.net>
- <http://societyofrobots.com>
- <http://www.active-robots.com>
- <http://edaboard.com>