Universidade Estadual de Campinas - UNICAMP

Faculdade de Tecnologia

Trabalho de Conclusão de Curso - Projeto

Desenvolvimento de um robô seguidor de linhas para percursos e desvio de objetos

Aluno: Guilherme Pellegrini Bertacini

Orientadora: Prof. Dra. Talia Simões dos Santos Ximenes

Co-Orientador: Rodrigo Luiz Ximenes

Limeira, 2023

Guilherme Pellegrini Bertacini

Desenvolvimento de um robô seguidor de linhas para percursos e desvio de objetos

Trabalho de Conclusão de Curso apresentado na Universidade Estadual de Campinas – UNICAMP, Campus Faculdade de Tecnologia como requisito para obtenção do título de Bacharel em Engenharia de Telecomunicações.

Orientadora: Prof.ª. Dra. Talía Simões dos Santos
Co-orientador: Rodrigo Luiz Ximenes

Limeira, 2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Mariana Xavier - CRB 8/9615

Bertacini, Guilherme Pellegrini, 1998-
B461d Desenvolvimento de um robô seguidor de linhas para percursos e desvio de objetos / Guilherme Pellegrini Bertacini. – Limeira, SP : [s.n.], 2023.

Orientador: Talía Simões dos Santos Ximenes.
Coorientador: Rodrigo Luiz Ximenes.
Trabalho de Conclusão de Curso (graduação) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Veículos autônomos. 2. Microcontroladores. 3. Direção de veículos a motor. I. Ximenes, Talía Simões dos Santos,1980-. II. Ximenes, Rodrigo Luiz,1981-. III. Universidade Estadual de Campinas. Faculdade de Tecnologia. IV. Título.

Informações adicionais, complementares

Título em outro idioma: Development of a line-following robot for routes and object avoidance
Palavras-chave em inglês:
Autonomous vehicles
Microcontrollers
Motor vehicle driving
Titulação: Engenheiro de Telecomunicações
Banca examinadora:
Talia Simões dos Santos Ximenes
José Carlos Magossi
Marcos Sergio Gonçalves
Data de entrega do trabalho definitivo: 14-12-2023

Sumário

1. Introdução.....5
2. Objetivos.....6
3. Metodologia 7
4. Resultados.....26
5. Conclusão.....28
6. Referências.....29

1. Introdução

Com o avanço da tecnologia, cada vez mais nos aproximamos da realidade de termos um trânsito composto por veículos autônomos, onde contaremos cada vez mais com mecanismos de controle de direção autônoma, com menos atuação humana. Atualmente, já temos alguns veículos com sistemas semi-autônomos trafegando pelas ruas, como por exemplo veículos da Tesla, os quais contam com a tecnologia de "*driver-assist*", englobando câmeras e sensores para manter um veículo na faixa, controle de frenagem e distância de outros veículos ou objetos, assistentes de estacionamento.

A tecnologia de veículos autônomos vem com a proposição de tornar a locomoção cada vez mais cômoda para os passageiros, uma vez que o "motorista" do veículo não precisa se preocupar com o percurso percorrido e com o tráfego ao seu redor, possibilitando que o mesmo possa executar outras tarefas enquanto vai ao seu destino, além de propor um trânsito mais seguro, tendo em vista que muitos acidentes acontecem por erro ou abuso humano como distração pelo uso de telefone enquanto dirige ou excesso de velocidade, reduzindo o número de acidentes [1].

Para compor o sistema controle de um veículo autônomo são utilizadas câmeras para monitorar o entorno dos veículos, sensores que possam monitorar distância de objetos, visando atender tarefas que atualmente são dos motoristas, tais como controle de velocidade, permanência em faixa, desvio de objetos que podem estar na via, controle da direção e trajeto a ser percorrido. Além disso, ideia é que os veículos sejam integrados com uma rede para compartilhamento de informações como análise de intensidade de tráfego, qualidade da via, possíveis incidentes e percalços a serem enfrentados, gerando uma necessidade de maior desenvolvimento da cyber segurança para garantir a minimização do roubo ou mau uso dos dados gerados pelos veículos.

2. Objetivos

O projeto tem como principal objetivo simular um sistema de controle de um veículo autônomo utilizando uma câmera de leitura linear para fazer o monitoramento dos percursos do veículo e eventuais objetos, sendo integrada a um microcontrolador para fazer o processamento dos dados captados e realizar a tomada de decisão sobre o controle da direção do protótipo.

Pelo fato de o chassi do carro seguir uma escala de 1:16, a outra proposta do projeto é seguir a mesma escala de proporção para a definição das medidas envolvidas no projeto como a largura da pista a ser percorrida pelo veículo e a altura e posicionamento da câmera visando desenvolver uma simulação próxima de situações reais enfrentadas por um veículo autônomo real.

3. Metodologia

Para a realização do projeto foi utilizado um kit da NXP disponibilizado para a competição *Freescale Cup*, composto por um microcontrolador Freedom KL25Z, uma câmera "*line*" *Freescale Cup LineScan Camera* composta por um sensor TAOS TSL1401CL de 128 pixels, estando tanto o microcontrolador quanto as câmeras integradas ao chassi do robô o qual já possui montado dois motores DC, rodas e um servomotor para realizar o controle da direção.

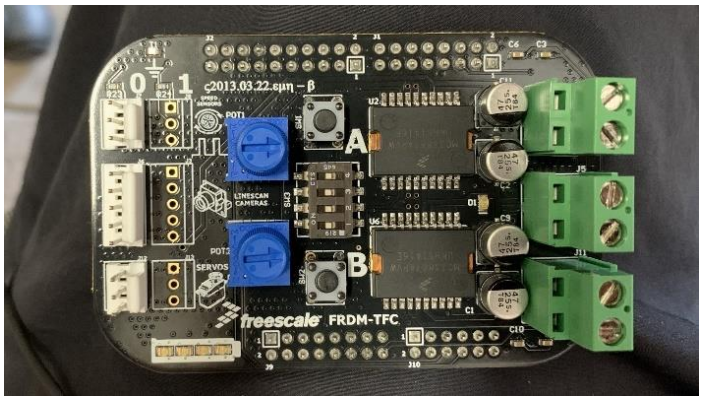
Acoplado com a placa microcontroladora FRDM KL25Z foi utilizada a Shield FRDM-TFC, sendo responsável por definir os pinos utilizados para o controle da câmera, servo motor e os motores.

Em termos de alimentação dos componentes, tanto a FRDM KL25Z quanto a câmera são alimentadas por 3,3V, enquanto os motores traseiros e o servo motor exigem uma tensão de 7,2V, assim sendo, é necessário o uso de uma bateria externa de 7,2V.

3.1 TFC Shield

Para realizar o controle e configuração das portas necessárias para o projeto, foi utilizado o Shield TFC, originário da competição Freescale Cup, tendo suporte para duas câmeras Line, dois motores DC, dois servos motores, dois sensores de velocidade, dois potenciômetros, dois botões *push* e quatro *switches*, além de entrada de alimentação de 7,2 V, sendo mostrado na Figura 1.

Figura 1 - Shield TFC



Fonte: Elaborado pelo autor

A Shield conta com suporte para duas câmeras *Line Scan* com os pinos de VDD, GND, SI, CLK e AOUT, dois servos motores com pinos de VDD, GND e PWM, dois potenciômetros, dois botões *push*, quadro chaves *switch*, ponte H para controle de sentido e velocidade dos motores traseiros, entradas para alimentação via bateria de 7,2V e LEDs sendo que cada um dos componentes é interligado com uma saída da placa FRDM KL25Z. A Tabela 1 mostra a relação dos pinos utilizados no projeto para cada componente.

Tabela 1 - Relação de pinos utilizados do Shield TFC com a FRDM KL25z

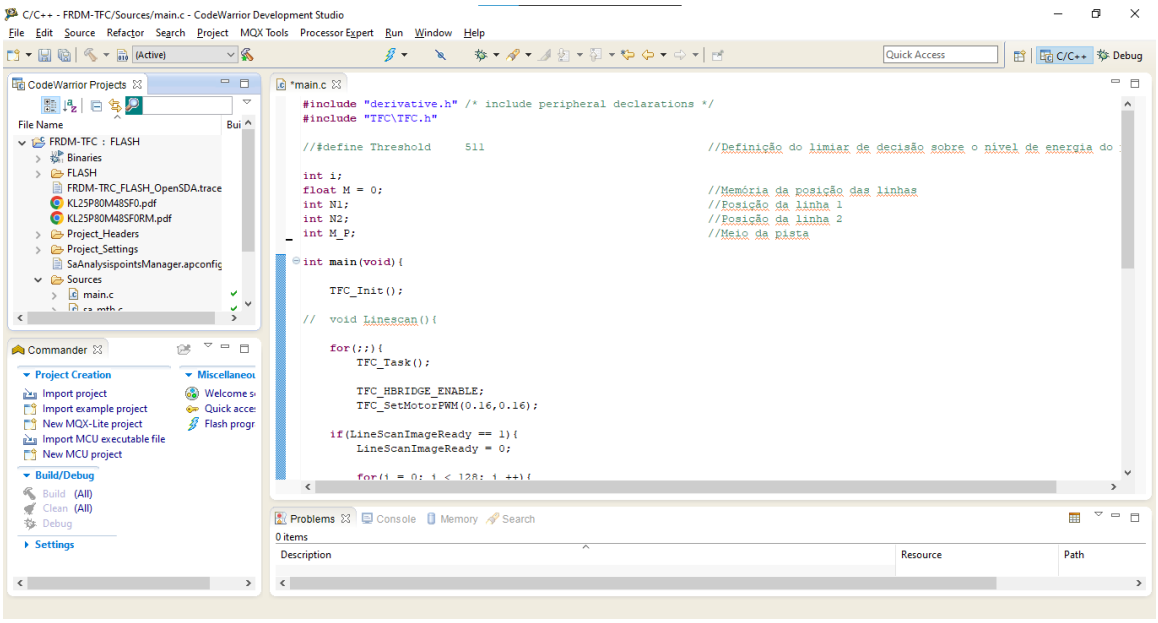
Relação de pinos do Shield TFC com a FRDM KL25z utilizados		
PWM Servo	Out 0	PTB0/FTM1_CHO
Motor A	In 1	PTC3/FTM0_CH2
	In 2	PTC4/FTM0_CH3
Motor B	In 1	PTC1/FTM0_CH0
	In 2	PTC2/FTM0_CH1
Câmera 0	AI	PTD5/ADC0_SE6b
	SI	PTD7
	CLK	PTE1

O controle e a programação do dispositivo podem ser feitos utilizando o Matlab Simulink Coder com apoio de uma biblioteca disponibilizada pelo fabricante

do dispositivo ou então via Code Warrior. O Code Warrior é uma IDE de desenvolvimento podendo ser desenvolvido código por uma linguagem de blocos, onde é possível selecionar e configurar os componentes tendo como resultado um código em linguagem C ou então ser desenvolvido o código em C diretamente, como o software Arduino IDE utilizado para programação do microcontrolador Arduino. Para utilizar o Matlab Simulink Coder é necessário ter uma licença específica e que não é suportada pela faculdade, assim foi escolhido o software Code Warrior como plataforma de desenvolvimento uma vez que eles disponibilizam licenças para estudantes de forma mais simples e fácil.

A Figura 2 mostra a interface do Code Warrior.

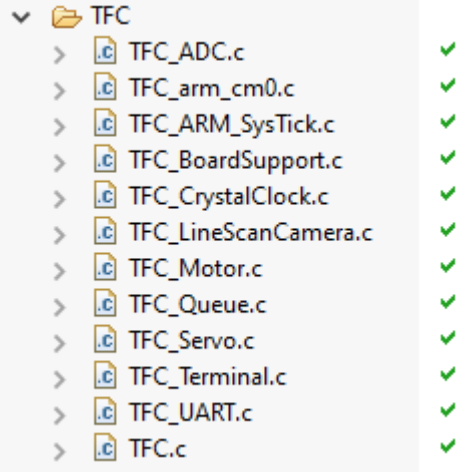
Figura 2 - Interface do Code Warrior



Fonte: Elaborado pelo autor

No caso para a aplicação do Shield TFC temos duas principais formas de trabalho sendo a primeira declarando e configurando os componentes, *buffers*, entradas e saída, multiplexadores, *clock* e demais componentes de forma manual. Já como segunda opção é utilizar a biblioteca “TFC.h” a qual já possui todas os componentes declarados e configurados prontos para o uso e aplicação da plataforma, disponibilizada no site da fabricante do kit [2] sendo ilustrada pela Figura 3.

Figura 3 - Componentes presentes na biblioteca TFC.h



Fonte: Elaborado pelo autor

Para a aplicação desse projeto optou-se por utilizar a biblioteca “TFC.h” visando otimização de tempo, sendo necessário o estudo da biblioteca e como a mesma poderia ser aplicada. No código principal “main.c” a biblioteca foi incluída via comando “#include TFC\TFC.H”, sendo que dentro da função principal era necessário chamar a função “TFC_Init()” para iniciar o usando dos periféricos da Shield, sendo o servo motor, a câmera, motores traseiros e ponte H.

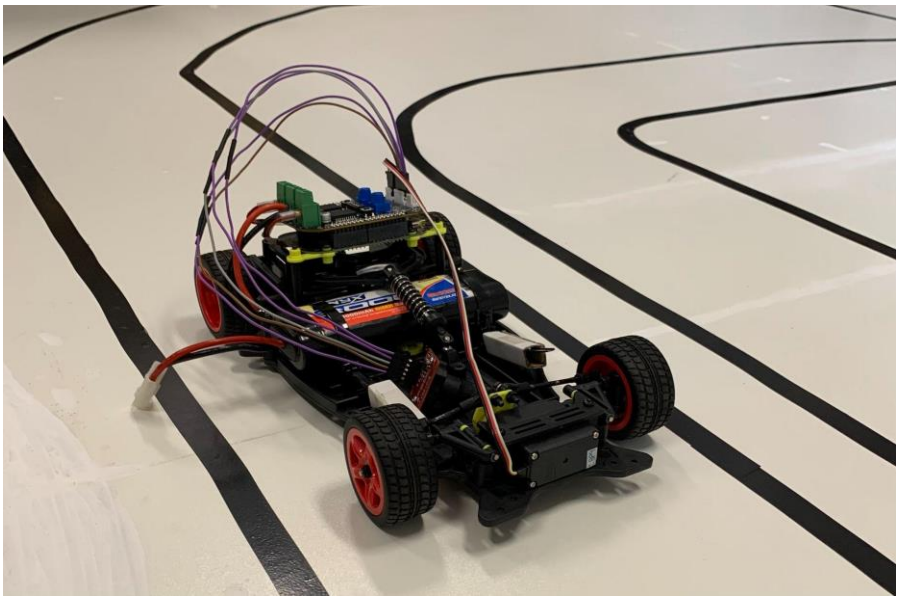
3.2 Chassi

As dimensões do chassi do carro são de 28,5 cm de comprimento por 16 cm de largura, logo, por ser uma escala de 1:16, tendo dimensões próximas a de um SUV médio como por exemplo, um Chevrolet Equinox que possui 4,652 m de comprimento por 2,105 m de largura contando com os espelhos laterais [3]. Como propósito de simular situações reais, as medidas acabam sendo importantes, tendo em vista que um dos nichos de mercado que mais crescem dentro do mercado automotivo é o de SUV’s médio, sendo que no primeiro semestre de 2023 cerca de 46% dos veículos vendidos no Brasil foram SUVs e segue uma tendência de que

nos próximos 5 anos essa margem pode crescer para 60% de acordo com dados da Fenabrave [4].

O chassi do carro já vem pré-montado com dois motores na parte traseira responsáveis pela tração sendo controlados por uma ponte H a qual controla a velocidade dos motores de forma independente simulando o diferencial de um carro, um suporte para bateria e suporte para as rodas dianteiras, conforme mostrado na Figura 4.

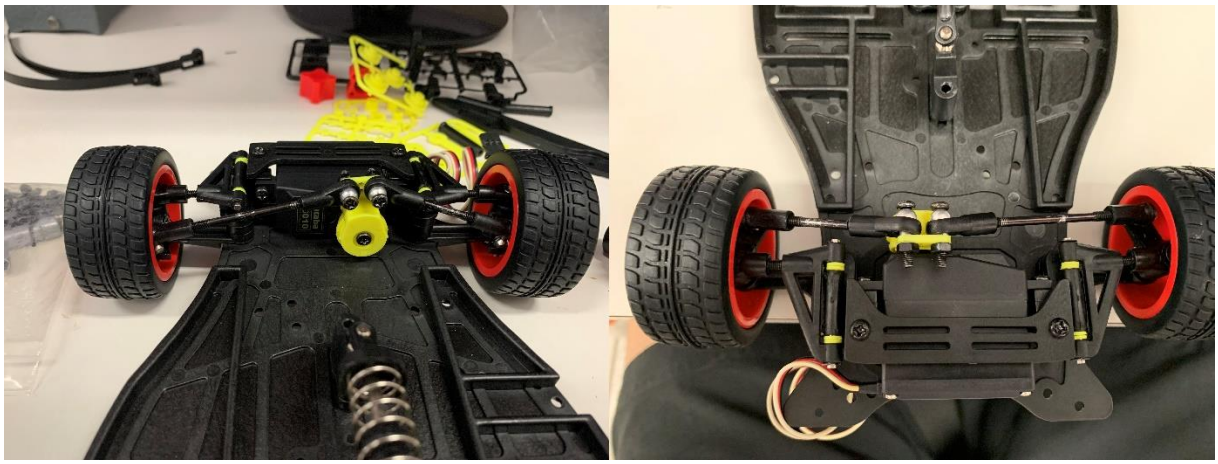
Figura 4 - Chassi Montado



Fonte: Elaborado pelo autor

Como complemento do chassi, foi montado o servo motor para fazer o controle da direção na posição, conforme mostrado na Figura 5, e um suporte para a haste da câmera sendo fixado de forma centralizada lateralmente ao chassi, o qual foi desenvolvido um projeto para impressão em 3D.

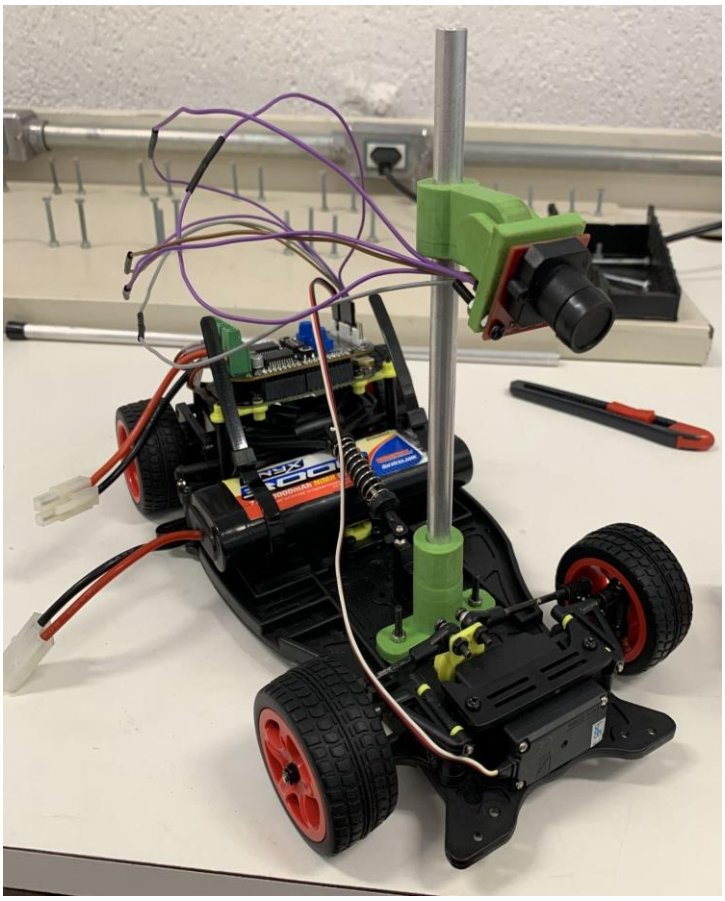
Figura 5 - Montagem do servo motor no chassi



Fonte: Elaborado pelo autor

Para posicionar a câmera também foi levado em consideração as dimensões de referência do Chevrolet Equinox, o qual tem uma altura declarada de 1,7 m [2]. Assim, a câmera foi posicionada a uma altura de cerca de 11 cm em relação ao solo, sendo mostrado na Figura 6.

Figura 6 - Suportes projetados da haste e da câmera



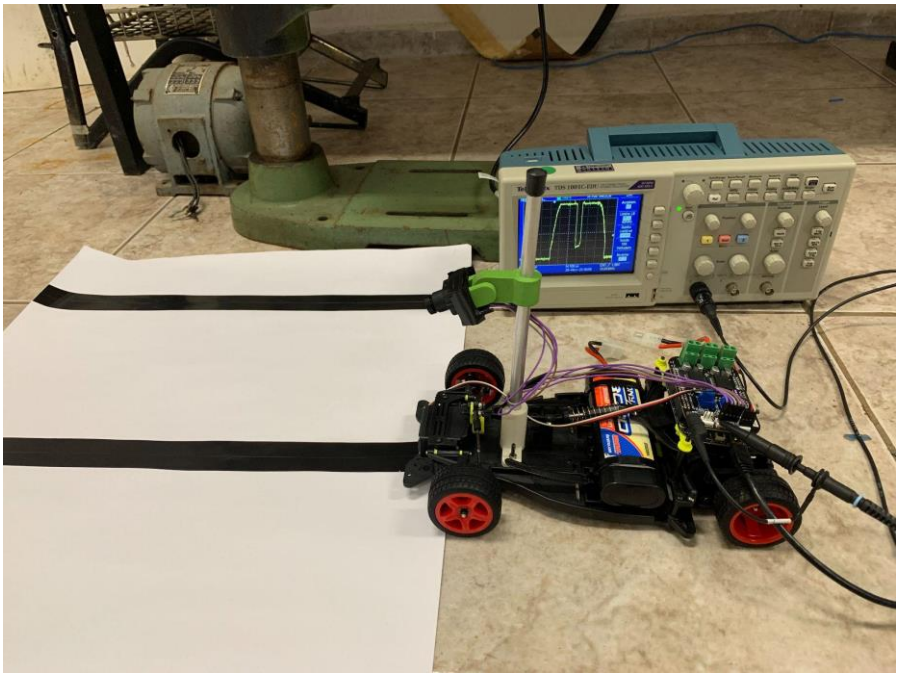
Fonte: Elaborado pelo autor

3.3 Câmera

Para fazer a captura da imagem foi utilizada a câmera *Freescale Cup LineScan* a qual conta *array* 128x1 de sensores da fabricante TAOS (*Texas Advanced Optical Solutions*), modelo TSL1401CL [5], composto por 128 fotodetectores em uma só linha. Assim, a imagem resultante é composta por níveis de energia onde intensidades mais baixas de luz resultam em níveis mais baixos de energia, enquanto intensidades maiores de luz resultam em níveis mais altos. Como a câmera é alimentada por uma tensão de 3,3V, as intensidades de luz variam dentro dessa margem, sendo cores escuras como o preto próximas de 0V e cores mais

claras como o branco resultam em tensões próximas a 3,3V, conforme mostrado na Figura 7.

Figura 7 - Imagem resultante no osciloscópio

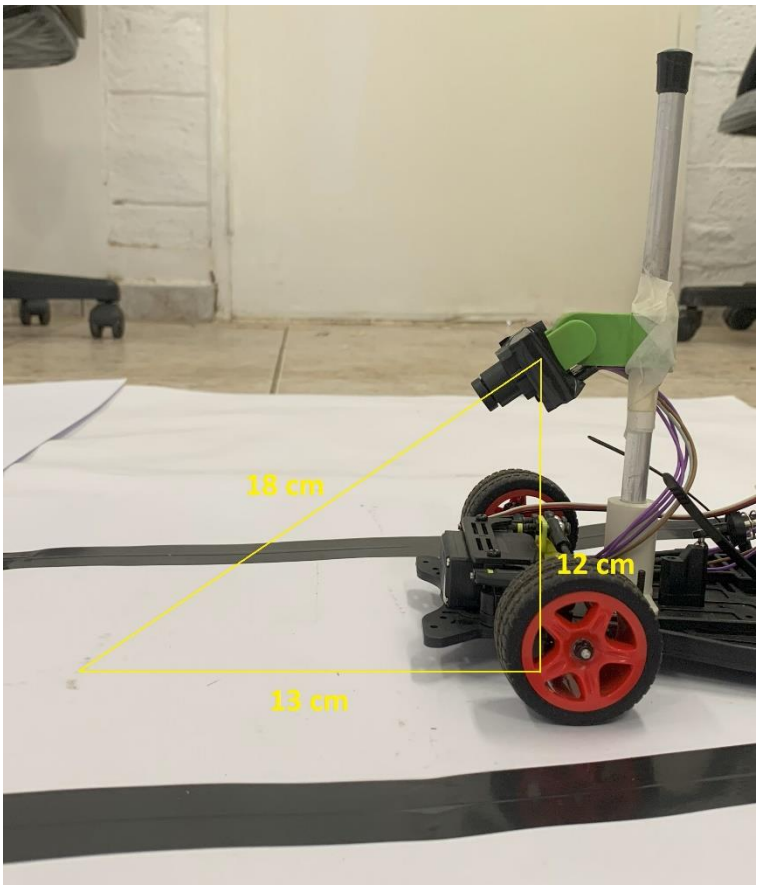


Fonte: Elaborado pelo autor

Conforme mostrado na Figura 7, quando a câmera detecta maior intensidade de luz da cor branca temos um pico de energia alcançando os 3,3V. Já quando temos a detecção da linha preta, no caso da imagem está centralizada, temos um vale na imagem do osciloscópio se aproximando de 0V.

Por recomendação da fabricante, originalmente a altura ideal da câmera deveria ser entre 15 e 30 cm em relação ao solo para garantir uma boa captação de imagem. Como o projeto segue as proporções de um carro real, a câmera foi posicionada a uma altura de 12 cm em relação ao solo, resultante em uma distância até o ponto de captura da imagem de 18 cm, sendo 13 cm a frente veículo, conforme mostrado na Figura 8.

Figura 8 - Posicionamento da Câmera



Fonte: Elaborado pelo autor

Para conseguir alimentar e controlar a câmera, que conta com 5 pinos, sendo o pino terra (GND), a alimentação (VDD), um pino para sinal de *clock* (CLK), um de entrada serial (SI) e um pino de leitura (AOUT), mostrada na Figura 9.

A captura da imagem é determinada pelos pulsos de clock e da entrada serial, sendo que ao ter um pulso no sinal SI começam a ser registrados os dados seguindo o pulso de clock, onde cada pulso de clock é responsável pela varredura de um pixel. Assim, para fazer a leitura completa do array são necessários 128 pulsos ciclo do sinal de clock (duty cycle). A biblioteca TFC.h vem programada com uma largura de pulso de clock de 3 ms, ou 333,3 Hz, enquanto o pulso SI é de 32,8 µs, ou 30,49 kHz.

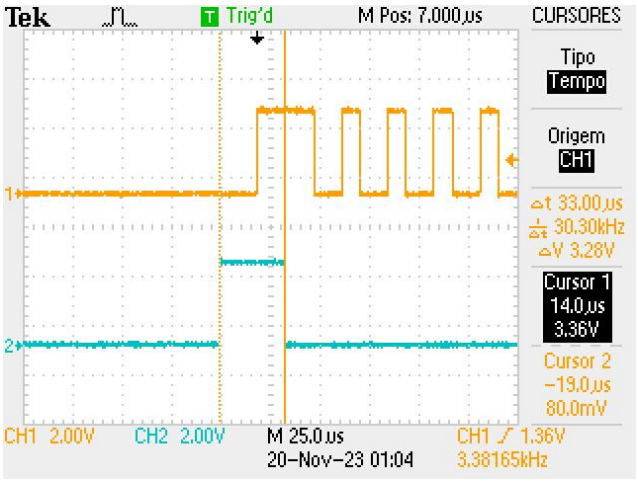
Figura 9 - Câmera utilizada

Fonte: Elaborado pelo autor

Para que tenhamos uma leitura correta do sinal, os pulsos de *clock* e SI devem estar defasados em ½ fase, onde o pulso SI está adiantado em relação ao *clock* [6], conforme mostrado na Figura 10.

Conforme varia-se a largura dos pulsos SI e *clock* tem impacto direto na leitura da câmera, uma vez que eles determinam o Tempo de Integração que é o tempo de "carga" do pixel. Por isso, quanto mais for a frequência de *clock* e pulso SI menor o tempo de integração, logo o pixel fica menos saturado, resultando em um sinal com menor tensão quando monitorado no osciloscópio. Quanto maior for o tempo de integração, ou seja, menor frequência e largura do pulso SI, teremos uma leitura mais saturada do pixel, com maior nível de energia detectado.

Figura 10 - Defasagem dos pulsos de Clock e SI



Fonte: Elaborado pelo autor

Além das frequências citadas anteriormente, outra variável importante para garantir uma boa leitura do pixel é o ajuste de foco da câmera. Caso a foco não esteja ajustado corretamente, a leitura da câmera monitorada no osciloscópio pode conter muito ruído prejudicando a detecção das faixas laterais, podendo não enxergar uma das faixas, e na diferenciação dos níveis de energia, tornando a leitura imprecisa.

Para a leitura da câmera, o código utilizava uma função chamada “LineScanImageReady” que habilitava a varredura da câmera resultando em um vetor com 128 posições, sendo que em cada posição era armazenado um dado inteiro sobre a intensidade detectada. Assim, era feita uma varredura no vetor procurando o momento em que ocorria uma mudança de intensidade captada com base em um limiar de decisão, fazendo com que fosse guardado em uma variável de memória e, em seguida, era feita uma segunda leitura procurando pela segunda faixa utilizando a mesma lógica, baseada na diferença de intensidade entre os pixels, armazenando o dado em uma segunda variável de memória. Como o processamento da imagem é feito com 12 bits de amostragem, temos que o sinal de 3,3V era discretizado em 4.096 amostras. Com isso, foi determinado um limiar de decisão (*thresohold*) de cerca de 50% da onda, sendo um limiar de 2.048 amostras ou 1.65V, tendo em vista que por se trabalhar com intensidade de luz o ambiente interfere diretamente na imagem captada, assim com esse limiar foi possível ter uma

leitura que desprezasse efeitos de sombras e luz excessiva. A Figura 11 mostra o trecho do código desenvolvido para a localização das faixas da pista.

Figura 11 - Trecho de código para localização das faixas

```
if(LineScanImageReady == 1){
    LineScanImageReady = 0;

    for(i = 0; i < 127; i ++){
        if(LineScanImage0[i] >= 2048 && M == 0){
            M = 1;}
        if(LineScanImage0[i] < 2048 && M == 1){          //Detecção da primeira faixa
            N1 = i - 1;
            M = 2;}
        if(LineScanImage0[i] >= 2048 && M == 2){         //Detecção da segunda faixa
            N2 = i;
            M = 0;}

    }
```

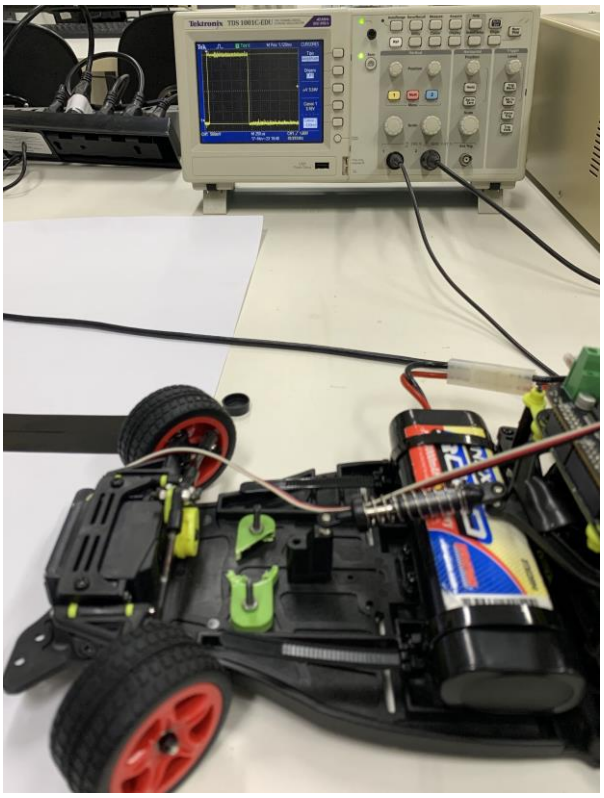
Fonte: Elaborado pelo autor

Com base nas posições encontradas da faixa, foi calculada uma posição alvo, sendo o centro da pista para se aplicar no controle de direção do servo motor. Essa posição alvo é encontrada tirando a média entre as posições das faixas.

3.4 Servo motor

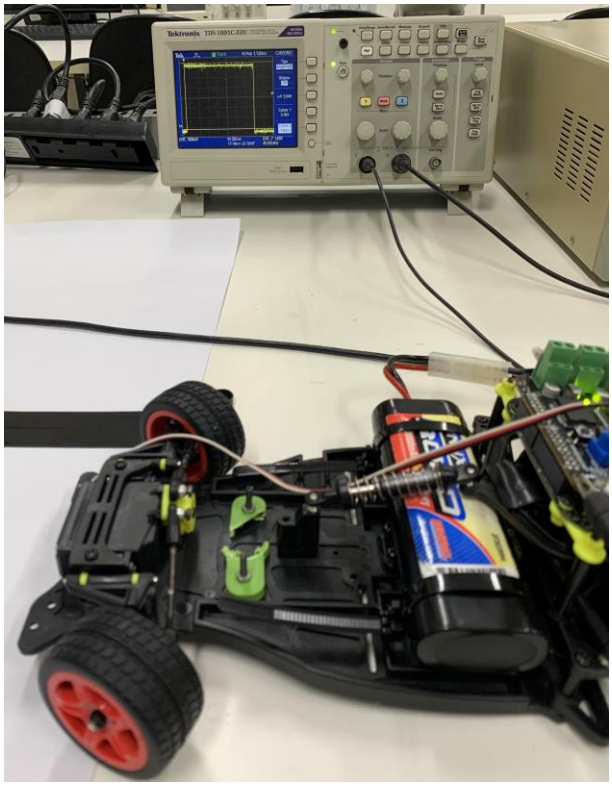
A função do servo motor no projeto foi de fazer o controle da direção, podendo ter uma variação de até 90º para cada lado. Entretanto, por limitação do chassi o ângulo máximo para virar era de apenas 54º seja para a direita ou para a esquerda, equivalente a apenas 60% do ângulo total para cada lado. O acionamento do servo era feito com base em um pulso PWM, ou seja, para que virasse para a direita foi utilizado um pulso mais largo, enquanto para virar para a esquerda era necessário um pulso mais estreito. Para virar à direita a largura de pulso máxima era de 533,3 Hz ou 1,9 ms, enquanto para virar para a esquerda era de 833,3 Hz, sendo a posição central do servo de 683,3 HZ. As Figuras 12 e 13 mostram o espectro para virar à esquerda e à direita, respectivamente.

Figura 12 - Espectro virando para a esquerda



Fonte: Elaborado pelo autor

Figura 13 - Espectro virando para a direita

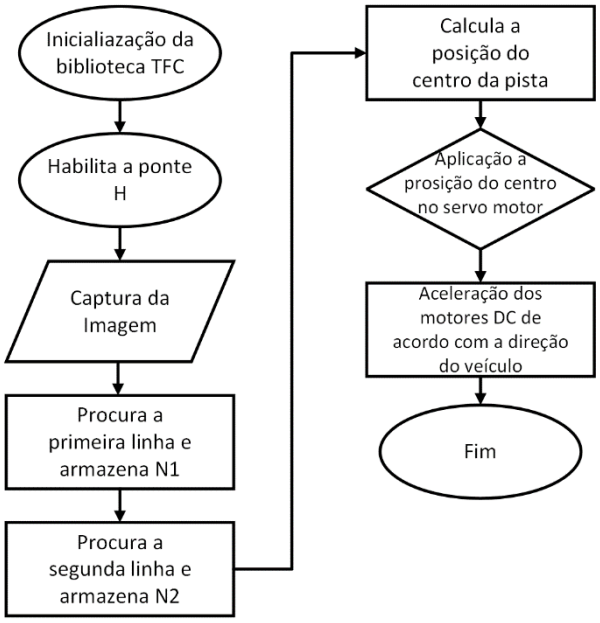


Fonte: Elaborado pelo autor

A tomada de decisão sobre qual direção a ser seguida se dá através imagem gerada pela câmera, na qual a lógica do código é baseada em encontrar a posição de cada faixa quando ocorre a mudança entre um pixel claro e um escuro. Assim, quando ocorresse essa mudança, a posição era armazenada em uma variável “Nx”, sendo N1 para a faixa da esquerda e N2 para a faixa da direita.

Com a posição das faixas encontradas, o algoritmo calcula a posição do centro do centro da pista fazendo a média entre as posições das faixas laterais, sendo a posição do centro da pista aplicada no servo motor. A Figura 14 mostra o fluxograma com a lógica do controle de direção.

Figura 14 - Fluxograma do funcionamento da direção



Fonte: Elaborado pelo autor

Para o valor da posição alvo ser aplicada no servo motor, foi necessária uma função de normalização para adaptar o valor dentro da margem de -0.6 a 0.6, onde a posição alvo era dividida por 128, que é o valor total de pixels ou posições possíveis, multiplicado por 1,2 por ser o espectro possível de variação (de -0.6 a 0.6) e subtraído 0.6 por ser o valor mínimo, conforme a Equação (01).

$$posição\ servo = \frac{posição\ alvo}{128} * 1{,}2 - 0{,}6 \qquad (01)$$

Na declaração do servo via código, era chamada a função “TFC_SetServo (servo X, posição)”, onde foi designado qual servo seria utilizado o servo 0. Mas, caso fosse necessário o uso de um outro servo ele seria declarado como “servo 1”, e a “posição” era determinada utilizando um *float,* podendo variar de -1.0 a 1.0, no caso a posição máxima que ele poderia girar é de -0.6 a 0.6.

Para evitar que o veículo se perdesse com facilidade, ao corrigir sua posição para o centro da pista, o código utilizado tinha três faixas de atuação. A primeira era para caso em que o veículo precisasse contornar uma curva para a esquerda tendo como meio da pista posições maiores que 70, assim o servo motor era acionado em 0.6. A segunda situação era quando tínhamos uma curva para a esquerda, ou seja, a posição do meio da pista era menor que 60, onde o servo era acionado em -0.6. Por fim, para pequenas correções de posição o servo recebia o valor normalizado pela equação anterior, sendo dentro da faixa de posição entre 60 e 70, sendo o código mostrado pela Figura 15.

Figura 15 - Lógica de controle do servo motor

```
if(M_P > 70){TFC_SetServo(0, 0.6);
        TFC_SetMotorPWM(0.15, 0.35);}
    if(M_P < 60){TFC_SetServo(0, -0.6);
        TFC_SetMotorPWM(0.35, 0.15);}
    else {(TFC_SetServo(0,servo));
        TFC_SetMotorPWM(0.15, 0.15);}
```

Fonte: Elaborado pelo autor

3.5 Motores

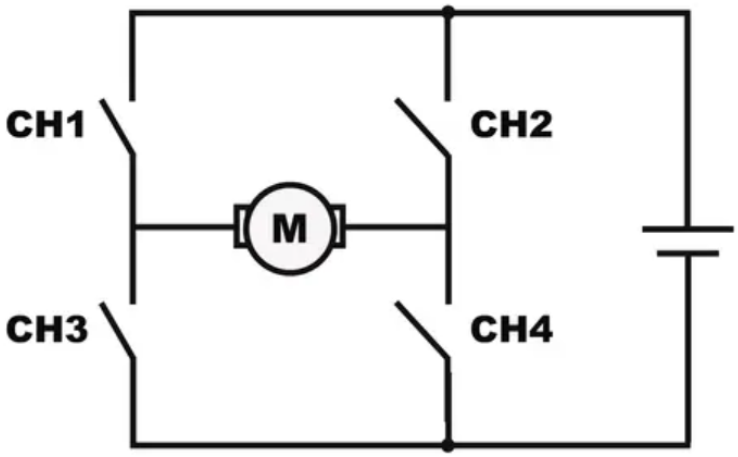
No kit da Freescale, o chassi conta com dois motores DC na parte traseira alimentados por 7,2V com a função de tracionar o carro pelo percurso, onde temos que a tensão (V) é responsável pela velocidade do veículo enquanto a corrente (I) é relacionada com o torque empregado pelos motores.

Em um veículo real, os motores seriam equivalentes a parte de transmissão e diferencial. Esses sistemas são importantes não só quando falamos de movimento

retilíneo onde ambas as rodas trabalham com a mesma velocidade, mas também em situações de curva onda as rodas experimentam velocidades diferentes, uma vez que a roda de dentro percorre uma distância linear menor em função de estar mais próxima do eixo do movimento. Logo, o raio é menor, então temos que a velocidade da roda interna da curva deve ser menor que a da roda externa. Caso isso não ocorra, o veículo irá perder tração e derrapar, podendo ser uma situação extremamente perigosa por ter com resultado a perda de controle do veículo.

Assim, na situação de mundo real são utilizados diferenciais abertos ou ativos que permitem as rodas desenvolverem velocidades diferentes de acordo com a trajetória que estão percorrendo. No protótipo utilizado, quem faz o controle das rodas de tração simulando um diferencial aberto são duas pontes H, controladas por chaveamento que permite corrente bidirecional, fazendo o controle das rodas de forma independente [7], conforme mostrado na Figura 16.

Figura 16 - Funcionamento da ponte H



Fonte: [7].

O fato de a ponte H permitir trabalhar com corrente bidirecional é possível de controlar o sentido do motor, para frente ou para trás, fazendo o chaveamento cruzado acionando chaves de nível alto e baixo de forma oposta, ou então frear o motor simulando um curto-circuito fechando as chaves de nível baixo resultando em um torque de parada (*stall torque*).

Para realizar o acionamento e controle dos motores e as pontes H, é necessário fazer a chamada da função “TFC_HBRIDGE_ENBALE” pra acionar a ponte H ou “TFC_HBRIDGE_DISABLE” para desabilitá-la, enquanto que o controle das velocidades dos motores era declarado na função “TFC_SetMotorPWM(Motor A, Motor B)”, sendo que a declaração de velocidade era variável de -1 até 1, onde o -1 seria a velocidade mais rápida para trás e o 1 o mais rápido para frente. A Figura 17 mostra como é feita a declaração e acionamento dos motores no código.

Figura 17 - Declaração e acionamento dos motores

```
TFC_HBRIDGE_ENABLE;
TFC_SetMotorPWM(0.16,0.16);
```

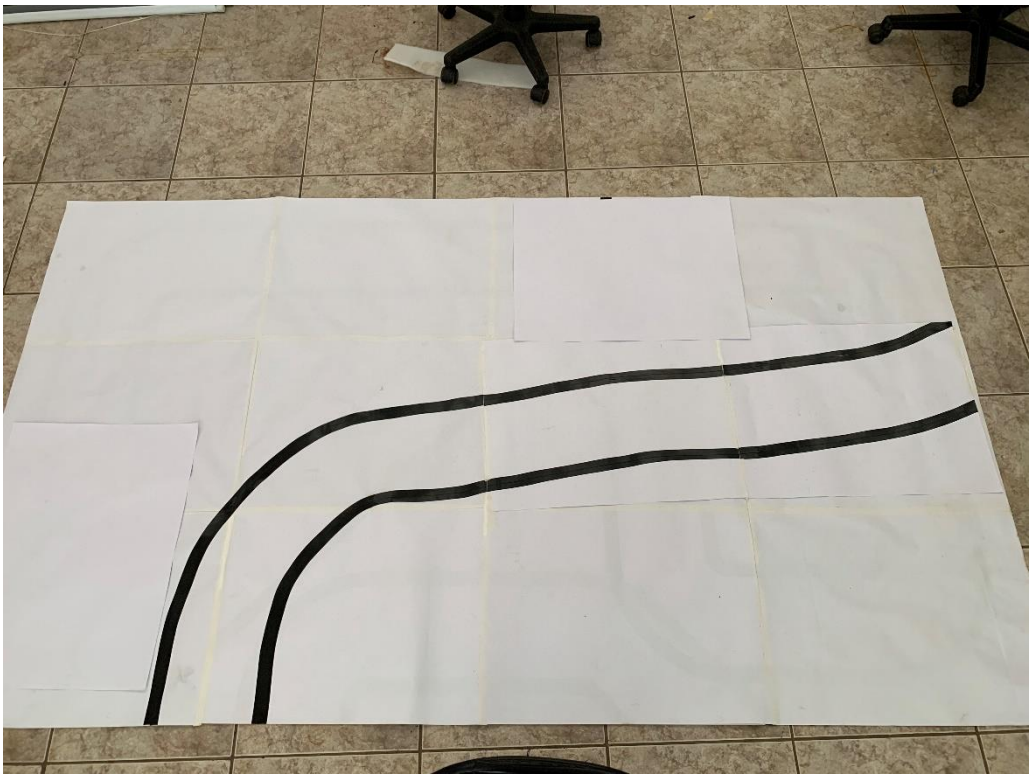
Fonte: Elaborado pelo autor

3.6 Pista

Para simular o circuito, foram montados trechos de pista com curvas para testar a capacidade do veículo de fazer curvas e testar sua capacidade de ajustar sua posição em relação ao centro da pista, sendo que originalmente para a competição Freescale Cup por regulamento a pista tinha a largura de 55 cm. Como foi respeitada uma escala de proporção, a largura da pista tinha 21 cm, uma vez que não temos um tamanho mínimo de faixa determinado pela legislação, mas temos que em média as faixas têm em torno de 3,40 m.

Com isso, foram montados trechos com curvas com raio maior e que evitavam ter curvas. Em seguida, para evitar que o veículo se perdesse, conforme mostrado na Figura 18.

Figura 18 - Trecho de pista utilizado



Fonte: Elaborado pelo autor

3.7 Desvio de objetos

A fim de simular situações próximas a de um caso prático, foi estudada uma lógica para detecção de objetos e obstruções da pista. A lógica utilizada foi baseada em encontrar o tamanho da pista com base nas posições das faixas que foram encontradas na varredura. Assim, quando a distância entre as faixas era menor do que um limiar esperado, o veículo deveria parar e caso os valores dessa distância fossem dentro do limiar o veículo deveria contornar o objeto, seguindo a mesma lógica do controle de curvas, onde encontra a posição central da faixa para aplicar e controlar o servo motor. O código utilizado é apresentado pela Figura 19.

Figura 19 - Lógica para desvio de objetos e obstrução

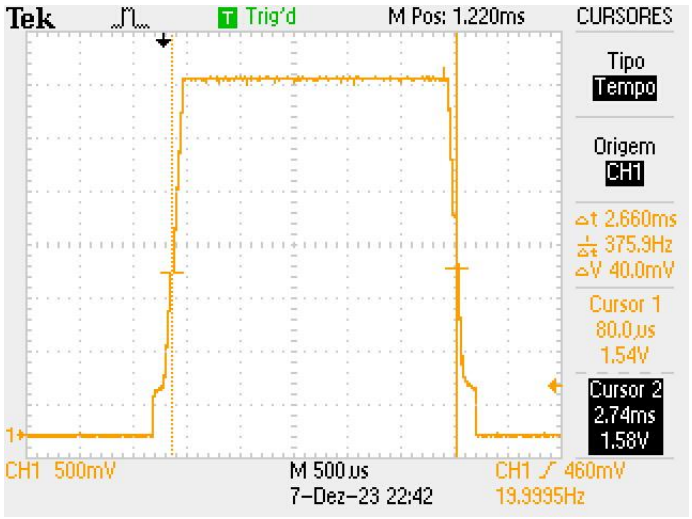
```
OB = N1 - N2;
M_P =(N1 + N2)/2;

float servo = ((float)M_P/128.0) * 1.2 - 0.6;

if (O_B < 60){TFC_SetServo(0, 0);
TFC_SetMotorPWM (0.0, 0.0);
TFC_HBRIDGE_DISABLE;}

if(O_B <= 61){
if(M_P > 70){TFC_SetServo(0, 0.6);
        TFC_SetMotorPWM(0.15, 0.35);}
    if(M_P < 60){TFC_SetServo(0, -0.6);
        TFC_SetMotorPWM(0.35, 0.15);}
    else {(TFC_SetServo(0,servo));
        TFC_SetMotorPWM(0.15, 0.15);}
}
```

Fonte: Elaborado pelo autor

4. Resultados

Para o sistema de detecção da pista temos, que o resultado do espectro mostrado no osciloscópio possui as faixas laterais, de cor preta, nas extremidades do espectro, sendo observado um baixo nível de energia, e responsáveis por centralizar o veículo em retas e determinar quando tinha uma curva, sendo mostrado na Figura 20.

Figura 20 - Espectro das faixas laterais



Fonte: Elaborado pelo autor

Com base nas metodologias aplicadas, os resultados obtidos foram que a plataforma utilizada possui algumas limitações principalmente para se contornar curvas. Na competição da Freescale a pista tinha uma largura maior e com um raio de curva grande para contornar esse problema.

Em um primeiro momento, foi tentado um circuito fechado com curvas com raio pequeno em 90º, entretanto o veículo ser perdia com facilidade. Em poucas das situações que ele conseguia fazer a curva eram quando a velocidade entre as rodas internas e externas tinham velocidades discrepantes, onde a roda externa tinha uma velocidade bem maior que a interna. Contudo, o problema desta solução foi que para pequenas correções de trajeto o veículo se perdia e excedia os limites das faixas.

Quando utilizado um trecho de pista com curvas com raio maior e com uma lógica de controle de direção progressiva, o veículo contornava a pista sem problemas e com boa precisão.

Em relação a tentativa de desviar ou parar ao detectar objetos e obstruções na pista, quando a obstrução era pequena o veículo conseguiu contornar com mínimos problemas. Todavia, ao tentar contornar objetos maiores ou lidar com a pista totalmente obstruída os resultados não foram com esperado, onde o veículo teve dificuldade em detectar o objeto.

5. Conclusão

Como proposta de estudar e desenvolver o controle de um veículo autônomo de forma simples o kit da Freescale foi uma boa plataforma de desenvolvimento, sendo que as principais limitações foram relacionadas a escala e proporções utilizadas.

Foram observados que a altura da câmera limitou o espectro observado pela câmera, sendo que se fosse posicionada a uma altura maior que possibilitasse captar imagens mais distantes do veículo, seria possível se trabalhar com velocidades maiores.

O sistema de direção do veículo limitou a capacidade de esterçar as rodas o que dificultou o contorno de curvas com raio menor, assim o circuito a ser explorado precisa ter um raio maior de curvas.

Como consequência, surgiram oportunidades para trabalhos futuros, uma vez que o desvio de objetos e obstruções não tiveram os resultados esperados. Além disso, a lógica de controle de direção pode ser melhorada explorando um controle exponencial da posição do servo, tendo a possibilidade de desenvolver um sistema de monitoramento e controle de velocidade, uma vez que o Shield TFC apresenta suporte para ele, o que auxiliaria tanto nas pequenas correções de percurso até curvas com diferentes raios.

6. Referências

[1] Allianz Partners. Estudo prevê zero mortes na estrada até 2040. Disponível em: https://www.allianz-partners.com/content/dam/onemarketing/awp/azpartnerscom/brazil/press-releases/Press-release-Report-Mobilidade-Allianz-Partners.pdf. Acesso em: 12/05/2023.

[2] NXP Comunity. (2013). Freescale Cup Shield for the Freedom KL25Z. Disponível em: https://community.nxp.com/t5/University-Programs-Knowledge/Freescale-Cup-Shield-for-the-Freedom-KL25Z/ta-p/1104533#toc-hId-1994452635 . Acesso em 15/06/2023

[3] Ficha Técnica Chevrolet Equinox. Disponível em: https://www.chevrolet.com.br/bypass/seg1_tools/vehicles-n02/suvs/equinox/versoes-e-ficha-tecnica/technical-specs.html. Acesso em: 03/09/2023.

[4] Veja. (2023). Em alta, SUVs representam quase metade dos carros vendidos no Brasil. Disponível em: https://veja.abril.com.br/comportamento/em-alta-suvs-representam-quase-metade-dos-carros-vendidos-no-brasil. Acesso em: 10/10/2023.

[5] NXP Comunity. (2012). Line Scan Camera Use. Disponível em: https://community.nxp.com/t5/University-Programs-Knowledge/Line-Scan-Camera-Use/ta-p/1105313. Acesso em: 18/06/2023.

[6] Freescale Semiconductor. (2011). Using Parallax TSL1401-DB Linescan Camera Module for line detection. Disponível em: https://www.nxp.com/docs/en/application-note/AN4244.pdf. Acesso em: 15/06/2023.

[7] Manual da Eletrônica. Ponte H - O que é e como funciona. Disponível em: https://www.manualdaeletronica.com.br/ponte-h-o-que-e-como-funciona/. Acesso em: 01/10/2023.