UNIVERSIDADE DE ÉVORA

ESCOLA DE CIENCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Sistema de Internet das Coisas para o Apoio aos Cuidados de Saúde da População Idosa

Pedro Miguel Borges da Palma Costa

Orientação Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, 11 de Maio de 2018

UNIVERSIDADE DE ÉVORA

ESCOLA DE CIENCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Sistema de Internet das Coisas para o Apoio aos Cuidados de Saúde da População Idosa

Pedro Miguel Borges da Palma Costa

Orientação Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, 11 de Maio de 2018

Aos eternos insatisfeitos.

Prefácio

A evolução da Internet das Coisas tem aumentado significativamente a nossa habilidade de avaliar, adquirir e atuar perante quantidades de dados nunca antes pensados. Permite cenários de utilização de dados de tecnologias de diversas áreas bem para alem das nossas necessidades, conveniências e eficiência. IoT é um conceito que reflete a conexão de um conjunto qualquer de pessoas, coisas, serviços, redes, em qualquer momento, em qualquer sitio. A IoT é uma mega-tendência na próxima geração de tecnologias. A IoT faz com que os objetos inteligentes sejam as melhores peças interligáveis no desenvolvimento dominante de sistemas inteligentes físicos cibernéticos em vários domínios, incluindo os cuidados de saúde. A área medicina e dos cuidados de saúde são áreas onde a aplicação da IoT é mais atrativa. A IoT tem o potencial de dar relevo a muitas aplicações medicas como a monitorização da saúde, programas de fitness, doenças cronicas, e cuidados de saúde a idosos. Em casa, complementada com tratamento, medicação e serviços de saúde domiciliarios é também uma área com bastante potencial da IoT. Por isto, vários dispositivos médicos, sensores, dispositivos de imagem e de diagnóstico podem ser interligados num sistema esperando assim com o tratamento da informação recolhida em tempo real reduzir custos dos cuidados de saúde, aumentar a qualidade de vida, e enriquecer a experiência do utilizador. Arritmias cardíacas ou batimento cardíaco anormal representam uma das fontes principais de mortalidade entre pacientes com doenças cardiovasculares e os Sensores Moveis Inteligentes são efetivamente confiáveis nos métodos de prevenção em muitas facetas da medicina. Apesar da utilidade clínica demonstrada continuam a ser pouco utilizados na industria dos cuidados de saúde. Estes aparelhos quando integrados nas rotinas dos cuidados de saúde aumentam a relação medico-paciente, aumentam a autonomia e o envolvimento do paciente nos seus cuidados de saúde e fornecem uma nova técnica de monitorização remota que revolucionará a gestão de cuidados de saúde.

Agradecimentos

Primeiramente e fundamentalmente agradeço aos meus pais, principalmente a minha mãe a pela motivação e apoio incondicional ao longo da minha vida académica e profissional, por reconhecerem e aceitarem a minha paixão pelas tecnologias de informação logo desde muito cedo, mesmo sem verem nem compreenderem a sua importância futura.

Em segundo lugar o agradecimento vai para o meu filho por prover ideias "inovadoras"alem do convencional com a sua ingenuidade e visão "simplificada"da vida.

Em terceiro lugar fica o agradecimento especial ao Nuno Lima, meu grande irmão, o principal e verdadeiro causador desta caminhada, e á minha companheira Isabel, pela sua paciência e apoio infinito.

Em ultimo lugar, nunca sendo o ultimo em importância, mas sim o principal a recordar, agradeço aos meus professores do curso, com o meu orientador de mestrado em especial destaque, por tudo o que me ensinaram e orientaram, em especial pela metodologia do aprender e da organização das ideias, e por sempre, mas sempre, mostrarem que tudo está ao nosso alcance e só depende do esforço para o alcançar.

Conteúdo

Lista de Figuras

Lista de Acrónimos

IEEE Institute of Electrical and Electronics Engineers

URL Uniform Resource Locator

IoT Internet of Things

CPS Cyber-Physical System

ECG Eletrocardiograma

REI Registador de Eventos Implantável

CDI Cardiodesfibrilhadores Implantáveis

BWT Bluetooth Wireless Technology

WPAN Wireless Personal Network

SIG Bluetooth Special Interest Group

ISM Industrial, Science, Medical

GHz Gigahertz

SSP Secure Simple Pairing

PIN Personal Identification Number

ECDH Elliptic Curve Diffie-Hellman

SDP Service Discovery Protocol

FCS Frame Checksum

CSMACA Carrier Sense Multiple Access Collision Avoidance

O-QPSK Offset-Quadrature Phase-Shift Keying

DSSS Direct Sequence Spread Spectrum

NIST ational Institute of Standards and Technology

AES Advanced Encryption Standard

LAN Local Area Network

WLAN Wireless LocalArea Network

Mbps Mega Bits Per Second

API Application Programming Interface

SMS Short Message Service

App Application

MQTT Message Queuing Telemetry Transport

HTTP Hypertext Transfer Protocol

TCP Transmission Control Protocol

UDP User Datagram Protoco

GSM Groupe Special Mobile / Global System for Mobile Communications

IDE Integrated Development Environment

FCM Frequência Cardíaca Máxima

BPM Batimentos Por Minuto

WBSN Wireless Body Sensor Network

BSN Body Sensor Network

LAMP Linux, Apache, MySQL, and PHP

RDBMS Relational DataBase Management System

Sumário

As tecnologias atuais e emergentes, aliadas à IoT, podem melhorar a qualidade de vida dos idosos e seus cuidados de saúde. Este tipo de aplicações informáticas voltadas para os idosos, para pessoas com restrições de saúde ou a população em geral, são cada vez mais uma preocupação e um alvo para a pesquisa em todo o mundo. As possíveis aplicações geriátricas atualmente disponíveis já são em si mesmas uma mais valia na monitorização do bem-estar, da saúde, da atividade física e vigilância da sua segurança. Estas aplicações, mesmo as mais simples, reduzem radicalmente vários problemas em populações idosas, aumentando sua independência e prevenindo emergências, podendo salvar vidas através da prevenção especialmente em casos de demência e problemas cardíacos. Pretende-se aqui de desmistificar os obstáculos possíveis e ainda existentes, dando um exemplo prático simples e de baixo custo, tendo já em conta as diferentes especificações e restrições da faixa etária alvo.

Palavras chave: IoT Apoio Cuidados Saúde Idosos

Abstract

Internet System of Things for the Aging Population Health Care Support

IoT system for the Aging Population Health Care Support

Current and emerging technologies, allied to IoT, can improve the quality of life of the elderly and their health care. This type of applications aimed at the elderly, at people with health restrictions or the general population, are increasingly a concern and a target for research around the world. The possible geriatric applications currently available are already themselves an asset in the monitoring of well-being, health, physical activity and security monitoring. These radically reduce various problems in older people, increasing their independence and preventing emergencies that can save lives through prevention, even the simplest, especially in cases of dementia and heart problems. The aim here is to demystify the possible and still existing obstacles, giving a practical example, simple and low cost, taking into account the different specificities and restrictions of the target age group.

Keywords: Iot Support Health Care Aging Population

1

Introdução

A revolução da IoT está a redesenhar os cuidados de saúde modernos com uma promissora perspetiva tecnológica, económica e social.

A IoT é um conceito que reflete a conexão de um conjunto qualquer de pessoas, coisas, serviços, redes, em qualquer momento, em qualquer sitio. A IoT é uma mega-tendência na próxima geração de tecnologias que pode ter impacto em todo o negocio. A IoT pode ser vista como a conexão de objetos inteligentes identificáveis como únicos e sistemas dentro da estrutura da Internet atual com benefícios acrescentados. Estes benefícios vão muito além dos conhecidos nos cenários da conexão maquina-maquina (M2M).

A área medicina e dos cuidados de saúde são áreas onde a aplicação da IoT é mais atrativa. A IoT tem o potencial de dar relevo a muitas aplicações medicas como a monitorização da saúde, programas de fitness, doenças cronicas, e cuidados de saúde a idosos. Em casa, complementada com tratamento, medicação e serviços de saúde domiciliarios é também uma área com bastante potencial da IoT. Por isto, vários dispositivos médicos, sensores, dispositivos de imagem e de diagnóstico podem ser vistos como dispositivos inteligentes ou objetos constituintes de uma parte de um núcleo da IoT, esperando assim reduzir custos dos cuidados de saúde, aumentando a qualidade de vida, e enriquecendo a experiência do utilizador. Além de que a IoT pode acrescentar os serviços de gestão e monitorização dos próprios equipamentos e consumíveis, recursos, medicamentos e visitas domiciliarias, sobe o posto de vista da urgência e eficiência.

1.1 IoT para os cuidados de saúde

A revolução da IoT está a redesenhar os cuidados de saúde modernos com uma promissora perspetiva tecnológica, económica e social. A IoT é um conceito que reflete a conexão de um conjunto qualquer de pessoas, coisas, serviços, redes, em qualquer momento, em qualquer sitio. A IoT é uma mega-tendência na próxima geração de tecnologias que pode ter impacto em todo o negocio. A IoT pode ser vista como a conexão de objetos inteligentes identificáveis como únicos e sistemas dentro da estrutura da Internet atual com benefícios acrescentados. Estes benefícios vão muito além dos conhecidos nos cenários da conexão maquina-maquina (M2M).

Figura 1.1: Interligações de objetos inteligentes na área da saúde

A IoT faz com que os objetos inteligentes sejam as melhores peças interligáveis no desenvolvimento dominante de sistemas inteligentes físicos cibernéticos (CPS)[1] em vários domínios, incluindo os cuidados de saúde[VS17], como nos exemplos da figura 1.1.

A área medicina e dos cuidados de saúde são áreas onde a aplicação da IoT é mais atrativa. A IoT tem o potencial de dar relevo a muitas aplicações medicas como a monitorização da saúde, programas de fitness, doenças cronicas, e cuidados de saúde a idosos. Em casa, complementada com tratamento, medicação e serviços de saúde domiciliarios é também uma área com bastante potencial da IoT. Por isto, vários dispositivos médicos, sensores, dispositivos de imagem e de diagnóstico podem ser vistos como dispositivos inteligentes ou objetos constituintes de uma parte de um núcleo da IoT, esperando assim reduzir custos dos cuidados de saúde, aumentando a qualidade de vida, e enriquecendo a experiência do utilizador. Além de

[1]CPS: Sistemas físicos cibernéticos são sistemas que são construidos pela integração transparente de algoritmos computacionais e componentes físicos que transformam a forma como as pessoas interagem com os sistemas de engenhos, tal como a Internet transformou a forma como interagem com informação, inovando vários sectores[Cor17]

que a IoT pode acrescentar os serviços de gestão e monitorização dos próprios equipamentos e consumíveis, recursos, medicamentos e visitas domiciliarias, sobe o posto de vista da urgência e eficiência [VS17], como nos exemplos da figura 1.2.

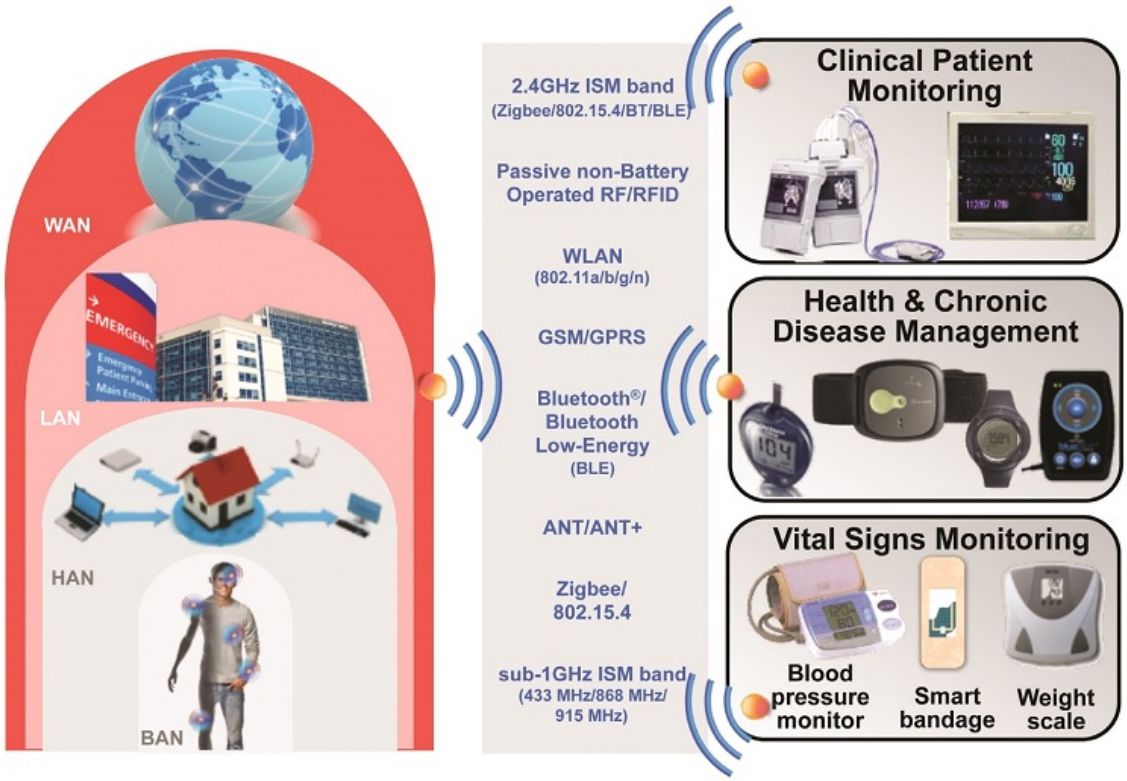

Figura 1.2: Aplicações IoT na área da saúde

Espera-se que redes munidas de tecnologias sem fios atuais apoiem doenças cronicas, pré-diagnósticos, monitorização em tempo real e emergências medicas. Portais, servidores para medicina e bases de dados para cuidados de saúde terão papeis vitais na criação de fichas medicas e enviarem instantaneamente serviços de saúde aos intervenientes autorizados.

A topologia da IoT na saúde baseia-se na agregação de diferentes redes e tecnologias sem fios, na ligação de vários elementos transparentes aos serviços de saúde, como exemplo: uma rede de computação heterogénea reúne quantidades de dados vindos de sensores tais como medidores de pressão arterial, temperatura corporal, nível de oxigénio, eletrocardiograma, etc. Estas tecnologias estão constantemente a ser melhoradas e podem ser desenvolvidas e exploradas em qualquer campo da medicina. Contudo, a integração requer uma infraestrutura valida fiável, persistente para cada aplicação na vasta variedade de doenças e problemas relacionados com as mesmas contribuindo também nos métodos preventivos. É essencial a colaboração entre médicos, pacientes, engenheiros e a industria "sem fios"para o desenho e otimização de sistemas "sem fios"acessíveis[ACA+14].

1.2 Oportunidades

Arritmias cardíacas ou batimento cardíaco anormal representam uma das fontes principais de mortalidade entre pacientes com doenças cardiovasculares. Por este motivo requerem-se aparelhos que monitorizem os batimentos cardíacos por um largo período de tempo e que detetem automaticamente certas

arritmias[Pol13].

No estudo "The Framingham Heart Study"concluiu-se que estimativa da variabilidade do batimento cardíaco pela monitorização em ambulatório oferece mais informação para um prognostico para alem da fornecida pela avaliação dos riscos tradicional. [TVM+94]

Também, mais especificamente, o estudo "Predictive Value Clinicand Ambulatory Heart Rate for Mortality in Elderly Subjects With Systolic Hypertension"positivamente associou o o ritmo cardíaco com o pior prognostico para a totalidade da mortalidade cardiovascular e não-cardiovascular entre 2293 pacientes com hipertensão sistólica isolada[Pal02].

Devido versatilidade do sistema, é possível acrescentar muitos outros aparelhos, nunca excluindo casos específicos dentro da variada gama de aparelhos disponíveis nos dias de hoje, como por exemplo entre muitos, o REI[2], no caso de doentes que tenham que ter uma monitorização constante[JKZ+10].

Outra possibilidade a titulo de exemplo seria uma inovação recente que abrange indivíduos que monitorizam constantemente os seus níveis de glucose é um sensor ocular de glucose não invasivo. Apesar de ainda estar em desenvolvimento, muitas companhias estão a criar lentes de contacto que integram um sensor de diagnostico que detetam os níveis de glucose e os transmite para um aparelho pessoal[ACA+14].

Só os Estados Unidos gastam aproximadamente 75% do orçamento anual de 2 triliões de dólares ara doenças cronicas, que representam 7 em cada 10 mortes anuais [Pal02] e os Sensores Moveis Inteligentes são efetivamente confiáveis nos métodos de prevenção em muitas facetas da medicina. Apesar da utilidade clínica demonstrada continuam a ser pouco utilizados na industria dos cuidados de saúde. estes aparelhos quando integrados nas rotinas dos cuidados de saúde aumentam a relação medico-paciente, aumentam a autonomia e o envolvimento do paciente nos seus cuidados de saúde e fornecem uma nova técnica de monitorização remota que revolucionará a gestão de cuidados de saúde[ACA+14].

1.3 O Sistema Proposto

O sistema aqui proposto pretende ser uma contribuição para a comunidade. De forma a que haja proveito e continuidade no contributo, está disponível e pronto para ser compilado no Android Studio 3.0 em: `https://github.com/tasdsf/HRM4MiBand2`

A implementação do sistema pretende-se que seja de baixo custo, sem que esta motivação limite nem condicione as funcionalidades e a utilidade proposta, que permita o acesso aos dados em tempo real, ao histórico, que esta informação seja disponibilizada de uma forma tal que possa servir de apoio a quem tenha de tomar decisões em tempo útil.. As funcionalidades que visam a monitorização e rastreio do ritmo cardíaco são possíveis e fáceis de implementar contemplando apenas uma banda e um smartphone com uma aplicação desenvolvida para esse objetivo.

A deteção de valores anormais, enviando mensagens de alerta das anormalidades em tempo real, são também funcionalidades possíveis da aplicação tendo em conta que esta já fará a monitorizarão e rastreio, bastando apenas que receber os dados e proceder á sua análise use o sistema de envio de mensagens SMS comum ao conceito de smartphone.

O registo das leituras, criando gradativamente um histórico configurável, de fácil adaptação a variados

[2]REI: "O Registador de Eventos Implantável (REI), é um pequeno dispositivo, que é colocado de forma subcutânea no tórax do doente, e permite obter um traçado como de um eletrocardiograma (ECG) registado automaticamente, graças à implementação de algoritmos e parâmetros de deteção semelhantes aos de Cardiodesfibrilhadores Implantáveis (CDI) e Pacemaker". [JKZ+10]

caso e tipos de análise, de fácil acesso, ótima disponibilidade obriga a que estas leituras sejam armazenadas exteriormente ao smartphone, não sendo esta questão preocupante visto que é também parte do conceito de smartphone ter um ou mais formas de conexão á Internet e formas de uma aplicação entregar dados a uma outra plataforma para esta os armazenar, tratar e disponibilizar de seguida.

2

Estado da Arte

A IoT é um conceito que reflete a conexão de um conjunto qualquer de pessoas, coisas, serviços, redes, em qualquer momento, em qualquer sitio.Tem um papel fundamental na fácil interligação da tremenda variedade de aplicações existentes e integração das futuras, devido à sua essência e propósito. A sua evolução tem aumentado significativamente permitindo cenários nunca antes pensados na utilização de dados, de tecnologias, de diversas áreas, bem para alem das nossas necessidades, conveniências e eficiência. É um mercado que tem crescido exponencialmente [sta] em que se prevê que se gastem perto de 6 Triliões nos próximos 5 anos[Gre16]. São inúmeras as possíveis utilizações em destaque as aplicações na área da saúde onde, por exemplo, os Estados Unidos gastam perto de 75% do seu orçamento anual de 2 triliões de dólares para doenças cronicas, que representam 7 em cada 10 mortes anuais [Pal02].

2.1 Definição de IoT

IoT, Internet of Things, Internet das Coisas em Português, Foi mencionada pela primeira vez em 1999, por Kevin Ashton numa apresentação na Procter & Gamble.

É um sistema de computação de dispositivos inter-relacionados, mecânicos e digitais, objetos, animais ou pessoas equipadas com identificadores únicos e com a habilidade de transferir dados através de uma rede sem requerer a interação entre humanos ou humano e computador.

Embora o conceito não ter sido nomeado até 1999, a Internet das coisas tem estado em desenvolvimento á décadas. A primeira aplicação na Internet, por exemplo, foi uma maquina de Coca Cola no Universidade do Instituto Carnegie Melon no inicio doas anos 80. Os programadores puderam conectar-se á maquina através da Internet, verificar o estado da maquina e determinar se existiria ou não uma bebida fresca á espera deles, caso decidissem se deslocar até á maquina.

Uma 'coisa', em IoT, pode ser uma pessoa com um implante de monitorização do coração, um automóvel com sensores embutidos de alertar a pressão dos pneus baixa, um animal com identificador, ou qualquer objeto que possa obter um endereço IP com a habilidade de transferir dados por uma rede informática.

"Os computadores de hoje — e, por isso, a Internet – estão quase totalmente dependentes da informação fornecida pelos seres humanos. Quase a totalidade dos aproximadamente 50 peta bytes (um peta byte são 1024 terabytes) dos dados disponíveis na Internet foram adquiridos e criados por seres humanos pela escrita, por fotografias tiradas ou um código de barras.

O problema é que, as pessoas têm tempo limitado, atenção e precisão limitadas – o que significa que não são muito bons a adquirir dados sobre coisas no mundo real. Se tivéssemos computadores que soubessem tudo o que havia para saber acerca das coisas – usando dados adquiridos sem qualquer nossa ajuda – nos seriamos capazes de seguir e contar tudo e reduzir enormemente o desperdício, perdas e custos. Nos saberíamos quando as coisas necessitariam de substituição, reparação ou ser retiradas e se eram recentes ou já tinham passado o seu melhor." [RW16]

2.2 Cenários de Utilização

São inúmeras as possíveis utilizações, são alguns exemplos:

- parques inteligentes, iluminação inteligente dentro e fora de casa, consumos de energia de edifícios/-casas,
- deteção de fogos e alertas para ambiente de risco, alarmes e segurança de áreas, casas, edifícios,
- rastreio condições, localização de embalagens ou produtos antes e pós-venda/transporte, diagnósticos de qualidade, utilização, garantia, funcionamento,
- monitorização das condições de saúde de pessoas, localização.[lib16]

Em diversas áreas é uma mais valia, como por exemplo:

- **Na robótica**, larga eficácia no processamento de inúmeros sensores combinada com a sofisticação de atuadores mecânicos e redes que os conectam na produção de maquinas que tem como objetivo a sua completa e/ou quase completa autonomia no controle das suas ações no mundo físico. Os veículos autónomos são um exemplo
- **Em Engenharia mecânica**, onde os sensores e atuadores usados por armas, robot e exames, diagnósticos médicos remotos e aplicações cirúrgicas que fazem uso de 'tecnologias sensoriais hapticas' são grandes exemplos. [Koe17]

- **Nos Cuidados de Saúde**, onde sensores embutidos no vestuário coletam informação acerca do estado de saúde do paciente, comunicando com tecnologia sem fios, com instalações conduzidas por médicos e investigadores e recebendo novas instruções, em tempo real, para o controlo da dosagem de medicamento a administrar baseada nas necessidades do paciente.
- **Na Ecologia e Ciências do Ambiente**, onde uma extensa rede de sensores e atuadores podem controlar e gerir qualidade do ar e da agua, onde a previsão do tempo e a medição do clima são usados para melhorar a produção agrícola.
- **No controlo de processos de produção**, uma das mais primeiras e mais maduras utilizações da IoT, evolui através de avanços de aplicações na robótica, desenho assistido por computador (CAD) e produção assistida por computador (CAM). Assim como muitos Data Centers trabalham autonomamente, no futuro pensamos que também será o controlo de produção e fornecimento[Koe17].

2.3 A evolução da IoT

A evolução da Internet das Coisas tem aumentado significativamente a nossa habilidade de avaliar, adquirir e atuar perante quantidades de dados nunca antes pensados. Permite cenários de utilização de dados de tecnologias de diversas áreas bem para alem das nossas necessidades, conveniências e eficiência. A evolução da IoT trouxe desenvolvimentos que desfocam a separação entre humanos e maquinas, como se pode verificar nos exemplos anteriores. A evolução implora por uma nova forma de pensar, novos sistemas de valores.

A indiferenciação cada vez maior entre as maquinas e humanos, a cada passo, trás algum desconforto. Existem questões que é inaceitável continuarem indefinidas:

- Que políticas precisamos para definirmos quem controla os dados e quem tem acesso a eles e a sua atualização?
- Até que ponto poderemos substituir os nossos órgãos por (em breve) órgãos mecânicos, por necessidade ou para aumentar as nossas capacidades físicas?
- O que é só domínio humano e ate´ onde é aceitável esta 'ajuda' das maquinas?
- Quem é o dono dos dados? As questões éticas e morais que rodam a proteção dos dados já se levantam hoje em dia.
- "A questão mais pertinente levantada pela evolução da IoT é qual o aspeto da humanidade numa era da IoT", diz Harvey Koeppel.

O problema com o pensamento de como nós protegemos os dados gerados pela Internet das Coisas e como determinamos a propriedade dos dados da IoT é que nós estamos a tentar aplicar as mesmas regras e metodologias a um contexto totalmente diferente para qual as velhas regras e metodologias nunca foram concebidas ou intencionadas.

Nós somos os homens e mulheres das cavernas dos tempos modernos lutando com o problema de quem é o dono da luz, calor, sons e aromas gerados pela fogueira. Talvez esta seja a justificação para que a IETF, quando definiu a IPv6 em 1998, estava a confiar na predição de Robert Metcalfe da 3Com em 1995 que "a Internet brevemente entrará numa espetacular super-nova e em 1996 colapsará catastroficamente.", a exemplo de outras previsões históricas equivocadas pela falta de visão: Thomas Watson da IBM previu em 1943 que haveria "um mercado mundial talvez para cinco computadores", ou ainda Ken Olsen da Digital

Equipment Corporation previu em 1977 que "Não há razão alguma para alguém querer ter um computador na sua casa.". [Hic17]

Talvez superando o problema de visão, fazendo uma análise realista acerca da IoT, daCosta e Byron dizem em 2013: "A arquitetura da Internet original foi criada muito antes da comunicação entre biliões de aparelhos muito simples como os sensores e as suas aplicações nunca previstas" e acerca da dimensão da IoT afirmam que: "Será ainda maior do que era esperada" [daC13]

2.4 Tecnologias Sem Fios

Nos últimos anos o mundo tem cada vez mais adotado a mobilidade em todos os sentidos. As comunicações tem facilitado e fomentado este aumento desimpedindo entraves e restrições do passado, por exemplo, em certos tipos de trabalhos o trabalhador pode trabalhar, partilhar e enviar de informação instantaneamente em qualquer sitio que esteja coberto por rede sem fios, como no caso da área dos cuidados de saúde, centros médicos e hospitais, segurança, entre outras, onde a mobilidade dos intervenientes é inevitável. alternativa a sítios onde é difícil de instalar cablagem, ou os custos são inferiores ou também onde a possibilidade de cortes ou erosão é significativa. facilidade e velocidade de instalação.

Todo o tipo de rede sem fios tem as suas vantagens e desvantagens, e diferenças nos serviços que prestam, trafico e cobertura. com a evolução dos computadores e das telecomunicações tem se vindo a substituir as redes com fios tradicionais como ponto de acesso á Internet por varias redes sem fios de grande cobertura (por exemplo 3G, 802,22), redes locais sem fios metropolitanas (IEEE 802.16), redes sem fios locais ('WI-FI' IEEE 802,11), redes via satélite, redes Bluetooth etc.. [Mis10]

Com tão grande diversidade, não é prudente fazer generalizações sobre o desempenho das redes sem fios. A maioria tem princípios e compromissos comuns, estão sujeitas a critérios comuns de desenpenho e restrições. As vantagens e desvantagens relacionam-se com a cobertura, mobilidade, flexibilidade, custos de instalação, manutenção e segurança. Para compreender mesmo que superficialmente o universo das tecnologias sem fios é necessário estar disposto a aprofundar sempre mais um pouco o universo esotérico dos acrónimos de 3 ou mais letras

2.4.1 Bluetooth

Bluetooth wireless technology (BWT) ou tecnologia Bluetooth[1] sem fios em português, foi desenvolvida em 1994 na Ericsson na Suíça. O objetivo era eliminar a necessidade de cabos especiais nas ligações entre aparelhos. A comunicação por luz infravermelha já existia na altura mas esta tecnologia requer que os pares emissor-recetor consigam detetar a luz emitida um pelo outro, ou seja um campo curto aberto sem interrupções. Por este motivo a Ericsson optou por usar um radio de baixo consumo e custo em cada aparelho, possibilitando a conexão sem fios de aparelhos através de paredes e de materiais que não seja metálicos. Depois disto, a ideia evoluiu para uma conexão sem fios que conecta vários aparelhos em simultâneo numa rede sem fios pessoal, a wireless personal network (WPAN). Por causa do potencial ilimitado da BWT, foi formado em 1998 o Bluetooth Special Interest Group (SIG) para desenvolver as especificações
indexBluetooth IEEE 802.15Bluetooth IEEE 802.15.

[1] O significado do nome Bluetooth: É um tributo ao rei viking Harald Blåtand que pacificamente uniu a Dinamarca e a Noruega. O rei Harald gostava de comer bagas azuis que conferiam uma coloração azulada aos seus dentes o que levou à sua alcunha de Dentes Azuis, Bluetooth em inglês.

Funcionamento

Os aparelhos com radio Bluetooth operam na banda 2.4GHz, a qual é livre sem restrições em todo o mundo (exceto em França), conhecida também pela banda "Industrial, Science, Medical"(ISM). Esta banda encontram-se entre os 2.400GHz e 2.483GHz. Os aparelhos com radio Bluetooth usam 79 frequências de 1-megahertz (entre 2.402GHz e 2.480GHz) desta banda como mostra a figura 2.1, e usam uma técnica chamada "frequency hopping"(salto de frequência) para minimizar a espionagem ou escutas não autorizadas e interferência de outras redes que usem esta . Com esta técnica, os dados são divididos em pequenos pacotes. O transmissor e recetor trocam um pacote de dados numa frequência, e então saltam para outra para trocar outro pacote, repetindo este processo até todos os dados serem transmitidos. Estes saltos ocorrem até 1600 vezes por segundo, muito mais rapidamente que os outros dispositivos que usam esta banda, o que significa que qualquer interferência de um outros dispositivo durará cerca de $1/1600$ de segundo, o tempo que demora mais um salto de frequência, o que confere aos dispositivos BWT uma alta imunidade a interferência. Existem 3 classes de dispositivos BWT cada qual com um alcance máximo: classe 1 = 100 metros; classe 2 = 50 metros; classe 3 = 10 metros.

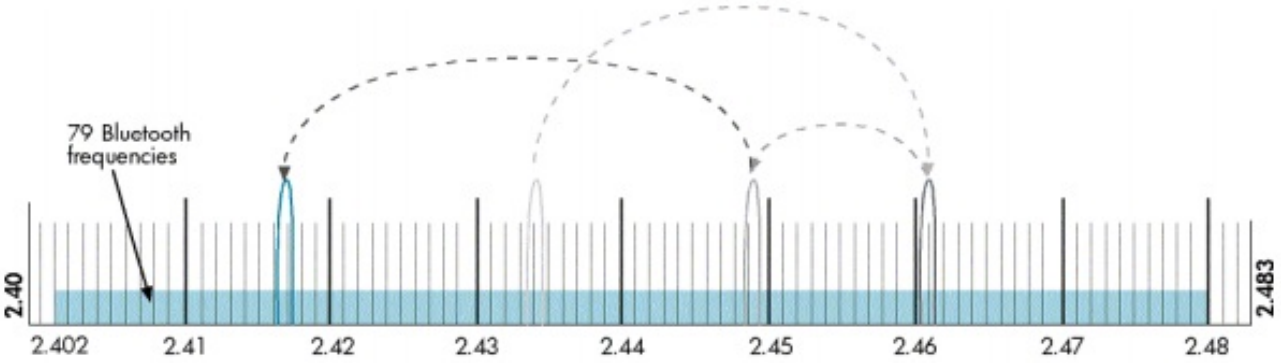

Figura 2.1: Banda de frequências ISM (GHz)

topologia

A ideia dos dispositivos BWT é formar pequenas redes sem fios numa topologia de rede chamada piconet que consiste em um dispositivo determinar o padrão da "frequency hopping"que controla todo o tráfego, que fica apelidado de dispositivo primário ou mestre, e os restantes, apelidados de dispositivos secundários ou escravos, sincronizam os seus sinais para o mesmo padrão. A comunicação entre os dispositivos secundários tem de passar pelo dispositivo mestre.

Cada piconet tem o seu padrão diferenciando assim o seu sinal dos sinais de todas as outras piconets. No exemplo da figura 2.2, um dispositivo atua como mestre e gere as comunicações dos escravos. Só pode existir um mestre e até 255 escravos, mas apenas somente 7 escravos poderão estar ativos em simultâneo, sendo os restantes denominados com "parked"(estacionados), podendo estes se tornar ativos rapidamente (trocando com outro dispositivo ativo caso seja necessário) e comunicar na piconet, fazendo assim com que todos os possíveis 255 dispositivos estejam virtualmente conectados. [IBM00]

Pode-se formar uma scatternet. Forma-se uma scatterednet quando duas ou mais piconets estão conectadas por um dispositivo BWT em comum, ficando este como o mestre e assim sincronizando as comunicações entre os mestres das piconets com o seu padrão de sinal, como no exemplo da figura 2.3 . [HP04]

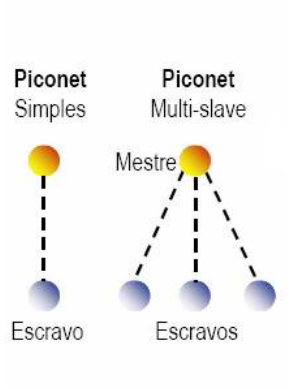

Figura 2.2: Piconet

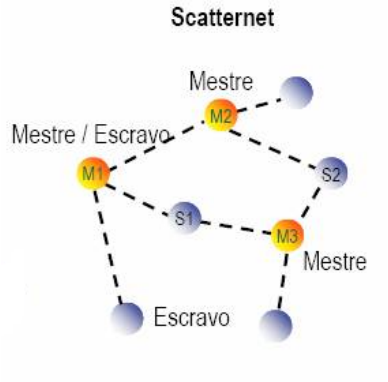

Figura 2.3: Scatternet

Segurança dos BWT

São 3 os mecanismos de segurança implementados nos dispositivos BWT: autenticação, autorização e encriptação. Na autenticação é verificada a identidade do dispositivo BWT que quer conectar-se. Após a fase da autenticação com sucesso é autorizado o acesso aos serviços a que tem permitidos. A informação é encriptada num formato que só pode ser decifrada apenas por um dispositivo que tem a mesma chave de encriptação. A segurança nos dispositivos BWT tem 4 modos, e o modo define como é permitido um dispositivo BWT conectar-se:

- Modo 1 - Sem segurança, em que todos podem usar o dispositivo. Por defeito numa impressora publica, por exemplo, sem autorização nem encriptação. Mas se o dispositivo que se conecta iniciar num modo seguro - emparelhamento, autenticação e encriptação - este participará no mesmo modo.
- Modo 2 - Segurança ao nível de serviços; O acesso está dependente dos serviços que foram autorizados, e somente a esses serviços. Após a autorização pode usar encriptação e autenticação.
- Modo 3 - Segurança ao nível da ligação; As medidas de segurança são iniciadas antes da ligação. O dispositivo tem de estar emparelhado antes de poder estabelecer a ligação e trocar informação.
- Modo 4 - Segurança ao nível de serviços, idêntico ao modo 2, com Secure Simple Pairing (SSP) como extensão; neste modo os procedimentos de segurança são iniciados depois da conexão física e lógica. Este modo usa uma chave Elliptic Curve Diffie-Hellman (ECDH) que substitui o acordo prévio

de uma chave, por exemplo um Personal Identification Number (PIN), para gerar a chave de ligação, embora também permita este método. Os requisitos de segurança para os serviços protegidos pelo modo de segurança 4 tem de ser classificados por um destes níveis:

- Nível necessária fornecer uma chave de ligação autenticada através de uma ligação segura
- Nível 3: É necessária fornecer uma chave de ligação autenticada
- Nível 2: É necessária fornecer uma chave de ligação sem ser autenticada
- Nível 1: Não é necessária segurança
- Nível 0: Não é necessária segurança. (Somente permitida para SDP)

Este modo requer encriptação para todos os serviços exceto no Service Discovery Protocol (SDP), o serviço de procura de dispositivos e serviços BWT perto do dispositivo BWT que inicia a procura.

2.4.2 ZigBee

ZigBee é uma norma de um protocolo para redes sem fios de baixo consumo tipo mesh. Os criadores desta norma fundaram em 2002 a the ZigBee Alliance. Hoje contam com mais de 225 companhias associadas. A norma de redes sem fios ZigBee encaixa num mercado que simplesmente não está preenchido por outras tecnologias sem fios como se pretende mostrar na figura 2.4. Enquanto a maioria das normas de tecnologias sem fios procuram mais velocidade e mais funcionalidades, a ZigBee tem como objetivo taxas baixas de transmissão de dados, baixo consumo e as funções indispensáveis para que seja utilizável por micro-controladores de 8 bits, com soluções também para 16bits e 32 bits. Outras tecnologias pretendem o melhor fornecimento e Internet e transmissão multimédia de alta definição, a ZigBee procura comunicar entre sensores e controladores de luz ou enviar o valor da temperatura de um termostato. A ZigBee é bastante barata e aparenta ficar ainda mais barata ao longo do tempo.[Gis08]

A ZigBee utiliza 16-bit CRC em cada pacote de dados transmitido, apelidado de "Frame Checksum"(FCS) garantindo assim que todos os bits estejam corretos. 16-bit CRC é um código de deteção de erros comum para detetar mudanças acidentais nos dados [Gie96].

Cada pacote reenviado até três vezes, podendo perfazer 4 transmissões. se o pacote não foi chegou ao destino depois da quarta transmissão, ZigBee informa o transmissor deste acontecimento para que este resolva. Também fornece confirmação ponto-a-ponto automática para que as aplicações saibam se um pacote de informação foi recebido pelo nó recetor. As aplicações estão livres de varias tarefas visto que ZigBee faz os reenvios e filtra pacotes duplicados. Numa rede mesh pode formar redes AD-HOC, estender o alcance através de multi-hop, até descobrir novos caminhos para passar a informação em caso de falha de um ou mais nós da rede. Neste tipo de rede a informação pode passar de um nó para outro na rede não importando a distancia, nem o numero de nós pelo meio, por os quais a informação terá de passar.[Gis08]

ZigBee usa Carrier Sense Multiple Access Collision Avoidance (CSMACA) para aumentar a fiabilidade. Antes de transmitir, ZigBee verifica o canal e transmite quando o canal estiver livre.[PD12]

Esta norma é bastante atual, é uma tecnologia de radio robusta construida pela IEEE a partir dos seus mais de 40 anos de experiência. Usa o apelidado Offset-Quadrature Phase-Shift Keying (O-QPSK) e Direct Sequence Spread Spectrum (DSSS), para um excelente desenpenho resultante da combinação destas tecnologias resulta em ambientes com uma baixa relação entre sinal e ruído. [Bes16] [Hay14]

Em adição ao seu baixo ciclo de trabalho, os rádios e micro-controladores que podem suspender-se. Um nó na rede ZigBee não precisa de estar em constante contacto com a rede para continuar em rede, a rede é frequentemente bastante silenciosa. Por exemplo, um sensor de temperatura só precisa de enviar o valor

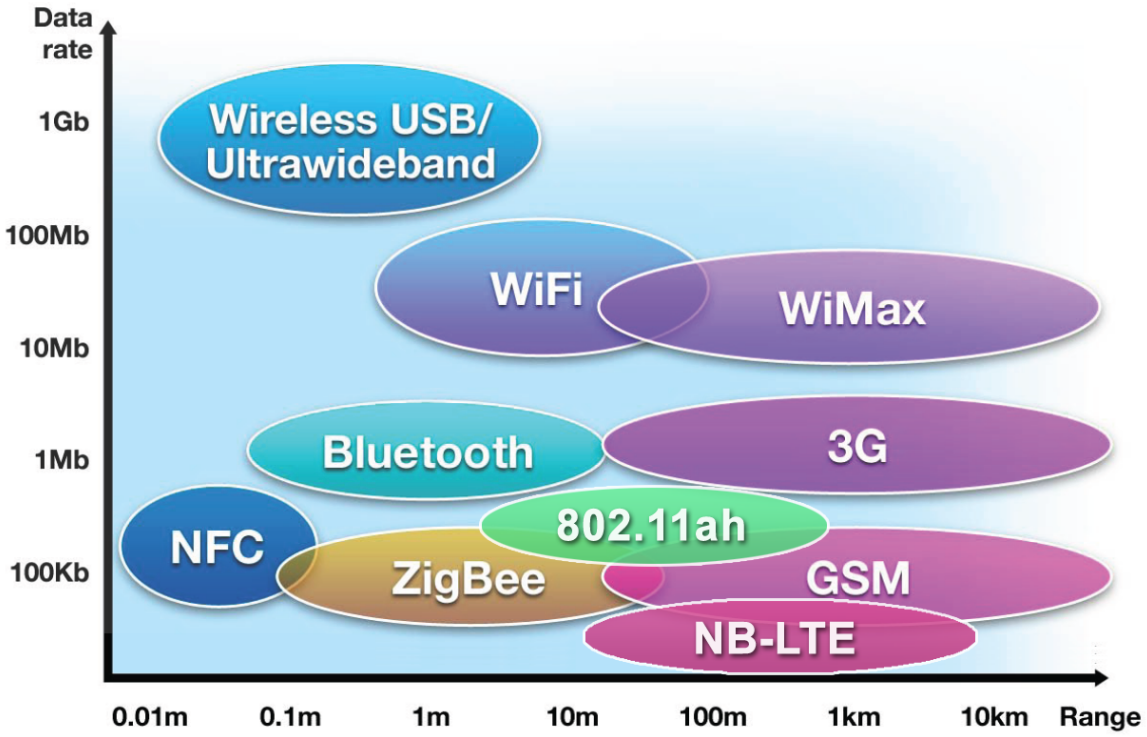

Figura 2.4: Comparação entre ZigBee e outras tecnologias sem fios

da temperatura uma vês por hora ou quando a temperatura muda subitamente, ou um sensor de presença pode ser ativado uma dezena ou uma centena de vezes por dia.

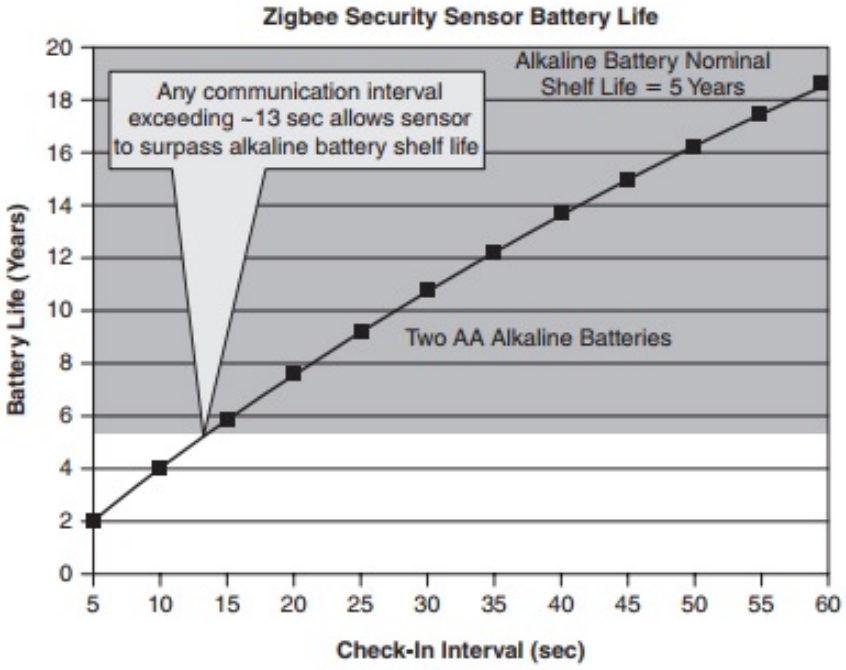

Figura 2.5: Exemplo de consumo de um nó na ZigBee

A titulo de exemplo, com um calculo simples podemos ver na figura 2.5 demonstrado que o consumo um nó ZigBee, que comunique uma vez a cada 13 segundos ou menos, é tão baixo que a carga de uma pilha chegaria para o alimentar para além do tempo de validade da própria pilha. Normalmente, dispositivos ZigBee podem ter alimentação por 5 anos com um par de pilhas AA.[Gis08]

ZigBee usa a norma da National Institute of Standards and Technology (NIST) Advanced Encryption Standard (AES). Esta norma AES-128, é uma codificação de blocos, que encripta e desencripta pacotes de informação de uma forma muito difícil de descobrir [DR02], por isso é uma das mais conhecidas e respeitadas normas. Muitos vendedores oferecem soluções ZigBee em varias areas, incluindo casa, comercial, industrial, medica e outras. As especificações ZigBee podem ser consultadas gratuitamente em http://www.zigbee.org .

2.4.3 WI-FI IEEE 802.11

A rede sem fio IEEE 802.11, que também é conhecida como rede Wi-Fi ou wireless, foi uma das grandes novidades tecnológicas dos últimos anos. Atuando na camada física, o 802.11 define uma série de padrões de transmissão e codificação para comunicações sem fio, sendo os mais comuns: FHSS (Frequency Hopping Spread Spectrun), DSSS (Direct Sequence Spread Spectrum) e OFDM (Orthogonal Frequency Division Multiplexing). Atualmente, é o padrão de fato em conectividade sem fio para redes locais. Como prova desse sucesso pode-se citar o crescente número de hotspots e o fato de a maioria dos computadores portáteis novos já saírem de fábrica equipados com interfaces IEEE 802.25. A rede IEEE possui como principal característica transmitir sinal sem fio através de ondas!

Os hotspots presentes nos centros urbanos e principalmente em locais públicos, tais como universidades, aeroportos, hotéis, restaurantes, entre outros locais, estão mudando o perfil de uso da Internet e, inclusive, dos utilizadores de computadores.[Gei10]

As versões da IEEE 802.11

Atualmente a versão 802.11n da norma IEEE 802.11 de rede locais sem fios (WLAN) foi homologada em Setembro do ano 2009 e permite que os sistemas 802.11 forneçam uma velocidade muito mais alta e alcance maior que antes. A maioria dos fabricantes agora produzem equipamentos com o versão 802.11n como primeira solução para redes WLAN. Redes WLAN baseadas nas versões anteriores ficarão obsoletas.[Gei10]

Comparando as versões

- **Velocidade de transmissão de dados:** As normas 802.11a e 802.11g são quase cinco vezes mais rápida que a original 802.1b, mas a 802.11n é cinco vezes mais rápida efetivamente.Na maior parte dos casos, qualquer equipamento WI-FI (seja 802.11g ou 802.11n) será mais rápida que a velocidade da Internet na sua casa, mas a velocidade extra da 802.11n poderá valer a pena se tiver de transmitir sinal de vídeo via rede sem fios
- **Preço:** O equipamento 802.11g (padrão hoje em dia) tem estado no mercado à bastante tempo, por isso o seu preço está bastante baixo. Os equipamentos 802.11n poderão ser bastante mais caros.
- **Alcance do sinal de radio:** As redes sem fios 802.11a tendem a ter um alcance máximo menor que as redes 802.11b e 802.11g. A distancias efetivas variam conforme o espaço e a construção onde estão implementadas. Por causa da nova tecnologia MIMO, a 802.11n pode chegar a ter duas ou mais vezes maior alcance.

- **Interferência no sinal de radio:** A frequência de radio usada por ambos equipamentos, 802.11b e 802.11g é usada também por outros equipamentos, numa casa podem existir muitos tais como fornos micro-ondas, telefones portáteis, aparelhos que comunicam por Bluetooth, causando por vezes interferências na rede. Poucos outros dispositivos usam agora a frequência de radio da 802.11a. Os equipamentos 802.11n pode usar ambas embora use mais frequentemente a mais povoada 2,4GHz.
- **Interoperabilidade:** Porque a 802.11a e a 802.11b/g usam bandas diferentes, eles não podem comunicar pela pela mesma banda de radio. Porem, vários fabricantes tem produtos que podem operar com ambos equipamentos 802.11a e 802.11b/g simultaneamente. Por contraste, o equipamento 802.11g foi desenhado para ser retrocompatível com o equipamento 802.11b, ambos operam na mesma banda. O 802.11n é retrocompatível com todas as três normas anteriores. No entanto o retrocompatível com 802.11a é somente o equipamento 802.n que opera na banda 5GHz.

  [BH11]

Síntese da IEEE 802.11

Em junho de 1997, a IEEE finalizou a norma inicial para redes sem fios IEEE 802.11. Esta norma especifica frequência de trabalho nos 2.4GHz com taxas de transmissão de 1Mbps e 2Mbps. A norma inicial 802.11 define duas formas de ocupar a modulacao do espetro:"frequency hopping"(salto de frequência: 802.11 FHSS) e DSSS (Direct Sequence Spread Spectrum) que é a sequência direta de espalhamento do espectro.

Nos finais de 1999, a IEEE publicou dois suplementos à norma 802.11: 802.11a e 802.11b. A IEEE 802.11b é a extensão referente à taxa de transmissão de dados da 802.11 DSSS inicial, oferecendo até 11Mbps na banda 2.4GHz.

A norma 802.11a oferece até 54Mbps usando nos 5GHz a Orthogonal frequency-division multiplexing (OFDM) consistem na multiplexação por divisão de frequência com multiplas sub portadoras ortogonais permitindo o transporte da informação paralelamente. A norma 802.11a tem uma grande variedade de altas velocidades de transmissão: 6,9,12,18,24,36,48 e 54Mbps e é imprescindível todos os produtos terem as velocidades de 6Mbps,12Mbps e 24Mbps. [Gei10]

3

Sistema Proposto

Propõe-se um sistema com funcionalidades com alguma utilidade para um pré-diagnósticos e que para isso faça monitorização em tempo real, gerando alertas assim que detetar situações que mereçam atenção . Escolhem-se estas primeiras funcionalidades por poderem vir a ser uma ajuda fundamental nas emergências medicas. Esta ajuda pode ser dada ás equipas de emergência através de portais, servidores para medicina e bases de dados para cuidados de saúde. Estes portais poderão ter papeis vitais na criação de fichas medicas muito mais completas. O sistema poderá requerer instantaneamente serviços de saúde aos intervenientes autorizados, ou somente alertar a situação a uma ou mais pessoas encarregadas dessa responsabilidade.

3.1 Descrição Geral

Como pequeno exemplo motivador da criação de sistemas neste âmbito, e desmistificador dos entraves e dificuldades que rondam a criação deste tipo de sistemas, propõe-se criar um sistema de monitorização continua do ritmo cardíaco, englobando apenas um mínimo necessário, como exemplo de uma das possibilidades, contempla-se apenas um Smartphone vulgar, uma conexão á Internet simples e comum, e uma banda desportiva também comum com a funcionalidade de medir o ritmo cardíaco.

A principal razão desta configuração como primeiro exemplo é a relativa acessibilidade a um smartphone, com ligação incluída á Internet via WI-FI, GSM ou ambas,e a uma banda desportiva de baixo custo que tenha a funcionalidade de medir o ritmo cardíaco.

As funcionalidades que serão implementadas visam a monitorização e rastreio do ritmo cardíaco e deteção de valores anormais, enviando mensagens de alerta das anormalidades em tempo real, criando gradativamente um histórico configurável, de fácil adaptação a variados caso e tipos de análise, de fácil acesso, ótima disponibilidade.

A escolha destas funcionalidades baseiam-se no facto de estas serem as mais importantes dentro das capacidades de um sistema como o proposto, destacando o facto de que uma mensagem de alerta, no limite, podem salvar uma vida ou evitar sequelas consequentes de uma situação anormal detetável pela variação ou valor do batimento cardíaco, dentro cenário descrito anteriormente nos capítulos anteriores, nomeadamente no ponto 1.2 na pagina 3 desta dissertação. Têm em conta também que os dados podem ser sempre relevantes e ajudar na elaboração de um diagnostico, parecer medico ou ainda no estudo ou prevenção de uma futura possível patologia.

A implementação do sistema pretende-se que seja de baixo custo, sem que esta motivação limite nem condicione as funcionalidades e a utilidade proposta, que permita o acesso aos dados em tempo real, ao histórico, que esta informação seja disponibilizada de uma forma tal que possa servir de apoio a quem tenha de tomar decisões em tempo útil.. As funcionalidades que visam a monitorização e rastreio do ritmo cardíaco são possíveis e fáceis de implementar contemplando apenas uma banda e um smartphone com uma aplicação desenvolvida para esse objetivo.

A deteção de valores anormais, enviando mensagens de alerta das anormalidades em tempo real, são também funcionalidades possíveis da aplicação tendo em conta que esta já fará a monitorizarão e rastreio, bastando apenas que receber os dados e proceder á sua análise use o sistema de envio de mensagens SMS comum ao conceito de smartphone.

O registo das leituras, criando gradativamente um histórico configurável, de fácil adaptação a variados caso e tipos de análise, de fácil acesso, ótima disponibilidade obriga a que estas leituras sejam armazenadas exteriormente ao smartphone, não sendo esta questão preocupante visto que é também parte do conceito de smartphone ter um ou mais formas de conexão á Internet e formas de uma aplicação entregar dados a uma outra plataforma para esta os armazenar, tratar e disponibilizar de seguida.

Para o desenvolvimento da aplicação para o sistema operativo Android, que é o predominante nos smartphones, em especial nos que tem uma boa relação preço/qualidade, existem muito boas ferramentas sem custos, sendo a opção dependente exclusivamente do programador. A aplicação deverá apenas abranger a maior variedade de marcas e modelos, por isso, exclui-se os últimos modelos e ultimas versões do Android sem qualquer problema, visto haver retro-compatibilidade.

Entre os smartphones acessíveis utilizaram-se os de uso pessoal do dia-a-dia, devido á natureza do projeto que necessita de recolha de dados constante por motivos de testes, verificações, análises, ensaios, etc..

Escolheu-se uma das bandas com mais baixo custo do mercado e está globalmente difundida disponível para venda, facilmente adquirível, a qual tem software desenvolvido pelo fabricante e já existe alguns passos na criação de software aberto e disponível para o publico geral.

Para armazenar os dados da aplicação a escolha recaiu numa plataforma orientada para as tecnologias da Cloud e da IoT, que fornece soluções prontas e gratuitas para estudantes, agnóstica no que se refere a protocolos de comunicação, ficando assim o caminho aberto sem limitações aparentes para o futuro desenvolvimento deste sistema.

O ambiente de desenvolvimento escolhido abrange todos os presentes dispositivos Android e é repleto de funcionalidades e facilidades para uso do programador.

Como primeiro passo procedeu-se á conexão entre a banda e o smartphone, através do Bluetooth, permitindo então a recolha de valores lidos pela banda, que neste caso em questão são o ritmo cardíaco e o nível de carga da bateria. Com esta recolha de valores para a aplicação foi possível enviá-los para a plataforma na Cloud, a qual permite avaliá-los, enviando uma mensagem SMS caso estes valores estejam for do intervalo pré-configurado, nomeadamente o valor do ritmo cardíaco. Redundantemente criou-se um método de replicar o processo de avaliação no smartphone, podendo a aplicação avaliar e enviar também uma mensagem SMS via GSM caso o valor esteja fora do intervalo definido pelo utilizador. Desta forma temos um sistema de alerta configurável.

A plataforma armazena e permite guardar, tratar e apresentar os valores por ela recebidos de uma forma facilmente configurável, destacando as possibilidades gráficas aprazíveis. Desta forma temos um histórico de valores lidos passiveis de uma análise fácil.

Programou-se na aplicação um temporizador, facilmente configurável pelo utilizador, para poder-se controlar o intervalo de tempo entre o acionamento do processo de leitura do valor do ritmo cardíaco. Desta forma têm-se leituras persistentes e a consequente recolha de valores, no intervalo de tempo que for conveniente.

Estas configurações locais do utilizador, dados de acesso à plataforma na Cloud e o numero de acesso são gravadas em ficheiro no smartphone para serem lidas e utilizadas logo a partir do inicio da aplicação.

3.2 Implementação

Neste capítulo é descrita a implementação de um protótipo do sistema proposto anteriormente, descrevendo a sua metodologia de desenvolvimento e implementação, a motivação das escolhas e origem das opções, limitações impostas ou encontradas, ferramentas utilizadas e o seu manuseamento.

3.2.1 Estudo e Escolha de Soluções

Smartphones

Utilizaram-se então vários smartphones comuns aptos para entrarem neste projeto, entre os quais, foram utilizados mais frequentemente o Alcatel OneTouch Idol 3 que tem o Android 6.0, e o Samsung Galaxy Note 3 que tem o Android 5.1.1.

Os smartphones conseguiram conectar-se com facilidade a uma banda Xiaomi Mi Band 2 Smartband, que mede o ritmos cardíaco entre outras funcionalidades, é das bandas com mais baixo custo do mercado e está globalmente disponível para venda frequentemente com promoção.Embora o seguinte facto não constitua um impedimento ou problema, ressalva-se que aparentemente esta facilidade surge apenas após a instalação da aplicação da marca e somente após a instalação do software Xiaomi MIBand2, disponibilizado gratuitamente através da Google Play Store.

Uma Plataforma Na Cloud Para Armazenar Dados

A escolha de uma plataforma na Cloud para armazenar os dados da aplicação recaiu na Ubidots que tem uma API[1] (Application Programming Interface) agnóstica conectando-se por isso com qualquer dispositivo por HTTP, MQTT, TCP, UDP[2], ou construir um protocolo industrial até mesmo personalizado, e oferece gratuitamente para estudantes as funcionalidades que pretendemos utilizar. A Ubidots é orientada para as tecnologias da Cloud e da IoT, fornecendo soluções prontas a usar neste âmbito, ficando assim o caminho aberto sem limitações aparentes para o crescimento futuro deste sistema.

Ambiente de Desenvolvimento Integrado (IDE)

Para desenvolver a aplicação para o sistema escolheu-se o Android Studio, que já vai na versão 3.0.1, devido às facilidades disponibilizadas (Instant Run, Editor de código inteligente, Modelos de código, integração com o GitHub, etc.), emulação rápida com recursos completos aliada a um Sistema de compilação robusto e ao mesmo tempo bastante flexível, abrangendo todos os dispositivos Android.

3.2.2 Desenvolvimento

Conexão entre Banda e Smartphone

Como primeiro passo, olhando para as peças do sistema, o mais óbvio foi fazer a ligação entre uma aplicação e a banda, tentando pedir e receber dados à banda. Para isso acontecer, houve uma fase de estudo da forma como a banda comunicava através do Bluetooth. Ficam aqui expressos os agradecimentos a referencia: slp e ashimokawa[3], yonixw[4], e finalmente a aashari[5] pelo trabalho disponibilizado no âmbito do UUID[6] dos Serviços e das Caraterísticas Bluetooth da MiBand2.

Guardar Dados na Cloud

Após a conexão via Bluetooth com a banda, com o emparelhamento facilitado pela instalação da app fornecida pela Xiaomi na Google Play Store, e o sucesso na entrega dos valores pedidos, seguiu-se o envio destes valores para a Ubidots, à qual se louva e agradece o apoio ao estudo e desenvolvimentos das tecnologias neste âmbito fornecendo os seus serviços de forma gratuita.

A criação e configuração da conta na plataforma Ubidots é bastante simples fácil, tendo acesso quase imediato ao necessário para começar. Configurámos a variável Batimento Cardíaco e a também a Bateria para receber os respetivos valores. O valor da carga da bateria também foi contemplado para que o tempo restante de funcionamento restante da banda fosse também observável. Dentro de outras possibilidades e formas, implementou-se a funcionalidade de envio de valores através da ApiClient fornecida, a APIUbidots.

[1] API provém do Inglês Application Programming Interface, é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

[2] HTTP, MQTT, TCP,e UDP são protocolos de comunicação de rede

[3] `https://github.com/Freeyourgadget/Gadgetbridge`

[4] `https://github.com/yonixw/mi-band-2`

[5] `https://github.com/aashari/mi-band-2`

[6] O objetivo dos UUID é possibilitar aos sistemas distribuídos a identificarem unicamente informações sem coordenação central significante.

Mesmo assim, foi criado em anexo um guia simples com imagens a descrever o processo de criação e configuração da conta.

Implementação de Envio de Alertas Por SMS

Observando o histórico de valores fornecido na plataforma, de fácil acesso e configuração, testou-se a possibilidade de existir alarmes e envio de mensagens SMS para alertar para os valores fora dos limites configurados. Esta possibilidade existe, a sua configuração é a mais simples que se possa sugerir, funciona, mas não é gratuita.

Como opção alternativa, para servir de complemento ou criar redundância na funcionalidade de envio de alertas por mensagem SMS, programou-se esta funcionalidade na aplicação também, acrescentando assim uma limitação: a necessidade de ter um plano com uma operadora telefónica, ou saldo no cartão GSM, e/ou ter saldo na conta Ubidots, para que através destes, ou pelo menos através de um destes serviços sejam enviadas as mensagens SMS, caso existam situações de alarme.

A solução criada localmente (Alarm Setting), programada na aplicação do smartphone, também tem por base uma configuração simples e intuitiva como se pode verificar na fig. 3.1.

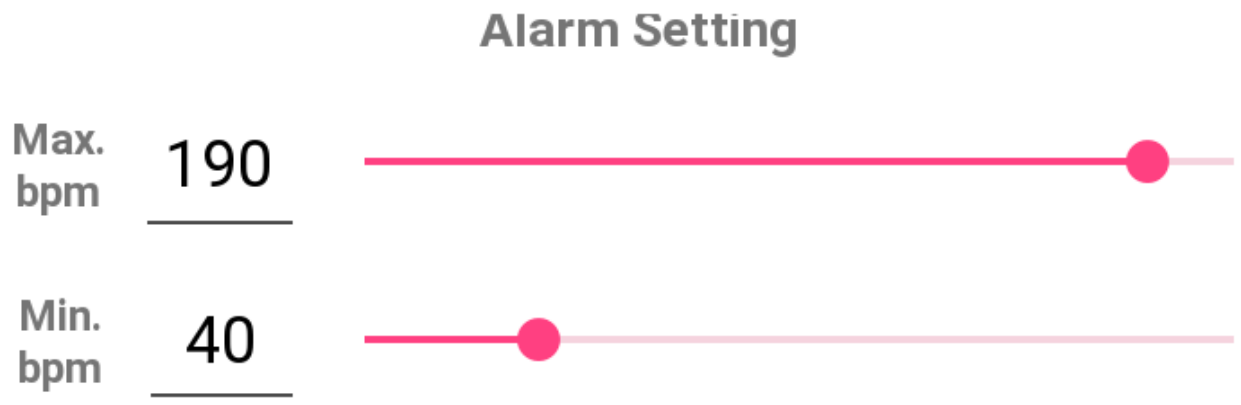

Figura 3.1: Configuração do intervalo de valores considerados normais no ritmo cardíaco

Existe uma barra com um seletor para mover até ao valor que é pretendido estabelecer como valor máximo (Max.bpm) e outra igualmente para o valor mínimo (Min. bpm). À medida que se move o seletor ao longo da barra o valor correspondente a posição atual do seletor é indicado ao lado da barra.

Qual o intervalo que o utilizador deve de escolher?

Esta é a questão natural que surge de imediato mas a questão verdadeira é "Qual o intervalo ideal para o utilizador?", pois estes máximos e mínimos variam de pessoa para pessoa e ao longo da idade da mesma.

O bom senso,a intuição e a observação dos valores ao longo do tempo da utilização do sistema levam a descobrir o intervalo ideal do utilizador, cruzando o seu estado de saúde, anímico e a idade, com a faixa de valores observados. Cada coração tem sua cadência, mas alterações bruscas podem ser sinais de problemas graves, os quais podem ter consequências dramáticas. Estes valores, o intervalo considerado e toda a informação possível deve ser apresentada a um especialista ou no mínimo ao médico responsável por acompanhar o estado de saúde do utilizador. O intervalo depende sempre do estado de saúde do utilizador e a gravidade de os batimentos cardíacos atingirem certos valores.

O sistema pode ser mais útil quanto mais precisa for a sua configuração, tendo sempre em conta que é preferível um falso alarme do que nenhum.

Para a maioria das pessoas o intervalo normal ronda em media entre os 60 bpm e os 100 bpm. O ritmo pode será afetado por fatores como o stress, ansiedade hormonas medicação e pelo nível de atividade física.

Ritmo Cardíaco Mínimo Em primeiro lugar pode-se começar pelo ritmo cardíaco em repouso para ter-se a ideia do valor a estipular como valor mínimo de bpm na aplicação. O ritmo cardíaco em repouso corresponde ao numero de batimentos quando o utilizador está em repouso. Uma boa altura para ver este valor é de manhã, depois de uma boa noite de sono, assim que acorda. Um atleta ou uma pessoa bastante ativa pode ter um batimento cardíaco em repouso tão baixo como 40 bpm. O coração é muito eficiente nestes casos e cada bombeada do coração entrega ao corpo muito mais sangue que o normal, consequentemente mais oxigénio, por isso precisa bater menos vezes. Em repouso, numa pessoa saudável, quanto mais baixo for o batimento cardíaco melhor. Porque significa que o músculos do coração estão em boas condições e não precisa de trabalhar muito para fazer o seu trabalho. Se o utilizador for uma pessoa saudável e bastante ativa, e estiver deitado e quieto, ou até dormindo, é provável que não haja nenhum problema se seu coração estiver com apenas 30 bpm.

Porque o ritmo cardíaco muito baixo faz com que menos oxigénio circule pelo corpo, mesmo numa pessoa assim, estando desperta, em pé, esse ritmo cardíaco pode ser o anuncio, a previsão de desequilíbrios, desmaios, quedas, lesões e que em casos extremos pode terminar ou causar na morte. O estudo de 16 anos do batimento cardíaco no individuo em repouso encontraram uma relação entre o batimento cardíaco elevado em repouso e a forma física fraca, pressão arterial alta e excesso de peso. Apesar de todos estes indicadores elevados serem fatores de risco, o estudo conclui que o batimento cardíaco elevado no estado de repouso é um fator independente da causa de morte, por si só.[JSHG13]

Frequência Cardíaca Máxima: o Ritmo Cardíaco Máximo Calcular a Frequência Cardíaca Máxima (FCM) normal, frequente num individuo numa certa faixa etária é fácil: de 220 subtrai-se a idade do utilizador. Por exemplo: uma pessoa com 30 anos deve subtrair 220 – 30, achando assim o FCM de 190. Posto isto para um individuo com 30 anos de idade a FCM é de 190 bpm (batimentos por minuto). [www15]

O coração tem dois movimentos: a diástole (quando o órgão se enche de sangue) e a sístole (quando o sangue é bombeado para o corpo). Quando o coração acelera, ele encurta a diástole. Assim, o órgão envia menos sangue para o corpo, causando cansaço e desmaios. Uma frequência cardíaca perto dos 180 bpm é sinal de alerta total e perigo de morte.

Para finalizar, selecione o botão laranja fig. 3.2 para voltar ao painel principal e gravar as configurações.

Figura 3.2: Botão para voltar ao painel principal e gravar as configurações

Amostragem dos valores do Ritmo Cardíaco

Para o sistema estar completo é preciso ter um processo de recolha dos valores do ritmo cardíaco. Para controlar este processo temos um temporizador facilmente e intuitivamente programável como se pode verificar na fig. 3.3. Este vai acionar o processo de ler o ritmo cardíaco no intervalo de tempo estipulado.

À semelhança do método de selecionar o batimento cardíaco máximo e mínimo, o intervalo de tempo entre as medições do batimento cardíaco também é selecionado através de um seletor que desliza ao longo de uma barra. Cada medição demora até 20 segundos. Cada posição do seletor na barra representada um valor

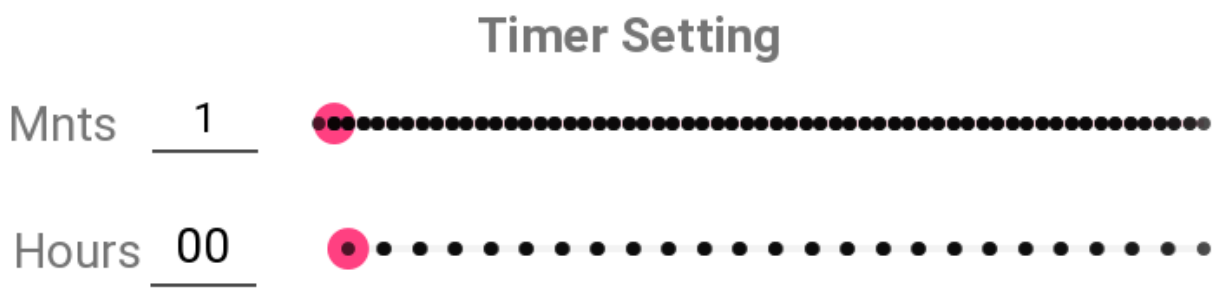

Figura 3.3: Configuração do intervalo de tempo entre o acionamento das leituras de valores

que é mostrado ao lado direito da barra. Configura-se a quantidade de horas num seletor e a quantidade de minutos noutro seletor, sendo o tempo de duração do intervalo entre medições correspondente á soma das horas com os minutos. Sendo assim possível selecionar um intervalo de tempo entre 0 minutos(não faz medições automáticas) e 25 horas (0 máximo de 24h no seletor das horas e mais o máximo de 60 minutos no seletor dos minutos).

Também, à semelhança do método de selecionar o intervalo do batimento cardíaco, o intervalo de tempo entre as medições do batimento cardíaco também varia, podendo variar até ao longo do dia com a necessidade de observação e registo dos valores , a atividade, o estado de saúde etc. Recomenda-se bom senso e o conselho um especialista ou no mínimo ao médico responsável por acompanhar o estado de saúde do utilizador para encontrar estes valores.

Para finalizar, selecione o botão laranja fig. 3.2 para voltar ao painel principal e gravar as configurações.

Configurações e Dados de Acesso à Plataforma na Cloud

A criação e configuração da conta na plataforma, está descrita passo a passo no capitulo 6 na pagina 75. É um processo normal, apenas requerendo que a conta, depois de criada, tenha um dispositivo virtual, com duas variáveis , de preferência com o nome Ritmo Cardíaco e Bateria respetivamente, para que qualquer pessoa que tenha de consultar o histórico tenha uma perceção clara do tipo de informação que está disponibilizada. A criação do histórico também está descrita no anexo.

Estas duas variáveis são aquelas que vão armazenar os dados que a aplicação envia, após os receber da banda. Caso a aplicação não esteja devidamente configurada com estes dados, continuará a funcionar, mas não acrescentará dados no histórico, inviabilizando assim uma parte importante do sistema.

É de salientar, que a plataforma também tem forma de enviar alertas, mas permite múltiplos recetores destas mensagens, podendo personalizar cada uma delas para cada recetor. Inclusivamente, permite que estas mensagens sejam enviadas também por email, ou por SMS, ou por ambos.

Tem um construtor de condições para acionar o envio destas mensagens, podendo criar inúmeras condições. Por exemplo, se a média do ritmo cardíaco ao longo do dia está mais alta do que o medico considerou normal, pode enviar um email a pedir que estes valores sejam analisados. Ou ainda, numa pessoa com a tensão alta, se o valor ultrapassou os limites de alerta e atingiu um valor preocupante, pode ser acordado com o medico enviar uma mensagem a pedir a sua atenção urgente com o numero de telefone da pessoa responsável enquanto esta leva o utilizador aos serviços de urgência. Num utilizar com um estado mais preocupante, para além desta mensagens, tendo sido lido um valor alarmante, tendo em conta este valor ser acionado uma mensagem aos serviços de urgência com o pedido de auxilio, descrevendo estado de saúde

e o valor que levou a enviar a mensagem, conjuntamente com a identificação e a morada.

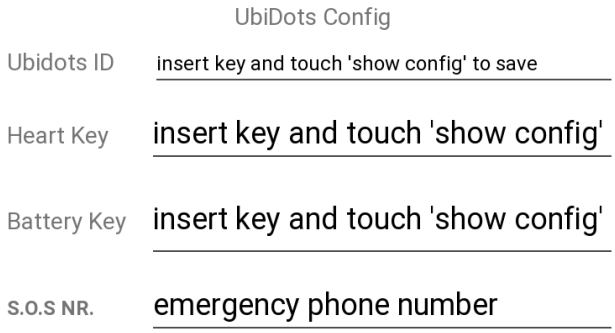

Figura 3.4: Configuração dos Dados de Acesso à Plataforma na Cloud

No mínimo, pretende-se um histórico de valores. Para se conseguir enviar dados para a é preciso introduzir nesta parte (ver fig. 3.4) a chave da conta de acesso à plataforma, a UbidotsID e introduzir também as identificações de cada uma das variáveis: a do batimento cardíaco que está identificada como "Heart Key", e a do nível da bateria identificada como "battery key". Neste parte também se pode introduzir um numero de telefone para o qual serão enviadas as mensagens SMS de alerta no espaço "S.O.S. NR.".

3.5.

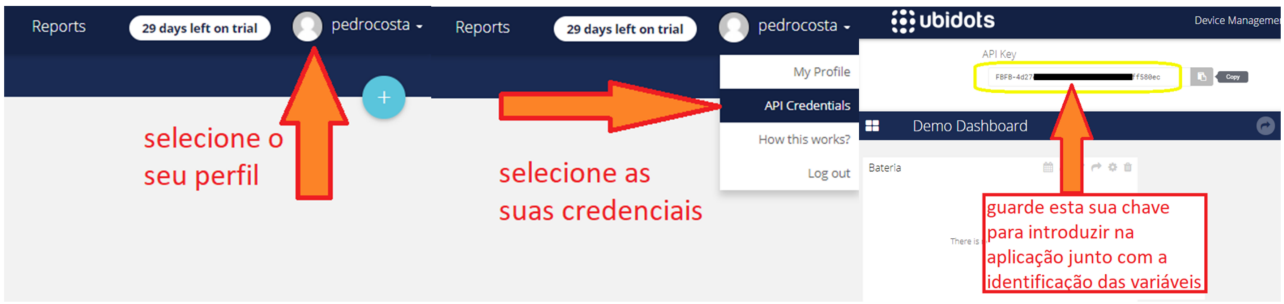

Figura 3.5: Acesso à chave da Plataforma na Cloud

Esta chave da conta, necessária para configurar o acesso da aplicação, pode-se obter acedendo à conta, procurando no menu perfil a opção "API Credentials"como é demonstrado na imagem 3.5.

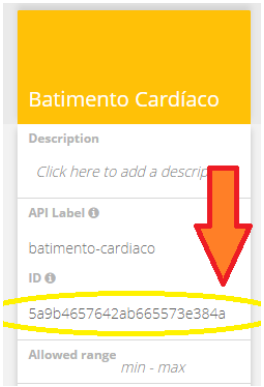

Figura 3.6: Acesso à identificação de uma variável

A identificação de uma variável obtém-se acedendo aos dispositivos (Devices), selecionando o dispositivo que tem a variáveis que estão a utilizar, finalmente selecionando uma variável tem a identificação da mesma tal como é mostrado no exemplo da fig.3.6.

Para finalizar, selecione o botão laranja fig. 3.2 para voltar ao painel principal e gravar as configurações.

Funções Manuais e Descrição dos Painéis Principal e de Configuração

Na aplicação fig.3.7 podem-se ver os seguintes itens:

- 1-Botão para alternar entre o painel das configurações e o painel atual
- 2-Indicador do estado da conexão Bluetooth com a banda(conectado ou a tentar conectar
- 3-Botão para iniciar uma leitura do ritmo cardíaco manualmente, com a indicação do ultimo valor lido
- 4-Botão para por a banda a vibrar/parar de vibrar. Esta funcionalidade pode servir para a por a vibrar para ser mais fácil a encontrar, ou simplesmente para testar a ligação.
- 5-Botão para atualizar o nível da bateria da banda, com a indicação do ultimo valor lido.

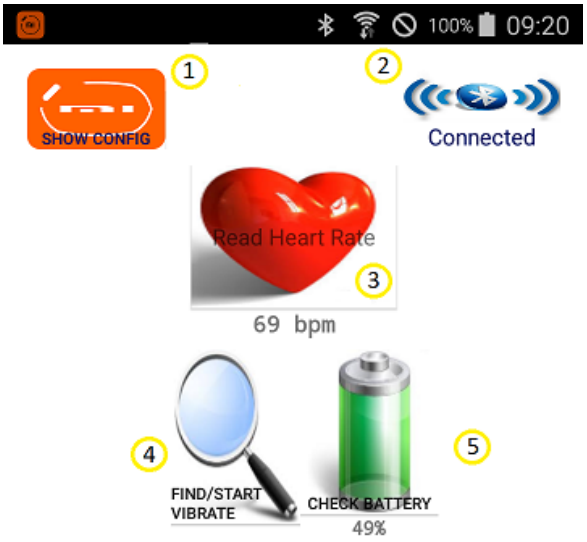

Figura 3.7: Ecrã Principal

3.3 Resultados

Cumpriu-se objetivo de construir o sistema a partir de equipamentos usuais na nossa sociedade, de baixo preço, e com a eficiência necessária para cumprir os objetivos. O smartphone é um objeto normal no dia a dia, são cada mais práticos e usados dentro da população mais idosa. A banda, devido ao seu design, tem a função de mostrar hora, o seu uso é idênticos ao de um relógio, e assemelha-se a um smartwatch moderno, por isso apetecível e prático. O resto do sistema, que é transparente ao utilizador, é uma mais valia devido á sua função e facilidade de utilização. O impacto de passar a utilizar o sistema no dia a dia é pequeno visto que são objetos comuns nos dias de hoje.

É possível configurar os sistema com um guia passo a passo, desde a configuração inicial da plataforma da Cloud à configuração pontual ou frequente da aplicação. Pode haver necessidade de ajustar a configuração dos intervalos do ritmo cardíaco normal e do intervalo do tempo de amostragem, ajustando assim os valores aos ótimos para o utilizador. Por este motivo houve um esforço para que estas configurações serem da maior facilidade e simplicidade conseguida.

Apenas com o objetivo de testar o funcionamento do sistema, ver a sua funcionalidade e resultados, não o detetar e avaliar as diferentes gamas de batimentos cardíacos saudáveis de cada individuo, utilizou-se o sistema ao longo de vários meses, numa primeira fase com 3 indivíduos diferentes, com o Alcatel OneTouch Idol 3 que tem o Android 6.0, procedendo-se a amostragens diversas: 24 horas com um intervalo de amostragens de 1 minuto e 5 minutos; 7 dias com um intervalos de amostragem de 10 minutos. Com o Samsung Galaxy Note 3 que tem o Android 5.1.1. e um quarto utilizador diferente, procedeu-se amostragens de 2 minutos durante 48 horas e posteriormente de 5 minutos durante alguns meses mas nem sempre continuamente procurando situações criticas do funcionamento do sistema, por exemplo, a posição da banda no pulso muda durante o dia-a-dia, se existem ambientes que influenciam as leituras ou a transmissão dos dados ao smartphone, avaliar se é significativo e digno de registo: o consumo extra de bateria tanto da banda como do smartphone; o consumo de dados; etc.

Verificou-se a necessidade de ajuste dos do intervalo de alarme conforme a atividade para uma monitorização mais apertada.

Embora não fosse esse o objetivo dos testes dados recolhidos permitem conhecer de uma forma mais precisa e pessoal da variação do batimento cardíaco do individuo e até detetar o seu estado emocional: nervoso/excitado ou calmo/repousado/relaxado.

Durante o funcionamento, por vezes algumas leituras são dispares por motivo de mau ajuste da pulseira MiBand2. Outras pontualmente são notoriamente anómalas, talvez por erro ou perda dos dados na transmissão entre a banda e o smartphone.

Por este motivo exclui-se o valor zero como valor valido para o nível da bateria e repete-se a leitura do ritmo cardíaco sempre que se leem valores fora do intervalo configurado, registando os mesmos, de qualquer forma na Cloud. Isto para que se possa fazer uma análise paralela ao sistema e as suas possíveis falhas. Principalmente porque as situações de emergência se podem confundir com anomalias. Estas ultimas leituras fora do intervalo serão tratadas como situações que requerem atenção pois, ou os alertas são validos, ou a sua causa estará a impedir o funcionamento correto do sistema, impedindo que este cumpra a sua função corretamente. De uma forma ou de outra é necessária uma avaliação.

Pretende-se que as anomalias de leituras sejam sempre motivo de alerta, mas somente as consecutivas, para não saturar o sistema, que inclui um ser humano a receber mensagens SMS, o qual ao fim de um determinado período de tempo deixaria de valorizar devidamente os alertas, como por exemplo, ao fim de algumas situações de a banda estar mal posicionada ou até mesmo de a banda estar fora do pulso do utilizador, por exemplo, por este estar a tomar banho.

Após ser lido um valor fora do intervalo configurado, é de imediato acionado o pedido de nova leitura, sucessivamente enquanto os valores estiverem fora do intervalo. É enviado o alerta somente se estes valores fora do intervalo forem mais do que 3 consecutivos, tanto por excesso como por defeito ou alternadamente. Após 3 leituras consecutivas com valores fora do intervalo, será enviado um alerta por cada uma destas. Uma leitura pontual fora do intervalo é uma situação a analisar, pois as outras estarão a sinalizar que o utilizar por enquanto está/continua bem, existe a possibilidade de ser uma leitura de um valor errado. no caso de serem 3 ou mais valores imediatamente consecutivos, é porque é necessária intervenção para com o utilizador ou no sistema.

A motivação desta opção/decisão de registar até mesmo estes valores anómalos é que alguém responsável por receber os alertas, a pessoa que está na posse do telefone para o qual estão a ser enviadas as mensagens SMS, tenha conhecimento destas anomalias e intervenha ou peça uma intervenção, num tempo considerado útil, independentemente de ser uma sucessão de anomalias ou de uma situação de emergência. Ou seja, a responsabilidade, a avaliação e o tipo da decisão tomada nestas situações será feita por alguém ciente do estado de saúde do utilizador deste sistema, será alguém capaz de acionar um serviço de emergência médica numa situação limite, num caso duvida ou num caso incerteza do estado de saúde do utilizador.

Paralelamente ou redundantemente, existe:

- o histórico dos valores do ritmo cardíaco, na plataforma na Cloud, para se poder consultar rapidamente,
- o próprio smartphone que pode receber chamadas e assim entrar-se em contacto com o utilizador para se ter mais informação: se atende e fala, se fala corretamente, se aceita uma vídeo-chamada para ver a sua expressão e estado anímico etc..

Resumindo, num caso especifico que assim o exija, que se proceda a um protocolo ações estipulado ou definido por alguém com essa competência.

Lembrando que mais vale um falso alarme do que nenhum, é sempre bom garantir que dentro de casa a cobertura da rede WI-FI complementa a ausência de sinal de rede móvel. Esta situação é frequente em casa com construção antiga. Na excecional ausência dos dois sinal sentiu-se a necessidade de a aplicação guardar os valores recolhidos para enviá-los na primeira oportunidade.

O resultado foi o esperado. O sistema tem o comportamento que se previa cumprindo o que era pretendido, efetivamente:

- aciona leituras periodicamente programadas,
- monitoriza e regista as leituras,
- analisa os valores pelo intervalo selecionado
- e manda mensagens a alertar caso estejam fora do mesmo.

3.4 Sistemas Semelhantes

Existem sistemas que se assemelham de alguma forma ao sistema aqui proposto, todos com um propósito valido, útil e honesto. Todos promovem as inúmeras e grandiosas vantagens da IoT e tecnologias para melhorar os cuidados de saúde, monitorizando, prevenindo, auxiliando o serviços de saúde, consequentemente melhorando a qualidade de vida e principalmente salvando vidas.

Após a análise concluímos que as diferenças são óbvias, a nível das funcionalidades aqui pretendidas: simplicidade, custo e escalabilidade.

Muitas das soluções optam por logo de inicio envolver placas Arduino e RaspberryPi, que é um passo mais á frente do projeto aqui apresentado, que está preparado para receber essas soluções, inclusivamente de forma redundante (pela aplicação e pela plataforma da Cloud). A maioria também se auxilia de uma ou mais plataformas para armazenar e disponibilizar os dados recolhidos.

De seguida vamos dar alguns exemplos:

A Modern Health Care System Using IoT and Android.[GBJA16]

O sistema propõe uma plataforma IoT moderna composta por uma caixa inteligente para cuidados médicos com sensores de monitorização de saúde e diagnostico para o domicilio com conexão Wireless a par de uma aplicação androide para estreitar a comunicação entre os pacientes e os médicos, onde o paciente poderá pedir uma consulta, que será marcada após a aceitação do medico e autorizada pelos serviços do hospital.

A caixa da plataforma inteligente tem como base um Arduino com um sensor de temperatura, uma campainha, um LED e WI-FI. A caixa da plataforma proposta alerta os pacientes para tomarem os medicamento certo nas horas especificada. Devido à ligação à Internet atualiza de tempos a tempos as especificações dos alertas no smartphone do paciente acerca dos medicamentos e as horas a que estes deverão de ser tomados.

Se forem detetados sinais vitais, serão enviados alertas via SMS ao guardião predefinido. O medico atribuído ao paciente poderá consultar os dados do paciente, a temperatura, gerir as prescrições e consultas.

Home Based Health Monitoting System Using Android Smartphone[PK14]

Mais com o intuito de remotamente monitorizar e vigiar o estado de saúde de um paciente que requeira uma atenção apertada, este sistema propõe recolher dados de bio-sensores, predizer o estado de saúde inteligentemente e retornar essa informação ao paciente e aos médicos através de uma aplicação no seus telemóveis.

O sistema de monitorização envolve um controlador ARM7 LPC 2148 que apresenta no ambiente gráfico do Matlab [7] os dados recolhidos do sensor de temperatura, medidor de ritmo cardíaco e ECG. No seguimento seria criado uma base de dados para guardar os dados caso o paciente fosse suscetível de voltar a ser consultado.

Smart Mobile Healthcare System based on WBSN and 5G[NM17]

Propõe-se aqui um sistema IoT de cuidados de saúde, de arquitetura generalizada para monitorizar riscos de saúde do paciente usando um smartphone e 5G[8].

O sistema aconselha e alerta médicos e assistentes em tempo real sobre alterações dos parâmetros vitais dos pacientes: temperatura corporal, pulsação, oxigénio, etc., e também sobre mudanças importantes nos parâmetros do ambiente de acordo com os dados lidos pela rede de sensores corporais sem-fios (WBSN[9]).

Os dados fisiológicos são processados usando 5G, sensores corporais ligados a uma placa Arduino e outra RaspberryPi. Envolve aplicações Web e moveis desenvolvidas numa filosofia multi-protocolar para apoiar as necessidades de informação dos pacientes, médicos, análises de laboratórios e serviços hospitalares. A informação é disponibilizada nas aplicações em tempo real para poder ser monitorizada, analisada e utilizada nos diagnósticos médicos.

IoT Based Health Monitoring System Using Android App[KMA+17]

O sistema proposto é constituído por uma aplicação Android, sensor de temperatura, humidade e ritmo cardíaco.

Os valores de temperatura, humidade e ritmos cardíaco recebidos pelo RaspberryPi são enviados para a

[7] software interativo de alta performance voltado para o cálculo numérico

[8] O 5G (Quinta Geração de Internet móvel ou Quinta Geração de sistema sem fio), representa a futura geração de telecomunicação móvel. O 5G já vem sendo estudado para substituir o 4G e ter a próxima geração lançada dentro os próximos 10 anos, seguindo o mesmo padrão de evolução das demais gerações anteriores.

[9] WBSN-Wireless Body Sensor Network é uma rede usada para a comunicação entre sensores operando no ou dentro do corpo humano para a monitorização dos parâmetros vitais e movimentos.

Cloud, requisitados pela aplicação Android, e então poderão ser visualizados no smartphone. Na continuação deste projeto é proposto a utilização do sistema de mensagens SMS para obter o estado do paciente e a sua localização via GPS.

Patient Health Monitoring System using IOT[PWKG17]

Neste sistema o paciente transportará uma caixa com uma placa Arduino ligada aos sensores de temperatura do corpo e do batimento cardíaco.

O smartphone fornece a localização GPS se necessária . Os dados dos sensores captados pela placa Arduino serão enviados para uma aplicação Android no smartphone via Bluetooth ou WI-FI. Estes dados serão guardados numa base de dados na Cloud que será consultada pelos médicos para os auxiliar . A aplicação Android detetará anormalidades no ritmo cardíaco e ataques cardíacos enviando neste caso uma mensagem SMS contendo a localização (via GPS) ao medico do paciente, parentes e hospitais requerendo cuidados médicos urgentes.

An IoT Application: Health Care System with Android Devices[CL16]

Propõe-se aqui o sistema mais semelhante na sua estrutura, com o propósito mais idêntico, mais próximo no custo e na simplicidade.

O sistema mede a saturação periférica de oxigénio e ritmo cardíaco utilizando oximetro de pulso inteligente com tecnologia Bluetooth. Um tablete com uma aplicação Android faz a ponte entre os dados recolhidos pelos dispositivos de medição e o seu armazenamento no servidor na Cloud. Como suporte na Cloud é usado uma instância de um Amazon Cloud Server EC2 com um LAMP[10]stack web server, onde os dados enviados de um ficheiro PHP são armazenados numa tabela de uma base de dados, para que estes dados possam ser consultados, analisados e tratados. Como trabalho futuro propõem um site para pesquisa dos dados recolhidos e gráficos para analises e estatísticas.

[10]O LAMP é um modelo de pilhas de serviços da Web, nomeado como um acrónimo dos nomes de seus quatro componentes originais de código aberto: o sistema operativo Linux, o Apache HTTP Server, o sistema de gestão de base de dados relacional (RDBMS) e a linguagem de programação PHP.

4

Conclusões e Trabalho Futuro

Arritmia cardíaca é um grupo de condições em que o batimento cardíaco é irregular, demasiado rápido ou demasiado lento. O ritmo cardíaco anormal representam uma das fontes principais de mortalidade entre pacientes com doenças cardiovasculares. Num estudo realizado em 147 idosos ativos e saudáveis, observaram arritmias em 93,2% [Fil00]. Noutro estudo em 98 indivíduos de ambos os sexos, com idades compreendidas entre os 60 a 85 anos de idade, verificou-se arritmias em pelo menos 80% [FK82].

Por este motivo sente-se uma necessidade aparelhos que monitorizem os batimentos cardíacos por um largo período de tempo e que detetem automaticamente certas arritmias[Pol13]. Atendendo a esta necessidade, tendo em conta que a maioria dos idosos estão sozinhos muito tempo em suas casas ou carecem de atenção nos lares ou internamentos, criou-se um sistema que pelo menos monitoriza e registe o ritmo cardíaco, e que principalmente alerta a deteção de um ritmo cardíaco anormal.

O sistema é construido a partir de equipamentos usuais na nossa sociedade, de baixo preço, e com a eficiência necessária para cumprir os objetivos: O smartphone com sistema Android, a banda com um design semelhante a um smartwatch, interligados por uma ligação Bluetooth. Só com esta parte do sistema já é possível a recolha de valores do ritmo cardíaco lidos pela banda, com uma frequência programada, enviando uma mensagem SMS caso estes valores estejam fora do intervalo pré-configurado, alertando o acontecimento.

Aproveitando a existência da ligação á Internet, o valores lidos são enviados para a plataforma que armazena, tratar e apresenta os valores por ela recebidos, destacando as possibilidades gráficas aprazíveis com as quais apresenta um histórico de valores lidos passiveis de uma análise fácil pela pessoa responsável pela saúde do utilizador do sistema.

É de salientar, que a plataforma também tem forma de enviar alertas, mas permite múltiplos recetores destas mensagens, podendo personalizar cada uma delas para cada recetor. Inclusivamente, permite que estas mensagens sejam enviadas também por email, ou por SMS, ou por ambos. E assim temos um sistema de alerta redundante.

Devido à vulgaridade dos equipamentos usados para construir o sistema, a aceitação de usa-los como parte do sistema é ótima. Seria bom aproveitar esta aceitação para contornar algumas dificuldades ainda existentes na população idosa na utilização dos smartphones nas suas funcionalidades mais simples, aproveitando a aplicação para resumir certas funções ao toque de um botão no ecrã, com um design intuitivo.

Evidentemente, o sistema trará algum conforto, e principalmente aumentará a independência tanto dos utilizadores como de quem tem pessoas a seu cargo neste quadro, prevenindo emergências, podendo salvar vidas. Relembrando o porquê desta conclusão:

- Este sistema trás segurança ao utilizador. Este saberá que mesmo estando sozinho, o sistema alertará para uma situação anormal.
- A pessoa que tem alguém á sua responsabilidade, a qual precisa de alguma monitorização ou vigilância, estará mais descansada, pois será alertada para alguma situação anormal. Isto é valido mesmo nas situações em que ambos estão próximos mas a atenção da pessoa responsável pode não ser temporariamente a necessária e o utilizador não conseguir, ou não ter tempo para pedir ajuda nesse momento.
- Alguém que tem várias pessoas a seu cargo, por exemplo, lares, centros de dia, ambulatórios, etc., pode monitorizar simultaneamente pessoas com quadros onde e este sistema é útil, como se pode observar a titulo de exemplo na fig.4.1.
- Especialmente nos períodos em que existem menos pessoas responsáveis nestes sítios, por exemplo, durante os períodos noturnos, podendo o sistema inclusivamente alertar os responsáveis que estão de prevenção.

É natural ter ideias de funcionalidades para a evolução é melhoramento do sistema. Como contribuição para a comunidade e de forma a que haja proveito e continuidade no contributo, o projeto completo pronto para ser compilado no Android Studio 3.0 está disponível em:

`https://github.com/tasdsf/HRM4MiBand2`

Este está num dos seus estados mais resumidos logo as suas possibilidades de melhorar e expandir são inúmeras. Vamos enumerar algumas a titulo de exemplo:

- Ter a opção de o utilizar selecionar vários estados: numa atividade ou em repouso, para alem no normal. Ou se possível ser a aplicação a detetar os estados e assim se poder fazer uma avaliação mais precisa.
- Em caso de ausência de qualquer sinal de rede, a aplicação deveria de guardar os dados para envio numa primeira oportunidade. Em conjunto com esta melhoria também ficou a hipótese de nestes casos haver uma forma de alerta, talvez redundante com a programação de um alerta na plataforma, acionado por não estar a receber dados.

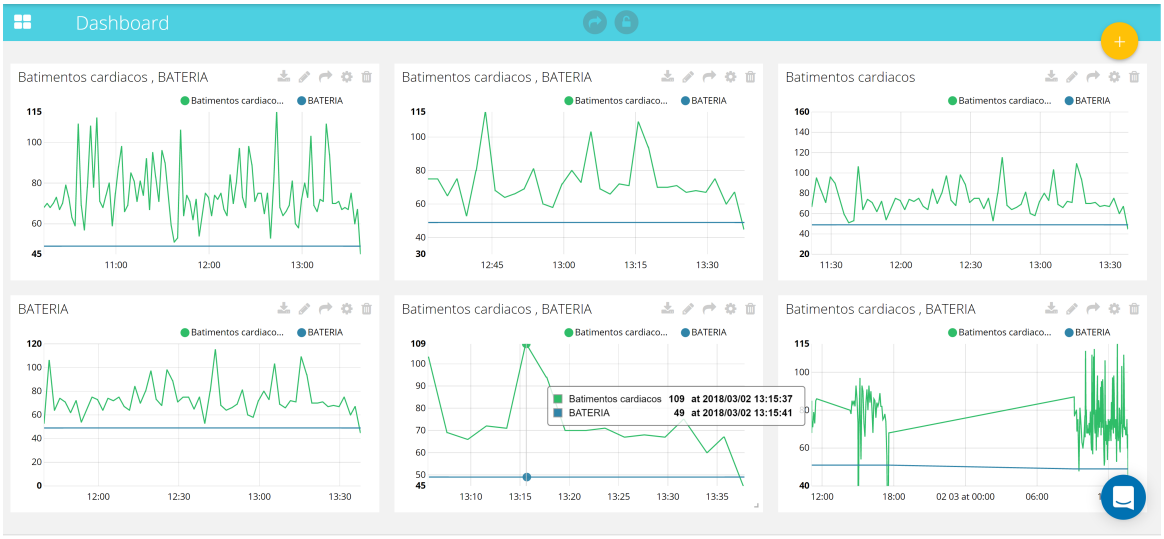

Figura 4.1: Ecrã Principal

- Fica também para fazer na próxima oportunidade, a criação da possibilidade de mandar mensagens, segundo modelos personalizáveis, por mais meio, distinguindo situações e os recetores.
- o dispositivo smartphone e as suas múltiplas possibilidades de estabelecer comunicação com o utilizador do sistema: por chamada de voz, vídeo-chamada, outras plataformas comuns em smartphones, etc..
- A possibilidade de o utilizador iniciar uma chamada de voz ou vídeo para pessoa ou pessoas responsáveis a partir da aplicação, sem ter de utilizar diretamente o sistema do smartphone, contornando assim a dificuldade que algumas pessoas mais idosas têm na utilização do sistema do smartphone.
- A possibilidade da pessoa responsável poder iniciar uma chamada de voz ou vídeo sem a necessidade do utilizador ter de aceitar, mediante condições acordadas e previamente definidas, como por exemplo, após um alerta. Assim como ativar a localização do smartphone do utilizador, para a pessoa responsável poder ir mais rapidamente ao seu encontro.
- Complementar o sistema com o próprio sistema de vídeo-vigilância da casa, se este já existir ou instalar-se um se se justificar.
- Acrescentar dados de outros dispositivos do género dos que referimos em 1.2 na pagina 4 que complementem ou acrescentem o existente, na possibilidade de criar um histórico na mesma plataforma, para que o cruzamento de dados permita também avaliar, prever e prevenir, só para citar algumas possibilidades.

Muito mais se poderá facilmente acrescentar, como por exemplo: um sistema inteligente que analise o histórico e detete ou preveja também situações anómalas no estado de saúde do utilizador, como referimos anteriormente em 1.2 na pagina 4.

São algumas sugestões mínimas para salientar as utilidades do sistema, para além das que já nele estão disponíveis. As suas possibilidades prontas a ser utilizadas são múltiplas, conjugando e explorando:

- o sistema de monitorização,
- o histórico e a sua acessibilidade

- os alertas por mensagem SMS como primeira forma de alarme
- o dispositivo smartphone e as suas múltiplas possibilidades de estabelecer comunicação com o utilizador do sistema: por chamada de voz, vídeo-chamada, outras plataformas comuns em smartphones, etc..

O caminho que se pretende realizar é o que leva a um sistema simples, mas onde se possa proceder à integração valida, fiável, solida e persistente de toda e qualquer aplicação existente, na vasta variedade de doenças e problemas relacionados com as mesmas, contribuindo nos métodos preventivos e nas intervenções de emergência.

5

Código Fonte

Descreve-se aqui o código da aplicação e os restantes ficheiros necessarios para o seu funcionamento no ambiente Android O projeto completo pronto para ser compilado no Android Studio 3.0 está disponivel em:

`https://github.com/tasdsf/HRM4MiBand2`

5.1 MainActivity.java

```
1  package com.id.hrm4miband2.Activities;
2
3  import android.Manifest;
4  import android.annotation.SuppressLint;
5  import android.app.Activity;
6  import android.bluetooth.BluetoothAdapter;
7  import android.bluetooth.BluetoothDevice;
8  import android.bluetooth.BluetoothGatt;
9  import android.bluetooth.BluetoothGattCallback;
10 import android.bluetooth.BluetoothGattCharacteristic;
11 import android.bluetooth.BluetoothGattDescriptor;
```

```
12 import android.bluetooth.BluetoothGattService;
13 import android.bluetooth.BluetoothProfile;
14 import android.content.Context;
15 import android.content.Intent;
16 import android.content.pm.PackageManager;
17 import android.graphics.Color;
18 import android.media.MediaScannerConnection;
19 import android.net.Uri;
20 import android.os.AsyncTask;
21 import android.os.Bundle;
22 import android.os.CountDownTimer;
23 import android.os.Environment;
24 import android.support.v4.app.ActivityCompat;
25 import android.support.v4.content.ContextCompat;
26 import android.telephony.SmsManager;
27 import android.util.Log;
28 import android.view.View;
29 import android.widget.Button;
30 import android.widget.EditText;
31 import android.widget.SeekBar;
32 import android.widget.TextView;
33 import android.widget.Toast;
34
35 import com.ubidots.ApiClient;
36 import com.ubidots.Variable;
37
38 import java.io.BufferedReader;
39 import java.io.File;
40 import java.io.FileInputStream;
41 import java.io.FileNotFoundException;
42 import java.io.FileOutputStream;
43 import java.io.IOException;
44 import java.io.InputStreamReader;
45 import java.io.UnsupportedEncodingException;
46 import java.util.Set;
47 import java.util.Timer;
48 import java.util.TimerTask;
49 import java.util.concurrent.ExecutionException;
50
51 import com.id.hrm4miband2.Helpers.CustomBluetoothProfile;
52 import com.id.hrm4miband2.R;
53
54 public class MainActivity extends Activity {
55
56     // Default normal Heart Rate Values
57     private static final int ABSOLUTE_MAX_BPM = 200;
58     private static final int ABSOLUTE_MIN_BPM = 30;
59
60     ////////////////PERMISSIONS/////////////
61     //Flag to signal if the Bluetooth Requests are possible
62     private final static int REQUEST_ENABLE_BT = 1;
63     //Flag to signal if the requests to send sms are possible
64     private static final int MY_PERMISSIONS_REQUEST_SEND_SMS = 0;
65     //Flag to signal if the read/write requests are possible
66     private static final int REQUEST_EXTERNAL_STORAGE = 1;
67     //Flags to signal if read/write permissions are asked and available
68     private static final String[] PERMISSIONS_STORAGE = {
69             Manifest.permission.READ_EXTERNAL_STORAGE,
70             Manifest.permission.WRITE_EXTERNAL_STORAGE
71     };
72
```

```
73      //NAME OF THE FILE THAT STORES THE APP USER SETTINGS
74      private final static String filesettsname = "settings.txt";
75      // Get the absolute path to the Directory where the app can write/read
76      private final static String path = Environment.getExternalStorageDirectory().
        getAbsolutePath();
77
78      ////////////MiBand related Variables//////////
79      // time to live of a app reading request in milliseconds
80      private final Integer mibandTimeOut = 30000;
81      // delay to start counting the time to live of a app reading request in milliseconds
82      private final Integer mibandTimeOutDelay = 5000;
83      // flag to signal to trace if requests can be received
84      private Boolean isListeningHeartRate = false;
85      private Boolean isListeningBateryLevel = false;
86      //flag to signal if MiBand is commanded to vibrate
87      private Boolean vibrate = false;
88      //flags to signal if the given ubidots service key/id are valid
89      private Boolean validHeartRateKEY = false;
90      private Boolean validBatteryKEY = false;
91
92      /////////Timer/////////
93      //Timer auxiliary TimerTask Var
94      private TimerTask timerTask;
95      //Timer to do Heart Rate readings
96      private Timer timer = new Timer();
97      //Time delay to initiate the timer schedule in milliseconds
98      private int delay=30000;
99
100     ////////BLUETOOTH service related/////////
101     // the device
102     private BluetoothDevice bluetoothDevice;
103     // the adapters device
104     private BluetoothAdapter bluetoothAdapter;
105     // the generic Attributes structure class
106     //to enable communication with Bluetooth (MiBand2)
107     private BluetoothGatt bluetoothGatt;
108
109     ////////////LAYOUT////////////
110     //--BUTTONS
111     // Restart Bluetooth Connection to MiBand2 button
112     private Button btnStartConnecting;
113     // Request MiBand Battery level info button
114     private Button btnGetBatteryInfo;
115     // request MiBand to read heart rate button
116     private Button btnGetHeartRate;
117     // request MiBand to start/stop vibrate button
118     private Button btnStartVibrate;
119     // Show/hide configurations and settings layout button
120     private Button btnMiBand_show_cfg;
121     //--EDITABLE TEXT
122     // shows MiBands Physical MAC Address
123     private EditText txtPhysicalAddress;
124     // Shows the maximum heart rate value without alarm
125     private EditText maxBpmAlarm;
126     // Shows the minimum heart rate value without alarm
127     private EditText minBpmAlarm;
128     // shows how many minute are set on the reading heart rate timer
129     private EditText readMin;
130     // shows how many hours are set on the reading heart rate timer
131     private EditText readHour;
132     // defines the Ubidots key
```

```
133     private EditText ubiKey;
134     // defines the Ubidots heart rate variable ID
135     private EditText ubiHeartID;
136     // defines the Ubidots battery level variable ID
137     private EditText ubiBatID;
138     //defines the telephone number to send the sms alarm messages
139     private EditText txtPhone;
140     //--LAYOUT SETS
141     //Layout with MiBands heart reading related buttons
142     private View heartLy;
143     //Layout MiBands Find/vibrate and battery reading level buttons
144     private View FBatLy;
145     //Layout with Ubidots related key and variable Id settings
146     private View ubiLx;
147     //Layout with the heart rate reading time interval  related buttons
148     private View timerLx;
149     //Layout with the heart rate values alarm related buttons
150     private View bpmLx;
151     //--SEEKBARs
152     // Sets the minimum heart rate value without alarm
153     private SeekBar setMinBpm;
154     // Sets the maximum heart rate value without alarm
155     private SeekBar setMaxBpm;
156     // Sets how many minute are set on the reading heart rate timer
157     private SeekBar setTimerMin;
158     // Sets how many hours are set on the reading heart rate timer
159     private SeekBar setTimerHour;
160     //--ONLY TEXT
161     //says the state Disconnected
162     private TextView txtState;
163     //shows the image connected / connecting
164     private TextView txtStateImg;
165     //shows the heart rate value
166     private TextView txtBpm;
167     //shows the battery level value
168     private TextView txtBat;
169
170     //stores last red battery level
171     private int battlevel;
172     //stores last red heart rate
173     private int bpm;
174     // stores how many heart rate readings out of boundaries
175     private int probCounter = 0;
176
177     // stores the Ubidots key
178     private String KEY = "";
179     // stores the Ubidots battery level variable ID
180     private String BATTERY_ID = "";
181     // stores the Ubidots heart rate variable ID
182     private String HEART_RATE_ID = "";
183     // stores the phone number to which the sms is to be sent
184     private String phoneNo = "";
185     //default value of the maximum heart rate alarm
186     private Integer MaxBpmAlarm = 190;
187     //default value of the minimum heart rate alarm
188     private Integer MinBpmAlarm = 40;
189     //default value of the minutes on the heart rate readings timer
190     private Integer Min_TIMER = 20;
191     //default value of the hours on the heart rate readings timer
192     private Integer Hour_TIMER = 0;
193     //Auxiliary variable to pass context to some functions
```

```
194     private final Context context = this;
195
196     ///////TTL Timer of MiBand Answers//////
197     /**
198      * TTL Timer of MiBand Answers
199      *
200      * @mibandTimeOut the time to wait for the answer
201      * @mibandTimeOutDelay the time to wait until start waiting
202      */
203     private final CountDownTimer miBandTimeOut = new CountDownTimer(mibandTimeOut,
204             mibandTimeOutDelay) {
205         public void onTick(long millisUntilFinished) {
206                 }
207
208         public void onFinish() {
209             runOnUiThread(new Runnable() {
210                 @Override
211                 public void run() {
212                     txtBpm.setText("Failed to read bpm. Please, try again.");
213                 }
214             });
215
216         }
217     };
218
219     // the generic Attributes structure callback function
220     // to enable communication with Bluetooth device (MiBand2)
221     private final BluetoothGattCallback bluetoothGattCallback = new BluetoothGattCallback
        () {
222
223         @Override
224         public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState)
         {
225             super.onConnectionStateChange(gatt, status, newState);
226
227             //Bluetooth is connected?
228             if (newState == BluetoothProfile.STATE_CONNECTED) {
229                 //start discovering available services
230                 bluetoothGatt.discoverServices();
231                 ShowConnected(); //change to the layout 'connected state'
232                 //info the user
233                 //says the state connected
234                 runOnUiThread(new Runnable() {
235                     @Override
236                     public void run() {
237                         txtState.setText("Connected");
238                     }
239                 });
240             } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
241                 //disconect de generic Attributes service structure
242                 bluetoothGatt.disconnect();
243                 showConnect(); //change to the layout 'able to connect state'
244                 //info the user
245                 //says the state Disconnected
246                 runOnUiThread(new Runnable() {
247                     @Override
248                     public void run() {
249                         txtState.setText("Disconnected");
250                     }
251                 });
252             }
```

```
253
254         }
255
256         @Override
257         public void onServicesDiscovered(BluetoothGatt gatt, int status) {
258             super.onServicesDiscovered(gatt, status);
259             //there is a service
260             //listen to the heart rate reading
261             listenHeartRate();
262         }
263
264         @Override
265         public void onCharacteristicRead(BluetoothGatt gatt,
266                                          BluetoothGattCharacteristic characteristic, int
    status) {
267             super.onCharacteristicRead(gatt, characteristic, status);
268             //store ccharacteristic
269             final byte[] data = characteristic.getValue();
270             //get the battery level value
271             battlevel = (int) data[1];
272             // is the value valid
273             if (battlevel > 0) {
274                 //info the user
275                 runOnUiThread(new Runnable() {
276                     @Override
277                     public void run() {
278                         txtBat.setText(Integer.toString(battlevel)+ "%");
279                     }
280                 });
281                 //if the given ubidots service key/id are valid
282                 if (validBatteryKEY) {
283                     //sends the value to Ubidots service
284                     new ApiUbidots().execute(KEY, BATTERY_ID, battlevel + "");
285                 }
286             }
287         }
288
289         @Override
290         public void onCharacteristicWrite(BluetoothGatt gatt,
291                                           BluetoothGattCharacteristic characteristic, int
     status) {
292             super.onCharacteristicWrite(gatt, characteristic, status);
293         }
294
295         @Override
296         public void onCharacteristicChanged(BluetoothGatt gatt,
297                                             BluetoothGattCharacteristic characteristic) {
298             super.onCharacteristicChanged(gatt, characteristic);
299             String message, phoneNo = txtPhone.getText().toString();
300
301             //store characteristic
302             final byte[] data = characteristic.getValue();
303             //get the heart rate value
304             bpm = (int) data[1];
305             //cancel the TTL counter
306             miBandTimeOut.cancel();
307             //////////////is the value above normal//////////
308             if (bpm >= MaxBpmAlarm) {
309                 message = "The last heart rate value of " + bpm + " it's exceeding the "
310                         + MaxBpmAlarm + " bmp Maximum Value";
311                 //if it is not just a bad reading
```

```
312                 if (probCounter > 2) {
313                     //send message through sms services
314                     sendSMSMessage(message, phoneNo);
315                     //keep reading the heart rate
316                     runOnUiThread(new Runnable() {
317                         @Override
318                         public void run() {
319                             startScanHeartRate();
320                         }
321                     });
322                 }
323             }
324             ///////////////is the value under normal/////////
325             if ((bpm <= MinBpmAlarm) && (MinBpmAlarm >= 0)) {
326                 message = "The last heart rate value of " + bpm + " it's under the "
327                         + MaxBpmAlarm + " bmp Minimum Value";
328                 //if it is not just a bad reading
329                 if (probCounter > 2) {
330                     //send message through sms services
331                     sendSMSMessage(message, phoneNo);
332                     //keep reading the heart rate
333                     runOnUiThread(new Runnable() {
334                         @Override
335                         public void run() {
336                             startScanHeartRate();
337                         }
338                     });
339                 }
340             }
341             final String hrbpm = String.valueOf(bpm) + " bpm";
342             //if it is a error
343             if (bpm < 0) {
344                 //if it is a error ask user to check MiBand
345                 runOnUiThread(new Runnable() {
346                     @Override
347                     public void run() {
348                         txtBpm.setText("Adjust the MiBand for better reading");
349                     }
350                 });
351             } else {
352                 //if heart rate is normal
353                 if (bpm < MaxBpmAlarm && bpm > MinBpmAlarm) {
354                     probCounter = 0;
355                 } else {
356                     probCounter++;
357                 }
358                 //if the given ubidots service key/id are valid
359                 if (validHeartRateKEY) {
360                     //sends the value to Ubidots service
361                     new ApiUbidots().execute(KEY, HEART_RATE_ID, String.valueOf(bpm));
362                     //write the heaart rate in the layout
363                     runOnUiThread(new Runnable() {
364                         @Override
365                         public void run() {
366                             txtBpm.setText(hrbpm);
367 
368                         }
369                     });
370                     //verifies th battery level status
371                     runOnUiThread(new Runnable() {
372                         @Override
```

```
373                         public void run() {
374                             getBatteryStatus(txtBat);
375                         }
376                     });
377                 }
378             }
379         }
380
381         @Override
382         public void onDescriptorRead(BluetoothGatt gatt,
383                                      BluetoothGattDescriptor descriptor, int status) {
384             super.onDescriptorRead(gatt, descriptor, status);
385         }
386
387         @Override
388         public void onDescriptorWrite(BluetoothGatt gatt,
389                                       BluetoothGattDescriptor descriptor, int status) {
390             super.onDescriptorWrite(gatt, descriptor, status);
391         }
392
393         @Override
394         public void onReliableWriteCompleted(BluetoothGatt gatt, int status) {
395             super.onReliableWriteCompleted(gatt, status);
396         }
397
398         @Override
399         public void onReadRemoteRssi(BluetoothGatt gatt, int rssi, int status) {
400             super.onReadRemoteRssi(gatt, rssi, status);
401         }
402
403         @Override
404         public void onMtuChanged(BluetoothGatt gatt, int mtu, int status) {
405             super.onMtuChanged(gatt, mtu, status);
406         }
407
408     };
409
410     /**
411      * Checks if the app has permission to write to device storage
412      * <p>
413      * If the app does not has permission then the user will be prompted to grant
    permissions
414      *
415      * @param activity the main activity
416      */
417     private static void verifyStoragePermissions(Activity activity) {
418         // Check if we have write permission
419         int permission = ActivityCompat.checkSelfPermission(activity, Manifest.permission
    .WRITE_EXTERNAL_STORAGE);
420
421         if (permission != PackageManager.PERMISSION_GRANTED) {
422             // We don't have permission so prompt the user
423             Toast.makeText(activity, "This permission is needed to be able to save
    application settings.", Toast.LENGTH_LONG).show();
424             ActivityCompat.requestPermissions(
425                     activity,
426                     PERMISSIONS_STORAGE,
427                     REQUEST_EXTERNAL_STORAGE
428             );
429         }
430     }
```

```
431
432     /**
433      * Checks if external storage is available for read and write
434      */
435     private boolean isExternalStorageWritable() {
436         String state = Environment.getExternalStorageState();
437         if (Environment.MEDIA_MOUNTED.equals(state)) {
438             return true;
439         }
440         return false;
441     }
442
443     /**
444      * Checks if external storage is available to at least read
445      */
446     private boolean isExternalStorageReadable() {
447         String state = Environment.getExternalStorageState();
448         if (Environment.MEDIA_MOUNTED.equals(state) ||
449                 Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
450             return true;
451         }
452         return false;
453     }
454
455     @Override
456     protected void onCreate(Bundle savedInstanceState) {
457         super.onCreate(savedInstanceState);
458         setContentView(R.layout.activity_main);
459
460         verifyStoragePermissions(this);
461         isExternalStorageWritable();
462         isExternalStorageReadable();
463
464
465         initializeBluetoothDevice();
466         initializeUIComponents();
467
468         // writeSettings(this, filesettsname);
469         initializeEvents();
470         //writeSettings(this, filesettsname);
471         //
472         readSettings(this, filesettsname);
473         getBoundedDevice();
474         startConnecting();
475         ShowConnected();
476         //initiate a task to atuate the heart rate readings
477         timerTask = new TimerTask() {
478             public void run() {
479                 //activate reading
480                 startScanHeartRate();
481             }
482         };
483         //if there is a time interval
484         if ((Min_TIMER > 0) || (Hour_TIMER > 0))
485             //set the timer schedule
486             timer.scheduleAtFixedRate(timerTask, delay, Min_TIMER * 60000 + Hour_TIMER *
    3600000);
487     }
488
489     protected void onResume() {
490         super.onResume();
```

```
491         getBoundedDevice();
492         startConnecting();
493         ShowConnected();
494     }
495 
496     /**
497      * gets the bounded Bluetooth devices and chooses MiBand
498      */
499     private void getBoundedDevice() {
500         //stores MiBand2 MAC Address
501         String address = "";
502         //stores a list of Bluetooth bounded devices
503         Set<BluetoothDevice> boundedDevice;
504         //waits for the list of Bluetooth bounded devices
505         do {
506             //gets a list of Bluetooth bounded devices
507             boundedDevice = bluetoothAdapter.getBondedDevices();
508             // searches and gets MiBand2 MAC Address
509             for (BluetoothDevice bd : boundedDevice) {
510                 if (bd.getName().contains("MI Band 2")) {
511                     address = bd.getAddress();
512                     txtPhysicalAddress.setText(address);
513                 } else {
514                     Toast.makeText(this, "Waiting for MiBand2 Connection," +
515                             " be sure it's paired", Toast.LENGTH_SHORT).show();
516                 }
517             }
518         } while (address.isEmpty());
519     }
520 
521     /**
522      * initialize the Bluetooth Device
523      */
524     private void initializeBluetoothDevice() {
525         // Ask for location permission if not already allowed
526         if (ContextCompat.checkSelfPermission(this,
527                 Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.
    PERMISSION_GRANTED) {
528             Toast.makeText(this, "We need Bluetooth to be able to connect to MiBand2.",
529                     Toast.LENGTH_LONG).show();
530             ActivityCompat.requestPermissions(this,
531                     new String[]{Manifest.permission.ACCESS_COARSE_LOCATION}, 1);
532         }
533         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
534         if (bluetoothAdapter == null) {
535             // Device does not support Bluetooth
536             Toast.makeText(this, "Device does not support Bluetooth!",
537                     Toast.LENGTH_SHORT).show();
538             onPause();
539         }
540         //enables Bluetooth adapter
541         if (!bluetoothAdapter.isEnabled()) {
542             Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
543             startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
544             Toast.makeText(this, "Getting the  Bluetooth Device...",
545                     Toast.LENGTH_SHORT).show();
546         }
547     }
548 
549     /**
550      * Initializing UI components
```

```
551      */
552     private void initializeUIComponents() {
553         // Restart Bluetooth Connection to MiBand2 button
554         btnStartConnecting = (Button) findViewById(R.id.btnStartConnecting);
555         //Layout with MiBands heart reading related buttons
556         heartLy = findViewById(R.id.heartLy);
557         //Layout MiBands Find/vibrate and battery reading level buttons
558         FBatLy = findViewById(R.id.FBatLy);
559         //Layout with Ubidots related key and variable Id settings
560         ubiLx = findViewById(R.id.ubiLx);
561         //Layout with the heart rate reading time interval  related buttons
562         timerLx = findViewById(R.id.timerLx);
563         //Layout with the heart rate values alarm related buttons
564         bpmLx = findViewById(R.id.bpmLx);
565         // Request MiBand Battery level info button
566         btnGetBatteryInfo = (Button) findViewById(R.id.btnGetBatteryInfo);
567         // request MiBand to start/stop vibrate button
568         btnStartVibrate = (Button) findViewById(R.id.btnStartVibrate);
569         // request MiBand to read heart rate button
570         btnGetHeartRate = (Button) findViewById(R.id.btnGetHeartRate);
571         // Show/hide configurations and settings layout button
572         btnMiBand_show_cfg = (Button) findViewById(R.id.miBand_show_addr);
573         // shows MiBands Physical MAC Address
574         txtPhysicalAddress = (EditText) findViewById(R.id.txtPhysicalAddress);
575         //says the state connected / Disconnected
576         txtState = (TextView) findViewById(R.id.txtState);
577         //shows the image connected / connecting
578         txtStateImg = (TextView) findViewById(R.id.txtState2);
579         //shows the heart rate value
580         txtBpm = (TextView) findViewById(R.id.txtBpm);
581         //shows the battery level value
582         txtBat = (TextView) findViewById(R.id.textBat);
583         // Shows the maximum heart rate value without alarm
584         maxBpmAlarm = (EditText) findViewById(R.id.maxBpmAlarm);
585         // Shows the minimum heart rate value without alarm
586         minBpmAlarm = (EditText) findViewById(R.id.minBpmAlarm);
587         // shows how many minute are set on the reading heart rate timer
588         readMin = (EditText) findViewById(R.id.readMin);
589         // shows how many hours are set on the reading heart rate timer
590         readHour = (EditText) findViewById(R.id.readHour);
591         // defines the Ubidots key
592         ubiKey = (EditText) findViewById(R.id.ubiID);
593         // defines the Ubidots heart rate variable ID
594         ubiHeartID = (EditText) findViewById(R.id.ubiHeartKey);
595         // defines the Ubidots battery level variable ID
596         ubiBatID = (EditText) findViewById(R.id.ubiBatKey);
597         // Sets the maximum heart rate value without alarm
598         setMaxBpm = (SeekBar) findViewById(R.id.setMaxbpmAlarm);
599         // Sets the minimum heart rate value without alarm
600         setMinBpm = (SeekBar) findViewById(R.id.setMinbpmAlarm);
601         // Sets how many minute are set on the reading heart rate timer
602         setTimerMin = (SeekBar) findViewById(R.id.setTimeerMin);
603         // Sets how many hours are set on the reading heart rate timer
604         setTimerHour = (SeekBar) findViewById(R.id.setTimerHour);
605         //defines the telephone number to send the sms alarm messages
606         txtPhone = (EditText) findViewById(R.id.sosTelef);
607     }
608
609     /**
610      * Initializing UI components events
611      */
```

```
612     private void initializeEvents() {
613 
614         // Restart Bluetooth Connection to MiBand2 button
615         btnStartConnecting.setOnClickListener(new View.OnClickListener() {
616             @Override
617             public void onClick(View v) {
618                 //reconnects to MiBand2
619                 start();
620             }
621         });
622         // Request MiBand Battery level info button
623         btnGetBatteryInfo.setOnClickListener(new View.OnClickListener() {
624             @Override
625             public void onClick(View v) {
626                 // Request MiBand Battery level info
627                 getBatteryStatus(txtBat);
628             }
629         });
630         // request MiBand to start/stop vibrate button
631         btnStartVibrate.setOnClickListener(new View.OnClickListener() {
632             @Override
633             public void onClick(View v) {
634                 // request MiBand to start/stop vibrate
635                 startVibrate();
636             }
637         });
638         // request MiBand to read heart rate button
639         btnGetHeartRate.setOnClickListener(new View.OnClickListener() {
640             @Override
641             public void onClick(View v) {
642                 // request MiBand to read heart rate
643                 startScanHeartRate();
644             }
645         });
646         // Show/hide configurations and settings layout
647         btnMiBand_show_cfg.setOnClickListener(new View.OnClickListener() {
648             @Override
649             public void onClick(View v) {
650                 // Show/hide configurations and settings layout
651                 toggleCfg();
652             }
653         });
654         // defines the Ubidots key
655         ubiKey.setOnClickListener(new View.OnClickListener() {
656             @Override
657             public void onClick(View v) {
658                 // set the Ubidots key
659                 KEY = ubiKey.getText().toString();
660             }
661         });
662         // defines the Ubidots heart rate variable ID
663         ubiHeartID.setOnClickListener(new View.OnClickListener() {
664             @Override
665             public void onClick(View v) {
666                 // sets the Ubidots heart rate variable ID
667                 HEART_RATE_ID = ubiHeartID.getText().toString();
668                 //stores the key/id and value
669                 String[] keyvarArray = new String[]{KEY, HEART_RATE_ID};
670                 try {
671                     //stores if the given ubidots service key/id are valid
```

```
672                     validHeartRateKEY = new ApiUbidots_VerifyVarId().execute(keyvarArray)
    .get();
673                     //is the given ubidots service key/id are valid
674                     if (validHeartRateKEY) {
675                         //the given ubidots service key/id are valid
676                         //letters are showed in blue color
677                         runOnUiThread(new Runnable() {
678                             @Override
679                             public void run() {
680                                 ubiHeartID.setHighlightColor(Color.WHITE);
681                                 ubiHeartID.setTextColor(Color.BLUE);
682                             }
683                         });
684                     } else {
685                         //the given ubidots service key/id are NOT valid
686                         //letters are showed in red color
687                         runOnUiThread(new Runnable() {
688                             @Override
689                             public void run() {
690                                 ubiHeartID.setHighlightColor(Color.GRAY);
691                                 ubiHeartID.setTextColor(Color.RED);
692                                 cfgOn();
693                             }
694                         });
695                     }
696                 } catch (InterruptedException e) {
697                     Log.v("ubiHeart_InterruptedExc", "the exception is " + e.toString());
698                 } catch (ExecutionException e) {
699                     Log.v("ubiheart_ExecutionExc", "the exception is " + e.toString());
700                 }
701             }
702         });
703 
704         // defines the Ubidots battery level variable ID
705         ubiBatID.setOnClickListener(new View.OnClickListener() {
706             @Override
707             public void onClick(View v) {
708                 //stores the key/id and value
709                 BATTERY_ID = ubiBatID.getText().toString();
710                 //stores the key/id and value
711                 String[] keyvarArray = new String[]{KEY, BATTERY_ID};
712                 try {
713                     //stores if the given ubidots service key/id are valid
714                     validBatteryKEY = new ApiUbidots_VerifyVarId().execute(keyvarArray).
    get();
715                     //is the given ubidots service key/id are valid
716                     if (validBatteryKEY) {
717                         //the given ubidots service key/id are valid
718                         //letters are showed in blue color
719                         runOnUiThread(new Runnable() {
720                             @Override
721                             public void run() {
722                                 ubiBatID.setHighlightColor(Color.WHITE);
723                                 ubiBatID.setTextColor(Color.BLUE);
724                             }
725                         });
726                     } else {
727                         //the given ubidots service key/id are NOT valid
728                         //letters are showed in red color
729                         runOnUiThread(new Runnable() {
730                             @Override
```

```
731                             public void run() {
732                                 ubiBatID.setHighlightColor(Color.GRAY);
733                                 ubiBatID.setTextColor(Color.RED);
734                                 cfgOn();
735                             }
736                         });
737                     }
738                 } catch (InterruptedException e) {
739                     Log.v("ubibat_InterruptedExc", "the exception is " + e.toString());
740                 } catch (ExecutionException e) {
741                     Log.v("ubibat_ExecutionExcept", "the exception is " + e.toString());
742                 }
743             }
744         });
745 
746         // Sets the minimum heart rate value without firing the alarm
747         setMinBpm.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
748             //the current value
749             int value = setMinBpm.getProgress();
750 
751             @SuppressLint("SetTextI18n")
752             @Override
753             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser
    ) {
754                 //actualize the current value plus offset
755                 value = progress + ABSOLUTE_MIN_BPM;
756                 // Shows the current minimum heart rate value without alarm
757                 minBpmAlarm.setText(Integer.toString(value));
758             }
759 
760             @Override
761             public void onStartTrackingTouch(SeekBar seekBar) {
762                 //the top value is the current value of  the maximum heart rate alarm
    value
763                 setMinBpm.setMax(Integer.valueOf(maxBpmAlarm.getText().toString()) -
    ABSOLUTE_MIN_BPM);
764             }
765 
766             @Override
767             public void onStopTrackingTouch(SeekBar seekBar) {
768                 // Shows the final minimum heart rate value without firing the alarm
769                 minBpmAlarm.setText(Integer.toString(value));
770                 // stores value of the minimum heart rate alarm
771                 MinBpmAlarm = value;
772             }
773         });
774 
775         // Sets the maximum heart rate value without alarm
776         setMaxBpm.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
777             //the current value
778             int value = MaxBpmAlarm, min;
779 
780             @Override
781             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser
    ) {
782                 //actualize the current value plus offset
783                 value = progress + min;
784                 // Shows the maximum heart rate value without alarm
785                 maxBpmAlarm.setText(Integer.toString(value));
786             }
787 
```

```
788             @Override
789             public void onStartTrackingTouch(SeekBar seekBar) {
790                 //the lowest value is the current value of  the minimum heart rate alarm
    value
791                 min = Integer.valueOf(minBpmAlarm.getText().toString());
792                 //the top value is the offset
793                 // (diference between the value of very top acceptable value and
794                 // the minimum heart rate alarm value)
795                 setMaxBpm.setMax(ABSOLUTE_MAX_BPM - min);
796             }
797
798             @Override
799             public void onStopTrackingTouch(SeekBar seekBar) {
800                 // Shows the final maximum heart rate value without firing the alarm
801                 maxBpmAlarm.setText(Integer.toString(value));
802                 // stores value of the maximum heart rate alarm
803                 MaxBpmAlarm = value;
804             }
805         });
806
807         // Sets how many minute are set on the reading heart rate timer
808         setTimerMin.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
809             //the current value
810             int value = Min_TIMER;
811
812             @Override
813             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser
    ) {
814                 //actualize the current value
815                 value = progress;
816                 // Shows the current value
817                 readMin.setText(Integer.toString(value));
818             }
819
820             @Override
821             public void onStartTrackingTouch(SeekBar seekBar) {
822             }
823
824             @Override
825             public void onStopTrackingTouch(SeekBar seekBar) {
826                 // Shows the final value
827                 readMin.setText(Integer.toString(value));
828                 // stores the final value
829                 Min_TIMER = value;
830                 //restart the timer with the new setting//
831                 timer.cancel(); //cancel actual timer
832                 //if there is a time interval
833                 if ((Min_TIMER > 0) || (Hour_TIMER > 0)) {
834                     //initiate a task to atuate the heart rate readings
835                     timerTask = new TimerTask() {
836                         public void run() {
837                             //activate reading
838                             startScanHeartRate();//requests reading heart rate value
839                         }
840                     };
841                     timer = new Timer(); //new Timer instance
842                     //finally schedulee the task with the new time settings
843                     timer.scheduleAtFixedRate(timerTask, delay, Min_TIMER * 60000 +
    Hour_TIMER * 3600000);
844                 }
845             }
```

```
846         });
847
848         // Sets how many hours are set on the reading heart rate timer
849         setTimerHour.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
850             //the current value
851             int value = Hour_TIMER;
852
853             @Override
854             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser
    ) {
855                 //actualize the current value
856                 value = progress;
857                 // Shows the current value
858                 readHour.setText(Integer.toString(value));
859             }
860
861             @Override
862             public void onStartTrackingTouch(SeekBar seekBar) {
863             }
864
865             @Override
866             public void onStopTrackingTouch(SeekBar seekBar) {
867                 // Shows the final value
868                 readHour.setText(Integer.toString(value));
869                 // stores the final value
870                 Hour_TIMER = value;
871                 //restart the timer with the new setting//
872                 timer.cancel(); //cancel actual timer
873                 //if there is a time interval
874                 if ((Min_TIMER > 0) || (Hour_TIMER > 0)) {
875                     //initiate a task to atuate the heart rate readings
876                     timerTask = new TimerTask() {
877                         public void run() {
878                             //activate reading
879                             startScanHeartRate();//requests reading heart rate value
880                         }
881                     };
882                     timer = new Timer(); //new Timer instance
883                     //finally schedulee the task with the new time settings
884                     timer.scheduleAtFixedRate(timerTask, delay, Min_TIMER * 60000 +
    Hour_TIMER * 3600000);
885                 }
886             }
887         });
888
889         //defines the telephone number to send the sms alarm messages
890         txtPhone.setOnClickListener(new View.OnClickListener() {
891             @Override
892             public void onClick(View v) {
893                 phoneNo = txtPhone.getText().toString();
894             }
895         });
896     }
897
898     /**
899      * reconnects to MiBand2
900      */
901     private void start() {
902         //hide thwe reconect button (already reconnecting)
903         btnStartConnecting.setVisibility(View.GONE);
904         //initialize the Bluetooth Device if must to
```

```
905         initializeBluetoothDevice();
906         //gets the bounded Bluetooth devices and chooses MiBand MAC
907         getBoundedDevice();
908         //connects to the stored MAC address
909         startConnecting();
910         //changes the layout items showing the connected state
911         ShowConnected();
912     }
913 
914     /**
915      * toggles the config layout on/off
916      */
917     private void toggleCfg() {
918         //If the txtPhysicalAddress() is visible
919         if (txtPhysicalAddress.getVisibility() == View.VISIBLE) {
920             //he txtPhysicalAddress() is visible, the config layout is on
921             cfgOff(true); //then turn it off
922         } else {
923             //he txtPhysicalAddress() is not visible, the config layout is off
924             cfgOn(); //then turn it on
925         }
926     }
927 
928     /**
929      * turns the config layout off
930      *
931      * @param save boolean
932      *             if true, saves the settings
933      *             to the settings file
934      */
935     private void cfgOff(boolean save) {
936         // shows MiBands Physical MAC Address
937         txtPhysicalAddress.setVisibility(View.INVISIBLE);
938         // Restart Bluetooth Connection to MiBand2 button
939         btnStartConnecting.setVisibility(View.GONE);
940         //Layout with the heart rate values alarm related buttons
941         bpmLx.setVisibility(View.GONE);
942         //Layout with the heart rate reading time interval  related buttons
943         timerLx.setVisibility(View.GONE);
944         //Layout with Ubidots related key and variable Id settings
945         ubiLx.setVisibility(View.GONE);
946         //Layout with MiBands heart reading related buttons
947         heartLy.setVisibility(View.VISIBLE);
948         //Layout MiBands Find/vibrate and battery reading level buttons
949         FBatLy.setVisibility(View.VISIBLE);
950         //if true, saves the settings to the settings file
951         if (save) writeSettings(context, filesettsname);
952     }
953 
954     /**
955      * turns the config layout on
956      */
957     private void cfgOn() {
958         // shows MiBands Physical MAC Address
959         txtPhysicalAddress.setVisibility(View.VISIBLE);
960         // Restart Bluetooth Connection to MiBand2 button
961         btnStartConnecting.setVisibility(View.GONE);
962         //Layout with the heart rate values alarm related buttons
963         bpmLx.setVisibility(View.VISIBLE);
964         //Layout with the heart rate reading time interval  related buttons
965         timerLx.setVisibility(View.VISIBLE);
```

```
966         //Layout with Ubidots related key and variable Id settings
967         ubiLx.setVisibility(View.VISIBLE);
968         //Layout with MiBands heart reading related buttons
969         heartLy.setVisibility(View.GONE);
970         //if true, saves the settings to the settings file
971         FBatLy.setVisibility(View.GONE);
972     }
973 
974     /**
975      * changes the layout items showing the connecting state
976      */
977     private void showConnect() {
978         runOnUiThread(new Runnable() {
979             @Override
980             public void run() {
981                 //hide thwe reconect button (already reconnecting)
982                 btnStartConnecting.setVisibility(View.VISIBLE);
983                 // shows MiBands Physical MAC Address
984                 txtPhysicalAddress.setVisibility(View.INVISIBLE);
985                 //Layout with the heart rate values alarm related buttons
986                 bpmLx.setVisibility(View.GONE);
987                 //Layout with the heart rate reading time interval  related buttons
988                 timerLx.setVisibility(View.GONE);
989                 //Layout with Ubidots related key and variable Id settings
990                 ubiLx.setVisibility(View.GONE);
991                 //Layout with MiBands heart reading related buttons
992                 heartLy.setVisibility(View.GONE);
993                 //Layout MiBands Find/vibrate and battery reading level buttons
994                 FBatLy.setVisibility(View.GONE);
995                 //says the state Disconnected
996                 txtState.setText("Disconnected");
997                 //shows the image Disconnected
998                 txtStateImg.setBackground(getResources().getDrawable(R.drawable.
    bluetooth_off));
999             }
1000         });
1001     }
1002 
1003     /**
1004      * changes the layout items showing the connected state
1005      */
1006     private void ShowConnected() {
1007         runOnUiThread(new Runnable() {
1008             @Override
1009             public void run() {
1010                 cfgOff(false);
1011                 //says the state connected
1012                 txtState.setText("Connected");
1013                 //shows the image connected
1014                 txtStateImg.setBackground(getResources().getDrawable(R.drawable.bluetooth
    ));
1015 
1016             }
1017         });
1018     }
1019 
1020     /**
1021      * connects to the stored MAC address
1022      */
1023     private void startConnecting() {
1024         String address = txtPhysicalAddress.getText().toString();
```

```
1025
1026            bluetoothDevice = bluetoothAdapter.getRemoteDevice(address);
1027            Log.v("startConnecting", "Connecting to " + address);
1028            Log.v("startConnecting", "Device name " + bluetoothDevice.getName());
1029            bluetoothGatt = bluetoothDevice.connectGatt(this, true, bluetoothGattCallback);
1030        }
1031
1032        /**
1033         * requests reading heart rate value
1034         */
1035        private void startScanHeartRate() {
1036            // is there a GATT (the generic Attributes service structure)
1037            if (bluetoothGatt != null) {
1038                // is the service there
1039                if (bluetoothGatt.getService(CustomBluetoothProfile.Basic.service) == null) {
1040                    //the service exists flag signal tracer
1041                    isListeningHeartRate = true;
1042                    return;
1043                }
1044                //info user wait for the reading ends
1045                runOnUiThread(new Runnable() {
1046                    @Override
1047                    public void run() {
1048                        txtBpm.setText("Started Reading HeartRate");
1049                    }
1050                });
1051                //get the characteristic (heart rate reading) from the service
1052                BluetoothGattCharacteristic bchar = bluetoothGatt.getService(
    CustomBluetoothProfile.HeartRate.service)
1053                        .getCharacteristic(CustomBluetoothProfile.HeartRate.
    controlCharacteristic);
1054                //write the value to be writen on the characteristic (heart rate reading)
1055                bchar.setValue(new byte[]{21, 2, 1});
1056                //write the value in the characteristic (heart rate reading)
1057                bluetoothGatt.writeCharacteristic(bchar);
1058                //the service exists flag signal tracer
1059                isListeningHeartRate = true;
1060                //start the reading timer
1061                miBandTimeOut.start();
1062            } else {
1063                Toast.makeText(this, "Bluetooth gatt is nuked, no bpm reading", Toast.
    LENGTH_SHORT).show();
1064
1065            }
1066        }
1067
1068        /**
1069         * listen to heart rate readings
1070         */
1071        private void listenHeartRate() {
1072            //get the characteristic value (heart rate reading) from the service
1073            BluetoothGattCharacteristic bchar = bluetoothGatt.getService(
    CustomBluetoothProfile.HeartRate.service)
1074                    .getCharacteristic(CustomBluetoothProfile.HeartRate.
    measurementCharacteristic);
1075            //enable characteristic notification (heart rate reading)
1076            bluetoothGatt.setCharacteristicNotification(bchar, true);
1077            //get the Bluetooth profile descriptor (heart rate reading)
1078            BluetoothGattDescriptor descriptor = bchar.getDescriptor(CustomBluetoothProfile.
    HeartRate.descriptor);
1079            //enable descriptor notification on the characteristic (heart rate reading)
```

```
1080        descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
1081        //write the descriptor with notifications enable
1082        bluetoothGatt.writeDescriptor(descriptor);
1083        //the service exists flag signal tracer
1084        isListeningHeartRate = true;
1085    }
1086
1087    /**
1088     * requests reading battery level value
1089     */
1090    private void getBatteryStatus(TextView txtBat) {
1091        //to store temporarily a GATT service (GATT the generic Attributes service
    structure)
1092        BluetoothGattService serviceTemp;
1093        //to store temporarily a GATT service  characteristic (GATT the generic
    Attributes service structure)
1094        BluetoothGattCharacteristic bchar;
1095
1096        // is there a GATT (the generic Attributes service structure)
1097        if (bluetoothGatt != null) {
1098            //store temporarily the GATT service
1099            serviceTemp = bluetoothGatt.getService(CustomBluetoothProfile.Basic.service);
1100            // is the service there
1101            if (serviceTemp == null) {
1102                Toast.makeText(this, "...waiting for miBand2 battery level answer...",
    Toast.LENGTH_SHORT).show();
1103                //the service exists flag signal tracer
1104                isListeningBateryLevel = true;
1105                return;
1106            }
1107            //get the characteristic value (battery level reading) from the service
1108            bchar = serviceTemp.getCharacteristic(CustomBluetoothProfile.Basic.
    batteryCharacteristic);
1109            //get and store the value red on the characteristic (battery level reading)
1110            byte[] z = bchar.getValue();
1111            //is the value on the characteristic been red (battery level reading)
1112            if (!bluetoothGatt.readCharacteristic(bchar)) {
1113                Toast.makeText(this, "Failed get battery info", Toast.LENGTH_SHORT).show
    ();
1114            } else {
1115                //got a valid battery level value
1116                if (z != null) {
1117                    //get the battery level value
1118                    battlevel = (int) z[1];
1119                    //if the given ubidots service key/id are valid
1120                    if (validBatteryKEY) {
1121                        //sends the value to Ubidots service
1122                        new ApiUbidots().execute(KEY, BATTERY_ID, battlevel + "");
1123                        // compose a string to show  in layout
1124                        String bat = String.valueOf(battlevel) + "%";
1125                        //show battery level value in layout
1126                        txtBat.setText(bat);
1127                    }
1128                }
1129            }
1130        } else {
1131            Toast.makeText(this, "Bluetooth gatt is nuked, no battery level reading",
    Toast.LENGTH_SHORT).show();
1132        }
1133    }
1134
```

```
1135     /**
1136      * requests MiBand to start/stop vibrating
1137      */
1138     private void startVibrate() {
1139         // is there a GATT (the generic Attributes service structure)
1140         if (bluetoothGatt != null) {
1141             //get the characteristic value (vibrate) from the service
1142             BluetoothGattCharacteristic bchar = bluetoothGatt
1143                     .getService(CustomBluetoothProfile.AlertNotification.service)
1144                     .getCharacteristic(CustomBluetoothProfile.AlertNotification.
     alertCharacteristic);
1145             // is not vibrating
1146             if (!vibrate) {
1147                 //get and store the value red on the characteristic (vibrate)
1148                 bchar.setValue(new byte[]{2});
1149                 //is the value on the characteristic been written (vibrate)
1150                 if (!bluetoothGatt.writeCharacteristic(bchar)) {
1151                     Toast.makeText(this, "Failed start vibrate", Toast.LENGTH_SHORT).show
     ();
1152                 } else {
1153                     //is vibrating so change the flag
1154                     vibrate = true;
1155                     //change information for the user know the new state
1156                     btnStartVibrate.setText("Found/Stop Vibrate");
1157                 }
1158             } else {
1159                 //is vibrating
1160                 //get and store the value red on the characteristic (vibrate)
1161                 bchar.setValue(new byte[]{0});
1162                 //is the value on the characteristic been written (vibrate)
1163                 if (!bluetoothGatt.writeCharacteristic(bchar)) {
1164                     Toast.makeText(this, "Failed stop vibrate", Toast.LENGTH_SHORT).show
     ();
1165                 } else {
1166                     //is vibrating so change the flag
1167                     vibrate = false;
1168                     //change information for the user know the new state
1169                     btnStartVibrate.setText("Find/Start Vibrate");
1170                 }
1171             }
1172         }
1173     }
1174
1175     /**
1176      * reads Ubidots configurations Keys and IDs,
1177      * Timer and Heart Rate reading Variables settings
1178      *
1179      * @param context  Activity context
1180      * @param filename name of the file that stores the Vars and Confs.
1181      */
1182     private void readSettings(Context context, String filename) {
1183         int index; //Auxiliary var to store temporary values
1184         try {
1185             //open File
1186             FileInputStream fs = context.openFileInput(filename);
1187             //get stream reader for the File
1188             InputStreamReader isr = new InputStreamReader(fs, "UTF-8");
1189             //Get a Buffered reader for the Stream Reader
1190             BufferedReader br = new BufferedReader(isr);
1191             //read a line
1192             String line = br.readLine();
```

```
1193            //line is not null
1194            if (line != null) {
1195                //stores only the value, cutting the user info part
1196                KEY = line.substring(line.indexOf("=") + 2);
1197                //Show the value in the layout
1198                runOnUiThread(new Runnable() {
1199                    @Override
1200                    public void run() {
1201                        ubiKey.setText(KEY);
1202                    }
1203                });
1204            }
1205            //read a line
1206            line = br.readLine();
1207            //line is not null
1208            if (line != null) {
1209                //stores only the value, cutting the user info part
1210                BATTERY_ID = line.substring(line.indexOf("=") + 2);
1211                //Show the value in the layout
1212                runOnUiThread(new Runnable() {
1213                    @Override
1214                    public void run() {
1215                        ubiBatID.setText(BATTERY_ID);
1216                    }
1217                });
1218                //stores the key/id and value
1219                String[] keyvarArray = new String[]{KEY, BATTERY_ID};
1220                try {
1221                    //stores if the given ubidots service key/id are valid
1222                    validBatteryKEY = new ApiUbidots_VerifyVarId().execute(keyvarArray).
    get();
1223                    //is the given ubidots service key/id are valid
1224                    if (validBatteryKEY) {
1225                        //the given ubidots service key/id are valid
1226                        //letters are showed in blue color
1227                        runOnUiThread(new Runnable() {
1228                            @Override
1229                            public void run() {
1230                                ubiBatID.setHighlightColor(Color.WHITE);
1231                                ubiBatID.setTextColor(Color.BLUE);
1232                            }
1233                        });
1234                    } else {
1235                        //the given ubidots service key/id are NOT valid
1236                        //letters are showed in red color
1237                        runOnUiThread(new Runnable() {
1238                            @Override
1239                            public void run() {
1240                                ubiBatID.setHighlightColor(Color.GRAY);
1241                                ubiBatID.setTextColor(Color.RED);
1242                                cfgOn();
1243                            }
1244                        });
1245                    }
1246                } catch (InterruptedException e) {
1247                    Log.v("ubibat_InterruptedExc", "the exception is " + e.toString());
1248                } catch (ExecutionException e) {
1249                    Log.v("ubibat_ExecutionExcept", "the exception is " + e.toString());
1250                }
1251            }
1252            //read a line
```

```
1253            line = br.readLine();
1254            //line is not null
1255            if (line != null) {
1256                //stores only the value, cutting the user info part
1257                HEART_RATE_ID = line.substring(line.indexOf("=") + 2);
1258                //Show the value in the layout
1259                runOnUiThread(new Runnable() {
1260                    @Override
1261                    public void run() {
1262                        ubiHeartID.setText(HEART_RATE_ID);
1263                    }
1264                });
1265                //stores the key/id and value
1266                String[] keyvarArray = new String[]{KEY, HEART_RATE_ID};
1267                try {
1268                    //stores if the given ubidots service key/id are valid
1269                    validHeartRateKEY = new ApiUbidots_VerifyVarId().execute(keyvarArray)
    .get();
1270                    //is the given ubidots service key/id are valid
1271                    if (validHeartRateKEY) {
1272                        //the given ubidots service key/id are valid
1273                        //letters are showed in blue color
1274                        runOnUiThread(new Runnable() {
1275                            @Override
1276                            public void run() {
1277                                ubiHeartID.setHighlightColor(Color.WHITE);
1278                                ubiHeartID.setTextColor(Color.BLUE);
1279                            }
1280                        });
1281                    } else {
1282                        Log.v("ubiVar", " Variable id <<INVALID>>: ");
1283                        //the given ubidots service key/id are NOT valid
1284                        //letters are showed in red color
1285                        runOnUiThread(new Runnable() {
1286                            @Override
1287                            public void run() {
1288                                ubiHeartID.setHighlightColor(Color.GRAY);
1289                                ubiHeartID.setTextColor(Color.RED);
1290                                cfgOn();
1291                            }
1292                        });
1293                    }
1294                } catch (InterruptedException e) {
1295                    Log.v("ubiHeart_InterruptedExc", "the exception is " + e.toString());
1296                } catch (ExecutionException e) {
1297                    Log.v("ubiheart_ExecutionExc", "the exception is " + e.toString());
1298                }
1299            }
1300            //read a line
1301            line = br.readLine();
1302            //line is not null
1303            if (line != null) {
1304                //stores only the value, cutting the user info part
1305                index = line.indexOf("=");
1306                MaxBpmAlarm = Integer.parseInt(line.substring(index + 2));
1307                //Show the value in the layout
1308                runOnUiThread(new Runnable() {
1309                    @Override
1310                    public void run() {
1311                        maxBpmAlarm.setText(Integer.toString(MaxBpmAlarm));
1312                        setMaxBpm.setProgress(MaxBpmAlarm);
```

```
1313                    }
1314                });
1315            }
1316            //read a line
1317            line = br.readLine();
1318            //line is not null
1319            if (line != null) {
1320                //stores only the value, cutting the user info part
1321                index = line.indexOf("=");
1322                MinBpmAlarm = Integer.parseInt(line.substring(index + 2));
1323                //Show the value in the layout
1324                runOnUiThread(new Runnable() {
1325                    @Override
1326                    public void run() {
1327                        minBpmAlarm.setText(Integer.toString(MinBpmAlarm));
1328                        setMinBpm.setProgress(MinBpmAlarm);
1329                    }
1330                });
1331            }
1332            //read a line
1333            line = br.readLine();
1334            //line is not null
1335            if (line != null) {
1336                //stores only the value, cutting the user info part
1337                index = line.indexOf("=");
1338                Min_TIMER = Integer.parseInt(line.substring(index + 2));
1339                //Show the value in the layout
1340                runOnUiThread(new Runnable() {
1341                    @Override
1342                    public void run() {
1343                        readMin.setText(Integer.toString(Min_TIMER));
1344                        setTimerMin.setProgress(Min_TIMER);
1345                    }
1346                });
1347            }
1348            //read a line
1349            line = br.readLine();
1350            //line is not null
1351            if (line != null) {
1352                //stores only the value, cutting the user info part
1353                index = line.indexOf("=");
1354                Hour_TIMER = Integer.parseInt(line.substring(index + 2));
1355                //Show the value in the layout
1356                runOnUiThread(new Runnable() {
1357                    @Override
1358                    public void run() {
1359                        readHour.setText(Integer.toString(Hour_TIMER));
1360                        setTimerHour.setProgress(Hour_TIMER);
1361                    }
1362                });
1363            }
1364            //read a line
1365            line = br.readLine();
1366            //line is not null
1367            if (line != null) {
1368                //stores only the value, cutting the user info part
1369                index = line.indexOf("=");
1370                phoneNo = line.substring(index + 2);
1371                //Show the value in the layout
1372                runOnUiThread(new Runnable() {
1373                    @Override
```

```
                    public void run() {
                        txtPhone.setText(phoneNo);
                    }
                });
            }
            //close FileStreamReader
            fs.close();
            //informs the user that the file has been red
            Toast.makeText(getBaseContext(),
                    "Done reading SD 'mysdfile.txt'",
                    Toast.LENGTH_SHORT).show();
        } catch (FileNotFoundException e) {
            Log.v("write file not found", e.getMessage());
        } catch (UnsupportedEncodingException e) {
            Log.v("UnsupportedEncoding", e.getMessage());
        } catch (IOException e) {
            Log.v("IO exception", e.getMessage());
        }
    }

    /**
     * writes Ubidots configurations Keys and IDs,
     * Timer and Heart Rate reading Variables settings
     *
     * @param context  activity context
     * @param fileName the name of the file to be written
     */
    private void writeSettings(Context context, String fileName) {
        try {
            //create File class instant
            File file = new File(path, fileName);
            // is the File has not benn created yet
            if (!file.exists()) {
                //create a File
                file.createNewFile();
            }
            // File output streamer to write in the file
            FileOutputStream fs = context.openFileOutput(fileName, Context.MODE_PRIVATE);
            //string to be written containing Ubidots configurations Keys and IDs,
            //Timer and Heart Rate reading Variables settings
            String s = "KEY = " + KEY + "\n"
                    + "BATTERY_ID = " + BATTERY_ID + "\n"
                    + "HEART_RATE_ID = " + HEART_RATE_ID + "\n"
                    + "MaxBpmAlarm = " + MaxBpmAlarm + "\n"
                    + "MinBpmAlarm = " + MinBpmAlarm + "\n"
                    + "Min_TIMER = " + Min_TIMER + "\n"
                    + "Hour_TIMER = " + Hour_TIMER + "\n"
                    + "SOS_PHONE = " + phoneNo;
            //write the string in the file
            fs.write(s.getBytes());
            //close FileStream writer
            fs.close();
            // Tell the media scanner about the new file so that it is
            // immediately available to the user.
            MediaScannerConnection.scanFile(this,
                    new String[]{file.toString()}, null,
                    new MediaScannerConnection.OnScanCompletedListener() {
                        public void onScanCompleted(String path, Uri uri) {
                            Log.i("ExternalStorage", "Scanned " + path + ":");
                            Log.i("ExternalStorage", "-> uri=" + uri);
                        }
```

```
1435                    });
1436
1437            Toast.makeText(getBaseContext(),
1438                    "Done writing SD " + fileName,
1439                    Toast.LENGTH_SHORT).show();
1440            Log.v("writing file", "Done writing to path " + path);
1441        } catch (FileNotFoundException e) {
1442            Log.v("write file not found", e.getMessage());
1443        } catch (UnsupportedEncodingException e) {
1444            Log.v("UnsupportedEncoding", e.getMessage());
1445        } catch (IOException e) {
1446            Log.v("IO exception", e.getMessage());
1447        }
1448
1449    }
1450
1451    /**
1452     * checks permission to send a sms message
1453     * written on a String variable named 'message'
1454     * to a number stored on a String var named 'phoneNo'
1455     *
1456     * @param smsMessage  String containing a valid sms message to send
1457     * @param phoneNumber String Containing a valid telephone number
1458     */
1459    private void sendSMSMessage(String smsMessage, String phoneNumber) {
1460        if (ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS) !=
    PackageManager.PERMISSION_GRANTED) {
1461            Log.v("sendmsg", "true-> (ContextCompat.checkSelfPermission(this, Manifest.
    permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED)");
1462            if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.
    permission.SEND_SMS)) {
1463                //      Toast.makeText(getApplicationContext(), "This permission is
    needed to be able to send SOS SMS.", Toast.LENGTH_LONG).show();
1464                Log.v("sendmsg", "true-> (ActivityCompat.
    shouldShowRequestPermissionRationale(this, Manifest.permission.SEND_SMS))");
1465            } else {
1466                //      Toast.makeText(getApplicationContext(), "This permission is needed
     to be able to send SOS SMS.", Toast.LENGTH_LONG).show();
1467                ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.
    SEND_SMS}, MY_PERMISSIONS_REQUEST_SEND_SMS);
1468                Log.v("sendmsg", "ActivityCompat.requestPermissions(this, new String[]{
    Manifest.permission.SEND_SMS}, MY_PERMISSIONS_REQUEST_SEND_SMS);");
1469            }
1470        } else {
1471            Log.v("sendmsg", "true-> (ContextCompat.checkSelfPermission(this, Manifest.
    permission.SEND_SMS) == PackageManager.PERMISSION_GRANTED)");
1472            //   Toast.makeText(getApplicationContext(), "Please Enter a Valid Phone
    Number", Toast.LENGTH_SHORT).show();
1473            Log.v("sendMessage", "sms sent");
1474            SmsManager smsManager = SmsManager.getDefault();
1475            smsManager.sendTextMessage(phoneNumber, null, smsMessage, null, null);
1476        }
1477    }
1478
1479    @Override
1480    public void onRequestPermissionsResult(int requestCode,
1481                                           String permissions[], int[] grantResults) {
1482        //on request permission code
1483        switch (requestCode) {
1484            //get the permission to send sms messages
1485            case MY_PERMISSIONS_REQUEST_SEND_SMS: {
```

```
1486                if (grantResults.length > 0 && grantResults[0] == PackageManager.
    PERMISSION_GRANTED) {
1487                    // The sms manager
1488                    SmsManager smsManager = SmsManager.getDefault();
1489                    // Variable to store the message to be sent
1490                    String message = "Autorization test";
1491                    //phone number to which the message will be send
1492                    String phoneNo = txtPhone.getText().toString();
1493                    //is there a pone number
1494                    if (phoneNo.isEmpty()) {
1495                        Toast.makeText(getApplicationContext(), "Please Enter a Valid
    Phone Number", Toast.LENGTH_SHORT).show();
1496                    } else {
1497                        //there is a phone number
1498                        //the message will be send to the phone number
1499                        smsManager.sendTextMessage(phoneNo, null, message, null, null);
1500                        Toast.makeText(getApplicationContext(), "SMS sent.", Toast.
    LENGTH_LONG).show();
1501                    }
1502                } else {
1503                    Toast.makeText(getApplicationContext(), "SMS faild, please try again.
    ", Toast.LENGTH_LONG).show();
1504                    Log.v("onReqPermResult", "dont have permission to send sms ");
1505                    //            return;
1506                }
1507            }
1508        }
1509    }
1510
1511    /**
1512     * Api from ubidots to
1513     * store values in Ubidots remote Variables
1514     * <p>
1515     * delivers values (params[2])
1516     * to variable with ID (parmas[1])
1517     * in a account with the valid KEY (params[0])
1518     *
1519     * @params is a String[] array
1520     * <p>
1521     * <p>
1522     * use:  ApiUbidots(String[])
1523     */
1524    public class ApiUbidots extends AsyncTask<String, Void, Void> {
1525
1526        @Override
1527        protected Void doInBackground(String... params) {
1528            //getting the Ubidots Api Client off account with the valid KEY (params[0])
1529            ApiClient apiClient = new ApiClient(params[0]);
1530            //getting the Ubidots Variable with ID (params[1])
1531            Variable variable = apiClient.getVariable(params[1]);
1532            //if the Variable ID and the account Key is Valid
1533            if (variable != null) {
1534                //sends the values (params[2]) in the Ubidots Variable
1535                variable.saveValue(Integer.valueOf(params[2]));
1536            } else {
1537                Log.v("UBI_SEND_FAIL", "FAIL to send Ubidots  rate value: " + params[2]
1538                        + " to " + params[1] + " Variable");
1539            }
1540            return null;
1541        }
1542    }
```

```
1543
1544     /**
1545      * Api from ubidots
1546      * to verify if the Ubidots Account key and
1547      * Variable ID are valid
1548      * <p>
1549      * in variable with ID (parmas[1])
1550      * in a account with the valid KEY (params[0])
1551      *
1552      * @params is a String[] array
1553      * @returns bollean true if both are valid
1554      * <p>
1555      * use:  ApiUbidots_VerifyVarId(String[])
1556      */
1557     public class ApiUbidots_VerifyVarId extends AsyncTask<String, Void, Boolean> {
1558         boolean keyIsValid = false;
1559
1560         @Override
1561         protected Boolean doInBackground(String... params) {
1562             //getting the Ubidots Api Client off account with the valid KEY (params[0])
1563             ApiClient apiClient = new ApiClient(params[0]);
1564             //getting the Ubidots Variable with ID (params[1])
1565             Variable variable = apiClient.getVariable(params[1]);
1566             //if the Variable ID and the account Key is Valid
1567             if (variable != null) {
1568                 return true;
1569             }
1570             return false;
1571         }
1572
1573         @Override
1574         protected void onPostExecute(Boolean isValidVarID) {
1575             super.onPostExecute(isValidVarID);
1576         }
1577     }
1578 }
```

5.2 Helpers: CustomBluetoothProfile.java

```
1  package com.id.hrm4miband2.Helpers;
2
3  import java.util.UUID;
4
5  /**
6   * Created by aashari on 21/05/17.
7   * Modified by Pedro Costa in 29/11/17
8   * This profile generated based on http://jellygom.com/2016/09/30/Mi-Band-UUID.html
9   */
10
11 public class CustomBluetoothProfile {
12
13     public static class Basic {
14         public static UUID service = UUID.fromString("0000fee0-0000-1000-8000-00805
    f9b34fb");
15         public static UUID batteryCharacteristic = UUID.fromString("
    00000006-0000-3512-2118-0009af100700");
16     }
17
18     public static class AlertNotification {
19         public static UUID service = UUID.fromString("00001802-0000-1000-8000-00805
```

```
    f9b34fb");
20      public static UUID alertCharacteristic = UUID.fromString("00002a06
    -0000-1000-8000-00805f9b34fb");
21  }
22
23  public static class HeartRate {
24      public static UUID service = UUID.fromString("0000180d-0000-1000-8000-00805
    f9b34fb");
25      public static UUID measurementCharacteristic = UUID.fromString("00002a37
    -0000-1000-8000-00805f9b34fb");
26      public static UUID descriptor = UUID.fromString("00002902-0000-1000-8000-00805
    f9b34fb");
27      public static UUID controlCharacteristic = UUID.fromString("00002a39
    -0000-1000-8000-00805f9b34fb");
28  }
29
30 }
```

5.3 AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.id.hrm4miband2">
4
5
6      <uses-permission android:name="android.permission.INTERNET"/>
7      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
8      <uses-permission android:name="android.permission.BLUETOOTH" />
9      <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
10
11     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
12     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
13
14     <uses-permission android:name="android.permission.SEND_SMS" />
15
16     <!--BLE scanning is commonly used to determine a user's
17         location with Bluetooth LE beacons. -->
18     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
19
20     <!-- if your app targets API level 21 or higher. -->
21     <uses-feature android:name="android.hardware.location.gps" />
22
23     <!--app is available to BLE-capable devices only. -->
24     <uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
25
26     <application
27         android:allowBackup="false"
28         android:icon="@mipmap/ic_launcher"
29         android:label="@string/app_name"
30         android:roundIcon="@mipmap/ic_launcher_round"
31         android:supportsRtl="true"
32         android:theme="@style/AppTheme">
33         <activity android:name="com.id.hrm4miband2.Activities.MainActivity"
34           android:screenOrientation="portrait">
35           <intent-filter>
36               <action android:name="android.intent.action.MAIN" />
37
38               <category android:name="android.intent.category.LAUNCHER" />
39           </intent-filter>
40       </activity>
```

```
    </application>

</manifest>
```

5.4 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context="com.id.hrm4miband2.Activities.MainActivity">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:orientation="horizontal"
        android:padding="5dp">

        <Button
            android:id="@+id/miBand_show_addr"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="0.1"
            android:background="@drawable/miband2"
            android:gravity="bottom|center"
            android:text="Show Config"
            android:textAppearance="@style/TextAppearance.AppCompat.Widget.Button.
    Borderless.Colored"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="10sp"
            android:visibility="visible" />

        <EditText
            android:id="@+id/txtPhysicalAddress"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="0.1"
            android:editable="false"
            android:hint="Waiting for Connection"
            android:textSize="14sp"
            android:visibility="visible" />

        <LinearLayout
            android:layout_width="100dp"
            android:layout_height="match_parent"
            android:layout_weight="0.1"
            android:orientation="vertical">

            <TextView
                android:id="@+id/txtState2"
                android:layout_width="match_parent"
                android:layout_height="34dp"
                android:layout_weight="030"
                android:autoText="true"
                android:background="@drawable/bluetooth"
```

```
54                  android:paddingTop="24dp"
55                  android:textAlignment="center"
56                  android:textAppearance="@style/TextAppearance.AppCompat"
57                  android:textColor="@android:color/black"
58                  android:textIsSelectable="false"
59                  android:visibility="visible" />
60
61              <TextView
62                  android:id="@+id/txtState"
63                  android:layout_width="match_parent"
64                  android:layout_height="wrap_content"
65                  android:layout_weight="030"
66                  android:autoText="false"
67                  android:gravity="center_vertical|center_horizontal"
68                  android:text="Connecting"
69                  android:textAlignment="gravity"
70                  android:textAppearance="@style/TextAppearance.AppCompat"
71                  android:textColor="@color/colorPrimaryDark"
72                  android:textIsSelectable="false"
73                  android:visibility="visible" />
74          </LinearLayout>
75
76      </LinearLayout>
77
78      <LinearLayout
79          android:layout_width="match_parent"
80          android:layout_height="119dp"
81          android:gravity="center"
82          android:orientation="horizontal"
83          android:paddingBottom="5dp"
84          android:paddingTop="5dp">
85
86          <Button
87              android:id="@+id/btnStartConnecting"
88              style="@style/Widget.AppCompat.Button.Borderless"
89              android:layout_width="match_parent"
90              android:layout_height="121dp"
91              android:layout_weight="1"
92              android:background="@drawable/bluetooth_off"
93              android:gravity="bottom"
94              android:text="Reconnect"
95              android:textAlignment="center"
96              android:textColor="@color/colorPrimaryDark"
97              android:textSize="14sp"
98              android:textStyle="bold"
99              android:visibility="gone" />
100
101         <LinearLayout
102             android:id="@+id/heartLy"
103             android:layout_width="match_parent"
104             android:layout_height="match_parent"
105             android:gravity="center_vertical|center_horizontal"
106             android:orientation="vertical"
107             android:visibility="visible">
108
109             <Button
110                 android:id="@+id/btnGetHeartRate"
111                 android:layout_width="127dp"
112                 android:layout_height="match_parent"
113                 android:layout_weight="0.3"
114                 android:background="@drawable/heart3d"
```

```
115                 android:elevation="24dp"
116                 android:gravity="center_vertical|center_horizontal|center"
117                 android:text="Read Heart Rate"
118                 android:textAppearance="@style/TextAppearance.AppCompat.Display2"
119                 android:textSize="14sp"
120                 android:visibility="visible" />
121
122             <TextView
123                 android:id="@+id/txtBpm"
124                 android:layout_width="match_parent"
125                 android:layout_height="41dp"
126                 android:layout_weight="0.3"
127                 android:scrollHorizontally="false"
128                 android:text="bpm"
129                 android:textAlignment="center"
130                 android:textStyle="bold"
131                 android:typeface="monospace" />
132
133         </LinearLayout>
134
135         <LinearLayout
136             android:id="@+id/bpmLx"
137             android:layout_width="match_parent"
138             android:layout_height="120dp"
139             android:layout_weight="1"
140             android:baselineAligned="false"
141             android:orientation="vertical"
142             android:visibility="gone">
143
144             <TextView
145                 android:id="@+id/alarmSet"
146                 android:layout_width="match_parent"
147                 android:layout_height="93dp"
148                 android:layout_weight="1"
149                 android:text="Alarm Setting"
150                 android:textAlignment="center"
151                 android:textStyle="bold" />
152
153             <LinearLayout
154                 android:layout_width="match_parent"
155                 android:layout_height="122dp"
156                 android:layout_weight="1"
157                 android:orientation="horizontal">
158
159                 <TextView
160                     android:id="@+id/maxBpm"
161                     android:layout_width="47dp"
162                     android:layout_height="match_parent"
163                     android:layout_weight="1"
164                     android:gravity="center_vertical|center_horizontal"
165                     android:text="Max. bpm"
166                     android:textSize="12sp"
167                     android:textStyle="bold" />
168
169                 <EditText
170                     android:id="@+id/maxBpmAlarm"
171                     android:layout_width="54dp"
172                     android:layout_height="match_parent"
173                     android:layout_weight="1"
174                     android:editable="false"
175                     android:ems="10"
```

```
176                     android:gravity="center_vertical|center_horizontal"
177                     android:inputType="number"
178                     android:text="190" />
179
180                 <SeekBar
181                     android:id="@+id/setMaxbpmAlarm"
182                     android:layout_width="280dp"
183                     android:layout_height="wrap_content"
184                     android:layout_gravity="center_vertical|center_horizontal"
185                     android:layout_weight="1"
186                     android:max="200"
187                     android:progress="180" />
188
189             </LinearLayout>
190
191             <LinearLayout
192                 android:layout_width="match_parent"
193                 android:layout_height="match_parent"
194                 android:layout_weight="1"
195                 android:orientation="horizontal">
196
197                 <TextView
198                     android:id="@+id/minBpm"
199                     android:layout_width="36dp"
200                     android:layout_height="match_parent"
201                     android:layout_weight="1"
202                     android:gravity="center_vertical|center_horizontal"
203                     android:text="Min. bpm"
204                     android:textAlignment="gravity"
205                     android:textSize="12sp"
206                     android:textStyle="bold" />
207
208                 <EditText
209                     android:id="@+id/minBpmAlarm"
210                     android:layout_width="41dp"
211                     android:layout_height="match_parent"
212                     android:layout_weight="1"
213                     android:editable="false"
214                     android:ems="10"
215                     android:gravity="center_vertical|center_horizontal"
216                     android:inputType="number"
217                     android:text="40" />
218
219                 <SeekBar
220                     android:id="@+id/setMinbpmAlarm"
221                     android:layout_width="267dp"
222                     android:layout_height="wrap_content"
223                     android:layout_gravity="center_vertical|center_horizontal"
224                     android:layout_weight="1"
225                     android:max="200"
226                     android:progress="40" />
227             </LinearLayout>
228
229         </LinearLayout>
230
231
232     </LinearLayout>
233
234     <LinearLayout
235         android:layout_width="match_parent"
236         android:layout_height="wrap_content"
```

```
237         android:gravity="right"
238         android:orientation="horizontal">
239 
240         <LinearLayout
241             android:id="@+id/FBatLy"
242             android:layout_width="150dp"
243             android:layout_height="match_parent"
244             android:layout_weight="1"
245             android:gravity="center"
246             android:orientation="horizontal"
247             android:visibility="visible">
248 
249             <LinearLayout
250                 android:layout_width="190dp"
251                 android:layout_height="match_parent"
252                 android:layout_weight="1"
253                 android:orientation="horizontal"></LinearLayout>
254 
255             <LinearLayout
256                 android:id="@+id/findL"
257                 android:layout_width="176dp"
258                 android:layout_height="match_parent"
259                 android:layout_weight="1"
260                 android:gravity="right"
261                 android:visibility="visible">
262 
263                 <Button
264                     android:id="@+id/btnStartVibrate"
265                     android:layout_width="match_parent"
266                     android:layout_height="108dp"
267                     android:layout_weight="1"
268                     android:background="@drawable/find_iconjpg"
269                     android:baselineAligned="false"
270                     android:gravity="bottom|left|center_horizontal|fill_horizontal|center
    |start|end"
271                     android:text="Find / Start Vibrating"
272                     android:textAlignment="gravity"
273                     android:textSize="10sp"
274                     android:visibility="visible" />
275 
276             </LinearLayout>
277 
278             <LinearLayout
279                 android:id="@+id/batL"
280                 android:layout_width="182dp"
281                 android:layout_height="match_parent"
282                 android:layout_weight="1"
283                 android:gravity="left"
284                 android:orientation="vertical"
285                 android:visibility="visible">
286 
287                 <Button
288                     android:id="@+id/btnGetBatteryInfo"
289                     android:layout_width="match_parent"
290                     android:layout_height="102dp"
291                     android:layout_weight="1"
292                     android:background="@drawable/batery_icon"
293                     android:elevation="24dp"
294                     android:gravity="bottom|center"
295                     android:text="Check Battery"
296                     android:textSize="10sp"
```

```
297                     android:visibility="visible" />
298
299                 <TextView
300                     android:id="@+id/textBat"
301                     android:layout_width="match_parent"
302                     android:layout_height="wrap_content"
303                     android:layout_weight="1"
304                     android:text="Level %"
305                     android:textAlignment="center"
306                     android:textAllCaps="false"
307                     android:textSize="12sp"
308                     android:textStyle="bold"
309                     android:typeface="monospace" />
310
311             </LinearLayout>
312
313             <LinearLayout
314                 android:layout_width="177dp"
315                 android:layout_height="match_parent"
316                 android:layout_weight="1"
317                 android:orientation="horizontal"></LinearLayout>
318
319         </LinearLayout>
320
321         <LinearLayout
322             android:id="@+id/timerLx"
323             android:layout_width="match_parent"
324             android:layout_height="wrap_content"
325             android:layout_weight="1"
326             android:orientation="vertical"
327             android:visibility="gone">
328
329             <LinearLayout
330                 android:layout_width="match_parent"
331                 android:layout_height="match_parent"
332                 android:layout_weight="1"
333                 android:orientation="vertical">
334
335                 <TableLayout
336                     android:layout_width="match_parent"
337                     android:layout_height="wrap_content"
338                     android:layout_weight="1">
339
340                     <TableRow
341                         android:layout_width="match_parent"
342                         android:layout_height="match_parent"
343                         android:layout_weight="10">
344
345                         <TextView
346                             android:id="@+id/timerSet"
347                             android:layout_width="match_parent"
348                             android:layout_height="wrap_content"
349                             android:layout_weight="1"
350                             android:text="Timer Setting"
351                             android:textAlignment="center"
352                             android:textStyle="bold" />
353                     </TableRow>
354
355                     <TableRow
356                         android:layout_width="match_parent"
357                         android:layout_height="76dp"
```

```
358                         android:layout_weight="1">
359
360                         <LinearLayout
361                             android:layout_width="match_parent"
362                             android:layout_height="match_parent"
363                             android:layout_weight="1"
364                             android:orientation="horizontal">
365
366                             <TextView
367                                 android:id="@+id/timerMin"
368                                 android:layout_width="39dp"
369                                 android:layout_height="match_parent"
370                                 android:layout_weight="1"
371                                 android:gravity="center"
372                                 android:text="Mnts" />
373
374                             <EditText
375                                 android:id="@+id/readMin"
376                                 android:layout_width="39dp"
377                                 android:layout_height="match_parent"
378                                 android:layout_weight="1"
379                                 android:ems="10"
380                                 android:gravity="bottom|center_horizontal|center"
381                                 android:inputType="number"
382                                 android:text="1"
383                                 android:textAlignment="gravity"
384                                 android:textSize="12sp" />
385
386                             <SeekBar
387                                 android:id="@+id/setTimeerMin"
388                                 style="@style/Widget.AppCompat.SeekBar.Discrete"
389                                 android:layout_width="309dp"
390                                 android:layout_height="33dp"
391                                 android:layout_weight="40"
392                                 android:max="60"
393                                 android:progress="1" />
394
395                         </LinearLayout>
396                     </TableRow>
397
398                     <TableRow
399                         android:layout_width="match_parent"
400                         android:layout_height="wrap_content"
401                         android:layout_weight="1">
402
403                         <LinearLayout
404                             android:layout_width="match_parent"
405                             android:layout_height="match_parent"
406                             android:layout_weight="1"
407                             android:orientation="horizontal">
408
409                             <TextView
410                                 android:id="@+id/timerHour"
411                                 android:layout_width="47dp"
412                                 android:layout_height="match_parent"
413                                 android:layout_weight="1"
414                                 android:gravity="center"
415                                 android:text="Hours" />
416
417                             <EditText
418                                 android:id="@+id/readHour"
```

```
                                android:layout_width="39dp"
                                android:layout_height="wrap_content"
                                android:layout_weight="1"
                                android:ems="10"
                                android:gravity="bottom|center_horizontal"
                                android:inputType="textPersonName"
                                android:text="00"
                                android:textAlignment="gravity"
                                android:textSize="14sp" />

                            <SeekBar
                                android:id="@+id/setTimerHour"
                                style="@style/Widget.AppCompat.SeekBar.Discrete"
                                android:layout_width="279dp"
                                android:layout_height="wrap_content"
                                android:layout_gravity="center_vertical"
                                android:layout_weight="1"
                                android:max="24"
                                android:progress="0" />

                        </LinearLayout>
                    </TableRow>

                </TableLayout>

            </LinearLayout>

        </LinearLayout>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <LinearLayout
            android:id="@+id/ubiLx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:visibility="gone">

            <TextView
                android:id="@+id/ubiCfg"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="UbiDots Config"
                android:textAlignment="center" />

            <TableLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TableRow
                    android:layout_width="match_parent"
                    android:layout_height="match_parent">

                    <TextView
                        android:id="@+id/ubiid"
                        android:layout_width="wrap_content"
```

```
480                     android:layout_height="37dp"
481                     android:layout_weight="30"
482                     android:text="Ubidots ID" />
483
484                 <EditText
485                     android:id="@+id/ubiID"
486                     android:layout_width="281dp"
487                     android:layout_height="wrap_content"
488                     android:layout_weight="1"
489                     android:ems="10"
490                     android:inputType="textVisiblePassword"
491                     android:text="insert key and touch 'show config' to save"
492                     android:textSize="12sp" />
493             </TableRow>
494
495             <TableRow
496                 android:layout_width="match_parent"
497                 android:layout_height="match_parent">
498
499                 <TextView
500                     android:id="@+id/heartKey"
501                     android:layout_width="wrap_content"
502                     android:layout_height="46dp"
503                     android:layout_weight="30"
504                     android:text="Heart Key" />
505
506                 <EditText
507                     android:id="@+id/ubiHeartKey"
508                     android:layout_width="wrap_content"
509                     android:layout_height="wrap_content"
510                     android:ems="10"
511                     android:inputType="none|textVisiblePassword"
512                     android:text="insert key and touch 'show config' to save" />
513             </TableRow>
514
515             <TableRow
516                 android:layout_width="match_parent"
517                 android:layout_height="match_parent">
518
519                 <TextView
520                     android:id="@+id/batKey"
521                     android:layout_width="wrap_content"
522                     android:layout_height="42dp"
523                     android:layout_weight="30"
524                     android:text="Battery Key" />
525
526                 <EditText
527                     android:id="@+id/ubiBatKey"
528                     android:layout_width="wrap_content"
529                     android:layout_height="56dp"
530                     android:ems="10"
531                     android:inputType="textVisiblePassword"
532                     android:text="insert key and touch 'show config' to save" />
533
534             </TableRow>
535
536             <TableRow
537                 android:layout_width="match_parent"
538                 android:layout_height="match_parent">
539
540                 <TextView
```

```
                        android:id="@+id/telef"
                        android:layout_width="wrap_content"
                        android:layout_height="32dp"
                        android:layout_weight="30"
                        android:ems="10"
                        android:inputType="textPersonName"
                        android:text="S.O.S NR."
                        android:textSize="12sp"
                        android:textStyle="bold" />

                    <EditText
                        android:id="@+id/sosTelef"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:ems="10"
                        android:inputType="phone"
                        android:text="emergency phone number" />
                </TableRow>

            </TableLayout>
        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

6

Criar e configurar uma conta Ubidots

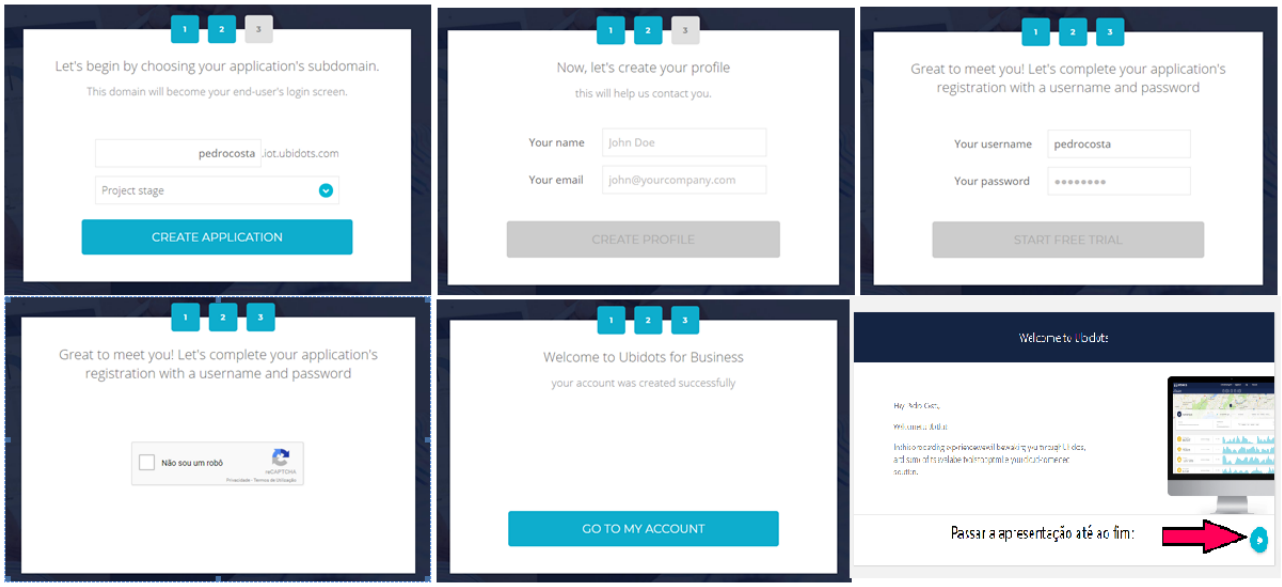

Figura 6.1: Criar conta Ubidots

.

Figura 6.2: Apagar o painel Demo e criar painel para histórico

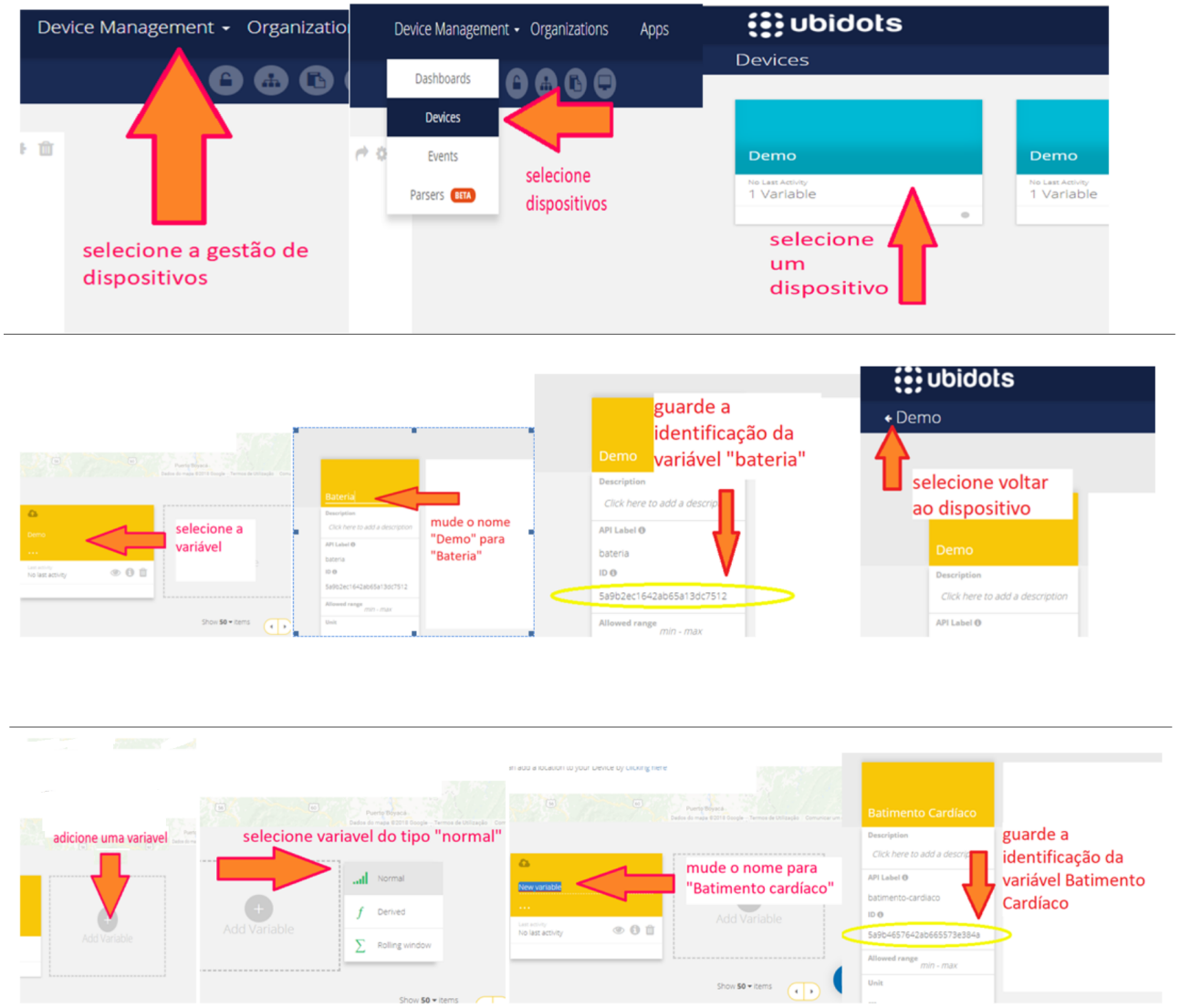

Figura 6.3: Criar Duas Variáveis

Figura 6.4: Criar Histórico

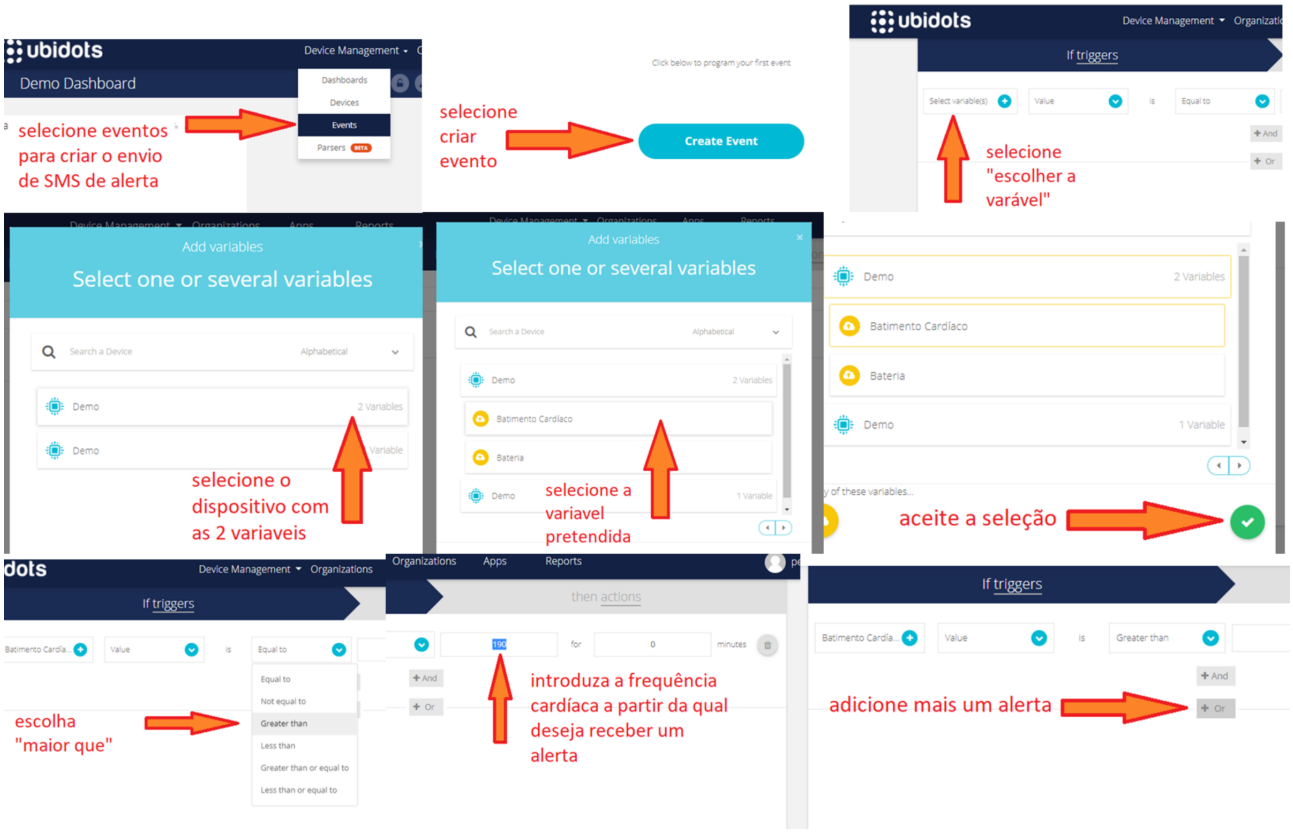

Figura 6.5: Criar Condições

Figura 6.6: Criar mensagem de alerta

Bibliografia

[ACA+14] G Appelboom, E Camacho, M E Abraham, S S Bruce, E L Dumont, B E Zacharia, R D'Amico, J Slomian, J Y Reginster, O Bruyère, and et al. Smart wearable body sensors for patient self-assessment and monitoring., Aug 2014.

[Bes16] R.E. et all Best. Tutorial on dynamic analysis of the costas loop, Sep 2016.

[BH11] Danny Briere and Pat Hurley. *Wireless home networking for dummies*. Wiley Pub., 2011.

[CL16] Guanqun Cao and Jiangbo Liu. An iot application: Health care system with android devices. In *International Conference on Computational Science and Its Applications*, pages 563–571. Springer, 2016.

[Cor17] David Corman. Cyber-physical systems (cps), April 2017.

[daC13] Henderson Byron daCosta, Francis. *Rethinking the Internet of Things*. Apress, 2013.

[DeL17] Jean-Jacques DeLisle. What's the difference between ieee 802.11af and 802.11ah?, Mar 2017.

[DH] G Demiris and B K Hensel. Technologies for an aging society: a systematic review of "smart home"applications.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES - The advanced encryption standard*. Springer, 2002.

[Fil00] E.T. de Carvalho Filho. Arritmias cardíacas no idoso, Apr 2000.

[FK82] Jerome L. Fleg and Harold L. Kennedy. Cardiac arrhythmias in a healthy elderly population. *Chest*, 81(3):302–307, 1982.

[GBJA16] Farah Gipsa, Varghese Benitta, Jose G Jezna, and Abraham AlbyMol. A modern health care system using iot and android. *International Journal on Computer Science and Engineering (IJCSE)*, 8(4), Apr 2016.

[Gei10] James T. Geier. *Designing and Deploying 802.11n Wireless Networks*. Cisco Press, 2010.

[Gie96] Filip M. Gieszczykiewicz. A painless guide to crc error detection algorithms index v3.00, 9 1996.

[Gis08] Drew Gislason. *Zigbee wireless networking*. Elsevier, 2008.

[Gre16] John Greenough. How the 'internet of things' will impact consumers, businesses, and governments in 2016 and beyond, Jul 2016.

[Hay14] Simon S. Haykin. *Digital communication systems*. Wiley, 2014.

[Hic17] Shurvinton Bill Beard Gemma Hicklin, John. *Internet of Things For Dummies*. John Wiley & Sons, Inc., 111 River St., Hoboken, NJ 07030-5774, 2017. Qorvo Special Edition.

[HP04] HP. Bluetooth wireless technology basics, 2004.

[IBM00] IBM. Bluetooth technology: Beginner's guide, 2000.

[JKZ+10] Sony Jacob, Naga V. A. Kommuri, Sandip K. Zalawadiya, Marc D. Meissner, and Randy A. Lieberman. Sensing performance of a new wireless implantable loop recorder: A 12-month follow up study. *Pacing and Clinical Electrophysiology*, 33(7):834–840, 2010.

[JSHG13] Magnus Thorsten Jensen, Poul Suadicani, Hans Ole Hein, and Finn Gyntelberg. Elevated resting heart rate, physical fitness and all-cause mortality: a 16-year follow-up in the copenhagen male study. *Heart*, 99(12):882–887, 2013.

[KMA+17] Ranjeet Kumar, Rajat Maheshwari, Amit Aggarwal, M. Shanmugasundaram, and Sundar S. Iot based health monitoring system using android app. *ARPN Journal of Engineering and Applied Sciences*, 12(19), Oct 2017.

[KMH+10] Hyun Gu Kang, Diane F. Mahoney, Helen Hoenig, Victor A. Hirth, Paolo Bonato, Ihab Hajjar, Lewis A. Lipsitz, for the Center for Integration of Medicine, and Innovative Technology Working Group on Advanced Approaches to Physiologic Monitoring for the Aged. In situ monitoring of health in older adults: Technologies and issues. *Journal of the American Geriatrics Society*, 58(8):1579–1586, 2010.

[Koe17] Harvey Koeppel. Iot evolution begs for new thinking, new value systems, January 2017.

[lib16] libelium. Top 50 iot sensor applications ranking @ONLINE, December 2016.

[Mis10] Sudip Misra. *Guide to wireless mesh networks*. Springer, 2010.

[NM17] Alex Nasri and Abdellatif Mtibaa. Smart mobile healthcare system based on wbsn and 5g. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 8(10), 2017.

[Pal02] MD Paolo Palatini. Predictive value of clinic and ambulatory heart rate for mortality in elderly subjects with systolic hypertension, Nov 2002.

[PD12] Larry L. Peterson and Bruce S. Davie. *Computer networks: a systems approach*. Morgan Kaufmann, 2012.

[PK14] Sushama Pawar and P. W. Kulkarni. Home based health monitoting system using android smartphone. *International Journal of Electrical, Electronics and Data Communication*, 2(2), Feb 2014.

[Pol13] Cigna Medical Coverage Policy. Cardiac event monitors, coverage policy number 0085, 2013.

[PSS+17] John Padgette, Karen Scarfone, Rhonda Smithbey, Lily Chen, John Bahr, Mayank Batra, and Marcel Holtmann. Guide to bluetooth security - nist page, may 2017.

[PWKG17] Prashant Patil, Rohan Waichal, Utkatsha Kumbhar, and Vaidehi Gadkari. Patient health monitoring system using iot. *International Research Journal of Engineering and Technology (IRJET)*, 4(1), Jan 2017.

[RW16] Margaret Rouse and Ivy Wigmore. What is internet of things (iot)? @ONLINE, June 2016.

[sta] statista. Global internet of things market size 2009-2019.

[TVM+94] H Tsuji, F J Venditti, E S Manders, J C Evans, M G Larson, C L Feldman, and D Levy. Reduced heart rate variability and mortality risk in an elderly cohort. the framingham heart study., Aug 1994.

[VS17] K. Vasant and J. Sbert. Creating solutions for health throughtecnology innovation @ONLINE, July 2017.

[www15] www.cdc.gov. Target heart rate and estimated maximum heart rate, Aug 2015.

Índice