Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Eduardo Paulo de Oliveira Pinto de Castro
Filipe Alexandre Gonçalves Fernandes
Daniel Pedro Vicente Gomes

Wheelchair HMU – *Head Motion Unit* – Controlo de uma cadeira de rodas por movimentos de cabeça

Projeto

Orientado por:

Prof. Doutor Gabriel Pereira Pires

Projeto
apresentado ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários
à obtenção do grau de Licenciado
em Engenharia Eletrotécnica e de Computadores

Dedicamos este trabalho às nossas famílias.

RESUMO

As pessoas que possuem limitações motoras têm na cadeira de rodas elétrica uma solução de grande benefício para a sua manobrabilidade. No entanto, devido a doenças crónicas ou agudas, algumas não podem utilizar o modo de controlo da cadeira mais comum, o *joystick*. Uma das alternativas de controlo existentes comercialmente é através da utilização de movimentos da cabeça do utilizador como controlador. Esta alternativa é bastante intuitiva e não necessita de muita destreza para uma condução normal. O projeto efetuado tem como objetivo o controlo do movimento de uma cadeira de rodas motorizada através dos movimentos de cabeça do utilizador. O sistema de controlo implementado, designado por *Wheelchair* HMU (*Head Motion Unit*), baseia-se na utilização de valores dos referenciais de dois sensores inerciais, colocados na cabeça do utilizador e na cadeira, calculando a diferença entre eles para enviar comandos ao controlador de potência, que controla os motores da cadeira. Para proporcionar segurança na sua utilização, foram adicionados diversos sistemas como deteção de obstáculos à frente da cadeira, deteção da presença da cabeça na zona segura e um botão de emergência, que bloqueia todo o funcionamento da cadeira. Foi acrescentado ainda um modo de controlo opcional com *joystick*. Este sistema foi desenvolvido com o intuito de proporcionar uma mobilidade intuitiva e segura a utilizadores de cadeiras de rodas, de forma a melhorar o seu bem-estar e qualidade de vida. Os objetivos propostos foram alcançados, tendo este sistema apresentado resultados positivos. Os resultados obtidos dos testes realizados indicam que este sistema é uma alternativa razoavelmente boa aos sistemas comerciais existentes.

Palavras-chave: Sensor inercial, Interface Humano-Máquina, Cadeira de rodas, Sistema de controlo, Arduino.

ABSTRACT

People who possess motion limitations have the electrical wheelchair as a greatly beneficial solution to their mobility. However, due to chronic or acute diseases, some of them can't use the most common control mode, the joystick. One control alternative which is commercially available is using the head movements of the user as a controller. This alternative is quite intuitive and doesn't require a lot of dexterity for regular driving. The objective of this project is the control of a motorized wheelchair through the user's head motions. The control system, named Wheelchair HMU (Head Motion Unit), is based on the usage of values from referentials of 2 inertial sensors, situated on the user's head and on the wheelchair, computing the differences of those values and translating them into commands for the power driver, which controls the wheelchair's motors. To deliver safety to the wheelchair's usage, some security systems were added such as obstacle detection near the wheelchair, presence of the user's head on the safety zone and an emergency switch to stop all the systems. An optional joystick control mode was also added. This system was developed with the intuitive and safe mobility of the user in mind, to improve their well-being and quality of life. The proposed objectives were accomplished, with the system presenting positive results. The results obtained through testing show that this system is a reasonably good alternative to the commercial systems available.

Keywords: Inertial sensor, Human-machine interface, Wheelchair, Control System, Arduino.

AGRADECIMENTOS

Em primeiro lugar, gostaríamos de agradecer ao Instituto Politécnico de Tomar (IPT) e ao laboratório VITA.ipt (Vida Assistida por Ambientes Inteligentes) pelas condições proporcionadas para a realização do projeto.

Este trabalho teve o enquadramento e apoio financeiro dos seguintes projetos de IC&DT:

- VITASENIOR-MT CENTRO-01-0145-FEDER-023659 com fundos do FEDER através dos programas operacionais CENTRO2020 e FCT;
- B-RELIABLE: SAICT/30935/2017 com fundos FEDER/FNR/OE através dos programas CENTRO2020 e FCT.

Por outro lado, agradecemos ao engenheiro Pedro Neves e aos colegas do VITA, com especial menção para o engenheiro Élio Lopes, pelo seu apoio técnico durante todas as fases deste projeto. Agradecemos também às nossas famílias e amigos pelo seu suporte.

Por último lugar, agradecemos ao nosso orientador, o Doutor Gabriel Pires pela sua disponibilidade, sugestões de melhoramento do projeto e encorajamento.

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas e siglas

ACK – *Acknowledge*

ADC – *Analog-Digital Converter*

AP – *Access Point*

DC – *Direct Current*

DCE – *Data Communication Equipment*

DTE – *Data Terminal Equipment*

GND – *Ground*

HMU – *Head motion Unit*

IMU – *Inertial Measurement Unit*

IP – *Internet Protocol*

I^2C – *Inter-Integrated Circuit*

LED – *Light Emitting Diode*

LiPo – *Lithium Polymer*

NACK – *Non-Acknowledge*

OSI – *Open Systems Interconnection*

PC – *Personal Computer*

PCB – Placa de Circuito Impresso

RC – *Radio Control*

RS232 – *Recommended Standard* 232

Rx – Receção

SCL – *Serial Clock*

SDA – *Serial Data*

TCP – *Transmission Control Protocol*

ToF – *Time of Flight*

Tx – Transmissão

UDP – *User Datagram Protocol*

UART – *Universal Asynchronous Receiver Transmitter*

USART – *Universal Synchronous Asynchronous Receiver Transmitter*

UPC – Unidade de processamento da cadeira

USB – *Universal Serial Bus*

1. Introdução

1.1. Contexto e motivação

Pessoas com limitações motoras estão sempre dependentes de terceiros no que diz respeito à sua mobilidade. Quando viável, a cadeira de rodas apresenta-se como uma das soluções mais usadas. Existem comercialmente vários modelos disponíveis e os modelos motorizados permitem, com menos esforço do utilizador, proporcionar-lhes alguma autonomia. A interface de comando mais comum neste tipo de modelo é o *joystick.* No entanto, para pessoas que não tenham controlo dos seus membros superiores esta interface não serve, pelo que tem de se recorrer a alternativas de controlo tais como *joystick* de queixo, sistemas de varrimento, comutadores de cabeça, ou tubo de sopro/sucção. No entanto, estas interfaces são em muitos casos difíceis de utilizar e pouco flexíveis.

Para pessoas que sofrem de doenças agudas ou crónicas tais como a Lesão Medular, a Esclerose lateral amiotrófica, Insuficiência vertebro-basilar, Hemorragias cerebrais ou ainda outras doenças do sistema nervoso como a Doença de *Huntington*, Doença de *Parkinson*, Atrofia muscular espinal progressiva ou Paralisia Cerebral, uma alternativa de controlo mais "orgânica" é a utilização dos movimentos de cabeça do utilizador para conduzir a cadeira. Os produtos comerciais existentes com comando de cadeiras através de movimentos de cabeça são baseados em sensores de contacto ou proximidade, precisando que a cabeça do utilizador esteja apoiada ou próxima de apoios de cabeça localizados atrás e de forma lateral. Existem no mercado alguns sistemas comerciais com características semelhantes, tais como os desenvolvidos pelos fabricantes *Permobil* [1] e *Adaptive Switch Laboratories* [2].

O sistema desenvolvido neste projeto, designado por HMU (*Head Motion Unit*), possui todas as funcionalidades destes sistemas comerciais, e ainda um conjunto de características adicionais, nomeadamente:

1) A possibilidade de usar ou não os apoios de cabeça (dependendo das funcionalidades do utilizador).
2) a possibilidade de comando de direção e velocidade de forma contínua;
3) a possibilidade do sistema ser calibrado de forma a que a cabeça esteja em diversas posições (ajustando-se por exemplo a situações de espasticidade em que o utilizador tem movimentos confinados a determinada posição);

Com estas características, o HMU inclui e complementa as funcionalidades dos sistemas existentes, podendo ser usado por um grupo mais abrangente de utilizadores com limitações motoras.

1.2. Objetivos

Este projeto teve como objetivo principal a robotização de uma cadeira de rodas motorizada, existente no laboratório VITA.IPT, e o desenvolvimento de uma interface homem-máquina para comando da cadeira (HMU), baseada em movimentos de cabeça do utilizador, reconhecidos através da informação devolvida por sensores inerciais (*Inertial Measurement Unit* -IMU). Adicionalmente, foi proposto que a cadeira de rodas também pudesse também ser controlada por meio de um *joystick*, possuindo assim dois modos de comando, selecionados através de um interruptor.

A HMU tem como base a utilização dos referenciais de dois sensores IMU, um deles colocado na cabeça do utilizador e o outro colocado na cadeira. Para a implementação da HMU foram desenvolvidos dois módulos: módulo da cabeça, designado por *Headset* e módulo da cadeira, designado por UPC (Unidade de Processamento da Cadeira), que serve de unidade central de controlo a todos os outros sistemas implementados.

Um dos principais requisitos do sistema foi o da segurança, pelo que foram implementados vários níveis de segurança, nomeadamente, deteção de proximidade de obstáculos em redor da cadeira, deteção da posição da cabeça dentro de área de segurança e botão de emergência para parar todo o funcionamento da cadeira. Foi construída uma placa com luzes indicadoras e botões que fornecesse a possibilidade de configuração do sistema e acesso a informação relevante durante a condução da cadeira.

Por último foi melhorada a ergonomia da cadeira, de modo a melhorar o bem-estar do utilizador bem como a facilitar-lhe o uso das interfaces. Foi também melhorada a caixa das baterias, de modo a facilitar o seu recarregamento.

1.3. Trabalho realizado

Na Figura 1 apresenta-se o diagrama de blocos do sistema implementado, dividido por módulos.

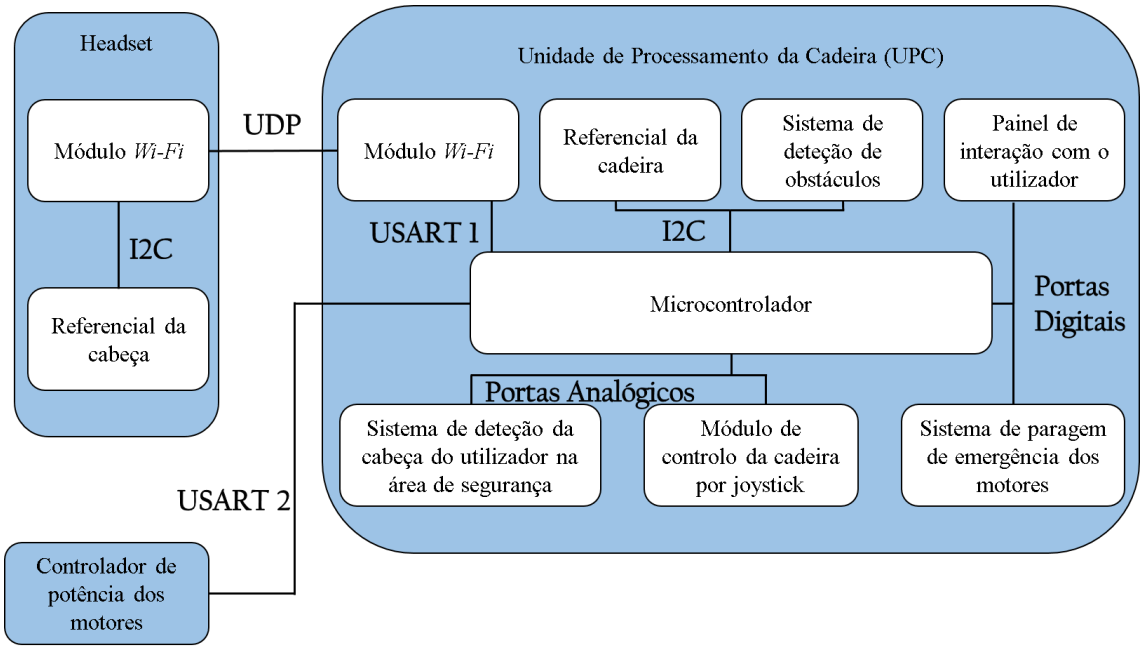


Figura 1*: Diagrama de todos os principais módulos implementados*

A HMU consiste na combinação do *Headset* com parte dos componentes da UPC (Microcontrolador, sensor inercial e Módulo *Wi-Fi*). Como referido anteriormente, o funcionamento da HMU baseia-se na utilização dos referenciais dos sensores inerciais para movimentar a cadeira. Inicialmente, o utilizador coloca a cabeça na posição que mais lhe conforta e define-a como sendo a posição neutra ao calibrar os referenciais (atribuição de offsets). Quando o utilizador move a cabeça, o referencial do sensor da cabeça muda de posição. A UPC está continuamente a receber a informação das posições dos 2 referenciais e quando existe diferença nos eixos YY ou ZZ, envia comandos ao controlador de potência Roboteq HDC2450. O controlador interpreta esses comandos e atua os motores, fazendo com que a cadeira se mova na direção pretendida. A velocidade da cadeira é proporcional ao valor das diferenças dos eixos dos referenciais, o que significa que quanto maior for a inclinação da cabeça do utilizador, maior a velocidade. Isto traduz-se em movimentos mais precisos e em melhor manobrabilidade em espaços mais apertados.

A primeira tarefa do projeto consistiu no teste do controlo motriz com recurso ao software Roborun+ da Roboteq. A comunicação entre o controlador e o PC (*Personal Computer*) foi realizada através de uma comunicação série RS-232. Para validar o controlo dos motores foram enviados vários comandos do controlador e monitorizada a resposta dos motores em termos de velocidade e aceleração.

Seguiu-se a robotização da cadeira de rodas. Para tal, foram desenvolvidos os seguintes módulos:

- Módulo de controlo da cadeira por movimentos de cabeça (HMU);
- Módulo de controlo por *joystick*;
- Sistemas de segurança: deteção de obstáculos, deteção da posição da cabeça na área de segurança, botão de emergência, e deteção de falhas nos módulos *Wi-Fi*;
- Painel informativo e de comando que inclui o botão de transição de modos de controlo (*joystick vs.* HMU), o botão de definição do referencial de posição inicial da HMU (calibração) e um conjunto de LEDs (*Light Emitting Diode*) informativos dos respetivos estados;
- Peças 3D com a finalidade de melhorar a ergonomia e usabilidade da cadeira, nomeadamente, suportes para o módulo da cabeça, para adaptação dos sensores dos sistemas de segurança, acoplamento do *joystick* e da placa informativa;

- Cablagem de comunicação embutida na estrutura da cadeira e remodelação da tampa da caixa das baterias e integração de uma caixa de derivação com fusíveis;

1.4. Estrutura do documento

Este documento encontra-se estruturado da seguinte forma:

- **Introdução**

Neste capítulo é efetuada uma breve descrição do projeto mencionando, o âmbito e os seus objetivos.

- **Estado da arte**

É apresentado o estado da arte de sistemas com os mesmos princípios deste projeto.

- **Tecnologias**

Este capítulo apresenta as principais tecnologias usadas e a fundamentação da sua utilização.

- **Desenvolvimento**

Apresentam-se e descrevem-se os componentes aplicados à cadeira de rodas para realizar os objetivos propostos bem como o diagrama de ligação e esquema de montagem de todo o sistema. De seguida são descritas em pormenor as várias características do sistema. Este capítulo divide-se em:

  - Modo de controlo com movimentos de cabeça (HMU);
  - Modo de controlo com *joystick;*
  - Controlador de potência;
  - Sistemas de segurança;
  - Alimentações elétricas;
  - Componentes auxiliares de suporte.

- **Ensaios e resultados experimentais**

Neste capítulo são apresentados e descritos os testes realizados para a consecução dos objetivos propostos.

- **Conclusão**

São apresentadas as conclusões retiradas da elaboração do projeto bem como sugestões para trabalhos futuros.

2. Estado da arte

Neste capítulo são descritos vários casos de interfaces de comando por movimento de cabeça aplicados a vários sistemas, tais como cadeiras de rodas, manipuladores robóticos, veículos motorizados, controlo de cursor no computador, analisando componentes, tecnologia utilizada, modo de utilização e sucesso de testes efetuados.

2.1. Interfaces para comando de cadeiras de rodas elétricas

Em Pajkanović *et al.* [3], os autores apresentam um sistema que ativa um atuador mecânico através de movimentos de cabeça (exclusivamente inclinações para a frente, trás, esquerda e direita, ações em 2 eixos). Esse atuador controla um *joystick* que move a cadeira de rodas na direção indicada pelo utilizador. Um acelerómetro é usado para recolher dados dos movimentos de cabeça. Os dados são enviados a um microcontrolador que implementa um algoritmo ativando o atuador mecânico para mover o *joystick* de acordo com a ordem do utilizador. Este algoritmo define inicialmente uma zona de repouso que é ajustável a cada utilizador. Um comando do utilizador é aceite quando essa zona é ultrapassada duas vezes por um tipo de movimento pré-estabelecido, durante um tempo definido. Um exemplo do comando "frente" é ilustrado na Figura 2, em que o movimento (3) consiste na repetição do movimento (1). O algoritmo funciona com estados e a transição entre cada um deles ocorre quando o utilizador dá um novo comando.

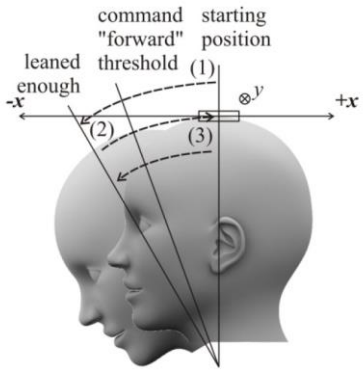


Figura 2: Exemplo do comando "frente" [3]

Para testar o protótipo, foram executados 2 testes com 3 participantes. Com o primeiro, testou-se a execução dos comandos intencionais de movimento e no seguinte testaram-se movimentos não intencionais (como por exemplo olhar para a direita ou ler um texto em frente do participante, requerendo a inclinação da cabeça para baixo). Os resultados foram positivos, mas os autores referem que os erros que surgiram de ambos os testes podiam ser reduzidos com a adição de um giroscópio para uma recolha mais precisa dos dados do utilizador. Os autores mencionam ainda que um dos movimentos não intencionais (frente-trás-frente) deveria ser realizado mais lentamente para não enviar o comando de movimento à cadeira. Outra observação efetuada foi o facto de a cadeira não possuir sistema de deteção de obstáculos que, junto ao facto dos testes terem sido efetuados num ambiente interior bastante estreito, dizem poder ser a causa da maioria dos erros obtidos.

Chen *et al*. [4] apresentou um aparelho de controlo da cadeira de rodas por movimentos de cabeça utilizando 2 sensores de inclinação como módulo de controlo de *input*. Um dos sensores deteta a inclinação anterior/posterior da cabeça e o outro a inclinação para a esquerda/direita. Inclinações de cabeça nas várias direções correspondem a uma combinação de valores analógicos de tensão dos 2 sensores. A informação recolhida deste módulo é enviada para um conversor ADC (*Analog-Digital Converter*) de 10 *bit* e a informação digital é enviada para o microcontrolador que cria 9 subdivisões dispostas numa matriz 3x3, dependendo da combinação dos valores enviados pelos sensores e que correspondem às 8 direções possíveis (movimento para a frente/trás, rotação para a esquerda/direita e movimentos diagonais) e à paragem, como se pode

observar na Figura 3. O microcontrolador envia o comando correspondente aos motores da cadeira para que esta siga na direção pretendida.

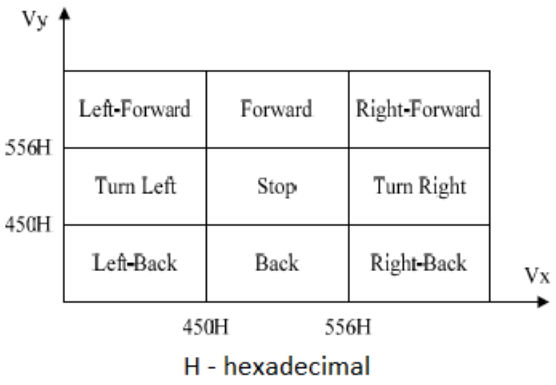


Figura 3: Região de controlo. Adaptado de [4]

Este sistema foi testado com um grupo de 6 pacientes com lesões na medula espinal de nível C2-C4 que ficaram quadriplégicos, mas que conseguem rodar o pescoço. Foi usado um grupo de controlo de 6 pacientes com lesões da medula espinal de nível C5-C6, habituados a usar *joystick* de queixo durante pelo menos 1 ano. Ambos os grupos tiveram 30 minutos de treino antes dos testes. Estes consistiram na execução de 3 tarefas, como mostra a Figura 4, sendo medido e analisado estatisticamente o tempo de execução das tarefas por cada grupo. Os resultados demonstram que não existe desvio significativo na média dos tempos necessários para realizar cada tarefa, o que significa que o sistema é *user-friendly* para pessoas com este tipo de lesões da medula espinal.

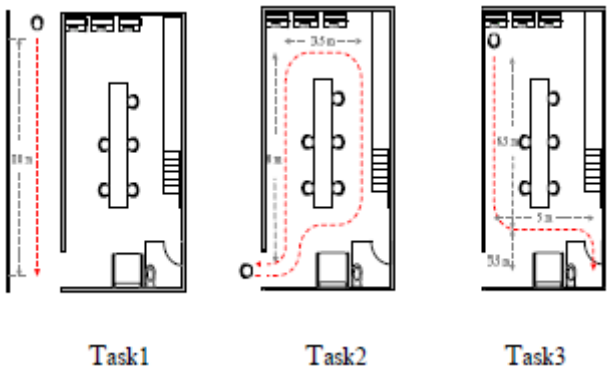


Figura 4: Testes clínicos [4]

O sistema de controlo apresentado por Kupetz *et al.* [5] mede o movimento de cabeça usando uma câmara, montada no topo do encosto da cabeça da cadeira, com um filtro de infravermelhos, que realiza seguimento de movimento da posição e ângulo de um quadrado 2x2 de LEDs (Figura 5) que é colocado na parte de trás de um chapéu colocado no utilizador. Do processamento da câmara, são enviados comandos aos motores da cadeira proporcionais ao movimento detetado. Este sistema possui ainda um

modo *standby*, onde os movimentos de cabeça não se traduzem em movimento da cadeira, que é ativado quando o utilizador pressiona a cabeça contra o assento até acionar um botão. Para sair deste modo basta voltar a pressionar o botão. Os autores mencionam que os testes realizados foram bem-sucedidos, mas sem especificar em que modos foram realizados, número de tentativas ou percentagem de sucesso.

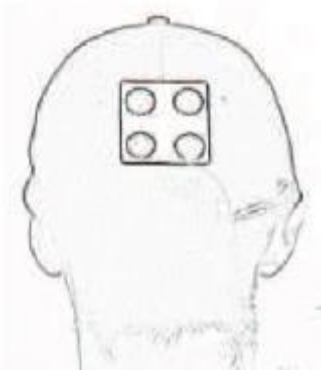


Figura 5: Quadrado de LEDs 2x2 [5]

Rechy-Ramirez *et al.* [6] apresentou uma interface que recorre ao giroscópio de um sensor eletroencefalográfico disponível comercialmente, o *Emotiv* EPOC, para capturar os movimentos de cabeça do utilizador e através do seu *software*, com recurso a um computador, enviar comandos aos motores da cadeira. Esta interface possui 2 modos de funcionamento. O primeiro modo utiliza 4 movimentos específicos de cabeça (inclinação para a frente/trás e rotação para a esquerda/direita) e apenas possui 4 comandos possíveis: movimento para a frente, rotação para a esquerda, rotação opara a direita e paragem. Após realizar o movimento pretendido (ultrapassando o limite estabelecido da zona de repouso), existe um *buffer* de 1 segundo para o utilizador colocar a cabeça na posição inicial, de modo a evitar um possível comando na direção oposta que não é pretendido pelo utilizador. Neste modo, através da interface gráfica é possível, mesmo em plena operação, ajustar os limites da zona de repouso bem como a velocidade máxima da cadeira. Este modo não é muito intuitivo pois o movimento correspondente ao comando de andar em frente é a inclinação da cabeça para trás, sendo o comando de parar executado pela inclinação da cabeça para a frente. Outra situação é o facto de a transição de uma rotação para a paragem ser efetuada com o movimento de inclinação para a frente ao contrário da rotação da cabeça para o lado contrário, que seria mais intuitivo. O segundo modo utiliza apenas 2 movimentos de cabeça (inclinação para a frente/trás) para efetuar os mesmos 4 comandos. Estes estão numa sequência e o utilizador

efetua o movimento para percorrer a sequência até selecionar o comando que pretende. Os comandos são mostrados na interface gráfica durante 1 segundo antes de serem executados. Se o utilizador não efetuar um movimento de cabeça para selecionar outro comando, após 1 segundo, o comando selecionado é executado. Foram realizados testes com 2 participantes sem problemas motores que consistiram em efetuar um percurso num espaço fechado (Figura 6) sem bater nos obstáculos utilizando os vários modos de controlo (1º modo - 4 movimentos, 2º modo- 1 movimento e ainda modo *joystick*), com ajuste da velocidade a 3 níveis e analisando o tempo decorrido e trajetórias. Ambos participantes executaram com sucesso todos os testes e reportaram que preferiram o modo 1 com o nível de velocidade maior e o modo 2 com o nível de velocidade menor. Os autores concluíram que o modo 1 é confiável e que o uso do modo 2 deve ser usado para uma velocidade baixa.

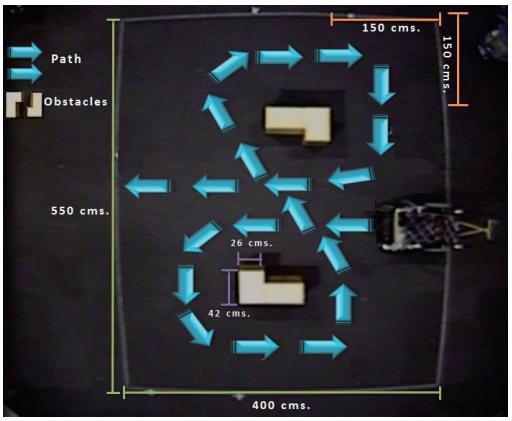


Figura 6: Percurso de teste [6].

Kondori *et al.* [7] apresentou um sistema de controlo em que os movimentos de cabeça são estimados com base numa sequência de imagens capturadas por uma *Kinect*, colocada à frente do utilizador, com recurso a uma equação de restrição de fluxo ótico para estimar diretamente os movimentos. Os movimentos de cabeça válidos são a inclinação para a frente/trás e a rotação para a esquerda/direita. Os possíveis comandos para a cadeira são: movimento frontal, paragem, rotação à esquerda e rotação à direita. Estes comandos são executados quando são ultrapassados os limites da zona de repouso, estabelecida inicialmente. Os testes efetuados ao algoritmo concluíram que este funciona com elevada precisão e em tempo real e foram também efetuados testes em simulação do seguimento de um caminho usando os dados recolhidos pelo sistema dos movimentos de um utilizador (Figura 7), mostrando resultados bastante positivos. Os autores concluem que será difícil a utilização deste sistema num ambiente exterior devido às limitações da tecnologia usada.

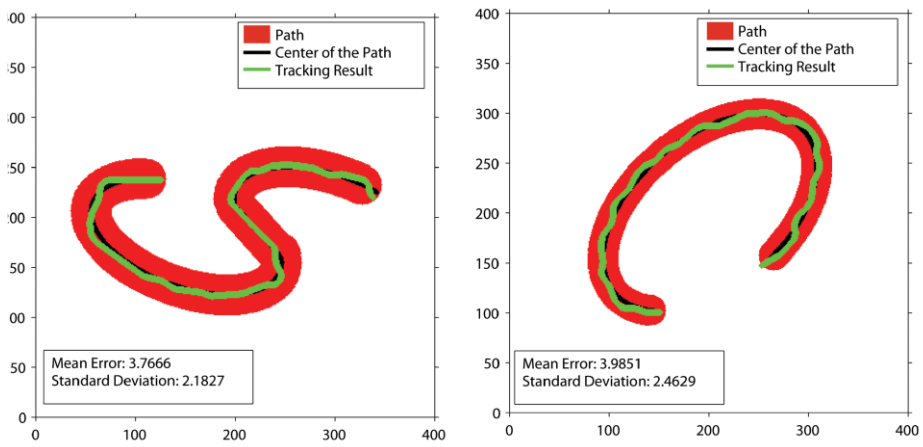


Figura 7: Testes em simulação do seguimento de um caminho [7].

O sistema de controlo apresentado por Qamar *et al.* [8] foi configurado para aceitar inclinações de cabeça pré-designadas do utilizador para controlo da velocidade e direção de um veículo motorizado. Com um módulo composto por acelerómetro, microcontrolador Arduino *Nano* e comunicação de dados *bluetooth*, colocado na cabeça do utilizador, os movimentos de cabeça em 2 eixos (inclinações da cabeça para a frente/trás e esquerda/direita) são traduzidos em primeiro lugar em ângulos nos 2 eixos pelo acelerómetro e de seguida em valores de tensão pelo Arduino. Os dados recolhidos são transmitidos via *Bluetooth* para o módulo instalado no veículo que consiste no recetor *bluetooth* e num controlador que utiliza lógica difusa *para* diferenciar entre movimentos de cabeça naturais e movimentos intencionais. O controlador converte os valores de tensão recebidos em comandos de velocidade e direção, que são enviados para os sistemas respetivos (regulador de velocidade de um motor DC (*Direct Current*) no caso do sistema de velocidade e motor DC com potenciómetro de *feedback* como sistema de direção). O esquema deste sistema está ilustrado na Figura 8.

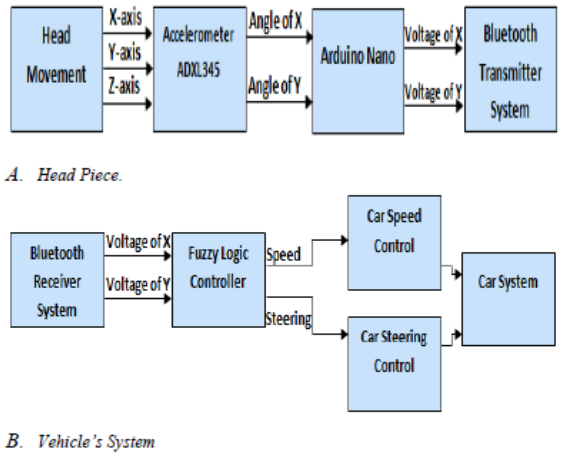


Figura 8: Esquema do sistema de controlo [8].

Os testes realizados foram um sucesso, estando o sistema funcional. Os autores mencionam que o controlador de lógica difusa é aplicável a outras aplicações como por exemplo cadeiras de rodas.

Prasad *et al.* [9] apresentou um protótipo em pequena escala de um sistema de controlo com movimentos de cabeça para ser utilizado futuramente numa cadeira de rodas. Este sistema é composto por um módulo da cabeça, que contém um acelerómetro, um microcontrolador e um módulo de comunicação baseado num transmissor *XBee*, e pelo módulo do veículo, que consiste num recetor *XBee*, microcontrolador e controlador dos motores. Os movimentos de cabeça são detetados pelo acelerómetro que envia a informação para o microcontrolador e este envia a informação processada para o transmissor. O recetor envia o sinal em série para o microcontrolador da cadeira, que processa o sinal e envia os dados ao circuito do controlador dos motores. Esse circuito envia comandos de entrada aos servomotores que realizam o movimento da cadeira. Este sistema possui ainda a possibilidade de controlo da cadeira de forma remota. Utilizando um módulo *Wi-Fi* e através de um telemóvel com uma aplicação *android*/*iOS*, o utilizador envia comandos que são transmitidos ao Arduino e posteriormente aos motores da cadeira através do módulo *Wi-Fi*. Os autores apresentam um sucesso geral de 87,5% em testes da inclinação do módulo da cabeça nas 4 direções possíveis. No entanto não apresentam qualquer detalhe sobre o que constitui na realidade um teste bem-sucedido ou o que representa um resultado negativo do teste.

Yoda *et al.* [10] apresentou um sistema de controlo baseado em reconhecimento de movimentos de cabeça como uma composição temporal de orientações faciais. Recorrendo a uma câmara estereoscópica e um sensor magnético colocado na cabeça do utilizador, conseguiram criar um *menu* (Figura 9) que depende apenas dos movimentos de cabeça para selecionar entre vários modos como virar no lugar, andar para trás, andar para a frente e mudança de velocidade.

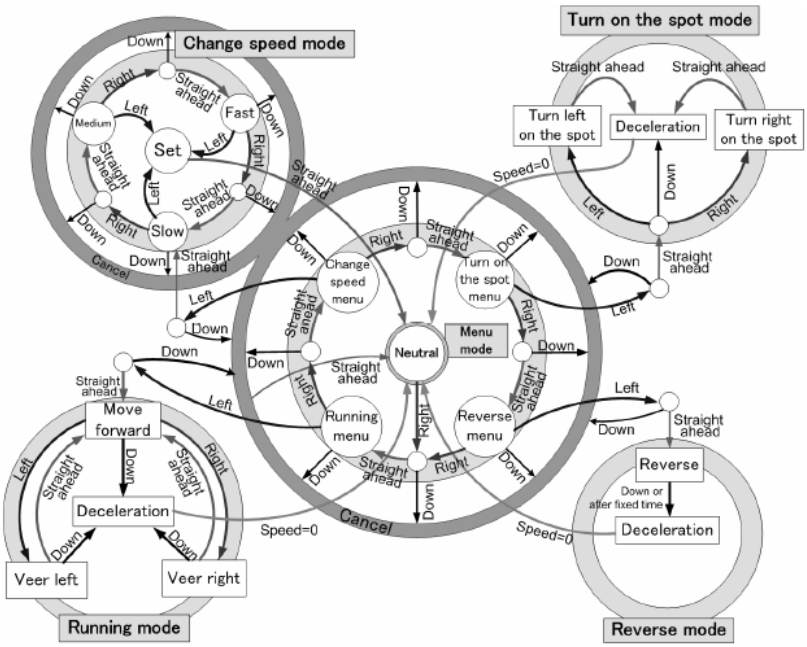


Figura 9: Menu de controlo da cadeira [10].

Foram realizados testes clínicos com 1 participante ao qual este sistema é dirigido. Numa primeira fase foram recolhidos dados de treino para o sistema com o sensor magnético. Numa fase posterior, foi analisado o reconhecimento de gestos por parte da câmara, cujas decisões de controlo foram baseadas nos dados de treino, sendo esta efetuada durante 45 minutos, em ambiente exterior e com várias condições como recebendo luz direta do sol e à sombra total e parcial. Os resultados demonstram uma elevada fiabilidade deste sistema. No entanto, neste caso, bem como em [6], este sistema não é intuitivo e o utilizador necessita de aprender a circular pelos diversos modos/comandos.

Em [11], é apresentado um sistema de controlo do cursor de um rato através de movimentos de cabeça. São utilizados o giroscópio e o acelerómetro de um sensor inercial LSM9DS0 ligado a um microcontrolador A-*Star* 32U4 Micro (clone do Arduino Leonardo) por I^2C (*Inter-Integrated Circuit*), necessitando de um conversor de níveis de tensão. Este módulo é colocado na cabeça do utilizador (Figura 10) e é ligado ao PC via USB (*Universal Serial Bus*). As leituras do acelerómetro e giroscópio dão valores de orientação que se traduzem em movimentos do cursor em 2 eixos, X e Y. Neste sistema surgiram alguns problemas tais como a posição inicial do acelerómetro não permanecer zero quando este está parado e o facto de o LSM9DS0 não iniciar tão rápido quanto o processador, trazendo com isto necessidade de descartar leituras no código, o que torna o sistema menos preciso.

Figura 10: Controlador de um rato por movimentos de cabeça

O sistema ASL105 da série S100 disponibilizado pela *Adaptive Switch Laboratories* [2] possui 3 sensores de proximidade não ajustáveis colocados dentro do encosto da cabeça (Figura 11), colocados um em cada ponta das laterais do encosto e um no centro. Os movimentos da cadeira são efetuados quando o utilizador aproxima a cabeça do sensor da direção para a qual se quer deslocar e para seguir em frente (ou ter ambas as opções de ir para a frente/trás, dependendo da programação) aproxima-se do sensor do meio. Para fazer as diagonais basta aproximar a cabeça do sensor do meio e do sensor do lado para o qual se quer deslocar. A rigidez da localização dos sensores pode dificultar o uso da cadeira para algumas pessoas.

Figura 11: ASL 105 S100 [2].

O sistema comercial *Total Control Head Array System* disponibilizado pela Permobil [1] apresenta uma unidade de matriz de cabeça com vários sensores de proximidade (Figura 12) que podem ser dispostos de várias maneiras à volta do utilizador, com recurso a braços extensíveis e rodáveis bem como um eixo esférico nas suas pontas, onde encaixam os sensores. No encosto encontram-se dois sensores occipitais para uma maior área de ativação e para simplificar as curvas na diagonal. Este sistema, embora possibilite o controlo da cadeira por movimentos de cabeça, não oferece tanta liberdade nesses movimentos, sendo que para mudar a posição da cabeça, têm de se ajustar todos os sensores. Devido a que se trata de um produto comercializado, não foi possível obter informação com detalhe sobre a tecnologia usada por este sistema.

Figura 12: Total Control Head Array System da Permobil [1].

De seguida, apresenta-se a Tabela 1, que contém os sistemas descritos anteriormente e algumas características.

Tabela 1: Resumo de sistemas de controlo por movimentos de cabeça

Autor	Tipo de sistema	Aplicação	Sensores	Movimentos de cabeça detetados	Movimentos efetuados	Local dos testes/	Nº de participantes
Pajkanović *et al.* [3]	Protótipo	Cadeira de rodas	Acelerómetro	4 (frente/trás, esquerda/direita)	4 (frente/trás, esquerda/direita)	Interior	3
Chen *et al.* [4]	Protótipo	Cadeira de rodas	Sensores de inclinação	8 (frente/trás/esquerda/direita/diagonais)	8 (frente/trás/esquerda/direita/diagonais)	Interior	12
Kupetz *et al.* [5]	Protótipo	Cadeira de rodas	Câmara de infravermelhos	Qualquer	Qualquer	-	-
Rechy-Ramirez *et al.* [6]	Protótipo	Cadeira de rodas	Giroscópio	4 (frente/trás, esquerda/direita)	3 (frente, esquerda/direita)	Interior	2
Kondori *et al.* [7]	Protótipo	Cadeira de rodas	Kinect	4 (frente/trás, esquerda/direita)	3 (frente, esquerda/direita)	Simulação	-
Qamar *et al.* [8]	Protótipo	Protótipo de veículo motorizado	Acelerómetro	4 (frente/trás, esquerda/direita)	-	-	-
Prasad *et al.* [9]	Protótipo	Cadeira de rodas	Acelerómetro	4 (frente/trás, esquerda/direita)	4 (frente/trás, esquerda/direita)	-	-
Yoda *et al.* [10]	Protótipo	Cadeira de rodas	Câmara estereoscópica, Sensor Magnético	4 (frente/trás, esquerda/direita)	4 (frente/trás, esquerda/direita)	Exterior	1
Millmore [11]	Protótipo	Rato	Sensor inercial	Todos	4 (cima/baixo, esquerda/direita)	-	-
ASL [2]	Comercial	Cadeira de rodas	Sensores de proximidade	-	Qualquer	-	-
Permobil [1]	Comercial	Cadeira de rodas	Sensores de proximidade	-	Qualquer	-	-

- Não refere/Não aplicável

3. Tecnologias

Este capítulo apresenta os principais conceitos, tecnologias e métodos utilizados neste projeto.

3.1. Dispositivos sensoriais, comunicação e processamento

Sensor BNO055

O sensor BNO055 (Figura 13) foi o sensor escolhido para a captação dos valores inerciais da cabeça do utilizador pois é um sensor relativamente barato, pequeno e que providencia a orientação absoluta em ângulos Euler [12] [13].

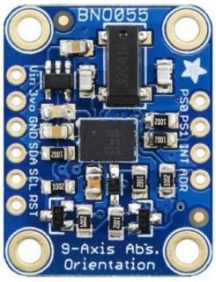


Figura 13:Sensor IMU BNO055

O BNO055 é um sensor inteligente de orientação absoluta obtida através da fusão de nove eixos. Este sensor é constituído por um microcontrolador que realiza a fusão dos dados obtidos de outros três sensores, todos eles de eixo triplo: um Giroscópio, que mede a aceleração rotacional; um Acelerómetro, que mede a aceleração tangencial; e um Magnetómetro, que mede a força do campo magnético local. O BNO055 pode ser alimentado por uma tensão entre os 3.3V e os 5V e possui três modos de gestão de energia: modo normal; modo de baixo consumo; e modo de suspensão. O sensor possui ainda um porto de comunicação I^2C bidirecional, para além de treze modos de operação distintos: um modo de configuração; sete modos de obtenção de dados independentes, isto é, sem

fusão; e cinco modos de fusão, dos quais dois são de orientação relativa e três de orientação absoluta (Figura 14).

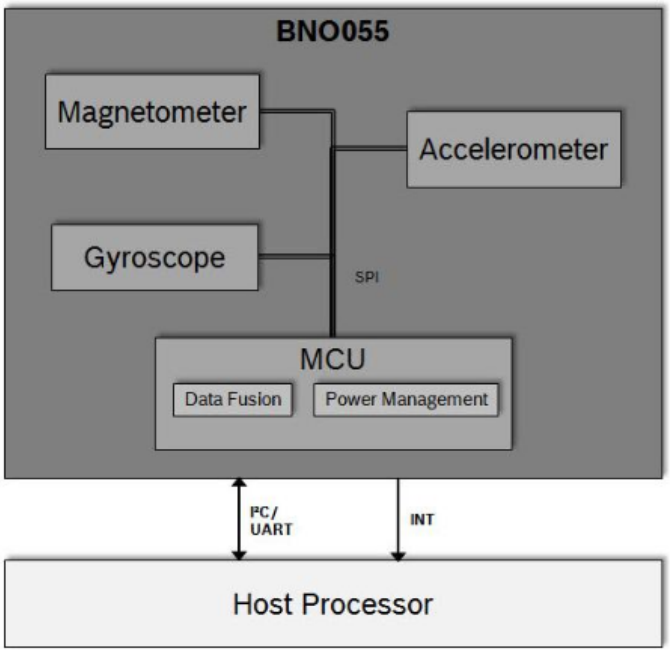


Figura 14: Arquitetura do sistema BNO055

Sensor VL53L0X

O sensor VL53L0X ilustrado na Figura 15 foi escolhido para o sistema de Deteção de Obstáculos devido ao seu tamanho e baixo custo, mas, principalmente por oferecer medições bastante rápidas e precisas [14] [15].

Figura 15:Sensor de distância VL53L0X

O VL53L0X é um sensor de distância de ToF (*Time of Flight*), isto é, um sensor de distância baseado em tempo de voo, composto por um emissor e um recetor de laser. Este sensor, devido ao laser de precisão, é capaz de fazer medições bastantes precisas, de qualquer objeto diretamente à sua frente, numa gama de distâncias entre os cinquenta e os mil e duzentos milímetros (Figura 16). O cálculo da distância, como é indicado pelo seu tipo, é realizado através do tempo que o laser demora a ir e a voltar depois de ter embatido na superfície à sua frente.

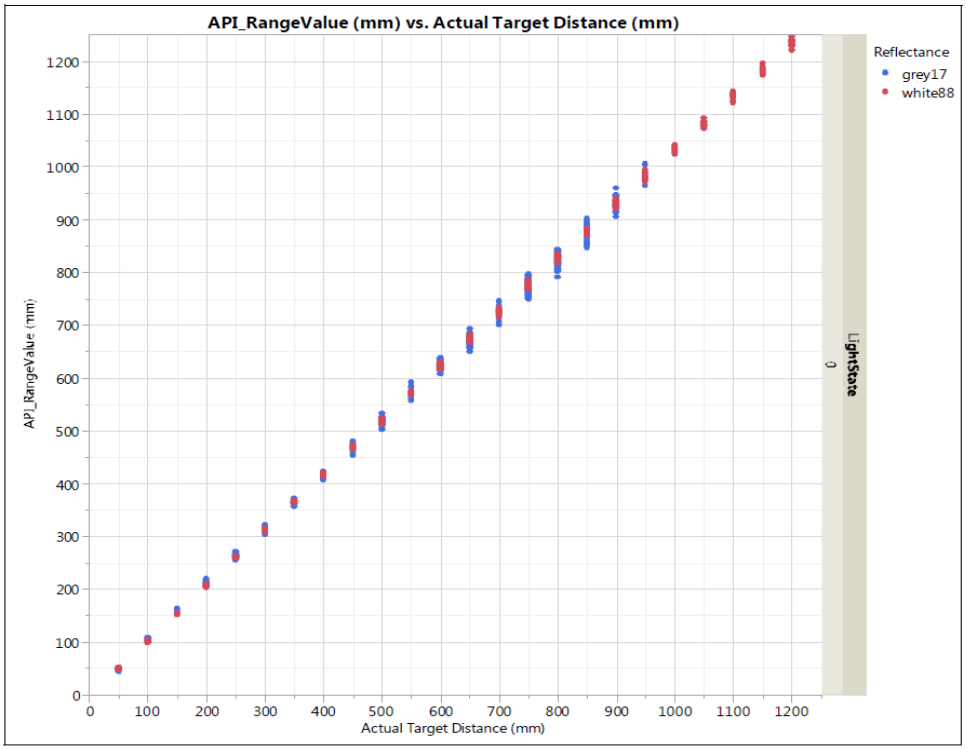


Figura 16: Alcance e precisão do VL53L0X por defeito (modo normal)

Sensor Sharp 2D120X

O sensor 2D120X (Figura 17) foi escolhido para o sistema de Área de Segurança da cabeça devido ao seu baixo custo e à precisão das suas medições para distâncias curtas [16].

Figura 17: Sensor de distância Sharp 2D120X

O *Sharp* 2D120X é um sensor de distância de infravermelho, sendo constituído por um emissor e um recetor. Este sensor é capaz de fazer medições numa gama de distâncias entre os quarenta e os trezentos milímetros, sendo essa distância dada em função da tensão de saída do sensor (Figura 18).

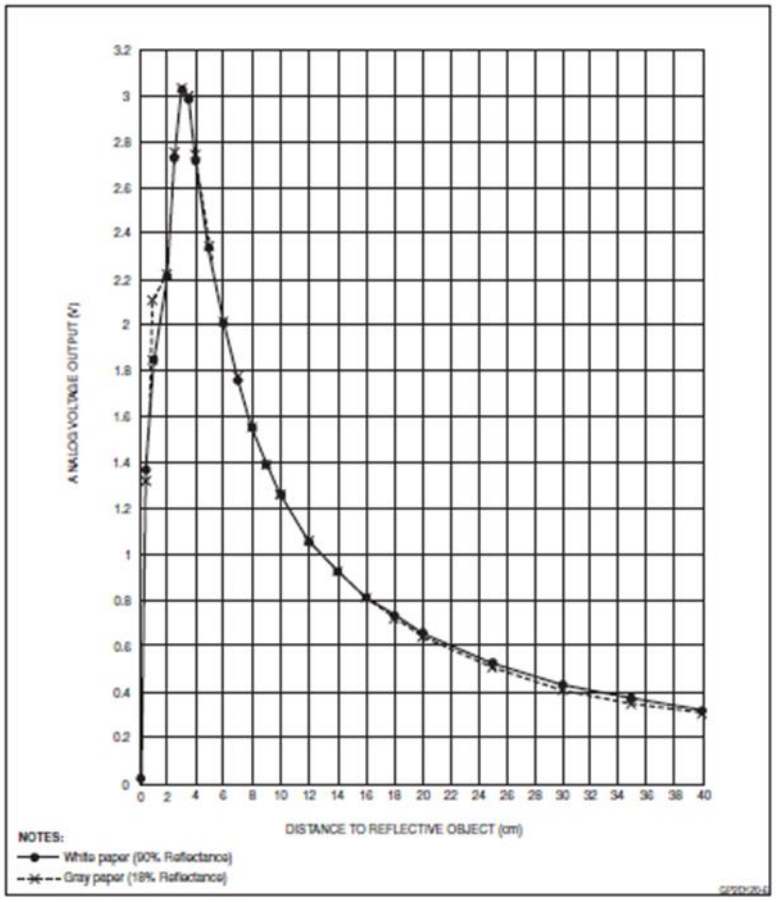


Figura 18: Alcance e precisão do sensor Sharp2D120X

Microcontrolador com Módulo Wi-Fi – ESP8266-Thing

O módulo *Wi-Fi* ESP8266-Thing (Figura 19) foi escolhido devido ao seu pequeno tamanho, baixo custo e eficiência energética, mas principalmente por ser programável através do Arduino IDE e por ser capaz de realizar comunicações *Wi-Fi*, I^2C e Série [17]. A utilização dada foi o envio de dados entre os dois módulos separados fisicamente (*Headset* e UPC).

Figura 19: Placa de desenvolvimento com módulo Wi-Fi - ESP8266-Thing

A ESP8266-*Thing* é uma placa de desenvolvimento constituída por um módulo *Wi-Fi*, que funciona a uma largura de banda de 2.4GHz, por um porto I^2C, por um porto Série, e por um microcontrolador com uma frequência de 80MHz. A ESP possui um *bootloader* série embutido que permite que ela seja programada facilmente através de um simples conversor de 3.3V de USB (*Universal Serial Bus*) para Série. A ESP precisa na sua alimentação de 3.3V, podendo esta ação ser feita através de um cabo USB, de uma bateria LiPo (*Lithium Polymer*) de 3.7V, ou através dos pinos Vin e GND (*ground*).

Microcontrolador – Arduino Mega2560

O Arduino Mega 2560 (Figura 20) é uma placa que possuí o microcontrolador ATmega2560 do tipo CMOS 8-bit e que opera a 16Mhz. Esta placa é constituída por 4 portos UART (*Universal Asynchronous Receiver Transmitter*), 2 portos I^2C, 54 pinos digitais,16 pinos analógicos, 2 saídas de alimentação: uma de 3,3V e outra de 5V; e 3 tipos diferentes de entradas de alimentação: um conector USB do tipo B; um *power jack;* e o pino Vin (em conjunto com um pino GND). Estas duas últimas entradas suportam tensões entre 6 e 20V (sendo o recomendado entre 7 e 12V) que vão diretamente para um regulador de tensão interno, uma vez que o ATmega2560 apenas suporta tensões no intervalo 4,5 a 5,5V. [18] [19]

Figura 20: Microcontrolador Arduino *Mega2560 [18].*

O Arduino Mega2560 foi utilizado no projeto por apresentar uma velocidade de processamento adequada, uma quantidade de memória interna satisfatória, diversos barramentos e protocolos de comunicação e um grande número de portas digitais e analógicas. Tornou-se também vantajoso o facto de este pertencer à plataforma Arduino, que é considerada *open-source* e *open-hardware*, o que permite facilmente adquirir informações úteis bem como aplicá-las. Esta plataforma utiliza uma linguagem de programação baseada em C/C++.

3.2. Módulo de controlo motriz e conversores

Controlador de potência Roboteq HDC 2450

O módulo HDC2450 (Figura 21) é projetado para converter comandos recebidos de um rádio RC (*Radio Control*), analógicos, *joystick*, modem wireless, PC (via RS232 (*Recommended Standard* 232) ou USB) ou microcomputador em saídas de tensão e correntes elevadas para alimentar um ou dois motores DC [20].

O controlador possui um microprocessador de alto desempenho de 32 *bit*, entradas de *encoders* de quadratura e um alto número de entradas/saídas analógicas, de pulso e digitais que podem ser mapeadas para diversas funções. Os dois canais dos motores do controlador podem ser operados independentemente ou misturados para definir a direção e rotação do veículo coordenando o movimento de cada motor.

Figura 21: Módulo de potência [20]

LM7809 – Regulador de tensão linear

O LMV7809 foi utilizado para se poder alimentar o Arduino através de uma bateria de 13.6V [21].

Um regulador de tensão linear é um componente eletrónico que independentemente das variações de tensão à sua entrada ou das condições da carga, consegue fornecer à saída um valor de tensão inferior e constante. É constituído por filtros ativos (com o uso de transístores) controlados por um amplificador de ganho diferencial

que compara a tensão de saída com a de referência, ajustando estes mesmos filtros de modo a manter a tensão de saída constante.

3.3. Barramentos e protocolos de comunicação

Barramento I^2C

O barramento I^2C foi necessário para a comunicação entre os microcontroladores e alguns sensores.

O barramento I^2C foi desenvolvido para diminuir a complexidade que existia para se fazerem trocas de informações entre vários dispositivos digitais de circuitos integrados. O I^2C responde a essa necessidade porque apenas requer dois sinais: SDA (*Serial Data*) e SCL (*Serial Clock*) para realizar comunicações síncronas entre múltiplos dispositivos que se encontrem no barramento, estando estes individualmente endereçados e definidos com o respetivo cargo da relação mestre/escravo que o I^2C possuí.

Qualquer dispositivo pode enviar dados para o barramento (transmissor) ou pode receber dados do barramento (recetor) mas é o mestre quem inicia a transferência, quem gera o sinal de relógio e quem termina a transferência. Durante este processo todos os outros dispositivos endereçados são considerados escravos.

Nos barramentos I^2C os sinais SDA e SCL fluem em ambos os sentidos e estão conectados a uma alimentação positiva através de resistências *pull-up* (Figura 22), porque nestes barramentos os comandos são do tipo "*open drain*" o que significa que é apenas possível colocar os sinais a *LOW* e estes só passam novamente a *HIGH*, com o auxílio das resistências, sempre que não exista um dispositivo a ordenar o contrário. [22] [23]

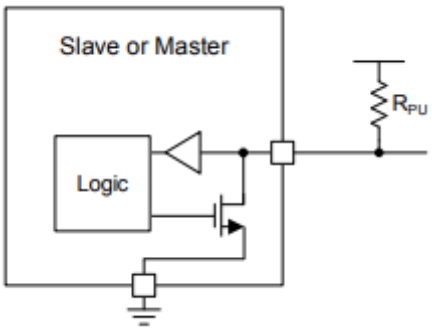


Figura 22: Ligação do barramento I^2C para os sinais SDA/SCL [24]

Procedimento geral das comunicações I²C

Qualquer transação de dados no barramento I²C começa com um *START bit* desencadeado por um mestre (Transição *HIGH→LOW* do sinal SDA quando o SCL se encontra a *HIGH*) (Figura 23), a partir desse momento considera-se que o barramento encontra-se ocupado e todos os outros dispositivos endereçados ficam preparados para receberem ordens. De seguida o mestre envia-lhes 8 *bits* em que 7 correspondem ao endereço do dispositivo com quem pretende comunicar e o 8º *bit* (R/$\overline{W}$) indica se pretende fazer uma escrita ou uma leitura. Apenas responde o escravo com o endereço correspondente e fá-lo enviando um *bit* de confirmação - o ACK (*Acknowledge*), que também é utilizado para confirmar cada receção de *bytes* e indica que já é possível enviar outro *byte*. Uma vez estabelecida a comunicação mestre-escravo inicia-se a escrita ou a leitura do barramento cujos procedimentos estão ilustrados na Figura 24 e diferem da seguinte forma:

- Escrita – Após a receção do ACK referido, o mestre indica o endereço do registo onde pretende escrever, aguarda pela resposta do escravo (outro ACK *bit*) e só depois envia os dados que pretende efetivamente escrever.
- Leitura – Após a receção do ACK referido, o mestre indica o endereço do registo que pretende ler, aguarda pela resposta do escravo (outro ACK *bit*) e de seguida emite um outro *START bit* juntamente com o endereço do escravo e o *bit* R/$\overline{W}$ a 1. Desta forma o mestre liberta-se do canal SDA e passa a ser um mestre-recetor enquanto que o escravo volta a enviar um ACK *bit* e torna-se num transmissor. Assim que o mestre recebe o número de *bits* previsto envia um NACK *bit* (*Non-Acknowledge*) que acaba com a transmissão do escravo, ou seja, dá a leitura por terminada.

Para terminar qualquer comunicação (escrita ou leitura) o mestre emite um *STOP bit* (Transição *LOW→HIGH* do sinal SDA quando o SCL se encontra a *HIGH*) (Figura 23), e o barramento fica novamente livre. [23] [24]

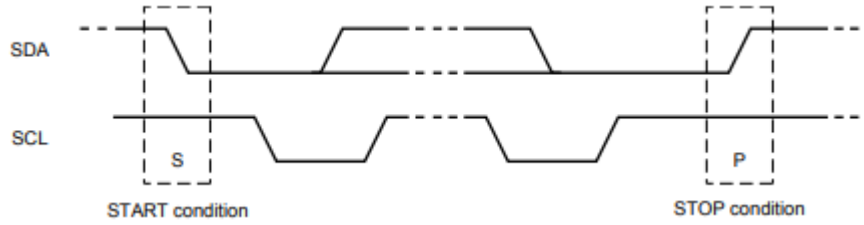


Figura 23: Condições dos START e STOP bits [23]

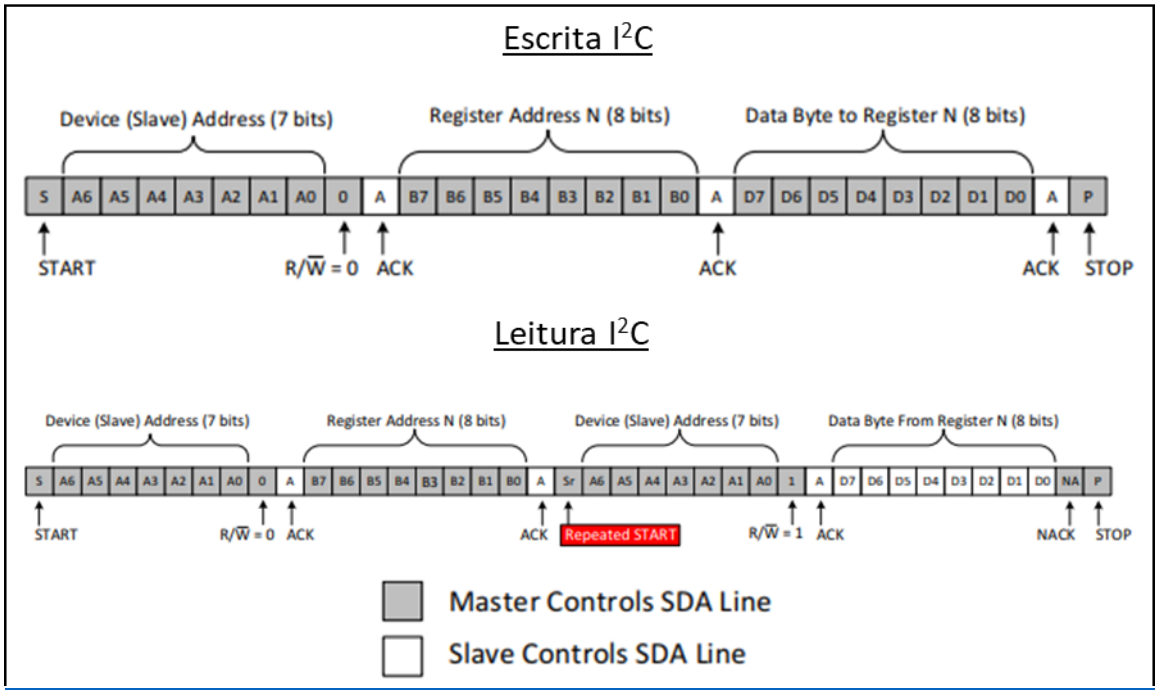


Figura 24: Exemplos de leitura e escrita I²C no dispositivo escravo. Adaptado de [24]

Protocolo UDP

O protocolo UDP (*User Datagram Protocol*) foi usado para a transmissão de dados entre os dois módulos *Wi-Fi* (*Headset* e UPC).

O UDP é um protocolo da camada transporte do modelo OSI (*Open Systems Interconnection*) que funciona em cima do protocolo IP (*Internet Protocol)*. Foi considerado em detrimento do protocolo TCP (*Transmission Control Protocol*) por apresentar uma complexidade de uso inferior. Esta diferença entre protocolos resulta do facto do UDP, ao contrário do TCP, operar com serviços *Best Effort* e *Non-Reliable*, isto é, não há garantia de que todos os dados enviados sejam recebidos e com o mesmo tamanho e a mesma ordem que foram definidos na transmissão, mas oferece um menor *overhead* e menor latência (características importantes para o nosso sistema).

O principal objetivo deste protocolo é servir como interface entre a aplicação da camada superior e o processo de interligação de redes que o IP fornece e tal como a Figura 25 ilustra, o fluxo de dados entre estas camadas OSI começa com o envio da mensagem por parte da aplicação que por sua vez é inserida na secção de dados do pacote UDP (Figura 26) que possuí também as informações sobre: o seu comprimento total em *bytes*; a informação opcional "*checksum*" que é utilizada para a deteção de alguns erros; a porta origem do dispositivo que realiza a transmissão; e a porta destino do dispositivo que

recebe a mensagem. Após o encapsulamento da mensagem todo o pacote UDP é colocado na secção de dados do pacote IP.

As portas UDP referidas anteriormente correspondem a números (0 a 65535) essenciais para distinguir as diferentes aplicações de rede que possam estar a operar num mesmo dispositivo. Desta forma é possível garantir que dentro do dispositivo que possuí o endereço IP alvo de uma receção, a mensagem seja efetivamente encaminhada para a aplicação desejada e com a identificação correta da aplicação que a transmitiu. [25]

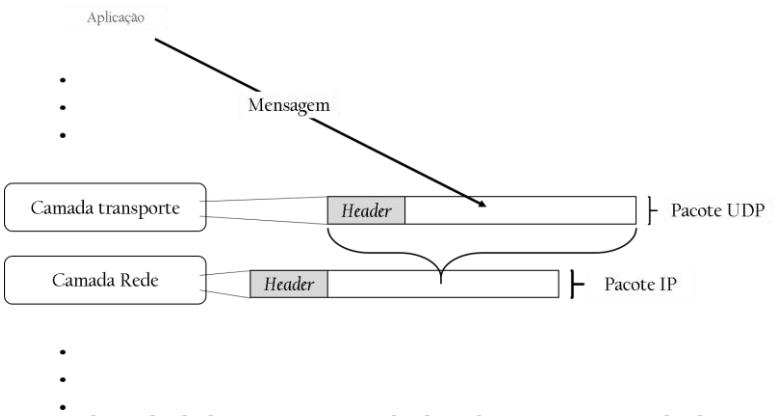


Figura 25: Fluxo de dados entre a camada de aplicação e a camada de transporte

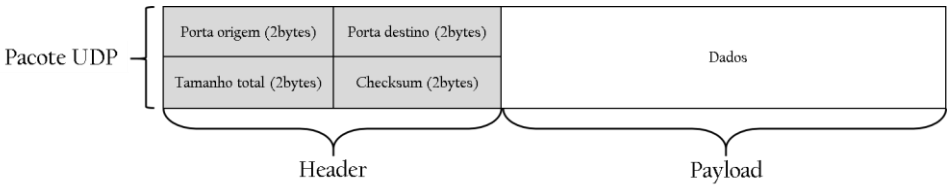


Figura 26: Constituição do pacote UDP

USART

O módulo USART (*Universal Synchronous Asynchronous Receiver Transmitter*) corresponde a um tipo de interface de comunicações série que se podem configurar como modo assíncrono ou síncrono.

Fez-se apenas uso do modo UART cujo nome remete ao facto de não haver um sinal de relógio que sincronize os dados desde o transmissor até ao recetor. Por outro lado, este modo permite operar em *full-duplex*, ou seja, fazer transmissões e receções em simultâneo uma vez que o UART possuí dois meios distintos denominados Tx (transmissão) e Rx (receção). Em alternativa ao sinal de relógio são adicionados dois *bits*

(*START* e *STOP bits*) no início e no fim dos dados a enviar (Figura 27) e todos estes são transmitidos a uma frequência específica (*Baud rate*), configurada com o mesmo valor em ambos dispositivos de receção e de transmissão. Desta forma o recetor sabe quantos *bits* estão a ser transmitidos por segundo e após detetar um *START bit* (transição *HIGH* → *LOW* do sinal) realiza uma contagem de N/2 ciclos de uma outra frequência N vezes superior ao *Baud rate* (no mínimo 16x),para centrar a primeira amostra no meio do pulso do *START bit*, sendo os restantes pulsos amostrados após N ciclos (Figura 28). Depois de o recetor contar o número de *bits* previstos (*Data Frame*), confirma que o próximo corresponde ao *STOP bit* (sinal a *HIGH*) e transfere os dados para o *buffer* local do dispositivo recetor o que significa que a transmissão terminou e que se pode começar outra. [26] [27]

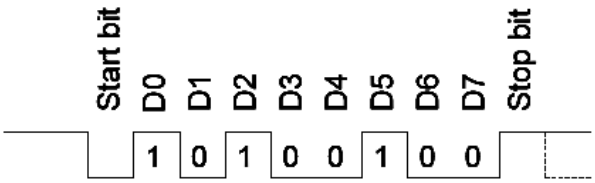


Figura 27: Exemplo de forma de onda de uma transmissão assíncrona [27]

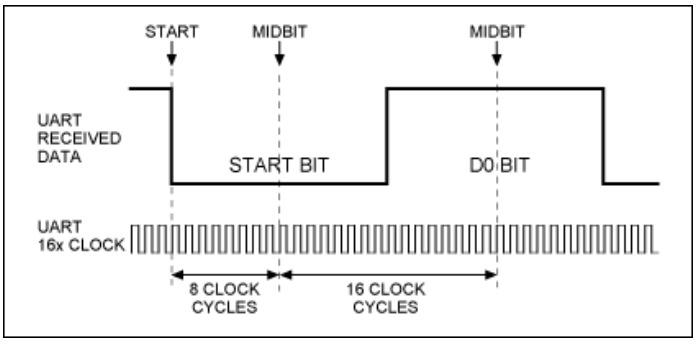


Figura 28: Geração da amostragem no modo assíncrono [28]

Protocolo RS232

Recorreu-se ao protocolo RS232 para se enviar comandos do Arduino para o Controlador Roboteq.

RS232 é um protocolo de comunicação série de transmissão de dados binários ("0" e "1") entre dois terminais, normalmente entre um terminal de dados, DTE (*Data Terminal Equipment*), e um comunicador de dados, DCE (*Data Communication Equipment*). Para que a comunicação seja estabelecida é necessário um mínimo de 3 sinais, Tx, Rx e GND, sendo que o Tx de um dos terminais liga ao Rx do outro e os GND

estão ligados entre si, como mostra a Figura 29. O RS232 suporta tanto transmissão assíncrona como síncrona, codifica o valor "1" entre -3 e -15V e o valor "0" entre +3 e +15V e, como tem dois circuitos separados de transmissão e receção, permite o funcionamento em *full duplex*.

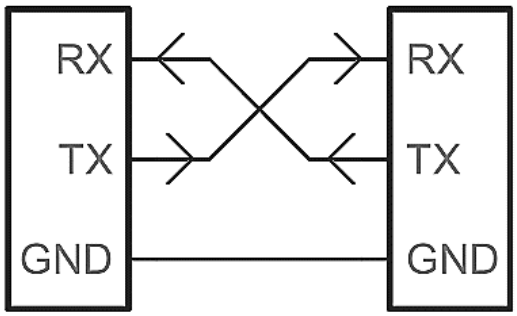


Figura 29: Esquema da ligação R232 [29]

Driver MAX232

Devido a diferenças de níveis de tensão entre controladores (Arduino e Roboteq) teve-se que implementar o MAX232. Este driver é um conversor de níveis de tensão que pode servir para converter níveis RS232 para CMOS como no sentido inverso. Um exemplo da conversão entre níveis encontra-se na Figura 30.

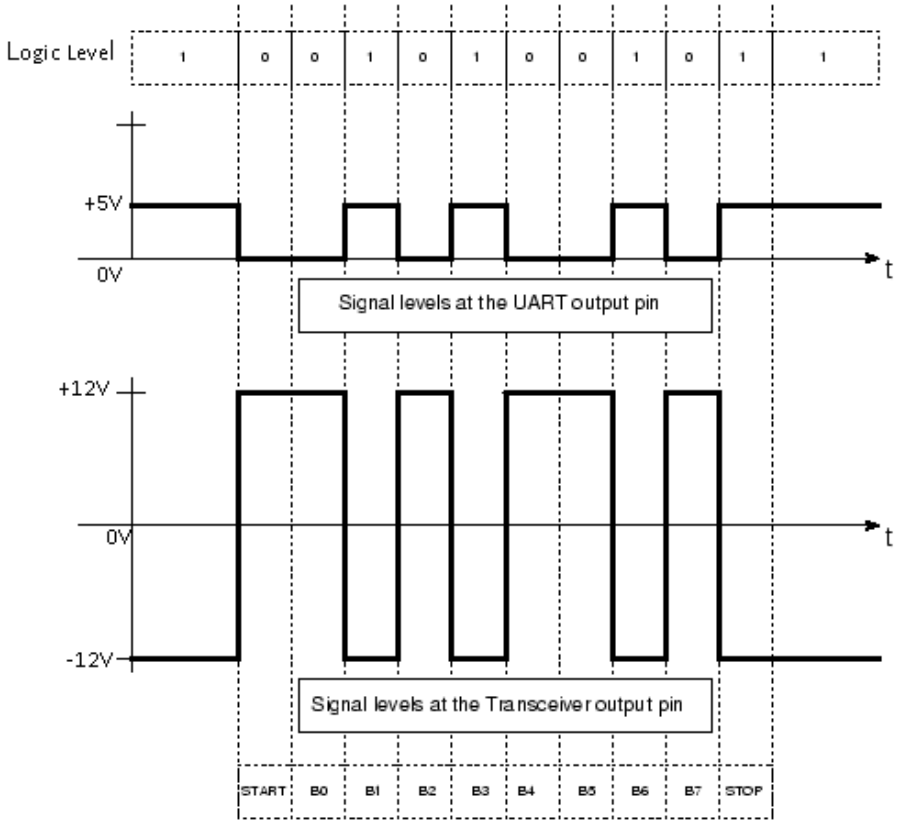


Figura 30: Exemplo de Conversão de dados [30].

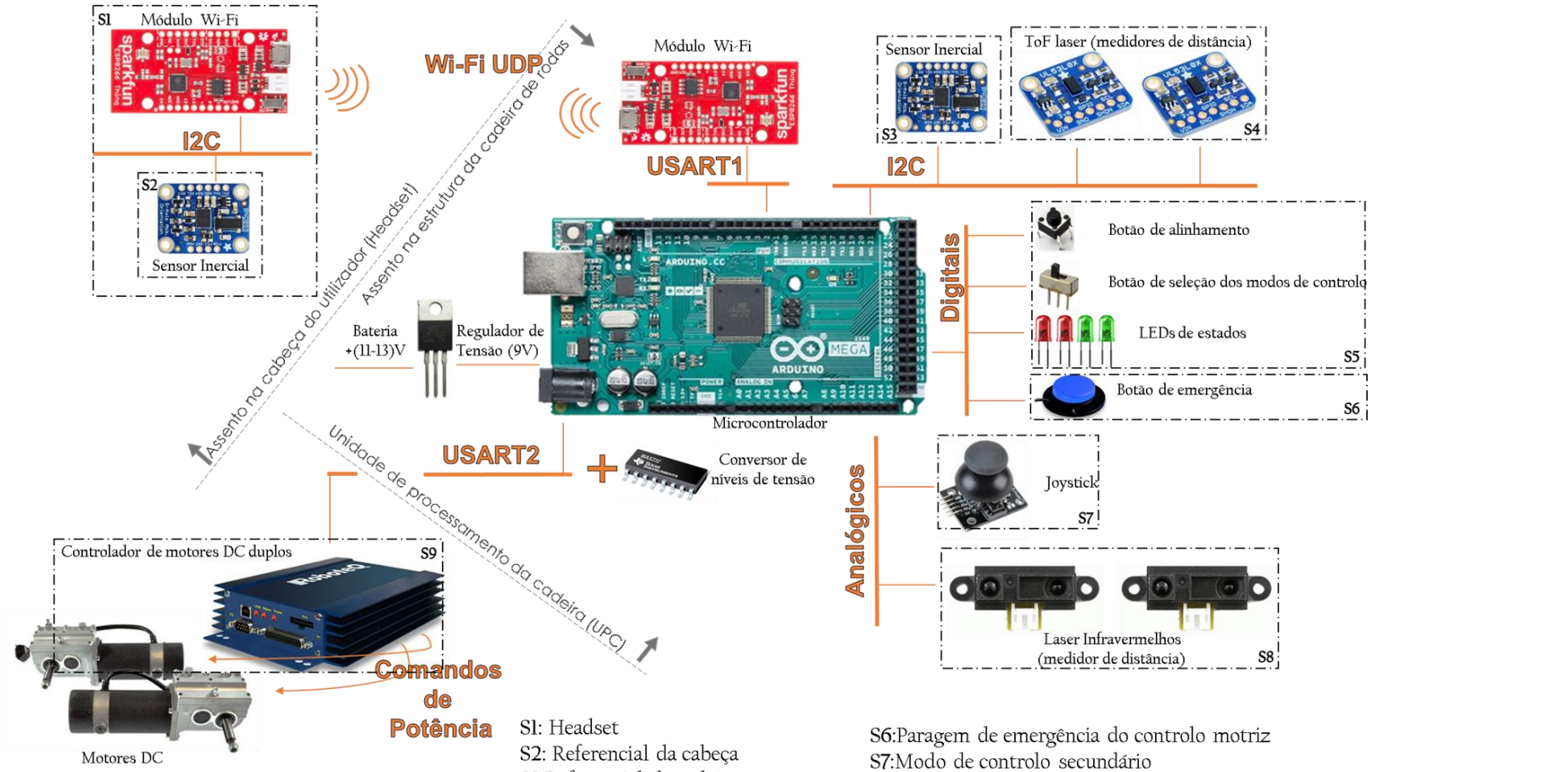
4. Desenvolvimento

Como objeto central e base do desenvolvimento deste projeto, foi utilizada uma cadeira de rodas dobrável que já tinha sido motorizada com o acoplamento de dois motores DC, ligados a cada uma das rodas traseiras. O controlo desses motores é efetuado por um controlador de potência Roboteq HDC2450. Para a sua robotização, foram utilizados diversos componentes e tecnologias de comunicação, ilustrados no esquema da Figura 31.

O sistema encontra-se dividido, fisicamente, em duas partes: um módulo colocado na cabeça do utilizador e um sistema integrado na cadeira de rodas, aos quais se designaram por *Headset* e UPC respetivamente. Este possui dois tipos de interface:

- A interface HMU do sistema - que é responsável pelos comandos de cabeça, sendo composta pelo *Headset* e uma parte da UPC, nomeadamente o BNO055, a ESP8266-*Thing* e o Arduino;

- A interface *joystick* – composta apenas por uma parte da UPC, nomeadamente o *joystick* e o Arduino.



S1: Headset
S2: Referencial da cabeça
S3:Referencial da cadeira
S4:Segurança-Detetção de obstáculos
S5:Painel de interação com o utilizador
S6:Paragem de emergência do controlo motriz
S7:Modo de controlo secundário
S8:Segurança-Deteção da presença da cabeça do utilizador na área de segurança
S9:Controlo Motriz

Figura 31: Diagrama de todos os componentes do sistema implementado

Começando pela UPC, este módulo tem como base o microcontrolador Arduino Mega2560, ao qual se ligaram:

- O módulo *Wi-Fi* (ESP8266-*Thing*) no porto USART. Através de *Wi-Fi* chegam os dados do IMU do *Headset*, enviados também por um módulo ESP8266-*Thing*;

- O botão de calibração de referenciais, o botão de mudança de modos de controlo, o botão de emergência e o painel informativo de LEDs. Todos eles ligados a pinos digitais;

- O *joystick* e dois sensores de distância de lasers infravermelhos (*Sharp* 2D120X). Todos ligados a pinos analógicos;

- Dois sensores de distância do tipo ToF (VL53L0X) e o sensor inercial BNO055. Todos ligados a um barramento I^2C comum;

- O controlador de potência (Roboteq HDC2450), com o auxílio de um conversor de níveis de tensão (MAX232). Ambos ligados via Série (UART);

- Existe ainda uma ligação a um regulador de tensão (LM7809CV), cuja saída se liga aos pinos de alimentação (Vin e GND).

Todos estes componentes encontram-se na placa ou ligados à mesma através de conectores e barramentos conforme ilustrado na Figura 32. Esta placa foi construída de maneira a encaixar sobre a placa do Arduino Mega2560.

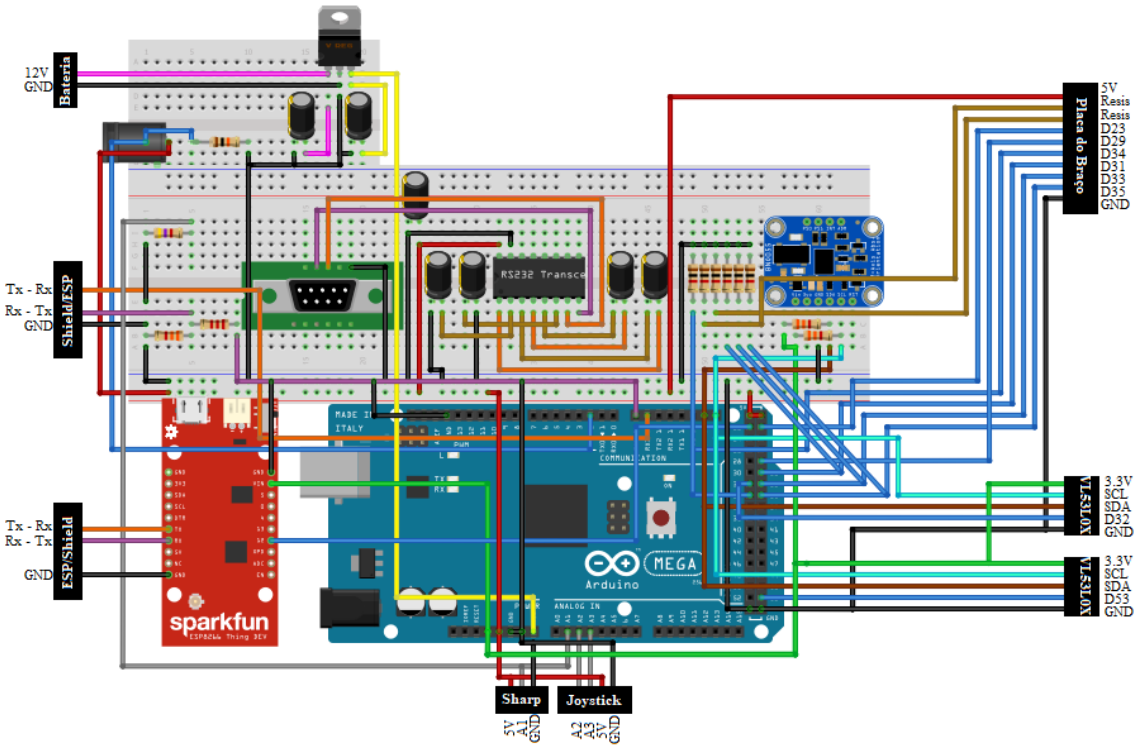


Figura 32: Esquema ligação da UPC

O *Headset* consiste apenas numa placa que integra um sensor inercial BNO55 e um módulo *Wi-Fi*, ESP8266-*Thing*, interligados por um barramento I^2C (Figura 33).

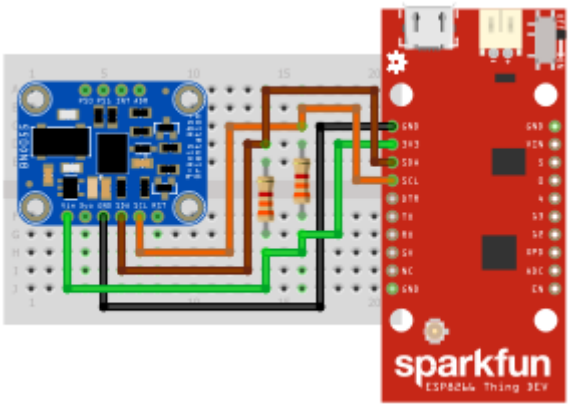


Figura 33: Esquema de ligação do Headset

De seguida detalham-se os modos de controlo da cadeira de rodas usando as duas interfaces referidas:

4.1. Modo de controlo com movimentos de cabeça

O controlo da cadeira através de movimentos de cabeça utiliza a interface HMU que é constituída pelo *Headset* e pela UPC, que comunicam entre si através de tecnologia *Wi-Fi*. Neste modo, o controlo da cadeira baseia-se na diferença entre o referencial da cabeça do utilizador, obtido com o *Headset*, e o referencial da cadeira de rodas obtido com a UPC (Figura 34).

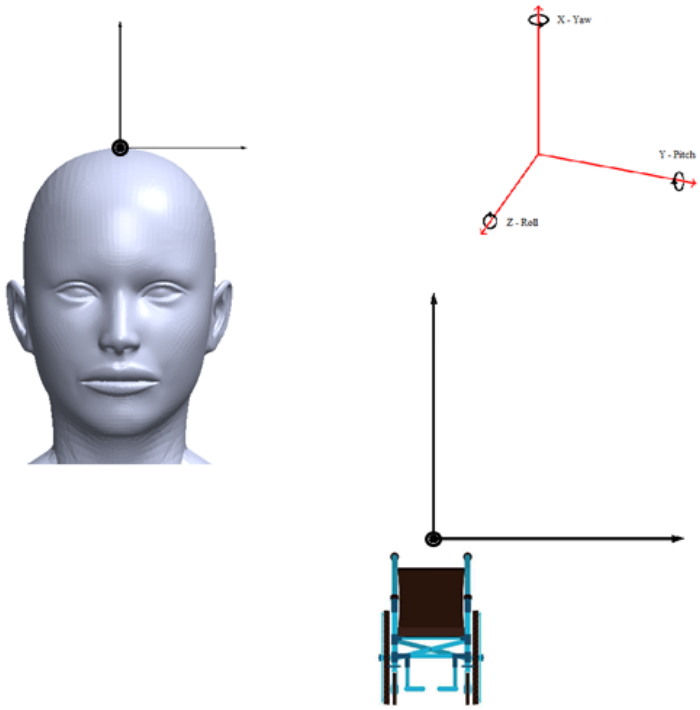


Figura 34: Referencial da cabeça e Referencial da cadeira

Os comandos que ditam os movimentos que a cadeira de rodas deve realizar são obtidos através do cálculo da diferença entre o referencial da cabeça e o referencial da cadeira, já com os seus valores ajustados à posição inicial, à posição de conforto do utilizador. A magnitude dos comandos a enviar ao controlador depende da diferença entre os eixos ZZ, *Roll*, e da diferença entre os eixos YY, *Pitch*, de ambos os referenciais. Em seguida apresentam-se as fórmulas de cálculo das diferenças, bem como a sua implementação e conversão em comandos de velocidade a serem enviados ao controlador (Figura 35):

$$DifZ = Z'cabeça - Z'cadeira$$

$$DifY = Y'cabeça - Y'cadeira$$

```
// Cálculo das diferenças com os ângulos corrigidos
DifZ =(Zcapacete-ZcapaceteAjust)-(Zcadeira-ZcadeiraAjust);// Diferença entre os
                                                          //ângulos Z depois de
                                                          //serem corrigidos com
                                                          //o ajuste

DifY =(Ycapacete-YcapaceteAjust)-(Ycadeira-YcadeiraAjust);//O mesmo para os Y

//Conversão das diferenças entre os ângulos em valores de velocidade para
//os comandos a enviar ao controlador HDC2450
if(DifZ >= 0)     //Se a diferença entre os ângulos do eixo Z
                  //for maior ou igual a 0, o seguinte código é executado
{
  VelPosZ = map(DifZ, 0, 30, 150, 500);//Cria uma função linear que faz corresponder
                                       //um valor de velocidade entre 150 e 500, a
                                       //uma diferença de Z entre 0 e 30,
                                       //continuando mesmo para lá destes limites
}
else//Se a diferença entre os ângulos do eixo Z for menor que 0,
    //o seguinte código é executado
{
  VelPosZ = map(DifZ, 0, - 30, - 150, - 500);//Cria uma função linear que faz
                                             //corresponder um valor de velocidade
                                             //entre - 150 e - 500, a uma diferença
                                             //de Z entre 0 e - 30, continuando mesmo
                                             //para lá destes limites
}
VelNegZ = - VelPosZ;     // O valor negativo da velocidade do eixo Z
VelY = map(DifY, 0, 20, 150, 500);//Cria uma função linear que faz corresponder um
                                  //valor de velocidade entre 150 e 500, a uma
                                  //diferença de Y entre 0 e 20,
                                  //continuando mesmo para lá destes
```

Figura 35: Cálculo da diferencia de referenciais e mapeamento dos resultados em intervalos adequados ao controlo motriz

De seguida são apresentados todos os movimentos admissíveis no controlo com movimentos de cabeça, a comparação entre os referenciais e as diferenças entre os eixos.

Na Figura 36, o utilizador encontra-se na posição de repouso, sendo os referenciais paralelos entre si e a diferença entre os ângulos nula.

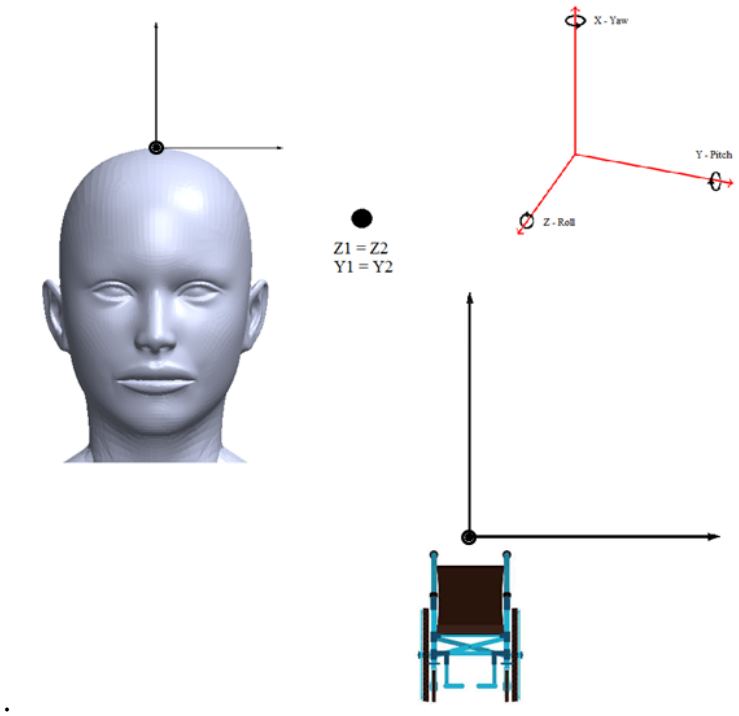


Figura 36: Referenciais na posição de repouso

Na Figura 37, o utilizador encontra-se a realizar um movimento para a frente, deslocando o referencial da cabeça e criando uma diferença positiva entre o eixo Y da cabeça e o da cadeira.

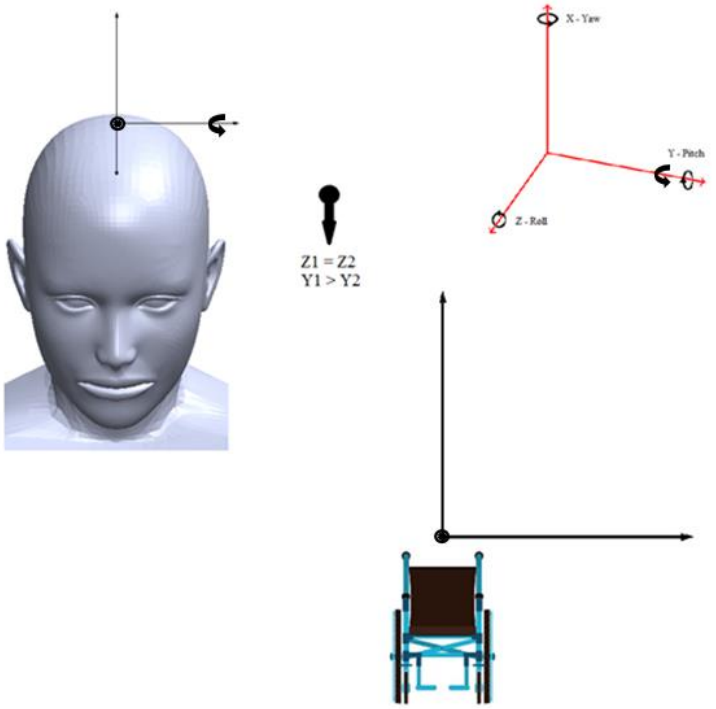


Figura 37: Referenciais do movimento em frente

Na Figura 38, o utilizador encontra-se a realizar um movimento para a esquerda, deslocando o referencial da cabeça e criando uma diferença negativa entre o eixo Z da cabeça e o da cadeira.

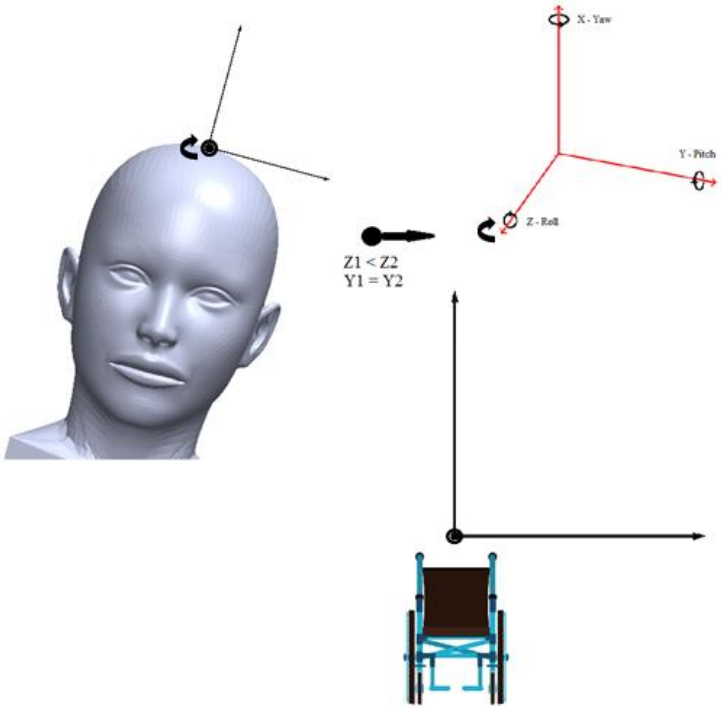


Figura 38: Referenciais do movimento para a esquerda

Na Figura 39, o utilizador encontra-se a realizar um movimento para a direita, deslocando o referencial da cabeça e criando uma diferença positiva entre o eixo Z da cabeça e o da cadeira.

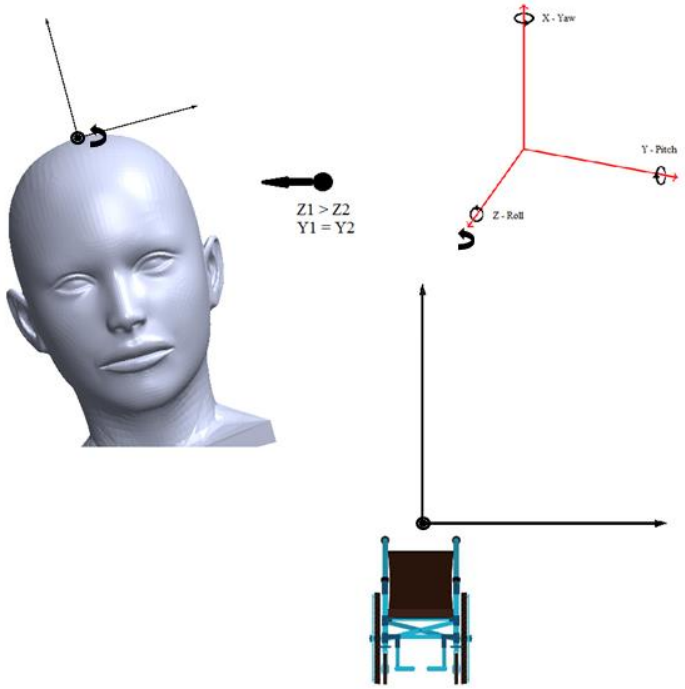


Figura 39: Referenciais do movimento para a direita

Na Figura 40, o utilizador encontra-se a realizar um movimento para a frente e para a esquerda, deslocando o referencial da cabeça e criando uma diferença positiva entre o eixo Y da cabeça e o da cadeira e uma diferença negativa entre o eixo Z da cabeça e da cadeira.

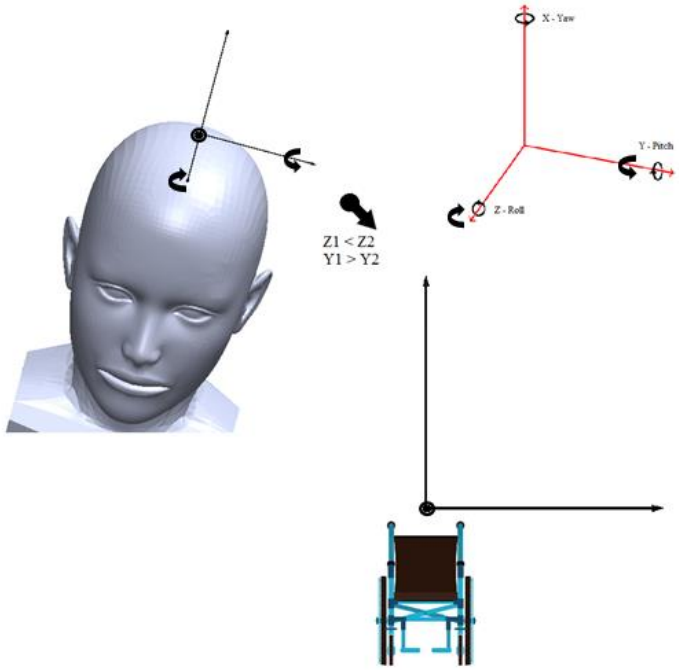


Figura 40: Referenciais do movimento para a frente e para a esquerda

Na Figura 41, o utilizador encontra-se a realizar um movimento para a frente e para a direita, deslocando o referencial da cabeça e criando uma diferença positiva entre o eixo Y da cabeça e o da cadeira e uma diferença positiva entre o eixo Z da cabeça e da cadeira.

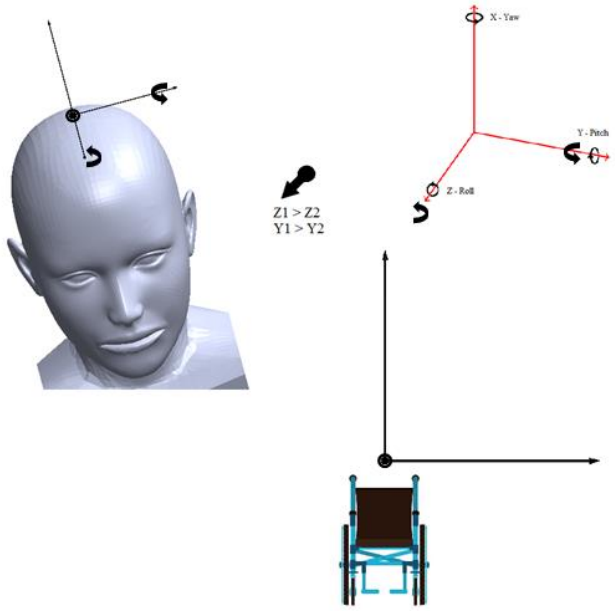


Figura 41: Referenciais do movimento para a frente e para a direita

Resumindo, de forma simplificada, o controlo com os movimentos da cabeça é inicializado com a leitura dos ângulos da cabeça, pelo BNO055, e enviados, pela ESP8266-*Thing*, através de *Wi-Fi*, para a outra ESP que os envia para o Arduino, através da comunicação Série. Recebidos os ângulos da cabeça, o Arduino passa à leitura dos ângulos da cadeira, seguindo-se o cálculo da diferença entre os ângulos da cabeça e os da cadeira. Por fim, converte as diferenças obtidas em comandos de velocidade para o movimento desejado e envia-os ao controlador HDC2450. O esquema encontra-se na Figura 42.

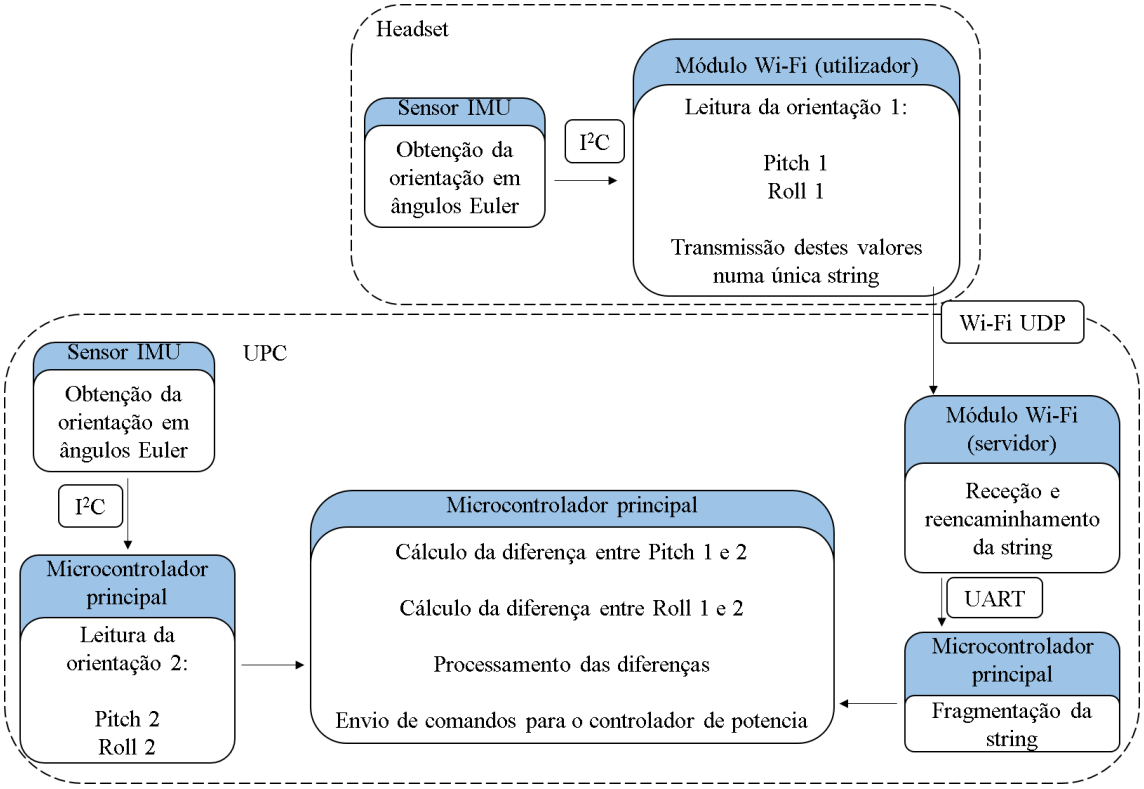


Figura 42: Fluxo de dados no modo HMU

Para garantir que os dados obtidos do BNO055 são o mais precisos possível foi realizada a calibração do sensor através das instruções fornecidas pelo fabricante. Durante o seu funcionamento, o BNO055 inclui algoritmos internos que realizam constantemente a calibração do Giroscópio, do Acelerómetro e do Magnetómetro, no entanto, estas calibrações, sendo imediatas, não possuem nenhum valor de *offset* original que permita corrigir o erro existente durante a calibração. Devido a este problema, o fabricante recomenda a realização da calibração específica sempre que se inicia o sensor, ou que se carregue os valores de uma calibração anterior.

Sendo assim, e com o propósito de realizar a melhor calibração possível, foi utilizado um algoritmo de calibração que permitia a visualização dos valores de *offset* obtidos, algoritmo esse executado dezanove vezes em locais diferentes, registando os valores de cada iteração. Concluída a calibração e tendo obtido valores consistentes e aceitáveis durante as dezanove iterações realizadas, seguiu-se o cálculo da média destes valores de *offset*, passando a serem utilizados para calibrar o sensor (Figura 43).

Posto isto, sempre que o HMU é iniciado, calibra-se o BNO055 do *Headset* ao enviar-lhe, a partir da ESP8266-*Thing*, os valores médios referidos anteriormente. Para estas transmissões, recorreu-se à livraria I^2C "Wire", própria do Arduino, e uma vez que são preciso enviar vários valores, criou-se a função EscreverI2C:

```
void EscreveI2C(int Registo, int Valor)
{
  Wire.beginTransmission(Address);//Inicilialização da transmissão I2C para
                                  //o escravo com o endereço especificado

  Wire.write(Registo);            //Põe em buffer o endereço do registo
                                  //onde se pretende escrever

  Wire.write(Valor);              //Põe em buffer o valor que se
                                  //pretende escrever

  Wire.endTransmission();         //Termina a transmissão e escreve no
                                  //registo especificado o valor pretendidc

  delay(20);                      //Atraso de 20 milissegundos
}
```

Figura 43:Transmissão I^2C para inserir valores nos registos dos dispositivos

Apenas após a calibração, é que se enviam para a ESP8266-*Thing*, também por I^2C, os valores de *Pitch* e *Roll* do IMU do *Headset*. Para estas transmissões recorreram-se a funções da livraria externa "Adafruit_BNO055", tal como ilustrado na Figura 44.

```
void loop()
{
     .
     .
     .
  sensors_event_t event;      //Estrutura que contém todos os
                              //dados produzidos pelo IMU

  bno.getEvent(&event);       //Chamada do evento do BNO055 de
                              //modo a obter-se a leitura dos dados

  Z = event.orientation.z, 4;//Obtensão do ângulo do eixo do z com 4 casas decimais
  Y = event.orientation.y, 4;//Obtensão do ângulo do eixo do y com 4 casas decimais
     .
     .
     .
}
```

Figura 44: Receção na ESP8266-Thing, por I²C, dos ângulos Euler Roll e Pitch

De seguida na ESP8266-*Thing*, agrupam-se os valores de *Pitch* e *Roll* num único pacote, o que simplifica a descodificação dos dados quando estes são recebidos no Arduino, cuja explicação se encontra mais à frente no relatório. Em código, o pacote é uma variável do tipo de *string* que resulta da soma de caracteres com os valores de orientação (Figura 45 e Figura 46).

```
String Pacote;     // Variável local que contém todos os dados a enviar
String DelimA = "a";     //  Variável local que delimita os dados do pacote
String DelimB = "b";     // Variável local que delimita os dados do pacote
String DelimC = "c";     // Variável local que delimita os dados do pacote
String DelimFinal = "?";     // Variável local que delimita os dados do pacote

Pacote = DelimA + Z + DelimB + Y + DelimC + DelimFinal;     // Formação do pacote a enviar
```

Figura 45: Valores de orientação delimitados por caracteres alfabéticos

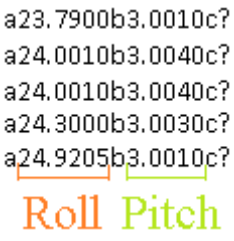


Figura 46: Exemplo dos dados enviados para a ESP8266-Thing da UPC durante a movimentação do BNO055 do Headset

Configuração ESP8266-*Thing*

Depois da formação deste pacote, as transferências dos valores de orientação do *Headset* para a UPC, efetuam-se, sem fios, entre as duas placas ESP8266-*Thing* via *Wi-Fi*, sendo o transporte dos dados efetuado com recurso ao protocolo UDP. Inicialmente,

esta rede *Wi-Fi* encontrava-se associada a um *router* (sala do laboratório VITA), o que limitava a área de funcionamento da cadeira de rodas. Eliminou-se este obstáculo ao criar-se uma rede através da configuração da ESP8266-*Thing* da cadeira de rodas como um *Soft*-AP (*Access Point*).

Com recurso às bibliotecas de código Arduino "WiFiUdp" e "ESP8266Wifi" programaram-se ambos os módulos *Wi-Fi* das placas ESP8266-*Thing* da seguinte forma:

Módulo *Wi-Fi* da UPC (servidor)

Na função setup() do programa definiu-se esta ESP8266-*Thing* como um *Soft*-AP, com o respetivo endereço IP, endereço da *gateway* e da máscara da sub-rede. Utilizou-se apenas um canal (canal 1) dos 13 que o módulo disponibiliza e ocultou-se a rede de modo a que seja apenas acessível aos utilizadores que associarem o nome da rede e a palavra-chave corretamente. Para terminar as configurações, ativou-se a porta UDP que recebe os dados (Figura 47) [31].

```
void setup()
{
  Serial.begin(115200);                          // Inicialização da comunicação
                                                 //série RS232 com uma Baud rate
                                                 //de 115200

  WiFi.mode(WIFI_AP);                            // Configuração como ponto
                                                 //de acesso
  WiFi.softAP(NomeServidor, PassWIFI, 1, true);//Configuração do SSID, password
                                                 //e canal do ponto de acesso e
                                                 //ocultação do SSID

  WiFi.softAPConfig(local_ip, gateway, subnet);//Configuração do IP, gateway e
                                                 //máscara da subnet do ponto
                                                 //de acesso
        :
        :
  Udp.begin(PortaUdp);                           //Inicialização da porta local
}
```

Figura 47: Configuração da ESP8266-Thing da UPC como servidor Soft-AP

Os parâmetros foram previamente definidos com os seguintes valores (Figura 48):

```
WiFiUDP Udp;                                // Objeto UDP
WiFiServer server(4210);                    // Objeto Servidor com a porta 4210
IPAddress local_ip (192,168,4,100);         // Endereço IP
IPAddress gateway (192,168,4,1);            // Endereço da gateway
IPAddress subnet (255,255,255,0);           // Endereço da máscara da subnet

const char NomeServidor[] = "ESP_Server";// Nome do servidor
const char PassWIFI[] = "Wheelchair";       // Password do servidor
char BufferPacote[255];                     // Buffer para o pacote a enviar
unsigned int PortaUdp = 4210;               // Porta local em que se vai fazer o sniffing
int Ciclo = 0;                              // Número de ciclos sem comunicação WiFi, ou
                                            //seja, sem se receber um pacote
```

Figura 48: Parâmetros da configuração da ESP8266-Thing da UPC como servidor Soft-AP

Módulo *Wi-Fi* do *Headset*

Na função setup() desta placa basta estabelecer a comunicação com o AP da cadeira de rodas e ativar também uma porta UDP que neste caso, servirá como saída de dados (Figura 49).

```
void setup()
{
        .
        .
        .
  WiFi.begin(NomeNet, PassNet);//Inicialização da ligação ao ponto
        .                      //de acesso da ESP da cadeira
        .
        .
  Udp.begin(PortaUdp);         //Inicialização da porta local
        .
        .
}
```

Figura 49: Configuração na ESP8266-Thing do Headset para estabelecer comunicação com o Servidor da outra placa

Os parâmetros foram previamente definidos com os seguintes valores (Figura 50):

```
WiFiUDP Udp;                          // Objeto UDP
IPAddress local_ip (192,168,4,101);// Endereço IP
IPAddress gateway (192,168,4,1);    // Endereço da gateway
IPAddress subnet (255,255,255,0);   // Endereço da máscara da subnet

const char* NomeNet = "ESP_Server";// Nome da network
const char* PassNet = "Wheelchair";// Password da network
unsigned int PortaUdp = 4210;       // Porta local do UDP
```

Figura 50: Parâmetros da configuração da ESP8266-Thing do Headset

A ESP8266-*Thing* da UPC está constantemente à espera de pacotes e, sempre que deteta algum (através da função "Udp.parsePacket()"), lê-o e armazena-o num *buffer* local com um tamanho definido de 255 bytes (Figura 51).

```
void loop()
{
  int TamanhoPacote;                              //Variável local que contém
                                                  //o número de bytes do pacote recebido

  TamanhoPacote = Udp.parsePacket();              //Função que retorna o número de bytes
                                                  //do pacote, guardando esse número
                                                  //na variável

  if(TamanhoPacote)                               //Caso se tenha recebido um pacote,
                                                  //é executado o seguinte código
  {
    int LeituraPacote;                            //Variável local que contém o número
                                                  //de byes lidos

    LeituraPacote = Udp.read(BufferPacote, 255); //Função que lê o pacote recebido até
                                                  //aos 255 bytes, e os coloca na variável
                                                  //BufferPacote e que retorna o número de
                                                  //bytes lidos, guardando esse número
                                                  //na variável
```

Figura 51: Receção do pacote UDP que contem os valores de orientação do Headset

Assim que a ESP8266-*Thing* da UPC recebe o pacote, este é enviado para o Arduino através de uma comunicação Série do tipo UART. À semelhança do que foi preciso para o barramento I^2C, para estas comunicações também se utilizaram funções próprias do Arduino, neste caso elas foram as funções do grupo *Serial*. Em ambos os portos UART utilizados no projeto, configuraram-se as ligações Série com valores compatíveis com os requisitos específicos do controlador Roboteq HDC2450 [20]:

- *Baud rate*: 115200;
- *Data frame*: 8bits;
- 1 *START bit*;
- 1 *STOP bit*;
- Sem paridade;
- Sem controlo de fluxo.

Em código, implementaram-se estas configurações apenas com a seguinte função Serial.begin() (Figura 52), uma vez que esta função por defeito, define os outros parâmetros de acordo com as necessidades do projeto, sendo apenas necessário definir a *Baud rate*.

:

```
Serial.begin(115200);    // Inicialização da comunicação
                         //série RS232 com o Arduino
                         //a uma Baud rate de 115200
```

Figura 52: Inicialização da comunicação serial entre o Arduino *e o Controlador de Potência*

Programou-se o Arduino para ler sempre estes dados recebidos até encontrar o caracter "?", que foi o caracter utilizado para indicar o fim de um pacote. Depois, o Arduino inicia o processo de fragmentação da *string* que se encontra no pacote, de modo a obter os valores do *Pitch* e do *Roll* separadamente. Deste pacote, associa os valores entre os caracteres "a" e "b" como sendo o *Roll* e os valores entre os caracteres "b" e "c" como sendo o *Pitch*, convertendo no fim os dados extraídos, de *string* para *float*. Todo este processo encontra-se na função "TrataPacote()" ilustrada na Figura 53.

```
void TrataPacote()
{
  Pacote = Serial1.readStringUntil('?');       //Os dados recebidos são lidos
                                               //até ao ponto de interrogação e
                                               //são armazenados na variável

  Delim0 = Pacote.indexOf('a');                //Encontra o primeiro carater "a"
                                               //do pacote e guarda o índice na variável
  Delim1 = Pacote.indexOf('b');                //Encontra o primeiro carater "b"
                                               //do pacote e guarda o índice na variável
  Delim2 = Pacote.indexOf('c');                //Encontra o primeiro carater "c"
                                               //do pacote e guarda o índice na variável

  PacZ = Pacote.substring(Delim0 + 1 ,Delim1);//Procura a string entre os índices
                                               //especificados e guarda-a na variável
  PacY = Pacote.substring(Delim1 + 1 ,Delim2);//Procura a string entre os índices
                                               //especificados e guarda-a na variável

  Zcapacete = PacZ.toFloat();      // Converte a string correspondente
                                   //ao ângulo Z da cabeça num float
  Ycapacete = PacY.toFloat();      // Converte a string correspondente
                                   //ao ângulo Y da cabeça num float
```

Figura 53: Formação do pacote a enviar para a UPC e que contém os valores de orientação do Headset

O Arduino para além de receber os dados do referencial da cabeça do utilizador, também recebe os dados do referencial da cadeira de rodas através do mesmo processo utilizado pela ESP8266-*Thing* do *Headset*. O microcontrolador procede então para o cálculo da diferença entre o *Roll* e o *Pitch* do *Headset* com o *Roll* e o *Pitch* da UPC, respetivamente. De seguida, realiza-se o processamento dos resultados, isto é, as diferenças obtidas entre os ângulos são traduzidas num movimento específico, dependendo do intervalo em que se encontram, e, fazem-se corresponder a essas diferenças valores de velocidade, obtidos através de uma função linear, este processo é descrito na Figura 54. Por fim, são enviados comandos ao controlador de potência, distinguindo a velocidade, a aceleração e a desaceleração de cada um dos motores que, de forma conjugada, movimentam a cadeira na direção desejada.

```
void TrataDados()
{
  //Diferença entre os ângulos Z e entre os ângulos Y,
  //depois de serem corrigidos com o Offset de calibração
  DifZ = (Zcapacete - ZcapaceteAjust) - (Zcadeira - ZcadeiraAjust);
  DifY = (Ycapacete - YcapaceteAjust) - (Ycadeira - YcadeiraAjust);

  // Conversão das diferenças entre os ângulos em valores de
  //velocidade para os comandos a enviar ao controlador HDC2450

  if(DifZ >= 0)
  //Se a diferença entre os ângulos do eixo Z for maior ou igual
  //a 0, o seguinte código é executado
  {
    VelPosZ = map(DifZ, 0, 30, 150, 500);
    //Cria uma função linear que faz corresponder um valor de velocidade
    //entre 150 e 500, a uma diferença de Z entre 0 e 30,
    //continuando mesmo para lá destes limites
  }
  else
  //Se a diferença entre os ângulos do eixo Z for menor que 0,
  //o seguinte código é executado
  {
    VelPosZ = map(DifZ, 0, - 30, - 150, - 500);
    //Cria uma função linear que faz corresponder um valor de velocidade
    //entre - 150 e - 500, a uma diferença de Z entre 0 e - 30,
    //continuando mesmo para lá destes limites
  }
  VelNegZ = - VelPosZ;     //O valor negativo da velocidade do eixo Z
  VelY = map(DifY, 0, 20, 150, 500);
  //Cria uma função linear que faz corresponder um valor de velocidade
  //entre 150 e 500, a uma diferença de Y entre 0 e 20,
  //continuando mesmo para lá destes
}
```

Figura 54: Conversão das diferenças entre os ângulos, em valores de velocidade para comandos a enviar ao controlador de potência

4.2. Modo de controlo com *joystick*

Embora o principal objetivo do projeto se foque no uso dos sensores inerciais para movimentar a cadeira de rodas, foi implementada uma alternativa bastante comum em cadeiras de rodas elétricas, um *joystick*, permitindo assim manobrar a cadeira mais facilmente.

O fluxo de dados deste modo parte dos movimentos dos eixos do *joystick*, aplicados pelo utilizador, e termina no controlador de potência já com os dados convertidos em comandos Roboteq (Figura 55).

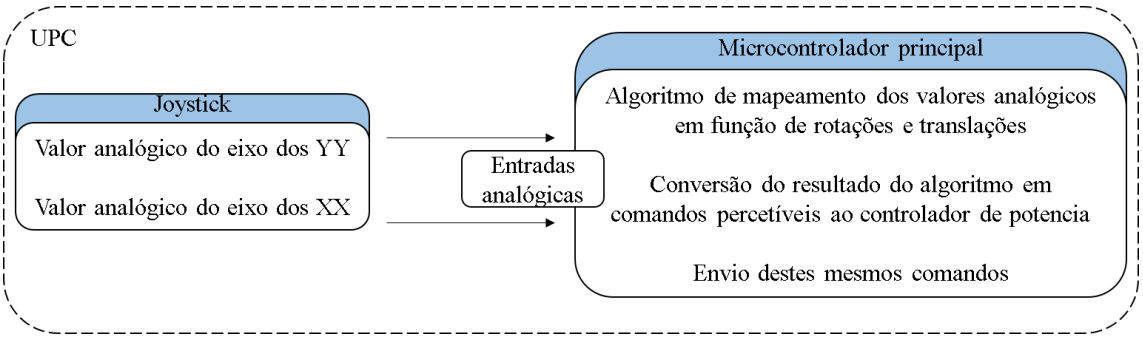


Figura 55: Fluxo de dados no modo joystick

Inicialmente, tentou-se dar uso ao *joystick* VR2-90 (Figura 56) já existente no laboratório VITA, que nunca chegou a ser usado pois exigia a aquisição de um módulo de controlo do mesmo fabricante para que fosse operável. Uma vez que a cadeira de rodas já possuía um controlador (Roboteq HDC2450) e visto que o VR2-90 é um sistema comercializado e cujo algumas especificações técnicas necessárias não são disponibilizadas pelo fabricante, decidiu-se usar um outro *joystick*.

Figura 56: joystick VR2-90 [32]

Optou-se pelo *joystick* Xinda (Figura 57) que consiste num potenciómetro biaxial que controla os movimentos nos eixos dos XX e nos YY. Este *joystick* foi ligado ao Arduino, que o alimenta com 5V e associa os pinos dos eixos a entradas analógicas.

Figura 57: Joystick Xinda [33]

Quando o utilizador atua o *joystick*, este fornece os valores analógicos correspondentes aos eixos XX e YY que são lidos pelo microcontrolador de duas entradas analógicas distintas (A2 e A3).

O controlo com o *joystick* baseia-se na junção do input do eixo XX com o input do eixo Y para a criação de um mapa de movimentos, em todas as direções, dependente desses mesmos inputs. O mapeamento dos movimentos frontais com componente do YY tem prioridade exceto quando este se aproxima da posição central, passando a dar-se prioridade aos movimentos de rotação, os com componente do X. Desta maneira, é possível manter o controlo de todos os movimentos frontais possíveis, conservando ainda os movimentos rotacionais quando o Y se aproxima do centro (Figura 58).

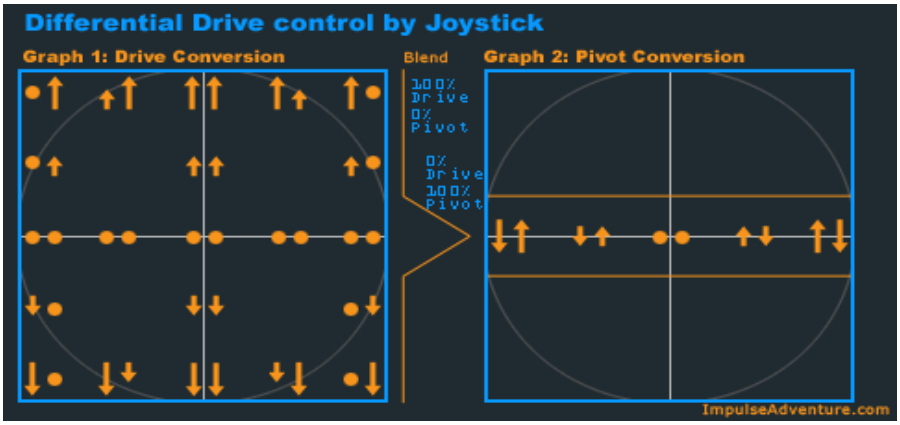


Figura 58: Implementação de um algoritmo usado em robótica que faz o mapeamento de joysticks para ser usado em robôs móveis diferenciais

A implementação do controlo começou com a normalização do valor analógico de cada eixo para um valor de 8 bits com sinal, isto é, para um valor que varia entre -128 e +127, de modo a que as velocidades calculadas pudessem vir dentro desse intervalo. De seguida, é calculada a componente do movimento frontal diagonal dada pelo *input* do eixo X, sendo esta depois ajustada pela componente do eixo Y, seguindo-se o cálculo do

movimento rotacional. Tendo as componentes dos dois movimentos, é calculado qual deles é o prioritário e qual o valor das velocidades ajustadas dos motores. Por fim, converte-se as velocidades obtidas para os seus valores reais e são enviados os comandos ao controlador para que se realize o movimento desejado.

O algoritmo utilizado para mapear os dados analógicos dos eixos do *joystick* encontra-se na Figura 59.

```
// Conversão dos valores analógicos em valores normalizados de um byte com sinal
VelJoyX = map(JoyX, 0, 1023, - 128, 127);//Cria uma função linear que faz
VelJoyY = map(JoyY, 0, 1023, - 128, 127);//corresponder um valor entre - 128 e 127,
                                         //aos valores analógicos dos eixos X e
    .                                    //Y compreendidos entre 0 e 1023
    .
    .

if(VelJoyY >= 0)// Se o valor da variável for maior ou igual a 0,
               //ou seja, para a frente, é executado o seguinte código
{
  VelJoyAux1 = (VelJoyX >= 0) ? 127.0 : (127.0 + VelJoyX);//Cálculo da velocidade do
                                                          //motor 1 em relação ao eixo
                                                          //X do joystick
  VelJoyAux2 = (VelJoyX >= 0) ? (127.0 - VelJoyX) : 127.0;//Cálculo da velocidade do
                                                          //motor 2 em relação ao eixo
                                                          //X do joystick
}
else     //Se o valor da variável for menor que 0, ou seja,
         //para trás, é executado o seguinte código
{
  VelJoyAux2 = (VelJoyX >= 0) ? 127.0 : (127.0 + VelJoyX);//Cálculo da velocidade do
                                                          //motor 2 em relação ao eixo
                                                          //X do joystick

  VelJoyAux1 = (VelJoyX >= 0) ? (127.0 - VelJoyX) : 127.0;//Cálculo da velocidade do
                                                          //motor 1 em relação ao eixo
                                                          //X do joystick
}

VelJoyAux1 *= VelJoyY / 128.0;//Ajustes das velocidades dos motores 1 e
VelJoyAux2 *= VelJoyY / 128.0;//2 com os valores do eixo Y do joystick

VelJoyCurv = VelJoyX;// elocidade do movimento lateral igual
                    //ao valor do eixo do X do joystick

JoyEsc = (abs(VelJoyY) > 32.0) ? 0.0 : (1.0 - abs(VelJoyY) / 32.0);//Cálculo do equilíbrio
                        //entre os movimentos da cadera com os valores do eixo Y do joystick

VelAux1 = (1.0 - JoyEsc) * VelJoyAux1 + JoyEsc * (VelJoyCurv)   //Cálculo da velocidade
VelAux2 = (1.0 - JoyEsc) * VelJoyAux2 + JoyEsc * (- VelJoyCurv);//auxiliar do motor 1 e 2

VelJoy1 = map(VelAux1, - 128, 127, - 500, 500);//Cria uma função linear que faz corresponder
VelJoy2 = map(VelAux2, - 128, 127, - 500, 500);//um valor de velocidade entre - 500 e 500,
                                               //a um valor auxiliar entre - 128 e 127,
                                               //continuando mesmo para lá destes
```

Figura 59: Algoritmo que se utilizou para mapear o joystick, possibilitando o controlo da cadeira de forma fluída aquando da sua utilização

E este algoritmo tem como resultado, a matriz direcional ilustrada na Figura 60.

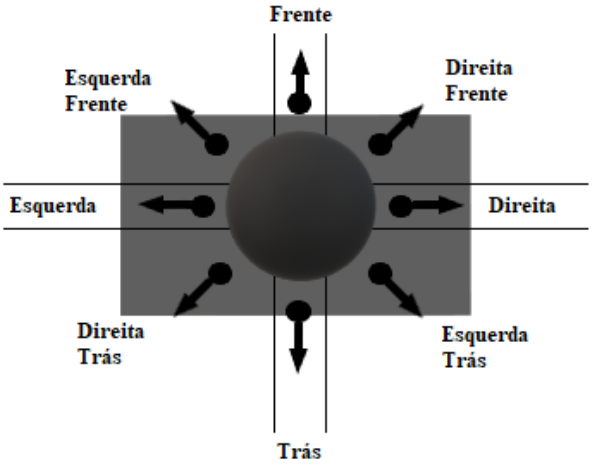


Figura 60: Movimentos que a cadeira de rodas pode fazer ao atuar-se o joystick

4.3. Controlador de potência

Inicialmente, conectou-se o Roboteq HDC2450 a um PC de modo a efetuar automaticamente as suas configurações recorrendo ao *software* Roborun+ da Roboteq. Destas configurações padrão, apenas se alterou o *watchdog* e os níveis de prioridade dos comandos o que fez com que o controlador aceitasse exclusivamente comandos via RS232.

É por este meio que o controlador recebe os comandos do microcontrolador resultantes de qualquer um dos modos de controlo da cadeira, dos quais se usaram 5:

- Comando de velocidade "!G X Y" é o comando principal que o controlador interpreta para enviar valores de potência ao motor X com um valor de velocidade Y (compreendido entre -1000 e 1000);
- Comando de aceleração "!AC X Y" faz com que o controlador aplique um fator Y (compreendido entre 0 e 50000) que define o tempo de aceleração entre os estados de repouso e de velocidade máxima do motor X;

- Comando de declaração “!DC X Y” que é semelhante ao comando anterior mas desacelera entre os estados de velocidade máxima e de repouso.
- Comando “!EX” que faz uma paragem de emergência do próprio controlador deixando-o bloqueado até ser reiniciado ou até receber o comando “!MG”;
- Comando “!MG” que liberta o controlador do estado de emergência.

Os valores de velocidade calculados para cada motor, independentemente da interface, são convertidos para uma *string* com um formato específico que corresponde a um comando percetível ao controlador de potência Roboteq HDC2450. Após a conversão, os dados são então enviados para o controlador de potência que se encarrega de reconhecer os comandos recebidos e convertê-los para comandos de potência que efetivamente controlam os motores.

Este processo foi implementado no microcontrolador Arduino como ilustrado na Figura 61.

```
void Velocidade(int Vel1, int Vel2)
{
  // Formação do comando
  ComandoTotal1 = Comando1 + Vel1;//Junção da string com o comando parcial de velocidade
                                  //do motor 1 com o valor de velocidade pretendido
  ComandoTotal2 = Comando2 + Vel2;//Junção da string com o comando parcial de velocidade
           ⋮                      //do motor 2 com o valor de velocidade pretendido
  // Envio do comando
  Serial2.println(ComandoTotal1);//Envio do comando de velocidade do motor 1, através
                                 //da comunicação série RS232, para o controlador HDC2450
  Serial2.println(ComandoTotal2);//Envio do comando de velocidade do motor 2, através
                                 //da comunicação série RS232, para o controlador HDC2450

  delay(5);     // Atraso de 5 milissegundos
```

Figura 61: Conversão das velocidades calculadas, em ambos os modos de controlo, para comandos percetíveis ao controlador de potência

Os parâmetros “Comando1 (2) ” foram previamente configurados de acordo com a Figura 62.

```
String Comando1 = "!G 1 ";//Comando parcial do nível de velocidade,
String Comando2 = "!G 2 ";//sem a velocidade, do motor 1 e 2 respetivamente
```

Figura 62:Parâmetros que quando agregados aos valores de velocidade formam um comando de velocidade percetível ao controlador de potência Roboteq

RS232

Neste projeto, o RS232 foi escolhido visto ser o recomendado pelo fabricante do controlador HDC2450 e por ser um protocolo bastante fiável. Este protocolo é utilizado para a transferência de dados entre o controlador de potência e o Arduino, podendo também permitir a ligação do controlador a um PC.

Como não existia nenhum cabo que correspondesse às especificações necessárias para fazer a ligação, contruiu-se um, com uma ficha DB25 macho do lado do controlador e uma ficha DB9 fêmea do lado do Arduino.

Apenas se necessitou de 3 dos 25 pinos: o 2, o 3 e o 5. O pino 2 da DB25 corresponde ao Tx do controlador e está ligada ao pino 2 da DB9, passando-se o mesmo para os pinos 3 (Rx) e 5 (GND) da DB25 e os seus correspondentes da DB9 (Figura 63).

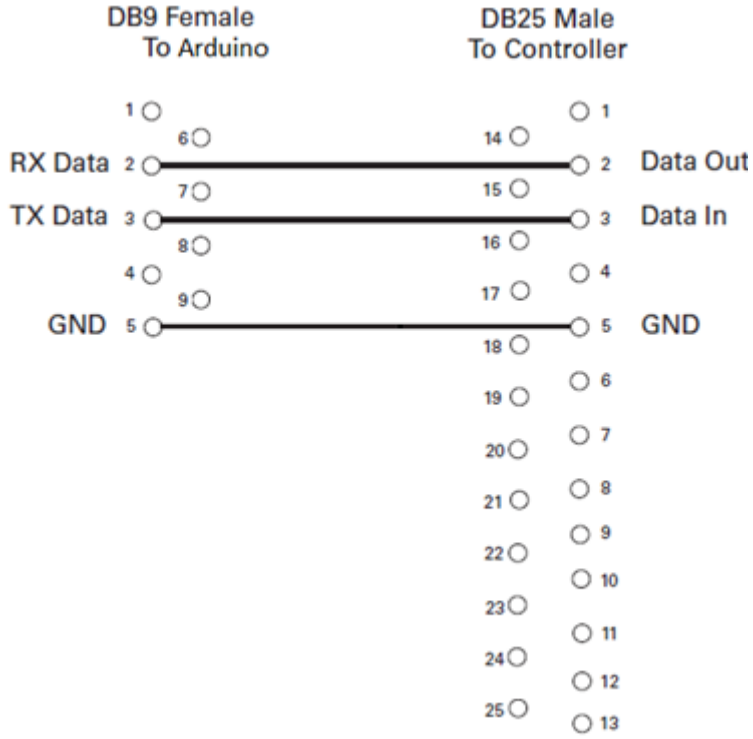


Figura 63: Ligações do cabo final RS232 DB25-DB9 [34]

MAX232

Visto que na comunicação Arduino/Roboteq HDC2450 os valores lógicos diferem bastante, pois o controlador de potência opera com valores entre -12 e 12Volt enquanto o microcontrolador opera entre 0 e 5Volt, recorreu-se ao conversor de níveis de tensão

MAX232 para auxiliar o protocolo RS232. O seu esquema de ligação encontra-se na Figura 64.

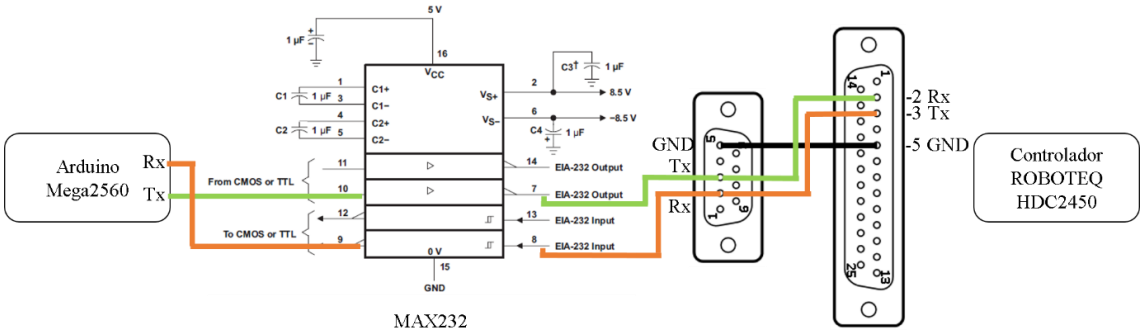


Figura 64: Esquema de ligação Arduino*/Max232/RS232/Controlador*

4.4. Sistemas de segurança

Deteção de Obstáculos

Um dos sistemas de segurança presente nos dois modos de funcionamento da cadeira, ainda que apenas com consequências no modo principal, é o da Deteção de Obstáculos na parte frontal da cadeira de rodas.

Esta deteção de obstáculos foi realizada com dois VL53L0X, sendo a sua função garantir que não existe nenhum obstáculo diretamente em frente à cadeira de rodas, e, no caso de existência, permitir apenas movimentos na direção que se encontre livre. A verificação das distâncias é feita uma vez por ciclo do microcontrolador, independentemente do modo, e consideram-se seguras caso não se encontre nada a menos de cinquenta centímetros dos sensores, sendo esta uma distância pequena o suficiente, de modo a não restringir o movimento da cadeira, e grande o suficiente, de modo a garantir a segurança do utilizador em qualquer situação de perigo, garantindo que a cadeira passa a um estado estático muito antes de chegar a qualquer objeto (Figura 65).

```
if(Dist1 >= 500)       //Se a distância medida pelo VL53L0X 1 for maior
                       //ou igual a 50 centímetros, o seguinte código
                       //é executado
{
  DistSegura1 = true; //A distância 1 é considerada segura
}
else                   //Se a distância medida pelo VL53L0X 1 for menor
                       //que 50 centímetros, o seguinte código é executado
{
  DistSegura1 = false;//A distância 1 não é considerada segura
}

if(Dist2 >= 500)       //Se a distância medida pelo VL53L0X 2 for maior ou
                       //igual a 50 centímetros, o seguinte código é executado
{
  DistSegura2 = true; //A distância 2 é considerada segura
}
else                   //Se a distância medida pelo VL53L0X 2 for menor que
                       //50 centímetros, o seguinte código é executado
{
  DistSegura2 = false;// A distância 2 não é considerada segura
}
```

Figura 65: Código que define estados do sistema de deteção de obstáculos em função da proximidade da cadeira aos mesmos na direção de cada um dos sensores de distância

Por consequência da análise da segurança das distâncias medidas, pode acontecer uma de quatro situações, uma em que não se encontra nada a menos de cinquenta centímetros dos dois sensores e o movimento não é restringido, uma em que apenas o sensor da esquerda deteta uma distância não segura permitindo apenas movimentos para a direita, outra em que apenas o sensor da direita deteta uma distância não segura permitindo apenas movimentos para a esquerda, e, por fim, outra em que ambos os sensores detetam que não existe uma distância segura para o utilizador poder realizar os movimentos desejados sendo então necessário mudar para o modo secundário, de forma a poder sair dessa situação. Todas estas situações estão ilustradas na Figura 66.

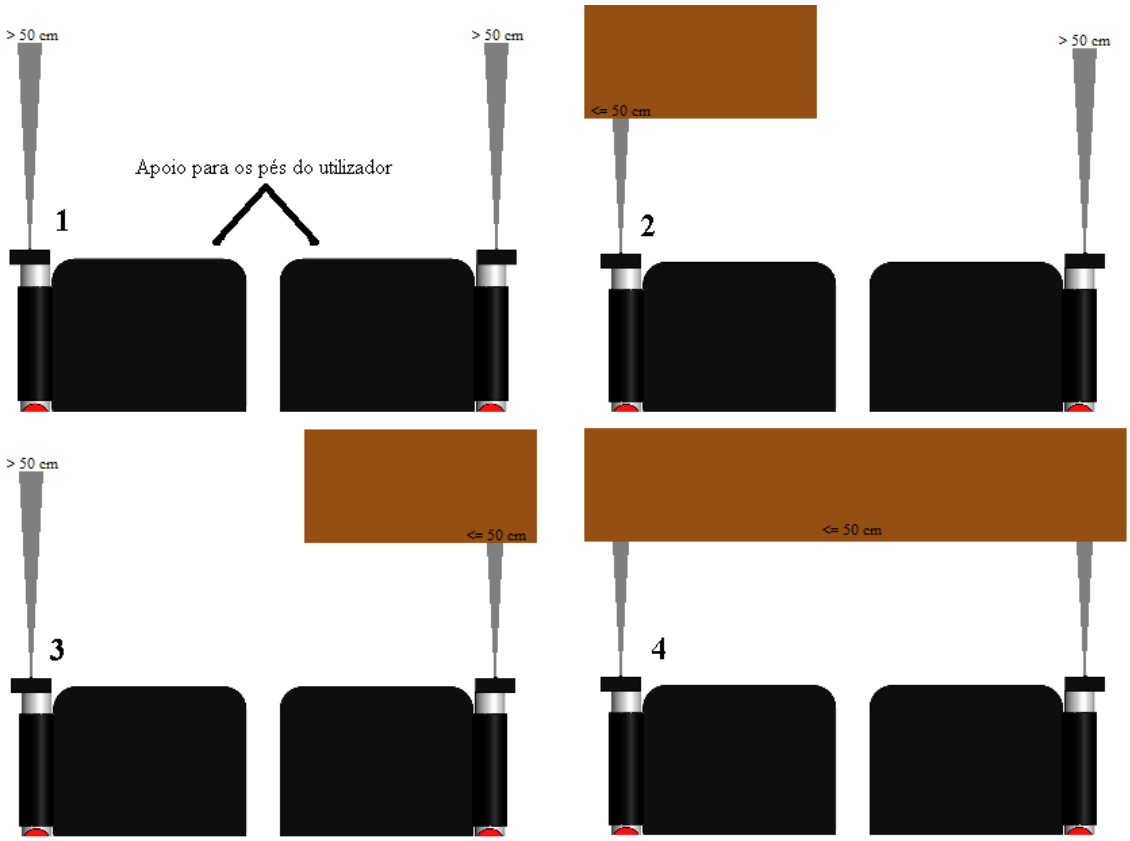


Figura 66: 1) Zona livre de perigo; 2) Cadeira para e só são aceites comandos que resultem em rotações da cadeira para a direita 3) Cadeira para e só são aceites comandos que resultem em rotações da cadeira para a esquerda 4) Cadeira para e apenas se pode movimentar a cadeira para trás (recorrendo ao joystick)

Área de Segurança da cabeça

Um dos sistemas de segurança necessários ao modo controlado pelos movimentos da cabeça é o da Área de Segurança, isto é, uma área que determina se a cabeça do utilizador se encontra numa posição de segurança aquando do controlo da cadeira. Quando a cabeça se encontra fora desta área, a cadeira não efetua qualquer movimento. Assim, é garantido que movimentos bruscos/acentuados da cabeça não intencionais por parte do utilizador, não resultem em movimentos perigosos da cadeira. Para implementar esta área de segurança testaram-se várias soluções, nomeadamente, sensores de força e sensores de distância por infravermelho. Estas duas soluções são apresentadas a seguir.

Sensores de força

Para construir o sistema de deteção da posição da cabeça, inicialmente testaram-se sensores de força resistivos de dois tipos diferentes:

Um potenciómetro de membrana *SoftPot* (Figura 67) que consiste numa fita de espessura maleável cuja resistência varia ao pressionarmos diferentes locais da mesma. Embora sejam normalmente utilizados para medir distância uma vez que a variação da resistência é linear ao longo da fita, no presente trabalho o sensor foi colocado no encosto de cabeça da cadeira de rodas apenas para detetar qualquer força da cabeça sobre o sensor. Desta forma, caso o utilizador não pressionasse a cabeça sobre o encosto, os movimentos da cadeira seriam bloqueados.

Figura 67: Sensor de força resistivo SofPot de 20mm [35]

Um potenciómetro flexível, "Flex 4.5" (Figura 68) que também consiste numa fita maleável, mas cujo valor da resistência varia com a deformação mecânica (dobra) que o próprio sofre. Este sensor seria colocado no encosto de cabeça da cadeira em direção ao utilizador, e à semelhança do sensor SoftPot, seria necessário pressionar nesta fita (dobrá-la) para permitir ao utilizador efetivamente controlar a cadeira.

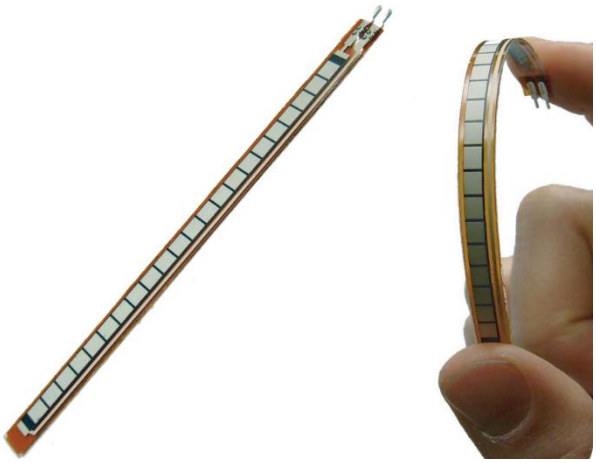


Figura 68: Sensor de força resistivo Flex 4.5 [36]

Nenhum dos sensores mostrou resultados satisfatórios e não fazem parte do protótipo final do projeto pelas seguintes razões:

- O SoftPot mostrou ótimos resultados nos movimentos da cabeça laterais que se traduzem apenas em rotações da cadeira sobre o próprio eixo. Mas por outro lado, para fazer diagonais ou andar em frente, requer que o utilizador incline a cabeça para a frente e, quando o faz, torna-se difícil de simultaneamente pressionar no

sensor porque se encontra no encosto, o que por sua vez compromete o controlo da cadeira.

- O Flex 4.5 para além de ter apresentado logo dificuldades na fixação do mesmo no encosto de cabeça, mostrou também que os valores de repouso são variáveis, ou seja, sempre que o utilizador se afasta totalmente do sensor, o valor da resistência pode não ser o mesmo que se registou da última vez que se encontrava nesta mesma situação. Embora o uso desta fita pudesse ultrapassar os problemas referidos no SoftPot, não tendo um valor de referência inicial constante, tornou-se impossível utilizá-lo para definir a área de operação mencionada.

Sensores de distância infravermelhos

Como nenhum dos sensores anteriores mostraram resultados razoáveis, decidiu-se utilizar dois sensores de distância do tipo infravermelhos.

Este sistema é composto pelos dois sensores *Sharp* 2D120X colocados por detrás das laterais do encosto da cabeça e a sua função é a medição de duas distâncias, uma por cada um, criando uma área na qual se determinou que, desde que a cabeça do utilizador permaneça dentro dela, ela é segura, permitindo o movimento da cadeira. No entanto, caso a cabeça não se encontre dentro dessa área, seja por o utilizador não estar sentado na cadeira, seja por estar demasiado afastado dos sensores, a presença da cabeça não é garantida, não deixando o movimento da cadeira prosseguir até ser novamente detetada a presença da cabeça do utilizador (Figura 69).

```
if(Presenca >= 260)        //Se a distância medida for suficiente para
                           //assegurar a presença da cabeça no encosto,
                           //é executado o seguinte código
{
  ContPresenca++;          //Incremento do número de leituras que garantem
                           //a presença da cabeça

  if(ContPresenca > 2)     //Se houver a garantia de que a cabeça se
                           //encontra dentro da área de segurança, é
                           //executado o seguinte código
  {
    PresencaSegura = true;//A presença da cabeça no encosto é assegurada
  }
}
```

Figura 69: Definição do estado do sistema de deteção de presença da cabeça do utilizador

Segue-se, na Figura 70, uma ilustração deste sistema de segurança.

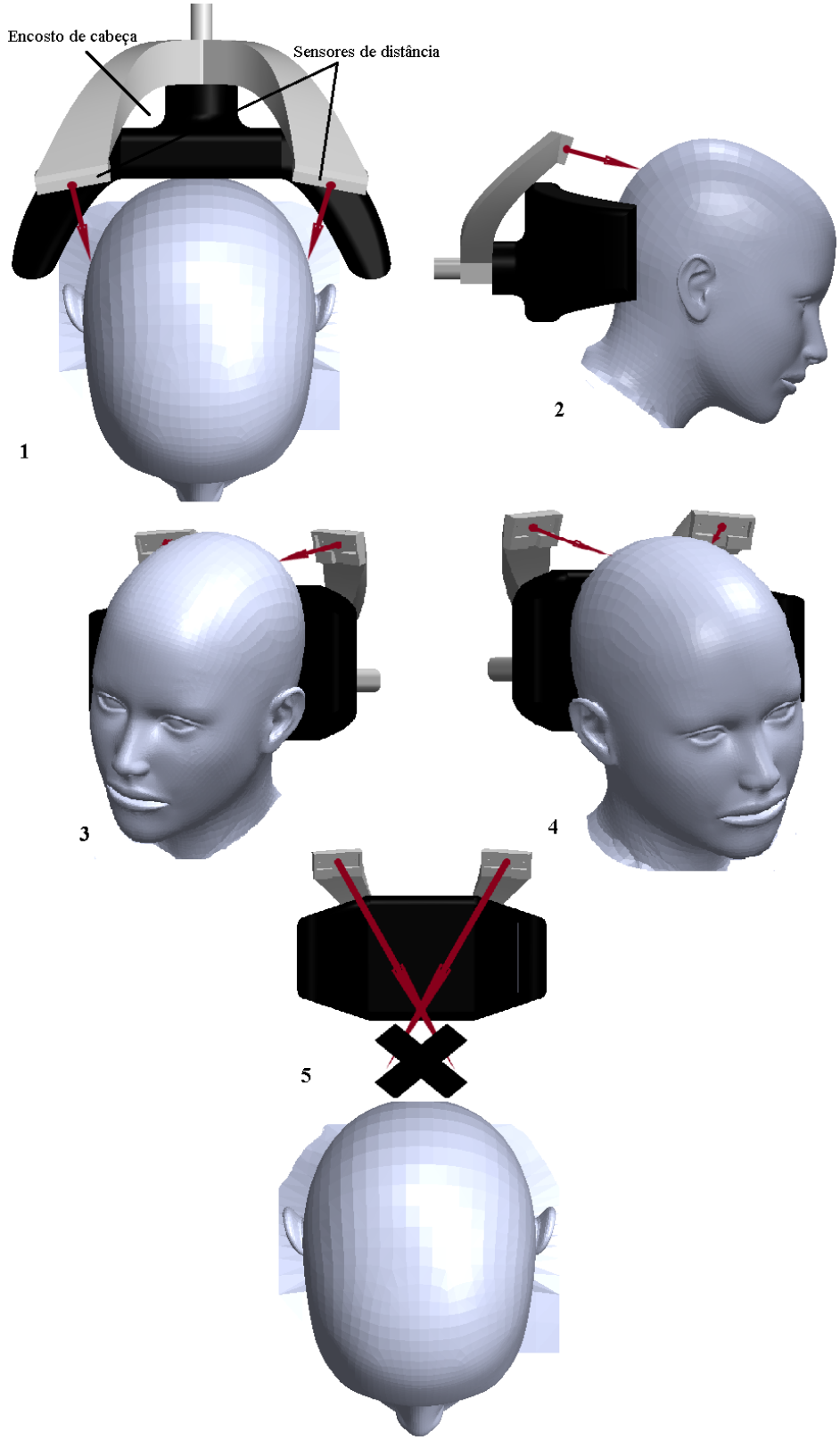


Figura 70: Sistema de deteção da presença da cabeça do utilizador. 1) Posição neutra; 2),3) e 4) Deteção da cabeça durante a execução de comandos HMU; 5) Presença da cabeça do utilizador não é garantida, o HMU para a cadeira

Botão de emergência

O sistema de segurança mais reativo na cadeira de rodas deste projeto é ativado pelo utilizador sempre que carrega no botão de emergência, uma vez que este para o controlo motriz da cadeira, independentemente do modo de funcionamento em que se encontra e do movimento que está a executar. Especificamente, o pressionar do botão, ativa uma interrupção no Arduino que envia logo uma *string* ao controlador de potência com o comando de emergência. Este comando é rapidamente interpretado pelo controlador que para os motores e descarta todos os comandos que venha a receber. Todas as interfaces da cadeira de rodas permanecem bloqueadas até que se reinicie a alimentação elétrica da própria cadeira ou até se enviar o comando de desbloqueio ao controlador de potência. Este desbloqueio é efetuado caso o utilizador pressione o botão de emergência durante aproximadamente dois segundos. As configurações da interrupção estão ilustradas na Figura 71.

```
void Emergencia()
{
  long Press = 0;                 //Variável local que representa o número
                                  //de ciclos que o botão de emergência
                                  //esteve pressionado

  if(Bloqueio)                    //Caso o controlador se encontre bloqueado,
                                  //o seguinte código é executado
  {
    while(digitalRead(BotaoSTOP))//Enquanto se estiver a pressionar o botão de
                                  //emergência, o seguinte código é executado
    {
      Press++;                    //Incrementação da variável do número de
                                  //ciclos de pressão do botão de emergência
    }
    if(Press > 150000)            //Caso a variável seja maior que 150000,
                                  //executa-se o seguinte código
    {
      Serial2.println("!MG");     //Comando de cancelamento do bloqueio de
                                  //emergência enviado ao controlador HDC2450,
                                  //através da comunicação série RS232

      Bloqueio = false;           //Passagem do estado do bloqueio do
                                  //movimento da cadeira a falso
       Velocidade(0, 0);          //Comandos de velocidade
    }
  }
 else                             //Caso o controlador não se encontre bloqueado,
                                  //o seguinte código é executado
 {
  while(digitalRead(BotaoSTOP))//Enquanto se estiver a pressionar o botão de
                                  //emergência, o seguinte código é executado
  {
    Press++;                      //Incrementação da variável do número de ciclos
                                  //de pressão do botão de emergência
  }

  if(Press > 100)                 //Caso a variável seja apenas maior que 100,
                                  //executa-se o seguinte código
  {
    TempoAnt = Tempo;             //Atualização do tempo medido anteriormente

    Serial2.println("!EX");      //Comando de paragem de emergência enviado ao
                                  //controlador HDC2450, através da comunicação série RS232

    Bloqueio = true;              //Passagem do estado do bloqueio do movimento
                                  //da cadeira a verdadeiro
  }
 }
```

Figura 71: Interrupção do botão de emergência

Limites de segurança definidos por *sofware*

Outro dos sistemas de segurança implementados foi a limitação da diferença máxima entre os ângulos do referencial da cabeça e do referencial da cadeira.

Esta limitação é executada exclusivamente a nível código e tem como função não aceitar comandos oriundos de movimentos potencialmente perigosos ou indesejados por parte do utilizador, mesmo que a sua cabeça seja detetada dentro da área de segurança. No caso da realização de algum desses movimentos, aqueles em que a diferença dos ângulos passe o valor máximo admitido, são enviados comandos ao controlador para manter a cadeira parada até que o utilizador realize um novo movimento dentro dos limites, tal como ilustrado na Figura 72.

```
// Parado Se os ângulos do movimento da cabeça estiverem dentro deste valores,
//executa o seguinte código
if(abs(DifZ) < 4 && DifY <= 6 || abs(DifZ) > 30 && DifY <= 5 || DifY >= 30)
{
  Velocidade(0, 0);      // Comandos de velocidade
}
```

Figura 72: Sistema de segurança por código que para a cadeira caso os valores de inclinações da cabeça do utilizador excedam os limites definidos

Segue-se, na Figura 73, uma ilustração deste sistema de segurança.

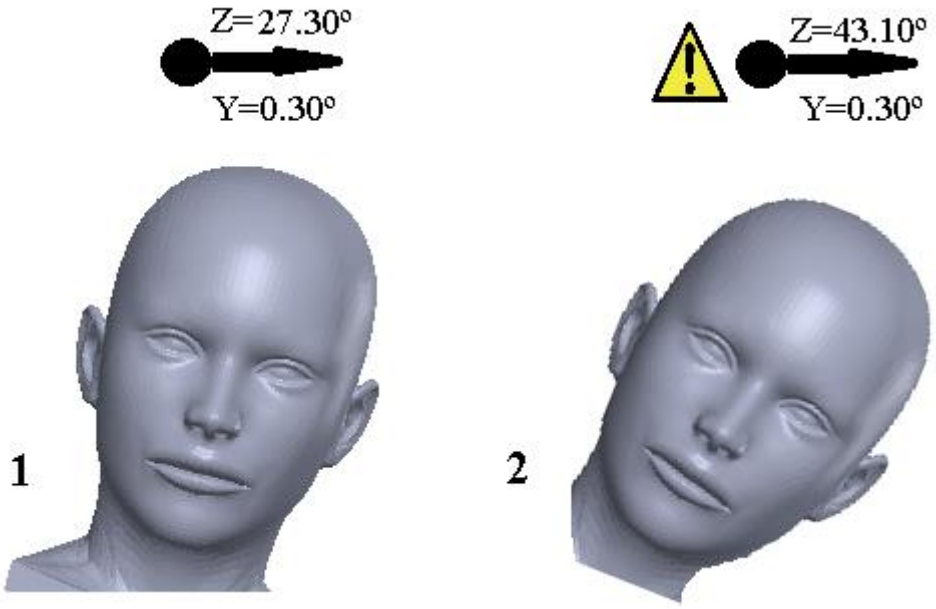


Figura 73: Limitação dos movimentos da cadeira de rodas em função dos ângulos da cabeça do utilizador: 1) Comando aceite para movimentar a cadeira; 2) Comando que para a cadeira devido à inclinação excessiva

Falhas possíveis nos módulos *Wi-Fi*

Um sistema adicional de segurança foi o desenvolvimento de código para quando a comunicação *Wi-Fi* entre as ESP8266-*Thing* falha.

A função deste sistema é garantir que, quando se perde a comunicação *Wi-Fi* ou enquanto esta não for estabelecida, a cadeira não realiza qualquer tipo de movimento, permanecendo estática, até se restabelecer a comunicação. Para tal, manipulou-se um pino digital na ESP8266-*Thing* da UPC que fica *HIGH* sempre que esta placa recebe um pacote, ou seja, ambos os módulos *Wi-Fi* estão operacionais, e fica *LOW* caso não receba nenhum pacote durante 250 ciclos da função loop(), ou caso a placa ESP8266-*Thing* da cadeira de rodas esteja desligada (Figura 74). Este pino está diretamente ligado a um outro pino digital do lado do Arduino que caso esteja a *LOW* faz com que a cadeira não se mova e descarta os valores de *offset* da calibração dos referenciais, de forma a garantir que o utilizador proceda de novo à calibração, uma vez que o ponto neutro anterior poderá ter sido afetado.

```
void loop()
{   ⋮
   if(TamanhoPacote)          //Caso se tenha recebido um pacote,
                              //é executado o seguinte código
   {
     ⋮
     digitalWrite(12, HIGH);  //Escrita no pino 12 da ESP do
                              //nível de tensão alto

     Ciclo = 0;               //Como existiu comunicação WiFi,
                              //o número de ciclos é colocado a 0

     else                      //Caso não se tenha recebido um
                               //pacote, é executado o
                               //seguinte código
     {
       Ciclo++;                //Como não existiu comunicação WiFi,
                               //o número de ciclos é incrementado

       if(Ciclo > 250)         //Caso não exista comunicação WiFi
                               //durante 250 ciclos, é executado
                               //o seguinte código
       {   ⋮
         digitalWrite(12, LOW);//Escrita no pino 12 da ESP do
                               //nível de tensão baixo

         Ciclo = 0;            //Reinicialização do número
                               //de ciclos
       }
     }
}
```

Figura 74: Código utilizado para detetar falhas no estabelecimento da comunicação Wi-Fi entre as ESP8266-Thing

Uma vez que todos os sistemas de segurança já se encontram descritos, apresenta-se o funcionamento geral por completo da cadeira de rodas desenvolvida neste projeto na Figura 75.

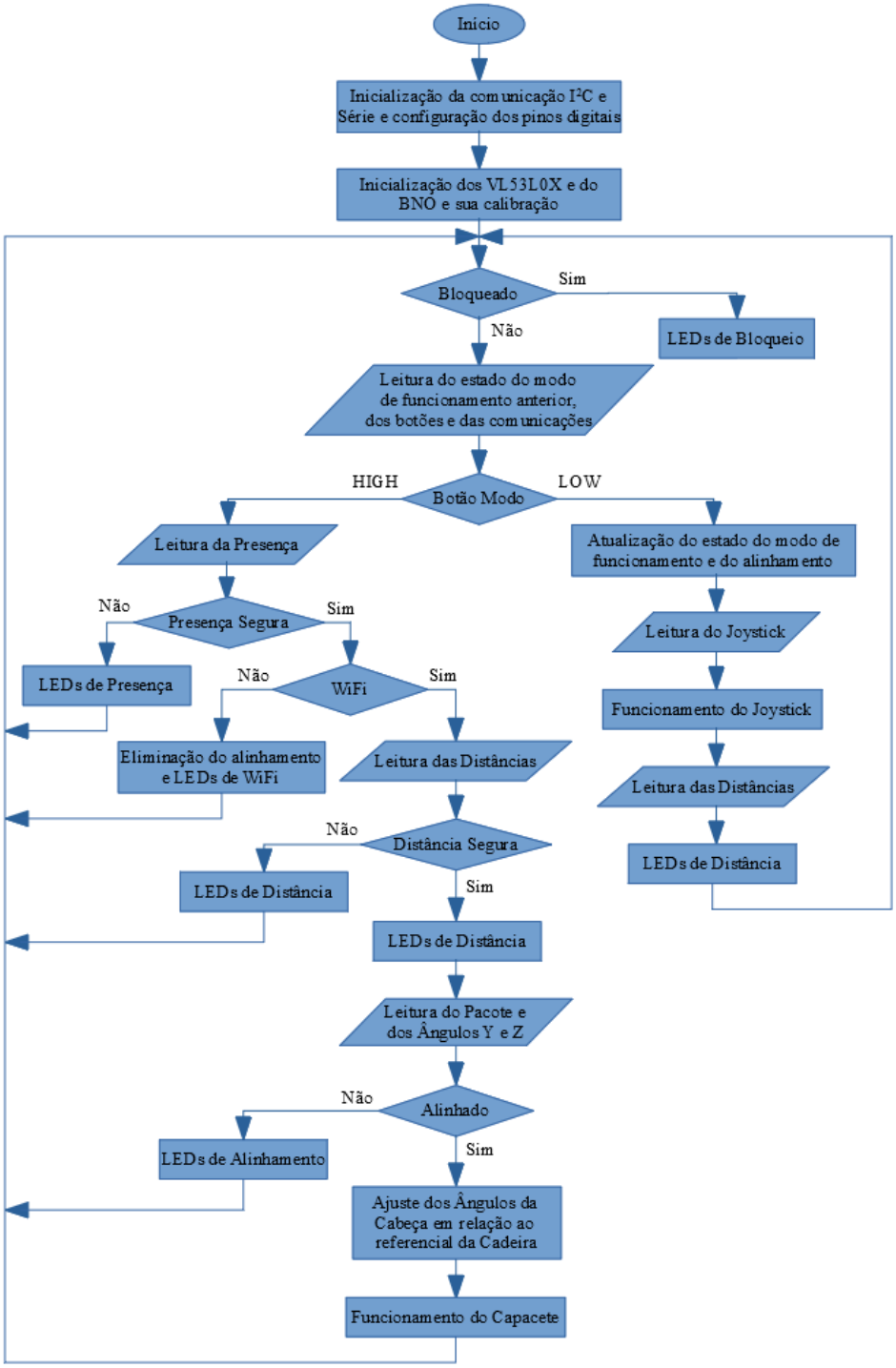


Figura 75: Funcionamento geral da cadeira de rodas desenvolvida.

4.5. Alimentações elétricas

Tensão de alimentação do Arduino

A alimentação elétrica do microcontrolador Arduino Mega2560 começou por ser feita a partir de um computador ligado via USB, ou através de uma pilha 9V conectada à entrada *power jack*. Contudo, encontrou-se uma solução que melhor se adequou aos objetivos do projeto, utilizando uma das duas baterias de chumbo (13.6V) já implementadas para alimentar o controlador de potência e os motores. Tal como referido anteriormente, o Arduino já possui um regulador de tensão interno que suporta valores máximos recomendados de apenas 12V, o que fez com que se adicionasse um outro regulador de tensão a montante do regulador interno (Figura 76).

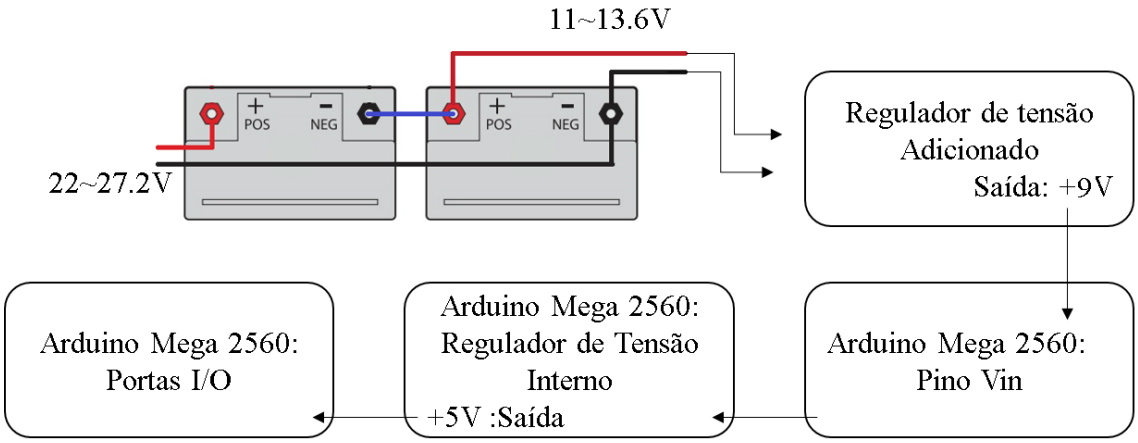


Figura 76:Esquema de alimentação do Arduino.

Na prática, utilizou-se o regulador LM7809CV cuja saída é de +9V e a entrada provém de uma das baterias da cadeira cuja tensão varia entre cerca dos 11 e 13.6V. Como exceção do que se referiu na teoria, existem ruídos de alta frequência vindos da alimentação ou da carga que podem afetar o valor de tensão à saída do regulador. Devido aos mesmos, adicionaram-se dois condensadores polarizados dispostos de maneira a filtrar esses mesmos ruídos (Figura 77).

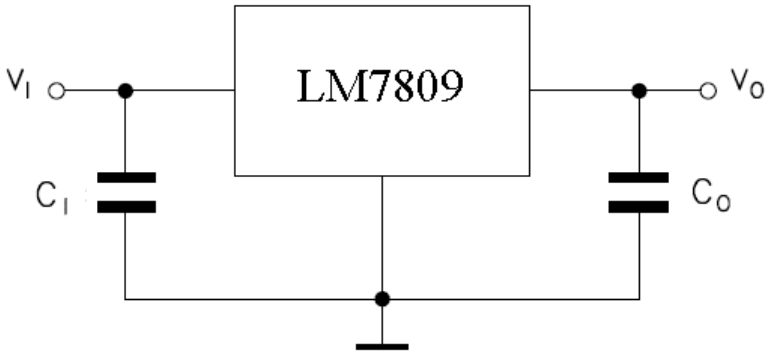


Figura 77: Regulador de tensão positiva. Adaptado de [37]

Embora estes reguladores produzam baixos valores de ruído, não são muito eficazes, uma vez que apresentam valores elevados de potência dissipada, nomeadamente perdas por calor. E apesar de não serem complexos, simplificou-se hipoteticamente o circuito interno do regulador para se obter uma fácil aproximação do valor de potência dissipada P_D assumindo a corrente de entrada I_i igual á corrente de saída I_o [38]:

$$P_D = P_I - P_0$$
$$\Leftrightarrow P_D = (V_i - V_o) \times I_0$$

Substituíram-se os parâmetros Vi, Vo e Io pelos seus valores médios reais e obteve-se o seguinte valor de potência dissipada:

$$P_D = (12.3 - 9)V \times 400mA = 1{,}32W$$

Segundo o *datasheet*, para o tipo de encapsulamento do LM7809 que se utilizou, a RthJA (resistência térmica de junção-ambiente) é de 50ºC/W, portanto a temperatura do regulador pode rondar os:

$$T_{regulador} = 1{,}32 \times 50 = 66°C$$

Perante este valor, decidiu-se acoplar ao regulador LM7809 um dissipador de calor.

4.6. Modelação de peças 3D

Para o funcionamento de todo este sistema, bem como para uma maior facilidade de utilização, seja por quem for, foram criados vários módulos "*plug and play*", isto é, todos os componentes que integram o sistema desenvolvido, podem ser retirados, seja por que motivo for, e conectados novamente sem ser necessário qualquer intervenção ou configuração adicional, tudo através de conectores compostos por cabos elétricos e por uma bainha protetora à sua volta para oferecer maior proteção e durabilidade. Estes módulos são, basicamente PCBs (Placas de Circuito Impresso) com os componentes, ou simplesmente os próprios componentes, e alguns incluem ainda peças feitas na impressora 3D, que permitem o posicionamento destes elementos em partes da cadeira que não seriam possíveis de outra maneira. As PCBs destes módulos foram projetadas usando o *software Fritzing* e concebidas no *software Eagle* e as peças impressas foram desenhadas usando o *software SolidWorks*.

Para além dos componentes eletrónicos aplicados à cadeira, foram construídos diversos componentes 3D, nomeadamente estruturas de suporte para os componentes eletrónicos como outros componentes que melhoram a estética e a usabilidade da cadeira em geral.

O *Shield* do HMU não necessitou de nenhuma peça para a sua montagem, pois encaixava diretamente no Arduino ficando bem preso. No entanto, este último precisou de um método de se fixar à cadeira, até porque é um elemento crítico do modo de funcionamento principal. O BNO055 da cadeira necessitava de permanecer imóvel, eliminando quaisquer erros devidos a qualquer tipo de oscilação durante o movimento, para apenas seguir as manobras da cadeira. Para garantir essas características foi desenhada uma caixa que envolve completamente o Arduino e o mantém seguro (Figura 78), ficando esta presa ao chassis da cadeira por meio de uma placa de metal.

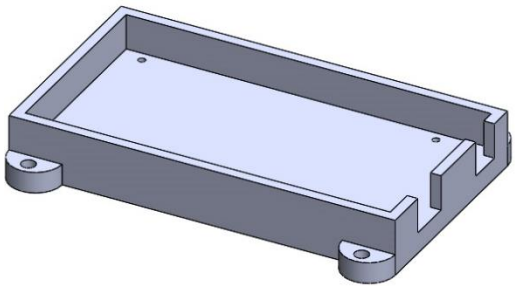


Figura 78: Caixa do Arduino.

Para a PCB do módulo da cabeça foi essencial criar uma estrutura que a suportasse e fixasse na cabeça do utilizador de modo a poder acompanhar todos os movimentos realizados. A solução encontrada para este problema foi a criação de um capacete em forma de bandelete com um suporte para a placa e para a bateria da ESP, que juntamente com uma fita elástica permitem o seu uso para qualquer tamanho de cabeça.

Este capacete foi então dividido em duas partes distintas: as tiras laterais com uma forma que segue o formato lateral do crânio humano e que o pressionam levemente para em conjunto com a fita, o capacete permanecer numa posição de estável; e uma peça na parte central do capacete, que é uma caixa com uma abertura frontal por onde se coloca a PCB e a bateria, e que também possuí duas reentrâncias onde se encaixam as tiras laterais. Tanto as partes como o suporte total estão ilustrados na Figura 79.

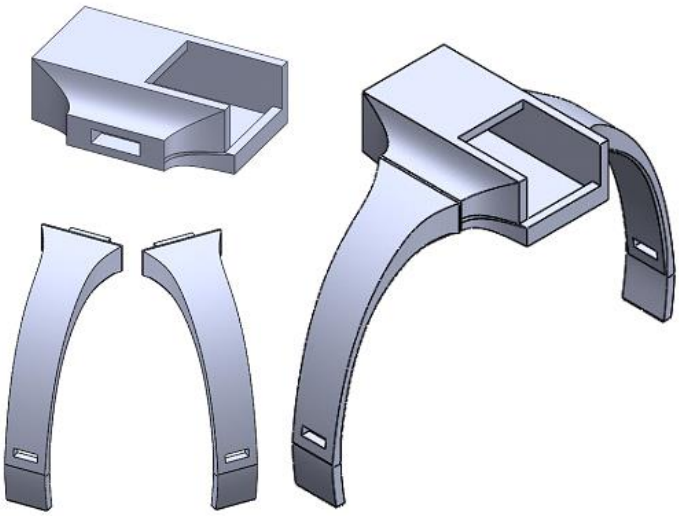


Figura 79: Suporte bandelete versão final

Antes da criação da solução anterior, desenvolveu-se outra (Figura 80) que não ficou de acordo com o esperado. Este primeiro capacete utilizava a mesma peça central do seu sucessor, no entanto, as tiras laterais eram maiores e não acompanhavam a curvatura do crânio, criando algo inadequado para o seu uso, independentemente da pessoa.

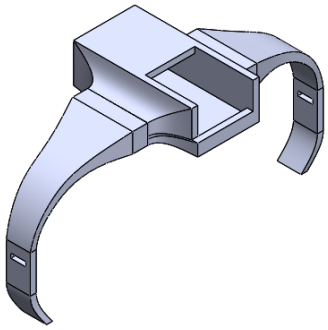


Figura 80: Suporte bandelete versão 1

Para o *joystick* e para a placa constituída pelos LEDs informativos, o botão de modo de operação e o botão de alinhamento, foi decidido fazer-se uma extensão para o apoio do braço direito já existente na cadeira, uma vez que este não oferecia uma localização ideal para os componentes.

Esta extensão do braço foi então projetada para ser um grande apoio que permitisse pousar completamente o braço e os outros componentes referidos. Devido ao grande tamanho da peça, esta teve de ser dividida em três partes, uma dianteira, uma traseira, e, uma central dimensionada apenas para corrigir uma folga entre as anteriores partes. A estrutura e as peças individuais podem ser observadas na Figura 81.

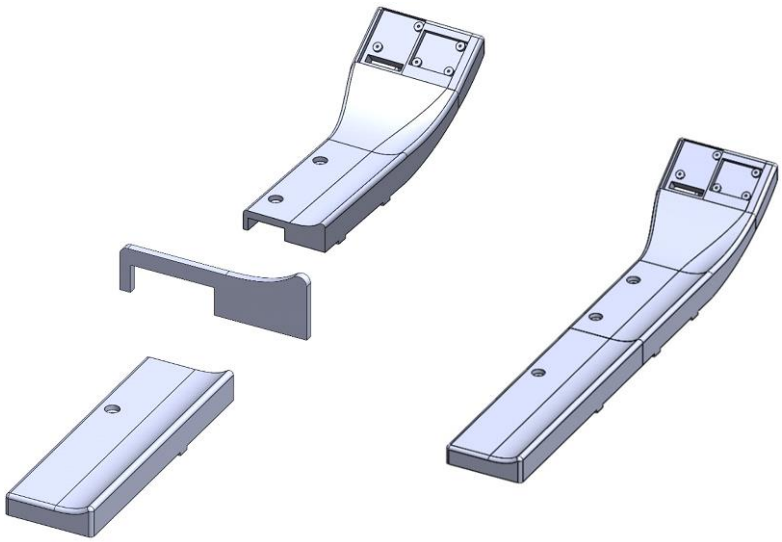


Figura 81: Peças e estrutura final do braço.

Os sensores de distância que pertencem aos sistemas de segurança necessitavam de ser instalados em locais específicos da cadeira. Os VL53L0X foram colocados diretamente nas laterais de fora dos suportes dos pés do utilizador. A escolha destes locais deveu-se à necessidade dos sensores precisarem de estar num local que lhes desse uma área, diretamente à sua frente, desimpedida, para poderem realizar a medição das distâncias em relação a qualquer objeto que se pudesse encontrar à sua frente.

A peça desenhada para esta função, necessitava, portanto, de encaixar nos tubos metálicos que já faziam parte da estrutura da cadeira, de forma a permanecer fixa durante a sua utilização e ficar num local relativamente resguardado de qualquer embate que não se pudesse evitar. Em relação a algum incidente incapaz de ser evitado, a peça precisava também de ser algo robusta e de incorporar o máximo do sensor, de forma a protegê-lo. Por fim, devido à necessidade de o sensor requerer um espaço desimpedido, em relação à caixa, à frente do laser, de forma a poder realizar a medição da distância, a peça foi feita com um buraco na parte da frente, suficiente grande para não afetar as medições, mas, suficientemente pequeno para providenciar a maior proteção possível dentro da restrição apresentada. Sendo assim, conseguiu-se produzir uma peça única, representada na Figura 82 que englobou todas as considerações necessárias para o funcionamento do sensor, isto é, uma caixa relativamente robusta para oferecer a proteção devida ao sensor, mas com uma abertura que não o afetasse, e com um encaixe cilíndrico que entrasse nos tubos da cadeira e que a fixasse neles.

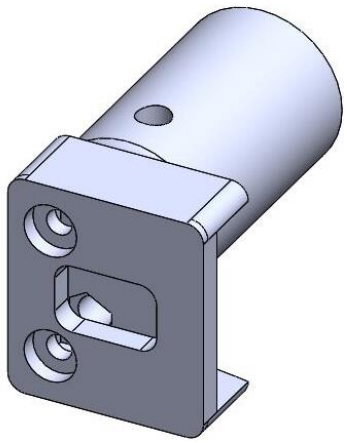


Figura 82: Suporte dos sensores de distância VL53L0X

Os *Sharp* 2D120X necessitavam de se encontrar nas laterais do encosto da cabeça, de forma a se conseguir monitorizar a presença da cabeça do utilizador numa área de segurança para o controlo da cadeira.

Projetou-se então uma peça para segurar os dois sensores diretamente por cima das laterais do encosto da cabeça, e, a maneira encontrada para a prender ao local, foi a utilização do próprio cabo que permite ao encosto segurar-se à cadeira. Teve assim de se desenhar um encaixe que ficasse preso ao tubo, que não roubasse distância de ajuste, para a frente e para trás, ao encosto, e em que se encaixariam os dois suportes que contivessem os sensores. Por fim, tiveram ainda de ser desenhados os dois suportes para os sensores, cada um deles com o comprimento necessário para chegar ao local desejado, e, no encaixe para o sensor, com um declive suficiente para a cabeça poder ser detetada, quer nos movimentos para a frente, quer nas diagonais, quer para os lados.

Ambos as peças em separado e em conjunto, encontram-se ilustradas nas Figura 83 e Figura 84.

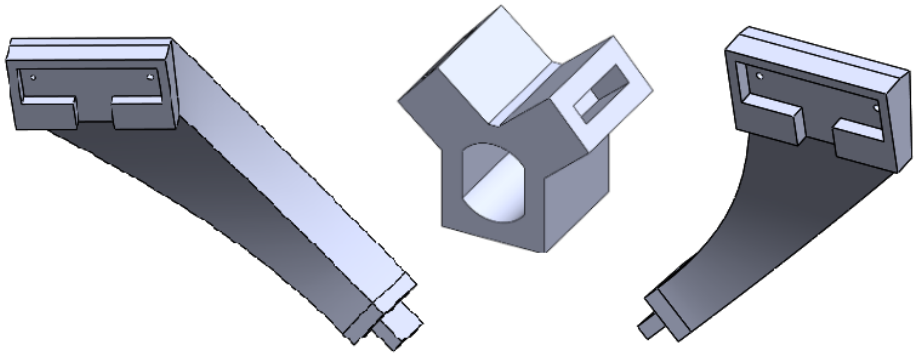


Figura 83: 3 Peças que constituem o suporte ao qual se prenderam os sensores de distância Sharp 2D120X

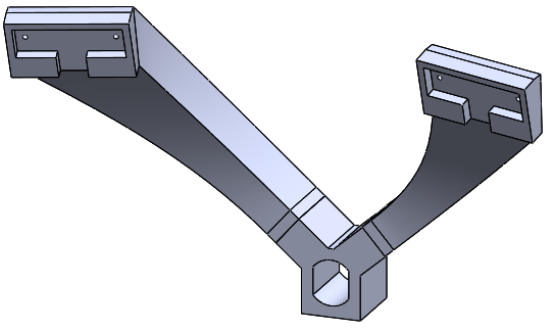


Figura 84: Versão final do suporte completo ao qual se prenderam os sensores de distância Sharp 2D120X

Antes do fabrico desta peça, fez-se outra, mais simples, que suportava apenas um sensor colocado diretamente por cima do centro do encosto da cabeça, com um declive menor, e com uma reentrância para se esconderem os fios elétricos do sensor. Esta peça, representada na Figura 85, servia o seu propósito e permitia detetar e garantir a presença da cabeça do utilizador numa área de segurança definida, no entanto, essa área, não era tão grande como a desejada, complicando um pouco os movimentos da pessoa, uma vez que o espaço em que ela se podia mover era mais limitado.

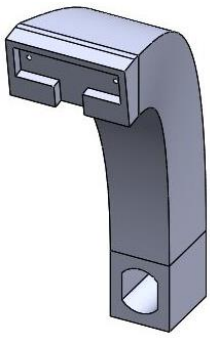


Figura 85: Primeira versão do suporte completo no qual se prendia um sensor de distância Sharp 2D120X

Por fim, desenhou-se uma última peça que consiste apenas numa pega para a tampa da caixa das baterias (Figura 86).

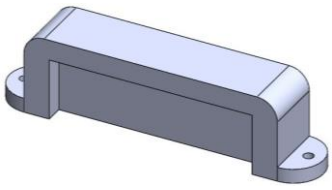


Figura 86: Peça utilizada como pega da tampa da caixa das baterias

5. Ensaios e resultados experimentais

Neste capítulo encontram-se descritos os ensaios realizados a fim de verificar e validar as tecnologias utilizadas bem como o resultado final de todo o trabalho implementado na cadeira de rodas.

5.1. Ensaios experimentais de validação de *hardware* e *software*

Foram realizados ensaios experimentais aos vários módulos desenvolvidos no decorrer da robotização da cadeira para garantir o seu correto funcionamento, tais como:

- Validação do controlo dos motores através do módulo Roboteq HDC2450. Esta validação fez-se com recurso ao *software* Roborun+ enviando diretamente comandos simples de potência por exemplo, rodar as rodas em separado e em simultâneo para averiguar o alcance da velocidade em ambos os sentidos e o uso do comando de emergência. Durante estes testes constatou-se o facto de um dos motores, para os mesmos valores de comandos de velocidade, proporcionar mais rotações por minuto que o outro motor. Estes ensaios também se fizeram com o uso do *joystick*.

- Integração de um *joystick* analógico. Para tratamento dos dados foi utilizado o algoritmo descrito no capítulo 4, tendo sido realizados ensaios para verificar a funcionalidade e linearidade dos valores recebidos.
- Comunicação série, I^2C e UDP – para todas estas comunicações foram desenvolvidos códigos de teste em separado do código principal onde foram ensaiados e validados envios e receções dos valores de orientação do referencial da cabeça e da cadeira bem como os comandos de velocidade para o Roboteq HDC2450, garantindo assim a correta comunicação entre os equipamentos correspondentes.
- Por último ensaiaram-se a navegação com todo o sistema da cadeira de rodas desenvolvido neste projeto. Os ensaios realizaram-se em espaços abertos e em espaços mais fechados e com mais obstáculos nos quais se testaram as interfaces e os sistemas de segurança implementados.

Os resultados dos sistemas de segurança e do sistema de interface HMU encontram-se detalhados ao longo deste capítulo, sendo que estes últimos foram descritos de forma qualitativa e com base num inquérito respondido por 7 indivíduos dos quais 4 não têm qualquer relação com o desenvolvimento deste projeto.

5.2. Módulos de *hardware* e peças 3D finais

De seguida, apresentam-se os vários componentes obtidos no final do desenvolvimento do projeto, tais como: a implementação das peças 3D concebidas, placas de circuito impresso, sistemas de segurança, comandos da cadeira das duas interfaces aplicadas e por fim, descrição de alguns problemas que surgiram relacionados com a própria estrutura da cadeira de rodas.

Placas de circuito impresso

No final do projeto foram implementadas 3 placas de circuito impresso concebidas para interligar os diferentes componentes eletrónicos descritos ao longo do presente relatório e as quais estão ilustradas na Figura 87, Figura 88 e Figura 89.

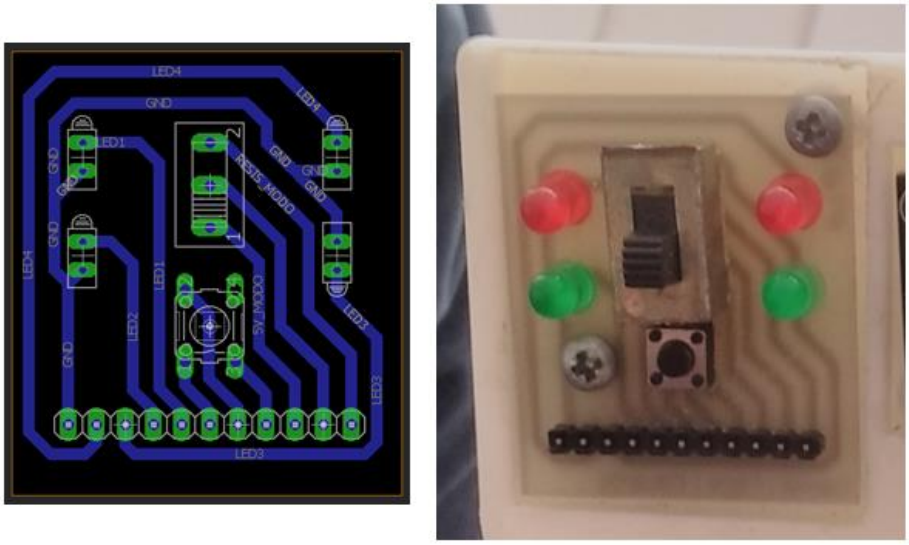


Figura 87: Esquema e resultado final da placa do painel de interação

Note-se a descrição do painel de interação encontra-se em anexo na tabela do **anexo IV**.

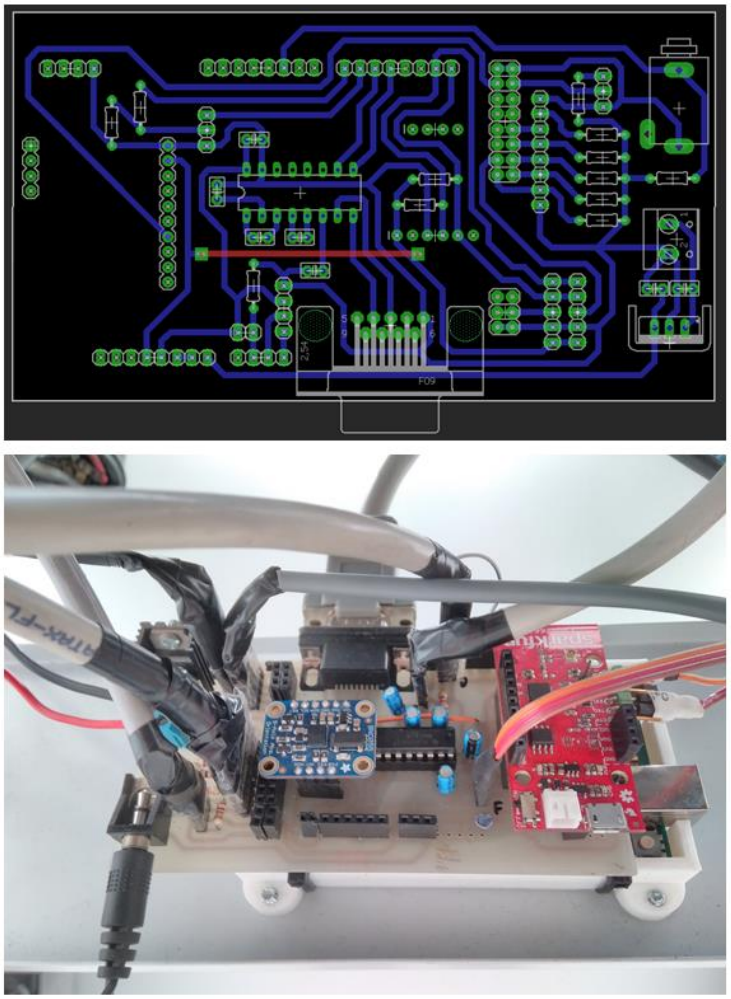


Figura 88: Esquema e resultado final da placa que assenta no Arduino e à qual se ligam todos os elementos da UPC

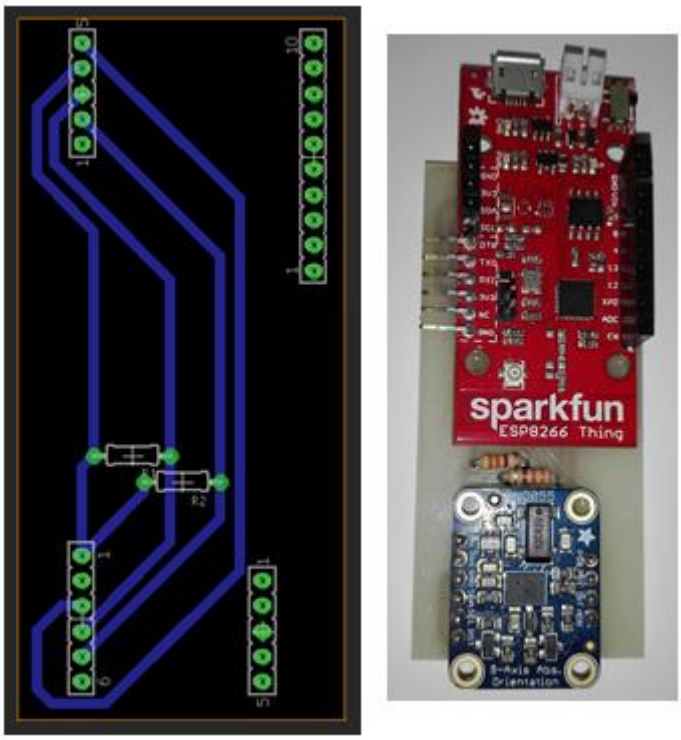


Figura 89: Esquema e resultado final da placa que possuí todos os elementos do Headset

Caixa de derivação

De modo a simplificar a resolução de problemas elétricos na alimentação (por exemplo, o recarregamento das baterias), foi utilizada uma caixa de derivação, acoplada à caixa das baterias. Esta contém o botão *On*/*Off* das fontes e 2 cápsulas, onde são colocados 2 fusíveis proteção, sendo um para o circuito da UPC e outro para o circuito do Roboteq (0,630 e 10A respetivamente), a alimentação destes mesmos circuitos sai da caixa de derivação por intermédio de fichas de conexão, deste modo basta desconectar estas alimentações sempre que se precisa de retirar a caixa das baterias e assim não é preciso desfazer as ligações que a compõem. O resultado da caixa de derivação encontra-se na Figura 90.

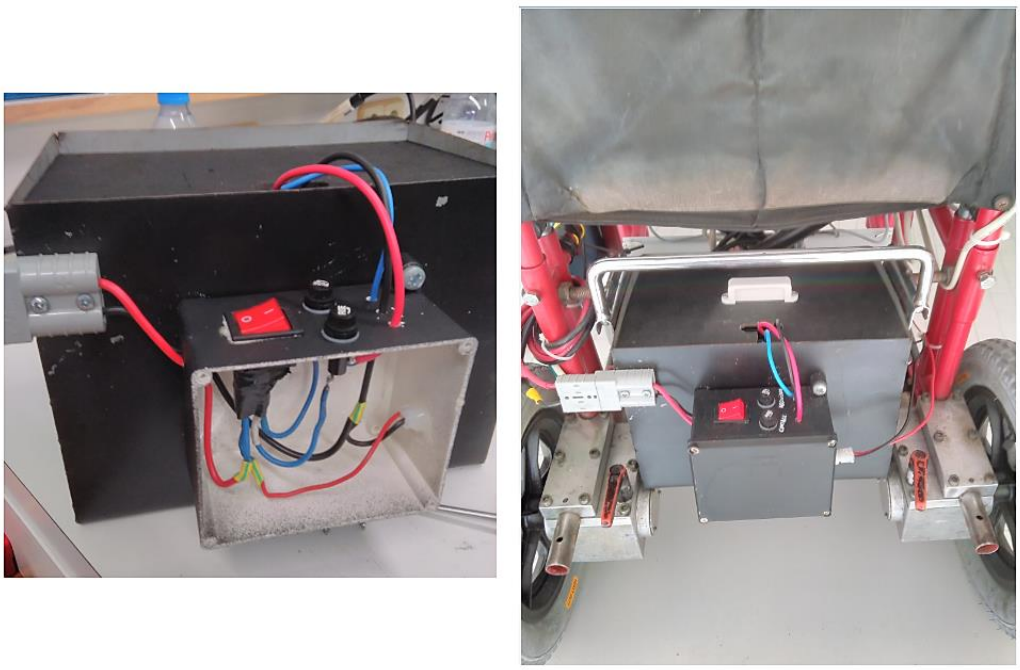


Figura 90: Resultado final e implementação de um caixa de derivação

Peças de impressão 3D

Para além da pega da tampa da caixa das baterias, que se encontra na Figura 90, todas as outras peças impressas em 3D cumpriram o propósito para o qual foram desenhadas. Seguem-se as fotografias dos resultados da implementação de todas as outras peças 3D (Figura 91, Figura 92, Figura 93 e Figura 94).

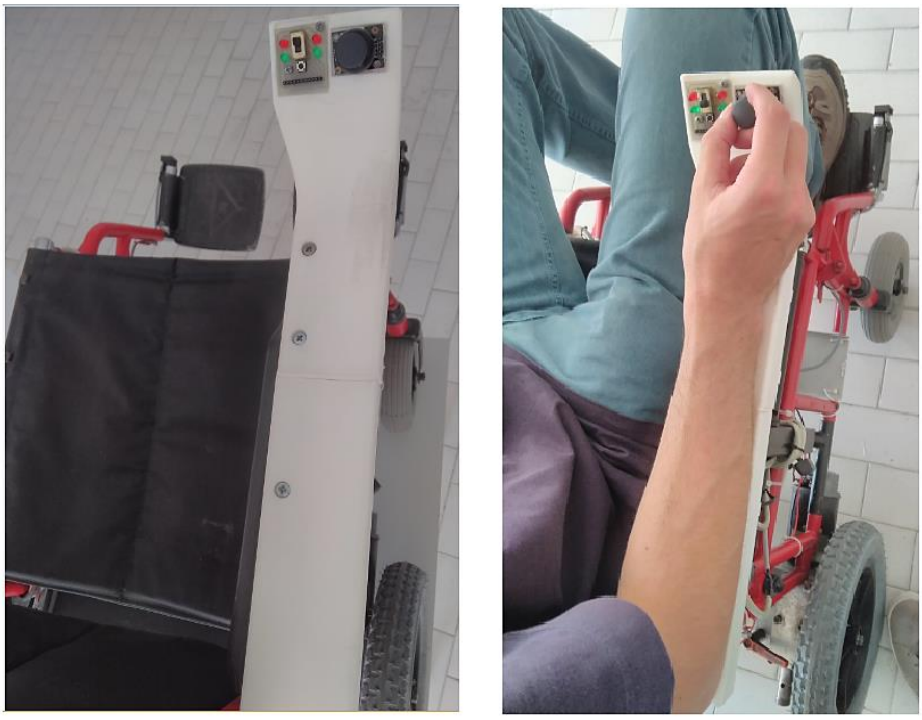


Figura 91: Resultado final e implementação do suporte de apoio ao braço, ao joystick e ao painel de interação

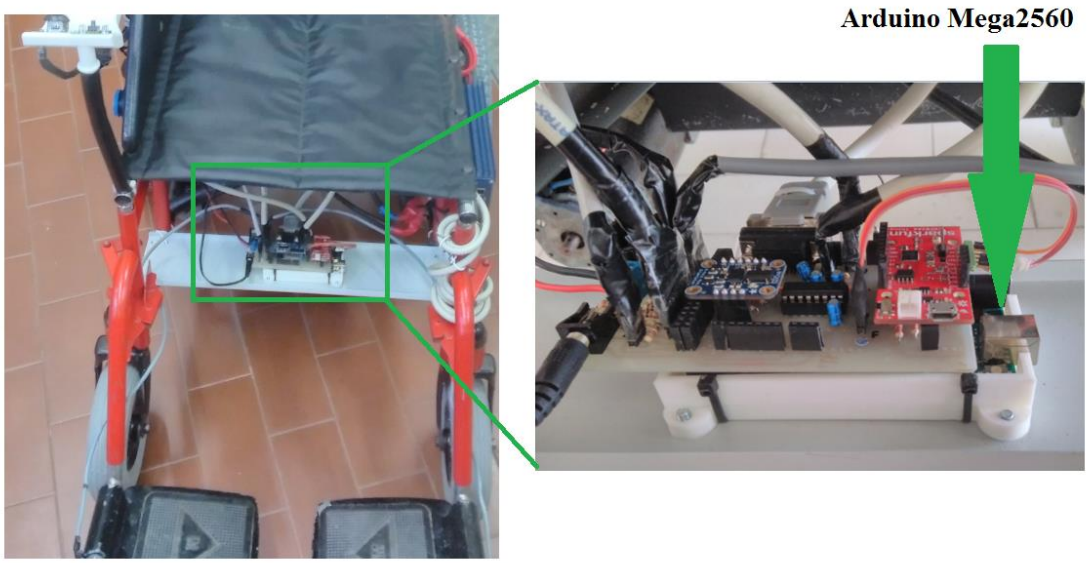


Figura 92: Resultado final e implementação da peça que suporta o Arduino e no qual assenta a placa que contém todos os elementos da UPC

Figura 93: Implementação do suporte dos sensores de distância do sistema que deteta a presença da cabeça

Figura 94: Resultado final do capacete em forma de bandelete no qual está inserido o Headset

5.3. Testes de condução e usabilidade

Sistemas de segurança:

Botão de emergência

O botão de emergência (Figura 95) foi implementado na lateral da cadeira, do lado do apoio de braço, deste modo é possível pressioná-lo rapidamente. Este sistema de segurança mostrou-se sempre eficaz.

Figura 95: Localização do botão de emergência e indicação por LEDs quando pressionado

Deteção de obstáculos

O sistema de segurança de deteção de obstáculos funcionou de acordo com as especificações dos sensores de distância e de acordo com o que foi implementado por *software*. O resultado não foi completamente satisfatório pelas seguintes razões:

- A UPC inicia a paragem da cadeira quando esta se encontra a 50cm ou menos de um obstáculo, o que em espaços mais fechados impede constantemente o controlo da cadeira através do HMU. A diminuição desta distância requer travagens mais bruscas e os 50cm foram definidos devido ao receio de danificar o controlador de potência quando se fazem travagens de emergência com valores de desaceleração ligeiramente mais elevados do que os normais (fator regenerativo dos motores), uma vez que se o fez em dois testes e que em ambos o fusível de proteção acabou queimado.
- Apenas se implementaram dois sensores VL53L0X na exterminada dianteira da cadeira e apontados para a frente. Embora estes sensores tenham um campo de visão de 25º não foram suficientes para englobar todos os movimentos possíveis da cadeira, o que se traduz em falhas da deteção de obstáculos na maioria dos movimentos com caracter rotacional.

  A Figura 96, ilustra as situações para as quais este sistema foi bem-sucedido.

Figura 96: Exemplo do sistema de deteção de obstáculos (com indicação por LED) a funcionar para os casos em que à frente da cadeira se encontra um obstáculo: chegado à direta, chegado à esquerda e em todo o plano

Deteção de área de segurança da cabeça

De seguida apresenta-se a implementação do sistema de deteção da presença da cabeça do utilizador na área definida como segura. Tal como referido ao longo do capítulo 4, este sistema foi o que mais sofreu alterações, sendo que o resultado final ficou muito perto do objetivo.

Com este sistema, a UPC para a cadeira sempre que a presença da cabeça do utilizador não é garantida e consegue detetá-la facilmente durante a execução dos movimentos de cabeça que se traduzem em movimentos de cadeira puramente rotacionais. O mesmo não se pode constatar para movimentos de cabeça que requerem inclinações para a frente e que se encontrem perto do limite possível aceite pelo HMU tendo em conta as especificações dos sensores de distância Sharp 2D120X. Surgiram então imensas situações em que este sistema induzia paragens da cadeira em condições aparentemente normais devido ao facto de:

- Os sensores Sharp 2D120X quando não detetam nenhum obstáculo no espaço de 30cm (alcance máximo), estes fornecem valores instáveis que rondam os 20cm o que implicou definir uma distância máxima entre a cabeça e a o encosto de 15cm.

- Ser difícil para o utilizador manter a cabeça perto do encosto (dentro dos 10cm referidos) ao mesmo tempo que a inclina para a frente.

Embora com dificuldades nos movimentos descritos, foi possível fazer o controlo da cadeira do HMU com este sistema implementado e os casos de sucesso estão ilustrados na Figura 97.

Figura 97: Exemplo do sistema de deteção da presença da cabeça do utilizador na área de segurança com a indicação por LEDs (todos a piscar) quando a presença da cabeça não é garantida

Deteção de falhas nos módulos *Wi-Fi*

À semelhança do sistema de segurança que contém o botão de emergência, o sistema que deteta as falhas dos módulos *Wi-Fi* referidas no capítulo 4, funcionou sempre como esperado tal como ilustrado na Figura 98.

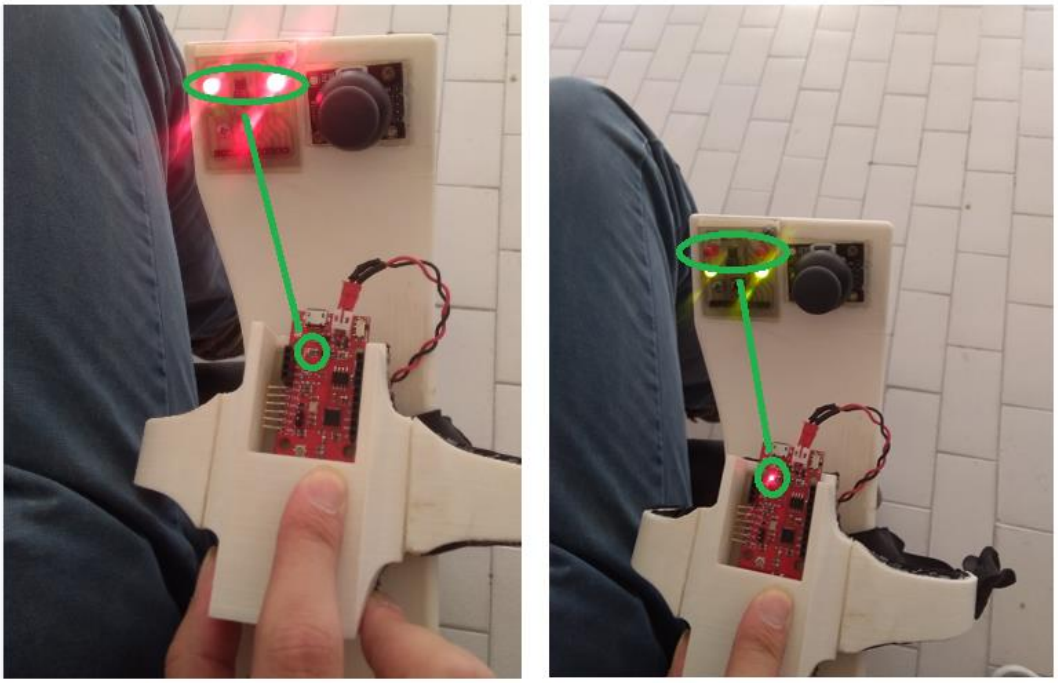


Figura 98:Exemplo do sistema de segurança que deteta falhas relacionadas com a comunicação Wi-Fi com indicação por LEDs: Módulo Wi-Fi do Headset desligado HMU parou a cadeira (LEDs vermelhos a piscar); Comunicação estabelecida (LEDs vermelhos desligados).

Testes de condução e utilização da cadeira com participantes

Por fim, foi pedido a sete pessoas, quatro das quais externas ao projeto, que comparassem o funcionamento entre o controlo desenvolvido e o de uma cadeira Salsa R2, existente no laboratório VITA, que apresenta uma solução disponível no mercado para controlo da cadeira de rodas com movimentos da cabeça. Após a utilização de ambas as cadeiras, foi pedido aos intervenientes que respondessem a um questionário, de forma a se poder obter uma perceção do grau de funcionalidade e usabilidade do protótipo desenvolvido em comparação com uma solução existente.

Foi pedido aos participantes que realizassem um conjunto de tarefas:

- Experimentar em primeiro lugar, a cadeira com o sistema comercializado;
- Aceder aos painéis interativos para escolher o modo de controlo por cabeça;
- Movimentar a cadeira em espaço aberto e livre e sem nenhum percurso definido;
- Movimentar a cadeira de modo a contornar duas mesas fazendo o formato de um 8;
- Passar para a cadeira de rodas desenvolvida neste projeto e repetir os procedimentos anteriores.

E respondessem ao inquérito da Tabela 2:

Tabela 2: Inquérito dos testes realizados com o sistema comercial e com o Wheelchair HMU

Perguntas
Classificação de 1 a 5 (1 - muito bom / 5 - muito mau)
Como avalia a facilidade na realização dos movimentos da cadeira?
Como avalia a naturalidade dos movimentos da cabeça?
Como avalia o tempo de resposta do movimento da cadeira em relação ao da cabeça?
Como avalia a liberdade dos movimentos da cadeira?
Como avalia os sistemas de segurança durante o movimento da cadeira?
Como avalia a informação apresentada e a interação com o painel de controlo?

Os resultados obtidos encontram-se na Tabela 3.

Tabela 3: Respostas dos participantes ao inquérito sobre os testes realizados.

Participante Nº 1		**Participante Nº 2**		**Participante Nº 3**		**Participante Nº 4**		**Participante Nº 5**		**Participante Nº 6**		**Participante Nº 7**	
W HMU	Salsa R2	W HMU	Salsa R2	W HMU	Salsa R2	W HMU	Salsa R2	W HMU	Salsa R2	W HMU	Salsa R2	W HMU	Salsa R2
3	1	3	1	3	4	2	4	2	3	3	2	3	1
2	2	3	2	3	3	1	2	1	3	2	3	2	2
2	1	4	1	2	4	2	3	2	3	2	2	3	1
2	2	2	2	2	3	3	2	2	3	2	3	3	1
1	3	2	3	3	2	1	1	2	1	1	1	1	3
1	3	1	2	2	3	2	3	1	3	2	1	2	2

Da análise dos resultados obtidos da comparação dos dois sistemas, retiraram-se as seguintes conclusões:

- Facilidade dos movimentos – Ambas as cadeiras apresentaram uma certa facilidade na realização de todos os movimentos, realizando o percurso de testes definido com problemas mínimos;
- Naturalidade dos movimentos – Todos os utilizadores indicaram que era necessária uma certa habituação aos movimentos requeridos para realizar o deslocamento da cadeira. A solução desenvolvida apresentou resultados consideravelmente melhores quando comparada com o sistema comercial;
- Tempo de resposta dos movimentos – A solução desenvolvida apresentou em grande parte dos testes um tempo de resposta muito superior à comercial, que foi considerada bastante lenta a reagir, no entanto, num dos testes, a solução desenvolvida apresentou alguma dificuldade em começar os movimentos;
- Liberdade dos movimentos – Apesar das dificuldades na realização de alguns movimentos, todos os utilizadores indicaram que a solução comercial é a que apresenta a maior liberdade de movimentos ao proporcionar um que não se encontra disponível na solução desenvolvida, o movimento de marcha atrás;
- Segurança proporcionada – Nesta categoria, os resultados obtidos foram bastante satisfatórios, pois todos os utilizadores consideraram que os sistemas de segurança implementados na solução desenvolvida realizavam bem as suas tarefas. Não obstante, a solução comercial considerou-se igualmente segura, apresentando problemas mínimos apenas em situações de exceção.
- Painel de controlo – Em ambas as cadeiras se concluíram que os respetivos painéis de controlo apresentavam a informação devida e necessária e que a interação com os mesmos era realizada bastante facilmente.

6. Conclusão

O projeto proposto consistiu no desenvolvimento do controlo de uma cadeira de rodas através dos movimentos da cabeça do seu utilizador, utilizando como componente base o sensor inercial BNO055. A abordagem tomada em relação a este projeto foi de tentar alcançar o maior nível de funcionalidade, segurança e usabilidade possíveis, pelo que tivemos de aplicar o máximo de rigor e seriedade, com o propósito de alcançar a meta estabelecida.

Os objetivos definidos foram cumpridos e ultrapassados, sendo que para a realização deste projeto, foi construído e implementado o sistema de controlo pretendido bem como um modo secundário em que os movimentos da cadeira de rodas são ditados por um *joystick*. Foram ainda adicionados vários sistemas de segurança e construídos os seus respetivos suportes físicos. Estes últimos tinham também o objetivo de melhorar a estética, a ergonomia e o funcionamento da cadeira.

Os resultados experimentais obtidos através dos testes realizados por intervenientes externos ao projeto mostram que o sistema proporciona um nível de funcionalidade suficientemente bom, permitindo concluir que o tipo sistema apresentado neste projeto deve continuar a ser explorado e desenvolvido, pois tem potencial para ser melhor que os sistemas atualmente existentes no mercado

Apesar do cumprimento dos objetivos propostos, existem vários aspetos a melhorar, dos quais se destacam:

- Escolha de níveis de velocidade, de modo a ajustar a cadeira a pessoas co diversos níveis de incapacidade motora;
- Possibilidade de marcha atrás efetuada por movimentos de cabeça, de maneira a ser possível sair de uma situação em que a cadeira esteja bloqueada por deteção de um obstáculo sem ter que mudar o modo de controlo;

- Implementação de um controlo dos motores em malha fechada recorrendo, por exemplo, a *encoders*;
- Substituição do *joystick* por um mais preciso;
- Desenvolvimento de um módulo de recarregamento das baterias em que não seja preciso retirar as mesmas da estrutura da cadeira;
- Otimização do código.

7. Referências bibliográficas

[1] Permobil, "TOTAL CONTROL HEAD ARRAY SYSTEM," [Online]. Available: http://countries.permobil.com/Global/head_array/Total%20Control%20Head%20Array%20System,%20Product%20Overview_EU%20version,%20NEW.pdf. [Acedido em 20 10 2018].

[2] "Products / 100 Series / ASL 105," asl, [Online]. Available: http://www.asl-inc.com/products/product_detail.php?prod=103#thumb. [Acedido em 19 10 2018].

[3] A. Pajkanović e B. Dokić, "Wheelchair Control by Head Motion," *SERBIAN JOURNAL OF ELECTRICAL ENGINEERING,* vol. 10, nº 1, pp. 135-151, 2013.

[4] C. Show-Hong, C. Yu-Luen, K. Te-Son, C. Chiung-Yu e H. Chao-Nan, "M3S-based Electrical Wheelchair with Head-controlled Device," em *Proceedings of the 28th IEEE EMBS Annual International Conference*, New York, NY, USA , 2006.

[5] D. J. Kupetz, S. A. Wentzell e B. F. BuSha, "Head Motion Controlled Power Wheelchair," em *Proceedings of the 2010 IEEE 36th Annual Northeast Bioengineering Conference (NEBEC)*, New York, NY, USA, 2010.

[6] E. J. Rechy-Ramirez, H. Huosheng e K. McDonald-Maier, "Head movements based control of an intelligent wheelchair in an indoor environment," em *Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics*, Guangzhou, China, 2012.

[7] F. A. Kondori, S. Yousefi, L. Liu e H. Li, "HEAD OPERATED ELECTRIC WHEELCHAIR," em *2014 Southwest Symposium on Image Analysis and Interpretation*, San Diego, CA, USA , 2014.

[8] I. O. Qamar, B. A. Fadli, G. A. Sukkar e M. Abdalla, "Head Movement Based Control System for Quadriplegia Patients," em *2017 10th Jordanian International Electrical and Electronics Engineering Conference (JIEEEC)*, Amman, Jordan, 2017.

[9] S. Prasad, D. Sakpal e S. Rawool, "Head-Motion Controlled Wheelchair," em *2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT)*, Bangalore, India, 2017.

[10] Y. Ikushi, S. Katsuhiko e I. Takenobu, "Development of Head Gesture Interface for Electric Wheelchair," em *Proceedings of the 1st international convention on Rehabilitation engineering & assistive technology: in conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting*, Singapura, Singapura, 2007.

[11] Millmore, "Head Mouse - Game Controller or Disability Aid," [Online]. Available: https://www.instructables.com/id/Head-Mouse-Game-controller-or-disability-aid/. [Acedido em 19 10 2018].

[12] Adafruit, "Adafruit BNO055 Absolute Orientation Sensor," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf. [Acedido em 22 10 2018].

[13] Bosh, "Intelligent 9-axis absolute orientation sensor," [Online]. Available: https://pt.mouser.com/pdfdocs/BST_BNO055_DS000_14.pdf. [Acedido em 22 10 2018].

[14] st, "World's smallest Time-of-Flight ranging and gesture detection sensor," [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l0x.pdf. [Acedido em 22 10 2018].

[15] Adafruit, "Adafruit VL53L0X Time of Flight Micro-LIDAR Distance Sensor," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout.pdf. [Acedido em 22 10 2018].

[16] Sharp, "GP2D120 Optoelectronic Device," [Online]. Available: https://www.pololu.com/file/0J157/GP2D120-DATA-SHEET.pdf SHARP. [Acedido em 22 10 2018].

[17] Sparkfun, "ESP8266 Thing (WRL-13231)," [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP8266ThingV1.pdf. [Acedido em 22 10 2018].

[18] "Arduino Mega 2560 Rev3," arduino, [Online]. Available: https://store.arduino.cc/arduino-mega-2560-rev3. [Acedido em 15 10 2018].

[19] "8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash," microchip, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. [Acedido em 15 10 2018].

[20] "2x150A or 1x300A High Performance Dual Channel Brushed DC Motor Controller with and CAN Interface," Roboteq, [Online]. Available: https://www.roboteq.com/index.php/docman/motor-controllers-documents-and-files/documentation/datasheets/hdc24xx-datasheet/60-hdc24xx-datasheet/file. [Acedido em 13 10 2018].

[21] "Understanding How a Voltage Regulator Works," analog, [Online]. Available: http://www.analog.com/media/en/technical-documentation/tech-articles/Understanding-How-a-Voltage-Regulator-Works.pdf. [Acedido em 15 10 2018].

[22] "I2C," Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/i2c. [Acedido em 13 10 2018].

[23] "I2C-bus specification and user manual," nxp, [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. [Acedido em 13 10 2018].

[24] "Understanding the I2C Bus," TI, [Online]. Available: http://www.ti.com/lit/an/slva704/slva704.pdf. [Acedido em 13 10 2018].

[25] "UDP – User Datagram Protocol," IPV6, [Online]. Available: https://www.ipv6.com/general/udp-user-datagram-protocol/. [Acedido em 13 10 2018].

[26] "Using the USART in Asynchronous Mode," microchip, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/USART.pdf. [Acedido em 13 10 2018].

[27] "Basics of UART Communication," circuitbasics, [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/. [Acedido em 13 10 2018].

[28] "ST10 UART recommendations," digchip, [Online]. Available: https://application-notes.digchip.com/005/5-10195.pdf. [Acedido em 13 10 2018].

[29] sparkfun, [Online]. Available: https://cdn.sparkfun.com/assets/2/5/c/4/5/50e1ce8bce395fb62b000000.png. [Acedido em 13 10 2018].

[30] "How RS232 works," best-microcontroller-projects, [Online]. Available: https://www.best-microcontroller-projects.com/how-rs232-works.html. [Acedido em 15 10 2018].

[31] "Soft Access Point Class," github, [Online]. Available: https://github.com/esp8266/Arduino/blob/master/doc/esp8266wifi/soft-access-point-class.rst. [Acedido em 15 10 2018].

[32] "VR2 - Drive-Only (60-90A)," cw, [Online]. Available: http://www.cw-industrialgroup.com/Products/Mobility-Vehicle-Solutions/VR2/Drive-Only-System. [Acedido em 15 10 2018].

[33] “Joystick Arduino 3 Eixos,” filipeflop, [Online]. Available: https://www.filipeflop.com/produto/joystick-arduino-3-eixos/ . [Acedido em 15 10 2018].

[34] “Advanced Brushed and Brushless Digital Motor Controllers,” roboteq, [Online]. Available: https://www.roboteq.com/index.php/docman/motor-controllers-documents-and-files/documentation/user-manual/272-roboteq-controllers-user-manual-v17/file. [Acedido em 15 10 2018].

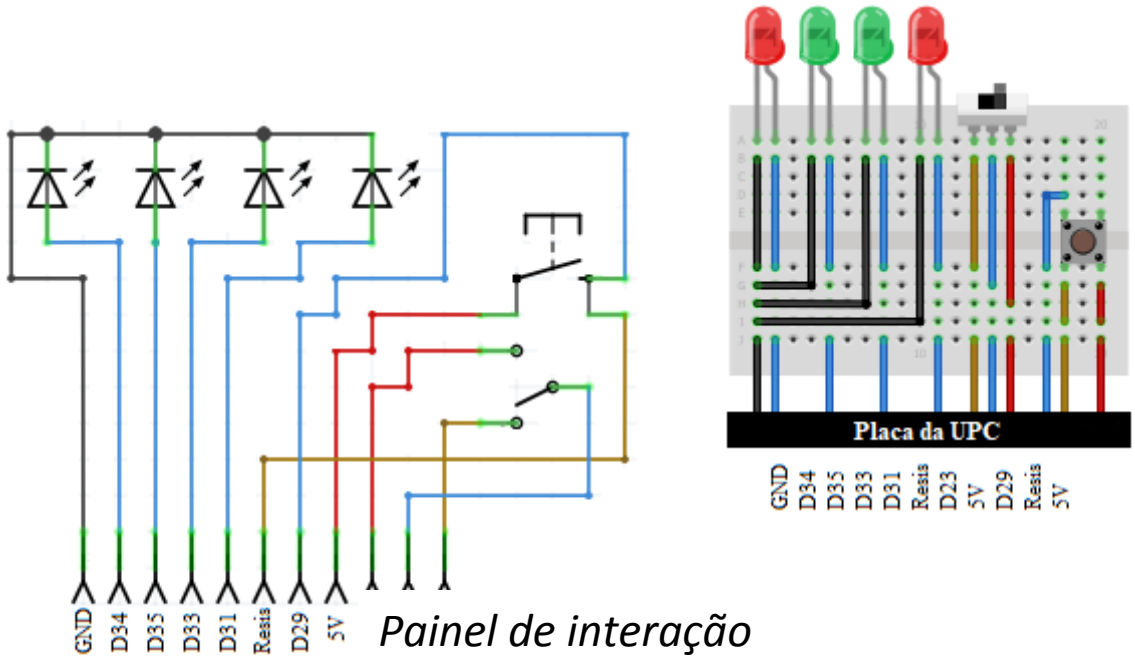
[35] “SoftPot Membrane Potentiometer - 100mm,” sparkfun, [Online]. Available: https://www.sparkfun.com/products/retired/8607. [Acedido em 15 10 2018].

[36] “Starting Robotics: Haptic Controlled Robotic Claw,” bayesianadventure, [Online]. Available: https://bayesianadventures.wordpress.com/2013/09/13/starting-robotics-haptic-controlled-robotic-claw/. [Acedido em 15 10 2018].

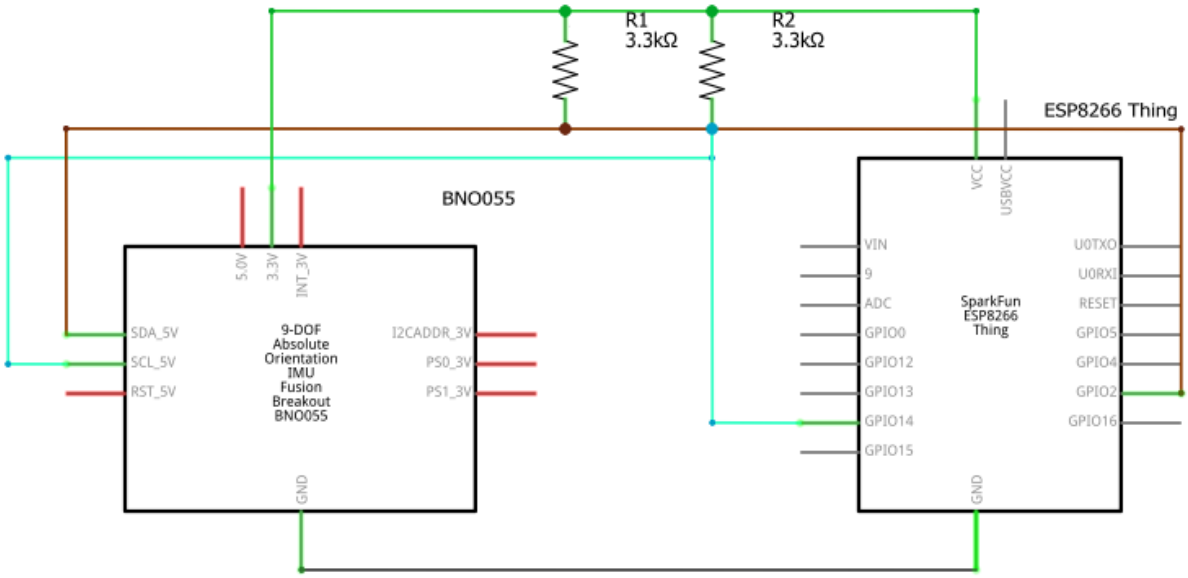
[37] “Positive voltage regulator ICs,” st, [Online]. Available: https://www.st.com/resource/en/datasheet/l78.pdf. [Acedido em 15 10 2018].

[38] “Linear Regulator Design Guide For LDOs,” ti, [Online]. Available: http://www.ti.com/lit/an/slva118a/slva118a.pdf. [Acedido em 15 10 2018].

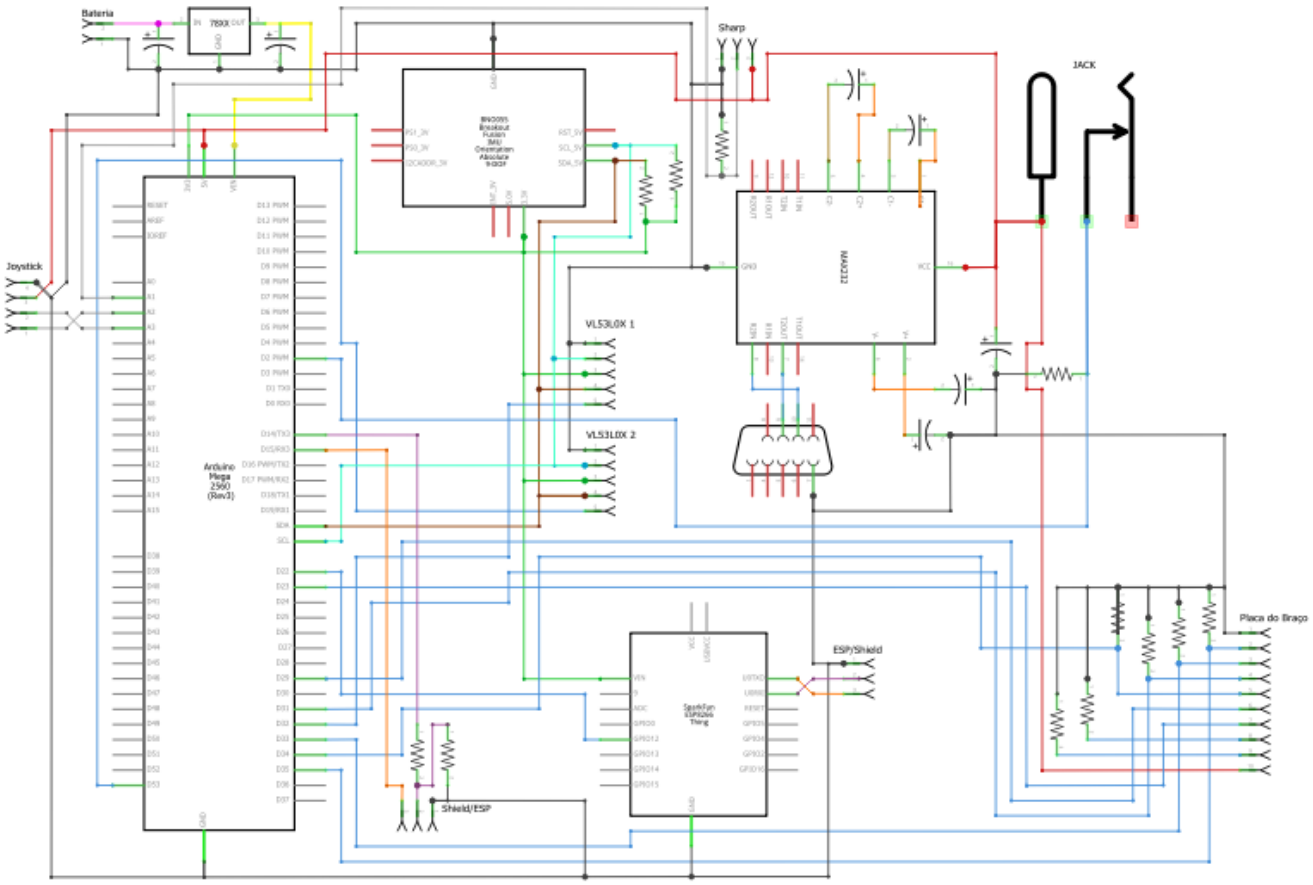
Anexo I –*Software* Fritzing – esquema da PCB do painel de interação e esquematicos das PCBs da UPC, do Headset e do painel de interação.



Painel de interação

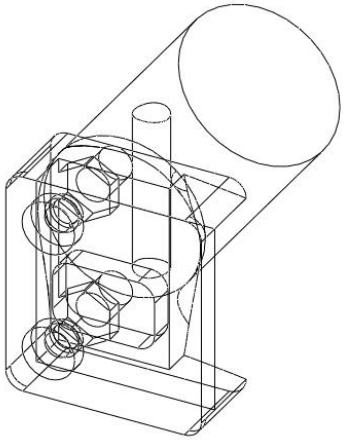


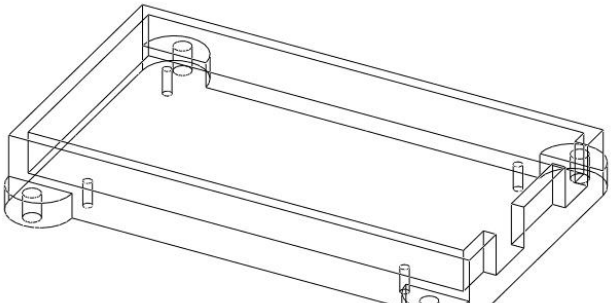
Headset

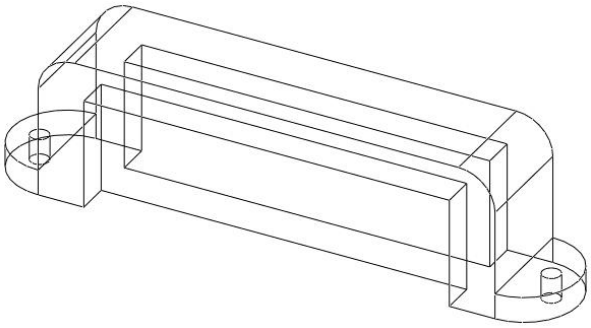

Bateria
Sharp
JACK
Joystick
VL53L0X 1
VL53L0X 2
Arduino Mega 2560 (Rev3)
ESP/Shield
Shield/ESP
Placa do Braço

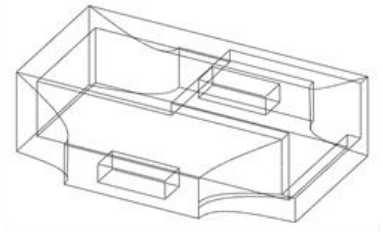
UPC

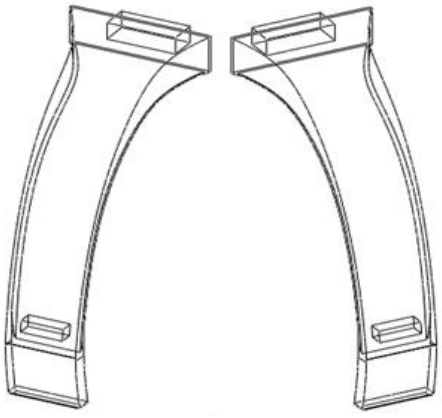
Anexo II – Esquemas do software Solidworks – Peças Finais com contornos

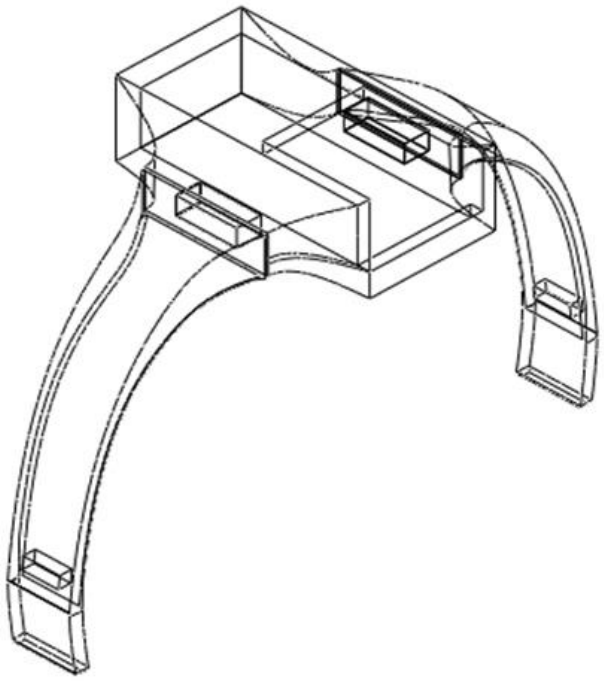


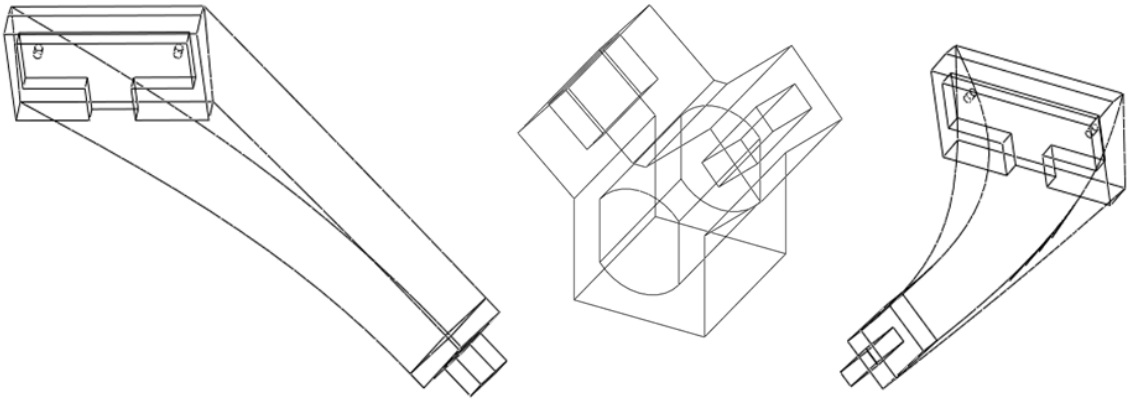


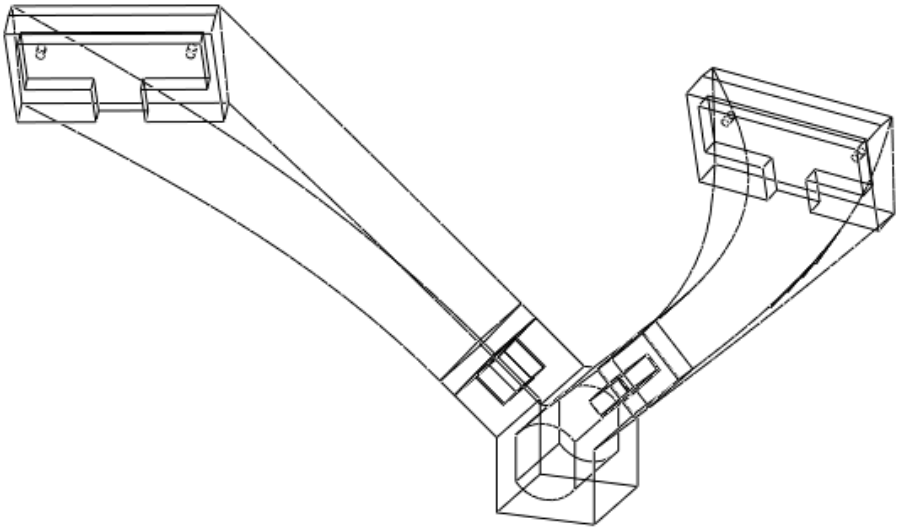




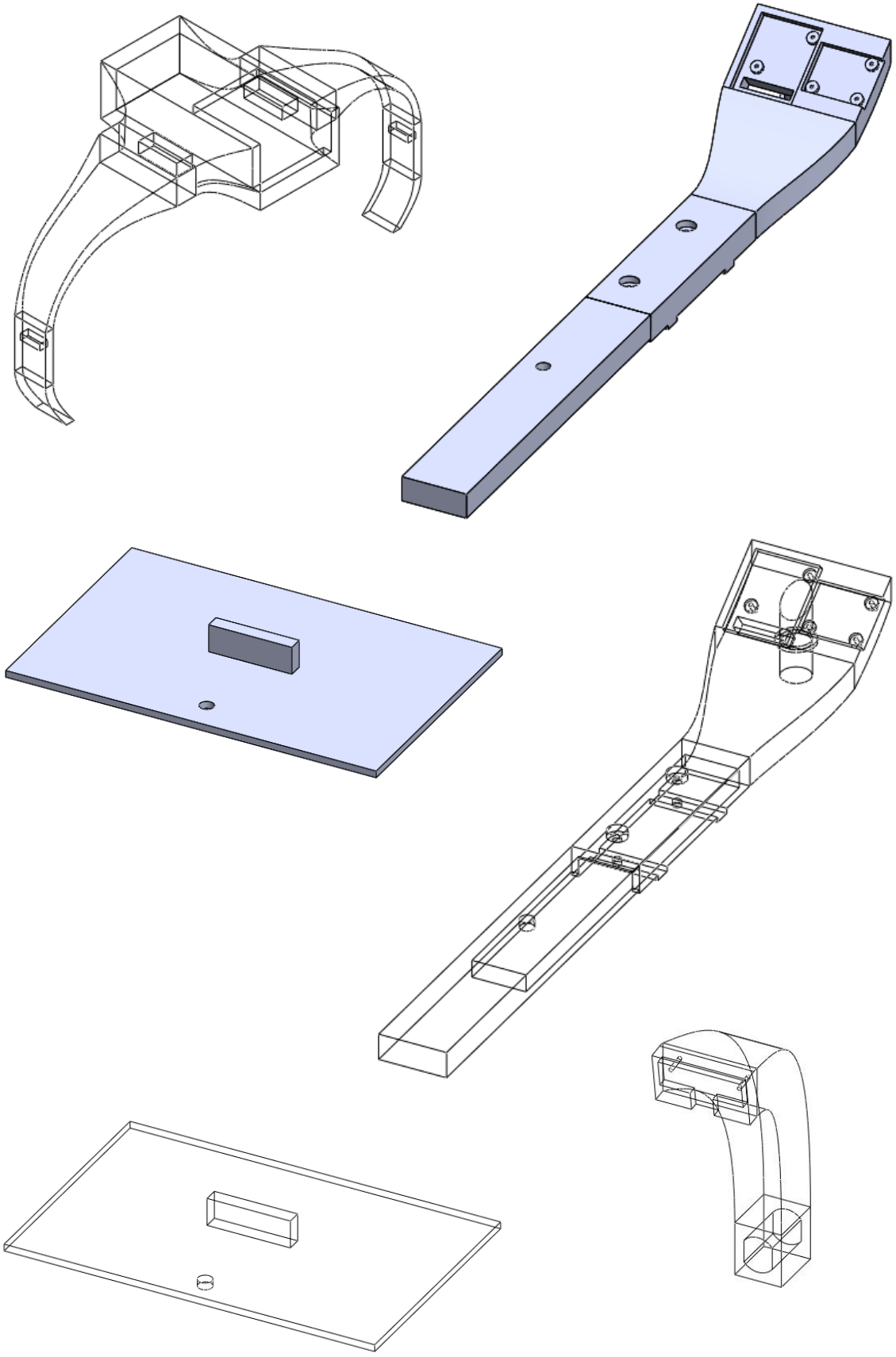




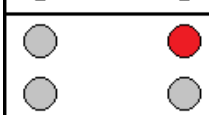
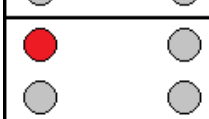
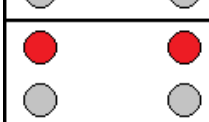




Anexo III – Esquemas do software Solidworks – Peças Rejeitadas

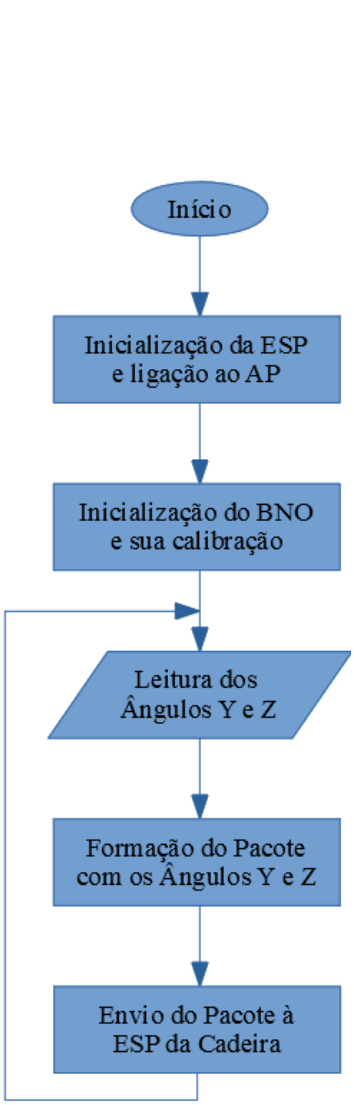


Anexo IV – Tabela de possíveis problemas, apresentados pelo painel informativo

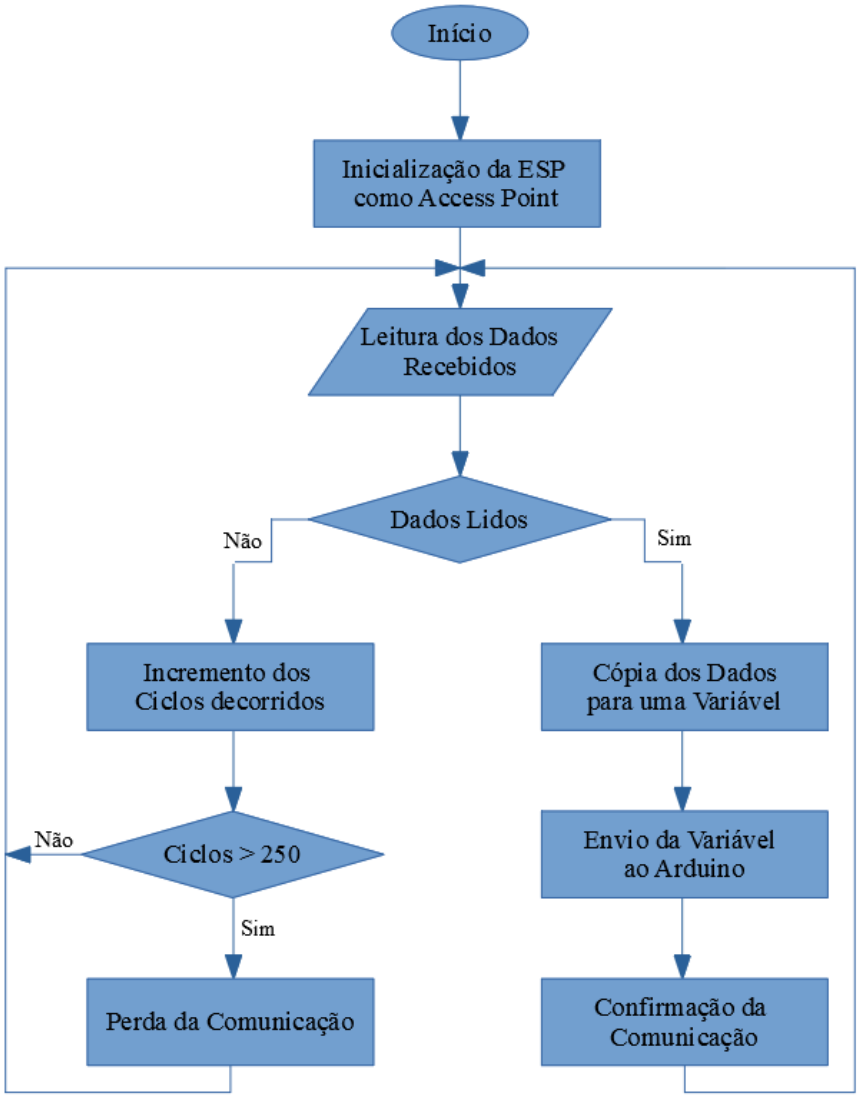
Problemas		
	**S**	O controlador encontra-se bloqueado.
	C	Bloqueou-se o controlador ao pressionar-se o botão de emergência.
	R	Pressionar o botão de emergência durante pelo menos dois segundos.
	**S**	A presença da cabeça não está garantida.
	C	A cabeça encontra-se a uma distância superior à necessária.
	R	Aproxima-se a cabeça ao encosto da cadeira.
	**S**	Não existe comunicação WiFi.
	C	A ESP da cabeça está desligada.
	R	Liga-se a ESP da cabeça
	**S**	O alinhamento não foi feito.
	C	Os ângulos do capacete e da cadeira não estão alinhados.
	R	Pressione o botão de alinhamento.
	**S**	O lado direito da cadeira não pode avançar.
	C	Encontra-se algo a menos de trinta centímetros do lado direito da cadeira.
	R	Endireite a cadeira para o lado esquerdo.
	**S**	O lado esquerdo da cadeira não pode avançar.
	C	Encontra-se algo a menos de trinta centímetros do lado esquerdo da cadeira.
	R	Endireite a cadeira para o lado direito.
	**S**	A cadeira não pode avançar.
	C	Encontra-se algo a menos de trinta centímetros da cadeira.
	R	Muda-se o modo de funcionamento para o Joystick.

Legenda	
	Os LEDs estão a piscar.
	Os LEDs estão acesos.
S	Significado
C	Causa
R	Resolução

Anexo V – Fluxogramas do funcionamento das ESP8266-*Thing* – *Headset e UPC*



Headset



UPC