UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SOLUÇÃO PARA LOCOMOÇÃO DA CADEIRA DE RODAS ATRAVÉS DOS MOVIMENTOS DA CABEÇA

GUSTAVO KLABUNDE

BLUMENAU
2018

GUSTAVO KLABUNDE

SOLUÇÃO PARA LOCOMOÇÃO DA CADEIRA DE RODAS ATRAVÉS DOS MOVIMENTOS DA CABEÇA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer, Mestre - Orientador

BLUMENAU
2018

SOLUÇÃO PARA LOCOMOÇÃO DA CADEIRA DE RODAS ATRAVÉS DOS MOVIMENTOS DA CABEÇA

Por

GUSTAVO KLABUNDE

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

	_____
Presidente:	Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
	_____
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
	_____
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB

Blumenau, 12 de julho de 2018

Dedico este trabalho aos meus pais, minha irmã, amigos e professores, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

A minha família pelo incentivo e confiança que sempre depositaram em mim.

Aos colegas e amigos pelo apoio e parceria em todos os momentos desde o início da minha trajetória no curso de Ciência da Computação até a finalização deste trabalho.

Ao meu orientador Miguel Alexandre Wisintainer, por me auxiliar no desenvolvimento do trabalho, sanando dúvidas e contribuindo com novas ideias.

A empresa Simadri Sistemas pela experiência profissional e pela disponibilidade nos horários.

No meio da dificuldade encontra-se a oportunidade.

Albert Einstein

RESUMO

Este trabalho apresenta o desenvolvimento de uma solução para locomoção da cadeira de rodas através dos movimentos da cabeça, tendo como objetivo facilitar a locomoção de pessoas portadoras de deficiência motora, que possuam apenas os movimentos da cabeça ou dificuldade nos movimentos do segmento mão-braço, apresentando um tipo de controle diferente das cadeiras de rodas elétricas comercializadas. O trabalho é composto por três partes, o protótipo de captura e transmissão dos dados, o protótipo de recepção dos dados e controle dos motores e adaptação mecânica da cadeira de rodas. Os movimentos da cabeça são capturados por um módulo MPU6050. O microcontrolador NodeMCU ESP8266 aplica o filtro complementar nos dados capturados do módulo MPU6050 e transmite os dados para outro microcontrolador NodeMCU ESP8266 através de uma rede Wireless. Os dados recebidos pelo microcontrolador NodeMCU ESP8266 são transmitidos para um Arduino UNO através de uma comunicação Serial. O microcontrolador NodeMCU ESP8266 operando como Access Point hospeda *endpoints* de serviços para calibração dos ângulos mínimos de acionamento da cadeira de rodas. O Arduino UNO processa os dados recebidos e controla os motores através do PWM. O Arduino UNO também verifica através de um módulo HC-SR04 se existem objetos atrás da cadeira de rodas. A solução funcionou corretamente, sendo possível controlar a rotação dos motores através dos movimentos da cabeça e identificar objetos atrás da cadeira de rodas para evitar colisões. A utilização de um filtro complementar combinando os dados do acelerômetro e do giroscópio proporcionou melhora na precisão dos movimentos.

Palavras-chave: Tecnologia assistiva. ESP8266. Acelerômetro. IBT-2.

ABSTRACT

This work presents the development of a solution for locomotion of the wheelchair through the movements of the head, aiming to facilitate the mobility of people with motor disabilities, that only have the movements of the head or difficulty in the movements of the hand-arm segment, presenting a different type of control than commercial electric wheelchairs. The work consists of three parts, the prototype of data capture and transmission, the prototype of data reception and motor control and mechanical adaptation of the wheelchair. The movements of the head are captured by a module MPU6050. The microcontroller NodeMCU ESP8266 applying the complementary filter to the captured data module MPU6050 and transmits the data to another microcontroller NodeMCU ESP8266 through a wireless network. The data received by the microcontroller NodeMCU ESP8266 are transmitted to an Arduino UNO through a Serial Communication. The microcontroller NodeMCU ESP8266 operating as Access Point hosts *endpoints* of services for calibration of the minimum angles of activation of the wheelchair. The Arduino UNO processes the data received and controls the motors through the PWM. The Arduino UNO also checks through a module HC-SR04 if there are objects behind the wheelchair. The solution worked correctly, being possible to control the rotation of the motors through the movements of the head and to identify objects behind the wheelchair to avoid collisions. The use of a complementary filter combining the data of the accelerometer and the gyroscope provided an improvement in the direction of the movements.

Key-words: Assistive technology. ESP8266. Accelerometer. IBT-2.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP - Access Point

bps - Bits Per Second

CSS - Cascading Style Sheets

DC – Direct Current

EEPROM - Electrically Erasable Programmable Read-Only Memory

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

I/O - Input / Output

I2C - Inter-Integrated Circuit

IDE - Integrated Development Environment

IMU - Inertial Measurement Unit

IP - Internet Protocol

JSON - JavaScript Object Notation

LED - Light Emitting Diode

MDF - Medium Density Fiberboard

MHz - Mega Hertz

PCB - Printed Circuit Board

PWM - Pulse Width Modulation

RF - Requisito Funcional

RNF - Requisito Não Funcional

SCL - Serial Clock

SDA - Serial Data

STA - Station

TCP - Transmission Control Protocol

UART - Universal Asynchronous Receiver/Transmitter

UDP - User Datagram Protocol

UML - Unified Modeling Language

URL - Uniform Resource Locator

USB - Universal Serial Bus

V - Volts

Wi-Fi - Wireless Fidelity

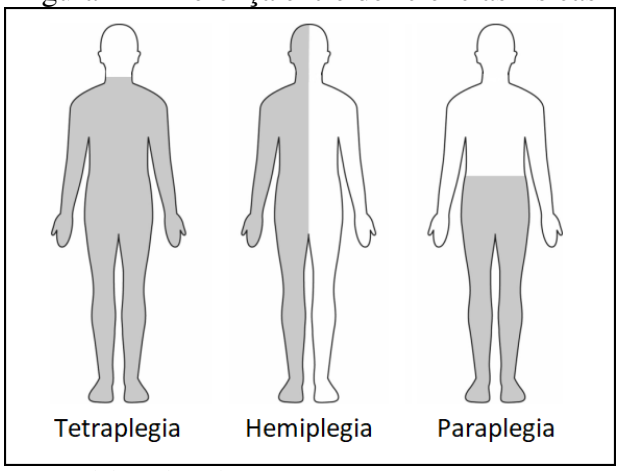
SUMÁRIO

1 INTRODUÇÃO

Dados do último censo do Instituto Brasileiro de Geografia e Estatística (IBGE) revelam que 23,9% da população Brasileira possuem ao menos um tipo de deficiência. A deficiência visual é a que mais está presente na população brasileira, afetando 18,6% da população, em segundo lugar está a deficiência motora, chegando a 7% da população, seguida da deficiência auditiva, com 5,10% e da deficiência mental ou intelectual, com 1,40%. Dos 45 milhões de Brasileiros com deficiência 1,62% não conseguem se locomover. Pessoas com dificuldades para se locomover utilizam acessórios que auxiliam na locomoção, como por exemplo: bengala, andador e cadeira de rodas (SECRETARIA DE DIREITOS HUMANOS DA PRESIDÊNCIA DA REPÚBLICA, 2012).

Segundo Mazzoleni (2017), diversas causas fazem com que uma pessoa precise de uma cadeira de rodas para se locomover, a pessoa pode ter algum controle sobre a perna, mas não ter força para conseguir caminhar sozinha. Outra causa são as deficiências físicas, dentre elas: paraplegia, hemiplegia e tetraplegia. Conforme pode ser observado na Figura 1, na paraplegia ocorre a paralisia da cintura para baixo, na hemiplegia ocorre a paralisia da metade do corpo e na tetraplegia ocorre a paralisia do pescoço para baixo (ASSOCIAÇÃO SALVADOR, 2017).

Figura 1 - Diferença entre deficiências físicas



Fonte: elaborado pelo autor.

A tetraplegia é um dos tipos mais graves de lesão medular. Uma pessoa tetraplégica perde a função motora e ou sensitiva nos segmentos cervicais da medula espinhal (IVO, 1999 apud DEFINO, 2016, p. 13), o que implica no comprometimento das funções motoras nos membros inferiores e superiores (IVO, 2016, p. 13). Segundo Ivo (2016, p. 15), no âmbito comercial não são encontradas no Brasil tecnologias que atendam às necessidades de mobilidade para portadores de tetraplegia.

Diante deste cenário, este trabalho disponibiliza uma solução de locomoção para pessoas com dificuldades na locomoção, que necessitam movimentar uma cadeira de rodas através de ações que não incluam a função motora dos membros inferiores e superiores.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma solução que identifique os movimentos da cabeça de uma pessoa tetraplégica e movimente uma cadeira de rodas com base nos movimentos identificados.

Os objetivos específicos são:

a) identificar os movimentos da cabeça através de um acelerômetro e um giroscópio;
b) transmitir os movimentos identificados para a cadeira de rodas através da radiofrequência;
c) movimentar a cadeira de rodas com base nos movimentos identificados;
d) identificar objetos atrás da cadeira de rodas para evitar colisões.

1.2 ESTRUTURA

O trabalho está organizado em quatro capítulos, sendo neste capítulo apresentada a introdução e os objetivos do trabalho.

O segundo capítulo apresenta a fundamentação teórica, contextualizando os temas abordados no desenvolvimento do trabalho, tais como: dificuldades na locomoção, módulos e sensores, filtro complementar, Pulse Width Modulation e trabalhos correlatos.

No terceiro capítulo é apresentado o desenvolvimento da solução, iniciando com os requisitos e seguido da especificação, implementação e operacionalidade. Por fim, há os resultados obtidos.

O quarto e último capítulo apresenta a conclusões do trabalho juntamente com sugestões de extensões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explorar os principais assuntos necessários para o desenvolvimento deste trabalho. A seção 2.1 aborda as dificuldades na locomoção. Na seção 2.2 são apresentados os módulos e sensores utilizados no projeto. A seção 2.3 aborda o filtro complementar que é utilizado para minimizar ruídos do acelerômetro e giroscópio. A seção 2.4 aborda o conceito e o funcionamento do PWM. Por fim, na seção 2.5 são apresentados os trabalhos correlatos ao trabalho proposto.

2.1 DIFICULDADES NA LOCOMOÇÃO

Segundo Duarte e Cohen (2004, p. 1), aspectos psicoculturais do espaço têm contribuído para o estudo da experiência dos usuários das cidades, subsidiando projetos de arquitetura e de desenho urbano e aperfeiçoando metodologias para aferir o grau de satisfação dos usuários do ambiente urbano. Percebe-se que apesar da produção do espaço urbano continuar sendo feita, características de acessibilidade física de pessoas portadoras de deficiência estão cada vez mais presentes na readequação dos espaços públicos. São consideradas como pessoas com dificuldades na locomoção os dependentes de cadeira de rodas, de muletas, os idosos, as gestantes, os obesos, pessoas com deficiências temporárias, entre outros, que constituem um contingente bastante numeroso de usuários.

A visão de uma pessoa que se locomove em cadeira de rodas é diferente de uma pessoa que se locomove a pé, o ângulo de visão de um cadeirante é a cerca de 1 metro do chão. Sob essa perspectiva, qualquer mobiliário urbano que tenha altura maior que 80 cm se tornará um obstáculo visual para um cadeirante, dificultando a percepção da cidade que só permitirá a visão de objetos próximos e alcançáveis (DUARTE; COHEN, 2004).

Já para a pessoa que precisam de bengalas ou muletas a visão está concentrada em olhar para o chão, apoiando-se em locais firmes. A percepção de espaço é diferente para pessoas que precisam de bengalas ou muletas daquelas que não precisam se preocupar com o equilíbrio, que não se locomovem olhando diretamente para baixo (DUARTE; COHEN, 2004).

Segundo Duarte e Cohen (2004, p. 6), um espaço construído com barreiras poderá estar acentuando a deficiência de uma pessoa com dificuldades na locomoção, aumentando sua dificuldade e tornando-a incapaz de viver uma vida cotidiana ativa, ou seja, muitas das limitações e incapacidades não se devem a uma falta de habilidade de se adaptarem ao ambiente, e sim a falta de adequação do meio. Tudo que afasta uma pessoa com dificuldades

na locomoção poderá ser reduzido se a vida cotidiana urbana for sustentada por uma cidade acessível.

2.2 MÓDULOS E SENSORES

Existem vários módulos e sensores no mercado, nos quais se encaixam conforme as características e finalidades de um projeto. Entre eles o módulo NodeMCU ESP8266, que é um microcontrolador composto por 10 entradas digitais, uma entrada analógica, um regulador de tensão 3,3V, uma interface USB-Serial e um processador ESP8266. A programação pode ser feita usando a linguagem de programação Lua ou C++ na IDE do Arduino, utilizando a comunicação via cabo Micro USB (BERTOLETI, 2016).

O ESP8266 que compõem o microcontrolador NodeMCU ESP8266 é um processador de 32-bits e possui uma frequência de 80MHz, podendo operar em até 160MHz. A tensão de alimentação do ESP8266 é de 3,3 volts. O firmware padrão do ESP8266 vem com um interpretador de comandos AT, para realizar a comunicação com redes Wi-Fi (ACROBOTIC, 2015).

Já um acelerômetro é um equipamento que serve para medir a aceleração de um corpo em relação à gravidade. Esta aceleração é diferente de velocidade, esta aceleração geralmente é mensurada como força *g*, que é basicamente a aceleração sentida como peso. Os tipos mais comuns de acelerômetros são: acelerômetro piezoeléctrico, acelerômetro por indução magnética e acelerômetro de capacitância. A maioria dos acelerômetros possuem três eixos, sendo: x, y e z. O módulo MPU6050 mede a aceleração através destes eixos, sendo assim, possível detectar a orientação do equipamento, tendo como resultado os valores da inclinação e rotação (PAULA, 2015).

Para realizar a comunicação entre um microcontrolador e o módulo MPU6050, é possível utilizar o protocolo Inter-Integrated Circuit (I2C), através desta comunicação, é possível obter a Unidade de Medida Inercial (Inertial Measurement Unit – IMU) do acelerômetro. Para utilizar os dados do acelerômetro em uma aplicação é necessário converter para valores em graus. O Quadro 1 apresenta as formulas utilizadas para calcular o ângulo dos três eixos. É possível medir o ângulo de um eixo, com apenas dois eixos, porém, com três eixos é possível determinar o ângulo com maior precisão. Para calcular a velocidade, é preciso tomar periodicamente medições do acelerômetro e multiplicar exatamente pela diferença de tempo e adicioná-lo à aceleração atual (PAULA, 2015).

Quadro 1 - Fórmulas para conversão dos valores do MPU6050

Eixo	Fórmula
Pitch	$\arctan\left(\frac{x}{\sqrt{z^2+y^2}}\right)$
Roll	$\arctan\left(\frac{y}{\sqrt{z^2+x^2}}\right)$
Yaw	$\arctan\left(\frac{z}{\sqrt{x^2+y^2}}\right)$

Fonte: Paula (2015).

O módulo HC-SR04 é um sensor ultrassônico que permite que você faça leituras de distâncias entre 2 cm e 4 metros, com precisão de 3 mm. O funcionamento do HC-SR04 se baseia no envio de sinais ultrassônicos pelo sensor, que aguarda o retorno do sinal, e com base no tempo entre envio e retorno, calcula a distância. O ângulo para leitura do posicionamento do sensor é de aproximadamente 30º (THOMSEN, 2011).

O processo de medição da distância do modulo HC-SR04 ocorre em 3 etapas: primeiro é enviado um sinal com duração de 10 us (microssegundos) ao pino *trigger* (pino de saída do sensor), indicando que a medição terá início; por segundo, o módulo envia 8 pulsos de 40 KHz e aguarda o retorno do sinal pelo receptor; por terceiro, caso haja um retorno de sinal, é possível determinar a distância utilizando a equação listada no Quadro 2. É necessário a divisão por 2 já que estamos contando o tempo de ida e volta do sinal. O resultado é óbito em metros já que a velocidade do som poder ser considerada igual a 340 m/s (THOMSEN, 2015).

Quadro 2 - Equação de distância do módulo HC-SR04

Distância = (Pulso * Velocidade do som (340)) / 2

Fonte: Thomsen (2015).

2.3 FILTRO COMPLEMENTAR

Quando em movimento, o acelerômetro está sujeito a altos índices de ruídos e o giroscópio tende a acumular erros gerando um escorregamento nas medidas, tornando o tratamento de dados necessário para a diminuição destes erros (BÁSSORA; GOMES, 2014). O filtro complementar tem como objetivo a fusão de dados redundantes ou similares para alcançar uma estimativa ótima de uma determinada variável (BUENO; ROMANO, 2011).

Este filtro opera no domínio da frequência e pode ser definido pelo uso de duas ou mais funções de transferência as quais complementam umas às outras. Num sistema típico de

duas entradas de dados, uma delas proverá informação com ruído de alta frequência, que será filtrada através de um filtro passa-baixas e outra entrada proverá informação com ruído de baixa frequência sendo filtrada por um filtro de passa-altas. Se estes filtros são matematicamente complementares, então o sinal filtrado será a reconstrução completa da variável sendo monitorada, eliminando os ruídos. Este método de tratamento de dados torna os sinais capturados mais próximos da realidade (BUENO; ROMANO, 2011).

Para reduzir o ruído dos dados do acelerômetro e giroscópio o filtro complementar utiliza a soma ponderada de um filtro passa-altas e um filtro passa-baixas, onde em altas frequências utiliza-se os valores de velocidade angular do giroscópio e em baixas frequências utiliza-se os valores do acelerômetro. O valor de qualquer ângulo com o filtro complementar é obtido através das equações listadas no Quadro 3, em que α varia de 0 a 1, ω é a velocidade angular do giroscópio e Δt é o intervalo entre uma amostra e outra (BÁSSORA; GOMES, 2014).

Quadro 3 - Equações do filtro complementar

Ângulo filtrado = α * (Ângulo giroscópio) + (1 – α) * (Ângulo acelerômetro)
Ângulo giroscópio = (Ultimo ângulo filtrado) + ω * Δt

Fonte: Bássora e Gomes (2014).

2.4 PULSE WIDTH MODULATION

Pulse Width Modulation (PWM) refere-se ao conceito de pulsar rapidamente um sinal digital em um condutor. Esta técnica de modulação pode ser utilizada em dispositivos onde é necessário simular uma tensão estática variável, e não apenas ligar e desligar um dispositivo. O PWM é comumente aplicado no controle de motores elétricos, aquecedores, LEDs ou luzes em diferentes intensidades ou frequências (SILVEIRA, 2016).

A técnica do PWM consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto. Esse tempo é chamado de ciclo ativo. A frequência da forma de onda tem o mesmo valor e varia-se o ciclo ativo da forma de onda. Considerando uma tensão máxima de 5 Volts, quando o ciclo ativo está em 0% o valor médio da saída encontra-se em 0V e consequentemente para um ciclo ativo de 100% a saída assume seu valor máximo, que no caso é 5V. Para um ciclo ativo de 50% a saída assumirá 50% do valor da tensão, 2,5V nesse caso e assim sucessivamente para cada variação no ciclo ativo. Para calcular o valor médio da tensão de saída de um sinal PWM pode-se utilizar a equação conforme o Quadro 4, onde Vout é a tensão de saída em Volts, ciclo ativo é o valor do ciclo ativo do PWM em percentual e Vcc é a tensão de alimentação em Volts (SOUZA, 2014).

Quadro 4 - Equação de tensão de saída do PWM

Vout = (ciclo ativo / 100) * *Vcc*

Fonte: Souza (2014).

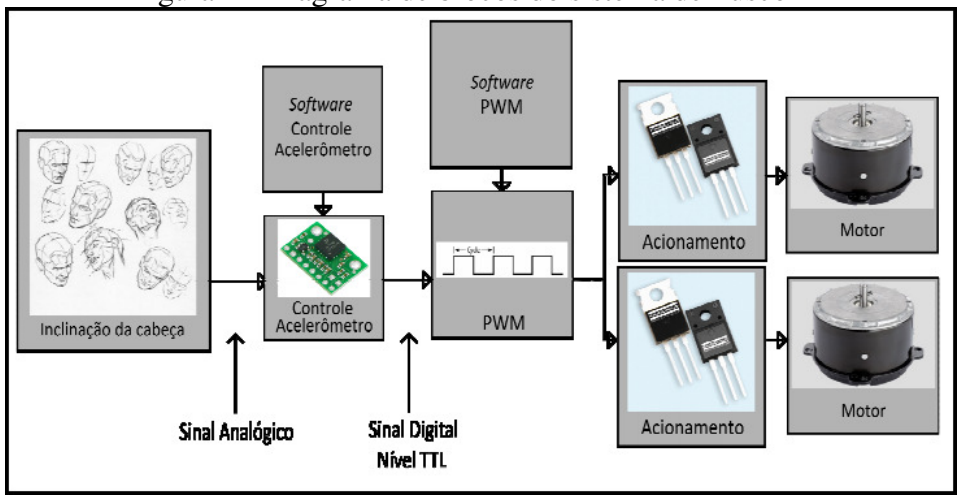
2.5 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos correlatos, que possuem características semelhantes à proposta deste trabalho. Na seção 2.5.1 será apresentado o trabalho de conclusão de curso de Fusco (2010) que consiste no projeto de uma cadeira de rodas controlada por um acelerômetro biaxial. Na seção 2.5.2 será apresentado o trabalho de conclusão de curso de curso de Dellagostin (2011) que consiste em um sistema de acionamento da cadeira de rodas através da inclinação do corpo. E, por fim, na seção 2.5.3 será apresentado o trabalho de conclusão de curso de Ivo (2016) que consiste na solução de uma cadeira de rodas para usuários portadores de tetraplegia.

2.5.1 Acionamento de uma cadeira de rodas através de um acelerômetro bi-axial inclinômetro

Fusco (2010) desenvolveu uma cadeira automatizada controlada por sinais oriundos de um acelerômetro capacitivo biaxial. Para captura dos sinais da inclinação da cabeça o usuário utiliza um boné com um acelerômetro MMA7240L fixado nele. Conforme mostra o diagrama da Figura 2, através do sinal da Modulação de Largura de Pulso (PWM) gerado pelo acelerômetro, juntamente com um microcontrolador AT89S52, é gerado um sinal PWM para os *mosfets* de acionamento, que controlam os motores da cadeira de rodas, dando velocidade e direção a cadeira.

Figura 2 - Diagrama de blocos do sistema de Fusco



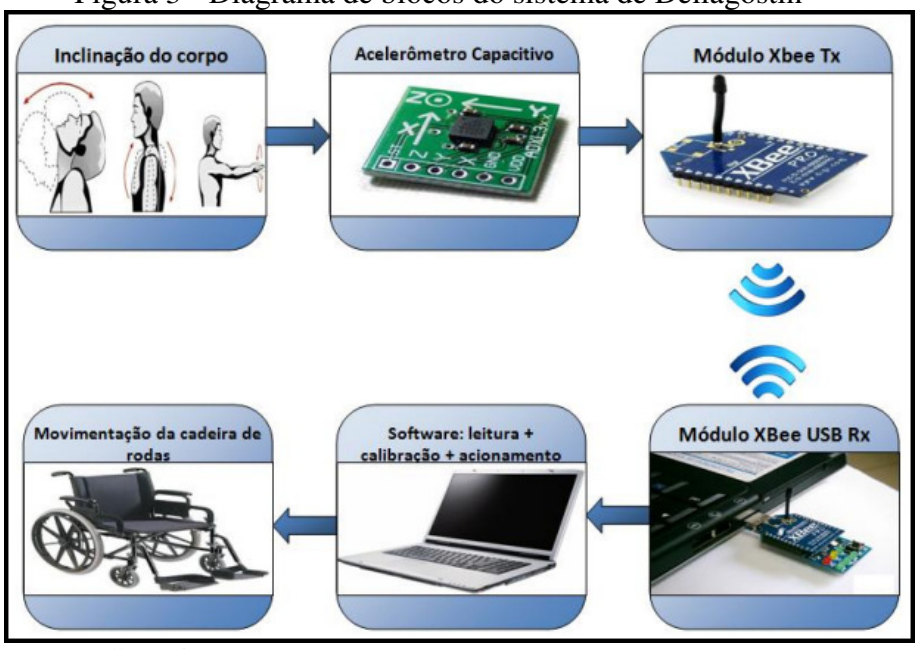
Fonte: Fusco (2010).

Fusco (2010) realizou também um teste funcional com quatro voluntários, onde foi solicitado aos voluntários que calibrassem o sistema e utilizassem a cadeira de rodas. Ele ressalta que o erro máximo encontrado no teste funcional foi de 1,37°, alcançando um erro abaixo do erro máximo teórico, chegando a ser imperceptível pelo usuário (FUSCO, 2010, p. 45). Os motores de redução utilizados por Fusco (2010) apresentaram ótima movimentação com poucos quilos de carga na cadeira de rodas, mais ao realizar testes com maior peso o eixo quebrou (FUSCO, 2010, p.47).

2.5.2 Rede de acelerômetros para tecnologia assistiva

Dellagostin (2011) desenvolveu um sistema de acelerometria sem fio para acionamento de uma cadeira de rodas baseada na inclinação de membros do corpo humano. O sistema desenvolvido consiste em realizar quatro movimentos com a cadeira de rodas de acordo com os movimentos capturados de acelerômetros triaxiais ADXL330 que firam dispostos em três regiões do corpo humano: cabeça, mão esquerda e mão direita. Conforme mostra o diagrama da Figura 3, as informações capturadas do acelerômetro são passadas para o módulo Wireless XBee Pro (transmissor), o qual transmite os dados para um computador que também possui um módulo XBee Pro (receptor) conectado, pelo computador é possível realizar a calibração dos movimentos do sistema. Após a calibração é acionado um protótipo de cadeira de rodas com a execução dos movimentos possíveis (movimentar para frente, para trás, rotacionar a esquerda e a direita) já transmitidos pelo modulo XBee Pro.

Figura 3 - Diagrama de blocos do sistema de Dellagostin



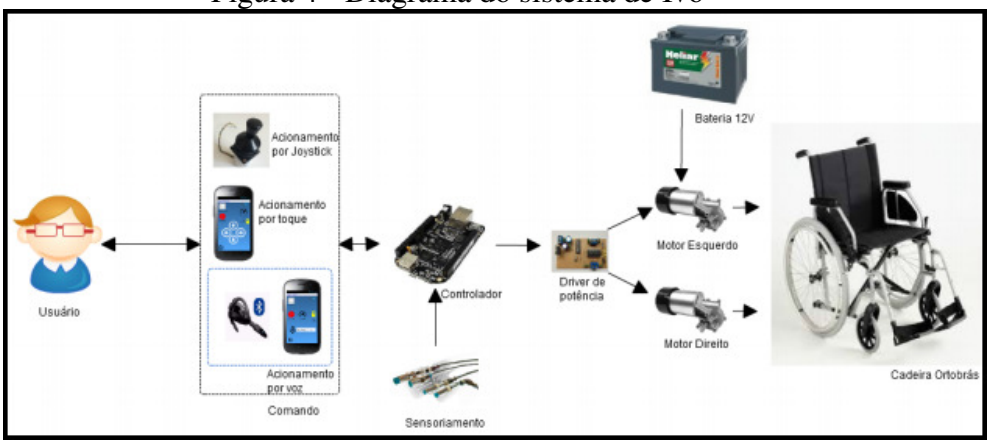
Fonte: Dellagostin (2011).

Segundo Dellagostin (2011), o objetivo estipulado no projeto foi alcançado, que era o desenvolvimento de um sistema de acelerometria sem fio e de baixo custo. Ele também ressalta que a transmissão e recepção dos dados dos acelerômetros apresentaram erros menores que 0,2% (DELLAGOSTIN, 2011, p. 71).

2.5.3 Sistema de controle de cadeira de rodas motorizada para usuários portadores de tetraplegia

O trabalho de Ivo (2016) tinha por objetivo fornecer uma solução de locomoção para usuários tetraplégicos por meio de recursos simples e de baixo custo. Conforme mostra o diagrama da Figura 4, o sistema é acionado através de um comando do usuário, que pode ser por uma das três opções: acionamento com a utilização de um joystick convencional, acionamento com a utilização de uma plataforma de um app-web ativado e gerenciado por toque ou acionamento com a utilização de uma plataforma de um app-web ativado e gerenciado por voz.

Figura 4 - Diagrama do sistema de Ivo



Fonte: Ivo (2016).

As entradas fornecidas pelo usuário e os dados obtidos por meio do sistema de sensoriamento são processados no controlador, o qual está conectado ao driver de potência para acionamento dos motores. O acionamento dos motores ocorre de acordo com as respectivas orientações do controlador, o controle de velocidade é realizado por meio das saídas pré-programadas em lógica baseada em PWM (IVO, 2016, p.46).

Segundo Ivo (2016), o projeto não foi concluído totalmente, a integração final dos módulos do driver de potência, implementação em PCB do driver de potência, implementação do sistema de controle da cadeira, construção de parte do sistema de sensoriamento e execução de rotinas de testes não foram concluídos. Ivo (2016) ressalta que se priorizou o projeto do driver de potência em função da sua essencialidade no sistema.

3 DESENVOLVIMENTO

Neste capítulo, estão descritas as especificações técnicas do protótipo da solução, com o levantamento de requisitos, especificação utilizando diagramas de Unified Modeling Language (UML), o hardware utilizado para o desenvolvimento, uma descrição de técnicas e ferramentas utilizadas, detalhamento dos códigos escritos. O capítulo é finalizado com a análise dos resultados obtidos com o desenvolvimento e testes realizados.

3.1 REQUISITOS

A solução descrita nesse trabalho deverá:

a) movimentar a cadeira de rodas através dos movimentos da cabeça (Requisito Funcional - RF);
b) identificar objetos na parte de trás da cadeira de rodas (RF);
c) permitir a calibração dos ângulos de movimento da cabeça (RF);
d) movimentar suavemente a cadeira de rodas (RF);
e) identificar e eliminar impossíveis movimentos da cabeça (RF);
f) utilizar o microcontrolador NodeMCU ESP8266 (Requisito Não Funcional - RNF);
g) realizar a comunicação sem fio entre os microcontroladores NodeMCU ESP8266 (RNF);
h) utilizar o acelerômetro e giroscópio MPU6050 (RNF);
i) enviar os dados do acelerômetro e giroscópio MPU6050 para o microcontrolador NodeMCU ESP8266 (RNF);
j) utilizar um motor de limpador de para-brisas em cada roda para locomoção da cadeira de rodas (RNF);
k) utilizar o driver motor IBT-2 para controlar os motores (RNF);
l) ser desenvolvida na linguagem C++ através da Arduino IDE (RNF).

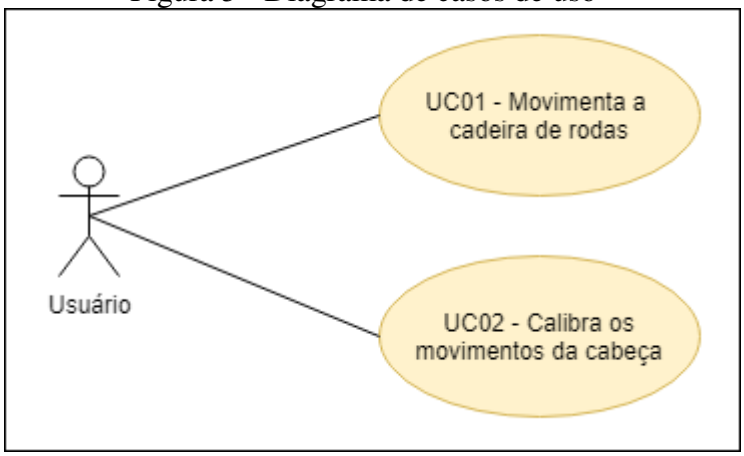
3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação seguindo a metodologia UML na elaboração do diagrama de casos de uso, diagrama de comunicação e diagramas de atividades, elaborados na ferramenta draw.io. Além destes diagramas, também são apresentados os esquemas elétricos do protótipo, elaborados na ferramenta Fritzing.

3.2.1 Diagrama de casos de uso

Na Figura 5 apresenta-se o diagrama de casos de uso do protótipo da solução. O ator `Usuário` é responsável pela usabilidade do protótipo realizando os movimentos com a cabeça que vão acionar a cadeira e rodas, e pela calibração dos movimentos da cabeça. O detalhamento dos casos de uso é apresentado no Apêndice A.

Figura 5 - Diagrama de casos de uso

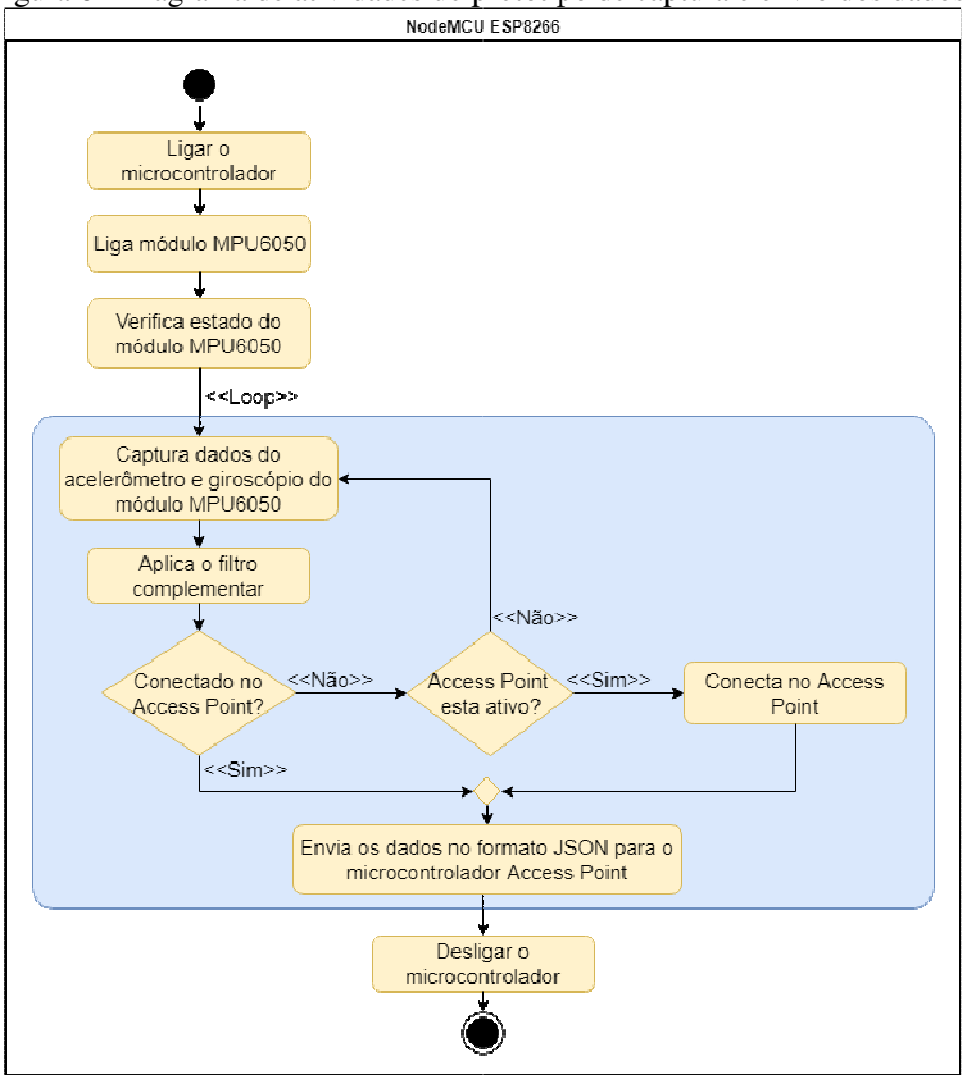


Fonte: elaborada pelo autor.

3.2.2 Diagramas de atividades

Na Figura 6 apresenta-se o diagrama de atividades do protótipo responsável pelo envio e captura dos dados do acelerômetro e giroscópio. Conforme Figura 6, observa-se que após ligar o microcontrolador NodeMCU ESP8266 é feito a inicialização e verificação do módulo MPU6050, responsável por fornecer os dados de aceleração e giro. Seguindo o fluxo de execução existe um *loop*, onde é feito a captura dos dados do modulo MPU6050, aplicado o filtro complementar nesses dados e enviado através da comunicação Wireless UDP para o microcontrolador operando como Access Point. Caso o microcontrolador operando como Access Point não esteja conectado e disponível, os dados são coletados e é aplicado o filtro complementar sobre eles, aumentando a eficiência do filtro, já que o tempo e os dados entre uma aplicação do filtro sobre outra são levados em consideração. O final da execução do software embarcado só ocorre com o desligamento do microcontrolador pelo usuário.

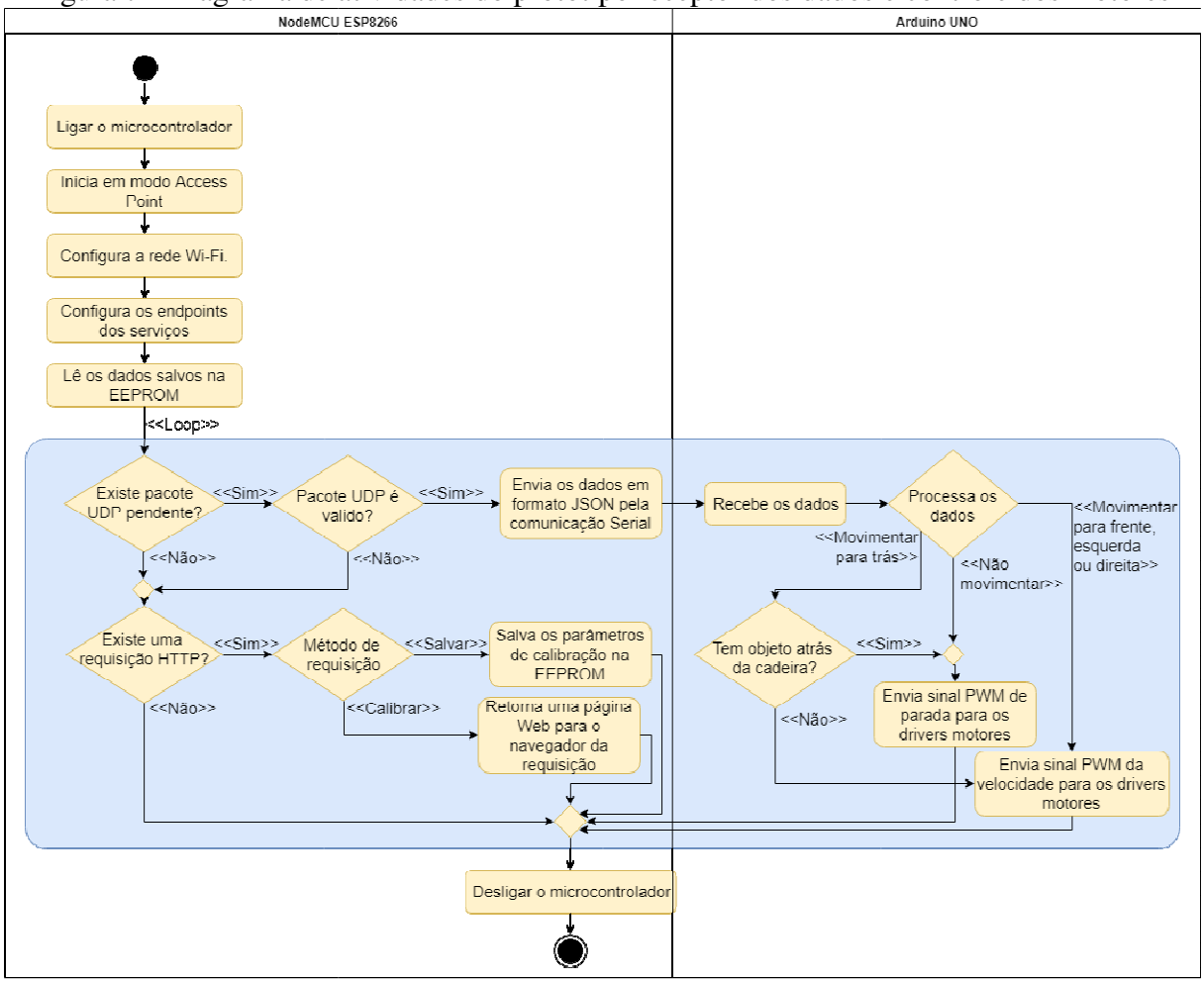
Figura 6 - Diagrama de atividades do protótipo de captura e envio dos dados



Fonte: elaborada pelo autor.

Na Figura 7 apresenta-se o diagrama de atividades do protótipo responsável pela recepção dos dados e controle dos motores. O microcontrolador NodeMCU ESP8266 é inicializado em modo Access Point, depois da inicialização, é realizada a configuração da rede Wi-Fi e dos *endpoints* dos serviços. Antes de entrar no *loop* é feito a leitura dos parâmetros de calibração na memória EEPROM. Em um *loop* de execução, é feito a verificação da existência de pacotes UDP e requisições HTTP. Havendo pacotes UDP pendentes e válidos, os mesmos são repassados juntamente com os parâmetros de calibração para o microcontrolador Arduino UNO através da comunicação Serial UART. Existem duas requisições HTTP possíveis para o protótipo, em uma é retornado uma página Web contendo um formulário para calibração dos ângulos mínimos dos movimentos e em outra é feito uma solicitação para salvar os parâmetros de calibração, recebidos através do método GET. Logo após receber a requisição HTTP para salvar os parâmetros de calibração, eles são salvos na memória EEPROM do microcontrolador NodeMCU ESP8266.

Figura 7 - Diagrama de atividades do protótipo receptor dos dados e controle dos motores



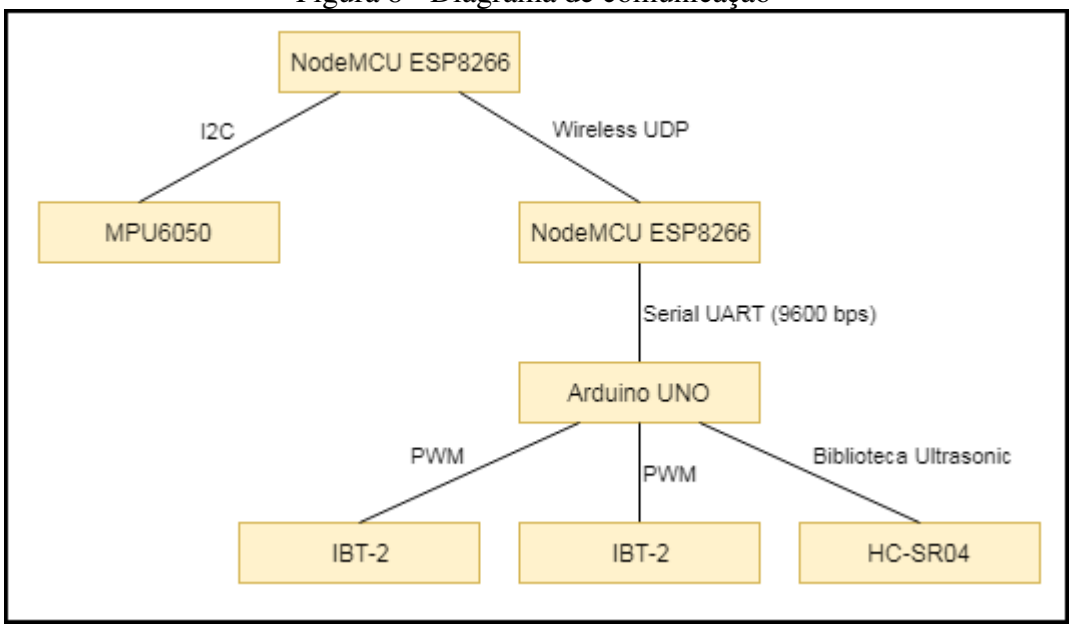
Fonte: elaborada pelo autor.

O microcontrolador Arduino UNO recebe os dados em formato JSON e processa esses dados. Caso o processamento dos dados defina que a ação a ser realizada deverá mover a cadeira de rodas para trás, é necessário que seja verificado a possível existência de objetos atrás da cadeira de rodas através do modulo HC-SR04. Não existindo objetos atrás da cadeira de rodas, será enviado um sinal PWM para cada driver motor, acionando os motores. Caso a ação a ser realizada não seja para mover a cadeira de rodas para trás, não é necessário verificar a existência de objetos atrás da cadeira de rodas, ocorrendo apenas o envio do sinal PWM para cada driver motor. O final do software embarcado ocorre quando os microcontroladores são desligados pelo usuário.

3.2.3 Diagrama de comunicação

Na Figura 8 apresenta-se o diagrama de comunicação, exibindo as formas de comunicação entre todos os componentes do protótipo.

Figura 8 - Diagrama de comunicação



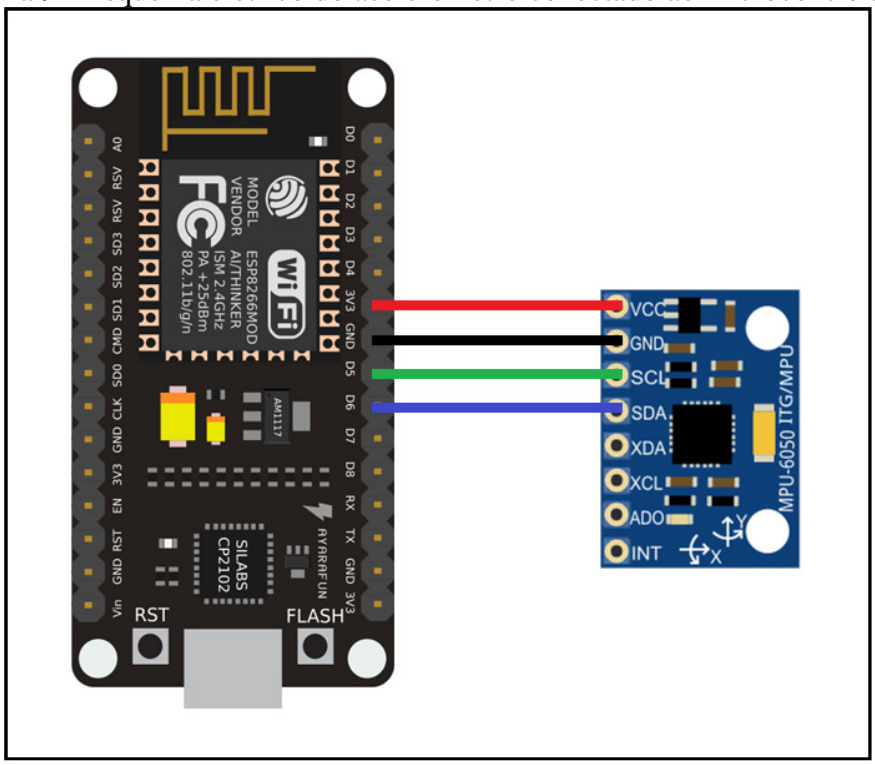
Fonte: elaborada pelo autor.

Como pode-se observar na Figura 8, o NodeMCU ESP8266 se comunica com o acelerômetro MPU6050 através do protocolo de comunicação I2C e com outro NodeMCU ESP8266 através de conexões Wireless UDP. O NodeMCU ESP8266 receptor envia as informações recebidas para o Arduino UNO em um formato JSON através da comunicação Serial UART com uma taxa de transmissão de 9600bps (bits por segundo). O Arduino UNO se comunica com os módulos IBT-2 através do PWM e com o sensor HC-SR04 por meio da biblioteca Ultrasonic, que realiza a comunicação com o sensor, conversão métrica (centímetros e polegadas) e cálculos de desvio padrão.

3.2.4 Esquemas elétricos

O esquema elétrico da solução é divido em duas partes, em uma das partes, consta o esquema elétrico do acelerômetro e do microcontrolador que fazem a captura dos movimentos e os transmitem. Em outra parte, consta o esquema elétrico que recebe os movimentos capturados, realiza o controle dos motores e a verificação de possíveis objetos atrás da cadeira de rodas. A Figura 9 apresenta o esquema elétrico da parte composta pelo acelerômetro e o microcontrolador. O microcontrolador NodeMCU ESP8266 é alimentado pela entrada Micro USB. O acelerômetro MPU6050 é alimentado com 3,3V providos pelo microcontrolador, já que o NodeMCU ESP8266 possui um regulador de voltagem de 5V para 3,3V e opera em 3,3V. A comunicação entre o microcontrolador e o acelerômetro é feita pelo protocolo de comunicação I2C.

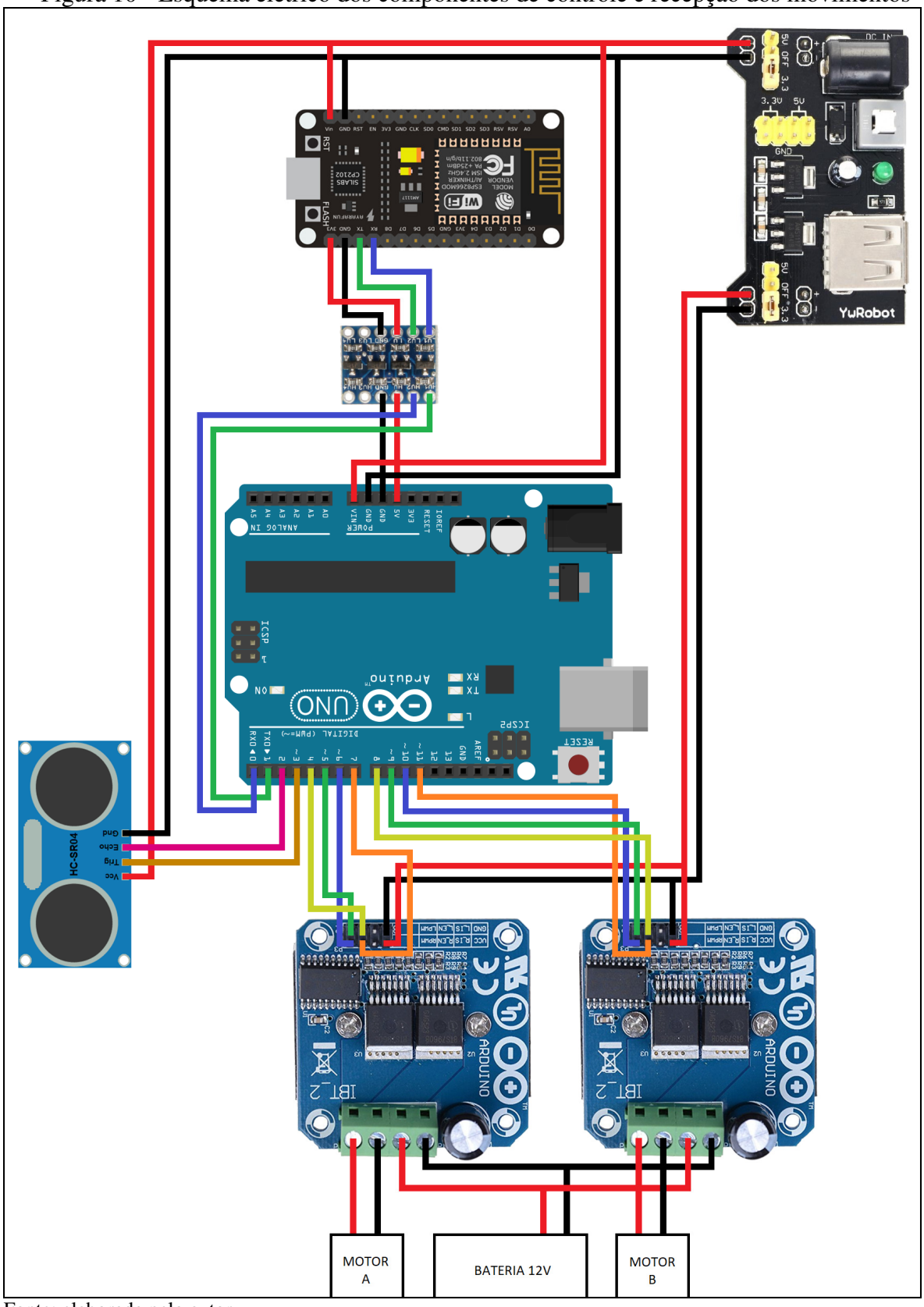
Figura 9 - Esquema elétrico do acelerômetro conectado ao microcontrolador



Fonte: elaborada pelo autor.

A Figura 10 apresenta o esquema elétrico da conexão entre os módulos que fazem a recepção dos movimentos, controle dos motores e verificação de possíveis objetos atrás da cadeira de rodas. O modulo NodeMCU ESP8266 transmite os movimentos para o Arduino UNO através de um conversor de nível lógico bidirecional, pois eles operam em tensões diferentes, o NodeMCU ESP8266 em 3,3V e o Arduino UNO em 5V. O Arduino UNO, o NodeMCU ESP8266, as entradas lógicas dos driver motores IBT-2 e o sensor HC-SR04 são alimentados por uma fonte ajustável 3.3V/5V. A fonte ajustável é alimentada por uma bateria 12V automotiva. A tensão de alimentação dos motores de limpador de para brisas conectados aos driver motores IBT-2 também é feita pela mesma bateria 12V automotiva.

Figura 10 - Esquema elétrico dos componentes de controle e recepção dos movimentos



Fonte: elaborada pelo autor.

3.3 IMPLEMENTAÇÃO

Nesta seção é apresentada a montagem do protótipo. Também são abordadas as implementações dos softwares embarcados nos microcontroladores. A seção é finalizada com a operacionalidade da implementação, apresentando também o processo de calibração dos movimentos.

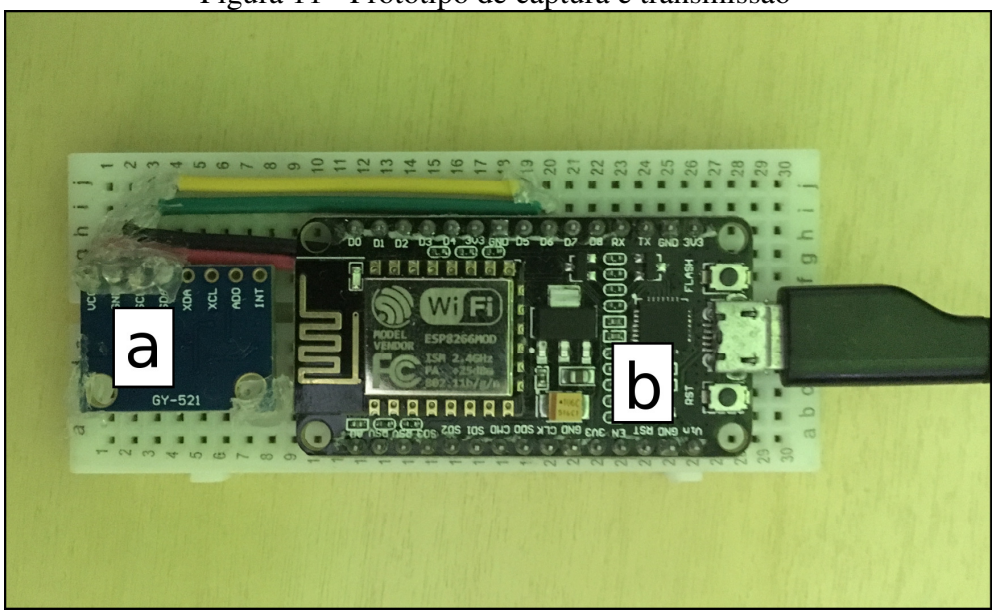
3.3.1 Montagem do protótipo

A montagem do protótipo é dívida em três partes, uma parte compõe a montagem dos componentes responsáveis pela captura e transmissão dos dados, outra parte os componentes de recebimento dos dados e controle dos motores e a última parte com adaptação mecânica na cadeira de rodas. O custo dos componentes utilizados foi disponibilizado no Apêndice B através da Tabela 1.

3.3.1.1 Componentes de captura e transmissão

O protótipo de captura e transmissão dos movimentos foi montado sobre uma protoboard de 400 pontos. A Figura 11 mostra a protoboard juntamente com o microcontrolador NodeMCU ESP8266 e o modulo MPU6050 conectados. A alimentação é feita através de um cabo Micro USB conectado a uma bateria 5V de pequeno porte.

Figura 11 - Protótipo de captura e transmissão



Fonte: elaborada pelo autor.

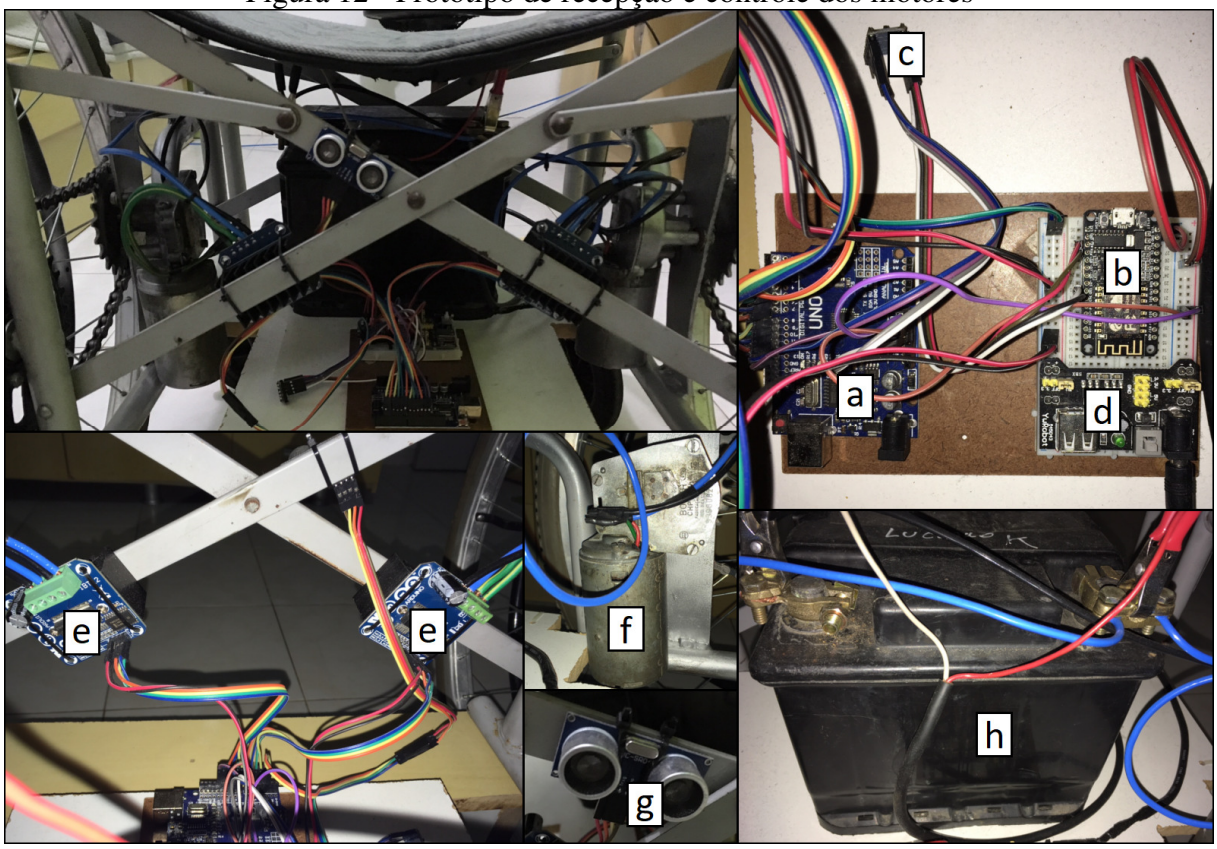
Os itens enumerados na Figura 11 são identificados a seguir:

a) módulo MPU6050: acelerômetro e giroscópio;
b) microcontrolador NodeMCU ESP8266.

3.3.1.2 Componentes de recepção e controle dos motores

O protótipo de recepção dos dados e controle dos motores foi montado sobre uma folha de MDF e dos suportes da cadeira de rodas. A Figura 12 mostra como os microcontroladores Arduino UNO e NodeMCU ESP78266 estão dispostos sobre a placa de MDF, e como os drivers motores, motores e o sensor HC-SR04 estão acoplados nos suportes da cadeira de rodas. A alimentação é feita por uma bateria 12V automotiva, utilizando uma fonte ajustável para alterar a saída para 5V. A bateria e a fonte ajustável também estão sobre a placa de MFD.

Figura 12 - Protótipo de recepção e controle dos motores



Fonte: elaborado pelo autor.

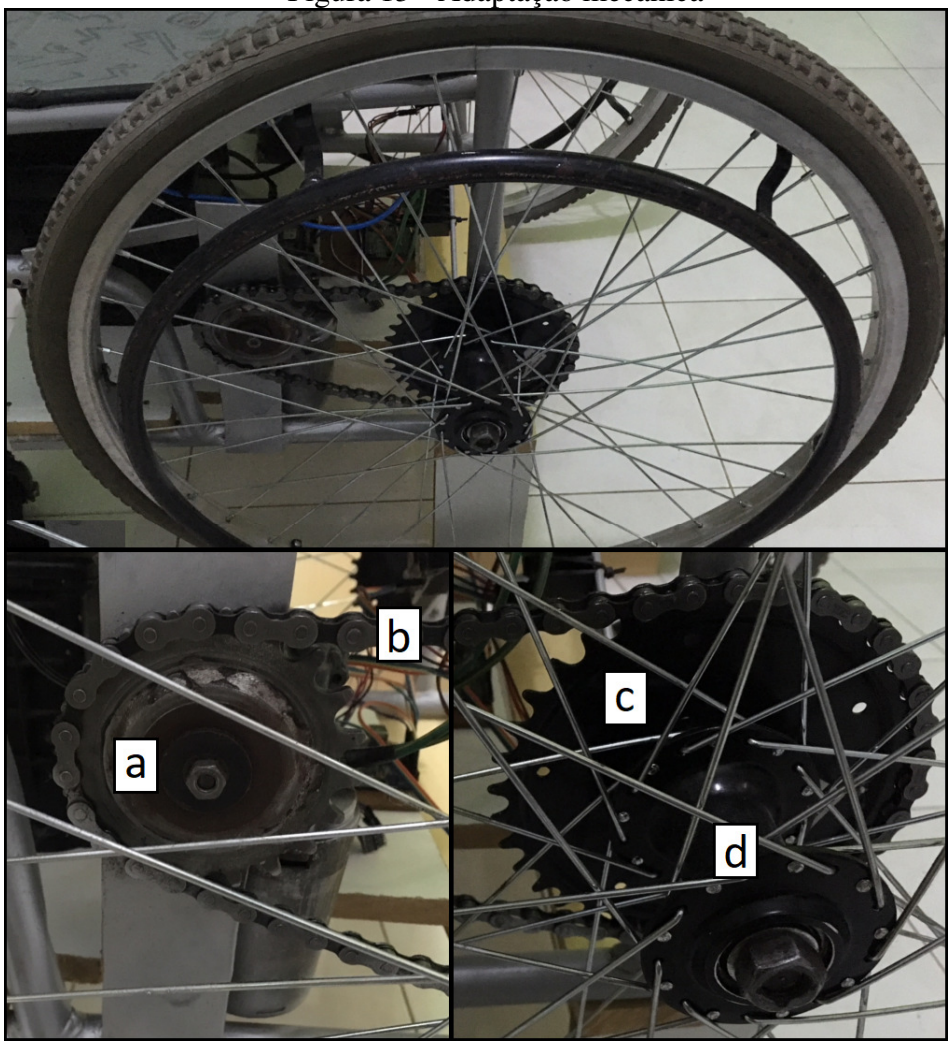
Os itens enumerados na Figura 12 são identificados a seguir:

a) microcontrolador Arduino UNO;
b) microcontrolador NodeMCU ESP8266;
c) conversor de nível lógico bidirecional;
d) fonte ajustável 3.3V/5V
e) driver motor IBT-2;
f) motor de limpador de para brisa;
g) sensor HC-SR04;
h) bateria 12V automotiva.

3.3.1.3 Adaptação mecânica

Conforme apresenta a Figura 13, na adaptação mecânica foi utilizado uma cadeira de rodas comum, não automatizada, que teve os cubos das rodas trocados e feito um suporte para colocar o motor.

Figura 13 - Adaptação mecânica



Fonte: elaborado pelo autor.

Os itens enumerados na Figura 13 são identificados a seguir:

a) pinhão de bicicleta;
b) corrente de bicicleta;
c) coroa de bicicleta;
d) cubo trocado.

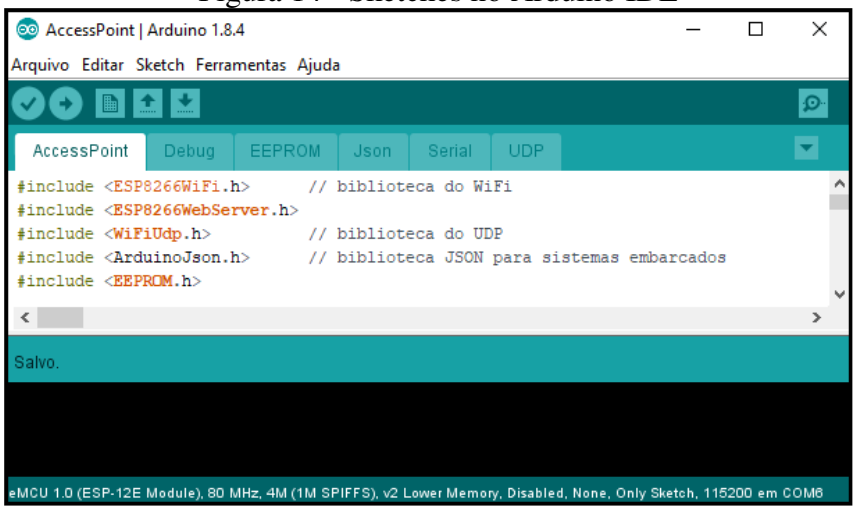
Ao realizar a troca dos cubos para cubos com suporte para freio a disco, foi possível colocar uma coroa de bicicleta e fixa-la com parafusos, pois o disco de freio é similar a uma coroa, sendo necessário apenas fazer os furos na coroa no local dos parafusos. Com a ajuda de um torneiro mecânico foi possível acoplar um pinhão de bicicleta no eixo do motor de

limpador de para brisas e criar um suporte para o motor, deixando o pinhão do motor rente e alinhado a coroa fixada no cubo da roda. Na coroa da roda e no pinhão do motor foi colocado uma corrente de bicicleta, fazendo com que ao rotacionar o eixo do motor a roda seja movida. A troca do cubo e acoplagem do pinhão para o uso de uma corrente se fez necessário para reduzir a velocidade e aumentar o torque.

3.3.2 Técnicas e ferramentas utilizadas

Cada software embarcado no seu dispositivo foi desenvolvido na ferramenta Arduino IDE 1.8.4 com a linguagem de programação C++. Para realizar o *upload* do software para os microcontroladores NodeMCU ESP8266 é necessário instalar o pacote de placas da família ESP8266 no Arduino IDE e selecionar o modelo de placa equivalente. Para realizar a comunicação entre o microcontrolador NodeMCU ESP8266 com o módulo MPU6050 foi utilizado a biblioteca Wire, que permite a comunicação com dispositivos I2C. Visando maior legibilidade e facilidade na escrita do código fonte, ele foi dividido em vários sketches como pode ser observado na Figura 14.

Figura 14 - Sketches no Arduino IDE



Fonte: elaborada pelo autor.

A página Web que contém o formulário para calibrar os movimentos foi desenvolvida na ferramenta Notepad++ com as linguagens de programação HTML e CSS. Para manipular os dados no formato JSON foi utilizada a biblioteca ArduinoJson, que trabalha com os microcontroladores utilizados no protótipo.

3.3.3 Código fonte

Nesta seção são apresentados os principais trechos de código dos dispositivos, ou seja, do software embarcado nos microcontroladores NodeMCU ESP8266 e Arduino UNO.

3.3.3.1 Microcontrolador NodeMCU ESP8266 transmissor

A comunicação com o módulo MPU6050 é realizada através da comunicação via interface I2C, utilizando os pinos analógicos D5 e D6 do microcontrolador, conectados nos pinos SCL e SDA do módulo MPU6050 respectivamente. Para realizar a comunicação via interface I2C é utilizado biblioteca Wire, nativa do Android. O Quadro 5 apresenta o método `initI2C`, que inicia a biblioteca Wire com a comunicação I2C nos pinos definidos nas variáveis `sda_pin` e `scl_pin`.

Quadro 5 - Método que define a comunicação I2C nos pinos analógicos

```
1  #define sda_pin D6 // definição do pino I2C SDA
2  #define scl_pin D5 // definição do pino I2C SCL
3
4  void initI2C() {
5    Wire.begin(sda_pin, scl_pin);
6  }
```

Fonte: elaborado pelo autor.

O Quadro 6 mostra a definição das variáveis de alguns endereços em hexadecimal mais comuns do módulo MPU6050. Esses endereços são utilizados em métodos abaixo para definir qual registro é manipulado.

Quadro 6 - Endereços do módulo MPU6050

```
1  // registro do endereço do sensor MPU6050
2  const int MPU_ADDR = 0x68;
3  // registro de identificação do dispositivo
4  const int WHO_AM_I = 0x75;
5  // registro de configuração do gerenciamento de energia
6  const int PWR_MGMT_1 = 0x6B;
7  // registro de configuração do giroscópio
8  const int GYRO_CONFIG = 0x1B;
9  // registro de configuração do acelerômetro
10 const int ACCEL_CONFIG = 0x1C;
11 // registro de leitura dos eixos do modulo
12 const int ACCEL_XOUT = 0x3B;
```

Fonte: elaborado pelo autor.

OQuadro 7 - Métodos de inicialização do modulo MPU6050 Quadro 7 apresenta os métodos de inicialização do modulo MPU6050. Entre as linhas 1 e 5 contém o método `initMPU`, que realiza a chamada para os métodos de configuração do gerenciamento de energia, configuração do acelerômetro e configuração do giroscópio. O método `setSleepOff` escreve no registro `0x6B` o valor 0, colocando o modulo em modo ativo. O método `setGyroScale` escreve no registro `0x1B` o valor 0, definindo a escala do giroscópio em 250°. O método `setAccelScale` escreve no registro `0x1C` o valor 0, definindo a escala do acelerômetro para 2g (8192 LSB/mg).

Quadro 7 - Métodos de inicialização do modulo MPU6050

```
 1  void initMPU() {
 2    setSleepOff();
 3    setGyroScale();
 4    setAccelScale();
 5  }
 6
 7  void writeRegMPU(int reg, int val) {
 8    // inicia comunicação com endereço do MPU6050
 9    Wire.beginTransmission(MPU_ADDR);
10    // envia o registro que deseja manipular
11    Wire.write(reg);
12    // escreve o valor no registro
13    Wire.write(val);
14    // termina a transmissão
15    Wire.endTransmission(true);
16  }
17
18  void setSleepOff() {
19    writeRegMPU(PWR_MGMT_1, 0);
20  }
21
22  void setGyroScale() {
23    writeRegMPU(GYRO_CONFIG, 0);
24  }
25
26  void setAccelScale() {
27    writeRegMPU(ACCEL_CONFIG, 0);
28  }
```

Fonte: elaborado pelo autor.

O Quadro 8 apresenta o método `checkMPU`, responsável por fazer as verificações no módulo MPU6050. Através do método `findMPU` é possível verificar se o modulo é encontrado no registro `0x68`. Entre as linhas 6 e 7 é realizado a leitura no registro `0x75` e feito a verificação para identificar se o modulo está disponível. Entre as linhas 10 e 11 é realizado a leitura no registro `0x6B` para identificar se o modulo está ativo.

Quadro 8 - Métodos de verificação do módulo MPU6050

```
1  void checkMPU(int MPU_ADDR) {
2    if(findMPU(MPU_ADDR)) {
3      Serial.print("Device found at address: ");
4      Serial.println(MPU_ADDR, HEX);
5      // verifica o registro de identificação do dispositivo
6      int data = readRegMPU(WHO_AM_I);
7      if(data == 104) {
8        Serial.println("MPU6050 available, answered OK! (104)");
9        // verifica o registro de gerenciamento de energia
10       data = readRegMPU(PWR_MGMT_1);
11       if(data == 64) {
12         Serial.println("MPU6050 in SLEEP mode! (64)");
13       } else {
14         Serial.println("MPU6050 in ACTIVE mode! (Ativo)");
15       }
16     } else {
17       Serial.println("MPU6050 not available!");
18     }
19   } else {
20     Serial.println("Device not found!");
21   }
22 }
23
24 uint8_t readRegMPU(uint8_t reg) {
25   uint8_t data;
26   // inicia comunicação com endereço do MPU6050
27   Wire.beginTransmission(MPU_ADDR);
28   // envia o registro que deseja manipular
29   Wire.write(reg);
30   // termina a transmissão
31   Wire.endTransmission(false);
32   // configura para receber 1 byte do registro escolhido acima
33   Wire.requestFrom(MPU_ADDR, 1);
34   // lê o byte e armazena na variável 'data'
35   data = Wire.read();
36   return data;
37 }
38
39 boolean findMPU(int MPU_ADDR) {
40   Wire.beginTransmission(MPU_ADDR);
41   int data = Wire.endTransmission(true);
42   if(data == 0) {
43     return true;
44   } else {
45     return false;
46   }
47 }
```

Fonte: elaborado pelo autor.

No Quadro 9 é apresentado o método `readRawMPU`, responsável pela captura dos dados de acelerômetro, giroscópio e temperatura do modulo MPU6050. Entre as linhas 5 e 9 é configurado o registro `0x3B` para retornar 14 bytes, 2 bytes para cada eixo do acelerômetro, 2 para cada eixo do giroscópio e 2 bytes para a temperatura. Entre as linhas 11 e 26 é feita a divisão desses bytes para sua respectiva variável.

Quadro 9 - Método de captura dos dados do módulo MPU6050

```
1  // variáveis para armazenar os dados do módulo
2  int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
3
4  void readRawMPU() {
5    Wire.beginTransmission(MPU_ADDR);
6    Wire.write(ACCEL_XOUT);
7    Wire.endTransmission(false);
8    // configura para receber 14 bytes do registro
9    Wire.requestFrom(MPU_ADDR, 14);
10
11   AcX = Wire.read() << 8;
12   AcX |= Wire.read();
13   AcY = Wire.read() << 8;
14   AcY |= Wire.read();
15   AcZ = Wire.read() << 8;
16   AcZ |= Wire.read();
17
18   Tmp = Wire.read() << 8;
19   Tmp |= Wire.read();
20
21   GyX = Wire.read() << 8;
22   GyX |= Wire.read();
23   GyY = Wire.read() << 8;
24   GyY |= Wire.read();
25   GyZ = Wire.read() << 8;
26   GyZ |= Wire.read();
   }
```

Fonte: elaborado pelo autor.

No Quadro 10 é apresentado o código responsável por aplicar o filtro complementar nos valores brutos lidos do módulo MPU6050. Antes de aplicar o filtro é necessário converter os valores brutos para ângulo. A conversão é realizada através do cálculo exposto entre as linhas 10 e 12 para os valores do giroscópio e para os valores do acelerômetro é utilizado o cálculo no método `convertAccelToAngle`. O valor do ângulo do giroscópio utilizado para aplicar o filtro complementar é baseado no tempo delta, no ângulo do giroscópio lido e nos valores do último filtro aplicado. A aplicação do filtro complementar é realizada entre as linhas 29 e 33 com base nos valores do ângulo do giroscópio e acelerômetro, sendo que, o giroscópio tem o fator de confiança de 96% e o acelerômetro de 4%.

Quadro 10 - Filtro complementar

```
1  float RAD_TO_DEGREE = 180/3.14159;
2  float FS_SEL = 131; // sensibilidade do giroscópio
3
4  float convertAccelToAngle(int a, int b, int c) {
5    return RAD_TO_DEGREE * atan(a / sqrt(pow(b, 2) + pow(c, 2)));
6  }
7
8  void filter(unsigned long now) {
9    // conversão dos valores brutos do giroscópio para ângulos
10   float gyro_x = (GyX - base_x_gyro) / FS_SEL;
11   float gyro_y = (GyY - base_y_gyro) / FS_SEL;
12   float gyro_z = (GyZ - base_z_gyro) / FS_SEL;
13
14   // conversão os valores brutos do acelerômetro para ângulos
15   float accel_angle_x = convertAccelToAngle(AcY, AcX, AcZ);
16   float accel_angle_y = convertAccelToAngle(AcX, AcY, AcZ);
17   float accel_angle_z = convertAccelToAngle(AcZ, AcX, AcY);
18
19   /* calcula o ângulo do giroscópio com base no tempo delta, no ângulo
20    * do giroscópio lido e nos últimos valores filtrados */
21   float dt = (now - last_read_time) / 1000.0;
22
23   float gyro_angle_x = gyro_x * dt + last_x_angle;
24   float gyro_angle_y = gyro_y * dt + last_y_angle;
25   float gyro_angle_z = gyro_z * dt + last_z_angle;
26
27   /* aplica o filtro complementar, com o fator de confiança de 0.96
28    * para o giroscópio e 0.04 para o acelerômetro */
29   float alpha = 0.92;
30
31   float angle_x = alpha * gyro_angle_x + (1.0 - alpha) * accel_angle_x;
32   float angle_y = alpha * gyro_angle_y + (1.0 - alpha) * accel_angle_y;
33   float angle_z = alpha * accel_angle_y + (1.0 - alpha) * gyro_angle_z;
34
35   // atualiza os últimos valores filtrados
36   set_last_read_angle_data(now, angle_x, angle_y, angle_z);
37 }
```

Fonte: elaborado pelo autor.

O Quadro 11 apresenta os métodos de conexão à rede Wi-Fi do AP receptor dos dados. O método `initWiFi` é responsável por colocar o microcontrolador em modo STA e conectar à rede Wi-Fi do AP. O método `connect` verifica se o microcontrolador está conectado no AP, caso não esteja conectado ele realizar a conexão e aguarda 2000 milissegundos. O método `connect` é executado antes do envio dos dados.

Quadro 11 - Métodos de conexão à rede Wi-Fi

```
1  const char *ssid = "Wheelchair";
2  const char *password = "**********";
3
4  void initWiFi() {
5    WiFi.mode(WIFI_STA);
6    WiFi.begin(ssid, password); // conecta à rede Wi-Fi
7
8    while(WiFi.status() != WL_CONNECTED) {
9      delay(500);
10     Serial.print(".");
11   }
12
13   Serial.println("");
14   Serial.println("Wi-Fi connected");
15   Serial.print("IP address: ");
16   Serial.println(WiFi.localIP());
17 }
18
19 void connect() {
20   if (WiFi.status() != WL_CONNECTED) {
21     WiFi.begin(ssid, password); // conecta à rede Wi-Fi
22     delay(2000); // espera até que a conexão seja feita
23   }
24 }
```

Fonte: elaborado pelo autor.

O Quadro 12 apresenta os métodos responsáveis pelo envio dos dados em formato JSON para a porta UDP do microcontrolador operando como AP. O método `send` faz o envio dos dados. Na linha 15 é especificado o endereço de IP e porta UDP do AP. Entre as linhas 1 e 4 está a definição das variáveis para a construção do JSON. O método `populateJSON` armazena nas variáveis `x` e `y` os valores dos eixos x e y resultantes do filtro complementar aplicado.

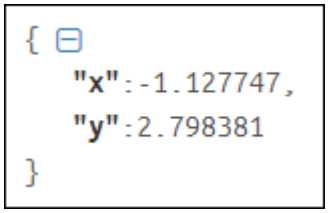
Quadro 12 - Métodos de envio do JSON para a porta UDP

```
1  StaticJsonBuffer<100> jsonBuffer;
2  JsonObject& object = jsonBuffer.createObject();
3  JsonObject& x = object.createNestedObject("x");
4  JsonObject& y = object.createNestedObject("y");
5
6  void populateJSON() {
7    object["x"] = last_x_angle;
8    object["y"] = last_y_angle;
9  }
10
11 void send() {
12   // só irá enviar os dados se estiver conectado à rede Wi-Fi
13   if (WiFi.status() == WL_CONNECTED) {
14     // inicializa o pacote de transmissão ao IP e PORTA
15     udp.beginPacket("192.168.4.1", 555);
16     // monta o JSON e adiciona ao pacote de transmissão
17     object.printTo(udp);
18     // finaliza o pacote e envia
19     udp.endPacket();
20   } else {
21     // pisca LED do microcontrolador
22     digitalWrite(LED_BUILTIN, LOW);
23     delay(300);
24     digitalWrite(LED_BUILTIN, HIGH);
25   }
26 }
```

Fonte: elaborado pelo autor.

A Figura 15 apresenta o texto no formato JSON enviado para a porta UDP do microcontrolador receptor. O JSON é composto por duas coleções de pares. A coleção com identificação de nome x apresenta o valor do ângulo do eixo x e a coleção de nome y apresenta o valor do ângulo do eixo y, ambos os eixos com o filtro complementar aplicado.

Figura 15 - JSON dos dados enviados



Fonte: elaborada pelo autor.

3.3.3.2 Microcontrolador NodeMCU ESP8266 receptor

Segundo Minatel (2016), EEPROM é uma memória não volátil com um limite de 10 mil gravações e infinitas leituras para cada Byte (no NodeMCU ESP8266). O NodeMCU ESP8266 não tem EEPROM dedicada, porém, há uma emulação em um segmento da memória flash de 4B até 4096KB. Normalmente a memória EEPROM é utilizada para armazenar dados sobre a inicialização do sistema ou algo relacionado que não seja necessário ser alterado com muita frequência.

A memória EEPROM é utilizada no protótipo para armazenar os dados dos parâmetros de calibração dos movimentos e a distância máxima (em centímetros) permitida para evitar colisão com objetos atrás da cadeira de rodas. O Quadro 13 apresentam os métodos responsáveis por fazer a leitura e gravação dos dados na EEPROM. Entre as linhas 1 e 11 é possível observar o método `saveEEPROM` que grava os dados na EEPROM e entre as linhas 13 e 22 o método `readEEPROM` que faz a leitura dos dados. Para realizar a manipulação dos dados da memória EEPROM é utilizado a biblioteca EEPROM nativa do Arduino.

Quadro 13 - Métodos de manipulação da EEPROM

```
 1  void saveEEPROM(int front, int back, int left, int right, int distance){
 2    EEPROM.write(0, front);
 3    EEPROM.write(1, back);
 4    EEPROM.write(2, left);
 5    EEPROM.write(3, right);
 6    EEPROM.write(4, distance);
 7    EEPROM.commit();
 8
 9    // povoa o JSON com os dados que foram salvos na EEPROM
10    populateJSON(front, back, left, right, distance);
11  }
12
13  void readEEPROM(){
14    int front = EEPROM.read(0);
15    int back = EEPROM.read(1);
16    int left = EEPROM.read(2);
17    int right = EEPROM.read(3);
18    int distance = EEPROM.read(4);
19
20    // povoa o JSON com os dados da EEPROM
21    populateJSON(front, back, left, right, distance);
22  }
```

Fonte: elaborado pelo autor.

O Quadro 14 apresenta os métodos de hospedagem dos *endpoints* no microcontrolador receptor e inicialização do microcontrolador em modo AP. Na linha 4 é realizada a definição do servidor web para requisições HTTP na porta passada por parâmetro. O método `initAP` é responsável por colocar o microcontrolador em modo AP e definir as credenciais de acesso à rede Wi-Fi. Entre as linhas 10 e 12 temos a definição da URL de acionamento de cada método. A URL raiz vai acionar o método `handleRoot`, que vai retornar uma página web contendo um formulário para calibrar o ângulo mínimo dos movimentos da cabeça. A URL `save` vai acionar o método `handleSave`, que vai salvar os valores recebidos da chamada GET na memória EEPROM. Qualquer outra URL vai acionar o método `handleNotFound`, que vai retornar um texto plano dizendo que esta página não foi encontrada.

Quadro 14 - Métodos de hospedagem dos *endpoints*

```
1  const char *ssid = "Wheelchair";
2  const char *password = "**********";
3
4  ESP8266WebServer server(80);
5
6  void initAP() {
7    WiFi.mode(WIFI_AP);
8    WiFi.softAP(ssid, password);
9
10   server.on("/", handleRoot);
11   server.on("/save", handleSave);
12   server.onNotFound(handleNotFound);
13   server.begin();
14 }
15
16 void handleRoot() {
17   server.send(200, "text/html", prepareHtmlPage());
18 }
19
20 void handleNotFound(){
21   server.send(404, "text/plain", "Page not found");
22 }
23
24 void handleSave() {
25   if(server.args() == 5){
26     saveEEPROM(server.arg("front").toInt(),
27                server.arg("back").toInt(),
28                server.arg("left").toInt(),
29                server.arg("right").toInt(),
30                server.arg("distance").toInt());
31     server.send(200, "text/plain", "Saved parameters");
32     printConfig = true;
33   } else {
34     server.send(200, "text/plain", "Number of invalid parameters");
35   }
36 }
```

Fonte: elaborado pelo autor.

O Quadro 15 apresenta o método `initUDP`, que inicializa a recepção de dados no protocolo UDP na porta passada por parâmetro.

Quadro 15 - Método de inicialização do protocolo UDP

```
1  // cria um objeto da classe UDP
2  WiFiUDP udp;
3
4  void initUDP(){
5    udp.begin(555);
6  }
```

Fonte: elaborado pelo autor.

O Quadro 16 apresenta o método `listen`, que verifica se há pacotes UDP pendentes. Entre as linhas 4 e 7 é feito a leitura dos caracteres recebidos no pacote UDP. Entre as linhas 8 e 13 é feito o envio dos dados pela comunicação Serial com o Arduino UNO, a variável `printConfig` é responsável pelo controle do envio dos parâmetros de calibração dos movimentos.

Quadro 16 - Método de recepção dos dados UDP

```
 1  void listen() {
 2    if (udp.parsePacket() > 0) {
 3      String json = "";
 4      while (udp.available() > 0) {
 5        char z = udp.read();
 6        json += z;
 7      }
 8      if(printConfig){
 9        object.printTo(Serial);
10        Serial.println();
11        printConfig = false;
12      }
13      Serial.println(json);
14    }
15  }
```

Fonte: elaborado pelo autor.

3.3.3.3 Microcontrolador Arduino UNO

A comunicação com o sensor HC-SR04 é realizada através dos pinos 2 e 3 do microcontrolador, conectados nos pinos *echo* e *trigger* do sensor. Para realizar a comunicação com o sensor é utilizado a biblioteca Ultrasonic. O Quadro 17 apresenta a definição dos pinos *echo* e *trigger* nas linhas 1 e 2, na linha 4 apresenta-se a inicialização da biblioteca Ultrasonic. O método `getDistanceCm` retorna a distância em centímetros de um objeto em frente o ângulo do sinal do sensor (precisão do sensor de 3 milímetros).

Quadro 17 - Método de retorno da distância do sensor HC-SR04

```
1  #define pino_echo 2    // definição do pino echo
2  #define pino_trigger 3 // definição do pino trigger
3
4  Ultrasonic ultrasonic(pino_trigger, pino_echo);
5
6  float getDistanceCm() {
7    return ultrasonic.convert(ultrasonic.timing(), Ultrasonic::CM);
8  }
```

Fonte: elaborado pelo autor.

O Quadro 18 apresenta o método `serialParser` que é executado em um loop infinito pelo microcontrolador Arduino UNO. O método é responsável por fazer a leitura dos dados que são recebidos na porta Serial e processá-los. Entre as linhas 1 e 2 está a definição dos pinos RX e TX da comunicação Serial. Na linha 10 ocorre a leitura de uma cadeia de caracteres até que seja encontrado uma quebra de linha. Na linha 13 é verificado se a cadeia de caráteres recebida é de dados de configuração ou de movimento. Caso não seja recebido nenhum caractere em um período de 1000 milissegundos, é enviado o comando para realizar a parada dos motores, os dados de movimento são recebidos repetidamente na média de 40 milissegundos.

Quadro 18 - Método de leitura da comunicação Serial

```
1  #define pino_rx 12 // definição do pino RX
2  #define pino_tx 13 // definição do pino TX
3
4  const int bSize = 60;
5  char Buffer[bSize];
6  int ByteCount;
7
8  void serialParser() {
9    ByteCount = -1;
10   ByteCount = SerialESP.readBytesUntil('\n',Buffer,bSize);
11   SerialESP.flush();
12   if (ByteCount  > 0) {
13     if(String(Buffer).substring(2,6) == "conf"){
14       setConfig(Buffer);
15     } else {
16       setAction(Buffer);
17     }
18   } else {
19     // atingido o tempo limite, parar motores
20     toMove(false,false,false,false);
21   }
22   memset(Buffer, 0, sizeof(Buffer));
23 }
```

Fonte: elaborado pelo autor.

A movimentação da cadeira de rodas é feita com base na verificação dos valores recebidos do acelerômetro após o filtro complementar aplicado. Esses dados são comparados com os limites dos ângulos mínimos dos movimentos da cabeça. Caso os valores recebidos ultrapassem algum dos valores dos ângulos mínimos, é feito o acionamento dos motores rotacionando para o lado referente ao ângulo limite que foi ultrapassado.

O controle dos motores é feito através do PWM, com o PWM é possível controlar a velocidade dos motores. O Quadro 19 apresenta os métodos que fazem o controle dos motores. Entre as linhas 1 e 8 está a definição dos pinos que estão conectados nos drivers motores. O método `motorActiveStatus` define o status de cada direção dos motores (sentido horário e anti-horário). O método `setMotor` recebe como parâmetro o motor, direção e o PWM da velocidade, e transmite para o pino referente ao motor a direção e a velocidade em que o motor deverá mover-se. O método `closeMotor` para o motor passado por parâmetro.

Quadro 19 - Métodos de controle dos motores

```
 1  #define MOTOR_1_L_EN 4
 2  #define MOTOR_1_LPWM 5
 3  #define MOTOR_1_RPWM 6
 4  #define MOTOR_1_R_EN 7
 5  #define MOTOR_2_L_EN 8
 6  #define MOTOR_2_LPWM 9
 7  #define MOTOR_2_RPWM 10
 8  #define MOTOR_2_R_EN 11
 9
10  void motorActiveStatus(int motor, char side, boolean s) {
11    if(motor == 1 && side == 'R') {
12      digitalWrite(MOTOR_1_R_EN,s);
13    }
14    if(motor == 1 && side == 'L') {
15      digitalWrite(MOTOR_1_L_EN,s);
16    }
17    if(motor == 2 && side == 'R') {
18      digitalWrite(MOTOR_2_R_EN,s);
19    }
20    if(motor == 2 && side == 'L') {
21      digitalWrite(MOTOR_2_L_EN,s);
22    }
23  }
24
25  void setMotor(int motor, char side, byte pwm) {
26    if(motor == 1 && side == 'R') {
27      analogWrite(MOTOR_1_RPWM,pwm);
28    }
29    if(motor == 1 && side == 'L') {
30      analogWrite(MOTOR_1_LPWM,pwm);
31    }
32    if(motor == 2 && side == 'R') {
33      analogWrite(MOTOR_2_RPWM,pwm);
34    }
35    if(motor == 2 && side == 'L') {
36      analogWrite(MOTOR_2_LPWM,pwm);
37    }
38  }
39
40  void closeMotor(int motor, char side) {
41    if(motor == 1 && side == 'R') {
42      digitalWrite(MOTOR_1_RPWM,LOW);
43    }
44    if(motor == 1 && side == 'L') {
45      digitalWrite(MOTOR_1_LPWM,LOW);
46    }
47    if(motor == 2 && side == 'R') {
48      digitalWrite(MOTOR_2_RPWM,LOW);
49    }
50    if(motor == 2 && side == 'L') {
51      digitalWrite(MOTOR_2_LPWM,LOW);
52    }
53  }
```

Fonte: elaborado pelo autor.

3.3.4 Operacionalidade da implementação

A primeira etapa envolve o processo de calibração dos ângulos mínimos de acionamento dos movimentos. Cada usuário deve calibrar os ângulos mínimos de

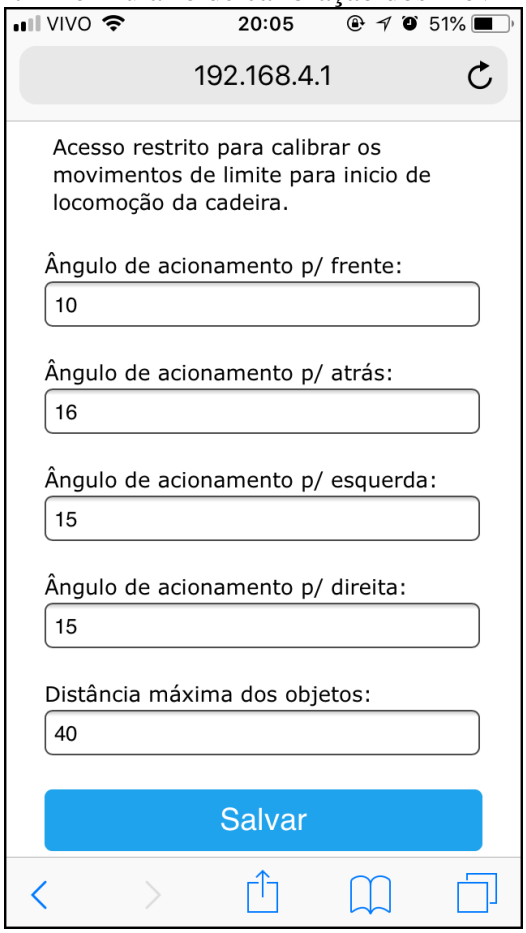
acionamento conforme a medida de ângulo adequada para o início dos movimentos. Para realizar o processo de calibração o usuário deve estar conectado à rede Wi-Fi do Access Point. O processo de calibração dos movimentos pode ser feito a partir de um dispositivo móvel com suporte para redes Wi-Fi. A Figura 16 mostra a rede Wi-Fi do AP com o nome *Wheelchair* entre todas as redes ao alcance do dispositivo.

Figura 16 - Rede Wi-Fi do Access Point

Fonte: elaborada pelo autor.

Após conectado à rede Wi-Fi, o usuário deve acessar a página raiz para ter acesso ao formulário de calibração dos ângulos. O endereço IP de acesso ao formulário é o endereço padrão do ESP8266 (192.168.4.1). A Figura 17 mostra uma página web do formulário de calibração, acessado a partir de um dispositivo móvel. O formulário de calibração dos ângulos mínimos para acionamento é composto por cinco campos, quatro campos para informar os ângulos referente ao movimento de acionamento para frente, para direita, para esquerda e para trás. Um dos campos do formulário é destinado para informar o limite máximo da distância com objetos atrás da cadeira de rodas. Os valores padrões carregados no formulário são os dados salvos na EEPROM, enviados no preenchimento do último formulário.

Figura 17 - Formulário de calibração dos movimentos



Fonte: elaborada pelo autor.

O protótipo que captura os dados do módulo MPU6050, responsável por gerar os dados dos movimentos é colocado sobre um boné. O usuário utiliza o boné e realiza os movimentos com a cabeça de acordo com o movimento que deseja que a cadeira de rodas realize. Os movimentos são capturados e efetuados pela cadeira de rodas. A Figura 18 mostra o boné disposto sobre a cabeça do usuário na utilização da solução.

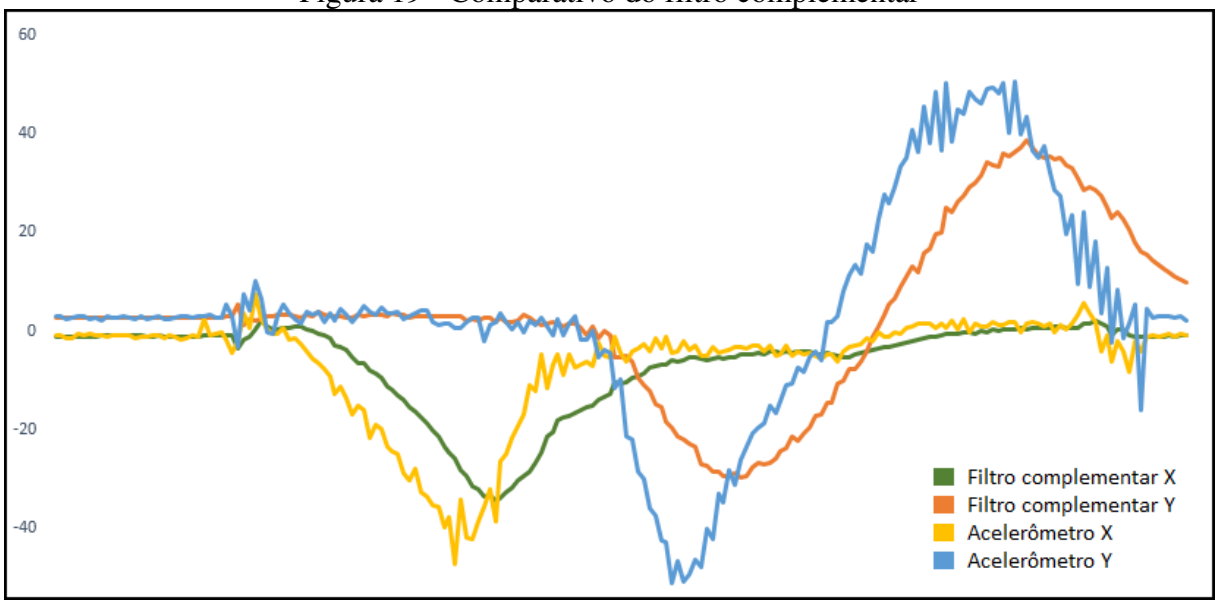
Figura 18 - Boné para utilizar a solução

Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Para detectar a inclinação da cabeça do usuário inicialmente seria utilizado apenas os dados do acelerômetro. Porém, após alguns testes realizados, percebeu-se uma grande quantidade de ruídos nos dados do acelerômetro capturados do módulo MPU6050. Para minimizar os ruídos, os dados do acelerômetro foram combinados com os dados do giroscópio, aplicando o filtro complementar. Na Figura 19 é apresentado um gráfico comparando o desempenho dos dados dos eixos x e y do acelerômetro com os dados do filtro complementar aplicado.

Figura 19 - Comparativo do filtro complementar



Fonte: elaborada pelo autor.

Uma pequena vibração no módulo MPU6050 faz com que o ângulo de inclinação calculado somente com os dados do acelerômetro resulte em uma inclinação falsa, porém, combinando com os dados do giroscópio, o resultado da inclinação ficou mais preciso. Para aplicar o filtro complementar, foi necessário implementá-lo em um loop de 40 milissegundos para garantir uma precisão consistente, já que o filtro leva em consideração o tempo entre as aplicações.

O driver motor H-Bridge DC *Mosfet* IRF3205 não desempenhou o rendimento necessário para o protótipo, por isso optou-se pelo driver motor IBT-2, que atendeu as necessidades de controle de velocidade dos motores através do PWM. O driver motor IBT-2 só controla um motor DC, sendo necessário dois driver motor, um para cada motor.

Inicialmente não seria utilizado o Arduino UNO para realizar o controle dos motores. Porém, após a necessidade da troca do driver motor, o número de I/Os do microcontrolador NodeMCU ESP8266 tornou-se insuficiente para interagir com os módulos do protótipo. O

Arduino UNO não tem suporte a redes Wireless, por isso, o microcontrolador NodeMCU ESP8266 foi mantido no protótipo de controle dos motores. O NodeMCU ESP8266 recebe os pacotes de dados através do protocolo UDP e os transmite para o Arduino UNO através da comunicação Serial.

Para realizar a comunicação por radiofrequência entre os microcontroladores ESP8266 foi utilizado uma rede Wi-Fi, pelo fato que as redes Wireless utilizam as ondas de rádio para serem propagadas através do meio físico. Inicialmente a troca de pacotes entre os microcontroladores seria feita através do protocolo TCP. Porém, o microcontrolador receptor estava acumulando uma alta quantidade de pacotes, causando um *delay* significativo nos movimentos. Para melhorar o desempenho na troca de pacotes, o protocolo foi alterado para o protocolo UDP, perdendo os benefícios da garantia de entrega, mas ganhando em velocidade de transmissão.

Para diminuir o tamanho do pacote no envio dos dados, o nome de identificação em cada coleção de pares do JSON teve que ser abreviado para um caractere. A velocidade na transmissão Serial também obteve ganho significativo quando alterado a taxa de transmissão de 2400 bps para 9600 bps. A velocidade final entre todo o processo de transmissão, desde o microcontrolador transmissor, passando pelo microcontrolador receptor no protocolo UDP até a transmissão Serial para o microcontrolador de controle dos motores, ficou praticamente imperceptível para o usuário.

A relação entre o diâmetro do pinhão no eixo do motor e o diâmetro da coroa da roda proporcionou uma redução na velocidade e aumento no torque, porém, com essa redução ainda não foi possível alcançar força suficiente para mover a cadeira de rodas com qualquer peso. Porém na continuidade desse projeto existe sugestões para a resolução desse problema. O movimento das rodas foi obtido levantando a cadeira de rodas com um suporte colocado em baixo, para que as rodas girassem livremente sem nenhum tipo de peso para movimentar.

Através do PWM foi possível controlar a velocidade dos motores. Porém, ao realizar testes no controle da velocidade dos motores visando movimentar a cadeira de rodas suavemente, com os motores interligados através da corrente nas rodas, aumentando a velocidade gradativamente, notou-se que a engrenagem dentro do motor de limpador de parabrisas acabou esfarelando nas pontas. O motivo do esfarelamento é pelo fato da engrenagem ser composta de material plástico, sendo necessário substituir a engrenagem por outra. Porém, não foi encontrado uma engrenagem de material mais resistente com os diâmetros para o motor adquirido, sendo necessário substituir por uma do modelo anterior.

O sensor ultrassônico HC-SR04 se mostrou preciso nos testes realizados. Os objetos que estão rentes ao ângulo do sensor foram identificados corretamente. O erro máximo identificado entre a distância do sensor ao objeto foi menor que um centímetro. Objetos que estão atrás da cadeira de rodas, porém não estão no ângulo do sensor não são identificados, para corrigir esse problema é necessário acoplar mais sensores na parte de trás da cadeira de rodas.

O Quadro 20 apresenta um comparativo entre as principais características dos trabalhos correlatos e do trabalho desenvolvido.

Quadro 20 - Comparativo entre trabalhos correlatos e o trabalho desenvolvido

Características	Trabalhos correlatos			Trabalho desenvolvido
	Fusco (2010)	Dellagostin (2011)	Ivo (2016)	
Acionamento	Movimentos da cabeça	Movimentos do corpo	Voz / joystick / web-app	Movimentos da cabeça
Comunicação sem fio	Não	Sim	Parcialmente	Sim
Adaptação mecânica da cadeira de rodas	Sim	Não	Sim	Sim
Detecta objetos atrás da cadeira de rodas	Não	Não	Não	Sim
Calibração dos movimentos	Sim	Sim	Não	Sim

Fonte: elaborado pelo autor.

Observa-se no Quadro 20 que o trabalho desenvolvido possui todas as funcionalidades dos trabalhos correlatos, se destacando por detectar objetos atrás da cadeira de rodas. O trabalho de Ivo (2016) se destaca pelas diversas formas de acionamento da cadeira, já no trabalho de Dellagostin (2011) o acionamento ocorre através dos movimentos do corpo, impedindo o uso da cadeira de rodas por um deficiente tetraplégico. A comunicação sem fio no trabalho de Ivo (2016) ocorre no acionamento por voz e pelo web-app.

4 CONCLUSÕES

O trabalho desenvolvido atendeu os objetivos estipulados inicialmente. A utilização do NodeMCU ESP8266 demonstrou um ótimo desempenho na coleta de dados do módulo MPU6050 e transmissão e recepção dos dados através da radiofrequência entre os módulos. Através do PWM foi possível controlar os motores da cadeira de rodas, girando as rodas no sentido horário e anti-horário. Pela falta de potência aplicada pelos motores, a cadeira de rodas não conseguiu se locomover no chão. As rodas foram movimentadas deixando a cadeira de rodas sobre um suporte, com as rodas flutuando. As rodas realizam os mesmos movimentos que serão realizados com cadeira de rodas no chão, possibilitando futuramente com a correção, movimentar a cadeira de rodas para frente e para trás ou rotacionar a cadeira para esquerda e para direita conforme os movimentos identificados da cabeça.

O sistema como um todo se encontra funcional e pode ser utilizado por qualquer usuário, até mesmo usuários que não são portadores de deficiências físicas ou possuem dificuldades na locomoção, pela liberdade do usuário poder calibrar os ângulos mínimos de acionamento dos motores. As características de alimentação embutida e comunicação sem fio com a cadeira de rodas permite inclusive aciona-la a distância.

Esse trabalho proporcionou um grande crescimento e aprendizagem em diversas áreas, como acelerometria, filtros complementares, sensores ultrassônicos, PWM, torque de motores, linguagem C++ de programação, além de superações com obstáculos que surgiram ao longo do desenvolvimento do trabalho

As ferramentas utilizadas para a especificação e desenvolvimento dos protótipos atenderam parcialmente todas as necessidades. A ferramenta Fritzing, utilizada para montagem do esquema elétrico não possui todos os módulos utilizados no protótipo a disposição. O custo total do protótipo ficou acessível em relação ao preço de uma cadeira de rodas elétrica, mesmo a cadeira de rodas elétrica não possuindo o acionamento pelos movimentos da cabeça. Sendo que a substituição de módulos, aumentando o desempenho e as funcionalidades da solução pode ser realizada com pequenas alterações na adaptação mecânica e no código fonte.

4.1 EXTENSÕES

No decorrer do trabalho foram observadas melhorias que podem ser implementadas:

a) corrigir possíveis problemas com ondulações no piso onde está a cadeira de rodas, como rampas e descidas que poderiam aumentar a velocidade ultrapassando os limites de segurança;

b) controlar a velocidade de movimentação da cadeira de rodas conforme o ângulo de inclinação da cabeça (quanto maior o ângulo, maior a velocidade);
c) comprimir os dados dos movimentos, diminuindo o tamanho do pacote de envio dos dados pela radiofrequência;
d) implementar uma camada de segurança no envio dos dados dos movimentos através da radiofrequência;
e) possibilitar ao usuário reposicionar o módulo MPU6050, deixando o usuário escolher onde ele quer colocar o protótipo de captura dos movimentos;
f) alterar o processo de calibração dos ângulos de acionamento, para que eles sejam capturados diretamente pelos movimentos e não por um formulário;
g) colocar uma caixa de redução que possibilite maior torque ao motor, para corrigir o problema da força de movimentação da cadeira de rodas;
h) utilizar uma bateria mais potente ou duas baterias automotivas, deixando os motores em velocidades equivalentes quando necessário, sem perdas de desempenho por carga de energia;
i) movimentar a cadeira de rodas na diagonal através do controle da velocidade das rodas;
j) adicionar um verificador de carga de bateria;
k) realizar o controle de temperatura dos drivers motores, evitando problemas com aquecimento.

REFERÊNCIAS

ACROBOTIC. **Getting Started With The ESP8266 ESP-12E Development Board.** Pasadena, 2015. Disponível em: <http://learn.acrobotic.com/tutorials/post/esp8266-getting-started>. Acesso em: 18 out. 2017.

ASSOCIAÇÃO SALVADOR. **Manual para Pessoas com Deficiência Motora.** 2. ed. Lisboa: Associação Salvador, 2017.

BÁSSORA, Luiz Antonio; GOMES, Malcoln Vellani. Sensoriamento Inercial de um Birrotor. In: SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA DA USP, 22., 2014, São Paulo. **Anais**... São Paulo: USP, 2014.

BERTOLETI, Pedro. **Controle e Monitoramento IoT com NodeMCU e MQTT.** Florianópolis, 2016. Disponível em: <https://www.filipeflop.com/blog/controle-monitoramento-iot-nodemcu-e-mqtt/>. Acesso em: 18 out. 2017.

BUENO, Aline Grotewold; ROMANO, Rodrigo Alvite. Filtro complementar aplicado a medida de inclinação de plataformas móveis. In: SEMINÁRIO MAUÁ DE INICIAÇÃO CIENTÍFICA, 3., 2011, São Paulo. **Anais…** São Paulo: Mauá, 2011.

DELLAGOSTIN, Jeremias Gelsomino. **Rede de acelerômetros para tecnologia assistiva.** 2011. 77 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre.

DUARTE, Cristiane Rose; COHEN, Regina. Afeto e Lugar: A Construção de uma Experiência Afetiva por Pessoas com Dificuldade de Locomoção. In: SEMINÁRIO INTERNACIONAL SOCIEDADE INCLUSIVA, 3., 2004, Belo Horizonte. **Anais...** Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2004. p. 1-8.

FUSCO, Daniel Alves. **Acionamento de uma cadeira de rodas através de um acelerômetro bi-axial inclinômetro.** 2010. 64 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre.

IVO, Regina Marcela. **Sistema de controle de cadeira de rodas motorizada para usuários portadores de tetraplegia.** 2016. 87 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica). Universidade de Brasília, Brasília.

MAZZOLENI, Alan. **Andar e se movimentar.** [S.l.], 2017. Disponível em: <https://vidamaislivre.com.br/colunas/andar-e-se-movimentar/>. Acesso em: 31 ago. 2017.

MINATEL, Pedro. **Utilizando a "EEPROM" do ESP8266.** Campinas, 2016. Disponível em: <http://pedrominatel.com.br/pt/esp8266/utilizando-eeprom-do-esp8266/>. Acesso em: 31 mar. 2018.

PAULA, Fabio Oliveira. **Sensores IMU**: Uma Abordagem Completa – Parte 1. [S.l.], 2015. Disponível em: <http://www.decom.ufop.br/imobilis/sensores-imu-uma-abordagem-completa-parte-1/>. Acesso em: 17 out. 2017.

SECRETARIA DE DIREITOS HUMANOS DA PRESIDÊNCIA DA REPÚBLICA. **Cartilha do Censo 2010:** Pessoas com Deficiência. Brasília: Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência, 2012.

SILVEIRA, Cristiano B. **O que é PWM e Para que Serve?.** [S.l.], 2016. Disponível em: <https://www.citisystems.com.br/pwm/>. Acesso em: 10 set. 2017.

SOUZA, Fábio. **Arduino:** Saídas PWM. [S.l.], 2014. Disponível em: <https://www.embarcados.com.br/arduino-saidas-pwm/>. Acesso em: 10 set. 2017.

THOMSEN, Adilson. **Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino.** Florianópolis, 2011. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Acesso em: 10 set. 2017.

THOMSEN, Adilson. **Como utilizar o sensor ultrassônico HC-SR04.** [S.l.], 2015. Disponível em: <http://buildbot.com.br/blog/como-utilizar-o-sensor-ultrasonico-hc-sr04/>. Acesso em: 17 out. 17.

APÊNDICE A - Casos de uso

O caso de uso UC01 - Movimenta a cadeira de rodas (Quadro 21) demonstra como o ator `Usuário` movimenta a cadeira de rodas com base nos movimentos da cabeça.

Quadro 21 - Detalhamento do UC01

Número	01
Caso de uso	Movimenta a cadeira de rodas
Ator	Usuário
Pré-condições	1. Os protótipos devem estar ligados 2. Os movimentos devem estar calibrados
Cenário principal	1. O Usuário realiza os movimentos com a cabeça 2. Os movimentos são capturados 3. Os movimentos são transmitidos para a cadeira de rodas 4. A cadeira de rodas realiza os movimentos

O caso de uso UC02 - Calibra os movimentos da cabeça (Quadro 22) demonstra como o ator `Usuário` calibra os movimentos da cabeça.

Quadro 22 - Detalhamento do UC02

Número	02
Caso de uso	Calibra os movimentos da cabeça
Ator	Usuário
Pré-condição	Usuário conectado à rede Wi-Fi
Cenário principal	1. O Usuário acessa a página de calibração 2. O Usuário informa os parâmetros de calibração 3. O Usuário clica no botão salvar

APÊNDICE B - Componentes utilizados e seus preços

A Tabela 1 contém todos os componentes utilizados no desenvolvimento dos protótipos e seus preços aproximados. Materiais como cabos, jumpers, cola quente entre outros foram desconsiderados. Os valores foram extraídos de lojas online de componentes eletrônicos FILIPEFLOP e Curto Circuito, da loja online de peças automotivas Mister Auto e da loja física Bike Limeira. A cadeira de rodas e a bateria automotiva foram adquiridas através de doações.

Tabela 1 - Custo dos componentes

Componente	**Quantidade**	**Preço unitário (R$)**	**Preço total (R$)**
NodeMCU ESP8266	2	32,75	65,50
Arduino UNO	1	49,90	49,90
Driver Ponte H 43A - IBT-2	2	84,00	168,00
MPU6050	1	12,30	12,30
HC-SR04	1	10,80	10,80
Conversor de Nível Lógico 3,3/5 V Bidirecional	1	3,10	3,10
Fonte Ajustável Protoboard 3,3V/5 V	1	7,00	7,00
Protoboard 400 Pontos	2	8,20	16,40
Coroa de bicicleta 40 dentes	2	8,00	16,00
Pinhão de bicicleta 20 dentes	2	6,00	12,00
Corrente de bicicleta 116 Elos	2	9,00	18,00
Cubo com suporte para freio a disco	2	40,00	80,00
Motor limpador de para brisas	2	210,00	420,00
		Total	879,00