

# Ponto de Controle 4

## Assimilação Sonora para Portadores de Deficiência Auditiva

Marcos Adriano Nery de Abrantes  
15/0062567  
Faculdade do Gama  
Universidade de Brasília  
St. Leste Projeção A - Gama Leste,  
Brasília - DF, 72444- 240  
Email: marcosadrianonery@gmail.com

Pedro Helias Carlos  
14/0158278  
Faculdade do Gama  
Universidade de Brasília  
St. Leste Projeção A – Gama Leste,  
Brasília – DF, 72444 – 240  
Email: pedroheliass95@hotmail.com

**RESUMO** – Este documento apresenta o ponto de controle III do projeto final para a disciplina de Microprocessadores e Microcontroladores.

**Palavras-chave** — MSP 430; Frequências Sonoras; Controle de tensão; Microprocessador;

### I. INTRODUÇÃO

A tecnologia deve ser acessível a todas as massas sociais, porém a falta de acessibilidade e reconhecimento torna o conceito de inclusão bastante debilitado. Para isso, é de total importância que tecnologias sejam produzidas visando este conceito social, o de inclusão, além de que é possível, e por vezes maior, o avanço nos conceitos eletrônicos dado a importância e o desafio que é projetar tecnologias para portadores de deficiência.

### II. OBJETIVOS

Tornar possível e viável a aproximação de pessoas deficientes auditivas à música através de uma pulseira vibratória, o usuário poderá sentir a intensidade da música, que sofrerá FFT para definir melhor os pontos a serem trabalhados pela MSP e o circuito auxiliar acoplado a protoboard.

Além do mais, será implantando um sistema de cores (a partir de uma matriz de LED's) indicando quando cada frequência for atingida, o que dará singularidade a cada frequência sonora sintonizada as notas existentes. Além de tornar mais visível ao usuário, o processo de singularidade de cada nota musical tornará mais próxima a relação “humano-máquina”.

### III. DESCRIÇÃO

Com o intuito de fazer a descrição de sinais sonoros a partir de um vibracall, para auxílio de portadores de deficiência, será construído um circuito assimilador de frequências sonoras. Este circuito será capaz de compreender e separar tensões segundo cada frequência sonora e demonstrar tal sinal a partir da vibração, daí a importância da separação de tensões.

Para que a operação acima funcione corretamente, é necessário utilizar alguns componentes e funções que serão especificados a seguir:

### HARDWARE

#### • Sensor de Som KY-038 Microfone

O sensor de Som KY-038 é um módulo que comporta um circuito de detecção sonora e um microfone de eletreto como parâmetro para detecção. O objetivo deste sensor é medir a intensidade sonora ao redor de seu alcance. A medida que o sensor afere a intensidade sonora, ele configura variações para sua saída digital caso ele detecte algum sinal.

Sua pinagem é relativa aos terminais analógico e digital, além do Ground e Vcc, responsáveis pela alimentação do circuito.

O circuito será alimentado pela tensão da própria MSP e aterrado igualmente. Inicialmente a proposta seria utilizar um circuito amplificador com AmpOp's depois de uma tentativa de se utilizar um circuito com transistores. Visto que a alimentação estaria presente de forma mais acessível, ocorreria por carga externa. Com o módulo, a alimentação pode ser direcionada diretamente da placa.

Os sinais digitais e analógicos sairão do módulo com o sinal captado pelo microfone e assim a informação chegará a placa.

O circuito de controle do microfone de eletreto, atualmente descartado, se dava pela construção de um circuito transistorizado a fim de se construir um modelo referente a um amplificador, contendo capacitores e resistores. Mas o mesmo foi descartado, pois fora utilizado apenas em fase de testes. O componente oficialmente a ser utilizado está presente abaixo:

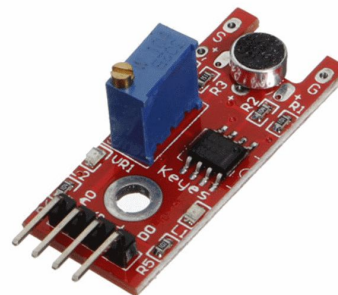


Figura 01. Sensor de áudio

#### • Motor vibracional

Este motor, é um dispositivo construído em um eixo metálico. Sua composição se dá por um eixo em formato de meia-lua. Sua polaridade é definida a partir de quais dos terminais são interconectados com o Vcc e o Gnd, ocasionando apenas a rotação em sentidos diferentes. A medida que ocorre a rotação do eixo, naturalmente há a oscilação do motor. Já que é controlado por tensão, tal componente foi selecionado tanto pela clara coerência com a proposta do projeto, fácil manipulação quanto facilidade em encontrar tais motores, estando os mesmos presentes em celulares mais antigos ( os utilizados pelo projeto se enquadra em motores vibracionais de celulares antigos).



Figura 02. Vibracall

Ao aplicar certa tensão, o motor rotaciona a determinadas RPM, referentes a valores pré selecionados via código. A medida que se aumenta a tensão aplicada, a rotação fica mais e mais rápida, a ponto de ser perceptível a mudança. Facilmente compactua-se com o projeto, visto que o controle de tensão pode ser realizado e a percepção da mudança de velocidade é claro.

Na ocorrência de sinais sonoros externos, o módulo vai operar e tratar o sinal, a fim de transmitir o sinal. Logo após o software compara o sinal e o direciona a tensão desejada e a vibração ocorre.

- **Display Nokia 5110**



figura 03. Display Nokia 5110

Foi utilizado um display Nokia 5110 para apresentação dinâmica do projeto, da identificação dos integrantes e título do projeto. A adição desse display foi considerada a partir da máxima inserção dos conhecimentos absorvidos na disciplina. Em uma definição secundária, o display exibe a intensidade sonora obtida, diretamente correlacionada as rotações do motor vibracall.

A codificação bastante simplificada dada por um modelo, inicializa-se a partir da biblioteca “PCD8544.h”, onde estão contidas as funções utilizadas. Os caracteres serão descritos no Display a partir de strings, e com o comando “write\_\_\_\_ToLCD”, sendo o espaço descrito apto a receber String, Char, Block, Graphic. Cada parâmetro determinando uma função específica.

## SOFTWARE

Acerca do código, destaca-se as entradas e saídas da MSP, como mencionado acima terá um sensor de entrada

responsável por captar o áudio do ambiente e torná-lo reconhecível ao microcontrolador por meio de uma tensão, essa tensão é analógica, logo, torna-se necessário o uso do conversor AD, assim, a porta selecionada para a conversão foi a, PORTA P1.1, correspondente ao BIT1. Sendo possível atestar na figura, a seguir.

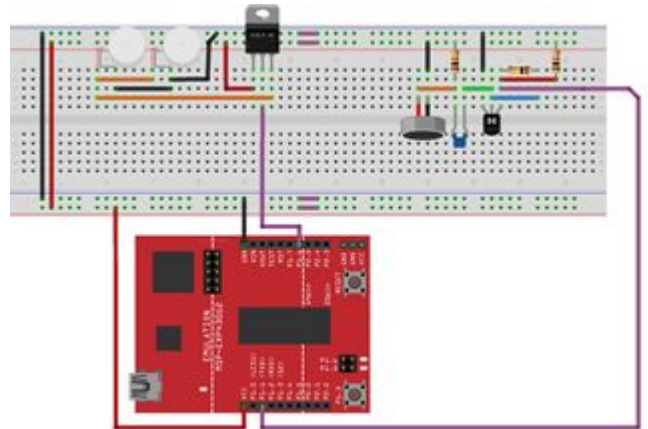


Figura 04. Esquemático do circuito associado

Considera-se antes de continuar a seguinte figura, acerca, do esquema de montagem do circuito externo na MSP.

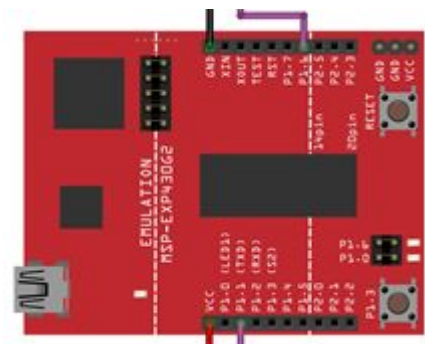


Figura 05. Vista superior MSP

A figura acima representa o circuito a ser montado, percebe-se os referidos anteriormente dispositivos de hardware. Considerando a próxima figura, uma aproximação das portas da MSP. Onde se tem o fio roxo, na PORTA P1.1, está a entrada vinda da saída do sensor de áudio. Em VCC e GND, estão os pinos responsáveis por alimentar o circuito. Tem-se agora a saída da MSP, a mesma é configurada como saída do comparador do TIMER1, funcionando como um PWM, onde, o ciclo de trabalho altera-se segundo a entrada do conversor AD. Segue o esquema da saída, na PORTA P1.6.

```
19 // Saída do comparação
20 //PINO 1.6
21 P1DIR |= LED2;
22 P1SEL |= LED2;
23 P1SEL2 &= ~LED2;
```

Figura 05. Definição saída do comparação

Como especificado no USER GUIDE, a disposição

foi de forma que a tornar o pino da PORTA P1.1, a saída do comparador, sendo por esta a saída do PWM. Assim foi-se determinado o valor do registrador do TIMERA, responsáveis pela saída do comparador, segue.

```
27 TACCR0 = 1024-1;
56 TACCR1 = ADC10MEM-1;
```

Figura 07. Registradores TIMERA

O registrador TACCR0, determina até quanto o TAR do TIMERA, deve contar e o TACCR1, determina o ciclo de trabalho, sendo esse determinado pelo último valor lido pelo conversor AD, estando da forma apresentada um total de 1024 possíveis níveis de ciclo de trabalho, atualizado a partir de uma interrupção. A interrupção é apresentada a seguir:

```
45 #pragma vector = TIMER0_A1_VECTOR
46 __interrupt void TA0_ISR(void)
47 {
48     // Muda o ciclo de trabalho
49     // O CICLO DE TRABALHO É ALTERADO DE ACORDO
50     // COM O VALOR PRESENTE NA ENTRADA ANALOGICA
51     // O NIVEL FOI DEFINIDO ACIMA DE UMA MARGEM
52     if(ADC10MEM > 10)
53     {
54         //TACCR1 DEFINE CICLO DE TRABALHO
55         // DEFINIR VALOR EM TACCR1
56         TACCR1 = ADC10MEM-1;
57     }
58
59     // RESETAR BIT DE INTERRUPTÃO DO TIMER
60     TA0CTL &= ~TAIFG;
61     // HABILITAR LEITURA DO DA ENTRADA A/D
62     ADC10CTL0 |= ENC + ADC10SC;
63
64 }
```

Figura 08. Estrutura do interrupção

Acima percebe-se que a interrupção é acionada a partir do TIMERA, sendo atualizada de forma a atualizar o registrador TACCR1, que segundo o modo de comparação selecionado RESET/SET (OUTMOD\_7), determina o ciclo de trabalho, assim se o valor lido pelo conversor AD, fosse ADCMEM = 512, ter-se-ia um ciclo de trabalho de 50%. Outro ponto importante a se atentar é que o código roda no mais baixo consumo de energia do MSP, como segue:

```
41 _BIS_SR(LPM0_bits + GIE);
```

Figura 08. Baixo consumo

Foi-se ao fim adicionado um display, com a intenção de realizar uma comunicação acerca do projeto com o usuário. O display utilizado como mencionado acima foi o Nokia-5111, o código de implementação do mesmo segue em anexo junto com a necessária biblioteca para o uso do mesmo.

Os demais detalhes estão comentados no código presente em anexo.

#### IV. RESULTADOS

Com o intuito de verificar a viabilidade do projeto pretendido pela dupla, alguns testes foram feitos com a

MSP430. Montando-se o circuito pretendido da figura 9.

O mesmo apresentou todas as considerações necessárias, apresentando em sua saída um valor de ciclo de trabalho diretamente disposto ao valor de entrada no conversor AD. Foi-se montado ainda em protoboard um protótipo do que será o projeto, e de fato alcançou-se o que era-se esperado. E utilizando-se o código mostrado no Anexo A, verificou-se o funcionamento do controle de velocidade dos motores vibracionais através da entrada AD da MSP. Conforme mostrado no código, a saída PWM possui 1024 diferente ciclos de trabalho, limitados pela entrada AD de 10 bits. Como adições posteriores, verifica-se adição dos motores em uma pulseira para melhor interação entre o usuário.

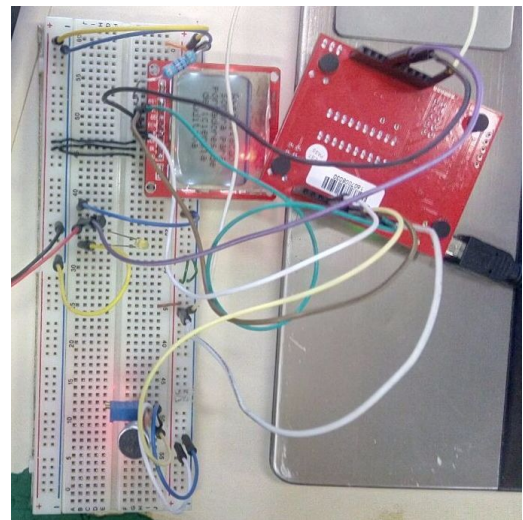


Figura 09. Projeto apresentado no ponto de controle

```
#include "msp430g2553.h"
#include "PCD8544.h"
#include <msp430.h>
#include <math.h>
#include <complex.h>
#define IN_AD BIT2
#define IN_AD_CH INCH_2 //A2
#define LED1 BIT0
#define LED2 BIT6
#define LEDS (LED1|LED2)
#define LCD5110_SCLK_PIN BIT5
#define LCD5110_DN_PIN BIT7
#define LCD5110_SCE_PIN BIT0
//SAIDA
#define LCD5110_DC_PIN BIT1
//SAIDA
#define LCD5110_SELECT P1OUT &= ~LCD5110_SCE_PIN
#define LCD5110_DESELECT P1OUT |= LCD5110_SCE_PIN
#define LCD5110_SET_COMMAND P1OUT &=
~LCD5110_DC_PIN
#define LCD5110_SET_DATA P1OUT |= LCD5110_DC_PIN
#define LCD5110_COMMAND 0
#define LCD5110_DATA 1
#define SPI_MSB_FIRST UCB0CTL0 |= UCSB // or UCA0CTL0
|= UCSB (USCIA) or USICTL0 &= ~USILSB (USI)
#define SPI_LSB_FIRST UCB0CTL0 &= ~UCMSB // or
UCA0CTL0 &= ~UCMSB or USICTL0 |= USILSB (USI)
void writeStringToLCD(const char *string);
void writeCharToLCD(char c);
void writeBlockToLCD(char *byte, unsigned char length);
void writeGraphicToLCD(char *byte, unsigned char transform);
void writeToLCD(unsigned char dataCommand, unsigned char data);
void clearLCD();
void clearBank(unsigned char bank);
void setAddr(unsigned char xAddr, unsigned char yAddr);
void initLCD();
int converte(int binario[]);
```

```

void Send_Int(int n);
unsigned char currXAddr = 0; //TODO this will be used for tracking
current addr
unsigned char currYAddr = 0; //not implemented
char testBlock[8] = {0x00, 0x7F, 0x7F, 0x33, 0x33, 0x03, 0x03,
0x03};
char testBlock2[8] = {0x00, 0x18, 0x18, 0x18, 0x7E, 0x3C, 0x18,
0x00};

void main(void) {
//PARAR WATCH DOG TIMER
WDTCTL = WDTPW + WDTHOLD;
// Clock 1MHZ
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;
// Saida do comparação
//PINO 1.6
P1DIR |= LED2;
P1SEL |= LED2;
P1SEL2 &= ~LED2;
// Configura o canal 1 do Timer A em modo de comparacao
// com periodo do tatal do "AD", 1024 partes, 10 bits
TACCR0 = 1024-1;
//MOD0 COMPARADOR
TACCTL1 = OUTMOD_2;
// REGISTRADOR TACTL
//TAIE HABILITA INTERRUPTÃO DO TIMER
TACTL = TASSEL_2 | ID_3 | MC_1 + TAIE;
ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON;
// ADC10SHT_3, espera 16x adc10clks.
ADC10AE0 = IN_AD;
// Com SHS_1, uma conversao sera requisitada
// sempre que o canal 1 do Timer_A terminar sua contagem
ADC10CTL1 = IN_AD_CH + SHS_0 + ADC10DIV_0 +
ADC10SSEL_3 + CONSEQ_0;
ADC10CTL0 |= ENC + ADC10SC;
//=====

P1OUT |= LCD5110_SCE_PIN + LCD5110_DC_PIN;
P1DIR |= LCD5110_SCE_PIN + LCD5110_DC_PIN;
// setup USIB _ HABILITANDO COMUNICAÇÃO SERIAL _
"BIT0 E BIT1"
P1SEL |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;
P1SEL2 |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;
UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC; //
3-pin, 8-bit SPI master
UCB0CTL1 |= UCSSEL_2; // SMCLK
UCB0BR0 |= 0x01; // 1:1
UCB0BR1 = 0;
UCB0CTL1 &= ~UCSWRST; // clear SW
//=====

_delay_cycles(500000);
initLCD();
clearLCD();
//=====

// VOLTAR AO INICIO
setAddr(0, 0);
// TITULO
clearBank(1);
writeStringToLCD(" Ponto de ");
clearBank(3);
writeStringToLCD(" controle 4 ");
_delay_cycles(2000000);
//=====
//=====

setAddr(0, 0);
clearLCD();
// LCD Apresentação
clearBank(0);
writeStringToLCD(" Prof: Diogo ");
clearBank(2);
writeStringToLCD(" Matriculas: ");
clearBank(3);
writeStringToLCD("> 14/0158278 ");
clearBank(5);
writeStringToLCD("> 15/0062567 ");

```

```

_delay_cycles(5000000);
//=====

setAddr(0, 0);
clearLCD();
// LCD Apresentação
clearBank(0);
writeStringToLCD(" Assimilacao ");
clearBank(1);
writeStringToLCD(" sonora para ");
clearBank(2);
writeStringToLCD("portadores de ");
clearBank(3);
writeStringToLCD(" deficiencia ");
clearBank(4);
writeStringToLCD(" auditiva ");
_delay_cycles(5000000);

//=====
// CHAMAR INTERRUÇÃO
//=====

_BIS_SR(LPM0_bits + GIE);
}
//=====
// FIM_MAIN
//=====
// INTERRUPTÃO TIMER
//=====
#pragma vector = TIMER0_A1_VECTOR
__interrupt void TA0_ISR(void)
{
// Muda o ciclo de trabalho
// O CICLO DE TRABALHO É ALTERADO DE ACORDO
// COM O VALOR PRESENTE NA ENTRADA ANALOGICA
// O NIVEL FOI DEFINIDO ACIMA DE UMA MARGEM
if(ADC10MEM > 0)
{
//TACCR1 DEFINE CICLO DE TRABALHO
// DEFINIR VALOR EM TACCR1
TACCR1 = ADC10MEM-1;
}
//=====
//=====

// RESETAR BIT DE INTERRUPTÃO DO TIMER
TA0CTL &= ~TAIFG;
// HABILITAR LEITURA DO DA ENTRADA A/D
ADC10CTL0 |= ENC + ADC10SC;
}

//=====
// write STRING To LCD
//=====

void writeStringToLCD(const char *string) {
while(*string) {
writeCharToLCD(*string++);
}
}

//=====
// write CHAR ToLCD
//=====

void writeCharToLCD(char c) {
unsigned char i;
for(i = 0; i < 5; i++) {
writeToLCD(LCD5110_DATA, font[c - 0x20][i]);
}
writeToLCD(LCD5110_DATA, 0);
}
//=====
// write To LCD
//=====

void writeToLCD(unsigned char dataCommand, unsigned char data)

```

```

{
LCD5110_SELECT;
if(dataCommand) {
LCD5110_SET_DATA;
} else {
LCD5110_SET_COMMAND;
}
UCB0TXBUF = data;
while(!(IFG2 & UCB0TXIFG));
LCD5110_DESELECT;
}
//=====
//                                CLEAR LCD
//=====

void clearLCD() {
setAddr(0, 0);
int c = 0;
while(c < PCD8544_MAXBYTES) {
writeToLCD(LCD5110_DATA, 0);
c++;
}
setAddr(0, 0);
}
//=====
//                                CLEAR BANK
//=====

void clearBank(unsigned char bank) {
setAddr(0, bank);
int c = 0;
while(c < PCD8544_HPIXELS) {
writeToLCD(LCD5110_DATA, 0);
c++;
}
setAddr(0, bank);
}

//=====
//                                SET ADDR
//=====

void setAddr(unsigned char xAddr, unsigned char yAddr) {
writeToLCD(LCD5110_COMMAND, PCD8544_SETXADDR |
xAddr);
writeToLCD(LCD5110_COMMAND, PCD8544_SETYADDR |
yAddr);
}

//=====
//=====
//
INIT LCD
//=====
//=====

void initLCD() {
writeToLCD(LCD5110_COMMAND, PCD8544_FUNCTIONSET |
PCD8544_EXTENDEDINSTRUCTION);
writeToLCD(LCD5110_COMMAND, PCD8544_SETVOP | 0x3F);
writeToLCD(LCD5110_COMMAND, PCD8544_SETTEMP | 0x02);
writeToLCD(LCD5110_COMMAND, PCD8544_SETBIAS | 0x03);
writeToLCD(LCD5110_COMMAND, PCD8544_FUNCTIONSET);
writeToLCD(LCD5110_COMMAND,
PCD8544_DISPLAYCONTROL | PCD8544_DISPLAYNORMAL);
}

//=====
//                                INTEIRO
//=====

void Send_Int(int n)
{
int casa, dig;
if(n==0)
{
//Send_Data('0');
writeCharToLCD(0X30);

```

```

return;
}
if(n<0)
{
//Send_Data('-');
writeCharToLCD(0X2d);
n = -n;
}
for(casa = 1; casa<=n; casa *= 10);
casa /= 10;
while(casa>0)
{
dig = (n/casa);
//Send_Data(dig+'0');
writeCharToLCD(dig + 0X30);
n -= dig*casa;
casa /= 10;
}
}
}

```

## REFERÊNCIAS

- [1] Davies, J., MSP430 Microcontroller Basics, Elsevier, 2008.
- [2] Apostila: Oficina seguidor de linha. Vieira, Gabriel Meneses.
- [3] McRoberts, Michael. Arduino Básico. [tradução Rafael Zanolli]. São Paulo: Novatec Editora, 2011.