// PIC16F877A Configuration Bit Settings

// 'C' source line config statements

// CONFIG

#pragma config FOSC = XT // Oscillator Selection bits (XT oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR disabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define \_XTAL\_FREQ 8000000

#define TMR2PRESCALE 4

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 move(int(R),int(L)){

PWM1\_Init(5000);

PWM2\_Init(5000);

//TRISD = 0xFF;

//TRISB = 0;

PWM1\_Duty(R);// Enable pin of right side

PWM2\_Duty(L);//enable pin of left side

//CMCON=0X07;

ADCON1=0b00001111;

//TRISA &=0x00;

PORTA=0b00000101;

PWM1\_Start();

PWM2\_Start();

\_\_delay\_ms(30);

PWM1\_Stop();

PWM2\_Stop();

}

float ultrasoundR(void){

float a;

float distance;

/\*TRISB4=0;

TRISB5=1;

TRISC5=0;\*/

T1CON = 0x10;

while(1)

{

TMR1H = 0;

TMR1L = 0;

RB4=1;

\_\_delay\_us(10);

RB4=0;

while(!RB5);

TMR1ON=1;

while(RB5);

TMR1ON=0;

a=(TMR1L|(TMR1H<<8));

a=a/58.82;

a=a+1;

distance=a/2;

return distance;

}

}

float ultrasoundL(void){

float a;

float distance;

/\*TRISB6=0;

TRISB7=1;

TRISC5=0;\*/

T1CON = 0x10;

while(1)

{

TMR1H = 0;

TMR1L = 0;

RB6=1;

\_\_delay\_us(10);

RB6=0;

while(!RB7);

TMR1ON=1;

while(RB7);

TMR1ON=0;

a=(TMR1L|(TMR1H<<8));

a=a/58.82;

a=a+1;

distance=a/2;

return distance;

}

}

void main() {

TRISC=0;

TRISD=0b11110000;

TRISB=0b10101111;

int Rspeed,Lspeed;

PWM1\_Init(5000);

PWM2\_Init(5000);

RD0 = 1 ;

RD1 = 0;

RD2 = 1;

RD3 = 0;

PWM1\_Start();

PWM2\_Start();

PWM1\_Duty(870);

PWM2\_Duty(870);

RD0 =1 ;

RD1 =0;

RD2 =1;

RD3 = 0;

\_\_delay\_ms(900);

while(1){

Rspeed = 963;

Lspeed = 850;

if (RB3==0){

Rspeed+=60;

Lspeed-=370;

}

if (RD5==0){

Rspeed-=370;

Lspeed+=60;

}

if (RB1==0){

Rspeed+=20;

Lspeed-=180;

}

if (RB2==0){

Rspeed+=34;

Lspeed-=250;

}

if (RD6==0){

Rspeed-=250;

Lspeed+=34;

}

if(RB0 ==0){

Rspeed = Rspeed;

Lspeed = Lspeed;

}

if(RD7==0){

Rspeed-=180;

Lspeed+=20;

}

PWM1\_Start();

PWM2\_Start();

PWM1\_Duty(Rspeed);

PWM2\_Duty(Lspeed);

\_\_delay\_ms(20);

PWM1\_Stop();

PWM2\_Stop();

if ((RB0 == 0) && (RB1==0)&&(RB2==0)&&(RD7==0)&&(RD6==0)&&(RB3==0)&&(RD5==0)){

PWM1\_Duty(0);

PWM2\_Duty(0);

RD0 = 0;

RD1 = 0;

RD2 = 0;

RD3 = 0;

break;

}

}

ADCON1=11111111; // for wall

float dist\_r\_wall;

float dist\_l\_wall;

float error;

float errorL;

float R\_pwm;

float L\_pwm;

float bend\_error;

float error\_rate;

float p\_error;

p\_error=0;

while(1){

dist\_r\_wall=ultrasoundR();

dist\_l\_wall=ultrasoundL();

error=dist\_r\_wall-dist\_l\_wall;

error\_rate=error-p\_error;

if (error>4){

error=4;

}

else if(error<-4){

error=-4;

}

if(error!=0){

R\_pwm=845-25\*error-10\*error\_rate;

L\_pwm=890+25\*error+10\*error\_rate;

}

else if(error=0){

R\_pwm=845;

L\_pwm=890;

}

move(R\_pwm,L\_pwm);

p\_error=error;

}

return;

}