

CONTENTS

1. Summary
2. Introduction
 - 2.1. What is a line follower?
 - 2.2. Why build a line follower?
 - 2.3. Prerequisites
 - 2.4. The AVR microcontroller
3. Overview
 - 3.1. Block Diagram and Architectural Overview
 - 3.2. The Algorithm
4. Implementation
 - 4.1. Sensor Circuit
 - 4.2. Motor Interface and Control Circuit
 - 4.3. Source Code
5. Possible Improvements
6. References and Resources
 - 6.1. Books and Links
 - 6.2. Tools of the trade
 - 6.3. Electronic shops
 - 6.4. Parts and Prices

INTRODUCTION

What is a line follower?

Line follower is a machine that can follow a path. The path can be visible like a black line on a white surface (or vice-versa) or it can be invisible like a magnetic field.

Why build a line follower?

Sensing a line and maneuvering the robot to stay on course, while constantly correcting wrong moves using feedback mechanism forms a simple yet effective closed loop system. As a programmer you get an opportunity to 'teach' the robot how to follow the line thus giving it a human-like property of responding to stimuli.

Practical applications of a line follower : Automated cars running on roads with embedded magnets; guidance system for industrial robots moving on shop floor etc.

Prerequisites:

Knowledge of basic digital and analog electronics.

(A course on Digital Design and Electronic Devices & Circuits would be helpful)

C Programming

Sheer interest, an innovative brain and perseverance!

The AVR microcontroller:

“Atmel's AVR® microcontrollers have a RISC core running single cycle instructions and a well-defined I/O structure that limits the need for external components. Internal oscillators, timers, UART, SPI, pull-up resistors, pulse width modulation, ADC, analog comparator and watch-dog timers are some of the features you will find in AVR devices.

AVR instructions are tuned to decrease the size of the program whether the code is written in C or Assembly. With on-chip in-system programmable Flash and EEPROM, the AVR is a perfect choice in order to optimize cost and get product to the market quickly.”

<http://www.atmel.com/products/avr/>

Apart from this almost all AVR's support In System Programming (ISP) i.e. you can reprogram it without removing it from the circuit. This comes very handy when prototyping a design or upgrading a built-up system. Also the programmer used for ISP is easier to build compared to the parallel programmer required for many old uCs. Most AVR chips also support Boot Loaders which take the idea of In System Programming to a new level. Features like I2C bus interface make adding external devices a cakewalk. While most popular uCs require at least a few external components like crystal, caps and pull-up resistors, with AVR the number can be as low as zero!

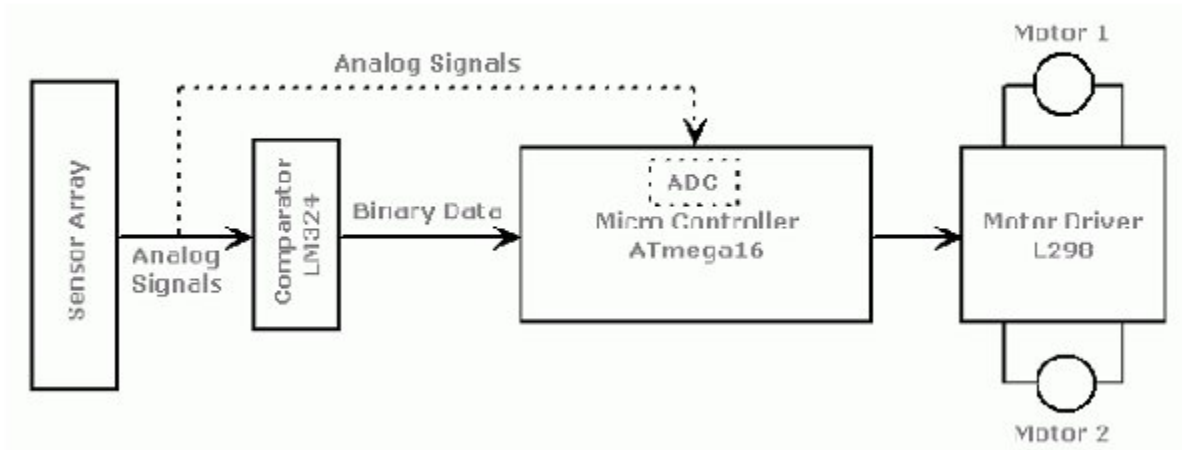
Cost: AVR = PIC > 8051 (by 8051 I mean the 8051 family)

Availability: AVR = PIC < 8051

Speed: AVR > PIC > 8051

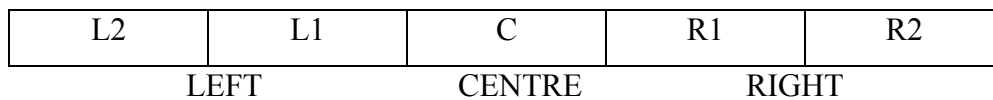
Built-in Peripherals: This one is difficult to answer since all uC families offer comparable features in their different chips. For a just comparison, I would rather say that for a given price AVR = PIC > 8051.

OVERVIEW



Block Diagram

The robot uses IR sensors to sense the line, an array of 5 IR LEDs (Tx) and sensors (Rx), facing the ground has been used in this setup. The output of the sensors is an analog signal which depends on the amount of light reflected back, this analog signal is given to the comparator to produce 0s and 1s which are then fed to the uC.



Sensor array

Let us assume that when a sensor is on (IR is on white surface) the line it reads 0 and when it is off (IR is on black surface) the line it reads 1, Since Rx output is given to inverting terminal of the comparator so sensor output gets complemented

Therefore, for black surface output is logic 1 and for white surface output is logic 0

Algorithm:

1.

L2	L1	C	R1	R2
0	0	1	0	0

Bot should go straight

2.

L2	L1	C	R1	R2
X	Y	Z	0	0

When $XYZ > 0$, i.e. for $X=1, Y=1, Z=1$ for eg. $XYZ=111=7$ (in decimal) which is greater than 0

Bot should turn left, turning speed will vary in accordance with the value of XYZ.

3.

L2	L1	C	R1	R2
0	0	X	Y	Z

When $XYZ > 0$, i.e. for $X=1, Y=1, Z=1$ for eg. $XYZ=111=7$ (in decimal) which is greater than 0

Bot should turn right, turning speed will vary in accordance with the value of XYZ.

4.

L2	L1	C	R1	R2
0	0	0	0	0

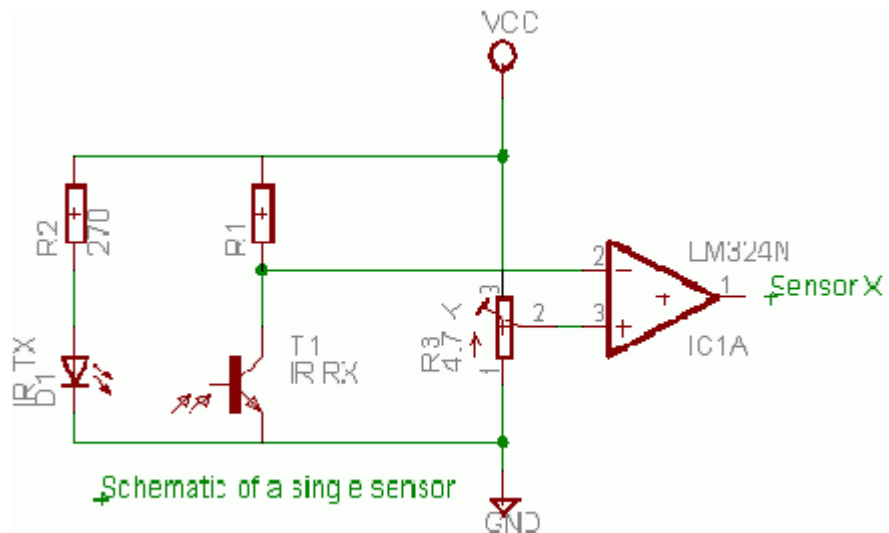
If there is no line we have to detect the line by 'U' turn, For effective path finding history should be taken and execute the last action.

Implementation

IMPLEMENTATION

Sensor Circuit:

The resistance of the sensor decreases when IR light falls on it. A good sensor will have near zero resistance in presence of light and a very large resistance in absence of light.

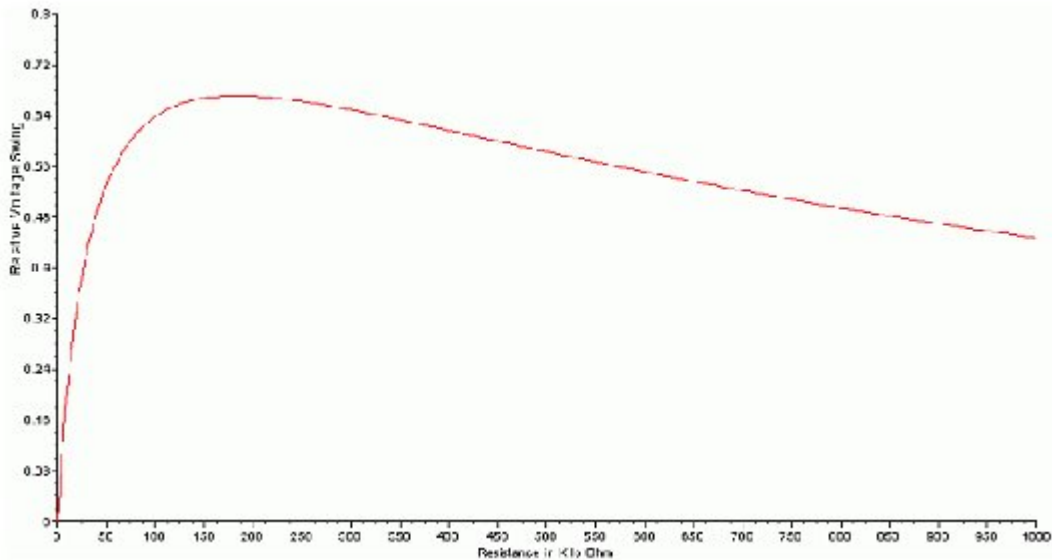


We have used this property of the sensor to form a potential divider. The potential at point '2' is $R_{\text{sensor}} / (R_{\text{sensor}} + R1)$.

Again, a good sensor circuit should give maximum change in potential at point '2' for no-light and bright-light conditions. This is especially important if you plan to use an ADC in place of the comparator

To get a good voltage swing, the value of R1 must be carefully chosen. If $R_{\text{sensor}} = a$ when no light falls on it and $R_{\text{sensor}} = b$ when light falls on it. The difference in the two potentials is:

$$V_{\text{cc}} * \{ a/(a+R1) - b/(b+R1) \}$$



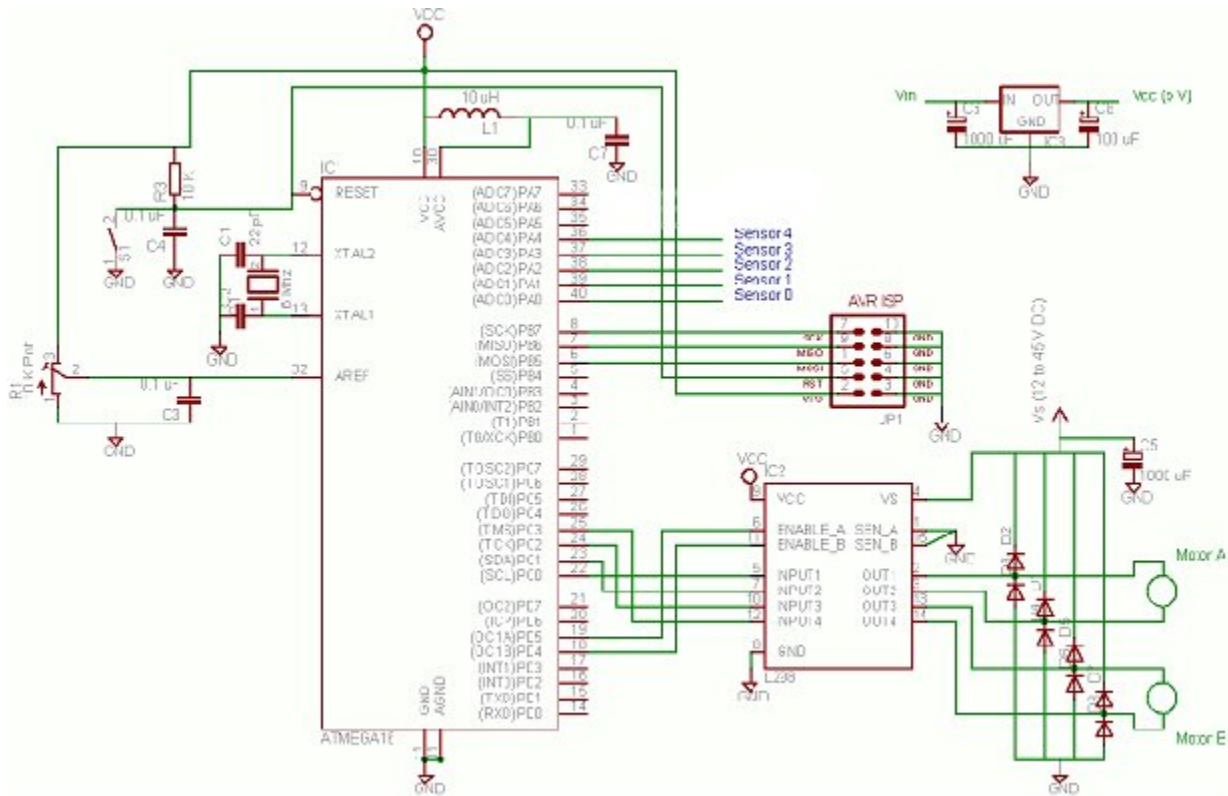
$$\begin{aligned}
 \text{Relative voltage swing} &= \text{Actual Voltage Swing} / V_{cc} \\
 &= V_{cc} * \{ a/(a+R1) - b/(b+R1) \} / V_{cc} \\
 &= a/(a+R1) - b/(b+R1)
 \end{aligned}$$

The sensor I used had $a = 930 \text{ K}$ and $b = 36 \text{ K}$. If we plot a curve of the voltage swing over a range of values of $R1$ we can see that the maximum swing is obtained at $R1 = 150 \text{ K}$ (use calculus for an accurate value).

There is a catch though, with such high resistance, the current is very small and hence susceptible to be distorted by noise. The solution is to strike a balance between sensitivity and noise immunity. I chose value of $R1$ as 60 K . Your choice would depend on the 'a' and 'b' values of your sensor.

If you found this part confusing, use a 10K resistor straightaway, as long as you are using a comparator it won't matter much.

Motor Interface and Control Circuit:

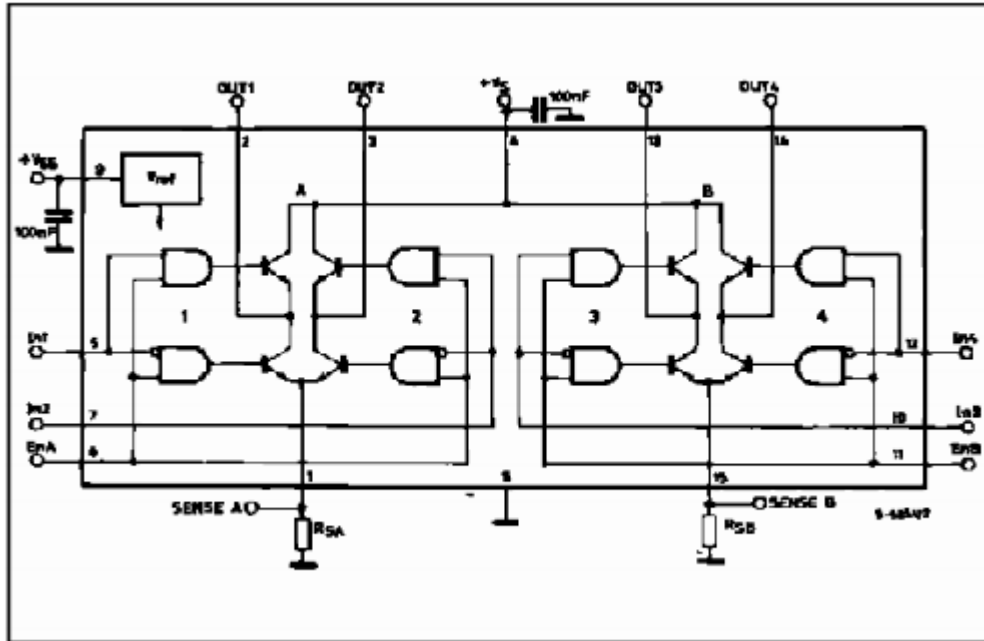


The 5 sensors are connected to PORTA.

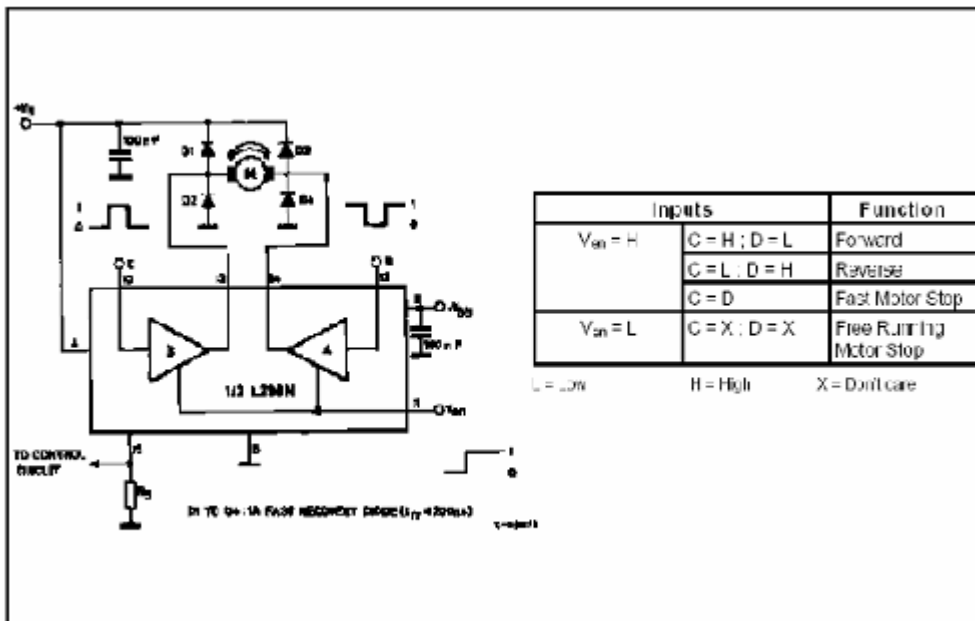
You need not connect anything to AVCC and AREF, it is required only if ADC is used.

The L298 Motor Driver has 4 inputs to control the motion of the motors and two enable inputs which are used for switching the motors on and off. To control the speed of the motors a PWM waveform with variable duty cycle is applied to the enable pins. Rapidly switching the voltage between Vs and GND gives an effective voltage between Vs and GND whose value depends on the duty cycle of PWM. 100% duty cycle corresponds to voltage equal to Vs, 50 % corresponds to 0.5Vs and so on. The 1N4004 diodes are used to prevent back EMF of the motors from disturbing the remaining circuit. Many circuits use L293D for motor control, I chose L298 as it has current capacity of 2A per channel @ 45V compared to 0.6 A @ 36 V of a L293D. L293D's package is not suitable for attaching a good heat sink, practically you can't use it above 16V without frying it. L298 on the other hand works happily at 16V without a heat sink, though

it is always better to use one. At beginning level use L293D because it is easy to replace and standard DIP IC



Internal Schematic of L298



Truth Table for controlling the direction of motion of a DC motor

Source Code

```
/******  
Project : Line Tracker  
Version :  
Date   : 09/09/2009  
Author : Prakash.N.M  
Company : Home  
Comments:  
Chip type      : ATmega16  
Program type   : Application  
Clock frequency : 1.0 MHz (Internal Clock)  
Memory model   : Small  
External SRAM size : 0  
Data Stack size  : 256  
*****/  
  
#include <avr\io.h>  
#include <avr\iom16.h>           //atmega16 header  
#include <avr\interrupt.h>      // interrupt header  
#define F_CPU 1000000           // clock frequency  
#include <util/delay.h>         // dealy header  
#define powerl OCR1A            // output compare register 1A  
#define powerr OCR1B            // output compare register 1B  
  
#define nc bit_is_clear(PINA,3) // nc - center sensor is not on black line  
#define nl1 bit_is_clear(PINA,1) // nl1 - left sensor 1 is not on black line  
#define nr1 bit_is_clear(PINA,6) // nr1 - right sensor 1 is not on black line  
#define nl2 bit_is_clear(PINA,2) // nl2 - left sensor 2 is not on black line  
#define nr2 bit_is_clear(PINA,0) // nr2 - right sensor 2 is not on black line  
  
#define c bit_is_set(PINA,3)     // c - center sensor is on black line  
#define l1 bit_is_set(PINA,1)    // l1 - left sensor1 is on black line  
#define r1 bit_is_set(PINA,6)    // r1 - center sensor1 is on black line  
#define l2 bit_is_set(PINA,2)    // l2 - center sensor2 is on black line  
#define r2 bit_is_set(PINA,0)    // r2 - center sensor2 is on black line
```

```

int motor(int pl,int pr)          //This function is used to vary the speed of motor by passing parameters
{
powerl=pl;
powerr=pr;
//return(powerl,powerr);        //Need not necessary just for checking
}

int main(void)
{
//Initialization
unsigned char left=0,right=0;
DDRD|=(1<<PD4)|(1<<PD5);
// PD2 and Pd3 for the signal receiving and the pins pd4 and pd5 for the PWM
PORTD|=(0<<PD4)|(0<<PD5);
// port c for motor control
DDRC|=(1<<PD0)|(1<<PD1)|(1<<PD2)|(1<<PD3);
PORTC=(0<<PD0)|(0<<PD1)|(0<<PD2)|(0<<PD3); //Need not necessary just for initialization
//Port A pins as input for sensors
DDRA=0X00;
//Enable internal pull ups
PORTA=0XFF;
//To set PWM generation in timer1
ICR1=20000;
TCCR1A|=(0<<COM1A0)|(1<<COM1A1)|(0<<COM1B0)|(1<<COM1B1)|(0<<FOC1A)|(0<<FOC1B)|(0<<WGM11)|(1<<WGM10);
TCCR1B|=(0<<ICNC1)|(0<<ICES1)|(0<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(0<<CS10);

//infinite loop
while (1)
{

left|=(12<<2)|(11<<1)|(c<<0);
right|=(r2<<2)|(r1<<1)|(c<<0);
if(left==right)

```

```
{
if(c)
    {
        PORTC = 0b0101;    // Forward direction
        motor(150, 150);   // Parameter value should not more than 255
                           //i.e speed of the motor
    }
else
    PORTC = 0b0000;       // stop the bot wen it is out of line
}
else
    {
if(left>right)
    {
        PORTC = 0b1001;    // left direction bit o&1 is for right motor
        motor((left-right)*35, (left-right)*35);// bit 2&3 is for left motor
    }
else
    {
        PORTC = 0b0110;    // right direction bit o&1 is for right motor
        motor((left-right)*35, (left-right)*35);// bit 2&3 is for left motor
    }
}
}
}
```

POSSIBLE IMPROVEMENTS:

- Use of Hysteresis in sensor circuit using LM339
- Use of ADC so that the exact position of the line can be interpolated
- Use of Wheel Chair or three wheel drive to reduce traction.
- General improvements like using a low dropout voltage regulator, lighter chassis etc

REFERENCES AND RESOURCES

Books:

Programming and Customizing the AVR Microcontroller – Dhananjay V. Gadre
Parallel Port Complete – Jan Axelson

Links:

Atmel Corp.

Makers of the AVR microcontroller

<http://www.atmel.com>

AVRbeginners.net

<http://www.avrbeginners.net/>

AVR assembler tutorial

Tutorial for learning assembly language for the AVR-Single-Chip-Processors AT90Sxxxx from ATMEL with practical examples.

<http://www.avr-asm-tutorial.net/>

One of the best sites AVR sites

<http://www.avrfreaks.net>

WinAVR

An open source C compiler for AVR

<http://sourceforge.net/projects/winavr>

PonyProg

A widely used programmer. Support for newer chips is added periodically. Can also program PICs and EEPROMS

<http://www.lancos.com/prog.html>

Basic Electronics

<http://www.kpsec.freeuk.com/>

Williamson Labs

Nice animated tutorials, articles and project ideas.

<http://www.williamson-labs.com/home.htm>

Small Robot Sensors

http://www.andrew.cmu.edu/user/rjg/websensors/robot_sensors2.html

Robotics India

An Indian site devoted to robotics. Must see

<http://www.roboticsindia.com/>

Seattle Robotics Society

<http://www.seattlerobotics.org/>

Line Follower ROBOT

Award winner from VingPeaw Competition 2543, the robot built with 2051, L293D, and four IR sensors. Simple circuit and platform, quick tracking and Easy to understand program using C language.

<http://www.kmitl.ac.th/~kswichit/LFrobot/LFrobot.htm>

Tools: AVR Studio,VMLAB

For writing code in assembly and simulation of code. Current versions has AVR-GCC plug-in to write code in C.

Compilers: IAR, Image Craft , Code Vision AVR, WinAVR

Programmers: Pony Prog, AVR Dude, AVRISP and many more.

Evaluation Boards: STK200, STK500 from Kanda Systems

Shops:

1. Anjanta electronics,Richie street, Chennai.
2. Modi electronics,Richie street, Chennai.
3. Kailash electronics,Richie street, Chennai.
4. Naresh electronics,Pondicherry.

Parts and Prices:

Part	Approximate Price in Indian Rupees
Visible Light Leds	1.00
White or Bicoloured	5.00
IR LED	3.00
IR Sensor	7.00
Capacitor (small values)	0.25 to 2.00
Capacitor (large values / electrolytic)	2.00 to 20.00 Or more
Resistors (1/4 W)	0.25
Variable Resistor (Preset)	2.50
Variable Resistor (Pot)	8.00
Microcontrollers	40 to 450
AT89C2051 (8051 Core)	40
AT89C51 (8051 Core)	60
AT89S52 (8051 Core)	150
PIC16F84A (PIC Core)	120
Atmega16(8051 Core)	160