FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sistema de Sensorização e Telemetria para o FEUP VEC

Tiago Francisco Carção Gil

PARA APRECIAÇÃO POR JÚRI

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Dr. Armando Luís Sousa Araújo

27 de Junho de 2016

Resumo

Os sistemas de telemetria em ambiente de competição automóvel são responsáveis pela monitorização do desempenho do veículo numa prova. Hoje em dia, a utilização de sensores nos automóveis tem crescido exponencialmente, na medida em que a sua utilização permite um controlo mais preciso dos diversos mecanismos dos veículos.

Deste modo, os sistemas telemétricos tornam-se indispensáveis em competições automóveis, permitindo efetuar afinações de modo a melhorar o desempenho do par veículo/condutor.

A realização deste projeto consistiu no desenvolvimento de um sistema de telemetria capaz de monitorizar o FEUP VEC durante provas de competição automóvel. Serão abordadas as tecnologias utilizadas na conceção do sistema em causa, o dimensionamento e a construção do sistema orientado à aplicação no veículo elétrico de competição bem como todo o material utilizado para a elaboração do sistema telemétrico.

Após a implementação será efetuada uma validação de todo o sistema através de diversos testes ao funcionamento do mesmo. Será verificada a validação dos requisitos inicialmente propostos, sendo também referenciados possíveis futuros melhoramentos do sistema.

Abstract

Telemetry systems in motorsport environment are responsible for monitoring vehicle performance on competition racing. Nowadays, the use of sensors in automobiles has increased exponentially and consequently it allows a more precise control of the various mechanisms of the vehicle.

That way, the telemetry systems become necessary in automobile competitions, allowing to make adjustments to improve the performance of the pair vehicle/driver.

The realization of this project consisted on a development of a telemetry system capable of monitoring the FEUP VEC in motorsport competition. All the technologies used will be evidenced, aswell the design and system contruction of the entire project oriented to application in the FEUP VEC. All the material used will be evidenced aswell.

After the implementation, validation tests will be performed for the entire system through various operation modes. The validation of the initially proposed requirements will be verified, and also referred to possible future improvements of the system.

Agradecimentos

A realização desta dissertação representa o fim de mais um ciclo de estudos muito importante para a minha vida pessoal e profissional. Gostaria de agradecer o apoio e incentivo de várias pessoas, nomeadamente:

Ao meu orientador, Prof. Dr. Armando Araújo que se mostrou disponível para orientar o meu projeto de dissertação, estando sempre atento à evolução do trabalho, contribuindo de forma construtiva para a elaboração do mesmo.

À minha família por todo o apoio e paciência durante toda a minha vida, em particular neste percurso académico. A eles lhes devo tudo o que sou hoje e serei um dia. Um Muito Obrigado por tudo.

À minha namorada Francisca Marques que acompanhou de perto o fim do meu percurso académico, sempre dando apoio e incentivo para terminar esta fase da minha vida, Muito Obrigado.

A todos os meus caríssimos amigos António Ribeiro, João Couto, Rui Alves, Carlos Coelho, Joaquim Ribeiro, entre muitos outros que fizeram também parte integrante deste ciclo, o meu Muito Obrigado pela amizade demonstrada, quer nos bons ou maus momentos, e que muito contribuíram para o meu sucesso.

Ao meu amigo e colega António Miguel Sousa por toda a ajuda prestada na realização deste projeto, uma vez que a sua contribuição foi fundamental para a conclusão do mesmo. Esteve sempre disponível para me ajudar a resolver os diversos problemas que foram surgindo. O meu Muito Obrigado.

Por último, gostaria de expressar o meu agradecimento à Faculdade de Engenharia da Universidade do Porto por todo o equipamento e espaço disponibilizados, aos docentes que fizeram parte do meu percurso, e aos diversos técnicos de laboratório que também se mostraram sempre disponíveis. Um Muito Obrigado.

Tiago Francisco Carção Gil

"The best preparation for good work tomorrow
Is to do good work today."

Elbert Hubbard

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ASCII	American Standard Code for Information Interchange
BBB	BeagleBone Black
EUA	Estados Unidos da América
EVCUP	Electric Vehicle Cup
FEUP	Faculdade de Engenharia da Universidade do Porto
FIA	Federation Internationale de l'Automobile
GPIO	Global Propose Input/Output
GPS	Global Positioning System
HDMI	High-Definition Multimedia Interface
I2C	Inter-Integrated Circuit
LSB	Least Significant Bit
MSB	Most Significant Bit
NMEA	National Marine Electronics Association
PCM	Pulse-Code Modutation
POE	Power Over Ethernet
PPM	Pulse Position Modulation
PWM	Pulse-Width Modulation
RNS-E	Radio/Navegation System (Audi)
RTP	Real Time Protocol
SCL	Serial Clock
SDA	Serial Data
SDK	Software Development Kit
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
TTL	Transistor-Transistor Logic
UART	Universal asynchronous receiver/transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VEC	Veículo Elétrico de Competição

Capítulo 1

Introdução

1.1 Contexto

Nos últimos anos tem-se observado um grande desenvolvimento de sistemas eletrónicos nos automóveis. Estes sistemas apresentam inúmeras funções, desde o controlo mecânico do motor até aos pequenos detalhes do habitáculo do condutor.

O número de sensores utilizados pelos automóveis tem vindo a aumentar ao longo dos últimos anos, permitindo ao veículo ter acesso a várias grandezas físicas/elétricas importantes para otimizar o processamento do motor e outros acionamentos dos automóveis [1].

Graças à evolução das tecnologias sem fios e do processamento de dados, os automóveis de hoje em dia são capazes de comunicar à distância, podendo enviar em tempo real vários parâmetros/valores lidos pelos sensores do veículo. Este tipo de comunicação permite diversas aplicações, quer no âmbito da competição automóvel quer no contexto de segurança rodoviária [2].

1.2 Motivação

Com a introdução dos veículos elétricos no mercado automóvel, rapidamente se verificou uma necessidade de introduzir estes mesmos veículos num contexto de competição.

Contudo, já existem diversas competições no âmbito dos automóveis elétricos, como por exemplo a Fórmula E, ou a EVCUP (Eletric Vehicle Cup) – competição de origem britânica com provas por todo o mundo.

Com a entrada deste tipo de viaturas no mundo automóvel e atendendo às suas diferentes características de motorização, surge então a necessidade de criar um sistema capaz de monitorizar diferentes tipos de grandezas presentes neste tipo específico de veículos.

A utilização do FEUP VEC neste projeto tem como objetivo criar um sistema capaz de monitorizar este carro num ambiente de competição através de um ambiente gráfico presente numa estação de comunicação, semelhantes às *Pit Stop* presentes na Fórmula 1.

É de salientar também a utilização deste sistema para melhorias na performance de um veículo de competição através da análise dos diversos dados referentes ao estado do veículo para possíveis afinações.

1.3 Objetivos

Com este trabalho pretende-se criar um sistema modular baseado numa rede de sensores de forma a ser monitorizado numa *Base-Station*, remotamente e em tempo real, de um veículo em movimento.

Pretendem-se ler as acelerações em pista, posição do travão e acelerador e velocidade de caixa engrenada. Para além destas medidas o sistema deve integrar uma câmara web para visualização do percurso em tempo real e um dispositivo GPS para localização do carro em pista.

Todo o processo terá de ser projetado num ambiente gráfico através de um monitor, quer no automóvel, quer na *Base-Station* permitindo uma análise visual sempre em tempo real.

1.4 Análise de requisitos

O sistema a desenvolver enquadrar-se-á num ambiente de competição automóvel, deste modo todo o hardware terá de suportar vibrações provenientes da condução desportiva e desgaste mecânico existente neste tipo de ambiente.

Uma vez que se trata de um veículo de competição, o peso é um fator relevante, deste modo, a meta a ser cumprida fixar-se-á nos 2Kg em todo o *harware* implementado no veículo. Quanto à *Base-Station* o peso não terá qualquer influência na performance do sistema.

A comunicação entre o veículo e *Base-Station* não poderá ter um atraso maior que 5 segundos e deverá ter autonomia de funcionamento de, pelo menos, 20 minutos.

A resolução da imagem de vídeo será, no mínimo, de 640x480 pixéis, podendo ser utilizadas maiores resoluções, nomeadamente 1280x720p ou 1080x1080p.

Os valores a serem lidos no veículo serão: acelerações, localização em pista, posição do travão, posição do acelerador e velocidade de caixa engrenada.

Capítulo 2

Revisão Bibliográfica

Neste capítulo apresentam-se as diferentes tecnologias utilizadas para a construção de todo o sistema, bem como a escolha do diferente material. São também abordados alguns aspetos importantes relativamente aos sistemas telemétricos.

2.1 História da telemetria

A palavra telemetria resulta da junção de duas palavras gregas: Tele – sinónimo de longe, e Meter – que significa medir. Desde modo, telemetria significa a realização de medições à distância, ou remotamente. A telemetria surgiu devido à necessidade de realizar medidas em locais de impossível acesso como, por exemplo, um forno com elevadas temperaturas ou até medições de um míssil em ambiente militar.

Esta tecnologia começou a ser desenvolvida no século XIX, sendo que um dos primeiros sistemas de telemetria desenvolvidos foi o de transmissão de dados entre o Winter Palace (Rússia) e o quartel do exército russo, no ano de 1845. Mais tarde, em 1912, a Commonwealth Edison desenvolveu um sistema de telemetria capaz de monitorizar cargas elétricas numa rede elétrica (*power grid*). Os primeiros sistemas de mísseis e de telemetria espacial soviéticos foram desenvolvidos na década de 1940 usado tanto *pulse-position modulation* (PPM) como *pulse-duration modulation* (PWM). Nos Estados Unidos da América foram desenvolvidos sistemas semelhantes, mas foram substituídos por *pulse-code modulation* (PCM) [3].

2.1.1 Telemetria em Automóveis de Competição

A telemetria é um fator chave no automobilismo de competição, permitindo aos engenheiros realizar uma interpretação dos diversos valores lidos em pista de maneira a modificar e otimizar o comportamento do veículo em ambiente de competição.

Este tipo de sistemas são utilizados em competições como a Fórmula 1. Os grandes objetivos consistem em calcular os tempos de volta à pista e em avaliar o comportamento do condutor de maneira a obter o menor tempo possível. Exemplos de medições num carro de corrida incluem

a aceleração (forças G) a três eixos, leituras de temperaturas, velocidades resultantes nas rodas e deslocamento da suspensão.

Para além da leitura das diversas grandezas físicas, é também registado todo o comportamento do condutor de modo a determinar possíveis erros de performance ou analisar possíveis acidentes em pista.

Anos anteriores a 2003 possibilitavam, neste tipo de competição, a comunicação de duas vias, isto é, piloto-equipa e equipa-piloto, permitindo aos engenheiros atualizações de calibrações do veículo em tempo real (durante a prova). Foi então decidido pela FIA em 2003 a proibição deste tipo de comunicação devido às desigualdades praticadas pelas diferentes equipas da Formula 1. Entretanto, a tecnologia pode ser usada noutros tipos de competições ou em carros de estrada convencionais [4].

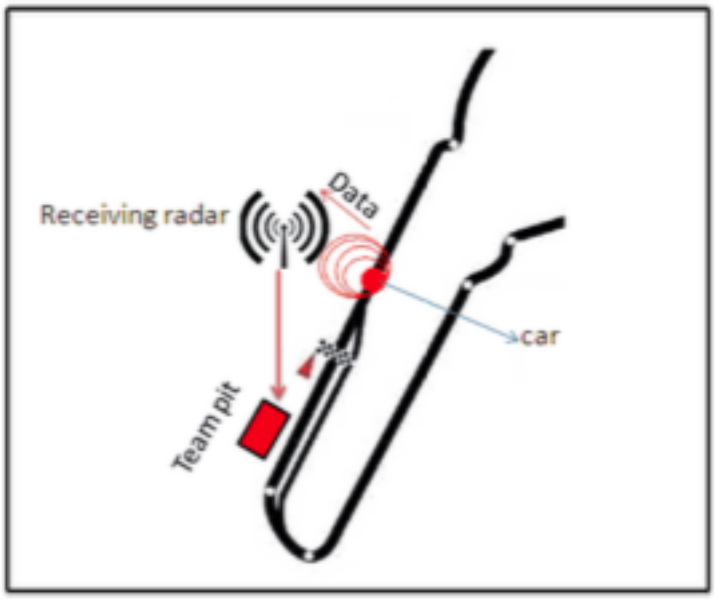


Figura 2.1: Diagrama de telemetria na Fórmula 1

2.2 O FEUP VEC

O projeto FEUP VEC, que foi inicializado em 2010, teve como objetivo o desenvolvimento de um veículo com tração puramente elétrica capaz de competir em provas de automobilismo, como é o caso do Desafio Único - FEUP. O projeto partiu de um automóvel com motor de combustão interna que foi adaptado para um veículo totalmente elétrico. A viatura base foi um Fiat Uno 45S (figura 2.2) uma vez que existem em grande quantidade no nosso país e também pelo facto de terem um preço reduzido. É de salientar que o projeto surgiu de uma parceria entre docentes e alunos do DEEC/DEM da FEUP com o apoio de algumas entidades privadas [5].

Figura 2.2: Fiat Uno 45S - FEUP VEC

2.3 Medição dos diversos sensores

Nesta secção será abordada a escolha do *hardware* necessário para efetuar a leitura/sensorização dos diferentes valores a serem lidos no FEUP VEC.

2.3.1 Plataforma de processamento

De modo a recolher e a processar toda a informação proveniente dos diversos sensores foi necessária a utilização de uma plataforma de processamento. A sua escolha foi baseada nas necessidades e requisitos do sistema a implementar, que são compostos pela aquisição, processamento e transmissão de toda a informação recolhida no veículo.

Tendo em conta as soluções existentes no mercado e o enquadramento do projeto, a alternativa escolhida foi a utilização de um computador de placa única. Estes dispositivos são computadores integrados em uma só placa de circuito impresso, geralmente de dimensões reduzidas, e com capacidades semelhantes às dos computadores convencionais. Existem inúmeros SBC (*Single-Board Computer*), sendo que a pesquisa foi principalmente focalizada na adaptabilidade do dispositivo para o projeto em causa. A escolha limitou-se a três equipamentos - Arduino Uno, BeagleBone Black e Raspberry Pi (tabela 2.1) - uma vez que são dispositivos que dispõem de um grande suporte através de inúmeros projetos elaborados na Internet por muitos utilizadores.

O escolhido foi o BeagleBone Black (figura 2.3) devido às suas características serem mais adequadas ao projeto. O Arduino Uno foi preterido pelas seguintes razões:

- Baixa capacidade de processamento - apenas 16MHz;
- Apenas 13 GPIO digitais;
- Não suporta comunicações com *Ethernet*;
- Apenas 32Kb de memória *Flash*.

Relativamente ao Raspberry Pi, a sua exclusão resultou principalmente de:

- Ter um número reduzido de GPIO (8);
- Elevado consumo energético - mínimo de 700mA;
- Necessidade da utilização de um cartão SD.

Tabela 2.1: Comparação entre Arduino Uno, Raspberry Pi e BeagleBone Black

Nome	**Arduino Uno**	**Raspberry Pi**	**BeagleBone Black**
Model	R3	Model B	Rev C
Processor	ATMega328	ARm11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	1000MHz
RAM	2Kb	512Mb	512Mb
Flash	32Kb	External SD Card	2Gb Onboard Optional External
EEPROM	1Kb		
Input Voltage	7-12V	5V	5V
Min Power	42mA	700mA	210 to 460mA
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6	1	8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Ethernet	N/A	10/100	10/100
Vídeo Out	N/A	HDMI, Composite	microHDMI
Audio Out	N/A	HDMI, Analog	Analog

As características mais relevantes do BeagleBone Black são:

- **Porta USB 2.0** — Interface universal que possibilita a conectividade a diversos periféricos, nomeadamente a uma *Webcam* a ser utilizada para *live-stream* de video;
- **Porta Ethernet** — Interface que permite a ligação a uma rede de comunicação (wireless);
- **Saída microHDMI** — Permite a saída de video para um monitor externo;
- **65 GPIO** — Grande variedade de entradas/saídas analógicas/digitais configuráveis, incuindo UART, I2C e SPI – para leitura de sensores;
- **Armazenamento por microSD** — Possibilidade da expansão do armazenamento interno, que está limitado a 4Gb;

- **Peso** — Apenas 0.039 Kg;
- **Preço reduzido** — Cerca de 45 euros.

Por outro lado, as grandes capacidades de processamento provenientes do processador 1GHz ARM® Cortex-A8 e a compatibilidade com *softwares* como Debian, Ubuntu, Android, Cloud9 (entre outros) facilitam todo o processo de tratamento e transmissão de dados [6].

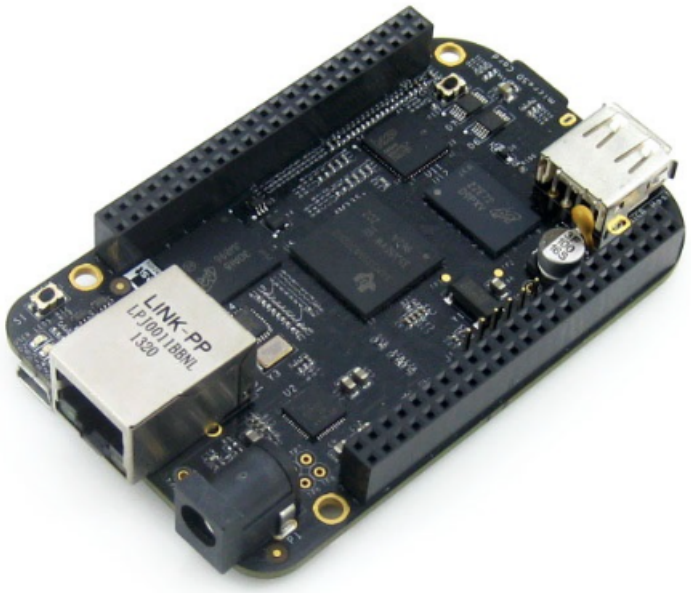


Figura 2.3: BeagleBone Black

2.3.2 Módulo GPS

O GPS é um sistema de radio-navegação por satélite desenvolvido e operado pelo Ministério da Defesa dos E.U.A. Ele é composto por 24 satélites distribuídos pela atmosfera terrestre. A sua utilização permite determinar a posição, velocidade e o fuso horário dos utilizadores em qualquer parte do mundo, nas mais diversas condições atmosféricas (figura 2.4). Os sinais GPS estão disponíveis para um número ilimitado de utilizadores, sendo um sistema de fácil acessibilidade. É um sistema que pode ser utilizado gratuitamente por qualquer pessoa ou entidade [7].

O seu funcionamento baseia-se num processo chamado de trilateração que consiste na receção de sinais de um mínimo de 4 satélites próximos permitindo o cálculo da posição atual. Os sinais representam a posição atual do satélite e o instante de tempo que o impulso (sinal) foi enviado. Desta forma, através de cálculos matemáticos é possível determinar a velocidade, posição e fuso horário de cada utilizador. O cálculo consiste na comparação do tempo em que o sinal foi enviado com o do instante da sua receção. Cruzando essa informação com a de três outros satélites na área, obtêm-se a posição do utilizador. Para além de medidas como a latitude e a longitude, o sistema de trilateração também permite calcular a altura do recetor em relação ao nível do mar.

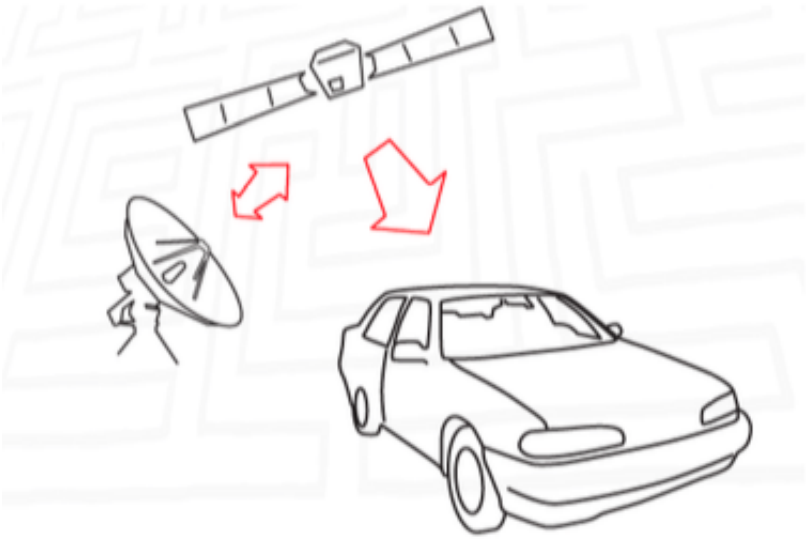


Figura 2.4: Esquema do sistema GPS convencional

Para se efetuar a leitura da localização em pista recorreu-se a um dispositivo GPS compatível com o BeagleBone Black. Existem algumas alternativas no mercado mas como a faculdade dispunha de módulos GPS, foi então utilizado um Ultimate GPS Breakout V3 (figura 2.5) da Adafruit [8].

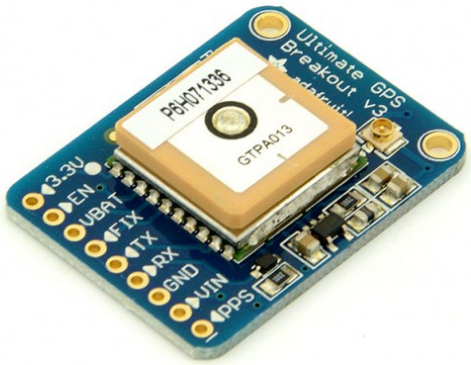


Figura 2.5: Adafruit Ultimate GPS Breakout - Version 3

Este dispositivo adequa-se ao projeto graças à sua comunicação via UART e pelo facto de ter um conector u.FL que permite a ligação de uma antena externa de modo a obter uma melhor aquisição do sinal GPS. Por outro lado, é uma boa opção também, pois funciona a uma tensão contínua de 3.0V a 5.5V (existente no BeagleBone Black) e tem um baixo consumo na ordem dos 25mA (corrente de *tracking*, isto é, aquando efetua a captação de sinal).

Tendo em conta que se trata de um sistema de posicionamento de um veículo em tempo real, a precisão dos dados recolhidos tem um peso considerável na sua escolha. O Ultimate GPS Breakout V3 da Adafruit tem uma precisão de posicionamento de, no máximo, 3 metros e uma precisão da medição de velocidade de 0.1 m/s. É também capaz de medir velocidades até 515 m/s, mais do que suficiente para o projeto em questão. Por outro lado, como se trata de um sistema em tempo real, a taxa de atualização dos valores a serem lidos é um outro fator de peso. O equipamento escolhido permite a leitura de dados entre 1 a 10Hz, que também é suficiente para ser interpretado como uma leitura "contínua" da posição e velocidade do veículo. Uma vez que o diapositivo integrará o sistema de telemetria no FEUP VEC, o seu peso é um fator relevante - apenas 8.5g.

O GPS em questão utiliza o protocolo NMEA 0138 que é utilizado na maioria dos sistemas de posicionamento comerciais [9]. É um protocolo baseado em ASCII que efetua a comunicação através de mensagens (*NMEA Sentences*) que seguem as normas estabelecidas pela NMEA.

Existem várias *NMEA Sentences* que contêm as diversas informações lidas pelo módulo, nomeadamente a posição em latitude e longitude, velocidade relativa ao solo, data e hora da leitura efetuada entre outros. Como o objetivo da utilização do GPS consiste na leitura de posição e velocidade relativa ao solo, a NMEA Sentence escolhida foi a *GPRMC – Recommended minimum specific GPS/Transit data* pois envia os dados pretendidos. O GPS emite uma *string* de caracteres como a do exemplo que se segue:

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7, 191194,020.3,E*68

Todos os valores estão separados por vírgulas, contendo a informação, pela seguinte ordem:

- **GPRMC** — Tipo de *NMEA Sentence*;
- **A** — Aviso recetor de navegação A = OK, V = Atenção;
- **4916.45,N** — Latitude 49 graus 16.45 Minutos Norte;
- **12311.12,W** — Longitude 123 graus 11.12 Minutos Oeste;
- **000.5** — Velocidade relativa ao solo, em Nós (knots);
- **054.7** — Direção relativa ao solo;
- **191194** — Data da leitura 19/Novembro/1994;
- **020.3,E** — Variação magnética 20.3 graus Este;
- ***68** — Verificação de envio.

Deste modo, os valores a serem lidos serão os da latitude, da longitude e da velocidade relativa ao solo.

2.3.3 Acelerómetro e giroscópio de 3 eixos

De modo a efetuar uma leitura da localização espacial do VEC, utilizou-se um acelerômetro/giroscópio de 3 eixos. Existem também diversas alternativas de equipamentos e a sua escolha baseou-se principalmente no reduzido tamanho, baixo consumo e simplicidade da sua implementação com o BeagleBone Black.

Tendo em conta os módulos acelerômetro/giroscópio já existentes na faculdade, o SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050 (figura 2.6) foi o escolhido graças à sua fácil implementação e dimensões reduzidas (25.5x15.2x2.48mm). Por outro lado, as suas especificações são igualmente adequadas ao projeto em causa:

- Possui uma saída digital I2C que é compatível com o BeagleBone Black;
- A tensão de alimentação situa-se nos 2.3V a 3.4V, valores estes suportados pelo BeagleBone Black que possuí uma tensão contínua de 3.3V;
- Corrente máxima de 3.8mA (aquando o funcionamento misto do acelerômetro e giroscópio);
- O giroscópio triaxial possui uma sensibilidade de 131 LSBs/dps (graus por segundo);
- Inclui um algoritmo embarcado para desvio de tempo de de funcionamento e calibragem da bússola sem a intervenção direta do usuário.

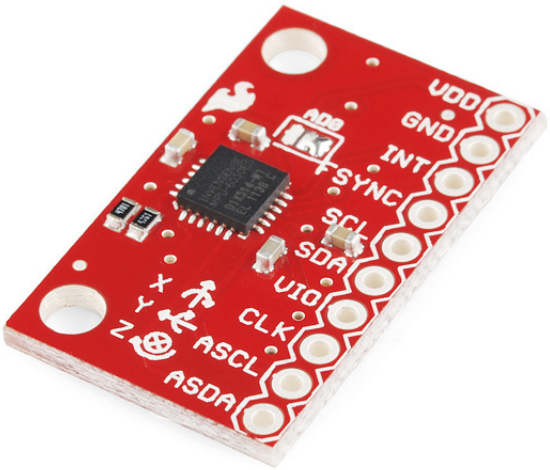


Figura 2.6: SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050

Utilizando então este equipamento é possível obter os ângulos que permitem efetuar uma localização espacial do VEC.

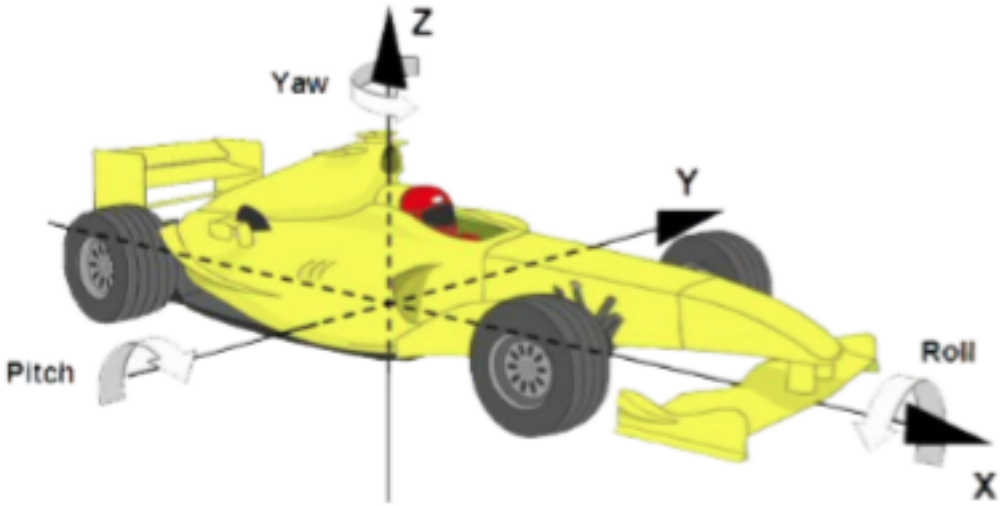


Figura 2.7: Ângulos de Cardan-Bryant

A figura 2.7 ilustra os ângulos introduzidos por Leohnard Euler e modificados por Cardan-Bryant para descreverem a orientação de um corpo rígido. Estes representam três rotações elementares nos eixos de coordenadas de um sistema, conhecidos por ângulos. Uma rotação em torno de Z denominada *yaw* que permite identificar manobras efetuadas. Rotação em torno de X denominada *roll* que permite identificar a diferença de nível entre os dois lados do carro. Por fim, uma rotação em torno de Y denominada por *pitch*, identifica se o carro está numa subida ou numa descida.

O acelerómetro é um sensor que indica o valor da aceleração linear em 3 eixos. No caso de um objeto estático, a única aceleração medida é a da gravidade, que aponta sempre para o centro da Terra (desprezando variações no campo gravítico terrestre). Utilizando a gravidade como referência, é possível definir um plano horizontal, perpendicular ao vetor da aceleração gravítica, e calcular a posição de um objeto relativamente a esse plano, nomeadamente os ângulos que faz com o mesmo. No entanto, numa translação de um objeto, a força exercida para o mover altera a aceleração medida pelo acelerómetro, não sendo possível fazer a anterior identificação com rigor [10].

Um giroscópio é um sensor que mede a velocidade angular de uma rotação, o que para um objeto estático não acrescenta valor, mas que para um objeto em movimento, consegue compensar a sensibilidade do acelerómetro a acelerações momentâneas.

Combinando as medições destes dois sensores, juntamente com um filtro para reduzir o ruído das mesmas, é possível estimar com bastante precisão de que forma se encontra orientado o carro. As equações abaixo permitem calcular a orientação de um objeto (em graus) através da projeção do vetor da aceleração da gravidade em cada um dos eixos do sistema de eixos ligado à partícula.

$$tan(accPitch) = \frac{G_x}{\sqrt{G_x^2 + G_z^2}} \tag{2.1}$$

$$tan(accRoll) = \frac{G_y}{\sqrt{G_x^2 + G_z^2}} \tag{2.2}$$

Para compensar o efeito de desvio criado por outras forças que não a da gravidade, é necessário considerar a informação do giroscópio. O giroscópio disponibiliza informação sobre a velocidade angular instantânea, quer isto dizer que é necessário multiplicar este valor pelo intervalo de tempo entre a medição atual e a anterior.

Quanto menor for este intervalo, mais exata é a medição. Este valor é somado ao calculado na iteração anterior do algoritmo. No entanto, os giroscópios apresentam um *drift* (desvio) de valor não desprezável. Devido a este *drift*, a integração das medições do giroscópio afasta-se de 0, mesmo com o objeto em repouso. Para compensar este efeito, é utilizada a informação mais estável do acelerómetro.

Note-se a forma como estes dois sensores se complementam mutuamente, sendo o acelerómetro útil para a medição de variações de baixas frequências, e o giroscópio para variações de alta frequência. Abaixo apresenta-se a metodologia utilizada para combinar a informação dos dois sensores. Foi utilizado um parâmetro responsável pela importância dos valores do acelerómetro e giroscópio no resultado final. As equações representadas abaixo foram baseadas num tutorial de utilização do MPU6050 [11] :

$$gyroAngle_{x,y,z} = gyroMeasurement_{x,y,z} \times \Delta t \tag{2.3}$$

$$tempPitch = previousPitch + gyroAngle_x \tag{2.4}$$

$$tempRoll = previousRoll + gyroAngle_y \tag{2.5}$$

$$roll = tempRoll \times \alpha + accRoll \times (1 - \alpha) \tag{2.6}$$

$$pitch = tempPitch \times \alpha + accPitch \times (1 - \alpha) \tag{2.7}$$

A comunicação efetuada entre o MPU-6050 e o BeagleBone Black é realizada através de I2C (ou IIC), estando o BeagleBone a atuar como Mestre e o MPU como Escravo.

O BeagleBone Black, como dispositivo Mestre, controla a rede à qual podem ser ligados inúmeros Escravos. Estes estão ligados ao *Serial Data* (SDA) - transmissão de informação bidirecional - e ao *Serial Clock* (SCL) - sincronização da transferência.

O MPU-6050 envia uma trama de dados por I2C como resposta ao pedido efetuado pelo BBB com a informação proveniente dos registos de interesse, listados no documento "MPU-6000 and MPU-6050 Register Map and Descriptions - Revision 4.1" [12].

2.3.4 Leitura acelerador e travão

Atualmente o setor automóvel tem vindo a substituir a maioria das suas peças mecânicas por instrumentos eletrónicos. Estes sistemas tendem a ser mais robustos e eficazes no que diz respeito à performance e ao rendimento dos automóveis.

Um exemplo desta evolução é a implementação do acelerador eletrónico, mais conhecido como *Drive-by-wire* [13], na maioria dos modelos comercializados e produzidos em grande escala. Inicialmente, este tipo de tecnologia surgiu nos carros de Fórmula 1 devido ao seu elevado desempenho e controlo, mas imediatamente foi aproveitada nos veículos destinados ao público comum.

Até aos dias de hoje, na maioria dos veículos automóveis, a união entre o pedal do acelerador e a borboleta de admissão era feita com o auxílio de um cabo acelerador mecânico. Este tipo de sistemas apresentava um maior desgaste mecânico e era muito limitado relativamente ao controlo da borboleta. Deste modo, o acelerador mecânico foi substituído por um sistema de controlo eletrónico: o pedal comunica a uma central de comando a posição atual, e consoante o modelo de controlo utilizado, um motor é responsável por adequar a posição da válvula-borboleta de admissão (figura 2.8).

Deste modo, o pedal de aceleração funciona como um potenciómetro que envia um sinal de comando à centralina. Sendo assim, a leitura deste tipo de sinal é bastante simples graças ao seu funcionamento de baixo nível.

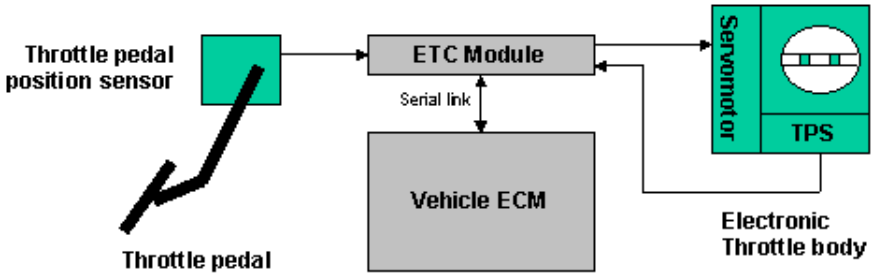


Figura 2.8: Sistema eletrónico de aceleração - Drive-by-wire

Um potenciómetro é um componente eletrónico capaz de variar a sua resistência. Deste modo, permite a variação dos valores de tensão aos seus terminais, segundo a linearidade da variação da resistência.

Sendo então o pedal do acelerador um componente em que o princípio de funcionamento se assemelha a um potenciómetro, então o valor que o pedal envia à centralina é variável em tensão.

2.3.5 Leitura da velocidade de caixa engrenada

Atualmente, os veículos vêm equipados com uma grande quantidade de sensores que têm como função enviar informação vital para os módulos de controlo. Os sensores de posição de caixa, também conhecidos como GPS - *Gear Position Sensor*, reagem com base em variáveis de mudança de velocidades [14].

Este tipo de sensores, que geralmente se encontram no eixo seletor da transmissão, consistem numa série de *switches* ou dispositivos eletrónicos que têm como função enviar informações para o módulo de controlo de transmissão em relação à posição atual da velocidade de caixa engrenada.

2.4 Protocolos de comunicação de dados

Nesta secção será feita uma abordagem aos tipos de comunicação de dados a serem utilizados entre os sensores em causa e a plataforma de processamento. Serão referidas as comunicações via UART e I2C uma vez que representam as tecnologias de comunicação por parte do GPS e Acelerómetro/Giroscópio respetivamente.

2.4.1 Comunicação série

A comunicação série é um dos métodos utilizados no envio e receção de dados entre dois ou mais dispositivos. Consiste no processo de enviar informação um bit de cada vez, de forma sequencial, através de um canal de comunicação. Existem diversas maneiras de efetuar esta transmissão de dados, mas serão apenas abordadas as comunicações via UART e I2C devido à sua utilização neste projeto.

2.4.2 UART

UART - *Universal Asynchronous Receiver/Transmitter* consiste num periférico geralmente associado a um microprocessador/microcontrolador capaz de traduzir entre dados em série e em paralelo, associados a comunicações série. O UART é descrito como assíncrono porque o remetente não necessita de enviar um sinal de *clock* de modo a sincronizar a transmissão, sendo substituído por bits de *start* e *stop*. Como não é necessário o envio de um sinal de *clock*, tipicamente estas comunicações só necessitam de dois fios de sinal para transmitir a informação corretamente. Como uma simples linha telefónica, a ligação de transmissão de dados (TXD) de uma extremidade é conectada à ligação de receção de dados (RXD) da outra extremidade, e vice-versa [15].

O protocolo de comunicação série assíncrona apresenta um conjunto de regras e mecanismos que ajudam a assegurar a robustez e a inexistência de erros na transferência dos dados. Estes mecanismos são os seguintes:

- Bits de dados;
- Bits de sincronização;
- Bits de paridade;
- *Baudrate* (taxa de transmissão).

É de salientar que de forma a efetuar uma comunicação série deste tipo, os dois dispositivos ligados mutuamente terão de estar configurados de forma a respeitarem o mesmo protocolo de transmissão.

O ***Baudrate*** (taxa de transmissão) especifica a rapidez que a informação é enviada através da comunicação série. É geralmente expressa em *bits-per-second* (bps). Invertendo o valor do *baudrate*, é possível obter o tempo que demora a enviar um único bit. O valor da taxa de transmissão

determina por quanto tempo o remetente mantém a comunicação série em *High/Low* ou em que período o dispositivo de receção se baseia para obter as amostras do sinal recebido. O requisito que permite definir o valor desta taxa é o de os dois equipamentos operarem ao mesmo *baudrate*. Um dos valores mais comuns, quando a velocidade não é um parâmetro crítico, é o de 9600 bps. Outros valores também bastante utilizados são 1200, 2400, 4800, 19200, 38400, 57600 e 115200. Quanto maior o valor do *baudrate*, maior a velocidade de envio e receção, mas existem limites que determinam o quão rápido a informação pode ser transmitida. Os valores normalmente não ultrapassam os 115200 bps pois a maioria dos microcontroladores não o suportam. Se o valor for aumentado em demasia, erros surgirão na receção dos dados devido à impossibilidade de recolher amostras dos dados nos tempos definidos.

Cada bloco de dados transmitidos (geralmente um byte) é enviado num *packet/frame* de bits:

Figura 2.9: Frame numa comunicação série

A figura 2.9 acima representa um *frame* de bits, que é composta por:

- *Start bit*;
- *Data bits* (dados);
- *Parity bit* (bit de paridade);
- *Stop bits*.

Data bits (dados) - O objetivo de cada pacote série define-se pelos dados/informação que carrega. A quantidade de dados a ser enviada é variável, podendo-se obter valores de 5 a 9 bits. O tamanho padrão está fixado nos 8 bits, que corresponde a um byte, mas os outros tamanhos também são utilizados. Depois de ambos os dispositivos estarem configurados para o mesmo comprimento dos *data bits*, também é necessário configurar a ordenação dos seus dados. A forma padrão respeita o envio do bit do menos significativo (LSB) para o mais significativo (MSB), podendo ser alterado, se necessário em determinadas situações.

Bits de sincronismo - Os bits de sincronismo são cerca de dois ou três bits especiais que estão presentes em cada pacote de dados enviados. Estes bits são o *start bit* e o(s) *stop bit(s)*. De acordo com a sua nomenclatura, estes bits são responsáveis por indicar o inicio e o fim de cada pacote de dados enviados. Existe apenas um *start bit*, mas o número de bits do *stop bit* é configurável - 1 ou 2 – mas geralmente é representado apenas com 1 bit.

Bit de paridade - A paridade é uma forma muito simples de verificação de erros. O bit de paridade pode ser de dois tipos: par ou ímpar. Para produzir o bit de paridade, todos os 5-9 bits do byte de dados são somados, e a uniformidade da soma decide se o bit é definido ou não. Por exemplo, assumindo que a paridade é definida como sendo par e se o byte de dados for igual a 0b01011101, que tem um número ímpar de 1 's (5), o bit de paridade seria definido como 1. Caso contrário, se o modo paridade for definido como ímpar, o bit de paridade seria definido como 0. A paridade é um método opcional e não é muito utilizado. Pode ser útil para a transmissão em meios ruidosos, mas consequentemente atrasa o envio dos dados e o recetor terá de suportar um mecanismo de tratamento de erros.

Este tipo de comunicação pode ser constituído de duas diferentes maneiras: *Full Duplex* ou *Half Duplex*.

- ***Full Duplex*** – ambos os dispositivos são capazes de enviar e receber dados simultaneamente;
- ***Half Duplex*** – cada dispositivo será configurado como transmissor ou recetor.

Fisicamente, as ligações entre dois dispositivos seguem a ordem da seguinte figura 2.10:

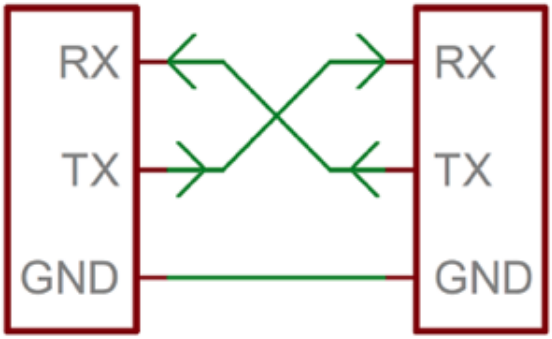


Figura 2.10: Ligação dispositivos série - RX e TX

A comunicação série consiste em apenas duas ligações: uma para enviar dados e outra para receber dados. Nos dispositivos, estas ligações estão representadas nos respetivos pinos RX (receção) e TX (transmissão).

Do lado do BeagleBone Black, o método de comunicação série consiste em TTL (lógica transístor-transístor) em que os limites de tensão são entre 0V e 3.3V. Um lógico "1" é representado por 3.3V enquanto que o lógico "0" simboliza os 0V.

Deste modo, após receber pacote a pacote, o UART do BeagleBone Black tem a tarefa de traduzir a receção de cada carácter vindo do GPS numa mensagem de caracteres. Neste caso é um conjunto de caracteres ASCII, que posteriormente será interpretado via *software*.

2.4.3 I2C

Inter-Integrated Circuit (I2C) é um barramento simples de dois fios que foi projetado pela Philips em 1980 para fazer a interface entre microprocessadores/microcontroladores para dispositivos periféricos de baixas velocidades [16]. Um dispositivo mestre controla todo o barramento, e muitos dispositivos escravos endereçáveis são ligados de modo comum aos mesmos dois fios.

Manteve-se popular ao longo dos anos, graças à sua simplicidade com a adoção por parte dos fabricantes. Atualmente é maioritariamente utilizado em *smartphones*, em grande parte dos microcontroladores e aplicações de gestão.

A comunicação via I2C destaca-se essencialmente por serem apenas necessárias duas linhas de sinal para efetuar a comunicação. Uma linha de dados série (SDA) para a transmissão bidirecional dos dados, e uma linha de *serial clock* (SCL) responsável pela sincronização da transferência de informação. Devido ao barramento utilizar o sinal de *clock*, a transferência classifica-se como síncrona. Relativamente à linha de sinal SDA, denomina-se de bidirecional pois tanto permite o envio como a receção de dados.

Cada dispositivo no barramento pode atuar como um mestre ou um escravo. O dispositivo mestre é aquele que inicia a comunicação e o dispositivo escravo é aquele que responde. Os dispositivos escravos não têm permissão para inicializar uma comunicação com os dispositivos mestres.

A cada dispositivo escravo conectado ao barramento é-lhe pré-atribuído um endereço único. Uma típica ligação entre mestres e escravos pode ser ilustrada como na figura 2.11:

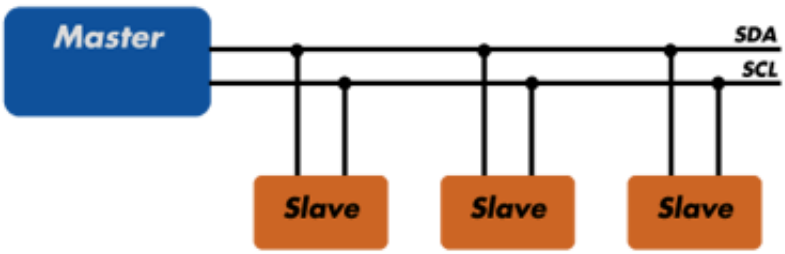


Figura 2.11: Esquema de ligação de dipositívos - I2C

2.5 Captação de vídeo em tempo real

A presente secção terá como objetivo a escolha da câmara de vídeo a ser utilizada no FEUP VEC.

2.5.1 Escolha da câmara de vídeo

Para efetuar a captação de vídeo foi necessária a escolha de uma câmara que cumprisse os requisitos inicialmente definidos. Os principais fatores que levaram à decisão da escolha da câmara foram o de suportar resoluções elevadas, nomeadamente de 1280x720p e 1920x1080p, a compatibilidade com a plataforma de processamento escolhida e o possível processamento e codificação de imagem *on-board*.

Existe um mercado enorme de câmaras web capazes de cumprir os requisitos acima mencionados, mas após uma pesquisa de projetos semelhantes já elaborados [6], chegou-se à conclusão que a câmara indicada seria a Logitech HD Pro C920 (figura 2.12) pelas seguintes razões:

- Conexão USB 2.0 (compatível com o BBB);
- Captura em 1080p e codificação H.264 via Hardware;
- Existência de Drivers e suporte para Linux;
- Suporte tripé-ready (fácil fixação no veículo).

Uma outra característica é o facto de a câmara efetuar compressão e codificação do vídeo em *hardware* diretamente no formato H.264, o que permite que o BeagleBone Black apenas encaminhe os dados da câmara para a porta de rede determinada, sem ter que efetuar processamento do vídeo. Isto liberta bastantes recursos de processamento, vitais num sistema que se pretende que seja rápido e de baixo consumo.

Figura 2.12: Webcam Logitech HD Pro C920

Posteriormente, a sua implementação baseou-se principalmente num projeto realizado pelo Professor Derek Molloy (Dublin City University) [6] que disponibilizou bastante informação e código que permitem para captação de imagem com um BeagleBone Black e a câmara Logitech C920 Pro.

2.6 Comunicação entre o FEUP VEC e a Base-Station

Nesta secção serão abordadas as tecnologias existentes para a comunicação sem fios entre o FEUP VEC e a Base-Station. Será abordada também a escolha da tecnologia a ser posteriormente implementada no projeto.

2.6.1 Tecnologias existentes

A comunicação entre o veículo e a *Base-Station* utilizará uma tecnologia sem fios. Após a aquisição dos dados, é necessário transmitir os mesmos para uma localização remota, onde serão analisados. Existem variadas tecnologias e métodos para o fazer. Em primeiro lugar deve ser considerada a frequência da portadora utilizada nas várias normas disponíveis. A tabela 2.2 apresenta alguns dos parâmetros mais relevantes na comparação, assumindo a utilização de emissores/recetores de características semelhantes (potência/sensibilidade).

Tabela 2.2: Comparação do desempenho de algumas tecnologias de comunicação.

	<1GHz	2.4 GHz	5 GHz
Transparência Ethernet	Não	Sim	Sim
Débito	-	+	++
Distância	+++	+	–

Tecnologias em que a frequência da portadora é inferior a 1GHz permitem a transmissão de dados a distâncias bastante grandes, no entanto, o débito que disponibilizam é incompatível com o necessário para a transmissão de vídeo de alta definição em tempo real. A grande maioria das soluções desta gama também não permitem que se realize uma abstração da rede por protocolos Ethernet, descartando por completo este conjunto de soluções.

Ambas as tecnologias de 2.4GHz e 5GHz permitem o uso de protocolos *standard* de comunicação em rede. A diferença entre as duas prende-se com o compromisso distância/débito disponibilizado. A tecnologia de 5GHz permite a transmissão com um débito superior, mas a atenuação do sinal, com o aumento da distância, é muito superior àquela sofrida por tecnologias de 2.4GHz [17].

2.6.2 Escolha da tecnologia a utilizar

Após a análise anterior, decidiu-se, portanto, optar por utilizar uma portadora de 2.4GHz, já que oferece um bom equilíbrio dos fatores considerados.

Tendo em conta que o BeagleBone Black suporta ligações de rede via Ethernet e que não disponibiliza qualquer dispositivo integrado de comunicações sem fios, a pesquisa focou-se principalmente na utilização desta interface (Ethernet).

Deste modo, procuraram-se dispositivos capazes de criar uma ligação entre dois pontos, via *wireless*. Os aparelhos utilizados foram os Ubiquiti Bullet M2-HP (figura 2.13), que são vistos pelos dispositivos conectados como placas de rede comuns, efetivamente abstraindo a metodologia

utilizada na conexão entre pontos. Desta forma permitem criar uma rede funcional através dos dois equipamentos: um deles será configurado como *router* e o outro como *station*.

O *router* será responsável por definir e atribuir endereço aos dispositivos que se ligam à rede, funcionando como um ponto de acesso. Todos os equipamentos que se pretendam ligar à rede devem fazê-lo ligando-se a este ponto de acesso como *station*.

As características mais importantes do Ubiquiti Bullet M2-HP são:

- Alta performance 802.11b/g/N (rede sem fios);
- Largura de banda ajustável (5/8/10/20/30/40 MHz);
- Compatível com a maioria das antenas;
- Suporte de *software* - Ubiquiti AirOS;
- *Plug & Play*;
- POE *(power over ethernet)*;
- Alcance até 50Km (dependendo da antena);
- Resistente à água e condições atmosféricas adversas;
- Peso - 0.18Kg.

Figura 2.13: Ubiquiti Bullet M2

2.6.3 Protocolos de comunicação

Existe uma grande variedade de protocolos para a transmissão de dados através de um canal *Ethernet*, os de interesse para este projeto são TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*) e RTP (*Real Time Protocol*), cada um apresentando as suas vantagens, representadas na tabela 2.3.

Tabela 2.3: Comparação do desempenho de alguns protocolos de i interesse.

	Acknowledge	Débito
TCP	Sim	–
UDP	Não	+
RTP	Não	++

A característica distintiva do protocolo TCP é a garantia de que a informação entregue no destino não está corrompida, graças a um processo de *acknowledge* que permite verificar a integridade dos dados, e pedir uma retransmissão se for o caso. No entanto, esta metodologia cria algum *overhead* na comunicação, que embora apresente vantagens para a transmissão de dados sensíveis, se torna um obstáculo na transmissão em tempo real de conteúdo multimédia [18].

De seguida considerou-se o protocolo UDP, que, por não conter o sistema de *acknowledge*, oferece bastantes vantagens para a transmissão de vídeo. Existe no entanto uma variação de UDP que proporciona ainda mais vantagens para este efeito, trata-se de RTP [19]. O RTP foi especificamente desenhado para suportar a transmissão de dados multimédia (e não só) em tempo real, permitindo efectuar, por exemplo, sincronização da *stream* (vídeo em tempo real) e compensação de *jitter* no recetor. Atendendo a estas características, definiu-se que seria utilizado TCP para a transição de dados de sensorização e RTP para a transmissão de vídeo.

2.7 Software utilizado para monitorizar o FEUP VEC na Base-Station

2.7.1 Cliente JAVA

Do lado da *Base-Station* será criada uma interface responsável pela apresentação dos dados dos diversos sensores. O objetivo principal será então, de forma intuitiva, conseguir representar os valores dos ângulos *pitch/roll*, a posição do veículo num mapa e os valores de acelerador/travão como também a velocidade de caixa engrenada.

A linguagem de programação escolhida foi o JAVA [20] pelo facto de existir suporte para a mesma em múltiplas plataformas, sendo necessários poucos ou nenhuns ajustes para portar a interface para outro dispositivo (por exemplo o Android) caso seja necessário. O *software* escolhido foi o IntelliJ IDEA 15 CE desenvolvido pela JetBrains [21] porque cumpre o requisito de ser um editor e compilador de código JAVA. Poderia ter sido escolhido outro semelhante, mas foi por uma questão de conveniência, uma vez que o sistema operativo utilizado no desenvolvimento do projeto foi o OS X, desenvolvido pela Apple Inc., e existe compatibilidade entre ambos.

2.7.2 Visualização do video em tempo real

Do lado da *Base-Station* será necessário reproduzir o vídeo transmitido pelo FEUP VEC. Tendo em conta que o protocolo utilizado para a sua transmissão é o RTP (*Real-time Transport Protocol*), o software a utilizar no PC terá de ser compatível com este.

O VLC (figura 2.14) é um leitor multimédia, multiplataforma, que reproduz a maioria dos formatos de conteúdo multimédia de forma bastante rápida [22]. Suporta diversos protocolos de *stream* de dados, destacando-se o RTP, que consiste no protocolo utilizado na transmissão do vídeo pela rede FEUP VEC <-> *Base-Station*. Por outro lado, o VLC tem a capacidade de efetuar a leitura do vídeo em formato H.264, uma vez que é o formato de vídeo a ser recebido pela *Base-Station*.

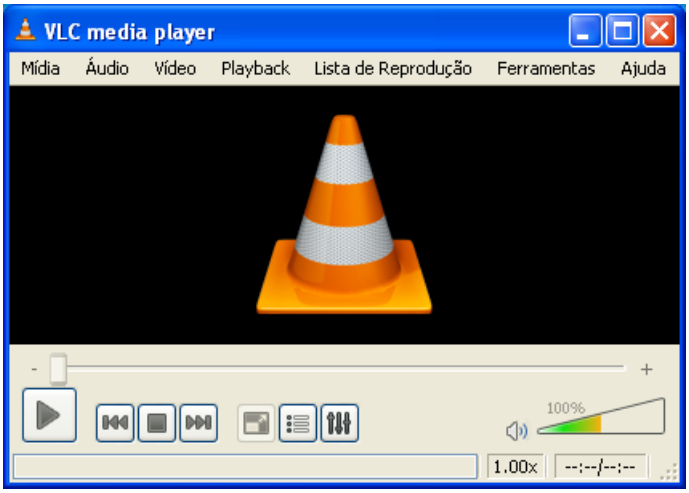


Figura 2.14: VideoLAN Media Player

2.8 Monitor de interface FEUP VEC - Condutor

Do lado do VEC também foi necessária uma interface que permitisse a leitura dos dados em tempo real por parte do condutor/piloto. Esta foi constituída por um monitor de pequenas dimensões que permitiu ao condutor ter uma noção bastante real do comportamento do veículo em pista.

2.8.1 Monitor 4DCAPE-70T

Sendo o BeagleBone Black a plataforma de processamento do sistema, a escolha de um monitor baseou-se na compatibilidade direta com este. Existem dois tipos de alternativas: um monitor com ligação HDMI ou um *cape* composto por um monitor. Um *cape* consiste num equipamento de encaixe direto no BeagleBone Black especialmente desenvolvido para trabalhar em conjunto.

A opção utilizada foi então um *cape* desenvolvido pela 4DSystems que consiste num monitor *touch-screen* de 7 polegadas (4DCAPE-70T) (figura 2.15). A escolha baseou-se principalmente nas capacidades acrescidas relativamente a um monitor convencional: encaixe direto no BeagleBone Black, a possibilidade de replicar os pinos de modo a poderem ser utilizados para os sensores e pelo facto de ser sensível ao toque. Por outro lado, o *cape* apenas tem um peso de 0.312 Kg, valor este que irá contribuir para o peso total do sistema do veículo.

Figura 2.15: 4DCAPE-70T - 4D SYSTEMS

O *cape* em questão utiliza uma grande parte dos pinos do BeagleBone Black, sendo então necessária uma posterior atenção na utilização dos pinos restantes, de modo a não entrar em conflito com os já reservados pelo *cape*. A figura 2.16 representa os pinos do BeagleBone Black reservados pelo *cape*:

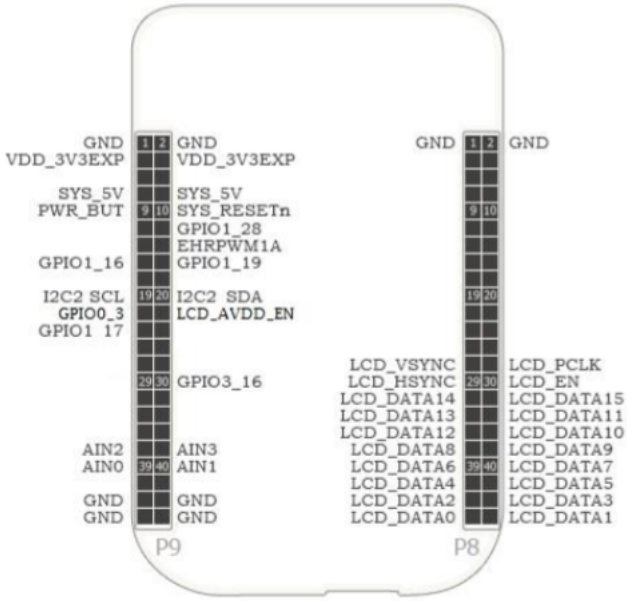


Figura 2.16: 4DCAPE-70T - utilização dos pinos do BeagleBone Black

É de salientar que para uma utilização correta do conjunto *cape* e BeagleBone Black, é recomendado pelo fabricante uma alimentação de 5V-DC @ 2A.

O *cape* dispõe de *headers* secundários de modo a serem reutilizados os pinos, uma vez que deixam de ter acesso aquando o BeagleBone Black encaixa no monitor (BeagleBone Black Only),

como sugere a figura 2.17.

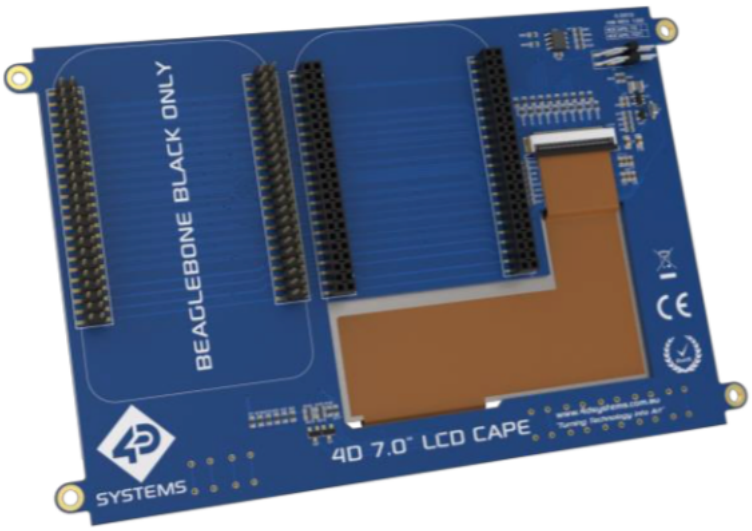


Figura 2.17: 4DCAPE-70T - replicação dos pinos

A disposição dos pinos nos *headers* secundários está descrita na *datasheet* do produto.

2.9 Software utilizado para monitorizar o FEUP VEC pelo condutor

Neste tópico será abordado o software utilizado no desenvolvimento de uma aplicação a implementar no BeagleBone Black de modo a permitir ao condutor do FEUP VEC efetuar uma monitorização em tempo real dos valores emitidos pelos sensores.

2.9.1 Aplicação QT

A interface presente no FEUP VEC consistiu numa projeção dos diferentes valores dos sensores num monitor ligado ao BeagleBone Black. Deste modo, foi necessária a utilização de um *software* que permitisse então fazer o *display* dos diferentes dados.

Para a criação da aplicação no veículo foi utilizado o Qt, que consiste num *framework* multiplataforma, incluindo Linux, que permite o desenvolvimento de interfaces gráficas em C++. É compatível com o BeagleBone Black, uma vez que tem como sistema operativo o Debian, que utiliza *kernel* Linux.

O IDE utilizado a criação da aplicação foi o QTCreator (figura 2.18) que é um IDE multiplataforma que inclui a Qt SDK. O Qt Creator possui uma componente de edição de código e um modelador de interface gráfica (Qt Designer) para desenvolvimento de GUIs (aplicações gráficas). O Editor entende diversas linguagens de programação incluindo C++, auxiliando a formatação de todo o código. O Designer, proporciona as ferramentas necessárias para se criarem os GUIs através de *widgets* disponibilizados pelo QT.

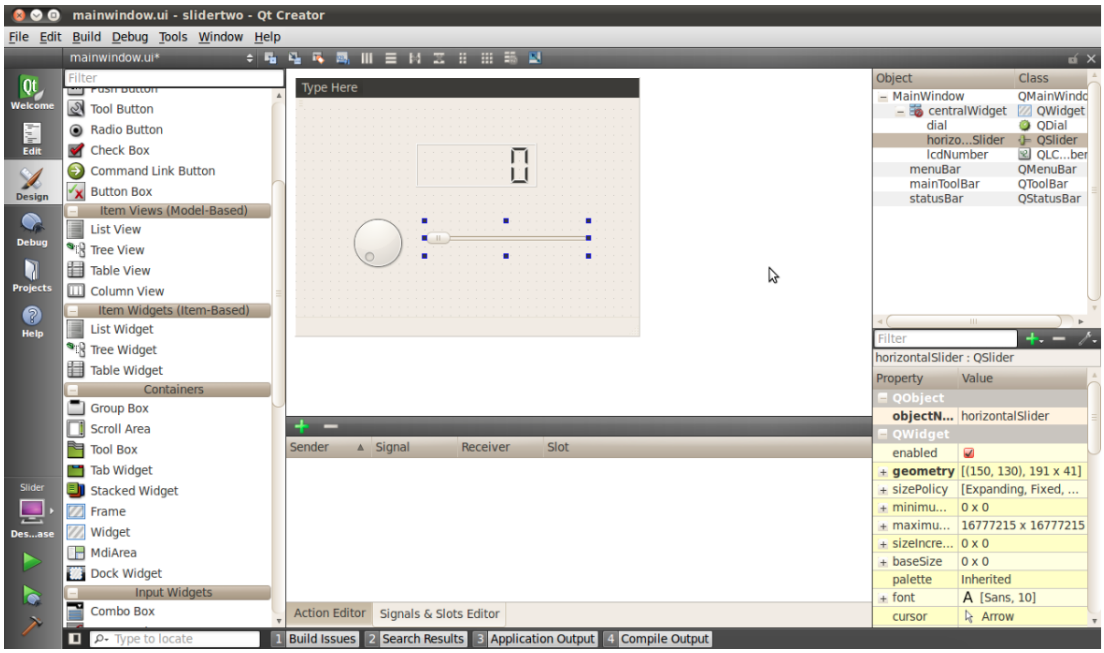


Figura 2.18: Qt Creator

Capítulo 3

Implementação

3.1 Leitura e processamento dos dados

3.1.1 Módulo GPS

O módulo Adafruit Ultimate GPS Breakout V3 é um dispositivo que comunica via UART. O seu funcionamento resume-se ao envio de *NMEA Sentences* pela porta série, contendo as informações relativas ao posicionamento, velocidade relativa ao solo, entre outros.

O esquema elétrico de ligação com o BeagleBone Black segue o esquema da figura 3.1:

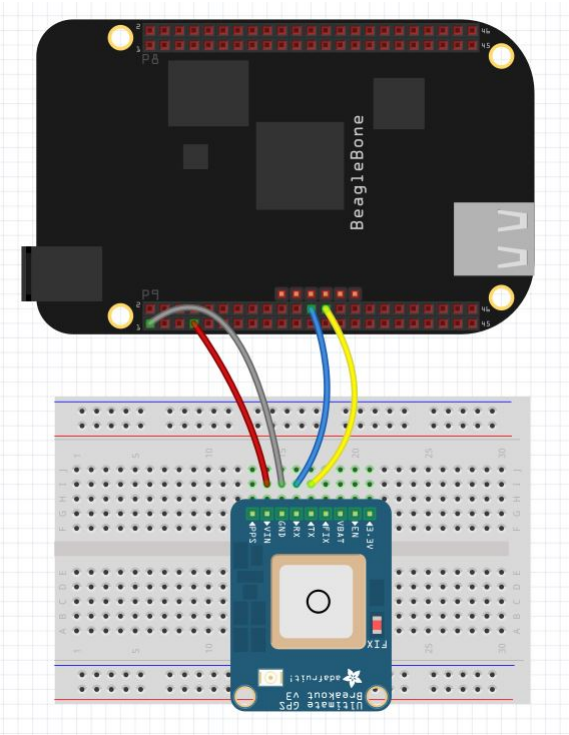


Figura 3.1: Esquema elétrico BeagleBone Black - GPS

As ligações elétricas entre o BeagleBone Black e o GPS seguem a informação da tabela 3.1:

Tabela 3.1: Pinout do esquema elétrico BeagleBone Black e GPS

GPS	Pino BBB	Informação pino
VIN	P9_7	SYS_5V
GND	P9_1	DGND
RX	P9_24	UART1_TXD
TX	P9_26	UART1_RXD

Como se pode ver na figura 3.1, os pinos RX e TX são ligados, respetivamente, aos pinos P9_24 (UART1_TXD) e P9_26 (UART1_RXD) do BeagleBone Black. A alimentação do dispositivo GPS é feita entre os 5V DC e o *Ground* do BBB, sendo então ligados os pinos VIN e GND (do GPS), respetivamente, aos pinos P9_7 (SYS_5V) e P9_1 (DGND).

O BeagleBone Black tem 4 UARTs [6] que estão acessíveis através dos pinos nos *headers* P8 e P9, segundo a tabela 3.2:

	UART1	UART2	UART3	UART4	UART5
TXD	P9_24 (0)	P9_21 (1)	Não exposto	P9_13 (6)	P8_37 (4)
RXD	P9_26 (0)	P9_22 (1)	Não exposto	P9_11 (6)	P8_38 (4)

Tabela 3.2: UART no BeagleBone Black

Tendo em conta que o *cape* do monitor irá utilizar uma grande parte dos pinos disponíveis, utilizou-se o UART1, uma vez que não entra em conflito com o *cape* utilizado.

Para efetuar a comunicação série entre o GPS e o BeagleBone Black utilizou-se a função *getNMEA(char* buffer)* (ficheiro ConnectionHandler.cpp), que tem como objetivo receber os dados do GPS. A função inicializa a comunicação entre os dois dispositivos através da UART1 do BeagleBone Black, permitindo depois o envio de dados para o GPS e a receção da informação das *NMEA Sentences*. O Adafruit Ultimate GPS Breakout V3 permite a receção de comandos que configuram o seu funcionamento, nomeadamente a taxa de atualização, a periodicidade da leitura e a escolha a *NMEA Sentence*. Deste modo, foram utilizados os seguintes comandos através da função *write()* que permitiram a alteração da configuração do GPS:

- write(fd, "$PMTK314,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0*29\r\n", 51) - Apenas GPRMC NMEA Sentence;
- write(fd, "$PMTK220,200*2C\r\n", 17) - Update a cada 200ms;
- write(fd, "$PMTK300,300,0,0,0,0*2F\r\n", 25) - Lê a cada 200ms;

Após enviar os comandos acima mencionados, o GPS irá atualizar e ler os dados a cada 200ms, enviando a GPRMC *NMEA Sentence*, que é a utilizada para efetuar a leitura dos dados de longitude, latitude e velocidade relativa ao solo. A leitura dos valores é efetuada com recurso à função *read()*, que guarda toda a informação lida na variável *char buf[255]*, que é retornada como parâmetro da função *getNMEA(char* buffer)* (ficheiro ConnectionHandler.cpp). Deste modo, a variável *buf* irá conter a seguinte cadeia de caracteres (por exemplo):

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7, 191194,020.3,E*68

O significado desta *string* está explicado no capítulo 2.3.2.

Como a informação é recebida numa *string* é necessária efetuar uma divisão dos dados. Para isto foi elaborado um algoritmo de *parsing* que consiste na separação dos diferentes dados. Como a informação está toda separada por vírgulas, o algoritmo foi bastante simples de elaborar. Este consistiu num ciclo *for* que, sempre que encontrava uma vírgula, era possível saber em que posição dos dados se encontrava. O algoritmo encontra-se na função *parseNMEA(char* buffer, char* latitude, char* longitude, char* velocidade, char* lat_letra, char* lng_letra)*, no ficheiro *ConnectionHandler.cpp*.

Após a utilização desta função, os valores retornados seguem o exemplo da seguinte figura 3.2:

```
Latitude (Graus e minutos decimais) - 4916.45 - tamanho: 7
Longitude (Graus e minutos decimais) - 12311.12 - tamanho: 8
Letra latitude - N - tamanho: 1
Velocidade (over the ground in knots) - 000.0 - tamanho: 5
```

Figura 3.2: Organização dos valores do GPS

Para uma análise mais completa recomenda-se a leitura do ficheiro *ConnectionHandler.cpp* que contém todo o algoritmo de comunicação UART entre o GPS e o BeagleBone Black, bem como o algoritmo de *parsing*.

3.1.2 Módulo Acelerómetro/Giroscópio

A comunicação efetuada entre o Acelerómetro/Giroscópio e o BeagleBone Black é realizada através de I2C, estando o BeagleBone a actuar como Mestre e o MPU-6050 como Escravo. O BeagleBone Black, como dispositivo Mestre, controla a rede à qual podem ser ligados inúmeros Escravos. Estes estão ligados ao *Serial Data* (SDA) - transmissão de informação bidirecional - e ao *Serial Clock* (SCL) - sincronização da transferência. O MPU-6050 envia uma trama de dados por I2C como resposta ao pedido efetuado pelo BBB com a informação proveniente dos registos de interesse, listados no documento "MPU-6000 and MPU-6050 Register Map and Descriptions - Revision 4.1" [12].

O BeagleBone Black dipõe de 3 *buses* I2C, dois deles acessíveis pelos *pin headers*, dispostos em conformidade com a tabela 3.3:

Tabela 3.3: I2C buses no BeagleBone Black

H/W Bus	S/W Device	SDA pin	SCL pin	Descrição
I2C0	/dev/i2c-0	N/A	N/A	Bus interno para o controlo HDMI
I2C1	/dev/i2c-2	P9_18	P9_17	General I2C bus. Desativado por padrão
I2C2	/dev/i2c-1	P9_20	P9_19	General I2C bus. Ativado por padrão

Tendo em conta que o I2C2 está ativo por padrão, foi então o escolhido para efetuar a ligação com o MPU6050.

O esquema elétrico que permite a conexão dos dois dispositivos está representado na figura 3.3:

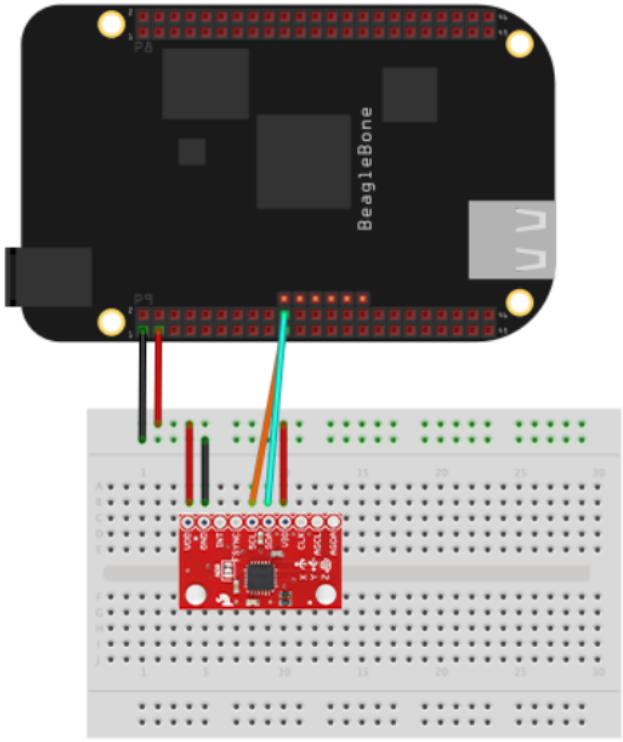


Figura 3.3: Esquema elétrico BeagleBone Black - MPU6050

As ligações entre o BeagleBone Black e o MPU6050 seguem a correspondência da seguinte tabela 3.4:

Tabela 3.4: Pinout do esquema elétrico BeagleBone Black e MPU6050

MPU6050	Pino BBB	Informação pino
VDD	P9_3	VDD_3V3
GND	P9_1	DGND
SCL	P9_19	I2C2_SCL
SDA	P9_20	I2C2_SDA
VIO	P9_3	VDD_3V3

Para confirmar que o dispositivo se encontra devidamente ligado, utilizou-se o programa *i2ctools* que foi obtido através da instalação via *sudo apt-get install i2c-tools*. Deste modo, executando o comando *sudo i2cdetect -r 1* que consiste na leitura dos registos associados ao *bus 1* do BeagleBone Black, permite confirmar se o dispositivo se encontra devidamente conectado como também o registo associado.

O resultado desta operação está representado na figura 3.4 , em que o registo 68 está reservado para o MPU6050.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figura 3.4: Mapa de registos I2C

Após validar a ligação é necessário configurar o dispositivo para, posteriormente, se efetuarem as comunicações entre o BeagleBone Black e o MPU6050. Para isso foi utilizada uma biblioteca chamda *I2CDevice* disponibilizada pelo Professor Derek Molloy (Dublin City University), que facilitou a comunicação entre os dois dispositivos. Esta biblioteca permite a inicialização de um dispositivo I2C, bem como o envio e receção de dados entre os dois dispositivos.

Inicialmente, foi definido o MPU6050 *i2c* através da função *I2CDevice i2c(1,0x68)* (ficheiro ConnectionHandler.cpp), em que o valor "1"corresponde ao *bus* e o valor "0x68"ao registo do MPU6050 (lido através do programa *i2cdetect* anteriormente referido). Posteriormente, foi necessário efetuar um *wake up* ao dispositivo através das duas seguintes funções:

- *i2c.open();* - função responsável por abrir a ligação I2C;
- *i2c.writeRegister(0x6B,0);* - função responsável pelo *wake up* do MPU6050, que é efetuado quando é escrito o valor "0"no registo "0x6B".

Após o processo de configuração estar completo, foi necessário criar uma função capaz de converter os valores lidos pelos registos do MPU6050 em ângulos. A função responsável por esta tarefa é a *getAngles(float* pitch, float* roll)* (ficheiro ConnectionHandler.cpp), que retorna os valores, em ângulos, do *pitch* e do *roll*, como parâmetros.

A função, inicialmente, faz a leitura dos valores do acelerómetro e do giroscópio nos devidos registos, que vêm constituídos, cada um, em dois bytes, *High* e *Low*:

$$accelxh = i2c.readRegister(0x3b); \tag{3.1}$$
$$accelxl = i2c.readRegister(0x3c); \tag{3.2}$$
$$accelyh = i2c.readRegister(0x3d); \tag{3.3}$$
$$accelyl = i2c.readRegister(0x3e); \tag{3.4}$$
$$accelzh = i2c.readRegister(0x3f); \tag{3.5}$$
$$accelzl = i2c.readRegister(0x40); \tag{3.6}$$
$$gyroxh = i2c.readRegister(0x43); \tag{3.7}$$
$$gyroxl = i2c.readRegister(0x44); \tag{3.8}$$
$$gyroyh = i2c.readRegister(0x45); \tag{3.9}$$

$$gyroyl = i2c.readRegister(0x46); \quad (3.10)$$
$$gyrozh = i2c.readRegister(0x47); \quad (3.11)$$
$$gyrozl = i2c.readRegister(0x48); \quad (3.12)$$

Uma vez que os valores vêm separados em 2 bytes, foi necessário efetuar a junção para uma variável de 16bits (2 bytes), que será utilizada para armazenar as forças g. As equações que se seguem garantem que, na passagem de um int8_t para um *float*, se mantém o sinal, uma vez que o valor recebido (valores *high*) estão em complemento para dois.

$$if(accelxh > 127) accelxh -= 256; \quad (3.13)$$
$$if(accelyh > 127) accelyh -= 256; \quad (3.14)$$
$$if(accelzh > 127) accelzh -= 256; \quad (3.15)$$
$$if(gyroxh > 127) gyroxh -= 256; \quad (3.16)$$
$$if(gyroyh > 127) gyroyh -= 256; \quad (3.17)$$
$$if(gyrozh > 127) gyrozh -= 256; \quad (3.18)$$
$$(3.19)$$

Quando o *Sensitivity Scale Factor* é igual a 0 (que o é por *default*), então 1g = 16384LSB, logo, sabendo a medida do acelerómetro em LSB, dividido o valor por 16384 obtém-se a aceleração em g. No caso do giroscópio, quando o valor do *Sensitivity Scale Factor* é igual a 0 (igualmente por *default*), então 1^o/s = 131LSB. Dividindo então por este valor, obtém-se os valores do giroscópio em graus por segundo.

$$accelx = \frac{(float)accelxh \times 256 + accelxl}{16384} \quad (3.20)$$
$$accely = \frac{(float)accelyh \times 256 + accelyl}{16384} \quad (3.21)$$
$$accelz = \frac{(float)accelzh \times 256 + accelzl}{16384} \quad (3.22)$$
$$gyrox = \frac{(float)gyroxh \times 256 + gyroxl}{131} \quad (3.23)$$
$$gyroy = \frac{(float)gyroyh \times 256 + gyroyl}{131} \quad (3.24)$$
$$gyroz = \frac{(float)gyrozh \times 256 + gyrozl}{131} \quad (3.25)$$

Após se obter a velocidade angular instantânea, é necessário multiplicar este valor pelo intervalo de tempo entre a medição atual e a anterior.

$$gyroXAngle = gyrox \times finish; \tag{3.26}$$
$$gyroYAngle = gyroy \times finish; \tag{3.27}$$
$$gyroZAngle = gyroz \times finish; \tag{3.28}$$

A medição temporal é efetuada através da função *finish = timer.stop();* que mede a diferença de tempo entre cada medição. Sempre que é efetuada uma iteração, é novamente inicializada a contagem através da função *timer.start();*

Segue-se então o cálculo dos ângulos propriamente ditos. Inicialmente, calcula-se o *pitch* e o *roll* através dos dados recolhidos pelo acelerómetro. Seguidamente são somados aos valores anteriormente calculados na iteração anterior, que foram calculados conjuntamente entre os dados do acelerómetro e giroscópio. No sistema em causa, o parâmetro *alpha* foi determinado por "tentativa-erro", tendo-se observado um bom desempenho para *coef* = 0.7.

$$acc\_pitch = \frac{180}{\pi} \times \arctan\left(\frac{accelx}{\sqrt{accely^2 + accelz^2}}\right); \tag{3.29}$$
$$acc\_roll = \frac{180}{\pi} \times \arctan\left(\frac{accely}{\sqrt{accelx^2 + accelz^2}}\right); \tag{3.30}$$
$$*roll \mathrel{+}= gyroXAngle; \tag{3.31}$$
$$*pitch \mathrel{+}= gyroYAngle; \tag{3.32}$$
$$*roll = *roll \times (1 - coef) + acc\_roll \times coef; \tag{3.33}$$
$$*pitch = *pitch \times (1 - coef) + acc\_pitch \times coef; \tag{3.34}$$

3.1.3 Pedal de acelerador, travão e velocidades de caixa

A medição dos pedais de acelerador, travão e das velocidades de caixa foi simulada devido à complexidade acrescida em efetuar a leitura diretamente dos pedais e na manete de velocidades no FEUP VEC. Deste modo a leitura da posição dos pedais e manete segue a seguinte estratégia:

- **Pedal de aceleração** - simulado por um potenciómetro;
- **Pedal de travão** - simulado por um botão de pressão.
- **Velocidade de caixa** - simuladas por 5 botões de pressão - 4 botões para as velocidades de 1 a 4 e o 5º botão para a marcha atrás (R).

O **pedal de aceleração** do FEUP VEC é simulado por um potenciómetro. Este encontra-se ligado a uma das portas do ADC (*Analog to Digital Converter*) do BeagleBone Black, de 12 bits, permitindo realizar 200.000 amostragens por segundo, mas estando limitado a uma tensão máxima de entrada de 1.8V. Dessa forma, o potenciómetro encontra-se ligado entre a massa e um pino com

a tensão de referência do ADC (1.8V), com o pino flutuante do potenciómetro ligado a uma das portas do ADC.

Segue-se na figura 3.5 o esquema elétrico do potenciómetro:

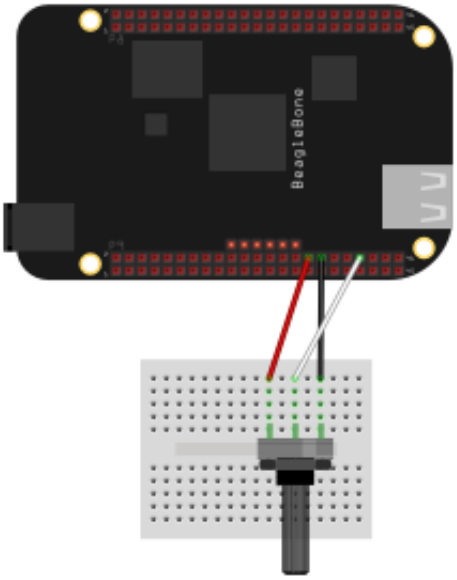


Figura 3.5: Esquema elétrico BeagleBone Black - Potenciómetro

Os pinos de ligação estão numerados na tabela 3.5.

Tabela 3.5: Pinout do esquema elétrico do acelerador via potenciómetro

Potenciómetro	Pino BBB	Informação pino
Terminal 1	P9_32	VDD_ADC
Terminal cursor	P9_40	AIN1
Terminal 2	P9_34	GNDA_ADC

O **pedal de travão** e as **velocidade de caixa** são simulados por botões de pressão. Estes botões, quando pressionados, ligam um respetivo pino GPIO (*General Purpose Input/Output*) do BeagleBone Black à tensão de rail superior, identificando que o travão se encontra cativo ou sinalizando a velocidade de caixa engrenada. O esquema elétrico da figura 3.6 representa as velocidades de caixa, a marcha atrás e o travão (através de botões de pressão). Os primeiros 4, de cima para baixo, representam as velocidades de caixa, da primeira até à quarta. Os restantes dois botões representam a marcha atrás e o travão respetivamente (de cima para baixo).

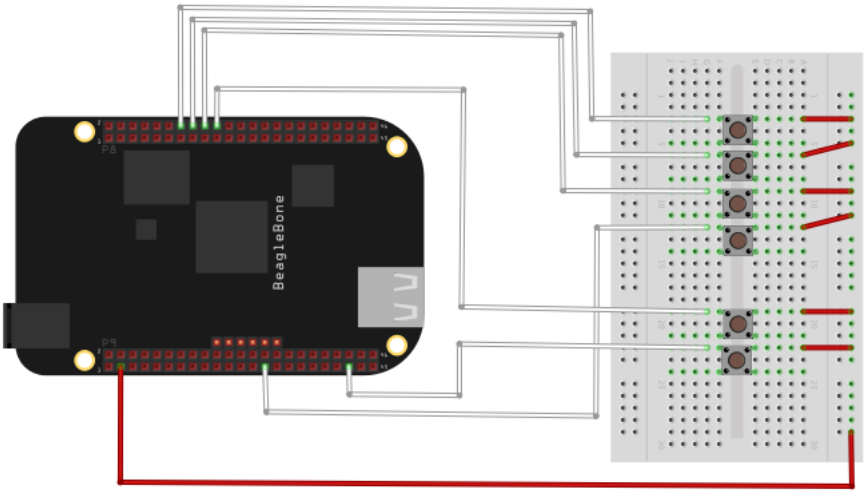


Figura 3.6: Esquema elétrico Velocidades de Caixa e Travão

Os pinos de ligação estão numerados na tabela 3.6.

Tabela 3.6: Pinout do esquema elétrico de velocidades de caixa e travão

Botão	Pino BBB	GPIO pin
Mudança 1	P8_14	GPIO_26
Mudança 2	P8_16	GPIO_46
Mudança 3	P8_18	GPIO_65
Mudança 4	P9_27	GPIO_115
Reverse	P8_20	GPIO_63
Travão	P9_41	GPIO_20

Para efetuar então a leitura dos valores foi utilizada uma biblioteca de funções disponibilizada pelo Professor Derek Molloy (Dublin City University) chamada de *GPIO.h*. Esta biblioteca permite efetuar leituras e escritas nos pinos GPIO (*General Purpose Input/Output*) de uma forma bastante simplificada. Inicialmente são atribuídas variáveis do tipo GPIO a cada sinal a ser lido: *GPIO Gear1(26), Gear2(46), Gear3(65), Gear4(115), Reverse(63), Brake(20);*, sendo que os valores numéricos representam os pinos utilizados. Por exemplo, o Gear1 é ligado ao pino P8_14 uma vez que está atribuído o GPIO_26 por *default*. A numeração dos pinos pode ser consultada no "*BeagleBone Black Pinout*".

Os GPIO podem ser configurados como Input ou Output. A função *setDirection();* (ficheiro ConnectionHandler.cpp) foi utilizada para este efeito. Neste caso todos os pinos são configurados como *Input*:

- Gear1.setDirection(INPUT);
- Gear2.setDirection(INPUT);
- Gear3.setDirection(INPUT);

- Gear4.setDirection(INPUT);
- Reverse.setDirection(INPUT);
- Brake.setDirection(INPUT);

Posteriormente, para efetuar a leitura da velocidade de caixa foi criada uma função *getGear();* (ficheiro ConnectionHandler.cpp) que é responsável por retornar o valor da posição da caixa. O seu funcionamento resume-se na leitura das 5 posições, e retorna um valor inteiro referente à posição. É de salientar que a posição de marcha atrás é representada pelo valor numérico de -1.

Relativamente ao valor do travão, foi utilizada uma função específica mas semelhante à utilizada para obter as posições da caixa de velocidade. A função *getBrake()* (ficheiro ConnectionHandler.cpp) retorna o valor 1 caso o travão esteja a ser pressionado, caso contrário retorna o valor 0.

Relativamente à leitura do pedal do acelerador, que no qual foi utilizada uma simulação por um potenciómetro, a abordagem foi diferente à de um GPIO. A leitura do valor simulado foi efetuada através do ADC de 12 bits do BeagleBone Black. A função responsável pela leitura do valor é a *getAccel()* que utiliza a função *readAnalog(int number)* (ficheiro ConnectionHandler.cpp) para ler diretamente do pino, em que o parâmetro *number* corresponde à numeração da porta do pino do ADC. Após ser efetuada a leitura diretamente do pino, é necessário dividir por 4095 uma vez que o ADC é de 12 bits. O sinal deste modo vem entre 0 e 1, pois representa a proporção da tensão lida em relação à tensão de *rail* do ADC. Finalmente, multiplicando o valor por 100, obtém-se o valor em percentagem, de 0 a 100, do pedal do acelerador simulado.

3.2 Comunicação Servidor/Cliente

Atendendo aos requisitos do sistema e aos possíveis casos de uso, a comunicação BeagleBone/*Base-Station* de dados de sensorização é realizada recorrendo a uma configuração Cliente/Servidor (figura 3.7). O Servidor, escrito em C++, que é executado no BeagleBone Black, responde a pedidos de informação por parte dos clientes com os dados a serem enviados.

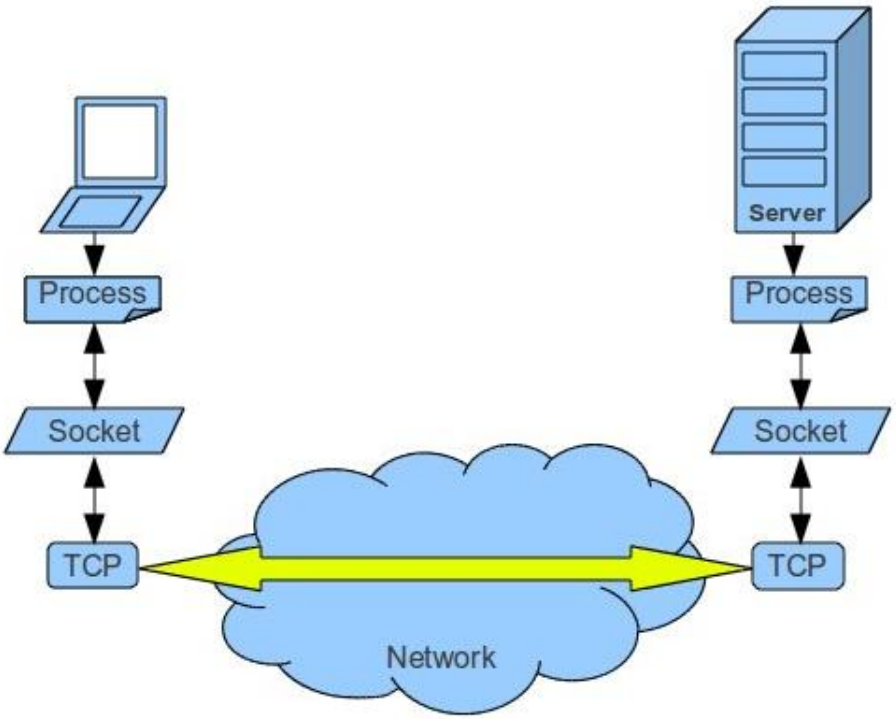


Figura 3.7: Comunicação via sockets

3.2.1 Sockets

A elaboração do código foi baseada num exemplo disponibilizado pelo Professor Derek Molloy (Faculty of Engineering and Computing, Dublin City University, Ireland), chamado de "threadedTemperatureServer" que consiste na criação de um servidor capaz de responder a pedidos de clientes. O funcionamento deste tipo de comunicação é baseado em *sockets* (C++), no qual o BeagleBone Black age como servidor e a *Base-Station* como cliente. Através do esquema que se segue, é possível entender o algoritmo responsável por esta comunicação:

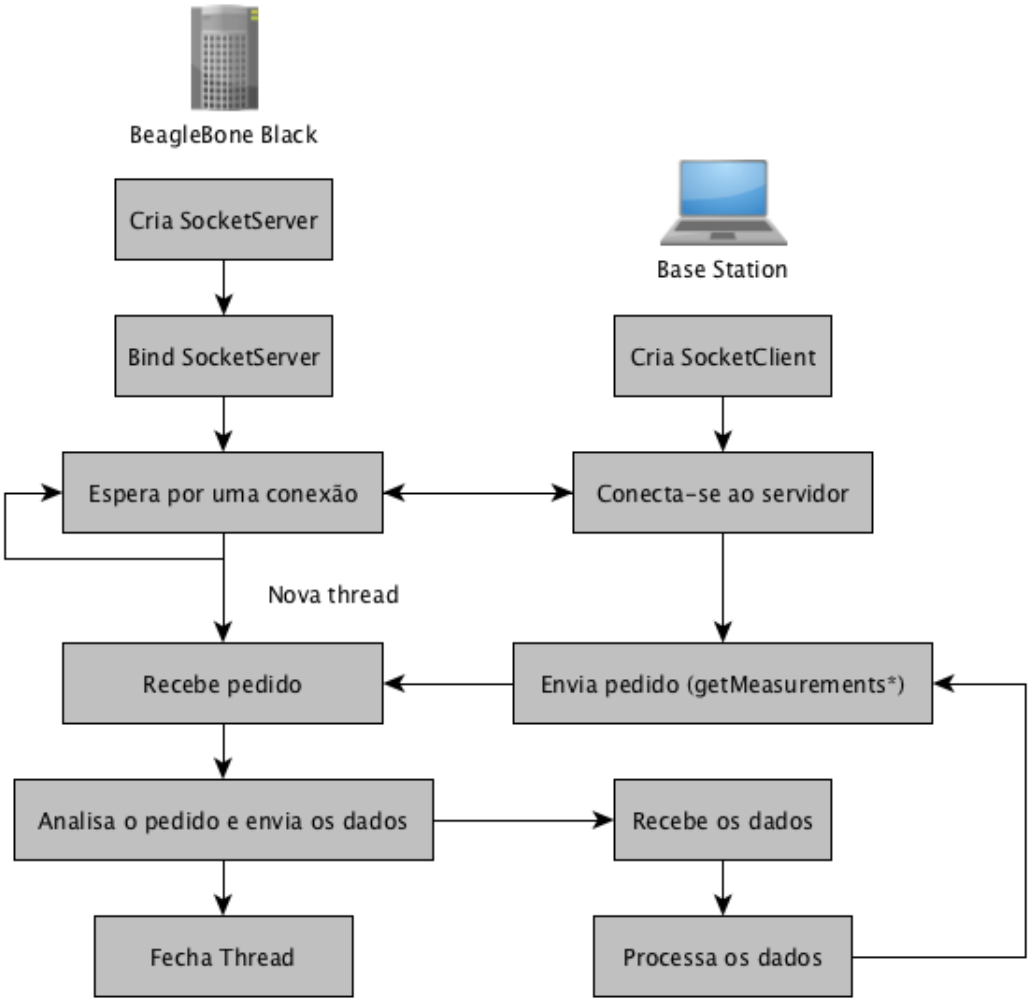


Figura 3.8: Diagrama do algoritmo SocketServer - SocketClient

Tal como o esquema da figura 3.8 sugere, o BeagleBone Black é responsável por criar um *SocketServer*, seguido de um *bind* que consiste na abertura do servidor numa porta de rede que esteja disponível. Após esta operação, o servidor espera continuamente que um cliente se conecte a ele. Deste modo, o cliente, que é criado também utilizando um *SocketClient*, conecta-se ao servidor destino. Após o cliente estar devidamente conectado com o servidor, envia um pedido de dados, que consiste numa mensagem chamada de "getMeasurements*". O servidor analisa esta mensagem, e se coincidir com "getMeasurements*", envia para o cliente os dados lidos pelos sensores no veículo.

Conforme sugere o diagrama, sempre que o servidor aceita um novo cliente, é criada uma nova *thread*. Isto é possível pois o servidor é do tipo *MultiThread*, isto é, tem a capacidade de executar várias *threads*. Esta consiste numa linha de execução, que neste caso está associada à comunicação individual do servidor para cada cliente. Sempre que o cliente recebe a informação, o servidor fecha a *thread* responsável pela comunicação com este. Os ficheiros responsáveis pela utilização dos *sockets* são *server.cpp, SocketServer.cpp e ConnectionHandler.cpp* (e respetivas bibliotecas).

3.2.2 Envio dos dados para o cliente

Conforme referido no diagrama da figura 3.8, o pedido *getMeasurements** por parte de um cliente resulta no envio dos dados do lado do servidor. Sempre que é inicializada uma nova *thread*, é utilizada uma biblioteca chamada *ConnectionHandler* (disponibilizada também pelo Professor Derek Molloy [6]) responsável pelo processamento da leitura dos dados. O seu funcionamento resume-se à leitura da mensagem recebida pelo cliente, e, consoante o pedido, são enviados para o cliente os dados numa *string*. Caso a mensagem enviada pelo cliente seja *getMeasurements**, o servidor invoca uma série de funções responsáveis pela aquisição de todos os dados.

Todo o processo é executado na função *threadLoop()* (ficheiro ConnectionHandler.cpp). O funcionamento desta resume-se a uma ordem sequencial de instruções, descritas na figura 3.9:

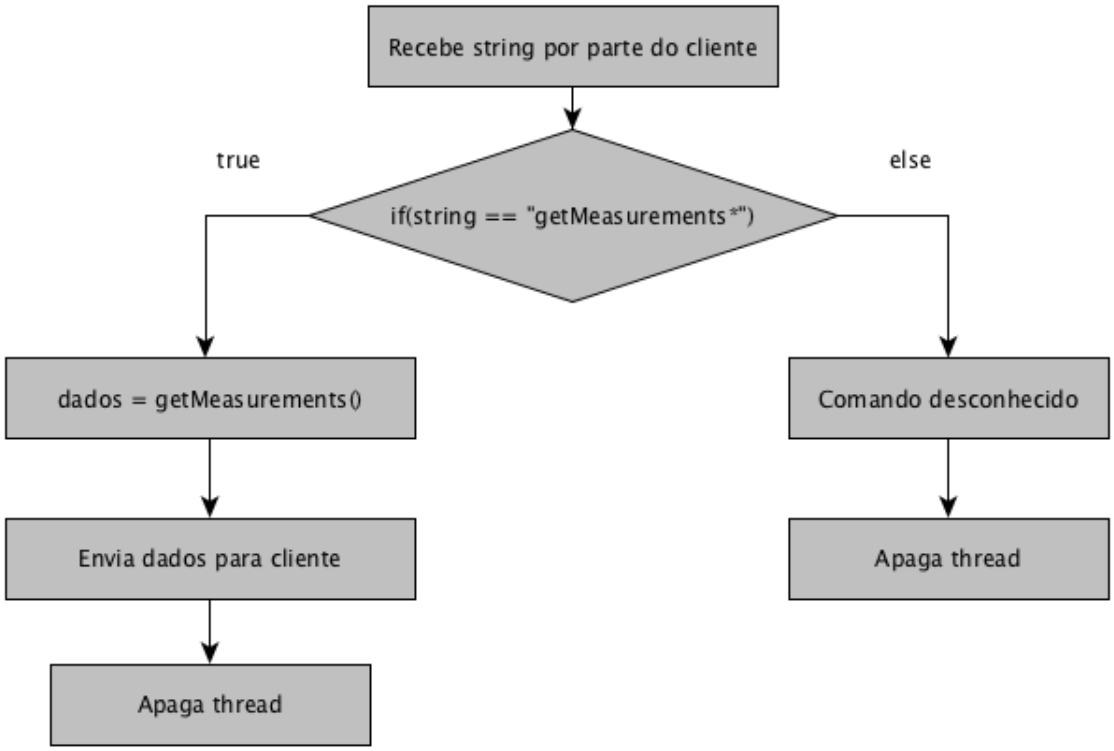


Figura 3.9: Diagrama do algoritmo - getMeasurements

A função *getMeasurements(void)* (ficheiro ConnectionHandler.cpp) retorna uma string, contendo todos os dados relativos aos sensores de GPS e Acelerómetro/Giroscópio. A *string* é constituída pela seguinte ordem:

- **1º** - velocidade de caixa;
- **2º** - pedal do travão;
- **3º** - pedal do acelerador;

- **4º** - ângulo *roll*;
- **5º** - ângulo *pitch*;
- **6º** - latitude;
- **7º** - letra latitude (N=*North*, S=*South*);
- **8º** - longitude
- **9º** - letra longitude (E=*East*, W=*West*);
- **10º** - velocidade (*knots*);

As funções responsáveis pela obtenção dos dados acima referidos são:

- **1º** - getGear(); - retorna o valor da velocidade de caixa;
- **2º** - getBrake(); - retorna o valor do pedal do travão;
- **3º** - getAccel(); - retorna o valor do pedal de aceleração;
- **4º** - getAngles(&pitch, &roll); - retorna o valor dos ângulos *pitch* e *roll*;
- **6º** - getNMEA(buffer); e parseNMEA(buffer, latitude, longitude, velocidade, lat_letra, lng_letra); - retorna os valores relativos ao GPS.

Todas as funções acima mencionadas encontram-se no ficheiro ConnectionHandler.cpp.

3.3 Comunicação sem fios entre o FEUP VEC e a Base-Station

A comunicação entre o BeagleBone Black e a *Base-Station* foi realizada por intermédio de dois dispositivos capazes de comunicar sem fios. Os Ubiquiti Bullet M2-HP foram ligados diretamente, via *Ethernet*, ao PC da *Base-Station* e ao BeagleBone Black. Graças às suas possibilidades de configuração, foi possível criar uma rede *wireless* que permitiu a comunicação entre os dois pontos.

3.3.1 Topologia da rede

Para a implementação do sistema de comunicação sem fios foi utilizada a estrutura representada na figura 3.10:

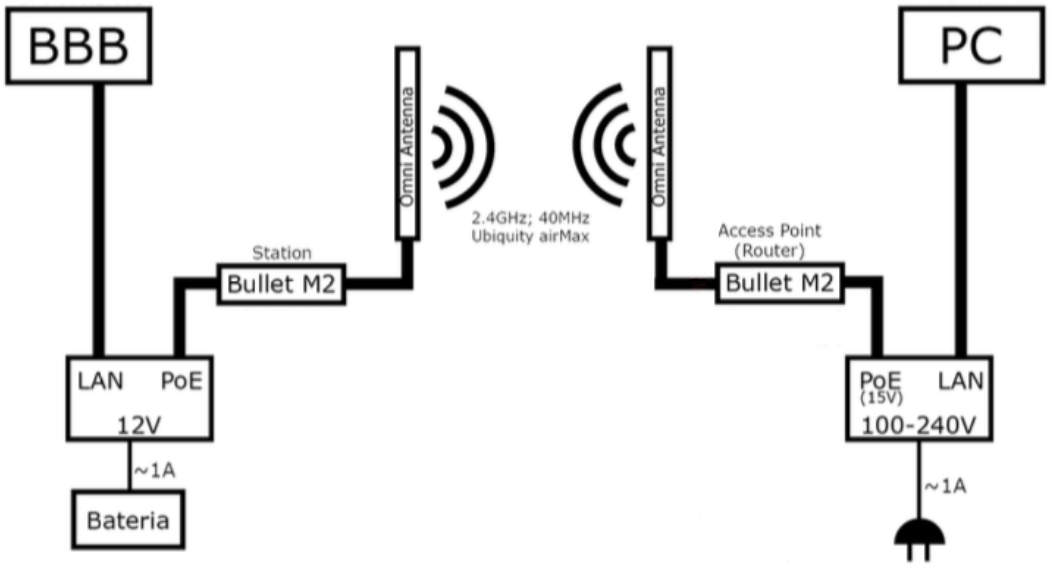


Figura 3.10: Topologia da rede wireless

O lado esquerdo do esquema representa o FEUP VEC, sendo o lado direito a *Base-Station*. Como a figura 3.10 sugere, tanto o BeagleBone Black como o PC na *Base-Station*, foram utilizados POE (*Power Over Ethernet*) de modo a permitir a transmissão de energia juntamente com os dados necessários, uma vez que os Bullet M2 requerem uma fonte de alimentação para funcionarem, que neste caso é via *Ethernet*. No caso do lado do FEUP VEC, o POE utilizado foi um que é alimentado a 12V, de modo a se aproveitar a energia proveniente das baterias do veículo. Uma vez que o projeto ainda não passou de um protótipo funcional, a alimentação do Bullet M2 do lado do FEUP VEC foi realizada através de uma bateria/fonte de alimentação de 12V 1A. No lado da *Base-Station*, utilizou-se um POE alimentado entre 100-240V uma vez que a energia utilizada é a da rede.

As antenas responsáveis pela transmissão e receção de sinal são as *Outdoor 8 dBi Omni-Directional Antenna EAG-2408* (figura 3.12). Tal como o nome sugere, dispõem de um ganho de 8dBi e são omni-direcionais, isto significa que são capazes de enviar/receber sinal em qualquer direção. Horizontalmente, cobre os 360^{o}, mas na vertical cobre apenas 18^{o}, tal como demonstra a figura 3.11:

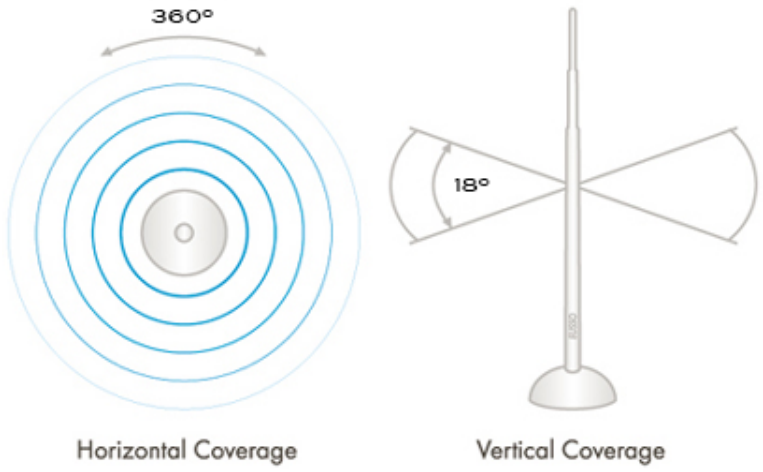


Figura 3.11: Graus de propagação das antenas

Figura 3.12: Outdoor 8 dBi Omni-Directional Antenna EAG-2408

Os Ubiquity Bullet M2 permitem a comunicação até cerca de 50Km, mas este valor é drasticamente alterado devido às antenas em utilização. Foi utilizada uma calculadora (disponibilizada no site http://hobbywireless.com/) que permite calcular um valor aproximado, em linha de vista, isto é, sem obstáculos, do alcance máximo da comunicação. Utilizando os valores disponibilizados pelos *datasheets* dos Ubiquity Bullet M2 [23] e das antenas, o valor estimado resultou em cerca de 2.8 Km, valor este que cumpre o requisito inicial de 2 Km.

Por outro lado, o conjunto Ubiquity Bullet M2 e antena omnidirecional contribuem com um peso de 0.18 Kg + 0.370 Kg = 0.55 Kg. Somados aos 0.312 Kg do 4DCAPE-70T, aos 0.039 Kg do BeagleBone Black e desprezando o peso dos cabos e dos sensores, é cumprido o requisito de peso máximo fixado nos 2 Kg.

Figura 3.13: Conjunto Bullet M2 e Antenas Omnidirecionais

3.3.2 Configuração da rede

O *router* é responsável por definir e atribuir endereço aos dispositivos que se ligam à rede, funcionando como um ponto de acesso. Todos os equipamentos que se pretendam ligar à rede devem fazê-lo ligando-se a este ponto de acesso, cujo SSID (*Service Set Identifier*), neste caso, é "VEC_AP". Uma vez criada a ligação, é possível interagir com a rede como se de uma *home network* se tratasse. Uma vez que o *router* é responsável pela gestão dos endereços na rede, este divide os endereços em duas sub-redes, "192.168.1.x" para ligações por cabo, e "198.168.2.x" para ligações por *wireless*, como indicado na figura 3.14. Dispositivos conectados por cabo, podem estar configurados com qualquer endereço IP, desde que esteja contido na sub-rede correta. No entanto, as aplicações utilizadas neste projeto, estão configuradas assumindo que o dispositivo utilizado na *Base-Station* é acessível através do endereço IP "192.168.1.144". Esta configuração pode ser alterada, se necessário.

Do lado da *station*, o dispositivo Bullet M2 apenas precisa de procurar uma rede cujo SSID e enderesso MAC correspondam aos pré-determinados, podendo então ligar-se à mesma. Uma vez que o *router* está configurado para permitir a utilização de DHCP (Dynamic Host Configuration Protocol), o BeagleBone deve estar configurado para obter o seu endereço de IP utilizando esse método. De forma a garantir consistência entre vária utilizações desta rede, o servidor DHCP foi configurado de forma a apenas permitir uma ligação, e atribuir-lhe o endereço "192.168.2.101". Quanto à comunicação entre antenas, tal como definido previamente, utiliza-se uma portadora de 2.4GHz, atribuindo 40MHz de banda a cada canal. Está também a ser utilizada a tecnologia *airMax*, que consiste numa modulação de sinal para acessos múltiplos, proprietária da *Ubiquity*.

Esta modulação, baseada em TDMA (*Time Divison Multiple Access*), aumenta em cerca de 30% o débito suportado pelo canal. No entanto, isto impede que dispositivos que não suportem a esta tecnologia se liguem à rede. Esta é uma opção configurável, que pode ser desativada em caso de necessidade.

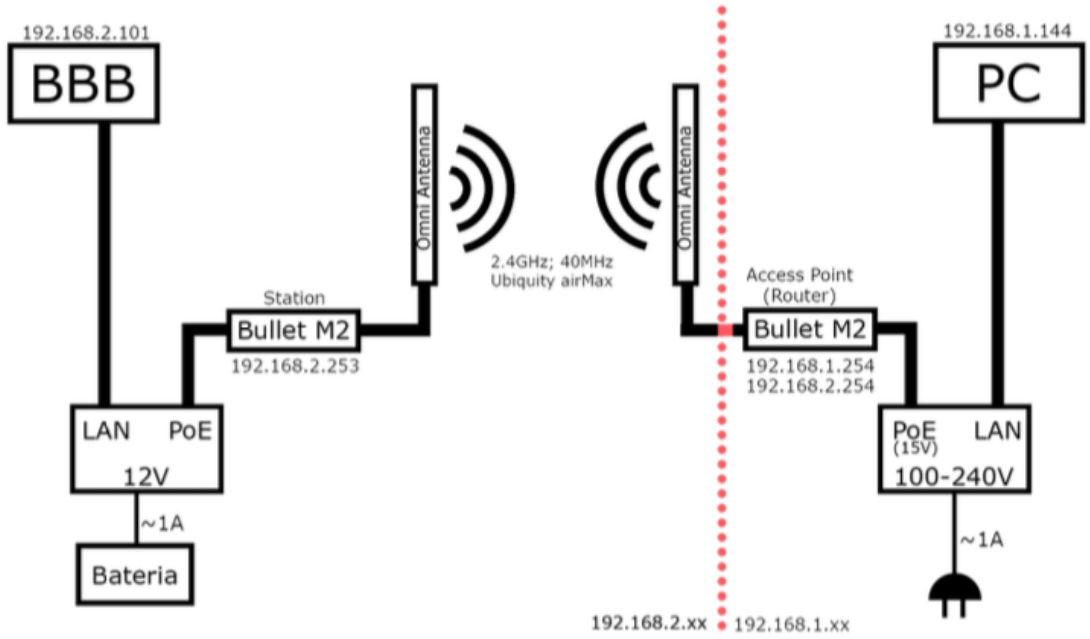


Figura 3.14: Topologia da rede com respetivos endereços

Para o processo de configuração dos Bullet M2 foi utilizado o *AirOS*, uma espécie de sistema operativo da Ubiquity para os dispositivos da marca. Graças ao *AirOS* [24], todo o processo de configuração foi elaborado de forma simples, adequando o funcionamento dos dois aparelhos em concordância com a rede pretendida.

3.4 Aplicação JAVA na Base-Station

De modo a monitorizar o comportamento do veículo em pista, do lado da *Base-Station*, foi desenvolvida uma interface em JAVA capaz de representar os diversos valores provenientes dos sensores. A interface é composta por duas *frames*: a primeira é responsável por, de forma organizada e intuitiva, efetuar a leitura dos valores dos ângulos *pitch* e *roll*, da velocidade de caixa engrenada e da velocidade instantânea do FEUP VEC; a segunda *frame* consiste na representação da posição atual do FEUP VEC num mapa.

3.4.1 Aquisição e representação dos dados

Estando o BeagleBone Black a executar um servidor TCP, do lado do JAVA foi apenas necessário implementar um cliente TCP capaz de comunicar com o FEUP VEC. Já descrito anteriormente, caso o servidor receba uma mensagem *getMeasurements**, automaticamente envia um pacote com todos os dados atuais relativos ao veículo. O JAVA é composto por 3 classes que possuem tarefas distintas:

- **TCPClient** - classe responsável pela inicialização das duas janelas (leitura de dados e posicionamento do FEUP VEC), responsável por efetuar a conexão com o servidor no BeagleBone Black, pela recolha dos dados provenientes do veículo e pela atualização periódica para representação dos dados;
- **GUI** - classe responsável pela elaboração da janela de leitura dos valores dos ângulos *pitch* e *roll*, da velocidade de caixa engrenada e da velocidade instantânea do FEUP VEC;
- **Mapa** - classe responsável pelo *plot* do mapa.

O algoritmo de funcionamento de todo o JAVA está representado na figura 3.15.

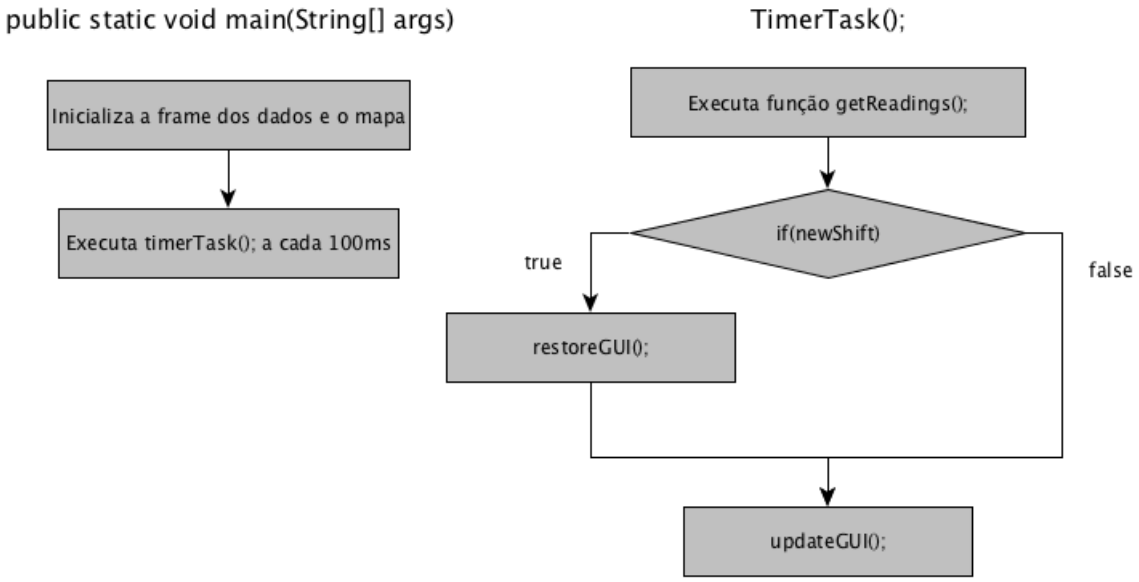


Figura 3.15: Diagrama do algoritmo base - JAVA

Tal como a figura 3.15 sugere, a função *main* é a função base de todo o JAVA. A sua principal função é invocar uma tarefa a cada 100ms, de modo a posteriormente se fazerem as leituras e *updates* nas respetivas *frames*.

A função *getReadings();* (ficheiro TCPClient.java) é a que efetua a comunicação com o servidor no BeagleBone Black e que, seguidamente, recebe todos os valores provenientes do FEUP VEC. A comunicação e receção dos dados segue a seguinte ordem:

- **1º** - cria um SocketClient que se conecta ao BeagleBone Black através do IP deste ("192.168.2.101", 5555);

- **2º** - é enviado para o servidor a mensagem "*getMeasurements**";

- **3º** - recebe a resposta do servidor, que consiste numa *string* com todos os dados (separados por um espaço entre eles);

- **4º** - é efetuada a separação de todos os dados e atribuídos às variáveis respetivas no JAVA;

- **5º** - é atualizada a posição do marcador que sinaliza o FEUP VEC no mapa (que será explicado mais à frente).

As funções *restoreGUI();* e *updateGUI();* (ficheiro TCPClient.java) têm como objetivo atualizar os valores da *frame* que contém os dados relativos aos pedais, à velocidade do FEUP VEC, à velocidade de caixa e ângulos *pitch* e *roll*, após terem sido atualizados pela função *getReadings();*.

Relativamente aos dados recolhidos, foi necessário efetuar a conversão da velocidade, que é recebida em *knots*, para Km/h. Sabendo que 1 nó equivale a 1,852 km/h, então o valor de velocidade foi multiplicado por 1.852.

Por outro lado, foi necessário converter a latitude e longitude de Graus e minutos decimais (DMM) para Graus decimais (DD), uma vez que o mapa utilizado apenas efetua leituras no formato DD. A conversão é efetuada através do seguinte cálculo:

$$GrausDecimais = Graus + \frac{minutos}{60} \tag{3.35}$$

Por exemplo, caso as coordenadas sejam Latitude = 4110.709 e Longitude = 835.7417, a conversão para Graus Decimais resulta em Latitude = 41.1784833333 e Longitude = 8.595695.

Outro parâmetro importante para a localização do FEUP VEC no mapa são as letras N (norte), S (sul), E (este) e W (oeste). Caso o valor da latitude venha junto com a letra N, o valor mantém-se positivo, caso a letra seja S, é necessário converter o valor da latitude para negativo, uma vez que representam o hemisfério norte e hemisfério sul. O mesmo acontence para a longitude. Neste caso, se a letra for E, o valor da longitude é convertido em valor negativo. Este tipo de conversão deve-se ao facto de o mapa que será utilizado (*Google Maps JxMaps*) necessitar de coordenadas no formato DD.

A *frame* que demonstra os valores dos sensores em questão está representada na figura 3.16:

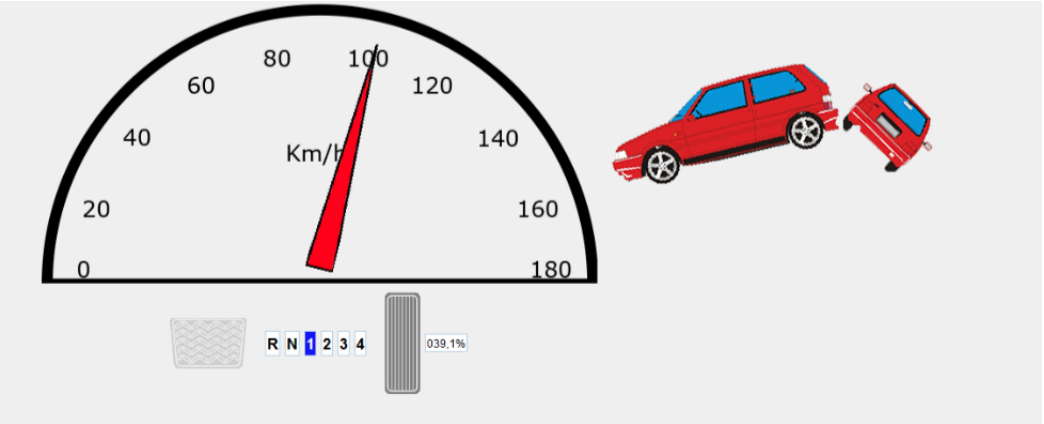


Figura 3.16: Frame de leitura dos diversos sensores do FEUP VEC

3.4.2 Sistema de rastreamento do FEUP VEC

Para projetar o veículo em tempo real num mapa, foi utilizado o JxMaps da TeamDev que consiste numa versão completa da Google Maps API escrito em JAVA [25].

De modo a incorporar o JxMaps no JAVA, apenas se incluiram ao projeto os ficheiros JAR (*Java Archive*) da JxMaps, que são arquivos compactados que contêm um conjunto de classes JAVA, que utilizadas em conjunto, permitiram reproduzir uma janela com o Google Maps. Para além destes ficheiros, foi necessário requisitar uma licença de utilização (1 mês, que pode ser prolongada), que neste caso foi para motivos académicos.

A classe "Mapa"é a responsável pela configuração da janela em si, enquanto que a atualização do marcador (que segue o FEUP VEC) que representa o veículo é feita na função *getReadings();*.

O algoritmo de atualização da posição do FEUP VEC é efetuado em conjunto com a função *getReadings();*. Sempre que são lidos novos dados relativos ao veículo, a variável responsável pela posição do FEUP VEC (VEC) é atualizada executando a seguinte instrução:

- *VEC.setPosition(new LatLng(myMap.mapa, gui.lat, gui.lng));*

Em que *gui.lat* e *gui.long* representam a latitude e longitude do FEUP VEC respetivamente.

Na figura 3.17 é possível observar um exemplo de uma localização do FEUP VEC através da *frame* responsável pelo rastreamento. O marcador vermelho simboliza o veículo, que é atualizado sempre que é efetuada uma nova leitura dos dados. Tal como a figura demonstra, é possível alterar o fundo do mapa entre "Mapa"ou "Satélite"e aumentar ou diminuir o *zoom* manualmente.

As definições que configuram o centro do mapa e o *zoom* inicial, estão descritas na classe Mapa do projeto JAVA.

Figura 3.17: Frame de rastreamento do FEUP VEC no Google Maps

3.5 Aplicação Qt no FEUP VEC

Para o condutor ter acesso aos dados do FEUP VEC em tempo real, foi elaborada uma aplicação em Qt que permitiu o a monitorização do comportamento do veículo em pista. Os dados a serem lidos consistem nos dos ângulos *pitch* e *roll*, velocidade de caixa, velocidade relativa ao solo e os pedais do acelerador e travão. Com o auxílio do *cape* 4DCAPE-70T da 4D Systems, projetou-se no monitor de 7 polegadas uma interface simples capaz de monitorizar os dados anteriormente referidos.

3.5.1 Aquisição dos dados

A aquisição dos dados na aplicação Qt foi semelhante à efetuada no programa JAVA anteriormente referido. Uma vez que o BeagleBone Black corre num servidor que responde a pedidos *getMeasurements** via TCP, a aplicação Qt baseou-se na criação de *sockets* que funcionam como clientes.

Desde modo, são efetuados pedidos periódicos ao servidor de modo a receber o pacote de dados relativo aos valores dos sensores. Após ser recebido o pacote de dados, é efetuado um algoritmo simples de *parsing* de modo a atribuir cada valor à respetiva variável no programa.

A escrita do código foi elaborada em C++, estando disponível para consulta em anexo.

O algoritmo de aquisição dos dados segue o esquema da figura 3.18.

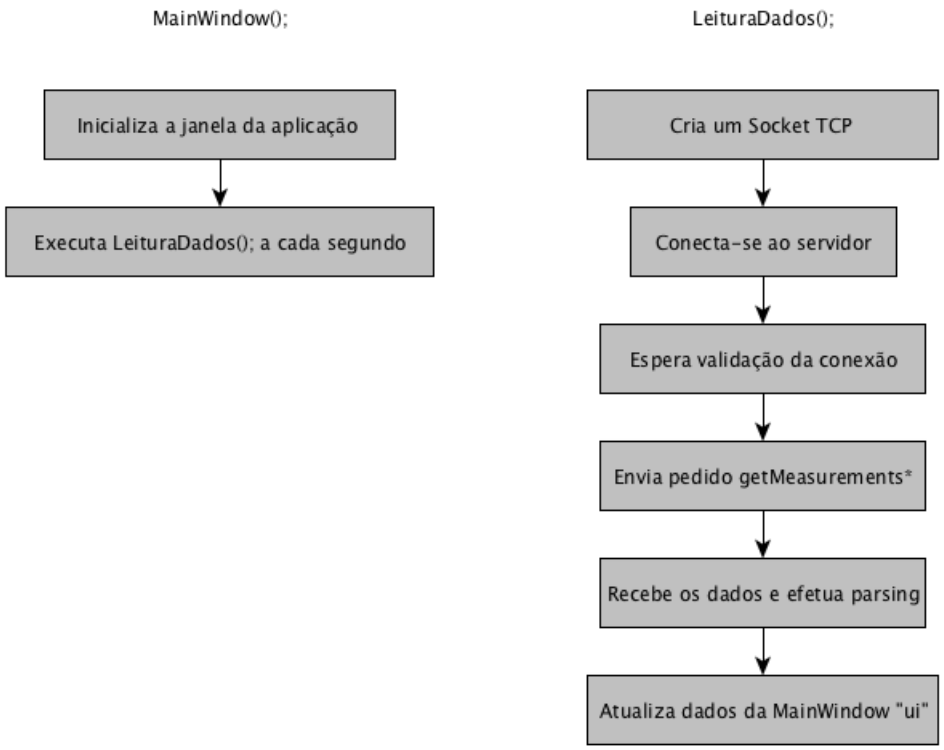


Figura 3.18: Diagrama do algoritmo - receção dados Qt via TCP Socket

Tal como o diagrama da figura 3.18 sugere, a função base da aplicação Qt é a *MainWindow()*. A sua função consiste na execução da função *LeitudaDados();* de segundo a segundo, que corresponde à periodicidade do algoritmo de requisição dos dados provenientes do FEUP VEC.

Cada vez que esta função é invocada, é criado um *socket* que posteriormente se conecta ao servidor que corre no BeagleBone Black. Após o êxito na conexão, é efetuado o pedido *getMeasurements** ao no qual é dada uma resposta com os dados provenientes dos sensores. Seguidamente, como os dados são recebidos numa *string* e separados por espaços, é efetuado um algoritmo de *parsing* e são atribuídos os valores dos sensores às respetivas variáveis no programa.

3.5.2 Representação dos dados

Após a receção dos dados dos sensores, é necessário representar-los numa janela. O Qt Creator (IDE utilizado) permite criar uma *frame*, que neste contexto se chama *Form*, à qual no projeto se designa por *mainwindow.ui*. A elaboração desta *frame* consistiu num processo bastante simples, uma vez que o IDE disponibiliza uma ferramenta bastante intuitiva de edição gráfica. Foram adicionados vários *displays* numéricos, alguns *labels* e uns *slides* de modo a poder representar o valor dos sensores do modo mais intuitivo possível.

A *frame* final está representada na figura 3.19:

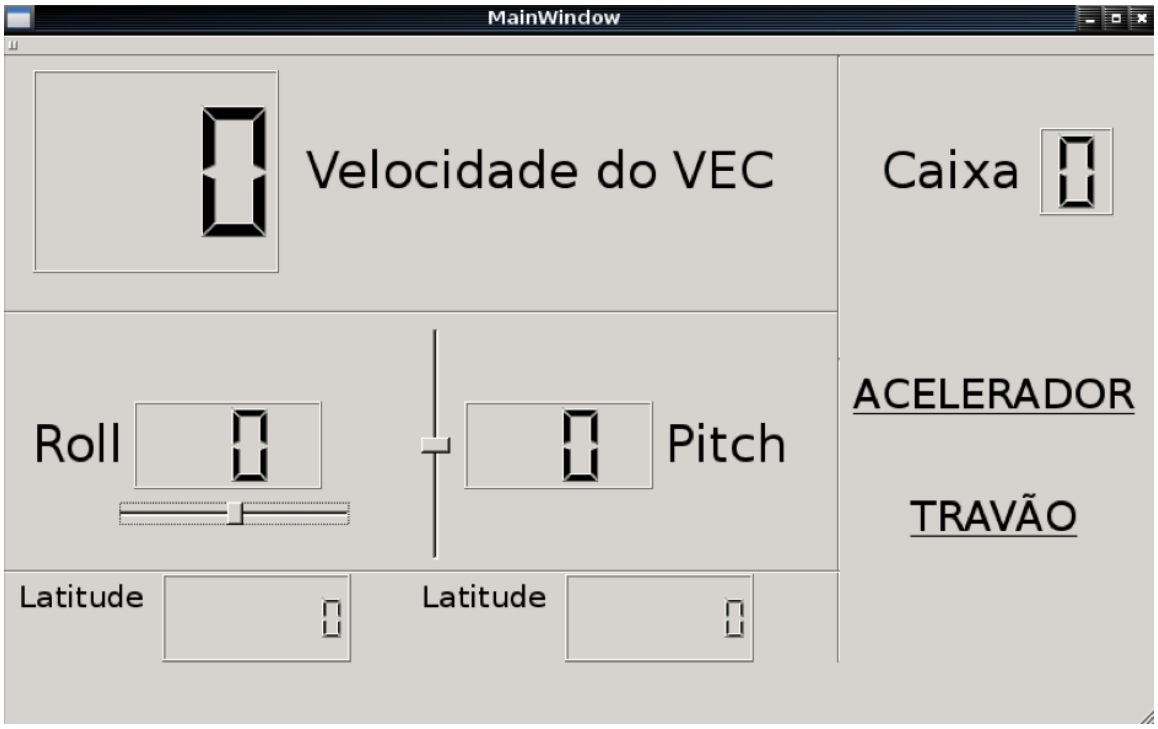


Figura 3.19: Frame de leitura dos valores no ecrã do FEUP VEC

A *frame* está dividida em 4 secções:

- **1º** - velocidade do FEUP VEC - nesta secção encontra-se um *display* responsável por representar o valor da velocidade atual do veículo, em km/h;
- **2º** - ângulos *pitch* e *roll* - neste caso é igualmente efetuado um *display* para o valor de cada um dos ângulos, bem como uma *scroll bar* para cada um dos ângulos, variando linearmente a sua posição entre -90+ e +90º.
- **3º** - latitude e longitude - esta secção é composta por dois *displays* que representam os valores de latitude e longitude em DMM;
- **4º** - velocidade de caixa, acelerador e travão - secção responsável por indicar a velocidade de caixa atual e, caso esteja o condutor a acelerar ou travar, é indicado através do texto correspondente.

3.6 Captação de vídeo no VEC

3.6.1 Aquisição e envio do sinal de vídeo pelo BeagleBone Black

A aquisição e transmissão de vídeo recorre a três aplicações distintas:

- **Video4Linux2 (V4L2)** - Drivers de *webcam* para Linux, que permite uma melhor interação e captura de vídeo através de câmaras USB;

- **capture.c** - Código C de captura de vídeo, originalmente desenvolvido por Derek Molloy, e posteriormente adaptado de modo a cumprir os requisitos do projeto. Recorre à *driver* V4L2 para comunicar com a câmara.
- **Avconv** - Software de *streaming* de conteúdo utilizado para encaminhar o *output* do programa "capture.c" para o destinatário.

De modo a inicializar os processos acima referidos, é necessário executar o seguinte ficheiro:

- *./streamVideoRTP*.

A figura 3.20 demonstra a utilização dos dois protocolos de comunicação entre o FEUP VEC e a *Base-Station*. No caso da transmissão de vídeo em tempo real, é utilizado o protocolo RTP. Relativamente ao envio dos dados provenientes dos sensores é então utilizado o protocolo TCP.

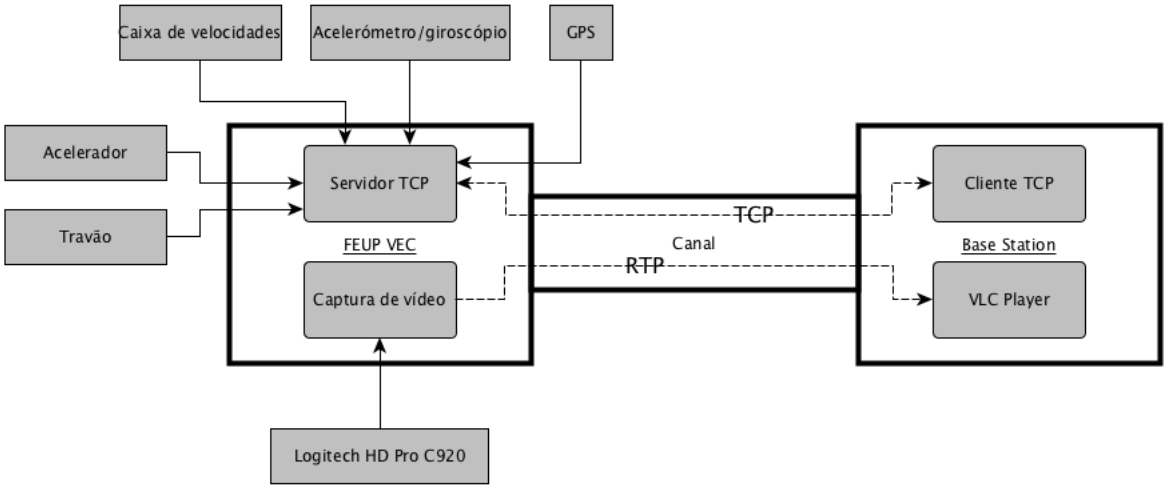


Figura 3.20: Protocolos de comunicação entre o FEUP VEC e a Base Station

3.6.2 Reprodução do vídeo na Base-Station

No dispositivo recetor da *stream* de vídeo, neste caso um PC, utilizou-se o VLC (Video LAN Codec), que é um *software* que permite a reprodução de *streams* RTP. A utilização de RTP implica que o recetor possua um ficheiro .sdp, gerado à priori, que contém todas as informações necessárias para aceder à *stream* de vídeo. Sempre que é executado o ficheiro *./streamVideoRTP* são enumeradas várias informações, estando então incluídas as mesmas responsáveis pela configuração da *stream* no dispositivo recetor.

Após o ficheiro .sdp estar configurado e o envio da *stream* estar a ser executada no BeagleBone Black, apenas é necessário abrir o ficheiro no VLC e o vídeo será automaticamente reproduzido.

Capítulo 4

Testes e resultados

O presente capítulo tem como objetivo descrever os testes e os resultados obtidos relativamente à implementação do projeto propriamente dito. Serão abordadas os métodos de como foram elaborados os testes, bem como a respetiva interpretação dos resultados obtidos. Serão também abordados os problemas que foram surgindo e a respetiva solução, quando possível.

4.1 Testes de envio dos valores sensoriais

O teste do envio dos diversos valores provenientes dos sensores consistiu numa transmissão dos dados via TCP. Para isso foi montado todo o sistema (incluindo as antenas) para se efetuar o teste de comunicação entre si. O teste consistiu num envio dos dados a diferentes periodicidades de transmissão conforme apresentado na tabela 4.1. O período da requisição de pedidos é definido no programa JAVA, tendo em conta que é na *Base-Station* que é comandado o processo de pedidos de leitura.

Tabela 4.1: Validação dos pedidos TCP com diferentes periodicidades

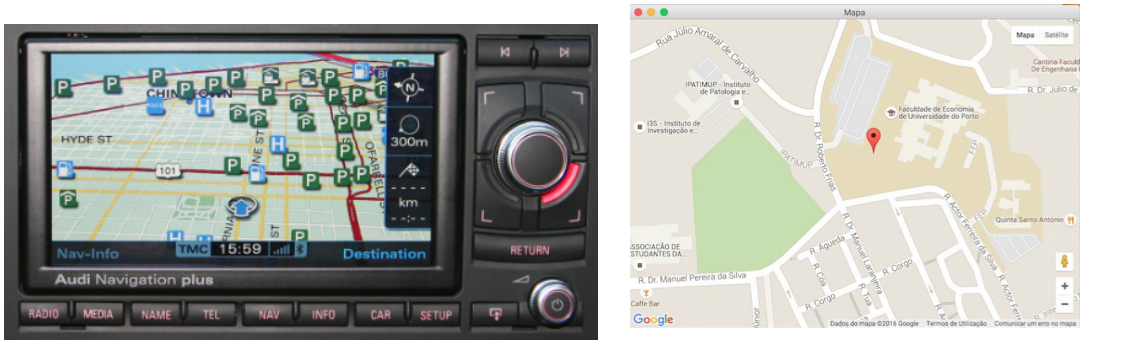
Período	**Verificação**
600ms	Aprovado
300ms	Aprovado
200ms	Aprovado*
100ms	Aprovado*

Conforme os valores da tabela 4.1 apenas nos períodos de medição de 600ms e 300ms se foram obtidos resultados com 100% de eficácia. Relativamente aos 200ms e 100ms, por vezes verificaram-se falhas na receção dos valores, pois o dispositivo de GPS atualiza os dados a cada 200ms (10Hz), entrando por vezes em conflito.

4.2 Testes módulo GPS

O teste do módulo de GPS consistiu na realização de um percurso com um automóvel de modo a validar a veracidade dos resultados obtidos relativamente à latitude, longitude e velocidade instantânea. O teste foi elaborado com um veículo comum em que o percurso consistiu numa volta ao parque da Faculdade de Engenharia da Universidade do Porto.

Para isto foi então utilizado um computador portátil que funcionou como cliente do servidor TCP implementado no BeagleBone Black. Para validar os resultados obtidos por parte do sistema, foi efetuada uma análise comparativa entre o GPS do veículo (Audi Navegation Plus <-> RNS-E, sistema desenvolvido pela Audi) e o GPS do sistema telemétrico. Uma vez que os dois sistemas permitem a reprodução da localização em tempo-real do veículo num mapa, a sua comparação permitiu a validação do sistema de telemetria para o FEUP VEC relativamente ao sinal de GPS.



(a) Audi Navegation Plus <-> RNS-E

(b) Frame de rastreamento do FEUP VEC no Google Maps

Figura 4.1: Validação do sistema de navegação do FEUP VEC

Relativamente ao valor da velocidade, foi efetuada uma comparação com os valores que o velocímetro do veículo apresentava. ao longo do percurso.

Os resultados finais foram obtidos com sucesso.

4.3 Testes módulo acelerómetro/giroscópio

A validação dos ângulos de *pitch* e *roll* provenientes do SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050 foi efetuada visualmente. Para a leitura dos valores dos ângulos o acelerómetro/giroscópio foi colocado o sensor numa *breadboard* e, posteriormente foram efetuados movimentos espaciais que permitiram efetuar uma leitura aproximada dos valores dos ângulos de *pitch* e *roll*. Quando a *breadboard* era inclinada na horizontal, era esperada uma variação entre -90º e +90º do valor do *roll*, valores estes que se verificaram aquando do movimento. Na figura 4.2 está representado um valor de -88.0222º do ângulo *roll* quando a *breadboard* se encontrava na posição da figura 4.3:

```
root@beaglebone:~# echo "getMeasurements*" | netcat 192.168.7.2 5555
0 0 0 -88.0222 1.96274 4916.45 N 12311.12 W 000.0
```

Figura 4.2: Teste valor do ângulo roll

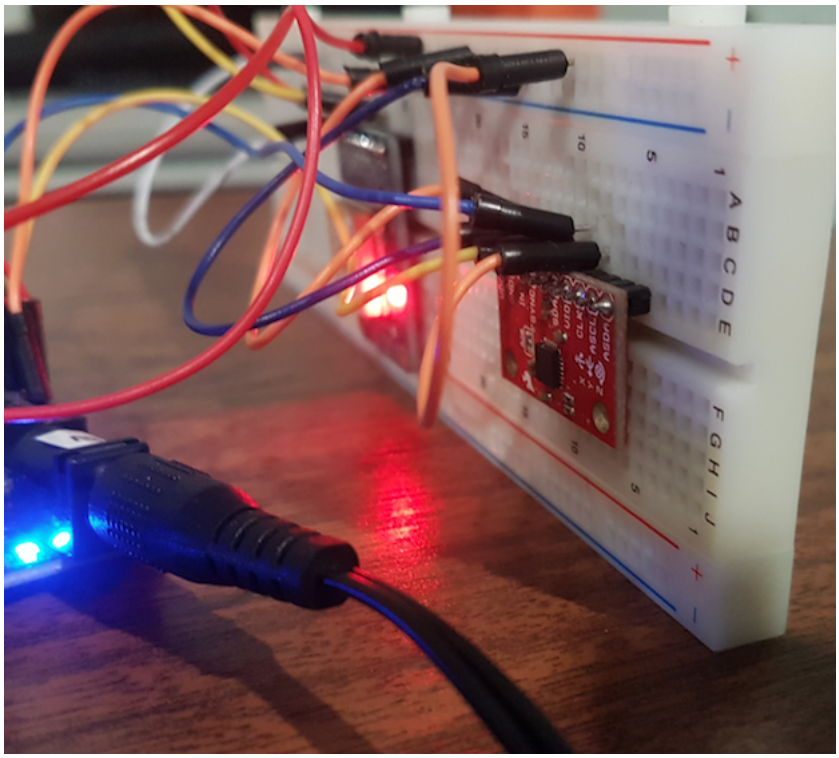


Figura 4.3: Posição da breadboard para a medição do ângulo roll

O valor neste caso não chegou a exatamente -90º pela falta de precisão na medição do sensor.

Relativamente ao valor do *pitch* utilizou-se o mesmo processo, mas neste caso, a posição da *breadboard* foi colocada na vertical, de modo a medir entre -90º e +90º. Os resultados foram semelhantes aos anteriores como se pode verificar nas figuras 4.4 e 4.5. O valor do ângulo *pitch* foi de 87.9485º também devido a erros de medição.

```
root@beaglebone:~# echo "getMeasurements*" | netcat 192.168.7.2 5555
0 0 0 -0.608345 87.9485 4916.45 N 12311.12 W 000.0
```

Figura 4.4: Teste valor do ângulo pitch

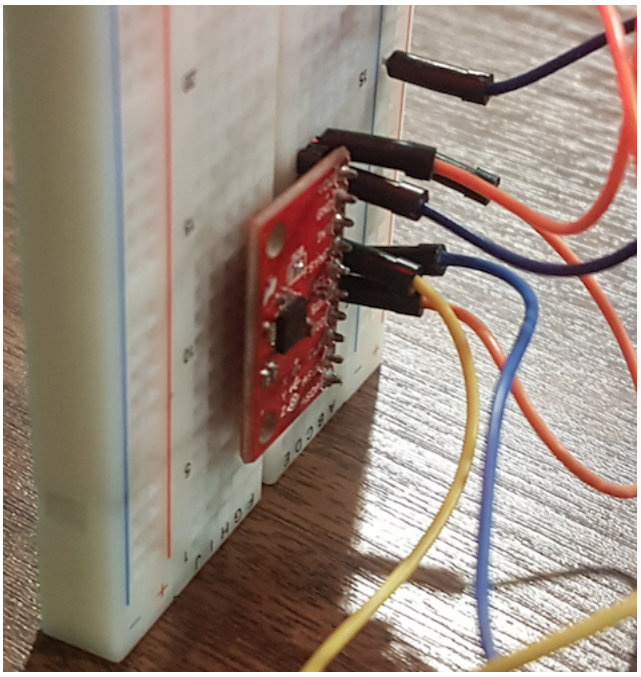


Figura 4.5: Posição da breadboard para a medição do ângulo pitch

4.4 Testes comunicação com as antenas

O teste de comunicação entre o FEUP VEC e uma Base-Station não foi efetuado com o próprio veículo devido à dificuldade em conseguir uma alimentação para os dispositivos (BeagleBone Black e Bullet M2) através das baterias do automóvel. Deste modo o teste consistiu numa comunicação entre dois pontos a uma distância de cerca de 150 metros. O cenário utilizado foi o corredor principal do edifício B da Faculdade de Engenharia da Universidade do Porto em que é possível obter uma linha de vista entre os dois pontos. O teste consistiu no envio dos dados sensoriais e na realização de uma *stream* de vídeo.

Conforme o esperado, os resultados foram obtidos com êxito. O vídeo foi enviado com resolução de 1920x1080 pixeis, a um *frame-rate* de 30fps (*frames per second*) com um *delay* de 3 a 4 segundos e os dados provenientes dos sensores foram também recebidos.

4.5 Problemas no envio do vídeo em tempo-real

Ao fim de alguns testes de emissão com a câmara, verificou-se que, por vezes, esta não se comportava como esperado. De acordo com as especificações do produto, e confirmado por outros utilizadores do mesmo produto, a câmara deve manter um *bitrate* constante de 3Mbits/s, no entanto verificou-se que este variava na gama de 3 e 6Mbits/s. Após alguns testes, quando o *bitrate* ultrapassava os 3.4Mbits/s, o canal vídeo estabelecido entre o BeagleBone Black e o computador saturava, conduzindo a uma reduzida qualidade de imagem, por vezes completamente imperceptível.

De forma a tentar resolver este problema, contactou-se a equipa de suporte técnica da marca (Logitech). A equipa de suporte não conseguiu identificar a origem da anomalia, aconselhando que se procedesse à troca do produto. Contactou-se também, por e-mail, o Professor Derek Molloy, da Dublin City University, e autor do livro "Exploring BeagleBone: Tools and Techniques for building with embedded Linux". Foi obtida uma resposta em tudo semelhante aquela dada pela equipa da Logitech. Confirmou que não era um comportamento normal da câmara e que nunca se tinha deparado com tal anomalia.

A solução consistirá então numa troca do equipamento.

4.6 Conclusões e requisitos cumpridos

Neste capítulo foram elaborados os diversos testes de implementação do sistema de telemetria. A metodologia utilizada consistiu numa simulação de um cenário de comunicação a uma distância elevada. Relativamente aos valores provenientes dos sensores foram utilizadas técnicas simples de validação de resultados. Para o sinal de GPS e respetiva velocidade instantânea foi utilizado um veículo convencional de modo a validar os resultados da latitude, longitude e velocidade, efetuando comparações com os resultados esperados. Para validar os valores dos ângulos de *pitch* e *roll* provenientes do acelerómetro/giroscópio, foram efetuados diversos testes com o auxílio de uma *breadboard*, efetuando diversas rotações de modo a realizar uma interpretação dos resultados. Os valores dos diversos sensores foram obtidos com êxito.

Relativamente à comunicação sem fios não foi possível efetuar a implementação diretamente no veículo, sendo então simulada entre dois pontos de distância considerável (cerca de 150 metros). O teste foi também obtido com êxito.

Os requisitos inicialmente propostos foram parcialmente cumpridos:

- Características mecânicas — o sistema desenvolvido ainda representa um protótipo apenas funcional. Ficará para trabalho futuro realizar uma integração completa no FEUP VEC. Por outro lado, a meta dos 2Kg de peso máximo foi atingida;

- Características elétricas — o consumo máximo de 20W foi cumprido, uma vez que o módulo BeagleBone Black/Cape consome, no máximo, 10W e o Ubiquity Bullet M2 consome, também no máximo, 7W. Por outro lado, fica para trabalho futuro incorporar uma solução capaz de extrair a energia proveniente das baterias do veículo para alimentar todo o sistema.

- Comunicação entre o veículo e *Base-Station* — o *delay* máximo de 5 segundos foi cumprido, ficando-se pelos 3/4 segundos na receção do vídeo proveniente da câmara;

- Resolução de imagem — o requisito mínimo de 640x480 pixéis foi cumprido, atingido até os 1080x1080 pixéis;

- Valores lidos — os valores das acelerações e localização em pista, posição do travão e acelerador e velocidade de caixa engrenada foram transmitidos.

- Distância de comunicação — apenas foram elaborados testes a 150 metros, mas o sistema está preparado para comunicar a distâncias de 2Km.

Capítulo 5

Conclusão e trabalho futuro

No presente capítulo são abordadas as principais conclusões retiradas na realização do projeto bem como algumas propostas para um desenvolvimento futuro e melhorias do sistema implementado. É de salientar que a realização da dissertação permitiu consolidar uma grande diversidade de conhecimentos anteriormente adquiridos bem como adquirir outros nunca antes abordados. Por outro lado, é de dar ênfase à obtenção de capacidades técnicas e científicas que resultaram do enfrentar dos diversos problemas que foram surgindo aquando a realização do projeto. Após uma análise teórica, respetiva implementação de um protótipo funcional para o projeto em causa e a consequente obtenção de resultados possibilitaram uma melhor perceção do verdadeiro significado da ciência de engenharia.

5.1 Conclusão da dissertação

Do ponto de vista do autor, a realização deste projeto possibilitou a aquisição de diversos conhecimentos, nomeadamente relativos a comunicações sem fios, comunicações série e programação em diversas linguagens. Entre estes, destacam-se principalmente protocolos de comunicação de redes com e sem fios, programação de *Single Board Computers* (em ambiente Linux), desenvolvimento de software em JAVA, Qt e C++, análise de grandezas espaciais (*yaw, pitch, roll*) e utilização de dispositivos GPS.

Os sistemas de telemetria, maioritariamente no contexto da competição automóvel, são responsáveis pelo aperfeiçoamento do comportamento quer do veículo, quer do condutor. É graças a este tipo de sistemas que se podem corrigir falhas e efetuar melhorias na performance dos veículos em pista, sendo um fator chave e de grande importância no mundo da competição.

Existem diversas tecnologias de comunicações sem fios que permitem a transferência de dados entre dois ou mais pontos, sendo que a frequência da onda portadora é a responsável pela distância e débito do envio dos dados. Relativamente às comunicações com fios, existe também uma grande diversidade de protocolos de comunicação que são responsáveis por satisfazer as necessidades das transferências de dados entre diversos dispositivos. Estas necessidades consistem na velocidade

de envio/receção de informação, distância entre dispositivos, rede de sensores em questão e a validação dos dados recebidos.

Graças ao enorme avanço tecnológico dos veículos automóveis, a necessidade de se efetuarem leituras de grandezas físicas ou sinais elétricos tem vindo a crescer. Deste modo, a indústria de sensores está em constante crescimento, acompanhado paralelamente a evolução dos veículos automóveis.

Na implementação propriamente dita do sistema de telemetria e sensorização para o FEUP VEC foram utilizados algoritmos relativamente simples de captação, transmissão e representação de dados em diversas linguagens, que permitiram a introdução a novas técnicas de programação.

É importante salientar que o maior desafio na implementação de um sistema deste género consiste na diminuição do tempo de comunicação, que em ambiente de competição pode marcar a diferença na obtenção de melhores resultados por parte do veículo.

Por outro lado, estes sistemas podem ser utilizados em contextos completamente diferentes, nomeadamente nos veículos autónomos ou mesmo na segurança rodoviária. O melhoramento deste tipo de sistemas pode vir a ser um passo grande na evolução dos automóveis em geral.

5.2 Trabalhos futuros

Para que este sistema seja perfeitamente funcional é necessária a sua implementação no veículo FEUP VEC, com o objetivo de ser utilizável em ambiente de competição. Para continuar o projeto, será então necessário implementar *hardware* dedicado à medição real dos valores dos pedais e da velocidade de caixa. Por outro lado, fica por desenvolver uma solução que permita incorporar o BeagleBone Black, juntamente com o *cape* utilizado, no *tablier* do veículo. Graças à utilização de um *cape* com um monitor tátil, será pertinente o desenvolvimento de uma aplicação que permita ao condutor interagir entre vários *layouts* de monitorização dos dados, através do toque. É de referir também que, uma vez que o veículo é elétrico, seria necessário efetuar a alimentação de todo o sistema de telemetria através das baterias do FEUP VEC.

Referências

[1] Daniel Correia e Daniel Rocha. Utilização de sensores na indústria automóvel. Relatório técnico, Instituto Superior de Engenharia do Porto, 2012.

[2] A.N. Safacas I. P. Tsoumas, E.D. Mitronikas. Design of a multifunctional data acquisition and telemetrysystem for electric vehicles - application in a solar car. April 2015.

[3] Wilfrid J. Mayo-Wells. The origins of space telemetry. *Technology and Culture*, 4(4):499–514, 1963. URL: http://www.jstor.org/stable/3101383.

[4] João Miguel Ribeiro Queirós. Sistema de sensorização e telemetria de um vec (veículo eléctrico de competição). Tese de mestrado, 2011.

[5] Justino Miguel Neto de Sousa. Sistema bidirecional de carga de baterias para o feup vec. Tese de mestrado, 2013.

[6] Derek Molloy. *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux*. Wiley, 2014. URL: http://www.exploringbeaglebone.com/.

[7] Jagannathan Parthasarathy. Positioning and navigation system using gps. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, 36(Part 6):208–212, 2006.

[8] Adafruit. Adafruit ultimate gps breakout - 66 channel w/10 hz updates - version 3, 2016. Disponível em https://www.adafruit.com/product/746.

[9] Paula Zamboni de Campos. Desenvolvimento de um aplicativo computacional de comunicação com gps e análise estatística das informações. 2004.

[10] Mark Pedley. Tilt sensing using a three-axis accelerometer. *Freescale Semiconductor Application Note*, páginas 2012–2013, 2013.

[11] Tr4nsduc7or. Tutorial de arduino y mpu-6050, Outubro 2014. URL: http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/.

[12] InvenSense. *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.0*, 09 2012. Revision 4.

[13] Joerg Dittmer. Are you ready for drive-by-wire?, 2001. URL: http://www.frost.com/prod/servlet/market-insight-print.pag?docid=CEHR-54YTXP [último acesso em 2016-04-12].

[14] Paul Towle. Gear position sensor, Novembro 6 2001. US Patent 6,311,552.

[15] Yi-yuan Fang e Xue-jun Chen. Design and simulation of uart serial communication module based on vhdl. Em *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, páginas 1–4. IEEE, 2011.

[16] Philips Semiconductors. The i2c-bus specification. *Philips Semiconductors*, 9397(750):00954, 2000.

[17] Andrew Luecke. Understanding wireless frequencies (900mhz vs 2.4ghz vs 5.8ghz), 2015. URL: https://jayvee.com.au/knowledge-base/networking/understanding-wireless-frequencies-900mhz-vs-2-4ghz-vs-5-8ghz.

[18] Cheng-Han Lee, Salman Abdul Baset, e Henning Schulzrinne. Tcp over udp.

[19] Henning Schulzrinne, Stephen Casner, Ron Frederick, e Van Jacobson. Rtp: A transport protocol for real-time applications. Relatório técnico, 2003.

[20] James Gosling. *The Java language specification*. Addison-Wesley Professional, 2000.

[21] I JetBrains. Intellij idea. *On-line at www. intellij. com*, 2003.

[22] Vlc playback features, 2016. URL: https://www.videolan.org/vlc/features.php.

[23] Ubiquiti Networks. *Ubiquiti Bullet M2 Datasheet*, 2011.

[24] Ubiquity Networks. airos 5 user guide, Maio 2016. URL: https://dl.ubnt.com/guides/airOS/airOS_UG.pdf.

[25] TeamDev. Jxmaps, Maio 2016. URL: https://www.teamdev.com/jxmaps.