//all four us sensors

/\* March 11 2:25am : This program runs 4 ultrasonic sensors round robin. tested with 4 but RB4 has no signal. The trigger for RB4 (RB2) is blinking as expected.

\* (tested with 3 and confirmed in multisensor.c. Reminder: run with KPD disconnected.\*/

#include <xc.h>

#include <stdio.h>

#include <stdbool.h>

#include "configBits.h"

#include "RTC.h"

#include "I2C.h"

#include "lcd.h"

#include "actuators.h"

#include "timer.h"

#define us\_delay 150

#define thrdist 30

int timer\_count = 0;

int cnt =0;

int a[4];

int prev\_dist\_final[4];

int b;

void echo();

volatile int sens;

void print\_echo();

const char keys[] = "123A456B789C\*0#D";

unsigned char temp;

short int temp\_int, tick;

volatile bool key\_was\_pressed = false;

int state = 0;

int disp = 0;

int i = 0;

int pressed = 0;

//I2C

unsigned char counter = 0; // Increments each time a byte is sent

unsigned char keypress; // Stores the data corresponding to the last key press

unsigned char data; // Holds the data to be sent/received

int dists[2] = {0,-1};

bool send = true; //PIC is sending

int tires\_deployed;

int poles\_detected;

volatile long avg\_dist;

//void start\_dc();

void start();

void brake();

void sense\_tires(int sensed);

typedef struct Poles{

int tires\_deployed; //check var

int tires\_final;//check var

int dist\_from\_start;//check var

int dist\_from\_cl;

} Poles;

struct Poles Pole[10] = {0,0,0,0,0,0,0,0,0,0,}; //declare pole array

int dist\_final[4][6] = {

{0, 0, 0, 0, 0,0}, // dist\_final[0]

{0, 0, 0, 0, 0,0}, // dist\_final[1]

{0, 0, 0, 0, 0,0}, // dist\_final[2]

{0, 0, 0, 0, 0,0}, // dist\_final[3]

};

int sum[4];

bool sumf[4] = {false,false,false,false};

volatile int meas =0;

void \_\_interrupt() ISR(){

if(RBIF == 1){ //Makes sure that it is PORTB On-Change Interrupt

if (sens){

RBIE = 0; //Disable On-Change Interrupt

echo();

RBIF = 0; //Clear PORTB On-Change Interrupt flag

RBIE = 1; //Enable PORTB On-Change Interrupt

}

if (~sens){

key\_was\_pressed = true;

INT1IF = 0; // Clear interrupt flag bit to signify it's been handled

}

}

if (TMR0IF){

INTCONbits.TMR0IF = 0; //CLEAR FLAG SO IT CAN BE TRIGGERED AGAIN

TMR0H = TIMER0\_COUNTS\_PER\_HALF\_SECOND >> 8;

TMR0L = TIMER0\_COUNTS\_PER\_HALF\_SECOND;

cnt++;

LEDPin ^= 1u; //Toggle LED

// lcd\_clear();

// printf("time");

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// if (cnt%2 == 0){

// printf("%d",cnt/2);

// }

}

}

void echo(){

if (b==1){

if (RB7 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB7 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[0] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

// a[0] = (TMR1L | (TMR1H<<8))\*0.034/2;

}

//Moving average

if (a[0]>=2 && a[0]<=200){

if ((~sumf[0])&&(dist\_final[0][0]!=0)&&(dist\_final[0][1]!=0)&&(dist\_final[0][2]!=0)){ //if haven't summed before

sum[0] = (dist\_final[0][0]+dist\_final[0][1]+dist\_final[0][2]);

sumf[0] = true;

}

if (sumf[0]){

sum[0] = sum[0] - dist\_final[0][meas] + a[0];

dist\_final[0][5] = (sum[0]/5);

}

dist\_final[0][meas] = a[0];

}

}

if (b==2){

if (RB6 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB6 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[1] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

// a[1] = (TMR1L | (TMR1H<<8))\*0.034/2;

}

if (a[1]>=2 && a[1]<=200){

if ((~sumf[1])&&(dist\_final[1][0]!=0)&&(dist\_final[1][1]!=0)&&(dist\_final[1][2]!=0)){ //if haven't summed before

sum[1] = (dist\_final[1][0]+dist\_final[1][1]+dist\_final[1][2]);

sumf[1] = true;

}

if (sumf[1]){

sum[1] = sum[1] - dist\_final[1][meas] + a[1];

dist\_final[1][5] = (sum[1]/5);

}

dist\_final[1][meas] = a[1];

}

}

if (b==3){

if (RB5 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB5 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[2] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

// a[2] = (TMR1L | (TMR1H<<8))\*0.034/2;

}

if (a[2]>=2 && a[2]<=200){

if ((~sumf[2])&&(dist\_final[2][0]!=0)&&(dist\_final[2][1]!=0)&&(dist\_final[2][2]!=0)){ //if haven't summed before

sum[2] = (dist\_final[2][0]+dist\_final[2][1]+dist\_final[2][2]);

sumf[2] = true;

}

if (sumf[2]){

sum[2] = sum[2] - dist\_final[2][meas] + a[2];

dist\_final[2][5] = (sum[2]/5);

}

dist\_final[2][meas] = a[2];

}

}

if (b==4){

if (RB4 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB4 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[3] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

// a[3] = (TMR1L | (TMR1H<<8))\*0.034/2;

}

if (a[3]>=2 && a[3]<=200){

if ((~sumf[3])&&(dist\_final[3][0]!=0)&&(dist\_final[3][1]!=0)&&(dist\_final[3][2]!=0)){ //if haven't summed before

sum[3] = (dist\_final[3][0]+dist\_final[3][1]+dist\_final[3][2]);

sumf[3] = true;

}

if (sumf[3]){

sum[3] = sum[3] - dist\_final[3][meas] + a[3];

dist\_final[3][5] = (sum[3]/5);

}

dist\_final[3][meas] = a[3];

}

}

}

void trig(){

for(b=1; b<5; b++){

TMR1H = 0; //Sets the Initial Value of Timer

TMR1L = 0; //Sets the Initial Value of Timer

if(b==1){

RBIE = 0;

RD1 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RD1 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==2){

RBIE = 0;

RD0 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RD0 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==3){

RBIE = 0;

RB3 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RB3 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==4){

RBIE = 0;

RB2 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RB2 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

}

}

/\*void print\_echo(){

lcd\_clear();

if(a[0]>=2 && a[0]<=400){ //Check whether the result is valid or not

printf("Distance 1 = %d",a[0]);

}

if(!(a[0]>=2 && a[0]<=400)){

printf("Distance 1 = X");

}

if(a[1]>=2 && a[1]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("Distance 2 = %d",a[1]);

}

if(!(a[1]>=2 && a[1]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("Distance 2 = X");

}

if(a[2]>=2 && a[2]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3= %d",a[2]);

}

if(!(a[2]>=2 && a[2]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3 = X");

}

if(a[3]>=2 && a[3]<=800){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4= %d",a[3]);

}

if(!(a[3]>=2 && a[3]<=800)){

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4 = X");

}

}\*/

void print\_echo(){

lcd\_clear();

if(a[0]>=2 && a[0]<=400){ //Check whether the result is valid or not

printf("Dist1=%d,Avg=%d ",a[0],dist\_final[0][5]);

}

if(!(a[0]>=2 && a[0]<=400)){

printf("Dist1 = X");

}

if(a[1]>=2 && a[1]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("Dist2=%d,Avg=%d",a[1],dist\_final[1][5]);

// printf("Dist 2 = %d",a[1]);

}

if(!(a[1]>=2 && a[1]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("Dist2 = X");

}

if(a[2]>=2 && a[2]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Dist3=%d,Avg=%d",a[2],dist\_final[2][5]);

// printf("Dist 3= %d",a[2]);

}

if(!(a[2]>=2 && a[2]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Dist3 = X");

}

if(a[3]>=2 && a[3]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Dist4=%d,Avg=%d ",a[3],dist\_final[3][5]);

// printf("Dist 4= %d",a[3]);

}

if(!(a[3]>=2 && a[3]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Dist4 = X");

}

}

void ultrasonic\_main(int actuator){

TRISB = 0b11110000; //RB5 6 as Input PIN (ECHO)

RBIF = 0; //Clear PORTB On-Change Interrupt Flag

RBIE = 1; //Enable PORTB On-Change Interrupt

TRISBbits.RB0 = 0;

LATBbits.LATB0 = 1; //DISABLES KPD

int sensed = 0;

int flag =0;

while(send){

for (meas=0; meas<3;meas++){ //3 measurements in the moving average array

trig(); //triggering all of them

RBIE = 0;

print\_echo();

// lcd\_clear();

// printf("wha");

// \_\_delay\_ms(1000);

RBIE = 1;

sensed++;

if (~actuator){

sense\_tires(sensed);

}

if ((actuator)&&(sensed==2)){

/////stops here///////// wtf

flag = 1;

break;

// return;

}

if (flag){

break;

}

}

if (flag){

break;

}

// if (~actuator){

// sense\_tires(sensed);

// }

// if ((actuator)&&(sensed==2)){

// lcd\_clear();

// printf("BREAK");

// \_\_delay\_ms(1000);

// break;

// }

}

while (~send){ //gets data from arduino

//Read Arduino's encoder data

I2C\_Master\_Start();

I2C\_Master\_Write(0b00010001); // 7-bit Arduino slave address + Read

avg\_dist = I2C\_Master\_Read(NACK); // Read one char only

I2C\_Master\_Stop();

if(avg\_dist){

// while(1){

lcd\_clear();

// if (dists[0] != 0 && dists[1] == -1){

// dists[1] = data;

// }

// if (dists[0] == 0){

// dists[0] = data;

// }

printf("%d",avg\_dist); //puts character on LCD

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// printf("Motor dist1 %d",dists[0]);

// lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

// printf("Motor dist2 %d",dists[1]);

// \_\_delay\_ms(5000);

// }

send = true; //return to PIC as sender

}

else{

send = false;

break;

}

}

}

void sense\_tires(int sensed){

// \_\_delay\_ms(1000);

// if ((a[0]>=2 && a[0] <= 15) ||(a[1]>=2 && a[1] <= 15)||(a[2]>=2 && a[2] <= 15)||(a[3] >=2 && a[3]<= 15)){

// \_\_delay\_ms(300); //delay to allow sensors to read more than just one reading

if (((sensed>2)&&(poles\_detected>0))||(poles\_detected==0)){ //this algorithm may need to be changed////////////////////////////////

if ((dist\_final[0][5]>=2 && dist\_final[0][5] <= thrdist) ||(dist\_final[1][5]>=2 && dist\_final[1][5] <= thrdist)||(dist\_final[2][5]>=2 && dist\_final[2][5] <= thrdist)||(dist\_final[3][5] >=2 && dist\_final[3][5]<= thrdist)){

//Tell Arduino to stop motors by writing 9

brake();

//Turn PIC into receiver mode

send = false;

// start\_dc();

// lcd\_clear();

// printf("tiemr %d",cnt);

// \_\_delay\_ms(1000);

}

}

// }

// \_\_delay\_ms(400);

// }

// \_\_delay\_ms(500);

/\*if ((a[0]>=2 && a[0] <= 15) ||(a[1]>=2 && a[1] <= 15)||(a[2]>=2 && a[2] <= 15)||(a[3] >=2 && a[3]<= 15)){

//Tell Arduino to stop motors by writing 9

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Write('9'); // Write key press data which shows up on Arduino's serial monitor

I2C\_Master\_Stop();

//Turn PIC into receiver mode

send = false;

}

// \_\_delay\_ms(400);

}\*/

}

int number\_deploy(int avg\_dist, poles\_detected){

int tires\_t=0;

int tires\_detected=0;

int t\_count = 0;

// while (DC motors off){

ultrasonic\_main(1);

// \_\_delay\_ms(1000);

float avg\_dist\_cm = (avg\_dist/1000);

while (t\_count<3){

if (poles\_detected != 0){

if ((avg\_dist\_cm)<30){

//Total number of tires tires\_t

tires\_t = 1;

}

if ((avg\_dist\_cm)>30){

tires\_t = 2;

}

}

if (poles\_detected == 0){

tires\_t = 2;

}

if ((a[0]>=2 && a[0] <= 15)){

// sensl\_1 = true; //lower sensor is high

if ((a[1]>=2 && a[1] <= 15)){

// sensl\_2 = true; //

tires\_detected = 1;

}

else{

tires\_detected = 0;

}

}

if ((tires\_detected==1)){

if((a[2]>=2 && a[2] <= 15)&&(a[3] >=2 && a[3]<= 15)){

tires\_detected++;

}

}

t\_count = tires\_t - tires\_detected; //tires need to be deployed

Pole[poles\_detected].tires\_deployed = t\_count;

Pole[poles\_detected].tires\_final = tires\_t;

lcd\_clear();

printf("det,%d, tcnt %d",tires\_detected,t\_count);

\_\_delay\_ms(1000);

if (t\_count<=2){

lcd\_clear();

printf("dfads");

\_\_delay\_ms(1000);

return (int) t\_count;

break;

}

}

}

//void start\_dc(){

// send=true;

// lcd\_clear();

// printf("I NEED HELP");

// \_\_delay\_ms(500);

// //Start I2C, writes 1 to the Arduino's serial monitor

// I2C\_Master\_Start(); // Start condition

// I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

// I2C\_Master\_Write('1'); // Write key press data which shows up on Arduino's serial monitor

// I2C\_Master\_Stop();

//}

void UI\_main(int t\_dep, int poles\_detected){

sens = 0;

// i2c\_mine();

// RD2 is the character LCD RS (Register Select pin)

// RD3 is the character LCD enable (E)

// RD4-RD7 are character LCD data lines

LATD = 0x00;

TRISD = 0x00; //set data direction as output

// Set all A/D ports to digital (pg. 222)

TRISD = 0x00; //set data direction as output

// Enable RB1 (keypad data available) interrupt

INT1IE = 1;

// Initialize LCD

initLCD();

// Enable interrupts

ei();

I2C\_Master\_Init(100000);

I2C\_Master\_Start();

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Stop();

// Main loop

Poles Pole[10]; //is this allowed

// I2C\_Master\_Start(); // Start condition

// I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

// I2C\_Master\_Write(temp); // Write key press data which shows up on Arduino's serial monitor

// I2C\_Master\_Stop();

unsigned long tick = 0;

if((pressed == 0)&&(sens==0)){

lcd\_clear();

printf("1 - Start");

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("2 - Summary");

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("3 - Date&Time ");

}

while(sens==0){

// if(pressed == 0){

// lcd\_clear();

// printf("1 - Start");

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// printf("2 - Summary");

// lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

// printf("3 - Date&Time ");

// }

if (send){

if(key\_was\_pressed){

pressed = 1;

key\_was\_pressed = false; // Clear the flag

unsigned char keypress = (PORTB & 0xF0) >> 4; //right shift

temp = keys[keypress];

temp\_int = (temp-'0');

if (temp\_int == 0){

state = 0;

lcd\_clear();

printf("1 - Start");

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("2 - Summary");

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("3 - Date&Time ");

}

if ((temp\_int == 1)&&(temp!= '\*')&&(temp!= '#')){

state = 0;

lcd\_clear();

printf("Machine In Use");

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu ");

//\_\_delay\_ms(2000);

//lcd\_clear();

start();

timer\_main(); //Start operation timer

sens = 1;

}

if ((temp\_int == 2)||(state == 2)){

if ((temp\_int == 2)){ //this loop is never traversed..

state = 2;

disp = 0;

lcd\_clear();

printf("Op Time:");

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("%d", cnt);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu #>");

}

if ((temp == '\*') && (disp != 0)){

disp = disp - 1;

}

if ((temp == '#')&& (disp <12)){

disp++;

}

if (disp == 0){

lcd\_clear();

printf("Op Time:");

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("%d", cnt);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu #>");

//\_\_delay\_ms(1500);

}

if (disp == 1){

lcd\_clear();

printf("No. Tires: %d",t\_dep);

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("No. Poles: %d", poles\_detected);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu #>");

//\_\_delay\_ms(1500);

}

if (disp > 1 && disp <11){

i = disp - 2;

lcd\_clear();

printf("P%d Dep:%d,Tot:%d", i+1 , Pole[i].tires\_deployed,Pole[i].tires\_final);

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("DistToStart:%d",i+1 , Pole[i].dist\_from\_start);

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("DistToCL:%d",i+1 , Pole[i].dist\_from\_cl);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu #>");

}

if (disp == 11){

i = disp - 2;

lcd\_clear();

printf("P%d Dep:%d,Tot:%d", i+1 , Pole[i].tires\_deployed,Pole[i].tires\_final);

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("DistToStart:%d",i+1 , Pole[i].dist\_from\_start);

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("DistToCL:%d",i+1 , Pole[i].dist\_from\_cl);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu ");

}

}

if (temp\_int == 3){

state = 0;

tick = 0;

while(~key\_was\_pressed){

if(tick % 1000 == 0){

lcd\_clear();

printf("DATE & TIME");

I2C\_Master\_Init(100000);

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b11010000); // 7 bit RTC address + Write

I2C\_Master\_Write(0x00); // Set memory pointer to seconds

I2C\_Master\_Stop(); // Stop condition

// Read current time

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b11010001); // 7 bit RTC address + Read

for(unsigned char i = 0; i < 6; i++){

time[i] = I2C\_Master\_Read(ACK); // Read with ACK to continue reading

}

time[6] = I2C\_Master\_Read(NACK); // Final Read with NACK

I2C\_Master\_Stop(); // Stop condition

// Print received data on LCD

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("%02x/%02x/%02x", time[6],time[5],time[4]); // Print date in YY/MM/DD

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("%02x:%02x:%02x", time[2],time[1],time[0]); // HH:MM:SS

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu ");

}

//\_\_delay\_ms(1000);

tick++;

//lcd\_clear();

}

//lcd\_clear();

}

}

}

}

}

void brake(){

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Write('9'); // Write key press data which shows up on Arduino's serial monitor

I2C\_Master\_Stop();

}

void start(){

I2C\_Master\_Init(100000);

I2C\_Master\_Start();

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Stop();

//Start I2C, writes 1 to the Arduino's serial monitor

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Write('1'); // Write key press data which shows up on Arduino's serial monitor

I2C\_Master\_Stop();

}

void main(){

int stack=1;

int t\_dep=0;

int t\_count;

TRISD = 0x00; // LCD Pins as Output

GIE = 1; //Global Interrupt Enable

ADCON1=0x0F; // Set all A/D ports to digital (pg. 222)

initLCD();

T1CON = 0x10; //Initialize Timer Module

int poles\_detected=0;

volatile long prev\_avg\_dist=0;

Poles Pole[10];

// int Pole[10];

// for (i=0; i<10; i++){

// Poles Pole[i];

// }

// actuators\_main(1);

if (~sens){

UI\_main( t\_dep, poles\_detected);

}

while (1){ //goes through this everytime a pole is detected and tires have been deployed

int actuator = 0;

// lcd\_clear();

// printf("happen");

// \_\_delay\_ms(1000);

if ((poles\_detected <=10)&&(avg\_dist<4000)){ //while there are less than 10 poles detected and the robot has travelled less than 4m

lcd\_clear();

printf("help");

// \_\_delay\_ms(1000);

// while (1){

// lcd\_clear();

// printf("sensing ");

// \_\_delay\_ms(2000);

if (sens){

ultrasonic\_main(actuator);

\_\_delay\_ms(2000);

}

// lcd\_clear();

// printf("fksfdas");

// \_\_delay\_ms(1000);

int pole\_cl\_dist = (avg\_dist)-(prev\_avg\_dist); //centerline to centerline distance

t\_count = number\_deploy(avg\_dist, poles\_detected);

lcd\_clear();

printf("tcnt %d", t\_count);

printf("tdep %d", t\_dep);

\_\_delay\_ms(500);

while(1){

if (t\_count<=2){ //check

lcd\_clear();

printf("bruh");

if (t\_dep <8){

stack = 1;

for(int i=0; i<(t\_count); i++){

actuators\_main(stack);

t\_dep++;

actuator=1;

}

}

if (t\_dep >= 8){

stack = 2;

for(int i=0; i<(t\_count); i++){

actuators\_main(stack);

t\_dep++;

actuator=1;

}

}

else{

actuator = 0;

}

}

else{

actuator = 0;

}

t\_count = number\_deploy(avg\_dist, poles\_detected); //check for the tires on the pole, assuming the ultrasonics keep running////////

lcd\_clear();

printf("here?");

\_\_delay\_ms(1000);

if (t\_count == 0){

lcd\_clear();

//doesn't get here /////////////////////////////////////

printf("BREAK");

\_\_delay\_ms(1000);

break;

}

}

// for (int i=0; i<10;i++)

Pole[poles\_detected].dist\_from\_cl = pole\_cl\_dist;

Pole[poles\_detected].dist\_from\_start = avg\_dist;

prev\_avg\_dist = avg\_dist ;

poles\_detected++;

// start\_dc(); //start DC motors running again after deployment

start();

// I2C\_Master\_Start(); // Start condition

// I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

// I2C\_Master\_Write('1'); // Write key press data which shows up on Arduino's serial monitor

// I2C\_Master\_Stop();

lcd\_clear();

printf("avg dist %d", avg\_dist);

printf("poles d %d", poles\_detected);

\_\_delay\_ms(2000);

}

}

}

// setup();

// lcd\_clear();

// printf("lol");

//

// start();

// \_\_delay\_ms(1000);

// brake();

// \_\_delay\_ms(1000);

// lcd\_clear();

// printf("fk");

//

// start();

// \_\_delay\_ms(1000);

// brake();

// \_\_delay\_ms(1000);

//}

//

//