FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Veículo Aéreo Não Tripulado para integração com Redes de Sensores Sem Fios

Daniel Filipe Valente da Silva

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Gil Gonçalves Eng.

Co-orientador: António Sérgio Ferreira Eng.

Julho de 2012

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2011/2012

A Dissertação intitulada

"Veiculo Aéreo Não Tripulado para Integração com Redes de Sensores Sem Fios"

foi aprovada em provas realizadas em 27-07-2012

o júri

Presidente Professora Doutora Maria do Rosário Marques Fernandes Teixeira de Pinho
Professora Associada do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Augusto Nunes Vicente Passos Morgado
Tenente Coronel da Força Aérea Portuguesa

Mestre Gil Manuel Magalhães de Andrade Gonçalves
Assistente Convidado do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Mestre Sérgio Ferreira
Investigador do DEEC - FEUP

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Daniel Filipe Valente da Silva

Faculdade de Engenharia da Universidade do Porto

Resumo

Com a evolução da tecnologia os *Unmaned Aerial Vehicles* (UAVs) começam cada vez mais a fazer parte de sistemas *lowcost*. As características modulares, típicas de sistemas UAV, tornam-nos uma mais valia para cenários extremamente dinâmicos pois possibilitam um alto nível de adaptação a mudanças nos objetivos de missão. Adicionalmente, *Unmanned Aerial Systems* (UAS) permitem uma redução de custos a nível logístico, bem como um aumento de segurança graças ao distanciamento que proporcionam do elemento humano, em relação à zona de operação. Todas estas possibilidades levam-nos a investigar e desenvolver sistemas mais complexos, potencializando assim a eficácia e a eficiência de equipas mistas (Robôs e Humanos). Para tal ser possível é necessário um elevado grau de autonomia por parte destes sistemas.

Para alcançar estes objetivos, tem existido um grande investimento em sistemas aéreos não tripulados, promovendo tecnologias e soluções que permitam satisfazer os requisitos exigentes das missões. O desenvolvimento de algoritmos de controlo avançado, tais como os de coordenação e cooperação entre equipas de UAVs e outras equipas no cenário, são um exemplo dos esforços efetuados nesta área.

A presente dissertação pretende dotar os UAVs com capacidade de interagir com outros elementos de missão, sejam elementos estáticos ou dinâmicos, podendo dessa forma existir uma maior cooperação entres os elementos presentes no teatro de operações. Possibilitando desta forma inclusive a monitorização de uma dada área de interesse. Para isso é necessário recolha de dados, a sua transmissão e a apresentação após um dado tratamento.

Neste documento é descrito a implementação uma RSSF (rede de sensores sem fios), de forma a que esta possibilite a recolha dos dados pretendidos. Assim como a adaptação realizada um veículo aéreo não tripulado para que este possa interagir com a rede criada. Para possibilitar a utilização do sistema foi também criada uma interface para que o utilizador possa visualizar a rede disseminada e os dados por ela recolhida, permitindo desta forma validar estratégias de colaboração entre sistemas num teatro de operações.

De acordo com os testes realizados, podemos afirmar que a RSSF tem as características teóricas para o correcto funcionamento do sistema pretendido. Mas em ambiente exterior, e sem a adição de hardware extra aos nós de sensores, estes não tem capacidade para realizar as comunicações pretendidas de uma forma eficaz, aquando presentes de um cenário de operação com UAVs.

Palavras-Chave:UAVs;Motes;RSSF;

Abstract

With the evolution of technology, Unmaned Aerial Vehicles (UAVs) begin to be a increasing part of lowcost systems. The typical modular characteristics of UAV systems, make them an asset for highly dynamic scenarios as they allow a high level of adaptation to changes in mission objectives. Additionally, Unmanned Aerial Systems (UAS) allow a costs reduction in logistics, as an increased safety through the distance that provide to the human element in relation to the test zone.

All these possibilities leads us to explore and develop more complex systems, leveraging thus the effectiveness and efficiency of mixed teams (Humans and Robots). To make this possible is needed a high degree of autonomy of these systems.

To achieve these goals, there has been a great investment in unmanned aerial systems, by promoting technologies and solutions that meet the demanding requirements of missions. The development of advanced control algorithms, such as coordination and cooperation between teams of UAVs and other teams on the scene, are an example of the efforts made in this area.

This dissertation aims to provide the UAVs with the ability to interact with other mission elements, whether they be static or dynamics elements and may thus be a greater cooperation between the elements present in the operation theater. Allowing a possibility to monitoring a given area of interest. This requires data collection as data transmission and the presentation of the data collected after a given treatment.

In this document a WSN (wireless sensor network) implementation is described, that enables the collection of required data. As an adaptation of an unmanned aerial vehicle so that it can interact with the network created. To enable the use of the system was also created an interface that permits the user view the network spreaded as the data collected, allowing also strategies validation for collaboration between systems in theater.

According to the tests performed, we can say that the WSN has the theoretical characteristics for the proper functioning of the system desired. But in the exterior environment, and without adding extra hardware to the sensors nodes, that is unable to carry an effective communications, when present to a operation scenario with UAVs.

Keywords: UAVs, Motes, WSN;

Agradecimentos

Gostaria de começar por agradecer Professor Gil Gonçalves, por ter acreditado em mim para o desenvolvimento deste projeto, pela sua disponibilidade e apoio no decorrer toda esta dissertação.

De forma igual, e com o seu merecido reconhecimento ao Engenheiro António Sérgio Ferreira por ter sido uma base de conhecimento e apoio, pois mesmo quando o trabalho complicou e afetou-me de uma forma mais negativa a moral, fez questão de estar sempre presente e com uma constante preocupação para que este projeto chegasse a bom porto, tornando-se num grande amigo.

Seguidamente, quero agradecer a todos os elementos do LSTS e do AsasF, pois só a boa vontade e paciência deste elementos é que permitiu a conclusão deste projeto, uma especial referência ao incansável e paciente Ricardo Silva, que sempre esteve disponível para transmitir o seu conhecimento, sem o qual não teria conseguido realizado este projeto, assim como questões relacionados com hobbys dos quais partilhamos.

Não esquecer de agradecer a Faculdade de Engenharia da Universidade do Porto por ter permitido, que durante os anos da vida académica que aqui passei, ter-me cruzado com os professores que tive, assim como com os colegas que tenho e tive, possibilitando estes transmitirem, nem que uma parte, do seu conhecimento, fazendo de mim uma pessoa mais instruída e mais Engenheiro. Foi também nesta muy nobre instituição que criei grandes amizades, das quais nunca me esquecerei.

Amizades essas, ás quais me dirijo agora para lhes agradecer, seja pelo apoio que deram nas noites de trabalho, nas horas de estudo para os exames, quer pela diversão que partilhamos nas festas, nas conversas fossem estas sobre maiores banalidades ou sobre atualidade e com interferência para o país. De destacar a Elisabete Fernandes, pelo seu apoio constante, pois esteve sempre presente durante o desenvolvimento deste projeto, a quem sempre recorri para fazer-me ver a outra perspectiva, quando eu sozinho não o conseguia. Também merece o seu devido destaque, pela amizade sempre demonstrada, pelos bons momentos musicais que proporcionou, assim como a grande ajuda que deu nas questões do português, o João Amorim.

Finalmente quero agradecer a toda a minha família, que desde sempre me apoiaram e acreditaram que conseguiria atingir os meus objetivos, em especial aos meus pais, que com todos os seus sacrifícios possibilitaram-me esta conquista na minha vida, assim como quero agradeço-lhes a educação que me deram e os valores que me transmitiram, fazendo que eu hoje seja o homem que sou.

Daniel Filipe Valente da Silva

"Improvement makes strait roads: but the crooked roads without Improvement are roads of Genius."

William Blake

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ABC	Assumption Based Coordianated
AM	Modulação em Amplpitude
AOA	Angle of Arrival
API	Application Programming Interface
APP	Application Layer
APS	AD-hoc positioning system
ARM	Advanced RISC Machine
ASK	Amplitude Shift Keying
BPSK	Binary Phase-Shift Keying
CCK	Complementary Code Keying
CISC	Complex Instruction Set Computer
CPU	Unidade Central de Processamento
CSMA-CA	Carrier Sense Multiple Access - Colision Avoidance
DDD	Dirty, Dull & Dangerous
DIY	Do it yourself
DOA	Direction of Arrival
DPE	Directed Position Estamation
DSSS	Direct Sequence Spread Spectrum
FEUP	Faculdade de Engenharia da Universidade do Porto
FFD	Full Function Device
FHSS	Frequency Hopping Spread Spectrum
FM	Modulação em Frequência
FSK	Frequencu Shift Keying
FTDI	Future Technology Devices Internacional, Ltd.
GPS	Global Position System
GTS	Guarantee Time Slots
HT	High Throughput
IEEE	Institute if Electrical and Electronics Engineers
IMC	Inter-Module Communication Protocol
IMU	Inertial Measurement Unit
ISM	Industrial, Scientific and Medial
I & T	Investigação e Tecnologia
LMB	Localização com uma âncora móvel
LSTS	Laboratório de Sistemas e Tecnologia Subaquática
MAC	Medium Acess Control
MAV	Micro Aerial Vehicle
MDS	Multidimensional Scaling
MRA	Mission Review and Analysis
MSB	Most Significant Bit

NesC	Networked Embedded Systems C
NWK	Network Layer
O-QPSK	Offset Quadrature Phase-Shift Keying
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
OSI	Open System Interconnection
PHY	Physical Layer
PITVANT	Projecto de Investigação e Tecnologia em Veículos não Tripulados
PM	Modulação em Fase
PSK	Phase Shift Keying
PSSS	Parallel Sequence Spread Spectrum
RC	Radio Controlo
RF	Radio Frequência
RFD	Reduced Function Device
RISC	Reduced Instruction Set Computer
RPE	Recursive Position Estimation
RSSF	Redes de Sensores Sem Fios
RSSI	Received Signal Strength Indicator
SPI	Serial Pheripheral Interface
TDOA	Time Dofference of Arrival
TERRAIN	Triangulation via Extended Range and Redundant Association of Intermediate Nodes
TOA	Time of Arrival
TOF	Time of Flight
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
UUV	Unmanned Underwater Vehicle
WIFI	Wireless Fidelity

Capítulo 1

Introdução

Os veículos aéreos não tripulados têm recentemente visto um crescimento surpreendente na sua utilização em inúmeras aplicações militares e civis [1]. É uma área em grande desenvolvimento e expansão podendo ser afirmado que é o sector com o crescimento mais dinâmico na industria aeroespacial [12]. Esta dissertação surge como continuação da dissertação intitulada "Low Cost Unmanned Aerial Vehicle for Testing and Validation of Advanced Control Algorithms" [12] e pretende expandir o trabalho já realizado na área de UAVs de forma a obtermos uma maximização dos recursos disponíveis e uma otimização do cumprimento do objetivo de uma missão. Com a maturação desta área, novas aplicações surgem, tais como inspeção de *pipelines* e linhas de energia, prevenção de fogos [13], vigilância de uma região, missões de salvamento [14] e outras mais. Como exemplos de atividades com propósitos mais lúdicos [15]. Tudo isto despoleta o interesse e o maior desenvolvimento para este tipo de sistemas. Esta investigação e desenvolvimento tem como objetivo final melhorar a performance dos UAVs e dos sistemas de supervisão.

Tendo em mente objetivos de uma missão percebe-se então a necessidade de interação deste género de sistemas com outros, possibilitando assim uma maior cooperação e colaboração entre os sistemas interligados, para isso este trabalho irá tentar interligar um sistema de veículos aéreos não tripulados com uma RSSF (rede de sensores sem fios).

As RSSF tornaram-se populares devido à sua aplicabilidade, em diversas áreas tais como meio ambiente, saúde, a indústria, agricultura, meteorologia espacial e militar [16]. Sendo objetivo principal de uma rede deste género o de monitorizar uma área de interesse, para que a rede consiga realizar este objetivo de forma eficaz, é necessário acomodar alguns pré-requisitos [19]. Um desses pré-requisitos é o conhecimento da posição física dos sensores, necessária para a viabilidade de muitas das aplicações de RSSF. A questão de obter o posicionamento físico de um objecto é uma questão ampla e extensa a várias áreas tais como redes ad-hoc, redes de telemóveis, aeronáutica e robótica.

1.1 Contexto/Enquadramento

A dissertação enquadra-se num projeto de maior complexidade lançado pela Academia da Força Aérea, em parceria com a FEUP (Faculdade de Engenharia da Universidade do Porto), tendo esse projeto a designação de PITVANT (Projeto de Investigação e Tecnologia em Veículos Não Tripulados). Será igualmente desenvolvido em parceria com os AsasF, equipa responsável pela esquadrilha dos UAVs do LSTS (Laboratório de sistemas e Tecnologia Subaquática). De seguida é apresentado o Projeto PITVANT, de forma a que seja possível a compreensão da dimensão do projeto.

O PITVANT é a evolução do projeto iniciado em 1996 na Academia da Força Aérea, explorando assim as oportunidades disponíveis de investigação e desenvolvimento de tecnologias para UAVs, com o objetivo de possibilitar à Força Aérea obter mais competências de exploração, vigilância, sejam estas com objetivos militares ou civis [12]. Este projeto tem como objetivos:

- Desenvolver protótipos de sistemas de UAVs, com maior foco de pequena e media dimensão, com a tecnologia mais moderna embebida, a serem utilizados em várias atividades e missões, sejam de carácter militar ou civil;
- Dotar a Força Aérea de *Know-how* relativo à utilização e operação de UAVs, assim como os requisitos técnicos e operacionais;
- Criação, em conjunto com a FEUP, de um projeto de I & T, em especifico com LSTS, para desenvolvimento e criação de hardware e software para UAVs.

Estes objetivos começaram a ser desenvolvidos em Novembro de 2008 e estender-se-á até Dezembro de 2015.

1.2 Motivação e Objetivos

A motivação deste tema surge do desafio da aplicação da crescente tecnologia UAV em tarefas DDD (Dirty, Dull & Dangerous), para que seja possível libertar recursos humanos sem comprometer os objetivos das atividades ou missões. Apercebemo-nos então das possíveis vantagens do uso da colaboração entre UAVs, permitindo missões mais complexas e com uma eficiência superior. Permitindo assim retirar o elemento humano de tarefas repetitivas e recolocando-o em tarefas de supervisão e gestão.

Como motivação pessoal, tive a possibilidade da incorporação numa equipa de trabalho, altamente especializada e com elevadas competências em várias áreas, possibilitando assim uma grande aprendizagem pessoal tanto em *Hard Skills* como em *Soft Skills*, evoluindo assim tanto a nível social como a nível profissional.

Os principais objetivos da dissertação serão:

- Validar estratégias de um UAV com outros sistemas no teatro de operações;

- Conferir aos sistemas UAVs existentes no LSTS a capacidade de interagir com outros elementos de missão, quer sejam estáticos (MOTES, sensores) ou dinâmicos (UAVs, AUVs);
- Integrar UAV em redes *Mesh* de forma a garantir uma maior cobertura e redundância de serviço.

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos.

- No capítulo 2, é realizada a revisão bibliográfica, onde são descritos outros trabalhos na área, apresentado o estado da arte de componentes, sistemas e abordagens para problemas com questões parecidas. Contêm a pesquisa realizada sobre sistemas de localização, onde é descrito a sua constituição e os métodos existentes, o respetivo estado da arte e abordagens utilizadas para sistemas com propósitos parecidos.
- Já no Capítulo 3, é apresentado o caso de estudo em que o trabalho apresentado nesta dissertação foi baseado, após isso é apresentado as abordagens que são plausíveis e que possíveis seguir para a realização deste trabalho, referentes aos vários constituintes sistema, após esta análise, é apresentado as escolhas e os métodos utilizados e as respetivas justificações, este capítulo é encerrado com a descrição do sistema obtido.
- No Capítulo 4, são apresentadas as implementações realizadas para se obter o sistema pretendido, assim como a explicação de como foi implementado cada um dos componentes deste sistema.
- O conteúdo do Capítulo 5, é referente aos testes que foram realizados e a apresentação dos resultados, a partir dos quais se fez a validação do sistema.
- Já no Capítulo 6, o capítulo final, são apresentadas as conclusões, tanto ao trabalho realizado, como à satisfação de resultados. Neste capítulo também são apresentadas várias propostas para trabalho futuro e a sua importância para a comunidade.

Capítulo 2

Revisão Bibliográfica

Esta dissertação tem como objetivo permitir que sistemas de UAVs possam interagir com outros sistemas ou elementos presentes no teatro de operações. Verificamos então que estamos perante equipas mistas, com uma componente Humana e outra robótica, podendo esta ultima ser homogénea ou heterogénea. De seguida definiremos cada um destes tipos de componentes robóticas, para melhor compreensão das características de cada uma, assim como as variáveis e a complexidade inerente a cada tipo de componente robótica.

2.1 Componente robótica homogénea

Este tipo de equipa é o mais investigado, pois permite uma representação indiferenciada dos elementos, assim como as suas restrições, *payload* e as capacidades, visto serem iguais, como exemplificado na Figura 2.1.

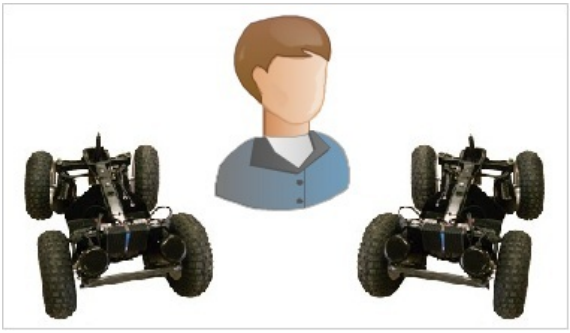

Figura 2.1: Exemplo de uma equipa mista com componente robótica homogénea [1]

Esta igualdade permite uma generalização, quer de conclusões relativas aos dados recolhidos, quer às interfaces de controlo. Mas existe a possibilidade destas conclusões serem inadequadas para aplicação a uma componente heterogénea, uma vez que não são contemplados diferentes tipos de funcionalidades, capacidades, autonomia e método operacional. Contudo existem várias lições aprendidas nos sistemas homogéneos de importante utilidade. [1]

2.2 Componente robótica heterogénea

Esta arquitetura de equipa apresenta uma maior complexidade, pois este tipo apresenta mais pormenores divergentes. No entanto esta complexidade permite ao mesmo tempo formar equipas mais abrangentes e com maiores competências. Na figura 2.2 [1] podemos ver uma exemplificação duma equipa deste género.

Figura 2.2: Exemplo de uma equipa mista com componente robótica heterogénea [1]

De seguida são apresentados algumas adversidades que se encontram aquando do uso deste tipo de equipas:

- Complicação na interface de controlo, pois cada tipo de UAV tem a sua.
- Complexidade na forma de representar e controlar a equipa, tendo em consideração diferentes autonomias, funções e *payload* [15].

2.3 Utilização de Veículos Aéreos Não Tripulados

Cooperação foi sempre um aspecto importante na aviação. O sucesso de uma missão com várias aeronaves, depende do sucesso da coordenação e cooperação entre as várias plataformas independentemente do seu tipo. Para conseguir este comportamento em equipas de UAVs têm sido exploradas técnicas de otimização e de coordenação deste tipo de equipas [22, 23]. Como exemplos desta aplicações temos:

- Deteção e perseguição de um alvo terrestre em movimento, ou estático [24, 25];
- Voo em formação, ataque cooperativo e sistemas de iniciativa mista, em ambientes de risco [26, 27];
- Algoritmos de controlo com capacidade de aprendizagem [28, 29];
- Algoritmos de controlo descentralizado para sistemas não lineares [30];

- Arquiteturas de controlo em tempo-real [31].

Assim sendo conclui-se que são necessárias soluções em tempo real para controlar cada UAV da forma desejada. De igual forma é necessário prestar-se atenção às comunicações e a cooperação entre UAVs, para construir tarefas e manobras com consenso na equipa.

Vários algoritmos de controlo cooperativo têm sido desenvolvidos recentemente, pela comunidade cientifica, com o objetivo de criar uma sinergia positiva nas operações com UAVs, aumentando a operacionalidade entre agentes e adaptação ao cenário [32]. No entanto é necessário decompor estes algoritmos de alto-nível, em algoritmos de controlo de baixo-nível a implementar nos sistemas de voo, de forma a suportar a investigação desses conceitos.

Um exemplo deste tipo de algoritmos é o *mapping and tracking* [33], onde cada agente atualiza a sua rota de acordo com a informação transmitida pelos outros agentes. Outro cenário refere-se à deteção de ameaças em ambiente hostil, com vários alvos conhecidos [34].

Outra investigação interessante, onde é demonstrada a importância das comunicações e as suas potencialidades, é a investigação descrita no artigo referido em [35] onde é criado um sistema de aeronaves não tripuladas tolerante à falha do GPS (Global Position System, onde é demonstrado que a partir das comunicações dos vários veículos aéreos se consegue localizar o agente com falha, evitando assim que este se despenhe, e até conseguindo-o recuperar.

Estes algoritmos interagem com os *autopilots*, enviando os comandos como rácios de curvatura, altura, velocidade e definindo *waypoints* para navegação, a Figura 2.3 é um exemplo ilustrativo.

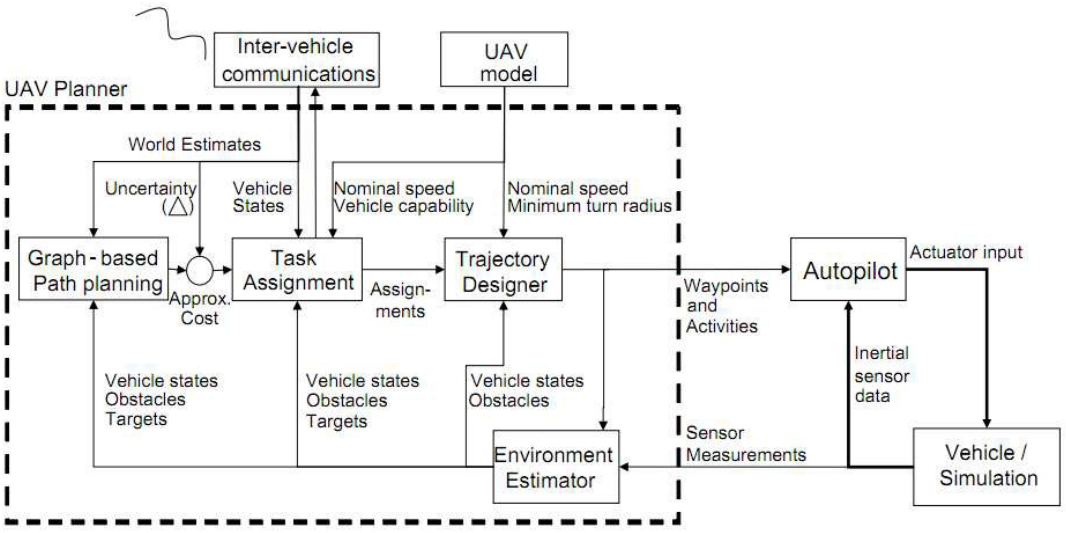

Figura 2.3: Architecture of multi UAV control planner [2]

2.4 Veículos Aéreos Não Tripulados

De forma a testar projetos em desenvolvimento, usualmente opta-se pelo uso de aeronaves de pequeno porte ou *Micro Unmanned Aerial Vehicles* (MAVs) e consequentemente de baixo custo. Os MAVs são normalmente baseados em modelos RC (*Radio Control*) de asa fixa, com as limitações inerentes a estes modelos, autonomia aproximadamente de 20 minutos, com uma velocidade a variar entre 2 a 10 metros por segundo e como voam a baixas altitudes sofrem uma grande interferência do ambiente e tempo meteorológico [12]. Apesar destas limitações, com uma boa adaptação conseguem servir o proposito pretendido, para teste e validação de algoritmos de controlo.

Outro ponto a ter atenção é que o *payload* é pequeno devido ás suas características.

Dois desses exemplos são os Zagi, demonstrado na Figura 2.4, e os Cularis.

Figura 2.4: Imagem de um Zagi

De porte pequeno existem várias soluções, tal como os B-Hunter [36], os Pilatus e os Alpha desenvolvidos exclusivamente para o PITVANT, apresentado na Figura 2.5.

Figura 2.5: Fotagrafia do Alpha 06

Existem também UAVs considerados de grande porte, sendo o exemplo mais conhecido o Predator [37] utilizado pela força aérea dos Estados Unidos da América. Abaixo é apresentado uma imagem exemplificativa deste tipo de UAV (Figura 2.6).

2.5 Unidade Central de Processamento de bordo (CPU)

Um dos componentes imprescindíveis é a unidade de processamento central (CPU), componente responsável pelo processamento da aeronave. Para este componente existem várias soluções desde soluções comerciais fechadas (Piccolo), passando por soluções comerciais *open-source*

Figura 2.6: Fotografia exemplificativa de um Predator

(ArduPilot-Mega) e soluções produzidas *in-house* (Igep e PC-104). Todas elas são placas que permitem performance e expansão de *Laptops*, sem o volume, ruído e despesa de um computador usual.

As grandes vantagens destes componentes residem no facto de serem uma alternativa sem ventoinha a um computador industrial, têm um baixo consumo de energia, e existem placas de expansão que permitem fornecer ainda mais capacidades a este sistema, além do uso de *open source software*, para além do seu tamanho reduzido.

2.5.1 Piccolo

O Piccolo [3] é uma família de sistemas de controlo e gestão de voo desenvolvido pela Cloud Cap Technologies, que é a maior referência dentro das soluções comerciais disponíveis para UAVs. Estes sistemas vêm munidos de *autopilot*, GPS, IMU, sensores de pressão, comunicações radio e interfaces para o *payload* que suportam vários periféricos possíveis, seja através de interfaces RS232 ou CAN. Esta solução também dispõe de *software* para *Ground Station* e simulação. A grande desvantagem é ser uma solução "fechada", sendo também um pouco dispendiosa quando comparada com outras soluções.

Figura 2.7: Imagem do Piccolo SL [3]

2.5.2 ArduPilot

O projeto ArduPilot [4] tenciona criar uma família de sistemas de auto-piloto de baixo custo e *open-source* baseado nas plataformas de Arduino.Este projeto é baseado no conceito DIY (*Do It Yourself*) que tenciona oferecer um sistema expansível, configurável e modular. Sendo uma placa de Arduino o *hardware* de base, uma solução completa de *hardware* e *software* para controlo e gestão de voo pode ser conseguida através da adição de expansões, como por exemplo sensores de voo, navegação e comunicações *wireless*. O *software* do auto-piloto é modular, adaptável ao *hardware* implementado no sistema e oferece vários modos de voo, desde um simples modo de estabilização até a navegação por GPS.

Figura 2.8: Imagem de um ArduPilot [4]

2.5.3 PC-104

Esta unidade é uma unidade de dimensões reduzidas com capacidade de processamento relativamente elevado para o seu tamanho, peso e consumo de energia, daí ter sido uma das escolhas para implementação de competências aos sistemas de UAVs do LSTS. PC-104 é referente a um standard de arquitectura que permite emparelhamento até três placas computacionais, o que permite expandir as capacidades da placa principal (por exemplo processamento de imagem dedicado). Este standard utiliza a arquitectura de CPUs CISC(*Complex Instruction Set Computer*). A PC-104 em si não é um CPU, pode é ter CPUs que chegam ao 1,3GHz.

2.5.4 Igep

A grande diferença com este CPU é a sua arquitectura, pois possuí arquitectura ARM (Advanced RISC (Reduced instruction set computer) Machine).

Figura 2.9: Imagem de uma PC/104 [5]

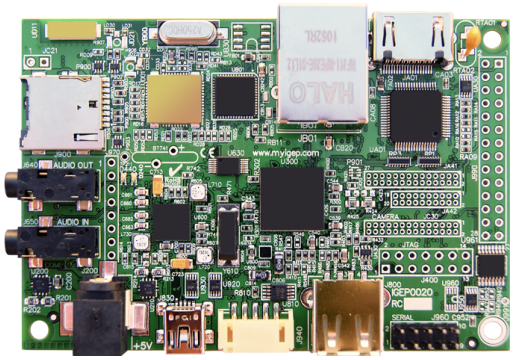

Figura 2.10: Imagem de uma Igep [6]

2.6 Sensores Sem Fios

Um componente importante de sistemas UAV são os sensores nele embutidos e aqueles que se encontram disseminados pelo teatro de operações. Estes podem ser muito diferentes, com sensibilidades diferentes, e com características muito diferentes entre si.

Na preparação deste trabalho foram identificados sensores que normalmente estão acoplados aos UAVs, tais como GPS, câmaras para dotar os UAVs da capacidade de visão [34, 38], radares, barómetros, etc [39, 40].

2.6.1 Motes

Juntamente com os sensores tipicamente acoplados a UAVs, também foram investigados sensores terrestres de pequenas dimensões e com capacidade de comunicação (Motes). Estes dispositivos autónomos constituem as RSSF, para isso as Motes são espacialmente distribuídas que,

com recurso a cooperação entre si, monitorizam condições físicas ou ambientais, como exemplos temos a temperatura, humidade, luminosidade, som, pressão, vibrações, etc.

Um exemplo de plataformas de Motes para redes de sensores de muito baixo consumo, e grande transferência de dados, é Tmote Sky, ver Figura 2.11, que tem integrado sensores, rádio compatível com as normas IEEE 802.15.4, antena, micro controlador e a capacidade de ser programada. Este género de dispositivos pode comunicar com um computador (hospedeiro) utilizando o controlador USB (Universal Serial Bus) da FTDI (Future Technology Devices Internacional, Ltd.) que possuem. O rádio é controlado pelo micro controlador através da porta SPI (Serial Pheripheral Interface), podendo o rádio ser desligado, quando se pretende operar num modo de baixo consumo energético. A antena interna é do tipo omnidirecional, que permite obter alcance de 50 metros, em ambientes fechados, até 125 metros, em ambientes abertos.

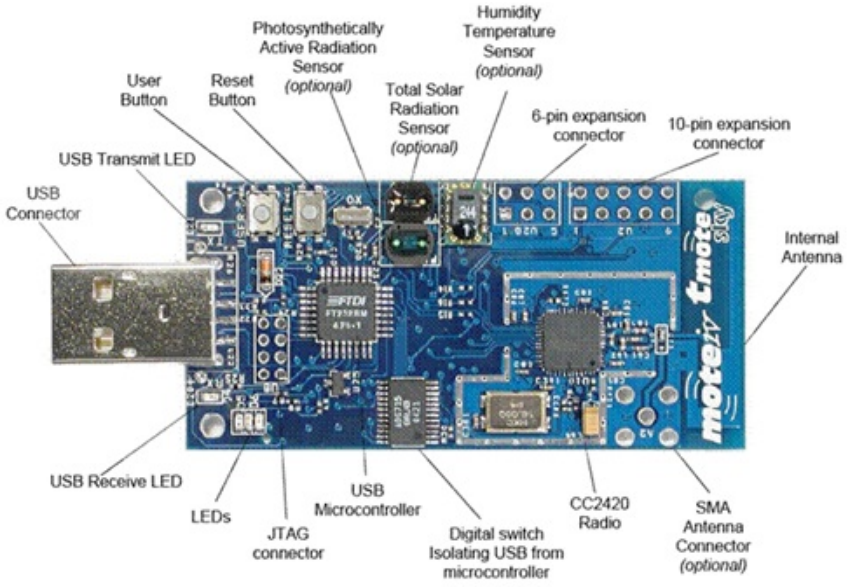

Figura 2.11: Imagem de uma Mote CrossBow TELOSB/TMote Sky

Foi verificado que o ênfase destes sensores é nos módulos de comunicação, pois são os módulos que consomem mais bateria, influenciando diretamente a autonomia destes sensores, e podendo colocar o objetivo da missão em risco [41].

2.6.2 Fusão de Dados

A comunidade cientifica, mostra um grande interesse na fusão dos dados recolhidos para melhor compreensão do teatro de operações e dos agentes intervenientes. Alguns sistemas realizam a fusão dos dados e a sua respetiva análise após a missão, isto acontece devido a restrições de *payload* e de comunicações em tempo real [42]. Alguns sistemas apresentam uma interface gráfica assim como guardam os *logs* binários. Esse sistema está implementado no LSTS, no Neptus com o módulo MRA (Mission Review and Analysis) [42].

Alguns investigadores deram maior importância a sistemas distribuídos [43, 44], até chegam a ser investigadas vantagens de criar *clusters* de sensores [45].

Tendo em atenção o tipo de sensores a usar e a área a cobrir, é definido um algoritmo para distribuição dos sensores para melhor cobertura do teatro de operações [46].

2.7 Comunicação

A comunicação entre os vários agentes presentes no teatro de operações, principalmente se se pretende tomar decisões em tempo real, é um ponto importante. Sendo necessário garantir segurança, fiabilidade, confidencialidade e uma baixa latência nas comunicações. Para isso usam-se vários protocolos de comunicação, como por exemplo WIFI (Wireless Fidelity), RF (Radio Frequency) e GPS. Para além disso, para este projeto será usado *ZgiBee* para comunicação com sensores terrestres.

Existem várias topologias de redes, cada uma com as suas características, vantagens e desvantagens. Para este projeto a topologia de rede que se pretende inicialmente utilizar é a topologia *Mesh*. A principal característica desta topologia é que cada nó da rede, não só tem de obter e difundir os seus dados, como de retransmitir os dados dos outros nós, ou seja, tem de colaborar de forma a propagar os dados pela rede.

É tipico neste tipo de sistema de UAV serem utilizados vários protocolos de comunicação, de forma a garantir que todas as funcionalidades são executadas e que todas as transmissões ocorram sem interferência uma nas outras, tentando-se assim garantir qualidade e fiabilidade do serviço.

2.7.1 ZigBee

A norma ZigBee [47] é uma norma para dispositivos *wireless* de baixo consumo energético, que operam nas bandas de radio para a industria, ciência e medicina (ISM), 868 Mhz na Europa 915 MHz no Estado Unidos da América e 2,4 GHz na maioria dos países do mundo [7].

O facto de usar a topologia *Mesh*, permite que a rede se ajuste autonomamente aquando a sua inicialização, na entrada ou perda de dispositivos.

A norma define dois tipos de dispositivos:

- FFD (Full Function Device);
- RFD (Reduced Function Device).

Um dispositivo de função completa (FFD) normalmente opera como coordenador ou *router*, apesar de poder ser utilizado também como dispositivo final. Tipicamente estes dispositivos estão ligados a uma alimentação principal. Já um dispositivo de função reduzida (RFD) é desenhado para baixo consumo energético e só pode ser utilizado como dispositivo final. Normalmente este género de dispositivos são alimentados por baterias.

Outra caraterística importante é o tamanho reduzido dos pacotes que são transmitidos. É relevante referenciar que o baixo consumo energético é conseguido pelo facto dos dispositivos

poderem operar em dois estados, sendo eles o estado ativo (*fully awaken*) e inativo/adormecido (*standby*), tendo como vantagem a possibilidade de poderem manter o estado inativo durante longos períodos de tempo, que podem ir de horas até mesmo dias. Uma exemplificação desta eficiência energética, é, uma vez um dispositivo associado a uma rede ZigBee, este pode passar do estado ativo, comunicar com os outros dispositivos ZigBee e voltar a ficar inativo. Os tempos típicos deste exemplo são:

- Tempo de acesso a rede = 30ms;
- Tempo de transição dos dispositivos do estado inativo para o estado ativo = 15ms;
- Tempo de acesso ao canal = 15ms.

A arquitectura da *stack* do protocolo ZigBee, ver Figura 2.12, é inspirada no modelo de sete camadas OSI (Open Systems Interconnection) [48], e está dividido em quatro camadas:

- PHY (A camada física);
- MAC (A camada de controlo de acesso médio);
- NWK (A camada de rede);
- APP (A camada de aplicação).

Podendo então concluir que as três camadas de mais baixo nível são implementadas como o modelo OSI, e a camada APP combina as quatro camadas de nível mais superior, isto quando comparado com o modelo OSI. As duas camadas mais baixas, PHY e MAC, são definidas pelo standard IEEE 802.15.4 [49].

Como o protocolo tem as duas primeiras camadas definidas de acordo com esta norma, lida com as interferências entre dispositivos utilizando as mesmas técnicas definidas na norma, são elas:

- CSMA-CA (Carrier Sense Multiple Access - Colision Avoidance);
- GTS (Guarantee Time Slots).

O CSMA-CA define que cada dispositivo escuta o meio de acesso, antes de qualquer transmissão. Ou seja se existir um nível de energia no meio, superior a um determinado valor, o transmissor espera um período aleatório e tenta novamente a sua transmissão.

O GTS define a utilização de um coordenador centralizado, que atribuí uma porção de tempo para cada dispositivo, habilitando assim cada um com o conhecimento de quando tem que transmitir. Existem 16 parcelas de tempo possíveis. Para a transmissão ser realizada, num primeiro passo, cada dispositivo deve enviar ao coordenador uma mensagem a solicitar a sua porção de tempo (GTS), depois o coordenador envia uma mensagem de sinalização que contém a informação da parcela de tempo alocado e número de *slots* tempo atribuídos.

Na norma IEEE 802.15.4 [49] é também especificado os seguintes PHYs:

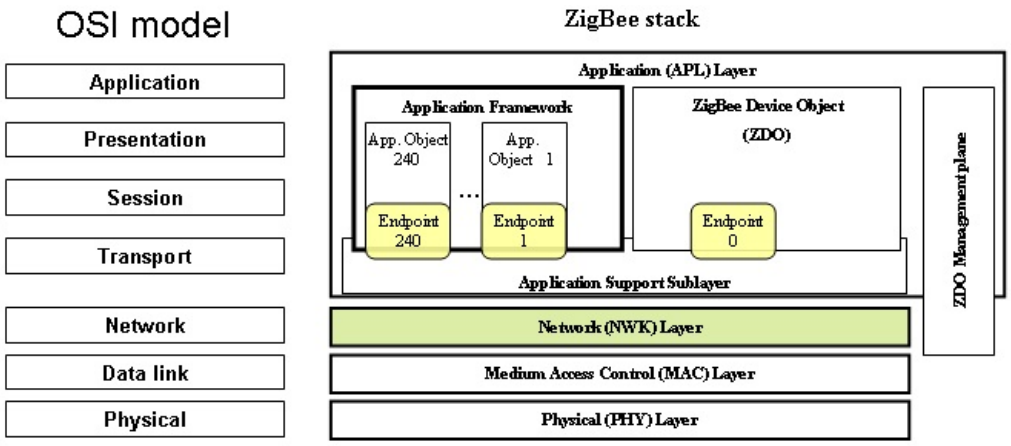

Figura 2.12: Modelo OSI e ZigBee Stack [7]

- 868/915 MHz *Direct Sequence Spread Spectrum* (DSSS) com modulação *Binary Phase-Shift Keying* (BPSK);
- 868/915 MHz DSS com modulação *Offset Quadrature Phase-Shift Keying* (O-QPSK);
- 868/915 MHz *Parallel Sequence Spread Spectrum* (PSSS) com modulação BPSK e *Amplitude Shift Keying* (ASK);
- 2450 Mhz DSSS com modulação O-QSPK.

Dos quatro PHYs apresentados, dois são opcionais e evoluiriam do PHY 868/915 MHz BPSK, que é especificado na edição original da norma referida [11]. Os dispositivos devem operar numa banda de frequência, usando a modulação e o formato *spread* resumido na Tabela 2.1.

Outra diferença do standard original para a sua atualização é numero de canais atribuídos, ultrapassa os 32 canais que haviam sido definidos originalmente no standard em 2003 [49], isto nas respetivas modulações ASK e O-QPSK, nas bandas de frequência 868/915 MHz.

Para ocorrer este aumento de número de canais, a atribuição de canal é executada através de uma combinação *channel number* e *channel page*. Para esta atribuição utiliza-se os 5 bits mais significativos (MSBs - *Most significant bits*) do *channel bitmap* de 32 bit que são utilizados como um valor inteiro para especificar 32 possíveis *channel pages*. Restando assim 27 bits do *channel bitmap* para especificar os *channel numbers* referentes a um *channel page* [11].

2.7.2 WIFI

Em 1997 foi aprovada a norma IEEE 802.11, utilizando uma modulação FHSS e a banda de frequência de 2.4 GHz, que permite uma taxa de transmissão de dados entre 1 e 2 Mbps. Desde esta data foram surgindo várias atualizações, com melhorias tanto nas taxas de transmissão como de segurança, onde houve necessidade de efetuar várias atualizações ao standard. Das várias atualizações existentes, é de destacar a 802.11a, 802.11b, 802.11g e a 802.11n.

Tabela 2.1: Bandas de frequência e taxas de transmissão [11]

PHY (MHZ)	Frequency band (MHZ)	Spreading parameters		Data parameteres		
		Chip rate (Kchip/s)	Modulation	Bit rate (Kb/s)	Symbol rate (Ksymbols/s)	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
868/915 (optional)	868-868.6	400	ASK	250	12.5	20-bit PSSS
	902-928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868-868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902-928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400-2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

A norma 802.11a, lançada em 1999, utiliza a modulação OFDM (Orthogonal Frequency Division Multiplexing), e utiliza a banda de frequência dos 5GHz. Devido ao uso desta frequência, consegue prevenir mais ocorrências de interferências do que os standards que operam a 2.4 GHz [50].

Também em 1999 surge a norma 802.11b onde com a utilização do DSSS e a modelação CCK (Complementary Code Keying), consegue atingir taxas de transmissão de 11, 5.5, 2 e 1 Mbit/s, isto numa banda de frequência de 2.4 GHz. Passado quatro anos, após atualização da norma 802.11b, surge o standard 802.11g que, em termos de aceitação e popularidade, aparece como uma solução substituta do 802.11b, operando na mesma frequência, e com compatibilidade com o standard 802.11b mas conseguindo atingir taxas de transmissão até 54 Mbits/s [51]. Já em 2009 surgiu a norma 802.11n, que funciona também na banda de frequência de 2.4GHz, compatível com as normas *b* e *g*. Neste standard, verificamos taxas de transmissão superiores, que podem atingir os 600 Mbit/s [52]. Para isto ser possível esta norma utiliza uma largura de banda 20 Mhz ou de 40Mhz, e com o HT (High Throughput) PHY presente neste standard [52].

2.7.3 IMC

O IMC (Inter-Module Communication Protocol) é um protocolo de comunicação, desenvolvido pelo LSTS, preparado para veículos autónomos [8], independentemente de serem terrestres marítimos ou aéreos. O protocolo foi concebido como *middleware* entre operadores, veículos ativos e sensores fornecendo uma camada transparente entre estes. O IMC é compatível com standards internacionais, com por exemplo o STANAG 4586 [53]. Para melhor compreensão de mensagens do IMC ver exemplo na Figura 2.13.

O IMC contem os seguintes tipos de mensagem:

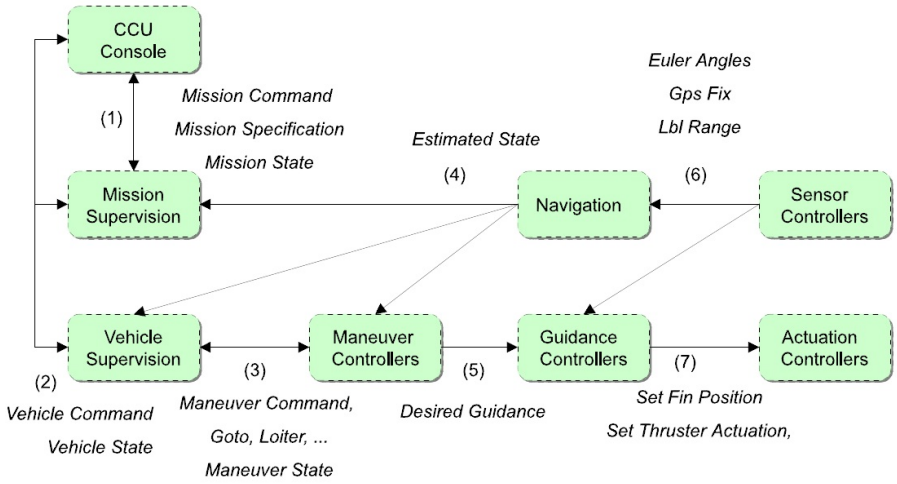

Figura 2.13: Diagrama exemplo de mensagens do IMC [8]

- Mensagens de controlo de missões;
- Mensagens de controlo de veículos;
- Mensagens de manobras;
- Mensagens de orientação;
- Mensagens de navegação;
- Mensagens de sensores;
- Mensagens de atuadores.

Com isto, é permitido a um dado veículo comunicar com várias estações terrestres ou mesmo ser operado por diferentes intervenientes no espaço de tempo. Este módulo tem a possibilidade de ser integrar com o Neptus (software desenvolvido e utilizado no LSTS), o que dá a capacidade a esta plataforma de coordenar vários veículos simultaneamente.

2.8 Componentes de um sistema de localização

Os sistemas de localização estão representados na Figura 2.14 [9], onde também é possível perceber o fluxo de informação entre os três módulos de um sistema de localização, sendo estes sistemas divididos nos seguintes componentes:

- **Estimar Distância / Ângulo:** Este módulo é responsável pela estimação da distância/ângulo entre dois nós da rede, disponibilizando esta informação para uso posterior dos outros módulos de um sistema de localização;

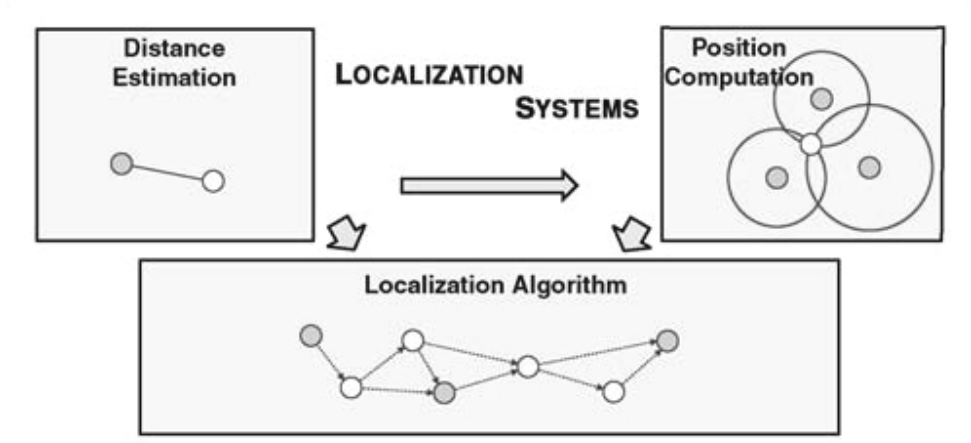

Figura 2.14: Componentes de um sistema de localização [9]

- **Calcular a Posição:** Este módulo têm a responsabilidade de calcular a posição do nó, recorrendo as informações previamente disponibilizadas das distâncias/ângulos e posições dos nós de referência;

- **Algoritmo de Localização:** Este módulo é o principal módulo de um sistema de localização. O qual indica de que forma é que são manipulados os dados disponíveis de forma a que seja possível estimar as posições da maioria ou de todos os nós;

2.9 Estimação da distância/ ângulo

A estimativa da distância/ângulo é um componente importante num sistema de localização, pois é utilizada para o cálculo da posição e pelo algoritmo de localização.

Existem vários métodos para realizar estas estimativas da distância/ângulo: uns com maior precisão, mas com maior consumo de recursos seja de hardware, energia ou processador, outros com menor precisão, usualmente disponíveis nos nós de sensores.

De seguida são apresentados alguns dos métodos possíveis para realizar este tipo de estimativa.

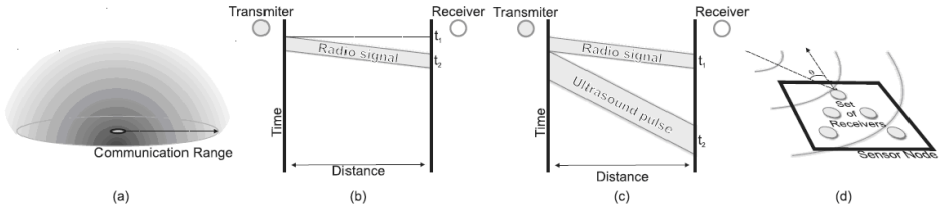

Figura 2.15: Métodos para estimar distância/ângulos [9]

2.9.1 Intensidade do sinal recebido - RSSI

Este método utiliza a intensidade do sinal recebido -RSSI (*Received Signal Strength indicator*) para estimar a distância entre dois nós, ou seja, medindo a potência do sinal no nó receptor, está estimativa é possível realizar porque o sinal é transmitido pelo nó emissor com uma determinada potência, a qual vai diminuindo à medida de o sinal se propaga.

A grande vantagem deste método é seu baixo custo, isto porque a maioria dos nós de sensores tem a capacidade de realizar esta medição da potência do sinal recebido, a desvantagem resulta da imprecisão dos resultados devido aos ruídos e interferências no sinal. Existem duas técnicas que utilizam o RSSI [54]

- **Conversão da potência do sinal em distância:** A teoria exemplifica que a potência do sinal diminui com o quadrado da distância. A unidade de medida utilizada para medir esta potência é o dBm (Decibel referido a um miliwatt) ou Watts (W). Foram realizados testes, que para um cenário onde todos os nós se encontram num campo plano, a uma altura de 1,5m e com um raio de comunicação de 10m, foram detetados erros de 2 a 3m [55].

- **Impressão digital (*fingerprinting*):** Esta técnica é composta por duas fases, uma primeira onde são recolhidas as medições de sinal em vários pontos, para cada um destes pontos são recebidos vários sinais. Essa informação forma a impressão digital desse mesmo ponto que, posteriormente, é guardada numa base de dados. Na segunda fase, um dado nó que se queira localizar recolhe os sinais que recebe nesse ponto, cria então a sua impressão digital que é comparada com as que estão na base de dados. A mais semelhante indica a sua posição. Esta técnica é mais precisa (chega a ter precisão de 1m() [54]), mas tem a desvantagem de ser necessário uma informação prévia [56, 54].

2.9.2 Medição por cronometragem - TOA/TOF/TDOA

Este são métodos que utilizam o tempo que o sinal demora a percorrer a distância que se pretende estimar. O método mais intuitivo é o TOA/TOF (*Time of Arrival or Fligth*), em que admite que a distância entre dois nós é diretamente proporcional ao tempo que sinal demora a percorrer de um nó para o outro. Este cálculo pode ser realizado pela fórmula:

$$d = S_r \times (t_2 - t_1) \tag{2.1}$$

em que:

- d é a distância;
- S_r é a velocidade de propagação do sinal (velocidade da luz);
- t_1 é o tempo em que o sinal foi emitido;
- t_2 é o tempo em que o sinal foi recebido.

Este método é de uma simplicidade elevada, mas é de difícil implementação devido à precisão do sincronismo necessário entre os relógios do sistema. Este é frequentemente utilizado, um exemplo é o GPS.

O método TDOA (*Time Difference of Arrival*) utiliza a diferença de tempo que um sinal demora a chegar vários nós, ou então a diferença de tempo que vários sinais demoram a chegar a um determinado nó. Em RSSF é usual encontrar dois sinais a serem transmitidos, ou seja, é necessário que conferir a capacidade aos nós de enviar/receber dois tipos de sinais em simultâneo. Para ser possível aplicar este método, estes sinais têm de ter velocidades de propagação diferentes, tais como rádio/ultra-som [57] ou rádio/acústico [58]. Normalmente o segundo sinal usado é um sinal de áudio devido à sua propagação ser mais lenta.

Um exemplo que foi encontrado durante a pesquisa foi a utilização de um impulso de ultra-som simultaneamente com o sinal rádio [55]. Para este exemplo os nós medem a diferença entre o tempo de chegada de cada um dos sinais. A fórmula para se realizar este cálculo é a seguinte:

$$d = (S_r - S_s) \times (t_2 - t_1) \tag{2.2}$$

em que:

- d é a distância;
- S_r é a velocidade de propagação do sinal de rádio (velocidade da luz);
- S_s é a velocidade de propagação do sinal de ultra-som;
- t_1 é o tempo em que o sinal de rádio foi recebido;
- t_2 é o tempo em que o sinal de ultra-som foi recebido.

Nesse mesmo exemplo os testes realizados demonstraram, erros por volta dos 2 a 3cm, no cenário em que os nós estavam distanciados 3m entre si [55].

Outra investigação revela resultados com erros superiores, a rondar os 23cm, num cenário em que os nós se encontram a uma distância de 2m, entre si. Esta investigação utilizou um som acústico (audível) [59].

A desvantagem deste método é o curto alcance (usualmente de 3m) e facto de ser necessário hardware adicional aumentando assim o custo de cada nó.

2.9.3 Ângulo/ Direção de chegada - AOA/DOA

Nestes métodos, os nós de sensores utilizam o ângulo de chegada para estimar a sua posição. O ângulo pode ter como referência o próprio nó ou um segundo sinal recebido. Este método exige antenas complexas que, devido ao custo, são, de um modo geral, unicamente instaladas nos nós âncora ou em estações base [57, 60]. O ângulo de recepção é calculado com a informação do tempo de chegada do sinal a cada um dos nós que pertencem a um conjunto de nós receptores separados uniformemente (mais do que três). Este método apresenta desvantagem em custo e

tamanho dos nós de sensores, pois requer hardware extra, assim como uma distância mínima entre os receptores.

2.9.4 Conectividade

Estes métodos são os mais simples, utilizam apenas a informação de ser identificado um nó no seu raio de ação. Esta informação indica que o nó identificado pelo outro se encontra a uma distância compreendida entre o zero e o alcance máximo do sinal (raio). Estas estimativas tentam analisar o grupo de nós no qual o nó está inserido e partir daí estimar a sua posição. Estes métodos como não requerem nenhum hardware exclusivo ou cálculos extra, tem a vantagem de ser os mais económicos e os mais simples, mas apresentam a desvantagem de apresentar um erro de metade do alcance do sinal na estimativa de distância, o que na maioria dos sistemas de localização não é viável [9].

2.9.5 Resumo

Após a leitura e a pesquisa realizada para esta seção 2.9 torna-se evidente que a escolha do método para realizar as estimativas das distâncias/ângulos entre nós têm uma influência na performance do sistema de localização.

Também é notório que usualmente é necessário pelo menos três distâncias estimadas para ser possível calcular a posição do nó. Se o único factor a ter em conta fosse a precisão das estimativas, não seria necessário existirem estes métodos todos, mas factores tais como o tamanho e custos (sejam por processador, hardware ou energéticos) de cada nó de sensores, são também factores a contabilizar e que influenciam a escolha do método a adotar ou seja, este método tem de ser escolhido de acordo com os recursos disponíveis e dos requisitos de aplicação.

Tabela 2.2: Comparação dos vários métodos usados para estimar as Distâncias / Ângulos [9]

Método	**Precisão**	**Distância máxima**	**Hardware extra**	**Desafios**
RSSI	2-4 m	Raio de comunicação	Nenhum	Variabilidade de interferências rádio
TOA	2-3 cm	Raio de comunicação	Nenhum	Sincronismo de relógio dos nós
TDoA	2-3 cm	2-10 m	Transmissor / Receptor de ultra-sons	Distância máxima de trabalho
AoA	5^o	Raio de comunicação	Conjunto de receptores	Implementação em nós pequenos
Conectividade	1/2 do raio de comunicação	Raio de comunicação	Nenhum	–

2.10 Cálculo da Posição

Quando já se dispõe das informações imprescindíveis em relação aos nós de referência, pode-se dar inicio ao cálculo da posição recorrendo a um método matemático, dos vários existentes para este fim. Aquando de tomar uma opção por um destes métodos deve-se ter em consideração quatro factores, dois com um peso maior na escolha, sendo estes a precisão dos resultados e a capacidade de processamento necessária, os outros dois factores são o consumo de energia e a quantidade de nós que influência diretamente o volume de dados para serem tratados.

De seguida serão explicados alguns destes métodos.

2.10.1 Trilateração e multilateração

A trilateração é um método usado em soluções como o GPS. Pode ser aplicado quando são conhecidas as posições de três nós de referência e a distância do nó do qual queremos estimar da sua posição à dos nós de referência. Representam-se os nós de referência graficamente por círculos centrados nas suas respetivas posições e com raio igual a distância até ao nó que se quer calcular a estimativa da sua posição, sendo essa estimativa à do ponto correspondente à intercepção dos círculos. Recorrendo a geometria, obtêm-se a formula:

$$\sqrt{(x-x_i)^2+(y-y_i)^2} = d_i \tag{2.3}$$

em que:

- (x,y) representam as coordenadas da posição que se pretende calcular;
- (x_i,y_i) representam as coordenadas da posição do de referência i;
- d_i é a distância do nó de referência i à posição que se pretende calcular.

Sendo assim, obtemos um sistema de tantas equações como de nós referência, com duas incógnitas. Como este é um sistema linear de equações, pode ser resolvido por vários métodos algébricos, tal como a eliminação de Gauss.

Na prática, existem erros nas distâncias calculadas, assim como as posições dos nós de referência usualmente contêm informações erróneas, sejam por erros introduzidos pela aparelhagem de medição quer por fenómenos de propagação dos sinais. Isto significa que os círculos não se interceptam unicamente num ponto, ficando assim um conjunto de elevadas soluções possíveis, o que faz que seja preciso recorrer a métodos mais complexos para se obter o resultado.

Um desse métodos é o dos quadrados mínimos, que pretende minimizar o quadrado da diferença das distâncias entre a distância d_i e o valor da distância estimado e_i. Este método pode ser representado matematicamente por:

$$f(loc) = min(\sum_{i=1}^{n}(\|loc - ref_i\|)^2) \tag{2.4}$$

em que:

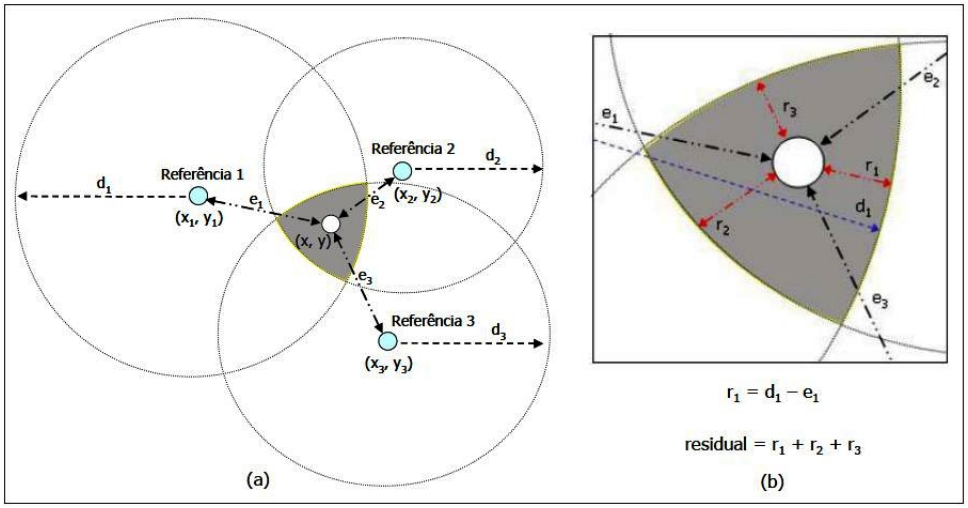

Figura 2.16: Trilateração considerando erros: (a) a área escura representa as soluções possíveis; e (b) solução do problema pelo valor residual mínimo [9].

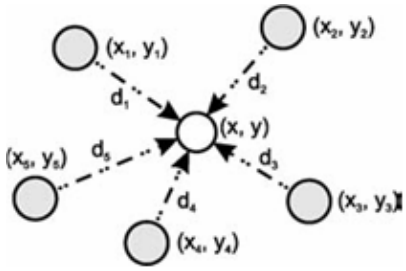

Figura 2.17: Multilateração [9].

- *loc* é a localização do ponto que se pretende calcular;
- ref_i é a localização do ponto de referência *i*;
- d_i é a distância calculada da referência *i* ao ponto pretendido;
- *n* é o numero de referências.

Foram encontrados outros métodos durante a preparação e elaboração deste trabalho, tal como a linearização de Taylor e o método de Gauss. A capacidade computacional disponível e a precisão pretendida influencia o método a escolher.

2.10.2 Triangulação

Este método recorre à informação de ângulos em vez de distâncias, ou seja, se um nó tiver capacidade de estimar o seu ângulo em relação a três referências (2.18a), então também tem capacidade de calcular a sua posição, isto com o recurso às relações trigonométricas. Este método também pode ser calculado se pelo menos dois nós referência puderem estimar o ângulo entre

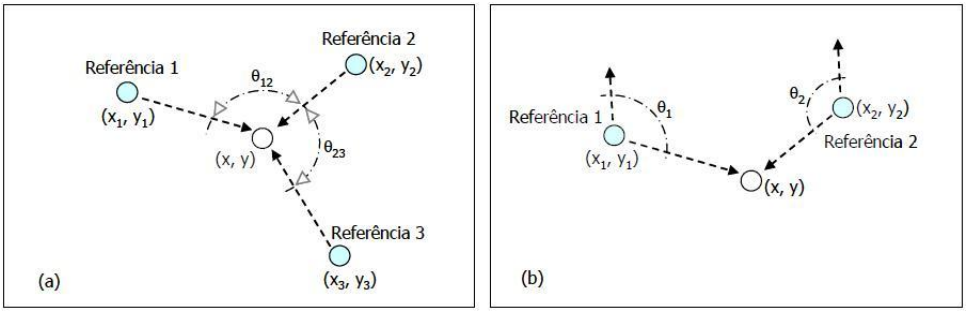

Figura 2.18: Triangulação (a) utilizando três referências um nó é capaz de calcular sua posição; e (b) dois nós podem calcular a posição de um terceiro nó [9].

uma referência até ao nó que se pretende localizar (2.18b), e essa mesma informação poder ser partilhada entre estes nós, sendo assim possível calcular a posição do nó pretendido. De notar que nesta segunda forma não é o próprio nó que calcula a sua posição [57, 60].

2.10.3 Bounding box

O método *bounding box*, também denominado como *min-max*, que utiliza quadrados teóricos em vez de círculos utilizados na multilateração [54]. Do recurso desta aproximação advém a vantagem de se reduzir de uma forma drástica os cálculos necessários, mas em oposição o resultado não ser tão preciso.

Para cada nó referência i é criado um quadrado (*bounding box*) centrado na respetiva localização (x_i,y_i) e o comprimento dos lados igual à $2d_i$. A seguir calcula-se a interseção destes quadrados, a vantagem deste método é que é possível realizar este cálculo sem a necessidade da utilização de operações com vírgula flutuante e de uma forma fácil, sendo o resultado um quadrilátero, e em que a posição desejada é calculada como o centro deste. A figura é um exemplo deste método.

2.10.4 Posição central em relação às referências

Este é um método em que não é necessário informação sobre distâncias ou ângulos. Simplesmente se considera que a posição do nó é o centro de todos os seus pontos de referência, ou seja a posição pode ser calculada pela equação [61]:

$$(x,y) = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n} \right) \tag{2.5}$$

É de notar que este método tem algumas vantagens: para além de não necessitar de muitas informações, requer pouco poder computacional e de ser necessário apenas $2n+2$ (em que n é o número de nós referência) operações para estimar a posição. Apresenta como desvantagem a pouca precisão.

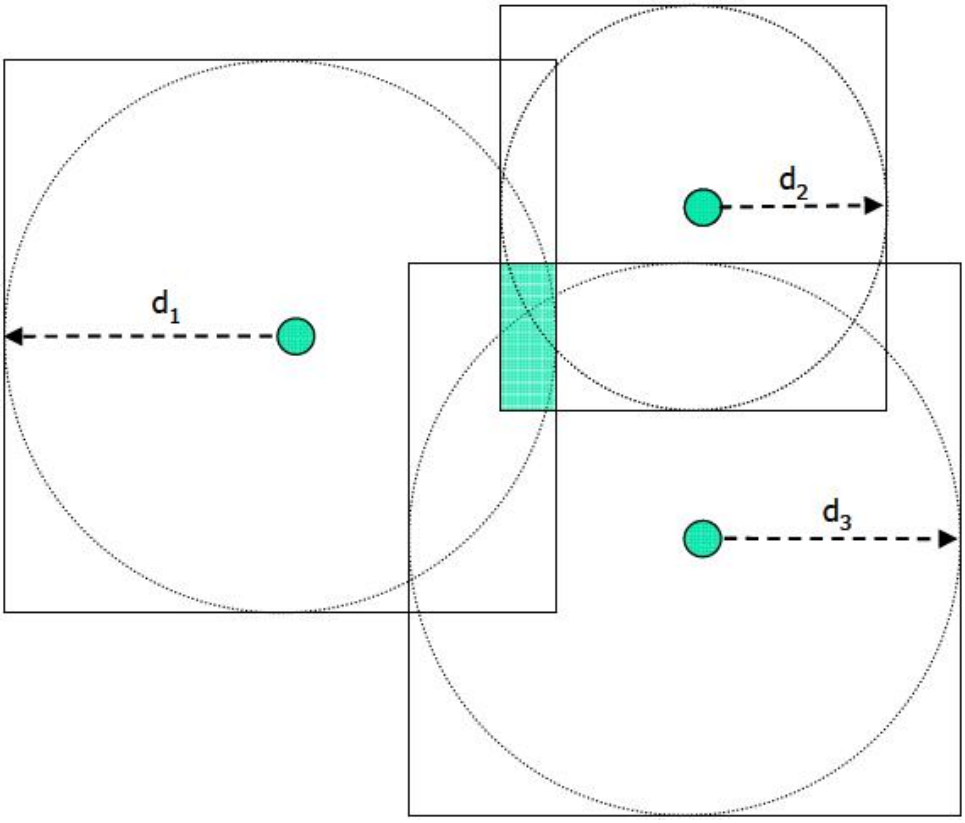

Figura 2.19: *Bounding box* [9].

2.10.5 Resumo

Nesta seção 2.10 foram apresentados possíveis métodos para calcular a posição dos nós de sensores, e de uma forma homónima a estimativa de distância / ângulo. Esta escolha interfere com o desempenho do sistema de localização, e tem que ter em consideração outros factores (já descritos anteriormente).

2.11 Algoritmo de localização

Este componente do sistema de localização é o principal e o que define como é que os dados obtidos, tanto pela estimativa de distância/ângulo e do cálculo de posição, sejam tratados de forma que seja possível determinar a localização do máximo dos nós da RSSF, caso não seja possível determinar de todos os nós.

Este componente pode ser classificado em algumas categorias, tais como:

- Distribuído ou centralizado: ou seja, a localização dos nós serem calculados de uma forma distribuída pelos nós da rede [62, 63, 64], ou de forma centralizada num sensor ou servidor [65, 66];

Tabela 2.3: Comparação dos métodos usados para cálculo de posicionamento

Método	**Nº de referências**	**Distribuído**	**Ângulo?**	**Complexidade**	**Desafios**
Trilateração	3	Sim	Não	O(1)	Susceptível à imprecisões nas distâncias
Multilateração	n $\geq$ 3	Sim	Não	O(n^3)	Complexidade computacional
Triangulação	3	Não	Sim	O(1)	Necessita de hardware extra
Probabilística	n $\geq$ 3	Sim	Não	O($3d^2$), d = grelha	Espaço de armazenamento e complexidade computacional
Bounding Box	n $\geq$ 2	Sim	Não	O(n)	Erro na posição final
Posição Central	n $\geq$ 1	Não	Não	O(n)	Erro na posição final

- Com ou sem infra-estrutura: se há necessidade de criar uma infra-estrutura previamente para o correcto funcionamento do sistema de localização [63, 67, 57] ou se não existe essa necessidade [62, 64];
- Posicionamento absoluto ou relativo: as localizações podem estar em coordenadas globais, tais como latitude e longitude [62, 63], ou em relação a um ponto da rede, como por exemplo um nó de sensores [68, 69];
- Um ou múltiplos saltos: se os nós apenas necessitam dos seus vizinhos para estimarem as suas localizações [63, 67], ou se necessitam de informação para além dos seus vizinhos [62, 64];
- Local aberto ou fechado: se o sistema é mais apropriado para locais abertos [63], ou para locais fechados [67, 57].

2.11.1 Assumption Based Coordinated (ABC)

Neste algoritmo são atribuídas a um nó as coordenadas zero e, com cálculo através das fórmulas abaixo apresentadas, os dois primeiros nós a receber a transmissão do nó inicial assumem as suas coordenadas. Estas fórmulas cumprem as condições geométricas das posições.

$$ref_0 = (0,0) \tag{2.6}$$

$$ref_1 = (d_{01},0) \tag{2.7}$$

$$ref_2 = (\sqrt{d_{01}^2 + d_{02}^2 + d_{12}^2}, \sqrt{d_{02}^2 - x_2^2}) \tag{2.8}$$

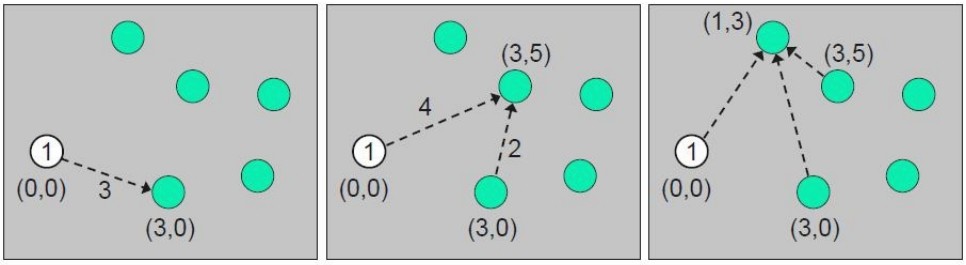

Figura 2.20: ABC. (a) um nó inicia-se como referência e um segundo nó assume posição, (b) um terceiro nó assume a sua posição baseado nos dois primeiros nós, (c) com 3 referências o sistema não é mais indeterminado [9].

A partir do momento em que temos a posição de três nós, o sistema deixa de ser indeterminado e já é possível recorrer a algoritmos mais usuais. É de notar que neste algoritmo não se recorre a uma infra-estrutura [70].

2.11.2 Triangulation via Extended Range and Redundant Association of Intermediate Nodes (TERRAIN)

O TERRAIN é uma evolução ao algoritmo ABC, onde a ideia inicial é que o algoritmo ABC seja iniciado por nós âncoras. Um nó, a partir do momento que recebe coordenadas de pelo menos quatro âncoras, pode calcular a sua posição recorrendo ao método da triangulação. É possível aumentar a precisão do cálculo posicional conforme o nó recebe informação de mais âncoras. Este algoritmo pode ainda aumentar mais a precisão se se implementar um processo interativo, onde as medições de alcance e as posições estimadas dos nós vizinhos são utilizados para recalcular a posição de cada nó, em que a condição de paragem deste processo fosse a convergência das posições dos nós [9].

2.11.3 AD-hoc positioning system (APS)

Este algoritmo tem duas fases. Uma primeira onde cada nó de sensores deve estimar a sua posição relativamente às âncoras. E uma segunda onde os nós recorrem a multilateração para calcular a sua posição, podendo esta fase ser realizada de três formas: Dv-hop, Dv-distance e Euclidiano. A vantagem deste algoritmo é de necessitar um número de reduzido de âncoras [64].

O Dv-hop é iniciado pela propagação da informação das localizações das âncoras, e depois cada nó armazena a informação da posição e o número de saltos para cada âncora. Quando uma âncora recebe informação doutra , tendo esta informação saltado por vários nós, então a âncora tem informação para realizar o cálculo do tamanho médio de um salto, ou seja, divide a distância Euclidiana pelo número de saltos. Este valor médio é então utilizado como factor de correção, sendo que o factor de correção de cada âncora só deve influenciar apenas os nós mais próximos.

Um nó só tem capacidade de passar a distância, que inicialmente se encontra em número de saltos, para metros após a recepção de um factor de correção e podendo, a partir desse momento, passar para a segunda fase do algoritmo. O Dv-distance é em tudo semelhante ao Dv-hop. No entanto, enquanto o Dv-hop utiliza o número de saltos, o Dv-distance utiliza a soma das distâncias estimadas entre os nós.

O terceiro método enunciado neste algoritmo (Euclidiano), funciona com a distância euclidiana real dos nós às âncoras. Para que este nó possa calcular a sua posição, é necessária a informação da distância de outros dois nós à uma âncora, da distância entre eles e da sua distância a estes nós, para aplicar o teorema de Pitágoras aos triângulos gerados pelas distâncias [9, 64].

Para além da vantagem já referida, de serem necessárias poucas âncoras para este algoritmo funcionar, quando se utiliza o Dv-hop também não se acumulam erros de medição entre nós. No entanto, devido à forma como se propagam as distâncias, principalmente nos métodos Dv-hop e Dv-distance, o cálculo da posição recorre a informação inconsistente, o que leva a um sistema com erros de localização elevados.

2.11.4 Hop-Terrain

Este algoritmo é composto por duas fases. A primeira designada de Hop-Terrain é parecida com o Dv-hop, sendo a segunda fase uma fase de refinamento que começa quando todos os nós já possuem uma estimativa da sua localização. Neste processo interativo, em cada interação os nós interagem com os seus vizinhos diretos, e calcula a sua posição de acordo com as informações da interação anterior. Sendo que tem de associar um nível de confiança a posição estimada, e de acordo com esse nível de confiança, decidir se devem ou não transmitir esta nova informação para os seus vizinhos [9].

2.11.5 Recusrsive Position Estimation (RPE)

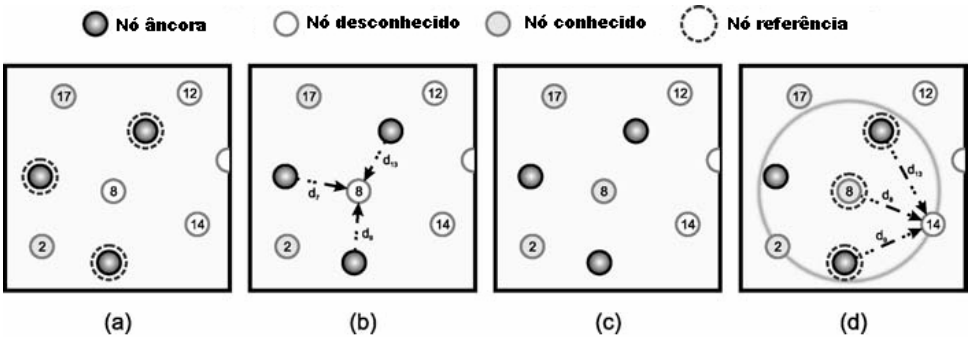

Figura 2.21: Exemplo e fases do RPE. (a) Inicialmente, o nó escolhe os seus nós de referência, então (b) este nó calcula a sua distância a cada um dos nós de referência, (c) calcula a sua posição usando triangulação, e (d) emite a sua estimativa de nova posição para ajudar os outros nós [9].

O RPE funciona de forma análoga à fase de refinamento ao Hop-Terrain, ou seja, os nós utilizam as informações dos nós vizinhos para estimarem a sua localização. Neste algoritmo cada

nó de sensores escolhe um mínimo de quatro referências entre os seus vizinhos diretos, e partir da informação dessas referências calcula a sua localização. A novidade deste algoritmo está no facto de atribuir um nível de confiança ao nó, sendo esse valor baseado no residual do cálculo da localização, determinando assim que referência com baixo nível de confiança, ou seja, as que não convergiram, não intervem no cálculo da localização da posição dos seus vizinhos [62].

Este algoritmo apresenta como vantagens a forma rápida como os nós calculam a sua posição, bem como o facto de ser distribuído e de não necessitar de infra-estrutura. Como desvantagem apresenta a propagação de erro localização, ou seja, quanto mais afastado estiver um nó das referências, maior será o seu erro de localização. Este algoritmo é mais apropriado para ambiente e a sua operação é realizada em multi-salto [9].

2.11.6 Directed Position Estimation (DPE)

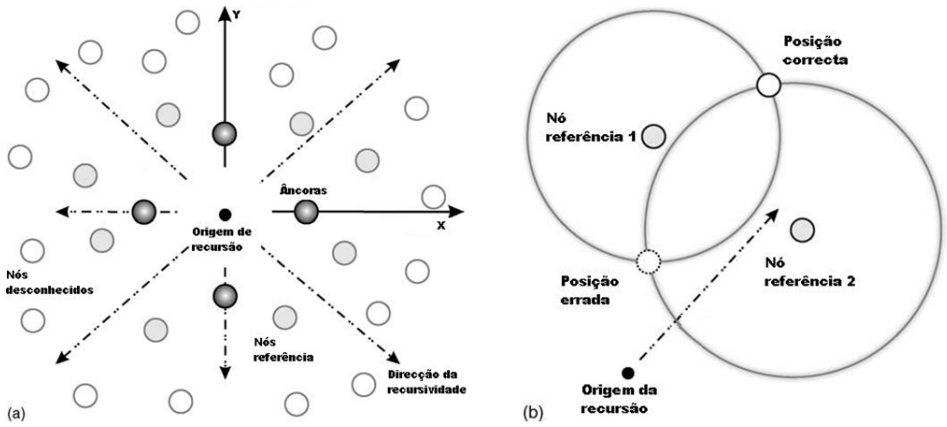

Figura 2.22: (a) O DPE a fazer uma recursão de localização dirigida. (b) A estimativa de posição utilizando apenas dois nós vizinhos de referência. Um par de resultados possíveis, soluções do sistema. A posição correta do nó é o ponto mais distante da origem recursão [9].

É possível fazer recursividade de localização de um ponto e fazer com que siga uma determinada direção conhecida, tal como demonstrado na figura 2.22a, acrescentando algumas restrições a um algoritmo recursivo como por exemplo o RPE apresentado anteriormente [9].

A partir do momento em que se consegue garantir este comportamento, torna-se possível então realizar a estimativa da localização de um nó com o recurso apenas a dois nós vizinhos. Como demonstrado na figura 2.22b, quando um nó recorre apenas a informação de dois nó para estimar a sua localização, o sistema encontra dois pontos possíveis para a sua estimativa. Mantendo-se a direção da recursividade, torna-se possível escolher qual das poções é a correta, sendo essa localização a mais distante da origem da recursão, sendo está a base do DPE [71].

Tal como apresentado na figura 2.23, este algoritmo divide-se em quatro fases, a primeira apresentada na figura 2.23a, que demonstra o início deste algoritmo que é iniciado por uma âncora. A segunda fase é quando um nó determina as suas duas referências e estima as distâncias entre si a cada uma destas referências, tal como apresentado na figura 2.23b. Na terceira fase o nó calcula

uma estimativa para a sua posição (figura 2.23c). Na quarta e ultima fase o nó transmite a sua localização estimada para os nós vizinhos (figura 2.23d).

Assim apercebe-se que o DPE funciona do centro da RSSF para as suas extremidades. A abordagem apresentada neste algoritmo torna o sistema de localização capaz de ser implementado numa RSSF de baixa densidade. Para além da forma como a recursividade é controlada, é capaz de induzir a um sistema com erros inferiores e mais previsíveis. Mas tal como acontece com o RPE este algoritmo apresenta a desvantagem da propagação do erro de localização.

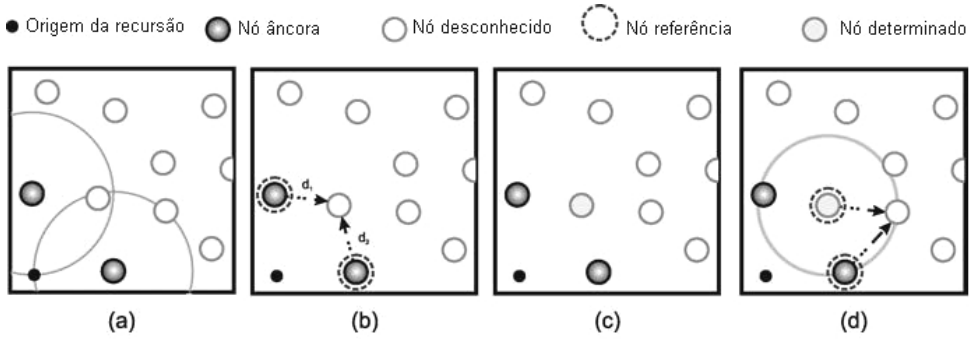

Figura 2.23: Exemplo e fases do DPE. (a) Primeiro, o nó âncora começa a recursividade. (b) Em seguida, um nó determina os seus pontos (dois nós) de referência. (c) estima a sua posição, e (d) então torna-se uma referência pela difusão dessa informação [9].

2.11.7 Localização com uma âncora móvel (LMB)

Alguma literatura recente propõe o uso de âncoras móveis para estimar as posições dos nós de uma RSSF [66, 72]. Este algoritmo baseia-se na premissa que uma âncora móvel tenha capacidade de possuir o conhecimento da sua localização, por exemplo estar equipada com GPS, e com possibilidade de se deslocar pelo meio da RSSF, sendo possível esta âncora ser desde um operador humano, a um veículo não tripulado, a um avião ou mesmo um robô. Um dos algoritmos propostos utiliza âncoras móveis para permitir que cada nó estime a sua localização [72], entretanto o outro algoritmo proposto, utiliza a própria âncora para estimar as posições dos nós da RSSF [66]. Aqui, o primeiro algoritmo enunciado é mais simples, pois uma vez a rede disseminada, a âncora móvel desloca-se pela RSSF difundindo a informação da sua localização. Ou seja, quando um nó desconhecido recebe pelo menos três mensagens da âncora, com recurso a aproximação probabilística, estima a sua localização baseado nas coordenadas recebidas e estimativas das distâncias obtidas pelo RSSI. Desta forma, a RSSF apresenta um custo nulo, uma vez que os nós não necessitam de enviar toda a sua informação, excetuando a âncora.

O facto deste algoritmo utilizar a informação do mesmo nó (âncora móvel) para calculo da estimativas, induz a vantagem do erro da posição ser relativamente baixo e prevenir a propagação do erro de localização, sendo apenas necessário o GPS, ou outro sistema de localização, para o nó móvel. Dependendo da mobilidade da âncora, da sua trajetória, influencia o tempo do cálculo da estimativa da posição, que pode ser longo, isto devido ao facto de um nó só calcular a sua posição

quando uma âncora passa perto da sua localização, sendo mesmo possível haver nós dos quais a âncora nunca passe perto o suficiente para estes poderem realizar o cálculo da sua posição.

Na figura 2.24 são apresentadas três possíveis trajetórias para o nó móvel, sendo que a trajetória da âncora móvel influencia diretamente as estimativas das localizações. Quanto menos rectilínea for a trajetória, mais fiáveis serão as estimativas, isto devido que ao facto de quanto menos colinearidade existir entre os pontos de referência, menor irá ser o erro da estimativa [17].

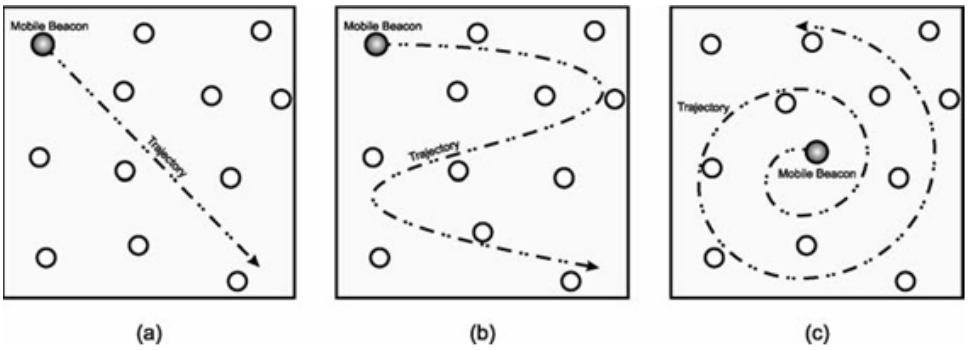

Figura 2.24: Operação e trajetórias possíveis para a localização com uma âncora móvel. (a) A âncora móvel que se desloca ao longo da RSSF em linha recta. (b) A trajetória menos rectilínea. (c) Uma trajetória em forma de espiral [9].

2.11.8 Multidimensional Scaling (MDS)

O MDS é um algoritmo que utiliza uma técnica designada da mesma maneira que o algoritmo que permite aproximar os dados estimados dos dados reais. Consiste, numa primeira fase, na ideia que um nó (*starting node*) possa fazer *broadcast* até que esta mensagem seja recebida pelo menos por três âncoras, designadas de *ending nodes*. Estas âncoras devolvem a mensagem para o nó inicial com as suas posições e da rota entre uma dada âncora e o nó inicial. Numa fase seguinte, após o nó inicial ter recebido as mensagens devolvidas pelas âncoras, este calcula a sua posição utilizando a multilateração com a informação das distâncias (acumuladas entre as rotas) até às âncoras. Numa ultima fase recorrendo ao MDS, o algoritmo ajusta as posições estimadas das âncoras às reais, assim como a de todos nós das rotas [9, 73].

2.11.9 Global Positioniing System (GPS)

O GPS é o método mais conhecido, e é um sistema composto por 24 satélites que estão em órbita em torno da Terra, em que cada um se encontra a uma altura de 20,2 quilómetros e realiza duas voltas completas por dia [63, 74]. As órbitas foram escolhidas de forma que seja possível que em qualquer lugar da Terra se consiga comunicar com pelo menos quatro dos satélites. Um receptor GPS tem a capacidade de receber a informação, que está a ser transmitida constantemente pelos satélites, e com ela calcular a sua distância aos satélites comunicáveis utilizando o TOA para esse efeito, e após isso estimar a sua posição com recurso a triangulação.

Uma solução seria então equipar todos os nós da RSSF com receptores GPS, onde se teria a vantagem da relativa precisão. Mas tal solução teria como desvantagens [55, 64, 65, 68]

- aumento de custo dos nós;
- aumento da dimensão dos nós;
- a impossibilidade do uso do sistema quando não existe visibilidade com só satélites (como por exemplo em locais fechados, debaixo de água, etc);
- o aumento do consumo energético.

Devido aos inconvenientes enunciados, geralmente o recurso a receptores GPS é limitado a um número reduzido de nós, como por exemplo âncoras.

2.11.10 Cricket Location Support System

O algoritmo *Cricket* utiliza o método TDOA para estimar as distâncias entres nós e as âncoras, para isso ser possível as âncoras têm de ter a capacidade de enviar um sinal RF e um sinal ultra-som em simultâneo. Após um nó ter informação suficiente utiliza a multilateração para calcular a sua localização. Apesar deste algoritmo ter sido projetado para nós móveis e localizados num espaço fechado, pode ser perfeitamente utilizado por uma RSSF com nós estáticos [67].

Para que o *Cricket* funcione corretamente é necessário criar uma infra-estrutura de nós âncora que possibilite que todos os nós tenham no seu alcance de comunicação pelo menos três âncoras. Isto pode ser visto como uma desvantagem deste método, mas em contra partida este método é bastante preciso e capaz de funcionar em ambientes fechados.

2.11.11 Resumo

Como referido anteriormente o algoritmo de localização é o principal componente de um sistema de localização, pois é este que define como a informação é transmitida entre nós assim como manipulada.

Com o aparecimento das RSSF os sistemas de localização foram amplamente estudados, com um ênfase nos de multi-salto, permitindo que, recentemente, fossem propostos outros algoritmos de localização focados noutras questões, tais como os erros e a sua propagação, o recurso a *hardware* extra, a necessidade de criar uma infra-estrutura, etc [9].

2.12 Resumo

Neste capítulo foi realizada uma introdução aos diferentes tipos de equipas robóticas, introduzido o conceito de controlo cooperativo e a investigação já realizada nesta área, onde foi visto algumas abordagens possíveis para esta questão.

Tabela 2.4: Comparação de vários algoritmos de localização (1ª Parte) [9]

Algoritmo	**Necessidade de âncoras**	**Nº de referências necessárias**	**Atribuição de nível de confiança ao nó**	**Processo de refinamento**
ABC	Não	0	Não usa	Não
Terrain	Não	≥ 4	Não usa	Sim
APS	Sim	≥ 3	Não usa	Sim
Hop-Terrain	Não	≥ 3	Média dos vizinhos	Sim, na 2ª fase
RPE	Não	≥ 4	Resíduo de fórmula	Sim
MDS	Não	≥ 3	Não	Sim

Após isso é introduzido o conceito de MAV, e a razão porque se realiza os primeiros teste neste tipo de plataforma.

É também apresentado dois tipos de CPU utilizados neste tipo de aeronave, onde está apresentado as suas características e uma breve explicação da opção por estes equipamentos.

De seguida são vistos vários géneros de sensores e redes constituídas por sensores, e que forma se pode utilizar num teatro de operações.

Também é descrito em que ponto está a comunicação deste tipos de sistemas, que tipo de protocolos usualmente usados e uma descrição de alguns desses protocolos.

Finalmente é apresentado o estudo realizado em sistemas de localização, onde são descritos os seus componentes, e várias técnicas e métodos possíveis para implementação de cada um desses componentes, assim como uma comparação entre esse métodos.

Tabela 2.5: Comparação de vários algoritmos de localização (2ª Parte) [9]

Algoritmo	**Nº de âncoras**	**Cálculo de posição**	**Infra-estrutura**	**Posicionamento**	**Cenário**	**Multihop**
APS	≥ 3	Distribuído	Não	Absoluto	Exterior	Sim
RPE	5% dos nós	Distribuído	Não	Absoluto	Exterior	Sim
DPE	4	Distribuído	Não	Relativo	Exterior	Sim
LMB	1 móvel	Distribuído	Não	Absoluto	Exterior	Não
GPS	–	Distribuído	Sim	Absoluto	Exterior	Não
Cricket	Grelha	Distribuído	Sim	Relativo	Interior	Não

Capítulo 3

Definição do sistema

3.1 Apresentação do Problema

Como demonstrado nos Capítulos 1 e 2 existem inúmeras aplicações para este género de sistemas. O problema que se pretende resolver é interligação de um sistema de UAVs com uma RSSF, podendo a RSSF monitorizar vários elementos no teatro de operações, facilitando assim o controlo e a supervisão da missão.

Pretendendo-se então com os sensores espalhados criar uma rede de sensores tendo em consideração logo à partida que, devido à gestão de autonomia e de segurança destes sensores, estes não estarão sempre a transmitir dados, e que, devido às distâncias dos próprios elementos à *ground station*, poderá existir falhas de comunicação. Para colmatar estas falhas de comunicação, pretende-se integrar os UAVs e os outros elementos presentes no teatro de operações numa rede *Mesh*, para desta forma podermos garantir uma maior cobertura e redundância de serviço, como visto na Secção 2.7, acrescentando assim qualidade as comunicações e prevenindo falhas no sistema, na Figura 3.1 podemos ver uma ilustração deste cenário.

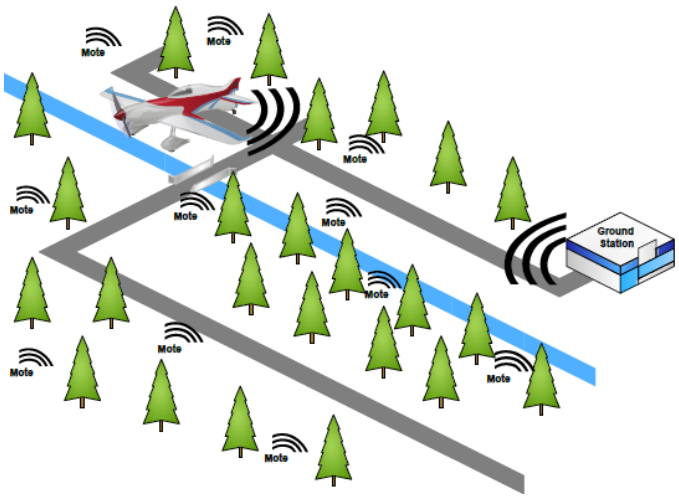

Figura 3.1: Ilustração do cenário descrito na secção 3.1.

Inicialmente pretende-se realizar comunicação entre um UAV (sensor dinâmico) e MOTES (sensores estáticos), utilizando o protocolo ZigBee por ser o protocolo utilizados pelos sensores estáticos referidos, após este objetivo pretende-se expandir esta rede de sensores com mais sensores dinâmicos, onde se lidará com a constante mudança da posição dos sensores e das adversidades inerentes a esse facto. Com esta ligação, temos então a possibilidade de identificar a localização deste sensor dinâmico, sem ser por meio de do métodos já implementados nestas aeronaves, o que poderá ser útil para *fail-safe mode*, também permitimos que estes sensores dinâmicos comuniquem entre eles até um conseguir comunicar para *ground station*.

3.2 Processo de Engenharia de Sistemas

Para desenvolvimento do trabalho pretendido, será usada uma metodologia de engenharia de sistemas, para desenvolvimento, teste e validação. A Figura 3.2 demonstra a o processo de engenharia de sistemas que irá ser usado, recorrendo a norma 1220 do IEEE [10].

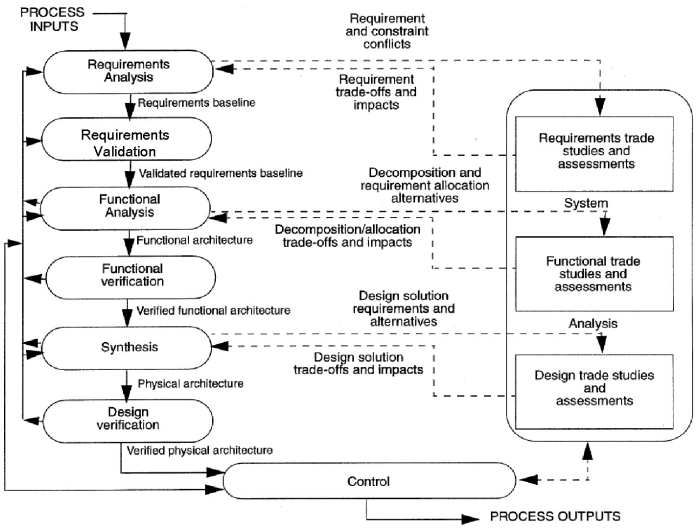

Figura 3.2: Processo de Engenharia de Sistemas [10]

O processo inicia com os respetivos *inputs*. Esta parte inicial incluí a identificação dos objetivos, requisitos, ambiente operacional, tecnologia disponível e restrições do sistema. Após a identificação dos pontos referidos, procede-se à sua análise e decomposição em requisitos funcionais e de performance, para assegurar que se percebeu totalmente as funções que o sistema deve realizar, assim como a performance a atingir.

A fase seguinte do processo consiste na análise das funções do sistema. Ao analisarmos as funções de alto-nível identificadas anteriormente, estas são decompostas em funções de mais baixo-nível, resultando assim uma arquitetura funcional do sistema.

Há que ter sempre em mente que todas as funções devem remeter a um requisito, garantindo que para cada função identificada está agregada a um respetivo requisito.

A arquitetura do sistemas ganha forma física na etapa da síntese do sistema, sendo aqui o sistema definido em termos de elementos físicos e de software, garantido assim que esta arquitetura aqui definida suporta as especificações do sistema. É realizada análise funcional nesta etapa, para verificar que as funcionalidades são cumpridas e realizar possíveis otimizações.

A solução é comparada com os requisitos iniciais para verificação e validação. Todo o processo de engenharia de sistemas é supervisionado por um sistema de análise e um método de controlo que incluí estudos *trade-off*, análises de design e eficácia, medidas de performance baseado em modelos, e ferramentas de teste garantindo assim ir ao encontro dos *inputs* do processo [10].

3.3 Metodologia de Operações com UAVs

Para melhor compreensão do problema, é fundamental observar e compreender o que acontece numa missão típica de UAVs, para que a solução vá o mais possível ao encontro das necessidades reais. A forma como as missões são conduzidas, os intervenientes, etc.

Existem três intervenientes principais numa missão, são eles:

- **Supervisor -** Responsável pela supervisão de toda a equipa. Este elemento é o único que, em qualquer momento da missão, sabe o estado de todos os UAVs ativos e sobre quem recaí toda a responsabilidade da segurança e coordenação da missão;
- **Piloto -** Responsável por supervisionar o UAV a si designado, atribuindo-lhe novas tarefas e supervisionar as que estão a decorrer. É também de sua responsabilidade monitorizar a integridade física do UAV durante a missão;
- ***Safety* -** Responsável pelo controlo manual, se for necessário, de um veículo aéreo.

Então como representado na Figura 3.3 o supervisor não tem qualquer poder direto para alterar planos de missão dos UAVs. Esta funcionalidade encontra-se nas consolas dos Operadores e os Pilotos podem-no fazer, pois têm controlo direto dos veículos. Ou seja, as ordens do supervisor têm de chegar ao resto da equipa. Para este efeito é usado um software externo: o TeamSpeak.

Utilizando esta ferramenta, todos os elementos conseguem interagir entre si, acatar as ordens do supervisor da equipa, e perceber as necessidades dos outros operadores. Neste momento não é possível de outra forma, pois cada UAV realiza a tarefa a si atribuída sem conhecimento dos outros agentes presentes no teatro de operações. Este facto aumenta a responsabilidade e o stress do supervisor da equipa.

3.4 Requisitos

Para que este projeto seja orientado da melhor forma, e o seu desenvolvimento siga constantemente em função dos objetivos pretendidos, forma recolhidos os requisitos necessários para que os objetivos fossem cumpridos. Requisitos a partir dos quais se podem então definir marcos intermédios para que no final se consiga atingir todos objetivos.

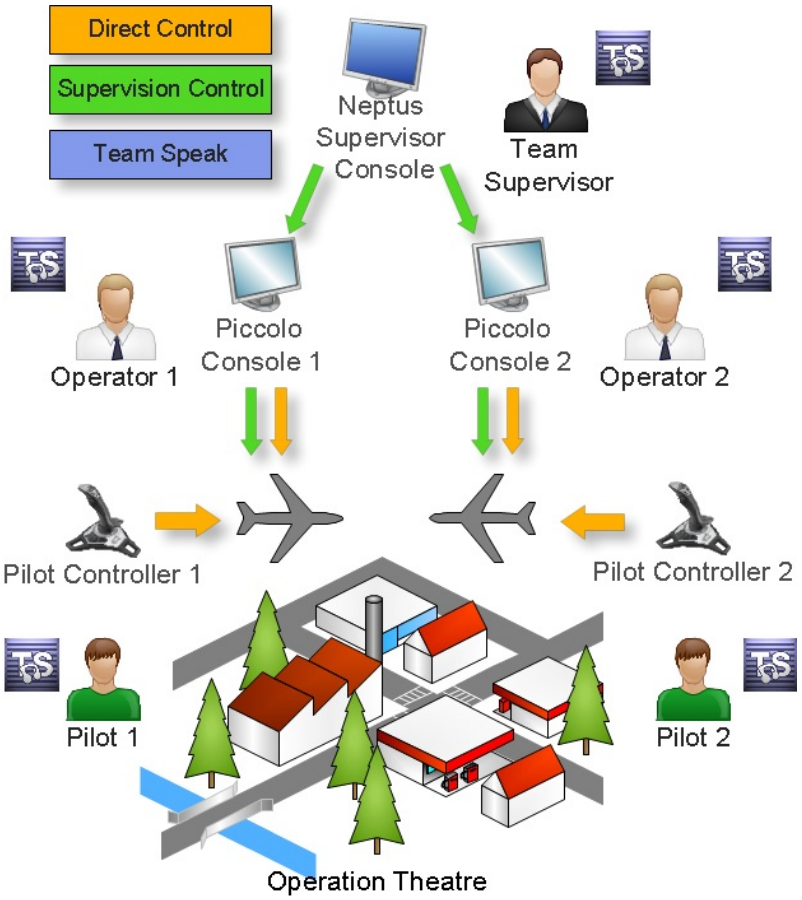

Figura 3.3: Esquematização dos elementos participantes numa missão de UAVs [1]

3.4.1 User story

O objetivo desta dissertação é de permitir que UAVs possam interagir com outros elementos presentes num teatro de operações. E que consiga utilizar uma rede *Mesh* para esse efeito. Então apresentemos um cenário de operação com estas funcionalidades disponíveis.

Tendo uma área de interesse para monitorizar, utilizamos uma rede de sensores sem fios para esse efeito. Pretendemos então recolher a informação detectada pelos nós desta RSSF, para isso utilizamos um UAV com uma mote a bordo, para esta interagir com as disseminadas na área de interesse. Com isto pretendemos que os dados recolhidos por esta rede sejam encaminhados para a mote a bordo do UAV.

A mote a bordo do UAV terá então que ser capaz de transferir as mensagens recebidas para a placa de processamento, para que esta possa reencaminhar as mensagens para a *Ground Station*.

Na *Ground Station* o operador de Neptus consegue visualizar os dados recolhidos conforme estes chegam. para melhor monitorização na *Ground Station* com informação transmitida pelos nós, onde se inclui o RSSI entre nós, é realizada a caraterização da rede disseminada na área de interesse, facilitando assim essa mesma monitorização.

3.4.2 Requisitos

Do cenário apresentado na *User story* podemos retirar vários requisitos que o projeto terá de cumprir. Sendo esses requisitos levantados os seguintes:

- **Recolher de dados -** Para que a recolha de dados de uma dada área de interesse seja possível, é preciso que os nós tenham integrados os devidos sensores para possibilitar a recolha da informação pretendida, podendo esta ser diversa e variada, dependendo da área de interesse em que se pretende monitorizar, assim como os objetivos de cada missão, podendo os objetivos ser muito diversificados e até mesmo opostos, como por exemplo objetivos de preservação, ou em cenários mais bélicos, deteção de presença de equipamento hostil.

- **Interação entre UAV e RSSF -** Sendo a RSSF usualmente constituída por nós de sensores designados de motes, e devido a forma como as suas comunicações foram desenvolvidas, a forma mais direta de permitir que um UAV interaja com uma rede desta características é, como descrito na *user story*, a integração a bordo de uma mote, a qual interage diretamente com a RSSF. Fica então a faltar a transmissão das mensagens desta mote a bordo para a unidade de processamento.

- **Transmissão dos dados -** Uma vez os dados recolhidos e disponíveis na unidade de processamento do UAV, é necessário fazer com que essa informação chegue ao operador no sistema que pode não se encontrar na área de interesse. Para isto ser possível, o UAV terá que a capacidade de retransmitir as mensagens recebidas para a *Ground Station*.

- **Visualização dos dados -** Após a recepção dos dados pela *Ground Station*, estes terão que ser disponibilizados para o utilizador do sistema, uma vez que estamos a referir UAVs do LSTS, a interface a ser utilizada será o Neptus, com o qual é preciso ter algum cuidado devido a sua arquitectura, tal como apresentado na secção 3.6.3.

- **Caraterização da RSSF -** Para que os dados recebidos pela *Ground Station* sejam percetíveis e que o operador do sistema possa realizar uma correcta monitorização, deverá ser possível realizar a caraterização da rede de sensores sem fios, ou seja, saber uma localização aproximada de cada nó de sensor da RSSF.

- **Facilitação da monitorização -** A maneira de como a localização e os dados recolhidos por cada nó de sensores é apresentado, deverá ser de tal forma, que o utilizador do sistema que pretende monitorizar a área de interesse tenha a maior compreensão possível dos dados recebidos, sem ter que realizar muitas acções e/ou configurações.

- **Comunicação entre UAVs e base utilizando a RSSF -** Uma vez a possibilidade da interação do UAV com a RSSF, esta ultima pode ser utilizada para servir de ligação entre dois UAVs ou mesmo entre um UAV e a *Ground Station*.

3.5 Métricas

Para se poder retirar as devidas conclusões do projeto proposto e verificar o seu sucesso, a sua fiabilidade, eficácia e eficiência, é necessário utilizar métricas. Com isto em mente, é apresentado nesta secção algumas métricas que poderão vir a ser utilizadas para validar e verificar o trabalho realizado. É de salientar que neste momento, as métricas são descritas de uma forma geral e subjetiva, pretende-se uma melhor definição das métricas após a validação dos requisitos.

Como estamos a lidar com comunicações, é necessário saber se existe redundância nas comunicações e, se existir redundância, quanta existe? Outra questão sempre importante nas comunicações relaciona-se com as perdas de pacotes de dados, logo, é preciso verificação e, se sim, a solução que irá ser apresentada contém técnicas para recuperação dos pacotes perdidos? Se tem, com que grau de fiabilidade? Como estamos perante uma rede *Mesh*, esta questão é resolvida com a verificação da eficácia e eficiência das técnicas utilizadas para ordenação dos pacotes.

No sistema de localização são realizadas estimativas da localização dos nós de sensores, isto faz com que seja obvio a precisão destas estimativas, sendo o cálculo do erro de estimativa mais uma métrica a ter em conta para a avaliação dos resultados que irão ser obtidos.

Finalmente uma questão importante e que define até que ponto o sistema pode ser utilizado, envolve a cobertura do raio das comunicações, pois esta influenciará a altura dos voos dos UAVs.

Tabela 3.1: Métricas Definidas

Métricas	**Como calcular**	**Condição de sucesso**
Redundância	Não aplicável	Não aplicável
Perda de Pacotes	Taxa de perdas de pacotes	Inferiores a 15 %
Estimativa de localização	Erro máximo da estimativa	Inferiores a 6m
Alcance máximo	Raio de cobertura	Superior a 100m

3.6 Software Operacional

Para além das plataformas e do hardware, utilizado no decorrer duma missão, é necessário que exista software que permita tirar partido das funcionalidades do hardware, sejam soluções comercias ou soluções criadas no seio do LSTS.

3.6.1 Poky

O poky é uma distribuição de linux para sistemas embebidos, desenvolvida pelo Yocto Project [75]. E a Igep de fábrica contém uma instalação de uma versão minimalista desta distribuição, já compilada para a arquitectura ARM.

Tal como a designação da distribuição indica (*Poky minimal*), esta é uma versão mais "leve"que não traz uma interface gráfica, nem gestores de pacotes. Mas o facto de fazer parte de uma comunidade relativamente grande, existe alguma documentação de suporte disponível na *wiki* desta comunidade.

3.6.2 TinyOS

As Motes conseguem correr um OS (Operating System) mas, devido às limitações inerentes ao género de micro controladores usados nesta família de sensores, este OS tem recursos limitados. O TinyOS suporta diretamente várias plataformas de Motes o que, conjugando este facto com a sua boa usabilidade, faz com que seja o mais reconhecido para utilização deste género de sensores.

Este OS permite a programação de cada um dos dispositivos, com um endereço único, sem ter de compilar o código fonte de cada vez que se realiza uma programação. O sistema TinyOS, as sua bibliotecas e respetivas aplicações são escritos em NesC (Networked Embedded Systems C), uma versão da linguagem C desenhada para programar sistemas embebidos. Para melhor compreensão do funcionamento, é de referir que nesta linguagem, os programas são construídos por componentes que são ligados para formar o programa completo. A ligação entre os componentes é realizada pelas suas interfaces, sendo que estas são bidireccionais e responsáveis pela especificação do conjunto de funções a implementar, tanto por quem fornece o serviço como por quem o utiliza.

3.6.3 Neptus

O Neptus é um software criado no LSTS que tem vindo ser adaptado, desde à algum tempo, para uma compatibilidade superior com as necessidades da equipa de UAVs presente no LSTS. Esta ferramenta é concebida em Java e está em constante expansão devido ao constante empenho e trabalho de uma equipa de desenvolvimento com a mesma designação que a *framework*. Esta ferramenta permite a interação com vários tipos de veículos (autónomos, semiautônomos ou teleguiados) e, apesar de se focar preferencialmente em controlo e supervisão marítima, o Neptus já apresenta uma consola para supervisão de UAVs, a qual poderá ser uma boa base para interação com o trabalho que irá ser desenvolvido nesta dissertação.

Para perceber melhor como esta ferramenta foi e é desenvolvida segue uma explicação da sua arquitectura (3.5 [1]). Para a manutenção e expansibilidade ser facilitada, o Neptus está concebido de forma modular, em que a independência entre cada módulo seja o máximo possível. Este software, no seu génese, é composto por um módulo base, que contêm os métodos mínimos exigidos para a implementação de uma consola, e vários módulos externos, denominados de *plugins*, que podem ser adicionados a consola base, habilitando assim esta de competências e capacidades novas. Sabendo que o Neptus é uma ferramenta com uma dinâmica de desenvolvimento elevada, esta arquitectura permite reduzir o máximo possível qualquer tipo de repercussões que novo código de *plugins*, possa causar a módulos previamente implementados e validados para utilização em missão.

O mesmo não pode ser dito sobre qualquer alteração realizada ao módulo base em que o Neptus se centra, pois como demonstrado na Figura 3.6 [1], qualquer alteração a este módulo poderá afetar de forma perjurativa consolas já validadas e consideradas estáveis.

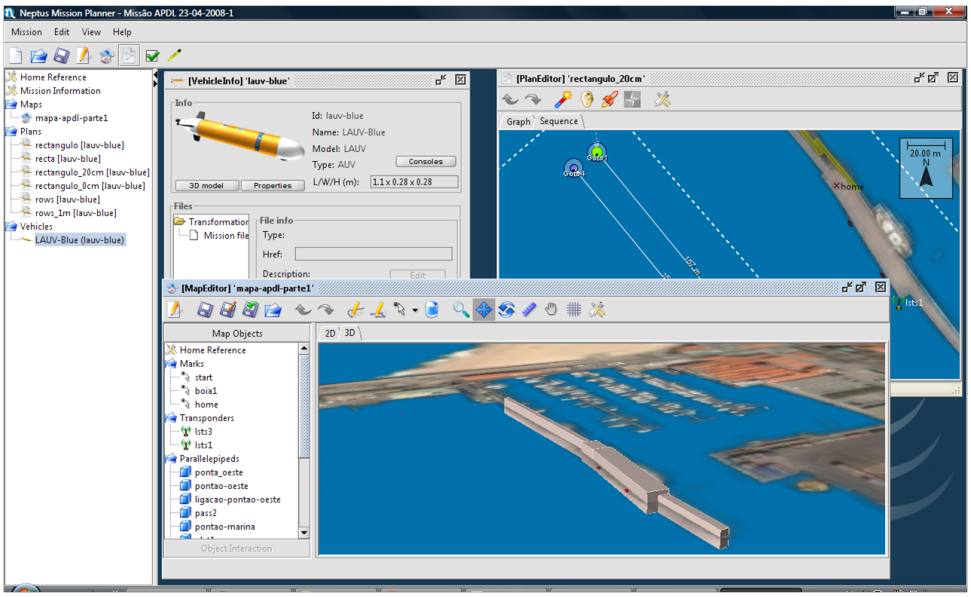

Figura 3.4: Exemplo da interface de uma consola Neptus [1]

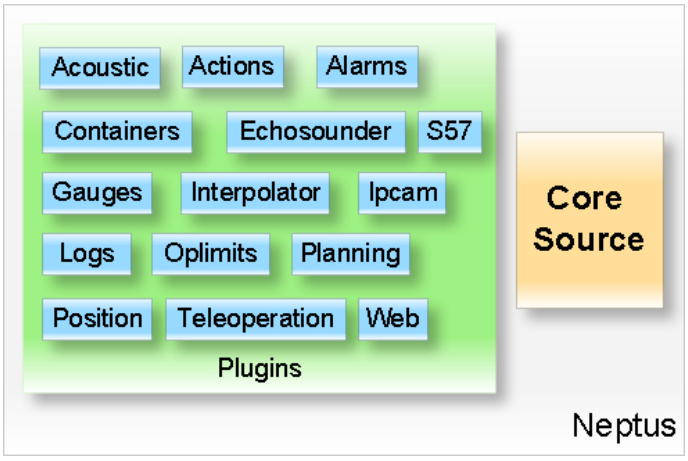

Figura 3.5: Arquitectura Simplificada do Neptus [1]

3.7 Abordagens possíveis para a implementação do sistema

Antes de ser realizada a escolha dos métodos, componentes e *software* a implementar, foi realizada uma selecção prévia do que foi pesquisado e apresentado no capítulo 2, sempre tendo em mente os requisitos do sistema, assim como algumas restrições impostas por material já disponível no LSTS. Sendo assim, de seguida será apresentada essa seleção prévia.

3.7.1 Aeronaves Não Tripuladas

Como referido na secção 2.4, para testar novas implementações e a sua validação, usualmente são usadas plataformas de pequenas dimensões.

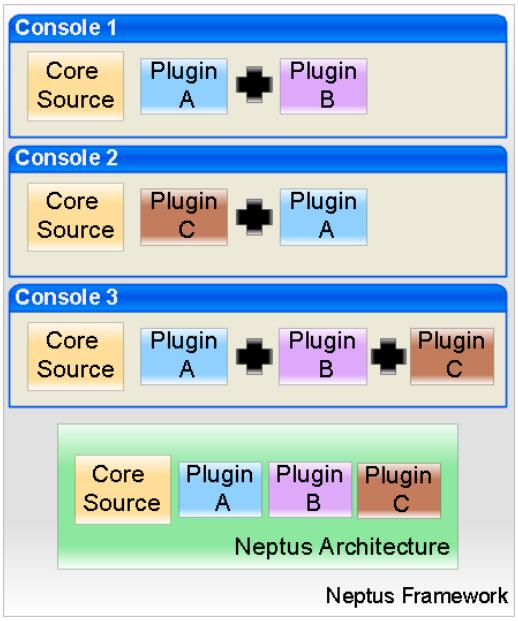

Figura 3.6: Exemplificação da constituição das consolas Neptus [1]

Atualmente o LSTS tem disponível dois tipos de MAVs: os Zagi, demonstrado na Figura 2.4 e o Cularis apresentado na Figura 3.7.

Outra aeronave muito fiável para testes de complexidade superior ou quando o *payload* necessário tem dimensões superiores às que as aeronaves antes referidas conseguem suportar, é o Pilatus. Podendo também esta plataforma ser utilizada para a implementação do trabalho a ser desenvolvido neste projeto.

Figura 3.7: Imagem de um Cularis

3.7.2 CPU

Tal como já referido na secção 2.5 existem, até a data, duas opções para este componente: o pc104 e a IGEP.

Devido às melhores características da IGEP, e pelo facto de ser uma *board* na qual ainda falta fazer muita integração nos sistemas de UAVs presentes nos AsasF, a escolha final deste componente recaíu sobre esta placa.

3.7.3 Motes

Neste componente a escolha foi imposta pelo que existia já disponível no LSTS, e verificado que, para o pretendido, este tipo de sensor contem características capazes de cumprir com os requisitos do sistema, sendo estes sensores sem fios as Moteiv's Tmote Sky.

A Tmote Sky é uma plataforma programável, que já traz integrado sensores, rádio controlado pela porta SPI que pode ser desligado para reduzir consumos quando a comunicação não é necessária, antena e um micro-controlador, uma FTDI para permitir a comunicação com um computador através de uma porta USB.

As características desta plataforma são as seguintes:

- transceptor sem fios Chipcon de 250kbps a 2.4GHz utilizando a norma IEEE 802.15.4;
- capacidade de interação com outros dispositivos que utilizem a norma IEEE 802.15.4;
- Microcontolador de 8MHz Texas Instruments MSP430 F1611 (10k RAM, 48k Flash, 16 bit RISC);
- ADC, DAC, Supervisor de voltagem e controladora DMA integrados;
- antena integrada com alcance máximo de 50 metro em espaços fechados e de 125 metros em espaços abertos;
- sensores de luz, humidade e temperatura integrados (opcionais);
- baixo consumo energético;
- rapidez em passar do modo *sleep* para o modo activo ($<6\mu$s);
- programação e coleta de dados através do uso da porta USB;
- possibilidade de expansão através de 16 pinos;

3.7.4 Reencaminhamento de mensagens

Ao analisar a *user story* é verificada a necessidade de encaminhar as mensagens para a mote que, de alguma forma (posteriormente explicado), consegue comunicar com a *Ground Station*, isto no primeiro cenário, assim como o reencaminhamento da mensagem recebida por esta mote

"base"para *Ground Station*. No segundo cenário torna-se também necessário o envio de uma mensagem por USB para uma mote e esta utilizar os meios já definidos no cenário anterior para que a mensagem chegue ao seu destino (*Ground Station*).

3.7.4.1 Ferramentas já disponíveis

Para evitar trabalho desnecessário, foi verificado quais as ferramentas disponíveis e quais as que podiam ser reaproveitadas para se atingir os objetivos pretendidos por este projeto, onde foram identificadas como principais ferramentas já existentes o Neptus e o TinyOS.

Do tinyOs foram vistas ferramentas como *Serial Forwarder* que permite encaminhar os pacotes recebidos pela mote "base"para uma *socket* do *host*, o *Listen* e o *Listen Raw* que possibilitam a leitura dos pacotes.

3.7.5 Estimar as distâncias entre motes

A primeira seleção para o método a utilizar para estimar a distância, foi baseada na escolha de um método que não necessitasse de *hardware* extra, o que nos levou a ficar com três métodos possíveis: o TOA/F, o RSSI e a conectividade. O método da conectividade devido ao seu erro foi imediatamente excluído.

Assim, a escolha recaiu entre os métodos TOA/F(2.9.2) e RSSI(2.9.1).

3.7.5.1 TOA/F - Time of Arrival or Flight

Esta técnica já foi apresentada na secção 2.9.2 e, como já descrito, utiliza a medição do tempo decorrido desde a emissão até à sua recepção no receptor. Para que isto aconteça entre motes é necessário que estas tenham os seus relógios sincronizados. Esta sincronização, é possível para RSSF por algoritmos de sincronização de tempo, como por exemplo *Flood Time Syncronization Protocol*.

Utilizando a aproximação que as ondas eletromagnéticas se propagam no espaço a velocidade da luz, e sabendo que esta velocidade é aproximadamente $3 \times 10^8 \, {}^{m}/_{s}$, isto no vazio, e de acordo com o teorema de Nyquist [76], a frequência de amostragem de um sinal analógico, para que este possa ser reconstruído com o mínimo de perda de informação, é no mínimo duas vezes a frequência máxima do espectro do sinal analógico que se pretende amostrar.

Sendo a frequência máxima do relógio da Tmote de 32768 Hz e, respeitando o teorema de Nyquist, a distância mais curta para qual esta técnica seria utilizável é aproximadamente 18310 metros, tal como justificado na equação 3.1, sendo esta distância mínima muito superior ao alcance máximo do transmissor da Tmote. O que torna este método uma escolha não viável para o sistema que se pretende implementar.

$$\frac{3 \times 10^8}{\frac{32768}{2}} \approx 18310m \tag{3.1}$$

3.7.5.2 RSSI - Radio Signal Strength Indicator

O uso do *Radio Signal Stength Indicator* tal como já referido na secção 2.9.1 para estimar a distância entre motes, leva a imprecisões, uma vez que qualquer desvio, mesmo que pequeno, pode ter como consequência um desvio elevado no valor obtido da estimativa da distância em comparação com a distância real. O uso desta técnica é usual em RSSF devido ao facto de não ser necessário *hardware* extra [68].

O funcionamento de transmissão de dados com o recurso a ondas de rádio (RF) consiste na emissão de uma onda eletromagnética de alta frequência, emitida pelo transmissor (Tx) e capturada por um receptor (RX). A utilização de uma onda de alta frequência permite que a comunicação seja estabelecida com baixa potência elétrica e com utilização de antenas de pequenas dimensões. O processo que permite inserir informação numa onda RF é designado de modulação [77], onde a onda de alta frequência é denominada de portadora e o sinal elétrico de modulante. A modulação é a alteração da característica da portadora de acordo com o sinal modulante. Sendo isto realizado no lado do transmissor, de forma inversa no lado do receptor é realizado a desmodulação que permite retirar a informação transmitida. Para a modulação ser realizada por sinais analógicos, existem três técnicas possíveis:

- modulação em amplitude (AM);
- modulação em frequência (FM);
- modulação em fase (PM).

Quando se trata da transmissão de dados digitais utiliza-se o chaveamento (*Keying*), que é a designação a modulação realizada a uma onda RF por um sinal digital. De forma análoga ao que acontece com a modulação por um sinal analógico, existem três técnicas para realizar o chaveamento:

- chaveamento em amplitude (ASK - Amplitude Shift Keying);
- chaveamento em frequência (FSK - Frequency Shift Keying);
- chaveamento em fase (PSK - Phase Shift Keying).

A Figura 3.8 exemplifica o funcionamento deste três tipos de chaveamento para a utilização de um sinal binário.

Devido à preocupação do consumo energético as RSSF utilizam rádios de muito baixa potência, podendo estas, na sua maioria, realizar FSK, PSK, ou uma combinação das três formas básicas de modulação. a transmissão é realizada com uma potência máxima de 10mW (10dBm) [USA], 1mW (1dBm) [UE]. O receptor usualmente consegue identificar sinais de muito baixa potência, conseguindo receber até sinais com uma potência de -100dBm (10^{-10}mW). Sendo o dBm

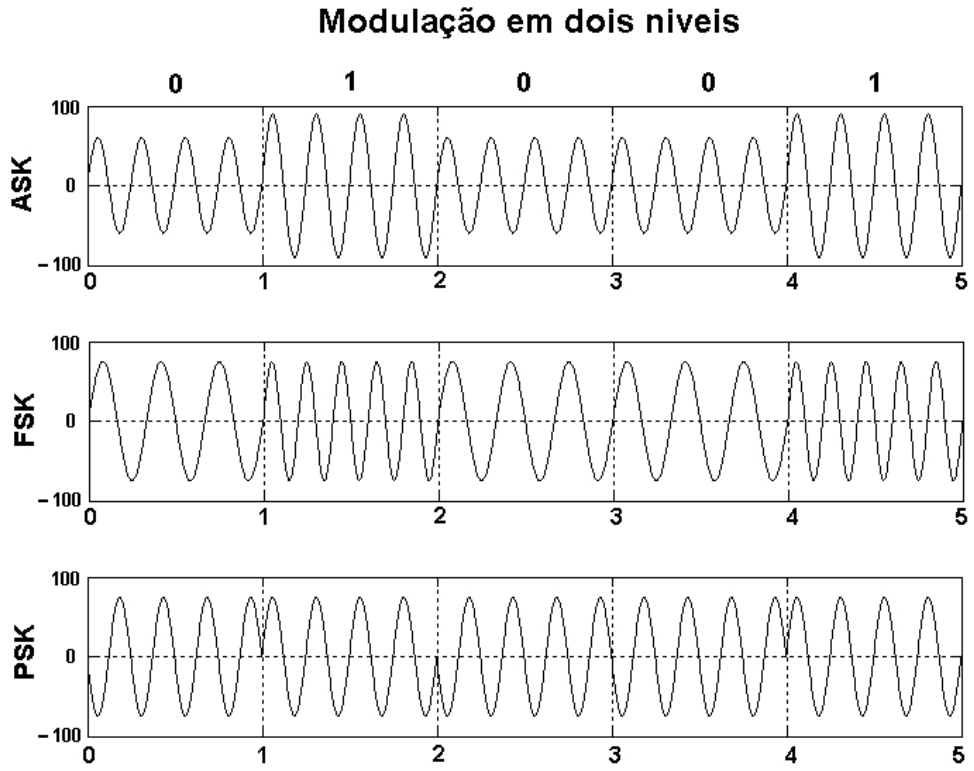

Figura 3.8: Modulação com um sinal binário [9].

uma unidade referida em decibéis a uma potência de 1 mW, logo para se calcular a potência nesta unidade utiliza-se a seguinte fórmula:

$$P[dBm] = 10\log\left(\frac{P[mW]}{1mW}\right) \tag{3.2}$$

Ora se o receptor de um nó consegue receber um sinal com -100 dBm, quer dizer este é capaz de receber sinais tão fracos como de 0,1 pW (pico watt). Esta sensibilidade é bem necessária no lado do receptor, pois uma onda RF com a sua dispersão e atenuação que sofrem ao decorrer a sua propagação, faz com que a potência do sinal na sua recepção seja sempre muito inferior à da utilizada na sua transmissão. Para descrever a propagação das ondas RF são utilizados modelos matemáticos baseados nas equações de Maxwell [77, 78]. Sendo um modelo simples o apresentado pela fórmula de Friis, que modela matematicamente a propagação de ondas eletromagnéticas no espaço livre [79].

$$\frac{P_t}{P_r} = \left(\frac{\lambda}{4\pi d}\right)^2 G_t G_r = Lfs G_t G_r \tag{3.3}$$

em que:

- P_t é a potência transmitida;

- P_r é a potência recebida;
- G_t ganho da antena do transmissor;
- G_r ganho da antena do receptor;
- λ é o comprimento de onda;
- d a distância entre o transmissor e o receptor;
- Lfs é a "perda de transmissão em espaço livre".

Quando aplicada uma antena omnidirecional num transmissor, teoricamente este transmite ondas de rádio de igual forma em todas as direções, mas uma antena real é não uniforme, e é descrita pelo seu ganho direcional. Pela aproximação de Friis percebe-se que no espaço livre, um receptor colocado a uma distância d do transmissor, receberá uma potência da onda RF que é o inverso do quadrado da distância [80], tal como demonstrado na Figura 3.9.

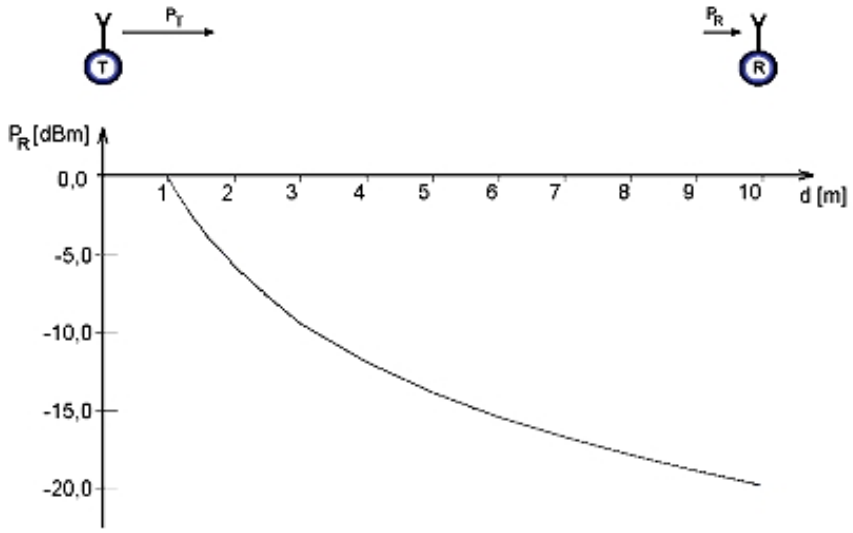

Figura 3.9: RSSI em função da distância entre transmissor (Tx) e o receptor (Rx) [9].

A aproximação de Friis pode ser matematicamente descrita em decibéis por:

$$P_r - P_t = -20\log(d) + 20\log(d_0)[dB] \tag{3.4}$$

Em que a parcela $20\log(d_0)$ já incluí o peso de G_t, G_r e $\left(\frac{\lambda}{4\pi}\right)$.

Este modelo é um modelo irrealista, pois existe sempre obstáculos entre o transmissor e receptor, o que faz com que o sinal sofra atenuação por causa da absorção eletromagnética desses obstáculos [68]. Alguma literatura explica o uso da fórmula de Friis para criar um modelo da propagação das ondas RF em ambientes com obstáculos, para isso utilizam uma modificação no expoente n da variável d. Sendo os valores típicos para n [80, 81]:

- Espaço livre 2
- Área urbana celular 2,7 a 3,5
- Célula sombreada urbana 3 a 5
- Dentro de edifícios (“LOS”) 1,6 a 1,8
- Obstruído em edifícios 4 a 6
- Obstruído em fabricas 2 a 3

O transceptor utilizado pelas motes Tmote sky é o CC2420 [82], construído num único circuito integrado, utilizando a tecnologia CMOS, e projetado para operar na faixa de 2,4 GHz cumprindo o standard IEEE 802.15.4, podendo a potência do transmissor variar entre -25dBm e 0dBm de forma programável, e a sensibilidade do receptor ir até aos -94 dBm, utilizando a modulação OQPSK (offset quadrature phase shift keying), assim como a codificação DSSS (direct-sequence spread spectrum). A modulação referida permite uma maior eficiência na transmissão, podendo esta atingir uma taxa de transferência de 250 Kbps. A utilização da codificação DSSS atribuí ao sinal de rádio uma maior robustez, evitando assim interferências e efeitos de propagação por caminhos múltiplos. Disponibiliza o RSSI no formato digital em complemento para 2 de 8 bits, com uma gama máxima de $\pm$ 6dB. Onde a intensidade do sinal calculada da seguinte forma [82]:

$$P_r = RSSI\_DIG + RSSI\_OFFSET[dBm] \tag{3.5}$$

sendo *RSSI_DIG* o valor digital na forma de complemento para 2 e *RSSI_OFFSET* o valor de -45dBm.

3.8 Resumo

Neste capítulo é apresentado o problema que se propõe solucionar de uma forma detalhada, e a importância dessa questão.

Seguidamente é apresentada a metodologia que se irá seguir, pois pretende-se seguir desde início boas normas de trabalho, de forma a que se consiga obter um sistema que cumpra todos os requisitos.

Para contextualizar, é apresentada a metodologia que é utilizada nas operações com os UAVs, tentando assim perceber o que é necessário e as restrições impostas pelos vários motivos inerentes a este tipo de operação.

Também é neste capítulo que se apresentam os requisitos para o sistema a desenvolver e, a partir dos quais, se criam as métricas apresentadas na secção seguinte.

De seguida é apresentado o *software* que irá ser necessário para a realização deste sistema, como também para fazer integrante do mesmo.

Finalmente é analisado possíveis abordagens para vários componentes e/ou técnicas e dessa forma realizarmos uma escolha ponderada.

Capítulo 4

Implementação do sistema

Antes da explicação de cada etapa realizada para a implementação do sistema, temos a arquitectura do sistema implementado. Podemos dividi-lo em três sub-sistemas:

- **RSSF -** rede disseminada responsável, pela recolha dos dados pretendidos, assim como a sua transmissão, e que contém a infra-estrutura necessária para o sistema de caraterização da mesma;
- **UAV -** veículo aéreo não tripulado com capacidade de comunicar com a RSSF disseminada, com o recurso a mote no seu interior, assim como de comunicar com a *Ground Station*, para o encaminhamento dos dados recolhidos pela RSSF;
- **Ground Station -** estação de operação em terra capaz de receber a informação retransmitida pelo UAV, assim como realizar o tratamento dos dados recolhidos e da sua apresentação para o utilizador do sistema, sendo este sub-sistema a interligação entre o utilizador e o sistema desenvolvido.

Na Figura 4.1 é possível ver a arquitectura funcional do sistema desenvolvido.

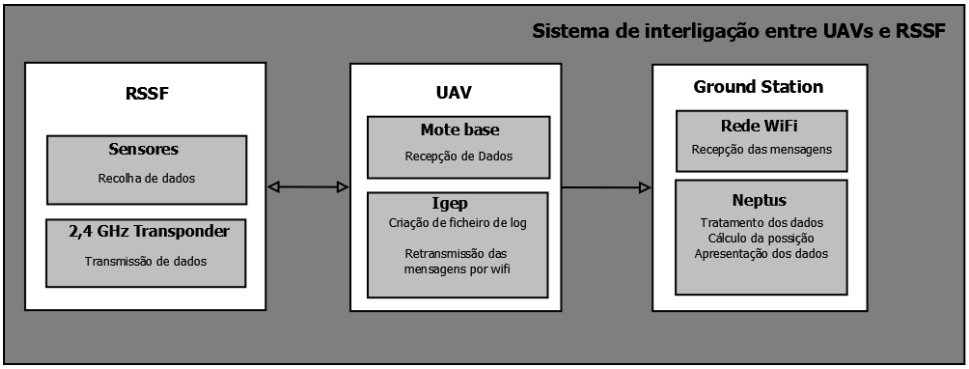

Figura 4.1: Arquitectura Funcional.

4.1 RSSF - Rede de Sensores Sem Fios

Inicialmente, para se perceber o funcionamento dos nós bem como a sua programação, foram realizados vários exercícios do tutorial da *wiki* do tinyOS [83]. Neste processo inicial, muitos conceitos necessários para a programação das motes foram assimilados, em que o conceito da criação de um módulo de configuração, para se realizar o "*bind*"das interfaces com quais se pretende trabalhar, era algo a qual não se estava habituado, tendo criado alguns entraves no desenvolvimento inicial de módulos para programar as motes.

Para se conseguir os objetivos de uma forma eficaz, tendo em conta o baixo conhecimento inicial neste tipo de tecnologia, optou-se por dividir as motes em três categorias de acordo com a sua função desejada para a rede, de forma a que esta permitisse o pretendido para o sistema. Essas três categorias são as seguintes:

- *Base*: Este nó é o que recolhe os dados das outras motes, ou seja é o "*root*"da rede, e o que realiza a interface entre a RSSF e a Igep. Para que este nó tenha estas competências, foi programado com o módulo "Base". Uma vez que este nó vai dentro do UAV, a estimativa da sua localização também é realizada;

- *Âncoras*: Como verificado na secção 2.11 existem algoritmos que precisam de uma infra-estrutura prévia para o seu funcionamento, sendo estes nós os que criam essa infra-estrutura para o correto funcionamento do algoritmo escolhido para implementar, ou seja o posicionamento destes nós é conhecido à priori. Estes nós têm a capacidade de medir o RSSI da mensagem recebida e reencaminhar para o nó base. Para estes nós obterem este funcionamento foram programados com o módulo "Intercepter";

- *Cegos*: Estes nós são os responsáveis por enviar uma mensagem periódica com os dados pretendidos para a RSSF, e é também a estes nós que se quer estimar a sua localização. Para terem este funcionamento foram programados com o módulo "BlindNodeCTP".

4.2 Igep

As implementações na Igep, foram demoradas, devido à sua arquitectura ARM do seu processador, o que impede a compilação usual de programas, requerendo um *cross-compile* para a realização de uma compilação funcional neste tipo de arquitectura.

Outra dificuldade encontrada foi a falta de ferramentas de base da distribuição instalada de fabrica (Poky, secção 3.6.1), o que levou à tentativa de mudança de distribuição, a qual só não chegou a ser realizada, devido aos links dos ficheiros de "bootloader"estarem indisponíveis, o que levou a procurar uma solução com a distribuição disponível.

Como a maioria das ferramentas disponibilizadas pelo tinyOS para funcionar com os nós de sensores estão disponíveis em java, optou-se por arranjar uma maquina virtual java para instalar na Igep, permitindo assim a utilização dessas mesmas ferramentas, ou mesmo a criação na mesma

linguagem. Após muita pesquisa e tentativas, foi encontrada uma máquina virtual de java para sistemas embebidos com arquitectura ARM, disponibilizada pela ORACLE [84].

Isto possibilitou a transferência das ferramentas do tinyOS para Igep, mas para o seu funcionamento foi necessário realizar o *cross-compile* dos ficheiros "libgetenv.so"e "libtoscomm.so", sem os quais não se conseguia utilizar as ferramentas. Assim já foi possível receber as mensagens que a mote base recebia e transmitia para a porta USB, e partir da qual eram reencaminhadas com o recurso da ferramenta *"SerialForwarder"*, que possibilita o encaminhamento das mensagens recebidas para uma *socket* sem se preocupar com o seu conteúdo.

Com esta funcionalidade obtida, era necessário um *software* que lesse as mensagens e as guardasse num ficheiro de "log", para este efeito utilizou-se o código fonte da ferramenta "Listen"e a partir do qual se criou uma ferramenta para este propósito, em que o seu funcionamento consiste em tornar-se cliente do "*SerialForwarder*"e dessa forma ler as mensagens da *socket* e escreve-las num ficheiro de log denominado de "log.txt".

4.3 Neptus

Depois de se perceber a arquitectura interna do Neptus, secção 3.6.3, é preciso perceber a arquitectura do *plugin* UAV, para o qual se desenvolveu um *panel* para servir de *interface* com o sistema desenvolvido.

4.3.1 Arquitectura do plugin UAV

O *plugin* UAV é subdividido em três componentes:

- *panels*: são os componentes base para módulo de operação. É nestes *panels* que são disponibilizadas as funcionalidades da ferramenta para o utilizador.
- *painters*: componentes responsáveis pelo desenho nas consolas, invocando o *StateRender2D* que se encontra na zona central do Neptus.
- *listeners*:componentes que colmatam a necessidade da sincronização entre um *panel* e os *painters* a si associados. Ou seja os *listners* são interfaces para os processos de atualização de informação. Quer isto dizer que cada *panel* que necessite possuí o seu respetivo *listner*.

4.3.2 Mote Panel para o UAV plugin

Foi então criado um *panel* para permitir que o utilizador possa interagir com o sistema desenvolvido. Este *panel* é composto por:

- um campo onde se introduz o IP da Igep, permitindo assim a conexão a diferentes Igeps, isto em momentos diferentes;
- um botão para escolha do ficheiro com as localizações das âncoras;

- um *toggle button* para se realizar a conexão ou desconexão com a Igep;
- dois *sliders* para se definir o tempo para três estados das motes em que, graficamente, se verificam estados por diferentes graus de transparência dos icons representativos das motes, ou seja, se a ultima comunicação de uma mote foi realizada a um tempo inferior ao minimo definido, esta não tem transparência, caso o tempo da ultima comunicação esteja entre os dois valores escolhidos, o icon representativo desta mote apresenta um grau pequeno de transparência, e o ultimo caso, em que o tempo da ultima comunicação é superior ao valor máximo escolhido, o icon apresenta o grau de transparência superior.

Sendo esta ferramenta desenvolvida em java, uma linguagem orientada a objectos e a qual não tinha sido utilizada anteriormente, apresentou um período de aprendizagem para sua aplicação.

Sendo o painel obtido o apresentado na Figura 4.2.

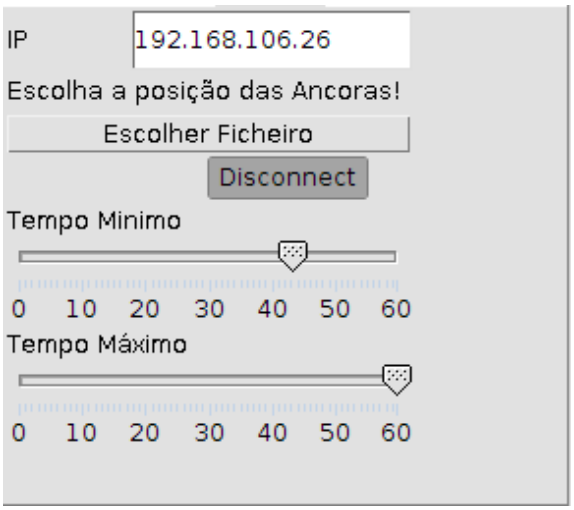

Figura 4.2: Imagem do *panel* criado.

4.3.3 Mote thread

Quando se pressiona o botão para conexão a Igep apresentado na secção 4.3.2, é iniciado uma *thread*. Sendo esta thread responsável pela conexão à *socket* da Igep na qual o *SerialForwarder* está a disponibilizar as mensagens recebidas e a partir das quais retira os dados, guarda no objecto Mote, os dados correspondentes ao id dessa mote. É também nesta *thread* que esta implementado o algoritmo de localização.

Sempre que uma nova mensagem é recebida, é verificado o id do nó que enviou a respetiva mensagem, retirando-se assim o id da mensagem, Sendo este id um numero serial da mensagem, compara-se com o ultimo guardado, caso exista perdas de pacotes enviados, guarda-se esse valor e é calculado uma taxa de perdas de pacotes para cada mote.

Para a implementação do algoritmo de localização foi necessário guardar a informação das ligações que cada mote têm com os seus vizinhos, para isso optou-se pela utilização de uma *Hash*

Table, que é uma boa opção para guardar informação, mas em que a sua implementação deveria ter sido diferente da realizada. Pois como foi implementada, é necessário remover a linha e volta-la a introduzir com os dados da Mote atualizados, isto porque a "chave"da *Hash Table* é o objecto Mote onde são armazenados os seus dados. Deveria-se ter usado uma "chave"só com os ids e a partir do qual se atualizava o respectivo objecto Mote com o mesmo id.

Este procedimento foi o escolhido devido ao fraco conhecimento destas estruturas (*Hash Table*), o que levou a várias dificuldades de implementação quer das funcionalidades quer dos algoritmos.

4.3.3.1 Mote location

Para realizar as localizações foi utilizado, como referido na secção 3.7.5, o método RSSI. Para calcular a posição dos nós de sensores foi escolhido o método *Bounding Box*, apresentado na secção 2.10.3, isto devido ao facto de requerer cálculos simples, de ser bastante percetível o seu funcionamento e ainda por possibilitar saber imediatamente a margem de erro para a posição estimada.

Implementados estes métodos com a conjugação da infra-estrutura criada previamente, apresentada na secção 4.1, a posição de cada nó de sensor é calculada, assim como a sua respectiva margem de erro, sendo estes dados guardados.

4.3.4 Mote painter

Para que toda esta informação esteja visível para o utilizador, foi necessário criar um *painter* para estes dados serem apresentados ao utilizador.

Sendo assim, o *painter* utiliza o *listner* associado ao *Mote panel* para ir atualizando os dados que apresenta.

Após ter sido carregado o ficheiro com a informação da posição das âncoras, este *painter* desenha-as nas localizações descritas no ficheiro escolhido. Após o algoritmo de localização ter estimado uma posição para um nó de sensores, é da responsabilidade deste módulo desenha-lo nessa posição, quando o utilizador clica com o rato em cima de uma mote, também é da responsabilidade deste módulo apresentar a informação dessa mote, sendo informação a seguinte:

- mote id;
- luz visível;
- luz não visível;
- humidade;
- temperatura;
- erro máximo da latitude;
- erro máximo da longitude;

- nº de pacotes perdidos em relação a mote selecionada;
- taxa de pacotes perdidos em relação a mote selecionada.

A Figura 4.3 exemplifica como o painter apresenta os dados para utilizador.

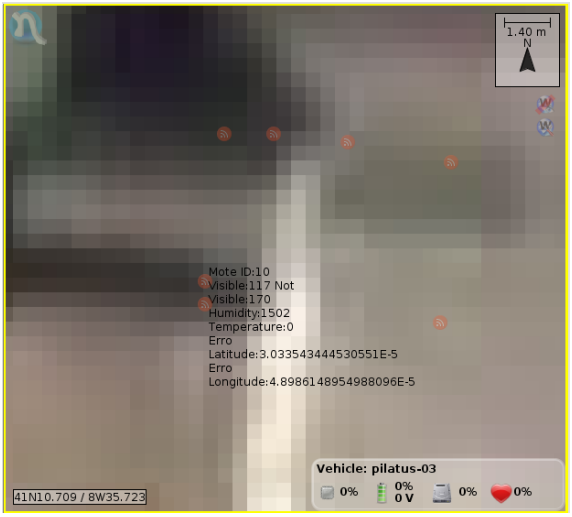

Figura 4.3: Exemplificação de como os dados são apresentados ao utilizador.

4.4 Resumo

Neste capítulo foram apresentadas as implementações realizadas para se obter o sistema pretendido, assim como a explicação de como foi implementado cada um dos componentes deste sistema, e também referenciadas as várias dificuldades encontradas na implementação do mesmo, bem como se tornou possível ultrapassar essas dificuldades.

Capítulo 5

Testes e Resultados

Após a descrição do sistema e da sua implementação, torna-se necessário apresentar os testes realizados e os seus resultados. Para verificar se o sistema funciona como previsto e se cumpre os requisitos previamente definidos para este projeto.

5.1 Teste da RSSF

Como a primeira parte que se implementou foi a RSSF, verificou-se se esta funcionava como previsto, com a transmissão das mensagens definidas e se eram recolhidas pelo nó definido como base. Nesta fase, a base estava ligada a um PC usual e utilizou-se a ferramenta "Listen"disponibilizada pelo tinyOS para se conseguir ler as mensagens recebidas por este nó. Desta forma foi possível verificar que as mensagens eram corretamente recebidas pelo nó base e que a rede dispunha da topologia pretendida.

Para utilizar esta ferramenta é necessário escrever na linha de comandos "*$: java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB0:telos*". Onde "*-comm serial@/dev/ttyUSB0*"serve para indicar que a mote está conectada na porta USB0 e é para fazer leitura dessa porta, já o "telos"é um *alias* para o *baudrate*, sendo o *baudrate* para as motes utilizadas de 115200, ou seja é possível escrever este comando com utilização dos "115200"no lugar de "telos".

Com a utilização desta ferramenta obteve-se os resultados apresentados na Figura 5.1.

Como verificado nessa figura, a forma como os dados são apresentados por esta ferramenta são pouco legíveis e percetíveis para se poder retirar qualquer tipo de conclusão, a não ser que as mensagens estão a ser transmitidas, propagadas pela rede e recebidas pela mote "base".

Então, para se perceber melhor o que se está a receber, recorreu-se a outra ferramenta disponibilizada pelo tinyOS, o "*MsgReader*", em que, para se utilizar esta ferramenta é necessário criar uma classe java com o *parsing* da estrutura da mensagem em utilização para esta ferramenta poder "perceber"as mensagens recebidas.

Na Figura 5.2 é apresentado os resultados obtidos com a utilização desta ferramenta.

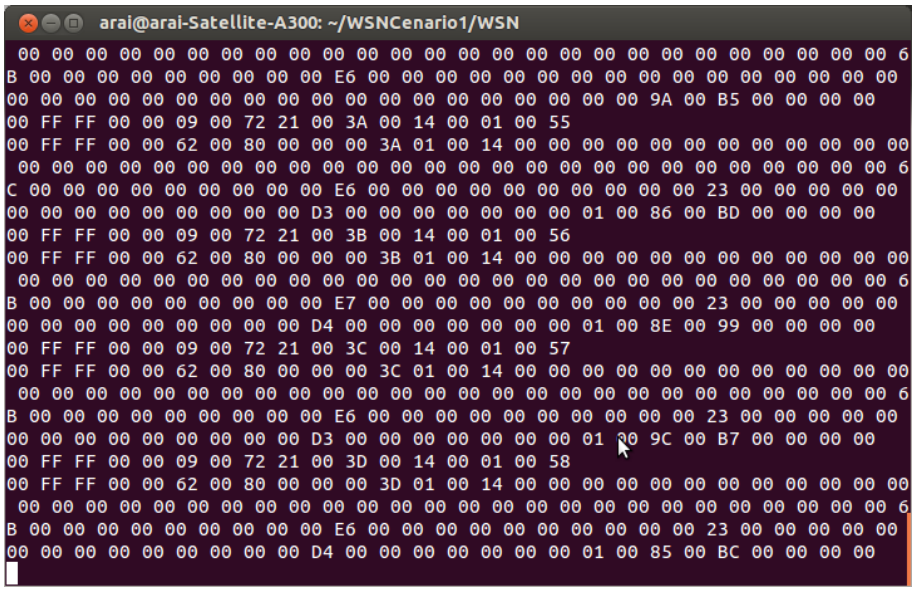

Figura 5.1: Verificação que as mensagens eram recebidas pela mote "Base".

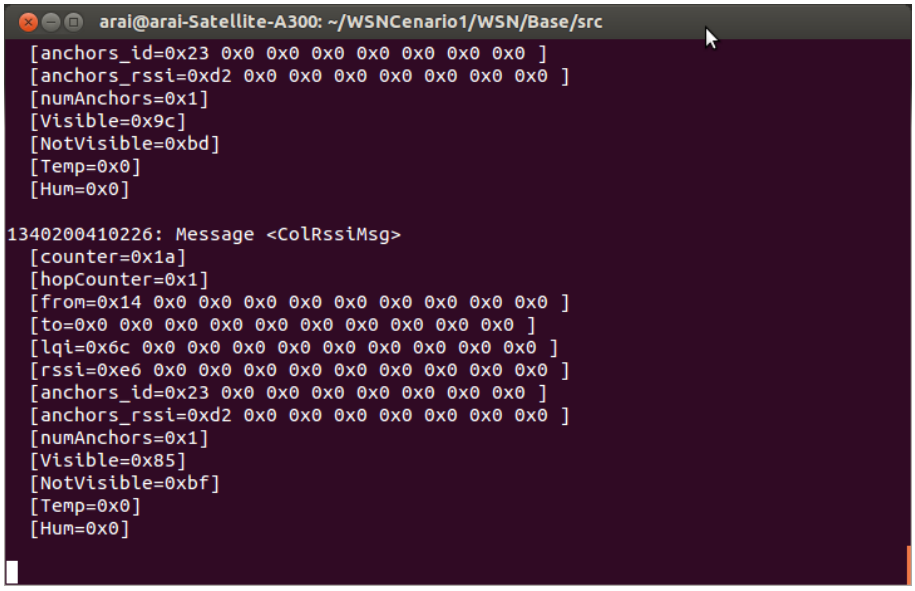

Figura 5.2: Resultados obtidos com a ferramenta "*MsgReader*"com a mote "base"ligada a um PC.

5.2 Teste da configuração da IGEP

Após a configuração da IGEP, era necessário verificar se essa configuração era valida, se os módulos e *software* instalado eram suficientes para o funcionamento pretendido para o sistema e, como algum deste software foi desenvolvido especificamente para este projeto, verificar se funcionava corretamente e que não continha *bugs* que impedissem o seu funcionamento pretendido, ou causassem falhas críticas de sistema.

Para isso, inicialmente recorrendo às mesmas ferramentas utilizadas na seção anterior (5.1),

verificamos se a leitura do nó base funcionaria da mesma forma, pois estamos perante uma arquitectura de processador diferente. Isto foi conseguido através de um túnel SSH para com a IGEP. E após tal ter sido verificado, como se pode verificar na Figura 5.3.

Figura 5.3: Verificação que as mensagens eram recebidas pela mote "Base"quando esta ligada a Igep.

De uma forma análoga aos testes feitos quando a mote "base"estava ligada num PC (5.1), transferiu-se a classe que permite o *parsing* da mensagem para a Igep, e assim voltou-se a utilizar o "*MsgReader*"para ser possível uma maior perceção do conteúdo das mensagens recebidas, tal como apresentado na Figura 5.4.

Foi realizado o teste à ferramenta que realiza o reencaminhamento da informação da porta série, neste caso USB, para um *socket* da própria IGEP. Para cumprir esta funcionalidade, utiliza-se o *SerialForwarder* disponibilizado pelo tinyOS, para verificar se o seu funcionamento era o suposto, recorreu-se de novo as ferramentas *Listen* e *MsgReader* e como demonstrado nas Figuras 5.5,5.6 , onde se pode que o funcionamento pretendido foi verificado.

Com estes requisitos funcionais validados, era necessário então verificar se a IGEP sempre criava o ficheiro de *log* pretendido. Esta funcionalidade é realizada por um software criado para esse propósito, e para verificar se este software fazia o pretendido, além da obvia verificação do conteúdo do ficheiro no fim de cada utilização do sistema, utilizou-se o comando de *shell* do linux *tail* para verificar o que é que era escrito em cada instante no ficheiro, e se correspondia com que estava a ser demonstrado pelo *listen*, este teste pode ser visto na Figura 5.7.

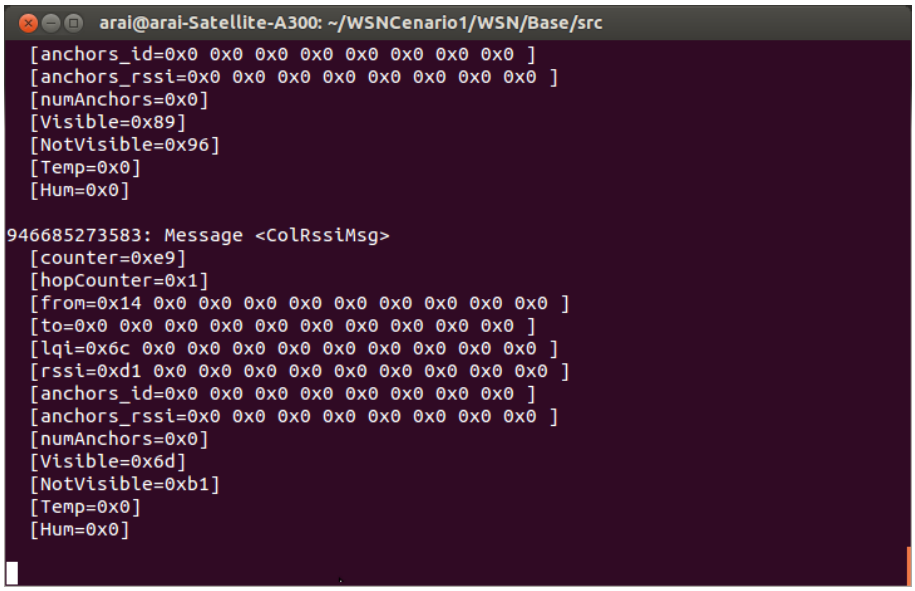

Figura 5.4: Resultados obtidos com a ferramenta "*MsgReader*"com a mote "base"ligada a Igep.

5.3 Teste da consola Neptus

5.3.1 Teste da interligação do Neptus ao resto do sistema

Antes do desenvolvimento do painel para agregar a interface gráfica e de implementar o algoritmo, foi evidente que era necessário verificar se as mensagens chegavam ao Neptus e que este sabia realizar o respetivo *parsing*. Para isso realizou-se num ciclo uma série de Print Outs na consola, com vários campos da mensagem, tal como a apresentado na Figura 5.8.

5.3.2 Teste do painel

Primeiro realizou-se um teste só com o painel criado, tendo sido criado um *"main"* para esse efeito, onde foi verificado se as funcionalidades criadas se encontravam disponíveis e funcionais. Verificou-se então se a introdução do IP (Internet Protocol) era realizada com sucesso, após isso se a conexão e a desconexão à IGEP era realizada com o pressionar do *Toogle Button*. Com um contador na *thread*, verificou-se de seguida se, ao pressionar o botão para desconectar a *thread* era interrompida, e se não teríamos várias threads iguais a correr ao mesmo tempo. Tudo isto foi verificado com simples Outputs nas respetivas consolas.

5.4 Teste do algoritmo Localização

Para testar o algoritmo de localização foram realizados dois testes, um primeiro em ambiente laboratorial, em que se conseguiu verificar o comportamento do algoritmo. Para este teste foram utilizados quatro nós como âncoras e dois como nós cegos e um como a base.

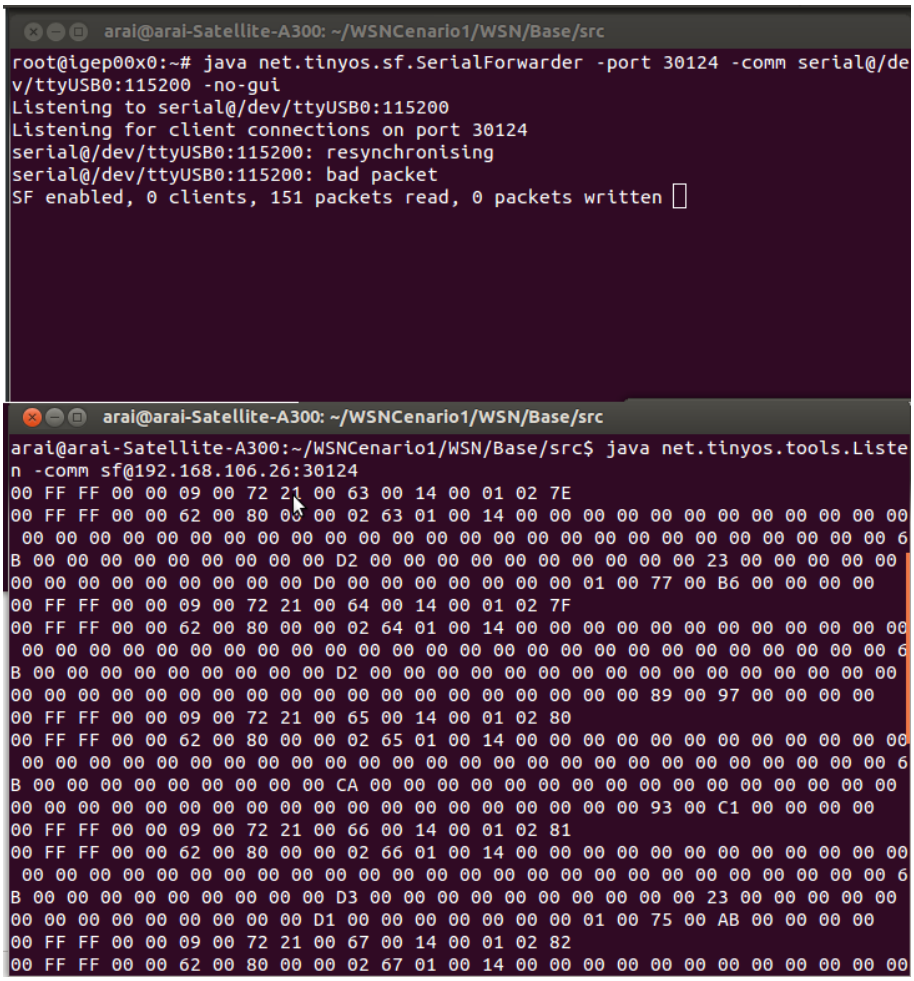

Figura 5.5: *Screen shot* com a utilização do "*SerialForwarder*"na Igep e o Listen no PC.

Neste teste colocou-se uma âncora aproximadamente em cada canto do laboratorio, e espalhou-se os dois nós mais a base, estando esta conectada com a Igep. O resultado obtido é o apresentado na Figura 5.9.

Como as distâncias entre nós durante este teste foram curtas, menos de 8 metros, não existiu qualquer tipo de perdas de pacotes.

Durante este teste para verificar se o sistema mantinha coerência em relação às medições de RSSI, durante 75 minutos, e com um período de 5 minutos, foi retirado o erro de latitude e de longitude para cada posição estimada.

Esses resultados, devido à sua dimensão, foram divididos em duas tabelas: a Tabela 5.1 e Tabela 5.2.

Com estes dados foram elaborados vários gráficos para que se percebesse melhor o significado destes dados obtidos.

Primeiro foi feito um gráfico para cada mote com os seus erros tal como apresentados nas tabelas, mais o erro euclidiano (ver Figuras 5.10,5.11,5.12).

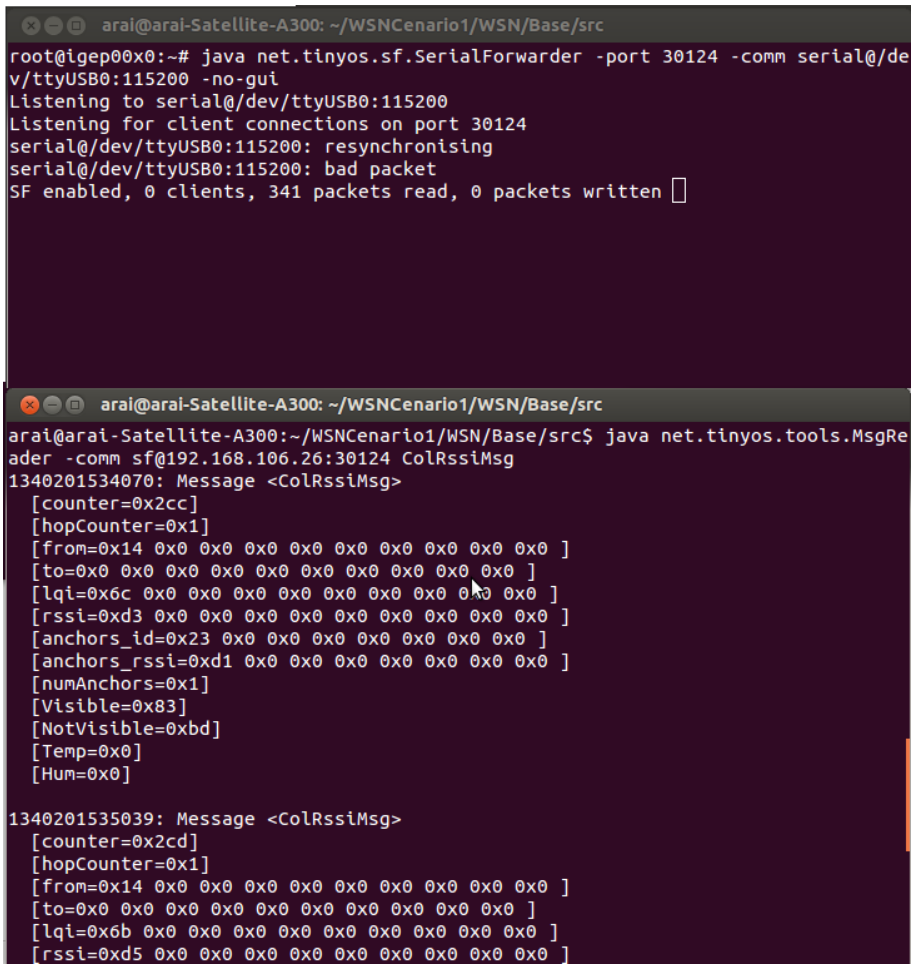

Figura 5.6: *Screen shot* com a utilização do "*MsgReader*"na Igep e o Listen no PC.

Tabela 5.1: Dados obtidos no teste efectuado no laboratório (1ª parte)

ID	Tempo (min)	5	10	15	20	25	30	35	40
10	Erro Latitude (m)	2,54	2,30	2,26	1,10	0,56	0,88	1,55	1,57
	Erro Longitude (m)	1,48	1,14	1,05	1,01	1,01	0,84	1,19	1,12
20	Erro Latitude (m)	1,38	1,15	1,12	1,24	1,15	1,26	1,25	1,15
	Erro Longitude (m)	0,74	0,59	0,58	0,62	0,62	0,63	0,69	0,63
0	Erro Latitude (m)	0,72	0,70	0,66	1,06	0,71	0,83	1,07	0,67
	Erro longitude (m)	0,93	1,10	0,98	0,62	0,96	0,86	0,60	1,00

Como as estimativas obtidas podem entrar no calculo das outras estimativas, caso essa mote seja vizinha desta nova mote da qual se quer estimar a sua posição, foi elaborado um gráfico com erros euclidianos das estimativas obtidas, para ser mais fácil a perceção da existência de alguma correlação entre os erros e, consequentemente, suas estimativas. Ver a Figura 5.13 para ver esta comparação entre os erros obtidos.

Ao analisarmos estes resultados, verificamos que o sistema tem uma boa fiabilidade e que, apesar da dependência que pode existir entre estimativas, este sistema mostra uma boa concordância

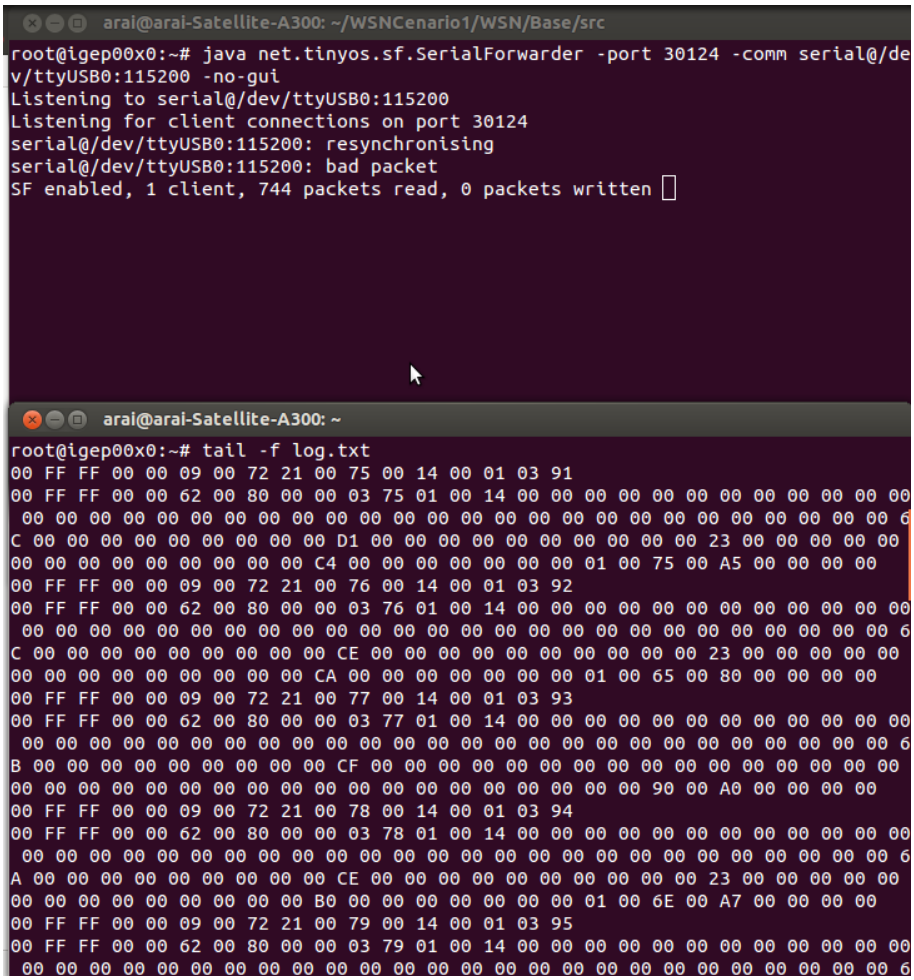

Figura 5.7: *Screen shot* do "*tail*"ao ficheiro "Log.txt".

Tabela 5.2: Dados obtidos no teste efectuado no laboratório (2ª parte)

ID	Tempo (min)	45	50	55	60	65	70	75
10	Erro Latitude (m)	2,20	2,19	2,43	3,47	2,40	2,91	1,93
	Erro Longitude (m)	1,03	1,13	1,94	2,27	1,22	1,73	0,95
20	Erro Latitude (m)	1,24	1,23	1,55	1,23	1,52	1,88	1,43
	Erro Longitude (m)	0,66	0,65	0,80	0,66	0,79	0,96	0,73
0	Erro Latitude (m)	0,79	0,71	0,75	0,76	0,88	0,82	0,92
	Erro longitude (m)	0,92	1,10	0,83	0,88	0,71	0,48	0,60

temporal das medições de RSSI.

Após estes resultados obtidos em ambiente laboratorial, fez-se um teste no parque de estacionamento dos alunos da FEUP. Foram selecionadas quatro localizações para se posicionar as âncoras, evitando-se a colinearidade destas posições, pois, pela literatura, quanto menos colinearidade, melhor será o funcionamento do sistema de localização, mas devido à disposição do parque, não foi possível evitar totalmente a colinearidade entre as posições. As posições escolhidas, assim

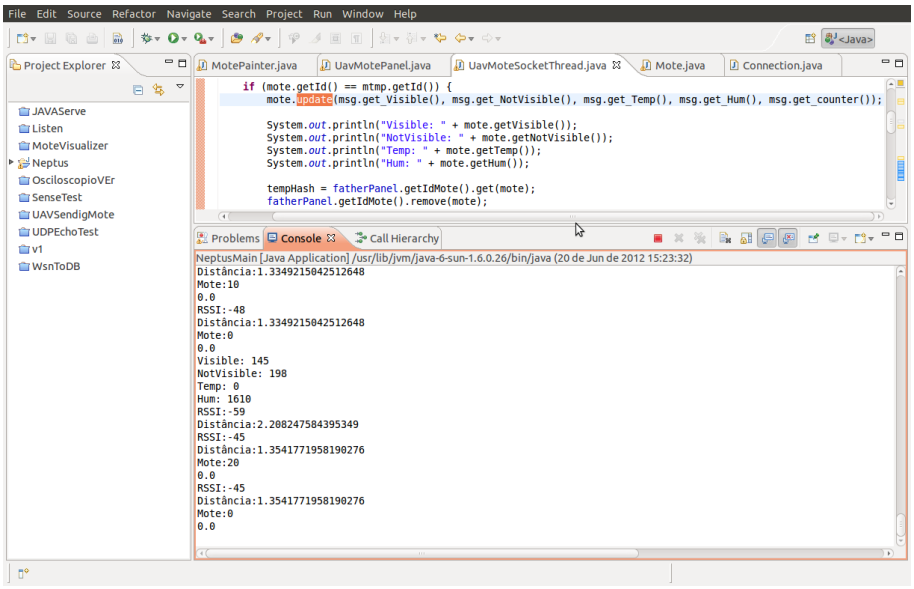

Figura 5.8: *Screen shot* do "Eclipse"quando a comunicar com a Igep.

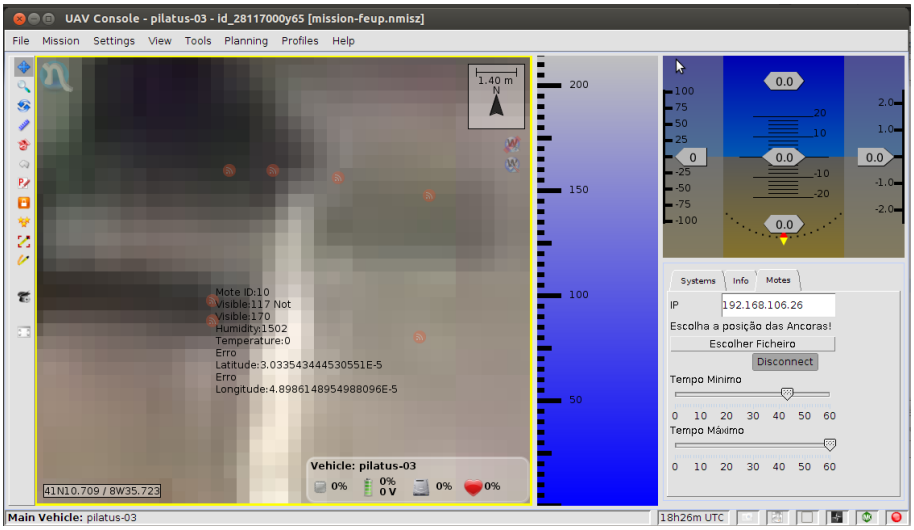

Figura 5.9: *Screen shot* da consola com o resultado obtido.

como as suas distâncias estão apresentadas na Figura 5.14.

Neste teste, devido às varias imprecisões das coordenadas para os nós de referência e às interferências provocadas por vários dispositivos emissores de sinal, em que uma boa parte deles a utilizarem a frequência 2,4 GHz, que é a mesma utilizada pelos nós de sensores, a quantidade de obstáculos (carros, arvores, etc), o facto do hardware utilizado já não ter o rendimento previsto, pois como dito no Capítulo 3, este foi utilizado por já existir disponível no LSTS, não se conseguiu obter comunicações suficientes entre as motes e entre elas e a base, para que o sistema de localização pudesse funcionar correctamente.

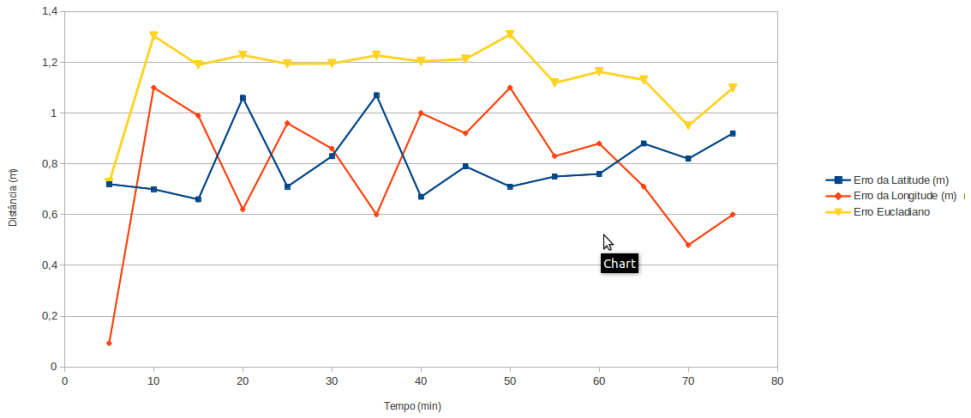

Figura 5.10: Gráfico com os erros obtidos para a estimativa da posição da mote com ID 0.

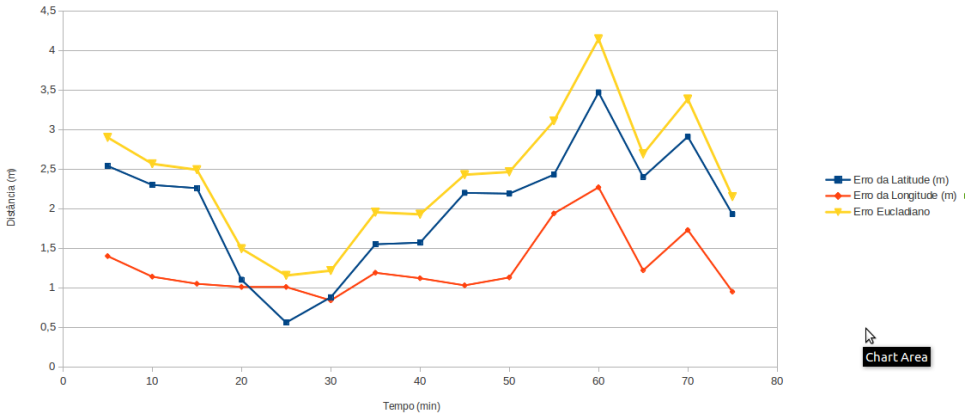

Figura 5.11: Gráfico com os erros obtidos para a estimativa da posição da mote com ID 10.

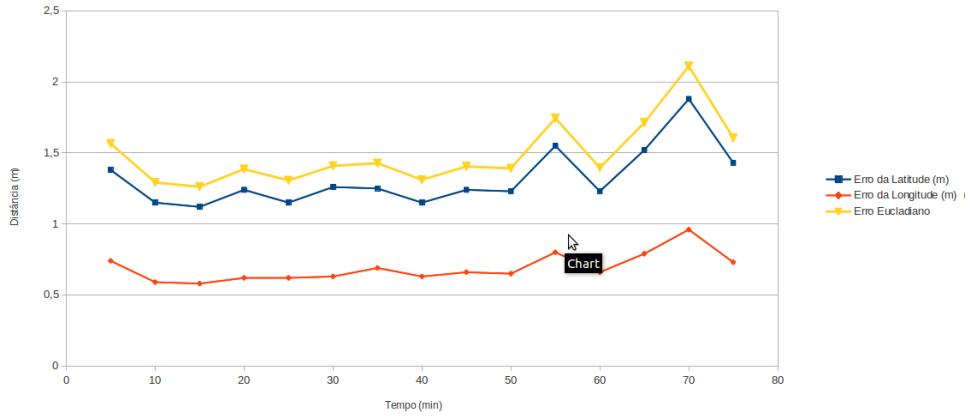

Figura 5.12: Gráfico com os erros obtidos para a estimativa da posição da mote com ID 20.

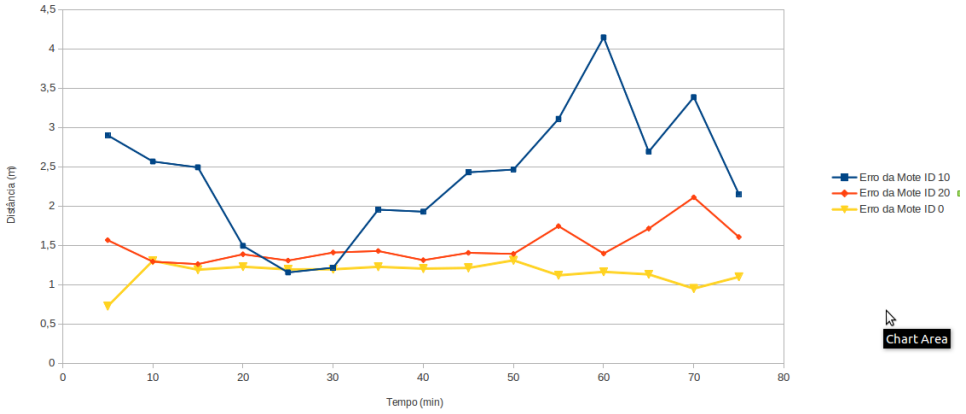

Figura 5.13: Gráfico com os erros euclidianos obtidos para as estimativas das posições das motes.

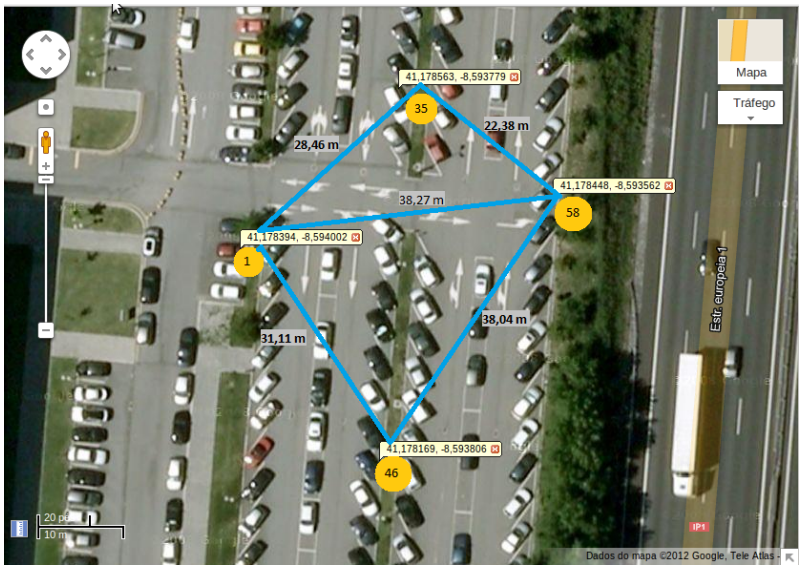

Figura 5.14: Imagem da loclaização das âncoras no teste no Parque 3 da FEUP.

Quer isto dizer que apesar do sistema ser funcional, tal como comprovado no teste Laborotarial, para este ter um bom funcionamento no exterior, necessita de uma revisão nós de sensores, ou então de um rádio diferente para comunicação.

É de referir que houve também uma tentativa de teste no Aeroclube da Costa Verde, situado em Espinho. Mas aquando deste teste, o sistema ainda não estava totalmente desenvolvido, o que tornou impossível a obtenção de qualquer resultado para validação do sistema.

5.5 Resumo

Neste Capítulo foram apresentados os testes realizados, assim como os resultados obtidos. Verificando a capacidade do sistema, e que este cumpre os requisitos definidos no Capítulo 3.

No próximo capítulo(Capítulo 6) são apresentadas as conclusões do trabalho realizado no âmbito desta dissertação.

Capítulo 6

Conclusões e Trabalho Futuro

Sendo este o capítulo final, é neste capítulo que será apresentado as conclusões, onde será dito qual a situação final deste projeto, e finalmente descrito uns tópicos para trabalho futuro.

6.1 Resumo do trabalho realizado

O objectivo fundamental deste projecto é o de possibilitar aos UAVs competências, para este comunicarem com uma RSSF. Neste trabalho a RSSF era uma rede de Tmotes Sky, mas serve para demonstrar a possibilidade destes sistema aéreos poderem interagir com outros sistemas num teatro de operações. Para se atingir este objectivo realizou-se os seguintes passos:

- Foi realizada então pesquisa inicial sobre sistemas com veículos aéreos não tripulados, para perceber o que já existe implementado neste tipo de sistemas. Juntamente a esta pesquisa teve-se acesso a informação em que ponto estavam este tipo de sistemas do LSTS;

- Foi pesquisado a metodologia utilizada nas operações que utilizam veículos aéreos não tripulados, de forma a perceber que interações costumam ser necessárias entre este género de aeronaves e com os outros elementos presentes num teatro de operações;

- Na pesquisa realizada no ponto anterior, percebeu-se que a localização de certos elementos, é algo necessário, passando esta necessidade um objectivo para este projecto. Após a adisão deste objectivo, foi realizada a devida pesquisa sobre sistemas com esta funcionalidade, dando um enfase maior a sistemas deste tipo que utilizasem nós de sensores de sem fios;

- Seguidamente foi realizada a implentação, tanto do *hardware* como do *software* que se optou para este sistema;

- Finalmente foram realizados vários testes, e com os resultados obtidos verificado o funcionamento do sistema desenvolvido.

6.2 Satisfação dos Objectivos

Uma vez que o software utilizado em na Igep é java, este pode ser também utilizado na PC-104. Uma vez que estas placas já são utilizadas pelo LSTS em veículos aéreos não tripulados, considera-se que este sistema permite atribuir a uma aeronave deste género a competência de interagir com outros elementos (motes) presentes no teatro de operações. De lembrar que devido as especificações das Tmote Sky utilizadas, o alcance máximo dito pelo fabricante é de 125m, o que torna a implentação deste sistema, tal como definido, não viável para sistemas de UAVs, pois estes usualmente operam a alturas do solo superios que este alcance máximo das motes. Além que mesmo para atingir estes valores teria de ser numa área aberta onde não existam grandes interferências, e de preferência com motes novas, pois as utilizadas no desenvolvimentos deste projecto já não se encontram nas melhores condições de uso, o que, como demonstrado no teste realizado no parque de estacionamento, as utilizadas já não conseguem num meio urbano atingir uma boa ligação com distâncias aproximadas de 30 metros.

O que demonstra que o sistema desenvolvido não é viável. Logo para contornar este novo problema encontrado é preciso encontrar outras hipóteses. Uma solução para esta questão seria desactivar o rádio da mote, e acoplar um radio com potência de transmissão superior, pois o rádio que a Tmote Sky utiliza, como descrito na secção 3.7.3, é de apenas 1 mW de potência, o que vai ao encontro do conceito base das motes, de baixo consumo energético. E uma vez que o rádio das aeronaves equipadas com Piccolo é de 1W de potência. Este facto immpossibilita este tipo de rádio nestas aeronaves, pois quando o rádio do Piccolo transmite "abafa"o rádio da Tmote Sky.

Como demonstrado no teste realizado no laboratório, este sistema consegue localizar os nós cegos e a base, assim como apresentar os dados adquiridos por cada nó de sensor, além das respectivas margens de erro, isto para os nós em que a sua posição foi estimada e também a taxa de pacotes perdidos por mote. Além que permite de uma forma intuitiva verificar o estado de uma mote. Mas de acordo com o teste realizado no exterior podemos verificar que as motes tem demasiada atenuação no seu sinal, devido a interferência dos obstáculos presentes no meio envolvente, o que leva ter que procurar outras alternativas para se superar esta dificuldade, existem já disponíveis no mercado motes que, utilizam comunicações por WiFi, algumas conseguem atingir 1,6km de raio de transmissão, como seria de esperar este género de motes tem consumos energéticos bem superiores.

6.3 Trabalho Futuro

O sistema obtido demonstra que o conceito inicial pretendido para este trabalho é plausível, ainda existe a realizar, muitas melhorias, e implementações de forma a que o sistema desenvolvido fique apto para testes em ambiente exterior ao laboratório, e que se torne mais robusto.

- Como verificado no teste realizado no exterior, é necessário arranjar uma solução para ultrapassar esta questão, podendo passar numa fase inicial, adaptação de antenas com mais

ganho, caso isto não seja suficiente a adaptação de amplificadores de sinal, ou mesmo adaptação de um rádio com potência superior de forma ao raio de alcance seja muito superior e claro não seja abafado pelas outras comunicações existentes no veículo aéreo não tripulado. Claro que isto traz custos tanto de hardware como energéticos, sendo sempre necessário realizar uma análise custo benefício.

- O método utilizado para estimação da posição utilizado não é muito preciso, com mais tempo é possível implementar métodos mais precisos e fiáveis, até mesmo algoritmos com calibração dinâmica.

- O algoritmo utilizado necessita de uma infra-estrutura prévia para o seu correcto funcionamento, poderá ser implementado outro algoritmo sem essa necessidade, sendo o mais atrativo para este tipo de sistema o LMB descrito na secção 2.11.7, pois a maioria dos veículos aéreos não tripulados têm um sistema que sabe a sua localização, ora conjugando essa informação para o algoritmo referido, pudesse realizar localização de nós de sensores com recurso apenas ao UAV em si.

- Estando o sistema apto para o exterior, este poderá ser utilizado para tracking de "alvos"de interesse, actualização em tempo real de vários elementos presentes num teatro de operações.

- Outra opção é a utilização da RSSF para comunicação entre veículos, tendo em conta que devido a latência existente neste tipo de rede, não permite que as mensagens enviados por este meio sejam de alta prioridade, mas por exemplo de redudância para robustez de um sistema mais amplo que incorpore estes sistemas.

- O painel criado permite ligação apenas a uma unidade de processamento, a adaptação do painel criado neste projeto para múltiplas conexões e assim conjugar com operações de Multi-UAVs, sendo neste caso até possível a adaptação do código dos nós de sensores para vários "*roots*".

Anexo A

Especificações das unidade de processamento central

Neste anexo é apresentado as especificações tanto da PC-104 como da Igep.

A.1 Especificações do PC-104

Tabela A.1: Especificações das características da PC/104 [5]

Descrição	**Características**
Processador	AMD® LX800 500MHZ
Memória	1 x DDR 200-pin DDR333/400 SO-DIMM support up to 1GB
Rede	10/100Mbps Realtek RTL8100C Ethernet chipset
Interfaces I/O	2 x USB 2.0 1 x LPT2 x RS-232 1 x RS-422/485 1 x PS/2 for KB/MS1 x IDE
Super I/O	W83627EHG
SSD	CF Type II
Unidade Gráfica	VGA integrated in LX800
Watchdog Timer	Software programable supports1-255 sec. system reset
Fonte de Alimentação	5V Only
Consumo energético	5V@1.13A (LX800-500MHz, DDR 400/1G RAM)
Dimensões	95 x 95 mm
Peso	GW: 500g / NW: 110g

A.2 Especificações da Igep

Tabela A.2: Especificações das características da Igep [6]

Descrição	**Características**
Processador	OMAP3530/ DM3730
Frequência do Processador	720MHz/ 1GHz
Memória SDRAM	512 MBytes LPDDR SDRAM - 200MHz
NAND FLASH	512 MBytes
DSP	TMS320DM-C64+
Frequência da DSP	500MHz/ 800MHz
Unidade Gráfica	PowerVR SGX 530 (100/200MHz)
Gestor de energia	TPS65950
Interfaces de *Debug*	consola RS232 + interface JTAG
Dimensões da PCB	93 x 65 x 1.6 mm

Anexo B

Plano de Testes

Neste anexo é apresentado o plano de testes elaborado para validação do sistema implementado.

B.1 RSSF

O primeiro teste é o teste à RSSF, verificando se o algoritmo de *collection* a implementar funcionará, e se os dados são correctamente disseminados. Para se testar isso tenciona-se usar as ferramentas disponibilizadas pelo tinyOs: "*Listen*"e o "*MsgReader*".

B.2 Igep

- Primeiro teste com a Igep é de conexão à mote "base"e a respetiva leitura dos dados recebidos por esta. Para se realizar esta verificação, utilizou-se de novo as ferramentas "*Listen*"e o "*MsgReader*", devido ao facto de ser um processador independente será utilizado um túnel SSH para permitir acesso à *Shell* do sistema operativo instalado na Igep.

- Segundo teste é o de verificação do correcto reencaminhamento das mensagens recebidas pela mote "base"para a rede. Como para verificação de mensagens transmitidas por motes, as ferramentas disponibilizadas pelo tinyOs são as mias simples e viáveis de se utilizar, recorrer-se-á as já utilizadas nos testes anteriores para a verificação pretendida.

- Terceiro teste é a verificação da correcta criação do ficheiro log, sendo o sistema operativo da Igep, uma distribuição de linux, a maneira mais simples de realizar esta verificação, é a utilização do comando "**tail**"para a visualização do irá ser escrito no ficheiro.

B.3 Neptus

- Primeiro teste é a verificação da comunicação entre a *framework* e a Igep. Mantendo os testes o mais simples, utilizar-se-á uns "*System.outs*"para permitir a leitura dos dados recebidos pelo Neptus.
- Segundo teste centrar-se-á no validação do painel. Onde se recorrer-á a utilização de "*System.outs*", de forma a ser possível verificar as funcionalidades instaladas no painel, através dos *outputs* gerados.
- Terceiro teste, incidirá sobre o funcionamento do algoritmo de localização, onde será feita a recolha da posição estimada e a real, assim como dos erros máximos obtidos, durante vários períodos de tempo.

B.4 Locais de teste

Todos estes testes serão realizados no laboratório. Após a implementação do sistema será realizado um teste no exterior, para se verificar o funcionamento do sistema a desenvolver neste tipo de meio, e se possível um teste no Aeródromo de Espinho, onde se pretende validar o sistema com a utilização de um UAV.

Anexo C

Tutorial de utilização do sistema

Neste anexo é apresentado um breve tutorial de como utilizar o sistema criado por este projecto. Após a disseminação da RSSF e do set-up todo montado. Existem duas fases de utilização, a primeira de arranque dos programas na Igep (Secção C.1) e uma segunda referente a utilização do painel criado para o Neptus.

Mas antes é preciso criar o ficheiro com a posição das âncoras, tal como apresenta na Figura C.1.

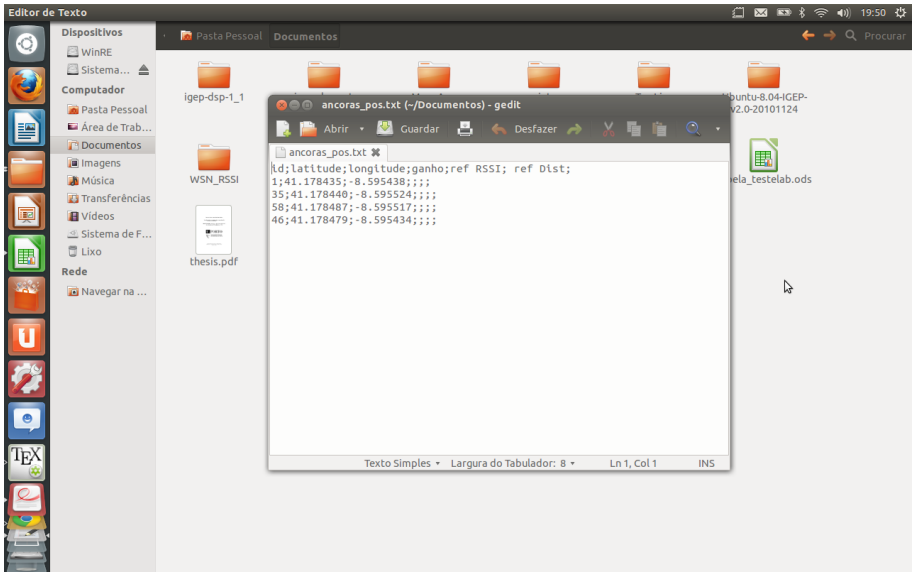

Figura C.1: Exemplo de um documento com a posição das âncoras.

C.1 Igep

O primeiro passo é a criação de un tunel SSH com a Igep, para se poder utiliza-la, devido ao script implementado no boot da Igep, esta fica configurada para a rede "PitVant Mission 2.4"com o IP "192.168.106.26". Sendo a password do utilizador "root"o predefinido (ver Figura C.2).

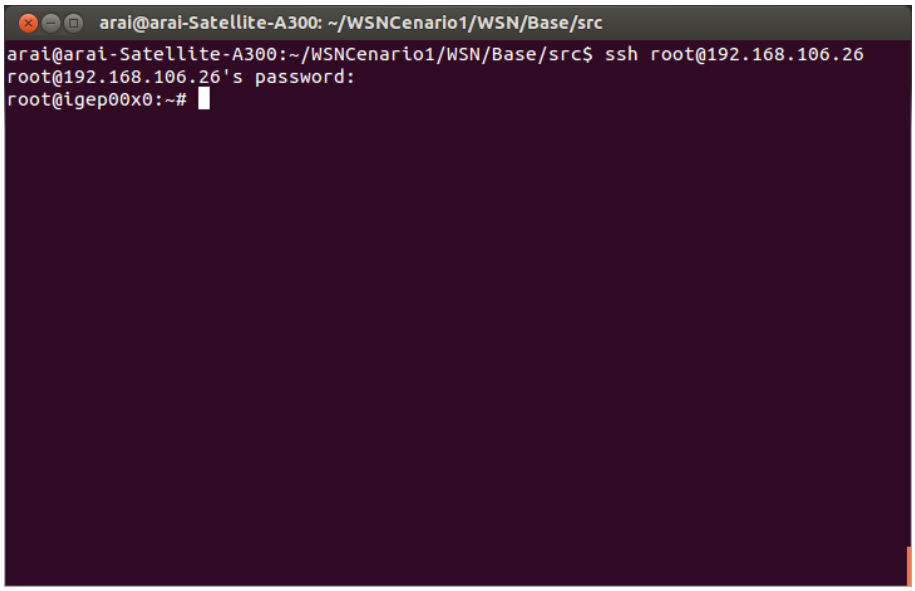

Figura C.2: Ligação a Igep.

O segundo passo devido a arquitectura ARM por estarmos a utilizar a Igep, é necessário correr umas instruções que estão dentro de um ficheiro para ser mais simple a configuração do java sempre que a Igep é iniciada, sendo esse ficheiro o "javaconfig.txt". Sendo então necessário introduzir o comando "source javaconfig.txt"(ver Figura C.3).

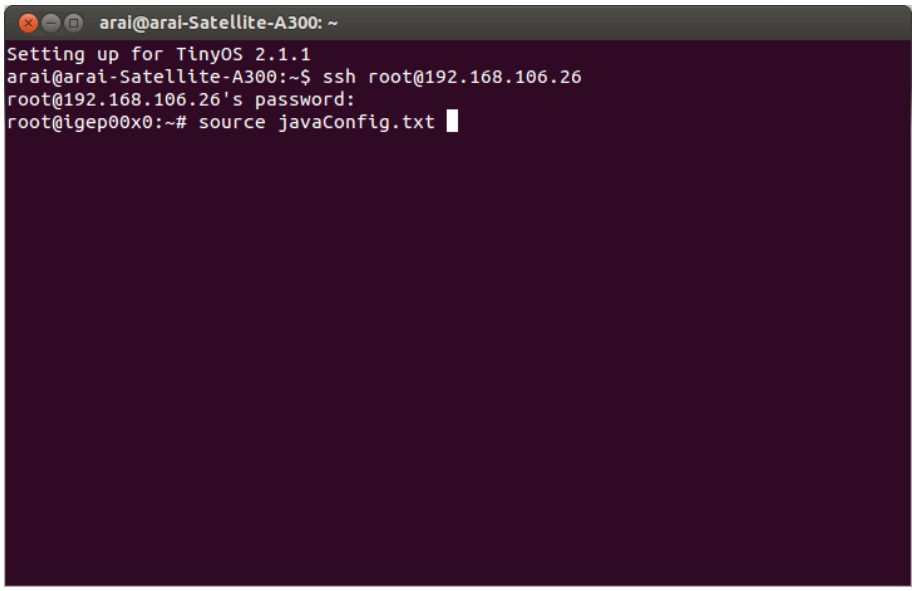

Figura C.3: Configuração do java na Igep.

o terceiro passo é correr o *SerialForwarder* do TinyOS, para isso é precisso introduzir o comando "java net.tinyos.sf.SerialForwarder -port 30124 -comm serial@/dev/ttyUSB0:11520 -nogui"como apresentado na Figura C.4, de referir que -port 30124"é para que as mensagens sejam

reencaminhadas para a porta 30124, que o -comm serial@/dev/ttyUSB0"é devido ser esta a ligaç ao utilizada pela mote base que esta ligada a Igep, o ":115200"é *baudrate* de uma mote Tmote Sky, e finalmente o -no-gui"por não se pretender uma interface gráfica para aplicação.

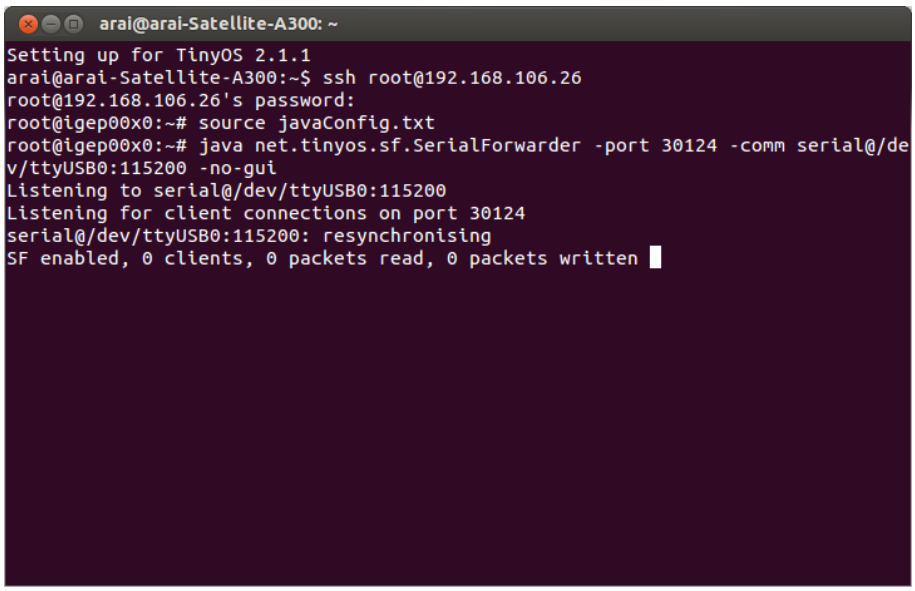

Figura C.4: Correr o *SerialForwarder*.

Os passos seguintes desta secção podem ser opcionais, mas são estes que permitem que seja realizado um ficheiro de log na Igep.

O quarto passo é repetir o procedimento do primeiro e segundo passo.

Após isso fazemos o quinto passo que é correr o program criado para se ligar a *socket* para leitura das mensagens e grava-las num ficheiro denominado "log.txt".

C.2 Neptus

Após a realização dos passo apresentados na secção anterior. Vamos referir os passo a seguir dentro da aplicação Neptus.

O primeiro passo é iniciar a aplicação Neptus, após o arranque da aplicação realizar a escolha da consola pretendida, se essa consola já tiver o painel das motes incorporado, passar para o passo seguinte, caso contrario adicionar esse painel a consola.

O segundo é introduzir no campo de IP, o IP atribuído a onde se pretende ligar para ler as mensagens, tal como aparece na Figura .

O terceiro passo é carregar o ficheiro com a posições das âncoras, para isso utilizar o botão "escolher ficheiro"que abre uma caixa onde se pode indicar o ficheiro para essa operação.

O quarto passo é carregar o botão "connect"para dar a instrução de ligação a Igep. A partir deste ponto temos o sistema a funcionar. Para o utilizador aceder aos dados de cada mote é só clicar com o rato em cima da mote pretendida e aperecerá as informações referentes a essa mote.

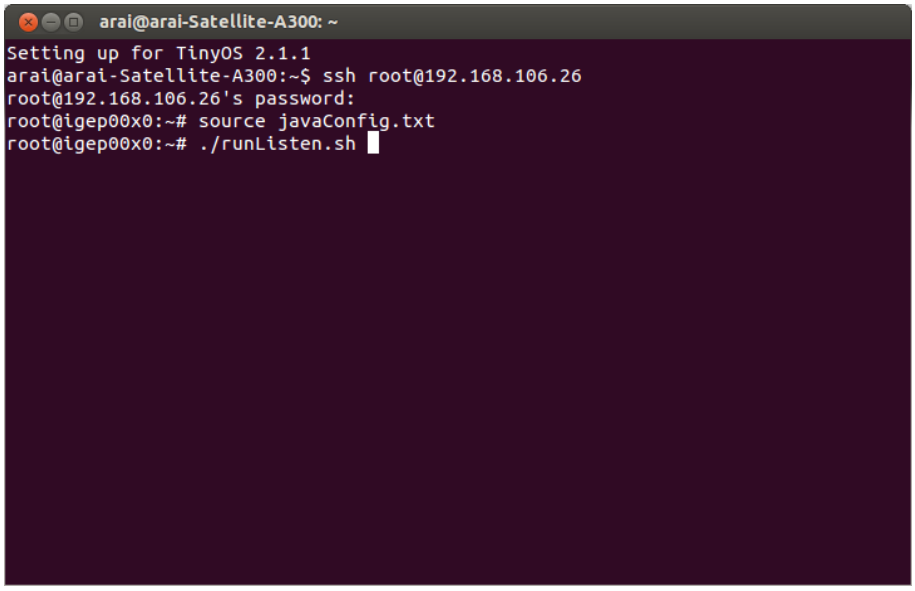

Figura C.5: Correr o programa para ser gravado um ficheiro de log na flash da Igep.

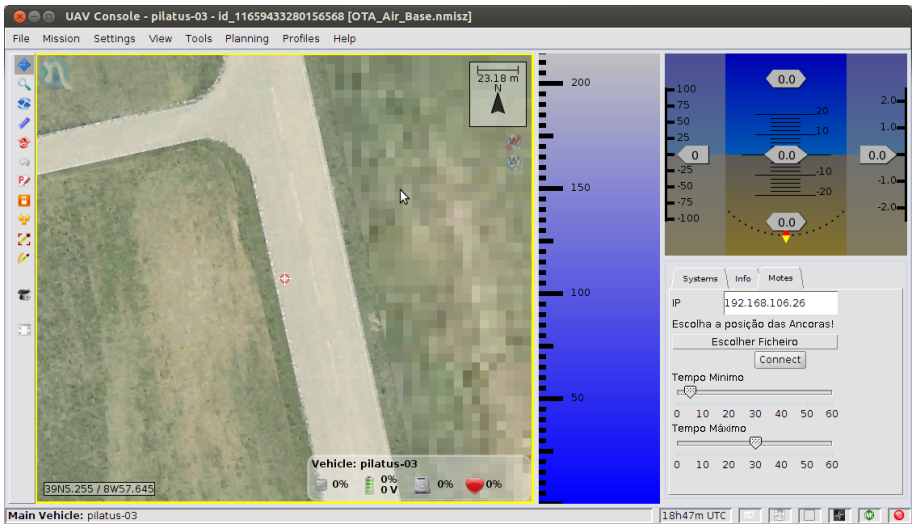

Figura C.6: Exemplo de uma consola com o painel mote.

Referências

[1] António Sérgio Ferreira. Interface de operação para veículos não tripulados. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2011.

[2] E. K. Jonathan How e Yoshiaki Kuwata. Flight demonstrations of cooperative control for uav teams, 2004.

[3] Cloud Cap Technologies. Piccolo flight managment system. `http://www.cloudcaptech.com/Sales%20and%20Marketing%20Documents/Piccolo%20Comparison%20Table.pdf`(visitado a 1 Junho de 2012).

[4] Jason Short J. M. Chris Anderson e Doug Weibel. Ardupilot 2.x manual. `http://code.google.com/p/ardupilot/wiki/ArduPilot?tm=6`,(visitado a 12 Fev 2012).

[5] The PC/104 Consortium. The pc/104. `http://www.pc104.org/`(visitado a 1 Junho de 2012).

[6] ISEE. Igep v2. `http://igep.es/products/processor-boards/igepv2-board`(visitado a 1 Junho de 2012).

[7] M. Armholt, S. Junnila, e I. Defee. A non-beaconing zigbee network implementation and performance study. Em *Communications, 2007. ICC '07. IEEE International Conference on*, páginas 3232 –3236, june 2007.

[8] R. Martins, P.S. Dias, E.R.B. Marques, J. Pinto, J.B. Sousa, e F.L. Pereira. Imc: A communication protocol for networked vehicles and sensors. Em *OCEANS 2009 - EUROPE*, páginas 1 –6, may 2009.

[9] Ricardo José dos Santos Reis Pinto da Silva. Localização e seguimento em redes de sensores sem fios. Tese de mestrado, Faculdade de Ciências da Universidade do Porto, 2010.

[10] Ieee standard for application and management of the systems engineering process. *IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998)*, páginas 1 –87, 2005.

[11] Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, páginas 0 –305, 2006.

[12] Joel Renato da Silva Cardoso. Low cost unmanned aerial vehicle for testing and validation of advanced control algorithms. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2011.

[13] Kimon P. (ed.) Valavanis. *Advances in unmanned aerial vehicles. State of the art and the road to autonomy.* Intelligent Systems, Control and Automation: Science and Engineering 33. Dordrecht: Springer. xxiv, 543 p. EUR 119.95/net; SFR 209.00;, 2007.

[14] M.W. Kadous, R.K.-M. Sheh, e C. Sammut. Controlling heterogeneous semi-autonomous rescue robot teams. Em *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 4, páginas 3204 –3209, oct. 2006.

[15] Holly A. Yanco, Jill L. Drury, e Jean Scholtz. Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition. *Human–Computer Interaction*, 19(1-2):117–149, 2004.

[16] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, e E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, páginas 393–422, 2002.

[17] Mohammad Ilyas e Imad Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press., 2004.

[18] D. Estrin, L. Girod, G. Pottie, e M. Srivastava. Instrumenting the world with wireless sensor networks. Em *n International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, páginas 2033–2036, 2001.

[19] Azzedine Boukerche, Horacio A. B. F. de Oliveira, Eduardo Freire Nakamura, e Antonio Alfredo Ferreira Loureiro. Towards an integrated solution for node localization and data routing in sensor networks. Em *ISCC*, páginas 449–454. IEEE, 2007.

[20] Satish Kumar e Deborah Estrin. SCalable object-tracking through unattended techniques (SCOUT). Relatório té, Satish Kumar (USC Information Sciences Institute); Deborah Estrin (UCLA and USC Information Sciences Institute);, 2000.

[21] Ya Xu, John Heidemann, e Deborah Estrin. Geography-informed energy conservation for ad hoc routing. Em *MobiCom 01: Proceedings of the 7th annual international conference on Mobile computing and networking*, páginas 70–84. ACM, 2001.

[22] M. Alighanbari, Y. Kuwata, e J.P. How. Coordination and control of multiple uavs with timing constraints and loitering. Em *American Control Conference, 2003. Proceedings of the 2003*, volume 6, páginas 5311 – 5316 vol.6, june 2003.

[23] *Authority Sharing in Mixed Initiative Control of Multiple Uninhabited Aerial Vehicles*, Orlando, Florida, USA, 07/2011 2011.

[24] E. J. Barth. A cooperative control structure for uav s executing a cooperative ground moving target engagement (cgmte) scenario. Em *American Control Conference, 2006*, páginas 2183–2190, 2006.

[25] M. Flint, M. Polycarpou, e E. Fernandez Gaucherand. Cooperative control for multiple autonomous uav's searching for targets. *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, 3:2823, 2002.

[26] Richard M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007.

[27] M. Flint, E. Fernandez Gaucherand, e M. Polycarpou. Cooperative control for uav's searching risky environments for targets. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 4:3567, 2003.

[28] J. Redding, A. Geramifard, A. Undurti, Choi Han-Lim, e J. P. How. An intelligent cooperative control architecture. Em *American Control Conference (ACC), 2010*, páginas 57–62, 2010.

[29] A. Geramifard, J. Redding, N. Roy, e J. P. How. Uav cooperative control with stochastic risk models. Em *American Control Conference (ACC), 2011*, páginas 3393–3398, 2011.

[30] Nian Xiaohong, Su Saijun, e Pan Huan. A bmi approach to the cooperative control of multiagent systems and optimization. Em *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, páginas 3656–3661, 2010.

[31] D. J. Musliner, E. H. Durfee, e K. G. Shin. Circa: a cooperative intelligent real-time control architecture. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(6):1561–1574, 1993.

[32] P. Almeida, G. M. Gonçalves, e J. B. Sousa. Multi-uav platform for integration in mixed-initiative coordinated missions. volume 1 de *1st IFAC Workshop on Multivehicle Systems, MVS 2006*, páginas 70–75, Salvador, BA, 2006.

[33] D. T. Cole, A. H. Goktogan, P. Thompson, e S. Sukkarieh. Mapping and tracking. *Robotics & Automation Magazine, IEEE*, 16(2):22–34, 2009.

[34] R. W. Beard, T. W. McLain, M. A. Goodrich, e E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *Robotics and Automation, IEEE Transactions on*, 18(6):911–922, 2002.

[35] Qu Yaohong e Zhang Youmin. Fault-tolerant localization for multi-uav cooperative flight. Em *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, páginas 131–136, 2010.

[36] Belgian Air Force. Iai - eagle b-hunter uav (unmanned aerial vehicle). `http://belmilac.wetpaint.com/page/IAI+-+Eagle+B-Hunter+UAV+%28Unmanned+Aerial+Vehicle%29`(visitado a 10 Junho de 2012).

[37] airforce technology.com. Predator rq-1 / mq-1 / mq-9 reaper uav, united states of america. `http://www.airforce-technology.com/projects/predator-uav/`(visitado a 10 Junho de 2012).

[38] Mitch Bryson e Salah Sukkarieh. Architectures for cooperative airborne simultaneous localisation and mapping.

[39] Elói Teixeira Pereira. Cooperative control of teams of unmanned air vehicles. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2011.

[40] S. Bayraktar, G. E. Fainekos, e G. J. Pappas. Experimental cooperative control of fixed-wing unmanned aerial vehicles. Em *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, páginas 4292–4298 Vol.4, 2004.

[41] A. M. Bernardos, J. R. Casar, e P. Tarrio. Efficient social routing in sensor fusion networks. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–8, 2006.

[42] P. S. Dias, R. Goncalves, J. Pinto, J. B. Sousa, G. M. Goncalves, e F. L. Pereira. Mission review and analysis. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–6, 2006.

[43] D. Akselrod, A. Sinha, C. V. Goldman, e T. Kirubarajan. Efficient control of information flow for distributed multisensor fusion using markov decision processes. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–8, 2006.

[44] N. Floudas, A. Polychronopoulos, M. Tsogas, e A. Amditis. Multi-sensor coordination and fusion for automotive safety applications. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–8, 2006.

[45] S. Das, P. Kanjilal, e D. Lawless. Spatiotemporal clustering for aggregating hostile units in cluttered environments. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–8, 2006.

[46] F. Dambreville. Optimal area covering by sensors for planning a tracks collection. Em *Information Fusion, 2006 9th International Conference on*, páginas 1–7, 2006.

[47] Z. Alliance. Zigbee specification, 1st ed. `http://www.zigbee.org`,(visitado a 14 Fev 2012).

[48] H. Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425 – 432, apr 1980.

[49] Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2003*, páginas 0 –670, 2003.

[50] Iso/iec standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications (includes ieee std 802.11, 1999 edition; ieee std 802.11a.-1999; ieee std 802.11b.-1999; ieee std 802.11b.-1999/cor 1-2001; and ieee std 802.11d.-2001). *ISO/IEC 8802-11 IEEE Std 802.11 Second edition 2005-08-01 ISO/IEC 8802 11:2005(E) IEEE Std 802.11i-2003 Edition*, páginas 1 –721, 2005.

[51] Iso/iec standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements- part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 4: Further higher data rate extension in the 2.4 ghz band. *ISO/IEC 8802-11:2005/Amd.4:2006(E) IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11-1999)*, páginas c1 –68, 2006.

[52] Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, páginas c1 –502, 29 2009.

[53] North Atlantic Treaty Organization. Standard interfaces of uav control system (ucs) for nato uav interoperability. *North Atlantic Treaty Organization Brussels*, 2007.

[54] Katelijne Vandenbussche. *Fine-grained Indoor Localisation using wireless Sensor Networks*. Tese de doutoramento, Delft University of Technology, Delft, August 2005.

[55] Andreas Savvides, Chih-Chieh Han, e Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. Em *MobiCom01: Proceedings of the 7th annual international conference on Mobile computing and networking*, páginas 166–179, New York, NY, USA, 2001. ACM.

[56] Paramvir Bahl e Venkata N. Padmanabhan. Enhancements to the RADAR user location and tracking system. Relatório té MSR-TR-2000-12, Microsoft Research (MSR), Fevereiro 2000.

[57] Nissanka B. Priyantha, Allen K. L. Miu, Hari Balakrishnan, e Seth J. Teller. The cricket compass for context-aware mobile applications. Em *MOBICOM*, páginas 1–14, 2001.

[58] Kamin Whitehouse e David Culler. Calibration as parameter estimation in sensor networks. Em *WSNA02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, páginas 59–67. ACM, 2002.

[59] Yi Shang e Wheeler Ruml. Improved MDS-based localization. Em *INFOCOM*, 2004.

[60] Dragos Niculescu, Badri Nath, e Dataman Lab. Ad hoc positioning system(APS) using aoA, 2003.

[61] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, e Tarek F. Abdelzaher. Range-free localization schemes for large scale sensor networks. Em David B. Johnson, Anthony D. Joseph, e Nitin H. Vaidya, editores, *MOBICOM*, páginas 81–95. ACM, 2003.

[62] J. Albowicz, Alvin Chen, e Lixia Zhang. Recursive position estimation in sensor networks. Em *ICNP*, página 35. IEEE Computer Society, 2001.

[63] B. Hofmann-Wellenhof, H. Lichtenegger, e J. Collins. *Global Positioning System. Theory and practice*. Springer, Wien (Austria), 1993.

[64] Dragos Niculescu e Badri Nath. Ad hoc positioning system (APS) *. Relatório té, Computer Science Department,Rutgers University, 2001.

[65] Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, e Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sen. Netw.*, 2(2):188–220, 2006.

[66] Pubudu N. Pathirana, Nirupama Bulusu, Andrey V. Savkin, e Sanjay Jha. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Trans. Mob. Comput*, 4(3):285–296, 2005.

[67] Nissanka B. Priyantha, Anit Chakraborty, e Hari Balakrishnan. The cricket location-support system. Em *MOBICOM*, páginas 32–43, 2000.

[68] Nirupama Bulusu, John Heidemann, e Deborah Estrin. Gps-less low cost outdoor localization for very small devices, 2000.

[69] Srdjan Čaronapkun, Maher Hamdi, e Jean-Pierre Hubaux. GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, 5(2):157–167, Abril 2002.

[70] Chris Savarese e Jan M. Rabaey. Locationing in distributed ad-hoc wireless sensor networks, 2001.

[71] Horacio A. B. F. de Oliveira, Eduardo Freire Nakamura, Antonio Alfredo Ferreira Loureiro, e Azzedine Boukerche. Directed position estimation: A recursive localization approach for wireless sensor networks. Em *14th ACM International Workshop on Geographic Information Systems, (IC3N05), S. R. Thuel, Y. Yang, and E. K. Park, editors, San Diego, USA, October 2005, Proceedings*, páginas 557–562, 2005.

[72] Mihail L. Sichitiu e Vaidyanathan Ramadurai. Localization of wireless sensor networks with a mobile beacon, 2003.

[73] Xiang Ji. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. Em *INFOCOM*, 2004.

[74] E. D. Kaplan. *Understanding GPS-Principles and Applications*. Artech House Publisher, Boston, 1996.

[75] Yocto Project. Yocto project. `http://www.yoctoproject.org/`(visitado a 10 Junho de 2012).

[76] Harry Theodor Nyqvist. Certain topics in telegraph transmission theory, April 1928.

[77] Pekka Eskelinen. *Introduction to RF Equipment and System Design*. Artech House Publisher, 2004.

[78] Cameron Dean Whitehouse, David Culler, e Kristofer Pister. The design of calamari: an ad-hoc localization system for sensor networks. Relatório té, University of California at Berkeley, 2002.

[79] F. T. Ulaby, R. K. Moore, e A. K. Fung. *Microwave Remote Sensing: Active and Passive*, volume I. Artech House, London, UK, 1981.

[80] J. B. Andersen, T. S. Rappaport, e S. Yoshida. Propagation measurements and models for wireless communications channels. *Communications Magazine, IEEE*, 33(1):42–49, 1995.

[81] D. Lymberopoulos, Q. Lindsey, e A. Savvides. An empirical analysis of radio signal strength variability in IEEE 802.15.4 networks using monopole antennas. Em *European Workshop on Sensor Networks (EWSN)*, 2006.

[82] Chipcon; CC2420 – 2,4 GHz IEEE 802.15.4 / RF Transceiver (rev. 1.2); Manual do fabricante. `http://www.chipcon.com`.

[83] TinyOS. Tinyos tutorials. `http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials`(visitado a 10 Fevereiro de 2012).

[84] Oracle. Java se for embedded. `http://www.oracle.com/technetwork/java/embedded/downloads/javase/index.html`(visitado a 10 Maio de 2012).