Dissertação

Mestrado em Engenharia Informática – Computação Móvel

Aplicação domótica para rastrear pessoas e produtos

Filipe Lopes Fernandes

Leiria, *julho* de *2024*

Dissertação

Mestrado em Engenharia Informática – Computação Móvel

Aplicação domótica para rastrear pessoas e produtos

Filipe Lopes Fernandes

Dissertação de Mestrado realizada sob a orientação do Doutor João da Silva Pereira, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e Investigador do Instituto de Telecomunicações.

Leiria, *julho* de 2024

Esta página foi intencionalmente deixada em branco

Resumo

Nesta pesquisa, examinamos as possibilidades de uso e vantagens das tecnologias de rastreamento e identificação, como antenas e *tags* RFID, em operações industriais e de fabricação. A complexidade destes ambientes exige soluções eficientes para a localização e identificação de cada objeto, com o objetivo de aumentar a produtividade, reduzir erros e aumentar a eficiência operacional. Assim, o propósito deste trabalho é desenvolver e analisar um sistema que permite localizar as RFID *tags* com precisão em ambientes industriais.

Ao abordar os diferentes tipos de *tags* e antenas disponíveis no mercado, é também apresentada uma revisão abrangente do estado da arte dessas tecnologias, incluindo estratégias e algoritmos para deteção e cálculo da posição das *tags*. O funcionamento da tecnologia OPC-UA, que é importante para a integração e comunicação entre os dispositivos do ambiente industrial, também é discutido para perceber o papel que este terá na solução apresentada.

Serão ainda examinados os hardwares e tecnologias utilizados no desenvolvimento do sistema proposto, reconhecendo as suas vantagens neste contexto em particular.

Além disso, o estudo inclui uma análise detalhada dos cenários e cálculos realizados para garantir o sucesso na determinação da posição das *tags* usando os dados captados pelas antenas. Os resultados dos testes são apresentados e discutidos, incluindo taxas de deteção, problemas encontrados e as correções efetuadas.

A arquitetura do sistema desenvolvido está devidamente explicada, destacando as suas camadas e componentes e como estes se integram para fornecer uma solução completa para a gestão de materiais em ambientes industriais.

Por fim, são discutidos os efeitos práticos dos resultados obtidos e são sugeridas áreas para melhorias e avanços futuros. Concluímos que a utilização de antenas e *tags* RFID permite que os materiais sejam localizados e movimentados com precisão, o que contribui significativamente para a melhoria dos processos industriais. Também evidenciamos a importância de pesquisas futuras para o desenvolvimento e aprimoramento contínuo destas tecnologias.

Palavras-chave: RFID, tags, antenas, OPC-UA, ambientes industriais

Esta página foi intencionalmente deixada em branco

Abstract

In this research, we examine the possibilities and advantages of using tracking and identification technologies, such as antennas and RFID tags, in industrial and manufacturing operations. The complexity of these environments demands efficient solutions for locating and identifying each object, aiming to increase productivity, reduce errors, and enhance operational efficiency. Thus, the objective of this work is to develop and analyze a system that allows for precise localization of RFID tags in industrial environments.

By addressing the different types of tags and antennas available on the market, we also provide a comprehensive review of the state-of-the-art of these technologies, including strategies and algorithms for tag detection and positioning calculation. The functioning of OPC-UA technology, which is important for the integration and communication among devices in the industrial environment, is also discussed to understand its role in the presented solution.

The hardware and technologies used in the development of the proposed system will also be examined, recognizing their advantages in this context. Additionally, the study includes a detailed analysis of scenarios and calculations performed to ensure the success in determining the tag positions using data captured by the antennas. Test results are presented and discussed, including detection rates, encountered issues, and the corresponding corrections.

The architecture of the developed system is properly explained, highlighting its layers and components and how they integrate to provide a comprehensive solution for material management in industrial environments.

Finally, the practical effects of the obtained results are discussed, and areas for improvements and future advancements are suggested. We conclude that the use of antennas and RFID tags enables precise localization and movement of materials, significantly contributing to the enhancement of industrial processes. We also emphasize the importance of future research for the continuous development and improvement of these technologies.

Keywords: RFID, tags, Antennas, OPC Unified Architecture, Industrial environments

Esta página foi intencionalmente deixada em branco

Lista de figuras

Esta página foi intencionalmente deixada em branco

Lista de tabelas

Esta página foi intencionalmente deixada em branco

Lista de siglas

1. RFID – *Radio Frequency Identification*
2. OPC-UA – O*penness, Productivity Collaboration - Unified Architecture*
3. ID – Identificação/ *Identity*
4. CI – Circuito Integrado
5. UC – Unidades de Controlo
6. HF – Alta Frequência
7. ISM - *Industrial, Scientific and Medical*
8. ISO - *International Organisation of Standardisation*
9. ETSI - *European Telecommunications Standards Institute*
10. FCC - *Federal Communications Commission*
11. RSS - *Received Signal Strenght*
12. SWA - Identificação por Ondas Acústicas de Superfície
13. TOA - *Time of Arrival*
14. LPM - *Local Position Measurement*
15. TDOA - *Time Difference Of Arrival*
16. MT - *Measurement Tags*
17. POA - *Phase Of Arrival*
18. AOA - *Angle Of Arrival*
19. kNN - *K-Nearest Neighbors*
20. SCADA - *Supervisory Control and Data Acquisition*
21. OLE - *Object Linking and Embedding*
22. OPC - OLE *for Process Control*
23. OPC-DA – OPC – *Data Access*
24. A&E - *Alarme & Events*
25. HDA - *Historical Data Access*
26. WS-Security - *Web Service Security*
27. API - *Application Programming Interface*
28. JSON - *JavaScript Object Notation*
29. JWT - JSON *Web Tokens*
30. SGBDR - Sistema de Gestão de Base de Dados Relacional
31. CI/DI - Integração Contínua e Implantação Contínua

32. IoT - *Internet of Things*

33. GPIO - *General Purpose Input/Output*

Esta página foi intencionalmente deixada em branco

Índice

Esta página foi intencionalmente deixada em branco

1. Introdução

Existe atualmente uma necessidade crescente de localizar e identificar cada componente do ambiente industrial nas fábricas devido ao movimento constante de materiais e objetos. A eficiência operacional, a redução de erros e o aumento da produtividade são objetivos cruciais para o sucesso das empresas neste setor.

Figura 1 - Exemplo de um par de antenas RFID

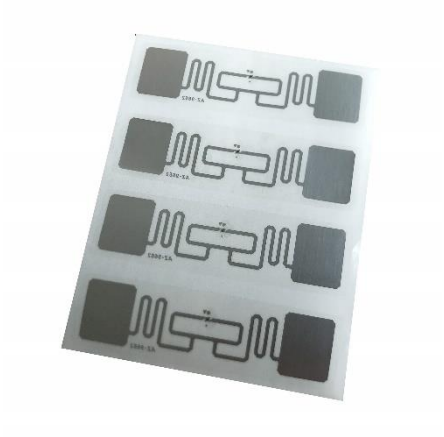


Figura 2 - Exemplo de uma tag RFID

Nesta situação, antenas e *tags* RFID (*Radio Frequency Identification*), apresentadas na Figura 1 e Figura 2 respetivamente, surgem como soluções tecnológicas importantes para mitigar o problema. As antenas, estrategicamente posicionadas capturam os sinais enviados por *tags* RFID, traduzindo-os em dados cruciais. Estas desempenham um papel crítico no processo. Elas são responsáveis por capturar os sinais de rádio enviados por *tags* RFID, estabelecendo uma comunicação eficiente. A sua capacidade de receber e transmitir informações em tempo real permite que a fábrica tenha controle preciso sobre a posição de cada objeto identificado. Ao implantar estrategicamente antenas em pontos importantes no

ambiente industrial, é possível construir uma rede de rastreamento eficiente. Isso resulta em benefícios como redução de erros, tempo de pesquisa simplificado para os materiais, produção aprimorada e perdas mínimas.

As *tags* RFID são fixadas ou incorporadas em cada material a ser rastreado. Ao inserir ou incorporar *tags* RFID em cada item, a fábrica adquire uma quantidade ilimitada de informações. Estas transmitem dados vitais, como identificação, localização, histórico de movimento e outras informações relevantes. Quando as etiquetas RFID são ativadas pelas antenas, os sinais de rádio são gravados e convertidos em dados importantes para o sistema de gestão da fábrica. Através de *tags* RFID, é possível rastrear objetos em tempo real, entender a sua trajetória dentro da fábrica, monitorizar os movimentos e prevenir perdas desnecessárias.

Com a posição precisa oferecida por antenas e etiquetas RFID, as empresas podem otimizar os seus processos de produção e logística. A minimização de erros e perdas evitáveis leva a uma melhor gestão de inventário e ao uso ideal dos recursos. Além disso, a rastreabilidade ajuda a identificar lacunas e possíveis melhorias nos fluxos de trabalho, melhorando a eficiência operacional.

A rapidez na localização dos materiais dentro das indústrias permite um aumento significativo na produção. O tempo economizado na busca de produtos resulta em maior utilização dos recursos humanos e um ritmo de trabalho mais eficiente.

Além disso, o uso de antenas e *tags* RFID aumenta a confiança nas operações industriais. Com informações precisas disponíveis em tempo real, relatórios e registos detalhados podem ser entregues aos clientes, parceiros e agências reguladoras, garantindo maior qualidade e segurança nos processos.

Assim este projeto tem como principal intenção utilizar e aproveitar as capacidades da tecnologia dentro do contexto das empresas e fornecer um sistema desenvolvido de forma a disponibilizar e facilitar a implementação desta tecnologia. O projeto terá assim como objetivo o desenvolvimento de um sistema que forneça a posição absoluta das *tags* dentro de um armazém ou zona de fabricação para os utilizadores através de um *website* ou através de um servidor OPC-UA.

No segundo capítulo será apresentado o estado da arte da tecnologia utilizada, onde serão evidenciados e comparados os diferentes tipos de *tags* e antenas existentes no

mercado assim como estratégias e algoritmos para deteção e cálculo da posição das *tags* detetadas e ainda o funcionamento e crescimento da tecnologia OPC-UA.

O terceiro capítulo explorará as tecnologias e hardware utilizados neste projeto que permitiu alcançar a solução final. Neste será ainda explicado as vantagens da utilização desta tecnologia e o porquê de ter sido optada para este projeto.

No quarto capítulo será abordado e detalhado os cenários e cálculos efetuados para obtermos sucesso no cálculo do posicionamento das *tags* consoante as posições das antenas com base no algoritmo e tipo de deteção escolhidos.

O quinto capítulo tem como intenção dar a conhecer o desenvolvimento e a arquitetura da solução desenvolvida.

O sexto abordará os testes desenvolvidos sobre o sistema e terá como principal objetivo apresentar os resultados dos desenvolvimentos, tanto taxas de deteção bem-sucedidas como possíveis correções que tem de ser efetuadas (ou que foram efetuadas).

Antes da conclusão será ainda apresentado o processo de instalação da solução. Este sétimo capítulo tem como principal objetivo acompanhar e ilustrar passo a passo os requisitos e passos necessários para utilizar de forma correta o projeto desenvolvido.

O último e oitavo capítulo está reservado para a conclusão onde será feita uma análise dos resultados, da solução e do processo de desenvolvimento. Será também indicado possíveis correções e melhoramentos futuros.

2. Estado da Arte

Este capítulo tem como funcionalidade enaltecer e apresentar ao leitor os desenvolvimentos e os conhecimentos necessários para o entendimento do que será discutido ao longo do relatório. Assim serão apresentados os principais estudos efetuados com base na tecnologia de RFID e os desenvolvimentos e funcionamento do protocolo de comunicação OPC-UA, atualmente conhecido como *Openness, Productivity Collaboration - Unified Architecture.*

2.1. Radio Frequency Identification System

2.1.1. História

O RFID é uma tecnologia que tem como base a utilização de ondas de rádio para identificar, controlar ou armazenar meta dados de objetos, usualmente utilizando um sistema que incorpora *tags*/transmissores e *readers*/leitores como referido em [1]. Esta tecnológica foi possibilitada graças à invenção de Alexander Watson-Watt em 1935, o radar, que tem a capacidade de localizar objetos físicos através de ondas de rádio e dos seus ecos. Esta descoberta, embora impactante, apresentava falhas, nomeadamente a incapacidade de, durante a 2ª Guerra Mundial, identificar e distinguir entre aliado ou inimigo. De forma a corrigir esta desvantagem foi apresentada em 1948 por H. Stockman a descrição para a tecnologia RFID quando tanto a Alemanha como a Inglaterra apresentaram soluções com base na reflecção do sinal [2].

Nos anos seguintes, como mencionado em [3] iria-se verificar uma diferença em abordagem entre os Estados Unidos e a Europa após a primeira aplicação comercial e a eventual comercialização da tecnologia, em que uma investiu e procurou o uso desta tecnologia para acesso próprio e transporte enquanto o outro direcionou-se para as aplicações empresariais, industriais e rastreio animal, respetivamente.

Embora a sua potência já fosse reconhecida foi apenas após a queda do custo e o aumento da capacidade dos equipamentos que os grandes negócios principalmente comerciais investiram, como em 2005 pela grande multinacional Wal-Mart [4].

2.1.2. Sistema RFID

De forma a obtermos um melhor entendimento do funcionamento da tecnologia RFID a Figura 3 representa a topologia usual para os sistemas RFID, que apresenta uma comunicação linear e ordenada entre os intervenientes: a aplicação, ou sistema, solicita ao leitor que explore o ambiente; este corresponde com um envio de sinal e aguarda a receção do sinal das *tags*/transmissores; após as *tags* enviarem a informação desejada o leitor retorna essa informação para a aplicação, concluindo assim o ciclo.

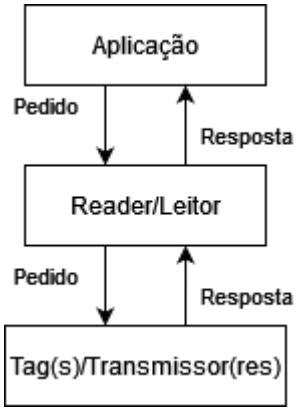


Figura 3 - Ilustração da topologia de um Sistema RFID

O ciclo de vida mencionado é possibilitado devido ao processo de solicitação (*query*) da identificação (ID) das *tags* através de ondas continuas de radio frequência, denominado *downlink* (comunicação leitor→*tag*) e o processo de *uplink* que consiste na alteração do coeficiente de reflexão do sinal enviado na antena e retorno do mesmo (comunicação *tag*→leitor) como mencionado em [5] e representado na figura abaixo.

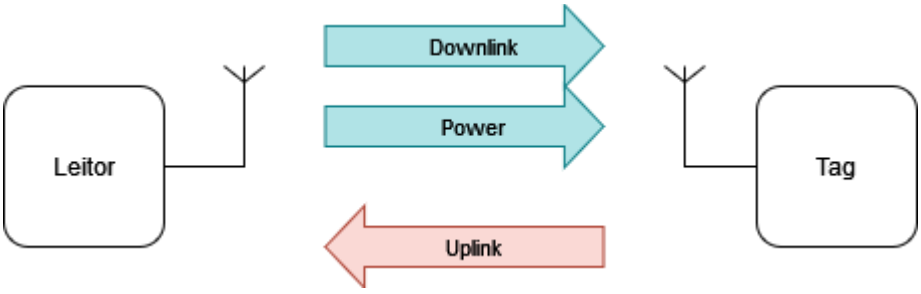


Figura 4 - Ilustração de Downlink e Uplink

Podemos ainda encontrar outro tipo de sistema RFID, como presente na Figura 5, onde existe uma diferença na comunicação entre o leitor e as *tags*. Isto deve-se ao tipo de *tag* que estão incluídas no sistema, sendo que este representa a transmissão de informação entre o leitor e *tags* ativas, ao contrário do sistema apresentado anteriormente que ilustrava a comunicação com *tags* passivas. Assim a forma de passagem de comunicação está dependente do tipo de intervenientes, que serão abordados nas próximas secções de forma a estudar cada componente e o seu funcionamento na identificação e localização de objetos.

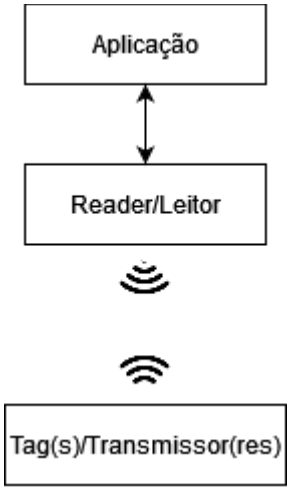


Figura 5 - Representação de um sistema com tag RFID Activa

2.1.3. Tags

Nesta secção serão estudados os tipos e o funcionamento de *tags* RFID presentes no mercado atual e os cenários em que melhor se adequam. Também serão identificadas e distinguidas as características que cada uma apresenta de forma a identificar qual a que apresenta melhores condições para ser aplicado no problema apresentado.

Uma *tag*, para melhor contextualização, serve de adereço de identificação do objeto pretendido sendo usualmente coladas, encostadas ou embutidas nos mesmos ou nas suas embalagens. Esta capacidade de identificação é possibilitada devido aos componentes presentes dentro do mesmo, assim, e independentemente do tipo, podemos encontrar as seguintes peças:

- Antena
- *Microchip*/Circuito integrado (CI)

É possível visualizar um exemplo dos componentes e da estrutura de uma *tag* na Figura 6, onde podemos encontrar um CI, onde está alojado o ID, ou *Identity*, que neste contexto é representado por um conjunto de caracteres que identifica a *tag*, e um circuito impresso na *board*, ou um substrato e a antena.

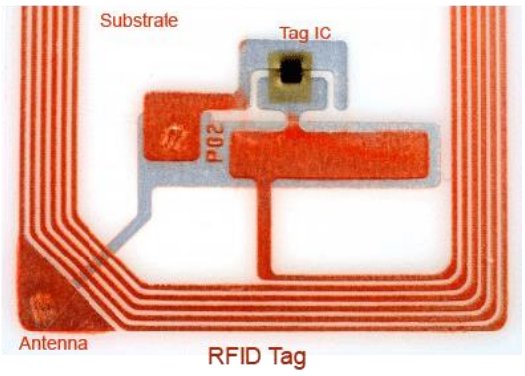


Figura 6 - Exemplo de uma RFID Tag

Dentro do conceito é possível encontrar diferentes tipos e formatos de *tags* RFID que oferecem capacidades e custos distintos, mas que são maioritariamente categorizadas segundo a sua cumplicidade dentro do sistema RFID, nomeadamente a dependência do

sinal do leitor, os alcances em que conseguem comunicar e os dados que conseguem disponibilizar. Assim as categorias são:

- *Tags* Passivas
- *Tags* Ativas ou Semi-Ativas

2.1.3.1.1. *Tags* Passivas

As *tags* passivas podem ser principalmente identificadas pelo facto de terem um papel passivo quanto ao sistema, em que esta apenas tem energia e atividade quando existe a *query* efetuado pelo leitor. Existe assim uma reação ao processo de *downlink* que, para este tipo de *tags*, requer o envio de energia suficiente para o carregamento e ativação do *microchip* da *tag* devido a esta não ter uma bateria própria e assim estar dependente da força do sinal da antena do leitor.

Figura 7 - Exemplo de uma RFID tag passiva

Na Figura 7 está ilustrado um exemplo de uma *tag* passiva que podem muitas vezes ser reconhecidas devido ao tamanho e formato reduzido devido à ausência de uma bateria.

Além desta característica que já por si separa e distingue este tipo de transmissor dos restantes, podemos ainda encontrar diferença da mesma para os restantes no alcance e na escrita e armazenamento de dados que este oferece. Usualmente estas apenas são detetadas pelo leitor dentro de um raio de 3 metros, numa quantidade dentro da casa das centenas, e apresenta uma capacidade ínfima de armazenamento leitura/escrita a rondar os valores dos 128 bytes que é a capacidade inferente no *microchip* que este possui [6].

2.1.3.1.2. *Tags* Ativas ou Semi-Ativas

Uma *tag* considerada ativa tem, ao contrário das *tags* passivas, e como mencionado anteriormente, a capacidade de interagir e comunicar com o leitor com capacidade própria sem a necessidade de ser requisitado o seu *output*. Este tipo apresenta essa capacidade devido à presença de uma bateria ou fonte de alimentação própria que permite à mesma manter-se ativa. Devido à não dependência da energia emitida no sinal da antena do leitor, este tipo de *tags* apresenta um alcance de quase 100 metros, sendo capaz de identificar até milhares de *tags* dentro deste raio, e uma capacidade de armazenamento de leitor/escrita mil vezes superior às *tag* passiva, na ordem dos 128KB [6].

Figura 8 - Exemplo de uma tag RFID ativa presente no mercado

Devido às características mencionadas as *tags* ativas são normalmente fáceis de identificar pois apresentam um volume e tamanho bastante significativo em comparação com as *tags* passivas como é possível comparar entre a Figura 7 e a Figura 8.

Tal como foi mencionado anteriormente podemos então distinguir dois tipos de *tags* RFID que podem ser utilizadas num sistema RFID. De modo a obtermos uma melhor comparação entre as duas podemos observar a Tabela 1.

Tabela 1 - Active VS Passive RFID Tags [6]

	***Tag* RFID Ativa**	***Tag* RFID Passiva**
Distância de comunicação	≥100 m	≤ 3 m usualmente (≤ 25 m caso a antena utilizada seja de elevado ganho)
Capacidade de deteção	Milhares de *tags* dentro de um espaço de 7 hectares por um leitor. 20 *tags* a uma velocidade até ≈160 km/h	Centenas de *tags* num raio de 3 metros 20 *tags* a uma velocidade até ≈ 5 km/h
Capacidade dos sensores	Leitura e transferência de dados continua	Leitura e transferência de dados apenas quando *tag* é ativada pelo leitor
Capacidade de armazenamento de dados	Grande (128KB)	Pequena (128 B)

Embora todos os indicadores mencionados anteriormente nos façam pensar que devido à capacidade de alcance e armazenamento as *tags* ativas apresentem sempre vantagem em relação às *tags* passiva existe ainda um termo de comparação que ainda não foi indicado: o custo. Este, apenas a nível de material e não implementação ou solução, apresenta uma vantagem gigante para as *tags* passivas sendo que cada uma não apresenta valores superiores a 1 euro no mercado internacional ao contrário das *tags* ativas em que uma pode custar valores a rondar entre os 25 a 100 dólares o que equivale a uma faixa entre os 23 e os 92 euros atualmente aproximadamente [7].

2.1.4. *Reader*

O leitor/*reader* é, num sistema RFID, o *middleman*, um intermediário entre a aplicação e a informação que está gravada nas *tags*. Assim a sua principal funcionalidade no sistema é solicitar e detetar a comunicação com as *tags* que se encontrem dentro do seu raio de observação e transmitir a informação captada para a aplicação.

Tal como referido por Xiaolin Jia et. al. em [1], este interveniente tem a capacidade de desempenhar as suas funções devido a dois blocos essenciais: Unidade de Controlo (UC) e a interface de Alta Frequência (HF) como é possível visualizar na Figura 9.

- UC - Unidade de Controlo - tem a função de codificar o sinal enviado para a interface HF e descodificar o sinal recebido da mesma, tal como comunicar com a aplicação e ainda garantir autenticação entre o *reader* e a *tags*.

- Interface HF – tem como responsabilidade: gerar energia e potência de alta frequência suficiente para a ativação das *tags*; modular o sinal de transmissão e ainda receber o sinal proveniente das *tags* de forma a enviar o mesmo para o UC.

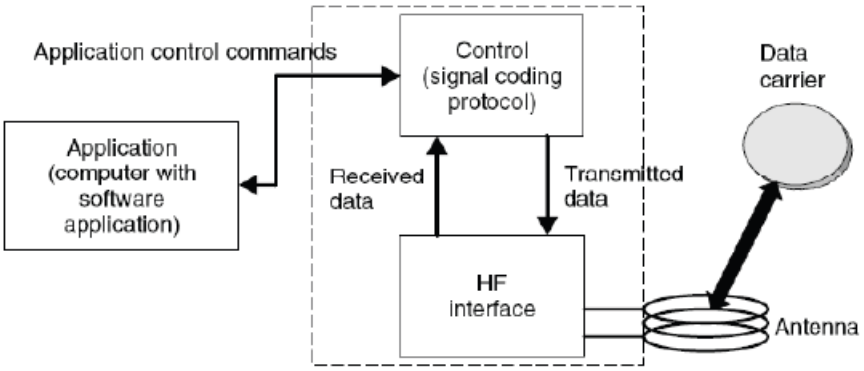


Figura 9 - Ilustração do funcionamento de um RFID reader [7]

2.1.5. Frequências

RFID, tal como indicado em [8], conta com ondas de rádio, que são um componente do espectro eletromagnético e possuem frequências entre 300 kHz e 3 GHz. É comparável ao Wi-Fi e Bluetooth, duas outras tecnologias sem fio. As três tecnologias têm funcionalidades diferentes porque foram criadas para propósitos bastante distintos, mas há pontos em comum entre elas e alguns híbridos começaram a surgir. Sendo que alguns sistemas RFID até incorporam Wi-Fi e Bluetooth nas suas soluções sendo que não é necessário existir concorrência ou exclusão entre qual tecnologia utilizar.

O RFID é executado dentro de uma área do espectro não licenciado conhecido como *Industrial, Scientific and Medical*, ou ISM, embora as frequências que compõem o ISM possam mudar dependendo das leis locais e da região. Assim podemos visualizar as várias bandas de ondas de rádio e as frequências mais predominantes usadas na tabela abaixo.

Tabela 2 - Frequências e bandas encontradas em sistemas RFID [8]

Band	**LF Low frequency**	**HF High frequency**	**UHF Ultra high frequency**	**Microwave**
Frequency	30–300kHz	3–30MHz	300 MHz–3GHz	2–30 GHz
Typical RFID Frequencies	125–134 kHz	13.56 MHz	433 MHz or 865 – 956MHz 2.45 GHz	2.45 GHz
Approximate read range	less than 0.5 metre	Up to 1.5 metres	433 MHz = up to 100 metres 865-956 MHz = 0.5 to 5 metres	Up to 10m
Typical data transfer rate	less than 1 kilobit per second (kbit/s)	Approximately 25 kbit/s	433–956 = 30 kbit/s 2.45 =100 kbit/s	Up to 100 kbit/s
Characteristics	Short-range, low data transfer rate, penetrates water but not metal.	Higher ranges, reasonable data rate (similar to GSM phone), penetrates water but not metal.	Long ranges, high data transfer rate, concurrent read of <100 items, cannot penetrate water or metals	Long range, high data transfer rate, cannot penetrate water or metal
Typical use	Animal ID Car immobiliser	Smart Labels Contact-less travel cards Access & Security	Specialist animal tracking Logistics	Moving vehicle toll

Com base nos dados atuais e apresentados na tabela podemos considerar a frequência UHF como a mais plausível para utilizarmos no sistema a implementar devido à cobertura oferecida de 0,5 a 5 metros e o número de *tags* que conseguem ser lidas o que diminui a necessidade de uma quantidade exagerada de leitores.

2.1.6. *Standards*

É bastante complicado, envolve várias organizações e ainda está em desenvolvimento quantos padrões serão utilizados pelo RFID e pelas indústrias que estão relacionadas a este. Foram criadas normas para cobrir os quatro usos e aplicações essenciais para RFID:

Normas:

- *International Organisation of Standardisation* (ISO)
- *EPCglobal Inc*
- *European Telecommunications Standards Institute* (ETSI)
- *Federal Communications Commission* (FCC)

Usos e Aplicações:

- *Air-interface standard* (para comunicação básica de dados *tag-to-reader*)
- conteúdo e codificação de dados (sistemas de numeração);
- conformidade (teste do sistema RFID);
- interoperabilidade entre sistemas e aplicações RFID.

2.1.7. Algoritmos de Localização

Devido ao problema apresentado no contexto deste trabalho estar relacionado com a captação e identificação de objetos com o recurso à tecnologia RFID dentro de instalações, e não numa local aberto, esta secção irá apresentar os algoritmos ou as soluções encontradas para poder ser calculada a distância de localização do sinal em *indoor systems* (sistemas interiores).

Dentro de um sistema como o referido, podemos encontrar diversos problemas como:

- *Multipath* ("Multicaminhos") – que no ramo da comunicação via radio frequência é um fenómeno relacionado com a propagação do sinal, em que este pode chegar à antena por diversos caminhos;

- *Line-of-sight* – ou linha de visão – em que uma antena apresenta um certo campo de visão onde consegue captar os sinais;

- Absorção, Difração e Reflexão – referem-se à absorção do sinal por outros equipamentos que impedem o sinal de chegar ao destino, à difração que o mesmo pode sofrer quando tem de contornar ou trespassar objetos no caminho e à reflexão que o sinal pode sofrer quando existem objetos que refletem o sinal e pode causar o *multipath* mencionado anteriormente.

No objetivo de tentar obter as distâncias e localização dos objetos identificados com *tags* RFID, o artigo de jornal [9], apresenta 3 tipos de abordagens diferentes para alcançar esse objetivo. Assim foram identificadas as abordagens por estimativa de distância, análise de cena e proximidade.

2.1.7.1.1. Estimativas por distância

Estes tipos de algoritmos recorrem à triangulação ou à trilaterização através de pontos de referência como podemos ver na figura abaixo.

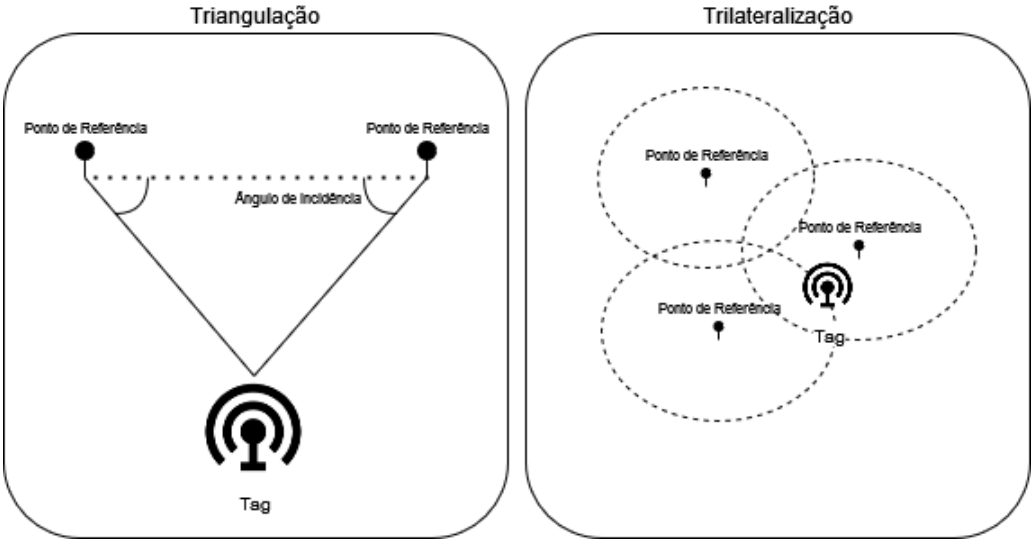


Figura 10 - Triangulação vs Trilaterização

Dentro deste tipo identificamos algoritmos como:

- SpotON - utiliza as medidas de *Received Signal Strenght* (RSS) obtidas de *tags* RFID ativas de longa distância com o auxílio de vários leitores de forma a calcular a distância recorrendo a métodos de lateralização;

- SAW ID-*tags* - utiliza *tags* passivas diferenciadas com Identificação por Ondas Acústicas de Superfície (SWA) com recurso a medidas de *Time of Arrival* (TOA) calculados à chegada do sinal nos leitores para localizar as mesmas, fazendo uso também do conceito de trilateralização;

- *Local Position Measurement* – LPM – pode ser usado em *tags* ativas utilizando técnicas de *Time Difference Of Arrival* (TDOA) com *tags* de referências ou *measurement tags* (MT) posicionadas estrategicamente no espaço para identificar a localização das *tags* que a rodeiam comparando os tempos de resposta entre três leitores;

- *Phase Of Arrival* – POA – localiza *tags* passivas com auxílio de dois leitores colocados em posições especificas para captar a direção de uma *tag* em movimento.

- Angle Of Arrival – AOA – localização das *tags* utilizando o ângulo de captura entre dois pontos de referência, em que o ponto de interseção entre as linhas formadas pelo ângulo das antenas será a localização estimada da *tag*.

2.1.7.1.2. Scene Analysis

Os algoritmos com base neste tipo de abordagens necessitam de dois passos fundamentais para o seu funcionamento: agregação de dados do ambiente e estimação da localização da *tag* com base nesses mesmos dados, sendo as duas técnicas mais conhecidas o *K-Nearest Neighbors* (kNN), ou mapa radial, e o método probabilístico.

- *Landmarc* – com o auxílio de *tags* de referência em posições conhecidas esta técnica de localização utiliza o princípio de kNN e diferentes níveis de sinal nos leitores, 8 para ser exato, para identificar a posição da *tags* a identificar.

- VIRE - *Active RFID-based localization using virtual reference elimination* – apresenta o conceito de mapas de proximidade para cada leitor em que a área de captação é também dividida em regiões em que os centros tem uma *tag* de referência, sendo que aproveita a técnica mencionada anteriormente.

- *Simplex* – utiliza leitores com diferentes níveis de leitura de forma incremental na sua potência até obter a *tag* que deseja identificar.

- *Scout* – este algoritmo tem como base o método probabilístico em que com o auxílio de *tags* de referência e uma quantidade significativa de leitores é possível localizar *tags* ativas através da aplicação de um modelo treinado.

- *Kalman filtering* – o algoritmo apresenta duas fases: obtenção da localização das *tags* através da força do sinal recebido (RSS) de 2 leitores entre a *tag* de referência e a *tag* a localizar e a obtenção de um mapa probabilístico do erro de deteção do leitor.

2.1.7.1.3. *Constraint-based approach*

Esta abordagem requer o uso de um nível elevado de leitores pois tem como base localizar uma *tag* pela aproximação e força de sinal transmitidos desta ao leitor no processo de *uplink*. Em termos bastante simples, devido ao elevado número de leitores, se uma *tag* for localizada por apenas um leitor esta será considerada como localizada dentro da região designada ao leitor e caso seja encontrada por mais que um será a força do sinal a desempatar o conflito.

- 3-D *Constraints* – utiliza um mapa de restrições através de pontos de referências conhecidos do raio de antena para apropriar a localização da *tag* caso este seja encontrado dento do alcance da mesma.

A análise destes métodos, obtidos do trabalho [9], serviu assim para observar os algoritmos existentes e verificar a forma como podemos obter a localização das *tags* RFID com alguma precisão para posteriormente devolvermos esta informação numa plataforma de acesso para clientes. Com base na acessibilidade e simplicidade de cada um podemos reter que a metodologia por estimativa de distância seria a forma mais prática e devido ao custo estimado para este projeto o algoritmo de AOA seria o mais adequado para ser apresentado pois focasse nas *tags* passivas, as *tags* que estão em melhor condição para serem utilizadas.

2.2. *Openness, Productivity Collaboration - Unified Architecture*

Nesta secção será abordado o desenvolvimento do protocolo de comunicação OPC-UA, ou *Openness, Productivity Collaboration - Unified Architecture* baseada na arquitetura OPC, conhecida inicialmente como OLE for *Process Control*. Será assim abordada a necessidade do mesmo e o que se sucedeu para o seu aparecimento.

2.2.1. História

Segundo [10], ao longo dos anos vários *bus systems*, vias de comunicação entre o processador do computador e a memória principal do mesmo composto por cabos e conectores, foram desenvolvidos, mas sem grande unificação e real capacidade de se estabelecer como *standard* na indústria. Sendo que para cada dispositivo era necessário a disponibilização de um *driver* por parte da empresa que forneceu o equipamento.

Foi apenas em 1975 que um agregado de empresas decidiu se juntar de forma a encontrar uma solução que visasse combater a situação. Visto que até ao momento os sistemas presentes no mercado apresentavam diversas adversidades quando a especificação do protocolo era alterada. Isto resultava em falhas que ocupavam imenso tempo de análise e restruturação para serem corrigidas. A solução apresentava assim uma arquitetura de software entre cliente e servidor para a conversão de dados reais de processamento para sistemas denominados *Supervisory Control and Data Acquisition* (SCADA), denominada OPC-DA, ou *Object Linking and Embedding* (OLE) for *Process Control – Data Access*, em 1976, que, tal como representado no nome, baseou-se na tecnologia OLE da Microsoft que atendia o desenvolvimento de estruturas orientadas a objetos.

Os sistemas SCADA, para melhor entendimento, tem como função o agrupamento e apresentação de dados de infraestruturas através de hardware, controladores, software, rede e comunicações, sendo que estes são bastante usuais nas indústrias de óleos e gás, água, gestão de resíduos e geração e distribuição de energia, como referido em [11].

Desde o resto dos anos 90 até aos dias de hoje a fundação OPC continua a lançar especificações para os *standard* de comunicação, tendo até alterado a designação da sigla

para *Openness, Productivity Collaboration*, no entanto a contribuição mais significativa foi apresentada em 2006, denominada OPC-UA, ou OPC – *Unified Architecture* que tinha como principal objetivo corrigir os erros apresentados no denominado atualmente *Classic* OPC, tendo sido obtido o reconhecimento de tecnologia padrão apenas em 2010. O *Classic* OPC é o agregado de especificações como o DA, *Alarme & Events* (A&E), *Historical Data Access* (HDA) e outros que serão enumerados e explicados na próxima secção para lucidar o porque do surgimento deste novo *standard* de comunicação.

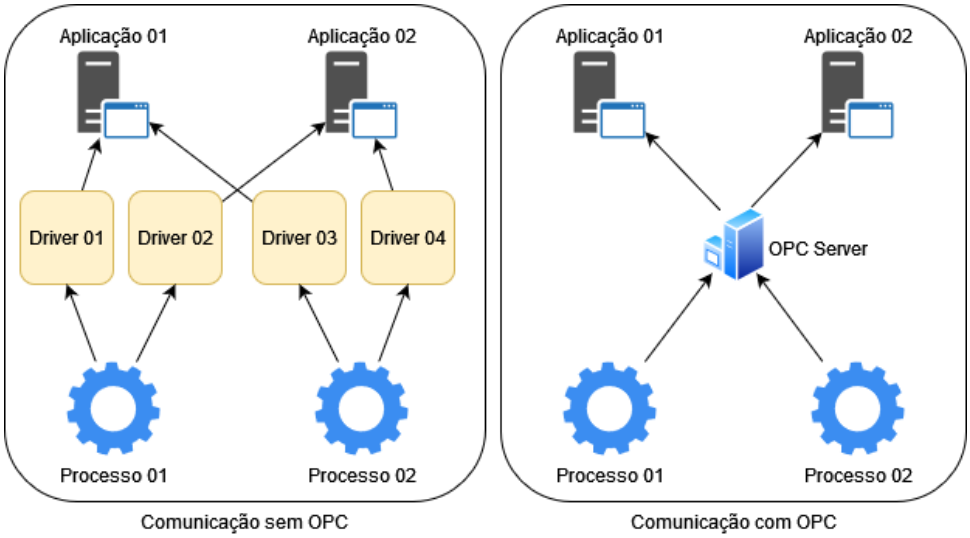


Figura 11 - Comunicação sem e com recurso à arquitetura OPC

2.2.2. *Classic OPC*

Tal como foi mencionado na secção anterior, existe um período pós OPC-UA e anterior ao mesmo, sendo este último referido como *Classic* OPC que denomina o grupo de especificações existentes antes do OPC-UA e de existir então a unificação da arquitetura.

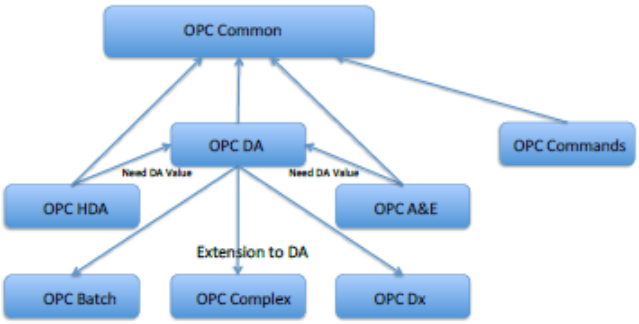


Figura 12 - Relação entre as especificações [10]

Como demonstrado na Figura 12, o *Classic* OPC tem no seu conjunto as seguintes especificações de arquiteturas:

- *Data Access* (DA) *Specification* – apresenta a informação em tempo real ao servidor num formato específico composta por: valor do processo; *timestamp* indicado pelo servidor e confiança que pode ser "bom", "mau" ou "desconhecido".
- *Alarme & Events* (A&E) *Specification* – baseia-se numa interface que regista e transmite informação com a subscrição a alarmes e eventos; necessita da presença de outro servidor DA para a passagem de informação do valor do processo.

- *Historical Data Access* (HDA) *Specification* – utilizado principalmente para avaliar e otimizar processos, esta especificação recorre do formato apresentado em DA, sendo considerada uma extensão do mesmo, para aceder não a dados que são captados em tempo real, mas sim a dados históricos. Apresenta e possibilita métodos de *query* para obtenção desses dados.

- *Command Specification* – interface que permite executar comandos específicos para controlo, configuração e reiniciar dispositivos.

- XML *Data Access Specification* – especificação com base no OPC-DA que tem como método de comunicação *web-services* e ficheiros XML

- *Data Exchange Specification* – interface de comunicação entre servidores com base na especificação OPC-DA, sendo também este considerado uma extensão.

- *Batch Specification* – especificação que estende o OPC-DA que visa casos especiais de processos *batch*.

2.2.3. OPC-UA

Após a leitura da secção anterior estamos agora na capacidade de avaliar a necessidade e o que proporcionou o aparecimento deste conceito de OPC-UA ou OPC – *Unified Architecture* que tal como indica o nome tem como função corrigir e unificar as especificações mencionadas anteriormente numa só arquitetura.

Figura 13 - Logotipo OPC-UA

O OPC-UA vem corrigir o erro proveniente do espaço de endereços que não se interligavam devido aos diferentes espaços usados por especificações, isto causava com que um processo que comunicasse com servidores de especificações diferentes não pudessem funcionar corretamente. Por exemplo os dados provenientes da comunicação entre servidor e cliente OPC A&E não fossem correspondidos num servidor DA.

Assim foi unificado o espaço de endereços de forma a ser possível ao cliente aceder ao objeto num único serviço e sem incoerência de dados.

Figura 14 - Modelo do Objeto presente no espaço de endereço de um servidor OPC-UA

Tal como podemos observar na Figura 14, que representa o modelo objeto que o utilizador pode aceder no espaço de endereçamento de um servidor OPC-UA, encontramos três distintas divisões no objeto, sendo estes devido ao modelo incluir blocos que outrora seria ocupado com dados de diferentes especificações:

- Variáveis → OPC-DA / OPC-HDA
- Eventos → OPC-A&E
- Métodos → OPC-*Command*

Estes objetos e as informações relacionadas são disponibilizadas para o cliente pelo espaço de endereçamento em que estes são representados como conjuntos de *nodes*. Um *node* é composto por atributos e interligado por referências com outros *nodes* presentes no espaço sendo que uma referência a um *node* no servidor tem de ser obtida por uma instância de uma classe "*Node*". Existem 8 tipos definidos nas especificações do OPC-UA:

- *Object* – representa entidades reais
- *Variable* – como já conhecido na programação orientada a objetos, este tipo de *nodes* serve para armazenar e aceder a dados
- *Method* – ações e operações disponibilizadas pelo *Object*

- *View* - tal como nas bases de dados este tipo de *node* serve para representar um conjunto predefinido para o espaço de endereçamento pretendido de forma a reduzir a pesquisa.
- *ObjectType* - especifica a estrutura que todos os *nodes Object* tem de apresentar
- *VariableType* – tipo de dado a armazenar
- *ReferenceType* – composto por uma indicação para o próximo *node* e uma indicação para o *node* de origem este tipo cria uma relação entre os *nodes*.
- *DataType* – tipos de dados básicos fornecidos pela linguagem de programação e ainda possíveis tipos de dados adicionados

Através da identificação do tipo de nodes podemos assim perceber a hierarquia de objetos presente no espaço de endereçamento apresentado na Figura 15.

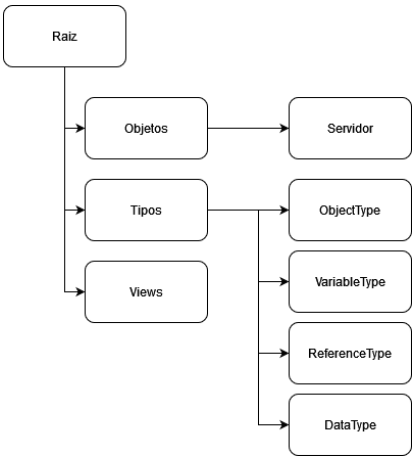


Figura 15 - Hierarquia de Objetos no espaço de endereçamento OPC-UA

2.2.4. Segurança

O procedimento padrão é configurar uma Avaliação de Segurança, na qual especialistas em segurança, especialistas em sistemas, especialistas em domínio e utilizadores do sistema de destino colaboram para definir metas de segurança, identificar ameaças e riscos e introduzir contramedidas após analisar o sistema de destino e o ambiente. O resultado de uma avaliação de segurança é normalmente um documento que lista todos os problemas descritos acima.

No contexto de sistemas de automação, quando o OPC-UA é utilizado, este especifica o resultado de uma avaliação de segurança como o tipo de aplicação industrial, mapeados os seis objetivos comuns de segurança – autenticação, autorização, confidencialidade, integridade, audibilidade e disponibilidade. Além disso, são discutidos os potenciais perigos que podem existir em contextos OPC-UA. Um perigo também pode comprometer mais do que um objetivo, e muitas ameaças podem comprometer o mesmo objetivo de segurança.

O próximo passo seria aplicar as contramedidas. As aplicações industriais são sistemas complexos que são controlados por várias pessoas e geridos por diferentes processos organizacionais, portanto, diferentes contramedidas e ações podem e devem ser usadas em diferentes locais e componentes de um sistema.

De forma a poder alcançar a segurança entre um OPC-UA cliente e um OPC-UA servidor existe uma comunicação entre os dois através de um canal seguro disponibilizado pela camada de comunicação existente em ambos que permite a camada aplicacional enviar dados sensíveis como autenticação. Assim a autenticação do cliente ou utilizador é providenciada através de uma password, certificado ou *token* de WS-*Security* (*Web Service Security*). Esta verificação permite definir roles e garantir acesso apenas aos serviços e funcionalidades descritas ou pensadas para o mesmo e após a mesma é garantida a confidencialidade através da encriptação dos pedidos e respostas efetuadas entre os dois pontos. É ainda complementada com assinatura da mensagem para garantir integridade através do canal seguro mencionado e representado na Figura 16.

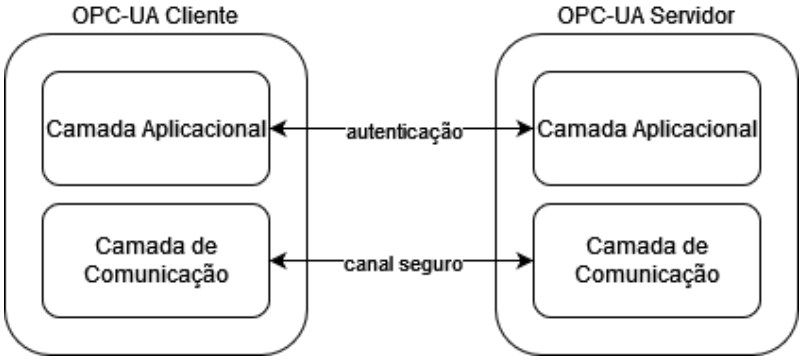


Figura 16 - Esquema representativa da comunicação entre cliente e servidor OPC-UA

A comunicação entre o cliente e o servidor é disponibilizada, como demonstrado na Figura 17, por diversos serviços *web* estandardizados suportados pelo servidor OPC-UA, entre estes:

- *Secure Channel Service* – utilizado para criar os canais de comunicação referidos anteriormente e assegurar a mesma entre servidor e cliente
- *Session Service* – cria e gere a comunicação
- *NodeManagement, View and Attribute Service* – gere o espaço de endereçamento permitindo ao cliente manusear e obter dados dos *nodes*
- *Subscription Service* – subscreve à atualização de dados entre cliente e servidor
- *Method Service* – permite ao cliente aceder aos métodos disponibilizados pelos objetos criados

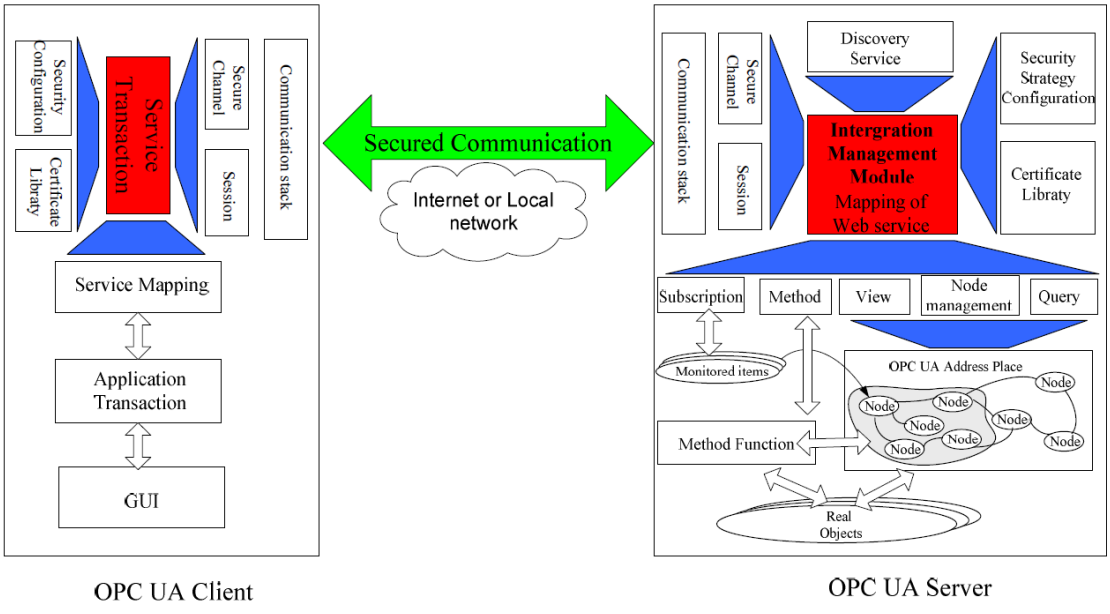


Figura 17 - Ilustração da arquitetura de comunicação OPC-UA [12]

3. Tecnologias Utilizadas

Neste capítulo serão enumeradas e descritas as ferramentas utilizadas de forma a poder atacar o problema e desenvolver uma solução para o mesmo. Assim durante as próximas secções serão apresentadas as tecnologias tanto de *hardware* como de *software* que aparentavam serem as ideias dentro do contexto. Como foi já mencionado o capítulo será então dividido em duas secções: "*hardware*" e "*software*" em que em ambas serão listadas e explicadas as vantagens de cada tecnologia bem como as relações que possam existir entre as mesmas.

3.1. *Software*

Nesta secção serão enumerados os *softwares* utilizados no projeto de forma a conseguirmos criar uma aplicação *web* responsiva e contextualizada no problema disponibilizando as funcionalidades para o consumidor final. Além de enumeradas são ainda descritas as suas vantagens e o porque da necessidade de utilizar cada um dos *softwares*.

3.1.1. .NET *Framework*

A primeira tecnologia a apresentar para podermos criar a solução *web* para o problema é a *framework* .NET que é uma plataforma de desenvolvimento de *software* criada pela Microsoft. Ela fornece uma estrutura para o desenvolvimento de aplicações, incluindo ferramentas e bibliotecas para criar, testar e implantar as mesmas.

Uma das vantagens do uso da .NET *Framework* é o desenvolvimento de API (*Application Programming Interface*) com o uso do .NET *Entity Core Framework*, um conjunto de ferramentas e bibliotecas que permitem trabalhar diretamente com bancos de dados de forma mais eficiente e segura, sendo uma das principais vantagens do .NET *Entity Core Framework* a sua capacidade de mapear objetos de código para tabelas de banco de dados. Esta ferramenta permitiu trabalhar com os dados sem haver preocupação com os detalhes de como os dados estão a ser armazenados graças ao elevado nível de abstração fornecido ao incluir métodos de operações gravação e remoção de dados de

bases de dados relacionais e ainda consultas complexas devido ao suporte para junções de tabelas, agregações, ordenação e filtragem de dados.

Em resumo, o uso do .NET *Entity Core Framework* para desenvolvimento de API traz uma série de vantagens, incluindo maior produtividade, menor custo, segurança e confiabilidade. Assim foi uma solução obvia e prática a implementar para o desenvolvimento de *backend* necessário para disponibilizar diversas funcionalidades na aplicação *web*.

3.1.2. *Swagger UI*

Esta ferramenta é extremamente útil para o desenvolvimento e documentação de APIs. O *Swagger UI* oferece uma interface interativa para explorar, testar e documentar eficientemente APIs. O que simplifica o processo de desenvolvimento e facilita o entendimento da API para os utilizadores e programadores. Através da mesma é possível criar uma documentação de uma API abrangente e interativa que inclua informações sobre *endpoints* disponíveis, parâmetros esperados, respostas devolvidas e exemplo.

Além disso, o *Swagger UI* oferece recursos de autenticação e autorização integrados. Isso significa que é possível simular uma autenticação e testar diferentes cenários de autorização diretamente na interface do *Swagger UI*. O que acaba por tornar-se especialmente útil durante o desenvolvimento e teste de API que exigem autenticação, como APIs com *tokens* de acesso ou autenticações baseadas JSON (*JavaScript Object Notation*), uma formatação de texto para troca de dados, como JWT *(*JSON *Web Tokens).*

Outra vantagem do *Swagger UI* é a sua capacidade de gerar código cliente automaticamente. Com base na definição da API no formato *Swagger* ou *OpenAPI,* a interface do usuário do *Swagger* pode gerar código cliente em várias linguagens de programação, como Java, Python, JavaScript, C#, entre outras. Assim o *Swagger* foi utilizado neste projeto como ferramenta de testes e documentação da API desenvolvida.

3.1.3. *Microsoft SQL Server*

De forma a podermos armazenar os dados que recebemos e consultamos na API foi necessário utilizar uma base de dados relacional. Dentro deste tópico existem diversas plataformas e recursos que poderia utilizar para alcançar o objetivo, entre elas MySQL, PostgresSQL, Oracle, DB2, SQLite entre outras gamas de fornecedores de bases de dados. No entanto dentro do contexto e tecnologia utilizada para desenvolver a API foi uma escolha fácil utilizar o SQL Server como o Sistema de Gestão de Base de Dados Relacional (SGBDR) para a aplicação. O SQL Server, ou Microsoft SQL Server é, tal como referido, um (SGBDR) que foi desenvolvido pela Microsoft, tal como a *framework* utilizada para o desenvolvimento *backend*, o que fornece desde logo uma compatibilidade superior em comparação com as restantes opções sendo capaz de se integrar facilmente com as ferramentas de desenvolvimento do .NET, como o Visual Studio. Outra vantagem de utilizar o SQL Server é a sua capacidade de gerar grandes quantidades de dados com eficiência e escalabilidade onde também oferece recursos avançados de segurança, permitindo que os dados sejam protegidos por meio de autenticação de utilizador e criptografia.

Em resumo, o Microsoft SQL Server é um SGBDR de alta qualidade que oferece escalabilidade, segurança e facilidade de uso. Ele é especialmente adequado para uso com o .NET *Framework*, oferecendo uma variedade de recursos avançados para armazenar, recuperar e manipular dados em aplicações .NET.

3.1.4. *Vuetify*

Após as ferramentas utilizadas para o *backend* terem sido definidas era necessário selecionar que *framework* seria utilizado para o *frontend*, ou seja, a parte gráfica ao qual o utilizador iria ter acesso. Após alguma pesquisa pelas *frameworks* de *frontend* mais utilizadas e recomendadas ficou decidido utilizar o Vue.JS devido à familiaridade com a *framework* em outros projetos.

Uma das principais preocupações era o ajuste para os dispositivos móveis, ou seja, a adaptação a diferentes dispositivos e tamanhos de ecrã, sem perda de qualidade ou funcionalidade, o que ficou rapidamente resolvido com a *framework* Vuetify, uma *framework front-end* baseada no Vue.js, que oferece um design responsivo. Além desta principal vantagem o Vuetify disponibiliza uma ampla variedade de componentes de interface prontos a utilizar com consistência na aparência visual, como botões, barras de navegação, formulários, tabelas, etc., tornando o desenvolvimento de aplicativos *web* mais rápido e fácil.

Em resumo, a Vuetify é uma *framework front-end* poderosa e flexível que oferece uma ampla variedade de componentes pré-fabricados, personalização, documentação clara e fácil de entender, compatibilidade e suporte à acessibilidade, o que tornou a *framework* ideal para o desenvolvimento deste projeto.

3.1.5. *Docker*

Docker é uma plataforma de virtualização de aplicações que permite empacotar aplicativos em containers isolados e portáteis. Sendo uma ferramenta essencial para criar aplicações em ambientes de desenvolvimento locais e na nuvem.

O Docker tem um papel fundamental no desenvolvimento em ambiente de *development*, pois permite aos programadores criar e executar aplicações em *containers* isolados que podem ser implantados em qualquer lugar. Isso significa que se pode criar um ambiente de desenvolvimento consistente e portátil, independentemente do sistema operacional ou hardware usado.

Este oferece também uma ampla gama de benefícios para o desenvolvimento local. Assim permite criar uma imagem do aplicativo e todas as suas dependências em um único *container*. Isso significa que se pode facilmente partilhar e replicar o ambiente de desenvolvimento local com outros membros da equipa.

Outra vantagem do Docker é a sua capacidade de facilitar o processo de integração contínua e implantação contínua (CI/CD). O Docker pode ser usado para criar imagens de *container* para cada etapa do processo de desenvolvimento, desde a compilação e teste até a implantação em produção. Isso torna o processo de desenvolvimento e implantação mais eficiente, garantindo que as mudanças possam ser testadas e implantadas rapidamente.

Além disso, o Docker permite que se criem ambientes de desenvolvimento semelhantes aos ambientes de produção. Significando que se pode testar e implementar aplicações em ambientes que são semelhantes ao ambiente em que o aplicativo será implantado em produção. Isso ajuda a garantir que a aplicações funcione corretamente em produção e reduz o risco de problemas de compatibilidade.

3.1.6. *Python*

Outra linguagem de programação utilizada de modo a alcançar os objetivos propostos neste trabalho foi a linguagem Python, uma linguagem de alto nível bastante popular devido à sintaxe simples e a ser *open-source* com uma grande comunidade online. Esta linguagem foi assim utilizada no âmbito deste projeto de forma a programar de que modo as antenas irão comunicar com a API e com o servidor OPC-UA permitindo estabelecer a comunicação com ambos os serviços.

Para tal recorremos à biblioteca Python *open-source FreeOpcUA* que permite definir os contornos e implementar um servidor OPC-UA e um cliente, definir modelos e disponibilizar métodos que permitem a inserção dos dados no servidor de forma a ser possível consultar os mesmo à posteriori.

Embora exista diversas linguagens de programação que permitissem implementar as condições referidas foi optado por esta linguagem não só pela sintaxe bastante simples e a alguma experiência na sua utilização, mas também com o *deploy* do programa em mente num NodeMCU ou Raspberry PI.

Em resumo o Python é uma linguagem bastante prática com uma vasta comunidade que disponibiliza uma biblioteca que fornece todos os recursos que seriam necessários para desenvolver uma solução para programar as antenas e desta forma permitir a comunicação dos seus valores para ambos os serviços REST API e o servidor OPC-UA.

3.2. *Hardware*

Nesta secção serão enumerados os *hardwares* utilizados no projeto que permitiram a captação e transmissão/receção da informação relevante para o funcionamento do projeto.

3.2.1. ESP8266

O ESP8266 é um microprocessador Wi-Fi de baixo custo bastante utilizado em projetos de *Internet of Things* (IoT). Devido à sua conectividade Wi-Fi integrada o ESP8266 facilita a partilha de dados em tempo real entre servidores em nuvem e dispositivos IoT. Alem da sua conectividade o microcontrolador apresenta ainda uma gama de pinos GPIO (*General Purpose Input/Output*) que permitem uma interface flexível com sensores, atuadores e outros. Esta adaptabilidade é essencial para adaptar soluções de IoT às necessidades específicas de um projeto.

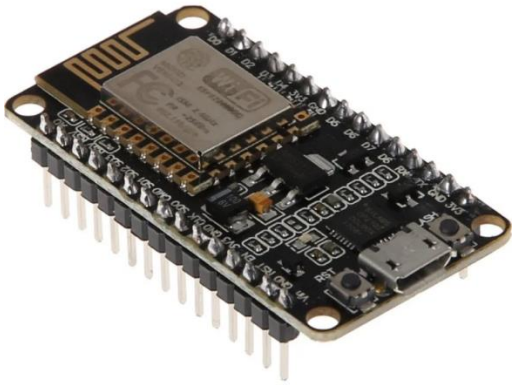


Figura 18 - ESP8266

Inicialmente este componente iria ser utilizado na interação entre as antenas, que também tem este hardware incorporado, e a solução web. Devido ao seu baixo custo energético e económico foi idealizado criar o servidor OPC-UA e REST com recurso a este dispositivo. No entanto essa situação foi descartada devido à incompatibilidade do sistema com a biblioteca *Python FreeOpcUa* utilizada para a criação do servidor OPC-UA com a

versão de Python disponível com o software do equipamento, que neste caso é MicroPython.

Embora não tenha sido utilizado com o propósito entendido este continua presente na solução ao ser incluído nas antenas que foram utilizadas para testar o sistema. Este componente apresenta-se conectado ao leitor RFID e quando uma captação é efetuada o RFID transmite o ângulo de captação da antena para o Raspberry Pi, componente que substituiu o ESP8266 na funcionalidade de conectar as antenas à aplicação.

3.2.2. Antena

As antenas utilizadas neste projeto, representadas na Figura 19, foram desenvolvidas pelo docente e orientador deste projeto com recurso a um leitor RFID e dois ESP8266. Os componentes que mantem os diversos hardwares juntos foram feitos com uma impressora 3D e com recurso a outros materiais. Tal como mencionado no capítulo de Estado da Arte este componente é essencial para o funcionamento de sistemas RFID e o que permite localizar e triangular a posição das *tags*. Nos capítulos seguintes será apresentado um estudo efetuado a este componente e será ainda explicado os valores e raio de ação do mesmo.

Figura 19 - Fotografia real da antena num dos cenários de teste

3.2.3. Raspberry Pi

O Raspberry Pi, introduzido em 2012 pela *Raspberry Pi Foundation*, é um dispositivo de computação acessível criado para facilitar a introdução à programação. Este é um computador versátil, apresentado na Figura 20, que pode funcionar como um dispositivo independente, interagindo com componentes de hardware e periféricos, e suporta vários sistemas operacionais podendo executar programas em Python, C, Java e outras línguas, tornando-se uma excelente ferramenta de aprendizagem.

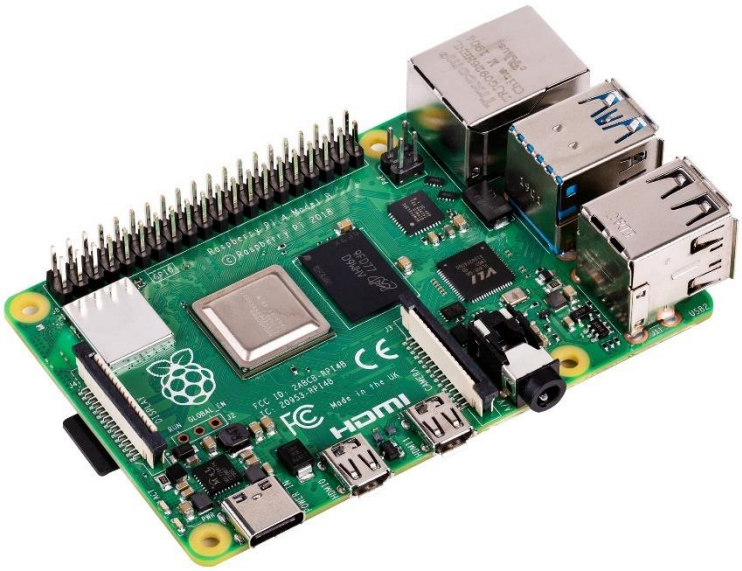


Figura 20 - Raspberry Pi

Devido à sua dimensão e o baixo consumo de energia este é ideal para aplicações *Internet of Things* (IoT), contribuindo para o crescimento de casas inteligentes e dispositivos interconectados, incluindo centros de media, consolas de jogos e centros de automação doméstica, entre outros projetos publicados.

Dentro do contexto deste projeto, este foi um hardware que só foi introduzido devido à incapacidade de recorrer ao ESP8266, abordado na secção anterior, como um servidor OPC-UA. Como foi referido acima, de modo a implementar este servidor e a sua comunicação recorremos à biblioteca de Python *FreeOpcUA* que corre em Python nativo, o que contraria o sistema do ESP8266 que corre apenas programas *MicroPython*, uma

versão criada para correr em microcontroladores, mas que não é compatível com todas as bibliotecas de Python.

3.2.4. *Tags*

As *tags* (etiquetas) ou identificadores utilizados neste projeto foram, tal como referido em capítulos anteriores, *tags* passivas devido a todos as propriedades e vantagens já referidas anteriormente. Assim dentro do contexto do projeto estas serão utilizadas para identificar objetos ou materiais em que o conhecimento da posição seja de interesse. De forma simplificada estas irão receber assim a energia emitida pelas antenas retratadas nas secções anteriores durante o seu processo de captação e irá retornar o id da mesma no processo de *uplink*.

4. Localização das *Tags*

Neste capítulo será discutido a solução e pensamento critico efetuado em relação ao problema encontrado na localização das *tags* RFID com recurso às antenas, neste serão então ilustrados e explicados os cálculos, cenários pensados e a resposta encontrada.

O problema foi crescendo à medida que o desenvolvimento era efetuado, sendo que sofreu várias alterações. Inicialmente teria sido pensado que as antenas seriam colocadas unicamente numa das paredes da sala o que permitiria implementar a solução básica de interseção das retas tendo a parede como referencial e sem termos em consideração se estávamos a obter as posições relativas ou absolutas.

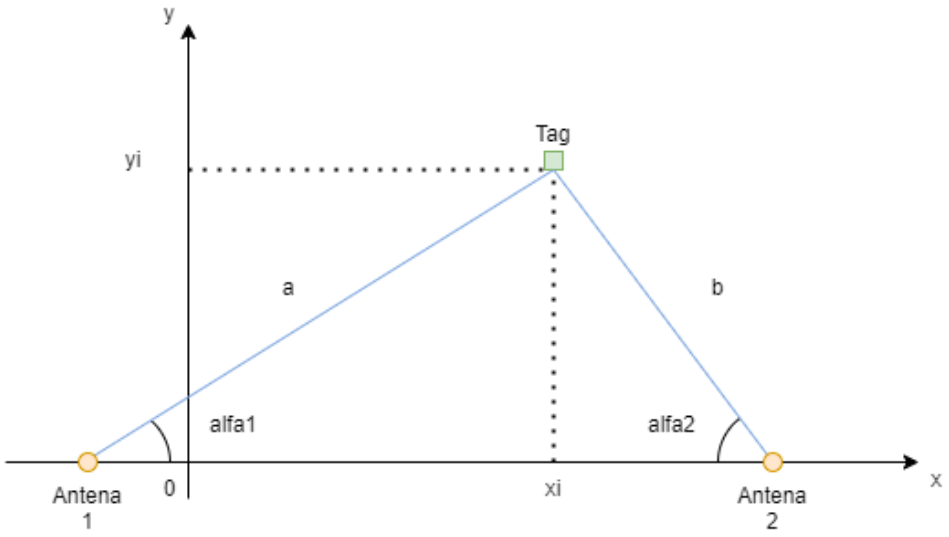


Figura 21 - Ilustração do problema num referêncial relativo

Na Figura 21 podemos encontrar a representação visual do que foi inicialmente pensado como solução para este projeto. Onde o cálculo seria extremamente direto sendo que podemos, como dito anteriormente, encontrar as antenas em apenas uma das paredes da divisão a cobrir e isto permitiria identificar facilmente a antena 1 e antena 2 apresentadas na figura e assim conhecer a posição da *tag* através do cálculo apresentado abaixo.

Utilizando a equação da reta $y = mx + b$ sendo "m" o declive da reta que pode ser alcançado através da tangente do ângulo, temos:

$$a1 = \tan(alfa1)$$

$$a2 = \tan(alfa2)$$

Assim as retas a e b são representas pelas equações da reta *y = a1x + b1* e *y = a2x + b2* respetivamente. Devido à posição da *tag* (*xi,yi)* ser o ponto de intercessão entre as duas retas, a fórmula que irá retornar a localização no referencial seria o seguinte sistema:

$$\begin{cases} y = a1x + b1 \\ y = a2x + b2 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y = a1x + b1 \\ x = \left(\frac{y - b2}{a2}\right) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y = a1 \times \left(\frac{y - b2}{a2}\right) + b1 \\ x = \left(\frac{y - b2}{a2}\right) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y = \left(\frac{a1 \times y}{a2}\right) - \left(\frac{a1 \times b2}{a2}\right) + b1 \\ x = \left(\frac{y - b2}{a2}\right) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y - \left(\frac{a1 \times y}{a2}\right) = -\left(\frac{a1 \times b2}{a2}\right) + b1 \\ x = \left(\frac{y - b2}{a2}\right) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y \times \left(1 - \frac{a1}{a2}\right) = -\left(\frac{a1 \times b2}{a2}\right) + b1 \\ x = \left(\frac{y - b2}{a2}\right) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y = \dfrac{-\left(\frac{a1 \times b2}{a2}\right) + b1}{1 - \frac{a1}{a2}} \\ x = \left(\frac{y - b2}{a2}\right) \end{cases}$$

$$\begin{cases} 0 = a1x + b1 \\ 0 = a2x + b2 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} b1 = -a1x \\ b2 = -a2x \end{cases}$$

Após alguma discussão o cenário foi repensado e foi desenvolvido tendo em conta que as antenas poderiam estar em qualquer parede da zona industrial o que a nível de eficiência e alcance iria melhorar substancialmente a solução sendo que seria coberta uma área maior da sala. No entanto não seria o suficiente, tendo em conta que uma sala ou divisão na zona de fabrico pode ser bastante vasta estaríamos bastante limitados caso as antenas fossem posicionadas somente nas paredes da divisão.

Foi então iniciado conversações para encontrar alguma forma de poderem ser introduzidas as antenas em qualquer posição da sala. Desta forma a cobertura da mesma seria alcançada consoante o número de antenas utilizadas e com a disposição que o cliente achasse melhor. Com este cenário em mente o problema passou a ser utilizar o sistema já implementado nesta solução visto que seria necessário termos conhecimento de qual antena seria a considerada mais à esquerda o que seria necessário termos o input do utilizador de qual seria a orientação das antenas em relação à sala. Não seria impossível, mas tornaria o processo e a certeza bastante sugestível a erro, agravando a possibilidade de já ser inserido com o erro a posição absoluta das antenas.

Assim foram imaginados os cenários possíveis das posições das antenas e de que forma poderia ser obtido a posição da *tag* sem ser necessário conhecer qual das duas se encontra mais à esquerda nem a orientação das mesmas.

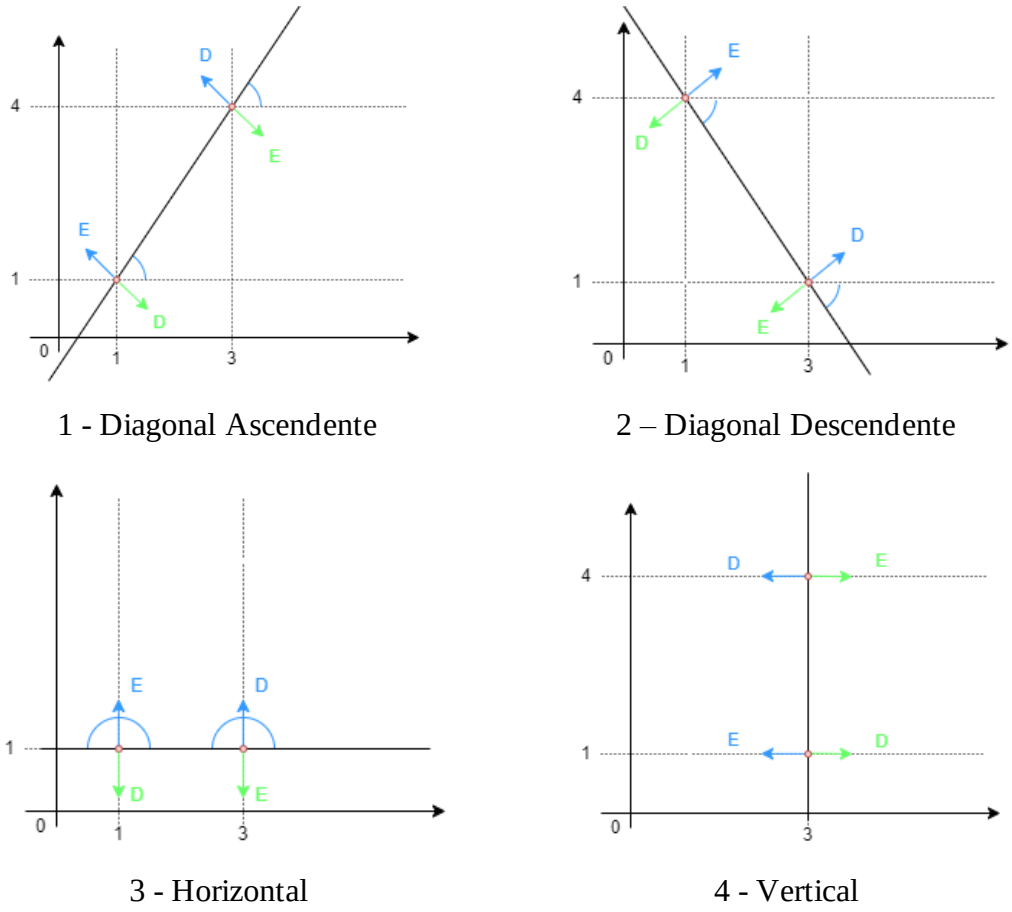


Figura 22 - Possíveis posições das antenas no referencial cartesiano (Diagonal Crescente, Diagonal Decrescente em cima e horizontal, vertical em baixo respetivamente)

A Figura 22 demonstra as possíveis situações que podíamos encontrar quando são introduzidas as antenas no chão da fábrica, onde podemos encontrar ocorrências em que estas se encontrem na diagonal (figuras 1 e 2), vertical e horizontal (figuras 3 e 4 respetivamente) em relação ao referencial. Também está indicado na figura qual seria considerada a antena à esquerda para cada situação pois será necessário para o algoritmo desenhado com esta solução em mente.

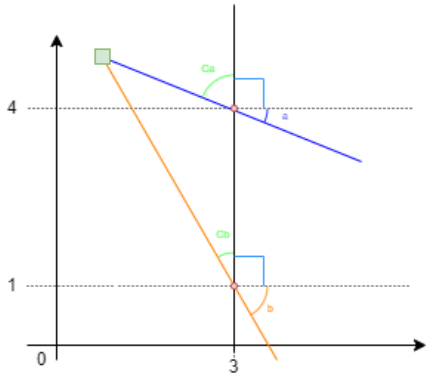
Após encontrarmos quais seriam os cenários possíveis analisámos caso a caso, que serão explicados e demonstrados nas secções seguintes, como obtivemos o ângulo entre a antena e a *tag* que nos permitirá obter a sua posição absoluta.

De forma a podermos compreender algumas das situações apresentadas e as suas soluções vamos considerar, tal como fizemos na análise do problema que o eixo das ordenadas representa a largura da divisão e que o eixo das abcissas o comprimento da

mesma sendo que o ponto D(xD, yD) representa o canto superior direito da divisão e assim delimita os valores máximos para aquela sala. Esta informação será necessária para definir a posição da *tag* quando existir complicações com a triangulação como será demonstrado mais à frente.

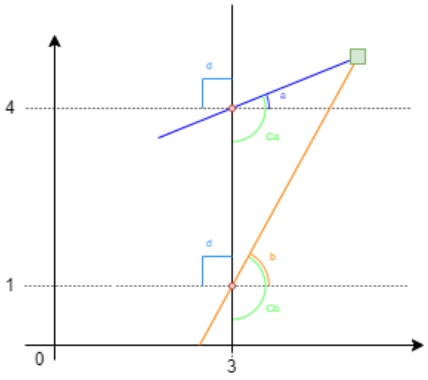
4.1. Ângulo entre a reta formada pelas antenas e o eixo das abcissas é 90º

Nesta secção será analisado os casos em que no referencial a reta criada entre as duas coordenas da antena formam uma reta vertical, x = xA, sendo xA os valores das abcissas das antenas, significando que não apresenta declive.



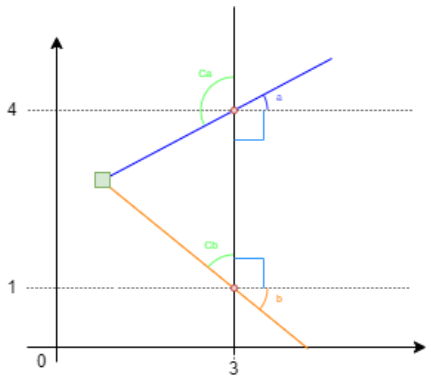
$\sphericalangle a = -(90 - \sphericalangle Ca)$

$\sphericalangle b = -(90 - \sphericalangle Cb)$



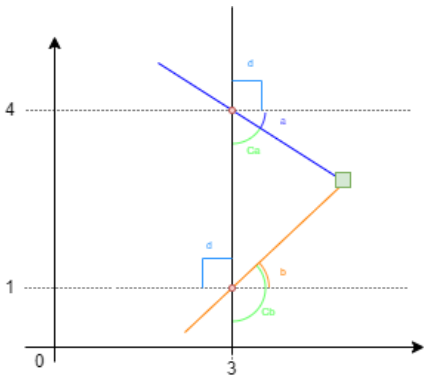
$\sphericalangle a = \sphericalangle Ca - 90$

$\sphericalangle b = \sphericalangle Cb - 90$



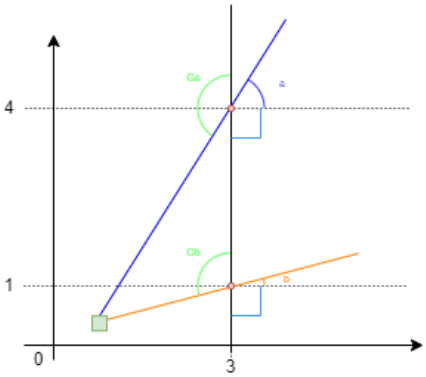
$\sphericalangle a = \sphericalangle Ca - 90$

$\sphericalangle b = -(90 - \sphericalangle Cb)$



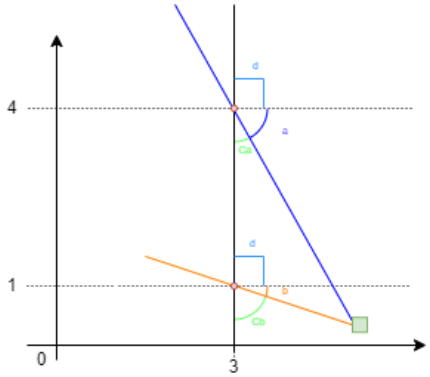
$\sphericalangle a = -(90 - \sphericalangle Ca)$

$\sphericalangle b = \sphericalangle Cb - 90$



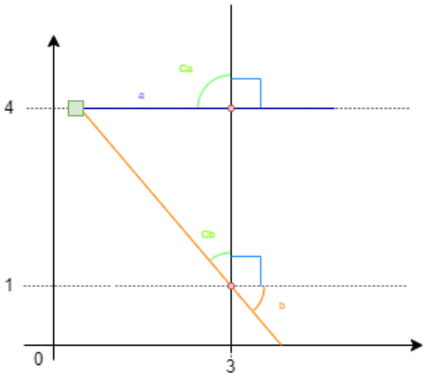
$\measuredangle a = \measuredangle Ca - 90$

$\measuredangle b = \measuredangle Cb - 90$



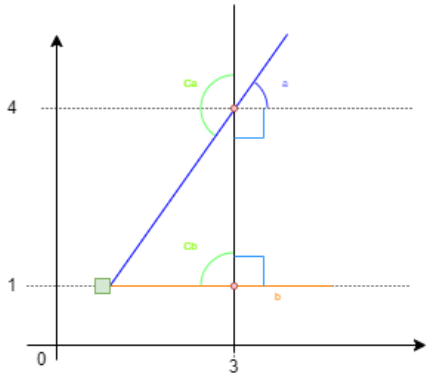
$\measuredangle a = -(90 - \measuredangle Ca)$

$\measuredangle b = -(90 - \measuredangle Cb)$



$\measuredangle a = 0$

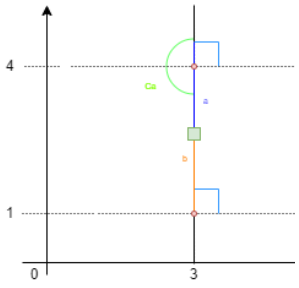
$\measuredangle b = -(90 - \measuredangle Cb)$



$\measuredangle a = \measuredangle Ca - 90$

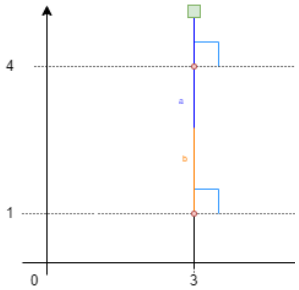
$\measuredangle b = 0$

Além destas situações podemos encontrar as exceções em que os ângulos das antenas são 0º e 180º o que significa que a reta formada pela antena e a *tag* seria vertical, ou seja sem declive. As figuras abaixo representam esta condição e os possíveis cenários que resultariam da mesma.



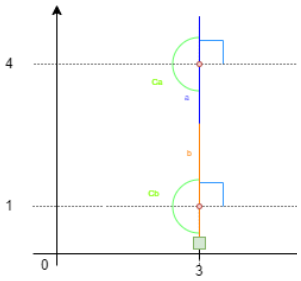
$\sphericalangle a = 90$

$\sphericalangle b = 90$



$\sphericalangle a = 90$

$\sphericalangle b = 90$



$\sphericalangle a = 90$

$\sphericalangle b = 90$

Como é possível visualizar iremos encontrar possíveis situações em que existe uma sobreposição das retas quando a reta formada entre a antena e a *tag* forma um ângulo reto com o eixo das abcissas o que causa constrangimento na triangulação não permitindo encontrar a posição exata da *tag*. Desse modo para cada uma das situações apresentadas a posição da *tag* será considerado o ponto médio entre as duas antenas, (xA, yD) e (xA, 0) respetivamente.

Através das ilustrações presentes acima foi possível criar o seguinte algoritmo para a condição em que antenas formam uma reta sem declive, em que $\sphericalangle C$ representa o ângulo com que a antena detetou a *tag*, como demonstrado na Figura 23.

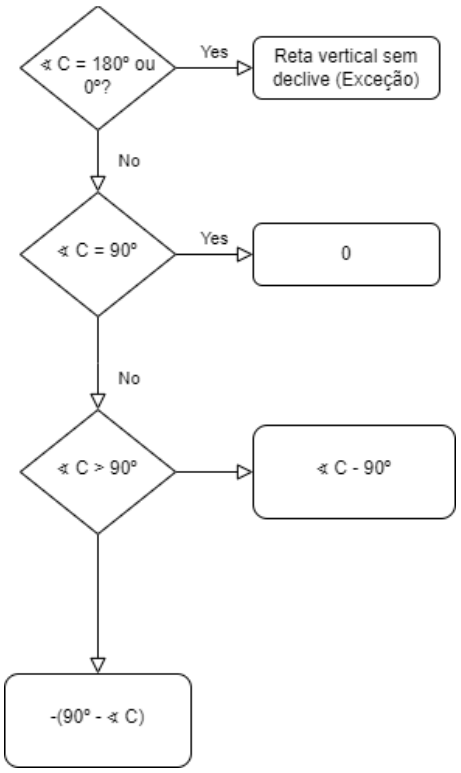
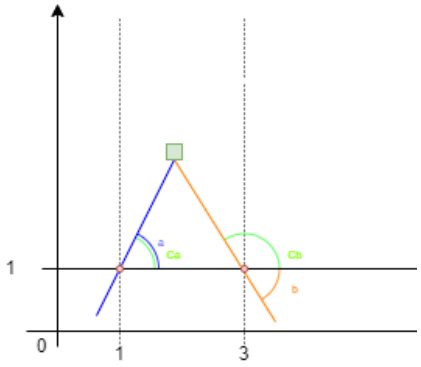


Figura 23 – Algoritmo encontrado para o quando o ângulo entre a reta formada pelas antenas e o eixo das abcissas é 90º

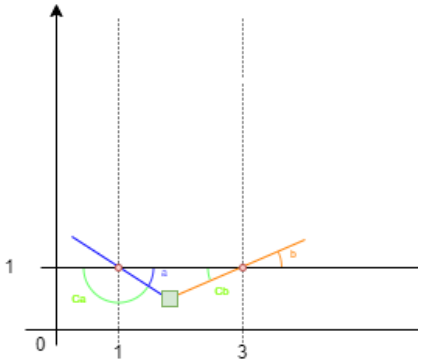
4.2. Ângulo entre a reta formada pelas antenas e o eixo das abcissas é zero

Nesta secção serão ilustrados os casos e os possíveis cenários em que a reta formada pelos dois pontos cartesianos das antenas é uma reta horizontal, y = yA, em que yA é o valor das ordenadas das antenas, e assim apresenta um coeficiente angular de valor zero.



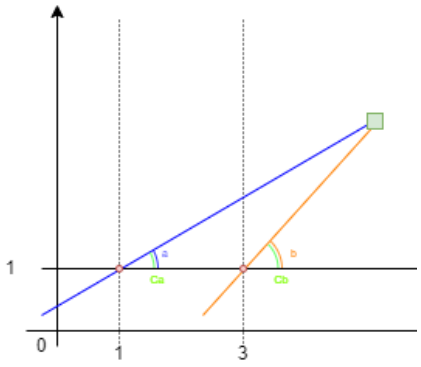
$\sphericalangle a = \sphericalangle Ca$

$\sphericalangle b = -(180 - \sphericalangle Cb)$



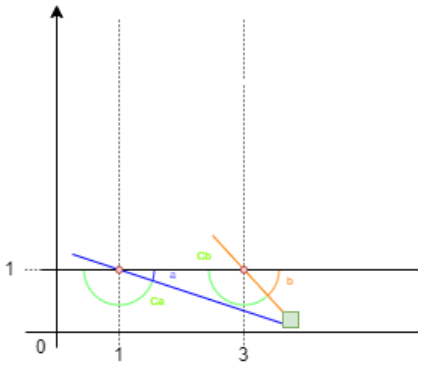
$\sphericalangle a = -(180 - \sphericalangle Ca)$

$\sphericalangle b = \sphericalangle Cb$



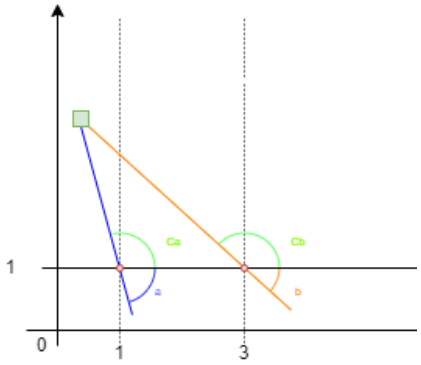
$\sphericalangle a = \sphericalangle Ca$

$\sphericalangle b = \sphericalangle Cb$



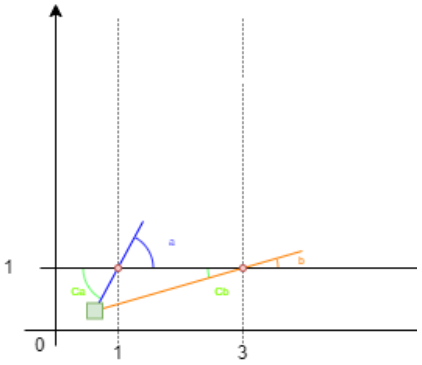
$\sphericalangle a = -(180 - \sphericalangle Ca)$

$\sphericalangle b = -(180 - \sphericalangle Cb)$



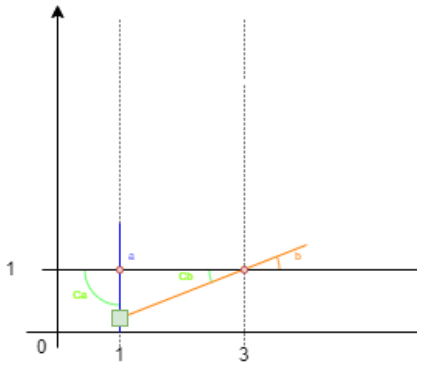
$\sphericalangle a = -(180 - \sphericalangle Ca)$

$\sphericalangle b = -(180 - \sphericalangle Cb)$



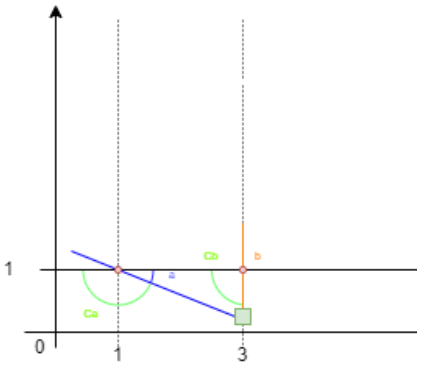
$\sphericalangle a = \sphericalangle Ca$

$\sphericalangle b = \sphericalangle Cb$



$\sphericalangle a = 90$

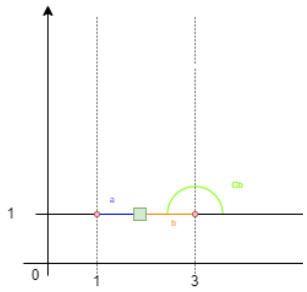
$\sphericalangle b = \sphericalangle Cb$



$\sphericalangle a = \sphericalangle Ca$

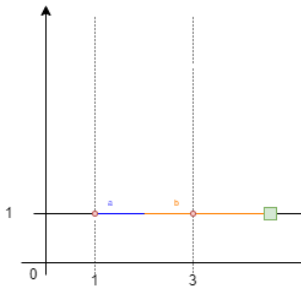
$\sphericalangle b = 90$

Tal como referido na secção anterior existem certas situações em que a posição da *tag* não será facilmente reconhecida devido à sobreposição das antenas, tal como ilustrado nas figuras abaixo.



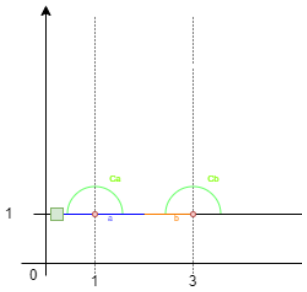
$\sphericalangle a = 0$

$\sphericalangle b = 180$



$\sphericalangle a = 0$

$\sphericalangle b = 0$



$\sphericalangle a = 180$

$\sphericalangle b = 180$

Repetido o exercício feito na secção anterior iremos, para os casos em que a *tag* se encontra posicionada em cima da reta formada entre as duas antenas, considerar que para cada figura a posição da *tag* no referencial será o ponto médio entre as duas antenas, (xD,yA) e (0, yA) respetivamente.

Assim foi possível construir um algoritmo, como representado na Figura 24, que nos permita obter o ângulo da reta formada entre a antena e a *tag* e o eixo das abcissas.

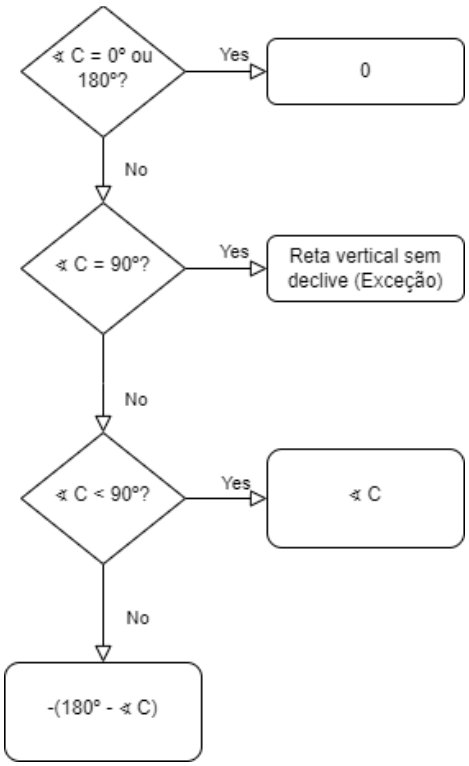
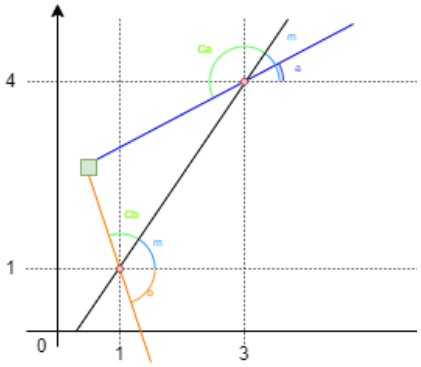


Figura 24 - Algoritmo encontrado para o quando o ângulo entre a reta formada pelas antenas e o eixo das abcissas é zero

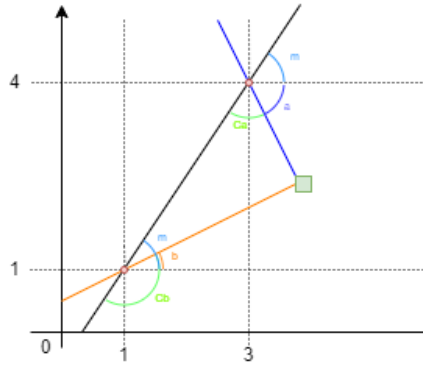
4.3. Ângulo entre a reta formada pelas antenas e o eixo das abcissas é inferior a 90º

Nesta secção serão explorados os cenários correspondentes à situação em que o ângulo formado entre as antenas e o eixo das abcissas é inferior a 90º o que determina que a reta é ascendente.



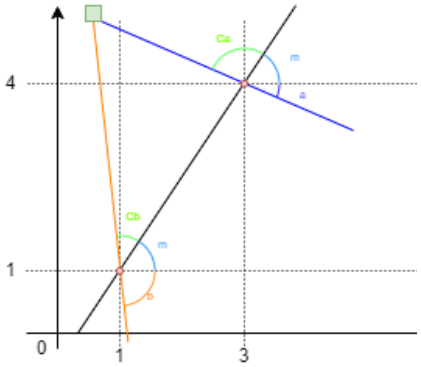
$$\measuredangle a = \measuredangle m - (180 - \measuredangle Ca)$$

$$\measuredangle b = -(180 - (\measuredangle m + \measuredangle Cb))$$



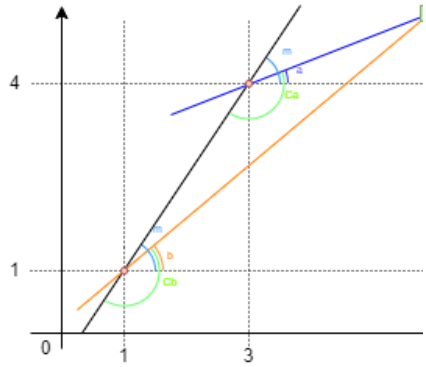
$$\measuredangle a = -(180 - (\measuredangle m + \measuredangle Ca))$$

$$\measuredangle b = \measuredangle m - (180 - \measuredangle Cb)$$



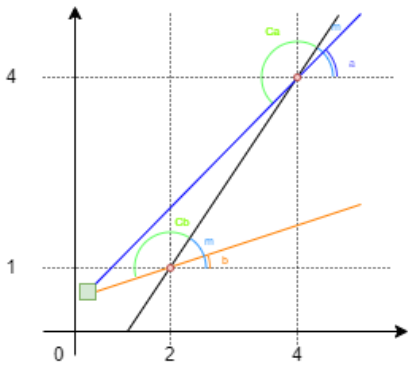
$$\measuredangle a = -\big(180 - (\measuredangle m + \measuredangle Ca)\big)$$

$$\measuredangle b = -(180 - (\measuredangle m + \measuredangle Cb))$$



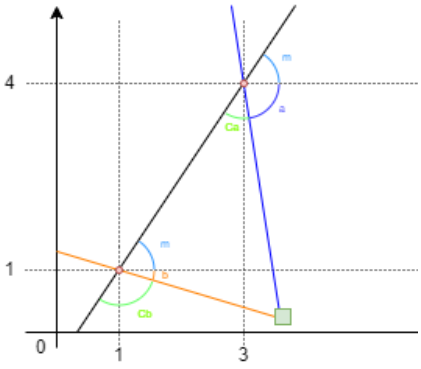
$$\measuredangle a = \measuredangle m - (180 - \measuredangle Ca)$$

$$\measuredangle b = \measuredangle m - (180 - \measuredangle Cb)$$



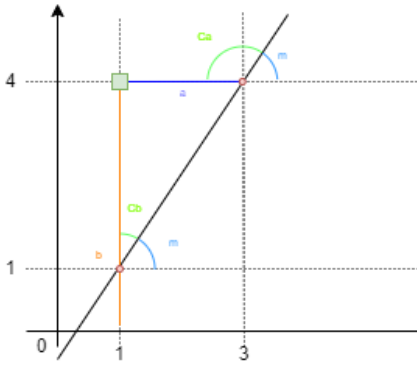
$$\sphericalangle a = \sphericalangle m - (180 - \sphericalangle Ca)$$

$$\sphericalangle b = \sphericalangle m - (180 - \sphericalangle Cb)$$



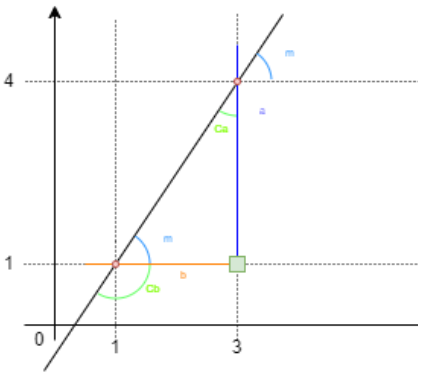
$$\sphericalangle a = -\big(180 - (\sphericalangle m + \sphericalangle Ca)\big)$$

$$\sphericalangle b = -(180 - (\sphericalangle m + \sphericalangle Cb))$$



$$\sphericalangle a = 0$$

$$\sphericalangle b = 90$$



$$\sphericalangle a = 90$$

$$\sphericalangle b = 0$$

Tal como visto anteriormente este exercício de análise permitiu obter o algoritmo presente na Figura 25, em que $\sphericalangle m$ representa o ângulo formado entre o eixo das abcissas e a reta criada entre as duas posições das antenas:

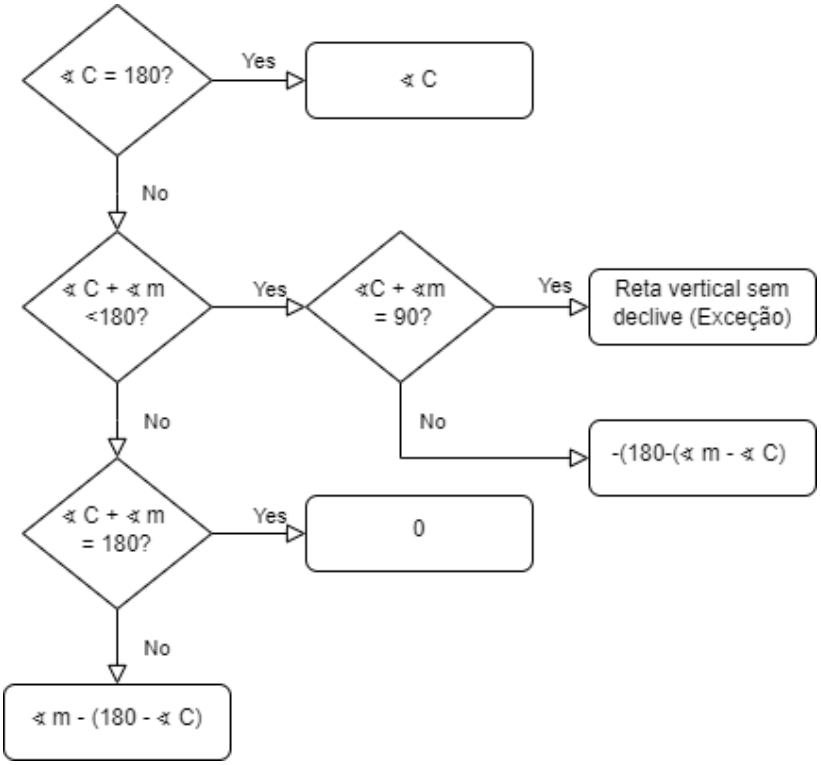
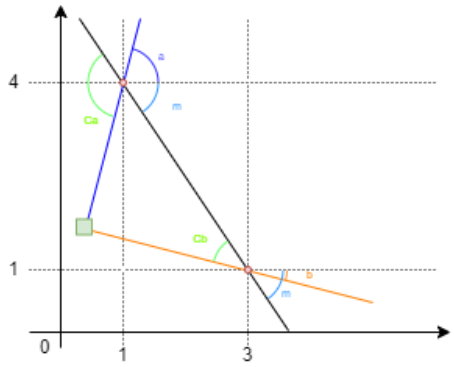


Figura 25 - Algoritmo encontrado para o quando o ângulo entre a reta formada pelas antenas e o eixo das abcissas é inferior 90º

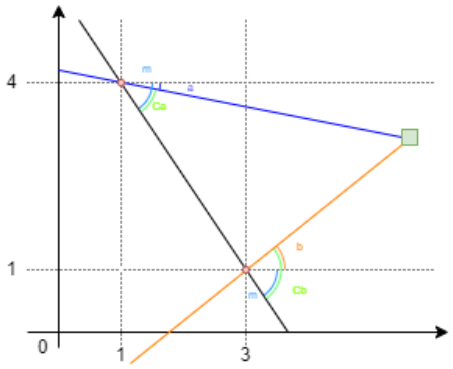
4.4. Ângulo entre a reta formada pelas antenas e o eixo das abcissas é superior a 90º

Nesta secção serão verificados os casos em que o ângulo entre a reta formado pelos dois pontos da antena e o eixo das abcissas formam um ângulo superior a 90º indicando que se trata de uma reta decrescente.



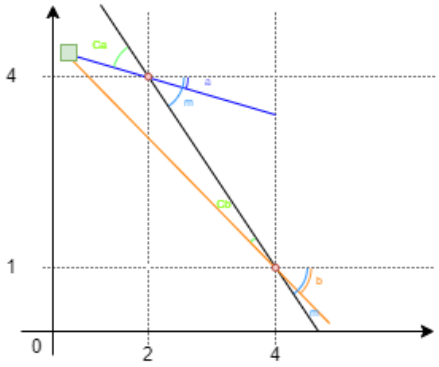
$$\measuredangle a = \measuredangle Ca - |\measuredangle m|$$

$$\measuredangle b = -(|\measuredangle m| - \measuredangle Cb)$$



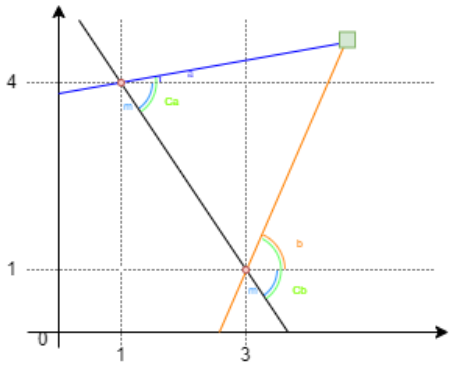
$$\measuredangle a = -(|\measuredangle m| - \measuredangle Ca)$$

$$\measuredangle b = \measuredangle Cb - |\measuredangle m|$$



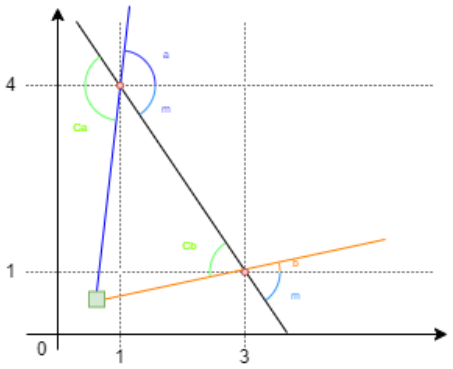
$$\measuredangle a = -(|\measuredangle m| - \measuredangle Ca)$$

$$\measuredangle b = -(|\measuredangle m| - \measuredangle Cb)$$



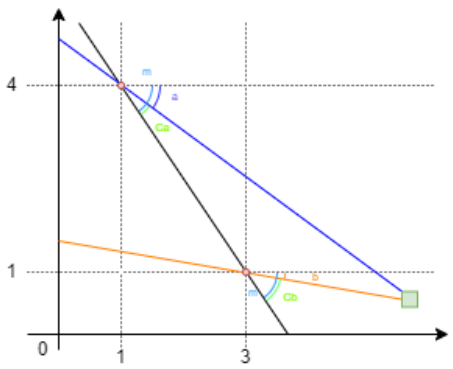
$$\measuredangle a = \measuredangle Ca - |\measuredangle m|$$

$$\measuredangle b = \measuredangle Cb - |\measuredangle m|$$



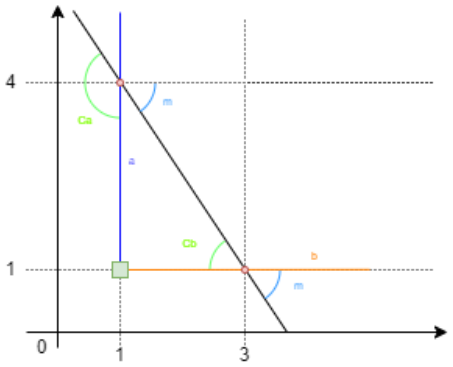
$\sphericalangle a = \sphericalangle Ca - |\sphericalangle m|$

$\sphericalangle b = \sphericalangle Cb - |\sphericalangle m|$



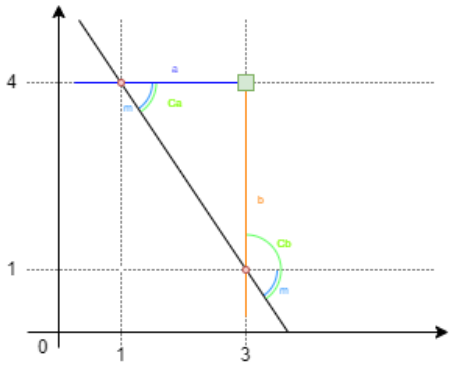
$\sphericalangle a = -(|\sphericalangle m| - \sphericalangle Ca)$

$\sphericalangle b = -(|\sphericalangle m| - \sphericalangle Cb)$



$\sphericalangle a = 90$

$\sphericalangle b = 0$



$\sphericalangle a = 0$

$\sphericalangle b = 90$

Após analisarmos os casos apresentados acima obtivemos o algoritmo representado na Figura 26.

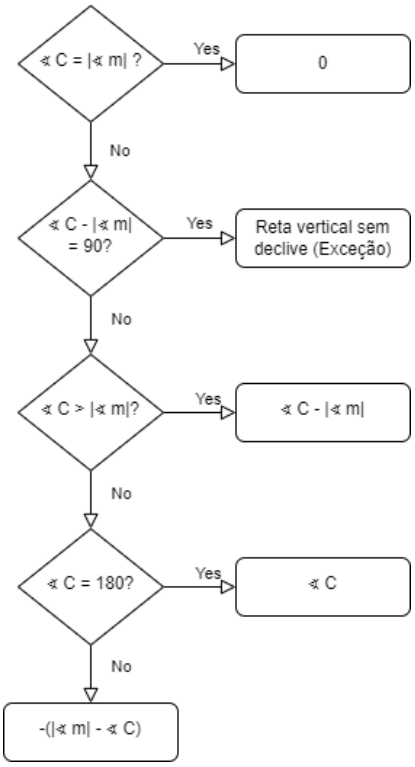


Figura 26 - Algoritmo encontrado para o quando o ângulo entre a reta formada pelas antenas e o eixo das abcissas é superior a 90º

4.5. Cálculo posicional da *tag*

Com a avaliação dos casos terminada e com os algoritmos encontrados para obtermos o ângulo que a reta entre a antena e *tag* forma com o eixo das abcissas conseguimos finalmente realizar a triangulação entre as duas retas, da antena mais à esquerda e da antena mais à direita de forma a conseguirmos encontrar a interseção das mesmas e obter assim a posição exata da *tag*.

$$\begin{cases} yT = m1 \times xT + b1 \\ yT = m2 \times xT + b2 \end{cases}$$

Sendo que m1 e m2 representam o declive das retas e sabendo o ângulo formado entre elas e o eixo das abcissas podemos obter o seu valor, ou seja o coeficiente angular da seguinte forma:

$$m1 = \tan(\measuredangle a)$$

$$m2 = \tan(\measuredangle b)$$

Tendo como conhecimento a equação da reta $y = mx + b$, podemos encontrar a variável *b*, ou seja, o valor do *x* quando a reta intercepta o eixo das ordenadas, através da equação $b = y - mx$. Tendo em conta que conhecemos um dos pontos que atravessam as retas, sendo estes o ponto A(xA,yA) e B(xB,yB) que representam as localizações das antenas da esquerda e direita respetivamente no plano cartesiano conseguimos obter os seus valores respetivos.

$$b1 = yA - (m1 \times xA)$$

$$b2 = yB - (m2 \times xB)$$

Assim continuando o sistema apresentado anteriormente, temos de igualar as retas de forma podermos trabalhar apenas numa das variáveis.

$$m1 \times xT + b1 = m2 \times xT + b2 \Leftrightarrow$$

$$\Leftrightarrow m1 \times xT - m2 \times xT = b2 - b1 \Leftrightarrow$$

$$\Leftrightarrow xT = \frac{(b2 - b1)}{(m1 - m2)}$$

Após obtermos o valor do *xT* podemos assim olhar para apenas uma das retas e recorrendo novamente à equação fundamental da reta obter o valor da variável *yT* que devido às retas se intersetarem no ponto em que o valor das abcissas é *xT* será igual.

$$yT = m1 \times xT + b1$$

Desta forma foi nos permitido encontrar e obter a posição no referencial da tag (xT,yT) através da interseção das retas obtidas com recurso ao ângulo calculado por cada uma quando é captado o sinal RFID.

4.5.1. Exceções

No desenrolar do processo foram identificados alguns casos que poderiam comprometer o desempenho do algoritmo encontrado, nomeadamente os casos em que o ângulo de captação da *tag* cria uma reta vertical entre a antena e a mesma, entre outros. O primeiro caso a considerar seria a situação em que ambos os ângulos têm o mesmo valor e a reta formada entre as antenas e o eixo das abcissas não ser vertical nem horizontal, escapando assim as condições mencionadas anteriormente. Este caso é um caso de erro de deteção visto criar retas paralelas que não se intersetam tornando assim o cálculo da posição impossível.

Outro caso será aquele em que o ângulo de deteção da *tag* de apenas uma das antenas F(xF,yF) forma uma reta vertical x = xF. Para este, o cálculo, utilizando H(xH, yH) como a posição da antena oposta e $\measuredangle c$ como o ângulo de deteção da *tag* da mesma, é:

$$m1 = \tan(\measuredangle c)$$

$$y = mx + b \Leftrightarrow$$
$$\Leftrightarrow y - mx = b \Leftrightarrow$$
$$\Leftrightarrow b = y - mx$$

Assim substituindo a expressa encontrada pelas variáveis conhecidas temos:

$$b = yH - (m1 \times xH)$$

$$y = m1 \times xF + b$$

Com este podemos encontrar a localização da *tag* sendo que a sua localização seria as coordenadas (xF,y)

4.5.2. Raio de captação das antenas

Embora inicialmente pensado que o ângulo seria como retratado no capítulo 3, ou seja, entre 0º e 180º, a antena fornecida tem uma rotação entre os -54º e 54º, sendo necessário existir uma correção do ângulo captado pela mesma.

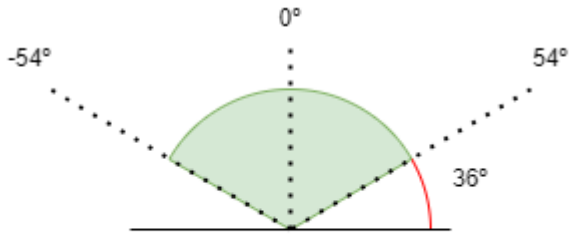


Figura 27 - Raio de captação do ângulo da tag em relação à antena

Como é possível visualizar na Figura 27, observando a antena a partir de cima, o raio de ação apresentado é de -54º a 0º quando a antena gire sobre si à esquerda e de 0º a 54º quando esta se vira para a direita. Tal como mencionado isto cria uma discrepância quanto ao originalmente pensado sobre o raio de ação da antena e sobre os cálculos efetuados. Assim para adaptar o sistema ao problema, e considerando que os algoritmos foram considerados com o ângulo sendo obtido do movimento da direita para a esquerda da antena:

$$54° \rightarrow 36° \ \wedge \ -54° \rightarrow 144° \ \Leftrightarrow$$
$$\Leftrightarrow |-54 \ - \ x| = 144 \ \wedge \ |54 - x| = \ 36 \ \Leftrightarrow$$
$$x = 90$$

Aplicando a fórmula $y = |\measuredangle - 90|$, em que $\measuredangle$ corresponde ao ângulo obtido pela antena, podemos sempre obter o valor "real" do ângulo dentro do contexto do problema.

5. Implementação

Este capítulo tem como objetivo demonstrar a arquitetura e soluções apresentadas para alcançar o objetivo apresentado no início deste trabalho. Assim este foi divido em várias secções de forma a abordar de forma aprofundada cada uma das tecnologias utilizadas e os desenvolvimentos feitos em prol de conectar todos os componentes desde a deteção das *tags* até à apresentação dos resultados para o consumidor final.

5.1. Arquitetura geral do sistema

Esta secção serve de introdução à informação que será discutida ao longos das restantes secções que demonstra as ligações entre os diferentes equipamentos utilizados no projeto demonstrando assim no seu global a arquitetura do sistema.

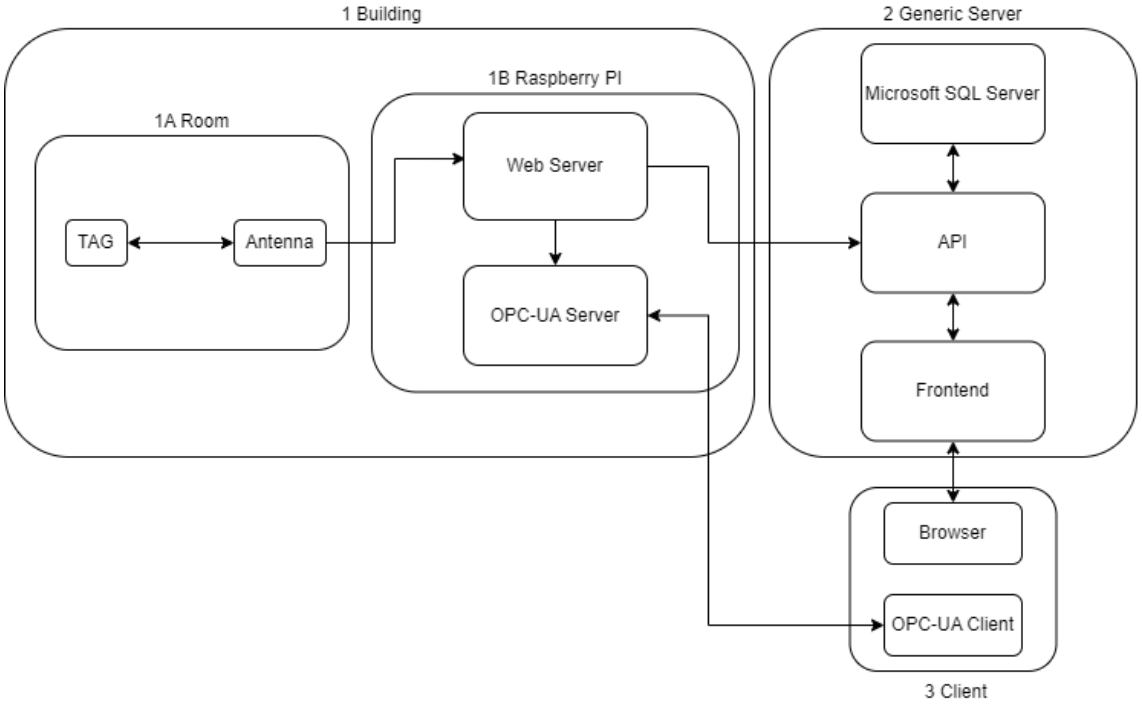


Figura 28 - Diagrama da arquitetura da solução

Na Figura 28 estão representados os fluxos e a arquitetura do sistema idealizado para este projeto, desde a conexão entre a *tag* até à possível visualização e alteração dos dados pelo cliente. De forma a se tornar mais fácil a compreensão do mesmo e para dividir em diferentes partes, a imagem apresenta uma numeração e consequente ordem que servirá de guia para a explicação da arquitetura. Assim podemos ver que o sistema se reparte em 3 grandes componentes:

- Módulo 1 - Edifício – que representa onde serão feitas as captações das *tags*. Podemos ver que está dividido ainda entre 1A e 1B, nomeadamente as divisões e o Raspbery PI respetivamente devido à complexidade existente no modulo 1B.

- Módulo 2 - Servidor Genérico – representando o servidor onde será feito o *deploy* da aplicação e em geral oferece a grande parte das funcionalidades exigidas para o sucesso deste trabalho

- 3 - Cliente – devido à necessidade de disponibilizarmos duas formas de visualização os dados criados pelas antenas, o cliente terá um modulo próprio de forma a explicar os requerimentos necessários para aceder a ambos os serviços

5.2. Módulo 1 (Edifício/Fábrica)

5.2.1. Módulo 1A (Divisão)

Ao longo desta secção será explicado o funcionamento teórico do que foi implementado dentro das divisões dos edifícios onde serão colocados os equipamentos físicos que fornecem a informação essencial para a deteção e obtenção da localização dos objetos.

O módulo 1A representa a interação entre as *tags* e as antenas. Nesta podemos ver a conexão já abordada entre as *tags* RFID passivas e as antenas no capítulo 1. O bom funcionamento da solução depende de as antenas estarem a apontar nas mesmas direções paralelas uma à outra (correspondendo ao ângulo de zero graus) de forma a ser possível efetuar a triangulação mencionada em capítulos anteriores.

Desta forma e com o enquadramento teórico desta conexão obtido o foco será no método de captação do ângulo da antena e como irá a antena comunicar esta informação para o módulo 1B (Raspberry PI).

Após a receção deste sinal, captação do ângulo e transmitido esta informação ao ESP8266 o mesmo deverá envia um POST *Request* para o *webserver* alojado no módulo 1B, que será abordado na secção seguinte.

5.2.2. Módulo 1B (Raspberry PI)

Esta secção serve de continuação à anterior e completa o fluxo descrito na componente 1 referido na secção da arquitetura. Dentro do enquadramento este módulo tem como principal funcionalidade servir de ponte entre os componentes 1 e 2 e ainda disponibilizar a informação obtida na antena para um servidor OPC-UA.

Os servidores foram desenvolvidos com recurso à linguagem Python devido à compatibilidade com o Raspberry PI e à existência das seguintes bibliotecas *FreeOpcUa* e *Web Framework FastAPI*. Estas permitiram estabelecer e criar um servidor OPC-UA de modo a conseguirmos criar os *nodes* e objetos necessários para apresentar a informação a um OPC-UA cliente e ainda criar uma API para assim criar um *endpoint* para os POST *Requests* efetuados pelo ESP8266, respetivamente.

A dificuldade apresentada neste módulo foi comunicar a informação recebida no *request* mencionado para o servidor OPC-UA. Para combater este problema a solução passou por conseguir incorporar no mesmo *script* ambos os servidores para haver assim uma partilha de informação e conhecimento dos objetos entre os dois.

Assim e como apresentado nos excertos seguintes com o conhecimento e acesso ao servidor o *endpoint* foi desenvolvido para quando receber o Log, antes da partilha do mesmo para a API desenvolvida na componente 2, é pesquisado no servidor OPC-UA o *node* correspondente à antena, que é reconhecido pelo endereço IP e se for encontrado um novo objeto correspondente ao *timestamp* é criado com as variáveis *angle*, RSSI e tagEPC que é o identificador da *tag* encontrada.

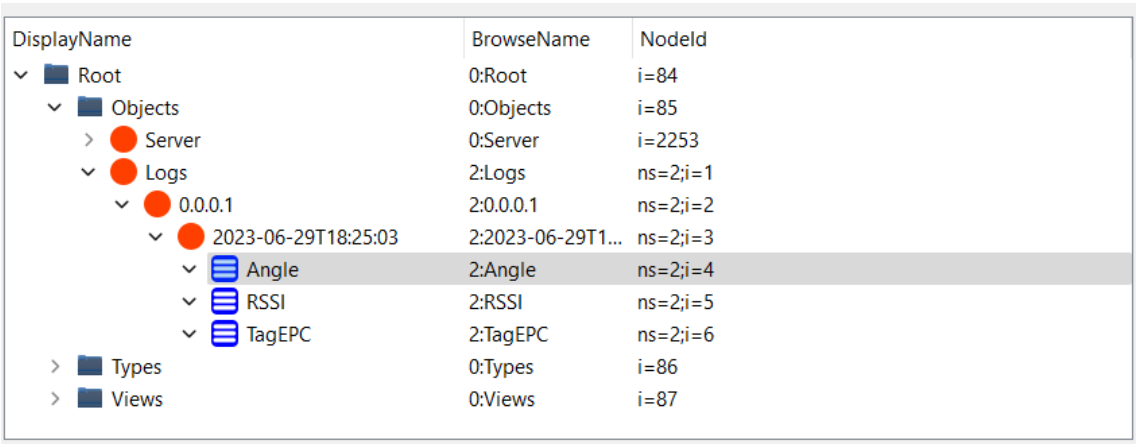


Figura 29 - Visualização do modelo dos dados no OPC-UA Cliente

A Figura 29 apresenta uma das componentes que será abordado mais à frente no componente 3, ou seja, o Cliente, pois demonstra um exemplo de uma conexão de um OPC-UA Cliente ao OPC-UA servidor requisitando a informação nele guardada e assim apresenta também a estrutura dos *nodes* e objetos desenvolvida nesta solução.

```
    from pydantic import BaseModel


    class Log(BaseModel):
        rssi : int
        ip_address : str
        angle : int
        tag_epc : str
```

No excerto de código acima podemos encontrar uma classe desenvolvida em Python utilizada como modelo de informação a receber no *endpoint* disponibilizado pelo servidor REST.

```
from opcua import Server,uamethod, ua
import requests
from fastapi import FastAPI, HTTPException
from models import Log
import time
from datetime import datetime
import threading
import json
import remoteapiconfig as conf

app = FastAPI()

@app.get('/')
async def root():
    return {"Hello":"World"}
```

```
@app.post('/api/v1/registerLog')
def register_log(log : Log):

    try:
        logsChildren = logs.get_children()
        ipAddressAlreadyExists = False

        for node in logsChildren:
            if node.get_display_name().Text == log.ip_address:
                IPAddress = node
                ipAddressAlreadyExists = True

        if not ipAddressAlreadyExists:
            IPAddress = logs.add_object(addspace, log.ip_address)

        timestampString = datetime.now().strftime("%Y-%m-%dT%H:%M:%S")
        TimeStamp = IPAddress.add_object(addspace, timestampString)
        TimeStamp.add_variable(addspace,"Angle", log.angle)
        TimeStamp.add_variable(addspace,"RSSI", log.rssi)
        TimeStamp.add_variable(addspace,"TagEPC", log.tag_epc)
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=f"{e}"
        )

    if log.angle < -56 or log.angle > 56:
        print(f"Log.angle is {log.angle}")
        raise HTTPException(
            status_code=400,
            detail="Will be ignored since angle is not between correct
range"
        )
    else:
        const_angle = 12
        angle = (log.angle + 90) + const_angle

    #Register in Web Server
    payload = {
        'rssi': log.rssi,
        'ipAddress': log.ip_address,
        'angle': angle,
        'tagEPC': log.tag_epc
    }
    print("Payload:",payload)
    result = requests.post('http://' + conf.remote_api_url +
'/api/logs',json=payload, verify=False, timeout=6)
    print("HTTP Result:",result.text)
```

```
    if result.status_code != 201:
        raise HTTPException(
            status_code=500,
            detail=f"something went wrong writing to server"
            )

    return {"opc_result" : 201, "http_result" : result.status_code}

def start_server():
    server.start()
    print("Server started at {}".format(url))

try:
    server = Server()

    url = "opc.tcp://192.168.1.169:4840"
    server.set_endpoint(url)

    name = "OPCUA_SIMULATION_SERVER"
    addspace = server.register_namespace(name)

    node = server.get_objects_node()
    logs = node.add_object(addspace, "Logs")

    t1 = threading.Thread(target=start_server)
    t1.start()

except KeyboardInterrupt:
    server.stop()
    print("server stopped at {}".format(url))
```

5.3. Módulo 2 (Servidor Genérico)

Nesta secção será abordada provavelmente a componente mais exigente e trabalhada ao longo do projeto e que sem contar com a ligação OPC-UA é a ponte entre os dados físicos e o cliente. Nesta será tratada toda a parte de armazenamento, tratamento e disponibilização dos dados para o utilizador ter a capacidade de os visualizar de uma forma apelativa e compreensível.

Esta componente tal como presente na secção 4.1.1 engloba na sua estrutura a API desenvolvida em .NET, o servidor de Base de Dados Microsoft SQL Server e o código de *frontend* desenvolvido em *Vuetify*.

Inicialmente teria sido concebido a ideia de fornecer um serviço global em que o cliente poderia registar a empresa dentro da aplicação e assim seria uma plataforma única disponibilizada globalmente para qualquer interveniente. No entanto, após alguma consideração, a solução passa por ser uma vertente de uma ferramenta industrial interna que deverá ser alojada nos servidores internos da empresa/cliente reduzindo a complexidade de distribuição e permitindo focar no desenvolvimento.

De forma a conseguirmos alcançar os resultados desejados foi necessário pensar na estrutura e relação entre objetos necessário para interligar os diferentes componentes e informações que pretendíamos disponibilizar para os utilizadores.

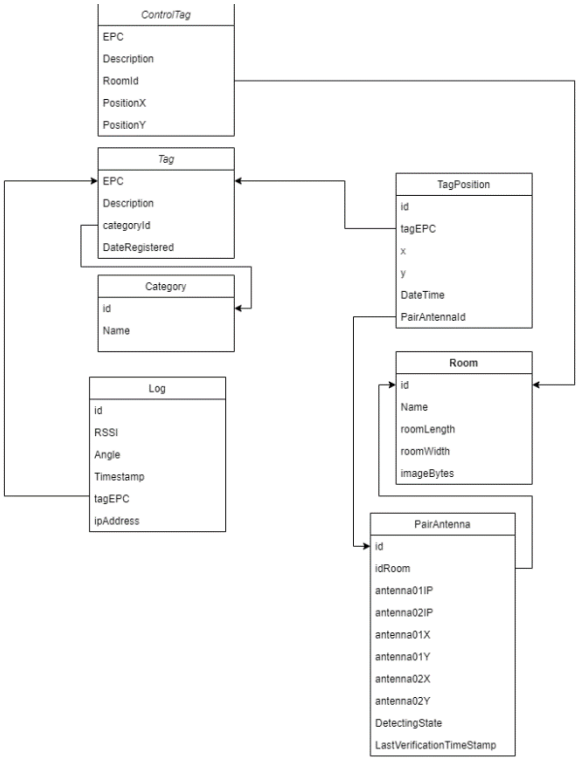


Figura 30 - Relações entre os objetos criados para replicar o problema

A Figura 30 apresenta o diagrama de relações entre os objetos e as suas variáveis. Esta foram as relações desenvolvidas na Base de Dados em que cada objeto presente no diagrama corresponde a uma tabela da base de dados.

Tal como é possível observar no centro das relações encontramos o objeto *Tag*. Através deste obtemos as relações para o Log que serão, tal como o objeto criado no OPC-UA, mencionado na secção 4.1.3, uma representação do ângulo obtido da captura da *tag* pela antena. Neste está representado ainda a força do seu sinal de forma a podermos posteriormente calcular e apresentar a posição absoluta da *tag* no referencial no objeto *TagPosition*. Este cálculo só será possível com auxílio à informação disposta nos objetos *PairAntenna* e *Room*, que representam um par de antenas, a sua posição no referencial e o seu estado, e a sala onde este par se encontra explicitando as suas dimensões, respetivamente.

Após a conceptualização dos objetos e relações necessárias o próximo passo foi naturalmente a criação dos *endpoints* e lógica de negócio.

5.3.1. Jobs

Nesta secção serão abordados os *jobs* ou *batchs* que estão responsáveis por automatizar algumas das tarefas e verificações necessárias para cumprir com o bom funcionamento do sistema.

Foram desenvolvidos 2 *jobs* em conjunto com a API na *framework* .NET e que faz uso da mesma para atualizar e criar registos. Estes são:

- *AntenaDetectionVerificationJob* – responsável por através do conhecimento da posição absoluta das *tags* de controlo, calcular e verificar se o posicionamento/deteção das antenas está a ser corretamente efetuado, com uma margem de erro de 2 metros em ambos os eixos.

  Após a verificação o estado da antena (campo *DetectingState*) será devidamente atualizado com os valores (0 – Nenhuma *tag* de controlo encontrada para verificar; 1 – Deteção errada da posição da *tag* de controlo; 2 – *Tag* de controlo detetada e posicionada corretamente) sendo que no seu estado inicial ao ser registada uma antena no sistema tem sempre o estado com o valor -1 de forma a indicar que ainda não foi utilizada pelo *job*. A tarefa é executada de 5 em 5 minutos para contrariar o caso em que esta sofra alguma deslocação e comece a registar *logs* de forma errada;

- *TagPositionCalculationJob* – sendo um dos requisitos apresentar a última posição das *tag*, foi desenvolvido este *job* que tem o trabalho de para cada *tag* que se encontre presente no sistema verificar os *logs* registados nos últimos 10 minutos para o último par de antenas que detetou a *tag* e calcular o seu posicionamento com base neste registo.

  Com a posição calculada, será inserido assim um novo registo na tabela *TagPosition* de forma a termos um registo da deslocação da *tag* e de onde se encontra atualmente. Esta funcionalidade está a correr em intervalos de 1 minuto;

6. Testes de Sistema

Neste capítulo serão demonstrados os resultados obtidos nos testes feitos à solução desenvolvida. Embora a maioria do código tenha sido testada através de testes unitários, neste capítulo serão apenas abordados os testes de integração que permitem validar a qualidade do projeto desenvolvido. Ao longo deste serão apresentados os cenários de testes efetuados, demonstrando a posição das antenas, a posição real das *tags* e os resultados obtidos pelo algoritmo criado mencionado no capítulo 3.

6.1. Cenário 1

No primeiro cenário foi reservado uma das salas no instituto onde foram posicionadas as *tags* e um conjunto de pares de antenas de forma a conseguirmos validar a solução com base em dados reais. O posicionamento do cenário está presente na Figura 31 onde o grupo de *tags* presente na imagem corresponde a uma tira com 37 *tags* e na Tabela 3 estão detalhadas as posições relativas às antenas, grupo de *tags* e até os valores máximos dos eixos da sala.

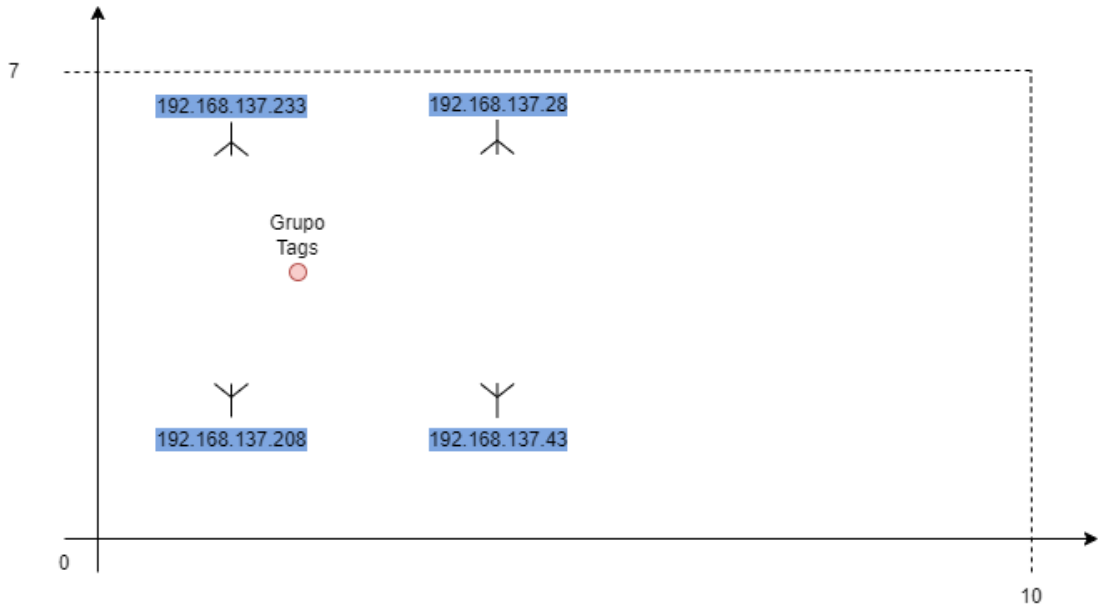


Figura 31 - Disposição das antenas e tags no cenário 1

Tabela 3 - Informação do primeiro cenário

		x	y
Grupo de Tags		3	4
Par de Antenas 1	192.168.137.43	6	2
	192.168.137.208	2	2
Par de Antenas 2	192.168.137.28	6	6
	192.168.137.233	2	6
Máximo da divisão		10	7

Devido à segurança da rede no edifício, neste primeiro teste não nos foi possibilitado enviar diretamente os resultados obtidos pelas antenas para o nosso serviço. Por isso foi necessário nesta primeira fase recolher os *logs* das antenas que estavam a ser enviados para um servidor externo e através de um código auxiliar desenvolvido em *python* enviar da nossa máquina local esta mesma informação para o Raspberry Pi, que efetuava o tratamento dos dados e posteriormente enviava os *logs* para o nosso serviço. Desta forma simulando o processo verdadeiro e contornando as limitações impostas pela segurança da rede da instituição.

Tabela 4 – Últimas posições das tags obtidas no primeiro cenário

TagEPC	x	y	angleAntenna01	angleAntenna02
E2001931325715707853	≈ 4.079	7.0	120.0	58.0
E2001931324715707842	≈ 4.721	7.0	108.86	54.0
E2001931322515707813	≈ 3.222	7.0	135.33	66.0
E20019313195157077D4	≈ 3.425	7.0	123.33	70.0
E20019313194157077CC	10.0	7.0	40.0	50.0
E20019313193157077D3	≈ 3.474	7.0	127.33	66.0
E20019313187157077C4	≈ 3.387	7.0	130.0	66.0
E20019313175157077B2	≈ 4.943	7.0	110.4	44.0

E20019313174157077AA	≈ 4.283	0.0	134.0	54.0
E200193131731570771B1	≈ 5.278	7.0	102.0	46.0
E2001931316115707793	≈ 4.854	0.0	142.0	62.8
E200193131601570778B	≈ 4.75	0.0	128.67	70.0
E2001931315915707792	≈ 4.531	0.0	140.0	55.33
E2001931315315707783	≈ 6.187	7.0	88.86	66.0
E200193131521570777B	≈ 4.073	0.0	128.0	54.0
E200193131421570776A	≈ 3.763	0.0	126.0	47.33
E2001931314115707771	10.0	7.0	70.0	66.0
E2001931314015707769	≈ 4.07	7.0	132.0	46.0
E200193131261570774A	≈ 4.247	7.0	116.0	58.0
E2001931312515707751	≈ 6.681	7.0	82.0	46.0
E20019313124158075 1D	≈ 3.493	7.0	140.67	54.0
E2001931312415707749	≈ 4.774	7.0	119.5	38.0
E2001931312315707744	≈ 3.258	0.0	106.0	58.0
E200193131181570773A	≈ 4.826	0.0	142.0	62.0
E20019313090157076FC	≈ 4.07	0.0	134.0	48.0

Os resultados obtidos neste cenário estão representados pelas últimas posições encontradas para cada *tag* apresentados na Tabela 4. Onde podemos verificar que em nenhum das deteções as *tags* apresentaram valores sequer aproximados da posição real dentro da divisão e muitas das vezes com o posicionamento da *tag* atrás da orientação das antenas.

Ao analisarmos os resultados foi possível identificar que o cálculo da interseção com os ângulos fornecidos estava a ser corretamente efetuados e que o problema estava relacionado com o ângulo obtido pelas antenas. Pegando por exemplo nos valores para a *tag* com o identificador E2001931325715707853. Podemos verificar que a antena 1 (considerada a antena mais à esquerda) registou a *tag* como tendo sido captada com uma

amplitude de 120º enquanto a antena 2 (considerada mais à direita) apresentou um valor de 58º. Estes valores fazem com que o ponto de interseção entre as retas formadas pelos ângulos das antenas se encontre atrás das próprias antenas. Ao visualizarmos os *logs* gerados pelas antenas foi possível reparar que na maioria das captações os ângulos registados pelas antenas não correspondiam à verdade. Este cenário ocorreu devido ao identificado problema de *multipath* em que o sinal das antenas pode sofrer refração nas superfícies em volta das mesmas e assim apresentarem ângulos completamente errados, como verificado neste cenário.

6.2. Cenário 02

O segundo cenário o teste foi executado remotamente pelo orientador deste projeto devido às dificuldades de conciliação entre os horários. Assim o docente montou um cenário dentro da sua habitação e disponibilizou os *logs* na sua base de dados de forma a poderem ser recolhidos os mesmos e validar a posição e captação das *tags* no sistema.

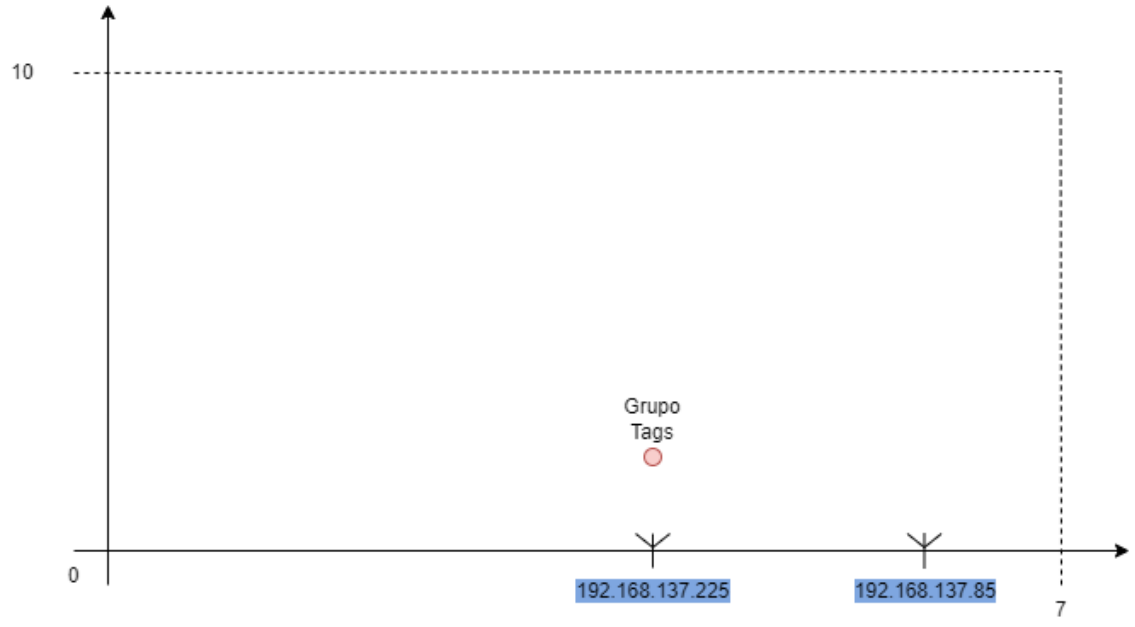


Figura 32 - Disposição das antenas e tags no cenário 2

Tabela 5 - Informação do segundo cenário

		x	y
Grupo de Tags		4	1.3
Par de Antenas 1	192.168.137.225	4	0
	192.168.137.85	6	0
Máximo da divisão		7	10

A grande diferença deste cenário para o cenário anterior foi, além da disposição das antenas e das *tags*, a redução da força do sinal das antenas de modo a prevenir *multipath*, que como mencionado no cenário anterior foi identificado como o principal responsável para os valores errados obtidos. De forma a corrigir a situação, a força do sinal foi enfraquecida o que diminui o alcance das antenas, mas aumenta a sua precisão na deteção do ângulo de captação. O ângulo de deteção da antena também foi invertido por decisão do orientador, proprietário da antena que assim passou a detetar o ângulo como apresentado na Figura 33, invertendo o que foi mencionado na secção 4.5.2.

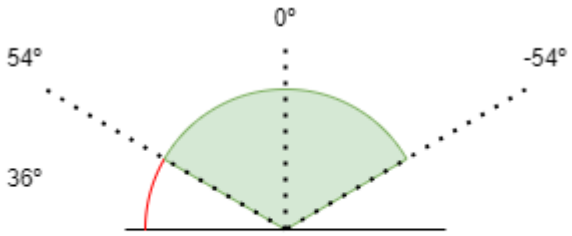


Figura 33 – Inversão do raio de captação do ângulo da tag em relação à antena

6.2.1. Primeira Execução

Na primeira execução deste cenário foram obtidos os valores na tabela abaixo onde podemos verificar que na sua maioria foram obtidos valores semelhantes aos valores reais muitas das vezes apresentando apenas pequenos desvios.

Tabela 6 - Últimas posições das tags obtidas na primeira execução do segundo cenário

TagEPC	x	y	angleAntenna01	angleAntenna02
E20019313090157076FC	≈ 4.321	≈ 1.512	78.0	138.0
E2001931375157077B2	≈ 4.677	≈ 1.191	60.4	138.0
E200193131581570778A	≈ 4.327	≈ 1.733	79.33	134.0
E2001931322515707813	≈ 4.851	≈ 1.662	62.89	124.67
E200193131521570777B	≈ 4.065	≈ 1.694	87.82	138.8
E2001931314015707769	≈ 4.656	≈ 1.05	58.0	142.0
E2001931314115707771	≈ 4.188	≈ 1.222	81.27	146.0
E2001931316115707793	≈ 4.296	≈ 2.53	83.33	124.0
E20019313212157077F9	≈ 4.877	≈ 1.163	53.0	134.0
E2001931312415707749	≈ 4.904	≈ 1.625	60.93	124.0
E2001931325715707853	≈ 4.218	≈ 1.55	82.0	139.0
E200193131261570774A	≈ 4.608	≈ 1.583	69.0	131.33
E20019313188157077C9	≈ 2.040	≈ 4.502	113.52	131.33
E20019313174157077AA	≈ 4.267	≈ 2.136	82.8	129.0
E200193131181570773A	≈ 4.69	≈ 1.023	56.0	142.0
E20019313195157077D4	≈ 4.585	≈ 1.608	70.0	131.33

Na Tabela 6 podemos visualizar as últimas posições calculadas pela solução. Com estes dados, bastante próximos do cenário real, decidimos calcular o erro de deteção. Através de uma consulta rápida à base de dados podemos verificar que existiram 374 *logs* registados no serviço e que o somatório dos valores registados para x e y foram respetivamente 1535 e 684 o que nos permite retirar o seguinte:

$$y_{médio} = \frac{\sum y}{\sum nº\ de\ logs} = \frac{684}{374} \cong 1.834 \text{ m}$$

$$x_{médio} = \frac{\sum x}{\sum nº\ de\ logs} = \frac{1535}{374} \cong 4.115 \text{ m}$$

$$erro = \sqrt{(x - x_{real})^2 + (y - y_{real})^2} \Leftrightarrow$$
$$\Leftrightarrow erro = \sqrt{(4.115 - 4)^2 + (1.834 - 1.3)^2} \Leftrightarrow$$
$$\Leftrightarrow erro \cong \sqrt{0{,}298} \cong 0{,}546 \text{ m}$$

Com estes cálculos podemos verificar que obtivemos um erro de aproximadamente 0,546 metros. Um valor bastante reduzido e que nos permite ter confiança na obtenção da posição das *tags* através do nosso sistema. O próximo passo seria evidentemente tentar reduzir ainda mais este valor.

6.2.2. Segunda Execução

Na segunda tentativa, utilizando o mesmo cenário, foi adicionado uma constante ao mapeamento do ângulo da antena efetuado no Raspberry Pi. Com base na execução anterior podemos verificar que o ângulo de captação das antenas, embora tenha melhorado bastante em relação ao primeiro cenário, continuava diferente do desejado. Assim para encontrar esta constante foi realizado o seguinte exercício:

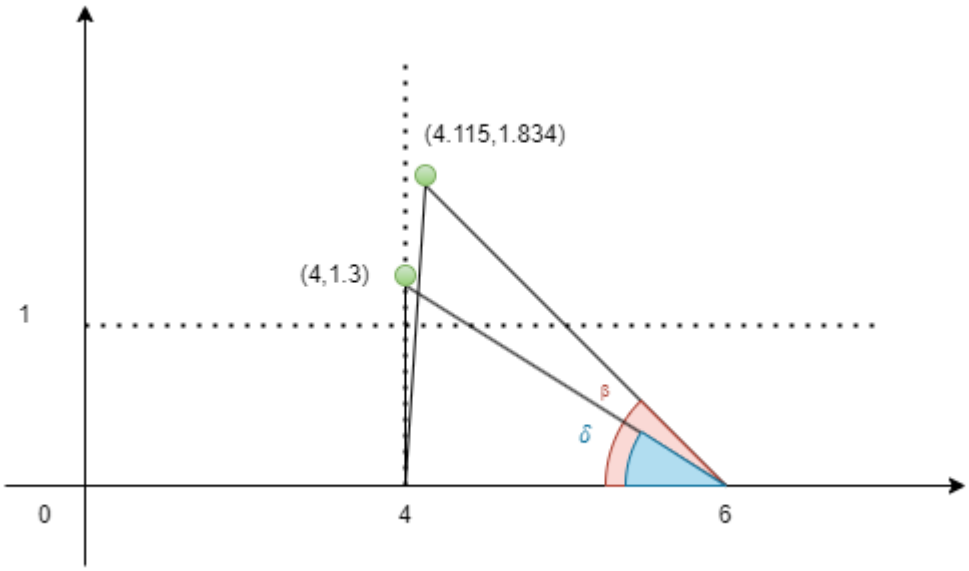


Figura 34 - Ilustração auxiliar do cálculo para a correção do erro

$$\tan\delta = \frac{1.3}{6-4} \cong 0.65$$

$$\delta = \tan^{-1}(0.65) \cong 33\ °$$

$$\tan\beta = \frac{1.834}{6-4.115} \cong 0.989$$

$$\beta = \tan^{-1}(0.989) \cong 45°$$

Constante de correção = 45-33 = 12 º

Com este cálculo encontrámos a diferente entre o valor do ângulo de captação da antena esperado e o real e podemos utilizar este valor como uma constante a reduzir ao ângulo enviado pelas antenas.

Tabela 7 - Últimas posições das tags obtidas na segunda execução do segundo cenário

TagEPC	x	y	angleAntenna01	angleAntenna02
E2001931324715707842	≈ 4.512	≈ 1.004	63.0	146.0
E2001931319315707D3	≈ 4.321	≈ 1.512	78.0	138.0
E2001931316115707793	≈ 4.078	≈ 1.11	86.0	150.0
E200193131421570776A	≈ 4.3	≈ 1.643	79.71	136.0
E2001931314115707771	≈ 4.245	≈ 0.9	74.0	154.0
E2001931317415707AA	≈ 4.284	≈ 0.991	74.0	150.0
E2001931318815707C9	≈ 4.61	≈ 1.251	64.0	138.0
E2001931317615707AB	≈ 4.418	≈ 1.9	77.5	130.0
E2001931313915707764	≈ 4.010	≈ 1.342	89.56	146.0
E2001931315915707792	≈ 4.190	≈ 0.962	78.8	152.0
E2001931314015707769	≈ 4.19	≈ 0.731	75.5	158.0
E200193131521570777B	≈ 4.19	≈ 0.883	78.0	154.0
E2001931319415707CC	≈ 3.710	≈ 2.062	98.0	138.0
E200193131181570773A	≈ 4.27	≈ 1.2	77.0	146.0
E200193131581570778A	≈ 3.903	≈ 1.211	94.57	150.0
E2001931322515707813	≈ 4.19	≈ 0.883	78.0	154.0
E2001931325715707853	≈ 3.9	≈ 1.44	95.33	146.0
E2001931309015707 6FC	≈ 4.13	≈ 1.1	83.14	150.0
E2001931315315707783	≈ 4.5	≈ 0.813	60.0	152.0
E2001931312415707749	≈ 4.607	≈ 0.972	58.0	145.08
E200193132121570 77F9	≈ 4.71	≈ 0.9	50.0	146.8
E2001931317515707 7B2	≈ 4.452	≈ 1.244	70.0	141.2
E200193131261570774A	≈ 4.75	≈ 1.027	54.0	140.67
E2001931319515707 7D4	≈ 4.621	≈ 0.994	58.0	144.22

Com estes dados, bastante próximos do cenário real, foi calculado o erro de deteção:

$$y = \frac{\sum y}{\sum nº\ de\ logs} = \frac{3435}{2646} \cong 1.298 \text{ m}$$

$$x = \frac{\sum x}{\sum nº\ de\ logs} = \frac{10582}{2646} \cong 4 \text{ m}$$

$$erro = \sqrt{(x - x_{real})^2 + (y - y_{real})^2} \Leftrightarrow$$
$$\Leftrightarrow erro = \sqrt{(4 - 4)^2 + (1.298 - 1.3)^2} \Leftrightarrow$$
$$\Leftrightarrow erro = \sqrt{0{,}000004} = 0.002 \text{ m}$$

Os cenários e as execuções efetuadas serviram para podermos validar e fazer algumas correções ao sistema de forma a melhorar a deteção de *tags* num ambiente controlado e conhecido. Tendo o erro diminuído a um valor tão mínimo demos os testes como concluídos. Sendo que ao longo destes testes foi verificada toda a execução do sistema, desde a simulação da captação dos *logs* até ao cálculo efetuado pelo cenário e ainda a visualização das posições das *tags* com recurso à aplicação *web*.

7. Instruções de Utilização da Solução

Neste capítulo será feito o acompanhamento ao processo de instalação e montagem do sistema num ambiente de produção real. Analisando a solução criada foi possível criar uma ordem que facilite a utilização do mesmo que deverá ser cumprida devido à dependência entre certos passos e tecnologias, sendo esta a seguinte:

1. *Deploy* do sistema
2. Instalação do Raspberry PI
3. Registo das divisões do edifico/fábrica
4. Instalação das antenas
5. Registo dos objetos com recurso às *tags*
6. Verificação das posições das antenas
7. Detetar as *tags*
8. Visualização da posição absoluta das *tags*

Nas seguintes secções serão explicados os passos e apresentadas algumas imagens exemplificativas referentes à plataforma e à montagem.

7.1. *Deploy* do sistema

O primeiro passo na implementação do sistema passa por efetuar o *deploy* da plataforma nos servidores internos da empresa. Como foi mencionado no capítulo sobre a arquitetura do sistema, esta plataforma deverá estar alojada num servidor ou sistema interno da empresa de forma a proteger a informação e facilitar a comunicação com os restantes módulos.

Quando este é efetuado, por *default,* será criado uma conta *admin* que permite aceder e ter total acesso às ferramentas da plataforma. Apenas após este será possível efetuar o registo de todos os outros componentes do sistema e outros possíveis utilizadores. Assim caso seja requerido o administrador do sistema poderá registar outro utilizador na plataforma de forma a garantir o acesso.

7.2. Instalação do Raspberry PI

De forma a criar a comunicação entre as antenas e as plataformas, tanto na plataforma online como no servidor OPC-UA, respeitando o que foi descrito no capítulo 5, é necessário instalar o módulo 1B, ou seja, o Raspberry PI. Assim o procedimento será simplesmente importar o código destinado, e importar os módulos Python necessários para o seu funcionamento num Raspberry PI com conexão à rede da empresa de forma a conseguir comunicar com a plataforma online. Após o código estar presente na máquina e os módulos *python* estarem devidamente instalados, para iniciar os servidores *REST* e OPC-UA acima executamos o seguinte comando na linha de comandos do Raspberry Pi:

uvicorn http_opc_server:app –reload –host 0.0.0.0

Após iniciar o software, a API e o servidor OPC-UA estarão disponíveis para serem utilizados pelas antenas e consultados pelo cliente assim que o terminal apresente a informação como demonstrado na Figura 35.

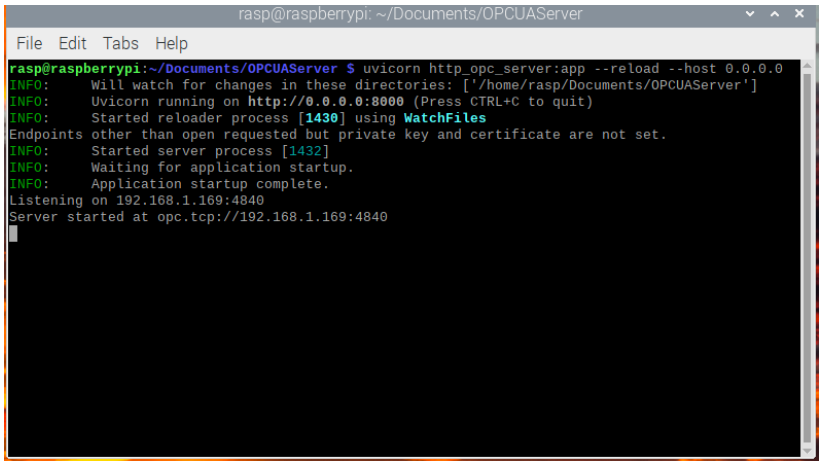


Figura 35 - Inicialização dos servidores REST e OPC UA no Raspberry Pi

7.3. Registo das divisões do edifico/fábrica

Com o registo na plataforma efetuado o próximo passo seria a identificação e registo das divisões onde serão utilizadas as *tags* e antenas. Para isso e em conformidade com o que está presente na figura será necessário obter as medidas das divisões em metros. Sem este componente não será possível registar os pares de antenas pois será nesta página da plataforma que posteriormente iremos associar os pares de antenas à divisão.

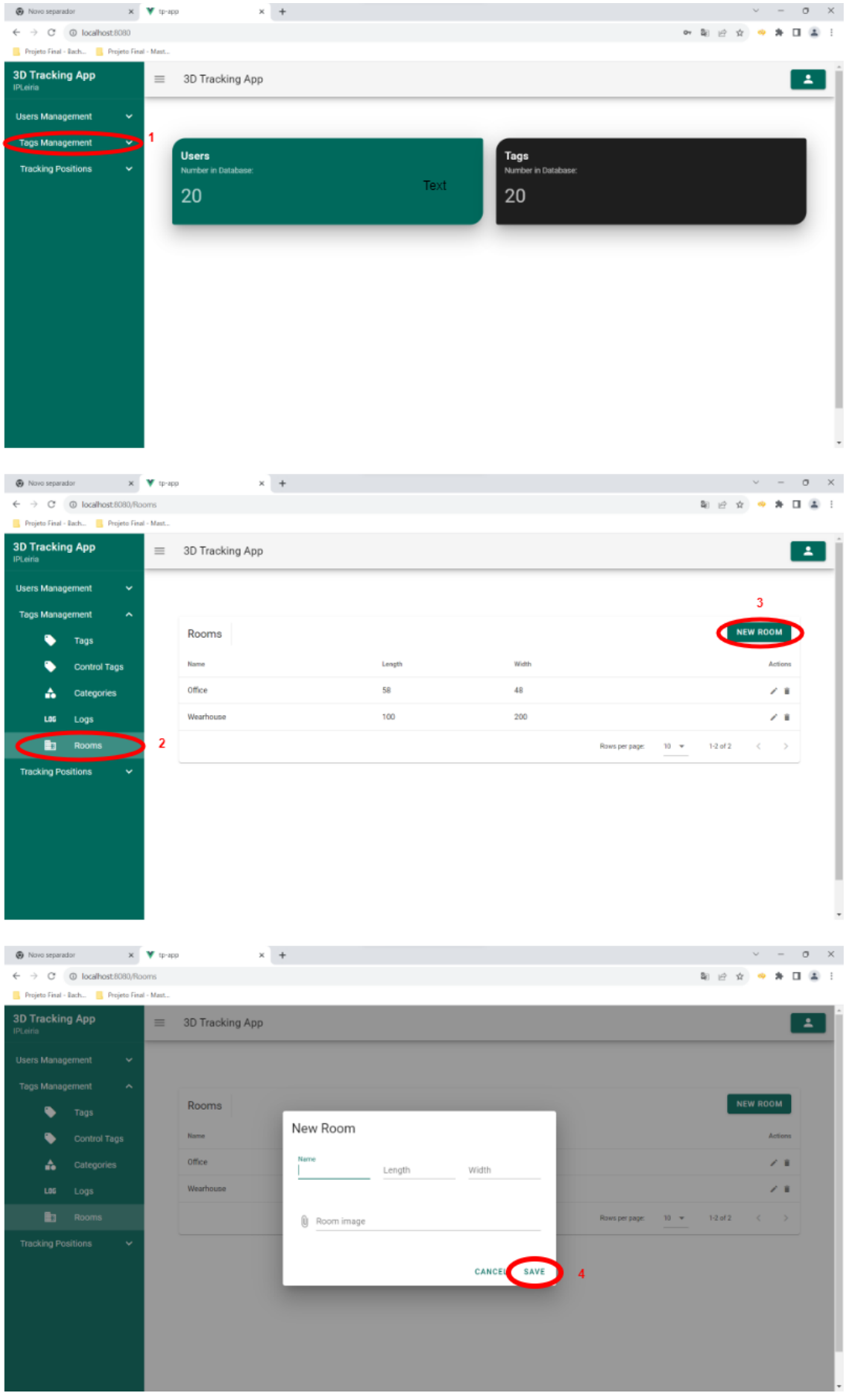


Figura 36 - Instruções para o registo das divisões/salas no sistema

7.4. Instalação das Antenas

Tal como havia sido referido em capítulos anteriores o posicionamento e instalação das antenas no interior da divisão/sala onde serão utilizadas para a deteção das *tags* e consequentemente dos objetos identificados pelas mesmas é da máxima importância para o bom funcionamento do sistema na sua globalidade e para atingir os resultados pretendidos.

Assim e tal como instruído as antenas deverão ser posicionadas paralelamente uma à outra de forma que a triangulação mencionada no capítulo 4 seja o mais real possível e que o cálculo efetuado para a localização da *tag* possa corresponder à verdade.

Deverão ainda ser colocadas a uma distância não superior a 5 metros devido ao seu alcance. Será, portanto, esperado por parte do cliente ou do responsável pela instalação do sistema que esta componente cumpra os requisitos mencionados.

Após o devido posicionamento das mesmas, estas deverão ser introduzidas no sistema com o *IPAddress* de cada ESP8266 como identificador e deverá ser indicada a posição de cada uma em relação à divisão. Aconselha-se a obter as medidas da divisão de forma correta e considerar sempre o mesmo ponto de origem para cada antena disposta na mesma.

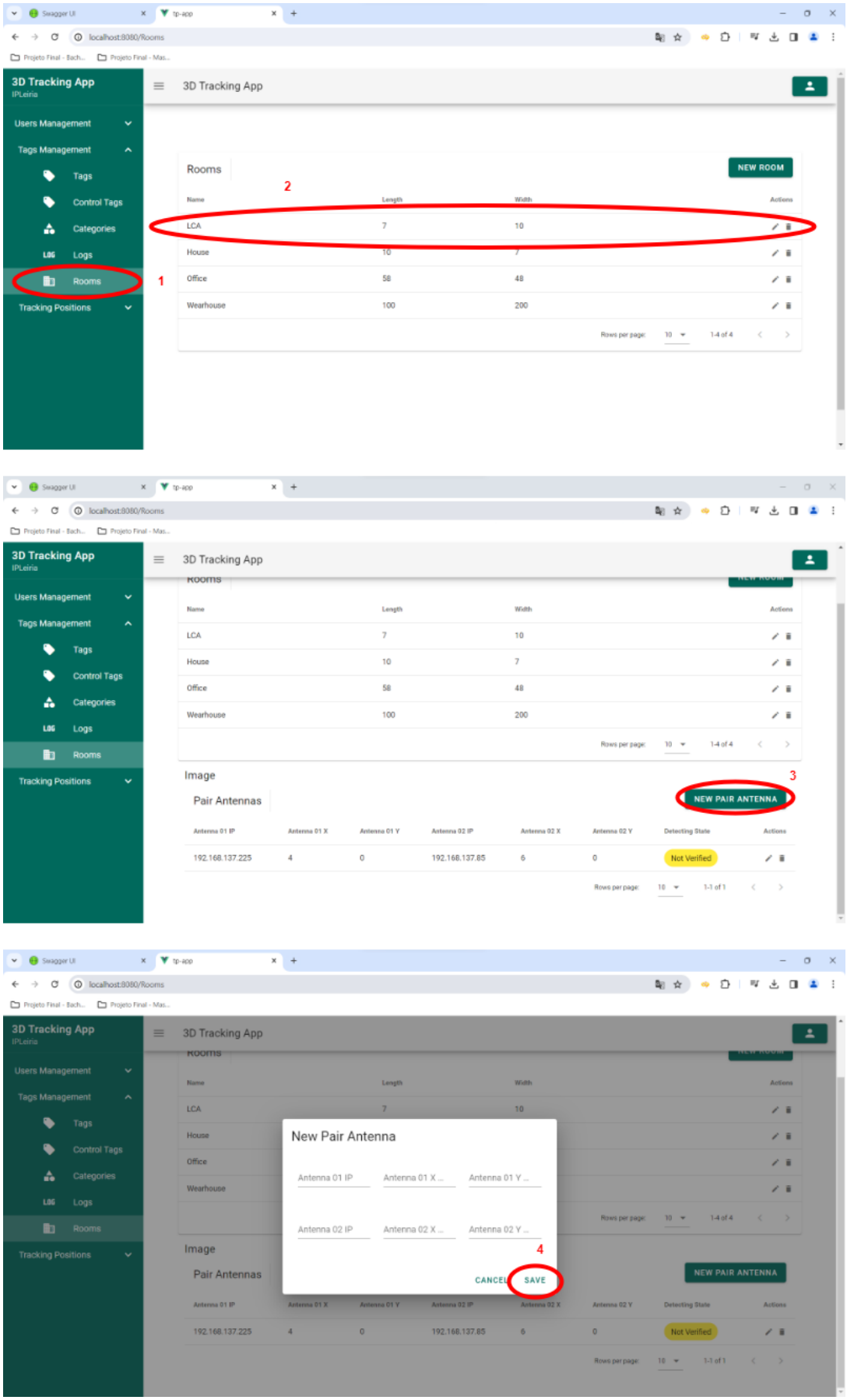


Figura 37 - Instruções para o registo de um par de antenas

7.5. Registo dos objetos com recurso às *tags*

De forma a alcançar o principal objetivo, ou seja, o rastreamento da localização dos objetos, dentro da zona de fabricação ou outra, é fundamental registar a *tag* que é desejada para a deteção e identificação da mesma.

Assim, após ser colocada a *tag* junto ou no interior do objeto de forma a identificar o mesmo, esta deverá ser registada no *website*. Caso a mesma não seja introduzida no sistema, embora a *tag* possa ser capturada no leitor RFID das antenas e apresentado o seu *log* no cliente OPC-UA este *log* não será contabilizado no sistema e não será possível consultar o posicionamento da mesma pois este não será calculado.

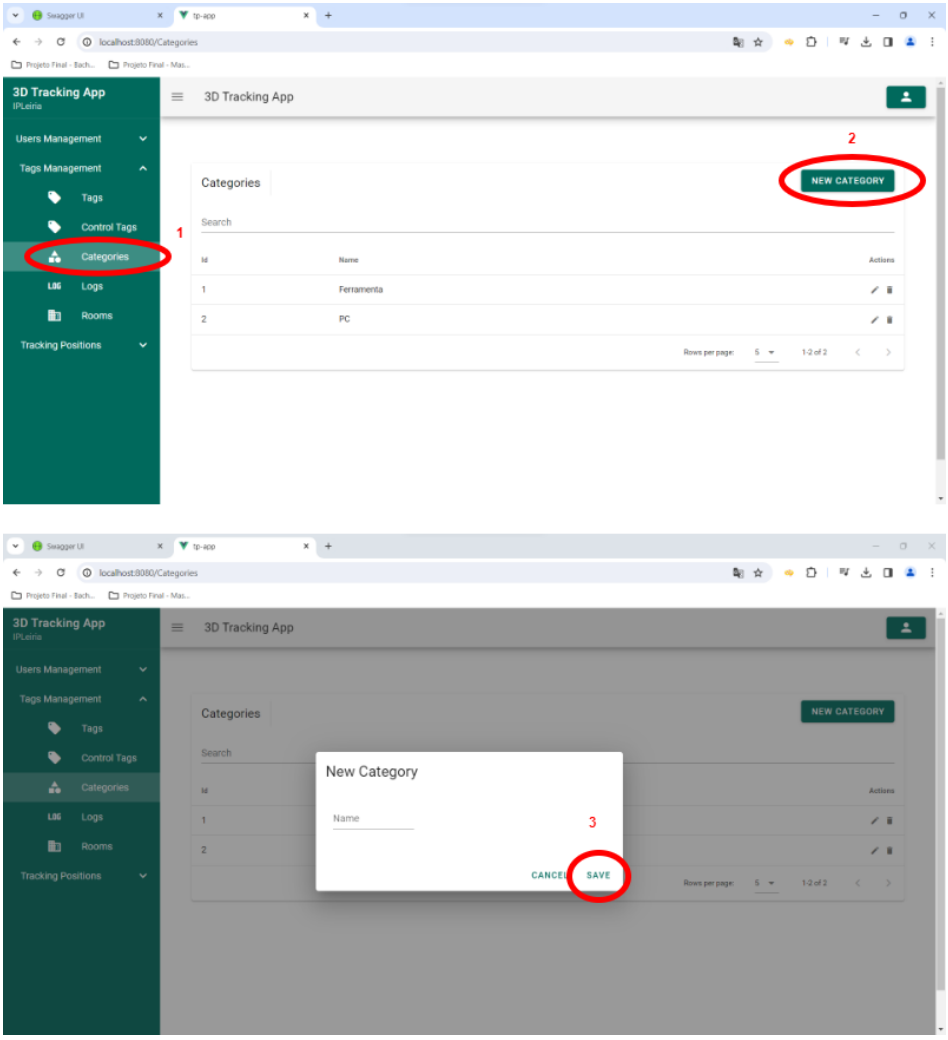


Figura 38 - Instruções para a criação de uma categoria

Antes de registar, para ser mais fácil identificar o que é ou a que classificação pertence a *tag*, poderá ser criada uma categoria, tal como apresentado na Figura 38.

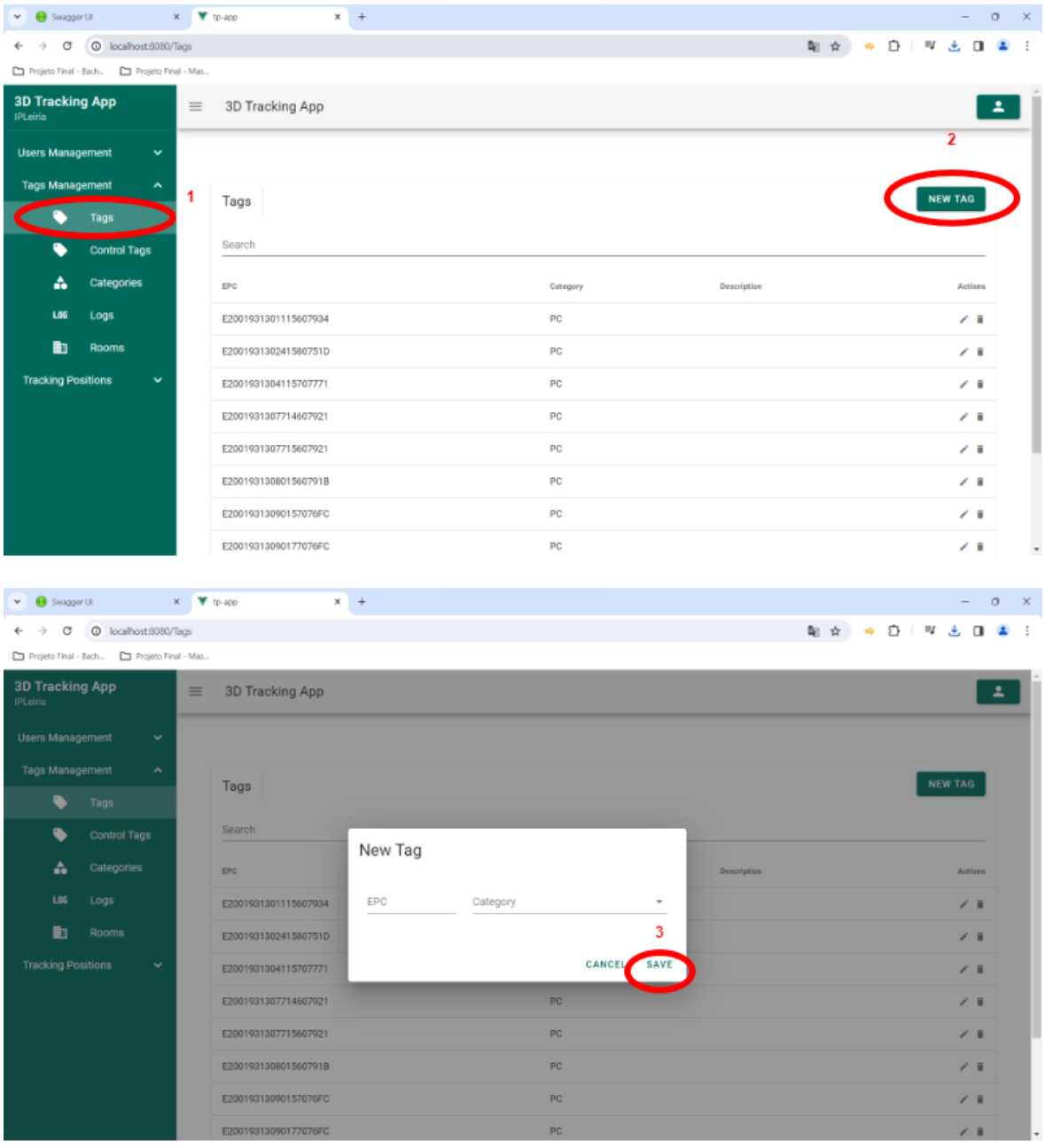


Figura 39 - Instruções do registo de uma tag

Após a criação da categoria a *tag* é bastante fácil de registar, como demonstrado na Figura 39, sendo apenas necessário guardar o seu identificador (EPC), escolher a categoria criada ou outra e posteriormente guardar a informação no sistema.

7.6. Verificação das posições das antenas

Tal como foi mencionado no capítulo sobre a implementação, foi desenvolvido um *job* que tem como objetivo verificar que a deteção efetuada pelas antenas esteja correta. Assim caso as antenas sofram alguma alteração de posição não registada ou por alguma razão seja alterado o ângulo de captação, o job está encarregue de verificar se as deteções das *tags* de controlo estejam corretas dentro de uma variação de dois metros.

A informação sobre o estado do par de antenas será disponibilizada na plataforma de forma a poder ser reconhecida a irregularidade e corrigida como presente na Figura 40.

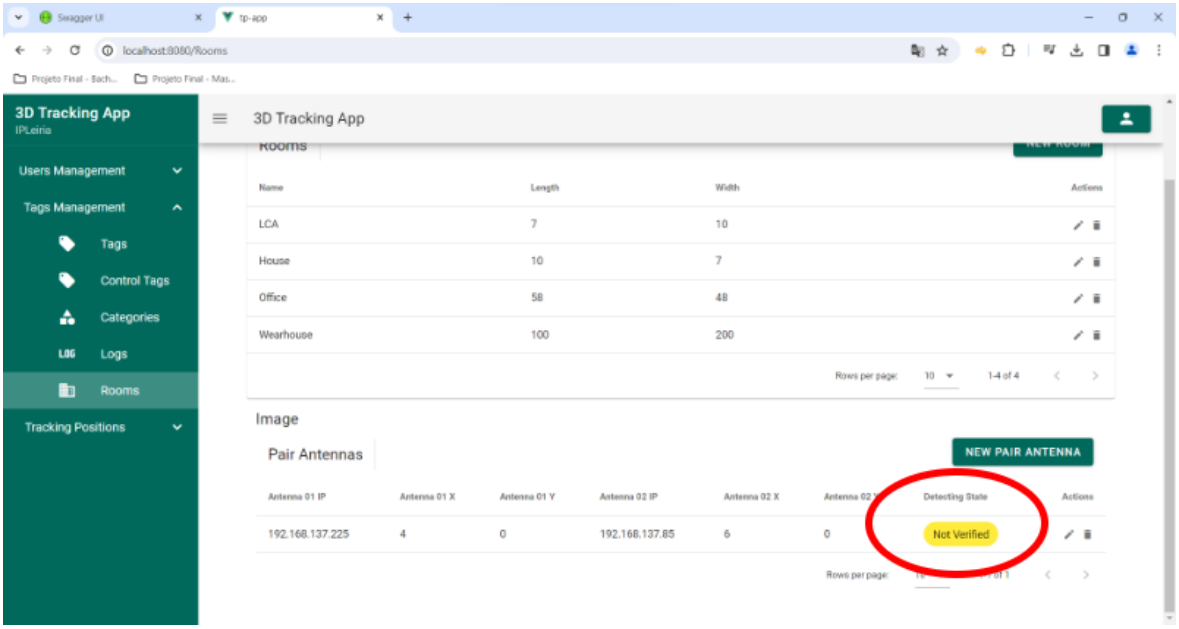


Figura 40 - Instruções da visualização do estado de captação da antena

7.7. Detetar as *tags*

Com as etapas anteriores concluídas as *tags* serão detetadas pela antena e serão registadas no OPC-UA como mencionado anteriormente. No entanto não serão registadas na plataforma *web* a menos que sejam registadas no sistema. Este funcionamento serve para garantir que apenas as *tags* desejadas estejam acessíveis e que tenhamos assim a informação correspondente às mesmas. Sem este processo, caso a *tag* não esteja registada irá acumular diversos *logs* sem termos nenhuma informação sobre a que objeto estes estão associados.

Quando a *tag* estiver registada assim que seja captada pela antena será criado um *log* da captação também na plataforma e assim poderá ser acedido.

7.8. Visualização da posição absoluta das *tags*

Por último e talvez mais importante será encontrar o objeto, saber a sua posição absoluta e em que sala este se encontra. Assim e de forma a ser possível visualizar essa informa o utilizador deverá utilizar a opção *Tracking Positions* > 2DPosition e tal como presente na figura indicar o ID da *tag* que deseja encontrar. Caso não tenha o conhecimento exato do ID da *tag*, tal como apresentado anteriormente, poderá encontrar o mesmo na opção *Tags Management* > *Tags*. Após confirmar o que deseja encontrar basta selecionar a mesma na lista de *tags* disponíveis e carregar no botão submeter. Desta forma e caso seja encontrada a *tag* será apresentada num gráfico cartesiano a posição referente à posição absoluta da *tag* e ainda a divisão em que esta foi encontrado como apresentado na Figura 41.

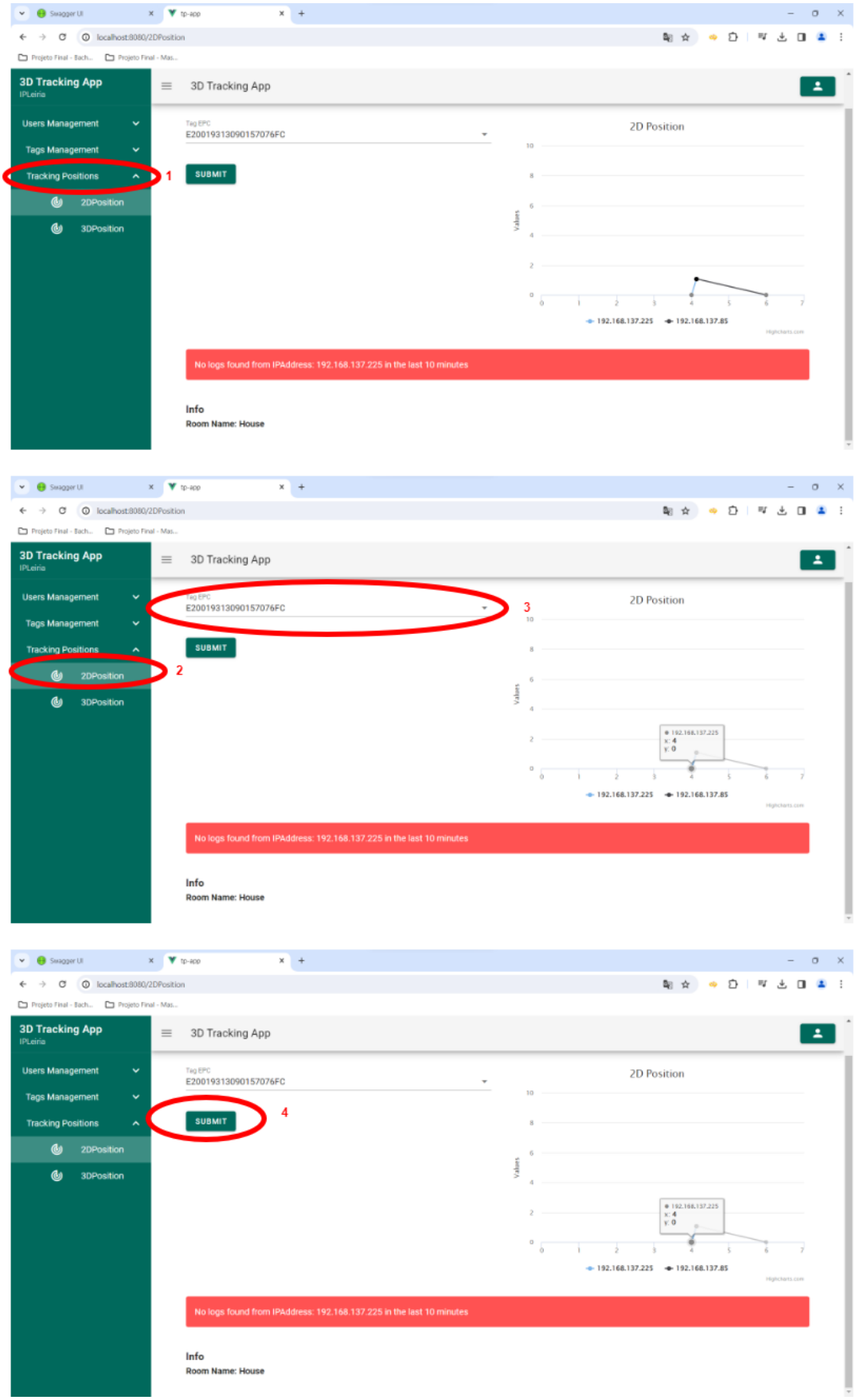


Figura 41 - Instruções da visualização da posição de uma tag

A Figura 41 apresenta a informação sobre a posição absoluta em relação ao ponto de origem da divisão (considerado aquando registo do par de antenas), em que sala esta foi identificada e ainda um pequeno alerta, caso seja necessário, se esta *tag* não foi capturada por nenhuma antena nos últimos 10 minutos.

8. Conclusão

Este capítulo serve como término e apreciação ao trabalho realizado durante a elaboração deste projeto.

Tal como foi mencionado no primeiro capítulo, em que foi exposto o problema presente no dia a dia das empresas e também apresentado o principal objetivo, sendo este a elaboração de uma aplicação que em conjunto com as tecnologias de RFID fornecessem a possibilidade às empresas de obter maior controlo sobre os seus equipamentos e posses, podemos dizer que tal aplicação foi devidamente desenvolvida e testada fornecendo assim as funcionalidades pensadas para este projeto.

Para atingirmos a solução pretendida era necessário ser efetuado em primeiro lugar um estudo sobre as tecnologias que não me eram familiares, nomeadamente a tecnologia RFID e OPC-UA. Assim o capítulo 2 apresenta o estado destas duas tecnologias ao dia de hoje e faz um breve resumo do que foi o entendimento sobre o assunto. Foi possível, graças a esse estudo, ter o conceito base do posicionamento e de que forma seriam utilizadas estas tecnologias dentro do projeto desenvolvido.

Posteriormente ao estudo mencionado foi elaborado um plano de execução para o decorrer do projeto, em que foram identificados os principais desenvolvimentos e as suas tecnologias. Assim no capítulo 3 estão identificados não só quais os softwares como os hardwares que seriam necessários. Foi então apresentado ao leitor as tecnologias utilizadas para criar as APIs utilizadas na solução, o tipo de base de dados utilizado, os componentes que formam a ligação entre as *tags* e o software que permitem visualizar os dados obtidos.

Com o plano de desenvolvimento definido, uma das principais preocupações identificadas logo desde o início estava relacionado com a deteção da posição absoluta das *tags*. Tal como mencionado no capítulo 4 foi explorado diversas situações com o objetivo de facilitar a deteção limitando o posicionamento das antenas a uma única parede da divisão, que foi prontamente descartada pois era impraticável para divisões maiores em que se poderiam encontrar *tags* fora do alcance do sinal das antenas. Assim e como referido no capítulo foi pensado uma solução em que se pudesse posicionar qualquer par de antenas ao longo de toda a divisão com o mínimo de restrições sendo que estas se encontrem paralelas e a uma distância máxima, e foi ainda presente no mesmo capítulo o

algoritmo que possibilitou esta solução sem que fosse necessário o registo de qual das antenas seria a que se encontrasse mais à esquerda ou direita do referencial relativo.

Após as demonstrações que apoiam o algoritmo encontrado é apresentado, no capítulo 5, a arquitetura da solução desenvolvida. Neste capítulo o projeto é repartido em diversos módulos de forma a podermos revelar e explicar a sua função e funcionalidade no todo. Desde a deteção das *tags* e registo no servidor OPC-UA e na API até a visualização dos dados por parte dos clientes temos nesta parte do relatório cada software e tecnologia utilizada, mencionada no capítulo 3, referido e explicado o seu uso.

No sexto capítulo estão apresentados os cenários e testes executados de forma a podermos avaliar e sistema encontrado. Neste encontramos alguns problemas, mais especificamente o caso de *multipath* que distorceu por completo os resultados do primeiro cenário, mas que foi corrigido no segundo onde foi obtido um erro de 0.546 metros. Este resultado, embora bastante animador não correspondia ao que pretendíamos e fomos atrás de melhorar o mesmo. Assim no cenário seguinte fomos corrigir o ângulo de captação das antenas programaticamente. Através do cenário encontrado e os resultados obtidos foi possível verificar a diferença entre o ângulo esperado e o ângulo médio calculado. Esta diferença foi então utilizada na execução do cenário seguinte sendo adicionada programaticamente ao ângulo enviado pelas antenas. Nesta segunda execução obtivemos um erro bastante melhor, sendo este de 2 milímetros. Com este valor demos como concluído os testes à solução devido ao sucesso dos mesmos.

No capítulo 7 podemos encontrar os passos de instalação necessários para o bom funcionamento da solução. Com a apresentação e explicação da necessidade dos passos para que a instalação seja um sucesso estão acompanhadas de ilustrações das plataformas de modo a visualizar onde deverão ser introduzidos os dados.

Podemos verificar que, embora exista ainda uma margem grande de progressão ao que pode ser desenvolvido dentro deste conceito e tema de disponibilizar a posição dos objetos dentro de uma determinada área registada, podemos afirmar que o objetivo foi concluído com sucesso dentro das condições fornecidas.

Bibliografia

X. Jia, Q. Feng, T. Fan e Q. Lei, "RFID technology and its applications in
1] Internet of Things (IoT)," 2012.

B. C. A. S. T. Melanie R. Rieback, "The Evolution of RFID Security," 2006.
2]

A. O. o. P. RFID, "Vipul Chawla and Dong Sam Ha, Virginia Polytechnic,"
3] 2007.

R. Weinstein, "RFID: A Technical Overview and Its Application to the
4] Enterprise," 2005.

G. Y. T. A. M. J. M. A. I. S. a. A. B. Z. Jun Zhang, "A Review of Passive
5] RFID Tag Antenna-Based Sensors and Systems for Structural Health Monitoring Applications," 29 Janeiro 2017.

Q. Systems, "Active and Passive RFID".
6]

RFID Journal, "How much does an RFID tag cost today?," [Online].
7] Available: https://www.rfidjournal.com/faq/how-much-does-an-rfid-tag-cost-today.

R. v. K. G. B. Matt Ward, "RFID: Frequency, standards, adoption and
8] innovation," 2006.

A. A. Shikha Singh, "Survey on Localization Techniques of RFID for IOT,"
9] *International Journal of Computer Applications,* vol. 137, nº 12, p. 0975 – 8887, 2016.

B. J. Schwarz M.H., "A Survey on OPC and OPC-UA," em *XXIV*
10] *International Conference on Information, Communication and Automation Technologies (ICAT),* 2013.

M. L. D. ,. K. C. Rajeev Kumar Chauhan, "Intelligent SCADA System,"
11] *International Journal on power system optimization and Control,* vol. 2, nº 1, 2010.

Esta página foi intencionalmente deixada em branco

Anexos

Link projeto no GitHub: https://github.com/FilipeLopesF/ProductTracking.git

Esta página foi intencionalmente deixada em branco

Glossário