UNIVERSIDADE D
COIMBRA

Cristina Pierdevară

ANDARILHO INSTRUMENTADO PARA REABILITAÇÃO E ASSISTÊNCIA

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Eletrotécnica e de Computadores, ramo de Robótica, Controlo e Inteligência Artificial, orientada pelo Professor António Paulo Mendes Breda Dias Coimbra, Professor João Paulo Morais Ferreira e apresentada à Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2022

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Andarilho instrumentado para reabilitação e assistência

Cristina Pierdevară

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador:	Professor António Paulo Mendes Breda Dias Coimbra
Co-orientador:	Professor João Paulo Morais Ferreira
Co-orientador:	Professor Manuel Marques Crisóstomo

Júri

Presidente:	Professor Urbano José Carreira Nunes
Orientador:	Professor António Paulo Mendes Breda Dias Coimbra
Vogal:	Professor Rui Paulo Pinto da Rocha

Coimbra, Setembro de 2022

”Para a maioria de nós, a tecnologia torna as coisas mais fáceis. Para uma pessoa com deficiência, torna as coisas possíveis.”

Judy Heumann

Agradecimentos

Com esta entrega encerro um dos capítulos mais importantes da minha vida. Contudo, existiram diversos subcapítulos, uns bons, outros maus, mas todos eles fazem parte de mim, por isso estarão comigo para o resto da minha vida.

Correndo o risco de parecer ingrata e poder deixar de lado alguma pessoa que fez parte desta fase deixo um sincero obrigada a todas as pessoas que de uma forma ou de outra estiveram comigo ao longo destes longos anos.

Estou profundamente grata a todas as pessoas que me apoiaram em todos os sentidos para tornar esta realização possível, no entanto existe um conjunto de pessoas dignas de serem mencionadas e que estiveram mais próximas de mim, tendo algumas até percorrido um caminho semelhante ao meu.

Em primeiro lugar, aos meus pais por todos os esforços e apoio ao longo do tempo deste meu percurso académico. A distância e as saudades apertaram, mas sempre se mostraram disponíveis para me dar força. Foi uma longa caminhada, com muitos obstáculos que nunca conseguiria superar sem eles.

Aos meus amigos, por toda a ajuda e apoio nos bons e maus momentos.

Um especial obrigada aos meus conselheiros e orientadores, Professor António Paulo Mendes Breda Dias Coimbra, Professor João Paulo Morais Ferreira e Professor Manuel Marques Crisóstomo, pela oportunidade de integração num projeto como este e por toda a orientação durante as fases de pesquisa e desenvolvimento. Foi uma ótima oportunidade de aprendizagem e muito importante para os meus desafios futuros. Desejo-lhes sucesso profissional e pessoal.

De seguida, ao Instituto de Sistemas e Robótica e à Fundação para a Ciência e Tecnologia pelo apoio financeiro ao projeto UIDB/00048/2020.

Por último, um especial obrigado a todos os que já não fazem parte da minha vida, obrigado por me terem permitido crescer e evoluir de uma maneira totalmente desconhecida para mim.

A todos, muito obrigado!

INSTITUTE OF SYSTEMS AND ROBOTICS
UNIVERSITY OF COIMBRA

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Resumo

Atualmente, com o aumento da longevidade, o número da população com compromisso na deambulação tem vindo a aumentar. Por outro lado, a qualidade de vida através da manutenção da autonomia e auto-cuidado tem vindo a ser procurada. Em muitas situações, esta é possível com o auxílio de equipamentos de apoio técnico, sendo que na deambulação interior e exterior o uso de andarilhos é mais frequente.

A criação de equipamentos inteligentes torna-se um desafio crescente para a área da engenharia aplicada à saúde. O desenvolvimento de novos andarilhos inteligentes aumentará a segurança e a independência na mobilidade das pessoas com marcha comprometida.

Para o aumento da segurança da pessoa com compromisso na deambulação, levou-se a cabo um projeto de concepção e inovação de um andarilho de três rodas, modela Delta, cujo objetivo foi atribuir-lhe novas funcionalidades inteligentes.

Como base utilizou-se um microcontrolador Arduino Mega 2560 e vários sensores de entradas analógicas e digitais. Estes permitiram a criação de vários sistemas inteligentes, tais como travagem automática, deteção de obstáculos, iluminação de presença e alertas luminosos, alarme sonoro e buzina, deteção de queda e visualização da velocidade instantânea e distância percorrida. A conexão dos sensores com o exterior é feita através de uma placa de circuito impresso para o Arduino Mega 2560, possibilitando a comunicação entre ambas.

Ainda foi desenvolvida uma aplicação móvel para o sistema operativo Android com a finalidade de melhorar a monitorização e comunicação entre o cuidador e a pessoa com dependência. Também se integrou na mesma um sistema de localização por GPS. Através de componentes *Progressive Web App* (PWA), a aplicação pode ser acedida por Web ou por sistemas operativos iOS.

Com este protótipo, conseguiu-se dar mais um passo na área de engenharia da saúde, relacionado com inovação de mais um dispositivo de ajuda de apoio técnico, com uso para ambientes interiores e exteriores.

Palavras-Chave

Andarilho inteligente; envelhecimento ativo; segurança ativa; IoT.

Abstract

Nowadays, with the increase of longevity, the number of people with roaming issues has been rising. On the other hand, the search for quality of life improvements through the maintenance of autonomy and self-care has increased. In many situations, this can be achieved with the help of technical support equipments, where walkers are more frequently used in indoor and outdoor roaming.

The creation of smart equipment is an ever growing challenge in health related engineering. The development of new smart walkers will increase the safety and independence of the mobility of people with impired march.

To increase the safety of the person with impaired roaming, a conceptual and innovative project was devised of a three wheel walker, Delta model, which purpose is to add new smart features.

An Arduino Mega 2560 microcontroller was the base of the embedded systems, comprising several analog and digital input sensors. These allowed the creation of several smart systems, such as automatic braking, obstacle detection, presence lighting and luminous alarms, sound alarms and horn, fall detection, visualization of current speed and travel distance.

The connection between the sensors and the exterior is made through a printed circuit board for the Arduino Mega 2560, enabling communication between both.

Furthermore, a mobile application was developed for the Android operating system for the purpose of improving the monitorization and comunication between the caregiver and the person with needs. It was also integrated with a GPS-based tracking system. Through Progressive Web App (PWA) components, the application can be accessed through Web or iOS operating systems.

With this prototype, another step in health related engineering was taken, in regard to the inovation of another technical help device, for indoor and outdoor use.

Keywords

Smart walker; active living; active safety; IoT.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AC	corrente alternada
AWS	*Amazon Web Services*
DMP	processador de movimento digital
I2C	*Inter-Integrated Circuit*
ISR	Instituto de Sistemas e Robótica
LDO	*Low-Dropout Regulator*
LDR	*Light Dependent Resistor*
LED	*Light-Emitting Diode*
MCU	microcontrolador
PCI	placa de circuito impresso
PSD	detetor sensível à posição
PSD	sensor de posição sensível
PTH	*Pin Through Hole*
PWA	*Progressive Web App*
PWM	*Pulse Width Modulation*
SMD	*Surface Mounted Device*
UC	Universidade de Coimbra
USB	*Universal Serial Bus*

1

Introdução

Hoje em dia, com o aumento da prevalência de alterações crónicas relacionadas com o sistema osteoarticular, sensorial e distúrbios neurológicos, está a haver um aumento das incapacidades funcionais e das alterações das atividades da vida diária das pessoas [1]. Pois, com o aumento da longevidade estas condições de saúde têm uma amplitude maior a nível mundial, fragilizando progressivamente a independência da pessoa [2]. Contudo, estas alterações de saúde podem comprometer a pessoa em qualquer faixa etária de uma forma progressiva ou repentina, com repercussões muitas das vezes na marcha e no equilíbrio [3], [4].

Lippert [5], em 2010, define a marcha ou deambulação como uma forma de caminhar, sendo que caminhar é o momento quando a pessoa se desloca de um lado para outro pelos próprios pés com ou sem ajuda de terceiros ou equipamentos. Ainda o mesmo autor menciona que, para um objeto estar equilibrado, todas as forças exercidas sobre o mesmo devem estar simétricas e quando não acontece a segurança da pessoa pode ser comprometida, levando a múltiplas complicações que podem interferir no grau de dependência da pessoa.

Para além do desequilíbrio, a marcha também pode ser alterada por outros fatores, sendo eles a perda da massa e força muscular, encurtamento muscular ou dos segmentos ósseos, rigidez, dor osteoarticular e até alterações derivadas da inatividade da pessoa.

Perante essas alterações do estado de saúde, cada pessoa afetada reage de uma forma diferente, dependendo das vivências e circunstâncias de cada um. Uma pessoa pode recuperar na totalidade, outra ficar totalmente dependente nas atividades do dia-a-dia e outra recuperar parcialmente, embora com dificuldades em mudanças posturais, deambular, levantar ou até carregar objetos para satisfazer as suas necessidades diárias [2], [6].

Frequentemente, estas limitações podem ser ultrapassadas com reabilitação funcional [2], com o auxílio de outras pessoas ou de dispositivos auxiliares de marcha. Assim permite-se à pessoa com dependência realizar tarefas mais complexas, participar no seu meio social e torná-la cada vez mais independente na vida quotidiana [7].

1.1 Motivação

Em 2015, segundo a Ordem dos Enfermeiros [8], os tipos de auxiliares de marcha mais comuns eram: muleta, bengala e andarilho, apresentando este último uma maior segurança devido à sua estabilidade. Existem diversos andarilhos no mercado, mas a sua escolha deve ser feita por profissionais de saúde habilitados, que terão em consideração não só o grau de alteração da mobilidade, como também a capacidade de recuperação da pessoa.

Para existir escolha do andarilho adequado, a oferta dos mesmos deverá ser variada de modo a que os profissionais de saúde os possam adequar a cada pessoa, conforme as próprias características. Esta exigência na área da saúde atribui um desafio constante à engenharia na pesquisa e desenvolvimento de novos modelos de andarilhos cada vez mais práticos e seguros [9], [10]. Nesta linha

de pensamento, a medicina e a engenharia são dois alicerces imprescindíveis para o aumento da qualidade de vida das pessoas.
Por conseguinte, este trabalho surge na sequência de uma outra dissertação de mestrado do autor Cavaleiro (2017) [11], realizada no Instituto de Sistemas e Robótica (ISR) da Universidade de Coimbra (UC). O seu objetivo foi controlar um andarilho de três rodas (andarilho Delta) através da introdução de diversas tecnologias, tais como; sistema de travagem automática, sistema de deteção de obstáculos, sistema de deteção de movimento das rodas, sistema de medição de força das mãos e sistema de alarme sonoro.
Perante a concorrência económica, que torna a inovação de produtos (pesquisa, desenvolvimento e lançamento de novos produtos) um meio essencial de conservação e conquista de faixas de mercado, a presente dissertação pretende modernizar ainda mais o trabalho já executado anteriormente, tornando-o num prototipo mais próximo possível de um produto fiável e seguro a inserir no mercado. É também uma componente deste trabalho melhorar o *design* estético do mesmo, tornando-o mais apelativo aos consumidores. Assim, este poderá servir como uma continuação para o desenvolvimento de trabalhos futuros, uma vez que os mercados se encontram cada vez mais saturados de produtos pouco diferenciados e a procura acaba por incidir sobre aqueles que apresentam aspetos inovadores.

1.2 Objetivos

O objetivo geral desta dissertação é modificar um andarilho de três rodas (andarilho Delta), atribuindo-lhe novas funcionalidades inteligentes para aumentar a segurança da pessoa com dependência.
Para se concretizar este objetivo, delineou-se os seguintes objetivos específicos:

- Desenvolvimento de um sistema de travagem automático;
- Criação de um sistema de deteção de movimento e sentido da direção de cada uma das rodas;
- Conceção de uma sistema de deteção de obstáculos;
- Desenvolvimento de um sistema de deteção de queda da pessoa;
- Adaptação de um sistema luminoso para assinalar a presença da pessoa e a deteção de obstáculos ou quedas;
- Conceção de um sistema sonoro para alarme de eventos acidentais, e para aviso de presença;
- Desenvolvimento de uma aplicação *Android* para meio de comunicação entre o cuidador e a pessoa com dependência e integração de um sistema de localização;

- Otimização funcional e na robustez do circuito através do desenho de uma placa de circuito impresso (PCI) e de uma caixa de proteção, conferindo também uma melhor estética a todo o sistema eletrónico.

1.3 Estrutura do documento

De forma a facilitar a compreensão e exposição dos diferentes elementos de investigação, a presente dissertação organiza-se em oito capítulos principais.
No capítulo 1 apresenta-se uma contextualização geral da investigação abordando a problemática do projeto, a justificação do desenvolvimento do trabalho assim como os objetivos geral e específicos.
Quanto ao capítulo 2, este aborda a revisão de literatura acerca dos tipos e funcionalidades dos andarilhos existentes no mercado.
No capítulo 3 estão detalhados todos os componentes utilizados na conceção deste projeto. A implementação e desenvolvimento da PCI e da caixa mecânica de proteção do circuito estão detalhadas no capítulo 4.
O *software* de todos os sistemas concebidos neste projeto está detalhado no capítulo 5.
No capítulo 6 está explícito o desenvolvimento da aplicação Android.
Os testes efetuados e os respetivos resultados para a avaliação do protótipo estão descritos no capítulo 7. Por último, o capítulo 8, apresenta as principais conclusões dos contributos da investigação e desenvolvimento executado, menciona as limitações encontradas e sugere novas linhas de pesquisa.
Por fim, é ainda incluído um apêndice, onde se descreve a pesquisa bibliográfica efetuada para uma melhor perceção de todo o processo de desenvolvimento do projeto.

2

Revisão de literatura

Ao longo dos últimos anos, com o avanço científico e tecnológico, assistimos a um envelhecimento populacional a nível mundial. Assim, paralelamente ao crescimento do índice de envelhecimento, o número de pessoas com dependência devido a diversas situações de dependência física, psíquica, sensorial ou motora também tem aumentado [3]. A deambulação demonstrou diminuir a prevalência das condições crónicas de saúde das pessoas, ajudando-os a manter a melhoria da qualidade de vida e conservação do estado de saúde [12]. Muitas vezes, as pessoas com alguma alteração do estado de saúde sentem-se inseguras a deambular, permanecendo por longos períodos confinadas, interferindo não apenas na parte motora como também na parte social e psicológica da pessoa. A utilização de auxiliares da marcha pode contornar esta situação [7], dando mais independência no auto-cuidado diário destas pessoas com necessidades específicas. Os andarilhos, comparativamente com outros auxiliadores da marcha, são os mais utilizados por pessoas com alterações no equilíbrio da marcha e promovem maior mobilidade e autonomia no desempenho e realização das atividades diárias [13]. Estes equipamentos permitem à pessoa maior estabilidade, são simples e de fácil utilização, atribuindo um sentimento de segurança [14].
Os andarilhos existem em diversos formatos e para finalidades específicas, podendo ser utilizados tanto no interior como no exterior [15].
Estes podem ser fixos, articulados, com duas, três ou quatro rodas. Os mesmos podem ser simples ou inteligentes. Todos os tipos de andarilhos são construídos em alumínio de elevada resistência e regulável em altura, permitindo um conforto máximo a cada pessoa, indiferentemente da sua estatura [16].

2.1 Andarilho fixo e articulado

Os andarilhos sem rodas ou tradicionais são simples e possuem borrachas nas extremidades, permitindo maior aderência com o pavimento. São projetados principalmente para pessoas com problemas de mobilidade reduzida. Para a utilização deste equipamento, a pessoa deverá ter força nos membros inferiores e superiores, uma vez que é necessário erguer o aparelho e movimentá-lo para a frente durante a locomoção [17].
Segundo Tereso [18], em 2014, este equipamento pode desencadear o fenómeno de queda para trás enquanto a pessoa segura nele. O uso do mesmo também poderá provocar um aumento do gasto energético em comparação com outros andarilhos.
O andarilho articulado é concebido para pessoas com dificuldades de equilíbrio e menos força nos braços. Permite que a pessoa movimente um dos lados do andarilho de cada vez, acompanhando os pés e garantido sempre um ponto de apoio no chão. Tem a mesma funcionalidade do andarilho fixo mas carece de mais atenção e coordenação por parte da pessoa [17], assim como uma capacidade cognitiva adequada [9].

(a) Andarilho fixo [19]

(b) Andarilho articulado [20]

Figura 2.1: Andarilhos simples

O critério de escolha entre os dois dispositivos, poderá ser o peso por vezes maior do andarilho articulado. Devido à forma como funcionam, estes andarilhos não são ideais para longas distâncias e são mais adequados para uso interior. Assim, este equipamento torna-se ideal para aquelas pessoas que ficam principalmente em casa [17].

2.2 Andarilho com rodas

No que diz respeito aos andarilhos com rodas, estes podem ter duas, três ou quatro rodas.
O andarilho de duas rodas tem as pontas proximais revestidas por borrachas que permitem o controlo do deslizamento. Este modelo é indicado para pessoas que não têm força física ou apresentam menos resistência nos membros superiores, podendo beneficiar do seu uso, uma vez que as rodas frontais podem ajudar a acelerar o seu ritmo natural. Durante a marcha e para um uso adequado do aparelho, a pessoa deve aplicar o peso do seu corpo em cima do mesmo, não necessitando erguer o aparelho para se movimentar, pois as rodas facilitam este movimento. Desta forma, este acaba por ser uma vantagem em comparação ao andarilho fixo ou articulado [17].

(a) Andarilho articulado de duas rodas [21]

(b) Andarilho de duas rodas com assento [22]

Figura 2.2: Andarilhos com duas rodas

Por outro lado, os andarilhos de três e quatro rodas apresentam uma melhor flexibilidade e estabilidade sem atrasar a marcha, comparativamente a um andarilho tradicional ou com duas rodas. São indicados para pessoas que necessitam de parar a marcha com frequência.
Os andarilhos de três rodas são mais estreitos e têm um raio de giro mais apertado em comparação aos andarilhos de quatro rodas. São aconselhados para as pessoas que se encontram em espaços mais confinados e são mais leves do que um andarilho de quatro rodas.

(a) Andarilho de três rodas [23]

(b) Andarilho de quatro rodas [24]

Figura 2.3: Andarilhos com três e quatro rodas

Contudo, o andarilho de quatro rodas está em vantagem não só por apresentar mais suporte e maior estabilidade, como também por ter um assento que permite à pessoa sentar caso seja necessário.
Segundo Wellmon *et al.* [25], as pessoas que usam estes andarilhos têm limitações na realização de múltiplas tarefas em simultâneo assim como em recolher informação do ambiente, uma vez que precisam de dedicar mais atenção à condução deste equipamento.
A estes equipamentos podem ser adicionados vários acessórios, tais como sacos de compras e cestas.

2.3 Andarilho inteligente

Nos últimos anos, um grande número de auxiliares inteligentes de marcha têm sido desenvolvidos e testados nos centros universitários [9], [14], [26]. Os mesmos baseiam-se na mesma estrutura dos convencionais, mas integram múltiplos sensores que proporcionam melhor assistência na deambulação, navegação e sustentação do peso corporal da pessoa [27].
Os andarilhos inteligentes são desenhados de modo a trabalharem em perfeita sinergia com pessoas com deficiências não apenas na locomoção, mas também a nível cognitivo e sensorial.
Martins *et al.* [10] defendem que os andarilhos inteligentes precisam de satisfazer três características obrigatórias, tais como a segurança, ergonomia e facilidade no uso do mesmo. Ainda, os mesmos reforçam que apesar de existir evolução na conceção destes auxiliares inteligentes de marcha, a inovação procura de forma contínua integrar as seguintes funcionalidades:

- Suporte físico;
- Assistência sensorial;
- Assistência cognitiva;
- Monitorização de saúde;
- Interface homem-máquina avançada.

Em termos de funcionamento, Frizera *et al.* [28] dividem os andarilhos inteligentes em andarilhos passivos e ativos. Os passivos deixam a responsabilidade de locomoção à pessoa que utiliza o equipamento enquanto nos ativos a locomoção é fornecida através da combinação do impulso motor e da pessoa.

2.3.1 Suporte físico

Para a melhoria do suporte físico, um grupo de investigadores do departamento de Bioengenharia e Robótica da Universidade de Tohoku [29], [30] e [31], assim como outro grupo multicêntrico da Universidade do Minho (Portugal), da Universidade Federal do Espírito Santo (Brasil) e do Conselho Nacional de Pesquisa de Espanha [10] alargaram a base de sustentação de vários tipos de andarilhos inteligentes. Pois, para estes autores é consensual que a principal vantagem dos andarilhos passivos é a leveza, com a consequência de que a pessoa se pode mover facilmente aplicando pequenas forças. Através do alargamento da base de sustentação, a segurança é aumentada, prevenindo quedas, uma vez que o dispositivo é muito sensível causando movimentos mesmo quando as forças são aplicadas de forma não intencional.
Em 2017, Alves substituiu o guiador do andarilho por plataformas de apoio para o antebraço, reduzindo a carga nos membros inferiores e articulações [26].
Como os andarilhos inteligentes têm um peso maior relativamente aos convencionais, devido às baterias e acessórios eletrónicos, podem aumentar a probabilidade da queda da pessoa. Para aumentar a segurança diminuindo o risco de queda, em 2007, Hirata *et al.* [30], figura 2.4, propõem um protótipo de um andarilho com duas rodas giratórias à frente e um par de travões eletromecânicos nas rodas traseiras. Incorporaram também um inclinómetro para detetar o declive do pavimento, fornecendo o recurso de compensação de gravidade. Esta informação permite que a pessoa diminua a velocidade dependendo se o dispositivo está descendo.

Figura 2.4: Protótipo de andarilho inteligente desenvolvido por Hirata [30]

Com o objetivo de compensar a força e a coordenação da pessoa ao empurrar e guiar o andarilho, Miró [32] desenvolveu um andarilho inteligente ativo, figura 2.5, integrando motores nas rodas traseiras que auxilia na deambulação, sem fazer força excessiva, compensando a gravidade em superfícies inclinadas.

(a) Vista frontal (b) Vista traseira

Figura 2.5: Andarilho inteligente ativo proposto por Miró
Fonte: [32]

Contudo, o autor refere que este protótipo com a presença de atuação do utilizador sobre o andarilho interrompe o sistema passivo, levando a problemas de segurança, podendo provocar a queda da pessoa.

2.3.2 Assistência sensorial e cognitiva

Relativamente à assistência sensorial foram desenvolvidos vários andarilhos inteligentes que têm a funcionalidade de auxiliar na navegação, isto é, atribuir-lhe a habilidade de controlo do movimento do robô. Este controlo do movimento é atribuído através da integração de sensores ultrassónicos, de visão ou infravermelhos, que permitem identificar obstáculos [26], [33].
Sempre que um obstáculo é identificado, o sistema de controlo age de uma forma ativa ou passiva. No sistema passivo o sistema auxilia a pessoa através de alertas sonoros ou vibratórios, enquanto no sistema ativo o sistema intervém diretamente nos atuadores do dispositivo, mudando imediatamente a direção da pessoa.
Com estas tecnologias, em 2016, Shin [33] desenvolveu um andarilho inteligente autónomo e assistido (andarilho SMARTWALKER), equipado com sensores e atuadores, figura 2.6. No modo autónomo, o andarilho navega através de uma interface baseada em gestos que aceita os comandos.

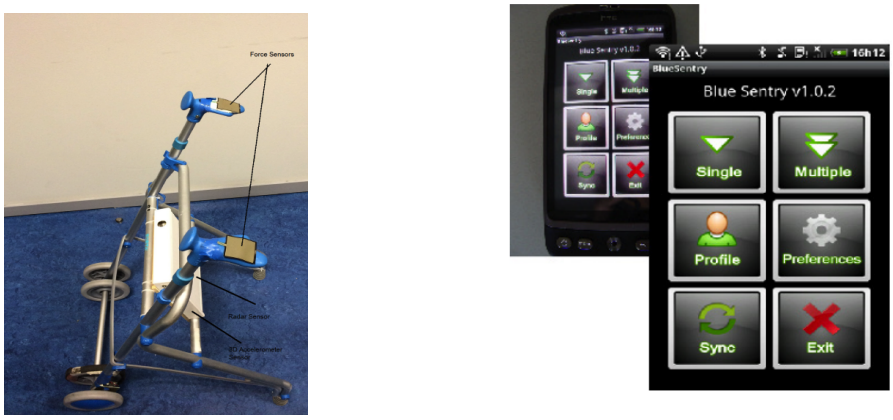
(a) SMARTWALKER (b) Interface gráfica do andarilho

Figura 2.6: Protótipo de andarilho inteligente autónomo e assistido desenvolvido por Shin
Fonte: [33]

No modo assistido, o controlador automático de velocidade determina a velocidade indicada para a pessoa. A localização da pessoa é feita através de um *scanner* e de outras informações sensoriais que localizam as pernas da mesma.
Este recurso é normalmente instalado em andarilhos inteligentes para fornecer assistência sensorial às pessoas com problemas visuais ou com respostas cognitivas limitadas.
De forma a evitar a possibilidade do andarilho rolar para a frente durante a deambulação, Martins [9] e Alves [26] desenvolveram outros auxílios sensoriais, ASBGo e ASBGo++, incorporando sensores de distância a laser e sensores de contacto.
Para adicionar assistência cognitiva, Kulyukin *et al.* [34] implementaram um andarilho inteligente, denominado iWalker. Este possui um serviço de nagevação que fornece à pessoa um mapa do ambiente e a localização da mesma. Assim, as pessoas com capacidades cognitivas alteradas conseguem orientar-se de forma mais facilitada em termos espaciais.

2.3.3 Monitorização de saúde

Alguns andarilhos inteligentes, dependendo do modo de operação, podem ser usados para monitorizar parâmetros de saúde da pessoa. Essas informações são coletadas e utilizadas para manter um histórico médico sobre a saúde do utente. Além disso, se o dispositivo estiver conectado à rede, também é capaz de informar através de uma comunicação sem fio um centro de saúde, a equipa médica ou o cuidador caso seja detetada uma situação de emergência. Um exemplo desse dispositivo foi desenhado por Postolache *et al.* [35], onde os profissionais de saúde habilitados podem auxiliar a pessoa durante o programa de reabilitação da marcha avaliando a força aplicada pela pessoa ao longo do tempo. Assim, os mesmos conseguem monitorizar os ganhos em saúde.



(a) Andarilho SWALK (b) Painel de aplicativos do andarilho

Figura 2.7: Protótipo do andarilho inteligente SWALK implementado por Postolache
Fonte: [35]

Em 2020, um grupo multidisciplinar liderado por Cristina Santos da Escola de Engenharia da Universidade do Minho (EEUM) [36], no âmbito do projeto ”EML – Equipamento Multifuncional

de auxílio técnico à Locomoção", estão a desenvolver vários modelos de andarilhos inteligentes. Estes equipamentos têm como objetivo permitir por um lado a estabilidade da marcha patológica e, por outro lado, disponibilizar aos profissionais de saúde uma ferramenta para monitorizar o estado da pessoa.
Estes dispositivos multifuncionais integram vários sensores que recolhem dados posturais e gestuais, contribuindo para uma correção da marcha atáxica[1].

2.3.4 Interface homem-máquina avançada

A interface homem-máquina é o meio de diálogo entre o equipamento e a pessoa e deve ser o mais eficiente e sensível possível. Os sensores incorporados desempenham um dos papéis mais relevantes nas interfaces, pois fazem a varredura do ambiente circundante e captam a interação humana com o dispositivo.
De acordo com Martins *et al.* [10], os andarilhos inteligentes podem apresentar uma interface direta ou indireta. Na interface direta os comandos impostos pela pessoa são enviados diretamente ao robô, enquanto na interface indireta as intenções da pessoa são estimadas através dos sensores incorporadas no dispositivo.
Um exemplo de protótipo de andarilho inteligente com uma interface direta desenvolveu Hashimoto [37]. Neste trabalho o *joystick*, além de transmitir as intenções da pessoa aos motores, fornece um *feedback* de força permitindo à pessoa perceber os obstáculos detetados (figura 2.8).

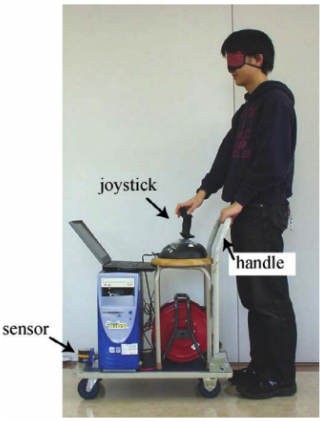


Figura 2.8: Protótipo de andarilho inteligente desenvolvido por Hashimoto [37]

De acordo com esta força repulsiva a pessoa pode saber a direção e a distância do obstáculo, permitindo-lhe evitar os mesmos e deambular em segurança.
Por outro lado, Taghvaei [38] propõe uma interface indireta, onde a câmara de profundidade é usada para captar o movimento dos membros da pessoa.

[1] **Marcha atáxica** - termo médico para descrever as alterações de caminhar caracterizadas por incoordenação dos membros inferiores.

2.4 Sumário

De acordo com a revisão de literatura analisada e com os objetivos propostos, o presente trabalho será um ponto de partida na continuação da inovação do desenvolvimento de funcionalidades adicionais de um andarilho inteligente de três rodas para um uso interior e exterior, sendo de extrema relevância não só para os profissionais de saúde em contexto de internamento como também para uso em contexto domiciliário e monitorização da pessoa com limitação de locomoção por parte dos cuidadores informais.

Tabela 2.1: Comparação entre os dispositivos descritos na revisão de literatura e o protótipo desenvolvido

Andarilho inteligente	Sensores	Funcionalidades
ASBGo walker [9]	Sensores resistivos de força, sensor infravermelho, telémetro a laser, sensor de profundidade ativo, sonares	Reconhecimento da intenção de movimento, adaptação ao utilizador, contorno de obstáculos, avaliação do padrão de marcha
RT Walker [29]	Sensor de força e *encoders*	Vários algoritmos para o controlo do movimento (evitar obstáculos, seguir caminho, entre outros)
Dispositivo Miró [32]	RFID, *encoders* e sensores externos	Transmissão da intenção de movimento para prevenir a queda, contorno de obstáculos
SMARTWALKER [33]	Diferentes atuadores, *encoders*, sensores infravermelhos, extensómetros, interruptores de contato, microcontrolador de baixo nível, computador e telémetro a laser	Aceita comandos de gestos, deteta o rosto e as pernas da pessoa, controlo de velocidade
iWalker [34]	RFID, *encoders* e sensores externos	Algoritmos de controlo
SWALK [35]	Sensores de força, acelerómetro, microcontroladores, *bluetooth*, telemóvel	Aplicação Android que monitoriza a qualidade da marcha
Dispositivo Hashimoto [37]	*Joystick*, sensor infravermelho	Sistema de *feedback* para reconhecer obstáculos ao redor
Dispositivo implementado	Wi-Fi, sensores infravermelhos, acelerómetro e giroscópio, seletores e botões, interruptores de contacto, sensores de Hall, servo motores, piezzo buzzer, sensores de força e fitas de LED	Diferentes algoritmos de controlo

Perante a análise e comparação entre os diferentes andarilhos inteligentes descritos na literatura, o protótipo desenvolvido é dirigido às pessoas com capacidades cognitivas ou motoras lentificadas, tornando a deambulação mais segura através da integração de novas funcionalidades não descritas nos trabalhos anteriores: sistema de travagem automático, sistema de deteção de movimento e sentido da direção, sistema de deteção de queda, sistema luminoso, sistema sonoro,

aplicação Android para meio de comunicação entre o cuidador e a pessoa com dependência e integração de um sistema de localização.
Para desenvolver o protótipo, utilizou-se o sistema de travagem mecânico concebido no anterior projeto de dissertação [11].
Para uma travagem suave e mais segura, implementou-se uma travagem proporcional à velocidade máxima permitida ao andarilho e inversamente proporcional à distância de um obstáculo.
Também foi calculada a velocidade instantânea e a distância percorrida pela pessoa.
Para o desenvolvimento dos sistemas supramencionados aplicou-se diferentes testes aos servo motores e aos sensores infravermelhos, descritos no capítulo 7. Na realização dos testes dos servo motores, o andarilho foi guiado por uma pessoa e verificou-se se os ângulos de travagem e destravagem cumpriam com os requisitos propostos. Para os testes da distância ao obstáculo, o andarilho foi colocado junto a uma parede e com a ajuda de uma fita métrica, validou-se os valores de segurança impostos. Por fim, para calcular a distância percorrida pelo andarilho, marcou-se no pavimento uma dada distância. Em seguida, percorreu-se esta distância e verificou-se se o valor coincide com o valor apresentado na aplicação móvel.

3

Hardware utilizado e base teórica

Neste capítulo pretende-se descrever o *hardware* utilizado e a sua base teórica, de modo a responder aos objetivos específicos propostos na secção 1.2.
Remarca-se que a escolha dos componentes foi feita através da literatura analisada e influenciada pelos componentes pré-existentes no protótipo anterior, levando em conta a inovação e a redução de recursos monetários.
Na figura 3.1 visualiza-se o protótipo final com a especificação dos componentes que o integram.

Figura 3.1: Protótipo final desenvolvido

Para se chegar a esta fase, dada a atual variedade de componentes existentes no mercado e levando em consideração o pré-requisito existente de reutilizar alguns componentes do protótipo anterior foi necessário uma correta seleção de modo a que este se adequasse ao projeto.
A seleção dos componentes integrantes do protótipo do projeto foi feita em duas fases. Primeiramente, escolheram-se todos os sensores analógicos e digitais e posteriormente, conforme as suas características de consumo, escolheram-se os componentes eletrónicos necessários para o desenho e construção da placa de circuito impresso (PCI), que será abordada no capítulo 4. A descrição dos componentes selecionados na primeira fase encontra-se na tabela A.9 do anexo A.3.1.

3.1 Andarilho

Para este projeto utilizou-se um andarilho dobrável de três rodas maciças, ou também conhecido como andarilho Delta, figura 3.2. Este é composto por duas rodas traseiras, uma roda frontal, um cesto com tabuleiro removível e dois travões mecânicos.

Figura 3.2: Andarilho Delta utilizado [39]

Possui punhos anatómicos, reguláveis em altura, que permitem um bom apoio e fácil acessibilidade ao travão. É adequado tanto para o uso exterior como para espaços interiores reduzidos. Para além dos componentes de origem, o andarilho vem acoplado com o sistema de travagem automática desenvolvido no anterior projeto de dissertação [11].

3.2 Arduino

Para o controlo de todos os sistemas propostos usou-se a placa Arduino Mega 2560, baseada no microcontrolador (MCU) ATmega2560 de 8 bits. Possui 54 pinos de entrada/saída digital (dos quais 15 podem ser usados como saídas PWM), 16 entradas analógicas, 4 UART (portas série de *hardware*), um oscilador de cristal de 16 MHz, uma conexão USB, um conector de alimentação, um conector ICSP e um botão RESET. Na tabela A.1 do anexo A.1 especificam-se algumas das características da placa [40].

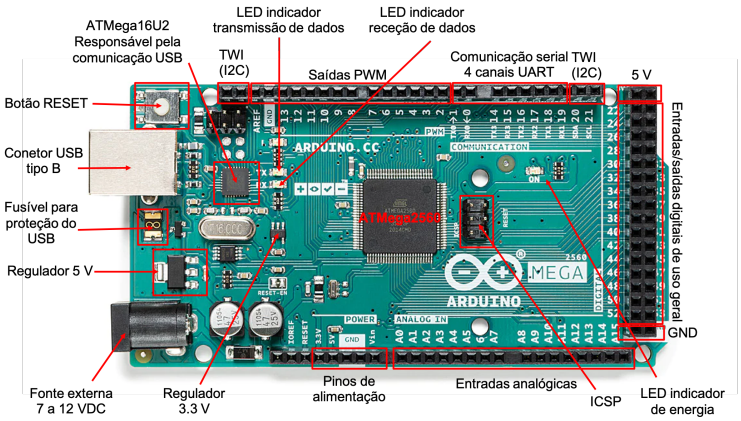


Figura 3.3: Arduino Mega 2560
Fonte adaptada: [40]

3.2.1 Ambiente de desenvolvimento

Para o controlo dos pinos do MCU, utilizou-se o ambiente de desenvolvimento Arduino IDE[1] que é um *software* gratuito e *open source*, que facilita o desenvolvimento e a compilação de código. Através deste, é possível realizar o *upload* do código para a placa em sistemas operacionais Windows, Mac e Linux, demonstrando a sua funcionalidade e versatilidade. Este ambiente suporta as linguagens de programação C e C++. O código inclui uma função `void setup()` que é um conjunto de configurações iniciais que serão válidas para o início do programa e uma função `void loop()` que fará as repetições dos comandos que estão dentro desse bloco de forma infinita ou até o mesmo encontrar algum tipo de interrupção, como por exemplo, falta de alimentação para a placa. O IDE conta também com um monitor de *debug* e um monitor da porta série.

3.2.2 Entradas e saídas

Tal como referido na secção 3.2, o Arduino oferece um conjunto de pinos que permitem a entrada e saída de sinais. Os sinais de entrada são enviados por sensores, enquanto que os de saída permitem definir o comportamento dos atuadores. Estas entradas podem ser utilizadas através das funções `pinMode()`, `digitalWrite()` e `digitalRead()`.

Uma entrada ou saída digital trabalha apenas com os dois valores lógicos 0 (LOW) ou 1 (HIGH). As entradas analógicas trabalham com os valores inteiros entre 0 e 1023 e para obter um sinal de saída inteiro que varia entre 0 e 255, o Arduino utiliza o conceito de *Pulse Width Modulation* (PWM).

O PWM é uma técnica para controlar circuitos analógicos com saídas digitais de um microcontrolador. Ao diminuir ou aumentar a largura do pulso, o controlador regula o fluxo de energia para o sensor [41]. A razão entre o período de pico e o total da onda é chamada de razão de ciclo (*duty cycle*). Assim, variando o *duty cycle*, pode-se imitar uma tensão analógica "média". Quando se pretende a tensão média, mantém-se o *duty cycle* em 50%. Da mesma forma, se se pretender atingir baixa e alta tensão, mantém-se o *duty cycle* em 10% e 90%, respetivamente, tal como se verifica na figura 3.4.

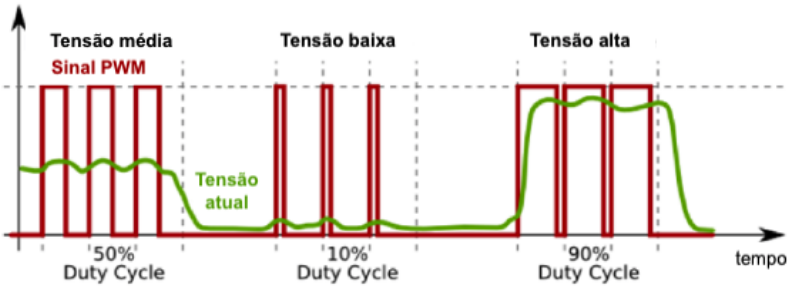


Figura 3.4: Sinais PWM com vários ciclos de trabalho
Fonte adaptada: [42]

[1] `https://www.arduino.cc/en/software`

Podem ser usados 15 pinos (2-13, 44-46) como pinos PWM. Esses pinos são controlados por temporizadores no chip, que alternam os pinos automaticamente a uma taxa de cerca de 490 Hz, com exceção dos pinos 4 e 13 que alternam a uma taxa de cerca de 980 Hz.
Para gerar um sinal PWM, a função `analogWrite()` é chamada.

A função `analogWrite()` usa o número do pino e o valor do pino como seu parâmetro. O valor do pino pode estar entre 0 e 255, com o *duty cycle* mapeado para 0 % e 100 %.

3.2.3 Alimentação

A placa Arduino Mega 2560 pode ser alimentada pela conexão *Universal Serial Bus* (USB) tipo B, sendo este protegido por um fusível de 500 mA, ou com uma fonte de alimentação externa proveniente de um adaptador DC ou bateria com conetor *jack* de 2,1 mm.
A fonte de alimentação é selecionada automaticamente. A placa aceita tensões entre 6 e 20 V. Se lhe for fornecido menos de 7 V, o pino de 5 V pode fornecer menos tensão e a placa pode tornar-se instável. Se lhe for fornecido mais de 12 V, o regulador de tensão pode sobreaquecer e danificar a placa. A faixa recomendada é de 7 a 12 V [40].
Se forem usados os 5 V ou os 3,3 V da placa para alimentar dispositivos externos, deve-se tomar atenção para não ultrapassar os limites de 300 mA para 5 V e 50 mA para 3,3 V.
Se for usado o conetor USB para alimentar o Arduino Mega, recomenda-se que se usem fontes externas para alimentar outros dispositivos. Dessa forma, evita-se mau funcionamento ou até a avaria de algum regulador interno.

3.3 Sistema de travagem automática

Para a atuação dos travões automáticos, manteve-se o sistema implementado no anterior projeto de dissertação [11]. Este é composto por um servomotor de 3 pólos Hitec HSR-5995TG-Ultra Torque em cada roda traseira do andarilho, figura 3.5.

(a) Hitec HSR-5995TG-Ultra Torque [11] (b) Sistema de travagem [11]

Figura 3.5: Sistema de travagem implementado no anterior projeto de dissertação

A posição do braço do motor é controlada com um sinal no formato PWM de 50 Hz e pulsos com largura de 1 ms até 2 ms. Variando-se a largura do pulso a posição pode ser variada até 180°, figura 3.6.

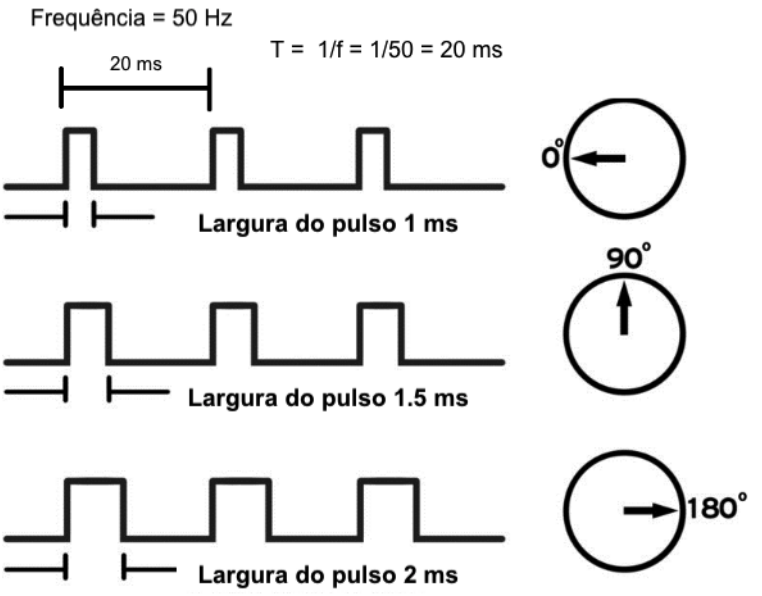


Figura 3.6: Ciclo de trabalho PWM do servo motor e requisito de frequência
Fonte adaptada: [43]. A figura não está à escala.

Assim, no período de 20 ms e ciclo de trabalho de 2 ms, o braço do servo motor move-se para 180°, no ciclo de trabalho de 1,5 ms para 90° e no ciclo de trabalho de 1 ms para 0°, figura 3.6.

As suas especificações técnicas estão descritas na tabela A.2 do anexo A.1.

3.4 Sistema de deteção de movimento das rodas e limite de velocidade imposto ao andarilho

Para este objetivo foram reutilizados os sensores de Hall utilizados no projeto anterior de dissertação [11]. O modelo dos sensores de Hall é A1101 da Allegro são do tipo unipolar. Estes operam na presença de um campo magnético positivo, ou seja, o sensor conduz apenas quando o polo sul de um íman se aproxima dele [44].

Figura 3.7: Sensor de Hall Allegro A1101
Fonte: [44]

No protótipo de Cavaleiro, foram colocados doze ímanes em volta de cada roda de forma a se

medir a velocidade e o sentido de rotação [11].

De modo a melhorar a resolução dos valores da velocidade instantânea e da distância percorrida, os ímanes foram substituídos por vinte e seis ímanes de neodímio de 8 mm × 1 mm, modelo N851S da ECLIPSE MAGNETICS, espaçados igualmente entre si. Com esta alteração conseguiu-se um aumento de resolução para mais do dobro.

Mantiveram-se os dois sensores de Hall em cada roda e foram colocados de modo a criar um ângulo perpendicular com os ímanes, figura 3.8.

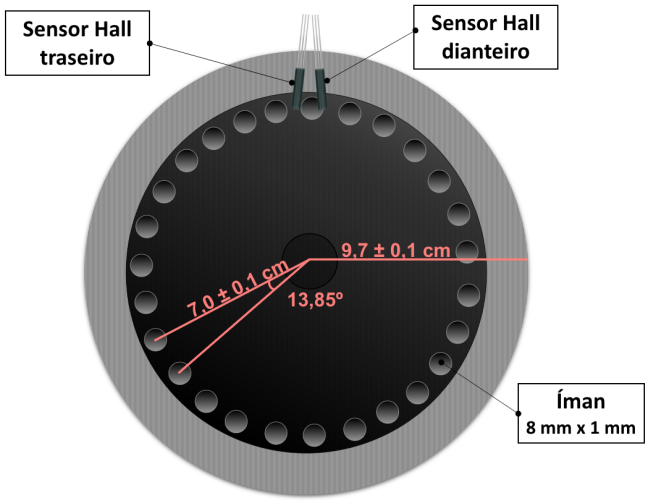
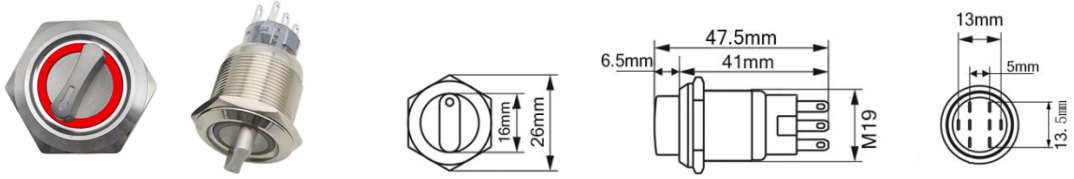


Figura 3.8: Sistema de deteção de movimento das rodas

No protótipo anterior, havia uma imposição fixa da velocidade máxima do andarilho, sendo esta de 1 km/h. De modo a responder às diferentes necessidades das pessoas com compromisso na deambulação, utilizou-se um seletor rotativo de três posições com retenção. Este possibilita limitar a velocidade máxima do andarilho para três valores de velocidade distintos, mínima, média e máxima. Ao girar o seletor para a esquerda, impõe-se a velocidade mínima, na posição central a velocidade média e para a direita a velocidade máxima. Este seletor de 19 mm tem um *Light-Emitting Diode* (LED) vermelho à sua volta, tornando-o assim mais notório. A sua tensão de operação é de 3 a 5 V e é controlado através dos níveis lógicos 0 (LOW) e 1 (HIGH), figura 3.9. Este foi integrado no tampo superior da caixa mecânica que aloja o circuito elétrico do protótipo.



(a) Seletor rotativo [45] (b) Dimensões do seletor rotativo [45]

Figura 3.9: Seletor rotativo de três posições em aço inoxidável

3.5 Sistema de deteção de obstáculos

Para o sistema de deteção de obstáculos usou-se quatro sensores infravermelhos capazes de medir a distância a que se encontra um dado obstáculo ou irregularidade no pavimento. Nas laterais e na dianteira do andarilho acoplaram-se sensores Sharp GPY0A21YK0F, sendo que a sua faixa de medição é entre 10 a 80 cm. Para responder atempadamente a uma deteção da irregularidade do pavimento, como por exemplo uma depressão ou desnivelamento, colocou-se obliquamente na parte inferior do cesto do andarilho o modelo Sharp GP2Y0A02YK0F, cuja faixa de medição é de 20 a 150 cm, figura 3.10.

(a) Sharp GP2Y0A02YK0F [46] (b) Sharp GP2Y0A21YK0F [47]

Figura 3.10: Sensores de distância infravermelhos

As especificações técnicas de cada um estão descritas na tabela A.3 do anexo A.1.

Estes sensores têm o seu funcionamento baseado no princípio da triangulação. Um feixe de luz infravermelho é emitido por um emissor e propaga-se até encontrar um objeto. Ao encontrar um obstáculo, esta luz é refletida e é detetada por um detetor sensível à posição, formando-se um triângulo entre o emissor, o ponto de reflexão e o detetor do sensor, conforme a figura 3.11. De acordo com a distância do objeto que refletiu a luz, esse raio incide de modo diferente no sensor de posição sensível (PSD).

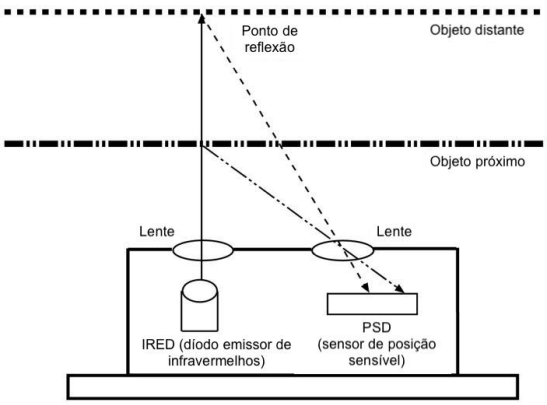
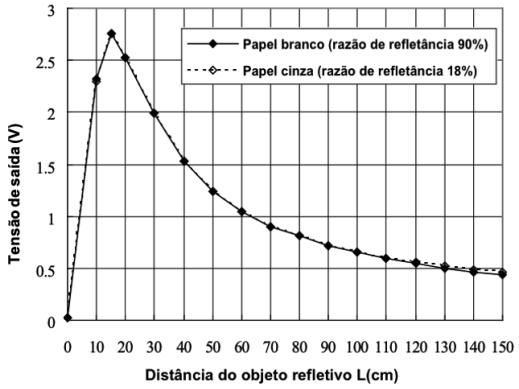


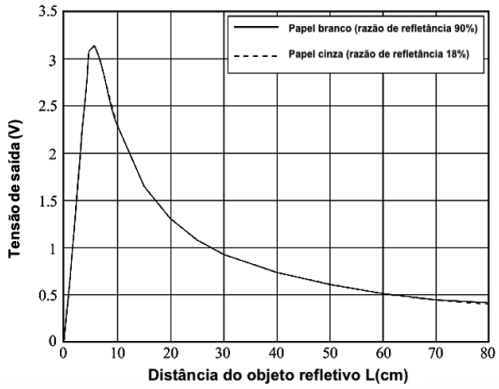
Figura 3.11: Diferentes ângulos com diferentes distâncias medidos por um Sharp IR
Fonte adaptada: [48]

Este método oferece grande imunidade à variação de cor do obstáculo, porém a relação entre a tensão de saída e a distância torna-se não linear, apresentando uma curva logarítmica, como

mostram as figuras 3.12a e 3.12b obtidas dos *datasheets* dos componentes. Para distâncias fora da faixa de operação do sensor, a saída deve ser rejeitada.
Estes tipos de sensores de distância tendem a ser um pouco ruidosos, por isso é recomendável adicionar um condensador entre `VCC` e `GND`. O *datasheet* sugere um condensador de 10 μF ou mais.



(a) Sharp GP2Y0A02YK0F [46]



(b) Sharp GP2Y0A21YK0F [47]

Figura 3.12: Curvas características da saída analógica dos sensores em função da distância ao objeto

3.6 Sistema de alarme sonoro e buzina

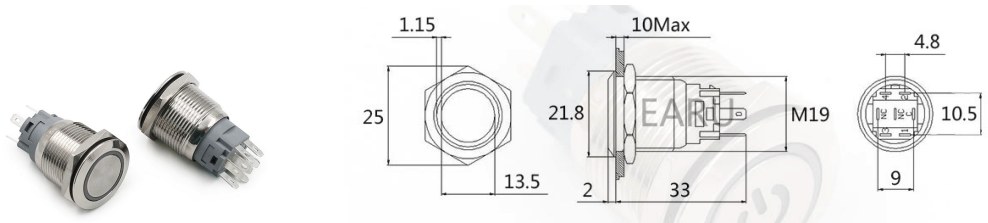
Para o sistema de alarme sonoro e buzina foi utilizado um piezo buzzer e um botão de pressão. Existem dois tipos de buzzers, ativos e passivos. O buzzer ativo tem incorporado um circuito oscilador que emite som diretamente sem necessidade de um circuito externo e basta ligá-lo diretamente a uma bateria DC. O buzzer passivo necessita de um sinal de corrente alternada (AC) para produzir som, isto é, o som é produzido pela alteração constante de um sinal de entrada.
Para este projeto, a escolha recaiu no piezo buzzer ativo THDZ de 12 mm, figura 3.13, e foi integrado na camada superior da PCI.

Figura 3.13: Piezo buzzer TMB12A05
Fonte adaptada: [49]

As suas especificações técnicas estão descritas na tabela A.4 do anexo A.1.

Para a buzina integrou-se um botão de pressão que é um componente que liga ou desliga dois pontos do circuito elétrico quando é pressionado.
Existem dois tipos de botão de pressão, o momentâneo e com retenção. A principal diferença entre

eles é que o botão de pressão com retenção não retorna para a posição inicial após pressionado e, portanto, deve-se pressioná-lo novamente para voltar ao estado de posição inicial. O botão momentâneo permanece ligado ou desligado enquanto o mesmo for pressionado.



(a) Botão de pressão [50] (b) Dimensões do botão de pressão [50]

Figura 3.14: Botão de pressão momentâneo em aço inoxidável

O botão escolhido é do tipo momentâneo de 19 mm com LED RGB em aço inoxidável, figura 3.14. O mesmo foi fixado no tampo superior da caixa de proteção do circuito. A sua tensão de operação é de 3 a 5 V.

3.7 Sistema de deteção de força nos punhos do andarilho

Para a deteção de força aplicada pela pessoa em cada punho do andarilho, utilizou-se dois sensores Flexiforce A201.

Este sensor é um sensor piezoresistivo de circuito impresso ultrafino, flexível e lê forças que são perpendiculares ao plano do sensor. São construídos por duas camadas de filme de substrato. Em cada camada é aplicado um material condutor, prata, seguido de uma camada de tinta sensível à pressão. A prata estende-se da área de deteção até aos conetores na outra extremidade do sensor. A área de deteção tem 9,53 mm de diâmetro e localiza-se na ponta direita da figura 3.15. No fim, o sensor termina com três pinos machos quadrados soldáveis para serem incorporados ao circuito. Os dois pinos externos do conetor estão ativos e o pino central está inativo [51].

Figura 3.15: Flexiforce A201
Fonte adaptada: [52]

À medida que se aplica uma dada força na área de deteção, o valor da resistência lido irá alterar de forma inversamente proporcional à força exercida.

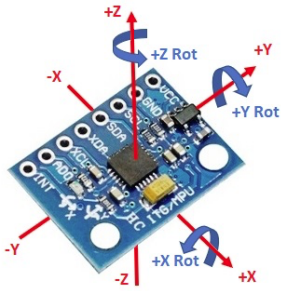
Estes sensores exibem quantidades consideráveis de histerese e erro de repetibilidade que inibem seu uso em aplicações que exigem leituras de força altamente precisas. Contudo, para o tipo de aplicação estes sensores demonstraram ser adequados.

Os sensores foram colocados debaixo de cada uma das pegas do andarilho.

3.8 Sistema de deteção de queda da pessoa ou do andarilho

Para este efeito utilizou-se um interruptor de emergência *kill-switch*, figura 3.16a, e um módulo acelerómetro-giroscópio de 6 eixos, figura 3.16b.

(a) Interruptor de emergência *kill-switch* [53]



(b) Eixos do módulo GY-521 MPU-6050 [54]

Figura 3.16: Componentes utilizados para a deteção de queda do andarilho e da pessoa com dependência

O interruptor de emergência é um dispositivo simples de segurança que depende de um botão de mola para manter o interruptor aberto, isto é, desligado. Consiste num cordão resistente extensível e espiralado, e numa das extremidades tem um clipe que é prendido à pessoa com dependência. Caso ocorra uma queda, o cordão é puxado e o interruptor é ligado.
Devido à sua alta extensibilidade, o cordão foi substituído por um fio de aço. O interrutor foi prendido ao lado da pega direita do andarilho.

O MPU-6050 combina um acelerómetro de 3 eixos e um giroscópio de 3 eixos com um processador de movimento digital (DMP) integrado no módulo GY-521, figura 3.16b.
Tanto o acelerómetro quanto o giroscópio usam três entradas ADCs de 16 bits com quatro faixas programáveis para alta sensibilidade. Um sensor de temperatura embutido também é fornecido para medir a temperatura do chip. As comunicações entre o sensor e o MCU são feitas pela interface *Inter-Integrated Circuit* (I2C) e suporta dois endereços diferentes: 0x68 e 0x69. Isto permite que dois dispositivos sejam usados no mesmo barramento ou caso haja conflito de endereço com outro dispositivo.
O sensor MPU-6050 opera a 3,3 V mas o módulo GY-521 contém um regulador de tensão de 3,3 V, então o módulo deve ser alimentado com 5 V no pino `VCC`.
O módulo GY-521 MPU-6050 foi integrado na camada superior da PCI e as especificações técnicas estão descritas na tabela A.5 do anexo A.1.

3.9 Sistema de iluminação noturna

Para este objetivo optou-se por utilizar uma fita LED RGB analógica, figura 3.17a, e um sensor *Light Dependent Resistor* (LDR), figura 3.17b.

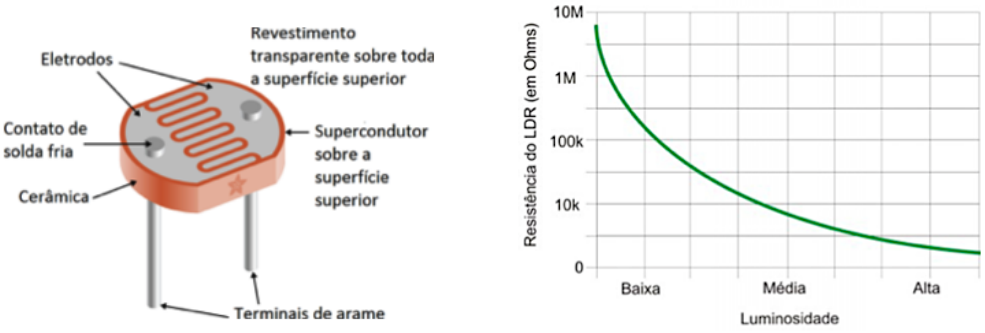
(a) Fita LED RGB HB-108005 [55]

(b) LDR GL5528 [56]

Figura 3.17: Componentes utilizados para o sistema luminoso

As fitas de LED são placas de circuito impresso flexíveis nas quais estão soldados os díodos emissores de luz para montagem em superfície assim como outros elementos necessários para o funcionamento dos mesmos.
Existem dois tipos básicos de fitas LED, do tipo analógico e do tipo digital, de várias cores e potências. As analógicas têm todos os LEDs conetados em paralelo, pelo que a cor dos mesmos não pode ser controlada individualmente. As digitais, ou endereçáveis, têm um chip para cada LED, podendo ser controlada a cor de cada um individualmente.
Quanto à cor existem três opções, fita de LED do tipo branco frio que emite uma luz branca e marcante, do tipo amarelado ou branco quente e, por fim, as fitas de LED coloridas ou RGB. O modelo RGB possui as três cores primárias, o vermelho (Red), o verde (Green) e o azul (Blue).
Todas as fitas são constituídos por segmentos de 10 cm e cada um composto por 3 LEDs.
Para este projeto utilizou-se duas fitas de LED do tipo RGB de 12 V, referência HB-108006, com 40 cm de comprimento cada. Cada segmento de 3 LEDs consome aproximadamente 20 mA por cor. A combinação das três cores primárias origina a cor branca, levando a um consumo de corrente máxima de 60 mA por segmento. Assim, o consumo total máximo das duas fitas é de 480 mA. A intensidade da sua cor é regulada através da técnica PWM, explicada na secção 3.2.2. Estas fitas são à prova de água e têm fita adesiva atrás, o que permitiu que fossem coladas na parte inferior da base do andarilho.

O LDR é um componente que varia a sua resistência conforme o nível de luminosidade que incide sobre ele e a sua resistência varia de forma inversamente proporcional à quantidade de luz incidente [57].



(a) Estrutura física de um LDR [57] (b) Curva característica de um LDR [57]

Figura 3.18: Estrutura física e curva característica de um LDR

O mesmo foi colocado no tampo superior da caixa protetora do circuito e as especificações técnicas estão descritas na tabela A.6 do anexo A.1.

3.10 Sistema de comunicação com a nuvem

Para o sistema de comunicação com a nuvem foi escolhido o módulo Wi-Fi ESP8266 ESP-01. Este é constituído pelo MCU ESP8266, com um chip de memória *flash* externo, utilizado no armazenamento de programas e de arquivos, e de uma antena impressa na própria placa.

A comunicação entre o Arduino Mega 2560 e o Wi-Fi ESP8266 ESP-01 é feita através da comunicação UART.

Podem ser utilizados até quatro GPIOs, porém duas estão livres e as outras duas compartilham a função de comunicação série, sendo uma `TX` e outra `RX`.

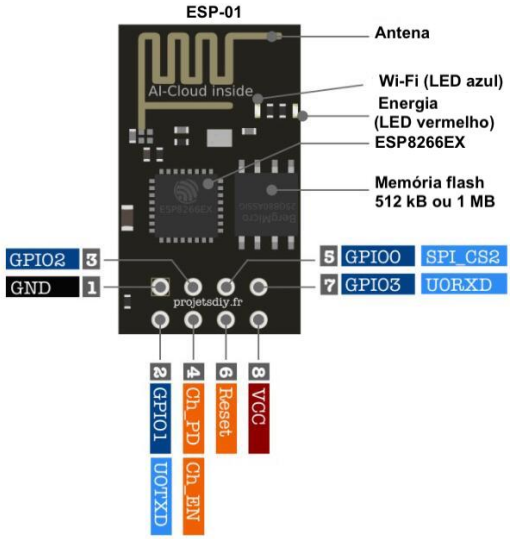


Figura 3.19: Descrição das funções dos pinos do módulo Wi-Fi ESP8266 ESP-01
Fonte adaptada: [58]

É de notar que a alimentação do módulo a partir do Arduino é de 3,3 V, portanto para evitar

danos ao módulo é necessário reduzir a tensão de 5 V para 3,3 V através de um divisor de tensão. Todo o processo de alimentação e montagem do circuito externo para a programação da placa são explicados na secção A.2 do anexo A.1.
Por fim, a placa Wi-Fi ESP8266 ESP-01 é encaixada a um conetor próprio soldado na camada superior da PCI.
As suas especificações técnicas estão descritas na tabela A.7 do anexo A.1.

3.11 Nível de carga da bateria

De forma a monitorizar o nível de carga da bateria, utilizou-se um ecrã de carga de baterias, figura 3.20. Assim, o utilizador evita a descarga total da mesma, contribuindo para o aumento da sua vida útil.

(a) Ecrã de nível de carga da bateria [59] (b) Percentagem e nível de carga da bateria [59]

Figura 3.20: Estrutura física do ecrã de nível de carga da bateria

Para configurar este ecrã é necessário pressionar e segurar o botão que se encontra na parte traseira do mesmo e com o botão pressionado conecta-se a bateria a ser testada.
Em seguida, solta-se o botão e pressiona-se várias vezes até que o tipo de bateria a ser testado seja exibido. Por fim, desconecta-se e reconecta-se à bateria.
Neste ponto visualiza-se a percentagem e o nível de carga da bateria, tal como se pode verificar na figura 3.20b.
O ecrã foi colocado no tampo superior da embalagem mecânica do produto e as suas especificações técnicas estão descritas na tabela A.8 do anexo A.1.

4

Desenvolvimento da placa de circuito impresso

A placa de circuito impresso (PCI) é uma alternativa à utilização das tradicionais placas de ensaio utilizadas para a fixação de componentes eletrónicos. Os circuitos adaptados nela podem funcionar perfeitamente durante anos, contudo são considerados frágeis e muito suscetíveis a captação de ruído.

Atualmente, as PCI são amplamente utilizadas em indústria eletrotécnica, uma vez que é uma peça imprescindível para o funcionamento de dispositivos eletrónicos. A sua construção é baseada numa placa de um material isolante e uma, duas ou mais camadas, revestidas ou separadas por uma película de cobre. As películas de cobre são utilizadas para efetuar os caminhos condutores, onde em conjunto com os componentes eletrónicos soldados também nesta película de cobre formam os circuito eletrónicos .

O desenho de uma PCI é baseado no diagrama esquemático do circuito, o qual é feito num *software* adequado. Para este projeto foi utilizado o *software* Altium Designer[1] que abrange quatro áreas funcionais principais, incluindo a captura esquemática, *design* 3D da PCI, desenvolvimento de matriz de portas programáveis em campo (FPGA) e gestão de dados. Integra-se com vários distribuidores de componentes para acesso aos dados do fabricante e possui edição 3D interativa da placa e exportação para ficheiros 3D MCAD para STEP.

4.1 Esquemáticos das ligações elétricas dos componentes

Um dos passos fundamentais para o fabrico e montagem da PCI é a conceção dos diagramas esquemáticos, que fornecem uma visão geral dos componentes e dos respetivos circuitos necessários para o seu funcionamento. Os circuitos elétricos de cada componente selecionado para a conceção do projeto estão especificados no anexo A.4.1.

Durante o desenho do circuito elétrico de cada módulo foi necessário proceder à escolha dos componentes resistivos e capacitivos destes circuitos. Na escolha destes componentes levou-se em consideração os requisitos elétricos, físicos e custos monetários dos mesmos. A tabela A.4.3 especifica todos os componentes escolhidos para o desenvolvimento da PCI.

Remarca-se ainda que para a tensão escolhida, a corrente máxima de cada motor é de 4,2 A. Contudo, o torque máximo nunca é atingido por isso cada motor foi limitado a 1,5 A através da inserção de um fusível entre a entrada `VCC` do servomotor e a alimentação de 6 V.

Seguidamente foi necessário avaliar as tensões exigidas por todos os módulos constituintes do produto, anexo A.3, e elaborar um esquema de fornecimento de energia. Verificou-se que houve a necessidade de vários níveis de tensão para os diferentes módulos que compõem o produto. Estes níveis de tensão são obtidos através de conversão e regulação por meio de diferentes reguladores de tensão.

A abordagem seguida pode ser vista no diagrama da figura 4.1, onde apresenta a forma como a

[1] `https://www.altium.com/products/downloads`

distribuição dos diferentes níveis de energia é feita para os diferentes módulos do produto.

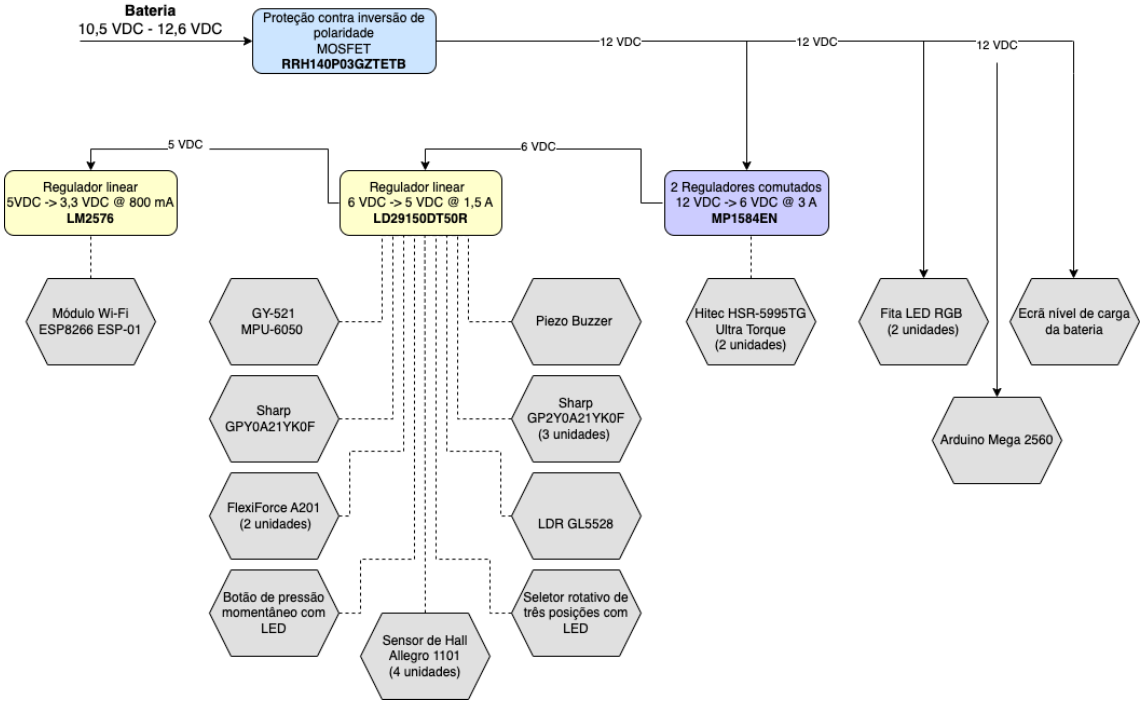
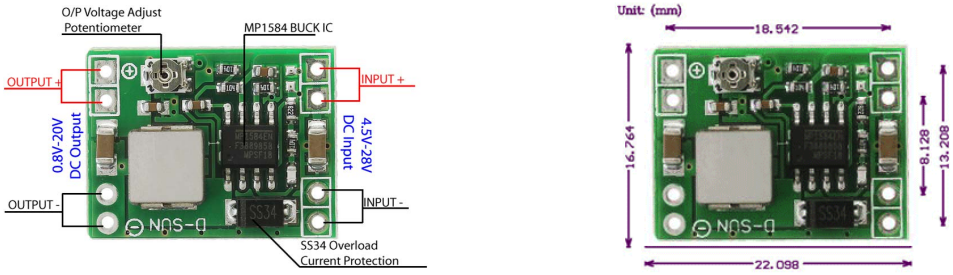


Figura 4.1: Diagrama de alimentação do produto

Para a escolha destes reguladores teve-se em conta a corrente exigida em cada nível de tensão e a corrente máxima que o regulador suporta. De forma a que o regulador não trabalhasse no seu limite, escolheu-se um regulador com uma corrente máxima de pelo menos o dobro da corrente exigida no nível de tensão. A corrente exigida no nível de tensão de 3,3 V é 170 mA, no de 5 V é de 554,90 mA e no de 6 V de 3554,90 mA. Por fim, as LEDs e o ecrã de nível de carga de bateria (12 V) necessitam de 485 mA.

Em termos de consumo, a potência total é de 27,144 W, pois no ramo de 6 V tem-se um consumo de 21,324 W e no ramo de 12 V o consumo é de 5,820 W.

Para a alimentação principal, escolheram-se dois reguladores comutados MP1584EN que convertem a tensão de entrada para 6 VDC e com capacidade de fornecer 3 A cada, um alimenta um dos servos e o outro alimenta a linha de tensão de 6 V e o segundo servomotor. Esta escolha recaiu num regulador comutado devido à elevada corrente e elevada diferença de tensão entre a entrada e a saída. Apesar deste conversor suportar uma tensão de entrada de 28 VDC, todo o circuito foi desenhado para suportar 12 VDC.



(a) *Pinout* do módulo

(b) Dimensões do módulo

Figura 4.2: *Pinout* do conversor *buck* DC-DC MP1584EN e as suas dimensões
Fonte: [60]

Posteriormente, foi introduzido nesta linha de tensão um LDO (LD29150DT50R) para converter os 6 VDC para 5 VDC, com capacidade de fornecer 1,5 A.
Pode observar-se um segundo regulador *Low-Dropout Regulator* (LDO) (LM2576) que regula 5 VDC para 3,3 VDC para fornecer energia ao módulo de Wi-Fi. O mesmo tem capacidade de fornecer 800 mA.
A fonte de alimentação utilizada é uma bateria recarregável de íões de lítio DC 12680, figura 4.3, inclui um conversor AC/DC externo, que fornece ao dispositivo uma tensão DC máxima de 12,6 V, e com capacidade de oferecer 6,8 Ah [61].

Figura 4.3: Bateria recarregável de lítio-íon DC 12680
Fonte: [61]

Projetou-se também a proteção contra inversão de polaridade, usando um MOSFET P-Channel RRH140P03GZETB como dispositivo de bloqueio. Este possui uma tensão de rutura entre o dreno e a fonte de 30 V e uma corrente de condução que pode chegar a 14 A @ 25ºC. A tensão V_{GS} deve ser de no máximo -20 V. Ao utilizarmos este PMOS, e tendo uma alimentação de 12 V e uma corrente $I_{Total} = 4,17$ A, o transístor irá operar na região de tríodo com uma resistência $R_{DS_{ON}}$ de 7 mΩ, pelo que a queda de tensão no PMOS será de $V_{DS} = 0,007 \times 4,17 = 0,03$ V e a potência dissipada será de $P_D = 0,007 \times (4,17)^2 = 0,12$ W.

4.2 Desenho e montagem da placa de circuito impresso

O primeiro passo do desenho da PCI consistiu no formato da *shield* do Arduino. De forma a manter o custo baixo da PCI, a mesma foi desenhada para conter apenas duas camadas, *TOP* e *BOTTOM*, onde cada uma delas contém um plano de potência e diversos sinais. O desenho 2D destas camadas encontra-se no anexo A.4.2

A camada *TOP* contém um plano de *power* e é alimentada a 5 V. A camada *BOTTOM* contém um plano de *ground*.

Em seguida organizaram-se todos os componentes provenientes dos esquemáticos nas camadas da placa. Estes foram agrupados na mesma região da PCI, ajudando a manter os caminhos condutores mais curtos, pois caminhos longos podem captar e radiar radiação eletromagnética que pode levar a ruído e interferências. Os caminhos por onde passavam correntes elevadas foram engrossados.

Por fim, utilizou-se a ferramenta de verificação de determinadas regras a serem cumpridas no desenho da PCI, tais como o mínimo de espaçamento, mínimo de isolamento, mínimo de separação entre componentes, mínimo do tamanho das pistas, entre outros. Também verifica se não existem pistas de sinais diferentes ainda não conectadas desde o seu ponto de início ao ponto de destino (e que estejam conectadas no esquemático) e se existem pistas de sinais diferentes sobrepostas. Esta ferramenta está integrada no *software* utilizado.

4.3 Soldagem e testes do funcionamento da placa

Após a impressão da placa e compra de todos os componentes necessários para o desenho da mesma, o passo seguinte foi a soldagem dos mesmos.

Os primeiros componentes a soldar foram os reguladores e em seguida testaram-se os níveis de tensão à entrada e à saída dos mesmos.

Após se verificar que estes têm os níveis de tensão certos, passou-se à soldagem dos condensadores, fusíveis, resistências, mosfets, díodos e entrada *jack* para a alimentação da *shield*.

A maioria dos componentes utilizados foram do tipo *Surface Mounted Device* (SMD) e por uma questão de organização, foram soldados na camada *BOTTOM* sempre que possível. Na camada *TOP* foram usados os componentes *Pin Through Hole* (PTH) e as fichas *jst*, onde encaixam todos os sensores do projeto.

À cablagem de todos os componentes mecânicos adicionaram-se fichas *jst*.

4.4 Caixa de proteção do circuito

Após a conceção da PCI houve a necessidade de se desenhar a embalagem do produto.

A caixa apresenta um perfil baixo e compacto, de forma a poder ser colocada em cima do tabuleiro do andarilho. O seu interior contém a PCI e a bateria e dos lados apresenta furos de refrigeração e

de entrada dos sensores, figura 4.5.

Figura 4.4: Vista frontal da embalagem final

A mesma foi desenhada no Fusion 360[2], que é uma plataforma de software CAD, CAM, PCB e de modelação 3D baseada na nuvem para o *design* e fabrico de produtos.
O render 3D e as dimensões da caixa estão apresentadas no anexo A.5.

4.5 *Hardware* final

Em seguida mostram-se algumas imagens da placa de circuito impresso finalizada. É de realçar que a alimentação do sistema é feita pela nova entrada DC de 12 V.

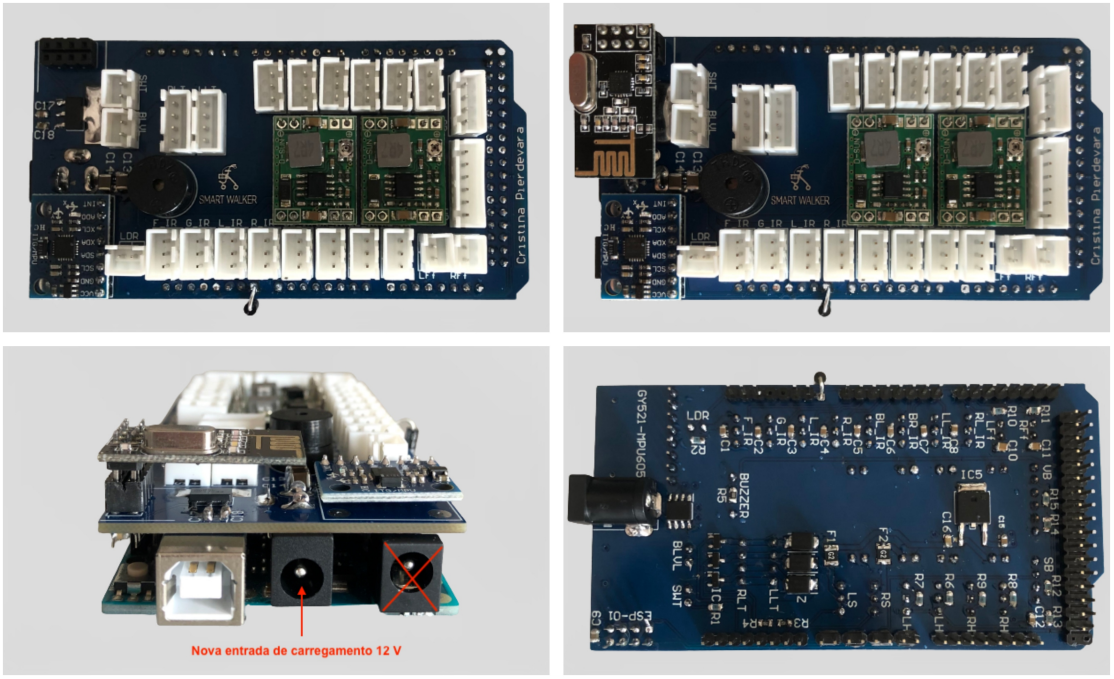


Figura 4.5: Placa de circuito impresso real

Após a conceção da placa verificou-se que a *shield* não podia estar encaixada no Arduino

[2] https://www.autodesk.pt/products/fusion-360/overview?term=1-YEAR&tab=subscription

Mega 2560 durante a programação do mesmo, pois a tensão proveniente do computador iria alimentá-la com uma tensão inferior, provocando danos aos reguladores. Assim, entre a entrada V_{in} do Arduino e o pino 12 V da *shield* foi inserido um díodo que permite programar a placa Arduino sem que a *shield* fosse alimentada.

Remarca-se que esta modificação está atualizada tanto no *software* como nos esquemáticos e camadas apresentadas nos anexos.

5

Software dos sistemas

A seguir irá ser explicado todo o *software* implementado relativo a todos os sistemas desenvolvidos e à programação da placa Wi-Fi ESP8266 ESP-01. O código fonte desenvolvido para o SMARTWALKER encontra-se no anexo A.7.1 e para a placa Wi-Fi no anexo A.7.2.

5.1 Código fonte do SMARTWALKER

Como mencionado na secção 3.2.1, o ambiente de desenvolvimento Arduino apresenta duas funções especiais, `void setup()` e `void loop()`. Antes destas duas funções incluem-se as bibliotecas necessárias para o funcionamento dos módulos, definem-se os pinos e declaram-se as variáveis globais necessárias para o desenvolvimento do código.
Em seguida, a função `void setup()` é chamada uma única vez e é nela que se configura a taxa de transferência para a transmissão série, anexam-se os pinos aos servos, inicializa-se o módulo MPU-6050 e definem-se pinos como pinos de entrada ou saída.
A função `void loop()` é chamada repetidamente e chama todas as funções desenvolvidas para o funcionamento dos sistemas propostos.

```
1 void loop(){
2   interruptLeftVelocity();    // Para calcular a velocidade da roda esquerda a
        cada meio segundo
3   interruptRightVelocity();   // Para calcular a velocidade da roda direita a
        cada meio segundo
4   walkerSpeed();              // Define a velocidade maxima do andarilho em
        funcao do botao de velocidade
5   walkerHorn();               // Buzina ON ou OFF
6   lightSystem();              // Avisos luminosos e luzes de presenca
7   fallSystem();               // Deteta se ocorreu a queda da pessoa ou do
        andarilho
8   objectDetection();          // Deteta obstaculos e trava proporcionalmente a
        medida que a distancia diminui
9   sendToESP();                // Envia eventos para o modulo Wi-Fi (ESP8266 ESP
        -01)
```

Também é nela que se faz a comparação dos ângulos de travagem determinados pelos vários sistemas, após serem atualizados pelas várias funções. Se algum sistema pretender travar as rodas é aplicado aos ângulos de destravagem um ângulo de ±40º, dependendo da roda.

```
1   // Compara qual dos angulos angDistMax, angVel ou angFall e maior de forma a
        determinar o angulo certo a ser aplicado
2   if(angVel > angDistMax && !fallDetected){
3     // Angulo final aplicado: unlockAngle + angVel
4     leftServo.write(LEFT_UNLOCK_ANG + angVel);
5     rightServo.write(RIGHT_UNLOCK_ANG - angVel);
6   }
7   else if(fallDetected){
8    // Angulo final aplicado: unlockAngle + angFall
9     leftServo.write(LEFT_UNLOCK_ANG + angFall);
10    rightServo.write(RIGHT_UNLOCK_ANG - angFall);
11  }
12  else if(objectDetected){
13    // Angulo final aplicado: unlockAngle + (angDistLeft ou angDistRight)
14    leftServo.write(LEFT_UNLOCK_ANG + angDistLeft);
15    rightServo.write(RIGHT_UNLOCK_ANG - angDistRight);
16  }
17  else servoUnlock();
18 }
```

Dado o facto do Arduino Mega 2560 ter apenas um núcleo de processador, todas as tarefas devem ser executadas sequencialmente. Normalmente, para aguardar um período de tempo usa-se a função `delay()` e enquanto esta não for concluída, nenhuma outra instrução é executada.
Como o sistema implementado tem várias tarefas que devem responder em tempo real, este não pode ser pausado.
O problema foi contornado através da função `millis()` que retorna o tempo, em milissegundos, decorrido desde que a placa Arduino ligou até a tarefa atual.
Primeiramente, guarda-se o valor da função `millis()` em variáveis para cada atraso necessário. Em seguida, no início de cada ciclo calcula-se a diferença de tempo entre as variáveis armazenadas e o tempo atual, sendo assim possível verificar se já passou o tempo necessário para que uma tarefa seja executada.
Após a execução da tarefa atualiza-se o valor das variáveis de atraso com o valor retornado pela função `millis()` e o ciclo recomeça.
Este método foi usado sempre que foi necessário controlar o tempo entre dois acontecimentos.

5.2 Sistema de travagem

Para este objetivo foi necessário criar três funções, `void servoLock()`, `void servoUnlock()` e `void servoSoftLock_velocity()`. Estas funções aplicam aos servos os ângulos necessários para a travagem máxima, destravagem ou para o controlo proporcional à velocidade máxima do andarilho.
Para controlar os servos é necessário incluir a biblioteca `Servo.h` que tem as funções necessárias para o seu funcionamento. Esta usa os *Timers* 1 e 2 do Arduino Mega 2560 e são responsáveis por criar os sinais *Pulse Width Modulation* (PWM) para o controlo da posição de cada um.
A função `void servoLock()` impõe o ângulo de travagem máximo a cada roda. Para a roda esquerda o ângulo é de 120º e para a roda direita é de 80º.
A função `void servoUnlock()` aplica o ângulo de desbloqueio a cada roda. Para a roda esquerda o ângulo é de 80º e para a roda direita é de 105º.

Para uma travagem mais suave, implementou-se a função `void servoSoftLock_velocity()`. Esta efetua a travagem de forma proporcional à velocidade. Tem como parâmetros de entrada a velocidade máxima imposta ao andarilho (`maxVelocity`) e a velocidade de cada uma das rodas, (`leftVelocity`) e (`rightVelocity`). Como parâmetro de saída tem-se o ângulo de travagem a ser aplicado (`angVel`).
Quando a velocidade máxima do andarilho é excedida o ângulo de travagem é máximo. Se a velocidade for inferior a 50% da velocidade máxima, a travagem automática não atua e, se esta velocidade for acima dos 50% e não exceder a velocidade máxima imposta ao andarilho, faz-se um controlo proporcional à velocidade, figura 5.1, isto é, os motores travam progressivamente até atingirem o ângulo máximo de ±40º.

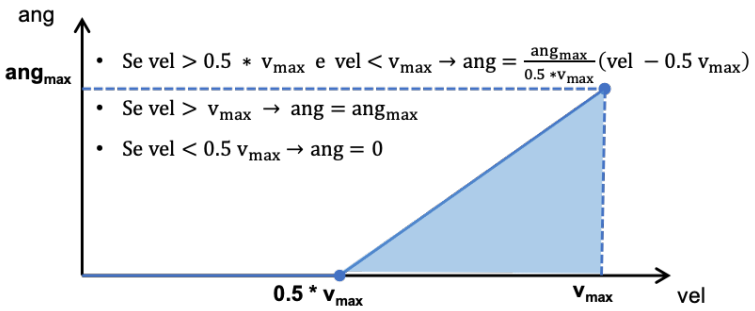


Figura 5.1: Controlo proporcional à velocidade

```
void servoSoftLock_velocity(){ // A travagem automatica atua de forma
    proporcional a velocidade
  // Permite que o andarilho gire e leve em conta a diferenca de velocidade
      entre as duas rodas
  float vel = (leftVelocity + rightVelocity)/2;
  if(vel > maxVelocity) angVel = LOCK_ANG;  // trava
  else if(vel < VMAX_THRESHOLD * maxVelocity) angVel = 0;   // Destrava
  else angVel = ((vel - VMAX_THRESHOLD * maxVelocity) * LOCK_ANG) / ((1 -
      VMAX_THRESHOLD) * maxVelocity);
}
```

5.3 Sistema de deteção de obstáculos

Como mencionado na secção 3.5, os sensores de distância infravermelhos têm uma resposta não linear. Assim, para determinar a distância entre o sensor e um objeto, utiliza-se um modelo de regressão não linear com recurso a uma função de potência. A distância em função da tensão analógica do sensor é dada através da fórmula 5.1.

$$y = \beta \times x_i^a + \varepsilon_i \tag{5.1}$$

Onde y é o valor da distância obtido através do ajuste da tensão de saída do sensor a uma função de potência, x_i é o valor da tensão de saída do sensor e ε_i é o erro, ou seja, a diferença entre o valor da amostra e o valor do modelo.

Assim, utilizou-se a biblioteca `Sharp IR`[1] desenvolvida por Guillaume Rico e Thibaut Mauon. Esta inclui as fórmulas necessárias para converter a tensão de saída medida numa distância em centímetros. Os autores aproximaram as curvas características dos sensores GP2Y0A02YK0F e GP2Y0A21YK0F através das fórmulas 5.2 e 5.3, respetivamente.

$$distanceCM = 12.08 \times analogRead^{-1.058} \tag{5.2}$$

$$distanceCM = 27.728 \times analogRead^{-1.2045} \tag{5.3}$$

Os autores da biblioteca notaram que as leituras do sensor podem flutuar, mas esse problema é colmatado pelas várias leituras seguidas, descartando valores discrepantes e calcula a média para

[1] `https://github.com/guillaume-rico/SharpIR`

obter uma leitura de distância mais estável.

Para a deteção de obstáculos ou pavimentos irregulares implementou-se a função `void objectDetection()`.

Se algum objeto ou irregularidade no pavimento for detetado, a travagem ocorre de forma inversamente proporcional à distância, isto é, à medida que a pessoa se aproxima do obstáculo, o ângulo aplicado aos servos é cada vez maior. Se o ângulo resultante for maior que o ângulo de travagem máximo, então este é limitado a ±40º, figura 5.2.

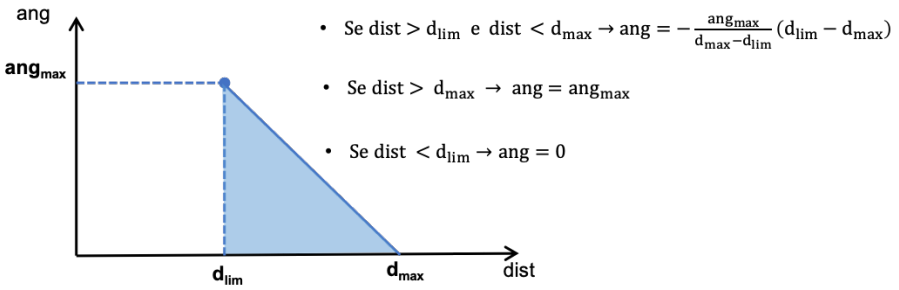


Figura 5.2: Controlo inversamente proporcional à distância do objeto

Começa-se por se fazer a leitura dos sensores de distância. Para um ganho na performance, esta leitura é feita de forma alternada.

```
// Com bool firstReading: ciclos = 512; Sem: ciclos passam para metade
  if(firstReading){
    firstReading = false;
    frontalIR_dist = frontalIR.distance();
    groundIR_dist = groundIR.distance();
  }
  else{ // first reading == false
    firstReading = true;
    leftIR_dist = leftIR.distance();
    rightIR_dist = rightIR.distance();
  }
```

As distâncias mínimas ao objeto diferem para cada sensor. O sensor que aponta para o chão deteta pavimento irregular (buracos, degraus para cima ou para baixo, etc). Caso a distância lida seja superior a 70 cm, significa que a pessoa está em risco de uma queda, logo aplica-se o ângulo de travagem máximo. Se a distância for inferior a 45 cm significa que há um obstáculo à frente do andarilho logo à medida que a pessoa se aproxima do obstáculo o ângulo de travagem aumenta progressivamente. Aplica-se o mesmo processo para o sensor dianteiro e para os laterais, sendo que as distâncias máximas são de 40 e 45 cm, respetivamente.

Por fim, compara-se qual dos sensores requer mais poder de travagem. Os sensores laterais tem a particularidade de travar apenas a roda oposta, criando um efeito que conduz o utilizador para longe do objeto.

```
    // Testa se o sensor IR deteta pavimento irregular
    if(groundIR_dist > DIST_GND_UP) angDistGnd = 0;
    else angDistGnd = - (LOCK_ANG / 15) * (groundIR_dist - DIST_GND_UP); // 15 =
         30-45 (cm)
    if(angDistGnd > LOCK_ANG) angDistGnd = LOCK_ANG;
```

```
5       .
6       .
7       .
8       if(angDistGnd > angDist) angDist = angDistGnd;
9       if(angDist > angDistRight) angDistRight = angDist;
10      if(angDist > angDistLeft) angDistLeft = angDist;
11
12      // Compara qual dos sensores requer mais poder de travagem
13      if(angDistLeft > angDistRight){
14        if(angDistLeft > angDist) angDistMax = angDistLeft;
15        else{
16            angDistLeft = angDist;
17            angDistRight = angDist;
18        }
19      }
20      else{
21        if(angDistRight > angDist) angDistMax = angDistRight;
22        else{
23            angDistLeft = angDist;
24            angDistRight = angDist;
25        }
26      }
```

Após esta leitura, se não for detetado qualquer objeto à frente de um dos sensores, a função principal passa para a próxima rotina.

5.4 Sistema de deteção do sentido de rotação de cada roda

No que diz respeito à determinação do sentido de rotação de cada roda foram implementadas as funções `void leftWheelDirection()` e `void rightWheelDirection()`.
Foi necessário ajustar a distância entre os sensores de Hall dianteiro e traseiro de cada roda de modo a obter quatro estados. Os sensores de Hall têm dois estados possíveis, 1 quando o sensor está sob o efeito de um campo magnético e zero em caso contrário. A distância entre sensores foi ser ajustada de modo a que se consiga obter os quatro estados possíveis: 00, 01, 10 e 11. O primeiro dígito corresponde ao valor lido pelo sensor traseiro e o segundo pelo sensor dianteiro. A cada ciclo faz-se a leitura dos sensores comparando o estado anterior com o estado atual de cada um. Se a leitura for feita na ordem $11 \rightarrow 10 \rightarrow 00 \rightarrow 01 \rightarrow 11$ significa que a roda tem o sentido do movimento para a frente. Se a ordem for $11 \rightarrow 01 \rightarrow 00 \rightarrow 10 \rightarrow 11$ o sentido do movimento é para trás, figura 5.3.

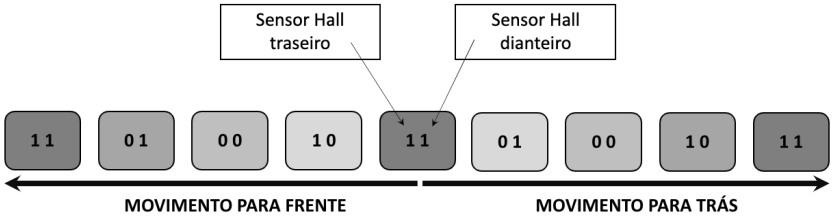


Figura 5.3: Determinação do sentido de rotação das rodas

O sentido de rotação de cada roda é guardado nas variáveis booleanas `rightForwardDirection` e `leftForwardDirection`, que irão ser necessárias em outras funções.

5.5 Cálculo da velocidade

Para calcular a velocidade de cada roda implementaram-se as funções `void interruptLeftVelocity()` e `void interruptRightVelocity()`. Inicialmente, as funções começam por atualizar as variáveis do estado de cada sensor de Hall da roda.
Em seguida, sempre que há a passagem de um íman pelo sensor dianteiro, o número de pulsos é incrementado. É também aqui que se incrementa o número de pulsos necessários para o cálculo da distância percorrida pelo andarilho, variável `odometerPulses`. Estes só são incrementados se a direção do movimento da roda for para a frente.
Para obtermos o valor da velocidade de cada roda é necessário primeiro calcular o número de rotações da roda por intervalo de tempo, equação 5.4. Posteriormente, aplica-se a fórmula da velocidade dada pela equação 5.5.

$$Rotações = \frac{número\ de\ ímanes\ que\ passaram}{número\ de\ ímanes\ totais} \tag{5.4}$$

$$velocidade = \frac{2 \times \pi \times Raio_{roda} * Rotações}{tempo}\ (m/s) \tag{5.5}$$

Como se pretendeu uma atualização mais rápida dos valores da velocidade, o intervalo de tempo foi alterado para 500 ms, variável `VELOCITY_TIME_INTERVAL`. No fim do cálculo da velocidade, a variável que incrementa o número de ímanes é igualada a 0 para reiniciar o contador.

```
1  if(lastStateLeftHallFw == 0 && stateLeftHallFw == 1){ // Um novo iman passou
       pelo sensor do Hall frontal
2      leftPulse++;
3      leftWheelDirection(lastStateLeftHallFw, lastStateLeftHallBw, stateLeftHallFw
           , stateLeftHallBw);
4      if(rightForwardDirection) odometerPulses++;
5    }
6    if(millis() - lastLeftVelocityTime >= VELOCITY_TIME_INTERVAL){  // Atualiza
         contador a cada meio segundo
7      leftRPM = leftPulse / WHEEL_PULSE;
8      leftVelocity = (2 *  PI * WHEEL_RADIUS * leftRPM) * 1000/(
           VELOCITY_TIME_INTERVAL);
9      lastLeftVelocityTime = millis();
10     leftPulse = 0;
11   }
```

5.6 Velocidade máxima do andarilho

A função `void walkerSpeed()` lê o estado do seletor rotativo de seleção de velocidade máxima do andarilho. Em seguida compara o valor atual da velocidade do andarilho com a velocidade máxima selecionada. Se a velocidade atual exceder um determinado valor (`maxvelocity` × `VMAX_THRESHOLD`) é chamada a função `servoSoftLock_velocity()` que irá travar de forma suave à medida que o andarilho se aproxima da velocidade máxima selecionada.

5.7 Distância percorrida pelo andarilho

A função `void walkerOdometer()` calcula a distância percorrida pela pessoa com dependência. Esta distância é incrementada apenas quando a direção de ambas as rodas for para a frente. É obtida através da soma dos ímanes detetados pelo sensor de Hall dianteiro de cada roda multiplicado pelo arco feito entre dois ímanes. A variável `odometerPulses` é incrementada nas funções que calculam a velocidade de cada uma das rodas logo tem-se duas vezes mais pulsos e deve-se dividir por dois para obter o valor real da distância percorrida.

```
1  odometerDist = (odometerPulses * 2 *  PI * WHEEL_RADIUS) / WHEEL_PULSE;
2  // OdometerDist recebe os pulsos de ambas as rodas, portanto deve ser dividido
       por 2
3  odometerDist = odometerDist / 2;
```

5.8 Sistema de deteção de queda da pessoa com dependência e do andarilho

A função `void fallSystem()` é responsável pela deteção de quedas, tanto por parte da pessoa com dependência como pelo andarilho.

Primeiramente, começa-se por atualizar o valor da aceleração atual no eixo Z do MPU-6050, variável `AccZ`. Em seguida, calcula-se a média deslizante dos últimos 16 valores de `AccZ` para evitar valores de falsos positivos.

```
1  // Calcule a m dia deslizante dos  ltimos  16 valores de AccZ
2      float rollingAverage = 0.0;
3      for(int i = 16; i > 0; i--){
4        rollAvg[i] = rollAvg[i-1];
5        rollingAverage += rollAvg[i];
6      }
7      rollAvg[0] = AccZ;
8      rollingAverage = (rollingAverage + rollAvg[0])/16;
```

Esta média é comparada com o valor de limiar de queda, `THRESHOLD_ACCZ` = 0.8. O seu valor é positivo, pois tem a normal do eixo Z a apontar para o chão. Se a média for inferior a este limiar é porque ocorreu uma queda.

```
1  // AccZ do MPU acima do THRESHOLD
2      else if(pow(rollingAverage, 2) < pow(THRESHOLD_ACCZ, 2)){
3         // Serial.print("QUEDA DETETADA = MPU rAvg= ");
4         // Serial.println(rollingAverage);
5        fallDetected = true;
6        angFall = LOCK_ANG;
7        // digitalWrite(buzzerPin, HIGH); // Buzzer ON quando queda detetada
8        lastFallTime = millis();
9      }
```

Por outro lado, a queda também é verificada através do estado do interruptor de emergência. Se o interruptor tiver o seu estado lógico a HIGH é porque ocorreu uma queda e ativa-se de imediato a travagem automática. O ângulo de travagem aplicado às rodas é máximo e os sistemas de avisos luminosos e sonoros são ativados.

Por fim, o estado de queda é interrompido através da força aplicada nos punhos do andarilho. Essa força é lida através das flexiforce e o valor tem de ser superior a FLEXIFORCE_VALUE = 15.

```
if((millis() - lastFallTime >= 3000) && (fallDetected == true)){
      if((analogRead(leftFlexiforcePin) >= FLEXIFORCE_VALUE) && (analogRead(
          rightFlexiforcePin) >= FLEXIFORCE_VALUE)){
        fallDetected = false;
        angFall = 0;
        digitalWrite(buzzerPin, LOW); // Buzzer OFF quando queda detetada
      }
}
```

5.9 Sistema de alertas sonoros e buzina

A função `void walkerHorn()` é utilizada como buzina ou é chamada sempre que é necessário emitir um alerta sonoro. A buzina é ativada sempre que a pessoa pressiona o botão de pressão passando para o estado HIGH. Consequentemente o LED vermelho do botão acende.
Sempre que é detetado um obstáculo ou pavimento irregular o buzzer emite alertas sonoros progressivos que aumentam à medida que o andarilho se aproxima do limite máximo de segurança imposto no sistema de deteção de obstáculos.
Por fim, sempre que ocorrer uma queda o buzzer emite um som intermitente até que a situação seja normalizada.

5.10 Sistema de luzes de presença e alertas luminosos

Para o sistema de luzes de presença desenvolvou-se função `void lightSystem()`. Esta controla a intensidade da luz branca das fitas de LEDs em função da resistência do LDR, reduzindo a intensidade em casos de muita luminosidade e aumentando em casos de escuridão. Em caso de deteção de objetos, pavimento irregular ou ocorrência de uma queda é chamada a função `void warningLights()`.
A função `void warningLights()` é responsável por emitir alertas visuais. Caso seja detetado um obstáculo ou pavimento irregular, é ativada a cor amarela e a sua intermitência aumenta à medida que o andarilho se aproxima do limite máximo de segurança imposto. No caso de uma queda os alertas visuais são de cor vermelha e são emitidos com uma periodicidade determinada pela constante `BLINKING_TIME_INTERVAL` com o valor de 300 ms.

Função void sendToESP()

Esta função é responsável por enviar os vários eventos que ocorrem ao longo da utilização do andarilho. Estes eventos são enviados, através da porta série, para a placa Wi-Fi ESP8266 ESP-01 e este encaminha-os para a AWS Cloud. Os eventos podem ser a deteção de uma queda, deteção de obstáculos ou pavimento irregular, a velocidade instantânea e a distância percorrida.
Esta função também limita o número de vezes que envia cada evento. A queda (evento 1) é enviada

de 5 em 5 segundos, a deteção de pavimento irregular (evento 2) ou deteção de obstáculos (evento 3) de 2,5 em 2,5 segundos. A velocidade e a distância (evento 4) são atualizados de segundo a segundo.

```
if(fallDetected){
    if((millis() - lastSentFall) >= 5000){
      walkerOdometer();           // calcula a dist ncia percorrida
      dtostrf(odometerDist, 4, 2, cStrDist);
      dtostrf((leftVelocity + rightVelocity)/2, 4, 2, cStrVel);
      sprintf(myString, "E1|%s|%s", cStrDist, cStrVel);
      Serial.println(myString);
      lastSentFall = millis();
      lastSentMsg = lastSentFall;
    }
```

É enviada uma *string* composta pelo identificador do evento e dois números *float*, que transmitem a velocidade e a distância.

5.11 Sistema de comunicação com AWS Cloud

Como mencionado anteriormente, a placa Wi-Fi ESP8266 ESP-01 é responsável por enviar os eventos, recebidos pela porta série, para a *Amazon Web Services* (AWS) *cloud*. Para este propósito foi necessário programar a placa e sempre que houve a necessidade de alterar o código fonte seguiram-se os passos descritos no anexo A.2.

Esta placa vem de origem com *software* ESP-AT, que permite que a placa seja usada como uma simples placa Wi-Fi, através da comunicação série UART com o Arduino. Contudo, para a estabelecer a conexão com a AWS *cloud* foi necessário programar a placa diretamente, pois MQTT não corre no Arduino Mega 2560.

Neste caso a placa ESP8266 ESP-01 funciona como um típico microcontrolador (MCU) mas com menos pinos disponíveis. A sua programação também é feita através da interface Arduino IDE.

Em seguida, passa-se à explicação do código fonte para a placa Wi-Fi ESP8266 ESP-01.

Na função *void setup()* o ESP-01 começa por inicializar a comunicação UART com o Arduino. Em seguida inicializa a comunicação Wi-Fi chamando a função `setup_wifi()` e a memória SPIFFS, onde estão guardados os certificados para a conexão com a AWS.

Na função `void loop()`, caso tenha ocorrido alguma falha de conexão com a AWS, começa-se por restabelecer novamente a conexão. Seguidamente, se houver uma nova mensagem UART, esta é recebida e enviada para a *cloud* através da função `client.publish()`. Se a mensagem foi enviada, o LED incorporado na placa pisca.

Por fim, para uma comunicação fiável implementou-se um protocolo de comunicação, onde o Arduino envia mensagens com a seguinte estrutura: `EX|%s|%s`, onde X é substituído por um `event_Id` que determina o tipo de mensagem, sendo que o aviso de queda corresponde ao número 1, pavimento irregular ao número 2, objeto detetado ao número 3 e a atualização da velocidade e da distância ao número 4.

Após uma barra de separação é enviada uma *string* contendo um número em vírgula flutuante que

representa a distância percorrida pelo andarilho. Após outra barra de separação vem a velocidade atual do andarilho.

Se o primeiro *byte* da mensagem não for E, então esta é descartada.

```
if(Serial.available()){
    char Buf = Serial.read();
    if(Buf == 'E'){
      int k = 0;
      while(!Serial.available());
      event_Id = Serial.read();
      while(!Serial.available());
      Serial.read();
      Buf= 'a';
      while(Buf != '|' && k < 10){
        while(!Serial.available());
        Buf = Serial.read();
        SerialBuf[k] = Buf;
        k++;
      }
      SerialBuf[k-1] = '\0';
      Serial.printf(SerialBuf);
      strcpy(distStr, SerialBuf);
      k = 0;
      Buf= 'a';
      while(Buf != '\n' && k < 10){
        while(!Serial.available());
        Buf = Serial.read();
        SerialBuf[k] = Buf;
        k++;
      }
      SerialBuf[k-1] = '\0';
      Serial.printf(SerialBuf);
      strcpy(velStr, SerialBuf);
      snprintf(msg, BUFFER_LEN,"{\"walker_Id\" : \"%d\", \"event_Id\" : \"%c\",
          \"dist\" : \"%s\", \"velocity\" : \"%s\"}", walker_Id, event_Id,
          distStr, velStr);
      Serial.print("Publicar mensagem: ");
      Serial.println(msg);
      client.publish("outTopic", msg);
```

6

Desenvolvimento da aplicação Android

A aplicação *Smart Walker* consiste numa aplicação móvel que permite a localização em tempo real do utilizador do andarilho, tanto por si como pelo seu cuidador.
Esta foi desenvolvida em *Outsystems*[1], uma plataforma portuguesa de desenvolvimento *low-code* de nível empresarial permitindo aos seus utilizadores criarem aplicativos responsivos mais rapidamente. Também permite desenvolver num ambiente intuitivo e visual, muitas vezes através de *Drag & Drop*, isto é, arrastar os módulos desejados e criar uma aplicação do zero, garantindo qualidade e economização do tempo, figura 6.1.

Figura 6.1: Interface gráfica do Service Studio da Outsystems
Fonte adaptada: [62]

A plataforma integra-se facilmente com os sistemas existentes e permite que os utilizadores a personalizem com o seu próprio código. As principais vantagens é a sua capacidade de resposta, integração flexível com qualquer sistema, serviço *web*, base de dados ou aplicação, costumização, escalabilidade, monitoramento e métricas, portabilidade e segurança.
Para terminar, ao desenvolver a aplicação foram usadas as boas práticas de *Outsystems* de forma a tornar o código limpo e claro [63]. Abaixo segue alguns exemplos de boas práticas usadas:

1. Usar o inglês para o código e comentários;
2. Evitar *labels* e descrições vazias;
3. Nomear as variáveis com nomes que tenham significado;
4. Manter os fluxos de ação verticais e organizados;
5. Reutilização da lógica com ações de utilizador e comentar sempre que necessário;
6. Remover referências não utilizadas;
7. Envolver extensões em módulos.

[1] `https://www.outsystems.com/downloads/`

No desenvolvimento do fluxo lógico, usam-se ações do tipo cliente (*client action*) e do tipo servidor (*server action*). As *client actions* são executadas a nível local, isto é, no dispositivo do utilizador, logo não pode manipular ou ter acesso à base de dados. As *server actions* são executadas a nível do servidor, ou seja, a nível de *cloud*, que permite o acesso e manipulação das bases de dados.
Na conceção da interface da aplicação, o foco foi sempre tentar criar uma interface intuitiva e familiar para o utilizador, de forma a garantir que durante a utilização da aplicação, nunca se sentisse perdido sobre qual a ação que deve tomar. Para garantir tais objetivos, foi feita uma análise cuidada dos requisitos funcionais e não funcionais, apresentados na secção 6.1.

6.1 Requisitos funcionais e não funcionais

Esta aplicação é composta por dois atores, sendo eles o cuidador e o utilizador do andarilho. O cuidador é do tipo utilizador registado com hierarquia e o utilizador do andarilho é um utilizador registado dependente do cuidador.
Os requisitos funcionais do *software* encontram-se na figura 6.2.

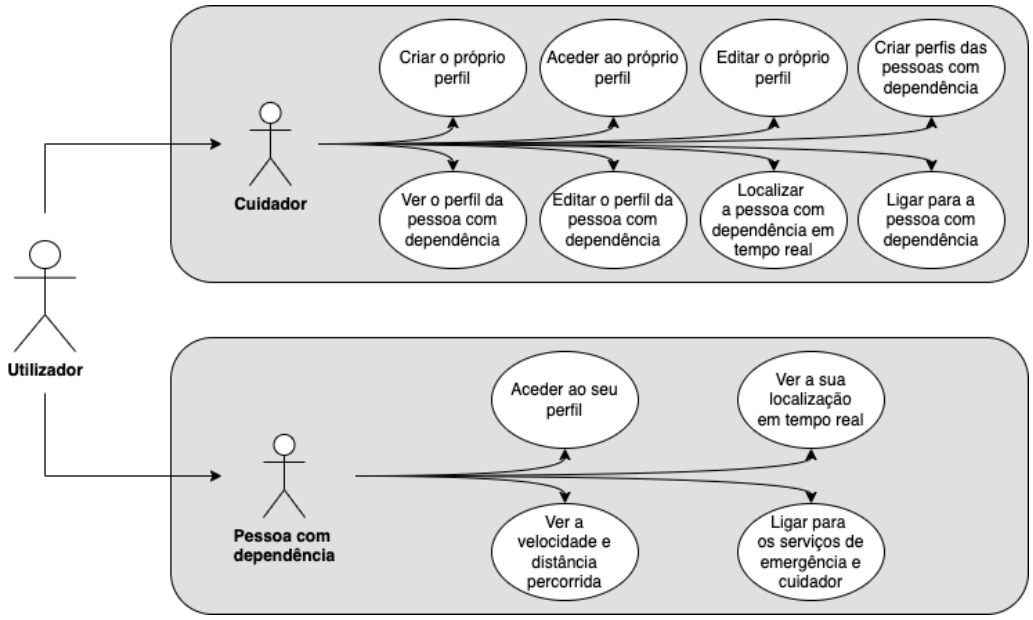


Figura 6.2: Requisitos funcionais da aplicação

Por fim, a aplicação tem dois idiomas, sendo estes português e inglês.

6.2 Modelo da base de dados

Para os requisitos propostos foi necessário criar o modelo de dados da figura 6.3.
Os fios pretos correspondem às relações com a *delete rule protect*, ou seja, não permite apagar registos que estão a ser usados por outras entidades. Os fios vermelhos correspondem a relações com a *delete rule delete*, que ao apagar um dado, os dados associados a esse registo irão também ser apagados.

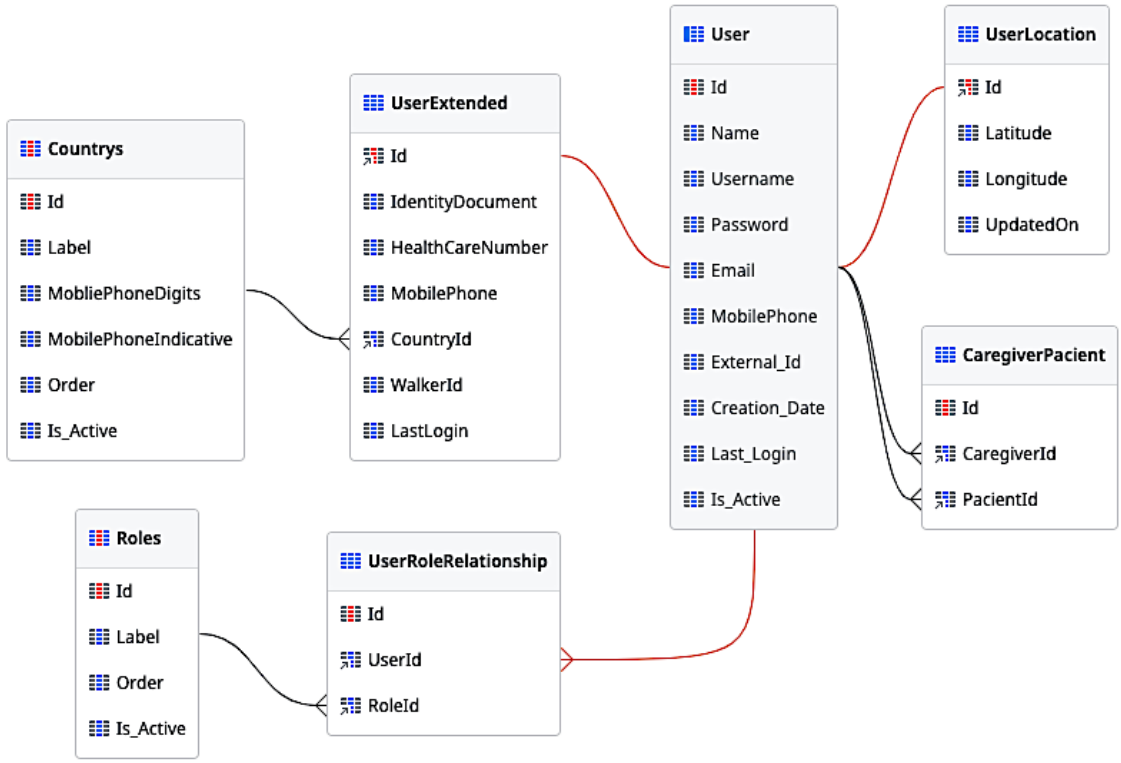


Figura 6.3: Modelo da base de dados

A entidade `User` é criada automaticamente pela Outsystems e armazena os dados referentes aos utilizadores registados na aplicação.
A entidade `UserExtended` foi criada para guardar dados extra associados ao utilizador registado. A chave primária da `UserExtended` é uma chave estrangeira que trata da chave primária da entidade `User`, criando uma relação de 1:1. Além disso, a *delete rule* desta relação está definida como *delete*, ou seja, quando é eliminado um registo da entidade `User`, é automaticamente eliminado o registo associado à entidade `UserExtended`.
A entidade `UserRoleRelationship` guarda os *roles* que foram atribuídos a cada utilizador da aplicação, criando-se assim uma relação N:N entre as entidades `User` e `Roles`.
A entidade `Roles` é uma entidade estática que guarda o tipo de *roles* existentes na aplicação.
A entidade `CaregiverPacient` guarda a relação entre o cuidador e a pessoa a seu encargo, permitindo no futuro que a uma pessoa com dependência seja atribuído a mais que um cuidador,

relação N:N.
A entidade `UserLocation` guarda a latitude, longitude e a hora da última atualização da localização do utilizador do andarilho.
A entidade `Countrys` é uma entidade estática que armazena o nome, o indicativo e o número de dígitos do telemóvel permitidos nesse país. Atualmente tem apenas dois registos, o de Portugal e o de Espanha.

6.3 *Mockups* e *layout* final

Uma vez que os papéis do cuidador e do utilizador do andarilho executam ações diferentes e a informação pertinente é diferente para cada um, decidiu-se criar duas interfaces com conteúdo distinto para cada perfil. Apenas os ecrãs de inicialização, autenticação e edição de informação referente a cada cuidador são iguais para os dois atores.
Assim, de acordo com os requisitos encontrados e de forma a simular-se o *layout* final da aplicação, desenharam-se os *mockups* da mesma. Estes foram desenvolvidos com o auxílio da ferramenta de *design* Framer[2]. No anexo A.6 encontram-se as figuras com as imagens dos *mockups* desenhados e as imagens reais da aplicação.

6.3.1 Inicialização (*splash screen*)

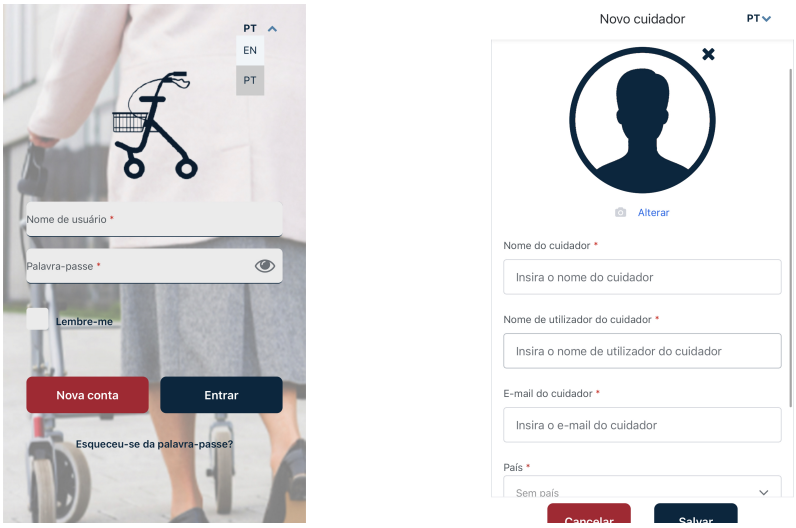
O *splash screen* é apresentado quando se inicia a aplicação. Este ecrã tem o propósito de apresentar a marca enquanto se realizam alguns processamentos iniciais necessários à utilização da aplicação, figura A.13 do anexo A.6.
Não existem gestos ou comportamentos extras associados a este ecrã de inicialização, sendo o utilizador redirecionado automaticamente para o ecrã de autenticação.

6.3.2 Autenticação na aplicação

Por questões de segurança e privacidade criou-se um sistema de autenticação para identificar unicamente cada um dos utilizadores e permitir que estes se conectem à respetiva área pessoal, figura 6.4a.
O *login* permite identificar unicamente cada um dos utilizadores da aplicação e é também o ponto de partida para a utilização da mesma. Tem como requisito um registo na aplicação através do preenchimento de um formulário que tem como dados obrigatórios o nome de utilizador e a palavra-passe. É também neste ecrã que se pode criar uma nova conta de cuidador, figura 6.4b, carregando na opção `Nova conta`. Teve-se o cuidado de verificar que não existiam utilizadores com o mesmo *username*, e-mail e número de telemóvel na base de dados. Por uma questão de segurança,

[2] `https://www.framer.com/fp/`

a palavra-passe associada à conta é guardada na base de dados de forma encriptada. Para isto usou-se uma *server action* que a Outsystem disponibiliza para encriptar texto.
As opções `Lembrar-me` e `Recuperar palavra-passe` não foram implementadas.



(a) Ecrã de *login* (b) Nova conta cuidador

Figura 6.4: Ecrã de *login* na aplicação e nova conta de cuidador

Caso o cuidador pretenda cancelar a criação de um nova conta poderá carregar na opção `Cancelar` ou arrastar o ecrã da esquerda para a direita, onde é automaticamente redirecionado para o ecrã de autenticação.
Para o utilizador entrar na respetiva conta foi necessário verificar se os dados introduzidos para o *login* estão corretos. Caso estejam corretos, o utilizador irá ser registado no *plug-in* One Signal[3], que permite as notificações *push-up* no dispositivo. De seguida, o utilizador é redicionado para o seu ecrã inicial, conforme o seu role (cuidador/pessoa com dependência).

6.3.3 Interface do cuidador

Após o cuidador fazer *login* na sua conta, depara-se com um ecrã que lhe permite adicionar pessoas a seu cuidado ou então vê a lista de pessoas já associadas. No canto inferior direito tem o botão de definições, onde o mesmo pode alterar uma série de dados pessoais, tais como informação pessoal, número de telemóvel, e-mail e palavra-passe, figura A.20 do anexo A.6. Por uma questão de reutilização de lógica, é utilizado o mesmo ecrã para a interface do cuidador e para a interface da pessoa com dependência, sendo que a única diferença entre eles reside na sua designação e informação pessoal a alterar. Caso tenha uma pessoa a seu encargo, ao arrastar botão referente a ele, da esquerda para a direita, pode alterar os dados pessoais do mesmo e, da direita para a esquerda, pode apagar a pessoa dos seus cuidados e da base de dados, figura A.17 do anexo A.6.

[3] `https://onesignal.com/`

Ao carregar no nome de uma pessoa associada, é possível ver alguns dados referentes ao andarilho, à pessoa, à sua localização atual e permite realizar uma chamada para a mesma, figura A.18 do anexo A.6.

Por fim, caso tenha ocorrido uma queda por parte da pessoa a seu cuidado, este recebe na sua aplicação e no dispositivo móvel uma notificação com o local e as horas da queda, figura 6.5.

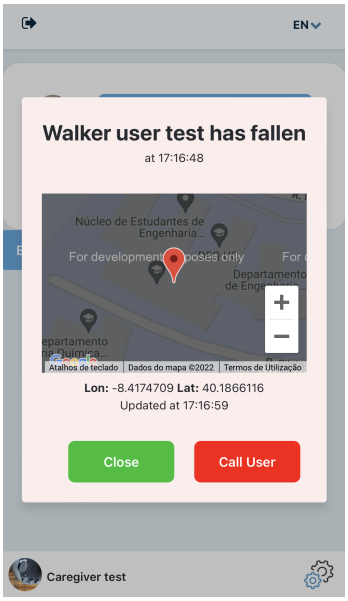


Figura 6.5: Notificação com o local e as horas da queda na interface do cuidador

6.3.4 Interface do utilizador do andarilho

Após este inserir os seus dados de *login*, é redirecionado para o seu ecrã inicial. Os seus dados de *login* são criados pelo seu cuidador. Não lhe é permitido a alteração de quaisquer dados associados ao seu perfil. Estes podem ser alterados apenas pelo seu cuidador.

No ecrã inicial, para além da distância percorrida e da velocidade atual, também tem o GPS que lhe indica a sua localização em tempo real, figura 6.6.

No fim do ecrã, tem dois botões que lhe permitem fazer uma chamada para os serviços de emergência ou para o seu cuidador. Também tem acesso a alguns dados associados a ele e do seu responsável.

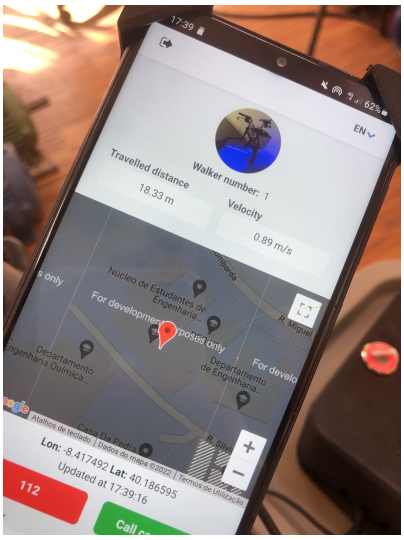


Figura 6.6: Localização do utilizador do andarilho em tempo real através da sua interface

Durante a utilização da aplicação, o utilizador do andarilho recebe diversas notificações, tais como ”objeto detetado”, ”degrau detetado” e ”queda detetada”. Quando ocorre uma queda, o ecrã fica bloqueado até a pessoa carregar no botão `I'm OK/ Estou OK`. Estas notificações podem ser observadas na figura 6.7.

Figura 6.7: Notificações da interface do utilizador do andarilho

6.4 Obtenção da localização e notificação de queda

A localização é obtida através da integração de um *plug-in* de localização, `LocationPlugin`. Esta é atualizada de segundo a segundo através do um temporizador que executa por uma *client action*. Nesta *client action* começa-se por parar o temporizador e verifica-se se o *plug-in* está disponível no dispositivo atual. Caso esteja, obtém-se a localização e atualiza-se o seu valor na base de dados. Caso o paciente tenha caído, é apresentada uma notificação *pop-up* no seu ecrã e

outra no ecrã do seu cuidador. Por fim, é atualizado o valor da velocidade e distância do utilizador do andarilho e inicializa-se o temporizador para a próxima execução.
Assim, é possível que o cuidador tenha acesso à localização da pessoa a seu cuidado através da informação guardada na base de dados. Para isto, no ecrã do cuidador tem-se um temporizador que de 2 em 2 segundos atualiza a informação guardada na base de dados.

6.5 Obtenção das notificações da AWS Cloud

De forma a obter as notificações do andarilho foi criada uma *server action* chamada `DB_Data_Treatment`, figura 6.8.

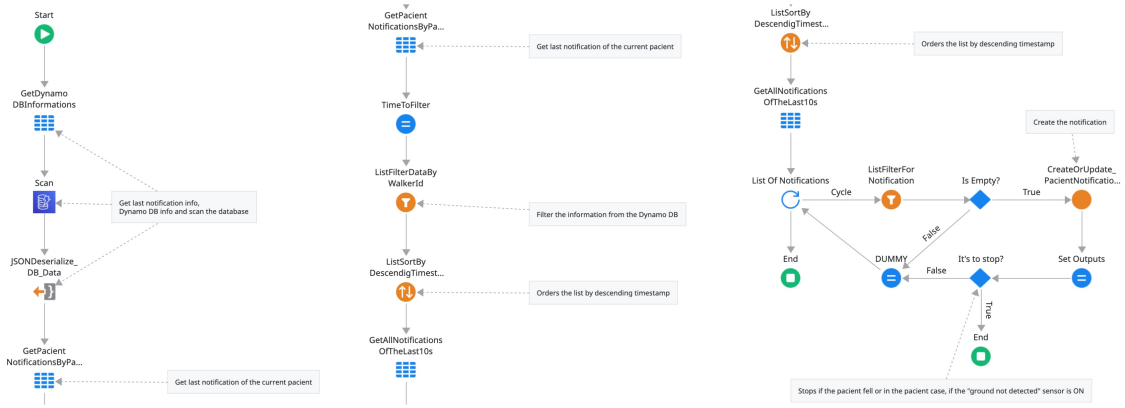


Figura 6.8: Obtenção das notificações da tabela Dynamo DB

Esta ação é utilizada para obter as notificações tanto no caso de o utilizador se tratar de um cuidador, como no caso de se tratar de um paciente. A única diferença entre estes dois casos é que no caso de se tratar de cuidador este é notificado caso algum dos seus pacientes tenha caído, figura 6.9.

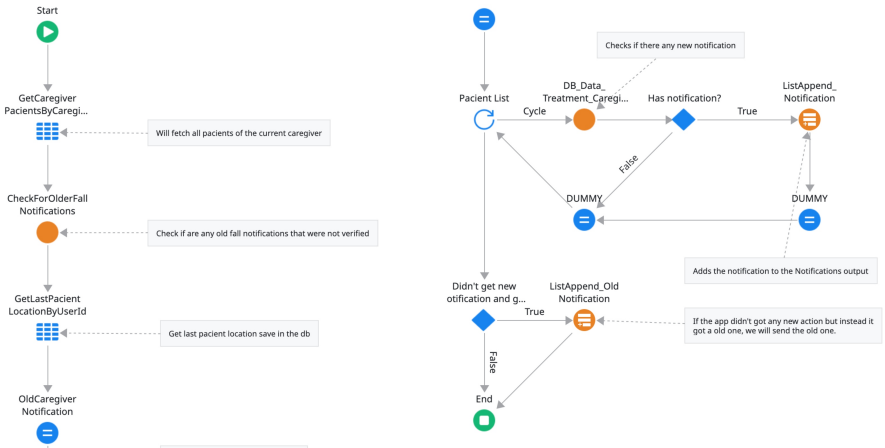


Figura 6.9: Notificações na interface do cuidador

No caso do paciente, notifica se encontrou um obstáculo, uma escada ou se houve uma queda, figura 6.10.

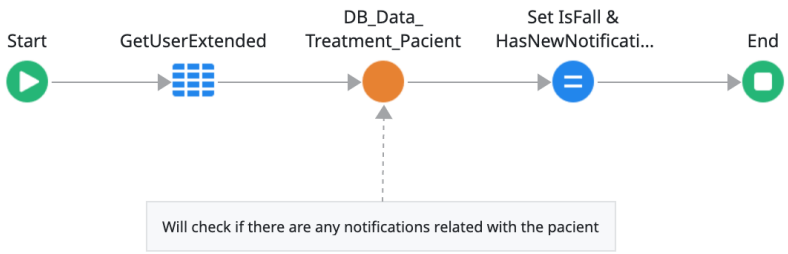


Figura 6.10: Notificações na interface do utilizador do andarilho

Primeiro começa-se pela busca de todas as informações de acesso à *Amazon Web Services* (AWS) e de seguida faz-se um *scan* da tabela completa. Após desserializar a informação retornada, procura-se a última notificação do paciente de forma a definir o intervalo de tempo de procura nas notificações retornadas.

Caso a última notificação tenha ocorrido há mais de 5 minutos, coloca-se na variável `TimeToFilter` a hora relativa à hora e minutos atuais menos 5 minutos, caso contrário utiliza-se a hora da última notificação como referência.

De forma a obter apenas notificações que sejam posterior ao tempo de referência filtra-se a lista de notificações retornada pelo *scan*, filtrado pelo `walkerId` do paciente atual e pelo tempo de referência definido anteriormente.

Adicionalmente ainda se organizam os registos filtrados pelo *timestamp* de forma descendente, pois os registos podem vir desorganizados e neste processo pretende-se dar prioridade às notificações mais recentes.

Finalmente procuram-se as notificações relativas a este utilizador dos últimos 10 segundos de forma a comparar com as notificações filtradas. Caso as notificações já existam, estas são ignoradas pois já foram registadas na base de dados. Se ainda não existirem na base de dados, estas são guardadas, e se se tratar de uma queda para-se de imediato o ciclo para avisar o dispositivo do utilizador.

No caso do paciente não ter recebido nenhuma notificação de queda, no entanto, recebeu pelo menos uma notificação de chão não detetado ou uma notificação de obstáculo, uma das saídas, `HasNewNotification` irá sair a verdadeiro e o `IsGroundNotDetectedIsGroundNotDetected` irá sair a verdadeiro, caso tenha recebido pelo menos uma notificação de chão não detetado.

Para evitar a lentidão na obtenção dos dados em tempo real criou-se um temporizador que corre de meia em meia hora de forma a limpar a base de dados, figura 6.11.

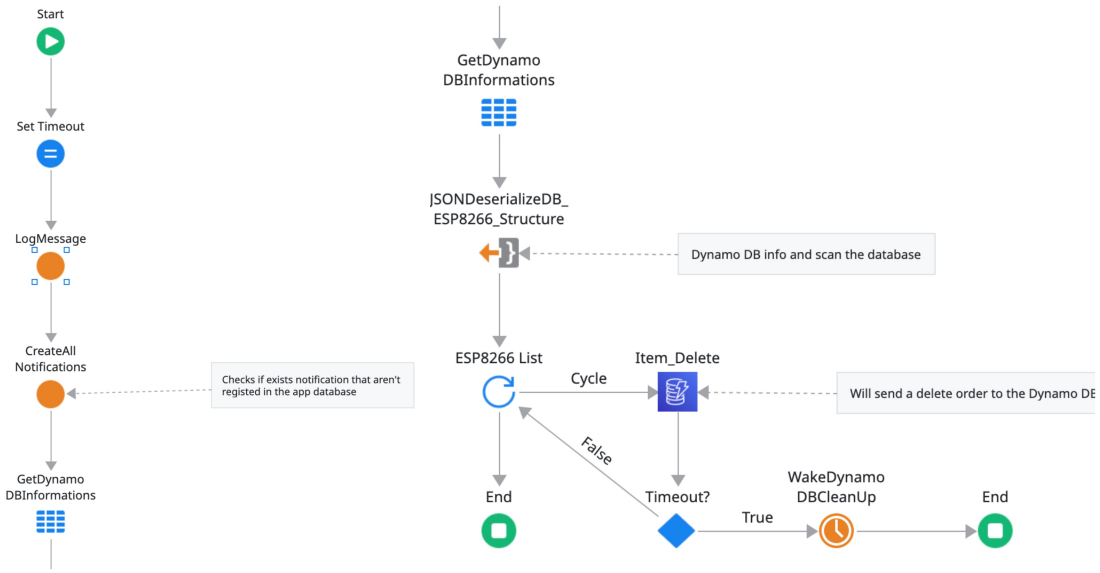


Figura 6.11: Processo de limpeza da base de dados

Primeiro trata-se de verificar se existe alguma notificação que ainda não tenha sido criada, para não perder nenhuma informação, e em seguida eliminam-se todas as notificações existentes.

6.6 Instalação da aplicação

A aplicação pode ser instalada lendo os *QR code* da figura 6.12a.

(a) *QR code* Android

(b) *QR code* iOS

Figura 6.12: *QR codes* da aplicação *Smart Walker*

Se for um utilizador iOS, não poderá instalar a aplicação no dispositivo mas pode explorá-la por *Progressive Web App* (PWA) através do *QR code* da figura 6.12b.

7

Testes e análise de resultados

7.1 Testes e resultados

A fim de avaliar a eficácia da travagem do andarilho fizeram-se vários testes aos servo motores. Assim, para obter o ângulo de travagem máximo, o mesmo foi ajustado até que as rodas ficassem completamente bloqueadas. Verificou-se que a roda esquerda necessitava de um ângulo maior, em comparação ao da roda direita.
Na tabela 7.1 especificam-se os ângulos de destravagem e travagem máxima de cada roda.

Tabela 7.1: Ângulos de destravagem e travagem máxima de cada roda

	Ângulo de destravagem	**Ângulo de travagem máxima**
Roda direita	105º	80º
Roda esquerda	80º	120º

Após a escolha do ângulo de travagem máximo das rodas fez-se testes para determinar o declive da rampa para um controlo da travagem proporcional à velocidade. Começou-se por um limiar de velocidade do andarilho de 80% da velocidade máxima imposta. Todavia este valor foi alterado para 50% para permitir uma melhor visualização do efeito da travagem suave.
Como referido na secção 3.4 do capítulo 3, o protótipo inicial trazia 12 ímanes de 20 mm à volta de cada roda. Após a aplicação dos testes de verificação através das funções `void interruptLeftVelocity()` e `void interruptRightVelocity()` verificou-se que a resolução dos valores da distância eram baixos. De forma a melhorá-la, trocaram-se os ímanes existentes por 26 ímanes com dimensão 8 mm $\times$ 1 mm, obtendo-se uma resolução maior que o dobro. Esta alteração exigiu a utilização de um paquímetro, de forma a conseguir-se obter a mesma distância entre cada íman. Houve alguma dificuldade na ajuste destas distâncias, pois uma variação mínima na casa das milésimas propagava o erro, implicando uma má leitura dos quatro estados que determinam o sentido de rotação da roda.

Durante os testes da deteção de queda do andarilho observou-se que o sensor apresentava uma alta sensibilidade, o que implicava uma deteção de queda em imperfeições pavimentares insignificativas. Para colmatar os valores de falsos positivos no eixo Z da aceleração, aplicou-se um filtro de média deslizante. O sistema foi melhorado, contudo carece melhorias em trabalhos futuros.

Relativamente à verificação da distância percorrida pelo andarilho, foi assinalado no pavimento um segmento de 3 metros. Ao comparar o valor percorrido com o valor apresentado na aplicação móvel desenvolvida, o resultado era sobreponível quando se percorriam velocidades inferiores a 2 km/h.

Como mencionado na secção 5.3 do capítulo 5, para obter as distâncias entre o sensor e o objeto, utilizou-se a biblioteca `Sharp IR`. Esta biblioteca faz a leitura de 25 valores, descartando valores discrepantes e calcula a média para obter uma leitura de distância mais estável. Contudo,

a leitura de 25 amostras interfere com a performance do código, uma vez que ocorrem menos ciclos de *loop* por segundo. Esta decaimento da performance leva a leituras erradas do número de ímanes que passam pelo sensor de Hall, que consequentemente resulta em valores de velocidade e distância percorrida errados.
Verificou-se que os ímanes só eram bem lidos quando o número de ciclos/segundo era superior a 251, figura 7.1. Abaixo disso, os valores da distância e da velocidade estavam errados.

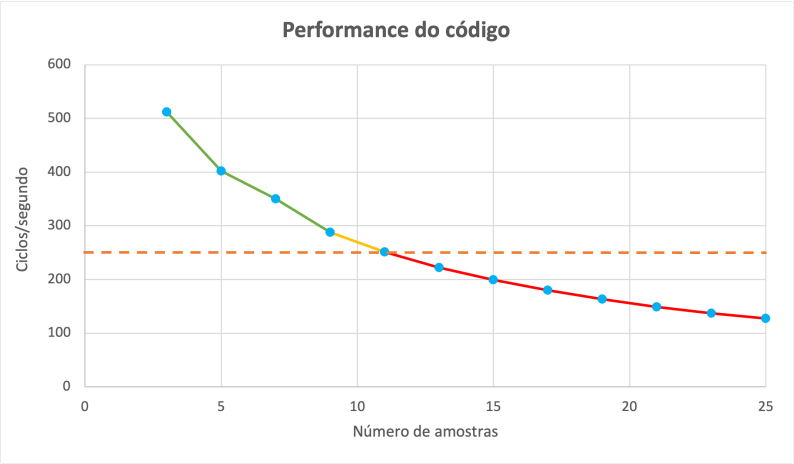
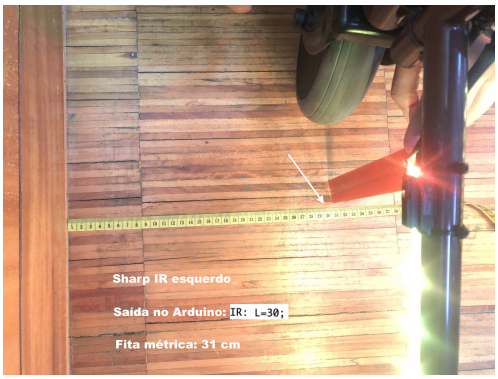


Figura 7.1: Número de ciclos por segundo sem leitura alternada dos sensores de distância

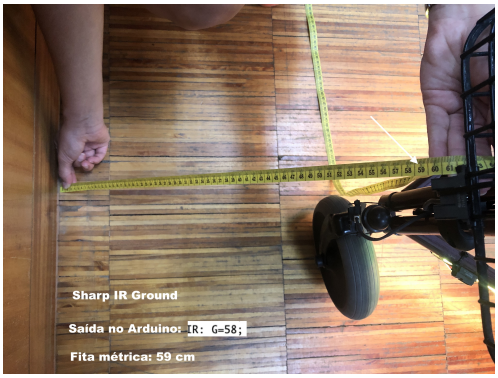
Para aumentar o número de ciclos/segundo, optou-se por ler as distâncias dos sensores Sharp IR de forma alternada. Primeiro leram-se dois sensores, e posteriormente os outros dois. Com esta mudança conseguiu-se obter 512 ciclos de *loops* por segundo.
Com a alteração de 25 para 3 amostras verificou-se que os valores das distâncias medidas variam cerca de 1 a 3 cm em comparação com as distâncias reais, figuras 7.2 e 7.3.
De modo a verificar os valores da distância dada pelos sensores Sharp IR em relação a um dado obstáculo, colocou-se o andarilho a apontar para paredes e degraus. Na situação das paredes, com o auxílio de uma fita métrica, mediu-se a distância entre a parede e o sensor que aponta para ela. Em seguida, comparou-se o resultado lido na fita métrica com o resultado dado pelo sensor. Repetiu-se o mesmo processo para todos os sensores.



(a) Sharp IR esquerdo

(b) Sharp IR direito

Figura 7.2: Distâncias obtidas pelos sensores de distância laterais



(a) Sharp IR Ground

(b) Sharp IR frontal

Figura 7.3: Distâncias obtidas pelos sensores de distância Ground e frontal

Para o controlo inversamente proporcional à distância, ajustaram-se as distâncias máximas permitidas. Estas distâncias foram escolhidas de tal modo a que fosse permitida a manobra de desvio do obstáculo e a passagem entre dois obstáculos, figura 7.4.

Figura 7.4: Sentido do movimento do andarilho

Em todas as tentativas de testagem dos sistemas de deteção de queda e obstáculos ou degraus, os sistemas luminoso e sonoro foram ativados.

Por fim, comparou-se o tempo de processamento de cada função desenvolvida para este protótipo.

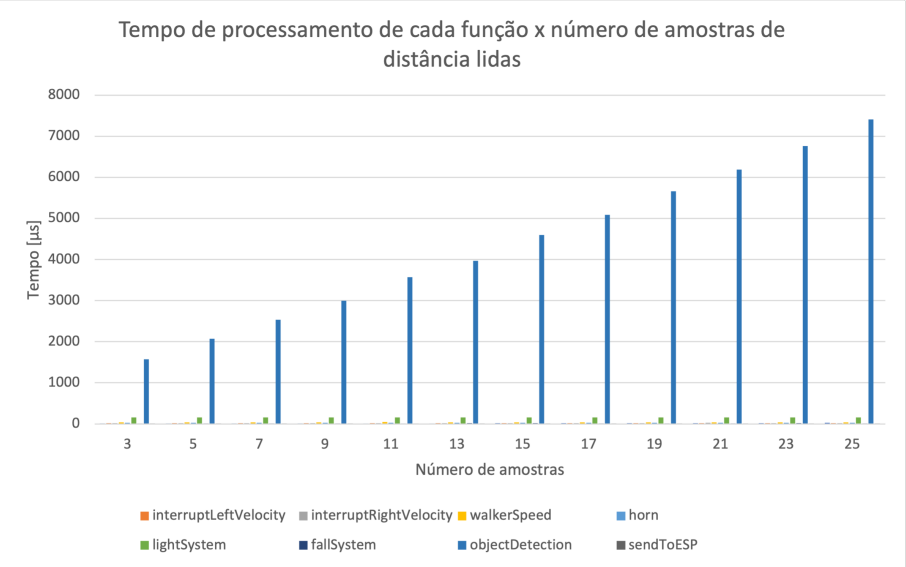


Figura 7.5: Tempo de processamento de cada função, em milissegundos, em função do número de amostras de distância lidas

Verificou-se que o tempo de computação da função `objectDetection()` é diretamente proporcional ao aumento das amostras de leitura da distância.
Também é notório que em comparação às restantes funções, a identificação de obstáculos é o que consome mais tempo de processamento.
Para obter um ponto de equilíbrio entre os valores da velocidade, distância percorrida e direção do sentido da roda, optou-se por diminuir o número de amostras de valores de distância ao obstáculo lidos, assumindo um maior distanciamento, de 1 a 3 cm, do obstáculo.

8

Considerações finais e perspetivas futuras

8.1 Considerações finais

A presente dissertação propõe a instrumentação e inovação de um andarilho de três rodas do tipo Delta para auxiliar as pessoas com compromisso leve a moderado da mobilidade ou alterações transitórias cognitivas e sensoriais. Para tal, utilizou-se o microcontrolador Arduino Mega 2560 e vários sensores de entradas analógicas e digitais, que permitiram a criação de alguns sistemas inteligentes, tais como sistema de travagem automático proporcional à velocidade máxima imposta ao andarilho e inversamente proporcional à distância de um obstáculo, sistema de deteção de movimento e sentido da direção de cada uma das rodas, deteção de obstáculos e pavimentos irregulares, deteção de queda da pessoa ou do andarilho, sistema de avisos luminosos e de presença, sistema de alertas sonoros e buzina e por fim a localização em tempo real. Ainda é calculado a velocidade momentânea e distância percorrida pelo andarilho.

Com o desenvolvimento tecnológico, cada vez mais os andarilhos instrumentados proporcionam às pessoas sistemas inteligentes que permitem o seu uso em diversos contextos facilitando cada vez mais a exploração de novos ambientes e o aumento da segurança dos mesmos.

Neste protótipo, de modo a aumentar a segurança conetaram-se sensores ao Arduíno Mega 2560 por intermédio de uma placa de circuito impresso. Adicionalmente, desenvolveu-se uma aplicação móvel para o sistema operativo Android com a finalidade de melhorar a vigilância e a comunicação entre o cuidador e a pessoa com necessidades especiais. Com o intuito de monitorizar o local onde se encontra a pessoa foi integrado também o sistema de localização por GPS.

Através do desenvolvimento e implementação de novos auxiliadores de marcha inteligentes a área de engenharia assume um papel preponderante na saúde, proporcionando às pessoas com alteração das funcionalidades motoras, cognitivas ou sensoriais uma melhor recuperação ou manutenção do estado de saúde, assim como mais qualidade de vida.

Com o desenvolvimento deste protótipo, indubitavelmente deu-se mais um passo na área da engenharia.

A introdução de novas funcionalidades aos equipamentos de apoio técnico é a chave para mitigar muitas lacunas existentes na área da saúde reabilitacional.

Por fim, a presente dissertação propiciou o alargamento do conhecimento relacionado com a engenharia eletrotécnica e desenvolvimento de aplicações móveis. Neste momento, o protótipo desenvolvido atende à expetativa proposta, contudo para a concretização do mesmo existiram vários constrangimentos e limitações. Uma dela é o pré-requisito inicial de reutilização de algum material pré-existente no protótipo anterior desenvolvido, que careceu um estudo exaustivo dos materiais disponíveis no mercado com compatibilidade dos materiais pré-existentes. Com o panorama atual da crise mundial e encerramento de determinados mercados existiram muitos constrangimentos na compra e entrega do material.

Para além disso, praticamente não existem ensaios clínicos aplicados em diversos grupos de pes-

soas com necessidades de andarilhos inteligentes, para averiguar a eficácia dos mesmos. Assim, pode-se afirmar que esta área de engenharia está em ascensão e com o presente trabalho sugere-se perspetivas futuras de atuação neste ramo.

8.2 Perspetivas futuras

A concretização dos objetivos traçados e a obtenção dos resultados positivos dos testes realizados deixa em aberto um conjunto de aspetos interessantes que sobressaem da experiência do trabalho realizado e que permitem conjeturar uma linha de evolução nesta área.
Passam-se a identificar alguns aspetos que são suscetíveis de serem desenvolvidos no futuro:

- Substituir o Arduino Mega 2560 por outra placa, como por exemplo a Hitex ShieldBuddy TC275[1]. Esta tem um processador de três cores @ 200 Mhz, uma frequência de PWM 300 vezes mais rápida e mantém o mesmo *pinout* do Arduino Mega 2560;
- Melhorar o sistema de travagem, para evitar as situações em que o andarilho derrapa e para o tornar esteticamente mais apelativo;
- Conceber um sistema de deteção do atraso na marcha e correção postural da mesma;
- Expandir a aplicação móvel para ser utilizada num ambiente hospitalar, como por exemplo através da inserção de um menu de medicação e avisos da toma da mesma, envio de notificações e análises clínicas por e-mail, etc;

Por fim, de modo a disseminar o trabalho desenvolvido e contribuir com a literatura nesta área ambiciona-se realizar um artigo e publicar na revista *Sensors* ou *Frontiers in Robotics and AI*.
Com o intuito de aproximar a academia à prática e à testagem das propriedades do protótipo, intenciona-se realizar um estudo multidisciplinar com o objetivo de aplicar o andarilho concebido em pessoas em fase de reabilitação, internados no Centro Hospitalar Universitário Algarve e em pessoas que se encontram integradas na Unidade Hospitalização Domiciliária da mesma instituição. Assim, para permitir aos engenheiros escolhas de produtos custo eficientes, novos estudos de análise do mercado dos produtos torna-se imperativo; para os profissionais de saúde ligados a área de reabilitação sugere-se a realização de novos ensaios clínicos para verificar a eficácia de novas funcionalidades inteligentes atribuídas aos andarilhos instrumentados. Certamente, que apenas com o *feedback* dos profissionais de saúde, dos cuidadores formais e pessoas afetadas a área de engenharia alarga os horizontes na área de desenho e conceção de novos equipamentos inteligentes.

[1] `https://www.hitex.com/microcontroller-support/aurix/shieldbuddy-tc275`

Bibliografia

[1] Instituto Nacional de Estatística, "Estatísticas Demográficas 2019", Lisboa, Tech. Rep. 25, 2020.

[2] HelpAge International and UNFPA, "Envelhecimento no Século XXI: Celebração e Desafio", *Fundo de População das Nações Unidas (UNFPA)*, p. 12, 2012.

[3] O. M. da Saúde, "Relatório mundial de envelhecimento e saúde", Organização Mundial da Saúde, Genebra, Suiça, Tech. Rep., 2015.

[4] A. G. A. Leitão and A. M. S. S. C. C. E. N. E. I. F. C. G. v. A. I. F. J. G. C. M. J. H. W. Oswald, "Programa Nacional de Saúde das Pessoas Idosas", *Direcção-Geral Da Saúde - Divisão De Doenças Genéticas, Crónicas E Geriátricas*, vol. 1, no. 1, pp. 1–28, 2006. [Online]. Acessível em: http://www.fafit.com.br/revista/index.php/fafit/article/viewFile/16/12

[5] L. S. Lippert, *Cinesiologia Clínica e Anatomia*, 5th ed., G. Koogan, Ed., 2013.

[6] D. Brandão, Ó. Ribeiro, and C. Paúl, "Functional, sensorial, mobility and communication difficulties in the Portuguese oldest old (80+)", *Acta Medica Portuguesa*, vol. 30, no. 6, pp. 463–471, 2017.

[7] D. da República, "Despacho n.º 7197/2016 de 1 de Junho, do Instituto Nacional para a Reabilitação, I.P." *Diário da República, 2.ª série*, pp. 17 168–17 185, 2016. [Online]. Acessível em: https://dre.pt/home/-/dre/74587625/details/maximized

[8] O. dos Enfermeiros, "Cuidados à pessoa com alterações da mobilidade - posicionamentos, transferências e treino de deambulação", Ordem dos Enfermeiros, Lisboa, Tech. Rep., 2013.

[9] M. M. Martins, "ASBGo: A smart walker for mobility assistance and monitoring system aid", PhD thesis in Biomedical Engineering, Universidade do Minho, 2016. [Online]. Acessível em: http://repositorium.sdum.uminho.pt/handle/1822/42531

[10] M. M. Martins, C. P. Santos, A. Frizera-Neto, and R. Ceres, "Assistive mobility devices focusing on Smart Walkers: Classification and review", *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 548–562, 2012. [Online]. Acessível em: http://dx.doi.org/10.1016/j.robot.2011.11.015

[11] P. F. Cavaleiro, "Andarilho de reabilitação assistido eletronicamente", Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores, Universidade de Coimbra, 2017. [Online]. Acessível em: http://hdl.handle.net/10316/83112

[12] C. R. de Moura Silva, J. Bezerra, F. C. Soares, J. Mota, M. V. G. de Barros, and R. M. Tassitano, "Perception of barriers and facilitators for users to participate in physical activity programs", *Cadernos de Saude Publica*, vol. 36, no. 4, pp. 1–12, 2020.

[13] R. Silva, F. Couto, M. L. Almeida, C. Malça, P. Parreira, J. Apóstolo, and A. Cruz, "Use of three types of walkers: Impact on the functional performance profile of institutionalized older adults", *Revista de Enfermagem Referencia*, vol. 2019, no. 22, pp. 139–148, 2019. [Online]. Acessível em: https://web.esenfc.pt/v02/pa/conteudos/downloadArtigo.php?id{_}ficheiro=2394{&}codigo=

[14] J. F. B. Barreira, "Deteção das pernas e controlo do andarilho recorrendo a um sensor laser range finder", Dissertação de Mestrado Integrado em Engenharia Biomédica (Ramo Eletrónica Médica), Universidade do Minho, 2014. [Online]. Acessível em: http://repositorium.sdum.uminho.pt/handle/1822/34042?mode=full

[15] The Mobility Aids Centre Ltd Started., "Understanding Different Types Of Walking Aids". [Online]. Acessível em: https://www.themobilityaidscentre.co.uk/understanding-different-types-of-walking-aids/

[16] S. B. Thies, R. Russell, A. Al-Ani, T. Belet, A. Bates, E. Costamagna, L. Kenney, and D. Howard, "An investigation of the effects of walking frame height and width on walking stability", *Gait and Posture*, vol. 82, no. December 2019, pp. 248–253, 2020. [Online]. Acessível em: https://doi.org/10.1016/j.gaitpost.2020.09.017

[17] R. Silva, "Estudo Comparativo De Três Tipos De Andarilho Em Idosos Institucionalizados: Desafios Para", 2019.

[18] A. M. F. Tereso, "Estudo e Avaliação da adaptação andarilho-doença do utilizador Ana Margarida Faria Tereso Estudo e Avaliação da adaptação andarilho-doença do utilizador", Dissertação de mestrado integrado em Engenharia Biomédica (área de especialização em Eletrónica Médica), Universidade do Minho, 2014. [Online]. Acessível em: http://hdl.handle.net/1822/34056

[19] "ANDARILHO FIXO", https://www.medicalexpress.net/andarilho-fixo-41757-.html, acedido em: 2022-02-19.

[20] "Andarilho Articulado de Encartar", https://www.google.com/url?sa=i{&}url=https{%}3A{%}2F{%}2Fcentrodeortopediaereabilitacao.com{%}2Fproduto{%}2Fandarilho-

articulado-de-encartar{%}2F{&}psig=AOvVaw35vviZCq3d4Qfsc8P7vpe4{&}ust=1655504223821000{&}source=images{&}cd=vfe{&}ved=0CA0QjhxqFwoTCLC88{_}v{_}svgCFQAAAAAdAAAAABAD, acedido em: 2022-02-19.

[21] "Andarilho com 2 rodas Biort". [Online]. Acessível em: https://portopedico.com/produto/andarilho-com-2-rodas-biort/

[22] "Andarilho Com 2 Rodas E Assento", https://www.gameiros.pt/rodado/63581-andarilho-com-2-rodas-e-assento-5601000002612.html, acedido em: 2022-02-19.

[23] "Andarilho com três rodas, cesta e travão", http://incomedicura.pt/andarilho{_}com{_}tr{\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\let\typeout\protect\begingroup\def\MessageBreak{■(Font)}\let\protect\immediate\write\m@ne{LaTeXFontInfo:\def{}oninputline176.}\endgroup\endgroup\relax\let\ignorespaces\relax\accent94e\egroup\spacefactor\accent@spacefactor}s{_}rodas{_}cesta{_}e{_}trav{~{a}}o, acedido em: 2022-02-20.

[24] IncoMediCura, "Andarilho com 4 rodas, cesta e assento para descanso", http://incomedicura.pt/andarilho{_}com{_}4rodas{_}cesta{_}e{_}assento{_}para{_}descanso, acedido em: 2022-02-20.

[25] R. Wellmon, K. Pezzillo, G. Eichhorn, W. Lockhart, and J. Morris, "Changes in dual-task voice reaction time among elders who use assistive devices", *Journal of Geriatric Physical Therapy*, vol. 29, no. 2, pp. 74–80, 2006. [Online]. Acessível em: https://pubmed.ncbi.nlm.nih.gov/16914064/

[26] J. Alves, I. Caetano, E. Seabra, and C. Santos, "Design considerations of ASBGo++ (Plus Plus) Smart Walker", *Robótica : Automação, Controlo, Instrumentação*, vol. 3º. trimes, no. 108, pp. 4–8, 2017.

[27] S. D. Sierra M, M. Múnera, T. Provot, M. Bourgain, and C. A. Cifuentes, "Evaluation of physical interaction during walker-assisted gait with the agora walker: Strategies based on virtual mechanical stiffness", *Sensors*, vol. 21, no. 9, 2021.

[28] A. Frizera, R. Raya, J. Pons, A. Abellanas, and R. Ceres, "The smart walkers as geriatric assistive device", *6th International Conference of the International Society for Gerontechnology*, 2008. [Online]. Acessível em: https://journal.gerontechnology.org/archives/2008papers/049.pdf

[29] Y. Hirata, A. Muraki, and K. Kosuge, "Motion control of intelligent walker based on renew of estimation parameters for user state", *IEEE International Conference*

on Intelligent Robots and Systems, pp. 1050–1055, 2006. [Online]. Acessível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=4058503

[30] Y. Hirata, A. Hara, and K. Kosuge, "Motion control of passive intelligent walker using servo brakes", *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 981–990, 2007.

[31] S. Taghvaei, Y. Hirata, and K. Kosuge, "Vision-based human state estimation to control an intelligent passive walker", *2010 IEEE/SICE International Symposium on System Integration: SI International 2010 - The 3rd Symposium on System Integration, SII 2010, Proceedings*, pp. 146–151, 2010. [Online]. Acessível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=5708316

[32] J. V. Miró, V. Osswald, M. Patel, and G. Dissanayake, "Robotic assistance with attitude: A mobility agent for motor function rehabilitation and ambulation support", *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009*, no. August 2014, pp. 529–534, 2009. [Online]. Acessível em: https://www.researchgate.net/publication/224580429{_}Robotic{_}assistance{_}with{_}attitude{_}A{_}mobility{_}agent{_}for{_}motor{_}function{_}rehabilitation{_}and{_}ambulation{_}support

[33] J. Shin, A. Rusakov, and B. Meyer, "SmartWalker: An intelligent robotic walker", *Journal of Ambient Intelligence and Smart Environments*, vol. 8, no. 4, pp. 383–398, 2016. [Online]. Acessível em: http://se.ethz.ch/{~}meyer/publications/concurrency/smartwalker.pdf

[34] V. Kulyukin, A. Kutiyanawala, E. LoPresti, J. Matthews, and R. Simpson, "IWalker: Toward a rollator-mounted wayfinding system for the elderly", *2008 IEEE International Conference on RFID (Frequency Identification), IEEE RFID 2008*, pp. 303–311, 2008. [Online]. Acessível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=4519363

[35] G. Postolache, P. S. Girão, and O. Postolache, "Applying smartphone apps to drive greater patient engagement in personalized physiotherapy", *IEEE MeMeA 2014 - IEEE International Symposium on Medical Measurements and Applications, Proceedings*, 2014. [Online]. Acessível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=6860094

[36] C. Santos, "Investigadores da EEUM criam andarilho inteligente", https://www.engium.uminho.pt/investigadores-da-eeum-criam-andarilho-inteligente/, acedido em: 2022-02-18.

[37] H. Hashimoto, A. Sasaki, Y. Ohyama, and C. Ishii, "Walker with hand haptic interface for spatial recognition", *International Workshop on Advanced Motion Control, AMC*, vol. 2006, pp. 311–316, 2006.

[38] S. Taghvaei, Y. Hirata, and K. Kosuge, "Control of a passive walker using a depth sensor for user state estimation", *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011*, pp. 1639–1645, 2011. [Online]. Acessível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6181524

[39] "Andarilho adultos — 3 rodas — Delta". [Online]. Acessível em: https://www.queralto.com/pt/21134-andarilho-adultos-3-rodas-delta.html

[40] T. Areas, "Arduino ® MEGA 2560 Rev3 Features", pp. 1–18, 2022. [Online]. Acessível em: https://docs.arduino.cc/static/904167872b883b39a905edd6ed31e3e3/A000067-datasheet.pdf

[41] M. Barr, "BEGINNER'S CORNER Lint", *Embedded Systems programming*, no. May, pp. 55–56, 2002.

[42] S. Shah and U. Shah, *Arduino BLINK Blueprints*.

[43] EG PRojects, "Controlling Servo Motor with Stm32f103 microcontroller using stm32cubemx code configurator by STMicroelectronics and keil uvision 5 ide for cortex m1 series microcontrollers", https://www.engineersgarage.com/interfacing-servo-motor-with-stm32/, acedido em: 2022-03-15.

[44] "Continuous-Time Switch Family A110X", *Allegro*, pp. 1–11, 2012. [Online]. Acessível em: https://www.allegromicro.com/-/media/files/datasheets/a110x-datasheet.ashx

[45] AliExpress, "16/19/22mm 2 3 position Metal Selector Rotary Switch Latching Push Button Switch with 12V 220V LED Illuminated Switch 1NO1NC Red", acedido em: 2022-01-13. [Online]. Acessível em: https://pt.aliexpress.com/item/1005003807902469.html?spm=a2g0o.detail.0.0.36ab5fd44gCeGh{&}gps-id=pcDetailBottomMoreThisSeller{&}scm=1007.13339.300834.0{&}scm{_}id=1007.13339.300834.0{&}scm-url=1007.13339.300834.0{&}pvid=b6519a6a-636d-4b97-80a5-b170dc2916b7{&}{_}t=gps-id

[46] "GP2Y0A02YK0F Datasheet", *SHARP*, pp. 1–9, 2006. [Online]. Acessível em: https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk{_}e.pdf

[47] "GP2Y0A21YK0F Datasheet", *SHARP*, pp. 1–9. [Online]. Acessível em: https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk{_}e.pdf

[48] "Pololu Carrier With Sharp GP2Y0A60SZLF Analog Distance Sensor", https://www.rhydolabz.com/wiki/?p=12850, acedido em: 2021-12-18.

[49] Subramanian, "What is Buzzer, Types of Buzzer and Working", https://www.androiderode.com/what-is-buzzer-types-of-buzzer-and-working/, acedido em: 2022-01-05.

[50] AliExpress, "19mm 2 Dual 3 Triple Color RGB LED Light Switch Momentary Self-reset Latching Fixation Waterproof Metal Push Button Switch Power", https://pt.aliexpress.com/item/4000437432585.html?gatewayAdapt=glo2bra{&}spm=a2g0o.order{_}detail.0.0.59ad4c7fkzFSkh, acedido em: 2022-01-13.

[51] P. Properties, "Standard Model A201", pp. 5–6, 2016. [Online]. Acessível em: https://tekscan.com/sites/default/files/resources/FLX-Datasheet-A201-RevI.pdf

[52] A. Freitas, R. M. M. Moraes Furlan, T. V. de Castro Perilo, A. Rodrigues, M. C. Batista Berbert, M. F. Santos Barroso, C. G. da Costa, I. M. Utsch Braga, and E. B. de Las Casas, "Development and Clinical Application of Instruments to Measure Orofacial Structures", *Applied Biological Engineering - Principles and Practice*, no. June 2014, 2012. [Online]. Acessível em: https://www.researchgate.net/publication/221928040{_}Development{_}and{_}Clinical{_}Application{_}of{_}Instruments{_}to{_}Measure{_}Orofacial{_}Structures

[53] AliExpress, "70cm Motorcycle Engine Kill Stop Switch Boat Outboard Engine Motor Kill Stop Switch Safety Lanyard For Marine ATV Quad Yacht", https://pt.aliexpress.com/item/4000337190619.html?spm=a2g0o.productlist.0.0.414ff668OLCWn5{&}algo{_}pvid=5a921a50-ba91-4291-ae31-3381258cab68{&}algo{_}exp{_}id=5a921a50-ba91-4291-ae31-3381258cab68-18{&}pdp{_}ext{_}f={%}7B{%}22sku{_}id{%}22{%}3A{%}2210000001380039059{%}22{%}7D{&}pdp{_}pi=-1{%}3B1.63{%}3B-1{%}3B-1{%}40salePrice{%}3BEUR{%}3Bsearch-mainSearch{&}gatewayAdapt=glo2bra, acedido em: 2022-01-13.

[54] "MPU-6050 GY-521 3-Axis Accel & Gryo Sensor Module", https://protosupplies.com/product/mpu-6050-gy-521-3-axis-accel-gryo-sensor-module/, acedido em: 2021-12-18.

[55] "Ficha ténica fitas", *Eurodin*, pp. 1–48. [Online]. Acessível em: https://dinolux.pt/uploads/2018-05-16-09-30-32-Fitas-led-e-acessrios.pdf

[56] "Sensor De Luz LDR 5mm GL5528", https://www.electrofun.pt/sensores-arduino/sensor-luz-ldr, acedido em: 2021-12-03.

[57] J. A. de Macêdo, L. S. Pedroso, and G. A. da Costa, "Aprimorando e validando um fotogate de baixo custo", *Revista Brasileira de Ensino de Física*, vol. 40, no. 4, 2018. [Online]. Acessível em: https://www.researchgate.net/publication/324992497{_}Aprimorando{_}e{_}validando{_}um{_}fotogate{_}de{_}baixo{_}custo

[58] Tessie, "ESP-01 Wi-Fi Module: ESP-01 Pinout, Programming and ESP-01 VS ESP8266 [FAQ]", https://www.utmel.com/components/esp-01-wi-fi-module-esp-01-pinout-programming-and-esp-01-vs-esp8266-faq?id=990, 2021, acedido em: 2022-05-10.

[59] AliExpress, "Blue Green 12V LY6W Lead Acid Battery Capacity Indicator LCD Digit Display Meter Lithium Battery Power Detector Tester Voltmeter Blue Green 12V LY6W Lead Acid Battery Capacity Indicator LCD Digit Display Meter Lithium Battery Power Detector Tester Voltmet", https://www.aliexpress.com/item/1005001375366049.html?gatewayAdapt=glo2bra{&}spm=a2g0o.order{_}list.0.0.21efcaa44OybC2, acedido em: 2022-01-13.

[60] "Mini MP1584 DC-DC 3A Adjustable Buck Module", https://components101.com/modules/mini-mp1584-dc-dc-3a-adjustable-buck-module, acedido em: 2022-05-01.

[61] "12V 6800mAh Super Rechargeable Protable Li-ion Lithium Battery for DC-12680", https://www.amazon.com/6800mAh-Rechargeable-Protable-Lithium-DC-12680/dp/B08NDJ4FL7, acedido em: 2022-05-02.

[62] Outsystems, "Service Studio: Getting Started."

[63] Outsystems Community, "OutSystems Platform Best Practices", https://success.outsystems.com/Documentation/Best{_}Practices/Development/OutSystems{_}Platform{_}Best{_}Practices, year = 2022, acedido em: 2022-05-20.

[64] J. H. Lee, "General specification of HSR-5995TG coreless digital servo", Korea, p. 1, 2004. [Online]. Acessível em: http://www.letmodel.cz/technical-data/HSR-5995TG.pdf

[65] "TMB12A05 Buzzer Datasheet". [Online]. Acessível em: https://www.quick-teck.co.uk/Management/EEUploadFile/1420788438.pdf

[66] "MPU-6000 and MPU-6050 Product Specification", *InvenSense, Inc*, no. 3.4, p. 52, 2013. [Online]. Acessível em: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[67] "GL55 Series Photoresistor", *GL55 Data Sheet*, pp. 1–6, 2016. [Online]. Acessível em: http://akizukidenshi.com/download/ds/senba/GL55SeriesPhotoresistor.pdf

[68] A.-T. team, "ESP-01 WiFi Module Version 1.0", pp. 1–19, 2015.

[69] "GP2Y0A41SK0F Datasheet", *SHARP*, pp. 1–9, 2002. [Online]. Acessível em: https://www.pololu.com/file/0J713/GP2Y0A41SK0F.pdf

A

Anexos

A.1 Especificações técnicas dos componentes utilizados

Tabela A.1: Especificações técnicas do Arduino Mega 2560

Microcontrolador principal	Atmel ATMega2560 (8 bits)
Microcontrolador USB/serial	Atmel ATMega 16U2
Tensão de operação	5 V
Tensão de entrada (limites)	6 V - 20 V
Tensão de entrada (recomendado)	7 V - 12 V
Pinos entradas/saídas digitais	54 (dos quais 15 oferecem saídas PWM)
Pinos entradas/saídas analógicos	16
Corrente DC por pino	20 mA
Corrente DC para o pino 3,33 V	50 mA
Memória de programas (*flash*)	256 kB dos quais 8 kB usados pelo *bootloader*
Memória volátil (SRAM)	8 kB
Memória não volátil (EEPROM)	4 kB
Velocidade do relógio	16 MHz
Comunicação	USB, ICSP, SPI, I2C, USART
Timers	6 (2 de 8 bits e 4 de 16 bits)
Dimensões	101,52 mm $\times$ 53,3 mm

Tabela A.2: Especificações técnicas do servo motor Hitec HSR-5995TG-Ultra Torque

Faixa de tensão de operação	4,8 V - 6,0 V	7,4 V
Torque máximo	24,0 kg.cm	35,0 kg.cm
Corrente torque máximo	4200 mA	5200 mA
Corrente sem carga	300 mA	380 mA
Velocidade angular	0,15 60º/seg	0,12 60º/seg
Intervalo de rotação	180º	
Ciclo de pulso	20 ms	
Comprimento do pulso	90 - 2100 μs	
Dimensões	40 $\times$ 20 $\times$ 37 mm	
Peso	62 g	
Tipo de motor	Coreless	
Material da engrenagem	Titânio	
Datasheet	Hitec HSR-5995TG [64]	

Tabela A.3: Especificações técnicas dos sensores de distância infravermelhos

	GP2Y0A02YK0F	**GP2Y0A21YK0F**
Tensão de operação	4,5 V - 5,5 V	4,5 V - 5,5 V
Corrente de operação	33 mA (tipicamente)	30 mA (tipicamente)
Faixa de medição	20 cm - 150 cm	10 cm - 80 cm
Saída	Analógica	Analógica
Dimensões	29,5 $\times$ 13 $\times$ 21,6 mm	29,5 $\times$ 13 $\times$ 13,5 mm
Datasheet	Sharp GP2Y0A02YK0F [46]	Sharp GP2Y0A21YK0F [47]

Tabela A.4: Especificações técnicas do piezo buzzer THDZ

Tensão de operação	4 VDC - 8 VDC
Corrente nominal	$\leq$ 32 mA
Som de saída a 10 cm	$\geq$ 85 dB
Frequência ressonante	2300 $\pm$ 300 Hz
Tom	Contínuo
Temperatura de operação	-25ºC a +80ºC
Diâmetro	12 mm
Altura	9,6 mm
Peso	62 g
Datasheet	THDZ 12 mm [65]

Tabela A.5: Especificações técnicas do módulo GY-521 MPU-6050

Tensão de operação	3 V - 5 V (tipicamente)
Comunicação	I2C
Sensibilidade do acelerómetro	±2g, ±4g, ±8g, ±16g (±3% tolerância)
Sensibilidade do giroscópio	±250°/s, ±500°/s, ±1000°/s, ±2000°/s
Gama de temperatura	-40ºC a +85ºC (±1ºC precisão)
Dimensões	21 mm $\times$ 16 mm
Datasheet	MPU-6050 [66]

Tabela A.6: Especificações técnicas do sensor de luminosidade LDR GL5528

Tensão máxima	150 VDC
Potência máxima	100 mW
Tensão de operação	-30ºC a 70ºC
Espetro	540 nm
Resistência no escuro	1 MΩ (Lux 0)
Resistência na luz	10 - 20 kΩ (Lux 10)
Diâmetro	5 mm
Comprimento dos terminais	32 mm
Datasheet	GL55 Series Photoresistor [67]

Tabela A.7: Especificações técnicas do módulo Wi-Fi ESP8266 ESP-01

Microcontrolador principal	Tensilica L106 (32 bits)
Microcontrolador USB/serial	Não possui
Tensão de operação	3.3 V
Corrente de consumo	170 mA
Oscilador (relógio)	80 MHz
Número de pinos	8
Memória de programas (*flash*)	512 kB/ 1 MB/ 4 MB/ 16 MB
Memória volátil (SRAM)	112 kB
Dimensões	14,3 mm × 24,8 mm × 3,0 mm
Datasheet	ESP-01 Wi-Fi Module [68]

Tabela A.8: Especificações técnicas do ecrã de carga de baterias

Tensão de operação	8 VDC - 78 VDC
Corrente de consumo	$<$ 5 mA
Baterias compatíveis	Chumbo-ácido: 12 V (1P); 24 V (2P); 36 V (3P); 48 V (4P) Lítio: 3 células (3S) - 15 células (15S)
Gama de temperatura	0ºC - 40ºC
Acurácia	1 %
Cor da luz do ecrã	Verde ou azul
Dimensões	60 mm × 23 mm × 9 mm

A.2 Circuito externo para a programação da placa Wi-Fi ESP8266 ESP-01

Como especificado na secção 3.10, a alimentação do módulo Wi-Fi ESP8266 ESP-01 com o Arduino é de 3,3 V, portanto para evitar danos ao módulo deve-se reduzir a tensão de 5 V para 3,3 V através de um divisor de tensão. Assim, escolheu-se uma resistência de 1 kΩ, o que resultou em 1,94 kΩ, equação A.1 e figura A.1.

$$\frac{R_2}{R_1} = \frac{V_{out}}{V_{in} - V_{out}} \Leftrightarrow \frac{R_2}{1\ k} = \frac{3,3}{5-3,3} \Leftrightarrow R_2 = 1,94\ k\Omega \tag{A.1}$$

Utilizou-se uma resistência de 2 kΩ, por ser o valor comercial mais próximo deste componente.

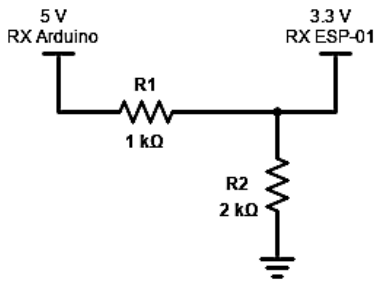


Figura A.1: Divisor de tensão para a tensão de saída do Arduino para ESP-01

Para programar a placa e após alimentá-la devidamente com 3,3 V e respeitar todos os níveis de tensão, é necessário aterrar o pino `GPIO0` e com um sinal em nível baixo (`GND`), acionar o pino `RESET`.
Para facilitar este processo, foi criado um circuito com dois botões que conetam o módulo ESP-01 ao `GND` da placa Arduino e o divisor de tensão supramencionado, figura A.2.

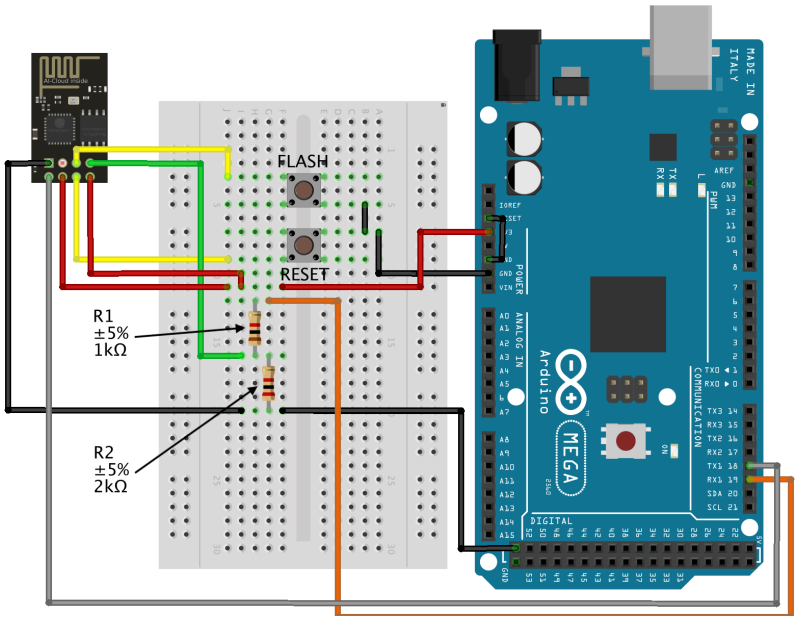


Figura A.2: Circuito para a programação do módulo ESP-01
Fonte: própria fonte

O botão FLASH, quando pressionado, conecta a GPIO0 ao aterramento, exatamente o que faz o botão RESET, que, quando apertado, coneta o pino RESET também ao GND.

Estes botões foram inseridos no esquemático com o propósito de facilitar a programação, pois, quando ela for realizada, basta segurar pressionado o botão FLASH e pressionar apenas uma vez o botão RESET. Feito isso, pode-se enviar o código para a placa e quando a mesma estiver a recebê-lo, um LED indicativo irá piscar.

Cada vez que o código for enviado para a placa, serão necessários os mesmos procedimentos.

Esta placa só deve ser encaixada no conetor da placa de circuito impresso apenas no final do desenvolvimento de todo o protótipo.

A.3 Alimentação e comunicação dos sistemas

A.3.1 Tabela de sistemas - alimentação e comunicação

Tabela A.9: Consumo, tensão de operação e comunicação de cada módulo

Sistema	Modelo	Qnt	Tensão escolhida (V)	Corrente total (mA)	Comunicação ou controlo	Pino digital	Pino anal.	Total pinos
Deteção queda pessoa	Interruptor de emergência [53]	1	-	-	Digital	X		1
Deteção queda andarilho	GY-521 MPU-6050 [66]	1	5	3,9	I2C	X		2
Travagem automática	Hitec HSR-5995TG-Ultra Torque [64]	2	6	3000	PWM	X		2
Deteção obstáculos	Sharp GPY0A21YK0F [46]	2	5	33	Analógico		X	1
	Sharp GPY0A21YK0F [47]	3	5	90	Analógico		X	3
	Botão pressão momentâneo 19 mm [50]	1	5	0,5	Digital	X		2
Alarme e buzina	TMB12A05 [65]	1	5	30	Digital	X		1
Medição força nos punhos	FlexiForce A201 [51]	2	5	1	Analógico		X	2
Sentido movimento das rodas e limite velocidade andarilho	Allegro 1101 [44]	4	5	100	Digital	X		4
	Seletor rotativo 3 posições [45]	1	5	0,5	Digital	X		3
Iluminação presença e avisos	GL5528 [67]	1	5	-	Analógico		X	1
	Fita LED RGB LED HB-108006 [55]	2	12	480	PWM	X		3
Deteção perna presa	Sharp GPY0A21YK0F [46]	2	5	66	Analógico		X	2
	Sharp GP2Y0A41SK0F [69]	2	5	60	Analógico		X	2
Comunicação nuvem	Wi-Fi ESP8266 ESP-01 [68]	1	3,3	170	TXRX	X		2
Nível carga bateria	Ecrã nível carga bateria [59]	1	12	5	VCC	-	-	-
Total						**20**	**11**	**31**

Para a tensão escolhida, a corrente máxima de cada motor é de 4200 mA. No entanto, como o torque máximo nunca é atingido, cada motor foi limitado a 1500 mA, dando um consumo total de 3000 mA pelos servomotores.

NOTA: Clique no modelo do componente para abrir sua folha de dados ou referência.

A.4 Desenvolvimento da placa de circuito impresso

A.4.1 Esquemáticos

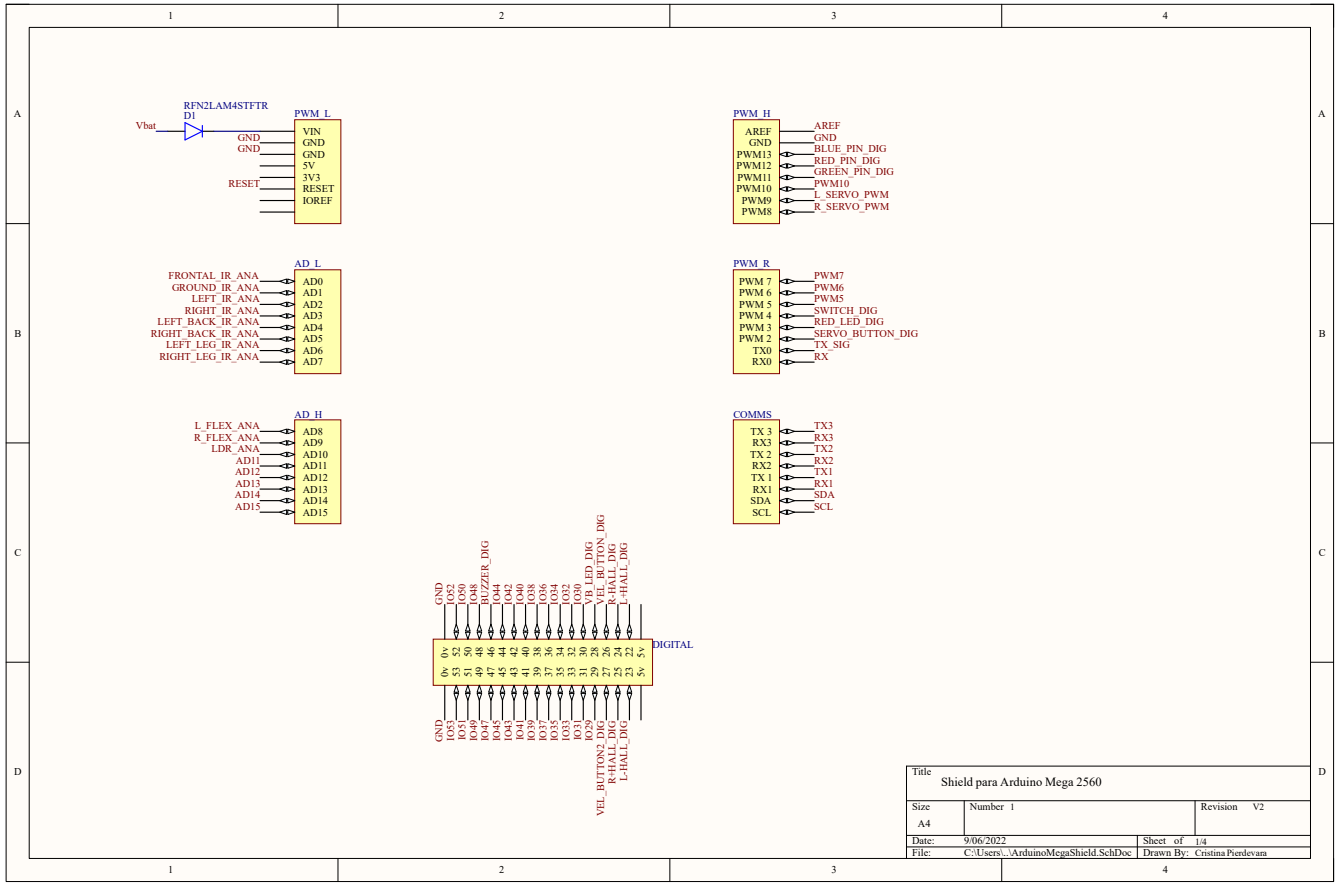


Figura A.3: Conexões da placa de circuito impresso (PCI) com o Arduino Mega 2560

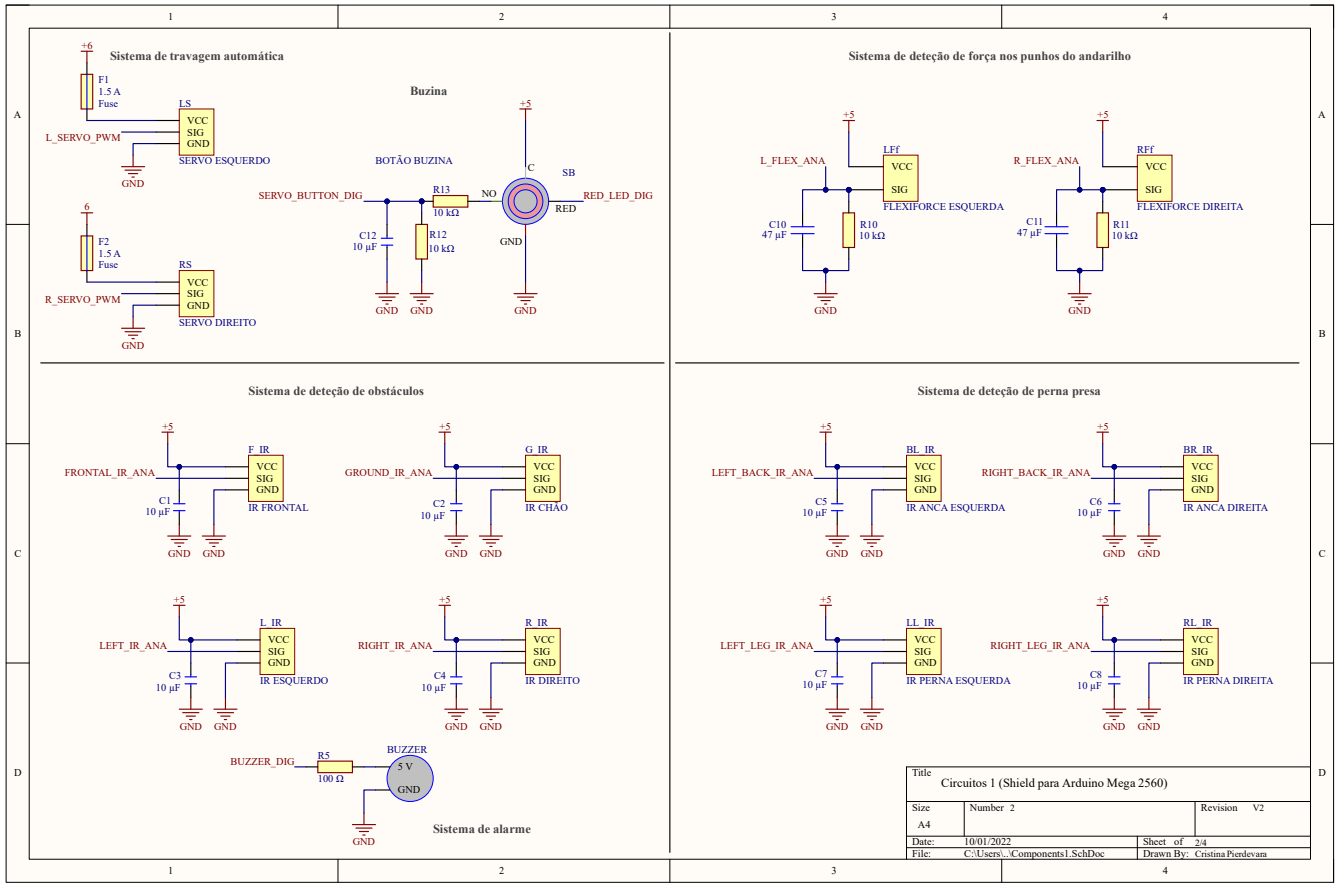


Figura A.4: Esquemáticos dos sistemas implementados - 1

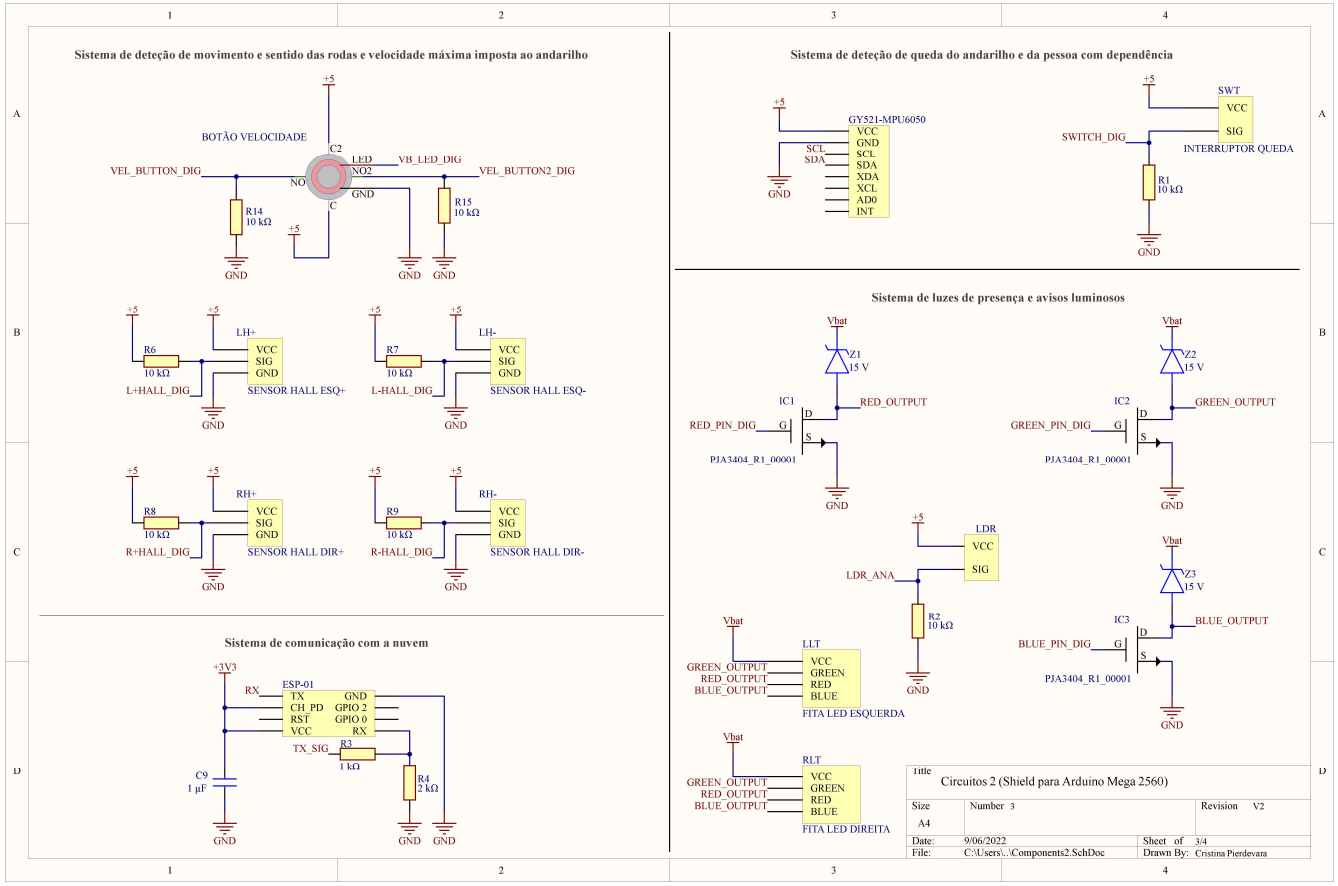


Figura A.5: Esquemáticos dos sistemas implementados - 2

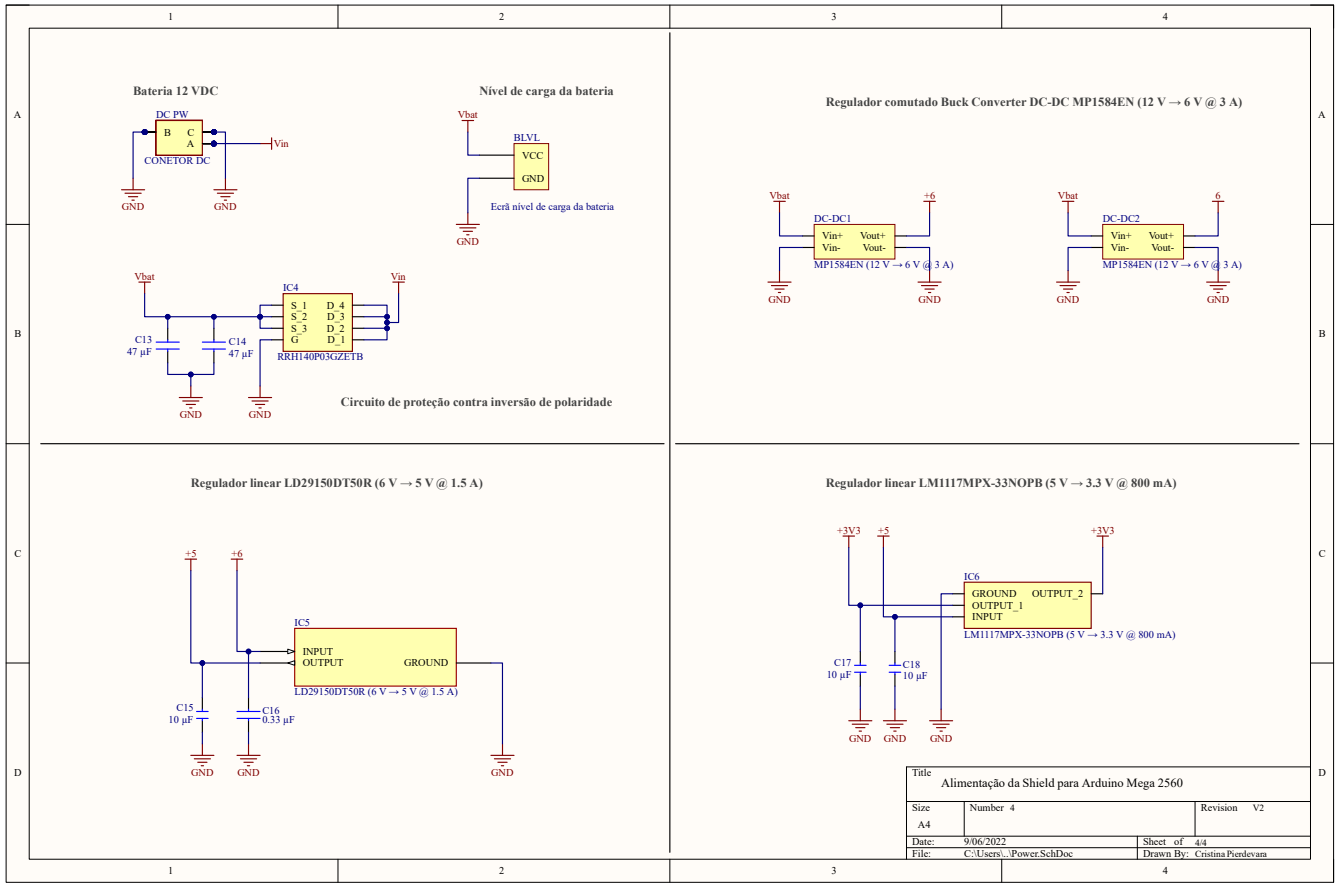


Figura A.6: Esquemático da fonte de alimentação

A.4.2 Camadas da placa

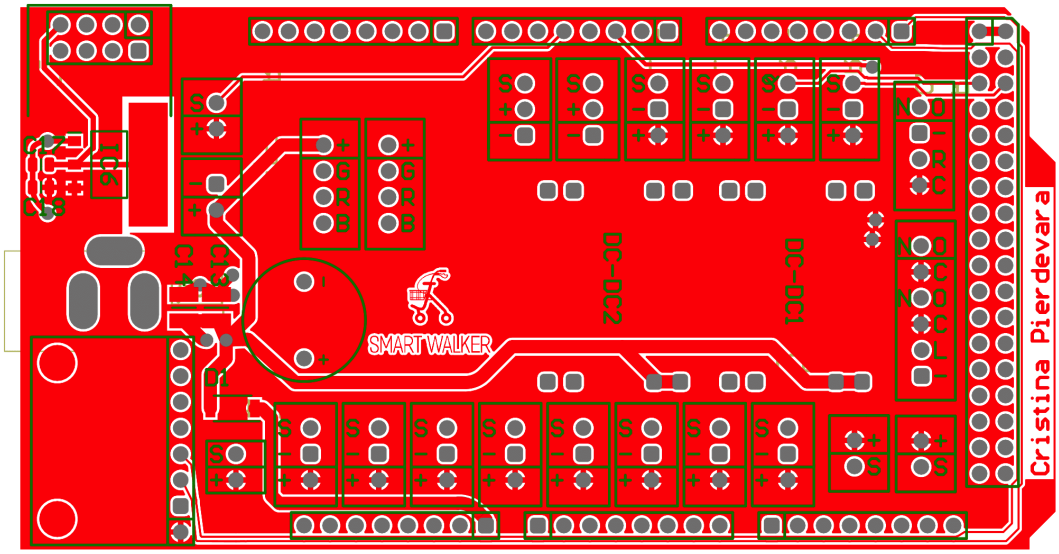


Figura A.7: Camada superior (*TOP*)

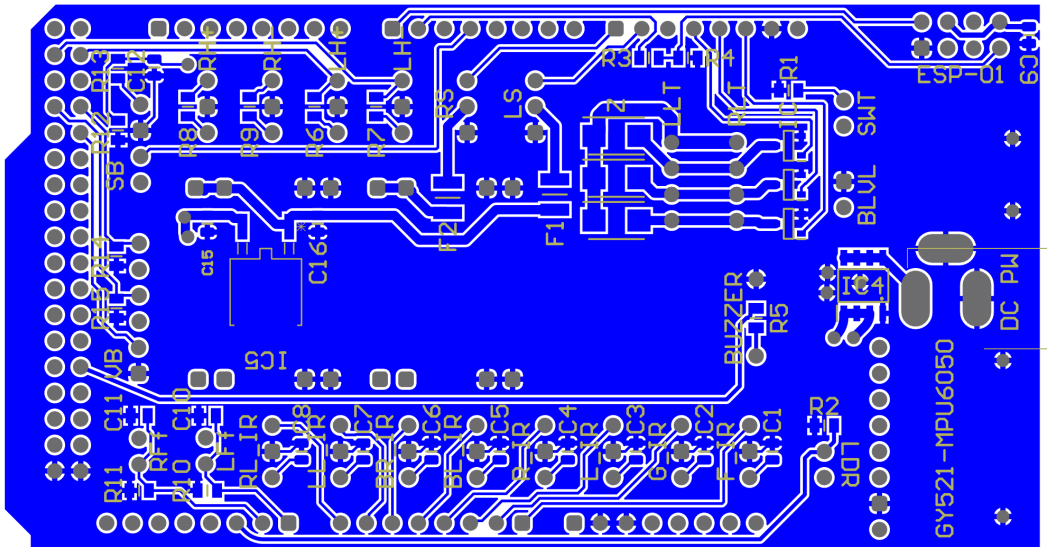


Figura A.8: Camada inferior (*BOTTOM*)

A.4.3 Lista de materiais (BOM)

Tabela A.10: Lista de materiais da placa de circuito impresso

Nome do componente	Número da peça do fabricante	Designador	Quantidade	Preço/un (€)
Condensador de cerâmica	CL21A106KPFNNNF	C1, C2, C3, C4, C5, C6, C7 C8, C12, C15, C17, C18	12	0,105
	885012207022	C9	1	0,095
	GRM21BR61A476ME15K	C10, C11	2	0,732
	885012109011	C13, C14	2	3.560
	CL21B334KBFNNNE	C16	1	0,114
Conetor DC	163-179PH-EX	DC PW	1	1,120
Conversor Buck DC-DC (12 V → 6 V @ 3 A)	MP1584EN	DC-DC1, DC-DC2	2	1,680
Díodo	755-RFN2LAM4STFTR	D1	1	0,475
Regulador linear (6 V → 5 V @ 1,5 A)	LD29150DT50R	IC5	1	1,510
Regulador linear (5 V → 3,3 V @ 800 mA)	LM1117MPX-33NOPB	IC6	1	0,561
MOSFET	PJA3404_R1_00001	IC1, IC2, IC3	3	0,352
	RRH140P03GZETB	IC4	1	2,200
Fusível	1210L075/24PR	F1, F2	2	0,903
Resistência	RK73H2ATTD1002F	R1, R2, R6, R7, R8, R9, R10 R11, R12, R13, R14, R15	12	0,101
	AC0805JR-071KL	R3	1	0,095
	AC0805FR-072KL	R4	1	0,095
	WCR0805-100RFI	R5	1	0,095
Díodo de Zener	SZ1SMB5929BT3G	Z1, Z2, Z3	3	0,770

NOTA: clique no número de peça do fabricante do componente para abrir sua folha de dados.

A.5 Renderização 3D e dimensões da embalagem final

Render 3D

Figura A.9: Vista frontal da embalagem final

Figura A.10: Vista lateral da embalagem final

Figura A.11: Vista traseira da embalagem final

Dimensões da embalagem

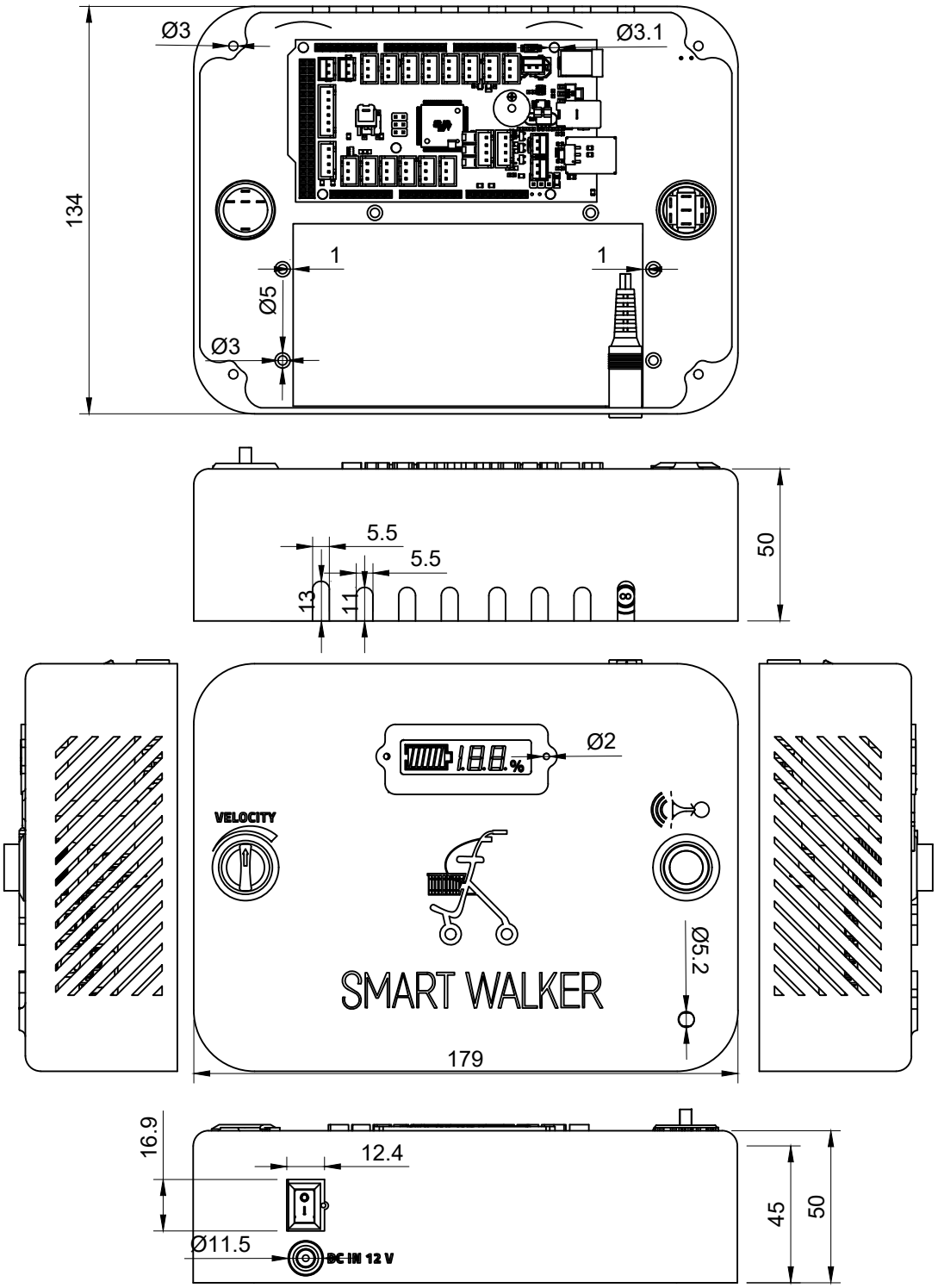


Figura A.12: Dimensões finais da embalagem

A.6 *Mockups* e *layout* final da aplicação

Ecrã *splash-screen*

(a) Versão maquete

(b) Versão implementada

Figura A.13: Ecrã *splash screen* da aplicação

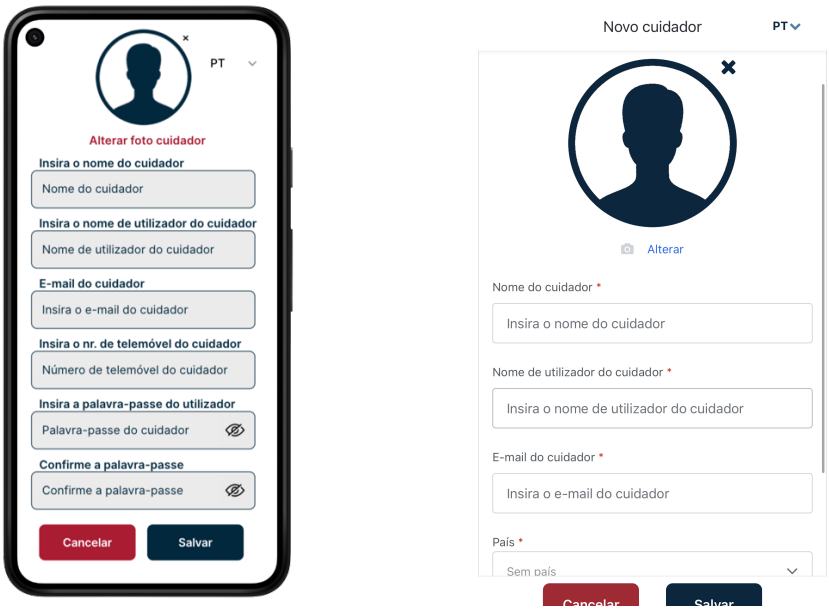
Ecrã de autenticação na aplicação

(a) Versão maquete

(b) Versão implementada

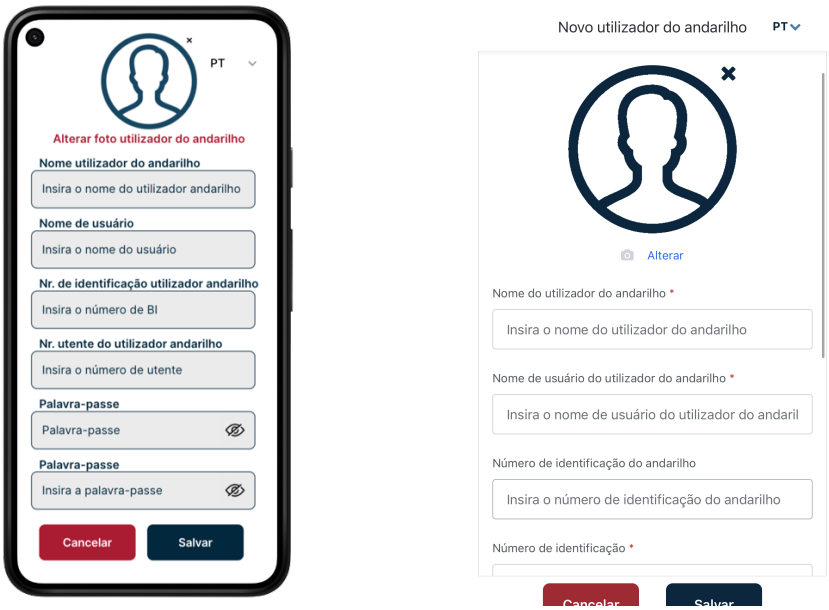
Figura A.14: Ecrã de autenticação na aplicação

Interface do cuidador



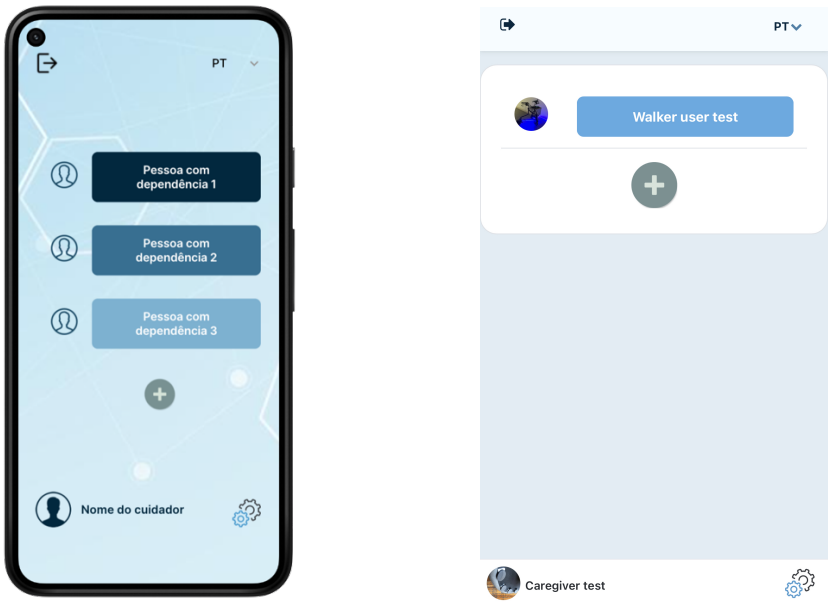
(a) Versão maquete (b) Versão implementada

Figura A.15: Ecrã de nova conta de cuidador



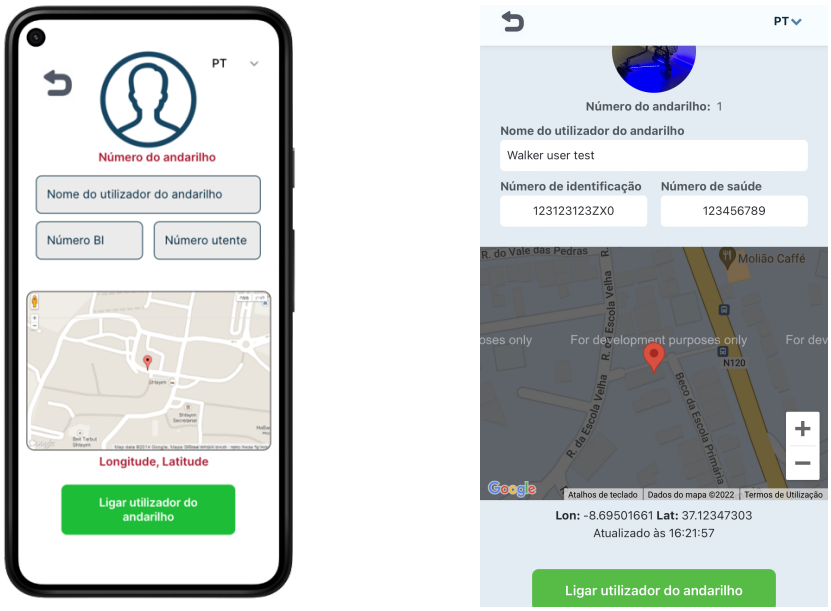
(a) Versão maquete (b) Versão implementada

Figura A.16: Ecrã de nova conta da pessoa com dependência



(a) Versão maquete (b) Versão implementada

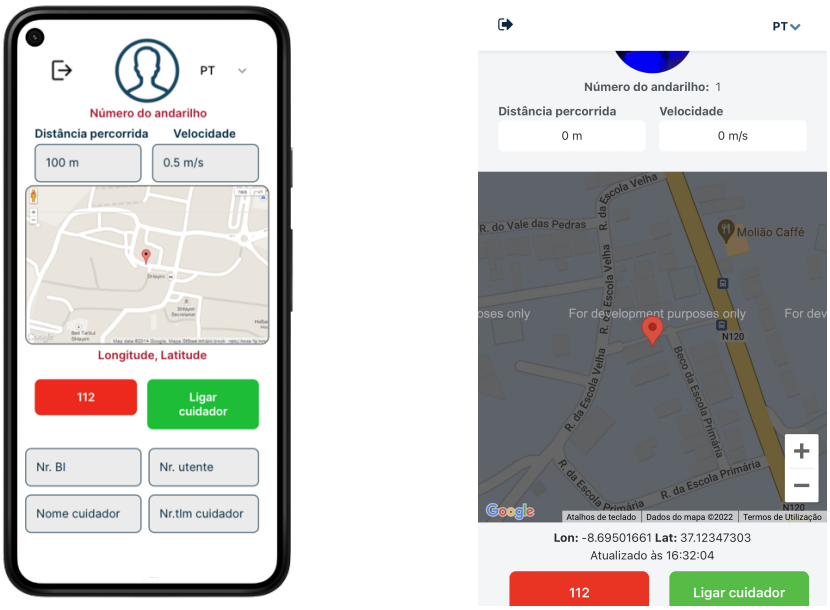
Figura A.17: Ecrã inicial do cuidador



(a) Versão maquete (b) Versão implementada

Figura A.18: Ecrã de localização da pessoa com dependência

A.6.1 Interface do utilizador do andarilho



(a) Versão maquete (b) Versão implementada

Figura A.19: Ecrã inicial do utilizador do andarilho

Ecrã de definições de ambos os utilizadores

Os ecrãs são iguais para ambos os utilizadores, a única alteração muda nos dados pedidos e títulos dos mesmos.

(a) Versão maquete (b) Versão implementada

Figura A.20: Ecrã de definições

A.7 Código fonte do SMARTWALKER

Em continuação irá ser explicado todo o *software* implementado relativo a todos os sistemas desenvolvidos e a programação da placa Wi-Fi ESP8266 ESP-01.

A.7.1 Código fonte para o Arduino

```
/*****************************************************************/
/*           SMARTWALKER - 2021/2022 - TESE DE MESTRADO          */
/*               Cristina Pierdevara - 2015239095                */
/*                  Ultima modificacao: 29Jul2022                */
/*****************************************************************/
#include <Servo.h>    // Biblioteca para os Servos
#include <SharpIR.h>  // Biblioteca para os sensores infravermelhos Sharp
#include <Wire.h>     // Permite a comunicacao I2C

// DEFINICAO DAS CONSTANTES
#define MODEL1080 1080        // IR Sharp com alcance de 10 cm a 80 cm
#define MODEL20150 20150      // IR Sharp com alcance de 20 cm a 20150 cm
#define DIST_GND_DOWN 70      // Valores da distancia quando IR GROUND deteta
    falta de chao (cm)
#define DIST_GND_UP 45        // Valores da distancia quando IR GROUND deteta
    obstaculo (cm)
#define DIST_LEFT 45          // Valores de limite de distancia do IR esquerdo
    (cm)
#define DIST_RIGHT 45         // Valores de limite de distancia do IR direito (
    cm)
#define DIST_FRONTAL 40       // Valores de limite de distancia do IR frontal (
    cm)
#define VELOCITY_TIME_INTERVAL 500 // A velocidade e lida a cada 500 ms
#define LEFT_UNLOCK_ANG 80    // Angulo de desbloqueio da roda esquerda
#define RIGHT_UNLOCK_ANG 105  // Angulo de desbloqueio da roda direita
#define LEFT_LOCK_ANG 120     // Angulo de bloqueio da roda esquerda
#define RIGHT_LOCK_ANG 80     // Angulo de bloqueio da roda direita
#define LOCK_ANG 40           // Graus necessarios para bloquear os travoes
#define THRESHOLD_ACCZ 0.8    // Limiar da aceleracao no eixo Z para se
    considerar queda (m/s^2)
#define BLINKING_TIME_INTERVAL 300 // O LED pisca a cada 300 ms
#define FLEXIFORCE_VALUE 15   // Forca minima para detecao (valor ADC)
#define WHEEL_RADIUS 0.097    // Raio da roda (m)
#define VMAX_THRESHOLD 0.5    // Limiar de travagem (a partir de 50% da
    velocidade maxima)
#define VELOCITY1 0.35        // 0.35 (m/s)
#define VELOCITY2 0.60        // 0.60 (m/s)
#define VELOCITY3 1.00        // 1.00 (m/s)

// DEFINICAO DOS PINOS ANALOGICOS
// Define os pinos do Arduino que se conectam aos pinos Sharp IR
#define frontalIRPin A0   // Pino do Sharp IR frontal
#define groundIRPin A1    // Pino do Sharp IR que aponta para o chao
#define leftIRPin A2      // Pino do Sharp esquerdo
#define rightIRPin A3     // Pino do Sharp direito

// Define os pinos do Arduino que se conectam aos pinos do flexiforce
#define leftFlexiforcePin A8  // Pino da flexiforce esquerda
#define rightFlexiforcePin A9 // Pino da flexiforce direita

// Define os pinos do Arduino que se conectam ao pino LDR
#define ldrPin A10

// DEFINICAO DOS PINOS DIGITAIS
// Define os pinos do Arduino que se conectam aos pinos dos botoes
#define buttonPin 2       // Pino do botao de pressao
#define buttonLed 3       // Pino do LED do botao de pressao

// Define os pinos do Arduino que se conectam ao pino do kill switch
```

```
53 #define fallSwitchPin 4
54
55 // Define os pinos do Arduino que se conectam aos pinos R, G e B da fita LED
56 #define greenLedPin 11
57 #define redLedPin 12
58 #define blueLedPin 13
59
60 // Define os pinos do Arduino que se conectam aos pinos dos sensores Hall
61 #define leftHallFw 22   // Pino do sensor de Hall esquerdo e frontal
62 #define leftHallBw 23   // Pino do sensor de Hall esquerdo e traseiro
63 #define rightHallBw 24  // Pino do sensor de Hall direito e traseiro
64 #define rightHallFw 25  // Pino do sensor de Hall direito e frontal
65
66 // Define os pinos do Arduino que se conectam ao botao de comutacao de tres
       estados
67 #define velocityButtonPin1 26   // Pino do botao de comutacao de tres estados
68 #define velocityButtonPin2 27   // Pino do botao de comutacao de tres estados
69 #define velocityLedButtonPin 28 // Pino do LED do botao de comutacao de tres
       estados
70
71 // Define os pinos do Arduino que se conectam ao pino do buzzer
72 #define buzzerPin 46
73
74 // DECLARACAO DE VARIAVEIS DO SISTEMA DE LUZ
75 int ldrValue; // Armazena o valor lido pelo LDR
76 // Armazena o valor lido pela cor vermelha, verde e azul da fita LED RGB
77 int redValue;
78 int greenValue;
79 int blueValue;
80 long int lastBlinkingTime = 0;   // Temporizador da luz quando pisca
81 bool blinkingState = false;      // True para piscar
82
83 // Variaveis que armazenam a forca aplicada a cada flexiforce
84 int leftFlexiforce;
85 int rightFlexiforce;
86
87 // Inicializa os sensor Sharp IR
88 SharpIR frontalIR = SharpIR(frontalIRPin, MODEL1080);
89 SharpIR groundIR = SharpIR(groundIRPin, MODEL20150);
90 SharpIR leftIR = SharpIR(leftIRPin, MODEL1080);
91 SharpIR rightIR = SharpIR(rightIRPin, MODEL1080);
92
93 // VARIAVEIS DO SENSORES SHARP IR
94 // Armazena o valor das distancias lidas pelo Sharp IR
95 int frontalIR_dist;
96 int groundIR_dist;
97 int leftIR_dist;
98 int rightIR_dist;
99 bool firstReading = true; // Le o primeiro conjunto de valores de distancia dos
       sensores
100 bool objectDetected = false; // Estado de detecao de objeto
101 bool GndNotDetected = false; // Estado da falta de pavimento
102
103 // Declara as variaveis dos servomotores
104 Servo rightServo;
105 Servo leftServo;
106
107 // VARIAVEIS SENSORES HALL
108 // 0 = iman nao detetado, 1 = iman detetado
109 int stateLeftHallFw = 0;
110 int stateLeftHallBw = 0;
111 int stateRightHallFw = 0;
112 int stateRightHallBw = 0;
113
114 // VARIAVEIS DA VELOCIDADE DO ANDARILHO (m/s) E DA DIRECAO DO MOVIMENTO
115 float WHEEL_PULSE = 26.0;   // Numero de pulsos por volta - numero de imanes em
       cada roda
116 float maxVelocity = VELOCITY1;     // Velocidade maxima atual
117 float leftRPM = 0.0;         // Numero de voltas da roda esquerda em meio
       segundo
```

```
118 float rightRPM = 0.0;         // Numero de voltas da roda direita em meio segundo
119 float leftVelocity = 0.0;     // Velocidade da roda esquerda
120 float rightVelocity = 0.0;    // Velocidade da roda direita
121 int leftPulse = 0;            // Numero de pulsos da roda esquerda
122 int rightPulse = 0;           // Numero de pulsos da roda direita
123 // Usado para deteccao da direcao da roda
124 int lastStateLeftHallFw, lastStateRightHallFw;
125 int lastStateLeftHallBw, lastStateRightHallBw;
126 boolean leftForwardDirection = false;   // Sentido de rotacao detetado na roda
        esquerda
127 boolean rightForwardDirection = false;  // Sentido de rotacao detetado na roda
        direita
128 unsigned long int lastLeftVelocityTime = 0; // Temporizador para detecao de
        velocidade da roda esquerda
129 unsigned long int lastRightVelocityTime = 0; // Temporizador para deteccao de
        velocidade da roda direita
130 int angVel = 0;          // Angulo sobre o estado destravado dos travoes para
        enviar aos servo motores - velocidade
131 int angDist = 0;         // Angulo sobre o estado destravado dos travoes para
        enviar aos servo motores - distancia
132 // Variaveis para armazenar o angulo da distancia proporcional atual de cada
        roda
133 int angDistGnd = 0;
134 int angDistLeft = 0;
135 int angDistRight = 0;
136 int angDistMax = 0;
137 float odometerDist = 0;  // Distancia percorrida
138 int odometerPulses = 0;  // Conta o numero de pulsos para calcular a distancia
        percorrida
139
140 // DEFINICOES DO MPU E VARIAVEIS DE QUEDA
141 #define MPU 0x68 // Endereco I2C para o MPU6050
142 float AccX, AccY, AccZ;     // Eixos da aceleracao
143 bool fallDetected = false;  // Estado detecao queda
144 long int fallTimer, lastFallTime; // Temporizadores para deteccao da queda da
        pessoa/andarilho
145 int angFall = 0;            // Angulo sobre o estado destravado dos travoes para
         enviar aos servo motores - queda
146 float rollAvg[16];          // Array para armazenar os valores de AccZ para a
        media deslizante
147
148 int cycles = 0;                 // Numero de ciclos de loops por segundo
149 unsigned long int cycleTimer = 0;   // Temporizador usado para contar o numero
        de ciclos de loops por segundo
150 unsigned long int lastSentMsg = 0;  // Temporizador para controlar o numero de
        mensagens enviadas
151 unsigned long int lastSentFall = 0; // Temporizador para controlar o numero de
        eventos de "quedas" enviados
152 unsigned long int lastSentObstacle = 0; // Temporizador para controlar o numero
        de eventos "objetos detectados" enviados
153 unsigned long int lastSentGndNotDetected = 0; // Temporizador para controlar a
        numero de eventos "falta de pavimento" enviados
154 char myString[256];         // String auxiliar para sprintf()
155
156
157 /*================================ SETUP ================================*/
158 void setup() {
159
160   // Inicializa a saida serie
161   Serial.begin(115200);
162
163   // Anexa pinos aos servos
164   rightServo.attach(8);
165   leftServo.attach(9);
166
167   // Inicializa o MPU-6050
168   Wire.begin();
169   Wire.beginTransmission(MPU);
170   Wire.write(0x6B);
171   Wire.write(0);    // Definido para zero (acorda o MPU-6050)
```

```
172   Wire.endTransmission(true);
173
174   // Escalas do acelerometro
175   /*
176       Wire.write(0b00000000); //  +/-2g
177       Wire.write(0b00001000); //  +/-4g
178       Wire.write(0b00010000); //  +/-8g
179       Wire.write(0b00011000); //  +/-16g
180   */
181
182   Wire.beginTransmission(MPU);
183   Wire.write(0x1C);
184   Wire.write(0b00000000);  // Alterar este comando para a escala desejada
185   Wire.endTransmission();
186
187   // Pino do botao de pressao configurado como entrada:
188   pinMode(buttonPin, INPUT);
189   // Pino do LEd do botao de pressao configurado como saida:
190   pinMode(buttonLed, OUTPUT);
191   // Pino do buzzer configurado como saida:
192   pinMode(buzzerPin, OUTPUT);
193   // Pinos dos sensores Hall configurados como entrada:
194   pinMode(leftHallFw, INPUT);
195   pinMode(leftHallBw, INPUT);
196   pinMode(rightHallFw, INPUT);
197   pinMode(rightHallBw, INPUT);
198   // Pinos das Flexiforce configurados como entrada:
199   pinMode(leftFlexiforce,INPUT);
200   pinMode(rightFlexiforce,INPUT);
201   // Pino do LDR configurado como entrada:
202   pinMode(ldrPin, INPUT);
203   // Pinos das cores da fita LED RGB configurados como saida:
204   pinMode(redLedPin, OUTPUT);
205   pinMode(greenLedPin, OUTPUT);
206   pinMode(blueLedPin, OUTPUT);
207   // Pino do interruptor kill switch configurado como entrada:
208   pinMode(fallSwitchPin, INPUT);
209   // Pinos do botao de comutacao de tres estados configurados como entrada:
210   pinMode(velocityButtonPin1, INPUT);
211   pinMode(velocityButtonPin2, INPUT);
212   // Pinos do LED do botao de comutacao de tres estados configurados como saida:
213   pinMode(velocityLedButtonPin, OUTPUT);
214
215
216   // Garantir que, ao ligar o sistema, o andarilho esta destravado
217   servoUnlock();
218
219   // Inicializa os valores do array da media deslizante com -1G (-9.8 m/s^2
          normalizado)
220   for(int i = 0; i < 16; i++){
221     rollAvg[i] = -1.0;
222   }
223
224 }
225
226 /*================================= LOOP ================================*/
227 void loop(){
228
229   interruptLeftVelocity();    // Para calcular a velocidade da roda esquerda a
          cada meio segundo
230   interruptRightVelocity();   // Para calcular a velocidade da roda direita a
          cada meio segundo
231   walkerSpeed();              // Define a velocidade maxima do andarilho em
          funcao do botao de velocidade
232   walkerHorn();               // Buzina ON ou OFF
233   lightSystem();              // Avisos luminosos e luzes de presenca
234   fallSystem();               // Deteta se ocorreu a queda da pessoa ou do
          andarilho
235   objectDetection();          // Deteta obstaculos e trava proporcionalmente a
          medida que a distancia ao obstaculo diminui
```

```
236   sendToESP();                // Envia eventos para o modulo Wi-Fi (ESP8266 ESP
          -01)
237
238   // Performance/debug do c digo:
239   cycles++;
240   if (millis() - cycleTimer >= 1000){
241    // Conta o numero de ciclos por segundo
242     cycleTimer = millis();
243    // Serial.print("Numero de ciclos por segundo: ");
244    // Serial.println(ciclos);
245     cycles = 0;
246   }
247
248   // Compara qual dos angulos angDistMax, angVel ou angFall e maior de forma
249   // a determinar o angulo certo a ser aplicado
250   if(angVel > angDistMax && !fallDetected){
251     // Angulo final aplicado: unlockAngle + angVel
252     leftServo.write(LEFT_UNLOCK_ANG + angVel);
253     rightServo.write(RIGHT_UNLOCK_ANG - angVel);
254   }
255   else if(fallDetected){
256    // Angulo final aplicado: unlockAngle + angFall
257     leftServo.write(LEFT_UNLOCK_ANG + angFall);
258     rightServo.write(RIGHT_UNLOCK_ANG - angFall);
259   }
260   else if(objectDetected){
261     // Angulo final aplicado: unlockAngle + (angDistLeft ou angDistRight)
262     leftServo.write(LEFT_UNLOCK_ANG + angDistLeft);
263     rightServo.write(RIGHT_UNLOCK_ANG - angDistRight);
264   }
265   else servoUnlock();
266
267 }
268
269 /*======================== SERVO MOTOR TRAVADO/DESTRAVADO
        ========================*/
270 /* ------------------------ SERVO TRAVA -------------------------- */
271 void servoLock(){ // Andarilho completamente travado
272     // angulos de travagem maxima
273     leftServo.write(LEFT_LOCK_ANG);
274     rightServo.write(RIGHT_LOCK_ANG);
275 }
276
277 /* ------------------------ SERVO NAO TRAVA ------------------------- */
278 void servoUnlock(){ // Andarilho destravado
279     // angulos de destravagem
280     leftServo.write(LEFT_UNLOCK_ANG);
281     rightServo.write(RIGHT_UNLOCK_ANG);
282 }
283
284 /* ------ TRAVAGEM PROPROCIONAL A VELOCIDADE ------ */
285 void servoSoftLock_velocity(){ // A travagem automatica atua de forma
        proporcional a velocidade
286   // ENTRADA: maxVelocity, leftVelocity, rightVelocity
287   // SAiDA: angVel -> angulo de travagem do servo devido a velocidade
288   // Se a pessoa atingir VMAX_THRESHOLD da velocidade maxima, a travagem
          automatica passa a atuar proporcionalmente
289   // angVel = lockAng, para vel > maxVelocity
290   // angVel = 0, para vel < VMAX_THRESHOLD * maxVelocity
291   // angVel = (vel - VMAX_THRESHOLD * maxVelocity) * LOCK_ANG / ((1 -
          VMAX_THRESHOLD) * maxVelocity),
292   // para vel > VMAX_THRESHOLD * maxVelocity e vel < maxVelocity
293
294   // Permite que o andarilho gire e leve em conta a diferenca de velocidade
          entre as duas rodas
295   float vel = (leftVelocity + rightVelocity)/2;
296   if(vel > maxVelocity) angVel = LOCK_ANG;  // Trava
297   else if(vel < VMAX_THRESHOLD * maxVelocity) angVel = 0;   // Destrava
298   else angVel = ((vel - VMAX_THRESHOLD * maxVelocity) * LOCK_ANG) / ((1 -
          VMAX_THRESHOLD) * maxVelocity);
```

```
299
300 }
301
302 /*======================== DETECAO DE OBJETOS  ========================*/
303 /* ------ ANDARILHO TRAVA EM FUNCAO DA DISTANCIA ------ */
304 void objectDetection(){ // Os servomotores atuam a medida que a pessoa se
        aproxima de um obstaculo
305   // ENTRADA: frontalIR_dist, groundIR_dist, leftIR_dist, rightIR_dist
306   // SAIDA: angDistX -> angulo de travagem do servo devido a distancia
307   // Se a pessoa atingir a distancia minima ao objeto, a travagem automatica
          passa a atuar
308   // de forma inversamente proporcional
309   // Para Sharp IR que aponta para baixo detetar o degrau: ang = lockAng para
          DIST_GND_DOWN > 70
310   // Para Sharp IR que aponta para uma escada/obstaculo:
311   // angDistGnd = lockAng para DIST_GND_UP < 45
312   // Para os restantes Sharp IR (10 cm - 80 cm):
313   // (se a pessoa estiver a 45 cm do obstaculo os travoes comecam a atuar e a 30
          cm travam completamente):
314   // angDistX = lockAng, para DIST_X < 30
315   // angDistX = 0, para DIST_X > 45
316   // angDistX = - (LOCK_ANG * (dist - 45))/15, para DIST_X > 30  e DIST_X < 45
317
318   // Faz a leitura alternada das distancias para economizar tempo e melhorar o
          desempenho
319   // Com bool firstReading: ciclos = 512; Sem: ciclos passam para metade
320   if(firstReading){
321     firstReading = false;
322     frontalIR_dist = frontalIR.distance();
323     groundIR_dist = groundIR.distance();
324   }
325   else{ // first reading == false
326     firstReading = true;
327     leftIR_dist = leftIR.distance();
328     rightIR_dist = rightIR.distance();
329   }
330
331   /* Sem bool firstReading, o numero de ciclos diminui para 282
332   frontalIR_dist = frontalIR.distance();
333   groundIR_dist = groundIR.distance();
334   leftIR_dist = leftIR.distance();
335   rightIR_dist = rightIR.distance();
336   */
337
338   if(groundIR_dist > DIST_GND_DOWN){ // Degrau para baixo
339     objectDetected = true;
340     GndNotDetected = true;
341     // A travagem automatica e totalmente ativada se DIST_GND_DOWN > 70 (degrau
          para baixo)
342     angDistMax = LOCK_ANG;
343     angDistLeft = LOCK_ANG;
344     angDistRight = LOCK_ANG;
345   }
346   else if(frontalIR_dist > DIST_FRONTAL && leftIR_dist > DIST_LEFT &&
        rightIR_dist > DIST_RIGHT && groundIR_dist > DIST_GND_UP){
347     // O andarilho destrava
348     angDistLeft = 0;
349     angDistRight = 0;
350     objectDetected = false;
351     GndNotDetected = false;
352   }
353
354   else{ // Obstaculo detetado
355     objectDetected = true;
356     GndNotDetected = false;
357
358     // Testa se o sensor IR deteta pavimento irregular
359     if(groundIR_dist > DIST_GND_UP) angDistGnd = 0;
360     else angDistGnd = - (LOCK_ANG / 15) * (groundIR_dist - DIST_GND_UP); // 15 =
            30-45 (cm)
```

```
361     if(angDistGnd > LOCK_ANG) angDistGnd = LOCK_ANG;
362
363     // Testa se o sensor IR deteta objeto a frente do andarilho
364     if(frontalIR_dist > DIST_FRONTAL) angDist = 0;
365     else angDist = - (LOCK_ANG / 20) * (frontalIR_dist - DIST_FRONTAL); // 20 =
            20-40 (cm)
366     if(angDist > LOCK_ANG) angDist = LOCK_ANG;
367
368     // Testa se o sensor IR deteta objeto do lado esquerdo do andarilho
369     if(leftIR_dist > DIST_LEFT) angDistRight = 0;
370     else angDistRight = - (LOCK_ANG / 15) * (leftIR_dist - DIST_LEFT);
371     if(angDistRight > LOCK_ANG) angDistRight = LOCK_ANG;
372
373     // Testa se o sensor IR deteta objeto do lado direito do andarilho
374     if(rightIR_dist > DIST_RIGHT) angDistLeft = 0;
375     else angDistLeft = - (LOCK_ANG / 15) * (rightIR_dist - DIST_RIGHT);
376     if(angDistLeft > LOCK_ANG) angDistLeft = LOCK_ANG;
377
378     if(angDistGnd > angDist) angDist = angDistGnd;
379     if(angDist > angDistRight) angDistRight = angDist;
380     if(angDist > angDistLeft) angDistLeft = angDist;
381
382     // Compara qual dos sensores requer mais poder de travagem
383     if(angDistLeft > angDistRight){
384       if(angDistLeft > angDist) angDistMax = angDistLeft;
385       else{
386           angDistLeft = angDist;
387           angDistRight = angDist;
388       }
389     }
390     else{
391       if(angDistRight > angDist) angDistMax = angDistRight;
392       else{
393           angDistLeft = angDist;
394           angDistRight = angDist;
395       }
396     }
397   }
398 }
399
400 /*============== FUNCAO DA DIRECAO DO MOVIMENTO DE CADA RODA ==============*/
401 /* ------------------------ DIRECAO DA RODA ESQUERDA ---------------------- */
402 int leftWheelDirection(int lastStateHallFw, int lastStateHallBw, int stateHallFw
        , int stateHallBw){
403 // Retorna a direcao da roda do andarilho: movimento para frente = 1, para tras
        = 0;
404 // Quando o andarilho esta parado, a funcao retorna o valor 0, ou seja, o
        andador tem a direcao para tras
405
406     /*
407       Serial.print(lastStateHallBw);
408       Serial.print(lastStateHallFw);
409       Serial.print(stateHallBw);
410       Serial.println(stateHallFw);
411     */
412
413   if((lastStateHallBw == 1 && lastStateHallFw == 1) && (stateHallBw == 1 &&
          stateHallFw == 0))
414     leftForwardDirection = true;
415   else if((lastStateHallBw == 1 && stateHallBw == 0) && (lastStateHallFw == 1 &&
          stateHallFw == 0))
416     leftForwardDirection = true;
417   else if((lastStateHallBw == 1 && stateHallBw == 0) && (lastStateHallFw == 0 &&
          stateHallFw == 0))
418     leftForwardDirection = true;
419   else if((lastStateHallBw == 0 && stateHallBw == 0) && (lastStateHallFw == 0 &&
          stateHallFw == 1))
420     leftForwardDirection = true;
421   else if((lastStateHallBw == 0 && stateHallBw == 1) && (lastStateHallFw == 1 &&
          stateHallFw == 1))
```

```
422     leftForwardDirection = true;
423   else
424     leftForwardDirection = false;
425
426   if(leftForwardDirection) return(1);
427   else return(0);
428
429 }
430
431 /* ------------------------ DIRECAO DO MOVIMENTO DA RODA DIREITA
      ---------------------- */
432 int rightWheelDirection(int lastStateHallFw, int lastStateHallBw, int
      stateHallFw, int stateHallBw){
433 // Retorna a direcao da roda do andarilho: movimento para frente = 1, para tras
      = 0;
434 // Quando o andarilho esta parado, a funcao retorna o valor 0, ou seja, o
      andador tem a direcao para tras
435
436     /*
437       Serial.print(lastStateHallBw);
438       Serial.print(lastStateHallFw);
439       Serial.print(stateHallBw);
440       Serial.println(stateHallFw);
441     */
442
443   if((lastStateHallBw == 1 && lastStateHallFw == 1) && (stateHallBw == 1 &&
        stateHallFw == 0))
444     rightForwardDirection = true;
445   else if((lastStateHallBw == 1 && stateHallBw == 0) && (lastStateHallFw == 1 &&
        stateHallFw == 0))
446     rightForwardDirection = true;
447   else if((lastStateHallBw == 1 && stateHallBw == 0) && (lastStateHallFw == 0 &&
        stateHallFw == 0))
448     rightForwardDirection = true;
449   else if((lastStateHallBw == 0 && stateHallBw == 0) && (lastStateHallFw == 0 &&
        stateHallFw == 1))
450     rightForwardDirection = true;
451   else if((lastStateHallBw == 0 && stateHallBw == 1) && (lastStateHallFw == 1 &&
        stateHallFw == 1))
452     rightForwardDirection = true;
453   else
454     rightForwardDirection = false;
455
456   if(rightForwardDirection) return(1);
457   else return(0);
458
459 }
460
461 /* ---------------------- VELOCIDADE RODA ESQUERDA ------------------------ */
462 void interruptLeftVelocity(){ // Velocidade da roda esquerda do andarilho
463 // Chama a funcao de velocidade sempre que o sensor Hall for de 0 a 1 (e
      equivalente a uma interrupcao)
464   lastStateLeftHallFw = stateLeftHallFw;
465   lastStateLeftHallBw = stateLeftHallBw;
466
467   stateLeftHallFw = !digitalRead(leftHallFw);
468   stateLeftHallBw = !digitalRead(leftHallBw);
469
470   // Serial.print(stateLeftHallFw);
471   // Serial.println(stateLeftHallBw);
472
473   if(lastStateLeftHallFw == 0 && stateLeftHallFw == 1){ // Um novo iman passou
        pelo sensor do Hall frontal
474     leftPulse++;
475     leftWheelDirection(lastStateLeftHallFw, lastStateLeftHallBw, stateLeftHallFw
          , stateLeftHallBw);
476
477     if(rightForwardDirection) odometerPulses++;
478
479     /*
```

```
480     if(leftWheelDirection(lastStateLeftHallFw, lastStateLeftHallBw,
            stateLeftHallFw, stateLeftHallBw) == 1){
481       Serial.println("Roda esquerda: movimento para a frente");
482     }
483     else{
484       Serial.println("Roda esquerda: movimento para tras");
485     }
486     Serial.println("Pulso esquerdo ++");
487     */
488
489   }
490
491   if(millis() - lastLeftVelocityTime >= VELOCITY_TIME_INTERVAL){  // Atualiza
          contador a cada meio segundo
492
493     leftRPM = leftPulse / WHEEL_PULSE;
494     leftVelocity = (2 *  PI * WHEEL_RADIUS * leftRPM) * 1000/(
            VELOCITY_TIME_INTERVAL);
495
496     lastLeftVelocityTime = millis();
497
498     /*
499     Serial.print("Left pulse: ");
500     Serial.println(leftPulse);
501     Serial.print(" ");
502     Serial.print("Left velocity:");
503     Serial.println(leftVelocity);
504     Serial.println("==================");
505     */
506
507     leftPulse = 0;
508
509   }
510 }
511
512 /* ---------------------- VELOCIDADE RODA DIREITA ------------------------ */
513 void interruptRightVelocity(){ // Velocidade da roda direita do andarilho
514 // Chama a funcao de velocidade sempre que o sensor Hall for de 0 a 1 (e
        equivalente a uma interrupcao)
515
516   lastStateRightHallFw = stateRightHallFw;
517   lastStateRightHallBw = stateRightHallBw;
518
519   stateRightHallFw = !digitalRead(rightHallFw);
520   stateRightHallBw = !digitalRead(rightHallBw);
521
522   // Serial.print(stateRightHallFw);
523   // Serial.println(stateRightHallBw);
524
525   if(lastStateRightHallFw == 0 && stateRightHallFw == 1){ // Um novo iman passou
          pelo sensor Hall frontal
526
527     rightPulse++;
528     rightWheelDirection(lastStateRightHallFw, lastStateRightHallBw,
            stateRightHallFw, stateRightHallBw);
529     if(rightForwardDirection) odometerPulses++;
530
531     /*
532     if(rightWheelDirection(lastStateRightHallFw, lastStateRightHallBw,
            stateRightHallFw, stateRightHallBw) == 1){
533       Serial.println("Roda direita: movimento para a frente");
534      }
535     else{
536       Serial.println("Roda direita: movimento para tras");
537      }
538     Serial.println("Pulso direito ++");
539     */
540   }
541
```

```
542   if(millis() - lastRightVelocityTime >= VELOCITY_TIME_INTERVAL){  // Atualiza
          contador a cada meio segundo
543
544     rightRPM = rightPulse / (WHEEL_PULSE);
545     rightVelocity = (2 *  PI * WHEEL_RADIUS * rightRPM) * 1000/(
            VELOCITY_TIME_INTERVAL);
546
547     lastRightVelocityTime = millis();
548
549     /*
550     Serial.print("Pulsos direitos:");
551     Serial.print(rightPulse);
552     Serial.print(" ");
553     Serial.print("Velocidade direita:");
554     Serial.println(rightVelocity);
555     Serial.println("==================");
556     */
557
558     rightPulse = 0;
559
560   }
561 }
562
563 /*============== VELOCIDADE MAXIMA IMPOSTA AO ANDARILHO ==============*/
564 void walkerSpeed(){ // Calcula a velocidade maxima do andarilho ate o maximo
        permitido
565   // SAIDA: maxVelocity
566
567   if(fallDetected) return;
568
569   if((digitalRead(velocityButtonPin1) == 1) && (digitalRead(velocityButtonPin2)
          == 0)){
570
571     // Velocidade minima
572     maxVelocity = VELOCITY1;
573
574     if(leftVelocity > VMAX_THRESHOLD * maxVelocity || rightVelocity >
            VMAX_THRESHOLD * maxVelocity)
575       servoSoftLock_velocity();  // Funcao que trava as rodas do andarilho
              proporcionalmente
576     else angVel = 0;     // Rodas do andarilho desbloqueadas
577   }
578
579   else if((digitalRead(velocityButtonPin1) == 0) && (digitalRead(
          velocityButtonPin2) == 0)){
580
581     // Velocidade intermedia
582     maxVelocity = VELOCITY2;
583
584     if(leftVelocity > VMAX_THRESHOLD * maxVelocity || rightVelocity >
            VMAX_THRESHOLD * maxVelocity)
585       servoSoftLock_velocity();
586     else angVel = 0; // Rodas do andarilho desbloqueadas
587   }
588
589   else if((digitalRead(velocityButtonPin1) == 0) && (digitalRead(
          velocityButtonPin2) == 1)){
590
591     // Velocidade maxima
592     maxVelocity = VELOCITY3;
593
594     if(leftVelocity > VMAX_THRESHOLD * maxVelocity || rightVelocity >
            VMAX_THRESHOLD * maxVelocity)
595       servoSoftLock_velocity();
596     else angVel = 0; // Rodas do andarilho desbloqueadas
597   }
598
599 }
600
601 /*============== BUZINA DO ANDARILHO ==============*/
```

```
602 void walkerHorn(){ // Buzina apita quando o botao e pressionado
603
604   if(digitalRead(buttonPin) == HIGH){ // Botao pressionado
605     digitalWrite(buttonLed, HIGH); // Luz do botao ON
606     digitalWrite(buzzerPin, HIGH); // Buzzer ON
607    }
608    else if (!fallDetected && !objectDetected && !GndNotDetected){ // Bottao nao
           pressionado
609     digitalWrite(buttonLed, LOW); // Luz do botao OFF
610     digitalWrite(buzzerPin, LOW); // Buzzer OFF
611   }
612 }
613
614 /*============================ SISTEMA LUMINOSO ============================*/
615 /* ------------------------ LUZES DE PRESENCA ------------------------- */
616 void lightSystem(){ // Controla o sistema de luzes de presenca e de emergencia
617
618   // Este sistema esta ativo somente quando nao ocorreu nenhuma queda ou objeto/
        falta de pavimentos nao detetados
619   if (!fallDetected && !objectDetected && !GndNotDetected){
620
621     ldrValue = analogRead(ldrPin);  // Le o estado do valor LDR
622
623     // Cor branca: rgb(255, 255, 255)
624     // Serial.println(ldrValue);
625
626     // LUZ DA NOITE (ESCURO)
627     if(ldrValue < 400){
628       // Cor branca: rgb(255, 255, 255)
629       analogWrite(redLedPin, 255);
630       analogWrite(greenLedPin, 255);
631       analogWrite(blueLedPin, 255);
632     }
633     // LUZ DA NOITE (ENTARDECER)
634     else if(ldrValue > 400 && ldrValue < 500){
635       analogWrite(redLedPin, 255/2);
636       analogWrite(greenLedPin, 255/2);
637       analogWrite(blueLedPin, 255/2);
638     }
639     // LUZ MeDIA DO DIA
640      else if(ldrValue > 500 && ldrValue < 800){
641       analogWrite(redLedPin, 255/3);
642       analogWrite(greenLedPin, 255/3);
643       analogWrite(blueLedPin, 255/3);
644     }
645     // LUZ DO DIA (MUITO CLARO)
646     else{
647       // Luzes OFF
648       analogWrite(redLedPin, 0);
649       analogWrite(greenLedPin, 0);
650       analogWrite(blueLedPin, 0);
651     }
652   }
653
654   // A luz vermelha pisca quando a queda e detetada e a cor amarela pisca quando
        um objeto e detetado
655   else warningLights();
656
657 }
658
659 /* ------------------------ LUZES PISCAM ------------------------- */
660 void warningLights(){ // Luzes piscam quando ha emergencia ou objeto detetado
661   // Cor para emergencia: vermelho (255, 0, 0)
662   if(fallDetected){
663     if((millis() - lastBlinkingTime) > BLINKING_TIME_INTERVAL){
664       if(blinkingState){
665         // Luz OFF
666         analogWrite(redLedPin, 0);
667         analogWrite(greenLedPin, 0);
668         analogWrite(blueLedPin, 0);
```

```
669         blinkingState = false;
670         digitalWrite(buzzerPin, HIGH);
671         digitalWrite(buttonLed, LOW);
672         lastBlinkingTime = millis();
673       }
674       else{
675         // Cor vermelha: (255, 0, 0)
676         analogWrite(redLedPin, 255);
677         analogWrite(greenLedPin, 0);
678         analogWrite(blueLedPin, 0);
679         blinkingState = true;
680         digitalWrite(buttonLed, HIGH);
681         digitalWrite(buzzerPin, LOW);
682         lastBlinkingTime = millis();
683       }
684     }
685   }
686
687   // Cor para objeto detetado: amarelo (255, 255, 0)
688   else if(objectDetected || GndNotDetected){
689     if((millis() - lastBlinkingTime) > (BLINKING_TIME_INTERVAL - angDistMax*6)){
690       if(blinkingState == true){
691         // Luzes OFF
692         analogWrite(redLedPin, 0);
693         analogWrite(greenLedPin, 0);
694         analogWrite(blueLedPin, 0);
695         blinkingState = false;
696         digitalWrite(buzzerPin, HIGH);
697         digitalWrite(buttonLed, HIGH);
698         lastBlinkingTime = millis();
699       }
700       else{
701         // Cor amarela: rgb(255, 230, 0)
702         analogWrite(redLedPin, 255);
703         analogWrite(greenLedPin, 230);
704         analogWrite(blueLedPin, 0);
705         blinkingState = true;
706         digitalWrite(buttonLed, LOW);
707         digitalWrite(buzzerPin, LOW);
708         lastBlinkingTime = millis();
709       }
710     }
711   }
712 }
713
714 /*============== QUEDA DETETADA ==============*/
715 void fallSystem(){  // Sistema de deteccao de queda
716
717   // 21 ms e o tempo entre cada amostra AccZ, para permitir 3 ciclos completos
        (16 amostras)/segundo
718   if(millis() - fallTimer >= 21){
719     fallTimer = millis();
720
721     // Comandos para iniciar a transmissao de dados
722     Wire.beginTransmission(MPU);
723     Wire.write(0x3F);    // Comeca no registro 0x3B (ACCEL_XOUT_H)
724     Wire.endTransmission(false);
725     Wire.requestFrom(MPU, 2, true); // Solicita dados do sensor, solicita um
          total de 2 registos
726
727     // Armazena o valor dos sensores nas variaveis correspondentes
728     //AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // 0x3B (ACCEL_XOUT_H)
          & 0x3C (ACCEL_XOUT_L)
729     //AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // 0x3D (ACCEL_YOUT_H)
          & 0x3E (ACCEL_YOUT_L)
730     AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // 0x3F (ACCEL_ZOUT_H) &
          0x40 (ACCEL_ZOUT_L)
731
732     /* Alterar a divisao de acordo com a base de escala escolhida:
733         Acelerometro
```

```
734         +/-2g = 16384
735         +/-4g = 8192
736         +/-8g = 4096
737         +/-16g = 2048
738     */
739
740     // Calcule a media deslizante dos ultimos 16 valores de AccZ
741     float rollingAverage = 0.0;
742
743     for(int i = 16; i > 0; i--){
744       rollAvg[i] = rollAvg[i-1];
745       rollingAverage += rollAvg[i];
746     }
747
748     rollAvg[0] = AccZ;
749
750     rollingAverage = (rollingAverage + rollAvg[0])/16;
751
752     if(digitalRead(fallSwitchPin) == HIGH){   // Kill switch ativado
753       // Serial.print("QUEDA DETETADA = KS rAvg= ");
754       // Serial.println(rollingAverage);
755       fallDetected = true;
756       angFall = LOCK_ANG;
757       lastFallTime = millis();
758     }
759
760     // AccZ do MPU acima do THRESHOLD
761     else if(pow(rollingAverage, 2) < pow(THRESHOLD_ACCZ, 2)){
762        // Serial.print("QUEDA DETETADA = MPU rAvg= ");
763        // Serial.println(rollingAverage);
764       fallDetected = true;
765       angFall = LOCK_ANG;
766       // digitalWrite(buzzerPin, HIGH); // Buzzer ON quando queda detetada
767       lastFallTime = millis();
768     }
769
770     // Para parar o sistema de queda (killSwitch = LOW ou MPU AccZ >
          THRESHOLD_ACCZ)
771     // As flexiforce devem detetar a forca de ambos os pulsos
772     if((millis() - lastFallTime >= 3000) && (fallDetected == true)){
773       if((analogRead(leftFlexiforcePin) >= FLEXIFORCE_VALUE) && (analogRead(
              rightFlexiforcePin) >= FLEXIFORCE_VALUE)){
774         fallDetected = false;
775         angFall = 0;
776         digitalWrite(buzzerPin, LOW); // Buzzer OFF quando queda detetada
777       }
778     }
779   }
780 }
781
782 /*============== ODOMIMETRO  ==============*/
783 void walkerOdometer(){  // Calcula a distancia percorrida pela pessoa
784 // Funciona apenas quando o movimento e para a frente
785 // (medida por interruptLeftVelocity() e interruptRightVelocity())
786
787   odometerDist = (odometerPulses * 2 *  PI * WHEEL_RADIUS) / WHEEL_PULSE;
788   // OdometerDist recebe os pulsos de ambas as rodas, portanto deve ser dividido
        por 2
789   odometerDist = odometerDist / 2;
790
791   /*
792   Serial.print("Distancia percorrida: ");
793   Serial.print(odometerDist);
794   Serial.print(" Pulsos: ");
795   Serial.println(odometerPulses);
796   */
797 }
798
799
```

```
800 /*======================== ENVIAR EVENTOS PARA ESP8266 ESP-01
        ==========================*/
801 void sendToESP(){ // Envia eventos para o modulo Wi-Fi (ESP8266 ESP-01)
802
803   char cStrDist[12];
804   char cStrVel[12];
805
806   if(fallDetected){
807     if((millis() - lastSentFall) >= 5000){
808       walkerOdometer();           // Calcula a distancia percorrida
809       dtostrf(odometerDist, 4, 2, cStrDist);
810       dtostrf((leftVelocity + rightVelocity)/2, 4, 2, cStrVel);
811       sprintf(myString, "E1|%s|%s", cStrDist, cStrVel);
812       Serial.println(myString);
813       lastSentFall = millis();
814       lastSentMsg = lastSentFall;
815     }
816   }
817   else if(GndNotDetected){
818     if((millis() - lastSentGndNotDetected) >= 2500){
819       walkerOdometer();
820       dtostrf(odometerDist, 4, 2, cStrDist);
821       dtostrf((leftVelocity + rightVelocity)/2, 4, 2, cStrVel);
822       sprintf(myString, "E2|%s|%s", cStrDist, cStrVel);
823       Serial.println(myString);
824       lastSentGndNotDetected = millis();
825       lastSentMsg = lastSentGndNotDetected;
826     }
827   }
828   else if(objectDetected){
829     if((millis() - lastSentObstacle) >= 2500 && ((millis() -
          lastSentGndNotDetected) >= 2500)){
830       walkerOdometer();
831       dtostrf(odometerDist, 4, 2, cStrDist);
832       dtostrf((leftVelocity + rightVelocity)/2, 4, 2, cStrVel);
833       sprintf(myString, "E3|%s|%s", cStrDist, cStrVel);
834       Serial.println(myString);
835       lastSentObstacle = millis();
836       lastSentMsg = lastSentObstacle;
837     }
838   }
839   else if((millis() - lastSentMsg) >= 1000){
840     walkerOdometer();
841     dtostrf(odometerDist, 4, 2, cStrDist);
842     dtostrf((leftVelocity + rightVelocity)/2, 4, 2, cStrVel);
843     sprintf(myString, "E4|%s|%s", cStrDist, cStrVel);
844     Serial.println(myString);
845     lastSentMsg = millis();
846   }
847
848 }
```

A.7.2 Código fonte do módulo WiFi ESP8266 ESP-01

```
1
2  /*****************************************************************/
3  /*           ESP8266 - 2021/2022 - TESE DE MESTRADO              */
4  /*               Cristina Pierdevara - 2015239095                */
5  /*                  Ultima modificacao: 29Jul2022                */
6  /*****************************************************************/
7  #include "FS.h"
8  // Fornece rotinas Wi-Fi especificas do ESP8266 que se chamam para conetar a
       rede
9  #include <ESP8266WiFi.h>;
10 // Permite a um cliente criar mensagens simples de publicacao/assinatura com um
       servidor que suporta MQTT
11 #include <PubSubClient.h>;
12 #include <NTPClient.h>; // Obtem o horario de um servidor NTP e mantem-no
       sincronizado
13 #include <WiFiUdp.h>; // Necessario especificamente para programacao de rotinas
       UDP
14
15 // DEFINICAO DAS CONSTANTES
16 #define BUFFER_LEN 256
17 long now;
18 long lastMsg = 0;
19 char msg[BUFFER_LEN];
20 char SerialBuf[BUFFER_LEN];
21 int value = 0;
22 byte mac[6];
23 char mac_Id[18];
24 char distStr[18];
25 char velStr[18];
26
27 // CREDENCIAIS REDE DE INTERNET
28 const char* ssid = "Cris";
29 const char* password = "05071996";
30
31 const int walker_Id = 1;
32
33 // Defina um cliente NTP para obter tempo
34 WiFiUDP ntpUDP;
35 NTPClient timeClient(ntpUDP, "pool.ntp.org");
36
37 const char* AWS_endpoint = "a2zvj9qd535xrn-ats.iot.eu-west-1.amazonaws.com"; //
       MQTT broker IP
38 WiFiClientSecure espClient;
39 void callback(char* topic, byte* payload, unsigned int length);
40 PubSubClient client(AWS_endpoint, 8883, callback, espClient); // MQTT porta 8883
41
42 /*================================ SETUP ================================*/
43 void setup() {
44
45   Serial.begin(115200);
46   // Inicializa o pino digital LED_BUILTIN como uma saida
47   pinMode(LED_BUILTIN, OUTPUT);
48   setup_wifi();
49   delay(1000);
50   if (!SPIFFS.begin()){
51     Serial.println("Falha ao montar o sistema de arquivos");
52     return;
53   }
54
55   // Carrega certificados
56   File cert = SPIFFS.open("/cert.der", "r");
57
58   if (!cert) Serial.println("Falha ao abrir o arquivo CERT");
59   else  Serial.println("CERT lido com sucesso");
60
61   delay(1000);
62
```

```
63    if (espClient.loadCertificate(cert)) Serial.println("CERT carregado");
64    else Serial.println("CERT nao foi carregado");
65
66    // Carrega arquivo com chave privada
67    File private_key = SPIFFS.open("/private.der", "r");
68    if (!private_key) Serial.println("Falha ao abrir o arquivo da chave privada");
69    else Serial.println("Chave privada lida com sucesso");
70
71    delay(1000);
72
73    if (espClient.loadPrivateKey(private_key)) Serial.println("Chave privada
          carregada");
74    else Serial.println("Chave privada nao foi carregada");
75
76    // Carrega arquivo CA
77    File ca = SPIFFS.open("/ca.der", "r");
78    if (!ca) Serial.println("Falha a abrir o arquivo CA");
79    else Serial.println("CA lido com sucesso");
80
81    delay(1000);
82
83    if(espClient.loadCACert(ca)) Serial.println("CA carregado");
84    else Serial.println("Falha ao carregar CA");
85
86    WiFi.macAddress(mac);
87    snprintf(mac_Id, sizeof(mac_Id), "%02x:%02x:%02x:%02x:%02x:%02x",
88    mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
89    Serial.print(mac_Id);
90
91  }
92
93  /*================================ LOOP ================================*/
94  void loop() {
95
96    if (!client.connected()) reconnect();
97
98    client.loop();
99
100   char event_Id = '6';
101
102   now = millis();
103
104   if(Serial.available()){
105     char Buf = Serial.read();
106     if(Buf == 'E'){
107       int k = 0;
108       while(!Serial.available());
109       event_Id = Serial.read();
110       while(!Serial.available());
111       Serial.read();
112       Buf= 'a';
113       while(Buf != '|' && k < 10){
114         while(!Serial.available());
115         Buf = Serial.read();
116         SerialBuf[k] = Buf;
117         k++;
118       }
119
120       SerialBuf[k-1] = '\0';
121       Serial.printf(SerialBuf);
122       strcpy(distStr, SerialBuf);
123       k = 0;
124       Buf= 'a';
125       while(Buf != '\n' && k < 10){
126
127         while(!Serial.available());
128         Buf = Serial.read();
129         SerialBuf[k] = Buf;
130         k++;
131       }
```

```
132         SerialBuf[k-1] = '\0';
133         Serial.printf(SerialBuf);
134         strcpy(velStr, SerialBuf);
135
136         snprintf(msg, BUFFER_LEN,"{\"walker_Id\" : \"%d\", \"event_Id\" : \"%c\",
                \"dist\" : \"%s\", \"velocity\" : \"%s\"}", walker_Id, event_Id,
                distStr, velStr);
137         Serial.print("Publicar mensagem: ");
138         Serial.println(msg);
139
140         client.publish("outTopic", msg);
141
142         digitalWrite(LED_BUILTIN, HIGH); // LED ON
143         delay(100); // wait for a 0.1s
144         digitalWrite(LED_BUILTIN, LOW); // LED OFF
145       }
146     }
147 }
148
149 /*============== MENSAGEM DE VOLTA Wi-Fi ==============*/
150 void callback(char* topic, byte* payload, unsigned int length) {
151   Serial.print("Mensagem recebida [");
152   Serial.print(topic);
153   Serial.print("] ");
154   for (int i = 0; i < length; i++) {
155     Serial.print((char) payload[i]);
156   }
157   Serial.println();
158 }
159
160 /*============== SETUP Wi-Fi ==============*/
161 void setup_wifi() {
162
163   delay(10);
164   // A conetar a uma rede Wi-Fi
165   espClient.setBufferSizes(512, 512);
166   Serial.println();
167   Serial.print("Conetar a ");
168   Serial.println(ssid);
169
170   WiFi.begin(ssid, password);
171
172   while (WiFi.status() != WL_CONNECTED){
173     delay(500);
174     Serial.print(".");
175   }
176
177   Serial.println("");
178   Serial.println("Wi-Fi conetado");
179   Serial.println("Endereco IP: ");
180   Serial.println(WiFi.localIP());
181
182   digitalWrite(LED_BUILTIN, HIGH); // LED ON
183   delay(100); // Esperar 0.1 s
184   digitalWrite(LED_BUILTIN, LOW); // LED ON
185   delay(100); // Esperar 0.1 s
186   digitalWrite(LED_BUILTIN, HIGH); // LED ON
187   delay(100); // Esperar 0.1 s
188   digitalWrite(LED_BUILTIN, LOW); // LED OFF
189
190   timeClient.begin();
191
192   while(!timeClient.update()){
193     timeClient.forceUpdate();
194   }
195
196   espClient.setX509Time(timeClient.getEpochTime());
197
198 }
199
```

```
200 /*============== WiFi RECONETANDO ==============*/
201 void reconnect(){
202   // Loop ate reconetar
203   while (!client.connected()) {
204     Serial.print(" Tentando conexao MQTT...");
205     // Tentativa de conexao
206     if (client.connect("ESP8266_thing")){
207     Serial.println("Conetado");
208     digitalWrite(LED_BUILTIN, LOW);
209     // Uma vez conectado, publicar
210     // client.publish("outTopic", "Hello world");
211     // Subscrever novamente
212     client.subscribe("inTopic");
213     }
214     else{
215     Serial.print("Falhou, rc=");
216     Serial.print(client.state());
217     Serial.println(" Tente novamente em 5 segundos");
218     char buf[256];
219     espClient.getLastSSLError(buf, 256);
220     Serial.print("Erro WiFiClientSecure SSL: ");
221     Serial.println(buf);
222     // Aguarde 5 segundos antes de tentar novamente
223     delay(5000);
224     }
225   }
226 }
```