Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN 1093 – Laboratory Practice I

Line Following Robot

A.M.Thathsara	160626F
T.W.H.Tribuwan	160637N
N.L.Udugampola	160640R
U.G.D.C.T.Udupitiya	160641V

This is submitted as a partial fulfillment for the module
EN1093: Laboratory Practice I
Department of Electronic and Telecommunication Engineering
University of Moratuwa

05th January 2018

Abstract

PIC 16F877A controlled digital line follower was to be built in this project. It was to be made to follow a white line in a black background with a Y junction and a ramp with 15-degree inclination. Circuits were designed using Eagle. Robot was powered by a 2200mAh 45cc Li Po battery and it was built with custom made 7 TCRT5000 sensor modules in a bar type sensor panel. Coding was done using MPLab. L298N motor driver module was made to control 2 dc motors of 1200rpm and a stole current of 1.5A. PID based code was implemented to run the PIC IC. Final outcome was a fast-reliable successful line follower.

Contents

1.Introduction:

1.1 Description

Line following robot is basically a robot, which can follow a white or black line and complete certain task according to the need of the programmer or the user. They are mostly made from PIC microcontrollers and Atmel based Arduino chips. Here we had been given to make a line following robot using PIC micro-controller to follow a white line.

The task of the final competition consisted of challenging items. Talking of the electronics of the robot first thing to be mentioned is that it has been made with three main parts(PCBs). First one is the main PCB which consists of the PIC micro controller, comparators and a power supply. Next one is the motor controller, which governs the voltage needed for two motors and controls the two motors according the commands sent by the micro controller. Last one is the sensor panel which detects the white line and sends signals to the micro controller.

Apart from the electronic side of the robot we had to consider two other sections in this project. One was the programming part and it was basically coded with PID control. Other one was the physical nature of the robot, shape of the chassis and type of gear motors and wheels to be used. Here we have used a chassis matching the given task and one caster wheel is used for the front of the robot. Two motors are there only for controlling the back wheels and turning the robot is done by changing the speeds of two back wheels. PWM pins of the micro controller controls the speed of the two motors which are connected to back wheels of the robot.

1.2 Task

Robot that can follow a given path, which consist of sensor module to recognize the path, motors to drive it. At a junction path get divided into two, with 120-degree angle. One direction leads the robot to monster’s cave, while the other direction will lead to the goal. There is a white square on one side of the main path, in the same direction of the path leading to the dead end. Robot should decide the direction to turn, by detecting the square mark.

1.3 Robot Specifications

- Robot is made to follow a white line of 3cm, on a black background.
- Dimensions of the robot don’t exceed 250mm × 250mm× 250mm limit.
- Robot is made to climb a ramp with a 15-degree inclination.
- Robot has a specific sensor panel to detect a 20mm × 20mm square which is located with a 30mm clearance, and with that evidence robot should be able to identify the side of rotation at the

  ‘Y’ junction.
- The robot is powered by a 2200mAh 45cc Li-Po battery.
- It is completely autonomous.

****

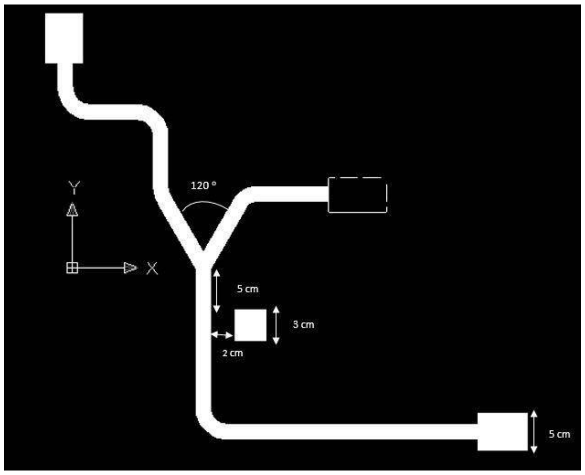


Fig 1: The arena draft, we have been given.

2.Method:

2.1 List of Components Used

2.1.1 Sensor Panel

It is a 16cm IR sensor panel with 7 parallel sensors.

- TCRT500 *7
- 330ohm *7
- 10k *7
- 10k Variable Resistors *7

2.1.2 Main PCB

Main PCB does handle the micro controller and comparators.

- 470ohm *2
- 10k *1
- 0.1u *2 Capacitors
- 22p *2 Capacitors
- 1N4004 *2
- 6A10 diode *1
- LED *1
- 7805 Voltage Regulator
- Crystal oscillator 16MHz
- PIC16f877A
- LM324 *2

2.1.3 Motor Controller

L298 motor controller module was made

- L298 Motor Drive
- 0.1u *2
- 10k *2
- 1N4004 *8

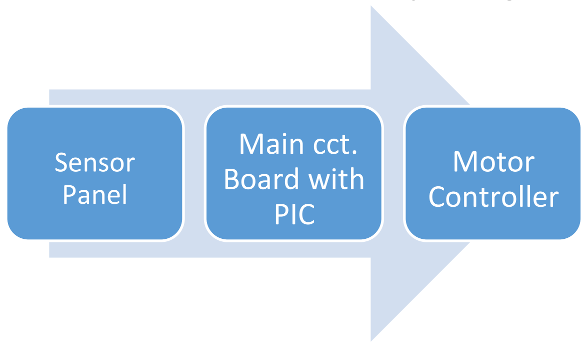
2.1.4 Alternative Power Supply Circuit

Two power 5V supply units with power resistors was made

- 470ohm *7
- LED *7
- 7805 Voltage Regulator
- 0.1u *2
- Fuse *1
- 6A10 diode *1
- 47ohm 1W *2

2.2 Experimental Design Procedure

This task is achieved mainly through following three steps.



Ch 1: How the design procedure has been done?

Main Circuit Board

This mainly comprises of a PIC micro controller, power supply(Power regulator) and comparators. PIC micro controller is there to take signals from the sensor panel and execute appropriate commands to govern the motor controller accordingly. This micro controller should be programmed before placing on the main circuit board. We first used Micro C for programming, but due to malfunctioning of the code we shifted to MPLAB. Finally, we have done this project with that software. RB3, RB2, RB1, RB0, RD7, RD6, RD5 are the pins of the micro controller which are connected to the sensor panel outputs. (That means pins which are connected to the photo transistor terminals) RD0, RD1, RD2, RD3 are the pins which are connected to the logic inputs of the motor controller. RD0 and RD1 are connected to control the logic of the left motor while other two pins are there to control the logic of the right motor.RC1 and RC2 are the pins which are considering as PWM pins. RC1 is set to control the speed of rotation of the left motor and RC2 is set to control speed of rotation of the right motor. Basically, the program of the micro controller has been done using PID control.

When we consider the power supply of the robot, we use two power supplies, since there was big heat dissipation through the power regulator. (This happened since we have used 7 sensors big current is transmitting through the regulator.) The regulator which we have used is 7805. We powered up the sensor panel using another power supply and this drastically reduced the heat dissipation in the main power supply. Other power supply is attached separately to the robot.

Comparators are set here to compare the logic states of the sensors and output the correct logic with certainty.

****

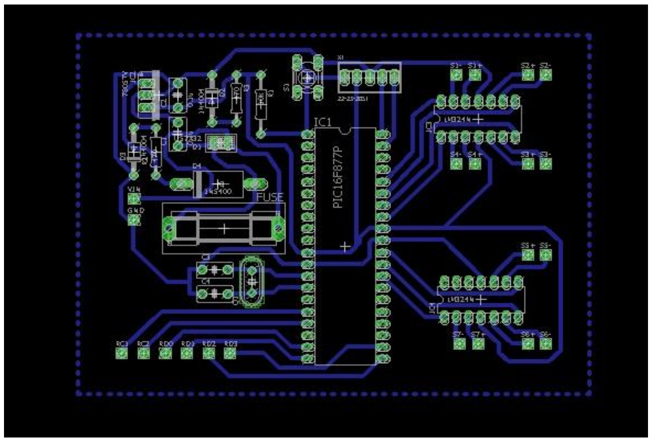


Fig 2: Layout of the Main PCB

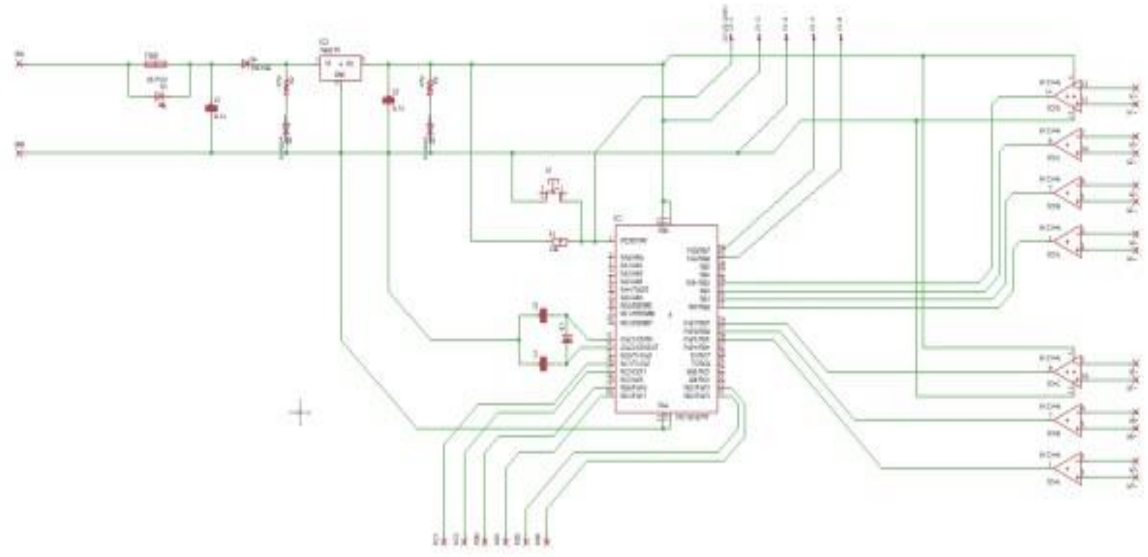


Fig 3: Schematic of the Main PCB

Sensor Panel

Sensor panel consists of 7 TCRT5000 sensors. One TCRT5000 sensor consists of one photo transistor and a IR emitter. Every TCRT couple is attached with a variable resistor to set a threshold value in order to have expected output. We have designed this sensor panel to achieve the given task. Middle three sensors are placed 0.75mm apart from each other. Another two couples have been set 2.75mm apart from the center of the sensor panel. Remaining two are set 4.75mm apart from the center of the sensor panel. Middle three sensors are the sensors which are set to identify the white line and track it accordingly. Two sensors which are 2.75mm apart from the center of the sensor panel are set to identify the white square. Other two sensors are set also to identify the square if this robot does not correctly track the white line (That means if the robot is unable to track the white line in a linear manner)

****

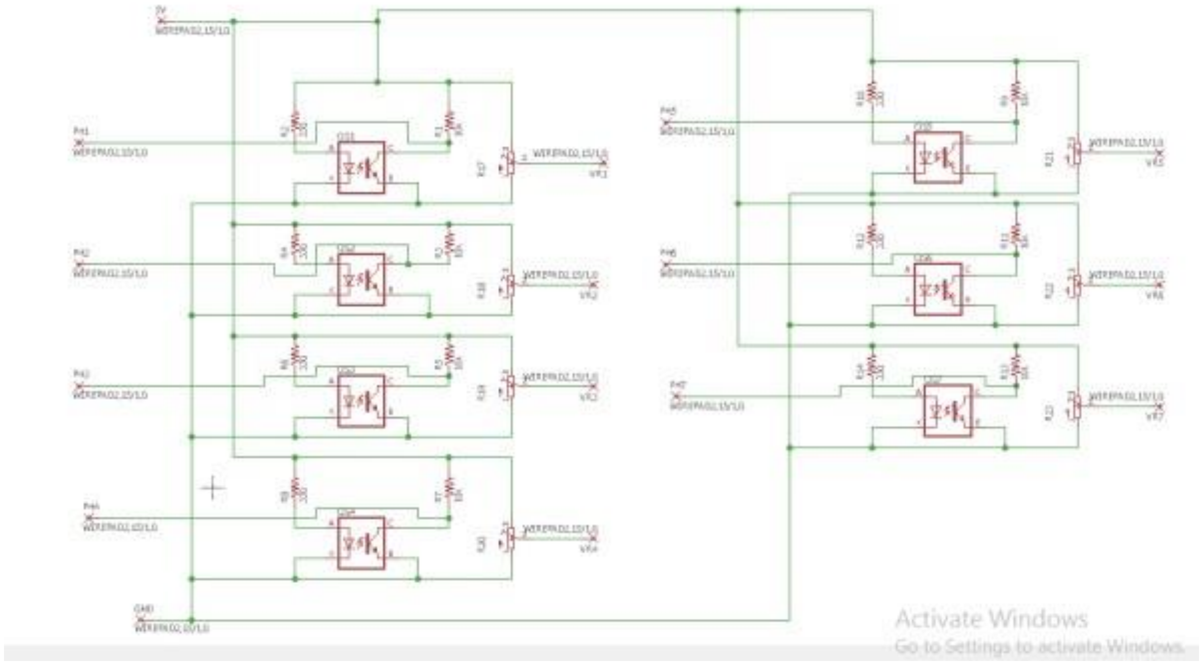


Fig 4: Sensor Panel Schematic

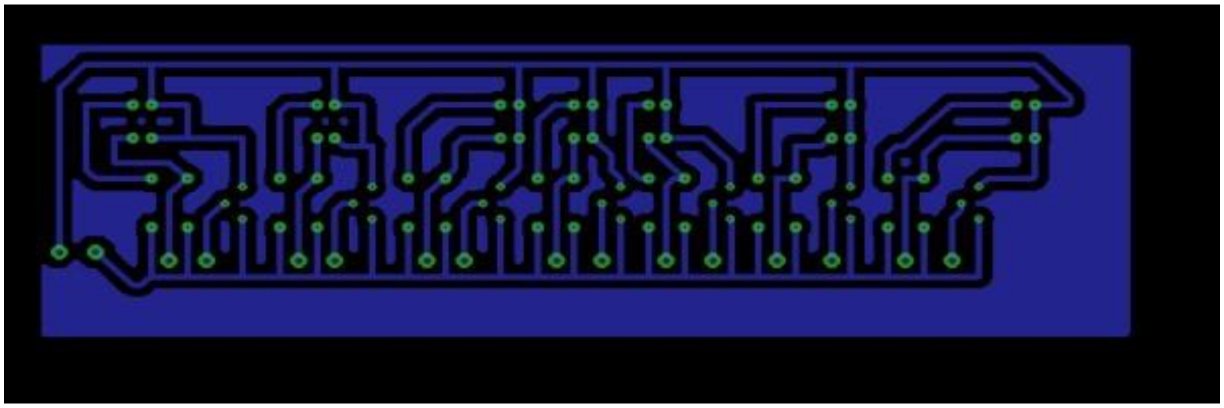


Fig 5: Sensor Panel Layout

Motor Controller

This is the device which controls the side of rotation of the motor and the speed of rotation of the motor. Motor controller which we have used here is L298. Basically, this controller needs to be powered up (We have been powered up using a 12V LiPo battery) and this has four logic input terminals, which helps to control the side of rotation of the motors and two PWM input terminals, which helps to control the speed of rotation of the motors. And there are four output terminals which are connected to two motors. These output terminals should be connected with the attached diodes in order to control the effect from back EMF.

This is the basic structure of the motor controller and what we had to consider much was the heat dissipation which occurs through this L298 device.

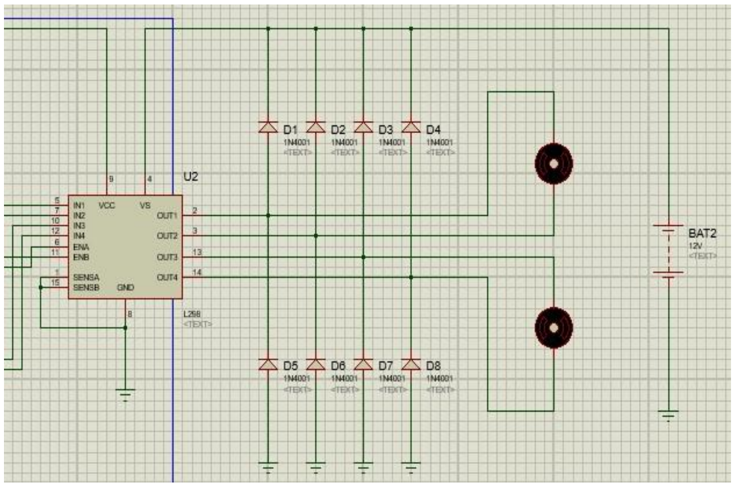


Fig 6: This represents the way that diodes are set to the motor controller

2nd Power Supply

Due to large heat dissipation through the power supply we have decided to include another power supply in order to safe guard the 7805 regulators. This was happened due to having 7 sensors and much larger current got absorbed by the sensor panel. So that this 2nd power supply was designed in order to dedicate power to the sensor panel. We have included 7 LEDs for that to indicate which sensors are detecting the white surfaces while running. This is placed at the forfront of the robot.

****

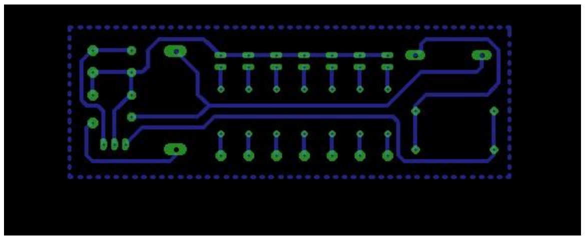


Fig 7: 2nd Power Supply Layout

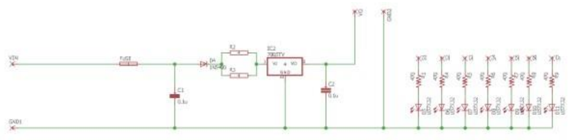


Fig 8: 2nd Power Supply Schematic

3. Basic Other Aspects We have Considered in Designing this Robot

3.1 Motors

We have been selected two motors which have comparatively high RPMs. Since that they have comparatively low torque. Since this is a competition we have been selected this high RPMs. Even we had think low torque will obstruct the robot's behavior it had not affect seriously since we have used a high duty to initialize the PWM in micro controller. Maximum voltages of these motors are 12V.

3.2 Chassis

Chassis has been selected according to the need of the robot that matches for the tasks, and we have been modified that according to our need. (In order to set PCBs, connect the Caster wheel etc.)

3.3 Coding (Program of the Robot)

First we have been coded using Micro C and due to malfunctioning of the robot with that software we have been shifted to MPLAB. Before we set the code in order to complete the task we made few codes to check whether PWM is working and so on. The code is basically done using PID controlling. Since we have been used 7 sensors first we were given 7 weights for those. Initially we have been given -3 to +3 values and then onwards we changed those weights when we are tuning the robot. Finally we have been ended up with weights -9, -5, -2, 0, 2, 5 and 9 respectively from the left corner sensor couple. Then we are calculating status of the robot by using these values. And also error (How much does the robot away from the line) and the derivative (Error – Last Error) are also calculating. Then speeds of the robot are changing according to these values. Instead of that PID we have been included some specialities in the code. One thing is by detecting the side where the robot went out it could turn back to the line and if the robot went straight out of the line (That means the end of the arena) then to stop. And one another thing what we have included was the algorithm which has to be run after detecting the square. That algorithm will run only about 30ms (By using a 'for' loop). In that case algorithm was basically forgetting the sensors at the side of the sensor panel for about 30ms until it turns from the "y" junction.

4. Results

4.1 Troubles What we have Faced and How did we Overcome Those?

- At the first time when we are manufacturing PCBs we have been used Protius for all our simulations and designing layout purposes. However after manufacturing PCBs they haven't worked. When we are analyzing the case what we found was, two connections that should exist within 4 PIC pins are not present in our layout. It cannot be actually set using manual routing. Since this case we have been decided not use Protius software again and proceed with another software. The software what we have selected was Eagle.
- At the initial stage due to lack of experience of our group members soldering made huge difficulties like burning of ICs etc. Due to these soldering problems we had to made PCBs again.
- At the later part of the project even we were done with the hardware part of the robot it didn't work with testing codes. What we found as reason was, copper boards what we have used were low quality, since that some copper tracks were blasted. Then we had to solder them by finding each and every connection.
- The grip of the wheels were not enough to grab the road thoroughly. Since that we were set the battery at the back of the robot like a spoiler in order to increase the weight which comes on to the two back wheels.
- Lots of heat dissipation was took place from the main PCB voltage regulator. This has been happened due to big current drawn through the sensor panel since there are 7 sensor couples are there. So in order to minimize the heat dissipation through that regulator, we made another power supply for sensor panel and the motor controller. In order to reduce the heat dissipation in that regulator we were set two 1W resistors before the regulator in order to drop the voltage up to about 9V before it passes to the regulator.

5.Conclusion

With many failures arose while going through the project group members were able to learn the optimum methods and the components needed to make a successful line following robot. Mainly regarding power supplying considering circuit current, handling IR sensors with comparators and microcontroller programming were the most significant learning outcomes of the project. Following the above mentioned methods we were finally able to make a solid robot which was easily programmable using MPLab and with a PID based coding robot was fully functional and was able to complete all the tasks given in the competition.

6. Appendix

Code

```
/*
* File:   newmain.c
* Author: Nalith
 *
* Created on December 17, 2017, 2:02 AM
 */
#define _XTAL_FREQ 16000000
#define TMR2PRESCALE 4
#include "Config_Bits.h"
#include <xc.h>

long freq;

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

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

PWM2_Init(long fre)
{
  PR2 = (_XTAL_FREQ/(fre*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 TMR2PRESCALAR == 1
    T2CKPS0 = 0;
    T2CKPS1 = 0;
  #elif TMR2PRESCALAR == 4
    T2CKPS0 = 1;
    T2CKPS1 = 0;
  #elif TMR2PRESCALAR == 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;
}

void main()
{

   TRISC=0;
   TRISD=0b11110000;
   TRISB=0b00001111;

   PWM1_Init(5000);
   PWM2_Init(5000);

   RD0=1;
   RD1=0;
   RD2=1;
   RD3=0;

   PWM1_Start();
   PWM2_Start();



   float position, error, lasterror, adjustment;
   float Kp=140;    float
Kd=350;    float
Lspeed, Rspeed;
   int n=0;  // n is to find the number of sensors on the line    int sum=0;
//sum is to get the sum of positional values of those sensors    int
LineOut=1; //To indicate whether the robot is on line or not    int
LastOut; //To find from which side, the robot went out of the line;    int
square;

  while(1){
    LineOut=0;
    Lspeed=720;
    Rspeed=720;

    n=0;
sum=0;

    if(RB3==0){
n=n+1;
LineOut=1;
sum=sum-9;
        LastOut=-1;
    }
    if(RB2==0){
n=n+1;
LineOut=1;
sum=sum-5;
        LastOut=-1;
    }
    if(RB1==0){
n=n+1;
LineOut=1;
sum=sum-3;
LastOut=0;
    }
    if(RB0==0){
n=n+1;
LineOut=1;
sum=sum+0;
LastOut=0;
    }
    if(RD7==0){
n=n+1;
LineOut=1;
sum=sum+3;
LastOut=0;
    }
    if(RD6==0){
n=n+1; LineOut=1;
sum=sum+5;
        LastOut=1;
    }
    if(RD5==0){
n=n+1;
LineOut=1;
sum=sum+9;
        LastOut=1;
    }

    if(n!=0){
position=sum/n;
    error=0-position;
    }

    adjustment = Kp*error + Kd*(error - lasterror);
    lasterror=error;
if(adjustment>0){
Lspeed=Lspeed-adjustment;
      Rspeed=Rspeed+adjustment/6;
    }
    else{
        Rspeed=Rspeed+adjustment;
        Lspeed=Lspeed-adjustment/6;
    }
    /*Rspeed=Rspeed+adjustment;
    Lspeed=Lspeed-adjustment;
     */

    /*Marginal Speed Limitations*/
if(Rspeed>1023){
        Rspeed=1023;
    }
    if(Lspeed>1023){
        Lspeed=1023;
    }
```

```
     if(Rspeed<0){
RD0=0;
        RD1=1;
        Rspeed=-Rspeed;

    }
    if(Lspeed<0){
RD2=0;
        RD3=1;
        Lspeed=-Lspeed;

    }



    /*Follow when robot goes out of line*/
while(LineOut==0){         if(LastOut==(-
1)){
            RD2=0;
            RD3=1;
            Lspeed=500;
            Rspeed=750;
        }
        if(LastOut==1){
            RD0=0;
            RD1=1;
            Lspeed=750;
            Rspeed=500;
        }
        if(LastOut==0){
            RD0=0;
            RD1=0;
            RD2=0;
            RD3=0;
        }

        PWM1_Duty(Lspeed);
        PWM2_Duty(Rspeed);
        __delay_ms(20);

         if((RB3==0)||(RB2==0)||(RB1==0)||(RB0==0)||(RD7==0)||(RD6==0)||(RD5==0)){
            LineOut=1;
        }
    }

    /*Y junction turn*/

    if((RB0==0)&&(RB3==0)){
        square=-1;



        int i;
        for(i=0;i<20;i++){
Lspeed=700;
Rspeed=700;
n=0;            sum=0;
if(RB0==0){
n=n+1;
sum=sum+0;
                LineOut=1;
            }
            if(RD7==0){
n=n+1;
sum=sum+3;

}
            if(RD6==0){
n=n+1;
sum=sum+5;
                LineOut=1;

            }
            if(RD5==0){
n=n+1;
sum=sum+9;
                LineOut=1;

            }
             if(n!=0){
position=sum/n;
error=0-position;
}
             LastOut=-1;

            adjustment = Kp*error + Kd*(error - lasterror);
            lasterror=error;
            Rspeed=Rspeed+adjustment;
Lspeed=Lspeed-adjustment/6;             if(Rspeed>1023){
                Rspeed=1023;
             }
            if(Lspeed>1023){
                Lspeed=1023;
            }
             if(Rspeed<0){
RD0=0;
                 RD1=1;
                 Rspeed=-Rspeed/2.5;

            }
            if(Lspeed<0){
RD2=0;
                RD3=1;
                Lspeed=-Lspeed/2.5;

            }

            PWM1_Duty(Lspeed);
            PWM2_Duty(Rspeed);
            __delay_ms(20);

            RD0=1;
            RD1=0;
            RD2=1;
            RD3=0;

            }
   }

    if((RB0==0)&&(RD5==0)){
        square=1;

int i;
        for(i=0;i<20;i++){
Lspeed=700;
Rspeed=700;             n=0;
            sum=0;
            if(RB3==0){
n=n+1;                  sum=sum-
9;
                LineOut=1;
            }
            if(RB2==0){
n=n+1;                  sum=sum-
5;
                LineOut=1;
            }
            if(RB1==0){
n=n+1;                  sum=sum-
3;
                LineOut=1;
            }
            if(RB0==0){
n=n+1;
sum=sum+0;
            }
if(n!=0){
position=sum/n;
            error=0-position;
            }
            LastOut=1;
            adjustment = Kp*error + Kd*(error - lasterror);
            lasterror=error;
            Lspeed=Lspeed-adjustment;
Rspeed=Rspeed+adjustment/5;              if(Rspeed>1023){
                Rspeed=1023;
             }
            if(Lspeed>1023){
                Lspeed=1023;
            }
             if(Rspeed<0){
RD0=0;
                 RD1=1;
                 Rspeed=-Rspeed/2.5;

            }
            if(Lspeed<0){
RD2=0;
                RD3=1;
                Lspeed=-Lspeed/2.5;
            }

            PWM1_Duty(Lspeed);
            PWM2_Duty(Rspeed);
            __delay_ms(20);

            RD0=1;
            RD1=0;
            RD2=1;
            RD3=0;
          }
        }

      /*
        if((square==-1)&&(RB2==0)&&(RD6==0)){
          RD0=1;
          RD1=0;
          RD2=1;
          RD3=0;
          PWM1_Duty(0);
          PWM2_Duty(1023);
          __delay_ms(250);

        }

        if((square==1)&&(RB2==0)&&(RD6==0)){        RD0=1;
          RD1=0;
          RD2=1;
          RD3=0;
          PWM1_Duty();
          PWM2_Duty(0 );
          __delay_ms(250);

        }
        */

        PWM1_Duty(Lspeed);
        PWM2_Duty(Rspeed);
        __delay_ms(20);

        RD0=1;
```

```
    RD1=0;
    RD2=1;
    RD3=0;


    }

}
```