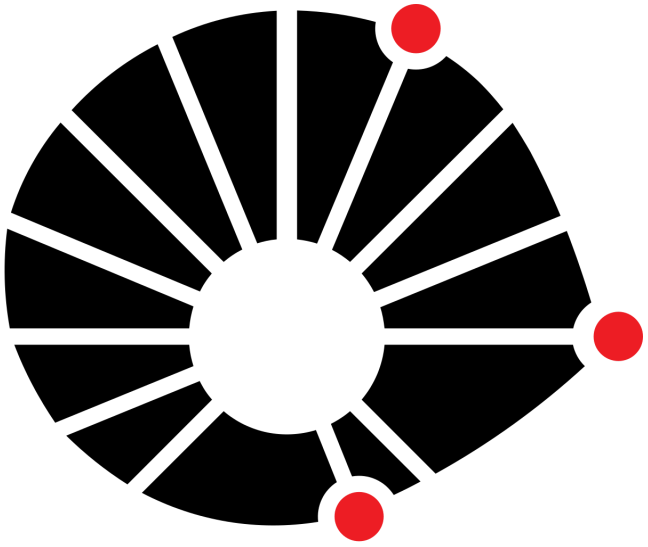
Relatório Projeto Final -EE871



UNICAMP

Campinas-São Paulo
Novembro 2017

Nome dos integrantes do grupo:

Fernando Fortes Granado RA:171575
Rodrigo Diniz Junqueira Santos RA:186755

Turma: M
Disciplina: EA871

Introdução e objetivos:

Nesse projeto final tinha-se como principal objetivo aplicar os conceitos aprendidos ao longo da disciplina e utilizar um dispositivo não utilizado anteriormente, para se fazer um medidor de distâncias semelhantes àqueles presentes em automóveis, com a exceção de que o dispositivo com o qual trabalhou-se era ultra-som e atualmente nos automóveis os sensores de distância no geral utilizam infravermelho para se medir distâncias.

Primeiramente, antes de se iniciar o experimento, realizou-se um esboço de como iria-se fazer para que o sensor funcionasse de forma adequada, pensou-se em utilizar o **tpm** para se medir o tempo que demoraria para o som ir e voltar do sensor de distância. No entanto, após reflexão e pesquisas, viu-se que a velocidade do som se altera dependendo das condições do meio na qual a mesma se propaga, principalmente por conta da variação da temperatura, com isso lembrou-se do módulo **adc** utilizado anteriormente na disciplina, justamente para se realizar a medição de temperatura. Ainda assim, visto que tínhamos como objetivo fazer um projeto com o qual fosse semelhante ao de um sensor de ré de um automóvel, faltava um sensor sonoro, então propôs-se a utilização de um buzzer por meio da utilização de outro módulo **tpm** para se controlar a frequência na qual se liga e desliga o buzzer. Além disso, imprimia-se no lcd a distância medida [1] e também os leds vermelhos seguiram uma lógica semelhante ao dos carros, que seria quanto mais distante maior o número de leds acesos e a medida que o carro for de aproximando os leds iam se apagando.

Portanto, os módulos utilizados e suas funções ficaram da seguinte forma:

- **ADC**: somente leitura de temperatura (ºC).
- **TPM**: contagem de tempo da distância e também controle da frequência na qual o buzzer iria ligar e desligar.
- **PIT**: não foi utilizado neste experimento.
- **LCD**: imprimir a distância medida.
- **LED**: acender e apagar à medida que o carro se aproxima ou se afasta.

Metodologia:

Após a definição dos módulos que seriam utilizados, teve-se que se pesquisar o funcionamento do sensor de distância. Tomou-se como base principalmente o datasheet do fabricante, onde se viu o seguinte diagrama temporal:

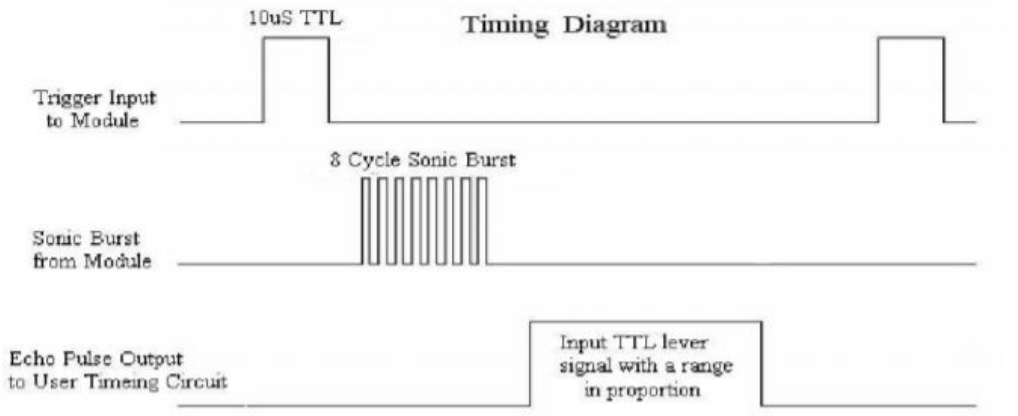


Figura 1.Diagrama de tempo do funcionamento do sensor HC-SR04

Proveniente dessa imagem, entendeu-se a seguinte lógica de funcionamento, que logo após o envio de um trigger de pelo menos 10 μs , isto é, na borda de descida do trigger, será enviado 8 ciclos de som, que assim que o sensor começar a captar de volta o sinal, haverá um sinal na saída que ficará em ativo alto até o término de captação dos sinais. Com esses fatos, viu-se que o tpm teria que gerar interrupção para ambas as bordas tanto de subida quanto de descida, pois o tpm deveria começar a contar, logo com a borda de subida do eco e parar de contar com a borda de descida do eco. Também na borda de descida, o mesmo deveria enviar um novo sinal de trigger iniciando o ciclo novamente.

Além disso, viu-se que constava nas especificações do fabricante que o sensor hc-sr04 somente funciona com uma tensão de 5V, apesar do fato da placa FRDM-KL25Z possuir pinos de alimentação de 5V não se tinha acesso aos mesmos, devido ao shield de EA871, porém teve-se a ideia de se aplicar uma fonte de tensão externa dessa forma solucionando o problema, no entanto tinha-se a disposição somente uma fonte de tensão de 6V, então teve-se a ideia de fazer um divisor de tensão em um protoboard de forma que a entrada fosse de 6V e a saída de 5V.

No uso do TPM, o mesmo seria sensível tanto à borda de subida quanto de descida, pois dessa forma na borda de subida do eco iria capturar o tempo total de viagem até a borda de descida quando será enviado um novo Trigger.Também fizemos uso das seguintes flags do circuito :

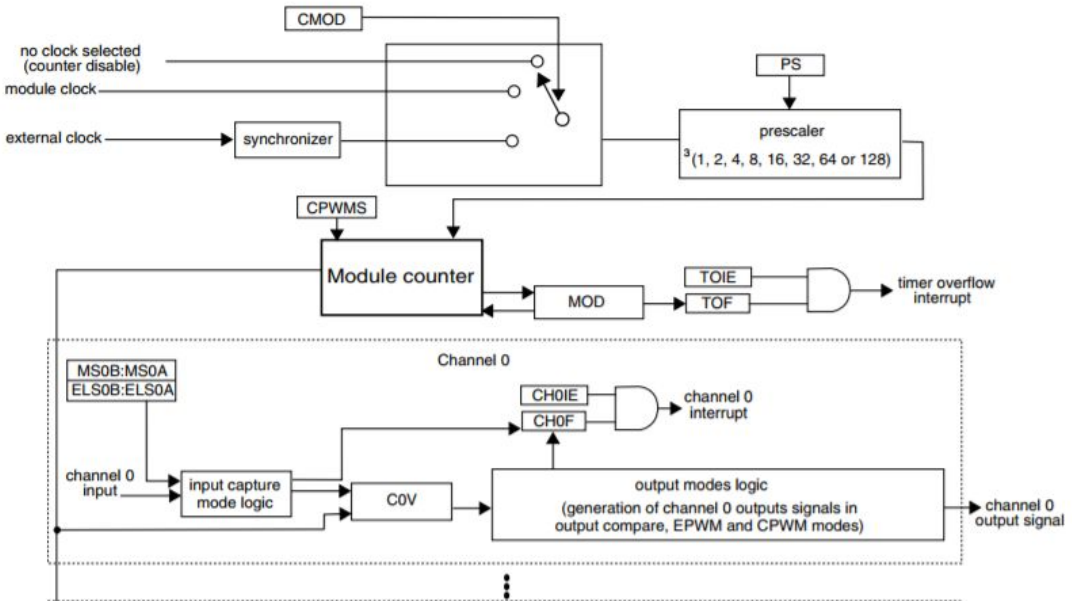


Figura 2.Circuito esquemático TPM

TOIE e TOF que seria toda vez em que se tiver um overflow o TOF levanta-se automaticamente por meio de hardware, e no caso a flag TOIE é setada por meio de software, que no nosso programa seria justamente quando o TPM captura a borda de subida do eco a flag TOIE é habilitada, e toda vez que o TOF for setado uma variável ciclos é incrementada.

observação: [1] inicialmente era realmente à medida que o sensor media o valor era impresso no lcd, no entanto para se deixar de forma a diminuir o erro optou-se por pegar a média de 220 medidas.

Proposta de solução:

Tabela 1: configuração de cada pino usado e suas funções no projeto

Pino	Configuração	MUX	descrição
PTE21	TPM1_CH1	0b011*	controle do buzzer: ligá-lo e desligá-lo
PTE22	GPIO saída	0b001	enviar sinal de trigger
PTE23	TPM2_CH1	0b011	contagem do tempo do eco
PTB1	ADC0 - canal 9	0b000	medição da temperatura

*Para ligar o buzzer MUX = 0b011, e para desligar o buzzer MUX = 0b000. Assim se fazia o bip-bip.

Após o envio do trigger o sensor envia um sinal, a onda de eco, que fica em nível lógico 1 pelo tempo equivalente à ida e volta do ultrassom emitido. Assim, para medir o tempo do eco configuramos PTE23 para ser TPM2_CH1 input capture em ambas as bordas. Após a borda de subida desse sinal, na handler de TPM2_CH1, habilitamos o campo TOIE do TPM2, para incrementar uma contagem de ciclos quando houver overflow no contador de TPM2. E quando há a borda de descida do eco desabilitamos esse campo TOIE e calculamos o tempo equivalente ao eco (tempo de viagem), por meio do qual calculamos a distância usando, também, a velocidade do som (dada por *velocidadeAR = 33130 + (60 * temperatura)* em centímetros/s)

Após calcular a distância, ela é armazenada no vetor distancias_recentes[i]. Após serem juntadas 220 distâncias, é calculada a média e usamo-na para calcular o número de ciclos de TPM em que o buzzer deve ficar ligado e desligado: quanto mais perto menor esse tempo.
Foi usada a fórmula: ciclos_desligado = (aux_dist_media/200)/0.000391.
Após esse número de ciclos, executamos
setaMux(&PORT_PCR_REG(PORTE_BASE_PTR,21), 0b011);
para ligar o buzzer ou
setaMux(&PORT_PCR_REG(PORTE_BASE_PTR,21), 0b000);
para desligar o buzzer.

Com a média das 220 distâncias, também ligamos e desligamos os LEDs vermelhos: acima de 1m de distância todos ligados; abaixo de 1m, a cada 12,5cm mais perto, um LED é desligado.

Pseudocódigo:

main

```
Inicialização:
        chama initPort(B)
        configura PTB1 como ADC
        chama initPort(E)
        configura PTE21 como TPM; PTE22 como GPIO; PTE23 como TPM
        configura PTE22 como saída
        inicializa TPM2 com pré-escala 8 e MOD=16
        inicializa TPM1 com pré-escala 8 e MOD = 1024
        inicializa TPM2 CH1 como input capture em ambas as bordas
        inicializa TPM1 CH1 como output compare
        inicializa LEDs apagados
        inicializa LCD
        inicializa ADC e habilita interrupção
        habilita interrupção do TPM2
        habilita interrupção do TPM1
        chama mandaString("Distancia  [cm]");

manda trigger
repetir (loop infinito):
        se(flag_ADC0)
                atualizar temperatura
                flag_ADC0 = 0
        se(flag_novo_tempo)
                velocidadeAR = 33130 + (60 * temperatura)
                distancias_recentes[i] = (velocidadeAR*tempoviagem)/2
                i++
                se(i == 220)
                        calcular media das 220 distancias
                        atualizar frequencia do buzzer
                        i = 0
                        flag_print_dist = 1;
        se(flag_print_dist)
                         escreve a media das distancias recentes no LCD; apaga
                LEDs vermelhos conforme menor distancia (todos acesos para mais
                de 1m)
```

Rotina de TPM2_CH1 (onda de eco)

```
subida (começo do eco):
        habilita interrupcao no overflow
        ciclos = 0


descida (fim do eco):
         desabilita interrupcao no overflow
         tempoviagem = ciclos*0.0000061
         flag_novo_tempo = 1
         enviar outro trigger (delay)

overflow (apenas se a interrupção por overflow estiver habilitada):
         ciclos++
```

Rotina de TPM1_CH1 (controla o buzzer)

```
aux_buzzer++;
/*! variavel que incrementa para contar o tempo em que o buzzer ficara ligado e
desligado */

se (aux_buzzer >= ciclos_desligado)
        desliga buzzer se estiver ligado; liga buzzer se estiver desligado
                /* isto é feito alterando o MUX de PTE21, onde está conectado o
                buzzer */
        aux_buzzer = 0
```

Testes:

Assim que vimos que nosso sensor de distâncias estava funcionando da forma esperada, utilizamos uma placa plana de madeira com dimensões inferiores requeridas pelo fabricante para realizar alguns testes, e obtivemos os seguintes resultados:

Validação da medida de distância:

Tabela 2: Distância real entre o sensor e o anteparo e distância medida pelo sensor

Medida sensor (cm)	Medida trena (cm)
62.41	65.5
44.31	51
49.81	55.5
31.11	34.2
23.3	27.5
72	80
17.21	19.5

Também realizamos algumas outras medidas que foram capturadas tanto pelo osciloscópio, como distância da mesa do le30 até o teto da sala, e obtivemos os seguintes resultados:

Medidas distância até o teto: 205 cm,que está de acordo visto que o sensor estava sobre uma mesa que tinha altura de aproximadamente 70 cm

ciclos	436	0x1ffff038
tempoviagem	0.0119072	0x1ffff034
ciclos_desligado	2625	0x1ffff004
ciclos	436	0x1ffff038
tempoviagem	0.0119072	0x1ffff034
ciclos_desligado	2625	0x1ffff004

$$\Rightarrow Distância = velocidade\ do\ som\ no\ meio * tempo\ de\ viagem/2$$

$$\Rightarrow Distância = 205,8\ cm$$

Que é condizente com a realidade visto que geralmente a altura de um andar é por volta de 2,8m até 3m se levarmos em conta a altura da mesa teremos que a altura do andar da sala de aula é por volta de 2,8m

Além disso também realizamos outra medida:

flag_print_dist	' '	0x20002fdf
i	'□'	0x20002fde
j	'd'	0x20002fdd
aux_dist_media	85.4021	0x20002fd8
distancias_recentes	0x20002e24	0x20002e24
flag_novo_tempo	0	0x1ffff02c
ciclos	761	0x1ffff038
tempoviagem	0.0054839	0x1ffff034

Figura: tempo de viagem medido pelo programa: 0.0054839s

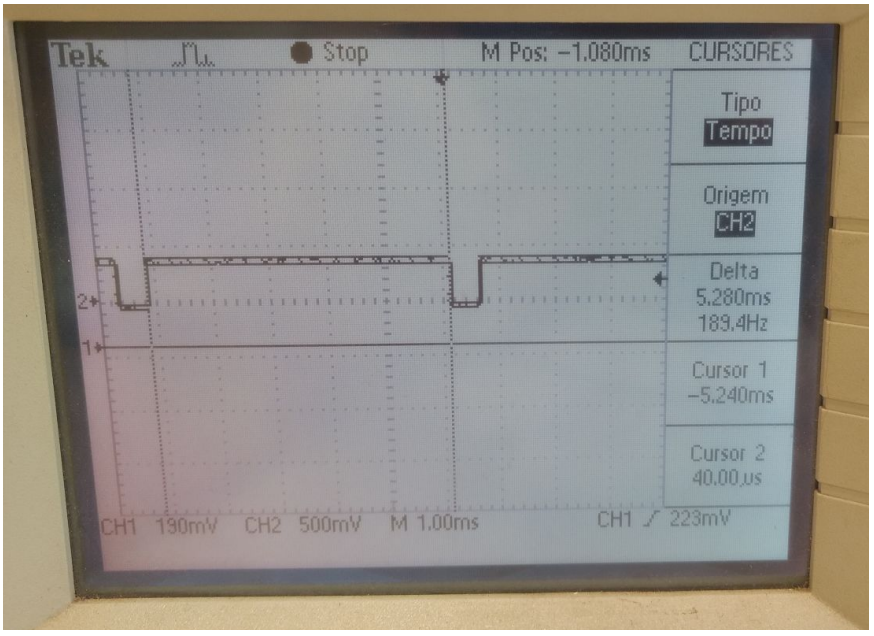


Figura: Onda do eco vista no osciloscópio; o tempo em nível 1 é de 5.280ms

Como vemos pelas duas imagens acima, o tempo do eco foi medido em 5.280ms pelo osciloscópio e em 5.4839ms pelo programa. Essas medidas são bem próximas, levando a concluir que o tempo está sendo corretamente medido pelo programa.

Validação do buzzer:

Figura: onda em PTE21 quando o tempo de viagem é 0.0053985s, conforme a figura abaixo:

flag_novo_tempo	0	0x1ffff02c
ciclos	885	0x1ffff038
tempoviagem	0.0053985	0x1ffff034
ciclos_desligado	1268	0x1ffff004
ciclos	885	0x1ffff038
tempoviagem	0.0053985	0x1ffff034
ciclos_desligado	1268	0x1ffff004

Este tempo de viagem equivale a 99.30 cm de distância e o período do buzzer (medido no osciloscópio) é de 990ms.

Por outro lado, jogando na fórmula teremos que o número de ciclos teria que ser por volta de 1270 ciclos (pois a distância é 99.3cm) e obtivemos :

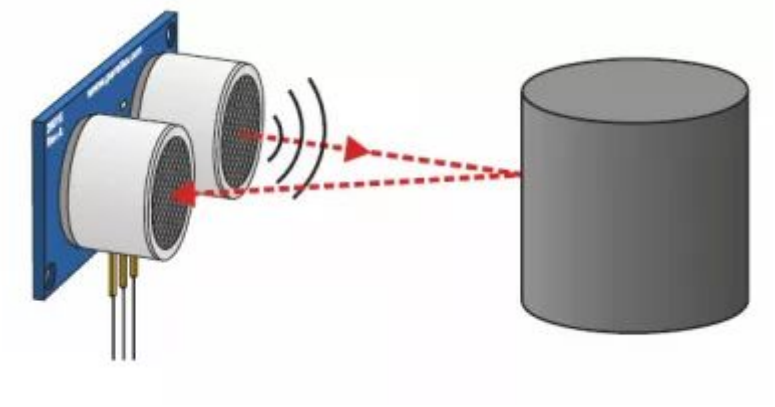
1268*0.000391 = 0,4965s que é o tempo desligado e no caso o período ligado e desligado são iguais, logo o período total será de 0.991s que se aproxima muito daquele

medido pelo osciloscópio. Desse modo, vemos que o bip-bip do buzzer está funcionando adequadamente.

Conclusão:

Nesse experimento, pode-se notar alguns fatores como possíveis fontes de erros, como:

- Não tínhamos como fazer o experimento em uma sala isolada, isto é objetos ao redor poderiam ser possíveis fontes de erro.
- De acordo com o fabricante é necessária uma placa/área com uma dimensão de aproximadamente 0,5m^2, entretanto não tinha-se disponível tal placa com essas dimensões.
- Acredita-se que a principal fonte de erro seria , ângulo de operação do sensor que de acordo com fabricante teria que ser por volta de 15^o no entanto,não tinha-se como garantir esse ângulo. Também, pode-se afirmar que a existem desse ângulo é em decorrência do modo como o mesmo opera. Que pode-se ser visto abaixo:



- Além disso, tem-se fatores como não conseguia-se manter a placa de madeira com a qual se realizou o experimento à uma distância constante, e ângulo constante.

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Figura 3. Especificações do fabricante entre elas, a principal o ângulo de medida.

Portanto, visto que mediu-se por meio do osciloscópio e de software o tempo de distância percorrido, e os mesmo eram iguais, pode-se afirmar que as fontes de erro não eram em decorrência do software, mas sim por conta de erros físicos como apresentados anteriormente, e devido ao fato de limitações ou ausência de materiais esses erros não puderam ser minimizados. Como no caso, se tivesse uma bolha de nível, transferidor, podia-se minimizar o erro cometido por conta do ângulo.

Além disso, pode-se afirmar que erros de eficiência ou na diferença de tempo contado pelos ciclos e realmente realizados foram minimizados ou até mesmo eliminados a partir do momento em que empregou-se o módulo tpm, que seu objetivo é justamente ser o mais preciso possível na contagem de tempo.

Referências:

1-http://www.micropik.com/PDF/HCSR04.pdf
2-ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/AmbienteDesenvolvimentoHardware.pdf
3-https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico-distancia/
4-ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KLQRUG.pdf
5-ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
6-http://www.dca.fee.unicamp.br/cursos/EA871/2s2016/UW/roteiros/relatorio_template.txt