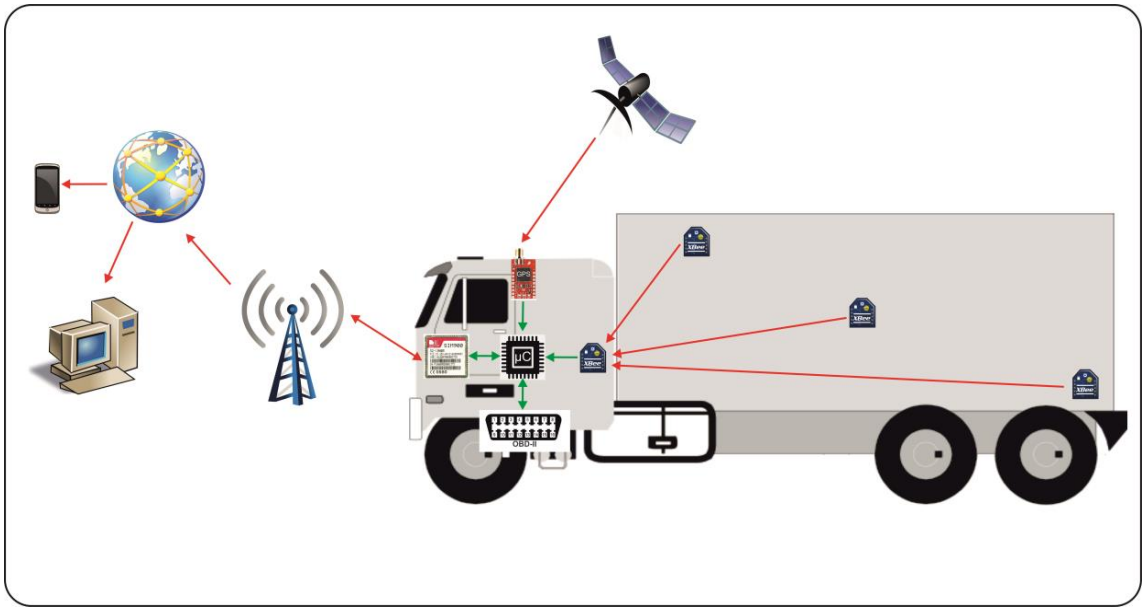
UNIVERSIDADE DO ALGARVE

UNIVERSIDADE DO ALGARVE

INSTITUTO SUPERIOR DE ENGENHARIA



Sistema Remoto para Monitorização de Ambientes Refrigerados em Viaturas Automóveis

Sérgio Ricardo Martins Vasconcelos

Dissertação para a obtenção do grau de Mestre em

Engenharia Elétrica e Eletrónica

Especialização em Tecnologias de Informação e Telecomunicações

Orientador: Professor Doutor Jorge Filipe Leal Costa Semião

Setembro, 2014

Título:

Sistema Remoto para Monitorização de Ambientes Refrigerados em Viaturas Automóveis.

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

_____

Sérgio Ricardo Martins Vasconcelos

AGRADECIMENTOS

Muitas foram as pessoas que desempenharam um papel fundamental, direta ou indiretamente, para que me fosse possível chegar à etapa de poder desenvolver este trabalho bem como para a conclusão do mesmo. Desta forma, não poderia senão registar os devidos agradecimentos.

Ao meu orientador, Professor Doutor Jorge Semião, pela disponibilidade e sinceridade ao longo deste trabalho e antes dele.

Ao Professor Doutor Pedro Cardoso, o primeiro a preocupar-se em dar um passo fora da rotina das aulas e olhar para a pessoa dentro do aluno. Pelos seus conselhos e motivação sem os quais não teria chegado até aqui. Também pela oportunidade de reinserção que me permitiu retomar este trabalho.

Ao meu amigo e colega, João Azinheiro, pelo apoio, pelas horas de estudo e trabalhos conjuntos e pelo espírito de equipa sem o qual não teria chegado a este ponto do caminho.

Aos meus pais, por não terem deixado de acreditar e por todos os outros motivos que possam vir à cabeça para se agradecer aos pais.

À minha companheira, Iva, por compreender este agradecimento sem que eu tenha que adicionar mais palavras ao mesmo.

Faro, 30 de Setembro de 2014

Sérgio Vasconcelos

RESUMO

Devido à constante necessidade das empresas de frotas de distribuição em aumentarem a eficiência dos seus sistemas através de processos de otimização, bem como o controlo de custos associado, os sistemas de monitorização remota e *tracking* de viaturas estão em franca evolução e expansão.

Este trabalho, integrado no projeto i3FR - QREN n. 34130, que envolve o consórcio entre duas empresas do Algarve, a X4DEV e a Aviludo, e o Instituto Superior de Engenharia da Universidade do Algarve, tem como objetivo desenvolver um módulo de aquisição e monitorização remota de dados de sensores, nomeadamente temperatura e câmaras frigoríficas de entrega de comida fresca e congelada. Estes dados serão integrados numa solução de otimização e gestão de rotas, não englobado no âmbito deste trabalho.

Foram estudados os tipos de sensores de temperatura existentes, de modo a selecionar o mais conveniente para os ambientes em questão, ou seja, temperaturas de até -20ºC. Segue-se de um estudo das tecnologias e protocolos de comunicação utilizados para a comunicação dos sensores com o módulo principal e deste com a central. Sendo um dos requisitos a instalação de sensores sem fios, foi escolhida a tecnologia ZigBee. Foi também acrescentado ao objetivo inicial as funcionalidades de localização por GPS, leituras de dados do veículo através de OBD-II e comunicação GSM/GPRS, para uma monitorização em tempo real.

Foi desenvolvido todo o *hardware* e respetivo *firmware*, sobre a plataforma *mbed*, uma plataforma de prototipagem rápida dotada de um microcontrolador baseado no processador *ARM Cortex M-3*, tendo as funcionalidades implementadas sido testadas e utilizando um monitor série para se visualizar os dados.

Palavras-chave: Monitorização remota de parâmetros, localização remota de viaturas, *tracking*, redes de sensores, Otimização de rotas de entrega, *On-Board Diagnostics*

ABSTRACT

Due to the constant need of increasingly efficiency by the distribution fleet companies, optimization and control associated costs, remote monitoring systems and vehicle tracking companies are in frank evolution and expansion.

This work, part of the project i3FR - QREN n. 34130, which involves a consortium of two companies in the Algarve, X4DEV and Aviludo, and the Instituto Superior de Engenharia, University of Algarve, aims to develop a data acquisition and remote monitoring of sensor data, such as temperature and cold delivery of fresh and frozen food. These data will be integrated into an optimization and route management solution, not encompassed in this work.

Different types of existing temperature sensors were studied, to select the most suitable for the environments concerned by this work, i.e., temperatures down to -20°C. Next, a study was made on the technologies and communication protocols to use for communication between the sensors and the main module, and this with the central. Being one of the requirements to install wireless sensors, ZigBee technology was chosen. It was also added to the initial objectives, the vehicle tracking through GPS, readings of vehicle data through OBD-II and GSM/GPRS communication for real-time monitoring.

All relevant hardware and firmware were developed, based on the mbed rapid prototyping platform, provided with a microcontroller based on the ARM processor Cortex M-3 processor, having the features implemented been tested using a serial monitor to display the data.

Keywords: Remote parameter monitoring, remote vehicle tracking, sensor networks, delivery routes optimization, on-board diagnostics

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ABREVIATURAS

ADC	Analogic to Digital Converter
BOM	Bill of Materials
CAN	Controller-Area Network
CEPT	Conférence Européene des Postes et Telecomunication
CSS	Cascade Style Sheet
ECU	Engine Control Unit
ETSI	European Telecommunications Institute of Standardization
GGA	Global Positioning System Fix data
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global Service for Mobile
MMC	Multimedia Cards
PHP	Hypertext Preprocessor
I2C	Inter-Integrated Circuit
IP	Internet Protocol
LiFePo4	Fosfato de Ferro – Lítio
LSB	Least Significant Byte
M2M	Machine-to-Machine
MSB	Most Significant Byte
OBD	On-Board Diagnostics
PCB	Printed Circuit Board
PID	Parameter Identifier
PWM	Pulse Width Modulation
SMPS	Switched Mode Power Supply
SMS	Short Message Transfer
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
WPAN	Wireless Personal Area Network

1. INTRODUÇÃO

No mundo atual, existe uma crescente procura do aumento de eficiência na execução de tarefas, sejam elas simples tarefas do dia-a-dia, ou sistemas complexos, o que requer a adaptação e a evolução dos próprios sistemas. Atendendo a esta evolução, os sistemas de monitorização remota desempenham um papel fundamental em diversas áreas de atuação, permitindo um maior controlo e observação sobre as condições dos produtos, bens ou pessoas, em tempo real e à distância. Estes sistemas, genericamente, consistem em conjuntos de sensores e módulos de comunicação que mantêm registos acessíveis em tempo real das leituras dos sensores e, se aliados aos sistemas de automação e controlo, podem permitir a interação para a resolução de situações, podendo até ter poder de decisão.

Para aplicações de gestão de frotas, a utilização da localização geográfica em tempo real é um fator fulcral, que permite monitorizar a posição dos veículos em tempo real e constitui uma ferramenta importante para a gestão dos recursos. Para além disso, permite também um maior controlo das viaturas e funcionários, a otimização das rotas, a poupança de combustível, o que gerará um acréscimo da produtividade e consequentemente uma aumento na qualidade de serviço prestado. No caso específico de frotas de distribuição de bens sensíveis à temperatura, como acontece com a comida fresca, a junção dos fatores descritos pode não só melhorar o desempenho e eficiência dos serviços, como também evitar situações de deterioração dos produtos alimentares. Monitorizando as condições climatéricas da carga e do exterior e a localização em tempo real da viatura, podem-se otimizar as rotas para que a degradação dos produtos frescos seja a menor possível, permitindo agir imediatamente no caso de uma mudança de temperatura dos alimentos (provocada, por exemplo, por tempo excessivo no tráfego, avarias no sistema de arrefecimento das câmaras da viatura, ou outra situação imprevisível).

1.1 OBJETIVOS E CARACTERÍSTICAS DO PROJETO

Este projeto pretende desenvolver um módulo de aquisição de dados e comunicação, capaz de adquirir, tratar, guardar e comunicar um conjunto de dados críticos, inerentes tanto às viaturas de uma frota de distribuição de comida fresca e congelada, como ao seu meio envolvente e à sua carga. Sendo esta uma aplicação empresarial, os fatores que a seguir se descrevem refletem a necessidade de ir ao encontro das especificidades da aplicação. Adicionalmente, e para tornar o módulo completamente autónomo e comercialmente mais apelativo, também foram implementados: a aquisição de coordenadas GPS, uma interface para obter dados da viatura e calcular os consumos, e uma comunicação que permite ligar o dispositivo à internet, possibilitando a monitorização remota em tempo real.

O módulo de aquisição de dados estará dotado de comunicação móvel GSM (*Global System for Mobile*) e GPRS (*General Packet Radio Services*), de modo a estar habilitado a comunicar as leituras, em tempo real, à central (através de uma ligação à internet). A componente de aquisição de dados relativos à viatura será executada através da interface OBDII, que permite a aquisição de dados como velocidade e consumo instantâneos, tempo de condução de acordo com a ignição e desligar do motor, entre outros. Também existe a possibilidade de monitorizar dados relativos à mecânica da viatura (com o objetivo de prevenir avarias), no entanto esta opção não está no âmbito deste projeto.

A componente de rede de sensores contará com sensores de temperatura no interior das câmaras refrigeradas, com possibilidade de instalação de outros tipos de sensores, como sensores de abertura de portas ou de humidade. Dado que as câmaras refrigeradas são herméticas, os sensores instalados no seu interior utilizarão comunicação sem fios por Rádio Frequência (RF), utilizando o protocolo *ZigBee* para comunicar as leituras ao módulo central. Visto que a temperatura média da câmara de congelamento ronda os −20ºC, estes dispositivos devem estar preparados para trabalhar a temperaturas negativas nesta gama de valores, o que se refletirá na seleção dos componentes a utilizar. Outro fator a ter em conta relativamente a estes módulos é o seu invólucro ou caixa, uma vez que as câmaras onde estão instalados estão sujeitas, diariamente, a condições extremas ambientais com condensação, humidade e até exposição a produtos químicos (desinfetantes para lavagens

das câmaras). Por outro lado, tratando-se de módulos sem fios, serão obrigatoriamente alimentados a baterias, e o tipo de baterias a utilizar e a sua vida útil será também um fator a considerar, uma vez que necessitam operar a baixas temperaturas.

Uma outra componente do módulo de aquisição é a possibilidade de adquirir constantemente a posição geográfica da viatura através do sistema GPS, permitindo o total conhecimento sobre as rotas realizadas pelas viaturas.

O objetivo final do desenvolvimento e instalação destes módulos é a comunicação de todas as informações atrás referidas a um sistema de otimização de rotas, visto que todos os fatores citados poderão vir a influenciar o cálculo da melhor rota de distribuição.

1.2 ENQUADRAMENTO DO TRABALHO

O módulo de aquisição de dados desenvolvido nesta tese de mestrado está enquadrado no âmbito do projeto i3FR - QREN n. 34130, que envolve o consórcio entre duas empresas do Algarve, a X4DEV e a Aviludo, e o Instituto Superior de Engenharia da Universidade do Algarve, com o objetivo de desenvolver um sistema integrado de otimização e distribuição.

Assim, a outra componente do referido projeto, o módulo de otimização de rota, foi desenvolvido em simultâneo com este trabalho, tendo em conta as múltiplas restrições e objetivos. O cálculo das rotas será dotado de inteligência baseada nos dados adquiridos por rotas prévias, deixando abertura para integrar outras fontes. Em consequência, o desenvolvimento deste módulo de aquisição de dados serve então o propósito de fornecer à componente de otimização de rotas com os dados referidos (i.e. GPS, temperaturas, consumos) em tempo real.

1.3 ORGANIZAÇÃO DO RELATÓRIO

O presente documento está organizado como a seguir se descreve.

O capítulo atual introduz o tema, bem como as motivações e objetivos que levaram ao desenvolvimento do projeto.

No capítulo 2, são tratados os conceitos fundamentais sobre monitorização remota de temperaturas e localização remota de viaturas, em duas subsecções, nas quais também são ilustradas algumas soluções residenciais e empresariais destes sistemas.

O capítulo 3 apresenta as tecnologias utilizadas no desenvolvimento e implementação de todos os subsistemas inerentes a este trabalho. São detalhados os aspetos relativos à plataforma para processamento, mas também os sensores de temperatura analisados, as tecnologias e protocolos de comunicação utilizados e a interface CAN bus para a comunicação com viaturas.

No capítulo 4 serão descritos os módulos principais e remotos. Começando pelos módulos remotos e seguindo para o principal, serão descritas detalhadamente todas as funções que estes módulos deverão desempenhar tendo em conta todos os requisitos da aplicação no ambiente empresarial em questão.

O capítulo 5 descreve a implementação do sistema, nomeadamente as implementações em *hardware* e *firmware*. São descritas todas as interligações entre os componentes dos módulos e periféricos, ilustrados e explicados os circuitos e suas configurações, e é detalhada a programação do microcontrolador.

O capítulo 6 está reservado para a demonstração e análise sobre o funcionamento dos módulos.

Por fim, no capítulo 7 são apresentadas as conclusões gerais sobre o trabalho de desenvolvimento dos módulos, bem como das tecnologias. É feito um resumo do trabalho realizado e uma análise das funcionalidades implementadas, analisando o funcionamento e os fatores positivos e negativos das tecnologias utilizadas/implementadas. No final do capítulo, serão apresentados algumas direções de trabalho futuro e possíveis alterações e atualizações à implementação desenvolvida.

2. MONITORIZAÇÃO REMOTA DE PARÂMETROS E LOCALIZAÇÃO DE VIATURAS

Sendo a monitorização remota de parâmetros e o rastreamento de viaturas duas áreas em franca expansão e desenvolvimento, têm surgido no mercado diversas soluções utilizando estes conceitos. Na área da monitorização, uma maior parcela de soluções existentes focam-se na temperatura e humidade de ambientes residenciais ou industriais, controlados ou não. Na área de localização de viaturas, existem várias soluções no mercado, com ou sem navegação na cabine, normalmente dotados de comunicação em tempo real através de operadores móveis.

A monitorização de dados dotada de conexão Ethernet[1]/Internet traz novas possibilidades e facilidades de acesso até então impossíveis. Com simples sensores analógicos pode-se monitorizar, controlar ou registar dados em praticamente qualquer local, no próprio prédio ou do outro lado do mundo.

Não existem no mercado soluções integradas, ou seja, aliar as leituras dos sensores, as leituras obtidas da Centralina e as posições GPS (*Global Positioning System*), criando assim um conjunto mais completo de dados agrupados e controlados por um só *hardware* central. Essa é a grande mais-valia do projeto aqui desenvolvido.

Neste capítulo sumarizam-se as soluções existentes para monitorização remota de dados, nomeadamente a monitorização de temperaturas e a localização remota de veículos.

2.1 MONITORIZAÇÃO DE TEMPERATURAS

Uma das aplicações de monitorização remota mais largamente utilizada é a monitorização da temperatura. Esta apresenta um amplo leque de aplicações residenciais e empresariais. Em ambiente residencial, pretende-se monitorizar a temperatura ambiente de modo a controlar os sistemas de climatização. Em ambientes industriais, podem ser

desde monitores de câmaras frigoríficas a fornalhas ou estufas, visando salvaguardar a qualidade dos produtos e evitando a exposição destes a níveis de temperatura não desejáveis.

É de referir que as soluções comerciais aqui apresentadas são exemplos comercializados em Portugal como soluções finais. Porém, existem várias empresas de Eletricidade e de Domótica que desenvolvem (ou podem desenvolver) soluções específicas adequadas ao cliente, tanto para clientes residenciais ou clientes empresariais. Não sendo soluções finais já desenvolvidas, são, portanto, muito mais dispendiosas.

2.1.1 SOLUÇÕES RESIDENCIAIS

Diferentes empresas vêm apresentando, cada vez mais, soluções residenciais para a monitorização de temperaturas e humidade, bem como sensores de deteção de chama, fumo, dióxido de carbono e outros. Estas aplicações visam o conforto e a segurança das pessoas que habitam os ambientes monitorizados. As soluções comerciais típicas utilizam funcionalidades de termostato, ou seja, detetam a temperatura e, a partir de valores introduzidos pelo utilizador, controlam o sistema AVAC (Aquecimento, Ventilação e Ar Condicionado) da residência, visando manter a temperatura ambiente na gama de valores confortáveis aos residentes.

O Nest Thermostat (Figura 1) da Nest Labs [2] é um exemplo de uma solução comercial para residências. Consiste num termostato "inteligente" cujo objetivo é substituir os termostatos convencionais de controlo de AVAC já existentes na residência. A instalação do Nest Thermostat é feita substituindo o termostato existente, ligando o novo aos conectores respetivos do sistema de climatização. Estes módulos são dotados de sensores de temperatura, humidade e luminosidade, e são dotados de tecnologia Wi-Fi [3]. Assim, uma vez configurados (aquando da instalação), ligam-se à rede sem fios da casa e, por sua vez, a um servidor. Mediante autenticação no servidor, o utilizador, através da internet num computador, tablet ou smartphone, consegue aceder remotamente aos termostatos, podendo consultar a temperatura, humidade e luminosidade em cada divisão onde um termostato esteja instalado.

Figura 1- *Nest thermostat*

Complementarmente, o utilizador cria uma conta no servidor da empresa *Nest*, onde ficam registados os dados históricos dos valores de temperatura, bem como de comandos que o utilizador tenha enviado ou modificado localmente nos termostatos. A análise destes dados permite ao sistema analisar os padrões de utilização na residência, prevendo que temperaturas utilizar nas divisões e em que alturas do dia. Fica deste modo explicada a nomenclatura "inteligente" que este módulo recebe pelo seu fabricante, que visa o aumento da eficiência energética da residência. Este produto é atualmente o único termostato "inteligente" presente no mercado português para clientes residenciais.

2.1.2 SOLUÇÕES EMPRESARIAIS

Os produtos para ambientes empresariais ou industriais requerem uma robustez, fiabilidade, durabilidade e capacidade superiores aos produtos residenciais. Para este tipo de utilização, pretende-se não apenas monitorizações ambientais que visam a componente humana, mas também as restrições ambientais a que certos produtos podem estar expostos, ou até mesmo as temperaturas máximas e mínimas a que certas máquinas devem trabalhar.

Um exemplo de uma solução empresarial é a solução *iServer* [4] comercializada pela empresa *Omega*. Esta consiste num servidor que faz o tratamento e disponibilização dos dados através de *Ethernet/Internet*, possibilitando assim o acesso aos mesmos em qualquer plataforma também com condições de acesso padrão à internet (e.g., Ethernet e um navegador de Internet). O *hardware* de aquisição deste sistema apresenta-se de duas formas, como a seguir se descreve. A primeira é por meio dos módulos da série *EIT*, que permitem acrescentar conectividade *Ethernet* / TCP/IP[1] a qualquer *hardware* com conectividade Série (RS232/RS495[2]) ou Modbus[3]. Ou seja, funcionam como ponte para essas tecnologias, permitindo adaptar equipamentos de qualquer fabricante que possua as mesmas, para poder ligar-se à rede TCP/IP. Se não for possível fazer-se chegar cabos aos mesmos, podem ser conectados a um ponto de acesso sem fios, usufruindo assim de conectividade Wi-Fi.

A Omega também possui uma variada gama de módulos de *hardware* na sua solução de monitorização remota, que consistem em módulos autónomos dotados de tecnologia *Ethernet*. Se a rede possuir acesso a internet, também os módulos terão. A série *iSD-TH* apresenta módulos de monitorização ambiental de segurança para sistemas críticos de AVAC, como salas de servidores, laboratórios, museus, armazéns ou qualquer outra instalação remota. Esta também é uma solução dotada de conectividade *Ethernet* para se comunicar com o sistema remoto, podendo enviar mensagens de correio eletrónico com alarmes críticos e ser consultada remotamente. Possui ligação USB (*Universal Serial Bus*), permitindo assim a conexão a uma câmara externa, de forma a se visualizar no navegador de *internet* o local de instalação do alarme, em direto. Dispõe também de conectividade RS232 e RS485.

[1] *Transfer Control Protocol* e *Internet Protocol*, são um conjunto de protocolos para a comunicação em rede.

[2] Protocolos padrão utilizados para a troca de dados em série.

[3] Protocolo de comunicação utilizado na automação industrial.

2.2 LOCALIZAÇÃO REMOTA DE VEÍCULOS

A gestão de grandes frotas é um problema complexo, que ao não ser levado em conta pode conduzir a grandes prejuízos para as empresas. Deve-se considerar fatores como consumos de veículos, condições mecânicas, tempos de condução/inatividade e rotas percorridas. Tendo acesso a todos estes dados, pode-se otimizar as soluções de frotas de modo a aumentar a sua eficiência e produtividade.

Em 2013 o mercado de localização remota de viaturas e telemática[4] estava avaliado em 7,5 mil milhões de euros e espera-se que até 2018 cresça até aos 23,5 mil milhões de euros[5].

Sendo empresas líderes no desenvolvimento de sistemas GPS pessoais e profissionais, a TomTom, apresenta a solução WEBFLEET [6] e a Garmin a solução Garmin Fleet 590. Ambas são soluções de tracking remoto através de posicionamento GPS e comunicação baseada em redes móveis. As soluções de ambas estão de acordo com a tendência atual de portabilidade, ou seja, consiste numa interface web que dá acesso à informação e ao controlo do sistema de forma remota e transversal a qualquer dispositivo com web *browser* compatível e ligação à internet, ou através de aplicação móvel.

Dada a dependência das comunicações para a implementação deste tipo de sistemas, uma outra empresa com influência no setor é a *Verizon Networkfleet*, uma filial da empresa de comunicações móveis americana *Verizon.*

2.2.1 TOMTOM WEBFLEET

Esta solução é compreendida por duas partes, a aplicação *WEBFLEET* e o *hardware TomTom LINK*. O modelo mais completo, o *LINK 510*, consiste num módulo instalado no interior da viatura com antenas GPS e GSM exteriores. Este módulo faz leituras sucessivas da posição da viatura e envia-as, periodicamente, ao sistema de gestão *WEBFLEET*. O

[4] Conjugação de serviços de telecomunicações e de informática, de modo a adquirir, tratar e enviar dados normalmente ligados a aplicações rodoviárias.

módulo também conta com um conector OBD-II. Desta maneira pode adquirir e enviar dados referentes ao veículo, como consumos, tempos desde a ignição, temperatura, pressão no sistema de combustível e outros. Estes dados permitem calcular e prever os consumos. O módulo também possibilita a ligação ao tacómetro da viatura, de modo a monitorizar os tempos de condução e velocidades previstas nas legislações aplicáveis. Este módulo também permite a instalação opcional de um sistema de navegação.

2.2.2 GARMIN FLEET

A empresa *Garmin* apresenta uma solução diferente da citada anteriormente. Dentro da sua gama de sistemas de navegação, apresentam uma série de dispositivos compatíveis com o seu sistema de *tracking* (Figura 2). É necessária a aquisição de um cabo proprietário da marca, para ligação a um sistema exterior, que fará a aquisição dos dados e envio através de redes móveis. Este dispositivo é fornecido por parceiros da marca que desenvolvem estes dispositivos, bem como a aplicação de interpretação de dados.

A marca disponibiliza ainda, livremente, a sua *API* para o público geral que tencione desenvolver *software* ou *hardware* compatível com os seus sistemas.

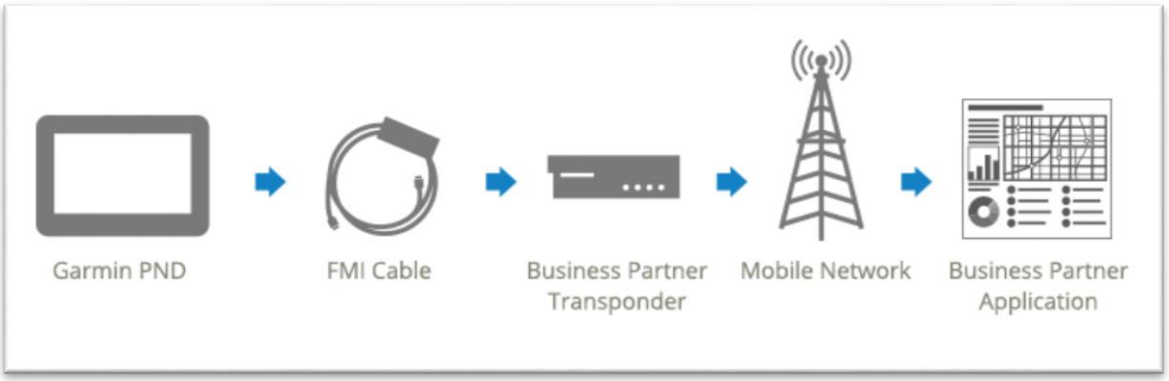


Figura 2 - Localização Remota de Veículo Garmin

2.2.3 VERIZON NETWORKFLEET

As soluções disponibilizadas pela Verizon apresentam um sistema completo de aquisição de dados de GPS e diagnóstico mecânico do veículo, disponibilizando estes dados permanentemente através das redes de comunicação móveis, sendo depois disponibilizadas e processadas por uma aplicação baseada em web (Figura 3).

Figura 3 - Verizon Networkfleet

Sendo a *Verizon* uma empresa originariamente de telecomunicações, o foco desta solução passa por oferecerem uma ligação segura sobre as suas próprias infraestruturas de redes de comunicação móveis. Deste modo, fornecem o serviço de servidor, onde os dados enviados pela viatura serão armazenados e tratados, ficando disponíveis através da *interface online* também fornecida pelos mesmos.

O módulo de *hardware* que fornecem, o *Networkfleet 5000* (Figura 4), adquire a localização geográfica periódica da viatura, bem como dados da mecânicos, como o consumo de combustível e alarme em tempo real em caso de aparecimento de avaria.

Esta solução não conta com monitorização sensorial da carga ou ambiente interior da cabina da viatura.

Figura 4 - Networkfleet 5000

3. TECNOLOGIAS UTILIZADAS

O presente capítulo apresenta e descreve todas as tecnologias a serem utilizadas no desenvolvimento e implementação de todos os subsistemas inerentes a este trabalho. Começa-se pela necessidade de processamento, sendo descrito o módulo de processamento a utilizar. De seguida serão descritos os sensores de temperatura e, na subsecção seguinte, todas as tecnologias e protocolos de comunicação utilizadas, ou seja entre os módulos locais e do módulo principal à central. São elas *ZigBee*, *Wi-Fi* e GSM/GPRS. De seguida serão também introduzidos e detalhados a interface CAN bus para a comunicação com a viatura e, por fim, a tecnologia de armazenamento de dados *Secure Digital*.

3.1 UNIDADE DE PROCESSAMENTO E PLATAFORMA MBED

Tratando-se do desenvolvimento de um sistema embebido que deverá executar tarefas complexas, como controlar dispositivos de comunicação, aquisição de dados, tratamento dos mesmos, efetuar cópias de segurança e formatá-los para o posterior envio, este deve ser desenvolvido utilizando uma unidade de processamento. Para esta tarefa utilizou-se um microcontrolador (MCU – *Microcontroller Unit*).

Tendo em conta os diversos periféricos envolvidos e a relativa complexidade da aplicação, e de forma a acelerar a prototipagem, foi selecionado para unidade de processamento a plataforma *mbed* [7]. Trata-se de uma plataforma de desenvolvimento *online*, baseada em microcontroladores *ARM*. A plataforma inclui um SDK (*Software Development Kit*) C/C++ padrão, com bibliotecas gratuitas, e um HDK (*Hardware Development Kit*) suportado por placas compatíveis.

O módulo utilizado neste projeto é o *mbed NXP LPC1768* (Figura 5), baseado no microcontrolador LPC1768 da NXP, por sua vez baseado no processador ARM Cortex-M3 [8], que é um processador de 96MHz com um poder de processamento já considerável e

que permite a sua utilização em aplicações complexas. A plataforma *mbed* NXP LPC1768 conta com uma série de periféricos e interfaces pré-instaladas/configuradas, nomeadamente:

- Ethernet
- USB
- RS232
- SPI (*Serial Peripheral Interface*) – interface utilizada para comunicar com dispositivos com funcionalidade SPI, como memórias *Flash* e LCD (*Liquid Crystal Display*).
- I2C (*Inter-Integrated Circuit*) – interface utilizada para comunicar com dispositivos com funcionalidade I2C, como memórias, sensores, ou outros módulos integrados.
- CAN (*Controller-Area Network*) – é um barramento padrão utilizado para microcontroladores transferirem informações entre si sem auxílio de um computador intermediário.
- Entradas analógicas – leem uma tensão analógica aplicada a um certo pino, como o nível de uma bateria ou a leitura de um sensor analógico.
- Saídas analógicas – servem o propósito de aplicar uma tensão específica a uma saída.
- PWM (*Pulse Width Modulation*) – aplica a uma saída uma onda PWM, ou MLP (modulação por largura de pulso).

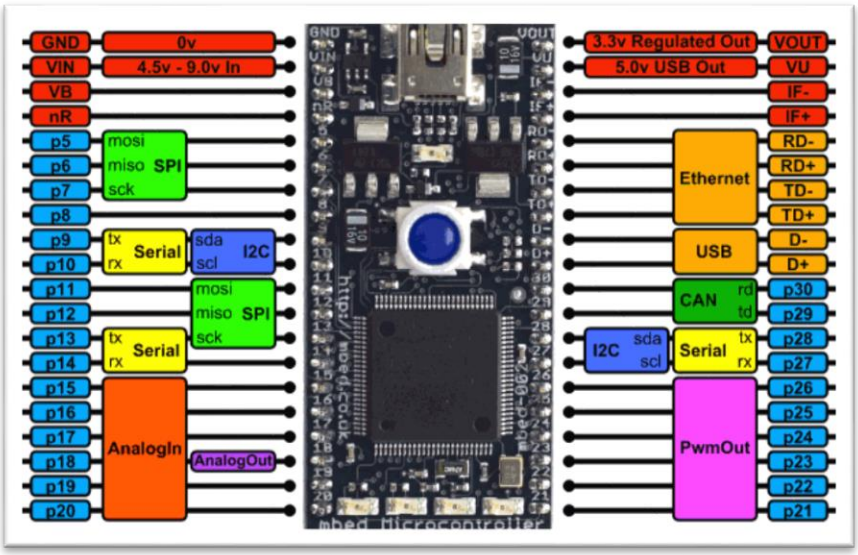


Figura 5 - Plataforma mbed NXP LPC1768 – Pinagem e Periféricos

3.2 SENSORES DE TEMPERATURA

Sendo a finalidade deste projeto a de desenvolver um módulo de aquisição de dados para viaturas de uma frota de distribuição de comida fresca e congelada, a medição da temperatura é um ponto de extrema importância. Estes módulos devem estar aptos a funcionar em temperaturas negativas até aos -20ºC. Esta secção estuda os diversos tipos de sensores existentes, de modo a selecionar o mais indicado para esta aplicação.

3.2.1 TERMOPAR

O princípio de funcionamento deste tipo de sensor de temperatura deve-se ao fenómeno observado por Thomas Seebeck, em 1821, que ao se juntarem dois condutores de metais diferentes em ambas as extremidades e aquecer uma destas extremidades, flui constantemente uma corrente pelo circuito termoelétrico [9] (Figura 6).

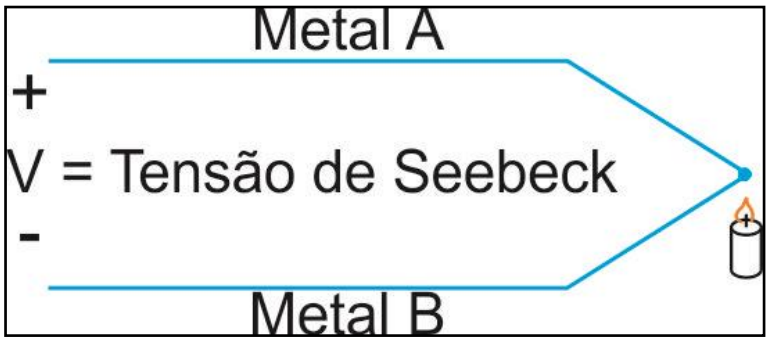


Figura 6 – Termopar

O termopar é um dos tipos de sensores de temperatura mais utilizados. De entre as vantagens na utilização deste tipo de sensor destaca-se a sua robustez, sendo capaz de operar em uma vasta gama de temperaturas e diversos ambientes. Também são vantagens deste sensor a simplicidade de utilização, a variedade de formas de apresentação disponíveis no mercado, o seu baixo custo e o fato de não necessitar de alimentação externa.

Porém, a relação tensão/temperatura no termopar não é linear e a tensão varia muito pouco com a temperatura. Para compensar estes factos negativos, podem (e devem) ser utilizados circuitos eletrónicos especiais para amplificar a tensão obtida dos sensores e também para compensar a sua não linearidade com a temperatura (de forma a tornar as leituras mais precisas).

Em suma, pode-se afirmar que o termopar é um tipo de sensor de temperatura que prima pela simplicidade, versatilidade e baixo custo, não sendo porém o mais indicado para aplicações onde a precisão é uma necessidade.

3.2.2 TERMÍSTOR

O termístor (***therm***o-sensitive res***istor***), como o nome sugere, trata-se de uma resistência sensível à temperatura. Este tipo de sensor de temperatura é, de facto, o mais sensível, apresentando uma grande variação do parâmetro de medição com a variação de temperatura. A maior parte dos termístores apresentam um coeficiente de temperatura negativo (NTC), ou seja, sua resistência decresce com o aumento da temperatura, apesar de existirem também versões com coeficiente positivo. O coeficiente negativo pode apresentar-se em percentagens por °C de tal ordem que pode-se verificar mudanças de temperatura ao minuto, característica não observada no termopar ou no RTD (*Resistive Temperature Device*) [10] (que irá ser apresentado na secção 3.2.3). Deste ganho na sensibilidade, advém uma extrema não linearidade. Desta forma, os fabricantes não padronizaram as curvas típicas dos termístor, como acontece com o termopar, por exemplo.

A grande resistividade do termístor traz uma vantagem na medição, que é a de não ser necessária a aplicação da medição de resistência com quatro fios (*four-wire*), produzindo um erro significativamente menor que o RTD. Por serem constituídos de materiais semicondutores, estes sensores apresentam a desvantagem de serem suscetíveis a erros de calibração permanentes por exposição prolongada a altas temperaturas, o que limita a gama de valores de aplicação destes dispositivos. Também são significativamente menos

robustos que os termopares e os RTD, sendo necessária especial precaução na sua instalação (por exemplo, para evitar esmagamento).

3.2.3 RTD (RESISTIVE TEMPERATURE DEVICE)

Sir Humphrey Davy anunciou, no mesmo ano em que Seebeck descobriu a termoeletricidade, que a resistividade dos metais apresentava alterações conforme a variação de temperatura [11].

O princípio de funcionamento do RTD (ou dispositivo de resistência térmica) baseia-se no fato da resistência elétrica dos metais apresentar dependência em relação à temperatura. Estes dispositivos são geralmente compostos por platina, que se revela especialmente adequada a esta aplicação, por suportar altas temperaturas e mantendo a sua estabilidade.

Várias técnicas foram utilizadas ao longo dos anos na construção destes dispositivos. As mais comuns utilizam filamentos, embora a mais moderna consista numa fina camada de platina sobre um pequeno substrato de cerâmica. Esta última apresenta algumas vantagens, como por exemplo a maior sensibilidade e velocidade na medição, devido ao tamanho reduzido. Porém, apresenta menos estabilidade que as antecessoras.

Os valores comuns da resistência de um RTD variam desde os 10 ohms (Ω), nos primeiros modelos, até aos milhares de ohms, nos modelos modernos em filme de metal. Quanto à medição da resistência, e tratando-se de valores pequenos de variação por °C, um problema associado é o de os próprios condutores ligados à resistência apresentarem, também eles próprios, resistência. Este fato pode causar erros grandes na medição de temperatura.

3.2.4 SENSOR EM CIRCUITO INTEGRADO

Os sensores em Circuitos Integrados (CI) apresentam-se com saída em tensão ou corrente, podendo ser digitais ou analógicos. No caso dos digitais, apresentam a vantagem

de poderem ser diretamente lidos por um microcontrolador. Os analógicos consistem em uma tensão de saída que se altera de acordo com variação da temperatura, seguindo uma variação fornecida pelo fabricante. Estes sensores partilham as desvantagens do termístor, apresentando uma gama limitada de temperaturas. Porém, são os que apresentam maior linearidade e facilidade de ligação a um microcontrolador, o que facilita a sua aplicação.

3.2.5 COMPARAÇÃO ENTRE TIPOS DE SENSOR DE TEMPERATURA

Na Tabela 1 encontram-se sintetizadas as vantagens e desvantagens estudadas nas secções anteriores.

Tendo em conta as necessidades da aplicação, é necessário que os sensores funcionem a temperaturas negativas até -20ºC, não havendo a necessidade de grande velocidade de operação, visto que as temperaturas nas câmaras terão quedas ou subidas gradativas lentas, mesmo em caso de falha do sistema de refrigeração (visto as mesmas serem isoladas termicamente).

Em soluções empresariais, um fator importante é sempre o custo, pelo que se deve optar pela solução menos dispendiosa que satisfaça os parâmetros desejados. Assim, os sensores em circuito integrado apresentam as características desejadas, sendo sensores de baixo custo, fácil e rápida implementação.

Neste âmbito, foi selecionado o sensor *MCP9700* do fabricante *Microchip,* por apresentar as características mencionadas acima e suportar temperaturas negativas nos valores desejados. Trata-se de um sensor denominado de Termístor Ativo Linear em Circuito Integrado (*Linear Active Thermistor™ IC*) e apresenta uma tensão na saída diretamente proporcional à temperatura medida. O sensor MCP9700 pode com precisão medir temperaturas entre -40°C e +150°C. A sua saída está calibrada para uma relação com a temperatura de 10mV/°C e apresenta uma tensão constante (DC) de *offset* de 500mV. Esta tensão de *offset* permite medir valores negativos de temperatura sem a necessidade de uma tensão de alimentação negativa.

As suas características resumidas são:

- Saída com relação linear com a temperatura: 10mV/°C

- -4°C/+6°C de precisão entre -40°C e +150°C
- Muito baixa corrente de operação: 12µA (máx)
- Operação com tensões de alimentação entre 2.3V e 5.5V
- Sem necessidade de componentes externos

Tabela 1 - Sensores de Temperatura, Prós e Contras

	Termopar	Termistência	RTD	CI
Vantagens	Simples; Robusto; Baixo Custo; Larga Gama de Temperaturas; Grande Variedade de Formatos;	Rapidez; Medida Óhmica com Dois Condutores;	Grande Precisão; Maior Linearidade que o Termopar;	Maiores Valores de Saída Baixo Custo; Alta Linearidade; Fácil Implementação;
Desvantagens	Não Linear; Baixa Tensão; Requer Medição de Referência; Baixa Estabilidade; Baixa Sensibilidade;	Não Linearidade; Limitada Gama de Medição; Fragilidade; Requer Fonte de Corrente; Auto Aquecimento;	Maior Custo; Lentidão; Requer Fonte de Corrente; Pequena Variação da Resistência; Medição com quatro condutores;	T <250ºC; Fonte de Alimentação Requerida; Lentidão; Auto Aquecimento;

3.3 COMUNICAÇÕES

3.3.1 ZIGBEE

ZigBee é um aperfeiçoamento ao protocolo IEEE 802.15.4, desenvolvido pela *ZigBee Aliance*, uma associação mundial sem fins lucrativos e cujo objetivo é desenvolver padrões

de redes sem fios sustentáveis de baixo consumo, vocacionadas principalmente para as redes de sensores. O padrão IEEE 802.15.4 tem como objetivo fornecer os parâmetros para as camadas inferiores (físicas) de uma rede de área pessoal sem fios (WPAN – *Wireless Personal Area Networks*) com enfoque no baixo consumo e na comunicação de baixa velocidade [12]. O padrão *ZigBee* complementa o padrão IEEE 802.15.4 com funções mais avançadas, como as topologias de rede que apresenta. Os módulos podem ser coordenadores, *routers* ou *end-devices* (terminais), organizados nas topologias de malha, árvore ou estrela (Figura 7). Para redes de sensores, a topologia de malha é especialmente interessante, pois permite adicionar e retirar módulos da rede e, uma vez que todos comunicam entre si, podem-se criar redes de grande porte e alcance.

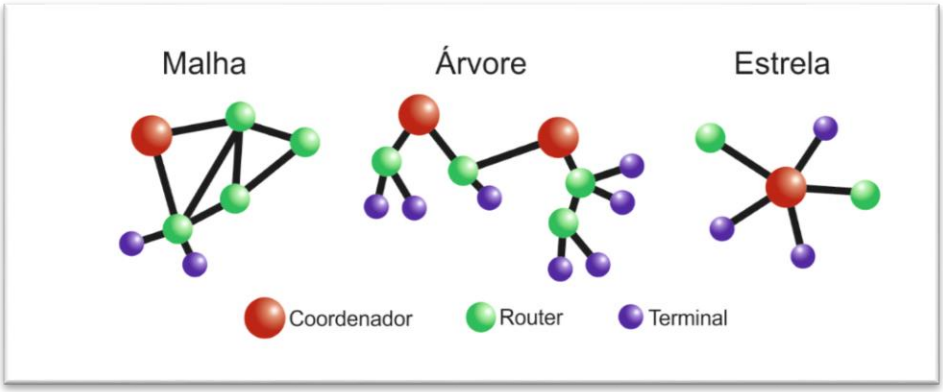


Figura 7 - Topologias ZigBee

Todas as redes devem possuir pelo menos um coordenador. Este é o responsável por receber, comunicar e organizar as comunicações entre todos os outros membros da rede. Os *routers* comunicam entre si, com o coordenador e com os *end-devices*. Os *end-devices*, como o nome indica, são os dispositivos de fim de linha, ou seja, não passam nunca a informação adiante, apenas enviando a sua própria informação à rede.

Das características do protocolo, destacam-se:

- Opera na banda de frequência global de 2.4GHz (de acordo com IEEE 802.15.4)
- Frequências regionais: Américas – 915MHz; Europa – 868MHz.
- Mais de 16 canais a 2.4GHz.
- Possui mecanismos de poupança de energia.
- Mecanismos de descoberta de novos módulos na rede.

- Várias opções de transmissão, incluindo *broadcast* (difusão)

De modo a criar abstração e devido à complexidade de implementação do protocolo e dos seus mecanismos, vários fabricantes apresentam soluções integradas que oferecem um certo grau de abstração. Não estando no âmbito deste projeto a implementação de todos os mecanismos inerentes à uma rede ZigBee, foram utilizados módulos XBee da fabricante Digi. Estes módulos possuem o protocolo ZigBee implementado e pode-se configurar através de uma aplicação própria ou através de comandos AT. Os módulos contam com ligação série, desta forma pode-se enviar os comandos automaticamente através do *firmware* programado para o efeito. As configurações dos módulos bem coo a sua aplicação neste projeto estão pormenorizadas na secção de implementação (capítulo 5).

3.3.2 COMUNICAÇÕES MÓVEIS - GSM/GPRS

Inicialmente, sob a alçada do *Conférence Européene des Postes et Telecomunication* (CEPT, a agência de padronização europeia) e intitulado *Groupe Special Mobile*, posteriormente *Global System for Mobile*, tendo o CEPT evoluído para o atual ETSI (*European Telecomunications Institute of Standardization*). Esta norma tinha o objetivo inicial de desenvolver e definir um padrão de telecomunicação digital de canais de voz (ao contrário do seu antecessor, que era puramente analógico) na banda dos 900MHz, atualmente existindo em outras bandas de frequência, sendo a mais significativa a de 1800MHz [13].

A segunda geração de sistemas de comunicações móveis, ou 2G, introduziu o sistema celular, ou seja, cada estação base cobre uma célula, estando ligadas entre si por instalações cabeadas ou microondas, permitindo a implementação de *handover*. O *handover* consiste na comunicação entre células, sendo que o aparelho deteta a sua passagem de uma célula para a outra, garantindo que a comunicação não seja interrompida. Tratando-se de um sistema digital, foi também implementado o serviço de SMS (*Short Message Service*), que permite enviar e receber mensagens escritas entre dispositivos GSM.

Mais tarde, surgiu a necessidade de algumas aplicações poderem enviar maiores quantidades de dados, para as quais seria mais vantajoso um sistema onde se faz um pedido para alocar os recursos de rede e, de seguida, libertá-los, quando a informação tivesse sido enviada. A informação teria de ser enviada em pacotes. A este método chama-se *packet switching*. Assim, é possível abrir uma ligação, um IP é atribuído, e os pacotes recebem um cabeçalho, de maneira a serem reconhecidos no destino. De modo a implementar este sistema e enviar pacotes sobre GSM foi criado o padrão GPRS [14]. Já não é necessária uma ligação lógica ponto a ponto entre os comunicadores (ver estrutura da rede na Figura 8).

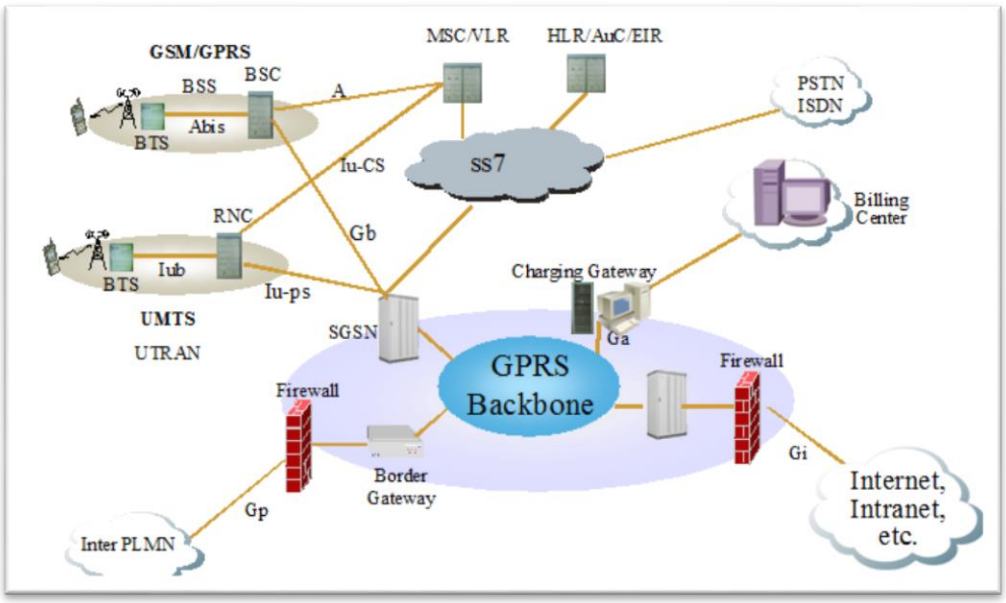


Figura 8 - Estrutura da rede GSM/GPRS

Este padrão ficou conhecido como 2.5G, pois apesar de ser uma evolução em relação ao GSM, é baseado no mesmo sistema físico e não num sistema independente. O GPRS está integrado no GSM.

Além da vasta gama de terminais com capacidades GSM e GPRS disponíveis no mercado de consumo, vários fabricantes disponibilizam módulos voltados à comunicação *M2M* (*machine-to-machine*), ou seja, para comunicação entre sistemas autónomos. No caso deste projeto, foi selecionado o módulo SIM900 do fabricante SIMCom.

Os módulos SIM900 possuem ligação por porta série e é comandado através de comandos *AT*. Desta forma, há abstração em relação à implementação da comunicação

com a rede GSM, com exceção de algumas configurações. Desta forma, o *firmware* enviará ao módulo as configurações de rede: utilizador, palavra-passe, nome do ponto de acesso móvel, possivelmente o PIN do cartão SIM inserido. Os comandos específicos necessários ao envio da informação gerada especificamente para esta aplicação estão detalhadas no capítulo 5.

3.4 LOCALIZAÇÃO GPS

Originalmente criado como um sistema militar do departamento de defesa dos Estados Unidos da América, oficialmente intitulado NAVSTAR[5], o GPS (sistema global de localização) é um sistema de navegação baseado numa constelação de satélites, que permite localizar qualquer ponto na Terra calculando a localização do recetor através de triangulação. Os satélites estão sincronizados, sendo monitorizados por estações terrestres, e emitem constantemente a sua posição exata. O mínimo para se estimar a posição em duas dimensões do recetor são três satélites, tratando-se de uma estimativa não precisa. Idealmente são necessários quatro ou mais satélites para ser calculada a posição em três dimensões, que é muito mais exata.

A precisão do sistema GPS pode chegar a 1cm ou menos, dependendo das características do próprio recetor e das técnicas de medição, sendo também afetada pela topografia envolvente e distribuição de construções e árvores, por exemplo[15].

Apesar da sua origem militar, rapidamente se percebeu a potencialidade do sistema para aplicações civis, cuja primeira grande aplicação foi a navegação marítima. Desde então, uma grande variedade de aplicações emergiram em aplicações científicas, comercias e de lazer, como o sistema de localização de frotas de camiões, sistemas de navegação pessoais automobilísticos, sistemas auxiliares de aterragem e descolagem de aeronaves, ou várias outras aplicações.

[5] Ao longo do tempo referido como Navigation Signal Timing and Ranging ou Navigation Satellite Timing and Ranging

Apesar de os EUA manterem utilização livre do seu sistema para todo o mundo, alguns países quiseram manter os seus próprios sistemas. A Rússia criou o sistema GLONASS[6], também livre para a utilização civil.

3.5 OBD - ON-BOARD DIAGNOSTICS

On-board Diagnostics ou "diagnóstico de bordo" é o nome que se dá ao sistema de auto diagnóstico e reportagem de erros existente na maioria dos automóveis modernos. Tem como finalidade proporcionar aos utilizadores ou reparadores o acesso a informações sobre os vários subsistemas da viatura. As primeiras versões de sistemas OBD eram simples sinais luminosos, que acendiam se houvesse falha no subsistema relativo à luz em questão.

3.5.1 INTERFACES PADRÃO

Diversas versões foram desenvolvidas ao longo dos anos, bem como legislações próprias sobre a obrigatoriedade da instalação de tais sistemas nos veículos, sendo estes enumerados em seguida.

3.5.1.1 ALDL

O ALDL (*Assembly Line Diagnostic Link*) foi um padrão proprietário criado pela General Motors e é considerado o precursor dos sistemas de diagnóstico automóvel. As primeiras versões utilizavam comunicações com ritmo binário na ordem dos 160 bit/s, tendo as versões seguintes passado a comunicação bidirecional a 8192 bit/s.

[6] *GLObalnaya NAvigatsionnaya Sputnikovaya Sistema*, do russo, Sistema de Navegação Global por Satélite

3.5.1.2 OBD-I

Tendo sido a primeira versão regulada e padronizada de *On-Board Diagnostic*, o OBD-I teve como objetivo inicial o controlo das emissões de gases poluentes pelos veículos ao longo da vida útil destes. Cada fabricante utilizava o seu próprio conector de diagnóstico (DLC - *Diagnostic Link Connector*), uma localização física própria, bem como definição dos códigos de avaria (DTC - *Diagnostic Trouble Codes*).

O cenário mais comum neste padrão seria que, ao ligar o conector, o ECU entraria em modo diagnóstico. Neste modo a luz de avaria do motor iria tornar-se intermitente, com padrões de tempo pré definidos que correspondiam aos dois dígitos dos códigos de avaria.

3.5.1.3 OBD-1.5

A designação OBD-1.5 não representa uma variante oficial dos padrões OBD. Refere-se a uma versão do sistema utilizado maioritariamente pela General Motors em alguns modelos entre 1994 e 1995. O motivo da designação é este padrão se tratar de um híbrido, ou seja, já possui um conjunto de códigos OBD-II.

3.5.1.4 OBD-II

O ODB-II é uma evolução do OBD-I e é sobre esta variante que assentam os sistemas de diagnóstico presentes na maioria dos veículos automóveis atualmente. A padronização possibilita a utilização de um único tipo de conector para todos os veículos compatíveis, especificando o conector padrão bem como a sua pinagem, os protocolos de comunicação disponíveis e o formato das mensagens. Também está contemplado neste padrão uma lista de possíveis parâmetros a serem monitorizados e a forma de codificar os dados relativos a estes.

O padrão também fornece uma extensiva lista de DTC's, que consistem em códigos de 4 dígitos precedidos por uma letra: B para corpo, C para o *chassis*, U para as comunicações e para o motor.

3.5.1.5 EODB

O EODB (*European On Board Diagnostics*) consiste na norma europeia do padrão OBD-II e aplica-se, obrigatoriamente, a todos os veículos ligeiros de passageiros a gasolina com primeira matrícula na União Europeia a partir de janeiro de 2001, e a partir de 2004 para os carros a gasóleo. A implementação técnica desta norma é a mesma de OBD-II, pelo que a nomenclatura EODB não é de utilização corrente.

3.5.2 PROTOCOLOS OBD-II

Existem vários protocolos disponíveis para a comunicação com o sistema OBD-II: SAE J1850 PWM, SAE J1850 SWM, SO 9141-2, ISO14230 KWP2000 e ISO 15765 CAN.

Normalmente pode-se identificar os protocolos presentes, verificando que pinos existem no conector J19652 (Figura 9), sendo que no caso da Europa, todos os carros possuem o protocolo CAN *Bus*, que será o utilizado neste projeto. O Protocolo CAN foi desenvolvido pela empresa *Bosch*, para a aplicação no controlo industrial, sendo um protocolo largamente utilizado também nessa área.

Figura 9 - Conector J1962

3.5.3 MODOS DE OPERAÇÃO

Existem dez modos de operação padrão definidos na norma OBD-I SAE J1979, descritos abaixo. Cada um destes modos tem um conjunto de PID's (*parameter ID's*) definidos na norma.

01 – Apresenta os dados atuais.

02 – Apresenta dados *freeze frame* (dados capturados no momento da ativação do último

DTC referente).

03 – Apresenta o DTC armazenado.

04 – Limpa DTC e valores armazenados.

05 – Resultado de testes e monitorização do sensor de oxigénio, exceto para CAN.

06 – Resultado de testes e monitorização do sensor de oxigénio apenas para CAN.

07 – Apresenta DTC's pendentes.

08 – Operações de controlo do sistema On-board.

09 – Requisição de informações do veículo.

0A – DTC's permanentes.

Os modos mais significativos para a monitorização do estado do veículo são os modos 01 e 02. Estes modos são quase idênticos, sendo que o modo 01 representa leituras em tempo real durante um ciclo de condução, enquanto o modo 02 apresenta dados "congelados", ou seja, dados que permanecem registados na memória do ECU relativos ao último ciclo de condução.

3.5.4 PID (PARAMETER IDENTIFIER)

A norma SAE J1979 define uma lista de dados de diagnóstico padrão, bem como os métodos para a requisição desses dados ao ECU. Os tipos de dados presentes nessa lista são identificados pelos PID's (*Parameter Identifiers*). Apesar de a norma definir os PID's

padrão, existem vários outros possíveis que variam consoante o fabricante. A comunicação/requisição de diagnóstico ocorre na seguinte ordem:

- Insere-se o código de diagnóstico (normalmente de forma manual ou escolhendo de uma lista na aplicação)
- A ferramenta OBD-II envia para o ECU o código inserido/selecionado (mais usualmente através de barramento CAN, ou sobre outro dos protocolos de comunicação mencionados anteriormente).
- O ECU do veículo verifica o PID e, se o reconhecer, identifica a que parte funcional este se refere e reporta o valor para o barramento de comunicação.
- A ferramenta ou a aplicação lê o valor reportado, interpretando-o e mostra ao utilizador.

A requisição é feita enviando ao barramento de comunicação o modo e o identificador do PID que se espera receber. Por exemplo, ao enviar "0100" significa que se está a requerer os dados relativos ao PID '00' do modo '01'. Este envia como resposta 4 bytes de dados, usualmente designados na literatura e nos sistemas por A, B, C e D.

3.5.4.1 Interpretação de Dados e Codificação dos PID

Na norma estão definidas tabelas para a interpretação dos dados referentes aos PID's. A codificação os dados é feita em notação binária, de acordo com a Tabela 2 - Codificação Binária PID

Tabela 2 - Codificação Binária PID

A								B								C								D							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Os valores A, B, C e D representam os *bytes* do código. Deste modo, o valor A5, por exemplo, representa o bit 5 do byte A. Enquanto alguns PID's podem ser interpretados com

a aplicação direta de uma fórmula que traduz o seu significado, alguns, porém, são mais complexos, necessitam de uma interpretação mais detalhada dos valores lidos.

Deve-se destacar o PID 00 do modo 01, pois este PID contém a informação de que PID's estão disponíveis no veículo em questão. Este retorna 4 *bytes* de dados. Cada bit destes *bytes* representa um PID do modo em questão. O valor lógico do bit indica a disponibilidade do PID com número igual à sua posição na *frame*, sendo disponível par ao valor lógico '1' e indisponível para o valor lógico '0'. Por exemplo, se o ECU devolvesse o valor DD863D1A (em hexadecimal) a disponibilidade de PID's seria de acordo com a Tabela 3.

Tabela 3 - Disponibilidade de PID

Byte	Bit	PID	Disponibilidade	Byte	Bit	PID	Disponibilidade
DD	1	01	Sim	3D	0	11	Não
	1	02	Sim		0	12	Não
	0	03	Não		1	13	Sim
	1	04	Sim		1	14	Sim
	1	05	Sim		1	15	Sim
	1	06	Sim		1	16	Sim
	0	07	Não		0	17	Não
	1	08	Sim		1	18	Sim
86	1	09	Sim	1A	0	19	Não
	0	0A	Não		0	1A	Não
	0	0B	Não		0	1B	Não
	0	0C	Não		1	1C	Sim
	0	0D	Não		1	1D	Sim
	1	0E	Sim		0	1E	Não
	1	0F	Sim		1	1F	Sim
	0	10	Não		0	20	Não

Enquanto alguns PID possuem codificação binária com significado específico, como o exemplo anterior, alguns são interpretados aplicando fórmulas aos valores numéricos extraídos da palavra binária. Na norma também estão definidas as unidades de medida de cada grandeza relacionada com o PID em questão. Alguns PID comuns, pertencentes ao modo 00, estão representados na Tabela 4, bem como suas respetivas fórmulas e unidades.

Tabela 4 - PID - Fórmulas e unidades

PID	Nº Bytes	Descrição	Fórmula	Unidade
04	1	Carga do motor	$A \times \frac{100}{255}$	%
05	1	Temperatura do refrigerante	$A - 40$	ºC
0C	2	Rotações por minuto do motor	$\frac{(A \times 256) + B}{4}$	rpm
0D	1	Velocidade da viatura	A	km/h
1F	2	Tempo desde a ignição	$(A \times 256) + B$	s
2F	1	Nível do combustível	$\frac{(A \times 100)}{255}$	%
0A	1	Pressão do combusível	$A \times 3$	kPa

3.6 ARMAZENAMENTO – SD

Secure Digital (SD) é uma memória não volátil, apresentada na forma de cartões (*SD Card, Mini SD Card ou MicroSD Card*) utlizados em dispositivos eletrónicos que apresentam necessidade de armazenar dados.

O padrão *Secure Digital* é um padrão que foi introduzido em 1999 e é mantido pela *SD Association*[7] (ADA), tratando-se de uma evolução do padrão MMC (*Multimedia Cards*).

A vantagem de se utilizar um cartão SD é a portabilidade. Por serem instalados num *slot*, pode-se retirar o cartão e utilizá-lo num leitor em outro equipamento, como computador,

[7] Associação fundada pela *Matsushita Electric Industrial Co.* (atual *Panasonic*), *SanDisk Corporation* e pela *Toshiba Corporation*, possuindo várias outras empresas associadas atualmente.

smartphone ou *tablet*, tendo-se acesso instantâneo aos dados gravados. Existem no mercado em vários tamanhos, tratando-se todos de tecnologia SD e compatíveis (Figura 10).

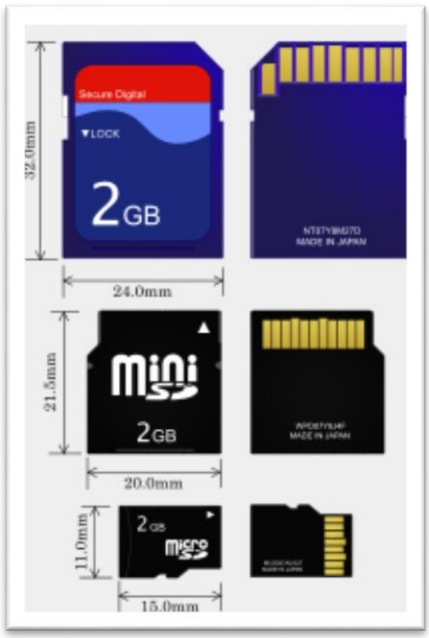
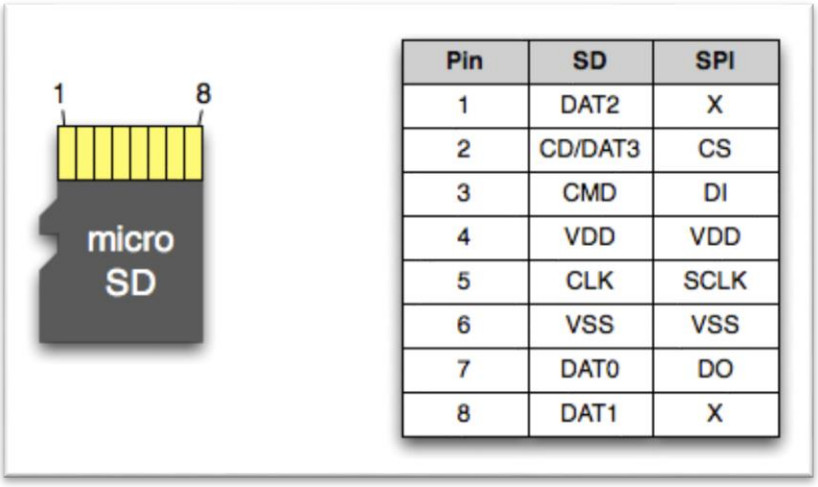


Figura 10 - Cartões SD

Neste projeto, estas memórias servirão de suporte aos *log files* e cópias de segurança dos dados em caso de falha de comunicações.

A memória SD é, essencialmente, uma memória *flash* [16] que comunica com o bloco de processamento como um periférico série, através do módulo *SPI* (*serial peripheral interface*), como representado na Figura 11. Desta forma, seria possível simplesmente ordenar a escrita de dados na mesma, em série, apenas tendo em conta o tamanho de cada bloco e o endereço de memória dos mesmos. Porém, estes dados só seriam possíveis de interpretar pelo próprio bloco de processamento. Assim, não faria sentido utilizar uma memória SD externa em detrimento de um bloco de memória *flash* integrado.

Porém, o emprego da memória SD nesta aplicação é possibilitar o *backup* dos dados e possibilitar a consulta destes dados em outro sistema que possua um leitor SD. Para tal, é necessário utilizar um sistema de ficheiros compatível de modo a formatar a memória e organizar e tornar legíveis os dados. O sistema de dados mais indicado para este fim será o sistema *FAT32* [17] [18], devido à sua larga compatibilidade e difusão atuais.



Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

Figura 11 - Ligação Cartão SD - SPI

O desenvolvimento integral do sistema de ficheiros está fora do âmbito deste projeto. Para a leitura e escrita dos dados nos cartão SD será utilizada a biblioteca *SD Filsystem* do *mbed* [19].

4. ESTRUTURA DO SISTEMA

Sendo este projeto desenvolvido para a aplicação real numa frota de distribuição de produtos, no âmbito do projeto i3FR, vários fatores levados em conta na definição e elaboração do projeto surgiram como requisitos impostos pela empresa parceira Aviludo. Assim, no final do projeto deve existir um produto completo e aplicável ao caso específico da empresa, colmatando as necessidades por ela apontadas (e que se irão descrever neste capítulo).

O sistema está dividido em três partes funcionais: Módulos Remotos, Módulo Principal e um servidor.

Os módulos remotos serão os módulos sem fios instalados nas câmaras frigoríficas da viatura. Estes módulos possuirão sensores de temperatura, no entanto estão preparados para a inserção de novos sensores posteriormente, como por exemplo, sensores de humidade. Os módulos remotos irão enviar periodicamente as leituras dos sensores ao módulo principal de processamento, instalado na cabine.

O módulo principal será o responsável pela recolha e tratamento de dados, vindos dos módulos remotos e módulo GPS, e por enviar os dados para o servidor.

O funcionamento do servidor está fora do âmbito deste trabalho e está sendo desenvolvido em conjunto no projeto i3FR.

A Figura 12 ilustra o funcionamento do sistema completo, bem como as ligações de dados e a comunicação efetuada entre os respetivos módulos. Em seguida será detalhado o funcionamento genérico de cada módulo, enquanto a implementação de cada um dos subsistemas será detalhada no Capítulo 1.

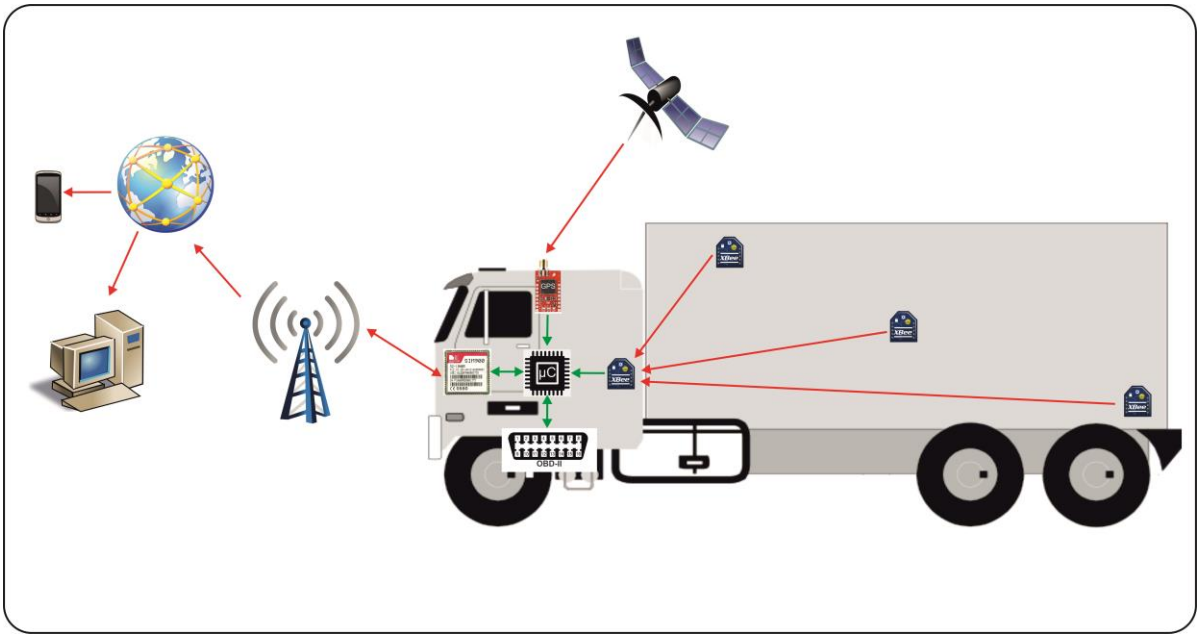


Figura 12 - Estrutura do Sistema

4.1 MÓDULOS REMOTOS

Alguns dos requisitos apontados recaem sobre a monitorização das câmaras refrigeradas.

Apesar de haver vários tipos de viaturas, as mais comuns apresentam uma câmara congeladora situada à frente, e uma câmara refrigerada situada atrás. Esta câmara refrigerada pode ter uma parede móvel, de modo a dividir essa câmara em duas câmaras refrigeradas (note que algumas viaturas não têm esta parede móvel, contando apenas com uma única câmara refrigerada). A câmara congeladora deve operar a -18ºC, podendo chegar aos -22ºC. As câmaras refrigeradas operam entre os 2ºC e os 5ºC. Assim, todo o *hardware* selecionado para os módulos remotos, uma vez que irão funcionar dentro destas câmaras, deve ser compatível com todas estas temperaturas.

Um requisito fundamental é que as câmaras não podem ser perfuradas, pois devem ser completamente herméticas (fator que é regulado pela autoridade competente, que fiscaliza e homologa as mesmas). Assim, surge a necessidade de se utilizarem sensores sem fios, pois para além de herméticas são construídas em fibra de vidro e material isolante não condutor, pelo que não impossibilitam a comunicação por RF (Rádio Frequência) do seu

interior para o exterior e vice-versa. Aliás, esta verificação foi efetuada no local no início do projeto, utilizando comunicação Bluetooth.

Assim, para a comunicação com os sensores são utilizados os módulos *XBee Series 2* (Figura 13) do fabricante *Digi International*. Estes módulos estão indicados para redes de sensores, utilizando o protocolo ZigBee, e possuem quatro ADC[8] (*Analog to Digital Converter*) onde se podem ligar diretamente sensores analógicos. Comunicam com uma taxa de 250 Kbps e apresentam um alcance *indoor* até 40m, com transmissor de 2mW e com consumo de 40mA a 3.3V. Como referido anteriormente, comunicam na banda dos 2.4GHz.

Figura 13 - Módulo Xbee Série 2

Outro fator fundamental na seleção destes módulos é que, tratando-se de módulos sem fios que serão necessariamente alimentados a baterias, apresentam consumos muito baixos, pois uma troca constante de baterias não seria viável. Para implementar estratégias de baixo consumo, estes módulos possuem um modo de *sleep* (hibernação), ou seja, permitem a definição de períodos cíclicos configuráveis em que o dispositivo "dorme", isto é, entra num estado de baixíssimo consumo, durante o qual está praticamente desligado,

[8] Conversor Analógico-Digital, converte a tensão analógica aplicada em um dos pinos munidos de ADC, em um valor digital, de acordo com as especificações de conversão do dispositivo.

ligando-se periodicamente para comunicar as leituras. Desta forma, a comunicação entre os módulos remotos e o módulo principal (instalado na cabine) será unilateral (ver Figura 14). Isto significa que os módulos remotos estarão em modo *sleep* e, ao acordarem, enviarão os seus dados e voltarão novamente a "dormir", sendo o tempo de funcionamento destes módulos em modo normal muito reduzido. Este processo é fundamental para aumentar substancialmente a autonomia das baterias.

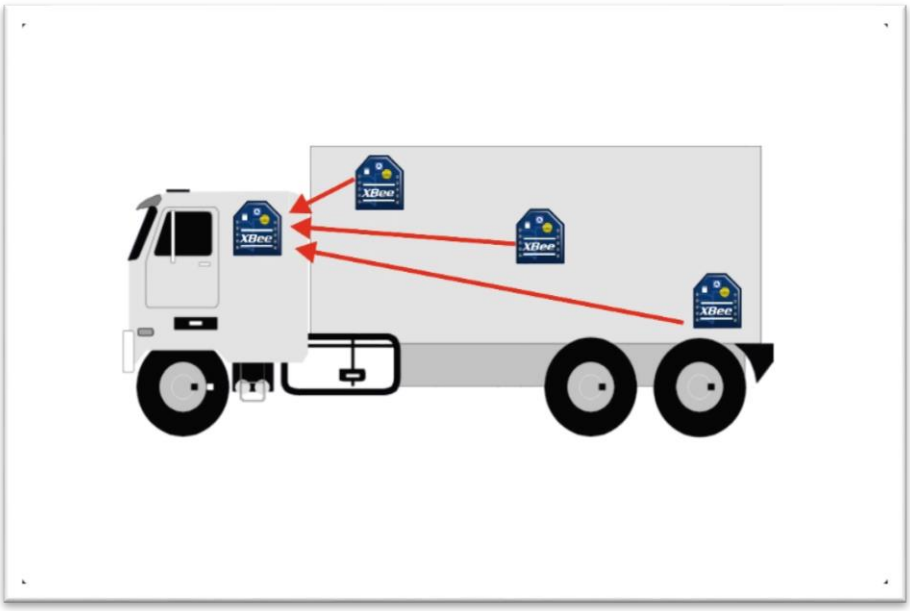


Figura 14 – Comunicação Módulos Remotos

Deste modo, a monitorização das câmaras será feita através de módulos compostos por comunicação RF sobre ZigBee, com sensores analógicos conectados aos respetivos ADC, efetuando envios periódicos de dados dos sensores e do estado das baterias, estando em modo *sleep* entre amostragens/envios. Os mesmos serão instalados em caixas herméticas, resistentes a químicos e humidade, de modo a poderem continuar instalados no interior das câmaras aquando da limpeza e higienização das mesmas (tarefa efectuada diariamente).

É de notar que o *hardware* utilizado nestes módulos remotos deve ser restringido ao mínimo necessário para a aquisição das temperaturas nas câmaras, mais uma vez pela necessidade de estender ao máximo a duração das baterias e a autonomia do módulo.

4.2 MÓDULO PRINCIPAL

Para coordenar todo o sistema de aquisição de dados será instalado um módulo principal na cabine da viatura, que será o módulo que agrega toda a informação recolhida e a envia para o servidor da empresa. Deste modo, o módulo principal tem como função receber, tratar, gravar coordenador de todo o processo de aquisição e enviar todos os dados necessários à aplicação.

Este módulo, por estar situado na cabina, terá como fonte de alimentação a bateria da viatura, não havendo portanto limitações ao seu modo de funcionamento, pois deverá estar sempre ligado sempre que a ignição da viatura também estiver.

De seguida, serão explicados os detalhes relativos a este módulo principal.

4.2.1 DADOS REMOTOS

Visto que o papel dos módulos remotos passa apenas por enviar amostragens dos diversos sensores, estes não dispõem de processamento local. Assim, o módulo principal não terá a capacidade de efetuar nenhum tipo de pedido aos módulos remotos, estando apenas "à escuta" e aguardando a disponibilidade dos dados no seu XBee local (Figura 14). Desta forma, o XBee do módulo principal estará sempre ligado, uma vez que está alimentado pela bateria da viatura, apenas se desligando quando esta se desligar. Este XBee do módulo principal comunica com o processador por RS-232.

Assim, o módulo principal contará com um *Xbee* em modo *Coordinator*, ligado ao módulo de processamento (*mbed*). A topologia a utilizar na rede ZigBee será a de estrela, sendo o coordenador ligado ao módulo principal e os módulos remotos configurados como *end-device*. Deste modo estarão ciclicamente a entrar em modo *sleep*, "acordando" apenas alguns milissegundos para enviar os dados. Para a aplicação, a empresa requer leituras ao minuto, de modo a que os módulos remotos estarão em modo *sleep* durante 59,98 segundos, acordando durante 20 milissegundos para enviar os dados e voltando a entrar em modo *sleep* em seguida.

4.2.2 ENDEREÇAMENTO

Uma questão associada tanto ao módulo principal quanto aos remotos é a necessidade de terem que ser identificados. Uma vez que serão instalados em viaturas e o objetivo é obter a centralização remota dos dados, deve existir um sistema de endereçamento de modo a identificar a origem dos dados. A rede de sensores também deve estar endereçada, de modo a que os módulos possam comunicar entre si, mas não com os módulos fora da sua rede, neste caso, com os módulos de outra viatura.

Outra questão relacionada com o endereçamento são as possíveis avarias, quando o sistema estiver instalado. Será também criado um mecanismo que facilite a programação automática de novos módulos a acrescentar ou substituir no sistema. Este mecanismo será detalhado na secção de implementação (capítulo **Erro! A origem da referência não foi encontrada.**).

Cada módulo presente na cabine das viaturas terá um endereço único relacionado com a referência daquela viatura. Esse endereçamento é importante por dois motivos principais descritos de seguida. Ao serem enviados, os dados transportarão a informação de que viatura fazem parte, possibilitando assim a distinção ente viaturas. O outro fator é a configuração de rede dos módulos Zigbee, de modo que os módulos remotos de uma viatura não comuniquem com módulos principais de outras viaturas próximas. Deste modo, cada viatura possuirá uma *PAN ID* único. Quando houver avarias, o módulo substituído deverá ser programado com a respetiva *PAN ID* da viatura em questão, bem como os seus módulos remotos. Deste modo cada rede de sensores só comunicará entre si dentro de uma mesma viatura, impossibilitando colisões de dados.

4.2.3 DADOS LOCAIS – GPS

Ao módulo principal está também ligado um recetor GPS (Figura 15), de modo a adquirir as posições GPS. Este receptor comunica por RS-232 com o *mbed*, enviando periodicamente uma *frame* com vários dados lidos do satélite de localização. Desta *frame*

são extraídos os dados de localização, ou seja, as coordenadas geográficas, que são armazenadas até serem enviadas ao servidor.

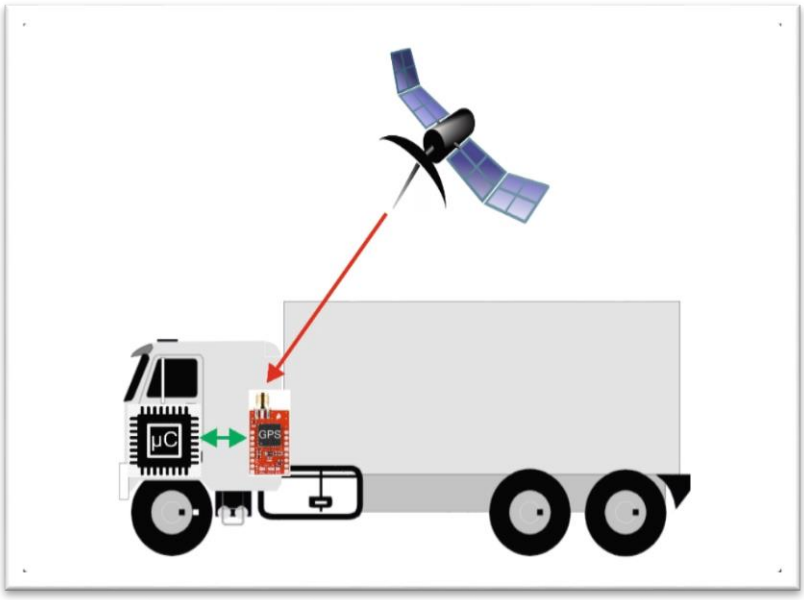


Figura 15 - Comunicação GPS

O módulo GPS utilizado é o Venus638FLPx, do fabricante *SkyTraq*. Este módulo envia, por porta série, tramas padrão *NMEA-0813*[9] ou binárias *SkyTraq*, a taxas de transmissão entre 9600bps e 115200bps. Estas tramas serão recebidas e interpretadas pelo *mbed*, de modo a extrair a informação pretendida, ou seja, as coordenadas da posição da viatura.

4.2.4 DADOS LOCAIS - OBD (*ON-BOARD DIAGNOSTICS*)

Para a aquisição de dados funcionais e mecânicos da viatura, a ligação OBD-II faz-se através das portas *CAN bus* do microcontrolador. Para tal, utiliza-se um *transceiver CAN* para a implementação da camada física de *interface* do sistema de diagnóstico do carro e o *bus* do microcontrolador, fazendo a ponte entre o protocolo CAN e o barramento físico do protocolo (ISO-11898). O *transceiver* utilizado é o *MCP2551* do fabricante *Microchip*.

[9] NMEA 0183 é um padrão de dados definido pela *National Marine Electronics Association*, utilizado na comunicação de dispositivos como sonares, giroscópios, GPS e piloto automático.

Esta ligação é bidirecional pois, de modo a obter os dados da viatura, o microcontrolador deve enviar para esta uma requisição. Esta requisição informa o *ECU* que dados pretendem ser obtidos, pela passagem do respetivo PID (Figura 16).

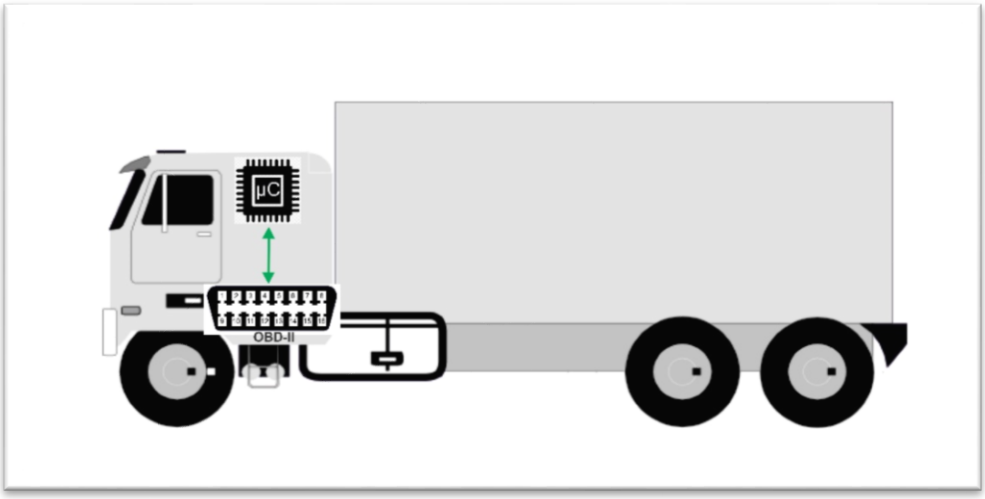


Figura 16 - Comunicação OBD-II

4.2.5 GSM/GPRS

O módulo principal terá ainda como função enviar a um servidor, periodicamente, todos os dados adquiridos dos sensores e do módulo GPS. Visto que o sistema visa a implantação em viaturas de entrega, estando estas durante o trabalho em circulação, a maneira mais viável de enviar os dados à central é através da rede móvel. Para tal, o sistema conta com um módulo GSM/GPRS (Figura 17).

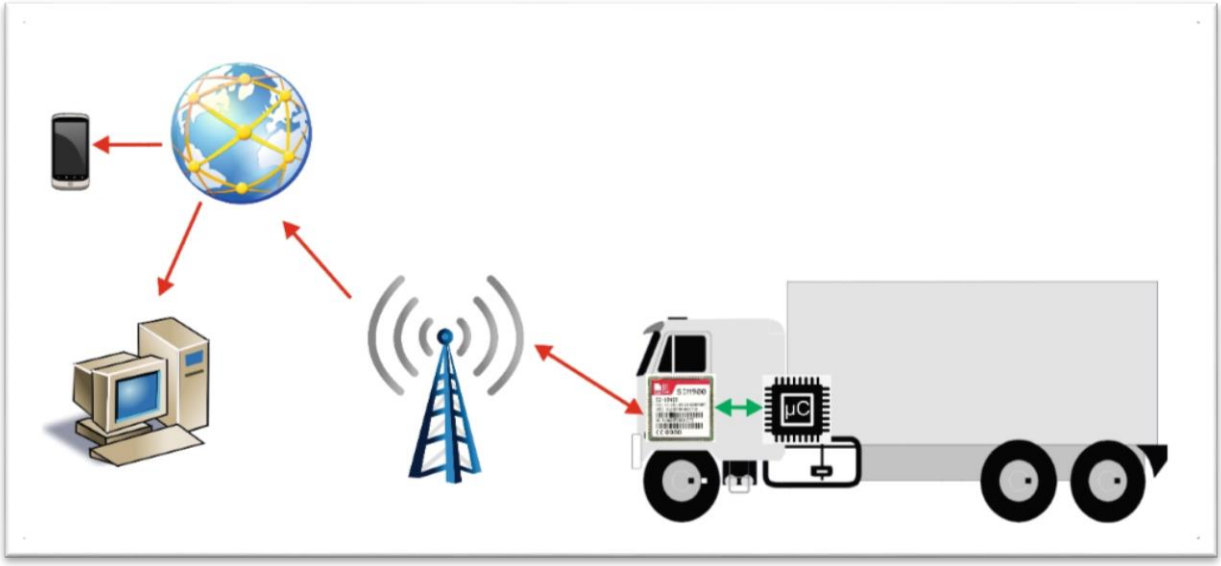


Figura 17 - Comunicação GPRS

O módulo principal, depois de adquirir e formatar os dados desejados, deverá ligar-se periodicamente à internet através de GPRS, enviando os dados para o servidor. Uma vez entregues, o módulo desliga-se da rede móvel, voltando ao estado de leitura de dados (e repetindo sucessivamente estes procedimentos).

Os dados, uma vez alojados no servidor, estarão disponíveis ao utilizador através de uma interface *web*. Para além de serem enviados para o servidor, os dados são também guardados como cópia de segurança numa memória *secure digital* removível (cartão *microSD*).

O módulo utilizado para esta ligação GSM/GPRS é o *SIM900* do fabricante *SimCom*. O microcontrolador liga-se a este módulo através de porta série, controlando-o através de comandos AT[10].

Todos estes processos atrás descritos estarão a acontecer periodicamente, de modo a proporcionar à central informação em tempo real e virtualmente ininterrupta sobre o estado de toda a frota, no que diz respeito às localizações, dados sensoriais e estado da viatura.

[10] Nome mais comumente utilizado para se referir ao *Hayes command set*, conjunto de comandos especificamente criados para configurar e controlar o modem *Hayes Smartmodem 300*, lançado em 1981. Desde então são largamente utilizados em equipamentos de telecomunicações.

5. IMPLEMENTAÇÃO

O presente capítulo descreve a implementação do sistema. Tratando-se de uma implementação baseada em sistemas embebidos, está dividido em duas secções, *hardware* e *firmware*, como é típico nestas aplicações. Na primeira parte serão descritas todas as interligações entre os componentes dos módulos e periféricos, bem como ilustrados e explicados os circuitos desenhados e configurações inerentes. Na segunda parte será detalhada toda a programação do microcontrolador, tendo em conta os periféricos e configurações específicas destes. Ainda na segunda parte serão detalhadas as configurações específicas a utilizar nos *XBees* remotos e no coordenador local.

5.1 HARDWARE

O *hardware* desenvolvido neste projecto corresponde à implementação física dos módulos principal e remoto, que está detalhada na presente secção. Todos os esquemáticos e placas de circuito impresso aqui apresentadas foram desenhados com o recurso à ferramenta *Altium Designer*.

5.1.1 MÓDULO PRINCIPAL

Foi desenvolvida uma placa de circuito impresso para o módulo principal, de modo a interligar o microcontrolador e respetivos periféricos.

5.1.1.1 Mbed

Como se pode observar na Figura 18, o *mbed* dispõe de três portas série bidirecionais (Rx e Tx), nas quais serão ligados os módulos GPS (pinos 27 e 28, respectivamente), o

módulo para a comunicação GSM/GPRS (pinos 9 e 10) e o *XBee* coordenador (13 e 14). Os pinos 29 e 30 pertencem ao *CAN bus*, onde será ligado o circuito do *transceiver* MCP2551, fazendo a interface ao conector J1962 para a ligação ao OBD-II. Nos pinos 5 a 8 será ligada a memória SD, utilizando tecnologia SPI. Nos pinos 16 e 17, pertencentes às entradas analógicas, estará ligado o circuito de endereçamento.

A alimentação de 5V é feita através do pino 2, sendo o pino 1 o *ground*. Ao pino 3 (VBAT) liga-se uma pilha de moeda, de maneira a manter correto o RTC (Real Time Clock). O pino 40 é uma saída de 3.3V, que será utilizada para alimentar os periféricos que trabalhem a esta tensão de alimentação, nomeadamente o *XBee* e o módulo GPS.

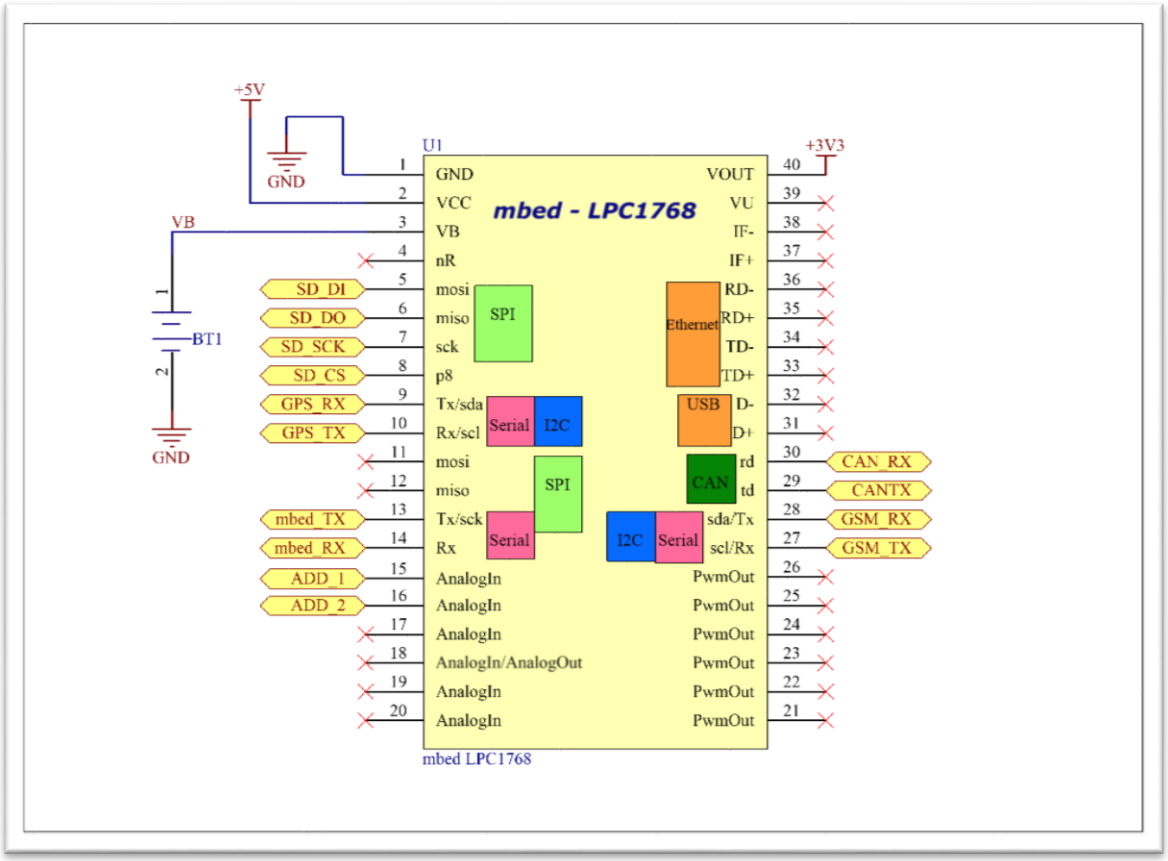


Figura 18 - Ligações ao mbed LPC1768

5.1.1.2 Endereçamento

Relativamente ao endereçamento dos módulos, tratando-se de uma rede ZigBee, os módulos participantes na rede devem ser configurados com o mesmo *PAN ID* (*Personal*

Area Network ID) ou identidade de rede, de modo a comunicarem uns com os outros e não interferirem com redes vizinhas (outras viaturas). Assim, cada viatura possui uma rede de sensores e, deste modo, um *PAN ID* próprio. No ato da instalação, ou mesmo em caso de necessidade de substituição dos módulos principal ou remotos, os módulos a instalar devem ser configurados com a referida identidade.

Deste modo foi criado um sistema para a programação dos mesmos. Será incluído na placa principal, um *DIL switch* com 8 interruptores (Figura 19), que representará o endereço da viatura em questão. Com este sistema é possível designar 255 endereços distintos, ou seja, cada viatura possuirá um código binário de 8 bits, representado pelos interruptores do *switch*, criando um método visual e pouco complexo de se endereçar o módulo.

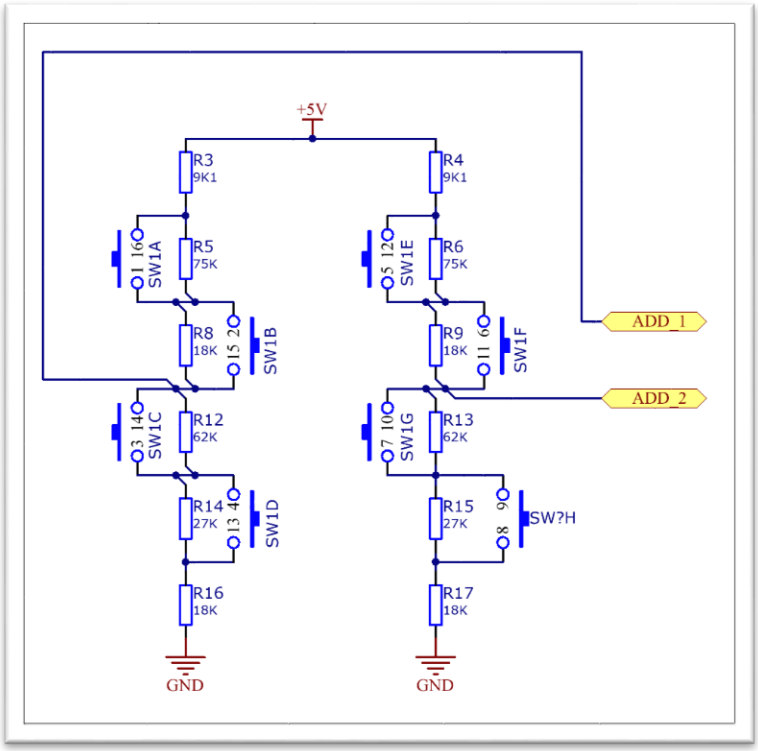


Figura 19 - Interruptores de Endereçamento

O sistema consiste em dois divisores de tensão idênticos, representando os *nibbles* menos e mais significativos. Os divisores são formados por seis resistências cada, em que quatro delas (de cada lado) estão ligadas a uma das chaves do *switch*. Desta forma, ao ligar-se uma das chaves, está-se a sobrepor a resistência com um curto-circuito. O resultado é uma tensão única para cada combinação de chaves ligadas de cada um dos lados. Estas

tensões são lidas pelos conversores analógico-digitais do *mbed* e interpretadas de acordo com uma tabela de valores, que será explicada na secção de *firmware*.

De maneira a facilitar também a programação de módulos remotos de substituição, um mecanismo também foi criado para este fim. Visto a plataforma *mbed* somente disponibilizar três das portas série do processador e já existirem três dispositivos ligados às mesmas, uma das portas deverá ser partilhada para este fim.

Este mecanismo consiste em um seletor e um conector, ou seja, conecta-se o módulo remoto ao principal e, através de um seletor físico deslizante (Figura 20), cujas posições conectam o módulo *XBee* local coordenador à porta série, de maneira a funcionar em modo normal, e na outra posição liga o módulo remoto, então conectado ao principal para que este seja programado.

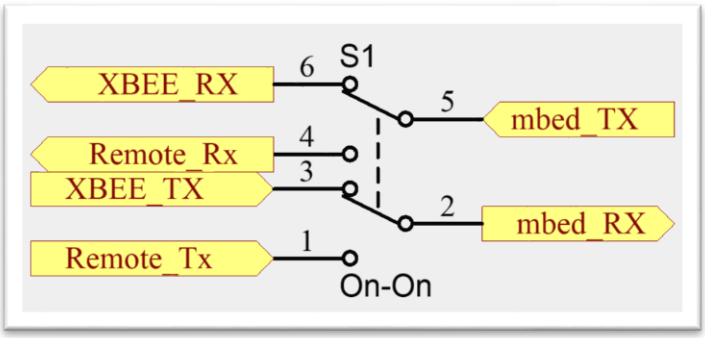


Figura 20 - Programador do Módulo Remoto

É de referir que este mecanismo foi pensado para que qualquer utilizador menos experiente conseguisse facilmente fazer a configuração dos módulos *zigbee*. Desta forma, apenas pela utilização de *switches* mecânicos, conse de forma a tornar transparente para o utilizador todos os aspectos complexos de configuração.

5.1.1.3 Transceiver CAN

O circuito do *transceiver* MCP2551 foi dimensionado de acordo com a sua documentação. Os portos CAN_TX e CAN_RX são as ligações do *transceiver* ao *mbed*,

enquanto os portos CANH e CANL são as ligações à ficha J19652 de ligação ao OBD-II da viatura (Figura 21).

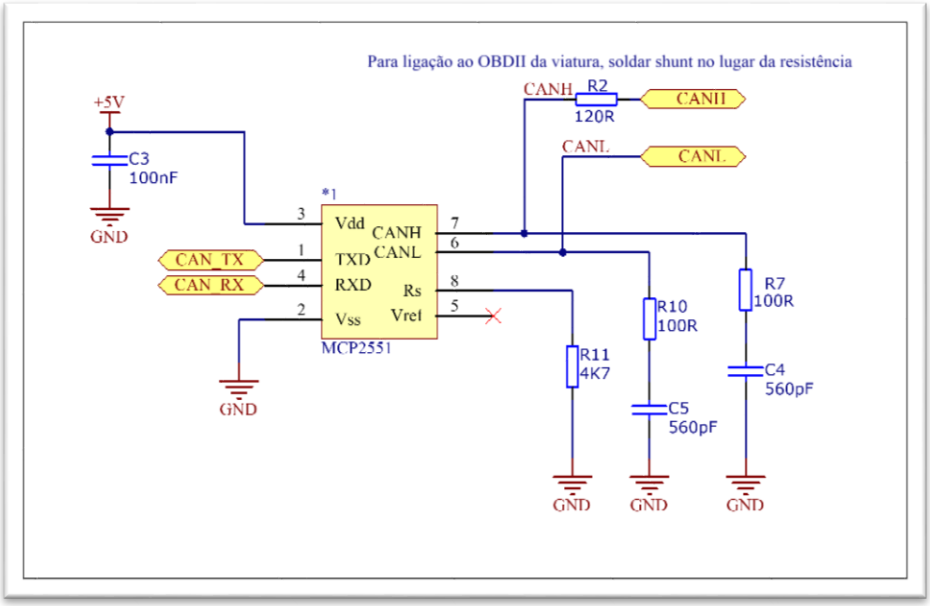


Figura 21 - Circuito CAN Bus

5.1.1.4 Fonte de Alimentação

O módulo principal deverá estar presente na cabine da viatura, não sendo deste modo necessária a utilização de baterias, uma vez que será ligado ao circuito elétrico da viatura. Por existirem dois padrões possíveis de alimentação em veículos, 12V e 24V, o circuito de alimentação foi dimensionado de maneira a suportar esta gama de valores.

Assim, foi selecionado o regulador LM2596 da *Texas Instruments*. Este regulador permite tensões de entrada até os 40V e saída fixa a 5V (no modelo escolhido). Os componentes adicionais foram selecionados de acordo com a documentação do fabricante, tendo-se obtido o circuito apresentado na Figura 22.

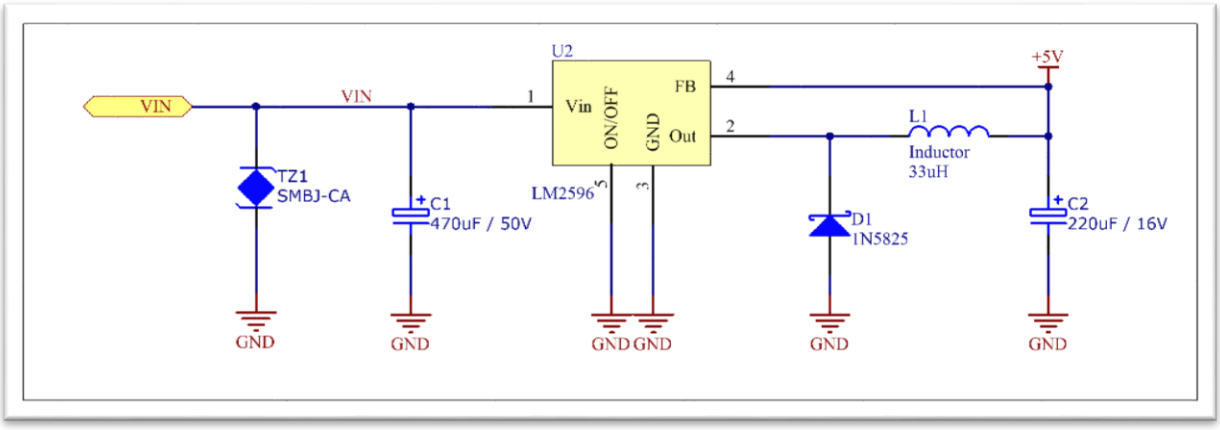


Figura 22 - Fonte de Alimentação

Uma viatura é um ambiente ruidoso, do ponto de vista elétrico, porque ligar e desligar o motor causa picos de tensão. Tratando-se de viaturas com outros equipamentos instalados, podem ocorrer ainda variações bruscas da tensão, para além de picos, devido a outros fatores e ao funcionamento de outros aparelhos (por exemplo, os sistemas de refrigeração). Deste modo, será inserido um *transorb* (*transient voltage supressor*, ou supressor de tensão transitória), modelo SMBJ-CA, à entrada da alimentação, com o objetivo de filtrar esses possíveis picos e proteger o circuito de sobretensões.

5.1.2 MÓDULOS REMOTOS

No desenvolvimento dos módulos remotos, o baixo consumo é um fator decisivo, uma vez que os módulos serão alimentados a baterias. Tendo em conta esse fator, não foram acrescentadas luzes automáticas de indicação de funcionamento ou de bateria, tendo sido apenas acrescentado um botão de pressão que liga um LED ao ser pressionado, de maneira a saber que ainda há carga, no ato da instalação ou da verificação e inspeção visual.

O módulo *Xbee* (Figura 23), será configurado como *end device AT*, ou seja, em modo terminal e controlado por comandos AT. Terá o modo de *sleep* cíclico definido, de modo a aumentar a longevidade da bateria, "acordando" periodicamente para enviar as amostras dos *ADCs*. De acordo com a documentação dos módulos e por ser boa prática na utilização

de *ADCs*, foram acrescentados condensadores de desacoplamento entre estes e a massa ou *ground* (Figura 23).

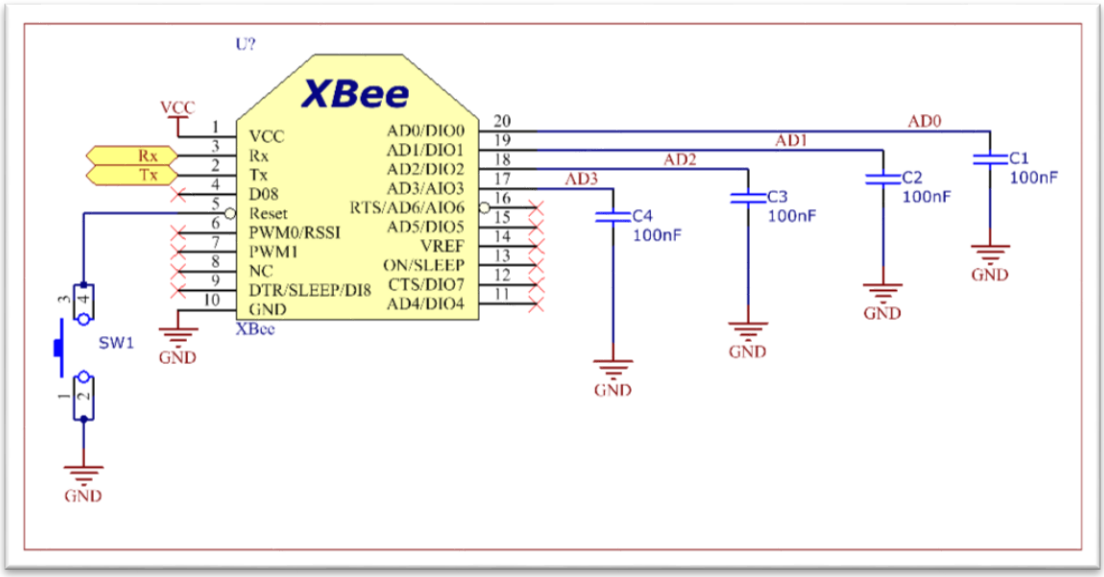


Figura 23 - Módulo Remoto - XBee

Dos quatro *ADCs* presentes no *XBee*, dois estarão ativos e, por defeito, utilizando a configuração mais simples dos módulos. Um estará conectado ao sensor de temperatura MCP9700 e o outro ao circuito monitor do nível da bateria. Uma vez que a tensão de referência interna do *Xbee* é de 1.2V e que o módulo será alimentado com duas baterias $LiFePo_4$ de 3.2V em série, totalizando 6.4V, foi necessário aplicar um divisor de tensão entre a tensão das baterias e o *ADC* que será utilizado para a monitorização do nível da bateria, de maneira a escalar o seu valor e adaptá-lo aos valores lidos pelo ADC. A equação (1) mostra o dimensionamento necessário para as resistências do circuito de medição do nível da bateria, enquanto que a Figura 24 apresenta as ligações do sensor de temperatura e o circuito para monitorização do nível de bateria.

$$V_{AD1} = V_{BAT} * \frac{R_2}{(R_1+R_2)} = 6.4 * \frac{460}{(2000+460)} \approx 1.2V \quad (1)$$

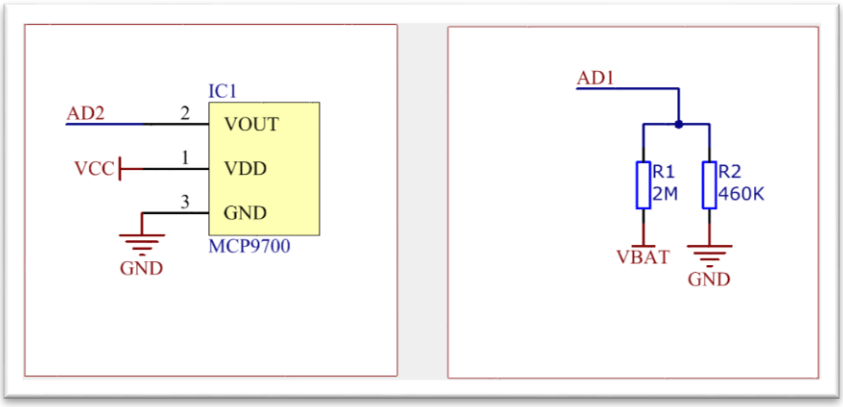


Figura 24 - Sensor de Temperatura e Monitor do Nível de Bateria

O módulo *Xbee* funciona a 3.3V, pelo que existirá a necessidade de aplicar uma regulação ao valor de tensão das baterias. Uma vez que a tensão mínima de descarga de uma célula $LiFePo_4$ é de aproximadamente 2.6V, e uma vez que o circuito estará alimentado por duas baterias em série, não se corre o risco de a tensão cair para níveis inferiores a 3.3V. Desta forma, não será necessária a utilização de um regulador *step-up*[11], tendo sido utilizado o regulador MCP1703 da fabricante Microchip, um LDO[12] (*low drop-out*) com tensão fixa a 3.3V (Figura 25).

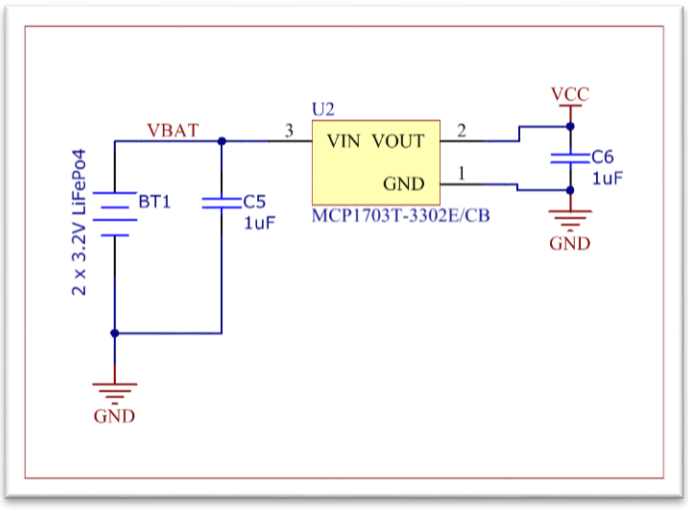


Figura 25 - Regulador DC-DC

[11] *Step up* ou *boost regualator* são reguladores de tensão comutados (*SMPS - switched mode power suply*), também chamados de conversores DC-DC, que têm tensão de saída superior à de entrada.

[12] *Low Drop-out regulator* trata-se de um regulador DC-DC com rácios saída/entrada muito baixos, chegando virtualmente ao valor mínimo de conversão. Apresentam elevada eficiência na conversão.

5.1.3 PLACAS DE CIRCUITO IMPRESSO

Uma vez definidas as ligações dos módulos principal e remotos e do desenho dos circuitos, estes foram montados e testados em *breadboards*, para depois se proceder ao projetado das placas de circuito impresso, utilizando o programa *Altium Designer*. Foram então fabricadas as *PCBs* (*Printed Cricuit Board* ou placas de circuito impresso), que são apresentadas na Figura 26 e Figura 27, respetivamente para o módulo principal e módulos remotos.

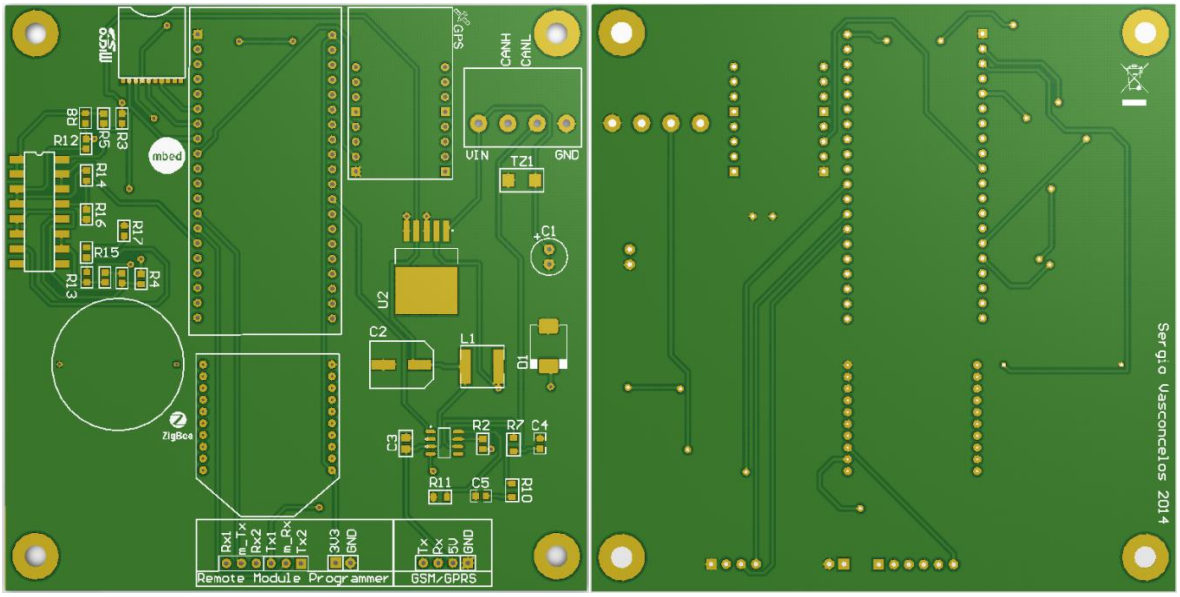


Figura 26 - Placas de Circuito Impresso – Módulo Principal - Frente e Verso

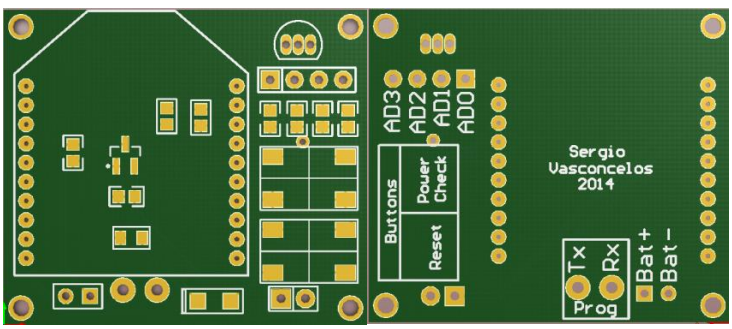


Figura 27 - Placas de Circuito Impresso - Módulos Remoto - Frente e Verso

5.1.4 LISTA DE MATERIAIS (*BOM – BILL OF MATERIALS*)

A lista de material necessário à construção dos módulos é apresentada seguidamente na Tabela 5 e na Tabela 6.

Tabela 5 - *Bill of Materials* - Módulo Principal

Botão Seletor	1
Schottky Regulador de Tensão 1N5825	1
2.54mm 1x4 Header	1
Bobina 200uH	1
Resistência 75K	2
Resistência 62K	2
Resistência 27K	2
Resistência 18K	4
Resistência 9K1	2
DIL Switch 8 posições	1
Suporte Bateria de Lítio em Moeda	1
Conector Cartão microSD	1
mbed LPC1768	1
Xbee Série 2	1
Seletor	1
Módulo Venus GPS SkyTaq	1
Condensador Eletrolítico 220uF / 16V	1
Condensador 100nF	1
Regulador de Tensão LM2596	1
Transceiver CAN Bus MCP2551	1
Resistência 4K7	4
Supressor de Picos SMBJ-CA	1
Bloco Terminal 1x4	1
Condensador 100nF	2
Condensador Eletrolítico 470uF / 50V	1

Tabela 6 - *Bill of Materials* - Módulo Remoto

2mm 1x10 header	2
Baterias 3.2V LiFePo4	2
Condensador 100nF	4
Condensador 1uF	2
Sensor de Temperatura MCP9700	1
Jumper	1
2,54mm 1x4 Header	1
LED SMD Vermelho	1
Resistência 470K	2
Resistência 10R	1
Botão de Pressão	2
Xbee Série 2	1
Regulador de Tensão MCP1703	1

5.2 FIRMWARE

Nesta secção será introduzida e detalhada a implementações do *firmware* para o módulo principal e a configuração do *XBee* dos módulos remotos e principal. O *firmware* do módulo principal foi desenvolvido na plataforma *online* do sistema *mbed*.

5.2.1 FLUXO DE PROGRAMA

Como é habitual na programação de microcontroladores, o *firmware* consistirá num ciclo infinito, dentro do qual estão incluídas todas as suas funções. Para mais facilmente ilustrar o fluxo de programa do módulo principal, apresenta-se na Figura 28 o seu fluxograma.

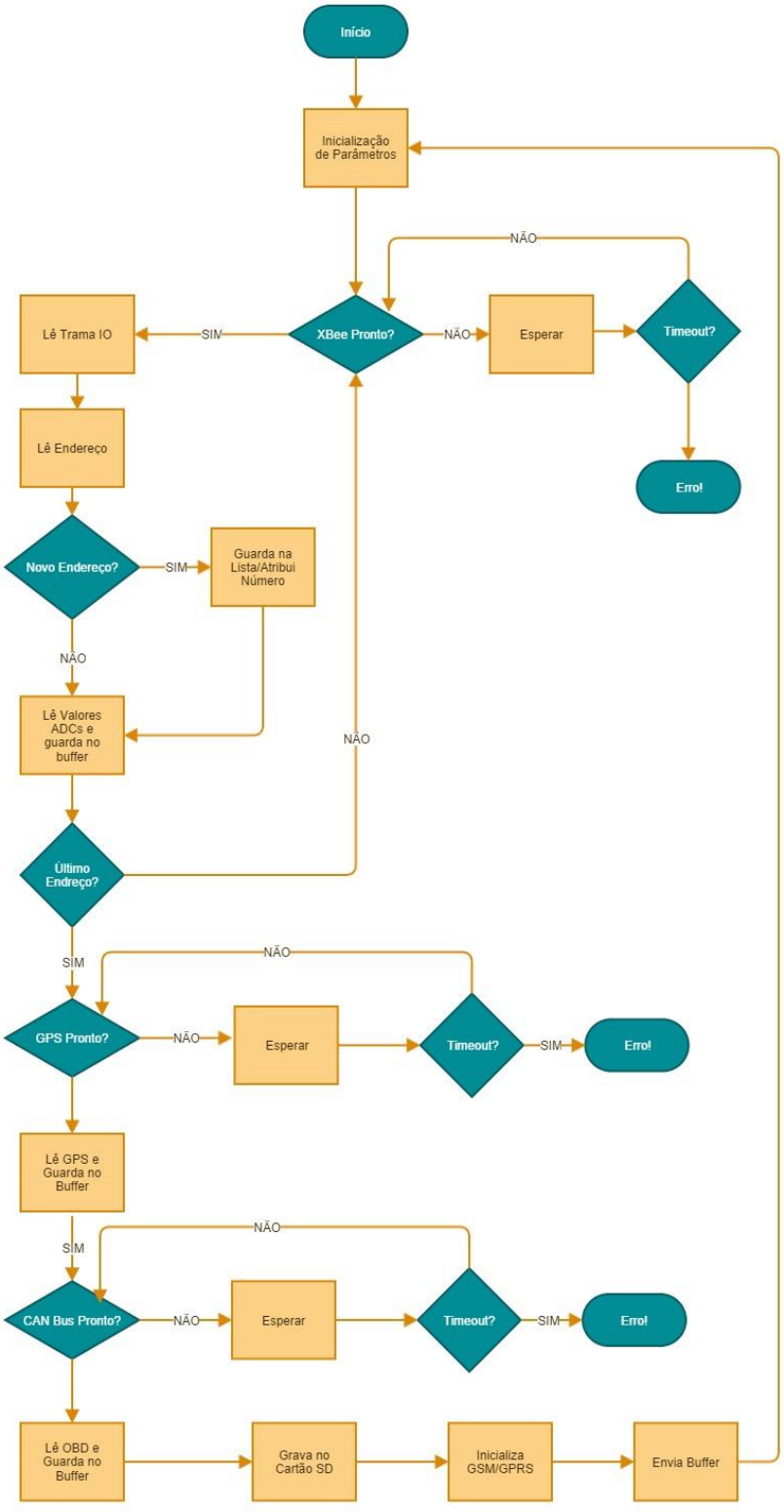


Figura 28 - Fluxograma – Firmware Ciclo Principal

Como ilustrado no fluxograma da Figura 28, o *firmware* estará ciclicamente a recolher todos os dados dos *XBee* remotos, coordenadas GPS, bem como os dados OBD em questão, de seguida gravando um *backup* no cartão SD e enviando para o servidor através de GPRS.

5.2.2 CONFIGURAÇÃO DOS XBEE E FORMATO DOS DADOS UTILIZADOS

Como referido em secções anteriores, neste projecto, o *XBee* ligado ao módulo principal será configurado como *coordinator*, estando os demais módulos da rede de sensores configurados como *end device* (módulos remotos).

Existem vários *firmwares* fornecidos pelo fabricante dos *XBee*, cada um deles apropriado para uma função ou conjunto de funções específicas. A instalação do *firmware* nos *XBee*, bem como as suas configurações específicas, é realizada através do *software* fornecido pela *Digi International*, o XCTU. Apesar de ser necessária a utilização do XCTU para a instalação/atualização de *firmware*, para a configuração dos módulos, é possível utilizar comandos AT enviados diretamente pela porta série, se assim for necessário. Este mecanismo permite que a própria aplicação interaja com os módulos remotos, enviando comandos e alterando a sua programação se a aplicação assim o exigir.

Na Figura 29 pode-se observar a interface do programa XCTU. Do lado esquerdo, sob *Radio Modules,* pode-se ver um módulo ligado localmente ao computador, na porta COM3, configurado como *coorddinator* e em modo API[13] (*application programming interface*). Por baixo deste lê-se *1 remote module*, ou seja, um módulo ligado remotamente ao módulo coordenador, configurado como *end device* em modo AT. Também é possível configurar e atualizar os módulos remotamente, utilizando esta ligação. Do lado direito pode-se ver algumas das configurações disponíveis, sendo as mais importantes detalhadas em seguida.

ZigBee Coordinator API e ZigBee End Device AT são os *firmwares* instalados nos dois módulos desenvolvidos neste projeto (principal e remoto, respetivamente).

[13] API, *application programming interface,* ou interface de programação de aplicações, consiste numa camada de abstração criada num *software* específico, possibilitando a comunicação com o mesmo sem a modificação da sua programação.

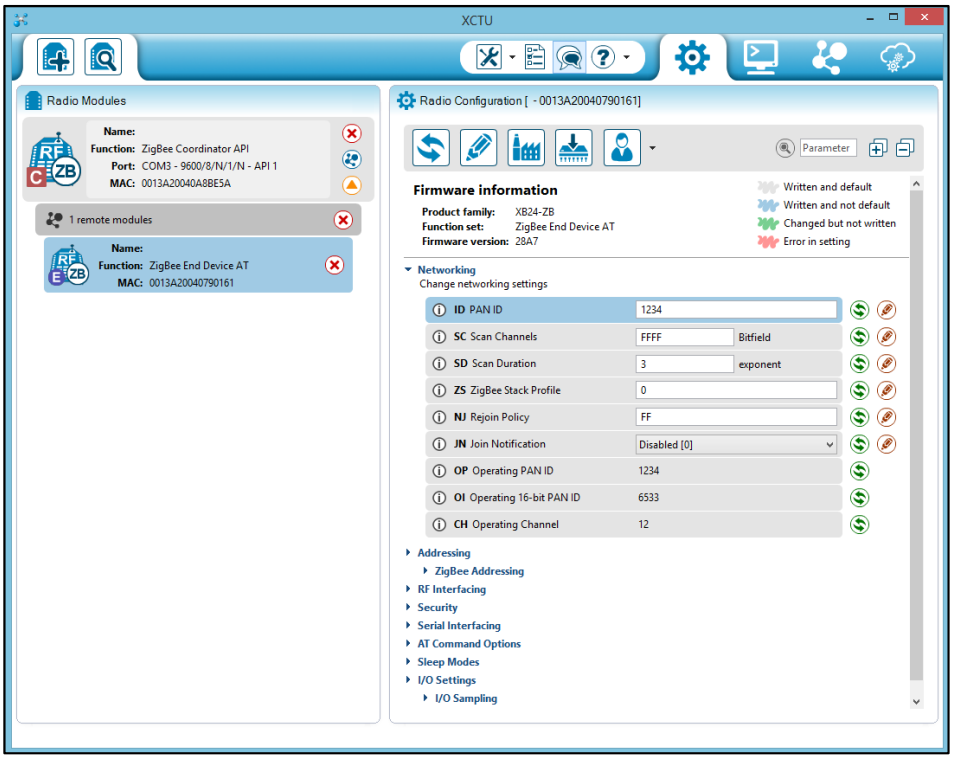


Figura 29 – XCTU

De maneira a que a comunicação seja feita apenas com os membros da rede e não haver interferência com redes adjacentes, todos os módulos de uma rede específica deverão ter o mesmo *PAN ID*. A título de exemplo, como se pode ver na Figura 29, os módulos pertencem à rede "1234".

No caso do *coordinator*, apenas a instalação do *firmware* e a configuração do *PAN ID* e o módulo está configurado.

Para os *end devices*, mais configurações deverão ser aplicadas, nomeadamente o período de *sleep*, a ativação dos *ADCs* necessários e a *IO sampling rate*, ou seja, os intervalos de tempo com que os módulos enviarão amostragens dos valores analógicos lidos.

Na Figura 30 pode-se observar os parâmetros disponíveis no XCTU relativos aos *Sleep Modes* (SM), que, como indicado pelo primeiro parâmetro, define o tipo de ciclo de *sleep* ao qual o módulo estará sujeito (neste caso o *sleep* cíclico). Sendo um dos requisitos do projeto fazer leituras aos sensores de minuto a minuto, os módulos serão configurados para "dormir" durante 59 segundos, "acordar" durante 1 segundo e voltar a "dormir".

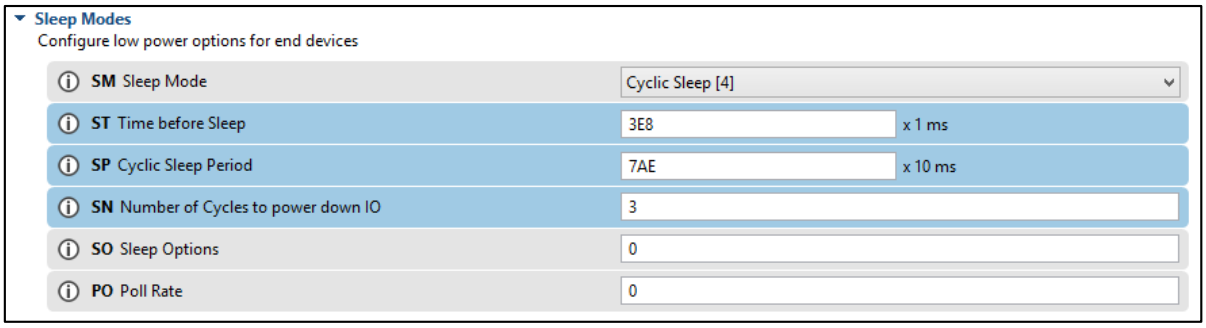


Figura 30 - XCTU Sleep Modes

O campo ST (*Time Before Sleep*) corresponde ao tempo que o módulo permanece ligado, até entrar novamente em modo sleep, e será configurado para 1 segundo ou seja, 1000 milissegundos, o que corresponde ao valor hexadecimal 3E8.

Os campos SP (*Cyclic Sleep Period*) e SN (*Number of Cycle to Power Down IO*), correspondem ao período de *sleep* propriamente dito e à quantidade de ciclos a efetuar até sair do *sleep*. Estes devem ser configurados em conjunto, ou seja, visto que o maior valor possível a ser introduzido no campo SP é de 28 segundos, e pretendemos um ciclo de 59 segundos, iremos então configurar o módulo de modo a executar 3 ciclos de 19,66 segundos, conseguindo um ciclo de 58.98 segundos, aproximadamente o tempo pretendido. O campo SP é multiplicado por 10 milissegundos, pelo que neste campo deve ser introduzido o valor de 1966, em hexadecimal 7AE. No campo SN será inserido o número de ciclos a executar, ou seja, 3, coincidindo com o seu valor hexadecimal.

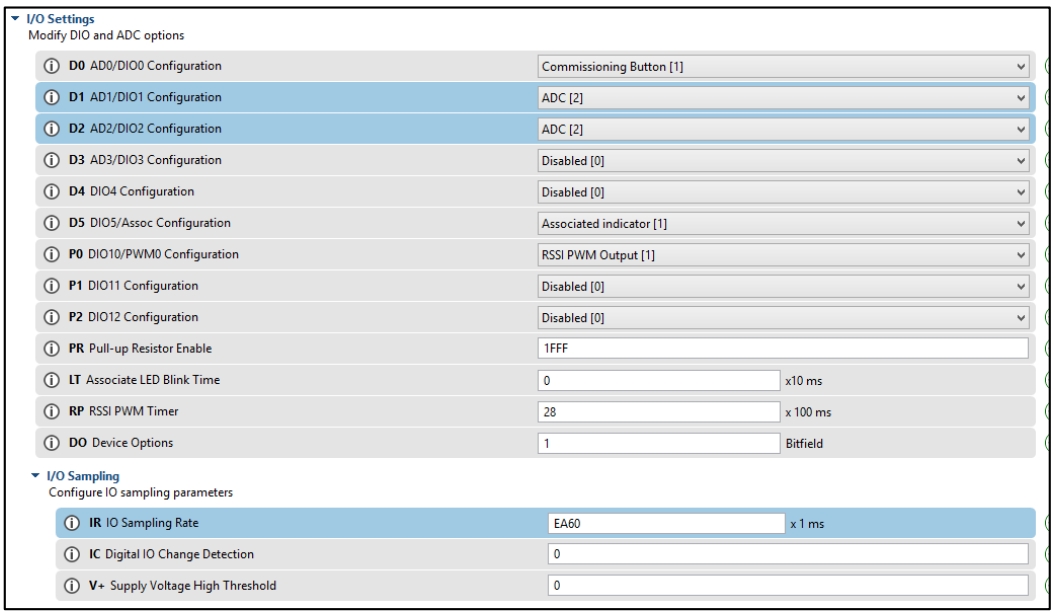


Figura 31 - XCTU Configurações I/O

Neste projeto, estarão ativos os *ADCs* 2 e 3 dos módulos. Desta maneira, também se deve configurar a partir do XCTU as funções de D1 e D2 como *ADCs* e a taxa de amostragens (Figura 31).

Uma vez configurados os *XBee*, é necessário conhecer a estrutura da trama de dados que vai ser utilizada no envio dos dados entre os módulos remotos e o módulo principal. Com esta informação, é possível retirar os dados, que sejam de interesse para a aplicação do módulo principal, da trama de dados recebida pelo *XBee* local.

A composição desta trama de dados recebida é apresentada na Tabela 7. Por defeito, os dois *ADCs* estarão ligados e assim as tramas terão 24 Bytes, podendo-se deste modo descartar os dados sem utilidade para a aplicação. A trama é lida no módulo principal pela função *XbeeRead(),* depois de detetar o carácter de início.

De acordo com a Tabela 7, pode-se descartar os 4 bytes iniciais, lendo-se os 8 bytes seguintes que se tratam do endereço físico do módulo remoto a transmitir esta trama. Descartam-se os próximos 6 byte, lendo-se então os 4 bytes seguintes, pois tratam-se do MSB (*most significant byte*) e LSB (*least significant byte*) dos valores lidos pelos dois *ADCs* ativos.

A função *ADCtoTemp* receba os bytes mais e menos significativos, aos pares, calculando de seguida o valor analógico lido e convertendo para temperatura. Utilizando a fórmula de conversão disponibilizada na documentação do sensor, neste caso o MCP9700, utilizam-se as equações (2) e (3), respetivamente para obter no microprocessador a leitura analógica do sensor e a sua conversão para valores de temperatura.

$$\boldsymbol{Leitura\ Anal\acute{o}gica} = \left(\boldsymbol{LSB} + (\mathbf{256} * \boldsymbol{MSB})\right) * \left(\frac{\mathbf{1.2}}{\mathbf{1024}}\right)[\boldsymbol{V}] \quad (2)$$

$$\boldsymbol{Temperatura} = \frac{\boldsymbol{Leitura\ Anal\acute{o}gica - V_0}}{\boldsymbol{T_c}} \quad (3)$$

Para a leitura do nível de carga da bateria, a função *ADCtoV* apenas aplica a fórmula (2) respetiva à leitura analógica.

Tabela 7 - XBee Trama de Dados do Tipo 0x92 - Data Sample Rx Indicator

Nome do Campo	Valor	Descrição
Delimitador	7E	Início dos dados
Tamanho	#### (2 bytes)	Nº de bytes entre os campos tamanho e *checksum.*
Tipo de Trama	92	Tipo de trama, neste caso 0x92 – IO Data Sample Rx Indicator
Endereço 64-Bit	00 00 00 00 00 00 00 00	Endreço físico
Endereço 16-Bit	00 00	Endreço físico
Opções de Receção	00	Estado de envio
Nº de Amostragens	00	Número de amostragens na trama
Máscara Digital	00 00	Número de entradas digitais ativas (de acordo com o *data sheet*)
Máscara Analógica	00	Número de entradas analógicas ativas (de acordo com o *data sheet*)
Amostragens	MSB+LSB x Numero de *ADCs*	Valores das leitoras dos *ADCs*
Checksum	##	0xFF menos a soma de 8 bits dos bytes entre o tamanho e o *checksum*

5.2.3 MÓDULO GPS

O módulo *Venus GPS* da *SkyTraq*, como referido anteriormente, estará constantemente a enviar tramas NMEA através da porta série. Assim, a função *GPSRead* encarregar-se-á de detetar e selecionar a trama desejada, ler esta trama e dividir em partes a informação que a mesma carrega, de maneria a se retirar a informação desejada. Esta função permite a

seleção da trama desejada, bem como a escolha dos dados que se pretende extrair da mesma. Através de um *parser*, a informação desejada é extraída e formatada. No caso deste projeto, a informação importante é a latitude, longitude e respetivos pontos cardeais. Estas informações estão localizadas na trama GGA (*Global Positioning System Fix data*), com a seguinte estrutura definida na Tabela 8:

Tabela 8 - Estrutura da Trama NMEA GGA

$GPGGA,hhmmss.sss,ddmm.mmmm,a,dddmm.mmmm,a,x,xx,x.x,x.x,M,,,,xxxx*hh<CR><LF>

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

Sendo o caracter "$" o início da trama e "GPGGA" a identificação da mesma, dos restantes campos serão utilizados nesta aplicação os campos 2, 3, 4 e 5.

- Campo 2 – Latitude no formato ddmm.mmmm ('d' são graus e 'm' minutos decimais).
- Campo 3 – Indicador de Norte ou Sul no formato "N" ou "S".
- Campo 4 – Longitude no formato dddmm.mmm
- Campo 5 – Indicador de Este ou Oeste no formato "E" ou "W"

5.2.4 DADOS DO *ECU* POR OBD-II/CAN BUS

Relativamente aos dados do *ECU*, para a programação as funções que permitem obter os dados da viatura, o ficheiro *ECU_Reader.h* (apresentado no Apêndice B – Firmware - Código) apresenta a lista completa dos *PID*'s padrão. A função *ECU_Read()* recebe como argumento uma opção referente ao PID que se pretende questionar ao *ECU* do veículo. Esta função é facilmente expansível no futuro, sendo possível acrescentar quaisquer opções baseadas nas listadas no ficheiro mencionado.

Neste projeto, serão lidos os valores do sensor *MAF* (*mass air flow*) e *VSS* (*vehicle speed sensor*). O sensor *MAF* mede a massa de ar consumida pelo motor em gramas por segundo.

O sensor *VSS* mede a velocidade instantânea do veículo. Sendo a quantidade de combustível proporcional à massa de ar passante para o motor, em condições normais a relação desta massa para quantidade de combustível é conhecida. Desta forma, é possível calcular o consumo de combustível instantâneo e, por conseguinte, a média e o histórico [20].

De acordo com a regulamentação atual, o valor ideal do rácio ar/gasolina é de 14.7g/g. Neste projecto apenas foi considerada a gasolina, devido ao carro disponível para testes ser movido a este combustível. Assim, o *firmware* executará os seguintes passos de modo a calcular o consumo:

- Dividir o valor recebido dos sensor *MAF* por 14.7, obtendo-se a quantidade de combustível em grama por segundo.
- Dividir o valor da massa de combustível por segundo em por 748.9 [21], de modo a obter o volume em litros por segundo.
- Multiplicar o valor obtido por 3600, obtendo-se o valor de litros por hora.
- Dividir o valor obtido pelo valor lido pelo sensor *VSS*. Este valor é em km/h, e assim obtém-se o consumo em l/km.
- Multiplicando o valor obtido por 100, obtém-se a medida de consumo mais comum utilizada na europa, litros/100 km.

A simplificação dos dados anteriores resulta na seguinte equação (4):

$$Litros/100km = \frac{MAF*32.7}{VSS} \quad (4)$$

5.2.5 TRATAMENTO, ENVIO E GRAVAÇÃO DOS DADOS

Depois de obtido, o conjunto de dados, descrito nas secções anteriores deverá ser formatado convenientemente de modo a ser enviado ao servidor, bem como ser guardado em cópia de segurança no cartão *microSD*. Desta forma, foi definido que os dados serão

formatados numa *string*, separados por vírgula, com o seguinte formato, que se descreve na Tabela 9:

$AAAA.DD.MM,HH.MM.SS,XX.X,YY.Y,ZZ.Z, LLLL.LLLL.C, lll.llll.c,V.V,C.C;

Tabela 9 - Formato dos Dados

Caracteres	Função	Exemplo
$	Caracter de início de trama	N/A
AAAA.DD.MM	Ano.Mês.Dia	2014.03.01
HH.MM.SS	Horas.Minutos.Segundos	12.25.10
XX.X, YY.Y, ZZ.Z	**x**, **y** e **z** são as temperaturas das três possíveis câmaras refrigeradas. Os sensores inativos enviam '0'.	-19.1, 4.0, 0
LLLL.LLLL.C, lll.llll.c	Latitude e longitude em formato decimal, mais respetivos pontos cardeais	3701.6413,N,756.3646,W
V.V	Tensão na bateria	1.2V
C.C	Consumo do veículo	5.3
;	Fim da trama	N/A

Depois de lidos e armazenados, os dados serão enviados para o servidor através de GPRS. Para tal, o *mbed* será pré-configurado com uma série de comandos AT específicos do módulo GSM/GPRS utilizado, de maneira a iniciar uma conexão IP com a rede móvel.

Será implementado um sistema de modo que o microcontrolador fique à espera de receber a resposta do módulo, a string "OK".

Os comandos enviados serão os seguintes:

AT+CGATT?

AT+SAPBR=3,1,"CONTYPE","GPRS"

```
AT+SAPBR=3,1,"APN","internet"
AT+SAPBR=1,1
AT+HTTPINIT
AT+HTTPPARA="CID",1
AT+HTTPPARA="URL","127.0.0.1/data.php?dat='string de dados dos sensores, GPS e OBD'"
AT+HTTPACTION=0
AT+HTTPTERM
```

5.3 SERVIDOR

De forma a testar completamente o sistema desenvolvido, foi desenvolvido um pequeno servidor para que o sistema fique completo. Nesta secção apresenta-se o parser desenvolvido e a interface *web* do servidor.

5.3.1 PARSER

Com recurso à linguagem PHP, foi desenvolvido um *parser* (interpretador de dados), que interpreta a trama recebida no lado do servidor. Esta aplicação no lado do servidor tem como objetivo a preparação dos dados para a sua apresentação ao utilizador bem como a gravação num ficheiro de *backup.*

A criação destes mecanismos, no lado do servidor, visa a simplicidade e eficiência no envio por parte do módulo presente na viatura. Assim, pode-se enviar a menor quantidade possível de dados, não sendo necessário alocar recursos no microcontrolador e minimizando a possibilidade de erro.

O *parser* foi desenvolvido tendo em conta a formatação proposta na secção anterior. Desta forma, a trama será enviada ao servidor pelo método *POST*, pelo módulo GPRS, de acordo com o exemplo seguinte:

http://127.0.0.1/data.php?dat="$AAAA.DD.MM,HH.MM.SS,XX.X,YY.Y,ZZ.Z,LLLL.LLLL.C,lll.llll.c,V.V,C.C;"

O ficheiro *data.php* (apresentado no Apêndice C – Interface - Código) possui o mecanismo para "aceitar" os dados enviados por *POST* e gravá-los em dois ficheiros, uma cópia no ficheiro temporário *data.bin* e outra cópia no ficheiro de *backup log.bin*. No ficheiro *data.bin*, apenas uma linha de dados estará presente de cada vez, ou seja, o *parser* irá, periodicamente, aceder a este ficheiro e retirar a trama presente de modo a interpretar e amostrar os últimos valores lidos. No ficheiro *log.bin*, os dados serão consecutivamente gravados, linha a linha e com uma *timestamp*, permitindo o acesso aos dados para a aplicação interna ou aplicações futuras.

Seguidamente, o *parser* situado no ficheiro principal irá abrir o ficheiro *data.bin* e, se houver dados presentes no mesmo, passará à leitura e interpretação dos dados (Figura 32). Estes serão temporariamente gravados em variáveis separadas, possibilitando assim a sua utilização na amostragem.

```php
<?php
  $LastDataFile = "data.bin"; //Ficheiro de dados
  $fh = fopen($LastDataFile, 'r') or die("can't open file");
  $stringData = fgets($fh);
  $Time = fgets($fh);
  fclose($fh);

  $SeparateString = explode(",",$stringData);
  $Temp1 = $SeparateString[0];  // Temperatura Sensor 1
  $VBat1 = $SeparateString[1];  // Tensao Bateria Sensor 1
  $Temp2 = $SeparateString[2];  // Temperatura Sensor 2
  $VBat2 = $SeparateString[3];  // Tensao Bateria Sensor 2
  $Temp3 = $SeparateString[4];  // Temperatura Sensor 3
  $VBat3 = $SeparateString[5];  // Tensao Bateria Sensor 3
  $Lat1 = $SeparateString[6];   // Latitude Decimal
  $Lat2 = $SeparateString[7];   // Latiude - Ponto Cardeal
  $Lon1 = $SeparateString[8];   // Longitude Decimal
  $Lon2 = $SeparateString[9];   // Longitde - Ponto Cardeal
  $Cons = $SeparateString[10];  // Consumo
 ?>
```

Figura 32 - *Interpretador de Dados*

Para a amostragem de gráficos de temperaturas e consumos, foi utilizado o serviço *Thingspeak*. Este serviço consiste numa plataforma de gráficos *online*. Depois de criados os gráficos desejados, é possível o envio de dados posteriormente através de uma *API*.

O envio de dados pela API segue o seguinte formato:

https://api.thingspeak.com/update?api_key=YOUR_CHANNEL_API_KEY&field1=-18

O campo "YOUR_CHANNEL_API_KEY" refere-se ao nome único do canal criado na *interface* de utilizador do Thingspeak. O campo "field1" refere-se ao 1º campo, ou seja, o primeiro gráfico pertencente a este canal. Cada canal pode ter até 8 campos, ou seja, 8 gráficos. No canal criado para este caso, foram adicionados 3 campos, temperatura, tensão na bateria e consumos de combustível. A Figura 33 apresenta um exemplo do gráfico gerado para os consumos de combustível.

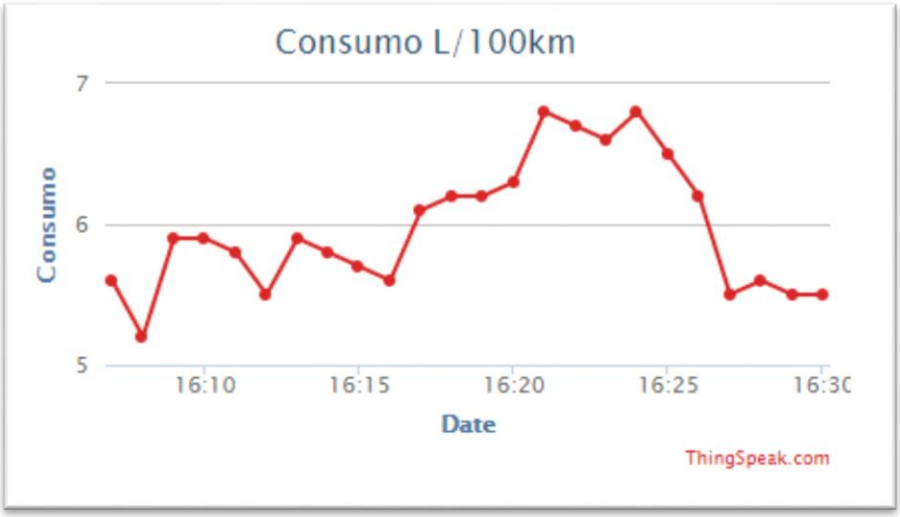


Figura 33 - Exemplo Thingspeak

Para a localização geográfica foi utilizada a *API* estática do *Google Maps*. Esta ferramenta permite enviar localizações para serem apresentadas em um mapa estático do *Google maps*, em formato de *bitmap* e apresentado em uma página. De modo a enviar para esta *API* os dados formatados corretamente, e de modo a gerar um histórico, foi criado um ficheiro "*gps.bin*" onde as coordenadas em formato decimal são acrescentadas e formatadas de acordo com os padrões da *API* em questão.

5.3.2 INTERFACE

De modo a apresentar os dados recolhidos e gravados ao utilizador, de forma sucinta e centralizada, foi desenvolvida uma interface *web* para o servidor. Para o desenvolvimento desta interface foram utilizadas as tecnologias *HTML5*, *CSS*, *Java Script, jQery* e *PHP*.

Foram criadas as diversas secções da página web por meios de *CSS*. Para facilitar e tornar mais agradável a navegação, foi criado um menu *drop down* com recurso ao *jQuery*, utilizando para tal a biblioteca *open source Sooperfish* [22]. A interface permite ao utilizador rapidamente selecionar uma viatura e passar à visualização dos dados relativos à mesma, como se pode ver na Figura 34.

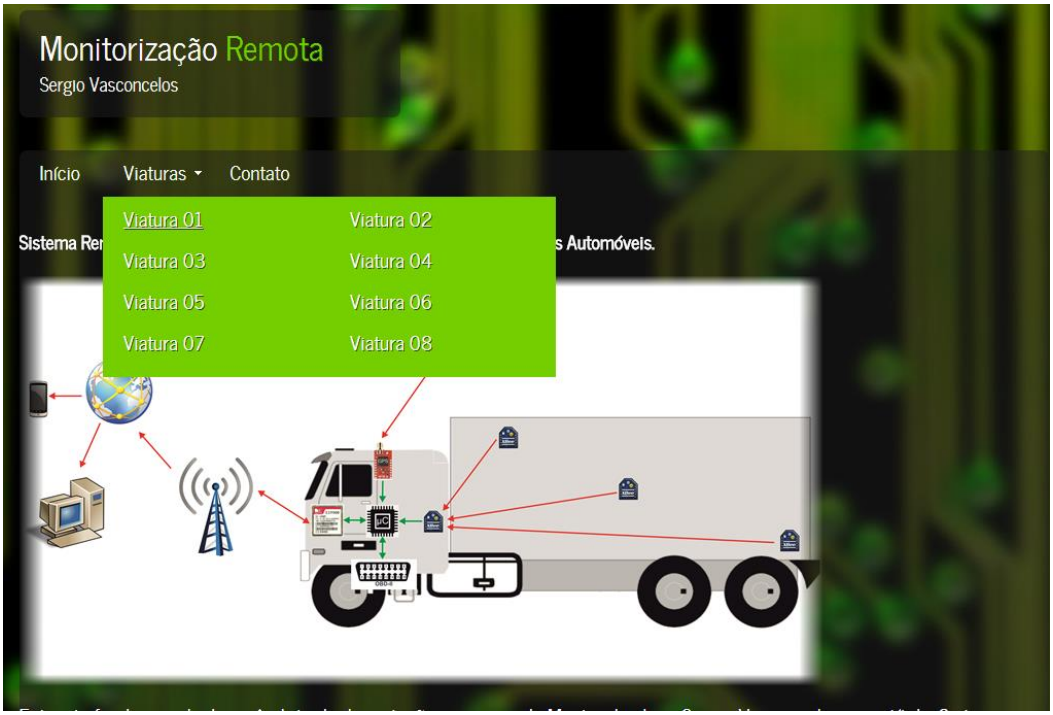


Figura 34 - Seleção de Viatura

Este servidor está disponível no seguinte endereço web: http://i3fr.goblineering.com/.

6. RESULTADOS EXPERIMENTAIS

Neste capítulo será demonstrado e exemplificado o funcionamento das partes integrantes do sistema, bem como o funcionamento geral deste e apresentação de resultados.

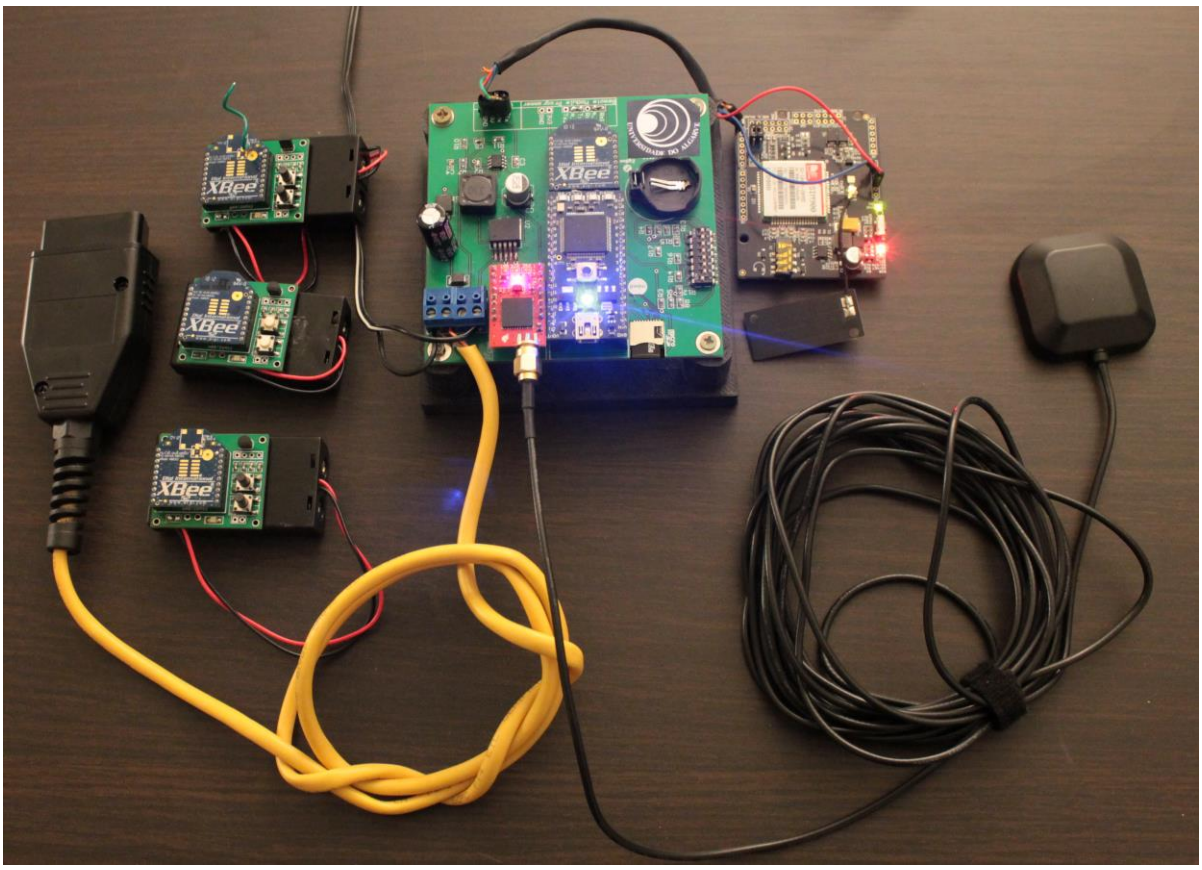


Figura 35 - Módulos Principal e Remoto

Primeiramente foram soldados os componentes às placas de circuito impresso. Foram montados um módulo principal e três remotos. Procedeu-se à ligação da placa de desenvolvimento GSM baseada no SIM900, à ligação do conector para OBD-II, às ligações para a entrada de alimentação e para a antena GPS. O resultado final é o apresentado na Figura 35.

A Figura 36 apresenta as partes integrantes do sistema, distinguidas por numeração, para de acordo com a lista seguinte:

1. Módulos Remotos
2. Conector OBD-II
3. Antena GPS Passiva
4. Placa de desenvolvimento SIM900
5. Antena GSM em PCB
6. Cartão microSD
7. Mbed
8. Módulo Venus GPS
9. Módulo Principal

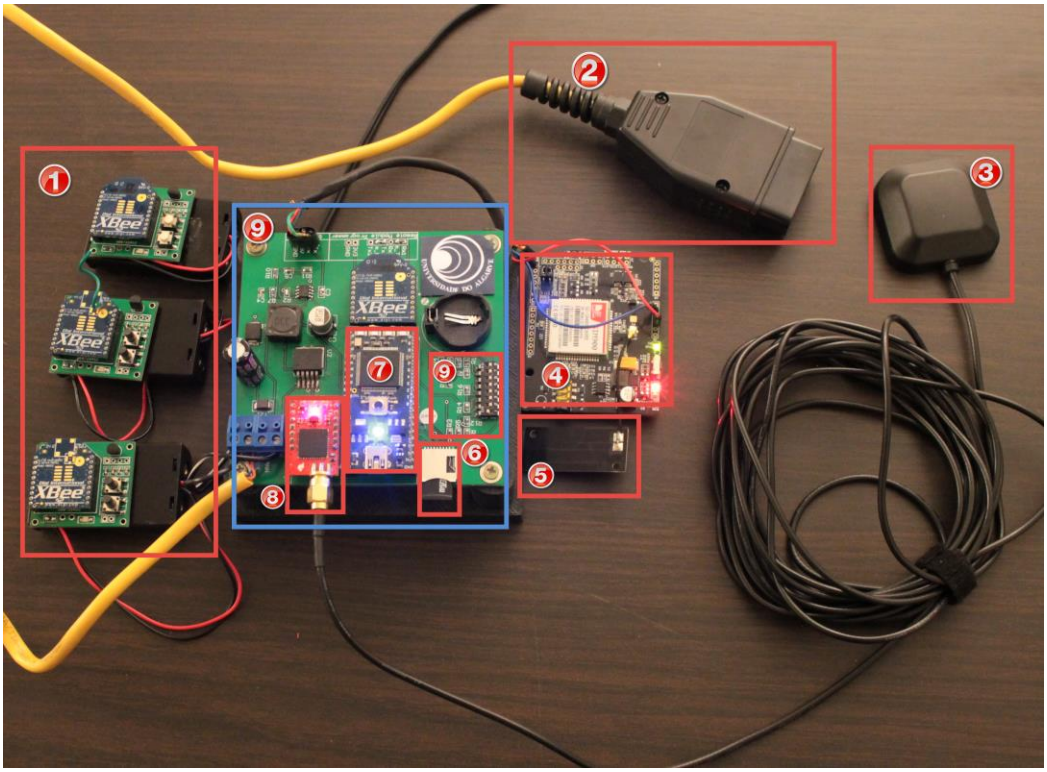


Figura 36 - Partes Integrantes do sistema

Para maior detalhe, na Figura 37 pode-se ver em pormenor o conector macho OBD-II SAE J1962.

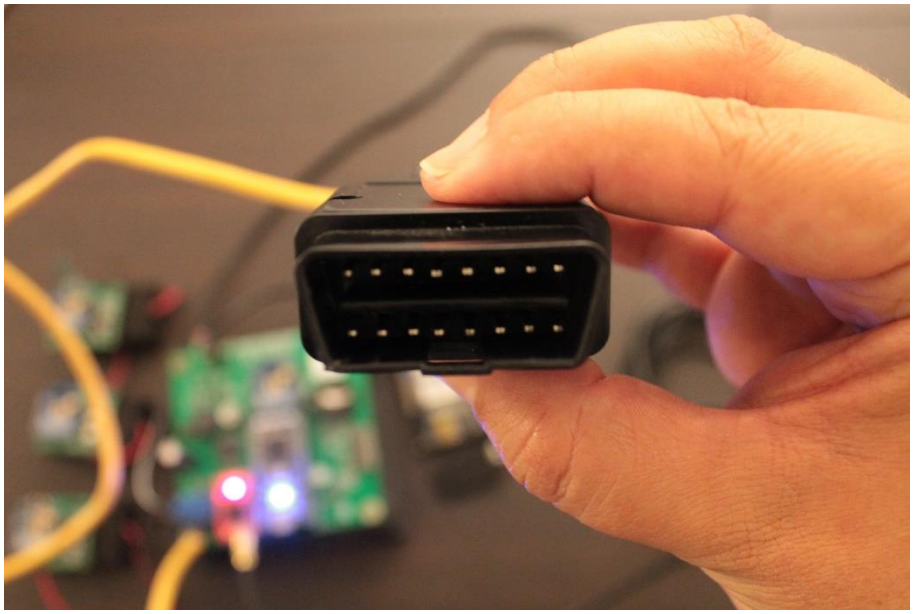


Figura 37 - Pormenor Conenctor OBD-II

Depois de montados, os módulos remotos foram ligados a duas pilhas LiFe Po4 em série, totalizando uma alimentação entre 5.4V a 6.4V, de acordo com as tensões mínima e máxima das mesmas (Figura 38, Figura 39 e Figura 40). Visto não possuírem LEDs de indicação e funcionamento para poupança das baterias, cada módulo remoto conta com um botão de pressão e um LED para verificar se existe carga nas baterias e, consequentemente, se existe alimentação no módulo (ver Figura 40).

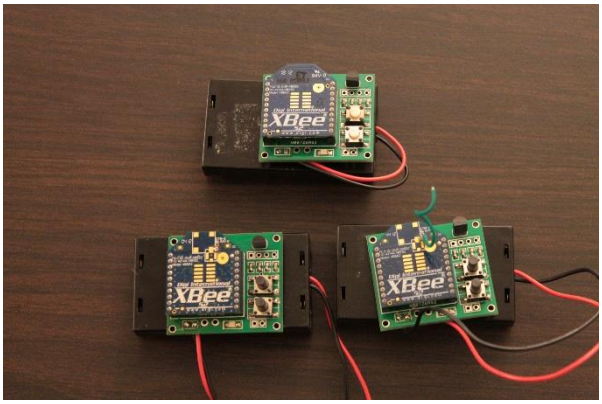


Figura 38 - Módulos Remotos

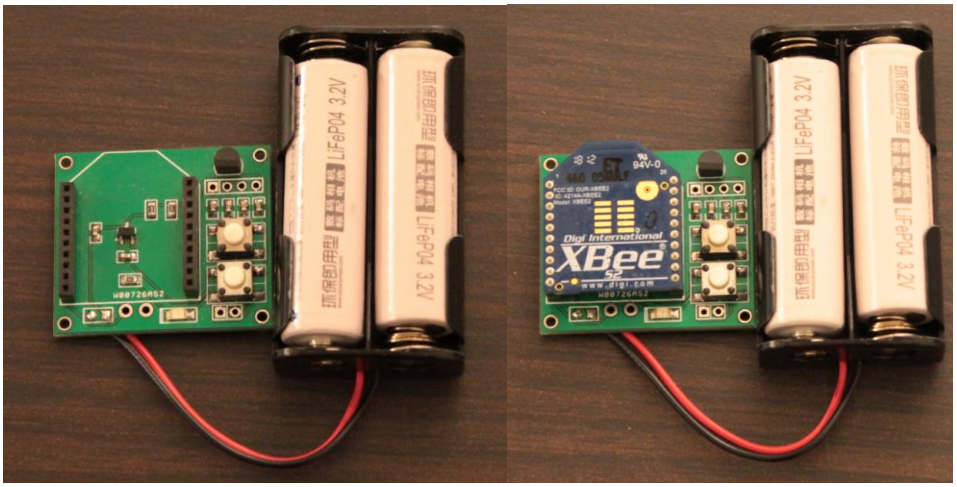


Figura 39 - Móduolo Remoto - PCB e Baterias LiFe Po4

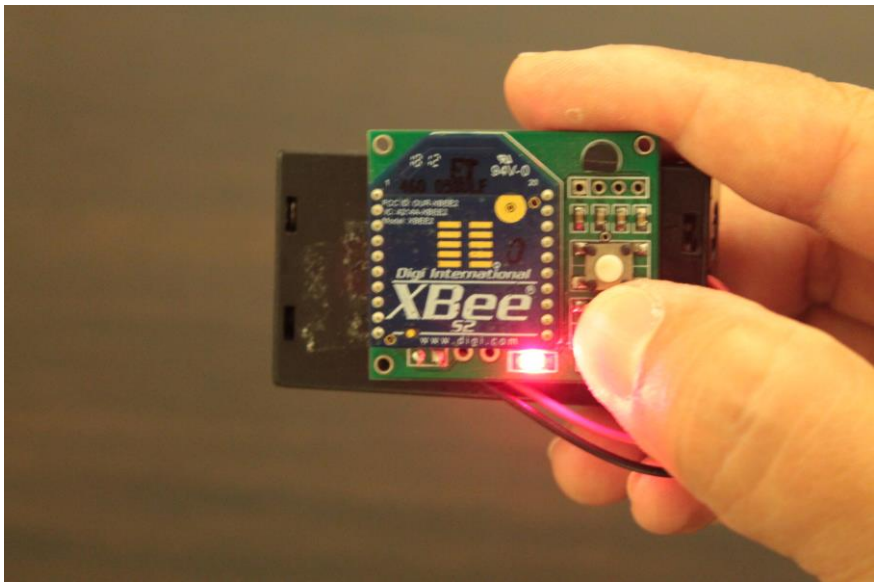


Figura 40 - Módulo Remoto - Teste de Bateria

6.1 COMUNICAÇÃO XBEE, TEMPERATURAS E TENSÃO DA BATERIA

Utilizando o programa XCTU, ligado via porta série ao XBee presente no módulo principal, pode-se verificar a comunicação entre este e o módulo remoto (Figura 41). Os módulos foram configurados de acordo com o procedimento descrito na secção 5.2.2, ou seja, o módulo remoto configurado como *end device* e a enviar, periodicamente, o valor lido no *ADC,* e o módulo principal "à escuta", configurado como coordenador.

Na Figura 41, encontra-se a comunicação série e pode-se visualizar o tipo de trama recebida pelo coordenador, bem como os intervalos de tempo entre as receções. Também se podem visualizar informações detalhadas de uma dada trama, selecionada na secção à esquerda. Por exemplo, pode-se confirmar tratar-se de uma trama do tipo *IO Data Sample Rx Indicator*, identificada pelo valor hexadecimal '92'. Outro campo relevante será o *Analog Sample*, que indica o valor da leitura nos *ADC* ativos.

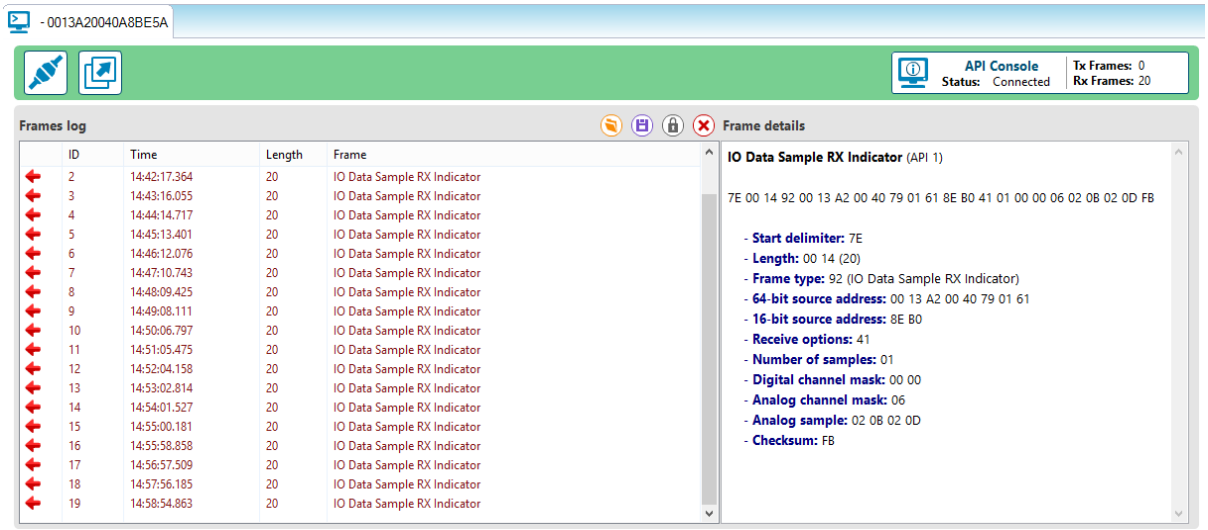


Figura 41 - Monitor Comunicação Série API

É preciso notar que no terminal série os intervalos das amostragens não correspondem a um minuto exato. Isso deve-se maioritariamente ao tempo de processamento do próprio monitor série anfitrião, bem como no intervalo de tempo despendido pelo módulo coordenador para amostrar na sua porta série o que recebeu por comunicação sem fios.

De modo a demonstrar e testar o funcionamento dos constituintes do sistema, numa fase inicial foram criadas rotinas de teste com *output* para a porta série dos dados desejados. Através da ligação USB do *mbed*, foi possível visualizar os dados num terminal série no computador. Foram ativados dois dos quatro *ADC* disponíveis no módulo remoto, um ligado ao sensor de temperatura e o outro ao circuito monitor de tensão da bateria. A Figura 42 mostra a comunicação dos valores lidos pelos *ADC* do módulo remoto, bem como a conversão dos valores para as grandezas desejadas, no terminal.

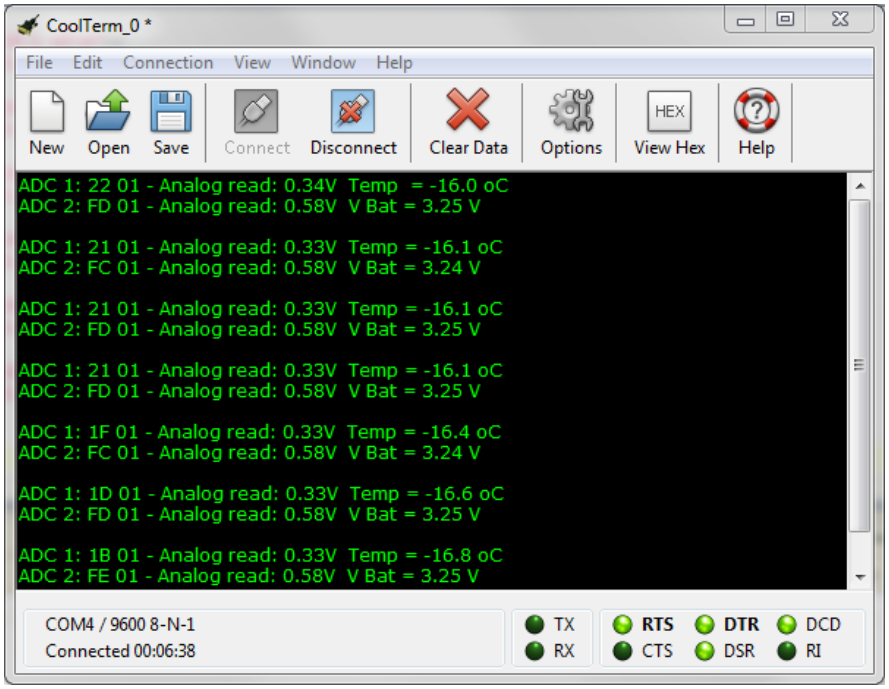


Figura 42 - Comunicação de Temperatura e Tensão na Bateria

É de referir que, de forma a testar o funcionamento dos módulos remotos num ambiente refrigerado, foi instalado um módulo numa câmara de congelação doméstica. O gráfico apresentado na Figura 43 mostra a variação da temperatura, ao longo de um período de 5 horas. Por outro lado, a Figura 44 apresenta o gráfico visível na página do servidor para um período menor e posterior à leitura apresentada na Figura 43.

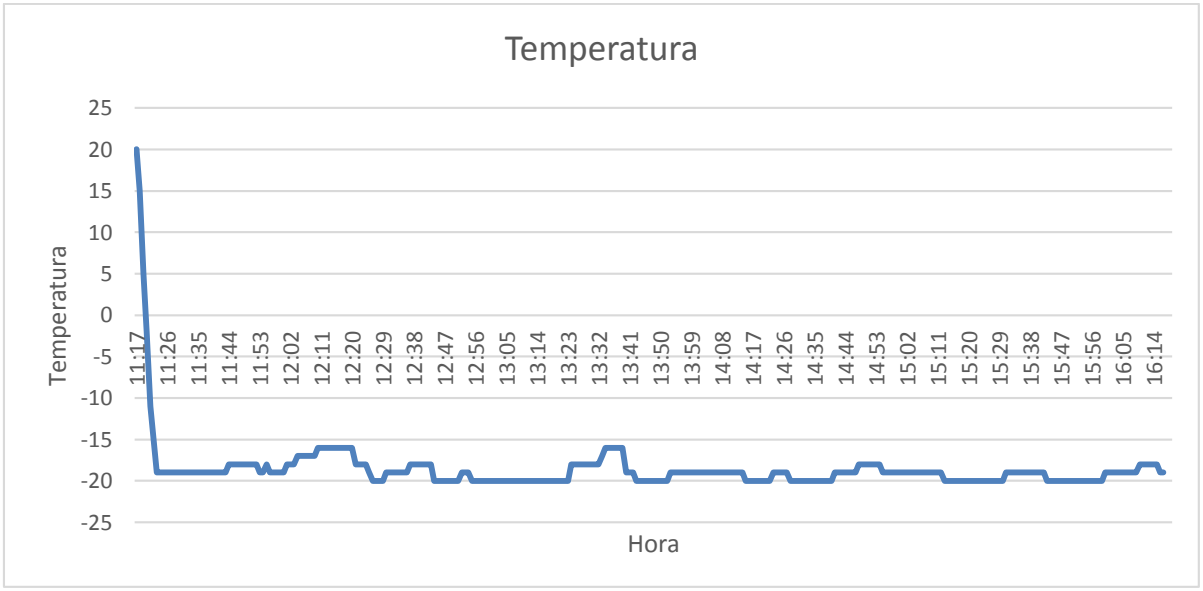


Figura 43 - Amostragem de Temperatura

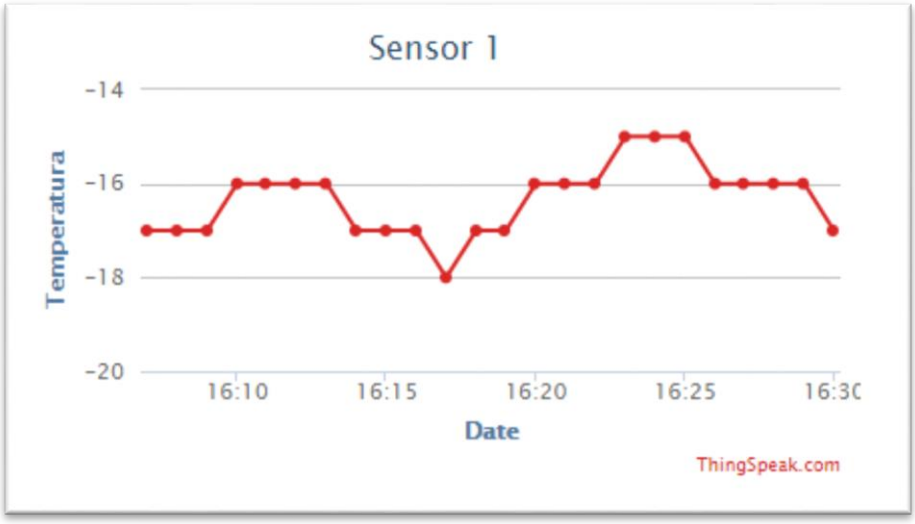


Figura 44 – Amostragem de Temperaturas Online

A monitorização foi efetuada desde o momento da introdução do sensor na câmara, explicando a temperatura inicial da Figura 43, na ordem dos 20ºC. Pode-se reparar, que há um período de transição entre as leituras da temperatura ambiente até o sensor alcançar a temperatura interna da câmara. Deve-se à transferência térmica do corpo do sensor, que deve "arrefecer" ou "aquecer" até à sua temperatura ambiente. Assim sendo, somente após alguns minutos se passarem, as leituras alcançam valores na ordem esperada, entre -18ºC e -21ºC.

Pode-se reparar também picos mais significativos de subida de temperatura, na ordem dos 3ºC (em ambas as figuras). Estes picos deram-se por abertura momentânea da porta, de modo a testar a eficácia de variação do sensor nestes casos. Os picos menores devem-se ao ciclo de desligamento do motor de frio, devido ao termostato interno do aparelho e ao fato das temperaturas terem sido arredondadas a valores exatos neste gráfico.

A Figura 45 apresenta ainda o gráfico para a monitorização da tensão da bateria. Pode-se observar que o valor é sempre constante, uma vez que, neste caso, o período de observação foi pequeno e não foi possível observar uma descarga significativa da bateria (ou pelo menos que seja visível no gráfico).

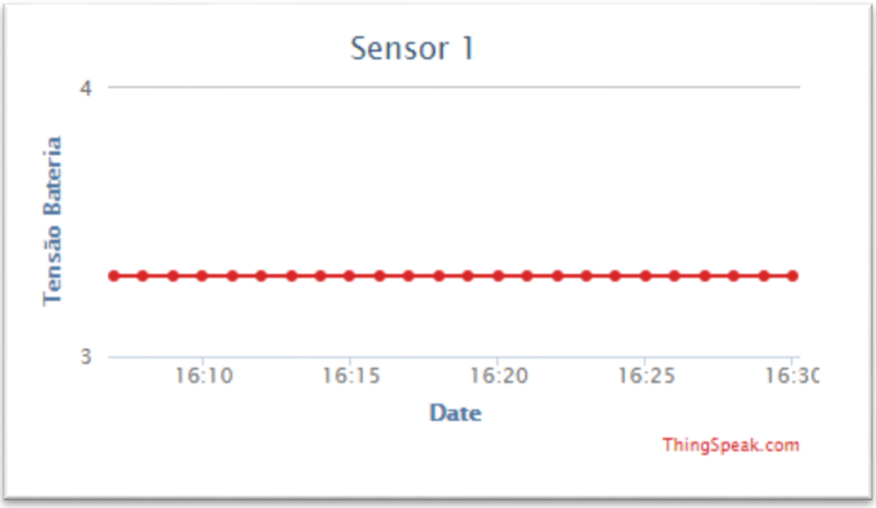


Figura 45 - Tensão na Bateria

6.2 COORDENADAS GPS

O módulo *GPS* envia, constantemente, tramas com informação para a porta série, como descrito na secção 5.2.3. De modo análogo ao teste da comunicação com o módulo *XBee* descrito anteriormente, também foi criada uma rotina para testar e demonstrar os valores recebidos do módulo GPS, bem como a conversão para coordenadas em graus, minutos e segundos (ver Figura 46).

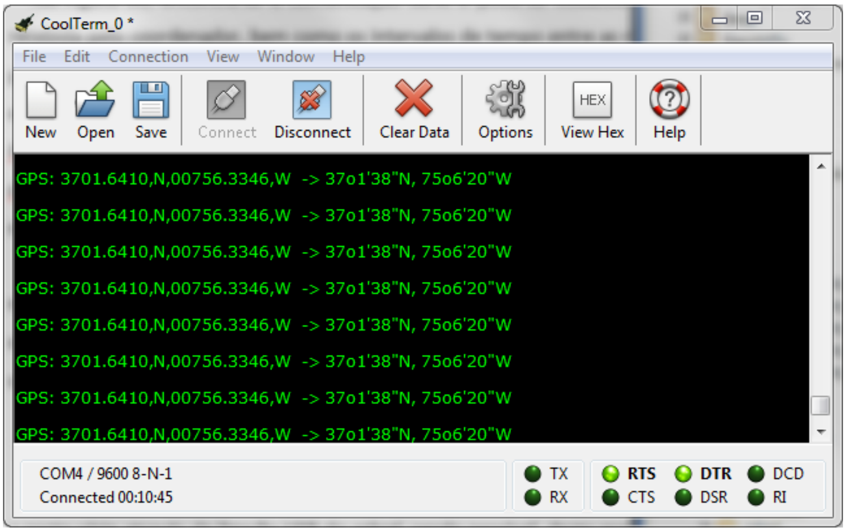


Figura 46 - Coordenadas GPS

De modo a testar a aplicação do GPS de uma forma mais próxima de uma aplicação real, foi realizado um trajeto em que foram recolhidos dados, que foram representados em gráfico utilizando a *API Google Maps*. Com esta *API* pode-se representar, geograficamente, as sucessivas coordenadas lidas num percurso de modo a facilitar interpretação. O resultado é o gráfico apresentado na Figura 47, que fica disponível automaticamente no servidor.

Figura 47 - Localização Geográfica

Este recurso é muito importante para a interface de utilizador, de modo a permitir a interpretação direta dos dados registados.

6.3 OBD-II – CONSUMOS

Para os testes à ligação OBD-II, foram lidos, periodicamente, os valores dos sensores de massa de fluxo de ar (*MAF*) e da velocidade do veículo, *PID's* 0x0D e 0x10, respetivamente, aplicando-se estes dados à equação 4, presente na secção 5.2.4. O objetivo final é o de obter o consumo instantâneo em litros por 100 quilómetros percorridos.

Como foi descrito na secção 5.2.4, tendo sido utilizada uma viatura a gasolina, foi tido em conta este tipo de combustível para se encontrar a constante utilizada na equação (4).

A Figura 48 apresenta um excerto das leituras periódicas dos valores dos referidos *PID*, recebidas no terminal através de ligação série, bem como o cálculo do consumo instantâneo. Estes dados são apresentados de forma gráfica no servidor, como é mostrado na Figura 49.

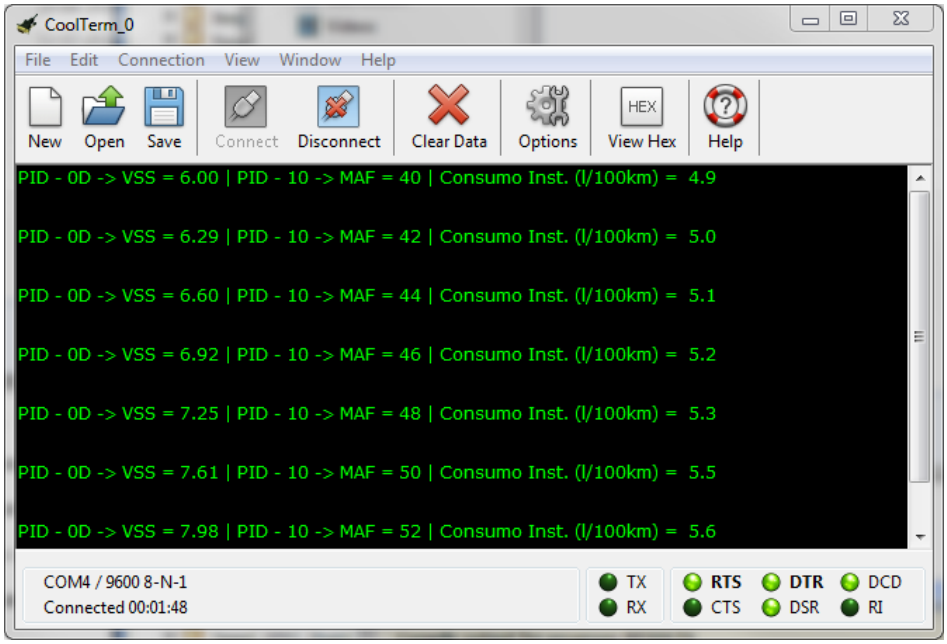


Figura 48 - OBD-II - Consumos Instantâneos

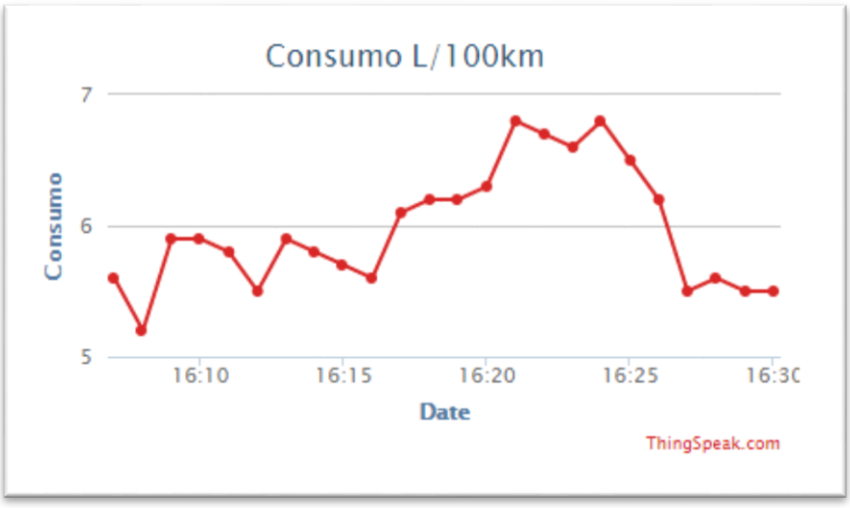


Figura 49- Consumos

6.4 INTERFACE E SERVIDOR

Como referido na secção 5.3, foi desenvolvido uma interface para o servidor que pode ser consultado no endereço http://i3fr.goblineering.com/. A página de apresentação de resultados desenvolvida reúne alguns gráficos já apresentados (Figura 44, Figura 45, Figura 47 e Figura 49) e pode ser vista abaixo na Figura 50.

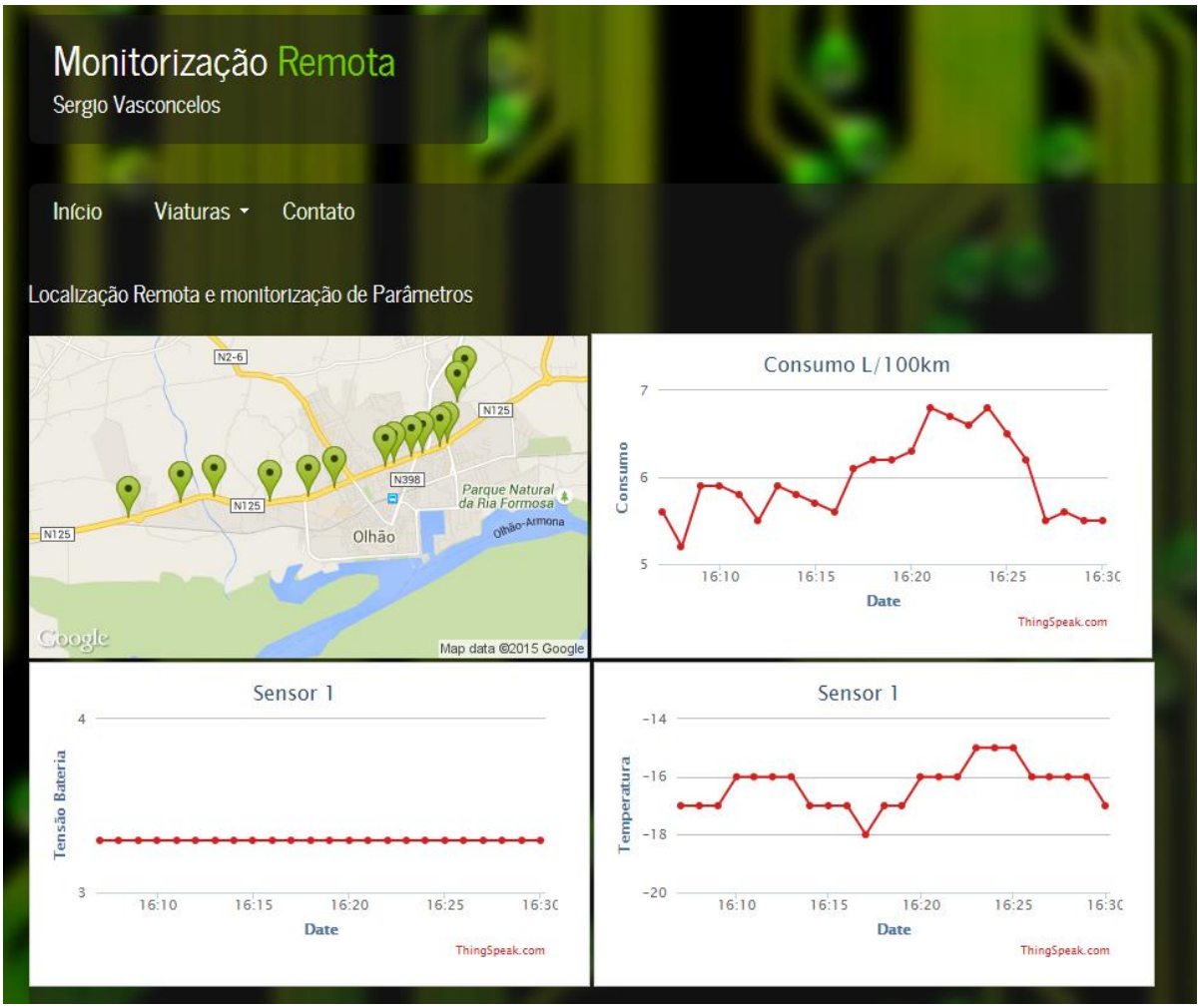


Figura 50 - *Web site* - interface de resultados

7. CONCLUSÕES

Este trabalho visou o desenvolvimento de *hardware* e *firmware* destinados à instalação em viaturas de entrega de comida fresca e congelada, com o objetivo de efetuar a monitorização remota de parâmetros ambientais, como temperatura e humidade, adquiridos por sensores, dentro e fora das câmaras frigoríficas. Foi também visada a implementação de outras funcionalidades, bem como a aquisição da posição geográfica da viatura através de coordenadas GPS, a obtenção de dados mecânicos, do veículo, obtidos via OBD e, também, a comunicação em tempo real via GSM/GPRS.

7.1 ANÁLISE DO TRABALHO EFETUADO

Sendo o principal objetivo deste trabalho desenvolver módulos capazes de comunicar a temperatura local através de comunicação sem fios por radiofrequência, com o propósito final de complementar um sistema de otimização de rotas, o trabalho inicial passou por estudar e identificar os sensores e módulos RF mais indicados a esta aplicação.

Todos os sensores apresentam vantagens e desvantagens, como ilustrado anteriormente. Inicialmente escolheu-se o sensor de temperatura LM35, por ser um padrão de sensor fabricado por diversos fabricantes, ter baixo custo e ser largamente utlizado. Porém, constatou-se que para atingir leituras negativas, seria preciso um referencial de tensão negativo, o que implicaria acrescentar *hardware*, logo, aumentar o custo. Assim, o LM35 foi, posteriormente, substituído pelo MCP9700.

A escolha do termístor em circuito integrado, MCP9700, passou pelo seu custo extremamente baixo e facilidade de implementação. Uma das desvantagens é o tempo que o seu invólucro leva até atingir a temperatura ambiente. Porém, como as câmaras frigoríficas são isoladas e arrefecem/aquecem gradativamente, e as leituras de temperaturas estarem espaçadas em minutos, este fator não afeta o sistema negativamente. Como pontos negativos observados durante os testes, notou-se

discrepância de até ±1ºC em sensores idênticos a enviar leituras do mesmo local. Para além disso, tendo sido o sistema baseado na leitura de sensores analógicos, a substituição ou adição de sensores seria um processo rápido, tendo o sistema sido testado apenas com os sensores de temperatura, de modo a testar o conceito.

Para a comunicação com os sensores, a escolha aconteceu naturalmente, ou seja, à partida pensou-se em utilizar módulos com capacidades ZigBee, sendo este um protocolo dedicado maioritariamente a redes de sensores. Os módulos *XBee* são os módulos ZigBee mais difundidos no mercado, tendo uma boa estrutura de suporte e vasta gama de *firmwares*. Estes módulos existem em duas versões e a *Series 2* foi escolhida por apresentar mais possibilidades de mecanismos de rede, maior alcance e capacidade. Outro fator decisivo na escolha destes dispositivos foi a facilidade de acrescentar ou retirar módulos da rede. Foram obtidos os resultados esperados com estes módulos, revelando-se de fato de fácil implementação, abstraindo a aplicação principal de parte da gestão da rede de sensores.

Pode-se dizer que uma desvantagem destes módulos é o seu custo, porém esta desvantagem pode ser colmatada quando comparados com outros módulos RF, tendo em vista as suas funcionalidades.

O módulo GPS, *Venus* da *SkyTraq,* selecionado não apresenta características marcantes, senão possuir comunicação série e a ação de funcionar no modo pretendido, ou seja, constantemente adquirir e enviar através desta porta os dados relativos às coordenadas.

A seleção do módulo GSM *SIM900* da *SMCOM* é um módulo largamente utilizado atualmente. Tal deve-se ao seu custo, inferior à maioria dos módulos de fabricantes com mais expressão, como a *Siemens*. Possuindo também comunicação série, a sua utilização demonstrou-se pouco complexa, tendo-se obtido o resultado esperado, ou seja, foi possível efetuar a ligação à internet, posteriormente enviando as informações para um servidor.

É de referir que foi ainda testada e implementada a ligação *Wi-Fi*, mas no âmbito deste trabalho esta apenas está presente como periférico opcional, eventualmente para ligação dos módulos ao servidor quando as viaturas se encontram no parque da empresa (para

poupar os custos da ligação GSM/GPRS). No entanto esta solução não faz parte da implementação final, sendo porém facilmente acrescentada em caso de necessidade.

A parte de ligação ao *on-board diagnostics* da viatura foi implementado em *hardware*, estando presente no módulo final. Porém, esta funcionalidade não foi otimizada, tendo sido utilizada uma biblioteca específica do *mbed* para este fim que implementou as funções necessárias.

Depois de efetuado um estudo de mercado, constatou-se que a monitorização remota e *tracking* de viaturas é uma área em franca expansão a nível mundial, sendo esperado um grande crescimento na demanda destes produtos a médio e curto prazo. Existem, cada vez mais, soluções voltadas para "Internet of Things", tendência esta que este trabalho também segue, tratando-se de aparelhos que, com algum tipo de conectividade com ou sem fios, enviam as suas informações através da internet, possibilitando a troca de informação e mesmo o controlo remoto destes.

Analisadas várias soluções para monitorização de viaturas, verificou-se que existem diversos sistemas de monitorização de temperaturas e outros dados sensoriais e, por sua vez, existem também diversas soluções de *tracking*, algumas delas propostas por empresas, líderes no mercado, de aparelhos de navegação pessoais. Porém, pode afirmar-se que não existe, no mercado, oferta de uma solução integrada como a que este trabalho oferece, ou seja, a monitorização remota de temperaturas, dados do veículo e *tracking* num só equipamento.

Outra característica diferenciadora desta solução é a utilização de sensores sem fios dentro das câmaras frigoríficas, alimentados por baterias de alto desempenho com longevidades esperadas de até dois anos, o que facilita a instalação e manutenção e diminui os custos.

É então possível gerar um conjunto completo e realista de dados inerentes aos veículos de entrega, que pretende oferecer grande apoio quando aplicado ao sistema de gestão e otimização de rotas, instalado na central. Além da otimização, será possível uma constante monitorização, em tempo real, das viaturas, sendo possível detetar avarias em fase inicial, possibilitando assim agir e evitar, por exemplo, a degradação da carga. Para além disso pode mesmo fazer alterar comportamentos no que toca ao seguimento das rotas indicadas,

bem como fornece um maior controlo sobre o consumo e as variações de temperatura na câmara, detetando possíveis situações de mau uso (por exemplo abertura excessiva de portas).

7.2 DIREÇÕES DE TRABALHO FUTURO

As primeiras propostas de trabalho futuro passam por completar as funcionalidades e testes aqui descritos mas não finalizadas. Um dos casos é relativo à comunicação GPRS, que não foi testada em tempo real e numa viatura em estrada, mas testada em isolado e não englobada no programa principal testado em viaturas. Outro caso é o da ligação ao OBD-II, também implementada mas não testada exaustivamente de forma a testá-la em diversas marcas de viaturas e explorar a obtenção de diferentes tipos de informação.

Além do trabalho não finalizado, um fator importante seria a adição de um monitor *touchscreen*, de modo a acrescentar a funcionalidade de navegação GPS e possibilitando a interação em tempo real do condutor com o sistema de otimização de rotas, recebendo alterações à mesma ou comunicando eventos a central (como acidentes e outras situações de interesse). Assim, seria possível substituir o módulo do outro fabricante citado no início deste trabalho.

Outro fator importante seria redesenhar o circuito, baseando o mesmo em um microcontrolador *stand alone* em vez de uma plataforma de prototipagem, como é o caso do *mbed*. Da mesma forma, também substituir as placas dos módulos GPS e GPRS pelos módulos apenas, tendo também este fator em conta no projeto da nova *PCB* de forma a reduzir consideravelmente o custo do sistema final e facilitar a comercialização da solução.

REFERÊNCIAS

[1] IEEE Standards Association, “IEEE 802.3: Ethernet.” [Online]. Available: http://standards.ieee.org/about/get/802/802.3.html. [Accessed: 13-Sep-2014].

[2] N. Labs, “Nest Thermostat.” [Online]. Available: https://nest.com/thermostat. [Accessed: 11-Sep-2014].

[3] IEEE Standards Association, “IEEE 802.11:Wireless LANs.” [Online]. Available: http://standards.ieee.org/about/get/802/802.11.html. [Accessed: 13-Sep-2014].

[4] Omega, “Introduction to Remote Monitoring.” [Online]. Available: http://www.omega.com/prodinfo/remotemonitoring.html. [Accessed: 13-Sep-2014].

[5] Marketsandmarkets.com, “Fleet Management Market (Fleet Analytics, Vehicle Tracking & Fleet Monitoring, Telematics, Vendor Services) By Vehicles (Trucks, Light Goods, Buses, Corporate Fleets, Container Ships, Aircrafts) Worldwide Market Forecasts and Analysis (2013 – 2018),” 2013.

[6] TomTom, “TomTom Telematics.” [Online]. Available: https://business.tomtom.com/pt_pt/products/webfleet/highlights/. [Accessed: 13-Sep-2014].

[7] “mbed.” [Online]. Available: https://mbed.org. [Accessed: 14-Sep-2014].

[8] Arm, “Cortex-M3 Processors.” [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-m3.php. [Accessed: 14-Sep-2014].

[9] Omega, “Thermocouples Introduction and Theory.”

[10] Omega, “The Thermistor.” [Online]. Available: https://www.princeton.edu/~cavalab/tutorials/public/Thermocouples.pdf. [Accessed: 14-Sep-2014].

[11] Agilent Technologies, “Practical temperature measurement,” 2001.

[12] “IEEE 802.15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).” [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf. [Accessed: 14-Sep-2014].

[13] G. Heine, *GSM Networks: Protocols, Terminology, and Implementation*. Boston-London, 1999.

[14] M. Sauter, *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley and Sons, 2011.

[15] F. G. Snider, “GPS: Theory, Pratice and Applications,” 2012.

[16] “A Flash Storage Technical and Economic Primer.” [Online]. Available: http://www.flashstorage.com/flash-storage-technical-economic-primer/. [Accessed: 10-Sep-2015].

[17] “Understanding FAT32 Silesystems.” [Online]. Available: http://www.pjrc.com/tech/8051/ide/fat32.html. [Accessed: 13-Sep-2013].

[18] “Description of the FAT32 Filesystem.” [Online]. Available: https://support.microsoft.com/en-us/kb/154997. [Accessed: 13-Sep-2015].

[19] “mbed SD Filesystem Documentaton.” [Online]. Available: https://developer.mbed.org/cookbook/SD-Card-File-System. [Accessed: 13-Sep-2015].

[20] “Monitoring Fuel Comsumption on Your Vehicle in Real-Time.” .

[21] “Aqua-calc - Conversions and Rates.” .

[22] “Sooperfish: jQuery plugin for ulti dropdown menus.” [Online]. Available: https://www.sooperthemes.com/open-source/jquery/jquery-sooperfish-dropdown-menus.

[23] I. Will, “Remote Weather Sensors Over ZigBee,” pp. 1–17, 2011.

[24] “XBee Buying Guide.” [Online]. Available: https://www.sparkfun.com/pages/xbee_guide. [Accessed: 03-Feb-2015].

[25] “Sparkfun XBee Explorer.” [Online]. Available: https://www.sparkfun.com/products/11812. [Accessed: 09-Feb-2014].

[26] “Data Sheet do microcontrolador LPC1768.” [Online]. Available: http://www.nxp.com/products/microcontrollers/product_series/lpc1700/. [Accessed: 08-Feb-2014].

[27] “Cortex M3 Devices - Generic USer Guide,” 2010. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.dui0552a/DUI0552A_cortex_m3_dgug.pdf. [Accessed: 06-Feb-2014].

[28] “ARM Cortex-M3 Technical Reference Manual,” 2006. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf. [Accessed: 05-Feb-2014].

[29] “Sparkfun Venus GPS Product Page.” [Online]. Available: https://www.sparkfun.com/products/11058. [Accessed: 05-Feb-2015].

[30] “Venus638FLPx GPS Receiver Data Sheet,” 2010. [Online]. Available: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/Venus/638/doc/Venus638FLPx_DS_v07.pdf.

[31] “Seeed Studio GPRS Shield V2.0 Documentation.” [Online]. Available: http://www.seeedstudio.com/wiki/GPRS_Shield_V2.0. [Accessed: 08-Feb-2014].

[32] “SimCOM SIM900 Data Sheet.” [Online]. Available: http://wm.sim.com/producten.aspx?id=1019. [Accessed: 09-Feb-2014].

[33] “MCP2551 - Controller Area Network (CAN) Interface Data Sheet.” [Online]. Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010405. [Accessed: 09-Feb-2014].

[34] “mbed Cookbook.” [Online]. Available: http://developer.mbed.org/cookbook. [Accessed: 03-Feb-2014].

[35] “Microsoft Extensible Firmware Initiative - FAT32 Sile System Specification.” [Online]. Available: https://staff.washington.edu/dittrich/misc/fatgen103.pdf. [Accessed: 09-Feb-2014].

[36] “Altium Designer Technical Documentation.” [Online]. Available: http://techdocs.altium.com/. [Accessed: 06-Feb-2014].

[37] “XBee S2 Quick Reference.” [Online]. Available: http://www.tunnelsup.com/xbee-guide. [Accessed: 02-Feb-2014].

[38] “E/E Diagnostic Test Modes - Equivalent to ISO/DIS 15031-5:April 30, 2002 - SAE J1979 Standard,” 2002.

[39] “XBee Command Reference Table.” [Online]. Available: http://examples.digi.com/wp-content/uploads/2012/07/XBee_ZB_ZigBee_AT_Commands.pdf. [Accessed: 06-Feb-2014].

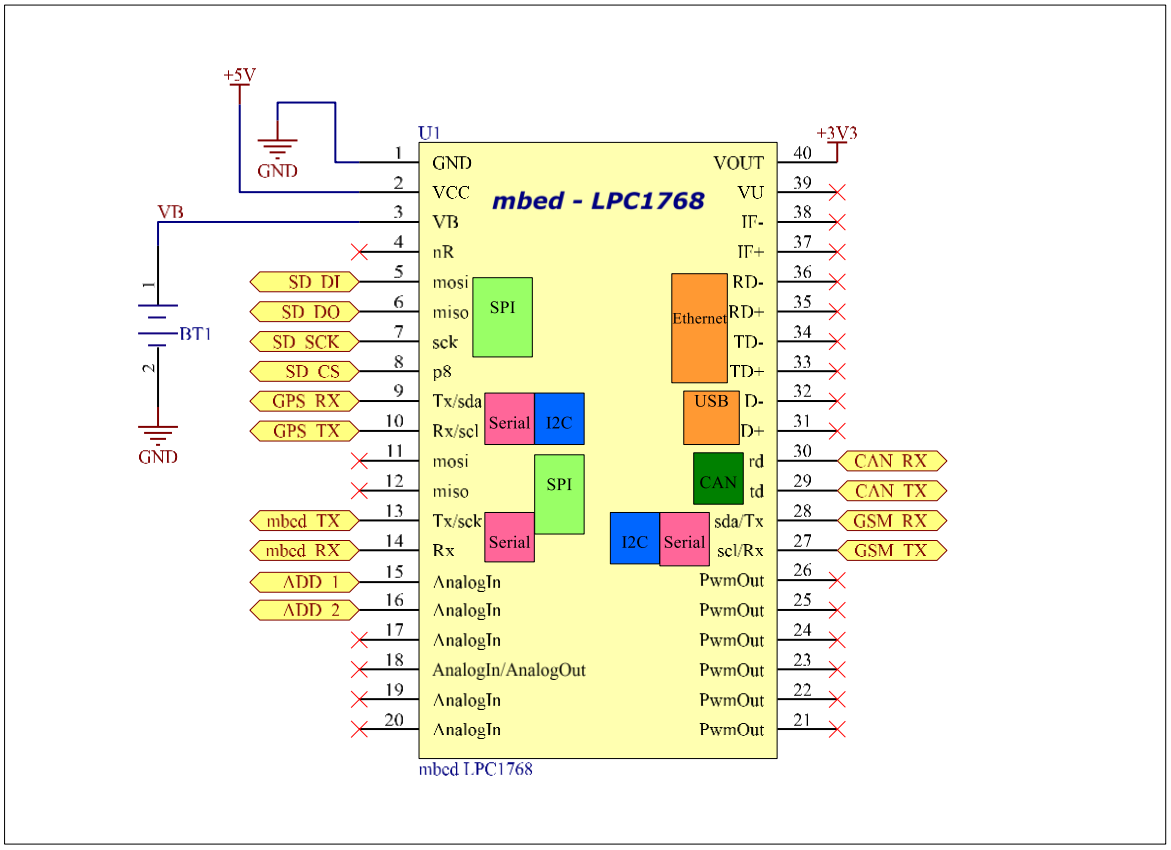
[40] R. Faludi, *Building Wireless Sensor Networks*. Sebastopol: O’Reilly Media Inc., 2011.

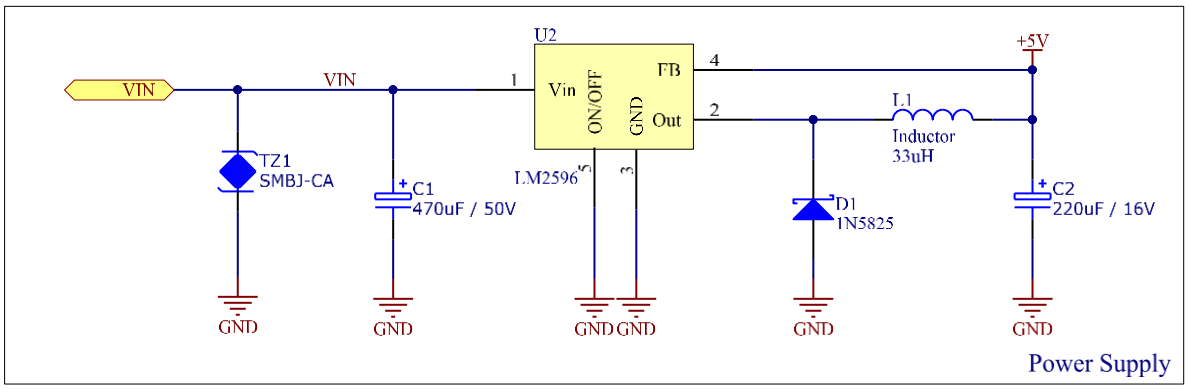
APÊNDICES

- Apêndice A – *Hardware* – Esquemáticos
- Apêndice B – *Firmware* – Código
- Apêndice C – Interface – Código

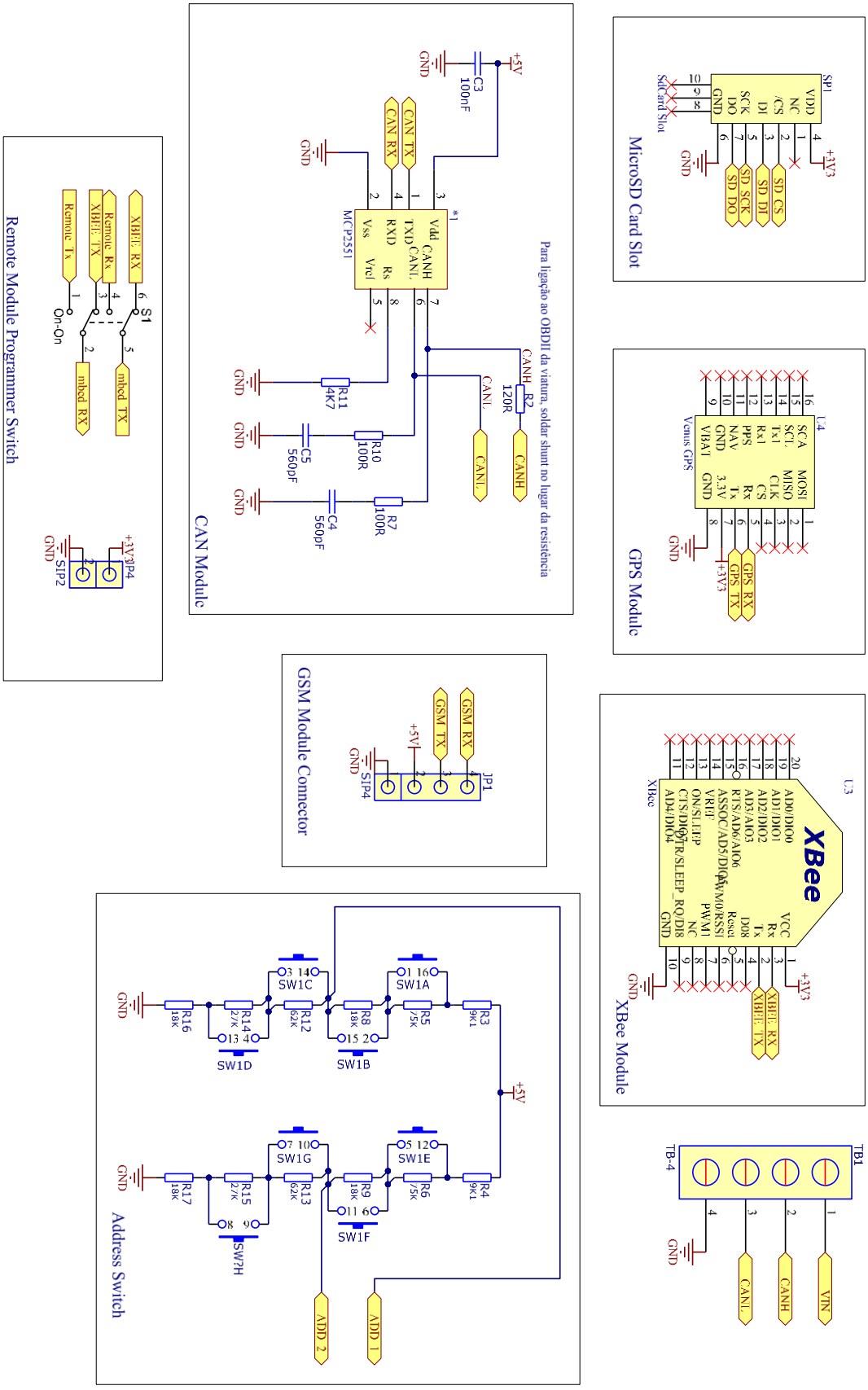
APÊNDICE A – HARDWARE – ESQUEMÁTICOS

MÓDULO PRINCIPAL

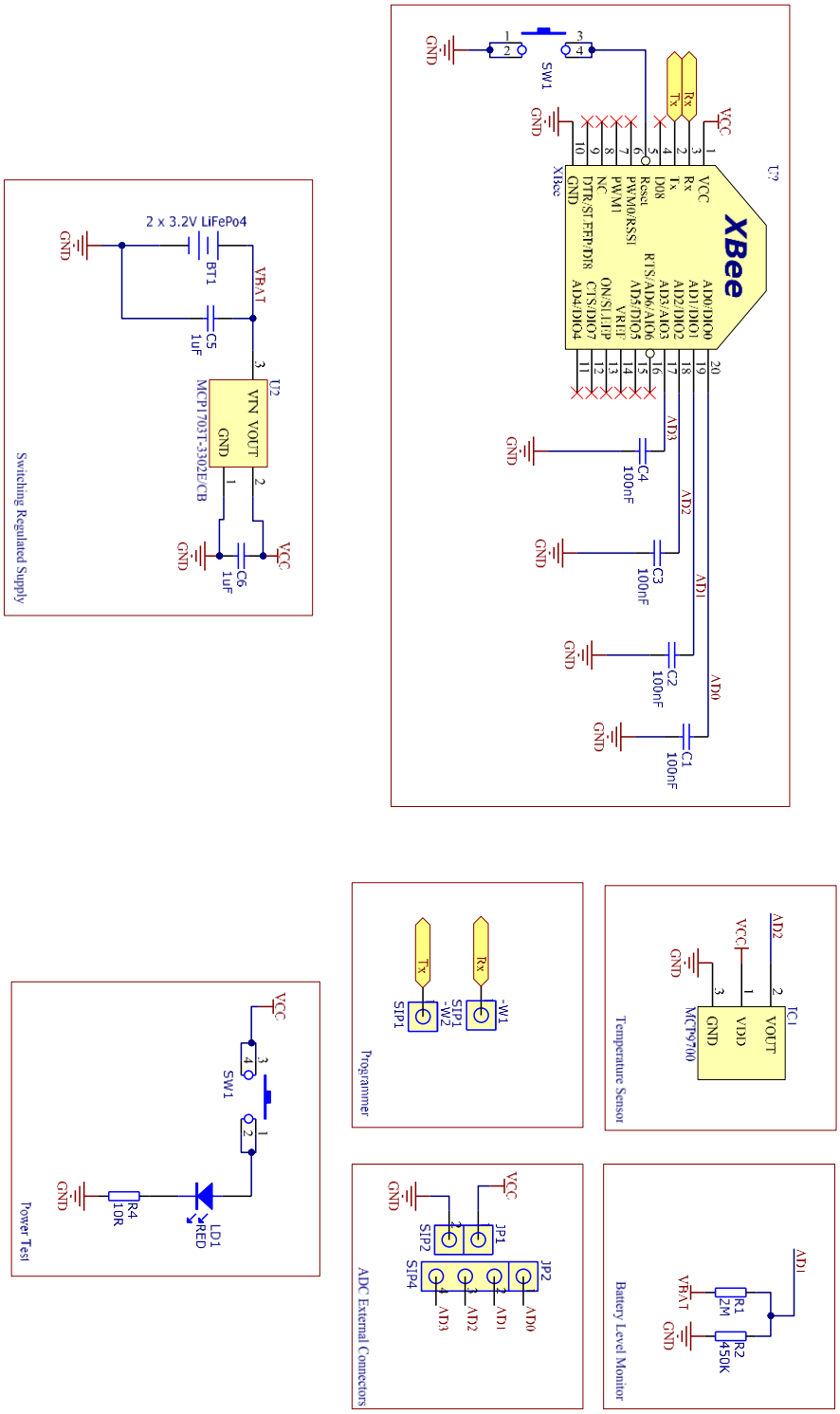




PERIFÉRICOS



MÓDULOS REMOTOS



APÊNDICE B – FIRMWARE - CÓDIGO

MAIN.CPP

```
#include "mbed.h"
#include "string.h"
#include "hardware_conf.h"
#include "ecu_reader.h"
#include "SDFileSystem.h"

#define DEBUG //If commented, debug ines will not be sent into the console

/*
Create ecu_reader objects
for different CAN bus speeds
*/
ecu_reader obdii1(CANSPEED_125);
ecu_reader obdii2(CANSPEED_250);
ecu_reader obdii3(CANSPEED_500);

SDFileSystem sd(p5, p6, p7, p8, "sd"); // the pinout on the mbed Cool Components workshop board

Timer timeCnt;

char *buffer_obd;
float value_obd;


/*****************************************************************
* Functions to set and show the Real Time Clock from and to the  *
```

```
* terminal                                *
**********************************************************************/
void set_time()
{
   //get the current time from the terminal
   struct tm t;

   pc.printf("Inserir data e hora:\n");
   pc.printf("AAAA MM DD HH MM SS[enter]\n");

   pc.scanf("%d %d %d %d %d %d", &t.tm_year, &t.tm_mon, &t.tm_mday, &t.tm_hour,
&t.tm_min, &t.tm_sec);

   //adjust for tm structure required values
   t.tm_year = t.tm_year - 1900;
   t.tm_mon = t.tm_mon - 1;

   set_time(mktime(&t));
}
void show_time()
{
   time_t seconds = time(NULL);
   printf("\nTime set => %s\n", ctime(&seconds));
}


/**********************************************************************
************
* Function to read the received XBee "IO Data Sample Rx Indicator" (frame type 0x92), *
* get the Address and ADC's readings and discard the unwanted data from it          *
**********************************************************************
*************/
```

```
int *XbeeRead()
{
  //Reads the intended values from the XBee received frame, discarding the unwanted data
  //18 bytes =>  Length - 2 Bytes | 64 Bit Sender Address - 8 Bytes | ADC's readings - 8 Bytes
  int frame[18] = { 0 }, i, msb, lsb, length;

  XBee_timeout.reset();
  XBee_timeout.start();

  while (xbee.getc()!=126  || XBee_timeout.read() > 60 ); //wait for the start character 0x7E = 126 in decimal

  if(XBee_timeout.read() > 60) {
    return(0);
  }
  //Get and calculate the frame length in order to know how many samples to consider
  msb = xbee.getc();
  lsb = xbee.getc();
  length = (256 * msb) + lsb;

  //Store the number of active ADC's on the first position of this function's returning frame
  frame[0] = (length - 16) / 2;

  //discard a byte - API frame type
  xbee.getc();

  // read the 64 bit address to the first 8 bytes of the output frame
  for(i=1; i<9; i++) {
    frame[i] = xbee.getc();
  }
```

```
  //discard 7 bytes - 16bit Address, Receive options, number of samples, digital mask and
analog mask
  for(i=0; i<7; i++) {
    xbee.getc();
  }

  //read the samples, according to the length, the disabled ADC will return '0'
  for(i=9; i<(9+(length-16)); i++) {
    frame[i] = xbee.getc();
  }

  return frame;

}

/*********************************************************************
************
* Function to convert the two byte read from the XBee ADC and convert to temperature  *
*     using the sensor formula, in this case, MCP9700                          *
* Inputs:                                              *
*      adc read byte values                                  *
* Outputs:                                              *
*      Adc read float value in ºC                              *
*                                                 *
*********************************************************************
*************/

float ADCtoTemp(int MSB, int LSB)
{
  float Temp, AnalogRead;
```

```
   // Hexadecimal values read from the ADC's converted to V
   AnalogRead = (LSB+(256.0*MSB))*(1.2/1024);

   // Parameters and formula from the MCP9700 Data Sheet
   float Vo=0.5, Tc=0.01;
   Temp = (AnalogRead-Vo)/Tc;

#ifdef DEBUG //Show the Analog Reading  and temperature values on the serial monitor
   pc.printf("%2.1f oC", Temp);
   pc.printf("Analog Read= %f\n",  AnalogRead);
#endif
   return Temp;
}

/**************************************************************************
************
* Function to convert the two byte read from the XBee ADC and convert to Volts       *
* Inputs:                                                 *
*      adc read byte values                                        *
* Outputs:                                                 *
*      Adc read float value in volts                                    *
*                                                 *
**************************************************************************
************/

float ADCtoV(int MSB, int LSB)
{
   float AnalogRead;

   // Hexadecimal values read from the ADC's converted to V
   AnalogRead = (LSB+(256.0*MSB))*(1.2/1024);
```

```
  return AnalogRead;
}

/*************************************************************************
************
* Function to read the GPS coordinates fom the Venus module              *
* Inputs:                                                                *
*     None                                                               *
* Outputs:                                                               *
*     Geografic coordinates formated as degree, minutes, seconds         *
*                                                                        *
*************************************************************************
*************/
char *GPSRead()
{
  char label[5], msg[100], *token;
  int comma;
  char *UTC, *Lat, *LatNS, *Lon, *LonEW, *quality;
  char format_LAT[15], Coord[21];

  while(1) {

    while(gps.getc()!='$'); //waits for the start of the line

    gps.scanf("%5s",&label); //stores the 5 char label of the frame

    if(strcmp(label,"GPGGA")==0) {

      while (gps.getc()!=',');

      gps.scanf("%s",&msg);
```

token = strtok(msg, ","); //begins isolating fields of information delimited by a comma

comma = 0; //variable to store the number of the comma, in orther to find which part of the message is read

```
while (token != NULL) { //finds the other tokens
   comma++;
   // pc.printf ("%s\n",token);

   switch(comma) {
      case 1: //UTC time hhmmss.ss
         UTC=token;
         break;
      case 2: //Latitude ddmm.mmmm
         Lat=token;
         break;
      case 3: //Latitude hemisphere N or S
         LatNS=token;
         break;
      case 4: //Longitude dddmm.mmm
         Lon=token;
         break;
      case 5:  //Longitude hemisphere E or W
         LonEW=token;
         break;
      case 6: //GPS quality indicator
         quality=token;
         break;
   }
   token = strtok(NULL,",");
}
```

```
        //pc.printf("Latitude = %s%s | Longitude = %s%s\n",Lat,LatNS,Lon,LonEW);
        sprintf(Coord, "%s,%s,%s,%s",Lat,LatNS,Lon,LonEW);
        return Coord;
      }
    }
}

/**************************************************************************
************
* Function to convert GPS coordinates to Decimal                          *
* Inputs:                                                                 *
*   Latitude:   ddmm.mmmm                                                 *
*   N/S Idicator: 'N' or 'S'                                              *
*   Longitude: dddmm.mmmm                                                 *
*   E/W Indicator: 'E' or 'W'                                             *
* Outputs:                                                                *
*   Latitude: DD.DDD  (signed N = '+'; S = '-')                           *
*   Longitude: D.DDDDD (signed W = '+'; E = '-')                          *
**************************************************************************
*************/
void GPS_Convert(char* coord)
{
   char* pEnd;
   float f1, f2, f3, f4, Lat, Lon;
   char c1, c2, SigLat = '-', SigLon = '-';

   f1 = strtof (coord, &pEnd);
   f2 = strtof (pEnd+1, &pEnd);
   c1 = coord[10];
   f3 = strtof (pEnd+3, &pEnd);
   f4= strtof (pEnd+1, &pEnd);
   c2 = coord[23];
```

```
   Lat = f1 + f2/60;
   Lon = f3 + f4/60;
   if(c1 == 'N')
      SigLat = '+';
   if(c2 == 'E')
      SigLon = '-';

#ifdef DEBUG
   printf ("%.0f  %.3f  %c  %.0f  %.3f %c  \n", f1,f2,c1,f3,f4,c2);
   printf("%c%f,%c%f",SigLat, Lat, SigLon, Lon);
#endif

}

/*************************************************************************
************
* Function to read chosen data from vehicle ECU                          *
* Inputs:                                                 *
*   PID choice, defined on file ECU_READER.h                             *
* Outputs:                                                  *
*   Value from chosen PID read from car ECU                              *
*************************************************************************
************/
int Read_ECU(int PID_Choice)
{

   pc.printf("\n\rCAN-bus demo CANSPEED_500...\n\r");

   /* while (1) {
      if (obdii3.request(ENGINE_RPM, buffer_obd) == 1)  // Get engine rpm and display on
USB port
```

```
            pc.printf("%s\n\r", buffer_obd);
        else
            pc.printf("Engine Request failed\n\r");
        wait(1);
    }
    */
    switch(PID_Choice) {

        case 1:
            if(obdii3.request(VEHICLE_SPEED, buffer_obd) == 1) {
                value_obd = buffer_obd[0];
            } else
                pc.printf("Vehicle Speed failed\n\r");
        case 2:
            if(obdii3.request(MAF_SENSOR, buffer_obd) == 1) {
                value_obd = ((buffer_obd[0]*256)+buffer_obd[1]) / 100;
                return value_obd;

            } else
                pc.printf("Maf sensor failed\n\r");
    }
    return 0;

}

/*************************************************************************
************
* Functions to send the string to the server through GPRS                *
* Inputs:                                                                *
*   Formated data string                                                 *
* Outputs:                                                               *
*   None                                                                 *
```

```
**********************************************************************
*************/
int waitForResp(char *resp, int timeout)
{
  int len = strlen(resp);
  int sum=0;
  timeCnt.start();

  while(1) {
    if(gprs.readable()) {
      char c = gprs.getc();
      sum = (c==resp[sum]) ? sum+1 : 0;
      if(sum == len)break;
    }
    if(timeCnt.read() > timeout) {  // time out
      timeCnt.stop();
      timeCnt.reset();
      pc.printf("Error:time out");
      return -1;
    }
  }
  timeCnt.stop();            // stop timer
  timeCnt.reset();             // clear timer
  while(gprs.readable()) {    // display the other thing..
    char c = gprs.getc();
  }

  return 0;
}

int sendCmdAndWaitForResp(char *cmd, char *resp, int timeout)
{
```

```
    gprs.puts(cmd);
    return waitForResp(resp,timeout);
}

int callUp(char *number)
{
    if(0 != sendCmdAndWaitForResp("AT+COLP=1\r\n","OK",5)) {
        pc.printf("Error:COLP");
        return -1;
    }
    wait(1);
    gprs.printf("\r\nATD%s;\r\n",number);
    return 0;
}

void GPRS_Send(char *data_string)
{

    char *whole_string, *http_command;

    *http_command = 'AT+HTTPPARA=\"URL\",\"i3fr.goblineering.com/data.php?data=';

    snprintf(whole_string, sizeof whole_string, "%s%s", *http_command, *data_string);


    for(int i = 0; i < 5; i++) {
        wait(1);
        pc.printf("wait\n");
    }
    if(0 != sendCmdAndWaitForResp("ATE0\r\n","OK",5)) {
        pc.printf("Error:Echo");
    }
```

```
if(0 != sendCmdAndWaitForResp("AT+CGATT?\r\n","OK",5)) {
    pc.printf("Error:CGATT");
}
wait(2);
if(0 != sendCmdAndWaitForResp(whole_string,"OK",5)) {
    pc.printf("Error:CONTYPE");
}
wait(2);
if(0 != sendCmdAndWaitForResp("AT+SAPBR=3,2,\"APN\",\"internet\"\r\n","OK",5)) {
    pc.printf("Error:SAPBR=3,1");
}
wait(2);
if(0 != sendCmdAndWaitForResp("AT+SAPBR=1,2\r\n","OK",5)) {
    pc.printf("Error:SAPBR=1,1");
}
wait(2);
if(0 != sendCmdAndWaitForResp("AT+HTTPINIT\r\n","OK",5)) {
    pc.printf("Error:HTTPINIT");
}
wait(2);
if(0 != sendCmdAndWaitForResp("AT+HTTPPARA=\"CID\",1\r\n","OK",5)) {
    pc.printf("Error:HTTPPARA");
}
wait(2);
if(0 != sendCmdAndWaitForResp("mbedteste\r\n","OK",10)) {
    pc.printf("Error:GET");
}
wait(2);
if(0 != sendCmdAndWaitForResp("AT+HTTPACTION=0\r\n","OK",5)) {
    pc.printf("Error:HTTPACTION");
}
```

```
    wait(2);
    if(0 != sendCmdAndWaitForResp("AT+HTTPTERM\r\n","OK",5)) {
        pc.printf("Error:HTTPTERM");
    }
    wait(2);

}

/**************************************************************************
************
* Functions to write string to file on SD card                    *
* Inputs:                                                  *
*   Formated data string                                        *
* Outputs:                                                   *
*   None                                                     *
**************************************************************************
*************/
void FileWrite(char *FileBuffer)
{
    mkdir("/sd/temp", 0777);

    FILE *fp = fopen("/sd/temp/temp.txt", "a");
    if(fp == NULL) {
        pc.printf("Could not open file for write\n");
    }
    fprintf(fp, "%s\n", FileBuffer);
    wait(1);
    fclose(fp);
}

/**************************************************************************
*  Main function
```

```
**********************************************************************/

int main()
{
   pc.printf("Inicio...\n");

   int *frame, numADC, i, MSB, LSB, VSS, MAF;
   char *GPSCoord, Readings[100], *DateTime, *FileBuffer;
   float AnalogRead[4], Temp, VBat, consumption;

   while(1) {
      time_t seconds = time(NULL); //read the actual time and store on "seconds"

      strftime(DateTime, 32, "%y,%m,%d,%H,%M,%S", localtime(&seconds));

      frame = XbeeRead();
      numADC = frame[0];
      GPSCoord = GPSRead();

      //according to the number of active ADC's, calculate the temperatures
      //for(i=9; i<(9+(numADC*2)); i=i+2) {

      //Read Battery Voltage
      i=9;
      MSB = frame[i];
      LSB = frame[i+1];
      VBat = ADCtoV(MSB, LSB);

      //Read Temperature Sensor
      MSB = frame[i+2];
      LSB = frame[i+3];
      Temp=ADCtoTemp(MSB, LSB);
```

```
#ifdef DEBUG
    pc.printf("%s |",ctime(&seconds)); //prints timestamp

    if(XbeeRead() != 0) {
      pc.printf(" Endereco XBee -> "); //prints read XBee address
      for(i=1; i<9; i++) {
        pc.printf("%x ", *(frame+i));
      }
    } else {
      printf("ERROR: XBee Error - Timeout\n");
    }
#endif
    //Read and calculate instant comsumption in l/100km
    VSS = Read_ECU(3);
    MAF = Read_ECU(5);
    consumption = (MAF * 32.7) / VSS;

#ifdef DEBUG
    pc.printf(" | Bateria= %2.1f | Temperatura = %2.1f| GPS: %s\n", VBat, Temp,
GPSCoord);
#endif

    //Group and format date in order to store/send
    sprintf(FileBuffer, "%s,%d,%2.2f,%2.1f, %1.1f",DateTime, i, *(AnalogRead+i), Temp,
consumption); //genarate string with comma separated values to write to file

    //Write string to SD card
    FileWrite(FileBuffer);

    //Send string through GPRS to the server
    GPRS_Send(FileBuffer);
```

```
    }
}
```

HARDWARE_CONF.H

```
#ifndef HARDWARE_CCONF_H
#define HARDWARE_CCONF_H
#include "mbed.h"

extern Serial pc;
extern DigitalOut led1;
extern DigitalOut led2;
extern DigitalOut led3;
extern DigitalOut led4;

extern Timer XBee_timeout;
extern Timer GPS_timeout;

extern Serial xbee;
extern Serial gps;
extern Serial gprs;


// We use can on mbed pins 29(CAN_TXD) and 30(CAN_RXD).
extern CAN can2;
extern CANMessage can_MsgRx;
extern int PID020;
extern int PID2140;
extern int PID4160; //PID Support Masks


#endif
```

HARDWARE_CONF.CPP

```
#include "hardware_conf.h"
#include "ecu_reader.h"

//SDFileSystem sd(p5, p6, p7, p8, "sd"); //SD card reader

Serial xbee(p13, p14);      //Opens serial port on for XBee connected on pins p9 and p10
Serial gps(p9, p10);      //Opens serial port on for GPS connected on pins 27 and p28
Serial gprs(p28, p27);
Serial pc(USBTX, USBRX);    //Serial USB port for PC debugging purposes



// We use can on mbed pins 29(CAN_TXD) and 30(CAN_RXD).
CAN can2(p30, p29);
CANMessage can_MsgRx;
int PID020, PID2140, PID4160; //PID Support Masks

Timer XBee_timeout;
Timer GPS_timeout;
```

ECU_READER.H

```
#ifndef ECU_READER_H
#define ECU_READER_H

#define CANSPEED_125      125000       // CAN speed at 125 kbps
#define CANSPEED_250      250000       // CAN speed at 250 kbps
#define CANSPEED_500      500000       // CAN speed at 500 kbps

/* Details from http://en.wikipedia.org/wiki/OBD-II_PIDs */
#define PID_0_20          0x00    //PID 0 - 20 supported
#define PID_0_20_DESC             "PID 0x00 - 0x20 Supported"
#define STATUS_DTC        0x01    ///
#define STATUS_DTC_DESC           "Status since DTC Cleared"
#define FREEZE_DTC        0x02    ///
#define FREEZE_DTC_DESC           "Freeze Diagnostic Trouble Code"
#define FUEL_SYS_STATUS     0x03    ///
#define FUEL_SYS_STATUS_DESC       "Fuel System Status"
#define ENGINE_LOAD       0x04    //
#define ENGINE_LOAD_DESC          "Calculated Engine Load"
#define ENGINE_COOLANT_TEMP 0x05
#define ENGINE_COOLANT_TEMP_DESC    "Engine Coolant Temperature"
#define ST_FUEL_TRIM_1      0x06    ///
#define ST_FUEL_TRIM_1_DESC       "Short Term Fuel % Trim - Bank 1"
#define LT_FUEL_TRIM_1      0x07    ///
#define LT_FUEL_TRIM_1_DESC       "Long Term Fuel % Trim - Bank 1"
#define ST_FUEL_TRIM_2      0x08    ///
#define ST_FUEL_TRIM_2_DESC       "Short Term Fuel % Trim - Bank 2"
#define LT_FUEL_TRIM_2      0x09    ///
#define LT_FUEL_TRIM_2_DESC       "Long Term Fuel % Trim - Bank 2"
#define FUEL_PRESSURE       0x0A    //
#define FUEL_PRESSURE_DESC        "Fuel Pressure"
```

```
#define INTAKE_PRESSURE     0x0B    //
#define INTAKE_PRESSURE_DESC        "Intake Manifold Absolute Pressure"
#define ENGINE_RPM          0x0C
#define ENGINE_RPM_DESC             "Engine RPM"
#define VEHICLE_SPEED       0x0D
#define VEHICLE_SPEED_DESC          "Vehicle Speed"
#define TIMING_ADVANCE      0x0E    //
#define TIMING_ADVANCE_DESC         "Timing Advance"
#define INTAKE_TEMP         0x0F    //
#define INTAKE_TEMP_DESC            "Intake Air Temperature"
#define MAF_SENSOR          0x10
#define MAF_SENSOR_DESC             "MAF Sensor Air Flow Rate"
#define THROTTLE            0x11
#define THROTTLE_DESC               "Throttle Position"
#define COMMANDED_SEC_AIR   0x12    ///
#define COMMANDED_SEC_AIR_DESC      "Commanded Secondary Air Status"
#define O2_SENS_PRES        0x13    ///
#define O2_SENS_PRES_DESC           "Detected O2 Sensors"
#define O2_B1S1_VOLTAGE     0x14    ///
#define O2_B1S1_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 1 Sensor 1"
#define O2_B1S2_VOLTAGE     0x15    ///
#define O2_B1S2_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 1 Sensor 2"
#define O2_B1S3_VOLTAGE     0x16    ///
#define O2_B1S3_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 1 Sensor 3"
#define O2_B1S4_VOLTAGE     0x17    ///
#define O2_B1S4_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 1 Sensor 4"
#define O2_B2S1_VOLTAGE     0x18    ///
#define O2_B2S1_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 2 Sensor 1"
#define O2_B2S2_VOLTAGE     0x19    ///
#define O2_B2S2_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 2 Sensor 2"
#define O2_B2S3_VOLTAGE     0x1A    ///
#define O2_B2S3_VOLTAGE_DESC        "O2 Sensor Voltage - Bank 2 Sensor 3"
```

```
#define O2_B2S4_VOLTAGE    0x1B   ///
#define O2_B2S4_VOLTAGE_DESC       "O2 Sensor Voltage - Bank 2 Sensor 4"
#define OBDII_STANDARDS    0x1C   //List of OBDII Standars the car conforms to
#define OBDII_STANDARDS_DESC       "Supported OBDII Standards"
#define O2_SENS_PRES_ALT   0x1D   ///
#define O2_SENS_PRES_ALT_DESC      "Detected O2 Sensors - Alternate Grouping"
#define AUX_IN_STATUS      0x1E   ///
#define AUX_IN_STATUS_DESC         "Auxiliary Input Status"
#define ENGINE_RUNTIME     0x1F   //
#define ENGINE_RUNTIME_DESC        "Run Time Since Engine Started"
#define PID_21_40          0x20   //PID 21-40 supported
#define PID_21_40_DESC             "PID 0x21 - 0x40 Supported"
#define DIST_TRAVELED_MIL  0x21   ///
#define DIST_TRAVELED_MIL_DESC     "Distance Traveled with MIL On"
#define FUEL_RAIL_PRESSURE 0x22   //
#define FUEL_RAIL_PRESSURE_DESC    "Fuel Rail Pressure Relative to Manifold"
#define FUEL_RAIL_PRES_ALT 0x23   ///
#define FUEL_RAIL_PRES_ALT_DESC    "MPI/Diesel Fuel Rail Pressure"
#define O2S1_WR_LAMBDA_V   0x24   ///
#define O2S1_WR_LAMBDA_V_DESC      "O2 Sensor 1 Equivalence Ratio Voltage"
#define O2S2_WR_LAMBDA_V   0x25   ///
#define O2S2_WR_LAMBDA_V_DESC      "O2 Sensor 2 Equivalence Ratio Voltage"
#define O2S3_WR_LAMBDA_V   0x26   ///
#define O2S3_WR_LAMBDA_V_DESC      "O2 Sensor 3 Equivalence Ratio Voltage"
#define O2S4_WR_LAMBDA_V   0x27   ///
#define O2S4_WR_LAMBDA_V_DESC      "O2 Sensor 4 Equivalence Ratio Voltage"
#define O2S5_WR_LAMBDA_V   0x28   ///
#define O2S5_WR_LAMBDA_V_DESC      "O2 Sensor 5 Equivalence Ratio Voltage"
#define O2S6_WR_LAMBDA_V   0x29   ///
#define O2S6_WR_LAMBDA_V_DESC      "O2 Sensor 6 Equivalence Ratio Voltage"
#define O2S7_WR_LAMBDA_V   0x2A   ///
#define O2S7_WR_LAMBDA_V_DESC      "O2 Sensor 7 Equivalence Ratio Voltage"
```

```
#define O2S8_WR_LAMBDA_V    0x2B    ///
#define O2S8_WR_LAMBDA_V_DESC      "O2 Sensor 8 Equivalence Ratio Voltage"
#define COMMANDED_EGR      0x2C    //
#define COMMANDED_EGR_DESC         "Commanded EGR"
#define EGR_ERROR          0x2D    //
#define EGR_ERROR_DESC             "EGR Error"
#define COMMANDED_EVAP_P    0x2E    ///
#define COMMANDED_EVAP_P_DESC      "Commanded Evaporative Purge"
#define FUEL_LEVEL         0x2F    //
#define FUEL_LEVEL_DESC            "Fuel Level Input"
#define WARMUPS_SINCE_CLR   0x30    ///
#define WARMUPS_SINCE_CLR_DESC     "Number of Warmups since DTC Cleared"
#define DIST_SINCE_CLR     0x31    ///
#define DIST_SINCE_CLR_DESC        "Distance Traveled Since DTC Cleared"
#define EVAP_PRESSURE      0x32    //
#define EVAP_PRESSURE_DESC         "Evap. System Vapor Pressure"
#define BAROMETRIC_PRESSURE 0x33    //
#define BAROMETRIC_PRESSURE_DESC    "Barometric Pressure"
#define O2S1_WR_LAMBDA_I    0x34    ///
#define O2S1_WR_LAMBDA_I_DESC      "O2 Sensor 1 Equivalence Ratio Current"
#define O2S2_WR_LAMBDA_I    0x35    ///
#define O2S2_WR_LAMBDA_I_DESC      "O2 Sensor 2 Equivalence Ratio Current"
#define O2S3_WR_LAMBDA_I    0x36    ///
#define O2S3_WR_LAMBDA_I_DESC      "O2 Sensor 3 Equivalence Ratio Current"
#define O2S4_WR_LAMBDA_I    0x37    ///
#define O2S4_WR_LAMBDA_I_DESC      "O2 Sensor 4 Equivalence Ratio Current"
#define O2S5_WR_LAMBDA_I    0x38    ///
#define O2S5_WR_LAMBDA_I_DESC      "O2 Sensor 5 Equivalence Ratio Current"
#define O2S6_WR_LAMBDA_I    0x39    ///
#define O2S6_WR_LAMBDA_I_DESC      "O2 Sensor 6 Equivalence Ratio Current"
#define O2S7_WR_LAMBDA_I    0x3A    ///
#define O2S7_WR_LAMBDA_I_DESC      "O2 Sensor 7 Equivalence Ratio Current"
```

```
#define O2S8_WR_LAMBDA_I    0x3B    ///
#define O2S8_WR_LAMBDA_I_DESC       "O2 Sensor 8 Equivalence Ratio Current"
#define CAT_TEMP_B1S1       0x3C    ///
#define CAT_TEMP_B1S1_DESC          "Catalyst Temperature Bank 1 Sensor 1"
#define CAT_TEMP_B1S2       0x3E    ///
#define CAT_TEMP_B1S2_DESC          "Catalyst Temperature Bank 1 Sensor 2"
#define CAT_TEMP_B2S1       0x3D    ///
#define CAT_TEMP_B2S1_DESC          "Catalyst Temperature Bank 2 Sensor 1"
#define CAT_TEMP_B2S2       0x3F    ///
#define CAT_TEMP_B2S2_DESC          "Catalyst Temperature Bank 2 Sensor 2"
#define PID_41_60           0x40    //PID 41-60 supported
#define PID_41_60_DESC              "PID 0x41 - 0x60 Supported"
#define MONITOR_STATUS      0x41    ///
#define MONITOR_STATUS_DESC         "Monitor Status This Drive Cycle"
#define ECU_VOLTAGE         0x42    //
#define ECU_VOLTAGE_DESC            "Control Module Voltage"
#define ABSOLUTE_LOAD       0x43    //
#define ABSOLUTE_LOAD_DESC          "Absolute Load Value"
#define COMMANDED_EQUIV_R   0x44    ///
#define COMMANDED_EQUIV_R_DESC      "Commanded Equivalence Ratio"
#define REL_THROTTLE_POS    0x45    ///
#define REL_THROTTLE_POS_DESC       "Relative Throttle Position"
#define AMB_AIR_TEMP        0x46    ///
#define AMB_AIR_TEMP_DESC           "Ambient Air Temperature"
#define ABS_THROTTLE_POS_B  0x47    ///
#define ABS_THROTTLE_POS_B_DESC     "Absolute Throttle Position B"
#define ABS_THROTTLE_POS_C  0x48    ///
#define ABS_THROTTLE_POS_C_DESC     "Absolute Throttle Position C"
#define ACCEL_POS_D         0x49    ///
#define ACCEL_POS_D_DESC            "Accelerator Pedal Position D"
#define ACCEL_POS_E         0x4A    ///
#define ACCEL_POS_E_DESC            "Accelerator Pedal Position E"
```

```
#define ACCEL_POS_F       0x4B   ///
#define ACCEL_POS_F_DESC          "Accelerator Pedal Position F"
#define COMMANDED_THROTTLE  0x4C   ///
#define COMMANDED_THROTTLE_DESC    "Commanded Throttle Actuator"
#define TIME_RUN_WITH_MIL   0x4D   ///
#define TIME_RUN_WITH_MIL_DESC     "Time Run with MIL on"
#define TIME_SINCE_CLR     0x4E   ///
#define TIME_SINCE_CLR_DESC       "Time Since DTC Cleared"
#define MAX_R_O2_VI_PRES    0x4F   ///
#define MAX_R_O2_VI_PRES_DESC       "Maximum Value - Equivalence ratio, O2 Voltage,
O2 Current, Intake Manifold Pressure"
#define MAX_AIRFLOW_MAF     0x50   ///
#define MAX_AIRFLOW_MAF_DESC      "Maximum MAF Airflow Value"
#define FUEL_TYPE         0x51   //
#define FUEL_TYPE_DESC           "Fuel Type"
#define ETHANOL_PERCENT     0x52   //
#define ETHANOL_PERCENT_DESC      "Ethanol fuel %"
#define ABS_EVAP_SYS_PRES   0x53   ///
#define ABS_EVAP_SYS_PRES_DESC     "absolute Evap. System Vapor Pressure"
#define EVAP_SYS_PRES      0x54   ///
#define EVAP_SYS_PRES_DESC        "Evap. System Vapor Pressure"
#define ST_O2_TRIM_B1B3     0x55   ///
#define ST_O2_TRIM_B1B3_DESC       "Short Term Secondary O2 Sensor Trim - Bank 1 and
3"
#define LT_O2_TRIM_B1B3     0x56   ///
#define LT_O2_TRIM_B1B3_DESC       "Long Term Secondary O2 Sensor Trim - Bank 1 and
3"
#define ST_02_TRIM_B2B4     0x57   ///
#define ST_O2_TRIM_B2B4_DESC       "Short Term Secondary O2 Sensor Trim - Bank 2 and
4"
#define LT_O2_TRIM_B2B4     0x58   ///
```

```
#define LT_O2_TRIM_B2B4_DESC        "Long Term Secondary O2 Sensor Trim - Bank 2 and
4"
#define ABS_FUEL_RAIL_PRES  0x59    ///
#define ABS_FUEL_RAIL_PRES_DESC     "Absolute Fuel Rail Pressure"
#define REL_ACCEL_POS       0x5A    ///
#define REL_ACCEL_POS_DESC          "Relative Accelerator Pedal Position"
#define HYBRID_BATT_PCT     0x5B    ///
#define HYBRID_BATT_PCT_DESC        "Hybrid Battery Pack Charge Percent"
#define ENGINE_OIL_TEMP     0x5C    ///
#define ENGINE_OIL_TEMP_DESC        "Engine Oil Temperature"
#define FUEL_TIMING         0x5D    //
#define FUEL_TIMING_DESC            "Fuel Injection Timing"
#define FUEL_RATE           0x5E    //
#define FUEL_RATE_DESC              "Engine Fuel Rate"
#define EMISSIONS_STANDARD  0x5F    ///
#define EMISSIONS_STANDARD_DESC     "Emmissions Requirements"
#define DEMANDED_TORQUE     0x61    ///
#define DEMANDED_TORQUE_DESC        "Driver's Demanded Torque - Percent"
#define ACTUAL_TORQUE       0x62    ///
#define ACTUAL_TORQUE_DESC          "Actual Engine Torque - Percent"
#define REFERENCE_TORQUE    0x63    //
#define REFERENCE_TORQUE_DESC       "Engine Reference Torque"
#define ENGINE_PCT_TORQUE   0x64    ///
#define ENGINE_PCT_TORQUE_DESC      "Engine Percent Torque"
#define AUX_IO_SUPPORTED    0x65    ///
#define AUX_IO_SUPPORTED_DESC       "Auxiliary Input/Output Supported"
#define P_MAF_SENSOR        0x66    ///
#define P_ENGINE_COOLANT_T  0x67    ///
#define P_INTAKE_TEMP       0x68    ///
#define P_COMMANDED_EGR     0x69    ///
#define P_COMMANDED_INTAKE  0x6A    ///
#define P_EGR_TEMP          0x6B    ///
```

```
#define P_COMMANDED_THROT   0x6C   ///
#define P_FUEL_PRESSURE     0x6D   ///
#define P_FUEL_INJ_PRES     0x6E   ///
#define P_TURBO_PRESSURE    0x6F   ///
#define P_BOOST_PRES_CONT   0x70   ///
#define P_VGT_CONTROL       0x71   ///
#define P_WASTEGATE_CONT    0x72   ///
#define P_EXHAUST_PRESSURE  0x73   ///
#define P_TURBO_RPM         0x74   ///
#define P_TURBO_TEMP1       0x75   ///
#define P_TURBO_TEMP2       0x76   ///
#define P_CACT              0x77   ///
#define P_EGT_B1            0x78   ///
#define P_EGT_B2            0x79   ///
#define P_DPF1              0x7A   ///
#define P_DPF2              0x7B   ///
#define P_DPF_TEMP          0x7C   ///
#define P_NOX_NTE_STATUS    0x7D   ///
#define P_PM_NTE_STATUS     0x7E   ///
#define P_ENGINE_RUNTUME    0x7F   ///
#define P_ENGINE_AECD_1     0x81   ///
#define P_ENGINE_AECD_2     0x82   ///
#define P_NOX_SENSOR        0x83   ///
#define P_MANIFOLD_TEMP     0x84   ///
#define P_NOX_SYSTEM        0x85   ///
#define P_PM_SENSOR         0x86   ///
#define P_IN_MANIF_TEMP     0x87   ///

#define PID_REQUEST         0x7DF
#define PID_REPLY           0x7E8

namespace mbed {
```

```
class ecu_reader {
public:
   ecu_reader(int can_speed);
   unsigned char request(unsigned char pid,  char *buffer, char *buffer2 = NULL, char
*buffer3 = NULL, char *buffer4 = NULL);
private:
   int i;
};
}
#endif
```

APÊNDICE C – INTERFACE - CÓDIGO

INDEX.PHP

```
<!DOCTYPE HTML>
<html>
<?php
 header("refresh: 60;"); //refresh the page by the number of seconds on the function

 $LastDataFile = "data.bin";
 $fh = fopen($LastDataFile, 'w');
 fclose($fh);
 $fh = fopen($LastDataFile, 'r') or die("Erro no ficheiro data.bin");
 $stringData = fgets($fh);
 $Time = fgets($fh);
 unlink($LastDataFile); //apaga ficheiro atual
 fclose($fh);
 $SeparateString = explode(",",$stringData);
 $Temp1 = $SeparateString[0];
 $VBat1 = $SeparateString[1];
 $Cons1 = $SeparateString[2];
 $Temp2 = $SeparateString[3];
 $Vbat2 = $SeparateString[4];
 $Cons2 = $SeparateString[5];
 $Lat = $SeparateString[6];
 $Lon = $SeparateString[7];

 //Escreve as coordenadas no formato da API Goole Maps para amostragem no mapa
 $GPSFile = "gps.bin";
 $fh = fopen($GPSFile, 'a') or die("Erro no ficheiro gps.bin");
 fwrite($fh, "|");
 fwrite($fh, $Lat);
 fwrite($fh, ",");
 fwrite($fh, $Lon);
 fclose($fh);

 //Envia os valores de temperatura, tensão da bateria e consumo para o canal no
Thingspeak
 //se necessário, a resposta do Thingspeak encontra-se na variável $response
 $response =
file_get_contents("
                                                ");
?>
```

```
<head>
 <title>Monitorização Remota</title>
 <meta name="description" content="website description" />
 <meta name="keywords" content="website keywords, website keywords" />
 <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
 <link rel="stylesheet" type="text/css" href="css/style.css" />
 <!-- modernizr enables HTML5 elements and feature detects -->
 <script type="text/javascript" src="js/modernizr-1.5.min.js"></script>
</head>

<body>
<div id="bg">
  <img src="images/background.jpg" alt="home">
</div>
 <div id="main">
  <header>
   <div id="logo">
    <div id="logo_text">
     <!-- class="logo_colour", allows you to change the colour of the text -->
     <h1><a href="index.php">Monitorização <span
class="logo_colour">Remota</span></a></h1>
     <h2>Sergio Vasconcelos</h2>
    </div>
   </div>
   <nav>
    <div id="menu_container">
     <ul class="sf-menu" id="nav">
      <li><a href="index.php">Início</a></li>
      <li><a href="#">Viaturas</a>
       <ul>
        <li><a href="viatura01.php">Viatura 01</a></li>
        <li><a href="under_const.html">Viatura 02</a></li>
        <li><a href="under_const.html">Viatura 03</a></li>
        <li><a href="under_const.html">Viatura 04</a></li>
        <li><a href="under_const.html">Viatura 05</a></li>
        <li><a href="under_const.html">Viatura 06</a></li>
        <li><a href="under_const.html">Viatura 07</a></li>
        <li><a href="under_const.html">Viatura 08</a></li>
       </ul>
      </li>
      <li><a href="contact.php">Contato</a></li>
     </ul>
    </div>
   </nav>
  </header>
  <div id="site_content">
```

```
      <h1><b>Sistema Remoto para Monitorização de Ambientes Refrigerados em Viaturas
Automóveis.</b></h1>
      <img src="images/system.png"/>

      <h2>Este site foi desenvolvido no âmbito da dissertação para grau de Mestre do aluno
Sergio Vasconcelos, com título <i>Sistema Remoto para Monitorização de Ambientes
Refrigerados em Viaturas Automóveis</i>, no Instituto Superior de Engenharia da
Universidade do Algarve.<br>
        Este projeto pretende desenvolver um módulo de aquisição de dados e comunicação,
capaz de adquirir, tratar, guardar e comunicar um conjunto de dados críticos, inerentes
tanto às viaturas de uma frota de distribuição de comida fresca e congelada, como ao seu
meio envolvente e à sua carga. Sendo esta uma aplicação empresarial, os fatores que nos
parágrafos seguintes se descrevem refletem a necessidade de ir ao encontro das
especificidades da aplicação. Adicionalmente, e para tornar o módulo completamente
autónomo e comercialmente mais apelativo, também foram implementados: a aquisição
de coordenadas GPS, uma interface para obter dados da viatura e calcular os consumos e
uma comunicação que permite conectar o dispositivo à internet possibilitando a
monitorização remota em tempo real. <br>

      Este <i> site </i> apresenta a interface que permite visualizar os dados obtidos.</h2>

                  </ul>

   </div>
   <div id="scroll">
    <a title="Scroll to the top" class="top" href="#"><img src="images/top.png" alt="top"
/></a>
   </div>
   <footer>
    <p><img src="images/twitter.png" alt="twitter" /> <img
src="images/facebook.png" alt="facebook" /> <img src="images/rss.png" alt="rss"
/></p>
    <p><a href="index.php">Home</a> | <a href="contact.php">Contato</a></p>
   </footer>
  </div>
  <!-- javascript at the bottom for fast page loading -->
  <script type="text/javascript" src="js/jquery.js"></script>
  <script type="text/javascript" src="js/jquery.easing-sooper.js"></script>
  <script type="text/javascript" src="js/jquery.sooperfish.js"></script>
  <script type="text/javascript">
   $(document).ready(function() {
    $('ul.sf-menu').sooperfish();
    $('.top').click(function() {$('html, body').animate({scrollTop:0}, 'fast'); return false;});
   });
  </script>
</body>
</html>
```

DATA.PHP

```php
<?php
 //$stringData = Temp1, VBat1, Temp2, VBat2, Temp3, VBat3,Lat1,Lat2,Lon1,Lon2
   setlocale(LC_TIME, 'portuguese'); //set locale to portuguese. If linux server, 'pt:PT.UTF-8'
   date_default_timezone_set('Europe/Lisbon');
   if(isset($_GET['data'])){

    $stringData = $_GET['data'] ;
    //Acrescenta informações ao ficheiro de log
    $LogFile = "log.bin";
    $fh = fopen($LogFile, 'a') or die("can't open file");
    fwrite($fh, "\n");
    fwrite($fh, strftime("%a %d %g %R"));
    fwrite($fh, " :");
    fwrite($fh, $stringData);
    fclose($fh);

    //Apaga o ficheiro atual e reescreve com os dados novos
    $LastDataFile = "data.bin";
    unlink($LastDataFile); //apaga ficheiro atual
    $fh = fopen($LastDataFile, 'a') or die("can't open file");
    fwrite($fh, $stringData);
    fwrite($fh, "\n");
    fwrite($fh, strftime("%a %d %g %R"));
    fclose($fh);
   }
?>
```

VIATURA01.PHP

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Monitorização Remota</title>
 <meta name="description" content="website description" />
 <meta name="keywords" content="website keywords, website keywords" />
 <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
 <link rel="stylesheet" type="text/css" href="css/style.css" />
 <!-- modernizr enables HTML5 elements and feature detects -->
 <script type="text/javascript" src="js/modernizr-1.5.min.js"></script>
</head>

<body>

 <?php
  //Lê todo o conteúdo para a variável
  $GPSFile = "gps.bin";
  $fh = fopen($GPSFile, 'r') or die("can't open file");
  $GPSCoord = fgets($fh);
  fclose($fh);

  $MapCenter = "37.030391,-7.851487";

 ?>
<div id="bg">
  <img src="images/background.jpg" alt="home">
</div>
 <div id="main">
  <header>
   <div id="logo">
    <div id="logo_text">
     <!-- class="logo_colour", allows you to change the colour of the text -->
     <h1><a href="index.php">Monitorização <span
class="logo_colour">Remota</span></a></h1>
     <h2>Sergio Vasconcelos</h2>
    </div>
   </div>
   <nav>
    <div id="menu_container">
     <ul class="sf-menu" id="nav">
      <li><a href="index.php">Início</a></li>
      <li><a href="#">Viaturas</a>
       <ul>
        <li><a href="viatura01.php">Viatura 01</a></li>
        <li><a href="under_const.html">Viatura 02</a></li>
```

```
            <li><a href="under_const.html">Viatura 03</a></li>
            <li><a href="under_const.html">Viatura 04</a></li>
            <li><a href="under_const.html">Viatura 05</a></li>
            <li><a href="under_const.html">Viatura 06</a></li>
            <li><a href="under_const.html">Viatura 07</a></li>
            <li><a href="under_const.html">Viatura 08</a></li>
          </ul>
        </li>
        <li><a href="contact.php">Contato</a></li>
      </ul>
    </div>
  </nav>
 </header>
 <div id="site_content">


    <h1>Localização Remota e monitorização de Parâmetros</h1>


                    <img
src="http://maps.googleapis.com/maps/api/staticmap?center=<?php echo
$MapCenter?>&zoom=13&size=450x260&markers=color:green <?php echo
$GPSCoord?>"style="width: 450px; height: 260px;" />


<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="http://api.thingspeak.com/channels/31574/charts/3?width=450&height=260&results
=60&dynamic=true&title=Consumo%20L%2F100km" ></iframe>

<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="http://api.thingspeak.com/channels/31574/charts/2?width=450&height=260&results
=60&dynamic=true&title=Sensor%201" ></iframe>


<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="http://api.thingspeak.com/channels/31574/charts/1?width=450&height=260&results
=60&dynamic=true&title=Sensor%201" ></iframe>

                </ul>

 </div>
 <div id="scroll">
  <a title="Scroll to the top" class="top" href="#"><img src="images/top.png" alt="top"
/></a>
 </div>
 <footer>
```

```
    <p><img src="images/twitter.png" alt="twitter" /> <img
src="images/facebook.png" alt="facebook" /> <img src="images/rss.png" alt="rss"
/></p>
    <p><a href="index.php">Home</a> | <a href="contact.php">Contato</a></p>
   </footer>
  </div>
  <!-- javascript at the bottom for fast page loading -->
  <script type="text/javascript" src="js/jquery.js"></script>
  <script type="text/javascript" src="js/jquery.easing-sooper.js"></script>
  <script type="text/javascript" src="js/jquery.sooperfish.js"></script>
  <script type="text/javascript">
   $(document).ready(function() {
    $('ul.sf-menu').sooperfish();
    $('.top').click(function() {$('html, body').animate({scrollTop:0}, 'fast'); return false;});
   });
  </script>
</body>
</html>
```