POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Dissertação de Mestrado em Engenharia Informática – Computação Móvel

Análise da Viabilidade de rede IP com nRF24L01+

Bruno Filipe Costa Horta

Dissertação realizada sob a orientação do Professor Doutor Nuno Alexandre Ribeiro Costa, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, *Setembro* de *2019*

Esta página foi intencionalmente deixada em branco

Dedicatória

Dedico todo o trabalho realizado no âmbito desta investigação a todos os engenheiros e investigadores e aos meus filhos que um dia possam fazer uso da mesma.

Fazendo uso das palavras de um grande inventor.

> "Não creio que haja uma emoção, mais intensa para um inventor do que ver as suas ideias a funcionar. Esta emoção faz-nos esquecer de comer, de dormir, de tudo."
>
> *Nikola Tesla*

Esta página foi intencionalmente deixada em branco

Agradecimentos

Seguem-se os meus verdadeiros e sinceros agradecimentos.

Ao meu orientador, Professor Doutor Nuno Costa, pelo desafio proposto, pela vontade de puxar pelos limites da tecnologia e pela sua disponibilidade, compreensão, dedicação e partilha de conhecimento ao longo de toda a investigação.

Ao Politécnico de Leiria pelos excelentes Professores capazes de dar todo o apoio necessário.

À VOID empresa na qual desenvolvo tecnologia e que desde o início me proporcionou todas as condições para realizar este estudo.

Ao investigador e programador TMRh20 pelo seu enorme trabalho desenvolvido, pela partilha de conhecimento e por todos os emails trocados sempre com respostas fantásticas sobre cada detalhe do seu trabalho.

À minha família, que sempre me apoiou mesmo quando eu não lhes consegui dar toda atenção merecida.

Aos meus amigos que me apoiaram durante todo este percurso, em especial à Vanessa Guapo e ao Nuno Rolo.

À minha fantástica esposa, Rita Oliveira, pela sua paciência e esforço nos momentos em que teve de ser Mãe e Pai.

Por fim aos meus dois filhos Alice e Rafael Horta, por terem sido a minha força e inspiração em tudo o que faço na vida.

Esta página foi intencionalmente deixada em branco

Resumo

Neste estudo é documentado um conjunto de testes que visam verificar a viabilidade da implementação do protocolo IP utilizando um rádio nRF24L01+.

O protocolo IP é a base de comunicação adotada pela Internet e pela grande maioria dos computadores. Com base nisso faz todo o sentido trazer o IP até aos dispositivos com pouco recursos, por exemplo: rede se sensores que já são parte integrante da Internet das coisas, possam enviar e receber dados através do mesmo, possibilitando assim a utilização de outros protocolos mais sofisticados como o MQTT.

O estudo apresentado na dissertação é, de alguma forma, pioneiro já que o levantamento do estado da arte indicou que, dos vários projetos de investigação analisados, que recorrem ao nRF24L01+ para assegurar as comunicações rádio, nenhum deles tentou usar o protocolo da Internet ou sequer avaliar essa possibilidade

De forma a avaliar a viabilidade do uso do protocolo da Internet em dispositivos que incluam o rádio nRF24L01+ no mundo real (e não em ambiente simulado), foram desenvolvidos módulos que incluem um nRF24L01+ ligado a um Arduino Nano, e as bibliotecas de software do autor TMRh20, disponibilizadas sob o formato de código aberto e com licenciamento GPL-2.0.

Os módulos foram alvo de testes de fiabilidade, robustez na comunicação e capacidade máxima de comunicar à distância sem fios e foi implementada uma arquitetura que simula uma implementação real utilizada no contexto da Internet das Coisas, fazendo uso das mais recentes tecnologias como MQTT, Node-RED, InfluxDB e Grafana.

Toda a estrutura serviu para criar um ambiente que gerasse dados intensivamente, dados esses que foram registados temporalmente para que fosse possível obter respostas sobre tempos de latência, falhas ou duplicações no envio de mensagens.

Durante o processo foram analisadas diferentes formas de alimentar o rádio nRF24L01+ e conseguiu-se apurar a melhor forma de tirar o máximo partido do mesmo, garantido assim uma maior estabilidade na comunicação.

Em análise, os testes comprovaram ser possível utilizar IP numa configuração nRF sem perdas de potência, mantendo os níveis de consumo e distâncias equivalentes a uma comunicação sem IP, com as vantagens de ser possível implementar um rede em Malha a enviar e receber mensagens via MQTT, sendo que a única desvantagem encontrada é a dos cerca de 16kb que o *firmware* requer de memoria flash e o consumo de memoria a rondar os 68% no caso do Arduino Uno/Nano

Palavras-chave: nRF24L01+, IP, SLIP, MQTT, *Gateway*, Baixo Consumo, Topologia em Malha, Internet das Coisas

Esta página foi intencionalmente deixada em branco

Abstract

This study documented a set of tests intended to verify the workability of implementing the IP protocol using an nRF24L01 + Radio.

The IP protocol is the communication base adopted by the Internet and the vast majority of computers. Based on this, it makes perfect sense to bring IP to low profile devices, for example, network sensors that are already an integral part of the IoT can send and receive data through it and making it possible to use other more sophisticated protocols like MQTT.

The study presented in the dissertation is somehow pioneering since the state-of-the-art survey indicated that of the various research projects analyzed using nRF24L01 + to ensure radio communications, none of them attempted to use the Internet protocol or even evaluate this possibility.

To evaluate the practicability of using the Internet protocol on devices that include the nRF24L01 + in the real world (not in a simulated environment), modules have been developed that include an nRF24L01 + connected to an Arduino Nano, and the TMRh20 software libraries, available in an open-source format and licensed under GPL-2.0.

The modules were tested for reliability, robustness communication and maximum strength to communicate wirelessly, and an architecture that simulates a real implementation used in the context of the Internet of Things was implemented, proffering use of the latest technologies such as MQTT, Node-RED, InfluxDB, and Grafana.

The whole structure served to create an environment that generated data intensively, this data was recorded temporally so that it was possible to get answers about latency times, failures or duplications in the sending of messages.

Different ways of powering the nRF24L01 + radio was analyzed during the process and the best way to get the most out of it was ensured, thus ensuring greater communication stability.

In review, testing has shown that it is possible to use IP in an nRF configuration without power loss while maintaining power levels and distances equivalent to non-IP communication, with the advantages of being able to use a Mesh Network to send and receive messages via MQTT. The only disadvantage is that the 16kb firmware requires flash memory and the memory consumption is around 68% for Arduino Uno / Nano.

Keywords: nRF24L01, IP, SLIP, MQTT, Gateway, Low Power, Mesh, Internet of Things

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 Acrónimos

A/D – Analog to Digital
ACK – Acknowledgement
ADC – Analog digital converter
AMR – Automatic Meter Reading
ANT – Adaptive Network Topology
BLE – Bluetooth Low Energy
D/A – Digital to Analog
IoT – Internet of Things
IP – Internet Protocol
MCU – Microcontroller Unit
MQTT – Message Queuing Telemetry Transport
PDM – Pulse density modulation
PLL – Phase-lock loop
PPI – Programmable Peripheral Interconnect
PWM – Pulse with modulation
RF – Radio Frequency
SLIP – Serial Line Internet Protocol
SMT– Surface-mount technology
SoC – System on chip
SPI – Serial Peripheral Interface
TCP – Transmission Control Protocol
TH – Through-hole
UART – Universal asynchronous receiver/transmitter
ULP – Ultra Low Power
USB – Universal Serial Bus

Esta página foi intencionalmente deixada em branco

Índice

1 Introdução

A comunicação sem fio é um dos paradigmas no qual vivemos atualmente. Esta tecnologia está presente em grande parte dos dispositivos utilizados no nosso dia a dia, por exemplo num telemóvel, num comando de televisão, no Wi-Fi para aceder à Internet ou qualquer outro que tenha capacidade de receber ou enviar dados sem recorrer a uma ligação física.

Em 2017 a empresa de consultadoria Gartner afirmou existirem 8.4 biliões de dispositivos conectados e que poderá chegar aos 20.4 biliões até 2020. Aqui estamos apenas a falar em dispositivos que utilizam a tecnologia sem fio Wi-Fi a 2.4GHz/5GHz. Este número seria bem maior se contabilizasse os dispositivos que utilizam outras tecnologias, por exemplo ZigBee, nRF entre outras.

A comunicação sem fio tem inúmeras vantagens que podem beneficiar tanto o utilizador final como também entidades que querem inovar e desenvolver novos dispositivos que se enquadrem na computação ubíqua [1].

O grande passo para realizar uma comunicação sem fio já foi dado há 100 anos atrás pelo inventor *Guglielmo Marconi* [2], ao longo dos quais foram realizadas muitas mais inovações na comunicação através de Ondas Eletromagnéticas que utilizam o espaço livre para se propagarem.

1.1 Enquadramento

O acesso à informação em tempo real é hoje uma constante necessidade. Sensores, atuadores e microcontroladores de baixo consumo, capazes de comunicar sem fios, estão a ser produzidos em massa devido à globalização da Internet das Coisas [3], ou *Internet of Things* (IoT).

Hoje em dia constitui ainda um desafio desenvolver um dispositivo capaz de operar em locais onde não existe eletricidade nem Internet fixa [4]. No entanto, já foram realizados inúmeros avanços no se refere a tecnologias de comunicação sem fio e é já possível comunicar a longas distâncias com um impacto energético reduzido [5]. A tendência recai para equipamentos de pequenas dimensões capazes de operar durante longos períodos de tempo, utilizando fontes de energia [6] de baixa capacidade e dimensão.

Apesar da oferta existente no mercado para um dispositivo emissor/recetor de baixo consumo, longo alcance e de custo e dimensões reduzidas, chega ao consumidor/investigador sob a

forma de uma solução fechada e com base num plano de serviços com custos de utilização da infraestrutura durante um período limitado de tempo ou tráfego [7].

É possível reaproveitar os desenvolvimentos efetuados ao longo da última década, contornando as limitações que outrora se pensava estarem no hardware do dispositivo através de *firmware* otimizado para tal.

As tecnologias de comunicação têm surgido nas mais diversas frentes, mas existe um foco comum: obter o máximo de alcance possível com o menor impacto no consumo [8]. Em paralelo temos os protocolos que usam esses canais de comunicação e tornam então possível a transmissão correta da informação. Nesse âmbito, o desenvolvimento tem-se focado em tornar os protocolos mais leves mantendo a versatilidade, fiabilidade e escalabilidade dos mesmos. Este é um aspeto importante uma vez que um bom protocolo deve ajudar na comunicação e não tornar a mesma mais complexa e demorada, caso contrário a janela de tempo do emissor/recetor aumenta e consequentemente leva a consumos desnecessários de energia ou até mesmo à inviabilização da utilização da tecnologia física de comunicação por motivos de legislação ou capacidade de retransmissão.

Fabricantes como a Nordic [9] tentam destacar-se no mercado com dispositivos versáteis, capazes de funcionar com consumos reduzidos de energia e ainda assim obter robustez na comunicação sem fio. Disponibilizam uma vasta gama de produtos que podem operar nas mais diversas frequências permitindo assim implementar as mais variadas tecnologias, como *Bluetooth Low Energy* (BLE), Wi-Fi e *Adaptive Network Topology* (ANT), tudo isto num microcontrolador pequeno e de baixo custo.

O sucesso da Nordic tem vindo a aumentar devido à produção de emissores/recetores com capacidades de operar sem fio a longas distâncias, com alta performance e baixo consumo. O nRF24L01+ é denominado pela comunidade de engenheiros como o Pai de toda a vasta gama rádio Radio Frequency (RF) da Nordic. No entanto, foi perdendo destaque e atualmente a Nordic já não o recomenda para novos projetos. Com características capazes de comunicar a distâncias consideráveis, este permite uma configuração de potência entre 0, -6. -12 ou -18dBm, sensibilidade de receção na ordem dos -82dBm a 2Mb/s ou -94dBm a 250kb/s, tecnologia *ShockBurst* que facilita a tarefa do programador, fazendo com que de todas as falhas de pacotes que necessitem de retransmissão fiquem a cargo do hardware. Resumindo, este rádio que tem um custo médio abaixo de um euro e poderá ser ainda em muitos dos casos uma solução viável no desenvolvimento de dispositivos sem fio.

Quando se considera uma comunicação sem fio é necessário que em algum ponto dessa arquitetura os dados cheguem a um ou mais computadores, para que sejam processados e analisados. Tudo isto é possível à custa de protocolos que funcionem num dispositivo denominado de *Gateway*,

que possui a capacidade de converter uma ou mais tecnologias de comunicação noutra que possa ser entendida num sistema diferente.

Uma *Gateway* pode ser vista como um tradutor ativo em tempo real da informação que chega e precisa de ser entregue num outro formato. Utilizando técnicas de conversão entre protocolos é possível enviar uma leitura de um sensor de temperatura utilizando um rádio nRF e visualizar a mesma num *Dashboard* instalado no nosso computador.

Atualmente existem *Gateways* prontas a instalar. Estas vêm equipadas com antenas para receber e enviar dados e, na maioria dos casos, dispõem de uma porta Ethernet para ligar à rede local, completando assim a ponte entre as duas tecnologias ao nível físico. No entanto, é ainda necessário converter todas as *frames* que chegam com dados binários em algo que o utilizador perceba, sendo esse processo de conversão realizado à custa de muito software. Temos o exemplo do protocolo *Serial Line Internet Protocol* (SLIP) que encapsula os dados provenientes de uma comunicação em série num pacote Internet Protocol (IP). Desta forma os dados que outrora não podiam navegar pela Internet e que estavam limitados à tecnologia série, têm agora características que permitem que sejam encaminhados corretamente por qualquer *Router* conseguindo chegar a qualquer parte do mundo e facilmente mostrados num computador.

Uma *Gateway* pode ter de converter um pacote de dados em diferentes protocolos devido a limitações existentes em cada uma das tecnologias, processo este que atrasa a comunicação. Num mundo ideal, se todos os dispositivos comunicassem via IP tal como a Internet funciona, o número de conversões protocolares seria drasticamente reduzido exponenciando a velocidade de comunicação. Como ainda não temos uma comunicação Global IP, ficamos à mercê de *software*/*firmware* capaz de entender cada bit que chega. Felizmente, tem existido uma comunidade ativa de software *Open Source* que tem tentado chegar com a tecnologia IP ao nível do Nó Cliente, para que este quando chegue à *Gateway* não tenha que sofrer grandes conversões. A conversão poderia ser a grande vantagem de trazer o IP ao Nó final, mas não. O facto de este ter IP faz com que possa ser visto pelos nossos computadores através do seu endereço. Assim sendo, é possível a partir de um computador endereçar um pacote diretamente a um nó cliente sem ter de perguntar à *Gateway* se este existe ou se está disponível. Isto reduz drasticamente a necessidade de alto processamento na zona da *Gateway*, mas faz com que esta necessite de mais largura de banda, pois agora os pacotes que chegam vêm com uns bits extra que fazem com que estes sejam interpretados como pacotes IP.

A biblioteca nRF24Network [10] desenvolvida pelo autor TMRh20 tem vindo a ser uma referência *Open Source* na implementação de IP em Nós Cliente. O mesmo autor tem também vindo a desenvolver a nRF24Mesh para implementar a topologia em malha em dispositivos baseados em nRF.

Analisando o que tem vindo a ser mencionado acima, para quem está para desenvolver algo novo, pode olhar para o processo e chegar à conclusão que é complexo, desnormalizado e sem suporte, seguindo assim por uma via mais segura que implemente *Standards*. Neste segmento temos algo que tem vindo a provar que funciona, que integra facilmente com outros sistemas e consegue comunicar a longas distâncias utilizando a topologia em malha: a tecnologia ZigBee. Esta tem vindo a ganhar terreno na implementação de dispositivos sem fio instalados em locais remotos, industriais e domésticos, e é neste último que se tem notado um aumento da adesão desta tecnologia que tem um cunho bem marcado e que garante um baixo consumo na comunicação, conseguindo assim com uma pilha do tamanho de uma moeda funcionar em média durante um período de 2 a 5 anos.

Quando analisamos o ZigBee, a tendência será seguir com esta tecnologia. No entanto esta é bastante dispendiosa, custando 15 vezes mais que um módulo nRF, e requer a adoção de todo um protocolo de comunicação, reduzindo o espaço para a experimentação e investigação. Daí a escolha do módulo nRF24L01+ para este estudo com vista a testar as mais diversas abordagens e perceber as potencialidades e as fragilidades existentes numa comunicação sem fio baseada em nRF24L01+ utilizando o protocolo da Internet.

1.2 Motivação

Aos olhos da humanidade as redes sem fios vieram para ficar e cada vez mais procuramos evitar a comunicação através de fios. Como qualquer outro equipamento eletrónico, estes dispositivos capazes de comunicar sem recurso a uma ligação física, necessitam também de uma fonte de energia. Os equipamentos sem fios apareceram em 1920 [11] para quebrar barreiras onde era impraticável ter um cabo ligado de uma extremidade a outra para comunicar. A outra grande vantagem é a simplicidade na mobilidade desse mesmo equipamento, o que resulta na necessidade de o mesmo ser pequeno e utilizar fontes de energia reduzidas.

O dispositivo emissor/recetor nRF24L01+ surge em 2008 e, 10 anos depois, este rádio é ainda capaz de resolver grande parte das necessidades atuais com um micro custo.

Por vezes, a incapacidade de um determinado equipamento não conseguir realizar algo, nem sempre recai sobre as limitações do hardware, mas sim sobre o esforço imposto a quem está a desenvolver o *firmware*. A tentativa de desenvolver ou otimizar código de bibliotecas que permitam ao hardware realizar algo que outrora nem sequer tinha sido considerado pelo fabricante nem sempre é realizada [12].

A Nordic desenvolveu este rádio há uma década atrás, sendo que os objetivos base eram dimensões reduzidas, baixo consumo, largura de banda entre os 250kb/s até 2Mb/s utilizando a banda de frequências de 2.4GHz ISM, tudo isto com um custo mínimo de produção.

Em 2014 o autor TMRh20 disponibiliza uma biblioteca em código aberto que permite comunicar com IP utilizando o nRF24L01+. O facto de ter sido possível implementar este tipo de tecnologia num rádio tão acessível e descontinuado é tão interessante como inovador. Um ano depois, o autor volta a surpreender e desenvolve uma vertente nova da biblioteca capaz de realizar comunicação numa rede de topologia em malha. Ambas as bibliotecas têm vindo a sofrer melhoramentos constantes até ao ano em que este estudo está a ser realizado.

A Nordic, empresa que desenvolve o nRF24L01+, atualmente desaconselha a utilização do mesmo para novos *designs*. Contudo, o estudo realizado e descrito neste documento procura perceber se é ou não possível utilizar este rádio em conjunto com a biblioteca totalmente otimizada e desenvolvida pelo TMRh20, para funcionar com tecnologias atuais que requerem necessidades especiais de largura de banda e protocolos específicos. Apesar de o fabricante ter descontinuado o nRF24L01+ existem ainda empresas concorrentes a fabricar o mesmo em grandes quantidades.

1.3 Objetivos

O objetivo principal deste estudo é detetar e perceber as limitações ou potencialidades do rádio emissor/recetor nRF24L01+ ao utilizar o protocolo da Internet.

Utilizando a biblioteca RFNetwork e RFMesh que implementa IP, na camada de comunicação, será analisado o impacto da mesma na performance do nRF24L01+. Ao ser adicionada uma camada extra na comunicação, o tamanho do pacote de dados aumenta. Em consequência disso, é necessária uma maior largura de banda, o número de retransmissões pode aumentar e a capacidade de processamento exigida é maior.

Para além das limitações de hardware, pretende-se conhecer as existentes na biblioteca RFMesh. Para tal será implementada uma rede em malha de longa distância.

Pretende-se ainda com os testes ter uma visão geral sobre o funcionamento e as capacidades deste rádio de baixo custo e descontinuado pelo fabricante original. Os módulos de testes serão desenvolvidos sob especificações base, podendo estas não ser as mais otimizadas no que toca ao consumo energético. No entanto serão suficientes para detetar o impacto na utilização de tecnologias atuais. Nesse sentido, será desenvolvida uma *Gateway* e três Nós Cliente.

O objetivo deste trabalho é verificar se é possível e comportável levar o protocolo da Internet até dispositivos (Nós folha) que recorram ao rádio nRF para a comunicação, quer seja em topologias

ponto-a-ponto, múltiplo salto (em linha ou em Malha). Dessa forma será possível utilizar bibliotecas de alto nível já existentes de forma a simplificar a programação, aumentando a rentabilidade e reduzindo custos.

A dificuldade em implementar tecnologia em dispositivos para o qual esta não está prevista é enorme. O facto de existir pouca documentação ou relatos de outras experiências realizados por investigadores, faz com que os testes realizados neste estudo possam ser uma referência para novas implementações com este microcontrolador.

Será importante manter a solução de teste o mais simples possível de forma a ser facilmente alterada e manipulada de modo a realizar o maior número de experiências em diferentes cenários.

1.4 Metodologia

Segundo Marconi M. e Lakatos E. [13] a Metodologia Científica é mais do que uma disciplina, significa que temos de realizar um conjunto de atividades sistemáticas e racionais. Com elas será permitido alcançar com maior segurança os objetivos. Desta forma terão de ser determinados os vários objetivos, já que o método para os atingir recai na realização de diferentes processos necessários para chegar ao resultado desejado sem que seja alterada a verdade.

O tema de estudo da dissertação envolve uma interpretação da viabilidade da utilização do rádio nRF24L01+ em novos designs. Com a realização de testes será possível quantificar alguns resultados, o que significa que será utilizada uma metodologia mista [14]. Esta preocupa-se principalmente com o sentido e entendimento do sistema em estudo, como também pela mensuração dos testes realizados.

Tratando-se de um caso de “hipóteses rivais plausíveis”, será definida uma estratégia onde podem ser retirados resultados plausíveis ou inconceptíveis. Para Yin [15] há dois paradigmas no método experimental, o primeiro que controla um número infinito de hipóteses rivais “sem especificar o que cada uma delas é” e apresenta cada uma delas como inconceptível com base num modelo estatístico. O segundo tem uma abordagem de experimentação, já que existe um isolamento experimental em que são documentadas e registadas todas as variáveis de teste. Por exemplo, temperatura do ar, voltagem aplicada, potência definida, etc...

Dando preferência ao segundo paradigma, esta metodologia tem como finalidade atingir os seguintes tópicos:

- Documentar cada caso de teste
- Analisar a robustez de um dispositivo baseado em nRF24L01+.

- Esclarecer eventuais investigadores ou engenheiros de hardware sobre as limitações do dispositivo.

1.5 Estrutura da Dissertação

A dissertação está dividida em capítulos de forma a permitir uma melhor compreensão da mesma.

Com início na introdução segue-se o capítulo do Estado da Arte, onde serão analisadas as diversas tecnologias sem fio que mais visibilidade têm no desenvolvimento de novos dispositivos. Ainda no mesmo capítulo serão enumeradas as principais vantagens e desvantagens de cada uma delas.

O terceiro capítulo pretende clarificar a arquitetura e as tecnologias utilizadas na realização do estudo de viabilidade. Aqui foi necessário desenvolver algum hardware com o rádio nRF24L01+.

O quarto capítulo apresenta a solução desenvolvida, detalhando as vantagens e desvantagens da mesma e justificando a escolha de cada componente para a realização do cenário de teste.

O quinto capítulo detalha o cenário de teste e inúmera minuciosamente os resultados espectáveis e obtidos em cada um deles com base na metodologia escolhida.

O sexto capítulo será uma conclusão do estudo acerca da viabilidade da utilização do rádio nRF24L01+, com mais de uma década, sendo que o próprio fabricante já não recomenda o mesmo em novos projetos, neste também serão abordadas possíveis iterações a este estudo.

Já no sétimo e último capítulo podem ser consultadas as referências utilizadas durante o estudo.

2 Estado da Arte

Neste capítulo serão apresentadas tecnologias de hardware que utilizam a frequência de 2.4GHz e protocolos utilizados atualmente no contexto da Internet das Coisas. Não sendo o objetivo desta dissertação comparar diretamente o nRF24L01+ a cada uma das tecnologias identificadas, é importante referir as mais populares, detalhando algumas das características que possam ter tido impacto na redução de dispositivos equipados com o hardware em estudo.

2.1 Hardware

Atualmente, quando é necessário desenvolver um dispositivo que utilize tecnologias sem fio, a pessoa responsável pela especificação recai normalmente para uma solução XBee ou Wi-Fi. Tendo isso em conta, neste capítulo serão apresentados os rádios, com ou sem microcontrolador associado, que mais se utilizam nos dispositivos de hoje em dia [16].

2.1.1 Xbee-Pro S2C 802.15.4

Os módulos RF Xbee / Xbee-PRO S2C 802.15.4 são soluções integradas que fornecem conectividade sem fio para dispositivos. Estes dispositivos usam o protocolo de rede IEEE 802.15.4 para redes ponto-a-multiponto ou ponto a ponto. Eles foram projetados para aplicações de alto rendimento que exigem baixa latência e tempo de comunicação previsível.

Existem duas versões do Módulo RF Xbee / Xbee-PRO S2C 802.15.4: *through-hole* (TH) Figura 2, e *surface-mount technology* (SMT) Figura 1. Os dispositivos TH incluem um conector de 20 pinos e exigem a colocação de dois adaptadores de 1x10 na placa de suporte para a montagem do dispositivo. Os dispositivos SMT incluem 37 *pads*. Eles são colocados diretamente na placa de circuito impresso, o que significa que não necessita de furos ou adaptador de montagem.

A versão TH pode ser útil para criação de protótipos e produção, mas recomendam o SMT para aplicações de alto volume de produção, já que o componente pode ser colocado automaticamente por uma máquina *pick-and-place* economizando assim no custo de um adaptador em cada placa.

O módulo RF Xbee / Xbee-PRO S2C 802.15.4 suporta as necessidades de redes de sensores sem fio de baixo custo e baixo consumo de energia. Os dispositivos requerem potência mínima e

fornecem entrega confiável de dados entre dispositivos. Os dispositivos operam dentro da faixa de frequência ISM de 2.4 GHz.

O módulo RF Xbee / Xbee-PRO S2C 802.15.4 usa o hardware S2C e o *Chipset* Silicon Labs EM357. Como o nome sugere, o módulo 802.15.4 é compatível com *over-the-air programming* e com o módulo *Legacy* 802.15.4 (hardware S1). As versões TH do novo produto também são compatíveis com projetos que utilizam o módulo antigo.

Os módulos XBee na sua maioria suportam uma configuração em Malha, aliás, a popularidade dos módulos XBee e ZigBee provém da sua capacidade de suportar este tipo de topologia. As redes em Malha são descentralizadas por natureza e cada Nó é capaz de descobrir automaticamente diversos caminhos na rede. Para além disso, sempre que um Nó sai ou entra estes têm a capacidade de se auto reconfigurarem para se adaptarem à nova configuração.

Este tipo de tecnologia fornece uma maior estabilidade em condições em que a falha de um ou mais Nós pode acontecer com frequência.

Uma característica importante destes módulos é o facto de consumirem pouca energia para comunicarem, permitindo assim funcionarem com uma pequena bateria durante anos.

Figura 1 - RF XBee SMT

Figura 2 - XBee-PRO TH

2.1.2 ESP8266EX

O ESP8266EX apresentado na Figura 3 é capaz de funcionar de forma consistente em ambientes industriais, devido à sua ampla faixa de temperatura operacional. Com recursos altamente integrados requer poucos componentes externos de suporte. O microcontrolador oferece fiabilidade, dimensões reduzidas e robustez.

Projetado para dispositivos móveis e IoT, o ESP8266EX tem um consumo reduzido de energia devido a uma combinação de várias tecnologias proprietárias. A arquitetura de economia de energia apresenta três modos de funcionamento: modo ativo, modo de suspensão e modo de suspensão profunda, os quais permitem que os projetos alimentados por bateria funcionem durante mais tempo.

ESP8266EX está integrado com um processador de 32 bits Tensilica, interfaces digitais, balanceador de RF, amplificador de potência, amplificador de baixo ruído, filtros e módulos de gestão de energia. Todos eles incluídos num encapsulamento pequeno.

O processador Tensilica L106 é desenvolvido utilizando tecnologia RISC de 32 bits, com um consumo de energia muito baixo, capaz de atingir uma velocidade máxima de relógio de 160 MHz.

O sistema operativo RTOS [17] e pilha Wi-Fi permite que cerca de 80% do poder de processamento possa estar disponível para programação e desenvolvimento de aplicações.

Figura 3 - ESP8266EX

2.1.3 nRF51822

O nRF51822 apresentado na Figura 4 usa o microcontrolador ARM Cortex M0 de 32 bits e uma memória flash, 256kB / 128kB, disponibiliza ainda 40kB-180kB para desenvolvimento de aplicações.

As velocidades de execução são consideravelmente superiores do que nas plataformas de 8/16 bits. O sistema *Programmable Peripheral Interconnect* (PPI) fornece um barramento de 16 canais para comunicação direta com periféricos. A possibilidade de colocar o processador em *Idle* traz reduções significativas no consumo. O dispositivo vem com dois modos de gestão de energia e todos os periféricos do sistema têm um controlo individual que permite uma troca automática *RUN / IDLE* de forma a realizar tarefas específicas.

O novo rádio é o responsável pelo aumento no desempenho do nRF51822 face às versões RF24. O rádio suporta *Bluetooth Low Energy* (BLE) na versão 4 e é compatível com os produtos da série nRF24L da Nordic Semiconductor.

A potência de saída é dimensionável de 4dBm até -20dBm em intervalos de 4dB. A sensibilidade pode ser aumentada em todos os níveis, permitindo ajustar (dependendo da taxa de dados).

O microcontrolador pode ser adquirido sob o formato *system on chip* (SoC) apresentado na Figura 4 ou em *Breakout Board* como mostra a Figura 5.

Figura 4 – SoC nRF51822

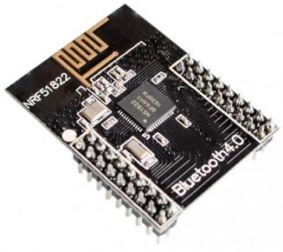


Figura 5 - Breakout Board nRF51822

2.1.4 nRF24X

Nesta secção será apresentada a família de dispositivos nRF24X com diferentes encapsulamentos, com e sem microcontrolador incluído e características que os permitem diferentes implementações.

2.1.4.1 nRF24LE1

O nRF24LE1 pertence à família de baixo custo e alto desempenho de emissores/recetores inteligentes RF de 2.4GHz com microcontrolador embutido. O nRF24LE1 é otimizado para fornecer uma solução integrada para aplicações sem fio *Ultra Low Power* (ULP).

A combinação de processamento, memória, osciladores de baixa potência, contador em tempo real, acelerador de criptografia AES, gerador aleatório e uma variedade de modos de economia de energia faz dele uma plataforma ideal para a implementação de protocolos RF. Os benefícios de usar o nRF24LE1 incluem segurança, menor consumo de energia e melhor desempenho de coexistência. Para a camada de aplicação, o nRF24LE1 oferece um vasto conjunto de periféricos, incluindo: *Serial Peripheral Interface* (SPI), 2 fios, *Universal asynchronous receiver/transmitter* (UART), Analog Digital Converter (ADC) de 6 a 12 bits, Pulse Width Modulation (PWM) e um comparador analógico de baixíssimo consumo de energia para o acordar.

Este microcontrolador é normalmente comercializado sob o formato de *Breakout Board* que pode ser utilizada em montagens SMT ou TH, Figura 6.

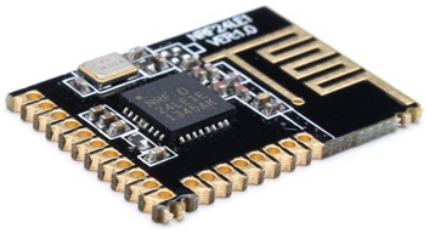


Figura 6 – Breakout Board *nRF24LE1*

2.1.4.2 nRF24LU1+

O nRF24LU1+ apresentado na Figura 7 é uma solução para *dongles* USB compactos. O nRF24L01+ de 2.4GHz suporta uma ampla gama de aplicações, incluindo periféricos de PC, acessórios desportivos e periféricos de jogos.

Com uma taxa de dados de 2 Mbps, combinada com o USB, suporta até 12 Mbps. O nRF24LU1+ atende aos mais rigorosos requisitos de desempenho de aplicações como ratos sem fio, controladores de jogos e controlos remotos de multimédia ou televisão.

Está integrado no nRF24LU1+ o seguinte:

- Um rádio RF nRF24L01+ que utiliza a frequência de 2.4 GHz;
- Um controlador compatível com USB 2.0;
- Um microcontrolador de 8 bits;
- 16 ou 32kb de memória flash.

Tudo isso é encapsulado num integrado compacto de 5x5mm, com um baixo custo. Equipado já com um regulador de tensão interno, permite assim que o microcontrolador seja alimentado diretamente pela USB, economizando assim custos e espaço na placa. Com RF totalmente integrado e *Phase-lock loop* (PLL) para o USB, não são necessários filtros externos. Tudo o que é necessário é um cristal de baixo custo de ± 60ppm de 16MHz e a antena.

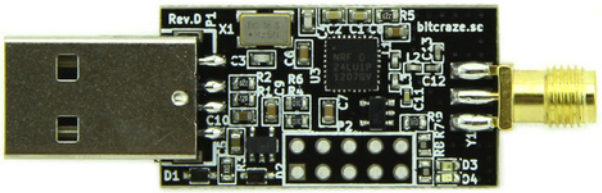


Figura 7 – Módulo nRF24LU1+

2.1.4.3 nRF24L01+

A Figura 8 ilustra o módulo nRF24L01+, um rádio de 2.4GHz com um mecanismo de protocolo de banda integrado (*Enhanced ShockBurstTM*), adequado para aplicações sem fio de baixíssimo consumo de energia. O nRF24L01+ é projetado para funcionar na banda de frequência ISM mundial dos 2.4 GHz.

Para projetar um sistema de rádio com o nRF24L01+ é simplesmente necessário um microcontrolador e alguns componentes passivos externos. O nRF24L01+ pode ser manipulado e atualizado através de *Serial Pheripheral Interface* (SPI). O mapa de registo, que é acessível através do SPI, contém todos os registos de configuração no nRF24L01+ e é acessível em todos os modos de operação do mesmo.

O mecanismo *ShockBurst* é baseado na comunicação de pacotes e suporta vários modos, desde a operação manual até a operação avançada de protocolo automático. O *Enhanced ShockBurstTM* reduz o custo do sistema ao lidar com todas as operações da camada de dados.

O *front end* do rádio usa a modulação *Gaussian Frequency shift keying* (GFSK) e possui parâmetros configuráveis como, o canal de frequência, potência de saída e taxa de dados. Suporta uma taxa de dados de 250kbps, 1Mbps e 2Mbps. Os dois modos de economia de energia tornam o nRF24L01+ muito adequado para projetos com consumo de energia extremamente baixo.

O nRF24L01+ é compatível com nRF24L01, nRF2401A, nRF2402, nRF24E1 e nRF24E2.

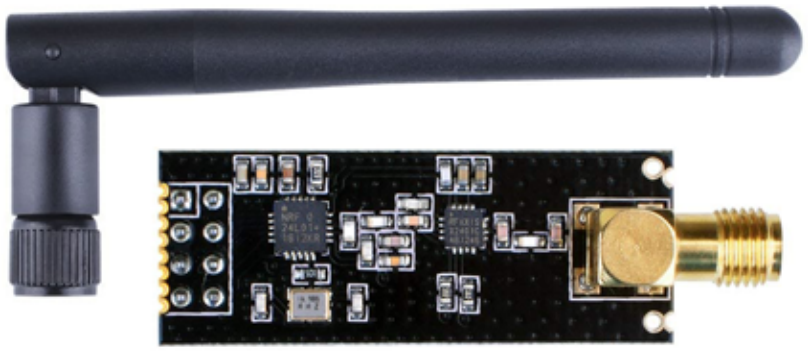


Figura 8 – Módulo nRF24L01+

2.1.4.4 nRF2460

O nRF2460 apresentado na Figura 9 fornece uma solução para *streaming* de áudio *Linear pulse code Modulation* (LPCM) mono de 16 bits a 32 kHz. A interface I²S é suportada para entrada ou saída de áudio. O dispositivo possui uma interface simples de baixo custo *Analog to Digital* (A/D) e *Digital to Analog* (D/A) para entrada e saída de áudio analógico. Tem um microcontrolador externo que controla o nRF2460 através de uma interface SPI ou *Inter-integrated circuit* (I2C).

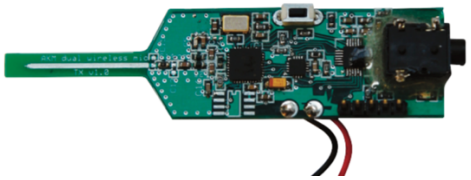


Figura 9 – Módulo nRF2460

2.1.4.5 nRF24AP2

O nRF24AP2 apresentado na Figura 10, é um membro da família de baixo custo e alto desempenho da Nordic Semiconductor para dispositivos de 2.4GHz com a pilha de protocolo ANT incorporada. O nRF24AP2 oferece a mais eficiente solução de rádio para redes *Ultra-Long Wavelengths* (ULW), através da integração da pilha de protocolo ANT.

É líder mundial em tecnologia RF de 2.4 GHz com oscilador de baixa potência e recursos de temporização.

Figura 10 – Módulo nRF24AP2

2.1.4.6 nRF52xxx

O nRF52832 apresentado na Figura 11 é o recente CPU da série nRF52 [18]. Preparado para suportar uma ampla variedade de aplicações que requerem recursos *Bluetooth Low Energy* 5, oferece até 512Kb de memória Flash e 64KB de memória RAM. Na Figura 12 é possível analisar os vários

elementos que compõem a sua arquitetura. O nRF52832 permite multiprotocolo com total concordância. Possui suporte para Bluetooth 5, Bluetooth Mesh, ANT a 2,4 GHz.

Foi construído com base num CPU ARM® Cortex ™ -M4 com uma velocidade de relógio de 64 MHz. Possui uma *Tag* NFC-A para uso em soluções simplificadas de emparelhamento e pagamento. Disponibiliza inúmeros periféricos e interfaces digitais, como *Pulse Density Modulation* (PDM) e I²S para microfones e áudio digital.

Dentro da família nRF52 existe diferentes modelos com características destintas que podem ser analisadas na Figura 13.

A eficiência energética deste microcontrolador é excecionalmente otimizada, alcançada através de um sofisticado sistema de gestão de energia integrado no chip.

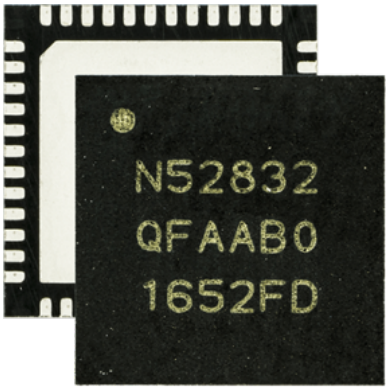


Figura 11 - SoC nRF52832, fonte Nordic

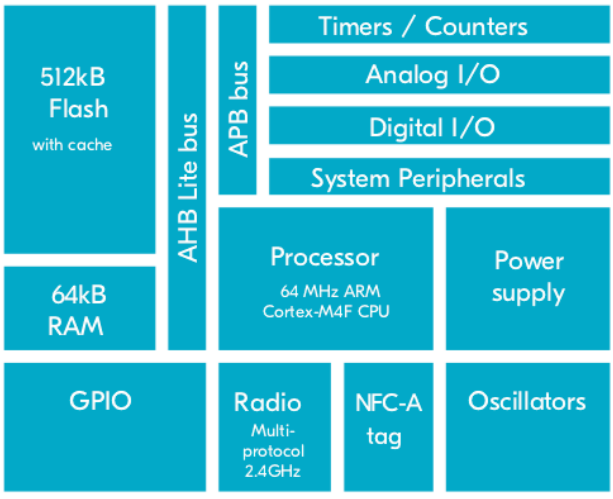


Figura 12 - Arquitetura nRF52xxx, fonte Nordic

Bluetooth 5	nRF52810	nRF52811	nRF52832	nRF52840
2 Mbps	X	X	X	X
CSA #2	X	X	X	X
Advertising Extensions		(X)	X	X
Long Range		(X)		X

(X) Software support planned

Figura 13 - Suporte de funcionalidade do nRF52xxx, fonte Nordic

2.2 Protocolos

Neste capítulo serão analisados alguns protocolos fundamentais no contexto da investigação, o ZigBee um protocolo proprietário que está atualmente a direcionar-se para o IP, o ANT que visa resolver problemas na comunicação e o MQTT que simplifica o envio e receção de mensagens entre dispositivos.

2.2.1 ZigBee 3.0

Atualmente o protocolo ZigBee está na versão 3 [19] e é suportado pela maioria dos módulos XBee. Este protocolo foi projetado para funcionar em ambientes com interferências causadas por outros equipamentos RF, por exemplo, ambientes comerciais ou industriais. A versão 3 unifica os perfis de comunicação de forma a interligar diferentes fabricantes, garantindo assim interoperabilidade.

Esta versão foi desenvolvida pela ZigBee Alliance em conjunto com mais de 300 fabricantes de tecnologia. A possibilidade de conexão a uma rede IP faz com que o ZigBee fique no radar da Internet das Coisas, possibilitando monitorização e controlo a partir de dispositivos como *smartphones* e *tablets* ligados a uma rede LAN ou WAN.

O protocolo inclui:

- Suporte a várias topologias de rede, ponto a ponto, malha ou multiponto;
- Baixo consumo / permite comunicar de forma a fornecer uma longa duração da bateria;
- Baixa latência;
- Até 65.0000 nós por rede;
- Segurança na ligação dos dados utilizando / Criptografia AES de 128 bits;
- Prevenção automática de colisões, reenvio e reconhecimento de Nós.

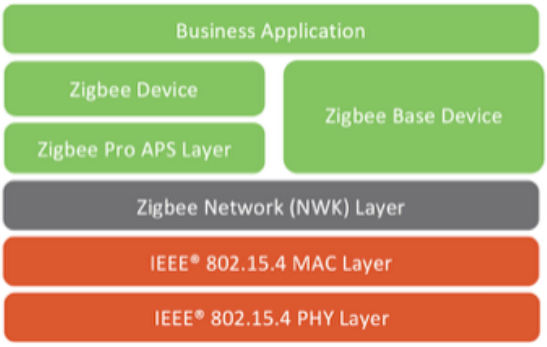


Figura 14 - Pilha do Protocolo ZigBee 3.0, fonte digi.com

A pilha do protocolo apresentado na Figura 14 incorpora um dipositivo base denominado de *Touchlink*, o qual fornece um comportamento consistente na forma de interligar os Nós e um conjunto de métodos que gere qual dos nós na proximidade é melhor para se ligar.

A versão 3 é compatível com versões anteriores, o que significa que as aplicações já desenvolvidas sob o perfil *ZigBee Light Link* 1.0, *Home Automation* 1.2, *Smart Energy* continuam funcionais.

A redes em Malha continuam a ser a componente chave do protocolo. Numa rede deste tipo os Nós estão interligados com outros Nós tal como a Figura 15 ilustra. As ligações entre estes são atualizadas e otimizadas dinamicamente por intermédio de uma sofisticada tabela de roteamento.

As redes em Malha são descentralizadas por natureza, já que cada Nó tem a responsabilidade de auto descobrir os melhores candidatos que lhe permitiam um acesso à rede. A vantagem desta topologia com roteamento *ad-hoc* é garantir uma maior estabilidade em condições em que determinados Nós podem falhar.

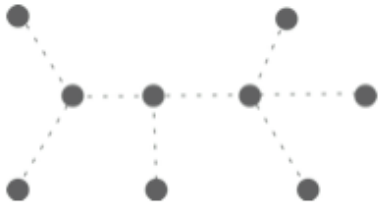


Figura 15 – Topologia em Malha, fonte digi.com

A aplicabilidade do ZigBee é muito vasta, passando pela implementação em redes sem fios que requerem baixo consumo de energia, fazendo uso de uma tecnologia inteligente, *Automatic*

Meter Reading (AMR), também utilizada para soluções de controlo de iluminação, sistemas de automação, monitorização de tanques, dispositivos médicos, entre outros, em que a tecnologia está a trazer avanços significativos.

2.2.2 ANT

Adaptive Network Topology (ANT) é uma tecnologia de comunicação sem fio bidirecional para redes sem fios de área pessoal (PAN) a 2.4GHz, otimizada para transferir pequenas quantidades dados com baixa latência entre vários dispositivos que também implementem ANT. O consumo de energia ultrabaixo do ANT garante uma vida útil prolongada até mesmo com fontes de baixa capacidade, como uma bateria do tipo moeda. É frequentemente utilizado em monitores de frequência cardíaca, conta quilómetros de bicicleta e relógios de pulso.

O ANT fornece um tratamento simplificado nas camadas de transporte, acesso a dados e física do modelo OSI e pode ser analisado na Figura 16. Além disso, incorpora os principais recursos de segurança, de baixo nível, que formam a base para implementações sofisticadas de segurança de rede definidas pelo programador. O ANT garante um controlo adequado e ao mesmo tempo reduz a carga computacional ao fornecer uma solução de rede sem fio simples, mas eficaz [20]. A sua arquitetura de camadas é apresentada na Figura 17.

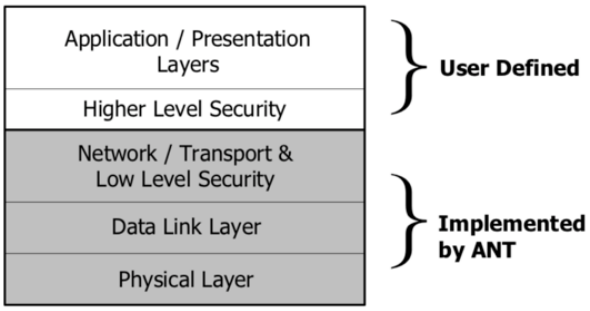


Figura 16 - Modelo OSI da ANT, fonte dynastream

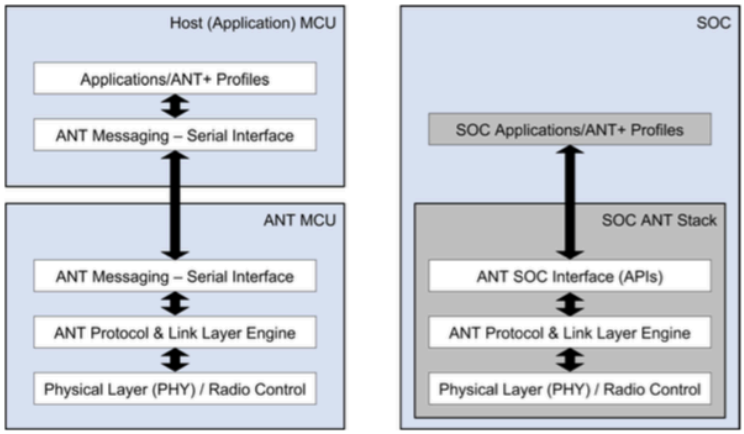


Figura 17 – Camadas do protocolo ANT, fonte dynastream

Na Figura 18 podem ser consultados os diferentes tipos de topologias suportadas pela tecnologia ANT.

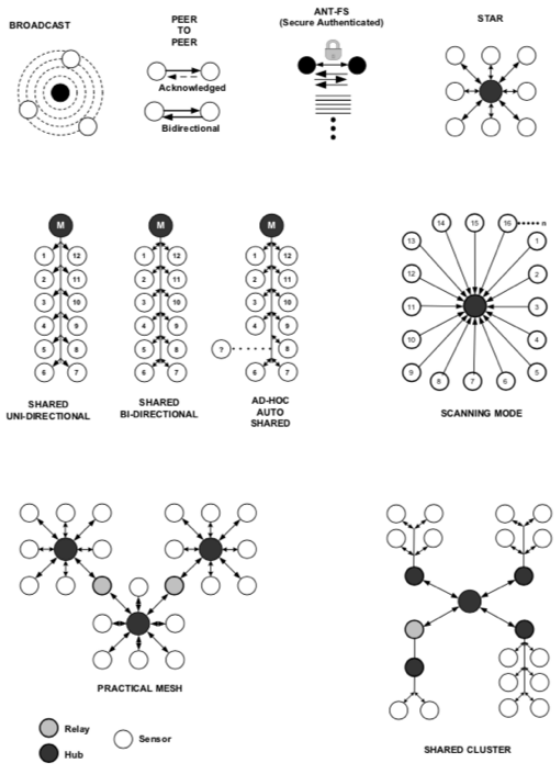


Figura 18 - Exemplos de redes ANT, fonte dynastream

2.2.3 MQTT

Message Queuing Telemetry Transport (MQTT), é um protocolo de mensagens extremamente simples e leve. Com base na publicação e subscrição, foi projetado para dispositivos restritos e redes de baixa largura de banda, alta latência ou não confiáveis. Os princípios de design do protocolo foram os de minimizar a utilização de largura de banda disponível na rede e os requisitos de recursos do dispositivo, ao mesmo tempo em que procura garantir a confiabilidade e garantia de entrega. A definição do pacote MQTT é exemplificada na Figura 19. Estes princípios também tornam este protocolo ideal para o emergente mundo da Internet das Coisas e para aplicações móveis onde a largura de banda e a bateria são aspetos a ter em conta.

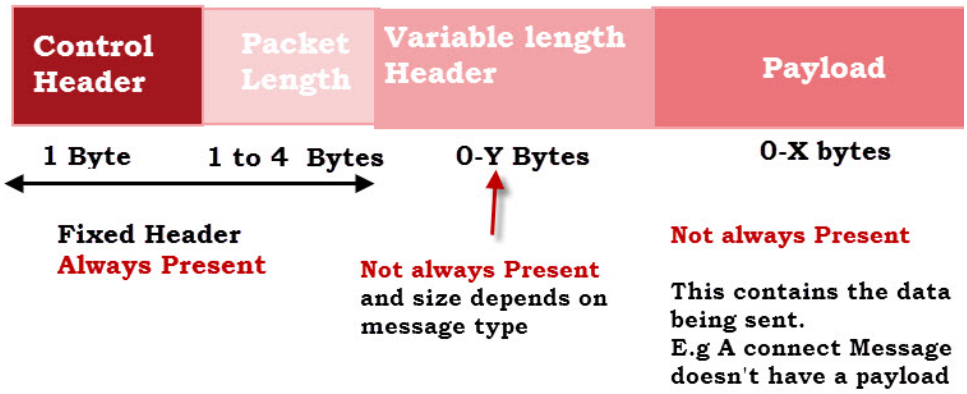


Figura 19 - Estrutura de um pacote MQTT, fonte hiveMQ

Para além do MQTT ser um protocolo leve, ainda implementa Qualidade de Serviço (QoS). Os 3 tipos de QoS possíveis de implementar são:

- **QoS 0 – *Most Once (Fire and Forget)*** - este é o nível mais baixo de QoS em que a mensagem é enviada apenas uma vez tal como a Figura 20 representa e não existe garantia de entrega. Apesar disso é o método que oferece maior rapidez na transferência de mensagens, daí ser o mais utilizado.

Figura 20 - MQTT QoS 0, fonte hiveMQ

- **QoS 1- *Least Once*** apresentado na Figura 21 - o emissor recebe de cada recetor informações que a mensagem foi recebida. Caso não receba a confirmação dentro de um determinado período de tempo, o emissor reenvia a mensagem. Com este método existe a garantia que a mensagem chega pelo menos a um recetor, mas poderá ser recebida em duplicado.

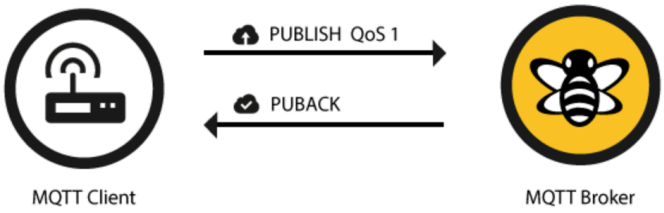


Figura 21 - MQTT QoS 1, fonte hiveMQ

- **Qos 2 – *Exactly Once*** apresentado na Figura 22, é o nível mais alto de QoS. Também designado de serviço assegurado, este dá a garantia da receção e de entrega de apenas uma cópia da mensagem, já que cada passo é reconhecido e validado. No entanto, esta confirmação torna o processo lento.

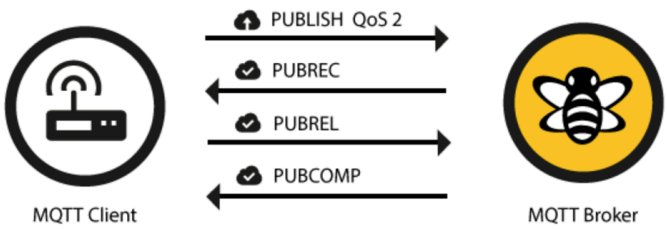


Figura 22 - MQTT QoS 2, fonte hiveMQ

O Qos é definido nas *flags* de controlo do pacote MQTT tal como a tabela da Figura 23 descreve.

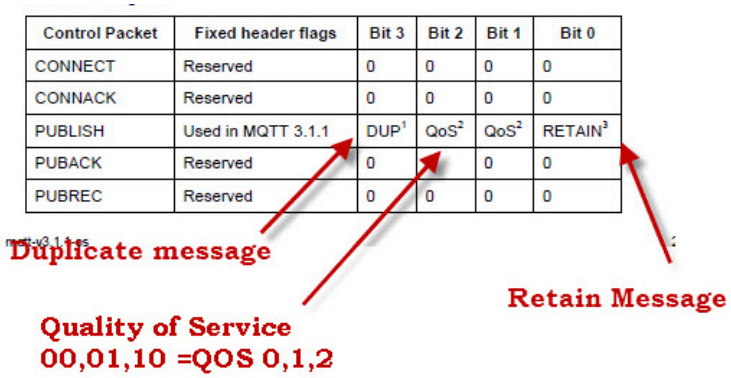


Figura 23 - Flags de controlo QoS MQTT, fonte hiveMQ

O protocolo MQTT é extremamente robusto e altamente leve e, com ele, podemos enviar mensagens estruturadas com o mínimo de impacto no tamanho do pacote. Para além disso, ainda existe uma economia de energia substancial, devido ao facto de existir menos tempo de processamento dedicado ao encapsulamento da mensagem. Para além de todas as vantagens já referidas, é de notar que o MQTT ainda oferece uma técnica que permite aos outros nós ou serviços saber que determinado Nó já não está disponível. Isto é conseguido com uma mensagem de "testamento" emitida pelo *broker* que avisa quem tiver subscrito ao Tópico de "Testamento" associado a cada nó. Esta mensagem indica que o *broker* perdeu o contacto e então avisa todos os interessados que aquela entidade já não está disponível. Existe ainda o inverso, em que o *broker* avisa os Nós interessados que outro Nó já está disponível.

A utilização deste protocolo de referência na IoT, num rádio nRF24L01+, aumenta e valoriza toda a solução.

2.3 Aplicações

A aplicabilidade do nRF24L01+ é vasta se pensarmos em dispositivos que apenas necessitam de enviar pequenas quantidades de dados em intervalos de tempo. Este rádio, com o seu baixo custo e fiabilidade, pode ser um fator chave face a outros produtos existentes no mercado [23]. No desenvolvimento de novos produtos é, de facto, interessante utilizar tecnologia antiga com um custo muito baixo capaz de competir com os mais atuais.

Um aspeto a ter em conta será perceber quanto tempo mais este rádio irá estar no mercado. Será que a Nordic vai continuar a suportar este dispositivo? Esta é uma questão pertinente, mas que deverá ser respondida de acordo com a necessidade e tempo de vida do produto para o qual estamos a pensar utilizar o nRF24.

Se se pretender um produto específico para resolver um caso isolado, então faz todo o sentido utilizar este rádio. No entanto, caso se pretenda um produto altamente comercializado e com um grande tempo de vida, por exemplo, uma panela inteligente em que utiliza o nRF para comunicar com outro dispositivo é essencial utilizar tecnologia atual para garantir o suporte durante vários anos e, durante o desenvolvimento da mesma, garantir que este pode funcionar com outros rádios caso um deles deixe de estar disponível ou seja detetada qualquer incompatibilidade ou instabilidade no mesmo. É importante ter em conta a versatilidade na comunicação num dispositivo, mas nem sempre é possível tanto a nível de arquitetura como em limitações no orçamento.

2.4 Projetos

Neste capítulo serão apresentados estudos onde a inovação passou por utilizar um rádio nRF24L01+. Veículos não tripulados, soluções sem fios para monitorização de espetáculos e dispositivos de localização Indoor de pessoas e objetos foram desenvolvidos tendo por base o rádio em estudo.

2.4.1 Quadcopter

Atualmente o aparecimento de Veículos Aéreos não Tripulados tem crescido exponencialmente. Governos e entidades privadas querem tirar o máximo partido desta tecnologia para fazer missões ou estudos em áreas inacessíveis ou perigosas.

A projeto desenvolvido pelo autor Leonardo Silva Ferreira, da Faculdade de Engenharia do Porto, tem como objetivo desenvolver um Quadcopter, apresentado na Figura 24, que utilize o nRF24 para o controlar e recolher dados [24]. O Veículo está equipado com GPS e Giroscópio. O rádio nRF e os motores estão ligados a um microcontrolador Arduino.

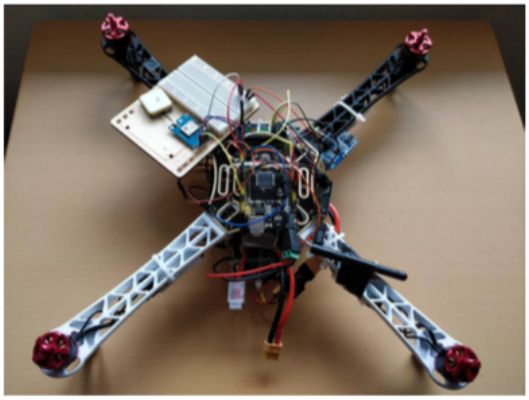


Figura 24 - Quadcopter com nRF24

Os testes apresentados na Figura 25 e Figura 26 foram realizados pelo investigador onde o nRF24 realizou a comunicação sem problemas.

Foram testadas configurações com e sem antena, registando a distância e o tempo de receção de dados em milissegundos.

Testes	Distância (m)	Tempo entre o envio e receção de dados (ms)
1	5	1767
2	20	1780
3	50	1784
4	> 100	Não capta sinal

Figura 25 – Testes nRF24 sem Antena, fonte Quadcopter

Testes	Distância (m)	Tempo entre o envio e receção de dados (ms)
1	1	1760
2	20	1784
3	50	1790
4	100	1804
5	500	1812
6	> 1000	Não capta sinal

Figura 26 - Testes nRF24 com Antena

Os testes demonstraram que o nRF24 consegue comunicar facilmente até 500 metros com antena, o que, por si só, é uma grande vantagem deste rádio que ainda existe no mercado em grandes quantidades e a preços muito baixos. Uma outra vantagem é o seu baixo consumo que permite ser alimentado com uma pequena bateria permitindo assim mais tempo de voo e veículos mais leves.

2.4.2 Localizador de Pessoas e Objetos Indoor

O trabalho realizado pelos autores Cesar Akasaka e William Aguiar [25] teve como objetivo desenvolver um dispositivo capaz de ser colocado em objetos ou pessoas de forma a obter a localização de cada um num espaço interior.

A Figura 27 apresenta a utilização de módulos nRF24L01+ em conjunto com um atenuador de sinal PE4302. Os autores pretendem utilizar técnicas de triangulação, trilateração e proximidade para estimar a localização do objeto ou pessoa em questão.

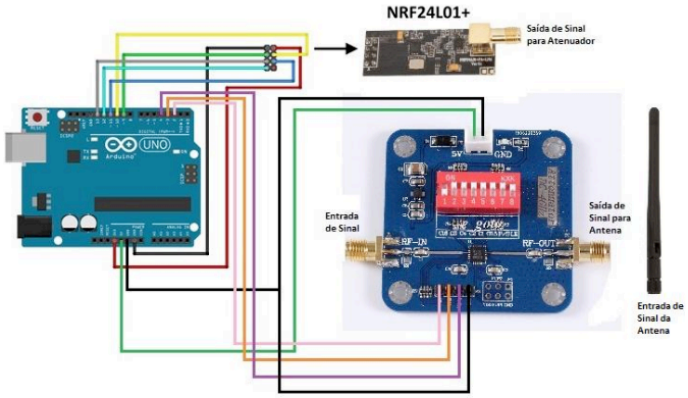


Figura 27 - Localizador Indoor com nRF24, fonte Localizador Indoor

A vantagem na utilização do nRF24 neste projeto recai sobre o seu baixo consumo, o que, em conjunto com o atenuador, é possível isolar de forma mais controlada a propagação de sinal, reduzindo assim interferências e sobreposição de dispositivos em ambientes mais pequenos.

2.4.3 Comunicação sem fios DMX512

O protocolo DMX512 é utlizado de forma massificada no controlo de dispositivos de iluminação em espetáculos. É possível definir regras de comunicação entre dispositivos e criar uma rede gigante que controla cada uma das iluminações de forma independente.

O trabalho de investigação realizado pelo autor Tiago Manuel de Oliveira Crespo, no Instituto Superior de Engenharia de Coimbra, teve como objetivo utilizar o rádio nRF24 para comunicar sem necessidade de fios, mantendo assim todas as potencialidades e garantias do protocolo DMX512 [26].

Esta investigação pretende solucionar a problemática das limitações do ambiente, já que nem sempre é possível passar cabos entre a mesa de controlo e o palco. Para alem disso, uma solução baseada em nRF24 teria custos menores devido ao facto de não necessitar de longos metros de cabo e ainda de reduzir o tempo de instalação.

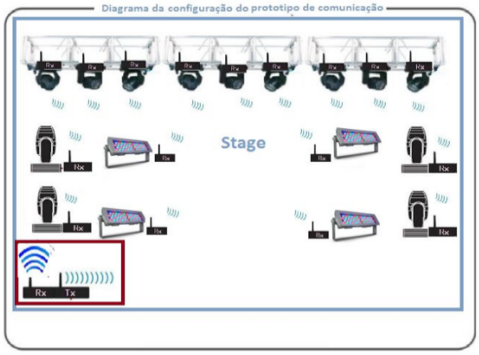


Figura 28 - Diagrama de comunicação DMX512

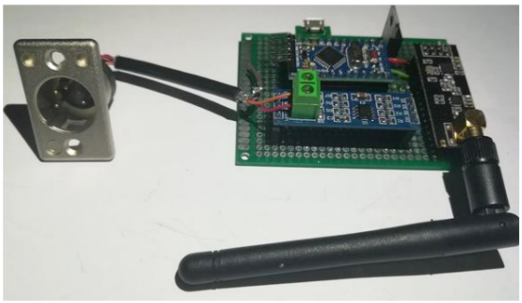


Figura 29 - Protótipo inicial DMX512

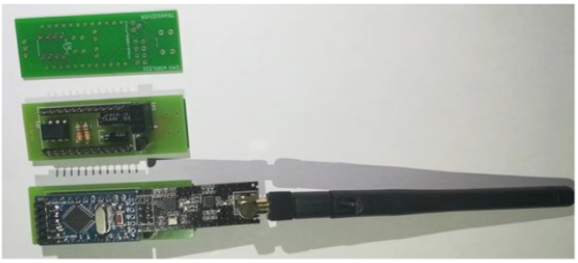


Figura 30 - Protótipo final DMX512

A Figura 28 apresenta a possibilidade de utilizar múltiplos nRF24 para controlar cada aparelho de iluminação. Já na Figura 29 é mostrado o resultado final do primeiro protótipo. Na Figura 30 o autor apresenta uma possível solução comercial de dimensões reduzias.

A fase de conclusão do projeto garante que o nRF24 cumpre com os objetivos de forma fiável e robusta, sendo possível realizar comunicações perto de 400 metros sem interferências devido à taxa mínima de transmissão do DMX512 ser de 250kbps [26]. No final foi possível reduzir o tamanho do protótipo tornando-o assim mais versátil e de fácil instalação.

3 Estratégia de Investigação

Conforme mencionado na secção 1.4 será utilizada uma metodologia mista com base na experimentação e análise dos resultados, para realizar todas as experiências que vão permitir analisar se o nRF24L01+ tem capacidade para lidar com cenários atuais, qualificando os resultados de forma plausível ou impraticável. Será necessário desenvolver uma rede de comunicação composta pelos mesmos rádios.

A rede de teste será composta por um conjunto de Nós e uma *Gateway* que irá fazer a entre a rede IP nRF24L01+ com uma rede IP local e posteriormente com a Internet.

3.1 Arquitetura

A arquitetura da solução a desenvolver passa pela simplicidade e tem como objetivo não causar entropia no estudo. Será composta por 3 Nós que comunicam, sem fios, utilizando a banda dos 2.4GHz, uma *Gateway* que vai receber os pacotes de dados do rádio RF e encaminhar os mesmos para a camada IP. A interface com um computador e o sistema operativo Linux/Unix é implementada via USB como camada física. Na camada lógica será utilizado o protocolo SLIP de forma a que o sistema operativo utilize a *Gateway* como mais uma interface de rede. Deste modo será possível atribuir um endereço IP à mesma.

O objetivo é simular uma rede funcional de Nós Sensoriais que trocam dados através do protocolo MQTT. Cada um dos Nós terá 2 modos de alimentação, um com alimentação dedicada, que terá toda a energia necessária disponível e outro que vai estar diretamente ligado a uma bateria.

3.1.1 Arquitetura do Nó

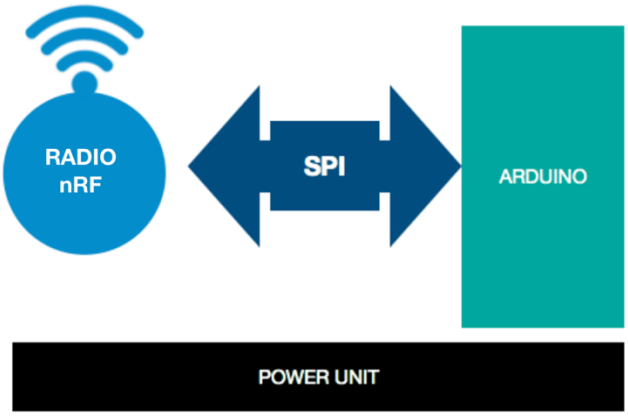


Figura 31 - Arquitetura do Nó

Na Figura 31 é apresentada a arquitetura de um Nó onde apenas o rádio nRF24L01+ e o Arduino são obrigatórios. A unidade de potência pode ser descartada no caso de a alimentação ser feita diretamente pelo pino VIN do Arduino, ficando assim responsável pela alimentação do circuito.

O rádio nRF24 faz a comunicação com Arduino recorrendo ao protocolo SPI. O processamento de dados entre rádio e microcontrolador é realizado recorrendo à *framework* Arduino e linguagem C++ na qual também a biblioteca RF24 foi desenvolvida.

3.1.2 Arquitetura da *Gateway*

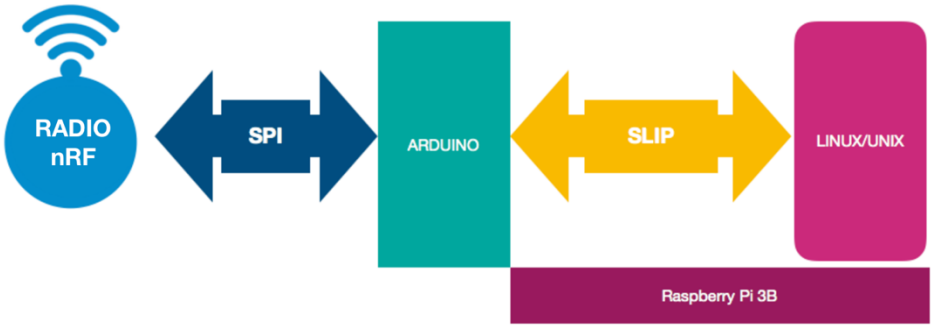


Figura 32 - Arquitetura da Gateway

A Figura 32 especifica como é que cada elemento de hardware interage entre si. Do lado esquerdo podemos verificar que existe uma replicação da arquitetura de um Nó. Do lado oposto

temos a parte de conversão dos dados RF, onde se recorre ao protocolo SLIP para encapsular os dados recebidos na interface série em pacotes IP. O sistema operativo Linux é instalado num mini PC Raspberry Pi.

Todo o tratamento de encaminhamento de pacotes na rede RF é gerido pela biblioteca RF24Network utilizada no microcontrolador Arduino. Todo esse processo será detalhado na secção de análise da biblioteca RF24 desenvolvida pelo autor TMRh20 [10].

3.1.3 Arquitetura de comunicação entre Nós e *Gateway*

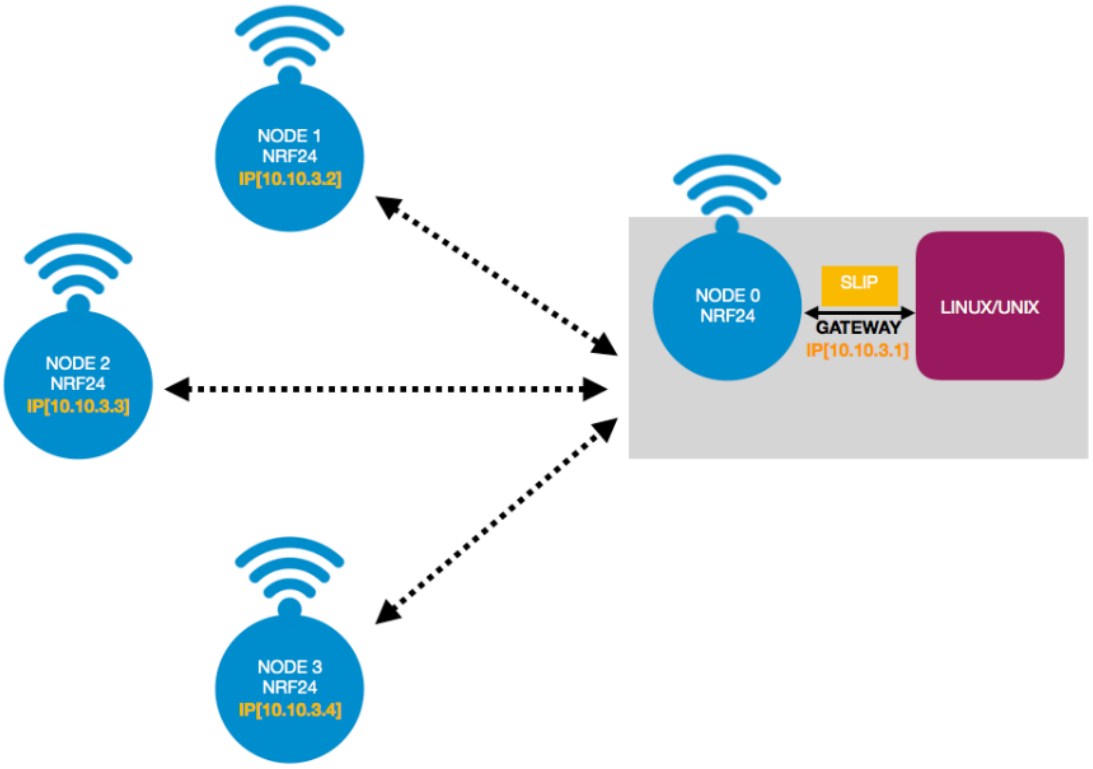


Figura 33 - Arquitetura de comunicação direta entre Nós RF e Gateway

A Figura 33 representa uma comunicação ideal, em que todos os Nós têm força de sinal para comunicar diretamente com a *Gateway*. Cada Nó é visível e alcançável dentro da rede IP, seja esta Local ou Remota, mediante determinadas configurações de segurança e roteamento.

3.1.4 Arquitetura de comunicação em malha

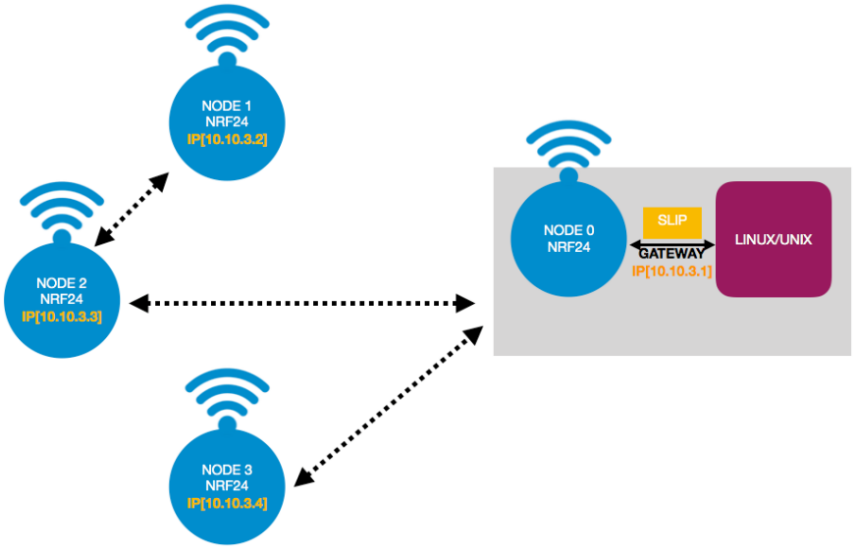


Figura 34 - Arquitetura de Comunicação em Malha entre Nós e Gateway

Na Figura 34 é possível verificar face à Figura 33 que o Nó 1 com o IP 10.10.3.2 está a comunicar diretamente com o Nó 2 que por sua vez tem o IP 10.10.3.3. Esta comunicação em malha é possível recorrendo à biblioteca RF24Mesh.

Se conciliarmos a biblioteca RF24Mesh com a RF24Ethernet e RF24Network é possível comunicar dentro da rede Local ou Remota para qualquer um dos Nós utilizando IP. Serviços baseados em IP podem agora comunicar de forma transparente dentro da rede Mesh mesmo que a *Gateway* não seja diretamente alcançável.

A rede em malha é totalmente dinâmica. Sempre que um Nó não consegue alcançar a *Gateway*, pesquisa nas redondezas por um ponto de saída. Todo o processo é automático e o programador não tem que se preocupar em gerir as tabelas de endereços, ou seja, toda a lógica complexa está abstraída e não existe preocupação em gerir a mutação da rede.

3.1.5 Arquitetura de armazenamento e análise

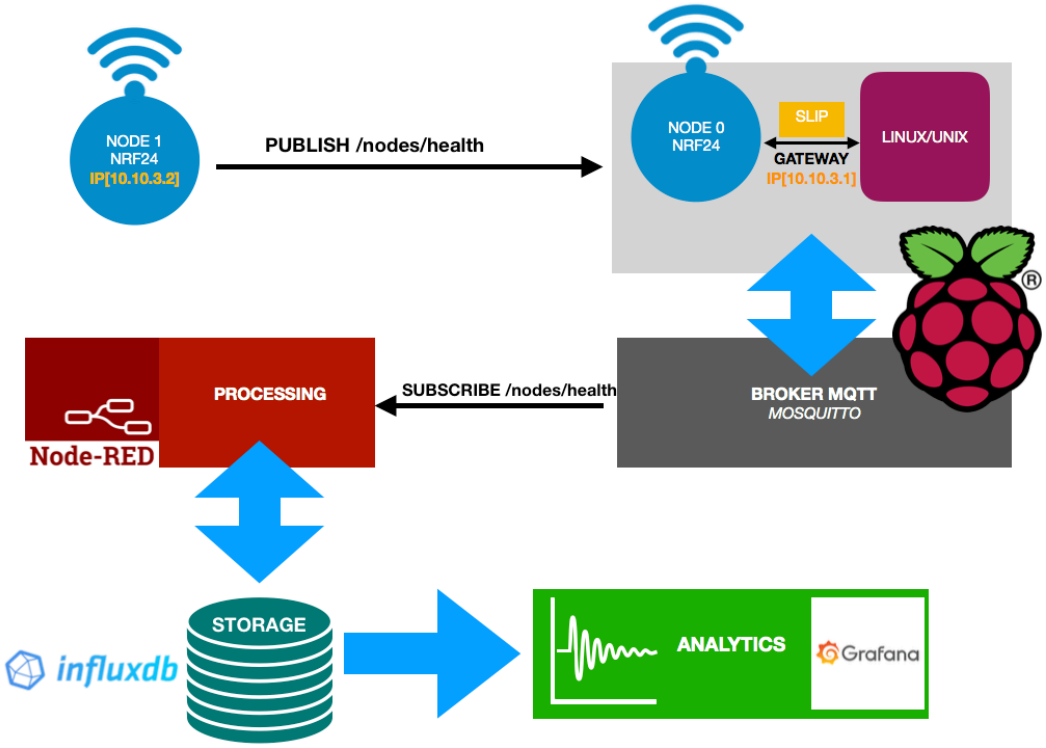


Figura 35 - Arquitetura de armazenamento e análise de dados

Na Figura 35 é apresentada a arquitetura desde o Nó até ao sistema de análise de dados. O processo é iniciado assim que a *Gateway* estiver disponível. Os Nós vão publicar mensagens de teste no *broker* Mosquitto e a Framework Node-RED subscreve ao tópico **/nodes/health** para receber as mesmas.

Os Nós publicam todos no mesmo tópico e a distinção de mensagens de cada um deles será tratada pelo Node-RED utilizando a propriedade *nodeId* encapsulada dentro de cada mensagem estruturada em *JavaScript Object Notation* (JSON). O tratamento de mensagens fica assim a cargo do Node-RED que é também o responsável por armazenar as mesmas, adicionando um *Timestamp* para análise e persistindo o resultado final na base de dados InfluxDB.

Como ferramenta de análise gráfica é utilizado o Grafana. Este permite, a partir de uma fonte de dados, gerar gráficos de forma a responder às várias questões de viabilidade relacionadas com este estudo.

A arquitetura faz recurso a tecnologia muito recente para análise e comunicação, o que, por si só, realça o facto de um rádio como o nRF24L01 ainda permitir, após uma década de existência, a integração com as soluções atualmente utilizadas em áreas como IoT.

3.1.6 Arquitetura da Topologia de rede da Biblioteca RF24Network

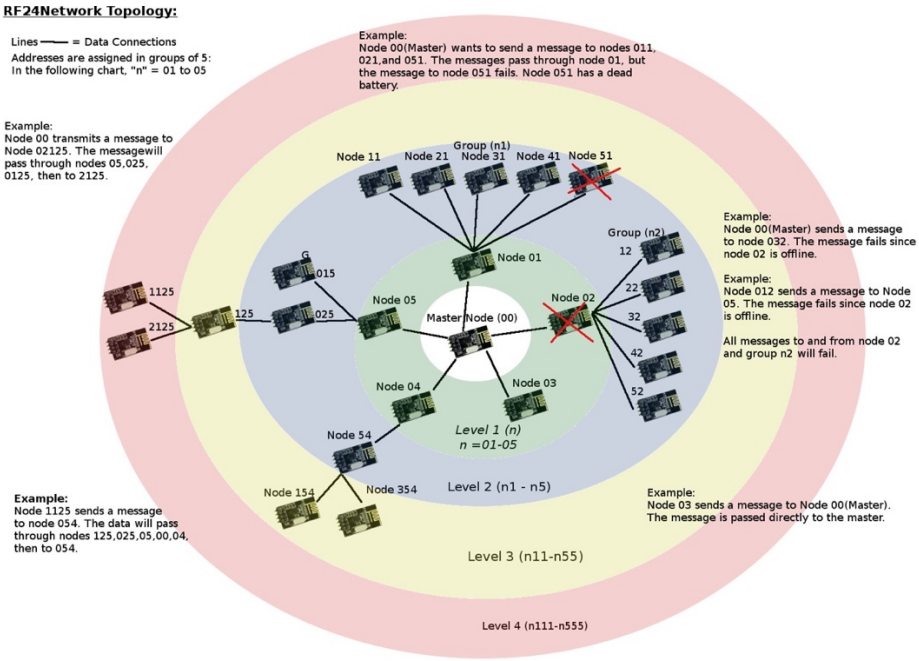


Figura 36 - Topologia da rede nRF24Network

A topologia da rede RF24Network apresentada na Figura 36, pode ser confundida com uma rede ZigBee. Apesar das semelhanças, estas não são compatíveis e utilizam conceitos diferentes.

Na rede gerada pela RF24Network quando uma transmissão ocorre de um módulo de rádio para outro, o recetor irá comunicar de volta ao remetente com um pacote de confirmação (ACK) para indicar sucesso. Se o remetente não receber um ACK, o rádio automaticamente faz uma série de tentativas intercaladas. Os rádios usam técnicas como endereçamento e numeração de carga para decidir a que nó devem ligar. Tudo é feito automaticamente pelo nRF24L01+, o programador não necessita de implementar qualquer lógica no *firmware*. Técnicas automatizadas de gestão de pacotes podem dificultar a transmissão de dados a um determinado nível, em que a repetição contínua de dados com falhas pode impedir a comunicação entre Nós ou reduzir o rendimento.

Os rádios da rede são associados por endereços atribuídos a canais. Cada rádio pode ouvir 6 endereços em 6 canais, portanto, cada um tem um canal pai e 5 canais filhos, os quais são usados

para formar uma estrutura em árvore. Comunicam diretamente com nós pai e filhos, qualquer outro tráfego para fora ou para um nó diferente deve ser encaminhado pela rede.

Quem está familiarizado com a rede IP deve ser capaz de compreender facilmente a topologia da RF24Network. O nó mestre pode ser visto como uma *Gateway*, com até 4 nós conectados diretamente, cada um desses Nós cria uma sub-rede abaixo dele, com até 4 nós filhos adicionais.

O esquema de numeração também pode ser relacionado a endereços IP, para fins de compreensão da topologia via sub-rede em que cada Nó pode ter até 5 filhos se o *multicast* estiver desativado.

3.1.6.1 Endereços RF24Network no formato IP

Para um endereço de um Nó Pai 10.10.10.10, os nós filhos seriam 10.10.10.1, 10.10.10.2, 10.10.10.3, 10.10.10.4 e 10.10.10.5. Os nós filhos de 10.10.10.1 seriam 10.10.1.1, 10.10.2.1, 10.10.3.1, 10.10.4.1 e 10.10.5.1

Na rede RF24Network, o pai é designado pelo 00, os seus filhos serão 01,02,03,04,05 e filhos de 01 são 011,021,031,041,051

O encaminhamento de tráfego é tratado de forma invisível para o programador. Este é realizado pela camada de rede. Se os endereços de rede forem atribuídos de acordo com o layout físico da rede, os Nós encaminham o tráfego automaticamente conforme necessário. Os programadores simplesmente constroem um cabeçalho contendo o endereço de destino apropriado e a rede trata do encaminhar para o nó correto. Nós individuais apenas encaminham fragmentos individuais, portanto, se for utilizada fragmentação, os Nós de encaminhamento não precisam de ser ativados, a não ser que enviem ou recebam dados fragmentados.

Se existir necessidade de encaminhamento de dados entre os Nós pai e filho, a rede usa funções de reconhecimento e repetição integradas do nRF24L01+ para evitar a perda de dados. Ao tentar enviar novamente dados registados como falha, há uma hipótese de duplicação de informação. Se o reconhecimento da rede não for bem-sucedido, nesse caso, cabe ao programador gerir as tentativas e filtrar os dados duplicados.

Os módulos de rádio nRF24 geralmente são capazes de enviar ou receber dados a qualquer momento, mas possuem mecanismos integrados de auto repetição para evitar a perda de informação. A parametrização de repetição é ajustada automaticamente pela biblioteca durante a sua inicialização, mas podem ser manualmente definidos para reduzir a perda de dados e, assim, aumentar a taxa de transferência da rede.

A biblioteca de rádio também tem a capacidade de ajustar a contagem interna de repetição automática dos módulos. A configuração padrão é de 15 tentativas automáticas por *payload* e pode ser configurando utilizando a variável *network.txTimeout*. É recomendável deixar as 15 tentativas padrão. Uma configuração com intervalos de repetição de (15,15) realiza 15 tentativas em intervalos de 4ms, levando até 60ms por *payload*.

3.2 Protocolos de rede

Os protocolos de rede são fundamentais para manter toda a infraestrutura robusta e transparente ao utilizador. Neste capítulo serão abordados o protocolo da Internet (IP) e o SLIP, um protocolo simples com uma longa historia.

3.2.1 IP

O protocolo da Internet (IP) realiza a transferência de pacotes com base no endereço lógico que é denominado de endereço IP. Os segmentos de dados são fragmentados, se necessário, para criar pacotes. A cada pacote é atribuído um endereço IP de origem e de destino. De seguida, com base no endereço destino, o pacote é encaminhado através das várias redes até chegar ao destino.

O IP é um protocolo sem ligação (*Connectionless* ou *CL-mode*), em que os pacotes podem seguir caminhos diferentes para chegar ao destino. Já no recetor, os pacotes são reagrupados e enviados para a camada de transporte.

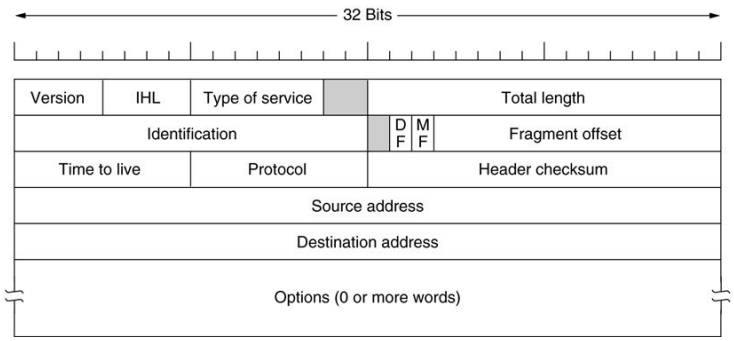


Figura 37 - Cabeçalho do protocolo IP

A vantagem de ser um protocolo endereçado traz imensos benefícios na comunicação, o que justifica a sua adoção globalmente pela Internet. Um dispositivo que "fale" IP tem a capacidade de comunicar facilmente com todos os sistemas que atualmente conhecemos e consumir os serviços

diretamente que correm nessas infraestruturas. Por exemplo, se um Nó IP quiser fazer um pedido HTTP, pode fazê-lo diretamente através do endereço onde esse serviço está alojado, sem necessidade de conversões de protocolo. Assim, todo o sistema fica mais simples e rápido.

3.2.2 SLIP

O protocolo *Transmission Control Protocol* (TCP) / IP é utilizado em vários tipos de rede tais como, IEEE 802.3 (Ethernet) e 802.5 (token ring) LAN's, X.25, satélites e *Bus* série.

Existe um encapsulamento padrão para pacotes IP definido para muitas dessas redes, mas não há padrão para *Bus* série.

O SLIP, *Serial Line* IP, é atualmente um dos padrões usado para ligações série ponto-a-ponto utilizando TCP/IP.

O SLIP apareceu por parte da 3COM UNET e define uma sequência de caracteres que contêm pacotes IP gerados a partir de uma comunicação série.

Não fornece endereçamento, identificação do tipo de pacote ou deteção de erros / correção. Por ser um protocolo tão simples é muito fácil de implementar. Em 1984, Rick Adams implementou o SLIP para o 4.2 Berkeley Unix e a Sun Microsystems tratou da disponibilização pública e este começou rapidamente a ser utilizado tornando-se uma forma simples de ligar *Hosts* TCP/IP a *Routers* ou computadores utilizando um *Bus* série.

A maioria dos sistemas baseados em UNIX/LINUX já inclui o SLIP. O Hardware necessário para a construção de uma *Gateway* é simples. Atualmente todos os microcontroladores já incluem uma interface *Universal asynchronous receiver/transmitter* (UART) *out-of-the-box*, logo este pode ser diretamente ligado a outra interface num sistema Linux sem complicações e sem custos utilizando o SLIP.

3.3 Fluxo de comunicação

Para validar a viabilidade de uma rede IP baseada em nRF24L01+ é necessário que exista comunicação dentro dessa mesma rede. Para tal, foi definido um fluxo de comunicação entre Nós e o *broker* MQTT. Cada Nó terá a capacidade de Publicar e Subscrever a determinados tópicos disponíveis num serviço IP. Como *broker* de mensagens será utilizado o Mosquitto. Este *broker* corre sobre o protocolo IP e está disponível no Porto 1883 com o IP 10.10.3.1. Para além dos Nós nRF, vai existir um segundo serviço de processamento que irá apenas subscrever aos

tópicos/nodes/health/ e **/nodes/health/available** de forma a escutar todas as mensagens que passam no serviço MQTT. Será o Node-RED a assegurar todo o processamento dessas mensagens e o seu respetivo armazenamento na base de dados para posteriores análises de viabilidade e performance.

A mensagem terá um formato em JSON com a seguinte estrutura:

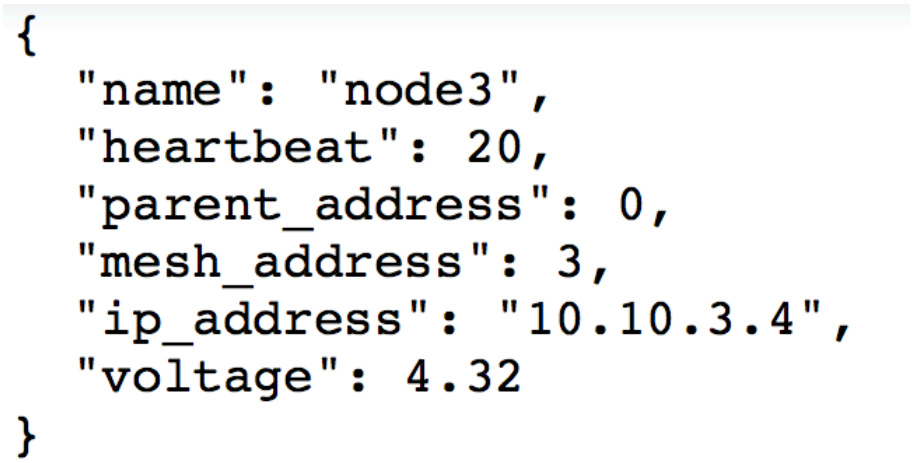


Figura 38 - Formato da Mensagem Health em JSON

Na Figura 38 podemos observar os campos que caracterizam a mensagem em que esta pode ter um tamanho entre os 100 a 140 Bytes. Com base no apresentado na Figura 19, sabemos que o pacote MQTT vai encapsular a mesma acrescentando assim entre 2 a 5 Bytes. A mensagem será posteriormente encapsulada em IP para ser enviada e este processo acrescenta mais 32 Bits tal como descrito na Figura 37, resultando num tamanho de mensagem entre 106 Bytes e 200 Bytes.

Apesar de a rede ser baseada em nRF, a biblioteca RF24Network implementa toda a arquitetura IP, logo para o programador é transparente consumir serviços que estão a correr num computador endereçado com IP. Com essa transparência na rede IP, é possível enviar e receber mensagens MQTT a partir de uma ligação direta usando o endereço IP do servidor de mensagens.

As principais bibliotecas que implementam clientes MQTT estão desenvolvidas em função do paradigma IP. O processo fica simplificado pelo uso das mesmas tal como se estivéssemos numa rede Ethernet ou Wi-Fi que pela sua natureza são redes IP.

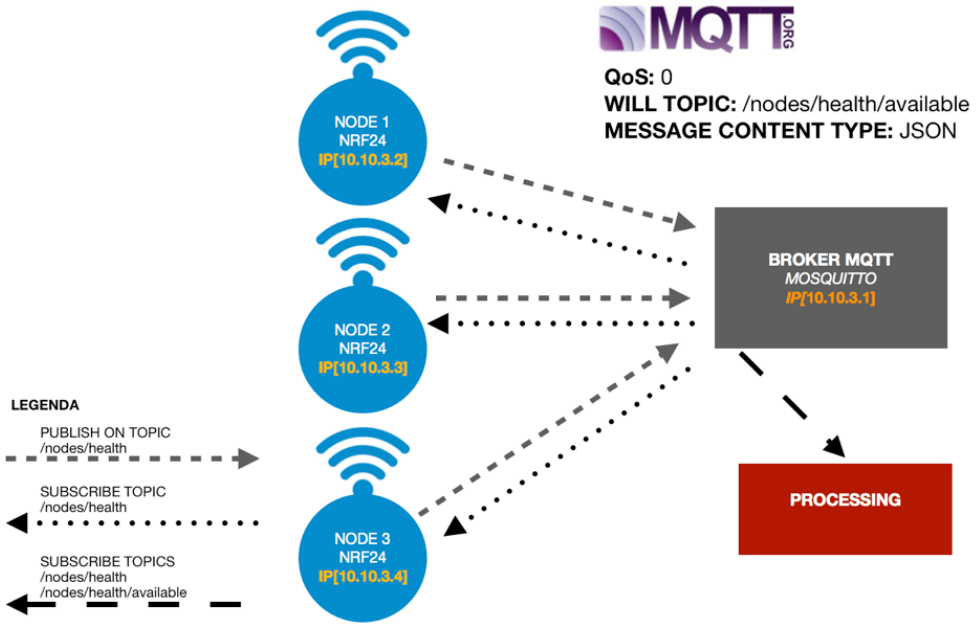


Figura 39 - Fluxo de Publicações e Subscrições MQTT

Ao analisar a Figura 39 não encontramos lá a *Gateway*. Do ponto de vista do nó, ele está a comunicar diretamente com o *broker*. Toda a "magia" é feita pela Biblioteca RF24Network em conjunto com o protocolo SPLIP.

No sector de processamento de mensagens e armazenamento, o Node-RED tem um fluxo que é iniciado quando recebe uma mensagem no tópico **/node/health**. A partir daí processa a mensagem em formato JSON, adicionando mais algumas métricas à mensagem, para permitir contextualizar a mesma a nível temporal. Por último persiste o resultado na base de dados InfluxDB. Todo o fluxo de processo pode ser visualizado na Figura 40 abaixo.

Figura 40 - Fluxo Node-RED para o processamento de mensagem

4 Apresentação do ambiente de teste

Na solução desenvolvida para realizar testes de viabilidade serão utilizados três Arduinos Nano, cada um deles ligado a um rádio nRF24L01+, com recurso ao protocolo SPI.

A *Gateway* será composta por um Arduino Uno e um rádio nRF24L01+. O microcontrolador Arduino liga por USB a um minicomputador Raspberry Pi Model 3B com Linux versão Raspbian Lite instalado.

No Raspberry é implementado o SLIP para relacionar os dados da porta série do Arduino numa interface de rede.

Para guardar todas as mensagens dos Nós será instalado no Raspberry o Node-RED, InfluxDB e Grafana para análise de dados.

4.1 Hardware

Nesta secção será descrito o *hardware* utilizado para o desenvolvimento dos dispositivos que serviram para testar a comunicação entre Nós.

A utilização da plataforma Arduino e do rádio nRF24L01+ teve como influência o facto da grande percentagem de projetos analisados utilizarem esta mesma configuração.

4.1.1 Rádio nRF24L01+

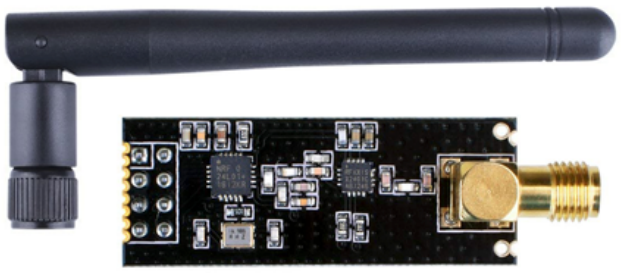


Figura 41 - rádio nRf24L01+ PA LNA com antena externa

A Figura 41 ilustra uma versão do nRF24 com um conector SMA que permite alterar o tipo de antena de forma a realizar diferentes testes de cobertura.

O nRF24L01+ vem equipado com dois amplificadores de potência, *Low-noise-amplifier* (LNA) e *Power-amplifier* (PA). A Figura 42 ilustra o modo como cada um deles é selecionado por intermédio de um *Duplexer*.

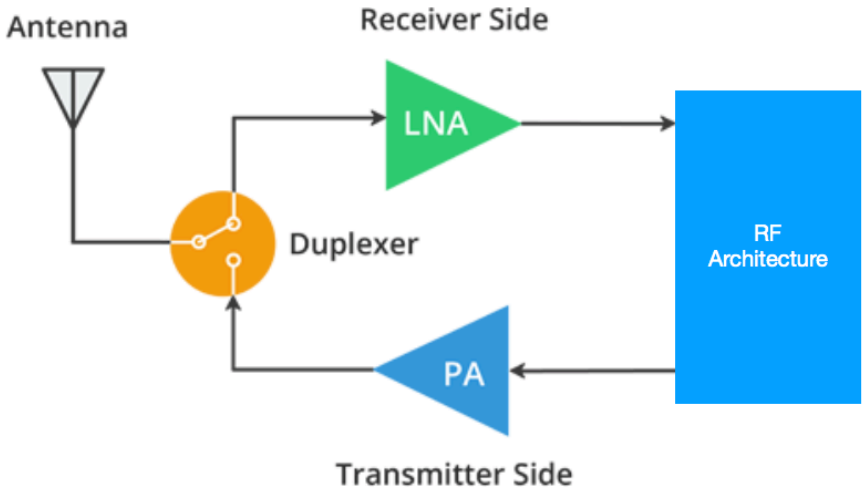


Figura 42 - Diagrama PA/LNA

Dentro da família nRF24 será escolhido o nRf24L01+ PA LNA, por ser o mais equilibrado a nível de características/preço. O conector SMA permite testar diferentes abordagens, trocando apenas a Antena por outras com diferentes tamanhos, formatos ou materiais. Pode ser ligado a qualquer microcontrolador que suporte SPI, o que traz enormes vantagens pois é possível escolher um onde a documentação é mais rica e a disponibilidade de bibliotecas é vasta.

Frequência	**2.4GHz ISM Band**
Taxa Máxima de Dados	2Mb/s
Modelações	GFSK
Output Máximo de Potência	0 dBm
Voltagem	1.9 a 3.6V
Corrente Máxima	13.5mA
Corrente Mínima (Modo Standby)	26µA
Distancia de Comunicação	800+ metro (linha de vista)

Tabela 1 - Especificação Técnica nRF24L01+

4.1.2 Arduino Nano

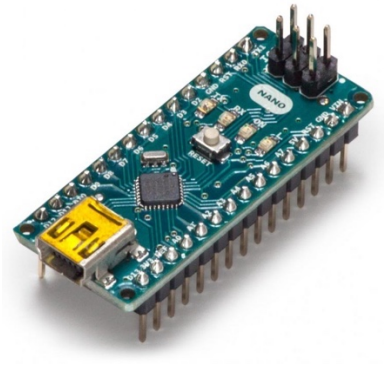


Figura 43 - Arduino Nano

O Arduino Nano apresentado na Figura 43 será utilizado no desenvolvimento dos Nós. É uma placa de desenvolvimento com microcontrolador Atmel e as suas dimensões não excedem os 4cm de comprimento por 1.5cm de largura. O microcontrolador Atmel é um ATmega328 com as seguintes características:

Arquitetura	**AVR**
Família	ATMega328
Voltagem de Entrada	5V
Memoria Flash	32Kb
SRAM	2Kb
Velocidade de Relógio	16MHz
Entradas Analógicas	8
Entradas/Saídas Digitais	22
SPI	SIM
Saídas PWM	8
Consumo energético	19mA
Peso	7g
Dimensões da PCB	18x45mm

Tabela 2 - Especificação técnica Arduino Nano

O facto de existirem bibliotecas bem documentadas para Arduino fez com que o Nano se tornasse uma escolha direta, já que as mesmas facilitam a implementação da rede IP sobre nRF24L01+ com um custo muito baixo de desenvolvimento.

4.1.3 Arduino Uno

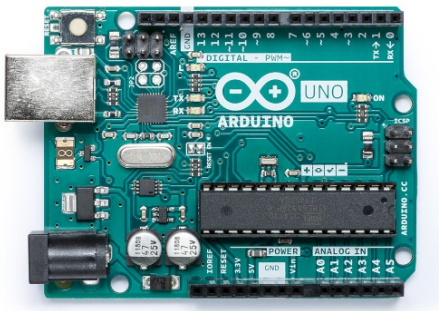


Figura 44 - Arduino UNO Rev. 3

A Figura 44 apresenta o Arduino Uno com uma nova revisão do microcontrolador, o ATMega328P.

Inicialmente a *Gateway* seria para ser composta por um Arduino Nano, no entanto durante o processo foram detetadas algumas instabilidades no sistema. Alterando para o Arduino UNO todo o sistema se manteve estável e a *Gateway* ficou capaz de lidar facilmente com os 3 Nós clientes.

Arquitetura	**AVR**
Família	ATMega328P
Voltagem de Entrada	7-12V
Memoria Flash	32Kb
SRAM	2Kb
Velocidade de Relógio	16MHz
Entradas Analógicas	8
Entradas/Saídas Digitais	14
SPI	SIM
Saídas PWM	8
Peso	25g
Dimensões da PCB	68x54mm

Tabela 3 - Especificação técnica Arduino UNO Rev3

4.1.4 Adaptador para nRF24L01

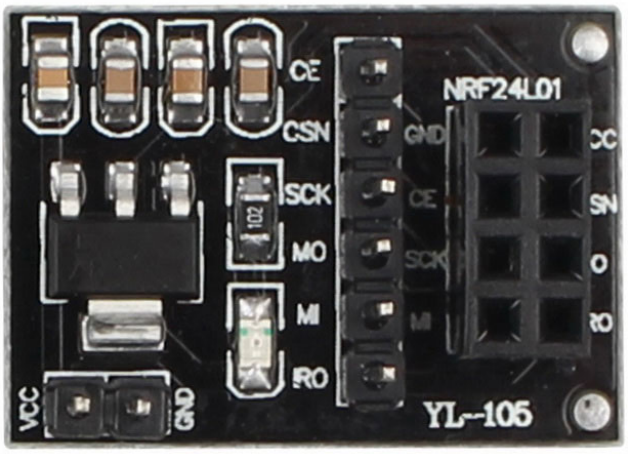


Figura 45 – O adaptador de nRF24L01

O adaptador ilustrado Figura 45 é um componente fundamental na ligação entre o microcontrolador e o rádio RF. Este pequeno módulo, para além de simplificar as ligações e garantir que a alimentação do módulo nRF é de 3.3V, adiciona uma filtragem na linha de alimentação para que seja retirado o máximo de partido do rádio. Sem este módulo foi possível notar um aumento na taxa de erros em cada comunicação ou até mesmo a uma ligação intermitente. A eletrónica implementada neste módulo é muito simples. Ele contém um regulador de tensão que permite uma entrada de até 7 Volts para uma saída de 3.3V. Por fim são adicionados quatro condensadores de bypass para filtrar ruído proveniente da fonte de alimentação ou do *Drop Down* do regulador.

Tanto os Nós como a *Gateway* vão utilizar este adaptador para fazer a interface entre os Arduinos e os rádios nRF24L01+.

4.1.5 Raspberry Pi

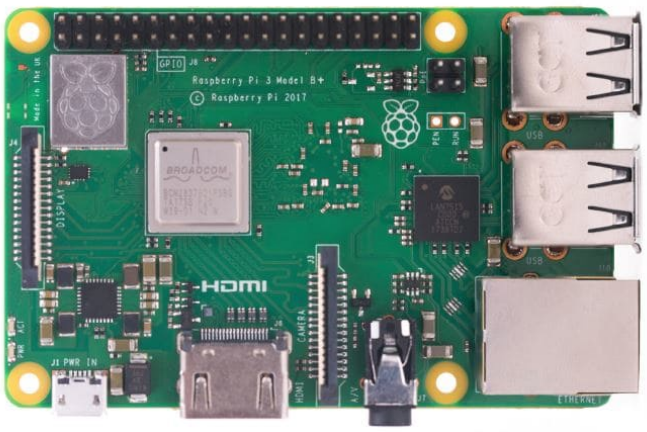


Figura 46 - Raspberry Pi Model 3 B

A escolha do minicomputador apresentado na Figura 46 apenas teve por base uma tentativa de reduzir os custos de investigação. O Raspberry Pi (RPi) é um minicomputador capaz de correr o Sistema Operativo Linux e toda a pilha de software necessário para armazenar e analisar as mensagens dos dispositivos nRF. O RPi é um minicomputador especial, uma vez que vem com diversos pinos acessíveis externamente para ligar diferentes tipos de hardware. Capaz de fazer a interface entre dispositivos SPI, este daria a oportunidade de excluir o Arduino Uno e ligar diretamente ao nRF24L01+. No entanto, estaríamos a afunilar toda a investigação que tinha como base uma *Gateway* com demasiada capacidade de processamento, acrescida de um sistema Operativo, sendo que o foco é perceber quais os requisitos mínimos para colocar a rede IP nRF a funcionar em pleno de forma estável, pelo que se optou por utilizar o nRF24 ligado ao Arduino.

Data de lançamento	**2/2016**
SoC	BCM2837
CPU	Quad Cortex A53 @ 1.2Ghz
Arquitetura	ARMv8-A
Armazenamento	Micro-SD
rede	10/100
Wireless	802.11n / Bluetooth 4.0
GPIO	40
Saída de Vídeo	HDMI

Tabela 4 - Especificação técnica Raspberry Pi 3 B

4.1.6 Solução assemblada

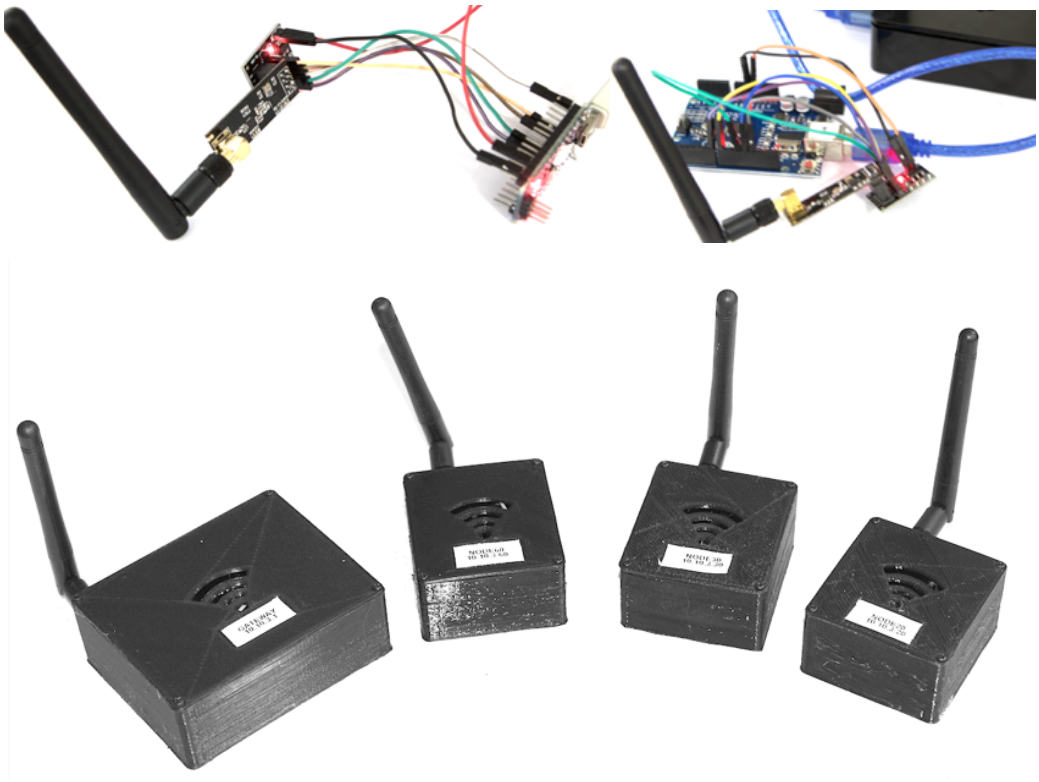


Figura 47 - Solução de Nós e Gateway

4.2 Firmware

No âmbito da investigação, desenvolveram-se dois *firmwares* distintos, um para a *Gateway* e outro para os Nós.

Recorreu-se à utilização da família de bibliotecas TMRh20. O autor tem vindo, desde 2014, a desenvolver e a aperfeiçoar bibliotecas capazes de enviar dados de um microcontrolador por nRF24, e disponibiliza as mesmas sob licenciamento GPL-2.0 no GitHub. Durante o desenvolvimento foram trocados diversos emails com o autor que rapidamente devolvia a resposta. Deste modo foi possível ter todo o apoio técnico e perceber como é que a tecnologia funcionava internamente.

Uma das questões a que este estudo procurava responder era se realmente seria possível criar uma rede em malha (Mesh) com recurso ao nRF24. A resposta do autor foi que a RF24Mesh fornece

uma *pseudo* Mesh em que baseia toda a comunicação sobre um único Nó Mestre. Este Nó mantém uma lista com os Ids de todos os seus Nós Filho e o endereço IP interno gerido pela RF24Network. O Id é um identificador único, exclusivo por dispositivo e o endereço atribuído pela RF24Network mostra a sua posição na rede. Esta técnica é muito engenhosa porque é possível simplesmente analisar o endereço RF24Network e a partir do mesmo um Nó pode determinar logo à partida o número de saltos para qualquer outro nó, dado o endereço de origem e de destino.

Para uma utilização TCP/IP, o TMRh20 desenvolveu a RF24Ethernet, e as especificações podem ser observadas na Figura 48.

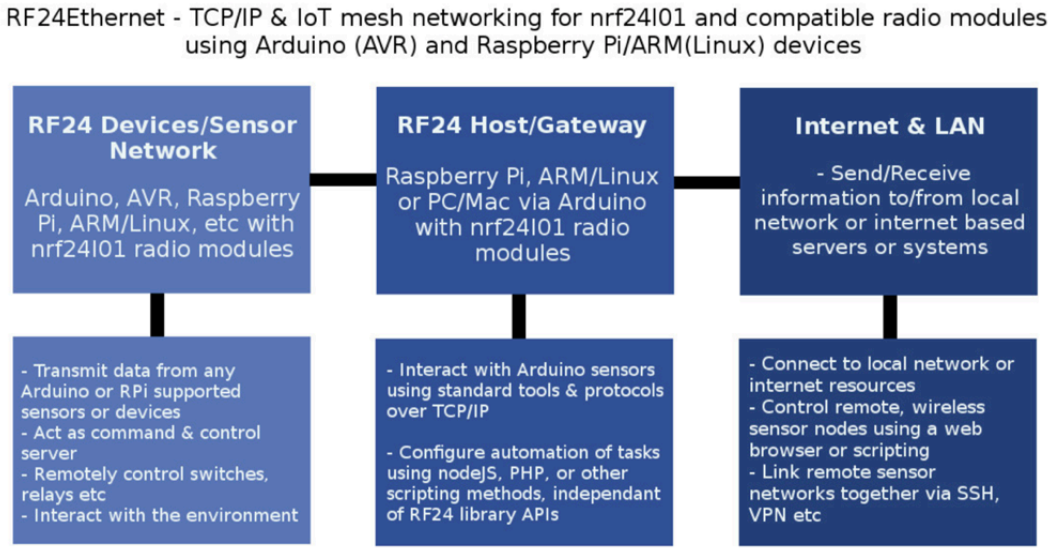


Figura 48 - Especificação da Biblioteca RF24Ethernet

Tratada a parte de transporte de dados, a comunicação MQTT é realizada pela biblioteca PubSubClient, disponibilizada diretamente e gratuitamente na comunidade Arduino.

O código fonte pode ser visualizado no Anexo 1 e 2.

4.3 Software

Neste capítulo serão descritas de forma sucinta as tecnologias de Software escolhidas para realizar os testes de viabilidade da solução. Através de uma pesquisa sobre as tendências da atualidade, a plataforma Node-RED apresenta uma enorme popularidade devido ao facto de simplificar a receção e tratamento de dados provenientes de diferentes fontes. Em relação ao Grafana,

é a ferramenta de eleição para a maioria dos sistemas de análise de dados, já que permite de forma muito rápida visualizar os dados através de gráficos em tempo real. Por fim, a arquitetura de persistência de dados implementada no motor InfluxDB permite realizar consultas tendo por base o tempo em que cada medição ocorreu. O InfluxDB está preparado para lidar com enormes quantidades de dados, garantido assim consultas rápidas dos mesmos através de intervalos temporais.

4.3.1 Node-RED

A Internet das Coisas impulsionou o desenvolvimento de plataformas de programação gráfica baseadas em fluxos de execução [28]. O Node-RED [29] é uma ferramenta que permite relacionar dispositivos de hardware com serviços locais ou remotos. O desenvolvimento nesta plataforma é feito através do browser de Internet e é tão simples como arrastar um ícone. O Node-RED pode ser instalado localmente ou em Cloud, é desenvolvido em Node.js e tira partido de todo o modelo assíncrono de processamento orientado a eventos. Isso faz com que seja ideal para ser instalado em plataformas com poucos recursos.

A escolha do Node-RED fez com que a receção de mensagens via MQTT, o processamento das mesmas e a sua persistência na base de dados InfluxDB, se tornasse um processo simples, já que o mesmo implementa todos os conetores necessários para lidar com todas as tecnologias selecionadas para a investigação. O tempo de configuração da plataforma foi rápido e direto devido ao facto do Node-RED ser tão simples de usar e de estar muito bem documentado.

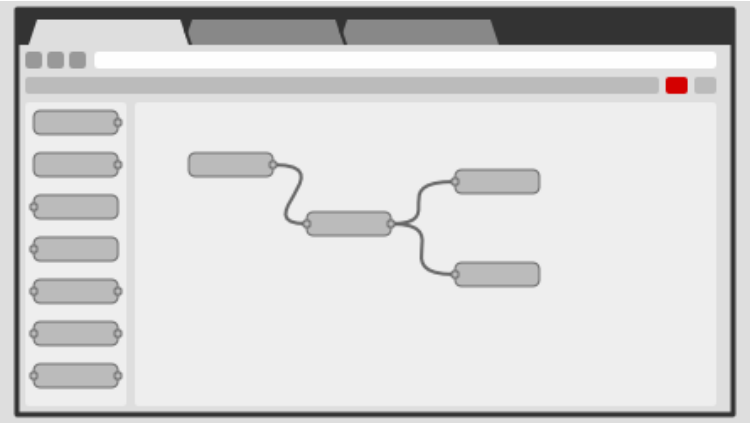


Figura 49 - Node-RED

4.3.2 Grafana

A cultura tecnológica de hoje em dia está cada vez mais orientada aos dados. A utilização de plataformas que possam transformar os dados em informação útil é uma constante necessidade. O Grafana [30] aparece sob a forma de uma ferramenta poderosa e *Open Source*, permitindo consultar, visualizar e até mesmo alertar dependendo das métricas configuradas pelo utilizador.

O Grafana tem a versatilidade de se conseguir ligar a diferentes tipos de fontes de dados, sejam folhas de Excel, bases de dados relacionais ou até mesmo tecnologias mais avançadas como séries temporais. Permite ao utilizador, em poucos minutos, criar um *Dashboard* composto com os mais diversos tipos de gráficos, com a característica de filtrar os mesmos em intervalos de datas ou em tempo real.

Recorrendo ao Grafana não existiu necessidade de desenvolvimento de uma plataforma para analisar os dados e latências produzidas durantes os testes de viabilidade. Esta ferramenta tem vindo a ser uma referência na comunidade e efetivamente reduz o tempo na criação de gráficos permitindo assim ao utilizador focar-se no que realmente importa, obter respostas de cada *dataset* transformado em informação.

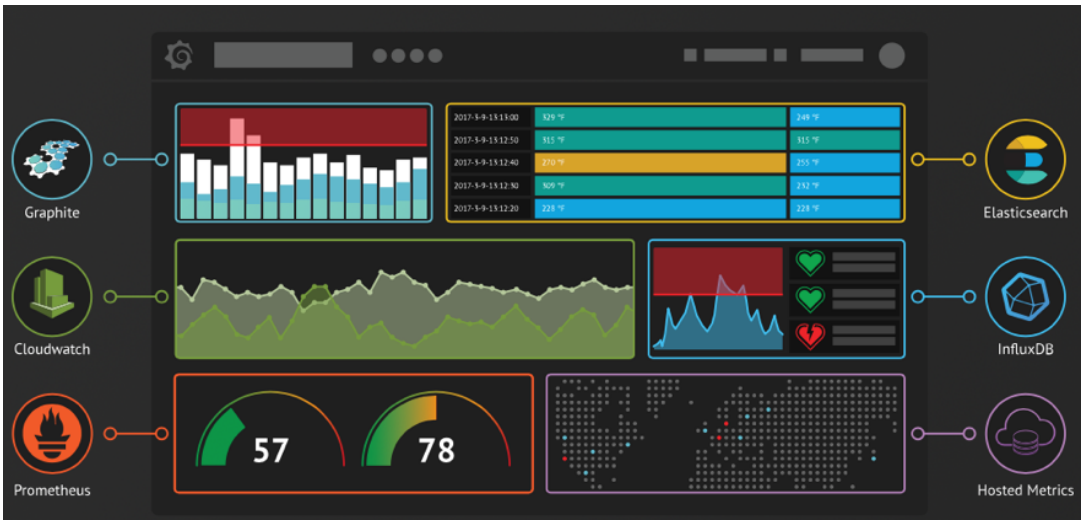


Figura 50 – Grafana

4.3.3 InfluxDB

O InfluxDB [31] é uma base de dados baseada em séries temporais (*time series database*) e está otimizada para ser extremamente rápida em leituras e escritas. A forma como esta tecnologia organiza os dados permite rapidamente obter respostas em intervalos de tempo específicos. Por exemplo, se existirem milhões de registos de temperaturas registados durante um ano onde é especificada a data em que a mesma foi medida, o InfluxDB gera automaticamente estruturas auxiliares já com intervalos de tempo pré-preparados. A vantagem desta otimização é notada quando o utilizador pede para ser mostrado em gráfico ou dados estatísticos relativos desses dados. Numa base de dados relacional uma ordenação temporal em milhares de registos poderia invalidar o serviço, já com uma base de dados temporal, a resposta ao pedido é devolvida em poucos segundos [32].

Figura 51 - Editor InfluxDB

5 Análise

Para verificar a viabilidade da implementação de IP numa rede baseada em nRF24L01+, será definida uma bateria de testes. Esta terá como objetivo compreender se o nRF24 consegue manter uma ligação estável e rápida. É importante relembrar que o nRF24 já tem quase uma década e foi pensado para enviar pequenos pacotes de dados numa rede não endereçada. Se os testes apresentarem bons resultados, será uma oportunidade de trazer de volta o nRF24 para o mundo da Internet das Coisas, tal como a ZigBee Alliance está a fazer com a versão 3.0.

5.1 Plataforma de Testes

Para a realização de testes foram desenvolvidos uma *Gateway* nRF24L01+, três Nós nRF24L01+ e foi utilizado um Raspberry Pi para monitorizar e validar todo o sistema de receção de mensagens.

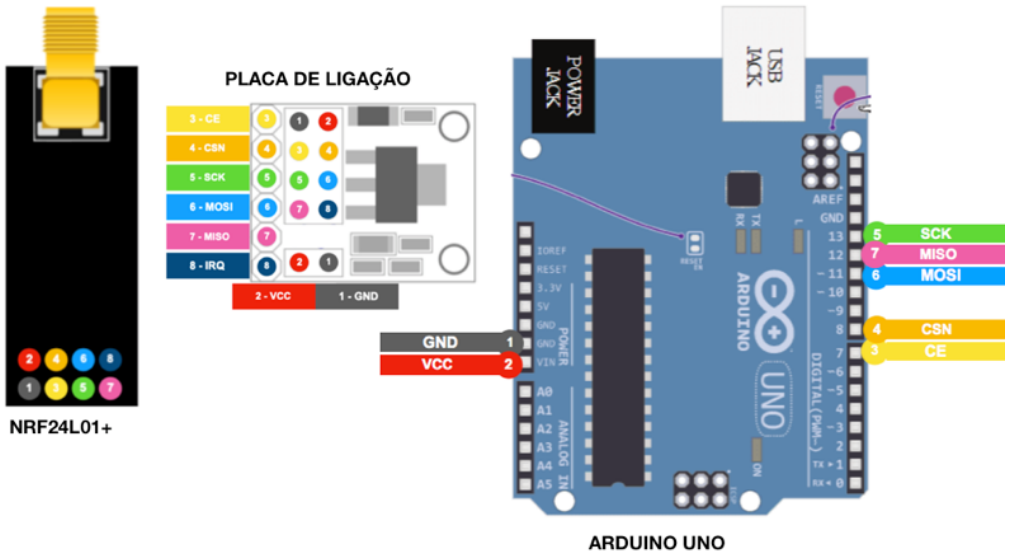


Figura 52 - Esquema de ligação Gateway

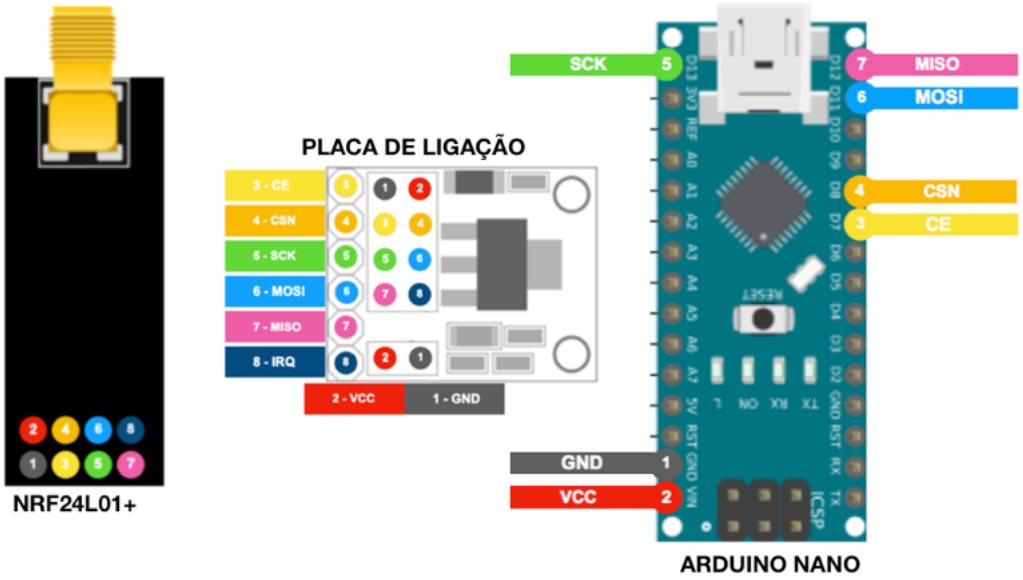


Figura 53 - Esquema de Ligação Nó

GND *Ground*, está definido em ambas as placas como GND.

VCC Energia, utilizado para fornecer energia ao dispositivo. O Arduino e a placa de ligação têm soldado um regulador de tensão AMS1117, dado que este o sistema aceita tensões recomendadas entre 1.9V a 9V. Sem a placa de ligação apenas seria possível utilizar tensões máximas de 3.6V, onde valores superiores destruiriam o nRF24L01+.

Chip Enable (CE) Controlo, permite ativar ou desativar a transmissão ou receção de dados sempre que ativo.

Chip Select Not (CSN) Controlo SPI, sempre que ativo o nRF24 inicia a escuta de dados com origem no *BUS* SPI.

Serial Clock (SCK) Relógio, aceita pulsos do *Master* ligado ao *BUS* SPI, desta forma permite acertar a velocidade de comunicação.

Master Out Slave In (MOSI) Entrada de dados no *BUS* SPI.

Master In Slave Out (MISO) Saída de dados no *BUS* SPI.

IRQ Interrupção, ativa sempre que chegam dados novos ao nRF24 e ficam prontos para entrega no *buffer*. Este pino pode ser muito útil em situações *Wake-Up/Sleep*, acordando o microcontrolador sempre que chegam novos dados.

Figura 54 - Raspberry Pi 3 Model B+

Figura 55 - Transformador 5V 3.0A

Figura 56 - Cartão Micro SD Kingston 32GB CL10

Como enunciado acima, será utilizado um Raspberry Pi 3 Model B+, com um processador Quad Core 1.4GHz, 4 Portas USB, Lan 10/300 e 1GB de memória LPDR2. O Sistema operativo Raspbian 4.14.90-v7+ foi instalado num cartão micro SD de 32GB CL10, o micro computador será alimentado com um transformador aconselhado pelo fabricante com uma potência de 15W. Fontes de potência inferiores causam instabilidade no sistema.

De forma a analisar todas as comunicações realizadas, estas serão guardadas numa base de dados de séries temporais InfluxDB versão 1.7.3. Os dados são recebidos e guardados na base de dados utilizando o Node-RED versão 0.19.5. Por fim, os dados são analisados e comparados com recurso à ferramenta gráfica Grafana versão 5.4.3.

Para realizar a comunicação entre a *Gateway* nRF24L01+ ligada via USB foi utilizado o protocolo SLIP e a configuração é apresentada na Figura 57. É importante referir que o rádio nRF24L01+ permite uma ligação direta aos pinos de interface do Raspberry Pi, deixando assim de

ser necessário o Arduino UNO. A biblioteca nRF24 do autor TMRh20 permite toda a implementação com Arduino ou Raspberry. Para a investigação achou-se que seria mais benéfico fazer testes com um microcontrolador de capacidades reduzidas face a um processador com uma capacidade 900x superior. Com os resultados será possível apurar se com apenas 16MHz e 2K de memória é possível desenvolver uma *Gateway* viável.

```
sudo modprobe slip
sudo slattach -L -s 115200 -p slip /dev/ttyUSB0 &
sudo ifconfig sl0 10.10.3.1
sudo route add -net 10.10.3.0/24 gw 10.10.3.1
```

Figura 57 - Configuração SLIP Linux Raspberry 4.14.90-v7+

5.2 Testes

Os testes descritos nesta secção foram realizados utilizando a configuração acima mencionada. Estabilidade, tamanho de *firmware*, performance, alcance e consumo fazem parte da bateria de testes aplicados ao sistema. Com eles será possível identificar limitações e/ou vantagens no desenvolvimento de sistemas baseados em nRF24L01 com IP.

5.2.1 Ligação Direta versus Placa de Ligação

Na fase inicial de desenvolvimento do sistema de testes, a abordagem inicial passou por ligar o rádio nRF24L01+ diretamente ao microcontrolador Arduino. Essa abordagem apresentou inúmeros problemas. A importância dos componentes que compõem a placa de ligação fazem toda a diferença para a continuação desta investigação. Sem ela não seria possível realizar todos os testes seguintes.

De forma a perceber melhor o benefício desta placa é necessário analisar de perto os seus componentes.

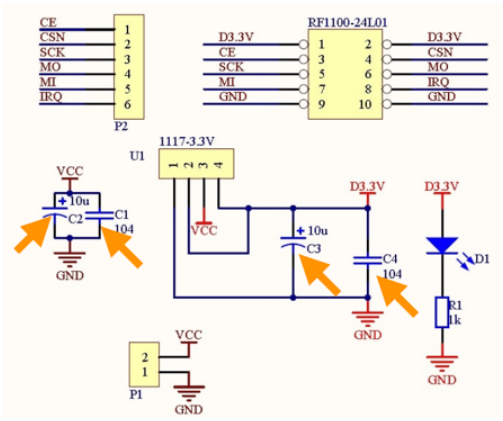


Figura 58 - Esquema elétrico da placa de ligação nRF24L01

Na Figura 58 é possível verificar a existência de quatro condensadores de *bypass*. Estes têm diferentes valores de forma a filtrar ruído proveniente do sistema de alimentação, o que, com diferentes valores permitem filtrar altas e baixas frequências.

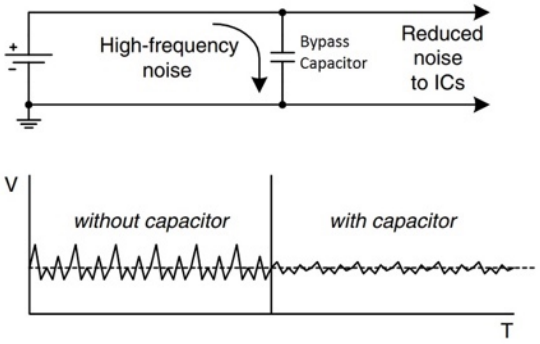


Figura 59 - Exemplo da utilização de condensador de bypass

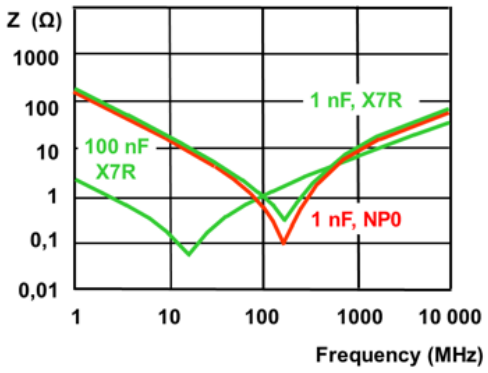


Figura 60 - Exemplo de utilização de múltiplos condensadores de bypass

Um circuito de rádio frequência como o desenvolvido no âmbito desta investigação é muito sensível ao ruído da fonte de alimentação. A menos que seja alimentado por baterias, existe uma grande hipótese do ruído existente no circuito possa estar associado à fonte de energia ou em alguns casos à placa de circuito impresso, caso esta esteja mal arquitetada.

A Nordic, fabricante do nRF24L01+, recomenda pelo menos um condensador de 10 µf entre o VCC e o GND e que este esteja o mais aproximado do rádio nRF.

Uma outra forma mais prática será utilizar a placa de ligação, que para além de ter uma melhor filtragem ainda traz um regulador de tensão linear que permite alimentar o circuito com tensões superiores a 3.6V.

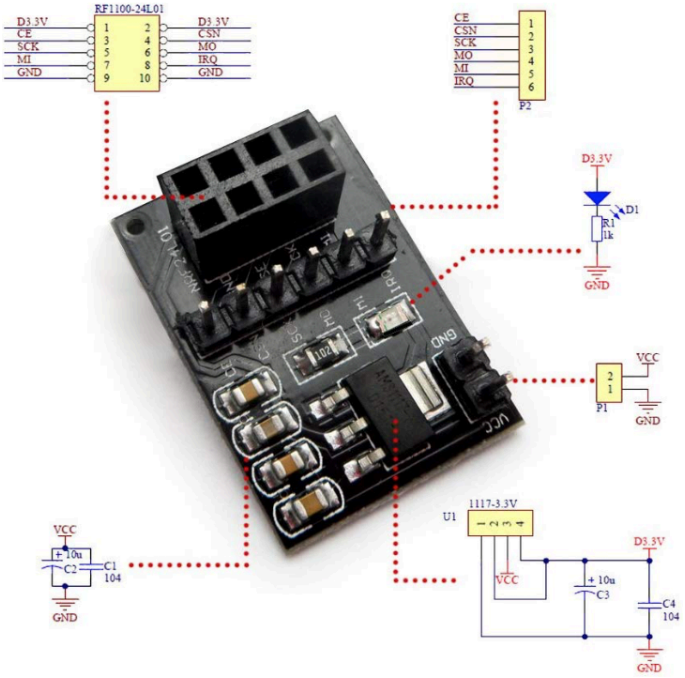


Figura 61 - Placa de Ligação para nRF24L01

A placa de ligação resolveu as seguintes anomalias:

- Incapacidade de ligação à rede IP Mesh;
- Desconexão aleatória;
- Erros na comunicação;
- Latência;
- Instabilidade.

É possível analisar na Figura 62 Figura 63 em detalhe a redução do *Ripple* presente na alimentação: a azul temos a leitura com o Adaptador que contem os condensadores para filtragem, a

amarelo a leitura com alimentação sem adaptador. Todo o ruído causado perturba a comunicação rádio. Na Figura 62 são apresentados dados com o Nó alimentado por um USB de computador e na Figura 63 é detetada uma melhoria considerável na redução de Ruído já que nesse cenário o Nó está ligado a uma bateria de 2200mAh.

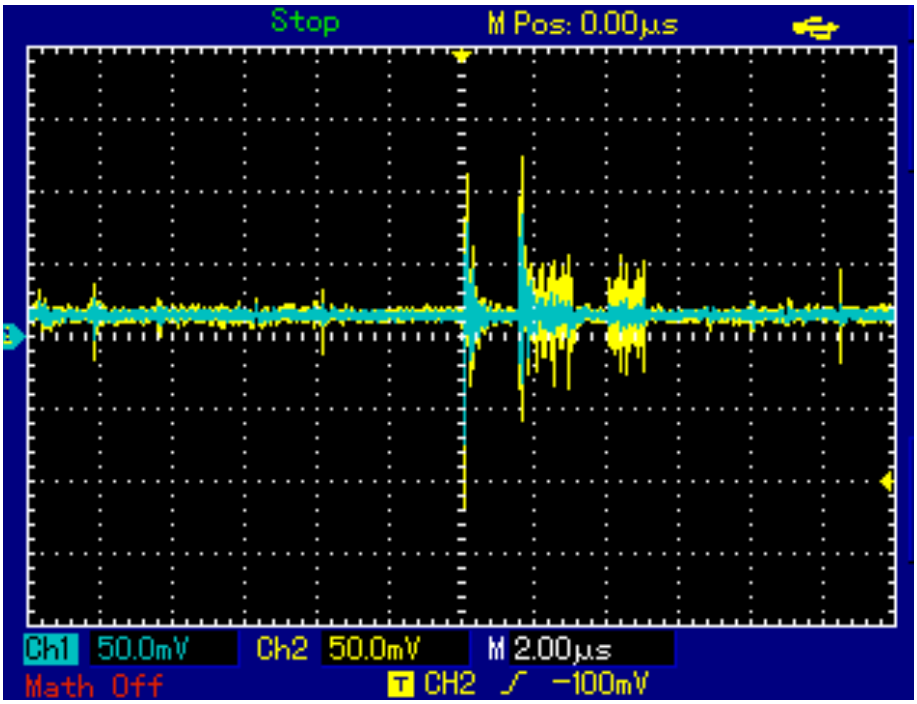


Figura 62 - Alimentação Via USB Com e Sem Adaptador

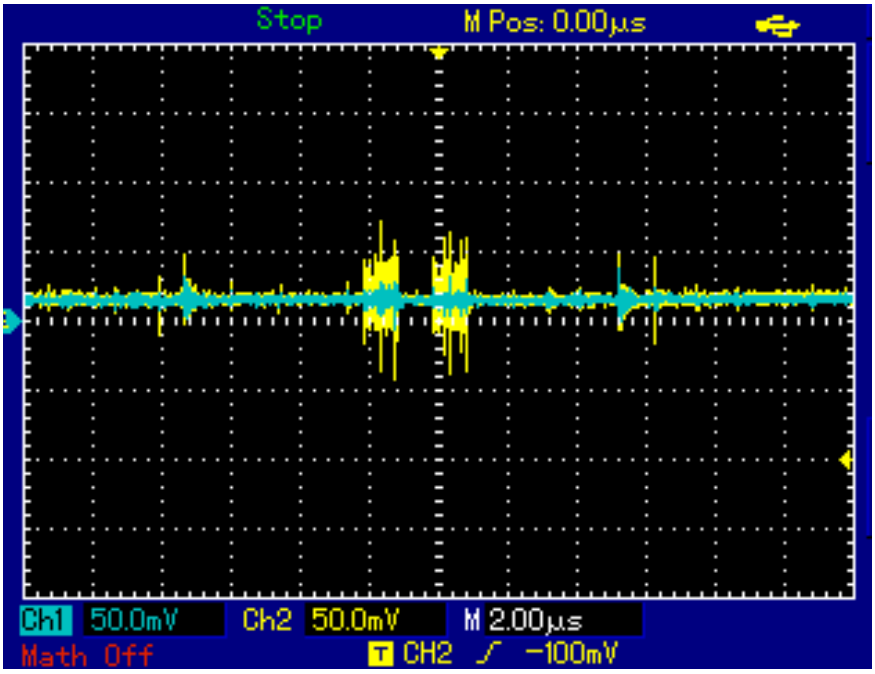


Figura 63 - Alimentação a Bateria Com e Sem Adaptador

O teste mostrou que com o adaptador e com uma alimentação a bateria o Nó deverá ter uma alimentação mais estável reduzindo assim erros na comunicação rádio.

5.2.2 Dimensão do *firmware* Sem IP versus com IP

No que diz respeito a sistemas embebidos, o espaço de memória disponível para o *firmware* é sempre uma limitação. Microcontroladores como o Arduino Nano ou Uno apenas disponibilizam 32Kb de Memoria Flash e se utilizarmos o *Bootloader* do Arduino ficamos apenas com 30Kb para gravar o *firmware*.

A Figura 64 apresenta um gráfico com quatro *firmwares* diferentes: a azul as implementações sem IP e a verde com IP.

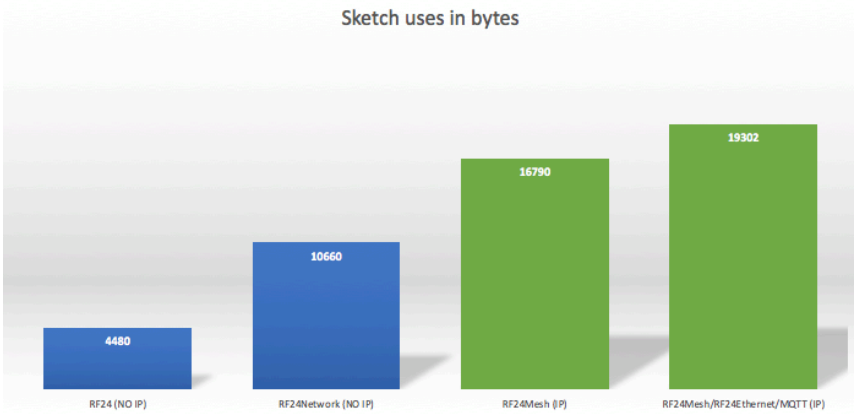


Figura 64 - Tamanho de firmware nRF24

A primeira coluna RF24 estabelece uma comunicação entre um ou mais Nós e utiliza um identificador para cada nó ficando à escuta de pacotes que tenham o seu identificador.

Como vantagem ocupa apena 4480 bytes, ou seja 14% da memoria do Arduino Nano/Uno, os pacotes têm cabeçalhos muito pequenos o que permite realizar comunicações rápidas. A desvantagem está em não ser possível comunicar diretamente de um computador para um determinado Nó fazendo uso do seu endereço. Sendo assim o Nó também não consegue comunicar diretamente para um serviço MQTT ou HTTP.

```
#include "RF24.h"
/****************** Config ***********************/
/***      Set this radio as radio number 0 or 1         ***/
bool radioNumber = 0;
byte addresses[][6] = {"1Node","2Node"};
// Used to control whether this node is sending or receiving
bool role = 0;

void setup() {
  // Open a writing and reading pipe on each radio, with opposite addresses
  if(radioNumber){
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1,addresses[0]);
  }else{
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1,addresses[1]);
  }
  // Start the radio listening for data
  radio.startListening();
}
```

Figura 65 - Configuração do Sketch RF24 (Sem IP)

Na segunda coluna o espaço ocupado passa para mais do dobro face a primeira implementação. No *sketch* é utilizada apenas uma das bibliotecas disponibilizados pelo autor TMRh20, a RF24Network, em conjunto com a utilizada no *firmware* anterior. Neste exemplo é possível comunicar com todos os Nós dentro da rede RF24Network utilizando os endereços definimos pela topologia apresentada na Figura 36.

Esta solução apresenta uma vantagem significativa face á anterior, já que, com ela é possível "Pingar" outros Nós e perceber se estão vivos. A desvantagem é que requer cabeçalhos com mais informação para organizar os pacotes dentro da rede.

```
/**
 * Send an 'N' message, the active node list
 */
bool send_N(uint16_t to)
{
  RF24NetworkHeader header(/*to node*/ to, /*type*/ 'N' /*Time*/);

  printf_P(PSTR("---------------------------------\n\r"));
  printf_P(PSTR("%lu: APP Sending active nodes to 0%o...\n\r"),millis(),to);
  return network.write(header,active_nodes,sizeof(active_nodes));
}

/**
 * Handle a 'T' message
 * Add the node to the list of active nodes
 */
void handle_T(RF24NetworkHeader& header){

  unsigned long message;
  network.read(header,&message,sizeof(unsigned long));
  printf_P(PSTR("%lu: APP Received %lu from 0%o\n\r"),millis(),message,header.from_node);


  if ( header.from_node != this_node || header.from_node > 00 )
    add_node(header.from_node);
}
```

Figura 66 - Exemplo de escrita e leitura de cabeçalhos numa rede RF24Network

Por fim, as duas últimas colunas do gráfico da Figura 64 apresentam duas soluções baseadas em IP e fazem uso das duas bibliotecas utilizadas nos exemplos anteriores. A primeira coluna a verde, com um tamanho 16790 bytes, que se reflete em 54% da memoria flash do Arduino Nano/Uno, uma vez que implementa uma rede em Malha com recurso á biblioteca RF24Mesh disponibilizada pelo autor das anteriores.

Agora os Nós que compõem esta rede têm associado um endereço IP, abrindo assim portas para todo o universo das redes IP, sendo agora possível enviar e receber dados diretamente da Internet ou de qualquer computador ligado á mesma rede Local para o Nó nRF24.

Com base no exemplo anterior, o último *firmware* faz uso direto do potencial de uma rede IP, acrescentado a este, a biblioteca RF24Ethernet e a PubSubClient, para passar a ser possível enviar e receber mensagens diretamente de um *broker* MQTT. Neste caso, o Nó está ligado ao Mosquitto MQTT.

Foi possível através dos parágrafos anteriores perceber que uma implementação IP permite simplificar todo o processo de comunicação entre Nós nRF24L01+ e os computadores ligados à Internet ou a uma rede local, tendo como desvantagem os quase 20000 bytes que a solução requer para funcionar, deixando assim apenas cerca de 30% de memoria ROM e 32% de memoria SRAM disponível num Arduino Nano/Uno para implementações de sensores ou atuadores ligados a este microcontrolador. A solução para colmatar esta desvantagem passaria pela utilização de um microcontrolador com mais memoria flash.

```
#include <SPI.h>
#include <RF24.h>
#include <RF24Network.h>
#include <RF24Mesh.h>
#include <RF24Ethernet.h>
#include <PubSubClient.h>

RF24 radio(7,8);
RF24Network network(radio);
RF24Mesh mesh(radio,network);
RF24EthernetClass RF24Ethernet(radio,network,mesh);

IPAddress ip(10,10,3,  60);
IPAddress gateway(10,10,3,1); //Specify the gateway in case different from the server
IPAddress server(10,10,3,1);
EthernetClient ethClient;
PubSubClient client(ethClient);

void setup(){
  client.setServer(server, 1883);//MQTT CONNECTION
  client.setCallback(callback);
  Ethernet.begin(ip);
  Ethernet.set_gateway(gateway);
  mesh.begin();
  clientID[13] = ip[3] + 48; //Convert last octet of IP to ascii & use in clientID
}
```

```
Done Saving.

Sketch uses 19174 bytes (62%) of program storage space. Maximum is 30720 bytes.
```

Figura 67 – Exemplo de implementação de rede IP e MQTT em RF24

5.2.3 Performance de Início de Sistema versus MQTT

O arranque do sistema é algo inevitável e esta rotina deve ser realizada o mais rapidamente possível. Um arranque rápido é benéfico porque o dispositivo fica pronto para realizar as tarefas programadas. Por outro lado, numa abordagem em que o dispositivo adormece após realizar todo o fluxo e acorda por intermédio de uma interrupção externa ou agendada, durante este período de preparação está a desperdiçar energia.

Com este teste pretende-se verificar o acréscimo de latência no arranque do sistema ao utilizar uma ligação IP.

Existem dois passos importantes que devem ser comparados. O primeiro passo diz respeito `ligação do nó à rede nRF24 Mesh. Neste ponto ainda não existe qualquer referência ao IP e podemos comparar este passo a uma simples ligação ponto a ponto via rádio No segundo passo é realizada uma ligação via IP ao *broker* Mosquitto de forma a ser enviada uma mensagem com 32 bytes. É espectável que este passo demore mais tempo, pois terá de ser estabelecida uma ligação e após sucesso então será possível enviar os dados. O processo interno de envio de uma mensagem MQTT requer cerca de 5 passos que podem ser analisados abaixo na Figura 68.

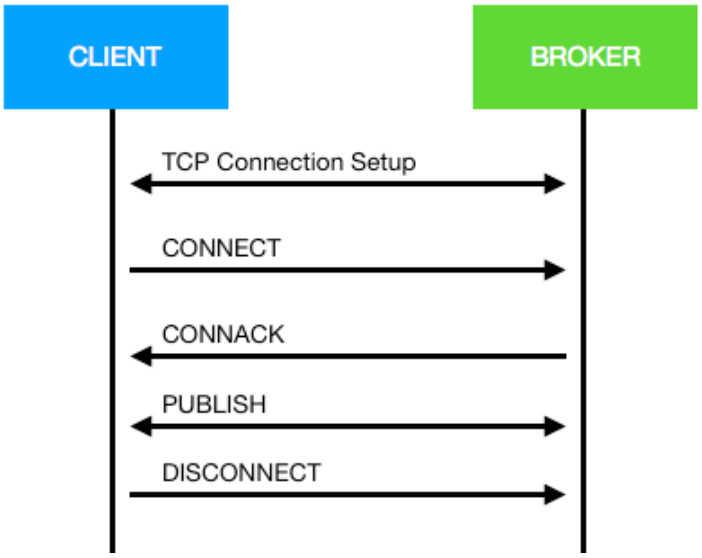


Figura 68 - Diagrama MQTT CONNECT e PUBLISH

A arquitetura do fluxo para este teste é mostrada na Figura 69 e tem três fases importantes. Na primeira fase, assim que o microcontrolador é energizado, é guardado um tempo de referência. Assim que ele estabelece a comunicação Mesh, o tempo decorrido até esse passo é novamente

guardado. Assim que a conexão com o *broker* MQTT é realizada (caso exista N tentativas de ligação o tempo acumula) é guardado também o tempo decorrido. Por fim é então enviada a mensagem MQTT com os dois tempos.

Figura 69 - Fluxo de Inicialização do Nó

Das onze vezes que o teste foi repetido as mensagens chegaram sempre com sucesso ao *broker*, o Node-RED persistiu cada uma delas na base de dados InfluxDB e foi criado um gráfico no Grafana. Os tempos entre repetições, ligação à rede Mesh e ligação IP ao *broker* Mosquitto podem ser analisados no gráfico da Figura 70.

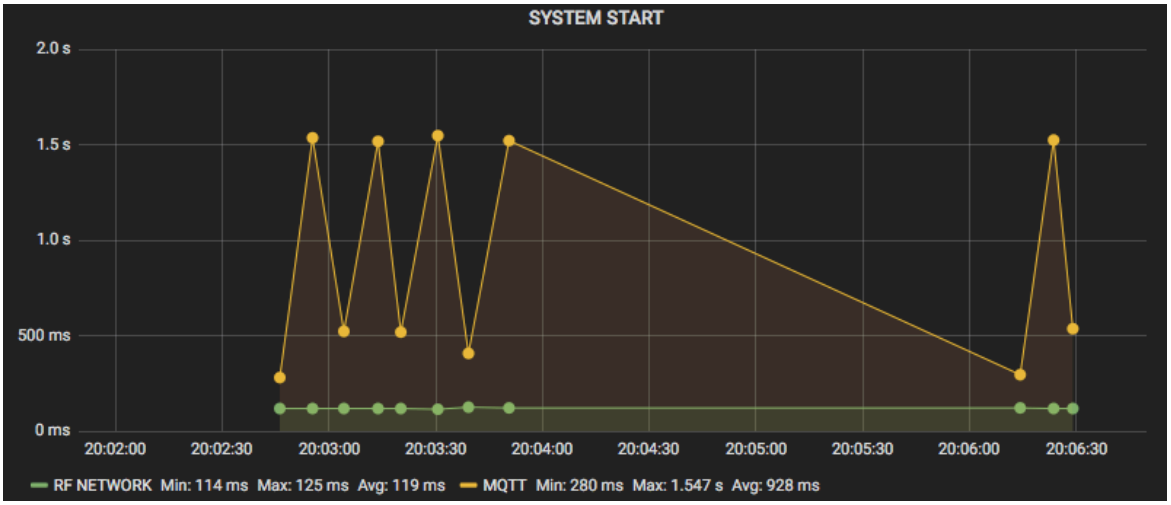


Figura 70 - Gráfico de tempos de inicialização do Nó

Acima na Figura 70 são apresentados dois tipos de tempos. A verde o tempo que o Nó demora a ficar visível na rede Mesh e a amarelo temos a latência de ligação ao *broker* MQTT via IP. Ao considerar apenas o tempo de aprovisionamento na rede nRF, em 11 repetições teve uma média de 119ms. Considerando o processo completo, que consiste também numa ligação ao *broker*, o tempo mínimo é de 280ms com uma média de 928ms.

A latência detetada em alguns casos na ligação ao MQTT está relacionada às várias tentativas de conexão. A incapacidade de ligar logo no primeiro pedido pode estar relacionada com interferências rádio existentes no meio forçando o nRF24 a ativar o *ShockBurst.*

5.2.4 Consumo energético IP versus Não IP

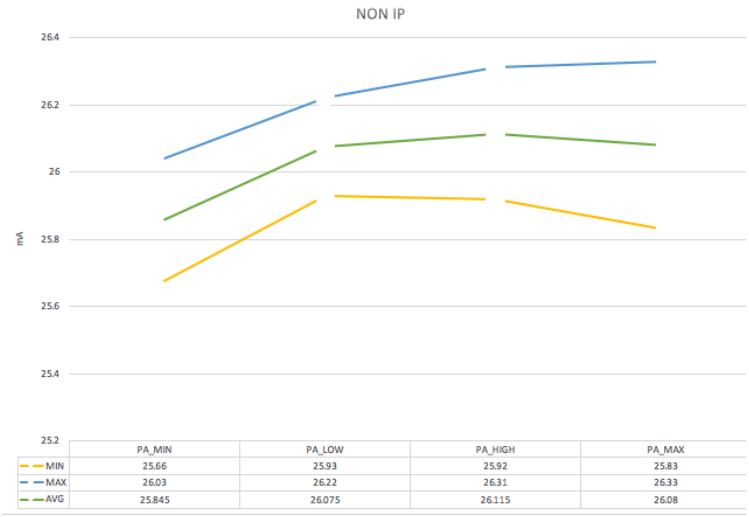
Este teste serve para detetar se existe um aumento ou redução no consumo ao utilizar a biblioteca IP em vez do envio da mensagem sem qualquer tipo de protocolo extra ao RF24. O *Datasheet* do nRF24L01+ [33] menciona que TX consome 11.3mA e RX 13.5mA. Apenas será medido o consumo do rádio sem influencia do microcontrolador.

Para a medição foi utilizado um multímetro VICTOR 70C. É conhecido que esta não é a melhor instrumentação para medir o consumo real do rádio, no entanto os valores base do mesmo servirão nos dois testes em que apenas o *firmware* será trocado, considerando assim que qualquer erro na medição será igual nos dois cenários.

Nos gráficos da Figura 71 e Figura 72 podemos observar os consumos mínimos, máximos e médios dos dois *firmware*s. O Primeiro firmware implementa uma rede Mesh com IP e envia uma mensagem "hello world" através de MQTT, o segundo utiliza apenas a rede RF24 para enviar a mesma mensagem sem protocolo IP nem MQTT e ambos fazem envio com intervalos de 1 segundo durante 1 minuto. O tamanho do pacote da primeira mensagem com IP e MQTT é maior. Devido a isso requer mais tempo para ser enviada o que se pode traduzir num possível aumento de energia.

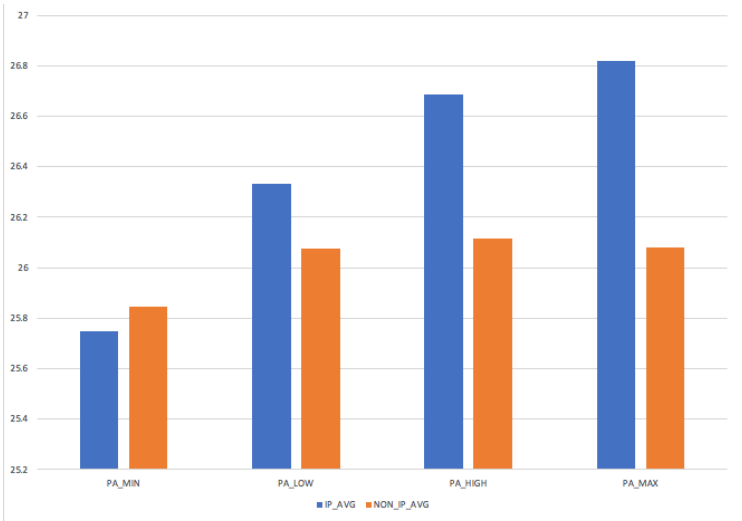
Durante o teste foram efetuadas quatro medições, onde em cada uma delas foi parametrizada a potência do rádio de PA_**MIN** -18dBm, PA_**LOW** -12dBm, PA_**HIGH** -6dBm até PA_**MAX** 0dBm. Este parâmetro influencia a potência de sinal, no entanto em qualquer um deles os Nós estavam separados por 1 metro de distância o que permitiu existir sempre uma comunicação bem-sucedida. O consumo sem estar a enviar mensagem é em média de 16,4mA em ambas as soluções.

Figura 71 - Medição de Consumo dos diferentes modos com IP



	PA_MIN	PA_LOW	PA_HIGH	PA_MAX
MIN	25.66	25.93	25.92	25.83
MAX	26.03	26.22	26.31	26.33
AVG	25.845	26.075	26.115	26.08

Figura 72 - Medição de Consumo dos diferentes modos sem IP



	PA_MIN	PA_LOW	PA_HIGH	PA_MAX
IP_AVG	25.75	26.33	26.685	26.82
NON_IP_AVG	25.845	26.075	26.115	26.08

Figura 73 - Comparação de consumo entre IP e Sem IP

O gráfio da Figura 73 descreve a diferença entre as duas abordagens: Com e sem IP. A abordagem com IP em modo PA_**MIN** conseguiu ser um pouco melhor que a sem IP, nos outros modos em média a versão sem IP tem uma eficiência energética melhor em cerca de 0.52mA por mensagem. Este valor pode ser considerado ou não dependendo da aplicabilidade da solução. Numa solução que em será mais vantajoso ter uma rede IP com um custo pouco significativo no consumo energético, ou numa solução *Wake-Up/Sleep* com poucas iterações diárias, a solução com IP continua a ser mais vantajosa.

5.2.5 Distancia IP versus Sem IP

O teste de distância tem como objetivo perceber se a utilização do IP reduz a distância entre rádios em termos de comunicação.

Para este teste foram testadas duas configurações. A configuração com IP é composta pela *Gateway* nRF24L01+ e um Nó nRF24L01+, enquanto que a configuração sem IP utiliza os mesmos dispositivos com a diferença no *firmware*, que apenas vai enviar uma mensagem para o outro Nó e este responde sempre que a receção é bem-sucedida.

No primeiro caso, com IP, vamos lidar com uma conexão MQTT para enviar a mensagem e o procedimento passa por estabelecer a ligação MQTT, perto da *Gateway* e de seguida ir afastando o Nó da mesma até parar de comunicar. Assim que este deixa de comunicar, o nó é fixado no local e aguarda-se um minuto para perceber se ele consegue estabelecer a ligação. Caso consiga, continuamos a aumentar a distância, caso não consiga reduzimos até retomar a comunicação e o local é assinalado como um máximo.

Na Figura 74 e Figura 75 o ponto inicial denominado de 0 com a cor verde. Pontos a laranja são locais onde perdeu a ligação por breves instantes, mas recuperou no tempo máximo de 1 minuto. Pontos a vermelho definem o local onde perdeu por completo a comunicação, o trajeto será assinalado com tracejado amarelo no mapa. Para efeitos de medição apenas é considerada a distância em linha reta até ao último ponto em que o Nó conseguiu comunicar sem interrupções.

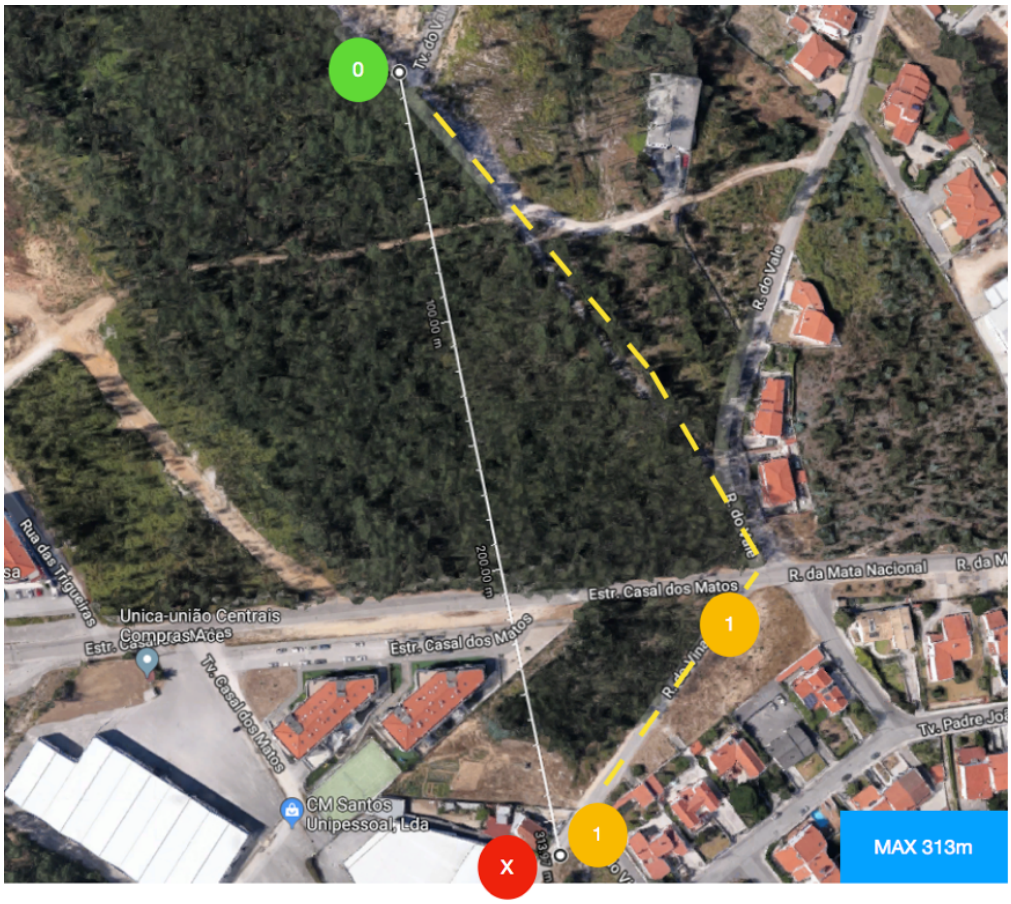


Figura 74 – Teste 1 Distância máxima com comunicação nRF24L01+, IP e MQTT

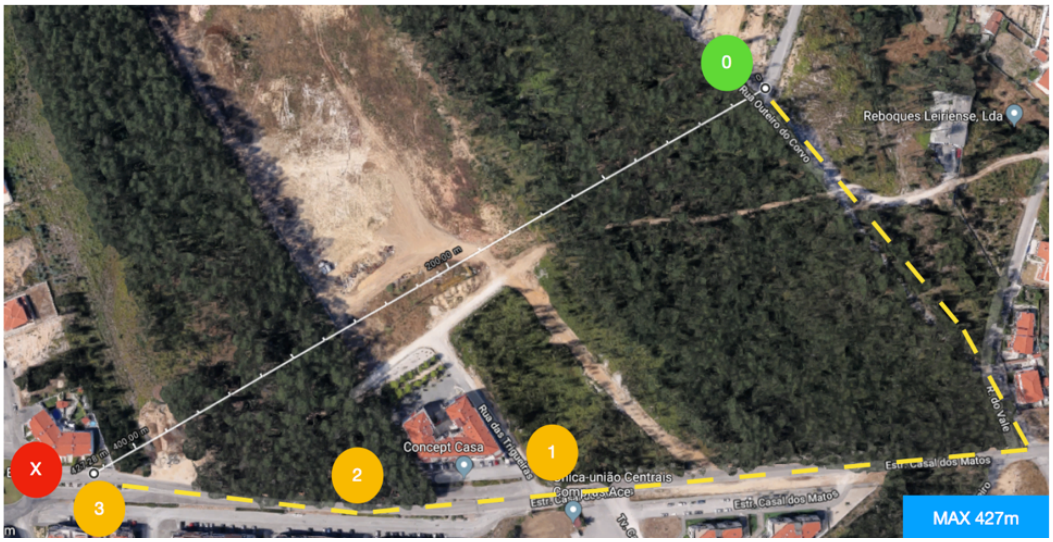


Figura 75 - Teste 2 Distância máxima com comunicação nRF24L01+, IP e MQTT

Os resultados dos testes apresentados na Figura 74 e Figura 75 foram realizados com os Nós alimentados a baterias de 2100mA, a uma distância do solo de 1.5 metros, num ambiente rural com

temperaturas a rondar os 22ºC e uma humidade do ar em cerca de 76%. A potência do rádio foi configurada para debitar o máximo de sinal, ou seja, PA_**MAX** 0dBm, sob estas condições, o nRF24L01+ demonstrou ser muito sensível a edifícios e a vegetação densa, como esperado, dada a gama de frequências utilizada. Apesar disso, foi possível comunicar até 427 metros com vegetação de baixa densidade, mas com linha de vista entre antenas.

Foi percetível que sempre que o Nó ficava atrás de um edifício a latência aumentava e o sinal era perdido. Um caso desses foi verificado no segundo teste da Figura 75 em que no ponto 1 o sinal perdeu-se e foi retomado rapidamente até ao ponto 2.

O Nó que estava nos 427 metros a enviar mensagem a cada segundo manteve sempre a cadência sem atrasos.

O mesmo percurso foi realizado, mas desta vez com o *firmware* alterado para enviar mensagens apenas utilizando a biblioteca RF24, sem IP nem MQTT.

A abordagem consistiu em ter um Nó fixo no local onde estava instalada a *Gateway* e ter outro Nó móvel que recebe a mensagem do Nó fixo e volta a enviar como confirmação. Durante o teste foi analisado o tempo de voo da mensagem como também se era possível atingir uma maior distância. Entende-se por tempo de voo, o tempo de envio de uma mensagem a partir do Nó emissor, até ao Nó recetor responder com uma confirmação, o tempo é então determinado assim que o emissor receber a confirmação (ACK).

Máximo tempo de Voo	0,23 segundos
Média de tempo de Voo	0,10 segundos
Desvio Padrão	0.09

Tabela 5 - Estatística de tempos de Voo sem IP

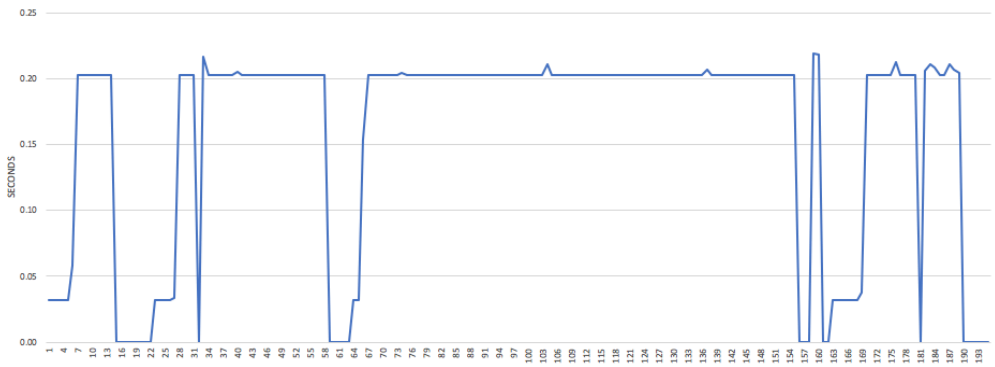


Figura 76 – Teste 2, Gráfico de tempos de Voo e perda de Ligação

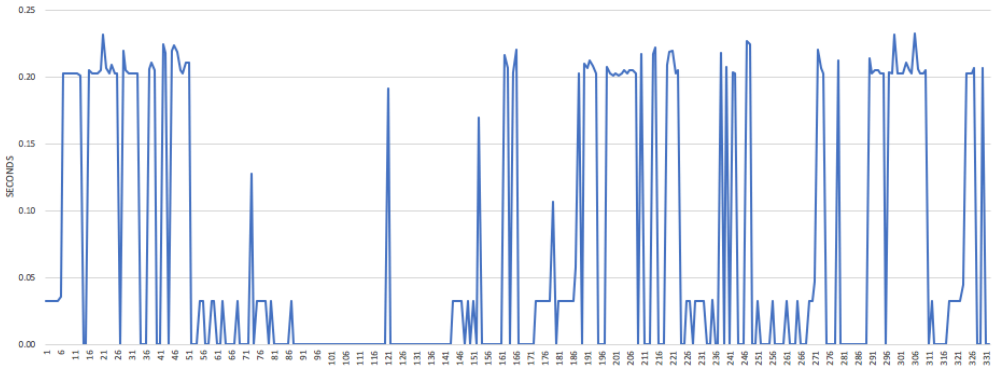


Figura 77 - Teste 3, Gráfico de tempos de Voo e perda de Ligação

Os resultados foram idênticos tanto na comunicação IP como sem IP o que leva a concluir que o facto de estarmos a implementar protocolos mais complexos, não se refletiu numa redução da distância.

O Nó que estava a comunicar de forma mais simples possível não se destacou em nada, e a perda de ligação pode ser analisada no gráfico da Figura 76 e Figura 77. Sempre que existem pontos a 0 indica que o Nó não conseguiu receber resposta do outro.

O primeiro gráfico é relativo ao percurso da Figura 74 e os pontos onde perde ligação foram relativamente os mesmos detetados com ligação IP e MQTT. Já no segundo percurso, em que o Nó encontrou maior vegetação e edifícios, a perda de ligação aconteceu durante alguns momentos e voltou a ser restabelecida assim que o edifico foi contornado, ficando apenas vegetação entre os 2 Nós, mas com linha de vista entre antenas.

Por fim esta configuração não conseguiu ultrapassar nenhuma das metas impostas pela configuração mais complexa com IP e MQTT, ficando-se pelos mesmo 313 metros no percurso 1 e 427 metros no percurso 2.

Este tipo de resultado é altamente positivo. É certo que o excedente de código necessita de mais memoria Flash e RAM, como foi verificado no teste realizado no subcapítulo 5.2.2, no entanto a facilidade na comunicação e a simplicidade no tratamento de mensagens fazem com que seja uma escolha válida para soluções que outrora necessitavam de Wi-Fi para usufruir de todas estas vantagens que o IP e o MQTT oferecem ao nível da programação como da infra estrutura para tratar os dados.

5.2.6 Envio de mensagens MQTT sobre IP durante 48 horas

Este teste terá como objetivo analisar a estabilidade e fiabilidade do sistema bem como o tempo de recuperação em caso de falha na comunicação.

Será configurado um Nó para enviar uma mensagem de "Ping" por MQTT, sendo enviada com uma cadência de segundo a segundo durante 48 Horas.

O Nó será alimentado por uma bateria de 2100mA em constante carregamento. A escolha deste tipo de alimentação provém do resultado do teste realizado no subcapítulo 5.2.1, onde percebemos que com uma alimentação a bateria e com o adaptador de ligação era reduzido o Ruído na alimentação melhorando assim a comunicação. Por fim, o dispositivo será colocado a uma distância de 10 metros em linha de vista com a *Gateway* ligada ao Raspberry. Na mesma área de cobertura onde o teste está a ser realizado, estão 41 dispositivos Wi-Fi e 16 dispositivos ZigBee em constante funcionamento. É importante voltar a mencionar que todos estes dispositivos então a utilizar a mesma frequência de 2.4GHz o que pode causar assim ainda mais interferência no canal de transmissão.

Assim que a mensagem é enviada pelo Nó e chega à *Gateway*, esta é processada pelo Node-RED e persistida na base de dados InfluxDB com a data em que chegou á *Gateway*.

Apesar deste teste ser muito simples permite responder às seguintes questões:

- O Sistema consegue manter a cadência?
- O Sistema consegue recuperar de um erro sem intervenção humana?
- Quantas mensagens falharam?
- Qual a latência máxima ocorrida entre 2 envios?
- Quantas mensagens tiveram latência acima de 1 ou mais segundos?

No final do teste se for definida uma janela de tempo de 48h, é expectável existirem registadas no InfluxDB cerca de 172800 mensagens (48h*3600s).

Figura 78 - Resultados do teste de envio de mensagens em 48h

Na Figura 78 é possível observar dois gráficos desenvolvidos na ferramenta Grafana. Como filtro foi definida uma janela de tempo com início no dia 21 de Setembro às 22:05 até ao dia 23 pela mesma hora, o que resulta numa janela temporal de 48h. No primeiro gráfico observam-se os pontos gerados a cada mensagem. Em conjunto, estes pontos perfazem uma linha perfeita de início ao fim do teste. Caso tenham ocorrido falhas na comunicação, o dispositivo conseguiu recuperar rapidamente e chegar ao fim do teste.

No segundo gráfico é possível obter respostas mais discretas, mas que podem gerar novas questões. Foram registadas 172340 mensagens de 172800 esperadas.

Latência Média entre leituras	1	Segundo
Latência Máxima	32.99	Segundos
Desvio Padrão	0.11	
Latência Inferior a 1 segundo	57	Mensagens
Latência superior a 1,5 segundos	111	Mensagens
Latência superior a 2 segundos	108	Mensagens
Latência superior a 3 segundos	4	Mensagens
Total de mensagens esperadas	172 800	Mensagens
Total de mensagens enviadas	172 340	Mensagens
Falhas ou mensagens fora de tempo	460	Mensagens

Tabela 6 - Resultados estatísticos do teste de envio em 48h

Para uma análise mais detalhada foi subtraído ao tempo de cada mensagem o tempo da anterior. Com este cálculo é possível apurar a latência média e máxima nos 172340 envios. É possível ainda analisar latências com intervalos consideráveis. Por exemplo, intervalos inferiores a 1 segundo podem ser associados a uma retransmissão originando uma duplicação na mensagem ou fora de tempo.

Os resultados da Tabela 6 servem de base para futuros desenvolvimentos e cuidados a ter em sistemas mais críticos. Em 48h 460 mensagens não chegaram a tempo e o sistema apresenta assim 99,7% de fiabilidade, mantendo sempre uma média de envio de 1 segundo entre cada envio. Existiram 57 mensagens que chegaram com intervalos inferiores a 1 segundo, sendo que estas podem estar associadas a entregas fora de tempo.

Num sistema critico pode ser necessário tratar algum destes tipos de ocorrência, serializando as mensagens para evitar repetições, ou ignorar leituras fora de tempo.

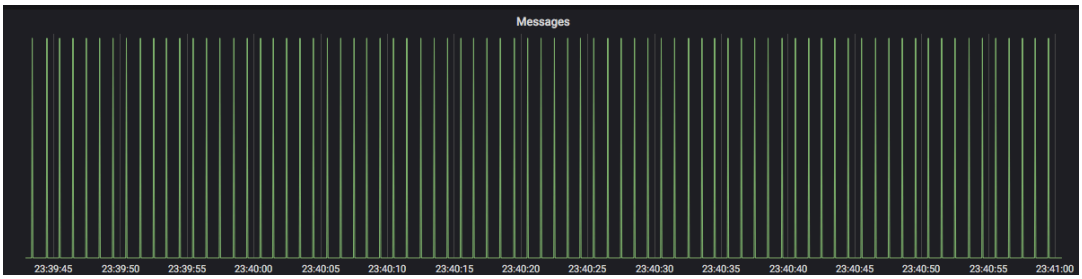


Figura 79 - Cadência de mensagens

Os resultados deste teste confirmaram a fiabilidade do nRF. Com base nos dados conclui-se que um dispositivo baseado em nRF24L01+ pode funcionar durante longos períodos enviando dados com uma cadência curta, paticamente sem falhas e mantendo em média o intervalo definido entre cada envio. O gráfico da Figura 79 ilustra a cadência de envios.

5.2.7 *Wake-Up / Sleep*, durante 5 minutos

Neste teste o Nó vai estar adormecido e será acordado por intermédio de uma interrupção por hardware a cada 30 segundos. Tem disponível uma janela de tempo de 20 segundos para se ligar ao sistema e enviar uma mensagem "*hello*" via MQTT através de IP. O sistema volta ao estado adormecido assim que enviar a mensagem ou caso tenha atingido o tempo máximo de 20 segundos.

Na Figura 80 é esquematizada a rotina padrão de funcionamento. Ao tempo verde o sistema acorda e tem 20 segundos para enviar a mensagem. Ao tempo vermelho o dispositivo adormece durante 10s e volta a repetir novamente todo o procedimento.

O teste foi realizado com o Nó alimentado a uma bateria de 10000mAh e posicionado a uma distância de 10 metros da *Gateway*.

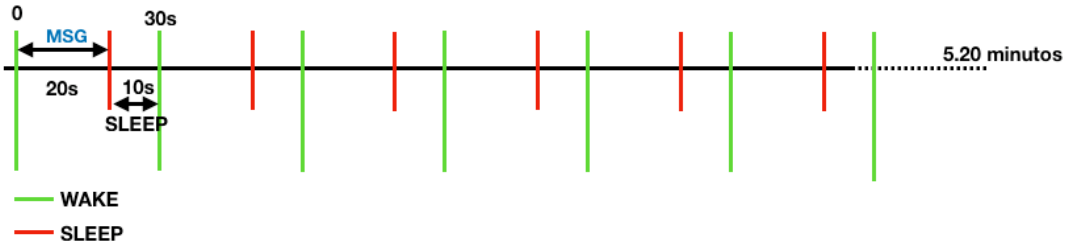


Figura 80 - Timeline WakeUp / Sleep

O Gráfico da Figura 81 mostra no tempo a chegada de cada mensagem. É possível perceber que das 11 iterações chegaram sempre mensagens, o que demonstra que o sistema ao acordar consegue estabelecer a ligação e enviar no tempo máximo de 20 segundos.

No subcapítulo 5.2.3 verificou-se que o sistema, no pior dos casos, poderia demorar até 1,5 segundos a estabelecer ligação com a rede, já neste teste verifica-se o impacto contabilizando o tempo de publicação/envio.

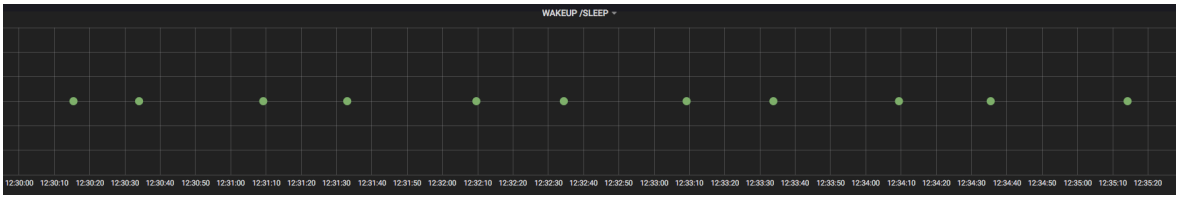


Figura 81 - Registo de mensagens enviadas após o Nó sair do estado SLEEP

WAKE	RECEIPT MESSAGE	TIME
12:30:00	12:30:15	0:00:15
12:30:30	12:30:33	0:00:03
12:31:00	12:31:09	0:00:09
12:31:30	12:31:33	0:00:03
12:32:00	12:32:09	0:00:09
12:32:30	12:32:34	0:00:04
12:33:00	12:33:09	0:00:09
12:33:30	12:33:33	0:00:03
12:34:00	12:34:09	0:00:09
12:34:30	12:34:35	0:00:05
12:35:00	12:35:14	0:00:14

Tabela 7 - Tempos de Mensagens WakeUp/Sleep

Mínimo	0:00:03
Máximo	0:00:15
Média	0:00:08
Desvio Padrão	4,76

Tabela 8 - Estatista de tempo de envio WakeUp/Sleep

Uma análise visual no gráfico da Figura 81 permite verificar que o padrão parece ser uniforme. Já na Tabela 7 é possível analisar, em detalhe, o tempo em que o Nó "acordou" até ao tempo em que a mensagem é registada e persistida pelo Node-RED. Os tempos variam a cada iteração.

A Tabela 8 esclarece algumas questões em relação à definição de tempo máximo para o envio de uma mensagem.

Em 11 iterações o Nó voltou a adormecer sempre antes de ter chegado ao limite dos 20 segundos. Este tipo de resultado é algo esperado num sistema alimentado por baterias, já que caso este consiga realizar as tarefas no menor tempo possível é traduzido em eficácia energética.

Este tipo de teste deve ser sempre realizado após ser definido o local onde será instalado o dispositivo. Numa rede composta por diversos Nós, cada um deles pode ter um limite máximo de envio diferente. Nunca deve ser desprezada a definição de um tempo máximo para o Nó voltar a

adormecer, caso contrário o Nó pode ficar a consumir energia desnecessariamente enquanto tenta infinitamente enviar a mensagem. Em alguns casos até pode nunca conseguir.

Janelas de tempo grandes devem ter em consideração o contexto da mensagem, isto porque quando a mensagem finalmente chegar pode já não fazer sentido. Por exemplo, na notificação de uma porta aberta, se for definido um tempo máximo de envio podemos receber a mensagem tarde de mais e já não fazer sentido porque a mesma já não está no estado em que a mensagem foi preparada. Neste caso devemos ter um tempo limite pequeno, e a cada tentativa de envio, deve ser considerado novamente o estado atual da porta.

5.2.8 Distancia Máxima com IP utilizando 3 Nós nRF24L01+ em Mesh

Com este teste será possível esclarecer 3 questões consideravelmente importantes:

- A biblioteca implementada pelo autor TMRh20 permite Mesh.
- Permite enviar em Mesh pacotes IP com mensagens MQTT?
- Qual a distância máxima conseguida com apenas 3 rádios em linha?
- A rede é estável?

Implementar a rede em Malha com os vários nRF24L01+ foi um desafio, já que o teste se tornou o mais complexo de todos.

Inicialmente pensou-se que com base na distância máxima de 427 metros conseguida no teste realizado no subcapítulo 5.2.5 bastaria colocar nesse mesmo local outro Nó e seria possível ampliar diretamente a rede. O mesmo não aconteceu, já que, assim que era ligado o novo Nó toda a rede parava de comunicar. Após alguma análise verificou-se que o Nó não conseguia estabelecer a ligação Mesh. A reduzir-se a distância para metade, foi possível ter comunicação em Malha com 2 Nós. Ao adicionar o terceiro Nó, a rede voltou a parar, mas após alguns minutos retomou o funcionamento em pleno, no entanto a distância máxima conseguida em malha utilizando 3 Nós e uma *Gateway* passou pouco mais da meta conseguida apenas com um Nó. É perfeitamente normal, na configuração de teste, existir este tipo de instabilidade, já que uma configuração que corre em Malha pressupõe diversos pontos de acesso para cada Nó. Tal não acontece neste teste. Os Nós estão propositadamente dispostos em linha reta tendo apenas um ponto de comunicação que consequentemente é um ponto critico de falha. Sendo assim, é possível apurar uma distância máxima na qual cada Nó Filho depende apenas de um Nó Pai.

Para conseguir atingir um máximo de 563 metros, foram necessárias várias tentativas até encontrar o ponto certo para cada Nó. Determinou-se, assim, que a melhor distância entre Nós é

dividir pela metade a distância atingida só com um Nó duplicando assim o custo dispositivo/distância.

Figura 82 - Distância máxima com rede em Malha a comunicar com IP e MQTT

Após várias iterações para tentar responder às questões inicialmente propostas foi possível apurar o seguinte.

A biblioteca desenvolvida permite funcionar em Malha, enviar pacotes IP e estabelecer comunicações com serviços como MQTT entre outros.

Foi possível chegar aos 563 metros, no entanto era expectável mais de o dobro, mas ao tentar ampliar a rede esta ficava instável deixando mesmo até de comunicar.

Verificou-se que a comunicação em ambientes *Indoor* ocorria sem problemas, foram colocados Nós pelas várias divisões onde o sinal anteriormente não chegava e conseguiu-se assim criar uma rede em Malha a operar durante várias horas sem qualquer falha, já no exterior os resultados alteraram-se, já que, era difícil manter a rede a funcionar de forma estável, visto que cada Nó não tinha vizinhos no seu redor para otimizar a comunicação.

O gráfico da Figura 83 mostra a estabilidade da rede Mesh em funcionamento no terreno. É possível verificar que, em média, cada Nó não consegue enviar mais de 20 mensagens sem falhas.

Na zona de testes estavam presentes vários postes de alta tensão e sendo uma zona urbana pode precipitar o aumento de interferências. As antenas utilizadas foram as originais que equipam o modulo nRF24L01+. Todos os dispositivos estavam a cerca de 1.5 metros do solo, fixados a uma cana seca com um elástico.

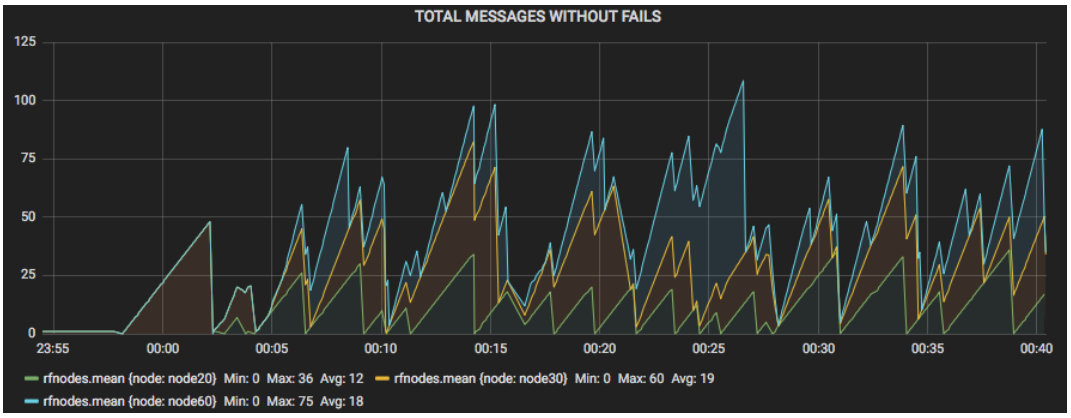


Figura 83 - Mensagens consecutivas sem falhas

Considerou-se os resultados deste teste plausíveis de serem melhorados, já que, passando os mesmos Nós para um ambiente *Indoor* onde cada Nó conseguia chegar a pelo menos 2 Nós Pai obtêm-se melhores resultados. No cenário apresentado na Figura 84 foram ainda mantidos ligados cerca de 41 dispositivos Wi-Fi e cerca de 16 dispositivos ZigBee em constante funcionamento. Com isto esperava-se precipitar ainda mais os Nós a mudarem de Pai com base no tempo de voo.

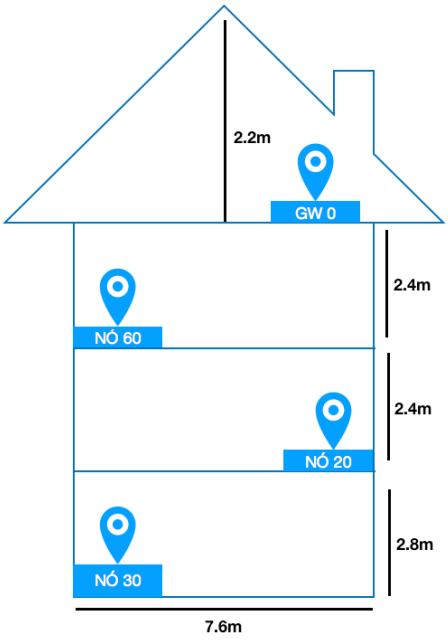


Figura 84 - Instalação Mesh Indoor

Para registar a mutação da rede foi criada uma mensagem MQTT constituída pelo endereço Mesh do próprio Nó e também o endereço Mesh do Nó Pai a que este estava ligado no momento de

publicação no tópico *heartbeat*. O resultado da mensagem gerada está ilustrado na Figura 85 e Figura 86.

Figura 85 - Mensagem MQTT do Nó 20 ligado diretamente à Gateway

Figura 86 - Mensagem do Nó 20 ligado ao Nó com o endereço Mesh 5

O gráfico da Figura 87 apresenta as várias mutações que aconteceram na rede, sendo o endereço 0 a *Gateway*. Podemos verificar, no tempo, que o Nó 20 esteve ligado por instantes ao Nó com o endereço Mesh 5 e passados alguns minutos ligou-se a outro Nó com o endereço Mesh 4, voltando-se a ligar-se à *Gateway* e novamente passado 4 Horas depois ligou-se novamente ao 5.

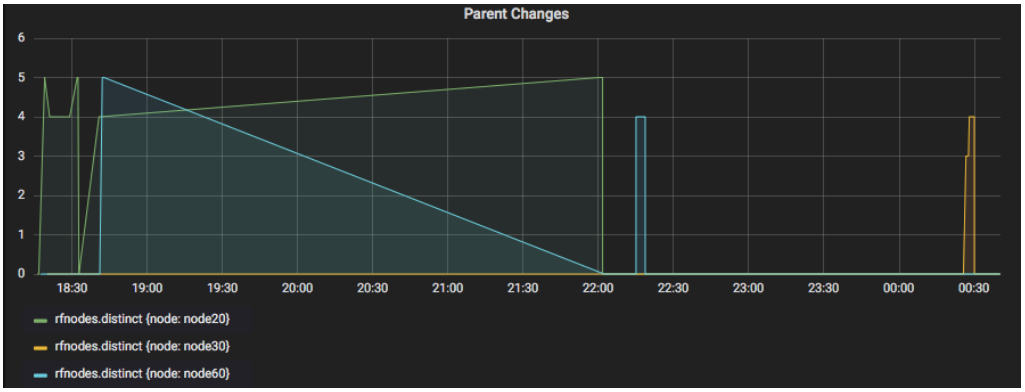


Figura 87 - Mutação da rede em Malha

Com base nos resultados obtidos, ao desenvolver um dispositivo que necessite de utilizar uma configuração em Malha, é aconselhável testes intensivos para verificar a estabilidade da rede face ao meio onde está inserida. Em habitações domésticas a rede apresentou resultados positivos, já no exterior verificou-se grande dificuldade em estabelecer ligação com os vários Nós em linha reta com apenas um Nó Pai. É aconselhável, numa arquitetura deste tipo, garantir pelo menos 2 Nós Pai por cada Nó filho.

6 Conclusão e trabalho futuro

A presente investigação teve como objetivo aumentar a informação baseada em testes reais, com base na viabilidade e fiabilidade em utilizar o protocolo IP com rádios nRF24L01+. De tal modo que foi necessário definir uma bateria de testes capaz de responder aos objetivos como também desenvolver alguns módulos protótipo para testar essa mesma bateria.

Durante o processo foram analisados vários projetos que fazem uso do nRF24L01+, no entanto nenhum deles mencionava a utilização de IP, nem as boas práticas para uma correta utilização do módulo de forma a obter a máxima performance. Todos eles passaram por tentar resolver um caso específico apresentado sob a forma de produto.

No caso desta investigação, o foco é virado para o engenheiro ou investigador que quer iniciar um projeto e precisa de obter respostas que lhe permitam avançar ou não para uma solução.

O trabalho desenvolvido passou por uma análise minuciosa de todo o processo de desenvolvimento de um dispositivo baseado em nRF24L01+, passando pelos problemas elétricos, tamanho de *firmware*, parametrização de potência de sinal e o seu impacto no consumo energético, até à fase de testes no terreno, onde foram consideradas distâncias máximas, obstáculos, fiabilidade, estabilidade e capacidade de recuperação em caso de falha.

A atualidade está agora direcionada para o inovador mundo da Internet das Coisas, daí fazer todo o sentido trazer o protocolo da Internet para o contexto desta investigação, ou seja, o protocolo IP.

A utilização deste protocolo em dispositivos com capacidade de comunicar com ou sem fios oferece uma enorme quantidade de vantagens. Com ele é possível comunicar com a grande maioria dos dipositivos e serviços hoje em dia oferecidos, é possível consultar uma página Web ou enviar uma mensagem através de protocolos sofisticados como o caso do MQTT. A prova disso é que os gigantes da tecnologia sem fios como o caso da ZigBee Alliance que outrora tinham desenvolvido protocolos proprietários sem recurso ao IP, estão a direcionar e a implementar esta forma de endereçamento que os liga diretamente a Internet.

A motivação de trazer um rádio com quase uma década novamente para o mundo competitivo da Internet das Coisas, fez com que cada teste realizado pudesse ser considerado para o desenvolvimento de novos produtos, no entanto foi necessário perceber se este seria capaz de funcionar em perfeitas condições, mantendo a performance e distâncias já conseguidas com a comunicação sem protocolo IP.

A fase de desenvolvimento de protótipo permitiu gerar documentação sólida sobre problemas que existem ao utilizar este rádio e a forma de os solucionar.

Percebeu-se que a arquitetura de desenho do circuito de alimentação influencia bastante a capacidade de comunicar do nRF24L01+. Para detetar esse tipo de anomalia e corrigir a mesma foi necessário recorrer a instrumentação específica, neste caso foi utilizado um osciloscópio para verificar se as modificações realizadas no segmento de alimentação melhoravam ou pioravam. No final passou por ser necessário a adição de alguns condensadores de filtragem e o rádio passou a funcionar sem quebras. Esta era já uma anomalia comum, no entanto não estava documentada de forma a perceber o real impacto de uma alimentação instável. Concluímos que sem esta adição de componentes o sistema funciona, mas nunca vai ser um sistema estável e em casos específicos em que o nRF24 necessita de mais potência pode mesmo deixar de comunicar.

Resolvido o problema de alimentação avançou-se para o desenvolvimento de 3 Nós e uma *Gateway*. Esta configuração permitiu testar enumeras soluções com e sem IP e devido a sua portabilidade e baixo consumo foi possível testar em diferentes cenários: ambientes rurais, florestais e domésticos.

O conjunto de bibliotecas desenvolvidas pelo programador canadiano TMRh20, foram um ponto fulcral em toda a investigação. De início ponderou-se em desenvolver toda a topologia de rede e camada protocolar, no entanto todo este trabalho já vinha sendo desenvolvido desde 2014 e disponibilizado sob licenciamento GPL-20 no GitHub. O autor contribui ativamente no melhoramento de cada uma das bibliotecas, documentado e dando suporte muito rapidamente. Todas as suas bibliotecas foram disponibilizadas e validadas pela comunidade Arduino, composta por uma massa gigante de programadores que garante a credibilidade e qualidade do código partilhado.

Durante o estudo verificou-se que o autor tinha ido ainda mais longe, desenvolvendo uma biblioteca que permitia criar uma rede em Malha em que continuaria a ser possível cada Nó da mesma ser endereçado em IP e comunicar diretamente com os serviços do mesmo protocolo. Esta funcionalidade passou a fazer parte da bateria de testes, pois concluiu-se que daria ainda mais vantagem ao nRF24. O facto é que a topologia em Malha funciona, em ambientes *indoor* é onde esta se destaca pelo facto de existir redundância até ao Nó para o qual se quer comunicar, o que aumenta a robustez da rede, no entanto em situações onde toda a topologia está limitada a um só caminho, verificou-se uma constante instabilidade e perda de comunicação.

Os resultados obtidos nos testes de consumo de enérgica com ou sem IP foram praticamente idênticos, o que leva à conclusão de que o impacto compensa a inovação.

Realizados todos os testes de laboratório, passou-se a uma implementação dos mesmos Nós no exterior, onde era necessário perceber se existia impacto na distância por estes estarem a utilizar IP. Durante os testes foram realizados dois percursos diferentes e repetidos diversas vezes para

garantir que os resultados não eram casuais. Foram testados os percursos com dois tipos de *firmware*, o primeiro enviava uma mensagem MQTT fazendo uso do IP, o segundo enviava uma mensagem sem IP e esperava que o outro rádio instalado num ponto fixo respondesse. Métricas como tempo de voo e latência entre envios foram registadas e percebeu-se que o protocolo não surtiu em qualquer efeito negativo. Em ambas as implementações a distância foi a mesma. Locais onde um perdera a comunicação por instantes o outro manifestava o mesmo sintoma.

Com base nos resultados obtidos conclui-se que a utilização do protocolo IP não inviabilizou o funcionamento normal do nRF24L01+ nem a sua implementação num microcontrolador com capacidades muito reduzidas. Pontos fortes já conhecidos do nRF24L01+ como comunicação a longas distâncias e baixo consumo, foram mantidas, permitindo assim desenvolver um dispositivo capaz de utilizar protocolos sofisticados que operam acima da camada IP tal como o MQTT, um protocolo que já deu provas que é o protocolo IoT da atualidade e que tem vindo a ser implementado por fabricantes como Philips, Xiaomi, Siemens, NOS, Vodafone entre muitos outros.

Não descartando a realização de mais testes ou melhoramentos nos já realizados, o estudo desenvolvido apresenta um conjunto de métricas válidas para investigadores ou engenheiros que se encontram numa fase de planeamento de dispositivos sem fio, agilizando assim todo o processo de orçamentação ou decisão.

Em termos de trabalho futuro seria interessante aumentar o número de Nós de forma substancial. Assim seria possível avaliar o nível de ruído causado por toda a rede, como determinar o impacto na performance na *Gateway*. É sabido pela especificação que quanto mais Nós são aprovisionados mais tráfego é gerado, seja este com *payload* do utilizador ou para controlo e manutenção da própria rede.

Durante o levantamento do Estado da Arte verificou-se a existência de um rádio Nordic também membro da família nRF24 o nRF24LE1 analisado na secção 2.1.4.1 na página nº 13. Este possui uma particularidade interessante, apesar de ser um pouco mais dispendioso, já inclui um microcontrolador embebido. Com ele é possível criar dispositivos ainda mais pequenos e possivelmente com uma redução substancial no consumo energético já não é necessário considerar o Arduino. No entanto, a curva de aprendizagem para o programar é maior que com o nRF24L01+, já que são programados recorrendo à *framework* Arduino. A interface com o computador terá de ser feita utilizando plataformas indicadas para tal e estas têm um custo substancial. O exemplo de uma dessas plataformas está ilustrado na Figura 88.

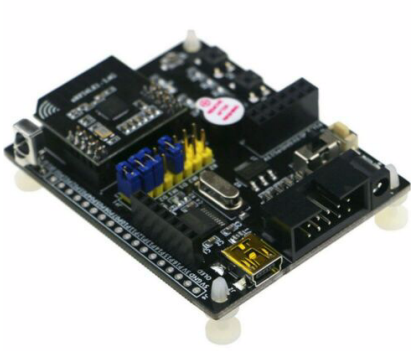


Figura 88 - Placa de Desenvolvimento para nRF24LE1

7 Referências

[1] E. Silva, L. Botelho, I. Santos, and G. Sanchez, "Computação Ubíqua – Definição e Exemplos," *Rev. Empreendedorismo, Inovação e Tecnol.*, 2015.

[2] L. Redazione, "Guglielmo Marconi," *Nuovo Cim.*, 1937.

[3] A. Lele, "Internet of things (IoT)," in *Smart Innovation, Systems and Technologies*, 2019.

[4] B. P. Santos *et al.*, "Internet das Coisas: da Teoria à Prática," *Minicursos SBRC-Simp*, 2016.

[5] H. C. Lee and K. H. Ke, "Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation," *IEEE Trans. Instrum. Meas.*, 2018.

[6] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," in *Wireless Personal Communications*, 2011.

[7] S. Alliance, "SigFox," *ONLINE*, 2016. .

[8] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT gateway: Bridging wireless sensor networks into Internet of Things," in *Proceedings - IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2010*, 2010.

[9] Nordic, "nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification v1.0," *Build. Res.*, 2008.

[10] Tmrh20, "RF24Network." [Online]. Available: http://tmrh20.github.io/RF24Network_Dev/index.html.

[11] Wikipedia, "Wireless communication." [Online]. Available: https://en.wikipedia.org/wiki/Wireless.

[12] I. S. Kryukov and A. I. Maksimkin, "Empowering the formation of wireless sensor network based on nRF24," 2017.

[13] M. Marconi and E. Lakatos, "Fundamentos de metodologia científica," *Ed. Atlas S. A.*, 2003.

[14] A. Andrade *et al.*, "Estudo de caso qualitativo," *Encontro Nac. da ANPAD - EnANPAD*, 2007.

[15] R. K. Yin, *Pesquisa qualitativa do início ao fim*. 2016.

[16] "Wireless Communication Technology Types and Advantages," 2019. [Online]. Available: https://www.watelectronics.com/different-types-of-wireless-communication-technologies/.

[17] Giovanni Di Sirio., "RTOS Concepts." [Online]. Available: http://www.chibios.org/dokuwiki/doku.php?id=chibios:articles:rtos_concepts.

[18] Nordic, "nRF52832." [Online]. Available: https://www.nordicsemi.com/-/media/Software-

and-other-downloads/Product-Briefs/nRF52832-product-brief.pdf?la=en&hash=2F9D995F754BA2F2EA944A2C4351E682AB7CB0B9.

[19] D. I. W. Headquarters, “ZigBee Wireless Mesh Networking.” [Online]. Available: https://www.digi.com/resources/standards-and-technologies/ZigBee-wireless-mesh-networking.

[20] D. I. Inc, “ANT Message Protocol and Usage.” Inc, Dynastream Innovations, p. 134, 2015.

[21] D. I. Inc., “ANT Message Protocol and Usage.” [Online]. Available: https://www.sparkfun.com/datasheets/Wireless/Nordic/ANT-UserGuide.pdf.

[22] A. S.-C. and H. L. Truong, “MQTT For Sensor Networks (MQTT-SN) Protocol Specification.” [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf.

[23] Z. Rahim, “Design and Implementation of a Low Cost Wireless Sensor Network using Arduino and nRF24L01+,” *Int. J. Sci. Res. Eng. Technol.*, 2016.

[24] L. S. Ferreira, “Construção e Integração de um Quadcopter numa Plataforma de Simulação de Missões Multi-Veículo.”

[25] C. Y. AKASAKA and W. G. V. DE AGUIAR, “Localização de pessoas e objetos em ambientes internos utilizando radio frequência,” 2017.

[26] T. M. de O. Crespo, “Comunicação sem fios DMX512,” 2018.

[27] TMRh20, “RF24Ethernet - TCP/IP over RF24Network,” 2015.

[28] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, “Developing IoT applications in the Fog: A Distributed Dataflow approach,” in *Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015*, 2015.

[29] “Node-RED,” 2019. [Online]. Available: https://nodered.org/about/.

[30] “Grafana,” 2019.

[31] P. Dix, “InfluxData (InfluxDB) | Time Series Database Monitoring & Analytics,” *InfluxData, Inc.*, 2017. .

[32] K. Ahmad and M. Ansari, “Hands-On InfluxDB,” in *NoSQL: Database for Storage and Retrieval of Data in Cloud*, 2017.

[33] N. Semiconductor, “nRF24L01+ Product Specification v1.0.” 2019.

Esta página foi intencionalmente deixada em branco

Anexos

Anexo 1 – Código fonte da *Gateway*

```
#include <RF24.h> //https://github.com/nRF24/RF24
#include <RF24Network.h> //https://github.com/nRF24/RF24Network
#include <RF24Mesh.h> //https://github.com/nRF24/RF24Mesh
#define SLIP_END     0300
#define SLIP_ESC     0333
#define SLIP_ESC_END 0334
#define SLIP_ESC_ESC 0335
#define UIP_BUFFER_SIZE MAX_PAYLOAD_SIZE
static uint16_t len, tmplen;
static uint8_t lastc;
HardwareSerial *slip_device;
RF24 radio(7, 8);
RF24Network network(radio);
RF24Mesh mesh(radio, network);
uint8_t slip_buf[UIP_BUFFER_SIZE]; // MSS + TCP Header Length
uint32_t active_timer =0;

void networkToSLIP();
void slipdev_char_put(uint8_t c){
 slip_device->write((char)c);
}

uint8_t slipdev_char_poll(uint8_t *c){
 if (slip_device->available()) {
   *c = slip_device->read();
   return 1;
 }
 return 0;
}

void slipdev_send(uint8_t *ptr, size_t len){
 uint16_t i;
 uint8_t c;
 slipdev_char_put(SLIP_END);
 for(i = 0; i < len; ++i) {
   c = *ptr++;
   switch(c) {
   case SLIP_END:
     slipdev_char_put(SLIP_ESC);
     slipdev_char_put(SLIP_ESC_END);
     break;
   case SLIP_ESC:
     slipdev_char_put(SLIP_ESC);
     slipdev_char_put(SLIP_ESC_ESC);
```

```
      break;
    default:
      slipdev_char_put(c);
      break;
    }
  }
  slipdev_char_put(SLIP_END);
}

uint16_t slipdev_poll(void){
  uint8_t c;
  if(slip  device->available()){
  while((uint8_t)slipdev_char_poll(&c)) {
    switch(c) {
    case SLIP_ESC:
      lastc = c;
      break;
    case SLIP_END:
      lastc = c;
      len = min(len,UIP  BUFFER  SIZE);
      tmplen = len;
      len = 0;
      return tmplen;
    default:
      if(lastc == SLIP_ESC) {
  lastc = c;
  switch(c) {
  case SLIP_ESC_END:
    c = SLIP_END;
    break;
  case SLIP_ESC_ESC:
    c = SLIP_ESC;
    break;
  }
      } else {
  lastc = c;
      }

      slip_buf[len] = c;
      ++len;

      if(len > UIP_BUFFER_SIZE) {
  len = 0;
      }

      break;
    }
  }
```

```
 }
  return 0;
}

void slipdev_init(HardwareSerial &dev){
  lastc = len = 0;
  slip_device = &dev;
}

void networkToSLIP(){
    RF24NetworkFrame *frame = network.frag_ptr;
    size_t size = frame->message_size;
    uint8_t *pointer = frame->message_buffer;
    slipdev_send(pointer, size);
}

void setup() {
  Serial.begin(115200);
  mesh.setNodeID(0);// Set this to the master node (nodeID 0)
  mesh.begin();
  slipdev_init(Serial);// Use the serial port as the SLIP device

}

void loop() {
  if(millis() > 10000){
    mesh.DHCP();
  }
  while(network.available()){
    RF24NetworkHeader header;
    network.read(header,0,0);
  }
  if(mesh.update() == EXTERNAL_DATA_TYPE) {
    networkToSLIP();
  }

  uint16_t len;
  if( (len = slipdev_poll()) > 0 ){
    if(len > MAX_PAYLOAD_SIZE){ return; }
    RF24NetworkHeader header(01, EXTERNAL_DATA_TYPE);
    uint8_t meshAddr;
    uint8_t lastOctet = slip_buf[19];
    if ( (meshAddr = mesh.getAddress(lastOctet)) > 0) {
      header.to_node = meshAddr;
      network.write(header, &slip_buf, len);
    }
  }
```

```
}
```

Anexo 2 – Código fonte do Nó

```
#include <RF24.h> //https://github.com/nRF24/RF24
#include <RF24Network.h> //https://github.com/nRF24/RF24Network
#include <RF24Mesh.h> //https://github.com/nRF24/RF24Mesh
#include <RF24Ethernet.h>//https://github.com/nRF24/RF24Ethernet
#include <PubSubClient.h>
#define NODE_IP 20

IPAddress ip(10,10,3,NODE_IP);
IPAddress gateway(10,10,3,1);

RF24 radio(7,8);
RF24Network network(radio);
RF24Mesh mesh(radio,network);
RF24EthernetClass RF24Ethernet(radio,network,mesh);
EthernetClient ethClient;
PubSubClient clientMqtt(ethClient);
int count = 0;
uint32_t mesh_timer = 0;
uint32_t mqtt_timer = 0;
void reconnect() {
  // Loop until we're reconnected
  if (!clientMqtt.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (clientMqtt.connect(String(String("node")+String(ip[3])).c_str())) {
      Serial.println("connected");
    } else {
      count = 0;
      Serial.print("failed, rc=");
      Serial.print(clientMqtt.state());
      Serial.println(" try again in 1 seconds");
      // Wait 5 seconds before retrying
      delay(1000);
      mesh.renewAddress();
    }
  }
}

void setup(){
  Serial.begin(115200);
  clientMqtt.setServer(gateway, 1883);
  Ethernet.begin(ip);
  Ethernet.set_gateway(gateway);
```

```
  mesh.setNodeID(NODE_IP);
  mesh.setChild(true);
  mesh.begin();

 }

 void loop()
 {
  int sensorValue = analogRead(A0); //read the A0 pin value
  float voltage = sensorValue * (5 / 1023.00) * 2;
  mesh.update();
  if (!clientMqtt.connected()) {
   reconnect();
  }else{
   if(millis()-mqtt_timer > 5000){ //Every 5 seconds send heartbeat via MQTT
    mqtt  timer = millis();

clientMqtt.publish("heartbeat",String("{\"name\":\""+String(String("node")+String(ip[3]))+"\",\"heartbeat\":"+String(cou
nt++)+",\"parent_address\":"+String(network.parent())+",\"mesh_address\":"+String(mesh.mesh_address)+",\"voltage\":"
+String(voltage)+"}").c  str());
   }
  }

  if(millis()-mesh_timer > 30000){ //Every 30 seconds, test mesh connectivity
   mesh_timer = millis();
   if( ! mesh.checkConnection() ){
     mesh.renewAddress();
   }
  }
  clientMqtt.loop();
 }
```

Esta página foi intencionalmente deixada em branco