Proposta de Projeto

Cinto de Ecolocalização para Cegos

Adriano Silva de Moraes
Faculdade do Gama
Universidade de Brasília - UnB
Gama-DF, Brasil

Hachid Habib Cury
Faculdade do Gama
Universidade de Brasília - UnB
Gama-DF, Brasil

I. Justificativa

Do total da população brasileira, 23,9% (45,6 milhões de pessoas) declararam ter algum tipo de deficiência. Entre as deficiências declaradas, a mais comum foi a visual, atingindo 3,5% da população brasileira. Entre os deficientes visuais 528.624 pessoas são incapazes de enxergar (cegos), de acordo com os dados do IBGE de 2010. Visando incluir da melhor maneira os deficientes visuais e resolver problemas que os portadores de deficiência visual enfrentam, como a mobilidade reduzida e a dependência de algum auxílio como cães guias, bengalas, parentes, amigos e etc.

O mercado oferece produtos similares ao proposto no projeto. As bengalas sensoriais funcionam com sensores ultrassônicos que possibilitam a identificação que obstáculos no caminho do usuário, o preço desse produto chega a 1350.00 €, ou seja R$ 5895.00 como mostrado na figura 1.

Figura 1: Bengala sensorial comparação de preço.

O mesmo projeto já foi trabalhado por engenheiros como o brasileiro Rodrigo Azevedo. Ele é um dos responsáveis pelo projeto Bengala IoT, que visa criar um modelo replicável de uma bengala inteligente para deficientes visuais. De código aberto e sem fins lucrativos, conforme o criador do projeto, o pacote básico que é composto por dois sensores e um microcontrolador simples, poderia sair na faixa dos R$ 100 ou R$ 200.

II. Objetivos

A dupla propõe a criação de um cinto eletrônico composto por sensores ultrassônicos e vibradores, que ofereçam ao usuário a capacidade de se orientar no ambiente sem os incômodos gerados pelas bengalas, que são comumente utilizadas, já que os produtos que utilizam a mesma tecnologia proposta possuem preços exorbitantes para a maioria da população cega.

III. Benefícios

Os usuários devem tirar os mesmos proveitos que as ferramentas atuais, ou seja, saber com antecedência a distância dos obstáculos, evitando assim acidentes de locomoção, dar mais liberdade ao usuário para se locomover em público com mais autoconfiança, mas com o diferencial de utilizar um equipamento sutil e sem inconvenientes para tal.

IV. hardware

Lista de materiais:

- MSP-EXP430G2553LP
- Quatro sensores de distância ultrassônicos (HC-SR04)
- Quatro vibracall's (Modelo F1030300112)
- Resistores 4.7 KΩ
- Resistores 10KΩ
- Quatro Transistores BCE549B
- Protoboard
- Jumpers
- Cinta elástica

Descrição do Hardware:

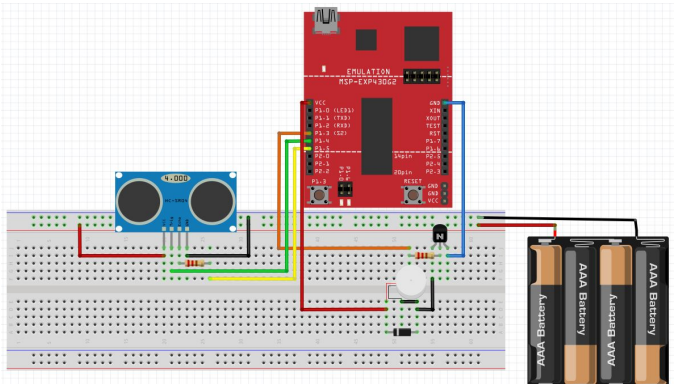

Figura 2: Esquemático do circuito apresentado.

A cinta elastica será a base para os quatro sensores, microcontrolador e para os demais dispositivos eletrônicos, conterá toda a circuitaria.

Os sensores ficarão distribuídos da seguinte maneira, 1 sensor apontado para cima de modo que identifique objetos a 50 cm do rosto do usuário, 1 sensor virado para baixo para que identifique obstáculos a 50 cm dos pés do mesmo e 2 sensores virados para frente, para que além de mostrar ao usuário obstáculos a 1 metro a sua frente, distingue em que lado se encontram, possibilitando assim ao utilizador da cinta de ecolocalização saber onde está a obstrução e a que distância se encontra do mesmo. Para isso, serão distribuídos vibracall's ao longo da cinta que funcionarão da seguinte forma, os vibracall da esquerda e da direita mostram objetos que estão na esquerda e à direita do usuário e sua proximidade por vibração (objetos mais pertos vibraram mais que objetos mais longes), o sensor de cima vibrará com força total quando identificar algo perto do rosto do utilitário e o sensor de baixo terá dois modos de vibração, vibrará ininterruptamente para degraus, meu fios e etc, e múltiplas vibrações para buracos.

O microcontrolador a ser utilizado no projeto será o M430G2553 da Texas Instruments (Figura 3). Esse dispositivo trabalha na faixa de 3.3V, assim como os vibracalls (Figura 4), enquanto os sensores de distância operam a 5V (Figura 5) e a bateria a 9V (Figura 6).

Embora os componentes escolhidos sejam bastante populares e acessíveis exigem a utilização de vários divisores de tensão para operarem em conjunto. Uma alternativa para esse inconveniente é utilização da placa MB102, mostrada na Figura 7. Trata-se de um módulo de fonte de alimentação, que recebe uma tensão de 5V a 12V em uma de suas entradas e oferece as tensões de 3.3V ou 5V em seus terminais de saída. Os terminais de entrada oferecem interface para alimentação via USB ou bateria comum.

Figura 3: Microcontrolador M430G2553 (3.3V)

Figura 4: Vibracall (3.3V)

Figura 5: Sensor de distância HC-SR04 (5V)

Figura 6: Bateria alcalina (9V)

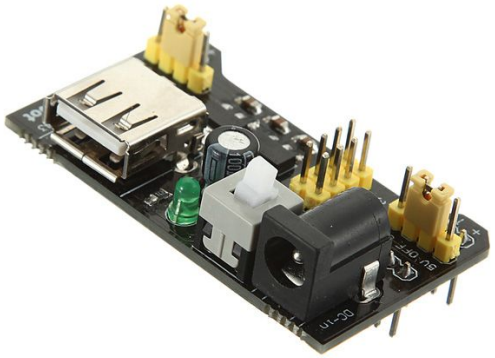

Figura 7: Módulo de alimentação MB102 3.3V/5V

A conexão entre os principais componentes é feita conforme o diagrama esquemático abaixo (Figura 8). Para solicitar a leitura de distância o microcontrolador envia um sinal de Trigger (fios verdes) ao sensor desejado. Em seguida, o sensor envia um sinal de Echo (fios amarelos) correspondente à distância de um obstáculo. Com essas informações o microcontrolador envia um sinal de PWM (fios azuis) a um determinado Vibracall.

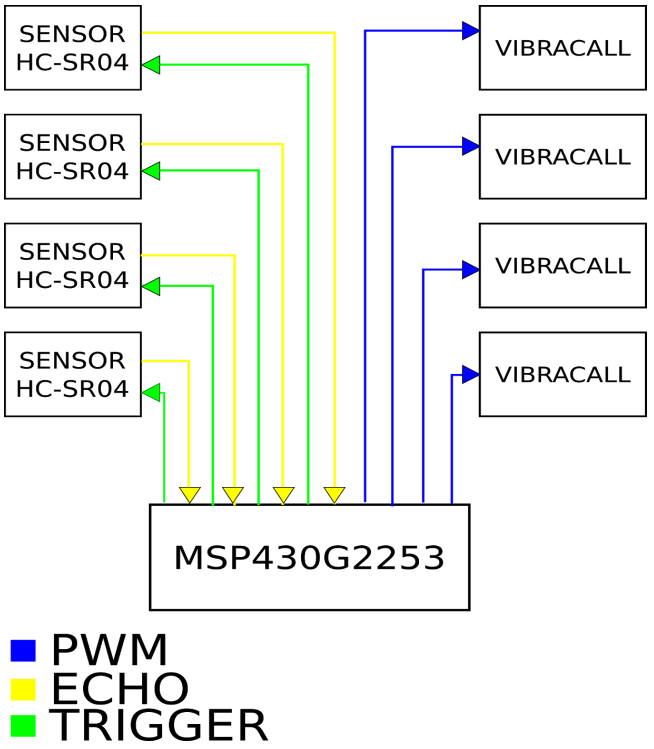

Figura 8: Esquema básico de interconexão

As características da onda referente a cada um desses sinais do sensor ultrassônico é mostrada na Figura 9. Nela podemos observar um sinal que não citamos até o momento, o Module Internal Signal. Esse é o sinal de trabalho interno utilizado pelo HC-SR04, o qual não nos será útil. Os sinais que nos interessam são apenas o Trigger e o Echo e a maneira que os tratamos via software será explicado adiante.

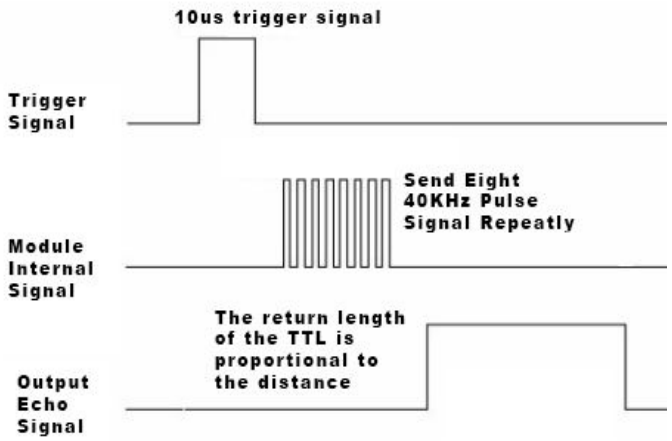

Figura 9: Sinais utilizados pelo sensor HC-SR04

V. SOFTWARE

O código foi criado para ser compilado pelo software Code Composer Studio e possui 2 funções principais, a função Atraso_us, que é utilizada para criar atrasos em microsegundos e é utilizada no código para controle do tempo mínimo que Trigger do sensor ultrassônico fique em alto. A função SensorUlt foi criada para que o projetista informe a posição do Trigger e Echo no MSP430 e seu retorno é a distância em centímetros, para isso é necessário levar Trigger para o nível alto durante 10µs e esperar que Echo retorne em nível alto, com isso é utilizado o TIMER_A1 CC2 do MSP em modo de captura para ler o tempo do pulso em microsegundos. Com a leitura do pulso utiliza-se a seguinte equação para descobrir a distância do objeto a frente do sensor.

$$Distancia = \frac{T_{Pulso} \times V_{Som}}{2} \simeq \frac{T_{Pulso}}{58}$$

Já que a velocidade do som igual a:

$$V_{Som} = 0{,}0343 \ \frac{cm}{\mu s}$$

A main do programa possui um loop infinito que chama a função SensorUlt e joga seu retorno em uma variável inteira nomeada “distancia”, o valor de distancia é comparado com faixas de distâncias e se seu valor encaixar na faixa é dado um valor para `DUTY_CYCLE` que com a utilização do TIMER_A1 em modo comparação é transmitido para o Vibracall uma onda quadrada PWM. A execução da main fornece ao motor vibracall uma intensidade maior com a diminuição da distância entre o sensor e o objeto, ou seja,

quanto mais perto do objeto maior deve ser a vibração do motor e a luminosidade do led.

A geração dos sinais PWM foi implementado nas funções PwmVibraFrontal(PWM, distancia), PwmVibraSuperior (PWM, distancia) e PwmVibraInferior (PWM, distancia) - omitimos o tipo dos argumentos para melhor visualização. O parâmetro PWM indica os pinos de saída do sinal e o parâmetro distancia fornece as informações para modulação desse sinal.

Para os vibracalls da direita e esquerda foram implementadas seis velocidades, baseadas na distância do obstáculo. Os sinais de PWM são gerados com base na distância através da relação demonstrada abaixo, na Figura 10.

Os sensores superiores e inferiores geram apenas 3 dessas ondas. Sao elas PWM com 0% de DUTY_CYCLE, PWM com 45% de DUTY_CYCLE e PWM com 98% de DUTY_CYCLE.

Uma particularidade do sensor superior é que a distância utilizada como referência para gerar os sinais PWM é calculada com relação à distância inferior durante a inicialização da execução e corresponde a 90% dela.

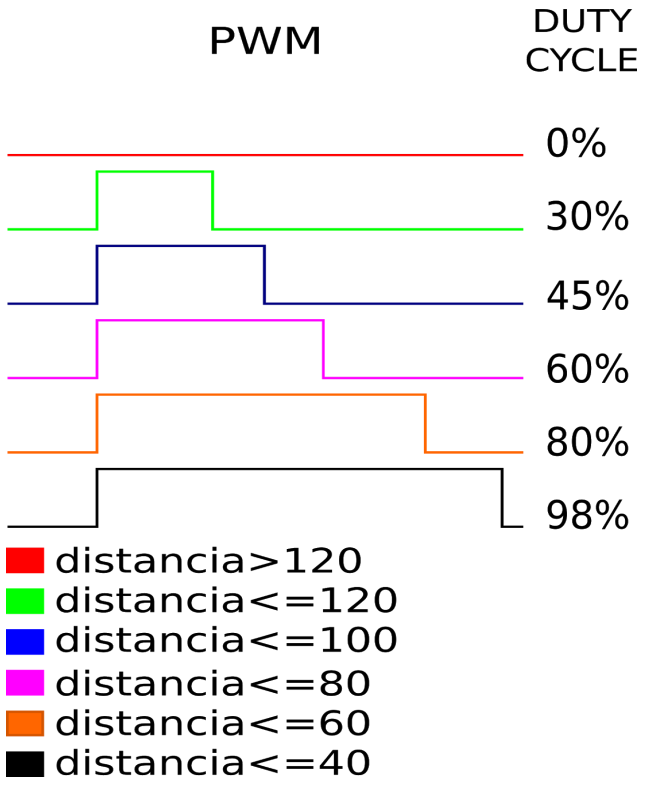

Figura 10: Relação entre sinais PWM e distância de um obstáculo.

A montagem do equipamento foi feita com o objetivo de tornar o produto final com o menor tamanho possível. Isso foi decidido por que um dos requisitos estabelecidos pelas pessoas cegas foi deixar o uso do cinto bastante sutil. Esse também foi um dos motivos da utilização de vibracalls ao invés de buzzers. Também foi utilizada apenas metade de uma protoboard para ligação dos equipamentos.

Todos esses dispositivos foram acomodados em uma caixinha de jóias. Os sensores ultrassônicos foram colocados na tampa e os vibracalls na parte externa inferior da caixa. Também foi colocado uma chave para ligar/desligar o circuito e um botão para determinação da altura referente ao chão do utilizador. Esses detalhes podem ser vistos nas figuras abaixo.

Figura 11: Alocação dos vibracalls

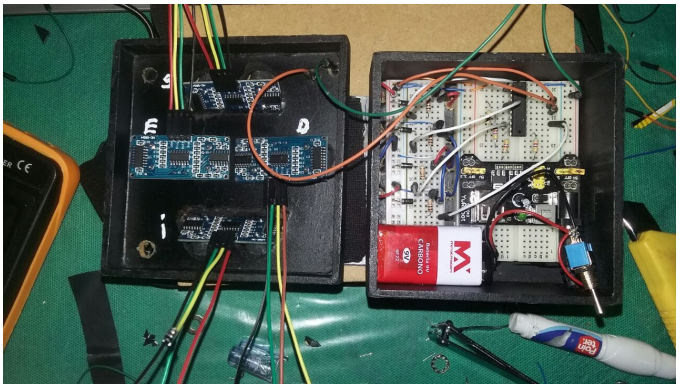

Figura 12: Acomodação interna dos dispositivos

Figura 13: Cinto de ecolocalização finalizado

VI. Discussão

A evolução do trabalho ocorreu como desejado, já que o projeto já está sendo executado no software CCS e todos elementos do projeto estão em funcionamento. Entretanto, o grupo encontrou algumas dificuldades técnicas com relação ao funcionamento do sensor HC-SR04 (sensor ultrassônico). Esse sensor não funciona muito bem quando apontado para superfícies muito planas, o que diminui sua confiabilidade no projeto.

VII. Conclusão

O projeto final foi realizado conforme proposto. Podemos afirmar que, em breve, as pessoas cegas poderão ter mais uma opção para melhorar sua qualidade de vida com o melhoramento do cinto para uso comercial.

Em relação à finalidade didática do projeto foram colocados em prática vários conceitos sobre o funcionamento dos microcontroladores e eletrônica digital. Além disso, também foram aplicados conhecimentos da área de engenharia para resolver os novos problemas que surgiam no desenvolvimento do projeto e demandas dos clientes (pessoas cegas).

Logo, conclui-se que o projeto final foi uma experiência bastante próxima daquela que será encontrada na vida profissional de um engenheiro, cheia de desafios e gratificações por melhorar cada vez mais a vida da população.

VIII. Revisão Bibliográfica

[1] Davies, J., MSP430 Microcontroller Basics, Elsevier, 2008.

[2] BRASIL, IBGE. Pesquisa sobre quantidade de cegos no Brasil, Censo Demográfico, 2010. Disponivel em: <www.ibge.gov.br>. Acesso em: 30/04/2018.

[3] Bengala eletrônica Ultracane , comparativo de preço no mercado. Disponivel em:<https://www.megaserafim.pt/online/orientacao-e-mobilidade/41-bengala-electronica-ultracane.html> Acesso em: 03/06/2018.

[4] Brasileiro cria bengala inteligente que usa sensores para guiar deficientes. Disponivel em:<https://www.tudocelular.com/tech/noticias/n119414/cpbr11-brasileiro-bengala-inteligente-iot.htmll> Acesso em: 03/06/2018.

```
#include <msp430.h>

//Entradas e Saídas do sensor 0 (Frontal Esquerda)
#define Vibra0 BIT0
#define Trig0 BIT1
#define Echo0 BIT2
unsigned int distanciaE=200;

//Entradas e Saídas do sensor 1 (Frontal Direita)
#define Vibra1 BIT3
#define Trig1 BIT4
#define Echo1 BIT5
unsigned int distanciaD=200;

//Entradas e Saídas do sensor 2 (Inferior)
#define Vibra2 BIT0
#define Trig2 BIT1
#define Echo2 BIT2
unsigned int distanciaI=200;

//Entradas e Saídas do sensor 3 (Superior)
#define Vibra3 BIT5
#define Trig3 BIT4
#define Echo3 BIT3
unsigned int distanciaS=200;

//Variaveis Gerais
unsigned int TempoI=0;
unsigned int TempoF=0;
unsigned int TempoPWM=0;
unsigned int DUTY_CYCLE=0;
#define PERIODO 0xFFFF
#define Botao BIT7
unsigned int AltInferior=108;
unsigned int AltSuperior=85;
unsigned int CondIn;

//Função Criada para causar atraso em microsegundos
void Atraso_us(volatile unsigned int us)
{
	TA1CCR0 = us-1;
	TA1CTL = TASSEL_2 + ID_0 + MC_1;

	while((TA1CTL & TAIFG)==0);
	TA1CTL = TACLR;
}
//Função Verifica PWM da direita e esquerda
void PwmVibraFrontal(volatile unsigned int PWM, volatile unsigned int distancia)
{
	//Configura PWM
	if(distancia<=40){
		DUTY_CYCLE= PERIODO-2;
```

```
        }
        else if (distancia<=60){
                DUTY_CYCLE= PERIODO*0.80;
        }
        else if (distancia<=80){
                DUTY_CYCLE= PERIODO*0.60;
        }
        else if (distancia<=100){
                DUTY_CYCLE= PERIODO*0.45;
        }
        else if (distancia<=120){
                DUTY_CYCLE= PERIODO*0.30;
        }
        else {
                DUTY_CYCLE= 0;
        }

        //Executa PWM
        TempoPWM=TA0R;
        if(TempoPWM>=DUTY_CYCLE){
                P1OUT &= ~PWM;
        }else{
                P1OUT |= PWM;
        }

}
//Função Verifica PWM
void PwmVibraSuperior(volatile unsigned int PWM, volatile unsigned int distancia)
{
        AltSuperior=AltInferior*0.9;
        //Configura PWM
        if(distancia<=AltSuperior){
                P2OUT |= PWM;
        }else {
                P2OUT &= ~PWM;
        }
}
//Função Verifica PWM
void PwmVibraInferior(volatile unsigned int PWM, volatile unsigned int distancia)
{
                //Configura PWM
                if(distancia>= AltInferior + 20){
                DUTY_CYCLE= PERIODO-2;
                }
                else if (distancia<=AltInferior - 20){
                DUTY_CYCLE= PERIODO*0.40;
                }
                else {
                DUTY_CYCLE= 0;
                }

                //Executa PWM
                TempoPWM=TA0R;
                if(TempoPWM>=DUTY_CYCLE){
                P2OUT &= ~PWM;
```

```
            }else{
            P2OUT |= PWM;
            }
}
//Função criada para controlar sensores ultrassônicos P1
int SensorUltP1(volatile unsigned int Trig, volatile unsigned int Echo)
{
      int distancia=0;
      int i;
      int calculo;
      calculo= 0;

      for(i=0;i<3;i++){

      P1OUT |= Trig;                          //Colocar Trig em alto durante 10 us
      Atraso_us(10);

      TA1CTL = TACLR;                         // Zerando o TAR
      TA1CTL = TASSEL_2 + ID_0 + MC_2;        // Configurando clock do timer
      TA1CCTL2 = CM_3 + CCIS_0 + CAP;         // Configurando modo de capitura

      P1OUT &= ~Trig;

      while((P1IN&Echo)==0);
      TempoI=TA1R;
      while((P1IN&Echo)==Echo);
      TempoF=TA1R;

            PwmVibraFrontal(Vibra0,distanciaE);
            PwmVibraFrontal(Vibra1,distanciaD);
            PwmVibraInferior(Vibra2,distanciaI);
            PwmVibraSuperior(Vibra3,distanciaS);

      calculo = (TempoF - TempoI)/58;  // Calculo da distancia
      distancia = distancia + calculo;
      }
      distancia = distancia/3;
      return distancia;
}
//Função criada para controlar sensores ultrassônicos P1
int SensorUltP2(volatile unsigned int Trig, volatile unsigned int Echo)
{
      int distancia=0;
      int i;
      int calculo;
      calculo= 0;

      for(i=0;i<3;i++){

      P2OUT |= Trig;                          //Colocar Trig em alto durante 10 us
      Atraso_us(10);

      TA1CTL = TACLR;                         // Zerando o TAR
      TA1CTL = TASSEL_2 + ID_0 + MC_2;        // Configurando clock do timer
      TA1CCTL2 = CM_3 + CCIS_0 + CAP;         // Configurando modo de capitura
```

```
        P2OUT &= ~Trig;

        while((P2IN&Echo)==0);
        TempoI=TA1R;
        while((P2IN&Echo)==Echo);
        TempoF=TA1R;

                PwmVibraFrontal(Vibra0,distanciaE);
                PwmVibraFrontal(Vibra1,distanciaD);
                PwmVibraInferior(Vibra2,distanciaI);
                PwmVibraSuperior(Vibra3,distanciaS);

                calculo = (TempoF - TempoI)/58;  // Calculo da distancia
                distancia = distancia + calculo;
                }
                distancia = distancia/3;
        return distancia;
}
void main(void)
{
        BCSCTL1 = CALBC1_1MHZ;
        DCOCTL = CALDCO_1MHZ;                   // SMCLK = 1M Hz

        WDTCTL = WDTPW + WDTHOLD;               // Stop WDT

        TA0CTL = TACLR;                         // Zerando o TAR
        TA0CTL = TASSEL_2 + ID_0 + MC_2;        // Configurando clock do timer
        TA0CCTL1 = CM_3 + CCIS_0 + CAP ;        // Configurando modo de capitura + CCIE

        P1DIR &= ~Botao;                        // Botao é entrada
        P1REN |= Botao;                         // Habilita pull-up
        P1OUT |= Botao;                         // Inicia em Alto
        P1IE |= Botao;                          // Habilita a interrupção no botão
        P1IES |= Botao;                         // Interrupção na borda de descida

        P1DIR |= Trig0;                         //Configura Sensor Esquerda
        P1OUT &= ~Trig0;                        //Configura Sensor Esquerda
        P1DIR &= ~Echo0;                        //Configura Sensor Esquerda
        P1DIR |= Vibra0;                        //Configura Sensor Inferior
        P1OUT &= ~Vibra0;                       //Configura Sensor Inferior
        P1DIR |= Trig1;                         //Configura Sensor Direita
        P1OUT &= ~Trig1;                        //Configura Sensor Direita
        P1DIR &= ~Echo1;                        //Configura Sensor Direita
        P1DIR |= Vibra1;                        //Configura Sensor Inferior
        P1OUT &= ~Vibra1;                       //Configura Sensor Inferior
        P2DIR |= Trig2;                         //Configura Sensor Inferior
        P2OUT &= ~Trig2;                        //Configura Sensor Inferior
        P2DIR &= ~Echo2;                        //Configura Sensor Inferior
        P2DIR |= Vibra2;                        //Configura Sensor Inferior
        P2OUT &= ~Vibra2;                       //Configura Sensor Inferior
        P2DIR |= Trig3;                         //Configura Sensor Superior
        P2OUT &= ~Trig3;                        //Configura Sensor Superior
        P2DIR &= ~Echo3;                        //Configura Sensor Superior
        P2DIR |= Vibra3;                        //Configura Sensor Inferior
```

```
	P2OUT &= ~Vibra3;                       //Configura Sensor Inferior

	CondIn=0;

	_BIS_SR(GIE);

	while(CondIn==0){
		P1OUT ^= (Vibra0 + Vibra1);
		P2OUT ^= (Vibra2 + Vibra3);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		Atraso_us(50000);
		}
	P1OUT &= ~(Vibra0 + Vibra1);
	P2OUT &= ~(Vibra2 + Vibra3);

	while(1){

	distanciaE= SensorUltP1(Trig0, Echo0);
	PwmVibraFrontal(Vibra0,distanciaE);
	PwmVibraFrontal(Vibra1,distanciaD);
	PwmVibraInferior(Vibra2,distanciaI);
	PwmVibraSuperior(Vibra3,distanciaS);

	distanciaD= SensorUltP1(Trig1, Echo1);
	PwmVibraFrontal(Vibra0,distanciaE);
	PwmVibraFrontal(Vibra1,distanciaD);
	PwmVibraInferior(Vibra2,distanciaI);
	PwmVibraSuperior(Vibra3,distanciaS);

	distanciaI= SensorUltP2(Trig2, Echo2);
	PwmVibraFrontal(Vibra0,distanciaE);
	PwmVibraFrontal(Vibra1,distanciaD);
	PwmVibraInferior(Vibra2,distanciaI);
	PwmVibraSuperior(Vibra3,distanciaS);

	distanciaS= SensorUltP2(Trig3, Echo3);
	PwmVibraFrontal(Vibra0,distanciaE);
	PwmVibraFrontal(Vibra1,distanciaD);
	PwmVibraInferior(Vibra2,distanciaI);
	PwmVibraSuperior(Vibra3,distanciaS);

 }
}
#pragma vector=PORT1_VECTOR
__interrupt void LeitorAltura(void)
{
	while((P1IN&Botao)==0);
	int i;
	int calculo;
```

```
        calculo= 0;

        for(i=0;i<5;i++){
        AltInferior= SensorUltP2(Trig2, Echo2);
        calculo= calculo + AltInferior;
        }
        AltInferior= calculo/5;
        P1IFG &= ~Botao;
        CondIn=1;
}
```