UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

A Internet das Coisas

Caso de estudo: Esfigmomanómetro Digital para Ambulatório

David José Ferro Martins

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores
(2º ciclo de estudos)

Orientador: Prof. Doutor Bruno Jorge Ferreira Ribeiro

Covilhã, outubro de 2015

Agradecimentos

Aos meus pais e padrinhos que sempre alimentaram a minha sede pela engenharia e que tornaram possível concretizar os meus estudos em Engenharia Eletrotécnica. A eles devo tudo.

À minha namorada, Milena Rodrigues, pelo apoio incondicional, pela sua capacidade de diálogo, por nunca desistir de mim e ainda pela sua enorme vontade em me ajudar todos os dias, tornando-os mais brilhantes com a sua felicidade contagiante.

Ao meu caríssimo orientador, professor Bruno Ribeiro, pela forma positiva e motivante com que aceitou embarcar comigo nesta aventura. Junto com os professores António Espírito Santo e Pedro Dinis pela sua boa disposição constante e incentivo.

Ao grupo dos *sobreviventes de luta*, Pedro Torres e Pedro Pinto, pela companhia e companheirismo mostrados desde o meu primeiro dia neste curso, revelando-se uma vez mais nesta reta final.

Aos antigos colegas de trabalho, Cláudia Tonelo e Vítor Goulão, pelos bons momentos criados em tempos e dias difíceis pelos quais passámos.

Ao camarada das *engenhocas*, António Sérgio Sena, pelo seu apoio, incentivo e troca de conhecimento, que muito contribuiu para o meu crescer como profissional.

Por último, mas não menos importantes, a todas as amizades construídas que, cada uma à sua maneira, contribuíram para o meu crescer pessoal. O apoio, a camaradagem e os bons momentos vividos, foram muito importantes e inesquecíveis para mim. Escrever sobre tudo isso, daria uma nova dissertação.

A todos vocês o meu mais sincero obrigado.

Resumo

Esta dissertação incide sobre a construção de um protótipo funcional que se propõe recolher registos de pressão arterial e batimentos cardíacos, e publicar esses valores numa plataforma *online* dedicada a *Internet of Things*. O envio desses resultados é feito diretamente via *Wi-Fi*, sem recurso a *gateways*.

Foi realizado um estudo pormenorizado sobre o funcionamento do esfigmomanómetro automático com o objetivo de compreender a sua constituição e forma de operar. Este estudo mostrou-se relevante para conseguir uma modificação bem-sucedida do esfigmomanómetro Sanitas SBC 25/1, utilizado neste trabalho. De modo a conseguir acesso aos valores resultantes das medições efetuadas, foi utilizado um analisador lógico para capturar e analisar os dados trocados no barramento da EEPROM presente no circuito do aparelho Sanitas SBC 25/1.

Com estas informações foi possível construir um algoritmo de *sniffing* capaz de capturar a *frame* certa (aquela que contém os valores resultantes da medição). Esse algoritmo foi implementado no microcontrolador PIC18LF22K50, da Microchip, que fica responsável não só pela recolha dos dados como pelo envio dos mesmos. O envio é efetuado via *Wi-Fi* com recurso ao recente módulo ESP-01 baseado no microcontrolador ESP8266 da Espressif.

A plataforma *online* utilizada, dedicada a *Internet of Things,* foi o ThingSpeak. Nela é criado um canal com três campos (*fields*) e é para onde fluem os dados enviados pelo esfigmomanómetro e resultantes das medições da pressão sistólica, diastólica e os batimentos cardíacos.

Para completar todo o sistema, foi projetado e construído uma placa de circuito impresso capaz de conter todas os componentes necessários ao funcionamento pleno do protótipo.

O trabalho cumpriu os objetivos iniciais com a construção do protótipo totalmente funcional que visa proporcionar uma recolha de registos biométricos completa e independente.

Palavras-chave

Pressão arterial, esfigmomanómetro, medidor de pressão arterial, ambulatório, Internet das Coisas, IoT, Wi-Fi, módulo ESP-01, ThingSpeak, PIC18LF25K22.

Abstract

This dissertation focuses on the construction of a functional prototype that aims to collect blood pressure and heartbeat records, and publish these values on an online platform dedicated to Internet of Things. The data transmission of these results is done directly via Wi-Fi, without using gateways.

A detailed study was conducted on the functioning of an automatic sphygmomanometer in order to understand its constitution and way of operating. This study proved relevant to achieve a successful modification of the sphygmomanometer Sanitas SBC 25/1, used in this project. In order to gain access to the resulting values of the measurement performed, a logic analyzer was used to capture and analyze data exchange in the EEPROM bus which is in the Sanitas SBC 25/1 circuit.

With this information it was possible to build a sniffing algorithm capable of capturing the right frame (the one that contains the values from the measurement). This algorithm has been implemented in the microcontroller PIC18LF22K50 from Microchip, which is responsible not only for collecting the data but also to provide them. The data transmission is made via Wi-Fi using the recent module ESp-01 based on the microcontroller ESP8266 from Espressif.

The online platform used, dedicated to Internet of Things was ThingSpeak. There was created a channel with three fields and this is where the data is transmitted by the sphygmomanometer, the result of the measurement of systolic and diastolic blood pressure and heartbeats.

To complete the whole system, there was designed and constructed a printed circuit board capable to contain all the necessary components to the full operating prototype.

The project met all the initial goals with the construction of a totally functional prototype which aims to provide a complete and independent gathering of biometric records.

Keywords

Blood pressure, sphygmomanometer, blood pressure monitor, ambulatory, Internet of Things, IoT, Wi-Fi, ESP-01 module, ThingSpeak, PIC18LF25K22.

Índice

Lista de Figuras

Nomenclatura

mmHg	Milímetros de mercúrio
bpm	Batimentos por Minuto
dBm	Decibel Miliwatt
bps	Bits Por Segundo
RF	Rádio Frequência
IoT	*Internet of Things*
M2M	*Machine-to-Machine*
I&D	Investigação e Desenvolvimento
MAP	*Mean Arterial Pressure*
I^2C	*Inter-integrated Circuit*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
USB	*Universal Serial Bus*
ADC	*Analog-to-Digital Converter*
SMD	*Surface Mounted Device*
PCB	*Printed Circuit Board*
PLA	Poliácido Láctico
HF	*High Frequency*
TCP	*Transmition Control Protocol*
IP	*Internet Protocol*
GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Service*
SDK	*Software Development Kit*
ACK	*Acknowledge*
NACK	*No Acknowledge*
MCU	*Micro Controller Unit*
SDA	*Serial Data*
SCL	*Serial Clock*
MIPS	*Millions of Instructions per Second*
MSSP	*Mechanical Systems and Signal Processing*
PLL	*Phase Locked Loop*
USART	*Universal Synchronous Asynchronous Reciever Transmitter*
CCP	*Capture/Compare/PWM*
CTMU	*Charge Time Measurement Unit*
SOIC	*Small Outline Integrated Circuit*

1. Introdução

1.1. Enquadramento e motivação

Atualmente as pessoas estão cada vez mais sensibilizadas para a vigilância do seu estado de saúde. A consciencialização da importância de manter um estilo de vida saudável, levou parte da população a envergar pela prática regular de exercício físico, por dietas cuidadas e pela monitorização de vários parâmetros biométricos[1]. Foi o avanço da tecnologia e da engenharia, principalmente devido à miniaturização e avanços nos sistemas embebidos, que permitiu à população o acesso a instrumentação para medição dos parâmetros biomédicos sem recorrer aos serviços de saúde, podendo fazê-lo em qualquer lugar e a qualquer hora.

Os dados recolhidos, quer durante o *fitness,* quer em avaliações de rotina, contribuem para o controlo preventivo e como método de deteção precoce de doenças. Para que esse controlo preventivo possa ser realizado, os dispositivos médicos de medição de parâmetros biométricos foram-se também adaptando às necessidades dos utilizadores, de modo a satisfazê-las. O facto dos dispositivos mais recentes serem capazes de, fazer as medições com maior exatidão e de preservarem de forma automática os registos resultantes permite um maior controlo da progressão do estado da saúde dos indivíduos[2].

Registos contínuos/periódicos são de extrema utilidade. Quando partilhados com um médico, permitem uma perceção mais alargada da evolução clínica do doente. Além das vantagens intrínsecas da partilha desses dados com o médico, pode ainda assinalar-se a mais-valia de os cuidadores de saúde poderem ponderar e considerar os mesmos na elaboração de alterações da dieta e na medicação do doente.

Um dos parâmetros mais importantes a ter em consideração é a pressão arterial. A pressão arterial é a *força* que o sangue em circulação exerce nas paredes das artérias. A medição regular deste parâmetro permite detetar antecipadamente doenças como a hipertensão arterial. Uma única medição acima do limiar "hipertensão", pressão sistólica maior que 140mmHg e diastólica maior que 90mmHg, apenas indica que a pressão está alta no momento da medição. Para que possa ser diagnosticada hipertensão é necessário repetir e realizar a medição em diferentes circunstâncias e ao longo de um dado período[3]. Esta doença afeta 25% da população adulta mundial e em Portugal atinge 42.1%, dos quais apenas 11.2% dos indivíduos têm a sua pressão arterial controlada[3]. O consumo de alimentos ricos em sal, gordura, açúcar e calorias, associado a uma diminuição da atividade física, consumo de tabaco e álcool, são a chave para

um tão grande número de pessoas com hipertensão, doença que quando relacionada com outras, pode provocar graves problemas de saúde e até mesmo a morte[4].

Existem fenómenos de hipertensão que são apenas diagnosticados corretamente, quando completados com registos regulares, realizados fora do ambiente hospitalar. Os três principais são a "hipertensão da bata branca", "hipertensão mascarada" e a hipertensão noturna ou noite sem redução de pressão arterial[5]. Mesmo após ser realizado o diagnóstico, surge a necessidade de controlo regular durante o tratamento da doença, para que o médico possa avaliar se está ou não a surtir melhorias[6].

Um equipamento de medição automático traz alguns benefícios na perspetiva de que o doente não necessita iniciar manualmente uma nova medição a cada intervalo de tempo, nem fazer o registo manual dos valores medidos. Se a isto se juntar a capacidade do equipamento se ligar por Internet[7] a um servidor para guardar os resultados, estão criadas condições para abrir as portas da monitorização remota da condição de saúde do doente. Essa monitorização pode ser realizada quer em ambiente hospitalar, quer em suas casas, permitindo a um médico, em qualquer lugar do mundo e em qualquer momento, ter acesso aos registos em tempo real.

Para que isso aconteça e se torne uma realidade, a *Internet of Things*, IoT, ou em português, Internet das Coisas, veio dar um importante contributo. Este conceito foi apresentado pela primeira vez em 1999 através do empresário britânico Kevin Ashton, que viu nele capacidades que vão muito além das comunicações M2M, *Machine-to-Machine*, cobrindo uma variedade de protocolos, domínios e aplicações. O conceito baseia-se na existência de uma rede de objetos físicos (coisas), interligadas através de redes informáticas. As redes possibilitam que as coisas possam interagir e/ou cooperar em qualquer momento, em qualquer lugar, com outros objetos, ou com qualquer pessoa[8].

1.2. Objetivos

Numa tentativa de fazer face aos desafios previamente relatados, apresenta-se uma proposta de esfigmomanómetro capaz de efetuar a medição das pressões sistólica e diastólica, assim como o batimento cardíaco. Para mais, e numa perspetiva de IoT, os resultados das medições são colocados numa plataforma online dedicada, possibilitando ao médico realizar melhores diagnósticos.

Em seguida são indicadas as principais características e consequentes vantagens do esfigmomanómetro:

- Está baseado num equipamento comercial com validação clínica, que com a devida autorização[1], foi possível ter acesso aos resultados das medições a fim de obter em *software* os valores resultantes da medição realizada.
- Utiliza rede *Wi-Fi*, não necessitando de *smartphones* ou de outros dispositivos intermédios de *gateway*. Este é um ponto diferenciador deste trabalho e com vantagens quando comparado a equipamentos semelhantes, pois tem a capacidade de realizar *all-in-one*.
- Baixo consumo energético obtido através de um constante cuidado e planeamento de lógica *low power*.
- Ligação a uma plataforma *online*, dedicada a objetos ligados por IoT, para onde são enviados os valores recolhidos e que fará gestão dos mesmos.

O último objetivo a atingir e possivelmente o mais importante, pois sem ele os anteriores não fariam sentido, é o planeamento, projeto e construção de um protótipo totalmente funcional que satisfaça todos os requisitos já apresentados. A concretização deste protótipo, assim como o seu funcionamento, é o culminar de todo o trabalho desenvolvido.

1.3. Estrutura da tese

A presente dissertação apresenta-se estruturada em oito capítulos, que se encontram organizados, o tanto quanto possível, por ordem cronológica do trabalho desenvolvido.

No capítulo 2 apresenta-se uma introdução de conceitos médicos relativos à pressão arterial e ao esfigmomanómetro, assim como os novos desafios desta temática.

No capitulo 3 procede-se ao estudo do medidor de pressão arterial, dando enfoque às suas características, à análise dos dados transferidos no barramento da EEPROM, quer em leitura de memória quer ao longo de uma medição completa, e à construção do algoritmo utilizado para recolher os dados do barramento.

No capítulo 4 é descrita a *interface* de comunicação *wireless* relativamente ao módulo *Wi-Fi* e à ligação à IoT. Trata também a comunicação com o módulo ESP-01, as características da plataforma *online* e por fim os processos de configuração, comandos AT e colocação de dados *online*.

[1] Consultar anexo 8.3.

No capítulo 5 descreve-se o protótipo, subdividindo-o na descrição do circuito e do protótipo em si.

O capítulo 6 conclui o trabalho, mas não sem antes proceder a uma verificação dos consumos do protótipo dará uma melhor perceção da viabilidade de todo o projeto.

2. Pressão Arterial – Tecnologias e Conceitos

2.1. Introdução clínica

O sistema circulatório é constituído pelo coração, vasos sanguíneos e sangue. O coração é oco, tem quatro cavidades, é constituído por tecido muscular e é do tamanho de um punho fechado. Este órgão com cerca de 300 gramas é a bomba essencial para a circulação do sangue em todos os vasos sanguíneos do organismo, contraindo-se cerca de 42 milhões de vezes por ano e bombeando mais de 2,5 milhões de litros de sangue[9].

A importância da circulação do sangue deve-se às funções vitais que desempenha, desde transporte, imunidade, regulação e coagulação, todas elas operam de forma a manter a homeostasia[9]. Sendo o coração a bomba responsável pela circulação do sangue, os vasos sanguíneos são os canais que o transportam a todos os tecidos do corpo e o trazem de volta ao coração. Além disso, os vasos sanguíneos interferem na regulação da pressão arterial e ajudam a dirigir o sangue para os tecidos onde a atividade é maior[10]. No entanto, a circulação do sangue para todas as áreas do corpo depende da manutenção de uma pressão adequada das artérias, só assim o controlo local do fluxo sanguíneo nos tecidos é consentâneo com as necessidades metabólicas. Se a pressão arterial for demasiado baixa pode o sangue não chegar a todos os tecidos, se for demasiado alta pode causar lesões nos vasos sanguíneos e coração[9][10].

A pressão sanguínea não é mais do que a força que o sangue exerce nas paredes internas dos vasos sanguíneos que atravessa. É nas artérias e arteríolas, sobretudo na artéria aorta e outras grandes artérias que esta pressão exercida é maior. À medida que o sangue que saiu do coração, após a contração do ventrículo esquerdo, se afasta do mesmo, menor é a pressão que ele exerce nas paredes dos vasos sanguíneos, sobretudo em pequenas e grandes veias, como a veia cava[9][10][11].

Tratando-se de um dos sinais vitais à vida humana e animal, juntamente com a frequência cardíaca e respiratória e temperatura[9], tornou-se essencial encontrar uma forma de medir quantitativamente a pressão arterial. Uma das técnicas pioneiras foi a utilização de um cateter inserido num vaso sanguíneo ligado a um manómetro. A evolução levou ao desenvolvimento de um método menos invasivo, e é este método que é utilizado na grande maioria dos casos – o esfigmomanómetro[10]. Este aparelho é constituído por uma braçadeira que é colocada no

braço superior coincidindo com a artéria braquial sendo a mesma insuflada de ar até ao colapso da artéria. Utiliza-se a braçadeira neste local pois é o mais próximo do coração e onde se consegue facilmente proceder à constrição da artéria, neste caso é feito através da compressão do vaso contra o úmero[9]. A pressão arterial é expressa por dois valores, a pressão arterial sistólica e diastólica. Esta distinção é realizada pelo esfigmomanómetro pois regista a pressão, em milímetros de mercúrio, em dois momentos diferentes do ciclo cardíaco, são eles a contração auricular – sístole, quando o sangue é ejetado para fora do coração em direção à artéria aorta, e o relaxamento ventricular – diástole, que ocorre durante a reentrada do sangue nos ventrículos[10].

A pressão arterial considerada normal para um adulto é de 120/80 mmHg[11].

Globalmente, cerca de 22% dos adultos com idade superior a 18 anos tinham a sua pressão arterial elevada em 2014 e Portugal acompanhou esta tendência[12].

2.2. História do esfigmomanómetro

"A capacidade de medir a pressão sanguínea levou cerca de duzentos anos a desenvolver"[13].

Usando um cavalo como cobaia, em 1733, Stephen Hales demonstrou que a pressão sanguínea, provocada pela contração do coração, podia ser medida através da deslocação do sangue. Para isso ele usou um método invasivo, inserindo um tubo de borracha numa artéria do cavalo por onde o sangue fluiria até uma proveta de vidro para ser medida.

Dado ser uma técnica tão agressiva, previa-se pouca aplicação para seres humanos e foi descartada a possibilidade de medir a pressão arterial sem provocar graves riscos de saúde nas pessoas que se sujeitassem a este método[13].

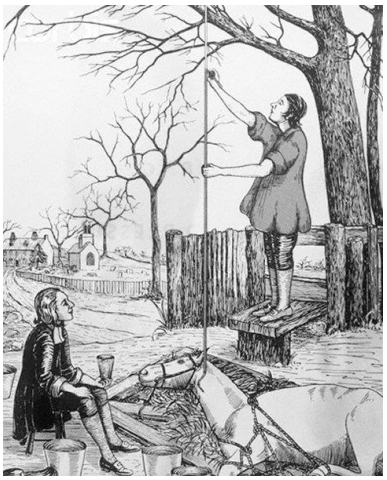


Figura 1 - Stephen Hales medindo a pressão arterial num cavalo, adaptado de [14]

Somente em 1828, passado cerca de um século desde as experiências de Stephen Hales, houve um avanço na medição da pressão sanguínea com a introdução do manómetro de mercúrio de Jean Leonard Marie Poiseuille como ferramenta chave para a medição da pressão sanguínea.

Nas experiências de Poiseuille, ao contrário dos manómetros inventados no século XVII em forma de U, ele usou um tubo oco com o interior retrátil. Uma extremidade ligava à artéria e a outra ao manómetro de mercúrio, fazendo com que a pressão arterial pudesse ser medida tomando por referência a quantidade de mercúrio deslocado.

A próxima inovação chegaria cerca de duas décadas mais tarde pela mão de Carl Ludwig com a invenção do quimógrafo (do grego *Kyma* = onda)[15]. Este equipamento produzia o registo gráfico de variações de ondulação numa "caneta", aparo, móvel que riscava um cilindro rotativo de velocidade constante. Esta ondulação era provocada pelo movimento do mercúrio dentro do manómetro, a cada pulsação o mercúrio movia-se no manómetro e por sua vez a caneta também se movia, desenhando o perfil temporal da pressão no cilindro [13].

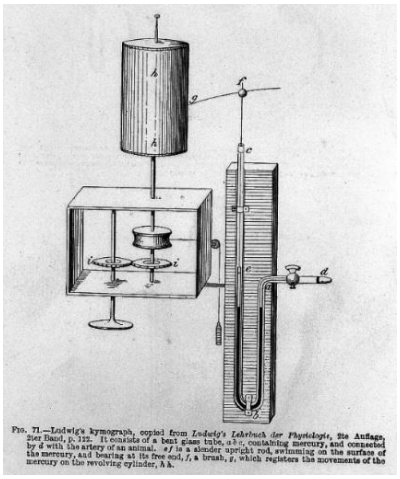


Figura 2 - Quimógrafo de Carl Ludwig, adaptado de [16]

Em 1855 novas técnicas não invasivas começaram a ser estudadas, tendo como base a pressão exercida externamente sobre a artéria até parar por completo a circulação de sangue na mesma. Cinco anos mais tarde, Etienne Jules Marey, identificou o que viria a ser a pressão sistólica. Aumentou a pressão, com recurso a uma bolsa de água em redor do braço, até que deixou de ter circulação sanguínea.

No ano de 1881 Samuel Siegfried Karl Ritter von Basch melhorou este método. Ele colocou um saco de borracha em torno do manómetro e encheu-o com água. Com o aumento de pressão o mercúrio deslocava-se dentro do manómetro, permitindo medir a pressão exercida. O saco era colocado no pulso e a pressão da água no saco era aumentada até que a pulsação deixasse de ser registada. Neste ponto a pressão apresentada pelo manómetro correspondia à pressão sistólica. A este aparelho de medição deu-se o nome de esfigmomanómetro[13].

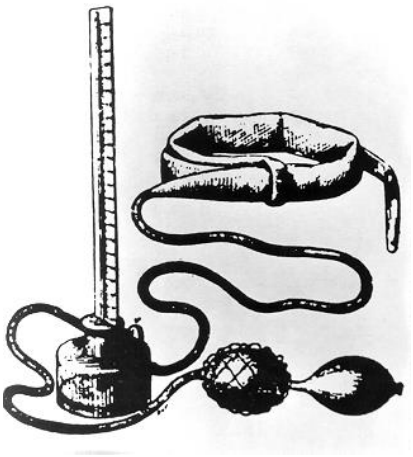


Figura 3 - Esfigmomanómetro primordial, adaptado de [17]

Apenas alguns anos mais tarde, em 1889, Pierre Potain substituiu a água e passou a usar ar para aumentar a pressão no saco.

Scipione Riva-Rocci iniciou, por volta de 1896, o uso de uma braçadeira insuflável em torno do braço que era cheia recorrendo a uma bomba manual ligada por um tubo. A pressão era aumentada até que o pulso não fosse mais sentido. Neste ponto a pressão é aliviada até voltar a sentir o pulso, ou seja, a pressão correspondente à pressão sistólica.

No início do século XX, era possível medir a pressão sistólica, mas a pressão diastólica continuava uma incógnita. O método para medir a pressão diastólica chegou apenas em 1905, pela mão do cirurgião russo Nikolai Korotkoff[18]. Ele identificou alguns sons (factualmente conhecidos por Sons de Korotkoff[19]) produzidos pelo sangue nas artérias e associados a mudanças de pressão. Para efetuar a leitura da pressão num esfigmomanómetro, utiliza-se um estetoscópio para identificar os sons de Korotkoff e poder medir as pressões sistólica e diastólica. Quando a braçadeira enche e bloqueia a circulação do sangue, nenhum som é ouvido. O primeiro som ouvido surge quando se atinge a pressão sistólica e algum sangue consegue passar o bloqueio provocado pela braçadeira causando ruído devido à turbulência da sua passagem. À medida que se continua a aliviar a pressão, os sons ouvidos ficam cada vez mais diminutos e por fim desaparecem, quando o sangue começar a fluir normalmente. A pressão medida aquando do momento em que se deixa de ouvir os sons de Korotkoff corresponde à pressão diastólica.

Em 1901, o neurocirurgião Harvey Cushing modernizou e popularizou junto da comunidade médica aquilo que se viria a tornar num dos instrumentos médicos mais utilizados, o medidor de pressão arterial[13][20].

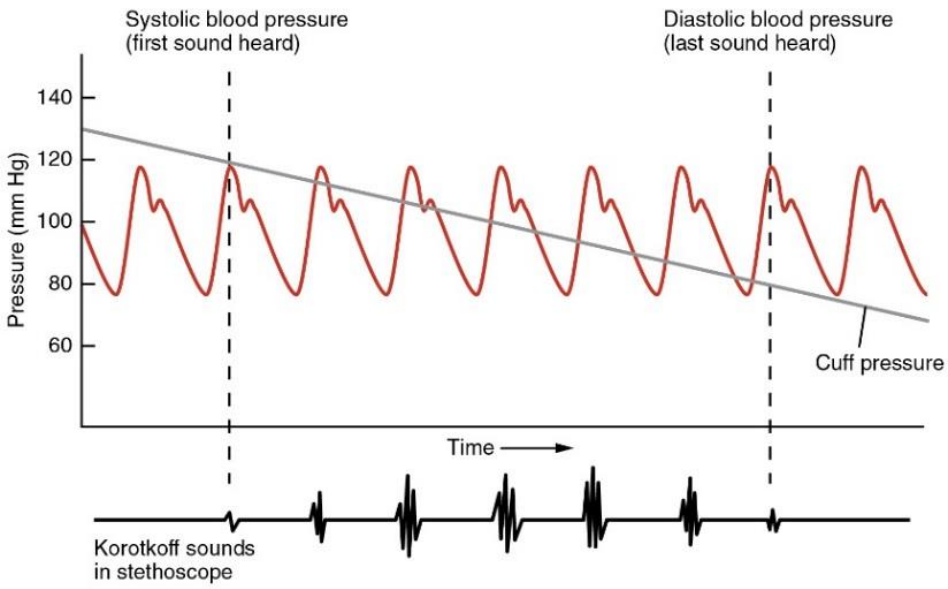


Figura 4 - Sons de Korotkoff, adaptado de [21]

Com o avanço tecnológico e da engenharia em si, pelas décadas de 60 e 70 criam-se os primeiros medidores eletrónicos de pressão arterial. A *Riester* afirma que em 1964 criou o seu "primeiro medidor de pressão arterial digital"[22], enquanto a *Omron* diz ter começado a investigação nesse mesmo ano[23], nas suas instalações de I&D e em 1978 terá criado o seu "primeiro medidor de pressão arterial de uso doméstico", o HEM-77.

Figura 5 - Medidor de pressão arterial Omron HEM-77, adaptado de [24]

No entanto o primeiro medidor digital totalmente automático, da Omrom (marca que possui um registo histórico acessível que ilustra bem o que todas as outras marcas tentavam fazer na mesma época[23]) em 1981, o HEM-88. Este equipamento enchia a braçadeira, esvaziava-a e fazia a medição, tudo de forma automática e sem necessidade de interação humana.

Figura 6 - Omron HEM-88, adaptado de [25]

Até aos dias de hoje, muitos formatos de medidores e técnicas foram adaptados, mas não se progrediu significativamente deste método. Com o avanço tecnológico toda a instrumentação foi ficando mais miniaturizada, fácil de trabalhar e por consequência também mais barata, e o uso do medidor de pressão arterial manual caiu em desuso.

2.3. Esfigmomanómetros automáticos

É especialmente relevante estudar o funcionamento do esfigmomanómetro automático visto que, como já foi referido anteriormente, parte deste trabalho passará pela realização da análise de um equipamento deste género. Para que se obtenha sucesso nessa parte é importante conhecer a forma como ele opera e como é constituído.

Existem 2 tipos de mecanismos de medição de pressão arterial: auscultatório e oscilométrico[26]. Nos seguintes parágrafos serão descritos estes dois mecanismos.

O método mais tradicional e usado pelos aparelhos de medição manuais é o auscultatório[27]. Este método utiliza o som produzido pela passagem do sangue nas artérias como referência. Quando a braçadeira está a exercer pressão a passagem do sangue encontra-se obstruída, mas quando se começa a aliviar essa pressão, pequenas quantidades de sangue começam a fluir aos solavancos produzindo um som ouvido com auxílio de um estetoscópio. À medida que se continua a aliviar a pressão, o sangue vai tendo mais facilidade em passar e os sons começam a ficar mais diminutos até que, quando deixa de haver um mínimo de pressão (diastólica), deixa de ser produzido som na passagem de sangue.

Embora alguns medidores (esfigmomanómetros) automáticos utilizem esta técnica, com recurso a microfone, não é de todo a metodologia mais largamente adotada. O mecanismo mais utilizado nos medidores de hoje em dia é o oscilométrico[28]. É um método mais indireto e os

equipamentos são empiricamente calibrados através do método auscultatório. Estes instrumentos, com este mecanismo de medição, não têm nem um estetoscópio nem um microfone e são de fácil utilização pois não é necessário um posicionamento preciso da braçadeira sobre a artéria. O seu funcionamento tem como base o princípio de que quando a artéria abre durante um ciclo de pressão, uma pequena perturbação é sobre imposta à pressão existente na braçadeira. Ou seja, a circunferência do braço é forçada a alargar infinitesimamente, causando variações de pressão no interior da braçadeira[29].

Os equipamentos automáticos que utilizam o método oscilométrico têm, na sua constituição, como elementos principais, a braçadeira, um tipo de câmara-de-ar onde se coloca o braço e que vai exercer pressão nele, a unidade de controlo, responsável por todo o processamento de dados e comando dos vários módulos, a bomba de ar, responsável por encher a braçadeira, a válvula de ar, responsável por retirar de forma controlada o ar de dentro da braçadeira e por fim o sensor de pressão, responsável pela medição da pressão de ar no interior da braçadeira[30][31].

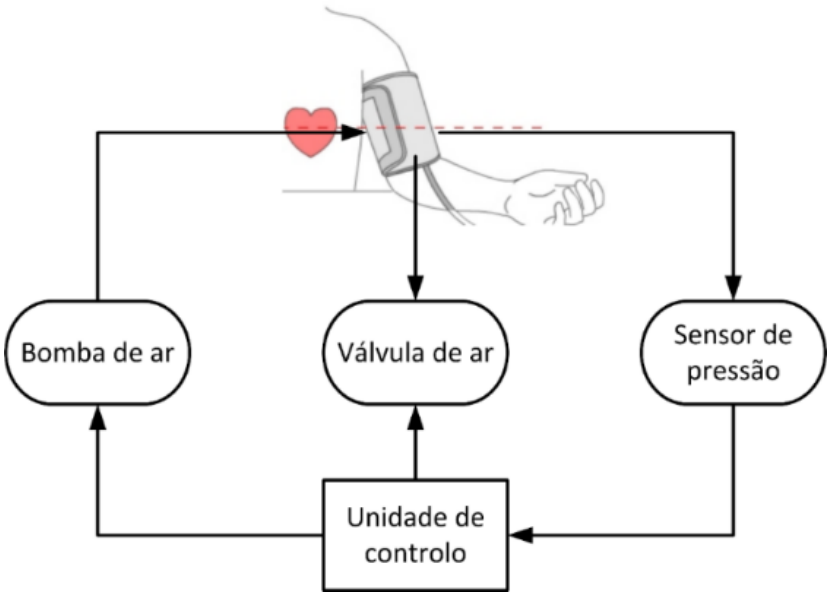


Figura 7 – Diagrama de blocos de um esfigmomanómetro automático

A braçadeira (câmara-de-ar) é colocada no braço esquerdo (para ter uma maior proximidade do coração, o causador do aumento de pressão - bomba) e quando o aparelho inicia o seu processo, de uma forma muito genérica, coloca em funcionamento uma bomba de ar para aumentar a pressão na câmara-de-ar e funcionar como um garrote. Essa bomba mantém-se em funcionamento até que o sensor de pressão de ar registe uma pressão acima da pressão sistólica típica[29].

Completado isso, a bomba de ar desliga deixando de aumentar a pressão no interior da braçadeira e em seguida a válvula de ar abre de modo a esvaziá-la lentamente (baixando a

pressão e aliviando o garrote). À medida que a pressão baixa, fica cada vez mais próxima da pressão sistólica e as pulsações começam a aparecer, sob a forma de pequenas variações de pressão dentro da braçadeira, provocadas pelo batimento cardíaco[30]. O ritmo cardíaco traduz-se no inverso do tempo entre pulsações medido em batimentos por minuto.

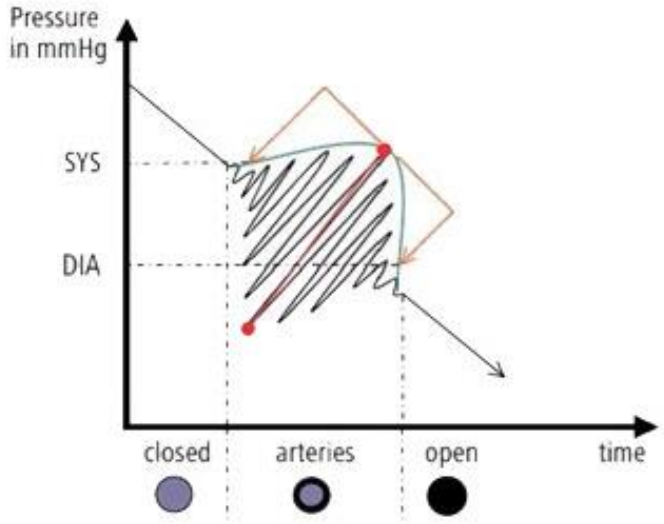


Figura 8 – Variação de pressão no interior da braçadeira, adaptado de [32]

Essas pulsações, variações de pressão, vão ficando cada vez maiores, até se atingir o MAP (*Mean Arterial Pressure*), pressão arterial média. Com o decréscimo da pressão na braçadeira, a amplitude dessas pulsações vai diminuir até desaparecer (ou atingir o valor arbitrado empiricamente como mínimo) registando nessa altura o valor de pressão, como a pressão diastólica.

De forma muito global, o fluxograma da Figura 9 completa toda a descrição, para uma perceção visual mais fácil do seu funcionamento.

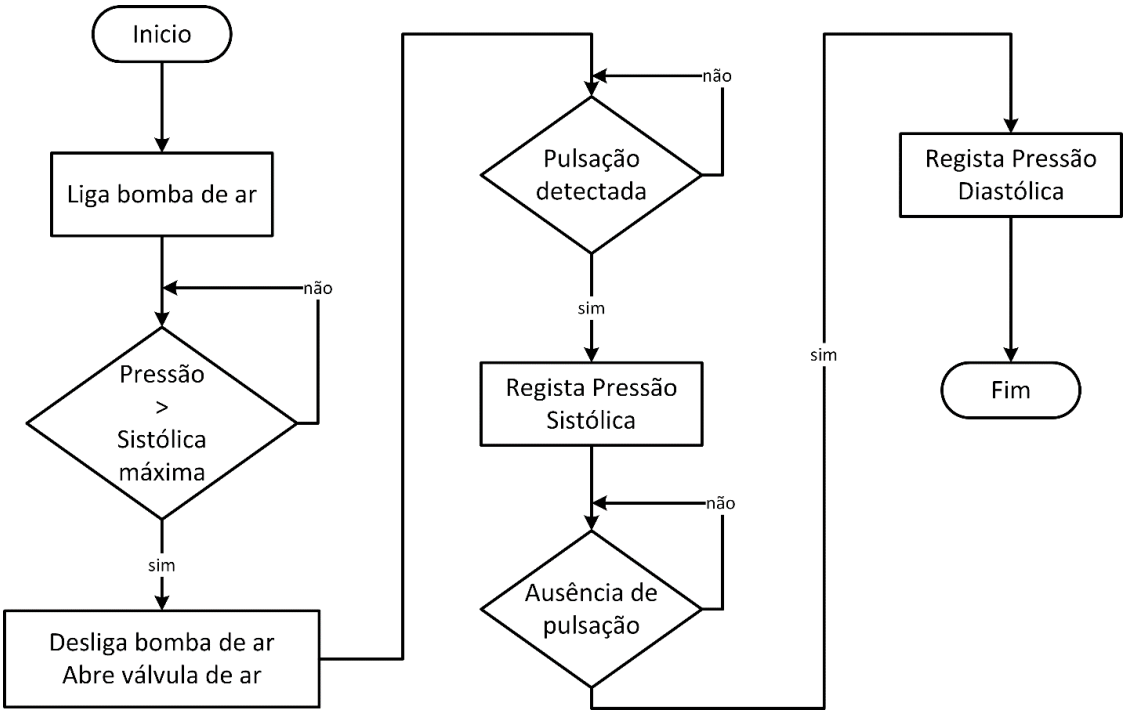


Figura 9 – Fluxograma de funcionamento de um medidor de pressão arterial automático

A maioria da população, que tem um esfigmomanómetro eletrónico portátil em casa, apenas o usa esporadicamente quando se sente indisposta e desconfia que a causa seja uma desregulação da pressão arterial. Estas medições não são válidas aquando da necessidade de um diagnóstico de hiper- ou hipotensão arterial, e surgiu a necessidade de encontrar uma forma em que os registos feitos fossem efetuados nos momentos certos para que um médico, após análise de várias medições, pudesse declarar o diagnóstico[3].

2.4. Esfigmomanómetros de ambulatório

A medição de pressão arterial em ambulatório define-se pela medição de pressão em intervalos de tempo regulares ao longo de um dia ou mais, enquanto os pacientes realizam as funções normais do seu dia-a-dia, incluindo nas horas de dormir. Os intervalos de medição são geralmente de 20 minutos[5][33].

Os medidores modernos são silenciosos, leves e fáceis de transportar, mas o enchimento da braçadeira para medição pode causar algum transtorno, principalmente em pessoas com hipertensão.

O uso de um medidor de ambulatório pode ser indicado quando existe:

- Suspeita de "hipertensão da bata branca" [34][35]

  Fenómeno no qual os pacientes mostram uma pressão arterial acima do normal, quando colocado em ambiente clínico, não correspondendo ao real/normal do dia-a-dia. Acredita-se que esta hipertensão momentânea seja provocada pela ansiedade na visita ao médico.

- Suspeita de "hipertensão mascarada"[34]

  Este fenómeno é o contrário da "hipertensão da bata branca". São pacientes que na visita ao médico mostram uma pressão arterial normal, mas que na verdade sofrem de hipertensão no real dia-a-dia.

- Suspeita de hipertensão noturna ou noite sem redução de pressão arterial[36]

  A medição em ambulatório permite analisar se a pressão arterial do paciente desce durante a noite, quando comparada com os registos durante o dia. Uma descida da pressão durante a noite é normal e desejada, correlacionando-se com a qualidade do sono, idade, apoio social, estado civil e estado hipertensivo.

  A ausência desta descida noturna está associada a um estado de saúde mais fraco incluindo, segundo um estudo recente, o aumento da probabilidade de morte[37].

  Para além disso, hipertensão noturna está associada à lesão terminal de um órgão e é um indicador muito melhor do que a leitura da pressão sanguínea durante o dia.

- Suspeita de um episódio de hipertensão

Além de todas estas indicações diretas, a medição de pressão em ambulatório também pode ser útil em[5]:

- Terapia anti-hipertensiva
- Hipertensão detetada no início da gravidez
- Suspeita ou confirmação de apneia do sono síncope ou outros sintomas ortostáticos sugerindo hipotensão.

2.5. Novos desafios

Relativamente a medidores de pressão arterial em ambulatório, nos dias de hoje, existem algumas dezenas de marcas a produzi-los. Umas mais históricas e conhecidas, como o caso da *Omron*, outras nem tanto. Estes medidores, têm em comum a simplicidade, por norma os poucos botões, e visores pequenos onde é apresentado o resultado da medição. Eles são desenvolvidos para operações de 24 horas (um dia completo de medições, com intervalos geralmente de 20 a 30 minutos[33]). As medições são guardadas internamente na memória não volátil durante as 24 horas [38]. Como consequência, após terminar o período das 24 horas, o utilizador irá descarregar os dados para o computador do médico para ser possível a visualização das suas alterações de pressão arterial ao longo do dia. Este processo implica uma nova deslocação ao posto de saúde com, a consequente, perda de tempo associada.

Em alternativa começam a surgir alguns equipamentos de monitorização remota e de acompanhamento clínico à distância. Com a adoção e implementação da IoT, oferecendo conectividade a todo o tipo de objetos, o número de ligações entre as "coisas" e a Internet crescerá como nunca antes e os dados gerados serão imensos. De acordo com a CISCO, haverá pelo menos 25 biliões de dispositivos ligados à internet até finais de 2015 e em apenas 5 anos, até 2020, prevê que esse valor duplique passando para 50 biliões[39][40]. O objetivo da IoT é permitir conectar objetos a qualquer momento, em qualquer lugar, com qualquer coisa ou qualquer pessoa que use uma tecnologia de conectividade e qualquer serviço[41]. E foi com isto em vista que se começou a ligar a coisa – medidor de pressão arterial – à Internet.

Em Setembro de 2010 Wun-Jin Li, Yuan-Long Luo, Yao-Shun Chang e Yuan-Hsiang Lin[41] desenvolveram um equipamento de medição de pressão arterial com ligação *wireless*, usando o protocolo *ZigBee*. Os dados eram enviados para uma *base station* que se encontrava ligada a um computador pessoal. Esta equipa apontou logo à partida como principal vantagem o facto de, devido à comunicação *wireless*, tornar o equipamento portátil e o utilizador ou o médico poder consultar a qualquer altura e em qualquer lugar, devido à ligação a uma base de dados na web, os registos anteriores numa interface gráfica amigável. Isso permite uma avaliação mais rápida e por sua vez a deteção precoce de eventuais anomalias na saúde do utilizador.

Posteriormente foram realizados mais alguns trabalhos neste mesmo âmbito, como:

- trabalho desenvolvido por Guangtao Zhuang, Xiaoping Zou, Changfei Guo e Yan Liu no ano de 2012[42]. Muito semelhante ao [41] usando também transmissão *wireless*, recorrendo a *Zigbee*, entre um esfigmomanómetro e o computador do médico. É frisada, como uma mais-valia, a poupança de recursos humanos (o clínico não necessita perder tempo a efetuar registos) e diminuição de erros (pois o fator humano pode levar sempre a alguns lapsos);

- trabalho levado a cabo por Zhe-Min Lin, Cheng-Hung Chang, Nai-Kuan Chou e Yuan-Hsiang Lin (participante também no [41]), apresentado em 2014 [43]. Neste foi integrada comunicação *wireless* com recurso a *Bluetooth Low Energy*, BT 4.0, num esfigmomanómetro. Ao contrário dos trabalhos anteriores, a receção é feita utilizando um *smartphone*, sendo por isso um trabalho realizado na perspetiva do utilizador. A energia gasta em comunicações é menor, aumentando o tempo de vida das baterias, e o recetor de dados vai estar a uma distância controlada, pois acompanha a pessoa.

Os trabalhos de I&D que procuravam combinar a importância dos dispositivos médicos com a utilidade do IoT continuaram, e já em 2014 foram lançados no mercado os primeiros medidores compactos com ligação sem fios. São exemplo disso o *Wireless Blood Pressure Monitor* da *iHealth* e o *Wireless Blood Pressure Monitor* da *Withings*[44], com possibilidade de ligar a telemóveis (com sistema operativo compatível) recorrendo à tecnologia *Bluetooth Low Energy*, BT 4.0. Junto com o equipamento é fornecida uma aplicação para que o utilizador possa fazer o registo dos dados (no telemóvel e na *cloud*) e com a possibilidade de partilha nas redes sociais.

Figura 10 – Medidor wireless da Withings, adaptado de [45]

No entanto não são muitas as marcas que atualmente têm equipamentos destes para venda, isto porque, apesar das tecnologias já existirem há alguns anos (BLE lançado em 2010[46]), só nos últimos anos se tem assistido a um real interesse e procura do mercado por *wearables*[47], IoT e semelhantes.

Em todos os casos descritos, quer a nível académico, quer de nível comercial, a necessidade de uma *gateway* é essencial para o funcionamento global do sistema. Esta *gateway* (seja um *smartphone*, *tablet* ou computador) tem como função reencaminhar os dados para a Internet, para um repositório próprio. Por outro lado, o trabalho desenvolvido e descrito nesta dissertação diferencia-se por não necessitar de intermediários e efetuar automaticamente o

envio dos dados, via *Wi-Fi*, para uma plataforma *online* dedicada a IoT. Estes dados podem ser consultados a qualquer momento com recurso a uma interface gráfica agradável. O protótipo final integra todos os componentes essenciais para que, diretamente, possa fazer parte das "coisas" que compõem a IoT.

3. Estudo do medidor de pressão arterial Sanitas modelo SBC 25/1

3.1. Características gerais

Construir um medidor de pressão arterial a título experimental é algo bastante difícil. Desde a construção da braçadeira, tubagens, bomba de ar adequada, etc. Recentemente alguns medidores chegam ao mercado a preços reduzidos o que torna mais apetecível o reaproveitamento de uma peça, de várias peças ou mesmo até de todo o conjunto para outro projeto com outros propósitos. Além da questão financeira, questões legais e de segurança devem também ser tidas em consideração, as certificações normativas, os testes clínicos realizados em pacientes e o *design* do produto, entre outras.

Um desses medidores, que está no mercado com um preço a rondar os 15 euros é o medidor de pressão arterial "Sanitas SBC 25/1". Este é um medidor totalmente automático para usar no pulso, bastante pequeno e leve (cerca de 105 gramas sem baterias).Possui um display grande e de fácil leitura, uma braçadeira para pulsos com perímetros de 13.5cm a 19.5cm e possibilita a deteção de arritmias cardíacas. Possui a capacidade de poder ser usado em duas pessoas e gravar as últimas 60 medições de cada uma.

Obviamente o preço foi um dos pontos a considerar na escolha deste equipamento, mas outros pontos também pesaram na sua escolha, como o seu diminuto tamanho, ser um medidor de pulso, a sua portabilidade, o facto de usar baterias (alimentado com duas baterias AAA) e em grande parte pelo facto de efetuar registo de "2 x 60 posições" (Figura 12).

Figura 11 – Medidor Sanitas SBC 25/1, adaptado de [48]

Figura 12 – Caixa do medidor

Permitir gravar 60 medições completas (entende-se por medição completa o registo da pressão sistólica, pressão diastólica, batimentos cardíacos por minuto, hora e data da medição) para duas pessoas, é um bom indicativo de que o equipamento possui uma EEPROM[2][49] capaz de gravar tudo isto. Se esta condição se verificar, para conseguir aceder às leituras, basta "pendurar" um *sniffer*[50] (possivelmente um *sniffer* de I^2C [3][51], dado que a grande maioria das memórias utilizam este protocolo) no barramento onde se encontra ligada a EEPROM. O *sniffer* constituído por *software* ou *hardware*, é capaz de intercetar e registar o tráfego de dados numa dada rede para posterior descodificação dos pacotes[52]. Esse é o objetivo final, mas numa fase inicial e para análise das *frames*[4] a guardar, o uso de um analisador lógico torna-se essencial.

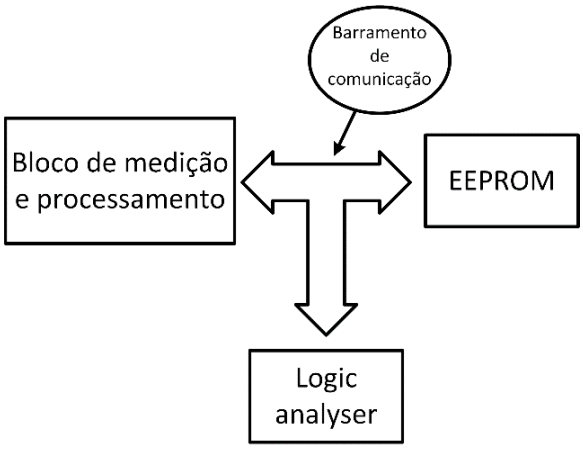


Figura 13 – Ligação entre os diferentes blocos para análise

[2] EEPROM é uma memória não volátil, que permite ler, apagar e reescrever posições de memória individuais.

[3] I^2C é um barramento serial desenvolvido pela Philips que é usado para conectar periféricos de baixa velocidade.

[4] Frame é um pacote de dados, também designada por trama.

3.2. Análise e interpretação dos dados transferidos no barramento da EEPROM

Para a realização desta etapa foi utilizado um analisador lógico com interface USB, que para funcionar necessita estar ligado a um computador com o *software* correto[5]. Em termos de *hardware*, este analisador permite uma frequência de amostragem de até 24MHz (variável e configurável via *software*), oito canais de entrada e compatibilidade de funcionamento com os sistemas típicos de 5V, 3.3V, 2.5V e 2V. O medidor de pressão arterial "Sanitas SBC 25/1", alimentado a um máximo de 3V fornecidos pelas duas baterias AAA, está compreendido nos valores de tensão do analisador, por isso ambos podem ser ligados para iniciar a análise. Os fios soldados na EEPROM são ligados ao analisador lógico para se fazer o estudo dos dados transferidos no barramento.

No entanto, para que haja dados a circular no barramento I^2C, que liga o processador à EEPROM do medidor de tensão, é necessário forçar essa mesma consulta ou a escrita de dados para a memória.

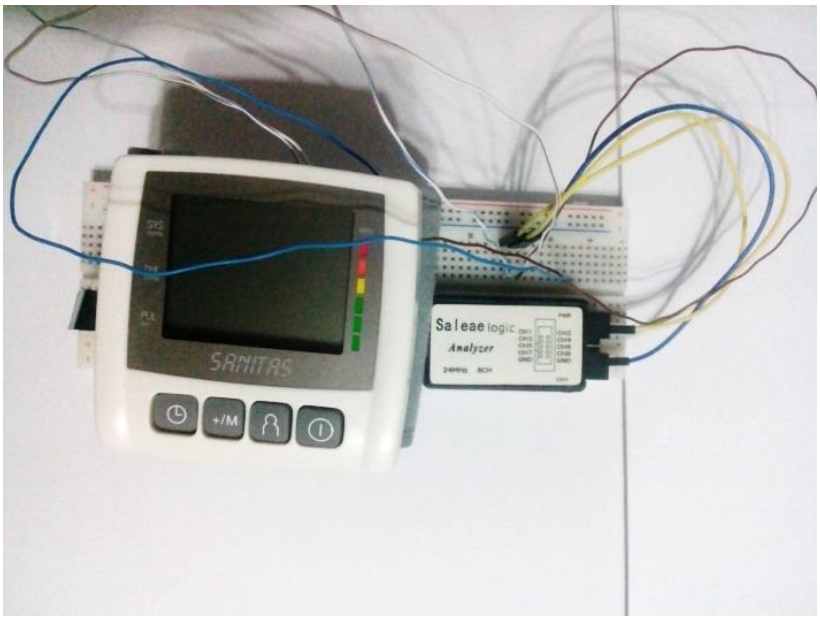


Figura 14 – Ligação de teste entre o analisador lógico e o Sanitas SBC 25/1

Existem duas formas de forçar essa troca de dados, uma delas é fazer uma medição real e registar todos os dados transferidos ao longo do processo e a outra passa por utilizar a função de consulta de registos anteriores de modo a forçar a leitura da EEPROM pelo processador e consultar medições antigas. A repetição deste último método fica somente à distância de "um

[5] *Saleae Logic 1.1.15*

clique" e por ser a técnica mais rápida, foi considerada a melhor escolha para a realização dos testes.

A próxima meta passa por conseguir entender como são os dados das medições gravados na EEPROM, em que formato (no que toca à organização de memória) e correlacionar esses mesmos dados com base nas informações mostradas no ecrã do medidor.

O *software* do analisador de estados lógicos foi configurado para que o *trigger*, que dá início à captura, disparasse no primeiro flanco descendente do pino de *clock* da EEPROM. Na transferência de dados no barramento é necessário que o mestre (neste caso o processador do medidor de pressão) gere os pulsos de *clock*. Assim ao configurar o *trigger* para a linha de *clock* existe garantia de começar a gravar transições sem a perda de quaisquer dados.

Ao pressionar o botão "+/M" (botão de memória), verifica-se que em memória se encontra o registo de medição mostrado na Figura 15, onde se observa:

- Pressão sistólica – 109mmHg
- Pressão diastólica – 68mmHg
- Batimento cardíaco (por minuto) – 74bpm
- Horas – 21:20
- Data – 21/7/2015

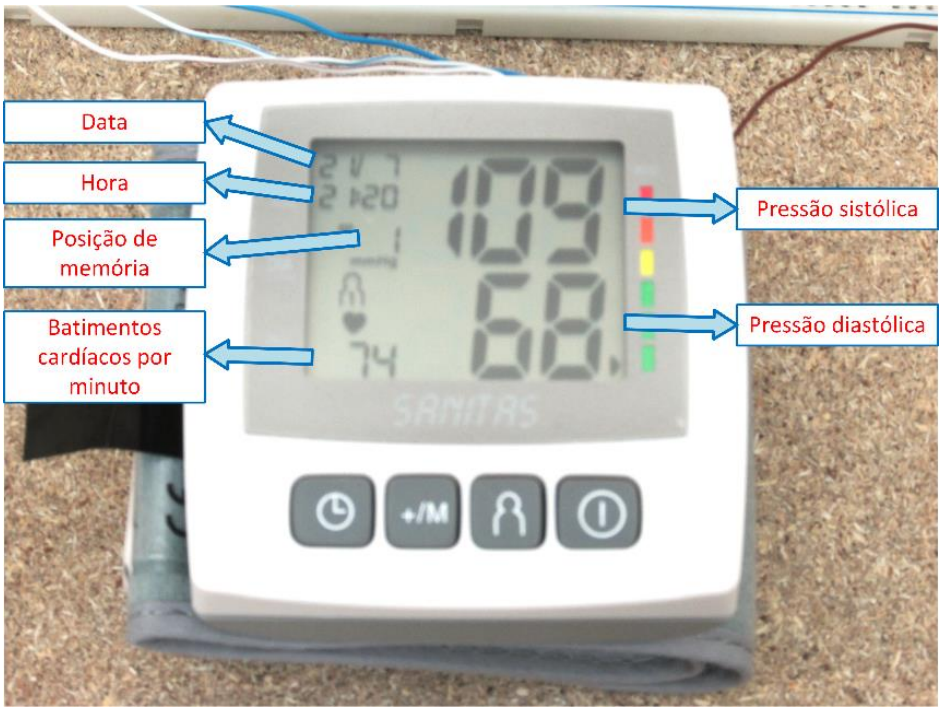


Figura 15 – Esfigmomanómetro mostrando os resultados de uma medição e suas legendas

A Figura 16 mostra a captura, de toda a comunicação, realizada durante a consulta desse registo de medição, guardada na EEPROM.

O *software* possui uma ferramenta de descodificação para alguns protocolos, como é o caso do I²C. Essa ferramenta será usada para facilitar o entendimento de como a *frame* é organizada e de como obter os valores resultantes de uma medição.

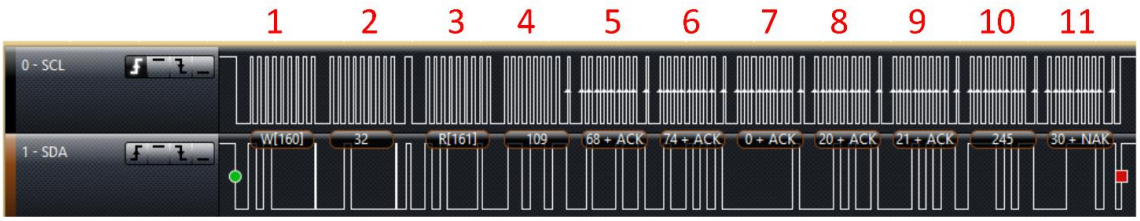


Figura 16 – Captura da comunicação numa leitura de memória

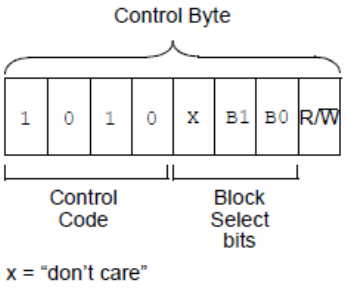
Dado o comprimento da *frame* é conveniente realizar uma análise *byte* a *byte* com o intuito de conseguir uma correspondência e encontrar forma de descodificar os dados nela contidos.

Bytes 1 e 2

Figura 17 – Pormenor de captura *bytes* 1 e 2

O primeiro *byte* contém o comando escrita, com decimal "160" e recebe ACK (*acknowledge*) por parte da EEPROM (escravo).

O decimal "160" corresponde a 0b10100000 em binário. Todas as comunicações, usando protocolo I²C, devem começar pelo endereço do escravo. Confirmando com o *datasheet* da EEPROM 24LC08, a construção deste primeiro *byte* tem a sua representação na Figura 18.



Operation	Control Code	Block Select	R/$\overline{W}$
Read	1010	Block Address	1
Write	1010	Block Address	0

Figura 18 – Byte de controlo[53]

O primeiro *nibble*, 0b1010, corresponde ao "*control code*", os três bits seguintes ao endereço da EEPROM (no caso em análise, 0b000) e o último bit define o tipo de operação a realizar, se se trata de uma operação de escrita, 0b0, ou de leitura, 0b1.

O *byte* 2 transporta o decimal "32" que, como é normal numa comunicação I^2C, corresponde ao endereço de uma posição de memória na EEPROM. É a partir desta posição que a leitura seguinte será iniciada. Dado que a informação apresentada no ecrã corresponde ao primeiro registo em memória, pode assumir-se que as medições futuras terão como primeiro endereço o seguinte:

$$endereço = 32 + 8 * (M - 1)$$

Cada medição de pressão em memória ocupará, como será mostrado mais à frente, oito *bytes*. Como o registo da memória nº1 tem como endereço de início de escrita a posição 32, então a memória nº1 será armazenada do endereço 32 ao 40, a memória nº2 armazenada do 40 ao 48 e assim por diante. Então o endereço de início de escrita de uma memória, M, é dado pelo cálculo em cima.

byte 1

bin	1	0	1	0	0	1	0	0	comando 'write'
	control code				ADD			W	

byte 2

bin	x	x	x	x	x	x	x	x	endereço inicial

Figura 19 – Resumo *bytes* 1 e 2

Bytes 3 e 4

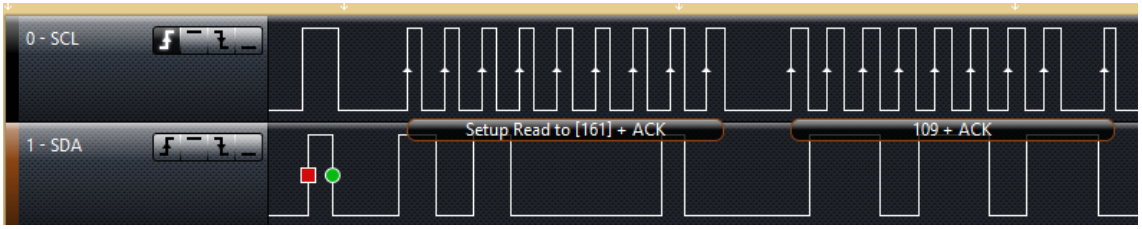


Figura 20 - Pormenor de captura *bytes* 3 e 4

Após ter sido enviado o comando de escrita para a EEPROM (*byte* 1) e de se ter enviado o endereço da posição de memória a ler (*byte* 2) é realizado o *restart*[51], *stop* seguido de *start* (Figura 20, conjunto quadrado vermelho e círculo verde), para enviar em seguida um novo comando[54]. O *byte* 3 é enviado com o endereço da EEPROM e com o comando de leitura,

último bit a 1 (161 ~ 0b10100001). Assim a EEPROM já tem a indicação para dar resposta e enviar o que está guardado nesse endereço de memória, *byte* 4 em diante.

Sempre que a EEPROM tiver respondido com o valor guardado na posição indicada pelo mestre, este último envia o *acknowledge*[51] e com isto a EEPROM reconhece que deverá incrementar o endereço a ler e enviar o valor guardado nele. Esta metodologia própria do I^2C repete-se até que o mestre (processador do medidor Sanitas) envie um NACK, *not acknowledge*[51], cessando a comunicação por parte do escravo (EEPROM).

De notar que o *byte* 4 surge com o decimal "109", cujo valor é igual ao mostrado no ecrã para a pressão sistólica. Este é um bom indicador de que o *byte* poderá de facto corresponder a essa medida.

byte 3									
bin	1	0	1	0	0	1	0	1	comando 'read'
	control code				ADD			R	
byte 4									
bin	x	x	x	x	x	x	x	x	decimal - pressão sistólica

Figura 21 - Resumo *bytes* 3 e 4

Bytes 5 e 6

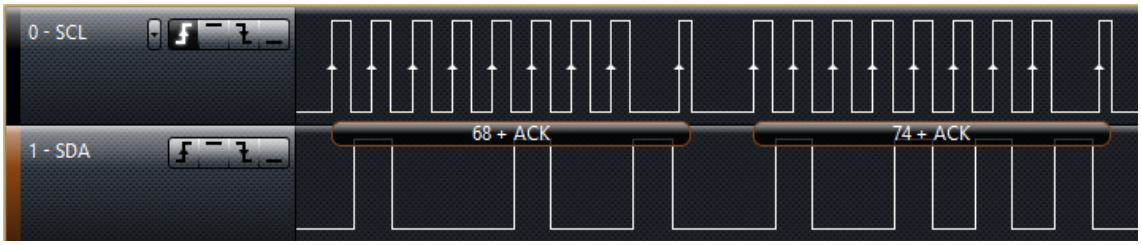


Figura 22 - Pormenor de captura *bytes* 5 e 6

Os *bytes* 5 e 6, surgem com o decimal "68" e "74", respetivamente. Estes mesmos valores são mostrados como pressão diastólica, "68", e como batimentos cardíacos por minuto, "74".

Apenas com a execução de várias medições e várias leituras se poderão confirmar estes valores como certos e não serem consideramos apenas por coincidência neste teste. No entanto, chegado aqui, existe uma forte suspeita de que os valores registados podem realmente estar contidos nestes *bytes*.

byte 5

bin	x	x	x	x	x	x	x	x	decimal - pressão diastólica

byte 6

bin	x	x	x	x	x	x	x	x	decimal - batimentos por minuto

Figura 23 - Resumo *bytes* 5 e 6

Bytes 7 e 8

Figura 24 - Pormenor de captura *bytes* 7 e 8

Embora já se tenha um bom indício de onde se localizam os dados mais importantes (pressão sistólica, diastólica e batimento cardíaco), analisar a *frame* por completo é uma boa prática e uma forma de ter mais informações sobre que dados circulam no barramento e qual a sua organização.

O *byte* 7, com o decimal "0", não aparenta ter nenhuma ligação lógica com a medição, mas o mesmo não acontece com o *byte* 8 que com o decimal "20" pode muito bem corresponder aos minutos em que a medição foi realizada (registo no ecrã, 21:20).

byte 7

bin	0	0	0	0	0	0	0	0	sempre zero - separação

byte 8

bin	x	x	x	x	x	x	x	x	decimal - minutos

Figura 25 - Resumo *bytes* 7 e 8

***Bytes* 9, 10 e 11**

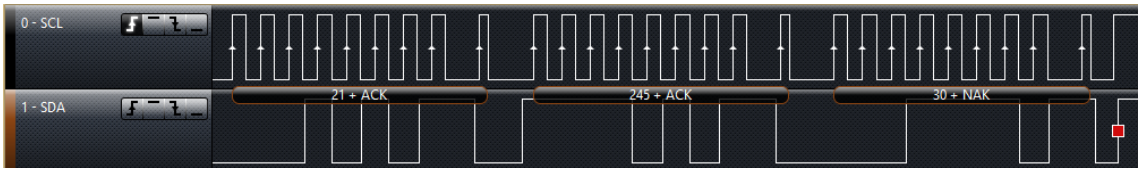


Figura 26 - Pormenor de captura *bytes* 9, 10 e 11

Por fim, são transferidos os *bytes* 9,10 e 11, antes de o mestre enviar o NACK para terminar com a comunicação do escravo (EEPROM) e enviar a condição de *stop*[51] (Figura 26 quadrado a vermelho).

O *byte* 9 possui o decimal "21" tal como as horas a que se realizou a medição, 21 horas. Pode ser mais uma vez um bom indício que que este *byte* poderá corresponder às horas.

O *byte* 10, com o decimal "245", não tem uma associação tão fácil e rápida como as anteriores. Para completar todas as informações que são mostradas no ecrã do medidor, fica a faltar a leitura da data, composta por dia, mês e ano. O valor 245 pode ser escrito em binário por 0b11110101. Os últimos cinco bits, 0b10101, correspondem ao decimal "21" cujo valor é igual ao dia da medição. Os primeiros três bits, 0b111, correspondem ao decimal "7" que é o mesmo número do mês do registo. No entanto, a verificar com repetição de análises deste tipo, 7 é o maior decimal que se consegue com três bits, por isso para conseguir guardar os meses seguintes, tem de existir um outro bit que complete estes três e assim consiga guardar o número de todos os meses do ano.

O último *byte* a ser transferido, *byte* 11, transporta o decimal "30", que por exclusão de partes terá que ter a informação do ano e possivelmente um bit para completar o que falta do mês. O valor 30 é sinónimo de 0b00011110. Os primeiros sete bits, 0b0001111, correspondem ao valor "15" o que leva a crer que se trata dos últimos dois dígitos do ano, 2015 – "15". O último bit, que está a zero, será portanto o bit que faltava na identificação do mês, sendo este mesmo, o bit mais significativo.

O mestre termina a transferência de dados com o envio do NACK.

byte 9									
bin	x	x	x	x	x	x	x	x	decimal - horas

byte 10									
bin	x	x	x	x	x	x	x	x	mês / dia
	mês				dia				

byte 11									
bin	x	x	x	x	x	x	x	x	ano (2015 = '15') / mês
	ano							mês	

Figura 27 - Resumo *bytes* 9, 10 e 11

Vários foram os testes realizados para que as conclusões já referidas deixassem de ser suposições e passassem a ser certezas. Foram realizados mais de uma dezena de testes em datas e circunstâncias diferentes, onde tudo foi coerente com o que já foi descrito e só por isso se admite que a organização dos dados corresponde ao previsto. No entanto, de forma breve e para provar isso mesmo, realizou-se mais uma medição (relatado em seguida), que contém alguns valores extremos, como o caso de meses superiores ao 7, julho.

Em memória encontra-se o registo de medição que pode ser visto na Figura 28, onde se observa:

- Pressão sistólica – 108mmHg
- Pressão diastólica – 58mmHg
- Batimento cardíaco (por minuto) – 80bpm
- Horas – 12:34
- Data – 31/12/2015

Figura 28 - Esfigmomanómetro mostrando os resultados de uma nova medição

Usando o mesmo método que anteriormente, recorreu-se à consulta de um registo de medição de pressão arterial já gravado na memória. Enquanto isso, com o auxílio do analisador lógico, registaram-se todos os *bytes* transferidos, resultando na Figura 29.

Uma vez que os dados realmente importantes para este trabalho e os que trazem alguma informação útil, se encontram dos *bytes* 4 a 11, é nestes que se dará uma maior enfoque.

Figura 29 - Captura da comunicação numa nova leitura de memória

Byte 4

Contém o decimal "108", coincidente com o valor da pressão sistólica.

Manteve-se em todos os testes.

Byte 5

Contém o decimal "58", coincidente com o valor da pressão diastólica.

Manteve-se em todos os testes.

Byte 6

Contém o decimal "80", coincidente com o valor do batimento cardíaco por minuto.

Manteve-se em todos os testes.

Byte 7

Contém o decimal "0". Em todos os testes realizados, este *byte* assumiu sempre o mesmo valor, pelo que se conclui que será uma separação entre os valores resultantes da medição e a data em que esta foi feita. Possivelmente para facilitar a organização de memória na EEPROM.

Manteve-se em todos os testes.

Byte 8

Contém o decimal “34”, coincidente com o valor dos minutos.

Manteve-se em todos os testes.

Byte 9

Contém o decimal “12”, coincidente com o valor das horas.

Manteve-se em todos os testes.

Byte 10

Contém o decimal “159”, corresponde ao binário 0b10011111. Os últimos cinco bits, 0b11111, correspondem ao valor decimal 31 coincidente com o dia de medição.

Manteve-se em todos os testes.

Os três primeiros bits sobrantes, 0b100, são analisados em conjunto com o *byte* 11.

Byte 11

Contém o decimal “31”, corresponde ao binário 0b00011111.

O último bit, 0b1, é o bit mais significativo do mês por isso o mês será obtido do binário 0b1100 que corresponde ao decimal “12” (mês de dezembro) e coincide com o mostrado no ecrã.

Manteve-se em todos os testes.

Os restantes bits, 0b0001111, correspondem ao decimal “15” e coincide com o ano do registo, 20**15**.

Manteve-se em todos os testes.

O quadro apresentado na Figura 30, mostra em jeito de resumo, as conclusões tiradas ao longo desta análise. Assim pode facilmente visualizar-se a forma como os dados são organizados, qual a sua ordem e a composição dos *bytes* trocados no barramento I^2C entre o processador do medidor Sanitas SBC 25/1 e a EEPROM onde é guardado o historial das medições anteriores.

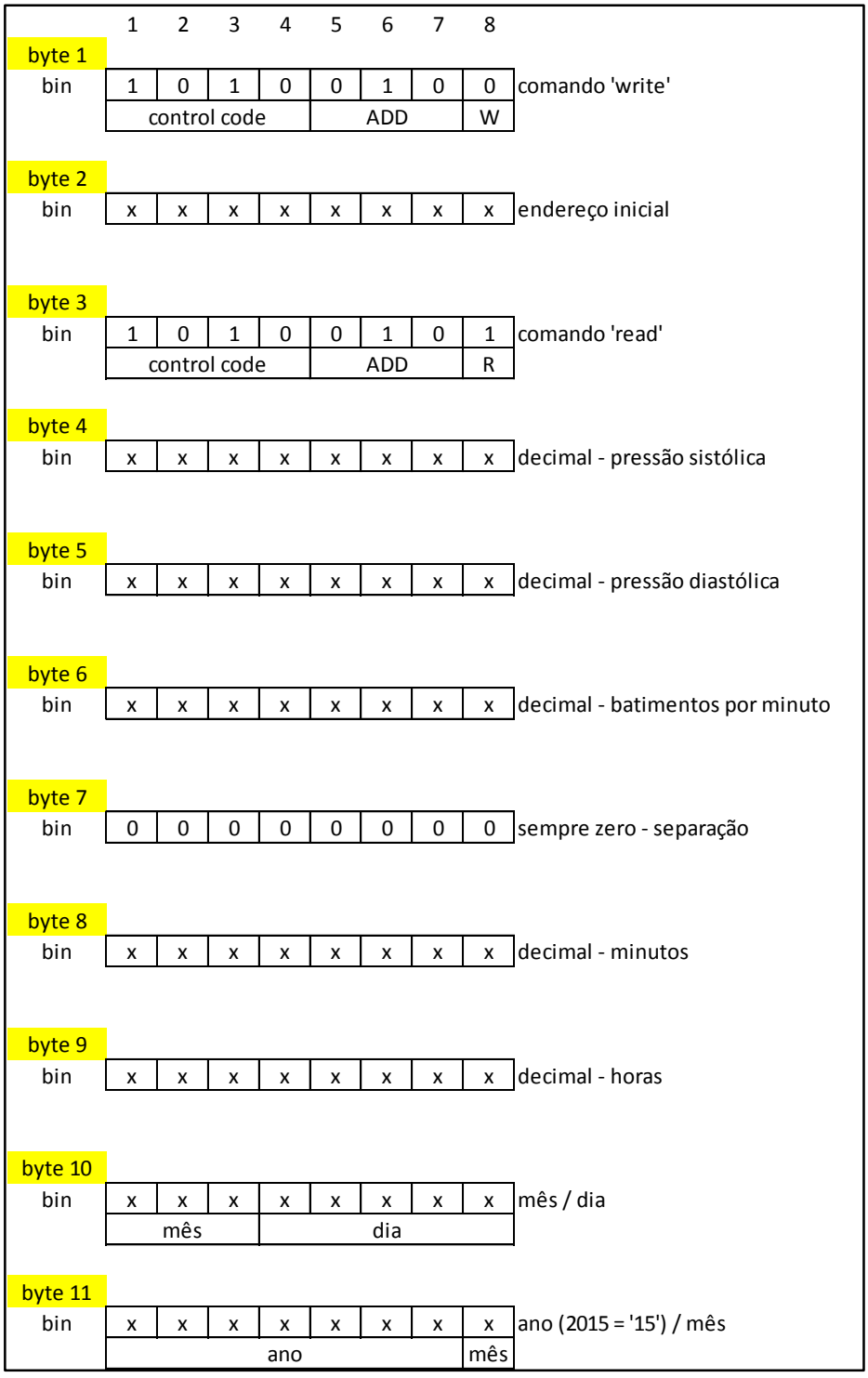


Figura 30 - Quadro resumo da organização dados na *frame*

3.3. Análise da transferência de dados ao longo de uma medição completa

Conhecer a forma como a memória e os dados são organizados é a base para o próximo passo, analisar as *frames* trocadas durante toda uma medição. Saber que comunicações são feitas nesse período é crucial para dimensionar um algoritmo, a correr num microcontrolador, capaz de reconhecer quando os dados referentes à medição mais recente estão a ser gravados na EEPROM assim como identificar esses valores.

Usando portanto o mesmo tipo de ligação, medidor de pressão – analisador lógico, aumentou-se a memória de gravação do *software* do analisador, mantendo o mesmo *trigger* (primeiro flanco descendente do *clock*) mas em vez de se consultar a memória do medidor de pressão, inicia-se uma nova medição, no botão "ON/OFF". Ao longo de todo o processo de medição de pressão arterial, o *software* foi registando todas as alterações ocorridas, aguardando que a medição findasse.

Terminada a medição, obtiveram-se os valores mostrados na Figura 31 e que devem ser encontrados em alguma *frame* da comunicação do processador com a EEPROM do aparelho.

- Pressão sistólica – 114mmHg
- Pressão diastólica – 67mmHg
- Batimento cardíaco (por minuto) – 79bpm
- Horas – 14:16
- Data – 29/07/2015

Figura 31 – Resultados considerados para interpretação da medição completa

Durante todo o processo de medição de pressão trocam-se, em momentos distintos, três blocos de comunicação, como mostrado na Figura 32. Cada medição leva em média cerca de 50 segundos, por isso mesmo um analisador lógico com ligação USB permite ter capacidade de armazenamento suficiente.

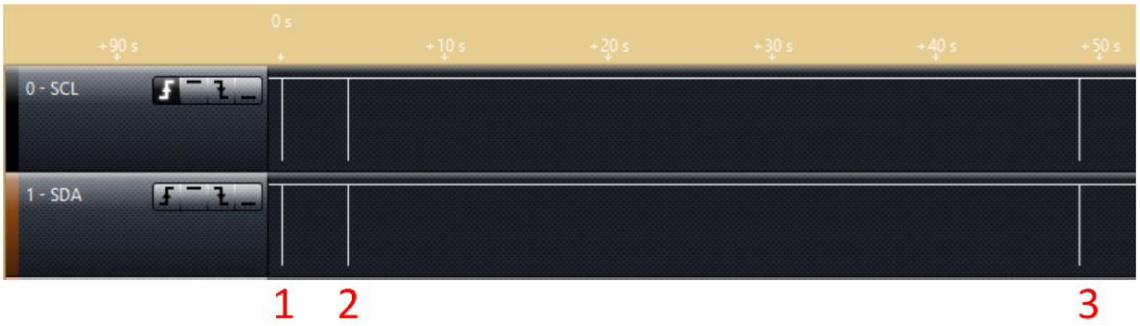


Figura 32 – Marcação dos 3 momentos distintos de acesso à memória

Nestes blocos, procura-se uma *frame* de escrita na EEPROM com a mesma organização que a mostrada na Figura 30 e que contenha os valores da medição efetuada, Figura 31.

O primeiro bloco de transferência de dados é composto por cinco *frames* (Figura 33).

- 1ª – contém informação constante e nenhum *byte* varia entre leituras;
- 2ª – contém informação referente à última leitura realizada;
- 3ª – contém informação referente à penúltima leitura realizada;
- 4ª – contém informação referente à antepenúltima leitura realizada;
- 5ª – contém informação, no 11º *byte*, do número na última leitura em memória (posição de memória);

Figura 33 – Pormenor do 1º bloco de transferência de dados

Nenhuma destas *frames* contém informações úteis, ou seja a informação com os valores resultantes da medição de pressão. No entanto, faz sentido que essa informação não seja

encontrada neste primeiro bloco, pois quando o processador acede à EEPROM, o processo de medição propriamente dito ainda não se iniciou.

Cerca de quatro segundos depois é iniciado mais um bloco de transferência de dados, composto por três *frames* (Figura 34).

- 1ª – contém informação referente à última leitura realizada;
- 2ª – contém informação referente à penúltima leitura realizada;
- 3ª – contém informação referente à antepenúltima leitura realizada;

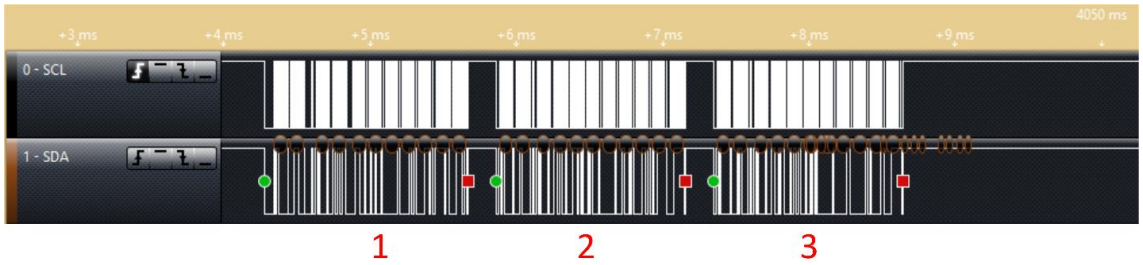


Figura 34 - Pormenor do 2º bloco de transferência de dados

Após aproximadamente 45 segundos termina o processo de medição e efetua-se um novo bloco de transferência de dados.

Desta vez o bloco é composto por quatro *frames* (Figura 35) e é neste que se espera encontrar o comando de escrita do processador para a EEPROM, por exclusão de partes.

- 1ª – contém informação referente à última leitura realizada;
- 2ª – grava os valores da medição na EEPROM;
- 3ª – contém informação, no 11º *byte*, do número na última leitura em memória (posição de memória);
- 4ª – incrementa a posição lida na *frame* anterior e atualiza esse valor na EEPROM;

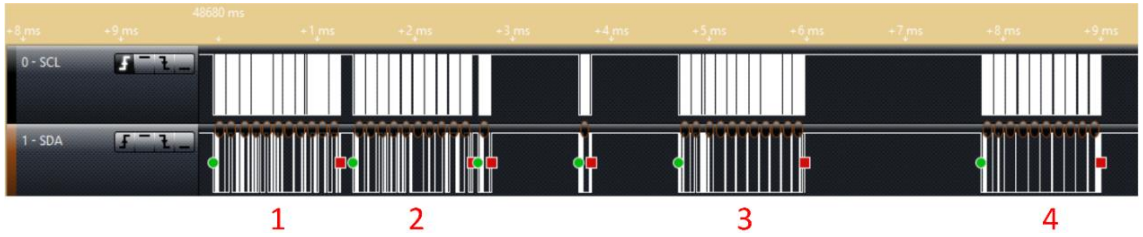


Figura 35 - Pormenor do 3º bloco de transferência de dados

A Figura 36 ilustra em pormenor a *frame* nº2 do terceiro bloco de transferência de dados, que faz gravar na EEPROM os valores da medição.

Esta é a sequência de *bytes* mais importante de todo o processo e aquela que deve ser procurada para obtenção dos valores da medição. A *frame* é composta por 10 *bytes* e a sua caracterização é feita em seguida.

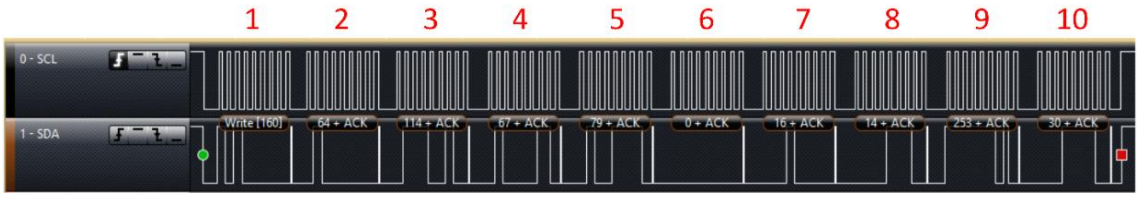


Figura 36 – *Frame* responsavel por gravar os dados da medição na EEPROM

Byte 1

Decimal “160”, endereço e ordem de escrita para a EEPROM.

Byte 2

A Figura 31 mostra que este registo de medição foi gravado na posição de memória nº5. Usando a equação deduzida para encontrar o endereço de início de escrita, pode ser feito o seguinte cálculo:

$$M = 5 \qquad endereço = 32 + 8 * (5 - 1) = 64$$

O *byte* 2 tem o decimal “64”, que corresponde ao valor calculado e que correspondente à posição de memória onde se inicia a escrita de valores, referente ao registo contido na memória nº5.

Byte 3

Decimal “114”, correspondente ao valor medido da pressão sistólica.

Byte 4

Decimal “67”, correspondente ao valor medido da pressão diastólica.

Byte 5

Decimal “79”, correspondente ao valor medido de batimentos por minuto.

Byte 6, 7, 8, 9 e 10

Obedecem à ao quadro da Figura 30, contendo a informação das horas (14), minutos (16), dia (29), mês (07) e ano (2015).

Por fim, tal como aconteceu com o resumo da composição das *frames* sob a forma de um quadro também a Figura 37 se justifica. Este quadro contém somente um resumo geral, para mais fácil visualização dos blocos e da sequência de *frames* trocadas no barramento.

De frisar que em toda a troca de *bytes* existem apenas duas *frames* com a ordem de escrita por parte do processador: *frame* 10, que contém os valores da medição e a *frame* 12 que incrementa o número de memória do aparelho em que ficou registada esta última medição. Encontrando a primeira *frame* com capacidade de escrever na memória, encontra-se a *frame* certa.

Trama	cmd	ADD	cmd	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8
1	W	0	R	165	3	62	205	218	66	0	205
	constante - não varia com novas leituras										
2	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-1)										
3	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-2)										
4	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-3)										
5	W	16	R	0	0	0	0	0	0	0	x
	número de memória da leitura (-1)										
	pausa de aprox. 4s										
6	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-1)										
7	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-2)										
8	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-3)										
	pausa para nova leitura, aprox. 25s										
9	W	>= 32	R	x	x	x	x	x	x	x	x
	memória de leitura (-1)										
10	W	>= 32		x	x	x	x	x	x	x	x
	guarda na eeprom a leitura actual										
11	W	16	R	0	0	0	0	0	0	0	x
	número da memória da leitura (-1)										
12	W	16		0	0	0	0	0	0	0	x
	guarda na eeprom o número de memória da leitura actual										

Figura 37 – Resumo das *frames* trocadas no barramento ao longo de uma medição completa

3.4. Algoritmo para a leitura dos dados no barramento

O algoritmo aqui exposto será executado num microcontrolador, MCU, que será o responsável pela implementação do *sniffer*, aquisição dos valores e gestão de todos os componentes. O microcontrolador escolhido foi o PIC18LF25K22[55] da Microchip. Este MCU possui imensas características, mas apenas as mais importantes e relevantes para este trabalho serão enumeradas.

Algumas das razões que pesaram na sua escolha foram o seu core de alta performance (incluindo um poder de processamento de 16 MIPS), a incorporação da mais recente tecnologia em *low power* da Microchip, a nanoWatt XLP[56], e o seu oscilador interno de 16MHz com erro de ±1% e a possibilidade de recorrer a uma PLL (Phase Locked Loop) capaz de quadruplicar a frequência do *clock* principal para 64MHz. Existem ainda alguns periféricos internos implementados diretamente em *hardware* e independentes do core principal que serão muito úteis neste trabalho, como os periféricos de comunicação com dois periféricos de MSSP (contém implementação em hardware de dois *transceivers* de I^2C) e dois periféricos de USART. A versão "LF" escolhida tem uma gama de operação com tensões de alimentação desde 1.8V a 3.6V, com algumas limitações de *clock* apenas. No que toca ao tipo de encapsulamento escolhido, dos vários existentes, escolheu-se o SOIC de 28 pinos por ser fácil de soldar em montagem de superfície e não haver necessidade de mais pinos que os 26 pinos de input/output disponíveis.

Embora este microcontrolador possua um periférico interno, implementado em *hardware*, dedicado para comunicação I^2C, não existe forma de o "pendurar" diretamente no barramento sem que ele interfira com as normais trocas de dados. O periférico que o MCU possui é capaz de funcionar tanto como mestre como como escravo. O funcionamento em mestre, neste caso, não faria sentido pois o objetivo é apenas receber as comunicações e criaria uma situação de *multi-master*[57]. Surge então a opção de funcionar como escravo, mas o periférico não vai receber dados se não tiver o mesmo endereço que a EEPROM, por outro lado se tiver o mesmo endereço que a EEPROM vai começar a enviar ACK, sobrepondo-se à comunicação da mesma. Esta situação fará com que o protocolo não seja cumprido e os dados trocados no barramento serão inúteis.

A opção única é criar um *sniffer* por *software*. Ligar os pinos de SDA e SCL a dois pinos de entrada do microcontrolador (que foram definidos com os mesmos nomes) e correr uma rotina que vai supervisionar o estado destes e procurar a *frame* válida para que possa extrair os dados úteis, dados correspondentes aos valores da medição.

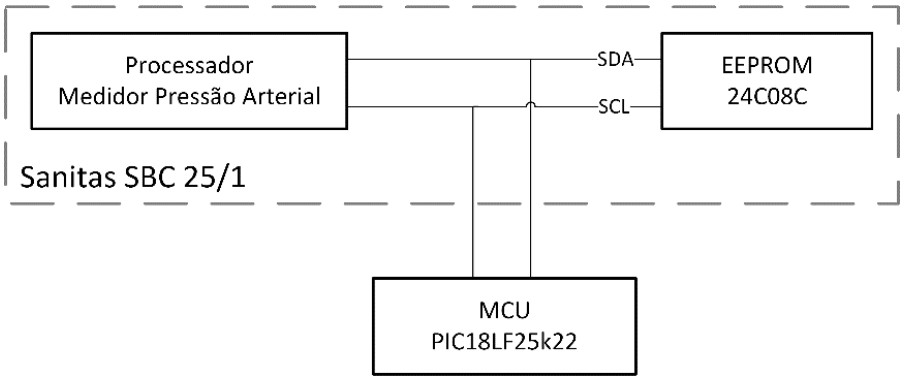


Figura 38 – Diagrama de ligações para implementação do *sniffer*

Qualquer comunicação I^2C começa com a condição de *start*[51] e é imperativo fazer a deteção da mesma, pois esta é a condição que indica o começo de uma nova *frame*. Olhando para a Figura 37 é na 10ª *frame* que se encontram os dados importantes e, como já concluído, o segundo *byte* transporta o endereço de início de escrita (dos oito *bytes* de dados) que terá de ser por obrigação superior ou igual ao decimal 32 e não ser seguido de uma condição de *restart*.

De modo a simplificar a descrição deste procedimento, o mesmo vai ser subdividido em processos, embora façam todos parte de uma rotina maior, a rotina *i2c_frame*.

O objetivo passa por detetar a condição de *start*, esta resulta de uma transição de *high* para *low* do pino de dados, SDA, enquanto o pino de *clock*, SCL, se mantém em estado *high*.

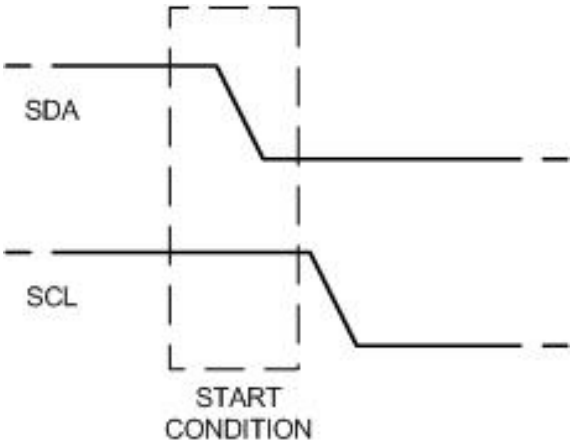


Figura 39 – Condição de *start*, adaptado de [58]

É feita uma verificação constante ao estado dos dois pinos para determinar qual deles passa para *low state* em primeiro lugar, se for o SCL é um bit normal, mas se for o SDA é uma condição de *start*.

O seguinte fluxograma verifica se o SDA muda de estado enquanto o SCL está em *high state* (condição de *start*).

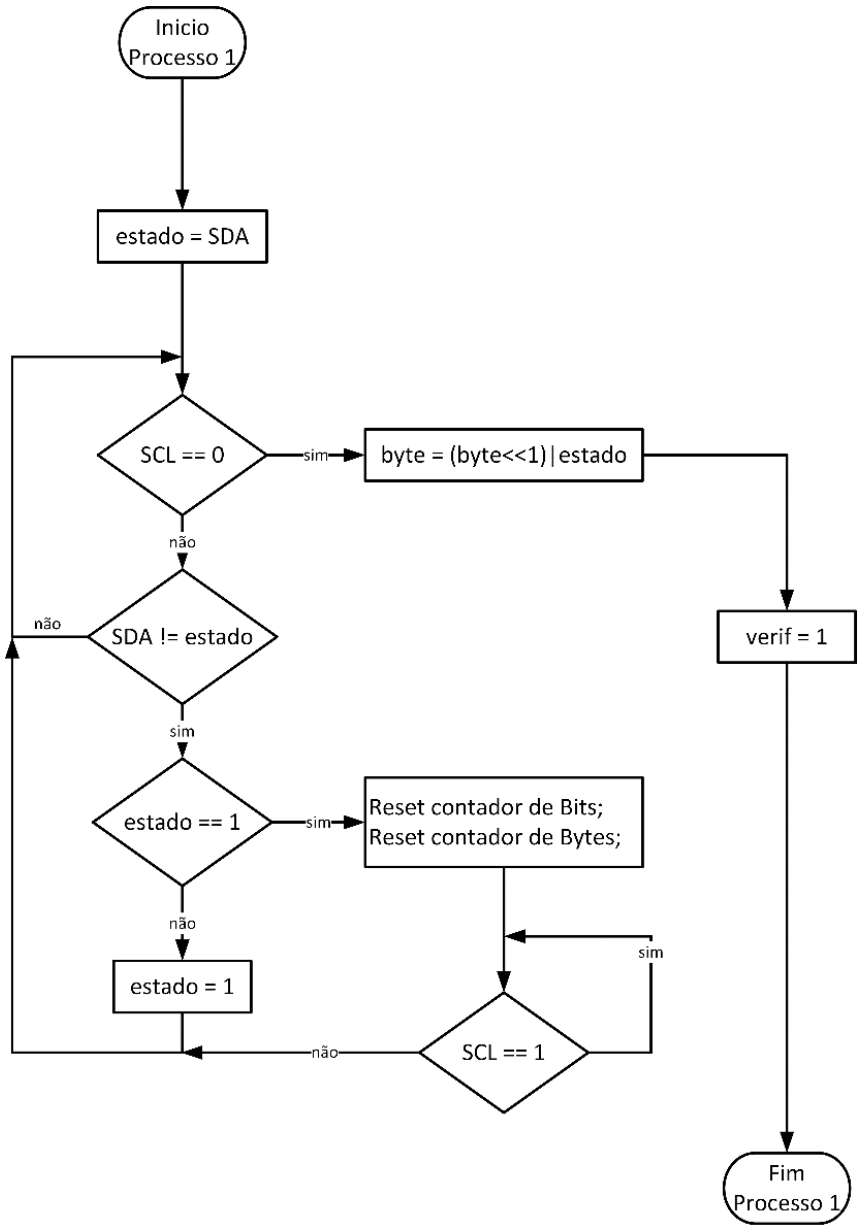


Figura 40 – Fluxograma explicativo do processo 1

Enquanto o pino SCL continuar em *high state* mas o SDA mudar de estado significa uma de duas coisas: ocorreu uma condição de *start* ou ocorreu uma condição de *stop*[51]. Manter o registo do estado anterior do SDA é muito importante para fazer a distinção destes dois casos.

Se o estado anterior era 1, *high*, e houve alteração de estado, ocorreu a receção da condição de *start* significando por isso o começo de uma nova *frame*. Quando isto se verifica, são reiniciados os dois contadores base: o contador de bits - responsável por contar o número de bits recebidos, para a deteção do *byte* completo; contador de *bytes* – responsável pela organização da *frame* e detetar quando a mesma está completa e vê o seu incremento a cada oito contagens do contador de bits.

Se o estado anterior era 0, *low*, então significa que houve uma passagem para *high* do pino de SDA, causando uma condição de stop. Neste caso é atualizada variável "estado" anterior e tudo se repete novamente até ser detetada a condição de *start*.

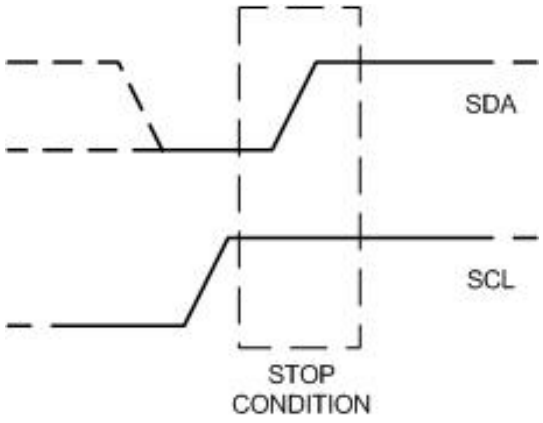


Figura 41 – Condição de stop, adaptado de [59]

Pode ainda o pino de *clock* passar a *low state* sem o pino de dados alterar, não caindo em nenhum dos casos anterior (condições de *start* e de *stop*), neste caso trata-se da receção de um bit. O bit mantém-se durante todo o período *high* do *clock* e com a passagem deste a *low* o bit é assumido. No fundo o bit é o estado do pino SDA momentos antes de ser feita a transição *high-low* do pino de *clock*, SCL. Quando ela ocorre, foi transmitido um bit pelo barramento.

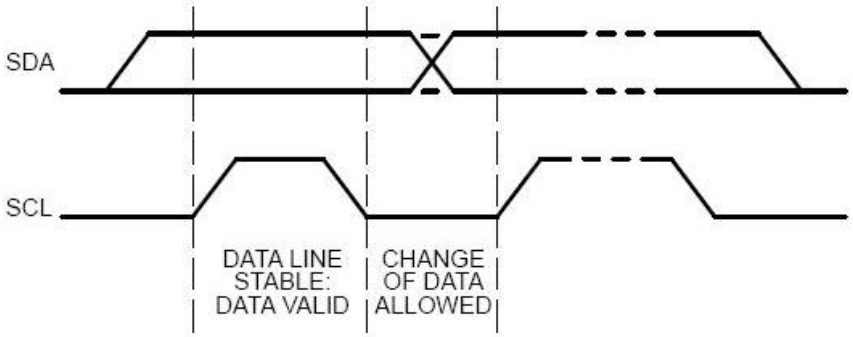


Figura 42 – Transferência de bit por I^2C, adaptado de [60]

Este bit que acaba de ser recebido é introduzido no *byte* em "construção". Em seguida é feito o *set* da *flag* "verif", confirmando a receção de um bit válido e podendo incrementar o contador de bits.

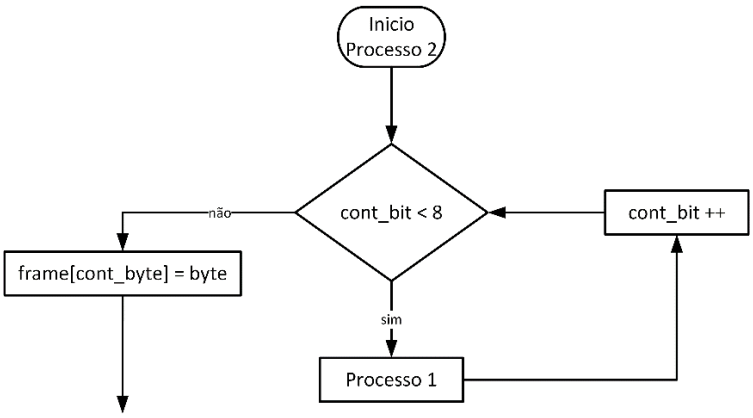


Figura 43 – Inicio do processo 2

O contador de bits vai incrementando até que sejam contados oito bits respeitando as condições anteriores. Nesse momento já se formou um *byte* com sucesso e este deve ser guardado junto da restante *frame* de dados recebidos (*frame[cont_byte]* = *byte*). Antes de todo o ciclo repetir e esperar por mais um *byte*, é esperado o 9º pulso no pino de SDA, correspondendo ao ACK da EEPROM.

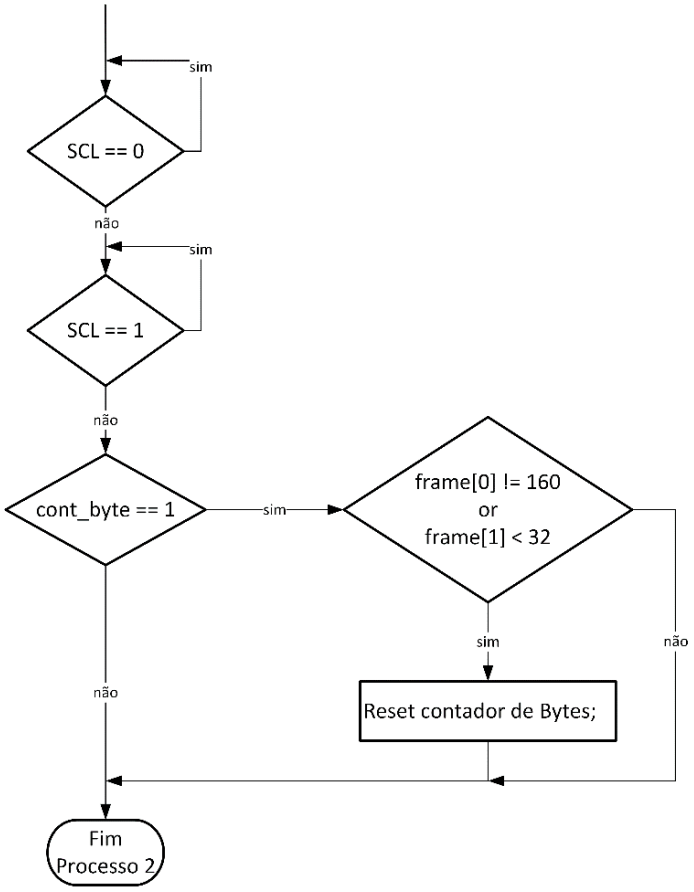


Figura 44 – Final do processo 2

Antes de o processo terminar, é verificado se já foram guardados dois *bytes* com sucesso (*cont_byte = 1*). No caso afirmativo uma nova verificação é feita com o intuito de descobrir se são os primeiros dois *bytes* da *frame* que se procura. Esses dois primeiros *bytes* devem corresponder ao comando de escrita (decimal "160") para uma posição sempre superior ou igual a 32 (tal como verificado no subcapítulo 3.2). Se estas condições não forem cumpridas significa que a *frame* que está a ser recebida de momento não corresponde à esperada, por isso a contagem de *bytes* é reiniciada e repete tudo novamente. Por outro lado, se os *bytes* estiverem certos (os dois primeiros), podem armazenar-se os *bytes* que se forem formando, pois esta é a *frame* certa: escrita para a EEPROM da medição mais recente.

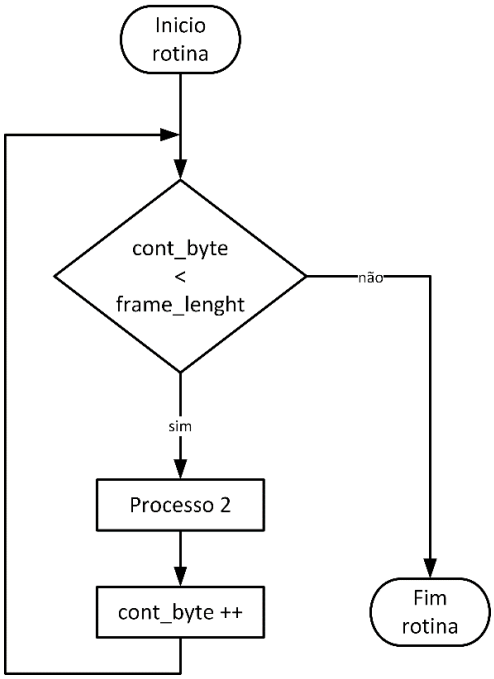


Figura 45 – Rotina i2c_frame

A *frame* procurada é composta por dez *bytes* (*frame_lenght*) e assim que, pelas condições anteriores, forem apurados sequencialmente dez *bytes* a *frame* está completa. Nesta fase pode proceder-se à recolha dos valores da medição de pressão arterial. O vetor *frame[]* (Figura 43) terá na sua posição 2 (3º elemento pois o vetor inicia com a posição zero) a pressão sistólica, na posição 3 a pressão diastólica e na posição 4 o número de batimentos cardíacos por minuto.

O algoritmo apresentado neste capítulo está totalmente pronto para implementação e funcional. O mesmo foi testado intensivamente com recurso ao microcontrolador PIC18LF25K22, mas cujo funcionamento será mostrado no capítulo 6 com o protótipo todo funcional, pois só desta forma se consegue transparecer a sua correta execução. A escolha do MCU foi uma boa aposta, pois somente graças à possibilidade do seu *clock* principal passar a 64MHz, core a 16MIPS, se conseguiu rapidez de análise suficiente para capturar as alterações de flancos próprias do I^2C. Embora a frequência de comunicação no barramento I^2C seja de 100KHz, fazer a deteção de flancos, proceder à construção do *byte*, identificar condições, etc,

requer uma capacidade e rapidez de processamento bem acima dessa frequência. Todos os testes realizados com frequência de *clock* abaixo desta mostraram-se infrutíferos.

Assim finda toda a fase ligada ao processo de análise, interpretação e aproveitamento dos dados recolhidos com medidor de pressão arterial Sanitas SBC 25/1. Nesta secção procurou-se não só entender quais são os princípios base para o funcionamento de um medidor deste tipo, como aplicar todo esse conhecimento adquirido a um caso prático. Partir de uma suspeita, analisar o equipamento, modificá-lo, interpretar os dados obtidos e conseguir entender o significado de cada um deles é sem dúvida um resultado positivo e tem de ser considerado um sucesso.

4. *Interface* de comunicação *wireless*

4.1. Módulo *Wi-Fi* e a ligação à IoT

O módulo *Wi-Fi* utilizado para tornar este medidor de pressão arterial, parte da IoT, é centrado no microcontrolador ESP8266[61].

O ESP8266 é um microcontrolador do fabricante chinês Espressif. Este inclui capacidades de *Wi-Fi*, com os periféricos internos necessários: interruptor de antena, balun (para casamento de impedância) e conversores de gestão de energia. Este integrado é relativamente recente e veio trazer uma enorme revolução na área da IoT, *Internet of Things*. Surgiu pela primeira vez em agosto de 2014 com o módulo ESP-01 (Figura 46). Esta pequena placa foi desenvolvida para que qualquer microcontrolador (capaz de suportar comunicação USART[6]) se pudesse conectar com uma rede *Wi-Fi* e fazer ligações TCP/IP recorrendo a comandos AT[62][63].

Da mesma forma que acontece com outros avanços chineses, no momento do lançamento não existia praticamente nenhuma documentação em inglês sobre o chip e/ou o módulo, assim como um descritivo dos comandos que aceitava.

O seu preço reduzido, a que os fabricantes asiáticos já nos habituaram, e o facto de não haver muitos componentes externos ao módulo, fez com que a sua aplicação ficasse a um custo muito reduzido. Isto atraiu as atenções da comunidade mundial de "*makers*"[7][64] que começaram a explorar o módulo, o seu *software* e a traduzir a documentação existente em chinês.

No final de outubro de 2014 o fabricante, Espressif, lançou um *kit* de desenvolvimento de *software* (SDK) que permitia a programação do integrado e, desta forma, fosse eliminada a necessidade de um microcontrolador externo. Esta possibilidade veio alargar ainda mais as hipóteses de aplicabilidade além de reduzir custos, pois um sistema simples de I/O podia ser implementado com IoT sem recorrer a qualquer microcontrolador externo[62][63].

Neste último ano, desde o surgimento do ESP8266, já foram criadas treze versões do módulo. Essas versões têm em comum o microcontrolador ESP8266 mas possuem diversos tamanhos, variedades de antenas, *pinouts* e configurações de modo a satisfazer necessidades que foram surgindo. Todos eles são de preço reduzido, mas em geral o ESP-01 continua a ser o mais barato, fazendo dele o mais popular de todos. A elevada quantidade produzida, apesar de o ESP-01 ter

[6] Formato padrão para comunicação de dados de forma serial.

[7] O movimento *Maker* é uma extensão da cultura Faça-Você-Mesmo (ou em inglês, *Do It Yourself*, DIY).

surgido há cerca de um ano, fez com que facilmente se consiga comprar este módulo por menos de cinco euros. A comunidade de *makers* já o considera a revolução *low cost* da IoT.

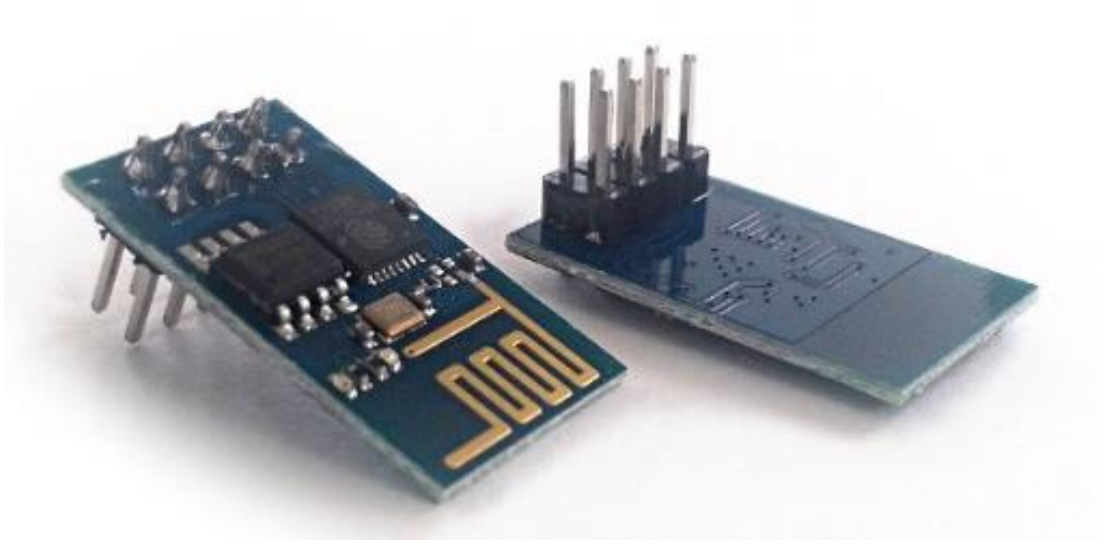


Figura 46 – Módulo *Wi-Fi* ESP-01, adaptado de [65]

A lista seguinte enumera algumas características e especificações relativas ao módulo ESP-01[66][67].

- Suporta protocolo 802.11b/g/n
- Comunicação via serial/USART com *baud rate* de 115200bps
- Antena impressa própria para a banda do *Wi-Fi*
- Capacidade de cobertura até 360m
- Saída de RF a +19.5dBm no modo 802.11b
- Capaz de criar um *Access Point*
- Capaz de criar um servidor *Web*
- Tensão de alimentação de 3.3V, com tolerância até 3.6V
- Consumo regular: ~70mA
- Consumo de pico: ~300mA
- Consumo em modo *power down*: <10µA
- Suporta WPA e WPA2, como modos de segurança em *Wi-Fi*
- Dimensões do módulo: 24.75mm x 14.5mm

O módulo ESP-01 possui um *header* duplo 2.54mm, com um total de oito pinos.

Os pinos estão identificados na Figura 47 e as suas ligações serão descritas em seguida juntamente com algumas notas.

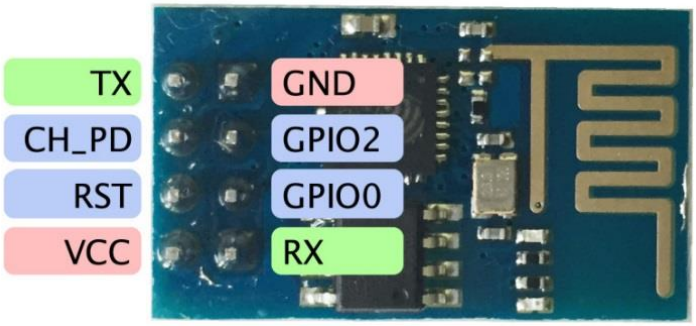


Figura 47 – Identificação do pinout do módulo ESP-01, adaptado de [68]

- **VCC** – pino de alimentação positiva deve ser ligado a uma tensão estável de +3.3V e com disponibilidade de fornecer até 300mA. Condensadores cerâmicos e de baixo ESR (resistência série equivalente) junto deste pino são essenciais de modo a garantir o menor *ripple* possível.
- **GND** – pino de massa/terra deve ser ligado ao comum do circuito.
- **TXD** – TX data comunicação serial a 3.3V.
- **RXD** – RX data comunicação serial a 3.3V.
- **CH_PD** – *Chip Power Down*, quando colocado em *low state* toda a atividade do módulo cessa.
- **GPIO0** – Pino de controlo possível usando o SDK, configurável como GPIO.
- **GPIO2** – Pino de controlo possível usando o SDK, configurável como GPIO.
- **RST** – pino de *reset*, quando colocado em *low state* executa o *reset* a todas as configurações do módulo.

Visto que existe a possibilidade de modificar o *firmware* de entre toda a panóplia de versões que começam a ser disponibilizadas, é importante citar que a versão do *firmware* do módulo utilizado neste trabalho é: versão de *bootloader* 0.9.2.4 com versão de comandos AT 0.21.0.0.

Os comandos de Hayes ou, posteriormente chamados de comandos AT foram criados em 1981 por Dennis Hayes, para resolver um problema de comunicação entre *modems* já produzidos. A sigla AT é uma abreviatura de "*attention*", atenção. Originalmente a sua aplicação tinha como principal alvo os *modems*, mas com o avançar dos anos a lista de comandos foi crescendo e adaptando-se a novos usos e acabou por se tornar bastante comum.

Todas as linhas de comando começam por "AT" e podem ser seguidas de mais caracteres, se forem comandos básicos, ou caso sejam comandos estendidos, é seguido pelo símbolo "+".

Estes comandos são muito utilizados nos dias de hoje em comunicações *modem-based*. Pode ser encontrado em módulos *Wi-Fi*, como o ESP-01, módulos rádio, módulos de *Bluetooth* e em *modems* GSM/GPRS.

4.2. Comunicação com o ESP-01

A comunicação entre o MCU e o módulo ESP-01 é feita por USART recorrendo a um protocolo de comandos AT[8], com um baud-rate de 115200bps (bits por segundo).

No final de cada sequência de envio do microcontrolador para o ESP-01 os caracteres especiais "\r", *carriage return*, e "\n", *new line*, devem ser enviados por esta ordem. É desta forma que o módulo identifica o final da comunicação e procede com a ação, mesmo que essa seja uma mensagem de erro enviada em retorno para o microcontrolador.

Relativamente à receção de caracteres pelo MCU, o módulo ESP-01, por defeito, envia automaticamente o eco (repetição) dos comandos recebidos. O exemplo disso pode ser visto na Figura 48, recorrendo uma vez mais ao analisador lógico.

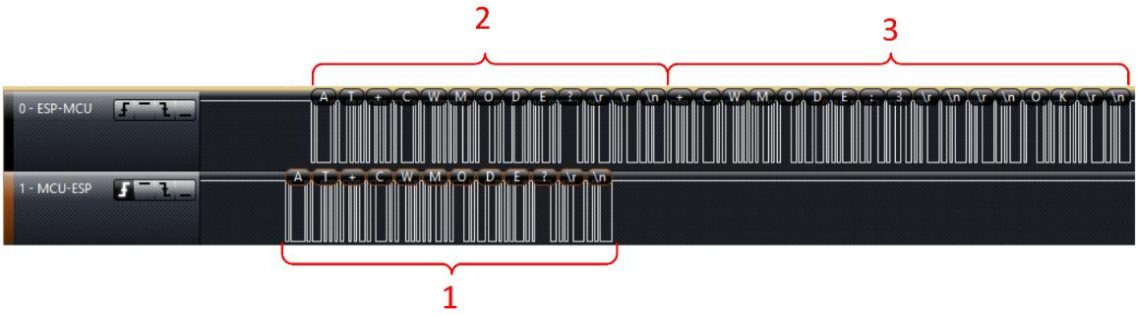


Figura 48 – Exemplo de resposta com eco do ESP-01

Na imagem são identificados três segmentos distintos:

1 – comando enviado do MCU para o módulo ESP-01;

2 – eco do comando enviado;

3 – resposta do ESP ao comando enviado.

[8] Consultar lista de comandos no anexo 8.2.

Os comandos repetidos no eco possuem um tamanho variável assim como as respostas relativas a cada comando, são diferentes entre si. Como apresentado na Figura 48, a sequência dos caracteres especiais "\r\n" surgem várias vezes seguidas, não sendo por isso um bom indicador de fim de mensagem. Torna-se, por esta razão, complexo saber quando o módulo acaba de transmitir a mensagem, qual a resposta enviada e qual o seu significado.

Após a análise de várias hipóteses testadas, o método descrito em seguida é o responsável pela comunicação, essencialmente, do módulo ESP-01 para o MCU.

Quando um *byte* é recebido pelo periférico da USART do MCU é provocada uma interrupção. Na Rotina de Serviço à Interrupção, ISR[9], o *byte* acabado de receber é guardado num *buffer* circular de oito *bytes*. A cada *byte* recebido o *buffer head* é incrementado, guardando na nova posição o *byte* chegado (*RCREG*). Quando o *buffer head* (*eusartRxHead*) atinge a posição 7 (8 *bytes* guardados, 0 a 7) a próxima posição retorna ao 0.

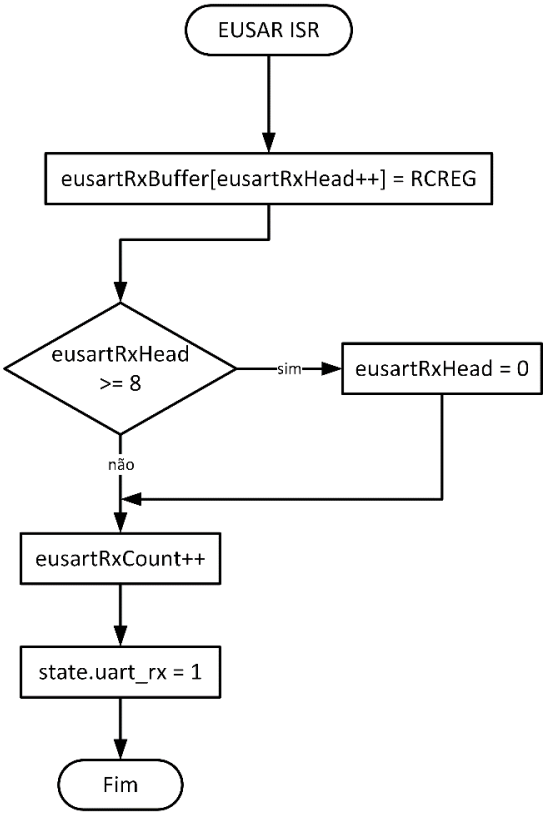


Figura 49 – Rotina de interrupção da USART

[9] Em inglês, Interrupt Service Routine, ISR.

No final da ISR é feito o *set* da *flag state.uart_rx*, simbolizando que ocorreu uma interrupção e um *byte* foi armazenado com sucesso. Mais tarde esta *flag* será usada numa máquina de estados que controlará todo este processo.

A cada *byte* recebido é também incrementada uma variável (*eusartRxCount*) que regista o total de *bytes* recebidos, desde que se habilita a receção. Esta variável é útil para saber como organizar o *buffer* circular num vetor linear com os valores por ordem cronológica. Se ela for inferior a oito o *buffer* circular ainda possui os valores por ordem de chegada, mas se for superior a oito é necessário fazer o ajuste e organizar da posição do *buffer head* até à posição 7 e da posição 0 ao *buffer head*.

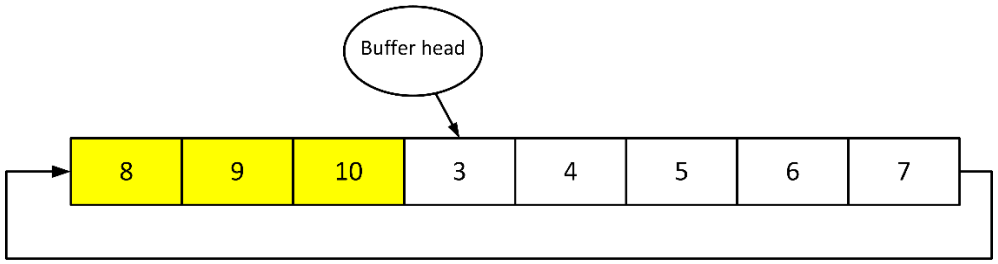


Figura 50 – Buffer circular

Essa organização temporal, por ordem de chegada, dos *bytes* é fundamental para que mais à frente se possa encontrar a sequência esperada de caracteres, que é a resposta do ESP-01. Para realizar essa tarefa foi criada a função *usart_last_8bytes(*rcv)* com o seu diagrama explicativo na Figura 51. Esta aceita o apontador **rcv* para a primeira posição de um vetor com oito elementos e onde ficarão, terminado o processo, organizados cronologicamente os *bytes* recebidos.

No caso de o número de *bytes* recebidos ainda não ser igual ou superior ao valor 8, dado que o *buffer head* assume valor 0 sempre que inicia nova receção de dados, significa que os *bytes* no buffer circular estão por ordem cronológica. Então, neste caso, basta simplesmente fazer a cópia para um vetor usado na deteção de sequências de resposta (como por exemplo a resposta "OK").

A função *EUSART_Read(),* auxilia a leitura do buffer circular, devolvendo *byte* a *byte* o seu conteúdo, da posição 0 à posição 7.

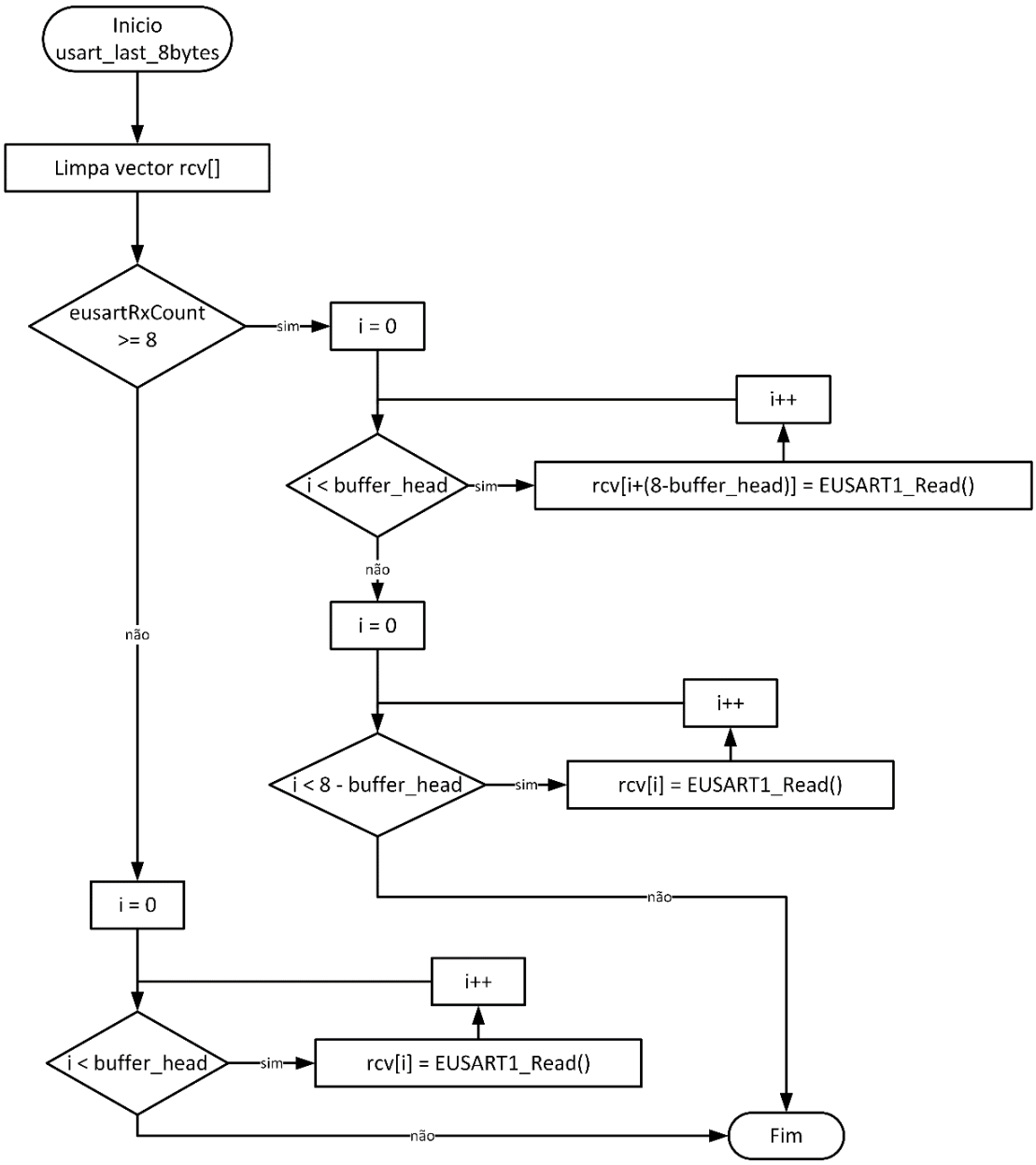


Figura 51 – Fluxograma da função *usart_last_8bytes*

No caso de já ter recebido oito ou mais *bytes* (*eusartRxCount* ≥ *8*) é necessário proceder à organização dos mesmos. Para isso são necessários dois ciclos, facilmente identificados no fluxograma. O primeiro começa por alinhar sequencialmente os *bytes* (do *buffer* circular) das posições abaixo do *buffer head* na sua devida posição do vetor *rcv[]*. De forma resumida, essa tarefa está ilustrada na Figura 52.

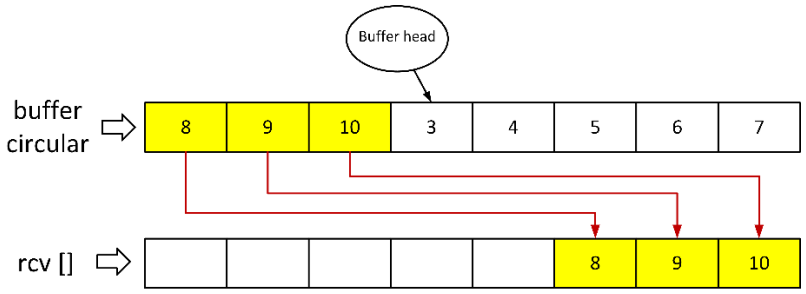


Figura 52 – 1ª organização do *buffer* circular

No segundo ciclo são alinhados os *bytes* que se encontram acima da posição do *buffer head*. Deste modo e terminando os dois ciclos, o vetor *rcv[]* já possui de forma cronologicamente ordenada os últimos oito *bytes* chegados à USART do MCU. Correspondendo a posição 0 ao *byte* mais antigo e a posição 7 ao último *byte* chegado.

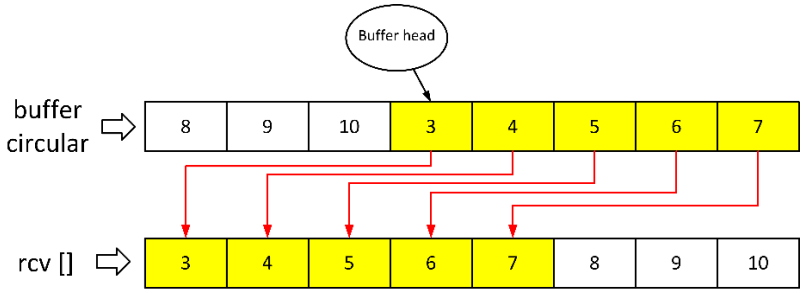


Figura 53 – 2ª organização do *buffer* circular

Até este ponto está descrita a forma como é implementado o *buffer* circular e como é feita a organização temporal dos *bytes* guardados no *buffer* circular num vetor de nome *rcv[]*. No entanto é necessário ainda encontrar uma forma de detetar quando o ESP-01 termina a sua comunicação e se pode prosseguir para a interpretação da resposta enviada.

Já foi mostrado anteriormente (Figura 48) que não existe nenhum caracter especifico que indique o final da transmissão. O método escolhido funciona tal como numa conversa entre duas pessoas. Uma pausa temporal ocorre no final da conversa e a outra pessoa sabe que a primeira terminou de falar. Adaptando à situação em estudo, se 10 milissegundos após a chegada do último *byte* (tempo resultante de vários testes e que garante não dar uma "conversa" por terminado estando esta ainda a meio) não for recebido mais nenhum, admite-se que o envio da resposta do ESP-01 para o MCU terminou, podendo avançar para o próximo estado.

Tudo o que já foi descrito, no que toca a comunicação com o ESP-01, funciona a comando de uma máquina de estados, planeada para que o programa nunca fique retido num ciclo interminável por erro.

Surge então a função *ESP_comm*, semelhante a uma máquina de estados, que foi desenvolvida para aceitar duas entradas por parte da rotina principal: uma *string* com o comando a ser enviado para o ESP-01 e a resposta esperada dele (continua nos últimos oito bytes recebidos). Nesta rotina começa por ser feito o *reset* do *buffer head* e ser limpa a *flag state.usart_rx*. Em seguida é enviada a *string* com o comando a ser executado pelo ESP-01 e é feita uma espera até receber o primeiro *byte*, que traz a resposta do módulo e que terá de ser interpretada. Com a chegada do primeiro *byte* a *flag state.timer3* é limpa, indicando que não ocorreu nenhuma interrupção do *timer* 3 e é feito o *start* desse mesmo *timer* (*TMR3_StartTimer*) para que ele comece a contar 10 milissegundos até à interrupção. O fluxograma da Figura 55 representa o que acontece depois de o *timer* ter sido colocado em funcionamento, assim como o ISR do *timer* 3.

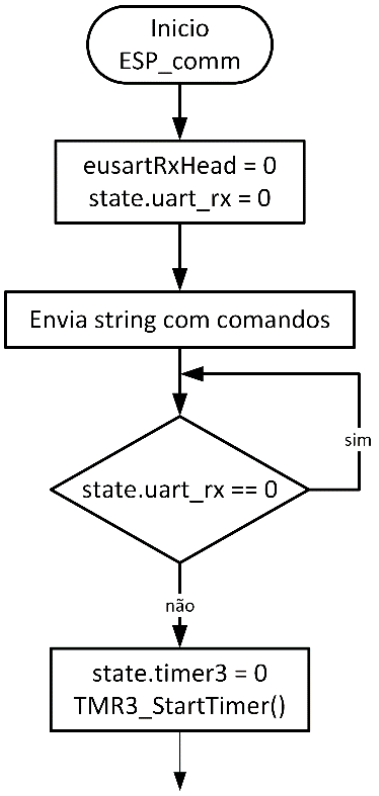


Figura 54 – Fluxograma da função *ESP_comm* (1ª parte)

A rotina verifica se surgiu um novo *byte* na USART (*state.uart_rx = 1*) ou se houve uma interrupção do *timer* 3 (*state.timer3 = 1*), sendo o programa retido em ciclo, até que algum dos dois aconteça. Se dentro dos 10 milissegundos for recebido um novo *byte*, a *flag* de estado *state.uart_rx* é limpa e é feito o *reload* do *timer* 3, que corresponde à reiniciação da contagem. Quando chegar ao fim a comunicação do módulo ESP-01, mais nenhum *byte* vai ser recebido e por isso não vai haver *reload* do *timer*. Assim nada o impede de ter o *overflow* e entrar na rotina de interrupção (Figura 55), onde é feito o *set* da *flag* de estado *state.timer3*.

Com a deteção do final da comunicação do ESP-01 e consequente alteração da *flag* de estado *state.timer3,* nada retém a rotina *ESP_comm* que pode prosseguir com a organização dos últimos oito *bytes* recebidos (presentes no *buffer* circular). Para isso recorre à função *usart_last_8bytes*, apresentada anteriormente.

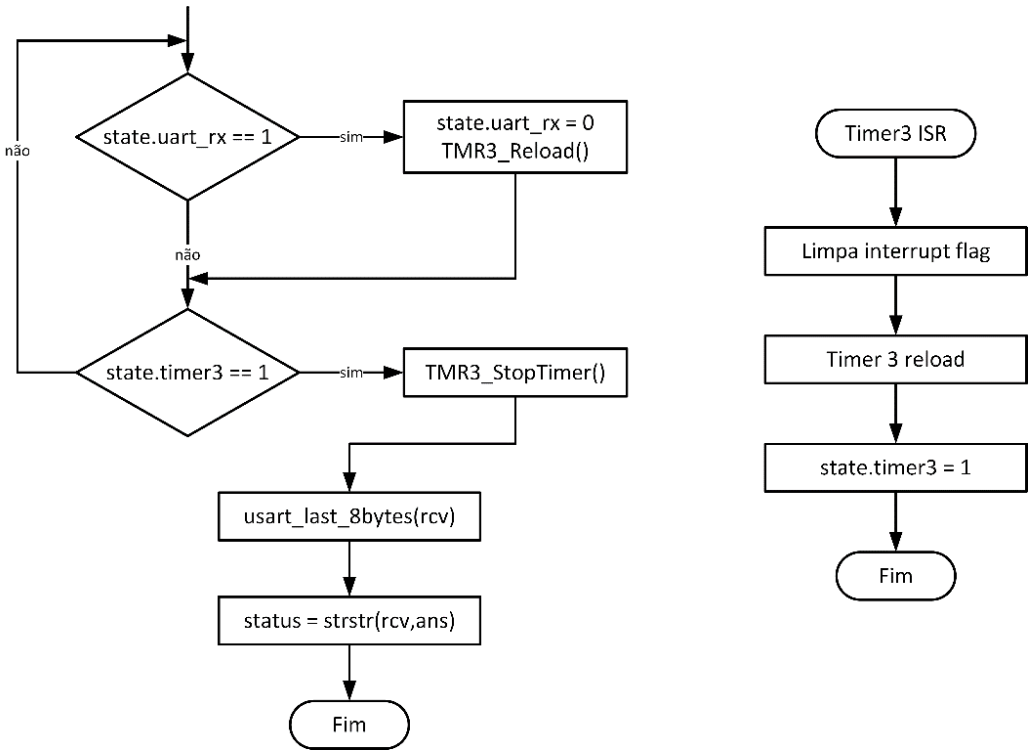


Figura 55 - Fluxograma da função *ESP_comm* (2ª parte) e *Timer* 3 ISR

Após ser realizada a organização cronológica dos *bytes* no vetor *rcv[]*, a resposta *ans* (*answer*) é procurada nos últimos oito *bytes* já organizados, de onde retorna a variável *status*.

Em suma, se *status* for diferente de zero, a resposta enviada pelo ESP-01 é a esperada e por isso pode prosseguir. Mas se, por outro lado, a variável *status* possuir o valor zero, significa que a resposta enviada pelo módulo não era a prevista. Conclui-se a ocorrência de um erro, pois algo não correu como previsto.

Estas são as funções base necessárias para uma correta comunicação com o módulo ESP-01. É sobre elas que os comandos AT e o próprio envio de dados vão ser executados. No entanto, para que os dados possam ser colocados na IoT, tem de haver uma plataforma configurada e pronta para os receber e processar.

4.3. Plataforma ThingSpeak

Apesar que a IoT ainda estar a tomar forma, o interesse pelo seu desenvolvimento está a ganhar um crescimento incrível. São várias as plataformas *online* dedicadas à IoT para a criação de aplicações e outros, que disponibilizem e tratem os dados fornecidos pelas "coisas". Uma dessas plataformas de aplicações de IoT que oferece uma ampla variedade de análises, capacidade de monitorização e de contra-ação é o ThingSpeak.

O ThingSpeak permite a recolha de dados em tempo real, visualização dos dados recolhidos na forma de gráficos, capacidade de criar *plugins* e aplicações para colaborar com serviços *web*, redes sociais e outros interfaces de programação de aplicações (API's) e é ainda um serviço *opensource*. Recentemente o ThingSpeak passou a ter suporte de computação numérica dada pelo MATLAB da MathWorks.

Esta plataforma permite que o utilizador crie vários canais, sendo que cada um desses canais tem um máximo de oito campos (*fields*). No caso deste trabalho, o canal receberá dados de apenas um medidor, mas nesse canal serão usados três campos para guardar dados: no campo 1 a pressão sistólica, no campo 2 a pressão diastólica e no campo 3 os batimentos cardíacos por minuto.

Para criar um canal no ThingSpeak basta seguir os seguintes passos (Figura 56):

1 – criar uma conta gratuita em "https://thingspeak.com";

2 – ir a "*Channels*" e clicar em "*New Channel*";

3 – seguir todos os passos de configuração apresentados e no final clicar em "*Save Channel*";

4 – obter a chave de API;

5 – o canal está pronto a receber dados nos campos configurados, através do URL (por exemplo campo 1 com valor 7).

https://api.thingspeak.com/update?api_key=CHANNEL_API_KEY**&**field1**=7**

O ThingSpeak permite que o seu utilizador configure o seu canal a inúmeras formas, principalmente visuais, no que toca à disposição de gráficos, tipo de gráficos usados, organização lógica dos dados, entre outros. Tratando-se de uma plataforma simples de usar e altamente configurável torna-se ideal para funcionar como suporte "interativo" neste trabalho.

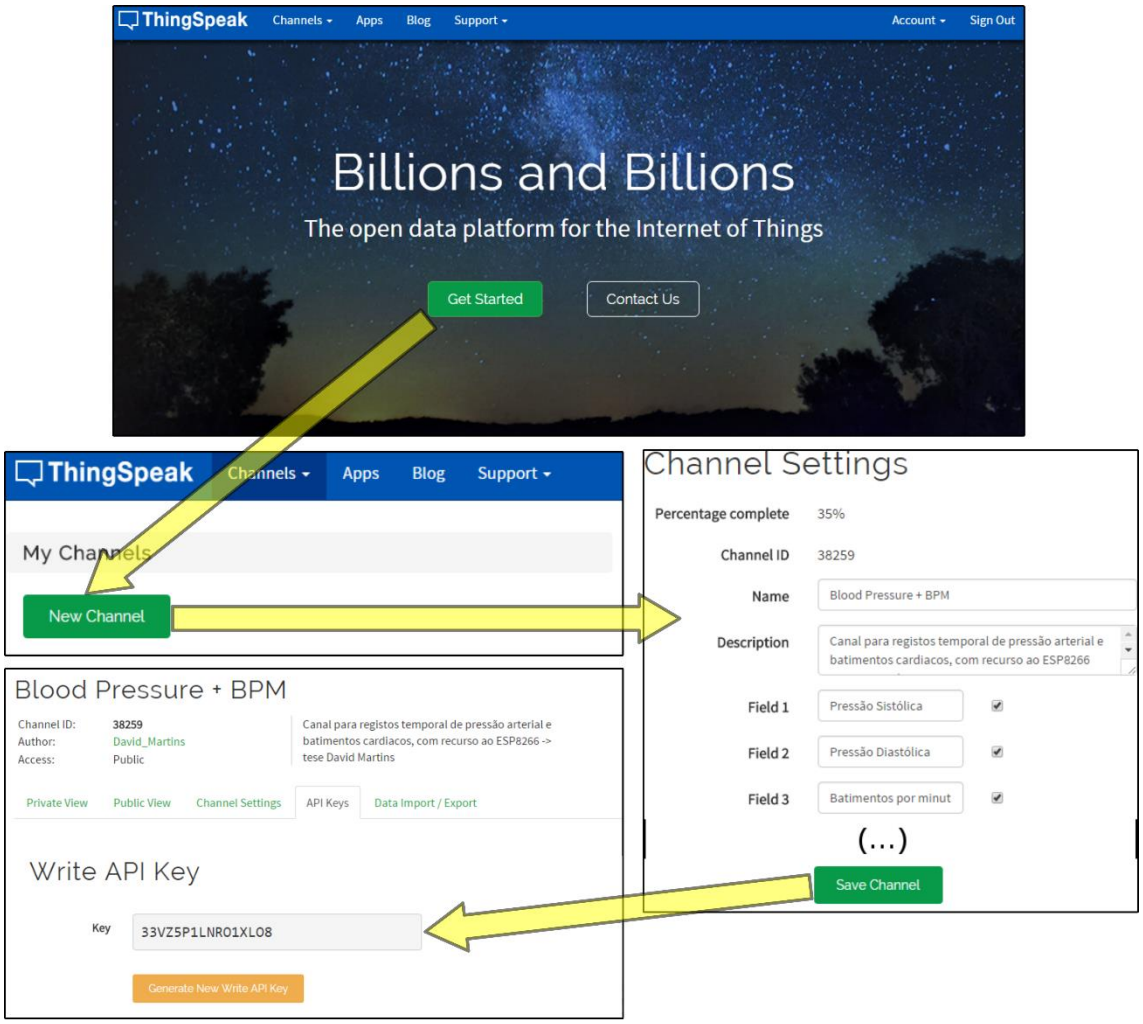


Figura 56 – Passos para criar um canal na plataforma ThingSpeak

A criação e configuração do canal e dos seus campos são fundamentais para obter a API *key* e deixar a plataforma pronta a receber. Estes dados são essenciais para que se possa dar início à configuração do módulo ESP-01, assim como a sua ligação à plataforma ThingSpeak.

4.4. Processo de configuração, comandos AT e colocação de dados online

Todas as funções descritas até ao momento podem ser consideradas como funções de *low level*. Em seguida será feita a descrição de toda a rotina de comunicação com o ESP-01, função *ESP_routine*. Esta utiliza as funções base e cria um nível de abstração maior deixando, de certa forma, o nível do *bit field*. Ela aceita os valores para os vários campos (*fields* do ThingSpeak) e se algum erro ocorrer é devolvido o número correspondente à etapa onde o mesmo aconteceu.

Para poupar o máximo de energia possível, o módulo ESP-01 é colocado em *power down* sempre que o mesmo não está em utilização. O pino CH_PD é por isso mantido em *low state* e para habilitar o módulo é necessário colocar o pino em *high state*. Imediatamente após ser habilitado, o módulo começa a realizar comunicação mas apenas são enviados caracteres aleatórios, isto porque o microcontrolador ESP8266 está em arranque e o *clock* não estabilizou ainda. O módulo só estará pronto a funcionar quando a sequência "ready" for enviada.

```
 ,n[1F]†$û!‹ƒƒA4ž,"*[18]'¦
£ÿ[14]_¤*Á
C4ž[02][1C]
[Vendor:www.ai-thinker.com Version:0.9.2.4]

ready
```

Figura 57 – Arranque do módulo ESP-01

Nesta fase inicial é usada uma técnica semelhante à já descrita para detetar o fim de comunicação, mas ao invés de aguardar 10 milissegundos pela chegada de um novo *byte* aguarda-se um período de 400 milissegundos. Isto porque, como se pode ver na Figura 58, existe um longo intervalo de tempo em que o ESP8266 se encontra na fase de arranque e deixa de enviar *bytes* pela USART. Esse tempo ronda os 250 milissegundos, mas com a noção de que poderá variar um pouco por excesso, a espera foi alongada para 400 milissegundos.

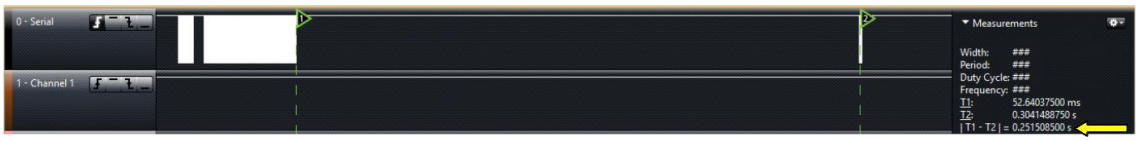


Figura 58 – Período de arranque do módulo ESP-01

Passado esse período de tempo desde o último *byte* recebido considera-se que a comunicação do ESP-01 terminou e, recorrendo às mesmas técnicas já descritas, verifica-se se nos últimos

oito *bytes* está presente a sequência "ready". Se a sequência não estiver presente, a rotina fica por aqui e a função retorna o valor de erro 1 para o *main program*.

Após confirmar o arranque do módulo é realizada a seguinte sequência de processos, que têm por base o recurso à função *ESP_comm*:

- *status = ESP_comm("AT\r\n", "OK");*

Enviado o comando AT para testar a prontidão e correto funcionamento do módulo.

É esperado como resposta "OK". Se *status* for diferente de 0, a resposta foi recebida e o processo continua, caso contrário a função *ESP_routine* termina retornando erro 2.

- *status = ESP_comm("ATE0\r\n", "OK");*

Enviado o comando ATE0 para descativar o eco dos comandos enviados para o módulo.

É esperado como resposta "OK". Se *status* for diferente de 0 o processo continua, caso contrário a função *ESP_routine* termina retornando erro 3.

- *status = ESP_comm("AT+CWMODE=1\r\n", "OK");*

Enviado o comando AT+CWMODE=1 para configurar o modo *Wi-Fi* para *station mode*, que é o modo padrão.

É esperado como resposta "OK". Se *status* for diferente de 0 o processo continua, caso contrário a função *ESP_routine* termina retornando erro 4.

- *status = ESP_comm("AT+CWJAP=\"toca_suricata\",\"59T5LDTN77R\"\r\n", "OK");*

Enviado o comando **AT+CWJAP="toca_suricata","59T5LDTN77R",** para que o módulo se ligue a um *access point* de *Wi-Fi*. Neste caso tenta ligação ao AP **toca_suricata**, com a palavra-chave (WPA2) **59T5LDTN77R.**

É esperado como resposta "OK", caso se consiga ligar com sucesso. Se *status* for diferente de 0 o processo continua, caso contrário a função *ESP_routine* termina retornando erro 5.

- *status = ESP_comm("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n", "OK");*

Enviado o comando **AT+CIPSTART="TCP","184.106.153.149",80** pede-se ao módulo que estabeleça uma ligação TCP com o servidor do ThingSpeak (identificado com o IP 184.106.153.149) utilizando o *remote port* 80, ou seja, protocolo de transferência de HiperTexto - HTTP[69].

É esperado como resposta "OK", caso se consiga ligar com sucesso. Se *status* for diferente de 0 o processo continua, caso contrário a função *ESP_rooutine* termina retornando erro 6.

- *size = sprintf(update_str, "GET/update?key=33VZ5P1LNRO1XLO8&field1=%u&field2=%u&field3=%u\r\n", p_sist, p_diast, bpm);*

Constrói uma *string*, *update_str*, com o comando **GET/update?key=33VZ5P1LNRO1XLO8** e com os valores a colocar nos três *fields*, sejam pressão sistólica (field1 = p_sist), diastólica (field2 = p_diast) e batimentos por minuto (field3 = bpm). A sequência "33VZ5P1LNRO1XLO8", corresponde à API *key* do canal criado (ver Figura 56) e é com esta chave que a ligação ao canal é permitida.

A variável resultante *size*, recebe o valor do número de caracteres contidos na *string update_str*.

- *sprintf(size_str, "AT+CIPSEND=%u\r\n", size);*

Constrói uma *string*, *size_str*, com o comando **AT+CIPSEND** igualado à variável *size* anterior, isto para indicar ao módulo ESP-01 quantos caracteres vai receber e, por sua vez, quantos caracteres vai usar na ligação HTTP.

- *status = ESP_comm(size_str, ">");*

Envia a *string size_str* já atualizada com a variável *size*.

É esperado como resposta ">" (*prompt*), caso o módulo esteja pronto a receber o endereço para ligação. Se *status* for igual a 0, a função *ESP_routine* termina retornando o erro 7.

- *status = ESP_comm(update_str, "OK");*

Envia a *string update_str* já atualizada com os valores de pressão sistólica (*field* 1), diastólica (*field* 2) e batimentos por minuto (*field* 3).

É esperado como resposta "OK" confirmando que os dados foram recebidos e vão ser enviados para o servidor do ThingSpeak. Se *status* for diferente de 0 o processo termina não retornando qualquer erro (*return* 0), caso contrário a função *ESP_routine* termina retornando erro 8.

A função *ESP_routine* é a função principal que gera toda a ordem de chamada para configurações e envio dos dados. Como entrada recebe três dados contendo os valores a enviar de pressão sistólica, diastólica e batimentos cardíacos e devolve uma variável de erro que indica se ocorreu ou não algum erro no processo e, em caso afirmativo, onde ele ocorreu.

Para melhor visualizar toda função *ESP_routine*, o fluxograma da Figura 59 contém o caminho principal assim como a listagem de todos os comandos apresentados anteriormente.

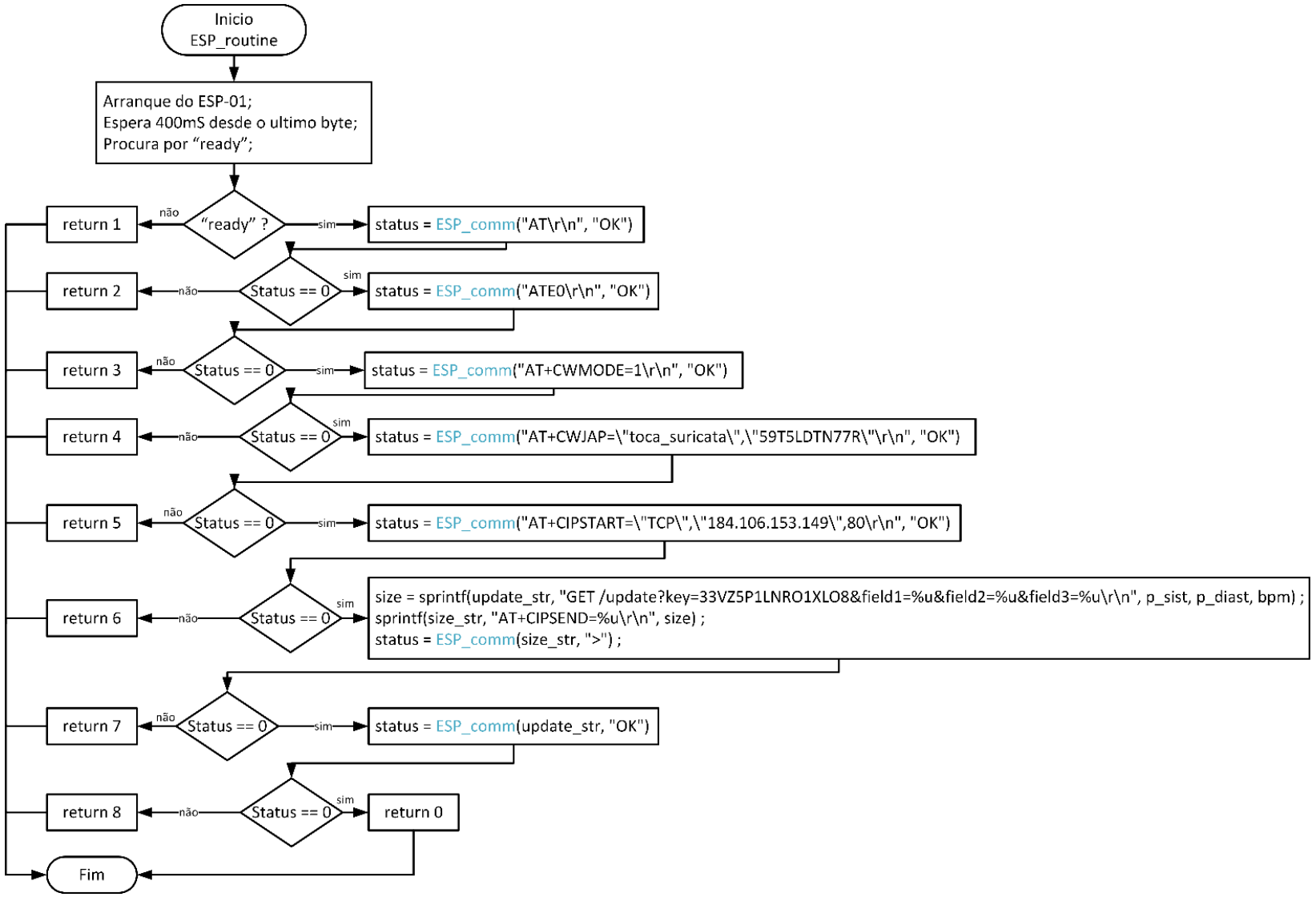


Figura 59 – Fluxograma da função *ESP_routine*

5. Descrição geral do protótipo

Descrever o programa *main* é descrever o funcionamento do protótipo. Praticamente tudo o que acontece tem de estar, de certa forma, ligado a ele. Por isso, esta descrição geral relata também o que decorre ao longo da função principal *main*.

Desde o momento em que o MCU é alimentado e após fazer toda a inicialização dos seus periféricos internos é feita uma pausa de três segundos, ao longo da qual se faz piscar um LED vermelho, a cada 500 milissegundos. Este período não tem nenhum significado específico apenas simboliza o arranque do todo o processo.

Terminado esse tempo, o pino *enable* do *boost converter* passa para *high state*, de modo a habilitar a sua entrada em funcionamento. Por segurança é feita uma larga espera de 50 milissegundos, de modo a garantir que o conversor entrou em funcionamento, carregou todos os condensadores e está estabilizado.

Seguidamente é iniciado o processo de medição de pressão arterial e para isso é necessário simular que o botão "ON/OFF" foi pressionado. Então é feita uma transição para *low state* do pino ligado ao medidor durante 10 milissegundos, período suficiente para que o medidor reconheça a alteração de estado e dê início à operação. Passado esse tempo o pino é colocado em *tri-state*, como *input*, para não causar qualquer tipo de influência.

A partir deste momento o processo de medição de pressão arterial fica a cargo do medidor Sanitas SBC 25/1. Para simbolizar o seu início de operação, um pino do MCU passará para *high state* acendendo um LED azul ligado a ele.

A função *i2c_frame* (descrita no subcapítulo 3.4) é chamada para se encarregar da procura e cópia da *frame* certa que transporta os dados resultantes da medição da pressão arterial. Quando essa *frame* é identificada com sucesso o LED azul desliga, simbolizando desta vez que a *frame* já foi recebida e que a operação do medidor de pressão termina por aqui.

O resultado da medição fica ainda no ecrã do aparelho para que possa dar algum *feedback* ao utilizador, é feita uma pausa de 3 segundos até iniciar o envio dos dados, mantendo sempre o ecrã ligado. A pausa tem como objetivo aumentar o período de *feedback*.

Passado esse tempo, é ligado novamente o LED azul marcando o começo de uma nova etapa. Desta vez referida ao início de operação do módulo ESP-01 e portanto o início das operações de envio.

Em seguida é chamada a função *ESP_routine*, responsável pela configuração do módulo, ligação ao ThingSpeak e envio dos dados recolhidos das pressões sistólica, diastólica e batimentos

cardíacos por minuto. Tal como descrito no subcapítulo 4.4, esta função devolve um indicador de erro se algo não expectável acontecer. Se acontecer algum erro o LED vermelho liga durante um período de 500 milissegundos e é feito o *reset* do módulo ESP-01. Depois disso é feita uma nova tentativa de arrancar com o módulo e fazer o envio dos dados. Se em três tentativas sequenciais ocorrer algum erro, o medidor, que ainda tem o ecrã ligado com os valores da última medição, é finalmente desligado. São também desligados o *boost converter* e o LED azul, que marcava o estado da operação. Após isto o LED vermelho liga de forma permanente, sendo sinónimo de impossibilidade em exportar os dados para IoT. No entanto, se das três tentativas, alguma der certo o medidor é desligado, assim como o *boost converter* e o LED azul. Marcando o sucesso do procedimento, o LED vermelho pisca, interminavelmente, a cada 500 milissegundos.

Esta é a descrição geral do programa *main* e a partir do qual todas as funções construídas são chamadas. O descrito corresponde apenas a um ciclo de operações, mas cujo objetivo é ser repetido em intervalos predefinidos de 20 minutos[5][33], segundo as recomendações dos especialistas.

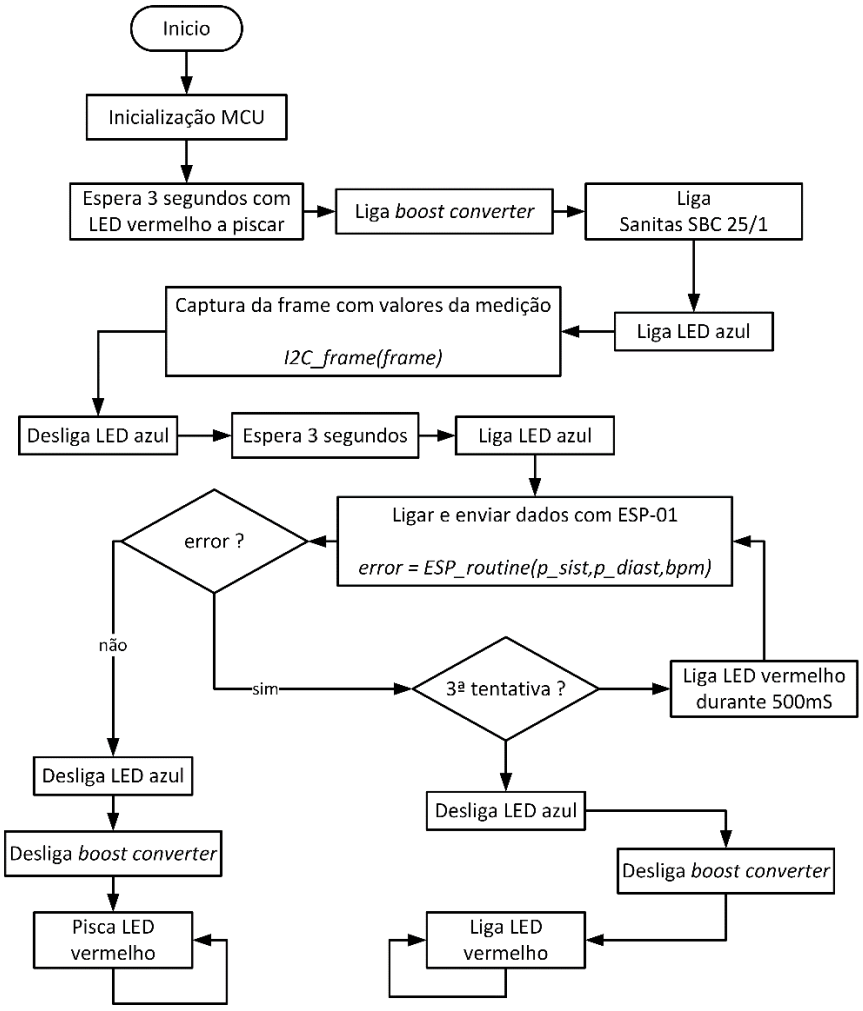


Figura 60 – Fluxograma do programa *main*

5.1. Descrição do circuito

De modo a satisfazer os objetivos iniciais, foi projetado e construído um circuito de protótipo. Este tem de corresponder às especificações de todas as partes e unir tudo num só sistema. O diagrama da Figura 61 representa um *overview* global de todas as partes constituintes daquilo a que se chegou como protótipo funcional. Cada uma delas tem a sua razão de existir e de fazer parte desta montagem, por isso o seguinte texto descritivo incluirá essas justificações.

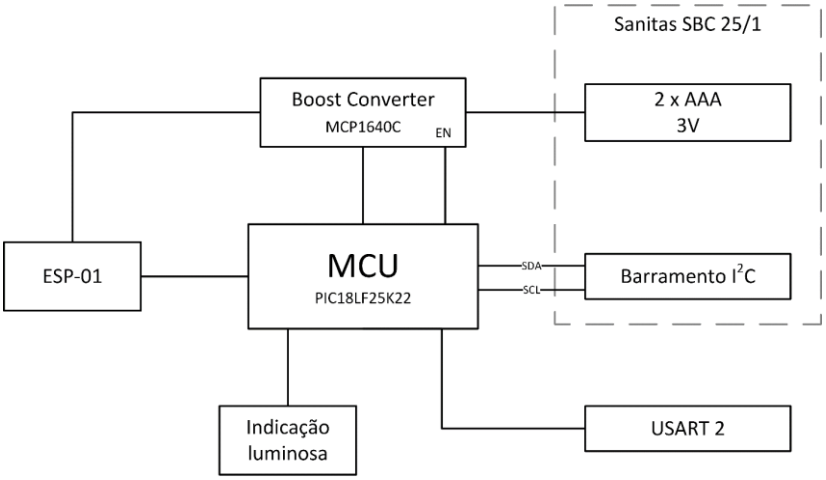


Figura 61 - Diagrama de blocos do protótipo

O medidor utilizado, Sanitas SBC 25/1, possui uma alimentação fornecida por duas baterias tipo AAA, que aquando em bom estado, novas, fornecem um total de 3V (2 x 1.5V). Com o desgaste a tensão nas baterias vai descendo e poderá ter até algum *ripple* mais acentuado, de acordo com o arranque do motor. Essas depressões ocorrem devido à queda de tensão na resistência interna da bateria (resistência iónica), que pela Lei de Ohm, quanto maior a corrente, maior a queda de tensão também.

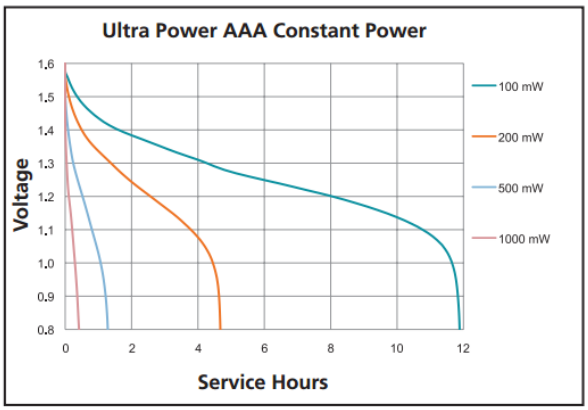


Figura 62 – Representação gráfica de desgaste de uma bateria AAA, adaptado de [70]

O microcontrolador utilizado nesta aplicação é o PIC18F25K22, que funciona com uma tensão de alimentação de até 1.8V, mantendo o oscilador interno a funcionar a uns incríveis 20MHz. No entanto um dos requisitos para realizar com sucesso o *sniffer* do barramento I^2C (resultante de experiências práticas) é a necessidade de subir o *clock* principal (do core) para 64MHz e, por isso, a alimentação nunca deverá ser inferior a 2.7V, como indicado graficamente no *datasheet* do MCU[55].

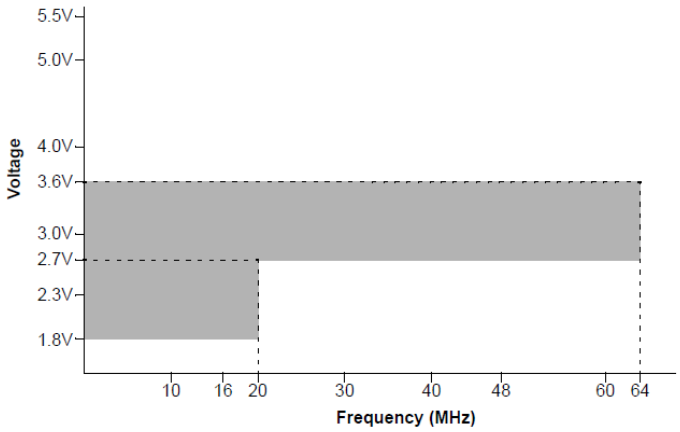


Figura 63 – Relação entre tensão e frequência máxima de *clock*, adaptado de [55]

Por outro lado, no que toca às especificações de alimentação, o ESP-01 tem como tensão ideal de funcionamento 3.3V.

A alimentação disponível é um problema por resolver. Para isso foi aplicado como solução um *boost converter*, o MCP1640C da Microchip[71]. Este conversor é pequeno, consegue entregar na saída até 800mA em pico, tem um *start-up* de 0.65V e ainda, no caso da versão C (a utilizada neste trabalho), capacidade de fazer o *input-output bypass*. Quando o pino de EN (*enable*) passa para *low state* o conversor deixa de funcionar, consumindo apenas um valor muito residual de corrente (<1μA), e faz o bypass da tensão de entrada, das baterias, para o pino de saída (pino Vout) que vai alimentar o restante circuito.

O pino de *enable* do MCP1640C é controlado pelo MCU, que durante todo o período de espera utilizará o cristal externo de 32.768 KHz como oscilador secundário e desliga o oscilador primário. Isto permite que o microcontrolador funcione praticamente até à descarga total das baterias, ou seja, tensão mínima de alimentação de 1.8V o que dá 0.9V por bateria. Analisando a Figura 62, quando a bateria baixar dos 0.9V mais de 98% da energia da mesma foi consumida.

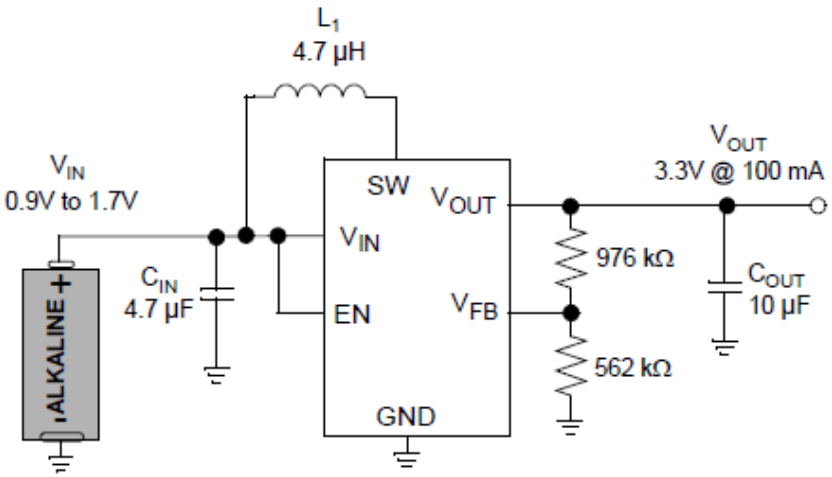


Figura 64 – Aplicação típica sugerida no *datasheet*, adaptado de [71]

A tensão mínima de alimentação do circuito tem por obrigação ser superior a 1.8V, tensão abaixo da qual o MCU deixa de funcionar corretamente. Quanto à tensão máxima que o circuito conversor irá estar sujeito pouco mais superior será que 3V.

O conversor irá ser desligado durante o período de espera, sendo apenas ligado para realizar a aquisição de dados da EEPROM e para fazer o envio dos dados com o ESP-01. Nestas condições o consumo de corrente saltará para valores acima de 10mA e durante o envio de dados para a rede rondará os 200mA. Com estas informações e olhando o da Figura 65 (admitindo o *Vin* intermédio de 2.5V, linha a verde), o rendimento do circuito conversor rondará os 90%.

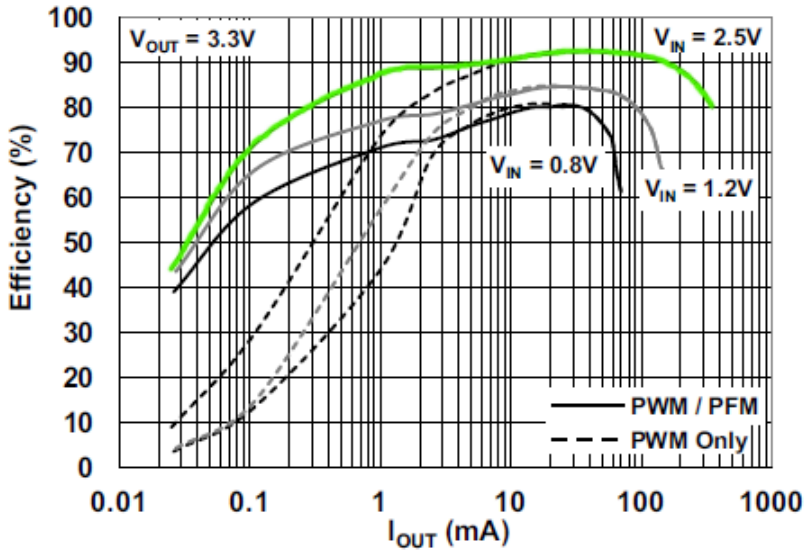


Figura 65 – Relação entre a eficiência e a corrente de saída, adaptado de [71]

As resistências de *feedback* utilizadas possuem os valores comerciais de 220kΩ e 120kΩ [10]. As duas compõem um divisor de tensão de feedback que deverá resultar em 1.21V e por isso o *boost converter* irá elevar a tensão *Vout* para 3.43V, valor que está perfeitamente dentro das especificações.

$$V_{FB} = \frac{R2 * V_{OUT}}{R1 + R2} \ (=)\ 1.21 = \frac{120k * V_{OUT}}{120k + 220k} \ (=)\ V_{OUT} = 3.43V$$

Após o pino de *Vout* foram utilizados condensadores de várias capacidades (100µF, 10µF, 1µF) de forma a construir um banco de condensadores capaz de responder a picos de corrente a diversas frequências.

Outros condensadores de 1µF e 100nF foram colocados o mais próximo possível do pino de alimentação do MCU para permitir uma alimentação estável durante as operações de alta frequência realizadas pelos seus periféricos internos.

Junto ao pino de alimentação do módulo ESP-01, foram colocados condensadores de 100µF (dois destes) e 10µF a fim de poder responder aos grandes picos de corrente ocorridos durante a fase de transmissão do rádio. Condensadores de valor mais reduzido, para atenuação do ruído de alta frequência na alimentação, não foram colocados porque o próprio módulo já contém alguns. Os condensadores usados são SMD cerâmicos e de tântalo, por isso têm uma boa resposta em HF e não há razão nem necessidade de colocação de mais condensadores.

Relativamente às conexões do MCU ao módulo ESP-01 não carecem de nenhum circuito extra (tudo tem o mesmo nível de tensão sem necessidade de *pull-ups* ou divisores resistivos), pelo que foram ligados diretamente.

O mesmo não acontece com as ligações do microcontrolador ao barramento I^2C do equipamento (Figura 66) que, por proteção, foram colocadas em série com os pinos de entrada, resistências de 10kΩ para proporcionar algum "isolamento elétrico". Se por algum erro ou falha, os pinos do microcontrolador forem configurados como *output*, essas resistências limitarão a passagem de corrente para valores seguros e suportáveis quer pelo porto do MCU quer pelo porto do processador central do medidor Sanitas SBC 25/1.

Por razões semelhantes às anteriores, também a ligação ao botão ON/OFF, que deverá pulsar em *low state* para ligar o aparelho e iniciar medição, possui uma resistência de 100Ω com o mesmo propósito de limitar a corrente neste ramo do circuito.

[10] Consultar esquemático elétrico no anexo 8.4.

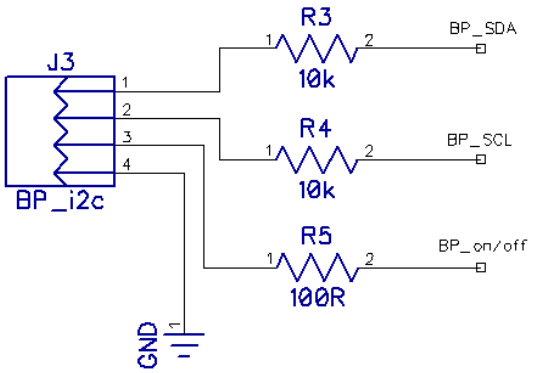


Figura 66 – Esquemático de ligação do circuito de controlo ao medidor de pressão arterial

Como não podia deixar de ser o circuito possui ainda dois LED's de sinalização através dos quais se pode analisar, de forma básica, onde se encontra o programa e se são ou não detetados erros.

Por último, este MCU (PIC18F25K22) possui dois periféricos de comunicação USART. O módulo ESP-01 requer um deles, já descrito, e o módulo sobrante foi ligado a um *header* fêmea na extremidade da PCB. É por aqui que se realiza toda a comunicação referente às configurações gerais da ligação *Wi-Fi*. Estas configurações incluem o nome do AP, *Access Point*, que pode ter nomes diferentes, a *password* de ligação e a API KEY de ligação ao ThingSpeak.

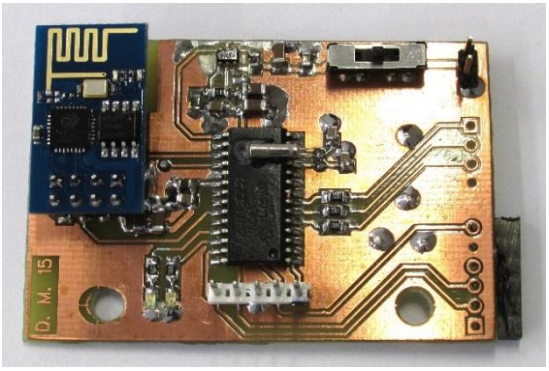


Figura 67 – Circuito protótipo

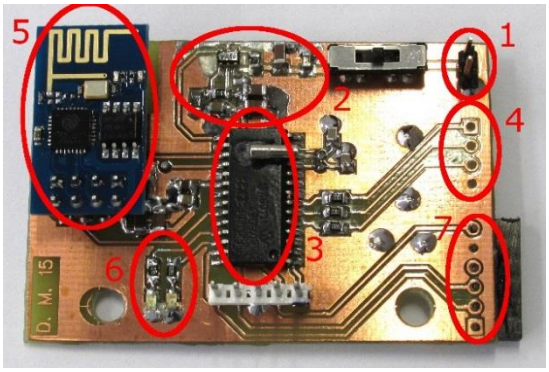


Figura 68 – Componentes do circuito protótipo

A Figura 67 é uma fotografia do circuito construído para o protótipo funcional. Na imagem da direita, Figura 68, estão assinalados os blocos e ligações principais para uma fácil identificação.

1 – entrada de alimentação, a ligar às baterias do Sanitas SBC 25/1;

2 – circuito do *boost converter*, usando o MCP1640C;

3 – microcontrolador, PIC18F25K22;

4 – ligações para o barramento I^2C do medidor e para o botão ON/OFF;

5 – módulo *Wi-Fi*, ESP-01;

6 – LED's de sinalização;

7 – ligação para um conversor USART-USB externo através do qual são feitas as configurações.

5.2. O Protótipo

A PCB que alberga o circuito protótipo foi projetada para que pudessem ser feitos dois furos para fixação. Esses furos têm diâmetro suficiente para poder apertar dois parafusos M3 e fixar a PCB ao medidor de pressão.

O local de aplicação escolhido foi a face lateral superior do aparelho para que não cause qualquer obstrução ao visor ou aos botões. Nesta posição também não oferece problemas à braçadeira, pois está paralelo à mesma.

Os fios ligados no interior do medidor passam para o exterior por um pequeno orifício lateral, onde são posteriormente soldados à PCB. Para que os fios não ficassem demasiado compridos e soltos, o orifício foi aberto o mais próximo possível do local onde seriam soldados os fios.

Foi ainda desenhada e impressa em plástico PLA uma peça de proteção (a preto na Figura 69) para salvaguardar o circuito no transporte e durante os testes.

Figura 69 – Protótipo, vista de frente

Figura 70 – Protótipo, vista de cima

6. Avaliação de desempenho e conclusões

6.1. Revisão dos objetivos

O principal objetivo deste trabalho foi, desde o início, a construção de um dispositivo protótipo, totalmente funcional capaz de desempenhar as suas funções, medir pressão arterial e colocar os dados *online* por IoT, num ambiente real. Felizmente esse objetivo foi alcançado com sucesso e o protótipo construído corresponde perfeitamente às expectativas iniciais.

Este êxito não teria sido possível sem as tecnologias criadas anteriormente que permitiram unir o melhor das ciências da saúde com o melhor da engenharia. A aplicabilidade deste dispositivo desenvolvido é muito útil e mostra como é possível conectar os mais variados objetos à Internet. Num contexto real irá permitir a monitorização remota, em tempo real ou não, de qualquer pessoa que esteja a usar este medidor de pressão arterial.

O público-alvo abrangido por este equipamento é enorme. Começando pelos indivíduos com desregulação da pressão arterial, sejam hipertensos ou hipotensos, ou tenham eles outras doenças cardíacas que levam a um descontrolo deste sinal vital. Este grupo, geralmente, tem que ter uma perceção diária da sua pressão arterial, seja para ajuste da terapêutica numa futura visita ao médico, seja pela prevenção de complicações cardíacas mais graves. A estes doentes, quando sujeitos a determinado tratamento, é-lhes prescrita a monitorização da pressão arterial em determinados períodos de tempo devido aos efeitos adversos que a medicação lhes pode causar. Outro grupo de indivíduos importante são os idosos, cuja preocupação com o coração aumenta com a idade e a sua falência se torna eminente, neste grupo é também uma mais-valia o registo automático visto que a taxa de demência é aumentada nesta faixa etária.

6.2. Exibição dos resultados

A ligação entre o medidor de pressão arterial e a *Internet of Things* tornou o procedimento de registo de pressões arteriais mais simplificado. Após o desenvolvimento deste trabalho com esta tecnologia em crescimento teve-se a perceção das muitas portas que se poderão abrir. Desde que exista a possibilidade de ligação à Internet, as medições podem ser acedidas a partir de qualquer lugar do mundo.

No que toca à aquisição de dados, não só é possível como se torna simples e agradável visualmente com a disponibilização de plataformas *opensource* como é o caso, anteriormente descrito, do ThingSpeak. Esta plataforma permite o registo gráfico e consulta dos dados fornecidos pelo medidor de pressão arterial. Estes gráficos são altamente configuráveis em títulos, legendas, cores, escalas, dinamismo, quantidade de valores a mostrar, etc.

São muito variadas as opções de configuração e tratamento dos valores, que neste trabalho compuseram 3 campos/*fields* no canal criado para o efeito:

- Campo/*field* 1: pressão sistólica
- Campo/*field* 2: pressão diastólica
- Campo/*field* 3: batimentos cardíacos por minuto

A cada chegada de dados à plataforma ThingSpeak, esta junta-lhes a data e a hora. É ainda possível selecionar se se pretende tornar o acesso aos registos público ou privado com um clique numa *checkbox*. Outra característica do ThingSpeak é a facilidade em exportar dados (fazendo o *download* de um ficheiro *.csv) e realizar análises *online* com o apoio da conhecida ferramenta MATLAB.

A Figura 71, mostra como o canal utilizado neste trabalho (com o ID 38259) é mostrado ao público, pelo endereço ***https://thingspeak.com/channels/38259***. O canal é de domínio publico e para ter acesso a esta visualização, basta aceder ao *link* do mesmo.

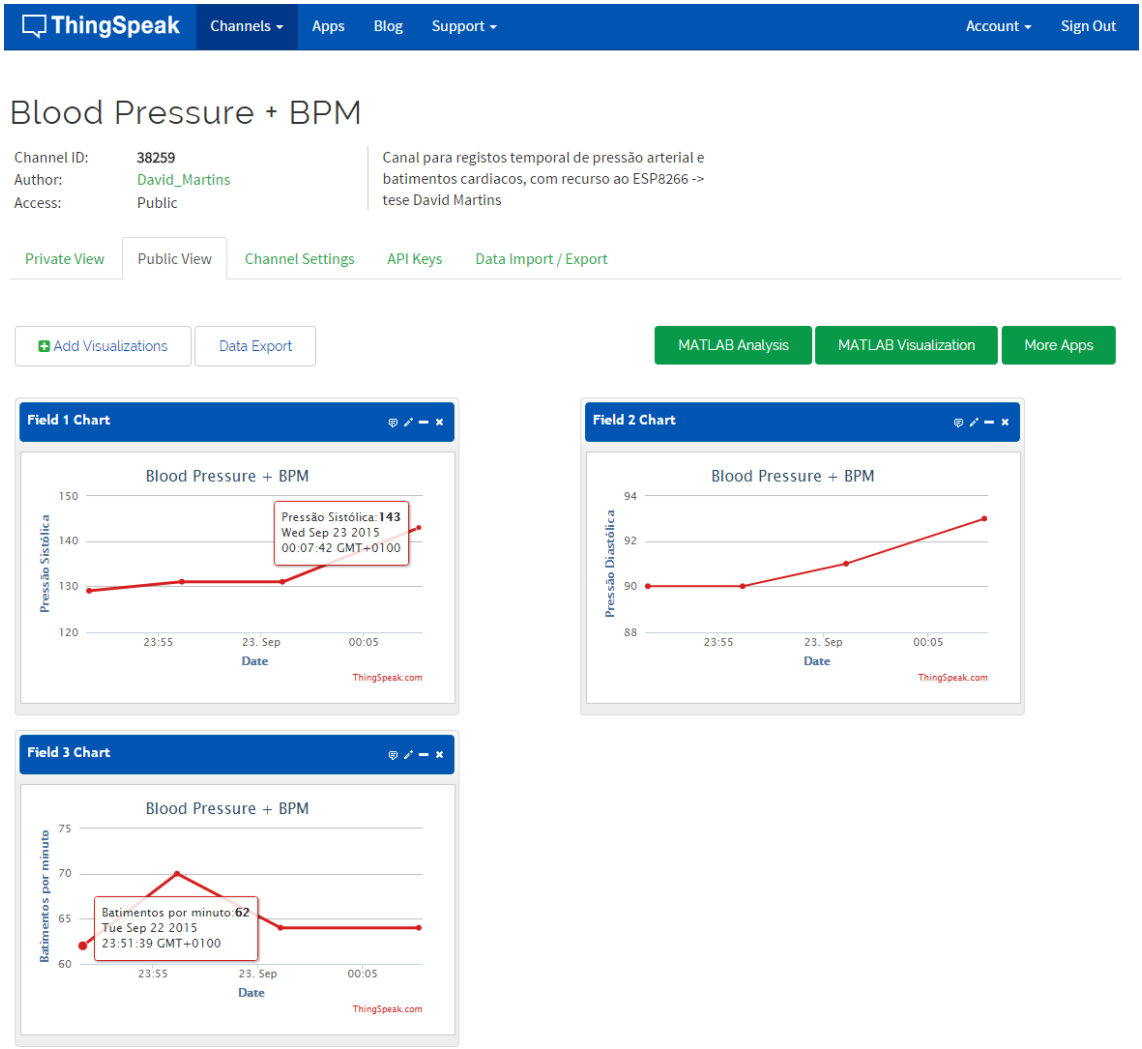


Figura 71 – Aspecto do canal criado no ThingSpeak

Para que se possa ter mais informações sobre cada um dos pontos, basta somente passar o rato sobre eles e aguardar que um separador abra com os detalhes.

No fundo, foi para permitir esta visualização de dados e esta possibilidade simples de partilha que todo este trabalho foi realizado. É aqui que o verdadeiro sucesso e cumprimento dos objetivos culmina: recolha dos dados com um protótipo funcional e envio dos mesmos, via IoT, para uma plataforma *online*.

Assim mais detalhes técnicos sobre o protótipo necessitam ainda ser analisados e isso será tratado do subcapítulo seguinte.

6.3. Verificação de consumos

Uma importante consideração acerca do protótipo resultante do trabalho desenvolvido é a sua viabilidade energética. O protótipo, embora funcional, pode perder aplicabilidade prática se se colocar em causa a sua portabilidade. Quer com isto dizer que um medidor que inicialmente era portátil e funcionava com duas baterias do tipo AAA, não pode ficar sujeito a um *upgrade* que consuma tanta energia ao ponto de o tornar sem utilidade. Onde antes um par de baterias daria para 300 medições se agora der para apenas 50 medições todo o trabalho desenvolvido se torna inviável.

Por esta razão, e já com todo o equipamento a operar da forma correta, foi realizada uma análise de consumo para determinar qual o encargo em tempo de vida útil, por par de baterias, que estas alterações vieram trazer. Para proceder a essa análise é necessário fazer uma medição rigorosa da corrente consumida pelo circuito assim como ter um registo com uma elevada cadência dessas mesmas medições de modo a elaborar cálculos com resultados confiáveis.

Para auxiliar na tarefa da medição e respetivo registo, foi utilizado um aparelho de medição em protótipo chamado *µAmp Serial Monitor*[72]. Este equipamento possui um ADC (Analog-to-Digital Converter) de 13bits, com seleção automática de um amplificador de ganho programável (PGA), de modo a ter em todas as leituras a resolução máxima possível, e uma frequência de amostragem de 50Hz (20 milissegundos entre amostragens). A medição da corrente é feita medindo a queda de tensão provocada em quatro resistências de 1Ω em paralelo, perfazendo uma resistência equivalente de 0.25Ω, até ao diferencial máximo de 100mV, medindo e registando por isso valores de corrente de até 400mA (100mV/0.25Ω = 400mA). Esse diferencial é posteriormente amplificado, digitalizado pelo ADC e lido pelo MCU contido neste aparelho, que avalia ainda o ganhos e procede ao envio por USB. Embora o ADC seja de 13bits, mas recorrendo ao amplificador programável, consegue resoluções mínimas de 6uA (na gama até 50mA), variando de acordo com a corrente a medir.

Foi utilizada uma fonte de alimentação de bancada para garantir uma alimentação constante de 3V (intuito de simular o conjunto das duas baterias AAA) e que não variasse com os aumentos do valor de corrente, como aconteceria na utilização de baterias devido à sua resistência interna. A Figura 72 ilustra a configuração da montagem utilizada para efetuar estas análises.

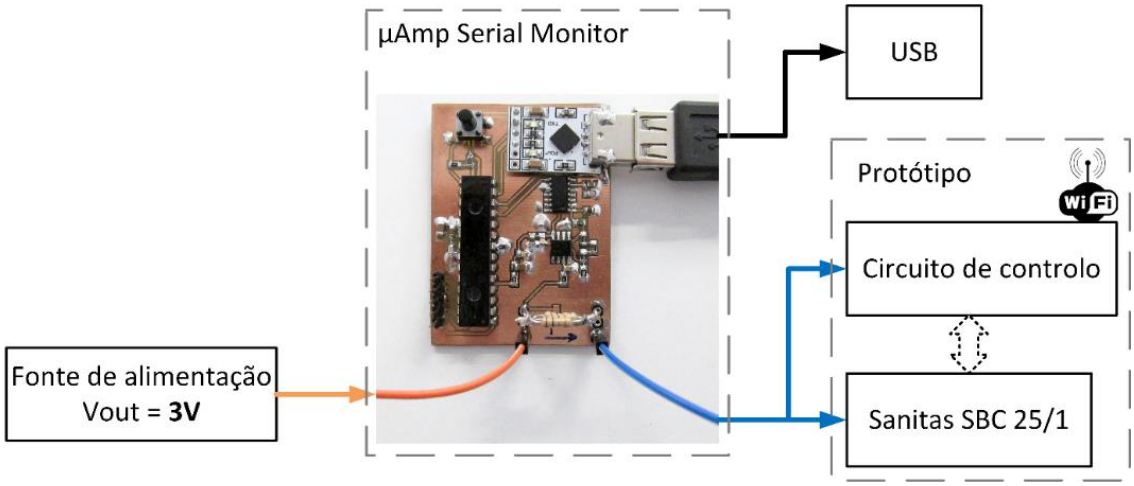


Figura 72 – Configuração de montagem para realizar as análise de consumos

Em série com a fonte de alimentação foi ligado o medidor de corrente (µAmp S. M.), cujo funcionamento já foi brevemente descrito, e deste partiu a alimentação para o protótipo, alimentando o medidor Sanitas SBC 25/1 e o circuito controlador desenvolvido. O resultado das medições é enviado por USB, com uma taxa de 50Hz, onde um terminal série no computador procede ao armazenamento dos dados.

Com toda a montagem instalada como mostrado na figura, iniciou-se o registo e procedeu-se a uma medição de pressão arterial, sendo o processo todo controlado de forma autónoma pelo aparelho protótipo. No decorrer da mesma, vários apontamentos temporais foram realizados de modo a criar correspondências no registo do perfil de corrente a diferentes ações e acontecimentos.

Na Figura 73 pode ver-se ilustrado o gráfico temporal das alterações de corrente ocorridas no ramo de alimentação do protótipo. O gráfico resulta da recolha de 2677 medições ao longo do período de 53.54 segundos (uma amostra a cada 20 milissegundos).

As respetivas marcações estão descritas em seguida e correspondem aos momentos de alteração de diferentes ações.

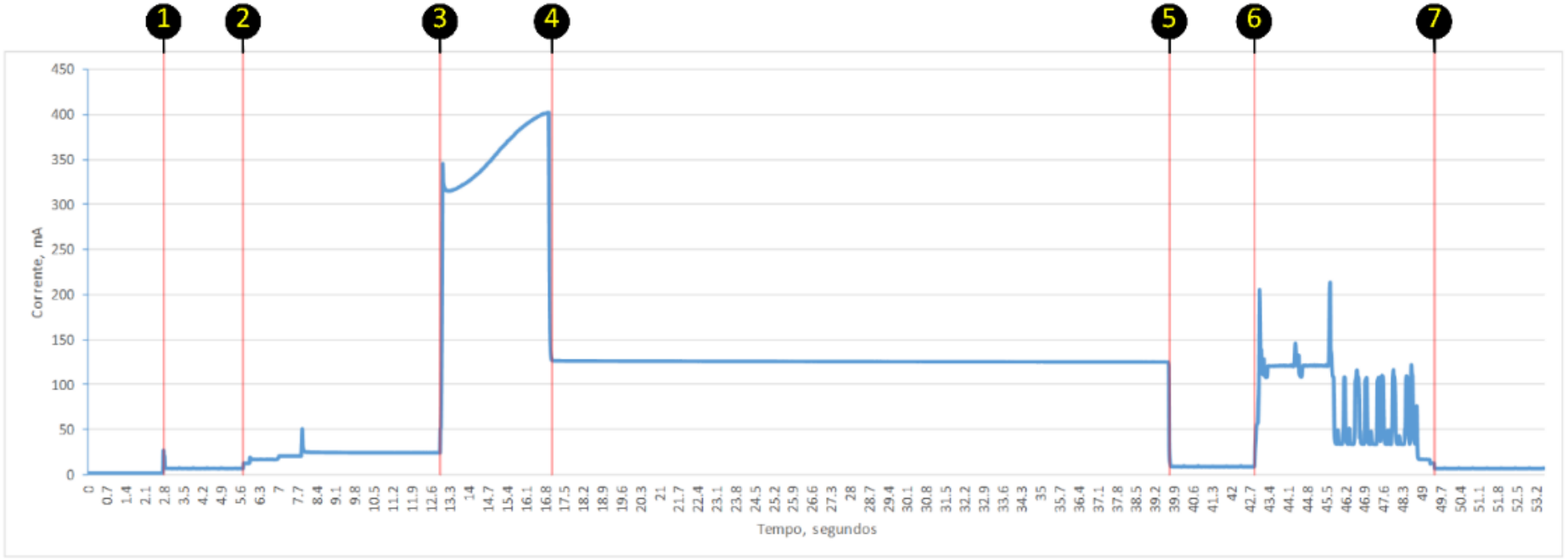


Figura 73 – Representação gráfica das alterações de corrente ao longo de uma medição

1) Momento em que se ligou o interruptor de alimentação do circuito controlador. Até aqui tudo estava desligado.
2) Após o circuito de controlo aguardar 3 segundos, é dada a ordem ao medidor de pressão arterial para iniciar a medição.
3) Até este momento o medidor Sanitas SBC 25/1 esteve a realizar o seu arranque e neste ponto é iniciado o processo propriamente dito de medição de pressão arterial, pois é o momento em que a bomba de ar é colocada em funcionamento.
4) Atingida a pressão máxima típica, o aparelho faz desligar a bomba de ar e aciona a válvula para reduzir de forma controlada a pressão na braçadeira. De referir que o perfil que a corrente assumiu se deve ao facto de a carga sobre o motor ir aumentando, da mesma forma que a pressão no interior da braçadeira aumenta. No final a bomba tem de exercer um esforço maior para ultrapassar a pressão acumulada na camara-de-ar.
5) Chega ao final a medição da pressão arterial e a válvula de ar é desligada deixando esvaziar por completo a braçadeira. Deste ponto em diante o consumo é atribuído somente ao funcionamento do circuito controlador.
6) Após uma pausa de 3 segundos, inicia-se o processo de envio de dados via *Wi-Fi*. Neste momento arranca o módulo ESP-01 e procede a todas as operações já descritas para fazer a transmissão dos valores.
7) Chega ao fim o envio dos dados com o uso do ESP-01 e o mesmo é colocado em *power-down*. O circuito controlador termina as suas funções.

Nesta breve análise temporal de acontecimentos não foram considerados alguns pormenores, como o consumo do circuito controlador antes do ponto 5, assim como o consumo do aparelho de medição Sanitas SBC 25/1, que embora cesse o seu funcionamento, o ecrã de cristais líquidos continua a mostrar os valores da medição realizada sendo apenas desligado pelo circuito controlador no ponto 7.

A justificação para isso é relativamente fácil de aceitar. No que toca ao consumo do LCD de cristais líquidos, estes são conhecidos pelo seu consumo mínimo, comum a todos a todos os ecrãs do género, e quando comparado o consumo do circuito de controlo (pontos 1 a 2) comparativamente com todo o gasto do medidor de pressão pode, para uma análise mais rápida, considerar-se insignificante e desprezível para esta avaliação. Além disso existe ainda a atenuante de no período compreendido entre os pontos 5 e 7 são também influenciados pelo leve consumo implicado pelo funcionamento do LCD.

Com todos os valores recolhidos e com a fácil visualização gráfica e respetivas conclusões, pode afirmar-se que o consumo ocorrido entre os pontos 2 e 5 é inerente ao Sanitas SBC 25/1 e que

sobre esses se despreza qualquer influência provocada pelo circuito controlador. Assim, durante esse período foi calculada uma corrente média de 129.8mA, ao longo de 34.02 segundos e com uma tensão de alimentação, fornecida pela fonte de bancada, de 3V.

Os dados apresentados permitem calcular a energia gasta, em Joules, pelo medidor Sanitas SBC 25/1 durante o seu processo de medição.

$$I_{média} = 129.8mA \qquad \Delta t = 34.02s \qquad V_{in} = 3V$$

$$P_{média} = V_{in} * I_{média} = 3 * 129.8 = 389.4mW$$

$$E = P_{média} * \Delta t = 0.3894 * 34.02 = \mathbf{13.3}\boldsymbol{J}$$

De acordo com os cálculos, durante o período de medição de pressão arterial, o aparelho teve um gasto energético de 13.3 Joules.

No decorrer do período entre o ponto 5 e o ponto 7, os gastos vão ser atribuídos ao circuito controlador e todo o processo realizado pelo mesmo. Desde já se assinala que grande parte de todo o consumo, nesta fase, é atribuída à utilização do módulo ESP-01 e à transmissão de dados por *Wi-Fi*.

A corrente média calculada durante esse período foi de 56.1mA ao longo de 9.78 segundos, com a mesma tensão de alimentação.

$$I_{média} = 56.1mA \qquad \Delta t = 9.78s \qquad V_{in} = 3V$$

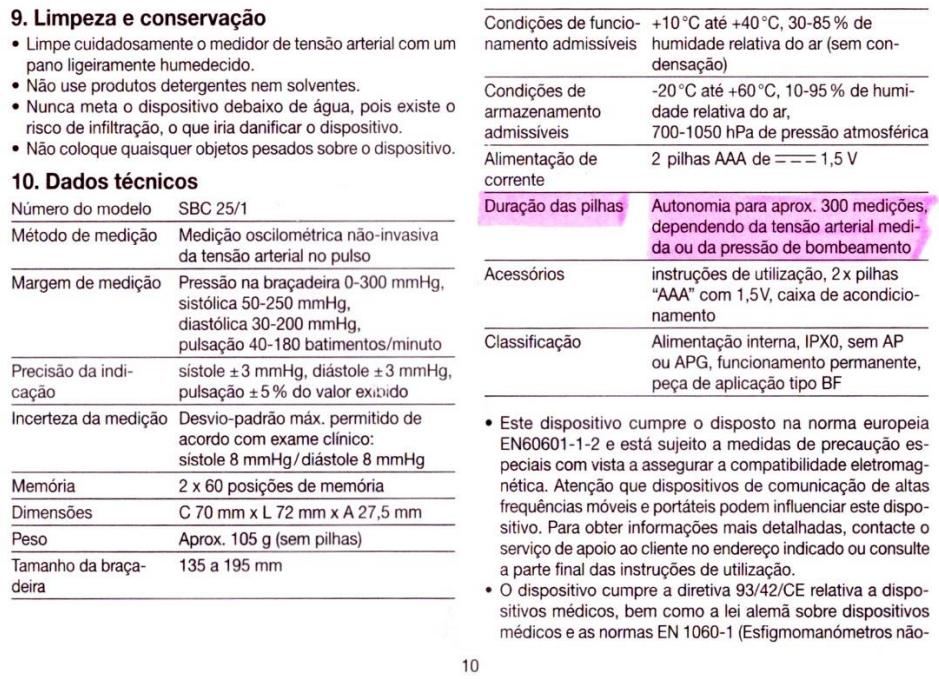
$$P_{média} = V_{in} * I_{média} = 3 * 56.1 = 168.3mW$$

$$E = P_{média} * \Delta t = 0.1683 * 9.78 = \mathbf{1.7}\boldsymbol{J}$$

O circuito controlador, durante o período mencionado, provoca um gasto energético de 1.7 Joules.

O protótipo gasta, no geral e de acordo com o calculado, em cada processo de medição e envio de dados 15 Joules, sendo que 13.3 Joules estão diretamente implicados ao medidor Sanitas SBC 25/1. Assim o circuito controlador veio acrescentar um gasto extra de 12.8% em relação ao medidor simples, ou seja, por outras palavras o protótipo desenvolvido verá o tempo de vida útil de cada par de baterias AAA encurtado em 12.8%.

Não avançando muito em detalhe, e para se ter uma ideia geral do que esta percentagem significa, considerou-se como base a Figura 74 retirada do manual de instruções do medidor de pressão arterial.



9. Limpeza e conservação

- Limpe cuidadosamente o medidor de tensão arterial com um pano ligeiramente humedecido.
- Não use produtos detergentes nem solventes.
- Nunca meta o dispositivo debaixo de água, pois existe o risco de infiltração, o que iria danificar o dispositivo.
- Não coloque quaisquer objetos pesados sobre o dispositivo.

10. Dados técnicos

Número do modelo	SBC 25/1
Método de medição	Medição oscilométrica não-invasiva da tensão arterial no pulso
Margem de medição	Pressão na braçadeira 0-300 mmHg, sistólica 50-250 mmHg, diastólica 30-200 mmHg, pulsação 40-180 batimentos/minuto
Precisão da indicação	sístole ±3 mmHg, diástole ±3 mmHg, pulsação ±5 % do valor exibido
Incerteza da medição	Desvio-padrão máx. permitido de acordo com exame clínico: sístole 8 mmHg/diástole 8 mmHg
Memória	2 x 60 posições de memória
Dimensões	C 70 mm x L 72 mm x A 27,5 mm
Peso	Aprox. 105 g (sem pilhas)
Tamanho da braçadeira	135 a 195 mm
Condições de funcionamento admissíveis	+10 °C até +40 °C, 30-85 % de humidade relativa do ar (sem condensação)
Condições de armazenamento admissíveis	-20 °C até +60 °C, 10-95 % de humidade relativa do ar, 700-1050 hPa de pressão atmosférica
Alimentação de corrente	2 pilhas AAA de ⎓ 1,5 V
Duração das pilhas	Autonomia para aprox. 300 medições, dependendo da tensão arterial medida ou da pressão de bombeamento
Acessórios	instruções de utilização, 2 x pilhas "AAA" com 1,5 V, caixa de acondicionamento
Classificação	Alimentação interna, IPX0, sem AP ou APG, funcionamento permanente, peça de aplicação tipo BF

- Este dispositivo cumpre o disposto na norma europeia EN60601-1-2 e está sujeito a medidas de precaução especiais com vista a assegurar a compatibilidade eletromagnética. Atenção que dispositivos de comunicação de altas frequências móveis e portáteis podem influenciar este dispositivo. Para obter informações mais detalhadas, contacte o serviço de apoio ao cliente no endereço indicado ou consulte a parte final das instruções de utilização.
- O dispositivo cumpre a diretiva 93/42/CE relativa a dispositivos médicos, bem como a lei alemã sobre dispositivos médicos e as normas EN 1060-1 (Esfigmomanómetros não-

10

Figura 74 – Duração das baterias segundo as indicações do fabricante

Quer por isso dizer que, sem modificações, o fabricante aponta para um número aproximado de 300 medições por par de baterias, e o aumento do consumo energético em 12.8% vai encurtar esse número de medições na mesma proporção. Desta feita, teoricamente, cada par de baterias AAA no protótipo realizado terá um tempo de vida de aproximadamente 261 medições, sendo uma quantidade perfeitamente aceitável.

6.4. Conclusões

Neste trabalho foi desenvolvido um protótipo capaz de, automaticamente, iniciar uma medição de pressão arterial, recolher os resultados medidos e fazer o envio dos mesmos para uma plataforma *online*, via *Wi-Fi*. Sendo este o principal objetivo pode afirmar-se que foi cumprido com sucesso. Os testes realizados ao protótipo final foram sistematicamente bem-sucedidos desde que as seguintes especificações fossem cumpridas:

- Assumir uma posição e comportamento adequado à medição da pressão arterial;
- Garantir uma tensão mínima por bateria de 1.2V (em carga);
- Utilizar o protótipo dentro do raio de cobertura do *Access Point* ao qual se deverá ligar;
- Parâmetros de ligação ao *Access Point* e ao ThingSpeak devidamente configurados.

No decurso deste trabalho foram encontradas dificuldades que, apesar de ultrapassadas, vieram criar atraso no seu progresso. Este facto veio impossibilitar a realização atempada de algumas melhorias que devem ser notadas. Estas incidem essencialmente sobre o gasto energético. O *firmware* do MCU não foi construído tendo em conta a otimização de consumos e essa é uma grande problemática. No estado em que se encontra, todo o sistema consome, em *standby*, cerca de 17mW. Este é um valor elevado, pois se entre cada medição houver um intervalo de 20 minutos, calcula-se para esse período um consumo de energia a rondar os 20 Joules.

$$E = P_{média} * \Delta t = 0.017 * 20 * 60 = \mathbf{20.4}\boldsymbol{J}$$

Pode afirmar-se que durante esse tempo o protótipo gasta mais energia que durante o processo de medição, recolha e envio de dados para a plataforma *online* (calculado em 15 Joules). Este é um consumo exagerado e que pode ser reduzido facilmente fazendo algumas alterações como: desligar o *clock* principal e respetivo oscilador interno com PLL (*clock* a 64MHz), passar para oscilador *low-power* secundário com recurso ao cristal externo e desligar todos os periféricos internos em funcionamento[11]. Estas modificações são suficientes para provocar uma redução bastante significativa no consumo. Sem elas o MCU continua a funcionar usando recursos e processamento muito acima do necessário para o estado de standby.

Do ponto de vista da aplicabilidade, e deixando de lado o aspeto mais técnico, a plataforma ThingSpeak não possui as características inerentes aos sistemas de saúde. Esta é uma excelente plataforma para realizar a prova de conceito de um projeto, como foi o caso deste trabalho. A plataforma já fornece os campos e configurações necessárias para uma rápida implementação de IoT, mas trata-se de uma plataforma demasiado genérica e que não oferece serviços específicos à área de aplicação, saúde. Como sugestão, poder-se-ia realizar a integração deste tipo de equipamentos a um sistema específico de gestão de saúde, como os que já existem e são utilizados pela comunidade médica.

A implementação dos melhoramentos e sugestões supra indicadas deverão tornar o protótipo mais robusto e mais próximo da realidade comercial. Equipamentos clínicos de monitorização de sinais de saúde usando a tecnologia implementada, no protótipo desenvolvido nesta dissertação, poderão trazer benefícios no futuro e estão cada vez mais perto de se tornarem comuns no dia-a-dia.

[11] Dois periféricos USART, seis *timers*, dois periféricos MSSP, cinco periféricos de CCP, um periférico de CTMU, dois periféricos de comparação analógica e um periférico de ADC.

7. Referências bibliográficas

[1] "Health wearables: Early days." [Online]. Available: https://www.pwc.se/sv/halso-sjukvard/assets/health-wearables-early-days.pdf. [Accessed: 25-Sep-2015].

[2] S. M. R. Islam, D. Kwak, H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care : A Comprehensive Survey," *IEE Access*, vol. 3, p. 31, 2015.

[3] "Caderno Saúde: Hipertensão Arterial." [Online]. Available: https://www.bial.com/imagem/Caderno saude_Hipertensao arterial_V2.pdf. [Accessed: 25-Sep-2015].

[4] "Tudo o que deve saber sobre Hipertensão Arterial." [Online]. Available: http://www.fpcardiologia.pt/wp-content/uploads/2013/08/Brochura-CRC-N%C2%BA-9-Hip.-Art..pdf. [Accessed: 25-Sep-2015].

[5] W. B. White and G. a Mansoor, "Ambulatory blood pressure monitoring.," *Curr. Opin. Nephrol. Hypertens.*, vol. 2, no. 6, pp. 928–934, 1993.

[6] I. Chiuchisan, H. Costin, and O. Geman, "Adopting the Internet of Things Technologies in Health Care Systems," *Int. Conf. Expo. Electr. Power Eng.*, no. Epe 2014, pp. 532–535, 2014.

[7] K. M. Hou, U. Blaise, and P. C. Ii, "A Internet e a Rede das coisas : desafios e oportunidades," p. 15, 2013.

[8] "Everything You Need To Know About The Internet Of Things - Forbes." [Online]. Available: http://www.forbes.com/sites/jacobmorgan/2014/10/30/everything-you-need-to-know-about-the-internet-of-things/print/. [Accessed: 25-Sep-2015].

[9] K. M. Van De Graaff, D. Strete, and C. H. Creek, *Human Anatomy*, 6th ed. McGraw Hill, 2001.

[10] P. Tate, R. R. Seeley, and T. D. Stephens, *Anatomia e Fisiologia*, 6ª ed. McGraw-Hill, 2005.

[11] K. Saladin, *Anatomy & Physiology: The Unity of Form and Function*. McGraw-Hill, 2006.

[12] "WHO | World Health Organization." [Online]. Available: http://gamapserver.who.int/gho/interactive_charts/ncd/risk_factors/blood_pressure_prevalence/atlas.html. [Accessed: 26-Sep-2015].

[13] "A short history of sphygmomanometers and blood pressure measurement | Museum of Health Care." [Online]. Available: https://museumofhealthcare.wordpress.com/2013/04/07/world-health-day-2013-a-short-history-of-sphygmomanometers-and-blood-pressure-measurement/. [Accessed: 25-Sep-2015].

[14] "pressionArterielle.jpg (483×600)." [Online]. Available: http://raymond.rodriguez1.free.fr/Documents/Organisme-A/Regulations/pressionArterielle.jpg. [Accessed: 25-Sep-2015].

[15] "A fisiologia e suas 'ferramentas' no século XIX (o quimógrafo e o prelúdios da psicologia científica)." [Online]. Available: https://historiapsi.wordpress.com/2011/04/24/a-fisisologia-e-suas-ferramentas-no-seculo-xix-preludios-da-psicologia-cientifica/. [Accessed: 25-Sep-2015].

[16] "Carl_F._W._Ludwig's_kymograph_Wellcome_L0012233.jpg (1327×1591)." [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/c7/Carl_F._W._Ludwig%27s_kymograph_Wellcome_L0012233.jpg. [Accessed: 25-Sep-2015].

[17] "rocci.gif (322×372)." [Online]. Available: http://mathematikos.mat.ufrgs.br/disciplinas/ufrgs/mat01039032/webfolios/grupo4/rocci.gif. [Accessed: 25-Sep-2015].

[18] "History of Stethoscopes and Sphygmomanometers | Marketing Gifts." [Online]. Available: http://www.halsun.net/node/40. [Accessed: 25-Sep-2015].

[19] E. A. M. Arcuri, T. L. de Araújo, E. V. Veigaa, S. M. J. V. de Oliveira, J. L. T. Lamas, and J. L. F. Santos, "Sons de Korotkoff: Desenvolvimento da pesquisa em esfigmomanometria na Escola de Enfermagem da USP," *Rev. da Esc. Enferm.*, vol. 41, no. 1, pp. 147–153, 2007.

[20] "What Is a Sphygmomanometer: Definition & History of Invention | Medical Equipment." [Online]. Available: http://medinstrum.com/what-is-a-sphygmomanometer/. [Accessed: 25-Sep-2015].

[21] "2111_Blood_Pressure_Graph.jpg (1648×1027)." [Online]. Available: http://cnx.org/resources/fdcac85ed1a77bc08ac2d0a29a2d68822a732d50/2111_Blood_Pressure_Graph.jpg. [Accessed: 25-Sep-2015].

[22] "Riester - Professional Diagnostic: History." [Online]. Available: http://www.riester.de/History.645.0.html. [Accessed: 25-Sep-2015].

[23] "History of Omron's Blood Pressure Monitor." [Online]. Available: http://www.healthcare.omron.co.jp/bpm/english/history/02.html. [Accessed: 25-Sep-2015].

[24] "04_01_pic02.jpg (213×163)." [Online]. Available: http://www.healthcare.omron.co.jp/bpm/english/history/img/04_01_pic02.jpg. [Accessed: 25-Sep-2015].

[25] "04_02_pic01.jpg (213×164)." [Online]. Available: http://www.healthcare.omron.co.jp/bpm/english/history/img/04_02_pic01.jpg. [Accessed: 25-Sep-2015].

[26] J. Landgraf, S. H. Wishner, and R. a. Kloner, "Comparison of automated oscillometric versus auscultatory blood pressure measurement," *Am. J. Cardiol.*, vol. 106, no. 3, pp. 386–388, 2010.

[27] "Blood Pressure: Auscultatory Method." [Online]. Available: http://www.medicine.mcgill.ca/physio/vlab/cardio/auscul.htm. [Accessed: 25-Sep-2015].

[28] "Medical Relevance of Oscillometric Blood Pressure Technology." [Online]. Available: http://www.microlife.com/healthguide/hypertension/medical/oscillometric/. [Accessed: 25-Sep-2015].

[29] "How blood-pressure devices work." [Online]. Available: http://www.edn.com/design/medical/4324017/Guest-Commentary-How-blood-pressure-devices-work. [Accessed: 25-Sep-2015].

[30] F. Semiconductor, A. Note, and S. Lopez, "Blood Pressure Monitor Fundamentals and Design," pp. 1–38, 2012.

[31] Z. Feng, "Blood Pressure Meter Design Using Microchip's PIC24F Microcontroller and Analog Devices," 2013.

[32] "key_visual_Oscillometric.jpg (597×269)." [Online]. Available: http://tensoval.com/images/key_visual_Oscillometric.jpg. [Accessed: 25-Sep-2015].

[33] M. C. Van Der Wel, I. E. Buunk, C. Van Weel, T. a B. M. Thien, and J. C. Bakx, "A Novel Approach to Office Blood Pressure sure vs Daytime Ambulatory Blood Pressure," pp. 128–136, 2011.

[34] "White coat hypertension," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/White_coat_hypertension. [Accessed: 25-Sep-2015].

[35] P. Owens, N. Atkins, and E. O'Brien, "Diagnosis of white coat hypertension by ambulatory blood pressure monitoring.," *Hypertension*, vol. 34, no. 2, pp. 267–272, 1999.

[36] F. Sayk, C. Becker, C. Teckentrup, H. L. Fehm, J. Struck, J. P. Wellhoener, and C. Dodt, "To dip or not to dip: On the physiology of blood pressure decrease during nocturnal sleep in healthy humans," *Hypertension*, vol. 49, no. 5, pp. 1070–1076, 2007.

[37] J. Amar, I. Vernier, E. Rossignol, V. Bongard, C. Arnaud, J. J. Conte, M. Salvador, and B. Chamontin, "Nocturnal blood pressure and 24-hour pulse pressure are potent indicators of mortality in hemodialysis patients.," *Kidney Int.*, vol. 57, no. 6, pp. 2485–91, Jun. 2000.

[38] E. Gunther, "Features and Benefits of IEC 61850."

[39] "The Internet of Things How the Next Evolution of the Internet Is Changing Everything." [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. [Accessed: 26-Sep-2015].

[40] A. Anurag, S. Rahimi Moosavi, A.-M. Rahmani, T. Westerlund, G. Yang, P. Liljeberg, and H. Tenhunen, "Pervasive Health Monitoring Based on Internet of Things: Two Case Studies," *Proc. 4th Int. Conf. Wirel. Mob. Commun. Healthc. - "Transforming Healthc. through Innov. Mob. Wirel. Technol.*, pp. 275–278, 2014.

[41] W.-J. Li, Y.-L. Luo, Y.-S. Chang, and Y.-H. Lin, "A wireless blood pressure monitoring system for personal health management.," *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2010, pp. 2196–2199, 2010.

[42] G. Zhuang, X. Zou, C. Guo, and Y. Liu, "WIRELESS SPHYGMOMANOMETER BASED ON ZIGBEE," pp. 8–11, 2012.

[43] B. A. Zneid, M. Al-zidi, and T. Al-kharazi, “Non-invasive Blood Pressure Remote Monitoring Instrument Based Microcontroller,” pp. 252–257, 2014.

[44] D. W. Fisher, “Withings Takes Home Health Monitoring to a Whole New Level with its Wireless Blood Pressure Monitor,” p. 2, 2014.

[45] “Withings-Wireless-Blood-Pressure-Cuff.jpg (1383×1500).” [Online]. Available: https://securityledger.com/wp-content/uploads/2014/07/Withings-Wireless-Blood-Pressure-Cuff.jpg. [Accessed: 25-Sep-2015].

[46] Wikipedia, “Bluetooth low energy.” [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth_low_energy.

[47] “CES aponta para futuro da computação vestível e de objetos conectados.” [Online]. Available: http://www1.folha.uol.com.br/tec/2014/01/1396196-ces-aponta-para-futuro-de-computacao-vestivel-e-objetos-conectados.shtml. [Accessed: 26-Sep-2015].

[48] “SBC21.jpg (500×500).” [Online]. Available: http://www.sanitas-online.de/web/_bilder/Produktbilder/1_Blutdruck/SBC21.jpg. [Accessed: 26-Sep-2015].

[49] Microchip, “Basic Serial EEPROM Operation - AN536.” [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00536.pdf. [Accessed: 26-Sep-2015].

[50] “What Is a Sniffer in Computer Networking?” [Online]. Available: http://compnetworking.about.com/od/networksecurityprivacy/g/bldef_sniffer.htm. [Accessed: 30-Sep-2015].

[51] P. Semiconductors, “The I2C-bus specification,” *Philips Semicond.*, vol. 9397, no. 750, p. 954, 2000.

[52] “Sanitas SBM30 teardown.” [Online]. Available: http://jdesbonnet.blogspot.de/2011/01/sanitas-smb30-aka-hl868ba-teardown.html. [Accessed: 28-Sep-2015].

[53] D. S. Table, “24AA08/24LC08B 8K I2C Serial EEPROM,” pp. 1–30, 2009.

[54] “Repeated Start Condition - I2C Bus.” [Online]. Available: http://www.i2c-bus.org/repeated-start-condition/. [Accessed: 26-Sep-2015].

[55] Microchip, *PIC18F2xK22 Data-sheet*, vol. 18, no. L. 2012.

[56] L. P. Bor, “PIC18F ‘ K22 ’ Family of Microcontrollers Broadest Line of Low Power , 5V 8-bit MCUs,” *Technology*, pp. 5–6.

[57] “MultiMaster - I2C Bus.” [Online]. Available: http://www.i2c-bus.org/multimaster/. [Accessed: 26-Sep-2015].

[58] “start_cond.jpg (229×180).” [Online]. Available: http://dlnware.com/sites/dlnware.com/files/images/start_cond.jpg. [Accessed: 26-Sep-2015].

[59] “stop_cond.jpg (229×180).” [Online]. Available: http://dlnware.com/sites/dlnware.com/files/images/stop_cond.jpg. [Accessed: 26-Sep-2015].

[60] "figure6.jpg (569×453)." [Online]. Available: http://www.byteparadigm.com/kb/admin/media_store/2/AA-00255/figure6.jpg. [Accessed: 26-Sep-2015].

[61] "ESP8266 | Espressif." [Online]. Available: http://espressif.com/en/products/esp8266/. [Accessed: 26-Sep-2015].

[62] Wikipedia, "ESP8266." [Online]. Available: https://en.wikipedia.org/wiki/ESP8266.

[63] "ESP8266 Serial WIFI Module." [Online]. Available: http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module. [Accessed: 26-Sep-2015].

[64] "What Is the Maker Movement and Why Should You Care?" [Online]. Available: http://www.huffingtonpost.com/brit-morin/what-is-the-maker-movemen_b_3201977.html. [Accessed: 28-Sep-2015].

[65] "31083.jpg (600×600)." [Online]. Available: http://cdn.boxtec.ch/images/31083.jpg. [Accessed: 26-Sep-2015].

[66] "ESP8266 ESP-01 WiFi Wireless Tranceiver Module." [Online]. Available: http://www.addicore.com/ESP8266-ESP-01-p/130.htm. [Accessed: 26-Sep-2015].

[67] "ESP-01 Module — ESP8266." [Online]. Available: http://esp8266.co.uk/modules/esp-01/. [Accessed: 26-Sep-2015].

[68] "esp-01.jpg (800×379)." [Online]. Available: http://fabacademy.org/archives/2015/doc/images/esp-01.jpg. [Accessed: 26-Sep-2015].

[69] "Lista de portas de protocolos – Wikipédia, a enciclopédia livre." [Online]. Available: https://pt.wikipedia.org/wiki/Lista_de_portas_de_protocolos. [Accessed: 26-Sep-2015].

[70] "Datasheet Battery AAA (LR03)." [Online]. Available: http://www.professional.duracell.com/downloads/datasheets/product/Ultra Power/Ultra-Power_AAA_MX2400.pdf. [Accessed: 27-Sep-2015].

[71] "MCP1640/B/C/D." [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/22234B.pdf. [Accessed: 27-Sep-2015].

[72] "Proj. uAmp Serial Monitor." [Online]. Available: http://blogoengenhocas.blogspot.pt/search/label/Proj. uAmp Serial Monitor. [Accessed: 27-Sep-2015].

[73] "24AA08/24LC08B 8K I2C™ Serial EEPROM," *DS21710J*, 2009. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/21710J.pdf. [Accessed: 26-Sep-2015].

8. Anexos

8.1. Análise técnica do medidor

Os componentes funcionais de um medidor já foram descritos (subcapítulo 2.3) e são esses que se espera encontrar neste medidor também.

Pode ser dividido em 5 partes distintas:

- Braçadeira de garrote;
- Bomba de ar;
- Válvula de ar;
- Sensor de pressão;
- Unidade de controlo e medição.

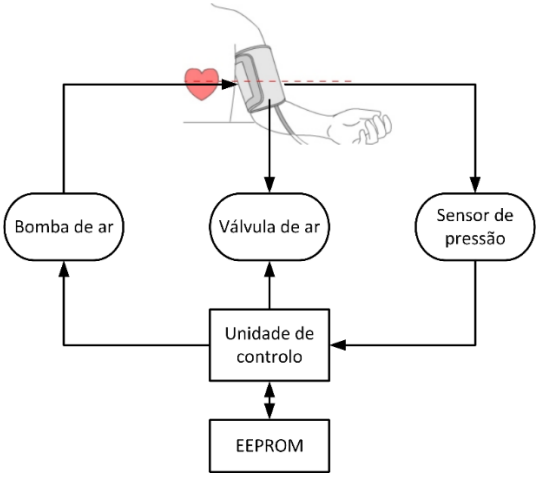


Figura 75 - Diagrama de blocos de um esfigmomanómetro automático com EEPROM

Neste caso específico, além dos 5 elementos previstos, espera-se também encontrar, algo que embora não esteja ligado diretamente ao processo de medição da pressão arterial, permitirá ter o acesso aos valores medidos: uma EEPROM.

As imagens seguintes mostram (com a devida autorização da marca para o efeito[12]) o exterior e o interior do aparelho, enquanto se procede à análise técnica e à identificação de componentes cuja função é já conhecida.

[12] Consultar anexo 8.3

Para abrir o equipamento e ter acesso ao circuito, é necessário retirar 4 parafusos em torno da caixa, 2 exteriormente visíveis e outros 2 dentro do compartimento das baterias. É neste compartimento que são ligadas as duas baterias do tipo AAA, com uma tensão por célula de 1.5V, perfazendo a alimentação total do aparelho de 3V.

Após retirar os parafusos, a tampa superior sai facilmente e pela primeira vez consegue ver-se o interior do medidor e o circuito que o compõe. Nesta fase apenas está visível o seu grande ecrã e as quatro *pads* dos botões de pressão.

Figura 76 – Medidor sem tampa

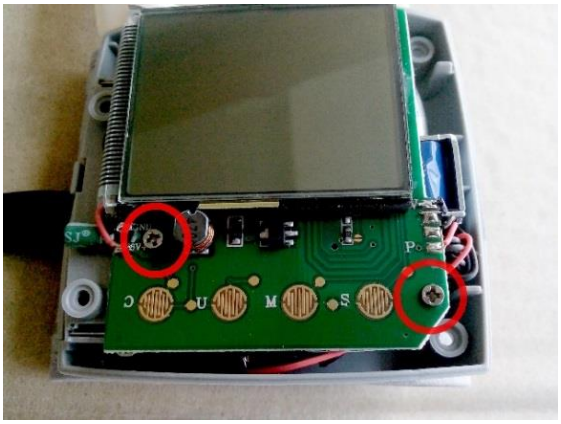


Figura 77 – Top layer do circuito

De modo a que toda a placa com o circuito se destaque e permita identificar o que se procura, a EEPROM, é necessário desapertar mais dois parafusos, assinalados a vermelho. Toda a placa de circuito impresso se desagrega da base do equipamento e podem ser observados os componentes principais, essenciais ao funcionamento do medidor.

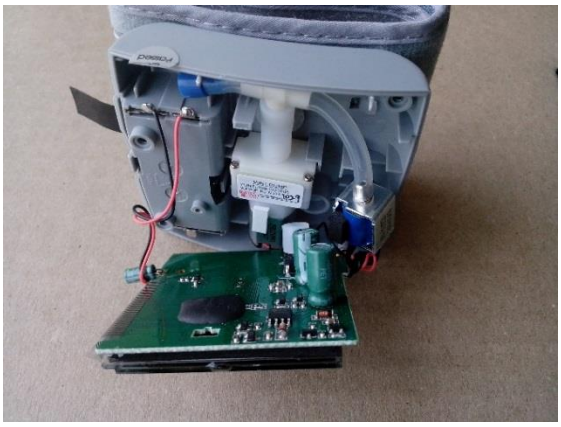


Figura 78 – PCB solta do suporte

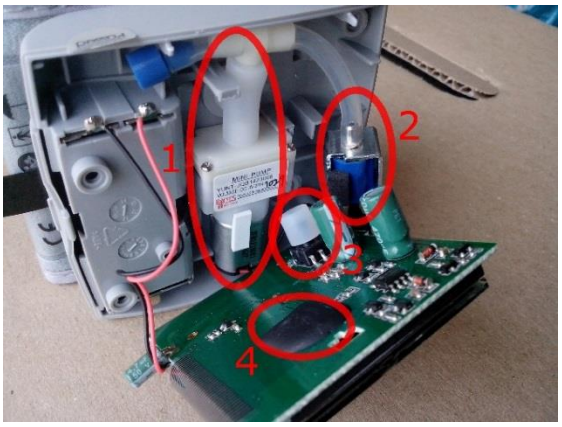


Figura 79 – Elementos principais do medidor

Na Figura 79 pode ver-se o interior do equipamento onde já foram assinalados, a vermelho, alguns componentes conhecidos. Esses componentes são facilmente identificados como:

1 – Bomba de ar;

2 – Válvula de ar;

3 – Sensor de pressão;

4 – Processador, em montagem *chip on board*, com a típica cobertura em epóxi ("glob top");

Como mostra a Figura 80, junto ao processador encontra-se um circuito integrado, tipo SOIC (*Small Outline Integrated Circuit*), com a referência "24C08C".

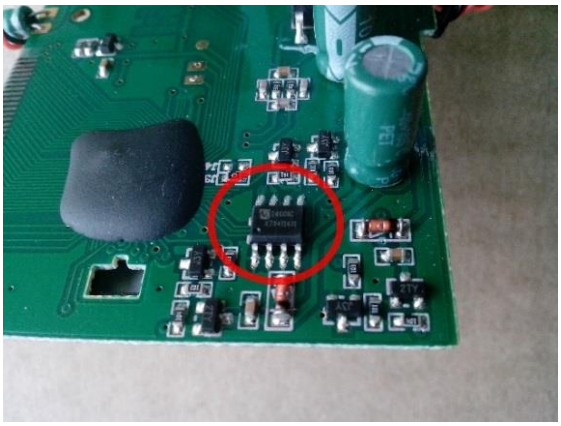


Figura 80 – Pormenor do SOIC, "24C08"

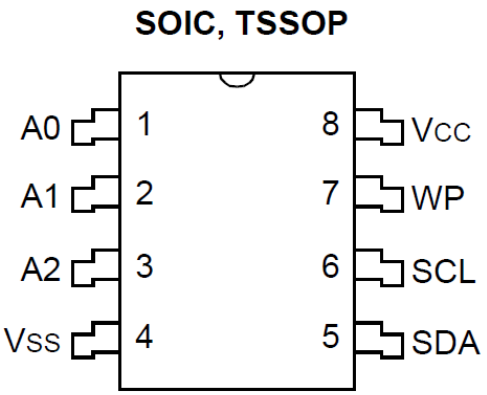


Figura 81 – Pinout do 24LC08

Com essa referência, foi possível encontrar o *datasheet* da Microchip[73], referente à memória "24LC08" e identificar o componente como sendo uma EEPROM de 8Kbit com comunicação I^2C. Embora a referencia não seja exatamente igual o tipo de ligações e o circuito em redor do circuito, não deixa margens para dúvidas, trata-se de facto da EEPROM.

Tal como previsto a memória EEPROM foi encontrada na placa de circuito do equipamento. O objetivo seguinte passa por recorrer a um analisador lógico "pendurado" no barramento I^2C da memória e recolher informação suficiente para interpretar como as medições são guardadas, qual a sua organização, qual o tamanho da *frame*, etc. Para isso é necessário soldar dois pequenos fios nos pinos 5 e 6 da pastilha (Figura 81),pinos SDA e SCL respetivamente[13], para uma fácil interligação ao analisador.

[13] SDA – serial data; SCL – serial clock;

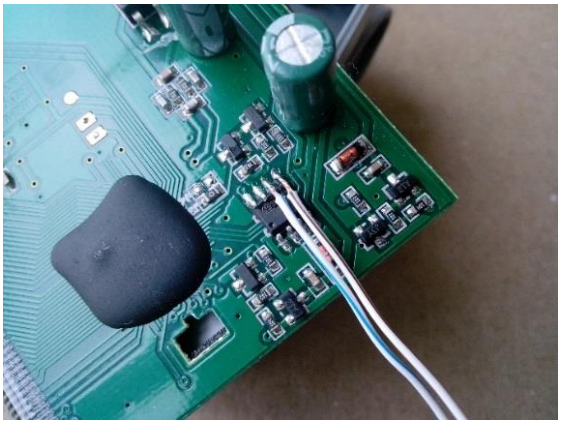


Figura 82 – Ligação dos pinos SDA e SCL

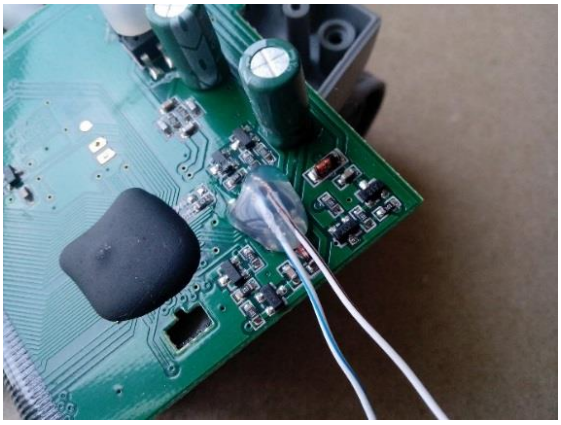


Figura 83 – Pormenor do isolamento das ligações

Devido à sua proximidade e ao facto de com o calor próprio da soldadura (ferro + solda), o isolamento plástico do fio encolhe um pouco. Por isso, para precaução, para dar mais rigidez mecânica e ter liberdade em usar os fios sem a preocupação de danificar a solda ou fazer curto-circuito, toda a zona foi coberta com cola termofusível (Figura 83).

Antes de montar novamente o equipamento, mais duas coisas foram procuradas: um ponto de ligação à massa (para fazer o comum dos circuitos) e a forma de poder iniciar automaticamente uma nova leitura, usando o circuito a ser construído e sem necessidade de pressionar o botão do medidor.

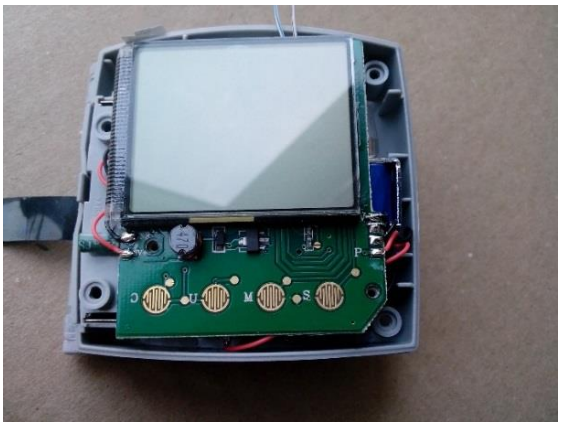


Figura 84 – Pormenor das *pads* do botões

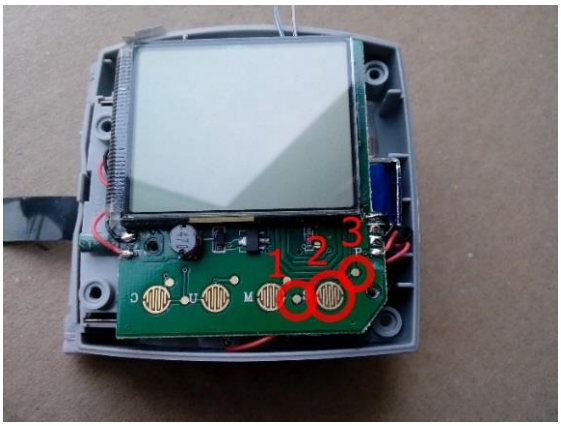


Figura 85 – Pontos de contacto

Procurou-se na *top layer* a *pad* do botão de pressão correspondente à função ON/OFF. Na Figura 85 a *pad* correspondente está assinalada a vermelho e com o número 2. Estes botões de pressão possuem uma almofada feita de um material condutor que quando pressionada baixa até tocar na PCB onde faz contacto com dois pontos. Neste caso, esses dois pontos são o *ground* e uma trilha que está, de certo, ligada a uma resistência de *pull-up*. Quando se pressiona o botão, é feito o contacto entre o ponto 1 (*ground*) e o ponto 3, que deverá estar conectado a um pino do processador e que passará do estado *high* para o estado *low*. Quando isso acontece o

processador sai do modo *sleep* e arranca com todo o processo de medição da pressão arterial, já descrito no subcapítulo 2.3.

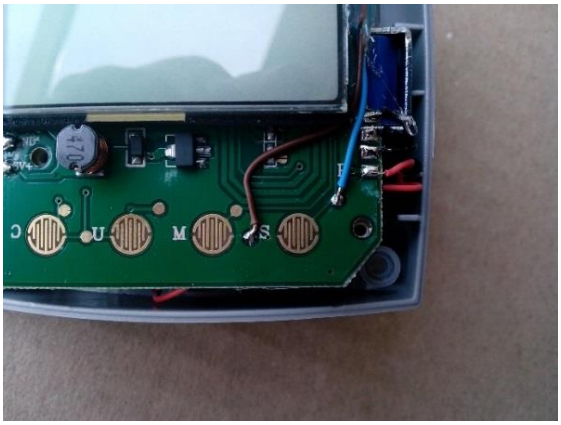


Figura 86 – Fios soldados para função ON/OFF

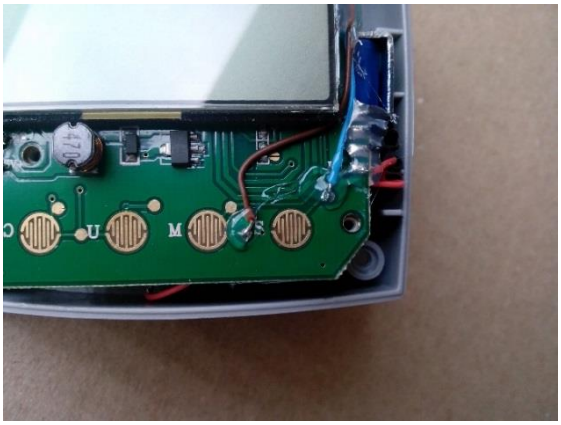


Figura 87 – Isolamento das ligações

Para que pudessem ser facilmente acedidos do exterior, aos pontos 1 e 3 foram soldados dois fios tal como mostra a Figura 86. Pelas mesmas razões dos fios soldados à EEPROM, também estes foram cobertos com cola termofusível para aumentar a resistência física das ligações. Dado que a superfície a soldar é muito pequena e não tem rugosidades para uma melhor aderência, a aplicação da cola neste caso, torna-se ainda mais útil que a do anterior.

Todas as ligações necessárias para o desenvolvimento deste trabalho já se encontram, devido aos fios soldados nos diversos pontos, facilmente acessíveis do exterior do equipamento. As modificações ao aparelho ficam concluídas e o mesmo está pronto para que se inicie o estudo dos pacotes de dados trocados no barramento.

8.2. ESP8266 AT Command Set

Function	*AT Command*	*Response*
Working	AT	OK
Restart	AT+RST	OK [System Ready, Vendor:www.ai-thinker.com]
Firmware version	AT+GMR	AT+GMR 0018000902 OK
List Access Points	AT+CWLAP	AT+CWLAP +CWLAP:(4,"RochefortSurLac",-38,"70:62:b8:6f:6d:58",1) +CWLAP:(4,"LiliPad2.4",-83,"f8:7b:8c:1e:7c:6d",1) OK
Join Access Point	AT+CWJAP? AT+CWJAP="SSID","Password"	Query AT+CWJAP? +CWJAP:"RochefortSurLac" OK
Quit Access Point	AT+CWQAP=? AT+CWQAP	Query OK
Get IP Address	AT+CIFSR	AT+CIFSR 192.168.0.105 OK
Set Parameters of Access Point	AT+ CWSAP? AT+ CWSAP= <ssid>,<pwd>,<chl>, <ecn>	Query ssid, pwd chl = channel, ecn = encryption
WiFi Mode	AT+CWMODE? AT+CWMODE=1 AT+CWMODE=2 AT+CWMODE=3	Query STA AP BOTH
Set up TCP or UDP connection	AT+CIPSTART=? (CIPMUX=0) AT+CIPSTART = <type>,<addr>,<port> (CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port>	Query id = 0-4, type = TCP/UDP, addr = IP address, port= port
TCP/UDP Connections	AT+ CIPMUX? AT+ CIPMUX=0 AT+ CIPMUX=1	Query Single Multiple
Check join devices' IP	AT+CWLIF	
TCP/IP Connection Status	AT+CIPSTATUS	AT+CIPSTATUS? no this fun
Send TCP/IP data	(CIPMUX=0) AT+CIPSEND=<length>; (CIPMUX=1) AT+CIPSEND= <id>,<length>	
Close TCP / UDP connection	AT+CIPCLOSE=<id> or AT+CIPCLOSE	
Set as server	AT+ CIPSERVER= <mode>[,<port>]	mode 0 to close server mode; mode 1 to open; port = port
Set the server timeout	AT+CIPSTO? AT+CIPSTO=<time>	Query <time>0~28800 in seconds
Baud Rate*	AT+CIOBAUD? Supported: 9600, 19200, 38400, 74880, 115200, 230400, 460800, 921600	Query AT+CIOBAUD? +CIOBAUD:9600 OK
Check IP address	AT+CIFSR	AT+CIFSR 192.168.0.106 OK
Firmware Upgrade (from Cloud)	AT+CIUPDATE	1. +CIPUPDATE:1 found server 2. +CIPUPDATE:2 connect server 3. +CIPUPDATE:3 got edition 4. +CIPUPDATE:4 start update
Received data	+IPD	(CIPMUX=0): + IPD, <len>: (CIPMUX=1): + IPD, <id>, <len>: <data>
Watchdog Enable*	AT+CSYSWDTENABLE	Watchdog, auto restart when program errors occur: enable
Watchdog Disable*	AT+CSYSWDTDISABLE	Watchdog, auto restart when program errors occur: disable

* New in V0.9.2.2 (from http://www.electrodragon.com/w/Wi07c)

8.3. Autorização para realização de engenharia reversa

David Martins <davidmartins.dm@gmail.com>

Engineering master's degree project -> Sanitas SBC 21
1 mensagem

David Martins <davidmartins.dm@gmail.com> 23 de abril de 2015 às 23:11
Para: service@sanitas-online.de

Hello, good morning.

My name is David Martins and I am working on my master's degree project of Electrical Engineering, University of Beira Interior, Covilha, Portugal.

My interest was to carry out a project related to health. A few months ago I went into a LIDL store and found the product "Sanitas SBC 21" for sale.

http://www.sanitas-online.de/web/en/products/bloodpressure/wrist/SBC21.php

I have studied your circuit and did some reverse engineering to try to get the values of the readings. I have been successfull and managed to intercept communications (in I2C) processor to EEPROM. I have used this as the basis for my work.

On of the reasons I am contacting you is to know if I can put your company name "Sanitas" in my work.

The other reason, would be to know if you would be willing, in some way, to support the project. With answers to some technical questions about the circuit or even with the possible provision of a better quality measuring equipment.

I'd be happy if I could count with your support.

Thank you for your available time.

With best regards,
--

David Martins, m6350
Engenharia Electrotécnica e de Computadores
Universidade Beira Interior

Gmail by Google

David Martins <davidmartins.dm@gmail.com>

WG: Engineering master's degree project -> Sanitas SBC 21

4 mensagens

Meternek Werner <Werner.Meternek@beurer.de> 27 de abril de 2015 às 12:18
Para: "davidmartins.dm@gmail.com" <davidmartins.dm@gmail.com>

Dear David,

Yes you can use our “Sanitas” name in your project.

Regarding the SBC21, I must say that we do not have detailed information about the circuit because the development was done by an external company.

We would be happy to get your proposals for a better quality measurement equipment.

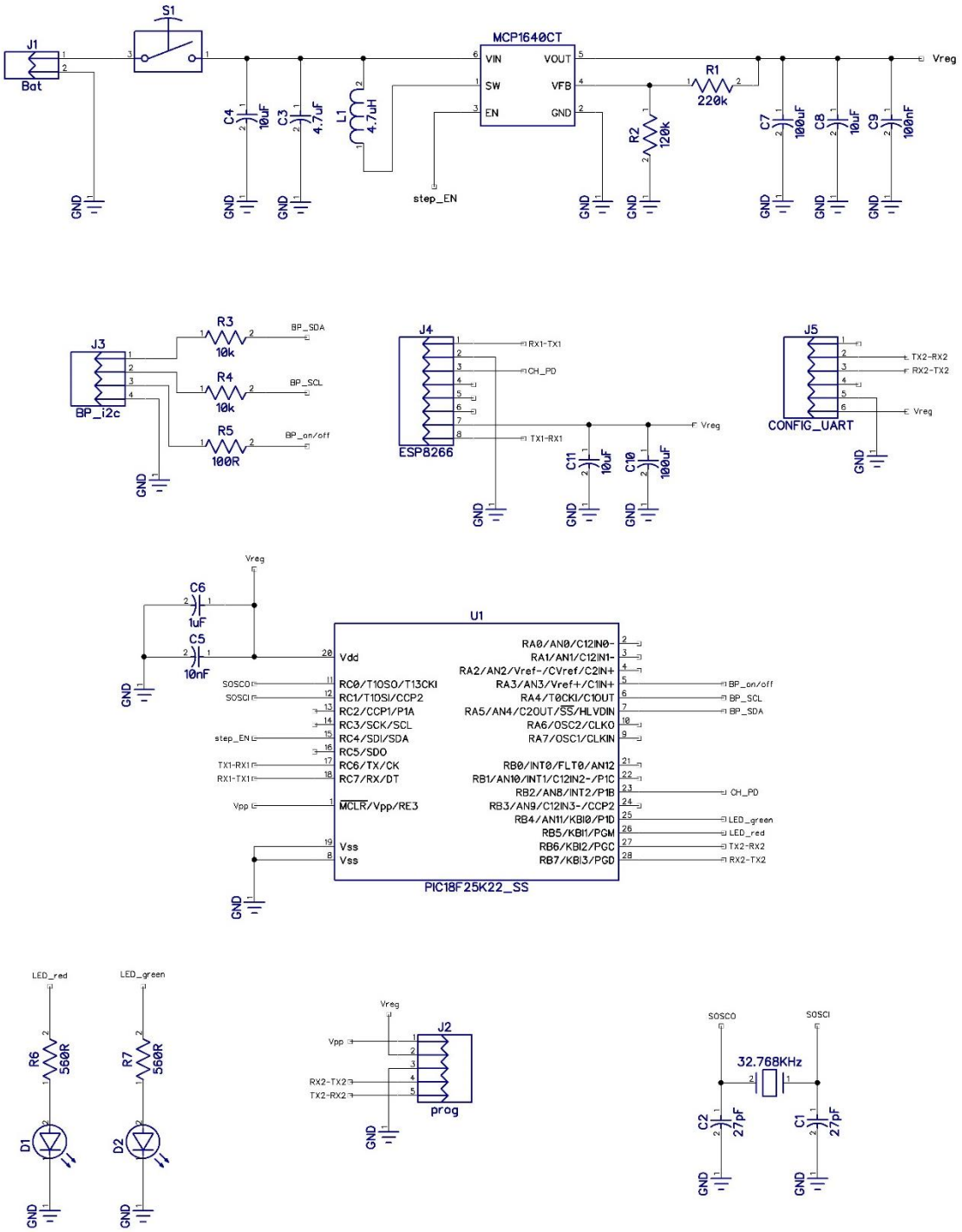
Best Regards

Werner

Dipl.-Ing. (FH) Werner Meternek
Leiter Technologie & Entwicklung/ Director Research & Development

Beurer GmbH * Soeflinger Strasse 218 * 89077 Ulm / Germany
Tel.: +49 731 3989-226 * Fax: +49 731 3989145*www.beurer.com

8.4. Esquemático do circuito protótipo



Title		
Circuito do prototipo		
Size	Number	Rev
Date	Drawn by David Martins	
Filename	Sheet	