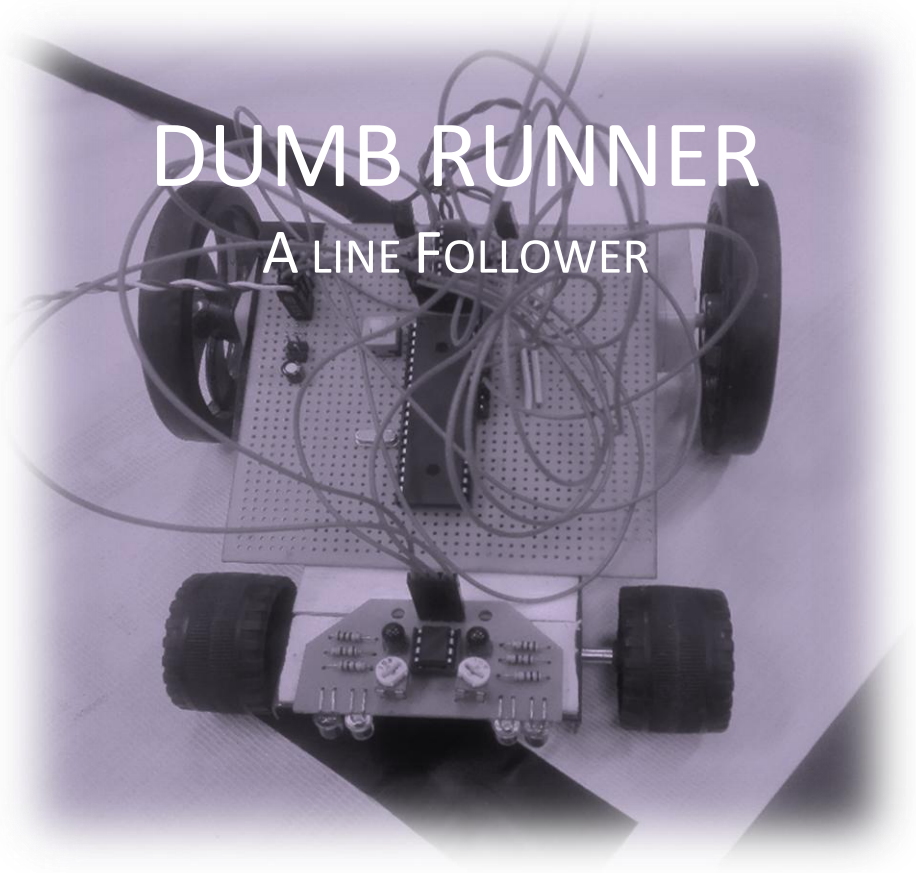
DUMB RUNNER

A LINE FOLLOWER



PROJECT BY GURU PRASATH G, II ECE A

DHINESH KUMAR , II ECE A

GUIDED BY MR. GANESH PRABHU, ASST PROOF.

ECE DEPT

Dumb Runner

1 OBJECTIVE

We began this project with a novel objective of learning *Embedded Systems.* We analyzed the current trend of choosing ***Arduino boards*** for amateur development of projects. But we thought of getting a professional exposure to the field of Embedded Systems and was guided by our Guide to Microchip's PIC® Microcontrollers, with the primary of objective of understanding the basic architecture of a Microcontroller and its working.

2 ABSTRACT

Our project is around PIC® Microcontroller 16F877A to develop a simple autonomous Line Follower which can be used to understand the necessary workflow of any Microcontrollers.

We made the project modular with three sections, as any system:

We made the project modular, so that we can upgrade or change the modules to suit our requirements. We are striving to upgrade this project to a Line Maze Solver, which may happen in near future.

3 PROJECT DESCRIPTION

Line Followers are basic autonomous robots which can trace the path of lines made by high contrast colours with the background. We were cost effective and made most of the modules by our own, which in turn refined our soldering and circuit designing skills.

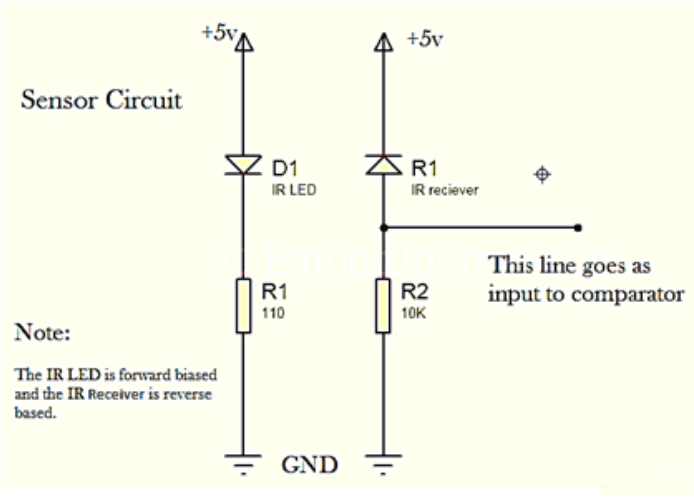
3.1 INPUTS

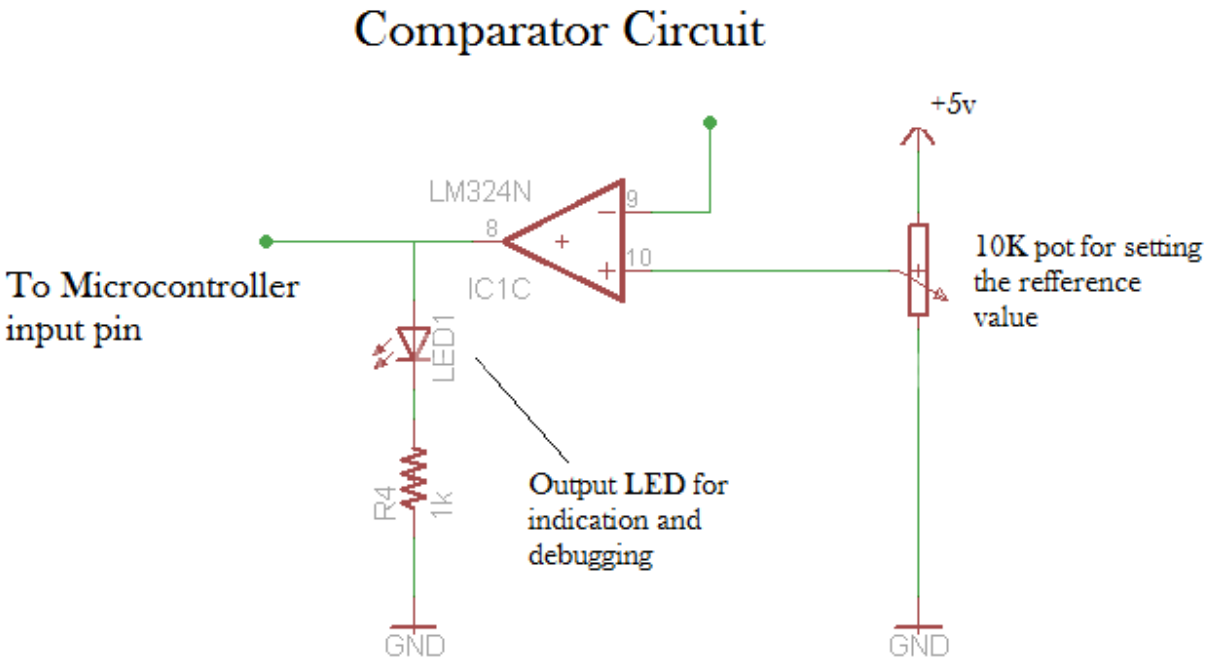
We used basic IR Tx and Rx pair as our sensing system.

And we used OPAMP as an external Comparator, since we have planned to upgrade the project with additional sensor pairs. So, PIC®'s internal voltage comparator is left free.

At present we have two sensor pairs, so we used a dual OPAMP IC LM358.

And as always, the input section is hand soldered by us, and we did etching to make the PCB.





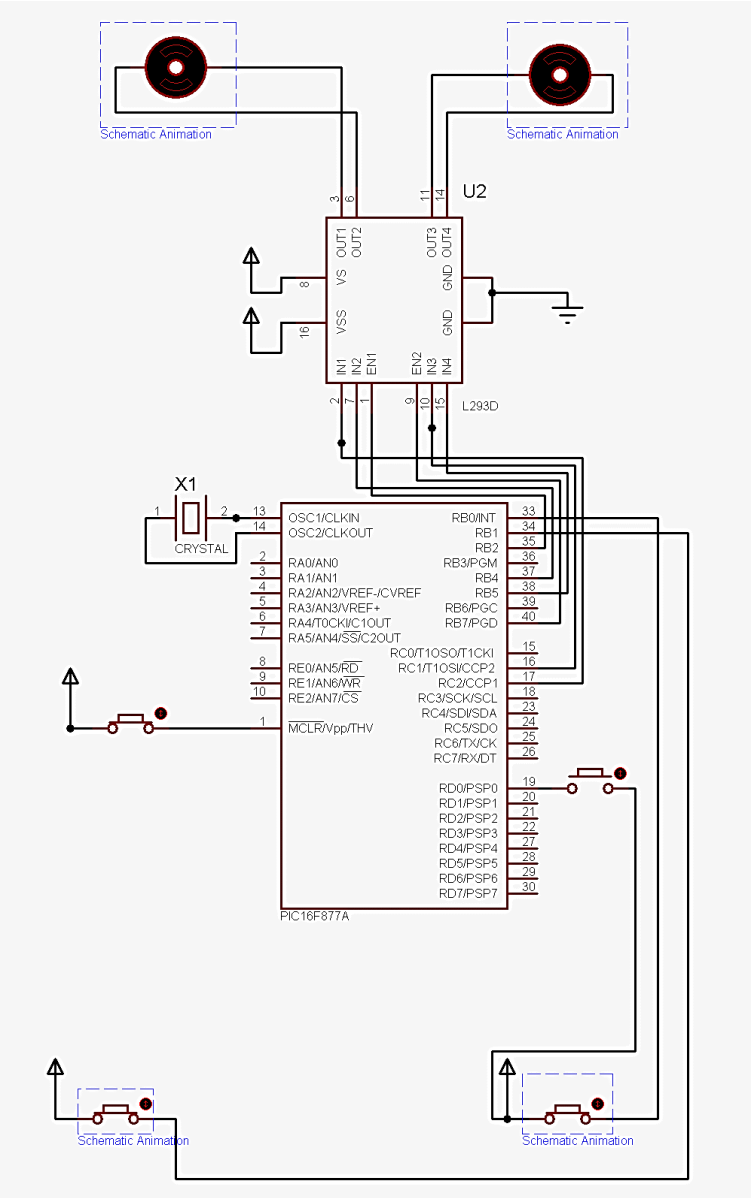
3.2 OUTPUTS

We used geared BO-DC motors at the outputs and used an H-Bridge to drive the motors. H-Bridge we used is IC L293D. The module is entirely modular and we can change the drive or the output to as any device.

3.3 CORE

Though we can drive the output section directly from the input section, as of present, we used the processing section to still refine our project.

We used Port B as our preliminary IO port and drived the motors using CCP modules so as to achieve PWM to control the motor speed.



3.3.1 Code

3.3.1.1 dumbFollower.c

```
#include "htc.h"
#define _XTAL_FREQ 12000000
#include "pwm.h"

#define EN1    RB2
#define C_M_R  RC2
#define CC_M_R RB4
#define CC_M_L RB5
#define C_M_L  RC1
#define EN2    RB7

#define S_L    RB1
#define S_R    RB0
__CONFIG(HS & WDTDIS & PWRTEN & BOREN & LVPDIS & DUNPROT & WRTEN & UNPROTECT);
void main(void)
{
   unsigned int d=500;
   TRISB=0x03;
   PORTB=0;
   TRISC=0x00;
   PORTC=0;
   TRISD=0XFF;
   PORTD=0;

   PWM1_Init(5000);
   PWM2_Init(5000);
   PWM1_Start();
   PWM2_Start();

   while(1){
      if(RD0==1){
        if(d<1000){
               if(d!=995)
        {d+=5;}
        else{d=50;}
     }
     __delay_ms(500);
      }

      if(S_L==1&&S_R==1){
     CC_M_L=0;
     CC_M_R=0;
     EN1=1;
     PWM1_Duty(d);
     EN2=1;
     PWM2_Duty(d);
      }
      if(S_L==0&&S_R==1){
     CC_M_L=1;
     CC_M_R=0;
     EN1=1;
     PWM1_Duty(d+25);
     EN2=1;
     PWM2_Duty(900-d);
```

```
      }
      if(S_L==1&&S_R==0){
     CC_M_L=0;
     CC_M_R=1;
     EN1=1;
     PWM1_Duty(900-d);
     EN2=1;
     PWM2_Duty(d+25);
      }
      if(S_L==0&&S_R==0){
     CC_M_L=0;
     CC_M_R=0;
     EN1=0;
     PWM1_Duty(0);
     EN2=0;
     PWM2_Duty(0);
      }
   }
}
```

3.3.1.2 pwm.h

```
#define TMR2PRESCALE 4
long freq;

int PWM_Max_Duty()
{
   return(_XTAL_FREQ/(freq*TMR2PRESCALE));
}

PWM1_Init(long fre)
{
  PR2 = (_XTAL_FREQ/(freq*4*TMR2PRESCALE)) - 1;
  freq = fre;
}

PWM2_Init(long fre)
{
  PR2 = (_XTAL_FREQ/(freq*4*TMR2PRESCALE)) - 1;
  freq = fre;
}

PWM1_Duty(unsigned int duty)
{
  if(duty<1024)
  {
    duty = ((float)duty/1023)*PWM_Max_Duty();
    CCP1X = duty & 2;
    CCP1Y = duty & 1;
    CCPR1L = duty>>2;
  }
}

PWM2_Duty(unsigned int duty)
{
  if(duty<1024)
  {
    duty = ((float)duty/1023)*PWM_Max_Duty();
    CCP2X = duty & 2;
    CCP2Y = duty & 1;
```

```
    CCPR2L = duty>>2;
  }
}

PWM1_Start()
{
  CCP1M3 = 1;
  CCP1M2 = 1;
  #if TMR2PRESCALE == 1
  T2CKPS0 = 0;
  T2CKPS1 = 0;
  #elif TMR2PRESCALE == 4
  T2CKPS0 = 1;
  T2CKPS1 = 0;
  #elif TMR2PRESCALE == 16
  T2CKPS0 = 1;
  T2CKPS1 = 1;
  #endif
  TMR2ON = 1;
  TRISC2 = 0;
}

PWM1_Stop()
{
  CCP1M3 = 0;
  CCP1M2 = 0;

}

PWM2_Start()
{
  CCP2M3 = 1;
  CCP2M2 = 1;
  #if TMR2PRESCALE == 1
  T2CKPS0 = 0;
  T2CKPS1 = 0;
  #elif TMR2PRESCALE == 4
  T2CKPS0 = 1;
  T2CKPS1 = 0;
  #elif TMR2PRESCALE == 16
  T2CKPS0 = 1;
  T2CKPS1 = 1;
  #endif
  TMR2ON = 1;
  TRISC1 = 0;
}

PWM2_Stop()
{
  CCP2M3 = 0;
  CCP2M2 = 0;
}
```

4 BILL OF MATERIALS

Component	Price/Estimation	Recycled from e-junk
Microchip® PIC® 16F877A	Rs.350	
IC L293D (H-Bridge)	Rs.100	✓
IC LM358 (Dual OPAMP)	Rs.60	✓
IC 7805 (Voltage regulator)	Rs.15	
IR Tx/Rx pair	Rs.30/pair	
Oscillator 12MHz	Rs.10	✓
Chassis (Metal)	Took from a toy set	
BO-DC Motors x 2	Rs.100/motor	
Miscellaneous (Resistors, Capacitors, Buttons, LEDs, Pots, Jumper Headers)	Rs.70 est.	✓

We did the project by recycling e-junk components so that it not only reduced cost but also helped to provide a CLEAN ENVIRONMENT.

5 ESTIMATED DURATION OF PROJECT

	Week 1	Week 2	Week 3	Week 4	Week 5
Planning and Prerequisite setup					
Hardware implementation					
Software implementation					
Errata Checking					