PM

Sistema de Transmissão de Áudio Digital

PROJETO DE MESTRADO

David Miguel Abegão Inácio

MESTRADO EM ENGENHARIA ELETROTÉCNICA - TELECOMUNICAÇÕES

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

fevereiro | 2020

Sistema de Transmissão de Áudio Digital

PROJETO DE MESTRADO

David Miguel Abegão Inácio

MESTRADO EM ENGENHARIA ELETROTÉCNICA - TELECOMUNICAÇÕES

ORIENTAÇÃO

Joaquim Amândio Rodrigues Azevedo

Por amor à ciência…

“A ciência é, portanto, uma perversão de si mesma, a menos que tenha como fim último, melhorar a humanidade.”

Nikola Tesla, cientista-inventor. (1856-1943)

Resumo

O presente trabalho pretende abordar a transmissão sem fios de áudio digital, assentando em dois objetivos primordiais: o apoio a pessoas em locais sem cobertura de redes GSM (*Global System for Mobile*) e a monitorização remota de fenómenos acústicos ambientais.

Desenvolveram-se protótipos para comunicação bidirecional, difusão de mensagens gravadas e transmissão de mensagens em diferido para uso em locais remotos sem acesso à rede GSM ou com acesso vedado ou condicionado por condições extremas (nomeadamente catástrofes naturais, incêndios e acidentes com elevado grau de destruição).

Um outro aspeto desenvolvido foi a gravação de áudio em alta qualidade 12 bits e 44,1 kHz (com vista a recolher informação acústica em locais remotos), recorrendo a protótipos com cartão de memória e ao posterior envio para um recetor.

Foram testados vários cenários de propagação em ambiente real, para os diferentes tipos de protótipos em causa. Para os sistemas de qualidade voz, em condições normais, obteve-se uma ligação bem-sucedida a cerca de 2 km de distância.

Para os protótipos de gravação e transmissão de alta qualidade, os resultados foram igualmente satisfatórios, com uma baixa taxa de erro e áudio sem alterações.

Inicialmente foi estudado o componente rádio nRF24L01+, como módulo integrado com Arduíno. Em seguida, foi equacionado o XBee, como potencial rádio alternativo. Foi escolhido o primeiro pelo facto de este componente permitir um débito máximo de transmissão de 2 Mbps, acesso ao meio com latência reduzida e o seu alegado baixo consumo de energia.

Palavras-chave:

Sistema de transmissão, Auxílio, Monitorização, Transmissão digital, nRF24L01+, Redes multimédia.

Abstract

The present work aims to address the wireless transmission of digital audio, based on two main objectives: the emergency support to people in places without coverage of GSM networks (Global *System for Mobile)*) and the remote monitoring of environmental acoustic phenomena.

Prototypes have been developed for two-way communication, broadcasting of recorded messages and transmission of deferred messages for use in remote locations without access to the GSM network or with access sealed or conditioned by extreme conditions (namely natural disasters, fires and accidents with a high degree of destruction).

Another aspect developed was the recording of audio in high quality 12 bits and 44.1 kHz (with the goal to collecting acoustic information in remote locations), using prototypes with memory card and subsequent sending to a receiver.

Several propagation scenarios were tested in real environment for the different types of prototypes concerned. For voice quality systems, under normal conditions, a successful connection was obtained about 2 km away.

For high-quality recording and transmission prototypes, the results were equally satisfactory, with a low error rate and unchanged audio.

Initially, the radio component nRF24L01++was studied as an integrated module with Arduino. XBee was then considered as an alternative radio potential. The first was chosen because this component allows a maximum transmission throughput of 2 Mbps, access to the medium with reduced latency and its alleged low power consumption.

Keywords:

Transmission system, Assistance, Monitoring, Digital transmission, nRF24L01+, Multimedia networks.

Agradecimentos

Seguem-se os meus verdadeiros e sinceros agradecimentos.

Ao meu orientador, professor Joaquim Azevedo, pela oportunidade que me proporcionou em desenvolver este projeto, pela disponibilidade, compreensão, dedicação e interesse durante todo o desenrolar do mesmo.

Aos docentes da Faculdade das Ciências Exatas e da Engenharia, pelo conhecimento transmitido, bem como pelo seu interesse e empenho em proporcionar a oportunidade de conhecer as mais vastas e diversas metodologias de trabalho.

Ao Dr. Vítor Osório, pelo apoio e motivação no início desta aventura que teve início em 2009 e só terminou agora.

Ao Eng.º Valter Monteiro, à Kosan Crisplant, SA e ao Eng.º Paulo Santos da CLCM pela total cooperação e apoio disponibilizado durante a realização do trabalho necessário para este projeto.

Ao Eng.º Eládio e ao Eng.º Drumond, bem como todos os outros colegas de trabalho, pelo esforço suplementar que por vezes tiveram de fazer.

À minha família (em especial à minha mãe e minha filha), que foi mais sacrificada em todo o meu percurso académico, pelo esforço e pelo apoio manifestado. À minha incansável e fantástica companheira, Maria Olim, pela paciência e pelo apoio demonstrado.

Aos meus colegas e amigos que tive oportunidade de conhecer, trabalhar e conviver ao longo de todos os meus anos nesta instituição, em especial ao Nuno Carreira, Miguel Quintal, Wilson Azevedo, João Castro, Vítor Aguiar, João Nuno, Miguel Ornelas, Iúri Viveiros, Sérgio Manuel, Rui Martins, Pedro Nunes, Dino e Dinarte Vasconcelos, Rodrigo Vasco, Duarte Alves, Humberto Gonçalves, Simão Pedro e Rafael Velosa, cujo apoio foi essencial projeto se concretizasse.

Finalmente ao meu pai, por ter me ter dado este espírito incansável e persistente. Apesar de já ter partido há muito tempo, foi a minha inspiração.

Índice

Lista de acrónimos

ACK	*Auto Acknowledge*
ADC	*Analogue to Digital Converter*
ADPCM	*Adaptative Differential Pulse Codes Modulation*
APCM	*Adaptive Pulse Code Modulation*
ANACOM	*Autoridade Nacional das Comunicações*
CODEC	*Compressor-Decompressor*
CRC	*Cyclic Redundancy Check*
DAC	*Digital to Analogue Converter*
DFT	*Discrete Fourier Transform*
ESB	*Enhanced Shock Burst*
FFT	*Fast Fourier Transform*
GSM	*Global System for Mobile*
I2C	*Inter-Integrated Circuit*
IDE	*Integrated Development Environment*
ISM	*Industrial Scientific and Medical*
JTAG	*Joint Test Action Group*
LED	*Light Emitter Diode*
LR-WPAN	*Low-Rate Wireless Personal Area Networks*
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
OTG	*On-The-Go*
PSU	*Power Source Unit*
PWM	*Pulse Width Modulation*
PTT	*Push to Talk*
RF	*Radio Frequency*
GFSK	*Gaussian frequency-shift keying*
RISC	*Reduced Instruction Set Computer*
RSSF	*Redes de Sensores Sem Fios*
RSSFA	*Rede de Sensores Sem Fios Áudio*
RSSI	*Received Signal Strength Indicator*
SCK	*Serial Clock*

SD	*Secure Digital*
SPI	*Serial Peripheral Interface*
SPI	*Serial Protocol Interface*
SS	*Slave Select*
TWI	*Two-Wire Interface*
UART	*Universal Asynchronous Receiver-Transmitter*
USB	*Universal Serial Bus*

Índice de figuras

Índice de tabelas

1. Introdução

Neste capítulo é apresentada a motivação para a realização deste projeto, descritos os objetivos a alcançar, assim como a estrutura definida para este relatório.

1.1. Motivação

Com o desenvolvimento tecnológico, as redes de sensores sem fios (RSSF) começaram a ter aplicação nas mais diversas áreas, havendo muitas vezes a necessidade de um débito de transmissão elevado recorrendo a *hardware* limitado. Por seu lado, quando se pretende um melhor desempenho, existe a necessidade de um consumo elevado e insustentável de energia. Desta forma, torna-se necessário recorrer a *hardware* mais eficiente, nomeadamente, melhores débitos de transmissão e com o menor consumo de energia possível.

Quando inseridas num contexto de transmissão de áudio, as redes de sensores implicam determinadas preocupações e requerem certos cuidados, tais como, uma maior garantia de fiabilidade, resistência a perturbações na rede, redundância, escalabilidade, baixo consumo energético e baixa latência, de modo a que o envio das mensagens seja o mais eficaz possível.

As redes de sensores sem fios apresentam diversas vantagens quando comparadas com as redes cabladas: custos de instalação e manutenção mais reduzidos, facilidade de instalação em locais remotos (serras, picos e vales sem acesso), podem ser instalados mais facilmente sistemas de redundância com recurso a mais do que um nó sensor, maior área de cobertura, entre outras.

Em relação à transmissão de áudio, existem locais onde não há cobertura de redes cabladas e o acesso da rede móvel é escasso ou mesmo inexistente. As redes de sensores sem fios para transmissão de áudio possibilitam a transmissão de vários tipos de mensagens, desde a recolha de informação sobre fenómenos da natureza, até à difusão de pedidos de socorro em áreas remotas, ou simplesmente como meio de comunicação através de mensagens de voz diferidas ou em direto, nas mais diversas e variadas circunstâncias. É necessário saber aproveitar e entender os recursos tecnológicos, com o intuito de desenvolver formas de maximizar o seu uso.

Um exemplo claro da necessidade de garantir redes alternativas de comunicação foi o colapso das infraestruturas técnicas do sistema de telecomunicações da Região Autónoma da Madeira, no 20 de fevereiro de 2010.

1.2. Objetivos

O principal objetivo estabelecido para o presente trabalho foi o estudo da transmissão de áudio digital em plataformas de baixo custo e o aproveitamento dessa potencialidade para um sistema de apoio ou ajuda a pessoas em locais onde não exista rede de GSM, ou em caso de calamidade, possa estabelecer uma rede independente para comunicações de emergência (em direto ou em diferido). Assim, os objetivos definidos a desenvolver neste projeto foram:

- Análise de viabilidade da transmissão de áudio (8 bits mono a 8 kHz) ponto-a-ponto com Arduíno RISC (*Reduced Instruction Set Computer*) da família AVR a 8 bits.

- Envio de mensagens de áudio previamente gravadas a partir de um cartão SD com Arduíno da mesma família.

- Estudo do desempenho na transmissão de áudio de alta qualidade com uma melhor resolução (12 bits a 44,1 kHz) para monitorização de fenómenos acústicos ambientais.

- Gravação e transmissão de áudio em diferido recorrendo a um Arduíno da família ARM Cortex-M3 a 32 bits, onde a mensagem é gravada para um cartão SD, no instante que precede a transmissão.

A frequência usada para o trabalho foi a banda livre dos 2,4 GHz, por forma a cumprir com as normas legais em vigor em Portugal para a transmissão de áudio digital.

1.3. Organização da tese

A tese está organizada nos seguintes capítulos: Introdução; Revisão do estado de arte; Implementação; Resultados experimentais; Conclusões e trabalhos futuros.

No primeiro capítulo, Introdução, apresenta-se o tema desenvolvido na tese, mencionando a motivação, os principais objetivos e a organização da mesma.

No segundo capítulo é apresentado o trabalho relacionado com o estado da arte, resultado da pesquisa em artigos científicos, trabalhos académicos relacionados com a área e módulos comerciais existentes.

No terceiro capítulo é apresentado o desenvolvimento do projeto, dividido em quatro partes (transmissão de áudio em tempo real, difusão de mensagens gravadas, transmissão de áudio em tempo real com alta qualidade, envio de ficheiros de voz entre dois pontos), correspondendo à implementação dos protótipos utilizados no presente estudo.

O quarto capítulo diz respeito aos resultados obtidos nas várias vertentes do projeto, bem como a análise dos resultados obtidos.

O último capítulo apresenta as conclusões extraídas dos resultados obtidos no trabalho e indica alguns trabalhos futuros a serem desenvolvidos.

2. Estado da Arte

Por forma a melhor entender o estado do desenvolvimento das aplicações digitais de transmissão de áudio, fez-se o levantamento de alguns sistemas existentes com estes objetivos. Alguns dos sistemas inserem-se dentro das redes de sensores sem fios multimédia, outros são simplesmente sistemas de transmissão ponto a ponto.

2.1. Monitorização Acústica com RSSF

Na Universidade de Victoria, em 2010 [1], foi desenvolvido um sistema de monitorização acústica remota, por forma a permitir a obtenção de dados tão diversos como monitorização de parâmetros ambientais (poluição sonora), de deteção sísmica, entre outros. Este sistema baseia-se numa rede de sensores sem fios, que utiliza rádios do tipo Mica2 de 900 MHz (Figura 2.1). Foi um dos primeiros sistemas de redes de sensores a ser utilizado na transmissão de áudio digital, ainda que apresentasse algumas limitações na largura de banda (38,4 kbps) e na capacidade de processamento (processador Atmel 128 de 8 MHz). Além disso, a própria memória do dispositivo era limitada a 4 kB de espaço, o que não permitia, de forma realista, a aquisição com uma boa taxa de amostragem. Neste caso, foi alcançada uma frequência de amostragem de 4 kHz, correspondendo a uma máxima largura de banda do sinal de 2 kHz, o que permitia alguma qualidade de transmissão de voz.

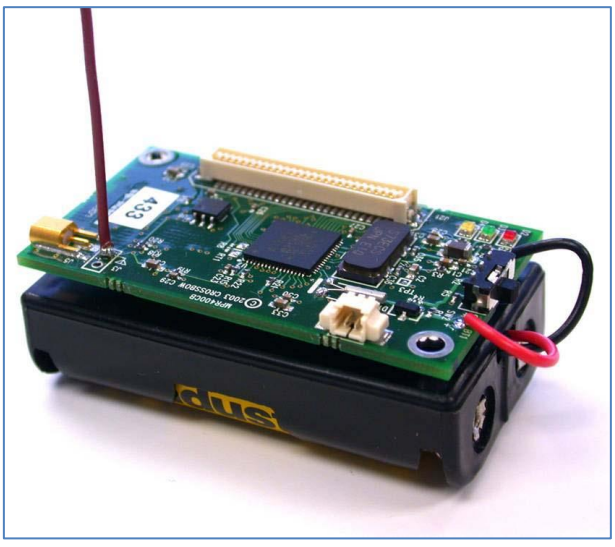

Figura 2.1 - MICA2 com dispositivo de aquisição de áudio [2].

Não foram disponibilizados os circuitos de áudio, o tipo de antena utilizada ou o respetivo alcance. As técnicas da poupança de energia foram baseadas em algoritmos de compressão e processamento, para minimizar os tempos de comunicação, uma vez que é nesse instante que o consumo é maior. Foram desenvolvidos 12 nós alimentados por bateria para um *gateway* de recolha de dados. Devido às limitações do sistema descritas, o circuito fazia o armazenamento na sua memória (muito limitada) e posterior envio.

Várias técnicas foram implementadas para otimizar o funcionamento deste dispositivo, desde algoritmos estatísticos à FFT (*Fast Fourier Transform*), nomeadamente a DFT (*Discrete Fourier Transform*), por forma poder tratar os sinais recorrendo a amostras que os caracterizam. Este protótipo possibilitava o envio de pequenas amostras de som para posterior análise num sistema recetor.

Em suma, foi um dispositivo que com as suas limitações permitia fazer com uma qualidade suficiente para a perceção humana, mas com durações muito curtas, pela falta de capacidade do *hardware* [1].

2.2. Intercomunicador digital de áudio com BIM2-433-160

Na Universidade de Aveiro foi realizado um trabalho, em contexto académico, que consistia na utilização do rádio BIM2-433 para transmissão de áudio e vídeo de baixa resolução na banda livre dos 433 MHz (Figura 2.2) [3].

Figura 2.2 - Radio BIM2-433-160 [4].

O trabalho realizado consistiu na utilização deste rádio e no reaproveitamento de uma consola de jogos de uma marca japonesa para a montagem do protótipo. Foi utilizado, como processador, um PIC18F458 da *Microchip*, implementada uma solução de baixo consumo (20 mA a 5 V). Não foi

abordada uma estratégia de poupança de energia, uma vez que o autor usou sempre alimentação da rede para os testes. Os esquemas do emissor e do recetor podem ser observados na Figura 2.3. Os principais componentes do sistema são o sistema de aquisição de áudio, composto por um microfone e circuito amplificador, os rádios [4], os amplificadores de saída e o altifalante do sistema.

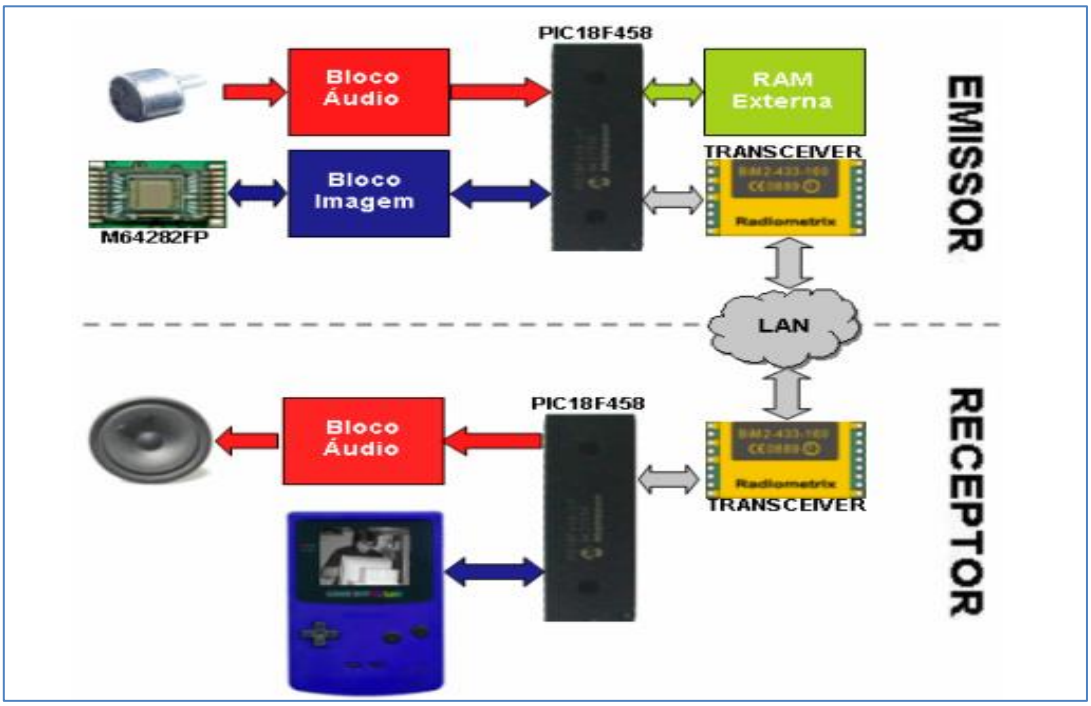

Figura 2.3 - Diagrama de blocos do sistema [3].

O sistema apresentava uma qualidade de áudio semelhante à do Mica2, com 8 kHz de frequência de amostragem. O seu funcionamento era em *half duplex* e a uma taxa de transmissão de 160 kbps, com alcances na ordem dos 200 metros em linha de vista. Como se tratava apenas de um protótipo de estudo académico, foi possível transmitir áudio na banda de frequências dos 433 MHz. No entanto, por imposição do regulador Autoridade Nacional das Comunicações (ANACOM), esta frequência é disponibilizada para aplicações de áudio apenas com uma janela de 10% de *dutty cycle*, ou seja, apenas 6 minutos por hora, o que inviabiliza o seu uso intensivo [5].

2.3. *Audio-on-Demand* em redes de Sensores sem fio

Em 2012 [6], na *University of Southern Denmark*, com o objetivo de transmitir áudio em diferido foi realizado um estudo académico para o efeito.

Este trabalho teve como principal estratégia a exploração das vantagens da transmissão em diferido, nomeadamente a necessidade de baixos débitos de transmissão e pouca necessidade de processamento. O rádio usado nesse trabalho foi o MicaZ, que opera na banda Iivre dos 2,4 GHz [7]. Foi utilizado o protocolo ZigBee, baseado na norma 802.15.4, com uma taxa nominal de 250 kbps [8]. Recorrendo a sistemas de armazenamento auxiliar, o áudio amostrado a uma frequência de 16 kHz foi recolhido, armazenado e enviado posteriormente usando a largura de banda disponível. Na Figura 2.4 pode-se observar a implementação de um modelo com vários nós a comunicar entre si (modelo de pequena escala) [6].

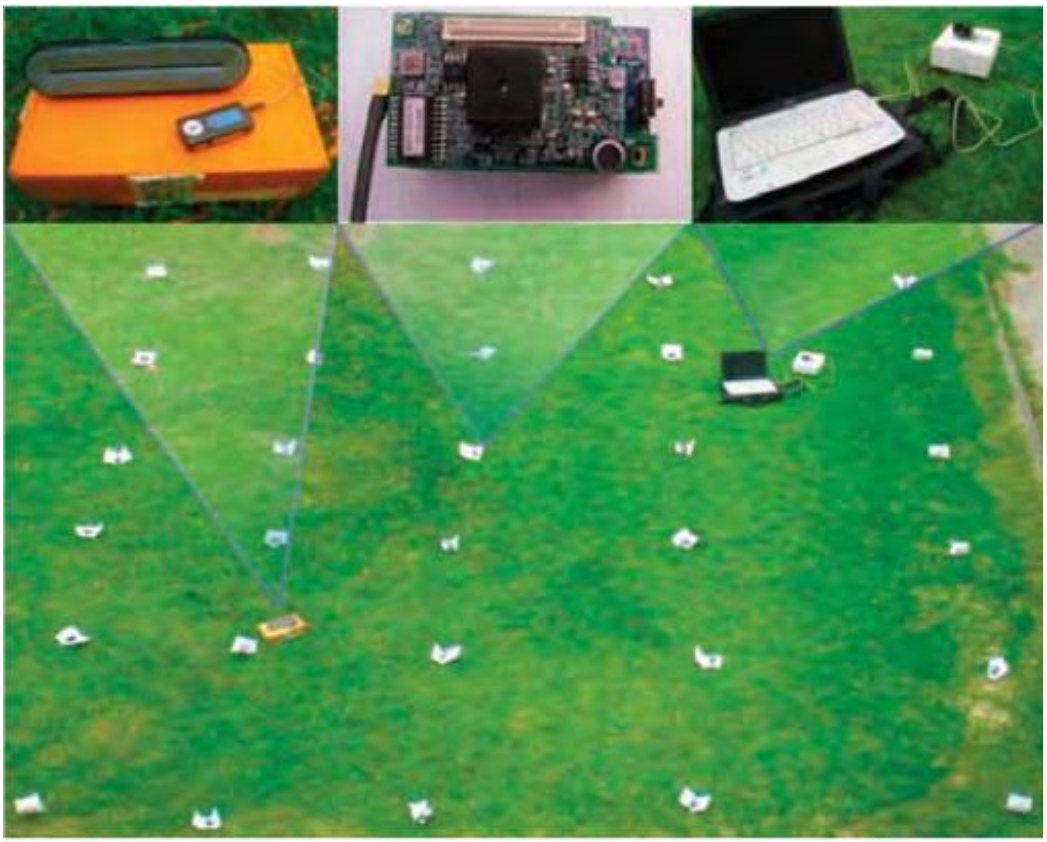

Figura 2.4 - Implementação do modelo e protótipo [6].

As vantagens deste sistema descritas pelos autores foram a possibilidade da obtenção, remota e a pedido, de áudio para os fins considerados e a possibilidade de adormecimento da rede, para um consumo muito reduzido. Como desvantagem apresentam a baixa capacidade de transmissão e o consumo relativamente elevado em transmissão (200 mA a 0 dBm) [8].

2.4. Rede de Sensores sem fios para diversas aplicações

Em 2016, Reinoso Carvalho e outros [9] abordaram o problema das redes de sensores sem fios para *design* sonoro, que consistia em reconstituir ou simular uma "paisagem sonora" a partir de sons obtidos numa rede de sensores de fios de áudio, no domínio da sociedade civil, nos cuidados de saúde e em biologia.

Ainda relativamente ao mesmo tema, os autores relataram que o uso desta técnica de transmissão de áudio pode ajudar e influenciar a sociedade civil de várias formas. Uma delas é a possibilidade de se recorrer a um esquema de transmissão independente, sem infraestrutura, com o propósito de proporcionar uma rede de emergência civil para comunicação em caso de calamidade. Outro exemplo referido foi no caso de manifestações populares, onde com recurso a esquemas adequados e concertados, se pode fazer o envio de multimédia para a *internet* [9].

No domínio dos cuidados de saúde, uma das formas de terapia relatada consistia na exposição do paciente a "paisagens sonoras relaxantes", com vista à sua reabilitação mental. A forma de fazê-lo em tempo real poderia ser com recurso a redes de sensores sem fios, onde se fazia a recolha remota de áudio para uma biblioteca de uso posterior.

Relativamente à biologia referem que os cientistas são consumidores de larga escala das redes de sensores sem fio, uma vez que se debatem com a recolha de dados em locais remotos e inóspitos. Embora as suas aplicações sejam maioritariamente de monitorização de parâmetros ambientais, alguns estudos conduzidos recentemente concluem que há alguma influência do som e das vibrações do meio no desenvolvimento das plantas. A necessidade de conduzir esta monitorização prende-se precisamente com a necessidade de aferir essas conclusões [9].

Todas as propostas apresentadas foram meramente conceptuais, não tendo sido verificado experimentalmente nenhuma delas. O conceito genérico pode ser observado na Figura 2.5, onde se pode observar um exemplo de reconstituição de ambiente sonoro. Do lado esquerdo é efetuada a recolha das amostras de som a partir de vários tipos de som, neste caso, de um ambiente

industrial. Do lado direito pode-se observar a reconstituição do ambiente sonoro, a replicar os sons obtidos remotamente.

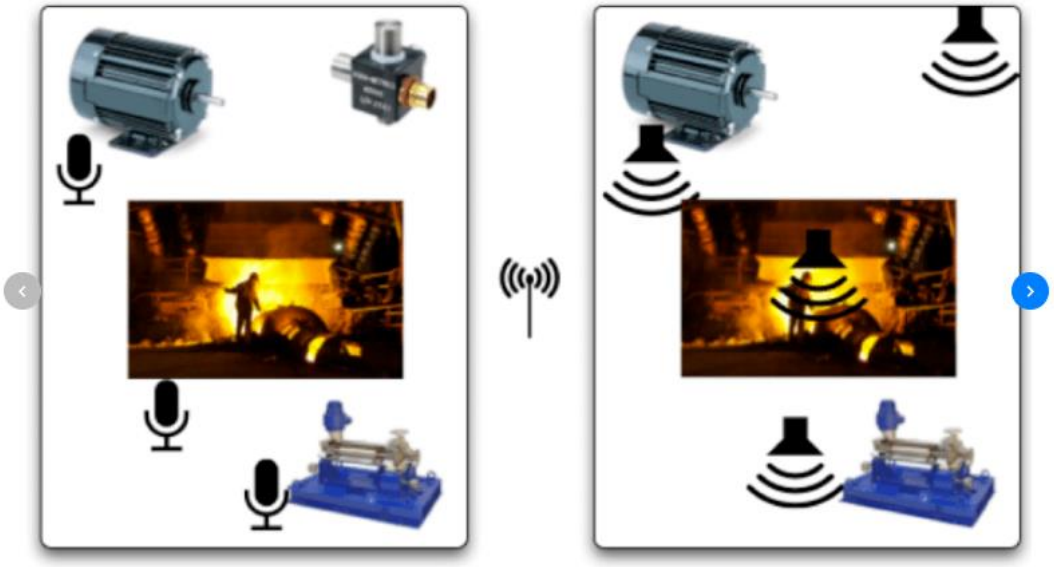

Figura 2.5 - Reconstituição de ambiente sonoro através de RSSF [9].

2.5. Transmissão de voz

Em 2014 foi publicado um trabalho [10] com o propósito de estudar a transmissão de áudio com "qualidade de voz", num contexto de ligações de baixa capacidade de transmissão e com muitas perdas de sinal. Os autores desse trabalho também abordaram a questão das redes de sensores sem fios para substituição das redes cabladas, em cenários de emergência.

A plataforma de comunicação escolhida foi o SenEar (Figura 2.6), baseado no ATMEL AT91SAM7256 e no rádio CC1100 que suporta até 500 kbps [11] de taxa de transmissão, com um consumo de 70 mA [10].

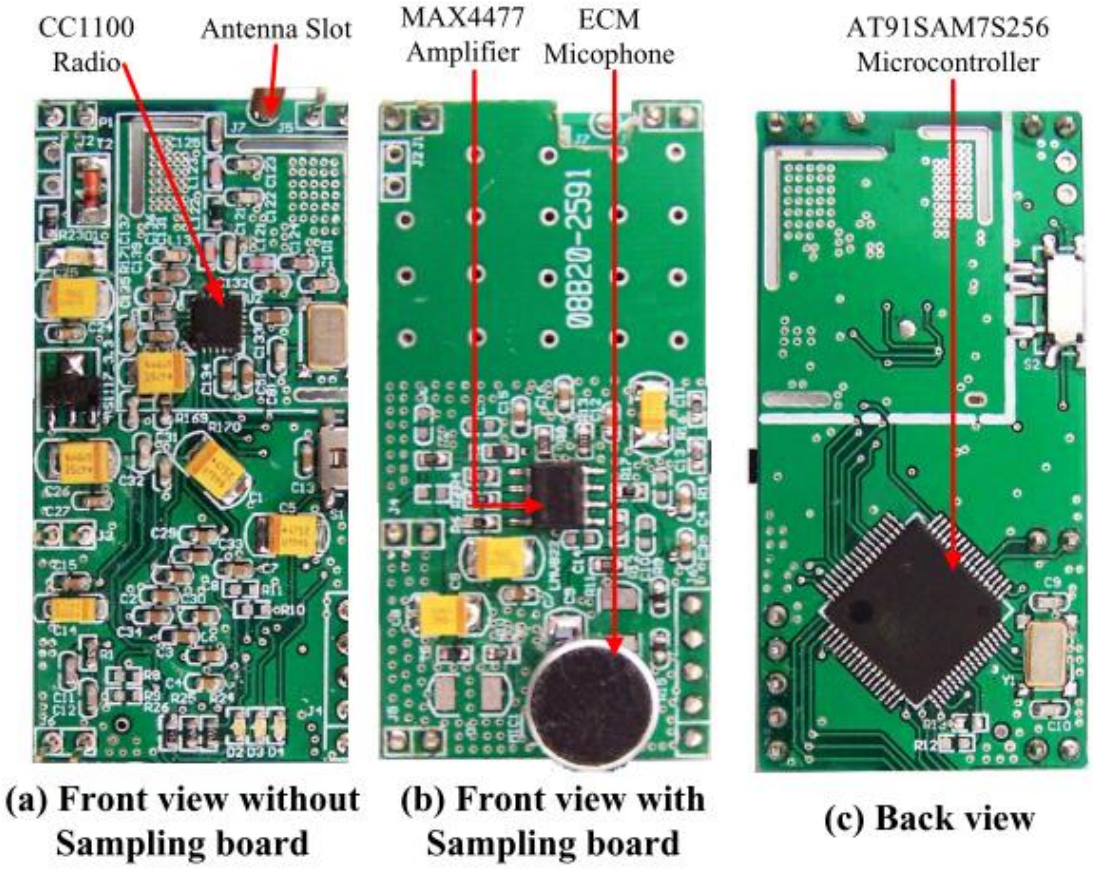

Figura 2.6 - Plataforma SenEar [10].

Com o objetivo de conseguir transmitir áudio nas condições mais desfavoráveis, foram usados algoritmos de APCM (*Adaptive Pulse Code Modulation)* para aperfeiçoar a transmissão e sistemas de controlo de admissão de novos emissores voz, para verificar se é possível acomodar mais intervenientes na rede.

Os resultados experimentais do trabalho revelaram que foi possível programar um sistema de transmissão de áudio com "qualidade voz". Por outro lado, recorrendo a algoritmos adaptativos de compressão dinâmica, foi possível mitigar os efeitos da variação da taxa de transmissão, em ligações com muitas perdas de sinal.

2.6. Comunicação sem fios de áudio digital

Em 2015, Anuj Gour e Preet Jain publicaram um artigo no qual propunham uma arquitetura de rede, para transmissão de áudio digital, recorrendo ao nRF24L01 [12]. Este rádio opera na banda ISM (*industrial, scientific and medical*)

de 2,4 GHz e a folha de características apresenta uma taxa de transmissão até 2 Mbps [13].

A proposta consistiu numa estrutura de nós sensores de recolha de voz e nós de encaminhamento.

A figura 2.7 apresenta o esquema de comunicação desde a recolha do áudio até ao seu destino.

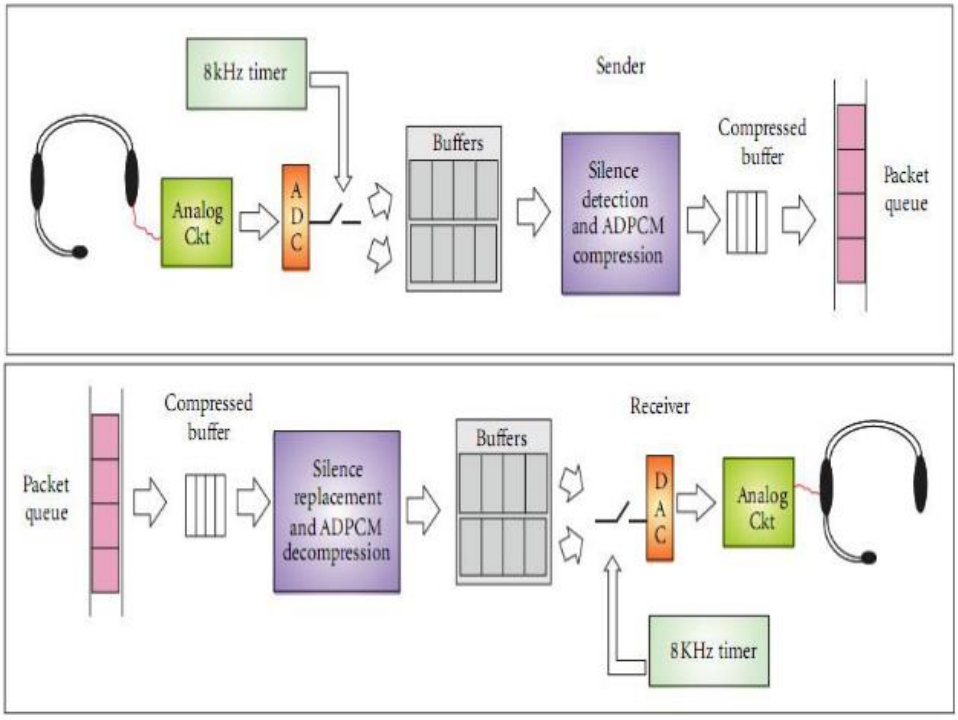

Figura 2.7 - Esquema de comunicações proposto [14].

A Figura 2.8 ilustra a arquitetura da RSSF utilizada para a comunicação áudio (recolha, encaminhamento e entrega de dados).

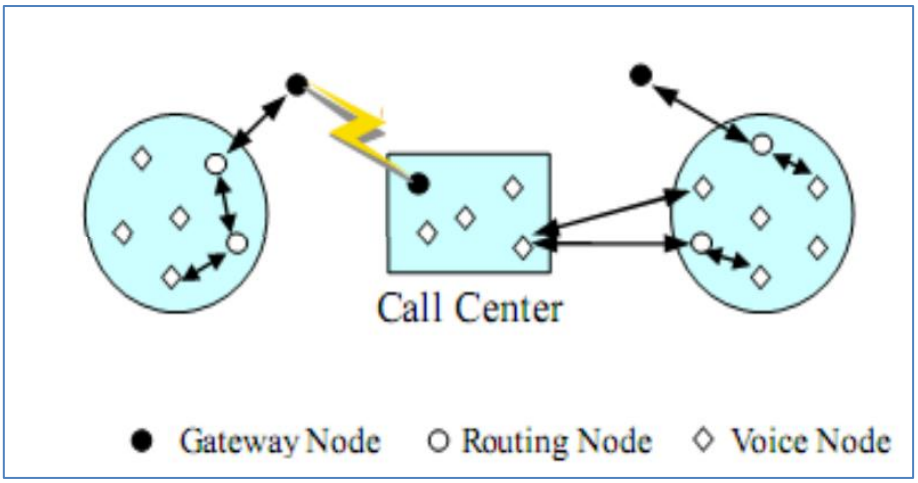

Figura 2.8 - Arquitetura da RSSF para voz [14].

A taxa de codificação de voz foi de 32 kbps, recorrendo ao ADPCM (*Adaptative Differential Pulse Codes Modulation*).

Os autores referem que testaram com sucesso a comunicação em *half duplex* entre dois nós, num alcance de poucos metros em linha de vista e sem obstáculos [14].

2.7. Dispositivo de transmissão de áudio XL-01M-T

O dispositivo XL-01-M é um produto comercial desenvolvido na China pela empresa *Lilly electronics* e que tem como objetivo o envio de áudio de alta qualidade entre dois pontos a curta distância, alegadamente 100 metros em linha de vista [15]. Trata-se um produto composto por dois módulos distintos, um emissor com um ADC (*Analogue to Digital Converter*) de 16 bits, para recolha e transmissão de áudio, e o recetor que, por sua vez, envia o sinal via DAC (*Digital to Analogue converter*) para um amplificador estéreo para alimentar os altifalantes respetivos. A Figura 2.9 apresenta os módulos do sistema.

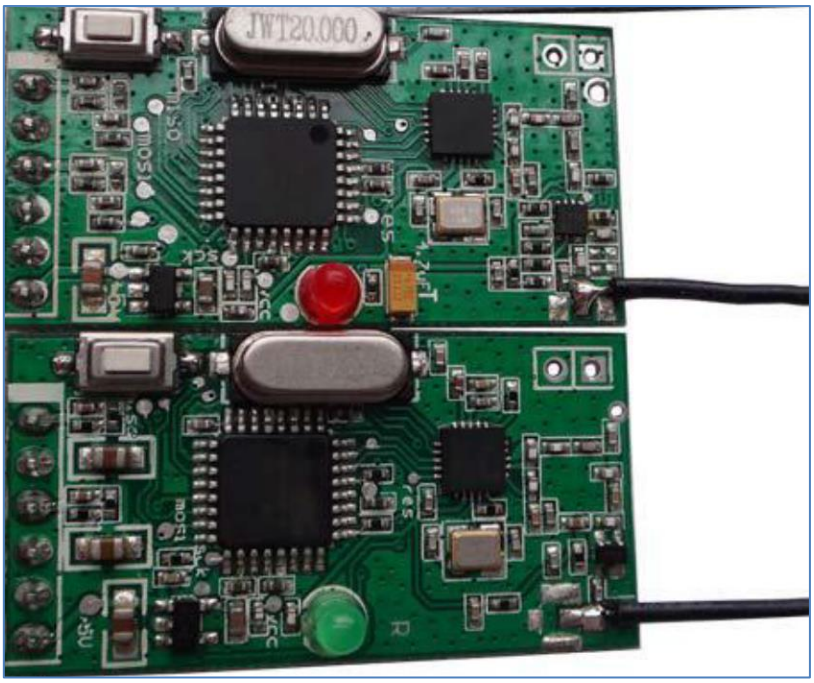

Figura 2.9 - Módulos de transmissão XL-01-M [15].

De acordo com a folha de características, o dispositivo funciona na frequência de 2,4 GHz, com 16 canais disponíveis e um consumo de 100 mA na emissão e 40 mA na receção. Permite, ainda, a transmissão em estéreo com separação do canal esquerdo e direito, respetivamente [15].

Um entusiasta português utilizou este sistema para uma ligação áudio entre uma guitarra e um amplificador, com elevada qualidade e baixa latência, apresentando um elevado consumo de energia entre 100 mA na emissão e 150 mA [16] na receção.

2.8. Análise da capacidade de transmissão de áudio - rede ZigBee

No contexto das redes LR-WPAN *(Low-rate wireless personal area networks*), em 2008, D.Brunelli e outros [17] publicaram um trabalho científico onde analisaram o desempenho das redes ZigBee para a transmissão de áudio.

Foram realizados vários testes recorrendo ao SoC (*System on Chip*) da Texas, que utiliza o rádio CC2430 (Figura 2.10).

Figura 2.10 - SoC ZigBee CC2430 da Texas Instruments [18]

Os autores afirmam ter conseguido realizar o envio de mensagens de áudio recorrendo a um sistema de CODEC (Compressor-Descompressor) por *hardware*, num contexto muito otimizado (dentro de uma habitação, sem obstáculos ou quaisquer outras condicionantes externas). Foram executados 20 testes de funcionamento de pacotes enviados para a rede em cada teste. Nas condições referidas, a taxa de transmissão foi de 68,9 kbps, sendo necessário, por isso, maximizar os algoritmos de transmissão, recorrendo ao preditivo ADPCM, ainda assim suficiente para um áudio de baixa qualidade.

Para os testes de comunicação foram montados protótipos, conforme o fluxograma da Figura 2.11.

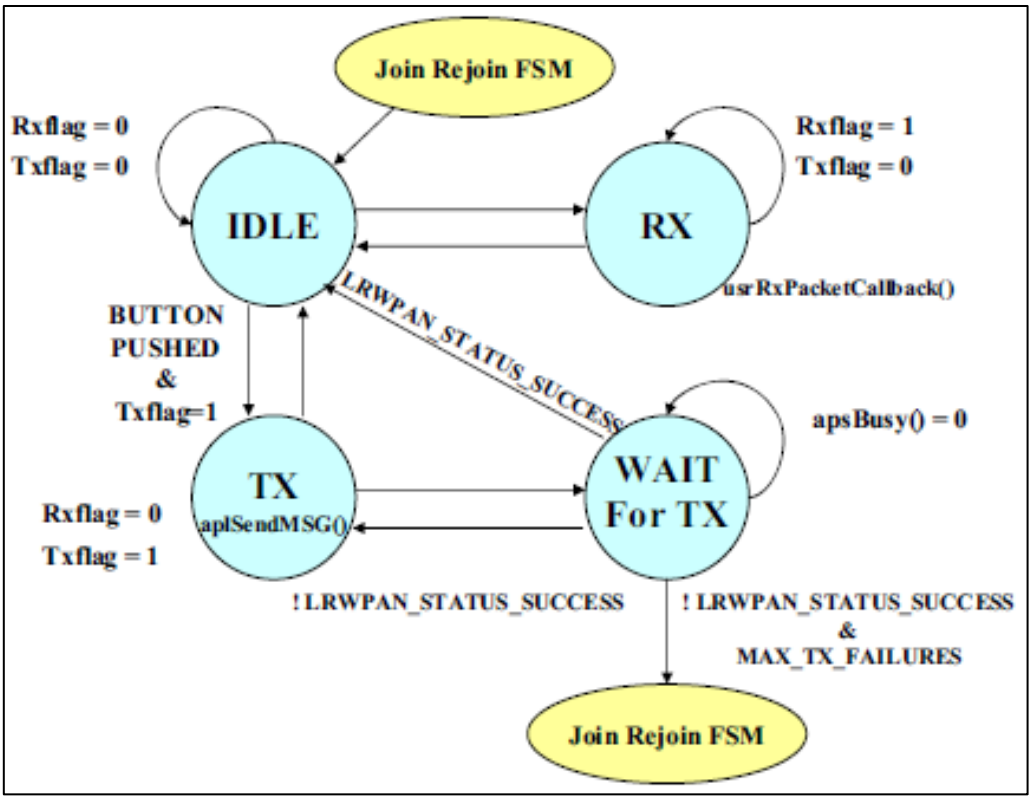

Figura 2.11 - Fluxograma da aplicação [17].

De referir que não há registo do consumo dos sistemas, nem quais os métodos de alimentação utilizados.

Os autores referem ainda que o desempenho real ronda os 30 kbps. Embora não seja o suficiente para áudio de alta qualidade, é suficiente para transmitir áudio minimamente percetível para as aplicações de envio de voz mais comuns. Os resultados não foram os melhores por apresentarem ainda algumas limitações devido às características de funcionamento do protocolo ZigBee, tempos de acesso ao meio, partilha de canal, entre outros.

Um outro trabalho realizado por Marcel Meli e outros [19], na *University of Applied Sciences*, refere igualmente um estudo efetuado que obteve resultados igualmente satisfatórios no domínio do áudio de baixa qualidade.

2.9. Transmissão de áudio com Raspberry Pi

Em 2015, Nalini Bagal publicou um trabalho [20] sobre a transmissão de áudio em tempo real, recorrendo ao Raspberry Pi em redes sem fio, neste caso recorrendo ao protocolo 802.11 (Wi-Fi).

Foi utilizado um Raspberry PI Modelo B+ (Figura 2.12) para a recolha do áudio e posterior envio via Wi-Fi para um computador PC.

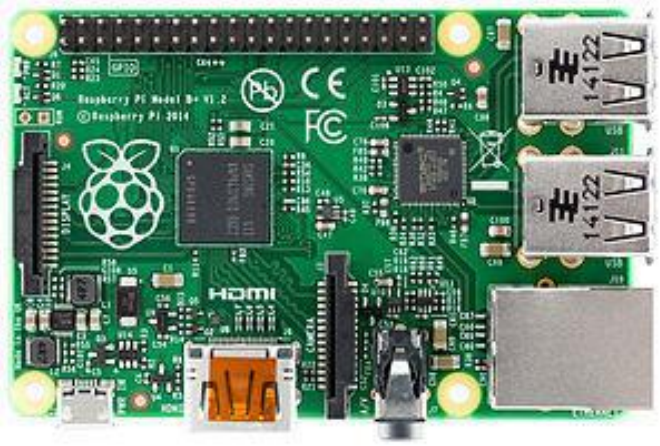

Figura 2.12 - Raspberry Pi [21].

A aquisição do áudio foi feita com recurso à entrada para microfone, cujo ajuste é efetuado com uma aplicação nativa do sistema operativo, ALSAmixer, ilustrada na Figura 2.13.

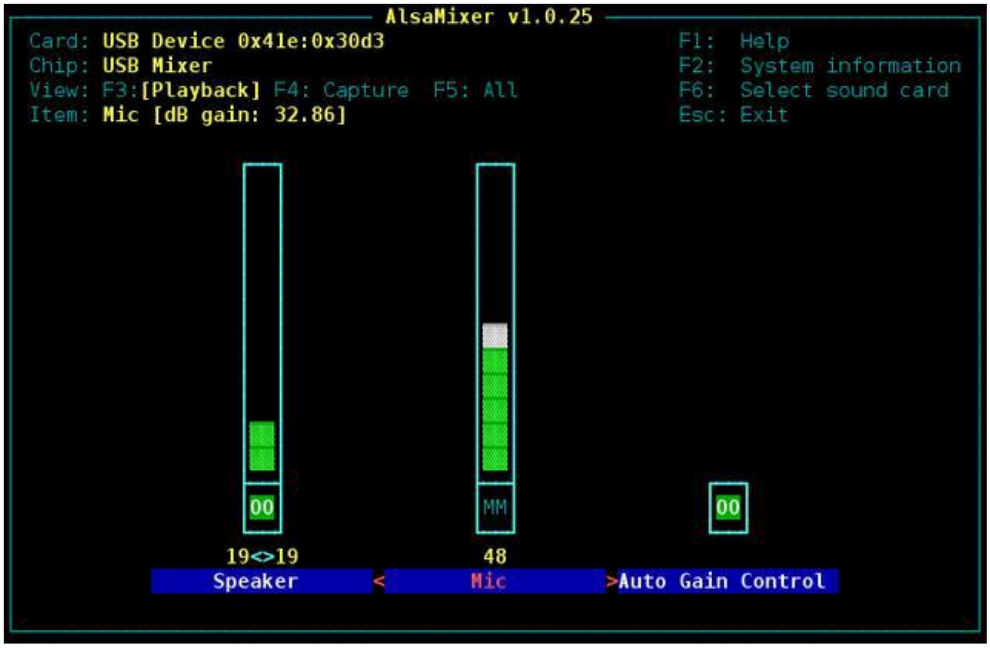

Figura 2.13 - Raspberry Pi ALSAmixer interface [21].

Para facilitar a gravação, armazenamento, reprodução e envio de voz foi concebida uma interface para a gestão do processo, recorrendo à linguagem de programação *Python*.

Recorrendo ao protocolo SSH, usando o cliente PuTTY, foi efetuada a ligação entre o Raspberry Pi e o computador PC, por Wi-Fi. Foi desta forma que o envio de *streaming* era feito em tempo quase real, com o atraso de 1 segundo

para processamento, gravação no Raspberry Pi e posterior envio para o computador. [21].

2.10. Teorema de Friis

A fórmula de Friis permite obter a potência recebida em espaço livre, sendo dada por

$$P_{rx}(dBm) = P_{tx} + G_{tx} + G_{rx} + 20log_{10}\left(\frac{\lambda}{4\pi D_r}\right) \quad (2.1)$$

em que P_{rx} é a potência recebida em dBm, P_{tx} é a potência transmitida também em dBm, G_{tx} e G_{rx} são os ganhos da antena de transmissão e de receção, respetivamente, λ é o comprimento de onda e D_r é a distância entre os dois sistemas.

O comprimento de onda λ é calculado por

$$\lambda = \frac{c}{f} \quad (2.2)$$

com c a velocidade da luz no vácuo (3x10^8 m/s) e f o valor da frequência. No âmbito deste trabalho, para a frequência de 2,4 GHz, tem-se 0,125 m.

2.11. Elipsoide de Fresnel

De modo a aferir as condições de espaço livre, foi utilizado o primeiro elipsoide de Fresnel, assinalado a verde na Figura 2.14, que deve estar livre de obstáculos.

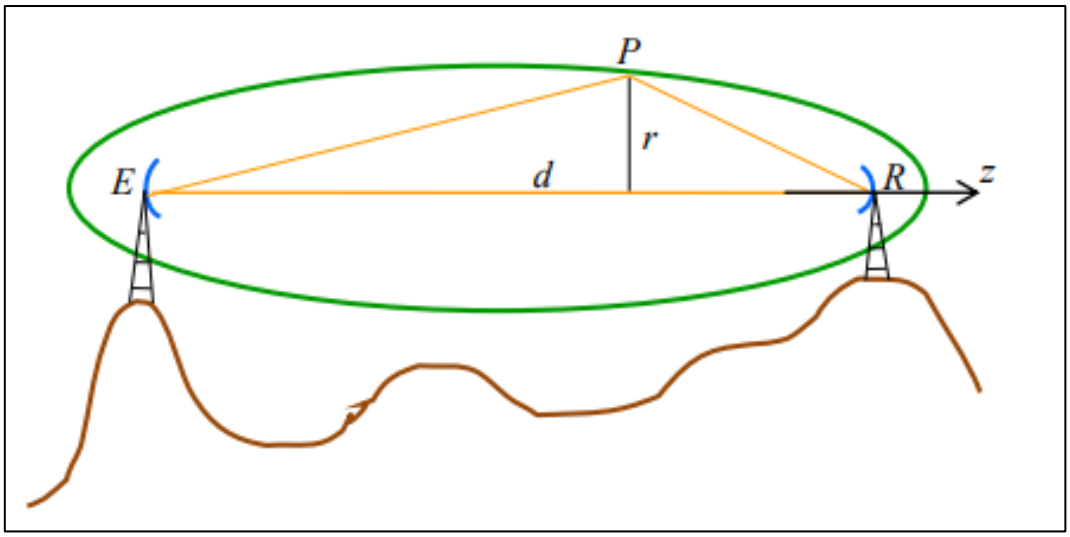

Figura 2.14 - Elipsóide de Fresnel [43]

Para garantir desimpedido o espaço referido, é necessário manter uma distância r ao eixo z. O elipsóide é obtido de modo que a diferença entre o percurso direto (distância ER) e o percurso formado pela distância do ponto E ao ponto P mais a distância do ponto P ao ponto R é igual a meio comprimento de onda. Por forma a calcular essa referida distância usa-se

$$\boldsymbol{r = \sqrt{\frac{d\lambda}{4}}} \tag{2.3}$$

em que d é a distância entre antenas e λ é o comprimento de onda.

2.12. Conclusões ao capítulo

Existem vários estudos e trabalhos efetuados na área da transmissão de áudio digital. Alguns não apresentam as principais características do sistema, tais como os tipos de antenas usadas, o máximo alcance obtido, o consumo de energia, as formas de alimentação usadas, os débitos máximos de transmissão, entre outros.

Já foram abordadas várias questões de interesse, nomeadamente a possibilidade de usar uma RSSFA (Rede de Sensores Sem Fios Áudio) em zonas onde, por exemplo, não exista acesso à rede móvel, ou esta tenha colapsado em virtude de catástrofe natural ou destruição intencional. Por outro lado, os sistemas existentes apresentam severas limitações, nomeadamente a necessidade de sofrerem alterações profundas para

corresponder à necessidade dos utilizadores no caso concreto da ajuda a pessoas. Por exemplo, seria muito interessante desenvolver um sistema que recolhesse ou gravasse uma mensagem de voz, num determinado local e enviasse para um destino predefinido. Seria então necessário proceder à criação de uma interface entre a RSSFA e a rede de emergência civil, por forma a receber as mensagens no local desejado.

3. Projeto de transmissão de áudio digital

Neste capítulo será apresentado o trabalho efetuado no domínio da transmissão de áudio digital, num contexto adaptável a vários cenários, como ajuda a pessoas ou monitorização remota de áudio. Começa-se por descrever os principais componentes utilizados no projeto e, em seguida, são apresentados os sistemas desenvolvidos para a transmissão de áudio.

3.1. Arquitetura do projeto

O projeto foi dividido em quatro partes, conforme os objetivos definidos para cada uma delas.

A primeira parte do trabalho foi definida como sendo o desenvolvimento do modelo baseado num Arduíno de 8 bits e no rádio nRF24L01+, onde se pretendeu fazer a transmissão de áudio digital entre dois ou mais pontos, em *half duplex*. A Figura 3.1 mostra a arquitetura deste sistema, onde dois ou mais utilizadores podem comunicar entre si.

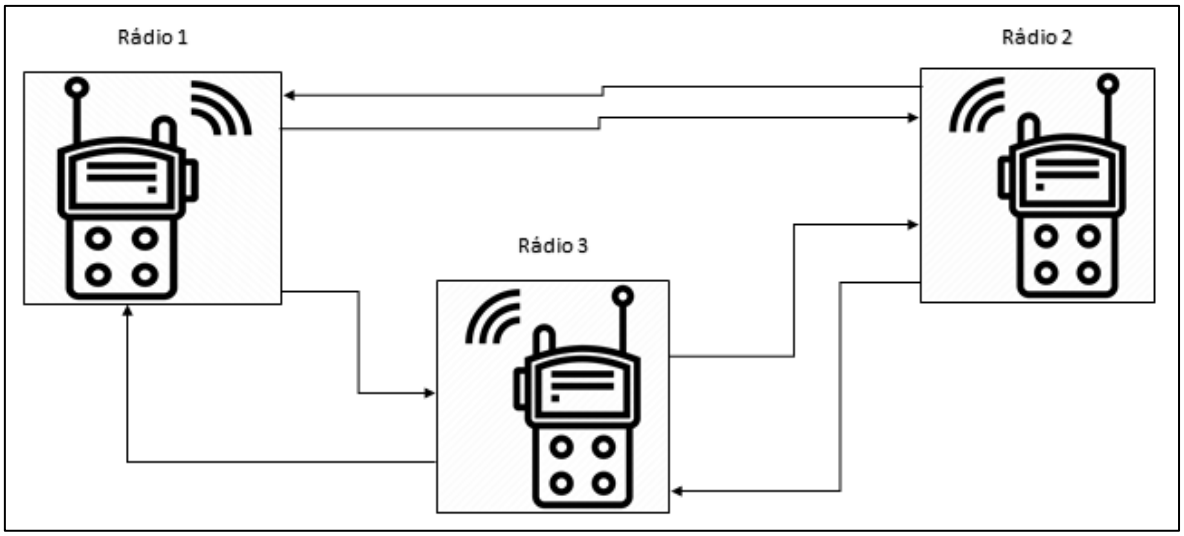

Figura 3.1- Esquema de comunicação a testar.

Posteriormente, mantendo o microcontrolador, adicionando um leitor-gravador de cartão SD, pretendeu-se fazer a mesma experiência, mas com o objetivo de enviar mensagens de áudio pré-definidas para serem apresentadas numa saída de áudio. O utilizador pode enviar uma das várias mensagens disponíveis, por exemplo, com a gravação de uma mensagem de pedido de ajuda em Português e através de um menu num ecrã com várias línguas o

utilizador seleciona mensagem a enviar, bastando para isso pressionar o botão correspondente à mesma. Este sistema é totalmente compatível com os rádios PTT (*Push To Talk*) usados na experiência anterior, desde que sejam respeitados os endereços indicativos na rede. A arquitetura simplificada do sistema pode ser observada na Figura 3.2.

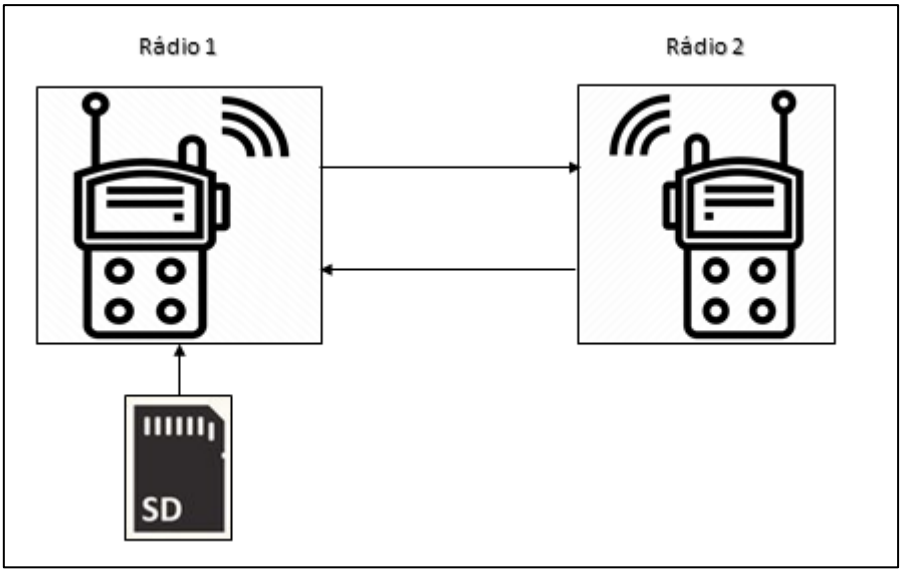

Figura 3.2 - Arquitetura AVR com envio de mensagens 8 bits mono 16 kHz a partir de cartão SD.

A terceira parte do trabalho consistiu na transmissão em direto, entre dois protótipos, com uma qualidade áudio de 12 bits e 44,1 kHz. O microcontrolador a utilizar é o SAM3X8E ARM Cortex-M3 de 32 bits (em Arduíno DUE), num enquadramento semelhante dos microcontroladores AVR, com a possibilidade de transmitir em *half-duplex* entre dois ou mais terminais. Por uma questão de simplicidade foi decidido utilizar apenas dois terminais, nesta fase do trabalho. A arquitetura básica do sistema é a que se pode observar na Figura 3.3.

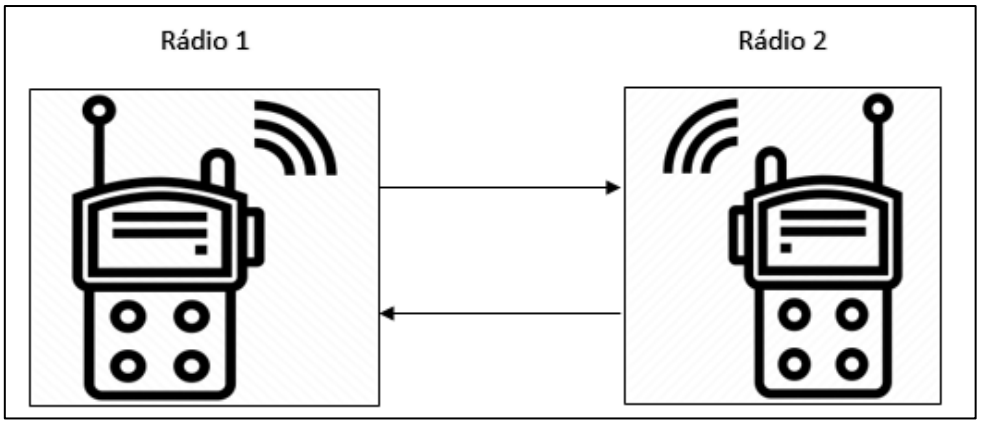

Figura 3.3 - Arquitectura de sistema de bidireccional com Arduíno DUE (12 bits a 44,1 kHz).

Por fim, o objetivo foi a gravação de mensagens de voz para envio diferido (momentos antes da transmissão e pelo utilizador que necessita de ajuda), onde para isso, se usou o Arduíno com cartão SD para armazenamento. O utilizador pressiona um botão para desencadear o processo de gravação (mediante a presença de um LED *(Light Emitter Diode)* de aviso). No fim da mensagem gravada, o envio é efetuado automaticamente, para o recetor, diretamente ou através de um repetidor. A representação deste conceito é ilustrada pela Figura 3.4.

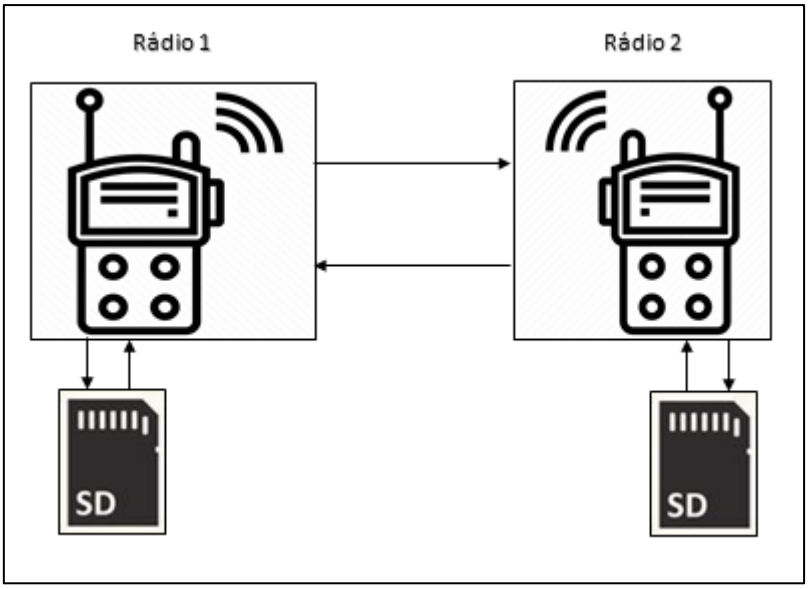

Figura 3.4 - Arquitetura com DUE com envio diferido.

3.2. Protocolos de comunicação

Os protocolos de comunicação a utilizar neste trabalho serão apresentados nesta secção, sendo neste caso concreto o SPI (*Serial Protocol Interface*) e o ESB (*Enhanced Shock Burst*).

3.2.1. Protocolo SPI

SPI é um protocolo de comunicação usado por microcontroladores para a transmissão rápida de informação a curta distância com dispositivos periféricos, incluindo outros microcontroladores [22].

Na comunicação realizada por este protocolo é utilizada uma relação mestre-escravo, onde o mestre é habitualmente o microcontrolador que controla o periférico, sendo este o escravo. Uma ligação deste tipo entre dois ou mais dispositivos efetua-se através de 4 ligações:

• MISO (*Master In Slave Out*) - O escravo envia informação para o mestre;

• MOSI (*Master Out Slave In*) - O mestre envia informação para o escravo;

• SCK (*Serial Clock*) - Sinal de relógio gerado pelo mestre para sincronização da transmissão;

• SS (*Slave Select*) - O mestre consegue ativar ou desativar a comunicação com certo dispositivo [22].

Enquanto os três primeiros pinos são partilhados por todos os dispositivos interligados, o pino de “seleção de escravo” difere para cada periférico. Este último é especialmente relevante quando existem vários periféricos simultaneamente conectados ao microcontrolador, pois permite uma multiplexagem das linhas MISO, MOSI e SCK.

3.2.2. *Enhanced Shock Burst*

O ESB, que surge como um aprimoramento do antigo protocolo *Shock Burst*, é um protocolo concebido e utilizado pela *Nordic Semiconductor*, pertencente à camada Ligação de Dados (Modelo OSI) que suporta comunicações baseadas em pacotes, unidirecionais ou bidirecionais. A utilização do ESB permite a implementação de comunicações de alto desempenho e consumo muito baixo, recorrendo a microcontroladores de baixo custo para desempenharem a função de *host* [12].

As principais características desde protocolo são: manipulação automática dos pacotes, incluindo mensagens de *acknowledgement* automáticas, retransmissões automáticas, dados úteis (*payload*) dinâmicos entre 0 e 32 bytes e a utilização de *pipes* para permitir topologias de rede em estrela (até 1:6) [12]. A versão *Enhanced* tem como principal diferença para a versão anterior, o campo de controlo de pacotes, onde se identificam os pacotes e se determina a validação de ACK (*Auto Acknowledge*) ou não.

Observe-se a Figura 3.5 que mostra o formato dos pacotes *Enhanced Shock Burst*, considerando à esquerda o bit mais significativo.

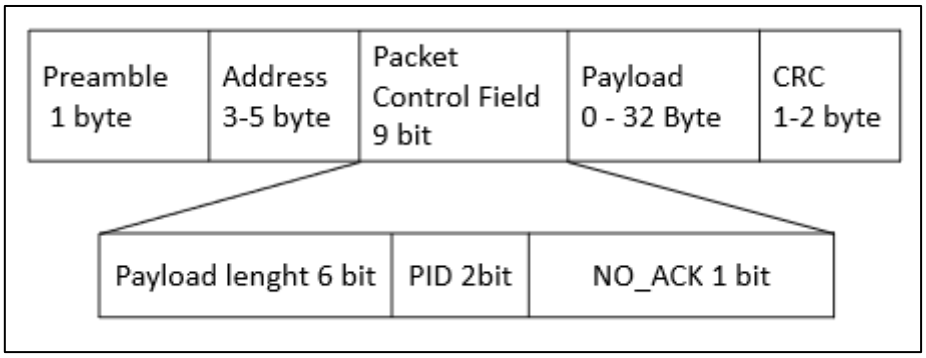

Figura 3.5 - Formato das tramas de *Enhanced Shock Burst* [12].

Cada um dos campos desempenha uma função específica, apresentadas de forma resumida, na Tabela 3.1.

Tabela 3.1 - Descrição dos campos que compõem a trama ESB.

Campo	Tamanho	Função
Preâmbulo	1 byte	Sequência de bits usada para sincronizar o desmodulador do recetor
Endereço	3-5 bytes	Endereço do recetor
Tamanho dos dados úteis	6 bits	Especifica o tamanho dos dados úteis em bytes
Identificação do pacote	2 bits	Usado para especificar se o pacote é novo ou retransmitido
Flag de *no acknowledgement*	1 bit	Avisa o recetor se deve ou não ser enviada uma mensagem *ACK*
Dados úteis	0-32 bytes	Conteúdo definido pelo utilizador
CRC (*cyclic redundancy check)*	1-2 bytes	Mecanismo de deteção de erros no pacote

3.3. Componentes

Nesta secção, são apresentados os componentes utilizados em todo o trabalho, desde microcontroladores, ao rádio, leitor-gravador de cartões SD e componentes para transmissão de áudio.

3.3.1. Microcontroladores

Nesta subsecção será efetuada a apresentação dos microcontroladores e placas de testes utilizados no desenvolvimento das várias etapas do trabalho.

3.3.1.1 Arduíno UNO

O Arduíno UNO, ilustrado na Figura 3.6, é uma placa com microcontrolador baseado na ATmega328P [23]. Tem 14 pinos de entrada/saída digitais, dos quais 6 podem ser usados como saídas PWM (*Pulse Width Modulation*), 6 entradas analógicas de 10 bits, um cristal de quartzo de 16 MHz, uma ligação USB (*Universal Serial Bus*), um conector de alimentação e uma UART (*Universal Asynchronous Receiver-Transmitter*). Em algumas configurações comerciais, o microcontrolador está colocado num encaixe, pelo que pode facilmente ser substituído caso se danifique.

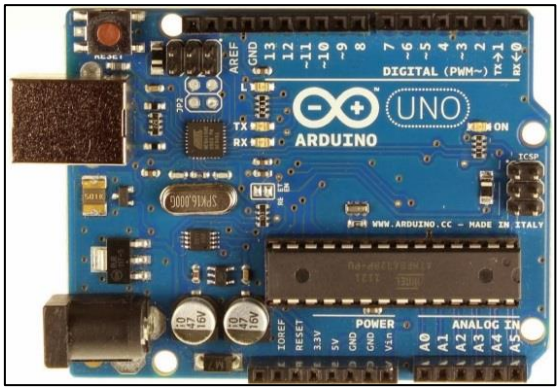

Figura 3.6 - Arduíno UNO [24]

3.3.1.2 Arduíno NANO

O Arduíno NANO é uma placa de experimentação e prototipagem baseada no ATmega328 (Arduíno NANO 3.x - Figura 3.7) [25]. Dispõe de 14 pinos digitais que podem ser usados como uma entrada ou saída, 8 entradas analógicas de 10 bits e uma UART. O componente utilizado opera a 5 V. Cada pino pode fornecer ou receber um máximo de 40 mA e tem uma resistência interna de *pull-up* (desligada por defeito) de 20-50 kΩ.

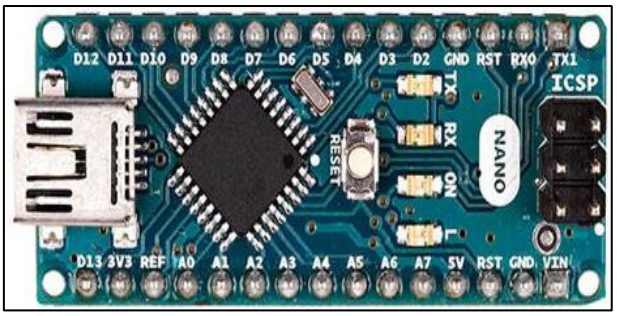

Figura 3.7 - Arduíno NANO [26].

3.3.1.3 Arduíno DUE

O Arduíno DUE, que se encontra na Figura 3.8, é um microcontrolador baseado no processador Atmel SAM3X8E ARM Cortex-M3 CPU [27] e é a primeira placa baseada num Arduíno com um processador de 32 bits ARM. Tem 54 entradas/saídas de pinos digitais, dos quais 12 podem ser usadas como saídas de PWM, 12 entradas analógicas, 4 UART, 84 MHz de velocidade de relógio, uma ligação USB (*Universal Serial Bus*) capaz de realizar ligações OTG (*On-The-Go*), 2 DAC, 2 TWI (*Two-Wire Interface*) também conhecidos por I2C (*Inter-Integrated Circuit*), um conector de alimentação, um conector SPI (*Serial Peripheral Interface*), um conector JTAG (J*oint Test Action Group*), um botão de *reset* e um botão para apagar a memória [28]. Ao contrário do Arduíno UNO e NANO, este componente opera a uma tensão de 3,3 V e com uma corrente 15 mA nos pinos digitais e ainda mais reduzida nos DAC (cerca de 3 mA). Em comparação com os anteriores, tem a vantagem de disponibilizar ADC com maior resolução (12 bits) e uma velocidade de relógio de bastante superior, de 84 MHz, o que se traduz numa maior capacidade de processamento.

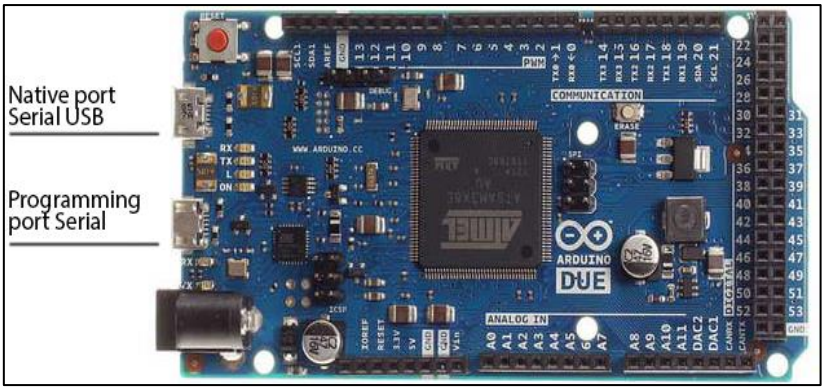

Figura 3.8 - Arduíno DUE [29].

3.3.2. Rádio nRF24L01+

O *nRF24L01+* é um rádio que emite na banda de frequências livres dos 2,4 GHz, concebido pela *Nordic Semiconductor* para aplicações sem fios de muito baixo consumo. Este integrado pode funcionar com um oscilador de cristal de 16 MHz e contém, ainda, um sintetizador de RF (*Radio Frequency*) e uma interface SPI de alta velocidade. O sistema de modulação é o GFSK (*Gaussian frequency-shift keying*) que sujeita os pulsos de dados a um filtro gaussiano para tornar as transições mais suaves. Este filtragem tem a vantagem de reduzir a potência da banda lateral, reduzindo a interferência nos canais vizinhos [12].

O rádio *nRF24L01+* tem uma gama de frequências entre os 2400 e os 2525 MHz, num total de 126 Canais, de 1 MHz cada. Em Portugal a Banda ISM é um pouco mais restritiva, impondo como limite máximo de frequência, os 2483,5 MHz (Canal 84 do dispositivo). Em todas as aplicações deste sistema será usado o canal 76 (2475 MHz) conforme assinalado na Figura 3.9. A escolha do mesmo prende-se com a recomendação dos programadores que desenvolveram as bibliotecas de base para o rádio [30] uma vez que está praticamente fora da banda do Wi-Fi (apesar de uma ligeira sobreposição com o canal 12). No entanto, se na área onde se desenvolve a aplicação esta frequência não estiver livre, ou estiver sujeita a ruído, pode ser escolhido outro canal recorrendo ao comando *radio.setChannel*() que pode receber como argumento o valor de uma variável ou de um comando de *Serial.read*().

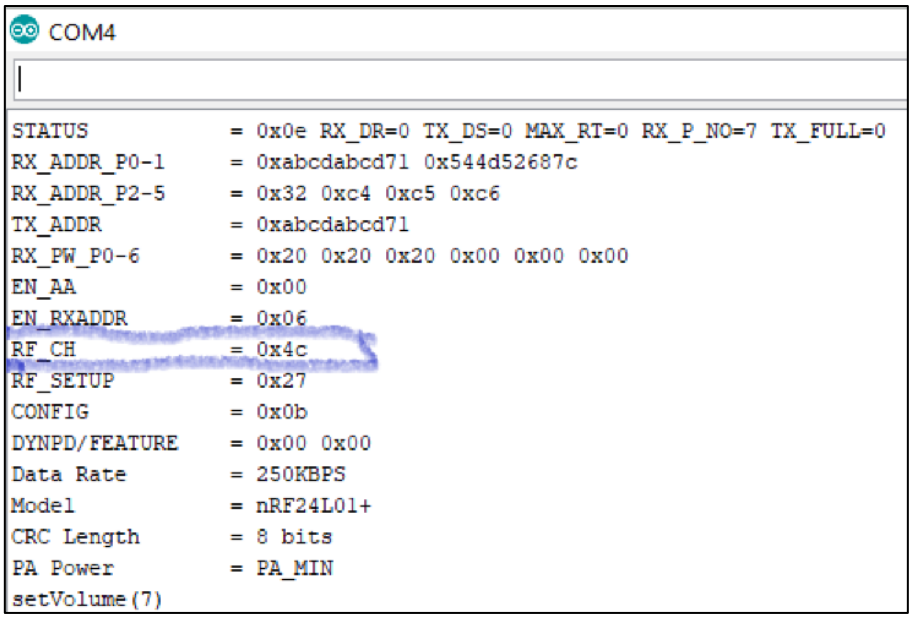

Figura 3.9 - Relatório de detalhes do Rádio nRF24L01+.

Este rádio foi escolhido por várias razões, entre elas o baixo consumo do mesmo, conforme descrito na Tabela 3.2, o elevado débito de 2 Mbps e a facilidade de o interligar com os microcontroladores disponíveis.

Tabela 3.2 - Tabela de consumo e potência do nRF24L01+ [31].

nRF24L01+ Emissor	
Níveis de Potência	0, -6, -12 e -18 dBm
Consumo a 0 dBm (Potência máxima)	11,3 mA
nRF24L01+ Recetor	
Níveis de Sensibilidade	-82 dBm @2 Mbps -85 dBm @1 Mbps -94 dBm @ 250 kbps
Consumo a -82 dBm (2 Mbps)	12,6 mA

O módulo integrado, que se pode observar na Figura 3.10, do lado esquerdo, é o módulo base que faz uso de uma antena impressa que permite reduzir o tamanho e o consumo do mesmo. Na mesma figura, no lado direito, observa-se o módulo com antena externa, que permite aumentar largamente o alcance devido ao seu amplificador de potência e ao seu pré-amplificador de receção de baixo ruído. Existem outros tipos de módulos que não foram utilizados neste projeto.

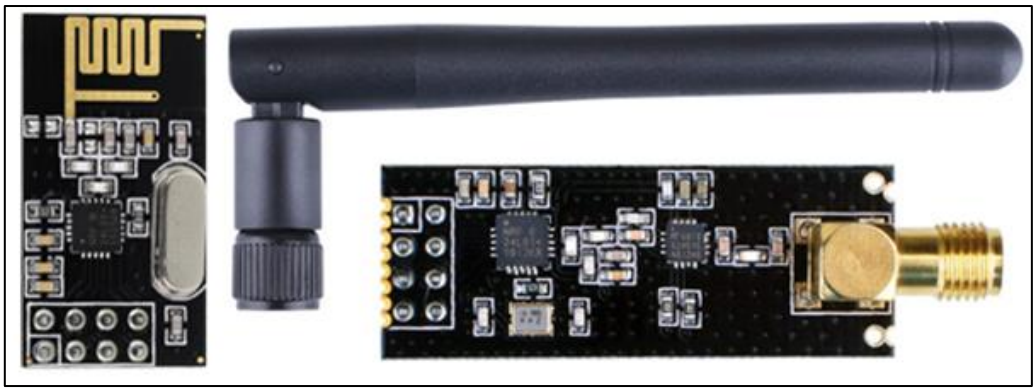

Figura 3.10 - Rádio nRF24L01+: à esquerda com antena patch; à direita com antena monopólo [32].

As descrições dos pinos podem ser observadas na Tabela 3.3.

Tabela 3.3 - Descrição dos pinos do nRF24L01+

Pino	**Nome**	**Abreviatura**	**Função**
1	Ground	Ground	Ligado à massa do sistema.
2	Vcc	Power	Alimentação do módulo a 3,3 V
3	CE	Chip Enable	Usado para habilitar a comunicação SPI
4	CSN	Chip Select Not	Este pino deve ser mantido em nível alto, senão desabilitará o SPI
5	SCK	Serial Clock	Fornece o pulso de *Clock* para a comunicação SPI
6	MOSI	Master Out Slave In	Conectado ao pino MOSI do MCU, para o módulo receber dados do Microcontrolador
7	MISO	Master In Slave Out	Conectado ao pino MISO do MCU, para o módulo enviar dados do Microcontrolador
8	IRQ	Interrupt	Ativa com nível lógico baixo e é usado somente se a interrupção for necessária

O *pinout* do rádio, a usar em todos os protótipos e montagem, pode também ser observado na Figura 3.11.

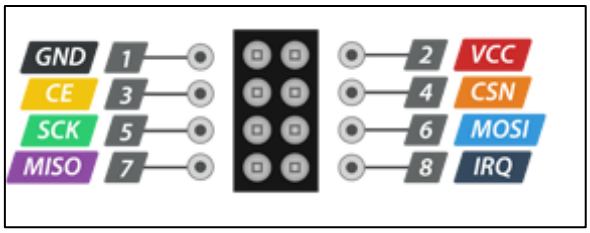

Figura 3.11 - Rádio nRF24L01+ Pinout [32].

3.3.3. Suporte adaptador para nRF24L01+

O adaptador com suporte para nRF24L01+ é um componente fundamental na ligação entre o microcontrolador e o rádio RF. Este pequeno módulo, para além de simplificar as ligações e garantir a estabilidade da alimentação do módulo nRF é de 3,3 V, adiciona uma filtragem na linha de alimentação para que seja retirado o máximo de partido do rádio.

A eletrónica implementada neste módulo é muito simples, sendo constituída por um regulador de tensão que permite uma entrada de até 7 V para uma saída de 3,3 V. Tem ainda na sua conceção quatro condensadores de *bypass* para filtrar ruído proveniente da fonte de alimentação ou do Conversor de *Step up*. A Figura 3.12 ilustra a versão comercial do modelo YL-105 adaptador para nRF24L01+.

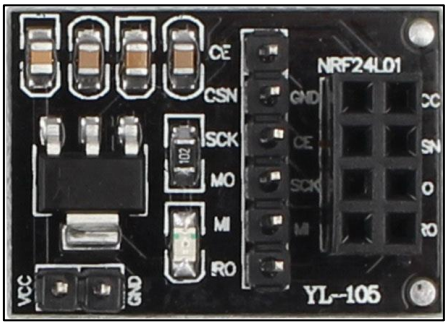

Figura 3.12 - Suporte regulador de tensão para nRF24L01+.

3.3.4. Conversor de tensão

Todos os protótipos foram alimentados pela linha de entrada de 5 V. Tipicamente as baterias disponíveis são de 3,7 V, pelo que foi necessário usar um circuito conversor de tensão. O circuito de alimentação comum a todos os protótipos foi baseado no elevador de tensão de 3,7 V para 5 V, recorrendo para esse fim ao circuito apresentado na Figura 3.13.

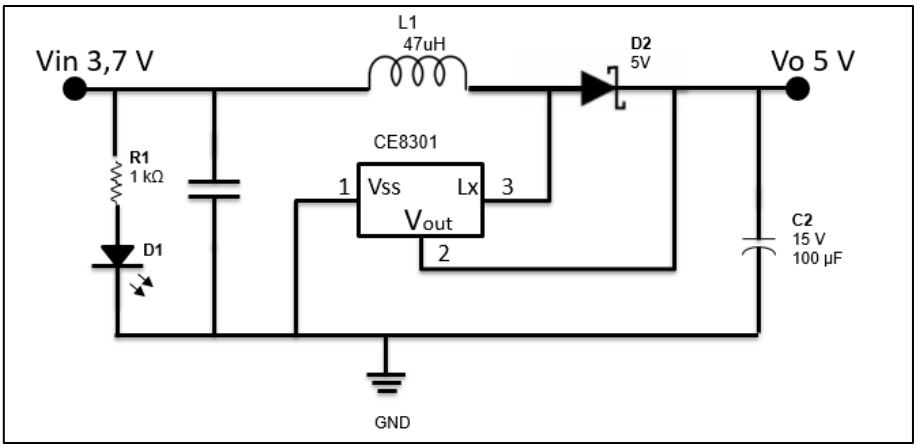

Figura 3.13 - Circuito de elevação de 3,7 V para 5 V.

3.3.5. Microfone

O elemento escolhido para captação de som foi um pequeno microfone de eletreto, apresentado na Figura 3.14, com uma resposta de frequência entre 20 Hz e 20 kHz. As suas pequenas dimensões e o seu baixo custo foram as principais razões para a sua escolha. [33].

Figura 3.14 - Microfone [34].

Como o microfone não é capaz de providenciar um sinal com amplitude suficiente para ser detetado, é necessário efetuar uma amplificação do mesmo antes da sua entrada no microcontrolador, a fim de este usar a totalidade da gama de tensão admitida pelo Arduíno DUE (0 a 3,3 V).

O circuito utilizado, apresentado na Figura 3.15, recorre a um amplificador operacional. Optou-se pelo LM386, pois foi concebido para ser um amplificador de *audio single supply,* característica que permite manter a simplicidade do circuito, muito solicitado para aplicações semelhantes à que se está a desenvolver neste projeto [35].

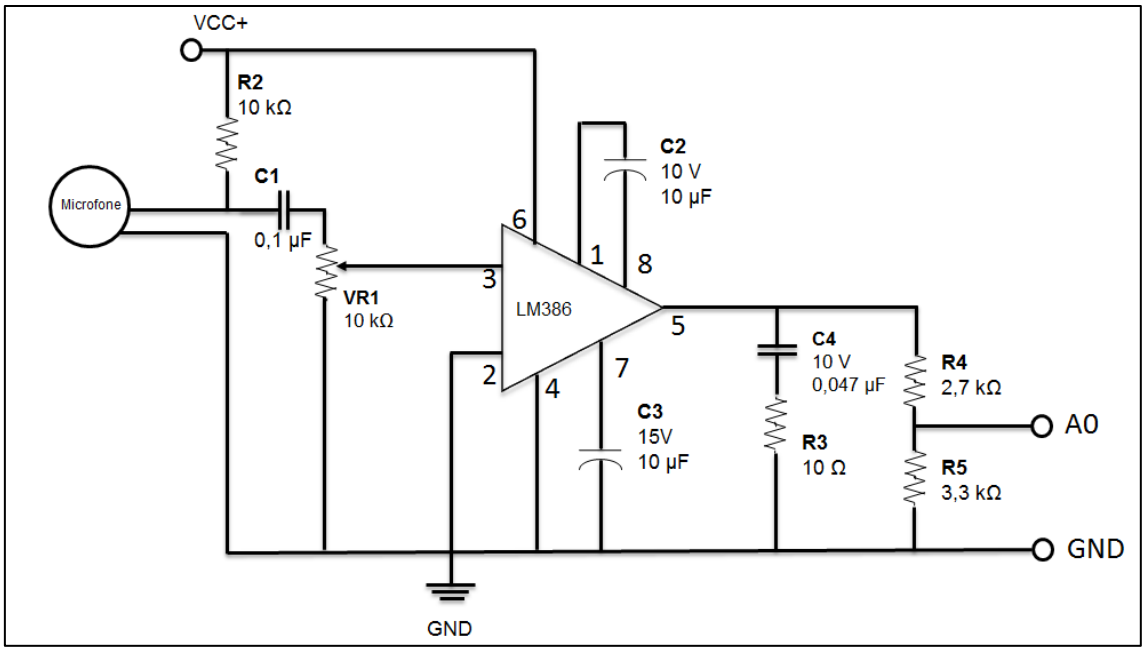

Figura 3.15 - Pré-amplificador para o microfone.

3.3.6. Altifalante

O altifalante escolhido, apresentado na Figura 3.16, tem uma impedância de entrada 8 Ω e pode alcançar no máximo 0,5 W de potência.

Figura 3.16 - Altifalante de 8 Ω e 0,5 W [36].

Os sinais obtidos na saída do Arduíno são de muito baixa amplitude. Por forma a minimizar as correntes de saída do microcontrolador, usa-se um amplificador de saída (no caso do DUE), sendo o circuito apresentado na Figura 3.17. Neste caso, escolheu-se o circuito integrado (CI) TBA820M [37], pela sua simplicidade e facilidade de adaptação. Pode ser alimentado com tensões desde os 3 V e os 15 V. A impedância de entrada é ajustada pela resistência RV1 de 100 kΩ. O ganho do amplificador é ajustado por C2 e R1. Na saída tem-se o condensador de acoplamento C6 para eliminar a componente DC, para não

danificar o altifalante. O condensador C1 serve para filtrar o eventual ruído da tensão de alimentação. A potência de saída é de 1,2 W.

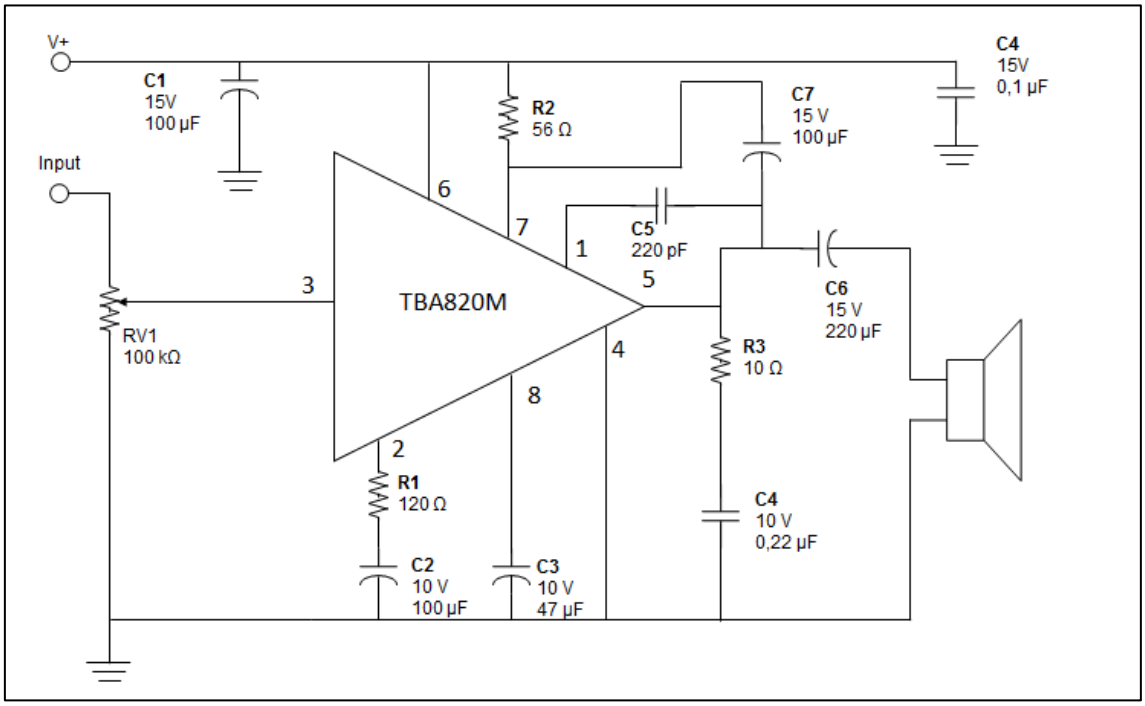

Figura 3.17 - Amplificador de saída para altifalante.

3.3.7. Cartão SD

Para armazenamento das amostras foi utilizado o módulo de memória SD representado na Figura 3.18. Este leitor-gravador de cartões de memória é compatível com os microcontroladores Arduíno e permite comunicar através do protocolo SPI.

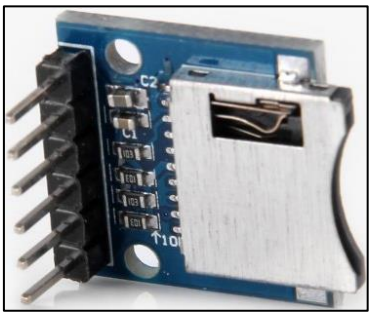

Figura 3.18 - Leitor de cartões SD [38].

O *pinout* e respetiva função de cada pino constam da Tabela 3.4.

Tabela 3.4 - *Pinout* do Cartão SD (Barramento SPI).

Pino	Nome	Abreviatura	Função
1	Ground	*Ground*	Ligado à massa do sistema.
2	Vcc	*Power*	Alimentação do módulo a 5 V
3	MISO	*Master In Slave Out*	Saída SPI do módulo Micro SD *Card* para o Microcontrolador
4	MOSI	*Master Out Slave In*	Saída SPI do microcontrolador para o módulo *Micro* SD *Card*
5	SCK	*Serial Clock*	Recebe o pulso de *clock* para a comunicação SPI
6	SS	*Slave Select*	Pino usado pelo microcontrolador para habilitar ou desabilitar dispositivos no barramento SPI

Este módulo dispõe de 6 pinos, sendo que 2 destes são dedicados à alimentação (a 3,3 V) do dispositivo de memória, enquanto os restantes são dedicados ao protocolo de comunicação. O *pinout* pode ser observado na Figura 3.19.

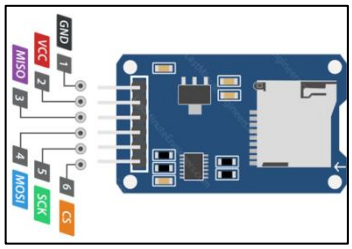

Figura 3.19 - *Pinout* do Leitor-gravador de Cartões SD [39].

3.4. Sistema de transmissão de áudio com nRF24L01+ e Arduíno UNO

Com base nos dados recolhidos, começou-se por desenvolver um sistema que permite a transmissão de áudio através do rádio nRF24L01+ usando o Arduíno UNO.

O protocolo utilizado para estabelecer a comunicação entre ambos foi o SPI. O bloco PC/PSU (*Power Source Unit)* corresponde à alimentação do Arduíno, que na fase inicial dos testes utilizou a alimentação proveniente do computador (através da porta USB) e na fase do protótipo portátil a PSU é uma bateria de 3,7 V. Neste caso é necessário um conversor para se obter os 5 V requeridos pelo Arduíno.

Os sinais gerados pelo microfone foram amplificados utilizando o pré-amplificador baseado no LM386, anteriormente apresentado no circuito da

Figura 3.15. O ajuste fino da amplificação é feito à custa do potenciómetro VR1 cujo objetivo é maximizar a amplificação e controlar o feedback. A montagem realizada pode ser consultada na Figura 3.20.

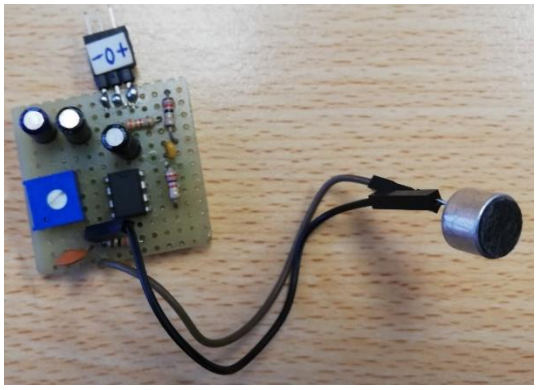

Figura 3.20 - Montagem do pré-amplificador de microfone.

As ligações entre o Arduíno UNO e o nRF24L01+ foram realizadas de acordo com o esquema ilustrado na Figura 3.21, onde se podem observar também os restantes componentes utilizados. O código utilizado foi uma evolução da biblioteca [30] criada pelo programador canadiano TMRh20 e permite a utilização de 2 pinos, tendo-se utilizado o 9 e 10 do Arduíno de 8 bits para gerar som através de um altifalante de 8 Ω 0,5 W no lado recetor Na biblioteca referida existem duas funções que ativam o circuito como emissor ou como recetor, sendo estas rfAudio.transmit() e rfAudio.receive(), respetivamente.

Quando é ativada a função rfAudio.transmit(), é realizada a transmissão do áudio e acende o LED D5.

Quando a função rfAudio.receive() está ativada e é recebido áudio no canal, o sistema pulsa um avisador luminoso na porta D6 e é gerado som no respetivo altifalante. Se não houver áudio a chegar no canal, o LED D6 mantém-se desligado.

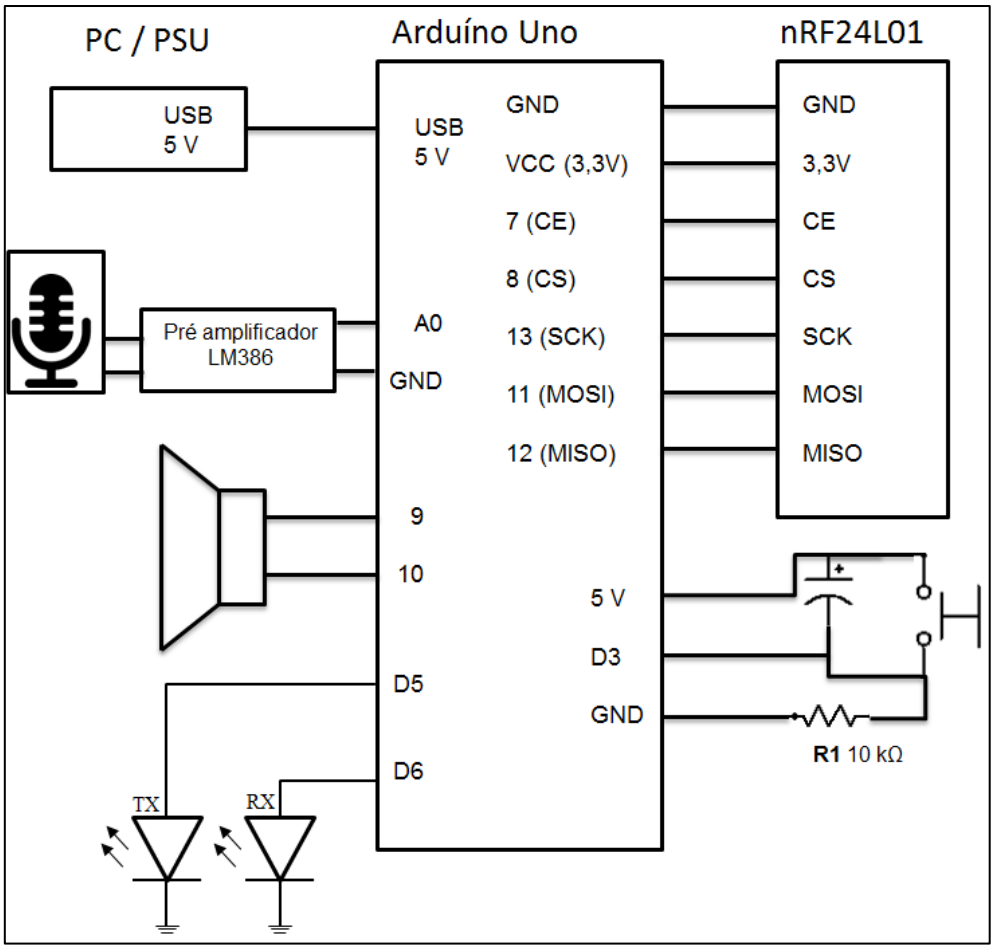

Figura 3.21 - Esquema para controlo de transmissão com botão para falar nRF24L01+ [40] [41].

Foi necessário o estabelecimento de uma técnica de controlo para o envio do áudio. Uma forma encontrada para o fazer foi o recurso à interrupção digital, ativada no pino 3 do microcontrolador. Para enviar áudio, basta pressionar um botão para enviar a mensagem ao destinatário do outro lado do Rádio PTT. Foi usado um botão de pressão, uma resistência de 10 kΩ e um condensador de 100 nF (adicionado ao circuito típico do botão que normalmente se utiliza). Este condensador tem a função de ajudar a prevenir os sinais erráticos do botão, normalmente designados por *debouncing*. Existem outras formas de evitar o *debounce*, no entanto, adicionar o condensador tem um desempenho satisfatório. As interrupções são um atributo dos microcontroladores que em situações como esta podem ser de grande utilidade, pois permitem "interromper" o código para proporcionar, com precisão, eventos cronometrados sem se deteriorar o desempenho do microcontrolador. Neste caso concreto, ao invocar a interrupção é chamada a função áudio rfAudio.transmit() para interromper a função rfAudio.receive() que foi concebida para trabalhar em ciclo contínuo, não

respondendo a inputs de outra natureza. Na Figura 3.22 apresenta-se o respetivo fluxograma. Para enviar voz ou áudio, pressiona-se o botão ligado à porta D3, que invoca a função *Talk*(); e acende o LED D5 (verde).

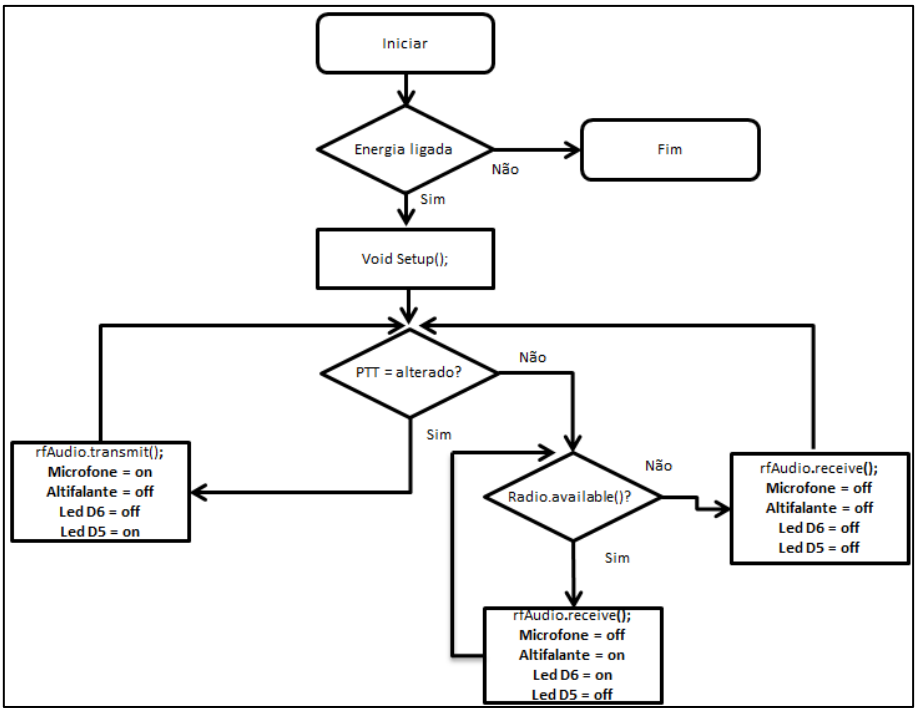

Figura 3.22 - Fluxograma inicial para controlo da emissão / receção.

Neste circuito não é usada uma referência de massa para o altifalante, trabalhando este em modo complementar, comportando-se como um circuito alternado. Para este valor de potência, não se utilizou qualquer amplificação pelo facto de o microcontrolador fornecer a corrente suficiente para produzir som minimamente audível. O altifalante utilizado pode ir até aos 0,5 W.

Foram construídos três protótipos, baseados nesta arquitetura, um com UNO (Figura 3.23) e dois NANO (Figura 3.24) com o objetivo de testar vários modos de transmissão e de receção.

O código necessário para estes protótipos consta no Anexo A.

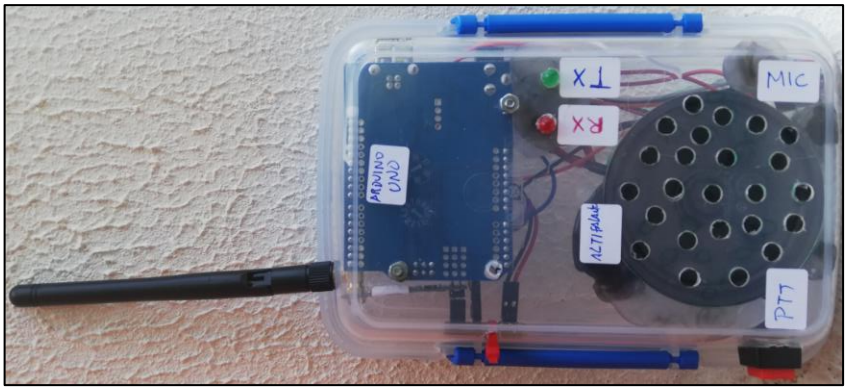

Figura 3.23 - Protótipo #2 com Arduíno UNO.

O Arduíno UNO, devido às suas dimensões e características torna mais fácil os testes de funcionamento e experimentação. Em seguida, devido à sua dimensão reduzida, fator chave na portabilidade, foram usados Arduínos NANO.

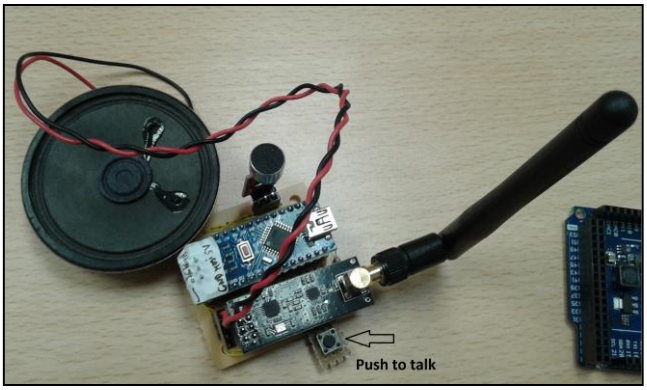

Figura 3.24 - Montagem do protótipo Arduíno NANO com nRF24L01+

Nos primeiros protótipos não foi usado qualquer encapsulamento para proteção uma vez que os mesmos se encontravam estáticos na mesa de testes. No entanto, devido à necessidade de garantir alguma robustez durante os testes de mobilidade, foi necessário criar uma proteção para o efeito. A proteção criada foi o encapsulamento mecânico, como se pode observar na Figura 3.23, que foi aplicada a todos os protótipos.

3.5. Difusão de mensagens pré-gravadas com Arduíno UNO

No sistema anterior a comunicação de áudio ocorre em tempo real. Pode ser útil nalguns cenários enviar mensagens pré-gravadas a partir de um cartão de memória SD. Para isso, considerou-se a possibilidade de criar um sistema baseado em Arduíno da família AVR 8 bits, totalmente compatível com o sistema PTT implementado anteriormente. Após um levantamento de requisitos, foi

decidido utilizar componentes da biblioteca básica já apresentada para fazer envio de áudio via nRF24L01+. O esquema proposto para o emissor pode ser observado na Figura 3.25.

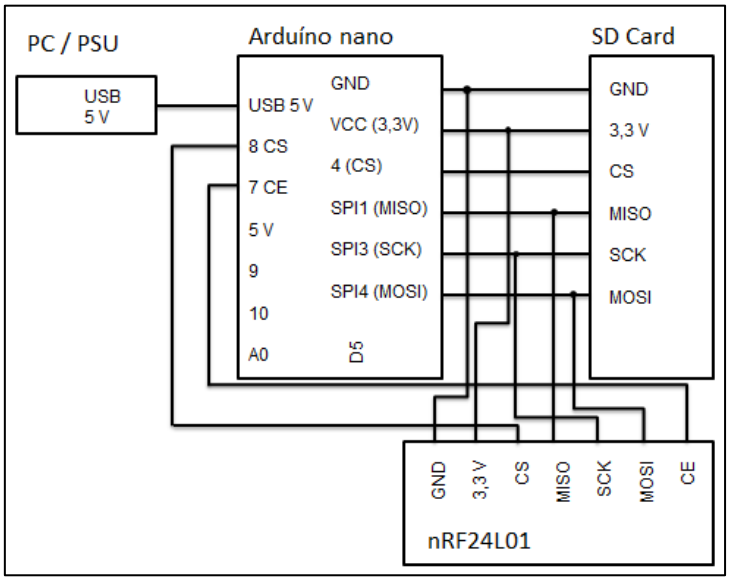

Figura 3.25 - Esquema do emissor de mensagens gravadas em cartão SD.

Para a receção das mensagens foi utilizada a montagem ilustrada pela Figura 3.21, uma vez que é compatível para o efeito.

Para funcionamento deste sistema basta que o recetor esteja ligado em *standby*. No contexto dos testes, a mensagem é enviada apenas uma vez. No entanto, podem existir casos em que esta poderia ser transmitida um número elevado de vezes, até o sistema ser reiniciado ou desligado em resposta à mensagem. O código fonte utilizado no protótipo pode ser consultado no Anexo B e o esquema de funcionamento na Figura 3.26.

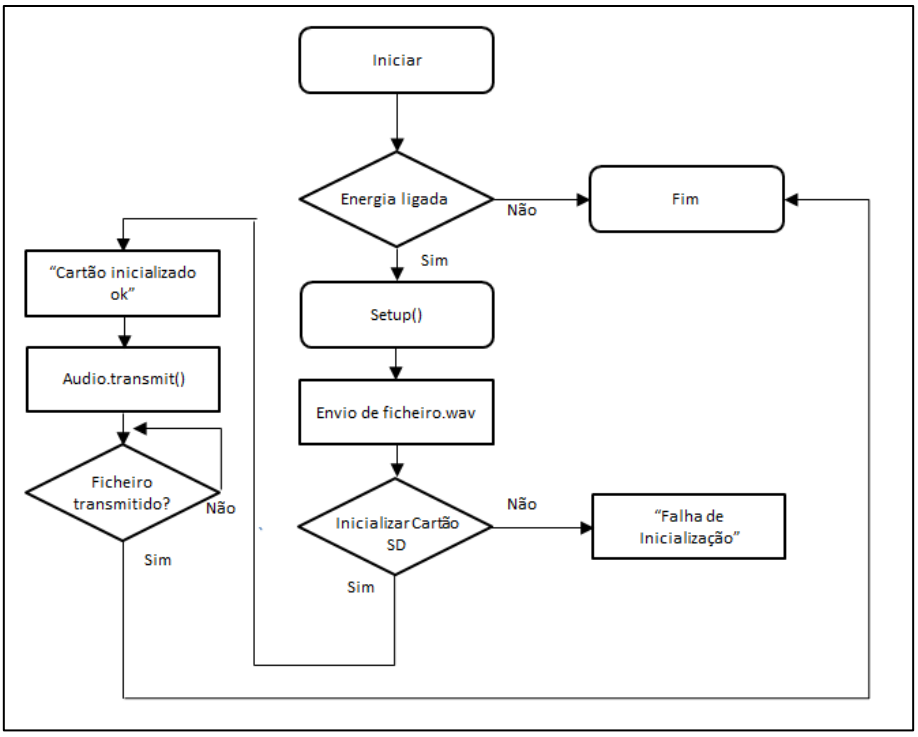

Figura 3.26 - Fluxograma do envio de mensagens gravadas em cartão SD

A ideia fundamental deste sistema é a de poder ser uma alternativa aos sistemas atuais, no que diz respeito à emissão de alertas de natureza urgente, através de altifalantes ou megafones. Imagine-se uma situação de terramoto, onde colapsam todas as vias de transmissão convencionais. Uma solução possível seria a colocação prévia em locais estratégicos, de sistemas devidamente preparados para o efeito, mediante um comando central (por exemplo através da proteção civil) que faria a escolha da mensagem a transmitir. De referir que este sistema também funciona em modo *broadcast*, ou seja, a informação pode ser enviada para diversos recetores.

3.6. Comunicação em direto Arduíno DUE

Inicialmente, nesta etapa, foi abordada a montagem do Arduíno DUE com amplificador de saída, para reproduzir ficheiros de áudio, previamente gravados, a partir do cartão SD, por forma a testar as capacidades do DAC incorporado no microcontrolador. Os ficheiros foram criados com 44,1 kHz de amostragem e com 16 bits.

A montagem foi realizada conforme o esquema que se pode observar na Figura 3.27. Por oposição ao Arduíno UNO, o DUE suporta uma corrente muito

reduzida nos seus pinos DAC (3 mA em vez dos 40 mA). Por essa razão foi imposto o uso do amplificador de áudio.

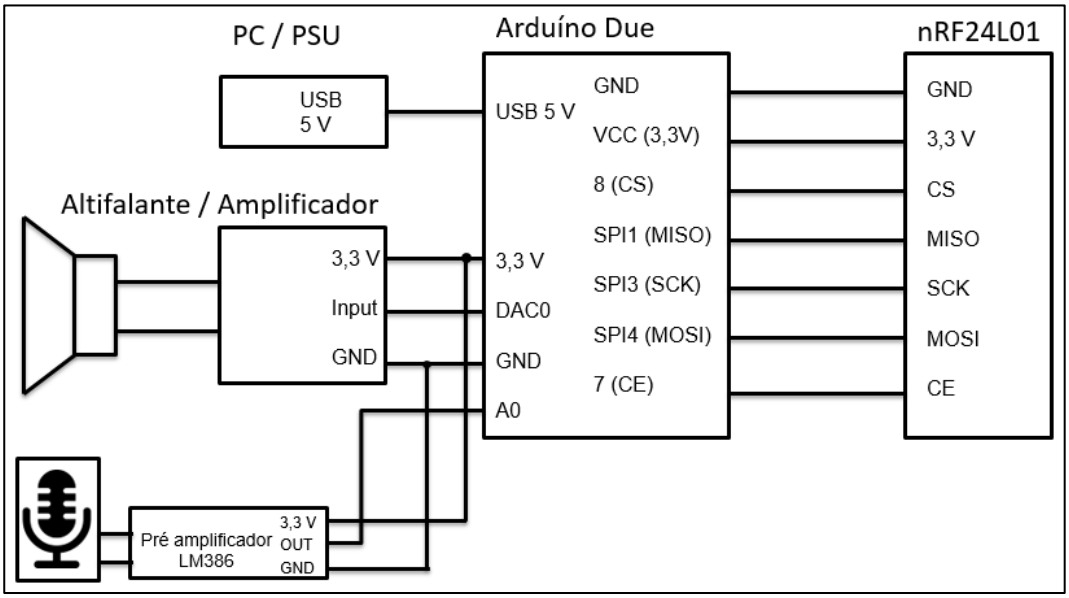

Figura 3.27 - Esquema do protótipo rádio Nrf24L01+ com Arduíno DUE.

A partir do esquema da Figura 3.27, foi construído o protótipo que pode ser observado na Figura 3.28.

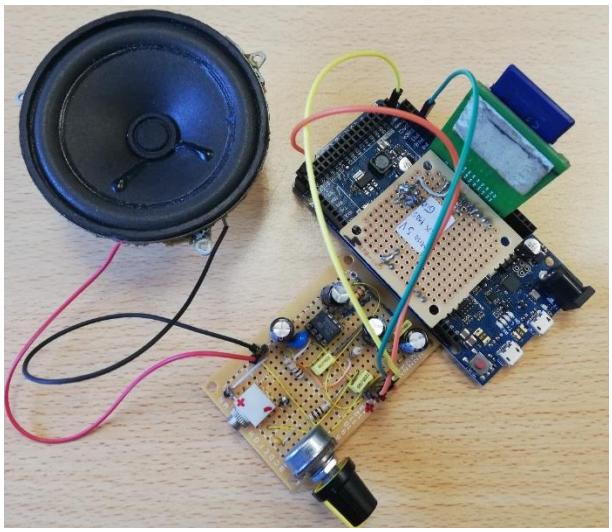

Figura 3.28 - Arduíno DUE com cartão SD - reprodutor de ficheiros WAV

Estando garantido o funcionamento do sistema de reprodução de áudio, foi então preparado um par de protótipos com vista à transmissão bidirecional de áudio com 12 bits e 44,1 kHz. O esquema é semelhante ao utilizado nos Arduíno AVR, com a particular distinção de o DUE apenas comunicar (protocolo SPI) através do barramento dedicado, conforme ilustrado na Figura 3.27.

O fluxograma ilustrado pela Figura 3.29 resume o mecanismo de configuração efetuado. O código foi adaptado da biblioteca áudio original do Arduíno e pode ser consultado no Anexo C.

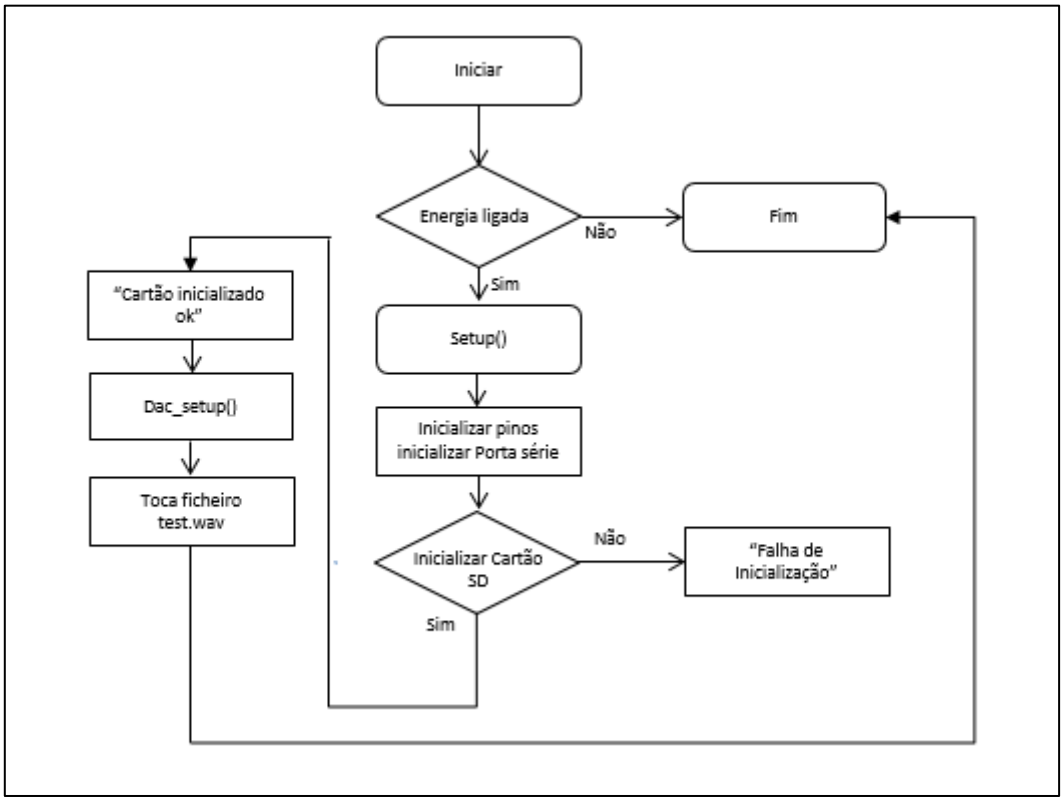

Figura 3.29 - Fluxograma do sistema de reprodução de áudio em Arduíno DUE.

O objetivo deste protótipo foi a demonstração do conceito. Ao contrário da parte relacionada com o AVR, onde se utilizaram bibliotecas de áudio como estrutura base de partida, nesta fase o código base foi escrito de raiz, recorrendo à manipulação de registos, temporizadores e interrupções de modo a permitir a aquisição de áudio de alta qualidade em tempo real. O código para a transmissão e receção estão disponíveis no Anexo D e no Anexo E, respetivamente.

3.7. Gravação e envio de ficheiros de áudio em diferido Arduíno DUE

Recorrendo aos protótipos já experimentados anteriormente no envio de áudio em direto, foi realizada integração com montagem em Arduíno DUE, nR24L01, amplificador de saída e leitor-gravador de cartões para envio diferido das mensagens.

Nesta secção, a abordagem centrou-se na integração do nRF24L01+ com o leitor-gravador de Cartões SD. Um cuidado a ter, tal como na transmissão de áudio gravado em AVR, é a partilha do barramento SPI pelos dois componentes. Na maioria dos projetos existentes em microcontroladores, utiliza-se um componente por barramento. Neste caso concreto, utilizaram-se dois conforme se observa na Figura 3.30.

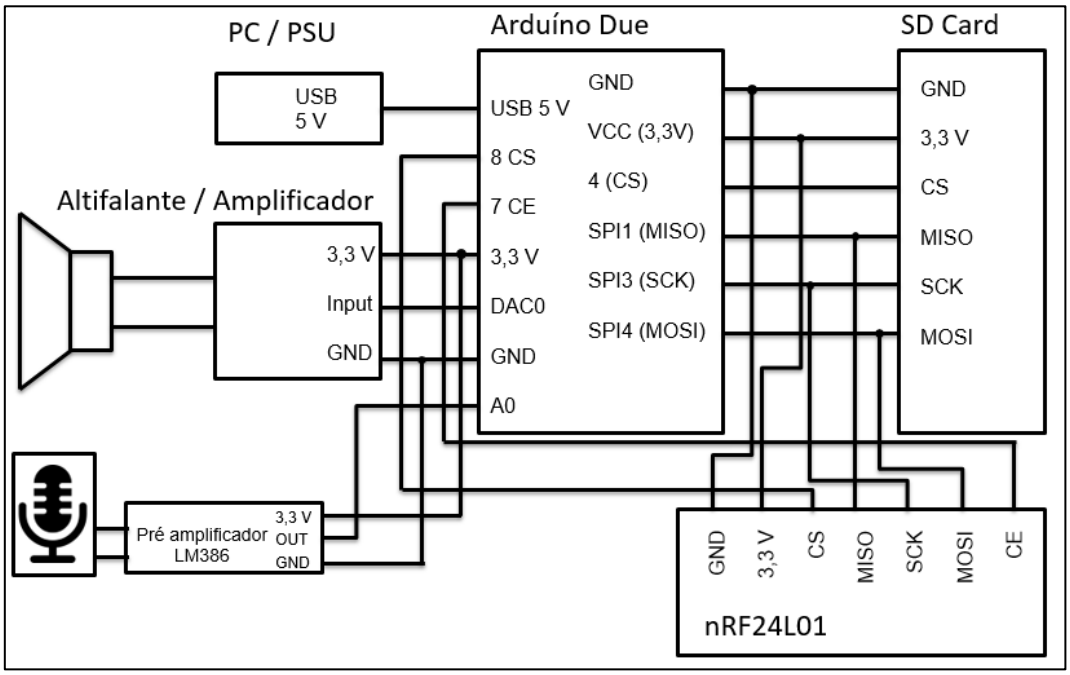

Figura 3.30 - Esquema da integração de dois dispositivos SPI (nRF24L01+ e leitor-gravador de cartões SD).

3.7.1. Gravação de áudio para cartão de memória

Por forma a poder aproveitar as capacidades do sistema para monitorização acústica remota, pode ser necessário amostrar áudio com alta qualidade (12 bits com 44,1 kHz de amostragem) e proceder ao seu envio diferido. Para isso recorreu-se ao protótipo apresentado na Figura 3.30.

Para a gravação de áudio de alta qualidade, utilizou-se o código que consta no Anexo F.

O fluxograma que descreve o processo de escrita da porta analógica para o cartão de memória pode ser observado na Figura 3.31. Trata-se de um processo que recorre à biblioteca SDFAT para otimização da velocidade de escrita.

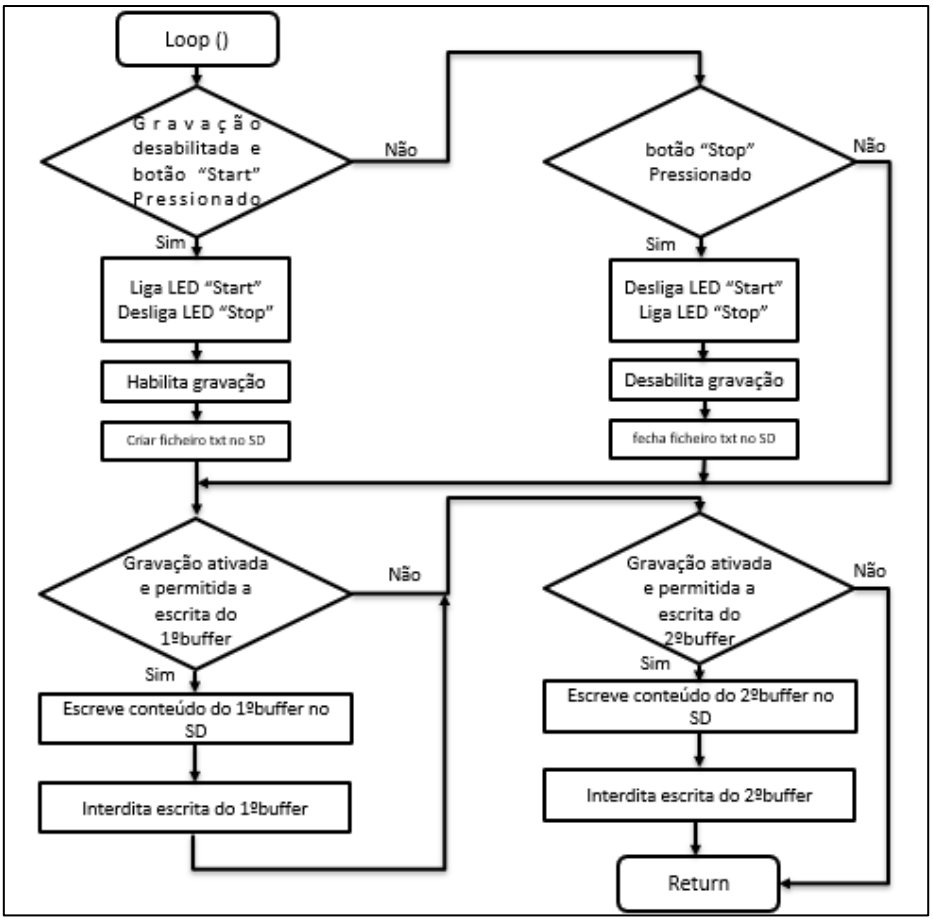

Figura 3.31 - Fluxograma do processo de escrita de áudio em cartão de memória

Após a gravação das mensagens de áudio para o cartão de memória, torna-se necessário um sistema para proceder ao envio das mesmas.

3.7.2. Transmissão de ficheiros de áudio

O sistema a ser implementado tem como objetivo ser uma evolução do anterior baseado em AVR, com a particularidade de poder permitir a gravação de áudio no momento prévio ao envio (implementado em separado), ou o envio de ficheiros para um destinatário que as recebe e conserva. O objetivo é permitir ao utilizador fornecer informação detalhada acerca da sua condição quando envia a mensagem. Além destes condicionalismos, a ideia é também criar uma possibilidade de ser ter um sistema que não seja depende da qualidade do áudio, que sendo transmitido em diferido, não depende diretamente da taxa de transmissão, podendo ser realizado com qualquer débito. Por exemplo, no caso do débito de 250 kbps, permite um alcance três vezes superior aos 2 Mbps, como se demonstrará posteriormente. Na Figura 3.32, é exibido o modo de

funcionamento da transmissão, detalhando a forma como o ficheiro é transmitido após a gravação do mesmo no cartão SD do emissor.

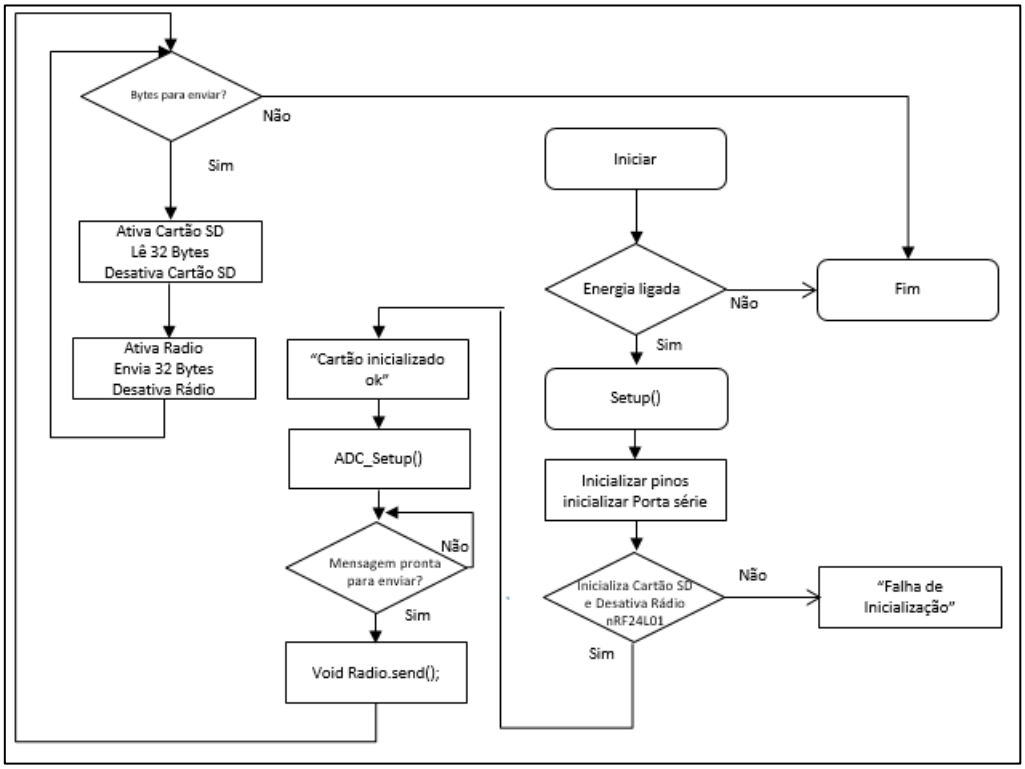

Figura 3.32 - Fluxograma do emissor diferido

No lado do recetor ocorre o processo inverso, ou seja, o rádio mantém-se à escuta do canal e quando deteta informação que lhe seja destinada, abre o cartão SD para escrita e de forma a respeitar o uso do barramento, faz a leitura e escrita alternadamente. Na Figura 3.33 pode ser observado o esquema de funcionamento do sistema, através de um fluxograma. O código do emissor de mensagens em diferido consta do Anexo G.

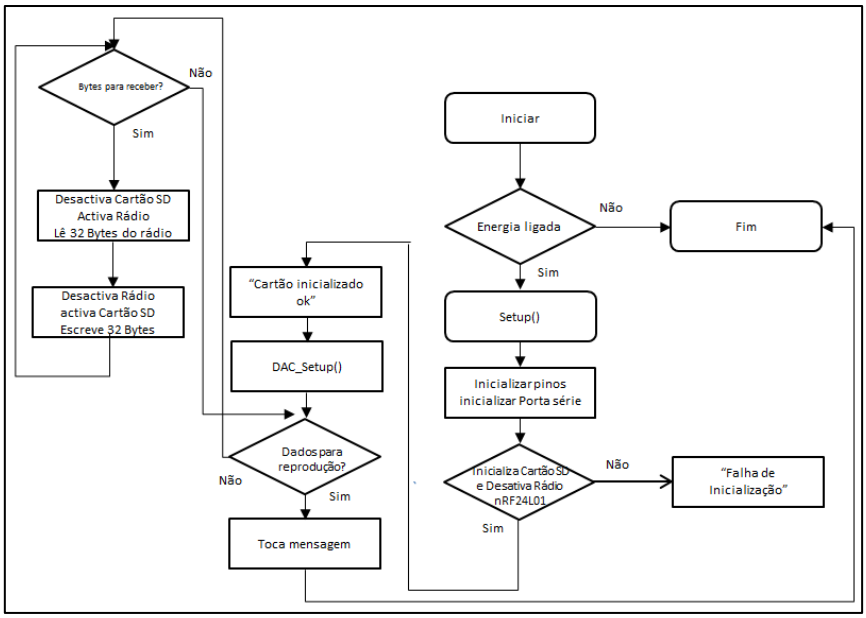

Figura 3.33 - Fluxograma do recetor diferido

Ao receber a mensagem, o sistema inicia a sua reprodução de imediato, podendo ser adicionado qualquer sistema de gatilho para reprodução posterior ou repetição da mensagem. O respetivo código encontra-se no Anexo H.

3.8. Conclusões ao capítulo

Foram montados sistemas distintos com o objetivo de transmitir áudio em direto ou em diferido, com 8 ou 12 bits de resolução, com 16 kHz ou 44,1 kHz de frequência de amostragem do áudio. A última parte consistiu na utilização dos protótipos já montados para a transmissão de áudio a 12 bits, adicionando apenas a interface para o cartão SD, tendo sido desenvolvido um processo de gravação e envio (em separado) de ficheiros de áudio. A evolução dos sistemas propriamente dita teve como objetivo testar vários cenários que se podem colocar. os resultados serão relatados e analisados no capítulo seguinte.

4. Resultados

Neste capítulo apresentam-se os testes e os respetivos resultados obtidos dos vários sistemas desenvolvidos. Foram efetuados testes com os sistemas portáteis com Arduíno UNO de 8 bits, com uma resolução de 8 bits a 16 kHz, em direto e em diferido (mensagens gravadas transmitidas a partir de cartão de memória). Em seguida, fizeram-se testes ao Arduíno DUE de 32 bits a produzir áudio com uma resolução de 12 bits e 44,1 kHz de taxa de amostragem para transmissão em direto com alta qualidade. Finalmente foram realizados testes de envio de mensagens gravadas previamente à transmissão das mesmas, bem como o seu armazenamento no cartão de memória de destino.

4.1. Potência de transmissão

Foram adquiridos rádios a partir de vários fornecedores e fabricantes [12], [42], pelo que se entendeu realizar a medição da potência de transmissão dos mesmos, por forma a garantir um sistema calibrado.

Para medir a potência, fez-se uma montagem com Arduíno UNO e uma base para o rádio nRF24L01+ e ligou-se o mesmo, via conector RP SMA fêmea para tipo N macho, ao medidor de frequências, através de um cabo coaxial de 50 Ω, de acordo com o esquema ilustrado pela Figura 4.1.

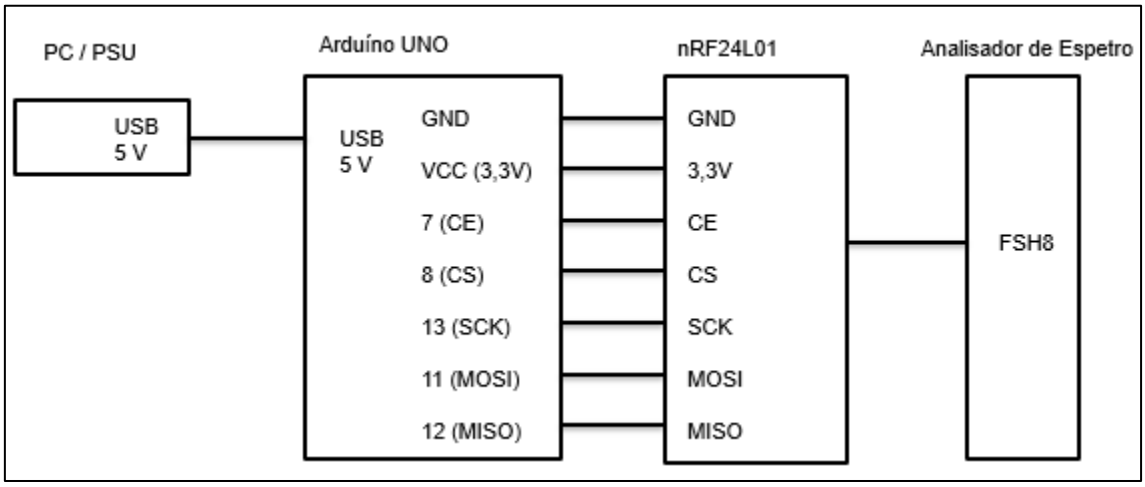

Figura 4.1 - Esquema de sistema para medição de potência dos rádios.

Todo o processo de medição foi controlado por comandos de *software*, enviados através do IDE (*Integrated Development Environment*) do Arduíno. Para efetuar a medição, o rádio foi colocado em modo transmissão, recorrendo

à função radio.write(). Para fazer o ajuste do nível de potência desejado foi utilizada a função radio.setPALevel(). Para esta última, existem quatro argumentos possíveis: RF24_PA_MIN (-18 dBm), RF24_PA_LOW (-12 dBm), RF24_PA_HIGH (-6 dBm), e RF24_PA_MAX (-18 dBm). O valor foi lido no analisador de espetros, em dBm, e por forma a garantir a qualidade dos resultados da potência medida, fizeram-se três séries de testes, os quais não apresentaram desvios entre medições.

Os valores das medições foram registados na Tabela 4.1, onde na primeira coluna se observa o número do rádio, na segunda o valor lido para o nível de PA_MIN (-18 dBm), na terceira o valor para PA_LOW (-12 dBm), na terceira o valor para PA_HIGH (-6 dBm) e na quarta, o valor de PA_MAX (0 dBm).

Tabela 4.1 - Valores de potência por níveis seleccionáveis (via software).

Radio Id	PA_MIN (dBm)	PA_LOW (dBm)	PA_HIGH (dBm)	PA_MAX (dBm)
1	-16,1	-11,8	-6,2	0
2	-16,6	-11	-6,6	0
3	-12,5	-6,2	0,2	8,1
4	-12,5	-6,6	0,1	7,6
5	-17,9	-12,5	0,1	8
6	-17,9	-12,5	0,3	8,1
7	9,6	12,1	16,4	19,9
8	10,2	12,3	16,3	19,1

Dos valores obtidos se conclui que existem alguns desvios dos valores definidos pelo fabricante do dispositivo original da *Nordic,* pelo que foram definidos os níveis de potência, por software, para os valores desejados. Por exemplo, os rádios 7 e 8 não podem ser usados porque, de acordo com a legislação definida pela ANACOM, o valor da potência do conjunto, ganho de amplificador e de antena (2 dBi) não pode ser superior a 10 dBm, na gama de frequências dos 2,4 GHz. Tendo em consideração que foram utilizadas antenas monopolo, com 2 dBi de ganho, a potência do amplificador não pode ser superior a 8 dBm. Em cada teste, foi selecionado o valor de potência por forma a trabalhar dentro dos valores pretendidos.

4.2. Débito de transmissão

Os débitos de transmissão teóricos propostos para o rádio nRF24L01+ são 250 kbps, 1 Mbps e 2 Mbps, em condições ideais. No entanto, no contexto de implementação, as condições nunca são as ideais e dificilmente se

conseguem alcançar tais débitos, razão pela qual se procedeu à medição experimental dos vários débitos reais. As medições foram realizadas recorrendo a dois computadores distanciados entre si com 10 m, cada um deles com um dos protótipos, ligado por USB, por forma a enviar e receber comando série. Os testes foram divididos em dois momentos, transmissão com *acknowledgement* e sem *acknowledgement*, por forma a entender o impacto deste mecanismo de resolução de erros, no débito de transmissão real.

4.2.1. Transmissão com *acknowledgement*

O mecanismo de controlo de erros que utiliza o pacote de ACK (*acknowledgement)* é bastante fiável, quando se trata de garantir a entrega efetiva de informação. Quando o recetor não obtém todos os pacotes da mensagem, ou os recebe com erros, o emissor tenta novamente a entrega com um número de vezes pré-definido, com tempos de espera igualmente definidos. Foram enviados pacotes de 32 bytes em grupos de 10000, medidos os tempos de início e fim da transmissão (no emissor e no recetor) e calculados os débitos, conforme o código no Anexo I. Na Figura 4.2 apresenta-se o teste realizado para o débito nominal de 250 kbps.

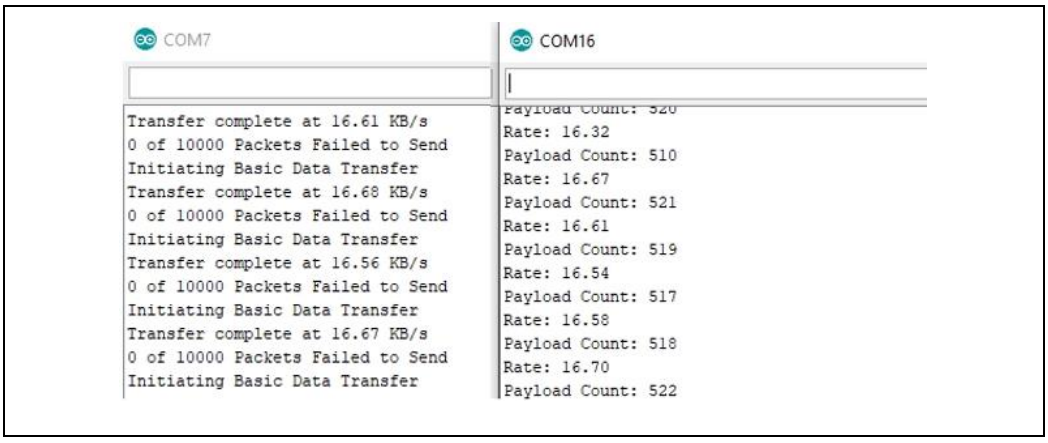

Figura 4.2 - Débito de transmissão a 250 kbps com ACK.

O mecanismo de gestão de ACK faz com que o débito seja significativamente menor, neste caso, cerca de 16 kBps (128 kbps), ou seja apenas 51% do débito nominal.

Na Figura 4.3 apresenta-se o teste realizado para o débito nominal de 1 Mbps com ACK.

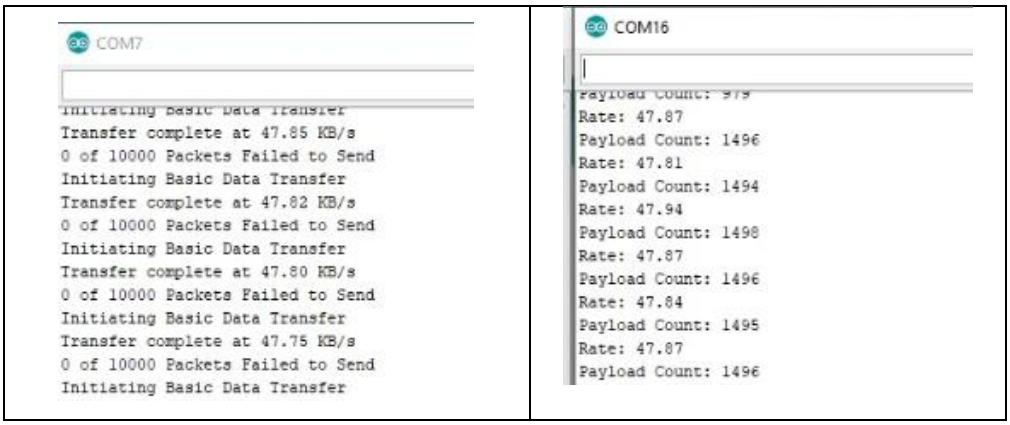

Figura 4.3 - Débito de transmissão a 1 Mbps com ACK.

Para o débito referido, o obtido foi de 47,87 kBps (376 kbps), ou seja 37,6% do débito nominal.

Na Figura 4.4 apresenta-se o teste realizado para 2 Mbps.

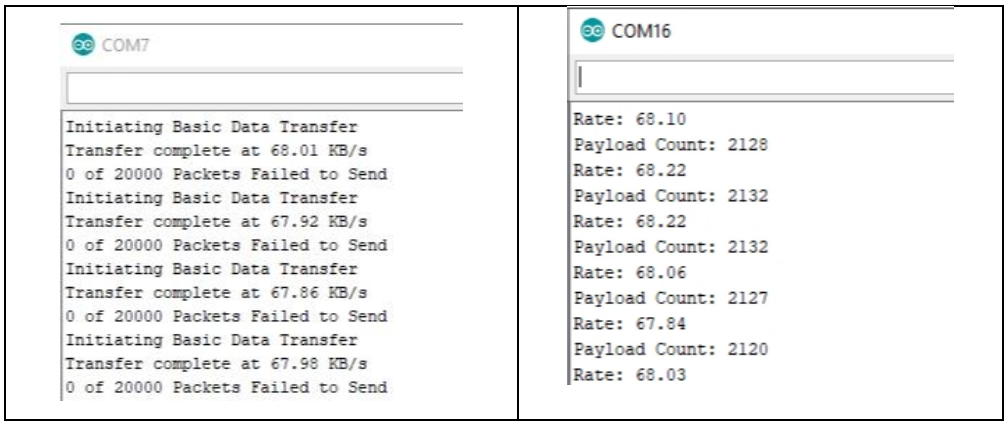

Figura 4.4 - Débito de transmissão a 2 Mbps com ACK.

Para o débito testado, o valor obtido foi de 68,22 kBps (544 kbps), ou seja 27,6% do débito nominal.

Verifica-se que com o aumento do débito nominal o real diminui mais rapidamente, ou seja, em débitos mais baixos, a taxa de transmissão efectiva é maior. Ainda assim, trata-se de um mecanismo que oferece mais garantias, por exemplo no caso de transferência de mensagens escritas ou ficheiros.

4.2.2. Transmissão sem *acknowledgement*

A perda de pacotes está relacionada com o ambiente de propagação, por exemplo, com a existência de obstáculos entre o emissor e o recetor. Para uma transmissão próxima da do espaço livre e a pequena distância a perda de pacotes é baixa. Quando a perda de alguns pacotes não é crítica, um mecanismo

baseado num código de correção de erros pode ser suficiente. Para os testes de envio sem ACK, foram enviadas rajadas de 30000 pacotes e usado o mesmo mecanismo de cálculo usado anteriormente. A Figura 4.5 ilustra o resultado do teste de débito de transmissão a 250 kbps sem ACK.

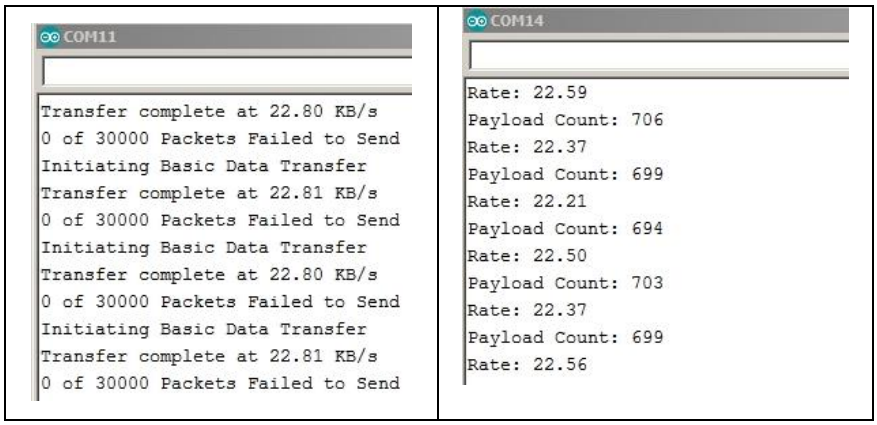

Figura 4.5 - Débito de transmissão a 250 kbps sem ACK.

Os valores obtidos foram de 22 kBps (176 kbps) e são bem mais favoráveis do que aqueles obtidos com ACK. O aproveitamento efetivo da taxa de transmissão foi de aproximadamente 74% do valor nominal.

Em seguida, o teste para 1 Mbps. A Figura 4.6 ilustra o teste realizado para o débito nominal de 1 Mbps sem ACK.

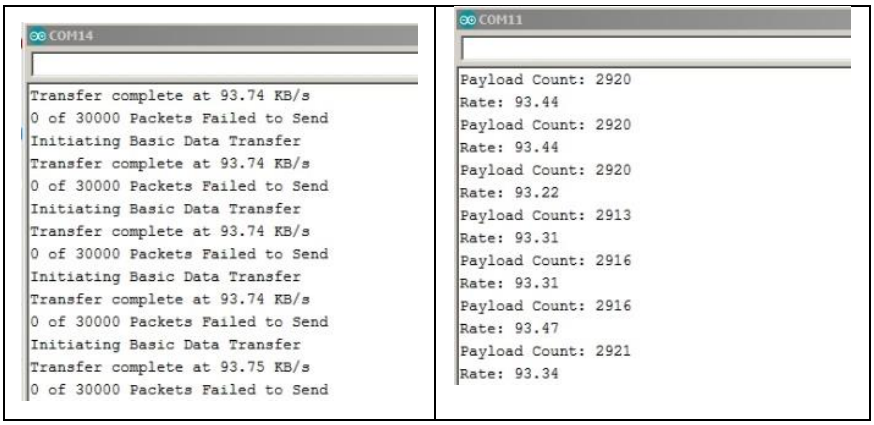

Figura 4.6 - Débito de transmissão a 1 Mbps sem ACK.

O Valor obtido foi de 93 kBps, ou seja, cerca 750 kbps, o que dá um valor de 75% do valor nominal, tal como no teste dos 250 kbps.

A Figura 4.7 ilustra o teste realizado para o débito nominal de 2 Mbps sem ACK.

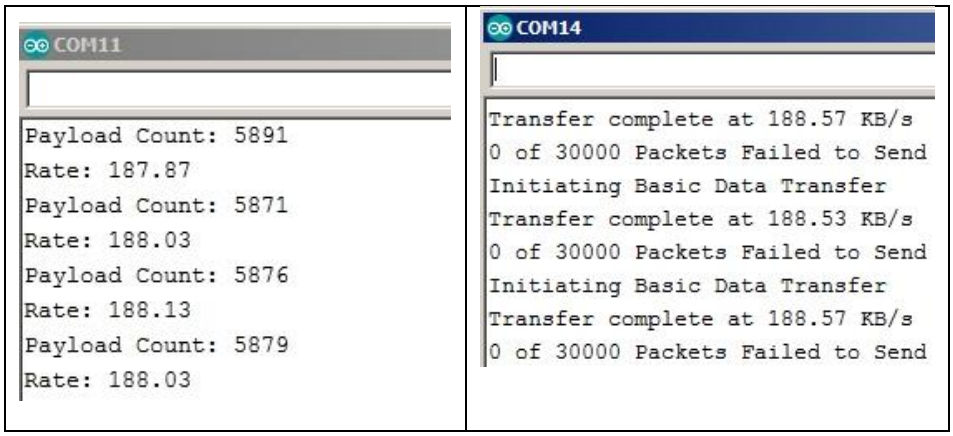

Figura 4.7 - Débito de transmissão a 2 Mbps sem ACK.

O resultado obtido foi de 188 kBps, ou seja cerca 1504 kbps o que representa 75% do valor nominal.

Os resultados obtidos sem ACK foram coerentes entre si na medida em que todos apontam para um débito real de 75% do valor nominal. Estes valores são os mais indicados quando se trata de transmissão de dados em tempo real, onde alguns erros que possam ocorrer sejam detectados por CRC (mecanismo de verificação de erros que ocorrem na transmissão e corrompem a mensagem), que conduz à exclusão dos dados corrompidos.

4.3. Transmissão de áudio em direto

Antes de dar início aos testes de transmissão em campo, fez-se um teste de largura de banda com um sinal de 4 kHz. A frequência escolhida está relacionada com o facto de a voz humana não ultrapassar esse valor. Recorrendo a um osciloscópio digital, comprovou-se o seu correto funcionamento, conforme ilustrado na Figura 4.8.

Figura 4.8 - Análise do áudio a 4 kHz - à entrada (amarelo) e saída do sistema (roxo).

Uma vez que se comprovou o seu correto funcionamento, deu-se início aos testes práticos em campo.

Os testes foram executados de acordo com a intenção inicial de transmitir áudio entre dois ou mais pontos, atendendo sempre à intenção de manter o sistema portátil. Os testes relativos a este sistema tiveram em consideração as condições de propagação, sendo estes o alcance em linha de vista sem interferências do meio (espaço de Fresnel garantido), desempenho em meio urbano e em meio florestal. Foi, ainda, testado o seu funcionamento em viaturas em movimento e dentro de um túnel.

Uma vez que este dispositivo rádio não dispõe de um medidor de RSSI (*Received Signal Strength Indicator*) que indica a intensidade do sinal recebido, o critério de aceitação para medição da receção do sinal foi de natureza qualitativa. Sabendo que no limiar da sensibilidade tem-se, segundo a folha de características do rádio, um bit errado por cada mil que são transmitidos, então se a voz transmitida é escutada sem erro, considera-se que o sistema está a funcionar dentro dos seus limites. Quando os primeiros estalidos começam a ser audíveis, sabe-se que o erro aumentou para fora do limiar do aceitável e considera-se que a receção está com um valor de potência abaixo do mínimo estabelecido pelo fabricante, para o débito de transmissão em causa.

Este sistema foi implementado com o objetivo de transmitir voz, com uma resolução de 8 bits e uma frequência de amostragem de 16 kHz.

4.3.1. Testes em linha de vista

Manipulando a expressão (2.2), e considerando a sensibilidade do sistema definida por P_{rx}, para se obter a distância em função dos outros parâmetros, obtém-se:

$$log_{10}D_r = log_{10}\lambda - log_{10}4\pi - \left(\frac{P_{rx} - P_{tx} - G_{tx} - G_{rx}}{20}\right) \quad (4.1)$$

Substituindo os valores dos parâmetros do sistema, com 2 dBi para o ganho de cada antena, 8 dBm para a potência de transmissão e uma sensibilidade do recetor de -94 dBm, tem-se

$$log_{10}D_r = log_{10}0{,}125\ m - log_{10}4\pi - \left(\frac{-94\ dBm - 8\ dBm - 2\ dBi - 2\ dBi}{20}\right)$$

$$D_r = 1985\ m$$

Foram realizados testes no terreno por forma a determinar os limites reais do sistema. Em Machico, foi colocado um emissor colocado no Pico do Facho e um terminal de receção na Vereda da Torre, na mesma localidade. O traçado direto da ligação está demarcado na Figura 4.9, num mapa obtido a partir do programa Google Earth [44].

Figura 4.9 - Traçado da ligação desde o Pico do Facho até Machico (1700 m) [44].

A linha de vista estava garantida, pelo que seria agora necessário garantir a inexistência de interferências (ausência de qualquer objecto dentro do elipsóide de Fresnel). Recorrendo à equação (2.3) calculou-se a distância ao centro do eixo elipsóide, sendo

$$r = \sqrt{\frac{1700\ m \times 0{,}125\ m}{4}} = 7{,}30\ m$$

O perfil topográfico do cenário em causa está representado na Figura 4.10, onde foram obtidos 3 pontos, cada com a altura do eixo e altura do solo, para calcular a altura relativa ao solo, doravante designada por h.

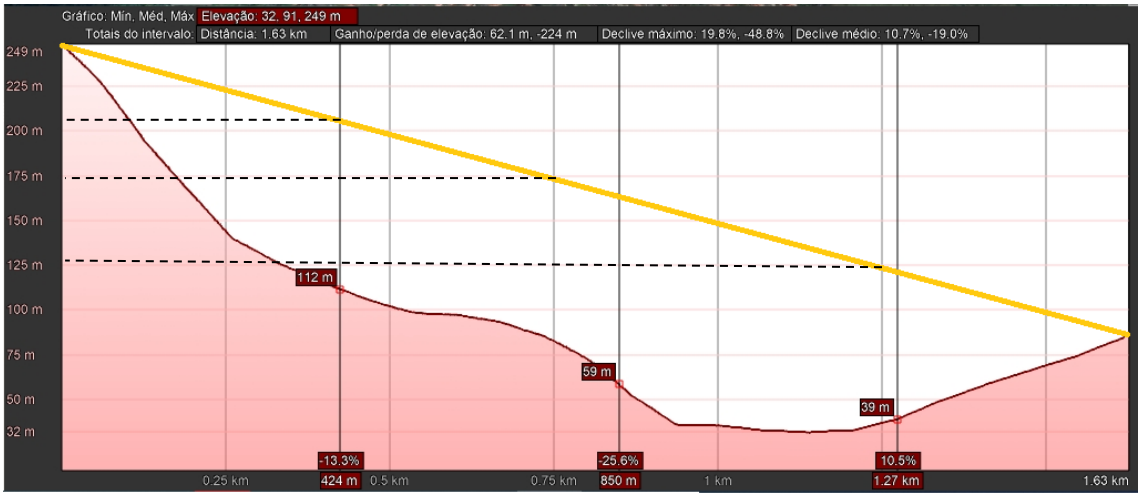

Figura 4.10 - Perfil topográfico Pico do Facho - Machico [44].

Os valores aproximados da altura do solo e do eixo nos pontos assinalados foram registados na Tabela 4.2, que servem para verificação do espaço no elipsóide de Fresnel.

Tabela 4.2 - Pontos de verificação de espaço livre no elipsóide de Fresnel

	Ponto 1	Ponto 2	Ponto 3
Distância (d)	424 m	850 m	1270 m
Eixo	210 m	175 m	125 m
Solo	112 m	59 m	39 m
h	98 m	116 m	86 m

Por observação da Tabela 4.2, verifica-se que o h varia entre 86 a 116 m, valor muito superior a r que pela equação (2.3) é de 7,30 m. Ou seja desde que se tenha este valor acima e abaixo da linha que une as duas antenas (exemplificado na Figura 2.14), sem qualquer obstáculo, considera-se espaço livre de Fresnel.

Por observação de todos os dados recolhidos, constata-se que neste teste verificam-se as condições de espaço livre. De referir ainda que a esta distância, não se sentiu qualquer alteração na qualidade da voz recebida.

Os cálculos teóricos apontam para uma distância máxima de, aproximadamente, 2000 m, por forma a garantir uma sensibilidade adequada ao sistema recetor.

Por forma a continuar a verificação do desempenho do sistema, repetiu-se o teste novamente a partir do pico do facho em direcção ao miradouro Francisco Alves Nóbrega, conforme Figura 4.11. O objectivo concreto deste teste foi a verificação das condições de funcionamento alterando apenas a localização geográfica do recetor e a distância de ligação para 1400 m.

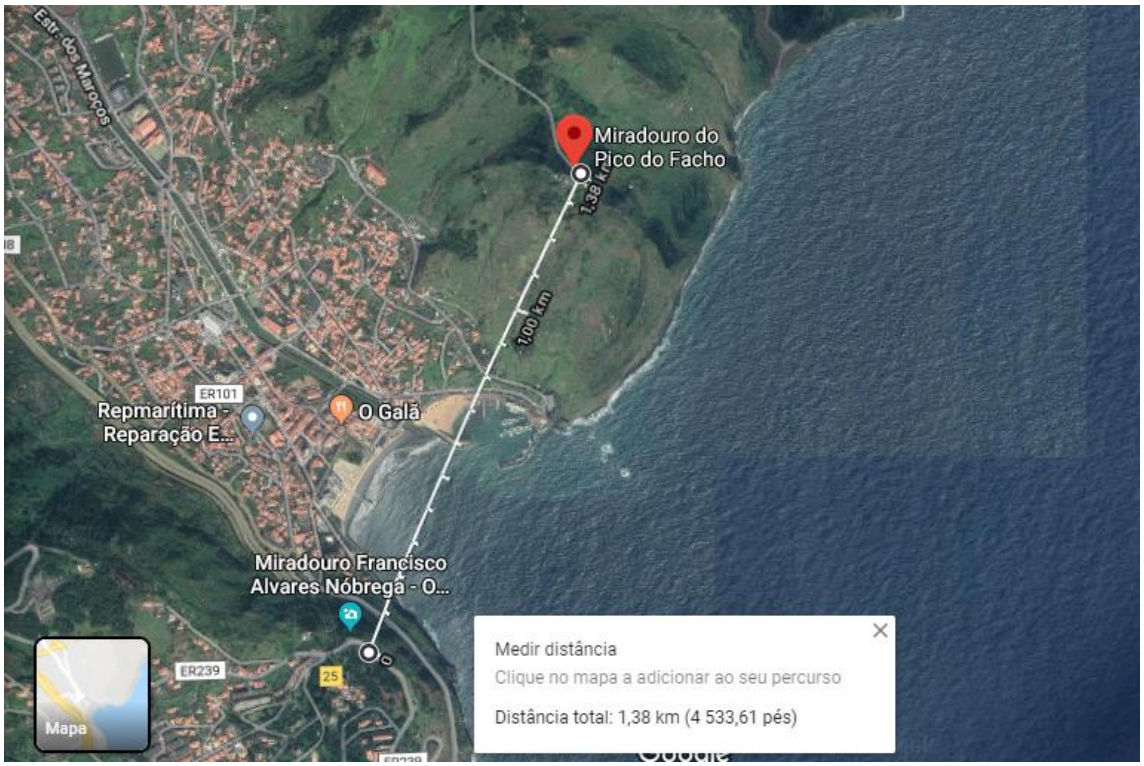

Figura 4.11 - Ligação Pico do Facho - Miradouro Francisco Alves Nóbrega (1400 m) [44].

O perfil topográfico do traçado referente ao percurso desta ligação é algo distinto do anterior. Conforme se pode observar na Figura 4.12, este é aparentemente mais vantajoso, uma vez que é ainda mais desimpedido que o anterior.

Figura 4.12 - Perfil topográfico da Ligação Pico do Facho - Miradouro Francisco Alves Nóbrega [44].

Os valores para o cálculo de h foram obtidos nos pontos e registados na Tabela 4.3 para verificação do espaço no elipsóide de Fresnel.

Tabela 4.3 - Pontos de verificação de espaço livre no elipsóide de Fresnel

	Ponto 1	Ponto 2	Ponto 3
Distância (d)	351 m	700 m	1050 m
Eixo	225 m	190 m	150 m
Solo	97 m	5 m	2 m
(h)	128 m	185 m	148 m

Para esta distância, o valor de r, calculado pela equação (2.3) é de 6,6 m, ou seja, muito inferior a qualquer ponto de h no traçado, relativamente ao solo.

Também neste teste se verificaram as condições ideais de funcionamento para este sistema que funcionou dentro do esperado, obtendo-se na receção um som claro e perfeitamente audível.

Em seguida, foi realizado um teste onde uma ligação foi estabelecida entre o pico da Atalaia e o edifício Moinhos Park na zona da Assomada no Caniço, conforme se pode observar na Figura 4.13.

Figura 4.13 - Testes em linha de vista a 1540 m [44]..

De acordo com o perfil topográfico ilustrado pela Figura 4.14 não é possível garantir um elipsóide de Fresnel sem obstáculos. O que se pretendeu testar foi o cenário onde existia linha de vista entre ambos os sistemas, havendo

a existência de pelo menos uma interferência causada pela existência de objetos no elipsóide de Fresnel.

Figura 4.14 - Perfil topográfico da Ligação Pico Atalaia - Moinhos Park [44].

Os valores para o cálculo de h foram obtidos nos pontos e registados na Tabela 4.4 para verificação do espaço no elipsóide de Fresnel.

Tabela 4.4 - Pontos de verificação de espaço livre no elipsóide de Fresnel (1560 m)

	Ponto 1	Ponto 2	Ponto 3
Distância (d)	390 m	780 m	1170 m
Eixo	90 m	152 m	220 m
Solo	51 m	152 m	202 m
(h)	39 m	0 m	18 m

Apesar de não ser possível garantir um meio limpo, livre de interferências, a comunicação foi estabelecida com sucesso, percetível sem erros. Teoricamente esta comunicação poderia apresentar problemas, devido às condições orográficas já apresentadas não serem as mais favoráveis para o estabelecimento da ligação. No entanto, tal não aconteceu e isso significa que este tipo de comunicação é bastante robusto, mesmo nestas condições menos favoráveis.

No processo de calibração da potência dos rádios, encontraram-se rádios com uma potência de transmissão máxima de 20 dBm, que como já foi dito, não podem ser usados em Portugal. No entanto, estando a ser realizado um estudo específico no domínio de apoio a pessoas, em contexto de emergência, ou

outras de superior interesse, poderia ser solicitada uma autorização ao regulador nacional, para uma potência máxima superior.

Com base numa hipotética exceção, foi preparado um cenário de teste com o objetivo de avaliar o desempenho do sistema a vários km de distância. A partir desse pressuposto, recorrendo aos rádios 5 e 6 na máxima potência (cerca de 20 dBm), foi efetivada a ligação entre os pontos ilustrados na Figura 4.15. A distância máxima que estes dispositivos podem alcançar em condições ideais calcula-se pela)

$$log_{10}D_r = log_{10}0.125\ m - log_{10}4\pi - \left(\frac{-94\ dBm - \ 20\ dBm - 2\ dBi - 2\ dBi}{20}\right)$$

$$D_r = 7900\ m$$

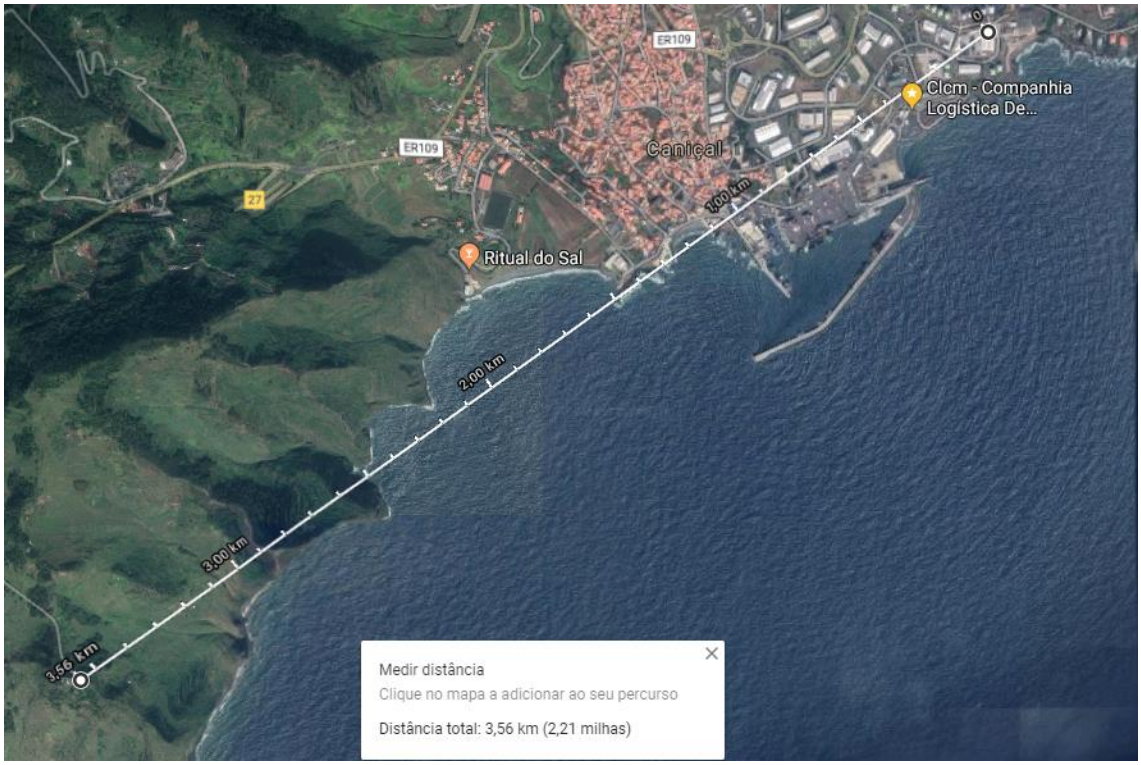

Figura 4.15 - Testes em linha de vista ligação Pico do Facho ao Caniçal (3560 m) [44].

O perfil topográfico do cenário em causa está documentado na Figura 4.16, onde foram obtidos 3 pontos, os quais representam a distância do eixo ao solo.

Figura 4.16 -Perfil topográfico da ligação Pico do Facho ao Caniçal [44].

Os valores para o cálculo de h foram obtidos nos pontos e registados na Tabela 4.5 para verificação do espaço no elipsóide de Fresnel.

Por análise do perfil topográfico da ligação observa-se que aos 825 m existe um ponto onde à primeira vista poderia existir uma perturbação do espaço, uma vez que aponta uma elevação de 195 m. No entanto, nesse mesmo ponto existe uma distância de 30 metros, o que garante uma margem suficiente para cobrir no máximo 9 metros, que seria o valor minimo a garantir no ponto médio do elipsóide.

Tabela 4.5 - Pontos de verificação de espaço livre no elipsóide de Fresnel (3560 m)

	Ponto 1	Ponto 2	Ponto 3
Distância (d)	826 m	1800 m	2700 m
Eixo	225 m	152 m	90 m
Solo	195 m	0 m	12 m
(h)	30 m	152 m	78 m

Esta distância foi conseguida sem qualquer dificuldade, tendo sido registada a audição da voz transmitida, clara e sem interrupções.

Com vista à não perturbação do espectro radioelétrico, estes testes foram feitos apenas durante o tempo absolutamente necessário ao registo das conclusões.

A fase seguinte consistiu no aumento da distância para um valor ainda maior, para perto dos 5000 metros. Na Figura 4.17, pode ser observado o traçado da ligação entre Machico e o Caniçal, com 4900 metros de distância.

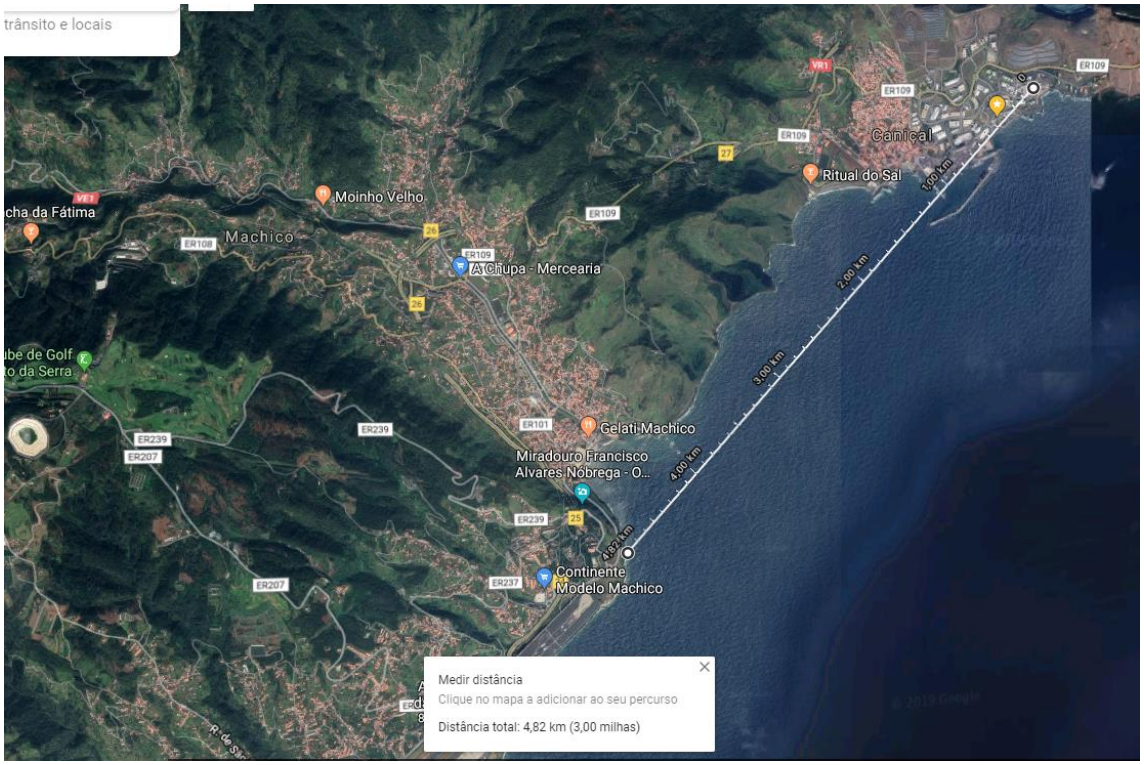

Figura 4.17- Testes em linha de vista a 4900 m, emissor a 70 m de altitude e o recetor a 46 m [44].

O perfil topográfico da ligação entre o Caniçal e Machico pode ser observado na Figura 4.18 onde se pode constatar que não existem obstáculos ao longo do traçado da ligação.

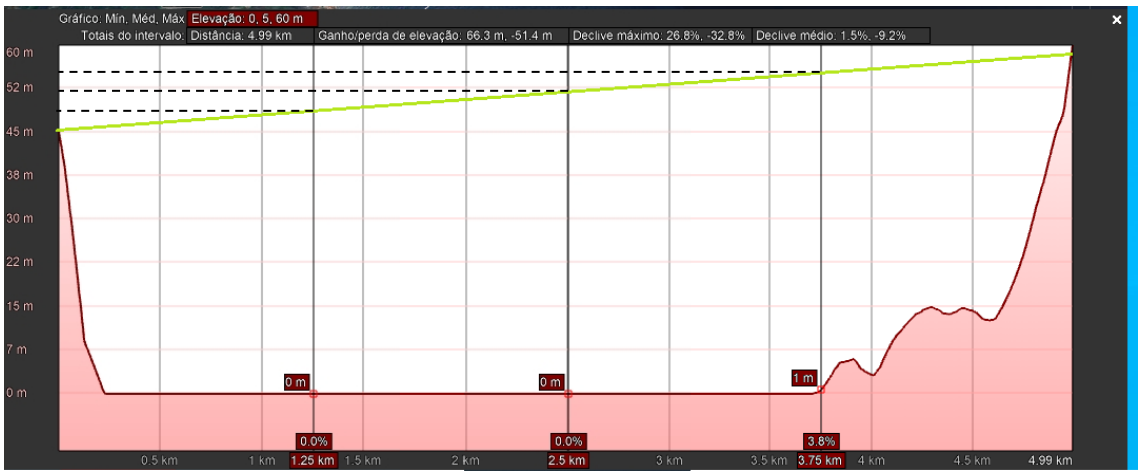

Figura 4.18 - Perfil topográfico da ligação Caniçal a Machico (4900 m) [44].

Os valores para o cálculo de h foram obtidos nos pontos e registados na Tabela 4.6 para verificação do espaço no elipsóide de Fresnel.

Tabela 4.6 - Pontos de verificação de espaço livre no elipsóide de Fresnel

	Ponto 1	Ponto 2	Ponto 3
Distância (d)	1250 m	2500 m	3750 m
Eixo	47 m	52 m	56 m
Solo	0 m	0 m	24 m
(h)	47 m	52 m	56 m

Verifica-se que o valor de h é muito superior ao valor de r que para esta distância seria de 12,5 metros.

Por observação de todos os dados recolhidos, constata-se que o desempenho destes protótipos enquadra-se nos resultados dos primeiros testes realizados, onde foi possível estabelecer uma transmissão de boa qualidade. Os cálculos teóricos apontam para uma distância máxima de 7900 m aproximadamente, por forma a garantir uma sensibilidade adequada ao sistema recetor. Na distância máxima experimentada, a cerca de 5 km, o desempenho foi satisfatório, mantendo o sistema a funcionar sem quebras ou erros, durante todo o teste.

4.3.2. Testes de funcionamento em meio urbano

Após a confirmação da operação do sistema em linha de vista, foram realizados testes de funcionamento em ambiente urbano (caracterizado por edifícios, estruturas metálicas, veículos e objetos de composição diversa).

O teste foi realizado num espaço com obstáculos de metal e paredes de betão. Um terminal foi colocado num ponto fixo, com uma montagem a obter áudio (voz gravada) a partir de um leitor de áudio com saída analógica. Em seguida, em vários locais foi testado o funcionamento dos protótipos móveis para verificação do seu alcance nas duas taxas de transmissão 250 kbps e 2 Mbps.

Nos testes realizados, o nó portátil deslocava-se na mão de um utilizador que verificava a receção do áudio em cada instante. Na Figura 4.19 estão ilustrados os vários percursos efetuados. Os trajetos a vermelho ilustram o percurso realizado para os 250 kbps e a verde o dos 2 Mbps.

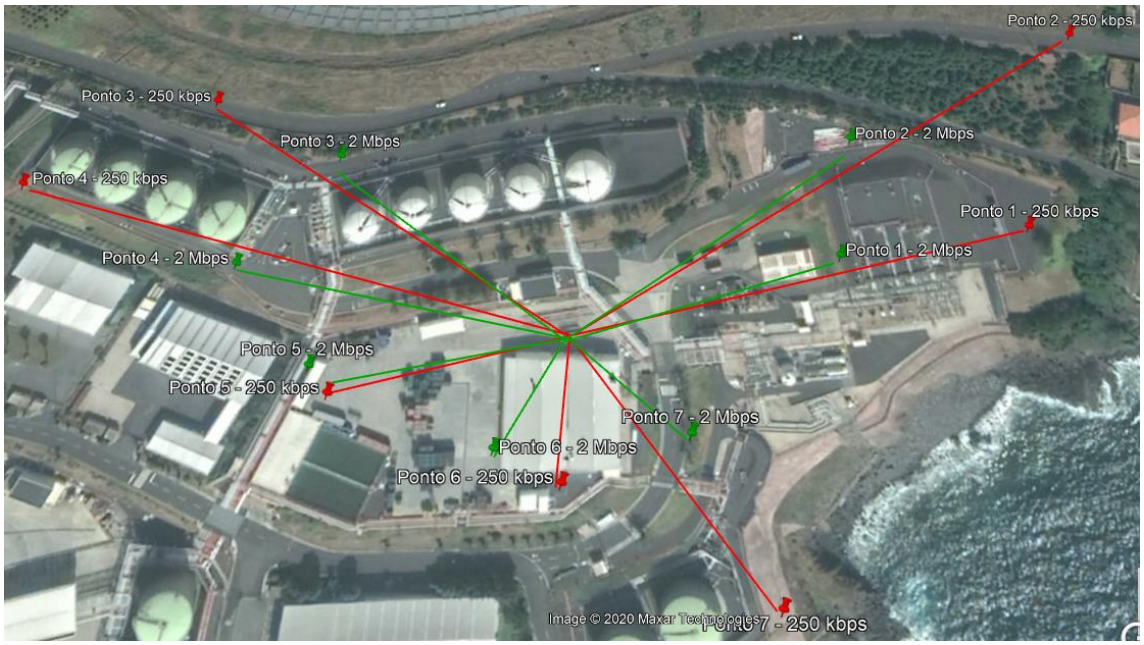

Figura 4.19 - Testes de transmissão e receção com a montagem Arduíno NANO e UNO [44].

A primeira configuração testada foi a de 250 kbps, aquela que garante uma maior sensibilidade do sistema recetor. Nos trajetos assinalados a vermelho, foi possível obter áudio com uma qualidade bastante boa, isenta de ruído ou falhas. O alcance máximo para cada teste foi assinalado no final de cada trajeto, com um pino. Considerou-se máximo alcance quando o áudio se tornou impossível de interpretar. A perda de qualidade, começava-se a sentir normalmente entre 20 e 30 m da distância limite.

Repetiu-se o teste, alterando o débito para 2 Mbps, onde o alcance teve uma redução imposta pela menor sensibilidade do rádio. Os critérios utilizados foram os mesmos. Na Tabela 4.7 foram registados os resultados dos testes de alcance.

Tabela 4.7 - Teste comparativo de alcance entre 250 kbps e 2 Mbps.

ID teste	250 kbps	2 Mbps
	Distância (m)	Distância (m)
1	183	109
2	255	140
3	186	124
4	229	133
5	91	86
6	60	55
7	129	60

Num outro teste, realizado com o débito de 250 kbps, tinha-se a intenção de obter o máximo alcance do sistema num ambiente diferente. Este teste foi realizado no parque de estacionamento da Universidade da Madeira, caracterizado por um vasto conjunto de obstáculos, tais como carros em movimento, carros estacionados, alunos a usar o telemóvel com Wi-Fi, entre outros. Tal como no teste anterior, a antena estava colocada a cerca de 2 m do chão. Neste contexto, foi realizado o teste recorrendo a 3 terminais, 1 emissor e 2 recetores. Durante o teste, um utilizador avançou até ao limite da captação e outro seguia cerca de 10 m atrás por forma a aferir o processo. O sinal perdeu-se por completo aos 235 m, conforme ilustrado pela Figura 4.20.

Figura 4.20 - Testes de funcionamento no parque de estacionamento da Universidade da Madeira [44].

Os resultados obtidos foram semelhantes em ambos os cenários, uma vez que os elementos de interferência são os mesmos.

4.3.3. Testes num ambiente com vegetação

Como o sistema de transmissão de áudio também pode ser instalado em meios com vegetação, como uma floresta, fez-se um teste de transmissão nesse tipo de meio. Utilizando o jardim em frente à Universidade da Madeira, fixou-se o emissor, com um dispositivo reprodutor de voz gravada, e os terminais

portáteis foram deslocados ao longo da vegetação por dois utilizadores, distanciados entre si por um máximo de 30 m. O emissor foi instalado num ramo de uma árvore, rodeada de vegetação. Durante quase todo o percurso, a voz emitida foi ouvida de forma clara e percetível. No entanto, ao passar a zona dos 100 m, os estalidos intensificaram-se e a perda de sinal tornou-se notória aos 115 m.

O espaço percorrido e a distância entre o ponto de emissão e o máximo alcance podem ser observados na Figura 4.21.

Figura 4.21 - Testes de recepção e alcance em ambiente florestal [44].

Estes resultados enquadram-se no perfil do meio florestal que apresenta uma forte atenuação nos sinais de rádio. Experiências realizadas anteriormente e descritas num trabalho publicado [45][fundamentam esta afirmação, conforme se pode observar na Figura 4.22. Segundo os resultados apresentados nesta figura, a atenuação suplementar para antenas de ganho baixo é de cerca de 25 dB para a distância dos 100 m.

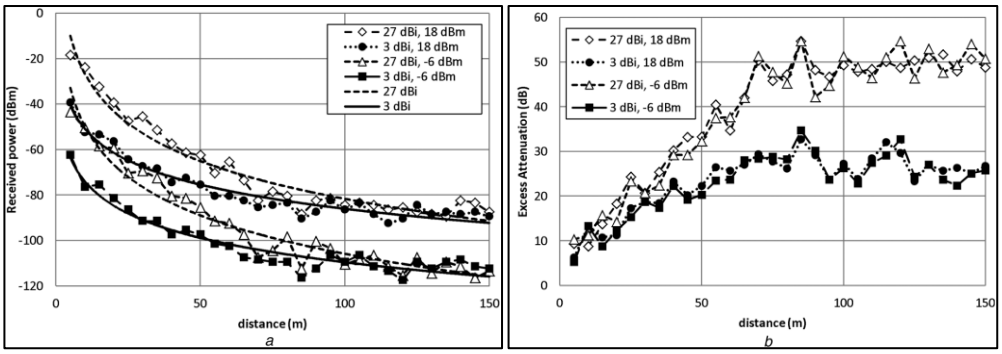

Figura 4.22 - Perdas de percurso em ambiente florestal na zona de testes [45].

No ambiente com vegetação as perdas por atenuação são muito mais elevadas a distâncias curtas. Na medida em que o recetor se desloca para longe da origem a potência recebida diminui de forma logarítmica.

4.3.4. Testes realizados com viaturas em movimento

Por forma a completar a variedade de testes efetuados, dois terminais foram instalados em duas viaturas que se fizeram deslocar em várias estradas da ilha da Madeira. Os terminais foram emitindo alternadamente em vários momentos da deslocação, em vários locais.

As configurações utlizadas neste teste foram idênticas às anteriores: 250 kbps de débito máximo admissível nas transferências a realizar.

Nos testes realizados, não existiam viaturas entre o emissor e o recetor, pelo que quase sempre se manteve uma ligação em linha de vista, como se pode ver na Figura 4.23 (sujeita às perturbações do plano de terra e do campo de Fresnel não garantido).

Figura 4.23 - Esquema típico de teste para viaturas em movimento [44].

Mantendo a distância mínima de 500 m, foi possível, colocando as antenas no exterior do automóvel, manter uma comunicação clara e sem erros. Quando a distância superava os 800 m, a comunicação perdia-se.

As distâncias máximas obtidas são relativamente mais baixas devido aos vários fatores que influenciam a transmissão, nomeadamente a atenuação introduzida pelo meio envolvente devido ao fenómeno multipercurso. Em todos os testes ambas as viaturas circularam à mesma velocidade para manter a distância.

4.3.5. Testes em túneis

Um cenário possível de necessidade de apoio é por exemplo os túneis (de circulação automóvel, de levadas ou circulação pedonal). Perante esse cenário, decidiu-se testar o desempenho do sistema no túnel da estrada antiga do Caniçal, onde foram realizados testes por dois utilizadores a pé.

No Caniçal, um utilizador ficou do lado de Machico, com um emissor de voz permanente e um terminal com PTT. Outro utilizador seguiu a pé nos 800 m até ao Caniçal.

A colocação dos protótipos de teste no Caniçal foi efetuada de acordo com a Figura 4.24.

Figura 4.24 - Túnel do Caniçal (pontos de colocação dos protótipos de teste).

Como se observa na Figura 4.24, pode-se verificar que existe linha de vista entre ambos os pontos.

O traçado da ligação Caniçal para Machico, dentro do túnel da estrada regional, que ilustra a disposição do emissor e do recetor, bem com a respetiva distância entre ambos, pode ser observado na Figura 4.25.

Relativamente ao espaço de Fresnel, o túnel tem cerca de 5 m de altura e um comprimento de 826 m. Pode-se constatar, através do cálculo do r, seria necessário que este tivesse pelo menos 10 m de raio, por forma a garantir a distância necessária do eixo ao obstáculo.

Figura 4.25 - Traçado subterrâneo do Tunel velho do Caniçal (826 m) [44].

O sinal manteve a qualidade até aos 600 m, apresentando uma perda progressiva a partir dessa distância. Aos 820 m, o sinal perdeu-se.

4.4. Difusão de mensagens pré-gravadas

Para a realização de testes na difusão de mensagens pré-gravadas, foi construído um protótipo específico, com um leitor-gravador de cartões e um rádio nRF24L01+ ligados ao Arduíno UNO, para ser usado como emissor (Figura 3.25). Do lado do recetor foram usados os protótipos utilizados nos testes anteriores, por serem totalmente compatíveis.

Foi realizada a gravação de voz para um cartão de memória, com uma qualidade de 8 bits e uma frequência de amostragem de 16 kHz. Em seguida, fizeram-se várias cópias e escreveu-se um código de repetição sequencial para que a emissão não cessasse.

Por uma questão prática, foram realizados apenas testes em linha de vista. O local escolhido para replicar este teste foi a Ligação Pico do Facho - Miradouro Francisco Alves Nóbrega (1400 m), que já se verificou previamente

funcionar bem, com o sistema PTT (linha de vista garantida e espaço de Fresnel desimpedido).

A imagem que retrata o teste é a mesma da Figura 4.11. Tal como seria de esperar, os resultados foram exatamente os mesmos, tendo sido possível obter uma transmissão clara e limpa, sem erros dentro do alcance do sistema.

As mensagens que poderiam ser pedidos de ajuda, difusão de estados de emergência, avisos de meteorologia, entre outros, foram recebidas sem qualquer dificuldade.

4.5. Transmissão em direto - áudio 12 bits

Em determinadas aplicações, nomeadamente na monitorização de fenómenos ambientais, pode ser necessário a transmissão de áudio a frequências de amostragem mais elevadas. Por forma a testar as capacidades de processamento de áudio de alta qualidade do Arduíno DUE, foram realizados testes de aquisição e reprodução de áudio a 12 bits e frequência de amostragem de 44,1 kHz. Para esse fim foi elaborado um código de *pass-through* (que se encontra no Anexo J), ou seja, o áudio que se recolheu na porta analógica, era replicado na saída digital, em tempo real. A forma de onda obtida pode ser visualizada na Figura 4.26, onde a amarelo se apresenta o sinal a 20 kHz na entrada da porta analógica e roxo, o sinal lido no DAC.

Figura 4.26 - Análise do áudio a 20 kHz - à entrada (amarelo) e saída do Sistema (roxo).

Não se regista qualquer alteração ao sinal devido ao processamento digital do sinal.

A transmissão de áudio de alta qualidade é algo muito exigente em termos de processamento digital de sinal. As portas analógicas do Arduíno DUE e as suas respetivas funções de processamento são demasiado lentas para processamento e envio em tempo real (seja o DAC, seja para o barramento SPI). Por esse motivo foi necessário fazer uso direto dos registos do microcontrolador. O objetivo desta secção do trabalho foi a transmissão de áudio digital com 12 bits e 44,1 kHz, com um débito de 2 Mbps, recorrendo aos protótipos apresentados na Figura 3.27.

Para se compreender a influência do canal rádio na comunicação e a forma como este afeta o sinal foi injetado na porta analógica do emissor, um sinal sinusoidal a 20 kHz devidamente condicionado pelo pré-amplificador. No lado do recetor, foi colocada uma ponta de prova do osciloscópio, cuja janela de visualização se observa na Figura 4.27.

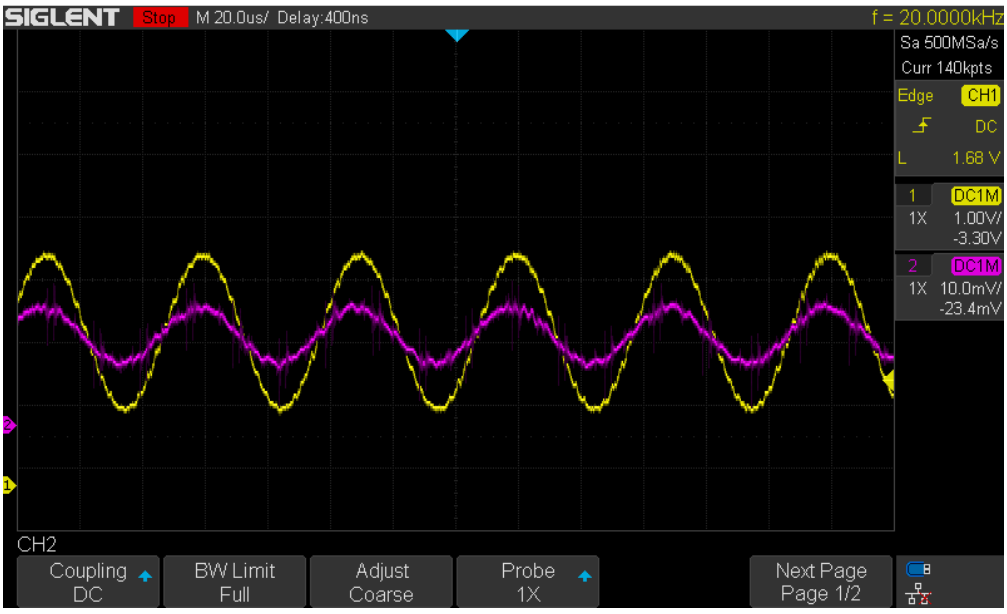

Figura 4.27 - Análise da influencia do rádio no áudio a 20 kHz - à entrada (amarelo) e saída do Sistema (roxo).

A amarelo, está representado o sinal de entrada, na entrada do emissor e a rosa o sinal na saída do DAC do recetor. Pode-se observar o efeito do canal rádio, onde se pode observar que o mesmo não introduz qualquer alteração no sinal.

4.6. Teste de gravação e transferência de áudio em diferido

Nesta secção apresentam-se os testes efetuados relativamente ao processo de gravação e transferência de mensagens gravadas entre Arduínos.

4.6.1. Teste de gravação de áudio para cartão de memória

Um teste teve como objetivo verificar o processo de gravação de áudio, para um cartão de memória, tendo em conta o tamanho que os ficheiros podem ter, mediante a qualidade do áudio gravado.

Relativamente ao tamanho das mensagens, sabendo que se trata de um sistema sem compressão, a dimensão do ficheiro é dada por

$$F_{size} = n \times f_s \times t \tag{4.2}$$

onde n é o número de bits por amostra, f_s é a taxa de amostragem em Hz e o t é a duração do tempo de gravação, em segundos.

Pela equação (4.2), determina-se que uma mensagem de voz com 8 bits, 8 kHz de amostragem e para um tempo de 25 s (tempo arbitrado para este processo), terá um tamanho aproximado de 200 kB.

4.6.2. Teste de envio de ficheiros de áudio

Um outro teste teve por objetivo avaliar o envio de mensagens contendo amostras de áudio. Os fatores a ter em conta neste procedimento são o tamanho das mensagens e o débito de transmissão.

O processo de transmissão de mensagens necessita de uma garantia de fiabilidade, por forma a assegurar que se perca o mínimo de informação possível. Foi ativado o processo de ACK, que vai fazer até 15 tentativas de entrega, com intervalos de espera de 15 µs entre cada uma delas.

O envio da mensagem consiste na leitura da mesma a partir de um cartão de memória, byte a byte, para um vetor de 32 bytes, onde ocupa as primeiras 31 posições do mesmo. A última posição foi utilizada como campo de controlo para identificação dos pacotes com o número 100 para pacotes de 31 bytes. Na última transmissão esse campo leva o número de bytes que compõem a trama final, indicando ao recetor o fim de ficheiro e fará o fecho do cartão e memória.

De modo a avaliar as velocidades de transmissão, foram estudados os débitos de 250 kbps e 2 Mbps, correspondendo ao débito mais baixo e mais elevado do rádio.

Foram definidos dois critérios de qualidade para a transmissão, um qualitativo que consiste na deteção de erros na reprodução dos ficheiros recebidos e um quantitativo que mede a quantidade de bytes perdidos (usando para isso o *plugin* compare do notepad++).

Recorrendo a um ficheiro contendo amostras de voz, colocou-se o mesmo no cartão de memória e procedeu-se ao seu envio para o terminal de destino, a cerca de 5 metros de distância. Este, por sua vez, guardou-o no respetivo cartão de memória. Em seguida foi colocado no reprodutor compatível, onde se pôde ouvir a mensagem.

Em seguida, colocaram-se ambos os ficheiros na mesma pasta e foram abertos no notepad++ e efetuada a sua comparação, ilustrada na Figura 4.28.

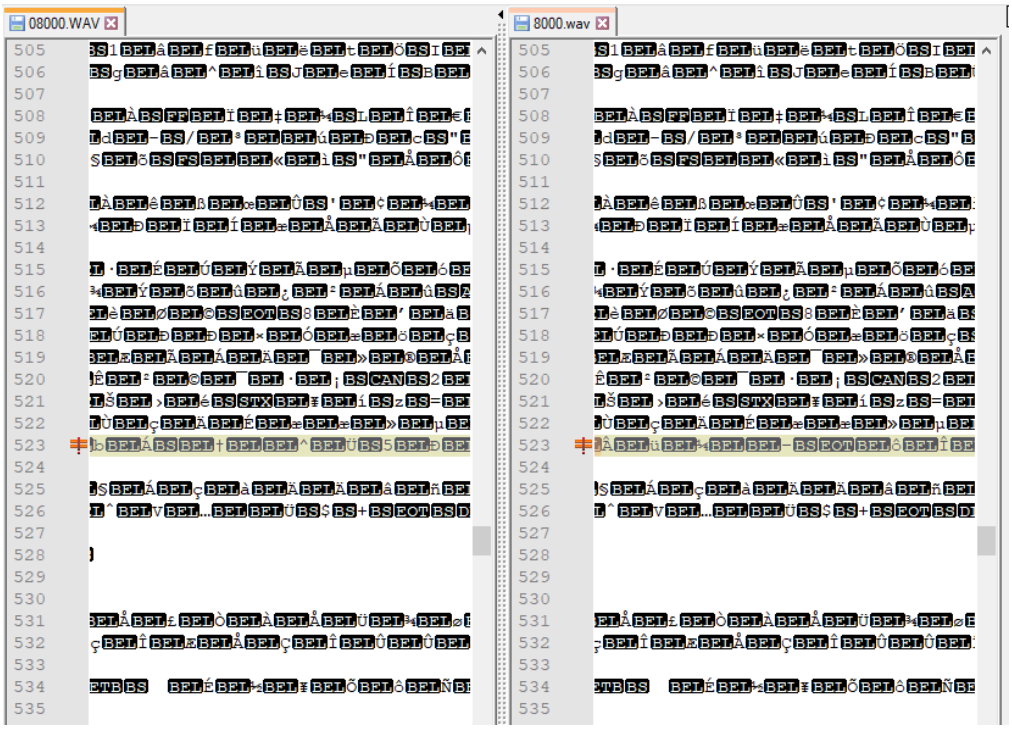

Figura 4.28 - Comparação byte a byte entre dois ficheiros, no notepad++

Apesar de não ter sido detectado qualquer erro na reprodução do ficheiro de áudio transmitido, existe uma diferenca de 31 bytes entre ambos. O algoritmo de CRC descarta um pacote inteiro se este estiver corrompido. A razão para este fenómeno se ter verificado pode estar relacionada com as colisões que se

verificam com pacotes provenientes de outros dispositivos na mesmas frequências.

Na transmissão dos dados é necessário dar um tempo para o sistema transmitir o pacote. Não se tendo presente qual o tempo a atribuir antes da transmissão do próximo pacote, começou-se por testar a transmissão sem atribuir um tempo de espera.

A taxa de perdas foi calculada da seguinte forma

$$T_{perdas} = \frac{n\ bytes\ errados}{tamanho\ ficheiro} \tag{4.3}$$

Dos testes, foi obtida uma taxa de perdas de 1,55%,

Após foi colocado um tempo de espera entre envios de 1 ms. A consequência inevitável é o abrandamento da taxa efectiva de transmissão.

Foi efetuado um teste na zona limite do alcance rádio, onde para isso se configurou o rádio emissor para o valor mínimo de potência, ou seja, -16,1 dBm, conforme a Tabela 4.1. Recorrendo à equação (2.1), determinou-se que a distância limite seria de 130 m, local onde se colocou o recetor.

Enviou-se o mesmo ficheiro contendo amostras de voz, com 25 s, 200kB de tamanho. Para um valor 200 kB enviados, perderam-se 6,375 KB de informação, o que corresponde a 3,18% de perdas. A mensagem continua perfeitamente audível, com pequenos estalidos que não impedem a compreensão da mesma.

O teste foi repetido 10 vezes e os resultados tiveram uma variação de 0 a 6% relativamente aos valores originais, ou seja, algumas vezes perderam-se menos pacotes (nalguns casos a transmissão foi perfeita, sem qualquer perda) e outras ainda mais.

Em seguida, repetiram-se os mesmos testes para o débito de 2 Mbps e o ficheiro foi transferido nos mesmos moldes. Os resultados foram similares, com o ajuste da distância de 130 para cerca de 30 m, para ficar dentro do alcance da sensibilidade (-82 dBm) para este débito.

A velocidade de transferência com os mecanismos de atraso adicionados é fortemente condicionada. Na Tabela 4.8, podem ser observados os valores comparativos entre o débito nominal e o real com um atraso de 1 ms.

Tabela 4.8 - Valores comparativos entre débito nominal e real com ACK e atraso 10ms.

	Débito nominal (kBps)	Débito real (kBps)
250 kbps	16,7	3,74
2 Mbps	68,03	15,85

Refira-se que consegue obter apenas cerca de 22% do débito nominal. É uma consequência do compromisso entre velocidade e transmissão de dados bem-sucedida. Um atraso menor tem como consequência uma menor capacidade de o sistema recuperar de falhas, nomeadamente de colisões que provocam perdas de pacotes.

4.7. Medição de consumos de corrente

Um fator a ter em consideração no momento da escolha dos sistemas a implementar é o consumo dos mesmos. Neste contexto, foram efetuadas medições de consumo de corrente recorrendo a medições diretas com amperímetros digitais. Os resultados estão registados na Tabela 4.9.

Tabela 4.9 - Consumos de corrente .

	Estado de Funcionamento (corrente)		
Protótipo	*Standby* (mA)	Transmitir (mA)	Receber (mA)
Arduíno UNO e nRF24L01	22	43	38
Rádio PTT (UNO)	80	154	110
Mensagem gravada	35	51	N/A
DUE Pass-through + amplificador	110	N/A	180
Arduíno DUE SD Player	145	N/A	200
Arduíno DUE PTT	150	160	410
Arduíno DUE Envio ficheiros	145	155	420

Dos resultados obtidos conclui-se que os sistemas apresentam um consumo significativo, que obriga a um planeamento no que diz respeito às formas de alimentação a implementar. O sistema que apresenta um consumo

mais favorável é o rádio PTT, por conter menos dispositivos de amplificação de som.

4.8. Conclusões ao capítulo

Neste capítulo foram descritas as várias etapas do processo de testes e de resultados. Foram efetuadas medições de potência, foram medidos consumos e testadas os limites das capacidades dos microcontroladores aplicados nos protótipos. Os consumos são relativamente elevados sobretudo para terminais alimentados a bateria. No entanto, podem ser ligados apenas no momento da utilização (exceto os terminais recetores que deverão estar permanentemente à escuta).

As transmissões de voz foram efetuadas com sucesso, em vários cenários, nomeadamente em direto, com baixa resolução em Arduíno UNO. Nesse contexto foi também testado e criado um sistema de *broadcast* de mensagens de voz gravadas, cujo conteúdo era reproduzido nos terminais desenvolvidos. Finalmente para áudio com qualidade superior, recorrendo a Arduíno DUE, em direto ou em diferido, este último gravado no momento, para ser enviado para armazenamento em terminal remoto. De referir que este último sistema pode transmitir ficheiros com qualquer taxa de amostragem, pois é independente desse parâmetro.

5. Conclusões e trabalhos futuros

Este capítulo descreve as conclusões obtidas ao longo do presente trabalho, bem como os futuros trabalhos propostos.

5.1. Conclusões

Foram desenvolvidos protótipos baseados em Arduíno de 8 bits, nomeadamente o UNO e o NANO. O áudio transmitido por estes protótipos foi de 8 bits a 16 kHz de frequência de amostragem. Estes protótipos revelaram funcionar de forma eficaz, ou seja, transmissão de voz clara e percetível, dentro das condições próximas do espaço livre e do alcance da sensibilidade do rádio. Os resultados sofrem variações em função do tipo de contexto de transmissão e diferem dos vários cenários onde se fizeram testes de funcionalidade. Em linha de vista, com as condições ideais transmite-se áudio a vários milhares de metros. No entanto, no limiar da sensibilidade, o áudio perde qualidade, passando a garantir apenas os mínimos para se poder ouvir a mensagem com alguns erros e estalidos. Foi ainda neste contexto de transmissão de voz, testado o envio de mensagens gravadas em Cartão SD, usando os rádios dos testes anteriores para receber e ouvir as mesmas. O envio foi realizado a partir de protótipos contruídos para o efeito. Por uma questão prática testou-se apenas um cenário em linha de vista, onde o desempenho foi positivo, com áudio claro e sem qualquer interferência.

Com o objetivo de explorar as capacidades do rádio e de melhorar a qualidade do áudio transmitido, passou-se a usar o Arduíno DUE. O primeiro teste foi sem rádio, apenas recolhendo o áudio na porta analógica do microcontrolador, fazendo a sua reprodução no DAC em tempo real. Foi então que se utilizaram protótipos para transmissão de áudio de 12 bits a 44,1 kHz recorrendo para isso à manipulação dos registos devido ao desempenho mais lento das funções básicas de leitura das portas analógicas do Arduíno DUE. Testaram-se as transmissões, em distâncias mais reduzidas, na medida em que a 2 Mbps, a sensibilidade dos rádios cai significativamente. Depois, foram realizados testes com cartão SD para aferir -o funcionamento dos registos e das velocidades de escrita e leitura das várias bibliotecas de gestão deste dispositivo. Na reprodução de áudio com estas características, nota-se um som limpo,

agradável ao ouvido sem interferências. O áudio recebido, quando dentro do alcance rádio ideal, não apresenta erros percetíveis, não oferecendo qualquer dificuldade à sua compreensão. No limiar da sensibilidade do rádio, a perda de pacotes condiciona muito o sinal. Neste caso, como a frequência de amostragem é muito elevada, o sistema minimiza alguns erros (pacotes perdidos não afetam significativamente o áudio). Depois foram realizados testes de gravação de áudio a partir de microfone com pré-amplificador, com a qualidade referida. O objetivo foi testar o desempenho do sistema em termos de qualidade e velocidade para preparar o objetivo seguinte.

O passo seguinte teve como objetivo criar condições para a integração da gravação, mediante o pressionar de um botão, de um pedido de ajuda, ou socorro, para envio posterior. Para isso, recorreu-se aos protótipos da transmissão de áudio em direto, com o Arduíno DUE, acrescentando apenas a interface do cartão SD. Foram testados vários cenários de transmissão de ficheiros contendo voz, com o mecanismo de ACK ativado, a 250 kbps e 2 Mbps em cenários de transmissão com e sem obstáculos. Para a transmissão a 250 kbps, o sistema funciona bem, desde que dentro do alcance rádio, com o mecanismo de ACK. No teste de transmissão e uma mensagem de voz de cerca de 200 kB (cerca de 30 segundos de duração), o tempo de envio da mensagem foi de cerca 4 minutos. Por outro lado, o mesmo ficheiro transmitido a 2 Mbps, demora apenas 70 segundos a ser transmitido, mantendo os mesmos mecanismos de ACK e respetivos atrasos, com a desvantagem da distância máxima ser inferior devido à redução do limite de sensibilidade do recetor.

5.2. Trabalhos futuros

No que diz respeito a trabalhos futuros, sugere-se que seja explorado o aumento do número de rádios, criando uma estrutura sólida de comunicação, convergindo para uma rede de implementação prática.

Por outro lado, a criação de repetidores para protótipos de comunicação baseados no rádio estudado, uma vez que só se pode considerar seriamente o uso de modelos que permitam a repetição do sinal, para alcançar distâncias mais robustas.

No que diz respeito à questão da interação humano com a máquina, fica ainda a faltar a criação de interfaces de chamada devidamente orientadas para o utilizador.

Outra vertente que pode ser melhorada é a integração do sistema de gravação mensagens através da porta analógica, para o cartão SD com o protótipo de transferência de ficheiros de voz, bem como o processo inverso, ou seja, a obtenção da mensagem de voz a partir do canal rádio para o cartão SD, por forma a guardar ou reproduzir o mesmo.

A inclusão de um sistema de georreferenciação (coordenadas GPS), por forma a identificar automaticamente o local de onde é feita gravação de áudio, tanto para pedidos de ajuda, como para áudio ambiental.

Para este tipo de testes, poderia ser testada uma variante do rádio usado, que é o nRF24LE1, que dispõe de microcontrolador integrado, pelo que dispensa o uso de Arduíno.

6. Referências

[1] B. Chen, "Audio Recognition with Distributed Wireless Sensor Networks, masters thesis," B.Sc. University of Victoria, 2004.

[2] J. L. Gutierres, "MICA2 wireless sensor device," *Download Scientific Diagram ResearchGate*, 2006. [Online]. Available: https://www.researchgate.net/profile/Jose_Lanza-Gutierrez/publication/262154514/figure/fig2/AS:345733032103942@1459440586804/MICA2-wireless-sensor-device.png. [Accessed: 10-May-2019].

[3] S. Ivan Lopes and F. Jorge Ventura, "Intercomunicador digital de imagem e som sem fios, Projeto de Licenciatura," Universidade de Aveiro, 2000.

[4] RadioMetrix, "The BiM2 transceiver," *433MHz high speed FM radio transceiver module*, 2004. [Online]. Available: http://www.radiometrix.com/files/additional/bim2.pdf. [Accessed: 10-May-2019].

[5] ANACOM, "Quadro Nacional de Atribuição de Frequências," *ADENDA 2013*, 2013. [Online]. Available: https://www.anacom.pt/streaming/Adenda_2013_QNAF.pdf?contentId=1172857&field=ATTACHED_FILE. [Accessed: 10-May-2019].

[6] H. Chen, H. Jin, L. Guo, S. Wu, and T. Gu, "Audio-on-demand over wireless sensor networks," in *IEEE International Workshop on Quality of Service, IWQoS*, 2012.

[7] C. E. R. Lopes and L. César e Melo, "Transmissão de áudio e video em Redes de Sensores sem Fio," *REDES DE SENSORES SEM FIO*, 2005. .

[8] crossbow technology, "WIRELESS MEASUREMENT SYSTEM," *MICAz*, 2006. [Online]. Available: http://www.cmt-gmbh.de/MICAz.pdf. [Accessed: 11-May-2019].

[9] F. Reinoso, C. Ku Leuven, and A. Touhafi, "Wireless Sensor Networks for sound design: A summary on possibilities and challenges," *ResearchGate*, 2016.

[10] L. Li, G. Xing, L. Sun, and Y. Liu, "A Quality-Aware Voice Streaming System for Wireless Sensor Networks," *ACM Trans. Sens. Networks*, vol. 10, no. 4, pp. 1–25, Jul. 2014.

[11] Texas Instruments, "Low-Power RF Transceiver," *CC1100 Low-Power Sub-1 GHz RF Transceiver*, 2006. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc1100.pdf. [Accessed: 10-May-2019].

[12] Nordic, "nRF24L01 Nordic Datasheet," 2015. [Online]. Available: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf. [Accessed: 11-May-2019].

[13] H. Saha, S. Mandal, S. Mitra, S. Banerjee, and U. Saha, "Comparative Performance Analysis between nRF24L01+ and XBEE ZB Module Based Wireless Ad-hoc Networks," *Int. J. Comput. Netw. Inf. Secur.*, vol. 9, no. 7, pp. 36–44, 2017.

[14] A. Gour and P. Jain, "Design of Wireless Audio Exchange Based on nRF module," *Int. Res. J. Eng. Technol.*, 2015.

[15] Lillyelectronics, "XL-01 M digital audio module," *Datasheet XL-01 M*, 2018. [Online]. Available: http://www.lillyelectronics.com/2-4g-nrf24l01-wireless-

digital-audio-transceiver-module. [Accessed: 08-Aug-2019].

[16] M. Caldeira, "DIY wireless guitar system using NRF24L01," 2017. [Online]. Available: https://www.youtube.com/watch?v=66lN1GD_TN4. [Accessed: 25-May-2019].

[17] D. Brunelli, M. Maggiorotti, L. Benini, and F. L. Bellifemine, "Analysis of audio streaming capability of Zigbee networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4913 LNCS, pp. 189–204, 2008.

[18] T. Instruments, "A True System-on-Chip solution for 2.4 GHz IEEE 802.15.4/ZigBee(TM) (Rev. F)." 2019.

[19] M. Gysel, D. El, I. Fh, and M. Sommerhalder, "Using IEEE 802.15.4 / ZigBee in audio applications," 2006.

[20] N. Bagal and P. S. Pandita, "Transmission of Audio Over Wireless Networks Using Raspberry Pi in Real-Time," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 4, no. 9, pp. 9435–9440, 2015.

[21] R. Foundation, "Raspberry Pi," *Setting up your Raspberry Pi*, 2018. [Online]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up. [Accessed: 15-Dec-2019].

[22] Arduino.cc, "SPI Protocol," *Arduino - SPI*, 2000. [Online]. Available: https://www.arduino.cc/en/Reference/SPI. [Accessed: 27-Oct-2019].

[23] Arduino.cc, "Arduino Uno Overview," *Arduino Uno Board*, 2000. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 05-Dec-2019].

[24] arduinomais.blogspot.com, "Arduino UNO," *Projectos Arduino UNO*, 2014. [Online]. Available: http://arduinomais.blogspot.com/2014/06/detalhes-sobre-o-arduino-uno.html. [Accessed: 01-Jun-2019].

[25] Arduino.cc, "Arduino Nano," *Description and datasheet*, 2009. [Online]. Available: https://store.arduino.cc/arduino-nano. [Accessed: 19-May-2019].

[26] J. C. Quer, "Arduino Nano," *User manual*, 2016. [Online]. Available: https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf. [Accessed: 11-May-2019].

[27] Atmel, "Microcontroller-SAM3X-SAM3A_Datasheet.pdf," 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf. [Accessed: 19-May-2019].

[28] Arduino.cc, "Arduino Due," *Arduino DUE - Datasheet*, 2010. [Online]. Available: http://www.fecegypt.com/uploads/dataSheet/1522331020_arduino due.pdf. [Accessed: 11-May-2019].

[29] Arduino.cc, "Arduino Due," *Arduino Due Overview*, 2014. [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoDue. [Accessed: 01-Jun-2019].

[30] TMRh20, "Optimized High Speed Driver for nRF24L01(+) 2.4GHz Wireless Transceiver," *Documentation Class RF24*, 2014. [Online]. Available: https://tmrh20.github.io/RF24/. [Accessed: 05-Aug-2019].

[31] S. K. Shesma, "Wireless transmission of voice signal using nRF24L01 module," *Communication System Project(Electronics Engineering)*, 2010. [Online]. Available: https://www.slideshare.net/SunilKumarShesma/communication-system-projectelectronics-engineering. [Accessed: 08-Aug-2019].

[32] Lastminuteengineers.com, "nRF24L01 Wireless Transceiver Module," 2015. [Online]. Available: https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/. [Accessed: 10-May-2019].

[33] N. C. Braga, "Curso de Eletrônica Fundamentos de Som e Acústica," *Fundamentos de Som e Acústica*, 2019. [Online]. Available: http://www.newtoncbraga.com.br/index.php/almanaque-tecnologico. [Accessed: 02-Jan-2020].

[34] Sparkfun, "Electret Microphone," *Electret Microphone Datasheet*, 2015. [Online]. Available: https://cdn.sparkfun.com//assets/parts/1/7/6/9/08635-03-L.jpg. [Accessed: 01-Jun-2019].

[35] L. A. Electronics, "Microphone Amplifier Circuit," *Build a Microphone Amplifier Circuit*, 2011. [Online]. Available: http://www.learningaboutelectronics.com/Articles/Microphone-amplifier-circuit.php. [Accessed: 19-May-2019].

[36] E. Sparkfun, "Speaker," *Speaker 0,5W (8 Ohm)*, 2015. [Online]. Available: https://www.sparkfun.com/products/9151. [Accessed: 01-Jun-2019].

[37] Eleccircuit.com, "Audio Amplifier circuit stereo," *Amplifier TBA 820*, 2000. [Online]. Available: https://www.eleccircuit.com/wp-content/uploads/2009/08/tba820-mini-audio-amplifier-12w.jpg. [Accessed: 04-Jun-2019].

[38] A. Thomsen, "Gravar dados no cartão SD com Arduino," *FilipeFlop*, 2014. [Online]. Available: https://uploads.filipeflop.com/2015/07/Modulo_Cartao_SD.jpg. [Accessed: 01-Jun-2019].

[39] L. M. Engineers.com, "SD Card Module with Arduinocro," *SD Card Module with Arduino*, 2015. [Online]. Available: https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/. [Accessed: 26-Jul-2019].

[40] Tinker Labz, "Arduino Nano and nRF24L01," *Connect nRF24L01 to Arduino*, 2016. [Online]. Available: http://tinkerlabz.com/wp-content/uploads/2015/07/Arduino_Nano_NRF24L01_Wiring_Cap.jpg. [Accessed: 11-May-2019].

[41] Instructables, "Call button for Arduino," 2012. [Online]. Available: https://cdn.instructables.com/FEN/587S/IHCBQW28/FEN587SIHCBQW28.LARGE.jpg. [Accessed: 05-Jun-2019].

[42] Nordic Semiconductor, "nRF24L01+ Datasheet," *Nord. Semicond.*, no. March, p. 75, 2008.

[43] C. Salema, *Feixes Hertzianos*, 3ª Edição. Lisboa: PRESS, Instituto Superior Técnico, 2011.

[44] G. Corporation, "Google Earth PRO," *Elevation Profile*, 2020. [Online]. Available: https://earth.google.com/web/. [Accessed: 03-Jan-2020].

[45] J. A. Azevedo, F. E. Santos, T. A. Sousa, and J. M. Agrela, “Impact of the antenna directivity on path loss for different propagation environments,” *IET Microwaves, Antennas Propag.*, vol. 9, no. 13, pp. 1392–1398, 2015.

ANEXOS

Anexo A - Código Emissor Recetor com PTT

```
#include <RF24.h>
#include <SPI.h>
#include <RF24Audio.h>
#include "printf.h"
#include <SD.h>

RF24 radio(7, 8);   // Pinos para activar SPI do sistema 7 (CE) 8 (CS)
RF24Audio rfAudio(radio, 0); // Chama a funcao rFAudio e activa o radio
para o grupo '0'

int botaodefalar = 3;
int falar = 0;

void setup() {
  Serial.begin(115200); //abre a ligação Serie
pinMode(4, OUTPUT); // LED indicativo de transmissão em curso.
  digitalWrite(4, HIGH);
  printf_begin();       // activa a função printf para imprimir detalhes
na porta série
  radio.begin();        // inicializa o Rádio a partir da biblioteca
global
    Serial.println("RADIO OK ");
  rfAudio.begin();      // inicializa as funções de audio
    Serial.println("RAUDIO OK");
//  radio.setPALevel(RF24_PA_MIN);
  radio.setPALevel(RF24_PA_LOW);

  radio.setDataRate(RF24_250KBPS);
//   radio.setDataRate(RF24_1MBPS);
  //radio.setChannel(0x4c);  //  Possibilidade de escolher qualquer Canal
em HEX  76
  radio.printDetails();   // Imprime os detalhes sobre funconamento do
rádio, canais, endereços, etc..
  pinMode(botaodefalar, INPUT);  // Botão para falar como INPUT
  pinMode(5, OUTPUT); // LED indicativo de transmissão em curso.
  digitalWrite(5, LOW); // inicializa o LED como desligado.
  //Ajusta o pino 3 como verificação de botão de falar pressionado
  attachInterrupt(digitalPinToInterrupt(botaodefalar), talk, CHANGE);
  Serial.println("setVolume(3)");
  rfAudio.setVolume(2); // ajusta o volume no máximo
  Serial.print("O canal escolhido foi ");
  Serial.println(radio.getChannel());
  Serial.println(radio.testRPD());
  rfAudio.transmit(); // Transmite por 10 ms para activar o sistema de
recepção
  delay(100);
  // o sistema passa a receber, por defeito.
  rfAudio.receive();
}

/*
  void talk()
  Chamado em resposta à interrupção. Verifica o estado do botão.
  Se o botão for pressionado (e mantido) entra no modo de transmissão para
enviar
  áudio. Se o botão for solto, entra no modo de recepção para ouvir.
*/
void talk()
{
```

```
  if (digitalRead(botaodefalar))
  {
    rfAudio.transmit();
    Serial.println("Em transmissão");
    digitalWrite(5, HIGH);
  }
  else
  {

    rfAudio.receive();
    digitalWrite(5, LOW);
    Serial.println("A ouvir");
  }
}

void loop()
{

  //Loop sem nada para fazer

}
```

Anexo B - Código para difusão de mensagens áudio a partir de Cartão SD

```
#include <SD.h>                 // need to include the SD library
#define SD_ChipSelectPin 4
#include <TMRpcm.h>             //
#include <SPI.h>
#include <RF24.h>
#include "printf.h"

RF24 radio(7,8);                    // I use 7,8 on Uno, Nano, etc
pcmRF rfAudio(radio);

TMRpcm audio;

void setup(){

  audio.speakerPin = 9; // Use pin 9 on 328 boards (Uno, etc)
  pinMode(10,OUTPUT);    // Pin 10 on 328 boards (Uno, etc)

  Serial.begin(115200);

  if (!SD.begin(SD_ChipSelectPin)) { // see if the card is present
    Serial.println("SD fail");
    return;                           // don't do anything more if not
  }else{
    Serial.println("SD ok");
  }

  printf_begin();                   // Comment this out to save space
  rfAudio.begin();
  radio.setDataRate(RF24_250KBPS);
  radio.printDetails();             // Comment this out to save space
}

void loop(){

  if(Serial.available()){
    switch(Serial.read()){
    case 'a': audio.stopPlayback(); rfAudio.play("camilo.wav",1);
break;   // Play to radio # 1 in the radio group
    case 'b': audio.stopPlayback(); rfAudio.play("camilo.wav",2);   break;
// Play to radio # 0 in the radio group
    case 'c': audio.stopPlayback(); rfAudio.play("camilo.wav",255); break;
// Play to all radios in the radio group
    case '1': rfAudio.broadcast(1);
break;   // Broadcast the current audio to a different radio/group
    case '2': rfAudio.broadcast(2);
break;   // Broadcast the current audio to a different radio/group
    case '3': rfAudio.broadcast(255);
break;   // Broadcast the current audio to a different radio/group
    case 'd': rfAudio.stop(); audio.play("camilo.wav");
break;   // Play a file locally
    case 'e': rfAudio.stop(); audio.play("camilo.wav");             break;
// Play a file locally
    case 's': rfAudio.stop();
break;   // Stop radio playback
    case 'S': audio.stopPlayback();
break;   // Stop local playback
    case '=': audio.volume(1);
break;   // Volume up
```

```
    case '-': audio.volume(0);
break;   // Volume down
    default: break;
    }
  }
}
```

Anexo C - Código Arduíno DUE Reprodução Áudio de Cartão SD.

```
#include <SD.h>
#include <SPI.h>
#include <Audio.h>

void setup() {
  // debug output at 9600 baud
  Serial.begin(9600);

  // setup SD-card
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
    Serial.println(" failed!");
    while(true);
  }
  Serial.println(" done.");
  // hi-speed SPI transfers

  // 44100kHz stereo => 88200 sample rate
  // 100 mSec of prebuffering.
  Audio.begin(88200, 100);
}

void loop() {
  int count = 0;
  File myFile = SD.open("test.wav");  // open wave file from sdcard
  if (!myFile) {
    // if the file didn't open, print an error and stop
    Serial.println("error opening test.wav");
    while (true);
  }

  const int S = 1024; // Number of samples to read in block
  short buffer[S];

  Serial.print("Playing");
  // until the file is not finished
  while (myFile.available()) {
    // read from the file into buffer
    myFile.read(buffer, sizeof(buffer));

    int volume = 1024;   // Prepare samples

    Audio.prepare(buffer, S, volume);
    // Feed samples to audio
    Audio.write(buffer, S);

    // Every 100 block print a '.'
    count++;
    if (count == 100) {
      Serial.print(".");
      count = 0;
    }
  }
  myFile.close();
  Serial.println("End of file. Thank you for listening!");
  while (true);
}
```

Anexo D - Código do Emissor Arduíno DUE nRF24L01

```
#include <SPI.h>
#include "RF24.h"
bool radioNumber = 1;
RF24 radio(7, 8);
const int Freq = 44100;
byte buf00[32] = {0};
byte buf01[32] = {0};
int bufByteCount = 0;
int bufWrite = 1;
int readBuf00 = 0;
int readBuf01 = 1;
int recPressed = 0;
byte addresses[][6] = {"1Node", "2Node"};
bool role = 0;

void setup()
{ pinMode(13, OUTPUT);
  Serial.begin(115200);
  Serial.println(F("R E C E T O R"));
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setPayloadSize(32);
  radio.setDataRate(RF24_2MBPS);
  radio.setAutoAck(0);
  radio.startListening();
  if (radioNumber)
  {
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1, addresses[0]);
  }
  else
  {
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1, addresses[1]);
  }
  //  radio.startListening();
  dac_setup ();

  pmc_enable_periph_clk (TC_INTERFACE_ID + 0 * 3 + 0) ;
  TcChannel * t = &(TC0->TC_CHANNEL)[0] ;
  t->TC_CCR = TC_CCR_CLKDIS ;
  t->TC_IDR = 0xFFFFFFFF ;
  t->TC_SR ;
  t->TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK2 | TC_CMR_WAVE |
TC_CMR_WAVSEL_UP_RC;
  int cycle = 42000000 / Freq;

  t->TC_RC = cycle;
  t->TC_RA = cycle / 2;
  t->TC_CCR = TC_CCR_CLKEN | TC_CCR_SWTRG ;
  TC_Start(TC0, 0);
  t->TC_IER = TC_IER_CPCS;
  t->TC_IDR = ~TC_IER_CPCS; NVIC_EnableIRQ(TC0_IRQn);
  digitalWrite(13, LOW);
  radio.printDetails();
}
void TC0_Handler ()
{ TC_GetStatus(TC0, 0);
  if (bufByteCount == 32 && bufWrite == 0) {
```

```
    bufByteCount = 0;
    bufWrite = 1;
    readBuf00 = 1;
  }
  else if (bufByteCount == 32 & bufWrite == 1) {
    bufByteCount = 0;
    bufWrite = 0;
    readBuf01 = 1;
  }
  if (bufWrite == 0) {
    DACC->DACC_CDR = (((buf00[bufByteCount] & 0xFF) << 8) +
(buf00[bufByteCount + 1] & 0xFF));
    buf00[bufByteCount] = 0;
    buf00[bufByteCount + 1] = 0;
  }
  if (bufWrite == 1) {
    DACC->DACC_CDR = (((buf01[bufByteCount] & 0xFF) << 8) +
(buf01[bufByteCount + 1] & 0xFF));
    buf01[bufByteCount] = 0;
    buf01[bufByteCount + 1] = 0;
  }
  bufByteCount = bufByteCount + 2;
}
void loop()
{
  if (recPressed == 0) {
    recPressed = 1; bufByteCount = 0;
    digitalWrite(13, LOW);
  }
  if (readBuf00 == 1 && recPressed == 1)
  {
    if (radio.available()) {
      Serial.println(F("TIC TAC"));
      digitalWrite(13, HIGH);
      readBuf00 = 0;
      while (radio.available()) {
        radio.read( &buf00, 32);
        break;
      }
    }
  }
  if (readBuf01 == 1 && recPressed == 1)
  {
    if (radio.available()) {
      readBuf01 = 0;
      while (radio.available()) {
        radio.read( &buf01, 32);
        break;
      }
    }
  }
}
void dac_setup () {
  pmc_enable_periph_clk (DACC_INTERFACE_ID);
  DACC->DACC_CR = DACC_CR_SWRST;
  DACC->DACC_MR = DACC_MR_USER_SEL_CHANNEL0, (3 << DACC_MR_STARTUP_Pos) |
DACC_MR_TRGEN_EN | DACC_MR_TRGSEL (1);
  DACC->DACC_IDR = 0xFFFFFFFF;
  DACC->DACC_CHER = DACC_CHER_CH0 << 0;
}
```

Anexo E - Código do Receptor Arduíno DUE nRF24L01

```
#include <SPI.h>
#include "RF24.h"
bool radioNumber = 1;
RF24 radio(7, 8);
const int Freq = 44100;
byte buf00[32] = {0};
byte buf01[32] = {0};
int bufByteCount = 0;
int bufWrite = 1;
int readBuf00 = 0;
int readBuf01 = 1;
int recPressed = 0;
byte addresses[][6] = {"1Node", "2Node"};
bool role = 0;

void setup()
{ pinMode(13, OUTPUT);
  Serial.begin(115200);
  Serial.println(F("R E C E T O R"));
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setPayloadSize(32);
  radio.setDataRate(RF24_1MBPS);
  radio.setAutoAck(0);
  radio.startListening();
  if (radioNumber)
  {
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1, addresses[0]);
  }
  else
  {
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1, addresses[1]);
  }
  //  radio.startListening();
  dac_setup ();

  pmc_enable_periph_clk (TC_INTERFACE_ID + 0 * 3 + 0) ;
  TcChannel * t = &(TC0->TC_CHANNEL)[0] ;
  t->TC_CCR = TC_CCR_CLKDIS ;
  t->TC_IDR = 0xFFFFFFFF ;
  t->TC_SR ;
  t->TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK2 | TC_CMR_WAVE |
TC_CMR_WAVSEL_UP_RC;
  int cycle = 42000000 / Freq;

  t->TC_RC = cycle;
  t->TC_RA = cycle / 2;
  t->TC_CCR = TC_CCR_CLKEN | TC_CCR_SWTRG ;
  TC_Start(TC0, 0);
  t->TC_IER = TC_IER_CPCS;
  t->TC_IDR = ~TC_IER_CPCS; NVIC_EnableIRQ(TC0_IRQn);
  digitalWrite(13, LOW);
  radio.printDetails();
}
void TC0_Handler ()
{ TC_GetStatus(TC0, 0);
  if (bufByteCount == 32 && bufWrite == 0) {
```

```
    bufByteCount = 0;
    bufWrite = 1;
    readBuf00 = 1;
  }
  else if (bufByteCount == 32 & bufWrite == 1) {
    bufByteCount = 0;
    bufWrite = 0;
    readBuf01 = 1;
  }
  if (bufWrite == 0) {
    DACC->DACC_CDR = (((buf00[bufByteCount] & 0xFF) << 8) +
(buf00[bufByteCount + 1] & 0xFF));
    buf00[bufByteCount] = 0;
    buf00[bufByteCount + 1] = 0;
  }
  if (bufWrite == 1) {
    DACC->DACC_CDR = (((buf01[bufByteCount] & 0xFF) << 8) +
(buf01[bufByteCount + 1] & 0xFF));
    buf01[bufByteCount] = 0;
    buf01[bufByteCount + 1] = 0;
  }
  bufByteCount = bufByteCount + 2;
}
void loop()
{
  if (recPressed == 0) {
    recPressed = 1; bufByteCount = 0;
    digitalWrite(13, LOW);
  }
  if (readBuf00 == 1 && recPressed == 1)
  {
    if (radio.available()) {
      Serial.println(F("TIC TAC"));
      digitalWrite(13, HIGH);
      readBuf00 = 0;
      while (radio.available()) {
        radio.read( &buf00, 32);
        break;
      }
    }
  }
  if (readBuf01 == 1 && recPressed == 1)
  {
    if (radio.available()) {
      readBuf01 = 0;
      while (radio.available()) {
        radio.read( &buf01, 32);
        break;
      }
    }
  }
}
void dac_setup () {
  pmc_enable_periph_clk (DACC_INTERFACE_ID);
  DACC->DACC_CR = DACC_CR_SWRST;
  DACC->DACC_MR = DACC_MR_USER_SEL_CHANNEL0, (3 << DACC_MR_STARTUP_Pos) |
DACC_MR_TRGEN_EN | DACC_MR_TRGSEL (1);
  DACC->DACC_IDR = 0xFFFFFFFF;
  DACC->DACC_CHER = DACC_CHER_CH0 << 0;
}
```

Anexo F - Código para gravação de Áudio para cartão SD

```
#include <SdFat.h>

SdFat sd;
SdFile rec;

const int chipSelect = 4;
//const int Freq = 44100;  // Sampling frequency
const int Freq = 8000;  // Sampling frequency
byte buf00[1024]; // buffer array 1
byte buf01[1024]; // buffer array 2

int bufByteCount = 0;
int bufWrite = 0;
int writeBuf00 = 0;
int writeBuf01 = 0;

const int ledStart = 2;
const int ledStop = 3;
const int btnStart = 5;
const int btnStop = 6;
int recPressed = 0;
int stopPressed = 0;


short SigIn = 0;


void setup()
{
  delay(2000);
  Serial.begin (115200) ;

  pinMode(ledStart, OUTPUT);
  pinMode(ledStop, OUTPUT);
  pinMode(btnStart, INPUT_PULLUP);
  pinMode(btnStop, INPUT_PULLUP);

  Serial.print("Initializing SD card...");
  if (sd.begin(chipSelect, SPI_FULL_SPEED)) {
    Serial.println("initialization done.");
  }
  else {
    Serial.println("initialization failed!");
    return;
  }

  adc_setup ();         // setup ADC

  pmc_enable_periph_clk (TC_INTERFACE_ID + 0 * 3 + 0) ; // clock the TC0
channel 0

  TcChannel * t = &(TC0->TC_CHANNEL)[0] ;    // pointer to TC0 registers
for its channel 0
  t->TC_CCR = TC_CCR_CLKDIS ;  // disable internal clocking while setup
regs
  t->TC_IDR = 0xFFFFFFFF ;     // disable interrupts
  t->TC_SR ;                   // read int status reg to clear pending
```

```
  /*Setup clock speed
    TCCLKS selects the timer clock based on a scaler of the 84 MHz MCK
    TIMER_CLOCK1 = MCK/2
    TIMER_CLOCK2 = MCK/8
    TIMER_CLOCK3 = MCK/32
    TIMER_CLOCK4 = MCK/128
    TIMER_CLOCK5(1) = SLCK
  */

  t->TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1 |    // use TCLK1 (prescale by 2,
= 42MHz)
              TC_CMR_WAVE |                   // waveform mode
              TC_CMR_WAVSEL_UP_RC;            // count-up PWM using RC as
threshold

  /*Create a slower clock based on the desired sample rate*/
  int cycle = 42000000 / Freq; // 42 MHz/desired frequency, yields the
counter used by registers
  t->TC_RC =  cycle;         // RegisterC counts at the specificed cycle
speed (ticks high each cycle)
  t->TC_RA =  cycle / 2;     // RegisterA counts twice as fast as RC,
  t->TC_CMR = (t->TC_CMR & 0xFFF0FFFF) |
              TC_CMR_ACPC_SET |             // RC set compare on TIOA,
creating a new clock from RC and RA cycle times
              TC_CMR_ACPA_CLEAR ;           // RA clear on TIOA

  t->TC_CCR = TC_CCR_CLKEN | TC_CCR_SWTRG ;  // re-enable local clocking
and switch to hardware trigger source.


}

void ADC_Handler (void)
{
  if (ADC->ADC_ISR & ADC_ISR_EOC7)
  { // initialize when ADC end-of-conversion is flagged (NOTE: pin A0 is
AD7, so the EOC flag is on EOC7)
    SigIn = *(ADC->ADC_CDR + 7) ; // get conversion result from the ADC
CRD. (NOTE: pin A0 is AD7, so conversion data is held in CRD7)

    if (bufByteCount == 1024 && bufWrite == 0) {
      bufByteCount = 0;
      bufWrite = 1;
      writeBuf00 = 1;
    } else if (bufByteCount == 1024 & bufWrite == 1) {
      bufByteCount = 0;
      bufWrite = 0;
      writeBuf01 = 1;
    }

    if (bufWrite == 0)
    {
      buf00 [bufByteCount] = SigIn >> 8 & 0xFF;
      bufByteCount++;
      buf00 [bufByteCount] = SigIn & 0xFF;
    }
    if (bufWrite == 1)
    {
      buf01 [bufByteCount] = SigIn >> 8 & 0xFF;
      bufByteCount++;
      buf01 [bufByteCount] = SigIn & 0xFF;
```

```
    }
    bufByteCount++;

  }
}

void loop()
{
  // put your main code here, to run repeatedly:
  // this will be interrupted when each new sample is called

  if (digitalRead(btnStart) == LOW && recPressed == 0)
  {
    digitalWrite(ledStart, HIGH);
    digitalWrite(ledStop, LOW);
    recPressed = 1;
    stopPressed = 0;
    rec.open("8000.wav", O_CREAT | O_TRUNC | O_RDWR);
  }

  if (digitalRead(btnStop) == LOW)
  {
    rec.close();
    digitalWrite(ledStart, LOW);
    digitalWrite(ledStop, HIGH);
    recPressed = 0;
  }

  if (writeBuf00 == 1 && recPressed == 1)
  {
    writeBuf00 = 0;
    rec.write ( buf00, 1024);
  }

  if (writeBuf01 == 1 && recPressed == 1)
  {
    writeBuf01 = 0;
    rec.write ( buf01, 1024);
  }

}

void adc_setup ()
{
  /* SETUP ADC to run with TI0A Ch0 as the trigger, with 12 bit accuracy
    Only A0 is activated by this setup, with end of conversion interrupt
too
    All other analog inputs disabled, will need reactivating in setup.
  */
  adc_disable_all_channel(ADC); // disable all ADC channels (less noise)
  eAnalogReference(DEFAULT);    // set ADC reference voltage to the 3.3 V
default on the Due
  NVIC_EnableIRQ (ADC_IRQn) ;   // enable ADC interrupt vector
  ADC->ADC_IDR = 0xFFFFFFFF ;   // disable interrupts
  ADC->ADC_IER = 0x80 ;         // enable AD7 End-Of-Conv interrupt
(Arduino pin A0), used to check for completed conversions
  ADC->ADC_CHER = 0x80 ;        // enable just A0

  /*CHANNEL ASSIGNMENTS ARE DIFFERENT TO WHAT IS MARKED ON THE DUE
    A0 = AD7, A1 = AD6, A2 = AD5, A3 = AD4, A4 = AD3, A5 = AD2,
    A6 = AD1, A7 = AD0, A8 = AD10, A9 = AD11, A10 = AD12, A11 = AD13.*/
```

```
  ADC->ADC_CGR = 0 ;   // All gains set to 1
  ADC->ADC_COR = 0x00000000 ;   // All offsets off

  ADC->ADC_MR = ADC_MR_ANACH |          // allow for different analog
settings on each channel (if desired)
                ADC_MR_LOWRES_BITS_12 |   // enforce 12 bit capture
                ADC_MR_FREERUN_OFF |      // turn off freerun to force
wait for trigger
                ADC_MR_TRGEN |            // enable external trigger mode
for ADC
                ADC_MR_TRGSEL_ADC_TRIG1 ; /* ADC_TRIG1 = Use TIOA output
from TC0 as the trigger

                                           ADC_TRIG0   External : ADCTRG
                                           ADC_TRIG1   TIOA Output of TC
Channel 0
                                           ADC_TRIG2   TIOA Output of TC
Channel 1
                                           ADC_TRIG3   TIOA Output of TC
Channel 2
                                           ADC_TRIG4   PWM Event Line 0
*/
}
```

Anexo G - Código para envio de ficheiros de áudio a partir cartão SD

```
#include <SPI.h>
#include <SD.h>
#include "RF24.h"
bool radioNumber = 0;
RF24 radio(7, 8);
byte buf[32] = {0};
File myFile;
byte addresses[][6] = {"1Node", "2Node"};
bool role = 0;
void setup() {
  pinMode(53, OUTPUT);
  digitalWrite(53, LOW);
  Serial.begin (115200) ;
  if (!SD.begin(4))
  {
    Serial.println("initialization of SD failed!");
    return;
  }
  Serial.println("initialization done.");
  digitalWrite(53, HIGH);

  Serial.println(F("E M I S S O R"));
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setPayloadSize(32);
 radio.setDataRate(RF24_250KBPS);
 //   radio.setDataRate(RF24_2MBPS);
  radio.setChannel(0x7D);
  radio.printDetails();
  radio.setAutoAck(1);                    // Ensure autoACK is enabled
  radio.setRetries(5, 15);
  if (radioNumber) {
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1, addresses[0]);
  }
  else
  { radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1, addresses[1]);
  }
  delay(2000);
  sendfile();
}
void readfile() {
  Serial.println("Transmissao iniciada");
  byte i = 0;
  while (myFile.available())
  {
    buf[i] = myFile.read();

    i++;
    if (i == 31)
    {
      buf[31] = 100;

      radio.write(buf, 32);
    //  Serial.write(buf, 31);
   //   Serial.println("");
    delay(1);
      i = 0;
```

```
    }
  }
  if (i > 0) {
    buf[31] = i;
    radio.write(buf, 32);
    Serial.write(buf, i); //Output results at serial port
    delay(1);
    Serial.println("");
    Serial.println("Last stream of less than 32 bytes");
    Serial.println(i);
    digitalWrite(53, LOW);
    i = 0;
  }
}
void sendfile() {

  myFile = SD.open("avicii25.wav");
  if (myFile)
  {
    readfile ();// Função que vai fazer o envio
    myFile.close();
    Serial.println("Ficheiro enviado");
//   radio.powerDown(); //desliga o rádio
  }
  else
  {
    Serial.println("error opening file"); // se não abre file, erro
  }
}
void loop() {
  // nada a fazer aqui
}
```

Anexo H - Código para receção de ficheiros e gravação em cartão SD

```
#include "RF24.h"
#include <SPI.h>
#include <SD.h>

const int chipSelect = 4;
const int fileclose = 6;
File myFile;
unsigned long tempo;
unsigned long tempofile;
//unsigned long tempofile2;
unsigned long tempo2;
int flag = 0;
bool radioNumber = 1;
RF24 radio(7, 8);

byte buf[32] = {0};
byte addresses[][6] = {"1Node", "2Node"};
bool role = 0;



void setup()
{ pinMode(53, OUTPUT);
  digitalWrite(53, LOW);
  pinMode(2, OUTPUT);
  pinMode(fileclose, INPUT_PULLUP);

  Serial.begin (115200) ;

  if (!SD.begin(4))
  {
    Serial.println("initialization of SD failed!");
    return;
  }
  Serial.println("initialization done.");
  // digitalWrite(53, HIGH);

  Serial.println(F("R E C E T O R"));
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setPayloadSize(32);
  radio.setDataRate(RF24_2MBPS);
  radio.setChannel(0x7D);
  // radio.setDataRate(RF24_250KBPS);
  radio.setAutoAck(1);                    // Ensure autoACK is enabled
  radio.setRetries(5, 15);
  digitalWrite(13, LOW);
  radio.printDetails();
  radio.startListening();
  if (radioNumber)
  {
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1, addresses[0]);
  }
  else
  {
    radio.openWritingPipe(addresses[0]);
```

```
    radio.openReadingPipe(1, addresses[1]);
  }
}
void openfile()
{
  myFile = SD.open("6600.txt", FILE_WRITE);
  Serial.println("Incoming file");
  tempofile = millis();
  flag = 1;
}

void loop()

{
  if (radio.available()) {
    if (flag == 0) {
      openfile();
    }
    radio.read(buf, 32);
    byte i = buf[31];
    if (i == 100) {

      myFile.write(buf, 31);
  //    Serial.write(buf, 31);
  //    Serial.println("");
      tempo = millis();
    }
    else
    {
    //   Serial.write(buf, i);
       myFile.write(buf, i);
       Serial.println("");

    }
  }
  tempo2 = millis();
  if (tempo2 - tempo > 4000 && flag == 1) {
    digitalWrite(2, HIGH);
    Serial.println(F("File closed"));
    Serial.println("Tempo de envio em segundos");
    Serial.print((tempo2 - tempofile) / 1000);
    myFile.close();
    flag = 0;
  }
}
```

Anexo I - Código para medição do débito de transmissão

```
/*
 General Data Transfer Rate Test

 */


#include <SPI.h>
#include "RF24.h"

/*************  USER Configuration *****************************/
                                        // Hardware configuration
RF24 radio(7,8);                        // Set up nRF24L01 radio on SPI
bus plus pins 7 & 8

/***************************************************************/

const uint64_t pipes[2] = { 0xABCDABCD71LL, 0x544d52687CLL };   // Radio
pipe addresses for the 2 nodes to communicate.

byte data[32];                          //Data buffer for testing data
transfer speeds

unsigned long counter, rxTimer;         //Counter and timer for keeping
track transfer info
unsigned long startTime, stopTime;
bool TX=1,RX=0,role=0;

void setup(void) {

  Serial.begin(115200);

  radio.begin();                        // Setup and configure rf radio
  radio.setChannel(1);
  radio.setPALevel(RF24_PA_MAX);        // If you want to save power
use "RF24_PA_MIN" but keep in mind that reduces the module's range
  radio.setDataRate(RF24_1MBPS);
  radio.setAutoAck(1);                  // Ensure autoACK is enabled
  radio.setRetries(2,15);               // Optionally, increase the
delay between retries & # of retries

  radio.setCRCLength(RF24_CRC_8);       // Use 8-bit CRC for
performance
  radio.openWritingPipe(pipes[0]);
  radio.openReadingPipe(1,pipes[1]);

  radio.startListening();               // Start listening
  radio.printDetails();                 // Dump the configuration of
the rf unit for debugging

  Serial.println(F("\n\rRF24/examples/Transfer/"));
  Serial.println(F("*** PRESS 'T' to begin transmitting to the other
node"));

  randomSeed(analogRead(0));            //Seed for random number
generation

  for(int i = 0; i < 32; i++){
     data[i] = random(255);             //Load the buffer with random
data
```

```
  }
  radio.powerUp();                          //Power up the radio
}

void loop(void){


  if(role == TX){
    delay(2000);

    Serial.println(F("Initiating Basic Data Transfer"));


    unsigned long cycles = 10000; //Change this to a higher or lower
number.

    startTime = millis();
    unsigned long pauseTime = millis();

    for(int i=0; i<cycles; i++){        //Loop through a number of cycles
      data[0] = i;                      //Change the first byte of the
payload for identification
      if(!radio.writeFast(&data,32)){   //Write to the FIFO buffers
        counter++;                      //Keep count of failed payloads
      }

      //This is only required when NO ACK ( enableAutoAck(0) ) payloads
are used
//      if(millis() - pauseTime > 3){
//        pauseTime = millis();
//        radio.txStandBy();          // Need to drop out of TX mode every
4ms if sending a steady stream of multicast data
//        //delayMicroseconds(130);     // This gives the PLL time to sync
back up
//      }

    }

   stopTime = millis();
                                       //This should be called to wait
for completion and put the radio in standby mode after transmission,
returns 0 if data still in FIFO (timed out), 1 if success
   if(!radio.txStandBy()){ counter+=3; } //Standby, block only until FIFO
empty or auto-retry timeout. Flush TX FIFO if failed
   //radio.txStandBy(1000);              //Standby, using extended timeout
period of 1 second

   float numBytes = cycles*32;
   float rate = numBytes / (stopTime - startTime);

   Serial.print("Transfer complete at "); Serial.print(rate);
Serial.println(" KB/s");
   Serial.print(counter); Serial.print(" of "); Serial.print(cycles);
Serial.println(" Packets Failed to Send");
   counter = 0;

   }



if(role == RX){
```

```
      while(radio.available()){
       radio.read(&data,32);
       counter++;
      }
    if(millis() - rxTimer > 1000){
      rxTimer = millis();
      unsigned long numBytes = counter*32;
      Serial.print(F("Rate: "));
      //Prevent dividing into 0, which will cause issues over a period of
time
      Serial.println(numBytes > 0 ? numBytes/1000.0:0);
      Serial.print(F("Payload Count: "));
      Serial.println(counter);
      counter = 0;
    }
  }
  //
  // Change roles
  //

  if ( Serial.available() )
  {
    char c = toupper(Serial.read());
    if ( c == 'T' && role == RX )
    {
      Serial.println(F("*** CHANGING TO TRANSMIT ROLE -- PRESS 'R' TO
SWITCH BACK"));
      radio.openWritingPipe(pipes[1]);
      radio.openReadingPipe(1,pipes[0]);
      radio.stopListening();
      role = TX;                  // Become the primary transmitter (ping
out)
    }
    else if ( c == 'R' && role == TX )
    {
      radio.openWritingPipe(pipes[0]);
      radio.openReadingPipe(1,pipes[1]);
      radio.startListening();
      Serial.println(F("*** CHANGING TO RECEIVE ROLE -- PRESS 'T' TO
SWITCH BACK"));
      role = RX;                // Become the primary receiver (pong back)
    }
  }
}
```

Anexo J - Código de *Pass-Through* para Arduino DUE

```
#include <AutoAnalogAudio.h>

AutoAnalog aaAudio;

//int32_t bufferSize = MAX_BUFFER_SIZE;
int32_t bufferSize = 32;

uint8_t dacChannel = 2; // Use DAC0 for output
// uint8_t dacChannel = 1; // Use DAC1 for output
// uint8_t dacChannel = 2; // Use both DAC0 and DAC1

void DACC_Handler(void){
    //Link the DAC ISR/IRQ library. Called by the MCU when DAC is ready
for data.
    aaAudio.dacHandler();
}

/*********************************************************/
void setup() {
    // Setup aaAudio using both ADC and DAC
    aaAudio.begin(true, true);

    // Disable auto adjust of timers (ensure a constant sample rate)
    aaAudio.autoAdjust = false;

    // Set quantization to 12 bits
    aaAudio.adcBitsPerSample = 12;
   aaAudio.dacBitsPerSample = 12;

    // Set a 16kHz sample rate
    aaAudio.setSampleRate(88200);

    // Set ADC channel
    aaAudio.enableAdcChannel(0);

    // Start loading ADC buffers
    aaAudio.getADC(bufferSize);
    aaAudio.feedDAC(dacChannel, bufferSize);
}

/*********************************************************/
void loop() {
    // Get samples from the ADC at the sample rate defined above
    // Note: This function only blocks if the ADC is currently sampling
and autoAdjust is set to 0
    // As long as any additional code completes before the ADC is finished
sampling, a continuous stream of ADC data
    // at the defined sample rate will be available
    aaAudio.getADC(bufferSize);

    // Pass-through
    for(int i = 0; i < bufferSize; i++){
        aaAudio.dacBuffer16[i] = aaAudio.adcBuffer16[i];
    }

    // Output
    aaAudio.feedDAC(dacChannel, bufferSize);

}
```