UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DISCIPLINA DE SISTEMAS MICROCONTROLADOS

FELIPE UKAN PEREIRA, HUDO CIM ASSENÇO

BATERIA DE PERCUSSÃO ELETRÔNICA

RELATORIO FINAL DE PROJETO

CURITIBA

2015

FELIPE UKAN PEREIRA, HUDO CIM ASSENÇO

BATERIA DE PERCUSSÃO ELETRÔNICA

Relatorio Final de Projeto apresentado a Disciplina de Sistemas Microcontrolados do curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná demonstrando o processo de desenvolvimento do projeto.

Orientador: Heitor Silvério Lopes

CURITIBA

2015

RESUMO

. BATERIA DE PERCUSSÃO ELETRÔNICA. 13 f. Relatorio Final de Projeto – Disciplina de Sistemas Microcontrolados, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

Este trabalho apresenta o processo de desenvolvimento de uma bateria de percussão eletrônica, englobando a explicação do código feito em *assembly* e também dos *hardwares* desenvolvidos para o projeto. A batéria em questão contém apenas dois módulos. Cada módulo possui um sensor piezoelétrico capaz de detectar a intensidade de uma batida possibilitando o microcontrolador 8051 tocar o som correspondente ao módulo da batida.

Palavras-chave: Bateria eletrônica, ADC, DAC, 8051

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este documento apresenta o desenvolvimento do projeto final da matéria de sistemas microcontrolados de 1/2015. O projeto consiste no desenvolvimento de uma bateria de percussão eletrônica de dois módulos apenas. Para a deteção de batidas foi utilizado um sensor piezoelétrico em cada módulo. O sinal elétrico oriundo do sensor é devidamento amplificado e filtrado para a utilização deste com o conversor analógico digital ADC0832, presente na placa de desenvolvimento com 8051 utilizada na disciplina.

Após a deteção da batida e sua intensidade, o microcontrolador envia o sinal digital referente ao som a ser produzido para um conversor digital analógico R2R já desenvolvido para práticas anteriores da disciplina. O sinal analógico é então direcionado para as caixas amplificadoras de áudio. A figura 1 apresenta um diagrama de blocos com a visão geral do projeto.

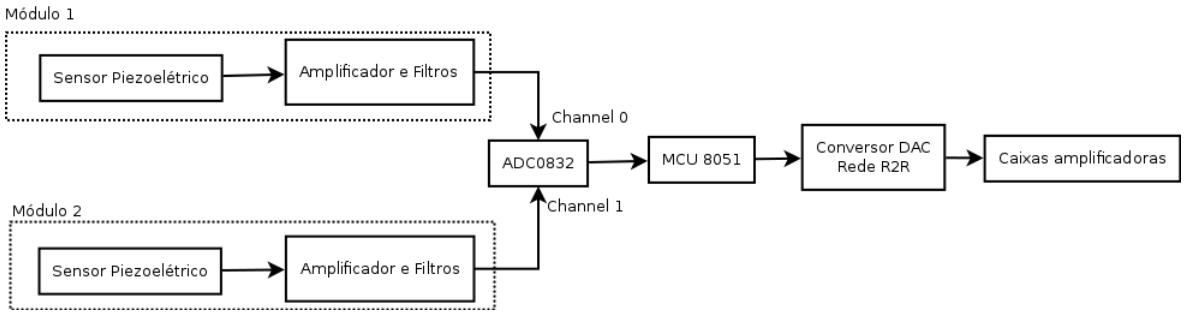

Figura 1: Diagrama de blocos do projeto desenvolvido.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Ter uma bateria de percussão eletrônica funcional que emita um som específico para uma batida, levando em consideração a intensidade da batida.

1.1.2 OBJETIVOS ESPECÍFICOS

- tratar o sinal do sensor de entrada com um ADC.
- tratar o sinal de saída para uma caixa de som com um DAC.
- tratar as amostras dos sinais coletados pelos 2 piezoselétricos no microcontrolador do kit didático da matéria (programa de tratamento em *assembly*).
- gerar um som coerente com a intensidade da batida de entrada.

2 DESENVOLVIMENTO

2.1 *HARDWARE*

O sinal de entrada vem dos sensores piezoelétricos, estes dispositivos possuem a capacidade de gerar tensão elétrica a partir de uma pressão mecânica. No entanto, a tensão elétrica oriunda de uma batida possui natureza senoidal e valores elevados para a utilização de microcontroladores.

A figura 2 apresenta o esquemático do circuito utilizado para tratar este sinal. Este circuito tem como objetivo permitir o controle da sensibilidade do sensor a partir do potênciometro $RV1$, assim como eliminar qualquer componente DC e negativo através do diodo $D1$ e filtro passa-alta formado pelos componentes $C1$ e $R4$. Após isto, o sinal é amplificado e passado por um filtro passa-baixa formado pelos componentes $C2$ e $R3$ fazendo com que o sinal mantenha um nível de tensão proporcional a intensidade da batida. O diodo zenner de 4.7V foi utilizado para a segurança do CI conversor AD que possui um limite de 5V no canal de entrada.

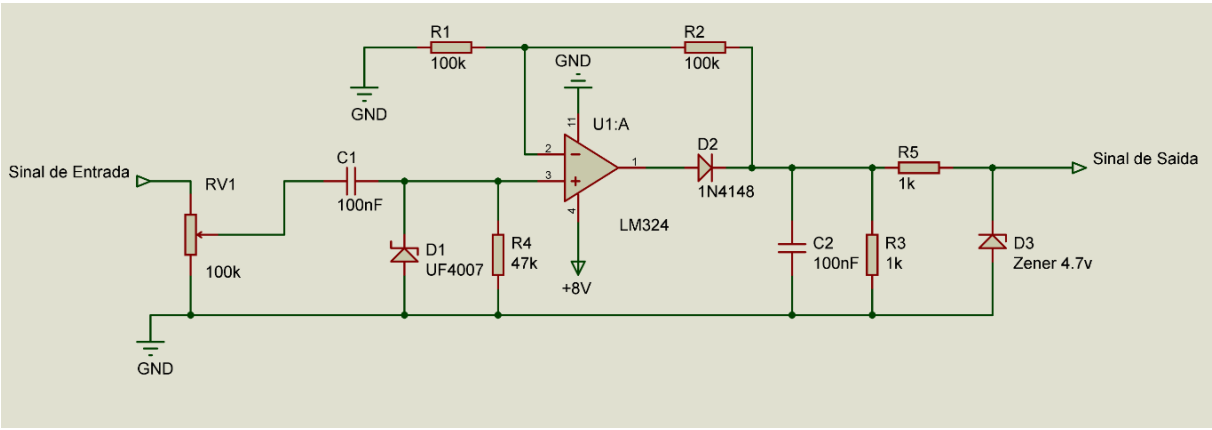

Figura 2: Esquemático do *hardware* que trata o sinal de entrada.

A figura 3 apresenta o circuito utilizado para converter o sinal digital de 8 bits gerado pelo 8051 para um sinal analógico utilizado pelas caixas amplificadoras para tocar o som.

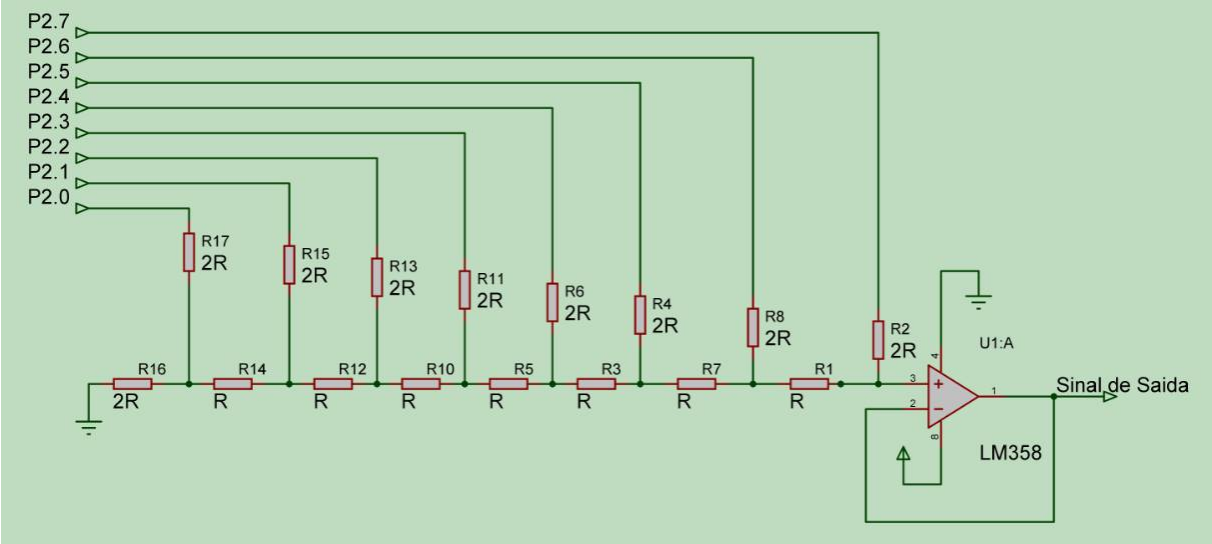

Figura 3: Esquemático do *hardware* conversor digital analógico.

2.2 *SOFTWARE*

Em geral, o objetivo do *software* é: dado um sinal de entrada obtido pelo ADC da placa didática, tocar um determinado som. O som, em um formato *.wav*, é convertido para uma frequência de amostragem de 22kHz e uma resolução de 8 bits em um computador. Após a conversão é gerado uma tabela com as amostras do som que é armazenada na memória de programa do 8051.

Para determinar o som a ser tocado é levado em consideração a intensidade da batida, sendo que uma batida mais forte gera um som mais alto e uma batida mais fraca um som mais baixo. A definição de uma batida forte e fraca, para gerar os sons alto e baixo, é feita por uma constante definida empiricamente.

Neste processo também é necessário definir o numero de amostrar, obtidas do ADC, que definem uma batida. Ou seja, batidas muito rapidas podem ser consideradas como uma. O ajuste fino desta definição também foi feito empiricamente. Outro valor importante é o valor mínimo que é considerado com uma batida, para evitar que ruidos gerem sons.

Após inicializadas as váriaveis na *main*, o *loop* obtem as amostras do ADC do canal 0 (CH0) e em seguida do canal 1 (CH1), deste modo o código explicado a seguir é válido para ambos os canais. As váriaveis de cada canal são zeradas após 200 amostras, sendo que este contador é inicializado a partir da primeira amostra que ultrapassa o valor de *threshold* (limite para evitar ruído), exceto a váriavel de intensidade que recebe o valor máximo da amostra que posteriormente é utilizada para definir se o som a ser tocado será alto ou baixo. O trecho de código a seguir mostra como está sendo feito isto para o canal 1:

```
MOV SAMPLE_COUNTER_CH1, #00h
MOV FLAG_HIT_CH1, #00h
MOV INTENSITY_CH1, PICO_CH1
MOV INDEX_TABLE_CH1_L, #00h
MOV INDEX_TABLE_CH1_H, #00h
MOV PICO_CH1, #00h
```

O pico (valor máximo) de uma amostra é detectado em cada canal dentro de um numero finito de amostras (200) substituindo o valor anterior pelo maior. Assim, enquanto o numero de amostrar é menor que 200 e foi detectado uma batida (sinal maior que o SAMPLE_TRESHOLD) o valor de pico é atualizado. O trecho de código a seguir apresenta como o pico está sendo detectado:

```
detecta_pico_SAMPLE_CH0:
        CLR C
        MOV A, SAMPLE_CH0
        SUBB A, #SAMPLE_TRESHOLD ; verifica se a amostra atual
        ; e menor que o threshold definido, se for menor retorna
        ; e se for maior segue para setar a flag que um som devera
        ; ser tocado
        JNC pega_pico_CH0
        RET

pega_pico_CH0:
; seta flag que tem sinal
        MOV FLAG_HIT_CH0, #0FFh ; flag que indica que um som devera
        ; ser tocado
        CLR C
        MOV A, PICO_CH0
        SUBB A, SAMPLE_CH0
        JC pega_pico_CH0_1
        RET

pega_pico_CH0_1:
        MOV PICO_CH0, SAMPLE_CH0
        RET
```

Os sons são gerados na interrupção por tempo do timer 0 a 22KHz (`MOV TH0, #0A5h`). Dentro da interrupção, antes de percorrer a tabela para o som ser

gerado, é verificada a variavel de intensidade, se houver algo diferente de 0 os indices da tabela são incrementados, caso contrário não há a necessidade de tocar o som. Assim, se a tabela já terminou de ser percorrida as váriaveis que apotam para a tabela são zeradas, assim como a intensidade do som e o valor a ser tocado no canal. O código a seguir mostra como isto é feito para o canal 0, para o canal 1 o código é similar mudando apenas algumas *labels*:

```
seta_som_CH0:

        MOV A, INTENSITY_CH0 ; verifica se algum som devera
        JZ seta_som_CH1 ; ser tocado

        INC INDEX_TABLE_CH0_L
        MOV A, INDEX_TABLE_CH0_L
        JNZ verifica_limite_index_CH0 ; verifica o limite da tabela

        INC INDEX_TABLE_CH0_H

verifica_limite_index_CH0:
        ; os limites da tabela sao verificados aqui, se ela terminou
        ; de ser percorrida os valores sao zerados, caso contrario
        ; vai para seta_som_na_var_CH0
        MOV A, INDEX_TABLE_CH0_L
        CLR C
        SUBB A, #TABLE_CRASH_SIZE_L ; verifica limite Low
        JC seta_som_na_var_CH0

        MOV A, INDEX_TABLE_CH0_H
        SUBB A, #TABLE_CRASH_SIZE_H ; verifica limite High
        JC seta_som_na_var_CH0

        ; reseta as variaveis
        MOV INDEX_TABLE_CH0_H, #00h
        MOV INDEX_TABLE_CH0_L, #00h
        MOV INTENSITY_CH0, #00h
        MOV SOUND_CH0, #00h
        JMP seta_som_CH1
```

Caso a tabela não tenha atingido o seu limite, ela será percorrida e sera verificado se a batida é forte ou fraca, para que o som mais alto ou mais baixo sejam tocados. Se for fraco, o

valor a ser escrito na porta de saída será dividido por 2. O código a seguir mostra como a tabela é percorrida e também como é feita a definição de um som alto ou baixo.

```
seta_som_na_var_CH0:
        ; percorre a tabela
        MOV DPTR, #TABLE_CRASH
        CLR C
        MOV A, DPL
        ADD A, INDEX_TABLE_CH0_L
        MOV DPL, A
        MOV A, DPH
        ADDC A, INDEX_TABLE_CH0_H
        MOV DPH, A

        ; verifica se a intensidade da batida e maior que o limite
        CLR C
        MOV A, INTENSITY_CH0
        SUBB A, #HARD_HIT_TRESHOLD
        JC seta_som_na_var_CH0_1 ; se for menor vai para
        ; seta_som_na_var_CH0_1

        MOV A, #00h
        MOVC A, @A + DPTR
        MOV SOUND_CH0, A ; salva o valor a ser escrito na porta

        JMP seta_som_CH1

seta_som_na_var_CH0_1:
        MOV A, #00h
        MOVC A, @A + DPTR

        CLR C
        RRC A ; divide por 2

        MOV SOUND_CH0, A ; salva o valor a ser escrito na porta
```

Por fim, o som a ser tocado deverá ser uma mistura dos sons dos dois canais. Os sons dos dois canais são divididos por 2, para não estourar o tamanho do byte, e então somados para gerar o som final, como apresentado no código a seguir:

```
toca_som_final:
        MOV A, SOUND_CH0
        CLR C
        RRC A
        MOV SOUND_CH0, A

        CLR C
        MOV A, SOUND_CH1
        RRC A

        ADD A, SOUND_CH0

        MOV SOUND_PORT, A
```

3 CONCLUSÃO

Ao fim do projeto obtivemos uma bateria que tem uma resposta no tempo razoável, ou seja, não é possível perceber grandes problemas no aúdio de saída. Existe grande possibilidade de ampliar este projeto, uma de nossas ideias futuras é multiplexar os módulos para colocarmos mais piezos para fazer sons diferentes utilizando o mesmo ADC da placa. Também é possível utilizar mais piezos em um único módulo a fim de se obter ondas de entrada mais precisas.

Outrossim, os objetivos espcíficos foram cumpridos e o projeto está funcional. O custo para o projeto foi relativamente pequeno, aproximadamente 50,00 reais, o que pode caracterizar uma bateria eletrônica de baixo custo. Por fim, o custo benefício de se usar um *hardware* simples mas com o conhecimento obtido na matéria possibilitou uma grande valorização do produto final.