ASPIRADOR DE PÓ RESIDENCIAL AUTÔNOMO

Universidade Federal de Santa Catarina (UFSC) – Araranguá, SC, Brasil Centro de Ciências, Tecnologia e Saúde (CTS)
Departamento de Computação (DEC)
Disciplina: DEC7564 – Projeto de Sistemas Ubiquos **Turma: 08655**
Professor: Jim Lau
Alunos: Nathan Vieira Marcelino[1], Vinícius Camozzato Vaz[2], Willer Lucas Barata Castanheti[3]
Email-s: nathan.v.marcelino@gmail.com[1], vinicvaz95@gmail.com[2], luccasatarab@gmail.com[3]

Abstract. The autonomous vacuum cleaner has as goal automate the cleaning of residential environments like rooms, bedrooms, kitchens and others. The autonomous functions provides to the user the capacity of schedule a clean without the necessity of their presence, and also the remote access to the device.

Resumo. O aspirador de pó autônomo tem como objetivo automatizar a limpeza de ambientes residenciais internos como salas, quartos, cozinhas, entre outros. As funções de automação fornecem ao usuário a capacidade de agendar limpezas sem a necessidade de sua presença física no local, e também acesso remoto ao dispositivo.

Palavras-chave – Aspirador, raspberry pi, RBP3, Navegação, Controle, Módulos, Inteligência Artificial, Machine Learning.

INTRODUÇÃO

O sistema consiste em um aspirador de pó autônomo que utiliza sensores em sua estrutura e algoritmos de controle e inteligência artificial para explorar seu ambiente de trabalho e realizar um mapeamento do mesmo assim otimizando rotas e adotando- as como melhores opções futuras. O computador de placa única Raspberry Pi 3 Model B tem o papel de realizar o processamento da navegação, ou seja, realizar o controle do aparelho definindo velocidades, direções e melhores rotas além disso também tem como tarefa realizar a comunicação do sistema com dispositivos externos.

1. DESCRIÇÃO FUNCIONAL

Descreveremos nesta seção o funcionamento do sistema e como ele se comporta na interação com o ambiente externo. As funcionalidades

serão exemplificadas e o modo que o mesmo responde aos eventos que podem ocorrer durante seu funcionamento. Os requisitos funcionais, não funcionais e as regras de negócio que demandam o sistema servirão de auxílio para melhor entendimento das necessidades e capacidades do aparelho.

1.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais definem as funções que o aparelho é capaz de realizar, ou seja, sua capacidade de realizar tarefas e serviços.

- RF01: O sistema deve realizar a limpeza da área de atuação
- RF02: O sistema deve realizar navegação inteligente
- RF03: O sistema deve otimizar o consumo de energia de maneira autônoma
- RF05: O sistema deve permitir agendamento eletrônico para realizar limpeza
- RF06: O sistema deve realizar mapeamento da região de maneira satisfatória

1.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais estão relacionados ao uso do aparelho, coisas como usabilidade, confiabilidade, desempenho, temos então:

- RNF01: O sistema deve ser de fácil uso (intuitivo
- RNF02: O sistema deve maximizar o mapeamento de maneira a minimizar pontos cegos
- RNF03: O sistema deve ter segurança no acesso remoto de maneira que seja controlável apenas por usuários autorizados

1.3 REGRAS DE NEGÓCIO

As regras de negócio refletem a lógica que guia o comportamento do sistema. Entre elas temos:

- RN01: Somente pessoas autorizadas devem ter acesso ao acesso remoto
- RN02: Se houver divergência nos mapeamentos da área de atuação, o mesmo deverá ser atualizado

2. MATERIAL UTILIZADO

1) 2 motores de corrente continua para navegação
2) 1 motor de corrente continua para aspiração
3) Ventoinha

4) Filtro para aspirador
5) Módulo de motor Ponte-H – L298N
6) Sensorde velocidade encoder
7) Módulo MPU6050 – Giroscópio e Acelerometro
8) Rodas
9) Raspberry PI 3
10) Arduino Nano ATMEGA 328
11) Bateria 9.6V - 700mA para os motores
12) Bateria 5V - 2.5A para alimentação da placa RBP3
13) Sensor Ultrassônico de distância
14) Ferramentas de limpeza

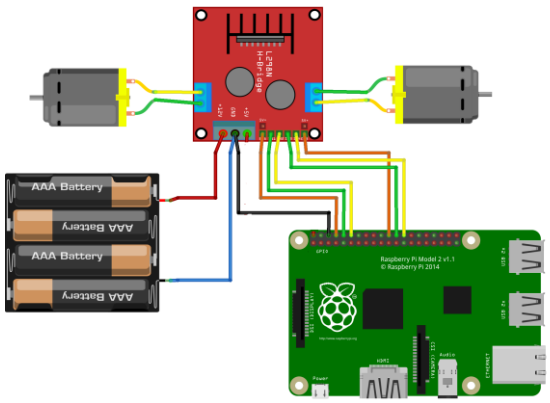
3. DESCRIÇÃO TÉCNICA E FUNCIONAMENTO

O primeiro desafio no desenvolvimento foi a montagem de um protótipo para realizar os testes de navegação. Para a montagem foi necessária a conexão dos seguintes componentes.

1) Ponte-H
2) Encoders
3) Motores
4) Raspberry Pi
5) Bateria para os motores
6) Giroscópio (*)
7) Sensor Ultrassônico

3.1 Motores

A ligação dos dois motores com o driver Ponte-H foi realizada como mostra a figura abaixo.



- A conexão com os pinos GPIO da placa não estão necessariamente condizentes com os utilizados no projeto.

A positivo da bateria é conectada ao VCC do módulo e o Ground do Rasp e da bateria ao GND do módulo. Além disso conectamos os pinos GPIO do raspberry pi nos pinos de “en” e “in” do módulo. Os pinos utilizados no primeiro protótipo foram os seguintes:

O número dos pinos da placa Raspberry Pi presentes na tabela abaixo está no padrão

BCM.

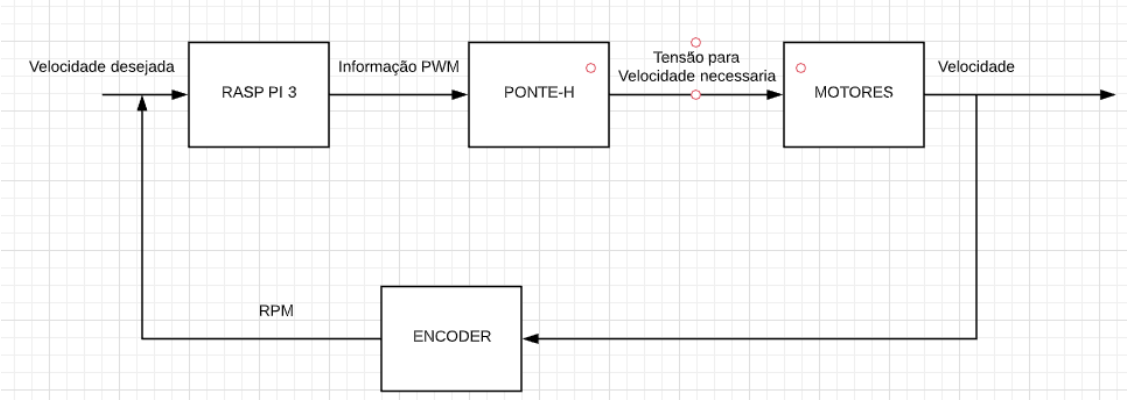
MOTOR 1	MOTOR 2
EN = 25	EN = 6
IN1 = 24	IN1 = 16
IN2 = 23	IN2 = 26

3.2 Encoders

Além da conexão dos motores com o módulo PWM e a placa, também foi necessária a conexão dos sensores de velocidade (encoder). Para aquisição do sinal do sensor utilizamos o pino digital de saída do mesmo. O VCC dos encoders foi conectado ao 5V da placa, o GND ao GND da placa e o pino de dados às GPIO da Rasp. As GPIO utilizadas foram as seguintes:

- Encoder 1: Pino 4 (BCM)
- Encoder 2: Pino 17 (BCM)

Esses sensores foram utilizados para medir o RPM de cada roda do robô e então realizar um controle de velocidade dos motores tentando garantir a mesma velocidade nos dois motores. Isso é feito através de um algoritmo de controle PID.



- **Diagrama PID encoders**

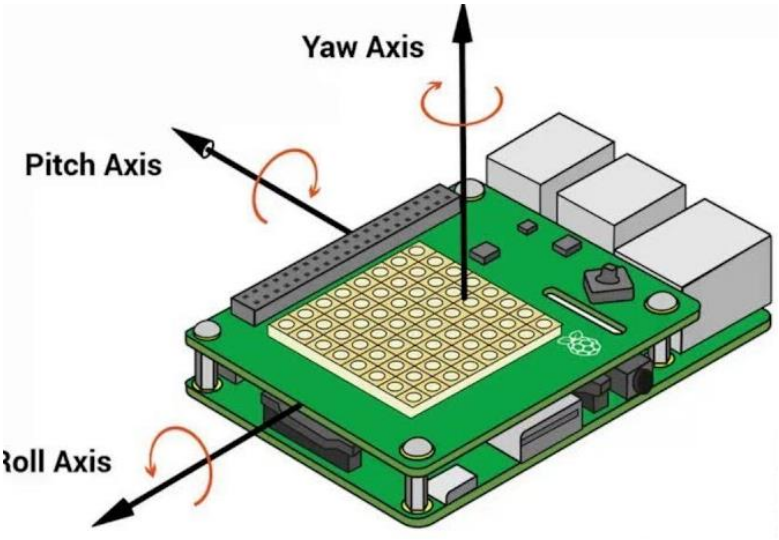
3.3 MPU6050 (Giroscópio e Acelerômetro)

A implementação desse algoritmo apenas seria suficiente caso não houvesse a necessidade de mapeamento da região, porém como necessitamos de precisão de posição e rotação do robô agregamos ao protótipo um giroscópio e um acelerômetro, ambos incluídos no módulo MPU6050.

Com esses sensores implementamos uma IMU (Intertial Measurement Unit ou Unidade de Medidas Inerciais) que consiste no dispositivo presente no protótipo que informa para o controlador a orientação do objeto em termos de rotação em relação aos eixos x, y e z. Além disso é possível obter a aceleração do objeto em função do tempo.

O módulo MPU6050 utiliza protocolo I2C para se comunicar com a placa. Em um primeiro momento estávamos implementando a IMU também no Raspberry, porém optamos por transforma-la em um componente a parte devido sua importância de precisão e poderiam haver problemas de interferência de frequências que comprometeriam os dados lidos. Sendo assim utilizamos um Arduino Nano para implementar a IMU e passamos os dados pela porta serial do Raspberry Pi.

- **Pitch: Eixo X**
- **Roll: Eixo Y**
- **Yaw: Eixo Z**



IMU

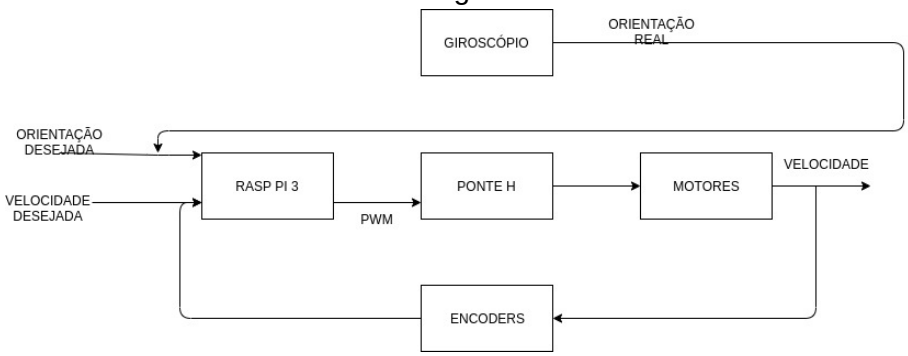
Implementamos a IMU em relação aos 3 eixos de orientação, porém descreveremos aqui apenas a orientação de rotação em relação ao eixo Z uma vez que nosso mapeamento ocorrerá em um plano 2D e o dispositivo não terá inclinação significativa em outras direções.

Para implementar o controle de orientação em relação ao eixo Z é necessário entender o funcionamento do giroscópio presente no MPU6050. O mesmo nos fornece dados brutos em graus/segundo. Para obtermos a rotação de um corpo basta integrar no tempo a rotação.

$$\alpha = \int_{t0}^{t} \boldsymbol{\omega dt}$$

Onde:
α = Ângulo rotacionado
ω = Velocidade angular

Obtendo o valor do ângulo podemos implementar um algoritmo de controle mais completo onde um motor irá ditar a velocidade do robô e o outro irá definir a rotação. A figura abaixo mostra o funcionamento do algoritmo.



- **PID Completo**

O algoritmo funciona de maneira onde temos um **target** de RPM fixo para um motor (motor de controle de velocidade), o encoder desse motor irá medir o RPM real e passar essa informação ao Rasp que irá calcular o erro e tentar corrigir essa diferença. Além do target de RPM teremos também um **target** de ângulo variável de acordo com a direção desejada e um **target** de RPM variável para o outro motor. Se o ângulo real medido pelo giroscópio for diferente do ângulo alvo o **target RPM** desse motor irá ser mudado e consequentemente sua velocidade fazendo com que o robô corrija a posição.

3.4 Sensor Ultrassônico

O sensor ultrassônico possui 4 pinos. Sendo eles: VCC, GND, TRIG, ECHO. O pino Trig é responsável por emitir o sinal de ultrassom e o pino Echo por receber o sinal de volta. Os pinos utilizados para conectar o sensor foram:

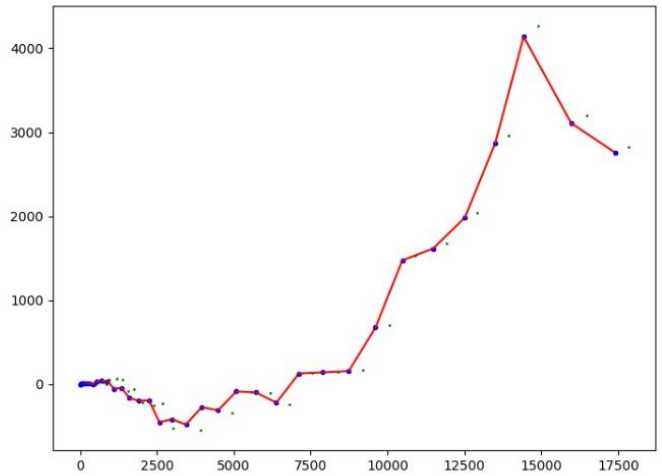
- VCC – 5V Raspberry
- GND – GND Raspberry
- TRIG – 27 (BCM)
- ECHO – 22 (BCM)

Com o sensor ultrassônico instalado implementamos um algoritmo que torna possível medir a distância de objetos à frente do sensor para evitar estes pontos.

3.4 Algoritmo de Posição e Mapeamento

Para realizar o mapeamento da área é necessário que o robô tenha bem definido sua posição. Para isso necessitamos da sua aceleração, obtida através do MPU6050. É possível integrar a aceleração no tempo para obter a velocidade e integrar a velocidade para obter então a posição.

Implementamos um algoritmo que realiza as integrais para obter os dados necessários e os decompõe nas componentes cartesianas X e Y utilizando o ângulo medido pelo giroscópio e então plotamos em um gráfico. Além da posição do robô, plotamos os pontos onde o sensor ultrassônico detectou algum objeto. Para teste do algoritmo utilizamos valores randômicos de aceleração, ângulo e distância. O gráfico abaixo mostra o teste de uma amostra de dados.



- Trajeto percorrido
- Ponto de atualização
- Objeto detectado

A frequência de atualização dos dados utilizada no teste foi de 1Hz, porém no dispositivo utilizamos a frequência de 250 Hz para leitura do MPU6050.

4. COMUNICAÇÃO E PROTOCOLOS

Como o protótipo é um dispositivo autônomo, não há a necessidade de comunicação com dispositivos externos, porém para facilitar a implementação dos códigos utilizamos o protocolo SSH para conectar o notebook com sistema operacional do Rasp e manipular os dados presentes nele. Os dados gerados pelo giroscópio são transmitidos ao Raspberry através da porta serial. No momento está sendo implementada uma comunicação entre o Raspberry e o computador via web-socket para enviar esses dados tornando possível a análise gráfica da IMU utilizando o software Processing.