Daniel Henrique Camargo de Souza

Projeto Integrador
Sistema automatizado de classificação por altura e cor

Florianópolis, SC - Brasil

27 de Julho de 2016

Daniel Henrique Camargo de Souza

Projeto Integrador
Sistema automatizado de classificação por altura e cor

Relatório final do projeto integrador 2016 do curso de gradução em Engenharia Eletrônica - IFSC, que consiste na implementação de um sistema automatizado de classificação por altura e cor desenvolvido em dispositivo lógico programável (PLD) em liguemagem de descrição de hardware (VHDL)

Instituto Federal de Santa Catarina - IFSC

Departamento de Engenharia Eletrônica

Florianópolis, SC - Brasil

27 de Julho de 2016

Resumo

O propósito deste trabalho consistiu na implementação de um sistema automatizado de classificação de objetos por altura e cor, desenvolvido prioritariamente em VHDL, com um sistema embarcado de condicionamento e conversão analogica/digital dos sinais externos.

Palavras-chaves: VHDL, FPGA, ALTERA, HCSR04, DE2-115.

Lista de ilustrações

Lista de abreviaturas e siglas

PLD	*Programmable Logic Device* - Dispositivio Lógico Programável
FPGA	*Field programmable gate array* - Arranjo de Portas Programável em Campo
VHDL	*VHSIC Hardware Description Languaget* - Linguagem de descrição de hardware VHSIC *Very High Speed Integrated Circuits*
ALTERA	Altera Corporation
Quartus	Suide de desenvolvimento para PLD's e VSIC's
HMI	*Human Machine Interface* - Interface Homen-Maquina
RTL	*Register Transfer Level* - Nível de Registrador de Transferência
FSM	*Finite State Machine* - Maquina de Estado
LDR	*Light Dependent Resistor* - Resistor dependente da Luz
IFSC	Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina
ABNT	Associação Brasileira de Normas Técnicas
abnTex	Normas para LaTeX

Sumário

1 Introdução

VHDL é uma linguagem de descrição de hardware, e tem por finalidade descrever o comportamento e a topologia de circuitos digitais. Segundo (PEDRONI, 2004) as dois principais aplicações imediatas de VHDL estão no campo dos dispositivos lógicos programáveis (PLD) incluindo CPLDs - *Complex Programmable Logic devices* e FPGAs *progammable gate array* e no campo dos ASICs (circuitos integrados de aplicação específica)

O presente trabalho teve como objetivo o desenvolvimento de um sistema automatizado de classificação de objetos por altura e cor, e foi totalmente desenvolvido utilizando-se a linguagem de descrição de hardware - VHDL.

A utilização de VHDL, deixa implicita que será necessário o uso de um PLD. Neste projeto, utilizou-se o Kit de desenvolvimento DE2-115 da terasic (TERASIC, 2014), que possui embarcado o PLD Cyclone IV E: EP4CE115F29C7 da ALTERA - um FPGA, será neste dispositivo que serão gravados todos os códigos compilados para o processamento do sistema automatizado.

O sistema como um todo possui basicamente dois grandes blocos, sendo: o bloco de processamente e interface humana, desenvolvido inteiramente no kit DE2-115, e o bloco de condicionamento de sinal e interface eletrônica, que consiste na integração dos sensores de ultrasom e cor (altura e cor) com uma interface de tratamento dos sinais que irão para o processamento no PLD (kit).

No Capitulo 2 será descrito os objetivos e caracteristicas do projeto, bem como uma breve revisão dos conceitos envolvidos para liguagem VHDL. No capitulo 3 será explanado toda a metodologia e procedimento experimental adotados, mostrando como se deu o desenvolvimento de cada parta deste projeto. No capitulo será discorrido sobre os resultados obtido, quais problemas ocorreram e como se comportou o sistema. Por fim, no capitulo 5 é transcorrido as conclusões e considerações finais.

2 O Projeto

O Sistema consiste em interpretar os sinais que se originam nos sensores, processa-los e a partir das escolhas do usuário - através de uma interface HMI (Humam Machine Interface) - classificar o objeto que esta em avaliação. Para isso foi utilizado os seguintes componentes:

- DE2-115 - Kit de desevolvimento contento embarcado o PDL: Cyclone IV E: EP4CE115F29C7
- HCSR04 - Sensor ultrassônico que será utilizado para medir altura
- Sensor de cor - utilizando-se de LED's e um LDR
- Motor de passo para a classificação final

2.1 Características exigidas

Como requisito exigido o sistema deveria atender as seguintes características:

- Através da interface com o LCD (embarcado no kit DE2-115), o operador seleciona o critério de classificação desejado para as peças (altura ou cor).
- Quando a peça é colocada em uma determinada posição, a medição selecionada deve ser realizada e o resultado deve ser apresentado no LCD.
- Na sequência, o motor de passo deve ser acionado para movimentar a peça para a direita ou esquerda, conforme classificação
- O sensor de cor deve ser capaz de identificar peças verdes e azuis.
- A medida de altura deve ser expressa em centimetros.
- Cada sensor deve ser montado em uma placa de circuito impresso, a qual deve possuir conectores apropriados para ligação com os pinos de entrada/saída do kit DE2-115
- Toda a parte de controle e leitura dos sensores deve ser implementada no FPGA utilizando a linguagem de descrição de hardware VHDL.

2.2 A linguagem VHDL

A linguagem VHDL foi originalmente desenvolvida sob o comando do Departamento de Defesa dos Estados Unidos (DARPA), em meados da década de 1980, para documentar o comportamento de ASICs que compunham os equipamentos vendidos às Forças Armadas americanas. Isto quer dizer que a linguagem VHDL foi desenvolvida para substituir os complexos manuais que descreviam o funcionamento dos ASICs. Até aquele momento, a única metodologia largamente utilizada no projeto de circuitos era a criação através de diagramas esquemáticos. O problema com esta metodologia é o fato de que desenho tem menor portabilidade, são mais complexos para compreensão e são extremamente dependentes da ferramenta utilizada para produzi-los.

Uma vez que o projeto VHSIC era de alta prioridade militar e havia dezenas de fornecedores envolvidos, o DoD estava preocupado principalmente com as questões de portabilidade, documentação e compreensibilidade dos projetos. Cada um destes fornecedores atuava desenvolvendo partes dos projetos ou mesmo fornecendo componentes que viriam a se encaixar em outros sistemas maiores. Desta forma o DoD optou por buscar desenvolver uma linguagem que servisse como base para troca de informações sobre estes componentes e projetos. Uma linguagem que, independente do formato original do circuito, pudesse servir como uma descrição e documentação eficientes do circuito, possibilitando os mais diferentes fornecedores e participantes a entender o funcionamento das outras partes, padronizando a comunicação.

Segundo Pedroni (2004):

> A motivação fundamental para usar VHDL é que esta linguagem fornece um padrão para dispositivos digitais, sendo portanto, portátil e reutilizável. Uma vez que o código VHDL foi escrito, ele pode ser usado tanto para implementar o circuito de um dispositivo lógico programável (Da Altera, Xilinx, Atmel, etc.) ou podem ser submetido diretamente para a fabricação de um chip (ASIC).

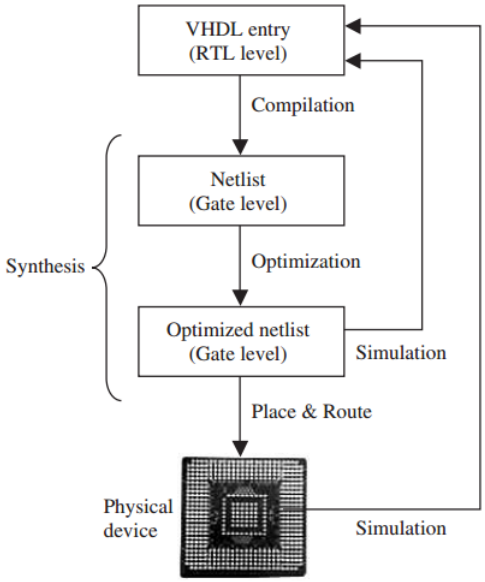
Pedroni (2004) ainda ressalta que, contrariamente aos programas de computador regulares que são seqüenciais, as declarações em VHDL são inerentemente simultâneas (paralelo). Por esta razão, VHDL é normalmente referido como um código em vez de um programa. Em VHDL, somente as declarações colocadas dentro de um processo, função ou procedimento são executadas sequencialmente.

As etapas seguidas durante um projeto em VHDL podem ser resumidas pela figura 1, que são basicamente: Compilação, Otimização e Gravação no PLD

Compilação é a conversão da linguagem de alto nível VHDL, que descreve o circuito no nível de registro de transferência (RTL), em um netlist no nível da porta. O segundo passo a otimização, é realizada no netlist em nível RTL. Nesta fase, o desenho pode

ser simulado. Finalmente, um software (montador) irá gerar o layout físico de um chip PLD/FPGA ou irá gerar as máscaras para um ASIC.

Figura 1 – Sintese de VHDL

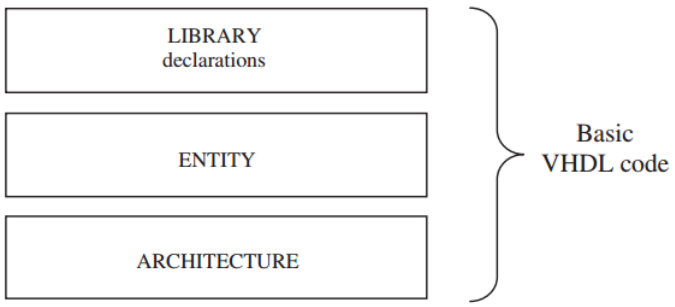


Fonte: (PEDRONI, 2004)

2.2.1 Estrutura de um Código VHDL

O código em VHDL é composto fundamentalmente por três seções: Bibliotecas (*LIBRARY*), Entidade(*ENTITY*), Arquitetura(*ARCHITECTURE*), conforme mostra a figura 2.

Figura 2 – Seções fundamentais do VHDL



Fonte: (PEDRONI, 2004)

As *LIBRARY* declarações de bibliotecas contém uma lista de todas as bibliotecas que podem ser usadas durante o projeto. exemplo: IEEE, std, WORK, etc. A *ENTITY* ou ENTIDADE: Especifica os pinos de I / O do circuito. e a *ARCHITECTURE* ou

ARQUITETURA: Contém o código VHDL que descreve como o circuito deve se comportar (função).

Abaixo subscreve-se o Código do arquivo *main.vhd* do presente projeto, para identificar as seções descritas anteriormente.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY main IS
  GENERIC (clk_divider: INTEGER := 50000000);
  PORT  (CLOCK_50: IN BIT;
    SW: IN STD_LOGIC_VECTOR (17 downto 0); --
    KEY: IN STD_LOGIC_VECTOR (3 downto 0);
    AD_COLOR : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    R_OUT,G_OUT,B_OUT: OUT STD_LOGIC;
    GPIO_ECHO : IN STD_LOGIC;
    GPIO_TRIG : BUFFER STD_LOGIC;
    LCD_RS, LCD_RW, LCD_BLON: OUT BIT;
    LCD_EN: BUFFER BIT;
    LCD_DATA: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    STEPPER: OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
    STEIRA: OUT STD_LOGIC;
    LEDG : OUT STD_LOGIC_VECTOR (8 DOWNTO 0);    -- LEDS para DEBUG
    LEDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0) );  -- LEDS para DEBUG
END main;

ARCHITECTURE behavior OF main IS
SIGNAL rst_system : STD_LOGIC := '1';
SIGNAL echo_in, trig_out : STD_LOGIC;
SIGNAL bcd : STD_LOGIC_VECTOR (9 DOWNTO 0);
SIGNAL en_color, ok_color, en_stepper: STD_LOGIC;
SIGNAL rgb_result: STD_LOGIC_VECTOR (2 DOWNTO 0);
SIGNAL comparador, sel_submenu1, sel_submenu2, sel_submenu31, sel_submenu32,
    presence : STD_LOGIC;

BEGIN
LEDR(17 DOWNTO 10) <= AD_COLOR;

echo_in <= GPIO_ECHO;

  PART_BRAIN: WORK.brain_system
    PORT MAP(CLOCK_50, ok_color, presence ,SW(0), rst_system, en_color,
    en_stepper, STEIRA);
    -- CLOCK, OK_COLOR, PRESENCE, LIGA, RST_SYSTEM, EN_COLOR, EN_STEPPER,
    STEIRA

  PART_HCSR04: WORK.hcsr04
```

```
    GENERIC MAP(8)
41  PORT MAP(CLOCK_50, rst_system, echo_in, trig_out, bcd, sel_submenu31,
    presence);
    -- CLOCK_IN, RST, ECHO, TRIGGER, BCD, SEL_31, PRESENCE
43
  PART_LCD: WORK.lcd_v2
45  GENERIC MAP (50000000)
    PORT MAP(CLOCK_50,KEY,bcd,rgb_result,LCD_RS,LCD_RW,LCD_BLON,LCD_EN,
    LCD_DATA,comparador,sel_submenu1,sel_submenu2,sel_submenu31,
    sel_submenu32,LEDG);
47  -- CLOCK_50, KEY, ALTURA, LCD_RS, LCD_RW, LCD_BLON, LCD_EN, LCD_DATA,
    LEDG

49  PART_COLOR: WORK.color
    GENERIC MAP (8)
51  PORT MAP (CLOCK_50, rst_system, en_color, AD_COLOR, ok_color, R_OUT,
    G_OUT, B_OUT, rgb_result);
    -- CLOCK_IN, RST, ENABLE, AD, OK_COLOR, R_OUT, G_OUT, B_OUT, IS_RGB
53
  PART_STEPPER: WORK.stepper
55  PORT MAP(CLOCK_50,en_stepper,rgb_result,comparador,sel_submenu1,
    sel_submenu2,sel_submenu32,STEPPER);

57 GPIO_TRIG <= trig_out;
  END;
```

3 Métodos e Procedimentos

A Metodologia empregada no desenvolvimento deste projeto consistiu em primeiramente dedicar os esforços no código VHDL que contém o bloco de interface com o usuário, ou seja, captura das seleções e visualização na tela LCD dos resultados. Por fim partiu-se para o sistema embarcado de condicionamento e conversão analógico/digital dos sinais de sensores.

Para sistematizar e prevenir erros, todo desenvolvimento foi subdividido em 6 etapas: Implementação do código de interface com o usuario, utilizando o kit DE2-115; Implementação do código para captura da altaura usando o sensor ultrassônico HCSR04; Implementação do Código e circuito para captura da cor; Implementação do código para acionamento do motor de passo (classificação); Implementação do sistema embarcado de interfaceamento eletrônico, para tratamento dos sinais e conversão analogico-digital; e por fim o Código de controle e processamento do sistema.

3.1 O Kit de desenvolvimento DE2-115

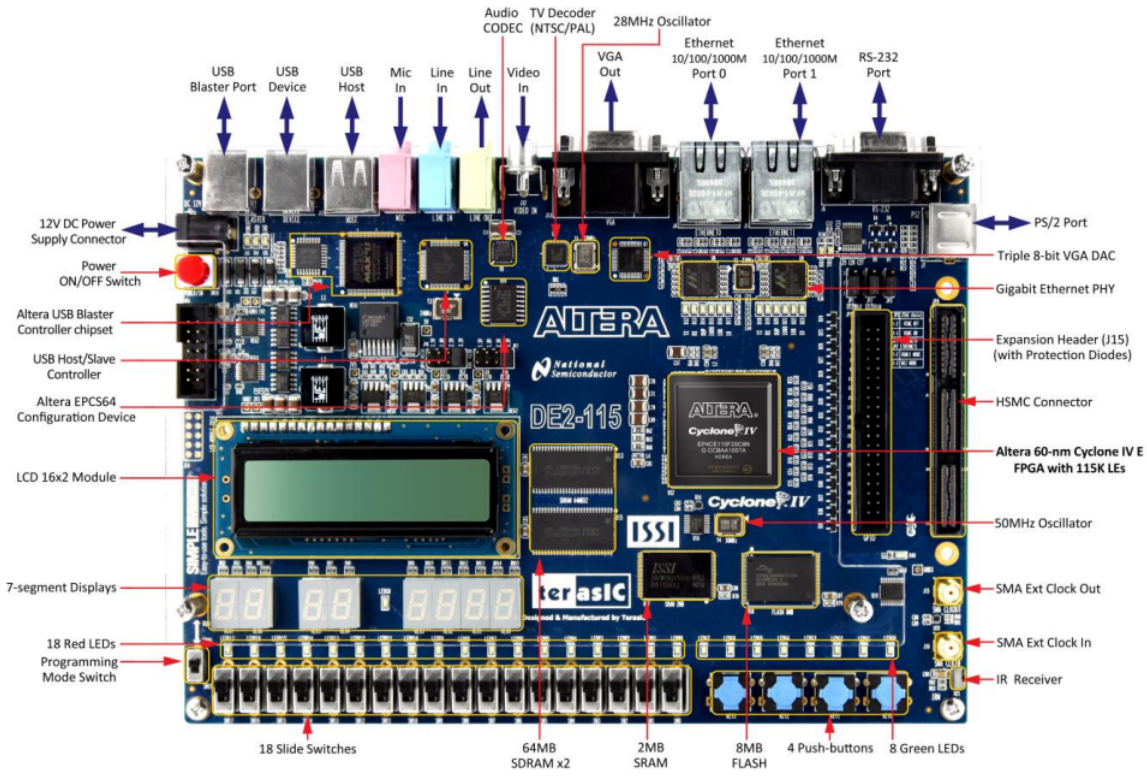
Para facilitar a implementação deste projeto, foi utilizado o kit de desenvolvimento DE2-115 fabricado pela Terasic (TERASIC, 2014), este kit é uma ferramente de desenvolvimento em FPGA poderosíssima e contém características como:

- PLD Altera Cyclone IV 4CE115 FPGA
- 4 Push-bottons
- 18 chaves deslizantes
- 18 leds vermelhos
- 9 leds verdes
- Clock interno de 50MHz
- Modulo interno de LCD 16x2

A figura 3 apresenta uma imagem do kit DE2-115

A partir da referência terasic (2014) pode-se conhecer o funcionamento do kit e assim segui-se com o desenvolvimento iniciando-se pelo acionamento do modulo de LCD

Figura 3 – Kit DE2-115



Fonte: (TERASIC, 2014)

3.1.1 Utilizando o LCD

O Modulo de LCD 16x2 embarcado ao kit D2-115, será utilizando como fonte principal de informação ao usuário, mostrará as opções para escolha (tipo de classificação) e os valores medidos.

Para acionar o modulo LCD pelo PLD embarcado no kit é preciso informar ao código quais pinos devem ser acionado, para isso é preciso conhecer os nomes (tags) dos pinos do LCD reconhecidos pelo Kit, a figura 4 mostra como estão rotulados os pinos do LCD.

Antes de escrever qualquer caracter no LCD é preciso inicia-lo, para isso existe um procedimento bem especifico que é mostrado na figura 5.

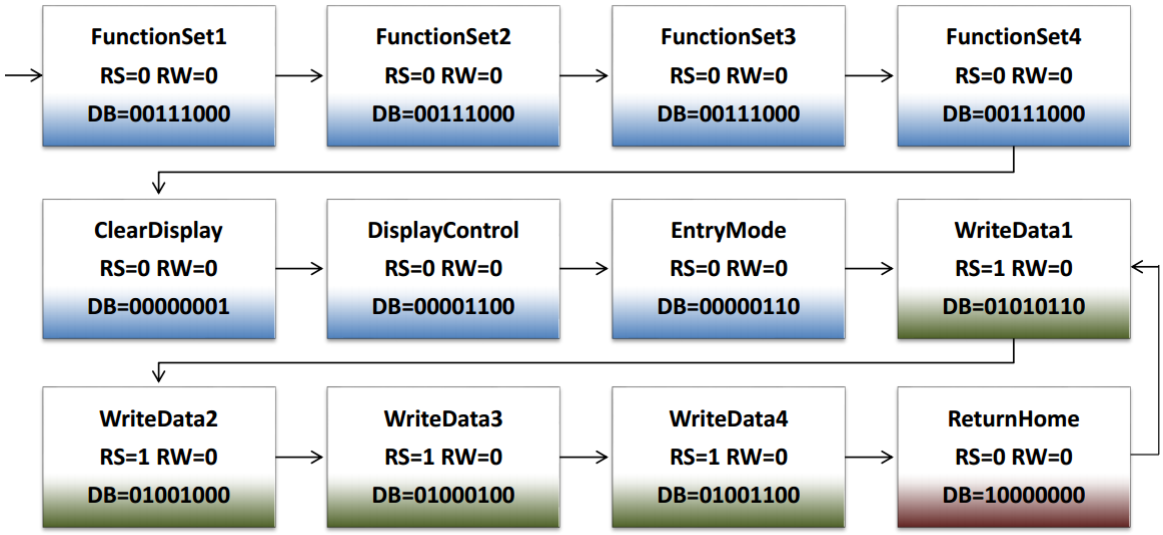
A seguir é apresentado o trecho do código em VHDL para inicialização do LCD usado no arquivo *menu1.vhd* deste projeto.

```
PROCESS (pr_state)
  BEGIN
    CASE pr_state IS
      WHEN FunctionSet1 =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00111000";
```

Figura 4 – Pinos do LCD no kit DE2-115

Fonte: (TERASIC, 2014)

Figura 5 – Inicialização do LCD



Fonte: Slides de Aula

```
        nx_state <= FunctionSet2;
    WHEN FunctionSet2 =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00111000";
        nx_state <= FunctionSet3;
    WHEN FunctionSet3 =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00111000";
```

```
        nx_state <= FunctionSet4;
    WHEN FunctionSet4 =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00111000";
        nx_state <= ClearDisplay;
    WHEN ClearDisplay =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00000001";
        nx_state <= DisplayControl;
    WHEN DisplayControl =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00001100";
        nx_state <= EntryMode;
    WHEN EntryMode =>
        LCD_RS<='0'; LCD_RW<='0';
        LCD_DATA <= "00000110";
        nx_state <= state_01;
    WHEN state_01 =>
        LCD_RS<='1'; LCD_RW<='0';
        LCD_DATA <= "01000011"; --'C'
        nx_state <= state_02;
    WHEN state_02 =>
        LCD_RS<='1'; LCD_RW<='0';
        LCD_DATA <= "01101100"; --'l'
        nx_state <= state_03;
```

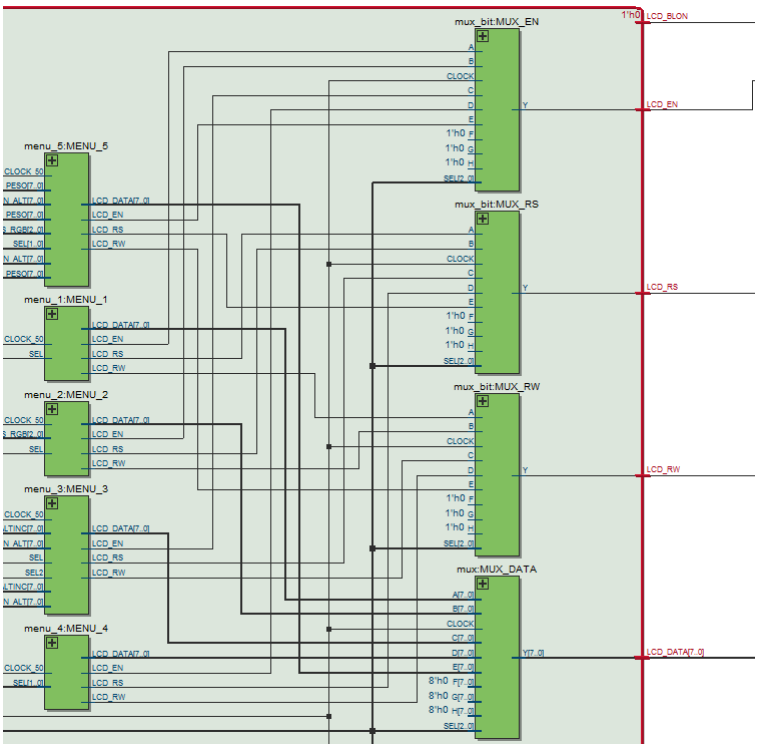
Afim de sistematizar o projeto, foi implementado um arquivo *.vhd* para cada menu do sistema, tendo ao todo 5 menus (Classficação, Menu Altura, Menu Cor, Menu Peso, e Geral) foi desenvolvido 5 arquivos, sendo que o método para exibição do menu no LCD foi através de um bloco de multiplexadores para cada pino do modulo de LCD ($LCD_{RW,}LCD_{RS}, LCD_{EN}, LCD_{DATA}$), sendo que na entrada de cada multiplexador deste é inserido o pino correspondente dos menus. Na figura 6 é possível visualizar em nível RTL a ligação entre os Menus (Códigos em VHDL) e os mulplexadores para cada pino de saída ao LCD.

Dessa forma de acordo com a seleção (que será feita pelos *Push-Bottons*) os mulplexadores trocam qual menu irá aparecer na modulo de LCD.

3.1.2 Utilizando os Push-Bottons

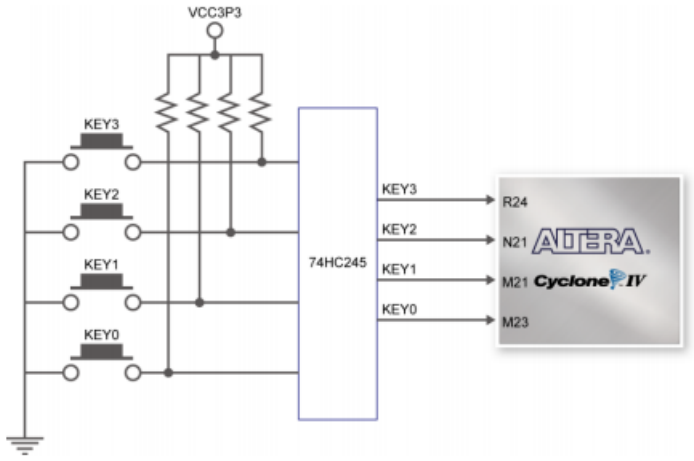
Todas as seleções do sistema serão feitas pelos push-bottons embarcados no kit DE2-115; sendo que da mesma forma que foi para o modulo de LCD é preciso identificar junto ao kit os nomes desses pinos. Na figura 7 é possível visualizar os nomes dos pinos junto ao kit.

Figura 6 – RTL dos menus do LCD



Fonte: Produzido pelo autor

Figura 7 – Pinos dos Push-Bottons no kit DE2-115



Fonte: (TERASIC, 2014)

Todo push-botton tem problema de trepidação de sinal, ou seja, ao pressionar e/ou liberar o push-botton este acaba gerando ruído e trepidações no sinal, o que pode gerar pulsos a mais no sistema e interferir de maneira negativa no processamento da informação. Afim de corrigir este problema foi aplicado aos push-bottons um tratamento de *Debouncing* via codigo. Abaixo subscreveu-se um techo do código de deboucing.

```
1 BEGIN
```

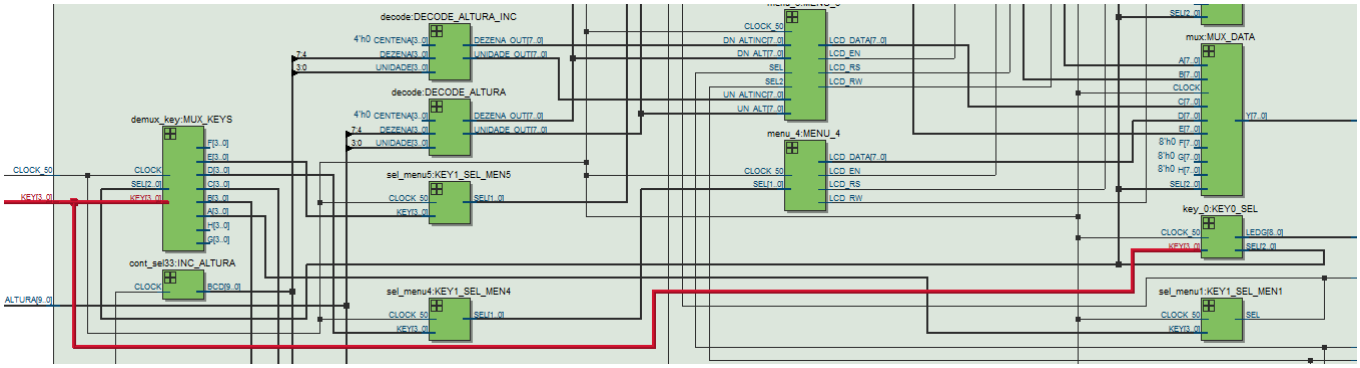
```
buttonAux <= KEY(1) WHEN defaultState = '0' ELSE (NOT KEY(1));
PROCESS (CLOCK_50)
  VARIABLE counter : INTEGER RANGE 0 TO CONT_MAX := 0;
  BEGIN
    IF (CLOCK_50'EVENT AND CLOCK_50='1') THEN
      IF (buttonPressed = '0' AND buttonAux = '1') THEN
        buttonPressed <= '1';
        counter := 0;
      END IF;

      IF (buttonPressed = '1' AND buttonAux = '1') THEN
        counter := 0;
      END IF;

      IF (buttonPressed = '1' AND buttonAux = '0') THEN
        IF (counter > CONT_MAX-1) THEN
          counter := 0;
          buttonPressed <= '0';
        ELSE
          counter := counter + 1;
        END IF;
      END IF;
    END IF;
END PROCESS;
-----------------------------------
PROCESS(buttonPressed)
  BEGIN
    IF (buttonPressed'EVENT AND buttonPressed='1') THEN
      saida <= NOT (saida);
    END IF;
```

Como foi feito códigos separados para cada menu (seção anterior) é necessário implementar a mesma sistemática para os push-bottons, dessa forma há aqui também um demultiplexador que irá distribuir o sinal dos push-bottons para cada codigo de menu; da mesma forma que foi feito anteriormente há uma sinal de seleção que irá trabalhar concomitantemente aos sinal de seleção do LCD e assim os push-botons serão redirecionais ao menu que estará "ativo"no modulo de LCD. Na figura 8 é possível visualizar essa sistemática em nível RTL

3.1.3 Utilizando os pinos de Entrada e saída

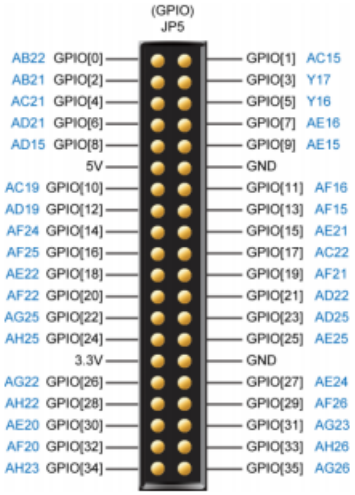
A comunicação com o circuito de interfaceamento será toda feita pela porta de comunicação GPIO embarcado no kit DE2-115; Na figura 9 é possível visualizar os rótulos para esses pinos, assim como identificar em qual pino do FPGA é conectado o GPIO correspondente.

Figura 8 – RTL dos Push-bottons



Fonte: Produzido pelo autor

Figura 9 – Pinos da porta GPIO do kit DE2-115



Fonte: (TERASIC, 2014)

Para utilizar os pinos da porta de comunicação GPIO, foi necessário apenas declara-los no *Port Map* do arquivo principal main.vhd, abaixo subscreve-se o trecho do código em que esta declaração é efetuada

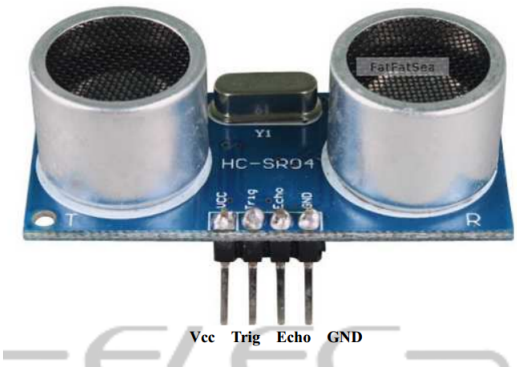
```
1  LIBRARY ieee;
   USE ieee.std_logic_1164.all;
3
   ENTITY main IS
5    GENERIC (clk_divider: INTEGER := 50000000);
     PORT  (CLOCK_50: IN BIT;
7      SW: IN STD_LOGIC_VECTOR (17 downto 0); --
       KEY: IN STD_LOGIC_VECTOR (3 downto 0);
9  --     GPIO: INOUT STD_LOGIC_VECTOR (35 DOWNTO 0);
       AD_COLOR : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
11     R_OUT,G_OUT,B_OUT: OUT STD_LOGIC;
       GPIO_ECHO : IN STD_LOGIC;
```

```
13      GPIO_TRIG : BUFFER STD_LOGIC;
        LCD_RS, LCD_RW, LCD_BLON: OUT BIT;
15      LCD_EN: BUFFER BIT;
        LCD_DATA: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
17      STEPPER: OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
        STEIRA: OUT STD_LOGIC;
19      LEDG : OUT STD_LOGIC_VECTOR (8 DOWNTO 0);    -- LEDS para DEBUG
        LEDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0) );  -- LEDS para DEBUG
21  END main;
```

3.2 O Sensor HCSR04

Para efetuar a medida de altura dos objetos, utilizou-se o sensor ultrassônico HCSR04 (ELECFREAKS, 2014) que funciona pelo principio da reflexão sonora. A partir da figura 10 podemos notar que o sensor HCSR04 possui dois modulos, o emissor e o receptor, dessa forma ao emitir um sinal ultrassônico (40Khz) inicia-se uma temporização e quando este sinal encontra um obstáculo tenderá a retornar (pelo principio da reflexão sonora) dessa forma o modulo receptor irá captar essa frequência e interrompendo de imediato a temporização previamente iniciada. Sabendo que a velocidade do som no ar é de aproximadamento 340m/s e tendo o valor em segundo da temporização é possível estipular o valor em metros do sensor até o obstáculo de que som em 40khz encontrou pelo caminho. Na figura 11 esta explicitado o gráfico do funcionamento deste sensor

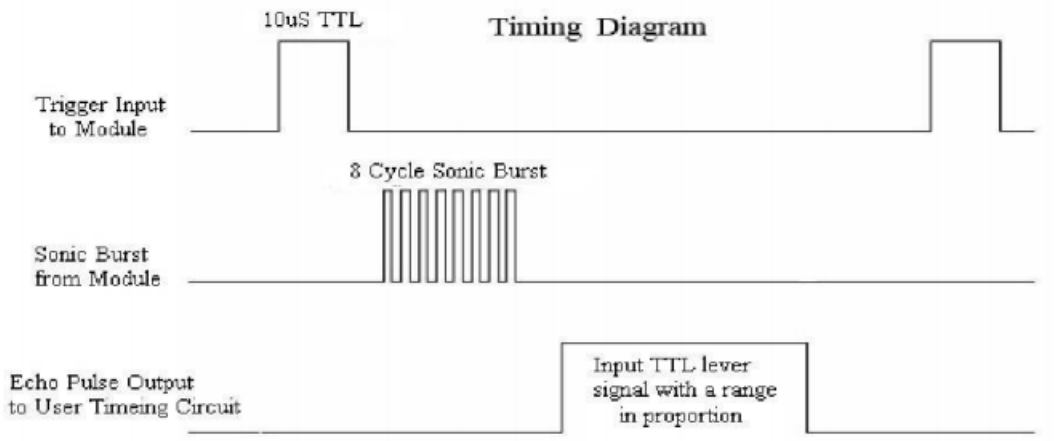
Figura 10 – Sensor HCSR04



Fonte: (ELECFREAKS, 2014)

Sabendo-se o principio de funcionamento, parte-se para a implementação do mesmo em VHDL, para isso foi elaborado uma maquina de estado (FSM) que contemple o principio de funcionamento descrito anteriormente, na figura 12 é demosntrado em RTL a maquina de estado implementada. Esta FSM funciona da seguinte maneira: o primeiro estado é um estado de espera (inicio) quando chegar um sinal de habilitação de leitura inicia-se

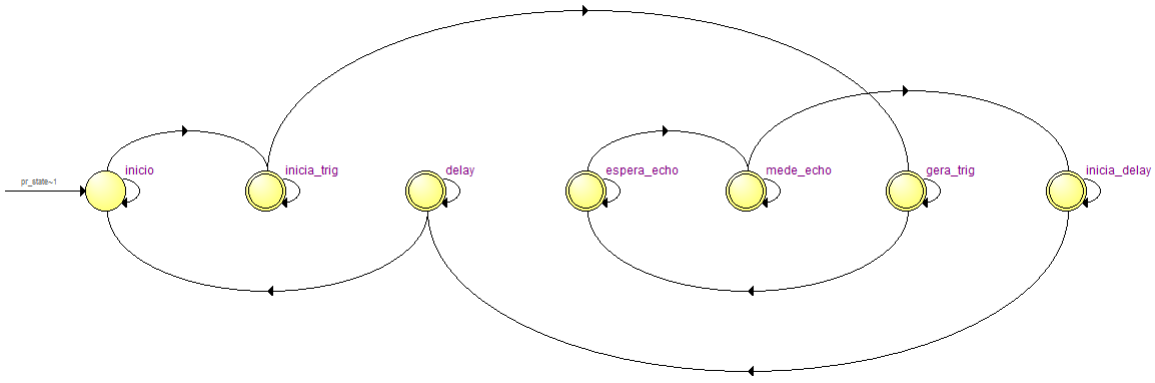
Figura 11 – Principio de Funcionamento do Sensor HCSR04



Fonte: (ELECFREAKS, 2014)

o processo indo para o próximo estado (inicia trig), no estado seguinte é levado a nível alto o pino de trigger do HCSR04 por pelo menos 60ms, dessa forma o sensor irá emitir os 8 pulsos de 40Khs e iniciar a contagem; no estado espera-echo, fazemos uma contagem de tempo, aqui vale frizar que utilizamos um relógio de 17khz obtendo assim um período de 58.82μs para cada ciclo de contagem, dessa forma se fizermos os cálculos é o tempo necessário para o som ir e voltar em uma distância de 1cm, assim se avaliarmos o tempo de espera do som contando a partir desse período de relógio, estaremos automaticamente contando a distância em unidade de cm, os estados seguintes são de um tempo de espera necessário para não haver erros no acionamento do sensor

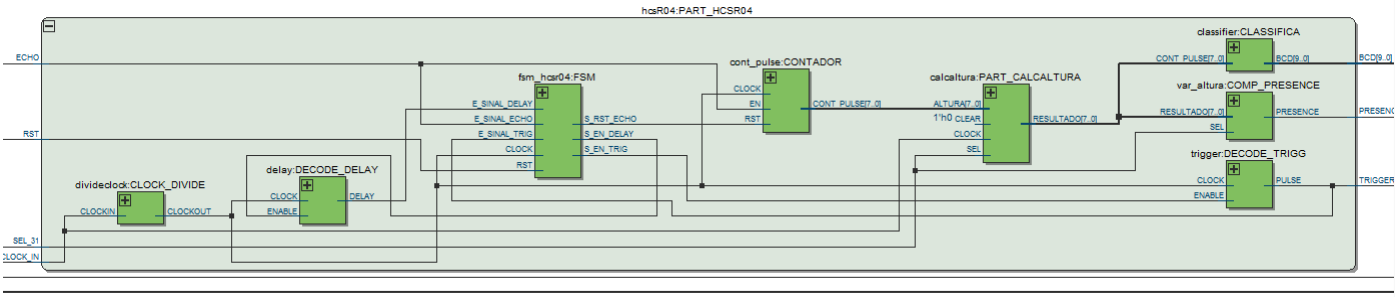
Figura 12 – Maquina de Estado do HCSR04



Fonte: Produzido pelo autor

Implementado a FSM que irá controlar a leitura do sensor foi necessário implementar os demais códigos auxiliares para o funcionamento do conjunto completo de leitura da altura, na figura 13 é possivel visualizar os blocos implementados

Figura 13 – RTL da Leitura de altura



Fonte: Produzido pelo autor

Podemos visualizar na figura 13 que saída do contador de tempo (já em unidade em cm) vai para um bloco que irá calcular a altura do objeto, pois deve-se lembrar que o sensor HCSR04 mede a distância entre o sensor e o obstáculo, para obtermos a altura é necessário marcar uma distância de referência e efetuar a subtração da nova distância medida em relação a referência configurada. Ainda há neste RTL um bloco de classificador, que irá efetuar a conversão do valor em binário para o equivalente em codigo BCD, esta conversão se faz necessária para poder mostrar na tela do LCD o valor correspondente do digito de unidade de dezena medidos, abaixo subscrevemos o codigo desta conversão *Binary to BCD*.

```
ENTITY classifier IS
  PORT(
  CONT_PULSE : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
  BCD : OUT STD_LOGIC_VECTOR (9 DOWNTO 0) );
END;

ARCHITECTURE Behavioral OF classifier IS
BEGIN
  PROCESS (CONT_PULSE)
  VARIABLE z: STD_LOGIC_VECTOR (17 DOWNTO 0);
  BEGIN
    for i in 0 to 17 loop
      z(i) := '0';
    end loop;
    z(10 downto 3) := CONT_PULSE;

    for i in 0 to 4 loop
      if z(11 downto 8) > 4 then
        z(11 downto 8) := z(11 downto 8) + 3;
      end if;
      if z(15 downto 12) >4 then
        z(15 downto 12):= z(15 downto 12) + 3;
      end if;
      z (17 downto 1) := z(16 downto 0);
```

```
25      end loop;
        BCD <= z(17 downto 8);
27    END PROCESS;
END;
```

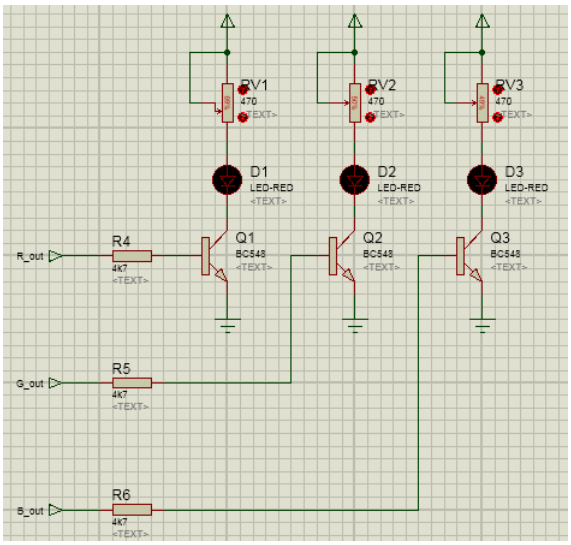
Ainda há presente um bloco que irá identificar a presença de objetos, pois no codigo de controle geral é necessário ficar esperando a presença de um objeto para iniciar todo o processo de classificação como veremos mais adiante.

3.3 O Sensor de cor

Para medir a cor do objeto utilizou-se um sensor de fabricação própria a partir de LED's (Vermelho, Verde e Azul) e um LDR. A partir do principio de reflexão luminosa, sabe-se que a a cor do objeto é justamente o comprimento de onda (cor) que este objeto não absorve, sendo os demais comprimentos de ondas absorvidos pela estrutura atômica da superfície do objeto, dessa forma ao "iluminarmos"um objeto com um cor especifica ou ele irá absorver esta cor (não sendo desta cor) ou irá refletir totalmente (sendo dessa cor).

Sabendo deste principio, projetou-se o circuito da figura 14 onde verifica-se que são apenas três transistores atuando como chave, dessa forma será possível implementar uma FSM capaz de seguir uma sequencia lógica acendendo um LED por vez e guardando o valor lido pelo LDR em um registrador de deslocamento (memória).

Figura 14 – Circuito de acionamento dos LEDS

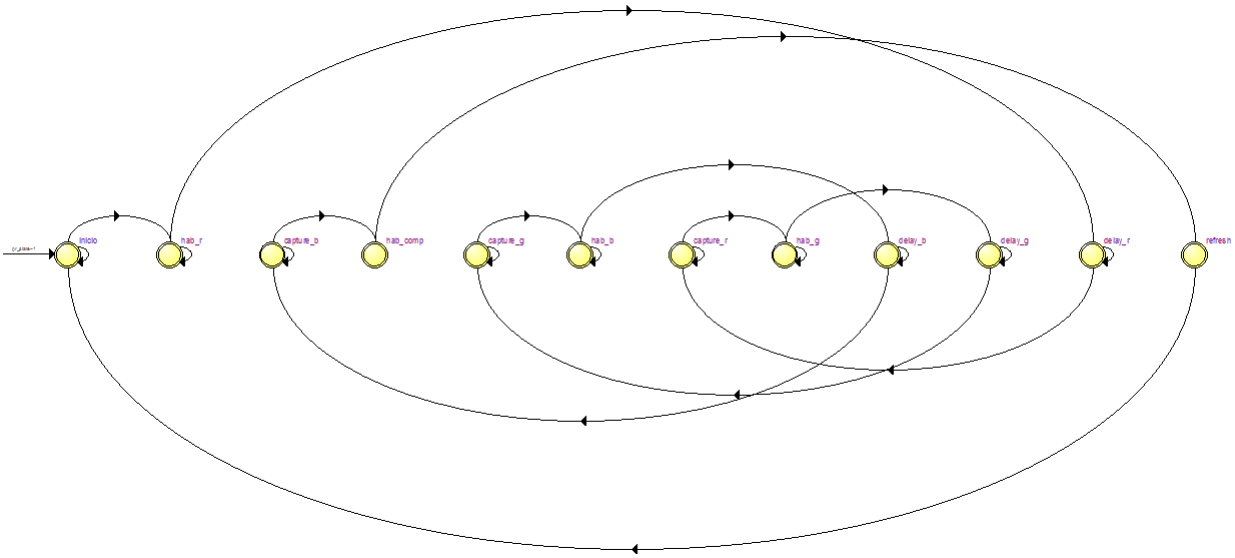


Fonte: Produzido pelo autor

Na figura 15 pode-se observar a FSM implementada para a leitura da cor. Podemos extrair desta figura que segue sempre uma sequencia de: habilitação (onde acende-se o

LED), um delay (espera de um determinado tempo) e a captura do valor do LDR que irá corresponder à intensidade luminosa refletida pelo objeto.

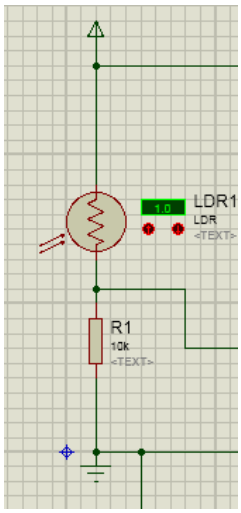
Figura 15 – FSM do sensor de cor



Fonte: Produzido pelo autor

Para capturar a intensidade luminosa utilizou-se um LDR, coforme figura 16, na configuração de divisor de tensão, dessa forma essa tensão foi inserida em um conversor Analógico/Digital (veremos este bloco na seção 3.5) e capturada na forma de vetor de bits, dessa forma quanto maior a intensidade luminosa menor é a resistividade do LDR, e por consequência sua resistência, assim o valor de tensão lido é maior.

Figura 16 – Circuito do LDR

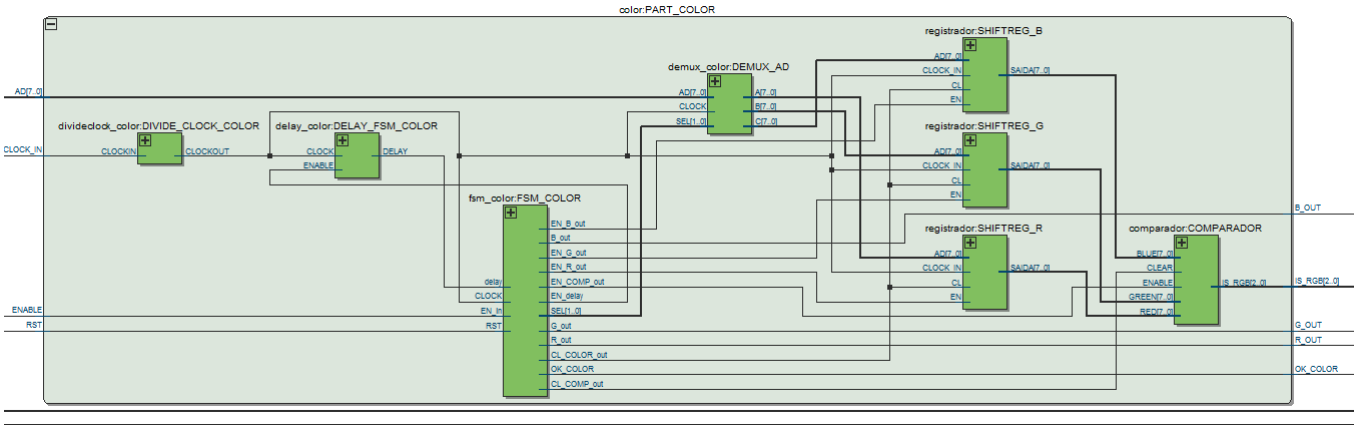


Fonte: Produzido pelo autor

Implementado a FSM que irá controlar a leitura do sensor foi necessário implementar os demais códigos auxiliares para o funcionamento do conjunto completo de leitura da cor,

na figura 17 é possível visualizar os blocos implementados, como descrito anteriormente foi necessário haver um registrador (memoria) para cada cor lida, e ao final do processo de leitura das três cores é efetuado uma comparação (verificar qual delas retornou uma tensão maior), dessa forma é possível estabelecer qual é a cor lida.

Figura 17 – RTL da Leitura de cor



Fonte: Produzido pelo autor

Nas figuras 18 e 19 é possivel visualizar o aspecto final do sensor de cor de fabricação própria.

Figura 18 – Sensor de Cor - Frontal

Fonte: Produzido pelo autor

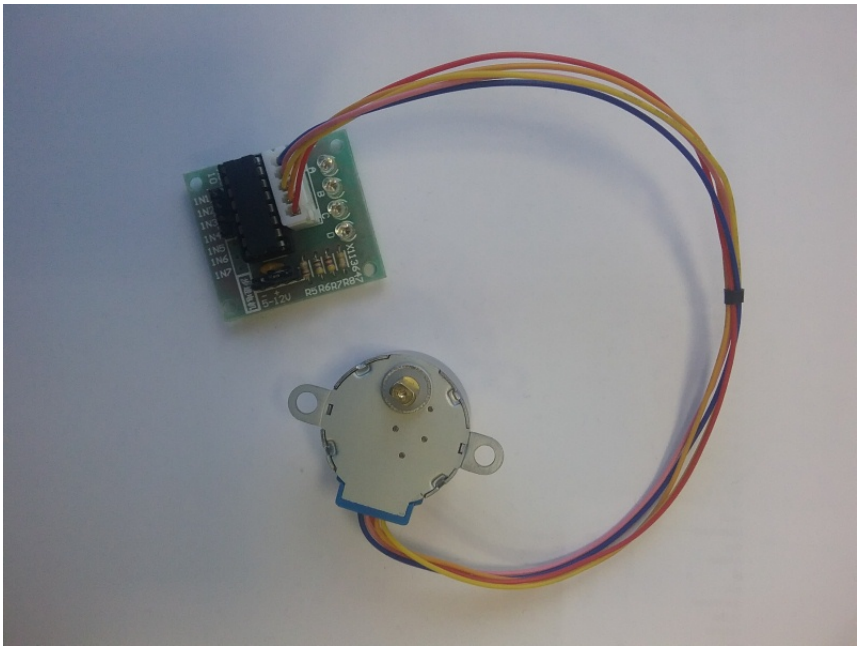
Figura 19 – Sensor de Cor - Traseira

Fonte: Produzido pelo autor

3.4 O Motor de Passo

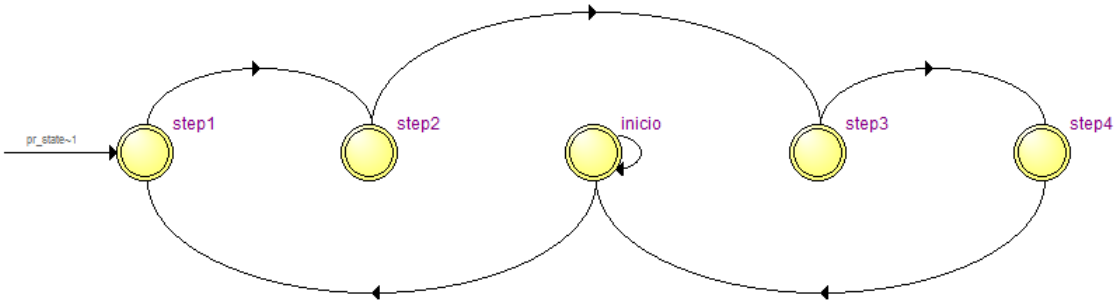
O motor de passo foi utilizado para efetuar a seleção pré-programada pelo usuário, dessa forma dependendo da escolha efetuada o motor irá girar em um sentido ou outro. O motor de passo escolhido foi o modelo: 28BYJ48 (vide figura 20) que possui alimentação em 5V, dessa forma a saída em 3.3V dos pinos de GPIO do kit DE2-115 possuem nível suficiente para excitar a entrada do driver de corrente deste motor, mais especificamente o CI UN2003.

Figura 20 – Motor de Passo



A implementação do acionamento do motor de passo em VHDL se deu através de uma FSM, conforme é mostrado na figura 21, e simplesmente há um estado em que verifica-se para qual direção o motor de passo deverá girar, dependendo dessa verificação os próximos passos indicarão o nível logico de cada bit de saída do motor de passo

Figura 21 – FSM do motor de passo



Fonte: Produzido pelo autor

Para um melhor entendimento da lógica que indica para qual direção de giro o motor de passo deve atuar, subscreve-se o código em VHDL abaixo.

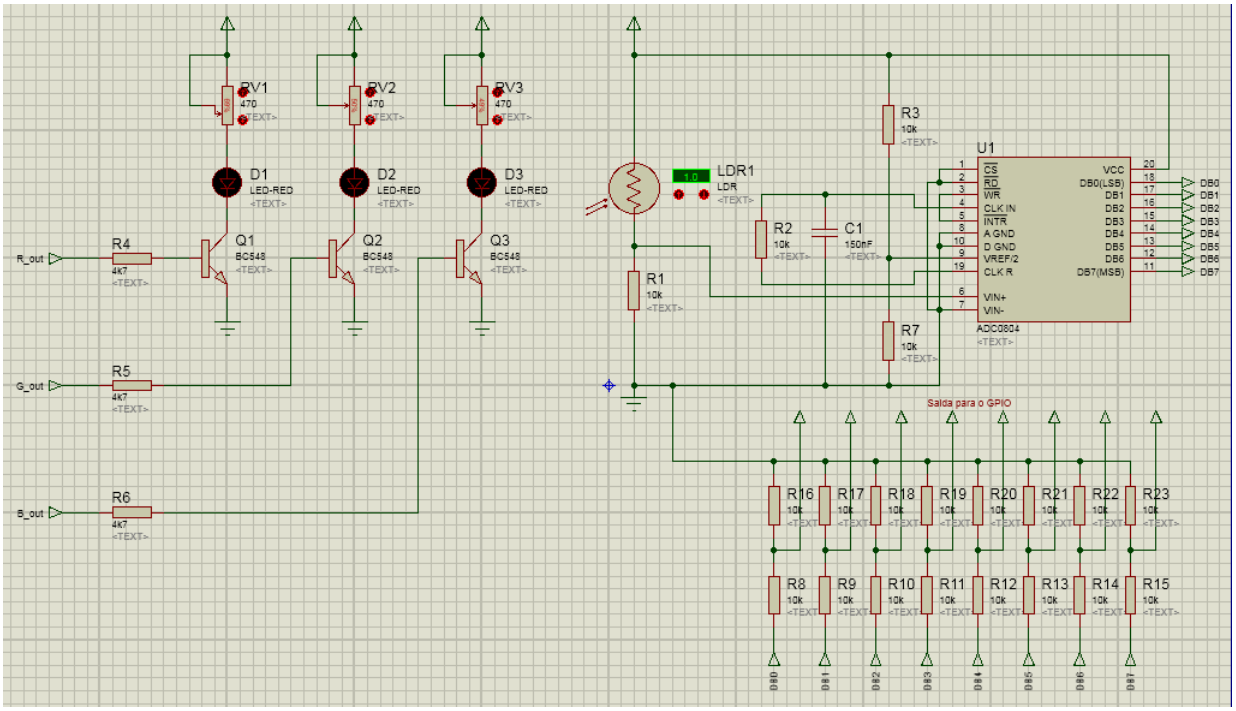
```
PROCESS (pr_state)
    BEGIN
      CASE pr_state IS
      WHEN inicio =>
        IF (ENABLE = '0') THEN
          nx_state <= inicio;
```

```
        ELSIF (ENABLE = '1') THEN
          IF (SEL_SUBMENU1 = '0') THEN
            IF (IS_RGB = "010" AND SEL_SUBMENU2 = '0') OR (IS_RGB = "001"
  AND SEL_SUBMENU2 = '1') THEN
              sentido <= '1';
            ELSIF (IS_RGB = "001" AND SEL_SUBMENU2 = '0') OR (IS_RGB = "010
  " AND SEL_SUBMENU2 = '1') THEN
              sentido <= '0';
            END IF;
          END IF;

          IF (SEL_SUBMENU1 = '1') THEN
            IF (SEL_SUBMENU32 = '0' AND COMPARADOR = '1') OR (SEL_SUBMENU32
   = '1' AND COMPARADOR = '0') THEN
              sentido <= '1'; -- Giro para Direita
            ELSIF (SEL_SUBMENU32 = '0' AND COMPARADOR = '0') OR (
  SEL_SUBMENU32 = '1' AND COMPARADOR = '1') THEN
              sentido <= '0';
            END IF;
          END IF;
          nx_state <= step1;
        END IF;
      WHEN step1 =>
        IF (sentido = '1') THEN STEPPER <= "0111";
        ELSIF (sentido = '0') THEN STEPPER <= "1110";
        END IF;
        nx_state <= step2;
      WHEN step2 =>
        IF (sentido = '1') THEN STEPPER <= "1011";
        ELSIF (sentido = '0') THEN STEPPER <= "1101";
        END IF;
        nx_state <= step3;
      WHEN step3 =>
        IF (sentido = '1') THEN STEPPER <= "1101";
        ELSIF (sentido = '0') THEN STEPPER <= "1011";
        END IF;
        nx_state <= step4;
      WHEN step4 =>
        IF (sentido = '1') THEN STEPPER <= "1110";
        ELSIF (sentido = '0') THEN STEPPER <= "0111";
        END IF;
        nx_state <= inicio;
      END CASE;
  END PROCESS;
```

3.5 Interface Eletrônica

Dado o tratamento adequado a cada subconjunto do projeto, é necessário agora integra-los a uma interface que irá capturar os sinais, tratar e se comunicar com o kit DE2-115 através dos pinos da porta de GPIO.

Figura 22 – Circuito da Interface Eletrônica



Fonte: Produzido pelo autor

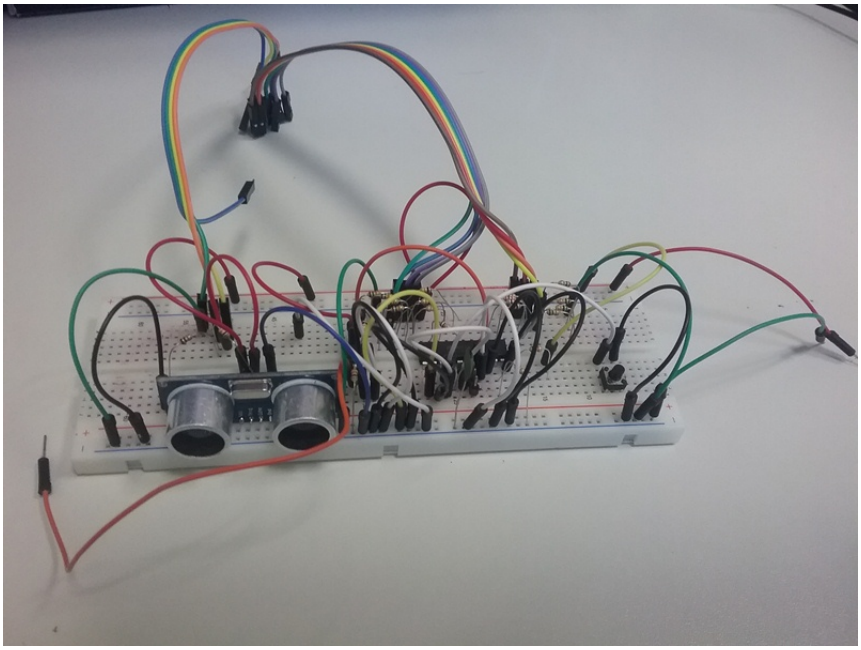
Como podemos perceber pela figura 22 para capturar o sinal de tensão (que corresponde a intensidade luminosa do LDR) utilizou-se de um conversor Analógico-Digital o CI AD0804, que tem a função de converter um nível de tensão em um valor correspondente de binário (pelas saída DB0 até DB7). como o nível de saída deste conversor é de 5Vdc foi necessário fazer um divisor de tensão, rebaixando a tensão de 5Vdc para 3Vdc, pois o nível de tensão de entrada dos pinos da porta GPIO do kit DE2-115 toleram no máximo um nível de 3,3Vdc.

O Aspecto final da montagem em protoboard da interface, é apresentado na figura 23.

3.6 O código VHDL final

Implementado cada conjunto VHDL so sistema (Sensor de Altura, Sensor de Cor, Motor de Passo e Interface Eletrônica) foi necessário unir esses blocos em um arquivo principal (apresentado no seção 2.2.1) e além disso criou-se mais um arquivo para controle

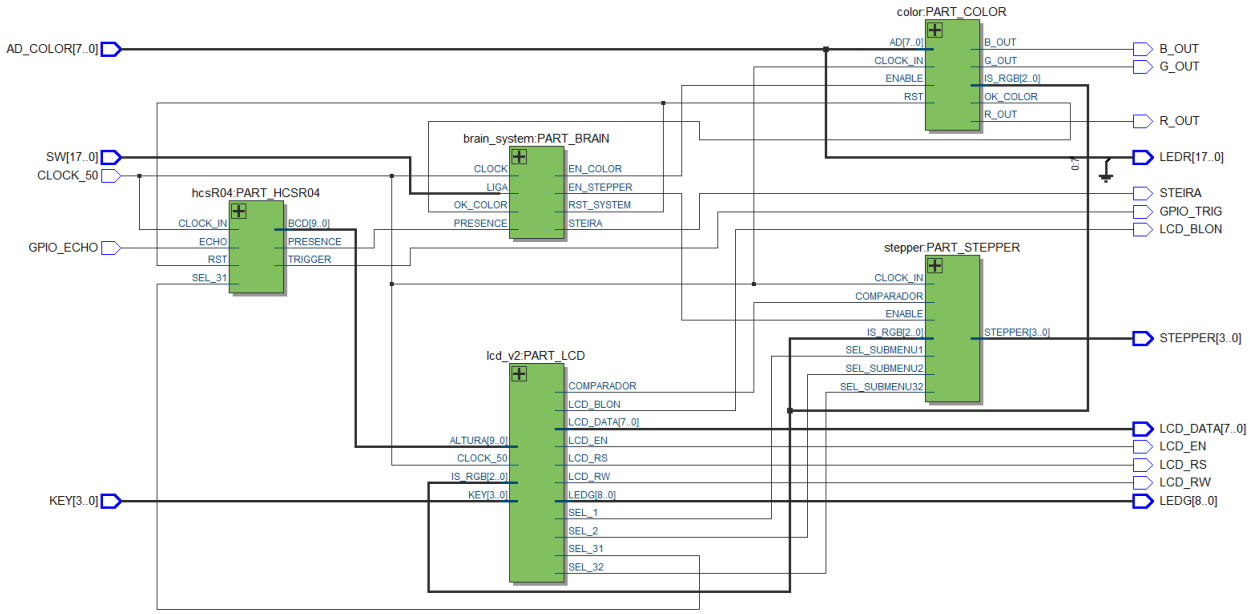
Figura 23 – Montagem da Interface Eletrônica



Fonte: Produzido pelo autor

de processamento do sistema. na figura **??** pode-se notar todos os blocos apresentados anteriormente unidos pela biblioteca *WORK* do VHDL

Figura 24 – RTL do sistema completo

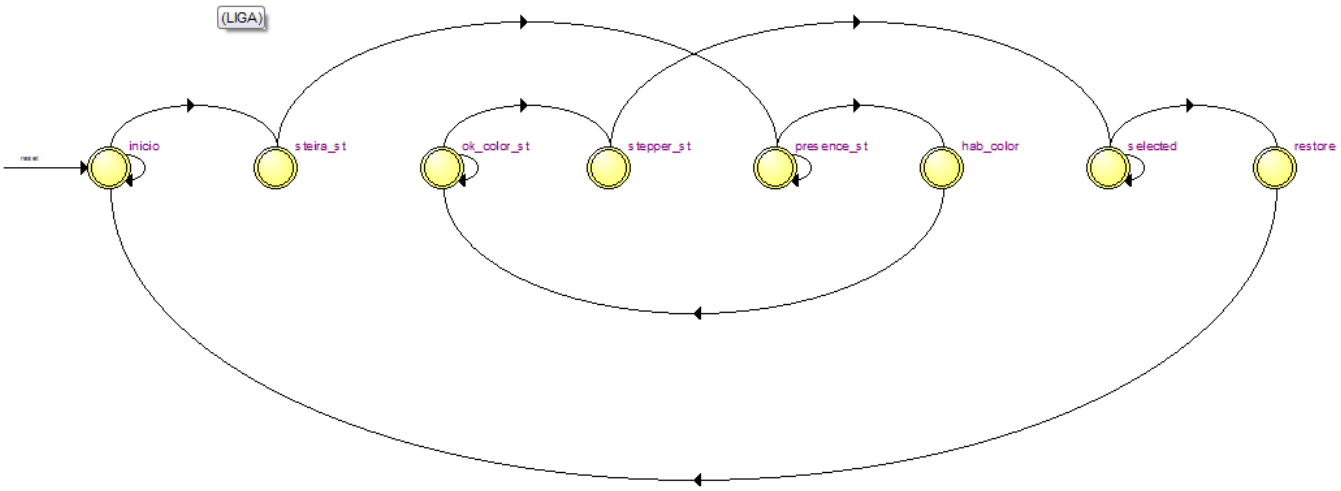


Fonte: Produzido pelo autor

3.6.1 Bloco de controle e processamento central

Para se tornar um sistema completo, foi necessário a implementação de uma FSM de controle e processamento, que dará a todo o conjunto a rotina de processamento, na figura 25 podemos visualizar a FSM desenvolvida para processar o sistema.

Figura 25 – FSM de controle e processamento



Fonte: Produzido pelo autor

Como podemos notar na figura 25 existe um estado inicial em que se aguarda o botão de inicio do processo, assim que acionado pelo operador o inicio do processo, o proximo passo á acionar a saída que ligará uma esteira (afim de trazer objetos para a leitura), consequentemente é necessário ficar verificando (no proximo estado "$presence_{st}$") se o objeto esta na posição de leitura, quando isso ocorre imediatamente desliga-se a esteira e hablita-se a bloco de leitura de cor - vale salientar que o bloco de leitura de altura esta continuamente em funcionamento, sendo utilizado também como o sensor de presença - efetuada a leitura da cor, sabe-se disso por um sinal de "ok"oriundo do bloco do sensor de cor, habilita-se o bloco do motor de passo que irá fazer a seleção (direita ou esquerda) quando não for mais detectado objeto (pelo sensor de altura) repete-se todo o processo.

Afim de explicitar melhor este processamento principal, subscreveu-se um trecho desta FSM abaixo.

```
PROCESS (pr_state)
    BEGIN
      CASE pr_state IS
        WHEN inicio =>
          RST_SYSTEM <= '1';
          IF (LIGA = '0') THEN
            nx_state <= inicio;
          ELSIF (LIGA = '1') THEN
            nx_state <= steira_st;
```

```
10            END IF;
          WHEN steira_st =>
12            STEIRA <= '1';
              nx_state <= presence_st;
14        WHEN presence_st =>
              IF (PRESENCE = '0') THEN
16              nx_state <= presence_st;
              ELSIF (PRESENCE = '1') THEN
18              nx_state <= hab_color;
              END IF;
20        WHEN hab_color =>
              STEIRA <='0';
22            EN_COLOR <='1';
              nx_state <= ok_color_st;
24        WHEN ok_color_st =>
              IF (OK_COLOR = '0') THEN
26              nx_state <= ok_color_st;
              ELSIF (OK_COLOR = '1') THEN
28              nx_state <= stepper_st;
              END IF;
30        WHEN stepper_st =>
              EN_STEPPER <= '1';
32            nx_state <= selected;
          WHEN selected =>
34            IF (PRESENCE = '1') THEN
                nx_state <= selected;
36            ELSIF (PRESENCE = '0') THEN
                nx_state <= restore;
38            END IF;
          WHEN restore =>
40            EN_COLOR <= '0';
              EN_STEPPER <= '0';
42            nx_state <= inicio;
      END CASE;
44  END PROCESS;
```

4 Resultados Obtidos

Afim de sistematizar o processo, durante todo o desenvolvimento ia-se testando cada implementação via a gravação da compilação do codigo VHDL, diretamente no kit DE2-115.

4.1 Resultados Obtidos

4.1.1 Menus do Usuário

Ao todo o sistema contêm 5 menus (menu de seleção da classificação, menu da cor, menu da altura, menu do peso, e visualização Geral), em todos os menus se obteve exito na implementação e teste, a troca entre os menus é efetuado pelo push-botton (KEY[0]) presente no kit. A seguir será explanado a função de cada menu.

O menu de seleção da classificação é o primeiro apresentado ao ligar os sistema, nele o usuario escolhe o tipo de classificação que o sistema atuará, ou pela cor ou pela altura. A troca dessa escolhe é efetuada pelo botão (KEY[1]) presente no kit.

No menu da cor é apresentado a última cor lida pelo sistema, independente do modo de classificação escolhido anteriormente. Há neste menu a possibilidade, se caso o sistema esteja trabalhando em modo de classificação por cor, da escolha do sentido do giro do motor de passo quando a cor for a verde, esta escolha é feita pelo botão (KEY[1]) presente no kit.

O menu altura é um dos mais importantes do sistema, nele esta a opção de programar (SETAR) a referência da altura; em todos os instantes de funcionamento o sensor de altura HCSR04 esta sendo acionado, porém quando neste menu não estiver presente a palavra 'SETADO' a leitura que se obtêm é a simples distância entre o sensor e o objeto, no momento em que é SETADO a medida de altura, neste menu, a medida obtida a partir de então será a diferença entre o valor setado e os próximos valores lidos, isto resultará na medida de altura relativa em relação ao valor setado, esta programação é feita pelo botão (KEY[1]). Ha outras duas opções neste Menu, uma delas é a que se refere quanto a escolha do sentido de giro do motor de passo (Escolha efetuada pelo botão (KEY[2]) quando a altura medida for maior que uma altura escolhida previamente pela proxima opção que é justamente escolher a altura referência para a classificação por altura, escolha esta efetuada pelo botão (KEY[3]).

O menu peso esta presente no sistema porém a implentação do circuito eletrônico com a celula de carga (para medir altura) não foi realizada.

Na parte de Visualização geral não há nenhuma seleção, apenas é mostrado todas as leituras que estão sendo realizadas no instante pelo sistema proposto.

4.1.2 Medidas dos Sensores

O sensor de altura apresentou resultados satisfatórios em suas leituras, sabendo que estamos operando em uma frequência de leitura em 17Khs, sendo que a precisão para esta frequência é de 1cm, o erro aproximado de leitura obtido esteve abaixo desta própria unidade de medida.

O sensor de cor também apresentou resultados satisfatórios. Vale salientar que para que haja acerto na leitura dos níveis de tensão que corresponderão a cor do led que esta sendo acionado, é necessário ajustar a intesidade luminosa dos leds afim que todos emitam a mesma potencia luminosa, desta forma esta garantido que a cor do led acionado terá o mesmo nível de tensão para as demais cores.

4.1.3 Classificador - Motor de passo

O motor de passo foi acionado com exito, porém vale salientar que para este projeto foi escolhido o motor modelo 28BYJ48, que apresentou baixa velocidade de giro. verificou-se experimentalmente que a frequencia de acionamento maxima do motor fica em torno dos 250Hz, o que o deixa com uma velocidade gira muito baixa para esta aplicação. Em aplicações futuras recomenda-se um motor de passo com menos precisão e maior velocidade de giro, assim a classificação será mais rápida.

5 Conclusão e Considerações Finais

O grande objetivo deste projeto é integrar areas de conhecimentos, para a resolução de um problema especifico. No decorrer do desenvolvimento deste projeto não foi diferente, utilizou-se ferramentas das principais areas da eletrônica, o VHDL como ferramente poderosíssima da eletronica digital, e os circuitos discretos para o tratamento dos sinais como ferramentas da eletronica analogica.

5.1 Dificuldades enfrentadas

Umas das principais dificuldades foi a alta inacessibilidade ao kit DE2-115, era possível utilizar o kit para testes apenas em momentos específicos (durante as aulas de Projeto Integrador II) por um período aproximado de 2 horas semanais, com isso muito do desenvolvimento foi prejudicado pois quando chegava-se ao final de uma implementação, tinha-se que esperar até o momento pré-agendado para os testes, e isso acabou acarretando em um tempo de desenvolvimento maior e ao final com poucos testes.

5.2 Conclusões

O desenvolvimento deste projeto foi de extrema valia para a formação academica, alêm de dar um enorme subsidio em linguagem VHDL, foi possivel verificar na prática a integração entre duas grandes áreas da eletrônica, a Digital e a Analógica.

Na parte da eletrônica digital, foi possível se familiarizar com uma das mais modernas e poderosas ferramentas nesta área o VHDL e os FPGA's, criu-se um sistema digital completo com base apenas em uma linguagem de descrição de hardware.

Na parte da eletrônica analogica, foi possível observar como se dá o devido tratamento a grandezas físicas que nos cercam, neste caso tratou-se apenas de distancias e cores (luz), mas foi possivel observar que há a necessidade de transdutores e a utilização de circuitos para tratamento destes sinais e posteriormente, se forem tratados por sistemas digitais, sua conversão para o mundo digital (bits).

Por fim, a realização deste trabalho engrandeceu a concepção pré-existente de sistemas digitais embarcados. A noção concebida pela utilização do VHDL para a implementação deste sistema será de extrema valia para o restando da vida academica e profissional.

Referências

ELECFREAKS. *HCSR04 Sensor Ultrassônico*. [S.l.], 2014. Disponível em: <http://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf>. Citado 2 vezes nas páginas 26 e 27.

PEDRONI, V. *Circuit Design with VHDL*. Books24x7.com, 2004. ISBN 9780262162241. Disponível em: <https://books.google.com.br/books?id=b5NEgENaEn4C>. Citado 3 vezes nas páginas 11, 14 e 15.

TERASIC. *DE2-115 Kit de desenvolvimento*. [S.l.], 2014. Disponível em: <ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE2-115/DE2_115_User_Manual.pdf>. Citado 6 vezes nas páginas 11, 19, 20, 21, 23 e 25.