CENTRO PAULA SOUZA

FATEC SANTO ANDRÉ

Tecnólogo em Mecatrônica Industrial

GABRIEL VIEIRA DA COSTA

SAMUEL PAULO DE OLIVEIRA

WILTON CLAUDINO MIRANDA

VEÍCULO CONTROLADO POR LUVA

SANTO ANDRÉ

2023

GABRIEL VIEIRA DA COSTA

SAMUEL PAULO DE OLIVEIRA

WILTON CLAUDINO MIRANDA

VEÍCULO CONTROLADO POR LUVA

Trabalho de Conclusão de Curso apresentado ao curso de Tecnólogo em Mecatrônica Industrial da FATEC – Santo André orientado pelo Professor Me. Luiz Vasco Puglia como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial.

SANTO ANDRÉ

2023

FICHA CATALOGRÁFICA

C837v

Costa, Gabriel Vieira da

Veículo controlado por luva / Gabriel Vieira da Costa, Samuel Paulo de Oliveira, Wilton Claudino Miranda. - Santo André, 2023. – 74f: il.

Trabalho de Conclusão de Curso – FATEC Santo André. Curso de Tecnologia em Mecatrônica Industrial, 2023.

Orientador: Prof. Me. Luiz Vasco Puglia

1. Mecatrônica. 2. Sensores. 3. Sensor acelerômetro. 4. Automodelo. 5. Tecnologia. 6. Veículo. 7. Brinquedo. 8. Projeto. 9. Microcontroladores. 10. Controlado por luva I. Oliveira, Samuel Paulo de. II. Miranda, Wilton Claudino. III. Veículo controlado por luva.

629.892

LISTA DE PRESENÇA

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 21 DE JUNHO DE 2023.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"CARRO CONTROLADO POR LUVA"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. FLAVIO AUGUSTO BARRELLA _____

MEMBROS:
PROF. WELLINGTON BATISTA DE SOUSA _____

PROF. LUIZ VASCO PUGLIA _____

ALUNOS:

GABRIEL VIEIRA DA COSTA _____

SAMUEL PAULO DE OLIVEIRA _____

WILTON CLAUDINO MIRANDA _____

Dedicamos esse trabalho aos nossos pais e familiares que nos incentivaram e nos apoiaram durante todo nosso curso.

AGRADECIMENTO

Agradecemos aos professores da Fatec – Santo André e amigos e familiares que nos auxiliaram, incentivaram e contribuíram para o desenvolvimento do trabalho. Foi um longo período de estudo e muitos esforços foram necessários, agradecemos a cada pessoa que diretamente ou indiretamente contribuiu para nossa formação nesse período.

O sucesso é a soma de pequenos esforços - repetidos dia sim, e no outro dia também.

Robert Collier

RESUMO

A engenharia moderna, permite controlar sistemas a distância ou remotamente. Esses controles encontram diversas aplicações no mundo industrial ou de segurança do homem. Por exemplo, robôs para desarme de bombas que protegem o homem do evento de explosão. Como já é de grande fama entre as crianças, um carrinho de brinquedo comum geralmente é controlado por um controle remoto, ou de forma manual pelo indivíduo. O projeto consiste em um automodelo de brinquedo que é controlado através de uma luva, que detecta os movimentos da mão e dos dedos, produzindo um movimento no mesmo. Para isso, foi utilizado um recurso moderno e bastante econômico, denominados acelerômetros. Acelerômetros são sensores que disponibilizam sinais de posição e aceleração em três eixos cartesianos, ele detecta variação de aceleração e velocidade angular, emitindo pulsos elétricos que são proporcionais a estas variações. Este trabalho propõe a utilização desse acelerômetro fixo a uma luva que quando vestida em uma mão humana, irá permitir o controle de um determinado dispositivo, em nosso caso o automodelo. Estes pulsos elétricos são enviados para o microcontrolador, que através de seu módulo de Bluetooth, envia estes pulsos para o automodelo, produzindo seu movimento. Os sensores detectam movimentos em três eixos, permitindo então o automodelo andar para frente, para trás e para os lados.

Palavras-chave: Acelerômetro; Automodelo; Sensores; Luva.

ABSTRACT

Modern engineering allows you to control systems remotely or remotely. These controls find many applications in the industrial or human security world. For example, bomb defusing robots that protect man from the explosion event. As it is already famous among children, a common toy car is usually controlled by a remote control, or manually by the individual. The project consists of a toy car that is controlled through a glove, which detects the movements of the hand and fingers, producing a movement in the car. For this, a modern and very economical resource, called accelerometers, was used. Accelerometers are sensors that provide position and acceleration signals in three Cartesian axes, it detects variations in acceleration and angular velocity, emitting electrical pulses that are proportional to these variations. This work proposes the use of this accelerometer attached to a glove that, when worn on a human hand, will allow the control of a certain device, in our case the cart. These electrical pulses are sent to the microcontroller, which, through its Bluetooth module, sends these pulses to the cart, producing its movement. The sensors detect movement in three axes, allowing the cart to move forwards, backwards and sideways.

Keywords: Accelerometer; Cart; Sensors; Glove.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

PIC	*Programmable Interface Controller*
I2C	*Inter-Integrated Circuit*
RS232	*Recommended Standard 232*
WiFi	*Wireless Fidelity*
V	*Voltage*
DC	*Direct Current*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
CPU	*Central Processing Unit*
RAM	*Random Access Memory*
VDC	*Voltage Direct Current*
IoT	*Internet of Things*
PWM	*Pulse Width Modulation*
GPIOs	*General Purpose Input/Output*
IDE	*Integrated Development Environment*

SUMÁRIO

1.INTRODUÇÃO

Na área da Tecnologia, a cada dia se vê novidades que proporcionam o aperfeiçoamento de projetos já existentes. O lançamento de novos recursos, permitem sistemas se tornarem mais rápidos e de serem controlados por uma distância ainda maior.

Há muito tempo os brinquedos movidos com controle remoto são muito famosos por proporcionarem momentos de diversão para as pessoas, sendo crianças ou até adultos. As inovações desse meio a cada dia estão crescendo mais, enquanto antes tínhamos carrinhos que não possuíam um alcance muito grande daquele que estivesse controlando, hoje temos drones que podem ser controlados e monitorados a quilômetros de distância.

Antigamente, o controle remoto era feito com antenas para que fosse feita uma comunicação em radiofrequência, com o transmissor enviando os sinais para o receptor, tendo um alcance muito limitado. Isso foi aperfeiçoado hoje graças a possibilidade desta comunicação ser feita com módulos *Bluetooth* ou *Wifi*, que permite uma transmissão de dados mais rápida e uma distância maior.

O carrinho controlado por uma luva, foi anteriormente criado por diversos apreciadores da robótica e instituições vinculadas a mecatrônica. Foi feito um modelo em que foi utilizado uma luva com um celular acoplado, e através do movimento que é detectado por um acelerômetro que a estrutura interna do próprio celular possui, é enviado um sinal para o Arduino através de um módulo para Bluetooth, realizando o movimento do carrinho. A comunicação do celular com o Arduino é feita com um aplicativo para Android disponível de forma gratuita, o *Robo Control Accelerometer*. O alcance máximo neste projeto seria de 14m sem obstáculos.

O projeto de controle de um carrinho remotamente controlado por uma luva sensoriada apresentado por (DE OLIVEIRA, 2015) utiliza uma luva, mas com o acelerômetro instalado separadamente, os microcontroladores utilizados foram o PIC18F4520 e o PIC18F2520, cada um possuindo uma função específica dentro do circuito, e a comunicação sendo feita com um módulo Wireless com rádio frequência. Isso permite que o alcance do carrinho seja maior do que o modelo anterior, e que a transferência de dados maior. Para o movimento retilíneo para frente e para trás, foi utilizado um módulo de ponte H, que permite a inversão da rotação do motor.

Já outro projeto de uma luva Háptica para controle de um manipulador apresentado por (SANTOS, 2015) utiliza o MPU6050 para captação dos movimentos realizados por quem utiliza a luva além de dois microprocessadores Arduino nano, para receber os sinais vindos do MPU6050 e através de um modulo RF transmissor + receptor de 433 MHz passa as informações para o segundo Arduino, o manipulador utilizado foi o kit roboTEK II, e esses sinais enviados pelo Arduino acionavam os servomotores presentes no manipulador.

Nesse projeto de implementação de uma luva microcontrolada para a captura de gestos, apresentado por (LOGARETTI, 2015) utiliza 6 sensores MPU6050 em que se comunicam pelo protocolo de comunicação I2C, para controle e comunicação entre os sensores foi utilizado o microcontrolador PAMPIUM, e para comunicação com o computador foi utilizada a interface RS232. A partir do MPU6050 os valores dos eixos x, y e z dos giroscópios e acelerômetros são armazenados e enviados digitalmente para o microcontrolador PAMPIUM. Assim, esses projetos aproximam-se muto do projeto aqui desenvolvido.

A ideia do nosso projeto foi criar um modelo diferente de controle para um automodelo, a nossa própria mão. Uma luva que possui um circuito que permite através da movimentação dela, enviar sinais e realizar o movimento. Assim podemos, por exemplo, movimentar a mão para cima em um determinado ângulo e o automodelo responder andando para frente, ou com o movimento para baixo o automodelo andar para trás. Isto é perfeitamente possível utilizando pequenos computadores denominados microcontroladores, que facilitam a transmissão de informações e a criação de sistemas embarcados.

O presente projeto visa uma melhora em relação aos projetos anteriores, foi utilizado o módulo MPU6050, que é um acelerômetro com giroscópio, medindo a variação de aceleração e velocidade angular, permite uma detecção mais precisa do movimento da mão. O microcontrolador utilizado foi o ESP32, ele já possui um módulo sem fio para *Bluetooth* e *Wifi*, para o nosso projeto foi usado o *Wifi*, sendo assim maior o alcance do automodelo e com velocidade de transferência de dados ainda maior.

Neste trabalho consta uma breve introdução, que apresenta o nosso projeto e projetos similares feitos anteriormente, que utilizamos como inspiração e base para a construção do nosso.

No desenvolvimento, uma fundamentação teórica é feita, demonstrando com detalhes cada componente utilizado, explicando de maneira objetiva o funcionamento de cada um deles, bem como explicar os conceitos necessários para o entendimento do projeto, o motivo de utilizarmos cada componente, e o custo.

Por fim, fizemos uma conclusão sobre tudo o que produzimos, e trazemos novas ideias para melhorias e aperfeiçoamentos futuros para o automodelo que nós fizemos.

1.1 Objetivo

Este trabalho tem por objetivo desenvolver uma variação do controle remoto de veículos, utilizando como meio as próprias mãos, partindo do uso de um acelerômetro e um microcontrolador para movimentar o veículo. O objetivo inicial é conseguir reproduzir os movimentos produzidos pela mão do usuário para o veículo, realizando as principais funções, como ir para frente, para trás, direita e esquerda. Dessa forma, permitir o controle remoto de veículos em que possa ser aplicado em diversas áreas, desde um brinquedo a controle real de veículos destinados ao mercado automobilístico ou a indústria. Desenvolvendo nossos conhecimentos e aplicando-os na área de estudo de nosso curso.

1.2 Motivação

Na área da tecnologia é comum vermos cada vez mais projetos de inovação e desenvolvimento, ampliando e melhorando cada dia mais nosso dia a dia. Porém com esse avanço é comum que antigos projetos ganhem vida novamente com atualizações e adições de novos componentes que permitem uma nova perspectiva sobre eles.

Este projeto destina-se a criar uma solução de controle de veículos a partir do uso apenas da mão, utilizando os movimentos da mão para controle do veículo, permitindo que possamos aplicar nossos conhecimentos adquiridos durante o curso, e adquirir novos durante todo o processo de desenvolvimento. Como futuros tecnólogos é nosso dever estarmos atentos as novas tecnologias e contribuirmos para a comunidade com um projeto. Com isso, poderemos evoluir e estar cada vez mais preparados para o mercado de trabalho e provar que estamos prontos para isso.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta etapa do trabalho escrito, é feita uma explicação dos principais conceitos técnicos que são necessários para o entendimento do funcionamento do nosso projeto, bem como os componentes utilizados, fundamentos dos circuitos eletrônicos e um breve histórico.

Após o surgimento da indústria automobilística, em meados do século XX, foram criadas as primeiras réplicas de carros com o intuito de entreter as crianças, eram feitos de madeira, plástico e metais, geralmente baseados nos grandes carros de luxo da época.

O aparecimento dos primeiros carrinhos de controle remoto no Brasil, aconteceu nos anos 80 pela marca Estrela, muita famosa até hoje pela produção de seus brinquedos, foi responsável depois pela fabricação de inúmeros modelos, alguns deles são: *Stratus*, *Pégasus*, *Colossus*, *Maximus*, como mostrado na Figura 1.

Figura 1 - Carrinho Stratus

Fonte: HOSPITAL DAS BONECAS, 2022

Para uma aproximação melhor dos carros reais, alguns destes brinquedos são feitos com motor a combustão, e tanque a gasolina, geralmente utilizados em competições. Por terem um custo muito alto, foram introduzidos no mercado carrinhos movidos a eletricidade, sendo mais acessíveis as crianças, por terem um baixo custo.

Nos dias de hoje já encontramos diversos outros veículos que podem ser controlados por um controle remoto, não se limitando apenas aos carros, como helicópteros, aviões, caminhões, e muito usado de hoje em dia, os drones.

2.1 Microcontroladores

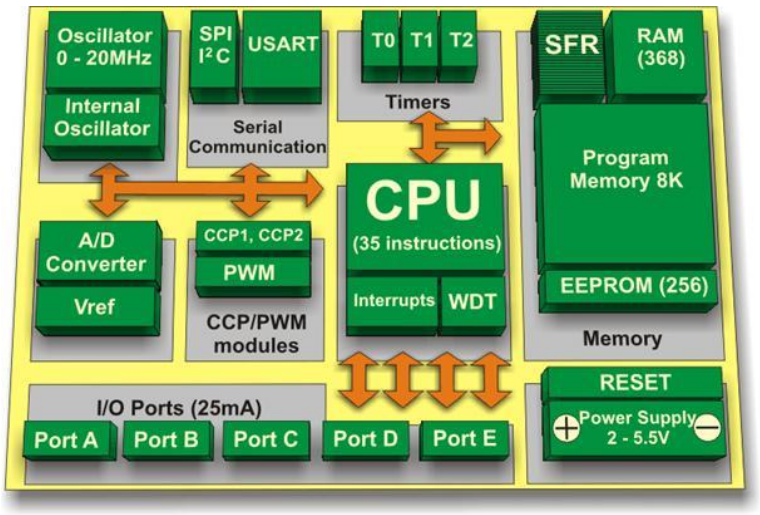
Para a realização de sistemas controlados, podem ser utilizados os microcontroladores, são computadores pequenos que realizam uma variedade de tarefas, eles possuem um circuito integrado com um núcleo de processor dentro dele, entradas e saídas digitais e analógicas, e memórias para o armazenamento de dados, a cada dia sendo eles com uma capacidade cada vez maior. Neles são feitas as comunicações, recebendo e enviando sinais, e o processamento deles.

Para o funcionamento de um microcontrolador, é necessário um programa gravado em sua memória e uma fonte de alimentação, que serão abordados mais adiante.

2.1.1 Arquitetura de um microcontrolador

Como ilustra a Figura 2 uma arquitetura genérica da maioria dos microcontroladores, com base no PIC16F887:

Figura 2 - Arquitetura



Fonte: SCRI Group, 2022

- A alimentação permitida, em sua maioria, está na faixa de 2 – 5.5V, assim como é visto na imagem, no bloco denominado *Power Supply*.

- Logo acima, temos o *Reset*, é responsável por reiniciar o programa interno, de acordo com uma tensão aplicada nesse pino.

- A Unidade Central de Processamento (CPU), é responsável pela execução do programa armazenado em sua memória, sendo assim, a parte mais importante do microcontrolador.

- A memória RAM, é um tipo de memória volátil, ou seja, quando é cortada a alimentação do circuito ela apaga o que está armazenado dentro dela, usada para guardar variáveis temporárias do programa.

- Já a memória Flash, não é volátil, sendo usada para armazenar o programa propriamente dito.

- A memória EEPROM, muito parecida com a Flash, porém com uma velocidade de escrita menor.

- A última parte principal de um microcontrolador são as entradas e saídas, que são usadas para alterar as variáveis presentes no programa e fornecer um valor nos pinos de saída.

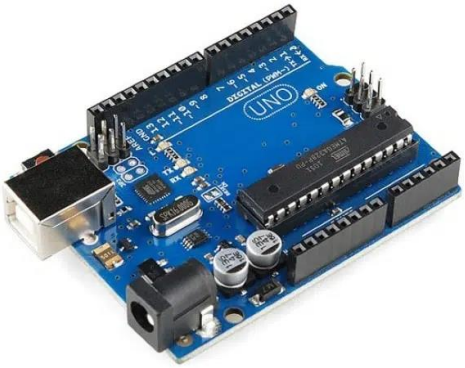
2.1.2 ATmega328 e ATmega2560

Este é um dos microcontroladores mais famosos no meio eletrônico, por sua ampla possibilidade de criação de projetos de diferentes tipos. Faz parte dos primeiros passos de muitos iniciantes na área da eletrônica e robótica, ele permite realizar projetos complexos de uma maneira simples, e um custo baixo, geralmente são incorporados nas placas Arduino.

Além de sua placa principal, o Arduino possui vários módulos que podem ser comprados separadamente e acrescentados, ampliando sua funcionalidade. A linguagem de programação que ele aceita é própria, mas baseada na Linguagem C/C++, por isso se tornou muito famoso, por proporcionar essa familiaridade com os desenvolvedores.

Um dois mais usados é o Arduino Uno como mostra na Figura 3, feito para projetos menores por possuir uma quantidade menor de entradas e saídas. São 14 pinos de I/O, sendo 6 analógicos e 6 *PWM (Pulse Width Modulation)*. Sua memória Flash é de 32KB e tem uma tensão de alimentação na faixa de 7 a 12 VCD.

Figura 3 – Arduino Uno



Fonte: Vida de Silício, 2021

Outro muito utilizado é o Arduino Mega 2560, como mostra na Figura 4, já possui uma grande quantidade de entradas e saídas, sendo possível uma aplicação em projetos maiores, como *IoT* e robôs. São 54 portas digitais e 16 analógicas, e com memória Flash de 256KB. Tensão de alimentação também na faixa de 7 a 12 VDC.

Figura 4 – Arduino Mega 2560

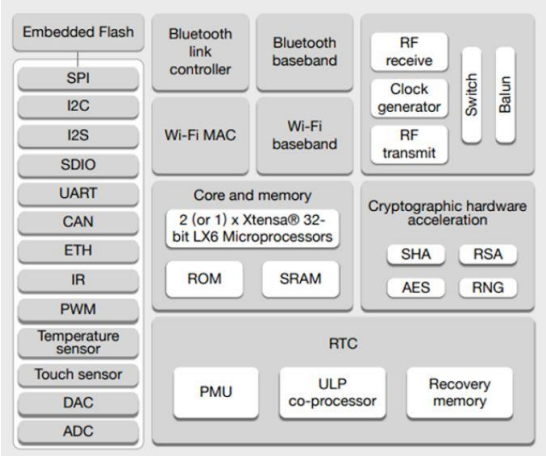
Fonte: EMBARCADOS, 2014

2.1.3 ESP-32

Lançado a não muito tempo, o ESP32 como mostra na Figura 5, tem sido de grande fama no meio eletrônico, um microcontrolador que possui uma variedade de funcionalidades que facilitam na construção de projetos, muito utilizado hoje para

aplicações de *IoT* (internet das coisas), como automação residencial. Ele é equipado com uma variedade de recursos, incluindo processador dual-core, WiFi integrado, Bluetooth, ADCs (Conversor Analógico Digital), DACs (Conversor Digital Analógico), PWMs (Modulação de Largura de Pulso), interrupções, GPIOs (Entrada/Saída de Propósito Geral) e muito mais. O ESP32 é capaz de executar tarefas avançadas e se comunicar com outros dispositivos por meio de *WiFi* ou *Bluetooth*, permitindo que ele seja usado em uma ampla variedade de projetos.

Figura 5 – Arquitetura ESP32



Fonte: DEINFO, 2019

O ESP32 é amplamente utilizado em projetos de IoT, incluindo sistemas de automação residencial, sistemas de monitoramento ambiental, dispositivos vestíveis, projetos de robótica e muito mais. Ele também é usado em projetos de prototipagem rápida, onde a simplicidade e o baixo custo são necessários. O ESP32 pode ser usado para desenvolver aplicativos e dispositivos que se conectam à Internet e se comunicam com outros dispositivos para trocar informações e executar tarefas.

Outra característica interessante no ESP32, é a função mestre-escravo, a qual refere-se a um modo de comunicação entre dispositivos, no qual um dispositivo atua como o mestre e controla a comunicação com um ou mais dispositivos escravos.

Quando o ESP32 opera em modo mestre, ele controla o fluxo de comunicação e inicia as transações com os dispositivos escravos. Ele é responsável por enviar comandos, solicitações de leitura de dados e receber as respostas dos dispositivos escravos.

Por outro lado, os dispositivos escravos agem como periféricos controlados pelo mestre. Eles respondem aos comandos e solicitações enviadas pelo mestre e fornecem os dados solicitados ou realizam as operações requisitadas.

2.2 Motores

Para a realização do movimento mecânico do automodelo, são necessários motores. São componentes que convertem energia elétrica em energia mecânica, eles podem ser encontrados em diversos equipamentos do nosso dia a dia, desde indústrias para a movimentação de robôs, eletrodomésticos, e como inspiração para o projeto, carros para a movimentação das rodas. Será utilizado nas rodas traseiras, que com engrenagens realizam o movimento giratório da roda, e servomotores nas rodas dianteiras, que farão elas girarem em um ângulo determinado para que o veículo possa se movimentar para os lados.

2.2.1 Motor de corrente contínua

Funciona com base na interação de dois imãs, um fixo e outro móvel. O estator é sua parte fixa, se comporta como um imã com dois polos nas suas extremidades. É adicionado um eletroímã no centro do motor, com enrolamentos, que quando tem a sua polarização invertida, é atraído pelas extremidades do motor, chamado de rotor. O que faz com que sua polaridade seja invertida é um elemento chamado comutador, um anel girante que é ligado com o rotor.

Seu funcionamento é baseado no princípio do eletromagnetismo, que diz que quando uma corrente percorre um fio, é formado em sua volta um campo magnético. Esse campo se comportará como um imã, e caso a corrente que percorre este fio for invertida, a polarização também inverte. Aqui entra o comutar, que será o responsável então com que a corrente será invertida várias vezes, para que o rotor tente se alinhar com o estator, que está fixo no motor, esse alinhamento faz o seu eixo do centro girar.

Este tipo de motor é muito usado por proporcionar a possibilidade de ajuste da sua velocidade, e inversão da sua rotação de uma maneira rápida.

2.2.2 Servo motor

Sua estrutura interior, se assemelha um pouco com a de um motor de corrente contínua, porém se comporta de uma maneira diferente. Ele não realiza um movimento giratório em 360 graus, mas faz movimentos rápidos de até 180 graus de um lado e de outro, com um torque muito elevado, permitindo um controle de posição angular. Em seu interior, ele possui um sensor de posição que comanda o que deve ser feito, distância, velocidade e ângulo.

Suas aplicações são inúmeras, pode ser utilizado em garras de robôs industriais, em automação para aberturas e fechamentos, e ajustes angulares, ou seja, tudo que exija movimentos precisos e rápidos é utilizado o servomotor. No nosso caso, foi usado para o movimento das rodas dianteiras, que permitirão que o carrinho possa se mover para os lados.

O servo motor utilizado neste projeto, é na verdade um micro servo motor como mostrado na figura 6, que é muito usado para aeromodelos e robôs, o Tower Pro 5G90, que funciona com uma tensão na faixa de 3V a 7V.

Figura 6 – Tower Pro 5G90

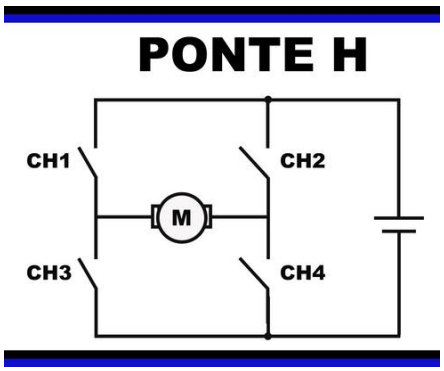
Fonte: MasterWalker, 2022

2.3 Ponte H

O movimento do carrinho se deve ao motor de corrente contínua que foi colocado nas rodas traseiras, mas o motor só gira em uma direção, o que permite o carrinho andar apenas para frente ou apenas para trás. Para que haja a liberdade de movimentação em qualquer sentido, é necessário no nosso circuito uma ponte H como

mostra a Figura 7, uma placa que geralmente é usada no Arduino que tem como função inverter o sentido de rotação do motor, assim como controlar a tensão que entra no motor.

Figura 7 – Ponte H



Fonte: Manual da Eletrônica, 2022

Ele recebe esse nome porque é arranjado em forma de H com quatro chaves, estando o motor no meio, se as chaves CH1 e CH4 forem acionadas, o motor girará em sentido horário, se as chaves CH2 e CH3 forem acionadas, o motor girará no sentido anti-horário. Outra condição posição também é realizar um freio no motor, que é feito acionando as chaves CH1 e CH2 ou CH3 e CH4. Caso sejam acionadas as quatro chaves, causará um curto-circuito podendo destruir o circuito. Para que o acionamento das chaves não seja feito de forma manual, alguns modelos utilizam transistores bipolares para que o acionamento seja eletrônico, de acordo com a corrente de base do transistor.

2.4 Acelerômetro

O acelerômetro é um dispositivo capaz de detectar qualquer aceleração ou vibração sofrida por sua estrutura e transformar essa informação em um sinal elétrico. São dispositivos que podem funcionar a partir de diversos efeitos físicos, contudo, dois tipos são os mais frequentemente utilizados: o acelerômetro de material piezoelétrico e o acelerômetro capacitivo. Normalmente, para cada elemento a ser monitorado, são utilizados três sensores em conjunto, um para cada dimensão espacial (eixos x, y e z). (Andrade, 2016).

2.4.1 Tipos de Acelerômetros

Existem muitos tipos de acelerômetros que usam diferentes tipos de efeitos físicos para medir a aceleração. Dentre eles os principais:

2.4.2 Piezoelétricos

O material piezoelétrico quando sofre o esforço de compressão, instantaneamente é gerada uma voltagem elétrica. Conforme vai aumentando a compressão, maior a voltagem gerada. Através dessa característica é possível fazer a medição da aceleração.

2.4.3 Piezoresistivos

Muito próximo do funcionamento do acelerômetro piezoelétrico, este opera através da variação na resistência conforme o movimento gerado, dessa forma é possível deduzir a aceleração. Um dos fatores para não utilização desses sensores, é o fato do grande consumo de energia, além de não suportarem altas temperaturas e tem alta detecção de ruídos.

2.4.4 Capacitivos

Utilizado para medir tanto aceleração estática como aceleração dinâmica, o sensor converte essa aceleração em correntes elétricas ou voltagem. É formado por um oscilador ou outro dispositivo estacionário capaz de armazenar capacitância. Quando o equipamento se move, é detectada pelos sensores, que estando conectados ao circuito elétrico é capaz de mensurar a intensidade da aceleração com relação a corrente elétrica.

2.4.5 Eixo único

Acelerômetros de eixo único: esses acelerômetros medem a aceleração em apenas um eixo. Eles são relativamente simples e baratos, mas não são adequados para medir acelerações em múltiplas direções.

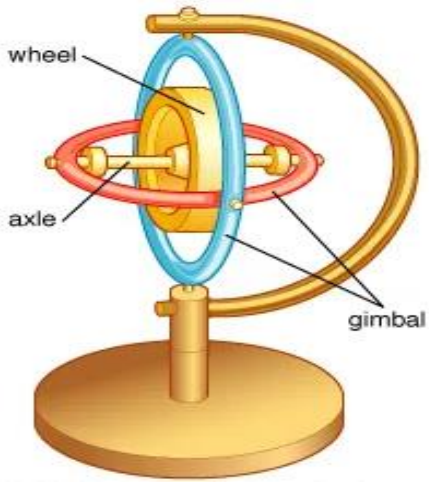
2.4.6 Três eixos

Esses acelerômetros são capazes de medir a aceleração em três direções diferentes. Eles são mais precisos e versáteis que os acelerômetros de eixo único, permitindo a medição de movimentos em todas as direções.

2.5 Giroscópio

O giroscópio como mostra a Figura 8, é um aparelho composto por uma roda giratória ou rotor onde seu eixo de rotação é livre para assumir qualquer orientação no espaço. O funcionamento do giroscópio se baseia na conservação do momento angular, que é uma propriedade física que determina a quantidade de movimento rotacional de um objeto. Quando um giroscópio é acelerado ou rotacionado, ele tende a manter essa rotação devido a essa propriedade, o que permite que ele seja utilizado para detectar movimentos rotacionais em diversos sistemas. A utilização do giroscópio é ampla em várias áreas, incluindo a aviação, navegação, indústria de transporte, robótica, entre outras. Ele é utilizado para medir e controlar a orientação de aeronaves, veículos, satélites, drones e robôs, permitindo que esses objetos se movam com precisão e segurança em seus respectivos ambientes.

Figura 8 - Giroscópio



Fonte: Portal São Francisco, 2022

2.5.1 Movimento de precessão

Um movimento de precessão é definido como uma forma de movimento que ocorre quando se aplica um momento a um corpo em rotação, que faz com que seu eixo tenda a mudar a direção do seu eixo de rotação. Isto acontece porque a resultante da velocidade angular de rotação, e o aumento da velocidade angular produzido pelo momento, é uma velocidade angular em torno de uma nova direção. Geralmente, esta faz variar o eixo do momento aplicado e tem como resultado manter a rotação em torno do eixo inicial.

2.6 MPU6050

Para que a luva interaja com o motor do automodelo, é necessário a utilização de um sensor, sua função é detectar sinais físicos e transformá-los em sinais elétricos. No nosso caso, utilizamos um módulo chamado MPU – 6050 como mostra a Figura 9, que é um giroscópio e um acelerômetro de alta precisão. O MPU contém em sua estrutura um acelerômetro e um giroscópio. É composto por 3 eixos para o acelerômetro e 3 eixos para o giroscópio, totalizando 6 graus de liberdade. Além disso, o MPU6050 possui um processador de sinal digital integrado, que permite a conversão dos sinais analógicos medidos pelo sensor em dados digitais. Ele também possui

recursos de filtragem e gerenciamento de dados para melhorar a precisão e estabilidade da medição.

O MPU6050 é um sensor de movimento de seis eixos que é amplamente utilizado em aplicações de controle de movimento. Algumas das aplicações mais comuns do MPU6050 são:

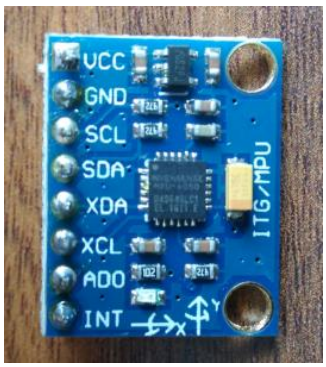
- Estabilização de drones: O MPU6050 é usado em drones para medir a aceleração e a velocidade angular do drone em vários eixos, o que permite que o drone seja controlado de forma mais precisa e estável.

- Robótica: O MPU6050 é usado em robôs para medir a aceleração e a velocidade angular do robô em vários eixos, permitindo que o robô se mova com mais precisão e estabilidade.

- Jogos: O MPU6050 é usado em jogos para medir os movimentos dos jogadores e permitir que eles controlem personagens ou objetos no jogo com movimentos naturais.

- Dispositivos portáteis: O MPU6050 é usado em dispositivos portáteis, como relógios inteligentes, para medir a atividade física, incluindo passos, calorias queimadas e distância percorrida.

Figura 9 – MPU6050



Fonte: Mundo Projetado, 2022

2.7 Programação

Com o avanço da tecnologia, fez-se necessária a comunicação com a máquina, de maneira que pudéssemos solicitar tarefas e operações para facilitar e tornar-se mais produtivo o trabalho feito pelas pessoas. Para isso, foram criadas linguagens que permitem essa interação, são escritas de uma forma amigável ao ser humano, mas que ao ser compilada são traduzidas para a linguagem do computador, ou seja, código binário. A área estudada para as aplicações deste método é denominada programação. Cada linguagem de programação possui um método particular de conversar com a máquina, assim como existem muitas linguagens que nós podemos nos comunicar.

Diversos programas e aplicativos que são usados hoje foram feitos dessa forma, softwares como Word, Excel, Power Point, ou jogos como *Call of Duty*, *Need For Speed*, entre outros.

Para a realização deste projeto, é necessário a comunicação com o dispositivo microcontrolador, para que ele possa processar os dados fornecidos pelos sensores e acionar os motores. Será feito um breve resumo das principais linguagens de programação que são usadas atualmente, bem como a que foi usada neste trabalho.

2.7.1 Linguagem C

Considerada a principal linguagem de programação, e o ponto de partida de muitos programadores, a linguagem C se tornou popular nos anos 80 por sua simplicidade e por permitir diversas aplicações em diferentes áreas. O sucesso da linguagem C foi tanto que a partir dela outras muitas linguagens surgiram tendo C como base, *Objective* C, C# e C++ são alguns dos exemplos que se inspiraram de sua sintaxe e estrutura.

Uma característica importante da linguagem C, é que ela é muito flexível e possibilita o desenvolvimento de qualquer projeto, o programador detém de total controle para fazer adaptação as demandas de seu programa.

2.7.2 Python

Python é considerada uma linguagem de alto nível, ou seja, se aproxima mais da linguagem humana do que da máquina, também é considerada uma das linguagens mais simples de se aprender devido a sua sintaxe de fácil compreensão, além disso possui uma grande biblioteca de padrões, que auxilia na criação do código e integração com outras famosas linguagens como C e C++.

Neste projeto foi utilizado uma linguagem derivada e mais enxuta que o *Python*, chamada *MicroPython*. Ela contém várias subfunções da biblioteca original de *Python* e foi criada especialmente para ser utilizada em microcontroladores, através dessa linguagem foi possível fazer a comunicação entre o ESP32, o MPU e o servomotor MG996R do automodelo.

2.7.3 C++

Linguagem C++ é considerada de nível médio, sua construção começou na década de 80, foi baseada na linguagem C. A linguagem C++ é composta por uma grande variedade de códigos, muitos desses são da linguagem C, além de seus códigos, isso torna a linguagem um pouco mais complexa e exige mais atenção na hora de desenvolver o código para excluir a possibilidade de erros.

Alguns fatos devem ser mencionados a respeito dessa linguagem, como a facilidade de transferência para a linguagem C, não exige uma plataforma de desenvolvimento potente para se trabalhar, além de ser muito eficiente quanto o C, entretanto uma dificuldade em C++ é que necessita tanto de um compilador, quanto de uma unidade de pré processamento, onde é examinado o programa fonte e então feita a modificação de alguns parâmetros, como inclusão de arquivos, a substituição de macros, etc.

2.8 Fonte de alimentação

A fonte de alimentação representa uma parte fundamental do projeto, todos os circuitos necessitam de energia para funcionar, nem sempre os dispositivos terão uma tomada para fazer a alimentação do circuito, então se faz necessário a utilização de

uma fonte de energia externa, seja baterias que possam ser recarregadas ou até mesmo pilhas, que após sua utilização são descartadas e substituídas.

A seguir será mencionado algumas baterias que são comercializadas atualmente e que poderiam ser utilizadas para fornecer alimentação para todos os dispositivos contido no projeto.

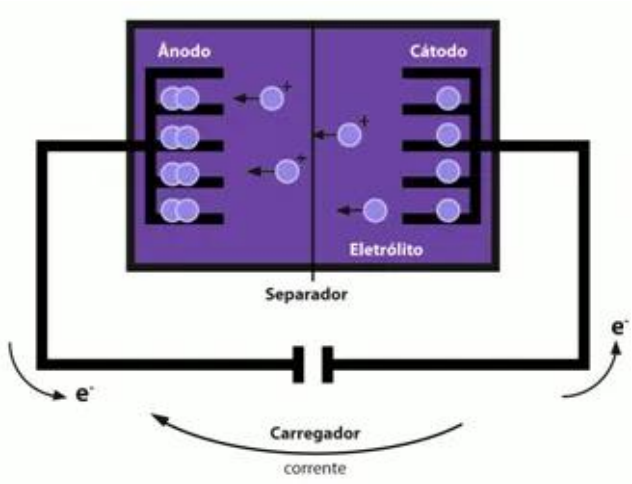
2.8.1 Baterias Íon-Lítio

A bateria Íon-Lítio utiliza um ânodo, um catodo e eletrólito como condutor, como mostra a Figura 10. Quando acontece a descarga elétrica, os íons partem do ânodo para o catodo pelo eletrólito.

Essas baterias compostas por células de íons de lítio possuem uma grande vantagem quando comparada as baterias de chumbo ácido, que é a grande vida útil, além de serem compactas e serem de baixo peso.

Hoje em dia as baterias de íon-lítio dominam o mercado de brinquedos, smartphones e até mesmo carros elétricos, o que era inviável duas décadas atras.

Figura 10 - Bateria de Íon-Lítio



Fonte: Brasil Escola, 2022

2.8.2 Baterias de chumbo-ácido

Implementadas por Henry Ford na década de 1910, as baterias de chumbo-ácido revolucionaram o mercado automobilístico no passado e ainda hoje é a principal bateria utilizada em veículos.

O oxido de chumbo atua como material ativo positivo, já o chumbo esponjoso como material negativo. E o ácido sulfúrico atua como eletrólito. Quando acontece a descarga da energia a interação entre o oxido de chumbo e o chumbo esponjoso são convertidos em sulfato de chumbo.

Apesar de sua larga utilização a bateria de chumbo-ácido tem algumas limitações, o eletrólito e o conteúdo da carga podem causar impactos negativos no meio ambiente, além da baixa utilização da capacidade nominal da bateria, entre 20% e 30%, outro ponto é o tamanho como mostra a Figura 11, e peso elevados, o que torna o deslocamento difícil e principalmente a utilização em produtos pequenos inviável.

Figura 11 - Bateria de Chumbo-ácido

Fonte: Clube da Química, 2022

2.8.3 Pilhas

Pilhas são dispositivos eletroquímicos capaz de produzir corrente elétrica através de reações de redução que significa a capacidade do material de ganhar elétrons e oxidação o contrário, ou seja, a capacidade de perder elétrons como mostra na Figura 12. O caminho que os elétrons fazem de um componente ao outro cria a energia elétrica, onde é utilizada para alimentar os dispositivos elétricos.

Figura 12 - Pilha

Fonte: Brasil Escola, 2022

Neste projeto para funcionar todos os componentes, foi usado pilhas alcalinas em série, por seu custo-benefício, peso e compactação, além da facilidade de aumentar as tensões até alcançar a tensão necessária para o funcionamento de todos os componentes do projeto, associando todas as pilhas em série, onde a tensão é somada proporcional ao número de pilhas, diferente da ligação em paralelo que a tensão diferencial ou ddp (diferença de potencial), é igual a tensão de cada pilha.

3. DESENVOLVIMENTO

A princípio foi realizado o planejamento de como seria o automodelo controlado pela luva, para otimização de recursos, decidimos por utilizar a plataforma de um carrinho de controle remoto, utilizando a estrutura e componentes que facilitassem a continuidade do projeto, onde seria adaptado como plataforma para testes e como projeto final. Dessa forma foi necessário retirar a placa de controle do carrinho, utilizando apenas os dois motores do veículo, onde seria possível controlar a direção e movimento através dos sinais gerados a partir do MPU preso a uma luva.

A seguir será detalhado o desenvolvimento do projeto, as dificuldades encontradas e as soluções que foram utilizadas que possibilitaram a conclusão e perfeito funcionamento do veículo.

A construção do projeto se iniciou com a desmontagem do veículo, onde foi feita uma engenharia reversa para entendimento do funcionamento de todos os componentes presentes, tornando mais fácil saber quais componentes poderiam ser aproveitados, e quais deveriam ser especificados e agregados para funcionar de acordo com o planejado. Como pode ser visto na Figura 13 o automodelo desmontado.

Figura 13 - Automodelo desmontado

Fonte: Própria, 2023

3.1 Motores do automodelo

Tendo como base o conhecimento em carrinhos de brinquedos antigos, foi desmontado e encontrado um motor DC 9V com uma caixa de redução (responsável por controlar a velocidade do automodelo, e os movimentos para frente e para trás) juntamente com um motor de passo responsável alterar as direções, variando para direita ou para a esquerda).

Optou-se por manter o motor DC que já estava no carrinho, conforme ilustrado na figura 14, devido à sua redução que se encaixava perfeitamente nas engrenagens. No entanto, decidiu-se substituir o motor de passo nas rodas dianteiras por um servomotor, devido à sua facilidade de programação.

Figura 14 - Motor DC 9V

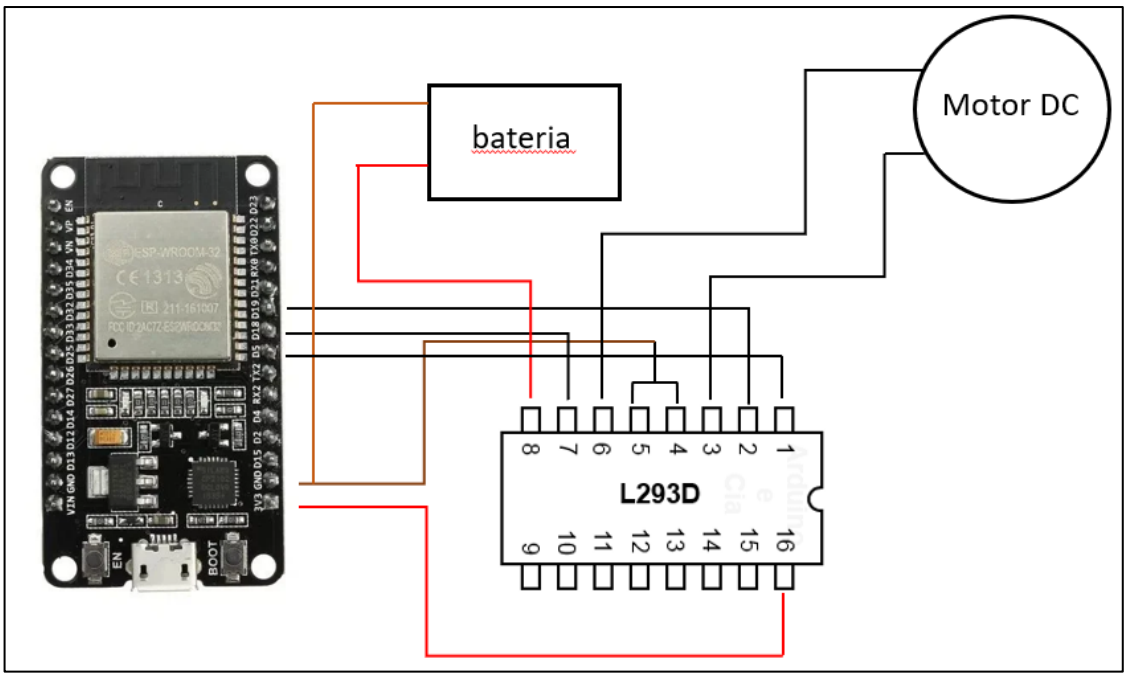
Fonte: Alibaba – Micro Motor DC 9V, 2023

3.1.1 Motor de corrente contínua 9V

Para o automodelo andar, foi utilizado um motor de corrente contínua, alimentado com uma bateria de 9V. Outra função indispensável ao qual o carro pode realizar é a de se mover para frente e para trás, isso acontece devido a inversão de rotação do motor, função que é realizada graças a Ponte H. Para isso foi utilizado a Ponte H L293D que possibilitou a utilização de apenas um motor para que o veículo andasse para frente e para trás. A ponte foi alimentada com o pino de 3.3V do ESP32.

O esquema de ligação para teste na protoboard da ponte para o motor e para o ESP é demonstrado na figura 15.

Figura 15 - Circuito com a ponte H



Fonte: Própria, 2023

Logo após, criamos um programa na IDE Arduino para fazer o teste:

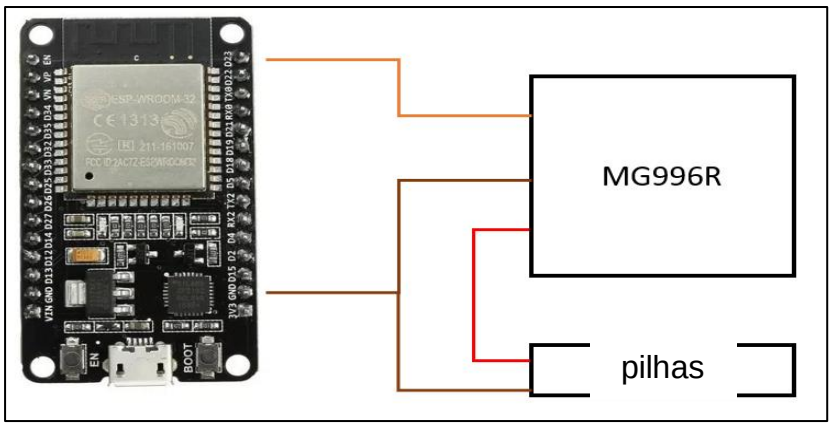
```
// Definindo as portas do Arduino conectadas ao módulo ponte H L293D
int enablePin = 5;
int in1Pin = 18;
int in2Pin = 19;

void setup() {

  // Configurando as portas como saídas
  pinMode(enablePin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  // Desligando o motor inicialmente
  digitalWrite(enablePin, LOW);
}

void loop() {
  // Girando o motor em sentido horário
```

```
  digitalWrite(in1Pin, HIGH);
  digitalWrite(in2Pin, LOW);

  // Aumentando gradualmente a velocidade do motor
  for(int i=0; i<=255; i++) {
    analogWrite(enablePin, i);
    delay(10);
  }

  // Reduzindo gradualmente a velocidade do motor
  for(int i=255; i>=0; i--) {
    analogWrite(enablePin, i);
    delay(10);
  }

  // Desligando o motor
  digitalWrite(enablePin, LOW);
  delay(1000);
}
```

3.1.2 Servomotor

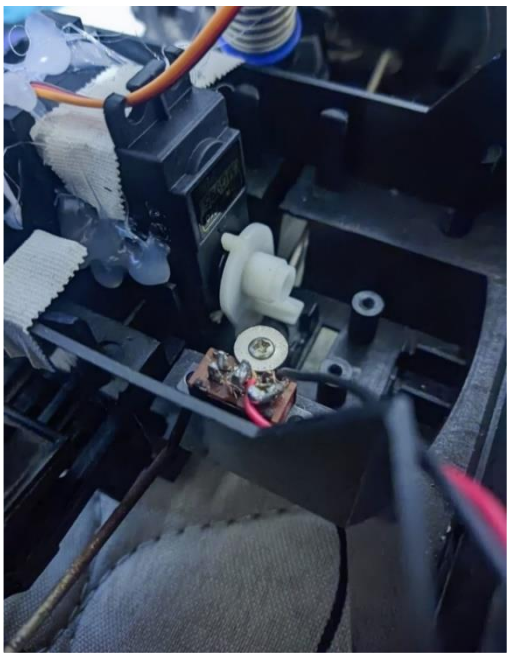
O modelo do servomotor escolhido foi o MG996R, o qual através de testes realizados, mostrou-se muito eficiente, onde era possível ver os movimentos das rodas (direita e esquerda), comprovando o perfeito controle das direções. Utilizando da estrutura do próprio carrinho, os dois eixos presos as rodas foram fixadas ao servomotor, onde ao movimentar o servomotor, as rodas acompanhavam as direções impostas a elas. Foi comprovado também através do programa criado no *software* IDE Arduino, responsável por realizar a lógica de controle do servo, onde ao executar o programa era possível controlar o servo através dos sinais recebidos do MPU6050. Para testar o servomotor, alimentamos ele com um conjunto de 4 pilhas de 1.5V em série e conectamos em uma saída digital do ESP, abaixo o esquema de ligação para teste na protoboard na figura 16 e na figura 17 o servo motor já fixado no automodelo.

Figura 16 - Circuito com o Servo Motor



Fonte: Própria, 2023

Figura 17 - Ponte H fixada ao automodelo



Fonte: Própria, 2023

O código criado movimenta o servo para duas direções em 180 graus, com uma velocidade determinada.

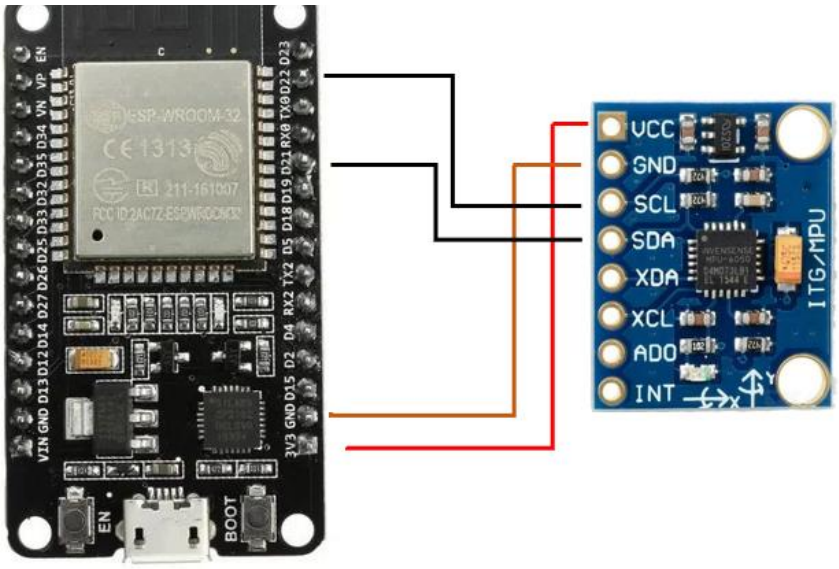
```
#include <Servo.h>

static const int servoPin = 23;

Servo servo1;

void setup() {
    Serial.begin(115200);
    servo1.attach(servoPin);
}

void loop() {
    for(int posDegrees = 0; posDegrees <= 180; posDegrees++) {
        servo1.write(posDegrees);
        Serial.println(posDegrees);
        delay(10);
    }

    for(int posDegrees = 180; posDegrees >= 0; posDegrees--) {
        servo1.write(posDegrees);
        Serial.println(posDegrees);
        delay(10);
    }
}
```

3.2 Captura dos Movimentos pelo MPU6050

Como o MPU6050 é capaz de medir a aceleração de um objeto (nesse caso a luva) através dele foi possível captar os movimentos da luva e transformá-los em sinais, que foram enviados para o ESP32 (mestre) para que seja possível estabelecer a relação necessária dos movimentos realizados com a luva com os movimentos realizados do veículo.

Após a análise da operacionalidade dos motores, foi necessário proceder à verificação da capacidade do sistema de controle eletrônico de estabilidade (ESP) em efetuar a leitura do MPU6050, após a detecção das variações de aceleração e velocidade angular, e exibir esse valor no monitor serial. Por meio da utilização da biblioteca MPU6050.h, os pinos designados para a conexão do MPU com o ESP estão padronizados, ou seja, é imprescindível que sejam esses pinos específicos, conforme demonstrado no esquema presente na figura 18.

Figura 18 - Circuito com o MPU6050



Fonte: Própria, 2023

O programa utilizado para o teste:

```
// Basic demo for accelerometer readings from Adafruit MPU6050

#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
```

```
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
    break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
```

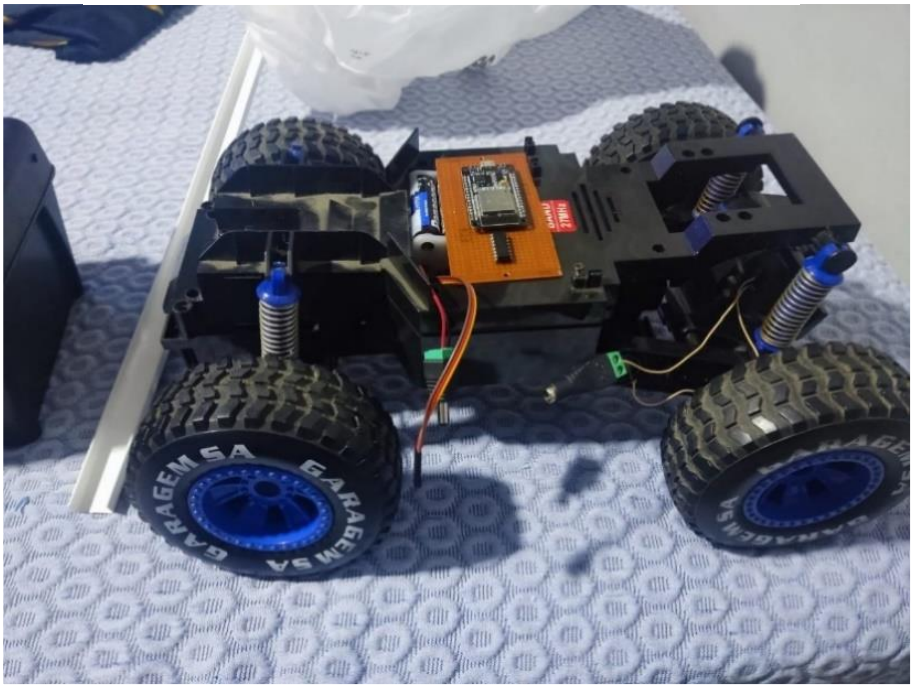
```
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println("");
  delay(100);
}

void loop() {

  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  /* Print out the values */
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.println(" rad/s");

  Serial.print("Temperature: ");
  Serial.print(temp.temperature);
  Serial.println(" degC");

  Serial.println("");
  delay(500);
}
```

3.3 ESP32 (escravo)

O ESP32 escravo recebe as mensagens e interpreta os comandos de controle enviados pelo mestre. Com base nas mensagens recebidas, o escravo como mostra a figura 19, controla os motores do automodelo para executar as ações solicitadas (por exemplo, mover para frente, para trás, girar para a esquerda ou para a direita).

No projeto o ESP32 que está no veículo tem a função de escravo pois é ele quem recebe os sinais de comando de direção e aceleração enviados pelo mestre que está na luva. Ele é o responsável por repassar os comandos recebidos do ESP32 (mestre) de forma remota por meio do *Wifi*, para o motor e servomotor, possibilitando o controle da velocidade e das direções do veículo, respectivamente.

Figura 19 - Automodelo com o ESP32 (escravo)



Fonte: Própria, 2023

3.4 ESP32 (mestre)

Uma das principais funções do ESP32 que está na luva tem a função de receber os sinais do MPU6050 e a partir disso enviar os sinais através da comunicação *WiFi* para o ESP32 que está no veículo. Como apresentado no apêndice A, a programação define que em um intervalo de variações dos sinais enviados pelo MPU o motor aciona

para um dos dois sentidos estabelecidos, e para o servomotor o mesmo, ao haver uma oscilação pré-definida, o veículo altera a sua direção para direita ou esquerda. Na figura 20 é possível analisar a placa com o circuito que é fixo a luva.

Figura 20 - Circuito do ESP32 (mestre) na luva

Fonte: Própria, 2023

3.5 Fontes

Com todos os componentes definidos, o último item, a fonte de alimentação era necessária atender dois requisitos, ser pequena para ser fixada junto aos componentes e no espaço reservado no veículo, e suprir a tensão de alimentação necessária de todos os dispositivos presentes no projeto.

Como a fonte fica fixa junto a luva e aos componentes que estão na luva, foi necessário a utilização de uma fonte pequena que não fosse incômoda ao usuário e que atendesse a potência necessária do circuito. Para isso foi escolhida a fonte 9V, junto ao um regulador de tensão LM7805.

Para alimentação do automodelo foi utilizado um conjunto de 8 pilhas Alcalinas AA como mostra a figura 21, sendo dois conjuntos em paralelo com cada um com 4 pilhas em série, resultando em 6Vdc para alimentação do servomotor, ESP32, já a ponte H é alimentada pela saída 3.3V do ESP32.

Figura 21 - Pilhas utilizadas no projeto

Fonte: Própria, 2023

Para alimentação do motor, uma bateria de 9V que pode ser vista na figura 22, que através da ponte H irá alimentar os dois sentidos de giro do motor.

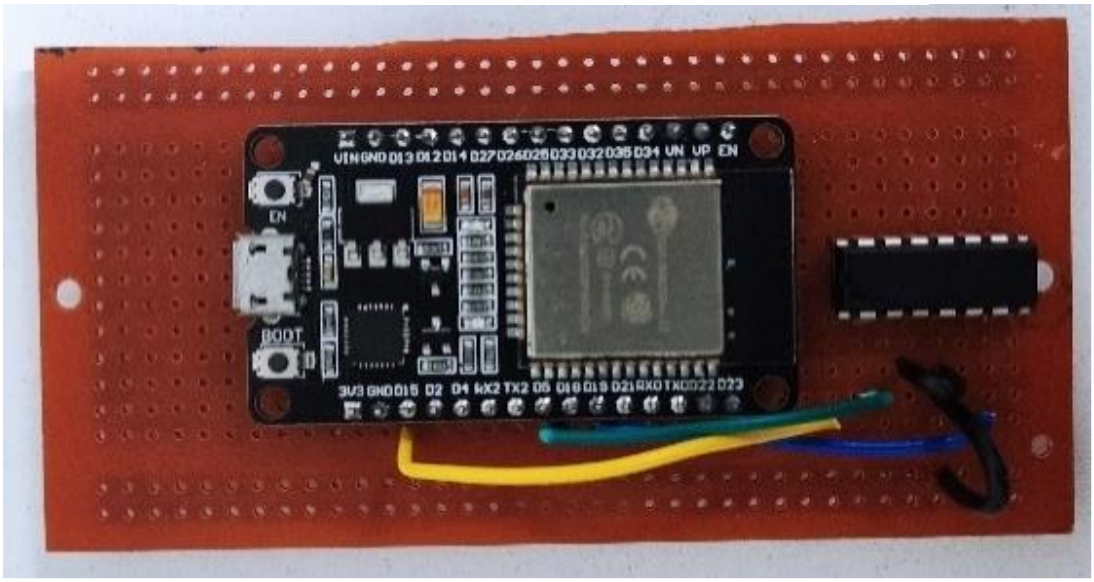
Figura 22 - Bateria utilizada no projeto

Fonte: Própria, 2023

3.6 Placa

Para a placa fixa ao automodelo foi utilizada uma placa padrão tipo trilha, onde estão presentes o ESP32 e a ponte H L293D, a disposição dos componentes foi realizada de forma que houvesse fácil acesso aos mesmos para manutenção. O modelo ainda conta com uma chave on/off para controlar a alimentação das pilhas. Essa placa possui medida de 100x50mm e na figura 23 é possível visualizar a disposição dos componentes.

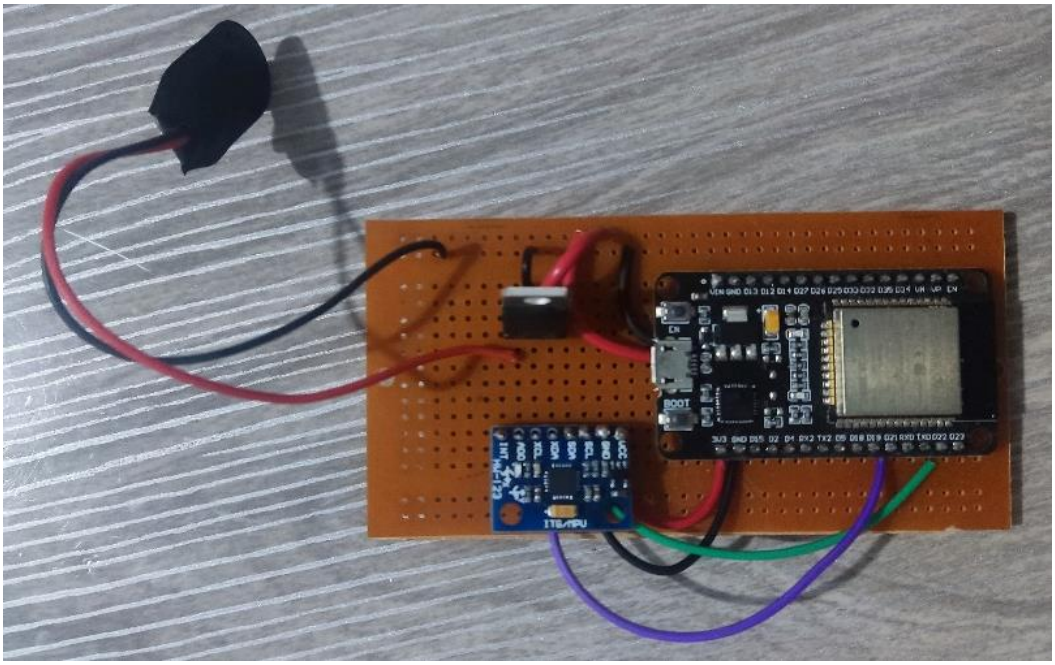
Figura 23 - Placa com o circuito do ESP32 (mestre)



Fonte: Própria, 2023

No desenvolvimento da placa que fica fixa a luva, foi primordial atender ao requisito do tamanho, pois como todos esses componentes precisam estar fixados na luva, não seria possível criar algo que fosse maior do que a própria luva, para isso foi utilizado também uma placa padrão tipo trilha reduzida para compor o ESP32, o MPU6050 e o regulador de tensão, esta placa com medida de 100x50mm atendendo ao requisito principal, como pode ser visto na figura 24.

Figura 24 - Placa com o circuito do ESP32 (escravo)



Fonte: Própria, 2023

3.7 Sistema *Wireless*

A comunicação foi feita através do protocolo ESP-NOW, a qual é baseada em um esquema mestre-escravo, onde um dispositivo é configurado como o nó mestre, no caso o ESP32 que está na luva e o outro ESP32 atua como escravo (o ESP32 que está no automodelo)

A seguir, um resumo do processo de comunicação usando o protocolo ESP-NOW:

Configuração dos dispositivos: Primeiro, foi configurado os dispositivos ESP32 para usar o protocolo ESP-NOW. Isso incluiu inicializar a biblioteca ESP-NOW, configurar os parâmetros de comunicação, definir os dispositivos como mestre ou escravo e registrar as funções de *callback* para processar os dados recebidos.

Registro dos dispositivos: Os dispositivos ESP32 precisam ser registrados uns com os outros antes de iniciar a comunicação. O dispositivo mestre envia uma solicitação de registro para o esp32 escravo, e este responde com um código de registro único.

Envio de dados: O dispositivo mestre envia dados para o esp32 escravo usando a função *esp_now_send()*. Onde os dados são enviados em pacotes de até 250 bytes. Recebimento de dados: No dispositivo escravo foi configurado uma função de *callback* para processar os dados recebidos do dispositivo mestre. Essa função é chamada automaticamente sempre que um pacote de dados é recebido.

Controle e resposta: Com base nos dados recebidos, o dispositivo escravo pode executar ações específicas ou enviar uma resposta de volta ao dispositivo mestre, caso seja necessário.

3.8 Montagem

A montagem do projeto foi dividida em duas partes distintas. Na primeira parte, utilizou-se como base um carrinho de controle remoto de brinquedo, o qual passou por um processo de desmontagem para a remoção dos componentes desnecessários ao trabalho em desenvolvimento. Nesse sentido, a placa de controle e o motor de passo, responsável pela direção do automodelo, foram retirados devido à falta de especificações compatíveis com a plataforma selecionada, o ESP32. Contudo, o motor DC 9V, responsável pelo movimento para frente e para trás, foi mantido.

Após alcançar esse estágio, deu-se início à montagem, substituindo o antigo motor de passo pelo servomotor MG996R, o qual permitiu o controle da direção, pois era compatível com o ESP32.

No controle do motor DC 9V, foi necessário empregar uma ponte-H, um circuito integrado responsável por controlar o movimento para frente e para trás. Optou-se pelo uso do circuito integrado ponte-H L293D, o qual resolveu esse problema, possibilitando os dois movimentos com um único motor.

Devido às diferentes exigências de alimentação, incluindo quatro componentes distintos, decidiu-se utilizar oito pilhas alcalinas AA de 1,5V. Essas pilhas foram agrupadas em dois conjuntos em paralelo, cada um composto por quatro pilhas conectadas em série. Essa configuração permitiu alcançar a tensão necessária de 6V, além de fornecer a capacidade de corrente adequada para alimentar o Servo Motor, o ESP32, e para a alimentação da ponte-H foi utilizado a saída de 3,3V do próprio

ESP32. Quanto à alimentação do motor DC 9V, foi utilizada uma bateria de 9V, sendo essa a única fonte externa empregada, exclusivamente para esse componente.

A fim de agrupar e interconectar esses componentes, optou-se por uma placa perfurada com trilhas de 50x100mm. Essa escolha se deve ao grande número de pinos do ESP32 e da ponte-H, tornando essa a forma mais prática de desenvolver o projeto. A placa consiste nos componentes ESP32, ponte-H L293D e um diodo 1N4007, além das conexões com as duas fontes de alimentação de 6V e 9V, o Servo Motor MG996R e o Motor DC 9V.

Na segunda parte do projeto, o controle do automodelo é realizado por meio de uma luva, permitindo controlar os movimentos do automodelo com base na inclinação da mão. Para esse controle, utilizou-se o acelerômetro e giroscópio MPU6050. A fim de interpretar essas informações e transmiti-las a outro ESP32, iniciou-se a montagem utilizando apenas esses componentes.

Dado que era necessário utilizar o ESP32 novamente, optou-se por uma placa perfurada com trilhas de 50x100mm para interconectar os dois componentes. No entanto, era preciso alimentar esse circuito, e para isso foi escolhida outra bateria de 9V devido ao seu tamanho e ao requisito de peso para o usuário que controla os movimentos do automodelo. Devido à diferença de tensão entre a bateria (9V) e a tensão de alimentação do ESP32 (5V), utilizou-se um regulador de tensão LM7805 para corrigir essa diferença. Ao finalizar a montagem da placa com os componentes, esta foi fixada a uma luva, tornando possível controlar as direções do automodelo através da inclinação da mão do usuário. Ao final do projeto o automodelo pode ser demonstrado através da figura 25.

Figura 25 - Automodelo finalizado montagem

Fonte: Própria, 2023

3.9 Programação

Para a programação, foram utilizados dois programas distintos, um para cada ESP32. A seguir, será apresentado uma explicação detalhada para cada um desses programas, destacando suas funcionalidades e características específicas.

3.9.1 Código Emissor

Este código é um dos modos de como enviar dados do módulo MPU6050 via ESP-NOW utilizando o ESP32. O ESP32 atua como o transmissor de dados, enquanto outro ESP32 atua como o receptor.

```
#include <Wire.h>
#include <MPU6050.h>
#include <esp_now.h>
#include <WiFi.h>
```

Este trecho do código inclui as bibliotecas necessárias para a comunicação com o módulo MPU6050, para a comunicação ESP-NOW e para a comunicação *WiFi*.

```
uint8_t broadcastAddress[] = {0x94, 0xB9, 0x7E, 0xE5, 0xA4, 0x64};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
 float inclinacaoX;
 float inclinacaoY;
 float velocidade;
 int anguloServo;
} struct_message;
```

Na primeira linha do código, é definido o endereço MAC do receptor. O endereço deve ser definido de acordo com o dispositivo receptor utilizado.

O segundo trecho do código define uma estrutura chamada *struct_message*, que é usada para armazenar os dados que serão enviados. A estrutura contém quatro variáveis: *inclinacaoX, inclinacaoY, velocidade* e *anguloServo.*

```
esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
```

Este trecho do código define uma estrutura *esp_now_peer_info_t* que será utilizada para armazenar as informações do dispositivo receptor e uma função de callback chamada *OnDataSent*. Esta função é chamada quando os dados são enviados com sucesso ou quando ocorre um erro.

```
MPU6050 mpu;

void setup() {

  Serial.begin(115200);
  // Inicializando o módulo MPU6050
  Wire.begin();
  mpu.initialize();

   // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}
```

Este trecho do código é a função *setup()*. Nesta função, o módulo MPU6050 é inicializado, a conexão Wi-Fi é configurada e a comunicação ESP-NOW é iniciada. Em seguida, a função *OnDataSent* é registrada como callback e o dispositivo receptor é registrado.

```
void loop() {
 // Lendo os valores do acelerômetro
 int16_t ax, ay, az;
 mpu.getAcceleration(&ax, &ay, &az);

 // Calculando a inclinação do módulo em relação ao eixo X
 float inclinacaoX = atan2(ay, sqrt(pow(ax,2) + pow(az,2))) * 180 / PI;

 // Calculando a inclinação do módulo em relação ao eixo Y
 float inclinacaoY = atan2(-ax, sqrt(pow(ay,2) + pow(az,2))) * 180 / PI;

 float velocidade = map(inclinacaoX, -90, 90, -255, 255);

 myData.inclinacaoX = inclinacaoX;
 myData.inclinacaoY = inclinacaoY;
 myData.velocidade = velocidade;

 // Send message via ESP-NOW
 esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

 if (result == ESP_OK) {
   Serial.println("Sent with success");
 }
 else {
   Serial.println("Error sending the data");
 }
 delay(50);
}
```

O código lê os valores do acelerômetro MPU6050 e calcula a inclinação do módulo em relação aos eixos X e Y. Em seguida, a inclinação em X é mapeada para um valor de velocidade entre -255 e 255. Os valores de inclinação em X e Y e a velocidade são armazenados em uma estrutura e enviados via protocolo ESP-NOW. O código também verifica se a mensagem foi enviada com sucesso e adiciona um atraso de 50 milissegundos antes de executar novamente o loop.

3.9.2 Código de Receptor

```
#include <esp_now.h>
#include <WiFi.h>
#include <Servo.h>

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
 float inclinacaoX;
 float inclinacaoY;
 float velocidade;
 int anguloServo;
} struct_message;
```

Como no emissor, as bibliotecas necessárias são importadas. Além disso, é definida a mesma estrutura, *struct_message*, que contém quatro campos: *inclinacaoX, inclinacaoY, velocidade* e *anguloServo*. Esta estrutura será usada para enviar dados do MPU6050 para o automodelo.

```
int enablePin = 15;
int in1Pin = 18;
int in2Pin = 19;

// Definindo as portas do Arduino conectadas ao servo motor
int servoPin = 23;

// Criando um objeto Servo
Servo servo;

// Create a struct_message called myData
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
```

Aqui, são definidos os pinos que o código usará para se comunicar com o módulo ponte H L293D e o servo motor. Além disso, é criado um objeto *Servo* e uma variável *myData* da estrutura *struct_message*.

```
// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
}

void setup() {
  // Configurando as portas como saídas
  pinMode(enablePin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  // Configurando o servo motor
  servo.attach(servoPin);
  servo.write(90);

  // Desligando o motor inicialmente
  digitalWrite(enablePin, LOW);

  // Initialize Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
}
```

O código acima define a função *OnDataRecv*, que será executada quando o ESP32 receber dados do MPU6050. Ela recebe os dados e os armazena na variável *myData*.

Na função *setup*, são configurados os pinos do L293D e do servo motor como saídas e o servo motor é definido na posição central. Além disso, a comunicação Wi-Fi é configurada e o *OnDataRecv* é registrado como função de callback para receber dados via ESP-NOW.

```
void loop() {

  // Movimentando o motor de acordo com a inclinação no eixo X
  if(myData.inclinacaoX < -5) {
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
    analogWrite(enablePin, abs(myData.velocidade));
  } else if(myData.inclinacaoX > 5) {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    analogWrite(enablePin, abs(myData.velocidade));
  } else {
    digitalWrite(enablePin, LOW);
  }

  // Movimentando o servo motor de acordo com a inclinação no eixo Y
  int anguloServo = map(myData.inclinacaoY, -90, 90, 0, 180);
  servo.write(anguloServo);

  delay(20);
}
```

Primeiramente, o código verifica a inclinação no eixo X recebida do transmissor. Se a inclinação for menor que -5, o motor é movimentado em uma direção, caso contrário, se a inclinação for maior que 5, o motor é movimentado em outra direção. Se a inclinação estiver entre -5 e 5, o motor é desligado.

Depois disso, o código verifica a inclinação no eixo Y recebida do transmissor e calcula o ângulo correspondente para o servo motor usando a função *map()*. Em seguida, o servo é movido para a posição correspondente a esse ângulo.

Por fim, o código espera um intervalo de 20 milissegundos antes de executar novamente a função *loop()*, para evitar que a execução ocorra muito rapidamente e sobrecarregue o sistema.

4. CONCLUSÃO

Este trabalho de conclusão de curso teve como objetivo desenvolver um sistema de controle remoto inovador para um carrinho, utilizando dois microcontroladores ESP32 em conjunto com o sensor MPU6050. Além disso, o sistema incorporou uma ponte H para controle do sentido de rotação das rodas de um motor DC, além de um servomotor para controle de direção. A utilização de dois ESP32 permitiu a divisão das tarefas, com um atuando como o controlador mestre, responsável pela leitura dos dados do MPU6050 e envio dos comandos, e o outro como o controlador escravo encarregado do acionamento da ponte H para controlar o motor DC e o servomotor, onde foi utilizada uma comunicação WIFI, graças ao recurso disponível no ESP32.

O carrinho de controle remoto demonstrou um desempenho notável durante os testes e validações realizados, capaz de receber os sinais do MPU6050 para determinar as variações de aceleração e velocidade angular, traduzindo-as em comandos de movimento para o carrinho. A integração da ponte H com o motor DC e o servomotor possibilitou o controle eficiente da tração e direção do veículo. O motor DC permitiu o deslocamento do carrinho para frente e para trás, de acordo com o movimento feito na luva, enquanto o servomotor proporcionou a capacidade de direcionamento preciso para direita ou esquerda. A comunicação via Wi-Fi proporcionou uma experiência de controle remoto conveniente, permitindo que o usuário interagisse com o carrinho a uma distância segura. A integração desses componentes proporcionou uma experiência de controle remoto aprimorada, demonstrando a aplicabilidade e o potencial dessas tecnologias na área de automação e robótica.

Este trabalho contribui para o avanço do conhecimento nesse campo, oferecendo uma solução viável e inovadora para o controle de um carrinho, estimulando a criação de soluções cada vez mais avançadas e eficientes.

5. PROPOSTA FUTURA

Uma das primeiras ideias de melhora do projeto é a utilização de apenas uma fonte para alimentar o automodelo, onde seria possível atender aos requisitos para alimentar motor, servomotor, ESP32 e ponte H, e através de reguladores de tensão ter as três tensões necessárias, 9V, 6V e 5V. Além da construção de uma placa onde houvesse uma melhor disposição dos componentes diminuindo a necessidade cabos. Também alterar os componentes da luva, utilizando menos componentes para que usar a luva não vire um incomodo ao usuário.

Outra possível melhoria de desenvolvimento seria equipar mais o veículo, podendo controlá-lo mesmo sem visualizá-lo, tendo uma câmera em que será possível acompanhar e guiá-lo em locais com difícil acesso. Para isso também é necessária uma pequena tela onde seja possível acompanhar o trajeto em que o veículo está realizando o percurso.

Adicionar sensores no entorno do veículo que sinalizasse nessa tela objetos perigosos, ou locais com grandes desníveis, pois ao entrar em lugares onde o controlador somete tiver como visualizar através da tela, haveria como sinalizar sobre os cuidados necessários.

Finalmente, algo que preservaria mais os dispositivos seria a adição de algum acessório na luva e no automodelo, onde contribuiria para que os componentes ficassem mais protegidos para evitar danos e diminuir a vida útil dos equipamentos, além de diminuir a fonte de energia, pois ao usar por longos períodos, o peso se torna um incomodo para o usuário.

6. REFERÊNCIAS BIBLIOGRÁFICAS

PERRONE, Gabriel Cury. “O Giroscópio e a conservação de Momento Angular”; AMLEF. Disponível, em: https://www.ufrgs.br/amlef/2020/06/01/o-giroscopio-e-a-conservacao-de-momento-angular/.Acesso em: 20 de maio de 2022.

SILVEIRA, Debora Priscila. "O que o giroscópio?"; Oficina da Net. Disponível em: https://www.oficinadanet.com.br/post/17290-o-que-e-giroscopio-nos-smartphones. Acesso em 19 de abril de 2022.

VANIN, Vito Roberto. "A precessão de um giroscópio"; Universidade de São Paulo. Disponível em: https://eaulas.usp.br/portal/video.action?idItem=23518. Acesso em 22 de abril de 2022.

"ESP32"; EXPRESSIF. Disponível em: https://www.espressif.com/en/products/socs/esp 32. Acesso em 15 de abril de 2022.

"O Que É ESP32? Pra Que Serve? Quando Usar?"; Lobo da Robótica. Disponível em: https://lobodarobotica.com/blog/o-que-e-esp32-pra-que-serve-quando-usar/. Acesso em 21 de abril de 2022.

OLIVEIRA, Euler. "Conhecendo o NodeMCU-32S ESP32"; Master Walker. Disponível em: https://blogmasterwalkershop.com.br/embarcados/esp32/conhecendo-o-nodemcu-32s-esp32. Acesso em 10 de maio de 2022.

"ESP-NOW: Comunicação direta entre módulos ESP8266 e ESP32"; Arduino e Cia. Disponível em: https://www.arduinoecia.com.br/esp-now-comunicacao-direta-entre-modulos-esp8266-e-esp32/. Acesso em 7 de junho de 2022.

"Getting Started with ESP-NOW (ESP32 with Arduino IDE)"; Random Nerd Tutorials. Disponível em: https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/. Acesso em 28 de julho de 2022.

"ESP-NOW"; EXPRESSIF. Disponível em: https://www.espressif.com/en/solutions/low-power-solutions/esp-now. Acesso em 18 de setembro de 2022.

MARQUES, Jemerson. "O que é ESP-NOW - E como Funciona? - Código exemplo explicado!!!"; FVM Learning. Disponível em: https://www.fvml.com.br/2020/01/o-que-e-esp-now-e-como-funciona-codigo.html. Acesso em 1 de outubro de 2022.

"O que é um Servo Motor, como funciona e quais as vantagens?"; KALATEC Automação. Disponível em: https://blog.kalatec.com.br/o-que-e-servo-motor/. Acesso em 13 de novembro de 2022.

MATTEDE, Henrique. "O que é Servo motor e como funciona?"; Mundo da Elétrica. Disponível em: https://www.mundodaeletrica.com.br/o-que-e-servo-motor-e-como-funciona/. Acesso em 25 de outubro de 2022.

GOGONI, Ronaldo. "O que é Bluetooth?"; tecnoblog. Disponível em: https://tecnoblog.net/responde/o-que-e-bluetooth/. Acesso em 17 de novembro de 2022.

"Bluetooth: o que é, como funciona e versões"; INFOWESTER. Disponível em: https://www.infowester.com/bluetooth.php. Acesso em 17 de novembro de 2022.

ALVES, Pedro. "Ponte H – O que é e como funciona!"; Manual da Eletrônica. Disponível em: https://www.manualdaeletronica.com.br/ponte-h-o-que-e-como-funciona/. Acesso em 18 de novembro de 2022.

CALACHE D. C. Caracterização de um Acelerômetro Baseado em SistemasMicroeletromecânicos (MEMS). Projeto de graduação, 2013

KISAKI H. S. Robô Com Sistema de Giroscópio e Acelerômetro. Trabalho de Conclusão de Curso, 2018.

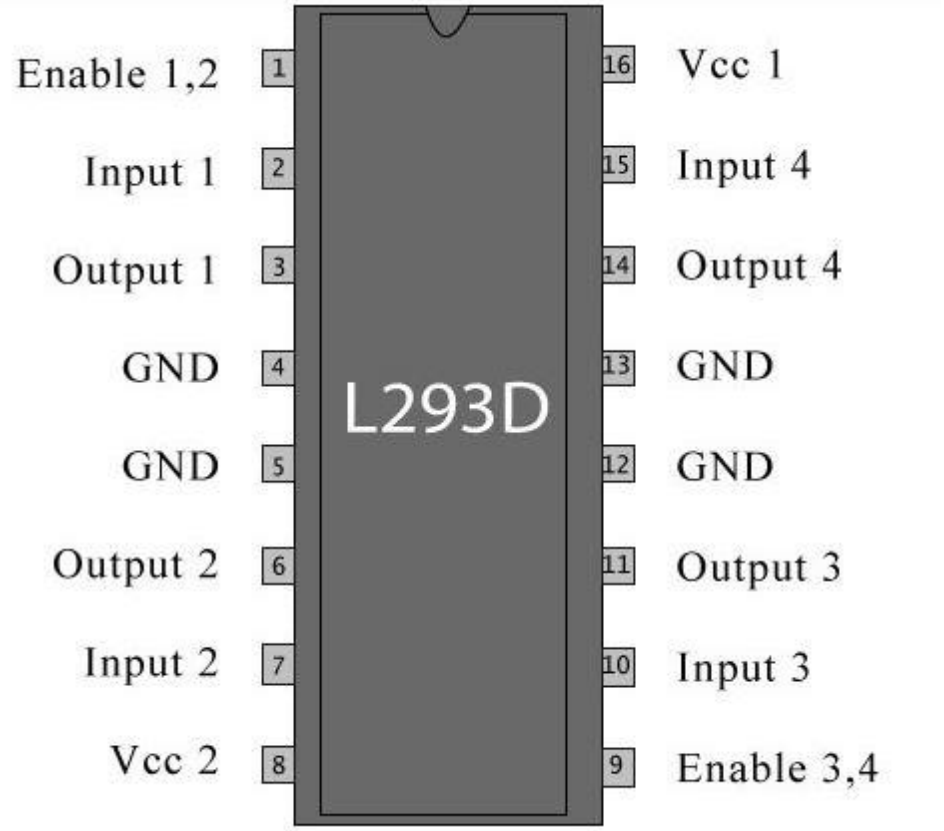
LONGARETTI D. Implementação De Uma Luva Microcontrolada Para A Captura De Gestos. Trabalho de Conclusão de Curso, 2015

DE OLIVEIRA HELANO M. ARCLUS – Automodelo Remotamente Controlado Por Luva Sensoriada.

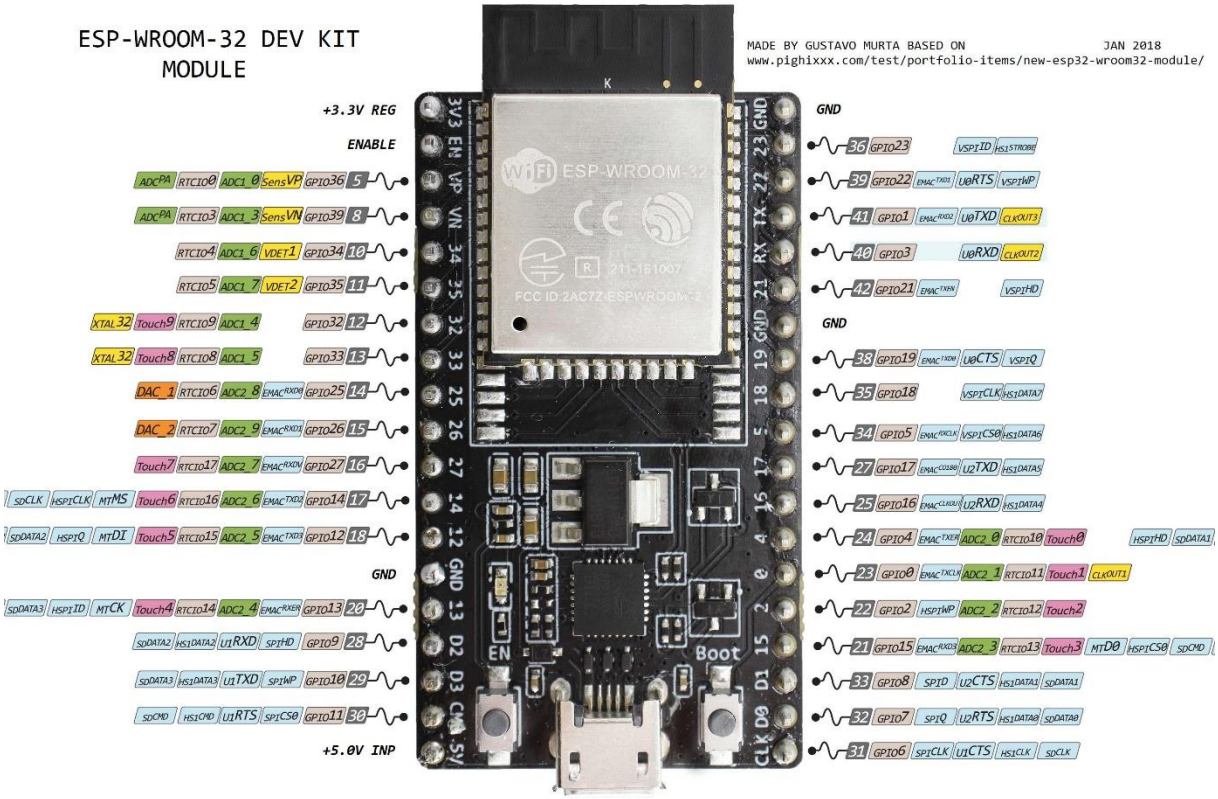
SANTOS, Gabriel Almeida. Luva Háptica para controle de um manipulador. Mostra Nacional de Robótica, 2015.

MACIEL, Clayton L. M.; SILVA, Rafael C.; BARBOSA, Luis Filipe Wiltgen. CONTROLE À DISTÂNCIA UTILIZANDO UM PROTÓTIPO EXPERIMENTAL DE UM LUVA SENSORIADA. Anais do XI Encontro Latino Americano de Iniciação Científica–INIC, Vale do Paraíba, 2007.

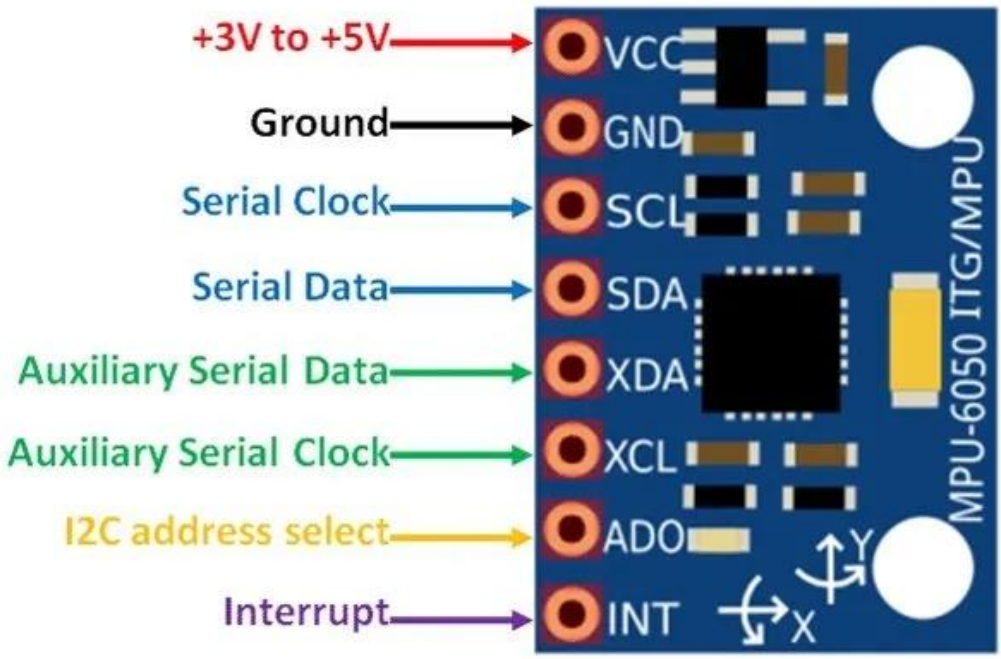
Anexo A - Controlador Ponte H L293D - Pinagem



Anexo B - ESP32 – Pinagem do ESP-32

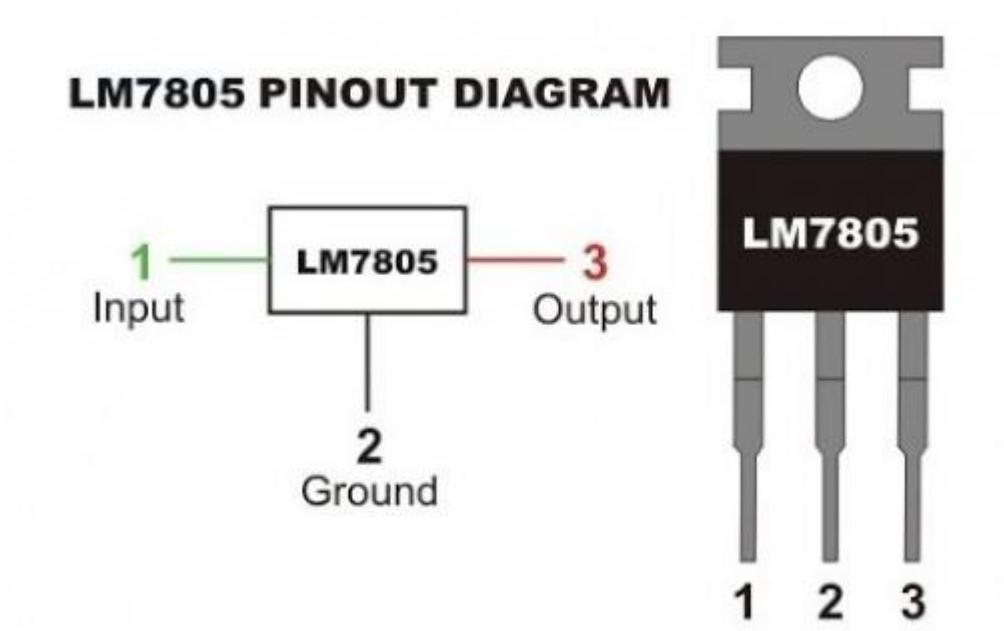


Anexo C - MPU6050 – Pinagem do MPU6050



MPU6050 Pinout

Anexo D – LM7805 – Pinagem do Regulador de Tensão LM7805



Apêndice A - Programação do ESP32 (Emissor)

```
#include <Wire.h>

#include <MPU6050.h>

#include <esp_now.h>

#include <WiFi.h>


// REPLACE WITH YOUR RECEIVER MAC Address

uint8_t broadcastAddress[] = {0x94, 0xB9, 0x7E, 0xE5, 0xA4, 0x64};


// Structure example to send data

// Must match the receiver structure

typedef struct struct_message {

float inclinacaoX;

float inclinacaoY;

float velocidade;

int anguloServo;

} struct_message;


// Create a struct_message called myData

struct_message myData;


esp_now_peer_info_t peerInfo;


// callback when data is sent

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
```

```
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

// Configurando o módulo MPU6050
MPU6050 mpu;
void setup() {

  Serial.begin(115200);
  // Inicializando o módulo MPU6050
  Wire.begin();
  mpu.initialize();

   // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
```

```
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}
void loop() {
  // Lendo os valores do acelerômetro
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);

  // Calculando a inclinação do módulo em relação ao eixo X
  float inclinacaoX = atan2(ay, sqrt(pow(ax,2) + pow(az,2))) * 180 / PI;

  // Calculando a inclinação do módulo em relação ao eixo Y
  float inclinacaoY = atan2(-ax, sqrt(pow(ay,2) + pow(az,2))) * 180 / PI;
  float velocidade = map(inclinacaoX, -90, 90, -255, 255);
```

```
  myData.inclinacaoX = inclinacaoX;
  myData.inclinacaoY = inclinacaoY;
  myData.velocidade = velocidade;

  // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(50);
}
```

Apêndice B - Programação do ESP32 (Receptor)

```
#include <esp_now.h>

#include <WiFi.h>

#include <Servo.h>


// Structure example to send data

// Must match the receiver structure

typedef struct struct_message {

float inclinacaoX;

float inclinacaoY;

float velocidade;

int anguloServo;

} struct_message;


// Definindo as portas do Arduino conectadas ao módulo ponte H L293D

int enablePin = 15;

int in1Pin = 18;

int in2Pin = 19;


// Definindo as portas do Arduino conectadas ao servo motor

int servoPin = 23;


// Criando um objeto Servo

Servo servo;

// Create a struct_message called myData
```

```
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
}

void setup() {
  // Configurando as portas como saídas
  pinMode(enablePin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  // Configurando o servo motor
  servo.attach(servoPin);
  servo.write(90);

  // Desligando o motor inicialmente
  digitalWrite(enablePin, LOW);

  // Initialize Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
```

```
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

  // Movimentando o motor de acordo com a inclinação no eixo X
  if(myData.inclinacaoX < -5) {
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
    analogWrite(enablePin, abs(myData.velocidade));
  } else if(myData.inclinacaoX > 5) {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    analogWrite(enablePin, abs(myData.velocidade));
```

```
  } else {
    digitalWrite(enablePin, LOW);
  }

  // Movimentando o servo motor de acordo com a inclinação no eixo Y
  int anguloServo = map(myData.inclinacaoY, -90, 90, 0, 180);
  servo.write(anguloServo);

  delay(20);
}
```

Apêndice C - Tabela de custo

Custo do projeto			
Item	Quantidade	Preço	Total
ESP32	2	R$ 34,20	R$ 68,40
Motor DC	1	R$ 27,00	R$ 27,00
Servomotor MG996R	1	R$ 56,90	R$ 56,90
MPU6050	1	R$ 21,76	R$ 21,76
Ponte H L293D	1	R$ 10,71	R$ 10,71
Placa Perfurada com Trilha	2	R$ 13,54	R$ 27,08
Bateria	2	R$ 29,99	R$ 59,98
Pilhas	8	R$ 5,73	R$ 45,84
Regulador de Tensão 7805	1	R$ 2,92	R$ 2,92
Suporte para pilhas AA	1	R$ 17,01	R$ 17,01
Luva	1	R$ 8,90	R$ 8,90
Veículo	1	R$ 59,99	R$ 59,99
TOTAL			**R$ 406,49**