Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia Eletrônica
Engenharia de Controle e Automação

API Para Operação de um Motor DC

Alunos:

Bruno Guimarães Bitencourt - 2015.113.279

Ian Fernandes Miranda - 2015.113.295

Professor:

Ricardo de Oliveira Duarte

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Trabalho apresentado a disciplina de Programação de Sistemas Embarcadas, ministrada pelo professor Ricardo de Oliveira Duarte, durante o primeiro semestre letivo de 2021. Os autores, Bruno e Ian, desenvolveram uma API para o controle de motores DC de veículo de duas rodas, implementada para a placa NUCLEO-F103RB. O objetivo da biblioteca é permitir a manipulação das velocidades dos motores a fim de realizar manobras no veículo.

2 Metodologia

2.1 Configuração STM32CubeMX

Ao abrir STM32CubeMx, selecionou-se a opção *Access To Board Selector*, conforme mostrado na Figura 1.

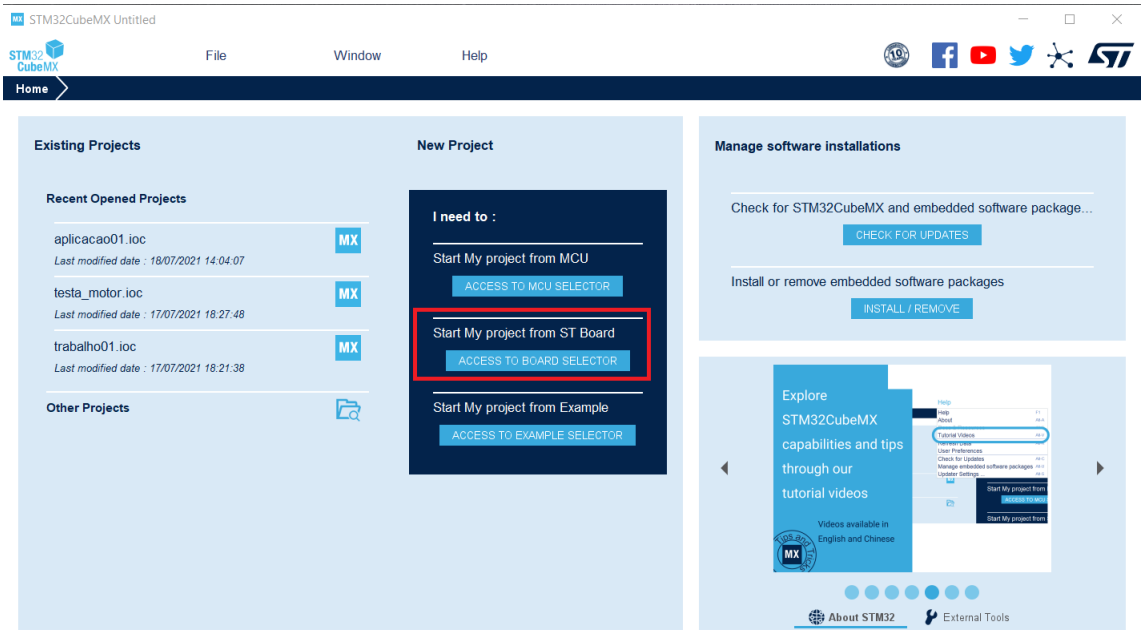

Figura 1: Configuração 01 ao abrir o software STM32CubeMX

Foi selecionado o modelo *NUCLEO-F103RB*, conforme mostrado na Figura 2.

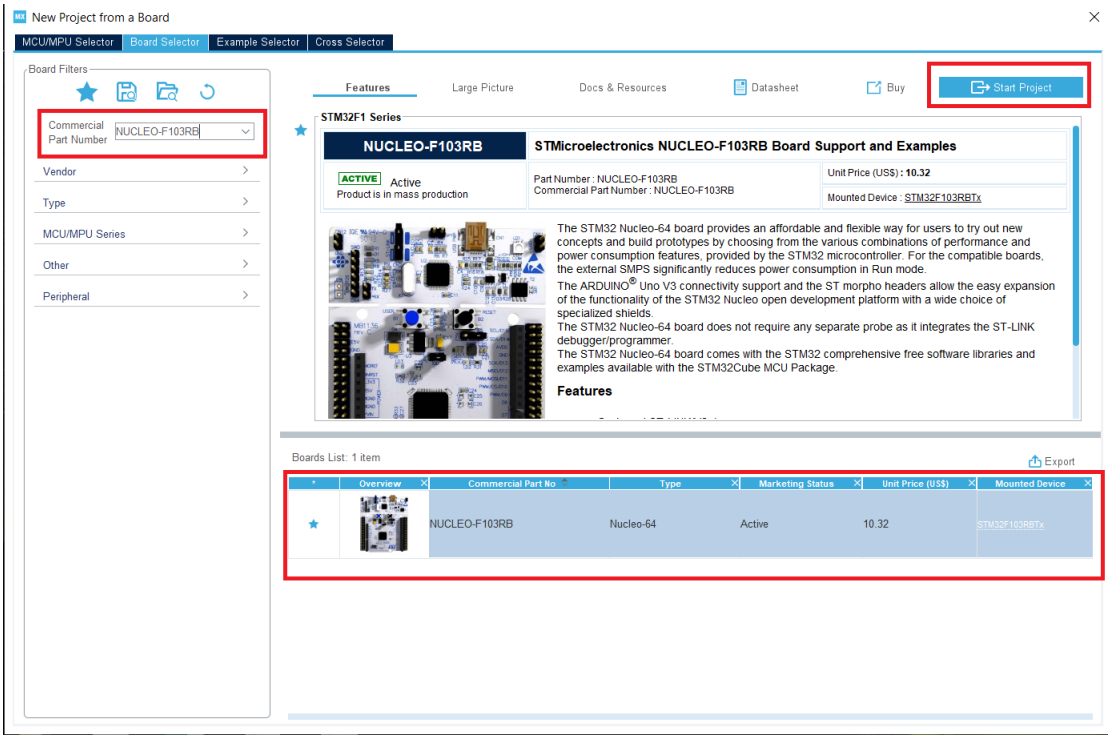

Figura 2: Seleção do modelo *NUCLEO-F103RB* nas configurações do STMCubeMX

Foi configurado o temporizador associado ao PWM utilizando o *TIM2* e considerando um Counter Period 1023. Foram habilitados os 4 canais do temporizador como *PWM Generation*, com a configuração de *Internal Clock* conforme mostrado pela Figura 3.

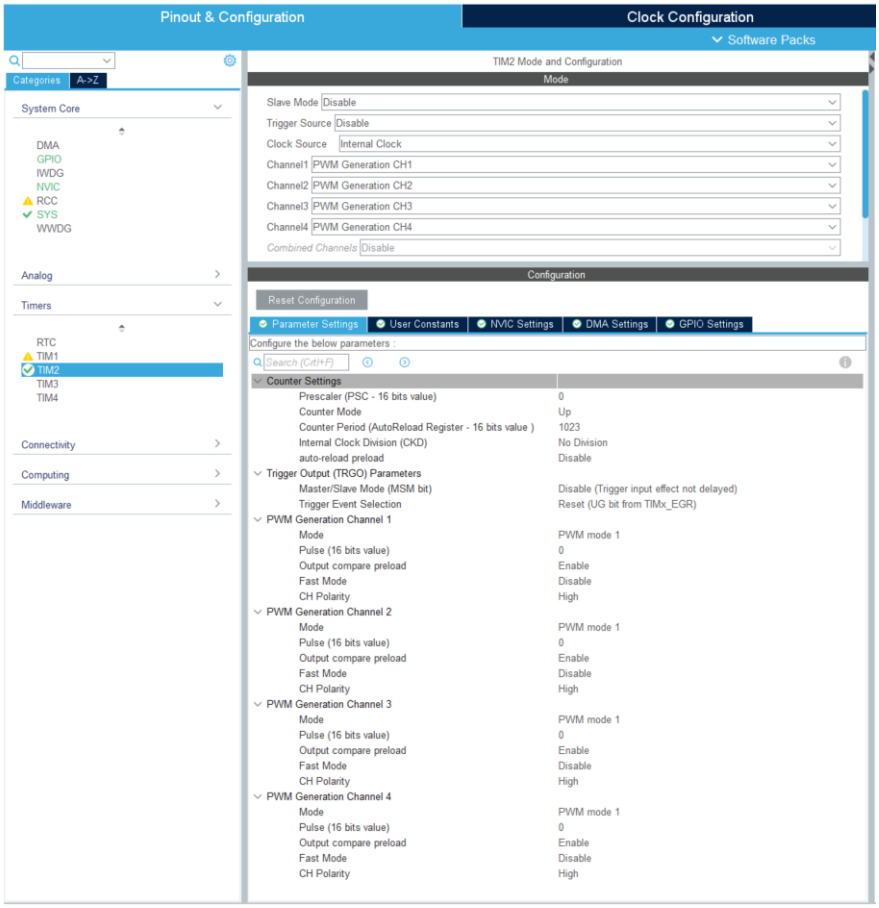

Figura 3: Configuração do Timer associado à geração de PWM, utilizando o STM32CubeMX

A Figura 4 mostra os pinos utilizados pelo microcontrolador. A Tabela 1 mostra a relação de entradas utilizada.

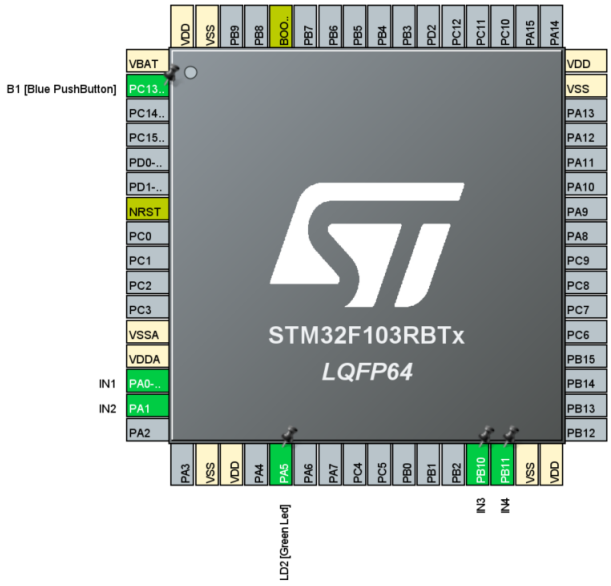

Figura 4: Pinagem utilizada para acionamento dos motores

Tabela 1: Relação de entradas para acionamento de dois motores no microcontrolador

Temporizador 2 PWM	Motor	Sentido Motor	Pinagem
Canal 01	Motor 01	Direto	IN1 - PA0
Canal 02	Motor 01	Reverso	IN2 - PA1
Canal 03	Motor 02	Direto	IN3 - PB10
Canal 04	Motor 02	Reverso	IN4 - PB11

2.2 API: Acionamento de Motores para um Carro Robótico

A partir do controle dos dois motores DC, é possível realizar movimentos para frente, movimentos para trás girar para um lado girar para o outro lado, girar 180° e dar ré. A partir desses movimentos é possível implementar um veículo autônomo simples utilizando uma série de sensores para realizar a percepção do ambiente, como por exemplo, sensores de fim de curso ou sensores ultrassônicos para medir a distância a objetos.

2.2.1 Estrutura Mecânica

A estrutura do carro robótico é mostrada em 5.

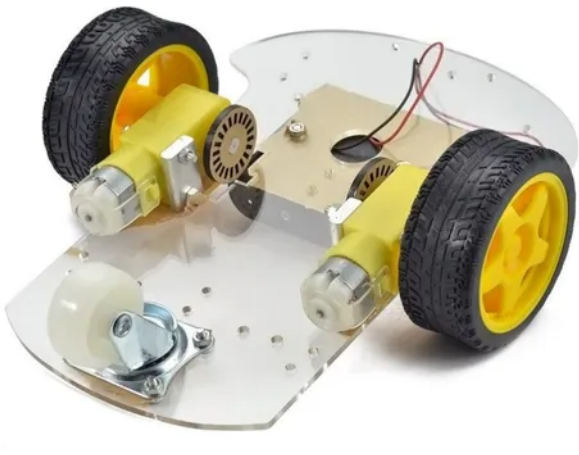

Figura 5: Estrutura mecânica do carro robótico utilizado. Fonte: [2]

O chassi apresenta dimensões de 21,2 x 15,2cm e as rodas de 7 x 7 x 2,6 cm.

2.2.2 Montagem Eletrônica

Para o controle da direção do giro dos motores, foi utilizado a um circuito denominado ponte H, cuja função é mudar o sentido de rotação dos motores. A Figura 6 mostra uma visão geral do circuito.

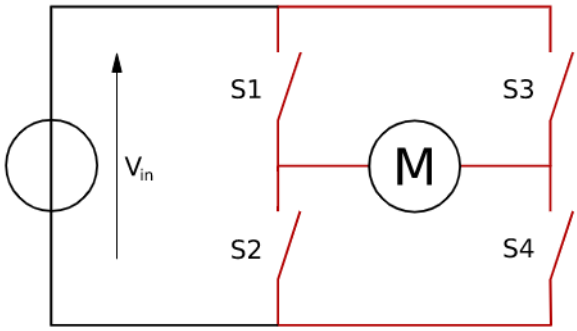

Figura 6: Visão geral de um circuito de ponte H. Fonte: [3]

Analisando a Figura 6, para acionamento do motor no sentido direto, deve-se acionar as chaves S1 E S4, ou no sentido inverso (chaves S2 E S3). Além disso, ligando S1 e S3 OU S2 e S4 é possível obter o efeito de frenagem. Para o projeto, foi utilizado o Driver Motor Ponte H L298n, mostrado na Figura 7. É baseado no chip L298N, construído para controlar cargas indutivas como relés, solenoides, motores DC e motores de passo.

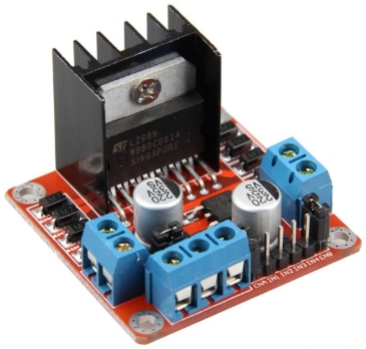

Figura 7: Driver Motor Ponte H L298n. Fonte: [4]

A Tabela 2 mostra os parâmetros mecânicos e elétricos relacionados à interface de potência.

Tabela 2: Parâmetros do Driver de Potência.

Parâmetros Elétricos e Mecânicos	
Tensão de Operação	6-35V
Corrente de Operação Máxima	2A
Limites de Temperatura	-20ºC a 135ºC
Potência Máxima	25W

A interface apresenta as seguintes entradas/saídas:

- Alimentação: Pino responsável por alimentar o circuito de potência. Sua faixa de alimentação é de 6V a 35V.
- GND: Terra.
- Sinais de entrada (IN1, IN2, IN3, IN4): Entradas de controle usadas para o acionamento.
- Enable: Permite o controle da velocidade quando em sinal alto.

O motor utilizado é mostrado na Figura 8.

Figura 8: Motor DC utilizado. Fonte: [5]

Ele tem como características:

- Eixo duplo
- Tensão de Operação: 3-6V
- Redução: 1:48
- Peso: 30g
- Corrente sem carga: 200mA (6V) e 150mA (3V)
- Velocidade sem carga: 200RPM (6V) e 90RPM (3V)

A Figura 9 mostra a montagem final do circuito proposto.

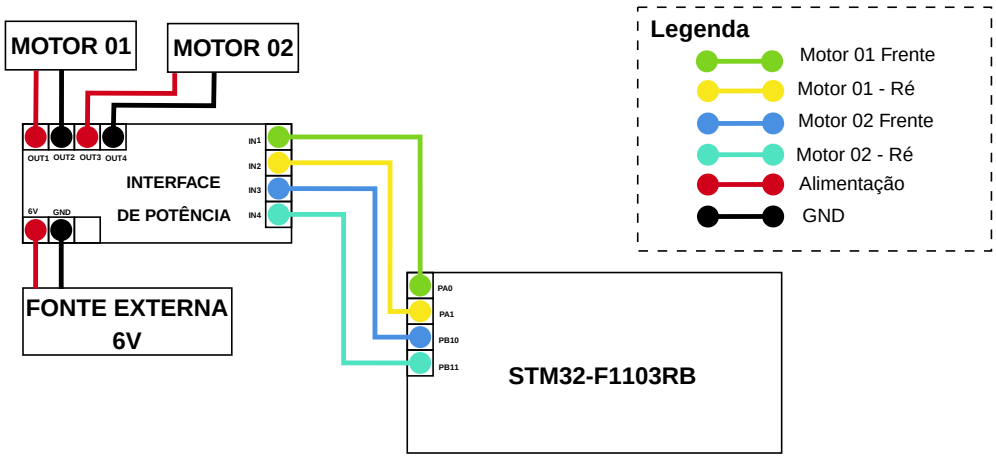

Figura 9: Montagem Final do Circuito da Aplicação

2.2.3 Programação

Para o acionamento dos motores, foi utilizada a função pré-existente *setPWM*, disponibilizada por [1]. A função mostrada no trecho de Código 1 tem como os parâmetros o identificador do Timer utilizado, o canal, o período e o valor do pulso.

```
1  void setPWM(TIM_HandleTypeDef timer, uint32_t channel, uint16_t period, uint16_t pulse){
2    HAL_TIM_PWM_Stop(&timer, channel); // stop generation of pwm
3    TIM_OC_InitTypeDef sConfigOC;
4    timer.Init.Period = period; // set the period duration
5    HAL_TIM_PWM_Init(&timer); // reinititialise with new period value
6    sConfigOC.OCMode = TIM_OCMODE_PWM1;
7    sConfigOC.Pulse = pulse; // set the pulse duration
8    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
9    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
10   HAL_TIM_PWM_ConfigChannel(&timer, &sConfigOC, channel);
11   HAL_TIM_PWM_Start(&timer, channel); // start pwm generation
```

```
}
```

Código 1: Função para variação do valor de PWM. Fonte: [1]

A partir disso foram criadas funções para acionamento dos motores de acordo com as necessidades para navegação. De forma geral as funções são mostradas a seguir.

```
#ifndef CARROROBOTICONUCLEO_64_STMF103_H_
#define CARROROBOTICONUCLEO_64_STMF103_H_
#include "stm32f1xx_hal.h"

    void AndaFrente(TIM_HandleTypeDef timer);
    void AndaRe(TIM_HandleTypeDef timer);
    void Parar(TIM_HandleTypeDef timer);
    void ViraDireita90(TIM_HandleTypeDef timer);
    void ViraEsquerda90(TIM_HandleTypeDef timer);
    void Gira180(TIM_HandleTypeDef timer);
    void ViraEsquerdaSuave(TIM_HandleTypeDef timer);
    void ViraDireitaSuave(TIM_HandleTypeDef timer);

#endif /* CARROROBOTICONUCLEO_64_STMF103_H_ */
```

Código 2: Declaração das funções desenvolvidas na API.

A seguir, cada função é detalhada.

```
void AndaFrente(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 512);
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 525);
  setPWM(timer, TIM_CHANNEL_4, 1023, 0);
}
```

Código 3: Função que faz o carro robótico andar para frente

No Código 3, os motores da direita e da esquerda são acionados com a mesma velocidade para que o veículo ande para frente em linha reta com velocidade constante. Vale ressaltar que os valores do PWM passados para cada um dos motores é diferente visto que apesar de serem do mesmo modelo, o comportamento de cada um dos motores é específico. Esses valores foram calibrados a partir de testes.

```
void AndaRe(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 0);
  setPWM(timer, TIM_CHANNEL_2, 1023, 512);
  setPWM(timer, TIM_CHANNEL_3, 1023, 0);
  setPWM(timer, TIM_CHANNEL_4, 1023, 520);
}
```

Código 4: Função que faz o carro robótico andar de ré

No Código 4, os motores da direita e da esquerda são acionados com a mesma velocidade para que o veículo ande para trás em linha reta com velocidade constante.

```
void Parar(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 0);
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 0);
  setPWM(timer, TIM_CHANNEL_4, 1023, 0);
}
```

Código 5: Função que faz o carro robótico parar

No Código 5, todos os motores são desligados para que o veículo pare de andar.

```
void ViraEsquerda90(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 480);
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 0);
  setPWM(timer, TIM_CHANNEL_4, 1023, 480);
  HAL_Delay(1500);
}
```

Código 6: Função que faz o carro robótico girar 90° a esquerda

A função presente no Código 6, aciona o motor da direita para frente enquanto aciona o motor da esquerda para trás, rotacionando o veículo em 90° para a Esquerda.

```
void ViraDireita90(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 0);
  setPWM(timer, TIM_CHANNEL_2, 1023, 480);
  setPWM(timer, TIM_CHANNEL_3, 1023, 480);
  setPWM(timer, TIM_CHANNEL_4, 1023, 0);
  HAL_Delay(1500);
}
```

Código 7: Função que faz o carro robótico girar 90° a direita

A função do Código 7, é semelhante à função do Código 6, no entanto para o sentido contrário. Dessa forma o veículo realiza uma rotação de 90° para a direita

```
void Gira180(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 480);
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 0);
  setPWM(timer, TIM_CHANNEL_4, 1023, 480);
  HAL_Delay(2000);
}
```

Código 8: Função que faz o carro robótico girar 180°

A função do Código 8, aciona o motor da direita para frente e o da esquerda para trás até o veículo rotacionar em 180°

```
void ViraEsquerdaSuave(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 480);
```

```
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 350);
  setPWM(timer, TIM_CHANNEL_4, 1023, 0);
  HAL_Delay(350);
}
```

Código 9: Função que faz o carro robótico fazer uma curva suave a esquerda

O Código 9 implementa a função para virar suavemente para a esquerda, ao invés de dar um giro em torno do próprio eixo como é implementado em 6. Isso é feito a partir da diferença de rotação dos motores da direita e da esquerda. Nessa caso, o motor da direita é acionado com uma velocidade maior que o da esquerda, levando o veículo a efetuar uma curva suave à esquerda.

```
void ViraDireitaSuave(TIM_HandleTypeDef timer){
  setPWM(timer, TIM_CHANNEL_1, 1023, 350);
  setPWM(timer, TIM_CHANNEL_2, 1023, 0);
  setPWM(timer, TIM_CHANNEL_3, 1023, 480);
  setPWM(timer, TIM_CHANNEL_4, 1023, 0);
  HAL_Delay(350);
}
```

Código 10: Função que faz o carro robótico fazer uma curva suave a direita

O Código 12, é análogo à função presente no Código 9, só que para realizar curvas suaves para a direita.

2.3 Aplicação utilizando a API

Como aplicação da API desenvolvida, foi proposto que o carro robótico faça o percurso de um quadrado da seguinte forma:

1. Anda para frente e depois de 3 segundos, para.
2. Vira a esquerda 90° e para.
3. Liga a ré por 3 segundos e para.
4. Vira a Direita 90°e para.
5. Liga a ré por 3 segundos e para.
6. Vira a Esquerda 90° e para.
7. Anda para frente e para.

Deve-se importar a biblioteca conforme mostrado abaixo:

```
#include "CarroRoboticoNUCLEO-64_STMF103.h"
```

Código 11: Importação da biblioteca na aplicação

Para isso, foi proposto o código mostrado abaixo:

```
while (1)
{
  /* USER CODE END WHILE */
  AndaFrente(htim2);
  HAL_Delay(3000);
  Parar(htim2);
  HAL_Delay(3000);

  ViraEsquerda90(htim2);
  Parar(htim2);
  HAL_Delay(1500);

  AndaRe(htim2);
  HAL_Delay(3000);
  Parar(htim2);
  HAL_Delay(3000);

  ViraDireita90(htim2);
  Parar(htim2);
  HAL_Delay(1500);

  AndaRe(htim2);
  HAL_Delay(3000);
  Parar(htim2);
  HAL_Delay(3000);

  ViraEsquerda90(htim2);
  Parar(htim2);
  HAL_Delay(1500);

  AndaFrente(htim2);
  HAL_Delay(3000);
  Parar(htim2);
  HAL_Delay(3000);


  /* USER CODE BEGIN 3 */
}
```

Código 12: Percurso em Quadrado do Carro Robótico

3 Conclusão

O trabalho concluiu seu objetivo no que diz respeito a implementação de uma API utilizando motores DC. Dentre as dificuldades encontradas pelo grupo a principal foi o ajuste dos parâmetros de PWM dos motores. Apesar de serem motores iguais e estarem alimentados com a mesma tensão eles apresentam um comportamento de velocidade diferente entre si, no qual, mesmo ajustando valores diferentes de PWM, ainda existem diferenças de velocidade. Tal comportamento se reflete nas ações de seguir reto ou virar.

Possíveis soluções de melhoria seria a adição de motores com encoders de velocidade, no qual seria possível o controle de velocidade em malha fechada, o erro seria eliminado. Além disso, para um seguimento adequado para linha reta por exemplo, poderia ser adicionado um acelerômetro, no qual medições sobre a curva a ser seguida do carro poderiam ser feitas resultando num erro zero para andar em linha reta por exemplo.

O grupo tentou utilizar encoders de velocidade para o controle da velocidade, entretanto, devido a resolução do sensor ser de baixa qualidade não foi possível a realização de um controle em malha fechada.

Referências

[1] Aula 13 - PWM (Pulse Width Modulation) - Modulação por Largura de Pulso - Notas de Aula - Ricardo de Oliveira Duarte.

[2] Kit Chassi 2WD Robô para Arduino - Filipe Flop <https://www.filipeflop.com/produto/kit-chassi-2wd-robo-para-arduino/>.

[3] Ponte H com bootstrap para acionamento de motores DC - Rodrigo Almeida <https://www.embarcados.com.br/ponte-h-bootstrap-acionamento-motores-dc/>.

[4] Módulo Ponte H - Felipe Flop <https://www.filipeflop.com/produto/driver-motor-ponte-h-l298n/>.

[5] Motor DC 3-6V com Caixa de Redução e Eixo Duplo <https://www.eletrogate.com/motor-dc-3-6v-com-caixa-de-reducao-e-eixo-duplo>.