Universidade de Brasília
Faculdade de Tecnologia

Plataforma Reconfigurável para Rastreamento e Controle de Veículos em Frotas

Douglas Lustosa da Silva

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2022

Universidade de Brasília
Faculdade de Tecnologia

Plataforma Reconfigurável para Rastreamento e Controle de Veículos em Frotas

Douglas Lustosa da Silva

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Jones Yudi Mori Alves da Silva

Brasília
2022

Lustosa da Silva, Douglas.
L972p Plataforma Reconfigurável para Rastreamento e Controle de Veículos em Frotas / Douglas Lustosa da Silva; orientador Jones Yudi Mori Alves da Silva . –– Brasília, 2022.
84 p.

Trabalho de Graduação em Engenharia de Controle e Automação –– Universidade de Brasília, 2022.

1. Veículos Autônomos. 2. Platooning. 3. Sistemas Embarcados. 4. Visão Computacional. I. , Jones Yudi Mori Alves da Silva, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Plataforma Reconfigurável para Rastreamento e Controle de Veículos em Frotas

Douglas Lustosa da Silva

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Trabalho aprovado. Brasília, 10 de Maio de 2022:

Prof. Dr. Jones Yudi Mori Alves da Silva, UnB/FT/ENM
Orientador

Prof. Dr. Gerardo Antonio Idrobo Pizo, UnB/FGA
Examinador interno

Prof. Enio Prates Vasconcelos Filho, IFG
Examinador externo

Brasília
2022

Este trabalho está dedicado a todos que acreditaram em minha capacidade.
Principalmente aos meus pais e amigos, que sempre estiveram comigo.

Agradecimentos

Agradeço primeiramente a minha família, já que ela forma a base da minha rede de apoio, me inspirando a buscar meu sonhos mesmo com tantas adversidades e incertezas presentes em nossas vidas. Espero dar bastante orgulho pela criação que me deram. Aos meu amigos da mecatrônica, incluindo os que não fazem mais parte, com quem sempre pude contar e me expressar de maneira que saberia que eles iriam entender, obrigado pelos momentos divertidos durante este período de graduação, mostrando um mundo que antes eu não tinha contato.

Também agradeço aos meu amigos de mais longa data, que estiveram comigo em vários momentos da minha vida, amigos que considero parte da minha família, agradeço conselho e a paciência que tiveram comigo.

Aos professores que formaram a minha base tenho uma profunda gratidão, cheguei neste ponto graças ao incentivo e instrução que cada um passou a mim desde dos meus primeiros anos de ensino até hoje. Agradeço aos professores Vadir de Casto, Myrian Batalini, Marcos, Renato e Jacob por me incentivar a querer mais e sonhar com um curso superior. Também agradeço ao professores amigos que tive contato durante a graduação, principalmente meu orientador que sempre me incentivou e ajudou muito.

Agradeço também a Juliana Dias por ser paciente e compreensiva, sempre me apoiando a continuar este projeto. Obrigado por estar comigo.

“O feito é melhor que o perfeito!”
(Autor Desconhecido)

Resumo

Na última década, emergiram diversos produtos e serviços de veículos autônomos, tanto para o usuário final, quanto para o setor corporativo: veículos de passeio, ônibus, vans, caminhões e veículos agrícolas têm sido lançados com frequência. Vários países já avançam em discussões sobre legislação específica para esse setor. Uma das principais preocupações dos legisladores, engenheiros e pesquisadores é a garantia de segurança dos veículos autônomos. Uma das alternativas estudadas é a formação de pelotões (*platooning*) com a colaboração entre veículos propiciando mais previsibilidade no trânsito, e consequente aumento de segurança dos sistemas. Um pelotão geralmente possui um líder (o veículo mais a frente do grupo) e um ou mais seguidores que atuam de acordo com informações sobre o deslocamento do líder. Este trabalho tem como foco implementar uma plataforma para estudos sobre *platooning* composta por dois veículos (um líder e um seguidor). Os sistemas embarcados envolvidos possuem requisitos de tempo real, em que tarefas devem ser executadas dentro de uma janela temporal, sob pena de falha. Para prover flexibilidade no desenvolvimento e aceleração de processamento, este projeto utiliza arquiteturas híbridas: CPU e Lógica Programável integradas. O foco deste trabalho é na integração de Câmera, CPU, FPGA e os sistemas eletromecânicos, resultando em uma plataforma completa e funcional. Para validação da plataforma, são implementados algoritmos de controle para os motores e a criação de módulos *hardware/software* de visão computacional para rastreamento do líder pelo seguidor.

Palavras-chave: Veículos Autônomos. *Platooning*. Sistemas Embarcados. Visão Computacional.

Abstract

Several autonomous vehicle products and services have emerged in the last decade, both for the end-user and the corporate sector: passenger cars, buses, vans, trucks and agricultural vehicles. Several countries are already advancing discussions on specific legislation for this sector. One of the main concerns of legislators, engineers and researchers is to ensure the safety of autonomous vehicles. One of the alternatives studied is the formation of platoons (*platooning*) with the collaboration among vehicles, providing more predictability in traffic and consequently increasing the safety of the systems. A platoon usually has a leader (the vehicle ahead of the group) and one or more followers that act according to information about the displacement of the leader. This work focuses on implementing a platform for studies about *platooning* composed of two vehicles (a leader and a follower). The embedded systems involved have real-time requirements, in which tasks must be executed within a time window, under penalty of failure. This project uses hybrid architectures: integrated CPU and Programmable Logic to provide flexibility in development and processing acceleration. This work focuses on the integration of Camera, CPU, FPGA, and the electromechanical systems, resulting in a complete and functional platform. To validate the platform, control algorithms are implemented for the motors and the creation of modules *hardware/software* of computer vision for tracking the leader by the follower.

Keywords: *Computer Vision*. FPGA. *Control*. *Instrumentation*. *Embedded Hardware*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AEC	*Automatic Exposure Control Mode*	43
ARM	*Advanced RISC Machine*	36
ASICs	*Application Specific Integrated Circuits*	25
AWB	*Automatic White Balance*	43
AXI	*Advanced Extensible Interface*	34
BRAM	Blocos de Memória de Acesso Randômico	36
CCD	*Charge-coupled Device*	24
CIR	Centro Instantâneo de Rotação	38
CLB	*Configurable Logic Block*	35
CM	centro geométrico	41
CMOS	*Complementary Metal-oxide Semiconductor*	24
DNT	*Disruption/Delay-Tolerant Networks*	18
FPGA	*Field Programmable Gate Array*	26
HSV	*hue, saturation e value*	29
ITS	*Intelligent Transportation Systems - Sistemas Inteligentes de Transporte*	15
Lidar	*Light Detection and Ranging*	23
LUT	*Look-Up Table*	35
OMS	Organização Mundial de Saúde	15
PL	*Programable Logic*	34
PS	*Processing System*	34
PWM	*Pulse Width Modulation*	48
SCCB	*Serial Camera Control Bus*	43
SoC	*System on Chip*	26
V2I	*Vehicle-to-Infrastructure*	18
V2V	*Vehicle-to-Vehicle*	18
VDMA	*Video Direct Memory Access*	53
ZYBO	*Zynq Board*	36

Sumário

1 Introdução

1.1 Contextualização

Com o avanço das dinâmicas relacionadas ao trânsito, atualmente enfrentamos muitos desafios. Desafios estes relacionados com o aumento do tráfego nas rodovias, que consequentemente aumentam proporcionalmente a emissão de gases poluentes e o crescimento do números de acidentados nas estradas.(SILVA, 2019/2020)

Segundo dados da Organização Mundial da Saúde (OMS) morrem por ano aproximadamente 1,35 milhões de pessoas devido a acidentes de trânsito. Somente no Brasil, em média, 40 mil pessoas perdem a vida anualmente no trânsito(OMS, 2019). Estima-se que 90% dos acidentes ocorrem devido à negligência humana. Para resolver este tipo de problema sistemas de veículos autônomos podem garantir a segurança de todos os envolvidos no trânsito.

Para superar as adversidades encontradas, a indústria automotiva vem se aprimorando em vários aspectos, desenvolvendo novas tecnologias para o consumo eficiente de combustível, redução na emissão de partículas de carbono, assistência na direção e segurança. Este último ponto tem sido alvo de muito interesse, pois garantir a segurança de motoristas e pedestres vem sendo um grande obstáculo. (ALMEIDA, 2019)

Com os esforços empregados, surgem sistemas de transporte mais complexos e inteligentes que vêm buscando dar mais conforto e segurança aos usuários, esses sistemas estão sendo chamado de ITS - *Intelligent Transportation Systems - Sistemas Inteligentes de Transporte*. A ITS trata-se de uma tecnologia de informação voltada para o transporte rodoviário que inclui a infraestrutura existente, os veículos e os usuários. Esse sistema captura dados dos agentes envolvidos para que assim possa melhorar o tráfego. A arquitetura das ITS é composta basicamente por três classes: uma classe central de controle e gerenciamento do sistema, uma classe que abrange toda a infraestrutura do ambiente determinando meios de comunicação com o usuário e uma classe para usuários e veículos, os modelos de ITS devem prever a capacidade dos veículos de se comunicarem entre si e com toda a infraestrutura inteligente presente na rodovia. Com a ITS busca-se otimizar o fluxo rodoviário e aumentar o grau de segurança assim, dentro deste contexto, surgem os veículos autônomos.(SISTEMAS..., s.d.)

Os automóveis autônomos surgiram como uma idealização do futuro do tráfego nas cidades, que vem ganhando cada vez mais destaque com muitos destes veículos em testes em várias partes do mundo. Muitos tópicos diferentes de pesquisa envolvendo sistemas de veículos autônomos vem sendo explorado nos últimos anos, seja sobre condições de segu-

rança, comunicação entre veículos, algoritmos de controle, entre muitos outros.(ALMEIDA, 2019)

Um destes tópicos de pesquisa é o pelotão de veículos, onde os veículos seguem uns aos outros mantendo uma distância limite mínima formando um pelotão, resultando em uma maior eficiência e segurança no transporte (LIANG; MÅRTENSSON; JOHANSSON, 2016). Isto é possível graças à comunicação entre os veículos ou por um sistema de sensoriamento presente nos veículos, para o caso do pelotão não colaborativo, assim o pelotão veicular torna-se possível, traduzindo-se em uma confiabilidade maior e uma aumento considerável na segurança rodoviária.(SILVA, 2019/2020)

O interesse pelo *platooning* vem crescendo de maneira exponencial no meio acadêmico e nas indústrias, fazendo com que surjam cada vez mais trabalhos relacionados ao tema. Muito destes estudos tem como foco o impacto que as frotas podem ter na sociedade, tendo como foco a eficiência e segurança no transporte. (GUEDES, 2019)

Um dos principais tópicos estudados para ser implementados em um sistema com *platooning* é como regular o aumento de veículos nas ruas usando este sistema de pelotão. Para isso, alguns estudos sugerem *platoon based driving* como uma solução viável, pois melhora o fluxo de tráfego nas cidades e rodovias (GUEDES, 2019).

O *platooning* tem muitas vantagens para o tráfego e os motoristas. Os carros em um pelotão tem uma distância de segurança constante que é menor do que em situações normais de trânsito, isso resulta em menos congestionamento e uma maior capacidade das estradas. Como os carros são próximos uns dos outros, a resistência do ar diminui, resultando em menos consumo de energia e menos emissões de gases poluentes para o meio ambiente. (GUEDES, 2019)

As aplicações mais comuns de pelotão de veículos concentram-se na otimização do tráfego e redução do consumo de combustível. Existem muitos estudos nestas áreas como o apresentado em (GONG; DU, 2018), onde há uma análise detalhada do fluxo de trânsito considerando um pelotão autônomo e um fluxo misturando veículos autônomos e não-autônomos. Um exemplo que pode ser apresentado em relação ao consumo de energia pode ser visto no trabalho (LARSSON; SENNTON; LARSON, 2015), onde avalia-se a redução da utilização de energia provocada pelos pelotões.

1.2 Sistemas de Transporte Inteligente

O controle de veículos autônomos pode ser gerenciado por vários sensores que ficam embarcados nos veículos, alguns deles serão discutidos na seção 1.2.2. Entretanto com o auxílio de uma rede de comunicação adequada é possível deixar o sistema mais robusto, trazendo mais segurança e confiabilidade para o pelotão autônomo. Para buscar

uma maior troca de dados entre os agentes envolvidos no trânsito utilizam-se os ITS, também definidos como "um sistema integrado que implementa uma ampla gama de tecnologias de comunicação, controle, detecção de veículos e eletrônica para ajudar no monitoramento e gerenciamento do fluxo de tráfego, reduzindo o congestionamento, fornecendo rotas ideais para os viajantes, aumentando a produtividade do sistema e salvando vidas, tempo e dinheiro"(SINGH BHUPENDRA, 2015).

Com o avanço da implementação das tecnologias ITS, vem surgindo a necessidade do sistema ser cada vez mais integrado e cooperativo, assim pode-se processar em tempo real uma grande quantidade de informação advindas de várias fontes para fornecer dados mais detalhados para prevenir situações de risco. Um bom exemplo pode ser verificado na figura 1, onde a infraestrutura presente na rodovia identificou uma situação crítica na estrada e através da rede de comunicação V2I (*Vehicle-to-Infrastructure*) transmitiu a informação para os veículos capazes de se comunicar na rede. Depois, através de uma comunicação V2V (*Vehicle-to-Vehicle*), a informação é espalhada para que os próximos veículos que aproximem-se do ponto de acidente possam traçar um trajeto mais apropriado.

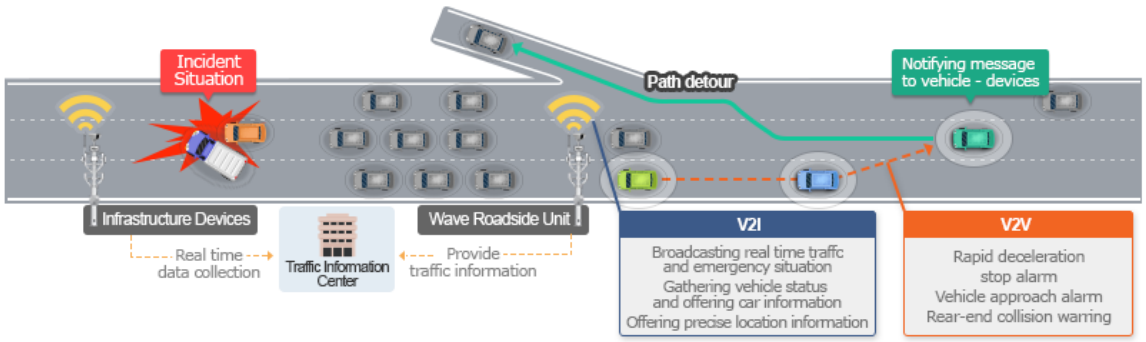


Figura 1 – Exemplo de ITS.(KOREA, 2022)

A aplicação de tecnologia ITS tem grande potencial para melhorar o fluxo de veículos, como demonstrado na figura 1, detectando acidentes e procurando maneiras de responder adequadamente a situação, também minimiza impactos ambientais, reduzindo os níveis de poluição local.

1.2.1 Rede de Comunicação Veicular

Como foi mencionado na seção 1.2 as ITS prevem um grande fluxo de comunicação entre os veículos com a infraestrutura existente e com os próprios veículos, este último bastante importante para que se possa haver uma cooperação veicular e assim tornar-se viável uma das formas de se ter pelotões. Para uma rede de comunicação veicular funcional a tecnologia *wireless* vem sendo de grande importância, pois assim pode-se haver troca de dados entre os veículos de maneira que se possar ter mais confiabilidade e segurança. (GUEDES, 2019)

Uma rede veicular pode surgir a partir da comunicação direta entre os veículos, chamada de *Vehicle-to-Vehicle*(V2V) ou também a partir de uma comunicação do veículo com a infraestrutura de rede, chamada de *Vehicle-to-Infrastructure* (V2I). Estes tipos de redes são baseados em modelos *Ad hoc*, que é uma rede Disruption/Delay-Tolerant Networks (DNT) específica onde os nós da rede podem trocar informações entre si sem a necessidade de um ponto de acesso comum. Logo, para funcionamento deste tipo de rede cada veículo pertencente ao pelotão deverá ter um sistema que receberá as informações dos seus sensores e as enviarão ou para uma unidade de aquisição de dados na estrada ou para os demais membros do pelotão. O propósito desta unidade de aquisição de dados na estrada é coletar dados de carros e/ou as condições das estradas e enviá-los para o pelotão, para que possam adaptar seu controle e escolher um caminho mais seguro. (GUEDES, 2019) (SOUSA NUNES, 2018)

A utilização de redes de comunicação como a V2V é bastante benéfica, pois, desta forma, seria possível enviar e receber informações relevantes em tempo real, tais como posição, velocidade, alertas de acidentes, mau tempo ou catástrofes, blitzes e informações sobre o trânsito em geral. Na figura 2 é possível observar um exemplos da comunicação veicular, observando a imagem verifica-se a possibilidade de comunicação entre os veículos e pontos remotos na estrada que transmitirão a informação da ocorrência de um evento no trânsito. (GUEDES, 2019)

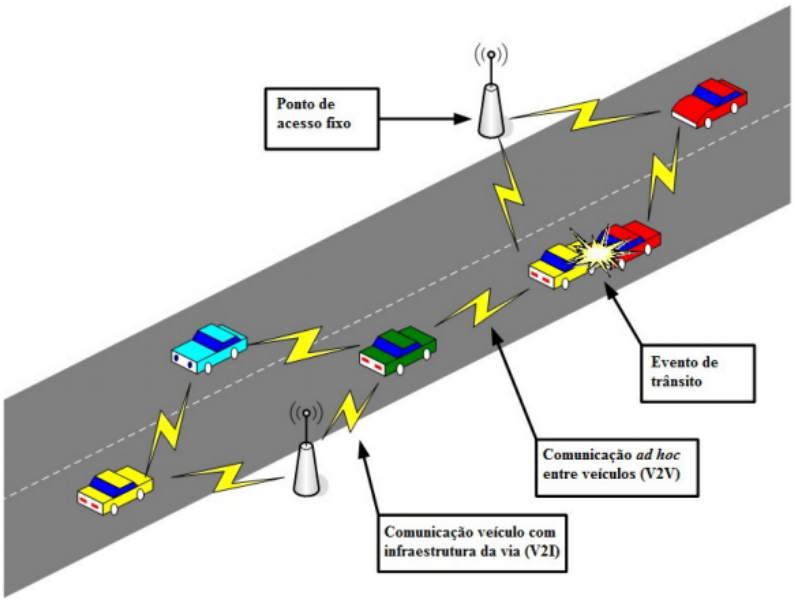


Figura 2 – Esquematização da comunicação V2V e V2I. (OLIVEIRA, 2013)

1.2.2 Pelotão Veicular

Considerando o contexto verificado com a implementação das tecnologias adotadas para a ITS percebe-se uma crescente cooperação entre os agentes presentes no trânsito, mais

especificamente uma cooperação entre veículos. Assim, verifica-se uma evolução natural desta cooperação para a formação de pelotões de veículos autônomos, em inglês *vehicle platooning*, pois essa evolução traria grandes vantagens em relação a segurança no transporte e o custo-benefício. O pelotão de veículos autônomos pode aumentar a capacidade de veículos na estrada e melhorar a eficiência energética dos veículos envolvidos dado o espaçamento entre os veículos menor fazendo com que o coeficiente de arrasto aerodinâmico seja reduzido, como pode ser verificado na figura 3.

O pelotão de veículos autônomos é composto por um líder que irá determinar o trajeto a ser seguido pelo pelotão, este veículo será responsável por guiar toda a frota e transmitir dados sobre qualquer situação que ocorra à frente. Os outros membros do pelotão são denominados seguidores, eles possuem um conjunto de sensores que os auxiliam a seguir o trajeto determinado pelo líder do pelotão, além de poder possuir uma comunicação efetiva entre os mesmos e o líder.

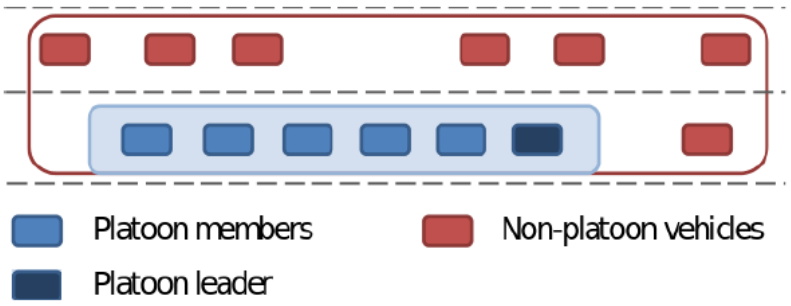


Figura 3 – Visão geral de um pelotão de veículos cooperativos.(GUEDES, 2019)

Para que o pelotão de veículos seja inserido de maneira segura nas estradas deve-se pensar em várias situações que esse conjunto de veículos possa enfrentar. A figura 4 mostra algumas situações que o pelotão pode enfrentar. A formação em linha deve ser mantida durante o trajeto para que se possa obter os benefícios energéticos, mas também o pelotão deve ter a capacidade de se dividir para outras rotas determinando novos líderes, além, também, de verificar o momento correto para entrar em uma nova via.

Atualmente existem vários projetos envolvendo o pelotão de veículos em desenvolvimento, um exemplo de projeto desenvolvido é o *European Truck Platooning Challenge* que ocorreu em 2016, esse desafio contou com a participação de 6 montadoras: DAF, Daimler, Iveco, MAN, Scania e Volvo, que partiram da Alemanha, Suécia e Bélgica em direção ao porto de Roterdã, na Holanda. Cada montadora ficou responsável por um pelotão composto por três caminhões,onde em cada pelotão foi aplicado meios diferentes para realizar o desafio, como por exemplo, a DAF fez com que os os caminhões do pelotão se comunicassem através do protocolo 802.11p transmitindo dados relacionados a condição das estradas e o que estava a frente do pelotão captados por radares e câmeras embarcadas nos veículos. Desde o desafio

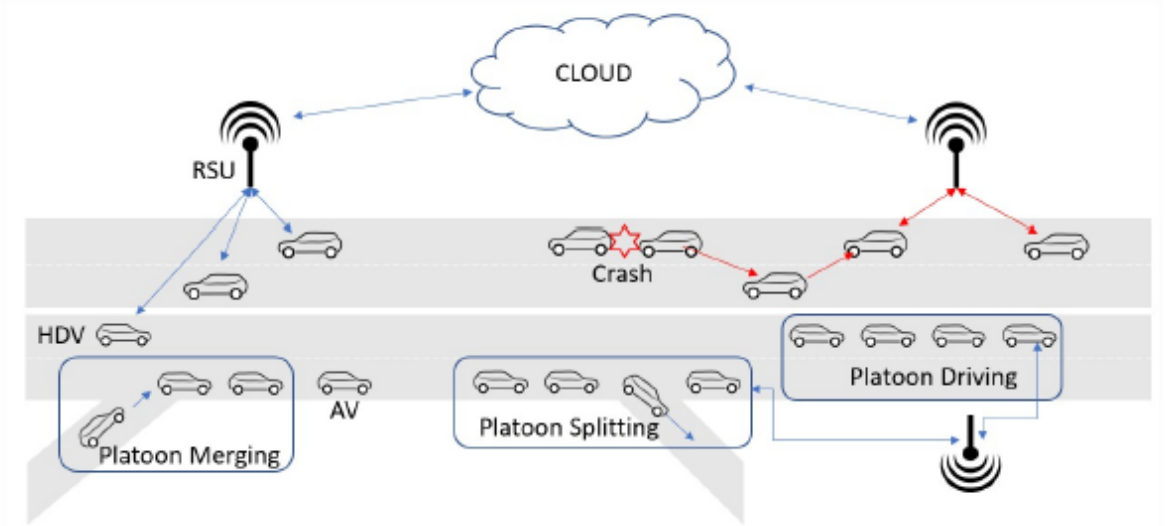


Figura 4 – Configurações necessárias para a implementação de uma cooperação de veículos autônomos.(GUEDES, 2019)

realizado em 2016 o *truck platooning* vem apresentando grandes avanços apesar de existirem algumas barreiras a serem superadas , como por exemplo a condição das estradas, pois uma infraestrutura inadequada pode fazer com que os sistemas percam sua referência ocorrendo uma dessincronização entre os veículos.(HOLANDÊS, 2022)

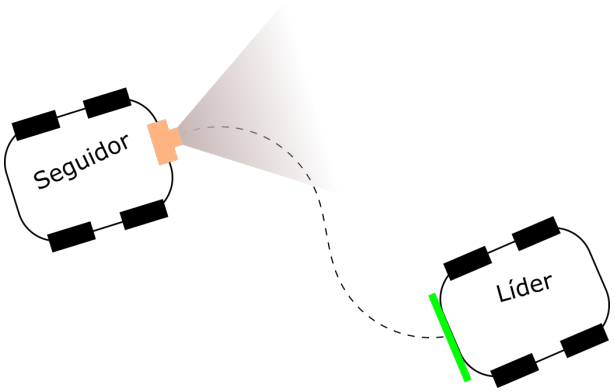


Figura 5 – Representação do sistema a ser desenvolvido.

Verifica-se que o sistema de cooperação veicular apresenta requisitos de tempo real com *deadlines* críticos, pois caso o sistema apresente uma falha que faça com que o sistema perca o tempo de execução de uma tarefa as consequências podem ser desastrosas. A perda de referência é um dos principais problemas enfrentados em um pelotão de veículos autônomos, pois, caso algum dado não esteja disponível, o veículo seguidor não conseguirá determinar a trajetória do líder ou não irá perceber que houve uma parada fazendo com que a formação se quebre ou ocorra um acidente. Para resolver este tipo de problemas estudos como os mostrados em (GUEDES, 2019) apontam que uma comunicação veicular entre os membros

do pelotão, para que haja compartilhamento das informações dos sensores presentes nos veículos, seja uma maneira eficaz de garantir segurança e redundância ao sistema.

Além do pelotão cooperativo, é possível que está formação não possua comunicação, que é caso mais simples a ser tratado. Neste tipo de formação os seguidores copiam o movimento do líder utilizando informações captadas pelos sensores embarcados no veículo. O projeto a ser desenvolvido será considerando este caso.

1.3 Escopo do Projeto

O foco de desenvolvimento deste projeto é o problema particular de um veículo seguidor que não terá comunicação com o líder. Assim, o seguidor deverá ser capaz de localizar o seu veículo líder e, ao segui-lo, manter uma distância de segurança pré determinada, seja parado ou em movimento. Na figura 5 mostra uma exemplificação do sistema a ser desenvolvido, onde, através dos sensores presentes no seguidor, é capaz de verificar a localização do líder e verificar sua movimentação para que assim possa reproduzir a mesma trajetória realizada pelo líder. Assim o seguidor deverá avançar quando o líder avançar e mudar a direção quando ele mudar.

O modelo líder-seguidor refere-se à estrutura que compõe os dois veículos utilizados. O líder é o primeiro veículo, este receberá comandos através de uma interface *bluetooth* do usuário indicando o trajeto a ser percorrido pela frota. O veículo seguidor é o segundo veículo, este possui sensoriamento para poder localizar o líder e identificar possíveis mudanças de direção feitas por ele, além de realizar uma estimativa da distância em que se encontra o líder. A figura 6 mostra as tarefas de interesse a serem executadas pelo veículo seguidor, onde é exemplificado que o seguidor deve manter uma distância mínima de segurança do líder e, ao identificar o alvo na traseira do líder, determinar o deslocamento lateral para verificar mudanças de direção feitas pelo líder. Logo estas tarefas são as estimativas de deslocamento lateral e distância líder-seguidor.

1.4 Sensores

Para desenvolver o projeto proposto, é viável empregar tecnologias de sensoriamento capazes de fornecer dados relativos a localização de objetos que possam estar no caminho, como também determinar a distância relativa entre o líder e o seguidor. Para isso é possível a utilização de sensores ultrassônicos, radares, câmeras, Lidar, entre outros, onde, cada um desses sensores apresentarão vantagens e desvantagens para que se alcance o desejado.

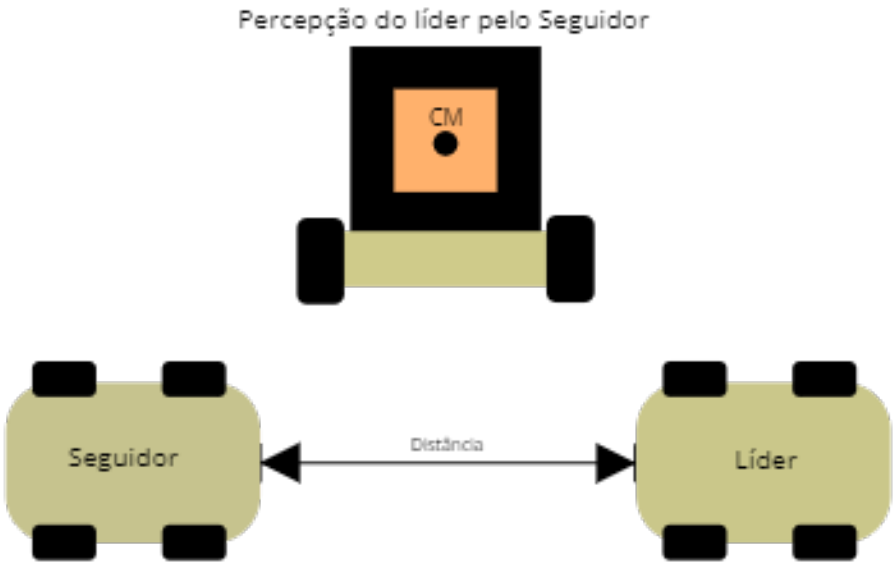


Figura 6 – Modelo Líder-Seguidor estudado.

1.4.1 Sensores Ultrassônicos

Os sensores ultrassônicos atualmente tem seu uso mais generalizado dado seu valor comercial baixo, seu funcionamento é baseado na velocidade de propagação de onda sonora emitida pelo emissor. O sensor emite um pulso de onda sonora de alta frequência e o receptor espera o retorno desta onda, que sofrerá reflexão ao encontrar um objeto. Através do tempo que a onda levou para ser emitida e recebida é capaz de se determinar a distância do sensor até o objeto em que a onda se chocou. Este tipo de sensor é bastante utilizado em veículos para auxílio ao realizar manobra de estacionamento, pois o sensor tem uma precisão maior com o veículo em baixa velocidade e baixo alcance. Na figura 7 tem uma exemplificação do funcionamento do sensor ultrassônico. (WEI, 2015)

Apesar do sensor ultrassônico parecer interessante no fato de determinar a distância entre o líder e o seguidor, ele não poderia ser utilizado como o sensor principal, pois uma das desvantagens deste elemento é não poder distinguir forma ou tamanho do objeto que está a sua frente, assim o seguidor poderia perder mais facilmente sua referência já que não conseguiria determinar com exatidão qual seria o líder. Entretanto este tipo de sensor pode ser utilizado como auxiliar para, por exemplo, situações de parada de emergência. Como fazer com que o veículo pare ao encontrar um obstáculo imóvel à sua frente.

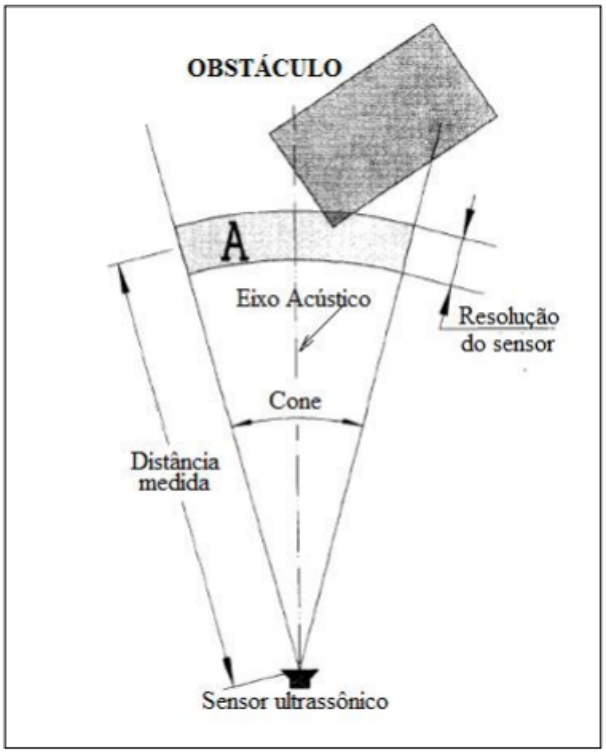


Figura 7 – Exemplificação do funcionamento de sensores que utilizam a propagação de onda para mensurar distâncias.(BORENSTEIN; KOREN, 1991)

1.4.2 Radares

Os radares, outro tipo de sensor para mensurar distância, também são comumente aplicados em veículos autônomos. O princípio de funcionamento deste tipo de sensor é semelhante ao do ultrassom, há um emissor que irá produzir pulsos de onda, mas na frequência de rádio, e quando essa ondar alcançar algum objeto, irá refletir e retornar ao sensor, assim um receptor capta o sinal. Considerando o tempo de propagação da onda e a velocidade das ondas de rádio no meio em que estão inseridas é capaz de se determinar a distância de um objeto ao sensor. Como este sensor utiliza frequências de rádio, tem capacidade de mensurar distâncias maiores do que os sensores ultrassônicos.

Apesar deste sensor possibilitar a medida de distâncias maiores do que o sensor ultrassônico, este elemento apresenta a mesma desvantagem verificada no sensor ultrassônico, fazendo com que o líder não seja determinado, podendo acarretar problemas para que o seguidor determine a trajetória do líder. Este sensor também pode ser utilizado como um sensor auxiliar.

1.4.3 Lidares

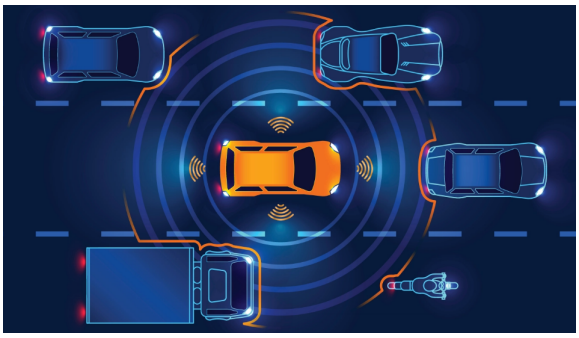
O sensor Lidar (*Light Detection and Ranging*) é uma tecnologia óptica de detecção remota que não só possibilita que realize a varredura do ambiente ao seu redor considerando um limiar, dependendo do tipo de sensor usado, mas também sua utilização possibilita obter

a distância de um determinado objeto. Este dispositivo é ideal para aplicações robóticas que exigem detecção e localização de obstáculos. Sua função princípio é o mesmo que um telêmetro simples, uma luz é emitida e sempre que reflete em uma superfície e é recebido pelo sensor, ele calcula a distância percorrida considerando o tempo que passou desde que foi emitido. Na figura 8a tem um exemplo de sensor lidar modelo HDL-64E da *Velodyne* que esta sendo usado no projeto de veículos autônomos da *Google*. Na figura 8b mostra o princípio de funcionamento do sensor lidar para conseguir rastrear os objetos ao seu redor e poder mensurar a distância entre ele e os veículos ao redor.

Este tipo de sensor apresenta grandes vantagens no âmbito de veículos autônomos, entretanto ele pode apresentar algumas complicações, pois esta é uma ferramenta que capta várias posições, formando uma nuvem de dados que apresentam complexidade para a interpretação. Isso pode acarretar em um processamento mais exaustivo para poder identificar o líder.

Uma outra desvantagem que pode ser destacada deste tipo de sensor para este projeto em específico é o alto custo agregado, pois pode ser necessário a utilização de mais sensores em conjunto para poder determinar tanto a sua localização como orientação. Isso faz com que o custo-benefício seja baixo.

(a) Exemplo de sensor lidar utilizado para veículos autônomos. (VELODYN, 2022)



(b) Exemplo de aplicação dada pelo sensor. (SUSLOV, 2022)

Figura 8 – sensor lidar

1.4.4 Câmeras

A câmera é outro sensor de bastante utilidade para veículos autônomos, ela é um instrumento óptico capaz de capturar diferença de luminosidade para formar a imagem. As câmeras digitais são compostas de sensores fotográficos (ou sensores de imagem) que podem ter em sua composição milhões de fotodiodos, que são sensíveis a intensidade luminosa. Quanto maior o número de elementos fotossensíveis, melhor é a resolução da câmera, esses elementos fotossensíveis são responsáveis pelos pixels da imagem. Os sensores de imagem podem ser feitos a partir de duas tecnologias: A CMOS e a CCD.

Esse tipo de sensor traz um custo-benefício maior entre os sensores citados, pois apresenta custo razoável, podendo identificar o líder com facilidade, além de poder ser possível mensurar a distância entre o líder e o seguidor de maneira razoavelmente precisa utilizando a distância focal da câmera. Entretanto este tipo de sensor apresenta desvantagens em ambientes com pouca visibilidade, fato que não ocorre com os sensores Lidar, afetando no desempenho em localizar o alvo. Na figura 9 verifica-se um tipo de sensor que é compatível com o projeto.

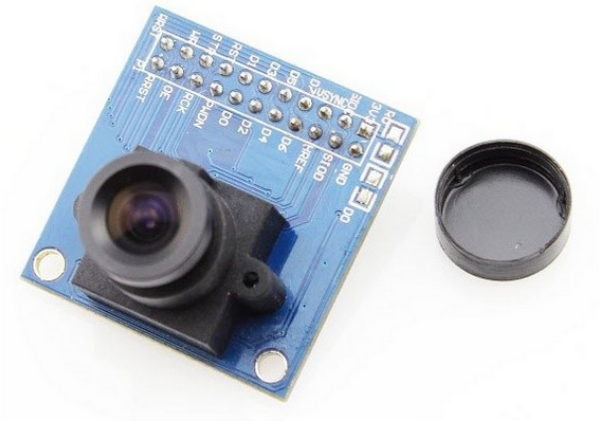


Figura 9 – Sensor câmera OV7670 que será utilizado durante o projeto.(INFINITO, 2020)

Para a realização deste trabalho optou-se, inicialmente, por usar uma modelo de câmera com sensor de tecnologia CMOS, essa escolha foi baseada levando em consideração custo-benefício do sensor em relação ao outros citados.

1.5 *Processamento Embarcado*

Como foi citado em 1.2.2 as tarefas realizadas, principalmente pelos seguidores, em um pelotão apresentam características de tempo real, ou seja, além das tarefas terem que atender os requisitos lógicos para o seu funcionamento pleno, deve-se respeitar o requisitos de natureza temporal, como período e *deadline*. Logo, dado a natureza do problema, necessita-se de *sistemas embarcados* de alto desempenho.

Em muitas aplicações computacionais de alto desempenho tem-se processadores de propósito geral como o coração do sistema, fornecendo flexibilidade e a capacidade de executar uma grande variedade de tarefas. Entretanto, embora sua flexibilidade seja vantajosa, isso resulta em um desempenho mais baixo já que a maneira como as tarefas são armazenadas, decodificadas e executadas faz com que o programa se torne forçadamente sequencial. Logo, para alcançar um maior desempenho, tem-se os dispositivos de aplicação específica , os ASICs (*Application Specific Integrated Circuits*). Entretanto a adoção de um processador para uma aplicação específica pode gerar um custo elevado. (BONDALAPATI; PRASANNA, 2002)

Então, com o objetivo de utilizar a flexibilidade gerada pelos processadores de propósito geral e o alto desempenho alcançado por ASICs, torna-se viável a utilização de arquiteturas reconfiguráveis do tipo FPGA - SoC (*Field Programmable Gate Array*). Neste tipo de arquitetura o *hardware* é modificável, se adaptando ao contexto em que está inserido, no caso específico da arquitetura da FPGA- SoC, o *hardware* reconfigurável exerce função de coprocessador fazendo com que tarefas que exijam mais poder de processamento sejam executadas de maneira mais rápida. (SCOTT HAUCK, 2008)

Considerando as vantagens de uma arquitetura híbrida reconfigurável, optou-se por utilizar uma FPGA, onde de acordo com (BONDALAPATI; PRASANNA, 2002), o processador hospedeiro presente será responsável por executar tarefas de controle do sistema, escalonar os dados de entrada e saída e realizar a interface externa, enquanto o *hardware* reconfigurável será responsável por realizar as tarefas mais custosas de captura e processamento de dados dos sensores.

1.6 Plataforma Veicular

Os veículos a serem utilizados na construção da plataforma serão veículos não holonômicos do tipo *skid steer*, apresentando na figura 10. Este tipo de veículo foi levado em consideração devido ao seu amplo uso na área de robótica, além de dar mais liberdade de movimentos, como, por exemplo, o giro em seu próprio eixo, que será útil ao seguidor. Outros modelos mais próximos de veículos reais poderiam ser utilizados, porém não são necessários, a priori, para os objetivos deste trabalho.

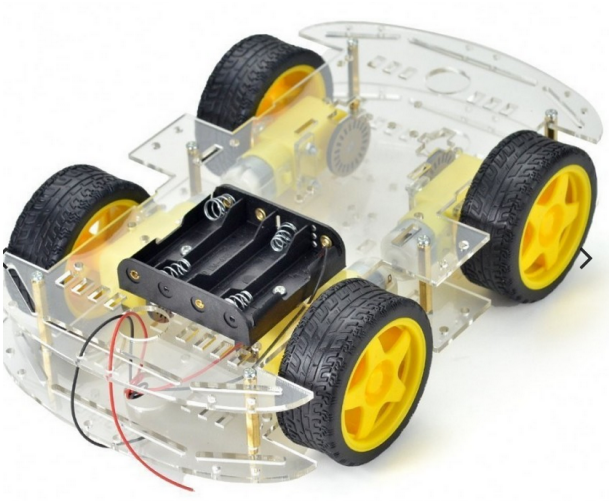


Figura 10 – Modelo de veículo utilizado para a realização do *platooning*.(INFINITO, s.d.[a])

Esse modelo de plataforma possui quatro motores de corrente contínua acionáveis individualmente, possibilitando diferentes formas de controle de trajetória.

1.7 Objetivos do Projeto

Com o desenvolvimento deste trabalho pretende-se implementar uma plataforma de testes para estudos de controle de um pelotão de veículos autônomos formado por dois veículos.

Como objetivos específicos para projeto tem-se:

- Construir a plataforma robótica composta por um líder e um seguidor;
- Implementar uma arquitetura para a realização de captura e processamento de imagem em uma plataforma híbrida FPGA-SoC;
- Implementar uma estratégia de rastreamento do líder pelo seguidor;
- Testar estratégias de controle para o pelotão.

1.8 Estrutura do Texto

Esta monografia está organizada da seguinte forma:

No capítulo 2 serão apresentados os modelos definidos para o líder e o seguidor do pelotão descrevendo os problemas a serem resolvidos com a abordagem proposta. Descreverá os algoritmos e técnicas utilizada para que se possa conseguir rastrear o líder de maneira eficiente. Também será apresentado o hardware escolhido para ser embarcado tanto no veículo líder como no seguidor e mostrar os conceitos adotados para realizar o rastreamento do líder pelo seguidor.

Em seguida, o capítulo 3 irá abordar os métodos adotados para o desenvolvimento da plataforma de testes, realizando uma descrição da plataforma, a interface de captura de imagens e como é realizado o acionamento dos motores a partir dos dados obtidos da câmera. Também será apresentando o projeto de integração hardware/ software desenvolvido com e sem aceleração do processamento para a realização da captura da imagens e rastreamento do líder, além de mostrar os sistemas de controle desenvolvidos, estimação da localização do líder e os resultados obtidos.

No capítulo 4 é apresentado a conclusão do projeto e os trabalhos futuros que podem ser desenvolvidos na plataforma.

2 Aspectos Teóricos e Desenvolvimento

2.1 Busca Bibliométrica

O sistema de pelotão veicular é um tema que já vem sendo estudado há algum tempo. O número de publicações científicas relacionadas a essa assunto é crescente. Muitos estudos estão voltados para os benefícios energéticos encontrados em pelotões, onde testes realizados em pista verificaram uma economia de até 6% de combustível para o líder e de até 10% para os seguidores. Na figura 11 verifica-se a quantidade de publicações envolvendo o *platooning* encontrada na plataforma SCOPUS nos últimos anos, mostrando a relevância que o tema vem ganhando.(BHOOPALAM; AGATZ; ZUIDWIJK, 2018)

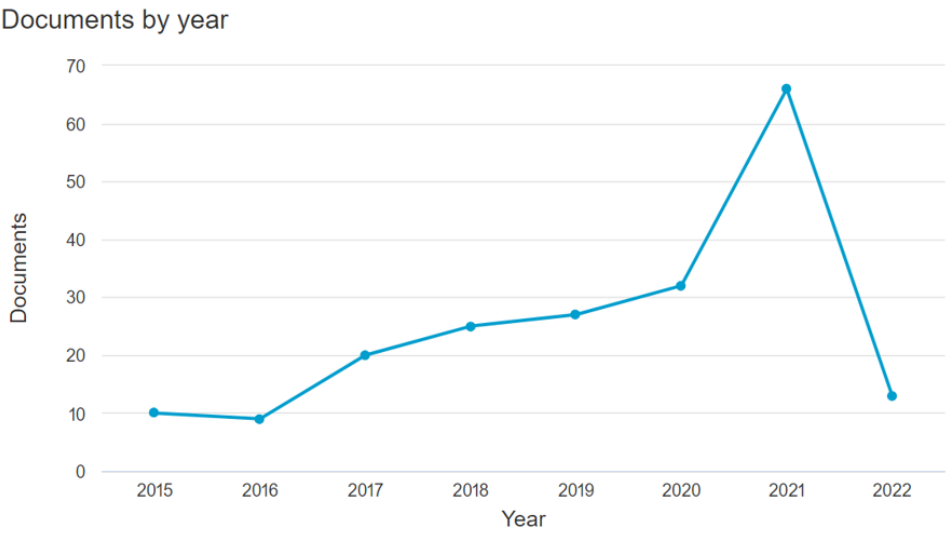


Figura 11 – Relação do número de publicações por ano encontradas na plataforma SCOPUS ao pesquisar as palavras chave *Vehicle AND Platooning*

Os benefícios citados são inversamente proporcionais à distância entre veículos: quanto menor a distância entre os veículos do pelotão mais benefícios são observados, pois uma maior quantidade de veículos podem ser alocados às vias, melhorando sua capacidade. No entanto a redução da distância entre veículos pode ser um fator crítico levando em consideração os aspectos de segurança. Este aspecto é bastante citado em várias referências, sendo um dos principais problemas no sistema de *platooning* (AXELSSON, 2017).

Levando isto em consideração, na plataforma a ser desenvolvida o veículo seguidor deverá executar as tarefas de localizar a posição do veículo líder e manter uma distância de segurança para que não ocorra o choque entre os veículos e não se quebre a formação.

2.2 Rastreamento Visual

Para que se possa alcançar o objetivo e realizar o *platoon* o seguidor deve ser capaz de localizar o líder e determinar a trajetória percorrida por este veículo e ser capaz e reproduzi-la. Então para isso o seguidor deve detectar um objeto e rastreá-lo. A detecção de um objeto, no caso analisado sendo o líder, é de grande importância, pois essa etapa interfere diretamente em etapas seguintes, como por exemplo, o planejamento da trajetória a ser seguida e a distância entre o sensor e o objeto a ser seguido.

Após detectar o objeto é possível rastreá-lo e obter informações úteis como, por exemplo, a velocidade com que se move o objeto, a trajetória que está seguindo e a orientação. A dificuldade que envolve este desafio é superar a perda de informação que pode ocorrer devido a complexidade do plano de fundo ou a dinâmica do ambiente em que o objeto está inserido, como mudanças de iluminação, sombras, etc. Então, o rastreamento de objetos pode ser entendido como segmentar os *pixels* pertencentes ao objeto de interesse a partir de cenas de vídeo. (FERREIRA, 2012)

Durante os últimos anos, vários métodos de rastreamento de objetos foram desenvolvidos ou aperfeiçoados. Esses métodos podem ser classificados de acordo com a abordagem adotada, podendo ser baseada em características, baseada em diferença entre *frames* e baseado em orientação. Uma técnica bem difundida é a subtração de plano de fundo, técnica baseada na diferença entre *frames*. Neste método os *pixels* são modelados com o tempo para determinar o objeto e a cena presente na imagem. Outra técnica é o fluxo Óptico, baseada em orientação, que visa detectar a direção do movimento. (HUANG et al., 2008) (FERREIRA, 2012) (MONTANARI, 2016)

No trabalho apresentado em (FERREIRA, 2012) há a implementação do algoritmo de subtração de fundo para detecção de um objeto móvel. Este trabalho tem semelhanças com o projeto desenvolvido por usar somente uma câmera e usar *hardware reconfigurável* para implementação da captura de imagens. Um dos pontos de divergência está no fato de que em (FERREIRA, 2012) utiliza-se escala de cinza enquanto neste projeto utiliza-se o espaço HSV (*hue, saturation e value*), pois segundo (BITTENCOURT, 2013) a segmentação em HSV apresenta melhores resultados.

Em (SÁNCHEZ-FERREIRA et al., 2013), é desenvolvida uma arquitetura de visão estéreo para sistema de medição de um Robô Subaquático utilizando duas câmeras que trabalham de maneira independente para detectar um objeto, calcular o seu centro de massa e determinar sua distância. Após essa verificação das câmeras as informações são enviadas para um coprocessador para verificar a disparidade das medidas e determinar com precisão a distância do objeto identificado. Os autores mostram que ao utilizar mais câmeras a precisão aumenta consideravelmente, mas também aumenta a complexidade e o espaço em *hardware* reconfigurável utilizado por causa da redundância.

Para o desenvolvimento deste projeto optou-se por um sistema monocular baseado em segmentação por cores no espaço HSV, que é um espaço de cores baseado no brilho, saturação e na tonalidade, para identificar e mensurar a distância do líder até a câmera.A escolha da segmentação por cores no espaço HSV é baseada nos resultados obtidos em (BITTENCOURT, 2013), neste trabalho mostrou-se que a segmentação apresenta resultados melhores considerando o espaço de cores HSV. Inicialmente o sistema será monocular para reduzir a complexidade do sistema e espaço em FPGA.

Para se conseguir estimar a distância entre os veículos, verifica-se, a cada *frame* recebido, a diferença da quantidade de *pixels* que representam a área do alvo localizada na traseira do veículo líder. Assim, quando o número de *pixels* da área apresenta um valor maior do que o valor tido como referência, significa que o líder realizou uma parada ou mudou o sentido de locomoção, isso fará com que o seguidor mude o seu sentido.

Para verificar mudanças na direção do líder, analisa-se o deslocamento do centro geométrico do alvo em relação a referência. A estimativa do centro geométrico se dá pela deformação do alvo, pois quando o líder realizar uma mudança de direção o alvo de formato quadrado se deformará, assim a câmera perceberá que o lado do alvo para onde o veículo se deslocou apresentará uma quantidade maior de *pixels*, assim ocorrerá uma mudança no centro geométrico.

2.3 Algoritmos e Técnicas

A realização da detecção do líder pode ser descrita pelo diagrama mostrado na figura 12. O sistema de detecção vai receber as imagens do sistema de captura já transformadas no espaço de cores HSV e realizará um processo de segmentação para que somente um alvo seja levado em consideração para determinar a distância deste objeto, calculando seu centro geométrico e determinando sua distância à câmera.

Figura 12 – Diagrama do algoritmo de processamento de imagens desenvolvido.

O sistema de captura de vídeo é referente ao interfaceamento do sensor em *hardware* reconfigurável, após a captura dos *frames* os dados passam por uma transformação no espaço de cores, mudando de RGB para HSV, pois este espaço trabalha com as grandezas referentes as cores de maneira desacoplada, dando mais liberdade para isolar informações, facilitando a segmentação e identificação do alvo.

Após esse tratamento do *frame* ocorre a segmentação por limiarização, onde o alvo a ser detectado é descrito por valores dentro de um limite de interesse, assim é possível distingui-lo do ambiente em que está inserido e com isso pode ser determinado o tamanho do objeto na imagem para poder calcular o seu centro de massa e estimar a sua distância em relação à câmera.

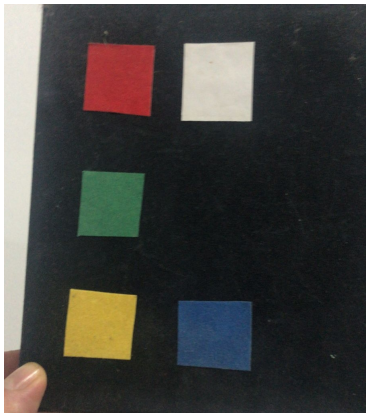
2.3.1 Segmentação por Cores

Como foi comentado na seção 2.3, antes de realizar a segmentação, a imagem passa por uma transformação no espaço de cores. O espaço de cores HSV utilizado pode ser descrito através das componentes de intensidade, saturação e brilho da imagem, este modelo mostra-se bastante adequado para descrever as cores em termos práticos, pois este espaço está associado à maneira como se interpreta as cores. Este modelo mostra-se adequado para o desenvolvimento de algoritmos de processamento de imagem baseado na descrição de cores por ter suas componentes independentes, fazendo assim com que se possa isolar de maneira mais eficaz interferências resultantes das variações de iluminação do ambiente, podendo detectar ou excluir cores. (BITTENCOURT, 2013) (GONZALEZ; WOODS, 2000)

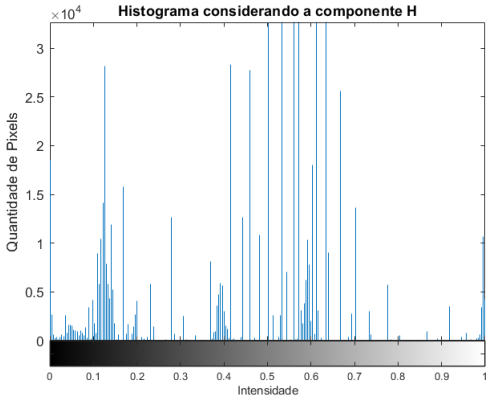
Antes de realizar a segmentação de uma imagem verificou-se qual das componentes do modelo HSV forneceriam um melhor resultado para identificar um alvo baseado em sua cor. Para isto montou-se um alvo teste de fundo preto e algumas cores para serem identificadas. A imagem deste alvo passou por uma transformação de RGB para HSV e foi montado histogramas de cada componente para serem comparados e verificar qual apresentava o melhor resultado. A figura 13 mostra os resultados obtidos.

Observando os dados mostrados em 13, verifica-se que o histograma 13d apresenta resultados melhores e com mais facilidade de interpretação dado as condições de iluminação da imagem, em 13d percebe-se a presença de sete pontos de máximo que ao serem analisados percebe-se que cinco destes picos são referentes às cores do alvo a serem identificadas. Com base neste resultado a componente V será a utilizada como entrada do algoritmo de segmentação.

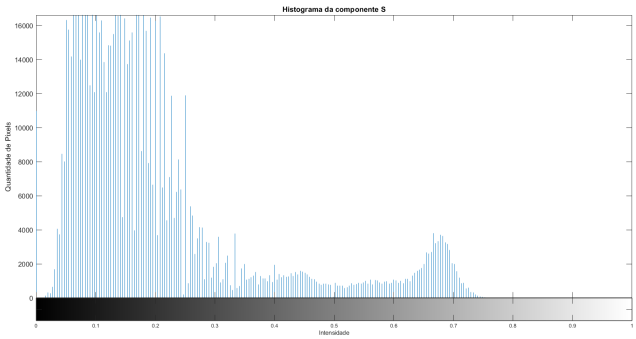
O processo de segmentação a ser realizado subdivide um imagem em regiões ou objetos que a compões. O nível de detalhe do resultado obtido dependerá dos objetos que devem ser identificados na região de interesse. A segmentação desenvolvida neste projeto é realizada através de uma limiarização, a limiarização é um processo pontual onde ocorre a comparação do valor de cada *pixel* da imagem com um valor limiar pré-determinado. Caso o *pixel* apresente valor igual ao limiar, a imagem segmentada apresentará valor 1 na posição deste *pixel*, caso contrário apresentará 0. (GONZALEZ; WOODS, 2000) (FERREIRA, 2012)



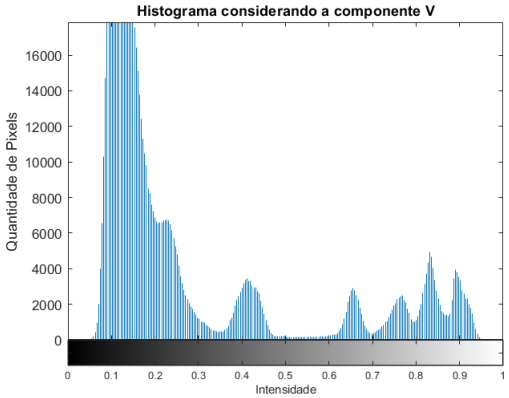
(a) Figura utilizada para os histogramas.(BITTENCOURT, 2013)



(b) Histograma H.



(c) Histograma S.



(d) Histograma V.

Figura 13 – Histograma de cada uma da componentes do modelo HSV para verificar qual componente é mais adequada para limiarização considerando como imagem de teste a apresentada em 13a

2.3.2 Cálculo do Centro Geométrico

Como foi mencionado na seção 1.6, para o seguidor estimar as mudanças de direções provocadas pelo líder, acompanha-se o deslocamento do centroide do alvo localizado na traseira do primeiro veículo. Quando o veículo realiza uma curva, haverá uma deformação no alvo que fara com que um dos lados apresente mais *pixels* que o outro, acarretando em uma mudança do centro geométrico.

Para realizar este cálculo considera-se o *frame* capturado uma matriz de *ixj* de tamanho *MxN* em que a localização do centro geométrico e definido pelas equações 2.2, 2.3 e 2.1, onde as equações 2.2 e 2.3 fornecerão uma posição *ij* de onde está o centro geométrico do alvo.

$$A = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \beta(i,j) \tag{2.1}$$

$$C_i = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} i.\beta(i,j)}{A} \tag{2.2}$$

$$C_j = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} j.\beta(i,j)}{A} \tag{2.3}$$

Das equações tem-se C_i e C_j que representam a coordenada *ij* do centro geométrico no *frame* atual, A corresponde a área, em *pixels*, ocupada pelo alvo na imagem e β é uma função binária que terá valor igual a 1 se o valor do *pixel* na posição *ij* estiver no intervalo fechado delimitado pelos limites superiores e inferiores da segmentação. Para poder ter um valor de C_i e C_j é necessário percorrer por todas as posições da imagem, assim os valores referentes ao centro de massa só estarão disponíveis ao final de cada *frame* obtido. No algoritmo 1 pode-se perceber como é realizada a segmentação e o cálculo do centro de massa ao receber um *frame*. O resultado obtido é então enviado para o bloco que calcula a distância entre o líder e o seguidor.

Algorithm 1 Segmentação e cálculo do centro geométrico

```
1:  while Pixel do
2:      if Pos_j < N then
3:          pos_j ← pos_j + 1
4:      else
5:          pos_j ← 1
6:          if pos_i < M then
7:              pos_i ← pos_i + 1
8:          else
9:              C_i ← Sum_i / Area
10:             C_j ← Sum_j / Area
11:         end if
12:     end if
13:     if L_sup >= Pixel >= L_inf then
14:         Area ← Area + 1
15:         Sum_i ← Sum_i + pos_i
16:         Sum_j ← Sum_j + pos_j
17:         β ← 1
18:     else
19:         β ← 0
20:     end if
21: end while
```

No algoritmo 1 percebe-se que o sistema funciona em *loop*, funcionando sempre que receber um novo valor de *pixel*. Na linha 13 verifica-se como é determinada a segmentação em que é definido um intervalo pequeno para delimitar o valor que estar se procurando. Se o dado recebido estiver dentro deste intervalo as variáveis de interesse são acrescidas de uma

posição. A variável β é uma matriz ixj que armazenará a imagem segmentada. Esta variável foi útil para verificar se a região de interesse estava sendo analisada.

2.3.3 Modelo Simplificado - Etiqueta monocromática

Para que se possa identificar o líder optou-se por alocar um alvo atrás deste veículo. Este alvo facilita o algoritmo de segmentação e a determinação da distância entre os dois veículos. Inicialmente optou-se por um alvo de duas cores, onde a cor principal tivesse mais destaque, se distinguido do ambiente e fosse fácil de segmentar. O alvo é formado pelas cores preto e laranja, onde a cor preta está presente na maior parte do alvo e a cor laranja está localizada em um retângulo ao centro.

2.4 Plataforma de Processamento Heterogêneo

A ideia principal para o desenvolvimento do projeto é utilizar um hardware reconfigurável. Nele será implementado os módulos de controle para os motores e da câmera. Para isso tem-se como ideia a utilização da arquitetura SoC Zynq que combina o processador ARM Cortex-A9, que pode ter um ou mais núcleos, com um *Field Programmable Gate Array* (FPGA).(CROCKETT et al., 2014)

O *Zynq* divide-se na parte de *Processing System* (PS) que é formada pelo processador, responsável pela execução do sistema operacional embarcado e os programas criados pelo usuário, e na parte de *Programable Logic (PL)*, onde se encontra o hardware programável formada pelas células lógicas da FPGA (CROCKETT et al., 2014). Na figura 14 encontra-se uma simplificação da arquitetura a ser utilizada, mostrando as duas partes existentes e o modo como é feita a interconexão entre elas, através de uma *Advanced Extensible Interface* (AXI).

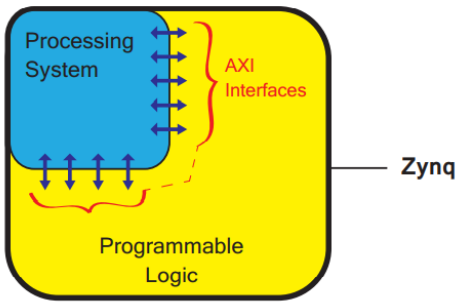


Figura 14 – Modelo simplificado da arquitetura ZYNQ.(CROCKETT et al., 2014)

O padrão *AXI* é a principal forma de conectar o PS com o PL, a maioria das outras comunicações presentes neste *hardware* se baseia neste padrão para realizar suas conexões. Esse tipo de comunicação é composto por diversos canais que fazem uma conexão dedicada ao PL e interconexões no PS. Essas interconexões são basicamente direcionadores de dados

entre as interfaces *AXIS* que por sua vez são conexões ponto-a-ponto responsáveis por passar endereços e sinais de sincronização entre clientes mestre-escravo. (CROCKETT et al., 2014)

A princípio utilizou-se a comunicação *AXI Lite* para implementação da comunicação entre PS e PL, apesar de não permitir a transferência de dados entre várias conexões, acarretando em *delay* na transmissão, essa é maneira mais simplificada de realizar a conexão. Essa comunicação é mapeada em memória fazendo com que somente um endereço e um conjunto de dado seja transmitido por vez.

A parte representada pelo *Programable Logic* é composto por sete estruturas principais. A *Look-up table* (LUT), que podem ser entendidos como blocos capazes de realizar funções lógicas através da descrição lógica em uma tabela, *flip flop*, que são elementos lógicos básicos capazes de armazenar valor, *Slice*, que é uma sub-unidade composta por 4 LUTs e 8 *flip flops*, o *Configurable Logic Block* (CLB), que pode ser descrito como um agrupamento de *slices* e uma matriz de comutação, elemento responsável por fazer a conexão entre os CLBs. O último elemento que forma o *Programable Logic* são os blocos de *inputs / outputs*, que farão a conexão com elementos externos à FPGA(CROCKETT et al., 2014). Na figura 15 mostra como é formado o *Programable Logic* exemplificando seus componentes.

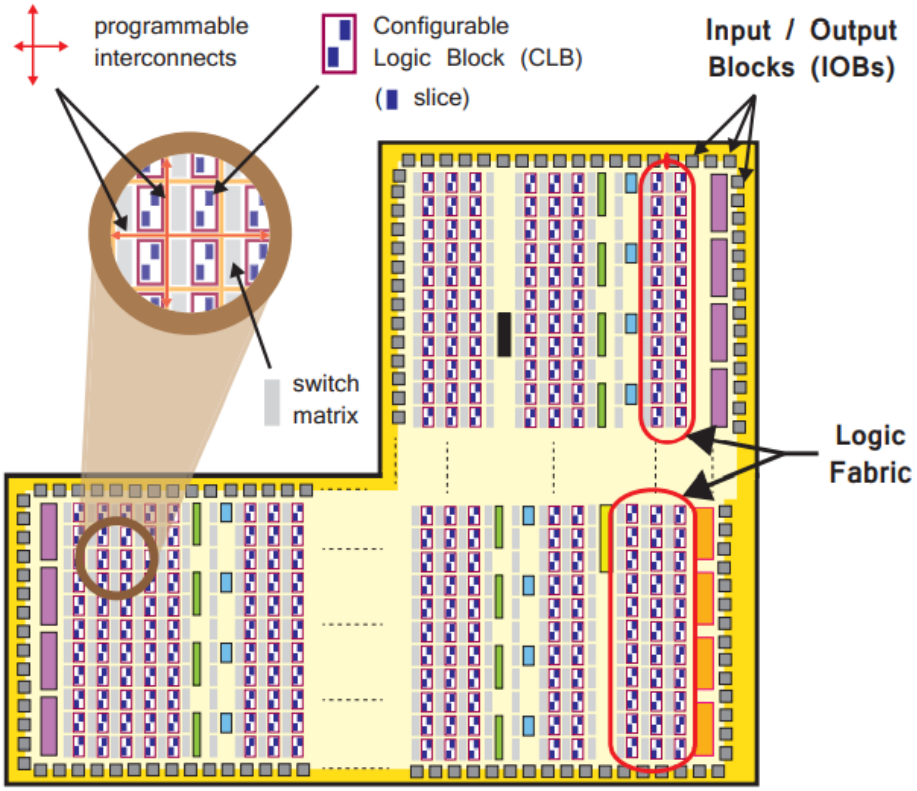


Figura 15 – Elementos constituintes do *Programable Logic*.(CROCKETT et al., 2014)

Além das *Look-up table* existe outra estrutura presente no PL. Esta estrutura são os blocos de memória de acesso randômico (BRAM) para requirimento de memória densa, este

recurso está integrado ao *array* lógico em um arranjo de colunas próximos uns aos outros. (CROCKETT et al., 2014)

O *hardware* reconfigurável a ser utilizado durante o projeto será a *Zynq Board (ZYBO)*. A ZYBO é uma plataforma de desenvolvimento com recursos construídos em volta do *chip xilinx Zynq-7010*, esse *chip* é baseado na arquitetura *Xilinx All Programmable System-on-Chip (AP SoC)* que tem embarcado um processador ARM Cortex-A9 *dual-core* de 650MHz integrado com um FPGA *xilinx 7-series*. Na figura 16 pode se observar como é a placa de desenvolvimento a ser utilizada, verificando os componentes presentes.

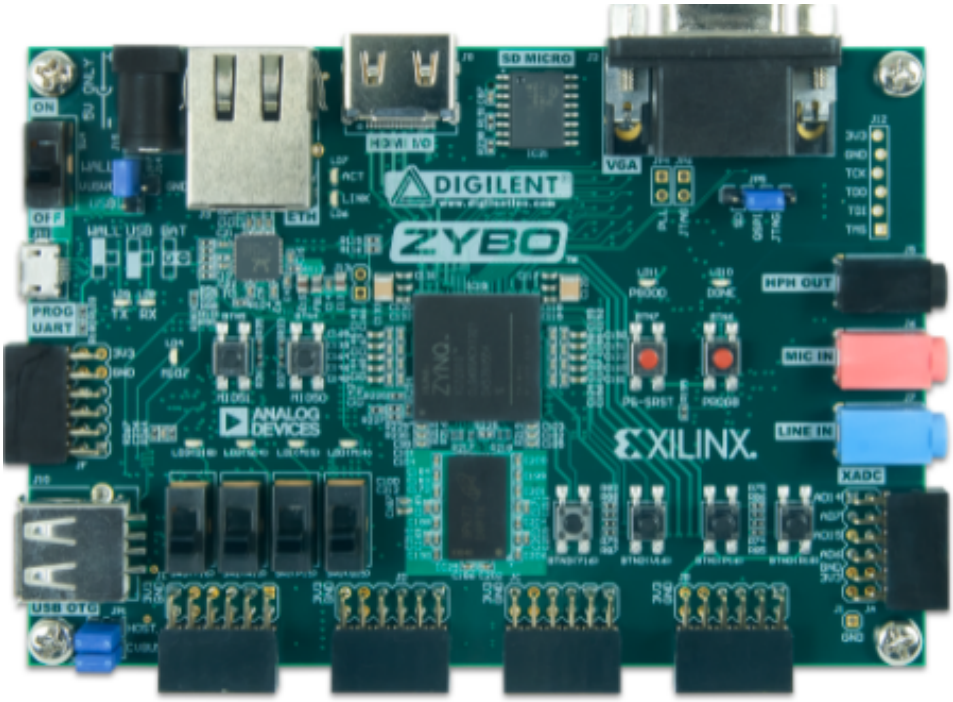


Figura 16 – Placa de Desenvolvimento ZYBO.

Entre as características do kit de desenvolvimento ZYBO encontra-se:

- 4400 *slices*;
- 240KB de BRAM;
- 6 portas PMOD(1 porta dedicada ao processador - 1 dedicada ao PL - 3 portas de alta velocidade - 1 dual analógico digital);
- Clock interno de até 450MHz;
- Entrada VGA;
- Controladores de periféricos de banda alta: Ethernet, USB 2.0, SDIO;
- Controladores de periféricos de banda baixa: UART, I2C, SPI, CAN;

- Entrada micro-SD com suporte para arquivos linux.

A ZYBO pode ser alimentada através de uma bateria externa limitada a 5.5V, onde a tensão mínima a ser utilizada é de 3.6V a depender da aplicação. Esta plataforma consegue fornecer em suas saídas até 3.3V de tensão com corrente típica de 1.5A.

3 Desenvolvimento da Plataforma

3.1 Plataforma Líder

O veículo líder é o responsável por receber comandos do usuário, esta plataforma tem como microcontrolador um $ATmega328P$ em um *arduino* Uno. Este *arduino* é responsável por realizar os acionamentos dos motores de acordo com o que é determinado pelo usuário. Os comandos são recebidos pelo *arduino* através de um módulo *bluetooth* que estabelece uma comunicação serial com o microcontrolador. Para que o seguidor possa seguir este veículo, em sua traseira, está localizado um alvo retangular laranja em fundo preto.

3.2 Plataforma Seguidor

O seguidor será o foco principal deste trabalho, pois é a parte autônoma do projeto. O veículo inicialmente possui somente a câmera como sensor, podendo ser adicionados outros sensores em projetos futuros. Para o gerenciamento das capturas de imagens e acionamento dos motores de maneira esperada a ZYBO agirá como o controlador do sistema, a forma como o controlador realiza a interface com a câmera e controla os motores será descrito nas seções 3.2.2 e 3.2.3.

Como foi informado na seção 1.6 o veículo utilizado é da classe *skid steer*, este tipo de veículo possui como característica a ausência de um sistema de esterçamento das rodas dianteiras, isso faz com que seja necessário o controle das velocidades relativas de cada uma das rodas para que se possa alterar a direção do veículo. Por causa desta característica, as manobras de locomoção em curvas irão requerer a derrapagem das rodas, dificultando a odometria e o controle de movimento ao comparar com o modelo diferencial e o modelo baseado na geometria de *Ackermann* (MANDOW et al., 2007)

Em (MANDOW et al., 2007) é desenvolvido um modelo cinemático para veículos do tipo *skid steer* com rodas baseado no centro instantâneo de rotação (CIR), este modelo desenvolvido será a base para o modelo cinemático e planejador de movimentos do seguidor. Na figura 17 mostra o modelo do veículo a ser analisado com seus CIRs, definido na figura como ICR_l, ICR_v, ICR_r. O modelo possui como variáveis independentes as velocidades do lado esquerdo, V_l, e do lado direito, V_r. Outra variável presente é a velocidade angular no eixo z do veículo, ω_z

Para o desenvolvimento deste modelo considera-se que sua origem está localizada no centro da área definida pelos contados das quatro rodas com o solo e seu eixo Y está alinhado à direção de movimento frontal do veículo. Outra consideração a ser feita é que este modelo

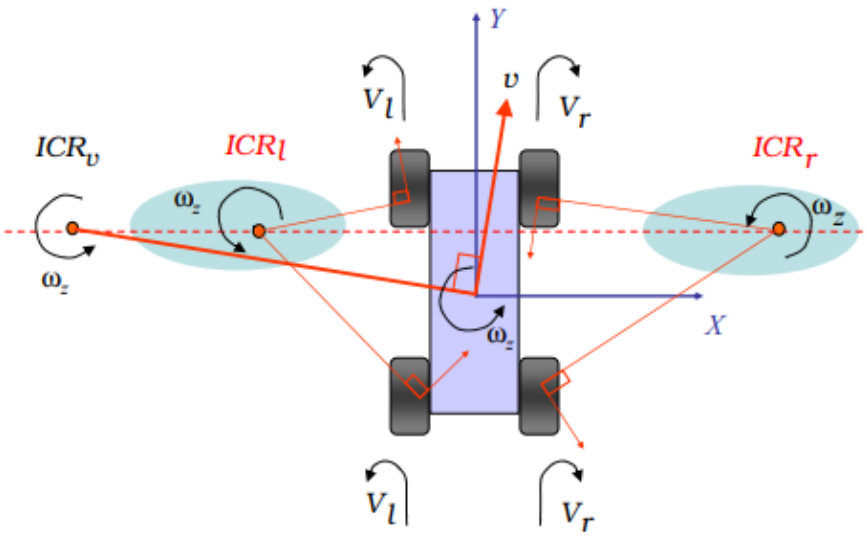


Figura 17 – Esquemático do veículo utilizado para determinar o modelo cinemático. (MANDOW et al., 2007)

apresenta duas variáveis de controle. A velocidades das rodas do lado esquerdo e direito. A terceira condição é que o centro geométrico do veículo encontra-se no centro geométrico da estrutura, assim a relação geométrica entre as posições dos CIRs e as velocidades de translação e rotação do veículo podem ser expressas pelas equações 3.1, 3.2, 3.3 e 3.4. (MANDOW et al., 2007)

$$x_{ICR_v} = -\frac{v_y}{\omega_z} \tag{3.1}$$

$$x_{ICR_l} = \frac{\alpha_l V_l - v_y}{\omega_z} \tag{3.2}$$

$$x_{ICR_l} = \frac{\alpha_r V_r - v_y}{\omega_z} \tag{3.3}$$

$$y_{ICR_v} = y_{ICR_l} = y_{ICR_r} = \frac{v_x}{\omega_z} \tag{3.4}$$

Os parâmetros α_r e α_l apresentados nas equações representam fatores de correção para englobar fatores mecânicos que podem afetar a velocidade nomina nas rodas. Manipulando as equações 3.1, 3.2, 3.3 e 3.4 Obtém-se a equação 3.5, onde o valor da matriz A é descrito na equação 3.6. (MANDOW et al., 2007)

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = A \begin{bmatrix} v_l \\ v_r \end{bmatrix} \tag{3.5}$$

$$A = \frac{1}{x_{ICR_r} - x_{ICR_l}} . \begin{bmatrix} -y_{ICR_v}.\alpha_l & y_{ICR_v}.\alpha_r \\ x_{ICR_r}.\alpha_l & -x_{ICR_l}.\alpha_r \\ -\alpha_l & \alpha_r \end{bmatrix} \tag{3.6}$$

O valor de A apresentado em 3.6 refere-se ao modelo assimétrico, em (MANDOW et al., 2007) também determina-se outra matriz, denominada matriz do modelo simétrico. Neste modelo considera-se que os CIRs dos lados esquerdo e direito estão simétricos em relação ao eixo Y, assim o valor de y_{ICR_r} vai ser igual a zero e os parâmetros α serão iguais além de $x_{IRC_l} = x_{IRC_r} = x_{IRC}$. Assim o modelo simétrico será dado por 3.7

$$A = \frac{\alpha}{2x_{ICR}} . \begin{bmatrix} 0 & 0 \\ x_{ICR} & x_{ICR} \\ -1 & 1 \end{bmatrix} \tag{3.7}$$

Para determinar os valores de α e x_{ICR} em (MANDOW et al., 2007) sugere a realização de dois experimentos para coletar a velocidade das rodas. O valor de x_{ICR} pode ser determinado com um teste de rotação pura em torno do eixo Z em que as velocidades das rodas esquerdas e direitas tem módulos iguais com sinais opostos. O valor de α é determinado com um teste de direção em linha reta, por uma distância conhecida. Este parâmetro tem valor contido entre 0 e 1, onde quanto mais perto de 1 melhor são as condições mecânicas do sistema.

No entanto, a determinação deste parâmetro de maneira experimental não é possível atualmente, pois a plataforma do seguidor ainda não possui *enconders* para mensurar a velocidade presente em cada uma das rodas. Então para determinar o valor de x_{ICR} utilizará uma abordagem mais teórica e focada no planejamento de rota a ser desenvolvida pelo seguidor. Para isso determina-se que o caminho a ser traçado pelo seguidor sejam descritos como círculos, ou seja, para que o seguidor se locomova para seguir o líder ele determinara uma circunferência de tal forma que o raio seja comum entre os dois veículos. A ideia do desenvolvimento deste raciocínio pode ser verificada na figura 18.

Observando a figura 18, que exemplifica o caminho que o seguidor deverá adotar ao seguir o alvo, verifica-se que o seguidor será tangente a uma circunferência e que o raio será coincidente com a distância do seu centro instantâneo de rotação. Logo a tarefa para o planejamento do caminho do seguidor está ligada a determinação do raio da circunferência que este veículo deve tangenciar. Considerando que o seguidor estará perto o suficiente do líder o problema traçado na figura 18 pode ser simplificado para uma situação que os dois veículos estão tangenciando o mesmo círculo. Assim, o problema pode ser simplificado na figura 19, onde R representa o raio do trajeto, d é a distância entre os veículos, e_{cm} representa a diferença entre a posição ideal do centro geométrico (CM) e a estimada. O

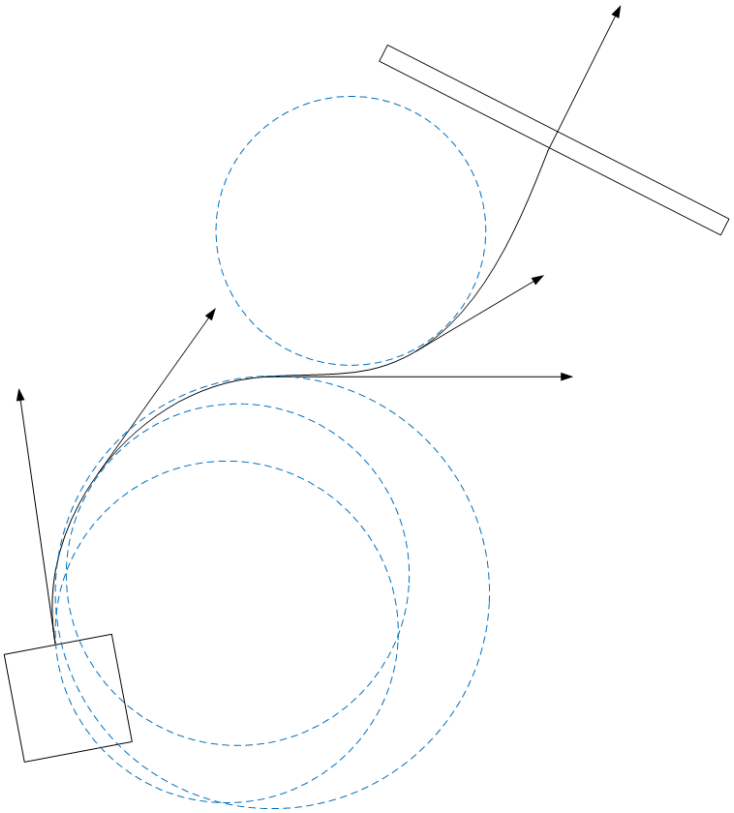


Figura 18 – Representação do caminho a ser traçado pelo seguidor.

ângulo ϕ representa a correção da orientação do seguidor e o ângulo θ é o ângulo do arco de circunferência que liga os dois veículos.

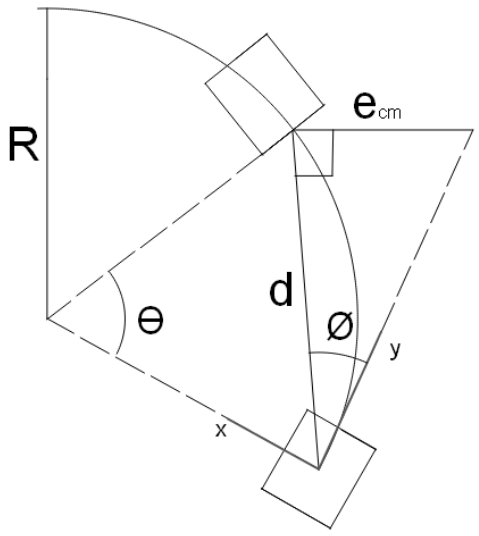


Figura 19 – Representação esquemática de movimentação do líder e seguidor.

Analisando o triângulo isósceles formado pelos segmentos de retas que ligam os dois veículos ao centro da circunferência e sabendo que a reta y é tangente à circunferência tem-se a equação 3.8 que relaciona o ângulo ϕ e θ .

$$\theta = 2.\phi \tag{3.8}$$

E, como é possível determinar o valor da distância entre os veículos e a diferença entre o centro geométrico referência e o atual, consegue-se determinar o valor de θ pela análise dos seus catetos, Assim, considerando a equação 3.8, tem-se a equação 3.9.

$$\phi = arctg\left(\frac{e_{cm}}{d}\right) \tag{3.9}$$

Considerando novamente o triângulo isósceles formado pelos segmentos de reta de ligam os dois veículos ao centro da circunferência e o segmento de reta que representa a distância d mostrada na figura 19 e considerando a lei dos senos. Tem-se uma equação, mostrada em 3.10, que relaciona as variáveis analisadas com o raio da circunferência trajeto do seguidor.

$$R = d \times \frac{sin(90° - \phi)}{sin(\theta)} \tag{3.10}$$

Entretanto esse sistema de planejamento de caminho não pode ser implementado, pois faltaram testes mais consistentes para determinar o seu funcionamento pleno. Então para que o seguidor fosse capaz de realizar sua tarefa foi implementado uma máquina de estados com 3 estados, onde o seguidor verifica o alvo, estima através do deslocamento do centro geométrico do alvo se ouve mudança de direção e, caso positivo, corrige sua orientação, após correção começa a se deslocar. A máquina de estados pode ser verificada na figura 20.

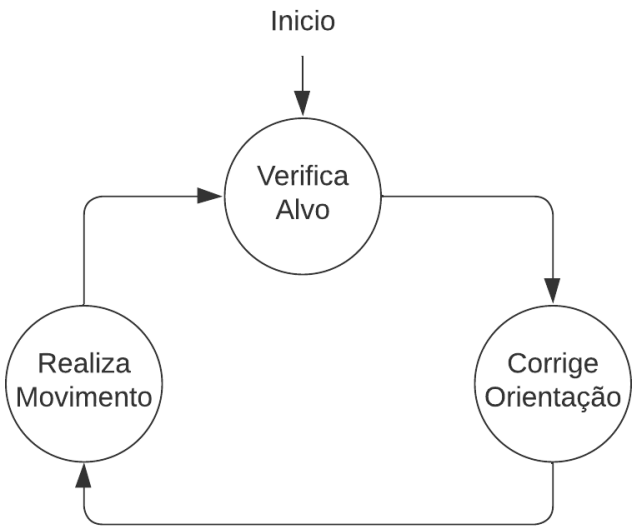


Figura 20 – Máquina de Estados Finitos representando o processo de locomoção do seguidor.

3.2.1 Sensor de Imagens

A câmera que será utilizada para este projeto será a câmera *OmniVision OV7670*, que é um *Soc* baseado em tecnologia *CMOS*. Este sensor tem diferentes funções de controle de imagem como, por exemplo, o modo de controle automático de Exposição (*AEC*), controle de balanço de brancos (*AWB*), cancelamento de ruído e de outras funções voltadas para controle de brilho, saturação e outros parâmetros de imagem. Também é possível trabalhar com diferentes resoluções com este sensor e diferentes espaços de cores. Estas funcionalidades são selecionadas através de um banco de registradores que são controlados através de uma interface *Serial Camera Control Bus (SCCB)*, que é um protocolo de comunicação para dispositivo *OmniVision*. Na tabela 1 mostra algumas especificações referentes à câmera e as configurações de resolução, espaço de cores e tamanho da imagem a ser utilizada. (OMNIVISION, s.d.)

Parâmetro	Valor
Resolução	QVGA
Tamanho da Imagem	320x240
Espaço de Cores	RGB565
Formato Óptico	1/6”
Tamanho do pixel	3.6x3.6 μ
Frame rate	30 fps
Consumo	ativado: 60mW
Package	24 pin- CSP2

Tabela 1 – Especificações técnicas para o módulo de câmera OV7670 CMOS. (OMNIVISION, s.d.)

Na figura 21 mostra o diagrama de blocos funcional da câmera, onde destaca-se entre os blocos o *SCCB interface*, responsável por configurar o sensor, o bloco *Image Array* que compõe de 320.128 *pixels*, onde, destes, 307.200 são ativos. Outro bloco de destaque é o *Image Scaler*, responsável por controlar os dados de saída adaptando-os ao formato escolhido e o DSP, que é responsável por interpolar o sinal original de 8 bits em padrão RGB incorporando um controle de qualidade no dado. (OMNIVISION, s.d.)

Na tabela 2 verifica-se os pinos de entrada e saída que se tem acesso,verificando-se a função de cada um destes.

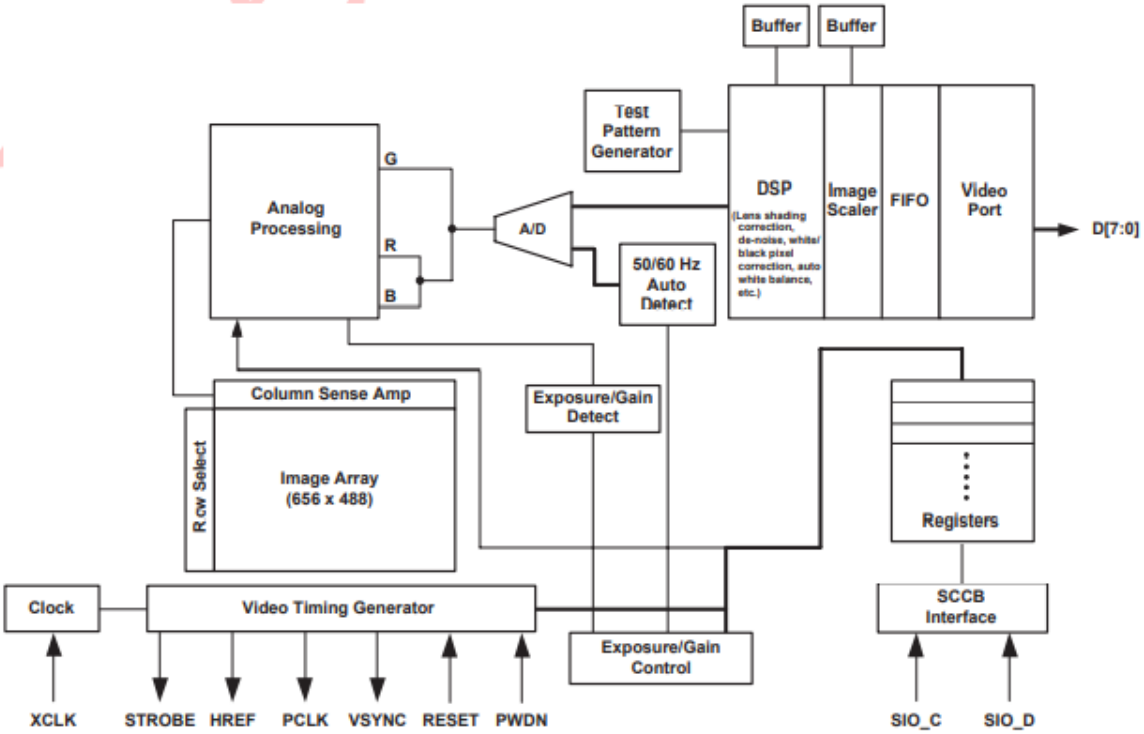


Figura 21 – Diagrama de Blocos Funcional do módulo de câmera OV7670.(OMNIVISION, s.d.)

Sinal	I/O	Descrição
d[7:0]	output	dados da câmera
XCLK	input	clock de entrada
PWDN	input	power down
PCLK	output	pixel clock
RESET	input	reset
SIOD	inout	dados do SCCB
SIOC	input	clock do SCCB
HREF	output	sincronização horizontal
VSYNC	output	sincronização vertical
GND	–	ground
3V3	–	alimentação

Tabela 2 – Pinos de entrada e saída presente no módulo de câmera OV7670. (OMNIVISION, s.d.)

A captura dos *pixels* realizada pelo sensor é sincronizada pelo *pixel clock* (PCLK). Na borda de decida deste sinal a entrada HREF da câmera mudará do nível lógico baixo para alto sinalizado que os dados referentes às imagem estão sendo capturados e quando o valor de HREF mudar novamente para o nível lógico baixo sinalizará que a câmera capturou a primeira linha da matriz imagem. Na figura 22 mostra a relação entre os sinais PCLK, HREF e D[7 : 0]. Segundo (OMNIVISION, s.d.) a fabricante do sensor informa que os atrasos de subida e decida destes sinais são de no máximo 5ns.

Para sinalizar que o quadro foi completamente capturado há o sinal de sincronização VSYNC. ao concluir a captura da imagem este sinal será ativado em forma de pulso e ao

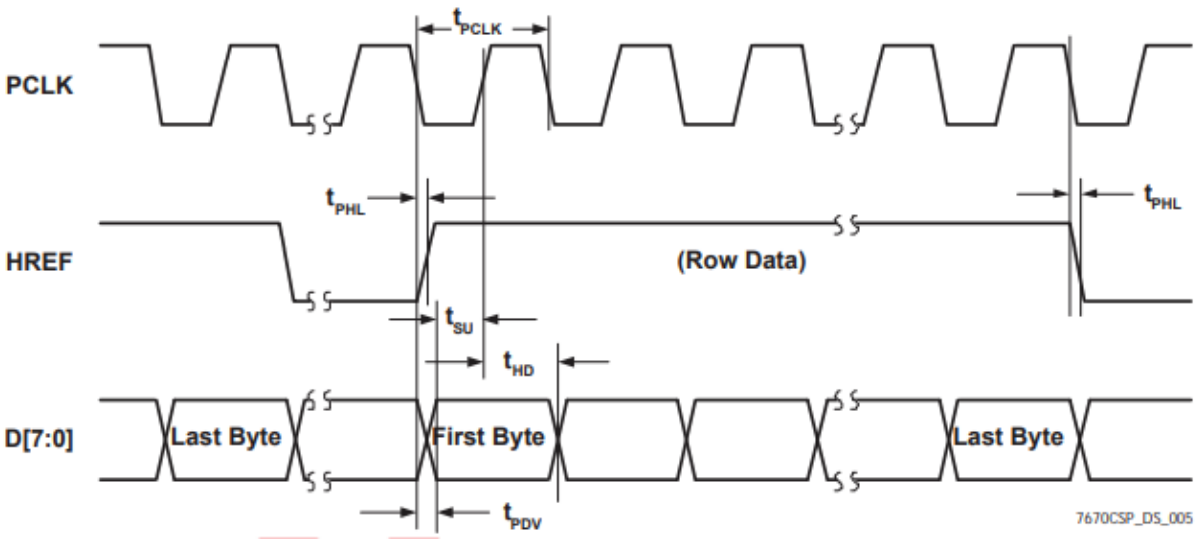


Figura 22 – Relação entre os sinais de *pixel clock*, HREF e dados capturados pelo sensor.(OMNIVISION, s.d.)

voltar ao nível lógico baixo começa uma nova captura. Na figura 23 mostra como funciona a captura de um quadro, considerando como exemplo o padrão VGA, onde a imagem tem dimensões 640x480.

Observa-se que o sinal HSYNC marca quando o sensor irá transferir as linhas do quadro capturado. O sinal tem o nível lógico modificado para zero e, após dois ciclos de PCLK, o sinal muda para 1, assim o sensor começa a transferir a primeira linha da matriz que apresenta os dados da imagem. Para sinalizar essa transferência o sinal HREF se mantém em nível lógico alto, esse nível durará até que todos 640 *pixels* sejam lidos, após isso o sinal volta para o nível lógico baixo e espera um novo pulso negativo de HSYNC.

O sinal VSYNC é o responsável por indicar quando um quadro completo for transferido. Ele apresentará nível 1 quando a transferência for completa. Os sinais HREF é estabilizado em nível baixo e o HSYNC estabilizado em nível alto, após um tempo para estabilização dos sinais de sincronização o VSYNC volta par ao nível baixo e aguarda a nova captura de dados.

A diferença entre o que foi apresentado na figura 23 e o projeto de captura de imagens desenvolvido é que os quadros são menores. No projeto a matriz com os dados da imagem tem dimensão 320x240. Como pode-se perceber na figura 23 leva-se dois ciclos de PCLK para que um *pixel* tenha seu valor totalmente determinado, pois trabalha-se com RGB444, no primeiro ciclo de PCLK os valores das componentes *Blue* e *Green* são transferidos, no próximo ciclo tem-se os valores da componente *Red*.

3.2.2 Interface de aquisição de imagens

Durante o desenvolvimento inicial deste projeto procurou-se integrar a câmera OV7670 com a plataforma de desenvolvimento ZYBO e verificar o armazenamento dos

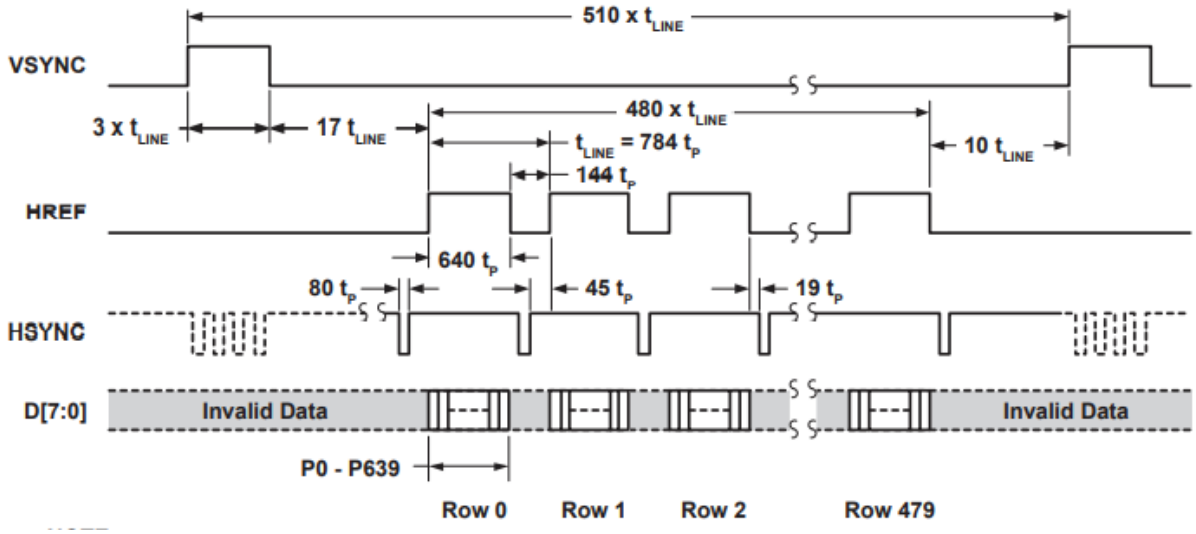


Figura 23 – Relação entre os sinais de sincronização VSYNC, HSYNC e HREF para a captura e transferência de uma imagem.(OMNIVISION, s.d.)

pixels capturados pela câmera na BRAM. A elaboração deste implementação tem como base os trabalhos desenvolvidos por(DA SILVA; LIMA; YUDI, 2020).

A figura 24 mostra os módulos necessários para fazer o controle da câmera, receber os dados do sensor e realizar o processamento desses dados para serem armazenados na memória. Pode ser observados cinco módulos que servem tanto para o controle da câmera como para amostragem dos dados obtidos. O módulo *OV7670 controller* é responsável pela inicialização da câmera. Os pinos SIOC e SIOD enviam dados e um clock, responsável por realizar a sincronia do módulo *OV7670 controller* com a câmera, através de uma interface SCCB. O controlador define o tamanho da imagem, formato dos dados de saída, contraste, orientação da imagem e outros parâmetros. (DA SILVA; LIMA; YUDI, 2020)

A estrada SW representa as chaves de seleção presente na placa de desenvolvimento, elas serão responsáveis por ativar o módulo de *Debounce* e mandar um sinal de *reset* ao módulo *OV7670 controller* quando necessário. A entrada PCLK é o sinal de clock responsável por sincronizar os dados recebidos da câmera com o módulo OV7670 Capture. Os pinos VSYNC e HREF sinalizarão quando terminar de receber um *frame* e quando termina a recepção de uma linha da imagem, respectivamente. O pino DADO manda um dado de 8 bits representando os *pixels* para serem processados no módulo OV7670.

O bloco *Camera Control IP*, responsável por fazer a comunicação da implementação feita no PL com o PS, utiliza uma interface de 32 bits, onde o endereço 0x00 é o sinal STOP que, quando está em nível lógico alto, para a aquisição de novos *frames* da câmera e realiza as capturas com nível lógico oposto. No endereço 0x04 encontra-se o sinal de VSYNC, este registro está configurando somente para leitura, através desse sinal presente neste endereço é possível saber quando a captura de um *frames* acaba. (DA SILVA; LIMA; YUDI, 2020)

O bloco de memória, que pode ser observado na figura 24, é um bloco de BRAM

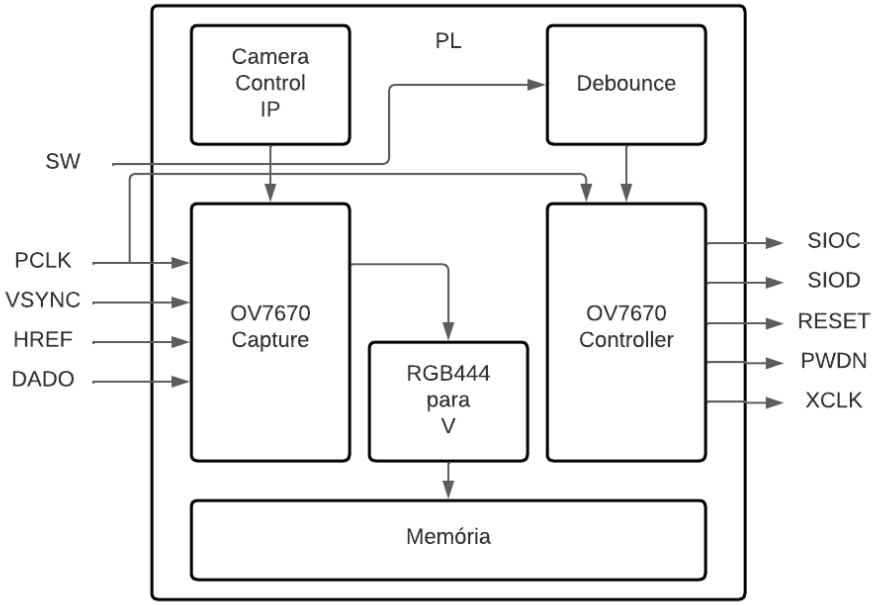


Figura 24 – Diagrama de Blocos para a aquisição de imagens pela câmera.

responsável por fazer a ligação entre os blocos de escrita de frames presente no PS com os módulos de processamento de dados presentes no PS. Para este bloco existem três parâmetros configuráveis: *Adress size*, *memory size* e *pixel size* que determinam a quantidade de endereços disponíveis, a quantidade máxima de *pixels* armazenados e atribui um número de bits para representar um *pixel*.(DA SILVA; LIMA; YUDI, 2020)

3.2.3 Interface de acionamento dos motores

Tensão DC	3V	5V	6V
Corrente (mA)	100	100	120
Taxa de redução		1:48	
RPM (com pneu)	100	190	240
Diâmetro do pneu (mm)		65	
Velocidade da Plataforma (m/min)	20	39	48
Peso do motor (g)		50	
Tamanho do motor (mm)		70x22x18	
Nível de ruído (dB)		<65	

Tabela 3 – Especificações técnicas dos motores DC a serem utilizados no desenvolvimento deste projeto.

Ambos os veículos utilizados neste projeto possuem motores de corrente contínua que operam com tensão de 3V, 5V ou 6V com discos para *encoders* que possibilitam mensurar o deslocamento do veículo, entretanto esses dispositivos não foram utilizados a princípio, fazendo com que não seja possível determinar com precisão a velocidade dos veículos.

Na tabela 3 encontra-se as especificações técnicas relacionadas aos motores presentes na plataforma.

Para auxiliar no acionamento dos motores DC utiliza-se uma ponte H com o circuito integrado L298N. O uso deste tipo de circuito é vantajoso por permitir uma maior dissipação de potência em relação ao acionamento direto pelo *hardware* utilizado, além de possibilitar de maneira fácil alterar o sentido de rotação do motor. Entre as características encontradas na ponte H utilizada destaca-se o grande intervalo de tensão de operação, variando de 4.5V até 46V, podendo fornecer até 2A para cada motor ligado ao circuito e uma potência máxima de 25W. Na figura encontra-se o modelo de ponde H utilizada no projeto.(BRAGA, 2017)

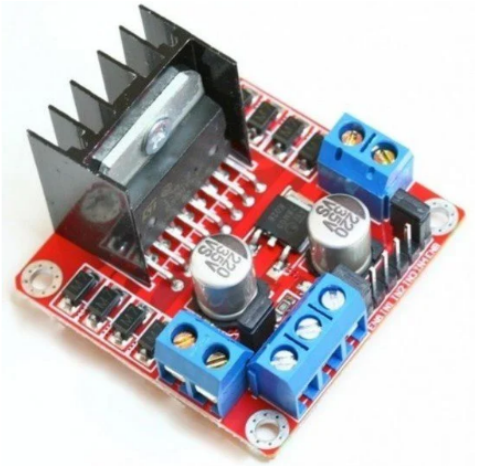


Figura 25 – Modelo de ponte H L298N.(INFINITO, s.d.[b])

Para que os motores apresentem a velocidade de rotação necessária utiliza-se módulos PWM (*pulse width modulation*) para controlar a potência na carga. A vantagem de utilizar esse tipo de tecnologia é que o controle de velocidade é mais preciso e evita do motor dar solavancos na partida. O PWM é uma técnica de modularização de sinal digital utilizando pulsos que, dependendo da frequência, resultará em níveis de tensão média variados. Esse resultado é alcançado através de contadores que determinarão o tempo em que o sinal apresentará o máximo valor de tensão e quando não apresentará. (BRAGA, 2017)

Na figura 26 verifica-se um exemplo de PWM, onde configura-se a largura do pulso em 50%, assim a tensão média aplicada aos motores será a metade da tensão máxima que o motor pode receber. O que definirá que o valor será a metade do valor de tensão máxima é a relação entre o período, definido como t na figura, e o tempo do ciclo ativo, definido como t_1, essa medida será o *duty cicle*, logo, a variação do *duty cicle* acarretará em um valor de tensão média diferente, pois o sinal PWM poderá apresentar uma largura de pulso maior ou menor.

Como foi informado, para implementar um sistema que forneça um sinal PWM basicamente é necessário de um contador crescente e um comparador, assim o contador vai sendo incrementado até o valor de parada que definirá o *duty cicle*. Na figura 27 exemplifica como é o bloco de PWM construído. Este sistema possui duas entradas,relacionadas ao

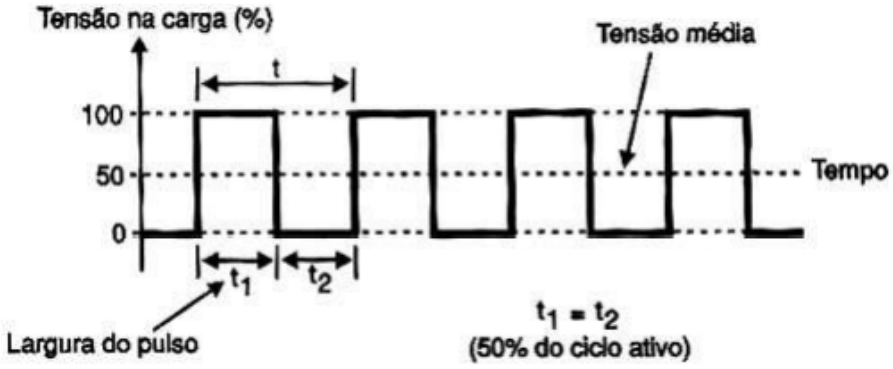


Figura 26 – Exemplificação de um sinal PWM em 50%.(BRAGA, 2017)

sentido de rotação do motor e largura do pulso, e duas saídas, sinal PWM e o sentido de rotação.

Para determinar a resolução do contador crescente do sistema considerou-se a escala de tensão para ativação do motor DC, assim determinou-se que 8 bits seriam suficientes para que as alterações no sinal PWM fosse significativas. Para que o sistema possa alcançar o máximo valor de tensão optou-se para que o sinal de entrada, *duty*, fosse de 9 bits, pois o sinal não irá para nível zero quando o contador chegar ao máximo valor de contagem, 255. Isso ocorreria caso a entrada *duty* tivesse 8 bits, pois para que a saída pwm_{out} apresente um nível alto, o valor do contador tem que ser menor que o valor da entrada *duty*, e, caso a entrada apresentasse o mesmo número de bits do contador, o valor 255 não representaria o valor máximo de tensão, já que neste valor o comparador iria colocar a saída pwm_{out} em zero quando comparada ao valor do contador. Para que isso não ocorra o valor 256 representará o valor máximo de tensão.

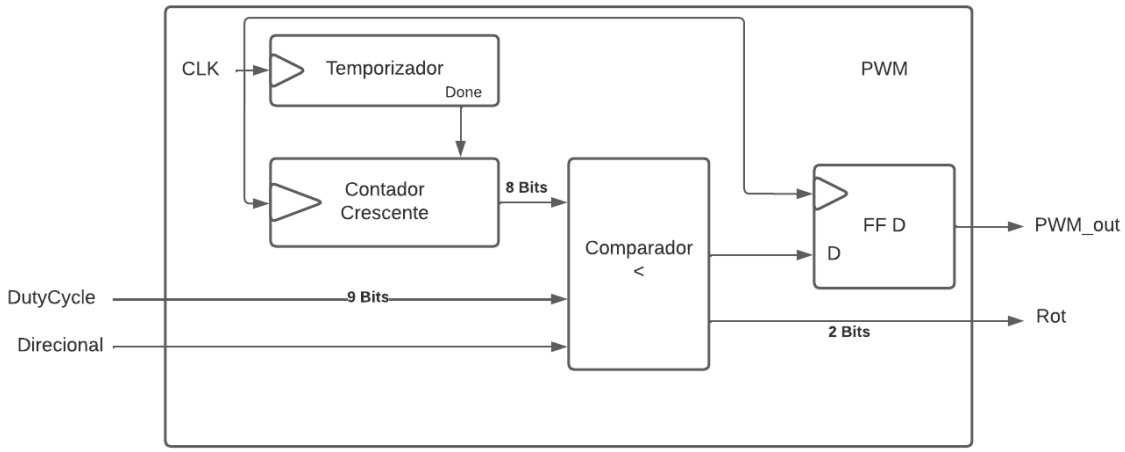


Figura 27 – Estrutura do Bloco de sinal PWM para controle de velocidade dos motores do seguidor.

Além do contador crescente e o comparador, o bloco PWM apresenta mais dois circuitos. Um temporizador e um *flip-flop D*. O primeiro circuito é basicamente um contador,

este circuito é necessário para que se possa definir uma frequência de PWM diferente da frequência do sistema. Isso é importante, pois o motor DC funciona em uma faixa de frequência definida, pois por ser uma carga indutiva, caso não seja acionado na frequência específica apresentará ruído, dificultando o controle. Portanto para que isso não ocorra a frequência deve considerar o tempo de contagem deste circuito. Isto é definido na equação 3.11, assim, para que o PWM apresente o sinal na frequência correta o circuito temporizador deverá ser incrementado para um valor determinado pela equação 3.12, onde T_{sys} é o período utilizado na arquitetura. A frequência desejada é de *2kHz*.

$$f_{pwm} = \frac{1}{2^r(T_{sys}(V_{time}+1))} \tag{3.11}$$

$$V_{time} = \frac{1}{2^r(T_{sys} \times f_{pwm})} - 1 \tag{3.12}$$

O segundo circuito é um *flip-flop D*, esse dispositivo é necessário devido a possíveis *glitches* na saída do bloco. Como há uma interação entre circuitos síncronos e um circuito combinacional poderá ocorrer ambiguidades no valor de saída e *flip-flop D* elimina essa ambiguidade. A equação 3.13 relaciona a saída do sistema considerando a entrada *Duty* determinada com a resolução do sistema e a tensão máxima disponibilizada para os motores.

$$pwm_{out} = \frac{Duty}{2^r} \times V_c \tag{3.13}$$

É importante salientar que a respostada dada pelo motor ao sinal PWM gerado também depende da carga disponível nas baterias presentes na plataforma, quando as cargas apresentavam baixa capacidade a potência entregue aos motores não correspondia ao máximo valor de PWM, fazendo com que os motores perdessem força.

3.3 Plataforma de Referência

A plataforma completa envolve uma integração entre o *Hardware* reconfigurável e o processador ARM, em que a interação entre as duas partes acontece através de processos de escrita de um *frame* na memória pela *FPGA* e leitura destes dados pelo ARM. Na figura 28 mostra a arquitetura implementada em que é possível verificar a integração ente o *hardware* reconfigurável (PS) e o programa embarcado (PS).

O sistema inicializa após o PS acionar um sinal de controle via barramento AXI, que configurará a câmera e aplicará um *reset* para limpar a memória. Após esse procedimento, os primeiros *pixels* serão escritos na memória para serem lidos pelo processador. Inicialmente a leitura destes dados na memória é feita utilizando o protocolo AXI *lite*, que apresenta um pequeno atraso no processo. Os recursos utilizados para implementação desta arquitetura

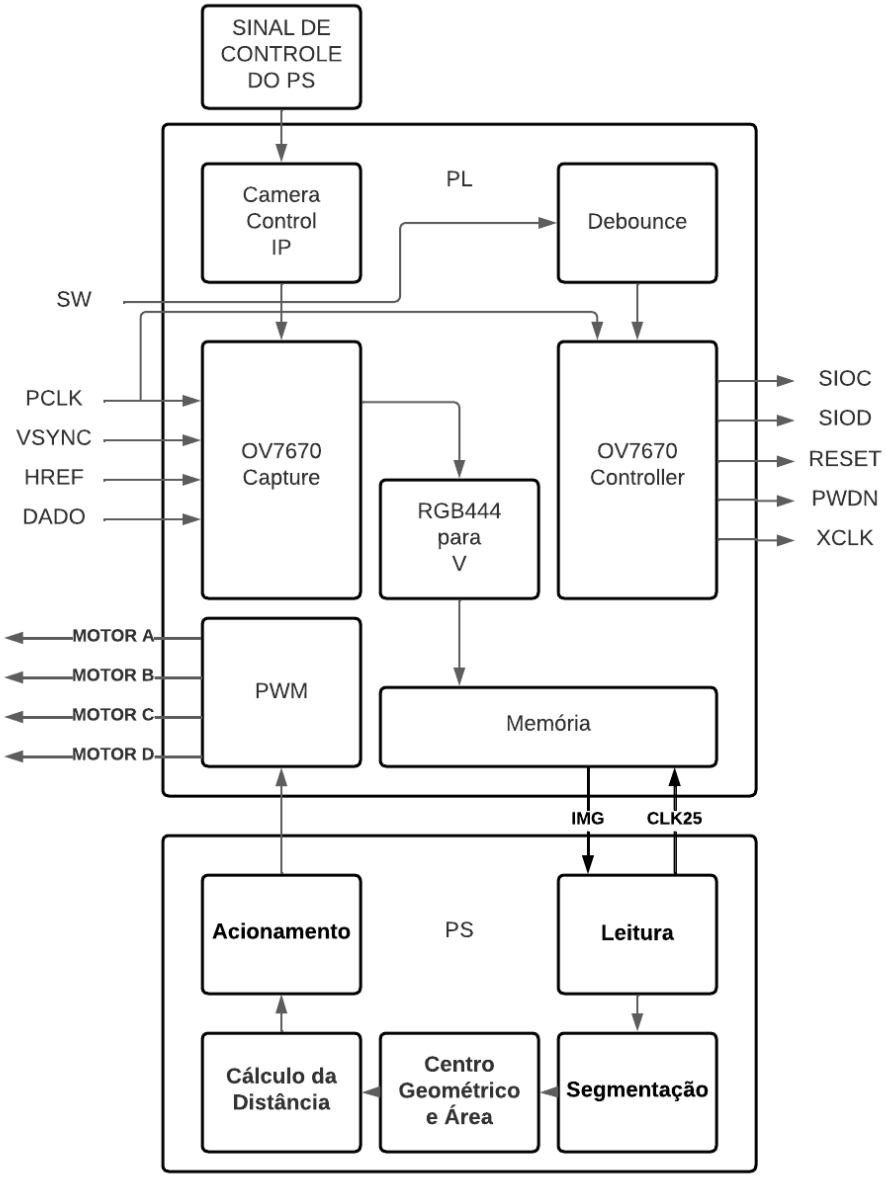


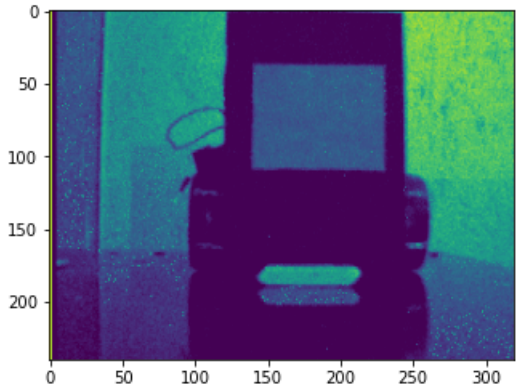
Figura 28 – Arquitetura desenvolvida para o seguidor.

podem ser verificados na tabela 4, onde verifica-se a quantidade de LUTs utilizadas, o número de pinos de entrada e saída presentes na arquitetura e a quantidade de blocos de memória utilizados.

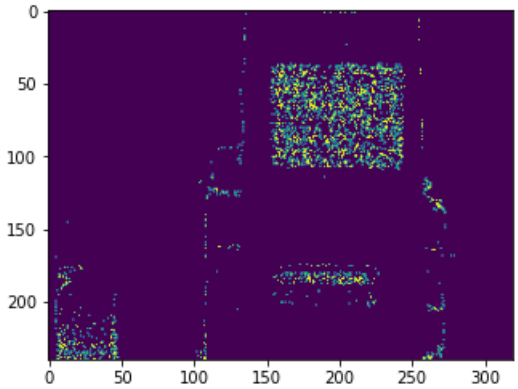
Recurso	Utilizado	Disponível	% utilizado
LUT	1264	17600	7.18
LUTRAM	62	6000	1.03
FF	1634	35200	4.64
IO	30	100	30
BRAM	36.5	60	60.83

Tabela 4 – Recursos utilizados da ZYBO ao implementar o sistema no Vivado usando *AXI Lite*.

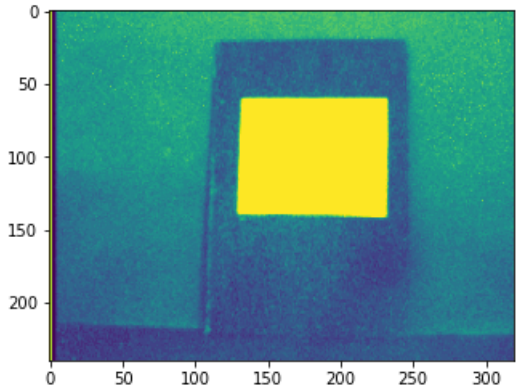
Após a leitura dos dados da memória, o processador passa para o processo de segmentação da imagem, realizando limiarização em torno do valor de cor presente no alvo a ser identificado. Na figura 29 mostra dois testes realizados mostrando um dos *frames* capturados pela câmera em cores artificiais, mostrado na figura 29a, e o resultado obtido após ser realizada a segmentação da imagem, figura 29b. Verifica-se ainda é possível verificar que a segmentação apresenta ruído de outros objetos, um dos motivos para isso é a iluminação presente no local e a qualidade da câmera, por este motivo fez um novo teste que pode ser verificado em 29c em que há mais luz. Na figura 29d percebe-se que o resultado foi mais satisfatório, pois o ambiente de testes tinha uma iluminação melhor. Para que as interferências verificadas em 29d não afetasse na determinação da orientação do seguidor, que é estimada através do centro geométrico na imagem, considera-se um valor que apresente *offset* na medida da posição ideal.



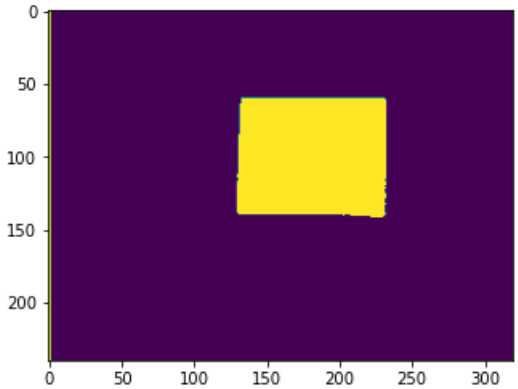
(a) Imagem Original colorida artificialmente.



(b) Imagem Segmentada.



(c) Imagem Original do 2° Teste com mais luz.



(d) Imagem Segmentada.

Figura 29 – Resultado da segmentação realizada na ZYBO.

Para verificar o tempo gasto para cada tarefa realizou-se um procedimento de captura de apenas um *frame* e mensurou o tempo gasto pelo processador em cada uma das tarefas. Assim percebe-se que o tempo de leitura dos dados da memória cria um gargalo no sistema, podendo acarretar em perdas de dados da câmera e fazendo com que a plataforma não funcione corretamente. Para tentar reduzir o gargalo foram propostos maneiras de acelerar

o processamento em hardware. Uma das maneiras é através do *video direct Memory Access* (VDMA) que elimina o uso da BRAM, pois os *frames* serão escritos diretamente na memória RAM do processador, então o acesso do processado ao valores de *pixel* é mais rápido comparado com o acesso na arquitetura implementada com protocolo AXI *lite*. O tempo mensurado em cada uma das tarefas pode ser verificado em 5.

	Config. Cam.	Controle	Leitura	Processamento	Distância
Tempo(s)	0.000003	0.0003	0.3840	0.0044	N

Tabela 5 – Tempo para a execução de cada tarefa da plataforma

3.3.1 Estimação de Distância em relação ao Líder

Para determinar a distância entre o líder e o seguidor utilizará como parâmetro a variação da extensão da área do alvo a cada *frame* capturado pela câmera. Assim, quando o alvo apresentar uma área extensa na imagem, então o seguidor deverá estar bem próximo do líder e caso apresente uma extensão menor, o seguidor deverá estar distante do líder.

Para determinar uma maneira de estimar esta distância realizou-se um experimento onde foi capturado imagens do alvo em diferentes distâncias verificando o efeito esperado. Para realização dos testes inicialmente o alvo ficou aproximadamente 15cm de distância da câmera e capturou-se um *frame* para análise, essa distância foi incrementada em 5cm a cada captura, até atingir o valor máximo de 100cm. Após a captura dos dados relacionado a área, verificou-se a relação entre a área estimada do alvo e a distância deste objeto até a câmera, essa relação pode ser verificada na figura 30.

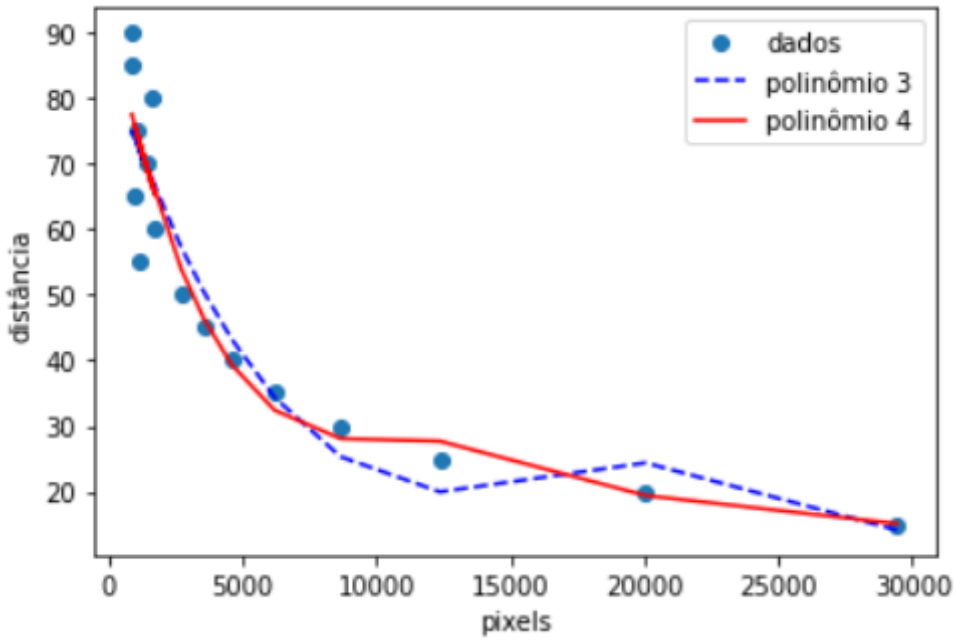


Figura 30 – Estimação de parâmetros para Distância usando aproximação polinomial.

Observando-se a figura 30 percebe-se que o modelo é não linear, isso é devido a distorção existente na lente da câmera e a falta da calibração. Sabe-se que a calibração da câmera apresenta grande importância, pois poderia melhorar a relação entre os parâmetros analisados, reduzido-se os erros devido distorções na imagem. Entretanto optou-se para que o modelo ajustado pela regressão polinomial abranja os efeitos de distorção de imagem, limitando possíveis erros.

Para verificar a relação entre a área, em *pixels*, e a distância, em centímetros, utiliza-se regressão polinomial. Este tipo de regressão foi utilizado por ser simples de ser trabalhado e incrementar o grau do polinômio até encontrar um adequado. Foram verificados dois modelos polinomiais que apresentaram valores de R-Quadrado consideráveis. O polinômio cúbico, que apresentou R-Quadrado de 87.63% e o polinômio de quarta ordem que apresentou R-Quadrado de 89.49%. Como o polinômio de quarta ordem apresentou uma proximidade com os dados maior, este foi o utilizado para a estimativa de distância no projeto. A equação 3.14 e a tabela 6 apresentam a função distância e os seus coeficientes.

$$d = p_1 a^4 + p_2 a^3 + p_3 a^2 + p_4 a + p_5 \tag{3.14}$$

p_1	p_2	p_3	p_4	p_5
1.355139×10^{-15}	$-8.966819 \times 10^{-11}$	2.027333×10^{-6}	-1.918485×10^{-2}	92.77373

Tabela 6 – Parâmetros da regressão polinomial de quarta ordem.

Após a determinação da função de distância realizou-se testes no sistema. Os testes consistiram em verificar a distância entre a câmera e o alvo modificando aleatoriamente a distância a cada nova captura de um *frame*.Os valores estimados de distância foram comparados aos valores reais e calculou-se o erro de cada medida. Essas informações podem ser verificadas na tabela 7.

Real (cm)	**Estimado(cm)**	**Erro(cm)**	**%e**
23	25.61	2.61	11.34
34	30.36	3.64	10.59
42.5	40.05	1.95	4.59
51.6	49.93	1.67	3.23
57	56.16	0.84	1.47
62	63.48	1.48	2.39
63.5	66.18	2.68	4.22
66.5	65.9	0.6	0.9
71	64.45	6.55	9.22
77	62.43	14.57	18.92

Tabela 7 – Dados de distância estimados comparados com o valor real.

Observando a tabela 7 verifica-se que a distância estimada tem um erro maior quando o alvo está muito próximo da câmera e quando estar muito distante. Esta característica pode estar associada as distorções presentes na câmera, além do ruído de segmentação verificado em algumas medidas. Como o veículo deverá ficar no intervalo de distâncias que apresentam menor taxa de erro, as distorções não afetarão consideravelmente o sistema.

3.3.2 Módulo de Acionamento

O módulo de acionamento é o responsável por receber o valor da distância mensurada e determinar um valor de *duty cicle* para envio, via barramento AXI, ao módulo de controle PWM dos motores. Para o funcionamento deste sistema leva-se em consideração que a distância é proporcional ao valor de acionamento, ou seja quanto maior a distância mais próximo de 1 será este valor, e quanto menor a distância, mais próximo de 0.

Também deve-se considerar a velocidade máxima dos motores e o *range* de distância para qual o modelo polinomial foi determinado, limitados entre 15cm e 100cm. Com isso, Quando a distância for igual ou superior a 100cm o valor de acionamento dos motores será contante e igual a 1, significando a entrega de máxima potência ao motores. Caso a distância seja igual a 15cm o valor de acionamento será 0, deixando o veículo parado, entretanto, se o valor for menor que 15cm, significar que o veículo líder pode ter invertido a direção de movimento, assim o seguidor deverá fazer o mesmo, para isso este bloco envia um sinal para inverter a rotação dos motores. A equação mostrada em 3.15 mostra como este valor de acionamento dos motores é definido caso a distância esteja entre 15cm e 100cm. A equação 3.16 determina o valor em casos de distância menor que 15cm. O valor enviado para o bloco PWM é calculado conforme mostrado na equação 3.17, onde r é a quantidade de bits da entrada *duty cicle*.

$$V_m = \frac{1}{85}d - \frac{3}{17} \tag{3.15}$$

$$V_m = -\frac{1}{15}d + 1 \tag{3.16}$$

$$duty = V_m \times 2^r \tag{3.17}$$

3.4 Aceleração de Processamento

Como observado na tabela 5 existe um gargalo no processamento causado por uma das tarefas, com isso procurou-se maneiras de acelerar o processamento. Uma das maneiras foi através da implementação do VDMA, esse sistema foi testado e apresentou bons resultados, acelerando na leitura de *frames*. A segunda maneira é transferindo o bloco que realiza

a segmentação e o cálculo do centro geométrico para o PL, assim eliminaria a memória, poupando recursos. Entretanto essa segunda maneira não pode ser implementada, sendo realizado somente testes.

3.4.1 Implementação de VDMA

Como este trabalho necessita uma memória para armazenamento de *frames* que serão alterados ao longo do tempo, é necessário um sistema que comporte esta estrutura e permita um acesso rápido para não se perder dados, então, para isto, existe um recurso específico para este caso, o AXI VDMA. Este tipo de sistema foi projetado para permitir uma acesso de alta largura de banda entre a interface de vídeo *AXI4-Stream* e a interface AXI4. Na figura 31 apresenta o diagrama de bloco referente a estrutura VDMA.(XILINX, s.d.)

Com a configuração de registradores utilizando o protocolo de comunicação AXI *lite* pode-se gerar comandos através do bloco *control and Status* para que se possa permitir um fluxo de dados referentes à captura da câmera para um *buffer* temporário, para que, em seguida, utilizando o protocolo *AXI-Stream*, possa enviar estes dados diretamente para a memória do processador. O AXI VDMA também fornece um sinal de interrupção que sincronizará os *frames* enviados.(XILINX, s.d.)

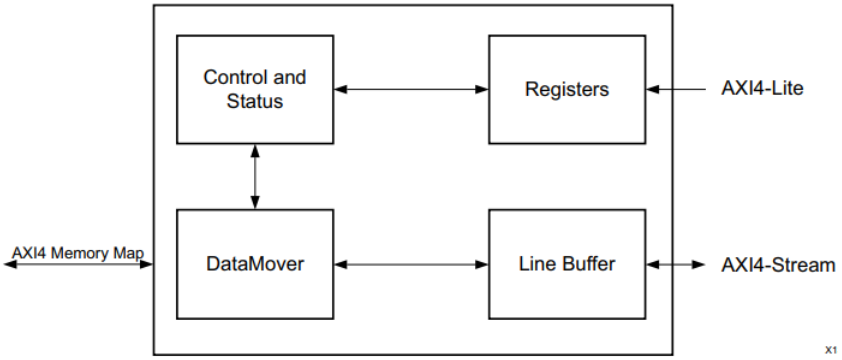


Figura 31 – Diagrama de Blocos referente ao AXI VDMA.(XILINX, s.d.)

A criação de *buffers* utilizando o AXI VDMA é interessante para passa *frames* entre dois domínios de *clock* distintos, como é o caso deste projeto em que há um clock de captura e um de processamento. O VDMA evita o cisalhamento da imagem transmitida. O cisalhamento ocorre quando um *frame* é lido da memória, mas este dado apresenta dados de duais imagens distintas, pois o tempo de leitura não foi suficiente para pegar o dado antes da escrita do próximo dado. Isto ocorre quando os sinais assíncronos de escrita e leitura se cruzam. Com a implementação do VDMA busca-se sincronizar estes sinais para que não haja crujamento e consequentemente o *frame* não apresente cisalhamento.

Durante os primeiros teste utilizando a arquitetura de referência apresentada em 3.3 verificou-se o cisalhamento de algumas imagens capturada, além de perda de dados, está é uma das razões para a implementação do VDMA, além de eliminar o gargalo causado pela

leitura de dados. A implementação desta nova arquitetura incluindo o VDMA ocupou os recursos mostrados na tabela 8, repare que o uso de BRAM se reduziu, pois os *frames* não são mais armazenados neste local.

Recurso	Utilizado	Disponível	% utilizado
LUT	5999	14400	41.66
LUTRAM	625	6000	10.42
FF	8005	28800	27.80
IO	28	54	51.85
BRAM	3.5	50	7

Tabela 8 – Recursos utilizados ao implementar o sistema na Minized usando *AXI VDMA*.

Realizando novos teste de temporização observa-se que o tempo de leitura se reduziu, alcançando-se o objetivo proposto. O tempo de configuração da câmera esta maior, pois a configuração dos registradores da câmera são passada pelo processador através protocolo AXI-*lite*, o tempo desta configuração não apresenta problemas, pois ocorre somente uma vez antes do início das capturas de imagem. Os resultados podem ser verificados na tabela 9.

	Configuração da Câmera	Leitura dos *pixels*	Processamento	Distância
Tempo(s)	2.000021	0.000475	N	N

Tabela 9 – Tempo para a execução de cada tarefa da plataforma considerando o VDMA

O uso do VDMA melhorou o sistema de captura de dados, entretanto apresentou-se dificuldade de tratar as interrupções provocadas a cada escrita de um novo *frame* na memória. Fazendo com que, em alguns casos, o processamento do dado não fosse completo. Assim buscou-se outras possíveis maneiras de melhorar o sistema.

3.4.2 Segmentação por Cores e centro geométrico

Outra maneira proposta para eliminar o gargalo de acesso à memória é a implementação do bloco de segmentação e cálculo do centro geométrico em FPGA, assim o processador só acessaria a memória uma vez para ler os dados de centro geométrico e área do alvo, reduzindo o tempo de leitura da memória. Outro fator de melhora seria o tempo de processamento de imagem, pois este bloco funcionaria com um *clock* mais alto já que deverá estar sincronizado com a captura dos *pixels* pela câmera.

Este bloco *Hardware* receberá diretamente o valor do *pixel* do bloco de transformação do espaço de cores, fazendo o processamento a cada subida de *clock* do sistema. Sua implementação não é muito extensa, cabendo no espaço em FPGA disponível na ZYBO. Os recursos consumidos por este bloco podem ser verificados na tabela 10.

Recurso	Utilizado	Disponível	% utilizado
LUT	622	17600	3.53
FF	98	35200	0.28

Tabela 10 – Recursos utilizados para implementar o sistema de segmentação e cálculo do centro geométrico na ZYBO.

Infelizmente este bloco não pode ser testado de maneira embarcada na ZYBO, integrada com a câmera, pois a sincronização entre os dados não foi corrigida em tempo hábil. Para verificar o funcionamento do sistema foi realizado simulações considerando um banco de dados que apresenta um valor de área e centro geométrico conhecido para que se possa fazer uma análise. O banco de dados é referente a uma imagem teste de dimensões 10x14, em que foi definido dois valores: 255 para a área a ser segmentada e 0 para a região de fundo. O exemplo utilizado pode ser verificado na figura 32.

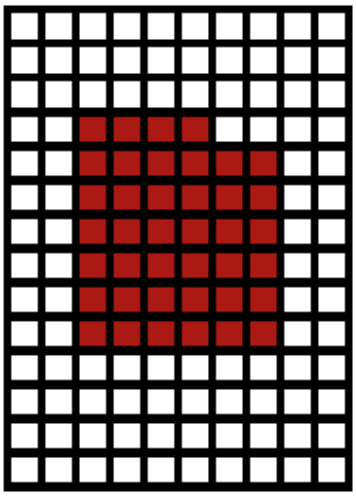


Figura 32 – Imagem utilizada para testes de segmentação em *hardware*

Na figura 33 verifica-se o resultado obtido da simulação funcional após o bloco receber o último dado. Percebe-se que o resultado da posição do centro geométrico é disponibilizado depois de $Tam_{img} \times clock$, como esperado, além de que o valor da área é disponibilizado antes do resultado do centro geométrico. Também verifica-se que os contadores de posição funcionam corretamente. Os valores esperados eram: $A = 40$, $C_i = 7.15$ e $C_j = 5.4$.

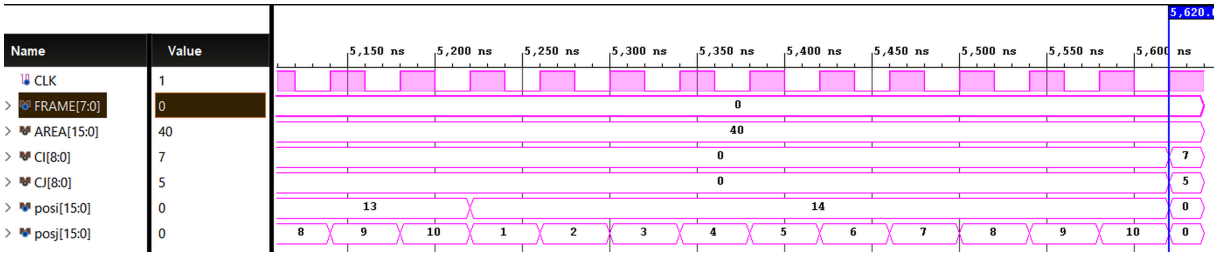


Figura 33 – Simulação do bloco de Segmentação de imagem e cálculo do centro geométrico em *hardware*.

4 Conclusão

Com o desenvolvimento deste trabalho foi implementado uma plataforma para o desenvolvimento de testes envolvendo o conceito de *platooning*. Com isso alcança-se o objetivo proposto, apresentando estudos de caso simples para exemplificar as funcionalidades do sistema.

A princípio formou-se um pelotão composto por dois veículos que possam seguir uma trajetória em linha reta. As análises a serem feitas com a malha fechada não foram possíveis devido à problemas encontrados nas baterias dos veículos, por este motivo foram feitas análises da resposta do sistema com os aceleradores de hardware e sem estas estruturas considerando somente os sinais dados pela arquitetura como resposta. Ao substituir as baterias pretende-se verificar o comportamento dos veículos. Assim que o sistema se comportar como o esperado será possível aumentar o número de veículos que compõe o pelotão, podendo verificar outras formas de organização, não somente em fila, e testar situações adversas.

4.0.1 Perspectivas Futuras

A conclusão deste projeto apresenta uma plataforma base para o desenvolvimento de novas aplicações envolvendo o conceito de *platooning*, abrindo possibilidades para o desenvolvimento de sistemas de rastreamento de um alvo através do processamento de imagens implementadas em *hardware* reconfigurável. Entre os trabalhos futuros a serem desenvolvidos nesta plataforma pode-se citar a criação de modelos de controladores para os veículos , inserindo mais sensores para ter dados mais precisos de velocidade.

Outros projetos a serem desenvolvidos é em relação a melhoria do sistema de rastreamento de alvo, como foi apresentado apenas casos simples, não foi implementados maneiras para melhoria da imagens capturada, além de uma calibração adequada da câmera para eliminar distorções.

Também é possível desenvolver um sistema de processamento em tempo real para administrar melhor as execuções das tarefas. Isto é necessário devido a natureza temporal existente nas tarefas implementadas para *platooning*. É interessante a ideia de um sistema que atenda as condições temporais para que não haja uma colisão entre os veículos devido a perdas de *deadlines*. A plataforma desenvolvida apresenta muitas possibilidades de projetos para o melhoramento do tema estudado.

Referências

ALMEIDA, D. Implementing and Tuning an Autonomous Racing Car Testbed. In: MASTERS Thesis CISTER-TR-191209. Porto, Portugal, 2019. Citado nas pp. 15, 16.

ASSOCIAÇÃO NACIONAL DE TRANSPORTES PÚBLICOS. **Sistemas Inteligentes de Transportes**. Disponível em: <http://files-server.antp.org.br/_5dotSystem/download/dcmDocument/2013/03/18/9AB9A3EB-97DC-4711-9751-162AD361D7F0.pdf>. Citado na p. 15.

AXELSSON, J. Safety in Vehicle Platooning: A Systematic Literature Review. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 5, p. 1033–1045, 2017. DOI: 10.1109/TITS.2016.2598873. Citado na p. 28.

BHOOPALAM, A. K.; AGATZ, N.; ZUIDWIJK, R. Planning of truck platoons: A literature review and directions for future research. **Transportation research part B: methodological**, Elsevier, v. 107, p. 212–228, 2018. Citado na p. 28.

BITTENCOURT, A. P. C. Controle de sistema de dois graus de liberdade com realimentação visual por meio de segmentação por cores em plataforma reconfigurável, 2013. Citado nas pp. 29–32.

BONDALAPATI, K.; PRASANNA, V. Reconfigurable computing systems. **Proceedings of the IEEE**, v. 90, n. 7, p. 1201–1217, 2002. DOI: 10.1109/JPROC.2002.801446. Citado nas pp. 25, 26.

BORENSTEIN, J.; KOREN, Y. The vector field histogram-fast obstacle avoidance for mobile robots. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 278–288, jun. 1991. ISSN 2374-958X. DOI: 10.1109/70.88137. Citado na p. 23.

BRAGA, N. C. **Manual de mecatrônica**. Editora Newton C. Braga, 2017. Citado nas pp. 48, 49.

CROCKETT, L.; ELLIOT, R.; ENDERWITZ, M.; STEWART, R. **The Zynq Book: Embedded Processing Withe ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC**. Strathclyde Academic Media, 2014. ISBN 9780992978709. Citado nas pp. 34–36.

DA SILVA, B. A.; LIMA, A. M.; YUDI, J. A manycore vision processor architecture for embedded applications. In: 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC). 2020. P. 1–8. DOI: 10.1109/SBESC51047.2020.9277867. Citado nas pp. 46, 47.

FERREIRA, C. S. **Implementação de algoritmo de subtração de fundo para detecção de objetos em movimento, usando sistemas reconfiguráveis**. 2012. Tese (Doutorado) – Departamento de Engenharia Mecânica, Universidade de Brasília. Dissertação de Mestrado em Sistemas Mecatrônicos. Citado nas pp. 29, 31.

GONG, S.; DU, L. Cooperative platoon control for a mixed traffic flow including human drive vehicles and connected and autonomous vehicles. **Transportation Research Part B: Methodological**, v. 116, p. 25–61, 2018. ISSN 0191-2615. DOI: https://doi.org/10.1016/j.trb.2018.07.005. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0191261517311827>. Citado na p. 16.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. Editora Blucher, 2000. Citado na p. 31.

GUEDES, N. A Robotic Platooning Testbed for Cooperative ITS Components. In: MASTERS Thesis CISTER-TR-191208. Porto, Portugal, 2019. Citado nas pp. 16–20.

HOLANDÊS, G. 2022. government of the Netherlands, Keywords = truck platooning, Title = European Truck Platooning Challenge 2016, Url = https://www.government.nl/documents/leaflets/2015/10/06/leaflet-european-truck-platooning-challenge-2016. Acesso em: 1 mai. 2022. Citado na p. 20.

HUANG, K.; WANG, L.; TAN, T.; MAYBANK, S. A real-time object detecting and tracking system for outdoor night surveillance. **Pattern Recognition**, v. 41, n. 1, p. 432–444, 2008. ISSN 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.05.017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0031320307002579>. Citado na p. 29.

INFINITO, H. 2020. HU infinito, Keywords = cam, Title = Módulo Câmera VGA OV7670, Url = https://www.huinfinito.com.br/modulos/957-modulo-camera-vga-ov7670.html. Acesso em: 13 dez. 2020. Citado na p. 25.

INFINITO, H. HU infinito, Keywords = chassi, Title = KIT Chassi (plataforma) para Robô 4WD - Acrílico, Url = https://www.huinfinito.com.br/chassis-plataformas/872-kit-chassi-plataforma-para-robo-4wd-acrilico.html, urldate = 2020-12-13, Year = 2020, citado na p. 26.

INFINITO, H. HU infinito, Keywords = ponte H, Title = Módulo de ponte H L298N, Url = https://www.huinfinito.com.br/controladores/583-modulo-driver-motor-com-dupla-ponteh-st-l298n.html, urldate = 2022-05-01, Year = 2022, citado na p. 48.

KOREA, M. of. 2022. Ministry of Land, Infrastructure and Transport, Republic of Korea, Keywords = ITS, Title = Concept of C-ITS (Cooperative-Intelligent Transport Systems), Url = https://www.c-its.kr/english/introduction.do. Acesso em: 23 fev. 2022. Citado na p. 17.

LARSSON, E.; SENNTON, G.; LARSON, J. The vehicle platooning problem: Computational complexity and heuristics. **Transportation Research Part C: Emerging Technologies**, v. 60, p. 258–277, 2015. ISSN 0968-090X. DOI: https://doi.org/10.1016/j.trc.2015.08.019. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0968090X15003204>. Citado na p. 16.

LIANG, K.; MÅRTENSSON, J.; JOHANSSON, K. H. Heavy-Duty Vehicle Platoon Formation for Fuel Efficiency. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 4, p. 1051–1061, 2016. DOI: 10.1109/TITS.2015.2492243. Citado na p. 16.

MANDOW, A.; MARTÍNEZ, J.; MORALES, J.; BLANCO, J. L.; GARCIA, A.; GONZÁLEZ-JIMÉNEZ, J. Experimental kinematics for wheeled skid-steer mobile robots. In: p. 1222–1227. DOI: 10.1109/IROS.2007.4399139. Citado nas pp. 38–40.

MONTANARI, R. **Detecção e classificação de objetos em imagens para rastreamento de veıculos**. 2016. Tese (Doutorado) – Universidade de São Paulo. Citado na p. 29.

OLIVEIRA, T. R. Certificados Sociais para Segurança em Redes Veiculares Tolerantes a Interrupções. **31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2013**, p. 835–848, 2013. Citado na p. 18.

OMNIVISION. Omnivision, Keywords = camera, Title = OV7670/OV7171 CMOS VGA (640x480) CAMERACHIP with OmniPixel® Technology, Url = https://www.voti.nl/docs/OV7670.pdf, urldate = 2020-12-13, Year = 2020, citado nas pp. 43–46.

OMS. **Global Status Report on Road Safety 2018**. World Health Organization, 2019. (Nonserial Publication). ISBN 9789241565684. Disponível em: <https://books.google.com.br/books?id=uHOyDwAAQBAJ>. Citado na p. 15.

SÁNCHEZ-FERREIRA, C.; MORI, J. Y.; LLANOS, C. H.; FORTALEZA, E. Development of a stereo vision measurement architecture for an underwater robot. In: IEEE. 2013 IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS). 2013. P. 1–4. Citado na p. 29.

SCOTT HAUCK, A. D. **Reconfigurable computing: the theory and practice of FPGA-based computation**. Elsevier, 2008. (Nonserial Publication). ISBN 978-0-12-370522-8. Citado na p. 26.

SILVA, J. M. M. F. da. **Enabling Cooperative Vehicle Platooning in a Robotic Testbed.** 2019/2020. Monografia (Licenciatura em Engenharia Eletrotécnica e de Computadores), isep (Instituto Superior de Engenharia do Porto), Porto, Portugal. Citado nas pp. 15, 16.

SINGH BHUPENDRA, A. G. Recent trends in intelligent transportation systems: a review. **Journal of Transport Literature**, p. 30–34, 2015. DOI: http://dx.doi.org/10.1590/2238-1031.jtl.v9n2a6. Citado na p. 17.

SOUSA NUNES, D. F. de. **Uso de comunicação V2V para o descarregamento de dados em redes celulares: uma estratégia baseada em clusterização geográca para apoiar o sensoriamento veicular colaborativo**. 2018. Tese (Doutorado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo. Tese de Doutorado em Ciências de Computação e Matemática Computacional. DOI: 10.11606/T.55.2019.tde-29032019-110813. Citado na p. 18.

SUSLOV, A. 2022. techcrunch, Keywords = Lidar, Title = Startups look beyond lidar for autonomous vehicle perception, Url = https://techcrunch.com/2021/01/16/startups-look-beyond-lidar-for-autonomous-vehicle-perception/. Acesso em: 23 fev. 2022. Citado na p. 24.

VELODYN. 2022. Velodyn, Keywords = Lidar, Title = Laser LIDAR para detecção de obstáculos HDL-64E, Url = https://velodynelidar.com/. Acesso em: 23 fev. 2022. Citado na p. 24.

WEI, D. C. M. **Método de desvio de obstáculos aplicado em veículo autônomo.** 2015. Diss. (Mestrado) – Escola Politécnica, Universidade de São Paulo. Mestrado em Engenharia de Transportes. DOI: 10.11606/D.3.2016.tde-17062016-142254. Citado na p. 22.

XILINX. xilinx, Keywords = xilinx vdma, Title = AXI Video Direct Memory Access v6.2, Url = $https://docs.xilinx.com/v/u/6.2-English/pg020\_axi\_vdma$, urldate = 2022-05-01, Year = 2022, citado na p. 56.

Apêndices

APÊNDICE A – Códigos de programação

A.1 Descrição de *Hardware* para o funcionamento do PWM

Código A.1 – Módulo para determinar valor de PWM.

```
1  `timescale 1ns / 1ps
2
3  //////////////////////////////////////////////////////////////////
4  // Company: UNB
5  // Aluno: Douglas Lustosa da Silva
6  //
7  // Create Date: 10.03.2022 17:31:08
8  // Module Name: pwm_control
9  // Project Name: pwm_control
10 // Target Devices:
11 // Tool Versions: Vivado 2019.1
12 // Description: Módulo  que determina um sinal PWM
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////
21
22 module pwm_control
23     #(parameter R = 8, TIMER_BITS = 15)(
24     input clk,
25     input reset_n,
26     input [R:0] duty, // Control the Duty Cylce
27     input [TIMER_BITS - 1:0] FINAL_VALUE, // Control the switching
          frequency
28     output pwm_out
29     );
30
31
32
33     reg [R - 1:0] Q_reg, Q_next;
34     reg d_reg, d_next;
35     wire tick;
36
```

```
37      // Up Counter
38      always @(posedge clk, negedge reset_n)
39      begin
40          if (~reset_n)
41          begin
42              Q_reg <= 'b0;
43              d_reg <= 1'b0;
44          end
45          else if (tick)
46          begin
47              Q_reg <= Q_next;
48              d_reg <= d_next;
49          end
50          else
51          begin
52              Q_reg <= Q_reg;
53              d_reg <= d_reg;
54          end
55      end
56
57      // Next state logic
58      always @(Q_reg, duty)
59      begin
60          Q_next = Q_reg + 1;
61          d_next = (Q_reg < duty);
62      end
63
64      assign pwm_out = d_reg;
65
66      // Prescalar Timer
67
68
69      temporizador #(.BITS(TIMER_BITS)) timer0 (
70          .clk(clk),
71          .reset_n(reset_n),
72          .enable(1'b1),
73          .FINAL_VALUE(FINAL_VALUE),
74          .done(tick)
75      );
76
77
78  endmodule
```

Código A.2 – Módulo Temporizador para determinar a frequência de saída do PWM.

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////
3  // Company: UNB
4  // Aluno: Douglas Lustosa da Silva
5  //
6  // Create Date: 10.03.2022 19:40:11
7  // Module Name: temporizador
```

```
8  // Project Name: temporizador
9  // Target Devices:
10 // Tool Versions: Vivado 2019.1
11 // Description: Módulo temporizador para controlar frequencia
12 //
13 // Dependencies:
14 //
15 // Revision:
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 //
19 //////////////////////////////////////////////////////////////////////////////////
20
21
22 module temporizador
23     #(parameter BITS = 4)(
24     input clk,
25     input reset_n,
26     input enable,
27     input [BITS - 1:0] FINAL_VALUE,
28 //    output [BITS - 1:0] Q,
29     output done
30     );
31
32     reg [BITS - 1:0] Q_reg, Q_next;
33
34     always @(posedge clk, negedge reset_n)
35     begin
36         if (~reset_n)
37             Q_reg <= 'b0;
38         else if(enable)
39             Q_reg <= Q_next;
40         else
41             Q_reg <= Q_reg;
42     end
43
44     // Next state logic
45     assign done = Q_reg == FINAL_VALUE;
46
47     always @(*)
48         Q_next = done? 'b0: Q_reg + 1;
49
50
51 endmodule
```

A.2 Programas de Teste para captura de imagem via UART.

Código A.3 – Leitura de imagem via UART

```
import serial


port = 'COM4'
bauderate = 115200
com_serial = serial.Serial(port, bauderate)


rgb =[]
arq = open('imagem.txt', 'w')
cont = 0
while(cont < 76800):
    data = com_serial.readline()
    rgb.append(data)
    #if(data == EOFError):
    #    break
    #data_str = str(data)
    #d = data_str.rstrip('\n')
    #arq.write(d)

    #print(d)
    cont = cont + 1
    print(len(rgb))
i = 0
while(i < len(rgb)):
    data_str = rgb[i].decode('utf-8')
    data_str = data_str.rstrip('\n')
    arq.write(data_str)
    i = i + 1
#print(len(arq))
arq.close()
com_serial.close()
```

Código A.4 – Código para construir imagem lida da entrada UART

```
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 25 13:17:16 2022

@author: dougl
"""

import numpy as np
import matplotlib.pyplot as plt


arq = open('imagem.txt', 'r')
d = []
#a = np.zeros(shape=(320,240))
a_new = np.zeros(shape=(240,320))
```

```
#print(a)
#novo_dado = open('data2.txt', 'w')
for linha in arq:
    linha = linha.rstrip()
    num = int(linha)
    d.append(num)


k = 0
i = 0
while(i < len(a_new)):
    j = 0
    while(j < len(a_new[0])):
        if(k < len(d)):
            a_new[i][j]= d[k]
            k = k+1
        else:
            a_new[i][j] = 0
        j = j+1
    i = i+1
#print(len(a))
#print(a)
print(a_new.shape)

plt.imshow(a_new)
plt.show()

arq.close()


'''
import numpy as np
import matplotlib.pyplot as plt

arq = open('dadosGrayMINIZED.txt', 'r')
d = []
a = np.zeros(shape=(320,240))
print(a)
#novo_dado = open('data2.txt', 'w')
for linha in arq:
    linha = linha.rstrip()
    num = int(linha)
    d.append(num)


k = 0
i = 0
while(i < len(a)):
    j = 0
    while(j < len(a[0])):
        if(k < len(d)):
            a[i][j]= d[k]
```

```
            k = k+1
        else:
            a[i][j] = 0
        j = j+1
    i = i+1
print(len(a))
print(a)
print(a.shape)

plt.imshow(a)
plt.show()

arq.close()
'''
```

Código A.5 – Código Teste para segmentação de imagem lida via UART

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 31 12:36:04 2022

@author: dougl
"""

import numpy as np
import matplotlib.pyplot as plt

k = 0
posi = 0
posj = 0
Ci = 0
Cj = 0
area = 0
sumi = 0
sumj = 0

arq = open('imagem.txt', 'r')
d = []
novo = []
masc = np.zeros(shape=(240,320))

for linha in arq:
    linha = linha.rstrip()
    num = int(linha)
    d.append(num)


while(k < 76800):
    if(posj < len(masc)):
        posj = posj +1
    else:
        posj = 1
```

```
36          if(posi < len(masc[0])):
37              posi = posi+1
38
39          else:
40              print("PASSEI")
41              posi = 1
42              Ci = sumi / area
43              Cj = sumj / area
44              area = 0
45              sumi = 0
46              sumj = 0
47
48      if (d[k] >= 64 and d[k] <= 112):
49          area = area+1
50          sumi = sumi + posi
51          sumj = sumj + posj
52          novo.append(255)
53      else:
54          novo.append(0)
55
56      k= k+1
57
58  arq.close()
59
60  k = 0
61  i = 0
62  while(i < len(masc)):
63      j = 0
64      while(j < len(masc[0])):
65          if(k < len(d)):
66              masc[i][j]= novo[k]
67              k = k+1
68          else:
69              masc[i][j] = 0
70          j = j+1
71      i = i+1
72
73
74  Ci = int(sumi / area)
75  Cj = int(sumj / area)
76
77
78  print("Ci: ", Ci)
79  print("Cj: ", Cj)
80  print("area: ", area)
81  print("Sumi: ", sumi)
82  print("Sumj: ", sumj)
83
84  plt.imshow(masc)
85  plt.show()
```

A.3 Códigos implementados no PS via SDK para comunicação com o PL, leitura de *pixels* e segmentação da arquitetura de referência

Código A.6 – Código *header* para configuração da câmera

```
/*
 * camera.h
 *
 *  Created on: Jan 18, 2021
 *      Author: arthu
 */

#ifndef SRC_CAMERA_H_
#define SRC_CAMERA_H_

#include "sleep.h"
#include "xparameters.h"



#define IPSTOP        XPAR_COMANDOSTOP_V1_0_0_BASEADDR
#define IPVSYNC      (XPAR_COMANDOSTOP_V1_0_0_BASEADDR + 0x4)
#define IPVDEBOUNCE   (XPAR_COMANDOSTOP_V1_0_0_BASEADDR + 0x4 + 0x4)
#define IPCONFEND    (XPAR_COMANDOSTOP_V1_0_0_BASEADDR + 0x4 + 0x4 +0x4)



void config_cam() {
  Xil_Out32(IPSTOP, 0);
  Xil_Out32(IPVDEBOUNCE, 1);
  while (Xil_In32(IPCONFEND) != 1) {

    asm("nop");
    usleep(1);
  }

  Xil_Out32(IPVDEBOUNCE, 0);
  //sleep(0.3);
}


void pausar_imagem() {
  Xil_Out32(IPSTOP, 1);
  while (Xil_In32(IPVSYNC) != 1) {

    asm("nop");
    usleep(1);
```

```
43    }
44  }
45
46  void continua_imagem() {
47    while (Xil_In32(IPVSYNC) != 1) {
48
49        asm("nop");
50        usleep(1);
51      }
52    Xil_Out32(IPSTOP, 0);
53
54
55  }
56
57  #endif /* SRC_CAMERA_H_ */
```

Código A.7 – Código *header* para realizar a leitura dos *pixels* da BRAM

```
1   /*
2    * visualize.h
3    *
4    *  Created on: 15 de jul de 2020
5    *      Author: Bruno Almeida
6    */
7
8   #ifndef SRC_VISUALIZE_H_
9   #define SRC_VISUALIZE_H_
10
11  #include "xparameters.h"
12  #include "xil_types.h"
13  //#include "noc.h"
14
15  #define TIMG_WIDTH  320
16  #define TIMG_HEIGTH 240
17
18  enum {
19    VIS_OK,
20    VIS_ERROR
21  };
22
23  #define VIS_BASE_ADDRESS  XPAR_IMAGE_VISUALIZATION_0_BASEADDR
24
25  typedef struct
26  {
27    volatile u32 GPR;
28    volatile u32 ADR;
29    volatile u32 DTI;
30    volatile u32 DTO;
31  } Vis_AxiStruct;
32
33  #define VIS_AXI       ((Vis_AxiStruct *)VIS_BASE_ADDRESS)
34
```

```
35 /* General Purpose Register - Control Bits */
36 #define VIS_AXI_EN      ((u32 )0x00000001)
37 #define VIS_AXI_WEN     ((u32 )0x00000002)
38
39 /* General Purpose Register - Status Bits */
40 #define VIS_AXI_OK      ((u32 )0x00000100)
41
42 /* Address Register */
43 #define VIS_AXI_ADR     ((u32)0x00000001)
44 #define VIS_AXI_ADR_SHIFT 0UL
45 #define VIS_AXI_ADR_MASK  ((u32)0xFFFFFFFF)
46
47 /* Data In Register */
48 #define VIS_AXI_DTI     ((u32)0x00000001)
49 #define VIS_AXI_DTI_SHIFT 0UL
50 #define VIS_AXI_DTI_MASK  ((u32)0xFFFFFFFF)
51
52 /* Data Out Register */
53 #define VIS_AXI_DTO     ((u32)0x00000001)
54 #define VIS_AXI_DTO_SHIFT 0UL
55 #define VIS_AXI_DTO_MASK  ((u32)0xFFFFFFFF)
56
57 int Vis_readPixel(u32 x, u32 y, u32 *px)
58 {
59   u32 addr = x + TIMG_WIDTH * y;
60   VIS_AXI->ADR = (u32 )addr;
61   VIS_AXI->GPR |= (VIS_AXI_EN);
62   while(!(VIS_AXI->GPR & VIS_AXI_OK))
63     __asm("nop");
64
65   *px = VIS_AXI->DTO;
66   VIS_AXI->GPR &= ~(VIS_AXI_EN);
67   while((VIS_AXI->GPR & VIS_AXI_OK))
68     __asm("nop");
69
70   return VIS_OK;
71 }
72
73 int Vis_readImage(u8 image[TIMG_HEIGTH][TIMG_WIDTH])
74 {
75     u32 px;
76   for(u32 i=0; i < TIMG_WIDTH; i++)
77   {
78     for(u32 j=0; j < TIMG_HEIGTH; j++)
79     {
80       Vis_readPixel(i,j,&px);
81       image[j][i] = (u8 )px;
82     }
83   }
84
85   return VIS_OK;
86 }
```

```
87
88 #endif /* SRC_VISUALIZE_H_ */
```

Código A.8 – Código principal que realiza a segmentação e calcula a distância do alvo

```
1  /*
2   * helloworld.c: simple test application
3   *
4   * This application configures UART 16550 to baud rate 9600.
5   * PS7 UART (Zynq) is not initialized by this application, since
6   * bootrom/bsp configures it to baud rate 115200
7   *
8   * ------------------------------------------------
9   * | UART TYPE   BAUD RATE                        |
10  * ------------------------------------------------
11  *   uartns550   9600
12  *   uartlite    Configurable only in HW design
13  *   ps7_uart    115200 (configured by bootrom/bsp)
14  */
15
16
17
18
19 #include "platform.h"
20 #include "camera.h"
21 #include "xil_printf.h"
22 #include "sleep.h"
23 #include "xparameters.h"
24 #include "visualize.h"
25 #include <stdio.h>
26
27 u8 final_image[TIMG_HEIGTH][TIMG_WIDTH];
28
29
30 #include "platform.h"
31 #include "camera.h"
32 #include "xil_printf.h"
33 #include "sleep.h"
34 #include "xparameters.h"
35 #include "visualize.h"
36 #include <stdio.h>
37 #include "xtime_l.h"
38 #include <math.h>
39
40 u8 final_image[TIMG_HEIGTH][TIMG_WIDTH];
41 u8 img_negativa[TIMG_HEIGTH][TIMG_WIDTH];
42 int posX = 0;
43 int posY = 0;
44 int CX = 0;
45 int CY = 0;
46 int area = 0;
47 int sumX = 0;
```

```
48 int sumY = 0;
49
50
51 int main()
52 {
53   float cost_time, d;
54   XTime gbl_time_before_test;
55   XTime gbl_time_after_test;
56
57     init_platform();
58
59     // ativa o debounce ate a configuracao completar
60   //while(1){
61
62     //XTime_GetTime(&gbl_time_before_test);
63     config_cam();
64     //XTime_GetTime(&gbl_time_after_test);
65
66
67     //XTime_GetTime(&gbl_time_before_test);
68   // ativa o stop e pausa imagem  no vsync '1'
69   pausar_imagem();
70   //XTime_GetTime(&gbl_time_after_test);
71
72   //printf("Comecar imagem .............\n");
73
74   //XTime_GetTime(&gbl_time_before_test);
75   // Le a imagem para final_image
76   Vis_readImage(final_image);
77   //XTime_GetTime(&gbl_time_after_test);
78
79   //XTime_GetTime(&gbl_time_before_test);
80   // passa a imagem pela interface UART- leitura matlab
81   for (int i = 0; i < TIMG_HEIGTH; i++) {
82     for (int j = 0; j < TIMG_WIDTH; j++) {
83             if(posX < TIMG_WIDTH)
84             {
85                 posX++;
86             }
87             else
88             {
89                 posX = 1;
90                 if(posY < TIMG_HEIGTH)
91                 {
92                     posY++;
93                 }
94                 else
95                 {
96                     posY = 1;
97                     CX = sumX / area;
98                     CY = sumY / area;
99                     area = 0;
```

```
100                     sumX = 0;
101                     sumY = 0;
102                 }
103             }
104             if(final_image[i][j] > 39 && final_image[i][j] < 49)
105             {
106                 area++;
107                 sumX = sumX + posX;
108                 sumY = sumY + posY;
109 
110                 img_negativa[i][j] = 255;
111             }
112             else
113             {
114               img_negativa[i][j] = 0;
115             }
116 
117       printf("%d\n", (int)final_image[i][j]);
118 
119     }
120   }
121 
122   //XTime_GetTime(&gbl_time_before_test);
123   //d = (1.35513915e-15)*pow(34188,4) +
      (-8.96681924e-11)*pow(34188,3) +
      (2.02733373e-06)*pow(34188,2) + (-1.91848507e-02)*34188 + (
      9.27737364e+01);
124 
125   //XTime_GetTime(&gbl_time_after_test);
126   //continua_imagem();
127 
128     //printf("Terminou imagem ..............\n");
129     //scanf("%d", &cont);
130 /*
131   CX = sumX / area;
132   CY = sumY / area;
133 
134   cotg = 1/tan(0.2181662);
135 
136   d = (31.2*31.2*cotg)/2*area;
137 
138 
139 
140   printf("Cx: %d\n", CX);
141   printf("Cy: %d\n", CY);
142   printf("SumX: %d\n", sumX);
143   printf("SumY: %d\n", sumY);
144   printf("area: %d\n", area);
145   printf("distncia: %.2f\n", d);*/
146 
147   //}
148 
```

```
149   cost_time = (float) (gbl_time_after_test -
          gbl_time_before_test)/(COUNTS_PER_SECOND);
150   printf("Test time = %f secs\r\n", cost_time);
151   printf("d = %f \r\n", d);
152   //xil_printf(str);
153
154     cleanup_platform();
155     return 0;
156 }
```

A.4 Programa para o VDMA.

Código A.9 – Programa teste para configuração do vdma e segmentação

```
1  #include "xparameters.h"
2  #include  "camera.h"
3  #include "xscugic.h"
4  #include "sleep.h"
5  #include "xaxivdma.h"
6  #include <stdlib.h>
7  #include "xil_cache.h"
8  #include "platform.h"
9  #include "xil_io.h"
10 #include "xstatus.h"
11 #include "xil_exception.h"
12 #include "xtime_l.h"
13
14 #define SIZE_ARR 640*480
15 #define PWM_BASE_ADDRESS                0x43C00000
16 #define HSize 640
17 #define VSize 480
18 #define FrameSize HSize*VSize*4
19
20
21 static XScuGic Intc;
22
23
24 //static int SetupIntrSystem(XAxiVdma *AxiVdmaPtr, u16 ReadIntrId);
25 u32 Buffer[3*FrameSize];
26
27 XAxiVdma myVDMA;
28 XAxiVdma_Config *config;
29 XAxiVdma_DmaSetup WriteCfg;
30
31 u32 sta;
32 u32 *address_bram = (u32 *)INIT_CONFIG ;
33
34 int posX = 0;
35 int posY = 0;
36 int CX = 0;
```

```
37 int CY = 0;
38 int area = 0;
39 int sumX = 0;
40 int sumY = 0;
41
42 float cost_time, d;
43 XTime gbl_time_before_test;
44 XTime gbl_time_after_test;
45
46
47 /************************** Variable Definitions
   *****************************/
48
49 /*
50  * The following are declared globally so they are zeroed and so
       they are
51  * easily accessible from a debugger
52  */
53
54 /* LED brightness level is now global to make is visble to the
   ISR. */
55 volatile u32 contador = 0;
56 /* The Instance of the Interrupt Controller Driver */
57
58
59 /*****************************************************************
60  /* Call back function for read channel
61 *****************************************************************
62
63 static void WriteCallBack(void *CallbackRef, u32 Mask)
64 {
65   if(contador>20){
66   XAxiVdma_DmaStop(&myVDMA,XAXIVDMA_WRITE);
67   // passa a imagem pela interface UART- leitura matlab
68   // Nao deveria fazer isso dentro de introut, so para mostrar a
       imagem,
69
70   //XTime_GetTime(&gbl_time_before_test);
71   for (int i = 0; i < 480; i++) {
72     for (int j = 0; j < 640; j++) {
73       if(posX < 640)
74                   {
75                       posX++;
76                   }
77                   else
78                   {
79                       posX = 1;
80                       if(posY < 480)
81                       {
82                           posY++;
83                       }
84                       else
```

```
85                          {
86                              posY = 1;
87                              CX = sumX / area;
88                              CY = sumY / area;
89                              area = 0;
90                              sumX = 0;
91                              sumY = 0;
92                          }
93                      }
94                      if(Buffer[j+640*i] > 39 && Buffer[j+640*i] < 49)
95                      {
96                          area++;
97                          sumX = sumX + posX;
98                          sumY = sumY + posY;
99
100                         //img_negativa[i][j] = 255;
101                     }
102                     else
103                     {
104                       //img_negativa[i][j] = 0;
105                     }
106     //printf("%d\n", Buffer[j+640*i]);
107
108     }
109
110   }
111   //XTime_GetTime(&gbl_time_after_test);
112
113   XAxiVdma_DmaStart(&myVDMA,XAXIVDMA_WRITE);
114   }
115 contador++;
116 }
117
118
119 static void WriteErrorCallBack(void *CallbackRef, u32 Mask) //
      Essa eh a funcao de interrupcao
120 {
121   /* User can add his code in this call back function */
122   printf("Read Call back Error function is called\r\n");
123
124 }
125 static int SetupIntrSystem(XAxiVdma *AxiVdmaPtr, u16 WriteIntrId)
126 {
127   int Status;
128   XScuGic *IntcInstancePtr =&Intc;
129
130   /* Initialize the interrupt controller and connect the ISRs */
131   XScuGic_Config *IntcConfig;
132   IntcConfig = XScuGic_LookupConfig(XPAR_PS7_SCUGIC_0_DEVICE_ID);
133   Status =  XScuGic_CfgInitialize(IntcInstancePtr, IntcConfig,
        IntcConfig->CpuBaseAddress);
134   if(Status != XST_SUCCESS){
```

```
135     xil_printf("Interrupt controller initialization failed..");
136     return -1;
137   }
138
139   Status =
         XScuGic_Connect(IntcInstancePtr,WriteIntrId,(Xil_InterruptHandler)XAxiV
         *)AxiVdmaPtr);
140   if (Status != XST_SUCCESS) {
141     xil_printf("Failed read channel connect intc %d\r\n", Status);
142     return XST_FAILURE;
143   }
144
145   XScuGic_Enable(IntcInstancePtr,WriteIntrId);
146
147   Xil_ExceptionInit();
148   Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,(Xil_ExceptionHandler)XS
         *)IntcInstancePtr);
149   Xil_ExceptionEnable();
150
151   /* Register call-back functions
152    */
153   XAxiVdma_SetCallBack(AxiVdmaPtr, XAXIVDMA_HANDLER_GENERAL,
         WriteCallBack, (void *)AxiVdmaPtr, XAXIVDMA_WRITE);
154
155   XAxiVdma_SetCallBack(AxiVdmaPtr, XAXIVDMA_HANDLER_ERROR,
         WriteErrorCallBack, (void *)AxiVdmaPtr, XAXIVDMA_WRITE);
156
157   return XST_SUCCESS;
158 }
159
160 int SetupVdma(){
161   int status;
162     int Index;
163     u32 Addr;
164     u32 Addr2;
165
166
167     config = XAxiVdma_LookupConfig(XPAR_AXI_VDMA_0_DEVICE_ID);
168     status = XAxiVdma_CfgInitialize(&myVDMA, config,
          config->BaseAddress);
169       if(status != XST_SUCCESS){
170         xil_printf("DMA Initialization failed");
171       }
172       Xil_Out32(XPAR_AXI_VDMA_0_BASEADDR + 0x30, 0x8B);
173 //       printf("MaxNumFrames: %d\n",(u8)config->MaxFrameStoreNum);
    /* Ja ta em 3*/
174 //       status=XAxiVdma_SetFrmStore(&myVDMA,
    (u8)config->MaxFrameStoreNum, XAXIVDMA_WRITE);
175 //       if (status != XST_SUCCESS) {
176 //         xil_printf("Write channel config failed %d\r\n", status);
177 //         return status;
178 //       }
```

```
179
180       WriteCfg.Stride = HSize*4;
181       WriteCfg.HoriSizeInput = HSize*4;
182       WriteCfg.VertSizeInput = VSize;
183       WriteCfg.FrameDelay = 0;
184       WriteCfg.EnableCircularBuf = 1;
185       WriteCfg.EnableSync = 1;
186       WriteCfg.PointNum = 0;
187       WriteCfg.EnableFrameCounter = 0;
188       WriteCfg.FixedFrameStoreAddr = 0;
189
190       status = XAxiVdma_DmaConfig(&myVDMA, XAXIVDMA_WRITE,
             &WriteCfg);
191       if (status != XST_SUCCESS) {
192         xil_printf("Write channel config failed %d\r\n", status);
193         return status;
194       }
195 //        printf("mama mia pica\n");
196       Addr = (u32)&(Buffer[0]); /* Acho que Addr esta apontando
             para o endereo do primeiro elemento do Buffer*/
197
198
199
200
201
202     for(Index = 0; Index < myVDMA.MaxNumFrames; Index++) {
203
204       WriteCfg.FrameStoreStartAddr[Index] = Addr;
205     //  WriteCfg.FrameStoreStartAddr[Index] = &Buffer[Index][0];
206 /*      WriteCfg.FrameStoreStartAddr[Index+1] = Addr2; */
207 /*      printf("Addr value: %d\n", Addr); /*rodou 3x*/
208       Addr +=  FrameSize; /* Soma com a quantidade de pixels*/
209
210     }
211
212
213
214     /*proximo passo: determinar size of array do buffer*/
215     //  int Buffer_size = sizeof(Buffer) / sizeof(int);
216     //  printf("Buffer size: %d\n", Buffer_size); /*Buffer size
           acusou 1228800 bytes : 1 frame*/
217
218     status = XAxiVdma_DmaSetBufferAddr(&myVDMA,
           XAXIVDMA_WRITE,WriteCfg.FrameStoreStartAddr);
219     if (status != XST_SUCCESS) {
220       if(status==XST_DEVICE_BUSY) xil_printf(" XST_DEVICE_BUSY
             Read channel set buffer address failed %d\r\n", status);
221       if(status==XST_INVALID_PARAM) xil_printf(" XST_INVAID_PARAM
             Read channel set buffer address failed %d\r\n", status);
222       if(status==XST_DEVICE_NOT_FOUND) xil_printf("
             XST_DEVICE_NOT_FOUND Read channel set buffer address
             failed %d\r\n", status);
```

```
223       return XST_FAILURE;
224     }
225
226
227     XAxiVdma_IntrDisable(&myVDMA, XAXIVDMA_IXR_COMPLETION_MASK,
          XAXIVDMA_WRITE);
228
229     SetupIntrSystem(&myVDMA,
          XPAR_FABRIC_AXI_VDMA_0_S2MM_INTROUT_INTR);
230
231     Xil_DCacheFlush();
232
233     status = XAxiVdma_DmaStart(&myVDMA,XAXIVDMA_WRITE);
234     if (status != XST_SUCCESS) {
235       if(status == XST_VDMA_MISMATCH_ERROR)
236         xil_printf("DMA Mismatch Error\r\n");
237       return XST_FAILURE;
238     }
239 }
240
241
242
243
244
245
246
247 union Data {
248    u32 i;
249    float f;
250 };
251
252
253 int main()
254 {
255
256
257       init_platform();
258
259
260       //XTime_GetTime(&gbl_time_before_test);
261     //Xil_Out32(IPSAMPLEALLOW,0);
262     write_config_cam(); // escreve os valores destinados aos
          registradores da camera na BRAM
263
264
265   config_cam(); //ativa o debounce ate a configuracao completar
        (add 2 segundos para que acomode estabilidade)
266
267   //XTime_GetTime(&gbl_time_after_test);
268   //pausar_imagem(); // ativa o stop e pausa imagem  no vsync '1'
269
270   //XTime_GetTime(&gbl_time_before_test);
```

```
271   SetupVdma();
272   //XTime_GetTime(&gbl_time_after_test);
273
274   sleep(5);
275   XAxiVdma_IntrEnable(&myVDMA, XAXIVDMA_IXR_COMPLETION_MASK,
         XAXIVDMA_WRITE);
276
277
278   XTime_GetTime(&gbl_time_before_test);
279     d = (1.35513915e-15)*pow(34188,4) +
          (-8.96681924e-11)*pow(34188,3) +
          (2.02733373e-06)*pow(34188,2) + (-1.91848507e-02)*34188 + (
          9.27737364e+01);
280
281     XTime_GetTime(&gbl_time_after_test);
282
283   cost_time = (float) (gbl_time_after_test -
         gbl_time_before_test)/(COUNTS_PER_SECOND);
284   printf("Test time = %f secs\r\n", cost_time);
285
286
287   while(1)
288   {
289     sleep(1);
290     //printf("%d \n \r", contador);
291   }
292
293     cleanup_platform();
294     return 0;
295 }
```