Line Follower Robot

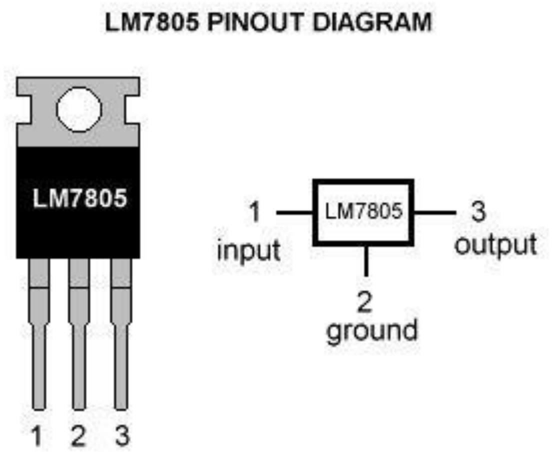
This will be a quick guide which will come handy for making your line-follower robot. We will divide this guide into three parts :

1. Hardware schematic
2. Coding

Hardware Implementation :

Voltage Regulator

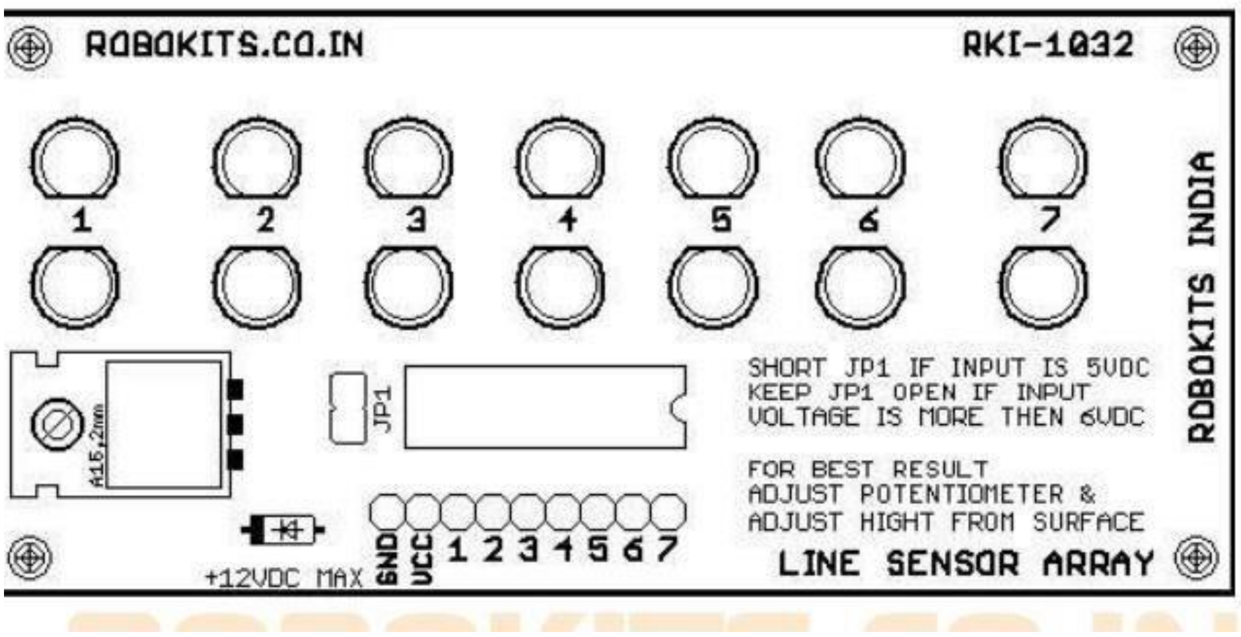
LM7805 :



This IC is used to get regulated voltage supply of 5V.PIN1 will be connected to 12v or 9v input. PIN2 will be connected to GND. We will get regulated 5v output on PIN3. Capacitors are also connected between Input and GND and also between Output and GND to filter out the AC component.

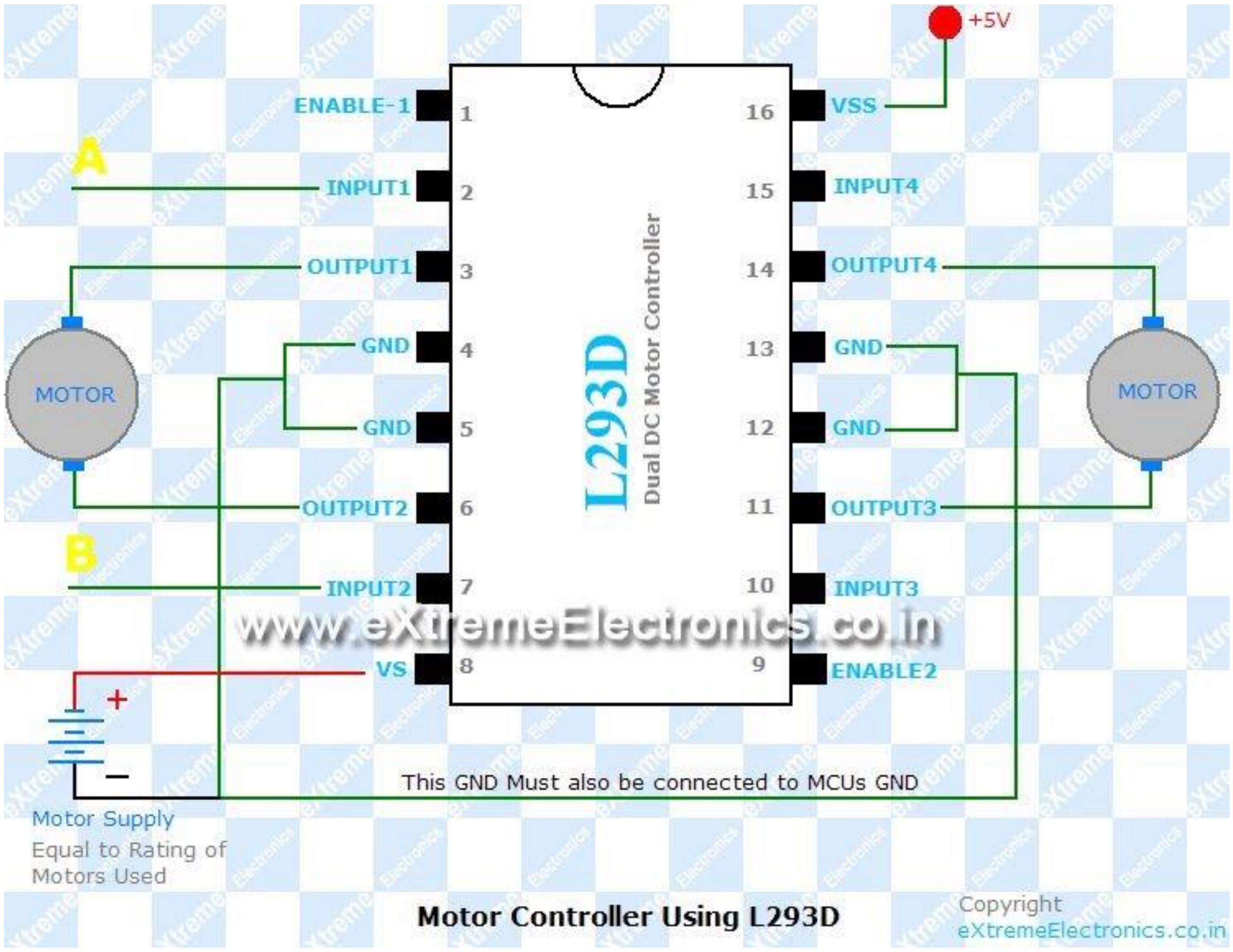
Line Sensor :

A line sensor is a collection of IR sensors aligned in a line. It consists of a IR emitter and a IR receiver. The IR reflected back from the line is received by the IR receiver and this gives output in the form of 0s and 1s.



- The above 9 circles (from GND to 7) should be soldered with male headers.
- The VCC pin will be given 5V and GND will be given ground using jumper wires.
- Headers from 1 to 7 will give output 0 or 1 depending on the position of the respective sensor, i.e. either on the line or off the line.
- It depends on your manufacturer whether your sensor will give 1 on line or off line.
- Their potentiometers provided with the line sensor to adjust the sensitivity.
- These 7 jumpers from your headers will go to your corresponding pins on the MCU where it is programmed to take input .

Motor Driver IC (L293D) :

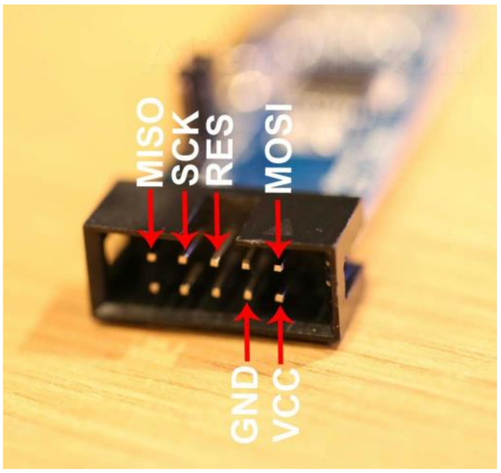


- Motor driver is used to amplify current and voltage which will make motor to run.
- Parallel data output from Decoder will go to 4 input pins of L293D. Corresponding output pins will be connected to motors.
- Enable-1 and Enable-2 pins will be connected to 5v to enable the motor driver circuit.
- Vs will be provided by the voltage required to drive motor. Vss is connected to 5v required for operation of IC circuit.

Visit this link for a detailed explanation of motor drivers. Click here

Interfacing the Microcontroller with the Computer :

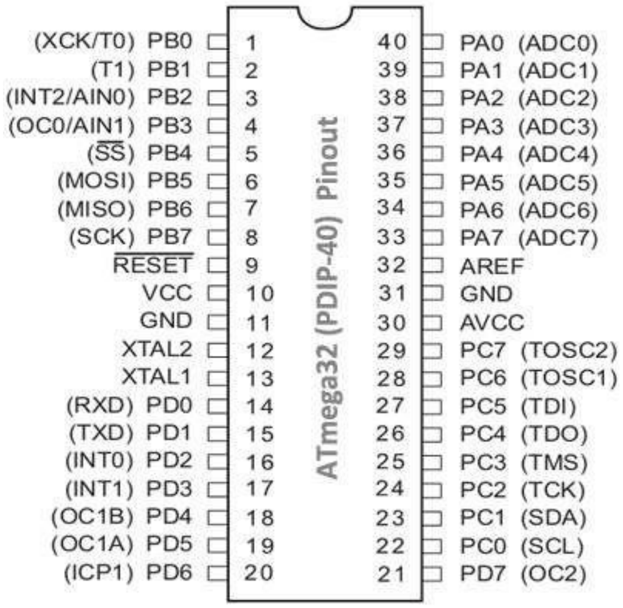
A USBASP programmer is used to interface the microcontroller with the computer. This is the pinout of the USBASP programmer:



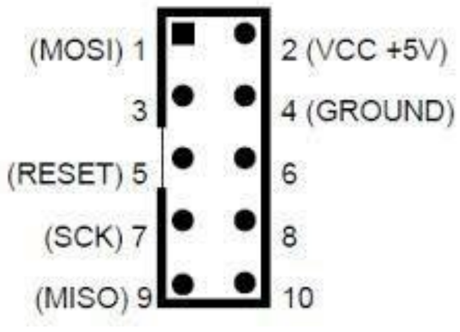
It is recognised by the triangular symbol present on the top right, which shows the position of the MOSI pin.

Connect the corresponding pin of the USBASP programmer with the corresponding pins of the microcontroller.

Use the following pinout of Atmega32 to find the corresponding pins:



MOSI - PIN6
MISO - PIN7
SCK - PIN8
RES - PIN9
VCC - PIN10
GND - PIN11



USBASP pinout

You can connect it using FRC cable on the USBASP programmer and on the GCB using FRC port.

FRC Cable

FRC Port

The orientation of the FRC cable on both sides should be kept in mind with the help of the nodge given on its black part.

Visit this link for a detailed explanation of setting up Atmel Studio, installing USBASP drivers. Click here

Visit this link for a detailed explanation of connection between Microcontroller and USBASP programmer using FRC cables. Click here

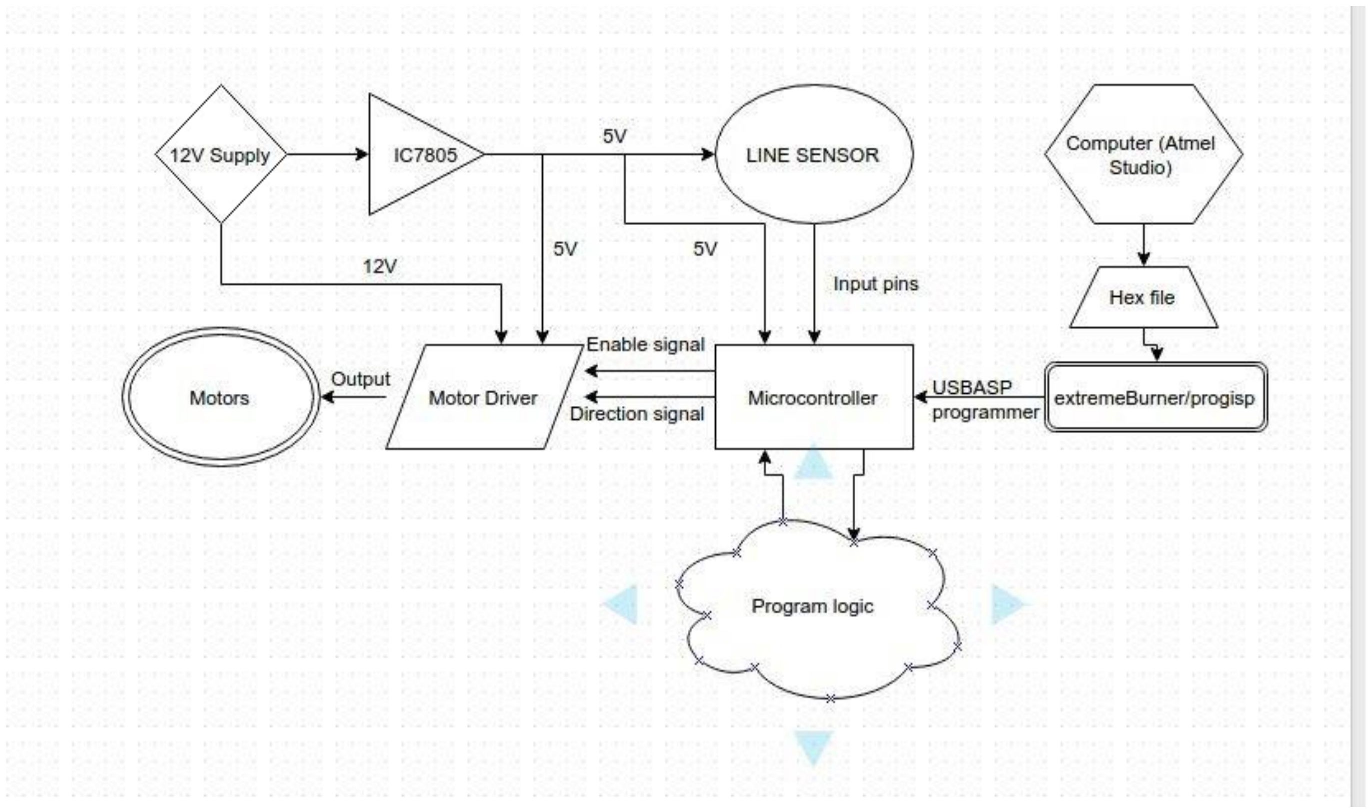
Your development board :

A development board is nothing but a general circuit board designed with a microcontroller solder on it, a power supply circuit and other connectors like male headers, FRC ports, screw connectors.

TIPS :

- Keep in mind to **NOT** solder any microcontroller or IC directly on the GCB, but use a bed.
- Make sure you do not short positive or negative terminals.
- **NEVER SUPPLY VOLTAGE WITH REVERSE POLARITY TO YOUR CIRCUIT,** it will cost you a lot.
- Make sure your MCU and other ICs get 5V from 7805 circuit wherever needed. **DO NOT GIVE 12V TO ANY COMPONENT UNLESS IT IS MENTIONED.**

COMPLETE SCHEMATIC OF LINE FOLLOWER



Coding :

- After making a project in Atmel Studio and selecting your microcontroller, start by writing DDR for the pins you will be using in the main() of your program.
  For example,

  DDRB |= 1<<PINB0; // Using PB0 for an output i.e. setting PB0

  DDRC &= ~(1<<PINC3) // Using PC3 for input i.e. clearing PC3

  DDRD = 0b00001111; // Using PD0 to PD3 for output and PD4 to PD7 for input.

- The input is ready at your input pins. You have to code your MCU to read the input on those pins and store somewhere for use. You may use functions **bit_is_set()** or **bit_is_clear()** for this.

  For example, **if(bit_is_set(PINB,3)** // Checking whether PB3 is getting 5V input or
  ```
  {                    //not
      //Do something
  }
  ```
  NOTE : For using this I will have to specify that I am using PB3 as input pin y using DDR before.
  Also, bit_is_clear() is negation of bit_is_set()

- Once you have the state of your line-sensor, you know what your robot's position is. Now you have to tell your robot where it should go.
- For this, design a logic using **if-else, while, switch-case etc.** to act accordingly, i.e. if your sensors **say** line is on the right side of the sensor, turn the robot right.
- To make the motors do the work, you will need 6 pins to be programmed for output.
- 4 pins for giving direction to the L293D motor driver (input pins on motor-driver) 2 pins for enable pins on the motor driver. Use **PORT** for this For example,
  ```
  DDRD |= 1<<PIND0; // Configuring PD0 for output
  while(1)
  {
      PORTD |= 1<<PIND0; // Sending 5V signal on PD0
  }
  ```
- The direction pins for one motor should have a 1 (5V) on one terminal and 0 (0V) on other and same for the other motor. On reversing the polarity of direction pins, the motor will turn in the other direction.
- You can use the enable pin to start and stop your motor to run. i.e. To make your motor turn right, you need to turn off right motor. Say the right motor is connected on the **ENABLE1** side of motor driver. Instead of changing direction pins, you can send a **0 (0V) on ENABLE1** to stop that motor.
- **NOTE : To make your motor run, it needs a valid signal pair(1 and 0) on direction pin and a 1 on ENABLE pin.**
- Make sure you right your logic and input reading code in the **while(1)** loop so that the program runs continuously.
- Once your program is ready, "Build" it in Atmel Studio and then open a burner program like "progisp" or "eXtremeBurner" and browse the hex file from "Documents/Atmel

Studio/6.2/Project-name/Project-name/Debug” and burn it on your microcontroller by clicking burn. (where Project-name is name of your project)
Make sure your USBASP programmer is plugged in computer and the microcontroller on the board.

Materials used :

Solder wire
GCB
Screw connectors
Male - female header
LED green and red
Resistor(330ohm)
7805
Capacitor(1uf and 10uf)
On/off push button
Line sensor
Steel grip
Atmega32
ATmega 32 IC bed
USBASP programmer
10 pin FRC port
Battery (12V)
Jumper wire
Single strand
L293D + IC bed
Castor wheel
Motor
Chasis (Readymade or Made by yourself)
Wheels
Multistrands wire
Rainbow jumper wire