Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Ricardo Filipe Carvalho Amaral Abrantes

SmartShoes

Relatório de projeto

Projeto orientado por:
Prof. Doutor Gabriel Pereira Pires – Instituto Politécnico de Tomar

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

If you can dream it
you can do it

Walt Disney

RESUMO

Nos hospitais portugueses todos os dias morrem, em média, 11 idosos por causa de ferimentos originados em quedas. Este problema torna-se ainda mais grave quando percebemos que a população idosa tem uma tendência de crescimento positivo e que cada vez mais idosos procuram viver de forma independente até mais tarde. Assim, é fundamental a criação de tecnologia de assistência que possa contribuir para um rápido atendimento médico aquando de eventos de queda.

Neste contexto, foi desenvolvido um protótipo de sapatos inteligentes, o Smartshoes, que integra um sistema de deteção automático e aviso de queda. O sistema monitoriza a atividade do utilizador através de sensores inerciais e sensores de força e, em caso de queda, poderá alertar as entidades de socorro ou as pessoas previamente definidas.

O projeto destaca-se pelo baixo custo, número de funcionalidades e elevada autonomia, o que o torna bastante competitivo relativamente a sistemas similares existentes atualmente. A deteção de queda baseia-se em regras heurísticas e na compatibilidade entre atividades detetadas no movimento anterior e no movimento atual.

O sistema de processamento e comunicação assenta num módulo ESP32, responsável pela leitura de todos os sensores e na execução do algoritmo de deteção de quedas. O uso tecnologias muito recentes como é o caso da comunicação LoRa permitiu que fossem feitas comunicações com uma reduzida quantidade de energia. Este tipo de comunicações estará presente na vida de todas as pessoas dentro de poucos anos, pois é uma das tecnologias com grande crescimento na comunicação da Internet das Coisas (IoT).

O sistema foi validado experimentalmente apenas por um participante em ambiente simulado, mas os resultados são promissores indo ao encontro das funcionalidades e especificações inicialmente definidas. No entanto, para uma efetiva validação dos SmartShoes, o sistema terá de ser validado por um maior grupo de pessoas, incluindo idosos.

Palavras-chave: sapatos, deteção queda, sos, idosos, esp32, sensores inerciais, sapatos inteligentes.

ABSTRACT

Everyday, on average, 11 elderly citizens die in Portuguese hospitals, victims of fall injuries. This problem becomes more serious when we realize that aging rate is increasing in Portugal and that elderly people aim to be independent and live alone for many years. Thus, the development of technologies contributing to fast medical cares, in case of fall, is crucial.

In this context, a prototype of intelligent shoes – named Smartshoes – was developed, including an automatic fall detection and warning system. This system monitors user's activity through inertial and force sensors and, in case of fall, may send an alert to assistance entities and pre-determined people.

This project differs by its low-cost, the number of functionalities and a big autonomy, making it very competitive when compared to similar existing systems. Such fall detection is based on heuristic rules and on the compatibility between the previous and the current movement.

The processing and communication system bases on an ESP32 module, responsible for reading all sensors and for executing the fall detection algorithm. Using very recent technologies, such as LoRa communication, it was possible to communicate with little energy. Such communication type will be present in every person life soon, because it is one of the high grow technologies used in the Internet of Things (IoT).

System validation occurred experimentally by only one participant in a simulated environment, but the observed results are relevant, meeting the pre-defined features and specifications. However, for an effective validation of the SmartShoes, the systems must be validated by a higher group of participants including elderly people.

Keywords: shoes, fall detection, sos, elderly people, esp32, inertial sensors, smart shoes.

AGRADECIMENTOS

Agradeço a todos os que me apoiaram neste percurso, em particular à minha família, que se privou de muitos momentos em conjunto para que este objetivo fosse alcançado.

Um obrigado muito sentido ao professor Gabriel Pires, que apesar da sua agenda muito preenchida, me deu a honra de definir e orientar este projeto.

Agradeço, também, à Silicália, S.A. pela flexibilidade que me permitiu conciliar o trabalho com esta etapa académica.

E, por último, dirijo um agradecimento especial à minha namorada, Cristina, por todo o apoio, incentivo, amizade e paciência demonstrada durante esta caminhada.

Este trabalho teve o suporte financeiro do projeto IC&DT VITASENIOR-MT CENTRO-01-0145-FEDER-023659 com fundos do FEDER através dos programas operacionais CENTRO2020 e FCT.

Muito obrigado!

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE GRÁFICOS

LISTA DE ACRÓNIMOS

IMU - Inertial Measurement Units

IoT - Internet of Things

ULP – Ultra Low-power Processor

LoRa – Long Range

LPWAN - Low Power Wide Area Network

RF – Rádio Frequência

ASM – Linguagem Assembly

FSR – Force-sensing resistor

CPU – Central Processing Unit

RTC – Real Time Clock

SAR – Successive Approximation Register

ADC – Analog-to-Digital Converter

NTP – Network Time Protocol

PCB – Printed Circuit Board

1. Introdução

1.1. Motivação e Contexto

Na população idosa existe um decaimento acentuado das capacidades visuais, auditivas, motoras, de perceção do meio envolvente, e tempos de reação. Todos estes fatores aumentam consideravelmente o risco de quedas, as quais são responsáveis por muitas mortes de idosos todos os anos, ou levam a imobilidade prolongada e incapacidade crónica com consequente perda de qualidade de vida.

A par da criação de ambientes seguros e da sensibilização para adoção de estilos de vida saudáveis, é também necessário desenvolver tecnologias que permitam detetar quedas e gerar alertas automaticamente, garantindo um socorro atempado das vítimas.

1.2. Objetivos

Com este projeto pretende-se dotar um par de sapatos com um sistema de deteção de queda e pedido automático de ajuda. Os sapatos são concebidos de forma a poderem conter no interior da sola todos os sistemas necessários à deteção, análise e comunicação da queda, garantindo assim a usabilidade normal do sapato. Por terem estas características optou-se por os designar “Smartshoes”.

Atualmente os sistemas de deteção de queda são ainda pouco fiáveis e por isso o seu uso ainda é pouco comum, no entanto, a evolução tem sido constante e atualmente existem soluções baseadas em botões de alarme, aplicações em *smartwatchs* e *smartphones*, sapatos inteligentes, ou uma conjugação de vários sensores. Com este projeto, pretende-se avaliar a fiabilidade e usabilidade de um sistema de deteção de quedas integrado num par de sapatos, procurando assim contribuir com novas soluções e abordagens dentro desta área de investigação.

Para garantir a usabilidade do Smartshoes definiram-se as seguintes especificações:

- Autonomia de 5 dias;
- Peso total do *hardware* de 50 gramas em cada sapato;
- Frequência de amostragem de 20Hz
- Comunicação com *software* para mostrar distâncias percorridas, calorias gastas, representações gráficas, etc.

1.3. Contribuições e trabalho realizado

A deteção e aviso de queda serão os pontos fulcrais abordados em todo o trabalho, bem como a usabilidade de todo o sistema. Os contributos do trabalho consistem no desenvolvimento dos seguintes módulos (ver arquitetura geral na Figura 16):

- Módulo de identificação de tipo de movimento de marcha, corrida, gatinhar, etc.;
- Comunicação entre sapatos por rádio frequência de baixo consumo;
- Sistema de gestão de bateria;
- Comunicação com *gateway* para envio de informação de queda;
- Comunicação de queda aos serviços de socorro através de uma aplicação;
- Base de carregamento da bateria e comunicação de informação.
- Etiquetagem temporal de todos os movimentos a fim de poder fazer o registo de toda a atividade e esta poder ser analisada posteriormente.

Para que todo o sistema seja funcional, é necessário agregar e integrar um microprocessador, um acelerómetro, um emissor e recetor de radio frequência, sensores de força, baterias e o respetivo circuito de carga e descarga.

Ainda é necessário garantir que a gestão de energia é eficaz e que todos os módulos são otimizados. A carga das baterias é feita através de uma sapateira especialmente concebida dotada de um carregador sem fios e que será o local onde o Smartshoes se vai conectar à Internet para descarregar todos os dados de atividade e fazer o sincronismo temporal.

A figura seguinte exemplifica todo o processo.

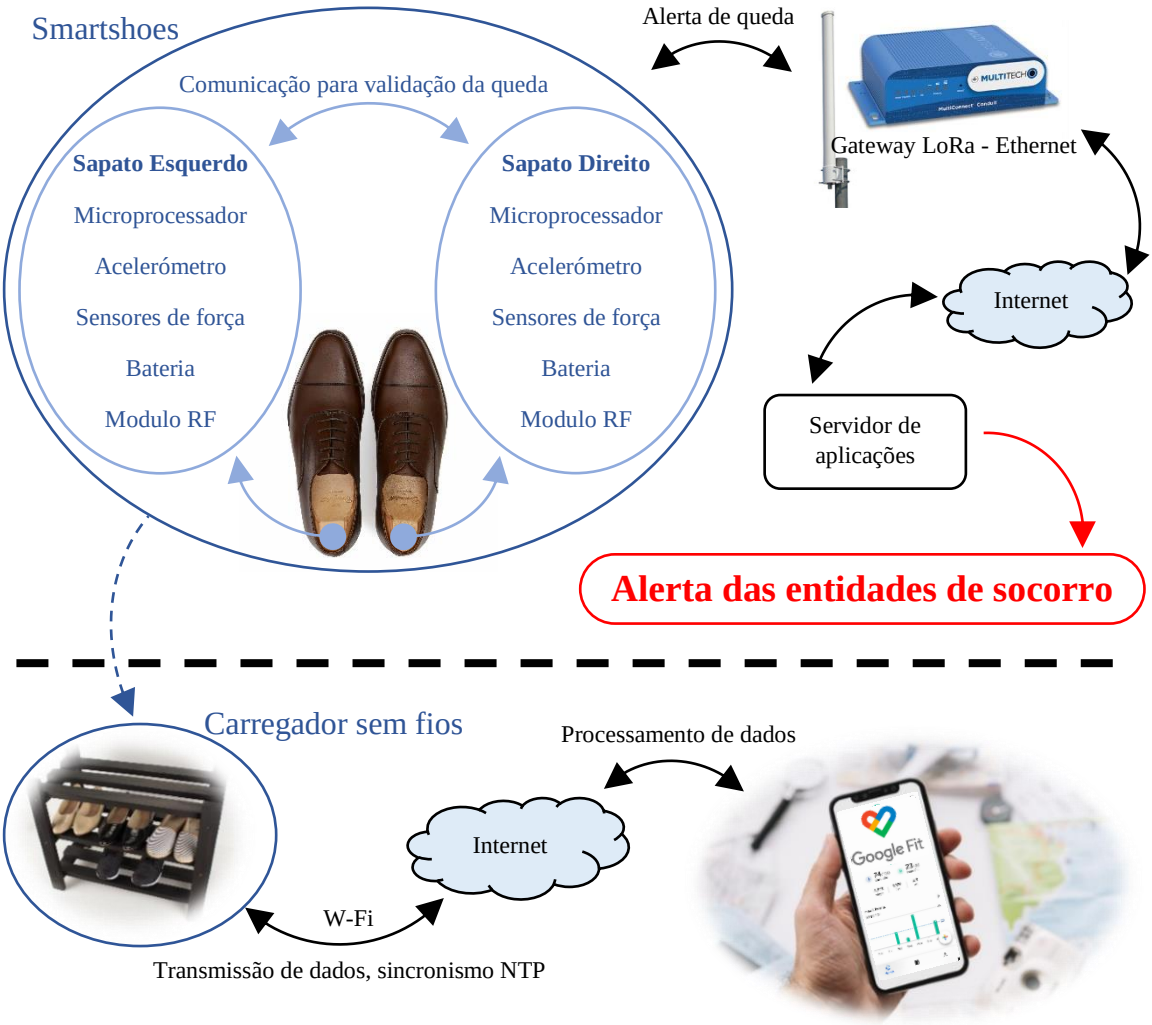


Figura 1 - Diagrama funcional de todo o sistema

1.4. Organização do relatório

O relatório encontra-se dividido em 6 capítulos.

- A Introdução onde é apresentado o projeto desenvolvido bem como o seu enquadramento e público alvo.
- O capítulo 2 com o estado da arte, onde se posiciona este projeto relativamente a outros sistemas de deteção de queda presentes no mercado e a outros projetos académicos.

- As tecnologias de software e hardware usadas são abordadas no terceiro capítulo.
- O quarto capítulo descreve a arquitetura de Hardware e Software , bem como a sua interação, características, funcionalidades e principais problemas encontrados.
- Os resultados de validação do sistema proposto são descritos no quinto capítulo.
- Por fim, nas conclusões e oportunidades de melhoria contam as apreciações finais e todos os pontos onde será possível intervir futuramente para que este projeto possa ser melhorado.

2. Estado da Arte

2.1. Objetivos esperados e análise de alternativas

Com a tendência para o decrescimento da população em Portugal e com o aumento da esperança média de vida [1], o envelhecimento demográfico é uma realidade que terá de ser tida em conta na criação de ambientes mais seguros, em especial para a população idosa.

No final de 2017 o Serviço Nacional de Saúde (SNS) lançou um livro intitulado "Tropeções, quedas e trambolhões" [2] onde refere que as quedas são o acidente mais frequente em casa e a principal causa de morte acidental dos idosos (pessoas com mais de 65 anos).

Os números apresentados no mesmo livro revelam que 3% dos internamentos de pessoas com mais de 65 anos em Portugal tiveram origem em quedas e cada internamento teve uma duração média de 13 dias. Das pessoas internadas 6% morrem ainda no hospital.

Extrapolando as estatísticas para o contexto atual, e segundo dados do INE de 2018 [1], se em Portugal existirem atualmente 2.244.225 idosos (como em 2018), durante o ano corrente 67.327 serão internadas por motivo de queda e destes, 4.040 acabarão por falecer no hospital.

A deteção automática de quedas pode permitir a atuação atempada das entidades de emergência, prevenindo consequências mais gravosas ou mesmo salvar a vida da vítima. No mercado já existem produtos que detetam quedas e as comunicam a centros de informação ou aos serviços de emergência. No entanto, todos eles apresentam debilidades que os levam a serem pouco usados ou até mesmo ignorados.

Na deteção de queda, o grande problema a ultrapassar são os falsos alarmes, ou seja, alarmes de queda que são gerados por erros de leitura dos sensores ou por movimentos equiparados a quedas. Este problema leva a que os avisos sejam ignorados pelas entidades ou cuidadores responsáveis, ou simplesmente desligados pelos próprios utilizadores, levando que alarmes verdadeiros não sejam detetados.

Atualmente existem vários tipos de sistemas de deteção de quedas, alguns com um raio de ação mais limitado como é o caso dos sistemas que fazem uso de camaras e processamento de vídeo em tempo real ou os pisos dotados de sensores de vibração. No entanto, este trabalho é direcionado para sistemas que possam ser usados dentro e fora de ambientes domésticos. Assim, os sistemas de deteção de queda serão divididos em 3 grupos:

- *Smartwatches*, pulseiras ou *smartphones*;
- Botões de pânico;
- Sapatos inteligentes.

Nos *smartwatch*, pulseiras ou *smartphones* a fiabilidade dos avisos é muito comprometida porque as quedas são detetadas apenas pela informação dada pelo acelerómetro. Assim, no caso de um utilizador ir dentro de um veículo numa zona de piso muito irregular, provoca movimentos anómalos nos dispositivos que originam falsos alarmes de queda. No entanto, este tipo de dispositivos são os mais usados porque não são necessários equipamentos extra além dos que já se usam no dia-a-dia, basta instalar aplicações desenvolvidas para o efeito (Figura 2) e usar o dispositivo normalmente.

Figura 2 - App de deteção de quedas [28]

Um sistema alternativo são os colares ou botões de pânico em que, quando a queda é detetada o colar soa um aviso sonoro durante algum tempo e, caso não haja desativação do alarme, a queda é confirmada e as entidades de emergência alertadas [3]. Neste caso, as quedas são comunicadas por GSM ou por uma rede sem fios de baixa frequência até uma central que posteriormente desencadeia o alarme.

Figura 3 - Sistemas de deteção de queda baseados em botão de alarme e colar [3]

Mais recentemente têm surgido abordagens em que a deteção da queda é feita através dos sapatos. No mercado existem muitos sapatos inteligentes, mas vocacionados para o desporto em que a deteção de queda nem sequer é abordada nas especificações.

A opção da deteção de queda através dos sapatos aparenta ser a mais vantajosa visto que os sapatos são acessórios que as pessoas usam naturalmente no seu dia-a-dia, não necessitando de ter consigo componentes extra como os *smartwatches* ou botões de alarme. Alem desta vantagem, os sapatos são usados aos pares, o que permite uma avaliação individual dos sensores e apenas lançar o alarme no caso de ambos detetarem a queda.

2.1. Sistemas de deteção de queda baseados em *smartwatches*, pulseiras e *smartphones*

Os sistemas de deteção de quedas baseados em *smartwatches*, *smartphones* e pulseiras baseiam-se em aplicações que leem constantemente os sensores inerciais, microfone e sensores de batimento cardíaco. Durante a aquisição dos dados, caso sejam detetadas acelerações anómalas no acelerómetro, sons similares aos emitidos em caso de aflição, o batimento cardíaco acelerar e não houver atividade durante os segundos seguintes, é feita uma chamada de emergência para os contactos predefinidos. Este é o princípio de funcionamento da deteção e queda do Apple Watch [4] ou o *incident detection* dos Garmin Watches [5].

Não é possível comparar a eficiência de cada uma das aplicações, pois todas prometem 100% de eficácia na deteção de quedas. Atualmente, a aplicação Android que reúne melhores críticas entre os utilizadores é a "Safety App - SOS, PERSONAL, WOMEN & EMERGENCY APP" (Figura 4). Esta aplicação, alem de detetar e alertar quedas, ainda permite alertar os contactos via SMS, partilhar a localização GPS e encontrar lugares públicos próximos para pedir ajuda.

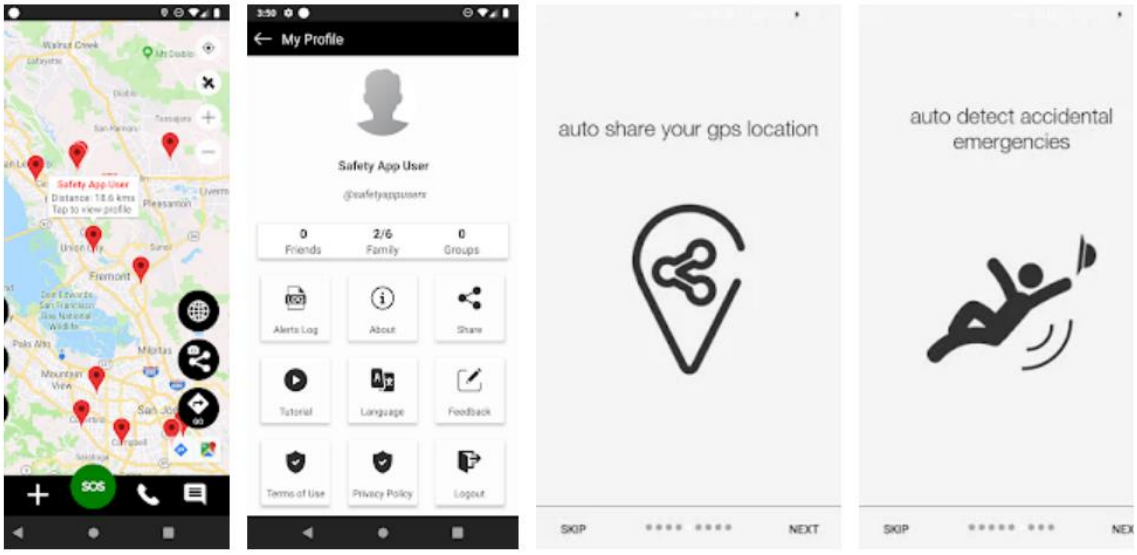


Figura 4 - Safety App - SOS, PERSONAL, WOMEN & EMERGENCY APP

2.2. Sistemas de deteção de queda baseados em botões de pânico

Existem várias alternativas comerciais para a deteção de queda baseadas em botões de pânico, dos quais se apresentam alguns exemplos tais como o Medical Care Alert [6], Life Call [7], Medical Guardian [8] e Alert-1 [3] que se baseiam no sistema de botão de aviso e central telefónica que comunica a queda aos serviços 24h da própria empresa. Em Portugal a A48 dispõe de um sistema (GLOBALHELP1) que incorpora um botão de alarme com comunicação GSM, localização GPS e uma linha de atendimento 24h para falar com o utilizador em caso de queda [9]. Estas soluções envolvem custos mensais para utilização do serviço.

A tabela a baixo posiciona este projeto com os produtos descritos:

	SmartShoes (PREVISTO)	A48 - GLOBALHELP 1	medicalcarealert	lifecall	medicalguardian	alert-1
Deteção de queda com mais de 80% de certeza	●	●				
Deteção de queda com 80% de certeza			●			
Deteção de queda com 75% de certeza				●	●	●
Comunicação RF com a central e GSM com exterior		●	●	●	●	●
Comunicação Wi-fi, Bluetooth ou LoRa	●					
Duração da bateria superior a 5 dias	●					
Custos mensais de manutenção do serviço		●	●	●	●	●

Tabela 1 - Comparação de vários sistemas de deteção e aviso de queda presentes no mercado

Apesar de serem produtos comercializados e de garantirem bons resultados, existem fóruns que expõem a insatisfação dos seus utilizadores. Este ponto negativo deverá será o primeiro a ultrapassar para que os sistemas de aviso de queda possam ser amplamente usados.

2.3. Sistemas de deteção de queda baseados em sapatos inteligentes

O primeiro sapato inteligente foi produzido pela Puma em 1986 [10]. Na época era usado para contar o número de passos e calorias gastas. Desde então os sapatos dotados de sistemas computacionais têm vindo a evoluir e atualmente são de uma grande importância na alta competição desportiva.

Os sapatos usados na competição não estão otimizados para a deteção de queda, no entanto poderão ser posicionados relativamente a este projeto por serem sapatos inteligentes.

A E-vone [11] é uma empresa francesa que apresentou em 2018 os seus sapatos com sistema de deteção de queda e pedido de auxílio. Sendo estes os que dispõem de mais informação online.

Na tabela a baixo são comparados 10 sapatos de treino com o previsto para o SmartShoes e com os sapatos da E-vone. Esta comparação tem por base o artigo publicado no site GadgetsAndWearables.com [12] onde são descritos os mais recentes produtos de *Tracking fitness*.

	SmartShoes (PREVISTO)	Digitsole Warm Series	Xiaomi Smart Shoes	Under Armour Speedform	MiCoach Speed Cell	Runscribe	Altra IQ Sports Shoes	Lechal Smart Insoles	Milestone Pod	Garmin Foot Pod	Stryd, Power Meter for Running	***E-vone***
Detecção de queda	●											●
Analise de força de impacto	●	●	●	●	●	●	●		●	●	●	●
Contagem de passos	●	●	●	●	●	●	●	●	●	●	●	●
Integração em software de treino				●	●	●				●	●	
Detecção de tipos de movimento	●		●	●			●	●	●	●	●	
Conectividade Wi-fi	●											
Conectividade bluetooth	●	●	●	●	●	●	●	●	●	●	●	●
Comunicação GSM	●											●
Integração com Google Fit	●		●					●				
Comunicação com smartphone		●	●	●	●	●	●	●	●	●	●	
Carregamento por indução	●											
Bateria substituível		●	●	●	●	●		●	●	●		
Duração de bateria > 5 dias	●		●	●				●		●		●
Regulação de temperatura		●										
Peso inferior a 125 gramas	●			●		●	●		●		●	●
Tracking system		●		●	●	●	●	●			●	
Resistente à água (IP67)		●	●	●	●	●	●	●		●	●	●
Adornos decorativos		●	●									
Sapato completo	●	●	●	●			●	●	●		●	●

Tabela 2- Comparação dos vários sapatos inteligentes presentes no mercado

Pela tabela é possível perceber que todos os sapatos comunicam com *smartphone* e que podem ser integrados com programas de treino.

No caso dos E-vone, a deteção de queda é o ponto fulcral pelo que, as características de integração com outros softwares não está presente. No entanto é a este produto que o Smartshoes se pode comprar diretamente.

3. Tecnologias de Hardware e Software

Neste capítulo apresentam-se as tecnologias que foram utilizadas ou foram equacionadas como potencial utilização. Aquando da definição do projeto os microcontroladores apontados para serem inseridos neste projeto seriam o Sam D21 da Atmel no seu modo de baixo consumo conjuntamente com o ESP8266 para se poder ligar à rede wi-fi ou o ESP32. Este último foi o eleito por agregar todas as funcionalidades dos outros 2 e apresentar consumos mais baixos.

A nível de comunicação entre sapatos a escolha imediata foi para os circuitos RFM95 de modulação em frequência e a utilização do protocolo LoRa. Esta escolha deve-se ao facto de ser uma tecnologia em grande ascensão, de código aberto e com uma crescente comunidade. A escolha do RFM95 em detrimento dos restantes RFM96 e 97 deve-se apenas à largura de banda livre na europa estar nos 868MHz.

No que diz respeito ao acelerómetro escolhido, a escolha recaiu no TDK MPU9250 por este ser um IMU de 9 graus de liberdade. Apesar de não serem necessárias todas as funcionalidades do mesmo para concluir este projeto, as mesmas ficam disponíveis para que, facilmente, se possam integrar novas funcionalidades em projetos futuros.

3.1. Microcontrolador – ESP32

3.1.1. ESP32

O módulo ESP32 [13] é o membro mais recente de uma serie de microcontroladores de baixo consumo e baixo custo que junta no mesmo chip um microprocessador dual-core; um co-processador de baixo consumo; Wi-Fi; Bluetooth e muitas outras funcionalidades, fazendo dele a melhor relação desempenho/custo do mercado.

Desenvolvido pela chinesa *Espressif Systems* é o sucessor do ESP8266, já com muita aceitação no mercado mundial. Usa uma tecnologia de 40nm e têm disponível 42 portas I/O para serem usadas com grande flexibilidade (Figura 5).

Uma das novidades relativamente ao ESP8266 foi a introdução de um co-processador (ULP) de baixo consumo que pode manipular entradas e saídas, ler entradas analógicas, acordar o processador principal, manipular memórias, entre outras funcionalidades. O seu baixo consumo (desde 5uA) permite fazer sistemas autónomos com uma ótima gestão da bateria.

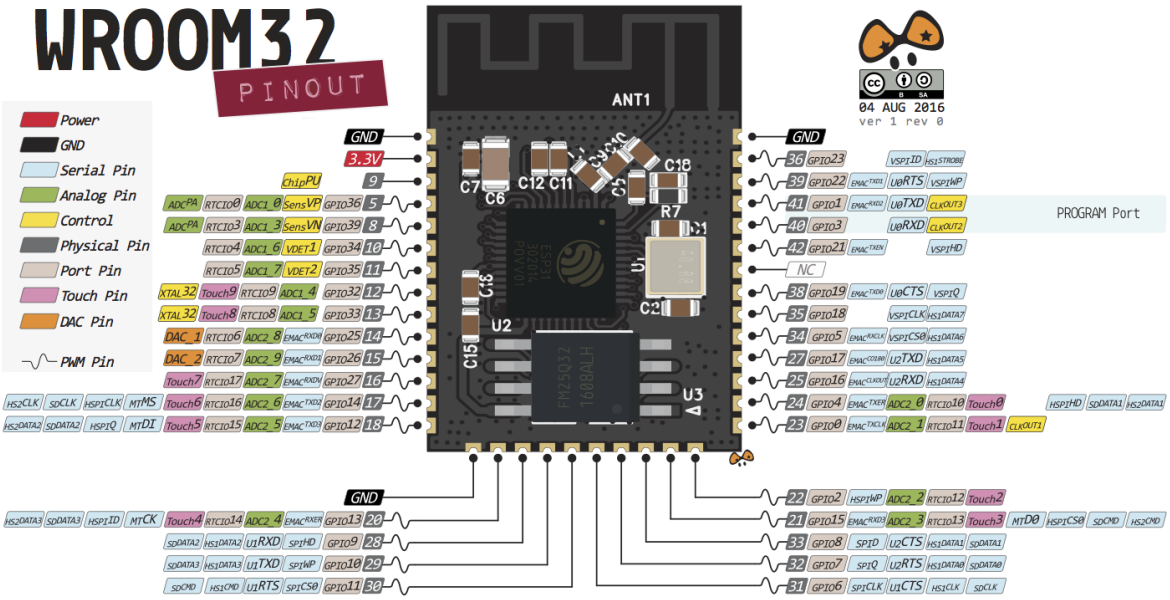


Figura 5 - ESP32 WROOM 32 Pinout

O ULP usa uma região de memória designada por RTC_SLOW_MEM de apenas 8kB onde é armazenado todo o código ASM, variáveis locais e variáveis globais partilhadas com o processador principal. Contém, ainda, 4 registos de 16 bits para uso geral e um registo de 8bits para contagens.

Este co-processador consegue ser rápido e eficiente em relação à memória e ao consumo mas ao mesmo tempo difícil de programar, pois todo o código tem de ser desenvolvido e carregado em *Assembly*.

Este código pode ainda ser escrito com o recurso a macros e carregado a partir do processador principal [14], sendo esta a melhor opção de baixo consumo a usar neste projeto.

Atualmente ainda não existe um compilador completo de código C para *Assembly* para este compilador, o compilador está a ser desenvolvido pela comunidade online de utilizadores dos ESP32 mas ainda se encontra em fase embrionária [15].

A *Espressif Systems* destaca como principais características do seu ULP os seguintes pontos:

- Monitorizar sensores digitais ou analógicos em *Deep Sleep* oferecendo um grande incremento no tempo de duração dos circuitos alimentados por baterias.
- Auxílio no processador principal, caso seja necessário um poder de processamento maior.
- Efetuar tarefas secundárias, como manipulação de GPIO's, sensores, criação de interrupções, etc, enquanto os *Cores* principais fazem tarefas mais pesadas ou *Time Sensitive*.

3.1.2. Características e interfaces

Segundo as especificações dadas pela *Espressif Systems*, o ESP32 é internamente resumido pelo diagrama funcional da figura seguinte:

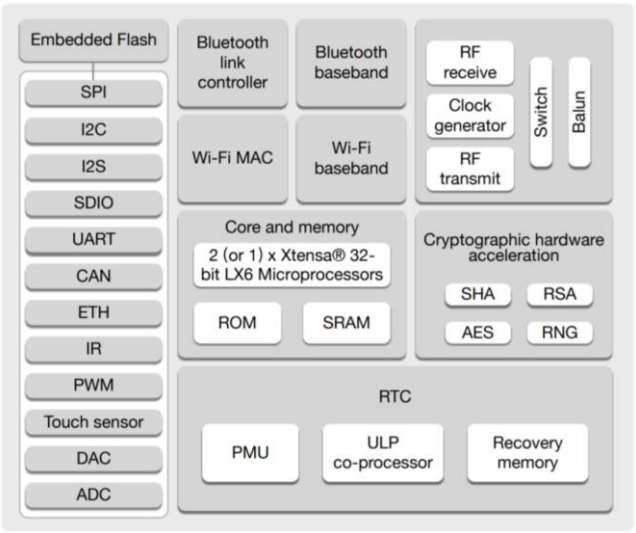


Figura 6 - ESP32: Diagrama funcional [13]

Pelo diagrama e especificações fornecidos pelo fabricante, destacam-se aqueles que são usados neste projeto:

CPU e Memória

- Xtensa® dual-core 32-bit LX6 microprocessor – processador responsável por todo o processamento de informação vinda dos sensores;
- 16 KB RTC SRAM – Memória compartilhada pelo processador principal e pelo ULP. Esta memória é importante para guardar informação enquanto o processador principal está adormecido

Clocks e Temporizadores

- Cristal oscilador externo de 32 kHz para o RTC com calibração – Oscilador interno usado pelo ULP.
- 2 temporizadores 64-bit – Um dos quais é responsável pela ativação da função de recolha e processamento de dados.
- Temporizador RTC – Usado na temporização da ativação do ULP

Interfaces periféricos

- 34 × entradas/saídas programáveis;
- 18 × 12-bit SAR ADC – Dos quais, 4 usados para leitura dos sensores de pressão e um para ler a tensão da bateria;
- 4 × SPI – Usado na comunicação com o RFM95, MPU9250 e Memória Flash;
- 2 × I²C – O barramento I2C não é usado no presente projeto, no entanto, está acessível através do conector para futuras necessidades.
- IR (TX/RX) – Usado na programação do microprocessador;

O esquema seguinte ilustra a disposição dos pinos do ESP32. Este circuito integrado está alojado num encapsulamento *Quad-Flat No-leads* (QFN) de 49 pinos e dimensões 6 x 6mm.

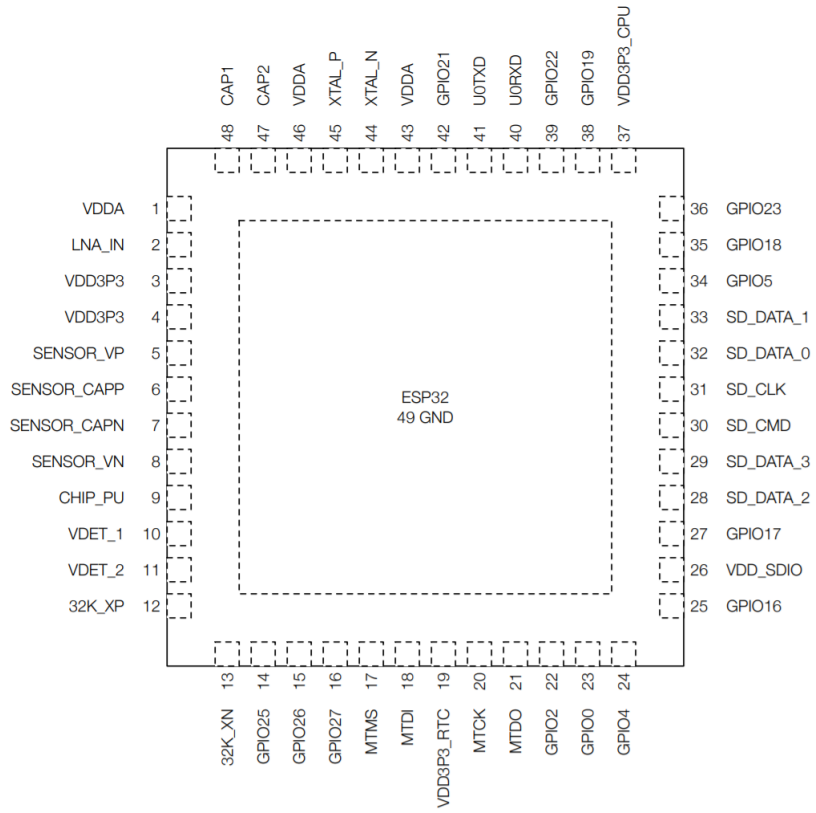


Figura 7 - ESP32: Pinout [13]

3.2. Periféricos I/O

3.2.1. MPU9250

O módulo de medição inercial MPU9250 desenvolvido pela *InvenSense*, uma subsidiária da TDK, contém um sensor MEMS de 9 eixos (9 DoF) que combina no mesmo pacote (SiP) dois chips individuais: um MPU-6500 responsável pelos 3 eixos do acelerómetro e pelos 3 eixos do giroscópio; e um AK8963 tem contem um magnetómetro de 3 eixos.

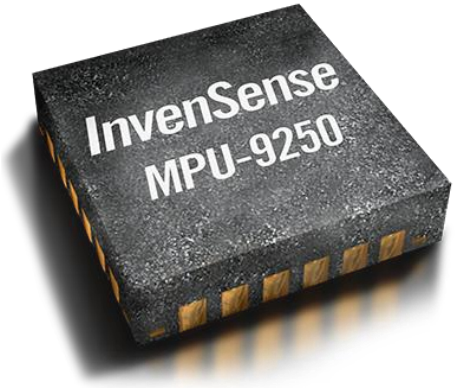


Figura 8 - MPU9250 (3mm x 3mm x1mm) [16]

Além da informação inercial, o MPU9250 ainda possui um termómetro interno, um conversor analógico-digital de 16 bits e portas de comunicação SPI e I2C (mestre ou escravo).

Neste projeto não são usadas todas as funcionalidades do IMU, apenas o acelerómetro, o que permite uma redução muito grande no consumo de energia (da ordem 450uA). Estas opções em conjunto com o despertar através de SPI permitem manter adormecido o MPU9250 adormecido grande parte do tempo, que fazem dele uma boa escolha para este projeto.

Já no decorrer do projeto, a *InvenSense* lançou o sucessor do MPU9250, o ICM-20948, com várias melhorias, uma delas no consumo do acelerómetro: 68.9uA. Este será uma boa escolha, caso os *Smartshoes* se comercializem.

3.2.2. LoRa

LoRa é a abreviatura para Long Range (grande distância) e é uma técnica de modulação de dados para transmissões sem fios a longa distância desenvolvida pela empresa *Semtech* e que faz parte do sistema de comunicação LPWAN. Erradamente chama-se LoRa a todo este sistema de comunicação.

É uma das opções consideradas em IoT (*Internet of Things*) estando a ter um crescimento muito grande um pouco por todo o mundo. Das suas vantagens destaca-se o baixo consumo de energia, o baixo custo (menos de 7 euros) e a transmissão de dados a longa distância (facilmente se transmitem dados a 20km) [17].

Todos os benefícios são obtidos à custa de uma pequena largura de banda e uma baixa capacidade de transmissão de dados que no caso do LoRa se fica pelos 5.5kbps [18], sendo no entanto perfeitamente suficiente para enviar informação de um sensor de temperatura a cada segundo mas insuficiente para transmitir música ou vídeo em tempo real.

Nem todos os dispositivos LoRa podem ser usados na Europa, existem limitações à sua utilização e apenas está autorizada a utilização da banda dos 863 aos 868 MHz com uma potência máxima de transmissão de 25mW conforme a norma sobre a harmonização do espetro de radiofrequências com vista à sua utilização por equipamentos de pequena potência e curto alcance da DECISÃO DE EXECUÇÃO (UE) 2017/1483 D [19].

A Figura 9 compara várias tecnologias de comunicação considerando a largura de banda vs. Alcance [18].

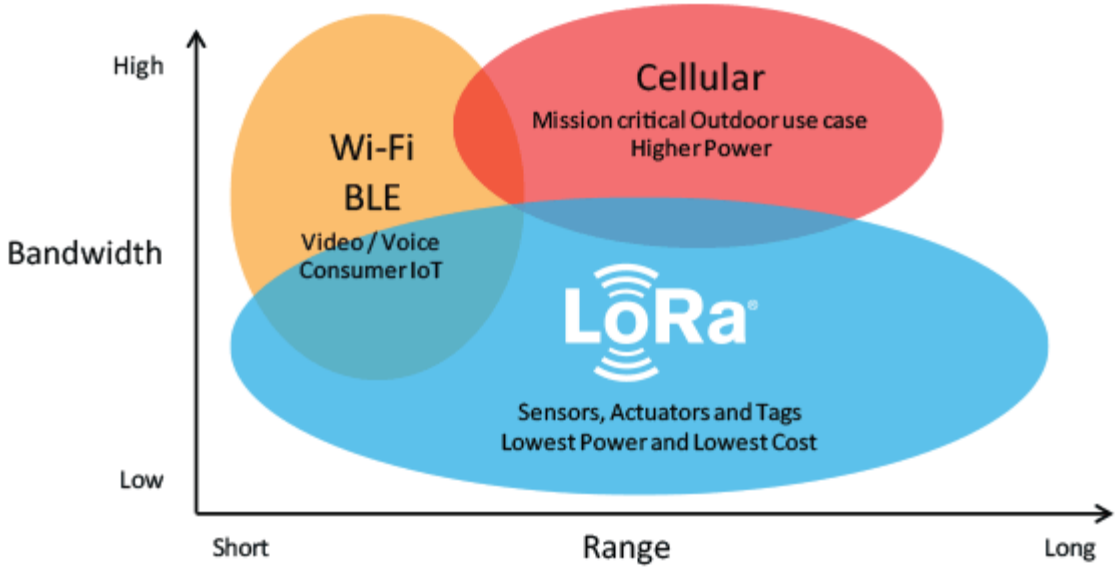


Figura 9 - Comparação da tecnologia LoRa com outras tecnologias RF, integradas no gráfico Largura de banda/Distância [18]

As previsões indicam que, atualmente, existem mais de 87 milhões de dispositivos LPWAN ligados a *gateways* em mais de 100 cidades [20].

LoRaWAN é um protocolo que determina os detalhes de segurança, funcionamento, qualidade do serviço e potência usando a comunicação LoRa. LoRa pertence à camada física de rede enquanto que LoRaWAN pertence à camada lógica.

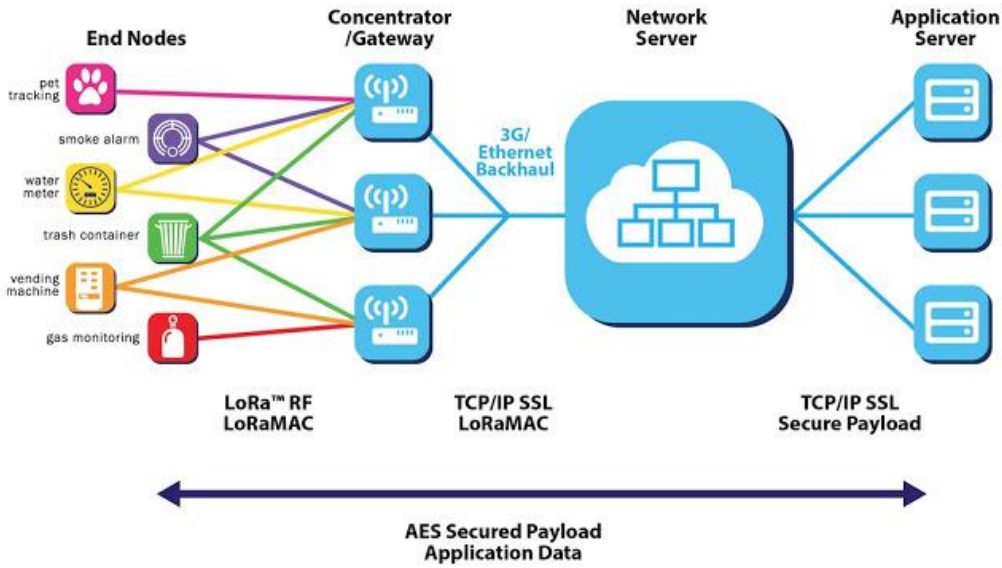


Figura 10 - Arquitetura da rede LoRaWAN [20]

Atualmente existem várias redes que permitem a criação de aplicações usando LoRaWAN, uma delas é a *The Things Network*. Uma rede que conta atualmente com mais de 77000 membros em 140 países e ligados através de mais de 7800 *gateways* [21]. Um dos *gateways* está localizado no laboratório FIKALAB da Critical Software em Tomar.

A utilização dos *gateways* e a *The Things Network* é gratuita e fornece os meios para que os Smartshoes possam comunicar quedas.

3.2.3. Gestão de bateria

A gestão de bateria é um dos pontos mais importantes deste e de outros projetos que necessitam de ser alimentados por baterias. No caso particular do *Smartshoes* é necessário garantir a melhor relação autonomia/peso da bateria.

As baterias de iões de lítio (Figura 11) são amplamente usadas em equipamentos eletrónicos pois são mais leves, carregam mais rápido, duram mais tempo e têm uma densidade energética maior quando comparadas com as baterias de Gel.

Figura 11 - Baterias de iões de lítio

O carregamento da bateria é também um tema fulcral no projeto tendo em conta que deverá ser feito regularmente e que deve ser algo simples. O simples gesto de ligar um cabo USB para carregar os sapatos pode não ser do agrado dos utilizadores, assim, o carregamento do Smartshoes será feito com recurso a carregadores sem fios (Figura 12).

Os sapatos terão uma sapateira especialmente concebida para serem colocados e que fará a carga da bateria. Os carregadores testados revelaram-se suficientes para carregar totalmente a bateria em 4 horas.

Para que seja possível garantir a carga, a distância entre emissor e recetor deverá ser a menor possível (cerca de 1 centímetro) e a alimentação do circuito transmissor deverá ser 12V.

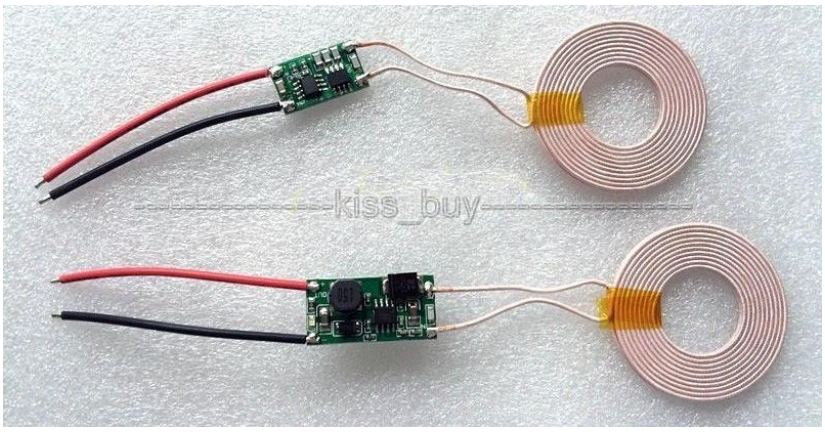


Figura 12 - Carregador sem fios (Rx e Tx)

A bateria de 3,7V é carregada a 4,2V e, para isso, é necessário um circuito de carga MCP73831 desenhado para que possa carregar baterias de uma célula de lítio e controlar as fases de carga bem como a velocidade. Este circuito encontra-se desligado durante a utilização dos sapatos fora da sapateira.

A tensão utilizada pelo controlador, IMU, memórias e sensores é 3.3V, logo, é necessário um conversor LDO para regular a tensão de saída. Os reguladores LDO são ideais para as aplicações eletrónicas de baixo consumo pois têm baixas quedas de tensão e aceitam uma ampla gama de tensões de entrada. O regulador de tensão usado é o AP2112-3.3.

O pino VDDIO do MPU9250 necessita de uma alimentação 1.8V. Assim, o circuito de alimentação terá também um LDO AP2112-1.8. O pino VDDIO do MPU9250 é tolerante a 3.3V. No entanto, este modelo de já foi descontinuado pela TDK e substituído pelo ICM-20948 que, fisicamente, é igual no antecessor, mas no referido pino só poderá receber tensões até 1.9V. Assim fica garantido que o hardware poderá suportar o upgrade do IMU.

3.2.4. Memória

O ESP32 possui uma EEPROM interna de 512bytes onde são guardados os dados da calibração do acelerómetro e a última data adquirida por NTP para que possa detetar se a data foi corretamente adquirida.

Possui também uma memória flash de 32Mbits (W25Q32JVSSIM) onde é guardada toda a informação que posteriormente será descarregada para o Google FIT.

3.2.5. Sensores de força

Os sensores de força foram selecionados por serem flexíveis, ultrafinos, com um consumo muito baixo e com respostas muito rápidas.

Figura 13 - Sensor de força (20mm x 5mm)

A constituição dos sensores baseia-se em materiais sensíveis e condutores separados por um espaçador não condutor e muito fino (Figura 14). Estes sensores são a evolução dos sensores ON/OFF de membrana. O que define estes sensores de força é sua característica de resistência dinâmica em relação à pressão aplicada sobre ele. Assim, quanto maior a pressão aplicada na superfície do sensor, menor será a sua resistência. Estes sensores de força são usados para medições qualitativas e não quantitativas ou de precisão.

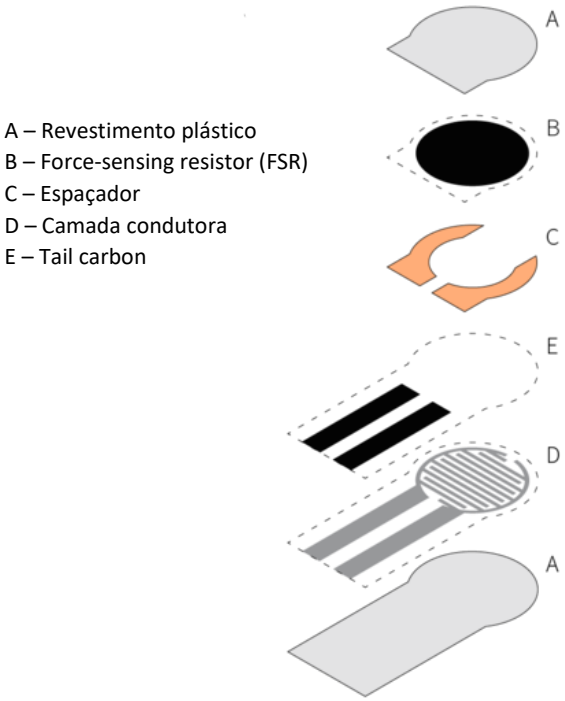


Figura 14 – Representação esquemática das camadas do sensor de força Tangio Printed Electronics [22]

Quando é exercida pressão no sensor, a resistência FSR varia, e consequentemente a tensão de saída também varia segundo o gráfico da Figura 15. A resistência FSR fica em serie com uma resistência de 3,9 ohms, sendo o ponto médio onde é medido a tensão de saída (Vout).

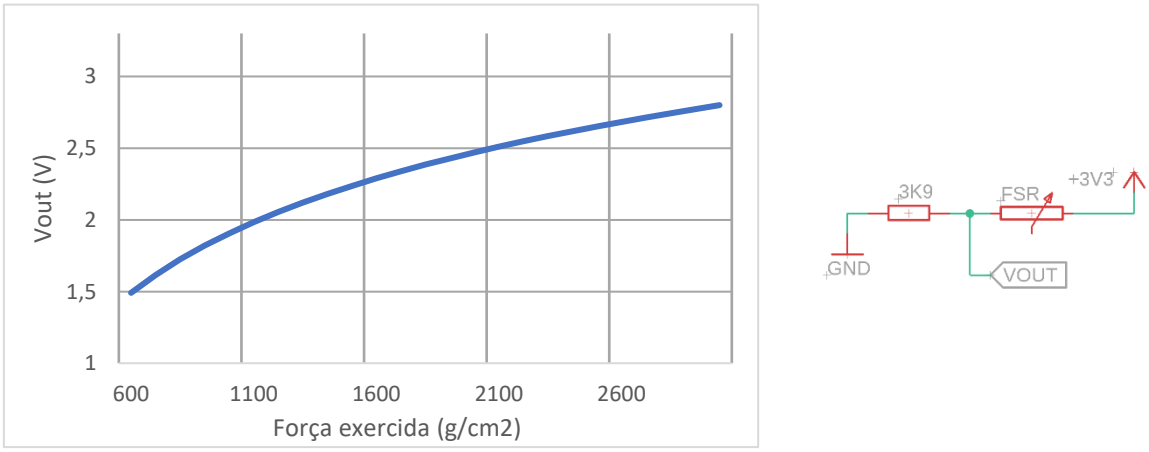


Figura 15 – Esquerda: Relação Vout e pressão exercida [23] ; Direita: Divisor de tensão usado

Os sensores de força usados têm como especificações:

- Modelo: RP-S5-ST
- Construção: Flexível
- Tipo de sinal de saída: Analógico
- Intervalo de pressão: 10g-1000g
- Diferença entre leituras: 10 g, Resistência <200 kΩ
- Espessura 0.25mm
- Resistência sem pressão: > 1MΩ
- Tempo de resposta: < 10us
- Temperatura de operação: -40℃~+85℃
- Número de operações: >1 milhão
- Deslizamento: <10% com 500g durante 24H
- Tamanho completo do sensor: 5mm*20mm
- Área de pressão: 5mm x 5mm (0,25cm^2)

Para fazer a sua ligação à PCB foram usados fios extra flexíveis de 0,205mm^2 com revestimento em silicone macio para uma maior durabilidade.

3.3. Linguagem e ambiente de programação

3.3.1. Arduino IDE

A *Espressif*, desenvolveu as suas próprias aplicações e bibliotecas em linguagem C para ser usada em IDE's como o Eclipse ou Visual Studio mas rapidamente percebeu que existia uma grande quantidade de utilizadores que usavam a IDE do Arduino não usavam as outras plataformas. Por esta razão, desenvolveram bibliotecas para permitir que a grande comunidade online do Arduino desenvolvesse o seu próprio código e melhorasse ainda mais as bibliotecas existentes.

O uso da plataforma do Arduino prende-se apenas com a disponibilidade de inúmeras bibliotecas online fazendo com que o desenvolvimento do código seja mais rápido.

3.3.2. Bibliotecas Arduino da Expressif

As bibliotecas são compostas por funções que podem ser chamadas no programa principal ou em outras bibliotecas. Estas bibliotecas são fundamentais para executar rotinas complexas e diminuir os tempos de programação. Dentro das bibliotecas, as funções ou classes que não são usadas não são carregadas no processador.

Neste projeto foram usadas as seguintes bibliotecas:

- WiFiClientSecure.h – Para implementação de ligações seguras usando TLS. Esta biblioteca é fundamental para criar a conexão com os servidores do Google;
- NTPClient.h – Dispõe de um conjunto de classes que permitem a ligação aos servidores de NTP e que devolvem a hora atual;
- EEPROM.h – Faz a gestão da memória ROM usada como memória permanente no ESP32;
- LoRa.h – Tem todas as funções para a comunicação via rádio entre os circuitos LoRa. Esta biblioteca permite codificação e descodificação de mensagens bem como conexões seguras a outros equipamentos;
- WiFi.h – Esta biblioteca contém todas as funções para gerir as ligação Wi-fi do ESP32. Permite encriptações WEP e WPA2 fazendo com que possa estabelecer ligação com totalidade dos routers domésticos. Contém funções para envio de pacotes UDP usados na configuração da rede Wi-fi do ESP32 pela app *SmartConfig*. A biblioteca é tão diversificada que neste projeto é usada apenas uma pequena parte das funções disponíveis;
- SPI.h – Permite estabelecer ligações SPI entre o ESP32 e a memória ROM, o IMU MPU9250 e o módulo LoRa;
- ArduinoJson.h – É a biblioteca que permite a comunicação entre diferentes linguagens de programação, neste caso, entre a linguagem C/C++ e linguagem web como HTML ou java script;
- esp.h, soc/rtc_io_reg.h, driver/rtc_io.h e rom/rtc.h – Estas bibliotecas são necessárias para executar código no ULP;
- MPU9250.h – Esta biblioteca permite a comunicação bidirecional entre o MPU9250 e o ESP32 quer por I2C ou por SPI (usada neste projeto). A troca de informação

permite receber os valores da aceleração, orientação, rotação e temperatura em tempo real e envio de comandos ou valores predefinidos de calibração do sensor.

- lmic.h e hal/hal.h – Usadas na comunicação LoRaWAN com o protocolo IBM LMIC (LoraMAC-in-C). Esta biblioteca permite a implementação do protocolo LoRaWAN para comunicação com a rede The Things Network [21].

4. Protótipo: *Hardware* e *Software*

Neste capítulo descreve-se os principais módulos do sistema bem como o seu funcionamento e interligação.

4.1. Hardware

Todas as comunicações entre o ESP32 e os periféricos estão representados na figura seguinte.

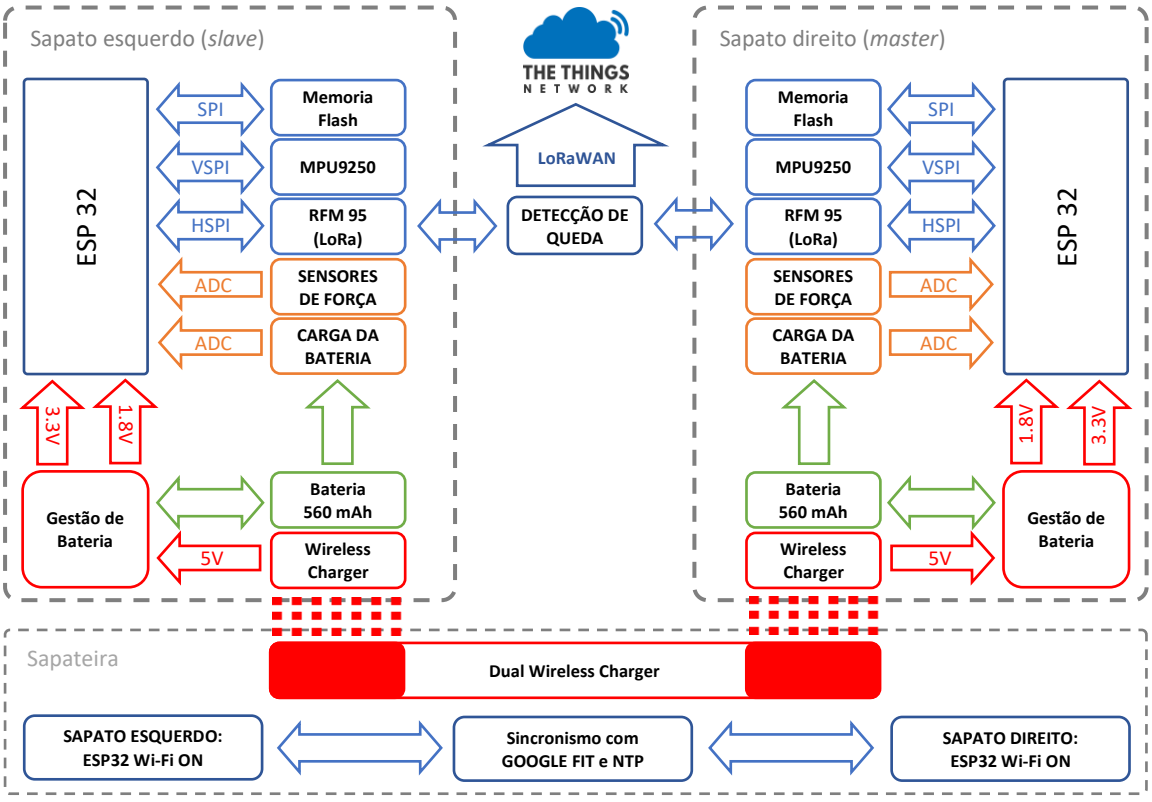


Figura 16 - Diagrama de blocos de hardware do sistema

O sistema apresentado é composto por quatro elementos principais:

- Sapateira;
- Sapato esquerdo (*Slave*);
- Sapato direito (*Master*);

- *Gateway* LoRaWAN.

No caso da sapateira apenas é composta por dois módulos transmissores sem fios para carregamento de cada um dos sapatos. Estes módulos são alimentados com 12V que através do circuito ressonante induzem tensão no circuito recetor gerando os 5V para o modulo de gestão de bateria.

Durante o carregamento, é ativada a porta GPI34 dando indicação para cada um dos sapatos comunicar a informação do movimento e fazendo o sincronismo com o servidor NTP.

Os elementos presentes nos sapatos são fisicamente iguais, diferindo apenas nas funções que lhes são assignadas.

Assim, cada um dos sapatos é constituído pelos seguintes módulos com as seguintes funcionalidades:

- **Memória flash**: Guardar toda a informação do movimento para posterior sincronismo com o Google FIT. Este modulo utiliza comunicação SPI com o ESP32.
- **MPU9250**: Sensor inercial usado na aquisição de dados de aceleração instantânea. A ligação ao controlador principal é feita através do segundo barramento SPI, o VSPI.
- **RFM95**: Modulo de comunicação LoRa que usa o 3º barramento SPI, o HSPI. Esta modulo é usado de duas formas: a primeira, em modo de baixo consumo para comunicar com o outro sapato e verificar se ambos os sapatos detetaram queda; e a segunda, para comunicar a queda no caso desta se confirmar. Neste modulo a potência é aumentada para conseguir a melhor qualidade e distância de transmissão.
- **Sensores de força**: Estes sensores resistivos ligam-se às portas ADC (1:0 a 1:3) do ESP32 a partir do ponto médio do divisor de tensão aplicado individualmente a cada um (Figura 15). Os valores lidos pelos conversores analógico-digital representam a força exercida de modo qualitativo. Para que fosse importante ter a informação quantitativa teria de se optar por outro tipo de sensores (células de carga, sensores de torque, …).

- **Gestão de bateria**: O modulo é responsável por alimentar todos os restantes módulos dentro de cada um dos sapatos. Além da entrada referida na descrição da sapateira,

este modulo ainda usa mais duas portas, a GPIO16 responsável por ativar o circuito de leitura da carga da bateria. Esta capacidade é lida na porta ADC1:7 através de um divisor de tensão.

Por fim, o *gateway* LoRaWAN faz o interface entre os módulos LoRa e o portal TheThingsNetwork.org. No caso deste projeto é utilizado o *gateway* da Fikalab em Tomar, mas poderá ser usado outro caso se encontre mais próximo do Smartshoes.

A interligação entre todos os módulos é feita segundo o diagrama dos anexos 1 e 2 (Esquemático da placa de circuito impresso).

Dentro do sapato, os sensores de força e todos os componentes são colocados nas posições representadas na figura seguinte:

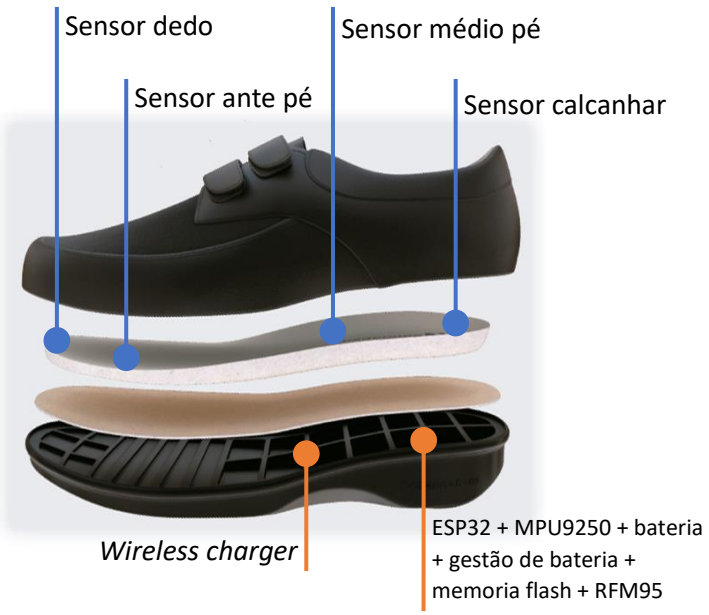


Figura 17 - Localização dos componentes

A posição dos eixos XYZ do sensor inercial MPU9250 está de acordo com a figura a baixo. O eixo Z está perpendicular ao solo com o sei sentido a apontar para cima (sentido contrário à aceleração da gravidade). O eixo Y fica paralelo ao solo com o sentido da direita para a esquerda. Por fim, o eixo X fica também paralelo com o solo e com o sentido de trás para a frente.

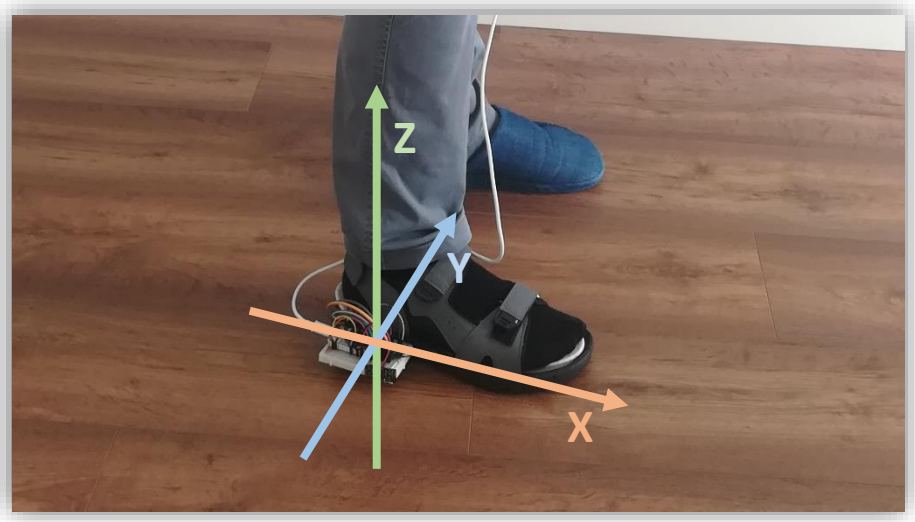


Figura 18 - Orientação dos eixos do sensor inercial

Dentro do MPU9250 estão disponíveis 10 sensores:

- Acelerómetro - Aceleração no eixo X, Y e Z;
- Giroscópio - Rotação sobre o eixo X, Y e Z;
- Magnetómetro - Campo magnético na direção X, Y e Z;
- Termómetro.

Neste projeto apenas são usados os 3 sensores de aceleração permanecendo os restantes 7 desligados para evitar consumos desnecessários de energia. A escolha de um sensor inercial composto por 10 sensores e usar apenas 3 prende-se apenas com as possibilidades futuras de ampliação das funcionalidades. Se esta possibilidade não fosse prevista poderia ser usado o acelerómetro de 3 eixos IAM-20381.

Em relação à bateria usada, e tal como referido nos objetivos, deverá ter uma autonomia de 5 dias e ser carregada por indução. Assim, utilizando o recurso de cálculo de autonomia de uma bateria do site www.of-things.de/battery-life-calculator.php foi possível dimensionar a capacidade de bateria mínima de 133mAh. Para isso foram usados os seguintes dados:

- Tempo de execução do código: 1 mS
- Tempo adormecido: 50 mS
- Consumo durante execução: 20 mA
- Consumo adormecido: 0,5 mA

- Dissipação de carga: 20%

Os valores usados de tempos de execução e consumo de bateria foram os valores máximos obtidos em condições não ótimas, pelo que, os valores numa utilização real serão inferiores.

Para que fosse possível aumentar a autonomia mantendo o peso reduzido, a bateria de 133mAh (2 gramas) será substituída por uma de 560mAh (9 gramas). Assim, em condições não ideais, a autonomia estimada é de 21 dias, podendo chegar a um mês sob condições ótimas e utilização de 16 horas por dia.

Além da bateria o sistema é composto por:

- Carregador sem fios com uma saída de 5V DC;
- Circuito de carga de bateria;
- Regulador com controlador LDO para 3.3V;
- Regulador com controlado LDO para 1.8V;
- Entrada analógica para leitura da capacidade da bateria;
- Entrada digital para avaliar quando o sistema está a ser carregado.

4.2. Placa de circuito impresso

A placa de circuito impresso, cujos esquemas e PCB estão em apêndice a este relatório, foi concebida com dimensão reduzida para que fosse possível ser inserida dentro do sapato. Assim, foi criada uma PCB com 2 lados, de dimensão 23.29mm / 50.16mm contendo todos os componentes descritos no ponto anterior e dispostos pelas duas camadas.

Os principais componentes usados são SMD’s 0603 (resistências, condensadores, bobinas e diodos) e SOT23-3 e SOT23-5 (LDO e mosfets). Esta escolha deve-se ao facto de serem componentes pequenos e fáceis de encontrar no mercado.

Na camada superior (Figura 19) estão presentes o MPU9250, a memória flash, um par de *pads* necessário para iniciar o ESP32 em modo de programação e as *pads* para fazer o upload do software para o ESP32.

A *pad* de programação tem um conector JST1.25 de 8 pinos. Esta escolha deve-se ao facto de ser um conector pequeno e muito robusto.

A capa superior tem ainda os textos identificativos do projeto.

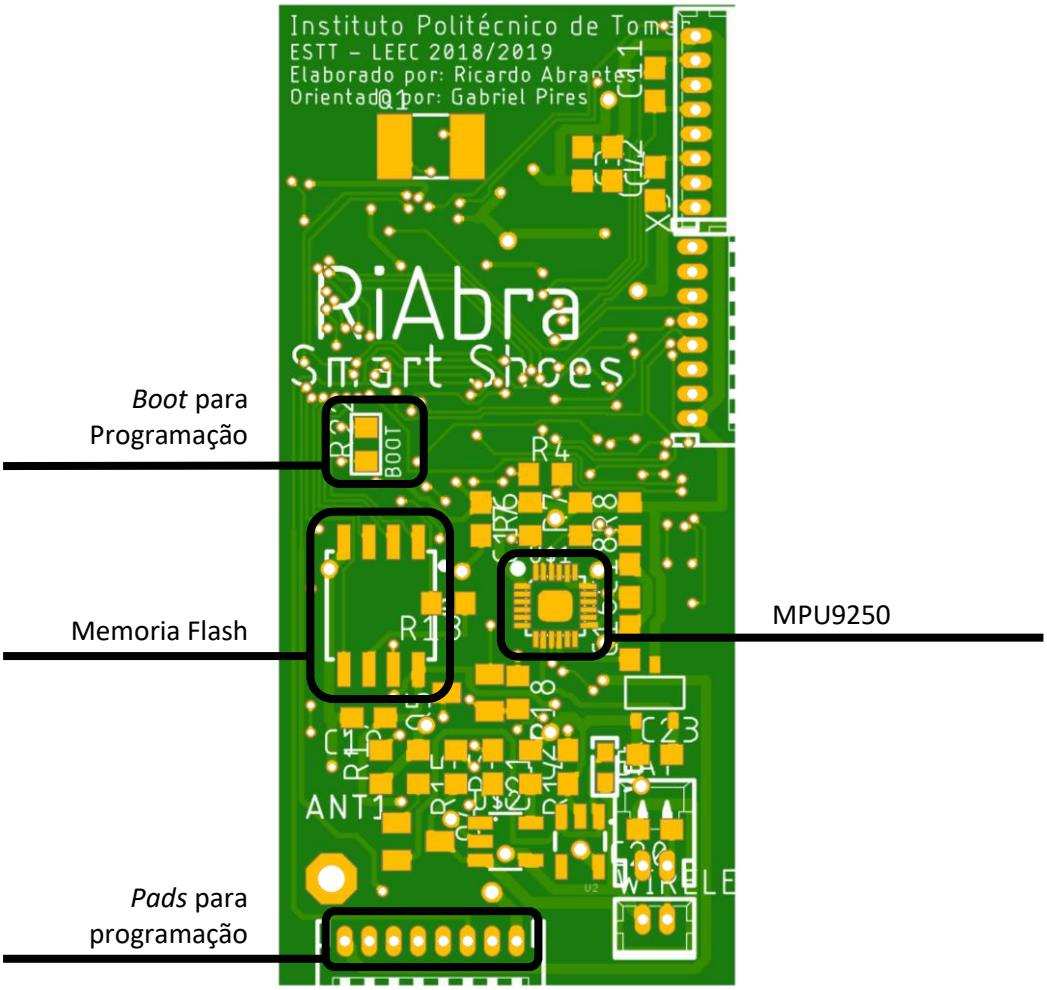


Figura 19 - PCB - Camada superior

Na camada inferior (Figura 20) encontra-se o ESP32 e o RFM95, elementos que têm maior probabilidade de aquecer e, assim, voltados para baixo, ficam mais longe do calor emitido pelo pé.

Nesta camada é também onde se encontram as *pads* para ligar os sensores de força e onde se encontram 8 *pads* que poderão ser usadas para ampliações futuras. Nestas *pads* encontram-se pinos para alimentação, comunicação I2C, portas digitais e portas ADC.

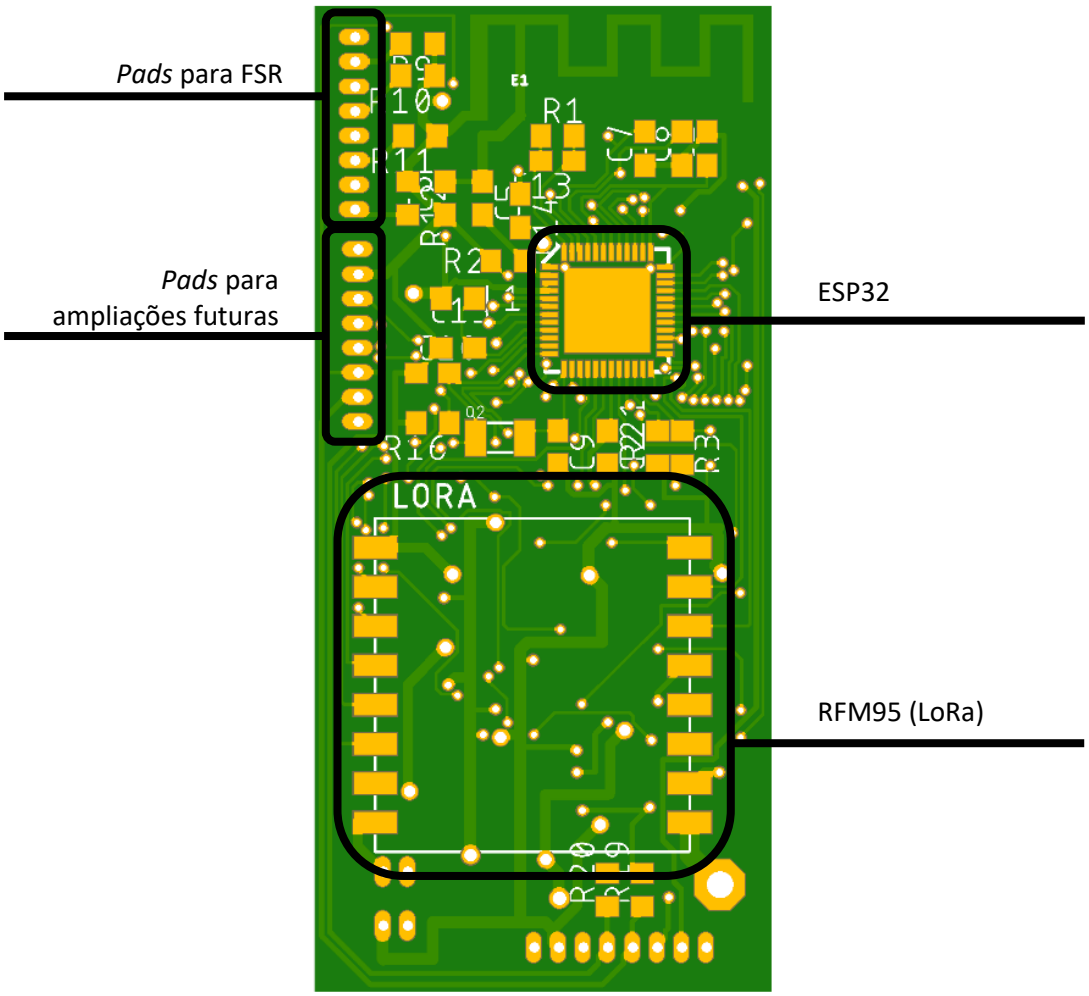


Figura 20 - PCB - Camada inferior

4.3. Arquitetura de software

O Software está dividido em 5 funções principais e 3 secções:

- **Aquisição de dados** – Funções onde são recolhidas as informações dadas pelo acelerómetro e sensores de pressão (figura seguinte em verde);
- **Comunicação** – Funções de transmissão de dados entre sapatos e comunicação LoRaWAN para alarme de queda (figura seguinte em vermelho);

- **Sincronização de dados** – Funções executadas quando o sistema está a carregar a bateria e o controlador se liga à rede wi-fi para transferir todos os dados guardados e fazer o sincronismo com o servidor NTP (figura seguinte em azul);

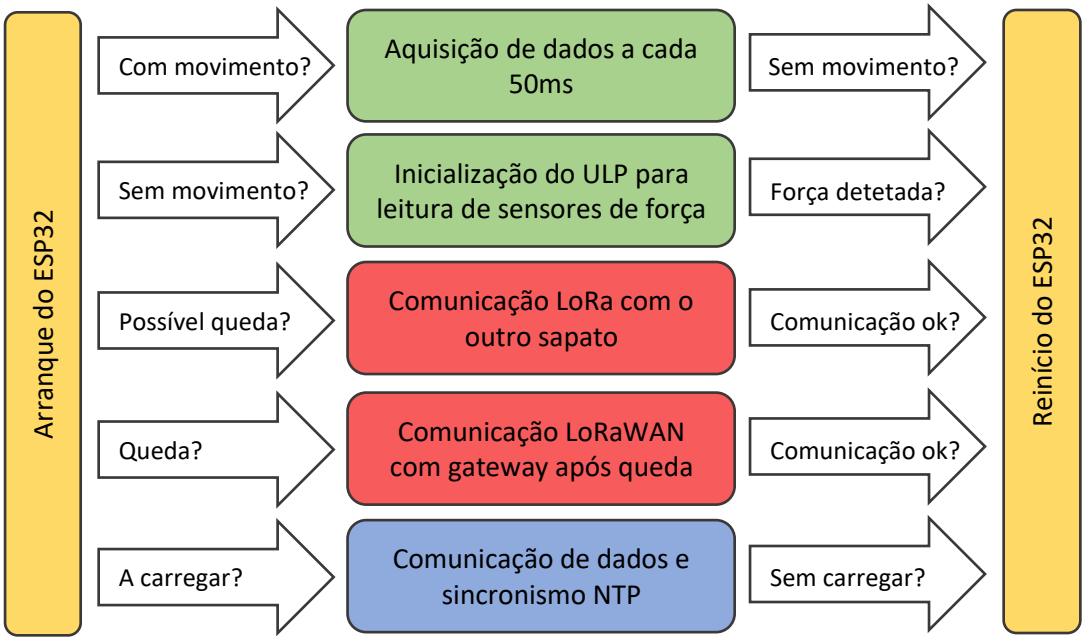


Figura 21 - Diagrama de software e principais funcionalidades

A execução da função de aquisição de dados é feita a cada 50 milissegundos demorando aproximadamente 200 microssegundos a executar todo o código. Caso a pessoa esteja sem atividade durante 10 segundos, o tempo entre amostras passa para 400 milissegundos. Caso continue sem atividade durante mais do que 1 minuto, o processador principal é desligado ficando apenas o ULP a adquirir dados dos sensores de força a cada segundo e, no caso de detetar diferenças significativas de pressão, o processador principal é novamente ligado e retoma a função de aquisição normal.

Os dados recolhidos em cada ciclo são armazenados numa tabela de 400 elementos. Cada elemento da tabela é do tipo *struct* composto por:

- **tempoActual** (uint64_t)
- **anDedo** (uint16_t)
- **anAntePe** (uint16_t)
- **anMedioPe** (uint16_t)

- **anCalcanhar** (uint16_t)
- **aX** (float)
- **aY** (float)
- **aZ** (float)

O tempo atual guarda número de segundos desde o dia 1 de Janeiro de 2018 e é usado para gerar uma etiqueta temporal dos dados que são enviados para o *GoogleFit*. As variáveis anDedo, anAntePe, anMedioPe e anCalcanhar guardam o valor obtido dos sensores de força. Por fim, as variáveis aX, aY e aZ armazenam os valores da aceleração enviados pelo sensor inercial.

Associados a esta tabela existem ainda as variáveis:

- **Índice de escrita**: Variável do tipo *"uint16_t"* onde é guardado o número do índice onde os dados são guardados.

- **Índice de mínimo e Índice de máximo:** *Struct* de variáveis inteiras do tipo *"uint16_t"* que guardam o índice da tabela onde cada um dos valores teve o seu valor mínimo e máximo respetivamente. O máximo só ocorre depois do mínimo e o mínimo depois do máximo, não é possível detetar 2 picos iguais e seguidos.

- **Confirmação de picos positivos**: *Struct* de variáveis booleanas que muda o estado de 0 para 1 aquando um pico em cada um dos valores obtidos pelos sensores de força quando as diferentes zonas do pé assentam no chão.

  Para que não sejam detetados 2 picos seguidos, o índice onde ocorreu o pico é guardado durante os 4 ciclos seguintes e, caso não exista um valor superior, o pico é confirmado ativando a o bit correspondente na *Struct* (Figura 22).

Figura 22 - Método para deteção de picos nos valores dos sensores de força

Em todo o código, as variáveis não foram usadas e criadas na sua forma mais comum, por exemplo, para guardar um número inteiro é usada uma variável INT que, no Arduino IDE compilando o código para num processador de 8 bit, corresponde a 2 bytes, no caso do ESP32, o compilador está configurado por defeito para que use 4 bytes. Como este projeto faz uso de grande parte da memória disponível, houve o cuidado de ajustar os *dataypes* mais adequados que otimizassem o espaço de memória. Todas as variáveis foram definidas como *uint8_t, int8_t, uint16_t,* etc. Este ajuste permite também que se possa usar o mesmo código em outro processador mantendo a quantidade de memória usada.

Com esta redefinição das variáveis na memória foi possível passar de 80% da memória ocupada para cerca de 50%.

A figura seguinte representa de forma mais pormenorizada o fluxograma de software dos ESP32 presentes nos sapatos.

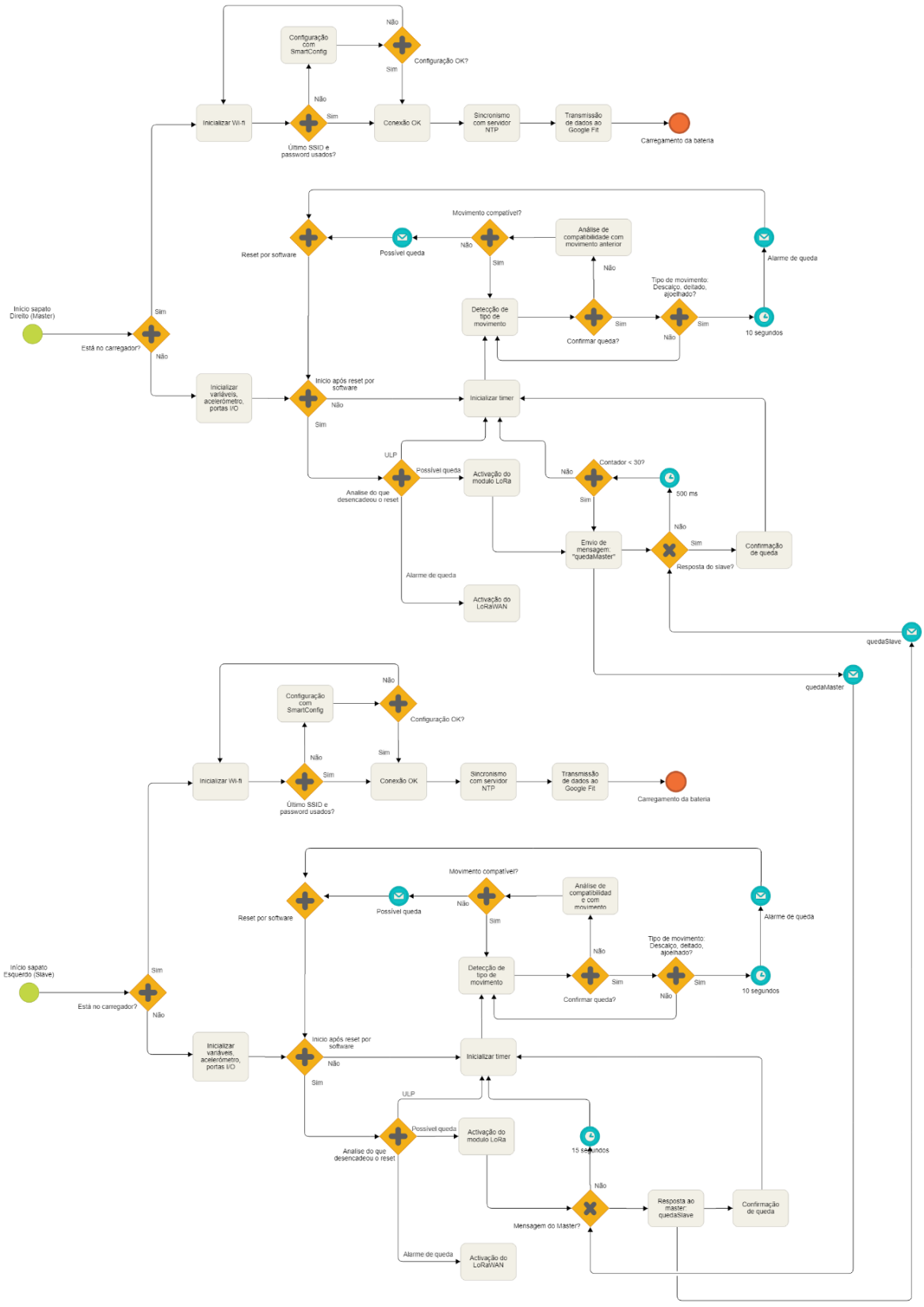


Figura 23 - Fluxograma de software

4.4. Deteção de tipos de movimento

Tal como referido no ponto 3, a deteção de queda não é feita apenas pela análise de movimentos que se assemelhem a queda, mas também pela análise dos diversos movimentos possíveis e a compatibilidade entre si. Assim, foram definidos 10 tipos de acontecimentos ou estados que podem ocorrer e ser detetados com os sapatos:

- **Andar** – Este tipo de movimento é detetado inicialmente através da análise dos sensores de força. Estes são lidos de forma independente e quando é detetado um pico máximo no valor lido dos 4 sensores é considerado que o pé se encontra no chão e o passo é iniciado.

  Quando existirem novamente os picos nos valores dos sensores de força, o passo termina e são analisados todos os valores do acelerómetro bem como o tempo entre passos.

  A Tabela 3 exemplifica o método para a deteção do passo. As cores usadas na tabela são as mesmas usadas no gráfico 13 de modo a uma compreensão mais simples.

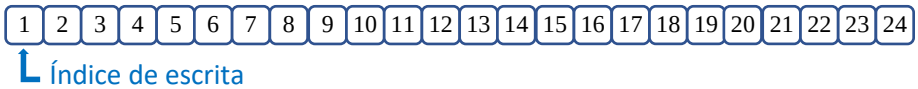
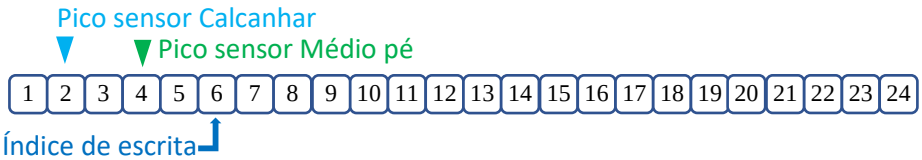
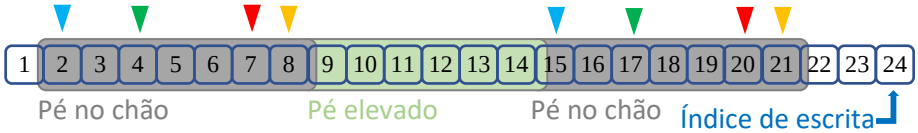
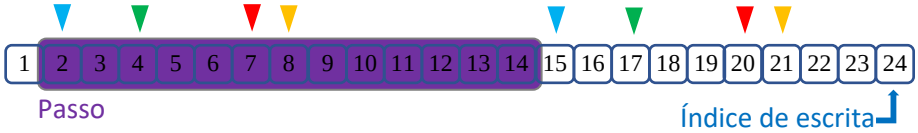
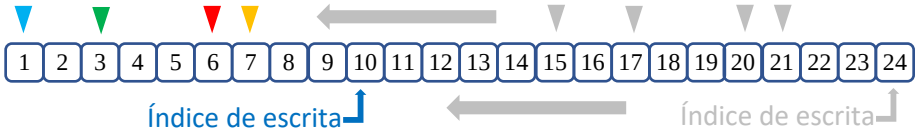
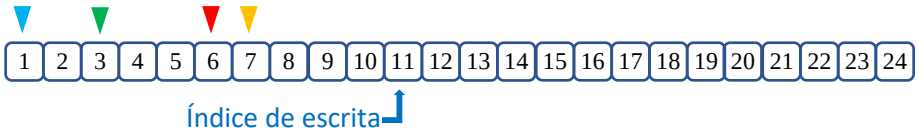
1 – Inicio da aquisição de dados e escrita da tabela.	
2 – Deteção de picos máximos nos valores dos sensores de força.	
3 – Quando são detetados picos em 3 dos 4 sensores é confirmado o pé no chão.	
4 – Deteção do seguinte "pé no chão".	
5 – Análise do passo feito entre o Índice 2 e o Índice 14.	
6 – Após a análise, o passo seguinte é movido para o início da tabela.	
7 – Continuação da leitura dos sensores para detetar novo passo.	

Tabela 3 - Método para deteção de passos

A análise do passo descrita no ponto 5 da tabela anterior tem por base valores recolhidos e representados graficamente no gráfico seguinte.

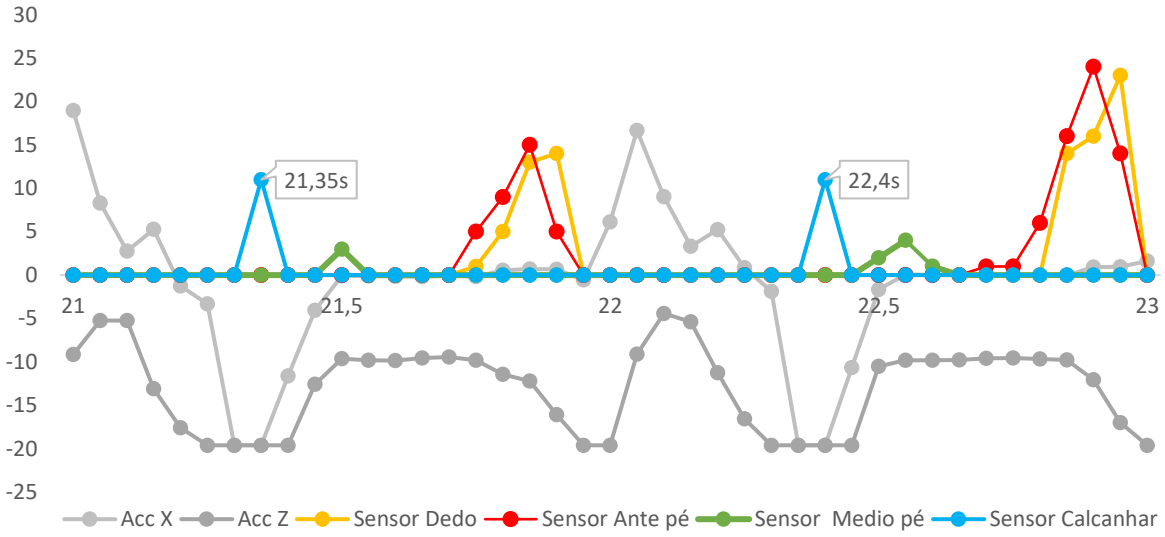


Gráfico 1 - Análise de valores de pico dos sensores de força

Pelo gráfico anterior é possível obter o tempo de 1 segundo entre passos (desde o primeiro pico no valor do calcanhar até ao momento imediatamente antes do segundo pico do valor do calcanhar).

Em cada um dos passos também é possível detetar uma aceleração positiva no eixo X (pé deslocar-se para a frente) e uma desaceleração do pé quando este assenta no chão.

Quando o pé está totalmente assente no chão a aceleração no eixo Z é igual à aceleração da gravidade (- 9.8 m/s^2).

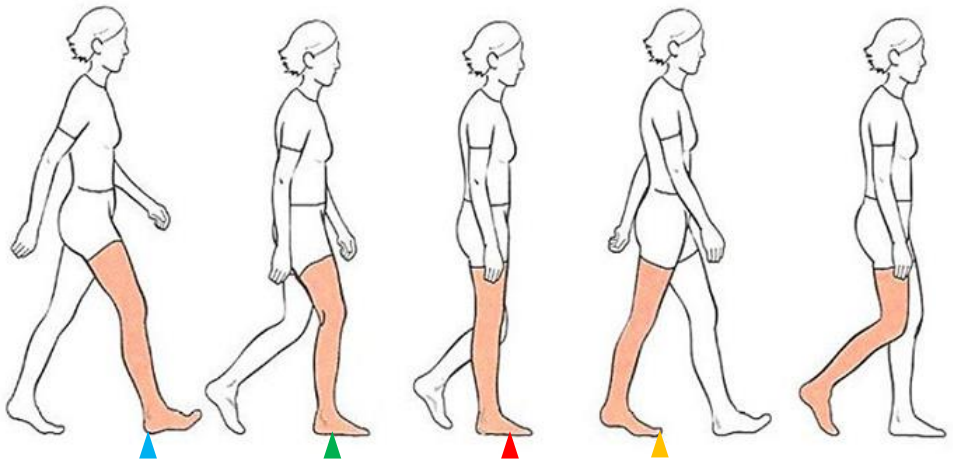


Figura 24 - Cinco fases de apoio do pé no solo [1] apontando em qual das fases os picos dos sensores de força são detetados

- **Correr** – É usado o mesmo método descrito no tipo de movimento “andar” com a diferença que os picos nos sensores de força se originam muito mais rápido, normalmente inferior a 300 milissegundos.

- **Saltar** – A deteção é feita durante o momento do embate do pé no chão onde se verifica uma aceleração negativa muito grande no eixo Z, assim, em duas amostragens consecutivas:

    - A primeira tem uma aceleração próxima de zero pois a pessoa está a cair puxada apenas pela força da gravidade;

    - A segunda, quando se dá o impacto, no eixo Z é detetada a força da gravidade e a inercia do acelerómetro.

  Após a sequência descrita, o sistema confirma com os sensores de força se, durante os 2 momentos acima descritos e nos 2 momentos imediatamente seguintes, houve um pico positivo em algum dos sensores de força. Nesta fase final do salto, bastam os sensores posicionados no dedo e no ante pé para confirmar o salto, ainda assim, é usado também o sensor posicionado no medio pé.

  Como exemplo, mostra-se o código fonte usado para detetar este tipo de movimento é o seguinte:

```
1  // Detecção de pessoa a saltar (quando a pessoa aterra)
2   if (tipoMovimentoActual != 3 && ciclo[writeIndex - 1].aZ >= -6 &&
    ciclo[writeIndex].aZ <= -15) {
3     possivelSalto = true;
4     indexPossivelSalto = writeIndex - 1;
5   }
6   if (possivelSalto == true && writeIndex == (indexPossivelSalto +
    3)) {
7     bool saltoConfirmado = false;
8     for (uint8_t i = 0; i < 2; i++) {
9       if ((ciclo[indexPossivelSalto + i].anDedo <
        ciclo[indexPossivelSalto + i + 1].anDedo) &&
        (ciclo[indexPossivelSalto + i + 1].anDedo >
        ciclo[indexPossivelSalto + i + 2].anDedo) &&
        (ciclo[indexPossivelSalto + i].anDedo > 200 ||
        ciclo[indexPossivelSalto + i + 2].anDedo > 200))
```

```
10          saltoConfirmado = true; //No caso de ocorrer um pico nos
                                    FSR no momento atual, antes ou
                                    depois o salto é confirmado
11        if ((ciclo[indexPossivelSalto + i].anAntePe <
          ciclo[indexPossivelSalto + i + 1].anAntePe) &&
          (ciclo[indexPossivelSalto + i + 1].anAntePe >
          ciclo[indexPossivelSalto + i + 2].anAntePe) &&
          (ciclo[indexPossivelSalto + i].anAntePe > 200 ||
          ciclo[indexPossivelSalto + i + 2].anAntePe > 200))
12          saltoConfirmado = true;
13        if ((ciclo[indexPossivelSalto + i].anMedioPe <
          ciclo[indexPossivelSalto + i + 1].anMedioPe) &&
          (ciclo[indexPossivelSalto + i + 1].anMedioPe >
          ciclo[indexPossivelSalto + i + 2].anMedioPe) &&
          (ciclo[indexPossivelSalto + i].anMedioPe > 200 ||
          ciclo[indexPossivelSalto + i + 2].anMedioPe > 200))
14          saltoConfirmado = true;
15      }
16      if (saltoConfirmado)
17        mudaMovimentoPara(saltar);
18      else
19        possivelSalto = false;
20    }
21    if (possivelSalto == true && writeIndex > (indexPossivelSalto +
      3))
22      possivelSalto = false;
```

Para um entendimento mais simples do código anterior deverá ser tido em conta:

- A variável "tipoMovimentoActual" guarda o movimento que está a ocorrer e o número do movimento é definido aquando da criação dos diferentes tipos de movimento (estrutura de dados com um inteiro de 8 bits e outro de 16 bits):

```
1  typedef struct  {
2    uint8_t num;
3    uint16_t compativelAnterior; //Este número é mostrado em bits e
                                  cada bit está na posição do número
                                  do correspondente estado
4  } tipoMovimento;
5  tipoMovimento ND       { 0, 0b1111111111111111};
6  tipoMovimento queda    { 1, 0b0000000010011101};
7  tipoMovimento andar    { 2, 0b0000000010011101};
8  tipoMovimento correr   { 3, 0b0000000000011101};
9  tipoMovimento saltar   { 4, 0b0000000010011101};
10 tipoMovimento ajoelhar { 5, 0b0000010010100001};
11 tipoMovimento deitar   { 6, 0b0000011111000001};
12 tipoMovimento parado   { 7, 0b0000011111110101};
13 tipoMovimento gatinhar { 8, 0b0000010100100001};
14 tipoMovimento sentado  { 9, 0b0000011010000001};
15 tipoMovimento descalco {10, 0b0000011011000001};
```

- O salto apenas é detetado caso a pessoa não esteja a correr, visto que durante a corrida são produzidas acelerações similares às do salto.

- A posição de escrita na tabela, descrita no ponto 4.3, é guardada na variável "writeIndex".

- A função "mudaMovimentoPara(tipoMovimento);" recebe o novo tipo de movimento e faz a analise de compatibilidade com o movimento anterior. No caso de não haver compatibilidade, o ESP32 é reiniciado e ativada a comunicação LoRa para comunicação com o outro sapato.

```
1  void mudaMovimentoPara(tipoMovimento mov) {
2    if (bitRead(mov.compativelAnterior, tipoMovimentoAnterior[0]) ==
     0 && tipoMovimentoAnterior[0] != 1) {
                                          //verifica compatibilidade
                                          com o movimento anterior
3      possivelQueda = true;
4      mov = queda;
5      endTimer();
6      esp_sleep_enable_timer_wakeup(10); //fica adormecido 10us
                                          apenas para reiniciar e
                                          ligar o LoRa
7      esp_deep_sleep_start();
8    }
9    if (tipoMovimentoAnterior[0] != mov.num) {
10     for (uint8_t i = 19; i > 0; i--)  //copia movimento para o
                                        "arquivo" de movimentos
11       tipoMovimentoAnterior[i] = tipoMovimentoAnterior[i - 1];
12     tipoMovimentoAnterior[0] = mov.num;
13     tipoMovimentoActual = mov.num;
14     actividadeDetectada = true;
15   }
16 }
```

- **Ajoelhar** – A deteção é feita exclusivamente com os valores do acelerómetro. Sempre que o valor da aceleração da gravidade do eixo negativo Z passa para o eixo positivo X é iniciado um contador (variável inclinacaoFrenteTras do tipo uint8_t) e caso este contador chegue a um limite estabelecido é analisado toda a atividade do acelerómetro no tempo em que o contador está ativo para garantir que efetivamente a pessoa se ajoelhou.

Considera-se que a pessoa se ajoelhou caso demore mais de 1,5 segundos (30 ciclos) entre passar de 90% da vertical (aZ= -9m/s^2) até 90% horizontal (aX= 9m/s^2). No caso do tempo ser inferior a 1,5 segundos é considerada uma possível queda.

Na tabela seguinte são representadas as possíveis situações da inclinação.

Figura 25 - Situações possíveis na deteção de inclinação dos sapatos

- **Deitado** – Este tipo de estado é detetado da mesma forma do anterior, mas com a aceleração da gravidade a passar do eixo negativo Z para o eixo negativo X ou eixo (positivo ou negativo) Y. Da mesma forma, caso o tempo de inclinação seja inferior a 1,5 segundos é logo desencadeado o procedimento de possível queda.

- **Gatinhar** – Para ser detetado este tipo de movimento é necessário que a aceleração da gravidade esteja a ser lida, em mais de 50%, no eixo positivo do X.

- **Parado** – Caso haja deteção de pressão em 2 dos sensores e o não haja inclinação durante 2 segundos e meio o estado é definido como parado. É também usado um contador similar ao usado no movimento “Ajoelhar”.

  Para a deteção deste tipo de movimento é usada a seguinte função:

```
1  // Detecção de pessoa parada
2  if (tipoMovimentoActual != 7 && confirmacaoPessoaParada < 50 &&
   somaSensoresPeNoChao > 1000 && absfloat(ciclo[writeIndex].aZ) >
   limiteSuperiorAccZ)
3    confirmacaoPessoaParada++;
4  if (confirmacaoPessoaParada > 0 && (somaSensoresPeNoChao < 1000
   || absfloat(ciclo[writeIndex].aZ) < limiteSuperiorAccZ))
5    confirmacaoPessoaParada--;
6  if (confirmacaoPessoaParada >= 50) {
7    confirmacaoPessoaParada = 0;
8    mudaMovimentoPara(parado);
9  }
```

Alem do descrito no salto, este tipo de movimento usa ainda a função “absfloat()” que devolve o valor absoluto de uma variável do tipo float. Usa também a variável “somaSensoresPeNoChao” onde é guarda a soma dos valores obtidos pelos FSR.

- **Descalço** – Caso não haja deteção de força nos FSR durante um período pré-determinado é considerado que o sapato está descalço fazendo com que possa adormecer o processador principal deixando apenas o ULP a funcionar usando a função seguinte:

```
1  void initialize_ULP() {
2    memset(RTC_SLOW_MEM, 0, CONFIG_ULP_COPROC_RESERVE_MEM);
3
4    adc1_config_channel_atten(ADC1_CHANNEL_0, ADC_ATTEN_DB_11);
5    adc1_config_channel_atten(ADC1_CHANNEL_1, ADC_ATTEN_DB_11);
6    adc1_config_channel_atten(ADC1_CHANNEL_2, ADC_ATTEN_DB_11);
7    adc1_config_channel_atten(ADC1_CHANNEL_3, ADC_ATTEN_DB_11);
8    adc1_config_width(ADC_WIDTH_BIT_12);
9    adc1_ulp_enable();
10
11   rtc_gpio_init(GPIO_NUM_36);
12   rtc_gpio_init(GPIO_NUM_37);
13   rtc_gpio_init(GPIO_NUM_38);
14   rtc_gpio_init(GPIO_NUM_39);
15
16   const uint32_t measurements_per_sec = 1;
17   const ulp_insn_t program[] = {
18     I_ADC(R0, 0, ULP_AnDEDO),       // lê o valor analógico e copia
                                       para R0
19     M_BGE(1, ULP_AnTRIGGER),        // se o valor contido no registo
                                       R0 é maior ou igual a
                                       ULP_AnTRIGGER, salta para o
                                       LABEL 1
20     I_ADC(R0, 0, ULP_AnANTEPE),
21     M_BGE(1, ULP_AnTRIGGER),
22     I_ADC(R0, 0, ULP_AnMEDIOPE),
```

```
23      M_BGE(1, ULP_AnTRIGGER),
24      I_ADC(R0, 0, ULP_AnCALCANHAR),
25      M_BGE(1, ULP_AnTRIGGER),
26      M_LABEL(0),
27      I_HALT(),  // caso chegue a este ponto é porque não detetou
                   nenhum valor analógico superior a ULP_AnTRIGGER e
                   nesse caso é adormecido.
28      M_LABEL(1),
29      I_WAKE(),
30      I_HALT()
31    };
32
33    // Carrega o programa do ULPna RTC_SLOW_MEM
34    size_t size = sizeof(program) / sizeof(ulp_insn_t);
35    ulp_process_macros_and_load(0, program, &size);
36
38    const uint32_t sleep_cycles = rtc_clk_slow_freq_get_hz() /
measurements_per_sec;
39    REG_WRITE(SENS_ULP_CP_SLEEP_CYC0_REG, sleep_cycles);
40
41    wakeupAn = true;
42
43    // Start ULP
44    ulp_run(0);
45    esp_sleep_enable_ulp_wakeup();
46    esp_deep_sleep_start();
47  }
```

- **Queda** – Este é o estado que será detetado sempre que todos os outros falharem ou que haja incompatibilidade entre eles.

A incompatibilidade entre estados e tipos de movimento é o ponto onde este projeto se destaca da grande maioria de outros projetos ou produtos presentes no mercado. Quando um tipo de movimento é detetado é imediatamente comparado com o movimento que tinha no momento imediatamente anterior, assim, caso não haja compatibilidade entre estados ou movimentos, é imediatamente definido o estado de queda.

A definição de estado como queda não desencadeia de imediato o alarme.

O estado de queda terá de passar por 3 estados intermédios até que seja desencadeado o alarme:

- Definição de estado para “queda” onde a queda ainda não está confirmada, trata-se apenas de uma possível queda.
- Confirmação da queda entre ambos os sapatos. Caso os 2 sapatos detetem uma queda a mesma fica confirmada;
- Caso a pessoa não se levante durante 20 segundos é desencadeado o procedimento de alarme.

Na tabela seguinte são apresentadas todas as compatibilidades do movimento detetado no presente face ao movimento em que se encontrava anteriormente.

Movimento anterior \ Movimento atual	Queda	Andar	Correr	Saltar	Ajoelhar	Deitado	Parado	Gatinhar	Descalço
Queda	●								
Andar		●	●	●			●		
Correr		●	●	●					
Saltar		●	●	●			●		
Ajoelhar	●				●		●	●	
Deitado	●					●	●		●
Parado		●		●	●	●	●		●
Gatinhar	●					●	●	●	
Descalço	●				●	●	●	●	●

Tabela 4 - Compatibilidade entre o movimento atual e o movimento em que se encontrava anteriormente

Pela tabela anterior podemos perceber que se uma pessoa está a correr, anteriormente apenas poderia estar a andar, a correr ou a saltar. Se a mesma pessoa está a correr e de seguida está deitada sem passar por nenhum dos estados compatíveis, possivelmente, deu uma queda.

4.4.1. Comunicação entre sapatos

Entre o sapato direito e o esquerdo a comunicação é feita usando o módulo de comunicação LoRa no seu modo de baixo consumo de energia. Quando é detetada uma possível queda no sapato direito, este envia a cada 500 milissegundos e durante 15 segundos a mensagem "quedaMaster".

No caso do sapato esquerdo detetar uma possível queda, liga também o modulo LoRa e fica em modo constante de escuta durante 15 segundos.

Caso ambos os sapatos detetem queda, o sapato esquerdo recebe a mensagem e responde com a mensagem “quedaSlave”. Assim, os 2 sapatos confirmam que realmente houve uma queda.

No caso de um dos sapatos não ter detetado a queda, não há comunicação entre eles e seguem com a aquisição normal de dados.

4.4.2. Comunicação LoRaWAN

Para que a comunicação LoRaWAN seja ativada é necessário que ambos os sapatos tenham detetado uma queda. Assim, apenas o sapato master irá comunicar a queda.

Na comunicação é transmitida a mensagem “queda”. Nesta transmissão, o sapato liga-se ao gateway mais próximo autenticando-se com o seu “Devive EUI” e transmitindo a mensagem com uma encriptação AES Secured de 128 bits, esta encriptação é única e disponibilizada na conta pessoal do utilizador.

4.4.3. Ligação wi-fi

No sincronismo de dados o controlador transfere todos os dados para que possam ser processados pelo *Google Fit*, antes disso, faz a atualização do seu relógio interno para que

possa referenciar temporalmente todos os movimentos que fará quando entrar novamente em modo de aquisição de dados.

O ESP32 tentará sempre conectar-se à última rede wi-fi que esteve conectada, caso a rede não esteja disponível entra em modo de configuração através da aplicação *SmartConfig*.

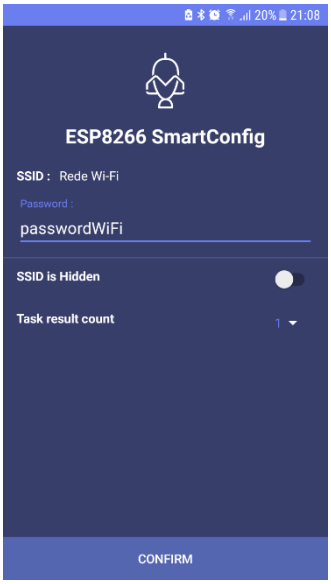


Figura 26 - App SmartConfig

SmartConfig [24] é uma aplicação disponível para android ou iOS em que apenas é necessário que o *smartphone* ou tablet estejam ligados à rede onde se irá ligar também o ESP32, definir a password da rede e clicar em “Confirm”. Assim, a aplicação envia pacotes encriptados com o SSID e password da rede por *broadcast* para que quando esse pacote for recebido pelo ESP32 seja usado para configurar a sua própria rede wifi. No final é sempre retornado uma mensagem em como o ESP32 já está conectado à rede Wi-fi.

A atualização do RTC interno é feita através do protocolo NTP [25], que aqui é usada na sua forma mais simples, procura o tempo no servidor NTP e atualiza o RTC.

5. Ensaios e resultados experimentais

Neste capítulo serão descritos todos os ensaios e testes a fim de validar os métodos adotados. Inicialmente foi testado o ESP32 numa montagem integrada e desenvolvida pela *Heltec* (Figura 27) que, além do ESP32-D0WDQ6, combina também na mesma PCB um ecrã OLED, um regulador de tensão, um conversor USB-Serial e um RFM95.

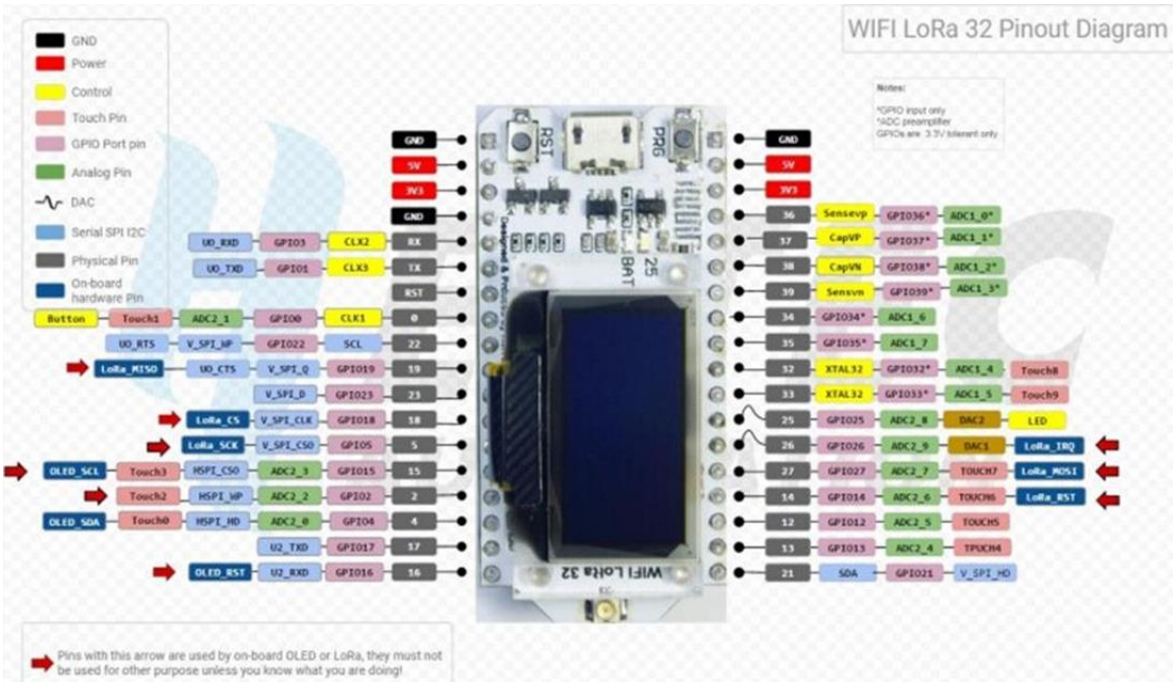


Figura 27 - WIFI LoRa 32 – Diagrama de pinos

Seguidamente, todos os periféricos foram testados isoladamente e só no final é que se procedeu à assemblagem e teste de todos os periféricos no mesmo circuito.

5.1. Ensaios com a placa Heltec LoRa 32 e circuitos periféricos

Inicialmente o ESP32 foi escolhido por possuir o ULP e assim conseguir desenvolver uma aplicação com um consumo muitíssimo baixo. No entanto, logo no início dos testes, a programação em *Assembly* e a pouca documentação existente obrigaram a abandonar o ULP como processador geral e a desenvolver o código para o processador principal. Assim, o processador principal é adormecido e acordado em períodos regulares executando todo o código em cerca de 200 microssegundos, enquanto que o ULP é usado quando não é detetado nenhum movimento, correndo um script mais simples e acordando o processador principal quando deteta movimento.

Para se conseguir cumprir o objetivo da duração da bateria superior a 5 dias fizeram-se vários testes com diferentes frequências de amostragem chegando-se à conclusão que o melhor compromisso fiabilidade/consumo seria obtido com uma frequência de amostragem de 20Hz (ou seja, o processador é acordado a cada 50ms).

A gestão de carga e descarga da bateria foi também tida em conta nos testes com a placa da Heltec visto ter um conector JST1.25 para se ligar uma bateria de 3.7V. Foram testadas baterias de 133mA e 560mA tendo sido estas última a opção escolhida por conseguir reunir um tamanho aceitável, baixo peso e a duração pretendida.

Para os testes de comunicação sem fios LoRa foram usadas duas placas Heltec LoRa 32 em que eram enviadas mensagens da placa master para a *slave*, as quais respondiam e eram enviadas novas mensagens. Estes testes visaram avaliar os consumos mais baixos, logo, usou-se o modo de baixo consumo do circuito LoRa que se mostrou muito eficaz até ao limite a que foi testado: 10 metros de distância com paredes no meio. Este foi um teste limite visto que, numa situação de queda, os 2 pés estarão próximos e nunca afastados de 10 metros.

Não menos importantes foram os testes com Wi-fi. Inicialmente foram usados exemplos de *webservers* básicos para fazer a ligação à rede e mostrar valores numa página web. Seguidamente foram introduzidas todas as funcionalidades exigidas, ligação ao servidor

NTC e atualização do RTC interno, ligação com os servidores do Google Fit e configuração do SSID e senha através da app ESP *SmartConfig*.

A Figura 28 mostra a montagem que foi usada para fazer todos os testes descritos anteriormente.

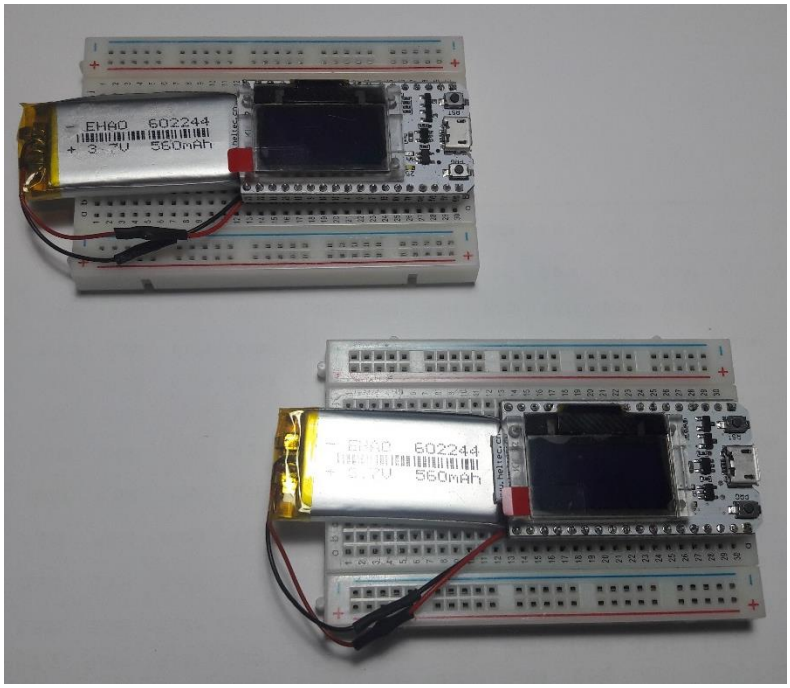


Figura 28 - Testes com placas Heltec LoRa 32

Após os testes com todos os circuitos presentes na Heltec LoRa 32 foram introduzidos circuitos periféricos, o sensor inercial MPU9250, os sensores de força e o carregador sem fios. E este último foi simples de implementar visto que, sendo um circuito comercial, já contém uma saída de 5 VDC que foi ligada diretamente no pino Vin da placa de testes. Ainda de referir que foi conseguida uma distância de 2 centímetros entre emissor e recetor.

Os sensores de forças foram usados para devolver os valores de pico (pé assente no chão) e os valores nulos (pé levantado ou descalço). Logo os valores numéricos exibidos não foram tidos em conta. Apenas é tida em conta a informação qualitativa e não quantitativa.

Por fim, foi testado o sensor inercial que, dada a sua vasta gama de aplicações, tem muitos parâmetros que são necessários configurar. Assim, começou-se por calibrar todos os sensores (acelerómetro, magnetómetro e giroscópio), aumentou-se o tempo entre aquisição dos valores do acelerómetro e validaram-se os dados devolvidos.

Todos os testes iniciais do MPU9250 foram feitos usando comunicação I2C que, apesar de muito rápida e fiável, mostrou-se menos eficaz no consumo de energia. Optou-se então pelo uso do protocolo SPI e de adormecer o MPU9250 sempre que este não está a comunicar. Apesar de usar um maior número de pinos do ESP32 foi a melhor opção.

Antes de ser possível obter valores do acelerómetro foi necessário fazer a sua calibração. Para isso, cada eixo dispõe de 2 registos onde se pode escrever o offset e o ganho do valor lido.

Para fazer a calibração pode ser usada a aceleração da gravidade sobre cada um dos eixos, assim, em cada em cada um dos eixos, quando estáticos e apontados para baixo a aceleração que se deve obter é 9.8 m/s^2. Após a calibração os valores obtidos para a calibração de cada acelerómetro estão apresentados na tabela seguinte:

	Sapato esquerdo		Sapato direito	
	Offset	Ganho	Offset	Ganho
Eixo X	-1.90	1.00	0.00	1.00
Eixo Y	21.50	1.01	0.00	1.00
Eixo Z	-0.20	0.98	0.30	0.99

Tabela 5 - Offset e ganho dos acelerómetros

Com os valores obtidos é possível concluir que houve algum defeito no fabrico durante a produção do MPU9250 aplicado no sapato esquerdo.

A Figura seguinte mostra a montagem final de todos os componentes na *breadboard.*

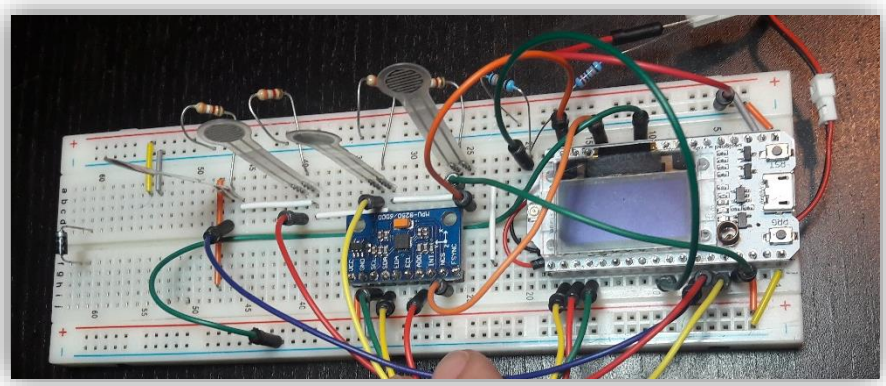


Figura 29 - Montagem final de todos os componentes

5.2. Ensaios com sandálias abertas

Foi com sandálias abertas que se realizaram todos os testes em ambiente real. As mesmas foram escolhidas tendo em conta o seguinte:

- Permitir a passagem dos cabos que interligam os sensores de pressão com a *breadboard;*
- Sola espessa para que fosse possível furar e colocar um parafuso “olhal” para que este pudesse fornecer sustentação suficiente à *breadboard* aquando dos impactos mais fortes durante a corrida;
- Tamanho 41.

A sequência de imagens da Figura 30 mostra as fases de montagem da breadboard nas sandálias.

Figura 30 - Montagem dos sapatos provisórios

Os sensores de pressão foram aplicados numa palmilha com base de cortiça fazendo quatro pequenos roços na mesma para que fosse possível passar os cabos sem serem sentidos pelos pés (Figura 31). Para que a pressão nos sensores fosse mais notada, foram adicionadas "lagrimas de silicone" por baixo dos mesmos.

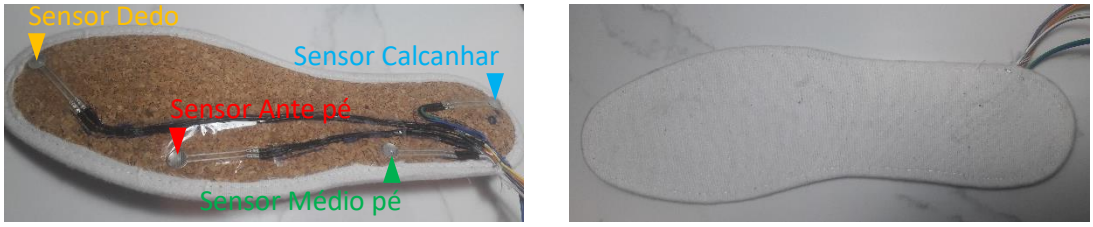


Figura 31 - Localização dos sensores na palmilha (esquerda) e palmilha vista de cima (direita)

A aquisição de dados foi feita apenas com uma sandália tendo em conta que cada um dos pés é lido e tratado de maneira independente.

Os dados foram recolhidos com auxílio de um computador portátil num ambiente real, tanto interior como exterior (Figura 32).

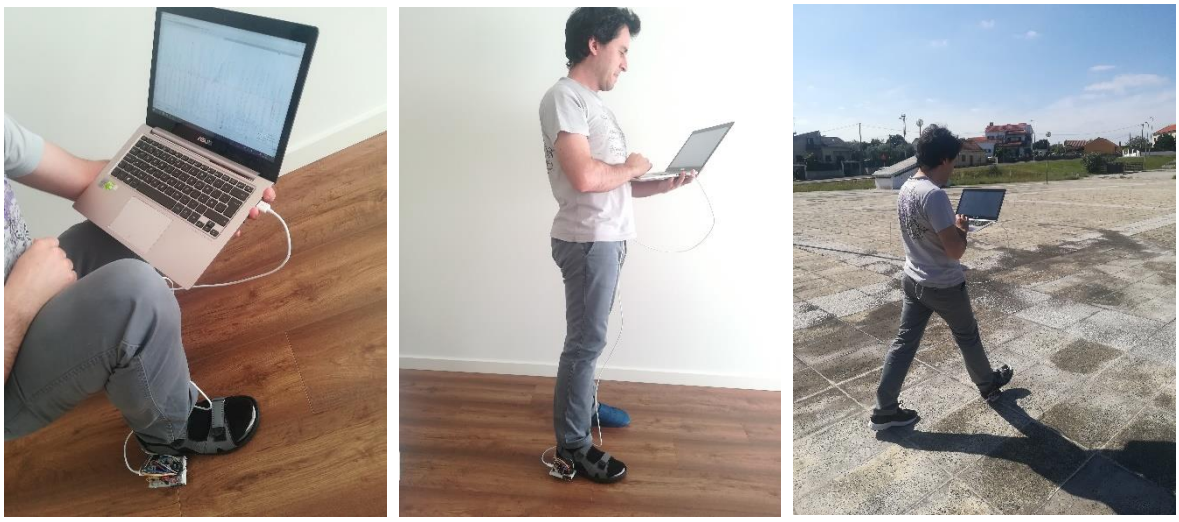


Figura 32 - Aquisição de dados em ambiente real

Toda a aquisição de dados foi feita através de comunicação em série e guardada com o software "SerialPlot v0.10". Este foi configurado para receber linhas com 7 números separados por vírgulas (Figura 33).

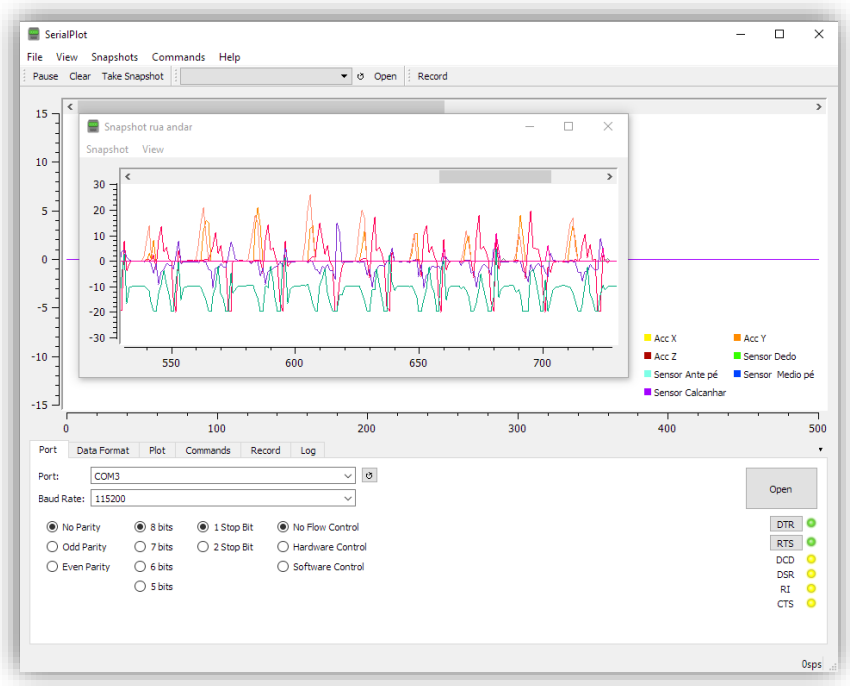


Figura 33 - SerialPlot v0.10

Para que fosse possível comunicar com o software foi criado o código da Figura 34 em que são enviados os valores obtidos em cada ciclo de leitura dos sensores, numa única linha e com um caracter "*newline*" para terminar.

Cada uma das variáveis é enviada e recebida conforme exemplo da Figura 35.

```
Serial.print(ciclo[writeIndex].aX);
Serial.print(",");
Serial.print(ciclo[writeIndex].aY);
Serial.print(",");
Serial.print(ciclo[writeIndex].aZ);
Serial.print(",");
Serial.print((long)ciclo[writeIndex].anDedo);
Serial.print(",");
Serial.print((long)ciclo[writeIndex].anAntePe);
Serial.print(",");
Serial.print((long)ciclo[writeIndex].anMedioPe);
Serial.print(",");
Serial.println((long)ciclo[writeIndex].anCalcanhar);
```

Figura 34 - Envio de variáveis através de porta serie (Arduino IDE)

```
-19.61,2.64,-19.61,0,0,0,0
-19.61,15.3,-19.61,0,0,0,17
1.01,-2.61,-19.61,0,0,1,0
-0.48,0.27,-10.41,0,20,0,0
0.28,0.55,-11.05,0,15,0,0
5.35,-0.05,-19.61,0,0,0,0
```

Figura 35 - Exemplo de informação recebida e guardada em ficheiros CSV

5.2.1. Deteção de "Andar"

A deteção de passos é considerada usando apenas os sensores de força como descrito no ponto 4.4.

O Gráfico 2 ilustra 2 segundos (entre o segundo 21 e 23) do uma aquisição de dados feita enquanto uma pessoa caminhava em terreno plano.

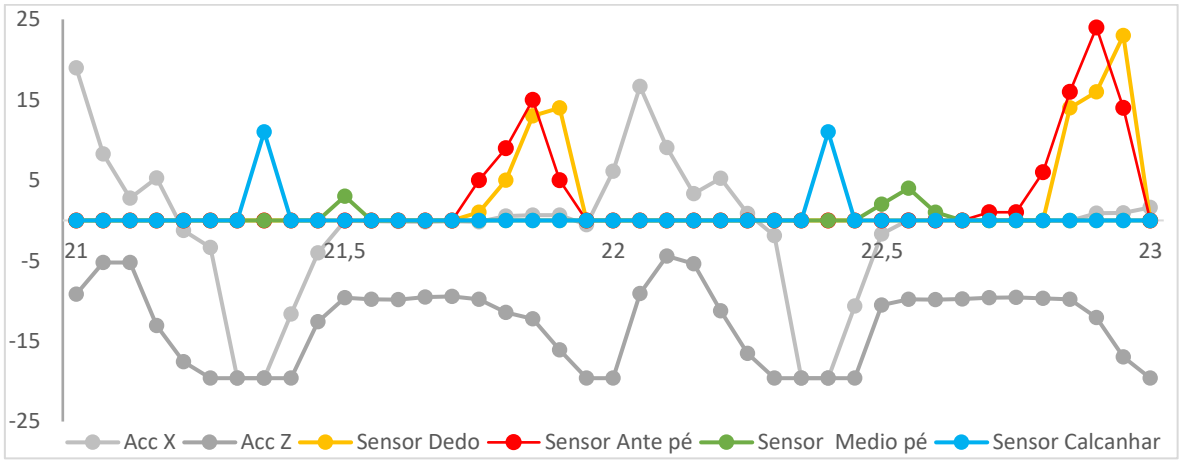


Gráfico 2 - Análise do tipo de movimento "Andar"

5.2.2. Deteção de "Correr"

A deteção de passos em corrida é considerada usando apenas os picos nos sensores de força. Aquando da corrida, o pé assenta no chão muito rapidamente fazendo com que os picos estejam muito próximos. Como a diferença entre o primeiro pico e o último é interior a 300 milissegundos é considerado que o utilizador está a correr.

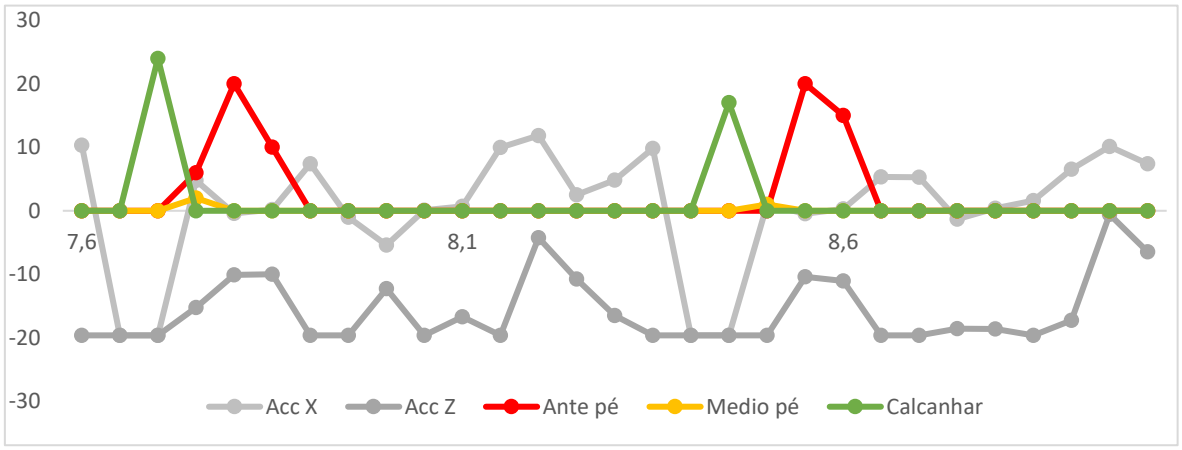


Gráfico 3 - Análise do tipo de movimento "Correr"

Pelo Gráfico 3 é também possível concluir que, na corrida, o ciclo da marcha é também mais rápido. A andar este ciclo era iniciado aproximadamente a cada segundo, enquanto na corrida é iniciado a cada 600 milissegundos. Este gráfico foi elaborado com base nos dados

adquiridos durante a corrida de uma pessoa num terreno plano, senso apenas mostrado o intervalo de tempo entre os segundos 7,6 e 9.

Quando comparados segmentos dos gráficos anteriores da caminhada e da corrida é possível ver a diferença temporal entre a deteção do primeiro e último pico dos sensores de força (Figura 1).

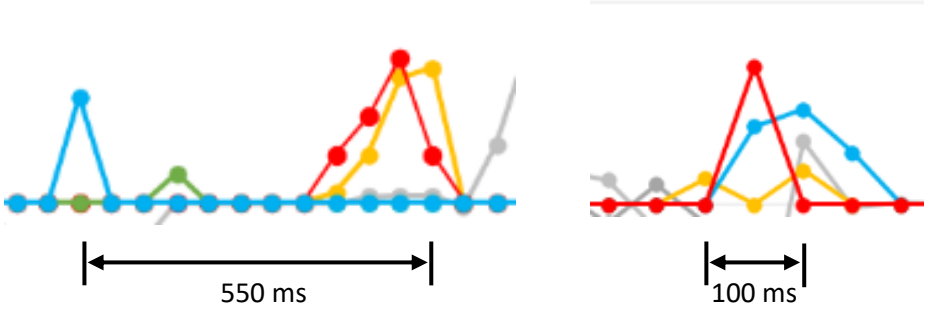


Figura 36 – Comparação de tempos entre picos dos sensores de força entre "Andar" (esquerda) e "Correr" (direita)

Em suma, quando mais rápida for a corrida, menor será o intervalo entre picos nos sensores de força e menor será o tempo do ciclo da marcha, assim, foi considerado que o movimento "Correr" é todo aquele que o ciclo da marcha tenha menos de 800ms e a diferença entre picos nos sensores de força seja inferior a 300ms.

5.2.3. Deteção de "Saltar"

O gráfico seguinte representa um salto real e mostra que entre duas amostras consecutivas, a aceleração no eixo negativo Z aumenta 18,33 m/s^2, assim, o sistema analisa os picos nos valores dos sensores de força das amostras seguintes e conclui que houve um salto. No caso representado pelos três sensores, dois no momento em que o acelerómetro ainda não tinha detetado o impacto do pé no chão e um quando o acelerómetro detetou o pico máximo negativo.

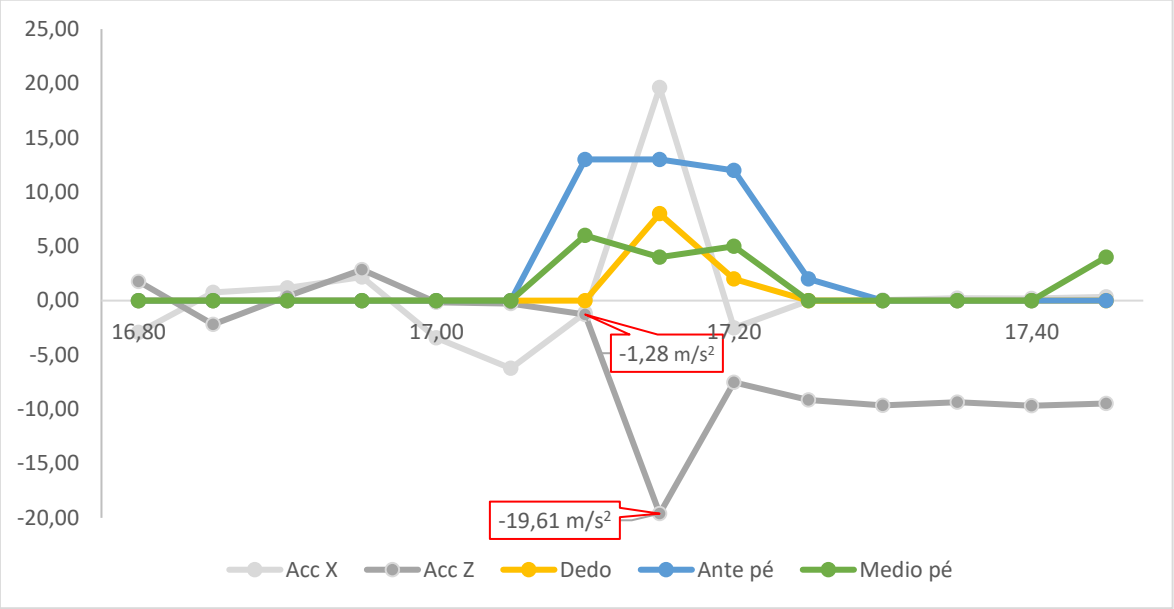


Gráfico 4 - Análise do tipo de movimento "Saltar"

Pode-se concluir que no salto descrito, o ante pé foi a parte do pé que absorveu mais impacto.

Este gráfico foi elaborado com os dados recolhidos durante um salto de uma pessoa num terreno plano elevando-se cerca de 30cm do solo.

O salto pode ser confundido com a última fase da marcha durante uma corrida, assim, caso o movimento anterior seja a corrida, o sistema nunca irá detetar um salto.

5.2.4. Deteção de “Ajoelhar”

O gráfico seguinte foi elaborado com os dados obtidos enquanto uma pessoa estava ajoelhada.

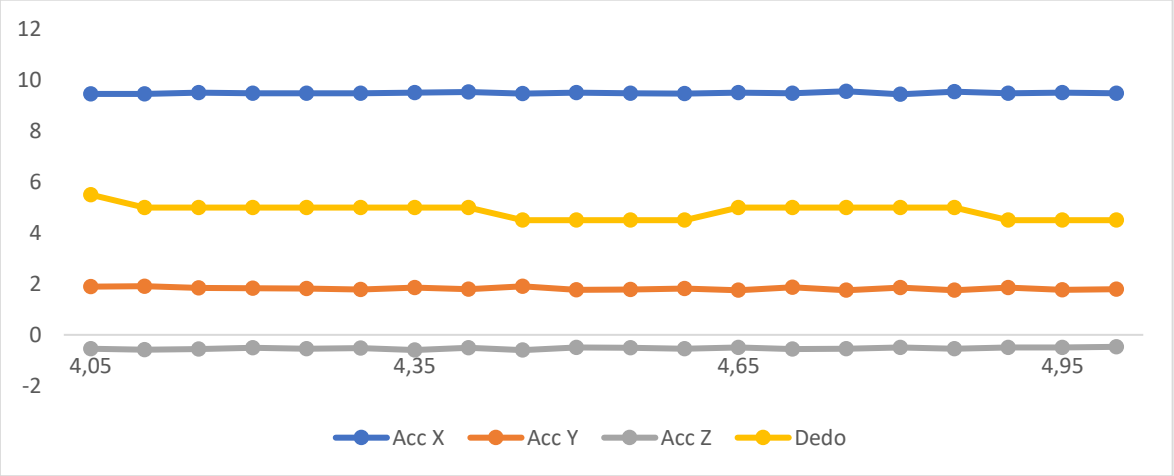


Gráfico 5 - Análise do tipo de movimento "Ajoelhar"

Pelo gráfico é possível ver que:

- O pé se encontra na vertical e a apontar para baixo, pois, a aceleração no eixo Z está próxima de zero e a aceleração da gravidade está no eixo positivo X.
- Existe uma ligeira força exercida no sensor do dedo pois os dedos estão dobrados.

A imagem seguinte representa a posição descrita bem como o gráfico apresentado:

Gráfico 6 - Pessoa ajoelhada

5.2.5. Deteção de “Deitar”

Existem 4 posições distintas para estar deitado, deitado para a frente, para trás ou para cada um dos lados.

Para que seja possível detetar cada uma das posições o algoritmo deteta o tempo que a pessoa leva entre estar na vertical e estar deitada. Nos gráficos seguintes pode ver-se que em ambos os casos houve mais de 1,5 segundos entre a pessoa passar da vertical para a horizontal.

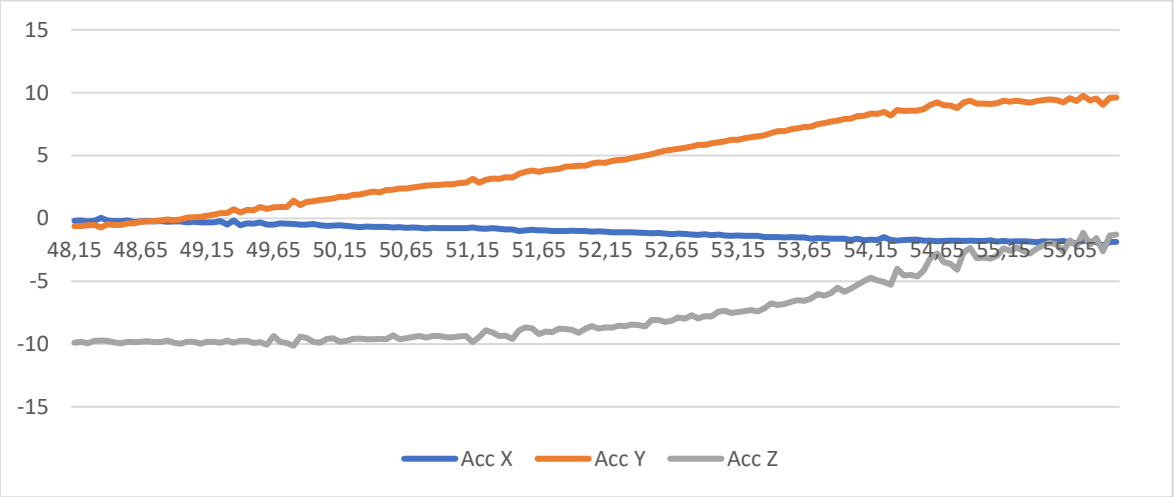


Gráfico 7 - Deitado de lado

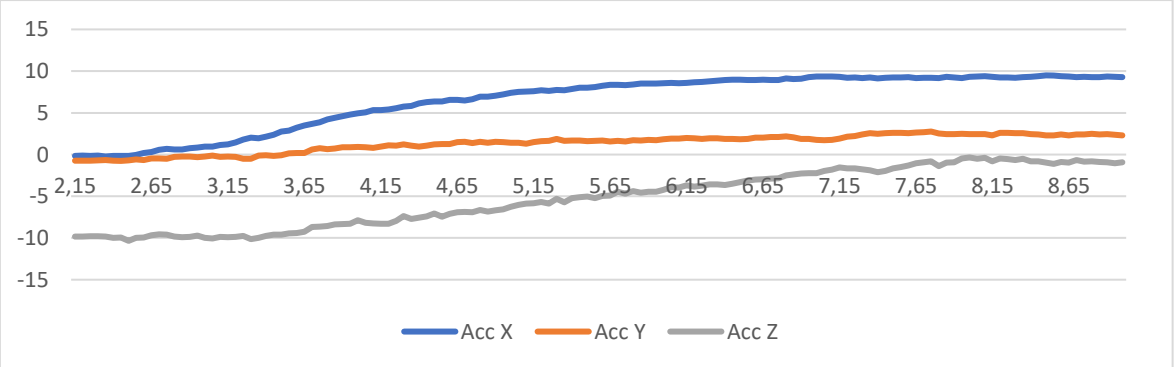


Gráfico 8 - Deitado de frente

5.2.6. Deteção de “Gatinhar”

O movimento feito a gatinhar está apresentado no Gráfico 9. Existe muita irregularidade nos valores recebidos pelo acelerómetro e uma inexistência de qualquer um dos FSR.

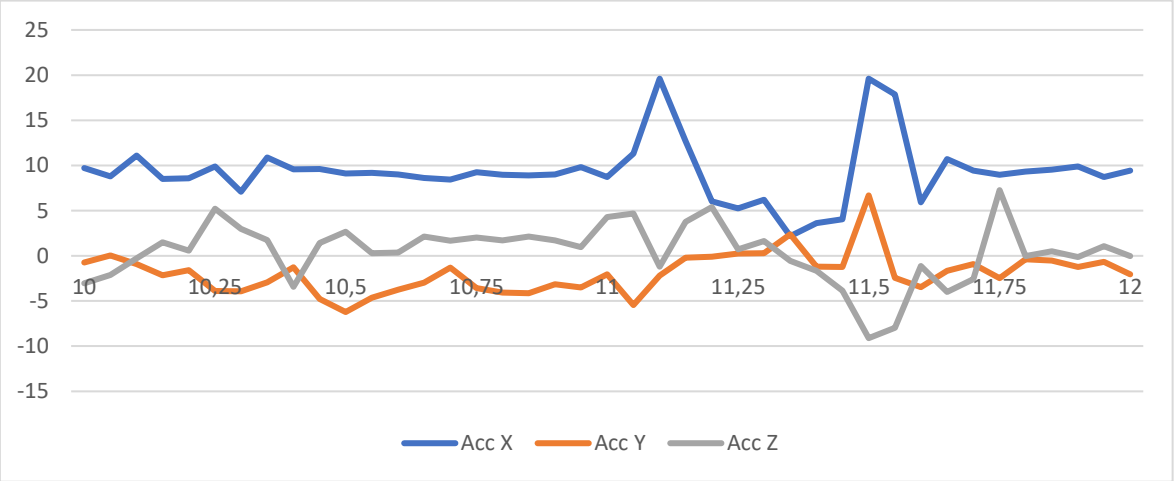


Gráfico 9 - Gatinhar

5.2.7. Deteção de “Parado”

O Gráfico 10 representa o estado “Parado”. Este estado é definido quando não há movimento e há, pelo menos, dois sensores a detetarem força.

Pelo gráfico, apenas os sensores do ante pé e do médio pé estão a detetar força. A aceleração nos eixos X e Y está praticamente nula enquanto que a aceleração da gravidade terrestre está a ser detetada no eixo Z. Para evitar erros de leitura, é considerado que a aceleração da gravidade se encontra no eixo Z até aos 9 m/s^2, que corresponde a uma inclinação de 23,3º ($\cos^{-1}(9/9{,}8)=23{,}3º$).

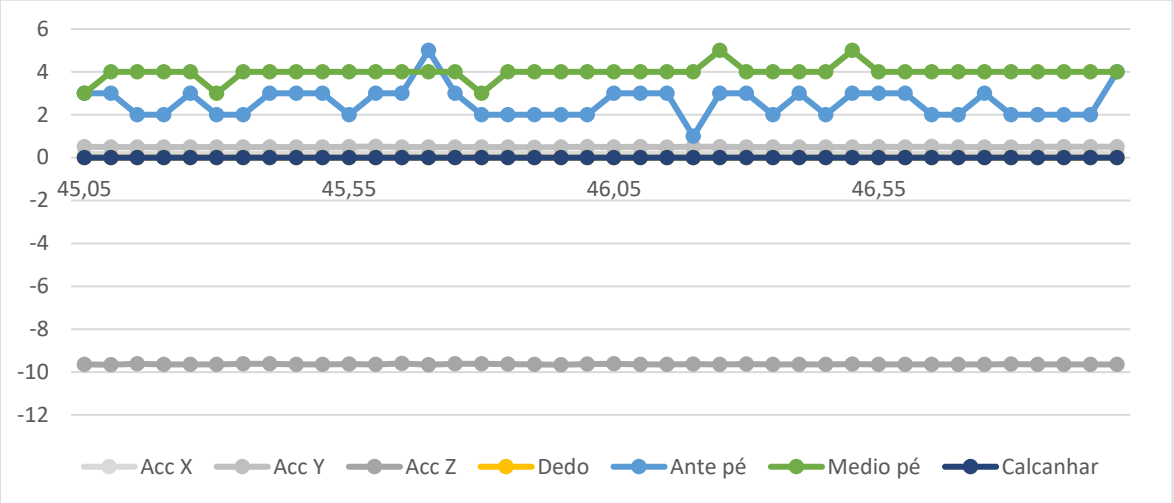


Gráfico 10 - Análise do estado "Parado"

5.2.8. Deteção de “Descalço”

Este estado define-se com uma grande irregularidade dos valores obtidos pelo acelerómetro e para que seja definido é necessário estar previamente definido o estado “parado” durante três segundos e, depois não existir força detetada nos FSR.

5.2.9. Deteção de “Queda”

Alem da deteção de quedas por incompatibilidade do movimento atual com o movimento anterior descrito na Tabela 4, existem também outras possibilidades de ser detetada uma eventual queda. É o caso da inclinação repentina, no caso da força da gravidade passa do eixo negativo Z para qualquer um dos outros eixos em menos de 1,5 segundos, é considerado uma queda.

O gráfico seguinte mostra um desses casos, é possível verificar que no segundo 20,65 a pessoa se encontrava em pé e no segundo 21,65 já se encontrava deitada de frente. Este movimento desencadeia o procedimento de queda e, no caso do outro pé também ter detetado uma inclinação similar, irão comunicar e chegar a “acordo” que realmente houve queda.

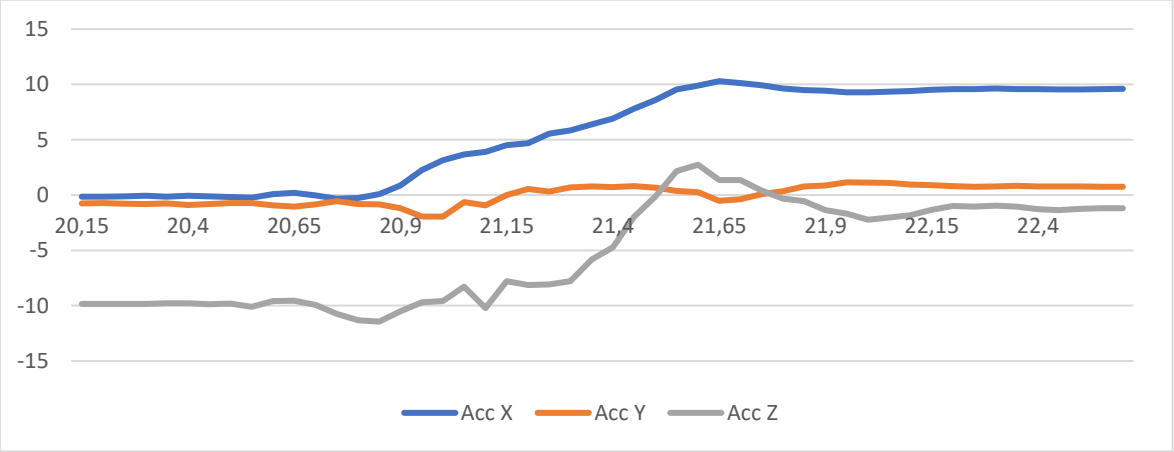


Gráfico 11 - Queda por inclinação repentina

5.2.10. Validação do sistema

Para validar o sistema de deteção de quedas foram realizadas 5 repetições de cada tipo de movimento e registou-se o movimento detetado pelo classificador. Pretende-se principalmente inferir a taxa de falsos positivos e falsos negativos na deteção de quedas.

Para a elaboração da Tabela 6 todos os movimentos, exceto a queda, foram tidos em conta apenas com a informação dada por um dos sapatos. A queda foi registada tendo em conta a validação por ambos os sapatos.

Tipo de movimento	**Tentativas**	**Detetadas**	**% de sucesso**
Andar	9	7	78
Correr	4	3	75
Saltar	10	8	80
Ajoelhar	10	8	80
Deitado	5	5	100
Parado	15	12	80
Gatinhar	5	1	20

Descalço	16	15	94
Queda	**12**	**20**	**60**

Tabela 6 - Validação do Sistema

Pela tabela é possível concluir que a queda foi detetada mais vezes do que aquelas em que o movimento foi efetivamente uma queda. Durante os ensaios alguns dos movimentos que foram detetados individualmente como queda, mas, como não foi detetada por ambos os sapatos num curto intervalo de tempo, esta não foi confirmada. Assim, a eficácia de cada sapato foi aproximadamente 60%, no entanto, a eficácia global do sistema na deteção de queda foi 100%.

O número total de tentativas para cada movimento foi reduzido. No entanto, com a continuação dos testes a taxa de sucesso poderá variar, principalmente na queda. Os restantes tipos de movimento tiveram uma percentagem de sucesso satisfatória, mas necessitam de algumas melhorias nos seus algoritmos de deteção.

A queda, quando detetada e estando dentro do edifício do IPT, foi sempre comunicada ao portal TheThingsNetwork.

A transmissão é feita e recebida como mostra a Figura 37. A mensagem é transmitida em hexadecimal e pode ser convertida em texto com base na tabela ASCII. O valor “51 75 65 64 61 21” convertido faz a palavra “Queda!”

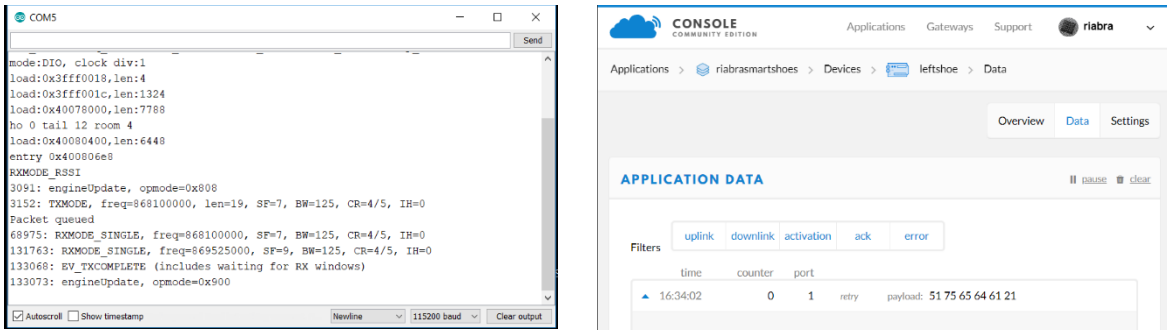


Figura 37- Informação enviada pelo ESP32 (à esquerda) e informação recebida na plataforma TheThingsNetwork (à direita)

5.2.11. Características funcionais

Com base no protótipo desenvolvido e nos resultados experimentais foram alcançadas as seguintes características funcionais:

- Tamanho:

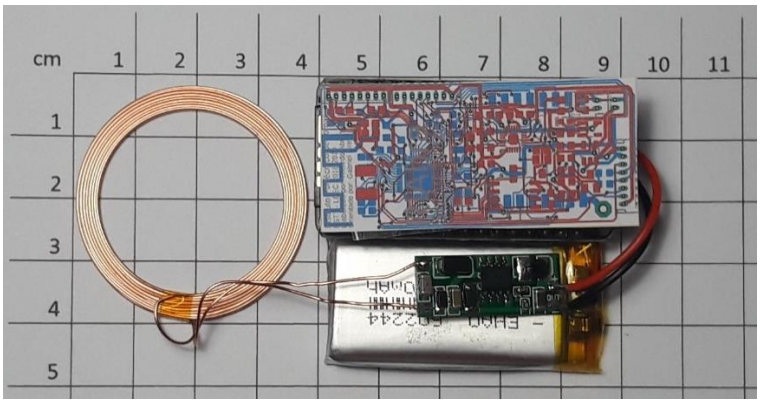


Figura 38 - Tamanho aproximado da montagem final (90mm X 50mm)

Para determinar o tamanho usou-se uma impressão em escala real da PCB final.

- Peso:

Figura 39 - Peso aproximado da montagem final (27gramas)

A PCB usada para determinar o peso total do sistema não é a PCB final. No entanto, foi usada uma PCB similar com um peso também ele semelhante.

- Autonomia:

  A autonomia máxima apenas será alcançada com a PCB final, visto que as placas de desenvolvimento usadas para os testes tinham muitos componentes que não serão usados posteriormente (conversor USB-serial, ecrã LCD, entre outros).

  Com o sistema montado como descrito no ponto 5.1 deste relatório, alcançou-se uma autonomia de 3 dias. Este é um bom indicador para que, com a montagem final, a autonomia seja superior a 5 dias, visto que o consumo durante os testes é muito superior ao consumo da PCB final.

- Custo:

  Na tabela seguinte estão presentes todos os componentes usados, respetiva quantidade, preço unitário e preço total.

  Os valores apresentados são valores para prototipagem, na produção em serie os custos baixam significativamente. Ainda assim, o custo total do Smartshoes, excluindo os sapatos, foi aproximadamente, € 167,17.

Quantidade	Tipo	Valor	Custo Unitário	Custo Total
14	Condensador	0.1uF	€ 0,31	€ 4,27
4	Resistência	100K	€ 0,36	€ 1,43
22	Resistência	10K	€ 0,36	€ 7,92
6	Condensador	10nF	€ 0,10	€ 0,59
4	Condensador	10pF	€ 0,21	€ 0,84
4	Condensador	10uF	€ 0,29	€ 1,16
4	Condensador	15pF	€ 0,14	€ 0,56

2	Resistência	1K	€ 0,31	€ 0,63
4	Condensador	1uF	€ 0,59	€ 2,38
2	Condensador	2.4pF	€ 0,57	€ 1,14
2	Bobine	2.7nH	€ 0,09	€ 0,17
2	Resistência	200R	€ 0,02	€ 0,05
2	Resistência	20K	€ 0,14	€ 0,28
2	Resistência	220K	€ 0,11	€ 0,21
2	Cristal	26MHz	€ 1,87	€ 3,74
6	Condensador	270pF	€ 0,43	€ 2,56
2	Regulador de tensão	3.3>1.8	€ 0,41	€ 0,82
2	Regulador de tensão	3.3V	€ 0,41	€ 0,82
2	Condensador	3.9pF	€ 0,57	€ 1,14
2	Cristal	32.768kHz	€ 0,98	€ 1,96
8	Resistência	3K9	€ 0,10	€ 0,80
2	Transístor	AO3401	€ 0,38	€ 0,75
2	Transístor	BSS123	€ 0,38	€ 0,75
2	Controlador	ESP32	€ 2,62	€ 5,24
2	Conector	JSTPH 1.25	€ 2,32	€ 4,64
2	Gestor de bateria	LTC4054	€ 3,19	€ 6,38
2	IMU	MPU-9250	€ 6,65	€ 13,30
2	LoRa	RFM95	€ 4,90	€ 9,80
2	Memoria flash	W25Q32FV	€ 1,35	€ 2,70
2	Díodo	iN5819	€ 0,23	€ 0,45
8	Sensor de força	FSR	€ 6,07	€ 48,56
2	PCB	PCB	€ 5,00	€ 10,00
2	Fio de silicone	-	€ 4,17	€ 8,34
2	ANTENNA	-	€ 2,40	€ 4,80
2	Wireless charger	-	€ 6,00	€ 12,00
2	Bateria	-	€ 3,00	€ 6,00

Tabela 7 - Lista de componentes e preço de custo

6. Conclusões e trabalho futuro

Neste projeto foi desenvolvido um protótipo de um sapato capaz de fazer a deteção automática e alerta de quedas, o qual poderá ser usado por idosos e assim permitir um auxílio mais rápido.

Na vertente de hardware do trabalho desenvolvido destaca-se a utilização de componentes amplamente utilizados que faz com que seja possível criar uma solução comercial a preços competitivos. No software desenvolvido, a deteção de queda por incompatibilidade dos movimentos anterior e atual trouxe um grande desafio, mas, nos testes iniciais revelou-se eficaz. No entanto, para uma validação efetiva, o sistema terá de ser testado por mais pessoas e pelos potenciais utilizadores, os seniores.

A comunicação LoRa permitiu que fossem feitas comunicações de curto e longo alcance com uma mínima quantidade de energia. Todos os conhecimentos adquiridos poderão ser melhorados ou replicados a outros sistemas que permitam uma melhor qualidade de vida dos idosos.

No futuro poderão ser adicionados módulos que permitam que o Smartshoes seja ainda mais autónomo em termos de comunicações e gestão de bateria. Assim, para trabalho futuro propõem-se:

- Adicionar um módulo de posicionamento GPS e um módulo de comunicações GSM para que seja possível comunicar a emergência por SMS e na mesma comunicação fornecer a localização do utilizador.
- Implementar uma bateria auto recarregável com o movimento natural do pé e assim reduzir ou anular as necessidades de carregamento na sapateira. Usando, por exemplo, uma Livecell [26].
- Desenvolvimento uma plataforma de comunicação de emergências usando o protocolo LoRaWAN.

BIBLIOGRAFIA

[1] INE, “Estimativas de População Residente em Portugal - 2018,” 14 06 2019. [Online]. Available: https://www.ine.pt/ngt_server/attachfileu.jsp?look_parentBoui=377985255&att_display=n&att_download=y. [Acedido em 27 06 2019].

[2] SNS, “Tropeções, quedas e trambolhões,” SNS, 19 12 2017. [Online]. Available: https://biblioteca.min-saude.pt/livro/. [Acedido em 27 06 2019].

[3] “Home Fall Detection Medical Alert,” [Online]. Available: https://www.alert-1.com/content/fall-detection-technology/1390. [Acedido em 09 06 2019].

[4] Apple, “Utilizar a deteção de queda com o Apple Watch Series 4,” Apple, 05 04 2019. [Online]. Available: https://support.apple.com/pt-pt/HT208944. [Acedido em 28 06 2019].

[5] Garmin, “New Safety and Tracking Features Now Available on Select Garmin Watches,” 07 05 2019. [Online]. Available: https://www.garmin.com/en-US/blog/featured-2/safety-and-tracking-features/. [Acedido em 28 06 2019].

[6] “Medical Care Alert,” [Online]. Available: https://www.medicalcarealert.com/Fall-Detection-Medical-Alert-System-s/1850.htm. [Acedido em 09 06 2019].

[7] “Life Call,” [Online]. Available: http://www.lifecall.com/products/#Advanced. [Acedido em 09 06 2019].

[8] “Medical Guardian,” [Online]. Available: https://www.medicalguardian.com/products/classic-guardian. [Acedido em 09 06 2019].

[9] A48, “Sistema de Apoio ao Cliente - GLOBALHELP 1,” [Online]. Available: http://www.a48.pt/sistemaapoioclienteglobalhelp1. [Acedido em 27 06 2019].

[10] Puma, “PUMA REISSUES THE RS-COMPUTER SHOE,” Puma, 10 12 2018. [Online]. Available: https://about.puma.com/en/newsroom/brand-and-product-news/2018/2018-12-10-puma-reissues-the-rs-computer-shoe. [Acedido em 09 06 2019].

[11] “E-vone presentation video,” e-vone, [Online]. Available: http://www.e-vone.com/our-shoes/?lang=en. [Acedido em 27 06 2019].

[12] “Smart shoes: Tracking fitness through your feet,” Gadgets & Wareables, 01 04 2019. [Online]. Available: https://gadgetsandwearables.com/2019/04/01/trackers-feet/. [Acedido em 10 06 2019].

[13] “ESP32 Resources,” Espressif systems, 10 04 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[14] ESPRESSIF, “Programming ULP coprocessor using C macros,” [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/ulp_macros.html. [Acedido em 27 06 2019].

[15] D. Hanson e C. Fraser, “The lcc retargetable ANSI C compiler,” [Online]. Available: https://github.com/jasonful/lcc. [Acedido em 27 06 2019].

[16] I. Sense, “MPU9250,” Inven Sense, [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/.

[17] T. Clausen Aloÿs Augustin, J. Yi e W. M. Townsley, “A study of lora: Long range and low power networks for the internet of things,” 09 09 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/9/1466/htm.

[18] A. Raj, “Interfacing SX1278 (Ra-02) LoRa Module with Arduino,” 02 2019. [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-lora-sx1278-interfacing-tutorial. [Acedido em 03 05 2019].

[19] COMISSÃO EUROPEIA, “DECISÃO DE EXECUÇÃO (UE) 2017/ 1483 DA COMISSÃO - de 8 de agosto de 2017,” 18 08 2017. [Online]. Available: https://eur-lex.europa.eu/legal-content/PT/TXT/PDF/?uri=CELEX:32017D1483&from=EN. [Acedido em 27 06 2019].

[20] Semtech, “Why LoRa?,” Semtech, [Online]. Available: https://www.semtech.com/lora/why-lora. [Acedido em 22 .5 2019].

[21] “The Things Network,” [Online]. Available: https://www.thethingsnetwork.org/. [Acedido em 26 06 2019].

[22] “How FSR Works,” Tangio Printed Electronics, [Online]. Available: https://tangio.co/pages/how-it-works. [Acedido em 26 06 2019].

[23] “High Accuracy 10g-1kg Pressure Sensor Smart Flexible Thin Film Force Sensor,” [Online]. Available: https://www.ebay.com/itm/High-Accuracy-10g-1kg-Pressure-Sensor-Smart-Flexible-Thin-Film-Force-Sensor/223422889770?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2057872.m2749.l2649. [Acedido em 05 05 2019].

[24] “How to use SmartConfig on Arduino ESP32,” 05 2017. [Online]. Available: http://www.iotsharing.com/2017/05/how-to-use-smartconfig-on-esp32.html.

[25] “O Funcionamento do NTP,” [Online]. Available: https://ntp.br/ntp.php#O_Funcionamento_do_NTP.

[26] S. Biswas, “Livecell,” Saikat Biswas, [Online]. Available: http://www.saikatbiswas.com/web/Projects/Livecell.htm. [Acedido em 27 06 2019].

[27] [Online]. Available: https://www.treinoemfoco.com.br/qualificando-seu-treino/cinesiologia-biomecanica-da-corrida/.

[28] "Fall Detection – Fall Alert Saves Lives," Deskshare, Inc, [Online]. Available: https://play.google.com/store/apps/details?id=com.fall_detection&hl=en_US. [Acedido em 09 06 2019].

ANEXOS

Anexo 1 – Esquema ESP32 para PCB

Anexo 2 – Esquema de periféricos para PCB

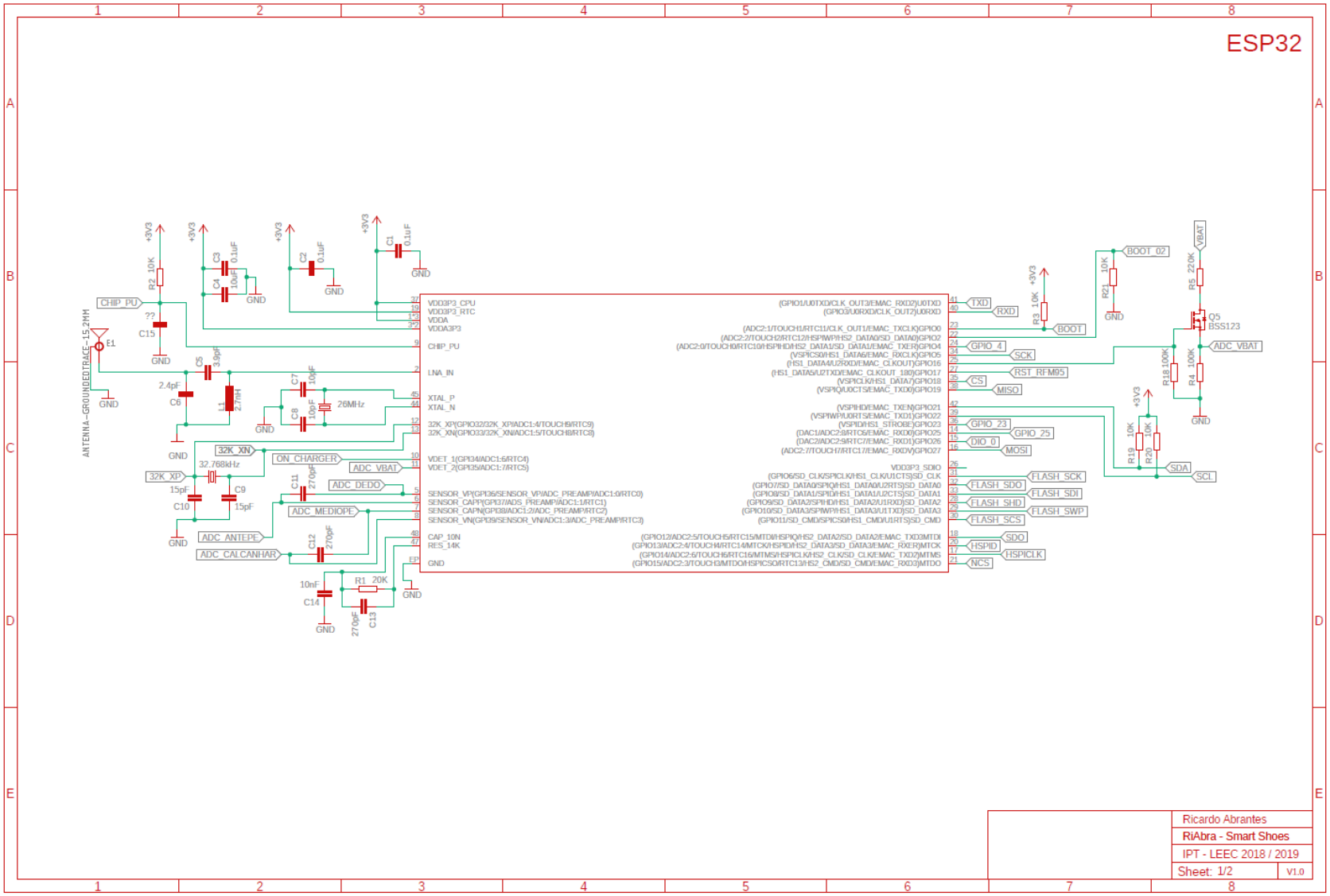
Anexo 3 – *Layout* Geral PCB

Anexo 4 – *Layout* PCB, Bottom layer

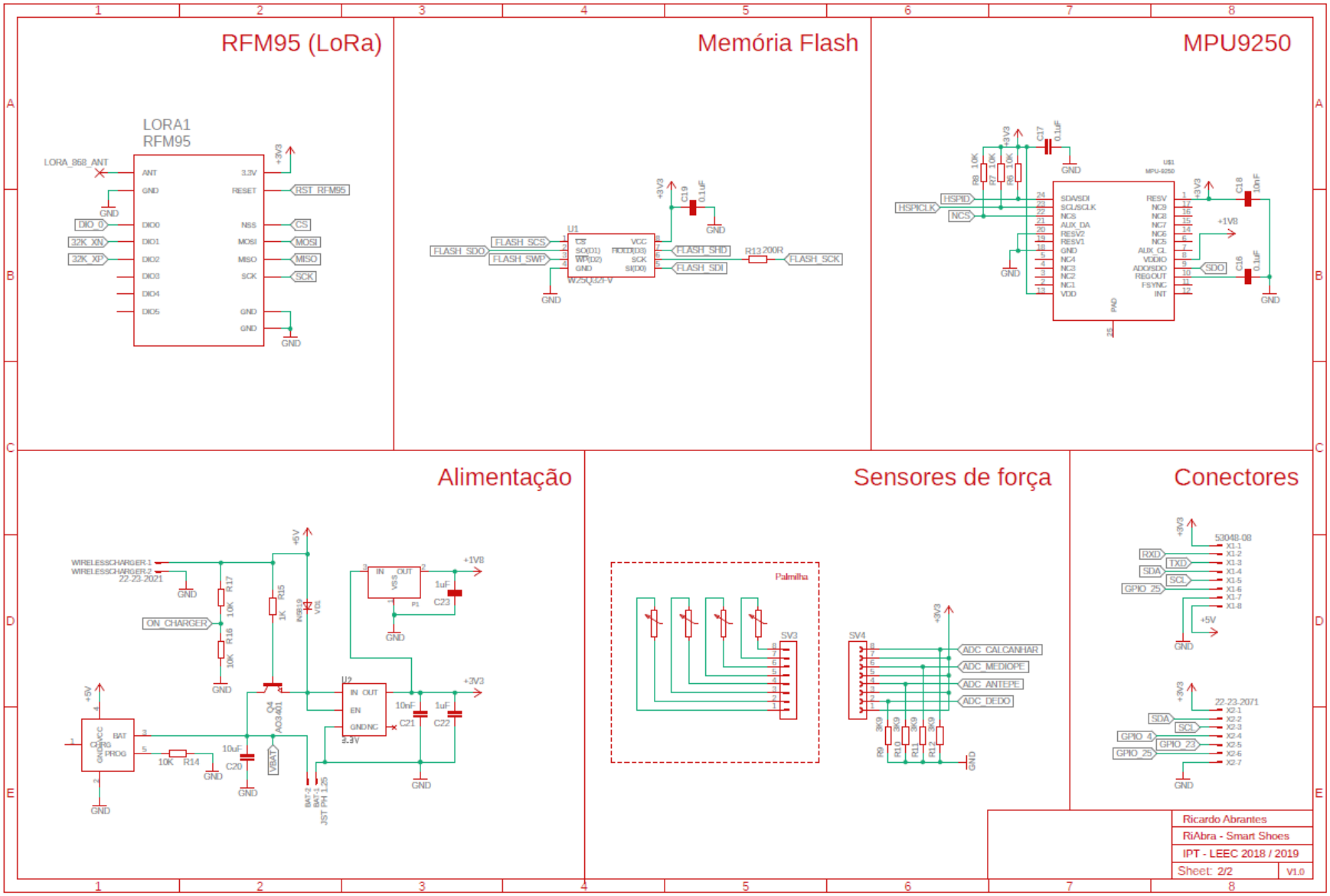
Anexo 5 – *Layout* PCB, Top layer

Anexo 6 – *Layout* PCB, Pads + Vias

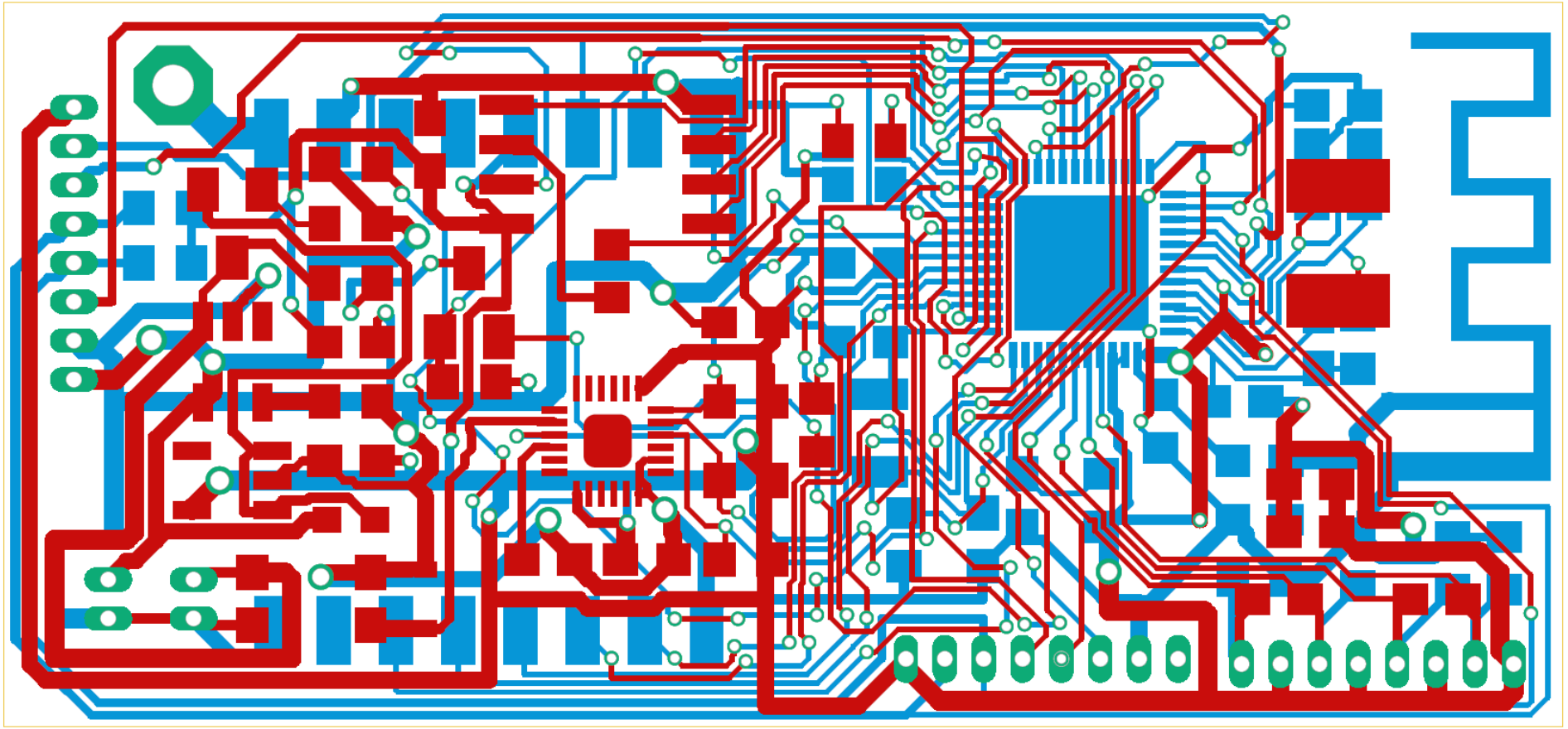
Anexo 1 – Esquema ESP32 para PCB



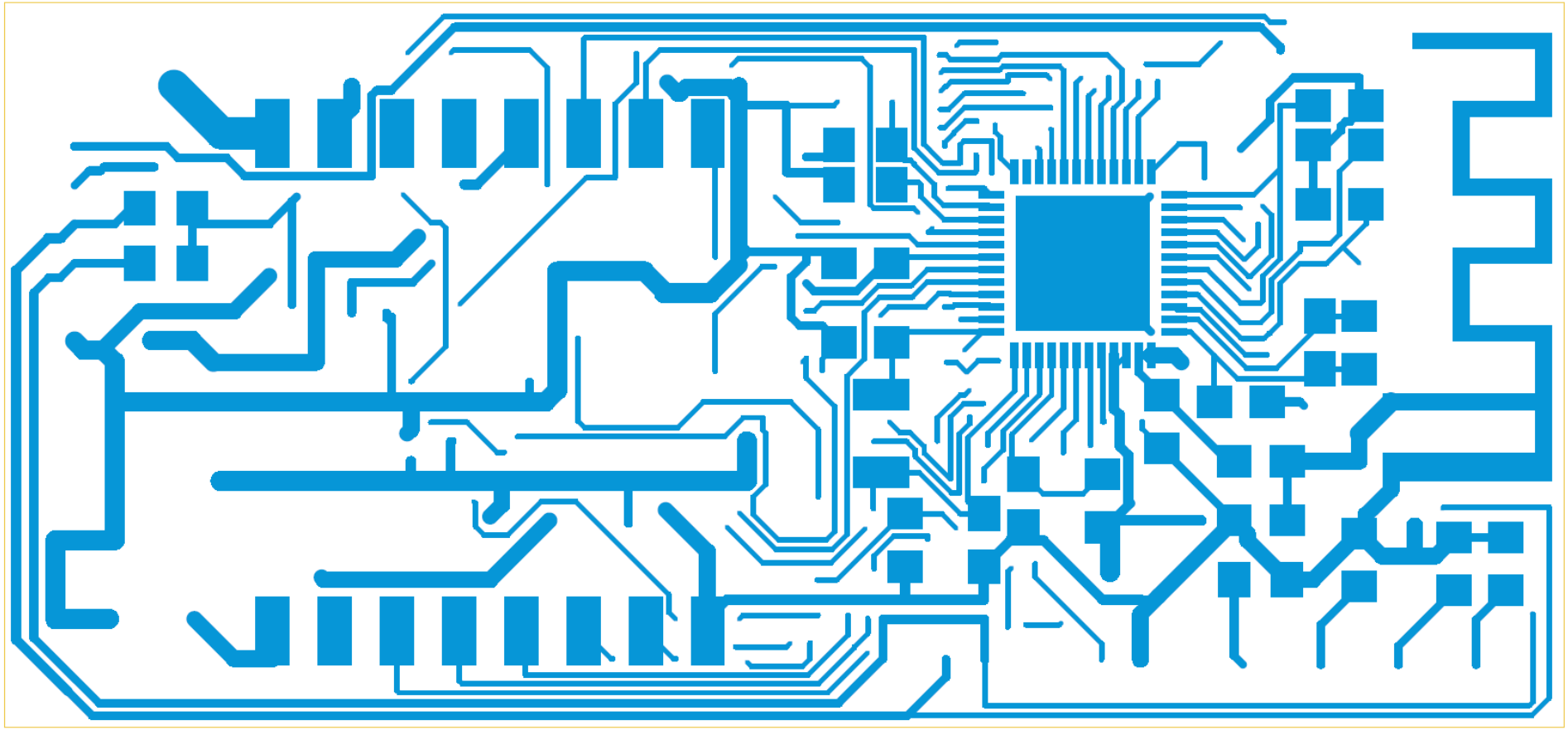
Anexo 2 – Esquema de periféricos para PCB



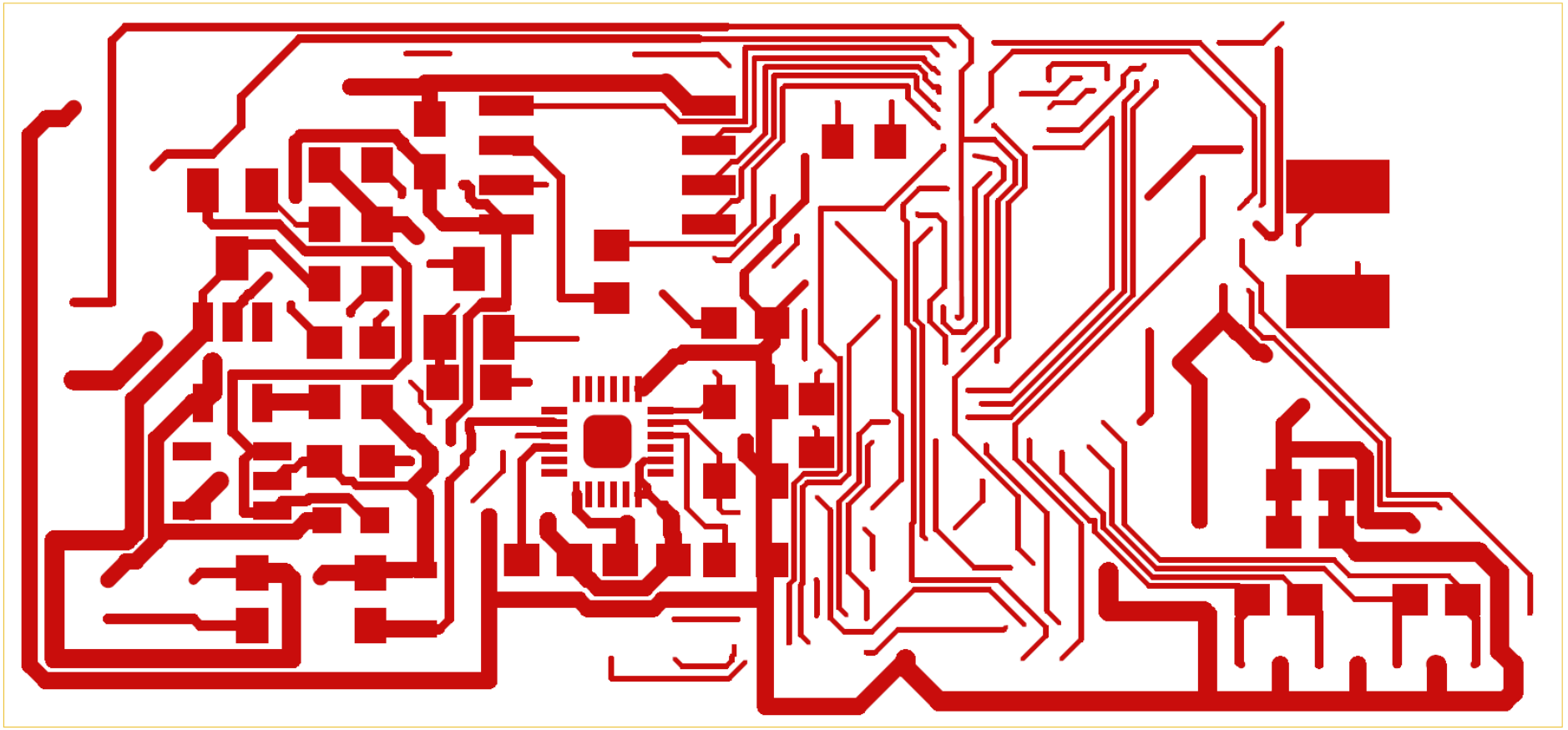
Anexo 3 – Layout Geral PCB



Anexo 4 – Layout PCB, Bottom layer



Anexo 5 – Layout PCB, Top layer



Anexo 6 – Layout PCB, Pads + Vias