#include <msp430g2553.h>

#define RX BIT1

#define TX BIT2

#define SEL1 BIT4

#define SEL0 BIT3

#define trigger BIT4

#define echo BIT5

#define LED0 BIT0

#define LED1 BIT6

#define CENTRO 0

#define DIREITA 1

#define ESQUERDA 2

#define EMBAIXO 3

int miliseconds;

long sensor;

#define BAUD\_9600 0

#define BAUD\_19200 1

#define BAUD\_38400 2

#define BAUD\_56000 3

#define BAUD\_115200 4

#define BAUD\_128000 5

#define BAUD\_256000 6

#define NUM\_BAUDS 7

void Send\_Data(unsigned char c);

void Send\_Int(int n);

void Send\_String(char str[]);

void Init\_UART(unsigned int baud\_rate\_choice);

struct sensoresUltrassonicos

{

int Medicao[4] ; // Medições para entrada do filtro média móvel

int valorFinal; // Resultado do filtro média móvel

};

int main(void)

{

struct sensoresUltrassonicos distancia[4];

volatile int i = 0;

WDTCTL = WDTPW + WDTHOLD;

int distance[3] = {0,0,0};

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

P2DIR |= SEL1; // P1.0 as output for LED

P2OUT &= ~SEL1;

P1DIR |= SEL0;

P1OUT &= ~SEL0;

P1DIR |= LED0;

P1DIR |= LED1;

P1OUT &= ~LED0;// P1.0 as output for LED

P1OUT &= ~LED1;

CCTL0 = CCIE; // CCR0 interrupt enabled

CCR0 = 1000; // 1ms at 1mhz

TACTL = TASSEL\_2 + MC\_1; // SMCLK, upmode

CCTL0 = CCIE; // CCR0 interrupt enabled

\_BIS\_SR(GIE);

Init\_UART(BAUD\_9600);

volatile int nroSensor = 0;

while(1)

{

switch(nroSensor)

{

case CENTRO:

P2OUT &= ~SEL1;

P1OUT &= ~SEL0;

break;

case DIREITA:

P2OUT &= ~SEL1;

P1OUT |= SEL0;

break;

case ESQUERDA:

P1OUT &= ~SEL0;

P2OUT |= SEL1;

break;

default:

// P1OUT &= ~SEL1;

// P1OUT &= ~SEL0;

}

P1IE &= ~0x01; // disable interupt

P1DIR |= trigger; // trigger pin as output

P1OUT |= trigger; // generate pulse

\_\_delay\_cycles(10); // for 10us

P1OUT &= ~trigger; // stop pulse

P1DIR &= ~echo; // make pin P1.2 input (ECHO)

P1IFG = 0x00; // clear flag just in case anything happened before

P1IE |= echo; // enable interupt on ECHO pin

P1IES &= ~echo; // rising edge on ECHO pin

distance[nroSensor] = sensor/58; // converting ECHO lenght into cm

Send\_String("Distância 1: ");

Send\_Int(distance[0]);

Send\_String("cm\n");

Send\_String("Distância 2: ");

Send\_Int(distance[1]);

Send\_String("cm\n");

Send\_String("Distância 3: ");

Send\_Int(distance[2]);

Send\_String("cm\n");

nroSensor++;

if(nroSensor >= 3)

nroSensor = 0;

if(distance[2] < 30)

{P1OUT |= LED1;

P1OUT |= LED0;}

else{

P1OUT &= ~LED1;

P1OUT &= ~LED0;}

if(distance[0] < 30)

P1OUT |= LED0;

else

P1OUT &= ~LED0;

if(distance[1] < 30)

P1OUT |= LED1;

else

P1OUT &= ~LED1;

}

}

void Send\_Data(unsigned char c)

{

while((IFG2&UCA0TXIFG)==0);

UCA0TXBUF = c;

}

void Send\_Int(int n)

{

int casa, dig;

if(n==0)

{

Send\_Data('0');

return;

}

if(n<0)

{

Send\_Data('-');

n = -n;

}

for(casa = 1; casa<=n; casa \*= 10);

casa /= 10;

while(casa>0)

{

dig = (n/casa);

Send\_Data(dig+'0');

n -= dig\*casa;

casa /= 10;

}

}

void Send\_String(char str[])

{

int i;

for(i=0; str[i]!= '\0'; i++)

Send\_Data(str[i]);

}

void Init\_UART(unsigned int baud\_rate\_choice)

{

unsigned char BRs[NUM\_BAUDS] = {104, 52, 26, 17, 8, 7, 3};

unsigned char MCTLs[NUM\_BAUDS] = {UCBRF\_0+UCBRS\_1,

UCBRF\_0+UCBRS\_0,

UCBRF\_0+UCBRS\_0,

UCBRF\_0+UCBRS\_7,

UCBRF\_0+UCBRS\_6,

UCBRF\_0+UCBRS\_7,

UCBRF\_0+UCBRS\_7};

if(baud\_rate\_choice<NUM\_BAUDS)

{

// Habilita os pinos para transmissao serial UART

P1SEL2 = P1SEL = RX+TX;

// Configura a transmissao serial UART com 8 bits de dados,

// sem paridade, comecando pelo bit menos significativo,

// e com um bit de STOP

UCA0CTL0 = 0;

// Escolhe o SMCLK como clock para a UART

UCA0CTL1 = UCSSEL\_2;

// Define a baud rate

UCA0BR0 = BRs[baud\_rate\_choice];

UCA0BR1 = 0;

UCA0MCTL = MCTLs[baud\_rate\_choice];

}

}

#pragma vector=PORT1\_VECTOR

\_\_interrupt void Port\_1(void)

{

if(P1IFG&echo) //is there interrupt pending?

{

if(!(P1IES&echo)) // is this the rising edge?

{

TACTL|=TACLR; // clears timer A

miliseconds = 0;

P1IES |= echo; //falling edge

}

else

{

sensor = (long)miliseconds\*1000 + (long)TAR; //calculating ECHO lenght

}

P1IFG &= ~echo; //clear flag

}

}

#pragma vector=TIMER0\_A0\_VECTOR

\_\_interrupt void Timer\_A (void)

{

miliseconds++;

}