Implementação em FPGA de robô autônomo seguidor de linha

Arthur S. Gonzaga
Engenharia Eletrônica
Universidade de Brasília
Campus Gama
Email: arthursgonzaga@gmail.com
Matrícula: 14/0016775

Leonardo Brandão B. de Freitas
Engenharia Eletrônica
Universidade de Brasília
Campus Gama
Email: leonardobbfga@gmail.com
Matrícula: 14/0025197

Victor Martins N. Luz
Engenharia Eletrônica
Universidade de Brasília
Campus Gama
Email: vcnluz@gmail.com
Matrícula: 14/0164910

***Resumo*—Relatório de trabalho final para a disciplina de Projetos com Circuito Reconfiguráveis, abordando a concepção de um robô autônomo seguidor de linha utilizando uma FPGA como módulo de controle.**

I. Introdução

A automação é de certo um recurso dos mais importantes para a aceleração da produção industrial. O uso de dispositivos robóticos tem garantido maior precisão de execução de atividades (inclusive em rotinas demasiadamente repetitivas) e disponibilidade de assumir mais turnos de trabalho em comparação com a mão-de-obra humana.

O Sistema de Armazenagem Automático tornou-se muito comum dentro dos estoques industriais por reduzir o tempo de movimentação de mercadorias e aumentar a eficiência dessas movimentações, proporcionando redução de gastos, aumento da produtividade e melhoria dos processos logísticos da empresa.

Segundo a *Amazon*, varejista do *e-commerce* americano, a operação de robôs autônomos iniciada em alguns de seus centros de distribuição a partir de 2012 aumentou a produtividade dessas unidades em 3 ou 4 vezes.[1]

"O robô recebe ordens de um computador para buscar objetos num armazém. Pequeno, pode passar sob a mercadoria armazenada quando está sem carga, o que evita o congestionamento nos corredores. Quando encontra o volume desejado, faz um movimento de rotação para fixar-se à pilha de caixas. Depois, segue para a avenida mais próxima e transita pelo menor caminho até o destino. Lá, um humano recebe o produto para embalar e despachar."[2]

A capacidade de controle de locomoção e correção de curso desses sistemas é um ponto crítico de sua operação. O bom desempenho dessa funcionalidade permite ao sistema alcançar seu ideal de eficiência, passando a contar minimamente com intervenções de correção externas à unidade.

Para tanto, o robô necessita identificar a condição de sua operação no momento, e avaliar o que deve ser feito para que se atinja o cenário ideal definido para seu funcionamento. Isso é caracterizado na teoria de controle como um sistema de malha fechada, que parte da comparação de seu estado atual com uma situação desejada para corrigir a saída entregue pelo conjunto.

Em uma aplicação seguidora de linha, o controle tem como situação ideal a detecção do traçado por um sensor central, indicando o alinhamento do conjunto com o traçado definido. Neste caso, o robô segue o trajeto sem efetuar correções na planta (motores). Eventuais correções precisam acontecer se este traçado deixa de ser detectado pelo sensor central (ou apenas por ele), indicando a saída do sistema para a direita ou esquerda. A partir desta avaliação, a lógica do controle deve definir as correções a serem efetuadas pelos motores para trazer de volta o conjunto ao percurso.

Ao estabelecer valores diferentes de rotação para o conjunto de motores, é possível efetuar a correção de curso da unidade. A lógica de ajuste é inversa ao posicionamento e resultados de detecção dos sensores. Caso o sensor mais à direita detecte o traçado, isso indica uma saída à esquerda do robô, e o motor da roda à esquerda precisa ter uma rotação maior que a da roda oposta para efetuar a correção do trajeto. De forma análoga pode-se descrever a correção para uma saída à direita.

Ambos os casos são esquematizados na imagem a seguir.

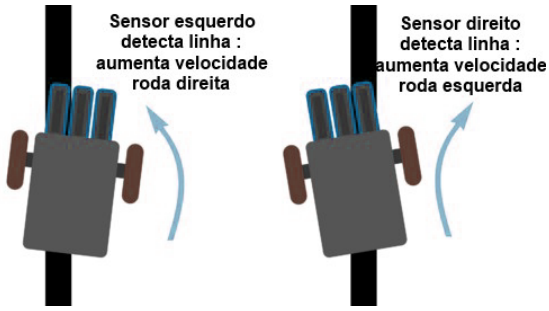

Figura 1. Exemplificação da correção de curso do seguidor de linha.

Da visão geral do sistema, pode-se destacar três grandes necessidades a serem integradas de modo a desempenhar o movimento e correção de curso do robô. São eles:

- A leitura de sensores;
- A avaliação do cenário (interpretação de dados);
- O controle de motores.

Dessa forma, portanto, também ficou dividida a circuitaria a ser desenvolvida para o projeto.

A. Fundamentação Teórica

1) Detecção óptica por infravermelho: Um sensor óptico reflexivo é um conjunto emissor-receptor de luz. O emissor emite um comprimento de onda infravermelho (IR), que, ao refletir em uma superfície, tem sua intensidade alterada de acordo com as características de reflexão da superfície. Essa variação é percebida pelo receptor, um fototransistor, que converte esse estímulo em valores de tensão, que também são distinguíveis de acordo com as características de reflexão do material.

Os extremos da variação ocorrem para superfícies pretas ou brancas, sendo então essas as cores mais utilizadas para a criação de trajetos para dispositivos de detecção que se utilizam desse método de sensoriamento.

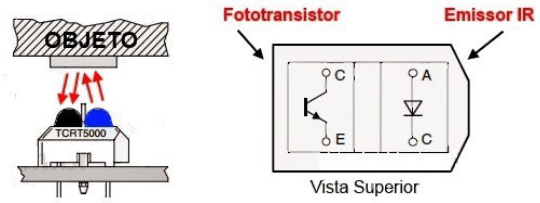

Figura 2. Esquema de funcionamento de um sensor de reflexão IR.

2) Operação de motores e PWM: A manutenção do torque para várias velocidades de rotação em motores DC é essencial para a sua operação. Essa força mecânica do motor é ajustada com variação de tensão, o que pode ser feito linearmente com um resistor variável. No entanto, esse método não mantém a velocidade em baixas rotações e não proporcionam uma partida leve para o motor, que dispara a partir do instante em que é fornecida a tensão requerida para sair da inércia.[3]

O uso do PWM para controle de motores DC se popularizou por ser capaz de lidar com essas questões que o ajuste linear apresenta desvantagens. Advindo do inglês *Pulse Width Modulation*, a modulação por largura de pulso é uma ferramenta bastante utilizada em circuitos de potência, tais como retificadores e inversores, chaveamento de motores e geradores, controle de iluminação e temperatura e outras diversas aplicações nas áreas de eletrônica. O princípio do PWM é controlar através de chaveamentos a potência média aplicada a uma determinada carga. [4].

O seu funcionamento se dá da seguinte maneira: imagina-se uma chave totalmente aberta, esta chave não permite passagem corrente, ou seja, a potência média entregue na carga é de 0%; o caso contrário, da chave totalmente fechada, permite total passagem de corrente, isto é, a potência média dissipada na carga será de 100%; uma situação intermediária é quando utiliza-se de uma regulação desta chave, ora abrindo, ora fechando, desta forma, a potência dissipada depende da largura do pulso. Caso o pulso seja dado com 50% do tempo aberto e 50% do tempo fechado, a potência média entregue seria de 50% da potência total.

Abaixo, segue a figura 3, que exemplifica o que foi explicado anteriormente:

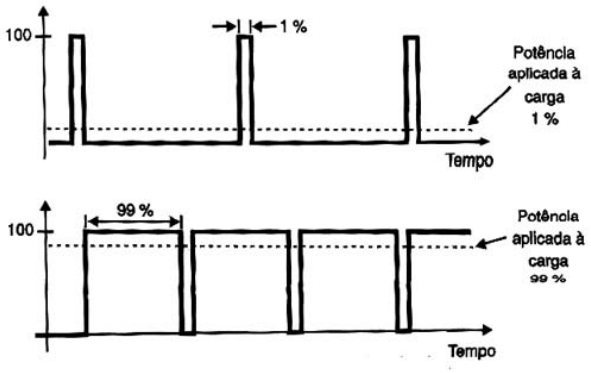

Figura 3. Exemplificação da teoria do PWM. Fonte: http://www.newtoncbraga.com.br/index.php/robotica/5169-mec071a

Essa relação de nível de tensão e entrega de potência é esquematizada pelo *duty*, que consiste na porcentagem de nível lógico alto presente no sinal. O valor de *duty*, caso trabalhado em resolução de 4 bits, pode ser ajustado para os valores apresentados na imagem a seguir:

DUTY	$\frac{nº\ Pulsos}{nº\ Total\ de\ Pulsos}$	DUTY%
"0000"	1/16	6.25%
"0001"	2/16	12.50%
"0010"	3/16	18.75%
"0011"	4/16	25.00%
"0100"	5/16	31.25%
"0101"	6/16	37.50%
"0110"	7/16	43.75%
"0111"	8/16	50.00%
"1000"	9/16	56.25%
"1001"	10/16	62.50%
"1010"	11/16	68.75%
"1011"	12/16	75.00%
"1100"	13/16	81.25%
"1101"	14/16	87.50%
"1110"	15/16	93.75%
"1111"	16/16	100%

Figura 4. Relação do *duty cycle* e da proporção do período (T) ocupado pelo pulso para um sistema de instruções binário de 4 bits.

De modo a suprir a corrente demandada pelos motores, o sinal de *PWM* gerado pelo controlador apenas coordena um circuito do tipo *ponte H*, evitando sobrecarga sobre o controlador, que comumente funciona com baixa demanda de corrente. O circuito de ponte H ainda define a polaridade do motor (sentido de rotação), o que também pode ser controlado.

Em linhas gerais, o dispositivo lógico controla o circuito de potência da ponte H para operar a planta, que por sua vez o isola das grandes demandas de corrente dos motores.

II. Procedimentos de Projeto

A. Idealização dos estados de funcionamento

De modo a dar autonomia ao robô, o seu sistema precisa identificar a sua situação em relação ao traçado - e

reagir a isso. Esse estado de leitura precisa ser verificado constantemente, de modo a tratar as variações do traçado conforme a locomoção da unidade. A partir dos resultados de leitura de sensores obtidos nesse estado, há uma decisão de ação caracterizada pela forma de operação dos motores a ser realizada. Esses estados de atuação regem a locomoção e a correção de curso do autônomo - isto é - geram as saídas de controle para que os motoroes mantenham o curso ou ajustem para a esquerda ou para a direita o deslocamento do robô.

Tomada a decisão por um dos estados de atuação, a leitura precisa continuar sendo verificada. Cada estado, portanto, retorna à leitura, mantendo as saídas atuais até uma possível modificação de estados regida por uma mudança de leitura de sensores.

A máquina de estados do sistema foi esboçada na figura a seguir. As entradas *Sl, Sc* e *Sr* são os valores binários da detecção dos sensores da esquerda, centro e direita do veículo, respectivamente.

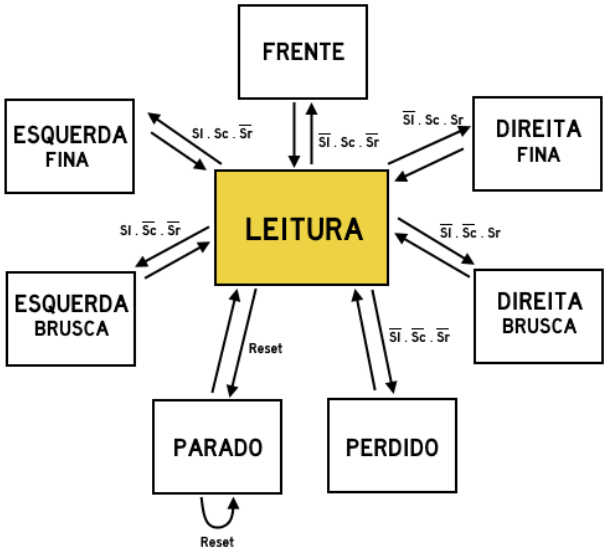

Figura 5. Máquina de estados finitos do sistema.

Os estados de atuação que efetuam correções para a esquerda ou direita se distinguem em dois tipos: *Fino* e *Brusco*. Os estados do tipo *Fino* são consequência da detecção de um leve desvio de curso, e não requerem uma grande diferença de rotação entre os motores para realinhar o robô com o curso do trajeto. Esses estados detectam a linha pelo sensor do centro e do lado oposto para o qual o curso errático está tendendo.

Já os estados do tipo *Brusco* detectam a linha apenas pelo sensor do lado oposto ao do sentido errático do sistema, não mais pelo sensor central, indicando maior desvio da rota. Para que se aplique a correção, exige-se nesses estados maior diferença de rotação entre os motores.

B. Sensoriamento e leitura de posição

De modo a detectar o traçado da linha, o sistema se utiliza de 3 sensores ópticos infravermelhos TCRT5000, um ao centro, e dois tendendo a cada um dos lados do veículo. O fototransistor integrado a cada um deles é ativado pela intensidade de reflexão da luz infravermelha, permitindo a passagem de corrente conforme o estímulo recebido.

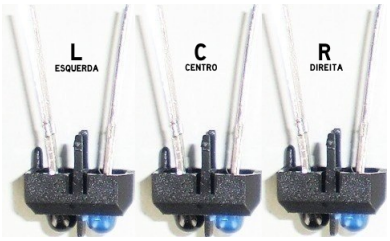

Figura 6. Disposição do conjunto de sensores.

O funcionamento básico dos sensores, no entanto, foi digitalizado para a aplicação com a placa FPGA. A conexão dos sensores demandou o uso associado de resistores, formando um circuito ligado aos barramentos *JB* da placa *Basys3*.

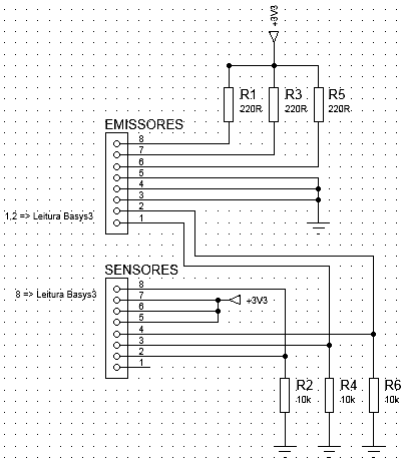

Figura 7. Esquemático do circuito dos sensores e ligação com a FPGA.

No esquemático acima, notamos 2 barramentos com 8 pinos cada que serão utilizados para conectar os sensores ópticos reflexivos TCRT5000 a seus respectivos resistores, bem como para enviar os sinais de saída para a leitura e processamento na FPGA. Esses barramentos representam a conexão *JB* da placa.

No barramento denominado *EMISSORES* tem-se a alimentação proveniente da FPGA de 3.3V e 3 resistores de 220 *ohms* que limitam a corrente para o LED emissor de luz infravermelho (pinos 8, 7, 6), bem como os pinos 5,4 e 3 - que fecham o circuito no *ground* - ativando os LEDs. Além da alimentação dos LED's infravermelhos, o barramento EMISSORES tem seus pinos 1 e 2 utilizados para enviar um sinal digital referente à reflexão dos sensores para FPGA, a partir de um divisor de tensão entre os fototransistores do módulo TCRT5000 e os resistores de 10K *ohms*.

No barramento *SENSORES* tem-se a alimentação dos fototransistores com 3.3V fornecidos pela placa, além do fechamento do circuito (configurando o divisor de tensão) nos resistores de 10K *ohms*, ligados ao ground do circuito. Além

disso, o Pino 8 desse barramento é utilizado como uma terceira entrada de sinal digital de leitura para a FPGA.

O cenário ideal para o sistema - com o robô alinhado com o traçado - se caracteriza pela detecção da linha apenas pelo sensor central. Os sensores laterais (esquerdo e direito) são necessários para a identificação de saída de curso. Caso um deles passe a detectar o traçado, a informação de desalinhamento é processada pela máquina de estados, que configura a planta (motores) para realizar a correção de curso.

O sinal enviado pelo sensor central é admitido na lógica desenvolvida como *Sc*, o do sensor à esquerda, *Sl* e do à direita, *Sr*.

C. Desenvolvimento da lógica em VHDL

1) Módulo sensor_read: O módulo de leitura dos sensores foi descrito de modo a detectar, em um processo, os níveis lógicos das entradas dos três sensores (*left* (esquerda), *center* (centro) e *right* (direita)). A detecção do traçado ocorre quando o nível lógico alto ('1') é percebido por qualquer um dos sensores, isso corresponde à detecção de uma superfície da cor branca.

Realizada a detecção, a informação de estado dos sensores é unida em um vetor de 3 posições, a saída *sensor_value*. A seguir, são listadas as possíveis leituras dos sensores (*Sl*, *Sc* e *Sr*), a respectiva saída *sensor_value* e a interpretação espacial dessa informação, que caracteriza como o robô se encontra no momento.

Tabela I
Leituras de sensor_read e situação espacial interpretada

Sl	**Sc**	**Sr**	**sensor_value**	**Interpretação**
0	0	0	000	Perdido do trajeto
0	0	1	001	Saiu à esquerda
0	1	0	010	Alinhado com a linha
0	1	1	011	Saindo à esquerda
1	0	0	100	Saiu à direita
1	0	1	111	**Leitura inválida**
1	1	0	110	Saindo à direita
1	1	1	111	Cruzamento no trajeto

2) Módulo da máquina de estados: A descrição de hardware para a máquina de estados finitos da lógica do robô autônomo procurou seguir o planejamento feito na etapa de idealização, como visto na Figura 5. O dado vetorial *sensor_value* é entrada da *FSM*.

Esse dado é lido no estado de leitura e, a depender da sua configuração, um dos estados de atuação é definido. Os casos de cruzamento de trajetos ou de eventuais leituras inválidas (retorno de sensor_value é '111') foram tratados de forma que o robô seguisse em frente.

A seguir, verifica-se a correlação dos valores de *sensor_value* (agora uma entrada da FSM) e a decisão de estados feita após a passagem pelo estado de leitura.

Tabela II
Passagem de estados da FSM de acordo com a leitura dos sensores

sensor_value	**Decisão de estado**
000	*PERDIDO*
001	*DIREITA BRUSCA*
010	*FRENTE*
011	*DIREITA FINA*
100	*ESQUERDA BRUSCA*
110	*ESQUERDA FINA*
111	*FRENTE*

Nos estados de atuação, configura-se o sentido de rotação de cada motor e também o *duty* do *PWM* desses. Vale recordar que o ajuste de *duty* reflete-se na velocidade de rotação adotada pelo motor.

O sentido de rotação de cada motor é definido por um vetor de duas posições, a ser mapeado para dois pinos de saída da placa. Os valores binários presentes nesses vetores têm nível lógico alternado e são utilizados para reger a ponte H dos motores, e ajustam a polaridade. A alternância se dá para evitar curtos. Sua interpretação se dá como na tabela a seguir:

Tabela III
Controle do sentido de rotação dos motores

Controle de polaridade	**Sentido de rotação**
'00'	Desligado
'11'	Ligação Inadequada
'10'	**Frente**
'01'	**Trás**

Os estados '00' e '11' não são sequer implementados na lógica da máquina de estados e constam na tabela acima apenas para que se considere todas as possibilidades de ligação.

Ainda como saídas dos estados da FSM, há dois dados de 4 *bits* que setam o *duty* dos geradores de *PWM* dos motores. A configuração de *duty* utilizada segue o padrão apresentado na Figura 4.

A listagem de estados e as saídas especificadas para cada um deles e para cada motor pode ser verificada nas tabelas a seguir. O estado de leitura mantém as saídas do estado anterior, havendo mudança apenas se ocorrer a transição para um estado de atuação diferente.

Tabela IV
Especificação das saídas de acordo com o estado da FSM para o Motor Direito (R)

Estado	**Sentido do Motor R**	**Duty_R**	**Duty_R%**
PERDIDO	'10' (Frente)	'0111'	50%
FRENTE	'10' (Frente)	'1111'	100%
DIREITA BRUSCA	'01' (Trás)	'1100'	81,25%
DIREITA FINA	'01' (Trás)	'0111'	50%
ESQUERDA BRUSCA	'10' (Frente)	'1100'	81,25%
ESQUERDA FINA	'10' (Frente)	'0111'	50%

Tabela V
Especificação das saídas de acordo com o estado da FSM para o Motor Esquerdo (L)

Estado	Sentido do Motor E	Duty_ L	Duty_L%
PERDIDO	’10’ (Frente)	’0111’	50%
FRENTE	’10’ (Frente)	’1111’	100%
DIREITA BRUSCA	’10’ (Frente)	’1100’	81,25%
DIREITA FINA	’10’ (Frente)	’0111’	50%
ESQUERDA BRUSCA	’01’ (Trás)	’1100’	81,25%
ESQUERDA FINA	’01’ (Trás)	’0111’	50%

3) Módulo gerador de PWM: A lógica do modulador por largura de pulso utilizada gera um sinal de 60Hz que tem seu período de nível lógico alto regulado pela entrada vetorial de 4 bits *duty*, a ser fornecida pela máquina de estados do sistema. Em uma mesma descrição, é feita a divisão do *clock* nativo da placa *Basys 3* (100MHz) e a modulação de acordo com o valor de *duty* recebido. Vale destacar novamente que a lógica do sinal gerado a partir do *duty* obedece o padrão estabelecido pela Figura 4.

A arquitetura adotada se utiliza de contadores para formar a modulação na frequência pedida na saída, tomando por base que, para um sinal de 60Hz com *duty cycle* de 50% (níveis lógicos com durações iguais), são necessários 850.000 ciclos de 100 MHz para cada um dos níveis.

Na tabela abaixo registram-se as *flags* limite do contador para cada uma das configurações de *duty* possíveis.

Tabela VI
Valores limite dos contadores para a definição do nível lógico alto do sinal de acordo com o duty selecionado

Duty (Duty%)	Limite do contador para nível lógico alto - *’flags’*
0000 (6.25%)	106.250
0001 (12.50%)	212.500
0010 (18.75%)	318.750
0011 (25.00%)	425.000
0100 (31.25%)	531.250
0101 (37.50%)	637.500
0110 (43.75%)	743.750
0111 (50.00%)	850.000
1000 (56.25%)	956.250
1001 (62.50%)	1.062.500
1010 (68.75%)	1.168.750
1011 (75.00%)	1.275.000
1100 (81.25%)	1.381.250
1101 (87.50%)	1.487.500
1110 (93.75%)	1.593.750
1111 (100%)	1.700.000

O *duty* ’1111’ estabelece nível lógico ’1’ diretamente na saída, sem percorrer toda a contagem para atualizar-se. Cada mudança de *duty* altera em 6,25% a duração do nível lógico alto do sinal de saída, o que corresponde a um período de 106.250 ciclos de *clock* nativos.

D. Integração da lógica em VHDL

Cada um dos módulos da lógica em VHDL (leitura de sensores, FSM e gerador de PWM) foi empacotado como um *IP-core* e adicionado ao repositório do *Block Design* do *Vivado*. Neste ambiente, estabeleceu-se as ligações entre os blocos e associou-se os sinais de *clock* e *reset* às ligações da placa *Basys3*, e as saídas do sistemas foram evidenciadas para que no arquivo de *constraints Basys3_Master.xdc* fossem designadas as portas necessárias para a interface da placa e dos circuitos complementares, seja o de sensoriamento, seja o de controle de planta.

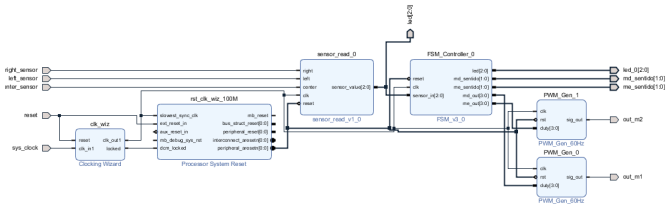

Figura 8. *Block Design* desenvolvido para o robô seguidor de linha.

E. Circuito de potência dos motores

Para o controle dos motores DC em cada uma das rodas de tração do robô, utilizou-se o circuito integrado L293D de 4 meias pontes Hs configurado para controlar o sentido de rotação das plantas de 7V, bem como suas respectivas velocidades, contando para isso com os sinais de PWM gerados na FPGA.

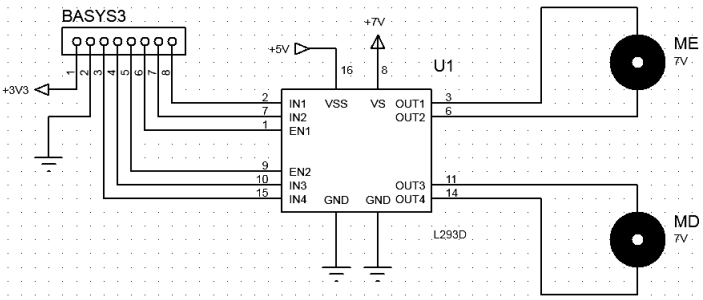

Figura 9. Esquemático do circuito de potência do autônomo e sua interface com a placa FPGA.

Nota-se na imagem acima um barramento chamado *’BASYS3’*, identificado desta forma para sinalizar as conexões da FPGA que controlam o funcionamento do *driver*, que por sua vez controla os motores. Sobre a pinagem do barramento:

- 1 - Tensão de 3.3 v, proveniente da *Basys3* (utilizada futuramente para o circuito de sensoriamento);
- 2 - Ground (GND), para aterrar a placa no circuito;
- 3 e 4 - Sinal digital para controla o sentido e rotação do motor direito (R);
- 5 - PWM para controle da velocidade de rotação do motor direito;
- 6 - PWM para controle da velocidade de rotação do motor esquerdo;
- 7 e 8 - Sinal digital para controlar o sentido de rotação do motor esquerdo (L).

O circuito integrado L293D isola internamente os sinais de comando dos terminais ligados aos motores, de forma que a placa FPGA *Basys3* mantenha-se protegida por isso de eventuais sobrecargas de corrente em seus pinos de GPIO causados pela operação dos motores. Toda a potência requerida

pela planta será fornecida diretamente pela alimentação, e não pela placa.

F. Alimentação do sistema

O sistema deverá ser alimentado por uma fonte de tensão DC principal de pelo menos 7V (até 9V), que pode alimentar diretamente os motores. Para a porção lógica do autônomo, há um abaixamento de tensão efetuado por dois reguladores 7805 distintos, cada um deles dedicado a alimentar uma das porções do circuito: o *driver* de motores (L293D) e a placa FPGA *Basys3*, via pinagem de alimentação externa.

O diagrama dessa montagem pode ser observado na figura a seguir.

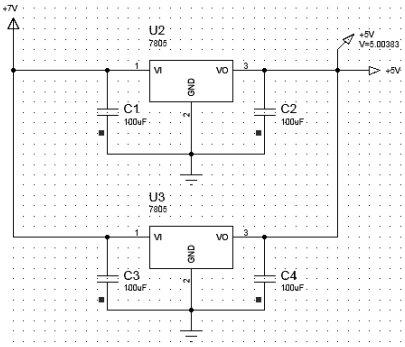

Figura 10. Esquemático do circuito de alimentação e suas malhas distintas de fornecimento de tensão.

A separação em duas malhas de 5V feita com as duas unidades do 7805 tem a intenção de evitar um gargalo de corrente, levando em consideração que o regulador de tensão suporta apenas 1A, além de uma maior separação dos meios de alimentação da placa FPGA em relação às demais circuitarias que formam o projeto.

III. Resultados e Discussões

A. Reports - Consumo da Placa

Ao realizar as sínteses de cada um dos módulos propostos e gerar os IPCores de cada um dos mesmos, pôde-se obter, a nível de comparação, o consumo de hardware e potência individual e coletivo (após inclusão dos módulos do block design), além das especificações de timing deste projeto. O block-design proposto para este trabalho pode ser revisto de acordo com o diagrama da figura 8.

Tabela VII
Especificações de tempo

Módulo	WNS	WHS
FSM	7.858 ns	0.242 ns
Leitor dos Sensores	NA	NA
Gerador PWM	4.331 ns	0.291 ns
Projeto Integrado	**3.820 ns**	**0.085 ns**

Observa-se que há um resultado melhor com a integração dos módulos, tendo em vista que utilizando a ferramenta do block-design, ocorrem algumas otimizações quanto ao clock gerado, podendo diminuir o WNS (*Worst Negative Slack*) e o WHS (*Worst Hold Slack*).

Com o consumo de hardware, observa-se a tabela III-A. Percebe-se que o consumo de hardware na aplicação total é maior do que a soma dos demais, isto deve-se ao fato de que na integração há uma utilização de módulos próprios ao reset e ao clock da placa, podendo gerar esta diferença na hora dos resultados obtidos.

Tabela VIII
Consumo de hardware obtido na aplicação

Módulo	LUT	FF	IO
FSM	15	11	21
Leitor dos Sensores	1	3	8
Gerador PWM	81	22	7
Projeto Integrado	**147**	**93**	**17**

Para a potência consumida pela placa, não foi diferente, já que os resultados obtidos são para cada módulo de forma separada, há sempre a parte estática, que está considerada na tabela III-A, que é inerente aos cálculos do Vivado.

Módulo	P. Dinâmica	P. Estática
FSM	0.006 W	0.072 W
Leitor dos Sensores	0.002 W	0.072 W
Gerador PWM	0.002 W	0.072 W
Projeto Integrado	**0.110 W**	**0.072 W**

B. Simulações e Testes de Bancada

Foi realizada a simulação de cada módulo separado, de forma a obter uma simulação comportamental mais fácil de debugar, permitindo uma rápida correção de erros e possíveis falhas.

O módulo de leitura dos sensores foi somente feito para filtrar algum possível ruído na entrada da FSM, portanto, sua simulação foi feita de maneira bem breve e as saídas corresponderam às entradas esperadas.

O primeiro módulo simulado e testado foi o gerador de PWM, experimentalmente, chegou-se a conclusão de que com uma frequência de aproximadamente 60Hz os motores apresentavam um comportamento esperado, com velocidade constante e sem ressonância. Portanto, na simulação foram escolhidas 3 possíveis *duty-cycles*, para podermos observar o comportamento da onda. As opções utilizadas foram "0111"de 50%, "0010"de 18.75% e "1110"de 93.75%.

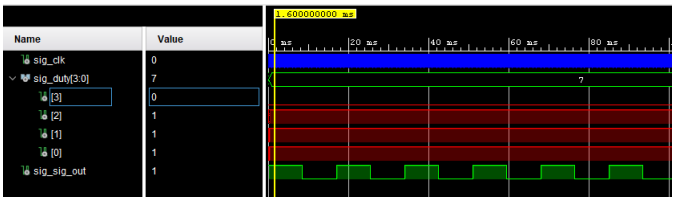

Figura 11. Forma de Onda simulada com duty de 50%.

Ao testar este módulo na FPGA com um osciloscópio, teve um comportamento conforme o esperado, a única diferença

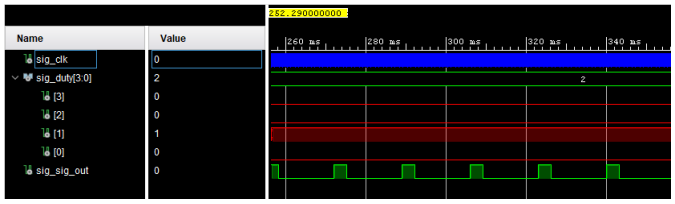

Figura 12. Forma de Onda simulada com duty de 18.75%.

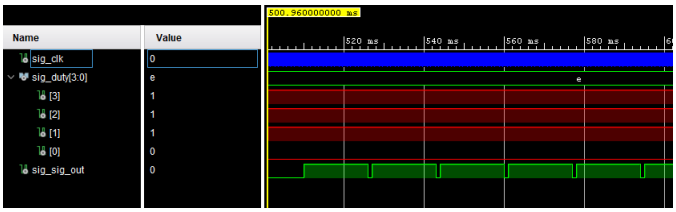

Figura 13. Forma de Onda simulada com duty de 93.75%.

observada é que a frequência desejada não era de 60Hz, mas ficava bem próxima disto, atingindo 58.82Hz. Um erro de aproximadamente 1,96%. Testando com as mesmas entradas de *duty* da simulações, obteve-se as seguintes saidas nas figuras 14, 15, 16.

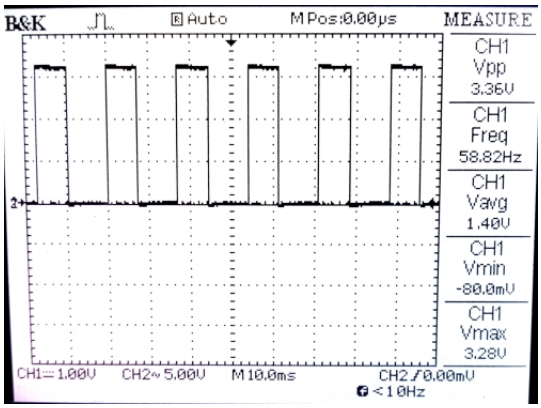

Figura 14. Forma de Onda gerada com duty de 50%.

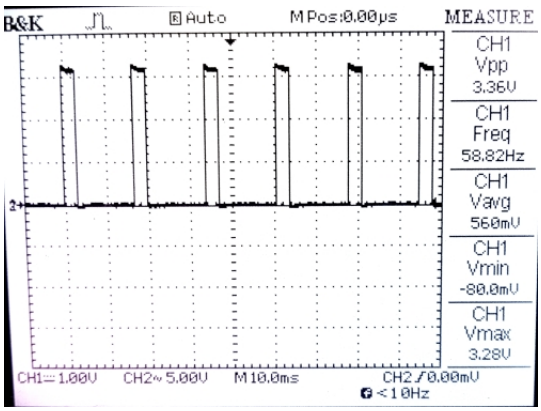

Figura 15. Forma de Onda gerada com duty de 18.75%.

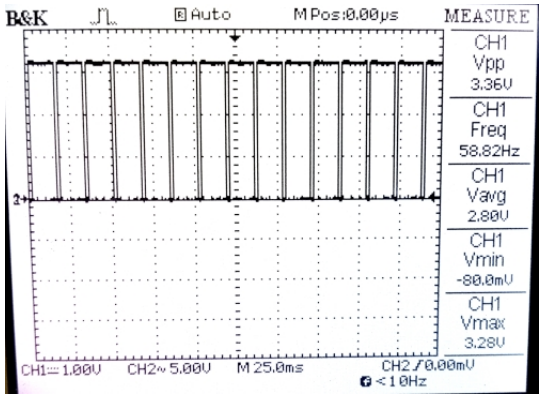

Figura 16. Forma de Onda gerada com duty de 93.75%.

Após estes testes, foram feitos os testes e simulações para cada a máquina de estados. Propõe-se então realizar as seguintes condições de acordo com o tempo: leitura normal da reta, breve desvio para direita, leitura normal da reta, breve desvio para a direita novamente seguido de um desvio mais brusco, retornando à reta, passando por um cruzamento e se perdendo. Na simulação, a visibilidade não foi das melhores, tendo em vista que a máquina de estados sempre retorna ao estado de leitura para averiguação dos sensores. Contudo, pôde-se observar o transitório, afim de comprovar a validação da máquina de estados. A figura 17 comprova a mudança de estados da reta para a curva à direita e retornado da curva para a reta, respectivamente.

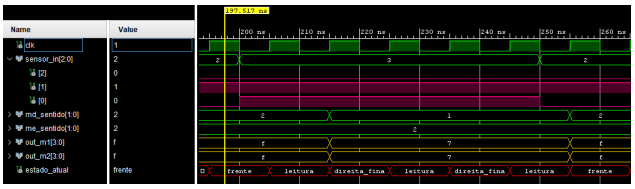

Figura 17. Comportamento do robô quanto à uma curva à direita, elucidando a máquina de estados.

C. Testes de Integração

Além das validações realizadas na lógica VHDL, outros testes foram realizados com a implementação também das circuitarias de sensoriamento e da planta do robô. Os motores, associados cada um a uma caixa de redução e roda tratora, foram acoplados lado a lado a um chassi plástico. Inicialmente, a circuitaria adicional foi implementada em *protoboard*, e pode ser finalizada em placa perfurada - garantindo maior estabilidade e resistência dos componentes e suas interconexões.

Nesses testes, as circuitarias adicionais mostraram-se aptas a fornecer e receber dados de/para a a lógica implementada em VHDL na placa FPGA, registrando sucesso na leitura dos sensores e no controle independente do estado dos motores.

Curiosamente, nesses testes, observou-se uma diferença na reação de um motor para o outro quando submetidos a uma mesma configuração de *'duty'*. Uma das plantas, independentemente dos ajustes e trocas de posicionamento e de conexões,

exibiu uma menor velocidade de rotação quando comparada ao outro motor. Concluiu-se que essa defasagem não dependia de qualquer particularidade do sistema implementado, mas de diferenças de fabricação entre os motores, idênticos entre si excluindo-se esse comportamento.

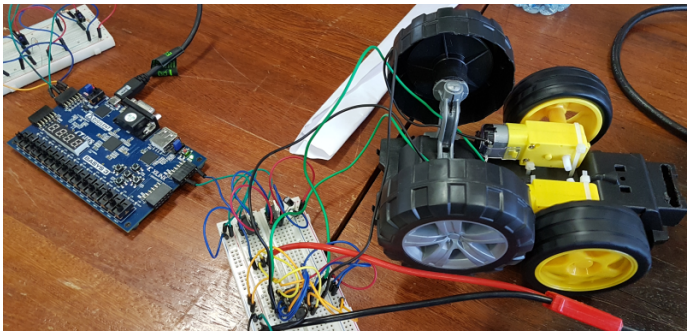

Figura 18. Registro de um dos testes de integração dos circuitos do robô autônomo.

De modo a contornar essa defasagem, é possível corrigir a lógica para criar um *"offset"* no *duty* do motor mais lento de modo que se possa gerar velocidades de rotação mais próximas de estar em fase com o motor oposto.

Com isso, destaca-se como uma vantagem de projeto ter-se feito a implementação de geradores de *PWM* dedicados para cada um dos motores.

IV. Conclusões

A difusão da robótica e automação nos meios de produção mostra uma fatia de mercado a ser explorada através do desenvolvimento de máquinas autônomas que executem procedimentos com maior precisão e robustez do que a habilidade humana. Como por exemplo, o seguidor de linha desenvolvido nesse projeto é um excelente ponto de partida para inumeras aplicações de automação residencial e Universitárias, como é o caso da equipe de robótica *Titans* que desenvolve esse tipo de robôs para competições.

A equipe de competição *Titans*, voltada para robótica, poderia usufruir de todo o material disponibilizado com o intuito de promover estudos na área de eletrônica embarcada voltada para róbotica a partir de FPGAs. Desta forma contribuindo ainda mais com a comunidade que desenvolve aplicações para este tipo de tecnologia e promovendo um desenvolvimento pessoal para os integrantes da equipe, já que o contato com VHDL e tecnologias FPGA estariam sendo recorrentes.

Uma possível continuação do projeto é promover aplicações de sistemas de controle no robô, como por exemplo, a implementação de um sistema em malha fechada usufruindo de um controlador PID. O controlador serviria para ajustar uma possível velocidade e tentar diminuir o erro no cálculo executado pela FPGA durante a tradução dos estados.

O desenvolvimento do projeto abordou os assuntos desenvolvidos durante a disciplina, abordando sempre os temas de VHDL mais recorrentes, como as ferramentas que o *Vivado* disponibiliza para os *reports*, a possibilidade de carregar um sistema *on-chip* diversas vezes na mesma plataforma, a elaboração de máquinas de estado e, principalmente, usufruir da ferramenta de propriedade intelectual (*IP-Core*) para produção modular da aplicação.

Todos os códigos feitos durante este projeto, estão disponíveis no repositório do GitHub, com endereço: https://github.com/arthursgonzaga/LineFollowerFPGA

Referências

[1] J. Manners-Bell, "The impact of robotics and automation on logistics - an insight into how the logistics sector could be impacted by advances in manufacturing technology," *Transport Intelligence*, pp. 1–7, 2014.

[2] M. G. Exame/Abril, "Kiva, os robôs espertos que a amazon comprou. https://exame.abril.com.br/tecnologia/kiva-os-robos-espertos-que-a-amazon-comprou/," 2012.

[3] N. C. Braga, "Controle pwm de motor dc. http://www.newtoncbraga.com.br/index.php/robotica/5534-mec139," 2013.

[4] B. P. Lathi, *Modern Digital and Analog Communication Systems 3e Osece*. Oxford university press, 1998.