# Sistema de transmissão de dados de uma aplicação remota

Projeto de Mestrado

João Carlos de Castro

Setembro 2015

Mestrado em Engenharia de
Telecomunicações e Redes de Energia

# Sistema de Transmissão de Dados de uma Aplicação Remota

João Carlos de Castro

Dissertação realizada sobre a supervisão do

*Professor Doutor Joaquim Amândio Rodrigues Azevedo*

Professor Auxiliar do
Centro de Competência de Ciências Exatas e da Engenharia da
*Universidade da Madeira*

# Resumo

O principal objetivo deste projeto foi propor um sistema de aquisição de dados de uma aplicação remota. Existem diversas aplicações que requerem a recolha de informação remota, sendo, para isso, necessário estabelecer um sistema de comunicação dedicado. Este trabalho procurou encontrar as melhores soluções para o problema em causa, testando e avaliado um sistema de comunicação.

Para testar o sistema foi desenvolvido um protótipo de monotorização de parâmetros ambientais, que mede periodicamente os valores de temperatura, humidade, luminosidade e pressão atmosférica. A comunicação entre sensores foi realizada com recurso a rádios XBee com protocolo Zigbee.

Foi, também, desenvolvido um nó de coordenação que tem como objetivo principal gerir e manter todo o sistema de aquisição de dados. Este protótipo recebe, valida e armazena num cartão SD todos os dados provenientes do nó sensor e periodicamente envia os dados para um servidor com acesso a *internet*.

A aquisição de dados em aplicações remotas, normalmente, é efetuada em zonas de ausência de energia elétrica. Então, tendo em consideração a capacidade reduzida dos sistemas de armazenamento de energia, foram desenvolvidos sistemas de alimentação através de energia solar, focando-se no mínimo de consumo possível.

Para a comunicação de longa distância utilizou-se um sistema de feixes hertzianos. Estudou-se a propagação, utilizando a banda isenta de licença dos 2,4 GHz. Projetou-se uma ligação entre dois pontos e procedeu-se à validação das áreas de cobertura, a qual requer a estimação do sinal nos pontos de interesse.

Verificou-se as zonas de interferência e as zonas onde o sinal é fraco ou está no seu limite. O desenvolvimento deste sistema de comunicação foi fundamentado com a análise e avaliação dos modelos de propagação, juntamente com o *software* criado em plataforma Matlab.

Finalmente foram apresentadas as conclusões e algumas sugestões de trabalhos futuros.

**Palavras-chave:** Aquisição de dados, rádio frequência, sistema de comunicação, feixes hertzianos, modelos de propagação, transmissão remota.

# Abstract

The main objective of this project was to propose a data acquisition system for remote applications. There are several applications which require remote information collection, and because of this, it is necessary to establish a dedicated communication system. This study tried to find the best solutions for the problem in question, testing and evaluated a communication system.

To test the system, an environmental parameters monitoring prototype was developed, which periodically measures temperature, moisture, light and atmospheric pressure values. Communication between sensors was carried out using XBee radios with the Zigbee protocol.

Also developed was a coordinating node that aims to manage and maintain the entire data acquisition system. This prototype receives all the data from the sensor nodes, validates and stores it on an SD card. Periodically sending the data to a server with Internet access.

Data acquisition in remote applications is usually performed in the absence of electricity areas. Then, taking into account the reduced capacity of energy storage systems, a solar powered energy system was developed, focusing on the least consumption possible.

A microwave system was used for long distance communication. The radio wave propagation was studied using the 2.4 GHz free licensed band. To analyze the coverage areas, which requires signal estimation in the points of interest, a connection between two locations was designed.

The interference zones and areas where the signal is weak or is at its limits was verified. The development of this communication system was justified with the analysis and evaluation of propagation models, together with software written in the Matlab platform.

We concluded this study with some findings and presented some suggestions for future work.

**Keywords**: Data acquisition, radio frequency, communication systems, microwave, propagation models, remote transmission.

# Agradecimentos

A realização deste trabalho constitui, para mim, a conclusão de uma etapa que há muito vinha sendo adiada. Todo este trabalho apenas foi possível com o contributo de algumas pessoas, a quem pretendo expressar a minha gratidão.

Em primeiro lugar gostava de agradecer ao meu orientador, o Professor Doutor Joaquim Amândio Azevedo, agradeço-lhe todo o apoio prestado, pela paciência, exigência que o caracteriza, disponibilidade e conhecimentos partilhados.

Agradeço também a todos os membros da academia com quem tive a felicidade de conhecer e privar nestes anos. Ao Eng. Filipe Santos pela amizade, apoio e experiencia partilhada. Aos meus colegas de curso, em especial ao Nuno Carreira, Vítor Wilson, Sérgio Pestana, Davide Inácio, Miguel Quintal, Miguel Teixeira, Jorge Valente e ao Siarhei Tamulionak pelo companheirismo, momentos de diversão, ajuda e amizade sincera.

Um agradecimento especial para a minha família, mãe, pai, esposa e aos meus filhos pelo apoio nos bons e maus momentos, pois sem eles nada seria possível.

A todos o meu muito obrigado.

# Índice

# Lista de acrónimos

ABS - *Acrylonitrile Butadiene Styrene*
ADC- *Analog-to-Digital Converter*
ANACOM- Autoridade Nacional de Comunicações
API- *Application Programming Interface*
CIP- Condições Ideais de Propagação
DHCP- *Dynamic Host Configuration Protocol*
DRIGOT- Direção Regional de Informação Geográfica e Ordenamento do Território
EIRP- *Effective Isotropically Radiated Power*
GPS- *Global Positioning System*
GUI- *Guided User Interface*
IP- *Internet Protocol*
IPMAInstituto Português do Mar e da Atmosfera
ISM- *Industrial, Scientific and Medical*
ITM- *Irregular Terrain Model*
LAN- *Local Area Network*
LIDAR- *Light Detection And Ranging*
LOS- *Line of Sight*
MEMS-*Microelectromecanical*
MG- *Magnetic North*
NLOS- *Non Line Of Sight*
NTP- *Network Time Protocol*
PAN ID- *Personal Area Network ID*
PHP- *Hypertext Preprocessor*
POE- *Power Over Ethernet*
PWM- *Pulse Width Modulation*
RF- Rádio Frequência
ROM- *Read-Only Memory*
RPSMA- *Reverse Polarity SubMiniature version A*
RSSI- *Received Signal Strenght Indication*
SD- *Secure Digital*
SOC- *State of Charge*
SPI- *Serial Peripheral Interface*
SRTM- *Space Shuttle Radar Terrain Mapping Mission*
SWR- *Standing Wave Ratio*
TCO- *Temperature Coefficiente of Offset*
TCS- *Temperature Coefficient of Sensitivity*
TN- *True North*
UARTS- *Universal Asynchonous Recever/Transmitter*
UBNT- *Ubiquiti Networks*
UHF- *Ultra High Frequency*
USGS- *U. S. Geological Survey*

UTM- *Universal Transverse Mercator*
UTP- *Unshielded Twisted Pair*
VHF- *Very High Frequency*
WiFi- *Wireless Fidelity*
WLAN- *Wireless Local Area Network*
WSN- *Wireless Sensor Network*

# Índice de Figuras

# Índice de Tabelas

# 1. Introdução

Um sistema de transmissão tem como objetivo fundamental a transmissão de informação entre dois pontos. A informação é transmitida através de canais de comunicação adequados.

Na implementação de uma rede de sensores sem fios a transmissão de dados tem um alcance relativamente curto. O recetor pode recolher dados a partir de um grande número de sistemas de controlo. A transmissão de dados de aplicações remotas depende muito da localização geográfica. Não existe apenas uma única tecnologia que satisfaça as necessidades de todos os utilizadores

Os sistemas de feixes hertzianos são normalmente utilizados com o intuito de fornecer uma ligação, sem recurso a fios, para um sistema de transmissão separado por uma grande distância. A propagação faz-se normalmente em linha de vista, em que a ausência de suporte físico torna este tipo de ligação vantajosa para distâncias elevadas ou quando existem obstáculos, como rios ou montanhas.

## 1.1 - Motivação e Objetivos

Existem diversas aplicações que requerem a recolha de informação remota, sendo, para isso, necessário estabelecer um sistema de comunicação dedicado. Este trabalho tem como objetivo principal encontrar as melhores soluções para o problema em causa testando e avaliado um sistema de comunicação.

Englobado neste objetivo principal sujem os objetivos secundários.

Para validar o sistema de comunicação foi necessário criar uma rede de monotorização sem fios, com o objetivo de recolher os dados de temperatura, humidade, luminosidade e pressão atmosférica.

Os dados são depois armazenados num nó coordenador que tem por objetivo a gestão e o envio para um servidor com acesso a internet, recorrendo a um sistema de feixes hertzianos.

Após a entrega dos dados, o servidor tem como objetivo armazenar os dados e permite a sua visualização gráfica.

Este sistema tem de poder ser colocado em locais remotos, assim, existi-o a necessidade de apresentar um sistema autónomo, portátil e alimentado a energia solar.

Finalmente será realizado o estudo de propagação entre o local de emissão e o de receção, apresentando e analisando as melhores soluções ao sistema proposto.

A execução deste projeto de Mestrado tem vista a vontade de apresentar um trabalho prático na resolução um problema real na área das telecomunicações.

## 1.2 - Estrutura da tese

Este documento apresenta-se estruturado em 6 capítulos, organizados da forma que se segue:

No primeiro capítulo é apresentada uma introdução ao projeto assim como a motivação e objetivos a cumprir no decorrer desta tese.

Segue-se um capítulo que visa introduzir a teoria necessária à compreensão e contextualização do presente trabalho. São apresentados os principais modelos de propagação em terrenos irregulares, vegetação e condições atmosféricas. Os sistemas de feixes hertzianos serão também abordados, descrevendo uma possível abordagem teórica e principais elementos a tomar em consideração na projeção de uma ligação ponto a ponto.

No seguimento do trabalho são descritos os *softwares* de propagação gratuitos, avaliados para este trabalho, e o programa de simulação desenvolvido perante a análise e previsão das perdas de propagação ao longo do percurso, em plataforma MATLAB, para planeamento de uma ligação rádio frequência.

No quarto capítulo relata-se o desenvolvimento de todo o sistema de transmissão e receção de dados de uma aplicação remota. Para isso, descrevem-se os protótipos desenvolvidos para a rede de sensores sem fios implementada, juntamente com o sistema de comunicação e *software* utilizado.

Na sequencia deste trabalho são descritos os testes e resultados obtidos no sistema de transmissão de dados de uma aplicação remota. São apresentadas as considerações a seguir num projeto de ligação de um sistema de feixes hertzianos e analisados os locais onde se efetuou os testes, validando os modelos de *software* desenvolvidos.

Para a finalização do trabalho desenvolvido apresenta-se o capítulo onde são apresentadas as conclusões finais, tendo em conta a contribuição para o desenvolvimento de um sistema de transmissão de dados de uma aplicação remota. São também referidos trabalhos a realizar futuramente com vista a otimização do projeto.

# 2. Modelos de Propagação e Feixes Hertzianos

Neste capítulo são apresentados os principais modelos de propagação em terrenos irregulares, vegetação e condições atmosféricas que servem de base a este trabalho. Os sistemas de feixes hertzianos serão também abordados, descrevendo uma possível abordagem teórica e principais elementos a tomar em consideração na projeção de uma ligação ponto a ponto.

## 2.1 Sistemas de rádio frequência

Um sistema de telecomunicações tem como objetivo fundamental a transmissão de informação entre dois pontos. A informação é transmitida através de canais de comunicação adequados, que podem ser, par de cobre, fibra ótica ou canal rádio.

A tecnologia sem fios tem vindo a conquistar um papel cada vez mais relevante nas comunicações. Impulsionada pelas comunicações móveis, aliadas aos *smartphones* e à demanda de informação, continuam a contribuir para o crescimento vertiginoso desta tecnologia. Ainda, a tecnologia sem fios tem sido aplicada durante muitos anos para garantir comunicações seguras na monitorização e controlo de processos, incluindo aplicações críticas. A escolha para uma determinada aplicação tem de tomar em consideração o desempenho e as características que melhor se adequam. Existem muitos tipos de tecnologias sem fios, diferindo uns dos outros nas suas técnicas de modulação, taxas de transmissão de dados, distância de transmissão, segurança, complexidade, custo e consumo de energia [1].

Em Portugal, a entidade responsável pela regulação e licenciamento dos segmentos de espetro de rádio frequência, destinados às tecnologias sem fios recai sobre a alçada da ANACOM (Autoridade Nacional de Comunicações) que é responsável pela atribuição de frequências, níveis de potência e ganhos das antenas do sistema. Devido à regulamentação do espetro, as tecnologias sem fios isentas de licença, ISM (*Industrial, Scientific and Medical*) vêm oferecendo diversas opções na resolução de necessidades específicas das aplicações [2].

A banda dos 2,4 GHz é geralmente reservada a transmissões isentas de licença. Como resultado, o congestionamento poderá ser um potencial problema. Existem situações em que se torna necessário o sacrifício da eficiência em função de uma comunicação segura. No entanto, o uso de tecnologia sem fios na banda *ISM* tem vindo a aumentar drasticamente nos últimos anos. Além de ter suplementado as tecnologias licenciadas no que diz respeito a comunicações de curtas e médias distâncias (alguns quilómetros), foram descobertas muitas novas aplicações.

A implementação de um sistema RF (Rádio Frequência) permite a transmissão de dados de alcance relativamente curto (alguns quilómetros). A vantagem destes sistemas é que um recetor pode recolher dados a partir de um grande número de sistemas de controlo remoto. Isso permite que exista uma estação base com localização central ou

uma plataforma móvel que pode mover-se dentro do alcance das estações de monitorização e recolher todos os dados sem realmente visitar o local. Uma vantagem adicional é que, além da manutenção ocasional, não existem custos suplementares com um sistema de rádio depois da sua implementação.

Os sistemas de transmissão de dados de aplicações remotas, que dependem muito da localização geográfica, podem apresentar várias condicionantes, como a existência ou não de infraestruturas de comunicação, ou até energia elétrica. Não existe apenas uma única tecnologia que satisfaça as necessidades de todos os utilizadores e, como resultado, são utilizadas várias tecnologias dependo apenas da aplicação.

Os sistemas de feixes hertzianos são normalmente utilizados com o intuito de fornecer uma ligação, sem recurso a fios, para um sistema de transmissão separado por uma grande distância. A transmissão de informação de sistemas de televisão, satélite, controlo aéreo e sistema de comunicação de voz têm a sua presença bem vincada nas ligações por feixes hertzianos.

A propagação faz-se normalmente em linha de vista LOS (*Line of Sight*). Sempre que o comprimento da ligação excede os 50 km, ou que a orografia o impõe, torna-se necessário a utilização de repetidores [3].

O recurso a frequências elevadas (normalmente superiores a 1 GHz), dada a largura de banda disponível, permite atingir capacidades de transmissão elevadas. A grande vantagem dos sistemas de rádio frequência é a ausência de suporte físico, tornando este tipo de ligação vantajosa para distâncias elevadas ou quando existem obstáculos, como rios ou montanhas.

## 2.2 Modelos de propagação

No projeto de um sistema de comunicação é essencial que os valores obtidos pelos modelos de propagação se aproximem o mais possível da realidade. No entanto, grande parte dos modelos de propagação apenas fornecem um valor médio de sinal, sendo necessário conhecer a estatística do sinal para a obtenção da sua variação. O conhecimento das distribuições estatísticas leva a uma compreensão adequada do comportamento do sinal no meio de propagação. Para a determinação correta do modelo a usar, tem-se que levar em conta todos os fatores que podem influenciar a propagação do sinal no canal de transmissão.

A bibliografia, apesar de já ser extensa no que diz respeito aos modelos de propagação, ainda não apresenta um modelo que seja capaz de prever a atenuação do sinal em qualquer meio [4].

O planeamento das áreas de cobertura de uma aplicação remota requer a estimação do sinal nos pontos de interesse. Torna-se, por isso, crucial prever as zonas de interferência e as zonas onde o sinal é fraco ou está no seu limite. A previsão destes sinais requer uma estimativa do seu do valor médio e as respetivas variações.

A propagação do sinal é geralmente dependente de dois tipos de desvanecimento, um lento, que depende da distância, com distribuição *log-normal*, e um rápido, normalmente relacionado com o movimento do terminal e ao fenómeno de multipercurso com distribuição de Rice.

Os modelos de propagação encontram-se divididos em duas categorias distintas:

- **Empíricos** - modelos baseados em medições, apresentando relações simples entre a atenuação e a distância.
- **Teóricos** - modelos que requerem bases de dados geográficas, utilizando métodos de ligações fixas.

Os modelos empíricos são baseados em medidas experimentais, e são, caracterizados por curvas e equações que se adequam às medições efetuadas. Têm como principal vantagem contabilizar a totalidade dos fatores que influenciam a propagação. A principal desvantagem reside na necessidade de estarem sujeitos a validação, para os diferentes locais, frequências, ambientes e condições de medição.

Os modelos teóricos são caracterizados por não contabilizarem todos os fatores, não têm em consideração o tipo de ambiente, permitem alterações aos parâmetros e dependem da existência de bases de dados geográficas.

Os modelos híbridos, além de utilizarem as vantagens dos anteriores, são flexíveis. São realizados com medições nos locais de propagação e conjuntamente com os modelos teóricos são minimizados os erros. Para a aplicação de modelos com parte empírica, existe a necessidade da classificação dos diversos ambientes em três grandes categorias: rural, suburbano e urbano [4].

Os modelos de propagação são extremamente relevantes no que diz respeito à propagação do sinal de transmissão. Os modelos em *Espaço Livre, Obstáculo Isolado em Lâmina*, *Epstein-Petterson*, *Deygout*, *ITU-R P530-14*, *Egli* e o Modelo de *Longley-Rice* são alguns dos principais modelos de propagação que podem ser utilizados em terreno irregular na frequência de operação de 2,4 GHz [4].

As características do terreno são as principais causas da alteração na propagação das ondas eletromagnéticas. No entanto, nos modelos base apenas está contemplado o relevo do terreno. A vegetação, os edifícios e as estruturas construídas não estão definidas. Esta falta levou a que fossem investigados mecanismos de propagação em áreas de floresta. Os modelos de Weissberger [5], Cost 235 [6], ITU [7], log-normal [8] e Azevedo *e Santos* [9] são exemplos de modelos utilizados na estimação de perdas em ambiente florestal. Estes baseiam-se principalmente em medições na banda dos UHF (*Ultra High Frequency*) e foram desenvolvidos modelos empíricos válidos nas frequências de 200 MHz a 95 GHz [10].

O desempenho de sistemas de rádio frequência é limitado pelo percurso de transmissão entre o transmissor e o recetor, que pode variar desde um canal em linha de vista até um severamente afetado por obstruções como edifícios, montanhas ou folhagem [8].

A propagação no espaço livre acontece quando existe comunicação entre antenas sem reflexão ou absorção das ondas eletromagnéticas. Isto raramente acontece e na prática é suficiente manter a primeira zona de Fresnel livre de obstáculos, como mostra a Figura 2.1. A primeira zona de Fresnel é determinada pela superfície do elipsoide de revolução, com as antenas de transmissão e receção nos focos, em que a onda refletida tem um percurso de meio comprimento de onda superior ao percurso direto entre as antenas [4].

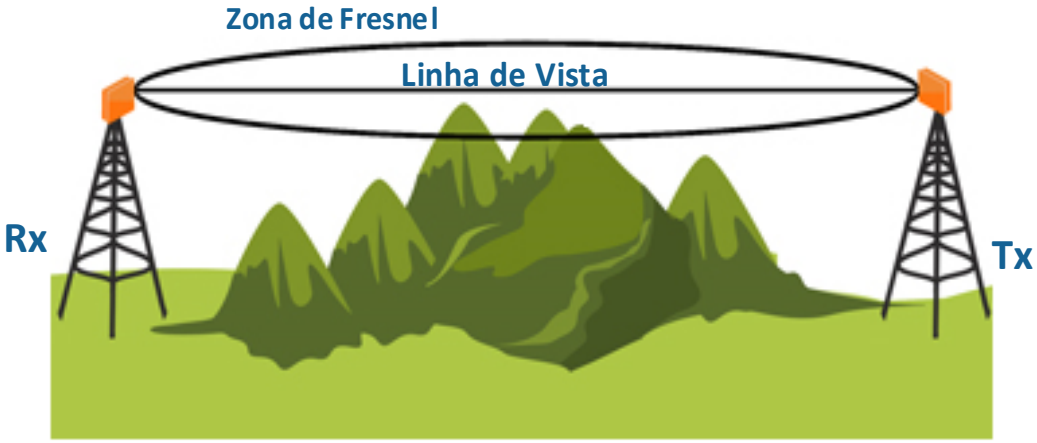



Figura 2.1 - Elipsoide de Fresnel [11]

A difração, reflexão e desvanecimento são os fenómenos mais importantes quando se fala numa propagação em que não exista linha de vista. Os sistemas de transmissão de rádio frequência recorrem à difração e reflexão em telhados, esquinas de edifícios e outros obstáculos para assegurarem a cobertura de rede.

## 2.3 Modelos em terrenos irregulares

Em sistemas de comunicações terrestres, a geografia do terreno prejudica significativamente a propagação das ondas eletromagnéticas. O relevo da área de transmissão tem de ser considerado quando se estima a atenuação. A presença de árvores, edifícios e outros obstáculos afetam significativamente a previsão das perdas de percurso.

Com a existência da Terra, existe a reflexão no solo e a difração terrestre deve-se à curvatura da terra. Os terrenos irregulares podem produzir perdas por difração, zonas de sombra, zonas de bloqueio e reflexões múltiplas mesmo em pequenas distâncias. O propósito de um modelo de terreno é proporcionar uma medida média da perda de percurso como uma função entre a distância e do tipo de terreno.

Existe um número considerável de modelos que tentam prever a atenuação em terrenos irregulares. Os métodos variam, assim, como a complexidade e a sua precisão.

### 2.3.1 Modelo em espaço livre

Existem vários modelos de atenuação, sendo o mais simples o da perda em espaço livre [4]. O seu valor apenas depende da frequência e da distância. No entanto, mesmo em condições ideais e perante a inexistência de obstáculos entre o transmissor e recetor, a potência recebida é sempre inferior à potência emitida. A Figura 2.2 apresenta este processo de perda de energia.

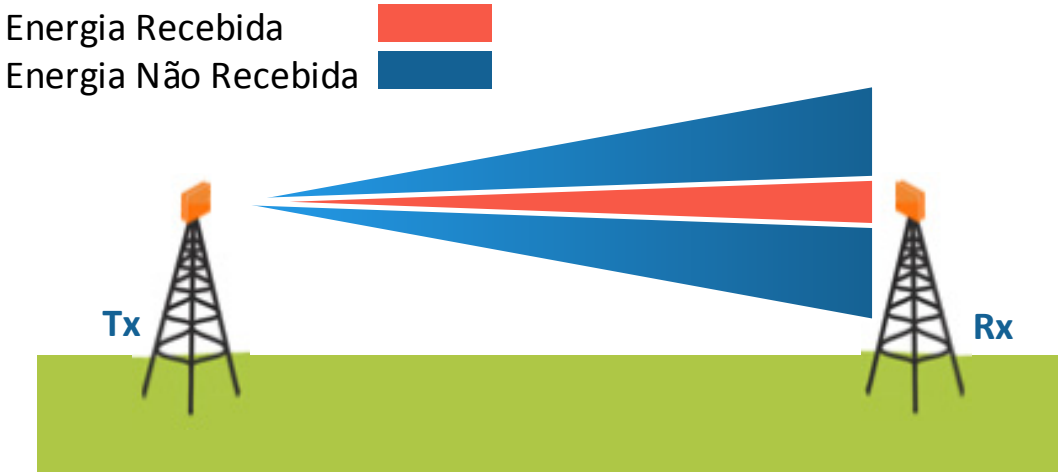



Figura 2.2 - Modelo em espaço livre.

Para o modelo em espaço livre, a perda de percurso pode ser determinada pelo fator da equação de *Friis*, dado por:

$$PL = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{2.1}$$

onde $\lambda$ é o comprimento de onda (m) e $d$ é a distância entre antenas em metros. A equação 2.1 pode ser reescrita em dB por:

$$PL_{FS}\ (\text{dB}) = 32{,}4 + 20\log_{10}(d_{[\text{km}]}) + 20\log_{10}(f_{[\text{MHz}]}) \tag{2.2}$$

Trata-se de um modelo teórico válido quando o primeiro elipsoide de *Fresnel* se encontra livre de obstáculos. Para antenas próximas da Terra deve-se incluir a reflexão no solo e a perda de percurso então é dada por:

$$\text{PL}_{\text{GR}}\ (\text{dB}) = PL_{FS} - 20\log_{10}\left[2\left|\sin\left(\frac{\beta h_{Tx} h_{Rx}}{d}\right)\right|\right] \tag{2.3}$$

em que $h_{Tx}$, e $h_{Rx}$, são as alturas das antenas emissora e recetora, respetivamente, e $\beta = 2\pi/\lambda$. Para grandes distâncias em relação ao emissor, $d \gg \beta h_{Tx} h_{Rx}$ verifica-se que (2.3) apenas depende de $h_{Tx}$, $h_{Rx}$ e $d$. Neste caso a atenuação excede o modelo de espaço livre [4].

### 2.3.2 Modelo de obstáculo em lâmina

Utiliza-se o modelo de difração por obstáculo isolado em lâmina (*knife-edge*), apresentado na Figura 2.3, quando existe um obstáculo normal à direção de propagação.

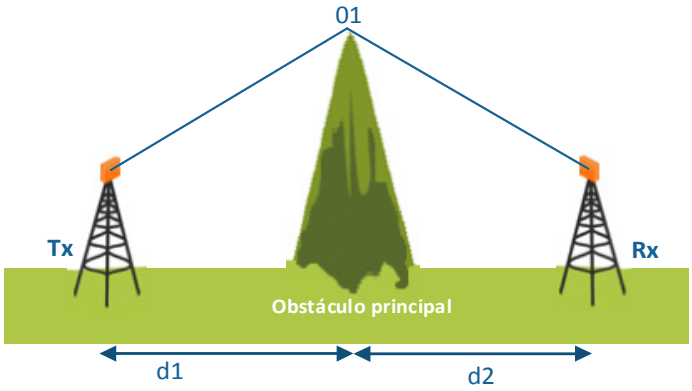



Figura 2.3 - Modelo de difração em obstáculo em lâmina isolado.

No caso ideal, o obstáculo caracteriza-se por um parâmetro de difração de *Fresnel-Kirchhoff*. Este parâmetro é adimensional e é dado por:

$$v = h\sqrt{\frac{2(d_1 + d_2)}{\lambda d_1 d_2}} \tag{2.4}$$

em que $\lambda$ é o comprimento de onda, $h$ é a altura do obstáculo acima do raio direto, $d_1$ *e* $d_2$ representam as distâncias entre o obstáculo e as antenas.

A atenuação suplementar devido ao obstáculo em dB é descrita por:

$$PL_{ke}(v) = -20\log_{10}\left(\frac{1}{2}\left(\frac{1}{2} + C^2(v) - C(v) + S^2(v) - S(v)\right)\right) \tag{2.5}$$

em que $C$ representa a função cosseno integral e $S$ a função seno integral [12]. Na Figura 2.4 tem-se uma representação gráfica da função.

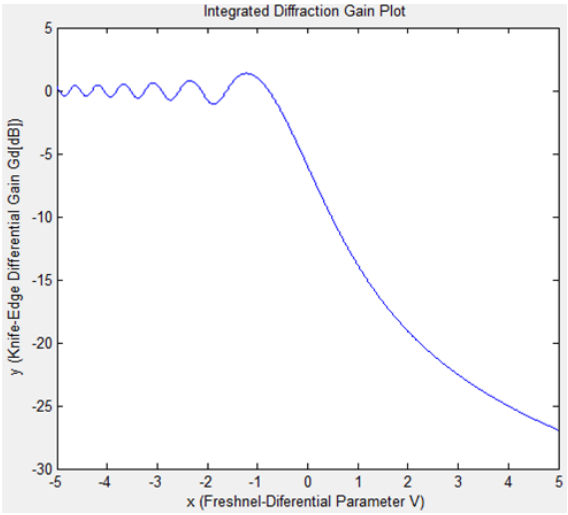



Figura 2.4 - Representação gráfica da atenuação suplementar devido ao obstáculo.

No entanto, na maior parte dos casos práticos de propagação, a transmissão não pode representar-se apenas por um único obstáculo entre o transmissor e o recetor [4].

### 2.3.3 Modelo de Epstein-Petterson

O modelo *Epstein-Petterson* [4] divide o percurso total em sub-percursos, cada um contendo um obstáculo. É calculada a atenuação para cada um dos percursos e a atenuação total é dada pela soma dos sub-percursos.

Para se aplicar o modelo, começa-se por desenhar uma linha entre o emissor Tx até ao vértice do obstáculo representado na Figura 2.5 por 02, a perda do percurso $d1$ - $d2$ que contém o obstáculo 01 é calculada com base no modelo de obstáculo em lâmina e a altura é dada por $h1$. De forma semelhante é calculada a perda para os percursos seguintes, contendo os obstáculos 02 e 03. Para o obstáculo 02 o percurso considerado é o composto por $d2 + d3$, (contém os obstáculos 01, 02 e 03). É traçada uma linha entre 01 e 03 e a altura utilizada é $h2$, fazendo uma perpendicular entre o topo do obstáculo e linha traçada. Finalmente, é usado o mesmo processo para o obstáculo 03, que contém o percurso composto por $d3$, $d4$ e Rx [13].

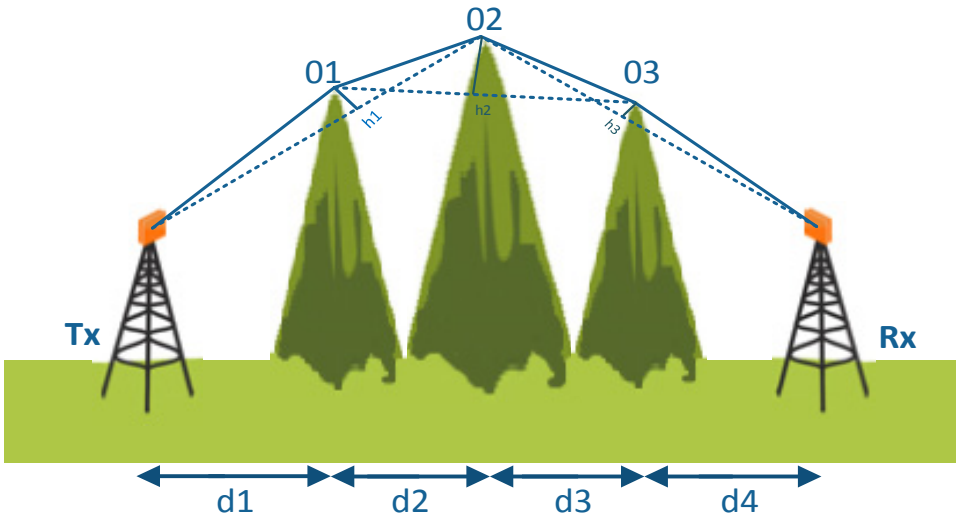



Figura 2.5 - Representação gráfica do Modelo de *Epstein-Petterson*.

Quando existem dois obstáculos isolados em lâmina bastante próximos a comparação de resultados revela erros por excesso [4]. Desta forma, foi deduzida uma correção que deverá ser aplicada quando os parâmetros $\nu$ são muito superiores à unidade. A correção deve ser somada à perda, sendo esta dada por:

$$L'(\text{dB}) = \log_{10}(\text{cosec}\,\alpha) \tag{2.6}$$

onde para dois obstáculos 1 e 2 se tem

$$\text{cosec}\,\alpha = \left[\frac{(d_1 + d_2)(d_2 + d_3)}{d_2(d_1 + d_2 + d_3)}\right]^{\frac{1}{2}} \tag{2.7}$$

O modelo de *Epstein-Petterson* sem correções leva a resultados demasiado otimistas e não leva em consideração atenuações urbanas. Quando elas existem, tem-se de considerar uma perda suplementar de 10 dB ou mais, que tem de ser adicionada [4].

### 2.3.4 Modelo de *Deygout*

O modelo de *Deygout* [4] é frequentemente referido como método do obstáculo principal devido a considerar o obstáculo que introduz maior atenuação. Para se selecionar o principal, tem-se que calcular os parâmetros $\nu$ de cada um dos obstáculos suprimindo os restantes e, para aquele que tiver maior valor, é então calculada a sua perda.

Na Figura 2.6 contendo três obstáculos, começa-se por dividir o percurso em três sub-percursos, Tx-01-Rx, Tx-02-Rx e Tx-03-Rx. Calcula-se os parâmetros $\nu$ e, neste caso, o obstáculo 02 é o obstáculo de maior valor. Então, os obstáculos 01 e 03 são calculados com referência a 02. Generalizando, para um percurso com vários objetos, a perda é calculada como sendo o somatório das perdas individuais, para todos os obstáculos, por ordem decrescente de parâmetros $\nu$, uma vez que o processo é repetido recursivamente.

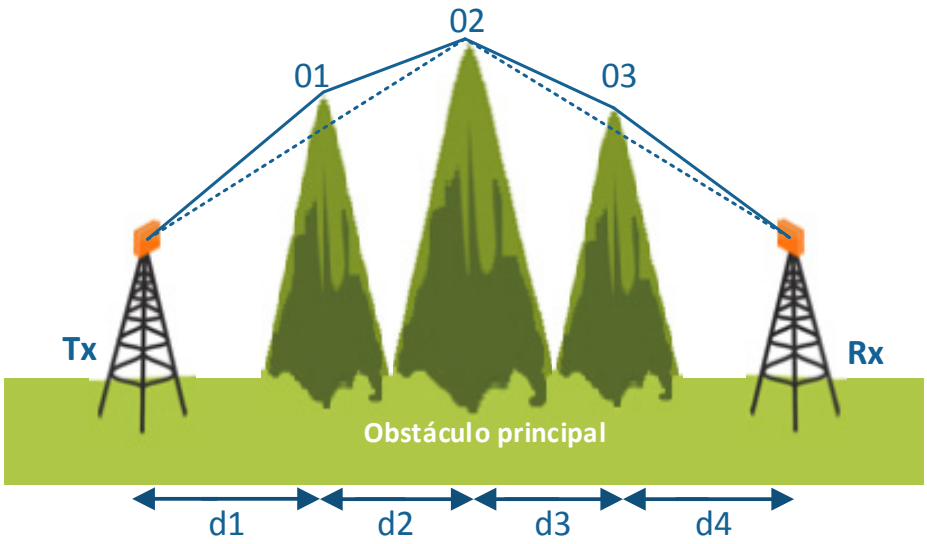



Figura 2.6 - Representação gráfica do Modelo de *Deygout*.

As estimativas de perda, utilizando este método, geralmente demonstram grande coerência, com boas aproximações à perda de percurso. No entanto, existem alguns problemas por excesso quando existem vários obstáculos ou quando os obstáculos estão muito próximos. O modelo tem um melhor desempenho quando existe claramente um obstáculo de referência, em relação aos restantes. No entanto, obtendo dois obstáculos comparáveis, já existem correções presentes na literatura [4].

Quando $\nu_1 \geq \nu_2$ e $\nu_1, \nu_2, (\nu_2 \operatorname{cosec} \alpha - \nu_1 \cot \alpha) > 1$ a correção é dada por:

$$L' = 20 \log_{10}( \operatorname{cosec}^2 \alpha - \frac{\nu_2}{\nu_1} \operatorname{cosec} \alpha \cot \alpha ) \qquad (2.8)$$

### 2.3.5 Modelo ITU-R P530-14

No modelo ITU-R P530-14 [7], as perdas de um determinado percurso são baseadas na teoria da difração, que toma em consideração o tipo de terreno e a vegetação. Para um dado percurso, o valor mínimo, quando apenas existe um obstáculo, é obtido pelo método do obstáculo em lâmina e o valor máximo é obtido pelo modelo da terra esférica. Com base nestes modelos, obtém-se uma forma rápida de determinar a atenuação média [14]. O modelo é representado na Figura 2.7 e descrito por:

$$A_d(\text{dB}) = -20\, h/\, F_1 + 10 \tag{2.9}$$

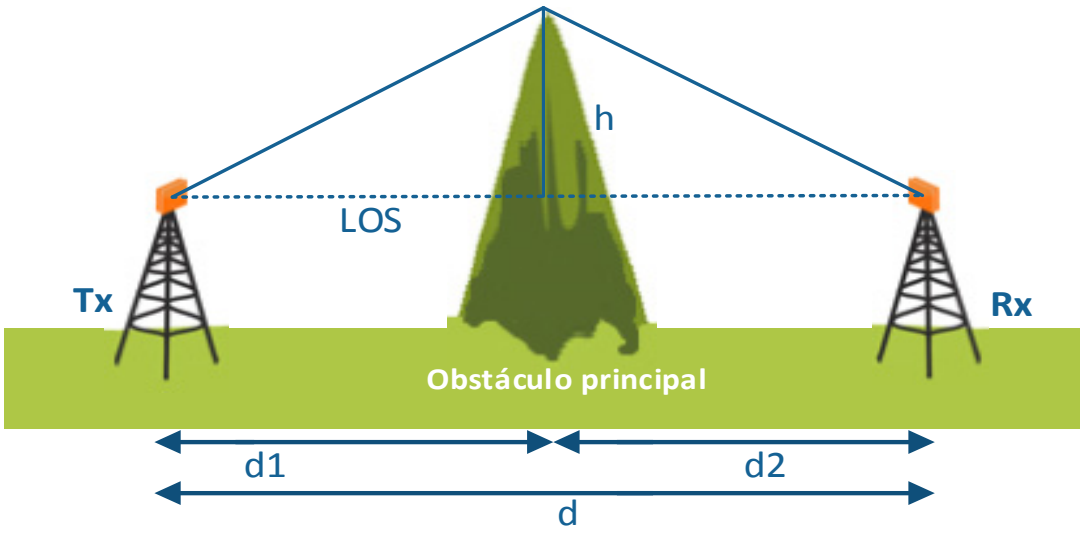



Figura 2.7 - Modelo ITU.

em que $h$ é a diferença de altura do percurso com o obstáculo mais significativo e o percurso em linha de vista entre o emissor e o recetor. No caso de o obstáculo estar acima da linha de vista, $h$ é negativo. O raio do primeiro elipsoide de *Fresnel*, é dado por:

$$F_1\,(m) = 17{,}3\sqrt{\frac{d_1 d_2}{f d}} \tag{2.10}$$

no qual $d_1$ e $d_2$ são as distâncias de cada terminal ao obstáculo, $d$ é a distância entre terminais e $f$ é a frequência em MHz. $h/\,F_1$ é o parâmetro que representa a normalização da área de terreno livre de obstáculos. Quando o perfil do terreno bloqueia a linha de vista tem-se $h/F_1 < 0$. A Figura 2.8 apresenta os gráficos de perda estimada em função do quociente $h/\,F_1$ da rugosidade do terreno.

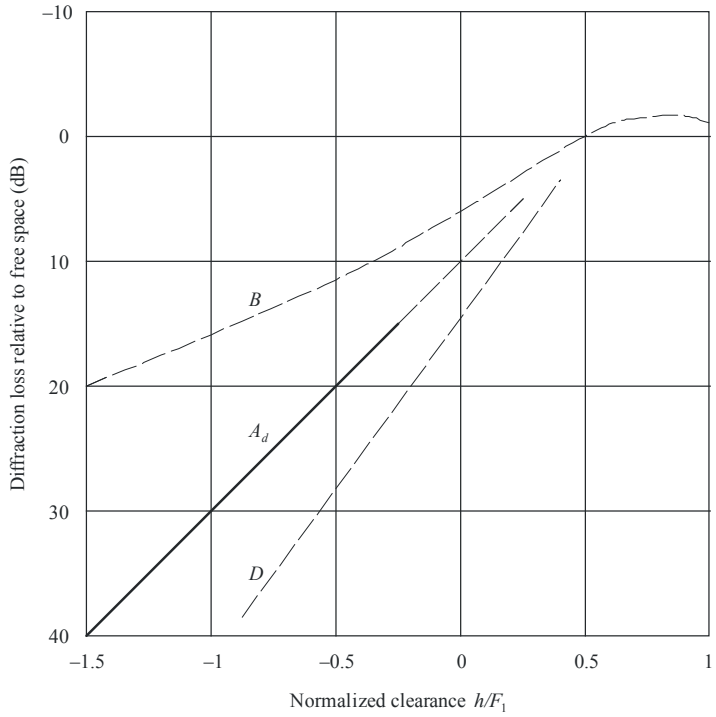



Figura 2.8 - Perda por difração em ligações RF para ligações em linha de vista [7].

O modelo teórico para um único obstáculo é representado pela curva B, enquanto a curva D representa o modelo de terra esférica a 6,5 GHz utilizando $4/3$ do raio da terra. A curva $A_d$ representa o modelo de propagação ITU com rugosidade média definida pela equação (2.10).

As curvas definem as perdas suplementares à perda em espaço livre. Este modelo normalmente é considerado quando as perdas são superiores a 15 dB, mas como se verifica pode ser extrapolado até aos 6 dB [7].

### 2.3.6 Modelo *Egli*

O modelo *Egli* é um modelo de propagação de RF em terrenos irregulares. Foi baseado em dados reais de transmissão de televisão em UHF (*Ultra High Frequency*) e VHF (*Very High Frequency*) em várias grandes cidades. O modelo prevê a atenuação para uma transmissão ponto-a-ponto. Sendo normalmente utilizado em transmissão em linha de vista, LOS (*Line-Of-Sight*), prevê a atenuação total da ligação, ao contrário de outros modelos que apresentam uma perda adicional à perda em espaço livre. Tipicamente é utilizado em comunicações móveis, no qual existe uma antena fixa e uma outra móvel. No entanto, pode ser aplicado em terrenos irregulares. Tem como desvantagem não tomar em consideração obstruções compostas por vegetação ou folhagem [4].

A perda de percurso é representada por:

$$PL_{50} = G_{Tx}\, G_{Rx} \left(\frac{h_{Rx} h_{Tx}}{d^2}\right)^2 \left(\frac{40}{f_{[MHz]}}\right)^2 \tag{2.11}$$

em que $G_{Tx}$ e $G_{Rx}$ representam os ganhos da antena base e da antena móvel, $h_{Tx}$ e $h_{Rx}$ são as alturas das antenas da base e móvel em metros e $d$ é a distância entre os terminais em metros. Egli assumindo uma distribuição log-normal da variação da altura do terreno, calculou uma família de curvas, representadas na Figura 2.9, que devem ser somadas à perda média da ligação.

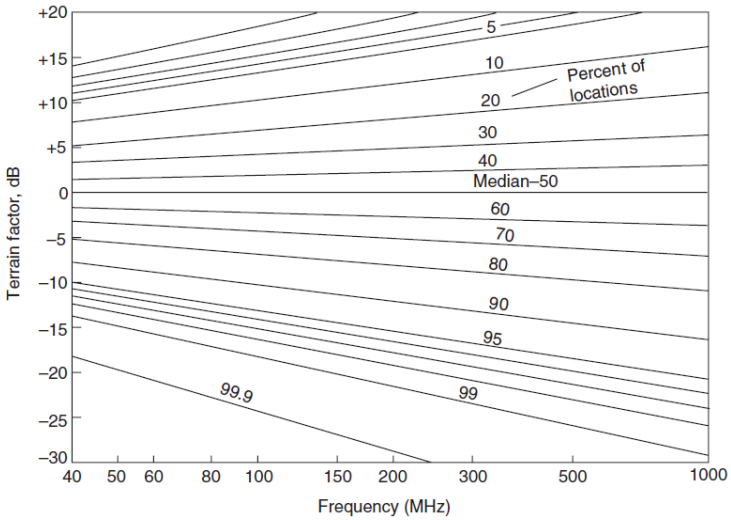



Figura 2.9 - Fator do terreno em relação a frequência [4].

Delisle, tendo em consideração o trabalho de Egli apresentou uma aproximação computacional mais simples [12].

$$PL_{Egli} = 40\ \log_{10}(d) + 20\ \log_{10}(f) - 20\ \log_{10}(h_{Tx}) + L_m \tag{2.12}$$

$$L_m = \begin{cases} 76{,}3 - 10\log_{10}(h_{Rx}) & se\ h_{Rx} < 10 \\ 76{,}3 - 20\log_{10}(h_{Rx}) & se\ h_{Rx} \geq 10 \end{cases} \tag{2.13}$$

para antenas com alturas elevadas a perda prevista pela equação (2.12) poderá ser inferior às perdas por espaço livre, devendo-se, neste caso, utilizar esse valor.

### 2.3.7 Modelo de Longley-Rice

O modelo de Longley-Rice foi desenvolvido na década de 60 e tem evoluído ao longo dos anos. É um dos modelos mais detalhados, tendo em consideração o tipo de terreno, o clima, o tipo de solo e a curvatura da terra [4] [14] [15].

O modelo Longley-Rice também é conhecido por ITM (*Irregular Terrain Model*) É um modelo pormenorizado, bem documentado, e geralmente usado em sistemas de comunicação, em terrenos irregulares. Cobre uma gama de frequências desde os 20 MHz até aos 20 GHz, suportando distâncias de 1 km até aos 2000 km, com antenas de polarização horizontal ou vertical. Por ser um modelo complexo, torna-se necessário recorrer a um programa de computador para introduzir todas as variáveis. O programa tem em consideração a altura das antenas em relação ao solo, o nível de refração do solo, o raio efetivo da terra, constantes do solo e clima.

O modelo de Longley-Rice contempla dois modos de operação, o Modo de Área e o Modo Ponto-a-Ponto. O primeiro é utilizado nos casos em que não se conhece o perfil do terreno do percurso definido. São, por isso, utilizados os parâmetros estatísticos e ambientais para estimar as perdas de propagação. Pode-se consultar estes parâmetros nas tabelas do Anexo I. O segundo modo utiliza mapas digitais com informações detalhadas sobre o relevo do terreno juntamente com os parâmetros estatísticos ambientais, de forma a prever com maior exatidão as perdas do percurso no terreno.

A Figura 2.10 representa a geometria utilizada pelo modelo de Longley-Rice. As distâncias dependem do relevo do terreno, ao longo do percurso. É necessário fornecer a altura efetiva das antenas, a distância horizontal das antenas $d_{Lb}$ e $d_{Lm}$, os ângulos de elevação $\theta_{eb}$e $\theta_{em}$ e a distância angular de transmissão $\theta_e$ .

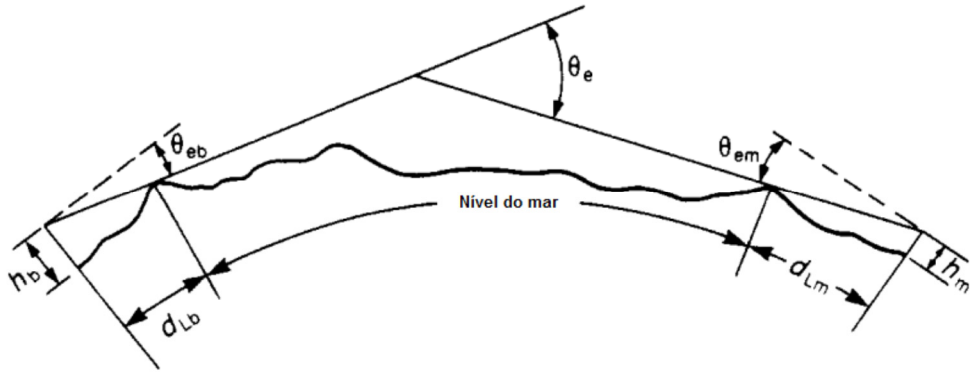



Figura 2.10 - Geometria associada ao modelo de *Longley-Rice* [4].

As distâncias $d_{Lb}$ e $d_{Lm}$ são dadas atrvés de [4]:

$$d_{Lb} = \sqrt{17h_{eb}}exp\left(-0{,}07\sqrt{\left(\frac{\Delta h}{h_{eb}}\right)}\right) \tag{2.14}$$

$$d_{Lm} = \sqrt{17h_{em}}exp\left(-0{,}07\sqrt{\left(\frac{\Delta h}{h_{em}}\right)}\right) \tag{2.15}$$

Para locais próximos à antena emissora, o modelo Longley-Rice adota a propagação em linha de vista. Para distâncias médias usa uma dupla difração com o obstáculo em forma de lâmina (*knife edge*) e finalmente um modelo de propagação troposférico, para distâncias elevadas. A Figura 2.11 apresenta um exemplo das faixas de atenuação ao longo do percurso de propagação.

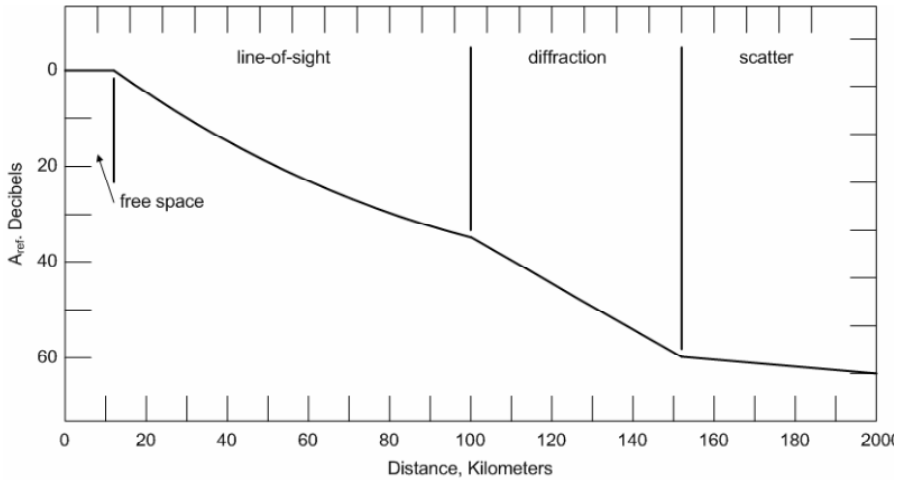



Figura 2.11 - Atenuação de referência [16].

As perdas totais pertencentes a um determinado percurso são dadas pelo somatório entre a perda do espaço livre, $L_{FS}$, e a atenuação de referência. A perda em espaço livre é calculada pela equação (2.2) enquanto o cálculo das restantes variáveis pode ser efetuado com base no algoritmo simplificado proposto por *Huffor* [16], na qual está descrito em detalhe a fundamentação matemática de cada parâmetro.

## 2.4 Modelos para vegetação

Os modelos para meios com vegetação dão um valor aproximado da atenuação extra em relação ao espaço livre para ambientes com folhagem ou troncos no percurso de propagação entre o transmissor e o recetor.

### 2.4.1 Modelo de Weissberger

O modelo de Weissberger [5] é um modelo de propagação que estima a atenuação devido à propagação de uma ou mais árvores, numa transmissão ponto-a-ponto. Este modelo é válido para frequências dos 230 MHz a 95 GHz e densidade de folhagem até 400 m.

O modelo é apenas aplicável quando existe um obstáculo composto por folhagem ou vegetação entre os pontos de comunicação. É, idealmente, aplicado em situações em que a linha de vista está bloqueada por espaços densos de floresta, composta por árvores com elevada folhagem ou vegetação seca.

O modelo de Weissberger é representado por

$$PL_{weissberger}(\text{dB}) = \begin{cases} 1{,}33\, f^{0{,}284}\, d^{0{,}588}, & \text{se } 14 < d \leq 400 \\ 0{,}45\, f^{0{,}284}\, d, & \text{se } 0 < d \leq 14 \end{cases} \quad (2.16)$$

em que $d$ (m) representa a distância de folhagem ao longo do percurso de transmissão e a frequência ($f$) é dada e GHz [14].

### 2.4.2 Modelo ITU

Tomando em consideração as recomendações descritas no ITU-R, em [14] desenvolveram o modelo FITU-R utilizando dados medidos nas frequências de 11,2 GHz e 20 GHz [17]. É representado por:

$$PL_{FITU}(\text{dB}) = \begin{cases} 0{,}39\, f^{0{,}39}\, d^{0{,}25} & \text{(com folhas)} \\ 0{,}37 f^{0{,}18}\, d^{0{,}59} & \text{(sem folhas)} \end{cases} \tag{2.17}$$

Meng, Lee e Ng [17] otimizaram estas equações nas bandas de VHF e UHF, com dados obtidos a 240 MHz e 700 MHz, num meio com vegetação pouco densa. A grande diferença é que este modelo toma em consideração a componente lateral, onde o sinal segue ao topo das árvores.

$$PL_{LITU}(\text{dB}) = 0{,}48\, f^{0{,}43}\, d^{0{,}13} \tag{2.18}$$

É de realçar que estes modelos apenas dependem da frequência e da distância e não de outros parâmetros importantes, quando se analisa propagação num ambiente de floresta.

### 2.4.3 Modelo COST 235

O modelo COST 235 [6] considera duas situações distintas. A primeira leva em consideração se as árvores estão com folhas e a segunda nos momentos em que a folhagem cai, normalmente no outono, em árvores caducas, sendo a atenuação em dB dada por:

$$PL_{COST\,235}(\text{dB}) = \begin{cases} 15{,}6\, f^{-0{,}009}\, d^{0{,}26} & \text{(com folhas)} \\ 26{,}6\, f^{-0{,}2}\, d^{0{,}5} & \text{(sem folhas)} \end{cases} \tag{2.19}$$

com a frequência $f$ apresentada em MHz.

### 2.3.4 Modelo Log-Normal

O modelo Log-Normal [8] é bastante usado para descrever o desvanecimento lento. As perdas de percurso são dadas por:

$$PL_{LN}\,(\text{dB}) = PL\,(d_0) + \,10\, n\, \log_{10}\left(\frac{d}{d_0}\right) + \, X_\sigma \tag{2.20}$$

O parâmetro $n$ é dado pelo indicador de perdas de percurso, que refere o decaimento do sinal em relação à distância. Considerando-se $n$ = 2 tem-se o decaimento do espaço livre. $PL\,(d_0)$ refere-se à atenuação a uma distância de referência $d_0$, mas no campo

distante. $X_\sigma$ é uma variável aleatória gaussiana, em dB, com desvio padrão $\sigma$ e reflete a variabilidade da potência recebida em relação a sua média. As três variáveis da equação (2.20) podem ser utilizadas para descrever a atenuação devido à perda de percurso num determinado meio.

### 2.4.5 Modelo de Azevedo e Santos

O modelo apresentado em [9] toma em consideração os parâmetros físicos da floresta, relacionando as perdas com a densidade de vegetação. Baseia-se em medições extensivas realizadas nas florestas da lha da Madeira, nas bandas de 870 MHz e 2,414 GHz com antenas a uma altura de 3 m acima do solo.

O modelo baseia-se no modelo da log-normal e permite prever a atenuação em função da distância (até 400 m) e da frequência (gama) em função da densidade de vegetação. Esta é definida por:

$$VD = TD * D \tag{2.21}$$

com $TD$ o número de árvores por m$^2$ e $D$ é o diâmetro dos troncos em cm. Os parâmetro da log-normal são definidos por:

$$n = 0{,}043\ (d_m - 40)^{0,47}\ VD + \left(0{,}54 f^{0,42} - 0{,}45\right) \times (d_m - 40)^{-0,15} + 2 \tag{2.22}$$

$$PL_{AS}(d_0) = \left(-0{,}026\ d_m + 0.49 f^{0,47}\right)\ VD + 20\ \log_{10}\left(\frac{c}{4\pi f}\right) \tag{2.23}$$

Para distâncias inferiores a 60 m utiliza-se $d_m$= 60, exceto na área junto ao emissor, em que se utiliza o modelo de espaço livre. A frequência é dada em GHz e $d_0$= 1 m.

## 2.5 Modelos devido a condições atmosféricas

A atmosfera é composta por vários gases, que absorvem a energia eletromagnética, independentemente da sua frequência. Pode-se considerar a atmosfera como sendo organizada em várias camadas horizontais, a diferentes altitudes, cada uma constituída por vapor de água e oxigénio, com densidades distintas [3]. A atenuação atmosférica depende da pressão, da temperatura e do vapor de água. Devido a estes fatores, a atenuação pode variar com a posição geográfica, altitude e o percurso de transmissão.

A presença de humidade na atmosfera pode tomar formas distintas, que por sua vez, também tem influência na atenuação na propagação do sinal. Estes efeitos estão obviamente correlacionados com a frequência.

### 2.5.1 Atenuação por oxigénio e vapor de água

A atenuação suplementar devido ao vapor de água e densidade de oxigénio é mínima para frequências de 1 a 10 GHz [3] [18].

Uma forma de expressar o valor da atenuação suplementar é através de:

$$L\,(\mathrm{dB}) = (\gamma_{o0} + \gamma_{w0})d \tag{2.24}$$

em que $\gamma_o$ representa o coeficiente de atenuação devido ao oxigénio e $\gamma_w$ é o coeficiente de atenuação do vapor de água medidos em dB/km em ligações terrestres, uma vez que não existe variação apreciável ao longo do percurso.

As atenuações específicas do vapor de água $\gamma_w$ e do ar seco $\gamma_o$ em dB/Km, a partir do nível do mar até altitudes até 5 Km, podem ser calculadas com uma precisão de $\mp 15\%$. Para frequências inferiores ou iguais a 57 GHz $\gamma_o$ são dados através de:

$$\gamma_0 = \left[\frac{7{,}27 r_t}{f^2 + 0{,}351 r_p{}^2 r_t{}^2} + \frac{7{,}5}{(f-57)^2 + 2{,}44 r_p{}^2 r_t{}^5}\right] f^2 r_p{}^2 r_t{}^2 \times 10^{-3} \tag{2.25}$$

com a frequência $f$ em GHz, $r_p$ dado por $p/1012$ sendo $p$ a pressão atmosférica em $hPa$ e $r_t$ é dado por 288/(273+ $T$) com $T$ a temperatura em graus centígrados.

A atenuação $\gamma_w$ por vapor de água é expressa por:

$$\begin{aligned}\gamma_w = \Bigg[&3{,}27 \times 10^{-2} r_t + 1{,}67 \times 10^{-3} \rho \frac{r_t{}^7}{r_p} + 7{,}7 \times 10^{-4} f^{0{,}5} \\ &+ \frac{3{,}79}{(f-22{,}235)^2 + 9{,}81 r_p{}^2 r_t} + \frac{11{,}73 r_t}{(f-183{,}31)^2 + 11{,}85 r_p{}^2 r_t} \\ &+ \frac{4{,}01 r_t}{(f-325{,}153)^2 + 10{,}44 r_p{}^2 r_t}\Bigg] f^2 \rho r_p r_t \times 10^{-4}\end{aligned} \tag{2.26}$$

em que $\rho$ é a quantidade de vapor de água em $\mathrm{g/m^3}$. O valor de $\rho$ não pode ultrapassar o valor da saturação à temperatura considerada.

Os valores da atenuação calculam-se com recurso às curvas presentes na Figura 2.12, que representam a atenuação específica dos gases atmosféricos para uma pressão atmosférica de 1013 hPa, a uma temperatura de 15 ºC e uma concentração de vapor de água de 7,5 $\mathrm{g/m^3}$ [3] [18].

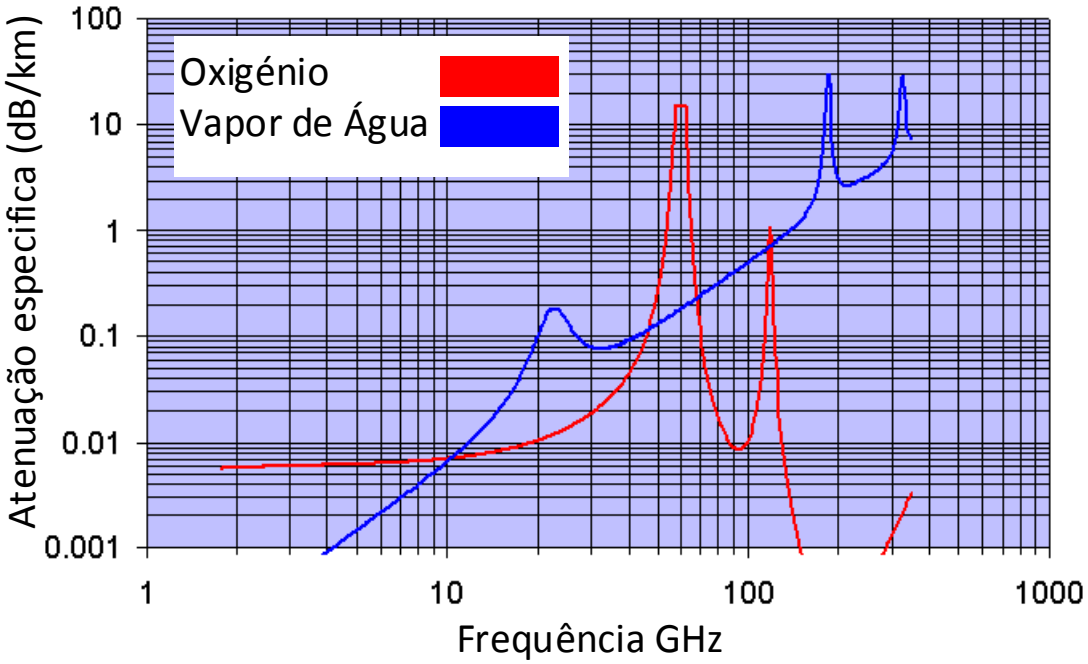



Figura 2.12 - Atenuação específica devida aos gases atmosféricos [19].

### 2.5.2 Modelo ITU para nevoeiro e nebulosidade

O ITU prevê um modelo para nevoeiro e atenuação por nebulosidade que é válido até aos 200 GHz [20] [21]. A atenuação específica é dada por:

$$\gamma_c = K_l M \qquad \text{dB/km} \tag{2.27}$$

onde $\gamma_c$ representa a atenuação específica na nuvem, $K_l$ é o coeficiente de atenuação específica em (dB/km)/(g/m$^3$) e $M$ representa a densidade de água líquida, na nuvem, em g/m$^3$.

Para frequências na ordem dos 100 GHz e acima, a atenuação devido ao nevoeiro, poderá ser significativa. Tipicamente, $M$ = 0,05 g/m$^3$ para nevoeiro com média densidade (300 m de visibilidade) e $M$ = 0,5 g/m$^3$ para nevoeiro com alta densidade (50 m de visibilidade) [14].

A Figura 2.13 representa o produto entre a atenuação total devido às nuvens e a frequência, para vários ângulos de elevação usando o pior caso possível a nível de densidade.

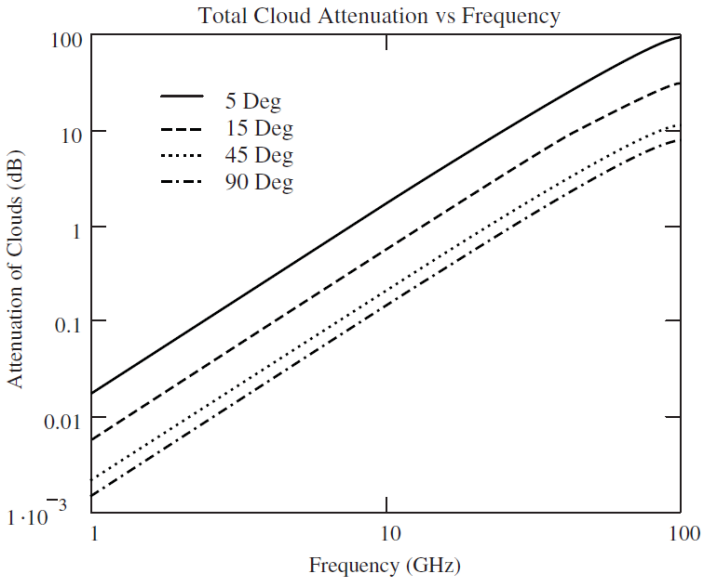



Figura 2.13 - Produto em a atenuação e a frequência [14].

### 2.5.3 Atenuação devido à precipitação

A precipitação tem influência na atenuação global do sistema de ligação [3]. O desvanecimento devido à chuva depende da taxa de pluviosidade, tamanho, forma da gota, densidade e volume. Destes fatores, apenas a taxa de pluviosidade é facilmente medida, sendo então o parâmetro mais utilizado para caracterizar o desvanecimento devido a chuva.

Um dos modelos mais utilizados é a recomendação ITU-R P.838-3 [22]. É calculado a partir de dados do ITU, sendo a atenuação específica determinada utilizando coeficientes de regressão e taxas de crescimento de chuva.

A interpolação dos coeficientes é efetuada utilizando-se uma escala logarítmica de frequência para os ângulos de elevação, $k$, e uma escala linear para a polarização, $\alpha$. Os coeficientes interpolados para frequências de 1 a 1000 GHz podem ser consultados em [22]. Estes valores são dependentes da frequência e da polarização. O ângulo de elevação do percurso de propagação é dado por:

$$k = \frac{\left[\ \mathrm{k_H} + \mathrm{k_V} + (\mathrm{k_H} - \mathrm{k_V})\, cos^2(\theta)\ cos(2\tau)\ \right]}{2} \tag{2.28}$$

enquanto a polarização é dada por:

$$\alpha = \frac{\left[\ k_H\alpha_H + k_V\alpha_V + (k_H\alpha_H - k_V\alpha_V)\, cos^2(\theta)\ cos(2\tau)\ \right]}{2k} \tag{2.29}$$

em que $\theta$ é o ângulo de elevação do percurso de propagação e $\tau$ é o ângulo de polarização (0° - Horizontal; 45° - Circular;  90° vertical).

Na Tabela 2.1 encontram-se os valores da percentagem de tempo no ano em que o valor da intensidade de precipitação é excedido no território Português, em que o arquipélago da Madeira insere-se na zona K [23].

Tabela 2.1 - Intensidade de precipitação (mm/h)

| **Intensidade da Precipitação ($mm/h$)** | | **Percentagem de tempo (%)** |
|---|---|---|
| **Zona H** | **Zona K** | |
| 2 | 1,5 | 1 |
| 4 | 4,2 | 0,3 |
| 10 | 12 | 0,1 |
| 18 | 23 | 0,03 |
| 32 | 42 | 0,01 |
| 55 | 70 | 0,003 |
| 83 | 100 | 0,001 |

O método de cálculo da atenuação devido a chuva, seguindo a Recomendação P.530-7 da ITU-R [24], consiste em primeiro lugar em obter a intensidade de precipitação excedida durante o pior mês, utilizando a Tabela 2.1, para o caso do arquipélago da Madeira. O segundo passo é calcular o coeficiente de atenuação, dado por:

$$\gamma_R = kR^{\alpha} \qquad \text{(dB/km)} \tag{2.30}$$

Finalmente, a atenuação excedida pelo percurso é dado pela multiplicação pela distância eficaz do percurso.

$$d_{ef} = \frac{d}{1 + \frac{d}{35e^{-0{,}015\, R_{i_{0,01}}}}} \tag{2.31}$$

## 2.6 Feixes hertzianos

Os feixes hertzianos são tipicamente ligações ponto a ponto na zona das micro-ondas. A designação deve-se ao trabalho de Hertz, que passou da teoria à prática e consegui-o comprovar o trabalho de Maxwell gerando ondas rádio em ambiente de laboratório [25].

A propagação faz-se normalmente em linha de vista, com saltos máximos de, aproximadamente, 50 km. O alcance encontra-se limitado pela curvatura da Terra. Para ligações longas ou obstruídas pela orografia do percurso, torna-se necessário o uso de repetidores. Utiliza-se este modo de propagação geralmente nas comunicações a longa distância, recorrendo-se a portadoras com frequência elevadas (2 a 40 GHz), utilizando antenas bastante diretivas (parabólicas), confinando a maior parte da energia transmitida

a um feixe. No entanto, o feixe hertziano requer um alinhamento cuidadoso das antenas [3].

### 2.6.1 Tipos de atenuação

Quando se refere a feixes hertzianos, tem-se que ter em conta os diversos tipos de atenuação, que podem prejudicar a qualidade do sinal. A Figura 2.14 apresenta os diferentes tipos de atenuação, bem como a sua influência numa ligação por feixes hertzianos, tendo sempre como referência a Recomendação P.341-5 da ITU-R [26] .

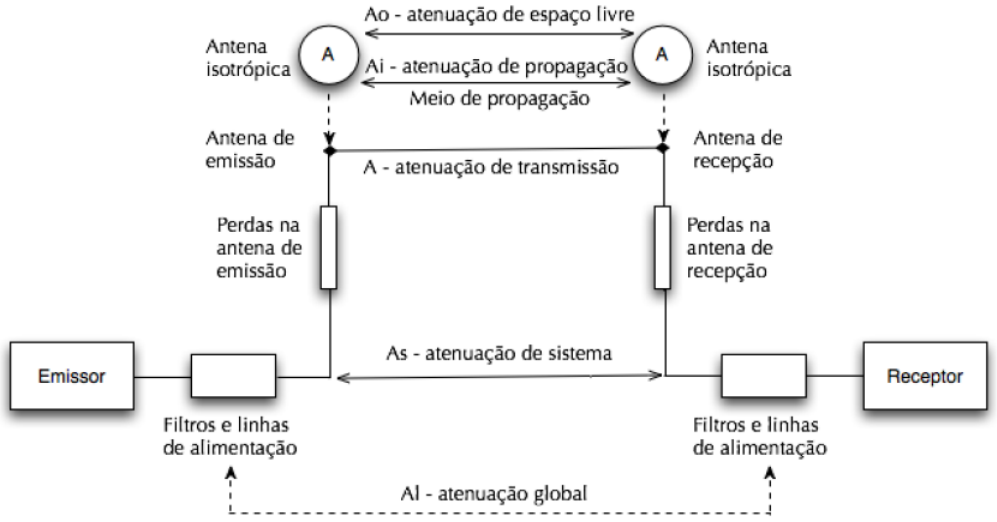



Figura 2.14 - Distinção dos diferentes tipos de atenuação [3].

A atenuação em espaço livre ocorre num meio dielétrico limitado e homogéneo. A atenuação de propagação refere-se a atenuação entre antenas, caso estas fossem substituídas por antenas isotrópicas. A relação entre a potência radiada pela antena de emissão e a potência disponível na antena recetora refere-se à atenuação de transmissão. A atenuação de sistema é a atenuação entre os terminais das antenas, incluindo todas as perdas dielétricas, circuitos de adaptação e resistências terminais. Finalmente, a atenuação global diz respeito à relação direta da potência fornecida pelo emissor e a potência recebida pelo recetor, em condições reais, ou seja, reúne todas as anteriores atenuações [3].

### 2.6.2 Desvanecimento

O desvanecimento ocorre devido às características do meio de propagação serem variáveis no tempo, fazendo com que a potência recebida no recetor oscile com o tempo, mesmo quando a potência transmitida é constante. O desvanecimento por multipercurso acontece quando existe mais do que um percurso distinto entre duas antenas terminais. A antena recetora recebe interferências, com atenuações semelhantes, dos vários percursos com fases relativas que dependem das diferenças dos caminhos percorridos. Podem existir, também, reflexões nas camadas da atmosfera mais próximas do terreno, como é o exemplo da neblina ou do nevoeiro em vales com muita humidade. No entanto, estas

condições só ocorrem em alturas sem vento e durante a madrugada ou nas primeiras horas da manhã [3].

A Recomendação P.530-14 do ITU-R [7] sugere um modelo para o cálculo dos efeitos multipercurso na atmosfera. Este divide-se em dois, um referente ao desvanecimento para uma qualquer probabilidade, descrito em pormenor em [7], e um outro para o caso do desvanecimento de baixa probabilidade. Neste método determina-se o fator geoclimático $K$, para o mês médio mais desfavorável para percursos terrestres, sendo definido através de:

$$K = 10^{-4,4-0,0027\mathrm{d}N_1}(1 + S_a)^{-0,46} \tag{2.32}$$

em que $\mathrm{d}N_1$, é o gradiente do índice de refração pontual que não excede 1% da média anual disponível na Recomendação ITU-R P.453. e $S_a$ refere-se a rugosidade do terreno.

Pode-se aproximar o valor de $K$ por:

$$K = 10^{-4,6-0,0027\mathrm{d}N_1} \tag{2.33}$$

Determinado o gradiente de refratividade da zona geográfica em estudo, calcula-se a amplitude da inclinação do percurso através de:

$$|\varepsilon_p| = \frac{|h_{Rx} - h_{Tx}|}{d} \tag{2.34}$$

sendo $h_{Rx}$ e $h_{Tx}$ as alturas acima do nível médio do mar, em metros, das antenas emissoras e recetoras, respetivamente, e $d$ a distância em km entre as antenas.

A probabilidade da potência recebida $p$ ser inferior ou igual a $p_0$, à frequência $f$ em GHz, com $h_L$ sendo a altura da antena mais baixa em metros, é dada por:

$$P(p \le p_0) = Kd^{3,4}(1 + |\varepsilon_p|)^{-1,03} f^{0,8} \times 10^{-0,00076h_L} \frac{p_0}{p_n} \tag{2.35}$$

ou usando a aproximação:

$$P(p \le p_0) = Kd^{3,1}(1 + |\varepsilon_p|)^{-1,29} f^{0,8} \times 10^{-0,00089h_L} \frac{p_0}{p_n} \tag{2.36}$$

O limite inferior da gama de frequências é calculado por:

$$f(\mathrm{GHz}) = \frac{1,5}{d} \tag{2.37}$$

O segundo método pode ser consultado detalhadamente na Recomendação P.530-14 da ITU-R [7]. No entanto, para as condições normais da Europa Ocidental é possível estimar, de forma grosseira, o desvanecimento não excedido durante mais de 80% do tempo, no pior mês por:

$$F_{80}(\mathrm{dB}) = 0,03\sqrt{f}_{[\mathrm{GHz}]} \tag{2.38}$$

A Figura 2.15 representa o desvanecimento dado pela equação (2.38) em função do comprimento do percurso, para vários valores de frequência.

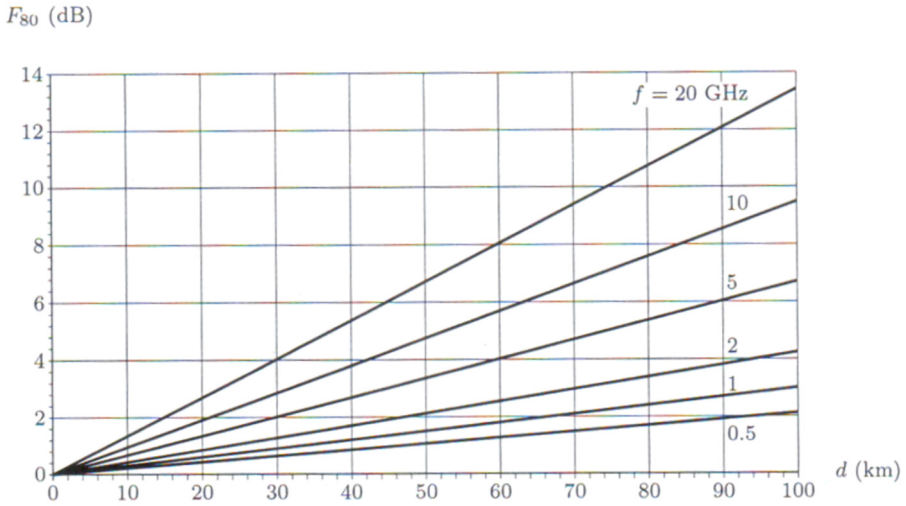



Figura 2.15 - Desvanecimento não excedido durante 80% do tempo em função da distância [3].

## 2.7 Projeto de ligação por feixes hertzianos

O projeto de uma ligação entre dois pontos engloba a seleção do percurso, a gama de frequências de trabalho e a escolha dos componentes principais, necessários para a transmissão de um ou mais sinais com qualidade preestabelecida e ao mais baixo custo. Em qualquer projeto é fundamental existir um equilíbrio entre o custo de estabelecimento da ligação, exploração e conservação. O projetista deve estar familiarizado com os detalhes da montagem, exploração e fazer uso de bom senso [3].

### 2.7.1 Escolha do percurso

A escolha do percurso torna-se num elemento fundamental para uma ligação por feixes hertzianos. Com os meios tradicionais, a escolha do percurso é feita por tentativas, sendo efetuado um estudo relacionando os custos com a fiabilidade de propagação [3].

O primeiro passo consiste em determinar a localização entre os quais se pretende estabelecer a ligação, fazendo uso de cartas militares ou *software* próprio. Para percursos da ordem dos 50 a 60 km normalmente usa-se uma carta à escala de 1:25000, com curvas de nível espaçadas de 10 m. Para ligações mais longas ou com vários saltos, é usual utilizar cartas com escala até 1:250000 [3] [27].

Identificados os locais para o emissor e recetor, desenha-se sobre a carta um arco de círculo máximo que os une, que representa uma aproximação à ligação pretendida. É efetuada uma leitura do valor dos pontos máximos relativos. Se numa primeira observação, este percurso encontra-se desobstruído ou for possível desobstruí-lo com recurso a mastros de antenas inferiores a 30 m, então o percurso pode ser estudado, caso contrário torna-se necessário um percurso alternativo [27].

### 2.7.2 Frequência

A instalação de sistemas de feixes hertzianos, fora das bandas livres, está dependente de licenciamento pela autoridade responsável pela gestão do espetro (ANACOM). A frequência de trabalho e a largura dos canais radioelétricos deve ser determinada de acordo com as recomendações aplicáveis pelo ITU-R, as condições de propagação e as restrições locais devido a sistemas já existentes ou futuros, de maneira a evitar interferências [3].

### 2.7.3 Antenas

Uma parte fundamental numa ligação por feixes hertzianos são as antenas, sendo escolhidos tendo em consideração a frequência e a finalidade do sistema de transmissão. Obviamente existem diversos tipos de antenas com as mais diversas aplicações. Para feixes hertzianos são normalmente utilizados dois tipos distintos de antenas, as antenas do tipo refletor parabólico e as antenas helicoidais ou Yagi-Uda. As antenas parabólicas são normalmente utilizadas para frequências superiores a 1 GHz, com tamanhos normalmente compreendidos entre os 0,5 a 4 metros de diâmetro. As antenas helicoidais ou Yagi-Uda são mais indicadas para frequências inferiores a 1 GHz [3] [25].

### 2.7.4 Cabos e guias de ondas

Numa instalação tem-se de tomar em consideração como é realizada a ligação entre a fonte emissora ou recetora e a antena. Esta ligação é calculada de forma específica de modo a conseguir suportar as altas frequências utilizadas no projeto deste tipo de ligações. O cabo coaxial é amplamente utilizado para frequências inferiores a 2 GHz. Para frequências mais elevadas pode-se recorrer a guias de onda metálicas. Se apenas for considerada a atenuação, normalmente utilizam-se guias de onda para frequências superiores a 2 GHz, uma vez que para frequências inferiores têm a secção transversal de dimensões elevadas o que o torna de difícil a sua aplicação. As guias de onda, normalmente são feitas de cobre e distinguem-se pela forma da secção, retangular, circular ou elíptica. Obviamente, cada tipo de secção tem características específicas que justifica a sua utilização [3].

### 2.7.5 Equipamento radioelétrico

O equipamento radioelétrico é tipicamente constituído por um emissor e um recetor, incluindo fontes de alimentação e oscilador local, modulador, desmodulador ou combinador de diversidade. São normalmente montados em estruturas metálicas, chamados bastidores.

A potência de emissão em sistemas de feixes hertzianos depende da frequência de trabalho mas é em geral, inferior a 10 W, podendo-se estimar por [3]:

$$P_E(W) \leq \frac{10}{f_{\,[\mathrm{GHz}\,]}} \tag{2.39}$$

### 2.7.6 Fornecimento de energia

Todos os sistemas de telecomunicações devem garantir uma elevada fiabilidade. Para esse efeito, torna-se imprescindível que os períodos de indisponibilidades devido a propagação, a avarias de equipamentos, a erros humanos, sejam baixos, mas também que o fornecimento de energia elétrica tenha uma fiabilidade e redundância elevada. Um dos maiores problemas é que muitas estações de feixes hertzianos, em particular as estações repetidoras, estão normalmente instaladas em locais remotos, de difícil acesso e não estão servidas pela rede pública de distribuição de energia elétrica [3].

Para consumos inferiores a 1000 W, na estação, pode-se considerar a energia proveniente de fontes renováveis como painéis fotovoltaicos ou geradores eólicos.

### 2.7.7 Cálculo de uma ligação

Apresenta-se de seguida, a título de exemplo, os cálculos de uma ligação por feixes hertzianos digitais. Na Figura 2.16 apresenta-se uma representação gráfica do cálculo de uma ligação por feixes hertzianos.

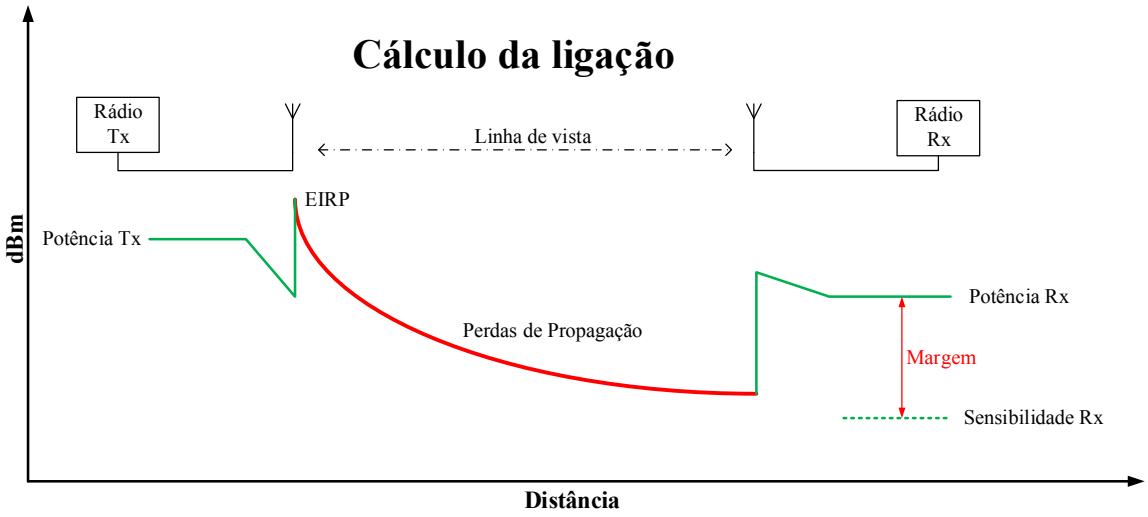



Figura 2.16 - Cálculo da ligação por feixes hertzianos.

Para a estimação da margem de segurança torna-se necessário em primeiro lugar calcular o sinal recebido. Para efetuar esse cálculo é necessário conhecer os valores da potência de transmissão, a sensibilidade do sistema e o valor dos ganhos das antenas do sistema de comunicação. Estes valores, normalmente, encontram-se documentados nas folhas de características dos rádios e das antenas de emissão e receção. Caso os parâmetros das antenas e rádios sejam iguais espera-se um valor de potência recebida igual nos dois sentidos. No entanto, caso exista rádios ou antenas diferentes, tem-se que efetuar a analise em ambas as direções.

O primeiro passo no cálculo de uma ligação é verificar-se se existem interferências no primeiro elipsoide de Fresnel. Caso não existam utiliza-se a estimação das perdas de

propagação em espaço livre, utilizando-se a equação (2.2), sendo, para tal, necessário saber-se a frequência de operação e a distância de transmissão. Este cálculo permite saber se a ligação é possível, uma vez que, caso não seja viável em espaço livre, nunca será possível na realidade, pois à atenuação provocada pelo espaço livre é depois somada as perdas nos cabos e nos conetores, as perdas de propagação e outros fenómenos que podem diminuir a quantidade de sinal que chega ao recetor.

O cálculo seguinte passa por descobrir o EIRP (*Effective Isotropically Radiated Power*) do sistema, que no caso de Portugal encontra-se regulamentado pela ANACOM, e é dado por:

$$EIRP = P_{Tx} - A_{Tx} + G_{Tx} \tag{2.40}$$

A potência de transmissão ($P_{Tx}$) é normalmente dada em dBm, a atenuação nos cabos da antena transmissora ($A_{Tx}$) em dB e o ganho da antena de transmissão ($G_{Tx}$) em dBi.

A potência recebida ($P_{Rx}$) em dB é dada por:

$$P_{Rx} = EIRP + G_{Rx} - A_{Rx} - PL_{Fs} \tag{2.41}$$

com o ganho da antena recetora ($G_{Rx}$) em dBi, a atenuação nos conetores e cabos da antena recetora ($A_{Rx}$) dado em dB e as perdas em espaço livre ($PL_{Fs}$) dadas em dB.

A sensibilidade é o valor mínimo que ainda garante o objetivo de qualidade preestabelecida. Abaixo deste valor a operação do feixe fica degradada, podendo, no limite, ficar fora de serviço.

Para existir uma ligação estável o sistema de comunicação por feixes hertzianos deve considerar outras causas de atenuação, ou condições metrológicas adversas. Deve-se por isso considerar uma margem de ligação dada por:

$$Margem = P_{Rx} - Sensibilidade\ R_x \tag{2.42}$$

## 2.8 Conclusões do capítulo

Neste capítulo estudou-se a fundamentação teórica que serve de base ao trabalho desenvolvido. Começou-se por introduzir os sistemas de rádio frequência e feixes hertzianos.

Depois, foram abordados os modelos de perdas de propagação, dando ênfase aos modelos em terrenos irregulares, nomeadamente os modelos, de Espaço livre, Obstáculo em lâmina, Epstein-Petterson, Deygout, ITU-R P530-14, Egli e o modelo de Longley-Rice.

Na continuação do trabalho, estudaram-se os modelos utilizados na previsão das perdas de propagação em vegetação, em particular os modelos de Weissberger, ITU, COST 235, Log-Normal e o modelo de Amândio & Santos.

Abordou-se, também os modelos devido a condições atmosféricas, sobretudo os modelos de atenuação devido ao oxigénio e vapor de água, modelo ITU para o nevoeiro e atenuação provocada pela precipitação.

Numa segunda fase do estudo optou-se por dar ênfase ao projeto dos feixes hertzianos, referindo os tipos de atenuação e o desvanecimento. Abordou-se os componentes e preocupações a considerar quando se projeta uma ligação e finalmente, apresentou-se os cálculos teóricos em relação ao cálculo da potência recebida, e margem de segurança da ligação.

# 3. Implementação dos modelos e Software existente

A simulação de uma ligação de rádio frequência e a respetiva cobertura da estação base podem ser efetuadas por diversos tipos de programas. A simulação permite reduzir o tempo despendido na análise do percurso de propagação, uma vez que um percurso inexequível em simulação será descartado da sua averiguação no terreno.

Neste capítulo descrevem-se os *softwares* de propagação gratuitos, avaliados para este trabalho, e o programa de simulação desenvolvido em plataforma MATLAB, para planeamento de uma ligação rádio frequência.

## 3.1 AirLink

O *AirLink* é uma ferramenta de planeamento de ligações RF, desenvolvido pela UBNT (*Ubiquiti Networks*) [28], que permite realizar uma simulação de uma ligação ponto-a-ponto, analisando as perdas por percurso do trajeto escolhido. O *software* contém um *plug-in* do *Google Earth* que, com a introdução das coordenadas geográficas de latitude e longitude juntamente com os parâmetros das antenas, permite simular o percurso do sinal.

A escolha deste *software* deve-se ao facto do fabricante das antenas utilizadas no projeto (antenas de feixes hertzianos) o disponibilizar gratuitamente. Atualmente, trata-se de uma versão beta e pode ser consultada gratuitamente no *site* [29]. O programa permite uma gama de frequências dos 900 MHz até aos 24 GHz, mas apenas admite o uso de rádios fabricados pela *Ubiquiti Networks*. O modelo de propagação que serve de base é o modelo de Longley-Rice, apresentado no capítulo 2, secção 2.3.7.

Na Figura 3.1 tem-se uma representação gráfica da interface principal, juntamente com os separadores de localização e fatores ambientais.

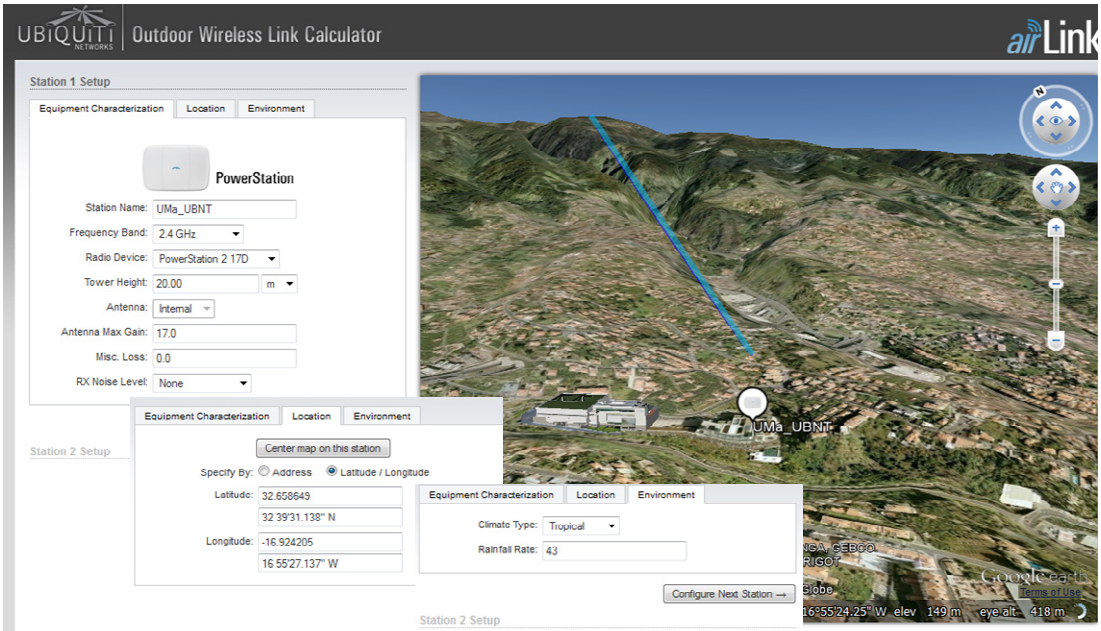



Figura 3.1 - Janela principal do programa airLink.

No separador principal tem-se os campos referentes à escolha do tipo de antenas, à frequência de operação e as características principais do rádio escolhido. No campo de localização pode-se optar por introduzir as coordenadas ou introduzir a morada, após o qual o *Google Earth* apresenta a localização da antena emissora, que pode ser arrastada para um ajuste mais fino (o processo é idêntico para a antena recetora). Finalmente o separador de fatores ambientais permite selecionar o tipo de clima e quantidade de precipitação em mm/h.

A Figura 3.2 apresenta um exemplo de cálculo de uma ligação, podendo-se visualizar o percurso de propagação, o perfil do terreno e os valores da potência e qualidade do sinal. O programa simula a taxa de transmissão juntamente com uma representação gráfica dos obstáculos presentes no percurso.

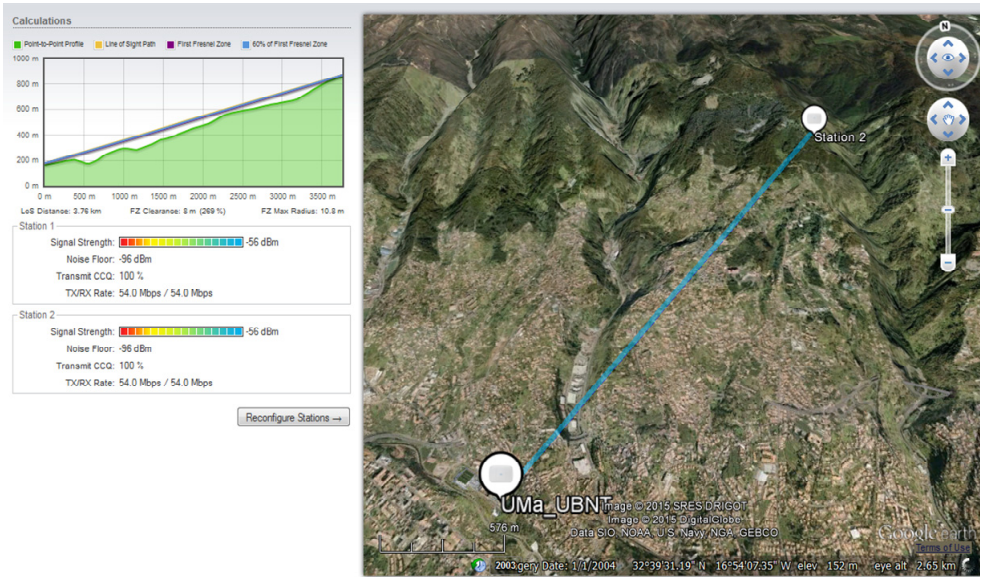



Figura 3.2 - Perdas de propagação com o *software AirLink*.

O *software* apresentado tem como vantagens ser simples e visualmente atraente, indicado para utilizadores inexperientes, que apenas pretendem saber qual o rádio mais indicado para o funcionamento do sistema de comunicação. Como desvantagens mais relevantes, pode-se referir a necessidade do uso contínuo de internet juntamente com a impossibilidade de escolher rádios de outros fabricantes e não contemplar os limites de potência impostos pelos diversos países.

## 3.2 RADIO MOBILE

O Radio Mobile é um *software* de simulação de propagação de rádio frequência, com distribuição gratuita, desenvolvido por *Roger Coudé* [30]. Baseia-se no modelo de Longley-Rice, apresentado no capítulo 2, secção 2.2.7, e trabalha na gama de frequências entre os 20 MHz e 20 GHz. Encontra-se disponível para *download* em [31], onde também é fornecido um manual de instalação e um guia com um exemplo de simulação.

O programa fornece diversas opções em detalhe para ligações ponto a ponto, facultando, inclusivamente níveis de sinal esperado em qualquer ponto do percurso, perdas por difração em obstáculos, interferência entre estações etc. O Radio Mobile

constrói automaticamente um percurso entre os pontos de emissão e receção no mapa digital, mostrando a zona de Fresnel, alturas requeridas pelas antenas e tolerância devido à curvatura da Terra, entre outros parâmetros de interesse. Na Figura 3.3 apresenta-se uma das interfaces de "*Radio Link"* utilizada onde se pode observar as diversas perdas de propagação.

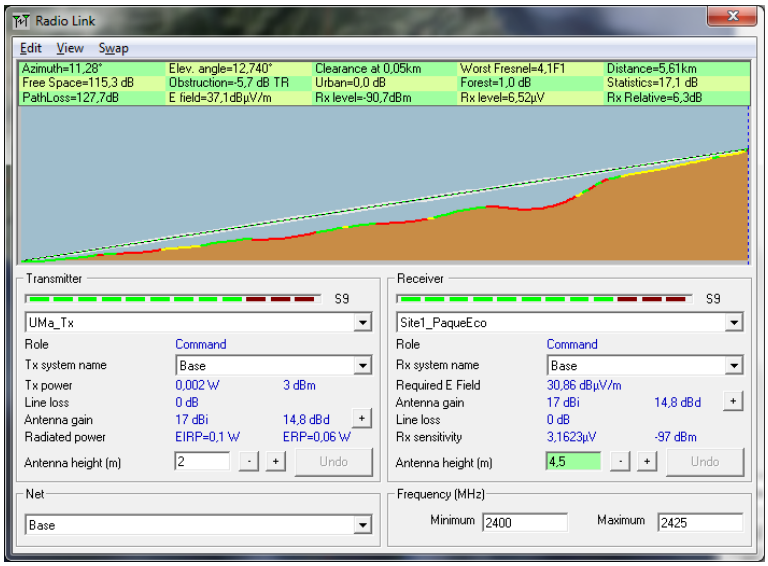



Figura 3.3 - Percurso de propagação Radio Mobile.

O *software* utiliza mapas de elevação digitais, SRTM (*Space Shuttle Radar Terrain Mapping Mission*), que podem ser adquiridos através de *download* no site da USGS (*U. S. Geological Survey)* [32]. Após o seu *download* não é necessário um acesso à internet. Juntamente com os mapas de elevação, o programa possibilita a integração de fotografias e diversos tipos de mapas, caso seja necessário.

Uma mais valia é que, apesar de ser fornecido com parâmetros genéricos de antenas, permite ao utilizador a introdução de antenas personalizadas, utilizando folhas de cálculo em Excel e depois alocar ao percurso de transmissão, avaliando o seu desempenho. Na Figura 3.4 apresenta-se um diagrama genérico de uma antena Yagi enquanto na Figura 3.5 representam-se os resultados para uma antena personalizada.

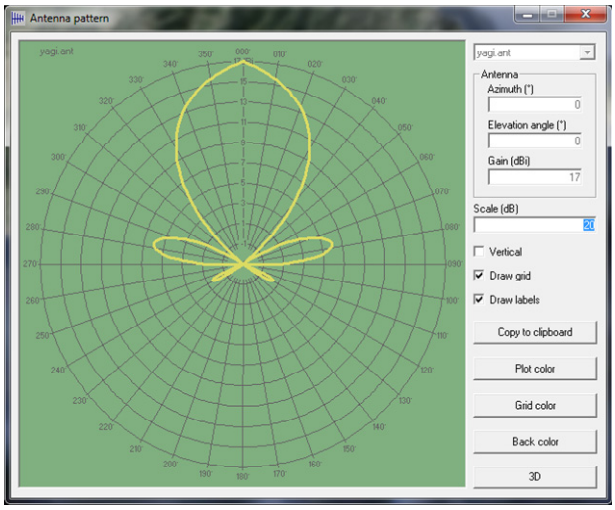
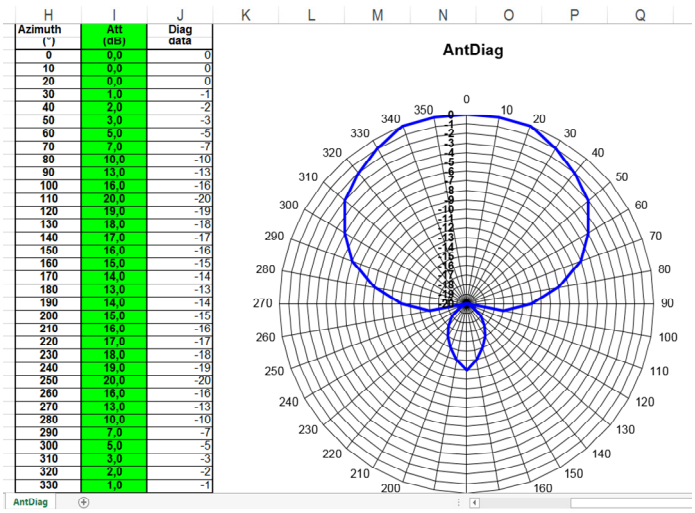



Figura 3.4 - Parâmetros da antena yagi no Rádio Mobile.




| Azimuth (°) | Att (dB) | Diag data |
|---|---|---|
| 0 | 0,0 | 0 |
| 10 | 0,0 | 0 |
| 20 | 0,0 | 0 |
| 30 | 1,0 | -1 |
| 40 | 2,0 | -2 |
| 50 | 3,0 | -3 |
| 60 | 5,0 | -5 |
| 70 | 7,0 | -7 |
| 80 | 10,0 | -10 |
| 90 | 13,0 | -13 |
| 100 | 16,0 | -16 |
| 110 | 20,0 | -20 |
| 120 | 19,0 | -19 |
| 130 | 18,0 | -18 |
| 140 | 17,0 | -17 |
| 150 | 16,0 | -16 |
| 160 | 15,0 | -15 |
| 170 | 14,0 | -14 |
| 180 | 13,0 | -13 |
| 190 | 14,0 | -14 |
| 200 | 15,0 | -15 |
| 210 | 16,0 | -16 |
| 220 | 17,0 | -17 |
| 230 | 18,0 | -18 |
| 240 | 19,0 | -19 |
| 250 | 20,0 | -20 |
| 260 | 16,0 | -16 |
| 270 | 13,0 | -13 |
| 280 | 10,0 | -10 |
| 290 | 7,0 | -7 |
| 300 | 5,0 | -5 |
| 310 | 3,0 | -3 |
| 320 | 2,0 | -2 |
| 330 | 1,0 | -1 |

Figura 3.5 - Resultados de uma antena diferente.

O *download* dos ficheiros Excel com representação em 2D ou 3D pode ser efetuado no *site* [33], juntamente com as instruções para a sua utilização.

O Rádio Mobile é um programa poderoso, mas apresenta algumas desvantagens. Requer um utilizador experiente e torna-se complexo quando se incorpora parâmetros mais avançados. Apesar de ser detalhado, apenas permite considerar as perdas de percurso em floresta como uma percentagem de atenuação suplementar. No entanto, pelas suas vantagens, tornou-se um dos programas mais conhecidos e utilizados para simulação de perdas de propagação, justificando a sua utilização neste trabalho.

## 3.3 MATLAB

Neste projeto escolheu-se o programa MATLAB [34] como plataforma base para o desenvolvimento de um *software* de simulação de perdas por propagação. O MATLAB é um programa poderoso e versátil, na área da engenharia, que organiza as funções por módulos. Este tipo de organização permite aos utilizadores o uso de funções desenvolvidas anteriormente ou publicadas no repositório do MATLAB, *Matlab Central Files* [35] de acordo com os seus requisitos.

Com o propósito de facilitar a introdução de dados e visualização de resultados desenvolveu-se uma interface gráfica, criada a partir da função *Guide* (GUI – *Guided User Interface*) do MATLAB. Este modo de programação é intuitivo e de simples implementação de *point-and-click*, ou seja, basta transportar o bloco de controlo desejado para a interface de trabalho e um duplo clique permite aceder e editar as suas propriedades, eliminando-se, assim, a necessidade de recorrer a um outro tipo de linguagem de programação. Após a gravação do ficheiro é criada automaticamente um ficheiro "nomegui.m" responsável pelas chamadas às funções *Callbacks*, algumas das quais permitem a edição de acordo com as necessidades do programador. Os controlos são modificados com recurso a apontadores *handles*, utilizando as funções de *get* no caso de se pretender adquirir o valor ou *set* quando se pretende fixar o valor.

### 3.3.1 Requisitos do programa

A transmissão de dados de uma aplicação remota requer, em primeiro lugar, a aquisição de dados e depois o envio dos mesmos para uma zona de recolha. Os meios de propagação não são sempre iguais, e mesmo dentro de meios análogos de transmissão existem sempre zonas de atenuação diferentes.

Neste projeto, além de existirem zonas de aquisição e envio de dados distintas, existem também tecnologias diferentes, rede de sensores sem fios e feixes hertzianos.

Nos programas estudados, averiguou-se existir uma lacuna no que diz respeito à análise detalhada, especialmente as perdas de propagação provocadas pela passagem em zonas de floresta ou perdas provocadas pela precipitação, oxigénio ou vapor de água. Verificou-se, então a necessidade de implementar um *software*, simples e visualmente atraente que possibilitasse o estudo das duas zonas, com foco na análise das atenuações apresentadas anteriormente, juntamente com a comparação entre os diversos modelos de propagação.

Definidos os principais objetivos, optou-se por desenvolver duas plataformas de cálculo de perdas por propagação. Numa primeira fase implementou-se um programa de comparação e visualização de curvas de atenuação devido à passagem por zonas de vegetação sem recurso ao *Guide*, fixando-se os campos variáveis, apenas trabalhando com valores fixos.

Na segunda fase do trabalho, optou-se pelo modo de programação GUI, e na parte final desenvolveu-se um programa de modulação das perdas de propagação referente aos modelos de propagação em terrenos irregulares, apresentados no capítulo 2 secção 2.3 juntamente com as perdas devido a condições

### 3.3.2 Estrutura e Organização

No Anexo II apresenta-se o código desenvolvido com o intuito de prever as perdas de propagação presentes numa transmissão de rádio frequência devido a passagem por zonas de vegetação. A lógica de funcionamento do programa percebe-se consultando o fluxograma da Figura 3.6.

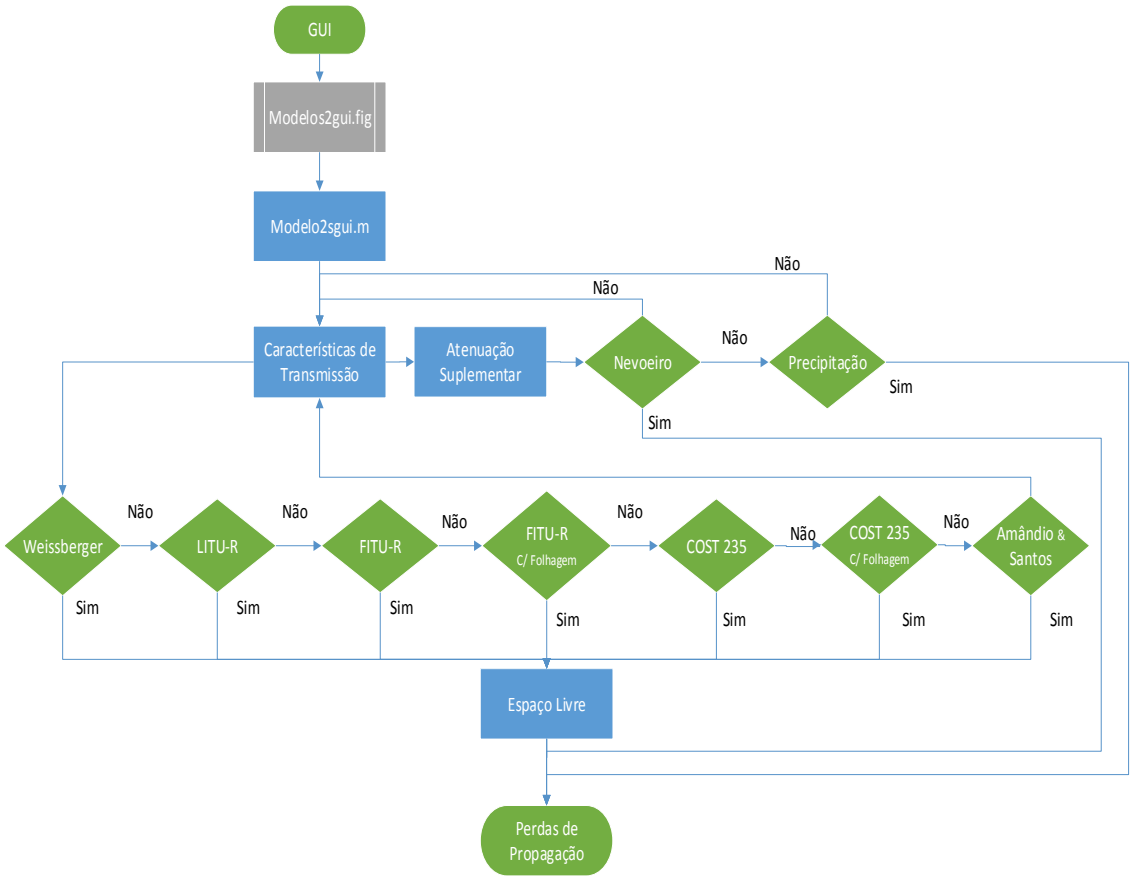



Figura 3.6 - Fluxograma do programa de perdas de propagação em vegetação.

O programa resultante apresenta uma interface gráfica que pode ser consultado na Figura 3.7.

Começa-se por introduzir as características de transmissão, nomeadamente a frequência de operação e a distância. Para a atenuação suplementar provocada por nevoeiro, caso seja selecionada, é necessário introduzir os valores de pressão atmosférica em hPa, temperatura em graus Celcius e a quantidade de vapor de água em $g/m^3$. Para a atenuação devido à precipitação introduzem-se os valores dos ângulos de elevação e polarização em graus e a taxa de precipitação em mm/h. Finalmente procede-se à escolha do modelo ou modelos a comparar.

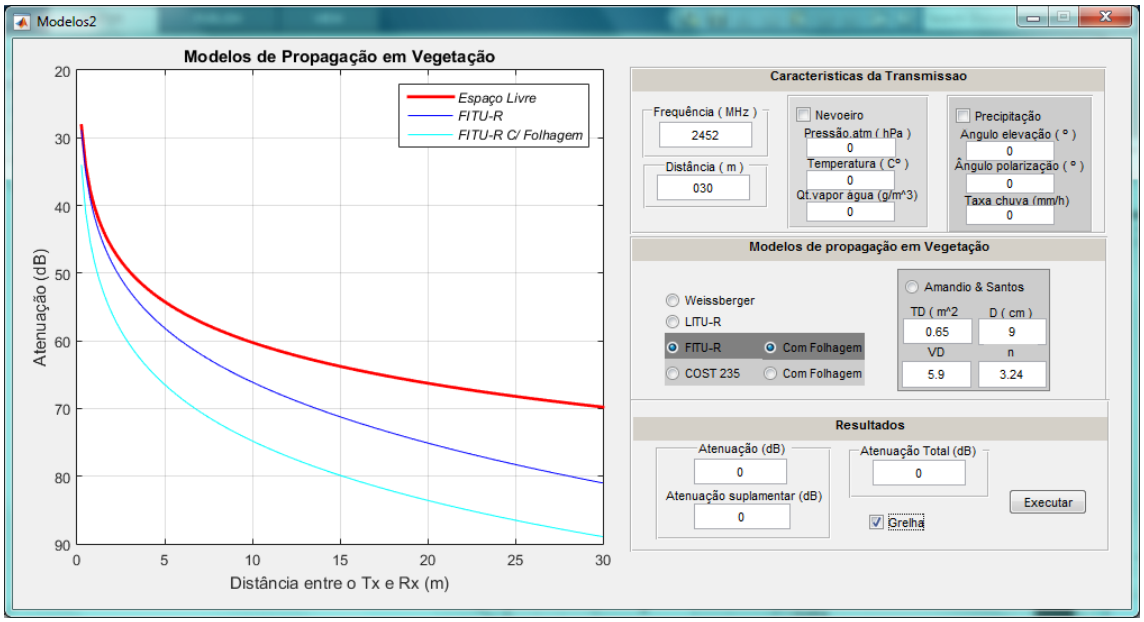



Figura 3.7 - Representação gráfica do programa de curvas de atenuação desenvolvido.

O modelo de Espaço Livre (a vermelho) serve como referência para os restantes modelos, sendo que, quando se seleciona um único modelo de propagação, o valor de atenuação é representado em dB no respetivo campo. O programa permite ainda selecionar uma grelha para melhor visualização das curvas de atenuação.

Ao selecionar-se a modelo de Azevedo & Santos, descrito no capítulo 2 secção 2.3.5, torna-se necessário também inserir os valores da densidade de árvores *TD* (árvores/m$^2$) e o diâmetro médio das mesmas *D* (cm) juntamente com as características de transmissão.

Na segunda fase optou-se pela programação da interface, responsável pela previsão das perdas de propagação, presentes numa transmissão de rádio frequência em linha de vista. Foi por isso, necessário incorporar o perfil do terreno, em simultâneo com os modelos de propagação apresentados no capítulo 2 secção 2.3 juntamente com as perdas devido a condições atmosféricas. Para tal, verificou-se a necessidade de integração no *software* de mapas detalhados.

O código principal pode ser consultado no Anexo III enquanto o fluxograma responsável pela dependência entre blocos pode ser observado na Figura 3.8.

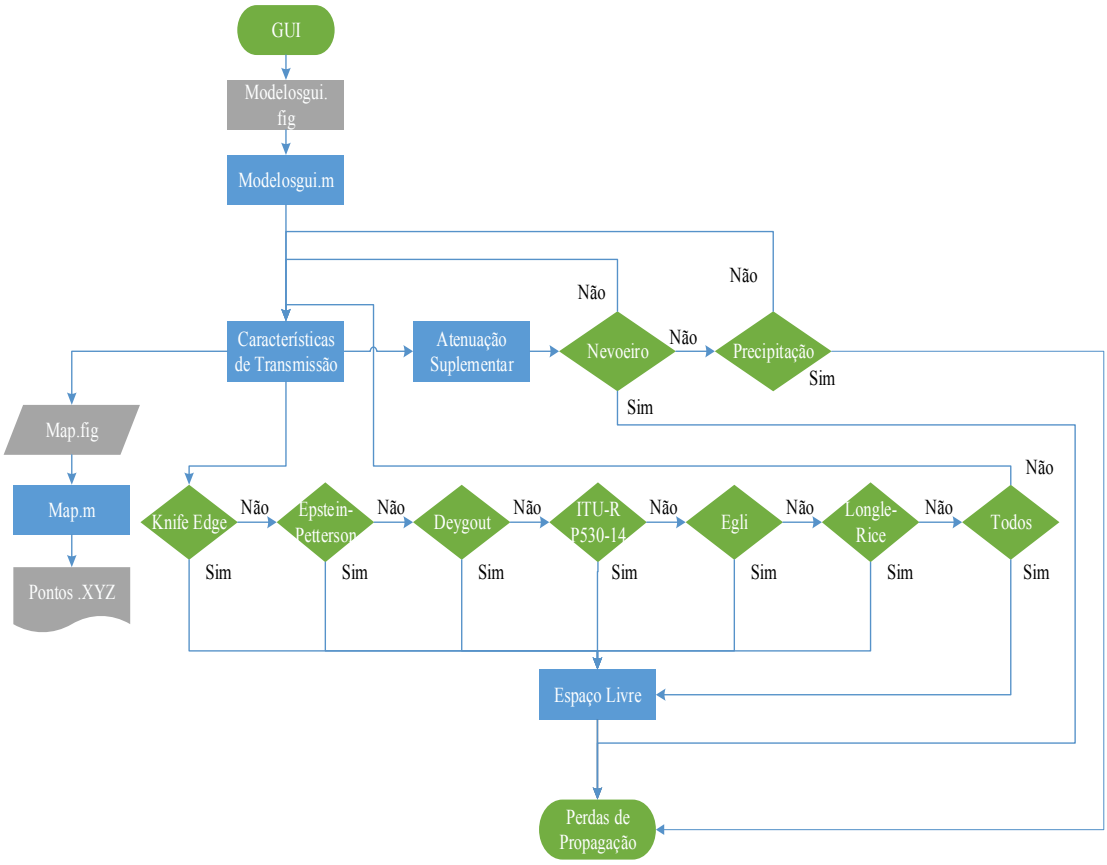



Figura 3.8 - Fluxograma referente a organização o programa de perdas de propagação.

No fluxograma verifica-se a relação entre os diversos blocos do programa. No entanto, existe uma dependência entre as funções dos modelos desenvolvidos. A relação entre funções pode ser consultada na Figura 3.9, onde se observa que os modelos de propagação em espaço livre e obstáculo em lâmina isolado servem de base aos restantes modelos. Os modelos de Egli e ITU-R não estão representados uma vez que são independentes destes modelos.

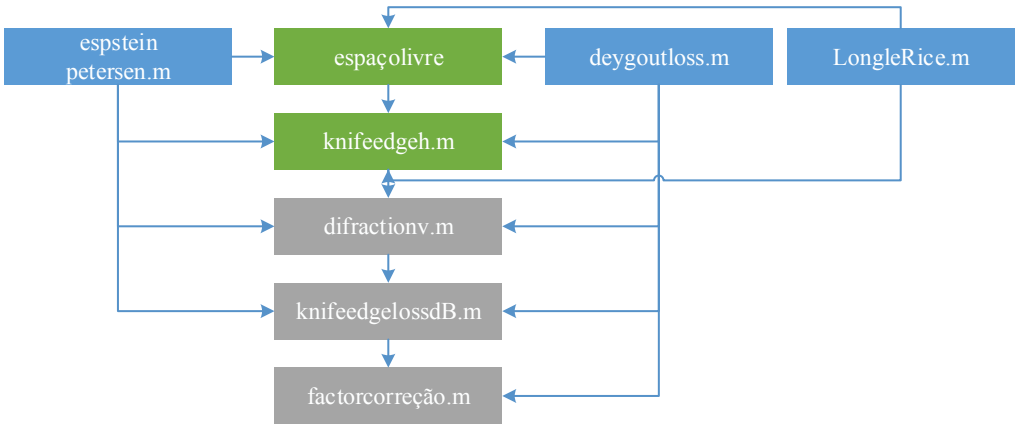



Figura 3.9 - Fluxograma de dependência de funções.

A interface gráfica resultante do programa desenvolvido pode ser observada na Figura 3.10 onde foram implementados os diversos modelos de perdas de propagação e atenuação suplementar devido a condições atmosféricas. Foram implementados os modelos em terrenos irregulares descritos em 2.3, modelos de perdas devido à vegetação descritos em 2.4 e a atenuação suplementar causada pelo vapor de água e densidade de oxigénio e, finalmente, por precipitação. Numa primeira fase é necessário introduzir os dados referentes a localização das antenas, emissora e recetora. Posteriormente introduzem-se os dados referentes às características de transmissão e, finalmente, seleciona-se o modelo de propagação pretendido.

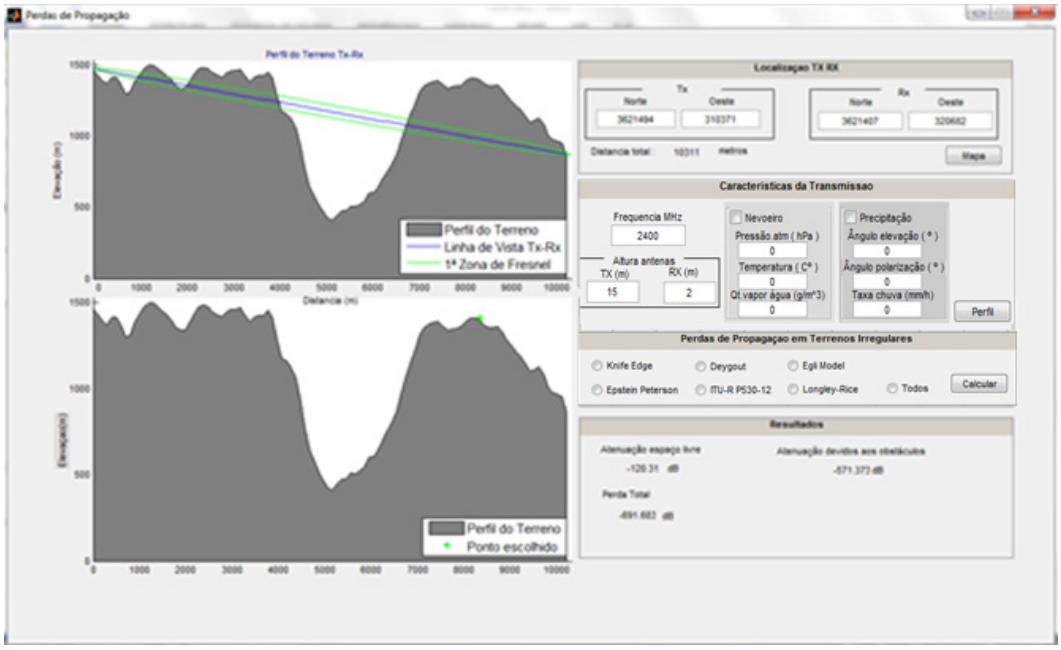



Figura 3.10 - Representação gráfica do programa de perdas de propagação desenvolvido.

Os mapas digitais de perfil em coordenadas "xyz" utilizados no programa desenvolvido foram disponibilizados pela DRIGOT (Direção Regional de Informação Geográfica e Ordenamento do Território), juntamente com ortofotomapas do Funchal e mapas em formato LIDAR (*Light Detection And Ranging*).

A aquisição dos pontos de emissão e receção de propagação pode ser efetuado de duas formas distintas. A primeira consiste na introdução das coordenadas em formato UTM (*Universal Transverse Mercator*) nos respetivos campos, como se pode verificar pela Figura 3.11.

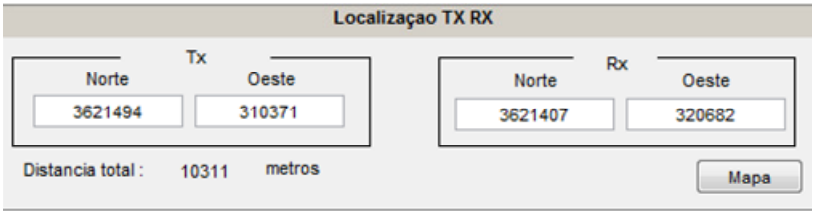



Figura 3.11 - Localização do emissor e recetor em coordenadas UTM.

A segunda reside na escolha de dois pontos através do cursor no mapa, como pode ser observado pela Figura 3.12.

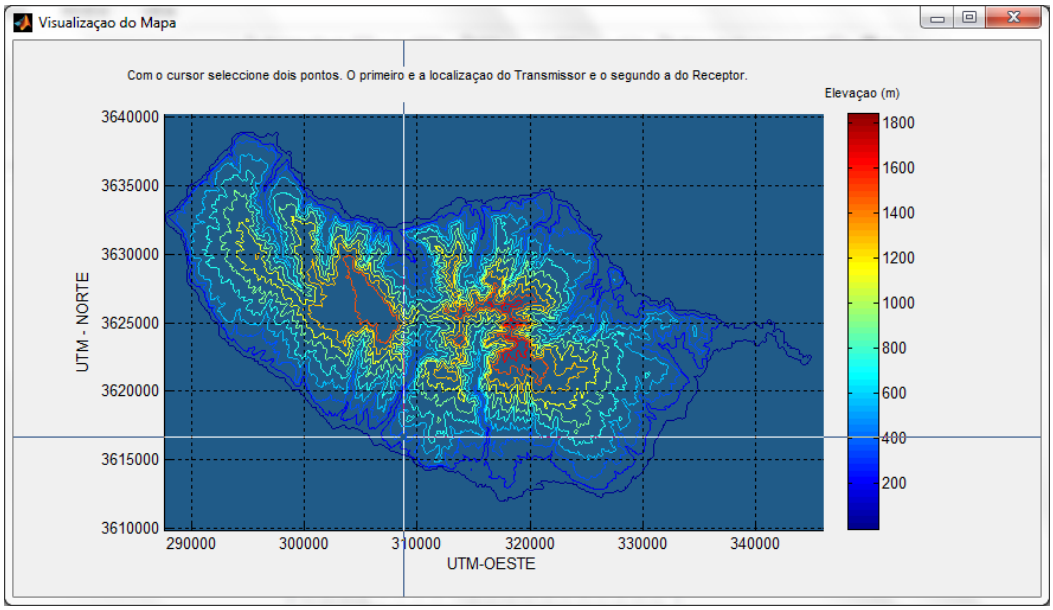



Figura 3.12 - Mapa para a aquisição dos pontos de propagação.

A representação do perfil do terreno na interface gráfica foi obtida utilizando-se o método apresentado em [13] para a análise de perfil de terreno. O núcleo de funcionamento baseia-se numa matriz com informação da elevação do terreno.

Na importação do mapa em formato xyz, são criados 3 vetores que correspondem a longitude, latitude e altitude. Caso a interpolação produza pontos com cota negativa estes são substituídos com valor igual a zero. A matriz é depois preenchida com a informação da altitude, e, para tal, é necessário converter um ponto geográfico representado por coordenadas (N e O) para um elemento da matriz. O processo encontrado foi a normalização de todas as coordenadas e respetiva conversão para os índices da matriz [13].

Após a importação do mapa para o Matlab, é necessário percorrer os elementos da matriz entre dois pontos quaisquer, permitindo desenhar de forma aproximada uma reta que une os dois pontos. Para tal é utilizado o algoritmo proposto por Jack E. Bresenham em 1962 (*Bresenham's line algorithm*). A amostragem do perfil do terreno e seleção dos obstáculos principais consiste em eliminar todo o perfil de terreno que não interfira com os 60% da primeira zona de Fresnel entre o emissor e o recetor [13].

A Figura 3.13 apresenta o menu onde se introduzem as características de transmissão, ou seja, a frequência de trabalho, alturas das antenas e atenuação suplementar causada por nevoeiro ou precipitação.

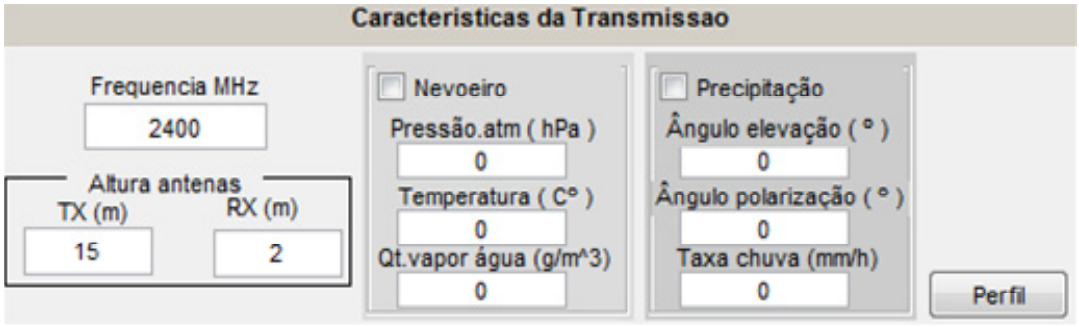



Figura 3.13 - Características da transmissão.

Ao selecionar-se atenuação suplementar, têm-se que preencher os campos referentes a pressão atmosférica, temperatura e quantidade de vapor de água no caso de nevoeiro e o ângulo de elevação do percurso, o ângulo de polarização e os valores da percentagem de tempo no ano em que o valor da intensidade de precipitação é excedido, apresentados no capítulo 2 secção 2.5.3, no caso de precipitação.

Tendo-se efetuado o preenchimento dos campos, das características de transmissão e após a introdução da localização da antena emissora e recetora, o botão de perfil permite uma representação gráfica do perfil do terreno escolhido, juntamente com a ilustração do primeiro elipsoide de Fresnel e percurso em linha de vista. Esta representação pode ser observada na Figura 3.14.

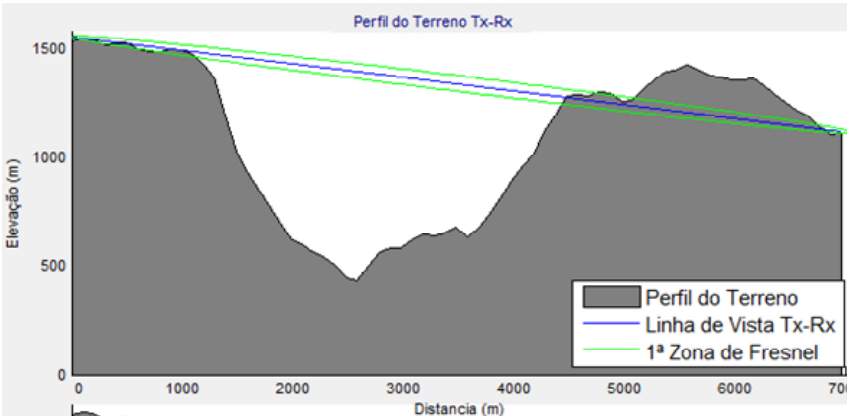



Figura 3.14 - Representação do perfil de transmissão.

Na Figura 3.15 tem-se os campos onde se seleciona os modelos de perdas de propagação em terrenos irregulares apresentados anteriormente.

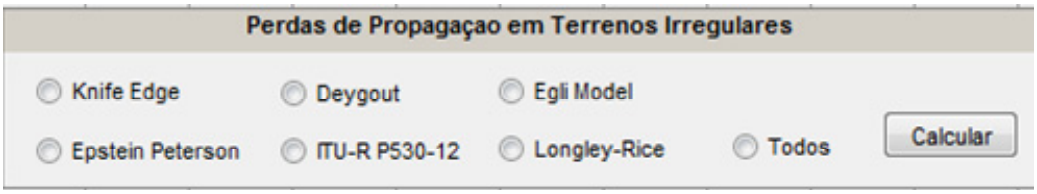



Figura 3.15 - Perdas de propagação em terrenos irregulares

Após a seleção e preenchimento dos campos referidos anteriormente, o programa permite, através do *icon* "Calcular", determinar uma representação a nível de ilustração

do modelo selecionado e são apresentados os valores de perdas de propagação do modelo ou modelos selecionados.

O modelo de *Knife Edge* dá uma representação gráfica dos parâmetros de difração de *Fresnel-Kirchhof* ($\upsilon$), enquanto o modelo de Egli foi escolhido porque prevê a atenuação total da ligação, ao contrário de outros modelos que apresentam uma perda adicional à perda em espaço livre, sendo útil como base de comparação para os restantes modelos.

No modelo de Deygout e Epstein Peterson optou-se por limitar o cálculo das perdas referente a quantidade de obstáculos (três), uma vez que, apesar de ser possível calcular para todo percurso, este cálculo torna-se complexo levando a tempos de execução elevados e resultados pessimistas.

Para possibilitar a comparação com os restantes modelos, é de referir que o ganho das antenas considerado é unitário.

## 3.4 Conclusões do capítulo

Neste capítulo estudaram-se alguns dos programas de simulação de perdas de propagação disponíveis.

O *AirLink* que é um programa fornecido juntamente com as antenas de feixes hertzianos que procura servir de ferramenta de auxílio a utilizadores inexperientes. Tem como vantagem ser bastante simples e como desvantagem requer uma ligação à internet e acesso ao programa *Google Earth*.

O programa *Radio Mobile* foi analisado uma vez que se trata de um dos programas mais utilizados a nível mundial. É gratuito e tem a vantagem que após o *download* dos mapas de perfil, não requerer acesso à internet e permitir o estudo de antenas personalizadas.

O programa de MATLAB implementado pode ser utilizado para fazer uma análise comparativa dos modelos existentes. Foram implementados os modelos de perdas de propagação em terrenos irregulares juntamente com as curvas de perdas dos modelos de vegetação, alem de permitir analisar as perdas devido a nevoeiro ou precipitação.

# 4. Desenvolvimento do protótipo

Neste capítulo relata-se o desenvolvimento de todo o sistema de transmissão e receção de dados de uma aplicação remota. Para isso, descreve-se a rede de sensores sem fios implementada, os sensores, a gestão de energia, a comunicação e *software* utilizado.

## 4.1 Requisitos referentes aos dispositivos a desenvolver

Pretende-se estudar e desenvolver um sistema de transmissão de dados de uma aplicação remota. Este sistema de comunicação permite a transmissão dos dados obtidos por uma rede de monitorização. Para esse efeito foi desenvolvida uma pequena rede, onde foram utilizados vários sensores e implementados os respetivos circuitos de condicionamento de sinal.

Os sensores utilizados referem-se à monitorização de parâmetros ambientais e devem respeitar as seguintes características:

- Módulos com sensores de pequenas dimensões e estanques, uma vez que podem ser colocados em ambientes com condições ambientais adversas.
- Reduzido consumo energético, uma vez que os módulos não têm acesso à rede de energia elétrica, sendo alimentados através de energia solar.
- Permitir a amostragem dos valores de temperatura, humidade relativa, luminosidade, pressão atmosférica e valores de tensão na bateria.

Os módulos de sensores, após recolha de amostras e reencaminhamento pela rede, devem entregar os dados a um nó coordenador que será responsável por:

- Processar os dados e armazená-los num cartão SD (*Secure Digital*).
- Sincronizar o relógio local e introduzir a marca temporal nas tramas recebidas.
- Gerir o sistema de envio de dados através da antena de feixes hertzianos.
- Ter um reduzido consumo energético, devido a não ter acesso à rede de energia elétrica, sendo alimentado através de energia solar.

### 4.1.2 Arquitetura do sistema

Na Figura 4.1 apresenta-se a arquitetura do sistema proposto em que se pode verificar a interligação entre os diversos componentes que constituem o nó que tem como principais funções a monitorização de parâmetros ambientais.

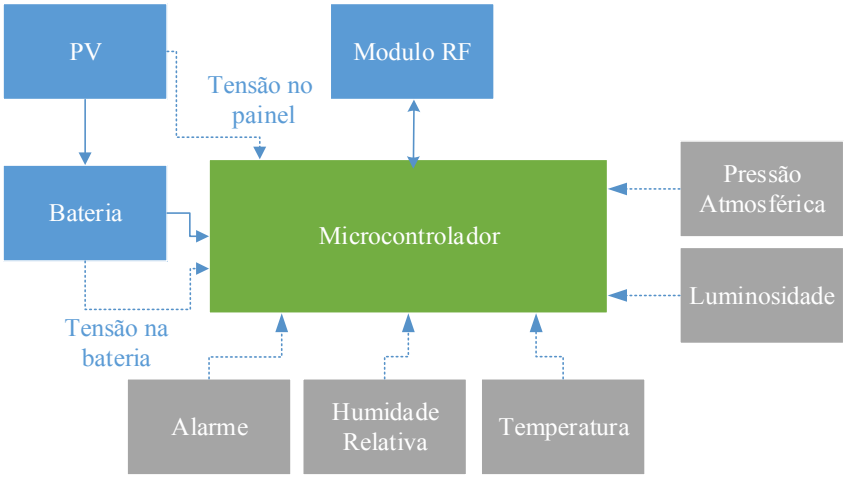



Figura 4.1 - Diagrama de blocos do nó sensor.

Pela arquitetura desenvolvida pode-se verificar a dependência entre os componentes constituintes do nó terminal e as suas características principais. Este nó tem de ser capaz de receber os valores dos diversos sensores, processar a informação e enviar os dados através do módulo de rádio frequência. Para além disso, também deve ser capaz de manter e controlar a carga da bateria utilizada na alimentação de todo o sistema.

A arquitetura presente na Figura 4.2 refere-se à do nó coordenador, tendo este as funções de recolha dos dados da rede de sensores sem fios. Permite a receção e envio de dados, tem a capacidade de processamento de dados e controlo sobre os sistemas de carregamento de baterias e envio de dados através da antena de feixes hertzianos.

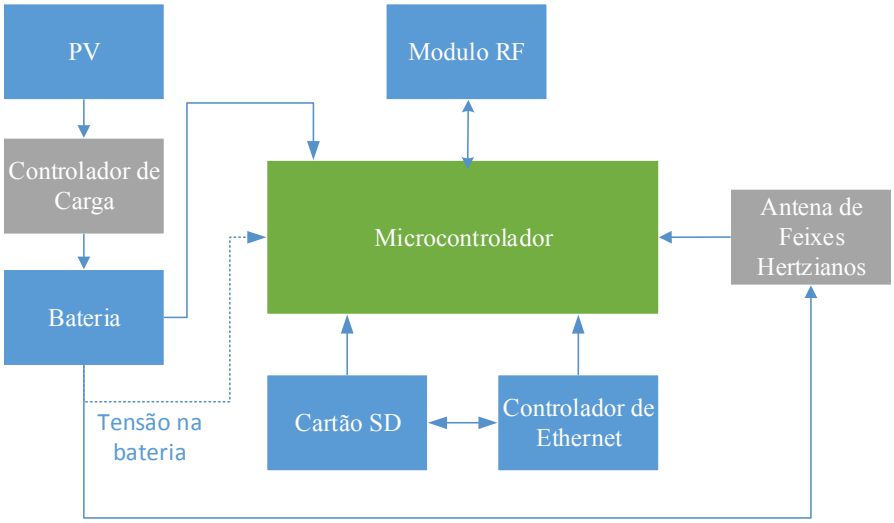



Figura 4.2 - Diagrama de blocos do nó coordenador.

Na Figura 4.3 apresenta-se a arquitetura do sistema proposto. O sistema contém uma rede de sensores sem fios ZigBee / 802.15.4, na qual a informação, depois de reencaminhada pela rede, é armazenada num nó coordenador. Periodicamente é enviada para um servidor ligado a internet, através de um sistema de comunicação por feixe

hertziano. O servidor permite a visualização da informação através de uma plataforma informática, juntamente com a extensão da própria internet.

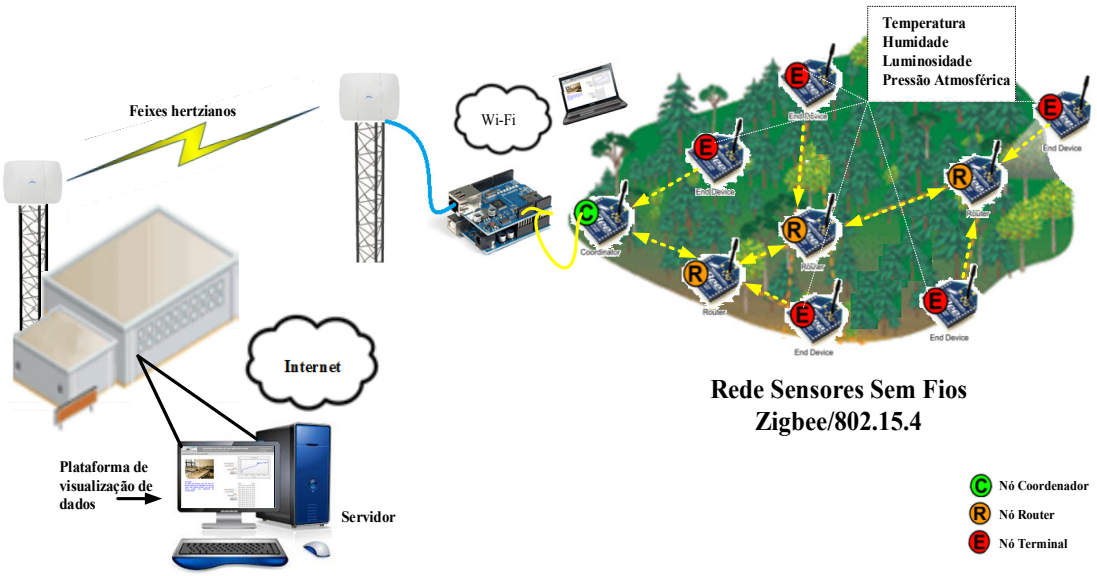



Figura 4.3 - Arquitetura do sistema proposto.

## 4.2 Rede de sensores sem fios

As redes de sensores sem fios datam dos anos 50 do século passado e consistem, tipicamente, numa rede de nós que são capazes de descobrir e comunicar com nós vizinhos, formando uma rede com topologia em malha que comunica com um servidor através de uma *gateway* [36].

Uma rede de sensores sem fios (WSN – *wireless sensor network*) é um caso particular de uma rede *ad-hoc* em que os nós sensores são pequenos dispositivos equipados com funcionalidades de sensoriamento, processamento e com sistema de comunicação sem fios. Cada nó mede os dados de uma determinada região e um algoritmo de encaminhamento escolhe o percurso mais adequado para transmissão através da rede. Os nós não têm de comunicar diretamente com a *gateway*, mas apenas com os seus vizinhos. Uma transmissão mais curta traduz-se em menos potência, permitindo o uso de dispositivos alimentados por baterias, com a vantagem de poderem ser recolocados com relativa facilidade.

Neste trabalho lidou-se com o protocolo *Zigbee*/802.15.4 [37]. O *ZigBee* é um protocolo de redes sem fios, baseado na norma 802.15.4, projetado para ser um protocolo simples, robusto e de baixo custo, utilizando sinais rádio digitais de baixo consumo, empregando uma topologia *mesh*, que proporciona uma alta confiabilidade e segurança.

## 4.3 Nó terminal

Os nós terminais ou nós sensores são responsáveis pela aquisição de dados através dos sensores adequados. Devem fazer uma utilização eficiente da energia e estarem equipados com memória e capacidade de comunicação. Os recursos de memória normalmente são reduzidos e a comunicação é limitada, sendo apenas efetuada com o *router* mais próximo. Os nós podem permitir o adormecimento, possibilitando a sua alimentação com recurso a baterias ou energias renováveis.

No nó terminal desenvolvido, o microcontrolador responsável pelo controlo do sistema é o ATmega328P. A comunicação rádio frequência é feita através do módulo de comunicação XBee. Os sensores utilizados são o SHT11 para medir a temperatura e a humidade relativa do ar, S1087 para medir a luminosidade e o kit MPL115A1 para medir a pressão atmosférica. Para a alimentação do nó sensor recorreu-se a uma bateria de lítio recarregável por energia solar através de um painel solar de 1 W. O controlo de carregamento da bateria é efetuado pela plataforma de desenvolvimento Arduíno Fio que serve de suporte ao próprio microcontrolador.

### 4.3.1 XBee

Neste projeto recorreu-se aos módulos de comunicação RF XBee produzidos pela empresa *DIGI* [38]. Os módulos XBee suportam o protocolo de comunicação *ZigBee* e operam na banda de frequências dos 2,4 GHz. Têm um baixo consumo e permitem o modo de adormecimento para poupar energia. As suas principais caraterísticas são as seguintes:

- Frequência: 2,4 GHz;
- Taxa de transmissão: até 250 Kbps;
- Consumo TX: 40 mA (@ 3,3 V);
- Consumo RX: 40 mA (@ 3,3 V);
- Modo adormecido: inferior a 1µA;
- Potência de transmissão: 2 mW (3 dBm);
- Gama de tensão de alimentação: 2,1 V a 3,6 V;
- Sensibilidade: -96 dBm.

A Figura 4.4 apresenta o XBee Serie 2 ZB utilizado neste protótipo, sendo de destacar o facto do modelo apresentado ter um conector RPSMA (*Reverse Polarity SubMiniature version A)*, o que permite a utilização de diferentes tipos de antenas.


Figura 4.4 - Módulo XBee serie 2 ZB [39].

O XBee foi configurado no modo de funcionamento API (*Application Programming Interface)*. Este modo faz uso de uma trama para encapsular os dados a serem transmitidos, possibilita uma comunicação dinâmica e permite explorar as potencialidades de uma rede em malha. O formato geral da trama API do XBee está representado na Figura 4.5.

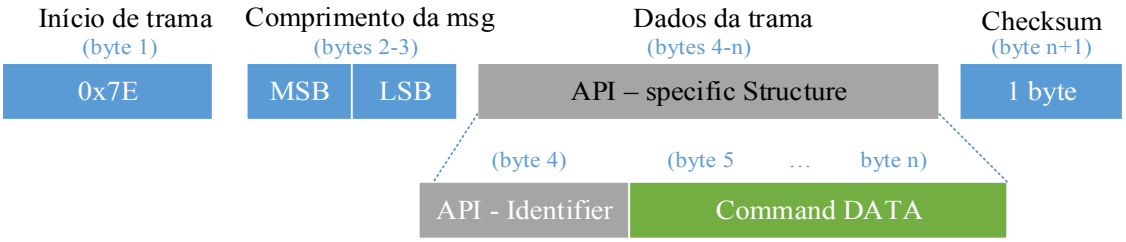



Figura 4.5 - Estrutura da trama API do XBee.

A trama de dados apresenta geralmente a seguinte estrutura: 1 byte de início de trama (0x7E), 2 bytes com a indicação do comprimento da mensagem (excluído os campos anteriores e o *checksum*), 1 byte identificador da mensagem, 8 bytes referente ao endereço de 64 bits do destinatário ou remetente, 2 bytes do endereço de 16 bits do destinatário ou remetente, 2 bytes de configurações opcionais, até 72 bytes de dados e finalmente 1 byte de *checksum*. Os identificadores de mensagem poderão ser consultados no Anexo IV.

A programação dos módulos RF dos nós é efetuada através do programa XCTU, apresentado na Figura 4.6. O programa foi desenvolvido pela Digi e o *download* pode ser efetuado gratuitamente em [40]. Os campos mais importantes a configurar são o PAN ID (*Personal Area Network ID*) e o endereço de 64 bits de destino.

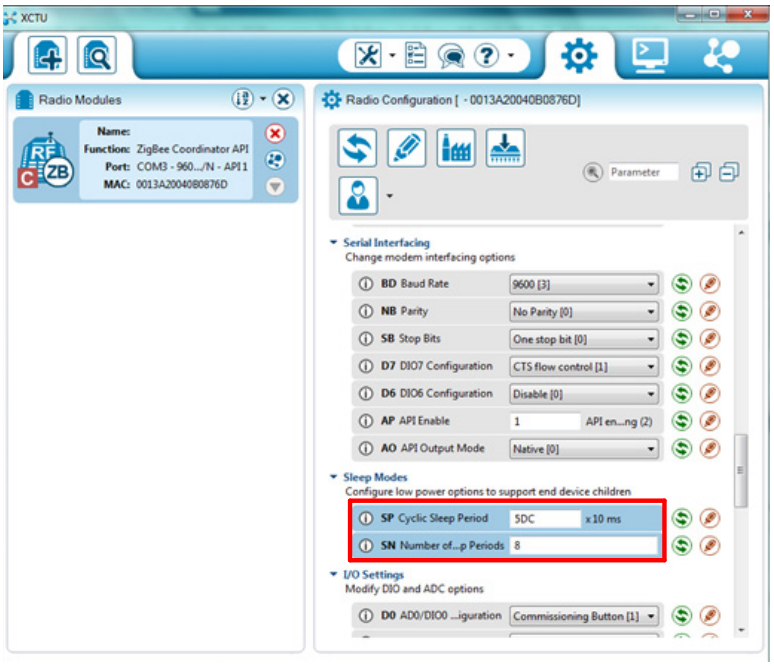



Figura 4.6 - Programa XCTU.

Para a programação correta dos módulos XBee, devido ao modo de adormecimento, torna-se importante o preenchimento dos campos de "*SP - Cyclic Sleep Period*" e de "*SN – Number of Cyclic Sleep Periods"* estes campos determinam o tempo para o qual a tabela de filhos é mantida apesar de não existir comunicação entre eles. Estes campos têm de ser idênticos tanto no coordenador como nos *routers*.

Na configuração da rede de sensores sem fios considerou-se a pior situação possível, em que o nó pode ficar sem comunicação devido a não ser possível estabelecer comunicação entre o nó coordenador e o servidor onde são entregues os dados. Neste caso, devido a "*time outs"* é possível ficar sem comunicação aproximadamente 5 minutos, tendo-se, então configurado o SP para 5DC (1500 em hexadecimal) e o SN para 8, obtendo um total de $3 \times SN \times (SP \times 10\ \text{ms}) = 360000\ \text{ms} = 6\ \text{m}$.

### 4.3.2 Arduíno Fio

Os microcontroladores são a base para a automação das aplicações e para esta aplicação escolheu-se o Arduíno Fio, apresentado na Figura 4.7, tratando-se de uma plataforma de desenvolvimento baseado no microcontrolador ATmega328P [41]. Possui 14 pinos digitais, 6 dos quais podem ser utilizadas como saídas PWM (*Pulse Width Modulation*) de 8 bits, 8 portas analógicas com resolução de 10 bits e um cristal de 8 MHz. Tem um conector para colocar uma bateria de polímeros de lítio e inclui um circuito de carga. Na parte posterior tem um *socket* XBee possibilitando a comunicação sem fios.

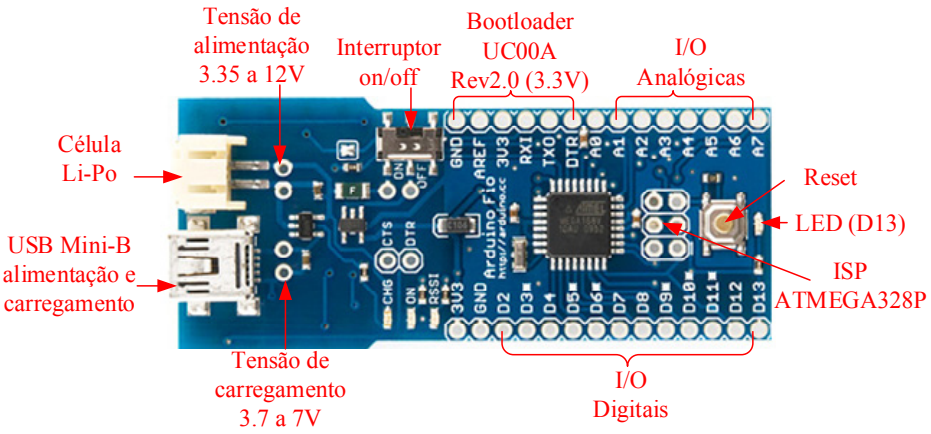



Figura 4.7 - Placa de desenvolvimento Arduíno fio.

A sua tensão de funcionamento é de 3,3 V e esta pode variar entre os 3,35 a 12 V. Disponibiliza uma corrente máxima de 40 mA por porta digital. O ATmega328P tem 32 KB de memória *flash* para armazenamento de código, dos quais 2 KB são reservados ao *bootloader*.

A escolha deste controlador deve-se à existência de um encaixe para o módulo XBee, devido ao seu baixo consumo e desenho compacto, sendo, neste caso, o mais indicado na construção de um nó sensor.

### 4.3.3 Sensor de humidade e temperatura

A escolha do sensor para medir os valores de humidade e temperatura recaiu sobre o SHT11 fabricado pela Sensirion [42] Figura 4.8, com um sinal de saída digital fornecido por um ADC (*Analog-to-Digital Converter*) de 14 bits.

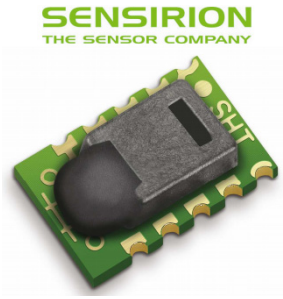



Figura 4.8 - Sensor de humidade e temperatura SHT11.

O SHT11 mede temperaturas numa gama de – 40º a + 123,8º C e valores de humidade relativa de 0 a 100%. O esquema representado na Figura 4.9 a) mostra a interligação entre o sensor e o microcontrolador, enquanto a Figura 4.9 b) apresenta o diagrama de blocos do sensor.

a) b)

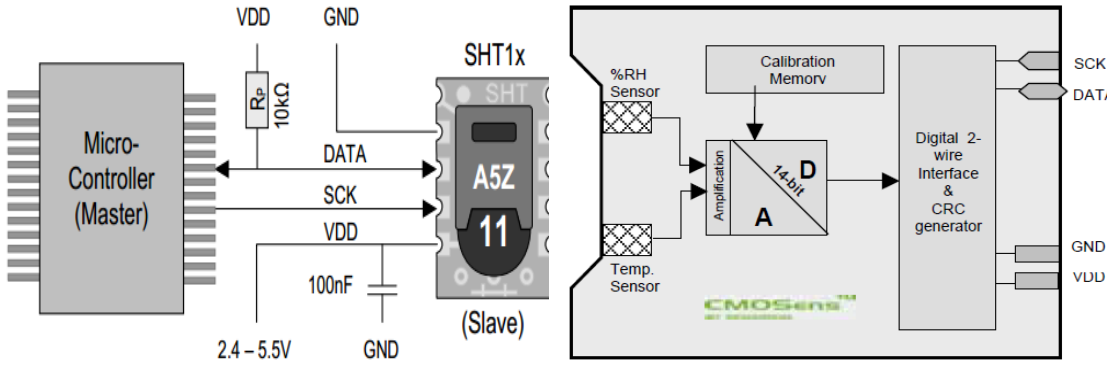



Figura 4.9 - a) Ligação entre o sensor e o microcontrolador. b) Diagrama de blocos.

A tensão de alimentação pode variar entre os 2,4 a 5,5 V sendo o valor recomendado de 3,3 V e o seu consumo é de 90 µW. O condensador permite a filtragem de variações bruscas no sinal de entrada enquanto a resistência $R_p$ evita impedimentos na comunicação entre o microcontrolador e o sensor.

Os sensores de humidade e temperatura enviam os dados analógicos através de interfaces série dedicadas, ligadas às portas SCK, que é responsável pela sincronização da comunicação e DATA que é responsável pela transferência de dados.

O valor da temperatura é calculado pela seguinte fórmula:

$$Temp(°C) = Amost_{Temp} * 0{,}01 - 40 \quad (4.1)$$

e o valor da humidade relativa:

$$Hum(\%) = (Temp - 25) * (t_1 + t_2 * Amost_{Hum}) + c_1 + c_2 * Amost_{Hum} + c_3 * Amost_{Hum}{}^2 \quad (4.2)$$

com $t_1 = 0{,}01;\ t_2 = 0{,}0008;\ c_1 = -4;\ c_2 = 0{,}0405;\ c_3 = -0{,}0000028$

### 4.3.4 Sensor de luminosidade

Para a recolha da luminosidade escolheu-se o fotodíodo S1087 fabricado pela *Hamamatsu* [43] e apresentado na Figura 4.10 a). Este é caracterizado por uma baixa corrente escura (*dark current* – corrente existente sem qualquer sinal ótico incidente) devido ao seu encapsulamento numa camada cerâmica. O sensor opera numa gama de temperatura entre os - 10 °C e + 60 °C e tem uma resposta espetral dos 320 aos 730 nm, com um pico de sensibilidade a 560 nm como mostra a Figura 4.10 b). É recomendado pelo fabricante a utilização do circuito condicionamento de sinal apresentado na Figura 4.10 c), que se baseia num circuito de transimpedância.

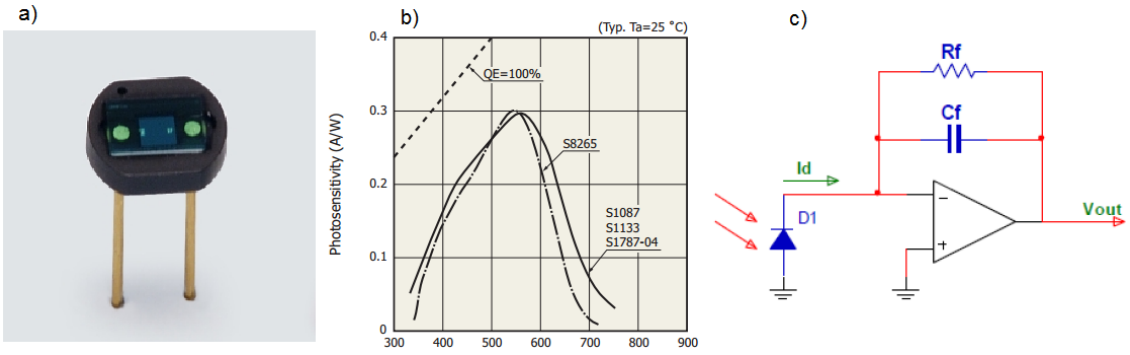



Figura 4.10 - a) Sensor S1087. b) Resposta do fotodíodo. c) Circuito de condicionamento.

No circuito de condicionamento de sinal utilizou-se um condensador cerâmico de 22 nF, um amplificador operacional LM358 e uma resistência de 1,8 kΩ em série com uma resistência variável de ajuste fino de 10 kΩ. A resistência variável é utlizada na calibração do sensor de luminosidade.

A tensão de saída em função da corrente presente no fotodíodo é dada por:

$$V_{out} = R_f * I_d \quad (4.3)$$

e a luminosidade dada por:

$$Lum = I_d * 10^9 \text{ (lux)} \quad (4.4)$$

Para o sensor não entrar em saturação, foi necessário dimensionar o circuito para uma tensão máxima na porta analógica do microcontrolador de 1,1 V utilizando o comando anlogReference(INTERNAL). O sensor foi calibrado para um valor máximo de 130 000 lux que é o valor máximo de luminosidade.

### 4.3.5 Sensor de pressão atmosférica

O MPL115A1 é um barómetro digital de pressão absoluta com saída SPI destinado ao mercado de aplicações de baixo custo. Com dimensões de 5 x 3 x 2,1 milímetros, é ideal para dispositivos eletrónicos portáteis ou para espaços limitados. A gama de alimentação vária entre os 2,375 V e 5,5 V sendo o valor recomendo de 3,3 V. O baixo consumo de 5 µA em modo ativo e 1 µA em modo de adormecimento são essenciais em aplicações de baixa potência. A temperatura de operação estende-se desde os - 40 ºC a + 105 ºC ajustando-se às mais exigentes condições ambientais.

A Figura 4.11 apresenta o diagrama de blocos do KITMPL115A1, que usa um sensor de pressão com tecnologia MEMS (*Microelectromecanical*) com um circuito integrado de condicionamento de sinal fornecendo, assim, medições precisas numa gama de 50 a 115 kPa.

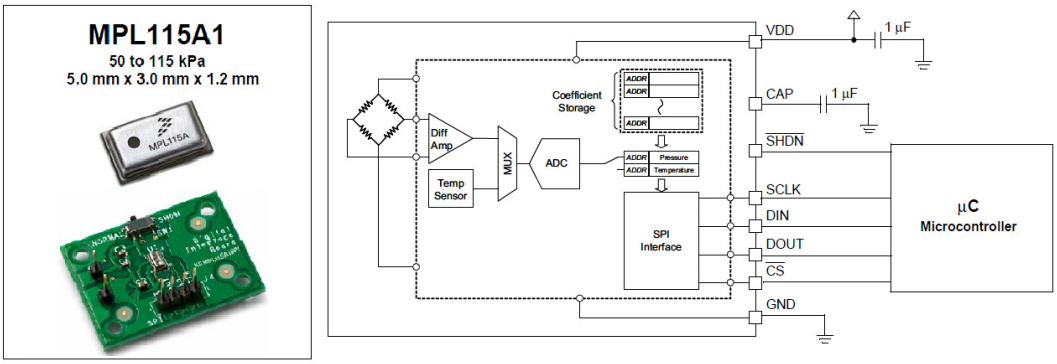



Figura 4.11 - Sensor de pressão atmosférica e diagrama de blocos.

O circuito integrado ADC converte as leituras dos sensores de temperatura e pressão em valores digitais à saída da porta SPI (*Serial Peripheral Interface*). As calibrações de fábrica são armazenadas numa ROM (*Read-Only Memory*) interna. Fazendo uso dos valores em bruto recebidos pelos sensores juntamente com os dados de calibração armazenados, o microcontrolador aplica um algoritmo de compensação fornecendo, assim, a pressão absoluta compensada com uma precisão de ± 1 kPa.

O valor da pressão compensada na saída de 10 bits, é calculada por:

$$P_{comp} = a0 + (b1 + c12 * T_{adc}) * P_{adc} + b2 * T_{adc} \quad (4.5)$$

sendo $P_{adc}$ e $T_{adc}$ o valor da pressão e temperatura à saída da ADC, respetivamente, $a0$ é o coeficiente de offset da pressão, $b1$ é o seu coeficiente de sensibilidade, $b2$ é o coeficiente de 1ª ordem de offset da temperatura TCO (Temperature Coefficiente of Offset) e $c12$ é o seu coeficiente de sensibilidade TCS (Temperature Coefficient of Sensitivity).

$P_{comp}$ irá produzir um valor 0 com a pressão de 50 kPa e o valor máximo de 1023 com a pressão de 115 kPa, senso a pressão dada por

$$Pressão\ (kPa) = P_{comp} * \left[\frac{115-50}{1023}\right] + 50 \quad (4.6)$$

### 4.3.6 Tensão na bateria e no painel solar

A tensão na bateria e a tensão no painel solar são parâmetros importantes quando se trata de aplicações de baixo consumo. A medição da tensão na bateria é importante para o circuito de controlo de carga e para se saber se é necessário proceder à troca das baterias. Em conjunto com a monotorização da tensão no painel solar, pode-se verificar e despistar possíveis avarias.

Para se efetuar a medição dos valores de tensão utilizaram-se divisores resistivos de modo a baixar o valor de tensão, uma vez que o microcontrolador apenas permite um máximo de 1,1 V por porta analógica, devido ao processo descrito na secção 4.3.4.

### 4.3.7 Sistema de interrupção

Os sistemas de monotorização em ambientes remotos podem ser adequados a muitas aplicações. Por isso, apesar de não constar nos objetivos do projeto, optou-se por incluir um sistema de alarme simples que pode ser aplicado a diversas situações. O circuito escolhido foi o da Figura 4.12.

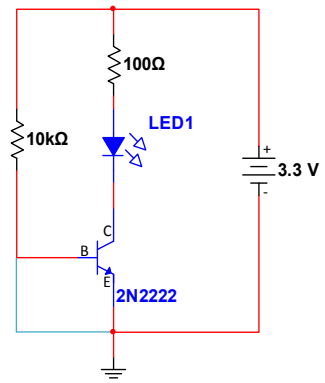



Figura 4.12 - Circuito de alarme.

Este circuito é composto por um transístor NPN 2N2222, uma resistências de 10 kΩ, uma resistência de 100 Ω e um led. A corrente circula desde o polo positivo pelo caminho de menor resistência até a terra. Quando é interrompida a ligação entre a base e o emissor do transístor (fio azul), a corrente vê-se forçada a polarizar a base do transístor, que atua como um interruptor, ligando o led que ativa uma interrupção no código. Esta função gera um alarme que interrompe o ciclo de adormecimento e envia um bit (1 ou 0) alertando o utilizador.

O fio azul representa a zona do circuito que pode ser interrompida. Pode ser um fio de cobre que ao ser danificado quebra a ligação, um feixe laser ou simplesmente um contacto entre dois elementos. Pode ser utilizado para a verificação de abertura do próprio nó sensor, queda de objetos ou outra aplicação semelhante.

### 4.3.8 Protótipo desenvolvido

O protótipo desenvolvido é composto por um painel solar, uma bateria, um XBee, um microcontrolador arduíno fio e uma placa de circuito impresso que serve de suporte aos sensores e aos seus circuitos de condicionamento.

O circuito completo está representado na Figura 4.13. Neste circuito observa-se as ligações entre os seus componentes.

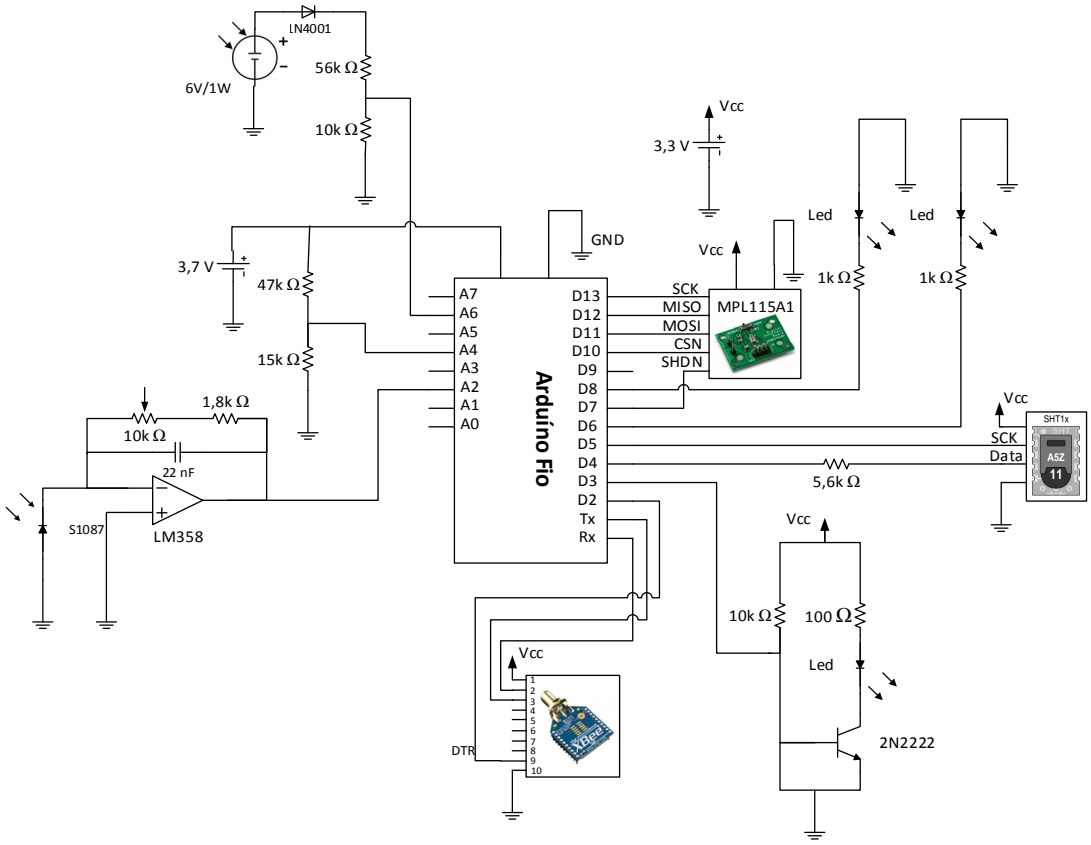



Figura 4.13 - Circuito desenvolvido.

Na Figura 4.14 observa-se o protótipo desenvolvido.

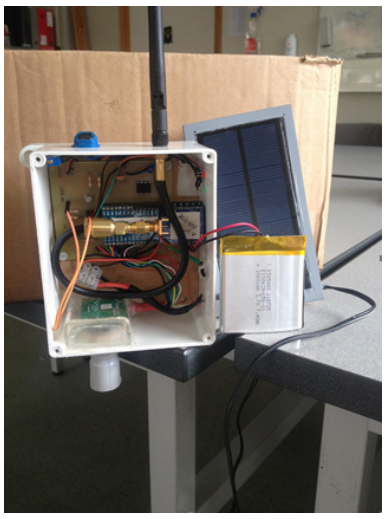


Figura 4.14 - Protótipo do nó sensor.

Encontra-se dentro de uma caixa estanque para exteriores de 120x100x60 mm fabricada em ABS (Acrylonitrile Butadiene Styrene), com índice de proteção de IP65 e índice contra choques de IK08 [44].

Dentro da caixa foram colocados os componentes eletrónicos com o suporte de uma placa de circuito impresso desenvolvida para esse efeito. Juntamente com os componentes colocou-se uma caixa em acrílico vedada com silicone que serve para colocar o sensor de pressão atmosférica. Para a localização do sensor de humidade e temperatura recorreu-se a instalação de um bucim com uma rede de proteção na parte inferior da caixa. Isto porque os valores do interior da caixa serão diferentes dos exteriores e aproveita-se esta abertura para igualizar a pressão atmosférica.

Colocaram-se dois leds de aviso, verde quando o sistema está a funcionar corretamente piscando quando envia a trama e vermelho quando é detetado um erro.

Na Figura 4.15 a) representa-se o fluxograma da programação do nó sensor, que pode ser consultada no Anexo V. Este nó após a inicialização das variáveis verifica o alarme. Caso a tensão seja diferente de zero ativa a função *Read and Send* e envia uma *Flag* que passa de "0" a "1". Caso o alarme não seja ativado, verifica se o Counter já chegou ao valor predefinido, após o qual executa a função *Read and Send* representada na Figura 4.15 b).

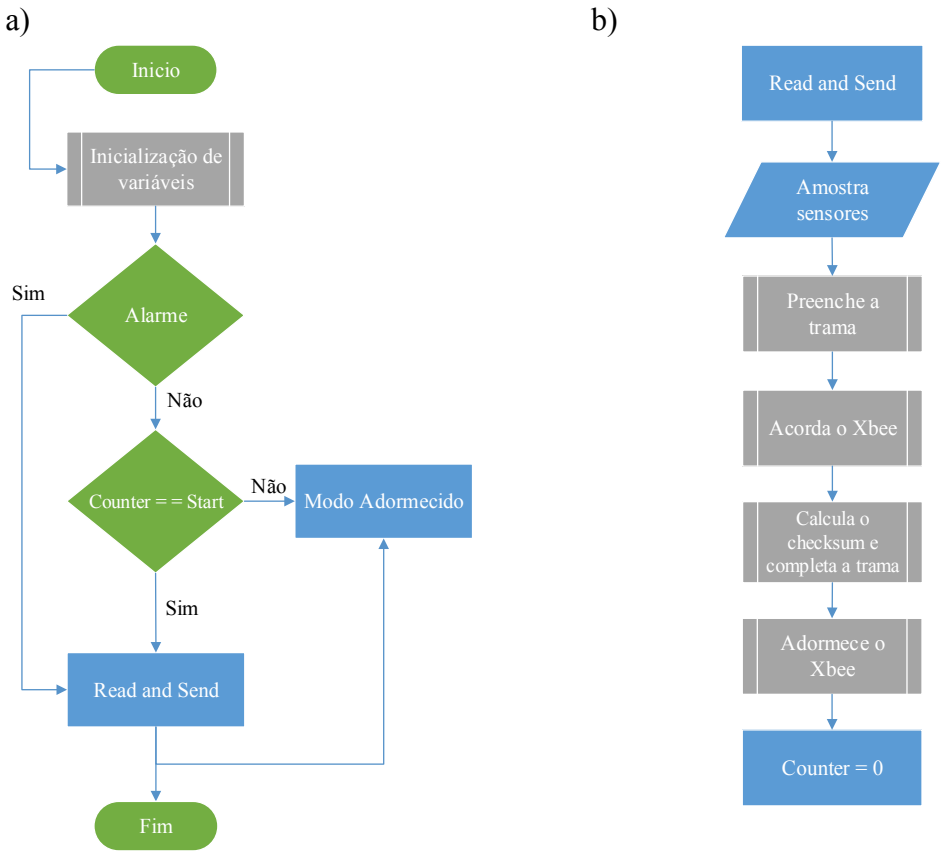



Figura 4.15 - a) Fluxograma do programa do nó sensor. b) Fluxograma da função *Read and Send*.

O adormecimento do XBee apenas é possível caso seja configurado no Arduíno e no próprio XBee. No Arduíno torna-se necessário associar um pino digital (pino D2) ao

pino de DTR do módulo XBee. No XBee configura-se com recurso ao programa X-CTU, apresentado anteriormente, o campo referente ao *Sleep Mode* selecionando-se o modo 1, ou seja, *Sleep Mode -1- PIN HIBERNATE*.

## 4.4 Nó Coordenador

O nó coordenador é responsável pelo maior conjunto de funções. É responsável pela de recolha e processamento de dados, pelo estabelecimento da rede, pela distribuição de endereços e tem de ser capaz de armazenar informação sobre a rede, incluindo as chaves de segurança.

A receção de dados no coordenador faz-se recorrendo ao módulo RF XBee que recebe as tramas provenientes dos nós terminais. Insere uma marca temporal e efetua o armazenamento através de um módulo micro SD.

O nó coordenador também é responsável pelo envio dos dados para o servidor. Para isso tem de coordenar a ligação entre a antena de feixes hertzianos e a leitura dos dados no cartão micro SD. Este processo é realizado utilizando-se um módulo de Ethernet para fazer a comunicação entre o microcontrolador e a antena.

Finalmente, também se atribuiu ao nó coordenador parte da gestão de energia. O controlo de carga da bateria, alimentada através de um painel solar, efetua-se utilizando um controlador de carga. No entanto, a alimentação da antena de feixes hertzianos e o envio de dados é controlado pelo nó coordenador.

### 4.4.1 Arduíno Mega

O Arduíno Mega 2560, apresentado na Figura 4.16, é um microcontrolador baseado no ATmega 2560 [45]. Possui 54 portas digitais, 14 das quais podem ser utilizadas como saídas *PWM* de 8 bits, 16 portas analógicas com resolução de 10 bits, 4 UARTS (*Universal Asynchonous Recever/Transmitter*), portas série de hardware e um cristal de 16 MHz.

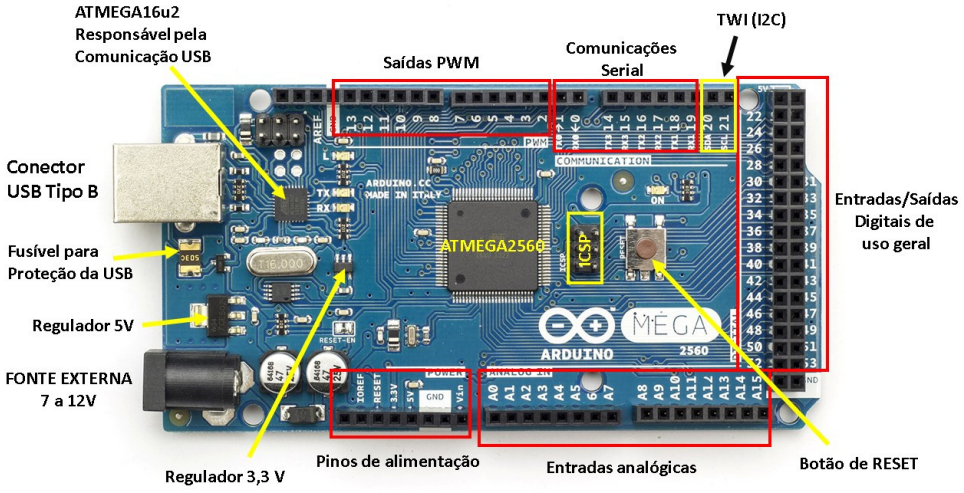



Figura 4.16 - Placa de desenvolvimento Arduíno Mega.

A sua tensão de funcionamento é de 5 V e pode variar entre os 7 a 12 V admitindo como limites de funcionamento uma gama entre os 6 a 20 V. Pode fornecer um máximo de 40 mA por porta digital ou 50 mA na alimentação de 3,3 V. O ATmega 2560 tem 256 KB de memória *flash* para armazenamento de código, dos quais 8 KB são utilizados pelo seu *bootloader*.

A escolha recai sobre este microcontrolador, uma vez que o código desenvolvido tinha 35 KB e este preenchia todos os restantes requisitos de energia e controlo.

### 4.4.2 Receção de dados

A recolha de dados é efetuada recorrendo ao módulo RF XBee, aproveitando-se o identificador de mensagem. Este processo segue o fluxograma da Figura 4.17 e está representado no fluxograma da Figura 4.25 pelo bloco da função ***Show_write_Trama***.

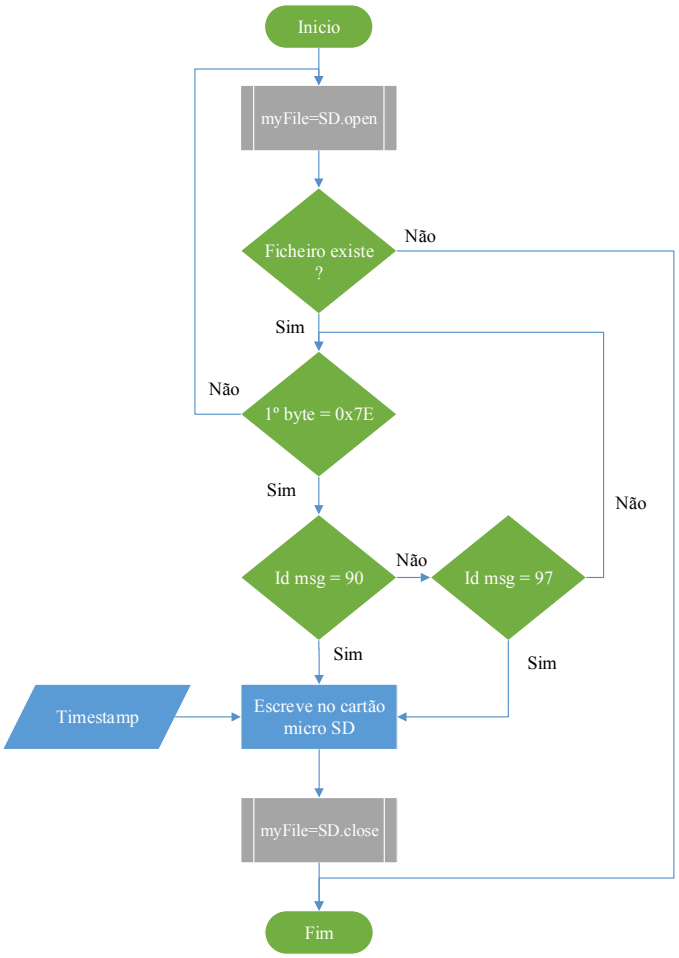



Figura 4.17 - Fluxograma da recolha de dados.

O rádio XBee recebe a trama, depois acede ao cartão SD e cria um ficheiro de texto. No passo seguinte é a verificado se o ficheiro criado existe. Caso exista, determina se o primeiro byte da trama é igual a 0x7E, validando a trama. Após o qual o código verifica o identificador de mensagem e, conforme o valor, escreve no cartão micro SD acrescentando uma marca temporal (*timestamp*) de receção de dados. Findo este processo o algoritmo fecha o ficheiro criado e continua o restante código.

### 4.4.3 Informação temporal

A informação temporal (*timestamp*) é de grande importância num sistema de aquisição de dados. São poucos os casos em que os dados podem ser desassociados do tempo de aquisição e neste sistema, devido ao nó coordenador não estar junto ao servidor, teve-se que acrescentar informação temporal a trama de dados recebida, que depois é escrita no cartão SD. Para tal, aproveitou-se o acesso que é feito ao servidor. Quando são entregues os dados, o código desenvolvido usa a ligação para aceder a um servidor de NTP (*Network Time Protocol*) e atualizar o relógio. Na Figura 4.18 observa-se o fluxograma deste processo de aquisição.

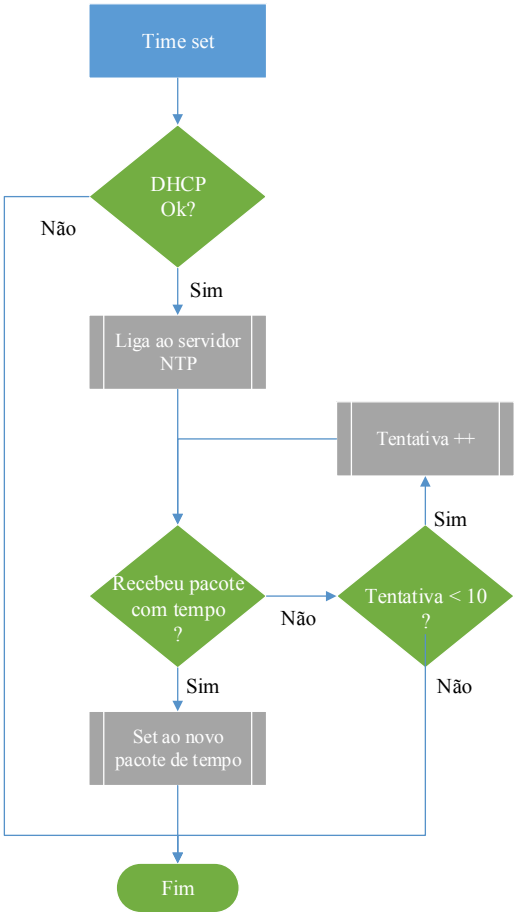



Figura 4.18 - Fluxograma com o processo de aquisição de relógio.

Para a atualização do relógio torna-se necessário, em primeiro lugar, verificar se existe resposta de DHCP (*Dynamic Host Configuration Protocol*). Caso exista, é feito um pedido ao servidor de NTP enviando um pacote de dados específico e ficando a espera da confirmação. No caso de não obter resposta são feitas 10 tentativas, após o qual o pedido de ***Timeset*** é descartado.

### 4.4.4 Sistema de armazenamento de dados

O armazenamento de dados tem uma importância cada vez maior nos sistemas de sensores, devendo ser extremamente confiáveis por causa da sua natureza autónoma, uma

vez que podem operar por longos períodos de tempo, requerendo pouco ou nenhuma supervisão e podem ser instalados em locais remotos ou hostis.

Tendo em consideração as condicionantes e a importância do sistema de armazenamento de dados, utilizou-se numa primeira fase os componentes presentes na Figura 4.19.

a) b)


Figura 4.19 - a) Módulo leitor cartão SD. b) Cartão micro SD.

O leitor de micro SD comunica através de interface SPI, o modelo de comunicação pode ser observado na Figura 4.20. O leitor suporta cartões micro SD com armazenamento até 2 Gb, tem uma gama de alimentação dos 4,5 a 5V e pode operar nos 5V ou 3,3 V devido à existência de um regulador de tensão para 3,3 V. Além disso tem dimensões reduzidas de 4,1 x 2,4 cm.

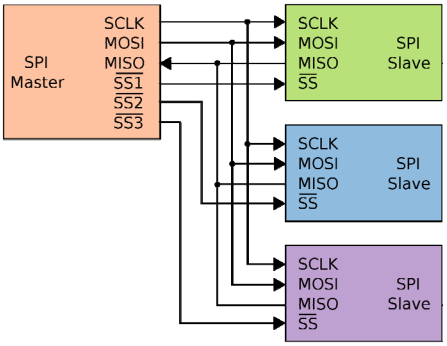



Figura 4.20 - Comunicação SPI.

O modo SPI foi inicialmente desenvolvido pela Motorola e tornou-se numa especificação de interface de comunicação série síncrona utlizado para comunicação a curta distância. O dispositivo *master* utiliza o relógio e dados em linhas separadas, com uma linha de seleção para identificar o dispositivo com o qual deseja comunicar.

### 4.4.5 Envio de dados

A Ethernet é o padrão mundial para a tecnologia de rede de área local LAN (*Local Area Network*). Não é de estranhar que quando se trata de redes sem fios seja amplamente utilizada a WLAN (*Wireless Local Area Network*). A Ethernet tornou-se numa parte integrante das redes de controlo e é adequada para a comunicação entre computadores pessoais, controladores programáveis ou outros sistemas de controlo sem fios. A tecnologia WiFi (*Wireless Fidelity*) permitiu ampliar o alcance da rede a funções remotas.

O sistema de envio de dados para o servidor apenas permite a comunicação através de Ethernet. Então, devido à necessidade de manter os níveis de consumo baixos, adequados a um sistema de transmissão em que se pretende um sistema portátil, eficiente e alimentado a energia solar, efetuou-se um estudo ao consumo dos módulos apresentados na Figura 4.21.

a) b) c)


Figura 4.21 - a) Ethernet W5100 shield. b) Ethernet W5100. c) Ethernet W5200 shield.

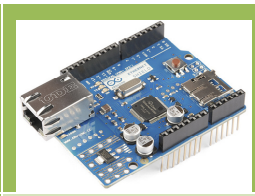
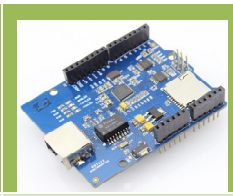
Na primeira fase do projeto começou-se por utilizar a *Shield* de Ethernet com o chip W5100 presente na Figura 4.21 a). Este componente traz a vantagem de já suportar o leitor de cartão micro SD, no entanto não foi possível desativar ou adormecer a parte de Ethernet do módulo.

Numa segunda fase optou-se por utilizar dois módulos, o módulo de armazenamento de dados, apresentado na Figura 4.19 a) em conjunto com o módulo de Ethernet apresentado na Figura 4.21 b). O módulo de armazenamento estaria sempre ativo enquanto o módulo de Ethernet seria desligado da alimentação através da ativação de um relé.

Finalmente utilizou-se o componente representado através da Figura 4.21 c), que apresenta uma versão diferente do módulo de Ethernet da Figura 4.21 a) que já permite desligar e adormecer, poupando significativamente no consumo do componente.

Estes módulos comunicam através de SPI e apresentam as características indicadas na Tabela 4.1.

Tabela 4.1 - Caraterísticas principais dos módulos de Ethernet.

| |  |  |  |
|---|---|---|---|
| **Alimentação** | 3,3 V ou 5 V | 5 V | 3,3 V ou 5 V |
| **Memória TCP/IP** | 16 Kbytes | 16 Kbytes | 32 Kbytes |
| **Comunicação** | 10/100 Mbs | 10/100 Mbs | 10/100 Mbs |
| **Com. simultânea** | 8 sockets indp. | 8 sockets indp. | 16 sockets indp. |
| **Consumo ativo** | 183 mA | 183 mA | 175 mA |
| **Consumo *Stand by*** | não permite | não permite | 4 mA |

Este estudo permitiu verificar qual o módulo a utilizar, dadas as restrições de consumo presentes na elaboração deste projeto. Pode-se observar na Tabela 4.1 que o módulo que permite uma melhor gestão de energia é o módulo de Ethernet W5200 que junta o baixo consumo a possibilidade de pôr o componente em *stand-by,* além de permitir um maior *buffer* de transmissão ou receção dos pacotes de comunicação.

Após concluída a escolha do componente procedeu-se a programação do microcontrolador do nó coordenador, para tal, seguiu-se o fluxograma da Figura 4.22 a). Nas Figuras 4.22 b), c) e d) observam-se os fluxogramas das funções auxiliares. Este fluxograma é representado no código do programa principal (Figura 4.25) através da função ***Web_SD_Send***.

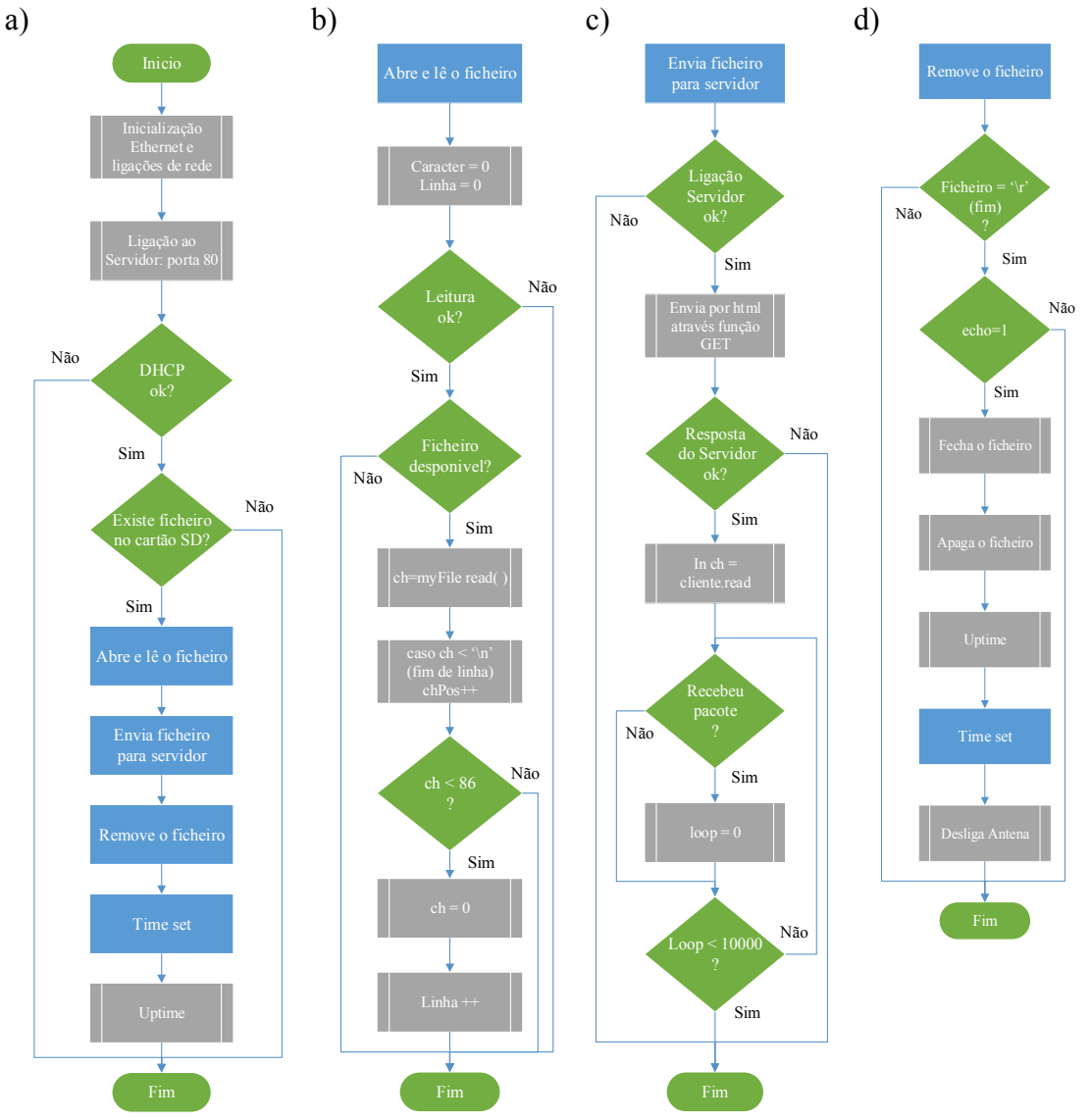



Figura 4.22 - Fluxograma da leitura e envio de dados do cartão SD.

O código começa por inicializar a biblioteca de Ethernet e as ligações de rede. Estabelece uma ligação ao servidor, através de IP (*Internet Protocol*) fixo pela porta 80 e verifica se obtém DHCP valido. Caso obtenha resposta afirmativa, verifica se existe algum ficheiro no cartão SD e lê. Envia a informação para o servidor, espera confirmação

que o servidor recebeu os dados e apaga o ficheiro. Finalmente aproveita a ligação para fazer uma atualização do relógio, verifica o tempo que as operações duraram, desliga do servidor e desliga o sistema de feixes hertzianos.

A função "Abre e lê o ficheiro" inicializa a zero os vetores referentes ao caracter e à linha. Verifica se a leitura é possível, caso não exista erro, lê os caracteres até encontrar o fim da linha ou enquanto o comprimento for inferior a 86, que neste caso, é o tamanho máximo da trama. Findo este processo reinicia a zero o vetor dos caracteres e incrementa para a próxima linha.

A função "Envia ficheiro para o servidor", em primeiro lugar, verifica se existe ligação ao servidor. Caso exista, envia o pacote contendo a informação e espera por resposta a confirmar a receção do envio. Caso não receba confirmação, entra em ciclo de espera durante um tempo determinado pela biblioteca de Ethernet.

A função "Remove o ficheiro" verifica se o ficheiro já chegou ao fim e, em caso afirmativo, verifica se a resposta do servidor é igual a 1. Este processo visa não apagar os dados do cartão em caso de erro no envio ou de receção incompleta. Após envio correto, fecha o ficheiro, apaga e determina o tempo decorrido. Aproveita a ligação para atualizar o relógio através da função "*Timeset*" (descrita anteriormente na secção 4.4.3) e finalmente procede ao encerramento do sistema de feixes hertzianos.

### 4.4.6 Protótipo desenvolvido

Na Figura 4.23 apresenta-se o circuito completo do nó coordenador.

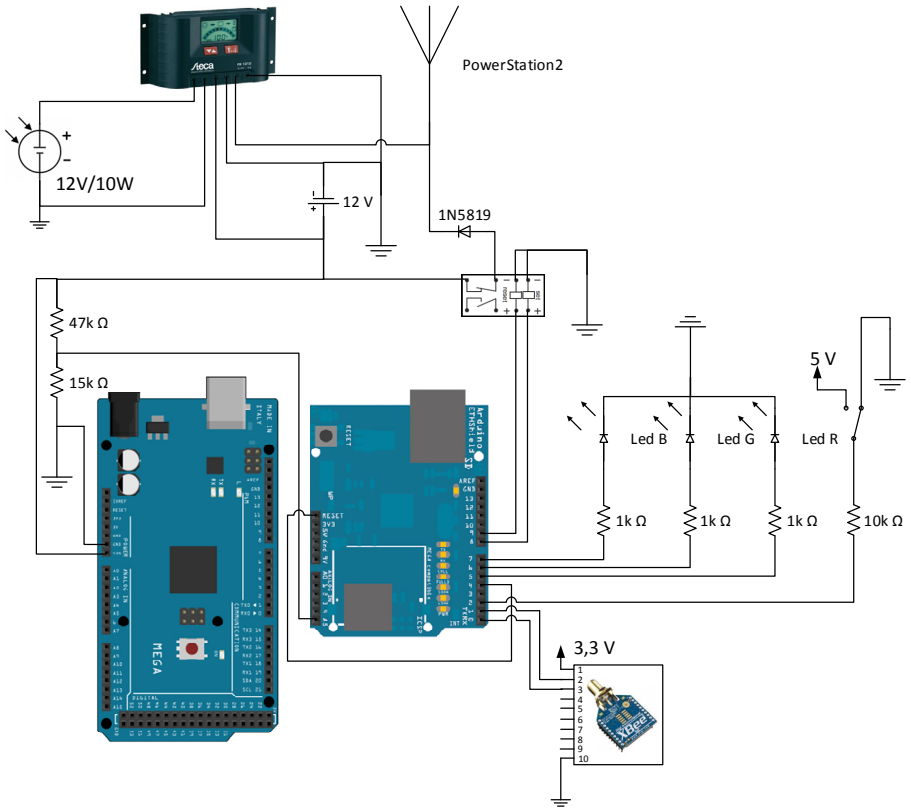



Figura 4.23 - Circuito do nó coordenador.

O circuito da figura anterior é constituído por um controlador de carga da bateria através de um painel solar, um microcontrolador arduíno Mega com uma *shield* de *Ethernet*, um circuito de *on/off* para ligar e desligar o sistema de feixes hertzianos, um módulo XBee e leds de aviso de estados.

Apresenta-se na Figura 4.24 o protótipo desenvolvido. Encontra-se dentro de uma caixa estanque para exteriores de 200x170x80 mm fabricada em ABS, com índice de proteção de IP65 e IK08 [44].

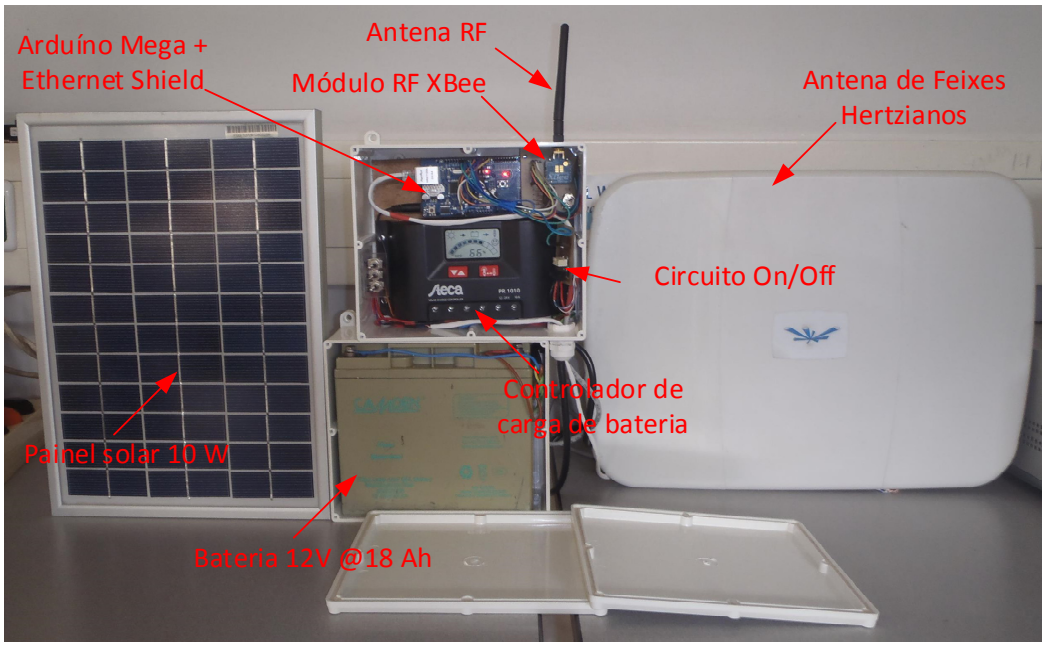



Figura 4.24 - Protótipo do nó coordenador desenvolvido.

Na Figura 4.24 pode-se observar os diversos componentes constituintes do nó coordenador. Foi desenvolvida uma placa de circuito impresso com recurso aos materiais presente no laboratório, que serve de suporte ao XBee e aos leds de aviso. O sistema tem três leds de aviso, correspondentes aos estados do código, com o led vermelho indicar erros, o azul pisca quando o protótipo estabelece ligação com o servidor e envia dados e finalmente o led verde pisca quando tudo está a funcionar corretamente.

Na Figura 4.25 representa-se o fluxograma do código completo desenvolvido para a programação do microcontrolador, que pode ser consultado no Anexo IV.

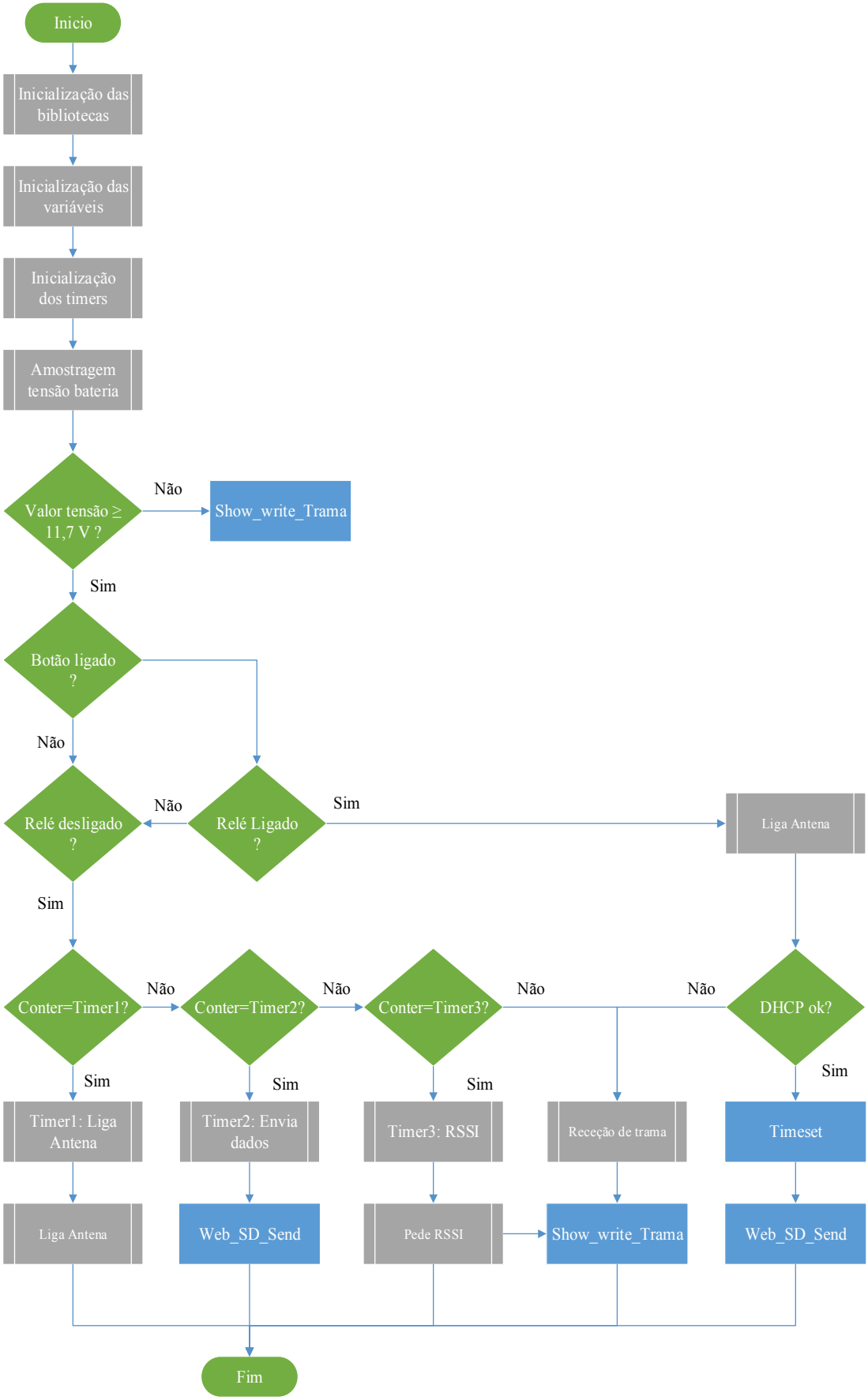



Figura 4.25 - Fluxograma do código desenvolvido para o nó coordenador.

Com a análise do fluxograma verifica-se que o código começa com a inicialização das bibliotecas, variáveis e timers. Logo de seguida é efetuada uma amostragem do valor de tensão presente na bateria, de modo a preservar a sua capacidade, caso o valor seja inferior a 11,7 V o algoritmo passa a executar a função ***Show_write_Trama*** descrita na secção 4.4.2.

Optou-se por incluir no protótipo um botão de acionamento manual, que permite ao utilizador suprimir o *timer* que controla o envio de dados, ou seja, permite utilizar o sistema de feixes hertzianos para ter acesso a internet. Quando o botão passa a posição contrária, o sistema reinicializa o *timer* 2 e retoma o funcionamento normal.

O *timer* 1 controla o temporizador responsável pelo acionamento do relé que permite ligar e desligar a antena e está programado para o ligar 2 minutos antes do envio de dados. Considerou-se este tempo depois de vários testes em laboratório que deram uma media de 1, 26 minutos, no entanto optou-se por juntar uma margem de segurança.

O *timer* 2 controla o temporizador responsável pelo envio de dados para o servidor (30 em 30 minutos), descrito anteriormente na secção 4.4.6 pela função ***Web_SD_Send***.

O *timer* 3 controla o temporizador responsável pelo envio da trama responsável pelo pedido de RSSI (*Received Signal Strenght Indication*) para a rede. Como este parâmetro não varia significativamente optou-se por configurar com um minuto de *offset* em relação ao *timer* 2.

## 4.5 Comunicação

Para uma rede sem fios comunicar corretamente é recomendável a existência de linha de vista entre os pontos terminais. Essa linha de vista deve incluir um espaço considerável livre de obstáculos como edifícios ou árvores, dentro do primeiro elipsoide de Fresnel. Os equipamentos têm de ser projetados de modo a cobrir a distância com a qualidade exigida. O diagrama de radiação da antena e ganho têm de ser cuidadosamente considerados e calculados.

A comunicação entre os nós sensores e o nó coordenador com funções de *gateway* foi implementada recorrendo a módulos com rádio frequência.

No sistema desenvolvido, o servidor onde é efetuada a manipulação dos dados pode encontra-se a uma grande distância da área de recolha. Para tal, torna-se necessário recorrer a uma ligação de maior alcance. Visando estabelecer a ligação entre o servidor e a *gateway* utiliza-se uma ligação de micro-ondas.

### 4.5.1 Antenas dos Nós

No sentido de dotar os nós de uma maior alcance de comunicação utilizaram-se antenas RPSMA W1030 da Pulse [46] representada na Figura 4.26.

a) b)

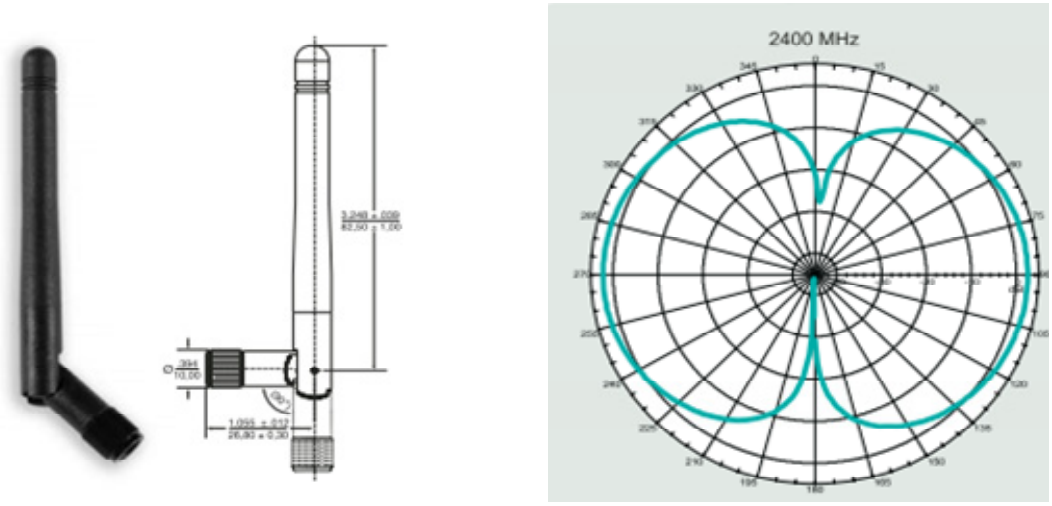



Figura 4.26 - a) Antena W1030. b) Diagrama de radiação com polarização vertical.

Estas antenas possuem um diagrama de radiação omnidirecional, operam na gama de frequências entre os 2,4 e 2,5 GHz, têm uma impedância nominal de 50 Ω e um ganho de 2 dBi. Estas antenas são apropriadas para utilização no exterior, mostrando-se adequadas para aplicações WLAN. A escolha desta antena deve-se ao baixo custo, ao tamanho e ganho, sendo adequadas a nós sensores compactos. É de referir que, caso seja necessário, a antena pode ser alterada de acordo com as necessidades de cada aplicação, tendo de seguir as recomendações da ANACON [47] que limita a potência total do nó a 10 mW (10 dBm).

### 4.5.2 Antena de feixes hertzianos

A comunicação de longa distância foi realizada com recurso ao sistema de feixes hertzianos Power Station 2, fabricado pela Ubiquiti Networks [48] e representado na Figura 4.27. Esta sistema funciona na frequência de 2,4 GHz e pode ser configurada como ponto de acesso ou *bridge*. Possui um rádio e *design* que permite estabelecer ligações ponto-a-ponto com distâncias até 50 km e apresenta as seguintes caraterísticas:

- Potência de transmissão: 26 dBm a 1 Mbit/s;
- Sensibilidade: -97 dBm;
- Ganho das antenas 17 dBi;
- Consumo: 1 A @ 12 V.

Figura 4.27 - Antena de feixes hertzianos Power Station 2.

No entanto, e devido às restrições de potência impostas pela entidade de regulamentação do espetro de rádio frequência, ANACOM, para o sistema a funcionar na banda dos 2,4 GHz só pode ter uma EIRP de 100 mW (20 dBm) para isenção de licença. O fabricante da antena, e de acordo com o país em questão, obriga na primeira utilização a concordar com os termos da licença, com se pode observar na Figura 4.28.

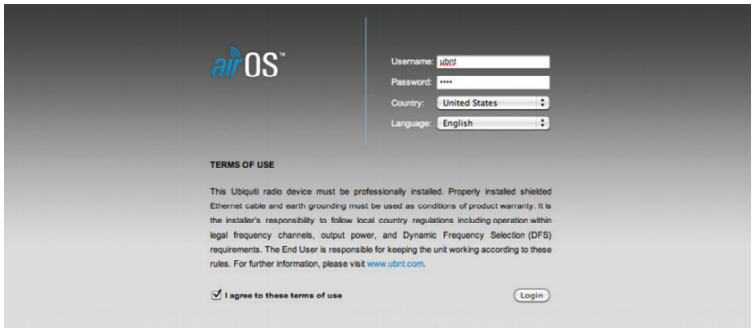



Figura 4.28 - Janela de primeira utilização da antena.

Visto que a antena possui um ganho de 17 dBi, ao escolher-se Portugal, sobra apenas 3 dBm de potência de transmissão, passando a ter as seguintes características:

- Potência de transmissão: 3 dBm a 1 Mbit/s;
- Sensibilidade: -97 dBm;
- Ganho das antenas 17 dBi;
- Consumo 545 mA @ 12 V

O consumo da antena foi determinado experimentalmente, efetuado-se vários testes de transmissão, chegando ao valor máximo de consumo de 545 mA a 3 dBm de potência.

Em modo de configuração é possível ligar ou desligar o acesso a transmissão de internet sem fios além de ser possível especificar a taxa de transmissão de dados, o tipo de polarização da antena e a qualidade de serviço esperada.

Os equipamentos WIFI precisam de ser configurados de modo a funcionarem corretamente, especificando o seu modo de funcionamento. O modo de *Bridge* especifica que a comunicação é feita ponto-a-ponto, em ambos os equipamentos. Assim, estes apenas podem comunicar entre si. Para a comunicação em multiponto, a unidade central é configurada em modo de ponto de acesso, (*Access Point*) enquanto os equipamentos satélite são configurados em modo de cliente ou estação.

Torna-se, depois, necessário determinar qual o equipamento que será a estação (*Station*) e qual será o ponto de acesso (*Access Point*). De um modo geral o ponto de acesso será localizado junto à parte da rede a que se pretende associar. Para uma configuração mais eficaz recomenda-se a programação e teste dos equipamentos em ambiente de laboratório, e só depois a montagem nos locais de destino.

O procedimento encontra-se dividido em 3 partes, sendo a primeira sobre a ligação à rede elétrica, a segunda sobre a configuração do computador e a terceira consiste na configuração dos equipamentos. Este procedimento pode ser encontrado no manual de instalação no Anexo VII.

## 4.6 Consumos e cálculos teóricos

Neste projeto escolheu-se a energia fotovoltaica como método de alimentação dos diversos componentes que constituem a parte do sistema remoto. Este poderá ser instalado em sítios isolados e sem recurso a energia elétrica. Devido a essa possibilidade, o sistema deve estar preparado para ter uma autonomia elevada e ser o mais portátil possível, de modo a não limitar a escolha do local de estudo.

O consumo do sistema torna-se num fator fundamental na escolha dos componentes constituintes do mesmo. Sabe-se que a energia solar é gratuita e na região autónoma da Madeira até existe em quantidades consideráveis, mas o sistema tem de estar preparado para funcionar, sem carregamento das baterias durante alguns dias, o que torna os consumos dos nós de extrema importância.

### 4.6.1 Nó sensor

Na Tabela 4.2 apresentam-se os consumos refentes ao valor teórico de consumo máximo dos componentes constituintes de um nó sensor.

Tabela 4.2 - Consumo do Nó Sensor.

| | Ativo | Em Espera | Adormecido |
|---|---|---|---|
| Atmega328P | 4 mA | 0,8 mA | 180 µA |
| Rádio XBee serie 2 | Tx - 40 mA<br>Rx - 40 mA | 15 mA | < 1 µA |
| Sensor SHT11 | 0,55 mA | | 0,3 mA |
| Circuito de transimpedância | 54 µA | | |
| Sensor pressão atmosférica | 5 µA | | 1 µA |
| **Total:** | **44,60 mA** | **15,8 mA** | **482 µA** |

Verifica-se através da tabela que o componente que apresenta maior consumo é o rádio XBee. Este, em modo ativo, consome 40 mA tanto na transmissão como na receção, tornando o adormecimento deste componente essencial para a preservação de energia na bateria.

O protótipo do nó sensor desenvolvido tem mais alguns componentes que não estão descritos na tabela de consumo devido a estarem apenas esporadicamente ativos, não constituindo um valor significativo de consumo.

Programou-se um tempo adormecimento de 10 s, ou seja, o nó de 10 em 10 s acorda, afere os valores dos sensores, envia os dados e volta a adormecer. Para obter os tempos dos diversos estados de consumo optou-se por inserir uma resistência de 1 Ω em série com o circuito. Utilizou-se uma bateria com a capacidade de 2000 mAh com uma tensão

nominal de 3,7 V (descrita na secção 4.7.1). Com a ajuda de um osciloscópio averiguaram-se os tempos e consumos mais significativos.

$$\text{Consumo médio} = \frac{(C_{ativo} \times T_{ativo} + C_{espera} \times T_{espera} + C_{adorm.} \times T_{adorm.})}{T_{total}} \quad (4.7)$$

Com base na equação anterior, para um tempo de adormecimento de 10 s chega-se a um consumo médio de 2,272 mA. A duração das baterias é dado por:

$$\text{Duração da bateria} = \frac{\text{Capacidade da bateria} \times \text{Coeficiente de segurança}}{\text{Consumo médio}} \quad (4.8)$$

$$= \frac{2000 \times 0{,}8}{2{,}272} = 704{,}23 \text{ horas} = 29{,}34 \text{ dias}$$

$$\text{Potência média consumida} = \text{Consumo médio} \times \text{Tensão nominal} \quad (4.9)$$

$$= 2{,}271 \text{ mA} \times 3{,}7 \text{ V} = 8{,}40 \text{ mW}$$

Verifica-se pela análise dos resultados obtidos que o nó está preparado para ficar 29,34 dias, ou seja, aproximadamente um mês sem recurso ao carregamento da bateria por energia solar, sendo adequado ao sistema projetado.

### 4.6.2 Nó coordenador

Os consumos teóricos, máximos, dos vários componentes que constituem o nó coordenador são importantes na projetação do sistema de alimentação. Este é responsável pelo funcionamento do sistema.

Tabela 4.3 - Consumo do nó coordenador.

| | Ativo | Em Espera | Adormecido |
|---|---|---|---|
| Atmega2560 | 14 mA | 4 mA | 220 µA |
| Rádio XBee serie 2 | Tx - 40 mA<br>Rx - 40 mA | 15 mA | < 1 µA |
| Ethernet W5200 shield | 175 mA | | 4 mA |
| Sistema UBNT | 1 A | | |
| Controlador de carga | 12,5 mA | | |
| **Total:** | **1,24 A** | **19 mA** | **4,22 mA** |

Verifica-se através da tabela que o componente que apresenta maior consumo é o rádio de feixes hertzianos. Este, segundo a folha de características [49], em modo ativo apresenta um valor máximo teórico de 1 A na transmissão e na receção. No entanto, e devido as limitações de potência impostas pela ANACOM [47], verificou-se

experimentalmente que o consumo médio da antena de feixes hertzianos, em modo ativo, para uma potência de 3 dB é de 545 mA.

Para se considerar um sistema de carregamento através de energia solar neste projeto teve-se de dimensionar o painel com dimensões reduzidas tendo-se optado pelo envio de dados de 30 em 30 minutos. Este período de transmissão reduz significativamente o consumo de energia, uma vez que o painel e o módulo de Ethernet apenas irá consumir energia durante o tempo de envio, estando o restante tempo desligado.

Para os cálculos do nó coordenador utilizou-se o valor de consumo médio do sistema de feixes hertzianos e considerou-se que tanto o microcontrolador como o XBee estão sempre em modo ativo. Utilizou-se uma bateria de 18 Ah com tensão nominal de 12 V (descrita na secção 4.7.2) e um envio de dados a cada 30 minutos.

Determinou-se experimentalmente que o tempo de ativação do sistema e envio de dados está estimado em 2 minutos. É neste tempo que é feita a ligação do sistema de feixes hertzianos e aquisição de DHCP para a transmissão.

Na primeira hora de funcionamento tem-se três períodos de dois minutos. A primeira coisa que o sistema faz após ligar é tentar adquirir DHCP, atualizar o relógio e enviar os dados para a base de dados. Caso exista algum erro, o sistema está programado para reiniciar evitando perdas de dados. Então o seu consumo é dado por:

$$\text{Consumo} = \frac{C_{S.\ feixes\ hertziano} \times T_{ativo} + C_{no\ coordenador} \times T_{no\ coordenador}}{T_{total}} \qquad (4.10)$$

$$\text{Consumo 1ª hora} = \frac{786{,}5\ \text{mA} \times 360\ \text{s} + 70{,}5\ \text{mA} \times 3240\ \text{s}}{3600\ \text{s}} = 142{,}10\ \text{mA}$$

Na segunda hora e restantes:

$$\text{Consumo} = \frac{786{,}5\ \text{mA} \times 240\ \text{s} + 70{,}5\ \text{mA} \times 3360\ \text{s}}{3600\ \text{s}} = 118{,}23\ \text{mA}$$

$$\text{Consumo médio} = \frac{142{,}10\ \text{mA} + 118{,}23\ \text{mA} \times 23}{24} = 119{,}23\ \text{mA}$$

Utilizando-se as equações (4.8) e (4.9) verificou-se a duração da bateria considerada e a potência média consumida pelo nó coordenador juntamente com o sistema de feixes hertzianos é dado por:

$$\text{Duração da bateria} = \frac{18000 \times 0{,}8}{119{,}23} = 120{,}77\ \text{horas} = 5{,}03\ \text{dias}$$

sendo a potência consumida de

$$\text{Potência média consumida} = 119{,}23\ \text{mA} \times 12\ \text{V} = 1{,}43\ \text{W}$$

A duração da bateria, apesar de não ser a ideal, verifica-se ser adequada a aplicação em questão devido a necessidade de portabilidade do sistema. Caso se verifique a necessidade de aumentar o período de duração sem carregamento do sistema, recomenda-se o aumento da capacidade da bateria ou o tempo entre envio de dados.

De modo a garantir um bom funcionamento programou-se o microcontrolador para fazer uma amostragem do valor de tensão na bateria, com recurso a um divisor de tensão, e quando o valor baixa dos 11,7 V o sistema apenas armazena os dados e volta a enviar quando o valor de tensão na bateria for superior. Utilizou-se este valor de tensão de referência, uma vez que o controlador de carga (descrito na secção 4.7.2) entra em modo de pré-aviso, desligando a tensão aos 11,1 V e apenas voltando a ligar aos 12,6 V.

## 4.7 Alimentação

Um aspeto fundamental a ter em consideração na implementação de uma rede de sensores sem fios é a alimentação dos dispositivos eletrónicos. Neste projeto, em particular, trata-se de um sistema de transmissão de dados de uma aplicação remota, que pode estar implementada em lugares sem acesso à rede de energia elétrica, então deve-se considerar energias renováveis, nomeadamente a solar.

Os sistemas solares têm uma fácil instalação e são indicados no caso de existir energia suficiente na zona de implementação do projeto.

### 4.7.1 Alimentação do nó sensor

No nó sensor, a alimentação é efetuada através de uma bateria de polímeros de iões de lítio apresentada na Figura 4.29. Trata-se de uma célula muito leve e fina, com uma capacidade de 2000 mAh e uma tensão nominal de 3,7 V. Está equipada com um conector JSP – PH padrão de dois pinos e é fabricada com proteção contra sobretensão, sobrecarga de corrente e tensão mínima [50].

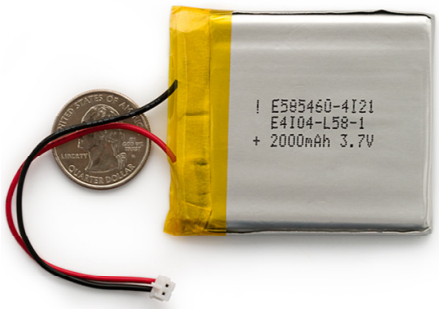



Figura 4.29 - Bateria de iões de lítio.

O painel solar utilizado neste projeto foi o 750-0030, apresentado na Figura 4.30. Este painel possui as seguintes características [51].

- Potência máxima: 1 W;

- Tensão na potência máxima: 6 V;
- Corrente na potência máxima: 167 mA;
- Tensão em circuito aberto: 4,2 V;
- Corrente em curto-circuito: 183 mA;
- Dimensões: 125 x 63 x 3,4 mm.

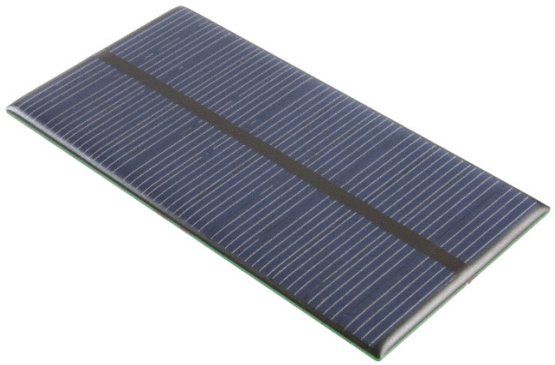


Figura 4.30 - Painel solar de alimentação do nó terminal.

O controlo de carga do sistema fotovoltaico é realizado através do controlador de carga MAX1555 [52], presente no próprio arduíno fio. Este controlador de carga, representado na Figura 4.31, é otimizado para carregar uma bateria de iões de lítio.

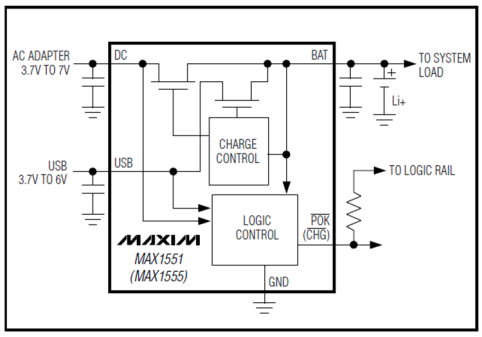



Figura 4.31 - Diagrama do controlador MAX1555.

Este controlador aceita tensão de operação até os 7 V. O *chip* dispõe de um controlo térmico, assim quando os limites térmicos são atingidos, o carregador não desliga, mas reduz progressivamente a corrente de carga.

Para utilizar o painel solar descrito anteriormente, foi necessário incluir um díodo 1N4001. Este díodo tem duas funções, sendo a primeira limitar a tensão que chega ao controlador, uma vez que o painel em circuito aberto pode atingir 7,2 V e o controlador apenas admite 7 V, e a segunda é limitar o sentido da corrente, como pode ser observado na Figura 4.32.

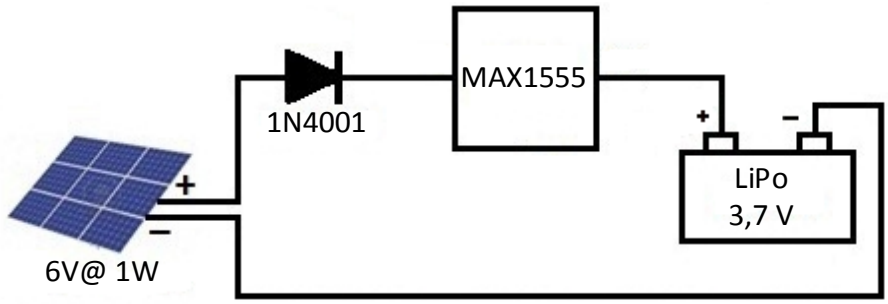



Figura 4.32 - Circuito de carga da bateria.

Este controlador, apesar de não ser o mais eficiente no que se refere ao sistema de controlo de carregamento da bateria, é adequado visto que os níveis de corrente consumido pelo nó sensor são bastante reduzidos quando comparados com a capacidade de carga do painel solar.

### 4.7.2 Alimentação nó Coordenador

A alimentação do nó coordenador, juntamente com o sistema de feixes hertzianos foi efetuada com recurso a bateria representada na Figura 4.33.


Figura 4.33 - Bateria de gel 12V 18 Ah.

A bateria representada é de chumbo-ácido BEV120180, selada e livre de manutenção com a capacidade de 18 Ah e uma tensão nominal de 12 V. A sua construção gelatinosa oferece um excelente desempenho, pode ser montada independentemente da posição e elimina a necessidade de adicionar água. Possui uma válvula interna de pressão máxima de 2,5 psi [53].

O painel solar utilizado neste projeto foi o FVG 10P da FVG Energy [54], apresentado pela Figura 4.34, que possui as seguintes características:

- Potência máxima: 10 W;
- Tensão na potência máxima: 17,5 V;
- Corrente na potência máxima: 570 mA;
- Tensão em circuito aberto: 12,8 V
- Corrente em curto-circuito: 660 mA;

- Dimensões: 380 x 270 x 28 mm.

Figura 4.34 - Painel solar de alimentação do nó coordenador.

Os painéis solares produzem corrente em resultado da incidência direta da luz solar. Os painéis têm duas zonas de funcionamento, uma correspondente a uma fonte de corrente e uma outra correspondente a uma fonte de tensão, daí surgindo a necessidade de controlo.

O elemento de controlo central em sistemas fotovoltaicos é o sistema de carga solar, apresentado na Figura 4.35, que tem o propósito de controlar o fluxo de energia, assegurando simultaneamente a preservação da bateria.

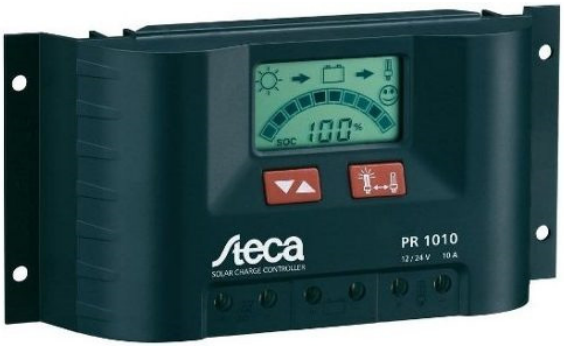



Figura 4.35 - Controlador de carga solar Steca.

Escolheu-se, então, o controlador de carga solar da *Steca* [55]. Este controlador apresenta-se adequado para baterias recarregáveis de 12 ou 24 V e deve ser montado num local protegido das influências meteorológicas.

Este regulador está munido de vários dispositivos de proteção do seu sistema, da bateria e da carga. Destes dispositivos de proteção destacam-se as proteções contra polaridade invertida, curto-circuito, funcionamento em vazio, temperatura excessiva e proteção contra descarga ou sobrecarga da bateria.

Para a conexão do regulador deve-se seguir a seguinte ordem (positivo e negativo):

- Conectar a bateria ao regulador de carga;
- Conectar o módulo fotovoltaico ao regulador de carga;
- Conectar a carga ao regulador.

Em caso de desinstalação o processo deve ser seguindo por ordem inversa, caso contrário o ajuste automático não funcionará. O regulador controla os parâmetros de tensão e corrente, através dos quais calcula o estado de carga da bateria SOC (*State of Charge*). O SOC é responsável pela seleção do modo de carga e proteção contra descarga profunda, no entanto, os limites de proteção já vêm predeterminados de fábrica.

O regulador efetua a carga da bateria utilizando uma tensão constante. Utiliza toda a corrente disponibilizada pela fonte para carregar a bateria até ser atingida a tensão de limite de carga.

A regulação da corrente de carga é efetuada através da modulação da largura de impulso PWM da corrente de entrada.

A tensão nominal pode ser de 12 V ou 24 V sendo o reconhecimento automático. A 12 V a zona de tensão vai dos 6,9 V a 17,2 V. Apresenta um consumo de 12,5 mA e uma frequência PWM de 30 Hz, permitindo uma corrente de carga a 25°C de 10 A.

Na Figura 4.36 observa-se o sistema completo de carregamento do nó coordenador, em que o controlador de carga é responsável pelo carregamento da bateria, sendo a gestão de energia é efetuada pelo microcontrolador.

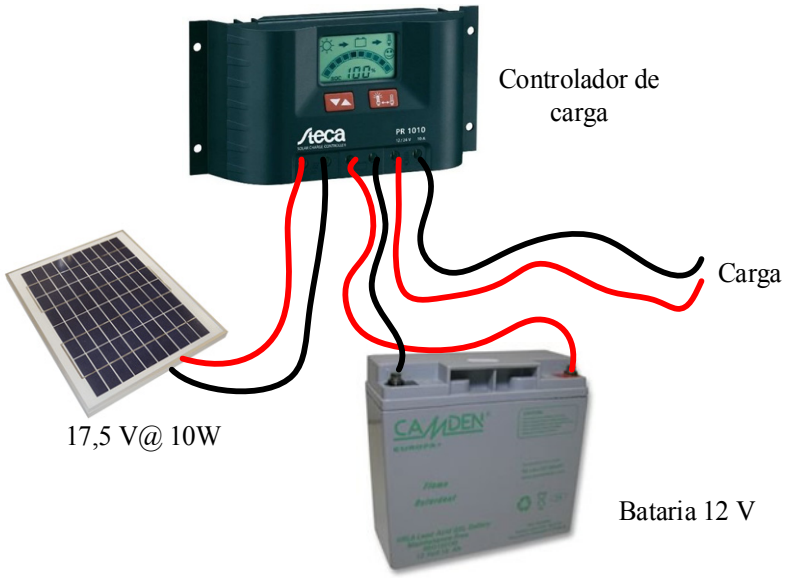



Figura 4.36 - Sistema de carregamento do nó coordenador

## 4.8 Plataforma de visualização de dados

A apresentação dos dados recolhidos pelo sistema de transmissão foi também um dos objetivos deste projeto. Para tal utilizou-se a aplicação de controlo e monitorização baseada numa interface *Web*, desenvolvida pelo Eng. Filipe Santos no âmbito de uma unidade curricular.

A utilização da plataforma torna-se possível com a instalação do pacote de *software* XAMPP, senso este pacote de software gratuito e encontrando-se em [56]. O XAMPP

contém um servidor de *web Apache* independente e de plataforma livre, uma base de dados em MySQL e as linguagens de programação *PHP* ou *Perl*. A Figura 4.37 apresenta o fluxograma de funcionamento da plataforma de visualização de dados.

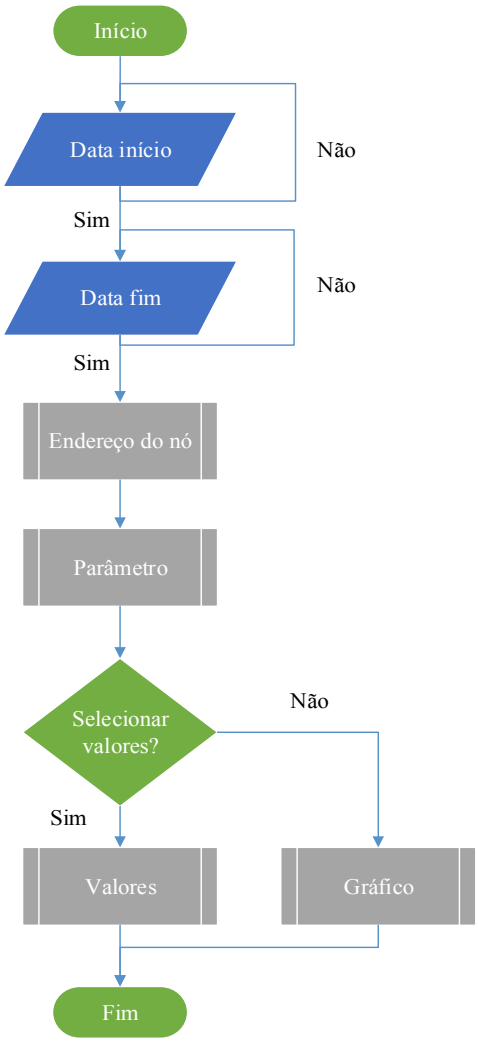



Figura 4.37 - Fluxograma da plataforma de visualização de dados.

Para utilizar a aplicação torna-se necessário em primeiro lugar, definir a data de início e fim da amostragem de dados. Esta inicialização pode ser feita através da introdução das datas ou pode-se utilizar o calendário disponível. Após a inicialização, seleciona-se o nó e o parâmetro pretendido. Estes podem ser visualizados graficamente ou através dos seus valores.

Devido à programação da aplicação, foram implementadas várias modificações no código PHP (*Hypertext Preprocessor*), porque a plataforma apenas encontra-se preparada para ter o XBee do nó coordenador ligado ao servidor. No entanto, no sistema desenvolvido, a recolha de dados é efetuada e armazenada pelo nó coordenador, que pode estar a uma distância de alguns quilómetros do servidor.

Os pacotes de dados são recolhidos de 30 em 30 minutos de forma automática e contêm as informações recolhidas pelos nós sensores de 10 em 10 s. A marca temporal é

adicionada na trama quando é escrita no cartão SD e, sendo assim, também tem de ser alterada na plataforma de visualização de dados.

O código em PHP desenvolvido pode ser consultado no Anexo VIII, com as alterações mais significativas descritas no fluxograma da Figura 4.38.

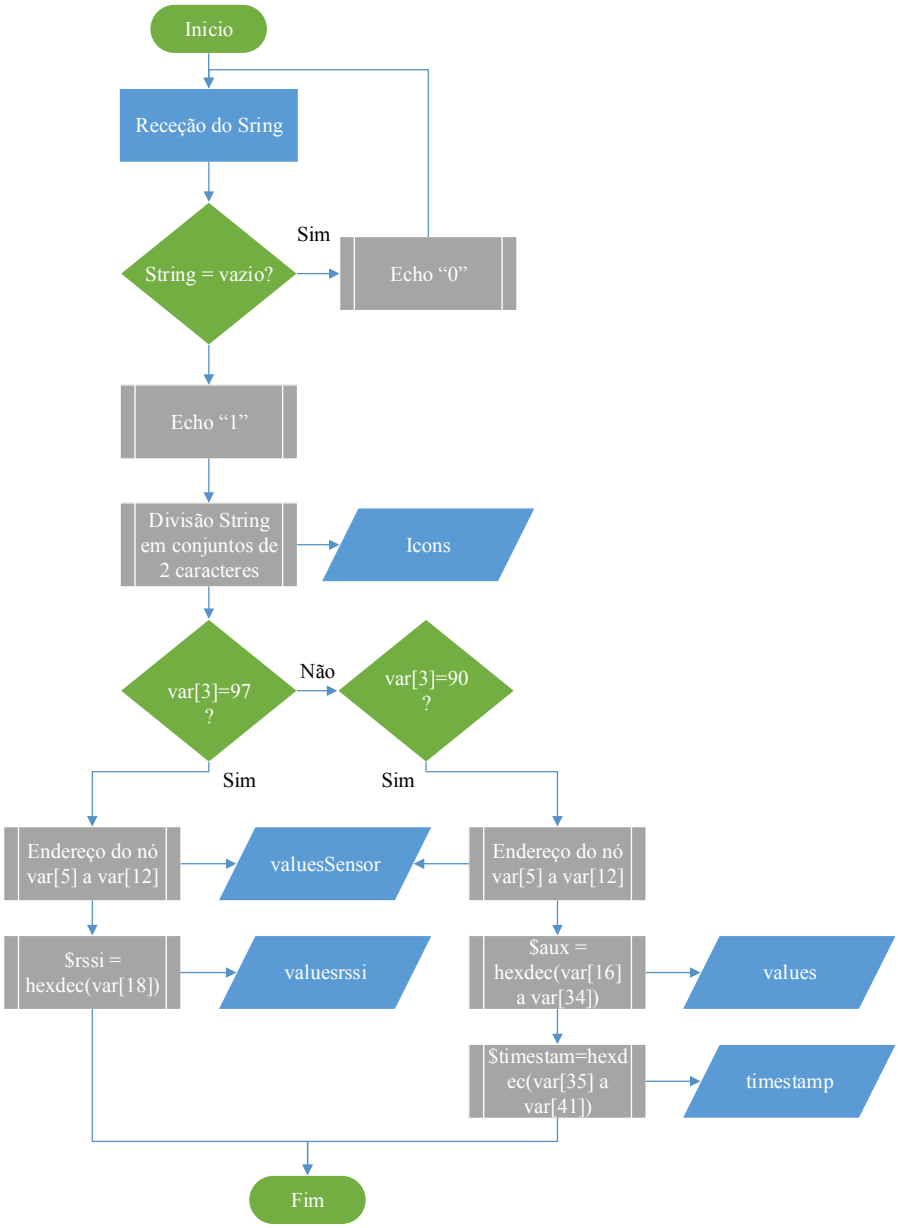



Figura 4.38 - Fluxograma das alterações a base de dados.

Analisando o fluxograma, verifica-se que a trama com os dados é recebida pela plataforma como uma *String* de valores. A primeira verificação é feita de modo a garantir que os dados não sejam perdidos, ou seja, se a *String* for diferente de vazio é enviado para o nó coordenador o valor "1" e nesse caso o nó envia a *String* seguinte. Após uma receção válida, o primeiro passo consiste em dividir a *String* num conjunto de dois caracteres uma vez que o servidor está preparado para receber bytes em hexadecimal. Depois da receção, a trama completa é armazenada na tabela *Icons*, é efetuada uma verificação para

determinar o tipo de trama, sendo armazenado o *mac address* do nó de envio no campo *valuesSensor*, os valores de RSSI no campo *valuesrssi*, os dados recolhidos no campo *values*, e finalmente é convertido e armazenado os valores de *timestamp*, ou seja, a data e hora em que o sensor adquiriu os dados enviados.

O código em linguagem PHP foi desenvolvido com recurso ao programa gratuito Notepad++ que se encontra em [57].

Na aplicação desenvolvida, representada na Figura 4.39, pode-se observar os dados recolhidos de tensão no painel solar, tensão na bateria, temperatura, humidade relativa, luminosidade, pressão atmosférica, RSSI e o estado do sistema de interrupção.


Figura 4.39 - Plataforma de visualização de dados.

Observa-se nesta figura os valores de temperatura do sensor SHT11 e, como foi descrito anteriormente, pode-se verificar na parte superior direita, os dados representados graficamente. Na parte inferior tem-se os mesmos dados de temperatura mas de forma numérica.

## 4.9 Conclusões de capítulo

Neste capítulo foram apresentados todos os componentes constituintes do protótipo, juntamente com os fluxogramas de funcionamento e programação efetuada.

Começou-se por descrever os componentes referentes ao nó sensor. Foram explicados o modo de funcionamento dos sensores de temperatura e humidade,

luminosidade e pressão atmosférica. Referiu-se como é feito o controlo de tensão na bateria para o carregamento através do painel solar. Descreveu-se o circuito e a programação do microcontrolador.

No nó coordenador foram explicados o funcionamento dos componentes constituintes, o modo de aquisição e armazenamento dos dados provenientes do nó sensor, a aquisição da marca temporal, o controlo de carga da bateria e a sua programação. Desenvolveu-se um estudo comparativo entre módulos com cartão SD com vista a escolha do conjunto com menor consumo.

Calculou-se o consumo médio, juntamente com a duração das bateria sem carregamento solar, tanto no nó sensor como no nó coordenador.

Explicou-se os modos de comunicação e as antenas utlizadas nos nós e na transmissão de longa distância através de um sistema de feixes hertzianos.

Finalmente explicou-se o método de envio para o servidor, conjuntamente com a programação e a visualização dos dados através de uma plataforma *on-line*.

# 5. Testes e Resultados

Neste capítulo apresentam-se os testes e resultados obtidos no sistema de transmissão de dados de uma aplicação remota.

Começa-se por explicar as considerações a seguir num projeto de ligação de um sistema de feixes hertzianos. Depois são definidos os locais onde se efetuou os testes ao sistema proposto.

Finalmente são registados e analisados os resultados obtidos, comparados com os resultados simulados e calculados teoricamente.

## 5.1 Projeto de ligação do sistema de feixes hertzianos

No capítulo 4, secção 4.1.2, definiu-se a arquitetura do sistema proposto. Para tal, foi criada uma rede de sensores sem fios, com um coordenador responsável pela gestão do sistema. Uma das vantagens de utilizar sistemas de sensores sem fios juntamente com um sistema de transmissão por feixes hertzianos, é a possibilidade de estender facilmente a rede.

Os nós sensores, além de servirem de monotorização de parâmetros, também podem ser utlizados como *routers* para retransmitir informação. Os sistemas de feixes hertzianos, normalmente, têm de ser situados em sítios com linha de vista, o que nem sempre se torna possível. Utilizando-se as capacidades das redes de sensores sem fios é possível deslocar o sítio da recolha de informação para um sítio onde seja possível uma transmissão por feixes hertzianos.

Os nós sensores adquirem os dados de 10 em 10 segundos e enviam através da rede para o nó coordenador. O sistema desenvolvido na primeira ligação adquire o tempo de relógio através de um servidor NTP e envia os dados armazenados. Após este procedimento, volta a recolher os dados provenientes do nó sensor, processa e armazena num cartão SD. Quando o seu relógio atinge os 30 minutos, faz nova tentativa de envio dos dados para o servidor e atualiza o relógio, repetindo o processo de 30 em 30 minutos.

Referiu-se no capítulo 2 secção 2.7, os passos a seguir num projeto de uma ligação de feixes hertzianos. Estes passos seguem o fluxograma da Figura 5.1 em que o primeiro passo foi definir a quantidade de trafego a transmitir.

O *firmware* da antena de feixes hertzianos permite ao utilizador definir a taxa de transmissão mínima de dados pretendidos, apenas variando a sensibilidade do sistema. Como se pretende utilizar o acesso a internet, fixou-se uma taxa de transmissão de 6 Mbps com sensibilidade do sistema de -94 dBm $\pm$ 1 dB. No entanto, é de referir que o sistema permite velocidades maiores caso a ligação o permita, num máximo de 54 Mbps.

O tráfego no acesso ao servidor NTP ou de envio de dados é bastante reduzido, sendo a sua única limitação a velocidade de envio dos dados. Pretende-se um envio rápido, uma vez que o sistema de transmissão apenas desliga após o envio completo dos dados ou por "*time out*" do sistema.

Na Figura 5.1 apresenta-se os passos principais no projeto de uma ligação por feixes hertzianos.

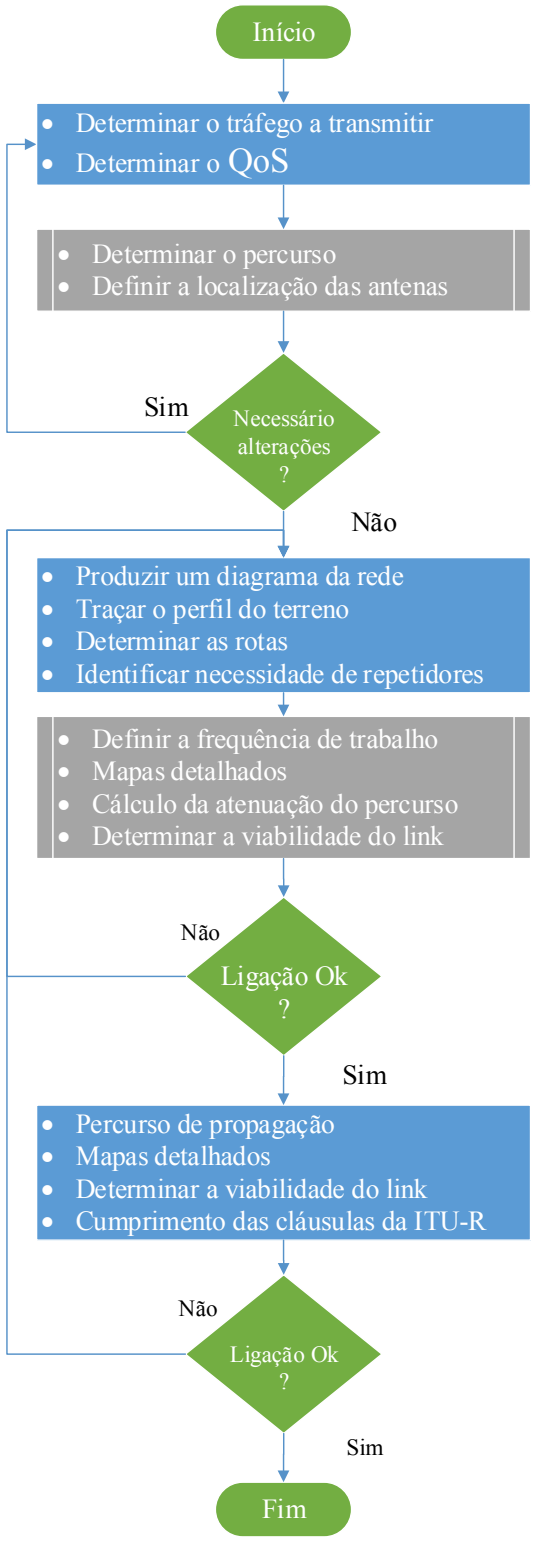



Figura 5.1 - Fluxograma de um projeto de ligação de feixes hertzianos.

No entanto, o sistema proposto já traz algumas limitações devido ao tipo de antenas utlizado. A frequência encontra-se limitada a gama dos 2,4 GHz, mas é possível definir o canal e largura de banda a utilizar. O *software* da antena possibilita o varrimento do

espetro ISM e assim verificar quais os pontos de acesso e respetivas frequências até à antena de transmissão. Escolheu-se então, de modo a minimizar as interferências e facilitar as medições, a frequência dos 2452 MHz o que corresponde ao canal 9 e uma largura de banda de 20 MHz.

## 5.2 Localização das antenas

A escolha do percurso é determinante para uma ligação por feixes hertzianos. O primeiro passo passa por determinar os pontos entre os quais se pretende estabelecer a ligação. Depois, utilizando-se cartas militares, mapas detalhados ou *software* específico, escolhe-se a melhor localização para as antenas do percurso de propagação.

Neste caso, um dos pontos é o laboratório de telecomunicações da Universidade da Madeira, onde está localizado o servidor com acesso à internet. Para validar o sistema desenvolvido, foi necessário realizar testes ao protótipo, tendo-se instalado de forma a ter uma linha de vista e num local seguro. Assim recorreu-se ao último piso (4) da Universidade da Madeira, para a instalação da antena base da ligação por feixes hertzianos, tendo a vantagem de ser um espaço com uma altura considerável e de acesso reservado. Observa-se o local de montagem na Figura 5.2.

Figura 5.2 - Instalação da antena de feixes hertzianos no edifício da UMa.

A comunicação com o laboratório de telecomunicações, que se situa no piso -2, foi possível com a passagem, de um cabo de rede UTP (*Unshielded Twisted Pair*) de categoria 5. Este cabo também serve de alimentação à antena através de POE (*Power Over Ethernet*). O esquema da ligação pode ser consultado na Figura 5.3 a) enquanto a Figura 5.3 b) apresenta o modo de ligação e cravagem das fichas de rede com POE.

a) b)

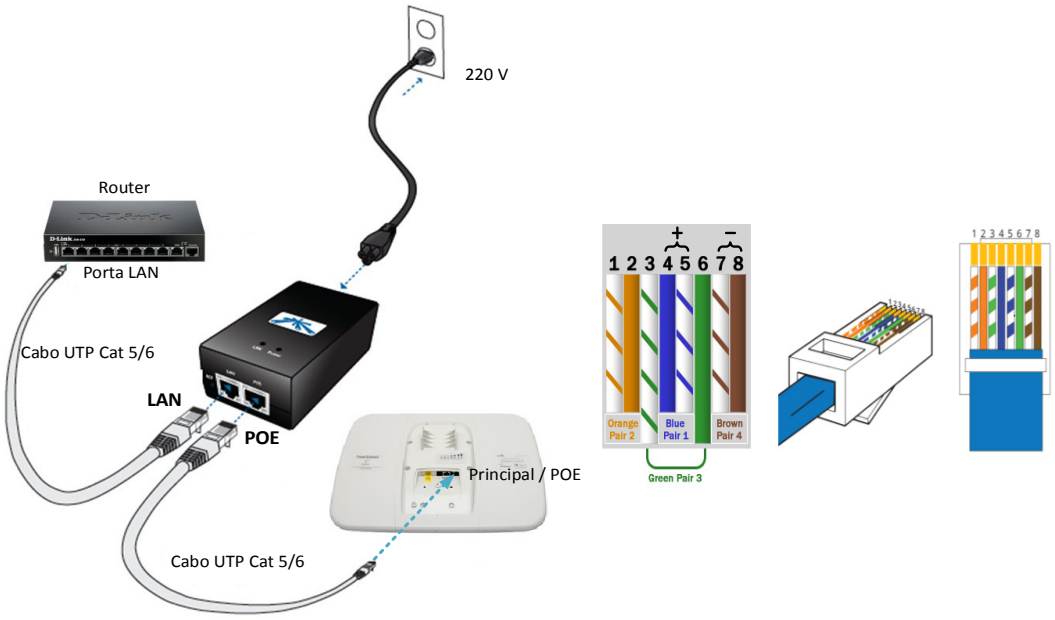



Figura 5.3 - a) Esquemático de ligação da antena. b) Ficha de rede UTP com POE.

O injetor POE usa um cabo de rede CAT 5 entre o *router* do laboratório com acesso à internet e o injetor. A alimentação é efetuada com recurso à rede elétrica, enquanto a alimentação da antena situada no terraço é feita com 15 V @ 0,8 A. Nos pinos 4 e 5 do cabo UTP faz-se a alimentação Vdc, reservando os pinos 7 e 8 para negativo. Os restantes pins servem para estabelecer a comunicação de dados.

As caraterísticas de localização da antena de transmissão estão na Tabela 5.1, observa-se que a cota do terreno onde se situa a Universidade da Madeira tem 154,1 m e depois considerou-se 3 m por piso e 1 m de poste onde está fixa a antena, chegando-se a um total de 25 m.

Tabela 5.1 - Caraterísticas da antena de transmissão.

| Latitude | Longitude | Elevação | Altura da antena |
|---|---|---|---|
| 32,658664° | -16,924117° | 154,1 m | 25,0 m |

A Universidade da Madeira fica localizada dentro de um vale, o que torna a escolha dos locais de transmissão mais complexo, no entanto, a antena fica localizada no terraço do edifício, o que é uma mais-valia.

Na Tabela 5.2 observa-se as coordenadas das localizações escolhidas para se efetuar os testes ao sistema desenvolvido. A estação base localiza-se no terraço da Universidade da Madeira, depois escolheram-se mais três localizações com características diferentes, duas das quais com linha de vista (LOS) e uma sem linha de vista (NLOS - *Non Line Of Sight*).

Tabela 5.2 - Coordenadas de teste do sistema.

| Localização | Latitude | Longitude | Distância (m) |
|---|---|---|---|
| Universidade da Madeira | 32º 39’ 31,19’’ | 016º 55’ 26,82’’ | 0 |
| Serras de São Roque (LOS) | 32º 42’ 06,25’’ | 016º 56’ 14,04’’ | 4944 |
| Chão da Lagoa (LOS) | 32º 42’ 24,44’’ | 016º 54’ 49,14’’ | 5440 |
| Chão da Lagoa (NLOS) | 32º 42’ 38,32’’ | 016º 54’ 54,15’’ | 5842 |

O primeiro percurso encontra-se nas serras de São Roque, tendo-se escolhido esta localização devido à linha de vista entre a Universidade da Madeira e devido a ser um percurso com interferência no primeiro elipsoide de Fresnel. O segundo percurso foi definido na zona do parque natural da Madeira, mais precisamente no Chão da Lagoa com linha de vista e sem grandes obstáculos no percurso de propagação. O terceiro percurso também se localiza no Chão da Lagoa, mas neste caso optou-se por uma zona sem linha de vista, mas mais elevado em relação aos restantes percursos.

Na Figura 5.4 representam-se os três percursos escolhidos. Introduziram-se as coordenadas geográficas no programa *Google Earth*, que pode ser transferido em [57], e optou-se por traçar, desde o ponto base até aos três percursos escolhidos, retas de modo a se visualizar melhor os caminhos de propagação escolhidos.

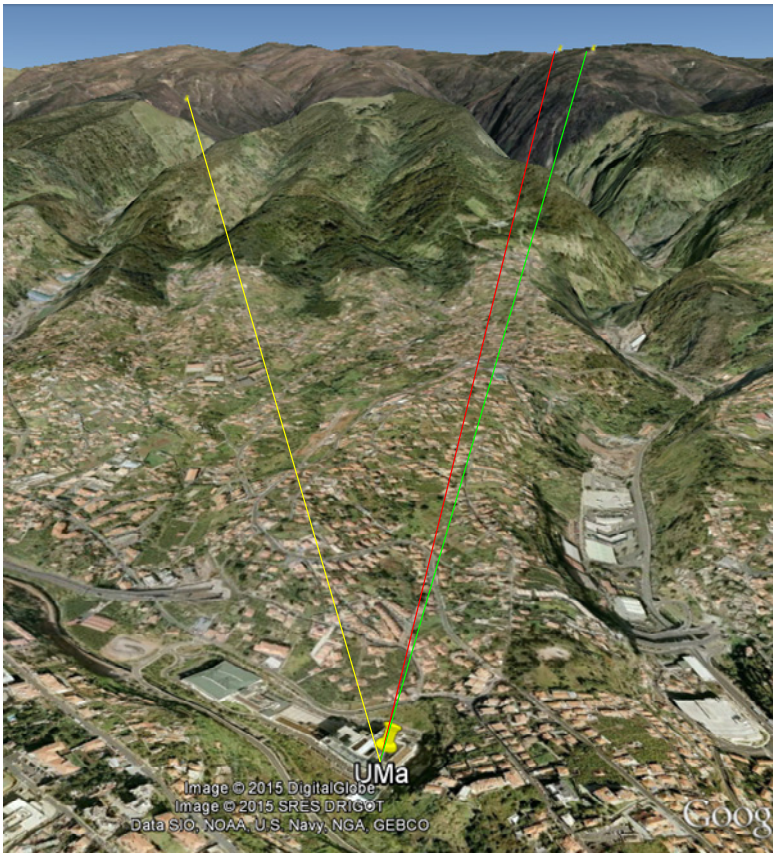



Figura 5.4 - Localização dos sítios de teste do sistema.

A amarelo representou-se o percurso com linha de vista entre a Universidade da Madeira e as serras de São Roque, a verde encontra-se o percurso com linha de vista entre a Universidade da Madeira e o Chão da Lagoa e finalmente a vermelho observa-se parte do percurso entre a Universidade da Madeira e a zona escolhida, também no Chão da Lagoa, mas sem linha de vista. A escolha destes locais como teste deve-se à necessidade de encontrar os limites de funcionamento do sistema de transmissão por feixes hertzianos desenvolvido.

Na Figura 5.5 representou-se o perfil de elevação do primeiro percurso, verificando-se uma elevação máxima de 1112 metros e um percurso de 5,29 km, sendo o percurso direto de 4,944 km, como foi descrito na Tabela 5.2.


Figura 5.5 - Perfil de elevação do percurso UMa - Serras de São Roque.

Na Figura 5.6 representa-se o perfil de elevação do segundo percurso, com uma elevação máxima de 1384 m e um percurso de 5,6 km, com percurso direto de 5,44 km.

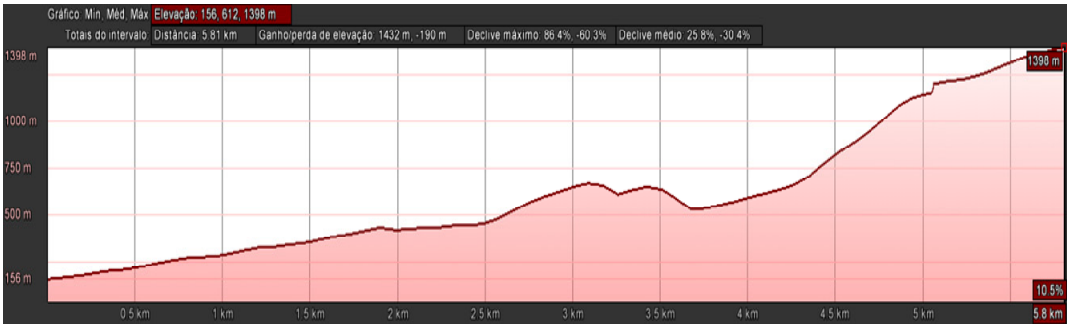



Figura 5.6 - Perfil de elevação do percurso UMa – Chão da Lagoa (LOS)

Na Figura 5.7 representa-se o perfil de elevação do terceiro percurso, com uma elevação máxima de 1442 m e um percurso de 6,22 km, com percurso direto de 5,842 km.


Figura 5.7 - Perfil de elevação do percurso UMa – Chão da Lagoa (NLOS).

Escolhidas as zonas de teste, tornou-se necessário verificar se é possível uma comunicação entre os pontos de emissão e receção. O primeiro passo a considerar-se foi o cálculo das potências e perdas teóricas esperadas.

Conhece-se a potência máxima de emissão (3 dBm), o ganho das antenas utilizadas (17 dBi) e o valor das perdas de propagação teóricas, assim torna-se necessário obter o valor de potência recebida.

Esta quantidade de sinal tem de estar acima da sensibilidade do rádio (-94 dBm) e deve ser mantida de maneira a manter o canal de comunicação estável, mesmo durante a presença de más condições atmosféricas.

## 5.3 Cálculo teórico da ligação

O cálculo do valor esperado de potência de sinal recebido é um dos parâmetros mais importantes num projeto de ligação por feixes hertzianos. A Figura 5.8 apresenta uma representação gráfica dos parâmetros de interesse.

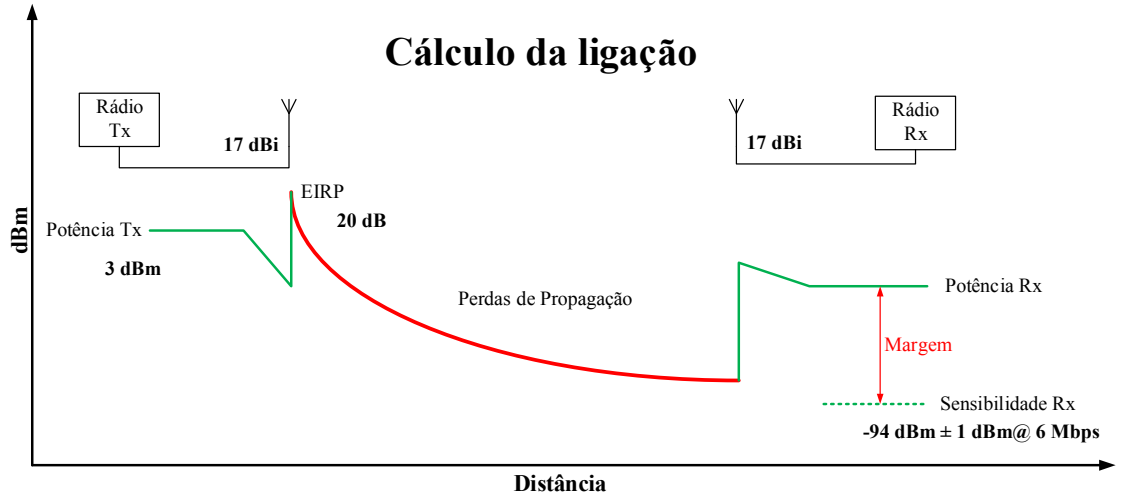



Figura 5.8 - Projeto da ligação por feixes hertzianos.

Neste sistema, como foi referido anteriormente, utiliza-se uma frequência de 2452 MHZ, estando a potência limitada a 100 mW (20 dBm) e sendo a sensibilidade do recetor, consultando as folhas de características de -94 ± 1 dBm @ 6 Mbps.

O primeiro passo para o cálculo das perdas de propagação presentes nos percursos é efetuado com recurso a equação (2.1), chegando-se aos valores da Tabela 5.3.

Tabela 5.3 - Atenuação em Espaço livre.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| Atenuação Espaço livre | 114,07 dBm | 114,90 dBm | 115,52 dBm |

Estas perdas de propagação apenas dependem da distância entre o emissor e o recetor.

Na Tabela 5.4 descrevem-se os resultados teóricos referentes aos cálculos da atenuação atmosférica nos vários percursos das ligações consideradas. Os cálculos resultam da aplicação da equação (2.24) em função dos parâmetros $\gamma_{o0}$ e $\gamma_{w0}$ considerando-se uma pressão atmosférica de 1013 hPa, a uma temperatura de 15 Cº e uma concentração de vapor de água de 7,5 g/m$^3$.

Tabela 5.4 - Atenuação provocada por oxigénio e vapor de água.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| $\gamma_0$ | 0,0069 | 0,0069 | |
| $\gamma_w$ | 0,000254 | 0,000254 | 0,0025395 |
| Atenuação Oxigénio e Vapor de água | 0,0354 dB | 0,0389 dB | 0,0418 dB |

Verifica-se uma atenuação mínima provocada pela presença de oxigénio e vapor de água.

O cálculo seguinte refere-se à atenuação provocada pela precipitação. É calculada através da interpolação dos coeficientes de $k$ (2.28) e de $\alpha$ (2.29) considerando-se uma polarização vertical, um ângulo de elevação de 0º e uma taxa de precipitação de 42 mm/h. A atenuação suplementar $\gamma_R$ (2.30) é depois multiplicada pela distância eficaz (2.31) obtendo-se os valores de atenuação presentes na Tabela 5.5.

Tabela 5.5 - Atenuação provocada por precipitação.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| $k$ | 0,000139 | 0,000139 | 0,000139 |
| $\alpha$ | 1,0234 | 1,0234 | 1,0234 |
| $\gamma_R$ | 0,0064 | 0,0064 | 0,0064 |
| $d_{eficaz}$ | 3,9076 | 4,2111 | 4,4480 |
| Atenuação de Precipitação | 0,0249 dB | 0,0269 dB | 0.0284 dB |

Analisando a tabela anterior verifica-se, novamente, uma perda pouco significativa provocada por precipitação. Os valores obtidos nas duas tabelas estão de acordo com os valores esperados para esta gama de frequências.

Tendo-se demonstrado os resultados teóricos, referentes às atenuações mais significativas, pretende-se de seguida mostrar os resultados teóricos da potência recebida em cada um dos percursos. Como o sistema de transmissão utiliza antenas com parâmetros idênticos, apenas é necessário calcular para um dos sentidos, sendo o valor igual no sentido inverso.

O cálculo seguinte pode ser apresentado como um somatório de ganhos e atenuações ou em CIP (Condições Ideais de Propagação), em que se exclui as atenuações externas.

No projeto teve-se em consideração o valor de atenuação nos cabos e conetores, como não existem dados relativos ao tipo de cabos ou conetores assumiu-se uma atenuação de 1 dB.

Utilizando-se a equação (2.41) calcula-se a potência recebida $P_{Rx}$ e registou-se na Tabela 5.6 seguinte:

Tabela 5.6 - Potência recebida em dBm.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| $P_{Rx}$ | -78,13 | -78,97 | -79,59 |
| $P_{Rx}$ CIP | -77,07 | -77,90 | -78,52 |

Analisando-se os valores teóricos, verifica-se uma diferença entre 1 a 2 dB de diferença entre os valores calculados com e sem atenuações suplementares, o que leva a concluir que existe pouca influência na banda de frequência dos 2,4 GHz.

No entanto, para existir uma ligação estável e regular o sistema deve considerar outras causas de atenuação, ou condições metrológicas adversas. Deve-se, por isso, considerar uma margem de ligação.

A fase seguinte do projeto foi a simulação dos vários percursos. Para tal utilizou-se o programa de simulação desenvolvido em MATLAB e descrito no capítulo 3, secção 3.3. Os resultados destas simulações podem ser observados na Tabela 5.7.

Tabela 5.7 - Simulação das perdas de percurso em dB.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| Espaço livre | 114,07 | 114,06 | 115,52 |
| Epstein-Peterson | - | - | 181,93 |
| Deygout | - | - | 148,65 |
| ITU-R | 125,03 | 124,92 | 137,51 |
| Egli | 139,94 | 141,74 | 135,06 |

Optou-se de seguida por apresentar os níveis de potência recebida para melhor comparação de resultados.

Tabela 5.8 - Simulação das potências recebidas em dBm.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| Espaço livre | -77,07 | -77,90 | -78,52 |
| Epstein-Peterson | - | - | -144,93 |
| Deygout | - | - | -111,65 |
| ITU-R | -88,03 | -87,92 | -100,51 |
| Egli | -102,94 | -104,74 | -98,06 |

Obtêm-se os valores da potência recebida, somando às perdas de percurso os valores da potência transmitida, ganho da antena de transmissão e ganho da antena de receção.

Analisando-se os valores da Tabela 5.7 verifica-se as tendências estudadas na fundamentação teórica do capítulo 2. Nos primeiros dois percursos não existem obstáculos à linha de vista, então não se aplica os modelos de Epstein-Peterson e Deygout.

O modelo de ITU-R apresenta valores aproximados aos esperados quando não se tem obstáculos significativos. O modelo quando aplicado em linha de vista acrescenta aproximadamente 10 dB ao valor obtido por espaço livre. Para percursos que incluem obstáculos apresenta um valor significativo de atenuação.

O modelo de Egli apresenta valores muito pessimistas, no entanto, estão de acordo com o esperado, uma vez que se trata de um modelo simplista que apresenta um valor médio de propagação. Apenas depende da distância e das características das antenas e sendo assim, apresenta valores pouco exatos.

Para a simulação do modelo de Longley-Rice optou-se por comparar o programa AirLink com o programa Radio Mobile descrito no capítulo 3, secções 3.1 e 3.2. A Tabela

5.9 apresenta os valores simulados pelo programa AirLink fornecido pelo próprio fabricante das antenas, em que se apresenta os níveis simulados de potência recebida.

Tabela 5.9 - Nível de potência simulada no AirLink em dBm.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| AirLink | -77,26 | -80,12 | -82,74 |

De seguida na Tabela 5.10 têm-se o relatório de perdas de propagação efetuado no programa de simulação do Radio Mobile, referente aos três percursos de teste. Trata-se de um programa muito completo, por isso apresenta-se apenas a informação mais relevante aos percursos em estudo.

Tabela 5.10 - Perdas de percurso no programa Radio Mobile.

| | Percurso 1 (LOS) | Percurso 2 (LOS) | Percurso 3 (NLOS) |
|---|---|---|---|
| Latitude | 32º 42’ 06,25’’ | 32º 42’ 24,44’’ | 32º 42’ 38,32’’ |
| Longitude | 016º 56’ 14,04’’ | 32º 42’ 24,44’’ | 016º 54’ 54,15’’ |
| Elevação do terreno | 1154,8 m | 1405,7 m | 1468,0 m |
| Altura da antena | 2 m | 2 m | 5 m |
| Azimute | 165,62 TN 170,94 MGº | 190,38 TN 195,69 MGº | 188,36 TN 193,68 MGº |
| Inclinação | -11,21º | -12,75º | -12,51º |
| **Perdas** | | | |
| Espaço livre | 114,20 dB | 115,10 dB | 115,70 dB |
| Obstruções | 0,6 dB | 0,6 dB | 6,50 dB |
| Floresta | 0,00 dB | 0,00 dB | 0,00 dB |
| Perdas urbanas | 0,00 dB | 0,00 dB | 0,00 dB |
| Estatísticas | 6,70 dB | 6,70 dB | 6,70 dB |
| **Total** | **121,50 dB** | **122,40 dB** | **128,90 dB** |
| **Desempenho** | | | |
| Distância | 4,944 km | 5,440 km | 5,842 km |
| Frequência | 2452 MHz | 2452 MHz | 2452 MHz |
| EIRP | 0,100 W | 0,100 W | 0,100 W |
| Ganho sistema | 132,00 dB | 132,00 dB | 132,00 dB |
| Fiabilidade | 70,00% | 70,00% | 70,00% |
| Sinal recebido | **- 84,50 dBm** | **- 85,40 dBm** | **- 91,90 dBm** |
| Margem | **9,5 dB** | **8,60 dB** | **2,10 dB** |

Analisando-se a tabela anterior, verifica-se as características dos percursos. O programa fornece o valor de elevação onde é colocada cada antena juntamente com o ângulo de azimute, sendo TN (*True North*) o norte geográfico enquanto MG (*Magnetic*

*North*) é o norte magnético, também é calculada a melhor inclinação para facilitar o alinhamento das antenas.

Calculam-se as perdas de propagação ao longo dos percursos, sendo apresentadas as perdas em espaço livre, as perdas devido a obstáculos, as perdas devido a passagem por zonas de floresta, as perdas devido a zonas urbanas e finalmente as perdas estatísticas para uma fiabilidade de 70% do tempo de funcionamento.

O programa analisa cada percurso, e, utilizando as perdas e o ganho do sistema chega a um valor de sinal recebido pela antena recetora e uma margem de ligação entre os percursos.

Analisando-se as tabelas 5.9 e 5.10 verifica-se uma clara tendência otimista para o programa do fabricante em relação ao programa Radio Mobile.

Nos dois percursos com linha de vista verifica-se uma diferença de aproximadamente 6 dB enquanto no percurso obstruído verifica-se uma diferença de aproximadamente 11 dB.

## 5.4 Ligação no terreno

Tendo os percursos e os valores esperados de potência recebida, o passo seguinte foi a confirmação no terreno. Para tal, foi necessário proceder ao alinhamento da antena.

Utilizou-se numa primeira fase o alinhamento visual, depois utilizou-se a aplicação de alinhamento, da própria antena, que faz uso da intensidade do sinal recebido em tempo real para os ajustes mais finos. Esta aplicação encontra-se representada na Figura 5.9.

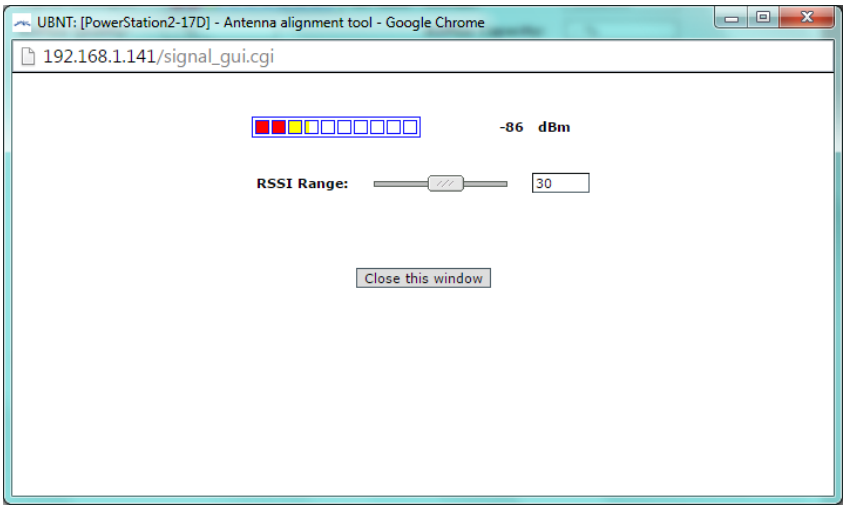



Figura 5.9 - Ferramenta de alinhamento da antena.

O programa Radio Mobile também é útil porque fornecer os valores de Azimute e Inclinação para a antena transmissora. Com base nos valores da Tabela 5.11 e com o procedimento já descrito anteriormente, conclui-se o alinhamento da antena.

Tabela 5.11 - Características de localização e alinhamento da antena transmissora.

| Latitude | Longitude | Elevação | Altura antena | | Azimute | Inclinação |
|---|---|---|---|---|---|---|
| 32°39'31.29"N | 16°55'26.98"W | 179,1 m | 1,0 m | Percurso 1 | 345,63° TN 350,95°MG | 11,16° |
| | | | | Percurso 2 | 10,37° TN 15,69° MG | 12,70° |
| | | | | Percurso 3 | 8,37° TN 13,68° MG | 12,46° |

Escolhidas as localizações e efetuado o alinhamento das antenas procedeu-se ao registo dos valores de sinal recebido.

A antena de feixes hertzianos fornece os valores de potência recebida. No entanto, os fabricantes tendem a ser otimistas em relação aos seus produtos e para aferir do sinal real, optou-se por utilizar o analisador de espetro FSH8 da ROHDE & SCHWARZ [59].

Para tal foi necessário separar a antena da parte de potência e adaptar um cabo RG59 para ligar a antena e o analisador. Observa-se este procedimento na Figura 5.10 a) e o resultado da adaptação na Figura 5.10 b).

a) b)

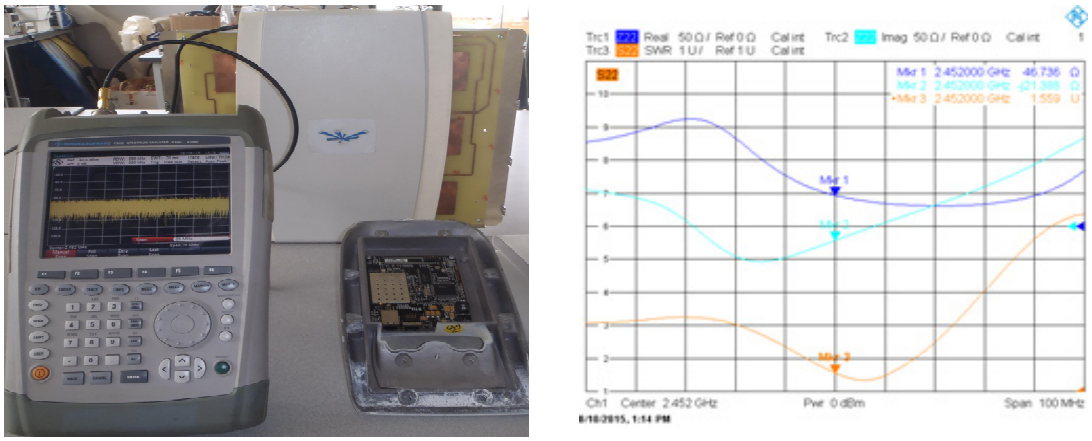


Figura 5.10 - a) Separação da antena da parte de potência. b) Gráfico de adaptação.

A adaptação do cabo visa não introduzir mais atenuação no sistema de aquisição de dados. Utilizou-se a frequência de operação na gama dos 2,452 GHz e atingiu-se um valor de SWR (*Standing Wave Ratio*) de 1,559.

Na Figura 5.11 observa-se o sistema completo montado no percurso 1 em linha de vista nas serras de São Roque.

Figura 5.11 - Protótipo completo montado nas Serras de São Roque.

Para chegar aos locais de teste, utilizou-se um programa GPS (*Global Positioning System*). Esta aplicação tem um erro de aproximadamente 10 m, o que serviu para escolher a melhor localização dentro das coordenadas obtidas pelos programas de simulação. Efetuou-se este procedimento para as três localizações.

Na primeira parte do teste montou-se o protótipo completo. Aferiu-se o sinal com recurso ao medidor interno da antena ligado ao computador e testou-se a existência ou não de acesso a internet, verificando-se a entrega dos dados recolhidos pelo nó sensor ao servidor.

Na segunda parte do teste foram desligados os cabos da antena da parte de potência e analisada a frequência dos 2,452 MHz. Repetiu-se o procedimento de minuto a minuto, efetuando-se 30 medições e obteve-se os gráficos da Figura 5.12, Figura 5.13 e Figura 5.14.

Na Figura 5.12 tem-se as medições realizadas no final do percurso 1, entre a Universidade da Madeira e as Serras de São roque.

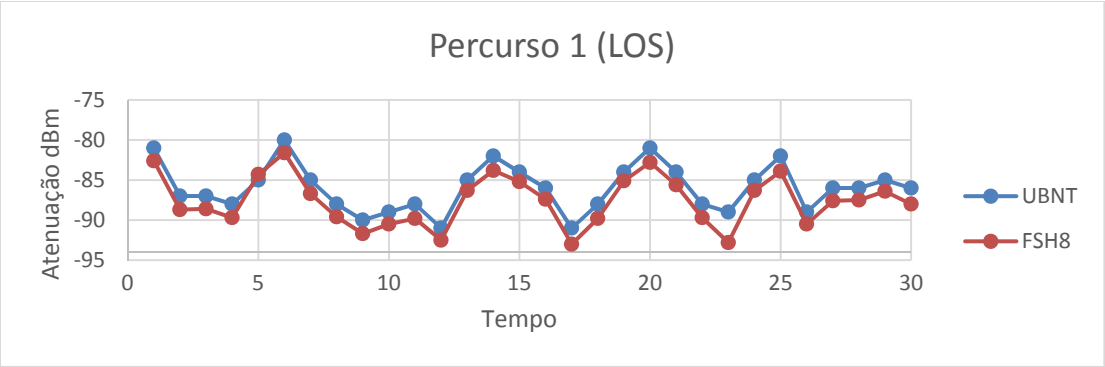



Figura 5.12 - Valores de potência recebida no percurso 1.

Na Figura 5.13 apresenta-se as medidas aferidas no final do percurso 2, entre a Universidade da Madeira e o Chão da Lagoa (Percurso em linha de vista).

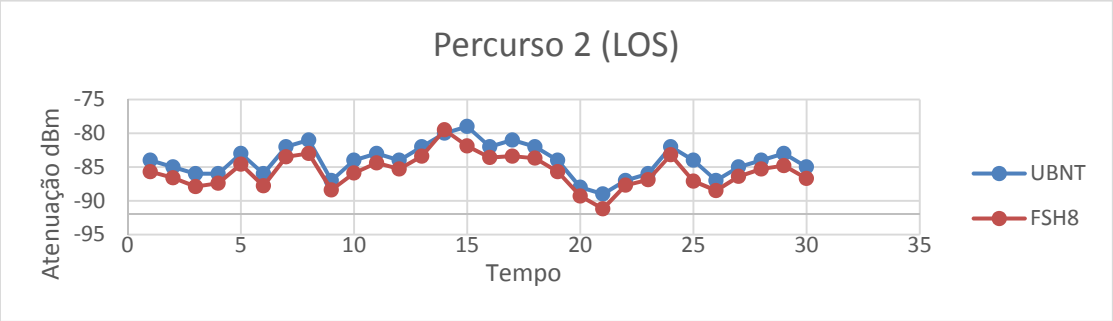



Figura 5.13 - Valores de potência recebida no percurso 2.

Na Figura 5.14 representa-se as medidas efetuadas no final do percurso 3, entre a Universidade da Madeira e o Chão da Lagoa (Percurso com linha de vista obstruído).

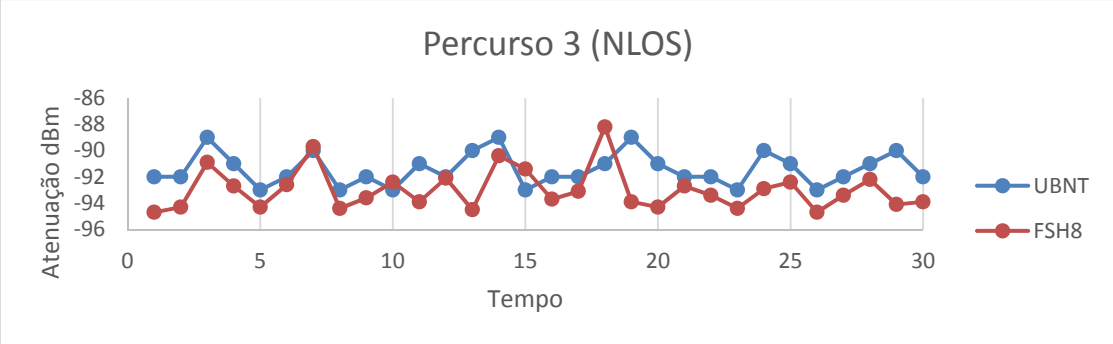



Figura 5.14 - Valores de potência recebida no percurso 3.

Analisando-se os gráficos verificam-se oscilações do sinal recebido ao longo do tempo. Está deve-se a atenuação do sinal pela passagem por zonas de vegetação.

Na tabela 5.12 apresentam-se aos valores médios de sinal recebido.

Tabela 5.12 - Valores de potência de sinal recebido em dBm.

| | Valor UBNT | Valor analisador | Diferença |
|---|---|---|---|
| Percurso 1 | -84,00 | -85,60 | 1,59 |
| Percurso 2 | -86,00 | -87,60 | 1,60 |
| Percurso 3 | -91,43 | -92,97 | 1,54 |

Observa-se uma diferença de 1,6 dBm entre os valores obtidos no medidor de sinal da própria antena e o analisador de espetro FHS8.

Assim, conclui-se da análise dos valores que a diferença existente entre os valores de potência recebida pode ser contabilizada pela atenuação introduzida pelo cabo, que foi necessário adicionar ao sistema de aquisição de dados para poder efetuar as medições.

Após a realização dos testes anteriores, apresenta-se o quadro de comparação de resultados da Tabela 5.13. Nesta tabela observa-se os valores simulados das perdas de propagação pelos vários modelos estudados.

Tabela 5.13 - Comparação de valores

| **Modelos** | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| Espaço livre | -77,07 | -77,90 | -78,52 |
| Epstein-Peterson | - | - | -144,93 |
| Deygout | - | - | -111,65 |
| ITU-R | -88,03 | -87,92 | -100,51 |
| Egli | -102,94 | -104,74 | -98,06 |
| AirLink | -77,26 | -80,12 | -82,74 |
| Radio Mobile | -84,50 | -85,40 | -91,90 |
| **Experimentais** | | | |
| Antena UBNT | -84,00 | -86,00 | -91,43 |
| Analisador FHS8 | -85,60 | -87,60 | -92,97 |

O modelo de ITU apresenta um valor médio de atenuação. O modelo utiliza uma relação entre o obstáculo mais significativo e o percurso em linha de vista entre o emissor e o recetor. Neste caso, apresenta um valor pessimista para o percurso sem linha de vista, com cerca de 7,5 dB de diferença entre o valor calculado e o valor analisado, e apresenta um valor de 2,4 dB de diferença para o percurso com interferência no elipsoide de Fresnel. Quando não existem obstáculos, o modelo apenas prevê uma soma de 10 dB ao valor em espaço livre. No caso em estudo até apresenta um valor aproximado, no entanto, não será o caso para todos os percursos.

O modelo de Egli prevê a atenuação total da ligação sem recurso ao perfil do terreno, verificam-se valores pessimistas. No primeiro percurso, com interferência, apresenta um valor de atenuação de 102,94 dB enquanto no segundo percurso apresenta 104,74 dB sem obstáculos e sem interferências. Isto deve-se ao modelo não tomar em consideração o perfil do terreno. A altura e ganho das antenas são iguais, mas no entanto as distâncias dos percursos são diferentes. O percurso 2, que não tem obstáculos, tem uma distância maior e por isso apresenta um valor previsto de atenuação superior. No terceiro percurso verifica-se um valor de atenuação de 98,06 dB devido a considerar-se uma altura de antena superior, para a receção de sinal. Assim, conclui-se que o modelo de Egli não é o mais indicado.

Através da análise da tabela anterior verifica-se que o modelo que mais se aproxima do valor real de sinal recebido pelo recetor é o modelo de Longley-Rice. Este modelo é

um modelo computacional complexo, no entanto, consegue estimar um valor de simulação com aproximadamente 2 dB de diferença para os 3 percursos testados.

## 5.5 Margem de ligação

Tendo-se avaliado os valores recebidos pela antena, torna-se necessário verificar se a margem apresentada está de acordo com as necessidades do sistema de aquisição de dados.

A margem de ligação pode ser calculada através da equação (2.42). A estes resultados calculados juntou-se os valores simulados no programa Radio mobile e obteve-se a Tabela 5.14.

Tabela 5.14 - Margem de ligação.

| | Percurso 1 | Percurso 2 | Percurso 3 |
|---|---|---|---|
| Espaço livre | 15,84 | 14,94 | 14,34 |
| Long-Rice | 9,50 | 8,60 | 2,10 |

Analisando a tabela anterior é possível verificar a diferença entre os valores da margem de ligação em espaço livre, ou seja, em condições ideias de propagação e o modelo que melhor se aproxima dos valores de sinal aferido nos percursos estudados.

Na Figura 5.15 é apresentada uma representação possível da margem de ligação de um sistema de comunicação por feixes hertzianos.

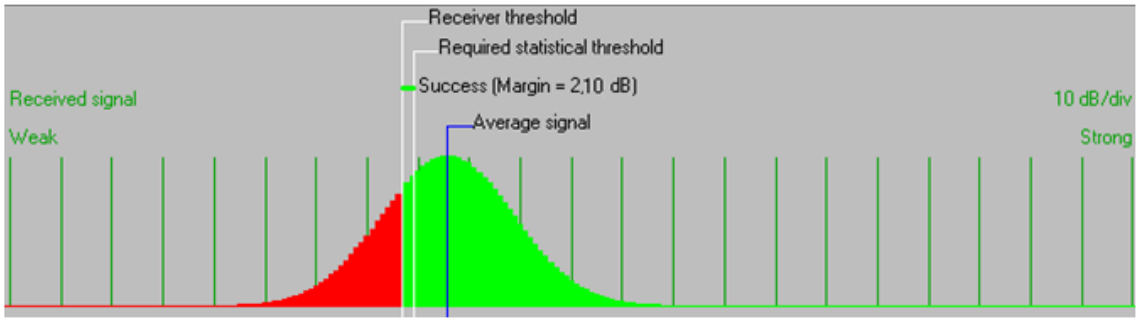



Figura 5.15 - Margem de ligação.

Obteve-se esta representação através do programa Radio mobile, e verifica-se que este corresponde ao percurso 3, ou seja, um percurso sem linha de vista e com varias obstruções. Verifica-se apenas 2,10 dB de margem com uma sensibilidade fixada em -94 dBm o que leva a uma ligação pouco estável.

A margem de ligação recomendada depende muito da aplicação em questão. No entanto, a maiorias dos autores recomenda uma margem mínima aceitável entre 5 dB a 10 dB. Estes valores de margem de ligação dependem da frequência, da taxa de transmissão, e do estudo da média dos valores recolhidos de sinal recebido no local de implementação.

## 5.6 Apresentação dos dados

Na Figura 5.16 apresenta-se a recolha e armazenamento de dados no cartão micro SD. Os dados são armazenados num ficheiro de texto e caso o sistema funcione corretamente, são enviados em blocos de aproximadamente 16 kB para o servidor. No entanto, o sistema está preparado, caso exista algum erro ou avaria, para manter os dados no cartão. Este cartão possui a capacidade de 2 GB, sendo o sistema capaz de armazenar uma grande quantidade de dados.

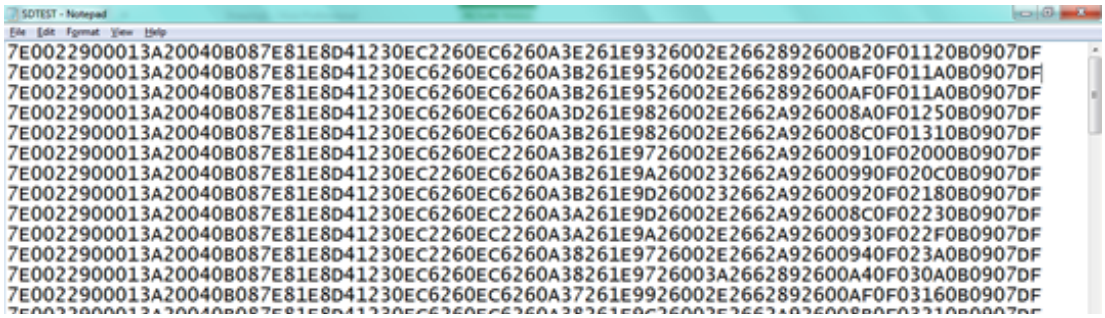



Figura 5.16 - Dados no cartão SD.

Na Figura 5.17 apresenta-se o processo de envio de dados, observado na porta série do microcontrolador. Em primeiro lugar, o sistema verifica se existe e consegue ler o ficheiro contendo os dados, depois verifica se obteve DHCP e atualiza o relógio. Finalmente procede ao envio dos dados linha a linha para o servidor.

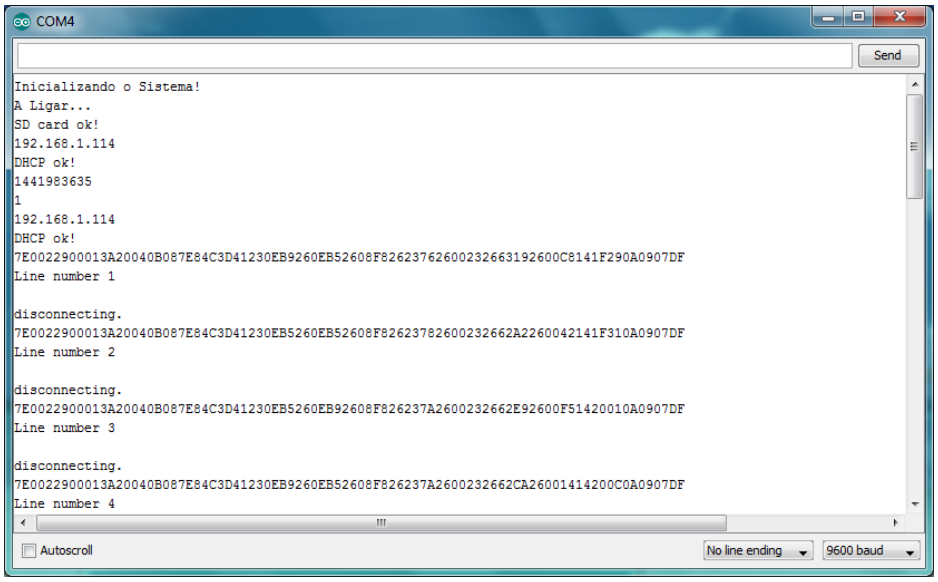



Figura 5.17 - Processo de envio de dados.

Na Figura 5.18 apresenta-se a plataforma de visualização de dados.

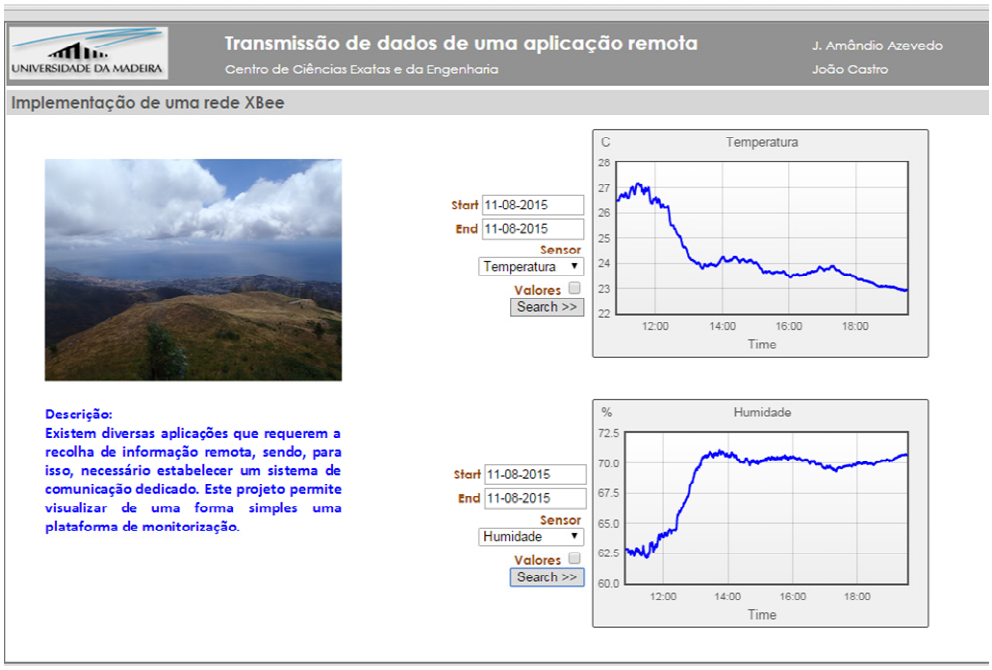
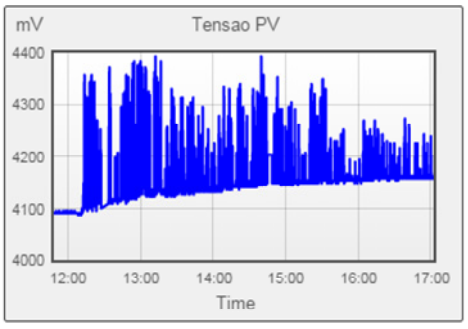



Figura 5.18 - Visualização dos dados na plataforma.

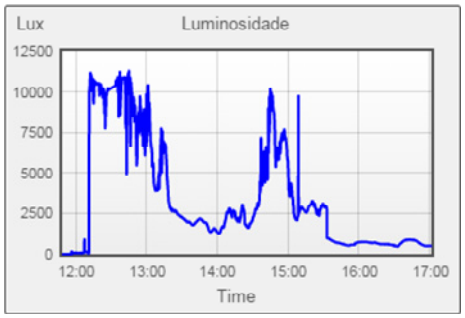
Os dados são inseridos nos respetivos campos e são representados graficamente ou através de valores. Nos gráficos apresentados visualizam-se os valores da temperatura e humidade ao longo do tempo. No dia em questão, pode-se observar um dia em que começou a chuviscar na zona de recolha dos dados. Verifica-se uma diminuição dos valores da temperatura e o normal aumento da humidade.

O nó sensor permite a recolha de outras informações. Na Figura 5.19 apresentam-se a tensão no painel fotovoltaico, a tensão nas baterias, a luminosidade e a pressão atmosférica.

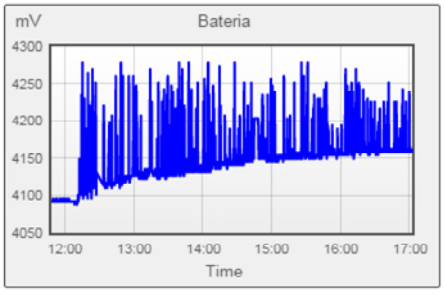
a)




b)




c)




d)

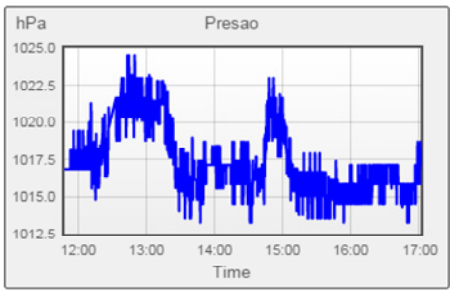



Figura 5.19 - a) Tensão no painel fotovoltaico. b) Luminosidade. c) Tensão na bateria. d) Pressão atmosférica.

Analisando as figuras anteriores observa-se na Figura 5.19 a) e na Figura 5.19 c) o ajuste entre a tensão do painel fotovoltaico e a tensão da bateria, e verificando-se o carregamento da bateria. A oscilação entre os valores deve-se ao controlador de carga do microcontrolador utilizado.

Na Figura 5.19 c) verificam-se os valores obtidos de luminosidade e, como já foi descrito anteriormente, tratou-se de um dia bastante nublado. Existindo períodos de precipitação fraca. Verificou-se um valor médio entre os 10000 a 25000 lux o que está de acordo com valores esperados para dias nublados.

Na Figura 5.19 apesenta-se os valores da pressão atmosférica, verificando-se uma descida entre as 13 e 14 horas o que coincide com o período de chuva fraca.

## 5.6 Conclusões de capítulo

Neste capítulo foram apresentados os testes e análise de valores de sinal simulado e recebido.

Começou-se por explicar as considerações a ter na implementação de um sistema de transmissão de dados de uma aplicação remota através de feixes hertzianos. Optou-se por considerar 3 percursos com características distintas de propagação para efetuar os testes de ligação. Recorreu-se ao programa Google Earth para verificar os perfis de propagação e a simulação em Matlab e Radio mobile para estimar as perdas de propagação.

Calcularam-se os valores teóricos das perdas de propagação e chegou-se a conclusão que o modelo de Espaço livre serve de base aos cálculos e aos modelos de propagação. Os modelos de Epstein-Peterson e Deygout não podem ser utlizados em percursos sem obstáculos.

O modelo de ITU apresenta uma aproximação do sinal recebido em percursos sem interferências significativas. Na ausência completa de obstáculos apresenta apenas uma soma de 10 dB ao modelo de espaço livre e nos percursos com obstáculos observam-se valores pessimistas.

O modelo de Egli dá um valor de perdas de propagação sem recurso ao perfil do terreno, no entanto, verificaram-se valores de atenuação pessimista.

O modelo que mais se aproxima aos valores recolhidos é o modelo de Longley-Rice. Utilizando-se o programa de simulação Radio mobile chegou-se a um valor com uma aproximação média de 1,45 dB nos 3 percursos considerados.

Verificou-se o modo de envio e entrega de dados, juntamente com a sua apresentação gráfica no servidor através de uma página de *internet*, que poderá ser consultada em qualquer lugar.

# 6. Conclusões e trabalhos futuros

Neste capítulo são apresentadas as conclusões finais do projeto referindo as contribuições na área de sistemas de aquisição de dados de aplicações remotas. São também ainda referidas algumas propostas de trabalhos futuros que poderão se de interesse o seu desenvolvimento.

## 6.1 Conclusões

A proposta desta tese de mestrado foi a realização de um projeto de um sistema de aquisição de dados de uma aplicação remota. Começou-se este trabalho com um estudo teórico das duas tecnologias de transmissão de dados, a tecnologia de rádio frequência utlizadas nas redes de sensores sem fios e a tecnologia de feixes hertzianos utlizadas na comunicação a longa distância.

Depois, numa primeira fase, estudou-se os modelos de perdas de propagação, com especial atenção para os modelos de propagação em terrenos irregulares. Deste estudo conclui-se que o modelo de Espaço Livre serve de base aos restantes modelos apresentados. O modelo de Obstáculo em Lâmina apenas considera o percurso com um único obstáculo.. O modelo de Epstein-Petterson divide o percurso total em sub-percursos e soma as atenuações dos sub-percuros o que leva a resultados otimistas quando não existem obstáculos próximos uns dos outros. O modelo de Deygout apenas considera o obstáculo principal e é pessimista quando não se tem um obstáculo destacado dos restantes. O modelo de Egli não toma em consideração o perfil. O modelo de Longley-Rice apresenta-se como o modelo mais complexo e requer computação para aferir os valores de propagação.

Na continuação do trabalho, estudaram-se os modelos utilizados na previsão das perdas de propagação em vegetação, em particular os modelos de Weissberger, ITU, COST 235, Log-Normal e o modelo de Azevedo & Santos. Finalmente, e para a conclusão do estudo teórico, abordou-se os modelos de atenuação devido as condições atmosféricas.

Deu-se relevância à análise dos feixes hertzianos, referindo-se os tipos de atenuação e o desvanecimento. Abordou-se os componentes e preocupações a considerar quando se projeta uma ligação. Finalmente deu-se a título de exemplo os cálculos teóricos de um sistema de feixes hertzianos e o cálculo da margem de ligação do sistema.

Estudaram-se os programas de simulação de perdas de propagação. O programa *AirLink* é fornecido juntamente com as antenas de feixes hertzianos. Tem como vantagem ser bastante simples e como desvantagens requer uma ligação à internet, acesso ao programa *Google Earth* e não considera as limitações impostas pela regulação do espetro dos vários países. Foi necessário considerar uma atenuação de 23 dB para atingir os 3 dB de potência da antena de transmissão. Na análise dos percursos de propagação considerados verificou-se valores muito abaixo dos medidos na realidade.

O segundo *software* analisado foi o *Radio Mobile* uma vez que se trata de um dos programas mais utilizados a nível mundial e utiliza o modelo de propagação de Longley-

Rice. É gratuito e tem a vantagem de, após o *download* dos mapas de perfil, não requerer acesso à internet e permitir o estudo de antenas personalizadas. Verificou-se através da análise do programa uma aproximação média de 1,45 dB entre os valores simulados pelo programa e os valores medidos com recurso ao analisador de espetro utilizado.

O *software* de simulação desenvolvido em plataforma MATLAB foi utlizado para fazer uma análise comparativa entre os modelos a qual juntou-se a análise da atenuação provocada perdas devido a nevoeiro ou precipitação.

Verificou-se uma aproximação dos resultados teóricos, no entanto, fez falta uma aquisição de cotas de perfil mais exatas uma vez que se tornou difícil utilizar os mesmos pontos de propagação.

Estudaram-se os modelos de atenuação provocada pelas condições atmosféricas às quais chegou-se a conclusão não serem significativas para a banda de frequência ISM dos 2,4 GHz.

Optou-se por apresentar o protótipo do sistema de transmissão de dados de uma aplicação remota desenvolvido, para tal, apresentou-se os componentes constituintes do protótipo, juntamente com os fluxogramas de funcionamento e programação efetuada.

O protótipo do nó sensor desenvolvido mede a temperatura, humidade, luminosidade e pressão atmosférica. Foram recolhidos vários dados e verificou-se uma boa aproximação entre os valores obtidos e os dados disponibilizados pelo IPMA (Instituto Português do Mar e da Atmosfera). Neste protótipo também optou-se por medir a tensão nas baterias e tensão do painel solar, chegando-se a um valor de duração da bateria de 29,34 dias com um consumo médio de 2,272 mA para uma bateria de 2000 mA, o que demostra ser adequado para um nó com capacidades de sensor.

O protótipo do nó coordenador desenvolvido controla o sistema de aquisição de dados juntamente com a gestão de energia e entrega de dados no servidor com representação gráfica *online*.

Neste sistema, o problema mais significativo foi o consumo do sistema no seu todo, uma vez que a antena de feixes hertzianos tem um consumo de 1 A @ 15 V. Como o objetivo é um sistema autónomo e alimentado através de energia solar teve-se que dimensionar os sistemas para apenas ligar de 30 em 30 minutos. Um outro componente que apresentava consumo elevado foi o módulo de Ethernet. Fez-se um estudo comparativo entre módulos de Ethernet e chegou-se a conclusão que o módulo de Ethernet W5200 apresenta o menor consumo de todos os componentes estudados.

Conclui-se qua a duração da bateria não era a ideal, no entanto demostra ser adequada a aplicação em questão. Chegou-se a um valor de duração da bateria de 5,03 dias com um consumo médio de 119,23 mA para uma bateria de 18 A. Caso se verifique a necessidade de aumentar o período de duração sem carregamento do sistema, recomenda-se o aumento da capacidade da bateria ou o tempo de intervalo entre o envio de dados.

Nos testes e análise de valores de sinal simulado e recebido, começou-se por explicar as considerações a ter na implementação de um sistema de transmissão de dados de uma aplicação remota através de feixes hertzianos. Optou-se por considerar 3 percursos com características distintas de propagação para efetuar os testes de ligação. Recorreu-se ao programa Google Earth para verificar os perfis de propagação e a simulação em Matlab e Radio mobile para estimar as perdas de propagação.

Calcularam-se os valores teóricos das perdas de propagação e chegou-se à conclusão que o modelo de Espaço livre serve de base aos cálculos e aos modelos de propagação. Os modelos de Epstein-Peterson e Deygout são muito otimistas e não podem ser utlizados em percursos sem obstáculos. Em percursos, sem linha de vista, os modelos apresentam valores de propagação muito pessimistas. Isto deve-se aos obstáculos principais considerados, que ocorrem nos primeiros 100 m e por não existir nenhum obstáculo que se destaque.

O modelo que mais se aproxima aos valores recolhidos é o modelo de Longley-Rice. Verificando-se um valor com uma aproximação média de 1,45 dB nos 3 percursos considerados.

Verificou-se que a variação existente na potência de sinal recebido deve-se a passagem por zonas de vegetação em que não são contabilizadas no modelo com melhor desempenho.

Conclui-se que é essencial prever as perdas de propagação no projeto de uma ligação por feixes hertzianos, sendo de igual importância prever uma margem de ligação devido as interferências e oscilações de sinal recebido.

A plataforma de visualização de dados, após as alterações efetuadas demostrou-se adequada a este tipo de sistemas de aquisição de dados, representando os dados graficamente ou através de valores.

Verificou-se que o modo de envio e entrega de dados do protótipo de comunicação desenvolvido é adequado a aplicação em questão, facultando a consulta e análise dos dados recolhidos, em ambientes remotos.

Na conclusão deste trabalho apresentou-se um protótipo de um sistema de comunicação dedicado que junta de duas tecnologias de comunicação, a comunicação através de radio frequência das redes de sensores sem fios e a comunicação por feixes hertzianos utilizado a banda de frequência ISM dos 2.4 GHz. Este sistema pode ser utlizado para vários fins sendo o apresentado nesta tese, apenas um.

## 6.2 Trabalhos futuros

Na elaboração desta tese foram tomadas decisões e opções que estabeleceram um rumo a seguir, eventualmente, outras opções poderiam ter sido consideradas levando a resultados distintos.

A nível da programação do microcontrolador sugere-se uma melhoria nos algoritmos desenvolvidos, verificando a possibilidade de diminuir o tamanho do código, e assim possibilitando a utilização de outros microcontroladores mais compactos.

O *software* desenvolvido em plataforma é um projeto que também pode sofrer alterações e melhorias, integrando mapas mais detalhados, possibilitando a exportação para o *Google Earth*.

Para um estudo mais completo e compreensão estatístico recomenda-se a implementação do sistema numa localização fixa e segura, durante um período de tempo alargado. Tendo em vista complementar o estudo efetuado neste trabalho.

# 7. Bibliografia

[1] D. P. Dickinson, "www.phoenixcontact.com," [Online]. Available: https://www.phoenixcontact.com/assets/downloads_ed/local_us/web_dwl_technical_info/What_Wireless_white_paper_final.pdf?filesize=727. [Acedido em 22 11 2014].

[2] "http://www.anacom.pt/," [Online]. Available: http://www.anacom.pt/. [Acedido em 2014 11 21].

[3] C. Salema, Feixes Hertzianos, Lisboa, Portugal: IST Press, 1998.

[4] J. D. Parsons, The Mobile Radio Propagation Channel, Chichester, U.K.: Wiley, 1996.

[5] M.A.Weissberger, "An initial critical summary of models for predicting the attenuation of radio waves by trees," Maryland, 1981.

[6] C. 235, "Radio Progation Effects on Next-Generation Fixed-Sevice Terrestrial Telecomunication Systems," Luxembourg, 1996.

[7] I.-R. P.530-14, "Propagation and Prediction Melhods Required for The Design of Terrestrial Line-of-Sight Systems," *ITU-R P.530-14,* 2012.

[8] T. S. Rappaport, Wireless Comunications, Second Edition ed., P. Education, Ed., Pearson Education, pp. 150-154.

[9] J. R. Azevedo e F. Santos, "An Empirical propagation Model for Forest Environments at Tree Trunk Level," *IEEE Transactions on Antennas and Propagation,* vol. 59 Issue 6, pp. 2357-2367, June 2011.

[10] O. Kurnaz, M. Bitigan e S. Helhel, "Procedure of Near Ground Propation Modle Development for Pine Tree Forest Environment," *Progress In Electromagnetics Research Symposium Proceedings,* pp. 1403-1406, August 2012.

[11] [Online]. Available: http://www.proxim.com/scripts/calculators/what-is-fresnel-zone.jpg. [Acedido em 05 01 2015].

[12] S. R. Saunders e A. Aragón-Zavala, Antennas and Propagation for Wireless Communication Systems, England: John Wiley & Sons Ltd, 2007.

[13] M. Rodrigues, "Sistema de Comunicações para as Ilhas Desertas," UMa, Funchal, 2009.

[14] J. S. Seybold, Introduction to RF Propagation, Hoboken, New Jersey: John Wiley & Sons, 2005.

[15] A. G. Longley e P. L. Rice, "Prediction of Tropospheric Radio Transmission Loss Over Irregular Terrain - A Computer Method - 1968," Institute for Telecommunication Sciences, Boulder, Colorado, 1968.

[16] G. Hufford, A. Longley e W. Kissick, "A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode," 1982.

[17] Y. S. Meng e Y. H. Lee, "Investigations of foliage effect on modern wirless comunication systems: A Review," *Progress In Electromagnetics Research.*

[18] I.-R. P.676-10, "Attenuation by atmospheric gases," Recommendation ITU-R P.676-10, (09/2013).

[19] [Online]. Available: http://www.mike-willis.com/Tutorial/PF_files/Picture48.gif. [Acedido em 01 02 2015].

[20] I.-R. P.840-5, "Attenuation Due to Clouds and Fog," *ITU-R P.840-5,* 2012.

[21] I.-R. F.634-4, "Error performance objectives for real digital radio-relay links forming part of the high-grade portion of international digital connections at bit rate below the primary rate within an intergrated services digital network," ITU-R, 1997.

[22] I.-R. P.838-3, "Specific attenuation model for rain for use in prediction methods," em *RECOMMENDATION ITU-R P.838-3*, 2001/3.

[23] I.-R. P.837-1, "www.itu.com," [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.837-1-199408-S!!PDF-E.pdf. [Acedido em 06 01 2015].

[24] "http://www.itu.int/," [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.530-7-199708-S!!PDF-E.pdf. [Acedido em 10 01 2015].

[25] C. A. Balanis, Antenna Theory - Analyis and Design, Hoboken, New Jersey: John Wiley & Sons, 2005.

[26] I.-R. P.341-5, "The Concept of Transmission Loss for Radio Links," Rec. ITU-R P.341-5, 1999.

[27] T. M. d. S. Nunes, "Projeto de uma ligação por feixe hertziano entre dois pontos terminais," IST, Lisboa, 2012.

[28] "http://airlink.ubnt.com/," [Online]. Available: http://airlink.ubnt.com/. [Acedido em 28 05 2015].

[29] U. Networks, "Ubiquiti Networks AirLink," [Online]. Available: http://www.ubnt.com/airlink/. [Acedido em 10 2 2015].

[30] "http://www.ve2dbe.com/english1.html," [Online]. Available: http://www.ve2dbe.com/english1.html. [Acedido em 29 05 2015].

[31] R. Coudé, "Radio Mobile," [Online]. Available: http://www.cplus.org/rmw/english1.html. [Acedido em 10 9 2014].

[32] U. Survey, "U.S.Geological Survey," [Online]. Available: http://www.usgs.gov/. [Acedido em 10 2 2015].

[33] R. Coudé, "Radio Mobile Downloads," [Online]. Available: http://www.g3tvu.co.uk/RM_Downloads.htm. [Acedido em 22 6 2015].

[34] mathworks, "MATLAB site," [Online]. Available: http://www.mathworks.com/products/matlab/. [Acedido em 8 9 2014].

[35] Matlabcentral, "Matlabcentral," [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/. [Acedido em 9 2 2014].

[36] A. Anada, M. C. Chan e W. T. Ooi, Mobile,Wireless, and Sensor Networks, New Jersey: John Wiley & Sons, Inc, 2006.

[37] "http://www.zigbee.org/," [Online]. Available: http://www.zigbee.org/what-is-zigbee/. [Acedido em 20 07 2015].

[38] "http://www.digi.com/pdf/ds_xbeezbmodules.pdf," [Online]. Available: http://www.digi.com/pdf/ds_xbeezbmodules.pdf. [Acedido em 24 11 2014].

[39] "www.digi.com," [Online]. Available: http://ftp1.digi.com/support/documentation/90000976_W.pdf. [Acedido em 22 06 2015].

[40] "XCTU Download," [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu. [Acedido em 20 03 2015].

[41] "ATmega328p," [Online]. Available: http://www.atmel.com/Images/doc8161.pdf. [Acedido em 24 11 2014].

[42] "www.sensirion.com," [Online]. Available: http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf. [Acedido em 14 11 2014].

[43] "http://www.hamamatsu.com," [Online]. Available: http://www.hamamatsu.com/resources/pdf/ssd/si_pd_kspd0001e.pdf. [Acedido em 17 11 2014].

[44] "www.electrorayd.com," [Online]. Available: http://www.electrorayd.com/pdfs/Tabelas/catalogo_AL_2007.pdf. [Acedido em 13 08 2015].

[45] "http://www.atmel.com/images/doc2549.pdf," [Online]. Available: http://www.atmel.com/images/doc2549.pdf. [Acedido em 24 11 2014].

[46] "http://www.farnell.com," [Online]. Available: http://www.farnell.com/datasheets/27611.pdf. [Acedido em 20 06 2015].

[47] "www.anacom.pt," [Online]. Available: http://www.anacom.pt/streaming/qnaf20092010_07042010.pdf?contentId=1019281&field=ATTACHED_FILE. [Acedido em 21 07 2015].

[48] "https://www.ubnt.com/," [Online]. Available: https://www.ubnt.com/. [Acedido em 21 6 2014].

[49] "www.ubnt.com," [Online]. Available: http://dl.ubnt.com/ps2_datasheet.pdf. [Acedido em 09 06 2014].

[50] "www.sparkfun.com," [Online]. Available: https://www.sparkfun.com/datasheets/Batteries/UnionBattery-2000mAh.pdf. [Acedido em 03 08 2015].

[51] "www.active-robots.com," [Online]. Available: https://www.active-robots.com/fileuploader/download/download/?d=0&file=custom%2Fupload%2FFile-1381402105.pdf. [Acedido em 20 07 2015].

[52] "www.sparkfun.com," [Online]. Available: https://www.sparkfun.com/datasheets/Components/MAX1551-MAX1555-1.pdf. [Acedido em 23 04 2015].

[53] "http://www.jprelec.co.uk/store.asp/c=1242/Lead-Acid-GEL-Deep-discharge-Batteries#," [Online]. Available: file:///C:/Users/Joao%20Castro/Desktop/Pen/Desktop%20LG/Datasheets/BEG120180-Camden-Boss.pdf. [Acedido em 23 07 2015].

[54] "http://www.enfsolar.com," [Online]. Available: http://www.enfsolar.com/pv/panel-datasheet/Polycrystalline/1580. [Acedido em 22 05 2015].

[55] "http://www.steca.com," [Online]. Available: http://www.steca.com/frontend/standard/popup_download.php?datei=191/191

78_0x0x0x0x0_Steca_PR_10_30_specification_EN_web_15_21.pdf. [Acedido em 01 07 2015].

[56] "https://www.apachefriends.org," [Online]. Available: https://www.apachefriends.org/download.html. [Acedido em 10 07 2015].

[57] "https://notepad-plus-plus.org/," [Online]. Available: https://notepad-plus-plus.org/. [Acedido em 03 02 2015].

[58] "https://www.google.com," [Online]. Available: https://www.google.com/earth/. [Acedido em 23 05 2015].

[59] "http://www.rohde-schwarz.pt/," [Online]. Available: http://www.rohde-schwarz.pt/. [Acedido em 07 08 2015].

# Anexo I

| **Parâmetros ambientais** | | |
|---|---|---|
| Irregularidade do Terreno | **Características do Terreno** | **Δh (m)** |
| | Plano, mar, lagos | 0 |
| | Planícies | 30 |
| | Montes (irregularidades médias) | 90 |
| | Montanhas | 200 |
| | Montanhas Irregulares | 300 |
| **Clima** | | |
| **Tipo de Clima** | | **Valores sugeridos para Ns** |
| Deserto | | 280 |
| Continental Temperado | | 301 |
| Continental Subtropical | | 320 |
| Marítimo | | 350 |
| Equatorial | | 360 |
| Marítimo Subtropical | | 370 |
| **Condutividade do Solo (Siemens/m)** | | |
| Solo médio | | 0,005 |
| Solo pobre | | 0,001 |
| Solo rico | | 0,02 |
| Água doce | | 0,01 |
| Água salgada | | 5 |
| **Permissividade Relativa** | | |
| Solo médio | | 15 |
| Solo pobre | | 4 |
| Solo rico | | 25 |
| Água doce | | 81 |
| Água salgada | | 81 |
| **Localização do Sistema** | | |
| **Tipo de Escolha** | **Descrição** | |
| Aleatória | Antenas situadas de forma aleatória | |
| Cuidada | Antenas situadas em posições elevadas | |
| Muito cuidada | Antenas situadas em posições elevadas para maior potência de sinal em locais específicos. | |
| **Parâmetros estatísticos** | | |
| **Nível de confiabilidade** | | 0,1% a 99,9% |

| **Cotas Inferiores** $(m)$ | $C_0$ | **Tipo** |
|---|---|---|

| | | |
|---|---|---|
| 0 a 400 | 0 | Planície |
| 0 a 400 | 3,5 | Colinas |
| 400 a 700 | 2,5 | Planície |
| 400 a 700 | 6 | Colinas |
| Maior 700 | 5,5 | Planície |
| Maior 700 | 8 | Colinas |
| Maior 700 | 10,5 | Montanhas |

# Anexo II

## Código Matlab 1

```
function varargout = Modelos2(varargin)
% MODELOS2 MATLAB code for Modelos2.fig
%      MODELOS2, by itself, creates a new MODELOS2 or raises the
existing
%      singleton*.
%
%      H = MODELOS2 returns the handle to a new MODELOS2 or the handle
to
%      the existing singleton*.
%
%      MODELOS2('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MODELOS2.M with the given input
arguments.
%
%      MODELOS2('Property','Value',...) creates a new MODELOS2 or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Modelos2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Modelos2_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Modelos2

% Last Modified by GUIDE v2.5 14-Jul-2015 12:51:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Modelos2_OpeningFcn, ...
                   'gui_OutputFcn',  @Modelos2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Modelos2 is made visible.
function Modelos2_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Modelos2 (see VARARGIN)

% Choose default command line output for Modelos2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Modelos2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Modelos2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in radiobutton4.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4

% --- Executes on button press in radiobutton4.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4

% --- Executes on button press in radiobutton3.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3

% --- Executes on button press in radiobutton5.
function radiobutton5_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton5

% --- Executes on button press in radiobutton5.
function radiobutton6_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton5

% --- Executes on button press in radiobutton6.
function radiobutton7_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton6

% --- Executes on button press in radiobutton7.
function radiobutton8_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton7


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2
a=(get(hObject,'Value'));

 if a == get(hObject,'Max')
     display('Selected');
    grid on;

 else
     display('Not selected');
    grid off;
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    N = str2num(get(handles.edit1,'String'));
    f = str2num(get(handles.edit8,'String'));
    TD = str2num(get(handles.edit22,'String'));
    D = str2num(get(handles.edit23,'String'));

%%%%%%%%%%%%%%%%%%%%%% Variaveis Fixas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

d=0.0:0.25:N;
VD=TD*D;
%TD  Diametro do tronco em cm
%D   Diametro do tronco em cm

set(handles.edit18,'String',num2str(VD,2))
```

```
%%%%%%%%%%%%%%%%%%% MODELOS DE FOLHAGEM E VEGETAÇÃO
%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%% Modelo em Espaço Livre
%%%%%%%%%%%%%%%%%%%%%%%%

f1=f/1000;
PLfs=32.45+20.*log10(d)+20.*log10(f1);
PLfs1=32.45+20.*log10(N)+20.*log10(f1)

%%%%%%%%%%%%%%%%%%%%%%%% Modelo de  Weissberger
%%%%%%%%%%%%%%%%%%%%%%%%

f1=f/1000;%frequency in GHz

if d >= 14
    PLwei=PLfs+(1.33*(f1^0.284).*((d).^0.588));
        PLwei1=PLfs1+(1.33*(f1^0.284)*((N)^0.588))
elseif d <=400
    PLwei=PLfs+(1.33*(f1^0.284).*((d).^0.588));
        PLwei1=PLfs1+(1.33*(f1^0.284)*((N)^0.588))
else
   PLwei=PLfs+(0.45*(f1^0.284).*(d));
       PLwei1=PLfs1+(0.45*(f1^0.284)*(N))
end

%%%%%%%%%%%%%%%%%%%% Modelo LITU
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PLlitu=PLfs+(0.48*(f^0.43).*((d).^0.13));
PLlitu1=PLfs1+(0.48*(f^0.43)*((N)^0.13))

%%%%%%%%%%%%%%%%%%%% Modelo Amadio & Santos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f1=f/1000;      %Frequency in MHz

if N <= 60
    dm=60;
    n=(0.043*((dm)-40).^0.47).*VD+((0.54.*(f1)^0.42)-0.45).*((dm)-
40).^-0.15+2;

    set(handles.edit19,'String',num2str(n,3))

    PLA_S= PLfs+((-0.026*(dm) + (0.49*((f1)^0.47))).*VD +
20*log10((4*pi*f1)/0.3));
    PLA_S1=PLfs1+((-0.026*(N) + (0.49*((f1)^0.47)))*VD +
20*log10((4*pi*f1)/0.3))

else
    n=(0.043*((d)-40).^0.47).*VD+((0.54.*(f1)^0.42)-0.45).*((d)-40).^-
0.15+2;
    n1=(0.043*((N)-40).^0.47).*VD+((0.54.*(f1)^0.42)-0.45).*((N)-
40).^-0.15+2;

    PLA_S= PLfs+((-0.026*(d) + (0.49*((f1)^0.47))).*VD +
20*log10((4*pi*f1)/0.3));
    PLA_S1= PLfs1+((-0.026*(N) + (0.49*((f1)^0.47))).*VD +
20*log10((4*pi*f1)/0.3))

    set(handles.edit19,'String',num2str(n1,3))
```

```
end

%%%%%%%%%%%%%%%%%%%% Modelo FITU-R
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%sem folhagem
PLfitu=PLfs+(0.37*(f^0.18).*((d).^0.59));
PLfitu3=PLfs1+(0.37*(f^0.18)*((N)^0.59))

%com folhagem
PLfitu2=PLfs+(0.39*(f^0.39).*((d).^0.25));
PLfitu1=PLfs1+(0.39*(f^0.39)*((N)^0.25))

%%%%%%%%%%%%%%%%%%%% Modelo COST 235
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%sem folhagem
PLcost=PLfs+(26.6*(f^-0.2).*((d).^0.5));
PLcost3=PLfs1+(26.6*(f^-0.2)*((N)^0.5))

%com folhagem
PLcost2=PLfs+(15.6*(f^-0.009).*((d).^0.26));
PLcost1=PLfs1+(15.6*(f^-0.009)*((N)^0.26))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

 aa = 1;% valor on/off modelo Espaço Livre - modelo de referência
 bb = get(handles.radiobutton2,'Value');% valor on/off modelo
Weissberger
 cc = get(handles.radiobutton3,'Value');% valor on/off modelo LITU-R
 dd = get(handles.radiobutton4,'Value');% valor on/off modelo Amandio
& Santos
 ee = get(handles.radiobutton5,'Value');% valor on/off modelo FITU-R
 ff = get(handles.radiobutton6,'Value');% valor on/off modelo FITU-R
s\ folhas
 gg = get(handles.radiobutton7,'Value');% valor on/off modelo COST 235
 hh = get(handles.radiobutton8,'Value');% valor on/off modelo COST 235
s\ folhas

 xx=bb+cc+dd+ee+ff+gg+hh;

plot(d,PLfs,'r','DisplayName','Espaço Livre','LineWidth',2);
hold on;

if(bb==1)
plot(d,PLwei,'-k','DisplayName','Weisemberger');
else
    set(plot(d,PLwei,'-
k','DisplayName','Weisemberger'),'HandleVisibility','off', 'visible',
'off')
end

if(cc==1)
plot(d,PLlitu,'-g','DisplayName','LITU-R');
else
    set(plot(d,PLlitu,'-g','DisplayName','LITU-
R'),'HandleVisibility','off', 'visible', 'off')
end
```

```
if(dd==1)
plot(d,PLA_S,'-w','DisplayName','Amandio & Santos');
else
    set(plot(d,PLA_S,'-w','DisplayName','Amandio &
Santos'),'HandleVisibility','off', 'visible', 'off')
end

if(ee==1)
plot(d,PLfitu,'-b','DisplayName','FITU-R');
else
    set(plot(d,PLfitu,'-b','DisplayName','FITU-
R'),'HandleVisibility','off', 'visible', 'off')
end

if(ff==1)
plot(d,PLfitu2,'-c','DisplayName','FITU-R C/ Folhagem');
else
    set(plot(d,PLfitu2,'-c','DisplayName','FITU-R C/
Folhagem'),'HandleVisibility','off', 'visible', 'off')
end

if(gg==1)
plot(d,PLcost,'-m','DisplayName','COST 235');
else
    set(plot(d,PLcost,'-m','DisplayName','COST
235'),'HandleVisibility','off', 'visible', 'off')
end

if(hh==1)
plot(d,PLcost2,'-y','DisplayName','COST 235 C/ Folhagem');
else
    set(plot(d,PLcost2,'-y','DisplayName','COST 235 C/
Folhagem'),'HandleVisibility','off', 'visible', 'off')
end

hold off;
legend show
set(legend,'FontAngle','italic');

%'LineWidth',2);

 set(findobj(gca,'Type','Line','Color',[1 1 1]),'Color',[1,0.7,0.7])
 set(findobj(gca,'Type','Line','Color',[1 0 0]))

if xx==0
     set(handles.edit5,'String',num2str(PLfs1,4))
 else
    if xx==1

    if bb==1
              if d >= 14
                 set(handles.edit5,'String',num2str(PLwei1,4))
              elseif d <=400
                 set(handles.edit5,'String',num2str(PLwei1,4))
              else
                 set(handles.edit5,'String',num2str(PLwei1,4))

              end
 elseif cc==1
     set(handles.edit5,'String',num2str(PLlitu1,4))
 elseif dd==1
```

```
     set(handles.edit5,'String',num2str(PLA_S1,3));
 elseif ee==1
     set(handles.edit5,'String',num2str(PLfitu3,4))
 elseif ff==1
    set(handles.edit5,'String',num2str(PLfitu1,4))
 elseif gg==1
    set(handles.edit5,'String',num2str(PLcost3,4))
 elseif hh==1
    set(handles.edit5,'String',num2str(PLcost1,4))
 end

 else
     set(handles.edit5,'String',num2str(0));
 end

end

at1 = get(handles.edit5,'String')
at=str2num(at1)
ats1= get(handles.edit62,'String')
ats=str2num(ats1)
atotal=at+ats
set(handles.edit61,'String',num2str(atotal))
%%%%%%%%%%%%%%%%%% Eixos e Legendas dos Gráficos
%%%%%%%%%%%%%%%%%%%%%%%%

set(handles.figure1,'CurrentAxes',handles.axes2);
set(gca,'YDir','Reverse') %inverte o eixo dos YY
gr=get(handles.checkbox2,'Value');
     if gr==1
        grid on;
    else
        grid off;
    end

    title('Modelos de Propagação em Vegetação');
            xlabel 'Distância entre o Tx e Rx (m)'
              ylabel 'Atenuação (dB)'


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18
as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22
as a double

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23
as a double

% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3

% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4

function edit44_Callback(hObject, eventdata, handles)
% hObject    handle to edit44 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit44 as text
%        str2double(get(hObject,'String')) returns contents of edit44
as a double

% --- Executes during object creation, after setting all properties.
function edit44_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit44 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit45_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit45 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit45 as text
%        str2double(get(hObject,'String')) returns contents of edit45
as a double

% --- Executes during object creation, after setting all properties.
function edit45_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit45 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit54_Callback(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit54 as text
%        str2double(get(hObject,'String')) returns contents of edit54
as a double

% --- Executes during object creation, after setting all properties.
function edit54_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit47_Callback(hObject, eventdata, handles)
% hObject    handle to edit47 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit47 as text
%        str2double(get(hObject,'String')) returns contents of edit47
as a double

% --- Executes during object creation, after setting all properties.
function edit47_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit47 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit46_Callback(hObject, eventdata, handles)
% hObject    handle to edit46 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit46 as text
%        str2double(get(hObject,'String')) returns contents of edit46
as a double

% --- Executes during object creation, after setting all properties.
function edit46_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit46 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit55_Callback(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit55 as text
%        str2double(get(hObject,'String')) returns contents of edit55
as a double

% --- Executes during object creation, after setting all properties.
function edit55_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit56_Callback(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit56 as text
%        str2double(get(hObject,'String')) returns contents of edit56
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit56_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit57_Callback(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit57 as text
%        str2double(get(hObject,'String')) returns contents of edit57
as a double

% --- Executes during object creation, after setting all properties.
function edit57_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit58_Callback(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit58 as text
%        str2double(get(hObject,'String')) returns contents of edit58
as a double

% --- Executes during object creation, after setting all properties.
function edit58_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit59_Callback(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit59 as text
%        str2double(get(hObject,'String')) returns contents of edit59
as a double

% --- Executes during object creation, after setting all properties.
function edit59_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit60_Callback(hObject, eventdata, handles)
% hObject    handle to edit60 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit60 as text
%        str2double(get(hObject,'String')) returns contents of edit60
as a double

% --- Executes during object creation, after setting all properties.
function edit60_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit60 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7

function edit61_Callback(hObject, eventdata, handles)
% hObject    handle to edit61 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit61 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit61
as a double

% --- Executes during object creation, after setting all properties.
function edit61_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit61 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit62_Callback(hObject, eventdata, handles)
% hObject    handle to edit62 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit62 as text
%        str2double(get(hObject,'String')) returns contents of edit62
as a double

% --- Executes during object creation, after setting all properties.
function edit62_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit62 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

# Anexo III

## Código MATLAB 2

```
function varargout = Modelos_tese_22(varargin)
% MODELOS_TESE_22 M-file for Modelos_tese_22.fig
%      MODELOS_TESE_22, by itself, creates a new MODELOS_TESE_22 or
raises the existing
%      singleton*.
%
%      H = MODELOS_TESE_22 returns the handle to a new MODELOS_TESE_22
or the handle to
%      the existing singleton*.
%
%      MODELOS_TESE_22('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MODELOS_TESE_22.M with the given
input arguments.
%
%      MODELOS_TESE_22('Property','Value',...) creates a new
MODELOS_TESE_22 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before testegui_OpeningFunction gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Modelos_tese_22_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Modelos_tese_22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Modelos_tese_22_OpeningFcn, ...
                   'gui_OutputFcn',  @Modelos_tese_22_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Modelos_tese_22 is made visible.
function Modelos_tese_22_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Modelos_tese_22 (see VARARGIN)

% Choose default command line output for Modelos_tese_22
%Variaveis Globais acessiveis em qualquer funçao
handles.output = hObject;
handles.recta=0;
handles.perfil = 0;
[long0,long1,lat0,lat1,amostragem,nome] = getconfig(2);
set(handles.amostragem,'String',num2str(10*amostragem));

% Update handles structure
guidata(hObject, handles);
%pixels

handle_figure=gcf
 set(handle_figure, ...
'Units', 'pixels' );

 screenSize = get(0, 'ScreenSize');

 position = get(handle_figure, ...
 'Position' );
 position(1) = (screenSize(3)-position(3))/2;
 position(2) = (screenSize(4)-position(4))/2;

 set( handle_figure, ...
     'Position', position );

% --- Outputs from this function are returned to the command line.
function varargout = Modelos_tese_22_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --------------------------------------------------------------------
function sobre_Callback(hObject, eventdata, handles)
% hObject    handle to sobre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
procurar.
function procurar_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to procurar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end


function edit4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function frequencia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to frequencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end



function frequencia_Callback(hObject, eventdata, handles)
% hObject    handle to frequencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of frequencia as text
%        str2double(get(hObject,'String')) returns contents of
frequencia as a double
%get the string for the editText component


% --- Executes during object creation, after setting all properties.
function htx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to htx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end



function htx_Callback(hObject, eventdata, handles)
% hObject    handle to htx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of htx as text
%        str2double(get(hObject,'String')) returns contents of htx as
a double


% --- Executes during object creation, after setting all properties.
function hrx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hrx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

function hrx_Callback(hObject, eventdata, handles)
% hObject    handle to hrx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hrx as text
%        str2double(get(hObject,'String')) returns contents of hrx as
a double

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14
as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

% --- Executes on button press in radioknifeedge.
function radioknifeedge_Callback(hObject, eventdata, handles)
% hObject    handle to radioknifeedge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioknifeedge
state=get(handles.radioknifeedge,'Value');
if state==1
     set(handles.radioegli,'Value',0);
     set(handles.radiodeygout,'Value',0);
     set(handles.radioep,'Value',0);
     set(handles.radioitu,'Value',0);
      set(handles.radioLR,'Value',0);
     set(handles.radiotodos,'Value',0);
end
  set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','');

% --- Executes on button press in radioegli.
function radioegli_Callback(hObject, eventdata, handles)
% hObject    handle to radioegli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioegli
state=get(handles.radioegli,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radiodeygout,'Value',0);
     set(handles.radioep,'Value',0);
     set(handles.radioitu,'Value',0);
      set(handles.radioLR,'Value',0);
     set(handles.radiotodos,'Value',0);
end
  set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','');

% --- Executes on button press in radioitu.
function radioitu_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to radioitu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioitu
state=get(handles.radioitu,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radiodeygout,'Value',0);
     set(handles.radioegli,'Value',0);
     set(handles.radioep,'Value',0);
      set(handles.radioLR,'Value',0);
     set(handles.radiotodos,'Value',0);
end
  set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','');

% --- Executes on button press in radiodeygout.
function radiodeygout_Callback(hObject, eventdata, handles)
% hObject    handle to radiodeygout (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiodeygout
state=get(handles.radiodeygout,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radioep,'Value',0);
     set(handles.radioegli,'Value',0);
     set(handles.radioitu,'Value',0);
      set(handles.radioLR,'Value',0);
     set(handles.radiotodos,'Value',0);
end
     set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','')

% --- Executes on button press in radioep.
function radioep_Callback(hObject, eventdata, handles)
% hObject    handle to radioep (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioep
state=get(handles.radioep,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radiodeygout,'Value',0);
     set(handles.radioegli,'Value',0);
     set(handles.radioitu,'Value',0);
      set(handles.radioLR,'Value',0);
     set(handles.radiotodos,'Value',0);
end
  set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','');
```

```
% --- Executes on button press in radioLR.
function radioep_Callback(hObject, eventdata, handles)
% hObject    handle to radioLR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioLR
state=get(handles.radioep,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radiodeygout,'Value',0);
       set(handles.radioep,'Value',0);
     set(handles.radioegli,'Value',0);
     set(handles.radioitu,'Value',0);
     set(handles.radiotodos,'Value',0);
end
  set(handles.perdaprincipal,'String','');
     set(handles.perdadireita,'String','');
     set(handles.dbprincipal,'String','');
     set(handles.dbdireita,'String','');
% --- Executes on button press in radiotodos.
function radiotodos_Callback(hObject, eventdata, handles)
% hObject    handle to radiotodos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiotodos
state=get(handles.radiotodos,'Value');
if state==1
     set(handles.radioknifeedge,'Value',0);
     set(handles.radiodeygout,'Value',0);
     set(handles.radioegli,'Value',0);
     set(handles.radioitu,'Value',0);
     set(handles.radioep,'Value',0);
      set(handles.radioLR,'Value',0);
end


% --- Executes during object creation, after setting all properties.
function amostragem_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amostragem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end



function amostragem_Callback(hObject, eventdata, handles)
% hObject    handle to amostragem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of amostragem as text
%        str2double(get(hObject,'String')) returns contents of
amostragem as a double
k=str2num(get(handles.amostragem,'String'))/50;
if k<1
errordlg('O valor minimo da amostragem e de 50 m','ERRO','modal')
set(handles.amostragem,'String','1');
end

% --- Executes during object creation, after setting all properties.
function percentagem_CreateFcn(hObject, eventdata, handles)
% hObject    handle to percentagem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end

function percentagem_Callback(hObject, eventdata, handles)
% hObject    handle to percentagem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of percentagem as text
%        str2double(get(hObject,'String')) returns contents of
percentagem as a double


% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
cla(handles.axes2);
%legend(handles.axes2,'off')
% hObject    handle to calcular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    %recupera o perfil
    recta=handles.recta;
    perfil=handles.perfil
    %Frequencia
    frequencia=str2num(get(handles.frequencia,'String'));
    %Distancia
    distancia=str2num(get(handles.valordistancia,'String'));
    %HTX
    ht=str2num(get(handles.htx,'String'));
    %HRX
    he=str2num(get(handles.hrx,'String'));
    %Amostragem
    [long0,long1,lat0,lat1,amostragemm,nome] = getconfig(2);
    amostragem=str2num(get(handles.amostragem,'String'))/amostragemm;
    %Percentagem
    percentagem=str2num(get(handles.percentagem,'String'))/100;
    $percentagem=60/100;
```

```
    %se nao for escolhido nenhum modelo - nao desenha nenhum grafico
no segundo eixo
    %mostra erro
    if ( get(handles.radioep,'Value')==0 &&
get(handles.radiodeygout,'Value')==0 &&
get(handles.radioegli,'Value')==0 && get(handles.radioitu,'Value')==0
&& get(handles.radiotodos,'Value')==0)
        errordlg('Tem de escolher um modelo','ERRO','modal')
        uicontrol(hObject)

  end


if (get(handles.radioep,'Value')==1)
    tobj =
text('String','Elevaçao(m)','Units','pixels','FontSize',8,'Backgroundc
olor',[0.863,0.863,0.863]);
    set(tobj,'Rotation',90);
    set(tobj,'Position', [-35,-215]);
     %set(handles.shadow,'Visible','off');
   %modelo EpsteinPeterson
    [perfilfresnel,perfilmaximos] =
amostragemperfil(perfil,distancia,amostragem,frequencia,ht,he,percenta
gem)

            hold on
            %Perfilfresnel
            f=[1:distancia/length(perfilmaximos):distancia];
            p=area(f,perfilfresnel);
            set(p,'FaceColor',[.5 .5 .5]);
            set(p,'Parent', handles.axes2);
            set(gca,'XTickLabel',num2str(get(gca,'XTick').'));
            set(gca,'YTickLabel',num2str(get(gca,'YTick').'));
            maxi=max(perfilmaximos)+30;
            set(handles.axes2,'XLim',[0 distancia],'YLim',[0 maxi])
            aux1=distancia/length(perfilmaximos)
            a=1;
            %perfilmaximos=(perfilmaximos);
            f=[1:distancia/length(perfilmaximos):distancia];
            p=plot(f,perfilmaximos);
            set(p,'Parent', handles.axes2);
           %Desenha as linhas entre os valores de cota maxima
                   for i=1:length(perfilmaximos)%1
                           if perfilmaximos(i)>0%2
                              x(a)=i*aux1;
                              y(a)=perfilmaximos(i);
                              a=a+1;
                           end
                    end
                     for p=1:(length(x)-1)%3
                             r=[x(p),x(p+1)];
                             t=[y(p),y(p+1)];
                             r=plot(r,t,'g');
                             set(r,'Parent', handles.axes2)
                     end
    %Recta do tx ao 1 obstaculo
    r=[0,x(1)]
    t=[ht,y(1)]
    r=plot(r,t,'g');
    set(r,'Parent', handles.axes2)
    %Recta do 1 obstaculo ao rx
```

```
    r=[x(length(x)),distancia]
    t=[y(length(x)),he]
    r=plot(r,t,'g');
    set(r,'Parent', handles.axes2)
    perfilmaximos(1)=ht;
    perfilmaximos(length(perfilmaximos))=he;
     %calcular a perda do modelo do EspteiPeterson
     [loss] =
epsteinpeterson(perfilmaximos,distancia,frequencia,ht,he)
     set(handles.atenuacao,'String',sum(loss));
     perda = -1*perdalivre(distancia,frequencia)
     set(handles.perdalivre,'String',perda);
     set(handles.perdatotal,'String',(perda)+sum(loss));
     if(sum(loss)==0)
      errordlg('Nao existem obstaculos entre os pontos selecionados ou
a distancia de amostragem e demasiado elevada. A perda e a do espaço
livre.','Erro')
     end
      legend(handles.axes2,'Obst. Selectionados',4)
%Calcular a perda por difracçao segundo o modelo de Deygout
elseif (get(handles.radiodeygout,'Value')==1)
     tobj =
text('String','Elevaçao(m)','Units','pixels','FontSize',8,'Backgroundc
olor',[0.863,0.863,0.863]);
     set(tobj,'Rotation',90);
     set(tobj,'Position', [-35,-215]);
%    set(handles.shadow,'Visible','off');
    hold on
     [perfilfresnel,perfilmaximos] =
amostragemperfil(perfil,distancia,amostragem,frequencia,ht,he,percenta
gem)
       [loss,xpicos,ypicos] =
deygoutloss(perfilmaximos,distancia,frequencia,ht,he)
        set(handles.atenuacao,'String',loss(1));
         perda = -1*perdalivre(distancia,frequencia)
          set(handles.perdalivre,'String',perda);
           set(handles.perdatotal,'String',perda+sum(loss));
         set(handles.perdaprincipal,'String',loss(2));
        set(handles.perdadireita,'String',loss(3));
       set(handles.dbprincipal,'String','dB');
      set(handles.dbdireita,'String','dB');

%Calculo da localizaçao para os respectivos obstaculos
%selecionados
xpicos(1)=0
 xpicos(5)=distancia
  ypicos(1)=perfilmaximos(1)
ypicos(5)=perfilmaximos(length(perfilmaximos))

if(xpicos(4)==0)
    xpicos(4)=xpicos(5)
     ypicos(4)=ypicos(5)
end
if(xpicos(2)==0)
    xpicos(2)=xpicos(1)
     ypicos(2)=ypicos(1)
end
 ylabel('Elevaçao (m)')
   r=scatter(xpicos(3),ypicos(3),50,'*')
     set(r,'Parent', handles.axes2)
 legend(handles.axes2,'Obst. Principal',4)
```

```
%perfilfresnel
  f=[1:distancia/length(perfilmaximos):distancia];
     p=area(f,perfilfresnel);
    set(p,'FaceColor',[.5 .5 .5]);
  set(p,'Parent', handles.axes2)
  set(gca,'XTickLabel',num2str(get(gca,'XTick').'));
  set(gca,'YTickLabel',num2str(get(gca,'YTick').'));
 maxi=max(perfilmaximos)+30;
set(handles.axes2,'XLim',[0 distancia],'YLim',[0 maxi])
 %perfilmaximos=(perfilmaximos);
  f=[1:distancia/length(perfilmaximos):distancia];
   p=plot(f,perfilmaximos,'b');
    set(p,'Parent', handles.axes2)
   ypicos(1)=ypicos(1)+ht;
 ypicos(5)=ypicos(5)+he;
%Calculo da localizaçao os obstaculos
%selecionados
if(sum(loss)~=0)
 if(loss(1)~=0)
         r=[xpicos(1),xpicos(2)];
          t=[ypicos(1),ypicos(2)];
           r=plot(r,t,'g');
             set(r,'Parent', handles.axes2)

             r=[xpicos(2),xpicos(3)];
            t=[ypicos(2),ypicos(3)];
           r=plot(r,t,'g');
         set(r,'Parent', handles.axes2)


     else
         r=[xpicos(1),xpicos(3)];
          t=[ypicos(1),ypicos(3)];
           r=plot(r,t,'g');
            set(r,'Parent', handles.axes2)
end

if(loss(3)~=0)
           r=[xpicos(3),xpicos(4)];
          t=[ypicos(3),ypicos(4)];
         r=plot(r,t,'g');
        set(r,'Parent', handles.axes2)

         r=[xpicos(4),xpicos(5)];
          t=[ypicos(4),ypicos(5)];
           r=plot(r,t,'g');
            set(r,'Parent', handles.axes2)

     else
            r=[xpicos(3),xpicos(5)];
          t=[ypicos(3),ypicos(5)];
         r=plot(r,t,'g');
         set(r,'Parent', handles.axes2)
end

else
   errordlg('Nao existem obstaculos entre os pontos selecionados ou a
distancia de amostragem e demasiado elevada.A perda e a do espaço
livre.','Erro')
    uicontrol(hObject)
end
```

```
 hold off
        elseif (get(handles.radioitu,'Value')==1)
hold on

            %calcular o ITU
            d1=0;d2=0
              y=0;
              xponto=0;
              yponto=0;
              aux=distancia/length(perfil)
                  %Procura o maior ponto cuja altura acima esta acima
da linha de vista
                      for i=1:length(perfil)
                        h=knifeedgeh((i-1)*aux,distancia-(i-
1)*aux,perfil(1)+ht,perfil(length(perfil))+he,perfil(i))
                          if(h>y)
                              y=h;
                              xponto=i;
                              yponto=perfil(i);
                              d1=round((i-1)*aux);
                              d2=round(distancia-d1);
                          end
                       end
                       if d1==0
                            Ad=10;
                          yponto=max(perfil);
                       else
                       f=raiofresnel(d1,d2,frequencia);
                       Ad=abs((-20*y/f)+10);
                    end
                     ylabel('Elevação(m)')
                       set(handles.atenuacao,'String',-1*Ad);
                        perda = perdalivre(distancia,frequencia);
                        set(handles.perdalivre,'String',-1*perda);
                        set(handles.perdatotal,'String',-
1*((perda)+Ad));

                        k=[1:distancia/length(perfil):distancia];
                        p=area(k,perfil)
                        set(p,'FaceColor',[.5 .5 .5]);
                        set(p,'Parent', handles.axes2);
                        set(handles.axes2,'XLim',[0
distancia],'YLim',[0 max(perfil)+30])
                if( xponto ~= 0)
                     r=scatter(aux*xponto,yponto,50,'g','*');
                       set(r,'Parent', handles.axes2);
                     legend(handles.axes2,'Perfil do Terreno','Ponto
escolhido',4);
                else
                    r=scatter(aux*xponto,yponto,50,'g','*');
                     set(r,'Parent', handles.axes2);
                    legend(handles.axes2,'Perfil do Terreno','Nao
existeem obstaculos que impeçam a linha de vista',4);
                end
hold off
           elseif (get(handles.radioegli,'Value')==1)
            cla(handles.axes2);
          %set(handles.shadow,'Visible','off');
            hold on
```

```
              tobj = text('String','Perda
(dB)','Units','pixels','FontSize',8,'Backgroundcolor',[0.863,0.863,0.8
63]);
               set(tobj,'Rotation',90);
                set(tobj,'Position', [-32,-215]);
                    hb=3.2808*ht
                     hm=3.2808*he
                      distancia=0.00062137*distancia

L50=117+40*log10(distancia)+20*log10(frequencia)-20*log10(hb*hm);
                         perda3=L50;
                         %g=plot(distancia/0.00062137,L50,'*');
                          %set(g,'Parent', handles.axes2)
                           ylabel('Elevação(m)')
                          set(handles.atenuacao,'String','--');
                         set(handles.perdalivre,'String','--');
                        set(handles.perdatotal,'String',-1*L50);
                        %Grafico
                       t=[];

r=[0:0.00062137*amostragemm:distancia+0.00062137*100]
                       for i=1:1:length(r)
                        t(i)=[(117+40*log10(r(i))+20*log10(frequencia)-
20*log10(hb*hm))]
                       end
                       r=[0:amostragemm:distancia/0.00062137+100]
                     e=plot(r,t,'r');
                     set(e,'Parent', handles.axes2)
                    set(handles.axes2,'YLim',[0 max(t)+20])
                    legend(handles.axes2,'Perda em funçao da
distancia',1)
 hold off
              %Calcular as perdas segundo todos os modelos disponiveis
             elseif (get(handles.radiotodos,'Value')==1)
                          set(handles.atenuacao,'String','--');
                           set(handles.perdalivre,'String','--');
                            set(handles.perdatotal,'String','--');
                          %calcular todos os modelos
                          legend(handles.axes2,'on')
%EpsteinPeterson
    [perfilfresnel,perfilmaximos] =
amostragemperfil(perfil,distancia,amostragem,frequencia,ht,he,percenta
gem);
    perfilmaximos(1)=ht;
    perfilmaximos(length(perfilmaximos))=he;
    [loss] =
epsteinpeterson(perfilmaximos,distancia,frequencia,ht,he);
    perda1=sum(loss);
%Deygout
    [loss,xpicos,ypicos] =
deygoutloss(perfilmaximos,distancia,frequencia,ht,he)
    perda2=sum(loss);
%Egli
   hb=3.2808*ht
   hm=3.2808*he
   distancia=0.00062137*distancia
   L50=117+40*log10(distancia)+20*log10(frequencia)-20*log10(hb*hm);
   perda3=-1*L50;
   distancia=distancia/0.00062137; %reconverte a distancia para km
%ITU
   d1=0;d2=0;
```

```
    y=0;
    xponto=0;
    ponto=0;
    aux=distancia/length(perfil)

              for i=1:length(perfil)
                          h=knifeedgeh((i-1)*aux,distancia-(i-
1)*aux,perfil(1)+ht,perfil(length(perfil))+he,perfil(i))
                            if(h>y)
                                y=h;
                                xponto=i;
                                yponto=perfil(i);
                                d1=round((i-1)*aux);
                                d2=round(distancia-d1);
                            end
                        end
                        if d1==0
                          Ad=10;
                           yponto=max(perfil);
                         else
                         f=raiofresnel(d1,d2,frequencia);
                         Ad=abs((-20*y/f)+10);
                    end
                    Ad=-1*Ad;
                    perda4=Ad;
                           %livre
                           distancia
                           frequencia
                           perda = perdalivre(distancia,frequencia)
                           %desenhando os 4 valores
                           %Grafico
                           perda1=perda1+(-1*perda);
                           perda2=perda2+(-1*perda);
                           perda4=perda4+(-1*perda);
                           hold on
                                Z = [perda1,perda2,perda3,perda4];
                           Z=Z*-1;
                           k=max(Z)+20;
                           set(handles.axes2,'XLim',[0 5],'YLim',[0
k+40]);
                                h = bar(Z,0.4);
                           set(h,'Parent', handles.axes2)
                                str1 = {'Epstein Pettersen'};
                                t1=text(0.4,Z(1)+20,str1)
                                set(t1,'Parent', handles.axes2)
                                str2 = {'Deygout'};
                                t1=text(1.7,Z(2)+20,str2)
                                set(t1,'Parent', handles.axes2)
                                str3 = {'Egli'};
                                t1=text(2.9,Z(3)+20,str3)
                                set(t1,'Parent', handles.axes2)
                                str4 = {'ITU-R P530-12'};
                                t1=text(3.6,Z(4)+20,str4)
                                set(t1,'Parent', handles.axes2)

                            else
                              errordlg('Tem de escolher um
modelo','ERRO','modal')
                             uicontrol(hObject)
end
```

```
% --- Executes on button press in actualizar.
function actualizar_Callback(hObject, eventdata, handles)
% hObject    handle to actualizar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.axes1);
    x0=str2num(get(handles.TXN,'String'));
    y0=str2num(get(handles.TXW,'String'));
    x1=str2num(get(handles.RXN,'String'));
    y1=str2num(get(handles.RXW,'String'));
    htx=str2num(get(handles.htx,'String'));
    hrx=str2num(get(handles.hrx,'String'));

 [myline,distancia,recta,declive] = ilheu(x0,y0,x1,y1,hrx,htx);

 %imprime o perfil do terreno

 hold on
    ylabel('Elevação (m)')
    xlabel('Distancia (m)')
    myline=fliplr(myline);
    f=[1:distancia/length(myline):distancia]
    distancia
    p=area(f,myline);
    set(p,'FaceColor',[.5 .5 .5]);
    maxi=max(myline)+30
    set(gca,'XLim',[0 distancia],'YLim',[0 maxi])
     save perfil3d myline; %%%%%
    set(p,'Parent', handles.axes1)
    set(gca,'XTickLabel',num2str(get(gca,'XTick').'));
    set(gca,'YTickLabel',num2str(get(gca,'YTick').'));

%imprime a recta de linha de vista

      recta=fliplr(recta);
      f=[1:distancia/length(myline):distancia];
      r=plot(f,recta,'blue');
      set(r,'Parent', handles.axes1)

%imprime a elipse de fresnel

      aux1=distancia/length(myline)
      frequencia=str2num(get(handles.frequencia,'String'));
      ra=sqrt((recta(1)-recta(length(recta)))^2+distancia^2)/2;
      rb=17.3*sqrt(distancia/(4*frequencia));
      x=distancia/2;
      y=(recta(1)-recta(length(recta)))/2+recta(length(recta));
      declive
      set(ellipse(ra,rb,-declive,x,y,'green',500),'Parent',
handles.axes1);

hold off
  set(handles.valordistancia, 'String',distancia);
 handles.legenda1=legend(handles.axes1,'Perfil do Terreno','Linha de
Vista Tx-Rx','1ª Zona de Fresnel',4);

 %partilhando as variaveis
  handles.recta = recta;
   handles.perfil = myline;
```

```
    guidata(hObject, handles);

 tobj =
text('String','Elevaçao(m)','Units','pixels','FontSize',8,'Backgroundc
olor',[0.863,0.863,0.863]);
 set(tobj,'Rotation',90);
 set(tobj,'Position', [-35,-215]);

function TXN_Callback(hObject, eventdata, handles)
% hObject    handle to TXN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TXN as text
%        str2double(get(hObject,'String')) returns contents of TXN as
a double
TXN = str2double(get(hObject,'string'));
if isnan(TXN)
  errordlg('Tem de inserir um valor numerico','ERRO','modal')
  uicontrol(hObject)
     return
end

function TXW_Callback(hObject, eventdata, handles)
% hObject    handle to TXW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TXW as text
%        str2double(get(hObject,'String')) returns contents of TXW as
a
%        double
TXW = str2double(get(hObject,'string'));
if isnan(TXW)
  errordlg('Tem de inserir um valor numerico','ERRO','modal')
  uicontrol(hObject)
     return
end

function RXN_Callback(hObject, eventdata, handles)
% hObject    handle to RXN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of RXN as text
%        str2double(get(hObject,'String')) returns contents of RXN as
a
%        double
RXN = str2double(get(hObject,'string'));
if isnan(RXN)
  errordlg('Tem de inserir um valor numerico','ERRO','modal')
  uicontrol(hObject)
     return
end


function RXW_Callback(hObject, eventdata, handles)
% hObject    handle to RXW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of RXW as text
%        str2double(get(hObject,'String')) returns contents of RXW as
a
%        double
RXW = str2double(get(hObject,'string'));
if isnan(RXW)
  errordlg('Tem de inserir um valor numerico','ERRO','modal')
  uicontrol(hObject)
     return
end

function analiseperfil_Callback(hObject, eventdata, handles)
% hObject    handle to Sobre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handle_fig = gcf
testegui2;
pause(0.5);
close(handle_fig);

function analiseglobal_Callback(hObject, eventdata, handles)
% hObject    handle to Sobre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1
 ylabel('Elevaçao (m)');
 xlabel('Distancia (m)');

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes2
ylabel('Elevaçao (m)');
xlabel('Distancia (m)');
%tobj = text('String','Elevaçao(m)','Units','pixels','FontSize',8);
%set(tobj,'Rotation',90);
%set(tobj,'Position', [-35,-215]);

% --- Executes during object creation, after setting all properties.
function numero_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if(get(0,'userdata')==0)

if ispc
```

```
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')
);
end
end


% --- Executes on button press in mapa.
function mapa_Callback(hObject, eventdata, handles)
% hObject    handle to mapa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
map
load pontos.xyz
x=fix(pontos(:,1));
y=fix(pontos(:,2));
set(handles.TXN, 'String', num2str(fix(x(1))));
set(handles.TXW, 'String', num2str(fix(y(1))));
set(handles.RXN, 'String', num2str(fix(x(2))));
set(handles.RXW, 'String', num2str(fix(y(2))));

% --------------------------------------------------------------------
function analisedecobertura_Callback(hObject, eventdata, handles)
% hObject    handle to analisedecobertura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handle_fig = gcf
cobertura;
pause(0.5);
close (handle_fig);

% --------------------------------------------------------------------
function vermapa_Callback(hObject, eventdata, handles)
% hObject    handle to vermapa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function doisd_Callback(hObject, eventdata, handles)
% hObject    handle to doisd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mapa_2d;
% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox5
```

```
% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox6

function edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit29 as text
%        str2double(get(hObject,'String')) returns contents of edit29
as a double

% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit30_Callback(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit30 as text
%        str2double(get(hObject,'String')) returns contents of edit30
as a double

% --- Executes during object creation, after setting all properties.
function edit30_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit31_Callback(hObject, eventdata, handles)
% hObject    handle to amostragem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of amostragem as text
```

```
%        str2double(get(hObject,'String')) returns contents of
amostragem as a double

% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amostragem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32
as a double

% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
}
```

# Anexo IV

## Estrutura de tramas específicas API do XBee

# Anexo V

## Código do Nó sensor

```
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                                                                    //
//                     Código do Nó Sensor                            //
//                                                                    //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

*************************** CIRCUITO**********************************

***************************** TESAO PV *******************************
      SOLAR:          pin A6

************************** TESAO BATERIAS ****************************
      SOLAR:          pin A4

************************** Kit MPL115A1 ******************************

     SCK:          pin 13
     SDO/DOUT/MISO:  pin 12
     SDI/DIN/MOSI:   pin 11
     CSN:            pin 10     (default, configurable)
     SHDN:           pin 7

************************** Sensor SHT11 ******************************
      CLK:         pin 5
      DataPin:          pin 4

************************** Sensor LUX ********************************
      LUX:           pin A3

**********************************************************************
      Alarme:        pin 3      interrupção usada como alarme
      XBee_DTR:      pin 2      pin de adormecimento do  Modulo XBee

////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                  Inicialização das bibliotecas                     //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

#include <avr/sleep.h>
#include <SHT1x.h>
#include <SPI.h>


#define dataPin 4
#define clockPin 5
#define XBee_DTR 2

#define PRESH   0x80
#define PRESL   0x82
#define TEMPH   0x84
#define TEMPL   0x86c
```

```
#define A0MSB   0x88
#define A0LSB   0x8A
#define B1MSB   0x8C
#define B1LSB   0x8E
#define B2MSB   0x90
#define B2LSB   0x92
#define C12MSB  0x94
#define C12LSB  0x96
#define CONVERT   0x24
#define chipSelectPin 10
#define shutDown 7

float A0_;
float B1_;
float B2_;
float C12_;
double aux;
int a=0;
int ledG = 6;
int ledR = 8;

// Inicializar os dados
         byte trama[] =
{0x7E,0x00,0x22,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF
,0xFE,0x00,0x00,//16
                         0x23,0x00,0x00,// 18 e 19 Tensão PV
                         0x26,0x00,0x00,// 21 e 22 Bateria
                         0x26,0x00,0x00,// 24 e 25 Temperatura
                         0x26,0x00,0x00,// 27 e 28 Humidade
                         0x26,0x00,0x00,// 30 e 31 Luminosidade
                         0x26,0x00,0x00,// 33 e 34 Pressão
                         0x26,0x00,     // 36      Flag
                         0x00};         // 37      CheckSum

  SHT1x sensor (dataPin, clockPin);

/* Timer2 reload value, globally available */
  unsigned int tcnt2;
/* Toggle HIGH or LOW digital write */
//int toggle = 0;

  int seconds = 10; // Período em segundos
  int start = seconds*30;
  int counter = 0;
  int solar = 0;
  int battery = 0;
  int temp_c=0;                       // int temperatura
  int humidity=0;                     // int humidade
  signed long lux=0                   // int luminosidade
  byte checksum = 0;
  int flag =0;


// Interrupt Service Routine (ISR)
void alarme ()
{
  flag = 1;
}               // end of isr
```

```
////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                                SETUP                               //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////


void setup()
{
  Serial.begin(9600);                 // Inicialização da porta serie
  analogReference(INTERNAL);
  attachInterrupt (1, alarme, HIGH); // Adiciona uma interrupção pin 3
  SPI.begin();                        // start the SPI library:

  pinMode (XBee_DTR, OUTPUT);
  pinMode (ledG, OUTPUT);
  pinMode (ledR, OUTPUT);
  pinMode(chipSelectPin, OUTPUT);
  pinMode(shutDown, OUTPUT);


  digitalWrite(shutDown, HIGH);// initialize the data ready and chip
select pins:
  digitalWrite(chipSelectPin, HIGH);
  delay (10);

    // read registers that contain the chip-unique parameters to do
the math
  unsigned int A0H = readRegister(A0MSB);
  unsigned int A0L = readRegister(A0LSB);
         A0_ = (A0H << 5) + (A0L >> 3) + (A0L & 0x07) / 8.0;

  unsigned int B1H = readRegister(B1MSB);
  unsigned int B1L = readRegister(B1LSB);
          B1_ = ( ( ( (B1H & 0x1F) * 0x100)+B1L) / 8192.0) - 3 ;

  unsigned int B2H = readRegister(B2MSB);
  unsigned int B2L = readRegister(B2LSB);
          B2_ = ( ( ( (B2H - 0x80) << 8) + B2L) / 16384.0 ) - 2 ;

  unsigned int C12H = readRegister(C12MSB);
  unsigned int C12L = readRegister(C12LSB);
          C12_ = ( ( ( C12H * 0x100 ) + C12L) / 16777216.0 )  ;


/* First disable the timer overflow interrupt while we're configuring
*/
TIMSK2 &= ~(1<<TOIE2);
/* Configure timer2 in normal mode (pure counting, no PWM etc.) */
TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
TCCR2B &= ~(1<<WGM22);
/* Select clock source: internal I/O clock */
ASSR &= ~(1<<AS2);
/* Disable Compare Match A interrupt enable (only want overflow) */
TIMSK2 &= ~(1<<OCIE2A);
/* Now configure the prescaler to CPU clock divided by 1024 */
TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20); // Set bits
//TCCR2B &= ~(1<<CS21); // Clear bit
/* We need to calculate a proper value to load the timer counter.
* The following loads the value 131 into the Timer 2 counter register
* The math behind this is:
```

```
* (CPU frequency) / (prescaler value) = 125000 Hz = 8us.
* (desired period) / 8us = 125.
* MAX(uint8) + 1 - 125 = 131;
*/
/* Save value globally for later reload in ISR */
tcnt2 = 0;
/* Finally load end enable the timer */
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
set_sleep_mode(SLEEP_MODE_PWR_SAVE);
sleep_enable();
}
////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------//
//                                ENDSETUP                           //
//-------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

/*
* Install the Interrupt Service Routine (ISR) for Timer2 overflow.
* This is normally done by writing the address of the ISR in the
* interrupt vector table but conveniently done by using ISR() */
ISR(TIMER2_OVF_vect) {
/* Reload the timer */
TCNT2 = tcnt2;
/* Write to a digital pin so that we can confirm our timer */
//digitalWrite(2, toggle == 0 ? HIGH : LOW);
//toggle = ~toggle;
counter++;
}

void readAndSend(){

  Serial.println("");

// amostra o valor da tensão a saída do painel solar

  solar = analogRead(A6);
  solar = map (solar,0,1023,0,7450);
  trama[18] = battery >> 8 & 0xff;
  trama[19] = battery & 0xff;
  Serial.print("Tensão no painel = ");
  Serial.println(solar,DEC);

// amostra o valor da bateria no divisor resistivo
  battery = analogRead(A4);
  battery = map (battery,0,1023,0,4526);
  trama[21] = battery >> 8 & 0xff;
  trama[22] = battery & 0xff;
  Serial.print("Bataria = ");
  Serial.println(battery,DEC); // debug valor da bateria

// Amostra o valor da Temperatura
  temp_c = 100*sensor.readTemperatureC(); //Temperatura em graus
Celcius
  trama[24] = temp_c >> 8 & 0xff;
  trama[25] = temp_c & 0xff;
  Serial.print("Temperatura = ");
  Serial.println(temp_c,DEC);

// Amostra o valor da humidade
```

```
  humidity = 100*sensor.readHumidity(); //Humidade em percentagem
  trama[27] = humidity >> 8 & 0xff;
  trama[28] = humidity & 0xff;
  Serial.print("Humidade = ");
  Serial.println(humidity,DEC);

// Amostra o valor da luminosidade

  lux = analogRead(A2); // Amostra do ADC 0 correspondente ao Sensor
de lumonosidade
  lux = map (lux,0,1023,0,12000);
  trama[30] = lux >> 8 & 0xff;
  trama[31] = lux & 0xff;
  Serial.print("Lumonosidade = ");
  Serial.println(lux)*100;

  // Amostra o valor da Pressão

  aux = (baropPessure()*10)/4;
  a=aux*100;
  trama[33] = a >> 8 & 0xff;
  trama[34] = a & 0xff;
  Serial.print("Pressao Atmosferica = ");
  Serial.println((aux)*4);

  trama[36] = flag;
  Serial.print("Alarme = ");
  Serial.println(flag);

 digitalWrite(XBee_DTR, LOW); // Liga xbee
 digitalWrite (ledG,HIGH);
 delay(15); //espera que acorde

// Cálculo do checksum
checksum = 0;
  for (int i = 3; i < sizeof(trama)-1; i++) {
    checksum+= trama[i]; // Soma os valores que contam para checksum
  }
  trama[37] = 0xFF - checksum; // Realiza o complemento para 2
  Serial.write(trama, sizeof(trama)); // Envia a trama pela porta
série
  delay(100);

  digitalWrite(XBee_DTR, HIGH); //XBee sleep
  digitalWrite (ledG,LOW);
}
////////////////////////////////////////////////////////////////////
//----------------------------------------------------------------//
//                              LOOP                              //
//----------------------------------------------------------------//
////////////////////////////////////////////////////////////////////

void loop() {

if(counter == start){
TIMSK2 &= ~(1<<TOIE2);

// executa a tarefa

readAndSend();
counter = 0;
```

```
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
}
// Sleep
sleep_enable();
sleep_mode();
sleep_disable();

//////////////////// ALARME //////////////

    if (flag==1){
      digitalWrite(ledR,HIGH);
      readAndSend();   // executa a tarefa
      counter = 0;
      TCNT2 = tcnt2;
      TIMSK2 |= (1<<TOIE2);
      digitalWrite(ledR,LOW);
    }
//////////////////////////////////////////
}

//leitura dos registos do Sensor Barométrico

unsigned int readRegister(byte thisRegister ) {
  unsigned int result = 0;   // result to return
  digitalWrite(chipSelectPin, LOW);
    delay(10);
    SPI.transfer(thisRegister);
    result = SPI.transfer(0x00);
  digitalWrite(chipSelectPin, HIGH);
  return(result);
}

//leitura da pressão
float baropPessure(){
  digitalWrite(chipSelectPin, LOW);
  delay(3);
    SPI.transfer(0x24);
    SPI.transfer(0x00);
    digitalWrite(chipSelectPin, HIGH);
    delay(3);
  digitalWrite(chipSelectPin, LOW);
    SPI.transfer(PRESH);
    unsigned int presH = SPI.transfer(0x00);
        delay(3);
    SPI.transfer(PRESL);
    unsigned int presL = SPI.transfer(0x00);
        delay(3);
    SPI.transfer(TEMPH);
    unsigned int tempH = SPI.transfer(0x00);
        delay(3);
    SPI.transfer(TEMPL);
    unsigned int tempL = SPI.transfer(0x00);
        delay(3);
    SPI.transfer(0x00);
      delay(3);
  digitalWrite(chipSelectPin, HIGH);

  unsigned long press = ((presH *256) + presL)/64;
  unsigned long temp  = ((tempH *256) + tempL)/64;
```

```
  float pressure = A0_+(B1_+C12_*temp)*press+B2_*temp;
  //método de correção dado pelo sensor
  float preskPa = pressure*  (65.0/1023.0)+50.0;

return(preskPa);
```

# Anexo VI

## Código do Nó Coordenador

```
////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                                                                    //
//                     Código do Nó Coordenador                       //
//                                                                    //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

//--------------------------------------------------------------------//
//                        Inicialização das bibliotecas               //
//--------------------------------------------------------------------//

 #include <SD.h>
//biblioteca referente ao cartão SD
 #include <SPI.h>
// biblioteca referente ao modo de comunicação SPI
 #include <Ethernet.h>
// biblioteca referente ao Modulo de Ethernet
 #include <EthernetUdp.h>
// biblioteca que permite a comunicação de pacotes por UDP
 #include <SimpleTimer.h>
// biblioteca referente ao temporizadores
 #include <Time.h>
// biblioteca referente ao relógio

//--------------------------------------------------------------------//
//                        Inicialização de Variáveis                  //
//--------------------------------------------------------------------//

 byte mac[] = {0x90, 0xA2, 0xDA, 0x0D, 0xD2, 0xC2};
// Mac address atribuido a porta de Ethernet
 IPAddress server(192,168,1,109);
// Endereço de Ip do Servidor para onde queremos enviar os dados
 EthernetClient client;
// Inicialização do servido de Cliente de Web

 int resetPin = 4;
// Pin utilizado para efetuar um reset ao arduíno
 int Pin8 = 8;
// Pin utilizado para acionar o relé do Módulo UBNT
 int Pin9 = 9;
// Pin utilizado para acionar o relé do Módulo UBNT
 int pushButton = 2;
// Pin utilizado para verificar o estado do botão
 int ledB = 7;
// Pin 7 LED Azul      (INTERNET OK)
 int ledG = 6;
// Pin 6 LED Verde     (SISTEMA  OK)
 int ledR = 5;
// Pin 5 LED Vermelho (ERRO)

 int connectLoop;
// Variável usada para fazer loop até a msg ser enviada
 int buttonState=0;
// Variável usada para registar o estado do botão
```

```
 int lastState=0;
// Variável usada para registar o ultimo estado do botão
volatile unsigned long releState;
// Variável usada para registar o estado do relé
boolean SDavailable;
// Variável utlizada para a verificação do cartão SD
boolean EthernetBegin;
// Variável utlizada para a verificação do módulo de Ethernet
char inChar;
// Variável utilizada para a entrada no vetor
char msgBuffer[100];
// Cria um buffer para armazenar os valores das tramas
byte Trama[70];
// Variável para guardar a trama do nó sensor num array de bytes
byte auxT[70];
// Cria uma copia do array da trama do nó sensor
byte rSSi[30];
// Variável para guardar a trama de RSSI num array de bytes
byte aux[30];
// Cria uma copia do array da trama de RSSI

byte Trama1[] =
{0x7E,0x00,0x0F,0x17,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFF,0xFF
,0xFE,0x01,0x44,0x42,0x65};
// Trama enviada para receber o valor de RSSI

File myFile;
// Cria o objeto que serve para registar os dados
SimpleTimer timer;
// Função que cria um timer para chamar as funções periódicas
int UBNT = timer.setInterval(1680000, LigaUBNT);
// Liga a antena UBNT de 28 em 28 minutos

////////////// Configurações do servidor de NTP ///////////////////////

IPAddress timeServer(194,117,9,130); // ntp02.oal.ul.pt

// O IP do servidor de tempo, pode ser
//escolhido conforme a disponibilidade
//IPAddress timeServer(194,117,9,136);// ntp04.oal.ul.pt
//IPAddress timeServer(130,149,17,21);// ntps1-0.cs.tu-berlin.de

const long timeZoneOffset = 0L + 3600;
// OffSet GMT + 1 Hora em segundos (hora local)
unsigned int ntpSyncTime = 0;
// Variável utilizada no sincronismo de relógio
unsigned int localPort = 8888;
// Variável com a porta de escuta por pacotes UDP
const int NTP_PACKET_SIZE= 48;
// Variável para guardar os primeiros 48 bytes com o time stamp
byte packetBuffer[NTP_PACKET_SIZE];
// Variável para guardar os pacotes de entrada e saída
EthernetUDP Udp;
// Função que permite o envio e receção de pacotes através UDP
unsigned long ntpLastUpdate = 0;
// Variável que regista a ultima atualização do servidor de NTP
time_t prevDisplay = 0;
// Variável com o ultimo valor de relógio

//////////////////////////////////////////////////////////////////////////
```

```
void Liga(){    // Função para ligar o relé (latching relay)
 digitalWrite(Pin8, LOW);
// Pin8 no estado LOW
 digitalWrite(Pin9, HIGH);
// Impulso no Pin9 para mudar para estado HIGH
 delay(15);
// delay de 15ms suficientes para mudar de estado
 digitalWrite(Pin9, LOW);
// Pin9 no estado LOW
 releState = 1;
// Estado do relé (1 para ligado)
}

void Desliga(){
// Função para desligar o relé
 digitalWrite(Pin9, LOW);
// Pin9 no estado LOW
 digitalWrite(Pin8, HIGH);
// Impulso no Pin8 para mudar para estado HIGH
 delay(15);
// delay de 15ms suficientes para mudar de estado
 digitalWrite(Pin8, LOW);
// Pin8 no estado LOW
 releState = 0;
// Estado do relé (0 para desligado)
}

void botao(){

  int buttonState = digitalRead(pushButton);
  //Serial.println(buttonState);
  delay(10);
// delay in between reads for stability
   if(buttonState == HIGH && buttonState != lastState){
    lastState = buttonState;
    timer.disable(UBNT);
    digitalWrite(ledB, HIGH);
    digitalWrite(ubnt, HIGH);
   releState = 1;
Liga();
}
  else{
  if(buttonState == LOW && buttonState != lastState){
  lastState = buttonState;
    digitalWrite(ledB, LOW);
    digitalWrite(ubnt, LOW);
  releState = 0;
Desliga();

timer.enable(UBNT);
  }
 }
}

void LigaUBNT(){

    digitalWrite(ledB, HIGH);
  Liga();
    Serial.println("UBNT LIGADO");
    Serial.println(releState);
}
```

```
void PedeRSSI(){

    digitalWrite(ledG, LOW);
    digitalWrite(ledB, HIGH);

    Serial.println("Enivio de pedido de RSSI");
    Serial.write(Trama1, sizeof(Trama1));
    Serial.println();
    digitalWrite(ledB, LOW);

    }

void timeset(){
 if(now()-ntpLastUpdate > ntpSyncTime) {
      int trys=0;
      while(!getTimeAndDate1() && trys<10){
        trys++;
      }
      if(trys<10){
        Serial.println("ntp server update success");
      }
      else{
        Serial.println("ntp server update failed");
      }
 }
}

void Web_SD_Send() {

// Ethernet shield and NTP setup

   Ethernet.begin(mac);
   int res_connect;
Serial.println("111111111");
    res_connect = client.connect(server, 80);
    Serial.println(res_connect);

   if (Ethernet.begin(mac) != false && res_connect != 0 )  {
   Serial.println(Ethernet.localIP());
   Serial.println("DHCP ok!");
   digitalWrite(ledB, HIGH);
   delay(200);

  if(SDavailable != false) {
// Caso não seja falso

    File myFile = SD.open("sdtest.txt",FILE_READ);
// Abre o ficheiro SDtest.txt
    int chPos = 0;
// Variável com a posição no vetor. (Igual a zero -> inicio da linha)
    int lineNo = 0;
// Variável com a linha inicial

    if(!myFile){
// Caso o ficheiro SDtest.txt não abra
      Serial.println("SD open fail");
// Imprime uma msg de erro
      digitalWrite(ledR, HIGH);
      digitalWrite(resetPin, LOW);
      delay(200);
```

```
// Não faz mais nada até a próxima interação
    }
       while(myFile.available())
// Enquanto o ficheiro estiver disponível
    {
      char ch = myFile.read();
// A variável ch passa a ser igual ao valor lido formando o vetor
      if(ch == '\n') {
// Caso o valor lido seja igual ao fim da linha
        chPos = 0;
// O próximo valor é igual a zero (Reinicializa o vetor)
        digitalWrite(ledB, HIGH);
        digitalWrite(ledG, HIGH);
        Serial.println(msgBuffer);
// Imprime o vetor no serial monitor
  lineNo++;
// Incrementa o número da linha
        Serial.print("Line number ");
        Serial.println(lineNo);
// Imprime o número de linha
        digitalWrite(ledB, LOW);
        digitalWrite(ledG, HIGH);
//---------------------------------WEB--------------------------------


       if(res_connect==1){
// Caso a ligação seja estabelecida no servidor
         client.connect(server, 80);
         client.print("GET /wsn_web2/save1.php?var=");
// Envia o vetor por html através da função "GET"
         client.print(msgBuffer);
         client.println("");
         client.println("Connection: close\r\n");
// Desliga a ligação após o envio do vetor

          while(client.connected())
// Enquanto o cliente estiver ligado
          {
            while(client.available())
// Enquanto existir resposta do servidor
            {
              inChar = client.read();
// A variável é preenchida pela resposta do servidor
              Serial.write(inChar);
// Imprime no serial monitor o valor recebido
              connectLoop = 0;
// Coloca o Loop a zero se recebe um pacote
            }
            connectLoop++;
// Incrementa o valor de Loop
            if(connectLoop > 100000){
// Caso passe mais de 10000 milissegundos desde a receção

// do último pacote
              Serial.println();
              Serial.println(F("Timeout"));
// Imprime uma msg de "timeout"
              client.stop();
// Desliga o cliente web
            }
              delay(1);
```

```
          }
          Serial.println();
          Serial.println(F("disconnecting."));
// Desliga o cliente após o envio completo
          client.stop();
// Desliga o cliente web
          digitalWrite(ledG, LOW);
          //client.flush();
//Discard any bytes that have been written to the client but not yet
read.
        }
        else{
// Caso contrário
          Serial.println("connection failure");
// Imprime uma msg de erro na ligação
          digitalWrite(ledR, HIGH);
          delay(50);
        }
      }
      else if(ch == '\r') {
        //fim do ficheiro
      }
      else if(chPos < 86) {
        msgBuffer[chPos] = ch;
        chPos++;
        msgBuffer[chPos] = 0;
      }
    }
    myFile.close();
    SD.remove("sdtest.txt");

    }
    Serial.print("Uptime (s): ");
    Serial.println(millis() / 1000);
    timeset();
    delay(100);

    Desliga();
    Serial.println("UBNT DESLIGADO");
    digitalWrite(ledB, LOW);
   }
   else{
     Serial.println("Não foi possível estabelecer ligação");
}
}

////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                                SETUP                               //
//                                                                    //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

void setup() {

digitalWrite(resetPin, HIGH);
// initialize the digital pin as an output.
  pinMode(pushButton, INPUT);
  pinMode(resetPin, OUTPUT);
  //pinMode(ubnt, OUTPUT);
```

```
  pinMode(ledR, OUTPUT);
  pinMode(ledG, OUTPUT);
  pinMode(ledB, OUTPUT);

Serial.begin(9600);
// Inicialização da porta Serie com baud rate de 9600

  timer.setInterval(1800000, EnviaDados);
// repete de 30 em 30 minutos
  timer.setInterval(1620000, PedeRSSI);
// repete de 27 em 27 minutos
  timer.enable(UBNT);
  pinMode(4,OUTPUT);
  pinMode(10,OUTPUT);
// SS W5100 no arduino
  pinMode(53, OUTPUT);
// ao utilizar o Arduíno Mega é preciso para funcionar o SPI

  pinMode(Pin8, OUTPUT);
  pinMode(Pin9, OUTPUT);
  digitalWrite(Pin8,LOW);
  digitalWrite(Pin9,LOW);

  digitalWrite(ledR, HIGH);
  delay(300);
  digitalWrite(ledR, LOW);
  digitalWrite(ledG, HIGH);
  delay(300);
  digitalWrite(ledG, LOW);
  digitalWrite(ledB, HIGH);
  delay(300);
  digitalWrite(ledB, LOW);

  Serial.println("Inicializando o Sistema!");
  digitalWrite(10, HIGH);

  digitalWrite(resetPin, HIGH);
  delay(200);

  Liga();
  Serial.println("A Ligar...");
  delay(10000);
// delay usado para obter endereço de Ip para atualizar o relogio

  digitalWrite(10,HIGH);
//desliga o chip W5100 de ETHERNET
  delay(50);

  if(!SD.begin(4)) {
// Caso o cartão não seja inicializado
   int i=0;
   int SDC=0;
   SDC=SD.begin(4);
   //Try to get the card to boot 10 times before giving up
   while( SDC == 0 && i < 10){
     delay(50);
     SDC = SD.begin(4);
     i++;
    Serial.println("SD card FAILED");
// Imprime msg de erro
    digitalWrite(ledR, HIGH);
```

```
    delay(200);
    SDavailable = false;
 // Cartão disponivel = falso
   }
        digitalWrite(resetPin, LOW);
// reinicia o arduino

  }
  else {
// Caso contrário
    SDavailable = true;
// Cartão desponivel = verdade
    Serial.println("SD card ok!");
// Imprime msg que o cartão está desponivel
    digitalWrite(ledG, HIGH);
    delay(200);
  }

////////////////////////////////////////////////////////////////////

  // Ethernet shield and NTP setup

   digitalWrite(4, HIGH);
// Cartão SD desligado
   delay(50);

   if (!Ethernet.begin(mac)) {
     EthernetBegin = false;
     Serial.println("falha ao adequerir DHCP!");
     digitalWrite(ledR, HIGH);
     delay(200);
  }
   else{
   EthernetBegin = true;
   Serial.println(Ethernet.localIP());
   Serial.println("DHCP ok!");
   digitalWrite(ledB, HIGH);
   delay(200);
   //Try to get the date and time
   int trys=0;
   while(!getTimeAndDate() && trys<10) {
     trys++;
   }
  Serial.println(now());

    Web_SD_Send();
     digitalWrite(ledR, LOW);
     digitalWrite(ledG, LOW);
     digitalWrite(ledB, LOW);
}
}



////////////////////////////////////////////////////////////////////
//-----------------------------------------------------------------//
//                              ENDSETUP                           //
//-----------------------------------------------------------------//
////////////////////////////////////////////////////////////////////
```

```
// Do not alter this function, it is used by the system
int getTimeAndDate() {

   int flag=0;
   Udp.begin(localPort);
   sendNTPpacket(timeServer);
   delay(1000);
   if (Udp.parsePacket()){
     Udp.read(packetBuffer,NTP_PACKET_SIZE);
// read the packet into the buffer
     unsigned long highWord, lowWord, epoch;
     highWord = word(packetBuffer[40], packetBuffer[41]);
     lowWord = word(packetBuffer[42], packetBuffer[43]);
     epoch = highWord << 16 | lowWord;
     epoch = epoch - 2208988800 + timeZoneOffset;
     flag=1;
     setTime(epoch);
     ntpLastUpdate = now();
   }
   return flag;
}

// Do not alter this function, it is used by the system
unsigned long sendNTPpacket(IPAddress& address)
{
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;
  Udp.beginPacket(address, 123);
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();
}

//////////////////////////////////////////////////////////////////////
//-----------------------------------------------------------------//
//                              FUNÇÕES                            //
//-----------------------------------------------------------------//
//////////////////////////////////////////////////////////////////////


void tempo(){
  prevDisplay = now();
}

void show_write_Trama(){

  digitalWrite(10,HIGH);

  myFile = SD.open("sdtest.txt",FILE_WRITE);
  if (myFile){
     if(Serial.read() == 0x7E){
      for (int i = 0; i < 2; i++){
       Serial.read();
            }
```

```
int A = Serial.read();

//--------------------------------------------------------------------//
//                Receção da Trama de Dados                           //
//--------------------------------------------------------------------//

if( A == 0x90){
// Verifica o primeiro byte da trama recebida é 0X90
                digitalWrite(ledG, HIGH);
          Serial.print("Trama Completa: ");
          Serial.print("7E,00,22,90,");
          myFile.print("7E002290");

      for (int j = 0; j <= 31; j++){
         Trama[j] = Serial.read(),HEX;
// Lê a trama recebida colocando-a num array de bytes em Hexadecimal
           auxT[j] = Trama[j];

     if(auxT[j]<=15){
                Serial.print('0');
                myFile.print('0');
                Serial.print(auxT[j],HEX);
                Serial.print(",");
                myFile.print(auxT[j],HEX);
   }
          else{
                Serial.print(auxT[j],HEX);
                Serial.print(",");
                myFile.print(auxT[j],HEX);
   }
 }

 Serial.print(" ");

    if (hour()<=15){
     Serial.print('0');
     myFile.print('0');
     Serial.print(hour(),HEX);
     myFile.print(hour(),HEX);
    }
    else{
     Serial.print(hour(),HEX);
     myFile.print(hour(),HEX);
    }
     //Serial.print(" ");

  if (minute()<=15){
     Serial.print('0');
     myFile.print('0');
     Serial.print(minute(),HEX);
     myFile.print(minute(),HEX);
     }
    else{
     Serial.print(minute(),HEX);
     myFile.print(minute(),HEX);
    }

  if (second()<=15){
     Serial.print('0');
     myFile.print('0');
     Serial.print(second(),HEX);
```

```
      myFile.print(second(),HEX);
      }
     else{
      Serial.print(second(),HEX);
      myFile.print(second(),HEX);
     }

  if (day()<=15){
     Serial.print('0');
     myFile.print('0');
     Serial.print(day(),HEX);
     myFile.print(day(),HEX);
          }
     else{
      Serial.print(day(),HEX);
      myFile.print(day(),HEX);
     }

  if (month()<=15){
     Serial.print('0');
     myFile.print('0');
     Serial.print(month(),HEX);
     myFile.print(month(),HEX);
     Serial.print(" ");}

     Serial.print('0');
     myFile.print('0');
     Serial.println(year(),HEX);
     myFile.println(year(),HEX);

     myFile.close(); // close the file
     digitalWrite(ledG, LOW);

}
else{

//-------------------------------------------------------------------//
//                          Receção da Trama de RSSI                 //
//-------------------------------------------------------------------//

if(A == 0x97){
// Verifica o primeiro byte da trama recebida é 0X97
          digitalWrite(ledG, HIGH);
          Serial.print("RSSI is: ");
          Serial.print("7E,00,10,97,");
          myFile.print("7E001097");

       for( int k = 0; k <= 15; k++){
          rSSi[k] = Serial.read(), HEX;
// Lê a trama recebida colocando-a num array de bytes em Hexadecimal
           aux[k] = rSSi[k];

        if(aux[k]<=15){

          Serial.print('0');
          myFile.print('0');
          Serial.print(aux[k],HEX);
          Serial.print(",");
          myFile.print(aux[k],HEX);
                    }
      else{
```

```
          Serial.print(aux[k],HEX);
          Serial.print(",");
          myFile.print(aux[k],HEX);
         }
    }
         Serial.println("");
         myFile.println("");

       myFile.close();
// close the file
         Serial.println();
         Serial.println(rSSi[14],HEX);
         Serial.print('-');
         Serial.print(rSSi[14],DEC);
         Serial.println("dB's");
         digitalWrite(ledG, LOW);
}
  }
    }
      }

  else{
    digitalWrite(ledG, LOW);
    digitalWrite(ledR, HIGH);
    Serial.println("error opening sdtest.txt");
// if the file didn't open, print an error:
    Serial.println(" Erro na abertura do ficheiro na receção de
tramas");
    digitalWrite(resetPin, LOW);
// reinicia o sistema
      }
}
////////////////////////////////////////////////////////////////////////
//void printDigits(int digits)
//{
// utility function for clock display:
//    Serial.print(":");
// prints preceding colon and leading 0
//  if(digits < 10)
//    Serial.print('0');
//    Serial.print(digits);
// }
////////////////////////////////////////////////////////////////////////
//void clockDisplay()     // Clock display of the time and date
(Basic)
//{
//     Serial.print(hour());
//    printDigits(minute());
//    printDigits(second());
//     Serial.print(" ");
//     Serial.print(day());
//     Serial.print(" ");
//     Serial.print(month());
//     Serial.print(" ");
//     Serial.print(year());
//     Serial.println();
//}
////////////////////////////////////////////////////////////////////////
//void clockDisplayHEX()
//{
//  if (hour()<=15)
```

```
//     Serial.print('0');
//     Serial.print(hour(),HEX);
//     Serial.print(" ");
//
//  if (minute()<=15)
//     Serial.print('0');
//     Serial.print(minute(),HEX);
//     Serial.print(" ");
//
//  if (second()<=15)
//     Serial.print('0');
//     Serial.print(second(),HEX);
//     Serial.print(" ");
//
//  if (day()<=15)
//     Serial.print('0');
//     Serial.print(day(),HEX);
//     Serial.print(" ");
//
//  if (month()<=15)
//     Serial.print('0');
//     Serial.print(month(),HEX);
//     Serial.print(" ");
//
//  if (year()<=15){
//     Serial.print(year(),HEX);
//                 }
//  else{
//    Serial.print('0');
//    Serial.print(year(),HEX);
//      }
//    Serial.println();
//}

////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//                                LOOP                                //
//--------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////

void loop() {

    timer.run();
// Update the time via NTP server as often as the time you set at the
top

    if (Serial.available() >= 17){
// Verifica se a trama recebida do coordenador é superior a 19 Bytes

       show_write_Trama();
}
botao();
 // Display the time if it has changed by more than a second.
    if( now() != prevDisplay){
      prevDisplay = now();

//      clockDisplay();
      //clockDisplayHEX();
    }
}
////////////////////////////////////////////////////////////////////////
```

```
//-------------------------------------------------------------------//
//                                LOOP END                           //
//-------------------------------------------------////////////////////
/////////////////////////////////////////////////////////////////////////

// Do not alter this function, it is used by the system
int getTimeAndDate1() {

  /* ALTER THESE VARIABLES AT YOUR OWN RISK */
// local port to listen for UDP packets
unsigned int localPort = 8888;
// NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE= 48;
// Buffer to hold incoming and outgoing packets
byte packetBuffer[NTP_PACKET_SIZE];
// A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;
// Keeps track of how long ago we updated the NTP server
unsigned long ntpLastUpdate = 0;
// Check last time clock displayed (Not in Production)
time_t prevDisplay = 0;
   int flag=0;
   Udp.begin(localPort);
   sendNTPpacket(timeServer);
   delay(1000);
   if (Udp.parsePacket()){
     Udp.read(packetBuffer,NTP_PACKET_SIZE);
// read the packet into the buffer
     unsigned long highWord, lowWord, epoch;
     highWord = word(packetBuffer[40], packetBuffer[41]);
     lowWord = word(packetBuffer[42], packetBuffer[43]);
     epoch = highWord << 16 | lowWord;
     epoch = epoch - 2208988800 + timeZoneOffset;
     flag=1;
     setTime(epoch);
     Serial.println(epoch);
     ntpLastUpdate = now();
   }
   return flag;
}
```

# Anexo VII

# Guia de instalação PowerStation2

## Instalação de Hardware

Na Figura VII.1 apresenta-se o método de ligação da antena à rede elétrica, o procedimento completo é descrito por:

1. Recomenda-se a configuração e programação das antenas em laboratório antes de proceder a montagem na sua localização final.
2. Após as configurações, montar a antena em local apropriado utilizando o kit de montagem fornecido juntamente com a antena.
3. Introduzir em primeiro lugar o cabo UTP de dados, no injetor na porta LAN.
4. Introduzir o cabo UTP na porta POE.
5. Introduzir o cabo UTP de POE na porta direita (a porta da esquerda é de dados).
6. Ligar o sistema à rede elétrica.

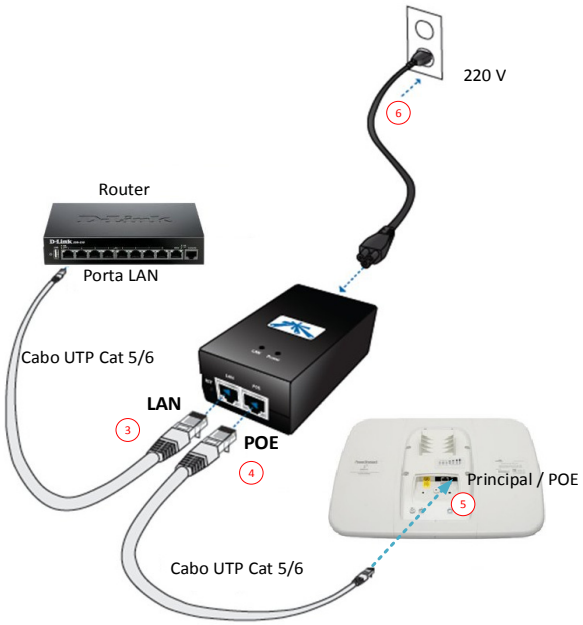



Figura VII.1 - Ligação da antena à rede elétrica.

## Configuração do endereço de IP

Este aparelho pode ser configurado como estação ou ponto de acesso. O IP de defeito para efetuar o login no aparelho é 192.168.1.20. No entanto, antes de se ligar, deve-se configurar as opções de TCP/IP da rede local do computador a utilizar.

### Configurações para utilizadores Windows

Na Figura VII.2 apresenta-se o procedimento a seguir para efetuar a configuração no sistema operativo Windows da Microsoft.

1. No painel de controlo, efetuar um duplo *click* no ícone "Centro de rede e partilha".
2. Utilizando o botão direito do rato aceder as propriedades.
3. Selecionar o ícone "Protocolo IP versão 4 (TCP/IPv4).
4. Selecionar o ícone "Utilizar o seguinte endereço de IP" e fixar o IP.

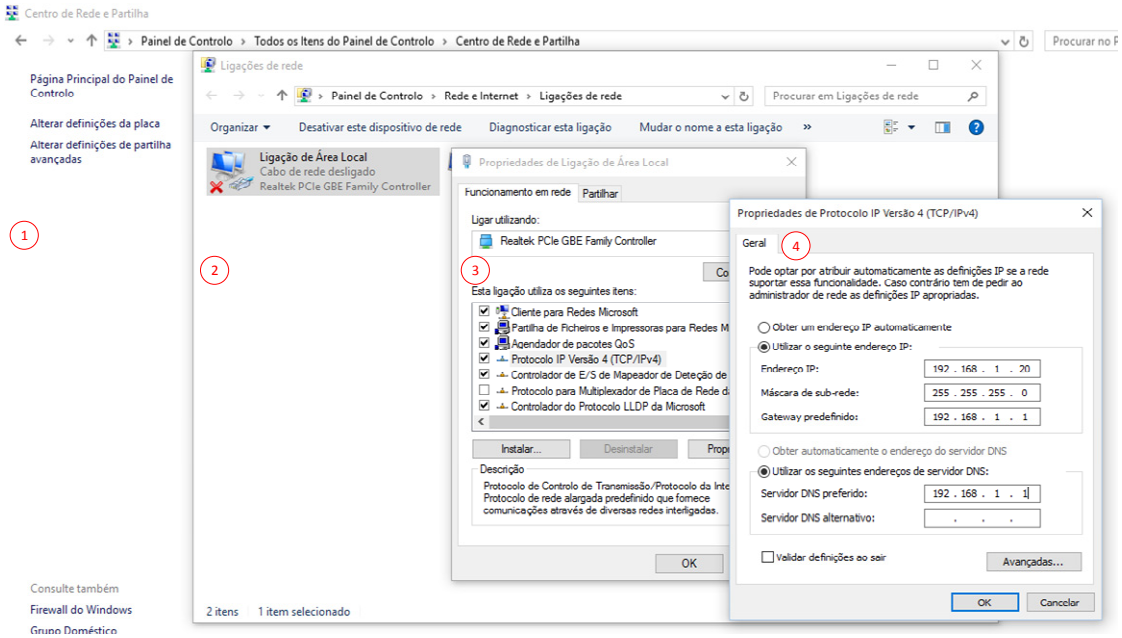



Figura VII.2 - Procedimento de configuração de Windows.

Um exemplo de configuração é:

Endereço IP antena: 192.168.1.20

Endereço IP do computador: 192.168.1.70

Máscara de sub-rede: 255.255.255.0

O servidor de DNS é opcional. Por defeito é igual ao Endereço de IP da *gateway*.

## Login na estação ou ponto de acesso

Na Figura VII.3 tem-se o processo de entrada da primeira utilização do sistema da airOS da Ubiquiti Networks. Neste primeiro login é necessário definir o país de utilização do sistema, devidas as regulamentações do espetro. Este acesso é efetuado seguindo os passos apresentados:

1. Introduzir o IP de defeito da antena UBNT no navegador: 192.168.1.20
2. Introduzir o nome de utilizador e a senha de defeito:

   User: **ubnt**
   Password: **ubnt**

3. Selecionar o país de utilização e idioma.
4. Por razões de segurança, após a configuração é recomendado a alteração da chave de segurança do sistema.

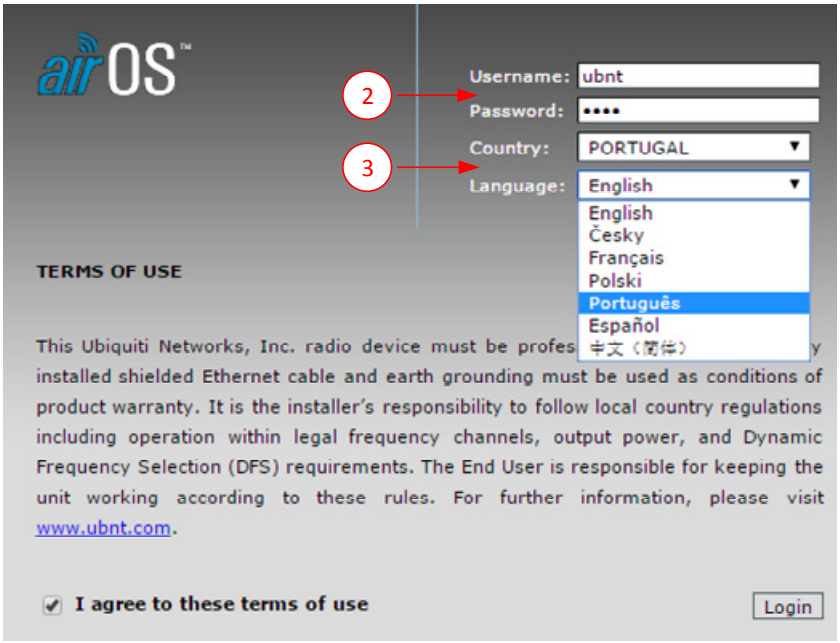



Figura VII.3 - Primeira configuração da antena UBNT.

**Configuração de uma ligação ponto a ponto.**

De seguida, na Figura VII.4 apresenta-se o processo de configuração para a uma ligação ponto a ponto, utilizando duas antenas da UBNT. Este processo segue os passos seguintes:

1. O primeiro passo passa por definir o modo de funcionamento da antena. Esta pode ser utilizada como *router* ou *bridge* (ponte). Neste caso pretende-se utilizar como ponte para ligar duas redes informáticas
2. No ponto de acesso é necessário definir o IP da rede na qual se pretende ligar.

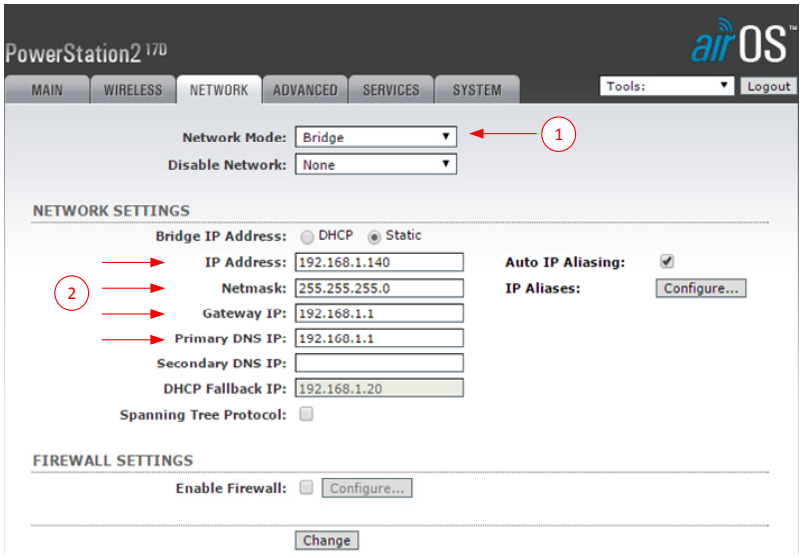



Figura VII.4 - Ligação ponto a ponto – rede (Access point).

Na Figura VII.5 apesenta-se as configurações de rede. O processo passa por:

1. Definir a antena como ponto de acesso, este costuma ser configurado do lado da rede na qual se pretende associar.
2. Dar um nome a rede, depois de configurado, para maior segurança pode optar-se por esconder o nome da rede.
3. Definir o canal em que se pretende transmitir, (pode ser deixado em auto).
4. É de salientar que após a definição do país de utilização, referido anteriormente, apenas é possível diminuir o valor de potência de transmissão.
5. Definir a taxa de transmissão, que depende do modo de funcionamento 802.11 G ou B. O modo auto gere automaticamente a velocidade de ligação de acordo com o sinal recebido.
6. Escolher uma *password* caso pretenda utilizar a função de WiFi da antena.

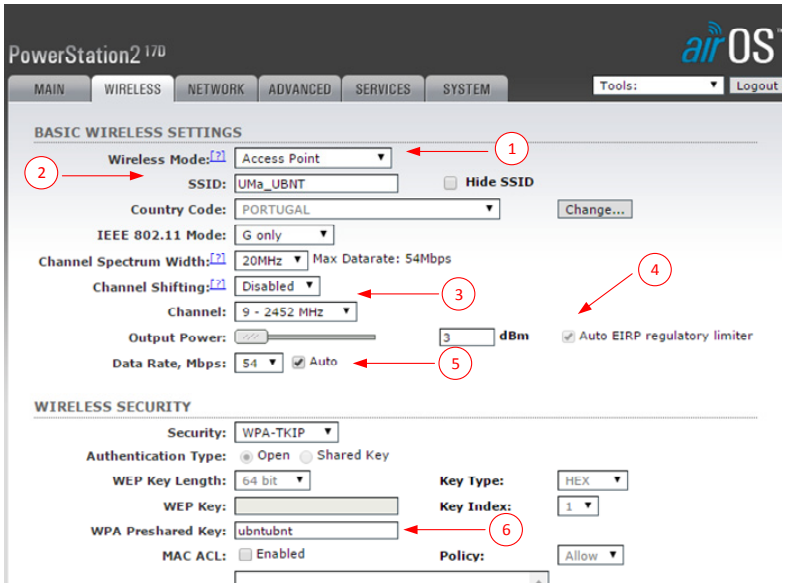



Figura VII.5 - Ligação ponto a ponto – WiFi (Access point).

Na Figura VII.6 apresenta-se a ligação da segunda antena ou estação. Os passo são descritos por:

1. Definir o modo de ponte.
2. É necessário definir o IP da rede na qual se pretende ligar e o IP da estação. Este não pode ser igual ao da estação de acesso nem de nenhum computador da rede.

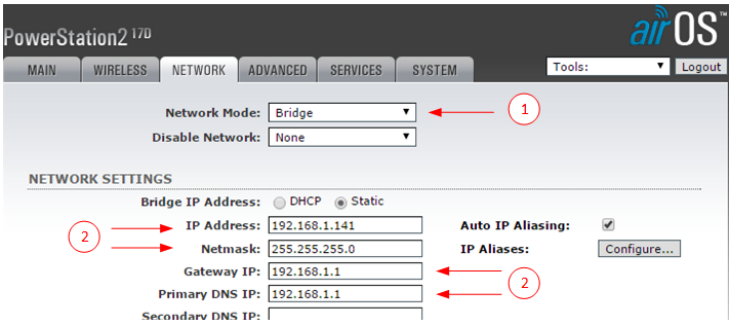



Figura VII.6 - Ligação ponto a ponto – rede (Station).

Na Figura VII.7 apresenta-se o modo de configuração da antena remota ou *station*. Este processo é dado por:

1. Definir o nome da rede na qual se pretende associar, este pode ser introduzido manualmente ou pode ser selecionado por varrimento das redes disponíveis.
2. Para facilitar e obter uma ligação mais robusta recomenda-se a introdução do *Mac ID* da antena na qual se presente associar a ligação. O *Mac ID* é fornecido pelo fabricante. ex: 00:15:6D:10:0C:47
3. Para acesso a rede WiFi torna-se necessário introduzir a mesma chave de segurança do ponto de acesso.

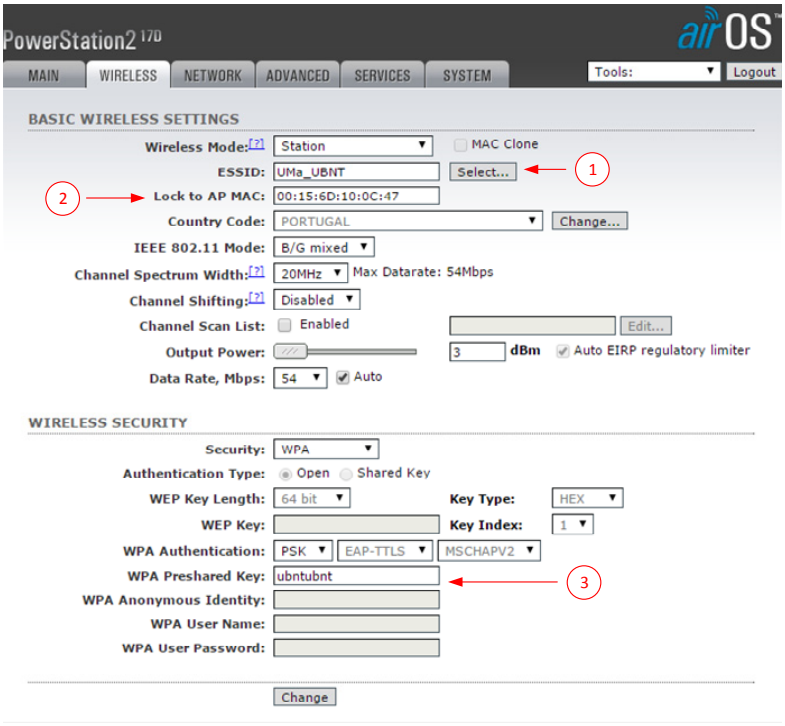



Figura VII.7 - Ligação ponto a ponto – WiFi (Station).

Finalizadas estas configurações têm-se uma ligação ponto a ponto a funcionar. O *software* da antena depois permite uma panóplia de funcionalidades extras, que depois poderão ser exploradas.

# Anexo VIII

## Código do PHP da plataforma de visualização de dados.

```
<?php

include('dbinfo.php');
require('mysql4.php');
$dbObject = new sql_db($serverIP, $username, $password, $database,
false);

     $var = "nada";
          if($var!=null){
               echo '1';
          if(isset($_GET['var'])){
     $var = $_GET['var'];
          //echo $var;

//echo "<br />Separacao de dados <br /><br />";

            $var_separado = str_split($var,2);
            $dados = array();

          if($var_separado[0] == '7E' && $var_separado[3]=='97'){

// receção de RSSI

$rssiAddress =
$var_separado[5].$var_separado[6].$var_separado[7].$var_separado[8].$
var_separado[9].$var_separado[10].$var_separado[11].$var_separado[12];

$rssi = hexdec($var_separado[18]);

$aQueryResult = $dbObject->sql_query("INSERT INTO icons (iconName)
VALUES ('".$var."');");

// introduz as tramas na Tabela
$aQueryResult = $dbObject->sql_query("INSERT INTO valuesrssi17 (value)
VALUES ('-".$rssi."');");

               }else{

                    for($i=16;$i<35;$i=$i+3){

                         $aux = hexdec($var_separado[$i])*256;
                         $aux = $aux+hexdec($var_separado[$i+1]);
                         $dados[]=$aux;
                    }

//[32] is checksum

     $hour = hexdec($var_separado[35]);
     $min = hexdec($var_separado[36]);
     $seg = hexdec($var_separado[37]);
     $day = hexdec($var_separado[38]);
     $month = hexdec($var_separado[39]);
     $year = hexdec($var_separado[40])*256 + hexdec($var_separado[41]);
```

```
$timestamp = $year."-".$month."-".$day.":".$hour.":".$min.":".$seg;
//echo $timestamp

//var_dump($dados);
     $aQueryResult = $dbObject->sql_query("INSERT INTO icons (iconName)
VALUES ('".$var."');");
// introduz as tramas na Tabela

          if($dados[0]< 7200 && $dados[0] > 3500){
// porque os restantes valores são outliers
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor11
(value,timestamp) VALUES ('".$dados[0]."','".$timestamp."');");
// tensão no painel solar
                    }
          if($dados[1]< 4280 && $dados[0] > 3500){
// porque os restantes valores são outliers
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor11
(value,timestamp) VALUES ('".$dados[1]."','".$timestamp."');");
 // tensão da bateria
                    }
          if($dados[2]< 4200){
// porque o valor recebido vem multiplicado por 100

     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor21
(value,timestamp) VALUES ('".round ($dados[2]*0.009891, 2)
."','".$timestamp."');");
 // temperatura
                    }
          if($dados[3]< 10000){
// porque o valor recebido vem multiplicado por 100
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor31
(value,timestamp) VALUES ('".round ($dados[3]*0.00778267, 2)
."','".$timestamp."');");
// humidade
                    }
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor41
(value,timestamp) VALUES ('".$dados[4]."','".$timestamp."');");
 // luminosidade

          if($dados[5]< 26000 && $dados[4] > 24250){
 // porque tem de ser devido ao valor ser multiplicado por .4
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor51
(value,timestamp) VALUES ('".round ($dados[5]*0.04027752, 2)
."','".$timestamp."');");
// presão
                    }
          if(hexdec($var_separado[31])< 2){
     $aQueryResult = $dbObject->sql_query("INSERT INTO valuesSensor61
(value,timestamp) VALUES('".$var_separado[31]."','".$timestamp."');");
 // flag
}
     }
          }
else{
     echo '0';
     }
}
?>
```