Sensor de Estacionamento

Microcontroladores e Microprocessadores

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

***Resumo*— O atual ponto de controle tem como intuito apresentar o refinamento do protótipo funcional do projeto em linguagem C, utilizando as ferramentas, interrupções e sub-rotinas necessárias para o projeto final da disciplina Microprocessadores e Microcontroladores, tendo como tema o Sensor de Estacionamento com visor LCD e indicação luminosa.**

Keywords*— *Microcontrolador; Sensor Ultrassônico; Distância; 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 estacionar um carro por 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 e Indicação Luminosa 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. Esse item, em alguns modelos de carros mais recentes, já vem instalado no veículo, porém ainda hoje muitos não possuem essa ferramenta auxiliando a estacionar.

III. REQUISITOS

Buscando alcançar uma precisa análise espacial e de distância com o sensor ultrassônico, tendo baixo consumo de energia e boa interface gráfica, serão necessários os seguintes requisitos:

A. Descrição do Hardware

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:

- *Protoboard* : plataforma base para montagem dos circuitos;
- LCD Nokia 5110: interface gráfica e visualização de medições/parâmetros de distância.
- Sensor Ultrassônico: HC-SR04;
- *Push Buttons*: funções como *iniciar, medir,* etc;
- Jumpers: conexões.

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

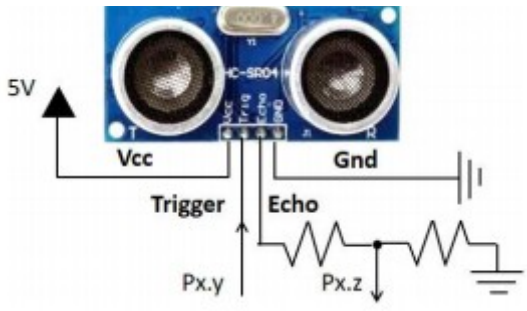


Figura 1 – Sensor ultrassônico de proximidade HC-SR04.

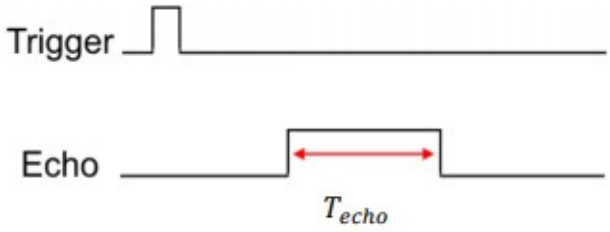


Figura 2 – Resposta no tempo de impulso de iniciação (trigger) e margem de tempo de captação do obstáculo (echo).

Segue abaixo uma visão de montagem do protótipo em protoboard (Figura 3) e esquemático (Figura 4), feita no *software Fritzing*, dos componentes que serão utilizados na montagem do Sensor de Estacionamento com visor LCD e Indicação Luminosa.

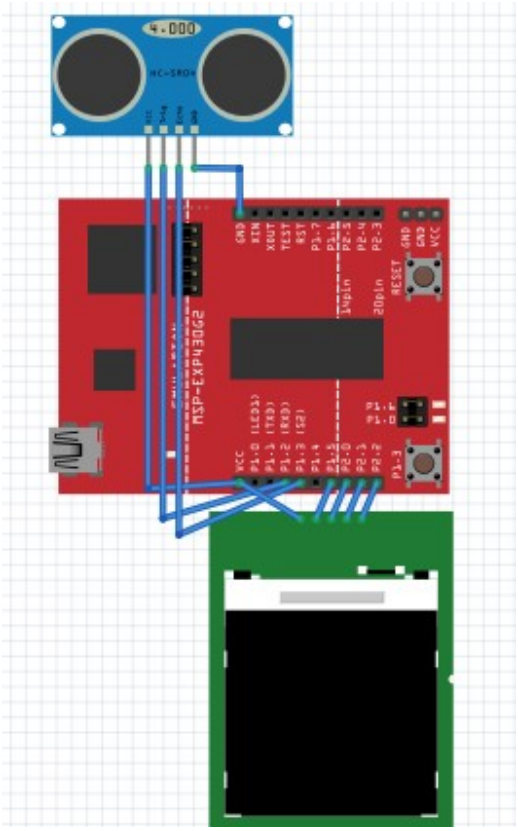


Figura 3 – Visão Protótipo.

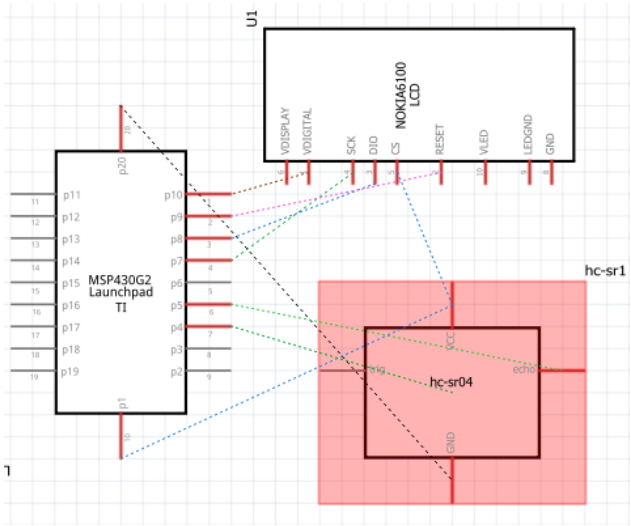


Figura 4 – Visão Esquemático.

O sistema é projetado para operar com o mínimo de consumo de forma constante, mostrando num display a distância em tempo real.

A programação do MSP430 será utilizada 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.

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 led. Nenhum led acende caso a distância seja superior a 1 metro (60cm). Se a distância estiver superior a 20cm e inferior a 60cm, acende o led verde indicando que ainda pode aproximar-se um pouco mais. Se a distância por menor que 20cm, o led vermelho acende indicando para imediata.

O espectro de parâmetros 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.

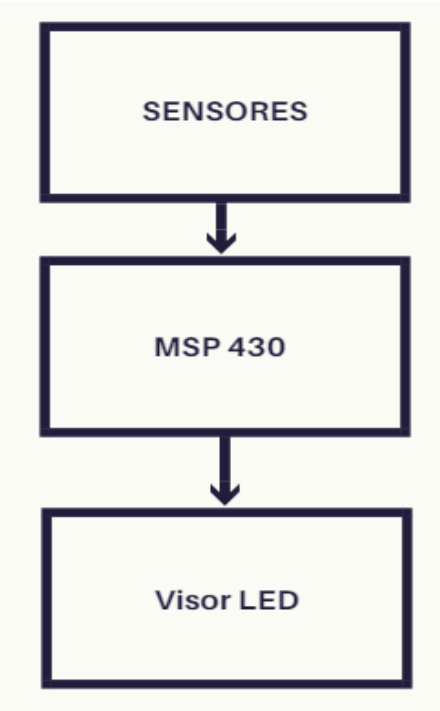
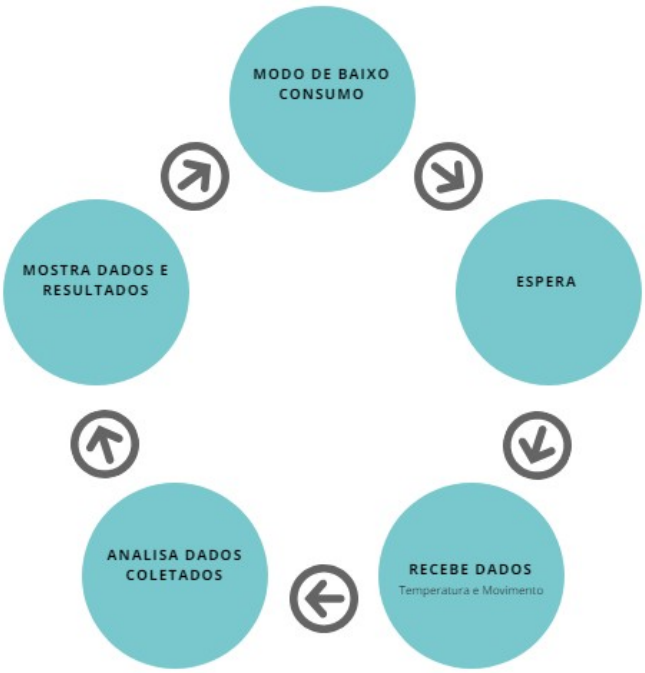


Diagrama de Blocos geral do sistema.

O fluxograma abaixo mostra sequencialmente os estágios que o sistema opera, levando em consideração as características dos componentes do projeto.



Fluxograma de estágios do sistema.

V. Resultados e Conclusões

Os resultados esperados são de um medidor consideravelmente robusto, sem *bugs*, boa aparência, baixo consumo de energia e, principalmente, precisão na finalidade principal de auxílio ao motorista. Desta forma, alcançar a segurança e prevenção contra batidas e amassados.

Os componentes da implementação de *hardware* já foram adquiridos, montados para testes e preparação da primeira versão do protótipo.

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

Por fim fica o entusiasmo de projetar este Sensor de Estacionamento e o otimismo que esse projeto seja um verdadeiro sucesso, mesmo tendo noção dos desafios e complicações que serão observadas no decorrer do projeto até a apresentação final.

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>

volatile int tempo;
volatile int distancia;
int main(void){
    WDTCTL = WDTPW | WDTHOLD;
// Stop watchdog timer
    config();
    __enable_interrupt(); //ativar interrupt
    while(1){
        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);
        }
    }
    return 0;
}

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
    P4DIR |= BIT7; // P4.7 SAIDA
    P1DIR |= BIT0; // P4.7 SAIDA

    // 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++);
}

#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;
        if(distancia < 20){
            P1OUT |= BIT0;
            P4OUT &= ~BIT7;
        }
        if(distancia > 60){
            P4OUT &= ~BIT7;
            P1OUT &= ~BIT0;
        }
        if(distancia >= 20 && distancia <=
60){
            P4OUT |= BIT7;
            P1OUT &= ~BIT0;
        }
    }
  }
```