

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

# 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.

# 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.

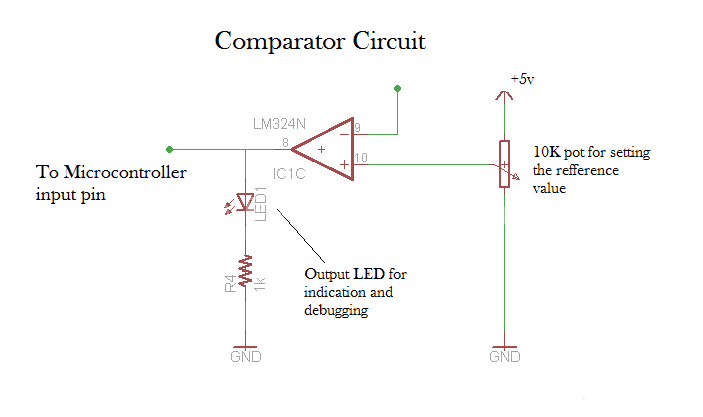
# Project Description

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.

## 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.

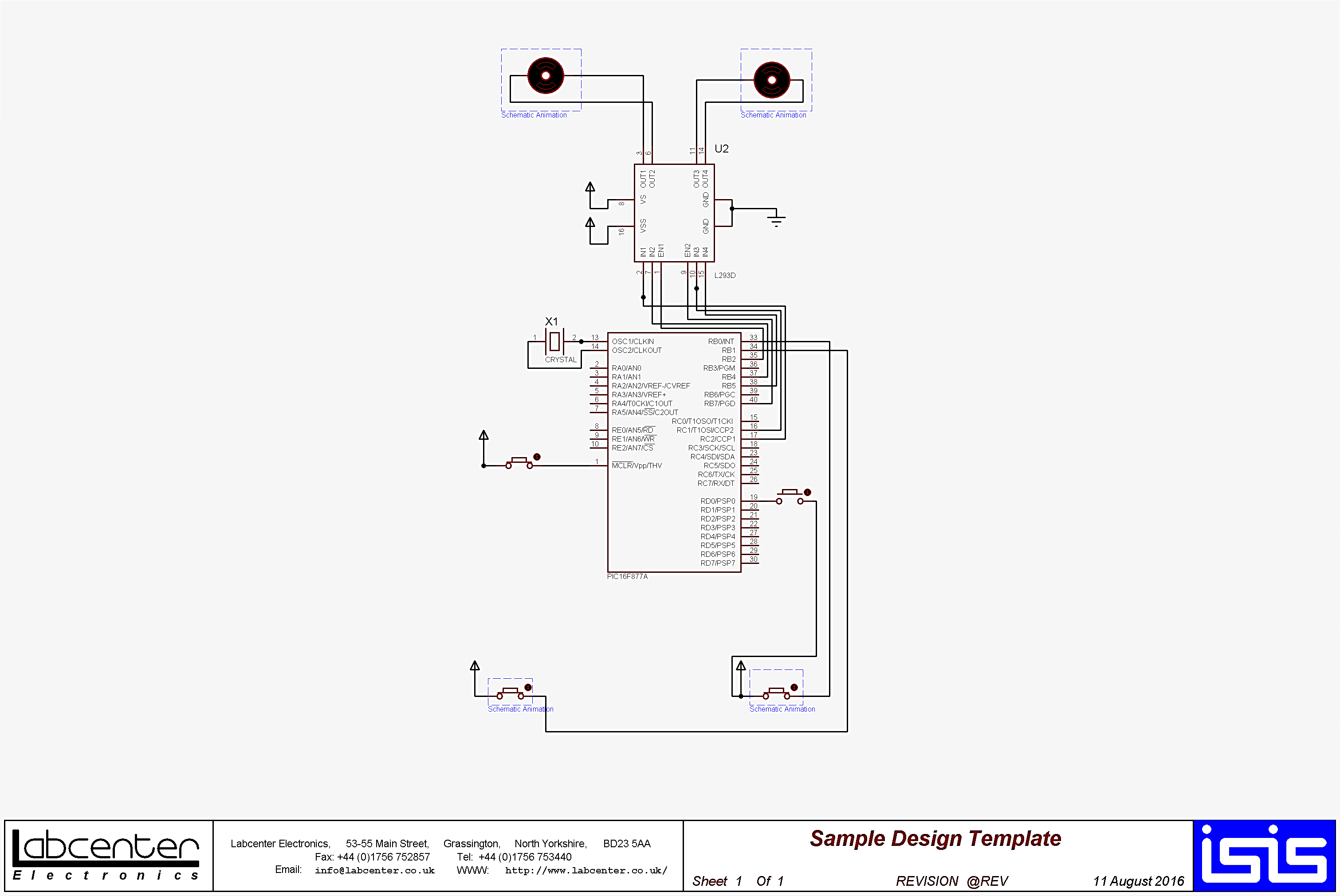
## Outputs

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.

## 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.



### Code

#### dumbFollower.c

1. #include "htc.h"
2. #define \_XTAL\_FREQ 12000000
3. #include "pwm.h"
5. #define EN1    RB2
6. #define C\_M\_R  RC2
7. #define CC\_M\_R RB4
8. #define CC\_M\_L RB5
9. #define C\_M\_L  RC1
10. #define EN2    RB7
12. #define S\_L    RB1
13. #define S\_R    RB0
14. \_\_CONFIG(HS & WDTDIS & PWRTEN & BOREN & LVPDIS & DUNPROT & WRTEN & UNPROTECT);
15. **void** main(**void**)
16. {
17. unsigned **int** d=500;
18. TRISB=0x03;
19. PORTB=0;
20. TRISC=0x00;
21. PORTC=0;
22. TRISD=0XFF;
23. PORTD=0;
25. PWM1\_Init(5000);
26. PWM2\_Init(5000);
27. PWM1\_Start();
28. PWM2\_Start();
30. **while**(1){
31. **if**(RD0==1){
32. **if**(d<1000){
33. **if**(d!=995)
34. {d+=5;}
35. **else**{d=50;}
36. }
37. \_\_delay\_ms(500);
38. }
40. **if**(S\_L==1&&S\_R==1){
41. CC\_M\_L=0;
42. CC\_M\_R=0;
43. EN1=1;
44. PWM1\_Duty(d);
45. EN2=1;
46. PWM2\_Duty(d);
47. }
48. **if**(S\_L==0&&S\_R==1){
49. CC\_M\_L=1;
50. CC\_M\_R=0;
51. EN1=1;
52. PWM1\_Duty(d+25);
53. EN2=1;
54. PWM2\_Duty(900-d);
55. }
56. **if**(S\_L==1&&S\_R==0){
57. CC\_M\_L=0;
58. CC\_M\_R=1;
59. EN1=1;
60. PWM1\_Duty(900-d);
61. EN2=1;
62. PWM2\_Duty(d+25);
63. }
64. **if**(S\_L==0&&S\_R==0){
65. CC\_M\_L=0;
66. CC\_M\_R=0;
67. EN1=0;
68. PWM1\_Duty(0);
69. EN2=0;
70. PWM2\_Duty(0);
71. }
72. }
73. }

#### pwm.h

1. #define TMR2PRESCALE 4
2. **long** freq;
4. **int** PWM\_Max\_Duty()
5. {
6. **return**(\_XTAL\_FREQ/(freq\*TMR2PRESCALE));
7. }
9. PWM1\_Init(**long** fre)
10. {
11. PR2 = (\_XTAL\_FREQ/(freq\*4\*TMR2PRESCALE)) - 1;
12. freq = fre;
13. }
15. PWM2\_Init(**long** fre)
16. {
17. PR2 = (\_XTAL\_FREQ/(freq\*4\*TMR2PRESCALE)) - 1;
18. freq = fre;
19. }
21. PWM1\_Duty(unsigned **int** duty)
22. {
23. **if**(duty<1024)
24. {
25. duty = ((**float**)duty/1023)\*PWM\_Max\_Duty();
26. CCP1X = duty & 2;
27. CCP1Y = duty & 1;
28. CCPR1L = duty>>2;
29. }
30. }
32. PWM2\_Duty(unsigned **int** duty)
33. {
34. **if**(duty<1024)
35. {
36. duty = ((**float**)duty/1023)\*PWM\_Max\_Duty();
37. CCP2X = duty & 2;
38. CCP2Y = duty & 1;
39. CCPR2L = duty>>2;
40. }
41. }
43. PWM1\_Start()
44. {
45. CCP1M3 = 1;
46. CCP1M2 = 1;
47. #if TMR2PRESCALE == 1
48. T2CKPS0 = 0;
49. T2CKPS1 = 0;
50. #elif TMR2PRESCALE == 4
51. T2CKPS0 = 1;
52. T2CKPS1 = 0;
53. #elif TMR2PRESCALE == 16
54. T2CKPS0 = 1;
55. T2CKPS1 = 1;
56. #endif
57. TMR2ON = 1;
58. TRISC2 = 0;
59. }
61. PWM1\_Stop()
62. {
63. CCP1M3 = 0;
64. CCP1M2 = 0;
66. }
68. PWM2\_Start()
69. {
70. CCP2M3 = 1;
71. CCP2M2 = 1;
72. #if TMR2PRESCALE == 1
73. T2CKPS0 = 0;
74. T2CKPS1 = 0;
75. #elif TMR2PRESCALE == 4
76. T2CKPS0 = 1;
77. T2CKPS1 = 0;
78. #elif TMR2PRESCALE == 16
79. T2CKPS0 = 1;
80. T2CKPS1 = 1;
81. #endif
82. TMR2ON = 1;
83. TRISC1 = 0;
84. }
86. PWM2\_Stop()
87. {
88. CCP2M3 = 0;
89. CCP2M2 = 0;
90. }

# 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.**

# Estimated Duration of Project

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