Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Pedro José Marques Antunes

Activity @ Home

Projeto

Orientado por:

Prof. Gabriel Pereira Pires – Instituto Politécnico de Tomar
Prof. Pedro Correia – 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

DEDICATÓRIA

Dedico este Projeto aos meus pais…

RESUMO

O projeto Activity@Home (A@H) tem por objetivo o desenvolvimento de sistemas de monitorização remota de atividade doméstica em casa, com o intuito de detetar situações de alerta, ou caraterizar o tipo de atividade realizado por pessoas. Estes sistemas podem contribuir para que pessoas idosas ou com mobilidade reduzida possam viver nas suas casas o máximo de tempo possível. Nesse sentido foram desenvolvidos dois subsistemas preliminares para prova de conceito:

1) Um botão de emergência, o qual permite a transmissão remota do sinal de alarme. Esse botão de emergência (colocado num colar) utiliza um microcontrolador incorporado num módulo de comunicação.
2) Uma pulseira que integra o mesmo módulo de comunicação do subsistema anterior e um sensor IMU (unidade de medição de inercia), para realizar dois tipos de tarefas: 1) o controlo contínuo de um dispositivo (para prova de conceito, testou-se o controlo de velocidade de uma ventoinha que mantém uma bola em suspensão); e 2) o reconhecimento de gestos baseados no movimento do pulso, o qual obrigou a um estudo exploratório de *Machine Learning* para classificação de gestos.

Palavras-chave: Botão de Emergência, ESP8266-01, WiFi, IMU, Ventoinha, Reconhecimento de Gestos, Machine Learning.

ABSTRACT

The project Activity@Home aims to develop a remote monitoring system of activity at home, with the objective to detect emergencies, and classify the type of activity. These systems can help the elderly and the movement impaired to remain at home for as long as possible. For this purpose, two subsystems were developed:

1) An Emergency button embedded into a pendent, that uses a microcontroller coupled with a communication module.
2) A wristband that uses the same module the emergency button did as well as an IMU (Inertia Measurement Unit), to accomplish two tasks, one of continuous control of an equipment (for proof of concept the control of a fan attached to a tube that keeps a ball mid-air), and gesture recognition based on the movement of the wrist, for this an exploratory study of Machine Learning Classification was done.

Keywords: Emergency Button, ESP8266-01, WiFi, Fan, Gesture Recognition, Machine Learning, IMU

AGRADECIMENTOS

Seguem os agradecimentos às pessoas que me auxiliaram neste projeto bem como aos que tornaram possível.

Aos meus pais pela força, paciência e por estarem presentes ao longo desta jornada.

Aos meus colegas presentes no laboratório Vita.ipt pela ajuda na montagem de equipamentos, desenvolvimento de algoritmos e apoio geral.

Aos orientadores do projeto, o Prof. Gabriel Pires e o Prof. Pedro Correia pela ajuda fornecida e pela oportunidade de realizar este projeto.

Ao Eng. Pedro Neves pela prontidão no material pedido, ajuda constante e apoio ao longo deste último ano.

Ao Instituto Politécnico de Tomar (IPT) e ao laboratório VITA.ipt (Vida assistida por ambientes inteligentes) pelas condições proporcionadas para a realização do projeto. Este trabalho teve o financiamento e enquadramento dos projetos IC&DT:

- VITASENIOR-MT, Ref. CENTRO-01-0145-FEDER-023659 com fundos do FEDER através dos programas operacionais CENTRO2020 e FCT;
- MOVIDA, Ref. CENTRO-01-0145-FEDER-023878 com fundos do FEDER através dos programas operacionais CENTRO2020 e FCT;

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

1.1. Motivação e Contexto

O projeto Activity@Home (A@H) tem por objetivo a monitorização remota de situações de alerta (por exemplo, acionamento de botão de emergência em situação de queda) e a deteção do tipo de atividade/gestos realizados por pessoas no seu dia-a-dia. Pretende-se que os dispositivos sejam *wearable*.

O projeto está integrado nas atividades dos projetos de I&D VITASENIOR-MT e MOVIDA, os quais têm por objetivos, respetivamente, o desenvolvimento de uma solução tecnológica de tele-saúde, especificamente dirigida a pessoas idosas que vivam sozinhas; e a monitorização de atividade física em casa.

1.2. Objetivos e Contribuições

Uma vez que o público alvo são pessoas com mobilidade reduzida ou idosos, existe uma grande possibilidade de haver uma ocorrência que as impossibilite de se movimentar e pedir ajuda, quando estas se encontram sozinhas.

Com isto em mente foi desenvolvido um botão de emergência que quando pressionado envia uma mensagem de pedido de auxílio a uma entidade que o possa fornecer, nomeadamente o cuidador (Figura 1).

O sistema de botão de emergência está descrito na Figura 1. O evento é desencadeado quando o botão é pressionado na ocorrência de uma emergência. Este botão está conectado a um microcontrolador com capacidade de comunicação através de um módulo de WiFi. O microcontrolador comunica com um servidor na Internet, enviando uma mensagem de alerta que é encaminhada para o cuidador.

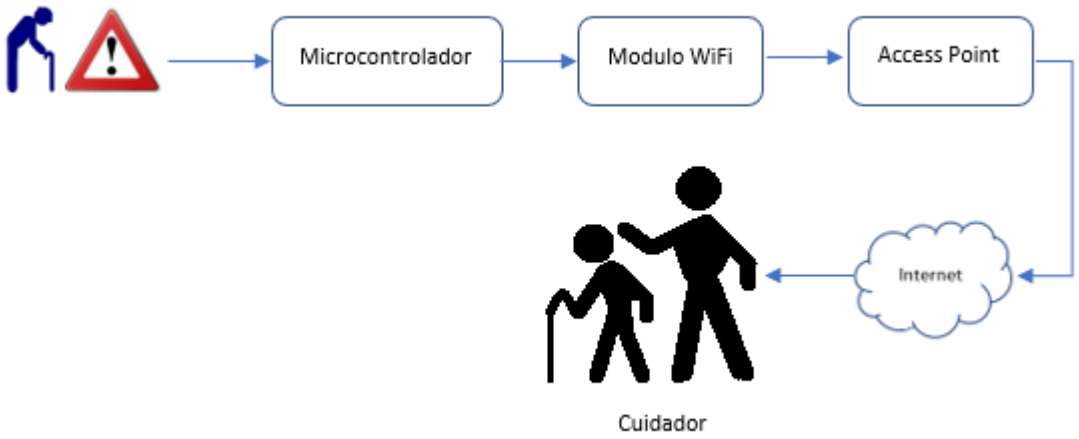

Figura 1- Arquitetura geral de Botão de Emergência

No sentido de explorar o conceito de monitorização de atividade física/tipo de atividade baseada em sensores inerciais, desenvolveu-se um framework para explorar este conceito. Foram realizados dois módulos: 1) deteção de movimentos contínuos do pulso para controlo de um dispositivo; e 2) módulo de reconhecimento de 2 gestos. Ambos os sistemas usam um sensor inercial (IMU) no pulso. Foram realizados dois algoritmos:

- Um algoritmo que que realiza o controlo contínuo de velocidade de um motor de uma ventoinha cuja arquitetura geral se encontra ilustrada na Figura 2. O microcontrolador está constantemente a receber dados de um sensor inercial, e envia-os através de um modulo WIFI para o sistema de controlo da ventoinha, o qual se encontra na mesma rede local. O sistema de controlo da ventoinha possui um microcontrolador que recebe os comandos e transforma-os em sinais PWM para variação de velocidade do motor DC da ventoinha.

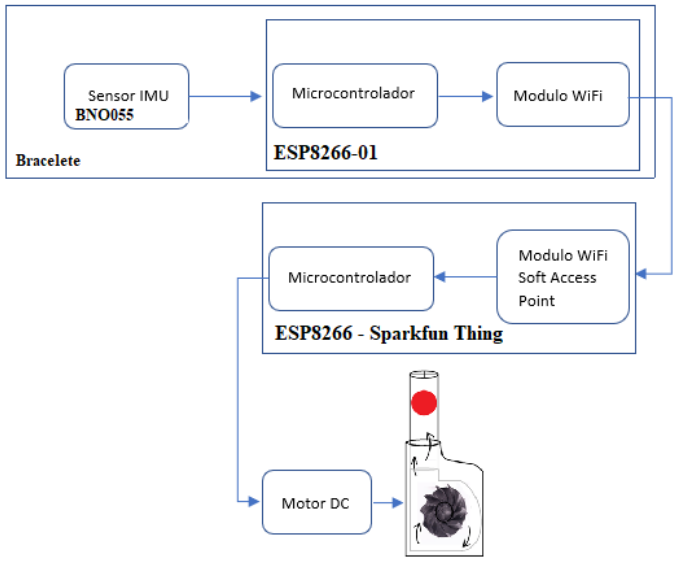

Figura 2- Arquitetura Geral Controlo Continuo

- Um algoritmo que classifica 2 movimentos específicos pré-estabelecidos e um 3º estado que pode ser qualquer outro movimento. O modelo da Figura 3 ilustra este subsistema, o qual tem por base o *hardware* do sistema anterior, sendo agora os dados fornecidos pelo sensor enviados para um computador conectado à mesma rede local. Este computador extrai características dos dados inerciais que alimenta um algoritmo de Machine Learning. O algoritmo é treinado e posteriormente passa a realizar a classificação dos gestos treinados em tempo real.

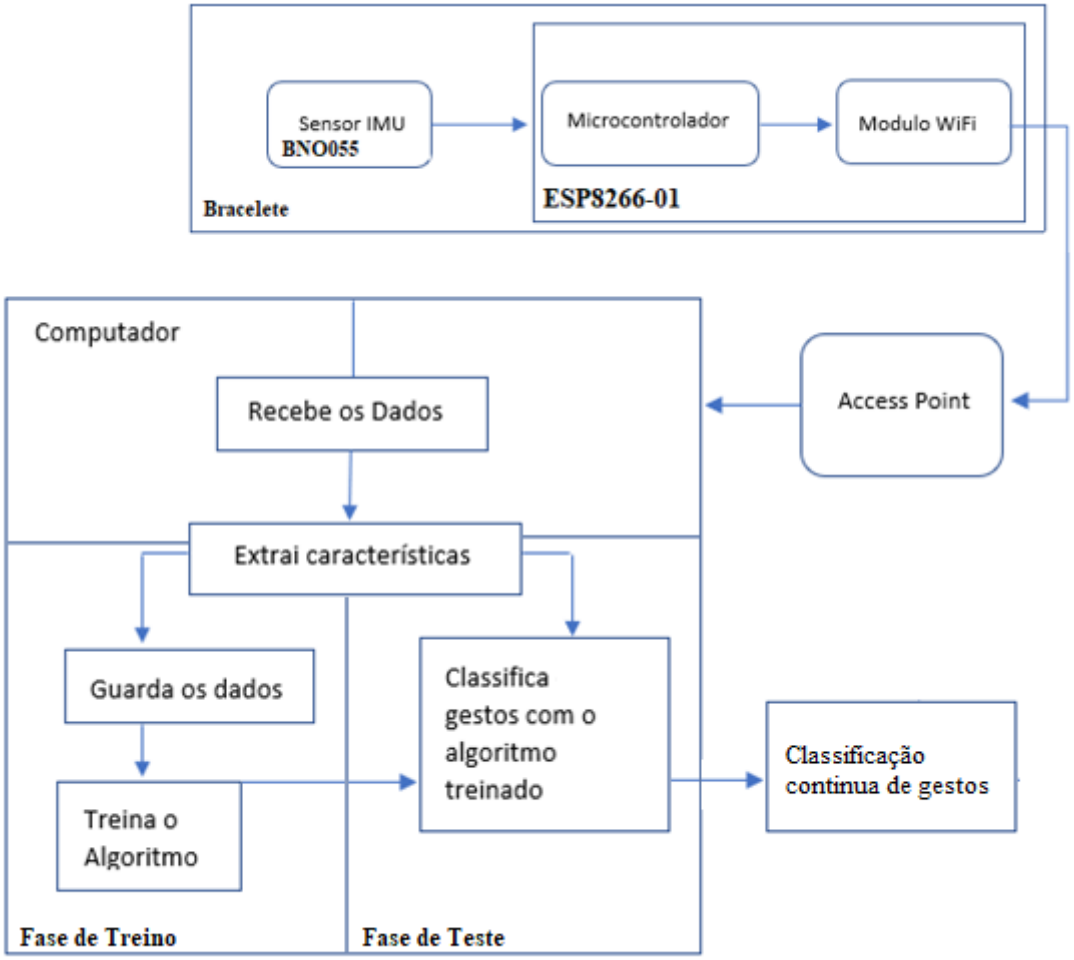

Figura 3 - Arquitetura Geral de Classificação de Gestos

1.3 – Organização do relatório

O relatório está subdividido nos seguintes capítulos:

- **Introdução**

No capítulo de introdução foram apresentados sucintamente os objetivos do projeto desenvolvido e o seu enquadramento

- **Estado da Arte e Contribuições**

Neste capítulo é apresentado o estado da arte respeitante aos temas abordados no projeto, nomeadamente botões de emergência e sistemas de reconhecimento de gestos, e apresentam-se também as contribuições do projeto A@H.

- **Tecnologias**

O capítulo de tecnologias usadas é referente aos equipamentos utilizados para a realização do projeto bem como as suas características.

- **Sistema A@H**

O capítulo de Sistema A@H deste relatório detalha o processo realizado para implementar as tarefas propostas:

- Botão de Emergência – Activity @Home
- Reconhecimento de Gestos – Controlo continuo - Activity @Home
- Reconhecimento de Gestos – Classificação de Gestos - Activity @Home

Para cada uma destas tarefas é também explicada a metodologia dos algoritmos utilizados e a implementação destes.

- **Protótipos finais e Conclusões**

Este capítulo fornece uma descrição sobre os protótipos finais obtidos para cada tarefa proposta, bem como comentários do autor sobre como melhorar estes sistemas futuramente.

2. Estado da Arte e Contribuições

Neste capítulo é apresentado o estado da arte no que diz respeito a sistemas semelhantes aos desenvolvidos no projeto A@H, estando então subdividido em Botões de Emergência e Reconhecimento de Gestos.

Devido ao ascendente mercado de equipamentos de assistência à vida no domicílio, o preço dos componentes que os constituem tem descido drasticamente. O mercado tem evoluído de modo a que o tamanho destes componentes diminua ao ponto que os equipamentos sejam embutidos na roupa ou adereços.

2.1 – Botões de Emergência

Em [1] os autores realizaram um estudo envolvendo pessoas idosas que estiveram envolvidas em situações de queda nas suas casas. Através desse estudo chegaram à conclusão que podiam dividir os idosos que tiveram quedas, em quatro diferentes grupos: i) os que possuíam alarmes pessoais de emergência e os utilizaram com eficácia, recomendando então a tecnologia; ii) os que possuíam alarmes mas não os conseguiram usar com eficácia aquando da queda; iii) os que não possuem alarmes devido a questões monetárias; iv) os que não possuem alarme de emergência pessoal por escolha

Em relação aos alertas de emergência, existe no mercado um variado leque de opções. Em [1] é apresentado um modelo mais convencional de botões de emergência, que fornece dois pendentes e uma unidade central. Os pendentes comunicam com a unidade central sem fios até um alcance de 30 metros de distância, e informam o cuidador da ocorrência através de sinais sonoros e luminosos, sendo possível distinguir entre dois tipos de pedido de auxílio diferenciando de uma situação urgente e não urgente. Este equipamento fornece também na unidade central uma luz de presença noturna.

Figura 4- Home Safety Alert Panic Alarm Pendant - WARM WHITE LIGHT 139514001 [1]

Em [2] temos o exemplo de um botão de emergência que fornece ao idoso e ao cuidador a oportunidade de estarem em contacto aquando da emergência. Este equipamento é composto por uma pulseira, um pendente e uma unidade central que está ligada à rede telefónica doméstica. Através de comunicação WiFi os pendentes criam uma chamada telefónica entre o idoso e o cuidador.

Figura 5- SureSafe Alarms – 1 Pendant & 1 Wristwatch [2]

O equipamento descrito em [3] fornece também a capacidade de realizar chamadas. No entanto este equipamento não utiliza como meio de transmissão a linha telefónica doméstica mas sim um cartão SIM que se conecta à rede com melhor sinal dos maiores operadores disponíveis. Associado ao serviço existe uma equipa que toma conta da ocorrência e encaminha a resposta adequada a situação. Esta equipa recebe também informação da localização do idoso através de GPS. Este equipamento destaca-se pela capacidade de não estar confinado num espaço de utilização. No entanto estes equipamentos que possuem capacidades de comunicação com o cuidador do idoso no momento da ocorrência, são acompanhados de uma mensalidade de serviço tendo um custo adicional e sendo incomportáveis para pessoas com poucas capacidades monetárias.

Figura 6- SureSafeGO 24/7 Connect 'Anywhere' Alarm - [3]

2.2 -Reconhecimento de Gestos

O reconhecimento gestual está interligado com três diferentes áreas, a de interação humano máquina (HMI), interação humano robô (HRI) e robótica de assistência social (SAR) [4].

No âmbito do reconhecimento de gestos, tem havido um crescimento muito grande nos últimos anos, sendo, no entanto, sistemas maioritariamente baseados em visão computacional, um ramo multidisciplinar que trata do reconhecimento computacional de imagens com o objetivo de simular o sistema visual humano.

Convencionalmente este reconhecimento gestual é realizado através da análise de imagens 2D, no entanto existem problemas com esta abordagem uma vez que estas imagens não podem estar sobre luz constante, objetos de fundo podem dificultar o reconhecimento dos gestos sendo também difícil discernir orientação dos objetos alvos desta forma.

Os autores de [5] criaram um modelo de reconhecimento gestual da mão através de sequências de imagens de vídeo analisando a diferente cor de pele da mão numa imagem 2D centrada na face (Figura 7). Este projeto tem em vista a classificação de dois tipos de gestos, dêiticos e simbólicos através de um modelo de classificação IOHMM (Input/Output Hidden

Figura 7 - Polygons a representarem face e mão [5]

Markov modelsSimplificando o teste do modelo tendo apenas que treinar com os outputs e o próximo estado que define o movimento.

As câmaras podem analisar o espectro visível, no entanto é possível realizar análise de imagens infravermelhas ou mais abaixo no espectro. Um resumo destes métodos pode ser encontrado em [6].

Os autores de [7] desenvolveram com 15 sensores EMG-Eletromiografia de superfície, um algoritmo que realiza o reconhecimento de gestos da mão analisando as imagens destes sensores. Para interpretar os dados utilizaram um algoritmo SVM (Support Vector Machine) por este revelar obter melhores resultados para reconhecimento discreto de gestos.

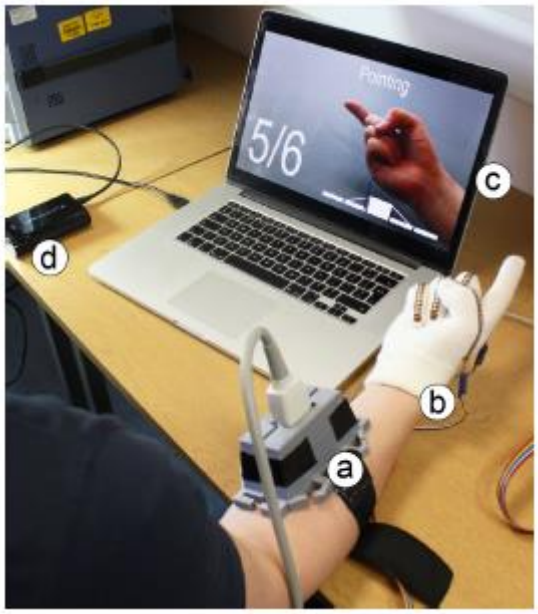

Figura 8 - Protótipo de [7] em que: a - transdutor; b- Luva sensorial; c - Computador, d - Camara

Os sensores IMU estão cada vez mais a ser introduzidos nos utensílios do cidadão comum proporcionalmente a redução no seu custo. O autor de [8], complementou a tecnologia IMU a sensores EMG para realizar o reconhecimento gestual de movimentos do braço e dos dedos.

O projeto Activity @ Home – Reconhecimento de gestos tem em vista criar uma framework para a utilização de sensores IMU no âmbito de classificação de gestos da maneira o mais económica possível.

2. Tecnologias

Neste capítulo descrevem-se os principais componentes de hardware usados no projeto A@H.

3.1 –Componente ESP8266

O módulo ESP8266-01 ilustrado na Figura 9 é um microchip de baixo custo que incorpora a stack protocolar TCP/IP e um microcontrolador, desenvolvido pela empresa chinesa Espressif Systems. [9]

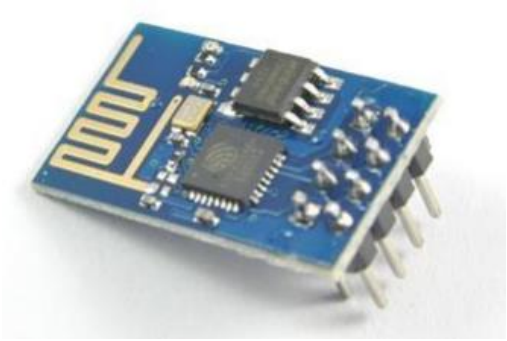

Figura 9 - ESP8266-01

As tabelas seguintes apresentam as características de hardware (Tabela 1) as características do software (Tabela 2) e as características do módulo WiFi (Tabela 3) referentes aos diferentes modelos do ESP8266. [10]

CPU	Processador Tensilica L106 de 32-bit
Interfaces de periféricos	ART/SDIO/SPI/I2C/I2S/ GPIO/ADC/PWM/LED Light & Button
Tensão de operação	2.5V ~ 3.6V
Corrente de operação	Valor medio: 80mA
Temperatura de funcionamento	–40°C ~ 125°C

Tabela 1- Características de Hardware dos modelos ESP8266

Modos de Wi-Fi	Station/SoftAP/SoftAP+Station
Segurança	WPA/WPA2
Encriptação	WEP/TKIP/AES
Atualização de firmware	UART Download / OTA (via network)
Desenvolvimento de Software	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
Protocolos de rede	IPv4, TCP/UDP/HTTP
Configuração de utilizador	AT Instruction Set, Cloud Server, Android/iOS App

Tabela 2- Características de Software dos modelos ESP8266

Certificação	Wi-Fi Alliance
Protocolos	802.11 b/g/n (HT20)
Alcance de frequência	2.4G ~ 2.5G (2400M ~ 2483.5M)
Potência de TX	802.11 b: +20 dB 802.11 g: +17 dBm 802.11 n: +14 dBm
Sensibilidade de RX	802.11 b: –91 dbm (11 Mbps) 802.11 g: –75 dbm (54 Mbps) 802.11 n: –72 dbm (MCS7)
Antena	PCB Trace, External, IPEX Connector, Ceramic Chip

Tabela 3- Características de WiFi dos modelos ESP8266

Os equipamentos ESP8266 foram concebidos para programação através do ESP-SDK. Existem, no entanto, outros IDEs que podem ser usados para programar o microcontrolador do módulo.

Segue a lista de IDEs alternativos recomendados, pelo fornecedor, para a programação dos equipamentos ESP8266:

- Moongoose OS
- MicroPython
- NodeMcu
- Arduino
- Platformio.org
- Zeryinth

No projeto Activity @ Home é utilizado o IDE do Arduino, com bibliotecas de compatibilidade criadas pela vasta comunidade de utilizadores.

3.1.1 – Pinout ESP8266-01

O pinout do ESP8266-01 encontra-se na Figura 10 e a descrição de cada pino encontra-se na *Tabela 4*

O circuito primário para o normal funcionamento do equipamento ESP8266-01 consiste na conexão do pino positivo da bateria ao pino de VCC do equipamento, o pino negativo da bateria ao pino de GND do equipamento. Estes pinos são unicamente responsáveis pela alimentação energética do equipamento. O Pino de CH_PD é o pino que quando conectado a VCC inicia o chip, sendo então sempre necessária esta ligação em qualquer circuito. (Figura 11)

Este módulo é alimentado a 3.3V. Muitos dos equipamentos FTDI com o qual é realizada a comunicação série funcionam a 5V, uma tensão que os pinos deste equipamento não toleram, podendo então danificar o equipamento irreversivelmente.

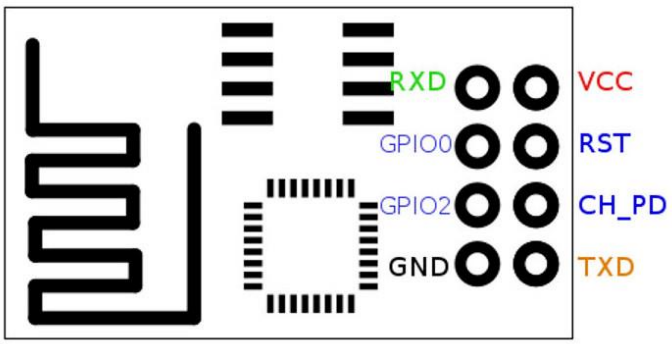

Figura 10-Pinout ESP8266-01

RXD	**Recebe Informação Serial**
GPIO0	Entrada/saída Analógica digital 0
GPIO2	Entrada/saída Analógica digital 2
GND	Ground
VCC	Vin
RST	Reset
CH_PD	Chip Enable
TXD	Envia Informação Serial

Tabela 4- Pinout ESP8266-01

3.1.2- Modos de Funcionamento

O modelo ESP8266-01 tem dois modos de funcionamento: o modo de programação e o modo de operação.

No modo de programação é possível fazer o upload do código realizado e no modo de operação este código é executado.

Para o dispositivo entrar em modo de programação é necessário iniciá-lo garantindo que o pino de GPIO0 esteja conectado a GND, sendo que quando o módulo é iniciado e o pino de GPIO0 não está conectado a GND o dispositivo entra em modo operação.

Isto torna-se um problema quando se pretende que no circuito final seja possível a reprogramação do equipamento, devido ao reduzido número de pinos existentes na ESP8266-01 e à necessidade de realizar comunicação I2C com o(s) sensor(es). Para se realizar esta comunicação são regularmente utilizados os pinos de GPIO0 e GPIO2 como entradas de SDA e SCL e estas ligações têm uma resistência de pullup para o VCC. É por isso necessária uma especial atenção ao switch utilizado para comutar entre os dois modos. Caso o botão quando pressionado conecte o ramo proveniente do GPIO0 a GND e o

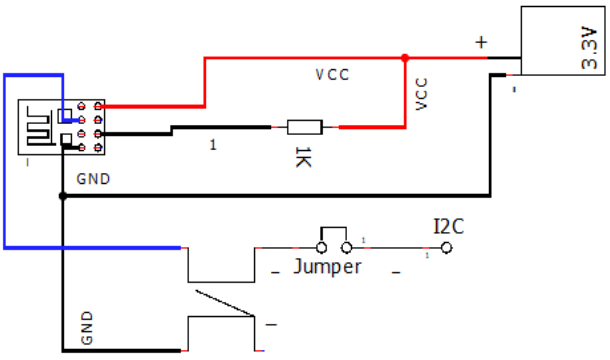

Figura 11 - Circuito básico do ESP8266-01

desconecte do ramo de comunicação I2C não existe o risco de criar um curto-circuito conectando os dois pinos da bateria ao mesmo ramo. No entanto caso isto não se verifique, como foi o caso neste projeto, é necessário ter um conector amovível para desconectar o ramo de comunicação de modo a evitar o mencionado curto circuito.

3.1.3 -Livrarias de compatibilidade

As livrarias de WiFi utilizadas para realizar a compatibilidade entre a ESP8266 e o IDE do arduino foram desenvolvidas pela comunidade de utilizadores deste equipamento, sendo então freeware.

Estas livrarias foram desenvolvidas com base no ESP8266 SDK da Espressif [11], mantendo a nomenclatura utilizada pelas livrarias de WiFi do arduino “ESP8266Wifi.h”.

Estas livrarias fornecem um conjunto de funções que são utilizadas ao longo deste projeto. As livrarias estão divididas por classes, sendo as seguintes utilizadas no projeto: WiFiClient, WiFiClientSecure e WiFiUDP. [12]

A classe WiFiClient recorre a classe Client para criar clientes que podem aceder a serviços fornecidos por servidores. Estes clientes podem enviar, receber e tratar dados, na Figura 12 é apresentado um exemplo típico desta comunicação.

```
class WiFiClient : public Client, public SList<WiFiClient> {
protected:
  WiFiClient(ClientContext* client);

public:
  WiFiClient();
  virtual ~WiFiClient();
  WiFiClient(const WiFiClient&);
  WiFiClient& operator=(const WiFiClient&);
```

Figura 13 - Classe WiFiClient

Figura 12 – Modelo de utilização da classe WiFiCliente - ESP8266 - ([12])

A classe WifiClientSecure é uma extensão da classe WiFiClient onde as conexões e transferência de dados com servidores é feita através de protocolos seguros. Suporta TLS 1.1, na Figura 14 é apresentado um exemplo típico desta comunicação.

Figura 14- Modelo de utilização da classe Cliente Seguro - ESP8266 - ([12])

A classe UDP possibilita o envio e a receção de mensagens UDP. O protocolo UDP utiliza um método simples de envio de pacotes, que ao contrário do protocolo TCP, cada pacote é uma unidade individual e é enviada sem preocupações de ordenamento ou duplicação de pacotes.

3.1.4 -Vantagens e Desvantagens do Equipamento ESP8266-01

O módulo ESP8266-01 tem como principal vantagem as suas dimensões, o que torna adequado para soluções *wearable*. No entanto, possui pouca capacidade de processamento e capacidade de memória.

Recorrendo às funções de DeepSleep é possível reduzir o consumo do equipamento de forma substancial, tornando-se esta uma vantagem bastante apelativa.

Uma desvantagem é a abordagem não intuitiva para a comutação entre dois modos de funcionamento que se revelou complicado em certas etapas deste projeto. A solução passaria por substituir este módulo por um modelo mais recente da mesma geração como por exemplo o ESP8266-Sparkfun Thing. No entanto estes têm dimensões maiores, o que não é o ideal.

3.2 – Sensor IMU – BNO055

O sensor inercial usado para reconhecimento gestual foi o BNO055 da BOSCH (Figura 15). Este sensor é bastante pequeno o que é uma característica apelativa para a sua utilização na bracelete. O sensor BNO055 incorpora um acelerómetro inercial triaxial, um giroscópio triaxial, um magnetómetro triaxial e um microcontrolador que realiza a fusão dos dados fornecidos pelos sensores e os convertem em dados prontos a ser processados

Este sensor também se destaca pela capacidade de fornecer a posição angular absoluta a um preço baixo em comparação com os restantes do mercado.

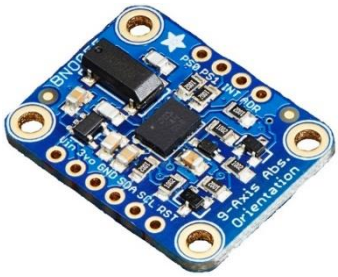

Figura 15 – Sensor inercial BNO055.

3.2.1 – Princípios de funcionamento de IMUs

Os acelerómetros são dispositivos eletromecânicos que reagem a forças de aceleração dinâmicas e estáticas. As estáticas incluem a gravidade e as dinâmicas incluem vibrações.

Normalmente, os acelerómetros são compostos por placas capacitivas, presas ou ligadas a molas minúsculas que movem internamente quando lhes é aplicada uma força de aceleração. Conforme estas placas se aproximam uma da outra a capacidade entre elas muda. Através destas mudanças de capacidade é possível fazer o cálculo da aceleração.

Outros métodos envolvem materiais piezoelétricos, que são estruturas cristalinas quando sujeitas a forças (gravidade no caso do exemplo da Figura 16) criam tensões com as quais se pode obter aceleração. [13]

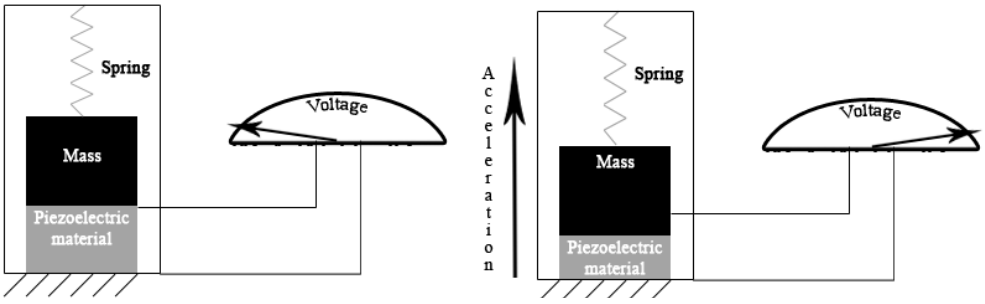

Figura 16 - Acelerometro de materiais piezoelétricos

Os giroscópios são dispositivos eletromecânicos que medem velocidade angular, podendo então determinar a rotação de um corpo. Pela Figura 17 é possível distinguir os diferentes tipos de giroscópios:

- Giroscópios de anéis de laser
- Giroscópios de fibra ótica
- Giroscópios de fluidos
- Giroscópios de viração

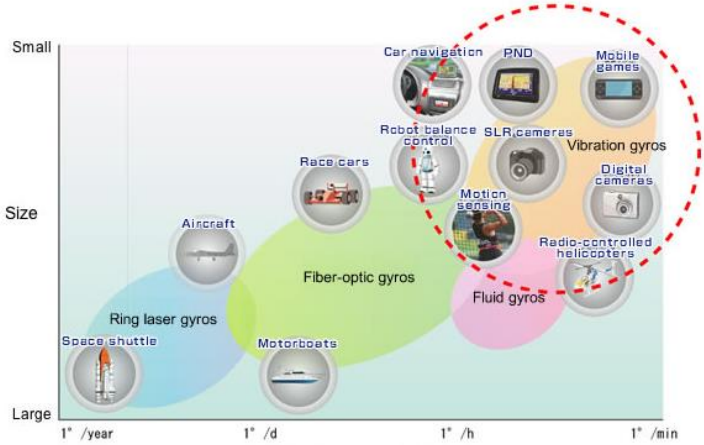

Figura 17- Gyro Sensor types [14]

Como exemplo será analisado o processo dos giroscópios de vibração, que reagem à velocidade ângular aplicada pela força de Coriolis a objetos em estado de vibração. Existem vários modelos de giroscópios de vibração, no entanto, tomemos como exemplo o Epsons's double-T structure crystal elemento ilustrado na Figura 18.

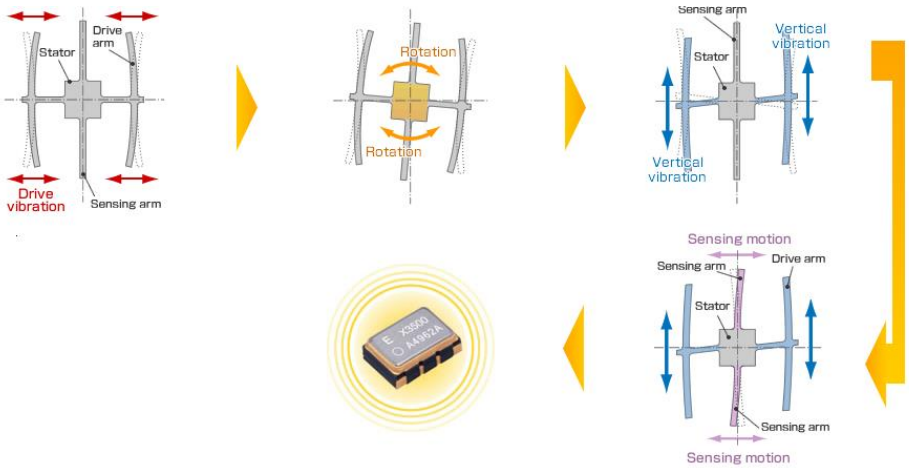

Figura 18-Processo do Epson 's double-T structure crystal element [14]

Inicialmente o braço condutor (Drive arm) vibra numa direção. Quando existe uma rotação esta causa, através do efeito de Coriolis, uma vibração vertical fazendo então com que o braço estacionário (Sensing arm) dobre. O movimento do par de braços estacionários produz a diferença de potencial sobre a qual é possível observar a velocidade angular. Esta velocidade é convertida e enviada como um sinal elétrico. [14]

O magnetómetro é um sensor de categoria portátil, e mede a alteração relativa do campo magnético numa localização.

3.2.2 - Fusão dos Dados Sensoriais

O sensor BNO055 incorpora um microcontrolador que faz a fusão dos dados enviados pelos diferentes sensores, que sendo triaxiais se encontram a 90 graus entre si sobre os eixos ortogonais.

A informação que é obtida através do sensor é de dois tipos, aceleração em m/s^2 sobre os eixos de orientação e orientação absoluta através ou de ângulos de Euler ou Quaterniões.

Os Ângulos de Euler (Figura 19) fornecem a rotação em graus, segundo os eixos ortogonais definidos quando o equipamento é iniciado. Quaterniões são o quociente de duas linhas retas num espaço tridimensional/dois vetores.

Por defeito, as livrarias de compatibilidade utilizadas para obter os dados do sensor fornecem orientação absoluta através de ângulos de Euler.

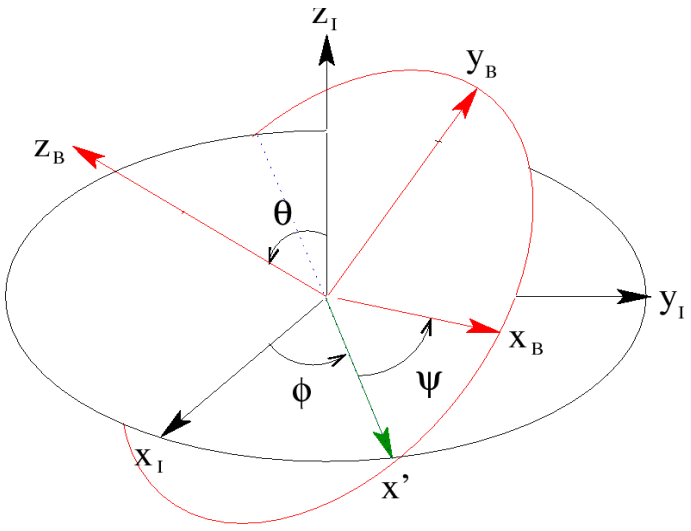

Figura 19 - Ângulos de Euler [23]

Tomando a posição A da Figura 20, como o ponto inicial sobre o qual todos os outros pontos de orientação são baseados, temos os seguintes valores apresentados pela Tabela 5 (assumindo que o sensor se encontra estático):

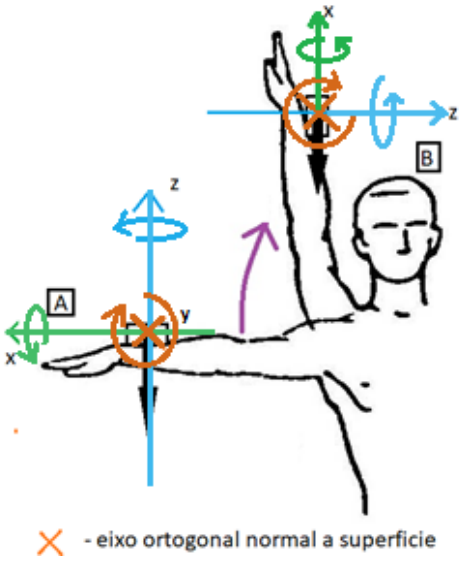

Figura 20 - Vetores de aceleração

Angulo Euler de rotação de x	0º
Ângulo Euler de rotação de y	0º
Ângulo Euler de rotação de z	0º
Aceleração em x	0
Aceleração em y	0
Aceleração em z	-9.8 m/s^2

Tabela 5- Vetores de aceleração para a posição A

A seta roxa na Figura 20 representa o sentido de movimento, após se encontrar na posição B obtemos os seguintes valores, apresentados pela Tabela 6:

Angulo Euler de rotação de x	**0º**
Ângulo Euler de rotação de y	-90º
Ângulo Euler de rotação de z	0º
Aceleração em x	-9.8 m/s^2
Aceleração em y	0
Aceleração em z	0

Tabela 6 - Vetores de aceleração para posição B

Através da análise das tabelas anteriores é possível concluir que os dados fornecidos por defeito pelo sensor, acartam particularidades que podem criar problemas para a algoritmia da tarefa em causa.

É ainda importante referir que, considerando unicamente um sensor no pulso é impossível de discernir entre uma rotação através de movimentos do braço e uma rotação de corpo inteiro.

3.2.3 – Modos de funcionamento

O sensor BNO055 dispõe de diferentes modos de operação listados na Tabela 7. Tirando o de configuração, os modos estão subdivididos em dois grupos, os que realizam a fusão dos dados e os que não o fazem. A Tabela 7 apresenta também quais sensores cada modo utiliza. Caso o modo escolhido não utilize todos os sensores, os que não são usados estão desativados não consumindo energia.

Operating Mode		Available sensor signals			Fusion Data	
		Accel	Mag	Gyro	Relative orientation	Absolute orientation
	CONFIGMODE	-	-	-	-	-
Non-fusionmodes	ACCONLY	X	-	-	-	-
	MAGONLY	-	X	-	-	-
	GYROONLY	-	-	X	-	-
	ACCMAG	X	X	-	-	-
	ACCGYRO	X	-	X	-	-
	MAGGYRO	-	X	X	-	-
	AMG	X	X	X	-	-
Fusion modes	IMU	X	-	X	X	-
	COMPASS	X	X	-	-	X
	M4G	X	X		X	-
	NDOF_FMC_OFF	X	X	X	-	X
	NDOF	X	X	X	-	X

Tabela 7 - Modos de operação do sensor inercial BNO055.

Existem também três modos de energia Tabela 8. A definição padrão de energia do sensor é consumo normal. É, no entanto, possível mudar para o modo de baixo consumo de energia no qual caso o sensor não detete movimento significativo durante um espaço de tempo (de origem 5 segundos) desliga os sensores de magnetómetro e giroscópio e volta a ligá-los quando o acelerómetro capturar movimento novamente. No modo de suspensão o sensor não está a detetar quaisquer movimentos sendo então o modo de conservação de energia que consome 0.4mA.

Parameter	Value	[Reg Addr]: Reg Value
Power Mode	Normal Mode	[PWR_MODE]: xxxxx00b
	Low Power Mode	[PWR_MODE]: xxxxx01b
	Suspend Mode	[PWR_MODE]: xxxxx10b

Tabela 8- Modos de energia do sensor inercial BNO055

3.3 – Protocolo de Comunicação I2C

I2C é um protocolo de comunicação de curta distância que funciona à base de conexões mestre(s)/escravo(s). Para realizar esta comunicação são necessárias duas ligações, SDA e SCL. Os dados são colocados na ligação de SDA quando SCL fica no estado low e são amostrados quando este fica no estado high. Cada ligação tem uma resistência de pull-up para colocar o sinal em high quando nenhum equipamento o definir como low [15].

Nesta comunicação existem dois tipos de pacotes, as de endereçamento e de dados (Figura 21).

Figura 21 - janelas de comunicação I2C [16]

3.4 – PWM – Pulse Width Manipulation

PWM é uma técnica utilizada para obter dados analógicos através de meios digitais. Esta técnica cria uma onda quadrada que é emitida num período de tempo, a percentagem desse período em que o sinal se encontra na referência positiva é chamado de Ton e o período total de Ttot. A divisão de Ton sobre Ttot é chamado de Duty Cycle e é referente à percentagem da potência total que pode ser fornecida ao equipamento que se pretende controlar (Figura 22).

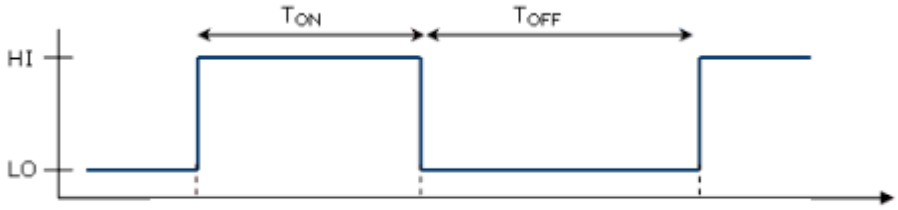

Figura 22- Exemplo de Sinal PWM

3.5 – Machine Learning– Técnicas de Classificação

Machine Learning é um ramo de inteligência artificial que utiliza técnicas probabilísticas para fornecer a sistemas a capacidade de "*aprender"* a realizar uma tarefa especifica sem que esta tenha que ser explicitamente programada.

Machine Learning está subdividido em dois ramos, aprendizagem supervisionada e não supervisionada.

Neste projeto são utilizadas técnicas de aprendizagem supervisionada, que mapeiam dados input/output através de exemplos de pares input/output. Estes criam funções através de dados etiquetados que podem ser utilizados para mapear novos dados.

Para o projeto A@H serão utilizados métodos de classificação, onde os inputs podem ser divididos em uma ou mais classes, e o sistema tem que produzir um modelo que designa novos inputs a uma ou mais destas classes.

Com o sentido de contextualização serão explicados dois métodos de classificação supervisionada:

> Knn –k-nearest neighbors, quando utilizado para classificação, a classe que designa ao input é definida pela distância Euclidiana dos vizinhos mais próximos dos dados treinados, escolhendo a classe etiquetada mais comum entre os k-vizinhos mais próximos.
>
> SVM – Support vector machines constroem um ou mais hiperplanos que podem ser utilizados para classificação. Uma boa separação das classes pode ser alcançada pelo hiperplano com a máxima distância entre qualquer amostra de treino de cada classe.

É de seguida apresentado um exemplo de cada um dos classificadores, Figura 23 e Figura 24 para Knn e SVM respetivamente, para os dataset Iris, que pretende classificar diferentes tipos de flores pelo comprimento e largura das pétalas.

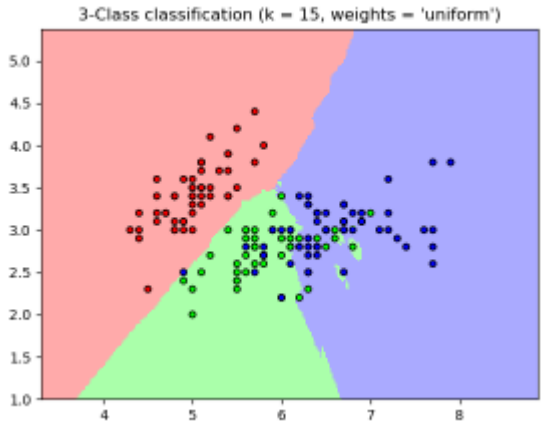

Figura 23- Classificador Knn - Dataset Iris

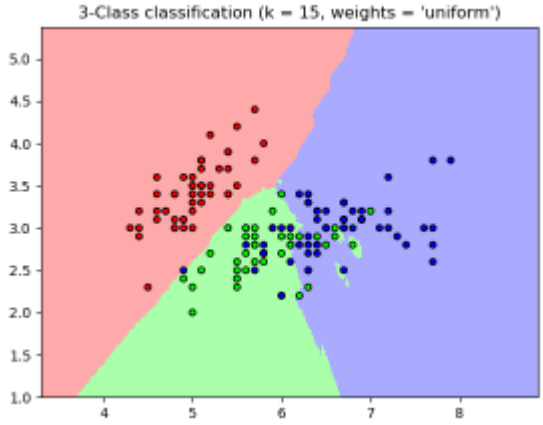

Figura 24- Classificador SVM - Dataset Iris

Os dados estão estruturados sobre matrizes onde as linhas são as diferentes amostras e as colunas são as features destas amostras. É possível mapear a fronteiras de decisão entre classes nas imagens anteriores por este dataset set bidimensional (tem duas features), acima de três dimensões é impossível mapear visualmente esta fronteira de decisão.

4 – Sistema A@H

Neste capítulo é explicado o desenvolvimento dos modelos das figuras Figura 1,Figura 2 e Figura 3 apresentados no capítulo 1. Serão explicados os circuitos realizados e a programação e funcionamento dos três subsistemas.

4.1 - Placa de reprogramação

A primeira etapa deste projeto foi criar uma maneira de poder reprogramar o equipamento sem ter que se utilizar o circuito implementado na *breadboard*. Foi então desenvolvida numa placa PCB um circuito de reprogramação para o equipamento ESP8266-01. Este circuito foi concebido tendo em mente que esta seria uma placa de desenvolvimento onde se poderia programar o equipamento ESP8266-01 e testar o código programado. No entanto isto só é possível se o código que estamos a testar não utilizar os pinos de GPIO0 e GPIO1, que não foi o caso dos circuitos desenvolvidos neste projeto.

A placa foi impressa tendo por base o circuito da Figura 25. No entanto ao soldar evitou-se o uso de um dos botões fazendo um shunt entre o pino de GPIO0 e GND de modo a que a ESP8266-01 inicie sempre o modo de programação. Ao longo deste projeto quando é referido que o ESP8266-01 é amovível significa que o dispositivo pode ser removido e sua reprogramação realizada através desta placa.

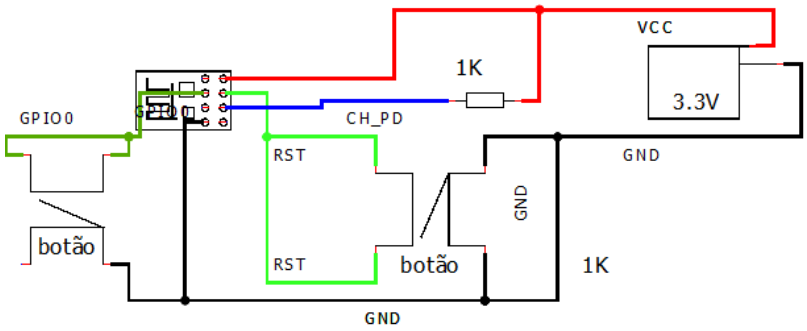

Figura 25- Circuito board de reprogramação

4.2 – Botão de Emergência – Activity@Home

O Botão de Emergência – A@H (BT-A@H) foi o primeiro sistema a ser desenvolvido neste projeto, servindo também então como uma introdução à configuração do ESP8266-01. O desenvolvimento pode ser divido em três partes:

- Hardware e configuração
- Envio de mensagem de alerta através de Email
- Código de firmware

4.2.1 – Hardware

Foram realizados protótipos de dois circuitos para o botão de emergência A@H. i) Versão A: um circuito com maiores dimensões mais robusto e recarregável; ii) Versão B: um circuito simplificado com o intuito de se tornar mais pequeno.

O primeiro protótipo (Figura 26) tem capacidades de recarregamento de bateria, um LED que pisca quando o alerta é concluído com sucesso e um botão de ON/OFF geral.

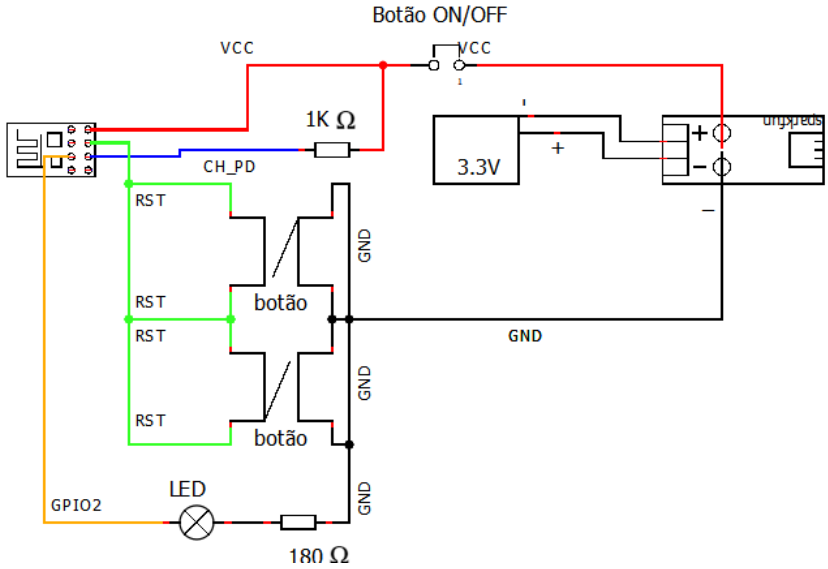

Figura 26-A@H - Circuito Botão de Emergência - Versão A

Nesta versão é utilizado o módulo de recarregamento micro USB de baterias lipo de 3.7V da Sparkfun, ilustrado na Figura 27.

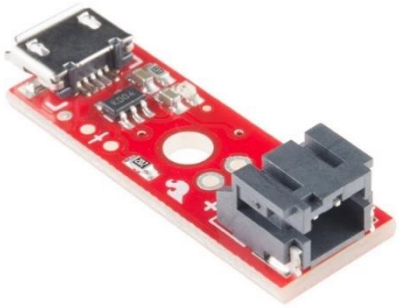

Figura 27- Carregador Sparkfun [17]

Este equipamento carrega uma bateria Lipo a 500mA. É importante referir que devido ao facto de se tentar minimizar o espaço utilizado foi adquirida uma bateria LiPo de 3.7V com dimensões menores que a disponível na altura, no entanto devido a esta troca, o adaptador para o módulo de recarregamento não encaixava. Observa-se no circuito da Figura 28 que o pino sobre o qual se conecta o circuito para funcionamento e o pino sobre o qual se insere a bateria para carregar é o mesmo. Decidiu-se então colocar um header macho que por cima conecta a bateria e por baixo se liga à placa PCB.

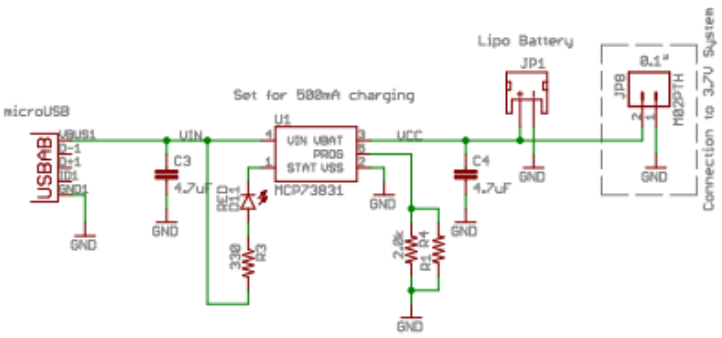

Figura 28- Circuito Carregador Sparkfun

O ESP8266-01 está montado em cima de headers sendo, portanto, amovível. Devido ao tamanho do botão, para garantir que este é pressionado colocaram-se dois botões em paralelo que realizam a mesma função. Este protótipo foi implementado na breadboard, tendo funcionado com sucesso.

O próximo passo foi a impressão para placa PCB. Para este efeito foi utilizado o software EAGLE da Autodesk. Devido ao tamanho da bateria e numa tentativa de criar um modelo com menos altura decidiu-se colocar um buraco na placa PCB de modo a que esta encaixe dentro da placa, como demonstrado na Figura 29.

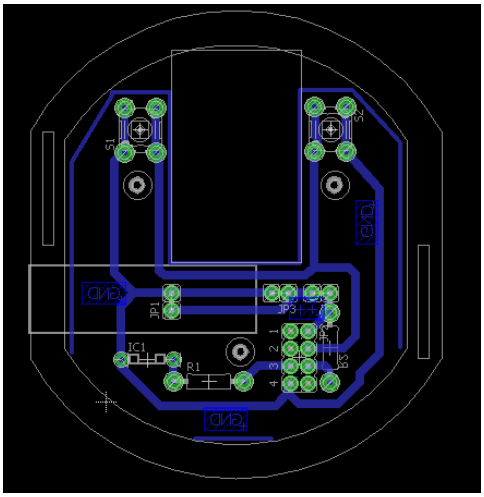

Figura 29- EagleBoard - BT-A@H - Versão A

Foi ao mesmo tempo desenvolvido e impresso um modelo 3D criado em SOLIDWORKS que encapsula esta placa (Figura 30). Este modelo é composto por duas partes: a base e a capa. A base é onde, através de saliências, é encaixada a placa PCB que tem buracos dimensionados de modo a encaixar a placa no sítio correto. A capa contém o botão físico que é percetível pela diferença de altura na sua superfície.

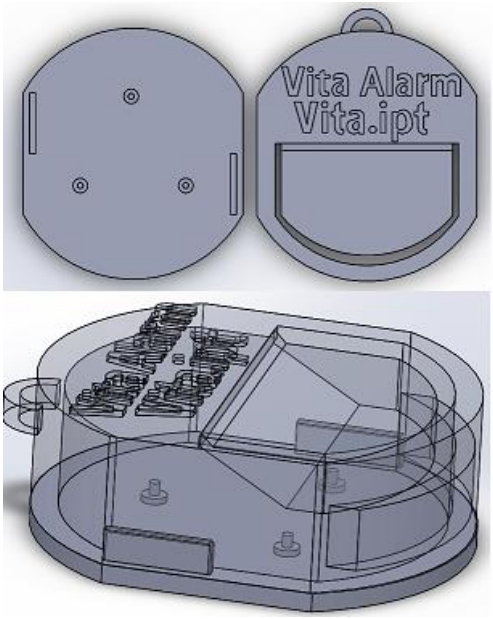

Figura 30 - Solidworks Model - A@H - Botão de Emergência - Versão A

Apesar do circuito ter sido montado com sucesso numa placa breadboard, quando impresso e testado o circuito não funcionou. O LED de comunicação encontrava-se ligado sem piscar e sem nenhuma mensagem a ser enviada através da porta Serial para a FTDI. O circuito foi revisto por continuidade e esta apresentava-se correta, procedeu-se então ao desmantelamento de pistas que ligam o pino de RST ao botão numa perspetiva de se encontrar o problema, mas sem sucesso.

Uma vez que as dimensões do botão estavam exageradamente grandes decidiu-se avançar para uma implementação de uma versão mais simplificada e mais pequena, a versão B.

A versão B do BT-A@H consiste nos seguintes componentes: ESP8266-01 montado em cima de headers, de forma amovível; um LED de sinalização que pisca quando o alerta é enviado com sucesso, e um botão de reset (Figura 31).

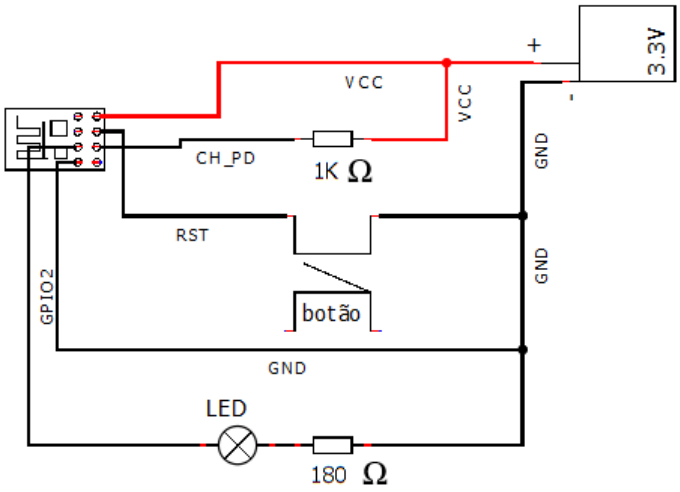

Figura 31-A@H - Circuito Botão de Emergência - Versão B

Este circuito foi modelado no EAGLE e impresso em PCB (Figura 32).

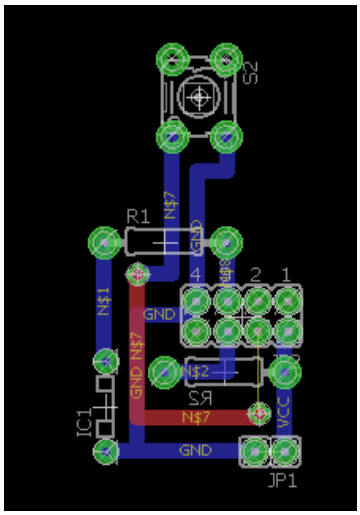

Figura 32- Eagle Board - A@H - Botão de Emergência - Versão B

No entanto esta versão apresentou o mesmo problema, e uma vez que o único componente externo que não tinha sido anteriormente testado com sucesso em placa PCB era o LED este foi removido e o circuito ficou funcional.

Foi decidida então a implementação da versão B e impresso em 3D uma capa que encapsula o circuito.

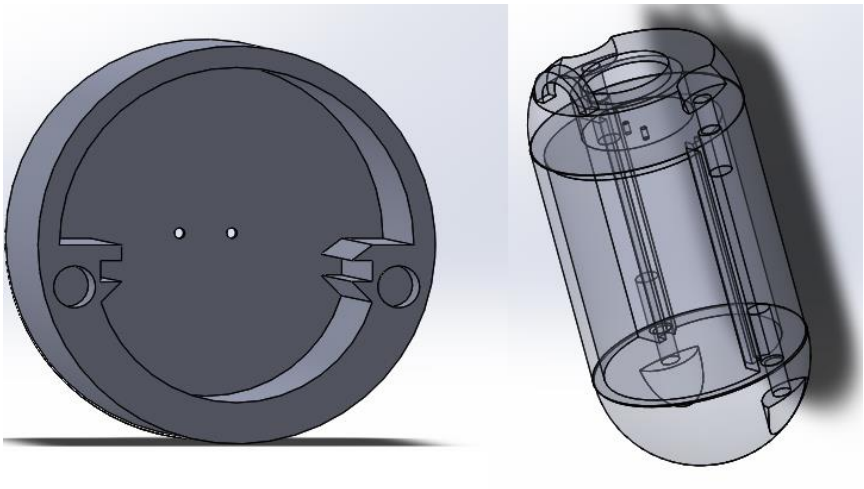

Figura 33- Solidworks Model - A@H - Botão de Emergência - Versão B

O modelo é composto por três diferentes partes, o centro capa e botão.

O centro foi concebido de modo a que a placa PCB encaixe na calha localizada aproximadamente a meio com comprimento suficiente para o ESP8266-01 e altura para o header no lado superior e bateria no lado inferior. Num dos lados do centro existe uma parede com dois buracos onde são passados os fios de um botão que está colado ao lado externo desta parede.

O botão é uma cápsula sobre a qual é pressionado e ativado o botão colado ao lado anteriormente mencionado. A capa é colocada no lado oposto e esta é usada para inserir e retirar a placa e a bateria da cápsula.

Ambos os modelos concebidos para o botão de emergência possuem um aro sobe o qual é passado um fio tornando-o então num pendente.

4.2.2 – Envio de mensagem de alerta

A mensagem de alerta é enviada através de um serviço de email. A execução deste serviço é feita com recurso às livrarias de compatibilidade da ESP8266-01 referidas na secção 3.1.3 -Livrarias de compatibilidade. Foi seguido o tutorial [18], que exemplifica um método de o fazer através de um serviço chamado SMTP2GO [20] que aceita mensagens do protocolo SMTP e as encaminha para o email pretendido.

Apesar deste serviço ter uma versão gratuita esta permite apenas um número limitado de emails diários, mas serviu para testar e validar o sistema.

Foi então encontrado o serviço IFTTT – IfThisThenThat [19] gratuito que realiza em background a interação de uma grande variedade de serviços inclusive mensagens por SMS, Email, Facebook, Twitter, Instagram entre outras. No entanto alguns destes serviços não se encontram disponíveis em Portugal.

Para prova de conceito foi apenas implementado o serviço de email, no entanto hoje em dia, dada a preponderância das redes sociais na vida do cidadão comum, os serviços envolventes podem também revelar-se uma solução alternativa bastante apelativa.

Neste serviço são criadas applets que funcionam sobre uma relação de causa efeito (If This Then That), caso algo aconteça o serviço encarrega-se da resposta pretendida.

A causa escolhida para o Botão de Emergência – Activity @ Home é a chamada de um link URL através do serviço de WebHooks do IFTTT. Quando se cria a applet é associada uma conta do WebHooks na qual se obtém o link que é preciso ser chamado, este link é privado e convém não ser disponibilizado a terceiros sendo que estes podem ativar a causa se o usarem (Figura 34).

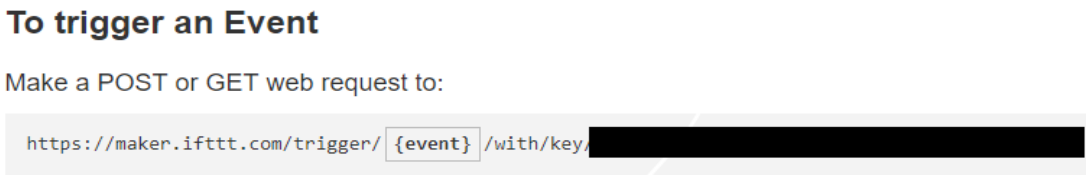

Figura 34- Webhooks

Como mencionado anteriormente, a resposta utilizada é um email, cujo endereço, assunto e corpo são configuráveis na applet criada como se pode ver na Figura 35 .

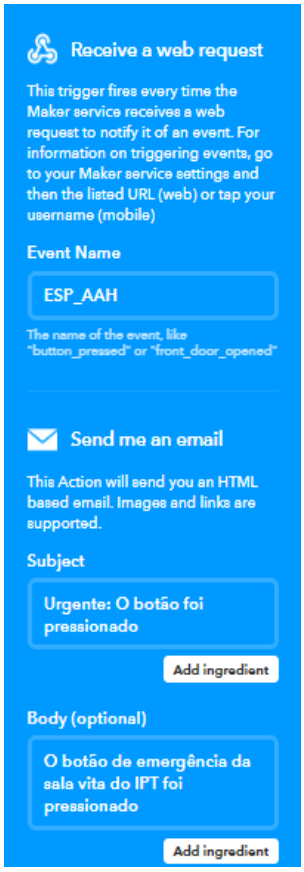

Figura 35- Applet ESP_A@H

O resultado obtido da ativação desta applet através do BT – A@H, é o email conforme representado na Figura 36Figura 36.

Figura 36- Email de Emergencia

4.2.3 – Código de firmware

O código do microcontrolador embutido no ESP8266 foi escrito usando o IDE do Arduino. Este tem duas rotinas, a rotina de setup que é executada uma vez quando o equipamento é ligado e uma rotina de loop que é executada em ciclo infinito, apenas para manter o programa em execução.

O fluxograma do código desenvolvido está ilustrado na Figura 37.

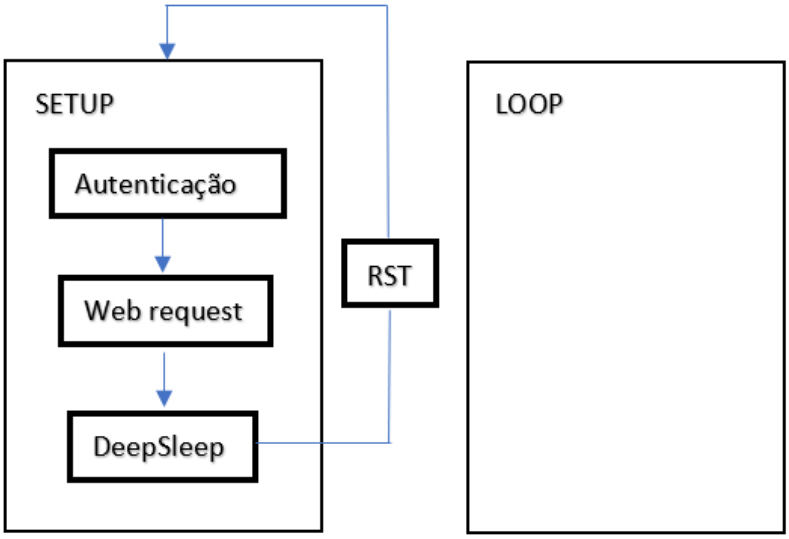

Figura 37 - Fluxograma Botão de Emergência - A@H

Antes da rotina de setup, são declaradas as livrarias de compatibilidade referidas na secção 3.1.3 -Livrarias de compatibilidade, e as variáveis globais necessárias para o funcionamento do algoritmo, como mostra a Figura 38

```
#include <ESP8266WiFi.h>              //Livrarias de compatibilidade com ESP8266 e comunicação WiFi
#include <WiFiClientSecure.h>         // --
const char* ssid = "VITA.IPT-WIFI";   // SSID - Nome da rede Local para connecção
const char* password = "leec_m2e";    // Password - Password da rede Local para Connecção
const char* host = "maker.ifttt.com"; // Serviço de Host para realizar o request
const int httpsPort = 443;            // Defenição da portal para protocolo HTTPS
#define LED_PIN 2                     // (Opcional) - LED para confirmação de comunicação
```

Figura 38- Livrarias e variáveis - Botão de Emergência - A@H

Para a comunicação com um servidor externo à rede local é recomendado utilizar a Classe de cliente seguro, apesar destas funções terem um overhead de memória (e processamento) por usarem algoritmos de encriptação. Isto causa problemas com a

ESP8266-01 a nível de memória RAM que não é possível suplementar. Sendo então possível apenas ter um objeto de Cliente Seguro de cada vez.

Na rotina de Setup, a etapa de autenticação, é inicializada pela comunicação através da porta serie. Isto é útil para fazer debugging, e para se saber em que etapa o programa se encontra. Esta comunicação pode ser desativada posteriormente para se obter acesso aos pinos de TX e RX do ESP8266-01, não sendo no nosso caso necessário. Autentica-se então à rede local através da função *WiFi.Begin* com parâmetros de *username* e *password* respetivamente. Enquanto a flag de autenticação *WiFi.status( )* não for ativada este fica em loop a tentar conectar-se à rede. Este loop pode ser mostrado através de duas maneiras, ou por comunicação serie com a função Serial.println( ) ou com um LED externo (Figura 39).

```
void setup() {
   Serial.begin(115200);              // (Opcional) - Inicia a comunicação Serial para debugging
   pinMode(LED_PIN, OUTPUT);          // (Opcional) - Define o GPIO2 como output para o LED
   digitalWrite(LED_PIN, LOW);        // (Opcional) - Define que o LED começa desligado

  WiFi.begin(ssid, password);         // Função que realiza a autenticação com a rede local
  while (WiFi.status() != WL_CONNECTED) { // Ciclo até realizar esta connecção
      digitalWrite(LED_PIN, HIGH);    // (Opcional) - Enquanto nao se connectar à rede local pisca o LED
      delay(500);                     // --
      digitalWrite(LED_PIN, LOW);     // --
      delay(500);                     // --
```

Figura 39- Etapa de Autenticação - Botão de Emergência - A@H

De seguida o algoritmo tenta estabelecer ligação com os servidores da IFTTT. Caso isto seja realizado com sucesso procede à execução da função *GET* com o URL fornecido pelo serviço Web Hooks e pisca o LED externo (que posteriormente foi removido) a informar o sucesso do envio (Figura 40).

```
      WiFiClientSecure client;           // Após estabelecer a conneção cria o objecto client referente ao cliente
       if (!client.connect(host, httpsPort)) { // Tenta estabelecer connecção com o serviço IFTTT
        Serial.println("connection failed"); // Confirma a connecção
        return;
      }

      String url = "https://maker.ifttt.com/trigger/ESP_AAH/with/key/bcRWj-NVFdjekE5MwpzHV"; // URL Secreto para tirgger da aplet

       client.print(String("GET ") + url + " HTTP/1.1\r\n" +        // Realiza a função Get sobre o URL
                    "Host: " + host + "\r\n" +                     //--
                    "User-Agent: BuildFailureDetectorESP8266\r\n" + //--
                    "Connection: close\r\n\r\n");                  //--
       while (client.connected()) {                                //--
         String line = client.readStringUntil('\n');               //--
         if (line == "\r") {                                       //--
           Serial.println("headers received");                     //--
           break;                                                  //--
         }                                                         //--
       }                                                           //--
       String line = client.readStringUntil('\n');                 // E confirma o seu sucesso

       digitalWrite(LED_PIN, HIGH);                                // Opcional - Confirmação fisica do sucesso para utilizador
       delay(4000);
       digitalWrite(LED_PIN, LOW);
```

Figura 40- Etapa de Web Request - Botão de Emergência - A@H

Após realizar este pedido é executada a função de DeepSleep que coloca o equipamento em modo de baixo consumo desligando todos os periféricos incluindo os que controlam o modulo de comunicação WiFi (Figura 41).

```
   ESP.deepSleep(0, WAKE_RF_DEFAULT);
  delay(10000); //Security, the sleep mode need some time to activate
}
void loop() {}
```

Figura 41- Etapa de DeepSleep Botão de Emergência - A@H

4.3 – Reconhecimento de Gestos – Activity @ Home

Ambos modelos do Reconhecimento de Gestos – A@H partilham a mesma Pulseira que adquire os dados inerciais. Como podemos ver nas Figura 2 e Figura 3 esta pulseira é responsável por receber os dados sensoriais do IMU BNO055 e encaminhá-los para os servidores desejados através da ESP8266-01. A este equipamento damos o nome de Pulseira – A@H (P-A@H).

4.3.1 – Pulseira – Activity @ Home

A P-A@H é composta por um sensor BNO055 que envia os dados por I2C para o microcontrolador do ESP8266-01.

Como podemos ver na Figura 10 o pinout da ESP8266-01 não tem pinos alocados de SDA e SCL, é no entanto possível através da função *wire.begin( )* das livrarias wire.h do Arduino definir os pinos de GPIO0 e GPIO2 para esta comunicação. É possível realizar dois tipos de comunicação escrita e leitura.

Para realizar a calibração do sensor BNO055 é utilizado o exemplo “restore_offsets” proveniente das livrarias do sensor, no IDE do Arduino.

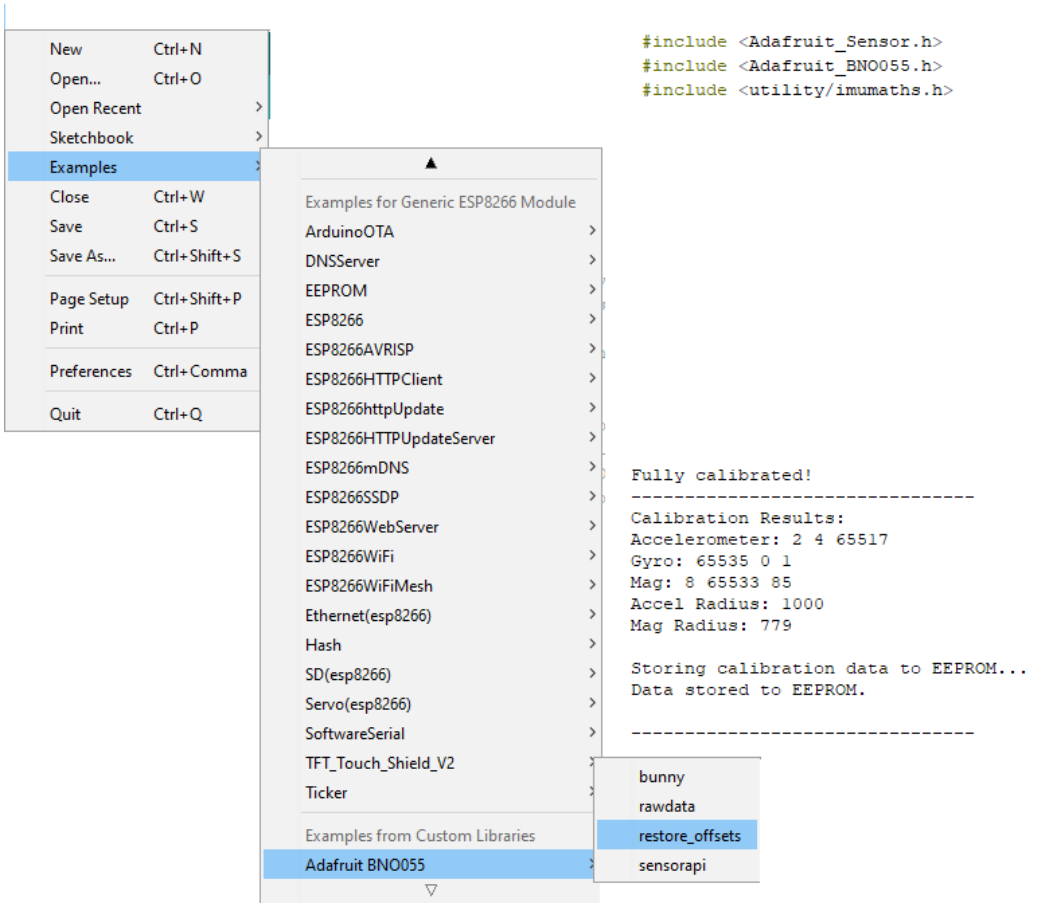

Figura 42- Exemplo "Reset_OffSets" das livrarias da adafruit e resultado

Através do exemplo da Figura 42 é possível calibrar o sensor BNO055. Para isto é realizado o upload do código para o ESP8266-01 e executado, mantendo a comunicação série através de uma FTDI até os valores de cada sensor obtidos no output da comunicação série serem 3 é necessário percorrer todas as superfícies de um objeto retangular. Uma vez que todos estes sensores se encontrem calibrados é apresentada a resposta vista na Figura 42

Apesar da informação proveniente da comunicação Serie afirmar que a calibração se encontra guardada na EEPROM (Electrically Erasable Programmable Read-Only Memory) não foi escolhido este método para configurar a calibração no Reconhecimento de Gestos – A@H, sendo feito cada registo manualmente no inicio do programa após se estabelecer a ligação I2C.

Através dos resultados de calibração é possível, após converter os valores para binário, obter os valores que têm de ser escrito em cada registo apresentados nas Tabela 10e. (MSB – Byte mais significativo; LSB – Byte menos significativo)

OFFSET	*MSB - X*	*LSB - X*	*MSB-Y*	*LSB-Y*	*MSB-Z*	*LSB-Z*
Acelerómetro	0000 0000	0000 0010	0000 0000	0000 0100	1111 1111	1110 1101
Giroscópio	1111 1111	1111 1111	0000 0000	0000 0000	0000 0000	0000 0001
Magnetómetro	0000 0000	0000 1000	1111 1111	1111 1101	0000 0000	0101 0101

Tabela 10 - Offsets de Calibração

Raio	***MSB***	***LSB***
Acelerómetro	0011	1110 1000
Magnetómetro	0011	0000 1011

Tabela- 9 - Raio de Calibração

Os registos correspondentes à calibração são obtidos através do datasheet do equipamento BNO055 [20], e podem ser vistos na figura

Register Address	Register Name	Default Value	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
7F-6B	Reserved	NA								
6A	MAG_RADIUS_MSB		Magnetometer Radius							
69	MAG_RADIUS_LSB		Magnetometer Radius							
68	ACC_RADIUS_MSB		Accelerometer Radius							
67	ACC_RADIUS_LSB		Accelerometer Radius							
66	GYR_OFFSET_Z_MSB	0x00	Gyroscope Offset Z <15:8>							
65	GYR_OFFSET_Z_LSB	0x00	Gyroscope Offset Z <7:0>							
64	GYR_OFFSET_Y_MSB	0x00	Gyroscope Offset Y <15:8>							
63	GYR_OFFSET_Y_LSB	0x00	Gyroscope Offset Y <7:0>							
62	GYR_OFFSET_X_MSB	0x00	Gyroscope Offset X <15:8>							
61	GYR_OFFSET_X_LSB	0x00	Gyroscope Offset X <7:0>							
60	MAG_OFFSET_Z_MSB	0x00	Magnetometer Offset Z <15:8>							
5F	MAG_OFFSET_Z_LSB	0x00	Magnetometer Offset Z <7:0>							
5E	MAG_OFFSET_Y_MSB	0x00	Magnetometer Offset Y <15:8>							
5D	MAG_OFFSET_Y_LSB	0x00	Magnetometer Offset Y <7:0>							
5C	MAG_OFFSET_X_MSB	0x00	Magnetometer Offset X <15:8>							
5B	MAG_OFFSET_X_LSB	0x00	Magnetometer Offset X <7:0>							
5A	ACC_OFFSET_Z_MSB	0x00	Accelerometer Offset Z <15:8>							
59	ACC_OFFSET_Z_LSB	0x00	Accelerometer Offset Z <7:0>							
58	ACC_OFFSET_Y_MSB	0x00	Accelerometer Offset Y <15:8>							
57	ACC_OFFSET_Y_LSB	0x00	Accelerometer Offset Y <7:0>							
56	ACC_OFFSET_X_MSB	0x00	Accelerometer Offset X <15:8>							
55	ACC_OFFSET_X_LSB	0x00	Accelerometer Offset X <7:0>							

Figura 43- Tabela de registos de calibração BNO055 [20]

Através das livrarias wire.h do Arduino anteriormente mencionadas é possível realizar a escrita destes registos, no entanto como mencionado na secção 3.2.3 – Modos de funcionamento para realizar esta escrita é necessário colocar o sensor BNO055 em modo de configuração através do registo 0x3D.

```
Wire.beginTransmission(address);  // Inicia o modo de configuração
Wire.write(0x3D);                 // --
Wire.write(0x0);                  // --
Wire.endTransmission();           // --
delay(20);                        // --

Wire.beginTransmission(address); // offset do x do acelarometro
Wire.write(0x56);                // byte mais significativo
Wire.write(0b0);                 // 0000 0000
Wire.endTransmission();          // --
```

Figura 44- Escrita de registo de calibração

O circuito modelado para a Bracelete – A@H é composto por:

- ESP8266-01 – Diretamente soldada na placa PCB sendo, portanto não amovível
- BNO055 – Diretamente soldado na placa PCB sendo, portanto não amovível.
- Entrada fêmea para comunicação Serie através de uma FTDI para fins de debugging e reprogramação
- Dois botões para comutar o modo de funcionamento da ESP8266-01 e para reiniciar o equipamento ESP8266-01

Este circuito pode ser observado na Figura 45, e a sua implementação em PCB na Figura 46.

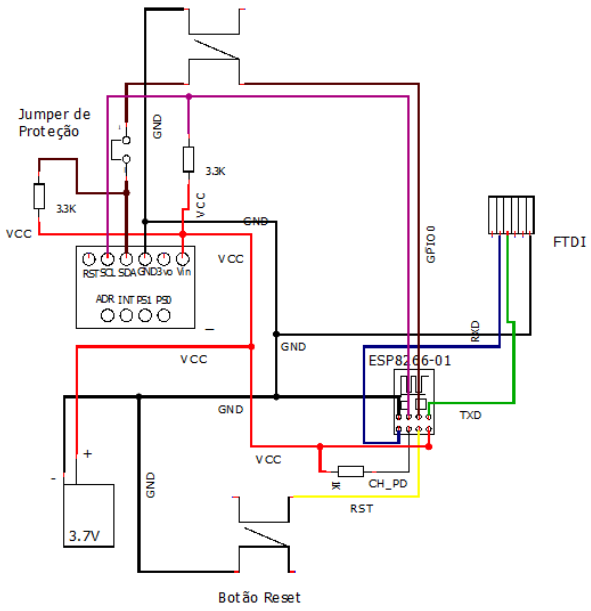

Figura 45- Circuito Bracelete A@H

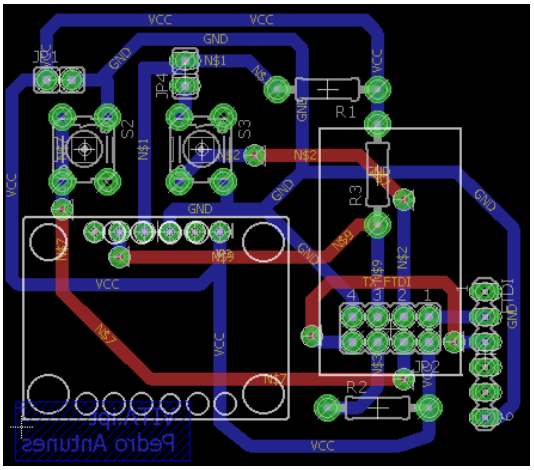

Figura 46- Placa PCB Bracelete - A@H

Foi também concebido um modelo 3D em Solidworks (Figura 47) para encapsular o circuito PCB sendo esta a estrutura da bracelete. Neste modelo é encaixada a placa PCB através de uma calha, a parte inferior a esta calha aloca a bateria e a parte superior dois três buracos para ter acesso aos botões e a entrada fêmea para a FTDI.

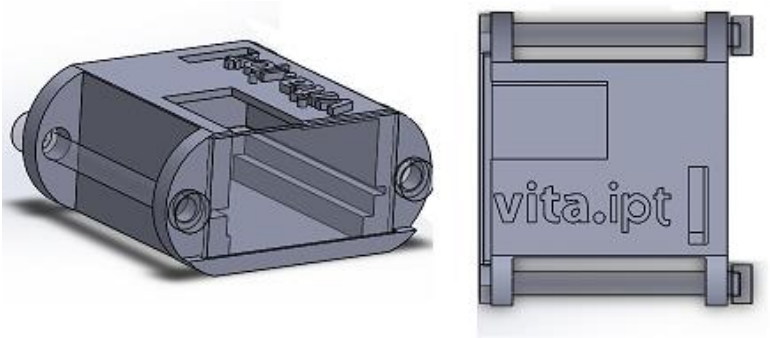

Figura 47- Modelo Solidworks - Bracelete - A@H

Este circuito possui um conector amovível para realizar a comutação entre os dois modos de funcionamento do componente ESP8266-01. Para realizar esta comutação com a Bracelete – A@H é necessário remover este conector, de seguida pressionar o botão de BOOT sem largar e por fim o botão de RST, só depois se pode deixar de pressionar o botão de BOOT. O equipamento deve então entrar em modo de programação. Para o comutar para o modo de operação insere-se novamente o conector e pressiona-se o botão de RST.

4.3.2 – Reconhecimento de Gestos – Controlo Contínuo – A@H

Nesta secção será explicado o processo de desenvolvimento do controlo de uma ventoinha através sinais de PWM provenientes de dados da Bracelete – A@H, para controlar a altura de uma bola dentro de um tubo acoplado a uma ventoinha.

Com os dados provenientes do sensor foram desenvolvidos dois algoritmos, um que utiliza os dados de aceleração e outro que utiliza os dados de orientação absoluta através de ângulos Euler.

O primeiro algoritmo desenvolvido, recorreu a dados de aceleração analisando os dados referentes ao vetor de aceleração z. Para este algoritmo recorreu-se à função das livrarias da Adafruit que devolve os vetores de aceleração inclusive de aceleração gravítica, o que faz com que para detetar um movimento ascendente ou descente este tem que ser puramente vertical, o que para além de ser inconveniente é extremamente difícil de fazer com movimentos do pulso.

O movimento da bola dentro do tubo não é diretamente proporcional ao volume de ar emitido pela ventoinha. Não foi encontrado nenhum valor de PWM que faça com que a bola fique imóvel independentemente de onde esta se encontrar no tubo por existirem perdas de ar no sistema, a bola ser afetada pela aceleração gravítica e quanto mais alta estiver a bola maior é o volume de ar necessário dentro do tubo para que este a afete , a Figura 48 tenta representar estes problemas

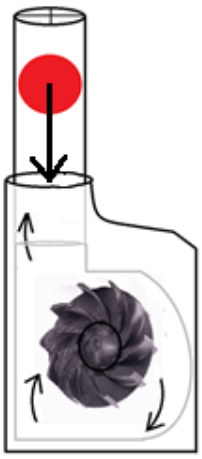

Figura 48 - Movimento de ar dentro do tubo

Foi então encontrada uma solução que recorre aos dados de orientação absoluta. No entanto, este algoritmo não mede a posição absoluta do pulso, mas verifica a posição relativa comparando as duas últimas leituras.

Neste caso, devido ao facto de estarmos a utilizar um algoritmo simples o processamento deste pode ser feito na pulseira e só depois enviado o estado pretendido. Por este motivo o equipamento que realiza a receção dos dados e produz sinal de PWM para controlar o motor também cria um SoftAP ao qual a P- A@H se conecta. Um SoftAP é um access point criado por um equipamento que não foi concebido com este intuito.

O fluxograma da Figura 49, representa o algoritmo acima referido. Este pode ser dividido em duas componentes:

- **ESP Pulseira** – A@H – CC
- **ESP Controlador** – A@H - CC

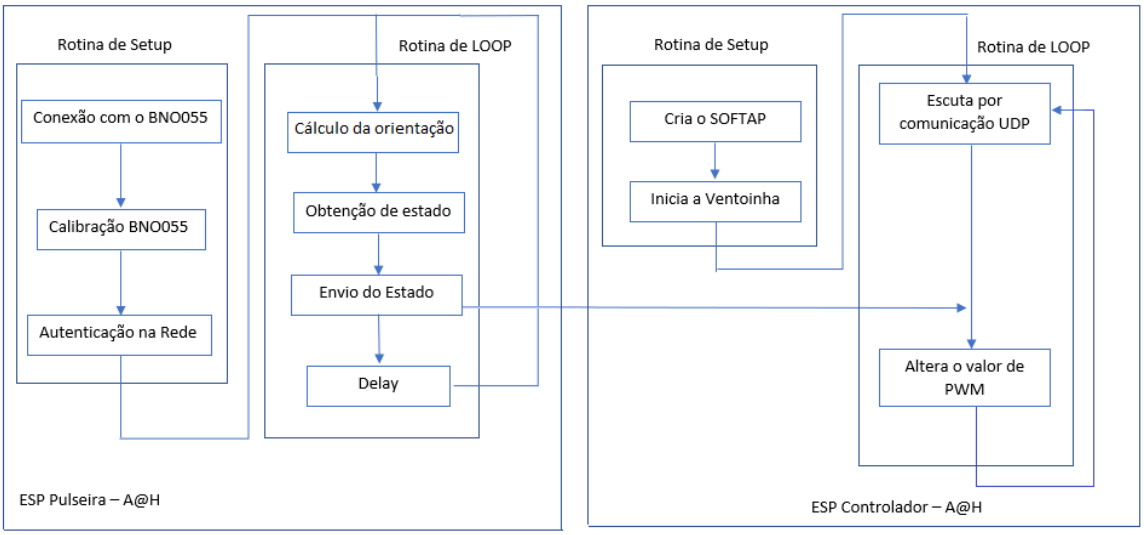

Figura 49- Fluxograma de Controlo Continuo - A@H

Como mencionado anteriormente. O código para programar as ESP8266 é de linguagem C, através do IDE do Arduino. Este código é dividido em duas rotinas, a rotina de Setup que é executada uma vez no arranque do microcontrolador e uma rotina de Loop, que é executada num ciclo infinito.

Para o código da ESP Pulseira – A@H - CC são então utilizadas as seguintes livrarias:

- **ESP8266WiFi.h** –Funções referentes à comunicação WiFi;
- **WiFiClientSecure.h** – Classe de Client com encriptação por motivos de segurança;
- **WiFi.UDP** – Classe com funções de comunicação UDP;
- **Wire.h** – Funções referentes à comunicação I2C;
- **Adafruit_Sensor.h/Adafruit_BNO055.h/utility/imumaths.h** – Funções para obter os dados do sensor BNO055;

Na Figura 50 é apresentado o código referente a variáveis globais. Estas são declaradas depois das livrarias e fora de ambas as rotinas, no início do código. A descrição de cada uma

encontra-se na imagem. A taxa de aquisição de dados dita também de quanto em quanto tempo e enviada informação para o ESP Controlador e consequentemente o período de atualização do valor de PWM. Por esta razão foi definido um valor conservativo de 200ms de modo a evitar picos de corrente constantes no motor.

```
const char* ssid = "ESP_AAH";                // Nome de rede
const char* password = "leec_m2e";           // Password da rede
int estado = 0;                              // Estado = 0 -> Para / Estado = 1 -> Sobe / Estado = 2 -> I
double event_ant = 0.0;                      // Estado Anterior

WiFiUDP Udp;                                 //Objeto da classe UDP
IPAddress ip (192,168,4,148);                //IP
IPAddress gateway (192,168,4,1);             //Gateway
IPAddress subnet (255,255,255,0);            //Mascara de Subnet

unsigned int localUdpPort = 8887;            //Portal local de UDP
#define BNO055_SAMPLERATE_DELAY_MS (200)     //Ratio de aquisiçao de valores do BNO055
```

Figura 50 - Declaração de variáveis globais Pulseira ESP - A@H

Na rotina de Setup, é realizada a escolha dos eixos quando a função *wire.begin(SDA,SCL)* é chamada. Esta função estabelece a ligação I2C entre a ESP8266 e o sensor BNO055 (Figura 51).

```
Serial.begin(115200); // inicia comunicação serie para debugging
//--------//--------////--------////--------// -CONNEÇÃO AO SENSOR --//
   Serial.println("Conexão com I2C ->");
        Wire.begin(0, 2); //funçao que define quais pinos do microcontrolador se destinam a SDA e SCL para comunicação
        if(!bno.begin()) // enquanto nao se autenticar com o sensor
                        {
                         Serial.print("Falhou"); // mensagem de erro na conecção I2C para debugging
                         while(1);
                        }

        Wire.beginTransmission(address);  // Inicia o modo de configuração
        Wire.write(0x3D);                 // --
        Wire.write(0x0);                  // --
        Wire.endTransmission();           // --
        delay(20);                        // --

        Wire.beginTransmission(address); // offset do x do acelarometro
        Wire.write(0x56);                // byte mais significativo
        Wire.write(0b0);                 // 0000 0000
        Wire.endTransmission();          // --
        delay(20);
                                  .
                                  .
                                  .
        Wire.beginTransmission(address); // Raio do acelarometro
        Wire.write(0x69);                // byte mais significativo
        Wire.write(0b1011);              // 0000 1011
        Wire.endTransmission();          // --
        delay(20);

        Wire.beginTransmission(address); // Retornar para modo Endof
        Wire.write(0x3D);
        Wire.write(0xC);
        Wire.endTransmission();
        delay(20);

        delay(1000);
        bno.setExtCrystalUse(true);
        Serial.print("Com Sucesso");
```

Figura 51- Conexão com BNO055 e Calibração Pulseira ESP - A@H

Após estabelecer a comunicação I2C é realizada a calibração do sensor com o método anteriormente mencionado. É importante referir que para realizar esta calibração é necessário colocar o BNO055 em modo de programação, e no final retornar ao modo Endof. Este processo está também presente na Figura 51.

De seguida é realizada a autenticação na rede com as credenciais declaradas nas variáveis globais, caso a rede não esteja ativa o equipamento fica neste ciclo até se poder conectar a esta. Terminando então a rotina de Setup.

```
  Serial.println("A autenticar");
  WiFi.begin(ssid, password); // Conecta à rede local com os parametros de ssid e password da rede
  //--------//--------////--------////--------// -AUTENTICAÇÃO --//
  while (WiFi.status() != WL_CONNECTED) {
                                          Serial.print(".");
                                          delay(500);
                                        }


 WiFiClientSecure client; // Objeto com o qual são chamdas funções da stack protocolar
 Serial.println();
 Serial.println("Autenticado");
}
```

Figura 52 - Autenticação na rede Pulseira ESP - A@H

A rotina de LOOP está encarregue de extrair informação dos dados fornecidos pelo sensor e de os enviar para o ESP Controlador – A@H (Figura 53).

```
void loop() {
sendacel();

delay(BNO055_SAMPLERATE_DELAY_MS);            // Atraso entre rotinas de LOOP (o mesmo valor de tempo entre aquisição dos dados)
}


void sendacel ()
{    sensors_event_t event;                  // Obtem o vetor de orientação absoluta defenido por defeito ( Angulos de Euler )
     bno.getEvent(&event);

     double temporario = event.orientation.y; // Guarda na variavel temporário o valor atual de rotação do eixo y

     if(event.orientation.y < event_ant ) {estado = 2;} // Se o valor de rotação atual for maior que o anterior o braço desceu
     if(event.orientation.y > event_ant ) {estado = 1;} // Se o valor de rotação atual for menor que o anterior o braço desceu
     if(event.orientation.y == event_ant) {estado = 0;} // Se o valor de rotação atual for igual que o anterior o braço está parado

      Udp.beginPacket("192.168.4.100", 4210);  // Inicia o Pacote UDP
      Udp.print(estado);                       // Coloca a variavel dado no Pacote
      Udp.endPacket();                         // Termina o Pacote UDP

      Serial.print("estado  --  ");            //
      Serial.print(estado);                    // Mensagem de debugging para a comunicação Serie
      Serial.println("");                      //
      event_ant = temporario;                  // Guarda o valor de rotação do eixo y anterior para comparar na proxima iteração

}
```

Figura 53- Rotina de LOOP Pulseira ESP A@H

Para o correto funcionamento deste algoritmo é necessário que o braço seja estendido na direção frontal o mais horizontalmente possível. Esta posição será considerada como uma rotação de zero graus. Caso se baixe o braço, o valor apresentado pela rotação será negativo

sendo, portanto, o movimento descrito como descida. Caso contrário, o valor apresentado será positivo e o movimento é descrito como subida (Figura 54).

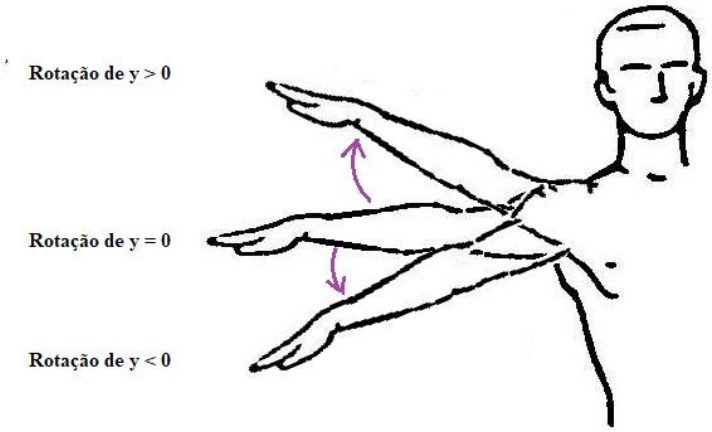

Figura 54- Exemplo de movimento para rotação de Controlo Continuo

Este algoritmo calcula a posição relativa da mão. Por outras palavras, se a mão se encontra em estado de ascendente ou descente relativamente a posição anterior. É esta informação que é usada para o controlo da ventoinha.

Para o código da ESP Controlador – AAH – CC são então utilizadas as seguintes livrarias:

- **ESP8266WiFi.h** – Funções referentes à comunicação WiFi
- **WiFiUdp.h** - Classe com funções de comunicação UDP

Antes de ser iniciada a rotina de Setup são declaradas as variáveis globais e constantes necessárias para a recessão do estado fornecido pela ESP Pulseira – AAH – CC.

```
const char rede[] = "ESP_AAH";       // Nome dado ao softAP criado
const char passrede[] = "leec_m2e"; // Password do softAP criado

WiFiUDP Udp;                         //objeto para comunicação UDP
WiFiServer server(4210);             //objeto de servidor

IPAddress local_ip (192,168,4,100); //IP Local
IPAddress gateway (192,168,4,1);     //Gateway
IPAddress subnet (255,255,255,0);    //Mascara de Sub-Rede

unsigned int UDP_port = 4210;        // Porta UDP que é escutada
int pwm = 0;                         //Valor de PWM
```

Figura 55- Declaração de variáveis e objetos - ESP Controlador - A@H

A rotina de Setup consiste na criação do SoftAP ao qual a ESP Pulseira – A@H se conecta, a abertura da porta 4210 para receção UDP e por fim a função que inicia a ventoinha(Figura 56).

```
void setup() {
              // put your setup code here, to run once:
              Serial.begin(115200);                       //comunicação Serie para Debugging
              WiFi.mode(WIFI_AP);                         //Defino o modulo de WiFi como Access Point
              WiFi.softAP(rede, passrede, 1, false);      //Define o Nome e Password da rede criada
              WiFi.softAPConfig(local_ip, gateway, subnet);//Configura os seus parametros dentro da rede

              Udp.begin(UDP_port);                        //Inicia-se a escuta da Porta 4210

              Serial.print("My IP is ");                  //Por Serie confirma o IP Local e conexão do servidor
              Serial.println(WiFi.softAPIP());            //--
              Serial.println("Configuration complete!");  //--
              Serial.println();                           //--

              startventoinha();                           // Chama a função startventoinha ();
              }
```

Figura 56 - Rotina de Setup - ESP Controlador - A@H

Na ESP8266 para se realizar esta onda quadrada é necessário definir um pino de GPIO como saída, e posteriormente através da função "analogWrite" é inserido o valor de Duty Cycle pretendido. A referência usada para obtenção do valor de Duty Cycle é a que vem por defeito no IDE do Arduino, sendo que o valor 0 corresponde a 0% de Duty Cycle e o valor 1023 corresponde a 100%.

A função que inicia a ventoinha consiste na definição da porta GPIO como saída, inicia este com a referência de GND, estabelece a frequência do PWM para 20KHz, uma frequência tolerada pelo motor, e por fim coloca um Duty Cycle de ~87% durante pelo menos dois segundos de modo a que a ventoinha injete ar suficiente no tubo para o controlo da bola ser possível.

```
void startventoinha()
{               // ---------------------------STARTUP DA VENTOINHA------------
  pinMode(4, OUTPUT); //coloca o pin 4 como saida
  digitalWrite(4,LOW); // inicia o pin 4 a referencia de GND

  analogWriteFreq(20000); // coloca a frequencia do PWM a 20kHz

  analogWrite(4,900); // Injeta ar no tubo
  delay(2000);//-------------------------------------------------------------
}
```

Figura 57- Função startventoinha ESP Controlador - A@H

A rotina de LOOP realiza a escuta da porta 4210 e quando recebe um pacote UDP interpreta o valor recebido de forma colocar o PWM da ventoinha entre um valor que sobe a bola ou a desce, em ciclo infinito (Figura 58). A variável BufferPacote[255] é declarada e

defina como uma string vazia no início de cada ciclo garantindo assim a limpeza dos dados anteriores

```
void loop()
{
   int TamanhoPacote;                          //Tamanho do pacote
  char BufferPacote[255] = "";                 // String na qual é colocada os dados do   pacote
  int temporario;                              //Variavel de conversão para int

  TamanhoPacote = Udp.parsePacket();           // Guarda nesta variavel o numero de bytes do pacote
  if(TamanhoPacote)                            // Caso se tenha recebido um pacote
  {

    int LeituraPacote;                         // Buffer do pacote
    LeituraPacote = Udp.read(BufferPacote, 255); //--
    if(LeituraPacote > 0)                      //--
      {
      BufferPacote[LeituraPacote] = 0;         //--
      }
        Serial.println(BufferPacote);
        temporario = atoi(BufferPacote);       // Conversão do pacote para inteiro
        Serial.println(temporario);
        if (temporario == 2) {pwm = 360;}      // Coloca o PWM no estado de subida
        if (temporario == 1) {pwm = 300;}      // Coloca o PWM no estado de descida
        if (temporario == 0) {pwm = 350;}      // Coloca o PWM no estado do de parado
        analogWrite(4,pwm);                    // --
    }
  delay(5);
}
```

Figura 58 - Rotina de LOOP - ESP Controlador - A@H

Será agora explicado o desenvolvimento da componente de Hardware relativa à ventoinha tubo e bola.

O sinal PWM e emitido por uma porta de GPIO do microcontrolador ESP8266-01 possui distorções e ruído que o tornavam impossível de utilizar.

Uma vez que está referido no Datasheet da ESP8266EX [10], referente a todos os modelos ESP8266, que estes têm a capacidade de produzir estes sinais e que havia uma ESP8266 – Sparkfun Thing (Figura 59) disponível foi testado na porta de número 4 e esta produzia um sinal de PWM satisfatório (Figura 60).

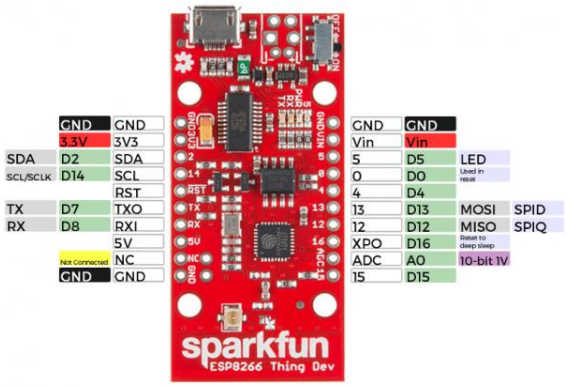

Figura 59 - Pinout ESP8266 Sparkfun Thing

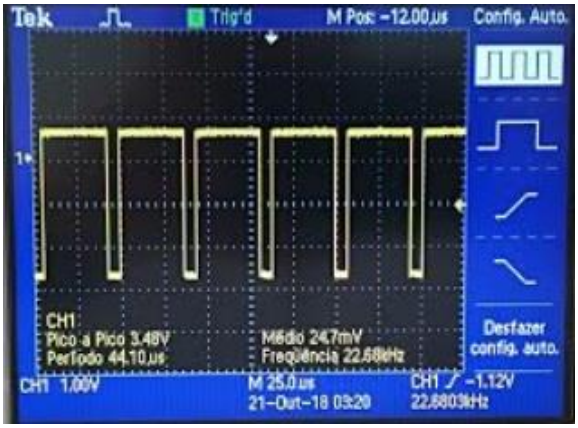

Figura 60- Sinal de PWM - ~87% Duty Cycle- Sparkfun Thing - CC - A@H

A medição dos valores de Duty Cycle foi através de tentativa erro com o intuito de encontrar o valor para o qual a bola fica teoricamente parada para referência. Os valores de movimento ascendente e descendente são o mais próximo possível deste valor para um controlo mais fluído da altura da bola.

Para realizar o controlo do motor é utilizado um transístor NPN, que é injetado com o sinal de PWM na base, e o motor é posto em série antes do coletor e no emissor são colocadas as referências para GND (Figura 61).

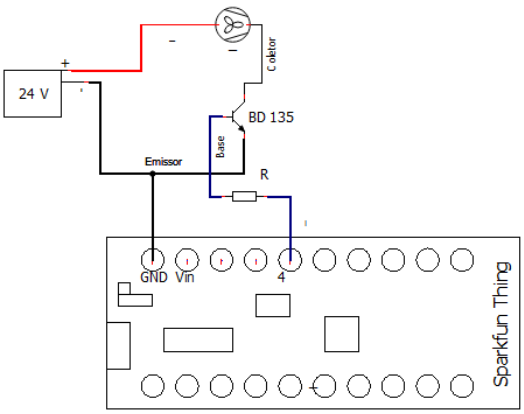

Figura 61- Circuito controlo da ventoinha - A@H

O transístor em causa é um NPN BD135 cujas características se encontram na Figura 62.

Rating	Symbol	Value	Unit
Collector−Emitter Voltage BD135G	V_{CEO}	45	Vdc
Collector−Base Voltage BD135G	V_{CBO}	45	Vdc
Emitter−Base Voltage	V_{EBO}	5.0	Vdc
Collector Current	I_C	1.5	Adc
Base Current	I_B	0.5	Adc
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	1.25 10	Watts mW/°C
Total Device Dissipation @ T_C = 25°C Derate above 25°C	P_D	12.5 100	Watts mW/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	−55 to +150	°C

Figura 62- Valores máximos tolerados pelo transístor BD 135

Foi então criado em SolidWorks um modelo em 3D que encapsula a ventoinha e redireciona o ar para o tubo (Figura 63). O modelo tem um orifício inferior sobre o qual é introduzido a ventoinha e um cone que encaixa o tubo transparente.

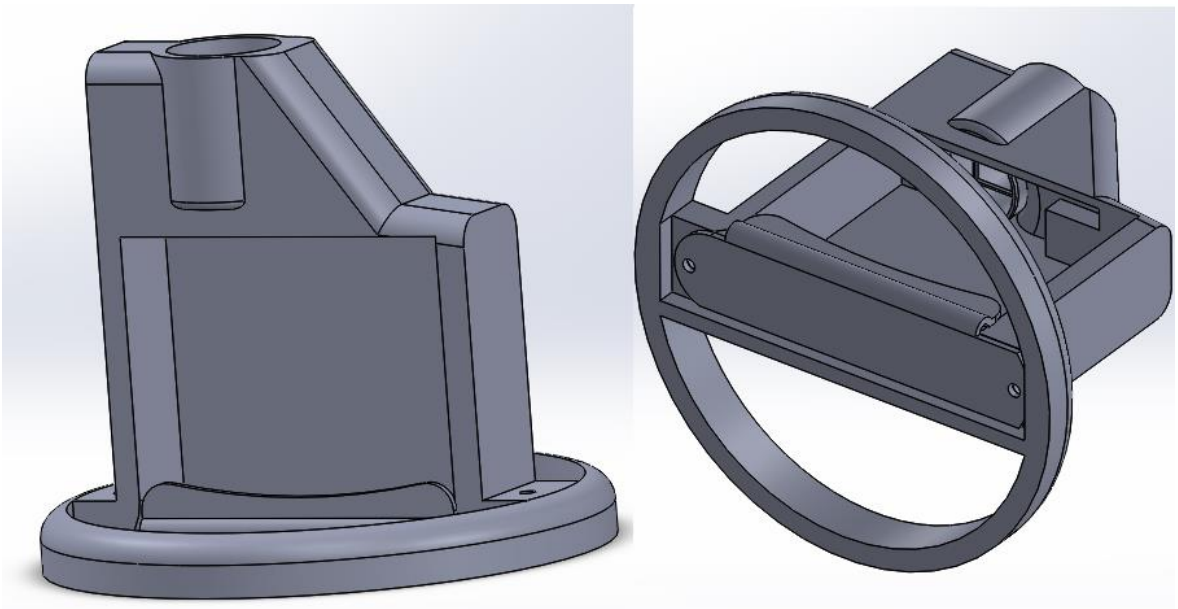

Figura 63- Modelo Solidworks - Adaptador Ventoinha Tubo - CC - A@H

4.3.3 – Reconhecimento de Gestos – Classificação de Gestos – A@H

Neste subcapítulo será explicado o desenvolvimento do modelo de classificação gestual Activity@Home (CG-A@H). Este modelo é composto por dois elementos, o cliente e o servidor. O cliente é a P–A@H que fornece os dados sensoriais referentes aos gestos e o servidor é um módulo de Python responsável pelo processamento.

O desenvolvimento do CG-A@H pode ser dividido em três etapas:

A fase de treino, onde dados dos gestos que se pretendem classificar são guardados, destes dados são extraídas características que são então inseridas no algoritmo de classificação de modo a este poder realizar previsões.

A fase de teste, onde no algoritmo de classificação treinado são introduzidas novas instâncias de gestos. Tendo de cada amostra inserida no algoritmo o gesto que se pretende classificar é possível comparar estes valores com os previstos. Consideramos a percentagem de previsões correta como a precisão do algoritmo.

A fase de execução, onde o algoritmo está constantemente a receber dados e a tentar realizar a classificação dos gestos executados.

Iremos primeiro analisar o algoritmo desenvolvido para a P-A@H, os dados fornecidos são todos os dados absolutos do BNO055, nomeadamente Aceleração Total, Ângulos de Euler e Quaterniões.

Os autores das livrarias de compatibilidade do BNO055 fornecem um exemplo deste tipo de comunicação com o IDE Processing, no entanto este IDE é maioritariamente utilizado em programas de visualização de imagens, o que não é ideal para aplicar técnicas de classificação.

O fluxograma da Figura 64 representa o algoritmo referente a P-A@H para o CG - A@H, a rotina de Setup é igual á apresentada em CC – A@H os restantes elementos do processo serão explicados de seguida.

Com a P– A@H para a Classificação de Gestos, todos os dados recolhidos foram com a referência inicial do pulso apontando para a frente.

Uma vez que estamos a realizar o reconhecimento de gestos do pulso, com apenas um sensor inercial é necessário colocar limitações a priori. Neste caso essas limitações envolvem a necessidade de o utilizador se encontrar na mesma direção e no mesmo lugar de onde iniciou a Bracelete. Caso isto não se verifique o algoritmo não consegue com sucesso realizar o reconhecimento. Em caso de se movimentar e desejar realizar esta classificação é necessário reiniciar a P – A@H conforme as condições anteriormente mencionadas.

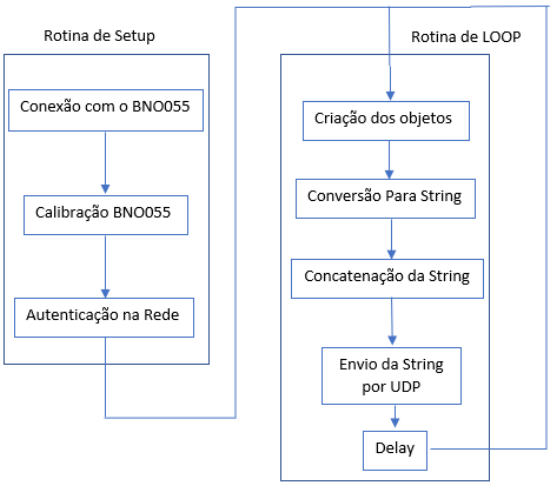

Figura 64- Fluxograma CG - Bracelete - A@H

Para o código da CG – Bracelete – A@H são então utilizadas as seguintes livrarias:

- **ESP8266WiFi.h** –Funções referentes à comunicação WiFi
- **WiFiClientSecure.h** – Classe de Client com encriptação por motivos de segurança
- **WiFi.UDP.h** – Classe com funções de comunicação UDP
- **Wire.h** – Funções referentes à comunicação I2C
- **Adafruit_Sensor.h/Adafruit_BNO055.h/utility/imumaths.h** – Funções para obter os dados do sensor BNO055
- **Stdlib**.h – Função para conversão de float para string

Na Figura 65 é apresentado o código referente a variáveis globais. Estas são declaradas depois das livrarias e fora de ambas as rotinas, no início do código. A descrição de cada uma encontra-se na imagem.

```
#define BNO055_SAMPLERATE_DELAY_MS (50) // ratio de aquisição do sensor
//--------//--------////--------////--------// -lIGACAO A INTERNET --//
const char* ssid = "VITA.IPT-WIFI";   // nome da rede WIFI
const char* password = "leec_m2e";    // password da rede WIFI
//--------//--------////--------////--------// -VARIAVEIS DE COMUNICAÇÃO --//
int kapa = 0; // contador de ciclos
//--------//--------////--------////--------// -VARIAVEIS DE SENSORES --//
byte address = 0x28; // referencia do BNO055
int i = 0;           // contador de segundos entre amostras
//--------//--------////--------////--------// -OBJECTOS --//
WiFiUDP Udp;        //Objeto de Comunicação UDP
Adafruit_BNO055 bno = Adafruit_BNO055();
```

Figura 65 - Declaração de variáveis globais e objetos CG - A@H

Mais uma vez a taxa de aquisição de dados dita o ritmo a que o algoritmo é executado. Este valor é extremamente importante para que a nível estatístico haja um espaço temporal entre amostras de modo que seja possível os reconhecimentos de padrões nos dados. Os autores de [21] afirmam que uma frequência mínima de 20Hz tem que ser garantida, ou seja 50 ms entre amostras.

Após a rotina de Setup, é executada em ciclo infinito a rotina de LOOP. Para o modelo CG-A@H existem duas rotinas de LOOP, uma que é utilizada nas fases de treino e teste (Figura 66) onde existe um espaço temporal entre a recolha de quarenta amostras de dados (referente a uma janela de tempo de dois segundos) de modo a que seja possível extrair uma amostra do gesto pretendido. A outra rotina de LOOP é utilizada na fase de execução quando se pretende realizar a classificação dos gestos continuamente.

```
void loop() {
              makestring();
              delay(50);

            }
```

Figura 67 - Rotina de LOOP - Fase de execução - CG - A@H

```
void loop() {
        if(kapa<40)
                {makestring();
                 delay(50);
                kapa = kapa +1;
                }

        if (kapa == 40)
          {while(i<2) { Serial.println(i);
                        delay(1000);
                        i++;}
           if (i == 2) { kapa = 0; i = 0;}
          }
}
```

Figura 66- Rolina de LOOP- Fase de Treino e Teste - CG - A@H

Ambas as rotinas de LOOP recorrem à função "*makestring()"*, que é responsável por criar os objetos sobre os quais os valores do sensor são amostrados, seguida da definição e declaração das variáveis temporárias referentes à conversão e envio do pacote. Começa pelos valores de quaterniões seguidos dos de aceleração e por fim de Ângulos de Euler. Estes são convertidos para uma String e separados por vírgulas de forma a estarem em formato CSV. CSV significa valores separados por vírgulas sendo um formato aceite por um leque variado de programas para interpretarem dados. Após se obter esta String a função envia o Pacote UDP e termina o seu processo.

```
void makestring ()
        {
    imu::Vector<3> acel  = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER); //acelerometros m^2
    imu::Quaternion quat = bno.getQuat();                                        //Vetor Quaterniao
    sensors_event_t event;                                                       //Ângulos de Euler
    bno.getEvent(&event);

    char tot[255] = "";             //string enviada por UDP
    double temporario = 0.0;        //Variavel para conversão
    char buff[10] = "";             //Variavel string para conversão

    temporario = quat.x();          //coloca numa variavel o valor x do vetor quaternião
    dtostrf(temporario,4,2,buff);   //converte esse valor para uma string com duas casas decimais
    strcat(tot,buff);               //concatena essa string à string que vai ser enviada por UDP
    strcat(tot,",");                //coloca uma virgula na string final para formatação CSV

          •
          •
          •

    temporario = event.orientation.y; //coloca numa variavel o valor y do vetor aceleração
    dtostrf(temporario,4,2,buff);       //converte esse valor para uma string com duas casas decimais
    strcat(tot,buff);                   //concatena essa string que vai ser enviada por UDP
    strcat(tot,",");                    //coloca uma virgula na string final para formatação CSV


      temporario = event.orientation.z;  //coloca numa variavel o valor z do vetor aceleração
    dtostrf(temporario,4,2,buff);        //converte esse valor para uma string com duas casa decimais
    strcat(tot,buff);                    //concatena essa string que vai ser enviada por UDP



    Udp.beginPacket("192.168.1.147", 8888);  //Inicia uma conexão com o servidor
    Udp.write(tot);
    Udp.endPacket();//Termina o pacote e envia-o


    Serial.println();
    Serial.print(tot);
    Serial.println();

     }
```

Figura 68- Função makestring() CG-A@H

De seguida é analisado o processo do servidor.

Existem várias linguagens de programação com as quais é possível realizar classificação, para o CG – A@H foi escolhida a linguagem Python, por esta ser freeware e ter acesso a um vasto leque de funções, de alto nível, gratuitas de classificação, processamento e visualização de dados.

Para Classificação a biblioteca escolhida foi Scikit-Learn, que fornece funções de Classificação de fácil utilização. Para a manipulação de dados foi escolhida a biblioteca Numpy, que cria e manipula com alguma facilidade arrays de várias dimensões e interage com sucesso com as funções da biblioteca Scikit learn. Para a visualização de dados foi escolhida a biblioteca Matplotlib.

O IDE escolhido para a programação é o Pycharm-Community. Foram realizados dois modelos de Python neste IDE, um referente à fase de treino e teste que tem o nome de "Receive_ESP.py" e um referente a fase de execução chamado "realtime.py". Inicialmente foi introduzido um método de multi-threading, que executa diferentes classes em paralelo, no entanto para a solução final optou-se por um método linear. Este método de multi-threading foi apenas utilizado na fase de treino em "Receive_ESP.py", onde existem três classes que são executadas em separado para realizar diferentes tarefas, estas classes são:

- Communication – Que escuta a porta UDP por informação e guarda os dados brutos num ficheiro CSV.
- Feat – Que extrai as features dos dados brutos do ficheiro criado pela classe Communication, coloca as etiquetas respetivas a cada amostra e guarda estes dados num ficheiro CSV.
- Classifica – Que cria o classificador com os dados provenientes do ficheiro criado pela classe Feat e a testa.

Para realizar a transição entre as classes é preciso ativar ou desativar as threads descomentando ou comentando, respetivamente, a linha de código referente à classe. Na Figura 69 podemos ver este processo para a ativação da classe *Communication*.

```
if __name__ == '__main__':
    q = Queue.Queue(10)

    p = Communication()
    f = feat()
    c = concatena()

    pt = threading.Thread(target=p.run, args=())
    ct = threading.Thread(target=c.run, args=())
    ft = threading.Thread(target=f.run, args=())

    ct.start()
    #pt.start()
    #ft.start()
```

Figura 69- Ativação da thread referente a classe Communication - Servidor - CG - A@H

O primeiro desafio do servidor foi receber os dados. Para isso é aberta o porto 8888 da firewall do computador, e uma vez que estamos a utilizar uma máquina virtual para correr o sistema operativo Ubunto é necessário no programa que cria esta máquina fazer o redireccionamento da porta referida (Port Forwarding). Para o CG – A@H foi utilizado o programa Oracle VM Virtualbox e esta configuração pode ser vista na Figura 70.

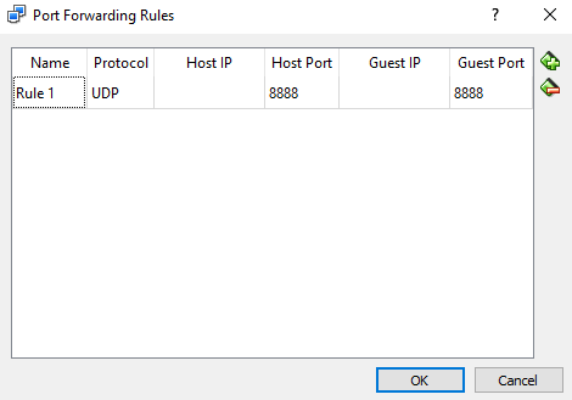

Figura 70- Port Forwarding Oracle VM VirtualBox

Na classe Communication é realizada a escuta da porta UDP. Para isto recorreu-se à biblioteca Socket. De modo a que a Bracelete – CG – A@H não tenha um IP estático (sendo, portanto, definido através do protocolo DHCP) o servidor realiza a escuta da informação da porta autorizando todos os IPs. O pacote de informação proveniente da Bracelete é de formato Byte, sendo, portanto, necessário fazer a sua descodificação com o parâmetro "utf-8" para a converter para uma String. De seguida esta String é guardada num ficheiro de formato CSV, não é necessário formatar para este modo uma vez que os dados fornecidos pela Bracelete já possuem este formato. Esta classe pode ser vista na Figura 71.

Utilizando o código da fase de treino da P–A@H, extraiu-se os dados em bruto dos dois gestos que se vista classificar. De modo a diferenciar estes dados cada classe de gestos foi guardada em bruto num CSV específico.

Para os dois gestos que se pretende classificar são necessárias três etiquetas: i) Não gesto (com etiqueta 0), Palma (com etiqueta 1) e Acenar (com etiqueta 2).

Foram recolhidos um total de quatro ficheiros CSV de dados brutos para a fase de treino. Três ficheiros com 60 amostras de cada classe para treino e um com aproximadamente 10 amostras de cada classe para teste. Os primeiros seis ficheiros são para realizar o treino e o último para testar o treino:

- "pedrooutrasbruto.csv" – 60 amostras de não gesto;
- "pedropalmabruto.csv" – 60 amostras de bater uma palma;
- "pedroacenarbruto.csv" - 60 amostras de acenar;

- “teste_treino_fin2_bruto.csv” – 10 amostras de não gesto, 10 amostras de palma e 11 amostras de acenar;

```
class Communication:
    def run(self):
        global q
        # Connection parameters
        UDP_IP = "0.0.0.0"
        UDP_PORT = 8888
        counter = 0
        sock = socket.socket(socket.AF_INET,
                             socket.SOCK_DGRAM)

        sock.bind((UDP_IP, UDP_PORT))

        while True:
            data, addr = sock.recvfrom(1024)

            teste = data.decode("utf-8")
            teste1 = teste.split(",")
            download_dir = "pedroacenarbruto.csv"
            csv = open(download_dir, "a")
            mandaparacsv = teste + "\n"
            csv.write(mandaparacsv)
            csv.close()
```

Figura 71- Classe Communication - Servidor - CG - A@H

Nas Figura 72,Figura 74 e Figura 73 podemos ver o mapeamento de aceleração vetorial dos três eixos de cada um dos gestos etiquetados

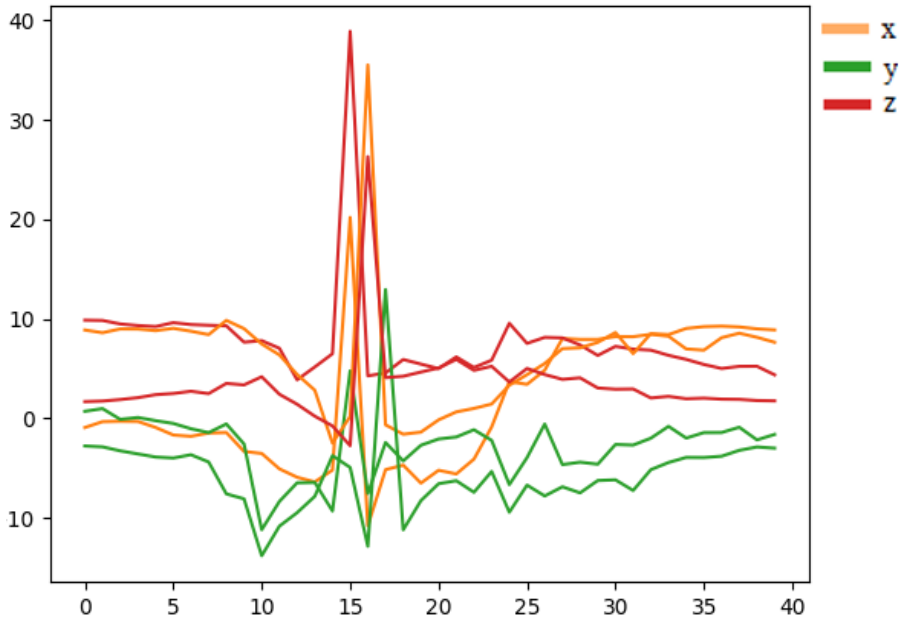

Figura 72- Duas amostras de gesto palma- dados brutos - Aceleração vetorial

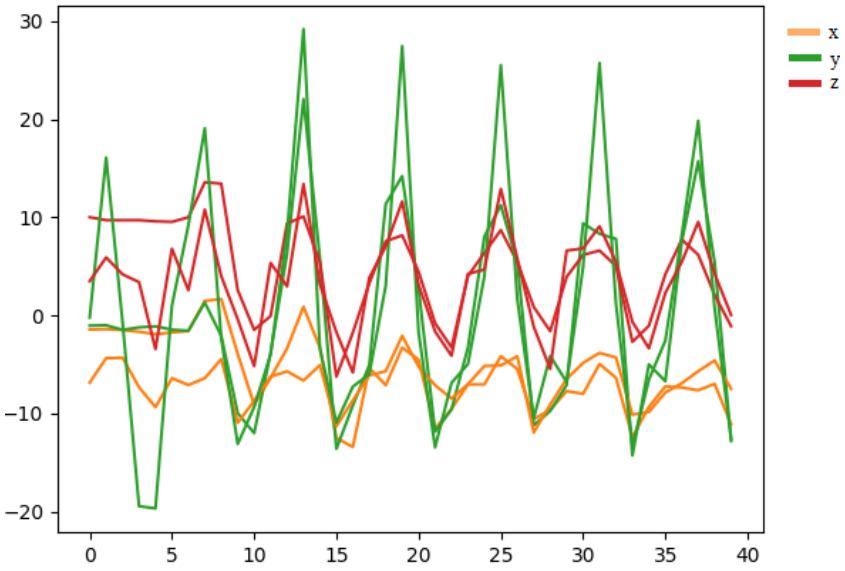

Figura 73 - Duas amostras de gesto acenar – dados brutos - Aceleração vetorial

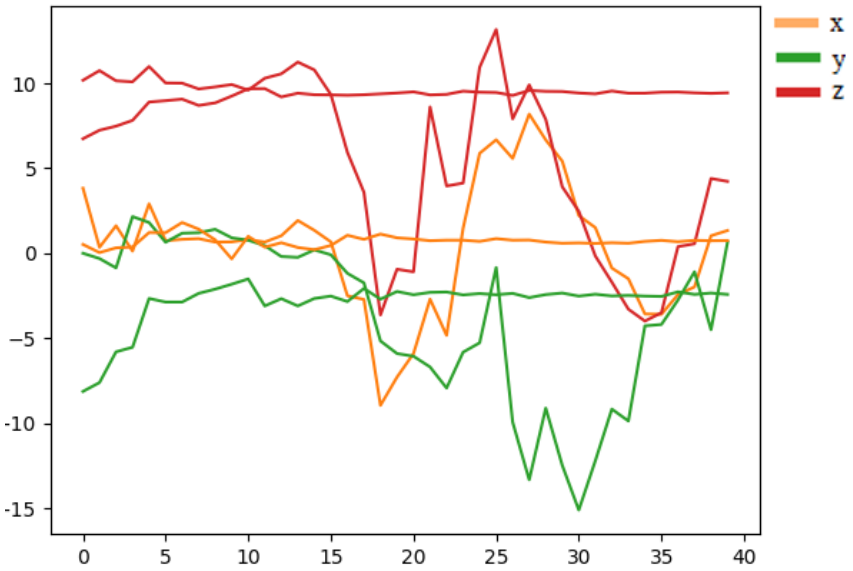

Figura 74-- Duas amostras de não gesto- dados brutos - Aceleração vetorial

. No CG – A@H as features escolhidas são de natureza estatística uma vez que estas para reconhecimento de gestos através de IMU apresentam por norma bons resultados [8]. Sendo então que as características retiradas da janela de tempo, sobre cada um dos eixos ortogonais , são:

- Média
- Mediana
- Mean
- Desvio Padrão
- Minimo
- Máximo

Na Figura 75 podemos ver um exemplo do mapeamento da feature de média de aceleração vetorial para os três eixos (media aceleração x – amarelo, media de aceleração y – verde, media de aceleração z- vermelho)

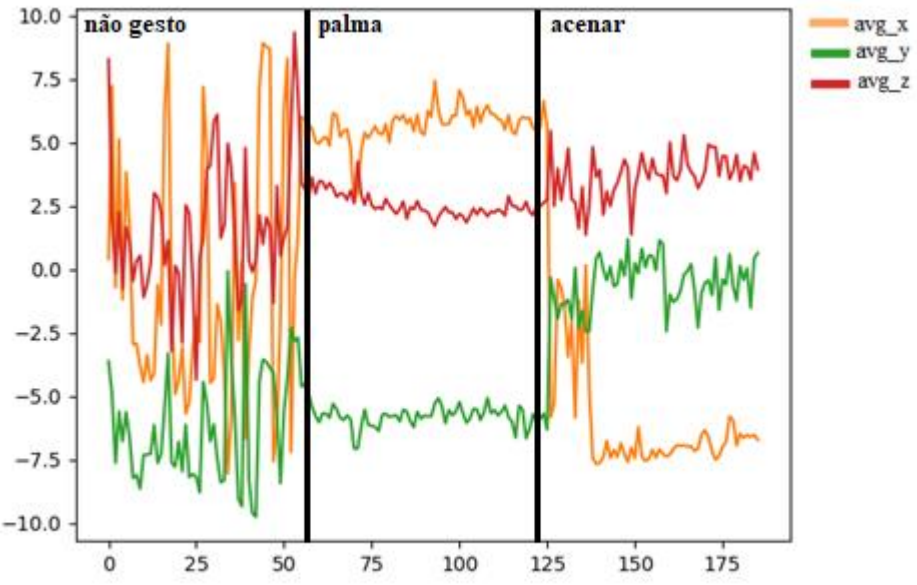

Figura 75- Features média de aceleração x/y/z - CG - A@H

A Figura 76 representa o código referente a extração destas características através das livrarias numpy.

```
avg_acelx = np.average(var[:,0])
med_acelx = np.median(var[:, 0])
mean_acelx = np.mean(var[:, 0])
std_acelx = np.std(var[:, 0])
min_acelx = np.min(var[:,0])
max_acelx = np.max(var[:,0])
```

Figura 76- Exemplo de extração de características

Apesar de a pulseira enviar todos os dados de orientação absoluta e aceleração do sensor nem todos estes foram utilizados nos vários ensaios realizados. Dos classificadores realizados serão destacados dois neste relatório.

- *Classificador A* – que recorre a extração destas características estatísticas sobre os valores de aceleração vetorial em m^2 e sobre os valores de orientação absoluta de ângulos de Euler em graus.
- *Classificador B* – que recorre a extração destas características estatísticas apenas sobre os valores de aceleração vetorial em m^2.

A Figura 77 representa a classe de Feat para o *Classificador A*. Inicialmente é lido o ficheiro de dados em bruto, e percorrido de 40 em 40 amostras retirando as features das últimas seis colunas desta matriz referentes a aceleração vetorial e orientação absoluta através de ângulos de Euler. Para cada ficheiro de dados em bruto é adicionado no final da String a etiqueta respetiva, e esta String é guardada num ficheiro de formato CSV.

```
class feat:
    def run(self):
        df = pd.read_csv('pedroacenarbruto.csv', sep=',', header=None)

        while aux_counter<=k-40:

            stringmine = ""
            i = aux_counter + 40;


            var = df.values[aux_counter:i, 4:]

            avg_acelx = np.average(var[:,0])
            med_acelx = np.median(var[:, 0])
            mean_acelx = np.mean(var[:, 0])
            std_acelx = np.std(var[:, 0])
            min_acelx = np.min(var[:,0])
            max_acelx = np.max(var[:,0])


            avg_acely = np.average(var[:, 1])
            med_acely = np.median(var[:, 1])
            mean_acely = np.mean(var[:, 1])
            std_acely = np.std(var[:, 1])
            min_acely = np.min(var[:, 1])
            max_acely = np.max(var[:, 1])

            avg_acelz = np.average(var[:, 2])
            med_acelz = np.median(var[:, 2])
            mean_acelz = np.mean(var[:, 2])
            std_acelz = np.std(var[:, 2])
            min_acelz = np.min(var[:, 2])
            max_acelz = np.max(var[:, 2])

            avg_orix = np.average(var[:, 3])
            med_orix = np.median(var[:, 3])
            mean_orix= np.mean(var[:, 3])
            std_orix = np.std(var[:, 3])
            min_orix = np.min(var[:, 3])
            max_orix = np.max(var[:, 3])

            avg_oriy = np.average(var[:, 4])
            med_oriy = np.median(var[:, 4])
            mean_oriy = np.mean(var[:, 4])
            std_oriy = np.std(var[:, 4])
            min_oriy = np.min(var[:, 4])
            max_oriy = np.max(var[:, 4])

            avg_oriz = np.average(var[:, 5])
            med_oriz = np.median(var[:, 5])
            mean_oriz = np.mean(var[:, 5])
            std_oriz = np.std(var[:, 5])
            min_oriz = np.min(var[:, 5])
            max_oriz = np.max(var[:, 5])

            stringmine = str(avg_acelx) + "," + str(med_acelx) + "," + str(mean_acelx) + "," + str(
                std_acelx) + "," + str(min_acelx) + "," + str(max_acelx) + "," + str(avg_acely) + "," + str(
                med_acely) + "," + str(mean_acely) + "," + str(std_acely) + "," + str(min_acely) + "," + str(
                max_acely) + "," + str(avg_acelz) + "," + str(med_acelz) + "," + str(mean_acelz) + "," + str(
                std_acelz) + "," + str(min_acelz) + "," + str(max_acelz) + "," + str(avg_orix) + "," + str(med_orix) + "," + str(
                mean_orix) + "," + str(std_orix) + "," + str(min_orix) + "," + str(max_orix) + "," + str(avg_oriy) + "," + str(
                med_oriy) + "," + str(mean_oriy) + "," + str(std_oriy) + "," + str(min_oriy) + "," + str(max_oriy) + "," + str(
                avg_oriz) + "," + str(med_oriz) + "," + str(mean_oriz) + "," + str(std_oriz) + "," + str(min_oriz) + "," + str(
                max_oriz) + "." + "2"

            download_dir = "treinofinal_A.csv"
            csv = open(download_dir, "a")
            mandaparacsv = stringmine + "\n"
            csv.write(mandaparacsv)
            csv.close()
            counter = 0
            aux_counter = aux_counter + 40
```

Figura 77- 1ª Fase de Treino – Classificador A - Servidor - CG - A@H

Após a classe Feat ser executada uma vez para cada ficheiro de dados em bruto (outras, palma, acenar e teste) obtém-se o ficheiro "treinofinal_A.csv". Este contém as características estatísticas de cada amostra bem como a etiqueta respetiva e o ficheiro "teste_treinofinal_A" apenas com as características sem etiqueta.

O modelo de classificação testado neste projeto é o SVM por este funcionar mesmo em casos que o número de features é maior que o número de amostras.

A normalização dos dados foi feita através da livraria "preprocessing" do scikit-learn testou-se dois tipos de normalização de dados (l1 e l2) que transformam os dados de modo que a soma de todos os dados de uma amostra seja 1. Um exemplo do efeito desta normalização nos dados que são treinados pelo algoritmo pode ser vista nas Figura 78 e Figura 79, estes gráficos estão divididos em três partes, cada uma referente a um gesto que se pretende classificar.

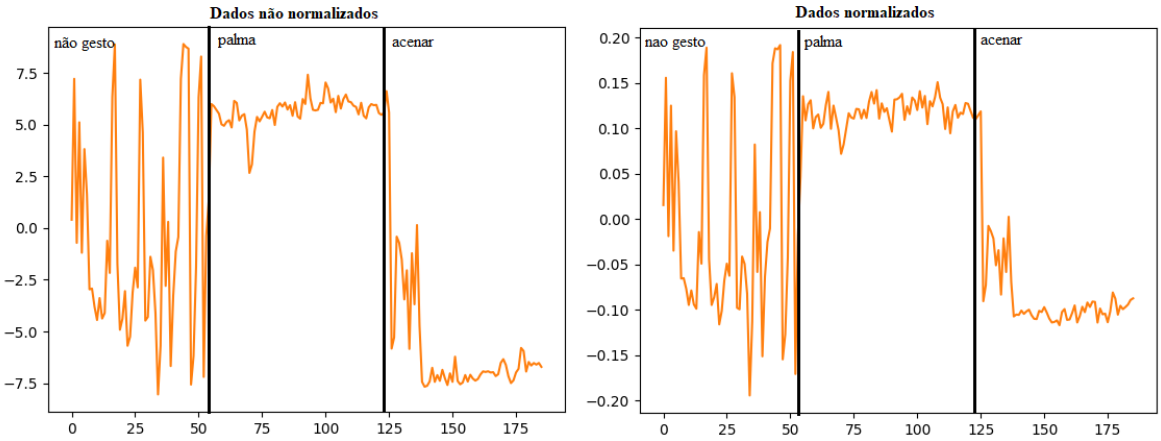

Figura 78 - Pré e pós normalização das features de avg_acelx - Servidor - CG-A@H

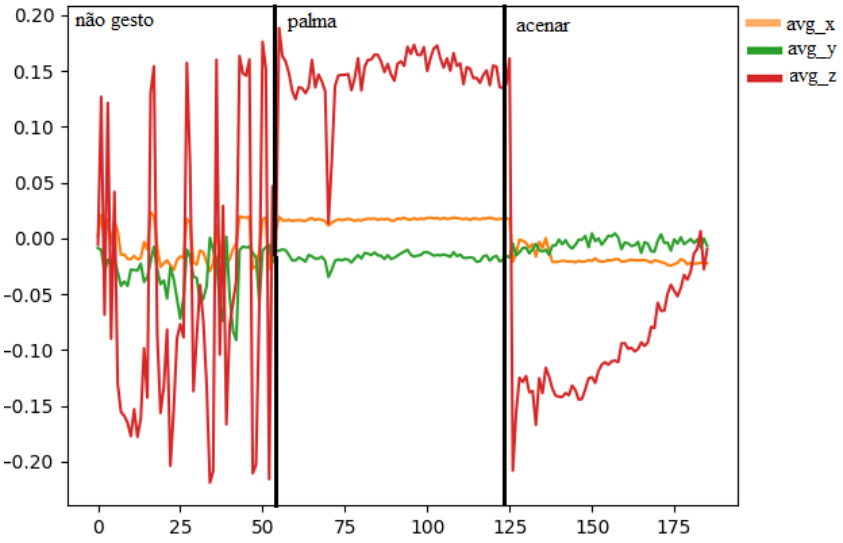

Figura 79 - Feature de avg_x/y/z após normalização - Servidor - CG-A@H

Também com o intuito de tornar o modelo mais correto foram utilizados métodos de estatísticas uni variadas para reduzir a dimensionalidade dos dados. Estes métodos calculam a correlação estatística entre as características e retiram as que apresentam uma menor correlação. Na Figura 80, o vetor boleano apresentado é referente as features que foram selecionadas através deste método automático, sendo True aceite e False não aceite, bem como uma representação gráfica deste vetor.

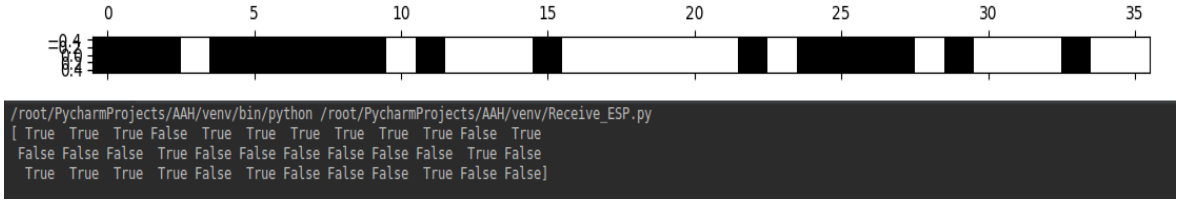

Figura 80 - Processo automatico de escolha de features - Classificador A - CG - A@H

Após se realizar este processo de tratamento de dados é treinado o modelo SVM com o ficheiro "treinofinal_A" e testado com o "teste_treinofinal_A". A Figura 81 representa o código referente à classe Classifica do *Classificador A* e a Figura 82 representa o resultado obtido. Os parâmetros alterados na função de classificação SVM foram C, referente a penalização do termo de erro, e gamma que é o coeficiente da função kernel. A função kernel utlizada, é por defeito a kernel RBF(Radial Basis Function).

```
class Classifica:
    def run(self):


        pf = pd.read_csv('treinofinal_A.csv', sep=',', header=None)
        tf = pd.read_csv('teste_treinofinal_A.csv', sep=',', header=None)

        dados = pf.values[:, :-1]
        target = pf.values[:, -1]

        select = SelectPercentile(percentile=50)
        select.fit(dados,target)
        dados_selected = select.transform(dados)

        dadosl1 = preprocessing.normalize(dados_selected, norm='l1')

        teste = tf.values

        teste_selected = select.transform(teste)
        teste_l1 = preprocessing.normalize(teste_selected,norm='l1')

        clf = svm.SVC(gamma=0.001, C=100)
        x,y = dadosl1,target
        clf.fit(x,y)

        ypred = clf.predict(teste_l1)
        ytrue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

        print(ypred)
        print(sklearn.metrics.accuracy_score(y_true=ytrue, y_pred=ypred,normalize=True,sample_weight=None))
```

Figura 81 – 2ª Fase de Treino- Classificador A - Servidor - CG - A@H

A Figura 82 apresenta os resultados da classificação com SVM. É obtida uma precisão de aproximadamente 64.5%, conseguindo realizar a classificação dos gestos com sucesso.

No entanto, não gestos não foram uma única vez classificados corretamente, apresentando então uma grande taxa de falsos positivos.

```
/root/PycharmProjects/AAH/venv/bin/python /root/PycharmProjects/AAH/venv/Receive_ESP.py
[1. 1. 1. 1. 1. 2. 1. 2. 2. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2.]
0.6451612903225806

Process finished with exit code 0
```

Figura 82 – Fase de Teste – Classificador A - Servidor - CG - A@H

No entanto ao longo do desenvolvimento do processo do ensaio A foi-se testando o classificador de modo a garantir que este funcionava corretamente. Observou-se que um modelo sem normalização nem redução dimensional e apenas com os dados de aceleração vetorial possuía precisão maior do que a do Classificador A. Este tem o nome de *Classificador B*.

De modo a facilitar o processo do *classificador B* foi realizada uma análise de correlação estatística entre as diferentes features de aceleração vetorial e o gesto que se pretende classificar. Os resultados podem ser vistos na Figura 83, estão destacadas a vermelho e verde as features que apresentam um coeficiente de correlação inferior e superior, respetivamente, a 40% com o gesto que se pretende classificar. As features a vermelho foram retiradas dos datasets analisados.

	Avg_acelx	Med_acelx	Mean_acelx	Std_acelx	Min_acelx	Max_acelx	Avg_acely	Med_acely	Mean_acely	Std_acely	Min_acely	Max_acely	Avg_acelz	Med_acelz	Mean_acelz	Std_acelz	Min_acelz	Max_acelz	Gesto
Avg_acelx	1.00																		
Med_acelx	0.99	1.00																	
Mean_acelx	1.00	0.99	1.00																
Std_acelx	0.50	0.44	0.50	1.00															
Min_acelx	0.79	0.77	0.79	0.05	1.00														
Max_acelx	0.76	0.70	0.76	0.82	0.49	1.00													
Avg_acely	-0.59	-0.58	-0.59	-0.15	-0.62	-0.37	1.00												
Med_acely	-0.46	-0.43	-0.46	-0.19	-0.48	-0.35	0.92	1.00											
Mean_acely	-0.59	-0.58	-0.59	-0.15	-0.62	-0.37	1.00	0.92	1.00										
Std_acely	-0.74	-0.73	-0.74	-0.14	-0.75	-0.43	0.85	0.69	0.85	1.00									
Min_acely	0.37	0.36	0.37	-0.02	0.43	0.18	-0.28	-0.21	-0.28	-0.60	1.00								
Max_acely	-0.65	-0.65	-0.65	-0.07	-0.70	-0.32	0.87	0.70	0.87	0.95	-0.47	1.00							
Avg_acelz	-0.15	-0.13	-0.15	0.11	-0.31	-0.02	0.54	0.55	0.54	0.33	-0.07	0.38	1.00						
Med_acelz	-0.24	-0.20	-0.24	0.03	-0.38	-0.15	0.60	0.61	0.60	0.40	-0.10	0.43	0.94	1.00					
Mean_acelz	-0.15	-0.13	-0.15	0.11	-0.31	-0.02	0.54	0.55	0.54	0.33	-0.07	0.38	1.00	0.94	1.00				
Std_acelz	-0.55	-0.57	-0.55	0.10	-0.58	-0.08	0.47	0.34	0.47	0.63	-0.44	0.61	0.21	0.22	0.21	1.00			
Min_acelz	0.02	0.02	0.02	-0.16	0.07	-0.06	0.14	0.20	0.14	-0.04	0.18	0.00	0.44	0.34	0.44	-0.42	1.00		
Max_acelz	-0.09	-0.11	-0.09	0.26	-0.19	0.35	0.22	0.19	0.22	0.23	-0.19	0.29	0.42	0.29	0.42	0.64	0.08	1.00	
Gesto	-0.40	-0.37	-0.40	0.08	-0.59	-0.18	0.79	0.65	0.79	0.84	-0.48	0.85	0.45	0.49	0.45	0.46	-0.01	0.25	1.00

Figura 83- Coeficientes de correlação entre as diferentes features de Aceleração vetorial

O *Classificador B* realiza apenas a extração de features destacadas a verde na Figura 83. A Figura 84 representa a classe de Feat para o *Classificador B.* Através desta classe obteve-se os ficheiros CSV de “treinofinal_B_2” e “teste_treinofinal_B_2”.

```
class feat:
    def run(self):
        df = pd.read_csv('teste_treino_fin2_bruto.csv', sep=',', header=None)
        aux_counter = 0
        columscheck = df.shape
        k = int(columscheck[0])


        while aux_counter<=k-40:

            i = aux_counter + 40;


            var = df.values[aux_counter:i, 4:]

            avg_acelx = np.average(var[:,0])
            mean_acelx = np.mean(var[:, 0])
            min_acelx = np.min(var[:,0])

            avg_acely = np.average(var[:, 1])
            med_acely = np.median(var[:, 1])
            mean_acely = np.mean(var[:, 1])
            std_acely = np.std(var[:, 1])
            min_acely = np.min(var[:, 1])
            max_acely = np.max(var[:, 1])

            avg_acelz = np.average(var[:, 2])
            med_acelz = np.median(var[:, 2])
            mean_acelz = np.mean(var[:, 2])
            std_acelz = np.std(var[:, 2])

            stringmine = str(avg_acelx) + "," + str(mean_acelx) + "," + str(min_acelx) + "," + str(avg_acely) + "," + s
                med_acely) + "," + str(mean_acely) + "," + str(std_acely) + "," + str(min_acely) + "," + str(
                max_acely) + "," + str(avg_acelz) + "," + str(med_acelz) + "," + str(mean_acelz) + "," + str(
                std_acelz)


            download_dir = "teste_treinofinal_B_2.csv"
            csv = open(download_dir, "a")
            mandaparacsv = stringmine + "\n"
            csv.write(mandaparacsv)
            csv.close()
            counter = 0
            aux_counter = aux_counter + 40
            print(aux_counter)
```

Figura 84-1ª Fase de Treino – classificador B - Servidor - CG - A@H

A Figura 85 representa a classe classifica para o *Classificador B* e Figura 86 os resultados obtidos.

```
class Classifica:
    def run(self):


        pf = pd.read_csv('treinofinal_B_2.csv', sep=',', header=None)
        tf = pd.read_csv('teste_treinofinal_B_2.csv', sep=',', header=None)


        target = pf.values[:, -1]
        dados = pf.values[:,:-1]


        clf = svm.SVC(gamma = 0.001, C=100)
        x,y = dados,target
        clf.fit(x,y)

        ypred = clf.predict(tf.values)
        ytrue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

       # print(ypred)
```

Figura 85- Classe Classifica - classificador B - Servidor - CG - A@H

```
/root/PycharmProjects/AAH/venv/bin/python /root/PycharmProjects/AAH/venv/Receive_ESP.py
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2.]
0.967741935483871

Process finished with exit code 0
```

Figura 86-Fase de Teste 1 - Classificador B - CG - A@H

Com uma precisão de aproximadamente 97%, o modelo de classificação do *Classificador B* é sem dúvida melhor para realizar o reconhecimento de gestos que o *Classificador A*. Sendo então escolhido para testes adicionais o *Classificador B*.

Com o intuito de avaliar melhor o *Classificador B*, foram amostrados e testados um total de noventa novas amostras com aproximadamente 30 de cada gesto, com o intuito destes serem mais representativos da população, este ficheiro CSV tem o nome "teste_treinofinal_B_ma.csv". Foi executada a classe Classifica mostrando adicionalmente a matriz de confusão (Figura 87).

```
Receive_ESP
/root/PycharmProjects/AAH/venv/bin/python /root/PycharmProjects/AAH/venv/Receive_ESP.py
[[29  0  1]
 [ 4 15 10]
 [ 0  0 31]]
0.8333333333333334

Process finished with exit code 0
```

Figura 87-Fase de Teste 2 - Classificador B - CG - A@H

Através da matriz de confusão é possível realizar uma análise de qualidade do *Classificador B*. Os valores de precisão geral e sensibilidade a cada classe podem ser calculados então:

$$Precisão = \frac{29 + 15 + 31}{90} \cong 83.3\%$$

$$Sensibilidade\ de\ palma = \frac{15}{4 + 15 + 10} \cong 51.7\%$$

$$Sensibilidade\ de\ acenar = \frac{31}{31} \cong 100\%$$

Uma vez que o objetivo é realizar a classificação continua de gestos, foi desenvolvido um modelo de Python que recorrendo à rotina de LOOP da fase de execução da P– A@H. Recebe de 50 em 50 ms os dados provenientes do cliente, guarda-os num ficheiro CSV, e de seguida lê este ficheiro de modo a que os dados estejam num formato que possa ser processado. Posteriormente, extrai as features necessárias para o *Classificador B* das quarenta amostras anteriores (40 * 0.05s = 2 segundos), e classifica (utilizando o treino do ensaio B) em tempo real. Este modelo tem o nome de "realtime.py" e pode ser visto na Figura 88. O resultado devolve uma String em tempo real caso seja detetada uma alteração na resposta do classificador perante os dados fornecidos.

Na fase de execução confirmou-se que o algoritmo tem uma boa precisão, anunciando cada vez que um gesto é reconhecido, no entanto confirmou também ter uma fraca sensibilidade no reconhecimento do gesto palma, que ao contrário do gesto acenar não preenche a janela de amostragem, este fenómeno pode ser observado comparando as Figura 72 e Figura 73.

Existem também gestos específicos que criam falsos positivos para o gesto acenar, porque as amostras retiradas de não gesto não são um bom exemplo de todos não gestos realizados por uma pessoa.

```
class realtime:
    def run(self):
        global q
        # Connection parameters
        UDP_IP = "0.0.0.0"
        UDP_PORT = 8888
        counter = 0
        sock = socket.socket(socket.AF_INET,
                             socket.SOCK_DGRAM)

        sock.bind((UDP_IP, UDP_PORT))
        pf = pd.read_csv('treinofinal.csv', sep=',', header=None)
        dados = pf.values[:, :-1]
        target = pf.values[:, -1]


        clf = svm.SVC(gamma=0.001, C=100)
        x, y = dados, target
        clf.fit(x, y)
        antiga = 3

        while True:
            data, addr = sock.recvfrom(1024)

            teste = data.decode("utf-8")
            testel = teste.split(",")
            download_dir = "continuobruto.csv"
            csv = open(download_dir, "a")
            mandaparacsv = teste + "\n"
            csv.write(mandaparacsv)
            csv.close()

            df = pd.read_csv('continuobruto.csv', sep=',', header=None)

            var = df.values[-40:, 4:7]

            avg_acelx = np.average(var[:, 0])
            med_acelx = np.median(var[:, 0])
            mean_acelx = np.mean(var[:, 0])
            std_acelx = np.std(var[:, 0])
            min_acelx = np.min(var[:, 0])
            max_acelx = np.max(var[:, 0])

            avg_acely = np.average(var[:, 1])
            med_acely = np.median(var[:, 1])
            mean_acely = np.mean(var[:, 1])
            std_acely = np.std(var[:, 1])
            min_acely = np.min(var[:, 1])
            max_acely = np.max(var[:, 1])

            avg_acelz = np.average(var[:, 2])
            med_acelz = np.median(var[:, 2])
            mean_acelz = np.mean(var[:, 2])
            std_acelz = np.std(var[:, 2])
            min_acelz = np.min(var[:, 2])
            max_acelz = np.max(var[:, 2])

            stringmine = str(avg_acelx) + "," + str(med_acelx) + "," + str(mean_acelx) + "," + str(
                std_acelx) + "," + str(min_acelx) + "," + str(max_acelx) + "," + str(avg_acely) + "," + str(
                med_acely) + "," + str(mean_acely) + "," + str(std_acely) + "," + str(min_acely) + "," + str(
                max_acely) + "," + str(avg_acelz) + "," + str(med_acelz) + "," + str(mean_acelz) + "," + str(
                std_acelz) + "," + str(min_acelz) + "," + str(max_acelz)

            download_dir = "realtime_feat.csv"
            csv = open(download_dir, "w")
            mandaparacsv = stringmine + "\n" + stringmine + "\n"
            csv.write(mandaparacsv)
            csv.close()
            tf = pd.read_csv('realtime_feat.csv', sep=',', header=None)
            temp =clf.predict(tf.values)

            if temp[1] != antiga:
                if temp[1] == 1:
                    print("palma")
                    antiga = temp[1]
                if temp[1] == 2:
                    print("acenar")
                    antiga = temp[1]
                if temp[1] == 0:
                    print("outras")
                    antiga = temp[1]
```

Figura 88- Fase de execução - Realtime.py - Servidor - CG - A@H

5 – Produtos Finais e Conclusões

5.1 – Botão de Emergência – A@H

O produto final obtido para o BT-A@H encontra-se na Figura 89. É um pendente com dimensões aceitáveis que realiza o objetivo pretendido de envia um e-mail de emergência.

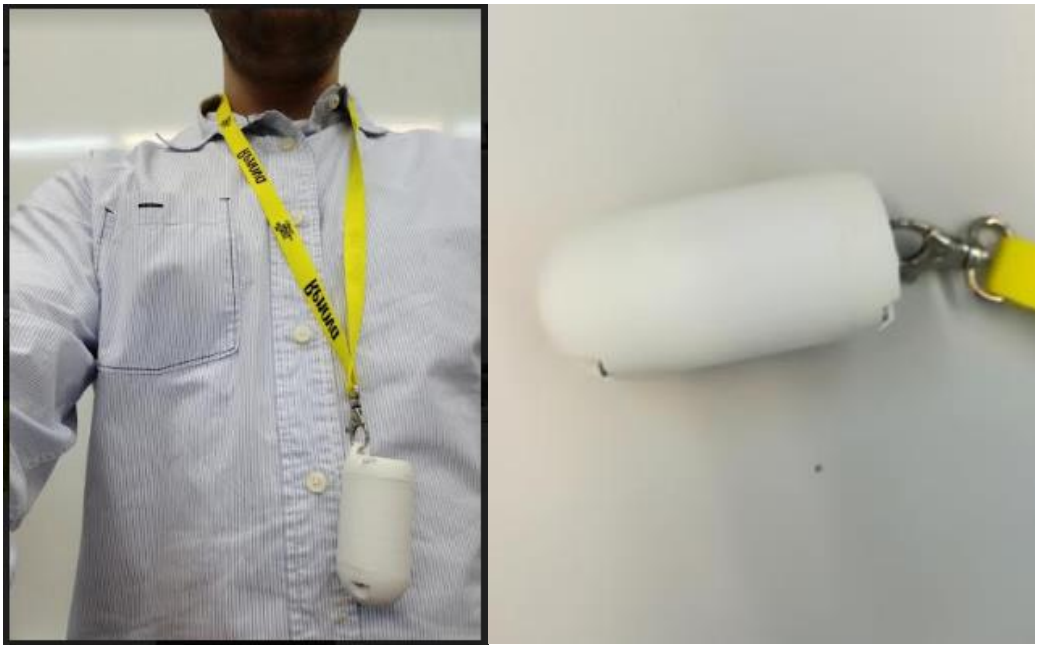

Figura 89- BT - A@H - Produto final

Apesar de ser um protótipo satisfatório, seria desejável a redução da sua dimensão total e a introdução de mensagens em redes sociais ou mensagens sms através do serviço IFTTT.

5.2 – Reconhecimento de Gestos – Bracelete A@H

O produto final obtido da bracelete A@H pode ser visto na Figura 90. É uma bracelete que apesar de todas as tentativas de redução do seu tamanho ainda é exageradamente grande. Realiza, no entanto, todas as funções para as quais foi modelada com sucesso.

Figura 90- Bracelete A@H – Produto final

Apesar de ser um protótipo funcional, seria desejável a redução do seu tamanho e a introdução de mais capacidades como recarregamento de bateria incorporado na bracelete e botão de On/Off geral.

5.3 – Reconhecimento de Gestos – Controlo Continuo

O produto final obtido para o CC – A@H pode ser visto na Figura 91, e consegue com bastante sucesso realizar o controlo de altura da bola vermelha no tubo transparente com os dados fornecidos pela Bracelete – A@H.

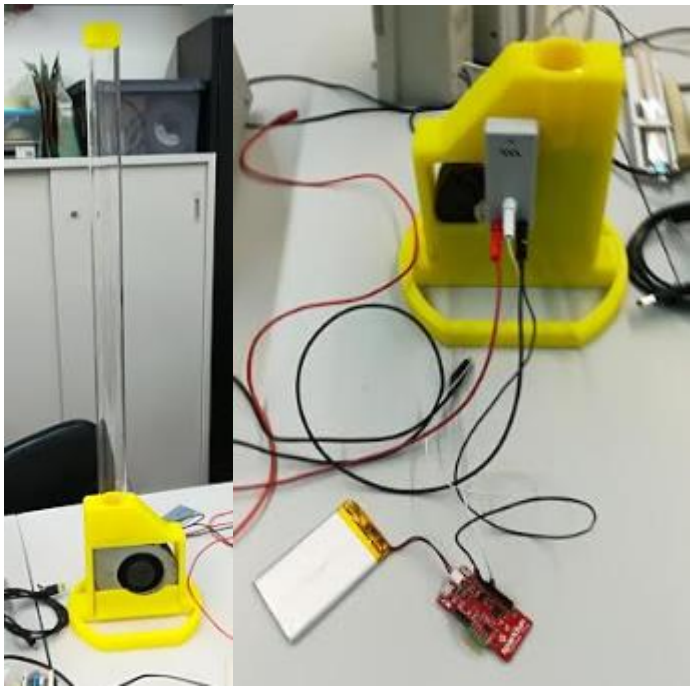

Figura 91- CC - A@H - Produto Final

Com mais tempo seria possível desenvolver um algoritmo que calcule a posição absoluta do pulso com os dados sensoriais. Adicionalmente, com sensores que consigam fornecer os dados de posição da bola no tubo seria possível realizar um controlo em malha fechada.

5.4 – Reconhecimento de Gestos – Classificação de Gestos

Por este ter sido o último de todos os objetivos e de veras o mais complicado envolvendo matérias que não foram lecionadas na licenciatura, o protótipo final obtido para a classificação de gestos através de métodos de classificação não se encontra de todo otimizado.

No entanto, dadas as condicionantes e a natureza dos assuntos envolvidos, considera-se que o protótipo final de classificação de gestos pode ser considerado um sucesso.

Este algoritmo pode ser melhorado, realizando uma análise dos dados mais aprofundada, uma extração de features muito maior do que a realizada, utilizando um universo de testes mais alargado. Podem ser explorados outros métodos de classificação.

Bibliografia

[1] [Online]. Available: https://www.gearbest.com/smart-home/pp_217257.html?vip=15888886&gclid=Cj0KCQjwrszdBRDWARIsAEEYhrcmufwRoUDYfoMSaYdNSkfCS0xPdlKDDlLEjDds1u6qu9ogjEN-hmcaAg40EALw_wcB.

[2] S. Safe. [Online]. Available: https://www.personalalarms.org/product/suresafe-alarms-1-pendant-1-wristwatch-combo/.

[3] SecureSafe. [Online]. Available: https://www.personalalarms.org/product/suresafego-24_7-connect-anywhere-alarm/.

[4] H. D. Y. O. S. Ye Gu, “Human Gesture Recognition through a Kinect Sensor”.

[5] O. B. J. V. a. D. C. Sebastial Marcel, “Hand Gesture Recognition using Input–Output Hidden Markov Models”.

[6] S. S. a. A. A. Rautaray, “Vision based hand gesture recognition for human”.

[7] “EchoFlex: Hand Gesture Recognition using Ultrasound Imaging,” 21 09 2018. [Online]. Available: https://dl.acm.org/citation.cfm?doid=3025453.3025807.

[8] R. A. E. Freixo, “Electromyography and inertial sensor-based gesture detection and”.

[9] E. Systems. [Online]. Available: https://www.espressif.com/.

[10] E. Systems, “ESP8266EX Datasheet”.

[11] Espressif, 20 09 2018. [Online]. Available: https://www.espressif.com/en/products/software/esp-sdk/overview.

[12] E. A. Core, 2018 09 20. [Online]. Available: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html.

[13] “Acelerometer-Basics,” 20 09 2018. [Online]. Available: https://learn.sparkfun.com/tutorials/accelerometer-basics.

[14] S. E. C. [JP], “Gyro Sensors - How they work and what's ahead,” 21 09 2018. [Online]. Available: https://www5.epsondevice.com/en/information/technical_info/gyro/.

[15] Sparkfun. [Online]. Available: https://learn.sparkfun.com/tutorials/i2c.

[16] microcontrolados. [Online]. Available: http://microcontrolandos.blogspot.com/2012/12/comunicacao-i2c.html.

[17] S. Eletronics, “SparkFun LiPo Charger Basic - Micro-USB,” Sparkfun Eletronics , [Online]. Available: https://www.sparkfun.com/products/10217.

[18] jean.perardel, “Emergency button : Reliable, Inexpensive & Design,” 21 9 2018. [Online]. Available: https://hackaday.io/project/27017-emergency-button-reliable-inexpensive-design.

[19] IFTTT, “IFTTT,” [Online]. Available: https://ifttt.com/discover.

[20] BOSCH, “BNO055 Datasheet,” 20 09 2018. [Online]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST_BNO055_DS000_14.pdf.

[21] *. K. T. M. K. M. V. R. K. a. J. D. J. Carlijn V. C. Bouten, “A Triaxial Accelerometer and Portable Data”.

[22] 20 09 2018. [Online]. Available: http://www.microchip.ua/wireless/esp01.pdf.

[23] “Euler Angles,” 21 09 2018. [Online]. Available: https://hepweb.ucsd.edu/ph110b/110b_notes/node31.html.

[24] I. L. C. WA. [Online]. Available: https://ilc.com.au/wp-content/uploads/2018/05/79030-ILC-FACT-SHEET_Personal-Alarm.pdf.

[25] D. A. A. W. Shamir Alavi, “Quaternion Based Gesture Recognition Using Wearable Motion Capture Sensors”.

[26] [Online]. Available: https://github.com/esp8266/Arduino.

[27] SMTP2GO. [Online]. Available: https://www.prweb.com/releases/2014/01/prweb11532651.htm.

[28] K. G.-S. a. M. S. Kylie Johnston, “Perspectives on use of personal alarms by older fallers”.