UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO DE PROCESSOS INDUSTRIAIS

ALEXSSANDRO BRAZ DE SOUZA
DEISE MARTINELLO

SISTEMA DE CONTROLE DE TEMPERATURA USANDO FPGA

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2013

ALEXSSANDRO BRAZ DE SOUZA

DEISE MARTINELLO

SISTEMA DE CONTROLE DE TEMPERATURA USANDO FPGA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Automação de Processos Industriais da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. André Macário Barros.

PATO BRANCO

2013

TERMO DE APROVAÇÃO

O Trabalho de Conclusão de Curso intitulado **“SISTEMA DE CONTROLE DE TEMPERATURA USANDO FPGA”**, dos acadêmicos **ALEXSSANDRO BRAZ DE SOUZA** e **DEISE MARTINELLO** foi considerado **APROVADO** de acordo com ata de defesa nº **145** de **2013**, a qual encontra-se na coordenação do curso.

Fizeram parte da Banca os Professores

ANDRE MACARIO BARROS (ORIENTADOR)

FÁBIO LUIZ BERTOTTI

SANTO TIVEROLI FILHO

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de nossas vidas. Portanto, desde já pedimos desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do nosso pensamento e de nossa gratidão.

Primeiramente queremos a agradecer a Deus por tudo o que ele tem nos proporcionado. Contudo, nos dando uma família maravilhosa que sempre nos incentivou a estudar e a seguir nossos ideais. Gostaríamos de deixar registrado também, o nosso reconhecimento pelo carinho, amor e compreensão de nossos conjugues. Reverencio o Professor André Macário Barros pela sua dedicação e pela orientação deste trabalho que em momento algum mediu esforços em dedicar seu tempo à nossas duvidas e dificuldades, por meio dele, eu me reporto a toda a comunidade da Universidade Tecnológica Federal do Paraná (UTFPR) pelo apoio incondicional.

RESUMO

Um controlador de temperatura tem por finalidade manter uma temperatura constante, dentro de um intervalo predeterminado de valores. Seguindo estas definições, foi implementado um circuito digital em VHDL que permite o usuário interagir através de FPGAs com componentes discretos. A ferramenta utilizada é o *ISE 12.4*, da *Xilinx*, onde consta um conjunto de programas que operam de forma integrada para que se possa implementar sistemas digitais com um kit de lógica reconfigurável. A computação reconfigurável baseia-se em dispositivos lógicos reprogramáveis que podem atingir um desempenho elevado e, ao mesmo tempo, fornecer a flexibilidade da programação a nível de portas lógicas. O sistema constitui de um circuito com componentes discretos, do qual tem como função receber a leitura do sensor, interagindo com o código VHDL sintetizado para realizar o controle de temperatura.

Palavras-chave: Controle de temperatura, FPGA, VHDL, computação reconfigurável.

ABSTRACT

A temperature controller aims to maintain a constant temperature, within a predetermined range of values. Following these definitions, a digital circuit has been implemented in VHDL that allows the user to interact with FPGAs using discrete components. The tool used is the ISE 12.4, *Xilinx*, which is a set of programs which operate in an integrated way so that can implement digital systems with a kit of reconfigurable logic. The reconfigurable computing is based on reprogrammable logic devices that can achieve high performance and at the same time providing the flexibility of describing systems at gate level. The system consists of a circuit with discrete components, which has the function of receiving the sensor voltage, interacting with the synthesized VHDL code to perform the control of the temperature.

Keywords: Temperature control, FPGA, VHDL, reconfigurable computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS E ABREVIATURAS

A/D	Analógico para Digital
ASIC	*Application Specific Integrated Circuit*
BCD	*Binary Code Decimal*
CI	Circuito Integrado
CPLD	*Complex Programmable Logic Devices*
FPGA	*Field-Programmable Gate Arrays*
IEEE	*Institute of Electrical and Electronic Enginee*
I/O	*In/Out*
IRR	*Infinite Impulse Response*
LED	*Light Emitting-Diode*
PLC	*Programmable Logic Controller*
PLD	*Programmable Logic Devices*
ROM	*Read-Only Memory*
VCC	Tensão Contínua
VHDL	*Very-High Speed Integrated Circuit Hardware Description Language*

SUMÁRIO

1. INTRODUÇÃO

A automação Industrial existe desde os tempos remotos da pré-história, com a invenção da roda, para a otimização de processos e a redução do esforço humano. Com o surgimento das linhas de montagem a automação alavancou e desde então, não parou mais de evoluir. Notou-se a necessidade da melhoria dos processos e o aumento da produtividade das indústrias, iniciando-se assim o desenvolvimento de máquinas para a realização das tarefas (MEIRA, 2008).

Com o aperfeiçoamento da eletrônica, sugiram os primeiros computadores industriais. Com o surgimento dos microcomputadores, ampliaram-se as possibilidades, e os mesmos já eram utilizados para o aumento da produção e redução de gastos, ligar, desligar e movimentar máquinas, sinalizar defeitos e até mesmo gerar relatórios.

Com este avanço, em meados da década de 1980, foram fabricados os CPLD (*Complex Programmable Logic Devices*), uma espécie de *chip* com esquema de roteamento sofisticado com tecnologia de silício mais avançada e várias características adicionais, o que o tornou muito popular devido a sua densidade relativamente alta, alto desempenho e baixo custo. Na mesma década, foram lançadas as FPGAs (*Field-Programmable Gate Arrays*), diferentes de CPLDs em arquitetura, tecnologia, características embutidas e custos, voltadas principalmente a projetos complexos e de alto desempenho (PEDRONI, 2010).

Considerando-se custo, lentidão, inflexibilidade e a curta duração da vida útil dos computadores, sentiu-se a necessidade da criação de uma técnica em que sua aplicação fosse atrativa e viável ao circuito lógico programável, ou seja, a computação reconfigurável. A computação reconfigurável baseia-se em dispositivos lógicos reprogramáveis que podem atingir um desempenho elevado e, ao mesmo tempo, fornecer a flexibilidade da programação a nível de portas lógicas. O conceito de computação reconfigurável exerce um importante papel no desenvolvimento de sistemas de computação de alto desempenho, especialmente com o surgimento de dispositivos reconfiguráveis como as FPGAs (SKLIAROVA e FERRARI, 2003).

No caso das FPGAs, o desenvolvimento inclui três fases principais, necessárias para projetar e implementar um circuito: especificação, implementação e verificação. A seguir, será detalhada cada uma delas (SKLIAROVA e FERRARI, 2003).

- Especificação: A especificação do circuito a ser implementado em uma FPGA, pode ser feita com o auxilio da VHDL (*Very-High Speed Integrated Circuits Hardware Description Language*).
- Implementação: Caracterizada principalmente por se tratar do encaminhamento do circuito que interliga os vários componentes reconfiguráveis.
- Verificação: Tem como função principal a simulação, verificando a funcionalidade lógica do circuito. Após a configuração da FPGA, é possível verificar a implementação física do circuito.

1.1. OBJETIVO GERAL

O objetivo deste projeto é desenvolver e implementar um sistema de controle automático de temperatura baseado em FPGA.

1.2. OBJETIVOS ESPECÍFICOS

Os objetivos específicos desse projeto são:

- Efetuar o monitoramento e controle da temperatura tendo como base um sistema digital implementado em FPGAs conectado a dispositivos de sensoriamento e atuação;
- Dispor de botões para iniciar, terminar e configurar o sistema;
- Informar valor da temperatura atual do sistema em um *display* de sete segmentos;
- Operar tendo como base em um sistema de temperatura de referência no qual o erro seja tecnicamente aceitável para condições acadêmicas;
- Oferecer uma faixa de operação tecnicamente aceitável para condições acadêmicas e que possa ser evidenciado o controle efetivo da temperatura.

1.3. JUSTIFICATIVA

Percebendo-se a necessidade da otimização de processos ligados a controle de temperatura, juntamente com os conhecimentos a serem adquiridos em lógica reconfigurável, optou-se por desenvolver um sistema de controle de temperatura baseado em lógica reconfigurável FPGA.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. FPGA

FPGA é um dispositivo lógico programável de maior capacidade disponível no mercado hoje em dia. Suas funcionalidades são definidas exclusivamente pelos usuários, e o permite corrigir, incrementar ou mesmo apagar o circuito criado. Uma de suas maiores vantagens é a sua arquitetura flexível que serve para uma ampla gama de aplicações, possibilitando a implementação de sistemas completos em um único encapsulamento (PEDRONI, 2010).

Uma FPGA pode ser usada para implementar qualquer função lógica que um circuito integrado ASIC (*Application Specific Integrated Circuit*) poderia realizar. Entretanto esta mesma FPGA pode ser reprogramada para outra função. Esta flexibilidade de programação, associada a potentes ferramentas de desenvolvimento, possibilita ao usuário, o acesso a projetos de circuitos integrados complexos sem o elevado custo de engenharia, se comparado aos circuitos integrados, já que este, por sua vez é implementado em tempo de manufatura, e não pode mais ser modificado.

Nas FPGAs, o programa pode ser carregado a partir do *software ISE 12.4*, da *Xilinx*, instalado no computador, ou gravado na própria placa *BASYS2*, da qual disponibiliza uma memória *flash* ROM, em sua plataforma. As FPGAs são compostas por blocos de entrada e saída (I/O), blocos lógicos configuráveis (CLB) e chaves de interconexões (interruptores programáveis), conforme ilustra a Figura 1.

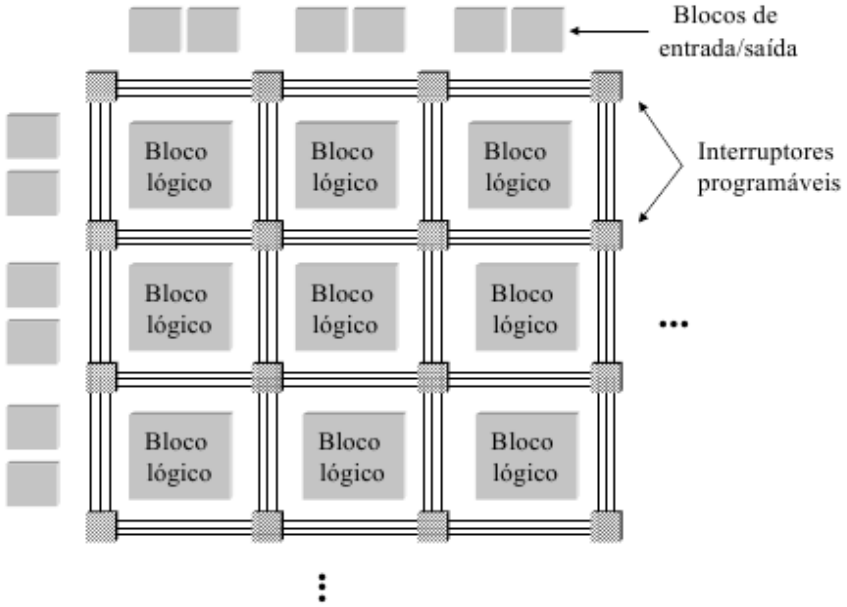

Figura 1: Arquitetura de um dispositivo FPGA
Fonte: (SKLIAROVA e FERRARI, 2003).

Os blocos lógicos configuráveis (CLB) contêm *flip-flops* e lógica combinacional que permitem a construção de funções lógicas. Os blocos de entrada e saída (I/O) são responsáveis pela interface de entrada e saída dos blocos lógicos. Os interruptores programáveis são responsáveis pela comunicação entre os blocos lógicos configuráveis e os blocos de entrada e saída. Quanto à programação das FPGAs, a mesma pode ser feita através de uma linguagem de descrição de hardware VHDL (SKLIAROVA e FERRARI, 2003).

2.2. VHDL

VHDL é uma linguagem de descrição de hardware, independente de tecnologia e de fabricante. O código descreve o comportamento, a partir do qual um circuito físico correspondente é inferido pelo compilador. Além da síntese, VHDL permite também a simulação de circuitos digitais (PEDRONI, 2010).

A VHDL surgiu através do governo dos Estados Unidos, pois foi a primeira instituição a reconhecer suas vantagens. Tornou-se claro que havia a necessidade de padronizar uma linguagem para descrever a estrutura e função dos circuitos integrados. Posteriormente, em 1987, o IEEE (*Institute of Electrical and Electronic Engineers*) padronizou a linguagem VHDL, tornando-a universal (ASHENDEN, 2008).

2.3. Componentes de um projeto VHDL

Os códigos VHDL são formados por três partes: Declarações de bibliotecas/pacotes, entidade e arquitetura. (PEDRONI, 2010).

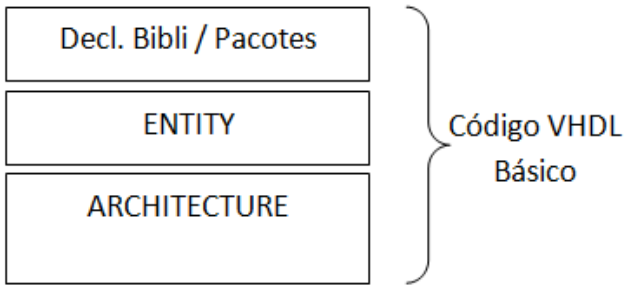

Figura 2: Estrutura de um código VHDL básico
Fonte: (PEDRONI, 2010).

2.4. PLDs

Dispositivos lógicos programáveis (PLDs) são dispositivos que possuem em seu interior diversas portas lógicas, flip-flops e registradores, que oferecem uma forma flexível de arranjo destes componentes em nível de hardware. Resumindo, PLD é um *chip* de uso geral cujo *hardware* pode ser configurado para atender as especificações particulares (PEDRONI, 2010).

3. ESTADO DA ARTE

Este capítulo tem por objetivo apresentar o confronto do trabalho desenvolvido com outros similares, visando desta forma, demonstrar a sua contribuição específica.

Em seu trabalho, SILVA JR (2011) aborda um estudo da VHDL, da tecnologia FPGA e de um tipo de filtro digital, IRR (*Infinite Impulse Response*) em um *Kit Spartan-3E*, da *Xilinx*, apresentando ao final uma documentação deste estudo, que pode ser utilizada como ferramenta didática para trabalhos com FPGA.

Em seu artigo, JODAS (2010) fala sobre os estudos de arquiteturas de computadores paralelos que utilizam dispositivos FPGA para obter um alto desempenho na execução de aplicações específicas.

Diante dos trabalhos encontrados, e de outros que aqui não foram citados, constatou-se a relevância de se desenvolver o trabalho proposto, entre outros motivos (conforme objetivos da introdução).

4. METODOLOGIA

Neste capítulo será descrito o sistema e as primeiras experiências que possibilitaram o domínio do conhecimento para suplementar o sistema proposto.

4.1. Primeiras Experiências

Para fins de testes, foi realizado um circuito utilizando *leds* nas saídas do conversor A/D como ilustra a Figura 3.

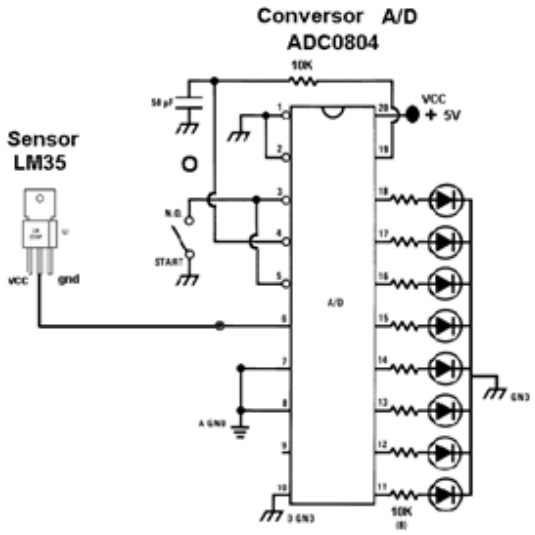

Figura 3: Circuito do conversor A/D com leds nas saídas.
Fonte: Autoria própria

O código abaixo, foi desenvolvido em VHDL onde o vetor “a” foi direcionado a 8 dos 16 pinos dos PMODs da placa BASYS2, o vetor “b” foi direcionado para 8 chaves SW, e as saídas “y” direcionadas pra 8 *leds* LD da mesma placa.

```
library ieee;
use ieee.std_logic_1164.all;

entity tcc4 is
    Port ( a,b : in  STD_LOGIC_vector (7 downto 0);

           y : out  STD_LOGIC_vector (7 downto 0));

end tcc4;

architecture Behavioral of tcc4 is

begin

 y(0) <= '1' when (a(0) <= b(0)) else
       '0';
 y(1) <= '1' when (a(1) <= b(1)) else
       '0';
 y(2) <= '1' when (a(2) <= b(2)) else
       '0';
 y(3) <= '1' when (a(3) <= b(3)) else
       '0';
 y(4) <= '1' when (a(4) <= b(4)) else
       '0';
 y(5) <= '1' when (a(5) <= b(5)) else
       '0';
 y(6) <= '1' when (a(6) <= b(6)) else
       '0';
 y(7) <= '1' when (a(7) <= b(7)) else
       '0';

end Behavioral;
```

O código tem a função de comparar as entradas do vetor “a” e, se caso for menor ou igual ao vetor “b”, a saída “y” recebe nível lógico alto, ou seja, o *led* é acionado, caso contrário, permanece desligado.

4.2. Descrição do Sistema

A Figura 4 demonstra através de um diagrama em blocos funcional o sistema como um todo.

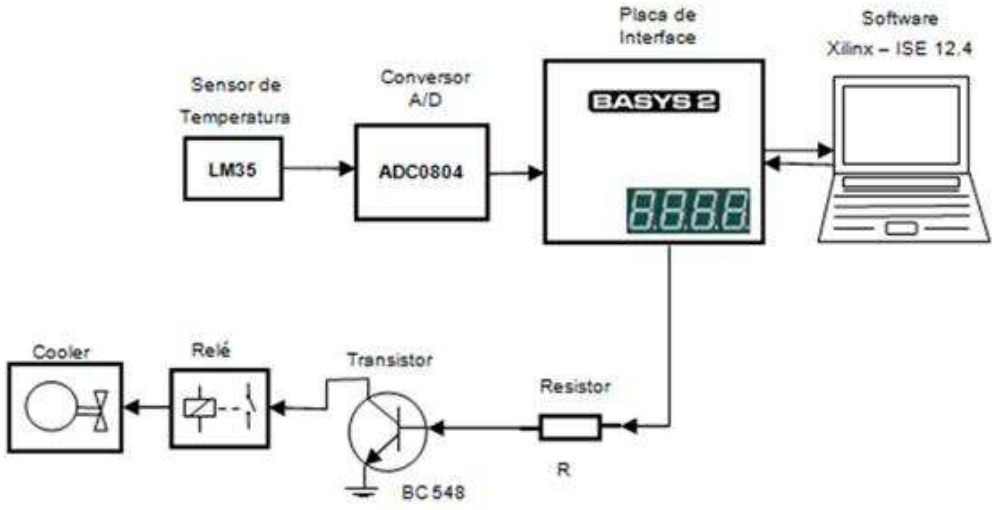

Figura 4: Circuito de funcionamento do sistema.
Fonte: Autoria própria

4.2.1. Circuito com Componentes Discretos

O ponto de entrada do sistema é a captação da temperatura por sensor, LM35, que converte temperatura em tensão. Sua resolução é de 10mV/ºC. A faixa na qual o sistema opera é de 0ºC a 150ºC, que corresponde de 0mV a 1500mV.

Recebe a tensão do sensor LM35 ou A/D, ADC 0804, de 8 *bits*, que converte a tensão recebida em *bits* na forma paralela a serem lidas pela placa BASYS2. A partir da faixa de valores lidas do sensor, o conversor A/D opera dentro da seguinte faixa: #00000000 (0mV) a #10010100 (1500mV).

Os dados na forma paralela do conversor A/D são convertidos da faixa TTL (0 a 5V) para LVTTL (0 a 3,3V) através de um *buffer*, 7407 para poder ser lido pelas portas paralelas (Conectores PMOD) de entrada da BASYS2, conforme Figura 5.

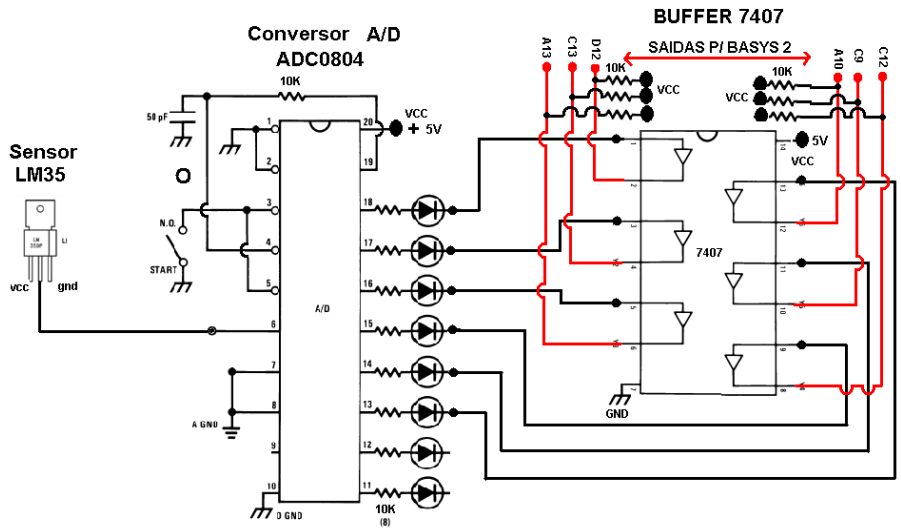

Figura 5: Circuito de funcionamento com aquisição do buffer.
Fonte: Autoria própria

O controle "*on/off*" é conhecido por apresentar problemas na região de transição do ponto de referência. Para resolver isto, foi implementado em hardware um circuito temporizador que provê uma histerese de 10 segundos, a Figura 6 ilustra este circuito.

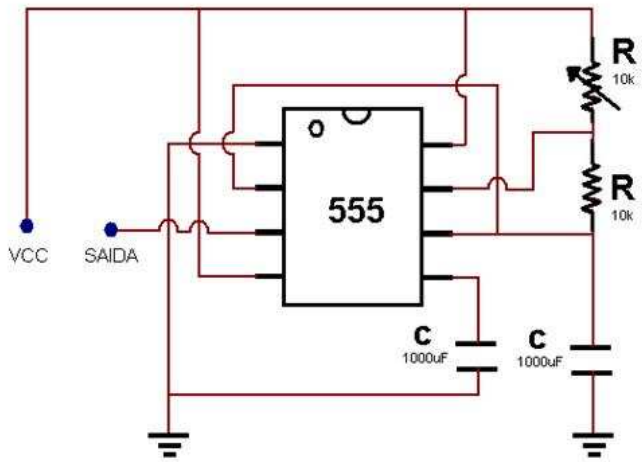

Figura 6: Circuito de funcionamento do temporizador.
Fonte: Autoria própria

Recebendo os dados provenientes do A/D através do *buffer*, o código VHDL sintetizado no dispositivo “XC35100E” da placa BASYS2 efetua o controle “*on/off*” da temperatura através da comparação da leitura com um valor pré-definido de temperatura e acionar (ou não) um circuito externo (também conectado através dos PMODs), baseado em um relé para ligar ou desligar uma ventoinha.

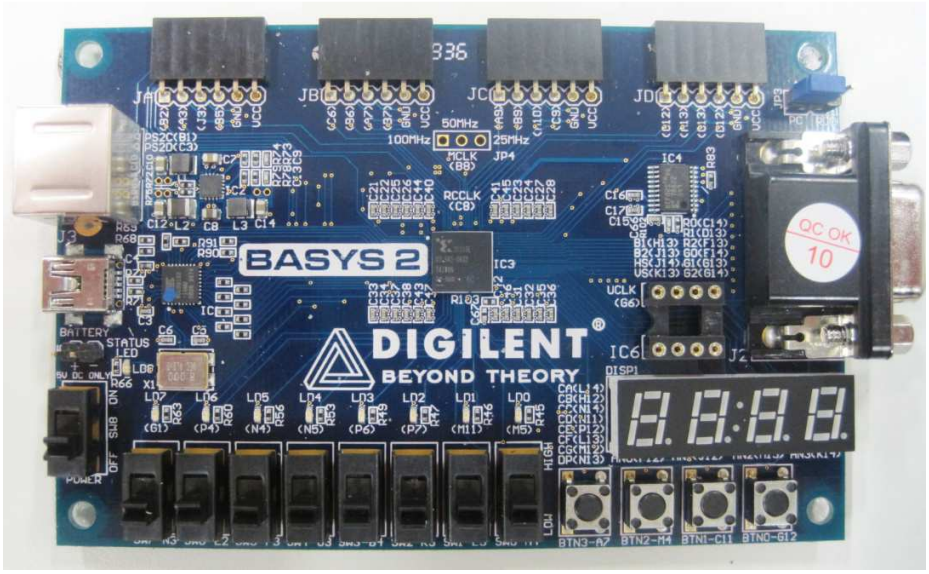

Figura 7: Placa BASYS2, da *Digilent*.
Fonte: Autoria própria.

4.2.2. Descrição do Subsistema da BASYS2

Para a implementação e simulação utilizaremos o software ISE 12.4-da *Xilinx*, onde contêm os seguintes códigos de configuração:

1) Código Principal;
2) Contador/C*lock* (u1);
3) Multiplexador (u2);
4) Decodificador (u3);
5) Conversor BCD (u4).

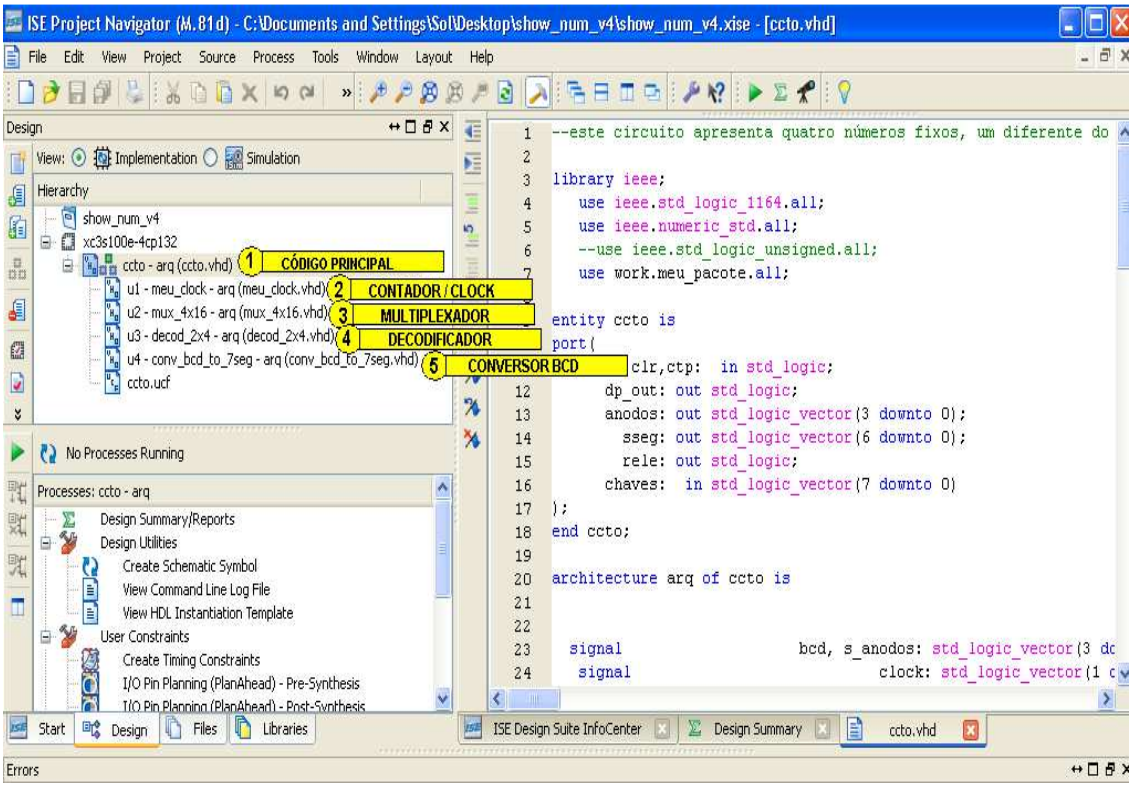

Figura 8: Software ISE 12.4, da *Xilinx*.
Fonte: Autoria própria.

O sistema implementado em FPGA está apresentado na Figura 9.

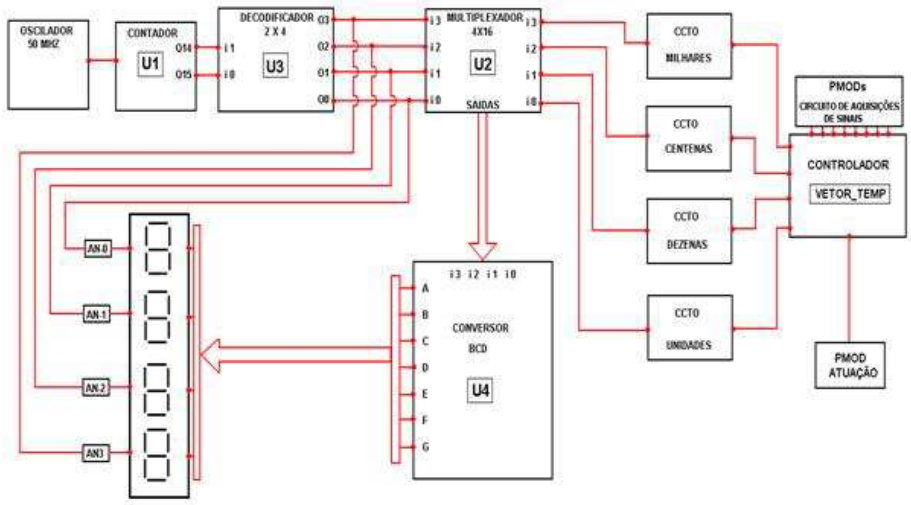

Figura 9: Diagrama de funcionamento do circuito.
Fonte: Autoria própria.

O código processo bloco U1 (contador), tem como função dividir as frequências de oscilações necessárias para que a persistência retiniana visualize que os 4 display estejam ligados juntos.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity meu_clock is
generic (n: integer := 26);
port(
   mclk, clr:  in std_logic;
   clock    : out std_logic_vector(1 downto 0)
);
end meu_clock;

architecture arq of meu_clock is
   signal count: std_logic_vector(n-1 downto 0);
   signal reset: std_logic;

begin
   process (mclk, count, reset, clr)
      begin

         reset <= '0';
         if((clr = '1') or (reset = '1')) then
            count <= (others => '0');
         elsif((mclk'event) and (mclk = '1')) then
            count <= std_logic_vector(unsigned(count) + 1);
         end if;
      clock(1) <= count(n-1);
      clock(0) <= count(n-2);
   end process;
end arq;
```

O vetor (mclk) foi direcionado a um oscilador de 50MHz contido na placa BASYS2. Através de um processo lógico de divisão de frequência, o código em questão realiza a operação divisória e fornece nos vetores (clock0, clock1) dois sinais de oscilações 763Hz e1.526Hz para o controle dos anodos dos *displays*.

O código processo bloco U2 (multiplexador), recebe as entradas milhares, centenas, dezenas e unidade e repassa para um conversor BCD.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mux_4x16 is
port(
   i3, i2, i1, i0: in std_logic_vector(3 downto 0);
   chave: in std_logic_vector(3 downto 0);
   saida: out std_logic_vector(3 downto 0));
end mux_4x16;

architecture arq of mux_4x16 is
begin
   with chave select
      saida <=     i3 when "0111",
                   i2 when "1011",
                   i1 when "1101",
                   i0 when "1110",
                "1111" when others;
end arq;
```

Este multiplexador obedece a seguinte tabela verdade:

Tabela 1: Tabela-verdade, ordem de processamento do multiplexador.

0111	MILHARES	I3
1011	CENTENAS	I2
1101	DEZENAS	I1
1110	UNIDADES	I0
1111	“1111”	

O código processo bloco U3 (decodificador), recebe os dois bits da entrada do clock (763 e 1.526Hz) e fornece quatro bits para a seleção dos anodos e do circuito do dígito a ser mostrado.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity decod_2x4 is
port(
   entrada: in std_logic_vector(1 downto 0);
   saida: out std_logic_vector(3 downto 0));
end decod_2x4;

architecture arq of decod_2x4 is
begin
   saida(3) <= not(entrada(1)) or not(entrada(0));
   saida(2) <= not(entrada(1)) or entrada(0);
   saida(1) <=     entrada(1)  or not(entrada(0));
   saida(0) <=     entrada(1)  or     entrada(0);
end arq;
```

O circuito decodificador obedece a seguinte tabela verdade:

Tabela 2: Tabela-verdade, ordem de processamento do decodificador.

i1	i0	O3	O2	O1	O0
0	0	1	1	1	0
0	0	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Este circuito funciona desta forma, pois os anodos são selecionados com zero.

O código processo bloco U4 (conversor BCD), recebe o digito binário em BCD selecionado pelo multiplexador e fornece os segmentos de “A” ao “G” para formarem o dígito no *display* de 7 segmentos.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity conv_bcd_to_7seg is
port(
   bcd: in std_logic_vector(3 downto 0);
   a_to_g: out std_logic_vector(6 downto 0));
end conv_bcd_to_7seg;

architecture arq of conv_bcd_to_7seg is
begin
   with bcd select         --abcdefg
         a_to_g(6 downto 0) <= "0000001" when "0000",
                               "1001111" when "0001",
                               "0010010" when "0010",
                               "0000110" when "0011",
                               "1001100" when "0100",
                               "0100100" when "0101",
                               "0100000" when "0110",
                               "0001111" when "0111",
                               "0000000" when "1000",
                               "0000100" when "1001",
                               "0111000" when others;
end arq;
```

O código VHDL principal é onde são declaradas as constantes, os tipos de dados e os subprogramas. Possibilitando o usuário em uma linha de código programar a lógica de acionamento dos atuadores, conforme a temperatura desejada.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.std_logic_unsigned.all;

entity ccto is
port(
   mclk, clr:  in std_logic;
      dp_out: out std_logic;
      anodos: out std_logic_vector(3 downto 0);
        sseg: out std_logic_vector(6 downto 0);
    chaves:  in std_logic_vector(7 downto 0)
);
end ccto;

architecture arq of ccto is
   signal                   bcd, s_anodos: std_logic_vector(3 downto 0);
   signal                           clock: std_logic_vector(1 downto 0);
   signal milhar, centena, dezena, unidade: std_logic_vector(3 downto 0);
   signal vetor_temp: std_logic_vector(11 downto 0);
```

```
component meu_clock is
generic (n: integer := 26);
port(
   mclk   ,       clr:  in std_logic;
   clock: out std_logic_vector(1 downto 0)
);
end component;

component decod_2x4 is
port(
   entrada: in std_logic_vector(1 downto 0);
   saida: out std_logic_vector(3 downto 0));
end component;

component conv_bcd_to_7seg is
port(
   bcd: in std_logic_vector(3 downto 0);
   a_to_g: out std_logic_vector(6 downto 0));
end component;


begin

u1: entity work.meu_clock
generic map (n => 15)
port map(mclk => mclk, clr => clr, clock => clock);

milhar <= "0000";

vetor_temp <= to_bcd(chaves(6) & chaves (5) & chaves(4) & chaves(3) &
                     chaves(2) & chaves (1) & chaves(0) &       '0');

centena <= vetor_temp(11 downto 08) when clr='0' else "0000";
dezena  <= vetor_temp(07 downto 04) when clr='0' else "0000";
unidade <= vetor_temp(03 downto 00) when clr='0' else "0000";



-- esta é a lógica pra se acionar o relé que vai ligar a ventoinha


  rele <= '1' when chaves > "00011011" else '0';


  u2: entity work.mux_4x16
  port map(i3=> milhar, i2=> centena, i1=> dezena,
           i0=> unidade, chave=> s_anodos, saida=> bcd);

  u3: entity work.decod_2x4
  port map(entrada=> clock, saida=> s_anodos);


  anodos <= s_anodos;


    dp_out <= '1';


        u4: entity work.conv_bcd_to_7seg
        port map(bcd => bcd, a_to_g => sseg);
   end arq;
```

4.2.3 Resolução de Trabalho do Sistema

De acordo com a tabela verdade, na Tabela 3, pode-se visualizar que cada seqüência binária corresponde a um valor de tensão e esta seqüência deve ser introduzida no código VHDL para programar os valores desejados de temperatura.

Tabela 3: Tabela-verdade, níveis de quantização relação relacionados com a temperatura.

D7	D6	D5	D4	D3	D2	D1	D0	Tensão (mV)	Temperatura (Cº)
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	19	1
0	0	0	0	0	0	1	0	39	3
0	0	0	0	0	0	1	1	58	5
0	0	0	0	0	1	0	0	78	7
0	0	0	0	0	1	0	1	98	9
0	0	0	0	0	1	1	0	117	11
0	0	0	0	0	1	1	1	136	13
0	0	0	0	1	0	0	0	156	15
0	0	0	0	1	0	0	1	175	17
0	0	0	0	1	0	1	0	195	19
..									
..									
..									
..									
..									
..									
1	1	1	1	1	1	0	0	4,941	494
1	1	1	1	1	1	0	1	4,96	496
1	1	1	1	1	1	1	0	4,98	498
1	1	1	1	1	1	1	1	4,999	499

5. RESULTADOS

Sabendo que a faixa de temperatura pode ser controlada pelo usuário, foram realizados alguns testes. Em um deles, o ponto de referência foi determinado em 52º C. O Gráfico 1 mostra o desempenho do controle. A Figura 10 mostra o sistema implementado em maquete onde possui lâmpadas incandescentes que em um período permanecem ligadas a fim de aquecer o ambiente até a temperatura desejada.

Figura 10: Sistema de aquecimento do ambiente.
Fonte: Autoria própria.

Ao chegar à temperatura configurada pelo usuário, entram em funcionamento os atuadores, que tem a função de resfriar o ambiente. A Figura 11 mostra este processo, onde é possível perceber a abertura das portas, o acionamento dos coolers, o desligamento das lâmpadas incandescentes e o acionamento dos LEDs.

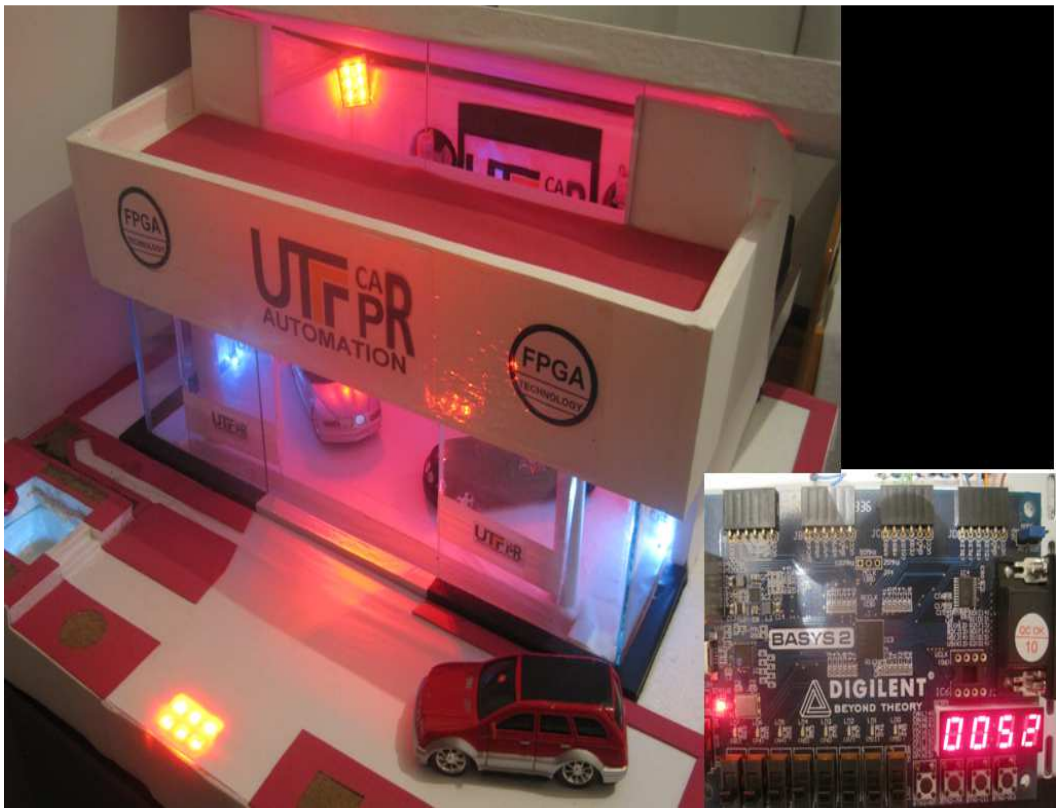

Figura 11: Sistema de resfriamento do ambiente.
Fonte: Autoria própria.

Observando o Gráfico 1 a seguir, é possível analisar a relação entre o tempo de atuação “t(s)” e a variação da temperatura “T(°C)”, do sistema de aquecimento e resfriamento.

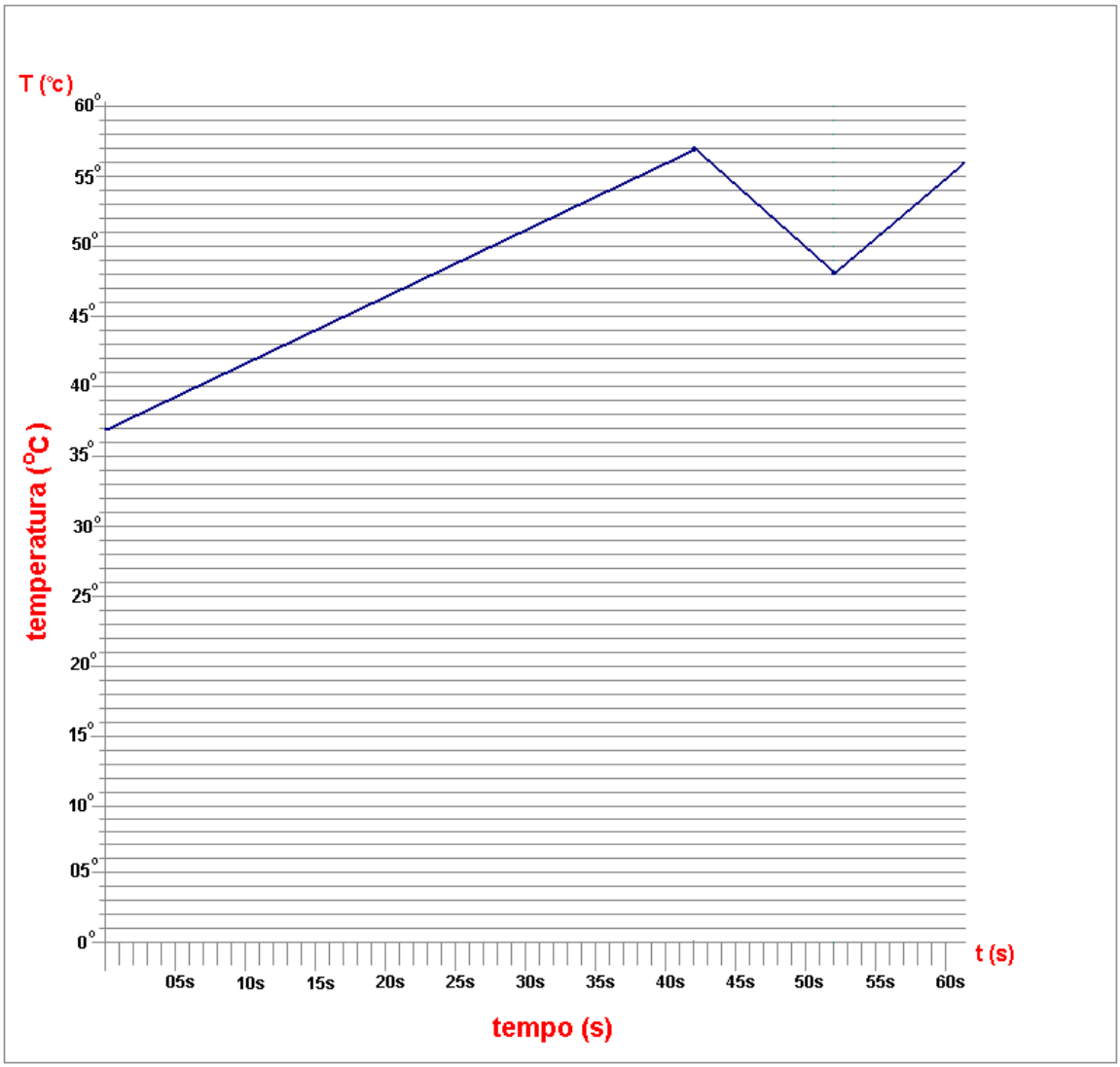

Gráfico 1: Atuação do sistema de aquecimento e resfriamento em 52°C.
Fonte: Autoria própria.

Para fins de testes, foram realizados controles para acionamento dos atuadores com 46ºC, já que o usuário tem a possibilidade de controlar os ajustes da temperatura desejada, conforme pode ser visualizado no Gráfico 2.

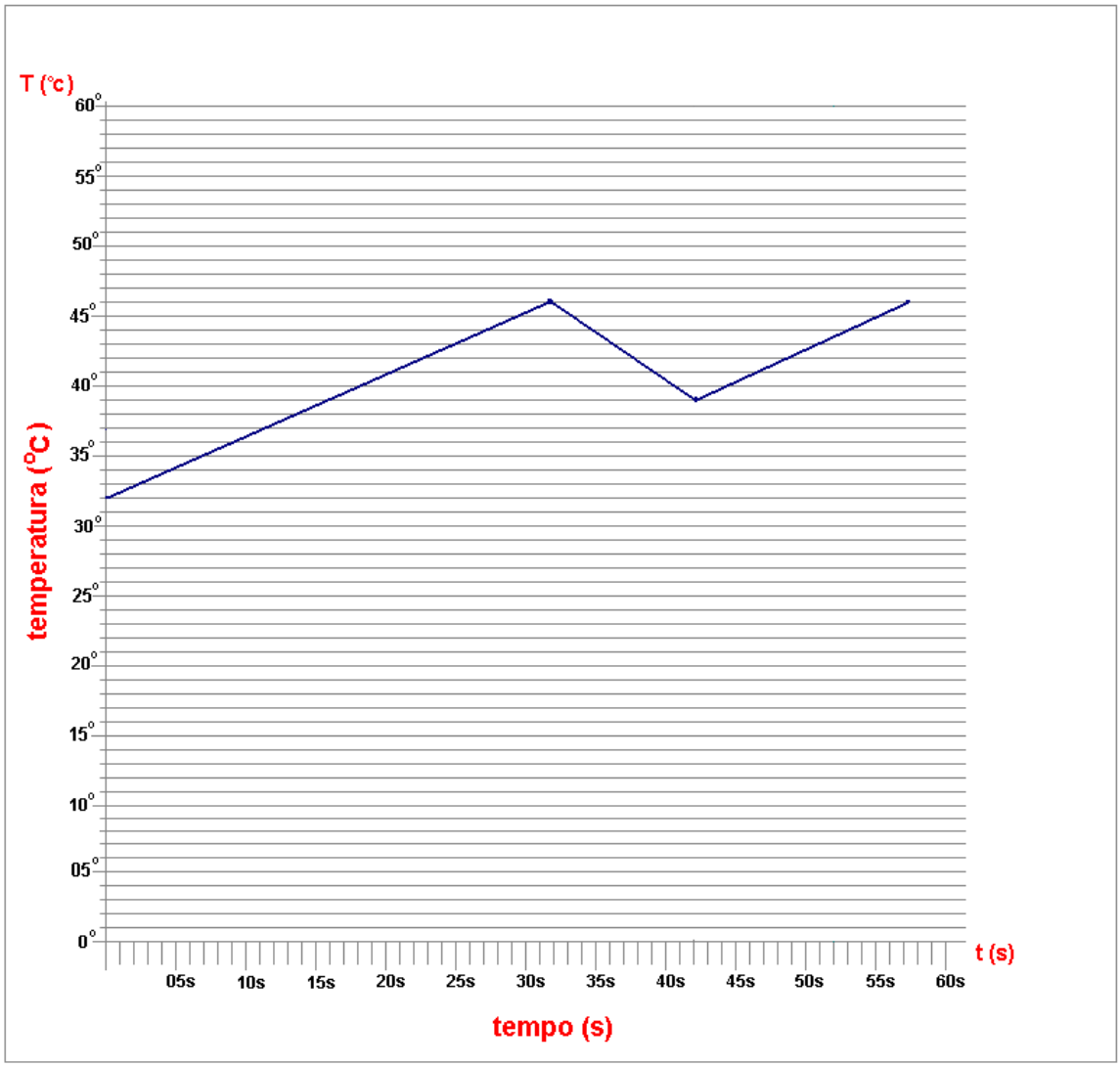

Gráfico 2: Atuação do sistema de aquecimento e resfriamento em 46^{0}C.
Fonte: Autoria própria.

A Figura 12 a seguir mostra a imagem do circuito de aquisição de sinais e a parte de atuação do sistema. Na sequencia, Figura 13, o circuito do temporizador.

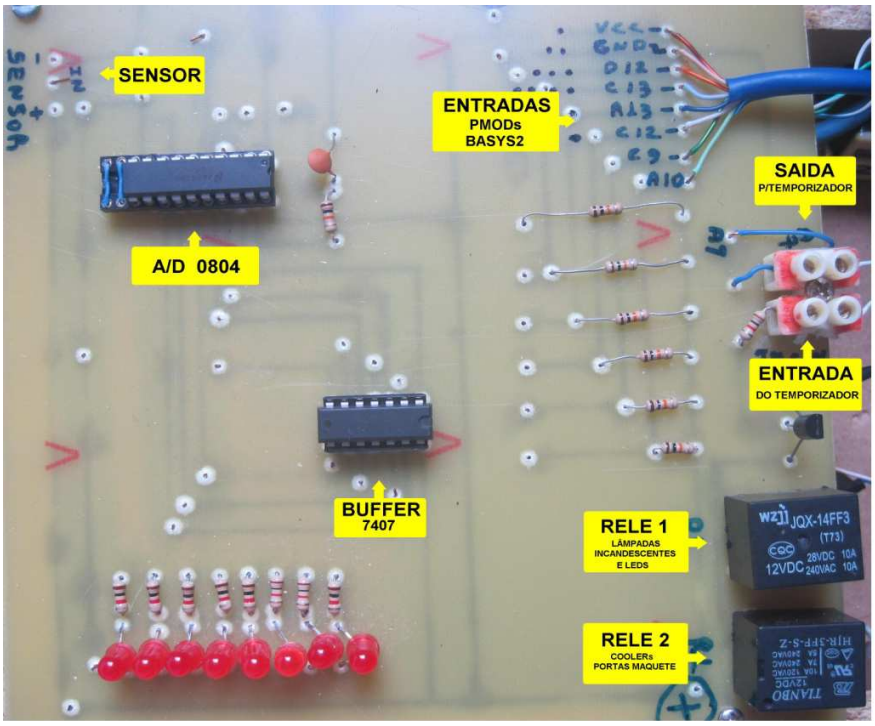

Figura 12: Circuito de aquisição de sinais.
Fonte: Autoria própria.

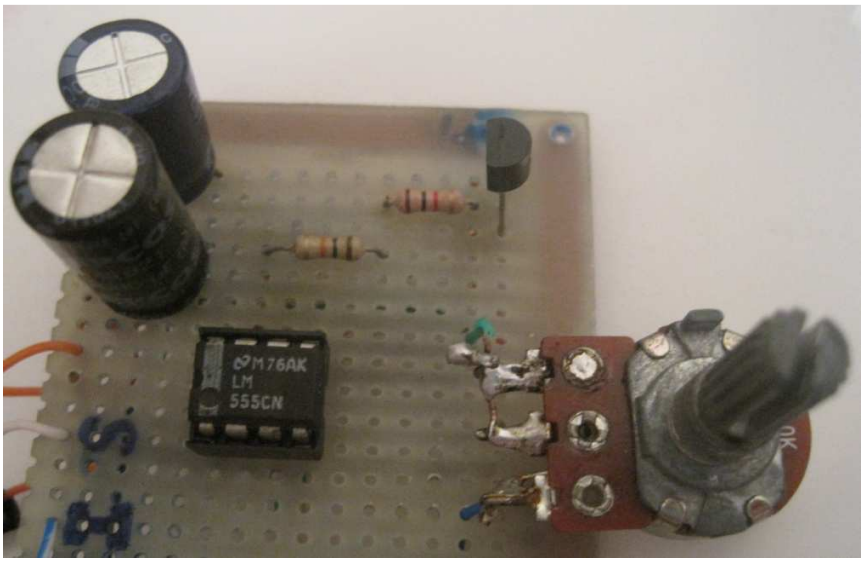

Figura 13: Circuito temporizador.
Fonte: Autoria própria.

6. CONCLUSÕES

Para que a conclusão desse projeto fosse possível, foi preciso obedecer algumas etapas, dentre elas o estudo sobre FPGA e linguagem VHDL, do qual exigiu dos acadêmicos a participação em aulas presencias nas matérias de “Circuitos Digitais” e “Lógica Reconfigurável” dos cursos de Engenharia Elétrica e Engenharia da Computação, conciliando neste mesmo período de tempo o desenvolvimento da parte prática do trabalho. Foram realizados pequenos testes de aquisição de sinais, tendo como princípio a leitura do sensor e suas atenuações. Na sequência a transformação do sinal analógico em digital para que fosse possível a comunicação com a placa *BASYS2*. Com tais conhecimentos iniciou-se a elaboração de circuitos digitais a fim de aperfeiçoá-los até atingir o objetivo final.

Precisou-se de um estudo aprofundado da tecnologia de lógica reconfigurável, onde se tornou uma das maiores dificuldades encontradas durante o projeto, obrigando os alunos a buscar informações sobre um assunto, até o momento, desconhecido. Outro desafio foi relembrar os conceitos fundamentais de circuitos digitais, eletrônica digital, acionamentos elétricos, aquisição de sinais dentre outros assuntos aprendido no curso de Tecnologia de Automação de Processos Industriais.

Os principais erros cometidos foram:

a) Não utilizar amplificador operacional entre o sensor LM35 e o conversor A/D 0804. Isto fez com que não fosse possível a total exploração da faixa de 0 a 5V permitida pela entrada do A/D;
b) Não efetuar uma gestão eficaz ao tempo disponível para desenvolver o projeto.

Entende-se que o objetivo em desenvolver e implementar um sistema de controle automático de temperatura baseado em FPGAs, foi atingido, possibilitando a expansão de projetos futuros sobre a mesma arquitetura do sistema em questão, dentre eles :

1) Oferecer flexibilidade de ponto de referência de controle de temperatura através das interfaces da placa BASYS2 (botões BTNs e *display*), não necessitando desta forma novas sínteses de código;
2) Oferecer um controle de histerese via código VHDL;
3) Substituir o controle on/off por PID*(Proportional Integral Derivative).*

REFERÊNCIAS

ADEPT. **Application User's Manual**. Adept, 2008.

ASHENDEN, Peter J. **The Designer´s Guide to VHDL**. 3 ed.Boston: Morgan Kaufmann, 2008.

CHU, Pong P. **FPGA Prototyping by VHDL Examples**. New Jersey: Wiley,2008.

DIGILENT. **Digilent Basys2 Reference Manual**. Digilent, 2010.

JODAS, Danilo S. **Utilização de FPGA em Aplicações de Alto Desempenho**. São José do Rio Preto: Instituto de Biociências, Letras e Exatas, 2010.

MEIRA, Fernando. 2008. **Automagate**. Disponível em:

http://automagate.com.br/?p=12. Acesso em: 19 out. 2012.

PEDRONI, **Volnei A. Eletrônica Digital Moderna e VHDL**. Brasil: Campus, 2010.

SILVA JR, Adirson M. **Estudo e Documentação da Tecnologia HDL e FPGA com Desenvolvimento de Filtros Digitais**. Pato Branco: Universidade Tecnológica Federal do Paraná, 2011.

Skliarova, Iouliia. Ferrari, Antonio B. **Introdução à Computação Reconfigurável**, 2003.

XILINX. **ISE Project Navigator**, version 12.4. Xilinx, 2010.