Projeto ROS
UFPE - Demec
ME653 Programação Robótica

1 Descrição do projeto

Este projeto visa colocar em prática em uma plataforma robótica os conceitos relacionados ao ROS vistos em sala de aula. O projeto está dividido em duas partes principais. A primeira consiste em implementar os nós que fazem a ligação entre o hardware e um usuário. Este trabalho é semelhante ao realizado por um engenheiro que **construindo** o robô. A segunda parte exigiu a implementação de algoritmos semelhantes aos do usuário para resolver os desafios. Este trabalho é semelhante ao realizado por um engenheiro usando o robô. Este trabalho é semelhante ao realizado por um engenheiro **usando** o robô.

2 Plataforma robótica

2.1 Modelagem

Neste projeto, utilizamos um robô diferencial, que consiste em duas rodas acionadas independentemente que giram sobre o mesmo eixo, bem como um caster ball que mantém o robô na horizontal. Denotamos o raio das rodas como r e a distância entre as rodas como $2d$. Além disso, definimos u_L e u_R come a velocidade angular da roda esquerda e direita.
As velocidades linear v e angular ω do robô podem ser obtidas usando as equações seguinte:

$$v = \frac{r}{2}u_L + \frac{r}{2}u_R \tag{1}$$

$$\omega = -\frac{r}{2d}u_L + \frac{r}{2d}u_R \tag{2}$$

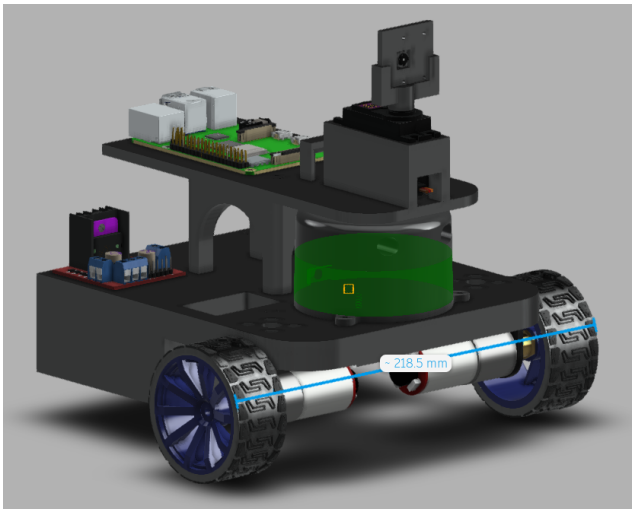

Figura 1: Modelo

2.2 Material

O robô usado neste projeto consiste nos seguintes elementos:

- 1x estrutura de plástico
- 2x motores DC 3-6V com caixa de redução
- 2x codificadores magnéticos
- 1x servomotor
- 1x câmera Raspberry Pi v2 8MP
- 1x laser YDlidar G2
- 1x mini-computador Raspberry Pi 4
- 1x placa ponte H
- 2x baterias

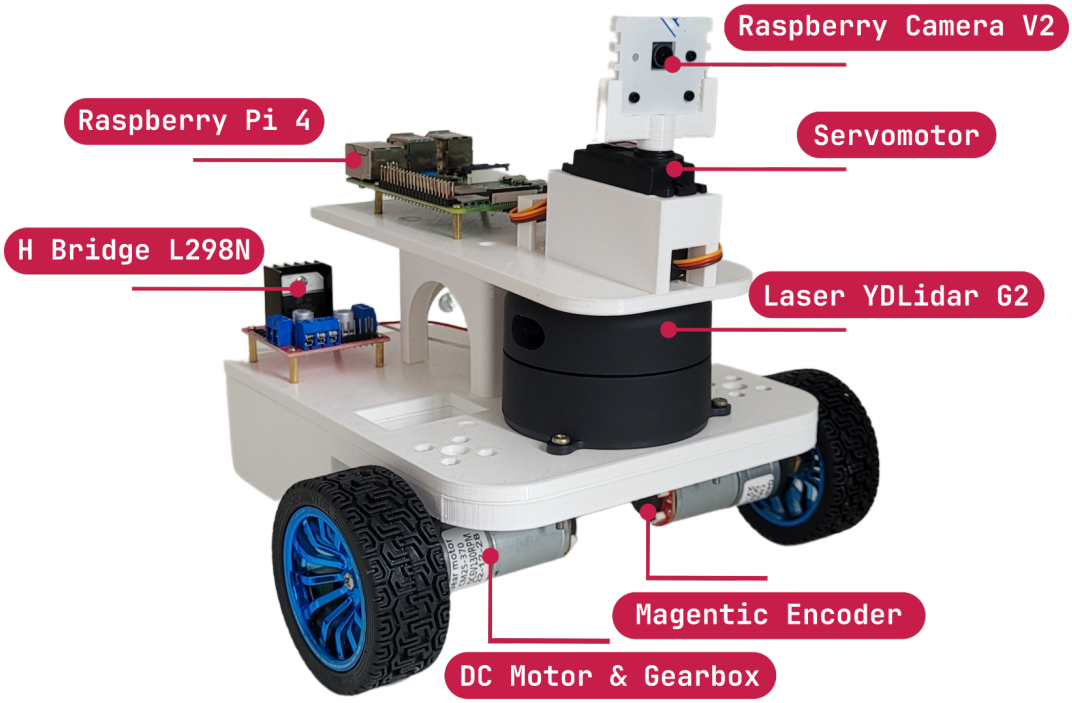

Figura 2: Plataforma robótica

Figura 3: Bateria 1 (Raspberry Pi & Laser)

Figura 4: Bateria 2 (Motores & codificadores)

3 Raspberry Pi 4

3.1 Hardware

O Raspberry Pi 4 (RP4) é um mini-computadores de placa única multiplataforma cujos componentes principais são mostrados na Fig. 5.

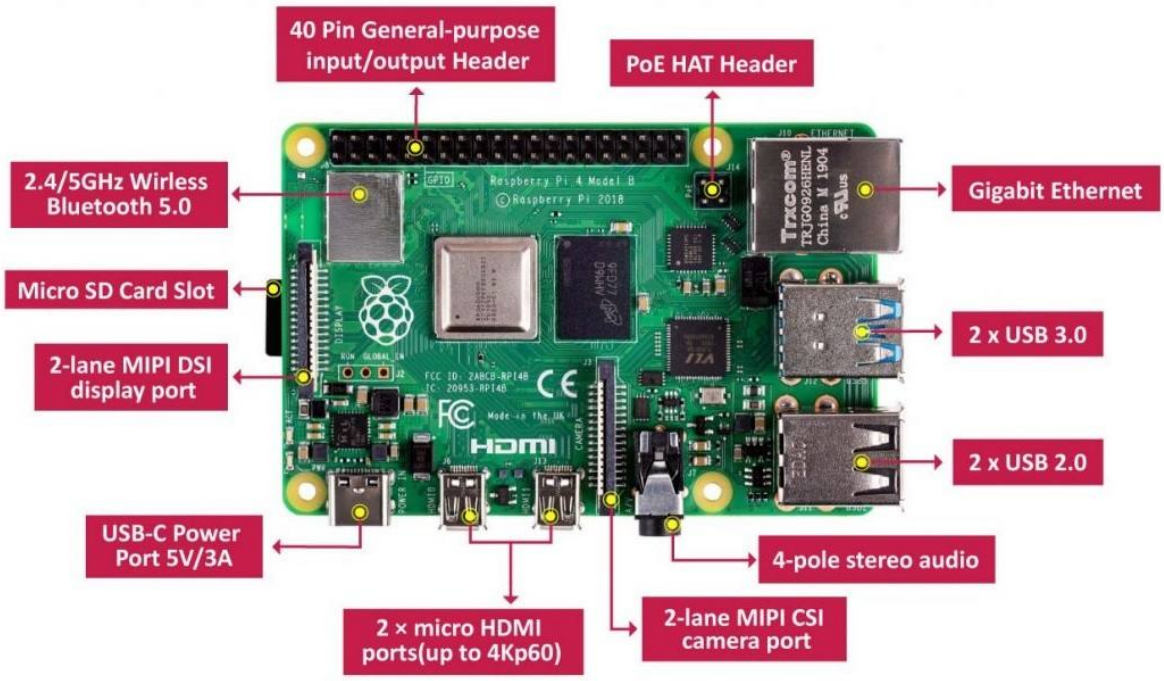

Figura 5: Raspberry Pi 4

3.2 Software

O Raspberry Pi4 está equipado com um cartão microSD contendo:

- O sistema operacional Ubuntu Mate 20.04
- O middleware ROS Noetic
- A linguagem de programação Python 3
- A biblioteca RPi
- A biblioteca OpenCv

.
Como o RP4 está a bordo do robô, não é prático usar um teclado e uma tela para interagir com ele. Neste projeto, portanto, usamos os comandos `ssh` e `scp` para interagir remotamente com o RP4. **Para usar os comandos apresentados nesta seção, o computador do usuário deve estar conectado à rede *meca_robo* com a senha *meca_robo*.**

3.2.1 ssh

Secure Shell (SSH) é um protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura. O SSH fornece um canal seguro sobre uma rede insegura em uma arquitetura cliente-servidor, conectando uma aplicação cliente SSH com um servidor SSH. Aplicações comuns incluem login em linha de comando remoto e execução remota de comandos.

Sintaxe:
`ssh [opções] usuário@host`
Exemplo:
`ssh student@168.192.1.100`
Todos os RP4 tem um usuário "adp". A lista do IP da cada robô é a seguinte:

Computador	Usuário	IP	Senha
asterix	robot	192.168.1.102	Mec@tr0n
obelix	robot	192.168.1.100	Mec@tr0n
idefix	robot		Mec@tr0n
panoramix	robot		Mec@tr0n

3.2.2 scp

Secure Copy, em Português cópia segura, ou simplesmente SCP, é um meio seguro de transferência de arquivos entre um servidor local e um remoto ou entre dois servidores remotos, usando o protocolo SSH.
Tipicamente a sintaxe do programa `scp` é parecida com a sintaxe do `cp`:
`scp /ArquivoFonte usuário@host:/diretório/ArquivoAlvo`
`scp usuário@host:/diretório/ArquivoFonte /ArquivoAlvo`
Usamos a opção `-r` para copiar/colar uma pasta.

HARDWARE

Conectar o Raspberry Pi à bateria 1 (ou fonte de alimentação)

ROS

Criar um `workspace` *robot_ws*
Criar um `package` *drivers*
Criar um `package` *maze*

3.3 Pinos GPIO

O RP4 possui pinos GPIO (General Purpose Input Output), portas programáveis de entrada e saída de dados, para permitir interação e o controle de LEDs, interruptores, sinais analógicos, sensores e outros dispositivos. Como os pinos GPIO não tem função definida e por padrão não são usadas, é possível usar a biblioteca RPi para programar os pinos, cujo mapa é mostrado na Fig. 6.

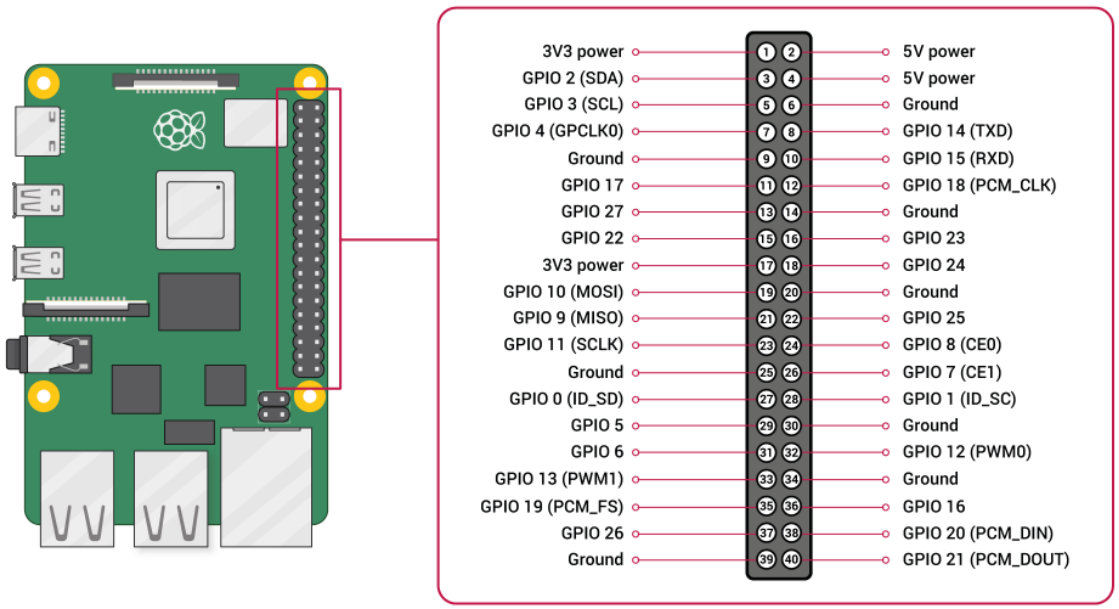

Figura 6: Pinos GPIO

3.3.1 Output

Os pinos GPIO podem ser usados como saída lógica: baixo (0 V) ou alto (5 V).

Listing 1: Output

```
#!/usr/bin/env python3

# To import the RPi.GPIO module
import RPi.GPIO as GPIO

# Set up the board layout
GPIO.setmode(GPIO.BCM)

# Set up pin 24 as output
GPIO.setup(24,GPIO.OUT)

# Turn OFF pin 24
GPIO.output(24,GPIO.LOW)

# Turn ON pin 24
GPIO.output(24,GPIO.HIGH)

# Cleanup at the end of the program
GPIO.cleanup()
```

3.3.2 PWM

Pulse Width Modulation, ou PWM, é uma técnica para obter resultados analógicos com meios digitais. O controle digital é usado para criar uma onda quadrada, um sinal alternado entre ligado e desligado. Este padrão on-off pode simular tensões entre o Vcc total da placa (por exemplo, 5 V) e desligado (0 V), alterando a parte do tempo que o sinal fica ligado versus o tempo que o sinal fica desligado. A duração do "on time"é

chamada de largura de pulso ou 'Pulse Width'. Para obter valores analógicos variados, você altera ou modula essa largura de pulso. Se você repetir esse padrão liga-desliga, o resultado será como se o sinal fosse uma tensão constante entre 0 e Vcc (Fig. 7).

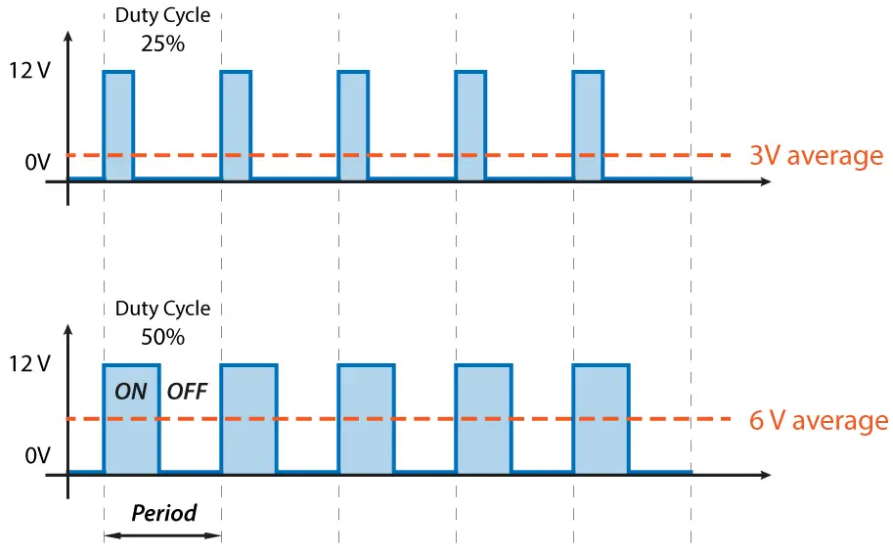

Figura 7: Pulse With Modulation

Listing 2: PWM

```
#!/usr/bin/env python3

# To import the RPi.GPIO module
import RPi.GPIO as GPIO

# Set up the board layout
GPIO.setmode(GPIO.BCM)

# Set up pin 25 as PWM with a 1000 Hz frequency
pwm25=GPIO.PWM(25,1000)

# Start with a duty cycle of 50%
pwm25.start(50)

# Change the frequency
pwm25.ChangeFrequency(500)

# Change the duty cycle
pwm25.ChangeDutyCycle(100)

# Stop
pwm25.stop()

# Cleanup at the end of the program
GPIO.cleanup()
```

3.3.3 Input

Existem várias maneiras de obter entrada GPIO em seu programa. A primeira e mais simples maneira é verificar o valor de entrada em um ponto no tempo. Isso é conhecido como 'polling' e pode potencialmente perder uma entrada se o seu programa ler o valor na hora errada. O polling é executado em loops e pode

consumir muito do processador. A outra maneira de responder a uma entrada GPIO é usando 'interrupções' (detecção de borda). Uma borda é o nome de uma transição de HIGH para LOW (borda descendente) ou LOW para HIGH (borda ascendente).
RPi.GPIO executa um segundo thread para funções callback. Isso significa que as funções callback podem ser executadas ao mesmo tempo que seu programa principal, em resposta imediata a uma borda.

Listing 3: Input

```
#!/usr/bin/env python3

# To import the RPi.GPIO module
import RPi.GPIO as GPIO

# Definition of the callback function for event detection
def cb_function(channel):
    print("Hello World")


GPIO.setmode(GPIO.BCM)

# Add event detection for pin 10
GPIO.add_event_detect(10,GPIO.RISING,callback=cb_function)

# Remove event detection for pin 10
GPIO.remove_event_detect(channel)
```

4 Motores DC e Codificadores

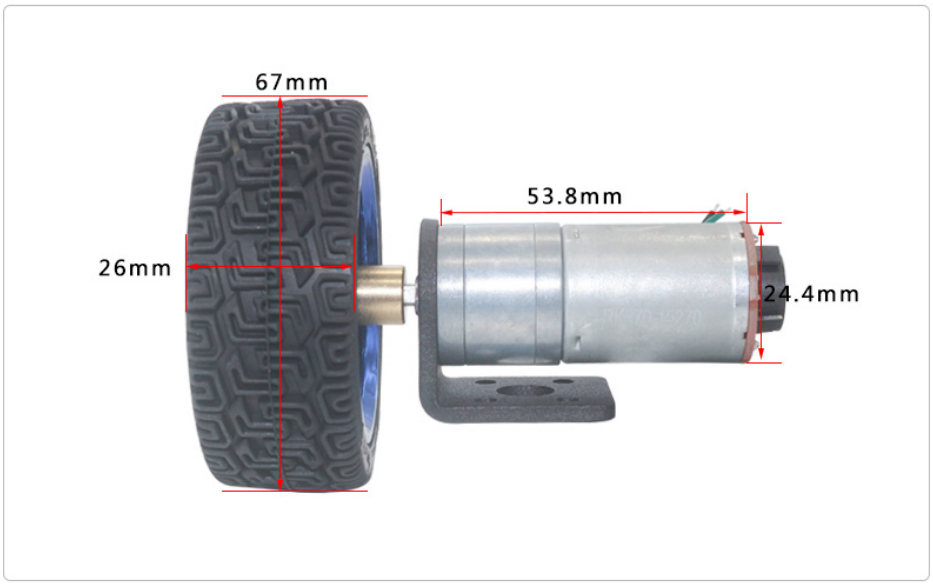

Figura 8: Roda, motor, caixa de redução e codificador

O robô é equipado com rodas acionadas por motores DC acoplados a caixas de redução, com mostrado na Fig. 8. As características dos motores, das caixas de redução e dos codificadores são dadas na Tab. 1.
O código de cores dos fios do par motor-codificador é dado na Tab. 2

Tensão maximal no motor	6V
Velocidade maximal do motor	130 RPM
Razão de redução da caixa	46

Tabela 1: Características dos motores com caixa de redução

Vermelho	Fonte de alimentação positiva do motor(+)
Branco	Fonte de alimentação negativa do motor(-)
Amarelo	Sinal feedback (uma volta do motor tem 11 sinais)
Verde	Sinal feedback (uma volta do motor tem 11 sinais)
Azul	Fonte de alimentação positiva do codificador (+)(3.3-5v)
Preto	Fonte de alimentação negativa do codificador (-)(3.3-5v)

Tabela 2: Código de cores dos fios do par motor-codificador

HARDWARE

O computador deve estar desligado
Conectar os cabos de feedback dos codificadores ao Raspberry Pi (usar pinhos com uma única função)
Conectar os cabos de alimentação dos codificadores ao Raspberry Pi
Verificar as conexões com o professor

ROS

Nome: Codificador
Entrada: -
Saída: Publisher (Float32MultiArray)
Descrição: Conta o número de sinais de cada codificador e publica os resultados

5 Ponte H

O chip L298N é um circuito integrado da STMicroelectronics que contém principalmente:

- 2 pontes H, cada uma permitindo acionar 1 motor elétrico DC (em uma direção ou outra)
- Uma lógica de controle de "baixa corrente", para conduzir essas pontes de "alta corrente"

Resumindo, o L298N permiti o controle direto de dois motores elétricos, por meio de controles lógicos de "baixa potência".
O L298N requer 2 fontes de alimentação separadas para operar:

- Uma tensão para a parte de potência, que será utilizada para alimentar os motores, através de transistores de potência
- Uma tensão para a parte de controle, que será usada para alimentar toda a parte lógica de controle, incluindo esses transistores de potência

No nível de controle lógico, distinguimos:

- Pinos de acionamento da ponte (ENA e ENB), que permitem a partida ou parada dos motores. Observe que essas entradas podem ser alimentadas em tudo ou nada (assim os motores irão "girar" na velocidade máxima), ou em PWM, para controlar sua velocidade de rotação

- Pinos de seleção de ponte (IN1, IN2, IN3 e IN4), que permitem selecionar os sentidos de rotação dos motores (e como aqui existem dois motores controláveis, existem 4 entradas, correspondentes às 4 possibilidades de sentido de rotação)

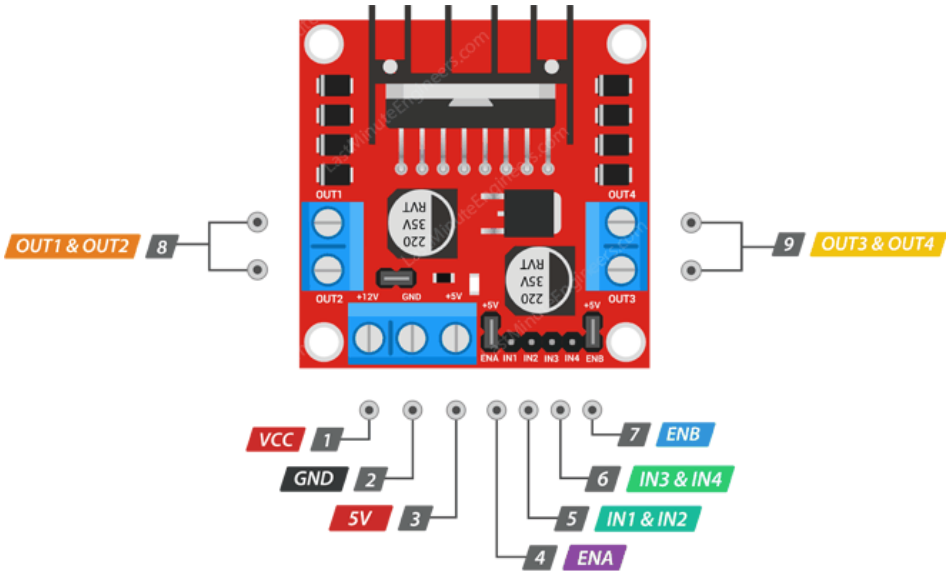

Figura 9: Ponte H L298N

1	VCC	Fonte de alimentação positiva da placa (+)
2	GND	Fonte de alimentação negativa da placa (-)
3	5V	Saída de 5V
4	ENA	PWM do motor A
5	IN1	Sentido de rotação do motor A
5	IN2	Sentido de rotação do motor A
6	IN3	Sentido de rotação do motor B
6	IN4	Sentido de rotação do motor B
7	ENB	PWM do motor B
8	OUT1	Alimentação negativa do motor A (-)
8	OUT2	Alimentação positiva do motor A (+)
9	OUT3	Alimentação positiva do motor B (+)
9	OUT4	Alimentação negativa do motor B (-)

Tabela 3: Entradas-saídas do L298N

HARDWARE

O computador deve estar desligado
Conectar os cabos de alimentação dos motores à placa L298N
Conectar as entradas logicas ao Raspberry Pi
Conectar o GND da placa L298N ao GND do Raspberry Pi
Conectar as fontes de alimentação da placa L298N à **bateria 2**
Verificar as conexões com o professor

ROS

Nome: Base_controller
Entrada: Subscriber (Float32MultiArray) - Subscriber (Twist)
Saída:
Descrição: 1: Calcula a velocidade angular dos motores - 2: Calcula a velocidade angular desejada - 3: Calcula o sinal PWM de cada motor usando um controlador PID - 4: Envia os sinais elétricos à placa ponte H

6 Servomotor

Figura 10: Servomotor TowerPro MG996R

O robô está equipado com um servomotor TowerPro MG996R que permite orientar a câmera. Estas características e o código colorido dos fios são dados nas Tab. 4 e 5

Tensão de operação	4,8 a 6,0 V
Tipo de Engrenagem	Metálica
Modulação	Digital
Velocidade de operação	0,19 seg/60 graus (4,8 V sem carga)
Velocidade de operação	0,15 seg/60 graus (6 V sem carga)
Torque (stall)	9,4 kgf.cm (4,8 V) e 11,0 kgf.cm (6 V)

Tabela 4: Código de cores dos fios do par motor-codificador

Os servos são controlados pela largura do pulso, a largura do pulso determina o ângulo. Uma largura de pulso de 1500 μs move o servo para o ângulo 0. Cada aumento de 10 μs na largura de pulso normalmente move o servo 1 grau mais no sentido horário. Cada diminuição de 10 μs na largura de pulso normalmente move o servo 1 grau mais no sentido anti-horário. Servos normalmente recebem 50 pulsos por segundo (50 Hz). Alguns cálculos em 50 Hz para larguras de pulso de amostra.

Marão	Fonte de alimentação negativa do motor(-) (0 V)
Vermelho	Fonte de alimentação positiva do motor(+) (5 V)
Amarelo	Sinal de controle PWM

Tabela 5: Código de cores dos fios do par motor-codificador

$$500/20000 = 0.025 \text{ ou } 2.5\ \% \text{ dutycycle}$$

$$1000/20000 = 0.05 \text{ ou } 5.0\ \% \text{ dutycycle}$$

$$1500/20000 = 0.075 \text{ ou } 7.5\ \% \text{ dutycycle}$$

$$2000/20000 = 0.1 \text{ ou } 10.0\ \% \text{ dutycycle}$$

$$2500/20000 = 0.125 \text{ ou } 12.5\ \% \text{ dutycycle}$$

HARDWARE

O computador deve estar desligado
A câmera deve estar desconectada
Conectar os cabos de alimentação do servomotor à placa L298N
Conectar a entrada logica ao Raspberry Pi
Verificar as conexões com o professor

ROS

Nome: Servo_controller
Entrada: ServiceRequest (JointTrajectory)
Saída: ServiceResponse (livre)
Descrição: O nó deve oferecer um serviço de posicionamento da câmera

7 Lidar

Figura 11: Lidar YDlidar G2

Ranging frequency	5000 Hz
Motor frequency	5 − > 12 Hz
Ranging distance	0.12 - 16 m
Field of view	0-360 Deg
Systematic error	2 cm
Luminous intensity range	0 - 1023
Angle resolution	0.36 − > 0.864 Deg

Tabela 6: Código de cores dos fios do par motor-codificador

O primeiro sensor exteroceptivo a bordo do robô é um lidar, modelo YDlidar G2. Estas características são dados na Tab. 6.
G2 define internamente um sistema de coordenadas polares. As coordenadas polares do sistema consideram o centro do núcleo rotativo de G2 como o pólo, e o ângulo especificado é positivo no sentido horário (vista superior). O ângulo zero está localizado na direção da saída da linha de interface G2 PH2.0-5P.

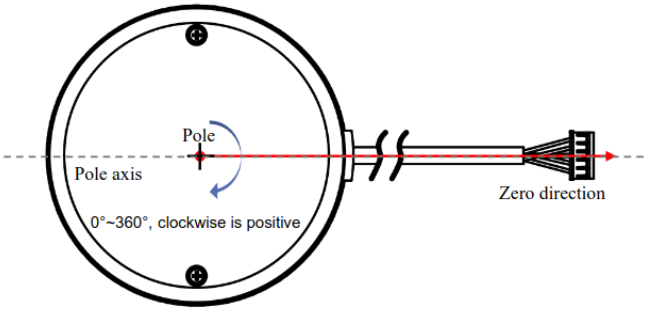

Figura 12: Lidar YDlidar G2

Um exemplo de código Python que permite trabalhar com o Lidar está disponível em 4.

Listing 4: Lidar

```
import os
import ydlidar
import time

if __name__ == "__main__":
    ydlidar.os_init();
    ports = ydlidar.lidarPortList();
    port = "/dev/ydlidar";
    for key, value in ports.items():
        port = value;
        print(port);
    laser = ydlidar.CYdLidar();
    laser.setlidaropt(ydlidar.LidarPropSerialPort, port);
    laser.setlidaropt(ydlidar.LidarPropSerialBaudrate, 115200);
    laser.setlidaropt(ydlidar.LidarPropLidarType, ydlidar.TYPE_TRIANGLE);
    laser.setlidaropt(ydlidar.LidarPropDeviceType, ydlidar.
   YDLIDAR_TYPE_SERIAL);
    laser.setlidaropt(ydlidar.LidarPropScanFrequency, 10.0);
    laser.setlidaropt(ydlidar.LidarPropSampleRate, 3);
    laser.setlidaropt(ydlidar.LidarPropSingleChannel, True);
    laser.setlidaropt(ydlidar.LidarPropMaxAngle, 180.0);
    laser.setlidaropt(ydlidar.LidarPropMinAngle, -180.0);
```

```
    laser.setlidaropt(ydlidar.LidarPropMaxRange, 16.0);
    laser.setlidaropt(ydlidar.LidarPropMinRange, 0.08);
    laser.setlidaropt(ydlidar.LidarPropIntenstiy, False);

    ret = laser.initialize();
    if ret:
        ret = laser.turnOn();
        scan = ydlidar.LaserScan();
        while ret and ydlidar.os_isOk() :
            r = laser.doProcessSimple(scan);
            if r:
                print("Scan received[",scan.stamp,"]:",scan.points.size(),
"ranges is [",1.0/scan.config.scan_time,"]Hz");
            else :
                print("Failed to get Lidar Data")
            time.sleep(0.05);
        laser.turnOff();
    laser.disconnecting();
```

HARDWARE

O computador deve estar desligado
Conectar o cabo de comunicação ao Raspberry Pi (USB 3)
Conectar a alimentação do Lidar à bateria 1
Verificar as conexões com o professor

ROS

Nome: Lidar
Entrada:
Saída: Publisher (LaserScan)
Descrição: Publica os dados do lidar

8 Câmera

O segundo sensor exteroceptivo a bordo do robô é uma câmera, modelo Raspberry Pi module V2. Essas características são dados na Tab. 7.

Figura 13: Raspberry Pi Camera Module v2

Size	25x24x9 mm
Weight	3g
Resolution	8 Megapixels
Sensor	Sony IMX219
Sensor resolution	3280 x 2464 pixels
Sensor image area	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.12 μm x 1.12 μm
Optical size	1/4”
Focus	Adjustable
Depth of field	10 cm to inf
Focal length	3.04 mm
Horizontal Field of View (FoV)	62.2 degrees
Vertical Field of View (FoV)	48.8 degrees
Focal ratio (F-Stop)	F2.0
Maximum exposure times (seconds)	11.76

Tabela 7: Parâmetros do Raspberry Pi Camera Module v2

Um exemplo de código Python que permite trabalhar com a câmera está disponível em 5.

Listing 5: Lidar

```python
import cv2

# open camera
cap = cv2.VideoCapture('/dev/video0', cv2.CAP_V4L)

# set dimensions
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 2560)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1440)

# take frame
ret, frame = cap.read()
# write frame to file
cv2.imwrite('image.jpg', frame)
# release camera
cap.release()
```

HARDWARE

O computador deve estar desligado
Conectar a câmera ao Raspberry Pi **Verificar as conexões com o professor**

ROS

Nome: Camera
Entrada: ServiceRequest (livre)
Saída: ServiceResponse (Image)
Descrição: O nó deve fornecer as imagens da câmera via um serviço