Sensor de Estacionamento

Microcontroladores e Microprocessadores

Pedro Willian Santos Ribeiro Calile
FGA - Universidade de Brasília, UnB
Gama-DF, Brasil
Calile-@hotmail.com
15/0144989

***Resumo*— Este relatório final tem como intuito apresentar o apresentar o projeto final da disciplina Microprocessadores e Microcontroladores, tendo como tema o Sensor de Estacionamento com visor LCD.**

Keywords*— *Microcontrolador; Sensor Ultrassônico; Distãncia; Linguagem C; MSP430.

I. INTRODUÇÃO

Os produtos que incorporam microcontroladores em seu sistema visam principalmente, aumentar seus recursos, reduzir seu tamanho e custo, melhorar sua confiabilidade e diminuir o consumo de energia.

Portanto podemos dizer que um microcontrolador é um dispositivo que integra hardware e software. Através do código de programação consegue-se controlar um hardware para fazer funções específicas de uma maneira fantasticamente simples, fácil, flexível e eficaz.

Depois de uma vasta pesquisa nas possibilidades que o microcontrolador MSP430 pode comportar, foi escolhido a proposta para o projeto final da disciplina de Microprocessadores e Microcontroladores de realizar a montagem de um Sensor de Estacionamento com objetivo de auxiliar motoristas no ato de estacionar seus carros, com precisão de medidas e interação visual facilitadora em pontos cegos do veículo.

Estacionar nem sempre é a coisa mais fácil do mundo. Muitos motoristas, por mais experientes que sejam, podem encontrar dificuldades na hora de posicionar seu carro por encontrar alguns obstáculos, como árvores, postes, vagas muito apertadas, balizas e etc. Nestes casos, o Sensor de Estacionamento com visor luminoso e indicação de metragem precisa auxiliam que a manobra fique mais simples e com menores chances de danificar o veículo com colisões indesejadas.

II. OBJETIVOS

A escolha do projeto do Sensor de Estacionamento com visor LCD justifica-se na necessidade de informações sensoriais mais precisas nas regiões de ponto cego dos automóveis. Entretanto o projeto tem como objetivo comercial baratear e popularizar esse dispositivo.

III. REQUISITOS

O microcontrolador utilizado no projeto será o MSP430 da família F5, que consiste em um microcontrolador de propósito geral de baixo consumo de potência, desenvolvido pela *Texas Instruments*. Devido às características de baixíssimo consumo de energia, alto desempenho e baixo custo, o microcontrolador MSP430 torna-se extremamente popular e indicado para implementação do Sensor de Estacionamento proposto.

Além do MSP430F5529, serão utilizados:

- Display Cristal Líquido (LCD 16x02 – VD/PT) com MÓDULO I2C: visualização de medições de distância.
- Sensor Ultrassônico HC-SR04: medição da distância dos obstáculos;
- Jumpers: conexões;
- Suporte de madeira: base dos componentes;

Muitos sistemas exigem algum tipo de implementação de medida ou sensoriamento de distância. Uma exigência de muitos desses aplicativos é o baixo consumo, de modo a estender a durabilidade da bateria.

Uma opção moderna é a que faz uso de sensores integrados como o HC-SR04 que pode ser interfaceado com um microcontrolador de baixo consumo como o MSP430.

O sensor de proximidade HC-SR04 é equipado de um autofalante e um microfone ultrassônicos (figura 1). Quando acionado por um pulso positivo na entrada trigger o sensor envia pulsos sonoros ultrassônicos inaudíveis ao ouvido humano.

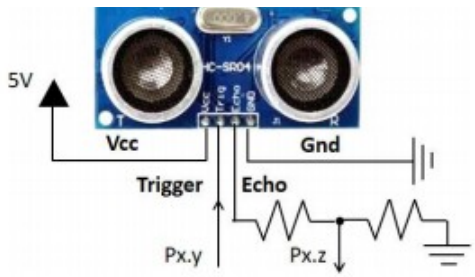

Figura1– Sensor ultrassônico de proximidade HC-SR04.

A resposta do tempo de propagação é devolvida no sinal echo (figura 2).

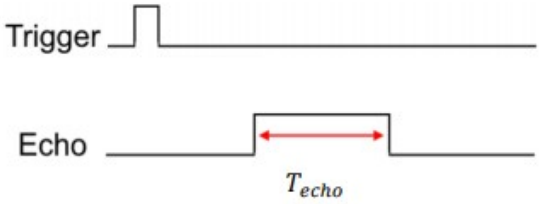

Figura 2 – Resposta no tempo de impulso de iniciação (trigger) e margem de tempo de captação do obstáculo (echo).

Segue abaixo uma visão de montagem do protótipo (Figura 3) feita no *software Fritzing*, dos componentes que serão utilizados na montagem do Sensor de Estacionamento com visor LCD.

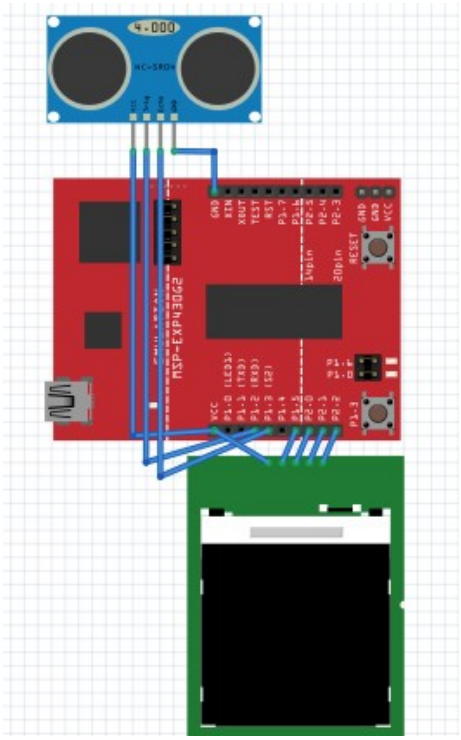

Figura 3 – Visão Protótipo.

A programação do MSP430 utilizada foi a linguagem C para MSP430, utilizando o software *Code Composer Studio* (CCS) e também conhecimentos de utilização de softwares como aplicativos, sistemas operacionais, compiladores em C, linguagens de máquina e programação.

O sistema é projetado para operar com o mínimo de consumo de forma constante, mostrando no display a distância entre o sensor e o obstáculo quando o botão da placa do MSP430 é pressionado. As informações são apresentadas no display via comunicação serial I2C.

A quantidade de pinos disponíveis no MSP430 é limitada, pois após conectar o display e o sensor sobram poucos pinos para uso. Contudo, com este Módulo Serial I2C controla-se o display lcd usando apenas 2 pinos SDA e SCL.

IV. Teste do Sistema Proposto

O Sistema recebe como entrada de dados no sensor ultrassônico os sinais de proximidade obstáculos próximos via Trigger, devolve para o sensor via Echo o tempo que levou para encontrar o obstáculo mais próximo e calcula a distância sabendo que D=VxT. Após receber e calcular a distância, avisa com sinal luminoso via Display LCD.

As medidas apresentadas no visor são dadas em centímetros, tendo em vista as especificações de fábrica do sensor ultrassônico que consegue medir até 100 centímetros (1 metro), A figura 4 mostra a medição a uma distância de 10 centímetros junto a verificação de uma régua milimetrada para averiguar a medição.

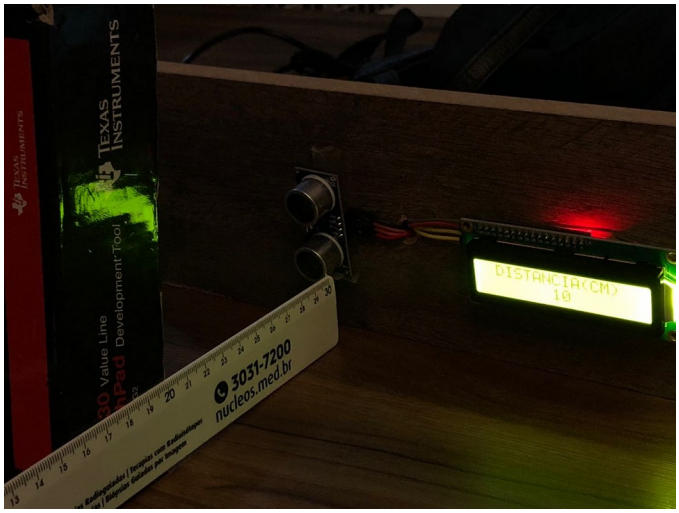

Figura 4 – Medição de 10 cm com auxílio da régua.

Figura 5 – Visão nítida da medição mostrada na figura 4.

No código foi usado a variável int, portanto apenas valores inteiros na medição em centímetros são apresentados, conforme a figura 6 apresenta uma distância de 10,5 cm, mas o display apresenta a distância de 10 cm. O arredondamento sempre será feito para o inteiro inferior para efeito de segurança, onde a proximidade do objeto é o fator que não deseja-se na ação de estacionar.

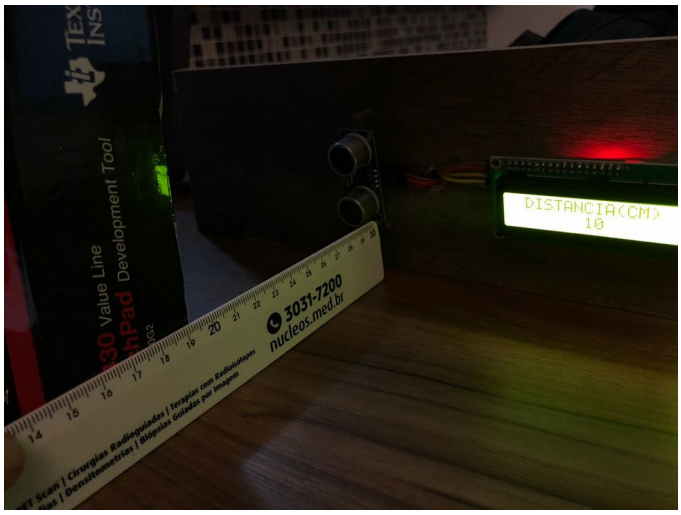

Figura 6 – Medição de 10,5 cm com auxílio da régua. No display apresenta medição do inteiro inferior (10 cm).

O espectro de parâmetros de medição da distância foram decididos na forma de boa conduta de direção e estacionamento, pensando que são medidas razoáveis. Não existe legislação que regule a distância exata exigida entre veículos ou entre obstáculos, apenas que devem encontrar-se dentro das limitações estipuladas nas indicações de demarcação do solo.

V. Resultados e Conclusões

Os resultados obtidos foram de um medidor de distância consideravelmente robusto, sem *bugs*, boa aparência, baixo consumo de energia e, principalmente, precisão na finalidade principal de auxílio ao motorista. Alcançando, desta forma, a segurança e a prevenção contra batidas e amassados .

A apresentação do protótipo final do projeto mostrou medições precisas, funcionando similar a uma régua eletrônica. Para funcionar automaticamente sem necessidade da interrupção do botão para fazer as medidas, é necessária uma implementação de rotina com loop infinito substituindo a interrupção do botão.

O código implementado para a medição, interrupção, comunicação serial I2C e demais comandos necessários, está em anexo.

Por fim fica a satisfação de projetar este Sensor de Estacionamento com as principais operações basicas realizadas com sucesso e o otimismo que esse projeto seja um primeiro passo para modificações mais avançadas de autonomia do sistema e implementação em larga escala.

References

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

[2] Sensor Ultrassônico – HC-SR04 . Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Acesso em 20 de Junho de 2018.

[3] Código de Trânsito Brasileiro. Disponível em : <http://www.detran.df.gov.br/o-detran/base-juridica/item/download/35_366b0acc74ed6d40f2a7370e29e88bd5.html> . Acesso em 19 de junho de 2018.

[4] Microcontroladores. Disponiível em: <https://pt.wikipedia.org/wiki/Microcontrolador>. Acesso em 15 de abril de 2018.

[5] MSP430. Disponível em: <https://pt.wikipedia.org/wiki/MSP430>. Acesso em 15 de abril de 2018.

[6] LCD – Liquid Crystal Display. Disponível em: <https://pt.wikipedia.org/wiki/LCD>. Acesso em 15 de abril de 2018.

[7] Pereira, F., Micrcontroladores MSP430: Teoria e Prática, 1a ed., Érica.

ANEXO

```
#include <msp430.h>

int adc_h;
int adc_v;
volatile int i;
volatile int tempo;
volatile int distancia;
volatile int teste;

void config(){
    // config botão para acionamento do
trigger
    P2DIR &= ~BIT1;      //P2.1 ENTRADA
    P2REN |= BIT1;       //P2.1 HABLITA
RESISTOR
    P2OUT |= BIT1;       //P2.1 PULL-UP
(RESISTOR EM VCC)

    // config trigger
    P1DIR |= BIT2;      // configurar como
saída da placa
    P1SEL |= BIT2;

    // config echo
    P1DIR &= ~BIT3;     // configurar como
entrada
    P1SEL |= BIT3;

    // config leds

    P6DIR |= (BIT0|BIT1);       // P6.0 e P6.1
SAIDA
```

```
    P6OUT &= ~(BIT0|BIT1);
    // config timers
    TA0CTL = TASSEL_2 | MC_1;
    TA0CCR0 = 0xFFFF;

    // config valor de trigger indo para 1
durante 10u seg
    TA0CCR1 = 10;

    // config captura com echo
    TA0CCTL2 = CM_3 | CCIS_0 | SCS | CAP |
CCIE;
}

void dbc(int x){
    volatile int i;
    for (i=0; i<x; i++);
}

void debounce(long lim){
    volatile long c=0;
    while (c++ < lim) ;
}

void I2C_escrever(char dado){
    UCB0I2CSA = 0x3F;
    UCB0CTL1 |= UCTR    |
                UCTXSTT;
    while ( (UCB0IFG & UCTXIFG) == 0);
    UCB0TXBUF = dado;
    while ((UCB0IFG & UCTXIFG) == 0);
    UCB0CTL1 |= UCTXSTP;
    while ( (UCB0CTL1 & UCTXSTP) == UCTXSTP);
    debounce(50);
}

void nibble(char nibble){
I2C_escrever(nibble);
I2C_escrever(nibble | BIT2);
    __delay_cycles(10);
    I2C_escrever(nibble);
    return;
}

void byte(char byte, int end){
    volatile char lsb = byte << 4;
    nibble((byte&0xf0) | end | BIT3);
    nibble((lsb&0xf0) | end | BIT3);
}

void cursor_h(char k){
    byte(0x80+k,0);
}
void cursor_v(char k){
    byte(0xC0+k,0);
}

}


void LCD_config(void){
    __delay_cycles(300000);

    nibble(BIT5 | BIT4);
    __delay_cycles(90000);

    nibble(BIT5 | BIT4);
    __delay_cycles(20000);

    nibble(BIT5 | BIT4);
    __delay_cycles(100);

    nibble(BIT5);
    __delay_cycles(100);

    byte((BIT5|BIT3),0);
    __delay_cycles(100);

    byte(BIT3,0);
    __delay_cycles(100);

    byte(BIT0, 0);
    __delay_cycles(100);

    byte((BIT2|BIT1), 0);
    __delay_cycles(100);
}

void main(void) {

.;    WDTCTL = WDTPW | WDTHOLD;
    config();
    __enable_interrupt();
// configurar portas
    P3SEL |=  BIT0 | BIT1;
    P3REN |=  BIT0 | BIT1;
    P3OUT |=  BIT0 | BIT1;

//configurar I2C
    UCB0CTL1 |= UCSWRST;
    UCB0CTL0 = UCSYNC | UCMST | UCMODE_3 ;
    UCB0CTL1 = UCSSEL_2;
    UCB0BRW = 10;

    LCD_config();
    byte(0xF,0);

    while(1){
//atualizar LCD
    if((P2IN & BIT1)==0){
     TA0CTL |= TACLR;      // ZERO A FLAG
```

```
     TA0CCTL1 = OUTMOD_6; // timer do pino
P1.2 como modo 6 (RESET) seta o trigger pra 1
e zera após atingir TA0CCR1
     tempo=0;
     dbc(1000);}

    if (distancia<10){
     cursor_v(7);
     byte(0x30,0x01);
     if (distancia==0){
      cursor_v(8);
      byte(0x30,0x01);}
     if (distancia==1){
         cursor_v(8);
         byte(0x31,0x01);}
     if (distancia==2){
         cursor_v(8);
         byte(0x32,0x01);}
     if (distancia==3){
         cursor_v(8);
         byte(0x33,0x01);}
     if (distancia==4){
         cursor_v(8);
         byte(0x34,0x01);}
     if (distancia==5){
         cursor_v(8);
         byte(0x35,0x01);}
     if (distancia==6){
         cursor_v(8);
         byte(0x36,0x01);}
     if (distancia==7){
         cursor_v(8);
         byte(0x37,0x01);}
     if (distancia==8){
         cursor_v(8);
         byte(0x38,0x01);}
     if (distancia==9){
         cursor_v(8);
         byte(0x39,0x01);}
    }
    if ((distancia>=10)&&(distancia<20)){
     teste=distancia-10;
         cursor_v(7);
         byte(0x31,0x01);
         if (teste==0){
          cursor_v(8);
          byte(0x30,0x01);}
         if (teste==1){
             cursor_v(8);
             byte(0x31,0x01);}
         if (teste==2){
             cursor_v(8);
             byte(0x32,0x01);}
         if (teste==3){
             cursor_v(8);
             byte(0x33,0x01);}
         if (teste==4){
             cursor_v(8);
             byte(0x34,0x01);}
         if (teste==5){
             cursor_v(8);
             byte(0x35,0x01);}
         if (teste==6){
             cursor_v(8);
             byte(0x36,0x01);}
         if (teste==7){
             cursor_v(8);
             byte(0x37,0x01);}
         if (teste==8){
             cursor_v(8);
             byte(0x38,0x01);}
         if (teste==9){
             cursor_v(8);
             byte(0x39,0x01);}
        }
    if ((distancia>=20)&&(distancia<30)){
     teste=distancia-20;
         cursor_v(7);
         byte(0x32,0x01);
         if (teste==0){
          cursor_v(8);
          byte(0x30,0x01);}
         if (teste==1){
             cursor_v(8);
             byte(0x31,0x01);}
         if (teste==2){
             cursor_v(8);
             byte(0x32,0x01);}
         if (teste==3){
             cursor_v(8);
             byte(0x33,0x01);}
         if (teste==4){
             cursor_v(8);
             byte(0x34,0x01);}
         if (teste==5){
             cursor_v(8);
             byte(0x35,0x01);}
         if (teste==6){
             cursor_v(8);
             byte(0x36,0x01);}
         if (teste==7){
             cursor_v(8);
             byte(0x37,0x01);}
         if (teste==8){
             cursor_v(8);
             byte(0x38,0x01);}
         if (teste==9){
             cursor_v(8);
             byte(0x39,0x01);}
        }
    if ((distancia>=30)&&(distancia<40)){
     teste=distancia-30;
```

```
          cursor_v(7);
          byte(0x33,0x01);
          if (teste==0){
           cursor_v(8);
           byte(0x30,0x01);}
          if (teste==1){
              cursor_v(8);
              byte(0x31,0x01);}
          if (teste==2){
              cursor_v(8);
              byte(0x32,0x01);}
          if (teste==3){
              cursor_v(8);
              byte(0x33,0x01);}
          if (teste==4){
              cursor_v(8);
              byte(0x34,0x01);}
          if (teste==5){
              cursor_v(8);
              byte(0x35,0x01);}
          if (teste==6){
              cursor_v(8);
              byte(0x36,0x01);}
          if (teste==7){
              cursor_v(8);
              byte(0x37,0x01);}
          if (teste==8){
              cursor_v(8);
              byte(0x38,0x01);}
          if (teste==9){
              cursor_v(8);
              byte(0x39,0x01);}
        }
if ((distancia>=40)&&(distancia<50)){
     teste=distancia-40;
          cursor_v(7);
          byte(0x34,0x01);
          if (teste==0){
           cursor_v(8);
           byte(0x30,0x01);}
          if (teste==1){
              cursor_v(8);
              byte(0x31,0x01);}
          if (teste==2){
              cursor_v(8);
              byte(0x32,0x01);}
          if (teste==3){
              cursor_v(8);
              byte(0x33,0x01);}
          if (teste==4){
              cursor_v(8);
              byte(0x34,0x01);}
          if (teste==5){
              cursor_v(8);
              byte(0x35,0x01);}
          if (teste==6){
              cursor_v(8);
              byte(0x36,0x01);}
          if (teste==7){
              cursor_v(8);
              byte(0x37,0x01);}
          if (teste==8){
              cursor_v(8);
              byte(0x38,0x01);}
          if (teste==9){
              cursor_v(8);
              byte(0x39,0x01);}
        }
if ((distancia>=50)&&(distancia<60)){
     teste=distancia-50;
          cursor_v(7);
          byte(0x35,0x01);
          if (teste==0){
           cursor_v(8);
           byte(0x30,0x01);}
          if (teste==1){
              cursor_v(8);
              byte(0x31,0x01);}
          if (teste==2){
              cursor_v(8);
              byte(0x32,0x01);}
          if (teste==3){
              cursor_v(8);
              byte(0x33,0x01);}
          if (teste==4){
              cursor_v(8);
              byte(0x34,0x01);}
          if (teste==5){
              cursor_v(8);
              byte(0x35,0x01);}
          if (teste==6){
              cursor_v(8);
              byte(0x36,0x01);}
          if (teste==7){
              cursor_v(8);
              byte(0x37,0x01);}
          if (teste==8){
              cursor_v(8);
              byte(0x38,0x01);}
          if (teste==9){
              cursor_v(8);
              byte(0x39,0x01);}
        }
if ((distancia>=60)&&(distancia<70)){
     teste=distancia-60;
          cursor_v(7);
          byte(0x36,0x01);
          if (teste==0){
           cursor_v(8);
           byte(0x30,0x01);}
          if (teste==1){
              cursor_v(8);
```

```
            byte(0x31,0x01);}
        if (teste==2){
            cursor_v(8);
            byte(0x32,0x01);}
        if (teste==3){
            cursor_v(8);
            byte(0x33,0x01);}
        if (teste==4){
            cursor_v(8);
            byte(0x34,0x01);}
        if (teste==5){
            cursor_v(8);
            byte(0x35,0x01);}
        if (teste==6){
            cursor_v(8);
            byte(0x36,0x01);}
        if (teste==7){
            cursor_v(8);
            byte(0x37,0x01);}
        if (teste==8){
            cursor_v(8);
            byte(0x38,0x01);}
        if (teste==9){
            cursor_v(8);
            byte(0x39,0x01);}
       }
if ((distancia>=70)&&(distancia<80)){
    teste=distancia-70;
        cursor_v(7);
        byte(0x37,0x01);
        if (teste==0){
         cursor_v(8);
         byte(0x30,0x01);}
        if (teste==1){
            cursor_v(8);
            byte(0x31,0x01);}
        if (teste==2){
            cursor_v(8);
            byte(0x32,0x01);}
        if (teste==3){
            cursor_v(8);
            byte(0x33,0x01);}
        if (teste==4){
            cursor_v(8);
            byte(0x34,0x01);}
        if (teste==5){
            cursor_v(8);
            byte(0x35,0x01);}
        if (teste==6){
            cursor_v(8);
            byte(0x36,0x01);}
        if (teste==7){
            cursor_v(8);
            byte(0x37,0x01);}
        if (teste==8){
            cursor_v(8);
            byte(0x38,0x01);}
        if (teste==9){
            cursor_v(8);
            byte(0x39,0x01);}
       }
if ((distancia>=80)&&(distancia<90)){
    teste=distancia-80;
        cursor_v(7);
        byte(0x38,0x01);
        if (teste==0){
         cursor_v(8);
         byte(0x30,0x01);}
        if (teste==1){
            cursor_v(8);
            byte(0x31,0x01);}
        if (teste==2){
            cursor_v(8);
            byte(0x32,0x01);}
        if (teste==3){
            cursor_v(8);
            byte(0x33,0x01);}
        if (teste==4){
            cursor_v(8);
            byte(0x34,0x01);}
        if (teste==5){
            cursor_v(8);
            byte(0x35,0x01);}
        if (teste==6){
            cursor_v(8);
            byte(0x36,0x01);}
        if (teste==7){
            cursor_v(8);
            byte(0x37,0x01);}
        if (teste==8){
            cursor_v(8);
            byte(0x38,0x01);}
        if (teste==9){
            cursor_v(8);
            byte(0x39,0x01);}
       }
if ((distancia>=90)&&(distancia<100)){
    teste=distancia-90;
        cursor_v(7);
        byte(0x39,0x01);
        if (teste==0){
         cursor_v(8);
         byte(0x30,0x01);}
        if (teste==1){
            cursor_v(8);
            byte(0x31,0x01);}
        if (teste==2){
            cursor_v(8);
            byte(0x32,0x01);}
        if (teste==3){
            cursor_v(8);
            byte(0x33,0x01);}
```

```
            if (teste==4){
                cursor_v(8);
                byte(0x34,0x01);}
            if (teste==5){
                cursor_v(8);
                byte(0x35,0x01);}
            if (teste==6){
                cursor_v(8);
                byte(0x36,0x01);}
            if (teste==7){
                cursor_v(8);
                byte(0x37,0x01);}
            if (teste==8){
                cursor_v(8);
                byte(0x38,0x01);}
            if (teste==9){
                cursor_v(8);
                byte(0x39,0x01);}
          }
    cursor_h(1);
    byte(0x44,0x01);
    cursor_h(2);
    byte(0x49,0x01);
    cursor_h(3);
    byte(0x53,0x01);
    cursor_h(4);
    byte(0x54,0x01);
    cursor_h(5);
    byte(0x41,0x01);
    cursor_h(6);
    byte(0x4E,0x01);
    cursor_h(7);
    byte(0x43,0x01);
    cursor_h(8);
    byte(0x49,0x01);
    cursor_h(9);
    byte(0x41,0x01);
    cursor_h(10);
    byte(0x28,0x01);
    cursor_h(11);
    byte(0x43,0x01);
    cursor_h(12);
    byte(0x4D,0x01);
    cursor_h(13);
    byte(0x29,0x01);
    cursor_h(1);
    cursor_v(1);
    }

}

#pragma vector =TIMER0_A1_VECTOR
    __interrupt void FLAGDOWN(){
    if(TA0IV = 0x4){
        if(tempo == 0){
            tempo = TA0CCR2;
        }else{
            distancia=(TA0CCR2-
tempo)*0.0162124634;
        }
        TA0CCTL1 = OUTMOD_0;

     }
    }
```