#include <msp430g2553.h>

//Motores

#define MOTORA1 BIT0 // P1

#define MOTORA2 BIT7 // P1

#define MOTORB1 BIT6 // P1

#define MOTORB2 BIT6 // P2

#define MOTORCORTE BIT2 //P2 // LIGADO EM NIVEL LÓGICO BAIXO

#define LIGAMOTORA1 (P1OUT |= MOTORA1);

#define DESLIGAMOTORA1 (P1OUT &= ~MOTORA1);

#define LIGAMOTORA2 (P1OUT |= MOTORA2);

#define DESLIGAMOTORA2 (P1OUT &= ~MOTORA2);

#define LIGAMOTORB1 (P1OUT |= MOTORB1);

#define DESLIGAMOTORB1 (P1OUT &= ~MOTORB1);

#define LIGAMOTORB2 (P1OUT |= MOTORB2);

#define DESLIGAMOTORB2 (P1OUT &= ~MOTORB2);

//Seletora do multiplexador

#define SEL1 BIT4

#define SEL0 BIT3

//Sensores ultrasssonico

#define triggerC BIT1

#define triggerD BIT2

#define triggerE BIT4

#define triggerB BIT0 ///P2

#define echo BIT5

//LEDs para teste

#define LED0 BIT0

#define LED1 BIT6

#define LED2 BIT5

//Direções/

#define CENTRO 0

#define DIREITA 1

#define ESQUERDA 2

#define EMBAIXO 3

int miliseconds;

long sensor;

typedef struct sensoresUltrassonicos

{

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

int valorFinal; // Resultado do filtro média móvel

}sensoresUltrassonicos;

void filtro\_Media\_movel(sensoresUltrassonicos \*sensor);

int main(void)

{

sensoresUltrassonicos distancia[4] = {{.Medicao = {0,0,0,0}, .valorFinal = 0},{.Medicao = {0,0,0,0}, .valorFinal = 0},{.Medicao = {0,0,0,0}, .valorFinal = 0},{.Medicao = {0,0,0,0}, .valorFinal = 0}};

volatile int i,j = 0;

WDTCTL = WDTPW + WDTHOLD;

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

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

P2OUT &= ~SEL1;

P1DIR |= SEL0;

P1OUT &= ~SEL0;

// Motores

P2SEL &= ~(BIT6 +BIT7);

P2SEL2 &= ~(BIT6 +BIT7);

P1DIR |= MOTORA1;

P1OUT &= ~MOTORA1;

P1DIR |= MOTORA2;

P1OUT &= ~MOTORA2;

P1DIR |= MOTORB1;

P1OUT &= ~MOTORB1;

P1DIR |= MOTORB2;

P1OUT &= ~MOTORB2;

P2DIR |= MOTORCORTE;

P2OUT &= ~MOTORCORTE;

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);

volatile int nroSensor = 0;

volatile int nroMedicao = 0;

//inicializar\_sensores

//\_\_delay\_cycles(10000000); // for 10us

LIGAMOTORA1

// DESLIGAMOTORA2

LIGAMOTORB1

//DESLIGAMOTORB2

while(1)

{

switch(nroSensor)

{

case CENTRO:

P2OUT &= ~SEL1;

P1OUT &= ~SEL0;

while(nroMedicao <= 3)

{

\_\_delay\_cycles(10000); // for 10us

P1IE &= ~0x01; // disable interupt

P1DIR |= triggerC; // trigger pin as output

P1OUT |= triggerC; // generate pulse

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

P1OUT &= ~triggerC; // 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

distancia[CENTRO].Medicao[nroMedicao] = sensor/58;

// Send\_String("Distancia Centro:");

// Send\_Int(distancia[CENTRO].valorFinal);

//Send\_String("\n");

nroMedicao++;

}

filtro\_Media\_movel(&distancia[CENTRO]);

if( distancia[CENTRO].valorFinal < 30)

{

DESLIGAMOTORB1

DESLIGAMOTORA1

}

else

{

LIGAMOTORA1

LIGAMOTORB1

}

nroMedicao =0;

break;

case DIREITA:

P2OUT &= ~SEL1;

P1OUT |= SEL0;

while(nroMedicao <= 3)

{

\_\_delay\_cycles(10000); // for 10us

P1IE &= ~0x01; // disable interupt

P1DIR |= triggerD; // trigger pin as output

P1OUT |= triggerD; // generate pulse

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

P1OUT &= ~triggerD; // 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

distancia[DIREITA].Medicao[nroMedicao] = sensor/58;

//Send\_String("Distancia Direita:");

//Send\_Int(distancia[DIREITA].valorFinal);

//Send\_String("\n");

nroMedicao++;

}

nroMedicao = 0;

filtro\_Media\_movel(&distancia[DIREITA]);

break;

case ESQUERDA:

P2OUT |= SEL1;

P1OUT &= ~SEL0;

while(nroMedicao <= 3)

{

\_\_delay\_cycles(10000); // for 10us

P1IE &= ~0x01; // disable interupt

P1DIR |= triggerE; // trigger pin as output

P1OUT |= triggerE; // generate pulse

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

P1OUT &= ~triggerE; // 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

distancia[ESQUERDA].Medicao[nroMedicao] = sensor/58;

// Send\_String("Distancia Esquerda:");

//Send\_Int(distancia[ESQUERDA].valorFinal);

// Send\_String("\n");

nroMedicao++;

}

nroMedicao = 0;

filtro\_Media\_movel(&distancia[ESQUERDA]);

P2OUT &= ~SEL1;

P1OUT &= ~SEL0;

break;

case EMBAIXO:

P2OUT |= SEL1;

P1OUT |= SEL0;

while(nroMedicao <= 3)

{

\_\_delay\_cycles(10000); // for 10us

P1IE &= ~0x01; // disable interupt

P2DIR |= triggerB; // trigger pin as output

P2OUT |= triggerB; // generate pulse

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

P2OUT &= ~triggerB; // 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

distancia[EMBAIXO].Medicao[nroMedicao] = sensor/58;

// Send\_String("Distancia Embaixo:");

// Send\_Int(distancia[EMBAIXO].valorFinal);

// Send\_String("\n");

nroMedicao++;

}

nroMedicao = 0;

filtro\_Media\_movel(&distancia[EMBAIXO]);

\_\_delay\_cycles(10000); // for 10us

break;

}

nroSensor = (nroSensor +1) % 4 ;

}}

void filtro\_Media\_movel(sensoresUltrassonicos \*sensor)

{

sensor->valorFinal= (sensor->Medicao[0]+ sensor->Medicao[1]+ sensor->Medicao[2] + sensor->Medicao[3])/4;

}

#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++;

}