Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Sistema de Tradução da Língua Gestual Portuguesa em Tempo Real

Sara Cristina Martins Ferreira

Dissertação submetida como requisito parcial para obtenção do grau de mestre em

Engenharia de Telecomunicações e Informática

Orientador

Professor Doutor Nuno Souto, Professor Auxiliar

ISCTE-IUL

Co-Orientador

Professor Doutor Octavian Postolache, Professor Auxiliar

ISCTE-IUL

Setembro 2019

"Sign language is a beautiful language."

Jackie Knight

Resumo

O avanço na área da saúde só é possível graças ao avanço da tecnologia. Novos conceitos e sistemas permitem melhorar a qualidade de vida das pessoas. Os sistemas de reconhecimento de gestos são um dos campos procedentes do avanço da tecnologia. Não só tem aplicação na área da saúde mas também em áreas como a realidade virtual, industria automóvel, e outras tantas.

O objetivo deste trabalho era criar um sistema *wireless* e *low-cost* de reconhecimento de gestos, capaz de traduzir Língua Gestual Portuguesa para Língua Portuguesa. Este projeto teve um âmbito mais restrito, na medida em que se foca no alfabeto português.

O sistema é composto por uma rede de sensores implementada numa luva através de fios condutores e *snaps*. Ao ser utilizada, a luva faz a leitura da posição dos dedos assim como da direção da mão em tempo real. Esta informação é dada por sensores de flexão baseados em *velostat* e pelo acelerómetro, respetivamente. A informação recolhida pela luva, que caracteriza o gesto, é então enviada para a aplicação móvel pelo protocolo de comunicação *Bluetooth*. Este protocolo tem a vantagem de ser *wireless* e de não necessitar da conexão do dispositivo móvel à Internet para comunicar. A leitura desta informação em conjunto com o k-Nearest Neighbors, algoritmo de aprendizagem supervisionada que se baseia na similaridade dos dados que caracterizam o gesto feito em tempo real com os dados que classificam cada letra do alfabeto, permite ao sistema fazer a tradução de Língua Gestual Portuguesa para a Língua Portuguesa. Esta tradução é posteriormente visualizada numa aplicação móvel.

Com o sistema desenvolvido foi atingida uma taxa de acerto quando utilizado o k-Nearest Neighbors de 93.31% e 89.66%, e de 78.27% e 71.16% quando utilizadas as Redes Neuronais, para uma pessoa com conhecimento em Língua Gestual Portuguesa e para uma pessoa sem conhecimento em Língua Gestual Portuguesa, respetivamente.

Palavras-chave: Língua Gestual Portuguesa, Sistema *Wireless*, Reconhecimento de Gestos, Interação Humano-Computador, Protocolo de Comunicação, *Machine Learning, Android.*

Abstract

Advancement in healthcare is only possible thanks to the advancement of technology. New concepts and systems improve people's quality of life. Gesture recognition systems are one of the fields coming from the advancement of technology. Not only is it applicable in healthcare but also in areas such as virtual reality, the automotive industry, and many more.

The main goal was to create a wireless and low-cost gesture recognition system capable of translate Portuguese Sign Language to Portuguese Language. This project focuses on the Portuguese alphabet translation.

The system is composed by a sensor network implemented in a glove through conductive thread and snaps. When used, the glove has the function of reading the position of the fingers as well as the direction of the hand. This information is given by flex sensors based on velostat and accelerometer, respectively. The information collected by the glove, which characterizes the gesture, is then sent to the mobile application by the Bluetooth communication protocol. This protocol has the advantage of not require the connection of the mobile device to the Internet to communicate. The reading of this information together with k-NN, a supervised learning algorithm which is based on similarity of data that characterizes the gesture made in real time with data that classifies letters of the alphabet, allows the system to translate the Portuguese Sign Language to Portuguese Language. This translation is shown in a mobile application.

With the developed system a hit rate was reached when using the k-Nearest Neighbors of 93.31% and 89.66%, and 78.27% and 71.16% when used the Neural Networks, for a person with knowledge of Portuguese Sign Language and for a person without knowledge of Portuguese Sign Language, respectively.

Keywords: Portuguese Sign Language, Wireless System, Gesture Recognition, Human-Computer Interaction, Communication Protocol, Machine Learning, Android.

Agradecimentos

O sucesso desta dissertação não seria possível sem o constante apoio de algumas pessoas aos quais dirijo o meu especial obrigada.

Em primeiro lugar gostaria de agradecer ao meu orientador e co orientador, Professor Nuno Souto e Professor Octavian Postolache, pela constante disponibilidade e ajuda ao longo da dissertação. Sem o seu apoio a conclusão desta dissertação não seria possível.

Aos meus pais agradeço por todo o apoio dado ao longo destes 5 anos. Sem eles seria impossível realizar este sonho que tinha não só por mim mas também por eles.

Um especial obrigada ao meu namorado, Henrique Ferreira, que foi o meu maior apoio durante esta fase. Obrigada pela amizade, paciência, ajuda e coragem que me deste todos os dias.

Ao Instituto de Telecomunicação agradeço a ajuda financeira prestada na compra de material necessário a este projeto.

Conteúdo

Resumo **v**

Abstract **vii**

Agradecimentos **ix**

Lista de Figuras **xiii**

Abreviaturas **xviii**

1 Introdução **1**
1.1 Motivação e Enquadramento 1
1.2 Objetivos 2
1.3 Contribuições 2
1.4 Estrutura da Dissertação 3

2 Revisão da Literatura **5**
2.1 Língua Gestual 5
2.1.1 Língua Gestual Portuguesa 6
2.1.2 Parâmetros da Língua Gestual Portuguesa 6
2.2 Sistemas de Reconhecimento de Gestos 10
2.2.1 Sistemas baseados em Sensores de Vídeo 11
2.2.2 Sistemas baseados em Sensores de Movimento 14
2.3 Protocolos de Comunicação *Wireless* 16
2.3.1 *Bluetooth* 17
2.3.2 *Wi-Fi* 18
2.3.3 *ZigBee* 18
2.4 *Machine Learning* 18
2.5 *Hardware Wearable* 21
2.5.1 Tipos de sensores 22
2.6 Aplicação Móvel 24

3 Arquitetura do Sistema **27**
3.1 Coordenador 29
3.2 Parâmetros Recolhidos 34
3.2.1 Movimento dos dedos 34

3.2.2 Direção da mão 38
3.3 Comunicação 42
3.4 Linha Condutora e *Snaps* 46
3.5 Protótipo 46
3.6 Consumo do sistema 49

4 Classificação **51**
4.1 Dados recolhidos 51
4.2 Preparação dos dados 52
4.3 Normalização dos Dados 56
4.4 Algoritmos 60
4.4.1 Classificação final com base nos sensores de flexão 60
4.4.1.1 k-NN 60
4.4.1.2 Redes Neuronais 62
4.4.2 Classificação final com base no acelerómetro 65

5 Aplicação Móvel **67**
5.1 Aplicação *Android* 67
5.2 Estrutura da Aplicação Móvel 69
5.3 Design e Implementação da Aplicação Móvel 71

6 Resultados **75**
6.1 Testes desenvolvidos com a luva 75
6.2 Testes desenvolvidos com os algoritmos 79
6.2.1 Testes feitos aos dois algoritmos com os sensores de flexão . 80
6.2.2 Testes feitos aos dois algoritmos com o acelerómetro 82
6.2.3 Testes feitos aos dois algoritmos com os sensores de flexão e com o acelerómetro 84

7 Conclusões e Trabalho Futuro **89**
7.1 Conclusões 89
7.2 Trabalho Futuro 91

Anexos **95**

A Artigos Científicos Publicados **95**

Bibliografia **101**

Lista de Figuras

2.1 Alfabeto da LGP 7
2.2 Representação da palavra "sentar"na LGP 7
2.3 Letras do alfabeto da LGP 8
2.4 8 exemplos de movimentos 8
2.5 Pontos de articulação 9
2.6 Exemplos de expressões faciais 9
2.7 Fases do reconhecimento de gestos 10
2.8 Sistema baseado na câmara colocada na mesa 11
2.9 Sistema baseado na câmara colocada no chapéu 12
2.10 Componentes da Kinect 12
2.11 Disposição das coordenadas espaciais do Leap Motion 13
2.12 Comando Move da PlayStation 13
2.13 Localização da IMU no sistema 15
2.14 Sensor eletromagnético 15
2.15 Sistema composto por sensores de flexão 16
2.16 Etapas Aprendizagem Automática 19
2.17 Esquema das Redes Neuronais 21
2.18 Curvatura do sensor de flexão 22
2.19 Disposição dos sensores capacitivos na luva 23
2.20 Luva baseada no *velostat* e 5-Dof IMU 24
2.21 Tradução da LGP para texto 25
2.22 Tradução de texto para LGP 25
2.23 Tradução da Língua Gestual Americana para texto 26
2.24 Interface da Aplicação Móvel 26

3.1 Diagrama de Informação/Tradução 27
3.2 Arquitetura da luva 28
3.3 LilyPad Arduino Simple 30
3.4 LilyPad Arduino Main Board 31
3.5 LilyPad Arduino USB 31
3.6 LilyPad Arduino Simple Snap 32
3.7 Datasheet LilyPad USB 33
3.8 LilyPad cosido na luva 34
3.9 Fio condutor sobre a fita adesiva 35
3.10 Velostat sobre o fio condutor 35

3.11 Sensor de Flexão 36
3.12 Sensores de flexão cosidos na luva 36
3.13 Circuito do sensor de flexão 37
3.14 Acelerómetro LSM303 3D 38
3.15 3-eixos Acelerómetro LSM303 3D 38
3.16 Protocolo SPI VS I^2C 39
3.17 Direção das coordenadas X, Y e Z das letras do alfabeto da LGP . . 41
3.18 Acelerómetro cosido no dorso da mão 41
3.19 Conexão com o BLE 43
3.20 Bluetooth BTBee PRO 43
3.21 Bluetooth RN41 44
3.22 LilyPad Xbee 45
3.23 LilyPad Xbee cosido na parte da frente da mão 45
3.24 Código *Arduino* 45
3.25 Linha Condutora 46
3.26 Snaps 46
3.27 Luva sem *hardware* implementado 47
3.28 Luva com os componentes eletrónicos 47
3.29 Luva com os componentes eletrónicos 48
3.30 Esquema elétrico da luva 48
3.31 Bateria de litio 49
3.32 Análise do consumo do sistema 50

4.1 Ficheiro valores.txt 53
4.2 Média de cada fase de amostragem 54
4.3 Fluxograma Recolha de dados 55
4.4 Tratamento de dados k-nn 56
4.5 Tratamento de dados Redes Neuronais 56
4.6 Valores da letra B 57
4.7 Ficheiro txt 58
4.8 Valores mínimos e máximos de cada dedo 58
4.9 Valores normalizados 59
4.10 Fluxograma dados normalizados 60
4.11 Valores do gesto feito em tempo real 61
4.12 Fluxograma classificação com base nos sensores de flexão 62
4.13 Função Sigmóide 63
4.14 Tratamento de dados Redes Neuronais 64
4.15 Valores do gesto feito em tempo real 64
4.16 Letras devolvidas como tradução 65
4.17 Fluxograma classificação com base no acelerómetro 66

5.1 Percentagem de utilização de cada Sistema Operativo Móvel em Portugal 68
5.2 Versões Android mais utilizadas 69
5.3 Diagrama de atividade da aplicação móvel 70

5.4 Logótipo da aplicação móvel 71
5.5 *Bluetooth* do dispositivo móvel ligado 72
5.6 Conexão entre o dispositivo móvel e a luva 72
5.7 Gesto a traduzir 73
5.8 Botões da aplicação móvel 73
5.9 Ficheiro da aplicação móvel 74
5.10 Tradução do gesto feito em tempo real 74

6.1 Protótipo com o *Arduino UNO* 76
6.2 Palma da mão voltada para baixo 77
6.3 Palma da mão voltada para cima 77
6.4 Palma da mão voltada para a esquerda 78
6.5 Palma da mão voltada para a direita 78
6.6 Conexão do *Bluetooth* da luva à aplicação móvel 79
6.7 Letras do alfabeto da LGP 80

Lista de Tabelas

2.1 Características dos principais protocolos de comunicação sem fio . . 17

3.1 Características das Plataformas *Arduino Wereable* 30

3.2 Valores médios dos sensores de flexão 37

3.3 Sinais das coordenadas da letra M e W 40

3.4 Sinais das coordenadas da letra A e R 40

3.5 Sinais das coordenadas da letra B, Q e E 40

3.6 Sinais das coordenadas da letra K e V 40

3.7 Sinais das coordenadas da letra L e T 40

3.8 Sinais das coordenadas da letra U e X 40

3.9 Principais características do *Bluetooth RN41* 44

6.1 Valores para a letra A 76

6.2 Percentagem de acerto de cada letra 81

6.3 Percentagem de acerto de cada letra 82

6.4 Percentagem de acerto de cada letra 83

6.5 Percentagem de acerto de cada letra 83

6.6 Percentagem de acerto de cada letra 85

6.7 Percentagem de acerto de cada letra 86

Abreviaturas

LGP	**L**íngua **G**estual **P**ortuguesa
OMS	**O**rganização **M**undial **S**aúde
LGS	**L**íngua **G**estual **S**ueca
IMU	**I**nercial **M**easurement **U**nit
EMG	**E**letro**m**io**g**ráfico
BLE	**B**luetooth **L**ow **E**nergy
k-NN	**k**- **N**earest **N**eighbours
JST	**J**apan **S**olderless **T**erminal
I^2C	**I**nter-**I**ntegrated **C**ircuit
SPI	**S**erial **P**eripheral **I**nterface
SPP	**S**erial **P**ort **P**rofile
IDE	**I**ntegrated **D**evelopment **E**nvironment
GUI	**G**raphical **U**ser **I**nterface

Capítulo 1

Introdução

1.1 Motivação e Enquadramento

Atualmente, a tecnologia simplifica e aumenta a qualidade dos serviços prestados e apesar do seu avanço constante, as condições providenciadas à comunidade surda estão longe de ser as melhores, uma vez que a forma mais usada de comunicação entre uma pessoa surda e uma pessoa não surda é por meio de texto escrito, o que é pouco eficiente. Presentemente existem aproximadamente 100.000 portugueses que sofrem de deficiência auditiva [1]. Isto traduz-se em 100.000 pessoas que diariamente enfrentam dificuldades em realizar simples tarefas quotidianas que envolvam a interação com uma pessoa não surda. Adicionalmente, a faixa etária mais jovem desta comunidade ainda encontra dificuldades na educação, o que se traduz num menor desenvolvimento linguístico e social em relação aos estudantes que ouvem [2]. Apesar de a Língua Gestual Portuguesa (LGP) ser uma das línguas oficiais de Portugal desde 1997 [3], a sua integração na sociedade está longe de ser perfeita. Embora existam projetos ligados à tradução automática desta Língua estes não são portáteis, envolvendo uma câmara *Kinect* [2], [4],que limita a utilização do sistema fora de casa.

1.2 Objetivos

De forma a solucionar a barreira de comunicação entre uma pessoa surda e uma não-surda, pretende-se criar uma luva *wireless low-cost* capaz de traduzir o alfabeto da LGP para texto. Almeija-se assim facilitar a comunicação em situações básicas da vida quotidiana sem ser necessária a tradução de um intérprete de LGP, dando desta forma "voz"às pessoas que não a têm. Nesta dissertação propõem-se desenvolver um sistema completo de tradução de LGP que inclui:

- Uma luva *wireless* com *hardware* integrado, colocado estrategicamente de forma a ser percetível o movimento dos dedos e da mão, conseguindo associar os mesmos a uma letra da LGP.
- Um algoritmo capaz de associar cada letra da LGP a uma letra da Língua Portuguesa.
- Uma aplicação móvel onde é mostrada em tempo real a informação adquirida pelos sensores, neste caso o texto traduzido. Este "tradutor" permite assim à comunidade surda uma integração completa na sociedade.

1.3 Contribuições

Os resultados obtidos com este projeto resultaram num protótipo *low cost* e de fácil utilização. Foi publicado um artigo científico numa conferência nacional em Lisboa, ConfTele 2019. A publicação deste artigo contribui para a sua disseminação na comunidade cientifica.

- S. Ferreira, N. Souto, O. Postolache, "Mobile Hand Gesture Recognition System for the Portuguese Sign Language", at the 11th Conference on Telecommunications (ConfTele), presented in June 2019.

1.4 Estrutura da Dissertação

A dissertação é composta por sete capítulos. Depois deste capítulo introdutório, o capítulo 2 relaciona o estado de arte, introduzindo projetos relacionados com temas como o reconhecimento de gestos, *hardware wearable*, protocolos de comunicação, e *Machine Learning*. No capítulo três é apresentada em pormenor a arquitetura do sistema, em específico o material utilizado, como este é cosido na luva e o consumo do sistema final. No capítulo 4 é feita uma descrição da classificação, nomeadamente que dados são utilizados, de que forma são recolhidos e preparados e como os dois algoritmos procedem. No capítulo 5 está representada a aplicação *Android* desenvolvida para a visualização da tradução. O capítulo 6 apresenta os testes e resultados obtidos com o sistema. Por fim no capítulo 7 constam as conclusões da dissertação apresentada, com foco nos aspetos a serem melhorados e trabalho futuro.

Capítulo 2

Revisão da Literatura

Este capítulo aborda a revisão da literatura dos aspetos mais importantes relacionados com o trabalho desenvolvido. O capítulo está dividido em seis secções: na secção 2.1 é introduzido o tema da origem da Língua Gestual e da Língua Gestual Portuguesa em específico assim como os parâmetros comuns a todas as Línguas Gestuais, na secção 2.2 são abordados os diferentes tipo de reconhecimento de gestos, na secção 2.3 estão descritos os vários tipos de protocolos de comunicação, na secção 2.4 é feita uma breve introdução ao tema da aprendizagem supervisionada, na secção 2.5 são introduzidos os sensores *wearable* e por fim na secção 2.6 é feita uma breve introdução de projetos relacionados com a tradução da Língua Gestual que contém uma aplicação móvel.

2.1 Língua Gestual

A Língua Gestual teve a sua primeira abordagem em 1500 pelo filósofo Girolamo Cardano que afirmou "...a surdez e mudez não é o impedimento para aprender e o meio melhor é através da escrita... e é um crime não instruir um surdo-mudo.". Os primeiros passos para a evolução da Língua Gestual foram dados, até que em 1880 aquando do Congresso Internacional de Surdo-Mudez, foi aprovado exclusivamente o método oral pelas escolas de surdos e a Língua Gestual foi proibida oficialmente.

O reconhecimento da Língua Gestual foi incitado mais tarde, em 1965, após a publicação de uma pesquisa realizada por William Stokoe. Em 1982 foi implementado o Sign Writing, método de escrita da Língua Gestual.

Existem atualmente 466.000.000 pessoas que sofrem de deficiência auditiva, número que irá chegar aos 900.000.000 em 2050, segundo a Organização Mundial de Saúde (OMS) [5].

2.1.1 Língua Gestual Portuguesa

A LGP teve origem na primeira escola para surdos em 1823, na Casa Pia. Por decisão do rei D.João VI o Professor Sueco, Par Aron Borg, veio para Portugal com o objetivo de dirigir esta escola. Borg foi responsável por introduzir uma adaptação do alfabeto e Língua Gestual Sueca (LGS) na comunidade surda Portuguesa, e por isso, apesar do vocabulário ser bastante diferente na LGP e na LGS, é possível encontrar semelhanças no alfabeto [6].

Não obstante o esforço de toda a comunidade surda para o reconhecimento da LGP, apenas em 1997 é que esta se tornou uma das Línguas oficiais de Portugal. Desde o seu reconhecimento que houve uma evolução e interesse notável pela Língua, não só por parte da comunidade surda como também pela comunidade não surda. Atualmente existem 100.000 pessoas que sofrem de deficiência auditiva em Portugal [1].

2.1.2 Parâmetros da Língua Gestual Portuguesa

A LGP, assim como todas as Línguas Gestuais, são definidas por 5 parâmetros essenciais: a configuração das mãos, a localização, o movimento, a orientação e a expressão facial e corporal. Para um gesto ser realizado e interpretado corretamente é necessário que estes 5 parâmetros sejam utilizados de forma correta.

De seguida irá ser feita uma breve explicação do significado e importância de cada parâmetro.

Configuração da Mão

Na LGP existem várias configurações que podem ser representadas com apenas uma mão, como o alfabeto (Figura 2.1), ou com as duas mãos, em palavras como "sentar"(Figura 2.2). Quando um gesto é representado com as duas mãos existe sempre uma que é a dominante, e outra que é a não dominante, que por norma assume o papel do local de articulação [6].

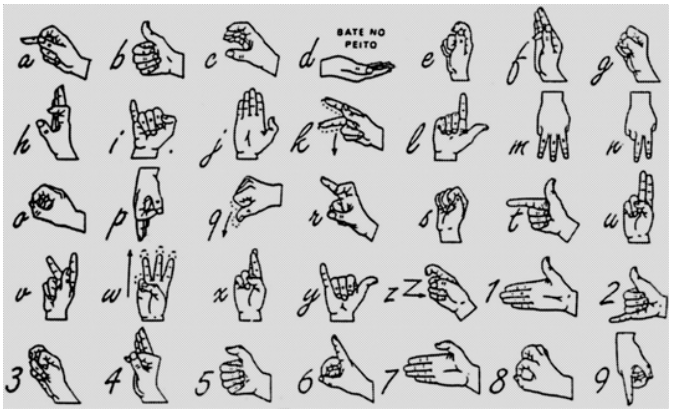


FIGURA 2.1: Alfabeto da LGP [6]

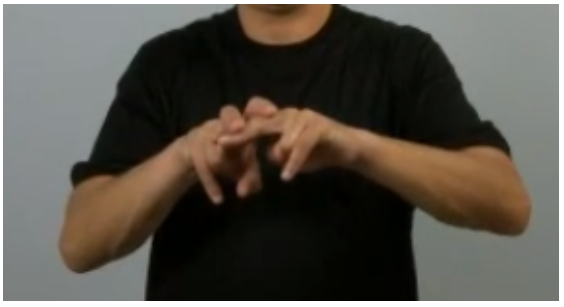
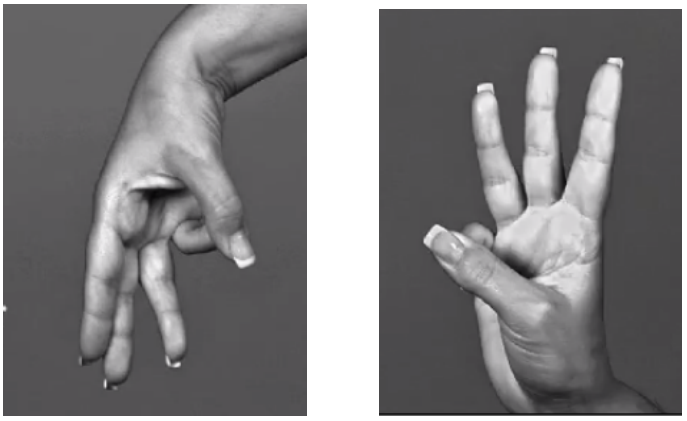


FIGURA 2.2: Representação da palavra "sentar"na LGP [7]

Orientação

A orientação da palma da mão no momento em que é representado um gesto constitui um parâmetro de distinção de gestos de configuração idêntica. É o caso da representação das letras "W" e "M" (Figura 2.3), em que os gestos têm configuração idêntica distinguindo-se pela direção da palma da mão.



(A) Letra M (B) Letra W

FIGURA 2.3: Letras do alfabeto da LGP [7]

Movimento

Os gestos podem ser caracterizados de duas formas, como estáticos ou dinâmicos. Um gesto estático é aquele que não envolve o movimento das mãos, enquanto que um gesto dinâmico é caracterizado por um ou mais movimentos. Os movimentos dinâmicos são dedilhar, torcer, friccionar, horizontal em circulo, horizontal em arco, entre outros (Figura 2.4) [6].

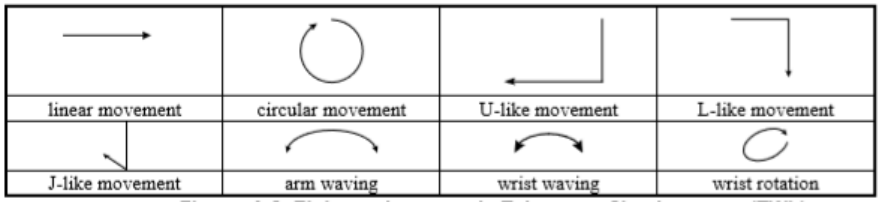


FIGURA 2.4: 8 exemplos de movimentos [8]

Localização

A localização refere-se à parte do corpo onde o gesto é realizado. Existem três grandes espaços articulatórios: o que abrange várias partes do corpo, o espaço mais ou menos próximo do corpo e ainda o espaço de articulação da mão dominante [6]. Alguns dos principais locais de articulação são a testa, o olho, o nariz, a boca, entre outros (Figura 2.5).

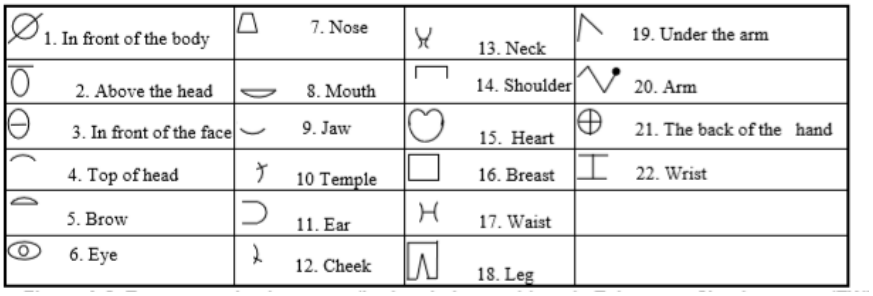


FIGURA 2.5: Pontos de articulação [8]

Expressão facial e corporal

Na Língua Portuguesa a entoação dada a uma frase ou palavra é feita através da variação de tom, no entanto, na LGP essa entoação é feita através de expressões faciais como o movimento das sobrancelhas, lábios... ou através da expressão corporal. Estas expressões imprimem significado a um gesto. Felicidade, tristeza, duvida, medo, interrogação e negação são algumas emoções que complementam os gestos (Figura 2.6). Têm-se como exemplos deste caso o "porque" e o "porquê", que, apesar de serem gestos com a mesma orientação, movimento, configuração e localização se conseguem distinguir através da expressão facial que os acompanha, no primeiro caso de neutralidade e no segundo de interrogação [9].

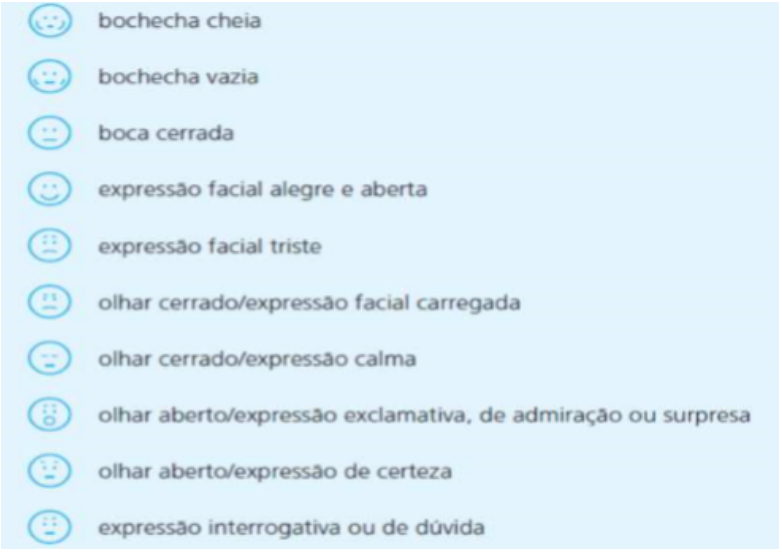


FIGURA 2.6: Exemplos de expressões faciais [6]

2.2 Sistemas de Reconhecimento de Gestos

Com o avanço constante da tecnologia é cada vez mais importante projetar uma interface de interação humano-computador. O reconhecimento de gestos é um passo importante nessa direção, uma vez que, os gestos feitos pelo utilizador são reconhecidos por uma máquina [10]. A habilidade que a máquina tem de reconhecer gestos permite que esta tecnologia seja usada não só para o reconhecimento de Língua Gestual mas também noutras áreas como a realidade virtual, engenharia robótica, aplicações militares, entre outras.

O processo que envolve o reconhecimento de um gesto é dividido em 5 fases, representadas na Figura 2.7.

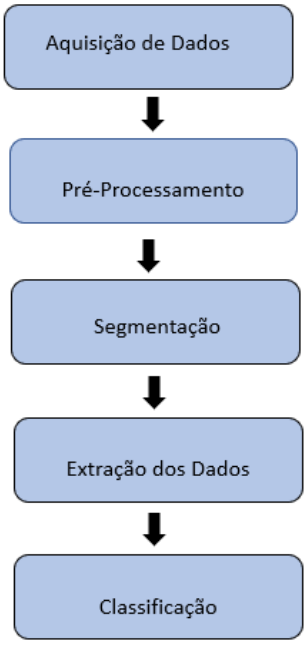


FIGURA 2.7: Fases do reconhecimento de gestos

O reconhecimento de gestos pode ser feito de diversas formas, por isso, as várias abordagens são separadas em duas áreas, as que usam imagem e as que usam sensores baseados em movimento [11]. A grande diferença de uma abordagem para outra encontra-se na forma como os dados são recolhidos [12].

2.2.1 Sistemas baseados em Sensores de Vídeo

Estes sistemas baseiam-se nas imagens recolhidas pelo sensor de vídeo, imagens essas que são associadas a uma configuração manual previamente definida [6]. Apesar desta abordagem ter a vantagem de não ser necessário o uso de acessórios, tem inúmeras desvantagens como limitações de espaço, interferência do meio em que o utilizador se encontra e as condições de luz, que tem de ser suficientes para a perceção do gesto feito pelo utilizador [11].

Esta abordagem de reconhecimento de gestos encontra-se dividida em quatro grupos [12]:

- **Câmaras de Vídeo:** consistem no uso de apenas uma câmara que se baseia em métodos de deteção como a forma ou a cor. Esta câmara pode ser de um dispositivo móvel, *web* ou ainda de vídeo.

  Thad Starner et al. [13], descreve dois sistemas para o reconhecimento de Língua Gestual Americana baseados em modelos ocultos de Makrov, tudo isto em tempo real. Ambos os sistemas fazem uso de uma única câmara que deteta o movimento das mãos, sem ser necessário a utilização de equipamento nas mãos do próprio utilizador. O primeiro sistema observa o utilizador através de uma câmara que se encontra colocada na mesa (Figura 2.8) atingindo uma precisão de 92%. No segundo sistema a câmara é colocado num chapéu usado pelo utilizador (Figura 2.9), onde se obteve uma precisão de 98%.

FIGURA 2.8: Sistema baseado na câmara colocada na mesa [13]

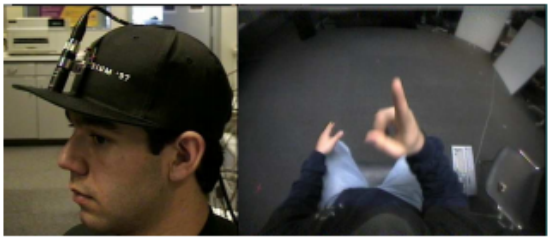


FIGURA 2.9: Sistema baseado na câmara colocada no chapéu [13]

- **Câmaras Estéreo:** fazem uso de câmaras monoculares para providenciar informação de profundidade.

- **Técnicas Ativas:** usam a projeção da luz estruturada. Deste tipo fazem parte sistemas como a *Kinect* (Figura 2.10) e o *Leap Motion* (Figura 2.11).

  Simon Lange et al [14], propõe um sistema baseado na *Kinect*, uma câmara de profundidade, que faz reconstrução 3D em tempo real. Depois da câmara detetar e rastrear a mão, recolhe os dados correspondentes a densidade que são classificados utilizando modelos de *Markov*. Nos primeiros resultados foi obtido uma taxa de precisão de 97%, mostrando que as câmaras de profundidade são indicadas para o reconhecimento gestual.

FIGURA 2.10: Componentes da *Kinect* [4]

Leigh Potter et al. [15], descreve um sistema baseado no *Leap Motion* adaptado à Língua Gestual Australiana. O *Leap Motion* foi capaz de rastrear de forma precisa as mãos, dedos e movimentos. No entanto este sistema continua a ter inúmeras desvantagens como a falha na captura da posição

dos dedos que estão escondidos do ponto de vista da câmara ou a falha na deteção quando dois dedos estão juntos.

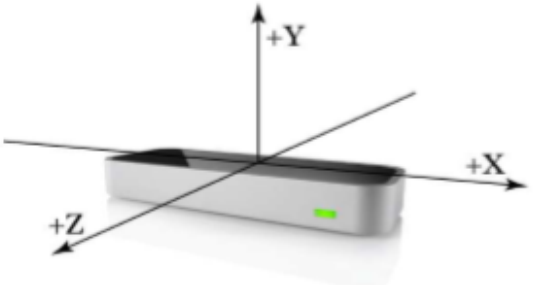


FIGURA 2.11: Disposição das coordenadas espaciais do *Leap Motion* [16]

- **Técnicas Invasivas** que fazem uso de marcadores corporais como pulseiras, luvas coloridas ou LED's.

  Exemplo desta técnica é o comando Move da *PlayStation* [17], que apesar de não ser utilizado para a tradução da Língua Gestual é utilizado para o reconhecimento gestual. A câmara da *PlayStation* capta a luz que se encontra no comando (Figura 2.12) de forma a perceber quais são os movimentos do jogador.

FIGURA 2.12: Comando Move da *PlayStation* [17]

2.2.2 Sistemas baseados em Sensores de Movimento

Estes sistemas utilizam parâmetros como a aceleração, o sinal do movimento e a velocidade angular para o reconhecimento da Língua Gestual. Têm várias vantagens quando comparados com os sistemas baseados na visão, como a ligação direta deste com o utilizador, fornecendo desta forma mais cobertura; este tipo de sensores não são influenciados pelo meio ambiente em que nos encontramos, constituindo por isso um reconhecimento mais apropriado em ambientes complexos; é um sistema *wireless* [11]. No entanto, apesar de ser a melhor área em estudo, estes sensores têm uma grande desvantagem associada às pequenas variações da configuração manual.

Esta abordagem de reconhecimento de gestos encontra-se dividida em quatro grupos [12]:

- **Unidade de Medição Inercial (IMU):** faz uso do giroscópio e acelerómetro para recolher informação de parâmetros como a aceleração e a posição dos dedos.

  O sistema representado por Jian Wu et al. [18] faz uso de uma IMU (Figura 2.13) e de uma superfície eletromagnética, responsáveis por detetar o movimento da mão e do braço. Ao serem utilizados em conjuntos estes dispositivos são capazes de aumentar o desempenho do sistema. Para o reconhecimento de cada palavra foram testados quatro algoritmos que avaliariam 80 gestos da Língua Gestual Americana de forma a perceber qual obtinha melhor precisão. Foi obtida uma precisão de 97.89% com o algoritmo *LibSVM*.

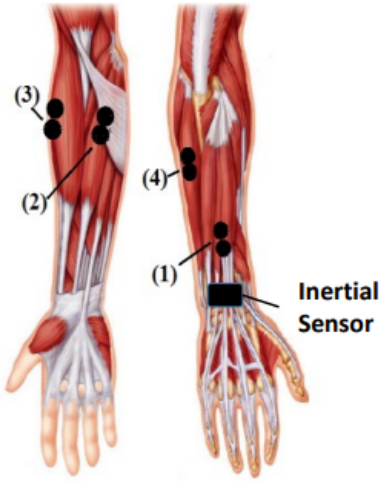


FIGURA 2.13: Localização da IMU no sistema [18]

- **Eletromagnético (EMG):** mede pulsos elétricos dos músculos para detetar o movimento dos dedos.

  Um exemplo de um sistema que integra um sensor EMG é o [19]. Jonghwa Kim et al. descreve um sistema onde o sensor EMG é colocado na parte interna do antebraço (Figura 2.14). Para além de características comuns como o valor médio e o desvio padrão de sinais elétricos, também foram calculadas características do domínio de tempo e frequência incluindo a variância de Fourier que indica quão periódico um sinal é. De forma a obter classificação em tempo real foram utilizados algoritmos como o k-NN e o Bayes. Foi obtida uma precisão de 94% ao utilizar os recursos em conjunto com os algoritmos.

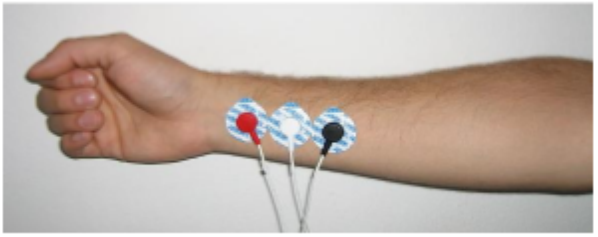


FIGURA 2.14: Sensor eletromagnético [19]

- ***Wi-fi* e radar:** utiliza radares de feixe amplo, ondas de rádio ou espectrograma para detetar mudanças na força do sinal.

Heba Abdelnasser et al. [20], descreve um sistema que utiliza *Wi-fi* denominado por *WiGest*. Este sistema faz uso das alterações da intensidade do sinal *Wi-fi* para detetar gestos executados próximo do dispositivo móvel do utilizador. Utilizando o ponto de acesso único a taxa de precisão foi de 87.5%, precisão essa que aumenta para 96% quando são utilizado três pontos de acesso.

- **Outros aparelhos:** nos quais estão incluídos sistemas compostos por sensores de flexão, ultrassónicos, mecânicos, eletromagnéticos e tecnologias hápticas.

  Nisha Kawale et al. [21] implementa um sistema baseado em cinco sensores de flexão, uma plataforma *Arduino Uno* e um módulo *Bluetooth* (Figura 2.15). Cada sensor de flexão é colocado estrategicamente em cada dedo e é recolhida a informação relativa ao gesto. De seguida o *Arduino* converte a informação de analógica para digital que é enviada através do *Bluetooth* para o dispositivo móvel do utilizador, onde é mostrado o texto traduzido.

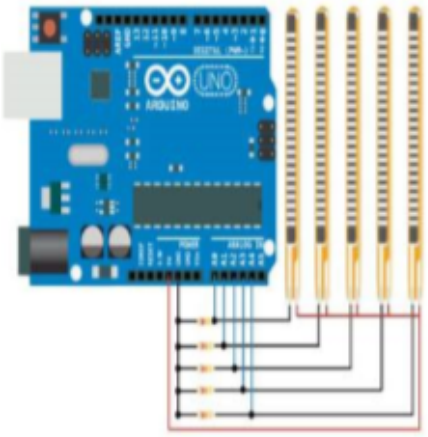


FIGURA 2.15: Sistema composto por sensores de flexão [21]

2.3 Protocolos de Comunicação *Wireless*

Para que todo o sistema de reconhecimento de gestos funcione é necessário que o protocolo de comunicação seja fiável, de forma a reencaminhar os dados recolhidos

pelo sistema, que posteriormente são analisados e tratados. Os protocolos *wireless* são uma boa opção na medida em que são protocolos sem fios, permitindo desta forma ao utilizador ter liberdade e flexibilidade no movimento [22].

Os principais protocolos de comunicação *wireless* para este tipo de aplicações são o *Bluetooth* (IEEE 802.15.1), o *ZigBee* (IEE 802.15.4) e o *Wi-Fi* (IEE 802.11), representados na Tabela 2.1 com as suas principais características.

Tabela 2.1: Características dos principais protocolos de comunicação sem fio

Características	***Wi-Fi***	***Bluetooth***	***ZigBee***
Ritmo Binário [Mbps]	11	1	0.25
Frequência [GHz]	2.4	2.4	2.4
Versão	802.11b	4.0	-
Alcance [m]	1-100	10-100	10-100
Nós	32	7	65540
Consumo [mA]	100-350	1-35	1-10
Complexidade	Alta	Média	Média
Segurança	WPA/WPA2	128 bit	128 bit

2.3.1 *Bluetooth*

O *Bluetooth* é um protocolo baseado em sistemas de rádio concebidos para dispositivos de comunicação de baixo custo e de curto alcance, substituindo desta forma os cabos de impressoras, ratos, teclados, entre outros [23]. Quando o dispositivo *Bluetooth* está ligado pode operar como *slave* ou *master*, tendo um máximo de 8 dispositivos, 7 *slaves* e 1 *master* [22].

Este protocolo permite uma comunicação direta com um dispositivo móvel, o que o torna apropriado para sistemas portáteis.

Existe também o *Bluetooth Low Energy* (BLE), versão 4.0 do *Bluetooth*, que ao contrário das versões anteriores deste, foi projetado como uma solução de baixo consumo de energia para controlar e monitorizar aplicações. A vida útil de um dispositivo BLE alimentado por uma bateria de célula tipo moeda pode atingir 14.1 anos [24].

2.3.2 *Wi-Fi*

O objetivo do protocolo *Wi-Fi* é fornecer conectividade *wireless* para dispositivos que requerem uma instalação rápida, como computadores portáteis ou telemóveis dentro de uma rede local sem fios [23].

Este protocolo permite ao seu utilizador navegar na Internet quando este se encontra num ponto de acesso ou no modo AD-HOC [22], e também, transferências de uma grande quantidade de dados de forma rápida. Com base nestas especificações pode-se concluir que este protocolo é fundamental quando o utilizador quer conectar-se rapidamente à *Internet*.

A sua grande desvantagem é o grande consumo de energia, que não permite aos dispositivos estar ligados à *Internet* por grandes períodos de tempo [25].

2.3.3 *ZigBee*

O *ZigBee* foi criado para áreas privadas *wireless*. Este protocolo é bastante utilizado quando se pretendem baixos custos, transferência de pequenas quantidades de dados e pequenos consumos de energia. [25].

2.4 *Machine Learning*

Atualmente dados como música, texto, imagens e vídeos, não são gerados apenas por pessoas mas também por computadores, telemóveis e outros dispositivos em fluxos que tendem a aumentar com o passar dos anos. O volume de dados processados determinou que os próprios sistemas de forma automática efetuassem a mudança nos padrões de dados, ao invés do que sucedia anteriormente em que estas mudanças eram inseridas manualmente. É por isso importante criar sistemas automáticos que aprendam com os dados e as suas mudanças.

A aprendizagem supervisionada consiste na programação de máquinas de forma a fazer previsões precisas, utilizando dados previamente etiquetados ou experiência do passado. Para além da sua aplicação no reconhecimento de gestos, esta tem diversas aplicações como reconhecimento de imagens, deteção de fraudes, condução automática, deteção de cancro, entre outras. Este processo, representado na Figura 2.16, divide-se em seis etapas [26], [27]:

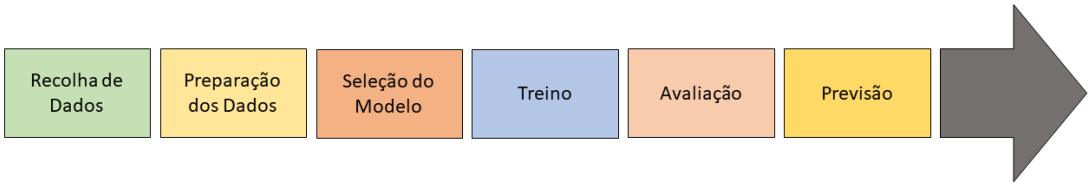


FIGURA 2.16: Etapas Aprendizagem Automática

- **Recolha de Dados:** Aqui são recolhidos os dados com que o programa é treinado. Esta etapa é bastante importante uma vez que a qualidade e quantidade de dados recolhidos determina a eficiência do modelo usado.
- **Preparação dos Dados:** Uma vez recolhidos, os dados são preparados para treino.
- **Seleção do modelo:** Existem vários modelos criados ao longo dos anos. Uns são mais adequados para a análise de imagens, outros para sequências como texto e música. É importante escolher um modelo que se adapte ao problema proposto.
- **Treino:** Feita a seleção do modelo a utilizar o passo seguinte consiste no treino desse modelo com os dados recolhidos inicialmente. Este passo é a base da *Machine Learning* uma vez que permite melhorar a capacidade do modelo identificar corretamente os dados.
- **Avaliação:** Nesta etapa o modelo é testado com os dados que não foram utilizados no treino, permitindo desta forma ver como o modelo se porta com dados que nunca viu. É fundamental que estes dados não sejam os mesmos uma vez que o modelo pode memorizar os dados utilizados no treino.

- **Previsão:** Depois de treinado o modelo é utilizado para prever resultados.

O modelo de aprendizagem supervisionada é utilizado quando existem amostras de dados previamente classificadas, que são utilizadas para treinar o algoritmo. Existem vários algoritmos deste modelo que podem ser escolhidos para o treino dos dados recolhidos, como o *k-Nearest Neighbours* (k-NN), as árvores de decisão, máquina de vetor de suporte, Redes Neuronais, entre outros [12].

O k-NN é uma das técnicas mais simples e tradicionais. Este modelo divide-se em seis etapas [12], [26]:

1. Recebe um dado de teste.
2. Mede a distância Euclidiana do novo dado em relação a todos os dados que já estavam previamente classificados.
3. Obtém X, que neste caso é o parâmetro k, menores distâncias.
4. Verifica a que classe de dados, previamente classificados, pertence cada um dos dados com menor distância.
5. Assume como resultado a classe que mais dados com menor distância tem.
6. Classifica o dado de teste com a classe que foi dada como resultado.

A sua vantagem é a eficiência em computação e a sua facilidade de implementação. As rede neuronais (Figura 2.17) dividem-se em seis etapas [26]:

1. Recebe dados de treino iguais para cada classe, tendo cada classe um conjunto de dados ativo (com valor 1) e o restante desativo (valor 0).
2. Cria várias redes neuronais, cada uma para cada classe.
3. Cada rede neuronal aprende um padrão para a respetiva classe.
4. Recebe dados de teste que vão ser o input de todas as redes neuronais.

5. Cada rede neuronal vai devolver um valor entre 0 e 1, em que 0 é quando os dados de teste não são semelhante aos dados de treino da classe e 1 quando são semelhantes.

6. Assume-se como resultado a rede neuronal que tiver o valor mais perto de 1.

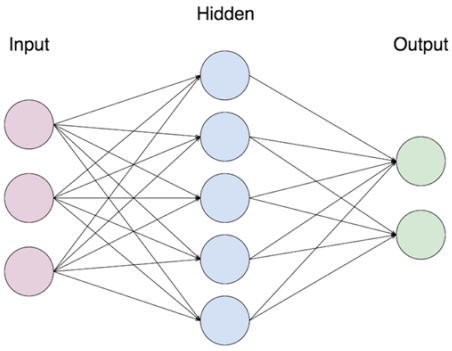


FIGURA 2.17: Esquema das Redes Neuronais [26]

Uma rede neuronal (Figura 2.17) é composta por vários nós. A primeira camada de nós (*input*) recebe as características, na segunda camada de nós (*hidden*) é feito o processamento e por fim na última camada de nós (*output*) é apresentado o resultado final.

2.5 *Hardware Wearable*

O interesse pela área que faz uso de sensores *wearable* tem crescido exponencialmente. Estes sensores são do tipo que podem ser cosidos à roupa do utilizador ou utilizados diretamente no corpo. Existem vários tipos de dispositivos *wearable*, em que os mais utilizados são os rastreadores de atividade e *smart watches*. A capacidade de se conectarem à *Internet* é uma das características principais da tecnologia *wearable*, permitindo que uma rede e um dispositivo troquem dados.

2.5.1 Tipos de sensores

Como foi estudado na secção 2.2, os sensores são uma parte fundamental no reconhecimento de gestos. Os sensores *wearable* tem como objetivo medições fisiológicas como a flexão.

A flexão é responsável por indicar a posição dos dedos da mão. Habitualmente os sensores de flexão, responsáveis por medir a quantidade de flexão, estão apegados a uma superfície e a resistência destes varia quando a superfície é dobrada. A resistência tem um correlação com o raio da curvatura feita pelo sensor, uma vez que quanto menor for o raio, maior será o valor de saída da resistência (Figura 2.18) [28].

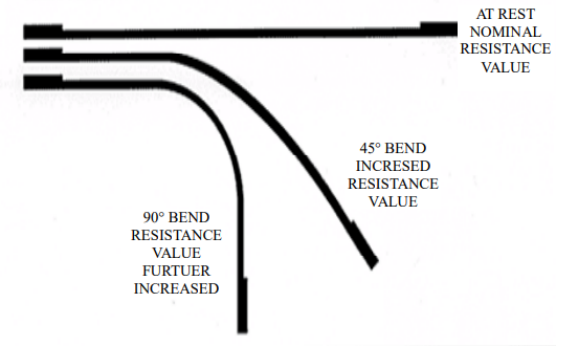


FIGURA 2.18: Curvatura do sensor de flexão [28]

Áreas de pesquisa como reabilitação, sistemas de segurança, interface humano-computador têm feito uso de sensores de flexão, sendo que este também é bastante utilizado por estudantes. Na área do reconhecimento de gestos são bastante utilizados dois tipos de sensores de flexão: o sensor capacitivo e o sensor piezoresistivo.

- **Sensor Capacitivo:**

  Este sensor baseia-se na variação do campo elétrico que varia de acordo com a distância do sensor à superfície de contacto. Este é bastante utilizado quando se quer medir o ângulo de flexão de um dedo de forma a localizar cada dedo em relação à mão ou o ângulo de flexão de uma articulação do

corpo humano, para determinar a posição relativa e movimento dos membros do corpo [29].

Kalpattu S. Abhishek et al. [30] desenvolveu uma luva baseada em sensores capacitivos (Figura 2.19) direcionada para a Língua Gestual Americana. Este sistema reconhece letras desde o A até ao Z e números do 0 ao 9 com uma precisão de 92%. As vantagens deste sistema são o *hardware* de baixo custo assim como a facilidade de mobilidade do sistema.

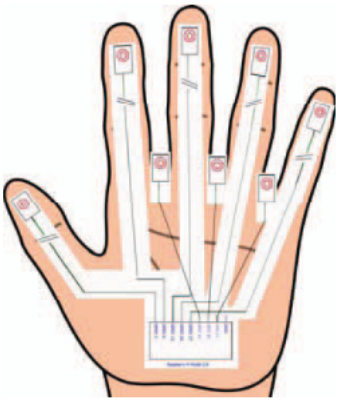


FIGURA 2.19: Disposição dos sensores capacitivos na luva [30]

- **Sensor Piezoresistivo:**

Este sensor faz uso de um material chamado *velostat*, que é bastante utilizado por amadores ou estudantes devido ao seu baixo custo. Este material é opaco, feito de uma folha poliolefina impregnada com carbono de forma a torná-lo eletricamente condutivo. A sua resistividade é de 500 ohm-metros, diminuindo quando o *velostat* é pressionado. Ao juntar-se duas camadas do material condutivo, este tem um alcance de pressão bastante elevado. Para além de ser capaz de medir a quantidade de flexão, também pode ser utilizado para deteção da posição [31], [32].

Eunseok Jeong et al. [31] desenvolveu um sistema, representado na Figura 2.20, baseado numa luva capaz de reconhecer o gesto dos dedos. Esta luva faz usos de materiais baratos como o *velostat*, um conversor analógico-digital, um micro computador, um módulo *Bluetooth* e um 5-Dof IMU. O sistema é capaz de medir

o ângulo de cada dedo quando são dobrados através do *velostat*, e calcular o ângulo de movimento da mão através do 5-Dof IMU.

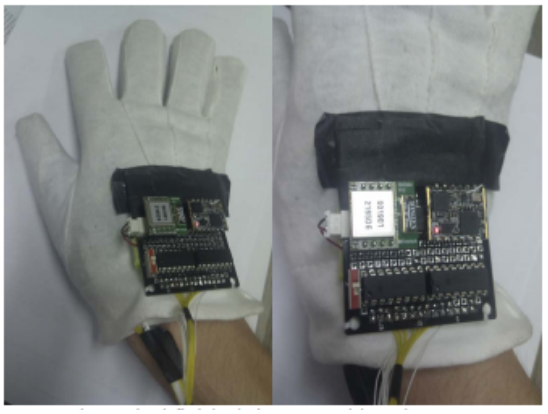


FIGURA 2.20: Luva baseada no *velostat* e 5-Dof IMU [31]

2.6 Aplicação Móvel

Existem várias aplicações ligadas à tradução de Língua Gestual, não só aplicações móveis mas também aplicações *web*. Uma aplicação *web* é uma excelente escolha se o objetivo for apenas a apresentação de conteúdo. No entanto, se o sistema projetado necessitar de ser rápido, confiável, com um desempenho superior e portátil, a escolha deve recair sobre uma aplicação móvel. Para além destas vantagens, ao criar uma aplicação móvel, existe sempre a possibilidade de não ser necessária a conexão à *Internet*, ao contrário da aplicação *web* que exige obrigatoriamente esta especificação.

Paula Escudeiro et al. [2] desenvolveu um sistema composto por uma luva e uma aplicação *web*. Esta aplicação *web* tem várias utilizações, desde a tradução da LGP para texto (Figura2.21) como de texto para LGP (Figura2.22), através de *avatars*. Contém também um jogo didático composto por vários níveis, a começar pelo alfabeto da LGP e a acabar em frases complexas. Apesar da aplicação ser bastante completa acaba por ter uma grande desvantagem, a sua mobilidade. Como esta apenas pode ser utilizada no computador acaba por não ser útil quando falamos da utilização do sistema para tarefas diárias no exterior.

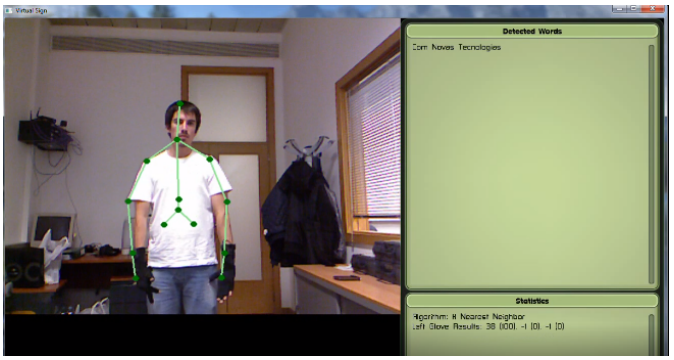


FIGURA 2.21: Tradução da LGP para texto [33]

FIGURA 2.22: Tradução de texto para LGP [33]

Um grupo de estudantes da Universidade de Nova Iorque criou uma aplicação [34] capaz de traduzir Língua Gestual Americana para texto, devido à necessidade de comunicar com um amigo surdo. Esta aplicação baseia-se em *Machine Learning* e faz uso da câmara do telemóvel que é capaz de reconhecer o gesto feito por uma pessoa e o traduz para texto (Figura2.23). Esta aplicação é ainda limitada na quantidade de frases que consegue detetar e traduzir, em que o seu único propósito é fazer a marcação de uma consulta médica.

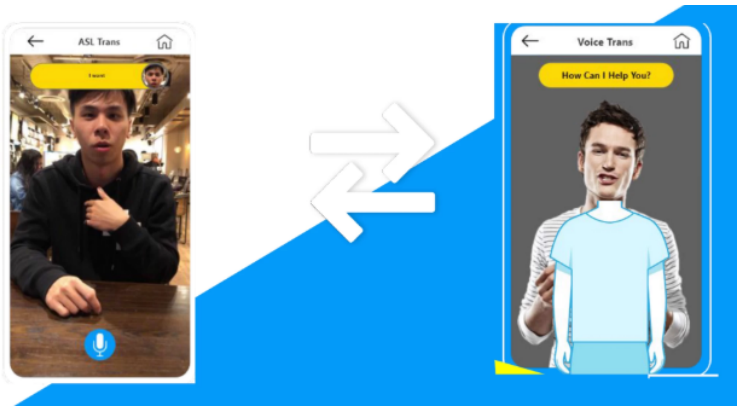


FIGURA 2.23: Tradução da Língua Gestual Americana para texto [34]

Roman Kozak desenvolveu um sistema [35] composto por uma luva e uma aplicação. A luva é responsável por recolher a informação relativa ao gesto e a aplicação é responsável por mostrar o texto traduzido(Figura 2.24). Esta aplicação foi desenvolvida para sistemas *Android* através da *MIT App Inventor*, aplicação de código aberto vocacionada para projetos escolares que impossibilita o desenvolvimento de aplicações profissionais.

FIGURA 2.24: Interface da Aplicação Móvel [35]

Capítulo 3

Arquitetura do Sistema

O sistema *wireless* e *low-cost* capaz de traduzir a LGP é baseado numa luva com sensores estrategicamente colocados e numa aplicação móvel. Este sistema é composto por duas fases como se pode verificar na Figura 3.1. A fase da obtenção da informação que diz respeito às medições efectuadas pelos sensores da luva e o envio destas para um dispositivo móvel, e a fase da tradução que está associada ao algoritmo de classificação e à aplicação móvel.

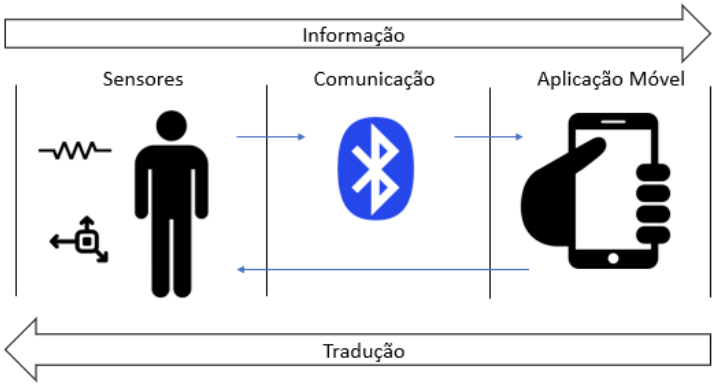


FIGURA 3.1: Diagrama de Informação/Tradução

Na primeira fase é utilizado um conjunto de sensores embebidos numa luva que providenciam a informação necessária para a tradução do gesto feito pelo utilizador. Estes sensores estão implementados na luva tendo em consideração o conforto

do utilizador. A fase da tradução destina-se a mostrar no ecrã do telemóvel o texto que é previamente traduzido por um algoritmo na própria aplicação.

A arquitetura da luva tem 6 blocos como se pode ver na Figura 3.2. No bloco 1 está representada a luva, onde os sensores e a bateria foram colocados. A luva mais à esquerda representa a perspetiva da luva vista de trás e a mais à direita representa a perspetiva da luva vista de frente.

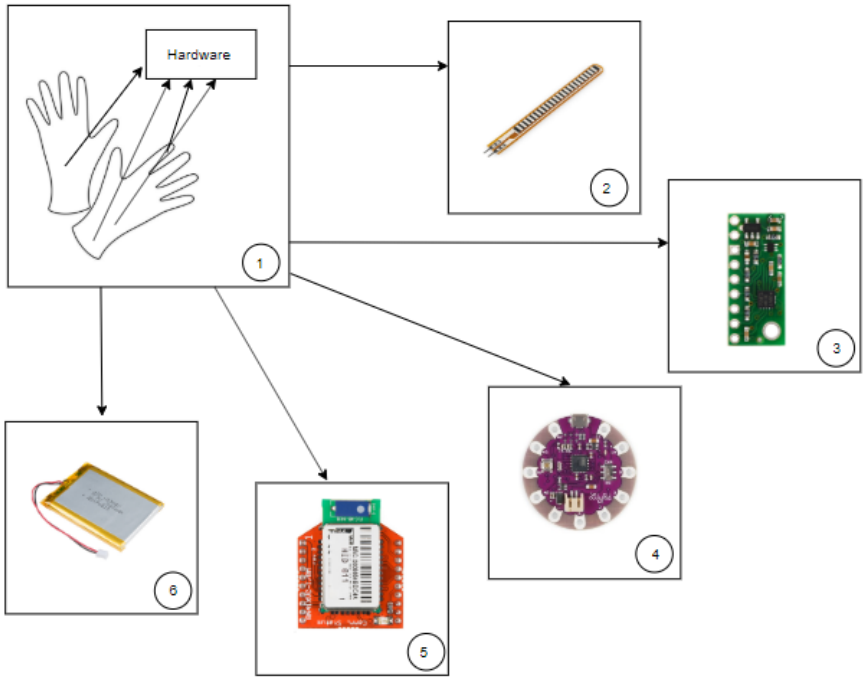


FIGURA 3.2: Arquitetura da luva

No bloco 2 está representado o sensor de flexão. Na luva existem cinco sensores de flexão, um por cada dedo . No capítulo 2 foram estudados dois tipos de sensores de flexão. O mais adequado para o sistema é o piezoresistivo, baseado no *velostat*, devido ao seu preço e à sua facilidade de implementação. Este tipo de sensor é flexível, leve e resistente a quase todos os tipos de meio, tornando-se a escolha mais viável. Estes estão cosidos no dorso dos dedos e têm como objetivo dar informação relativa à posição do dedo

O bloco 3 mostra o acelerómetro, responsável por indicar a posição da palma da mão. Este é cosido no dorso da mão. Tanto os sensores de flexão como o

acelerómetro dão a informação necessária para classificar cada uma das letras do alfabeto.

O bloco 4 mostra o coordenador, que é cosido também no dorso da mão. Na secção seguinte serão estudadas várias plataformas *wearable LilyPad* de forma a perceber qual a que melhor se adapta ao sistema.

No bloco 5 está representado o módulo *Bluetooth* que será responsável pela troca de informação entre a luva e o dispositivo móvel. No capítulo 2 foram estudados três tipos de protocolos, o *Bluetooth*, o *Zigbee* e o *Wi-fi*. Uma vez que não é possível realizar testes com cada um deles, a escolha do protocolo a utilizar baseia-se nas vantagens e desvantagens de cada um, assim como na respetiva adaptação às especificações do projeto. O protocolo *Wi-fi* não é o mais apropriado uma vez que apesar de permitir ritmos muito altos, tem a desvantagem de ter elevados consumos de energia, sendo mesmo superior ao consumo de energia do *Bluetooth* e do *Zigbee* (Tabela 2.1). O protocolo *Zigbee* também não é adequado uma vez que no sistema proposto, os dados recolhidos pelo sensor são enviados diretamente para o telemóvel e não para uma base de dados. Dado que o *Zigbee* não consegue conectar-se diretamente com o telemóvel, necessitando de um servidor, a comunicação fica comprometida. Uma vez que o projeto necessita de uma comunicação direta entre o dispositivo móvel e a luva e o *Bluetooth* e está disponível na maioria dos dispositivos móveis, será este o protocolo escolhido. A versão utilizada é a 4.0, pelos motivos descritos anteriormente. Este é cosido na parte da frente da mão.

Por fim o bloco 6 mostra a bateria responsável por alimentar todo o sistema, permitindo desta forma que este seja portátil.

3.1 Coordenador

O principal componente da luva é a plataforma *wearable* que funciona como um coordenador. Esta é responsável por alimentar e controlar todos os componentes *hardware* que se encontram na luva.

Para um sistema de tradução de LGP, características como tamanho reduzido ou componentes discretos são fundamentais. O *Arduino LilyPad* surgiu como uma boa solução para o sistema que se pretende implementar já que é uma placa micro controladora desenhada para ser utilizada na roupa, tendo a particularidade de, em regra poder ser lavada. Este componente pode ser ligado a fontes de alimentação, sensores e atuadores com fio condutor cosido na roupa [36].

O principais tipos de *Arduino LilyPad*, são o *LilyPad Arduino Simple*, o *LilyPad Arduino Main Board*, o *LilyPad Arduino USB* e o *LilyPad Arduino Simple Snap*, representados na Tabela 3.1, com as suas principais características [37], [38], [39], [40].

TABELA 3.1: Características das Plataformas *Arduino Wereable*

Características	***Simple***	***Main Board***	***USB***	***Simple Snap***
Micro Controlador	ATmega328P	ATmega168V	ATmega32U4	ATmega328
Linguagem de Programação	C/C++	C/C++	C/C++	C/C++
Pins Analógicos	4	6	4	4
Pins Digitais	9	14	9	9
Memória [kB]	32	16	32	32
RAM [kB]	2	2	2.5	2
Diâmetro [mm]	50	50	50	50
Preço Médio [€]	18	18	22	26

- ***LilyPad Arduino Simple*:** Este *Arduino* é um micro controlador baseado no ATmega328P, tem 9 pins para entrada/saída, um conector *Japan Solderless Terminal* (JST) e um circuito integrado para baterias de polímero de lítio. O *LilyPad Arduino Simple* (Figura 3.4) foi projetado de forma a eliminar a necessidade de costurar uma fonte de alimentação [37].

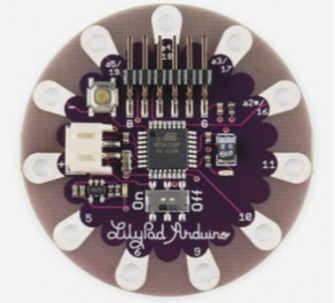


FIGURA 3.3: *LilyPad Arduino Simple* [37]

- ***LilyPad Arduino Main Board*:** Este micro controlador é baseado no ATmega168V (uma versão *low-power* do ATmega 168) ou no ATmega 328P, também utilizado pelo *LilyPad Arduino Simple*. Tem 14 pins digitais para entrada/saída e 6 pins analógicos, o que faz com que seja fácil conectar outros componentes a este *Arduino*. Esta placa (Figura 3.4) é das mais completas, socorrendo-se do mínimo de componentes externos, o que a torna simples e pequena [38].

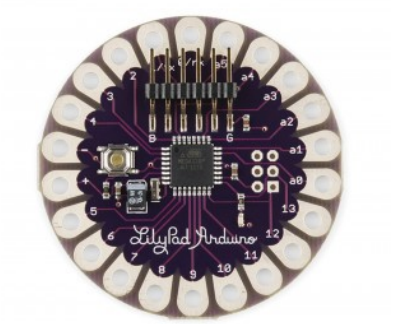


FIGURA 3.4: *LilyPad Arduino Main Board* [38]

- ***LilyPad Arduino USB*:** Este *Arduino* (Figura 3.5) é controlado pelo ATmega32U4 e é composto por 9 pins digitais de entrada/saída. A única peça de *hardware* necessária para programar esta placa é um cabo micro-USB, já que o chip integrado tem suporte USB embutido. A bateria é recarregada através da placa o que faz com que não seja necessário o uso de carregadores externos especiais para baterias de polímero de lítio [39].

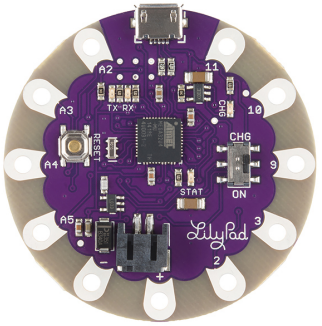


FIGURA 3.5: *LilyPad Arduino USB* [39]

- ***LilyPad Arduino Simple Snap*:** Este micro controlador (Figura 3.6) é baseado no ATmega328 e é composto por 9 pins de entrada/saída. É semelhante ao *LilyPad Arduino Simple*, distinguindo-se por ter incorporado uma bateria de polímero de lítio e *snap* condutor em vez de um buraco para cada pin, o que faz com que seja possível remover a placa de forma a lavar a roupa onde o *Arduino* está cosido [40].

FIGURA 3.6: *LilyPad Arduino Simple Snap* [40]

Sendo o sistema final portátil e *low-cost*, é importante que o *LilyPad* escolhido seja de dimensões e preço reduzidos, parâmetros que em todos são semelhantes. Outros fatores a ter em conta são o número de pins analógicos, uma vez que cada dedo da luva está ligado diretamente a um destes pins, os pins necessários para a conexão com o acelerómetro e com o módulo *Bluetooth*, e as funcionalidades integradas de forma a diminuir a necessidade de componentes externos extra.

Ao analisar todos estes parâmetros o *LilyPad Arduino USB* é o que mais se adequa ao sistema. Este permite recarregar a bateria através da placa não sendo desta forma necessário carregadores externos, especificação que é bastante importante no sistema uma vez que quanto mais simplificado estiver mais adaptável é para a rotina dos seus utilizadores. Adicionalmente é o que tem maior RAM quando comparado com os restantes.

Apesar do *LilyPad USB* ter apenas quatro pins analógicos visíveis, como está especificado na Tabela 3.1, é percetível pelo *datasheet* do *Lilypad* (Figura 3.7) que tanto o pin 9 como o pin 10 são pins ligados ao ADC do ATmega32U4 podendo por isso ser usados como entradas analógicas adicionais.

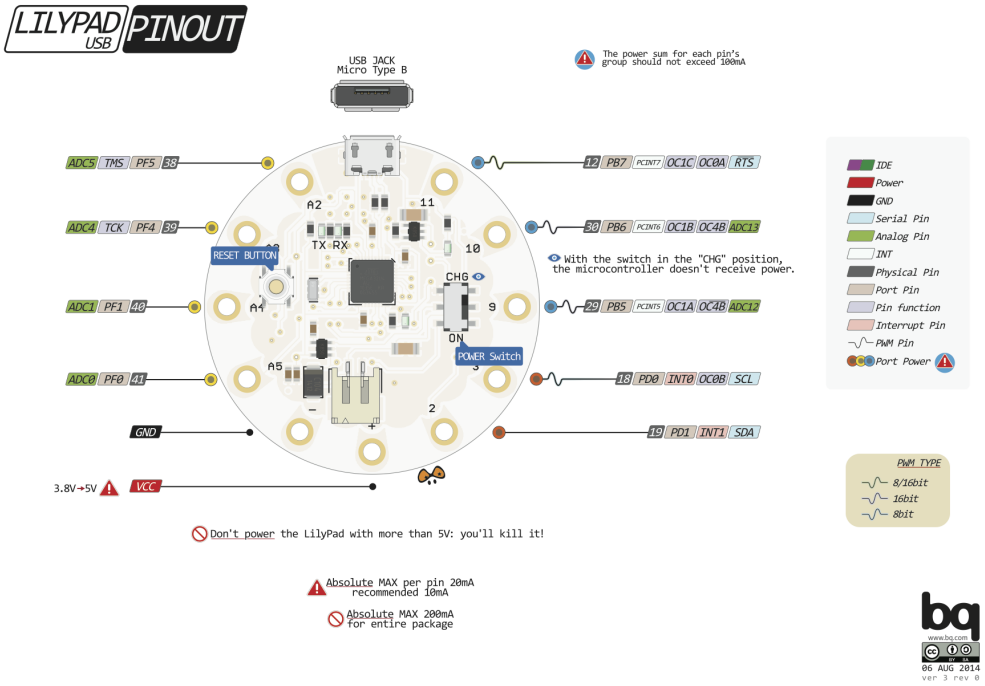


FIGURA 3.7: *Datasheet LilyPad USB* [39]

Desta forma cada sensor de flexão pode estar associado a uma entrada analógica.

Além dos pins necessários para os sensores de flexão, o *LilyPad USB* cumpre todos os requisitos para estabelecer uma conexão correta com o módulo *Bluetooth* e com o acelerómetro.

O *LilyPad USB* foi cosido no dorso da mão (Figura 3.8) de forma a não interferir no movimento da mão do utilizador.

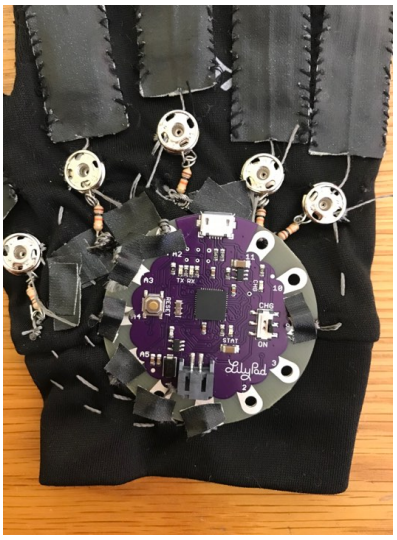


FIGURA 3.8: *LilyPad* cosido na luva

3.2 Parâmetros Recolhidos

Os componentes *hardware* são responsáveis por recolher e processar a informação necessária para a tradução da LGP. No sistema existem dois parâmetros que são recolhidos pelo *hardware*, o movimento dos dedos e a orientação da mão. A junção destes parâmetros permite identificar um gesto da LGP.

3.2.1 Movimento dos dedos

O movimento dos dedos feito pelo utilizador é medido pelos sensores de flexão, como foi visto no Capítulo 2, secção 2.5.

Para construir os cinco sensores de flexão necessários à luva, foram precisos materiais como fita adesiva, fio condutor e *velostat*, materiais estes de baixo custo. O processo de construção do sensor divide-se em cinco etapas [41]:

1. Medir o comprimento desde a ponta do dedo até metade do dorso da mão. Cada sensor vai ficar com um comprimento diferente, dependendo do dedo em questão.

2. Cortar duas fitas adesivas com as medidas retiradas no passo 1. As duas fitas vão revestir o sensor de flexão.

3. Cortar o fio condutor de forma a que tenha mais 5 centímetros sobre uma das pontas da fita e menos 3 cm sobre a outra ponta da fita (Figura 3.9). Repetir este passo para a outra fita adesiva.

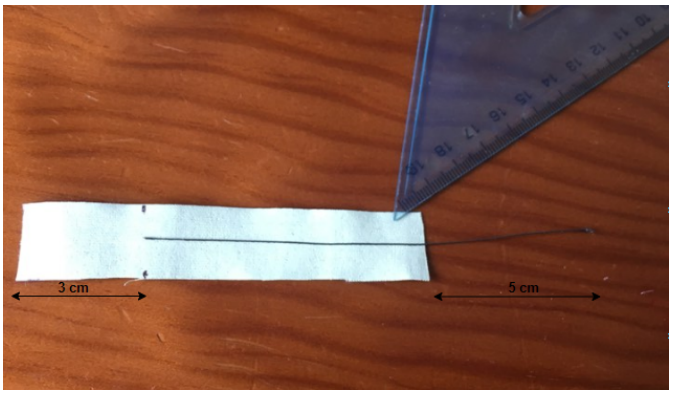


FIGURA 3.9: Fio condutor sobre a fita adesiva

4. Cortar três tiras de *velostat*, com menos 1 cm de comprimento e 0.5 cm de largura que a fita adesiva e colocar uma tira por cima do fio condutor de forma a ficar centrado (Figura 3.10). Repetir este passo para a outra fita adesiva e guardar a terceira tira de *velostat*.

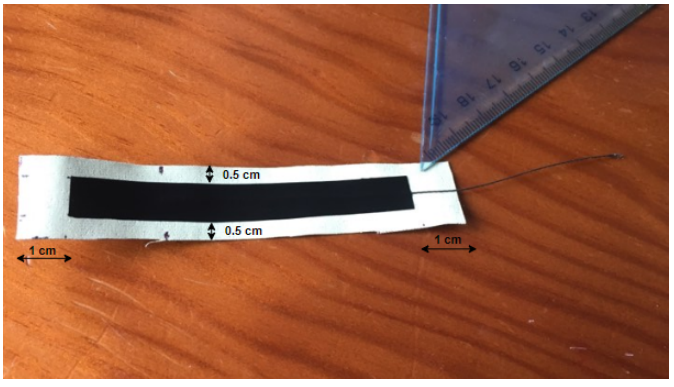


FIGURA 3.10: *Velostat* sobre o fio condutor

5. Unir as duas metades, em que a extremidade mais longa do fio condutor de uma metade fica para um lado e a da outra metade para o lado contrário, colocando a terceira tira de *velostat* entre as duas metades (Figura 3.11).

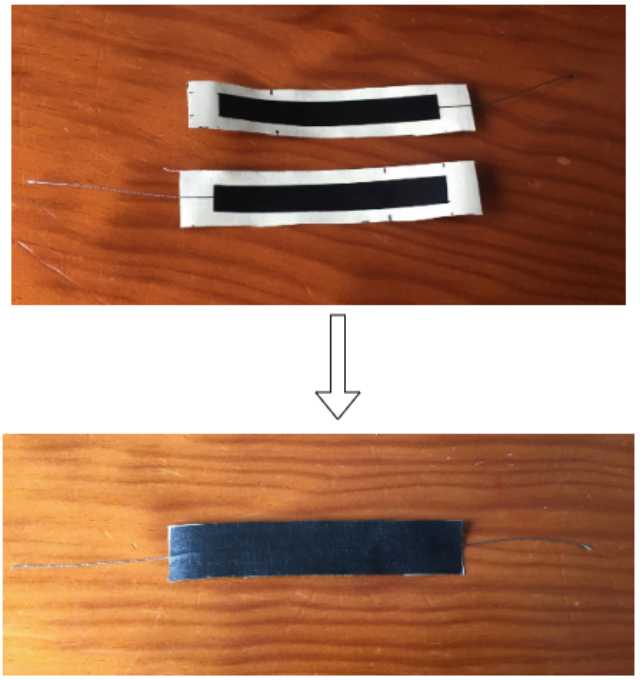


FIGURA 3.11: Sensor de Flexão

Neste projeto os sensores são cosidos na parte de trás da luva, em que cada dedo tem um sensor de flexão colocado na parte superior (Figura 3.12).

FIGURA 3.12: Sensores de flexão cosidos na luva

Uma vez que o sensor de flexão atua como uma resistência, foi necessário perceber qual a sua resistência máxima e mínima. Para tal foram retirados valores com a ajuda do multímetro para ambas as situações (Tabela 3.2). Estes valores dizem

respeito a uma média de cinco testes feitos para calculo da resistência máxima e mínima.

TABELA 3.2: Valores médios dos sensores de flexão

Dedos	**Máximo**	**Mínimo**
Polegar	93kΩ	5kΩ
Indicador	100kΩ	6kΩ
Meio	98kΩ	6.5kΩ
Anelar	90kΩ	7kΩ
Mindinho	95kΩ	6kΩ

Para obter valores mais precisos à saída dos cinco sensores de flexão é necessário utilizar um circuito de condicionamento (Figura 3.13).

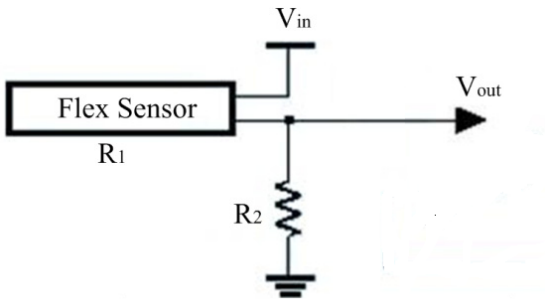


FIGURA 3.13: Circuito do sensor de flexão

Onde os parâmetros estão definidos como:

- Vout: Tensão de saída [V]
- $R2$: Resistência que limita [$k\Omega$]
- $R1$: Resistência variável que é dada pelos sensor de flexão [$k\Omega$]
- Vin: Voltagem de referência [V]

O valor Vin utilizado foi de 3.3V, valor dado à saída do *LilyPad USB*. Para a resistência *R2* foi escolhido o valor de 10 kΩ. Foram testadas várias resistências de forma a perceber o intervalo de valores a que cada sensor de flexão ficou limitado com a resistência, sendo que a de 10kΩ foi a que melhor se adequou ao sistema.

Após vários testes feitos ao sistema foi percetível que os valores dos sensores de flexão aumentaram. Nos primeiros testes realizados estes tinham uma janela média na ordem dos kΩ, valores que se mantiveram durante vários testes realizados. Mais tarde estes valores começaram a apresentar uma janela com valores médios na ordem dos MΩ. A variância de valores parece estar relacionada com a tensão de costura.

3.2.2 Direção da mão

Para calcular a direção da palma da mão foi utilizado o acelerómetro LSM303D 3D (Figura 3.14) que combina um acelerómetro de 3 eixos, o qual permite detetar a direção do utilizador em relação ao eixo da Terra ou a aceleração da placa num espaço 3D (Figura 3.15), com um magnetómetro de 3 eixos que deteta de onde vem a força magnética mais forte, de forma a perceber onde se encontra o norte magnético. Para o sistema apenas é importante ter em consideração a funcionalidade do acelerómetro.

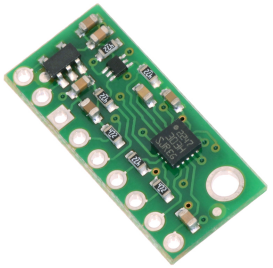


FIGURA 3.14: Acelerómetro LSM303 3D [42]

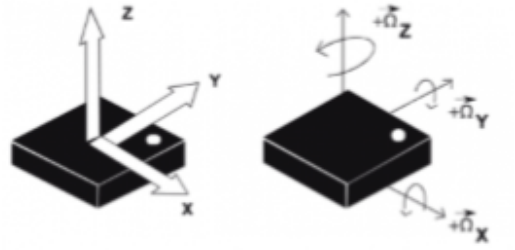


FIGURA 3.15: 3-eixos Acelerómetro LSM303 3D [42]

Para a comunicação feita entre o acelerómetro e o *LilyPad USB* foram pensados dois protocolos, o *Inter-Integrated Circuit* (I^2C) e o *Serial Peripheral Interface* (SPI). Quando comparados foi claro que o I^2C requer muito menos ligações que o SPI (Figura 3.16), fator a ter em conta uma vez que a luva é de tamanho reduzido e quanto menos ligações existirem menos confuso é o sistema. Devido à maior facilidade de implementação, a comunicação é feita através do protocolo I^2C. Este protocolo opera no modelo *master-slave*, em que um dispositivo atua como *master* tendo a função de coordenar a comunicação, e os outros todos atuam como *slave* [43].

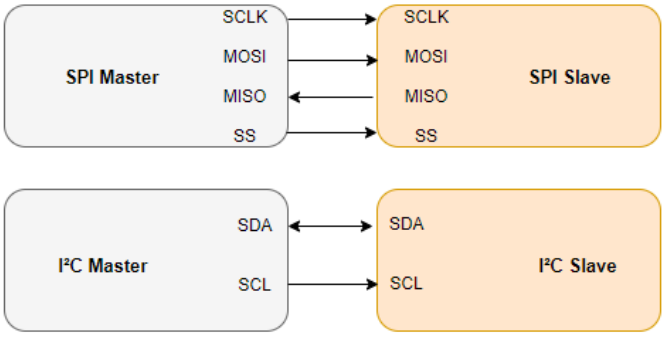


FIGURA 3.16: Protocolo SPI Vs. I^2C

O acelerómetro é responsável por diferenciar as letras que tenham valores semelhantes para os sensores de flexão. Temos como casos dessa situação as letras: A/R, B/Q/E, K/V, M/W, L/T e U/X. Cada um destes conjuntos de letras apresenta uma configuração semelhante, sendo as letras distinguíveis entre si pela direção da palma da mão.

Uma vez que o acelerómetro tem valores inconstantes, basta uma pequena variação da mão para os valores alterarem drasticamente, as letras com configuração igual vão ser distinguidas entre si não pelos valores das coordenadas X, Y e Z mas sim pelo seu sinal (positivo ou negativo). Da Tabela 3.3 à Tabela 3.8 estão representados os sinais das coordenadas para cada conjunto de letras semelhantes.

Tabela 3.3: Sinais das coordenadas da letra M e W

Letra	Coordenada X	Coordenada Y	Coordenada Z
M	+	-	+
W	+	+	-

Tabela 3.4: Sinais das coordenadas da letra A e R

Letra	Coordenada X	Coordenada Y	Coordenada Z
A	+	+	-
R	+	+	-

Tabela 3.5: Sinais das coordenadas da letra B, Q e E

Letra	Coordenada X	Coordenada Y	Coordenada Z
B	+	+	+
Q	-	-	-
E	+	+	-

Tabela 3.6: Sinais das coordenadas da letra K e V

Letra	Coordenada X	Coordenada Y	Coordenada Z
K	+	+	-
V	+	+	+

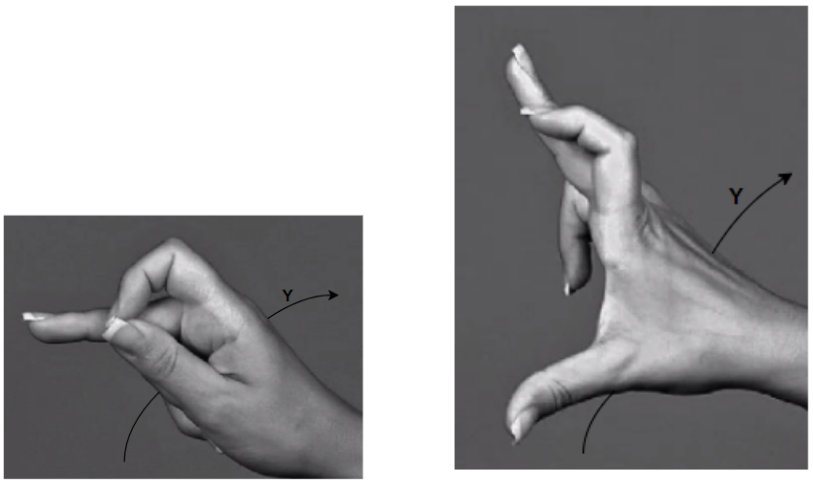
Tabela 3.7: Sinais das coordenadas da letra L e T

Letra	Coordenada X	Coordenada Y	Coordenada Z
L	-	+	-
T	+	+	+

Tabela 3.8: Sinais das coordenadas da letra U e X

Letra	Coordenada X	Coordenada Y	Coordenada Z
U	+	+	+
X	+	+	-

A Tabela 3.4 mostra sinais iguais para as três coordenadas das duas letras, o que significa que este conjunto de letras não pode ser distinguido com base no sinal das coordenadas. As duas letras da Tabela 3.4 serão distinguidas com base numa janela para a coordenada Y, uma vez que é a única coordenada que tem uma variação de valores considerável quando feitas as duas letras (Figura 3.17).



(A) Letra A (B) Letra R

FIGURA 3.17: Direção das coordenadas X, Y e Z das letras do alfabeto da LGP [7]

Com uma janela de 0 a 18205 para o eixo positivo do acelerómetro, quando a coordenada Y atingir um valor maior que 6500 é assumida a letra R, caso o valor seja menor é assumida a letra A.

De forma a obter os valores necessários o acelerómetro foi cosido no dorso da mão (Figura 3.18).

FIGURA 3.18: Acelerómetro cosido nas costas da mão

Tendo em conta a posição do acelerómetro os eixos X, Y e Z vão ficar com a disposição representada na Figura **??**.

3.3 Comunicação

Para a transferência de dados foi utilizado o *Bluetooth*, protocolo de comunicação sem fios, que possibilita a comunicação entre o *LilyPad USB* e a aplicação *Android*.

Inicialmente a comunicação do sistema iria ser implementada usando o *Bluetooth BTBee* (Figura 3.20), uma vez que este é um módulo *Bluetooth low energy*. No entanto, apenas foi possível obter um dispositivo móvel com BLE incorporado no final da dissertação. Uma vez que a aplicação móvel que suporta o sistema apenas permite a conexão com dispositivos *Bluetooh* tradicionais, a conexão do BLE foi testada na aplicação "*Serial Bluetooth Terminal*"disponibilizada no *Google Play* [44]. É possível concluir que apesar de não ser utilizado o BLE no protótipo de teste, este é compatível com o sistema (Figura 3.19).

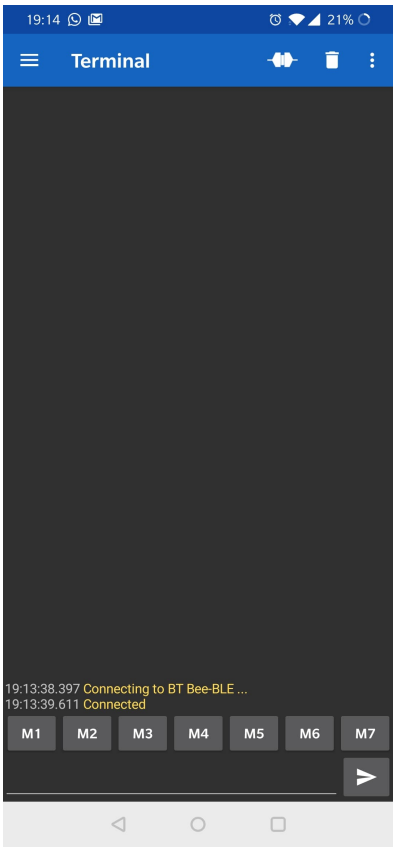


FIGURA 3.19: Conexão com o BLE [44]

A comunicação recai então sobre o módulo *Bluetooth RN41*.

FIGURA 3.20: *Bluetooth BTBee PRO* [45]

O *Bluetooth RN41* (Figura 3.21) tem uma ligação *Serial Port Profile* (SPP). Este pode operar em dois modos: "*data mode*"e "*command mode*", em que o primeiro modo é utilizado para troca de informação entre o *Bluetooth* e o dispositivo e o segundo modo é utilizado para configurar o módulo ou recolher informação acerca deste.

FIGURA 3.21: *Bluetooth RN41* [46]

Na Tabela 3.9 é possível observar as principais configurações por definição do módulo *Bluetooth*.

TABELA 3.9: Principais características do *Bluetooth RN41*

Taxa de transmissão	115,200 [Kbps]
Bits	18
Paridade	não
Low power	*off*
Modo	*slave*

Dado que o sistema é *wireless*, é pretendido que a bateria do sistema dure o mais possível, por isso a propriedade *low power* do *Bluetooth* foi ativada, desta forma existe um menor consumo de energia por parte do sistema. Para configurar este modo do *Bluetooth* foi utilizado o *TeraTerm*, emulador de terminal sugerido na referência [47] e seguidos os respetivos passos referenciados.

Para conforto do utilizador e por falta de espaço no dorso da mão, o módulo *Bluetooth* ficou colocado na palma da mão. Uma vez que este hardware não é *wearable* foi necessário utilizar o *Lilypad Xbee* (Figura 3.22), módulo que inclui guias que o tornam fácil de coser a projetos e que tem a regulação de potência necessária para correr no sistema *LilyPad*.

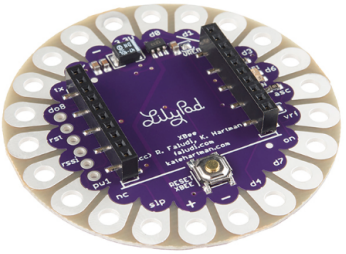


FIGURA 3.22: *Lilypad Xbee* [48]

O *Lilypad Xbee* foi cosido na luva e permite desta forma conectar o módulo *Bluetooth* ao *LilyPad USB*, como se pode verificar na Figura 3.23.

FIGURA 3.23: *Lilypad Xbee* cosido na parte da frente da mão

Apesar do *LilyPad USB* não disponibilizar os pins RX e TX, necessários para a conexão com o *Lilypad Xbee*, foi possível utilizar a biblioteca *Software Serial* [49] para definir no código *Arduino* quais os pins utilizados para esse efeito, neste caso o pin 10 e 11 correspondentemente (Figura 3.24).

```
#include <SoftwareSerial.h>

SoftwareSerial BTserial(10, 11); // RX | TX
```

FIGURA 3.24: Código *Arduino*

3.4 Linha Condutora e *Snaps*

Para interligar todos os componentes *hardware* é utilizada uma linha condutora (Figura 3.25) e *snaps* (Figura 3.26).

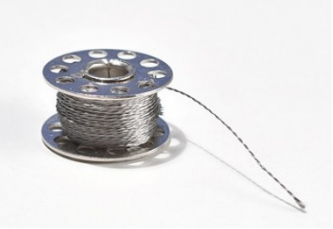


FIGURA 3.25: Linha Condutora [50]

FIGURA 3.26: *Snaps* [51]

A linha condutora escolhida para este projeto é uma linha fina de aço inoxidável do tipo 2. Como esta linha é feita inteiramente de aço inoxidável, não corre o risco de oxidar e comprometer o funcionamento da luva. Para este projeto foi escolhida a linha do tipo 2 pois o sistema vai ser costurado à mão e este tipo de linha é o mais fino. Os *snaps* são utilizados para os casos onde é necessário unir duas linhas condutoras, uma vez que não é possível soldar este tipo de linha.

3.5 Protótipo

A luva utilizada para desenvolver o protótipo é feita de poiliéster e foi escolhida devido ao seu material fino e de fácil confeção.

Na Figura 3.27 está representada a luva original, sem a implementação do *hardware*.

FIGURA 3.27: Luva sem *hardware* implementado

A luva com o *hardware* implementado está representada na Figura 3.28 (luva vista de frente) e na Figura 3.29 (luva vista de trás) com a respetiva legenda para cada um dos componentes eletrónicos.

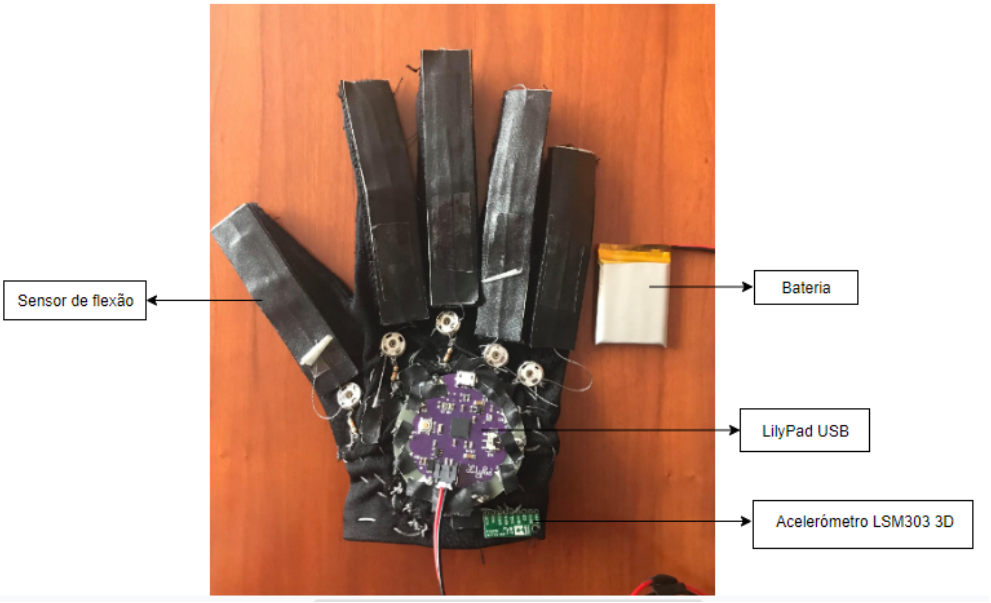


FIGURA 3.28: Luva com os componentes eletrónicos

FIGURA 3.29: Luva com os componentes eletrónicos

Na Figura 3.30 é possível observar com mais detalhe todas as ligações da luva.

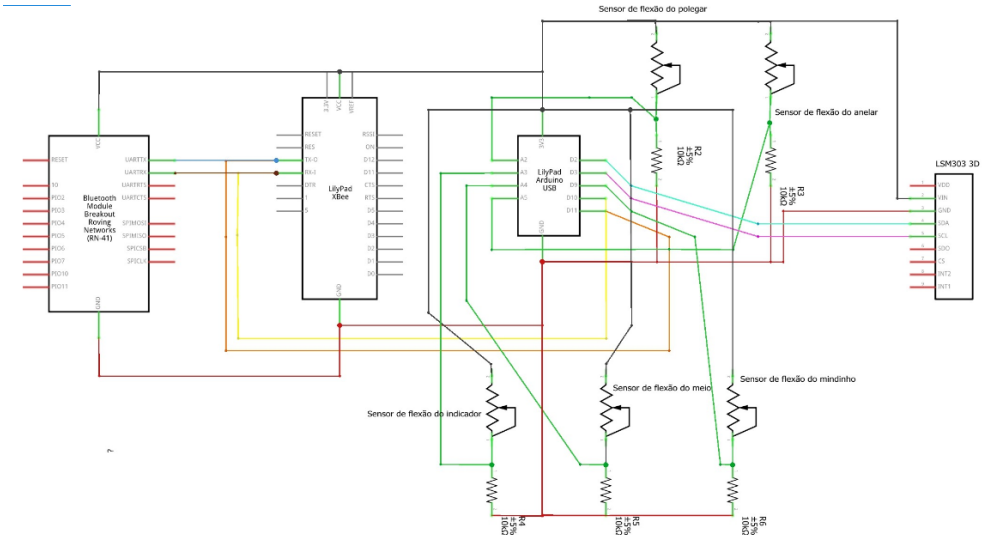


FIGURA 3.30: Esquema elétrico da luva

Esta implementação foi feita à mão uma vez que requer conhecimento sobre cada um dos componentes e a interceção da linha condutora com ela própria pode causar curto circuito, levando à falha do sistema e no pior cenário à sua avaria.

Neste protótipo o interior da luva não está revestido o que significa que a linha condutora está diretamente em contacto com a pele, o que não é cómodo para o utilizador. A pensar no conforto do utilizador, no futuro será utilizado um tecido para revestir esta linha.

Inicialmente o peso da luva era aproximadamente de 10 gramas. No sistema final o peso é de 45 gramas, fazendo com que os 35 gramas dos componentes não interfiram na mobilidade da mão do utilizador.

3.6 Consumo do sistema

Para alimentar o sistema de forma a este ser portátil, foi utilizada uma bateria de lítio com uma capacidade de 550mAh (Figura 3.31).

FIGURA 3.31: Bateria de litio [52]

Para determinar o consumo eléctrico foi utilizado um multímetro. É possível observar o esquema em série na Figura 3.32. Foi percetível então que o sistema tinha uma consumo de 32 mA, como se observa na Figura 3.32.

Para determinar o consumo do sistema foram feitos cálculos com base na Equação (3.1).

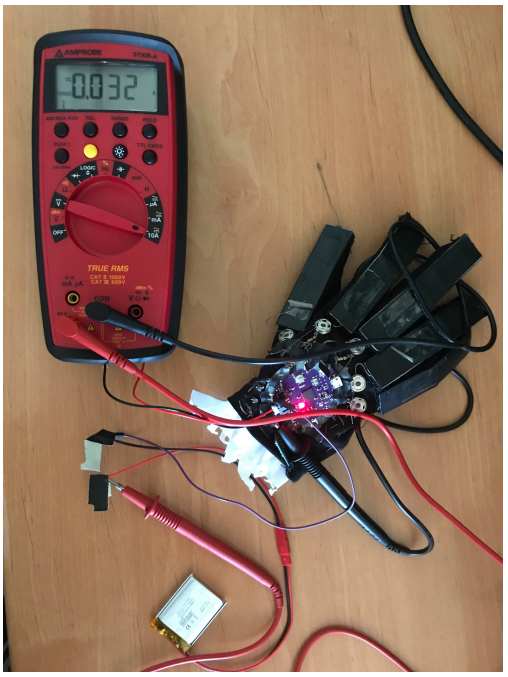


FIGURA 3.32: Análise do consumo do sistema

$$duracao = \frac{capacidade}{consumo} \tag{3.1}$$

Onde os parâmetros estão definidos como:

- duração: duração em horas do sistema
- capacidade: capacidade da bateria (mAh)
- consumo: consumo do projeto (mA)

$$duracao = \frac{550}{32} = 17h11 \tag{3.2}$$

Em suma, o sistema tem uma autonomia de 17 horas e 11 minutos antes de ser necessário um novo carregamento.

Capítulo 4

Classificação

A classificação dos dados é responsável por associar cada gesto da LGP medido pelos sensores da luva a uma letra da Língua Portuguesa. Neste capítulo estão descritos os dois algoritmos de aprendizagem supervisionada, k-NN e Redes Neuronais, que vão ser utilizados para a tradução.

4.1 Dados recolhidos

Os dados recolhidos para a classificação, tanto de treino como final, do gesto são de dois tipos:

- Posição dos dedos: São recolhidos conjuntos de medições, cada um correspondente a cada um dos dedos da mão.
- Posição da palma da mão: São recolhidas três medições, cada uma associada a um dos eixos da palma da mão.

A posição dos dedos serve para classificar todas as letras enquanto que a posição da palma da mão serve apenas para classificar as letras que tenham configuração semelhante entre si. Tem-se como exemplo deste caso as letras M e W, que ao

terem valores semelhantes apenas serão distinguidas pela posição da palma da mão.

A recolha dos dados foi feita por duas pessoas, uma pessoa com conhecimento na LGP e outra sem conhecimento na LGP. Como será percetível mais à frente, os dados de treino que classificam cada letra são recolhidos em cinco fases de amostragem. As três primeiras fases de amostragem correspondem aos dados recolhidos por uma pessoa com conhecimento em LGP enquanto que as duas ultimas fases de amostragem correspondem aos dados recolhidos por uma pessoa sem conhecimento na LGP. Desta forma os valores que classificam cada letra serão mais variados.

4.2 Preparação dos dados

Para que os algoritmos sejam eficazes é importante uma cuidada recolha e preparação dos dados, dados estes que vão servir para treinar os algoritmos. Isto significa que os dados recolhidos vão definir cada letra do alfabeto.

A posição dos dedos é retornada como um vetor com cinco valores, em que cada uma das posições está associada a um valor de cada dedo pela respetiva ordem: polegar, indicador, médio, anelar e mindinho.

Para a recolha dos dados de treino relativos a cada letra foi utilizado o *Arduino IDE*. De forma a recolher 1000 amostras de cada letra, foi feito um contador que quando atinge o valor de 999 pára o programa. Este processo foi repetido cinco vezes de forma a ter uma maior janela de valores, uma vez que o mesmo gesto nunca irá ser feito de forma igual e os valores têm pequenas variações. Foram recolhidas no total 5000 amostras de cada letra, 1000 por cada fase de amostragem. No final de cada fase de amostragem é calculado um valor médio de cada posição do vetor.

Depois dos dados de todas as letras estarem recolhidos tem-se um total de vinte e seis ficheiros, cada um correspondente a uma das letras do alfabeto.

Foi então definido que seriam apenas utilizadas 50 amostras para caracterizar cada letra, número suficiente de dados quando estes tem uma cuidada recolha.

Para tal, por cada fase de amostragem foram retiradas as dez amostras mais próximas da média com base na distância Euclidiana. Para obter as dez amostras que caracterizam cada letra foi utilizado o *Eclipse, IDE* para desenvolvimento em *Java*.

De forma a obter as 10 amostras mais próximas da média é necessário dar ao programa um ficheiro txt (Figura 4.1), onde estão os valores de uma fase de amostragem. Uma vez que cada letra tem cinco fases de amostragem, o ficheiro "valores.txt"tem de ser alterado para cada uma destas fases. No final foram então obtidos as 50 amostras para cada letra como pretendido.

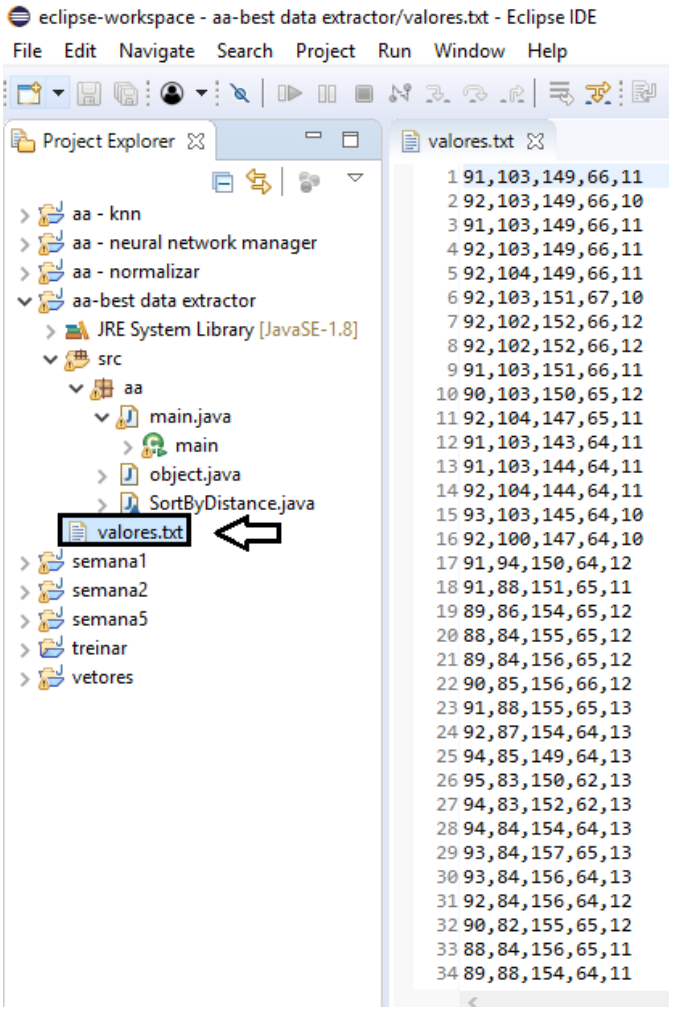


FIGURA 4.1: Ficheiro valores.txt

O programa começa por pedir a média obtida na fase de amostragem (Figura 4.2). Esta média é guardada num vetor de números inteiros.

```
public class main {

    //média de 1 fase de amostragem
    public static int mean[] = {39,191,169,116,27};
    public static void main(String[] args) {
          // ficheiro com as 1000 amostras
          File file = new File("valores.txt");
```

FIGURA 4.2: Média de cada fase de amostragem

Quando todos os valores do ficheiro txt são lidos é calculada a distância Euclidana (Equação 4.1) entre os valores do txt com a média dada.

$$d = \sqrt{\sum_{k=1}^{N} (a - b_k)^2} \tag{4.1}$$

Onde os parâmetros estão definidos como:

- N: número de atributos de cada letra
- a: média da fase de amostragem
- b_k: valores da fase de amostragem

No final são obtidos os 10 valores mais próximos da média, na consola do *Eclipse*. Estes valores são posteriormente guardados num ficheiro txt representando os valores de cada letra do alfabeto.

Na Figura 4.3 está representado o fluxograma correspondente à fase de recolha dos dados para treinar os algoritmos.

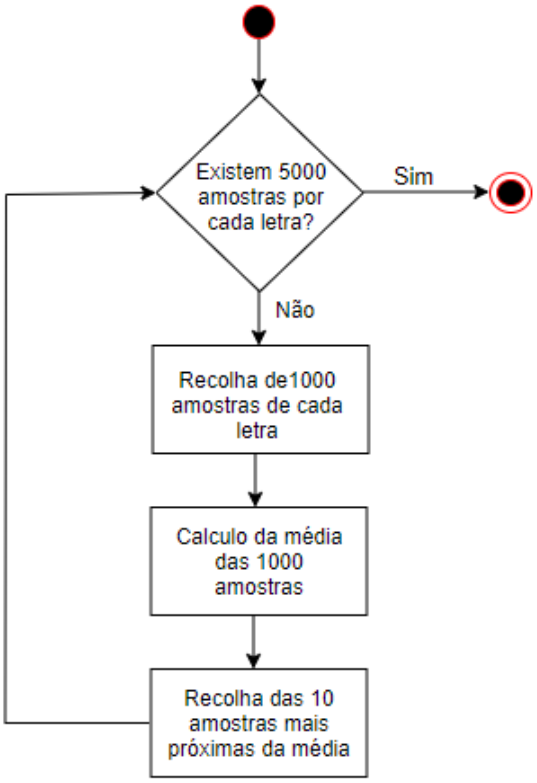


FIGURA 4.3: Fluxograma Recolha de dados

Após os dados serem recolhidos e guardados em ficheiros é importante preparar cada um desses ficheiros consoante o *Software* a utilizar. Para a classificação do gesto feito em tempo real foi utilizado o *Eclipse*. De forma a saber que valores caracterizavam cada letra, foi necessário dar ao programa um ficheiro com essa informação. A forma como estes dados se apresentam no programa diferiu entre os dois algoritmos.

No caso do k-NN é necessário classificar as amostras consoante a respetiva letra e juntar todas no mesmo ficheiro. Na Figura 4.4 é possível observar uma parte do set de dados da letra A e B.

```
285,767,564,845,841,A
281,764,559,844,841,A
287,767,562,845,841,A
283,761,562,844,841,A
287,763,562,841,831,A
285,767,562,845,843,A
287,763,562,845,841,A
326,790,892,863,792,B
326,790,892,863,792,B
326,790,892,863,792,B
320,790,892,876,792,B
323,792,886,876,792,B
326,786,892,863,792,B
```

FIGURA 4.4: Tratamento de dados K-nn

Nas Redes Neuronais a preparação dos dados não se processa da mesma forma, para este algoritmo cada letra tem o seu próprio ficheiro. Para cada letra as amostras respetivas às mesmas são marcadas como 1 no *output* e as restantes como 0, desta forma é dada a instrução para a rede neuronal apenas devolver 1 quando encontra a respetiva letra e 0 quando encontra outra letra. Na Figura 4.5 é possível observar uma parte do set de dados da letra A.

```
281,764,559,844,841,1
287,767,562,845,841,1
283,761,562,844,841,1
287,763,562,841,831,1
285,767,562,845,843,1
287,763,562,845,841,1
326,790,892,863,792,0
326,790,892,863,792,0
326,790,892,863,792,0
320,790,892,876,792,0
323,792,886,876,792,0
326,786,892,863,792,0
320,790,889,866,800,0
```

FIGURA 4.5: Tratamento de dados Redes Neuronais

4.3 Normalização dos Dados

Devido a variações nos valores dos sensores de flexão é necessário normalizar os valores de treino de cada letra, isto porque os valores que classificam uma letra num determinado momento podem ter ligeiras diferenças dos valores que vão classificar a mesma letra mais tarde, o que comprometeria a tradução. Temos como exemplo

a Figura 4.6 onde na Figura (A) os valores retirados num primeiro teste para definir a letra B são bastante mais altos que na Figura (B).

313,793,887,860,793	162,570,620,583,572
314,792,898,878,792	160,566,620,583,569
315,796,889,886,800	151,570,620,583,570
320,802,892,859,790	156,562,627,572,575
320,802,889,859,791	147,570,614,581,567
320,800,892,859,802	156,564,627,576,579

(A) Primeiro teste

(B) Décimo teste

FIGURA 4.6: Valores da letra B

Para normalizar estes valores foi necessário perceber qual o valor mais baixo e mais alto que os sensores de flexão apresentam no momento da classificação das letras. Para tal foi feita uma leitura dos valores dos dedos quando a mão estava esticada e quando a mão estava fletida. Para a mão esticada obteve-se os valores mínimos e para a mão fletida obteve-se os valores máximos destes sensores.

Uma vez que os sensores foram implementados de raiz é expectável que a janela de valores de cada um seja diferente. Para cada dedo foi medido um valor mínimo e um valor máximo no momento em que os dados de treino foram preparados:

- dedo polegar: [185, 702]
- dedo indicador: [217, 854]
- dedo médio: [335, 920]
- dedo anelar: [319, 914]
- dedo mindinho: [323, 912]

Para normalizar os valores foi utilizado o *Eclipse*. É necessário dar ao programa um ficheiro txt (Figura 4.7) onde estão guardados todos os valores correspondentes e a respetiva letra.

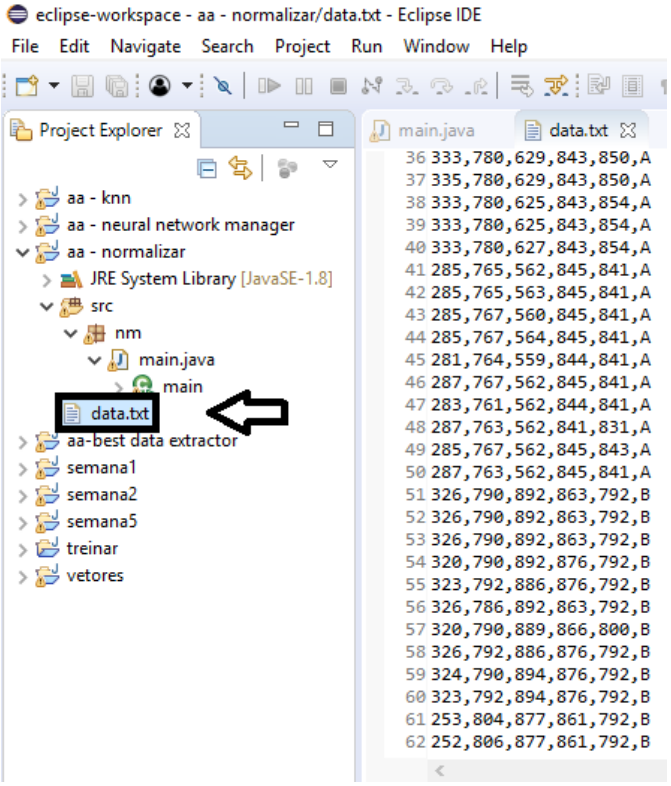


FIGURA 4.7: Ficheiro txt

No código é necessário dar a janela de valores de cada dedo, para tal foi guardado no programa dois vetores, o primeiro com os valores mínimos de cada dedo e o segundo com os valores máximos (Figura 4.8).

```
int[]minDedos = {185,217,335,319,323};
int[]maxDedos = {702,854,920,914,912};
```

FIGURA 4.8: Valores mínimos e máximos de cada dedo

O programa é então responsável por normalizar todos os valores segundo a Equação 4.2.

$$N' = \frac{valor - min}{max - min} \tag{4.2}$$

Onde os parâmetros estão definidos como:

- valor: valor atual do sensor de flexão
- max: valor máximo do sensor
- min: valor mínimo do sensor

É devolvido na consola os valores normalizados com a correspondente letra (Figura 4.9).

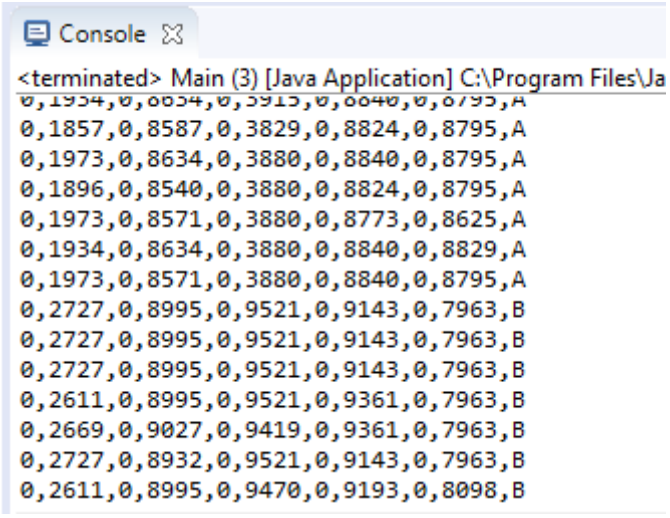


FIGURA 4.9: Valores normalizados

Para além da normalização dos dados de treino, também os dados recolhidos em tempo terão de ser normalizados. Para tal, cada vez que o utilizador usar a luva será necessário uma fase de calibração em que este estique e flita a mão de forma a retirar o valor máximo e mínimo de cada dedo, sendo feita de seguida a normalização dos valores. O objetivo é evitar erros de tradução no caso dos sensores apresentarem valores diferentes na classificação final e na classificação de treino.

Na Figura 4.10 está representado o fluxograma correspondente à fase da normalização dos dados.

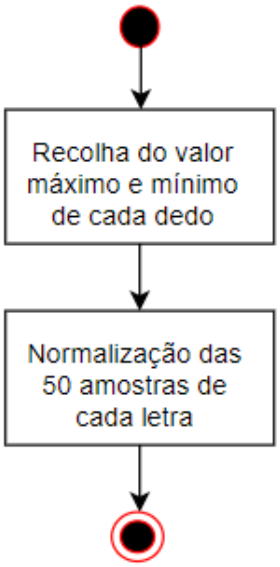


FIGURA 4.10: Fluxograma dados normalizados

4.4 Algoritmos

Depois de recolhidos e preparados os dados, foram utilizados dois algoritmos: o k-NN e as Redes Neuronais. Numa primeira fase os dois algoritmos classificam o gesto feito em tempo real com base nos sensores de flexão e numa segunda fase com base no acelerómetro, caso seja necessário.

4.4.1 Classificação final com base nos sensores de flexão

A classificação final das letras com base nos sensores de flexão comporta-se de forma diferente para cada um dos algoritmos.

4.4.1.1 k-NN

Para o k-NN foi desenvolvido um programa em *Java* que dado um set de dados previamente classificados, com base na distância Euclidiana, devolve as cinco amostras com maior proximidade do *input*. Este programa tem um comportamento

semelhante ao programa utilizado para retirar as 10 amostras mais próximas da média, uma vez que ambos se baseiam na distância Euclidiana.

O algoritmo k-NN começa por pedir os valores correspondentes ao gesto que o utilizador fez em tempo real, onde os valores correspondentes aos dedos são guardados no vetor de inteiros *input* e os valores correspondentes ao acelerómetro são guardados nas três variáveis X, Y e Z. (Figura 4.11).

```
// valores do gesto feito em tempo real
public static int input[] = {0,1102,0,7125,0,4882,0,6547,0,8762};
//valores do acelerómetro
public static int x = 7087;
public static int y = 5202;
public static int z = -1240;
```

FIGURA 4.11: Valores do gesto feito em tempo real

É então lido o ficheiro txt com os valores correspondentes a cada letra e feita a distância Euclidiana entre cada um desses valores e os valores dados na variável *input*. A função para calcular esta distância é a mesma utilizada na Equação 4.1.

Na Figura 4.12 está representado o fluxograma correspondente à classificação final com base nos sensores de flexão.

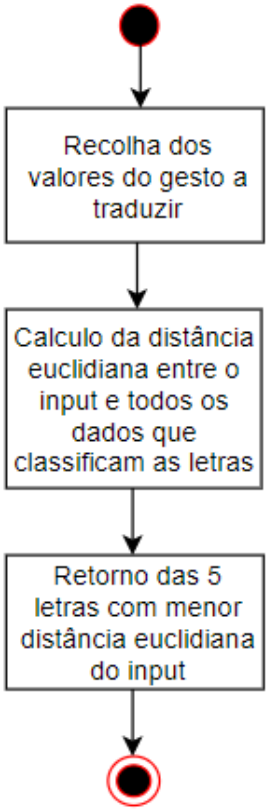


FIGURA 4.12: Fluxograma classificação com base nos sensores de flexão

4.4.1.2 Redes Neuronais

Para as Redes Neuronais foi utilizado o *Neuroph Studio* para treinar as várias redes, e desenvolvido um programa em *Java* capaz de gerir as várias redes e obter a tradução correta com base numa amostra de *input*.

No *Neuroph Studio*, foram dados cinco nós para a camada de *input*, cada um respetivo ao valor obtido do sensor de cada dedo. Dado que os cinco valores são igualmente importantes para definir cada letra, todos têm o mesmo peso no momento da decisão. Para a camada escondida foram definidos 10 nós, e para o *output* um nó.

Tanto os valores dos nós da camada escondida como do *output* correspondem à soma dos valores dos nós das camadas anteriores multiplicado pelo seu peso, passados por uma função de ativação. A função de ativação serve para introduzir

não linearidade, caso contrário, o nó de *output* era uma equação linear de somas de valores multiplicados por pesos.

Algumas das funções de ativação mais utilizadas são a de Sigmóide, Tangente Hiperbólica e Unidade Linear Retificada. A grande diferença entre as várias funções está no valor obtido como *output*.

A função Sigmóid foi a escolhida devido ao seu *output*, que varia entre 0 e 1. Desta forma é possível comparar o valor do *output* como uma taxa de acerto em que 0 corresponde a 0% e 1 a 100%. A letra que tiver a taxa de acerto mais próxima de 100% corresponde à tradução.

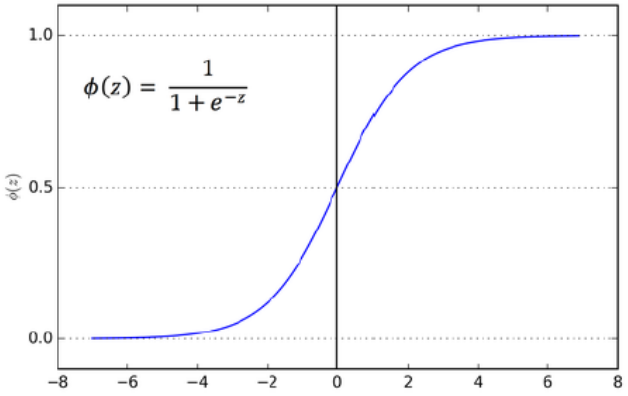


FIGURA 4.13: Função Sigmóide

A Figura 4.13 mostra o funcionamento da função Sigmóid. Sendo 'z' a soma dos valores das camadas anteriores multiplicados pelos respetivos pesos. A função tende para 1 quando 'z' tende para infinito e tende para 0 quando 'z' tende para menos infinito.

Foram então definidos 26 diferentes sets de dados (cada set correspondente a uma letra), onde 70% serviu para treinar e 30% para validar os resultados (Figura 4.14).

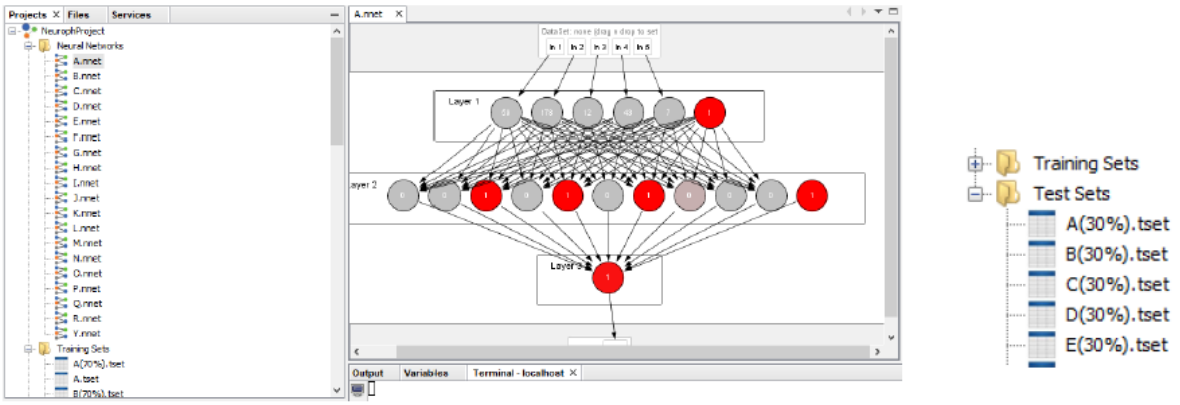


FIGURA 4.14: Tratamento de dados Redes Neuronais

Uma vez treinadas as várias redes, é agora esperado que seja possível distinguir todas as letras do alfabeto entre si.

Todos os ficheiros obtidos do *Neuroph Sutdio*, correspondentes ao treino das varias redes, são importados para o *Eclipse*.

Depois de obtidos os valores do gesto feito em tempo real estes são dados ao programa (Figura 4.15).

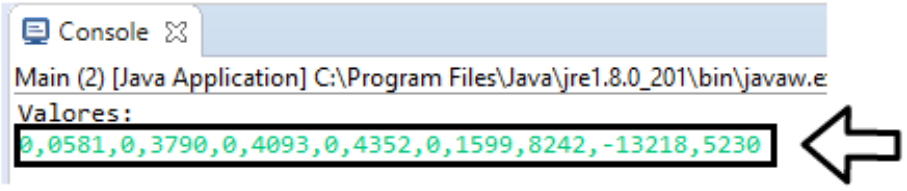


FIGURA 4.15: Valores do gesto feito em tempo real

São então lidos todos os ficheiros e são devolvidas todas as letras com o valor de acerto, em que a letra que devolver o valor mais próximo de 1 corresponde à tradução.

Caso a letra dada para tradução, tanto no k-NN como nas Redes Neuronais, não pertença ao caso em que existem outras letras com configuração semelhante a essa, a tradução é apenas dada com base nos sensores de flexão.

4.4.2 Classificação final com base no acelerómetro

As letras dos grupos A/R, B/Q/E, K/V, M/W, L/T e U/X têm configuração semelhante entre si, o que significa que os valores dos sensores de flexão vão ser semelhantes. Por essa razão a classificação final é feita com base não só nos sensores de flexão mas também no acelerómetro.

Na fase de classificação com base no acelerómetro, os dois algoritmos têm um comportamento igual. Ambos identificam qual a letra que foi dada como tradução com base nos sensores de flexão, e caso pertença a um dos grupos referidos em cima, a escolha é feita com base na coordenada de peso que diferencia as letras do mesmo grupo.

No caso do k-NN (Figura 4.11) os cinco valores dados no *input* tanto podiam corresponder à letra A como à letra R, uma vez que as duas letras têm configurações iguais. Neste caso, o algoritmo lê os valores do acelerómetro e dá a resposta correta com base neste parâmetro. No Capítulo 3, Secção 3.2, Tabela 3.4 foi percetível que as duas letras se distinguiam com base na coordenada Y, que caso fosse inferior a 6500 assumia como resposta a letra A e caso fosse superior a 6500 assumia como resposta a letra R. Observando o exemplo da Figura 4.11 é possível constatar que a coordenada Y tem valor inferior a 6500 o que leva o algoritmo a fazer a tradução correta para a letra A.

No final são então devolvidas as cinco letras com base na coordenada de peso (Figura 4.16).

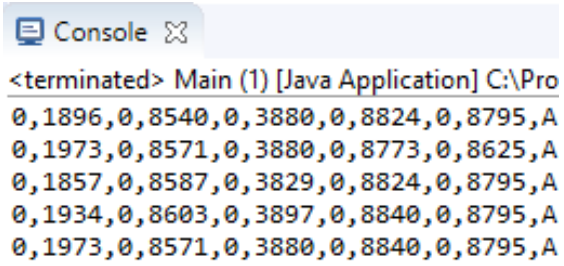


FIGURA 4.16: Letras devolvidas como tradução

As amostras que são devolvidas como tradução podem não ser todas iguais. Para tal considera-se como resposta correta a letra que for devolvida mais vezes entre as cinco amostras.

Na Figura 4.17 está representado o fluxograma correspondente à classificação com base no acelerómetro.

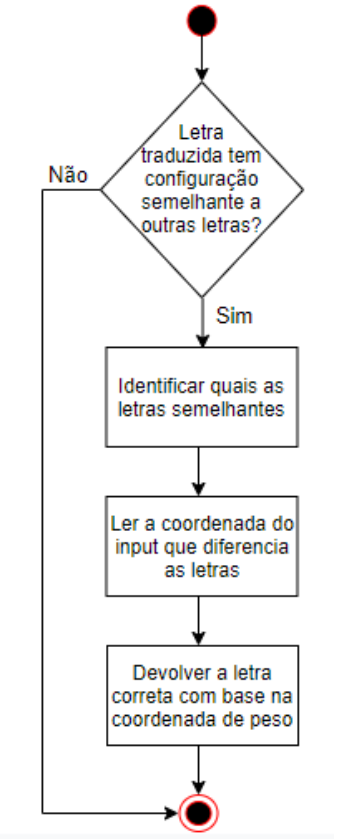


FIGURA 4.17: Fluxograma classificação com base no acelerómetro

Capítulo 5

Aplicação Móvel

Este capítulo descreve o funcionamento da aplicação móvel e de que forma deve ser utilizada.

5.1 Aplicação *Android*

A escolha da aplicação móvel visa assegurar a utilização do sistema no exterior nos diversos contextos em que se move uma pessoa surda.

A aplicação móvel desenvolvida tem como objetivo mostrar a tradução da LGP para Língua Portuguesa. O algoritmo implementado na aplicação móvel para a tradução é o k-NN uma vez que obteve uma taxa de acerto para a tradução do alfabeto bastante superior às Redes Neuronais, como será possível observar no capítulo seguinte.

Os sistemas operativos mais utilizados atualmente são o *Android* e o *IOS*. O *Android* é um sistema baseado em *Linux* desenvolvido pela *Google*, desenhado para dispositivos como telemóveis e *tablets*. O *IOS* é um sistema operativo da *Apple* que apenas opera em *Iphone* e *Ipad*. Ao compararmos características como a percentagem de uso de cada um dos sistemas, é notório que o mais utilizado pela população Portuguesa é o *Android*, como mostra a Figura 5.1 [53]. Por esse

motivo, o sistema desenvolvido terá vertente para telemóveis que tenham sistema *Android*, uma vez que desta forma é possível chegar a mais cidadãos portugueses.

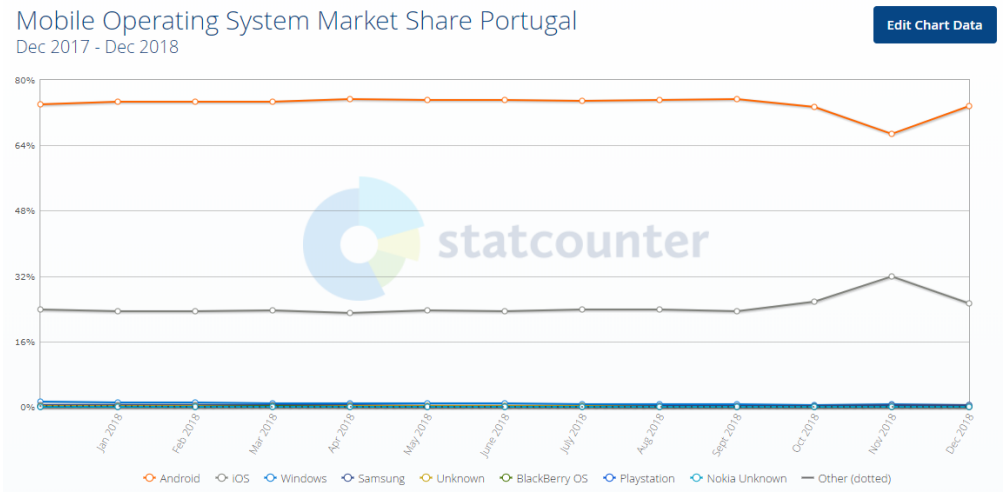


FIGURA 5.1: Percentagem de utilização de cada Sistema Operativo Móvel em Portugal [53]

Para desenvolver a aplicação foi utilizado o IDE oficial (*Android Studio*) do sistema operativo *Android* da *Google* e a linguagem de programação utilizada foi *Java*. No site oficial *Android* [54] foi possível encontrar informação correspondente a cada uma das versões que o *Android* disponibiliza assim como a sua taxa de utilização. Na Figura 5.2 está representada essa informação, relativa ao estudo completado a 12 de Julho de 2019. Pode-se constatar que a versão 6.0 (*Marshmallow*) é a mais utilizada. Por esse motivo foi a versão utilizada para desenvolver a aplicação*Android*.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

FIGURA 5.2: Versões *Android* mais utilizadas [54]

5.2 Estrutura da Aplicação Móvel

A aplicação *Android* pode ser utilizada por qualquer pessoa que queira estabelecer conexão entre a luva e o dispositivo móvel.

Na Figura 5.3 está representado o diagrama de atividade correspondente à aplicação móvel.

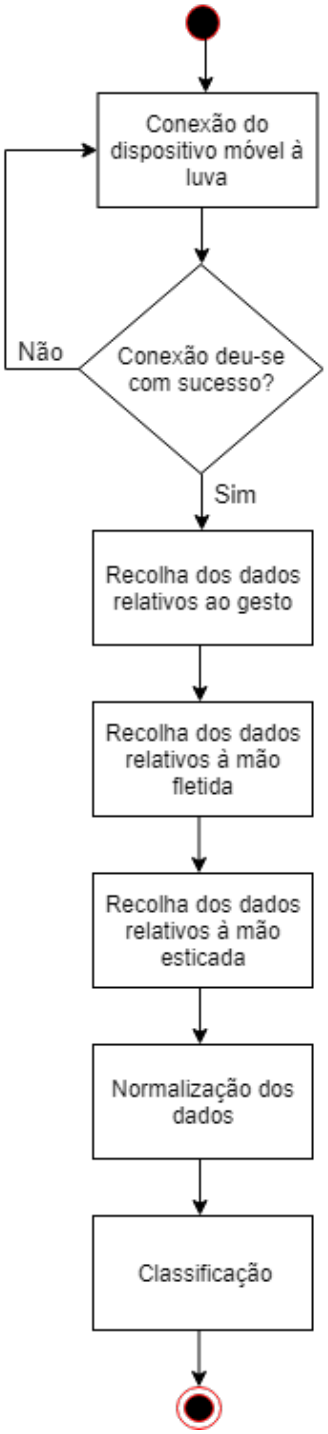


FIGURA 5.3: Diagrama de atividade da aplicação móvel

Depois de uma conexão com sucesso entre o dispositivo móvel e a luva, o utilizador faz o gesto que pretende traduzir. É necessário que o utilizador flita e feche a mão de forma a tirar o valor mínimo e máximo que os sensores apresentam (fase de calibração) para posterior normalização dos dados. Todo este processo ocorre

na luva e posteriormente os dados recolhidos aqui, são enviados para a aplicação móvel através do *Bluetooth*. Quando a aplicação recebe todos os dados necessários, faz a classificação final da letra com base no algoritmo k-NN, classificação esta que é apresentada no ecrã do dispositivo móvel.

5.3 Design e Implementação da Aplicação Móvel

A aplicação que suporta o sistema foi desenvolvida de raiz. De forma a que o utilizador tenha a melhor experiência durante a sua utilização, os padrões de *Android f*oram utilizados para criar e desenvolver a interface gráfica do utilizador (GUI).

As principais características da aplicação estão descritas nesta secção. A Figura 5.4 mostra o logótipo da aplicação.

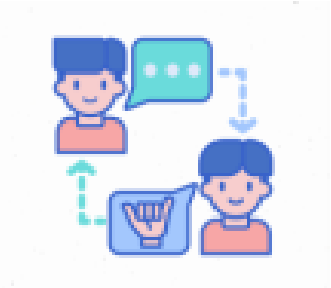


FIGURA 5.4: Logótipo da aplicação móvel

Conexão da luva com o dispositivo móvel

Os utilizadores têm de conectar a luva ao dispositivo móvel de forma a poder utilizar a aplicação. Depois de ligado o *Bluetooth* do dispositivo móvel (Figura 5.5), o utilizador pode descobrir a luva ao qual se pretende conectar (botão Descobrir luva) ou ligar-se a uma luva que já foi anteriormente conectada aquele dispositivo móvel (botão Luvas emparelhadas).

FIGURA 5.5: *Bluetooth* do dispositivo móvel ligado

Na Figura 5.6 está representada a atividade da conexão.

FIGURA 5.6: Conexão entre o dispositivo móvel e a luva

+

Feita a conexão com sucesso, é possível recolher os dados necessários dos vários sensores da luva que são enviados diretamente para a aplicação.

Recolha dos dados

Para ser feita a tradução do gesto, é necessário recolher três tipos de dados: o gesto feito em tempo real, o valor máximo de cada dedo e o valor mínimo de cada dedo.

O utilizador começa por executar o gesto que pretende traduzir (botão Recolher). Os valores relativos ao gesto são recolhidos e apresentados no campo "Gesto"da aplicação. Na Figura 5.7 está representado o exemplo de um gesto recolhido em tempo real. O gesto efetuado corresponde à letra D.

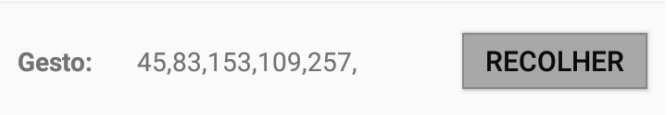


FIGURA 5.7: Gesto a traduzir

Após a recolha dos primeiros dados, o utilizador flete (botão Máximo) e estica a mão (botão Mínimo) de forma a tirar o valor máximo e mínimo de cada dedo, correspondentemente. Os respetivos valores são guardados e utilizados na normalização dos dados (botão Normalizar).

Na Figura 5.8 estão representados os três botões relativos à recolha dos dados descritos anteriormente e à sua normalização.

FIGURA 5.8: Botões da aplicação móvel

Tradução

Depois de recolhidos todos os dados é feita a tradução (botão Traduzir) com base no algoritmo k-NN. É lido o ficheiro guardado na aplicação (Figura 5.9) que contém todos os dados que caracterizam cada letra do alfabeto. O algoritmo, com base na distância Euclidiana, devolve então a letra correspondente ao gesto.

No capítulo 4, foi descrito que o algoritmo k-NN devolvia as cinco letras mais próximas do *input* dado. Na aplicação móvel também são identificadas as cinco letras, no entanto, é devolvida apenas a letra que aparece mais vezes.

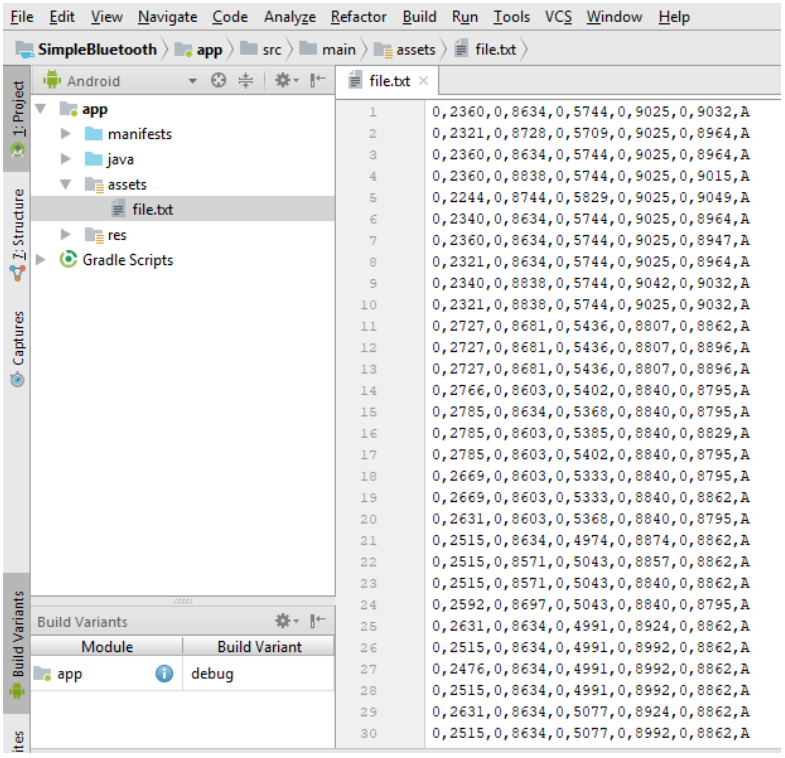


FIGURA 5.9: Ficheiro da aplicação móvel

A letra traduzida é apresentada no campo "Tradução". Na Figura 5.10 está representada a tradução da letra correspondente ao gesto efetuada na Figura 5.7.

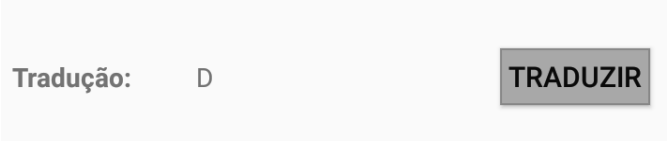


FIGURA 5.10: Tradução do gesto feito em tempo real

Capítulo 6

Resultados

Neste capítulo é apresentado o protótipo da luva assim como os resultados obtidos com o sistema desenvolvido. Os testes efetuados ao sistema foram divididos em duas partes:

- testes desenvolvidos com a luva
- testes desenvolvidos com os algoritmos

6.1 Testes desenvolvidos com a luva

Os testes desenvolvidos com a luva servem para confirmar que todos os componentes que foram cosidos estão a funcionar corretamente. Os componentes fundamentais da luva foram então testados individualmente e no sistema final.

De seguida seguem-se os testes efetuados aos sensores de flexão, ao acelerómetro e ao *Bluetooth*. Uma vez que se tratam de testes de validação foram efetuados apenas por uma pessoa.

Validação dos sensores de flexão

Antes destes sensores serem cosidos na luva e a cada um dos seus pins correspondentes no *LilyPad USB*, foram testados com o *Arduino UNO* (Figura 6.1),

desta forma evitou-se potenciais erros que só poderiam ser notados depois da luva estar parcialmente cosida.

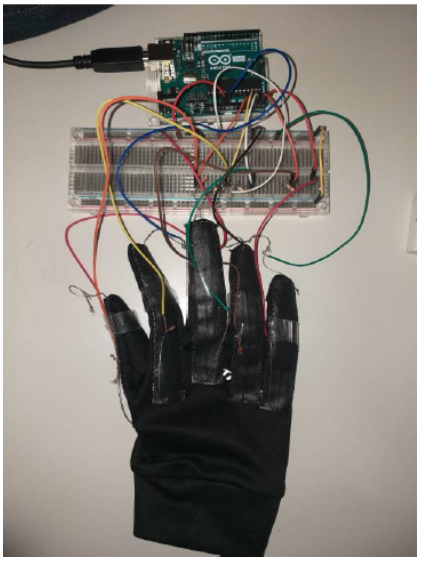


FIGURA 6.1: Protótipo com o *Arduino UNO*

Os sensores foram colados com fita-cola a cada dedo e cada letra foi testada três vezes. Apesar deste protótipo não dar grande estabilidade aos sensores, permitiu verificar que estes apresentavam valores semelhantes para cada letra.

A placa *LilyPad USB* dispõe de um conversor analógico-digital de 10 bits, o que significa que este irá estruturar tensões entre 0 e a tensão operacional de 3.3V para valores inteiros entre 0 e 1023.

Na Tabela 6.1 é possível observar os valores obtidos para a letra A em cada um dos três testes.

1º teste	**2º teste**	**3º teste**
(313,765,647,845,841)	(327,759,658,845,831)	(340,770,638,831,841)

TABELA 6.1: Valores para a letra A

Depois da validação dos sensores estes foram cosidos juntamente com o *LilyPad USB* na luva.

Validação do acelerómetro

O acelerómetro foi cosido no dorso da mão e testado com a rotação da mesma. Foi verificado que para cada posição as coordenadas X, Y e Z adquiriam valores expectáveis consoante o eixo representado no próprio acelerómetro.

Quando a palma da mão está voltada para baixo (Figura 6.2) os valores obtidos foram positivos para o eixo X e negativos para os eixos Y e Z.

FIGURA 6.2: Palma da mão voltada para baixo

Quando a palma da mão está voltada para cima (Figura 6.3) os valores obtidos foram negativos para o eixo X e positivos para os eixos Y e Z.

FIGURA 6.3: Palma da mão voltada para cima

Quando a palma da mão está voltada para a esquerda (Figura 6.4) os valores obtidos foram positivos para os eixos X, Y e Z.

FIGURA 6.4: Palma da mão voltada para a esquerda

Quando a palma da mão estava voltada para a esquerda (Figura 6.5) os valores obtidos foram negativos para os eixos X, Y e Z.

FIGURA 6.5: Palma da mão voltada para a direita

Validação do *Bluetooth*

Como referido anteriormente foi necessário coser o *LilyPad Xbee* à luva de forma a ser feita a comunicação entre o *Bluetooth* e o *LilyPad USB*. Para testar se as ligações estavam corretas o *LilyPad USB* foi alimentado através do cabo USB, e foi utilizada a aplicação "*Serial Bluetooth Terminal*" [44] para emparelhar o *Bluetooth* do dispositivo móvel com o *Bluetooth* da luva.

Na Figura 6.6 é possível observar que o emparelhamento deu-se com sucesso.

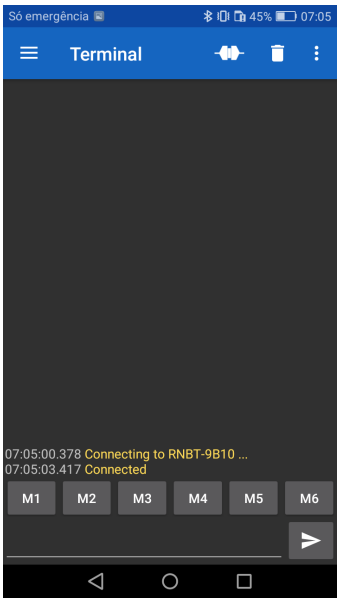


FIGURA 6.6: Conexão do *Bluetooth* da luva à aplicação móvel

6.2 Testes desenvolvidos com os algoritmos

Para testar os dois algoritmos desenvolvidos, o k-NN e as Redes Neuronais, foram feitos 800 testes de validação para cada letra. Cada vez que um gesto é realizado, os valores correspondentes são testados em cada um dos algoritmos.

Os testes realizados com os algoritmos foram efetuados por duas pessoas. A primeira pessoa tinha conhecimento na LGP e foi a mesma que que caracterizou as várias letras do alfabeto nas três primeiras fases de amostragem, como referido no capítulo 4. A segunda pessoa não tinha conhecimento na LGP e não foi a mesma que caracterizou as várias letras do alfabeto nas duas ultimas fases de amostragem, como referido no capítulo 4. Desta forma foi possível verificar se o sistema é igualmente eficaz para estes dois grupos.

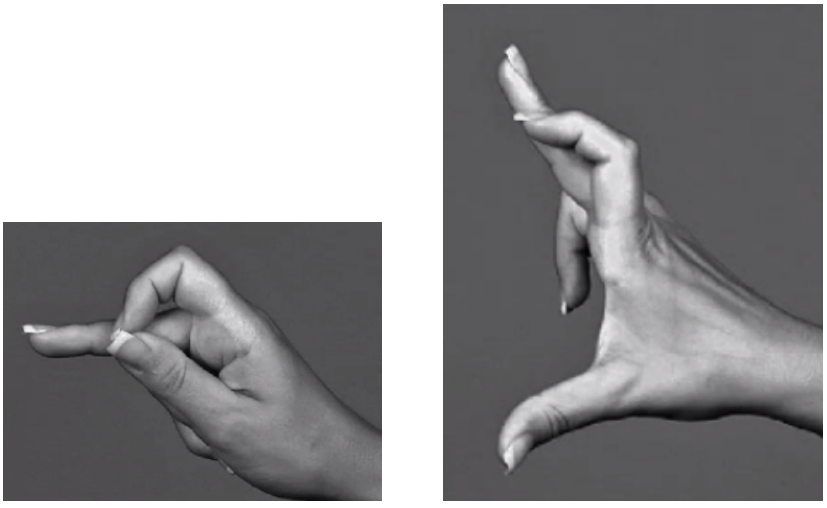
Como foi visto anteriormente existem letras que são classificadas apenas com base nos sensores de flexão enquanto que outras são classificadas com base nos sensores de flexão juntamente com o acelerómetro. Desta forma os testes efetuados

com os algoritmos foram divididos em três partes: os testes com os sensores de flexão, os testes com o acelerómetro e os testes com os dois componentes.

6.2.1 Testes feitos aos dois algoritmos com os sensores de flexão

Para testar os algoritmos com os sensores de flexão foram efetuados 200 testes por uma pessoa com conhecimento em LGP e 200 testes com uma pessoa sem conhecimento na LGP.

Durante esta fase o acelerómetro não teve peso na decisão da letra, sendo apenas utilizados para a tradução os valores dos sensores de flexão. Para tal as letras que apenas se distinguem com base no acelerómetro foram agrupadas e testadas como uma só. Assim para os grupos de letras A/R, B/Q/E, K/V, M/W, L/T e U/X, que têm configuração igual, os resultados foram considerados corretos desde que a resposta correspondesse a uma das letras do próprio grupo. Como exemplo deste caso a letra A e R (Figura 6.7), que por ter configuração igual foi considerada correta a tradução R mesmo que o gesto executado fosse o A, e vice-versa.



(A) Letra A (B) Letra R

FIGURA 6.7: Letras do alfabeto da LGP [7]

Cada letra foi testada 200 vezes à exceção dos grupos A/R, K/V, M/W, L/T e U/X onde foram feitos 100 testes para cada letra do grupo, já que para o algoritmo estes grupos são vistos como uma letra só. Para o grupo B/Q/E foram feitos 66 testes para a letra B e Q e 68 testes para a letra E.

A Tabela 6.2 corresponde aos testes feitos por uma pessoa com conhecimento na LGP, para os dois algoritmos. Para o k-NN foi obtida uma média total da taxa de erro de 6.05% enquanto que para as Redes Neuronais essa taxa foi de 21.28%.

Letra	**K-NN**	**Redes Neuronais**
A/R	98%	60%
B/Q/E	97%	78%
C/O	85%	84%
D	100%	90%
F	96%	72%
G	90%	100%
H	88%	62%
I	91%	54%
J	91%	53%
K/V	99%	80%
L/T	100%	75%
M/W	100%	100%
N	88%	84%
P	90%	71%
S	95%	100%
U/X	92%	75%
Y	100%	87%
Z	91%	92%

Tabela 6.2: Percentagem de acerto de cada letra

A Tabela 6.3 corresponde aos testes feitos por uma pessoa sem conhecimento na LGP, para os dois algoritmos. Para o k-NN foi obtida uma média total da taxa de erro de 8.11% enquanto que para as Redes Neuronais essa taxa foi de 27.06%.

A Tabela 6.2 e a Tabela 6.3 mostram que o algoritmo k-NN foi mais eficaz na tradução da LGP, independentemente de quem utiliza o sistema.

Ao comparar os resultados obtidos para uma pessoa com conhecimento na LGP e uma pessoa sem conhecimento na LGP, em ambos os algoritmos, verifica-se que

Letra	K-NN	Redes Neuronais
A/R	91%	50%
B/Q/E	92%	58%
C/O	80%	77%
D	100%	82%
F	99%	64%
G	88%	100%
H	85%	64%
I	89%	48%
J	93%	42%
K/V	95%	71%
L/T	97%	78%
M/W	80%	91%
N	85%	73%
P	88%	84%
S	92%	100%
U/X	93%	68%
Y	94%	79%
Z	93%	90%

Tabela 6.3: Percentagem de acerto de cada letra

a taxa de erro é bastante mais baixa para uma pessoa com conhecimento na LGP, que consegue obter um melhor desempenho.

6.2.2 Testes feitos aos dois algoritmos com o acelerómetro

Nesta fase a escolha da tradução recai apenas sobre o acelerómetro. Apenas irão ser testadas as letras que precisam do acelerómetro para ser distinguidas entre si. Tem-se então as letras A, R, B, Q, E, K, V, M, W, L, T, U e X.

Para os testes com o acelerómetro foi utilizado o conjunto de dados que nos testes com os sensores de flexão foi traduzido corretamente. Na Tabela 6.2, no caso da validação do A/R, dos 200 testes feitos apenas 98% dos dados foram identificados como sendo a letra A ou R. Esses dados, que correspondem à letra A e R, foram testados e validados apenas com base no acelerómetro. Seria incorreto utilizar os outros 2% dos dados, uma vez que não sendo validados como A ou R já têm a tradução comprometida.

Uma vez que tanto o k-NN como as Redes Neuronais classificam o gesto com base no acelerómetro da mesma forma, não ouve distinção dos resultados entre os dois algoritmos uma vez que será igual para ambos.

Na Tabela 6.4 estão os testes correspondentes a uma pessoa com conhecimento na LGP, baseados na Tabela 6.2.

Letra	**k-NN/Redes Neuronais**
A	100%
R	100%
B	100%
Q	100%
E	100%
K	100%
V	100%
L	100%
T	100%
M	100%
W	100%
U	100%
X	100%

Tabela 6.4: Percentagem de acerto de cada letra

Na Tabela 6.5 estão os testes correspondentes a uma pessoa sem conhecimento na LGP, baseados na Tabela 6.3.

Letra	**k-NN/Redes Neuronais**
A	100%
R	100%
B	100%
Q	100%
E	100%
K	100%
V	100%
L	100%
T	100%
M	100%
W	100%
U	100%
X	100%

Tabela 6.5: Percentagem de acerto de cada letra

No caso em que o único peso para a tradução é o acelerómetro verifica-se que a classificação final é bastante eficaz tanto para uma pessoa com conhecimento

na LGP como para uma pessoa sem conhecimento na LGP, havendo uma taxa de erro de 0% em ambos os casos.

A eficácia do sistema deve-se à forma como o acelerómetro é validado. Ao contrário dos sensores de flexão que são validados com base em valores, o acelerómetro é validado com base no sinal das coordenadas, o que faz com que o erro diminua drasticamente.

6.2.3 Testes feitos aos dois algoritmos com os sensores de flexão e com o acelerómetro

Depois de testados os algoritmos com os sensores de flexão e com o acelerómetro individualmente, os dois algoritmos foram testados também com os sensores e o acelerómetro em conjunto.

Nesta fase, em que o sistema foi testado como um só, foram feitos 200 testes por uma pessoa com conhecimento na LGP e 200 testes por uma pessoa sem conhecimento na LGP.

Na Tabela 6.6 estão representadas as taxas de acerto referentes ao sistema final, relativas a uma pessoa com conhecimento na LGP. Para o k-NN foi obtida uma média total da taxa de erro de 6.69% enquanto que para as Redes Neuronais essa taxa foi de 21.73%.

Na Tabela 6.7 estão representadas as taxas de acerto referentes ao sistema final, relativas a uma pessoa sem conhecimento na LGP. Para o k-NN foi obtida uma média total da taxa de erro de 10.34% enquanto que para as Redes Neuronais essa taxa foi de 28.84%.

As taxas de acerto obtidas com as Redes Neuronais para o sistema final, tanto para uma pessoa com conhecimento na LGP como para uma pessoa sem conhecimento na LGP foram bastante fracas, 78.27% e 71.16% respetivamente. Uma explicação para estes resultados poderá ser o número de *inputs* dados nas Redes. Quanto maior for o número de *inputs* mais eficiente este algoritmo será e neste

Letra	**k-NN**	**Redes Neuronais**
A	87%	64%
B	100%	84%
C	94%	86%
D	100%	87%
E	95%	74%
F	97%	72%
G	100%	98%
H	85%	69%
I	97%	51%
J	93%	48%
K	92%	84%
L	90%	83%
M	80%	87%
N	85%	82%
O	97%	89%
P	90%	76%
Q	93%	84%
R	93%	72%
S	90%	91%
T	100%	70%
U	97%	72%
V	94%	79%
W	90%	92%
X	93%	78%
Y	96%	81%
Z	98%	85%

Tabela 6.6: Percentagem de acerto de cada letra

caso a rede apenas tinha cinco *inputs* em que cada um correspondia a um dedo da mão.

Ao analisar os resultados obtidos para o algoritmos k-NN quando testado o sistema final, constata-se que a média total da taxa de erro foi baixa, 6.69% para uma pessoa com conhecimento em LGP e 10.34% para uma pessoa sem conhecimento em LGP. Isto traduz-se numa média total de taxa de acerto de 93.31% e 89.66% respetivamente.

Dado que o sistema tem a finalidade de ser utilizado por uma pessoa com conhecimento em LGP, considera-se a taxa obtida bastante encorajadora.

Letra	**K-nn**	**Redes Neuronais**
A	92%	52%
B	95%	54%
C	87%	75%
D	100%	80%
E	94%	64%
F	93%	58%
G	85%	97%
H	78%	68%
I	90%	52%
J	83%	50%
K	87%	68%
L	97%	82%
M	94%	84%
N	81%	69%
O	83%	70%
P	78%	82%
Q	84%	52%
R	92%	62%
S	92%	95%
T	96%	82%
U	87%	64%
V	84%	66%
W	91%	87%
X	85%	72%
Y	89%	71%
Z	94%	84%

Tabela 6.7: Percentagem de acerto de cada letra

No Capítulo 2, Secção 2.2.1, foram referidos alguns projetos ligados à tradução da Língua Gestual baseados em sensores de vídeo. Thad Starner et al. [13], descreveu um sistema para a tradução da da Língua Gestual America onde obteve uma taxa de acerto de 92% quando a câmara se encontra na mesa e de 98% quando a câmara se encontra no chapéu do utilizador. Já Simon Lange et al [14], descreve um sistema de tradução de Língua Gestual Australiana baseado no *Kinect*, obtendo uma taxa de precisão de 97%.

Ambos os projetos obtiveram uma taxa de acerto maior que a do projeto aqui desenvolvido, mostrando que as câmaras continuam a ser a melhor opção para

sistemas de tradução de Língua Gestual. No entanto, maior parte destes sistemas têm a grande desvantagem de não serem portáteis uma vez que envolvem o uso do computador, o que faz com que não possam ser utilizados em tarefas quotidianas que envolvam o exterior.

No Capítulo 2, na Secção 2.2.2 e na Secção 2.5.1, também foram referidos projetos ligados à tradução da Língua Gestual baseados em sensores de movimento. Kalpattu S. Abhishek et al. [30] desenvolveu um projeto para a tradução da Língua Gestual Americana baseado em sensores capacitivos, obtendo uma taxa de acerto de 92%. Kim et al. [19] desenvolveu um sistema baseado num sensor EMG e obteve uma taxa de precisão de 94% e Jian Wu et al. [18] desenvolveu um sistema de traduão de Língua Gestual Americana baseado num giroscópio e num acelerómetro onde obteve uma taxa de acerto de 97.89%.

Ao comparar o resultado obtido com o sistema desenvolvido com os resultados dos sistemas descritos anteriormente, conclui-se que as taxas de acerto são bastante semelhantes. No entanto os sistemas anteriores não tem uma interface gráfica móvel onde é possível ver a tradução do gesto em tempo real.

Capítulo 7

Conclusões e Trabalho Futuro

Neste capítulo estão descritas as principais conclusões assim como o trabalho futuro que pode ser desenvolvido.

7.1 Conclusões

Foi desenvolvido e implementado um sistema *wireless* e *low-cost* para traduzir o alfabeto da Língua Gestual Portuguesa, principal objetivo desta dissertação. Este sistema é composto por uma luva com *hardware* colocado estrategicamente, um algoritmo capaz de assegurar a tradução do alfabeto e uma aplicação móvel que disponibiliza o texto traduzido em tempo real. Foi possível obter dados relativos à posição dos dedos e à orientação da palma da mão que classificam cada gesto. A comunicação entre a luva e o dispositivo móvel foi assegurada com o protocolo mais adequado, o *Bluetooth*. Dos dois algoritmos implementados para assegurar a tradução, o k-NN destacou-se pelas sua baixa média total da taxa de erro, sendo por isso o algoritmo escolhido para implementar na aplicação móvel.

Foi possível tirar conclusões sobre o *hardware* utilizado no sistema. O *LilyPad USB* foi uma ótima escolha para assegurar o funcionamento do sistema, na medida em que a sua implementação foi acessível e que o seu funcionamento correto foi

assegurado durante todo o processo, bem como o consequente processamento de dados.

O acelerómetro LSM303 3D foi fundamental para o bom funcionamento do sistema. Ao ser utilizado em conjunto com os sensores de flexão, o acelerómetro foi responsável por diferenciar 13 letras com configuração semelhante entre si. Sem a utilização deste *hardware* a percentagem de acerto destas 13 letras nos testes feitos ao sistema final seria muito baixa.

Os sensores de flexão baseados em *velostat* foram o *hardware* menos confiável do sistema. Apesar de ter sido possível utilizar os valores destes sensores para a tradução da Língua Gestual Portuguesa, os mesmo estavam em constante mudança. Por esse motivo foi necessário implementar uma funcionalidade extra no sistema, a normalização dos dados. Com esta normalização foi então possível obter valores constantes para a caracterização das letras, no entanto seria muito mais simples se o sistema não necessitasses desta funcionalidade extra.

O *Bluetooth* foi a escolha acertada para protocolo de comunicação entre o *LilyPad USB* e a aplicação móvel. Para além de ser fiável a nível de transferência de dados, este protocolo não limita o utilizador ao uso do sistema apenas na presença de *Internet*.

Depois de vários testes, é possível concluir que o material utilizado no sistema, mais precisamente os componentes eletrónicos, são fiáveis.

Foi também possível tirar conclusões a nível do *software* do sistema. O algoritmo Redes Neuronais não foi uma boa escolha para a tradução, uma vez que este algoritmo é tão mais eficaz quanto maior for o número de parâmetros que caracterizam os dados, que neste caso eram apenas cinco. Inicialmente os dados seriam caracterizados por oito parâmetros, no entanto devido à instabilidade do acelerómetro estes valores não puderam ser incluídos nos parâmetros dos dados. Por outro lado, o algoritmo k-NN foi uma escolha bastante eficaz para a tradução.

A aplicação móvel, apesar de bastante simples, foi fundamental para o sistema uma vez que permitiu que este fosse portátil e pudesse ser utilizados no quotidiano do utilizador.

Um aspeto importante a ter em conta é que a luva deve ter o tamanho correto da mão do utilizador, caso contrário os sensores de flexão podem não estar em contacto com os dedos do utilizador.

No geral o sistema funciona corretamente e de acordo com o objetivo proposto. O produto poderia ser utilizado no futuro pela comunidade surda em situações onde não existe apoio para esta comunidade.

7.2 Trabalho Futuro

O sistema desenvolvido pode ser otimizado no futuro, com o desenvolvimento dos pontos que a seguir se enunciam:

- **Implementação de novos sensores:** Foi estudado que existem cinco parâmetro comuns a todas as Línguas Gestuais sendo eles a configuração da mão, a orientação, o movimento, a localização e a expressão facial e corporal. Com o sistema desenvolvido foi possível dar cobertura a dois destes parâmetro, a configuração da mão e a orientação. Como trabalho futuro deveria ser implementado um sensor que permitisse detetar a localização do gesto efetuado.

  Os sensores de flexão baseados em *velostat* deveriam ser substituídos por sensores de flexão comprados, dando estabilidade aos valores que definem as letras.

- **Algoritmo:** Neste momento o sistema apenas está preparado para gestos estáticos. No futuro o algoritmo deveria ser capaz de distinguir gestos estáticos e dinâmicos, ou seja, o algoritmo seria capaz de reconhecer o início e fim do gesto. Este campo diz respeito ao parâmetro movimento da Língua Gestual.

- **Células solares fotovoltaicas para têxteis:** Estas células podem ser adicionadas ao sistema para uma maior duração da bateria do sistema. Isto permite que a bateria seja carregada ao mesmo tempo que está a ser utilizada [55].

- ***Site Web*:** Para completar a aplicação móvel seria interessante o utilizador ter um site *web* com a mesma função que a aplicação móvel. Este poderia ser utilizado em casa uma vez que um computador não é tão fácil de transportar como um telemóvel. Para além da tradução o site *web* poderia ter outras funções como uma escola de Língua Gestual Portuguesa, onde a comunidade não surda pudesse aprender esta Língua.

Anexos

Anexo A

Artigos Científicos Publicados

Mobile Hand Gesture Recognition System for the Portuguese Sign Language

Sara Ferreira
Dept. of Science and Information Technology
ISCTE-University of Lisbon and Instituto de Telecomunicaes
Lisbon, Portugal
scmfa@iscte-iul.pt

Nuno Souto
Dept. of Science and Information Technology
ISCTE-University of Lisbon and Instituto de Telecomunicaes
Lisbon, Portugal
nuno.souto@iscte-iul.pt

Octavian Postolache
Dept. of Science and Information Technology
ISCTE-University of Lisbon and Instituto de Telecomunicaes
Lisbon, Portugal
octavian.adrian.postolache@iscte-iul.pt

***Abstract*—Gesture Recognition is a fundamental technology nowadays. Not only is it applied in games and medicine, but also in Sign Language Translation. This tool assists the deaf community since it allows them to communicate with people who do not understand Sign Language. This paper describes the development of an automatic Translator of Portuguese Sign Language (PSL) which comprises a glove that is supported on wireless sensors, an android application and a machine learning algorithm to associate a gesture to a letter. A survey of all the related work is described, followed by a description of the architecture that is being implemented for this project.**

***Index Terms*—Portuguese Sign Language, Wireless System, Gesture Recognition, Human Computer Interaction, Bluetooth, Machine Learning, Android.**

I. Introduction

Nowadays, technology simplifies and increases the services quality provided, but despite its constant advance, the conditions provided to the deaf community are far from the best, since the most widely used form of communication between a deaf person and a non-deaf person is by written text, which is not very efficient.

There are approximately 100.000 Portuguese who suffer from hearing loss [1]. This translates into 100.000 people who face daily difficulties in simple quotidian tasks involving interaction with a non-deaf person. In addition, the younger age group of this community still finds education difficulties, which translates into a slower linguistic and social development in relation to the students who listen [2].

Even though Portuguese Sign Languauge (PSL) is one of the official Portuguese Languages since 1997 [3], its integration in the society is far from perfect. Although there are some projects connected to PSL translation these are not portable, involving a Kinect camera [2], [4], which limits the use of the system outside a home.

In order to solve this communication barrier, it is the objective of this work, the development of a portable system capable of translating PSL using a sensor based glove.

Advanced human-computer interaction (HCI) is under the spotlight these days. The gesture recognition is an important step in that direction, since gestures made by the user are recognized by the machine [5]. The gesture recognition, can be divided into five stages: Data Acquisition, Pre-Processing, Segmentation, Feature Extraction and Classification.

Combining these elements it is possible to create a system that recognizes PSL. The proposed system will be supported on a low-cost wireless glove, an android applcation and employing Machine Learning algorithms. Together they will be able to associate PSL to Portuguese Language.

II. Related Work

There are already Gesture recognition systems using Sign Language, however, most parts of these systems are applied only for American Sign Language [6].

Sign Language recognition can be done in many ways, so the various approaches will be separated into two areas, those that use video-based sensors and those that use motion-based sensors [7]. The big difference from one approach to the other lies in data acquisition [8].

The systems that rely on video-based sensors have the advantage of not requiring the use of accessories. The images collected in this approach can be done by single cameras, stereo-cameras, active techniques and invasive techniques [8].

Reference [6] proposes a system based on Kinect, wich is a depth camera, that makes 3D-reconstruction in real time. After the camera detects and tracks the hand it collects the data, that is classified using Markov models. In the first results it was possible to obtain an accuracy rate of 97%, showing that depth cameras are effective for Sign Language recognition.

Reference [9] describes a system based on Leap Motion for Australian Sign Language. Leap Motion was shown to be capable of providing accurate tracking of hands, fingers, and movements. The disadvantages of this system is that Leap Motion can not capture the finger positions that are hidden from the viewpoint of the camera and when two fingers are

brought together the detection fails. Both [6] and [9] make part of the active techniques.

The systems that rely on motion-based sensors have many advantages when compared to video-based sensors such as: direct linking with the user; more coverage provided; not influenced by the environment; and most important is a wireless system. The data in this approach can be collected by an inertial measurement unit (IMU), electromyography (EMG), wifi and radar, and others like flex sensors and haptic technologies [7].

The system presented in [10] uses a glove with flex sensors and an IMU to track the motion of the hand int three-dimensional spaces. The purpose was to convert Indian Sign Language to voice output.

Reference [11] proposes a glove based in capacitive touch sensors, that translates American Sign Language. This system recognizes gestures from A to Z and numbers from 0 to 9 with an accuracy rate over 92%. The advantages of this advice are the low-cost hardware used and the portability of the device.

Although there are systems that translate Portuguese Sign Language [2], [4], [12], they are not portable or have high costs. This work intends to answer these limitations, using low-cost and wireless hardware.

The purpose is to create a system for the translation of the LGP based on methods used for the translation of other languages.

III. Proposed System Architecture

The system is aimed at a single user, containing a sensor based glove, a bluetooth module and an Android App, represented in Fig. 1. When the user represents a gesture, the machine learning algorithm compares a set of predefined values that are associated with a letter with the measured values received from the flex sensors and acelerometer, and if those values are close to the predefined ones, it tries do identify the letter. This information will be sent to the Android App via Bluetooth. In the first phase of the work, which is described in this paper, the collected data is obtained solely from the flex sensors. In a later phase, the accelerometer will be integrated and the data of this sensor will also be collected. In the work developed in this article the Android App is not yet fully implemented. It already receives data provided by LilyPad but the alogorithm has not yet been implemented directly in the app. At this stage the translation done is displayed in the graphical interface of the processing program, that is an open-source graphical library and integrated development environment (IDE). To develop the proposed system architecture, it is necessary to consider two fundamental parts, the glove and the algorithm. Both parts will be described in detail in subsection A and B, respectively.

A. Hardware Description

The sensor glove is composed of three hardware elements, as shown in Fig. 2: five flex sensors, a LilyPad USB and a bluetooth. When combined, these sensors allow hand gesture recognition.

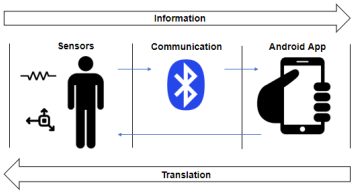


Fig. 1. System Architecture

Fig. 2. Glove seen from the front and back

Each one has its own function, such as:

- **LilyPad Arduino USB:** This is a wearable board, that is the central component of the system [13]. It is responsible for powering the system. Every hardware component will be connected to LilyPad USB by conductive threads and sewable snaps. The LilyPad USB only has four analog outputs, however observing the datasheet of the central microcontroller (ATmega32U4) we were able to understand that the LilyPad has two digital outputs (pins 9 and 10) that can also be programmed as analog outputs. Since each finger will have its own analog output it was necessary to program pin 9 through the ADC.
- **Bluetooth RN41:** This bluetooth is responsible for sending the arduino data to the android application. This module use simple secure pairing (SSP) if it is attempting to pair with devices that supports the bluetooth specification version 2.1 + EDR [14]. Since the bluetooth can not be sewn directly on the glove, a Lilypad XBee was used and the bluetooth attached. This board includes easy-to-sew tabs and all the necessary power regulation to run on the LilyPad system.
- **Flex sensor:** This sensor is responsible for tracking the finger orientation. In order to obtain a low-cost design, it was implemented using velostat, which is a opaque material made of a polymeric foil impregnated with carbon to make it electrically conductive. The velostat measures the amount of deflection or bending allowing to understand which position the finger is [15], and the conductive thread carries the current allowing to create a circuit. The glove is composed by five flex sensors, each one on one finger [16].
  Since the flex sensor acts as a resistor, it was necessary to

measure the value of this resistance when it is stretched and flexed. For this, values were taken with the help of the multimeter for both situations. The values presented an average of 7 kΩ when the finger was stretched and an average of 100 kΩ when the finger was flexed.
In order to obtain more accurate values of the five flex sensors it was necessary to use a conditioning circuit, like the one showin in Fig. 3.

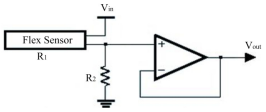


Fig. 3. Flex Sensor

The choice of resistance to be used (*R2*) is calculated by the following (1),

$$R2 = \frac{R1(Vcc - V0)}{V0} \tag{1}$$

where the parameters are defined as:

- *V*0: Output Voltage [V]
- *R*2: Resistance limiting [kΩ]
- *R*1: Variable resistance that is given by the flex sensors [kΩ]
- *Vcc:* Reference Voltage [V]

The *R*2 resistance will assume a value of 10 kΩ.

B. Software Description

- **Learning Algorithm:** To implement the letter detection task we explored the use of Neural Networks. First five gestures were performed for each letter. Since a person never makes a gesture in the same way, there were small variations of values in the five tests of each letter. For each sampling phase, a thousand samples were taken and their respective avereage. Then the ten samples closest to this average, based on the Euclidean distance (2), were considered. For each letter, 50 samples were obtained.

$$d(a,b) = \sum_{i=1}^{N} (a_i - b_i)^2 \tag{2}$$

Parameters in (2) defined as:

- a: Average of one sampling
- b:Every values of the corresponding sampling

Since there are 26 letters in the alphabet, there are 26 files with the respective values for each letter. In each of these files were placed all the letters, marking with 1 the own letter and the remainder as 0. This way the instruction for the neural network is only given 1 when it finds its letter and 0 when it finds another one. A portion of the data set of the letter 'A' is shown in Fig. 4.
For this algorithm Neuroph Studio was used to train the various neural networks [17]. In this case, since there is a neural network for each letter, we had to define 26

```
65,177,13,50,7,1
68,176,14,50,6,1
65,176,13,50,7,1
65,177,12,50,8,1
35,180,141,71,14,0
36,180,142,72,13,0
34,178,143,72,13,0
```

Fig. 4. Data preparation

different sets of data, where 70% was used for training and 30% to validate the results (Fig. 5).

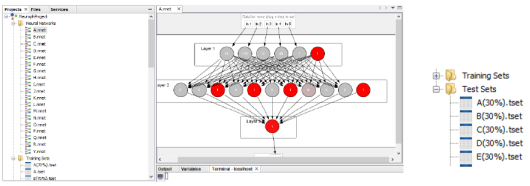


Fig. 5. Neuroph Studio

Initially there was no normalization of the input values, so the neural networks were not training well. The weights were not being initialized with a value low enough by Neuroph Studio. In order to do this it was found the highest and lowest value in the data of the all letters and applied the (3). In this case the highest was 197 and the lowest 0.

$$v' = \frac{value - min}{max - min} \tag{3}$$

Parameters in (3) defined as:

- value: Real time value of flex sensor
- max: Higest value
- min: Lowest value

This way it was possible to obtain the data in a format suitable for training (Fig. 6).

```
0.3299,0.8934,0.0660,0.2538,0.0355,1
0.3299,0.8985,0.0609,0.2538,0.0406,1
0.1777,0.9137,0.7157,0.3604,0.0711,0
0.1827,0.9137,0.7208,0.3655,0.0660,0
0.1726,0.9036,0.7259,0.3655,0.0660,0
0.1777,0.9086,0.7259,0.3706,0.0660,0
```

Fig. 6. Normalized Values

- **processing application:** The processing application has the purpose of displaying a letter associated with a gesture using the machine learning algorithm which takes input from the values of the flex sensors. The end result is represented in Fig. 7. The five fields on the left correspond to the values of the fingers, beginning in the little finger and finishing in the thumb, by their respective order. The field on the right corresponds to the letter given as translation.
- **Android application:** The app consists of four buttons, the first to turn on the buetooth, the second to turn

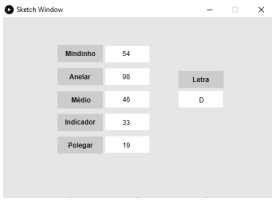


Fig. 7. processing application

it off, the third to show paired devices and the fourth to discover new devices. In the test we performed we paired the mobile phone with bluetooth RNBT-9B10 (glove bluetooth) and the bluetooth status was changed to connected. LilyPad then sent the app, via bluetooth, the phrase ”receiving data” that was printed in the text field of the app, from 200 ms in 200 ms (Fig. 8). The app is not yet fully implemented, the next step is to implement the algorithm in it.

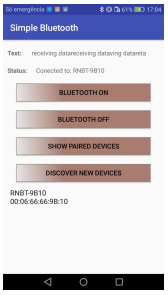


Fig. 8. Android application

IV. Results and Discussion

The first tests were done with the Arduino Uno instead of the LilyPad USB. This way it was guaranteed that the five flex sensors were working properly before the LilyPad was sewn to these sensors.

Since the hardware has not yet been fully implemented the results obtained were not the most desired, reaching a hit rate of 65.3%, when tested for the translation of the alphabet:

- Correct letters: C, D, E, F, G, I, K, M, N, O, P, S, T, U, W, Y, Z
- Wrong letters: A, B, H, J, M, Q, R, V, X

. This percentage corresponds to a single test attempt in which each letter was tested only once. In the final system each letter will be tested several times to see what is the correctness of each hit. However, there is still a long way to go, where the next step is the implementation of an accelerometer. With this implementation it is expected that the rate of accuracy will increase since this sensor will allow to distinguish gestures with equal configurations and with different orientations, as is the case of ”M” and ”W”.

In the final system the processing program will be replaced by a mobile application, where it will be possible to observe, in real time, the translation of the gesture. The transfer of data from the LilyPad USB to the android aplication will be supported through a Bluetooth connection, allowing the system to be totally wireless.

V. Conclusions and Future Work

This article introduced a low-cost and wireless sensor based glove capable of translating Portuguese Sign Language. The hardware and software were selected taking into account the characteristics and requirements of the system. Future work consists on implementing an accelerometer and complete the mobile application in order to display the letter associated with the gesture in real time.

References

[1] “Associação de Surdos do Porto,” 2018.
[2] P. Escudeiro, N. Escudeiro, R. Reis, J. Lopes, M. Norberto, A. B. Baltasar, M. Barbosa, and J. Bidarra, “Virtual Sign - A Real Time Bidirectional Translator of Portuguese Sign Language,” *Procedia Computer Science*, vol. 67, pp. 252–262, 2015.
[3] “Reconhecimento oficial da LGP,” 2018.
[4] I. G. S. Neiva, “Desenvolvimento de um tradutor de Língua Gestual Portuguesa,” p. 107, 2014.
[5] V. I. Pavlovic and T. S. Huang, “Visual Interpretation of Hand Gestures for Human-Computer Interaction : A Review,”
[6] S. Lang, M. Block-berlitz, S. Lang, M. Block, and R. Rojas, “Sign Language Recognition Using Kinect,” no. November, 2015.
[7] J.-H. Kim, N. D. Thang, and T.-S. Kim, “3-D hand motion tracking and gesture recognition using a data glove,” *2009 IEEE International Symposium on Industrial Electronics*, no. ISIE, pp. 1013–1018, 2009.
[8] M. Jin, C. Zaid, O. Mohamed, and H. Jaward, “A review of hand gesture and sign language recognition techniques,” *International Journal of Machine Learning and Cybernetics*, 2017.
[9] L. E. Potter and J. Araullo, “The Leap Motion controller : A view on sign language,” pp. 175–178, 2013.
[10] K. A. Bhaskaran, A. G. Nair, K. D. Ram, K. Ananthanarayanan, and H. R. N. Vardhan, “Smart Gloves for Hand Gesture Recognition Sign Language to Speech Conversion System,” *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, pp. 1–6, 2016.
[11] K. S. Abhishek, C. F. Q. Lee, and H. Derek, “Glove-Based Hand Gesture Recognition Sign Language Translator using Capacitive Touch Sensor,” pp. 334–337, 2016.
[12] A. D. Oliveira and M. Castanheira, “Reconhecimento de Língua Gestual,”
[13] Sparkfun, “LilyPad Arduino USB - ATmega32U4 Board,” 2018.
[14] R. Networks, “Bluetooth Data Module Command Reference & Advanced Information User’s Guide MODULES: RN24 RN25 RN41 RN42 RN41XV RN42XV SERIAL ADAPTERS: RN220XP RN240 RN270 RN274 RN-BT-DATA-UG,” pp. 1–83, 2013.
[15] E. Jeong, J. Lee, and D. Kim, “Finger-gesture Recognition Glove using Velostat (ICCAS 2011),” *2011 11th International Conference on Control, Automation and Systems*, no. Iccas, pp. 206–210, 2011.
[16] N. Takahashi, “Arduino flex sensor glove 3.957 6 2 —,” 2017.
[17] A. Goel, M. Sheezan, S. Masood, and A. Saleem, “Genre classification of songs using neural network,” *Proceedings - 5th IEEE International Conference on Computer and Communication Technology, ICCCT 2014*, pp. 285–289, 2015.

Bibliografia

[1] "Associação de Surdos do Porto," [Accessed: 2018-11-17]. [Online]. Available: http://www.asurdosporto.org.pt/artigo.asp?idartigo=77

[2] P. Escudeiro, N. Escudeiro, R. Reis, J. Lopes, M. Norberto, A. B. Baltasar, M. Barbosa, and J. Bidarra, "Virtual Sign - A Real Time Bidirectional Translator of Portuguese Sign Language," *Procedia Computer Science*, vol. 67, pp. 252–262, 2015.

[3] "Assoçiação Portuguesa de Surdos - Reconhecimento oficial da LGP." [Online]. Available: http://www.apsurdos.org.pt/index.php?option=com{_}content{&}view=article{&}id=41{&}Itemid=18

[4] I. G. S. Neiva, "Desenvolvimento de um tradutor de Língua Gestual Portuguesa," Ph.D. dissertation, Faculdade de Ciências e Tecnologia, 2014.

[5] "World Health Organization - Deafness and hearing loss." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[6] O. Oliveira, "Tradutor da Lingua Gestual Portuguesa - Modelo de Tradução Bidireccional," Ph.D. dissertation, Instituto Superior de Engenharia do Porto, 2014.

[7] "Infopédia - Dicionário de Língua Gestual Portuguesa," 2003. [Online]. Available: https://www.infopedia.pt/dicionarios/lingua-gestual

[8] R.-H. Liang and M. Ouhyoung, "A Real-time Continuous Gesture Recognition System for Sign Language," *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 558–567, 1998.

[9] I. S. Correia, "O parâmetro expressão na Língua Gestual Portuguesa: unidade suprassegmental," *Revista Científica - Educação e Formação*, no. 1, pp. 57–68, 2009.

[10] V. I. Pavlovic and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction : A Review," vol. 19, no. 7, pp. 677 – 695, 1997.

[11] J.-H. Kim, N. D. Thang, and T.-S. Kim, "3-D hand motion tracking and gesture recognition using a data glove," *2009 IEEE International Symposium on Industrial Electronics*, pp. 1013–1018, 2009.

[12] M. J. Cheok, O. Zaid, O. Mohamed, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, 2019.

[13] T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 466, pp. 1371–1375, 1996.

[14] S. Lang, M. Block-berlitz, S. Lang, M. Block, and R. Rojas, "Sign Language Recognition Using Kinect," *11th International Conference*, pp. 394–402, 2012.

[15] L. E. Potter and J. Araullo, "The Leap Motion controller: A view on sign language," *The 25th Australian Computer-Human Interaction Conference*, pp. 175–178, 2013.

[16] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," *2014 IEEE International Conference on Image Processing*, pp. 1565–1569, 2014.

[17] "PlayStation®Move Motion Controller." [Online]. Available: https://www.playstation.com/en-us/explore/ps3/

[18] J. Wu, L. Sun, and R. Jafari, "A Wearable System for Recognizing American Sign Language in Real - time Using IMU and Surface EMG Sensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 5, pp. 1281–1290, 2016.

[19] J. Kim, J. Kim, S. Mastnik, and E. André, "EMG-based hand gesture recognition for realtime biosignal interfacing EMG-based Hand Gesture Recognition for Realtime Biosignal Interfacing," *13th international conference on Intelligent user interfaces*, pp. 30–39, 2008.

[20] H. Abdelnasser, "WiGest : A Ubiquitous WiFi-based Gesture Recognition System," *2015 IEEE Conference on Computer Communications*, pp. 1472–1480, 2015.

[21] N. Kawale, P. Kaspate, H. Vanjari, and P. P. Sarode, "Implementation Paper on Sign Language Using Flex Sensor," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 6, no. 3, pp. 2623–2629, 2018.

[22] J.-s. Lee, Y.-w. Su, and C.-c. Shen, "A Comparative Study of Wireless Protocols : A SURVEY AND A COMPARISON," *The 33rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 46–51, 2007.

[23] E. Ferro and F. POTORTÌ, "Bluetooth and wi-fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications magazine*, vol. 12, no. 1, pp. 12–26, 2005.

[24] C. Gomez, J. Oller, and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," vol. 12, no. 9, pp. 11 734–11 753, 2012.

[25] F. Cercas, N. Souto, and A. Glória, "Comparison of Communication Protocols for Low Cost Internet of Things Devices," *2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference*, 2017.

[26] S. B. Kotsiantis, "Supervised Machine Learning : A Review of Classification Techniques," *2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 31, pp. 3–24, 2007.

[27] C.-f. Tsai, Y.-f. Hsu, C.-y. Lin, and W.-y. Lin, "Expert Systems with Applications Intrusion detection by machine learning : A review," *Expert Systems With Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[28] A. Syed, Z. T. H. Agasbal, T. Melligeri, and B. Gudur, "Flex Sensor Based Robotic Arm Controller Using Micro Controller," *Journal of Software Engineering and Applications*, vol. 5, no. 5, pp. 364–366, 2012.

[29] J. S. Neely and P. Restle, "CAPACTIVE BEND SENSOR," 1997.

[30] K. S. Abhishek, C. F. Q. Lee, and H. Derek, "Glove-Based Hand Gesture Recognition Sign Language Translator using Capacitive Touch Sensor," *2016 IEEE International Conference on Electron Devices and Solid-State Circuits*, pp. 334–337, 2016.

[31] E. Jeong, J. Lee, and D. Kim, "Finger-gesture Recognition Glove using Velostat," *The 11th International Conference on Control, Automation and Systems*, pp. 206–210, 2011.

[32] PtRobotics, "Velostat." [Online]. Available: https://www.ptrobotics.com/sensores-forca-vibracao-e-pressao/4754-pressure-sensitive-conductive-sheet-velostatlinqstat.html?search{_}query=velostat{&}results=1

[33] "Virtual Sign." [Online]. Available: http://193.136.60.223/virtualsign/pt/index.php

[34] "MotherBoard - Augmented Reality App Can Translate Sign Language Into Spoken English, and Vice Versa." [Online]. Available: https://motherboard.vice.com/en{_}us/article/zmgnd9/app-to-translate-sign-language

[35] "Sign Language Translator Glove, Made with Arduino, Inspired by Jeremy Blum." [Online]. Available: http://www.romanakozak.com/sign-language-translator/

[36] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett, "The LilyPad Arduino : Using Computational Textiles to Investigate Engagement , Aesthetics , and Diversity in Computer Science Education," *Conference on Human Factors in Computing Systems*, pp. 423–432, 2008.

[37] Sparkfun, "LilyPad Arduino Simple Board." [Online]. Available: https://www.sparkfun.com/products/10274

[38] ——, "LilyPad Arduino 328 Main Board," 2018. [Online]. Available: https://www.sparkfun.com/products/13342

[39] ——, "LilyPad Arduino USB - ATmega32U4 Board." [Online]. Available: https://www.sparkfun.com/products/12049

[40] ——, "LilyPad Arduino SimpleSnap," 2018. [Online]. Available: https://www.sparkfun.com/products/10941

[41] N. Takahashi, "Arduino flex sensor glove." [Online]. Available: https://www.instructables.com/id/Arduino-Flex-Sensor-Glove/

[42] Pololu, "LSM303D 3D Compass and Accelerometer Carrier with Voltage Regulator." [Online]. Available: https://www.pololu.com/product/2127

[43] F. Leens, "An introduction to I2C and SPI protocols," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.

[44] GooglePlay, "Apps - Serial Bluetooth Terminal." [Online]. Available: https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=en

[45] "Bluetooth BTBee PRO - Comunicações." [Online]. Available: https://www.botnroll.com/pt/bluetooth/559-bluetooth-btbee-pro.html

[46] Sparkfun, "RN41-XV Bluetooth Module - Chip Antenna." [Online]. Available: https://www.sparkfun.com/products/retired/11600

[47] *Bluetooth Data Module Command Reference & Advanced Information User's Guide.* Roving Networks, 2013.

[48] Sparkfun, "LilyPad Xbee." [Online]. Available: https://www.sparkfun.com/products/12921

[49] Arduino, "Software Serial Example." [Online]. Available: https://www.arduino.cc/en/tutorial/SoftwareSerialExample

[50] Adafruit, "Stainless Thin Conductive Thread - 2 ply - 23 meter/76 ft." [Online]. Available: https://www.adafruit.com/product/640

[51] ——, "Sewable Snaps - 5mm Diameter." [Online]. Available: https://www.adafruit.com/product/1126

[52] Botnroll, "Lithium-ion Polymer Battery 3.7V 550mAh." [Online]. Available: https://www.botnroll.com/pt/baterias/1409-lithium-ion-polymer-battery-37v-550mah.html

[53] "StatCounter GlobalStates - Mobile Operating System Market Share Worldwide." [Online]. Available: http://gs.statcounter.com/os-market-share/mobile/portugal/{#}monthly-201712-201812

[54] "Paíneis | Android Developers." [Online]. Available: https://developer.android.com/about/dashboards/?hl=pt-br

[55] M. B. Schubert and J. H. Werner, "Flexible solar cells for clothing," vol. 9, no. 6, pp. 42–50, 2006.