UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI

PROJETO FINAL

ANDRÉ CASADEI MARQUES - 2019010640
GUSTAVO PRIETO ROCHA - 2019013802

ELTD13
Laboratório de Microcontroladores e Microprocessadores 1

ITAJUBÁ

2022

INTRODUÇÃO

O teclado musical a ser projetado, deverá ser capaz de fornecer duas oitavas musicais, com as frequências demonstradas na tabela 1. As chaves SW1 e SW2 irão controlar a oitava que está ativa. O display LCD irá informar, na primeira linha, a oitava ativa e as chaves de SW5 a SW17 irão fornecer as notas musicais. A chave SW3 será utilizada para ajustar o ciclo de trabalho do sinal quadrado enviado ao buzzer, respeitando os valores predeterminados, sendo eles de 25%, 50% e 75%.

O display LCD irá informar na segunda linha o valor de ciclo de trabalho atual e o potenciômetro irá ser usado aumentando a frequência da nota tocada. Por fim, nenhum som deve ser emitido se nenhuma chave estiver pressionada e também não será possível formar sons com mais de uma chave pressionadas ao mesmo tempo.

CÁLCULOS

As frequências de cada nota em cada oitava foram fornecidas na Tabela 1.

Tecla	Primeira oitava [Hz]	Segunda oitava [Hz]	Nota musical
SW5	132	264	C
SW13	140	280	C#
SW6	148	296	D
SW14	157	314	D#
SW7	166	332	E
SW8	176	352	F
SW15	187	374	F#
SW9	198	396	G
SW16	209	418	G#
SW10	222	444	A
SW17	235	470	A#
SW11	249	498	B
SW12	132	280	C

Tabela 1 - Frequência das notas musicais a serem utilizadas.

Para o cálculo das frequências geradas no PWM, tem-se:

$$f_{onda} = \frac{f_{clk}}{(ARR+1)*(PSC+1)} \Rightarrow (ARR + 1)(PSC + 1) = \frac{f_{onda}}{f_{clk}}$$

$$\Rightarrow PSC = \frac{f_{clk}}{(ARR+1)*(f_{onda})} - 1$$

Utilizando ARR = 99, temos por exemplo:

$$PSC_{SW5} = \frac{72MHz}{(132Hz \cdot 100)} - 1 = 5454$$

Assim, os resultados podem ser vistos na Tabela 2.

Tecla	Primeira oitava		Segunda oitava	
	ARR	PSC	ARR	PSC
SW5	99	5454	99	2726
SW13	99	5142	99	2570
SW6	99	4865	99	2431
SW14	99	4565	99	2292

SW7	99	4336	99	2168
SW8	99	4090	99	2044
SW15	99	3849	99	1924
SW9	99	3635	99	1817
SW16	99	3444	99	1721
SW10	99	3242	99	1621
SW17	99	3063	99	1531
SW11	99	2891	99	1445
SW12	99	5454	99	2570

Tabela 2 - Resultados obtidos para o valor de PSC.

CONFIGURAÇÕES

Inicialmente é importante mencionar que foi desenvolvida uma biblioteca para a implementação das funções de configuração do LCD onde também foram implementadas as funções de delay. Esta foi chamada de lcd.

No programa principal chamado main, inclui-se as bibliotecas e são feitas as definições dos pinos utilizados:

```
#include "stm32f10x.h"
#include "lcd.h"

//teclas
#define SW1 12
#define SW2 13
#define SW3 14
#define SW4 15
#define SW5 5
#define SW6 4
#define SW7 3
#define SW8 3
#define SW9 4
#define SW10 8
#define SW11 9
#define SW12 11
#define SW13 10
#define SW14 7
#define SW15 15
#define SW16 14
#define SW17 13

//perifericos
#define BUZZER 0
#define POTENCIOMETRO 1
```

Definição dos valores das notas:

```
27  //notas
28  #define C     132
29  #define Csust 140
30  #define D     148
31  #define Dsust 157
32  #define E     166
33  #define F     176
34  #define Fsust 187
35  #define G     189
36  #define Gsust 209
37  #define A     222
38  #define Asust 235
39  #define B     249
```

Em seguidas as funções desenvolvidas para o projeto são declaradas, junto das variáveis de controle usadas.

```
41  //funcoes desenvolvidas
42  void inicializar(void);
43  void som(uint16_t nota);
44  void timbre(void);
45  void print(uint16_t valor);
46  void tecla(uint32_t GPIOA_IDR, uint32_t GPIOB_IDR, uint32_t GPIOC_IDR);
47  void att_ciclo(uint32_t GPIOB_IDR);
48  void att_pot(void);
49  void att_lcd(void);
50  void clear(void);
51
52  //variaveis de controle
53  uint8_t oitava = 1;
54  uint8_t ciclo = 25;
55  uint32_t pot = 0;
56
```

É implementada a função main em seguida, com o objetivo de inicializar o dispositivo, e entrar num loop infinito para a leitura do valor do potenciômetro e verificação das teclas para emissão do som.

```
57  int main() {
58    inicializar();
59
60    while(1) {
61      att_pot();
62      tecla(~(GPIOA->IDR), ~(GPIOB->IDR), ~(GPIOC->IDR));
63    }
64  }
65
```

A função de inicialização configura os clocks, GPIOs, entradas e saídas analógicas e o LCD.

```
void inicializar(void) {
  //desativar JTAG
  RCC -> APB2ENR |= RCC_APB2ENR_AFIOEN ;
  AFIO -> MAPR |= AFIO_MAPR_SWJ_CFG_JTAGDISABLE ;

  //Ativação dos clocks
  RCC->APB2ENR |= 0xFC | (1<<9);
  RCC->APB1ENR |= (1<<1);

  //Configuração do GPIOA
  GPIOA->CRL = 0x43344444 ; //lcd output
  GPIOA->CRH = 0x34433443 ; //o resto input

  //Configuração do GPIOB
  GPIOB->CRL = 0x3344430B; //analog input PA1 e PB0 push-pull
  GPIOB->CRH = 0x44444444; //input SW1~SW7 e SW10~SW13

  //Configuração do GPIOC
  GPIOC->CRH = 0x44433333; //input PC15 PC14 e PC13

  //Configuração do ADC
  ADC1->CR2 = 1;      // ADON
  ADC1->SMPR2 = 1<<3; //SMP1

  //Configuração do buzzer
  TIM3->CCMR2 = 0x0060;  //configura pwm
  TIM3->CCER = 0x1 << 8; //CC3P=0 e CC3E=1
  TIM3->PSC = 10-1;

  //Configuração do LCD
  lcd_init();
  att_lcd();
}
```

Uma função para verificar as teclas foi implementada, disparando o som relativo à frequência da tecla pressionada.

```
void tecla(uint32_t GPIOA_IDR, uint32_t GPIOB_IDR, uint32_t GPIOC_IDR) {
  att_ciclo(GPIOB_IDR);                              //verifica mudanca de ciclo
  if(GPIOB_IDR & (1<<SW5))        som(oitava * C);   //verifica teclas
  else if(GPIOB_IDR & (1<<SW13))  som(oitava * Csust);
  else if(GPIOB_IDR & (1<<SW6))   som(oitava * D);
  else if(GPIOA_IDR & (1<<SW14))  som(oitava * Dsust);
  else if(GPIOB_IDR & (1<<SW7))   som(oitava * E);
  else if(GPIOA_IDR & (1<<SW8))   som(oitava * F);
  else if(GPIOC_IDR & (1<<SW15))  som(oitava * Fsust);
  else if(GPIOA_IDR & (1<<SW9))   som(oitava * G);
  else if(GPIOC_IDR & (1<<SW16))  som(oitava * Gsust);
  else if(GPIOB_IDR & (1<<SW10))  som(oitava * A);
  else if(GPIOC_IDR & (1<<SW17))  som(oitava * Asust);
  else if(GPIOB_IDR & (1<<SW11))  som(oitava * B);
  else if(GPIOB_IDR & (1<<SW12))  som(oitava * C);
  else                            TIM3->CCR3 = 0;
}
```

A função de cálculo da frequência do PWM e do timer baseado no valor da frequência.

```
void som(uint16_t nota) {
  //calculo de ARR
  TIM3->ARR = (72000000/((nota + (pot/25))*(TIM3->PSC+1)))-1;
  TIM3->CCR3 = ((TIM3->ARR+1)*ciclo)/100;
  TIM3->CR1 = 1;  //inicia timer
}
```

Em seguida, a função que recebe o valor do potenciômetro, e realiza a conversão analógico digital e salva o valor em uma variável de controle.

```
125 void att_pot(void) {
126     ADC1->SQR3 = 9; //canal 9 input
127     ADC1->CR2 = 1;  //conversao
128     while((ADC1->SR & (1<<1)) == 0); //aguarda flag
129     pot = ADC1->DR; //salva valor
130 }
131
```

Depois, a função que controla a atualização dos ciclos e das oitavas, também chamando a função que atualiza o display.

```
132 void att_ciclo(uint32_t GPIOB_IDR) {
133   if(GPIOB_IDR & (1<<SW1)) { //primeira oitava
134     oitava = 1;
135     att_lcd();
136   }
137
138   else if(GPIOB_IDR & (1<<SW2)) { //segunda oitava
139     oitava = 2;
140     att_lcd();
141   }
142
143   if(GPIOB_IDR & (1<<SW3)) { //mudar ciclo
144     ciclo += 25;
145     if(ciclo > 75) ciclo = 25;
146     att_lcd();
147   }
148 }
149
```

Por fim, a função que atualiza o display e outras funções, para limpar o display e printar os dígitos.

```
150 void att_lcd(void) {
151     clear();
152     lcd_print("Oitava: ");
153     print(oitava);
154     lcd_command(0xC0); //pula linha
155     lcd_print("Ciclo: ");
156     print(ciclo);
157 }
158
159 void print(uint16_t valor) {
160     lcd_data((valor/100)%10 + 0x30); //centena
161     lcd_data((valor/10)%10 + 0x30);  //dezena
162     lcd_data(valor%10 + 0x30);       //unidade
163 }
164
165 void clear(void) {
166     lcd_command (0x01); //limpa
167     lcd_command (0x02); //volta linha 1
168 }
169
```