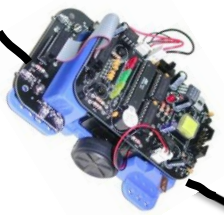
# SOCIETY OF ROBOTICS AND AUTOMATION

# WALL-E

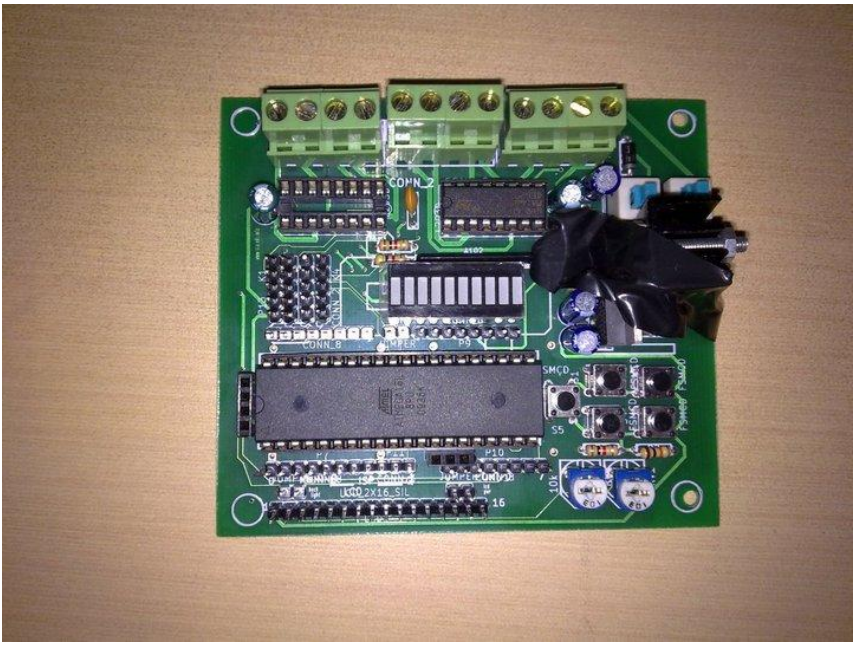
## *A Guide to understanding your line following robot*



MLR 11,
Mechanical Department,
Veermata Jijabai, Technological Institute,
Matunga,
Mumbai-400019.

# INDEX

# The Development board

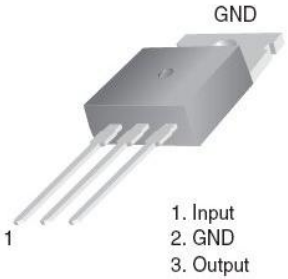



**The main blocks of the development board –**

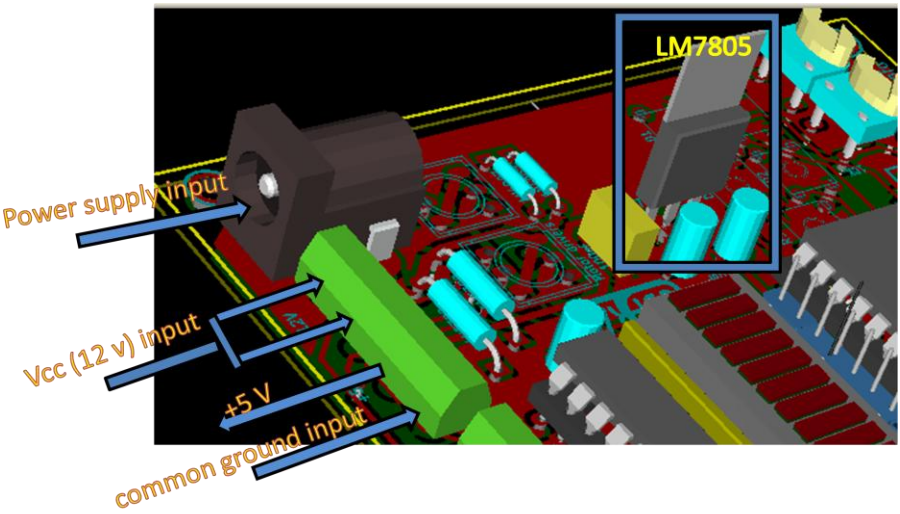
1. Power supply and voltage regulation
2. Microcontroller (ATmega 16/32)
3. Motor drivers
4. LCD display ports
5. ASK RF Tx and Rx ports
6. FT232

## Power supply and voltage regulation

- All digital ICs work at a voltage around 5v and may get damaged at higher voltages but motors require a high voltage of about 12 volts to function. To solve this discrepancy, a voltage regulator circuit is added on to the board. The voltage regulator takes an input of 12V and gives a 5V output and hence only one power supply is needed to power the entire robot.
- The voltage regulator on the board is LM7805.



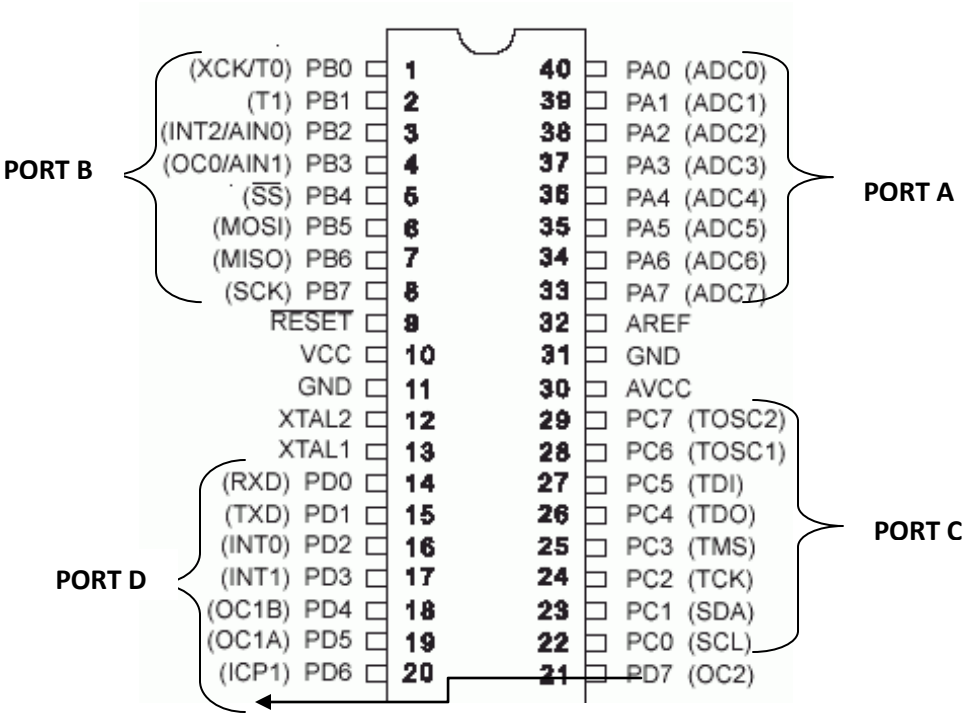

The board has provisions for taking 12 V input from a battery (through wire connections) as well as from a power supply (through a power jack).



## The Microcontroller
## (The brain of your bot)

A **microcontroller (MCU)** is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
The MCU provided on our board is an ATMEL ATmega 16/32.

The pin configuration is as given below –




**PIN DESCRIPTIONS AND THEIR USAGE ON BOARD:**

| | |
|---|---|
| **VCC** | Digital supply voltage. |
| **GND** | Ground |
| **RESET** | Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. |

**AVCC** AVCC pin is the supply voltage for Port A and the A/D converter. It should be externally connected to Vcc.

**AREF** AREF is the analog reference pin for the A/D converter.

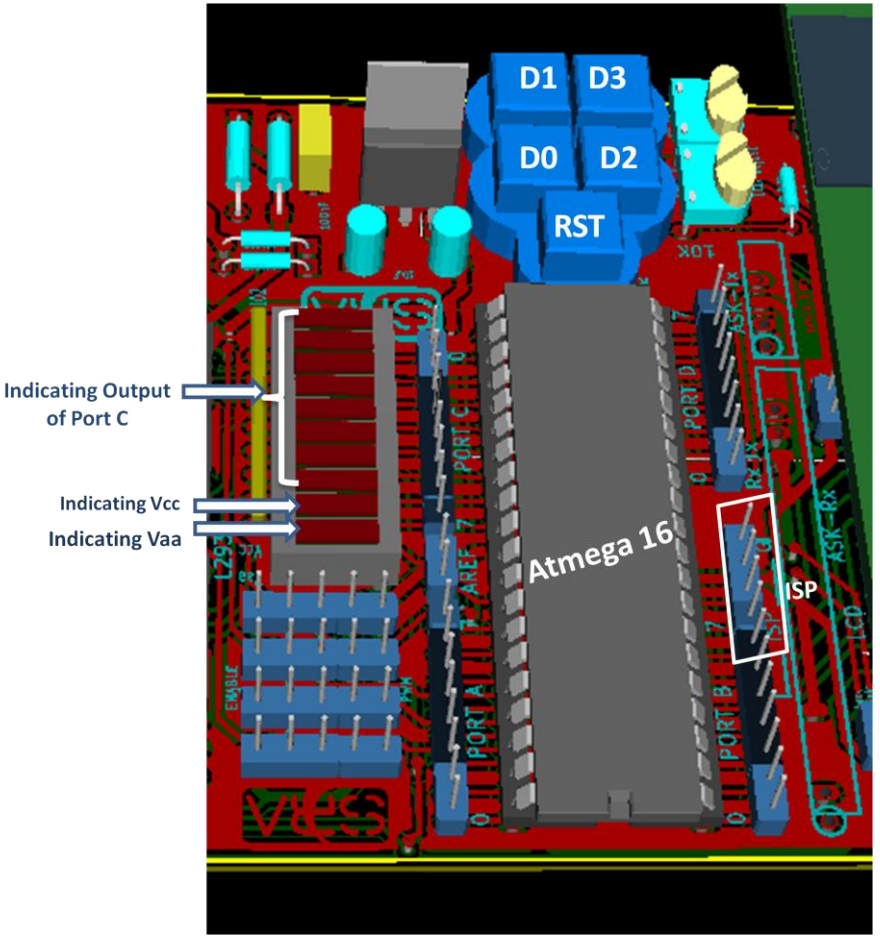
**PORT A (PA7..PA0)** PORT A serves as the analog inputs to the A/D Converter. It also serves as an 8-bit bi-directional I/O port. The sensors are connected to PORT A.

**PORT B (PB7..PB0)** PORT B is an 8-bit bi-directional I/O port. PB7(SCK), PB6(MISO), PB5(MOSI), PB4(SS) form the SPI(Serial Peripheral Interphase) pins that are used for in-serial programming of the MCU. PB3 (OC0) is an ouput pin for the PWM mode timer function and is connected to the enable of the motor driver.

**PORT C (PC7..PC0)** PORT C is an 8-bit bi-directional I/O port. The LED bar is connected to the PORT C pins. Also PORT C serves as an input to the motor drivers.

**PORT D (PD7..PD0)** PORT D is an 8-bit bi-directional I/O port.
PD0 (RXD) – Receive data (Data input for the USART)
PD1 (TXD) – Transmit data (Data input for the USART)
PD4 (OC1B), PD5(OC1A), PD7(OC2) - Ouput pin for the PWM mode timer function and is connected to the enable of the motor driver.
Also the pins D0, D1, D2, D3 are connected to the switches as given below.

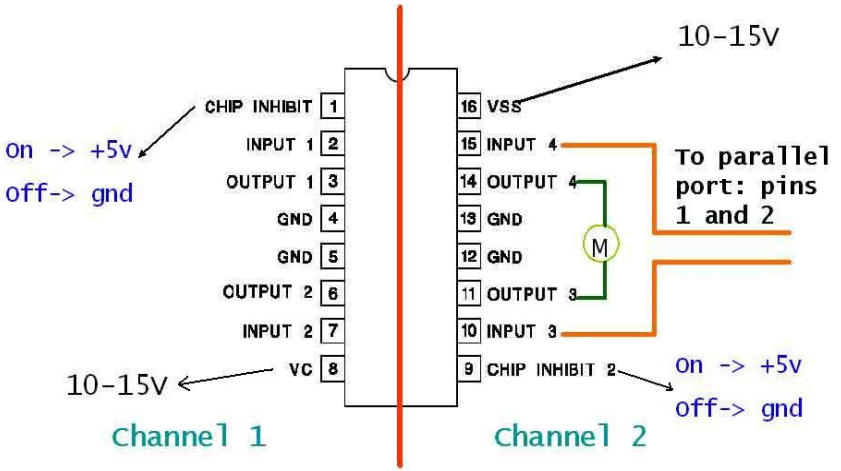
*(Please note that in addition to the above mentioned functions, the ports of MCU perform various other functions that is given in detail in the datasheet of the MCU)*

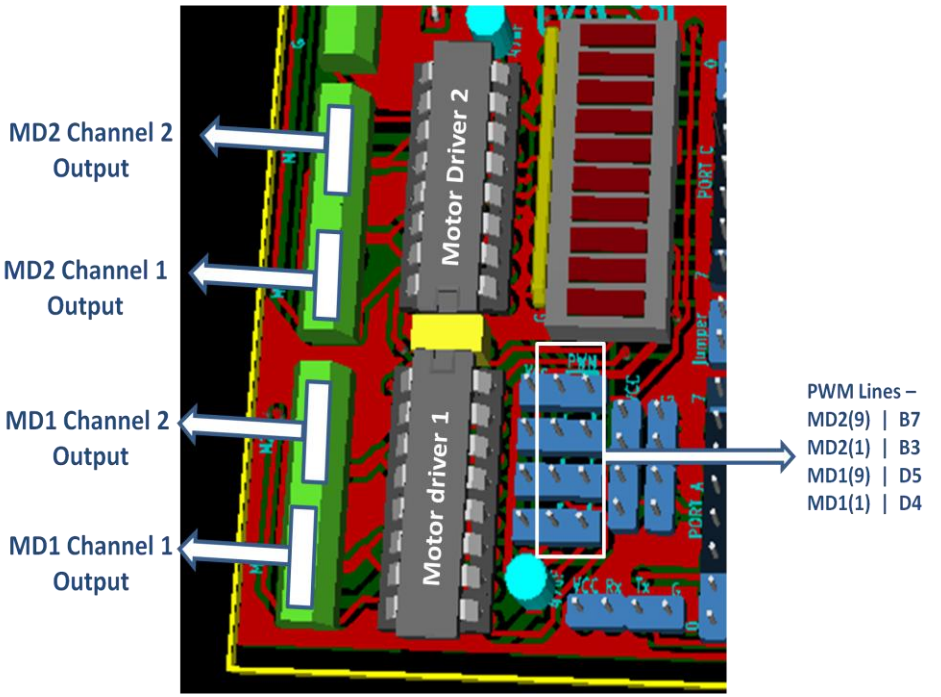

D1
D3
D0
D2
RST
Indicating Output of Port C
Indicating Vcc
Indicating Vaa
Atmega 16
ISP

## MOTOR DRIVERS

Motor drivers are essentially little current amplifiers; their function is to take a low-current control signal, and turn it into a proportionally higher-current signal that can drive a motor.

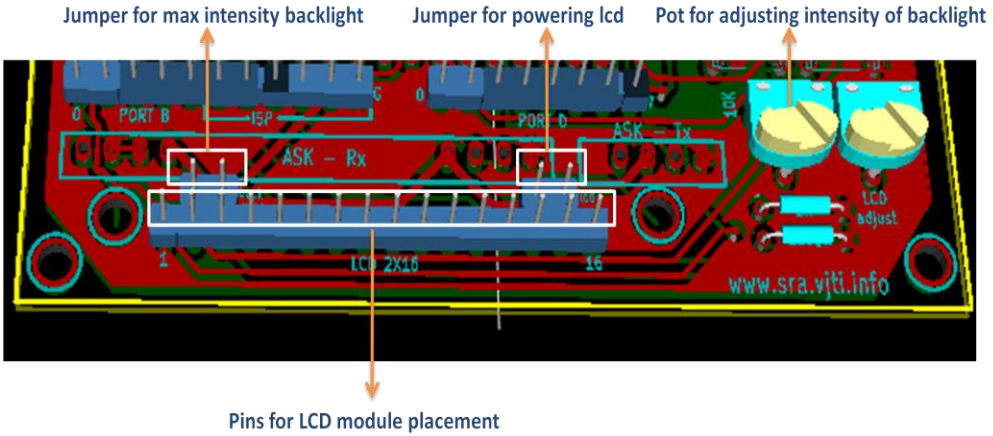
The motor driver provided on board is IC L293D.

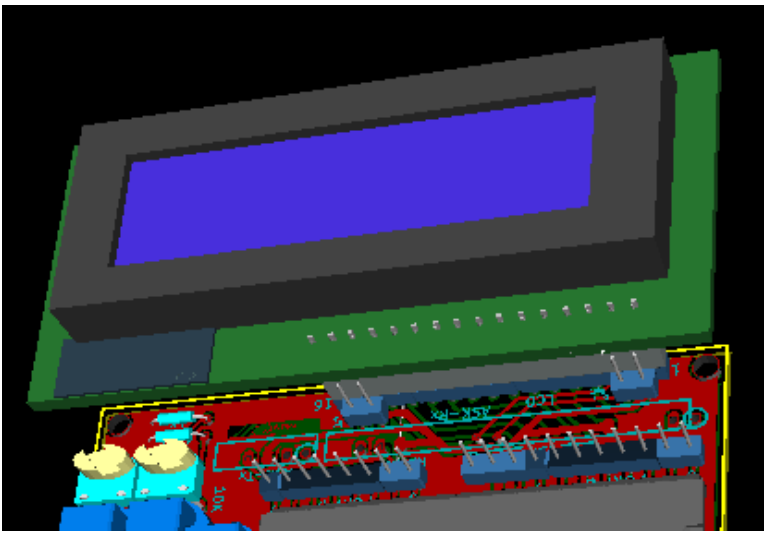


## LCD DISPLAY

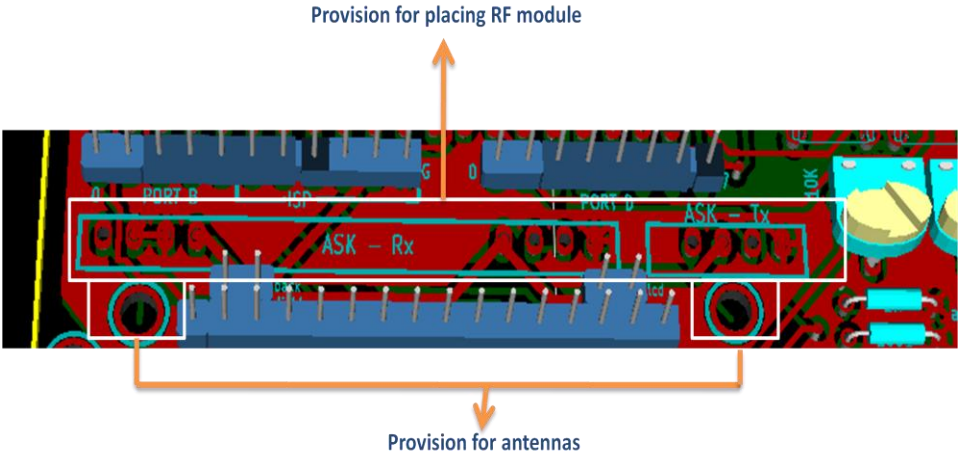
The board has provisions for connecting a standard 16 pin LCD Display. However, it must be noted that the port provided for LCD interfacing using the ATmega is for 4- bit interfacing and not 8-bit wherein the pins B0, B1, B2 of the MCU are used for RS, R/W and E and the pins B4, B5, B6, B7 are used for data bus lines.

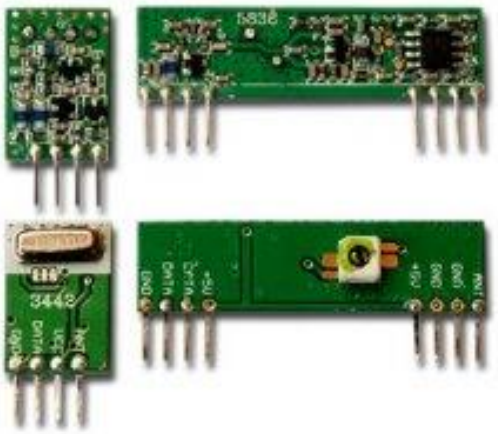


## RF communication

The board includes provision for connecting 433 MHz ASK Transmitter and Receiver Modules to transmit and receive serial data with a 100 m range in open space.

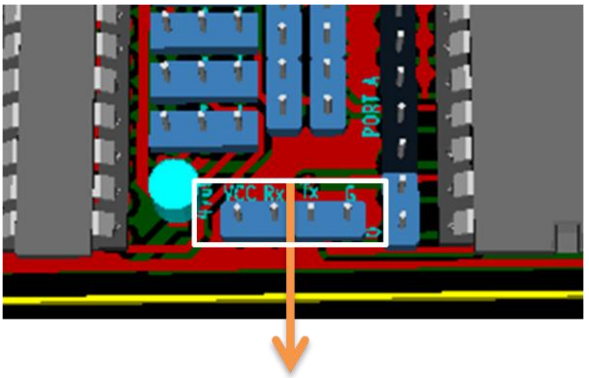


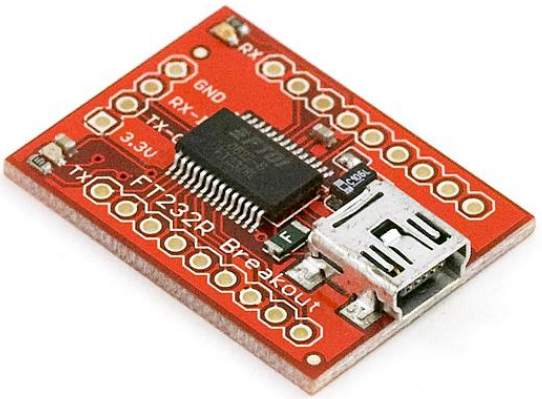
**Rx and Tx modules**

## FT232

The FT232 is a USB to serial UART interface with optional clock generator output.



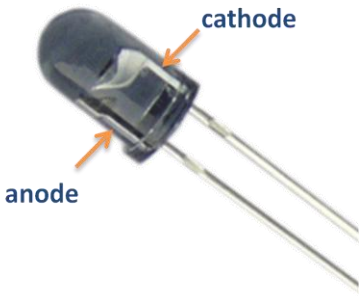

Provision for placing Ft232 module



## Sensor Board

The sensor board mainly consists of 4 IR Receiver-Transmitter pairs and an IC LM324.

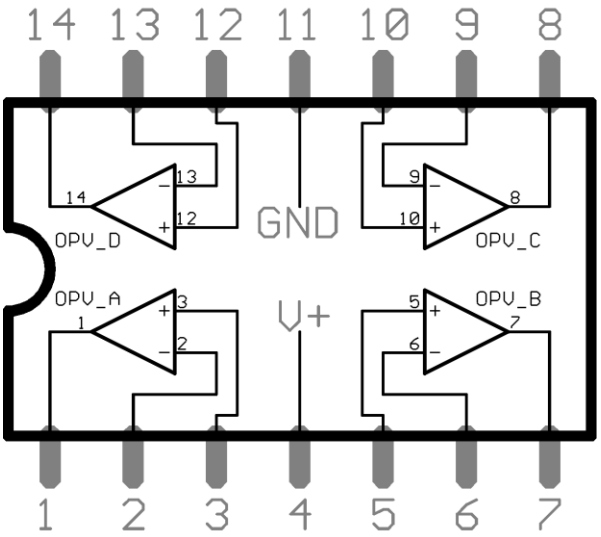
**IR Transmitter – IR LED.**

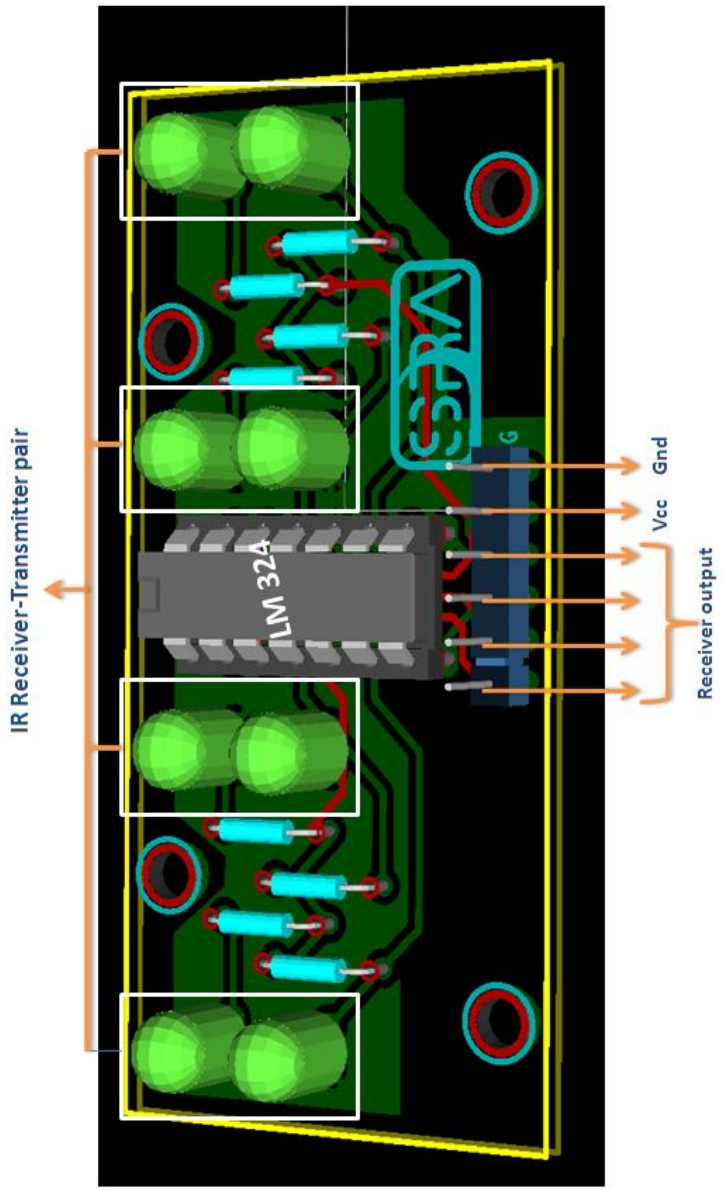



**IR Receiver - Photodiode**

**LM 324**

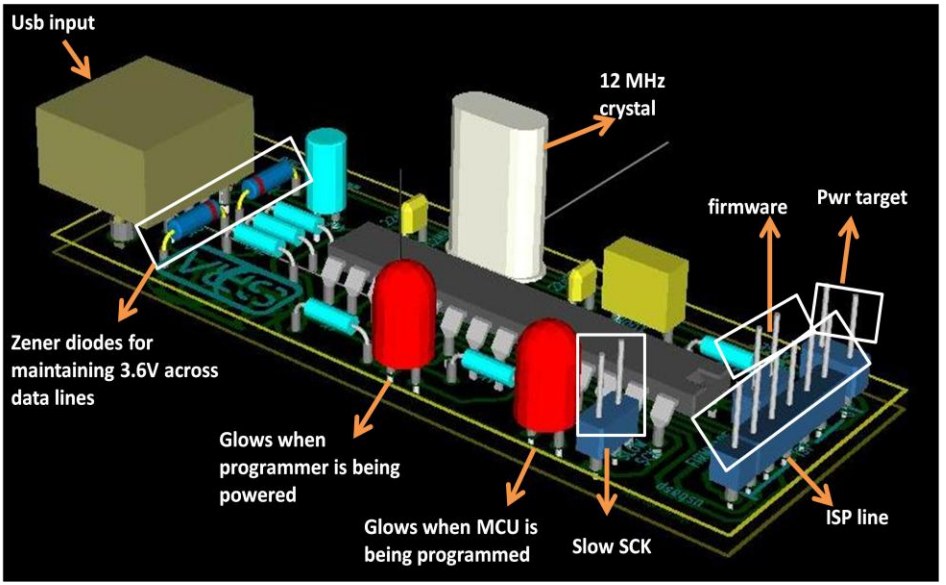
LM 324 consists of 4 independent, high gain operational amplifiers. Here, the op amps are used as buffers (i.e. the inverting input is connected directly to the output)




IR Receiver-Transmitter pair
LM 324
Gnd
Vcc
Receiver output

## USBasp

USBasp is a USB programmer used for burning the codes into the MCU on your development board.




**Some pins to know when using the programmer:**

- When programming your development for the first time, put a jumper on the pins with SLOW SCK and firmware written near them. (You don't need to this, we've already programmed it once for you ;) ) Putting the SLOW SCK jumper reduces your programming speed, which is desirable at times, when it fails to program at a faster speed.

- Of the extreme pins, whose wires are connected to the development board, the one having an initial 'G' near it printed on the board, is the ground pin. The pin adjacent to it is the VCC pin. The remaining 4 are the programming pins which go

to pins 6-9 of the ATMEGA. So, the 6-wire ribbon from the programmer board goes to the 6-11 pins of the ATMEGA.

- The power target (pwr target) jumper when put on, allows the MCU to be powered by the PC, when connected to the PC.

  **Warning: Do not put the pwr target jumper when the MCU is already supplied with external 12V power supply. You can do so ONLY if the supply switch on the development board is not pressed.**

- The glowing of red LED signifies that the process of programming is on. The green LED should ideally glow when the programmer is connected to the PC (with the power target jumper on the board) and is idle.

# Coding essentials

## Some commonly used functions:

Remember that setting a bit is setting it high or to 1, and clearing a bit is making it low or 0.

1. sbi - set a bit.

Syntax : void sbi( u08 register , u08 bit)

Sets a bit in a register. For example, to set the 0th bit of Port D, you can use:

sbi ( PORTD , 0 ) ; or sbi ( PORTD , PD0 ) ;

2. cbi - clear a bit.

Syntax : void cbi( u08 register , u08 bit)

Clears a bit in a register. For example, to clear the 2nd bit of Port B, you can use:

cbi ( PORTB , 2 ) ;

3. delay - introduces a delay of ms milliseconds

Syntax : void _delay_ms ( double ms )

## AVR Registers:

The 32 IO pins of the ATMega16 are divided into 4 ports, A, B, C, and D. Each port has 3 associated registers. For example, for port D, these registers are referred to in C-language by PORTD, PIND, and DDRD. For port B, these would be PORTB, PINB, and DDRB, etc. In C-language, PORTD is really a macro, which refers to a number that is the address of the register in the AVR, but it is much easier to remember PORTD than some arbitray hexadecimal number.

- DDR stands for Data Direction Register. There is one DDR register for each Data Input Port, and they are named for the port they control: DDRA, DDRB, DDRC, DDRD.A 1 makes the corresponding pin an output, and a 0 makes the corresponding pin an input.
  For example: DDRB = 0x0F ; //set first 4 pins of port B to output and next 4 to input
- Independent of the setting of Data Direction bit DDx, the port pin can be read through the PINx Register bit.
- The PORTx register functions differently depending on whether a pin is set to input or output. PORTx controls the value at the physical IO pins on PORTx**.**

## AVR Headers:

#include<avr/io.h> : This header file includes the appropriate IO definitions for the device.

#include<stdlib.h> : This file declares some basic C macros and functions as defined by the ISO standard, plus some AVR-specific extensions.

#include<compat/deprecated.h> : This header file contains several items that used to be available in previous versions of this library, but have eventually been deprecated over time. These items are supplied within that header file for backward compatibility reasons only, so old source code that has been written for previous library versions could easily be maintained until its end-of-life. Use of any of these items in new code is strongly discouraged.

#include<util/delay.h> : Contains convenience functions for busy/wait delay loops. The functions available allow the specification of microsecond, and millisecond delays directly.

#include<avr/eeprom.h> : This header file declares the interface to some simple library routines suitable for handling the data EEPROM contained in the AVR microcontrollers.

## Points to remember –

1. Murphy's Law ("If anything can go wrong, it will") will haunt you throughout your attempts, but refuse to give up!
2. Patience is the biggest virtue!
3. Sometimes banging the board helps.
4. Whenever in doubt, first check the connectivity.
5. You haven't learnt until you have blown up a few components on your board.
6. Last and the most important – **HAVE FUN!!**


## *In case of queries* contact –*

**ELECTRONICS**

Saurabh Doiphode
T.Y.B.tech
saurabhd29@gmail.com

Shweta Khushu
S.Y.B.Tech
shweta.khushu@gmail.com

**CODING**

Vivek Nhattuvetty
T.Y.B.Tech
vivek.9967@gmail.com

Bhavi Jagwani
S.Y.B.Tech
jagwani.bhavi@gmail.com

*Conditions apply on queries :-P