

INSTITUTO SUPERIOR TÉCNICO



SISTEMAS EMBEBIDOS EM REDE

2º SEMESTRE 2020/21

Relatório de Projeto

Smart Helmets para explorações subterrâneas

Alexandra Fernandes
nº 90001

Duarte Oliveira
nº 94192

Pedro Antunes
nº 90170

GRUPO: 4

DOCENTE: PROFESSOR CARLOS ALMEIDA

6 de junho de 2021

1 Introdução

Em cenários de exploração mineira ou de escavações arqueológicas podem ocorrer inúmeras situações de perigo para os indivíduos que se aventuram em espaços subterrâneos. Na eventualidade de ocorrer um acidente, a pessoa pode não ter tempo suficiente para comunicar o problema através dos seus dispositivos.

Desta forma, este projeto pretende criar um dispositivo que alerte automaticamente os companheiros de escavação, para que se possam deslocar rapidamente em seu auxílio. Dando um exemplo de cenário plausível, em que o dispositivo alertaria em caso de exposição a gases nocivos, que em segundos podem colocar uma pessoa num estado inconsciente ou a longo prazo desenvolver problemas de saúde. Desta modo, procurou-se desenhar uma solução que seria colocada em capacetes de mineiros ou exploradores subterrâneos. Daqui em diante, ir-se-á referir a este sistema como "capacete" e na próxima secção deste relatório será descrito o que o constitui.

Numa primeira instância, imaginou-se um cenário complexo em que se colocariam vários pontos de acesso ao logo dos túneis ou grutas, de forma a criar uma rede oportunista na qual os vários "capacetes" se poderiam ligar de acordo com a sua proximidade a cada ponto de acesso. Qualquer mensagem seria depois encaminhada através desta rede até outro "capacete", permitindo assim comunicação entre "capacetes" e até ao exterior, em que a caixa mais próxima do exterior ligar-se-ia a uma interface, podendo dar qualquer informação sobre as pessoas no interior a quem estivesse no exterior a controlar a operação.

No projeto desenvolvido realizou-se um sistema que permite a monitorização de cada "capacete" através da internet. É assim possível detectar se algum mineiro se encontra em perigo através dos valores medidos pelos sensores. Para além disso, é possível verificar o estado actual das condições ambientais em que cada um dos mineiros se encontra.



Figura 1: Imagem ilustrativa

2 Arquitetura da rede e dos nós

Cada "capacete" é constituído por dois mbeds ligados através de um barramento CAN. Um dos mbeds estará a fazer as medições dos diversos sensores e de seguida enviará os seus valores através da rede CAN. Para comunicar com os diversos sensores este mbed pode utilizar diferentes interfaces. Este pode ler os valores da qualidade do ar através de uma entrada analógica e obter a temperatura através de uma conexão I2C, por exemplo.

O segundo mbed estará a escuta a verificar se existe alguma mensagem na rede CAN para ler. Se existir vai processar a mensagem e retirar os dados enviados pelo primeiro mbed. À medida que processa os dados actualiza o LCD com os novos valores. Para além disso, os valores dos sensores serão enviados através de uma porta série para o ESP8266. Se existir algum valor fora do normal será enviada uma mensagem de SOS.

O ESP8266 executará a conexão à Internet através de WiFi. Os dados vão ser enviados e publicados num servidor na nuvem, na secção 6 será explicado em mais detalhe o protocolo utilizado.

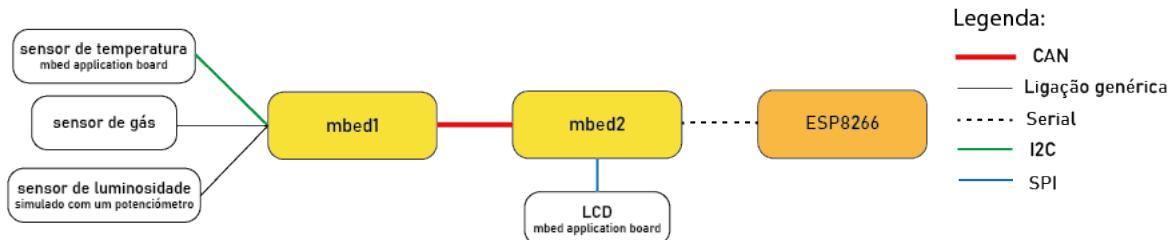


Figura 2: Arquitetura de um nó

Na figura 3, é possível visualizar o cenário global, tanto do caso mais complexo como o caso simplificado. O primeiro caso corresponde às caixas laranja que serviriam como pontos de acesso e comunicariam com a internet através de Ethernet. A ligação destas com os capacetes seria executada por WiFi com o ESP8266. Era ainda ideal que os capacetes conseguissem comunicar entre si. Para isso seria usado um rádio xbee 802.15.4.

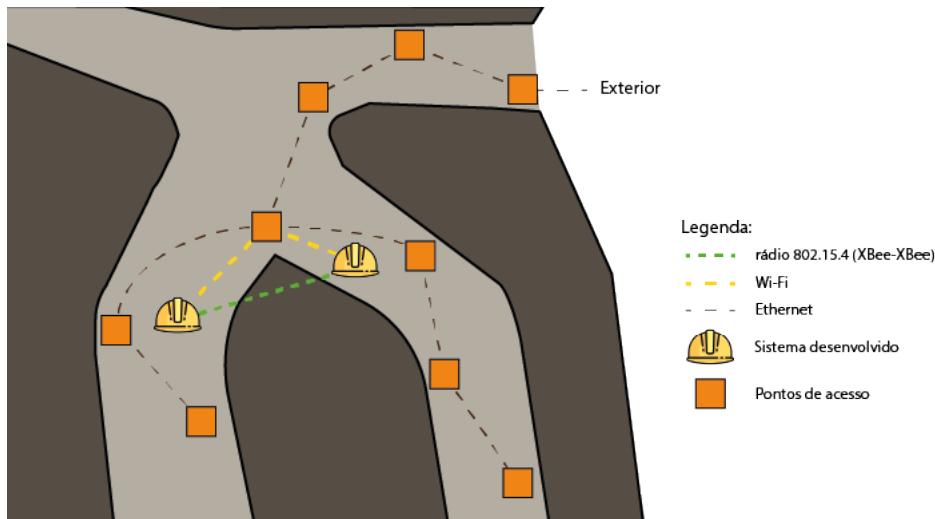


Figura 3: Cenário global

3 Gestão de Energia

Nesta secção será explicado como é executada a gestão do consumo de energia neste projecto. Começou-se por discutir o consumo e as prioridades energéticas de cada componente. De seguida, apresentam-se os requisitos energéticos para manter os dispositivos ligados. Por fim, justifica-se quais as fontes de energia que vão ser utilizadas.

3.1 Gestão do consumo

No projecto executado não existe nenhum mecanismo especial para a gestão do consumo. Os microcontroladores não estão a ser colocados em modo *idle* ou *sleep* de modo a economizar energia. Como os microcontroladores usados têm de estar constantemente à escuta para saber se receberam alguma mensagem, não fazia sentido desactivá-los.

No entanto, algo que poderia ser implementado se existisse um mbed dedicado à leitura dos sensores, era colocá-lo em *idle* ou *sleep* sempre que não estivesse de ler valores.

3.2 Restrições e requisitos específicos

Neste projecto, os seguintes componentes têm requisitos de energia diferentes que são especificados nos pontos seguintes:

- O mbed LPC1768 tem de estar conectado por USB de forma a ser alimentado por 5V ou a uma bateria entre 4.5 e 9V.
- O ESP8266 será alimentado pela saída V_{out} do mbed que fornece 3.3V.
- O sensor de qualidade do ar MQ2 estará conectado aos 3.3V do mbed.
- A *mbed application board* tem como fonte de alimentação o mbed LPC1768. A *mbed application board* contém vários sensores e actuadores que podem ser usados pelo controlador. Quando se utiliza um maior número de periféricos, maior será o consumo energético.

Para além destes elementos tinha-se ainda pensado incluir um Xbee.

- O Xbee tem de ser alimentado por uma fonte de 3.3V. Neste caso, o Vin e o Gnd do Xbee estarão conectados aos pinos respectivos no mbed. De modo a facilitar a testagem do protótipo o módulo Xbee será conectado directamente à *mbed application board*.

3.3 Fontes de Energia

Num cenário real, o projecto teria como fonte de alimentação uma bateria. No entanto, para a realização de testes e no âmbito desta unidade curricular, mantêm-se os mbed's conectados a um computador, via USB.

O ESP8266 tem como fonte de energia o mbed, contudo, se estiver conectado ao computador utilizará a porta USB para alimentação.

Todos os sensores presentes na *mbed application board*, o LCD e o sensor MQ2 serão alimentados somente pelo mbed.

4 Suporte e requisitos especiais do sistema

Nesta secção, irá-se abordar os requisitos do sistema, bem como o melhor método para lidar com as suas necessidades.

4.1 Requisitos de tempo-real

A nível de requisitos, há que ter especial atenção à prioridades das mensagens. As mensagens mais relevantes para este trabalho são as mensagens de SOS dos mineiros. Deste modo, deve-se minimizar o intervalo de tempo entre o momento em que é detectada uma situação de perigo e o envio de uma mensagem de SOS, juntamente com o tempo de recepção desta pelos outros "capacetes". Assim, no desenvolvimento do código para o mbed teve-se em atenção que a *thread* que realiza o processamento das mensagens tem de ser a de maior prioridade.

A transmissão por CAN, realizada entre o mbed que mede os valores dos sensores e o mbed que realiza a transmissão da mensagem de SOS, por si só é rápida e segura o suficiente. No entanto, no mbed que recebe os valores é necessário ler a mensagem assim que esta estiver disponível. Deste modo, a leitura dos dados na rede CAN é executada na *thread* principal.

A leitura dos sensores no projecto final está a ser feita de com um intervalo de medição entre cada sensor de 1 segundo. O período de leitura pode ser reduzido. A *thread* que executa a leitura dos sensores também é responsável por enviar os dados por CAN. Esta tem a mesma prioridade que a *thread* que executa a leitura por CAN.

4.2 Suporte de concorrência

No projecto desenvolvido, a maneira como o mbed faz a gestão das suas diversas atividades é através de *threads*. Existe uma *thread* que executa a leitura dos sensores e envia os seus valores por CAN ao outro mbed. Outra das *threads* está apenas encarregue de receber a informação enviada por CAN e de seguida coloca-a numa "mail-box". Esta "mail-box" vai ser lida pela *thread* que faz o processamento de mensagens. A *thread* do processamento de mensagens verifica se existe alguma coisa para ler na "mail-box". Se existir, a mensagem vai ser lida, obtendo assim o valor do sensor a que esta corresponde. Esta informação vai de seguida ser enviada para o ESP8266 e exibida no LCD.

No ESP8266 verifica-se constantemente se foi recebida alguma informação na porta série. A mensagem que está a ser enviada para o ESP8266 contém um *char* que indica o início da mensagem e outro que indica o fim. Entre estes dois identificadores, vai estar um *char* que indica o tópico a que se referem os dados da mensagem transmitida. Se necessário, este *char* é seguido pelo valor medido por um sensor. Assim que o ESP8266 receber a mensagem na totalidade, vai processá-la. Se existir alguma mudança dos valores anteriormente lidos o ESP8266 vai publicá-los na *cloud*. Para além disso o ESP8266 também vai processar mensagens da *cloud*.

4.3 Tipo de sistema operativo

No âmbito deste projecto vai-se tirar partido do mbed OS. Este é um RTOS (*Real-time operating system*) desenvolvido pela ARM para funcionar com as suas placas de desenvolvimento, por exemplo, o mbed LPC1768, e estas por sua vez funcionam com os microcontroladores à base de CPU's da ARM.

O mbed OS permite realizar operações de *multi-tasking*, usando *threads*, "mail-boxes", mutex's e semáforos. Outra vantagem da utilização deste sistema operativo, é ter sido desenhado para funcionar com um baixo consumo energético. Para além disso, o mbed OS suporta vários tipos de comunicação e já tem *device drivers* desenvolvidos para os diferentes tipos de rede (por exemplo, I2C, CAN, SPI e portas Série). Desta forma, este sistema operativo torna-se ideal para o desenvolvimento de aplicações de IoT (i.e. Internet das Coisas).

4.4 Gestão de recursos

A nível dos recursos, em cada "capacete" serão necessários: dois mbeds, um ESP8266, uma *mbed application board* (que inclui o LCD, um sensor de temperatura e um potenciômetro) e dois *transcievers* CAN. Para além disso, em termos de material auxiliar para o protótipo, será utilizada uma *breadboard* e *jumpers* para realizar as conexões físicas.

4.5 Ambiente de desenvolvimento

Para o desenvolvimento da aplicação, que estará em execução nos mbeds recorre-se ao mbed CLI.

O mbed CLI é uma ferramenta que se utiliza no terminal. Para criar o ambiente de desenvolvimento, tem de se iniciar um projecto novo importando a versão mais recente do mbed OS. De seguida, precisa-se de adicionar as bibliotecas já disponíveis para funcionar com o LCD C12832 e com o sensor de temperatura LM75B. Por fim, cria-se o ficheiro "main.cpp" onde se desenvolveu o código para funções específicas deste projeto e no ficheiro "main.h" foram definidos os códigos das mensagens como variáveis globais e importadas as bibliotecas necessárias.

No desenvolvimento do código a executar no ESP8266 utilizou-se o Arduino IDE. Tendo sido importadas as bibliotecas do ESP8266WiFi, EEPROM e MQTT. A biblioteca ESP8266WiFi é utilizada para conectar a rede Wi-fi. A biblioteca EEPROM é utilizada para comunicar com a memória não volátil do ESP8266. E por fim, a biblioteca do MQTT é disponibilizada pela EasyIOT cloud e está adaptada para funcionar com os seus serviços.

5 Material utilizado

Nesta secção, irão abordar-se as especificações e requisitos dos vários materiais a utilizar

5.1 Placa de desenvolvimento mbed LPC1768

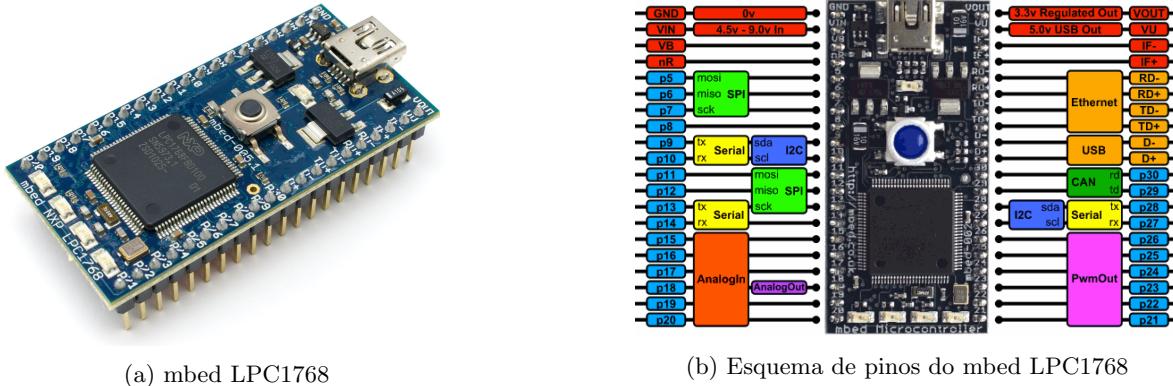


Figura 4: Características do mbed LPC1768

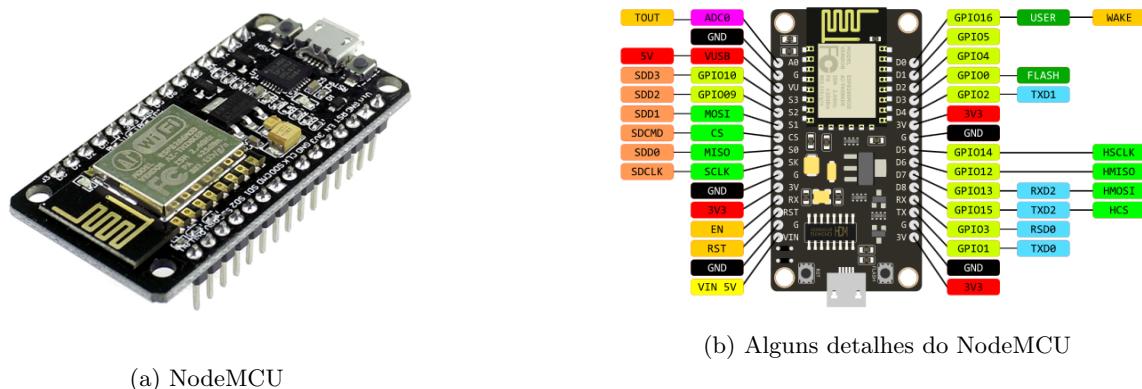
A placa de desenvolvimento mbed LPC1768 apresentado na figura 4 é desenhada para funcionar com vários tipos de dispositivos, entre os quais se podem destacar os que possuem USB, Ethernet e memória FLASH. Esta placa é baseada num micro-controlador de 32-bit designado por ARM Cortex-M3 de 96MHz e é muito completo devido às inúmeras funcionalidades que pode apresentar. Destas destacam-se as seguintes:

- Porta Ethernet integrada;
- USB Host & Device;
- Controladores: CAN, SPI, I2C;
- Conversores: ADC, DAC;
- Interfaces de entrada/saída (I/O).

Este módulo é fundamental para o projeto proposto visto que apresenta, como referido na lista anterior, um controlador CAN, que permite a comunicação em rede CAN que se visa atingir. É possível verificar as suas especificações em termos do próprio *hardware* e *software* a utilizar para a posterior programação do dispositivo. Note-se:

- Dimensões: 54x26mm;
- Memória: 32kB para RAM e 512kB para FLASH;
- Tensão de alimentação: 5V via USB ou 4.5-9V recorrendo a uma fonte externa;

5.2 Placa de desenvolvimento NodeMCU v3



O NodeMCU é uma placa de desenvolvimento *open-source* com um micro-controlador da família ESP8266, apresentado na figura 5a. Contrariamente a algumas placas desta família, esta não necessita de um conversor USB *serial* externo para trocar informação entre o computador e o dispositivo, visto que já inclui um conversor USB integrado, algo bastante vantajoso. Este dispositivo é constituído por:

- Módulo WiFi ESP8266-12E;
- Porta micro-USB que pode ser utilizada como meio de alimentação ou de programação;
- Conversor USB *serial* integrado;

Para além disto, é possível observar algumas características deste microcontrolador na figura 5b e na tabela 1 os vários pinos visíveis no dispositivo, assim como a sua função e os pinos correspondentes no ESP8266-12E. Sobre as suas especificações, é possível notar os seguintes pontos:

- Temperatura de funcionamento: -40°C até 125°C;
- Dimensões: 4.8 x 2.5cm;
- O processador ESP8266-12E apresenta uma Arquitetura RISC de 32-bits e pode operar num alcance 80MHz-160MHz;
- Memória: 4MB de memória flash, 64kB para instruções e 96kB para dados;
- O módulo WiFi nativo é 802.11b/g/n;
- Tem 11 pinos digitais e 1 analógico com resolução de 10 bits;
- Os pinos operam a 3.3V (nível lógico);

Pino	Função	Pino ESP8266-12E
TX	TXD	TXD
RX	RXD	RXD
A0	Input analógica	A0
D0	Input/Output (I/O)	GPIO16
D1	I/O - SCL	GPIO5
D2	I/O - SDA	GPIO4
D3	I/O - 10K PULL-UP	GPIO0
D4	I/O - 10K PULL-UP/BUILTIN_LED	GPIO2
D5	I/O - SCK	GPIO14
D6	I/O - MISO	GPIO12
D7	I/O - MOSI	GPIO13
D8	I/O - 10K PULL-UP/SS	GPIO15
GND	Ground	GND
5V	VCC 5V	-
3V3	VCC 3.3V	3V3
RST	Reset	RST

Tabela 1: Informações sobre os pinos do NodeMCU

5.3 XBee 802.15.4 Wire Antenna - Series 1



Figura 6: XBee 802.15.4 - Series 1

Antes de se referirem as características do XBee é importante notar que este não chegou a ser utilizado no projeto final. No entanto as suas características foram estudadas com vista a o implementar.

O XBee 802.15.4 - Series 1 é um *chip* que consegue comunicar com outros dispositivos de forma *wireless* utilizando, portanto, ondas de rádio. Estes módulos XBee utilizam o protocolo IEEE 802.15.4 essencialmente para conexões *peer-to-peer* e *point-to-multipoint*. Estão projetados para aplicações de alto rendimento que requerem, portanto, uma latência muito baixa, isto é, o melhor tempo de comunicação. Por defeito, não necessitam de configuração e são compatíveis com *gateways* e *range extenders*, sendo portanto muito versáteis, podendo-se complementar o seu uso com um leque variado de componentes. Em termos de especificações, é de realçar os seguintes aspetos:

- Dimensões: 27x24mm;
- Alcance *indoor*: aproximadamente 60m;
- Alcance *RF Line-of-sight*: aproximadamente 1,2km;
- Tensão de alimentação: 2.1 a 3.6V
- Corrente para VDC = 3.3V: 33mA (transmissão) e 28mA (recepção);
- Frequência: 2.4GHz;
- Potência de transmissão: 6.3mW no seu máximo;
- Ritmo de transmissão de dados: entre 250kbps (RF) e 1Mbps (Serial);
- Antena: PCB.

5.4 mbed application board

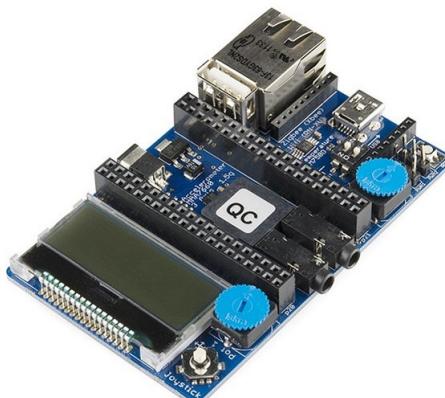


Figura 7: mbed application board

Este dispositivo serve de suporte para o componente referido na secção anterior e apresenta um conjunto de *sockets* de comunicação e vários periféricos. Indubitavelmente é uma placa muito completa, apresentando as seguintes funcionalidades:

- Módulo mbed LPC1768;
- Backlit LCD (128 x 32);
- 9 VDC *power socket*;
- RJ45 Ethernet;
- 2 x 3.5mm *audio jack* para I/O analógico;
- Mini-USB para periféricos;
- 2 potenciómetros;
- Acelerómetro de 3 eixos;
- Sensor de temperatura;
- LED RGB;
- Interface SPI;
- *Joystick* para navegação;
- USB-A para *flash drives* e *Bluetooth*;

No projecto desenvolvido é utilizado o sensor de temperatura, um potenciómetro e o LCD da placa.

5.4.1 Sensor de Gás MQ-2

O sensor de gás MQ-2, apresentado na figura 9, é um dispositivo que apresenta uma precisão imensa para a deteção de Metano, Butano, Gás de Petróleo Liquefeito (GPL) e fumo. Estes gases podem ser prejudiciais para a saúde humana, principalmente em percentagens significativas. O objetivo do uso deste componente é a monitorização da situação respiratória dos utilizadores dos *Smart Helmets* a grandes profundidades, tal como o cenário global do projeto o enuncia. Para além deste sensor, podiam-se utilizar conjuntamente outros sensores da família MQ [MQ]. Desta forma, qualquer anomalia ao nível da saúde pode ser detetada a tempo de tomar ação.



Figura 8: Sensor MQ-2

5.4.2 Transciever CAN - MCP2551

O MCP2551 é um *transciever* CAN de alta velocidade. Este é um dispositivo tolerante a falhas e serve como interface entre um controlador CAN e o barramento físico. O MCP2551 fornece capacidade de transmissão e recepção diferencial para o controlador CAN e é totalmente compatível com o padrão ISO-11898, incluindo requisitos de 24V. O *transciever* opera em velocidades de até 1 Mb / s.

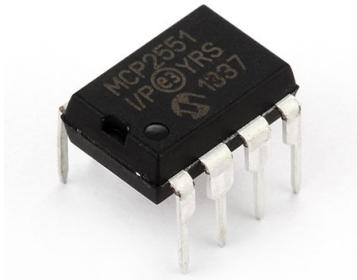


Figura 9: Transciever CAN - MCP2551

6 Comunicações da Rede

6.1 CAN

O protocolo da rede CAN é CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), ou seja, cada nó no barramento consegue detectar colisões e esperar um certo período de tempo antes de retransmitir mensagens. Essa detecção de colisão é obtida por meio de uma arbitragem de prioridade com base nos identificadores de mensagem.

Numa rede CAN, um nó é responsável por transmitir e receber as mensagens disponíveis no barramento. Para a execução desta comunicação, são necessários 2 componentes: o micro-controlador e o *transciever* CAN. O micro-controlador a usar contém o controlador CAN. Porém, a placa de desenvolvimento não apresenta *transcievers*, daí utilizar os MCP2551 à parte, como referido na secção anterior deste relatório.

A rede utiliza um sinal diferencial com dois estados lógicos, um dominante e outro recessivo. O recessivo indica que a tensão diferencial é menor que a mínima tensão *threshold*. O dominante é, portanto, o oposto. Porém, para o dominante ser activado o barramento tem de estar a '0' e o recessivo a '1', contrariamente ao que acontece nos sistemas tradicionais. O facto de usar sinais diferenciais torna a comunicação muito mais resistente ao ruído, principalmente em zonas subterrâneas onde há interferências electromagnéticas significativas.

Para implementar a rede CAN, tem de se ligar os dois pinos do *mbed* que são controladores CAN, rd/p30, td/p29, aos pinos do *transciever*, RXD/4, TXD/1 respectivamente. Os pinos p9 e p10, que funcionam como pinos de Serial também podem funcionar como controladores CAN.

6.2 MQTT

O protocolo de comunicação MQTT (*Message Queue Telemetry Transport*) é construído sobre TCP/IP e tem um modelo de *publish* e *subscribe*. São definidos dois tipos de entidades neste protocolo: os *brokers* e um número de clientes. A função do *broker* é receber todas as mensagens dos clientes e depois direcioná-las para os destinatários corretos. Por outro lado, os clientes podem ser qualquer entidade que interaja com o *broker* e que possa receber mensagens, pode ser por exemplo uma aplicação móvel que processa dados vindos da *cloud*.

Algo muito curioso e vantajoso deste protocolo é que os clientes após se conectarem a um *broker* podem subscrever a tópicos desse *broker*. Para além disso, outros clientes ou os mesmos podem publicar mensagens para um certo tópico. Quando isto acontece, o *broker* reencaminhará essas mensagens para todos os clientes subscritos a esse tópico.

Neste projeto, utilizou-se este protocolo conjuntamente com a plataforma *EasyIoT* [Eas]. O *broker* com o qual o microcontrolador ESP8266 comunica encontra-se nos servidores da EasyIOT cloud. O ESP8266 publica os dados que recebe de um dos mbeds nos respetivos tópicos nesse *broker*.

Esta plataforma permite visualizar em tempo real estes dados e exibi-los num gráfico temporal, como se pode ver na figura 10. Para além disto, foi criado um módulo com um botão que é ativado quando o valor de luminosidade é inferior a 50%. É possível visualizar na figura 10 o momento em que se mudou o valor da luminosidade rodando o potenciômetro de modo a que tivesse um valor inferior ao mencionado anteriormente. E note-se na figura 11 a alteração no módulo associado à luminosidade no exato momento em que é rodado o potenciômetro (os tempos estão assinalados dentro de caixas verdes, é de realçar que coincidem com os do gráfico apresentado na figura 10).

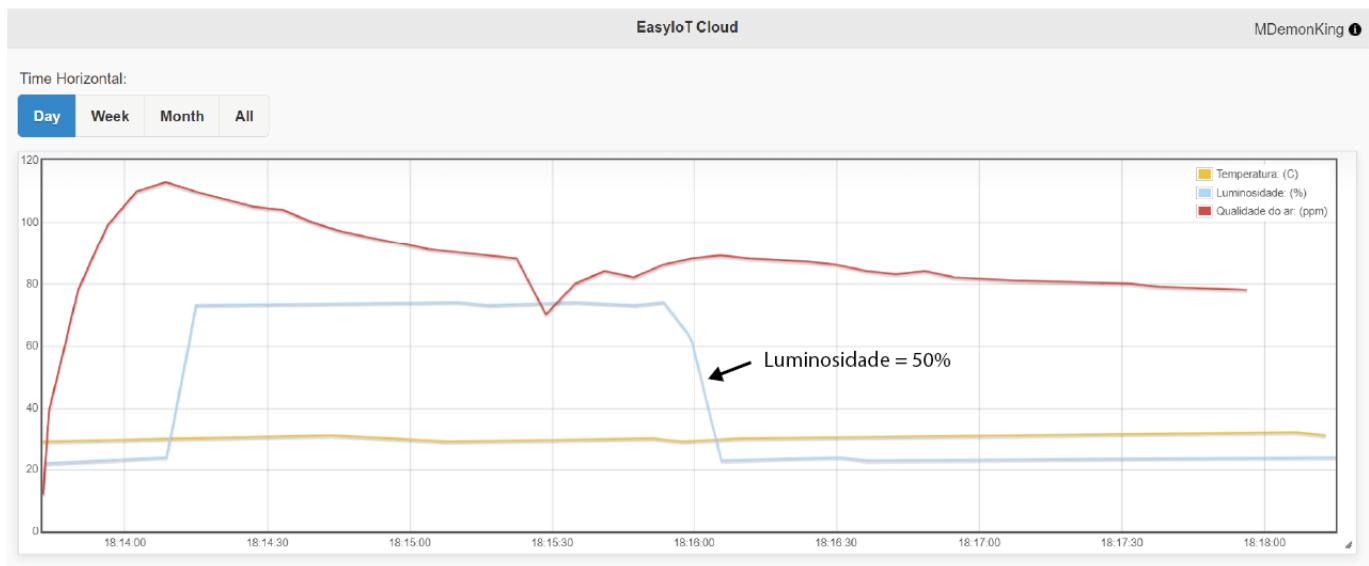


Figura 10: Printscreen dos gráficos temporais na plataforma EasyIot

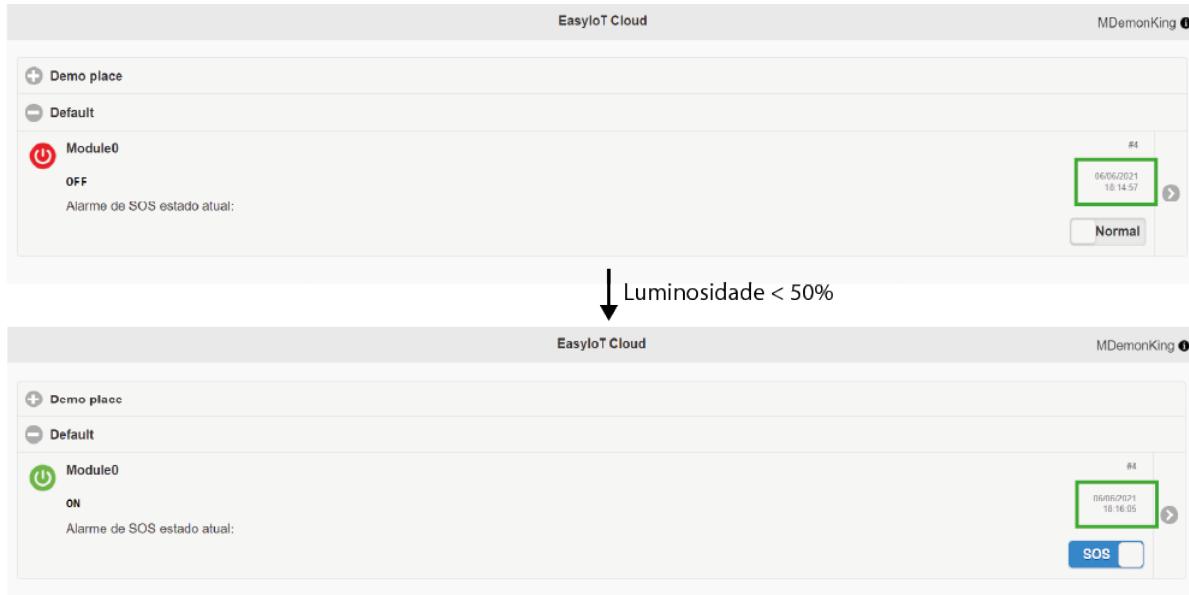


Figura 11: Printscreen do módulo criado na plataforma EasyIot

6.3 Serial

Neste projeto, a comunicação *Serial* é utilizada entre o mbed2 da figura 2 e o NodeMCU. Foram utilizados os pinos p9(tx) e p10(rx) do mbed aos pinos RX e TX do NodeMCU, respectivamente. Apenas se utiliza o pino p9 do mbed e o TX do NodeMCU, dado que apenas o NodeMCU necessita de receber dados do mbed. No futuro, pretendia-se utilizados a outra conexão para que o mbed pudesse receber feedback do NodeMCU que está conectado à Internet utilizando o microcontrolador ESP8266.

6.4 SPI

A comunicação SPI é utilizada entre o LCD da *mbed application board* e o mbed. Os pinos do mbed utilizados foram os MOSI:p5, nRESET:p6, SCK:p7 e A0:p8.

6.5 I2C

Esta comunicação foi utilizada entre o sensor de temperatura da *mbed application board* e o mbed. Os pinos do mbed utilizados foram os SCL:p27 e SDA:p28.

7 Conclusões

Algumas alterações foram feitas relativamente ao estado intermédio, nomeadamente:

- Concluiu-se que era impossível utilizar o XBee dado que os pinos necessários já eram utilizados para a comunicação por CAN;
- No entanto, encontrou-se uma solução viável que envolveu um microcontrolador ESP8266;
- Esta solução trouxe algumas vantagens como ter uma interface gráfica prática e visualmente agradável,
- No contexto prático deste projecto, esta interface poderia ser utilizada pelos supervisores no exterior das grutas, podendo ver o estado de cada um dos "capacetes" em tempo real;
- Outra vantagem da solução encontrada é que permite ter mais subscritores diferentes, como por exemplo, uma aplicação móvel;

7.1 Notas finais dos autores

Este projecto foi concluído com sucesso, no entanto, tem abertura para várias melhorias. Uma das coisas que gostávamos que fosse implementada era a possibilidade dos capacetes comunicarem entre si sem recorrer a Internet. Desta forma seria possível comunicar um sinal de SOS mesmo que algum dos capacetes tivesse perdido a conexão Wi-Fi. Para além disso, ainda era possível fazer um estudo mais aprofundado sobre o consumo energético e verificar onde podíamos poupar energia.

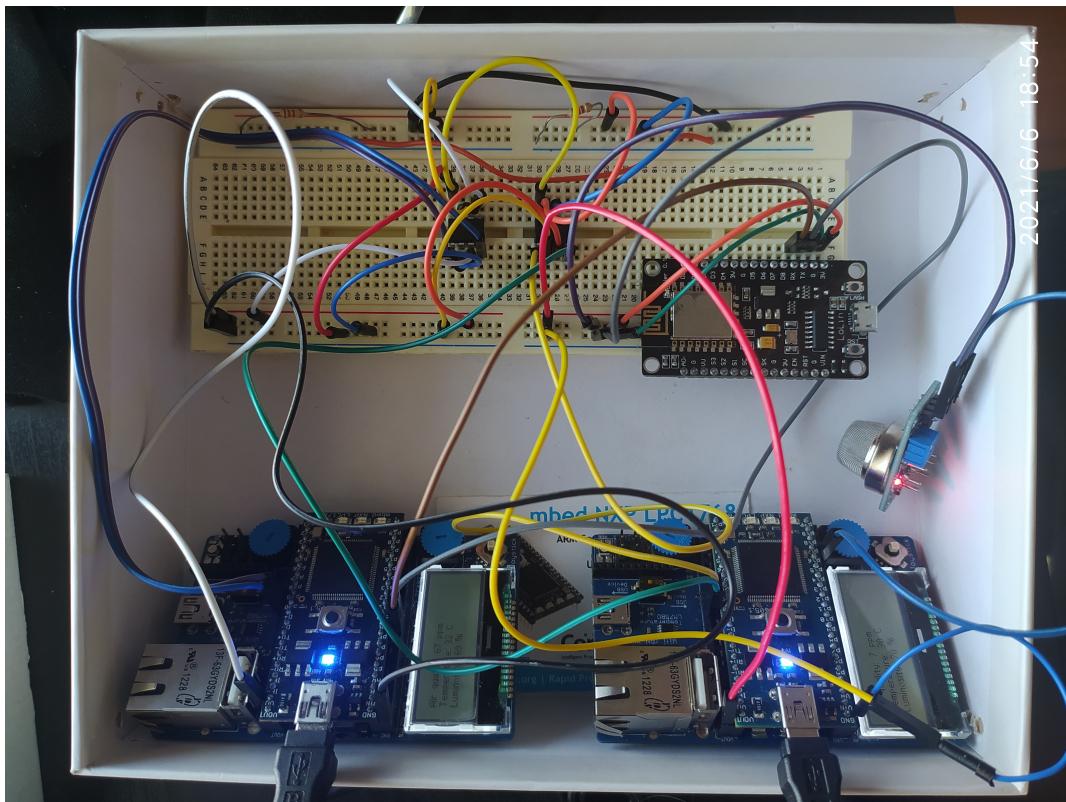


Figura 12: Montagem final

Referências

- [boa] mbed application board. URL: <https://os.mbed.com/components/mbed-Application-Board/>.
- [Eas] EasyIoT. <https://easyiot-cloud.com> and <https://iot-playground.com>.
- [fee] Leaky feeder. URL: https://en.wikipedia.org/wiki/Leaky_feeder.
- [Glo] MTS Global. URL: <https://os.mbed.com/users/no2chem/notebook/mbed-power-controlconsumption/>.
- [GC] Xbee Guide e Power Consumption. URL: https://www.sparkfun.com/pages/xbee_guide.
- [LPC] mbed LPC1768. URL: <https://os.mbed.com/platforms/mbed-LPC1768/>.
- [mbe] Website mbed. URL: <https://mbed.org/>.
- [Mic] Microchip. *MCP2551*. <https://ww1.microchip.com/downloads/en/DeviceDoc/20001667G.pdf>.
- [MQ] Família de sensores MQ. URL: <https://components101.com/sensors/mq2-gas-sensor>.
- [MQT] Protocolo MQTT. <https://developer.ibm.com/technologies/messaging/articles/iot-mqtt-why-good-for-iot/> and <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/>.
- [OS] mbed OS Power Management. URL: <https://os.mbed.com/cookbook/Power-Management>.
- [Wei] Michael Wei. *mbed LPC1768 power consumption*. URL: <https://os.mbed.com/users/no2chem/notebook/mbed-power-controlconsumption/>.