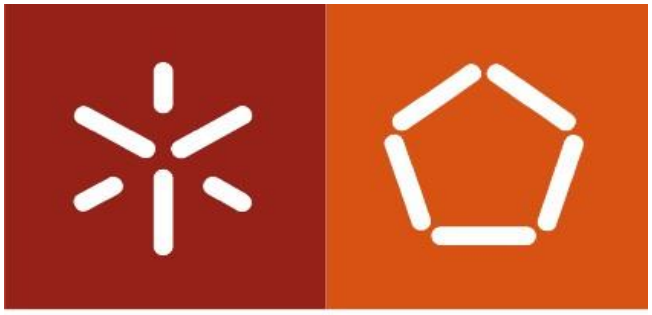


Universidade do Minho

Relatório

DEMONSTRAÇÃO 1

LABORATÓRIOS DE TELECOMUNICAÇÕES E INFORMÁTICA I

UNIVERSIDADE DO MINHO | MESTRADO INTEGRADO EM TELECOMUNICAÇÕES E INFORMÁTICA

3º ANO

Índice

Constituição do grupo

Nome: André Machado
Número Mecanográfico: A66693
E-mail: andre.machado93@gmail.com
Correio eletrónico institucional: a66693@alunos.uminho.pt

Nome: Carlos Carvalho
Número Mecanográfico: A75401
E-mail: carlosc.96@hotmail.com
Correio eletrónico institucional: a75401@alunos.uminho.pt

Nome: Rui Silva
Número Mecanográfico: A69987
E-mail: rui.silva.kj@gmail.com
Correio eletrónico institucional: a69987@alunos.uminho.pt

1. Introdução

No âmbito da Unidade Curricular de Laboratórios de Telecomunicações e Informática I do curso Mestrado Integrado em Engenharia de Telecomunicações e Informática, foi proposta aos alunos o desenvolvimento de um serviço de partilha de ficheiros entre dois ou mais dispositivos, utilizando comunicação sem fios. Este projeto envolve a investigação e aprendizagem de tecnologias novas, nomeadamente, trabalhar com módulo Arduíno e com módulo nRF24L01.

Este projeto vai incidir sobretudo em 3 camadas, a camada física, a camada aplicação e a camada de ligação de dados.

A fase I é o início do projeto e pretende-se que o grupo seja capaz de obter conhecimentos sobre o funcionamento dos componentes de hardware, que ainda também consiga entender o funcionamento do diverso software necessário para o projeto e que sejam capazes de fazer o planeamento de todos os objetivos e tarefas ao longo do tempo.

Após toda essa descoberta e aquisição de conhecimentos teóricos e planeamento do projeto, é necessário que se prossiga para a execução do projeto.

Devem ser implementados diversos testes de modo a que seja possível levantar valores práticos para posterior análise.

É deste modo que o grupo pretende em seguida demonstrar e documentar todo esse progresso do projeto e toda a análise de dados efetuada.

2. Enquadramento do projeto

O projeto consiste na elaboração de um sistema capaz de realizar comunicação de dados (ficheiros) entre dois ou mais dispositivos. A partilha deve ser realizada através de uma rede local sem fios por radiofrequência (RF). A partilha será bidirecional, contudo não pode ser simultânea, pois um módulo não poderá enviar informação enquanto recebe e vice-versa. O objetivo é atingir a maior distância possível mantendo uma comunicação fiável no âmbito de transmissão de ficheiros.

Além dos objetivos técnicos que este processo envolve, também se pretende que os elementos do grupo aumentem a sua capacidade de trabalho, tanto autonomamente como em grupo. Que tenham uma aprendizagem de novas tecnologias e que compactem o conhecimento adquirido em outras unidades curriculares, de modo a que o aluno tenha melhores competências para a sua vida profissional. Entre as várias competências encontra-se o conhecimento e capacidade de implementação modelo de OSI (*Open Systems Interconnection*), conhecimento de interfaces de comunicação com periféricos, saber utilizar componentes disponíveis no circuito comercial (Arduíno e nRF24L01), melhorar a gestão de tempo e a coordenação/colaboração com os vários elementos de um grupo.

A partir dos requisitos técnicos do projeto foi elaborada a seguinte arquitetura para o sistema:

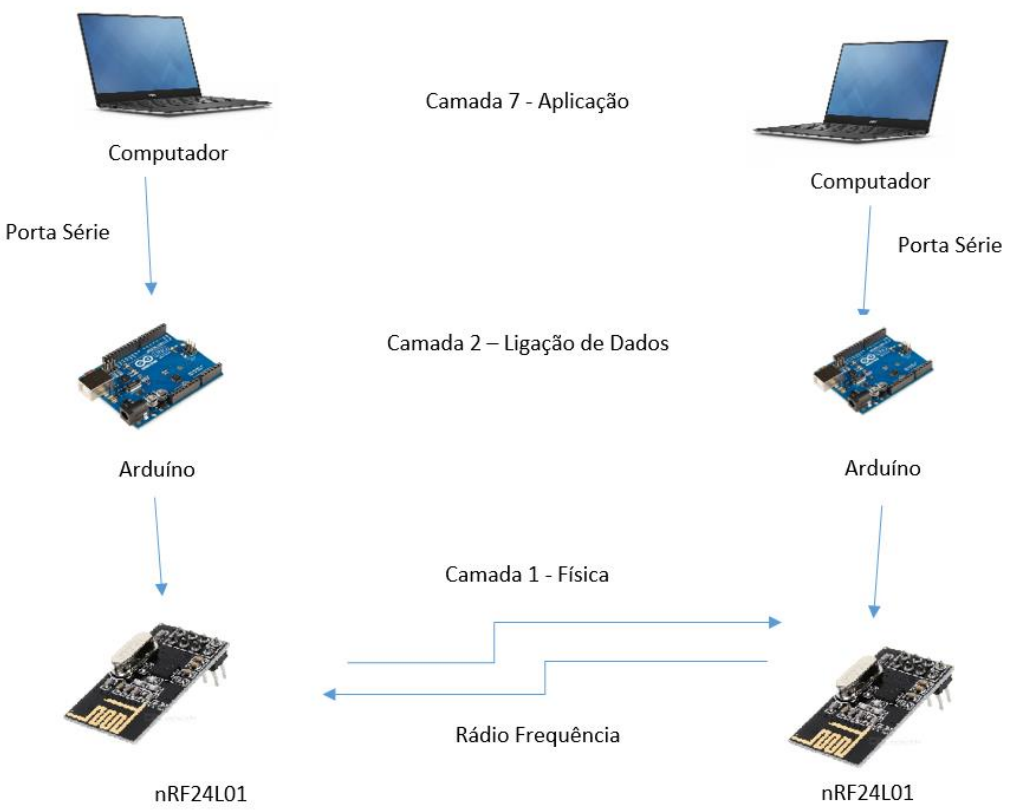


Figura 1 - Arquitetura do projeto.

2.1. Funcionalidades a serem oferecidas pelo Sistema

A funcionalidade base do sistema será a partilha de ficheiros entre, pelo menos, dois dispositivos, através de comunicação sem fios.

Esta funcionalidade base será alargada à possibilidade de existir bidirecionalidade na comunicação entre dispositivos, cada transceiver poderá efetuar transmissão e receção de dados.

O sistema irá fazer também gestão de concorrência na receção de dados assim como efetuar a deteção de possíveis erros induzidos pela colisão de ondas, pelo ruído, pela atenuação.

Tendo em conta as especificações do transceiver apenas será possível a receção de 6 dispositivos diferentes, uma vez que o módulo RF tem apenas 6 *pipes* de leitura/escrita.

Na primeira fase do projeto será apenas apresentada a comunicação de dados aleatórios entre os transceivers. Durante esta fase o grupo irá tentar compreender todas as variáveis que influenciam a transmissão de dados e funcionamento dos transceivers, focando apenas a comunicação entre os módulos RF e a compreensão do funcionamento destes.

2.2. Camadas OSI

A comunicação entre o emissor e o recetor é baseada no modelo de OSI desenvolvido pela ISO (*International Organization for Standardization*). O modelo OSI é um modelo que tem como principal objetivo definir um standard para protocolos de comunicação e assim estabelecer comunicações entre vários sistemas independentemente da sua constituição e tecnologia usada.

O modelo OSI é constituído por 7 camadas e cada uma tem a seu cargo funções especificas, cada uma fornece serviços ás superiores e é servida pela anterior. As camadas são as seguintes (1):

7 – Camada da Aplicação (*Application*)
6 – Camada da Apresentação (*Presentation*)
5 – Camada do Sessão (*Session*)
4 – Camada do Transporte (*Transport*)
3 – Camada da Rede (*Network*)
2 – Camada de Dados (*Data Link*)
1 – Camada Física (*Physical*)

Em cada camada são implementados vários protocolos que visam criar uma norma nas comunicações, como por exemplo:

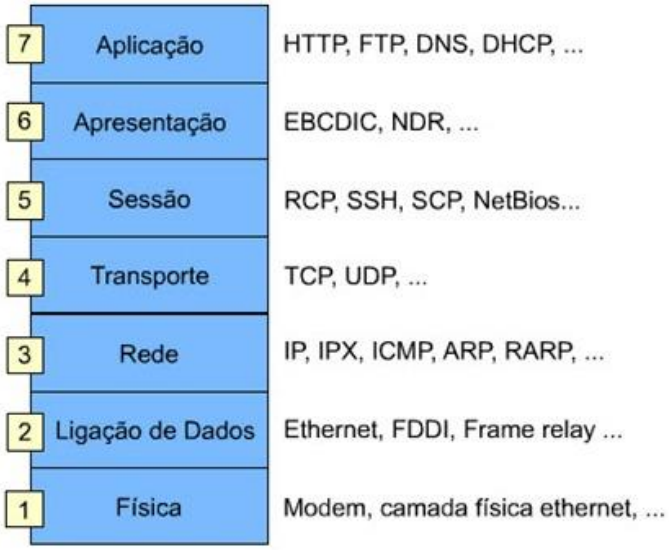


Figura 2 - Representação das diferentes camadas do modelo OSI. (2)

Durante a elaboração deste projeto serão utilizadas três das sete camadas, sendo elas:

- a camadas da aplicação;
- a camada de ligação de Dados;
- a camada física.

2.2.1. Camada Física

A camada física é a mais baixa do modelo OSI e diz respeito à emissão e receção de bits num meio físico. Esta camada descreve as características elétricas e físicas da conexão. Define a relação entre um dispositivo e o meio de transmissão, isto inclui a disposição dos pinos, tensões, cabos, etc. (3)

No caso deste projeto, a transmissão de informação será realizada através de transceivers de radiofrequência, nomeadamente o módulo nRF24L01.

2.2.1.1. Funcionamento da Comunicação por RF

A comunicação por RF é feita através do envio, por parte do emissor, de ondas eletromagnéticas que se propagam à velocidade da luz e posteriormente captadas por um determinado recetor. As ondas eletromagnéticas são criadas a partir da oscilação de um eletrão numa determinada localização. A oscilação desse eletrão vai definir a frequência da onda, sendo a frequência inversamente proporcional ao comprimento da onda. (4)

Dentro do espectro eletromagnético, a gama de frequências das ondas rádio foi dividida em diversas faixas.

f	λ	Band	Description
30–300 Hz	10^4–10^3 km	ELF	Extremely low frequency
300–3000 Hz	10^3–10^2 km	VF	Voice frequency
3–30 kHz	100–10 km	VLF	Very low frequency
30–300 kHz	10–1 km	LF	Low frequency
0.3–3 MHz	1–0.1 km	MF	Medium frequency
3–30 MHz	100–10 m	HF	High frequency
30–300 MHz	10–1 m	VHF	Very high frequency
300–3000 MHz	100–10 cm	UHF	Ultra-high frequency
3–30 GHz	10–1 cm	SHF	Superhigh frequency
30–300 GHz	10–1 mm	EHF	Extremely high frequency (millimeter waves)

Figura 3 –Gama de frequências da radiação.

2.2.2. Camada de Ligação de Dados

A camada de ligação de dados é a segunda camada.

A camada de ligação de dados tem como tarefa controlar a troca de informação local (Logic Link Control – LLC). Nomeadamente:

- Delimitação das tramas, que consiste na identificação do início e fim de uma trama, no emissor e no recetor
- Endereçamento das estações envolvidas.

Esta camada tem também atribuída a tarefa de gerir o acesso ao meio, para que, num meio onde hajam múltiplos dispositivos a transmitir hajam regras sobre o

funcionamento de forma a evitar colisões quando se enviam dados. (Media Access Control - MAC). (1)

As funcionalidades desta camada vão ser implementadas em Arduíno UNO.

2.2.3. Camada Aplicação

A camada da aplicação é a sétima camada do modelo OSI.

Esta camada fornece uma interface para a comunicação de rede. Por convenção, esta camada faz a comunicação entre a rede e as aplicações instaladas na máquina. Apesar de ser a camada mais perto do utilizador e do nome parecer indicar que esta camada se relaciona com a aplicação em si, está mais relacionada com os protocolos que gerem a troca de mensagens. (1)

2.3. Módulo nRF24L01+

O nRF24L01+ é um transceiver de Radiofrequência que trabalha nas frequências *ISM* de 2.4 GHz (5).

O nRF21L01+ é configurado e operado através da sua SPI (*Serial Peripheral Interface*), com o auxílio de um microcontrolador. O mapa de registos do transceiver, acessível através da SPI, pode ser acedido a partir de qualquer modo de funcionamento do chip. Todo o funcionamento do transceiver será controlado e manuseado através da manipulação de bits nestes registos.

Interface de Controlo e Dados – Interface que permite aceder a todos os recursos do nRF24L01, através dos pinos:

- **IRQ –** Sinal ativo a *Low, Interruption Request*.
- **CE –** Sinal ativo a *High*, usado para definir o transceiver como RX ou TX
- **CSN –** Sinal SPI (linha *Slave Select*)
- **SCK –** Sinal SPI (*linha Serial Clock*)
- **MOSI –** Sinal SPI (linha *Master Out*/*Slave In)*
- **MISO –** sinal SPI (linha *Master In/Slave Out*)

Figura 4 - Pinos do Módulo nRF24L01+ (10).

Para manipular os registos do nRF24L01+ o fabricante disponibilizou os comandos necessários no *datasheet*. Segue-se a tabela referente a esses comandos.

Tabela 1 - Comandos SPI.

Command name	Command word (binary)	# Data bytes	Operation
R_REGISTER	000A AAAA	1 to 5 LSByte first	Read command and status registers. AAAAA = 5 bit Register Map Address
W_REGISTER	001A AAAA	1 to 5 LSByte first	Write command and status registers. AAAAA = 5 bit Register Map Address Executable in power down or standby modes only.
R_RX_PAYLOAD	0110 0001	1 to 32 LSByte first	Read RX-payload: 1 – 32 bytes. A read operation always starts at byte 0. Payload is deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	1010 0000	1 to 32 LSByte first	Write TX-payload: 1 – 32 bytes. A write operation always starts at byte 0 used in TX payload.
FLUSH_TX	1110 0001	0	Flush TX FIFO, used in TX mode
FLUSH_RX	1110 0010	0	Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, that is, acknowledge package will not be completed.
REUSE_TX_PL	1110 0011	0	Used for a PTX device Reuse last transmitted payload. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deacti-vated during package transmission.
R_RX_PL_WID[a]	0110 0000	1	Read RX payload width for the top R_RX_PAYLOAD in the RX FIFO.
W_ACK_PAYLOAD[a]	1010 1PPP	1 to 32 LSByte first	Used in RX mode. Write Payload to be transmitted together with ACK packet on PIPE PPP. (PPP valid in the range from 000 to 101). Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled using first in - first out principle. Write payload: 1– 32 bytes. A write operation always starts at byte 0.
W_TX_PAYLOAD_NO ACK[a]	1011 0000	1 to 32 LSByte first	Used in TX mode. Disables AUTOACK on this specific packet.
NOP	1111 1111	0	No Operation. Might be used to read the STATUS register

Para sumarizar, o nRF24L01 disponibiliza:

- Transmissão e Receção de rádio na banda ISM, ao longo de 127 canais;
- Potência de output programável para o transmissor de: 0, -6, -12 ou -18dBm;
- On-Air data rate de 250 Kbps, 1 Mbps ou 2 Mbps;
- *Payloads* dinâmicos;
- Tratamento automático de tramas;
- 6 canais de dados multiceiver;
- Uma interface SPI.

2.4. Conhecer as características das interfaces de comunicação série entre o transceiver de RF e o módulo Arduino e entre este e o computador pessoal;

O nRF24L01+ irá comunicar com o Arduino através da interface SPI (*Serial Peripheral Interface*).

SPI (6)– É uma forma de comunicação série síncrona usada para transferência de dados sobre curtas distâncias, geralmente utilizada na comunicação entre microcontroladores e sensores ou outros dispositivos periféricos. É uma comunicação *Full-Duplex*, ou seja, pode haver comunicação em simultâneo por parte dos intervenientes.

Comunicação série implica que os dados serão transmitidos bit por bit pelo canal de transmissão/comunicação. Ser síncrona implica que a comunicação é regulada por um *clock*, isto significa que a leitura dos bits vai ocorrer aquando uma transição de *clock*.

A SPI utiliza linhas separadas para a comunicação de dados e para o *clock*. Este irá sinalizar o recetor no momento em que deve ler um bit da linha de dados, seja através da transição ascendente ou da transição descendente. Quando o recetor deteta essa transição ele irá imediatamente ler o bit que está a receber na linha de dados.

Na SPI o "lado" do sistema que define o *clock* é chamado de *master* e o outro lado é chamado *slave (*Na comunicação entre o nRF24L01+ e o Arduino este último será sempre o master). Na SPI só existe um *master,* mas podem existir vários *slaves*.

Quando dados são enviados do master para o *slave* é utilizada a linha de dados chamada MOSI (*Master Out/Slave In*). Caso um *slave* precise de enviar uma resposta ao master, o último continuará a gerar o *clock* e o *slave* irá enviar os dados através de uma linha de dados chamada MISO (*Master In/Slave Out*). Neste último caso o *clock* gerado pelo master é predefinido uma vez que este sabe quando um *slave* tem de responder e a quantidade de dados com que vai responder.

Um outro aspeto da SPI é o *Slave Select*, uma linha de comunicação que serve para dizer ao *slave* que este vai ser utilizado para receber/enviar dados. Esta linha é igualmente utilizada para selecionar o *slave* a ser utilizado quando o sistema tem dois ou mais *slaves*. A *Slave Select* está normalmente a nível alto, no momento anterior a serem enviados dados para o *slave* a linha é posta em nível baixo, que irá ativar o recetor.

2.5. Ligações Físicas entre os componentes utilizados

Para o desenrolar do projeto tivemos que analisar os *Datasheets* dos diversos componentes para poder efetuar as diferentes ligações.

O módulo nRF24L01 é constituído por 8 pinos:

- VCC
- CE
- CSN
- SCK
- MOSI
- MISO
- IRQ
- GND

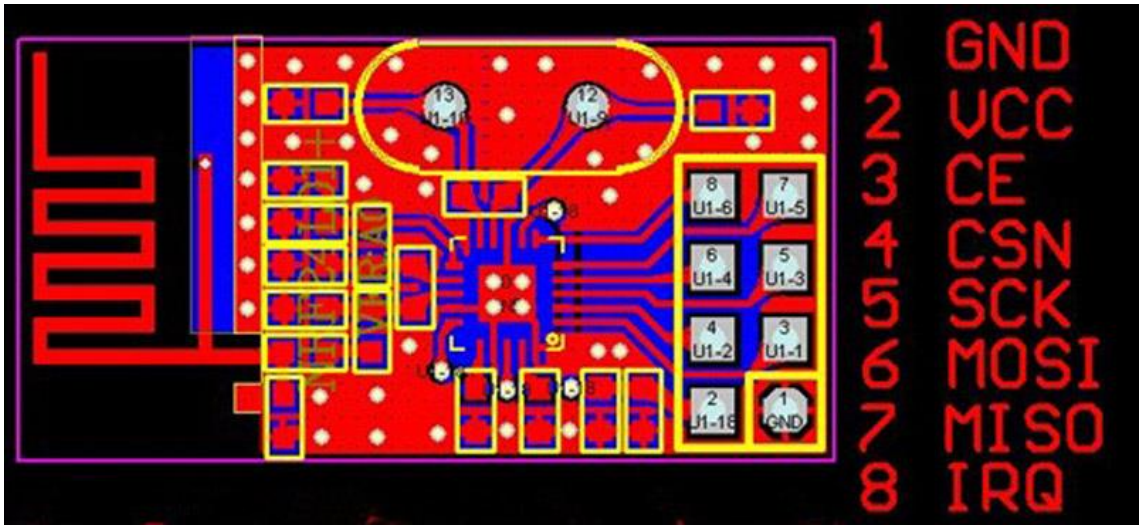


Figura 5 - Pinos módulo nRF24L01 (7).

.

O pino VCC necessita de uma tensão para alimentação de 3.3 volts, de modo que ligamos ao 3.3v do Arduino. Para o *Ground* (GND) ligamos a um dos GND's disponíveis no Arduino. Já os pinos CSN, MOSI, MISO e SCK ligamos aos pinos 10 (SS), 11(MOSI), 12 (MISO) e 13(SCK), respetivamente pois são pinos que suportam comunicação SPI que podem ser acedidos pela biblioteca SPI.h. O pino CE ligamos ao pino 9 apesar de termos mais opções (8).

Tabela 2 - Ligações Anduino-Módulo nRF24L01 e suas funcionalidades.

Pino	Nome	Função	Arduino Uno
1	GND	Alimentação	GND
2	VCC	Alimentação	3.3V
3	CE	Chip Enable RX/TX	Pino 9
4	CSN	SPI Chip Select	Pino 10
5	SCK	SPI Clock	Pino 13
6	MOSI	SPI Slave Data Input	Pino 11
7	MISO	SPI Slave Data Output	Pino 12
8	IRQ	Interrupção	Não utilizado

A ligação computador-Arduino foi feita através de um cabo USB A-B em cada um.

3. Planeamento do projeto

Para que o projeto tenha um desenvolvimento consistente e para que cumpra todos os requisitos é necessário que haja um planeamento muito bem estruturado e pensado.

Com os conhecimentos que se foi obtendo de outras unidades curriculares, em complemento ao que foi exigido no enunciado, o grupo partiu para a elaboração de um diagrama *Gantt*. O diagrama é um recurso muito eficaz para uma exposição cronológica das diferentes tarefas e objetivos assim como a exposição ao longo do tempo das tarefas pelas quais cada elemento do grupo é responsável. O diagrama possibilitou uma visão mais clara e identificação da falta ou sobrecarga de tarefas, otimizando assim o tempo.

Para que este seja bem estruturado é necessário entender a complexidade do projeto e suas diferentes fases.

3.1. Fases do projeto

O projeto está dividido em 4 fases de implementação e a partir deste capítulo pretendemos abordar o que se pretende em cada uma delas.

3.1.1. Fase 1 – Especificação do projeto e comunicação sem fios por radiofrequência

Nesta fase inicial, é pedido aos elementos do grupo que façam uma planificação do projeto, e que sejam capazes de fazer a arquitetura do sistema. É também lhes pedido que compreendam o funcionamento do hardware utilizado.

3.1.2. Fase 2 – Controlo da ligação lógica

Nesta fase é pedido ao grupo que aumente a complexidade do sistema e que aumentem o seu conhecimento acerca da transferência de dados e comunicação em redes.

3.1.3. Fase 3 – Controlo de acesso ao meio

Neste ponto do projeto pretende-se que os alunos percebam em que situações é necessário utilizar um mecanismo de controlo de acesso ao meio. Pretende-se ainda que aprofundem o conhecimento sobre diferentes protocolos de modo a entender as suas vantagens e desvantagens e testá-los no sistema.

3.1.4. Fase 4 - Camada de aplicação e respetiva aplicação

Nesta fase final pretende-se que o grupo trabalhe no nível 7 da camada OSI, ou seja, ao nível da camada de aplicação e que desenvolva um programa de partilha de ficheiro que utilize este protocolo juntamente com uma interface gráfica.

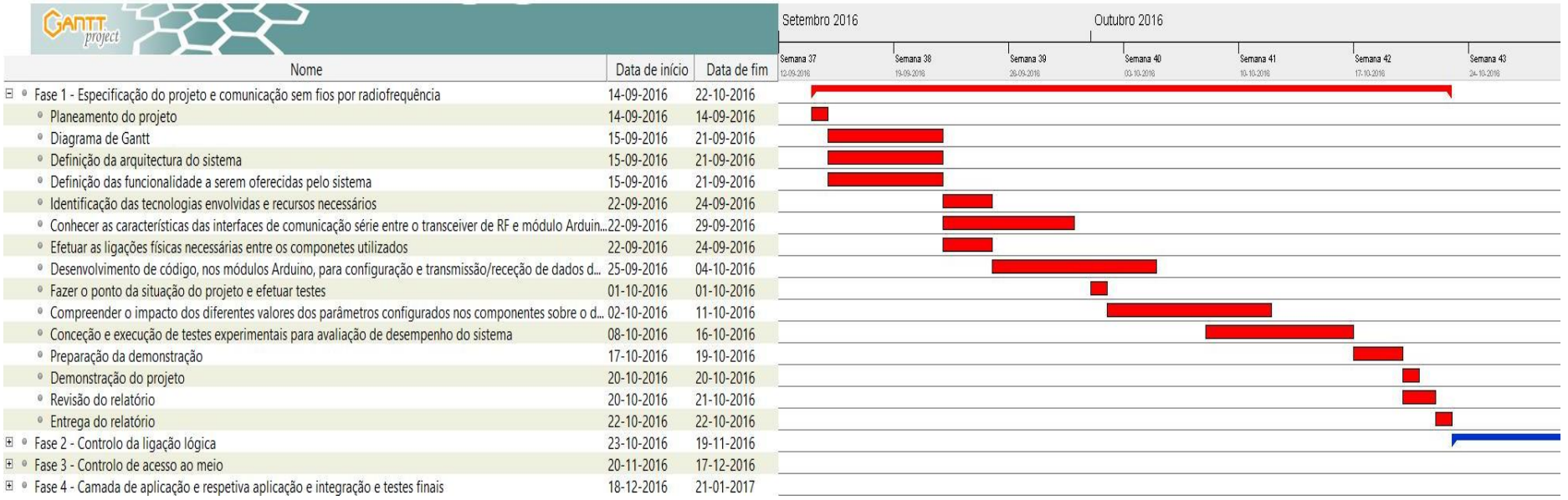
Também é pedido aos elementos do grupo que otimizem o sistema ao máximo e comparem os resultados provenientes da análise teórica.

3.2. Diagramas Gantt

3.2.1. Diagramas Gantt das fases

Nome	Data de início	Data de fim
Fase 1 - Especificação do projeto e comunicação sem fios por radiofrequência	14-09-2016	22-10-2016
Fase 2 - Controlo da ligação lógica	23-10-2016	19-11-2016
Fase 3 - Controlo de acesso ao meio	20-11-2016	17-12-2016
Fase 4 - Camada de aplicação e respetiva aplicação e integração e testes finais	18-12-2016	21-01-2017

Figura 6 - Diagrama de Gantt – Geral.

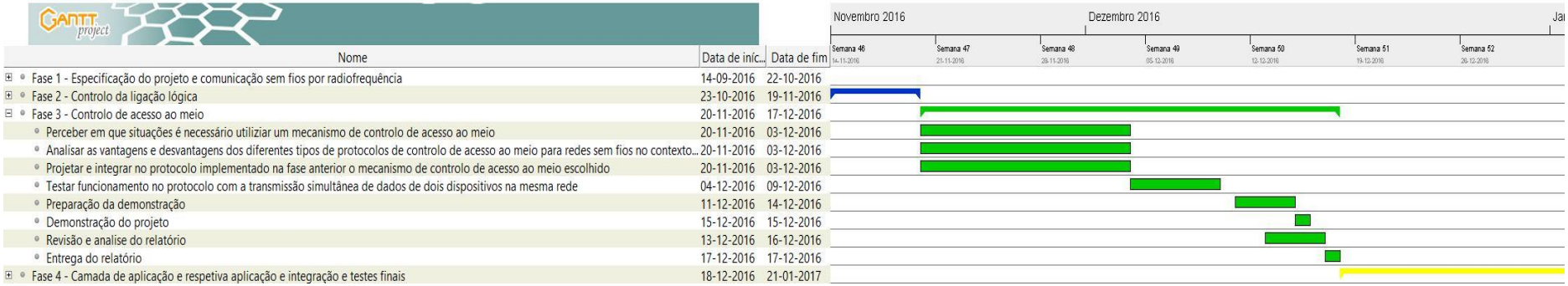


Nome	Data de início	Data de fim
Fase 1 - Especificação do projeto e comunicação sem fios por radiofrequência	14-09-2016	22-10-2016
Planeamento do projeto	14-09-2016	14-09-2016
Diagrama de Gantt	15-09-2016	21-09-2016
Definição da arquitectura do sistema	15-09-2016	21-09-2016
Definição das funcionalidade a serem oferecidas pelo sistema	15-09-2016	21-09-2016
Identificação das tecnologias envolvidas e recursos necessários	22-09-2016	24-09-2016
Conhecer as características das interfaces de comunicação série entre o transceiver de RF e módulo Arduin...	22-09-2016	29-09-2016
Efetuar as ligações físicas necessárias entre os componetes utilizados	22-09-2016	24-09-2016
Desenvolvimento de código, nos módulos Arduino, para configuração e transmissão/receção de dados d...	25-09-2016	04-10-2016
Fazer o ponto da situação do projeto e efetuar testes	01-10-2016	01-10-2016
Compreender o impacto dos diferentes valores dos parâmetros configurados nos componentes sobre o d...	02-10-2016	11-10-2016
Conceção e execução de testes experimentais para avaliação de desempenho do sistema	08-10-2016	16-10-2016
Preparação da demonstração	17-10-2016	19-10-2016
Demonstração do projeto	20-10-2016	20-10-2016
Revisão do relatório	20-10-2016	21-10-2016
Entrega do relatório	22-10-2016	22-10-2016
Fase 2 - Controlo da ligação lógica	23-10-2016	19-11-2016
Fase 3 - Controlo de acesso ao meio	20-11-2016	17-12-2016
Fase 4 - Camada de aplicação e respetiva aplicação e integração e testes finais	18-12-2016	21-01-2017

Figura 7 - Diagrama de Gantt - Fase 1.

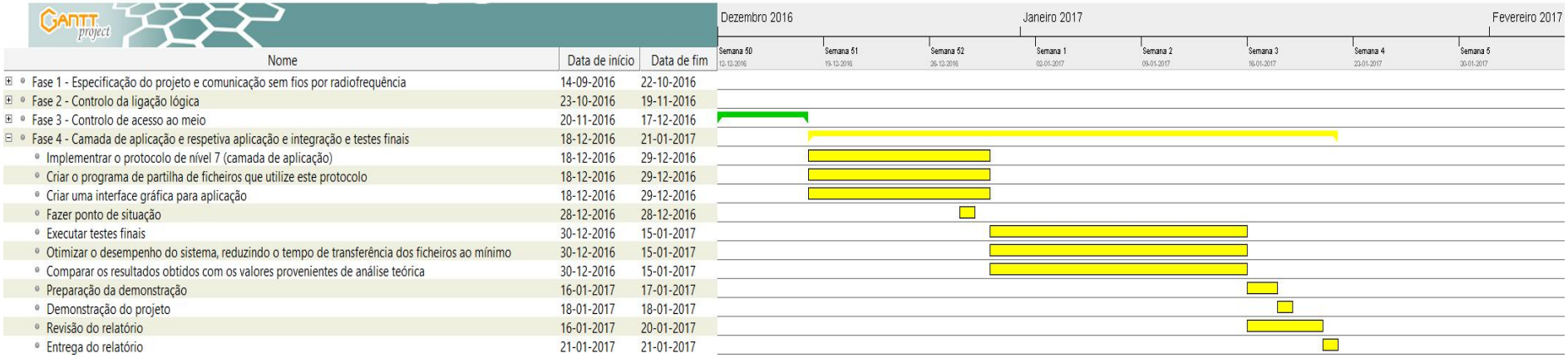
Nome	Data de iníc...	Data de fim
Fase 1 - Especificação do projeto e comunicação sem fios por radiofrequência	14-09-2016	22-10-2016
Fase 2 - Controlo da ligação lógica	23-10-2016	19-11-2016
Conhecer os dois modos de funcionamento: normal e transparente (raw) da porta série	23-10-2016	30-10-2016
Conhecer e configurar os parâmetros e modo de funcionamento da porta série	23-10-2016	30-10-2016
Perceber a noção de trama/pacote, cabeçalho, payload e cauda	23-10-2016	30-10-2016
Efetuar ponto de situação e testes ao sistema	31-10-2016	31-10-2016
Perceber como representar corretamente o estrutura de um pacote (nome, tamanho e contéudo de cada campo)	01-11-2016	07-11-2016
Definir os tipos de trama e os campos que estas necessitam de conter para proporcionar as funcionalidades desejadas	01-11-2016	07-11-2016
Especificar e conceber as primitivas de serviço a oferecer à camada superior e especificar as iterações entre os dois componentes	01-11-2016	07-11-2016
Aplicar o conceito de API	08-11-2016	14-11-2016
Efetuar a transferência de ficheiros entre dois computadores sobre a ligação sem fios	08-11-2016	14-11-2016
Preparação da demonstração	15-11-2016	15-11-2016
Demonstração do projeto	16-11-2016	16-11-2016
Revisão do relatório	17-11-2016	18-11-2016
Entrega do Relatório	19-11-2016	19-11-2016
Fase 3 - Controlo de acesso ao meio	20-11-2016	17-12-2016
Fase 4 - Camada de aplicação e respetiva aplicação e integração e testes finais	18-12-2016	21-01-2017

Figura 8 - Diagrama de Gantt - Fase 2.



Nome	Data de iníc...	Data de fim
Fase 1 - Especificação do projeto e comunicação sem fios por radiofrequência	14-09-2016	22-10-2016
Fase 2 - Controlo da ligação lógica	23-10-2016	19-11-2016
Fase 3 - Controlo de acesso ao meio	20-11-2016	17-12-2016
Perceber em que situações é necessário utiliziar um mecanismo de controlo de acesso ao meio	20-11-2016	03-12-2016
Analisar as vantagens e desvantagens dos diferentes tipos de protocolos de controlo de acesso ao meio para redes sem fios no contexto...	20-11-2016	03-12-2016
Projetar e integrar no protocolo implementado na fase anterior o mecanismo de controlo de acesso ao meio escolhido	20-11-2016	03-12-2016
Testar funcionamento no protocolo com a transmissão simultânea de dados de dois dispositivos na mesma rede	04-12-2016	09-12-2016
Preparação da demonstração	11-12-2016	14-12-2016
Demonstração do projeto	15-12-2016	15-12-2016
Revisão e analise do relatório	13-12-2016	16-12-2016
Entrega do relatório	17-12-2016	17-12-2016
Fase 4 - Camada de aplicação e respetiva aplicação e integração e testes finais	18-12-2016	21-01-2017

Figura 9 - Diagrama de Gantt - Fase 3.



Nome	Data de início	Data de fim
Fase 1 - Especificação do projeto e comunicação sem fios por radiofrequência	14-09-2016	22-10-2016
Fase 2 - Controlo da ligação lógica	23-10-2016	19-11-2016
Fase 3 - Controlo de acesso ao meio	20-11-2016	17-12-2016
Fase 4 - Camada de aplicação e respetiva aplicação e integração e testes finais	18-12-2016	21-01-2017
Implementrar o protocolo de nível 7 (camada de aplicação)	18-12-2016	29-12-2016
Criar o programa de partilha de ficheiros que utilize este protocolo	18-12-2016	29-12-2016
Criar uma interface gráfica para aplicação	18-12-2016	29-12-2016
Fazer ponto de situação	28-12-2016	28-12-2016
Executar testes finais	30-12-2016	15-01-2017
Otimizar o desempenho do sistema, reduzindo o tempo de transferência dos ficheiros ao mínimo	30-12-2016	15-01-2017
Comparar os resultados obtidos com os valores provenientes de análise teórica	30-12-2016	15-01-2017
Preparação da demonstração	16-01-2017	17-01-2017
Demonstração do projeto	18-01-2017	18-01-2017
Revisão do relatório	16-01-2017	20-01-2017
Entrega do relatório	21-01-2017	21-01-2017

Figura 10 - Diagrama de Gantt - Fase 4.

3.2.2. Diagrama de objetivos de cada elemento

3.2.2.1. Geral

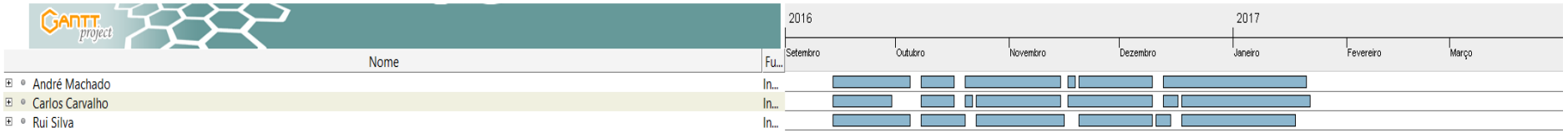


Figura 11 - Diagrama geral de tarefas.

3.2.2.2. Diagrama de tarefas do elemento André Machado

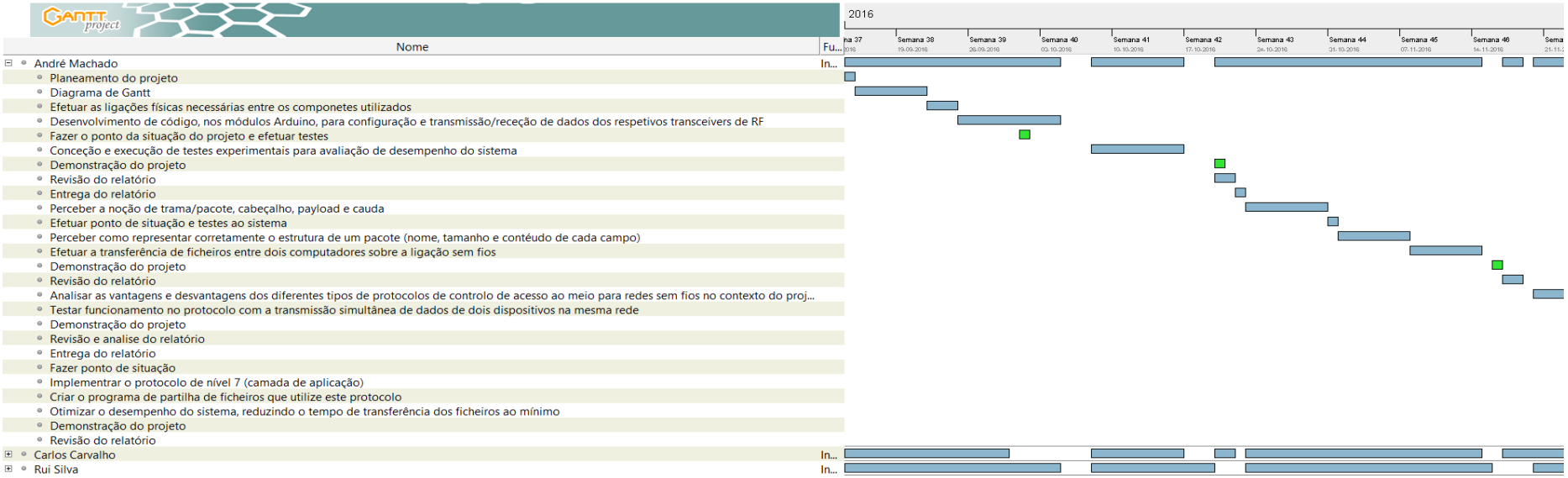


Figura 12 - Diagrama de tarefas do elemento André Machado, parte 1.

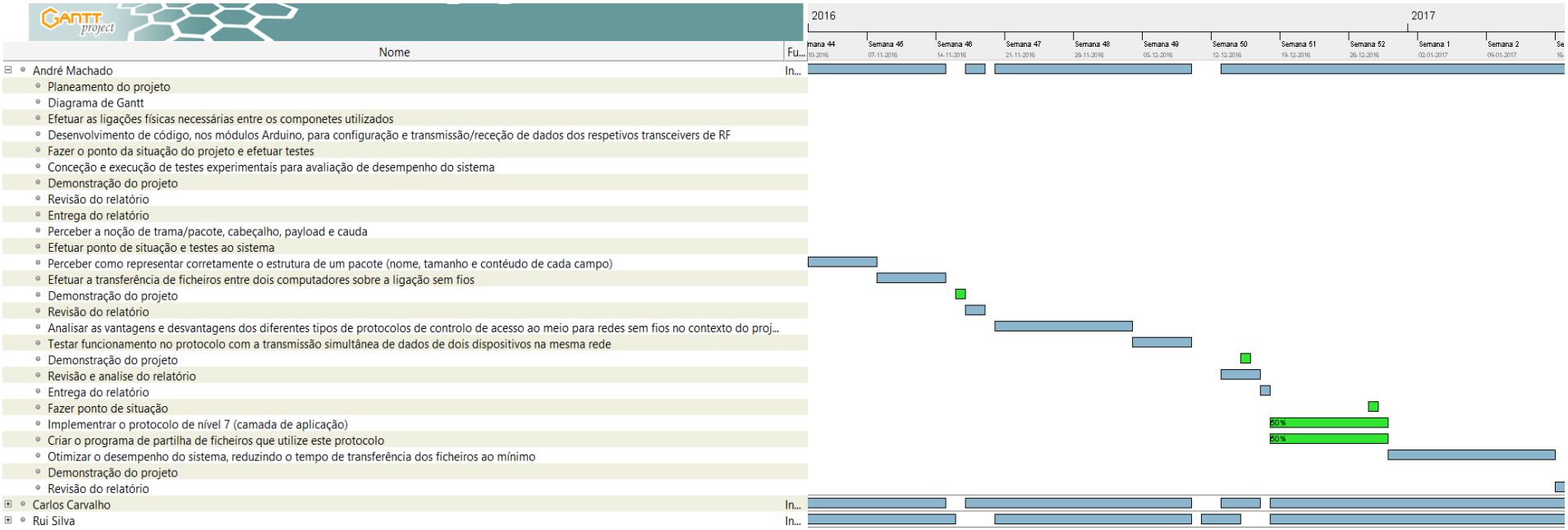


Figura 13 - Diagrama de tarefas do elemento André Machado, parte 2.

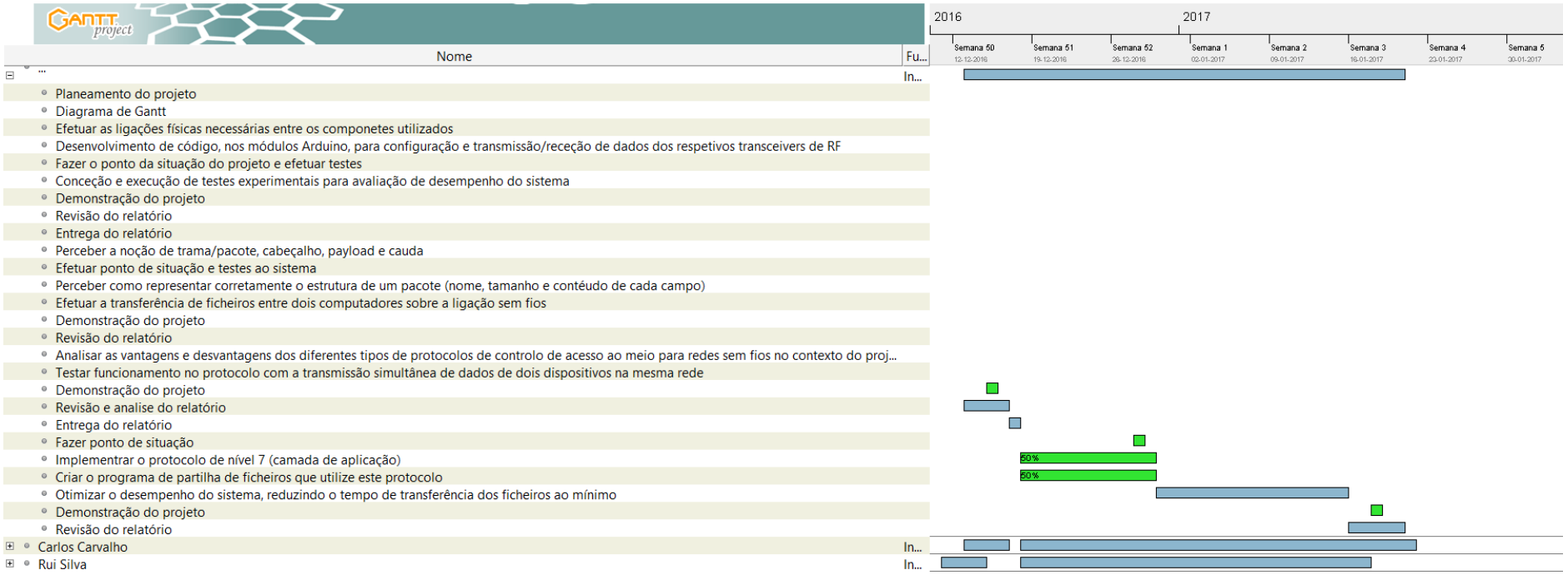


Figura 14 - Diagrama de tarefas do elemento André Machado, parte 3.

3.2.2.3. Diagrama de tarefas do elemento Carlos Carvalho

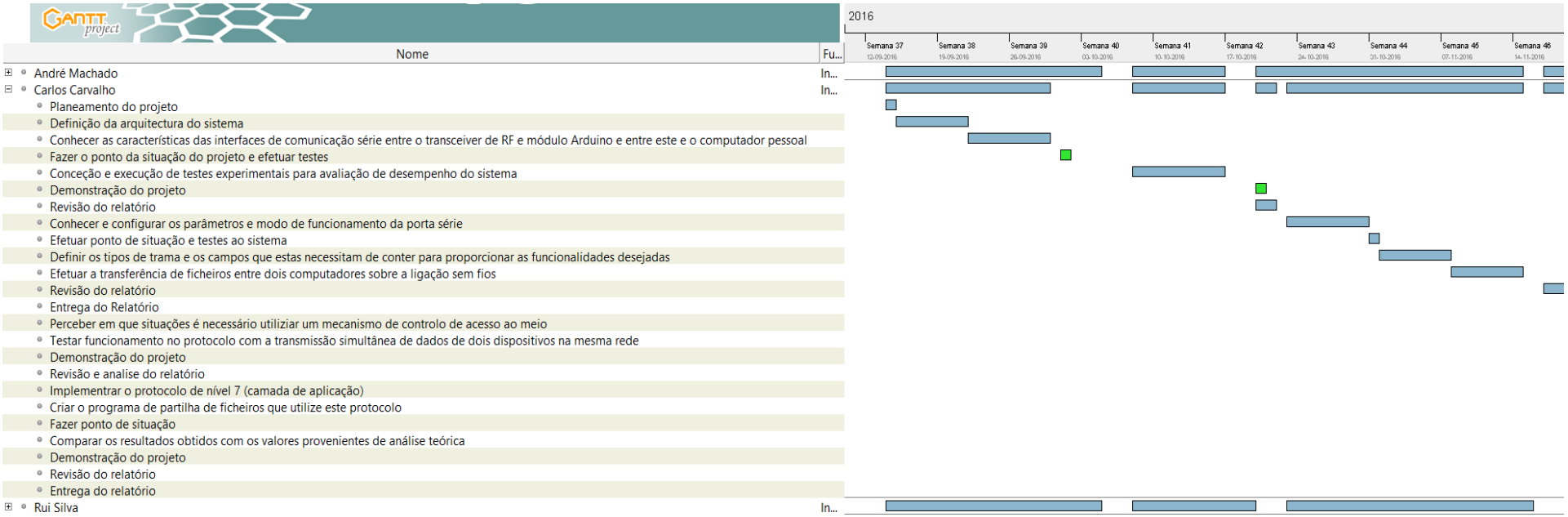


Figura 15 - Diagrama de tarefas do elemento Carlos Carvalho, parte 1.

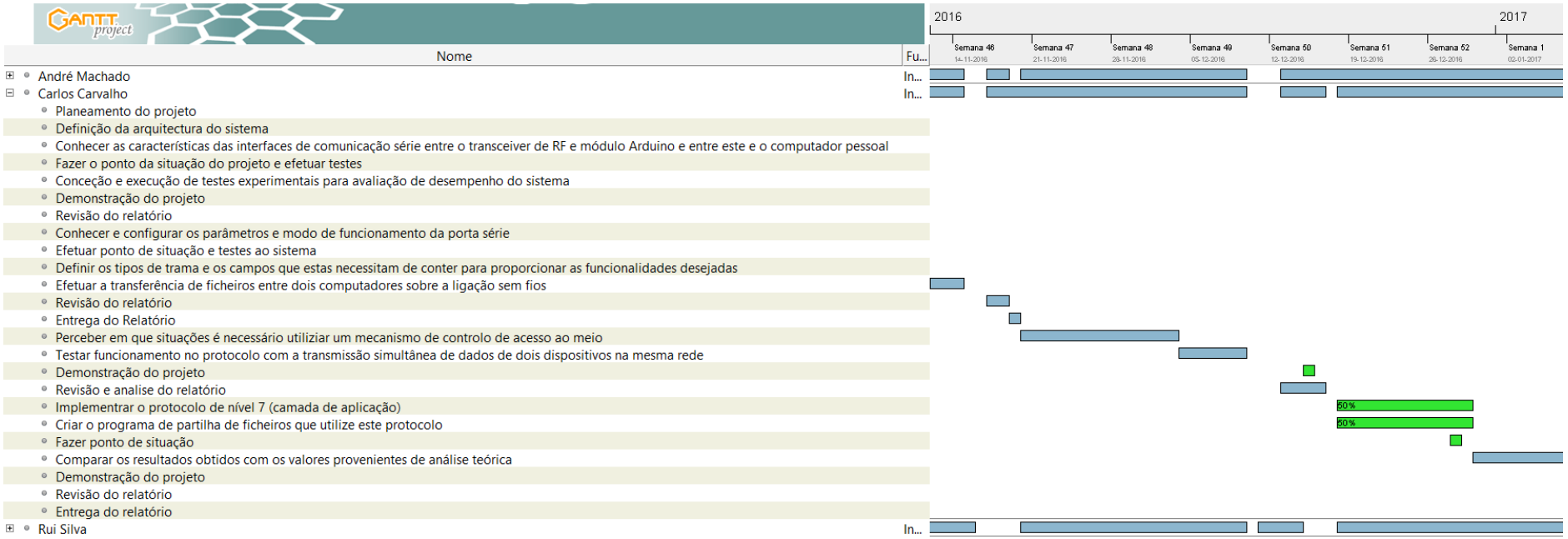


Figura 16 - Diagrama de tarefas do elemento Carlos Carvalho, parte 2.

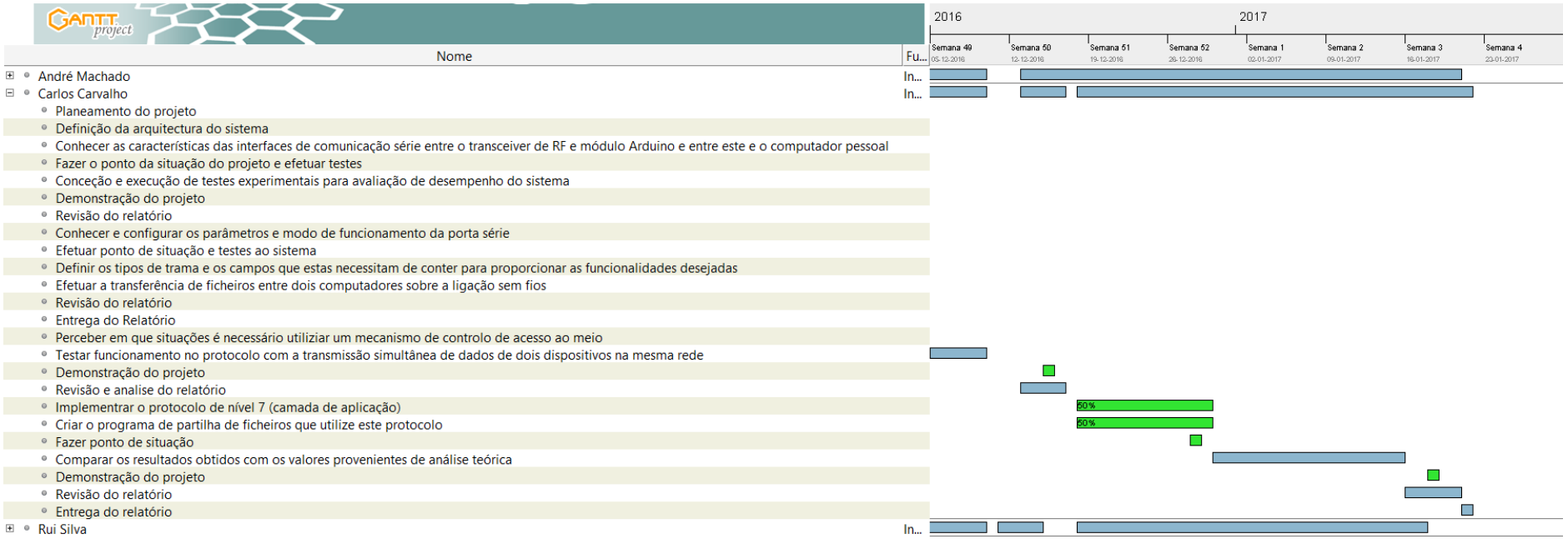


Figura 17 - Diagrama de tarefas do elemento Carlos Carvalho, parte 3.

3.2.2.4. Diagrama de tarefas do elemento Rui Silva

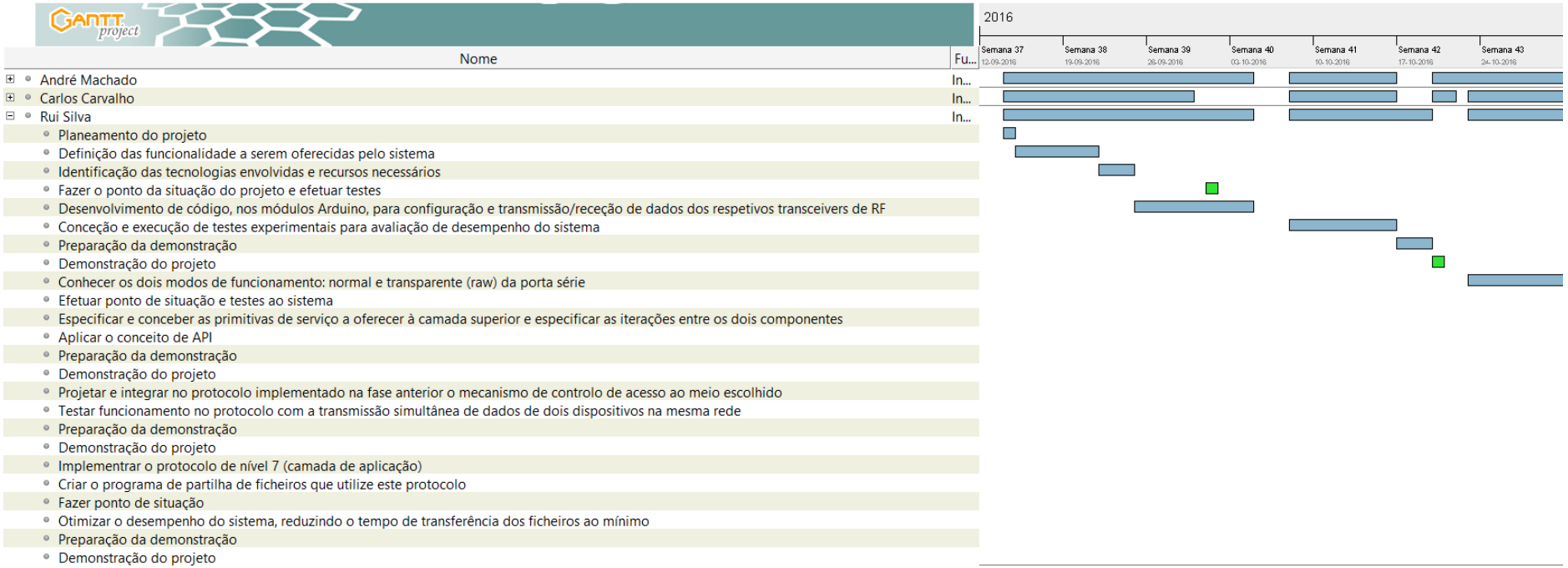


Figura 18 - Diagrama de tarefas do elemento Rui Silva, parte 1.

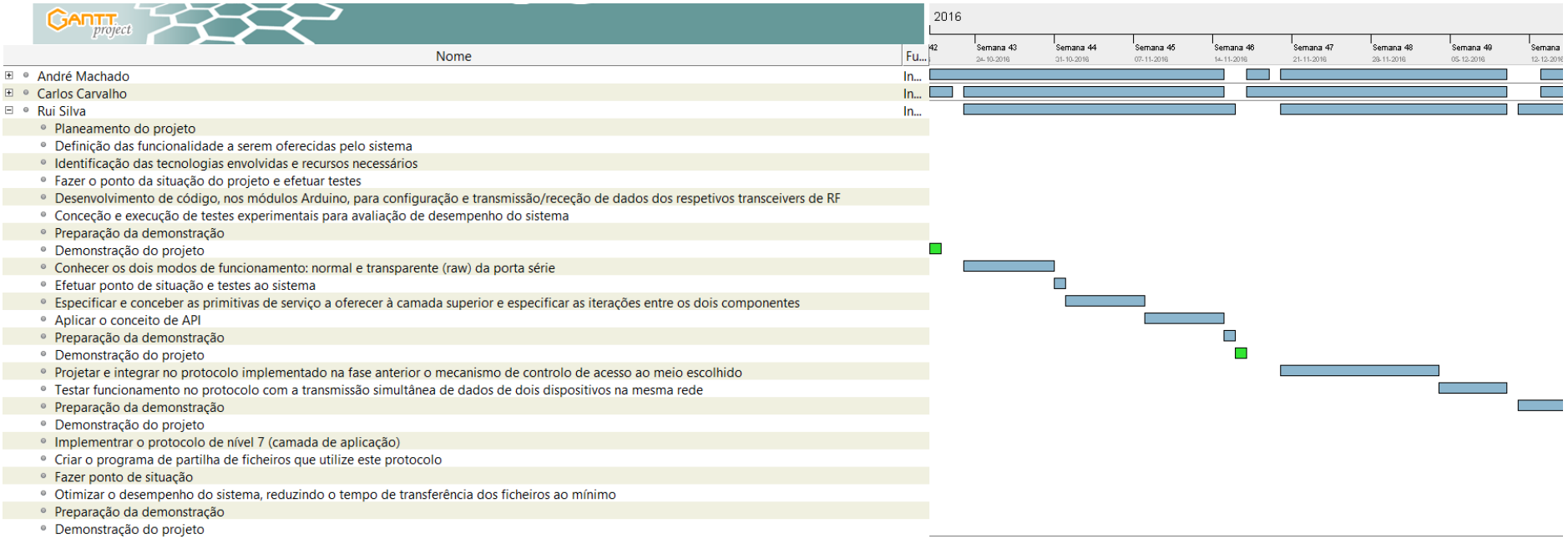


Figura 19 - Diagrama de tarefas do elemento Rui Silva, parte 2.

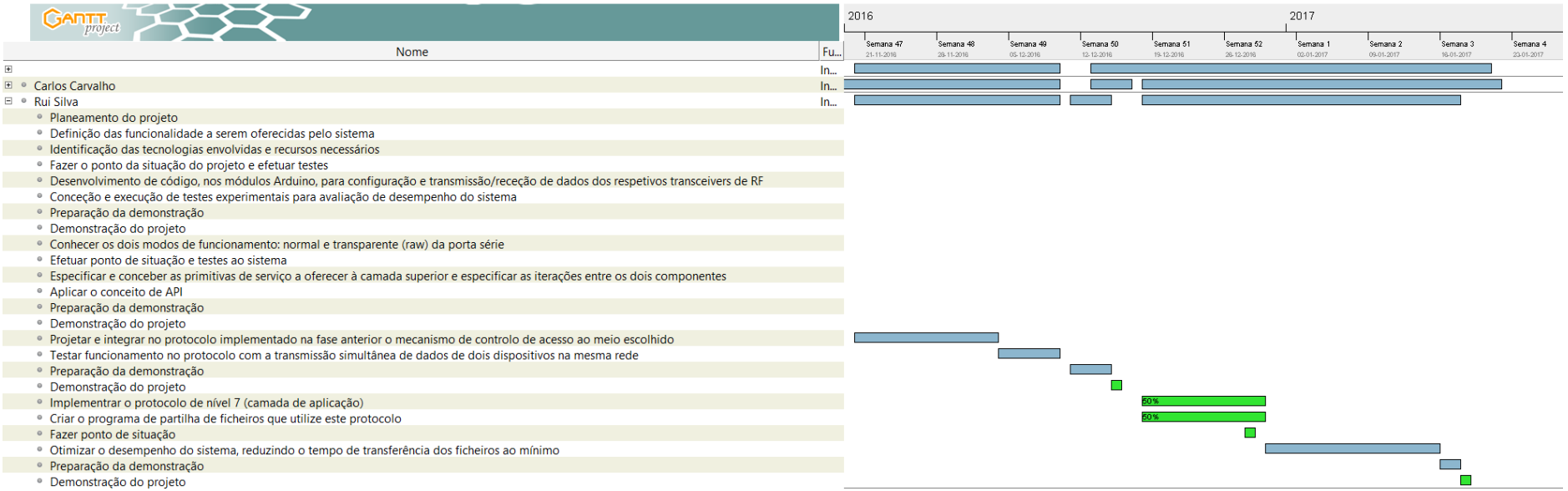


Figura 20 - Diagrama de tarefas do elemento Rui Silva, parte 3.

3.3. Riscos

Nesta secção pretende-se analisar todos os riscos que podem surgir ao longo deste projeto, assim como as medidas que devem ser tomadas para a sua prevenção.

Os principais riscos abordados serão:

- **Má gestão de tempo:** Devido à dimensão do projeto existe um grande risco de haver uma má gestão do tempo. Para tal, o grupo elaborou várias planificações das diferentes fases, de acordo com as datas de entrega, dividindo tarefas pelos respetivos elementos. Foi também decidido que deve haver várias reuniões semanais para desenvolvimento do trabalho;

- **Falta de conhecimentos por parte dos elementos do grupo**: Como se vai lidar com hardware e software nunca antes utilizado pelos elementos, há um risco de não ser possível cumprir os objetivos pedidos. Para evitar o risco foi decidido recorrer à ajuda dos diferentes docentes da cadeira e ao uso de diversos recursos como a internet, livros e outro material;

- **Falta de comunicação entre os elementos do grupo:** Este projeto exige muita comunicação entre os membros do grupo. Deste modo, para além das reuniões foi decidido criar grupos e conversas específicas para debater determinados assuntos relacionados com todas as tarefas e dificuldades;

- **Má interpretação do projeto:** Para entregar todos os objetivos é necessária uma boa compreensão do enunciado. Além do comparecimento de todos os membros à aula onde foi elaborada a explicação do projeto, recorreu-se ao uso das aulas desta unidade curricular para esclarecer todas essas dúvidas;

- **Sobrecarga horária:** O risco mais comum devido ao elevado número de projetos que os membros do grupo têm de realizar neste semestre, o que conduz a um elevado número de horas extracurriculares individuais. Para minimizar este risco é necessária uma cautelosa planificação das tarefas a serem realizadas assim como um supervisionamento do projeto.

3.4. Infraestruturas

Ao longo deste projeto ocorrerá a necessidade de usar algumas infraestruturas, para diversas reuniões e desenvolvimento do mesmo. Esta gestão será feita consoante o horário e disponibilidade de cada elemento.

Serão usados os seguintes locais:

- **LAP 3, Escola de Engenharia, Universidade do Minho no polo de Azurém –** Serão realizadas todas as aulas desta unidade curricular assim como a maior parte das reuniões;

- **Biblioteca da Universidade do Minho, Azurém –** Na qual será realizado reuniões com o objetivo de continuar todas tarefas planeadas para o projeto;

- **Biblioteca da Universidade do Minho, Gualtar –** Devido ao facto de cada elemento ter residências fora do local de trabalho e ter necessidade de reuniões extras, por vezes teremos que usar esta infraestrutura.

3.5. Tecnologias e recursos utilizados

As principais tecnologias a utilizar neste projeto serão o Arduino, o nRF24L01, a linguagem Java e a linguagem C++, na qual se baseia a programação do Arduino.

- **Arduino –** É uma plataforma de eletrónica que facilita a utilização de hardware e software. Uma placa Arduino permite a leitura de inputs de dispositivos periféricos e transforma esse input num output útil ao utilizador. Uma placa Arduino é programada ao mandar um conjunto de instruções para o microcontrolador da placa, através da linguagem de programação Arduino (baseada em C++) e o IDE de programação "Arduino Software" (9) .

- **Java –** O Java é uma linguagem de programação que se distingue por ser uma linguagem orientada a objetos e baseada em classes. Código Java compilado pode correr em qualquer máquina que tenha a "Java virtual machine" sem haver a necessidade de recompilar.

- **C++** - A linguagem C++ é orientada a objetos, imperativa, mas que também permite a possibilidade de manipular níveis mais baixos de memória. Foi desenhada de maneira a oferecer boa performance e eficiência. A linguagem C++ foi criada como uma extensão da linguagem C.

- **nRF24L01+** – O nRF24L01+ é um módulo de Radiofrequência caracterizado pelo seu baixo custo mas boa eficiência. Trabalha na banda de frequências dos 2.4 GHz.

O principal recurso a ser utilizado será o **computador**.

3.6. Software utilizado

Nós ao longo do projeto vamos utilizar diversas ferramentas por isso nesta secção pretendemos falar um pouco de todo software que achamos que vai ser necessário.

- **Arduino (IDE) 1.6.12 –** Este software é o IDE oficial disponibilizado pela Arduino para compilar e testar todo código dirigido aos Arduinos UNO r3 e monitorizar resultados obtidos, através da porta série;
- **Google drive –** Como serão gerados muitos documentos, programas e outros conteúdos que todos elementos do grupo necessitam de ter acesso, decidiu-se por bem recorrer ao uso de uma cloud;
- **Microsoft Office Word –** No final de cada fase é necessário entregar um relatório. Para tal recorrer-se-á ao uso do *Word* visto ser uma ferramenta com muitos recursos com a qual já todos elementos estão ambientados, ao contrário de outros softwares como o "LATEX";
- **Messenger do Facebook –** Neste projeto existe a necessidade de contato frequente com cada elemento do grupo. Apesar de esta ser uma ferramenta de uma rede social, é uma ferramenta que todos elementos do grupo usam constantemente e é muito completa, com suporte para vários sistemas operativos e possibilita o envio de imagens, ficheiros e links web. Ponderou-se o uso do software "Slack", o "Skype", entre outros, mas de todos o mais cómodo.
- **Proteus –** O grupo entendeu que haverá necessidade de desenvolver diversos esquemáticos e efetuar algumas simulações de componentes eletrónicos;
- **Ganttproject -** Usado para elaboração dos diagramas de *Gantt* presentes no relatório referente a cada fase e ao projeto geral.

4. Análise de testes

4.1. Round-trip time

4.1.1. Análise prática do Round-trip Time

Nesta fase inicial foram realizados testes com o intuito de calcular o round-trip, ou seja, o tempo que demora um dado Arduino a enviar um pacote através de um dispositivo nRF24L01 até outro ponto e receber a confirmação de volta. O tamanho desse intervalo vai depender de vários fatores, como por exemplo, o tamanho dos pacotes, a velocidade de transmissão, a distância e o meio de propagação.

Numa primeira fase, foi enviado um pacote um array de 32 caracteres. O emissor envia esse pacote e espera pela confirmação por parte do emissor. A confirmação é composta por um inteiro.

Testou-se o envio desse pacote para variadas velocidades, nomeadamente 250Kbps, 1Mbps e 2Mbps. A distância foi cerca de 20 centímetros. Com isto tenciona-se encontrar que modo o *round-trip* varia para diferentes velocidades.

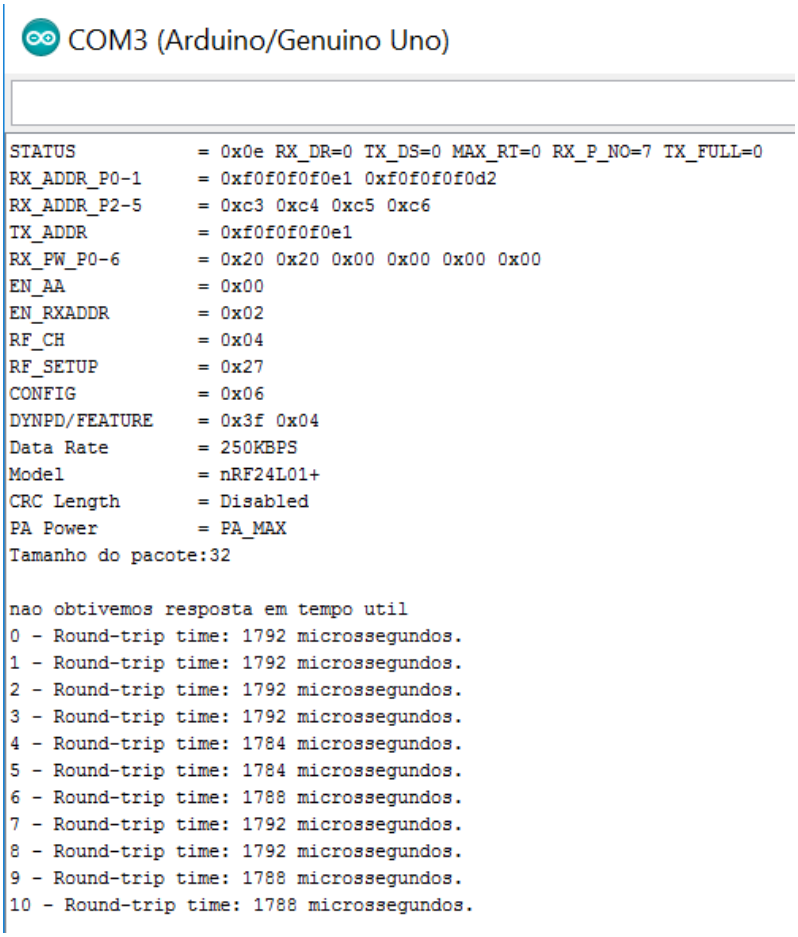


Figura 21 - Round-trip Time para um Data Rate de 250 Kbps, para um pacote de Tamanho 32.

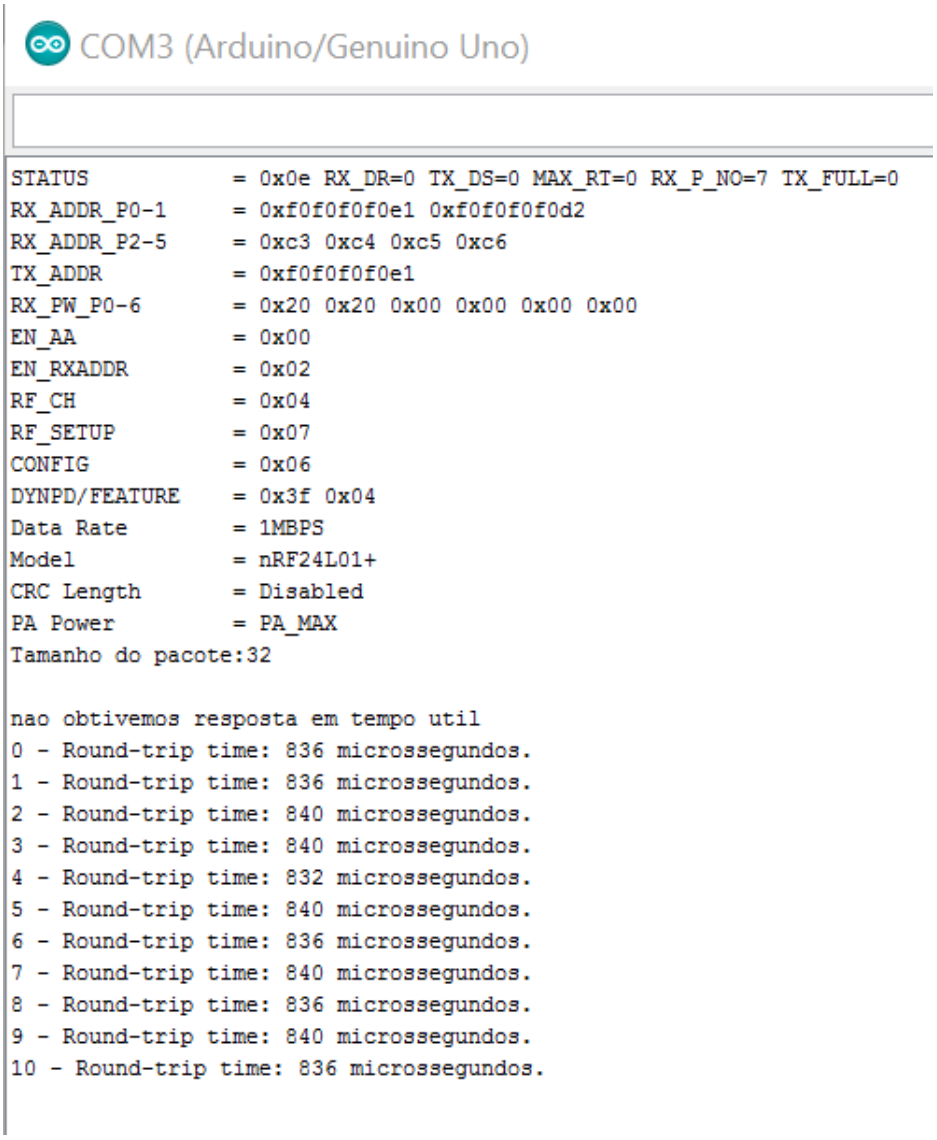


Figura 22 - Round-trip Time para um Data Rate de 1 Mbps, para um pacote de Tamanho 32.

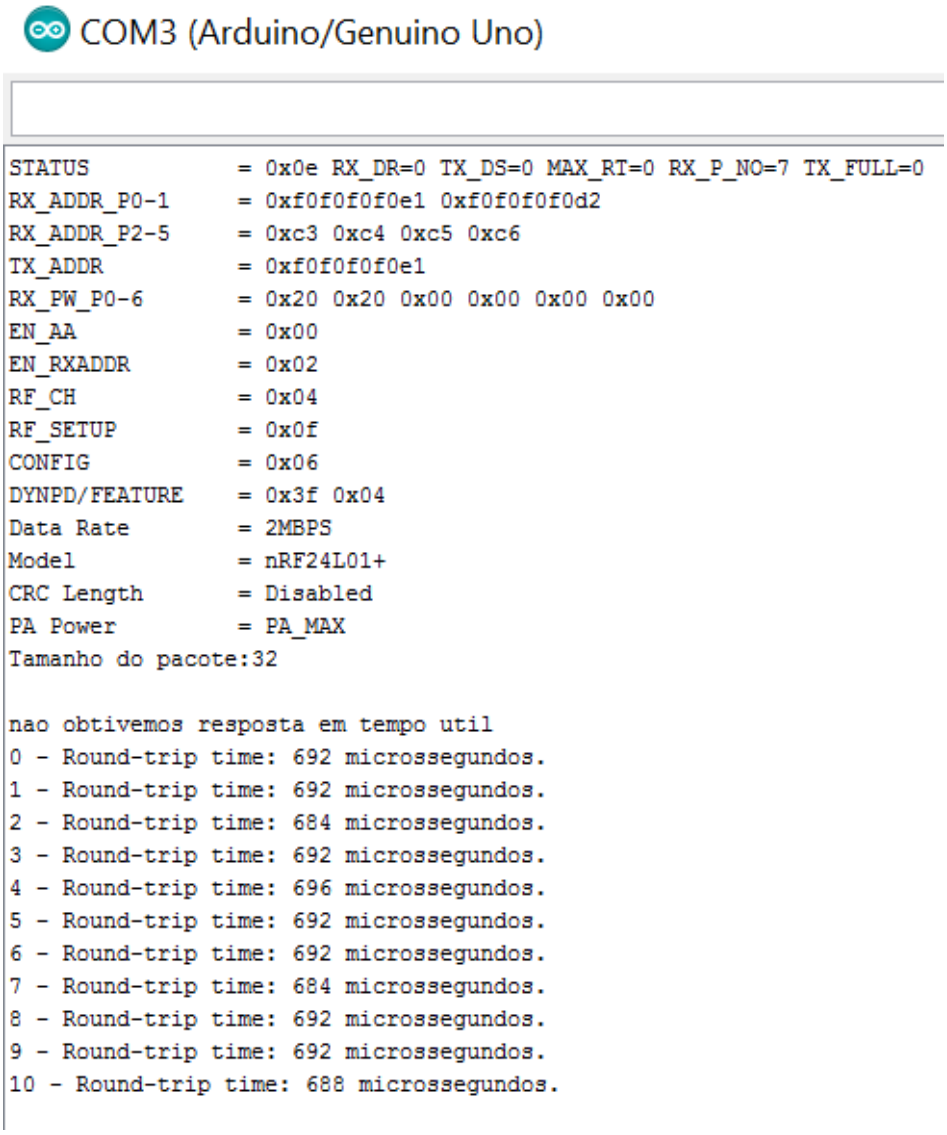


Figura 23 - Round-trip Time para um Data Rate de 2 Mbps, para um pacote de Tamanho 32.

Além do envio de o mesmo pacote com diferentes velocidades, testou-se, para a mesma velocidade, o envio de pacotes diferentes. O pacote consistia num array de char, num caso com 32 bytes e noutro com apenas 1.

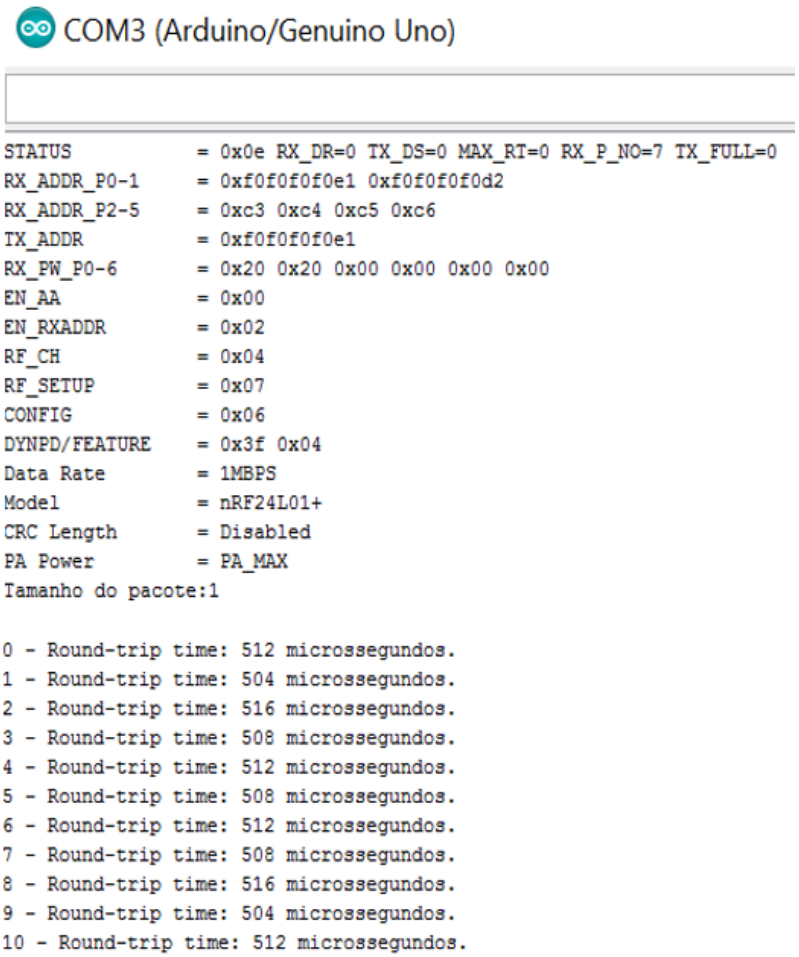


Figura 24 - Round-trip Time para um Data Rate de 1 Mbps, para um pacote de Tamanho 1.

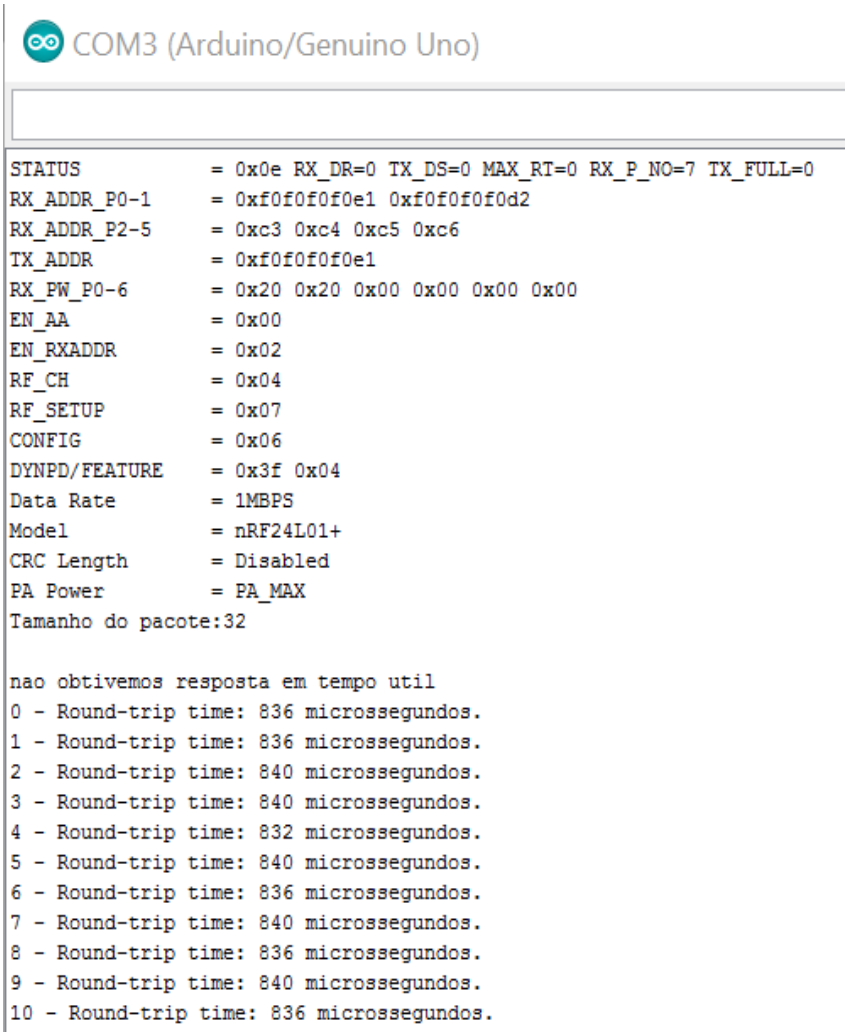


Figura 25 - Round-trip Time para um Data Rate de 1 Mbps, para um pacote de Tamanho 32.

Com base nos diferentes resultados obtidos nesta primeira fase de testes, para o envio do mesmo pacote, o tempo de round-trip vai ser tanto menor quanto maior for o *bit rate*. Para a mesma velocidade, nas tentativas onde os pacotes eram compostos por mais bytes verificou-se tempos maiores.

Pode-se então concluir que, dentro dos fatores que afetam o round-trip time, o tempo que demora o Arduino a processar a informação e o tempo de propagação no meio têm reduzido impacto em relação ao tempo de transmissão da informação.

4.1.2. Análise teórica do Round-trip time

Após fazer a análise dos resultados práticos, irá ser realizado um cálculo teórico para os tempos de round-trip. Para depois poder ser feita uma melhor analogia entre os valores práticos e teóricos. Vai-se analisar o problema usando valores que foram usados nos testes. Sendo:

- Arduíno emissor como o emissor do pacote e o que recebe a confirmação;
- Arduíno recetor como o recetor do pacote e o que envia a confirmação;

- A trama 1 consistindo num pacote com: 32 bytes de *payload*, 5 bytes de *Adress* e 1 byte de *preamble*, num total de 38 bytes;
- A trama 2 (confirmação) consistindo num pacote com: 1 byte de *payload*, 5 bytes de *Adress* e 1 byte de *preamble*, num total de 7 bytes;
- SPI_rate = 10 Mbps;
- Data_rate_on_air = 1 Mbps.

Considerando as várias etapas ao longo da comunicação, o tempo de round-trip teórico é dado pela seguinte expressão:

$t_{round\text{-}trip\_time} = t_{processamento1} + t_{transmissão1} + 2\ p_{propagação} + t_{transmissão2} + t_{processamento2}$

Para a trama 1:

$t_{transmissão1} = t_{transmissão\_spi1\_1} + t_{transmissão\_ar1\_1} + t_{transmissão\_ar2\_1} + t_{transmissão\_spi2\_1};$

$t_{transmissão1} = \frac{(38*8)\ bits}{10\ Mbps} + \frac{(38*8)\ bits}{1\ Mbps} + \frac{(38*8)\ bits}{1\ Mbps} + \frac{(38*8)\ bits}{10\ Mbps} = 668{,}8\ \mu s.$

Onde:

- $t_{transmissão\_spi1\_2}$, tempo transmissão trama 1 do Arduíno emissor para o rf24 emissor:
- $t_{transmissão\_ar1\_2}$, tempo transmissão trama 1 do rf24 emissor para o ar:
- $t_{transmissão\_ar1\_2}$, tempo transmissão trama 1 do ar para o rf24 recetor;
- $t_{transmissão\_spi1\_2}$, tempo transmissão trama 1 do rf24 recetor para o Arduíno recetor.

Para a trama 2:

$t_{transmissão2} = t_{transmissão\_spi2\_2} + t_{transmissão\_ar2\_2} + t_{transmissão\_ar2\_1} + t_{transmissão\_spi2\_1};$

$t_{transmissão2} = \frac{(7*8)\ bits}{1\ Mbps} + \frac{(7*8)\ bits}{10\ Mbps} + \frac{(7*8)\ bits}{10\ Mbps} + \frac{(7*8)\ bits}{1\ Mbps} = 123{,}2\ \mu s.$

Onde:

- $t_{transmissão\_spi2\_2}$, tempo transmissão trama 2 do Arduíno recetor para o rf24 recetor;
- $t_{transmissão\_ar2\_2}$, tempo transmissão trama 2 do rf24 recetor para o ar.
- $t_{transmissão\_ar2\_1}$, tempo transmissão trama 2 do ar para o rf24 emissor;
- $t_{transmissão\_spi2\_1}$, tempo transmissão trama 2 do rf24 emissor para o Arduíno emissor.

Como a velocidade de propagação da onda é muito elevada comparativamente à distância que esta percorre, podemos considerar o tempo de propagação = 0, obtendo:

$t_{round\text{-}trip\_time} = t_{processamento1} + t_{transmissão1} + t_{transmissão2} + t_{processamento2}$

$t_{round\text{-}trip\_time} = 792\ \mu s + t_{processamento1} + t_{processamento2}.$

Comparando os resultados experimentais obtidos onde o $t_{round\text{-}trip\_time}$ = 840 µs, podemos concluir que o $t_{processamento}$ dos Arduíno é aproximadamente 58 µs. Como o tempo teórico e prático estão bastante próximos, podemos concluir que durantes os testes não houveram interferências com grande relevo nos resultados.

Analisando o mesmo problema, porém considerando apenas os bytes do *payload* das tramas e desprezando o *overhead* introduzido pelo envio a trama, obtemos que:

$t_{transmissão1}$ = 563,3 µs;

$t_{transmissão2}$ = 17,6 µs;

e consequentemente,

$t_{round\text{-}trip\_time}$ = 580,8 µs + $t_{processamento1}$ + $t_{processamento2}$.

Considerando o $t_{processamento1}$ + $t_{processamento2}$ ≈ 58 µs

Logo, $t_{round\text{-}trip\_time}$ = 638,8 µs

Podemos concluir que, o tempo extra necessário para processar e enviar o *overhead* é cerca de 153,2µs.

A percentagem útil da transmissão é:

$$U = \frac{638{,}8\ \mu s}{840\ \mu s} \approx 76.0\%$$

A taxa efetiva da transmissão é de:

$$Taxa_{efetiva} = \frac{Dados\ úteis}{Tempo\ total} = \frac{(32*8)\ bits}{840\ \mu s} \approx 304{,}8\ Kbps.$$

4.2. *Throughtput*

Em comunicações, o *throughput*, ou taxa de transferência, é a quantidade de dados transferidos com sucesso do emissor para o recetor por unidade de tempo. O *throughput* é medido normalmente em bits por segundo (bps).

4.2.1. Análise Prática do *Throughput*

Para fazer testes ao *Throughput* foi adotada a seguinte metodologia:

O emissor estará a enviar pacotes de 32 bytes ao recetor. Quando este recebe 10.001 pacotes, este envia um ACK de volta ao emissor. Depois de o recetor receber o ACK este vai calcular o tempo que demorou a enviar esses mesmo pacotes e ver qual foi a taxa efetiva de transmissão. O número de pacotes a enviar é elevado de forma a diluir o tempo de transmissão do ACK no tempo de envio dos pacotes e assim poder ser considerado nulo.

O emissor e o recetor estarão posicionados o mais próximo possível um do outro de forma a que o tempo de propagação e as perdas devido às interferências sejam o mais reduzidas possível ou até mesmo nulas. A velocidade do SPI é cerca de 10 Mbps e a taxa de dados no RF24 será de 250 Kbps, 1 Mbps e 2 Mbps.

Após correr o programa, foram obtidos os seguintes resultados:

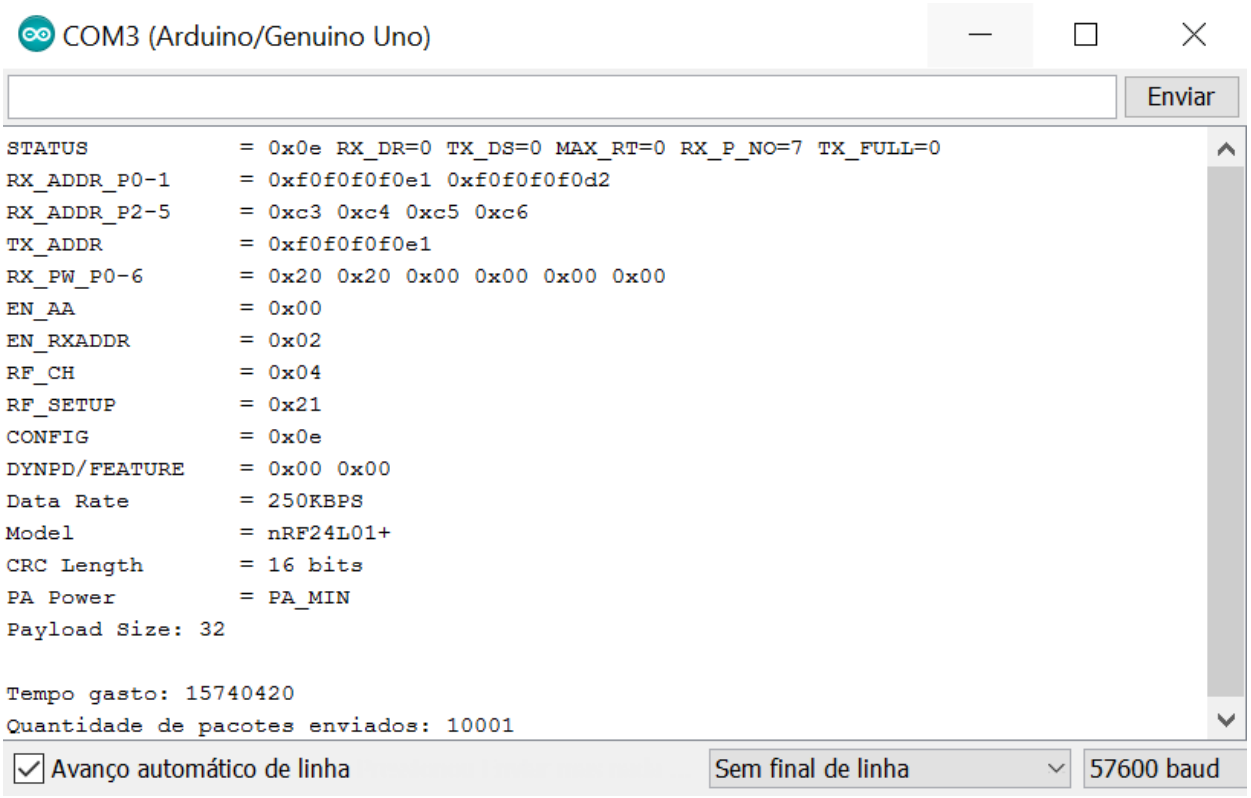


Figura 26 - Throughput para um Data Rate de 250 KBPS.

$$\text{Throughput} = \frac{10001*32*8}{15{,}740420} = 162{,}65 \text{ Kbps}$$

$$U = \frac{t_{\text{transmissão } efeitva}}{t_{\text{transmissão esperada}}} = \frac{162{,}65 \text{ k}}{250 \text{ k}} \approx 65.06\%$$

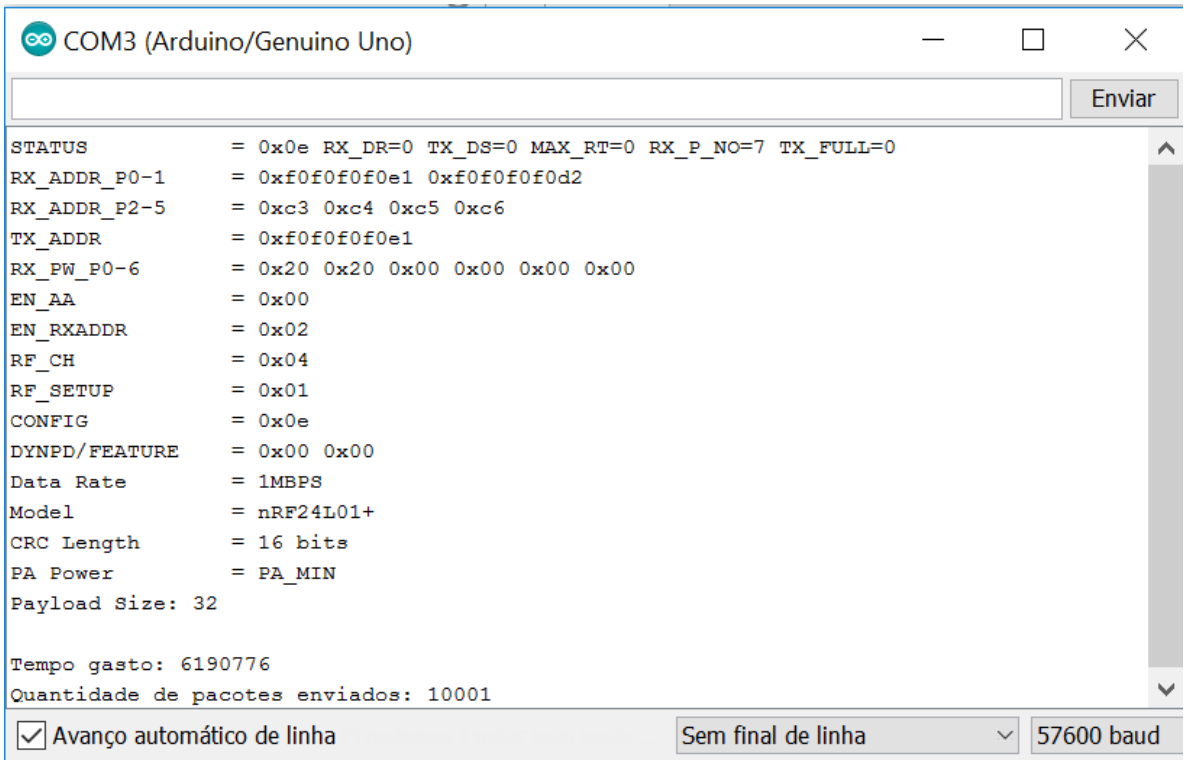


Figura 27 - Throughput para um Data Rate de 1 Mbps.

$$\text{Throughput} = \frac{10001*32*8}{6{,}190776} = 413{,}56 \text{ Kbps.}$$

$$U = \frac{t_{\text{transmissão } efeitva}}{t_{\text{transmissão esperada}}} = \frac{0{,}41356 \text{ M}}{1 \text{ M}} \approx 41{,}36\%$$

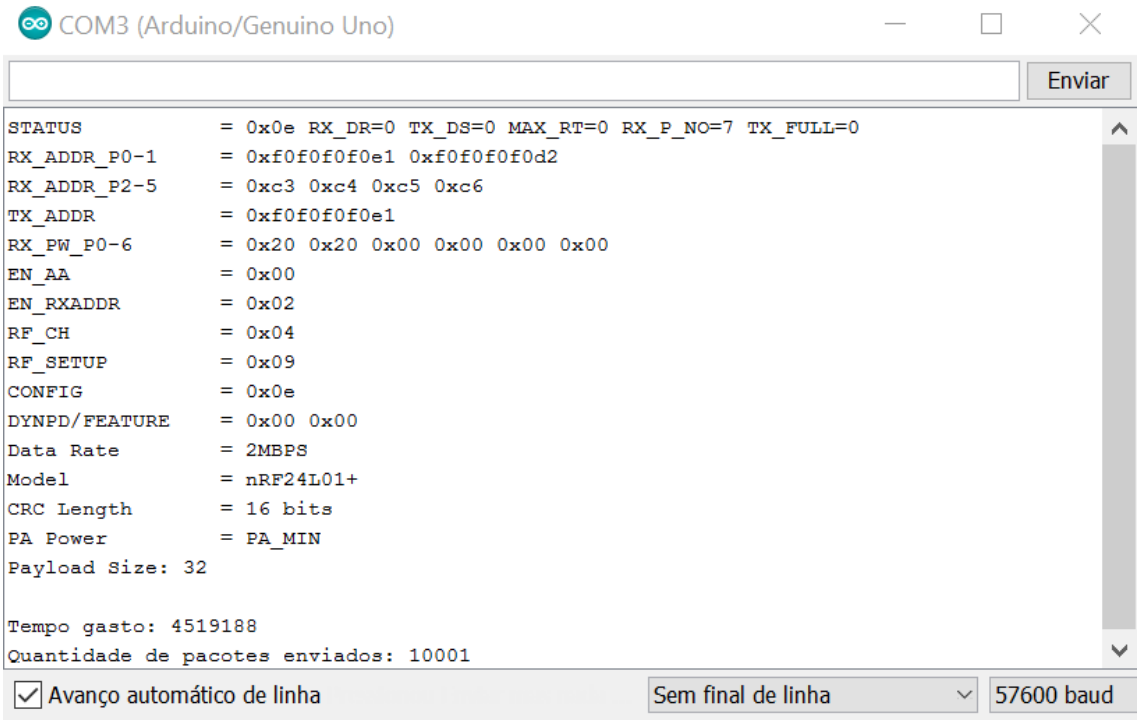


Figura 28 - Throughput para um Data Rate de 2Mbps.

$$\text{Throughput} = \frac{10001*32*8}{4{,}519188} = 566{,}5 \text{ Kbps}$$

$$U = \frac{t_{\text{transmissão } efeitva}}{t_{\text{transmissão esperada}}} = \frac{566{,}5 \text{ k}}{2 \text{ M}} \approx 28{,}33\%$$

4.2.2. Análise Teórica do *Throughput*

Para fazer a análise teórica do *throughput*, obtidos com os dados teóricos pode-se concluir que o tempo de processamento tem com 32 bytes de *payload* e cerca de 8 bytes de *overhead* (1 introdução / *preamble*, 5 bytes de endereçamento e 2 de CRC), num total de 40 bytes.

Será feita a análise considerando o *overhead* e posteriormente desprezando-o para assim comparar a diferença nos tempos finais e o respetivo impacto.

O tempo total de transmissão é dado por:

$t_{transmissão} = N \times (t_{transmissão\_spi} + t_{transmissão\_ar})$,

onde:

- $t_{transmissão\_spi}$, é o tempo de transmissão das tramas do Arduino para o RF24;
- $t_{transmissão\_ar}$, é o tempo de transmissão das tramas do Arduino para o ar.
- N é o numero de pacotes enviados. Tal como nos testes práticos, N toma o valor de 10001.

- Para data rate de 2 Mbps no rf24 obtêm-se o seguinte resultado:

Considerando o *overhead*:

$t_{transmissão}$ = 10001 x ($\frac{(40*8)\ bits}{10\ Mbps} + \frac{(40*8)\ bits}{2\ Mbps}$) = 1,920 s.

Desprezando o *overhead*:

$t_{transmissão}$ = 10000 x ($\frac{(32*8)\ bits}{10\ Mbps} + \frac{(32*8)\ bits}{2\ Mbps}$) = 1,536 s.

$U = \frac{t_{transmissão\ útil}}{t_{transmissão\ total}} = \frac{1,536}{1,920} \approx 80.0\%$

$Taxa_{efetiva} = \frac{Dados\ úteis}{Tempo\ Total} = \frac{10001*32*8}{1,920} = 1,333\ Mbps$

- Para data rate de 1Mbps no rf24 obtêm-se o seguinte resultado:

Considerando *overhead*:

$t_{transmissão}$ = 10001 x ($\frac{(40*8)\ bits}{10\ Mbps} + \frac{(40*8)\ bits}{1\ Mbps}$) = 3,520 s

Desprezando o *overhead*:

$t_{transmissão}$ = 10001 x ($\frac{(32*8)bits}{10\ Mbps} + \frac{(32*8)bits}{1\ Mbps}$) = 2,816 s.

$U = \frac{t_{transmissão\ útil}}{t_{transmissão\ total}} = \frac{2,816}{3,520} \approx 80\%$

$Taxa_{efetiva} = \frac{Dados\ úteis}{Tempo\ Total} = \frac{10001*32*8}{3,520} = 727,27\ Kbps$

- Para data rate de 250Kbps no rf24 obtêm-se o seguinte resultado:

Considerando o *overhead*:

$t_{transmissão}$ = 10001 x ($\frac{(40*8)\ bits}{10\ Mbps} + \frac{(40*8)\ bits}{250\ Kbps}$) = 13,121 s.

Desprezando o *overhead*:

$t_{transmissão}$ = 10001 x ($\frac{(32*8)\ bits}{10\ Mbps} + \frac{(32*8)\ bits}{250\ Kbps}$) = 10,497 s.

$U = \frac{t_{transmissão\ útil}}{t_{transmissão\ total}} = \frac{10,497}{13,121} \approx 80.0\%$

$$\text{Taxa}_{\text{efetiva}} = \frac{Dados\ úteis}{Tempo\ Total} = \frac{10001*32*8}{13{,}121} = 195{,}1\ Kbps$$

Comparando os dados práticos obtidos com os dados teóricos podemos concluir que o tempo de processamento tem um grande impacto no tempo total, especialmente para velocidades mais reduzidas. Se fizermos:

$T_{total} = t_{transmissão} + t_{processamento}$, se se considerar que os $t_{transmissão}$ como sendo o valor teórico obtido, para a comunicação a 1Mbps, obtemos que:

$t_{processamento} = 6{,}19 – 3{,}520 = 2{,}67$ s.

4.3. Perdas

Para calcular as perdas do sistema procedeu-se ao envio de um número elevado de tramas do transmissor para o recetor sem que este último enviasse um ACK.

Cada trama é constituída apenas pelo seu número de série. O recetor quando recebe uma trama guarda o seu número de série e depois incrementa o número total de tramas que recebeu até ao momento. Se o número de série não corresponder ao número de tramas recebidas até ao momento então é atualizado o número de perdas. Finda a transmissão é apresentado o número de pacotes recebidos, para calcular as perdas subtrai-se esse total de recebidos ao número fixo de tramas que foram transmitidas. Outra maneira de calcular era apresentar o contador de tramas perdidas, o que era menos fiável devido aos erros introduzidos nos pacotes que iriam corromper o número de série.

Constatou-se que à medida que a distância aumentava o número de perdas era maior, o que vai de acordo com o que era esperado. Ao longo de uma maior distância a onda transmissora irá ser atenuada, sofrer colisões e será suscetível a uma maior influência por parte do ruído. Todos os fatores mencionados anteriormente contribuem para o aumento das perdas.

Verificou-se também nos testes que durante os momentos em que existiam obstruções no meio entre os transceivers as perdas aumentavam, assim como a corrupção dos dados a serem enviados. Estas obstruções foram principalmente mediadas por pessoas que caminhavam entre os transceivers ou pela existência de paredes entre estes.

```
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
nr de pacotes recebidos:835
```

Figura 29 - Perdas a uma distância de 90 metros, parte 1.

```
964
965
969
972
975
980
982
983
984
985
986
987
989
991
992
994
995
997
999
1000
1002
nr de pacotes recebidos:687
```

Figura 30 - Perdas a uma distância de 90 metros, parte 2.

```
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
nr de pacotes recebidos:994
```

Figura 31 - Perdas a uma distância de 40 metros.

```
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
nr de pacotes recebidos:984
```

Figura 32 - Perdas para uma distância de 10 metros.

```
974
975
976
979
980
981
984
985
986
989
990
991
993
994
995
996
999
1001
nr de pacotes recebidos:705
```

Figura 33 - Perdas para uma distância de 30 metros.

4.4. Atraso

O Atraso é definido como o tempo entre o momento em que o transmissor envia o primeiro bit de um pacote e entre o momento em que o recetor recebe o último bit desse pacote.

As variantes que vão influenciar o Atraso serão a velocidade de transmissão, o tamanho do pacote a ser enviado, o tempo de processamento e o tempo que demora a enviar, do Arduino até ao módulo RF, a trama. O tempo de propagação dos sinais serão ignorados uma vez que os testes foram efetuados a distâncias mínimas.

4.4.1. Análise prática do Atraso

Para fazer a medição do Atraso procedeu-se à transmissão de um pacote de tamanho total de 8 bytes.

Os tempos foram gravados através do uso das interrupções do Arduino. No instante em que a transmissão começava o transmissor punha a *High* um pino que se encontrava ligado ao segundo Arduino. Com uma interrupção associada à transição de *Low* para *High* no pino referido, o Arduino recetor entrava numa ISR (*Interrupt Service Routine*) que ia anotar o instante em que recebeu essa interrupção, que pouco difere do momento inicial da transmissão. Tendo registado o tempo inicial da transmissão o recetor esperava que esta acabasse e anotava esse instante. Com o tempo inicial e final de transmissão foi então possível proceder ao cálculo do Atraso.

Sendo o tempo inicial definido como t_i , o tempo final como t_f e o tempo de atraso como t_a

então:

$$\mathbf{t_a = t_f - t_i}$$

Foram efetuadas medições do tempo de atraso recorrendo a diferentes taxas de transmissão. 250 Kbps, 1 Mbps e 2 Mbps.

Para obter uma boa estimativa do atraso procedeu-se ao cálculo da média de 20 transmissões consecutivas para cada uma das velocidades mencionadas no parágrafo anterior.

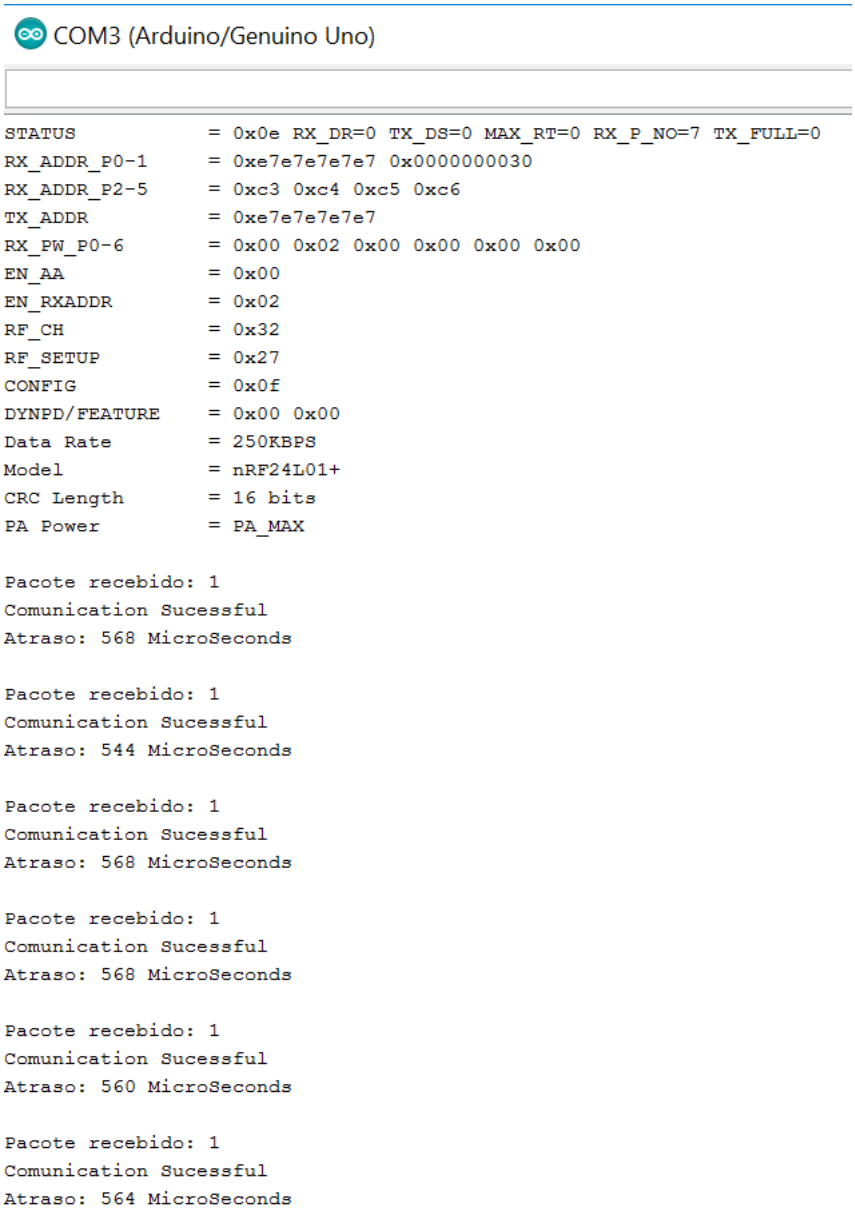


Figura 34 - Atraso para um Data Rate de 250 Kbps.

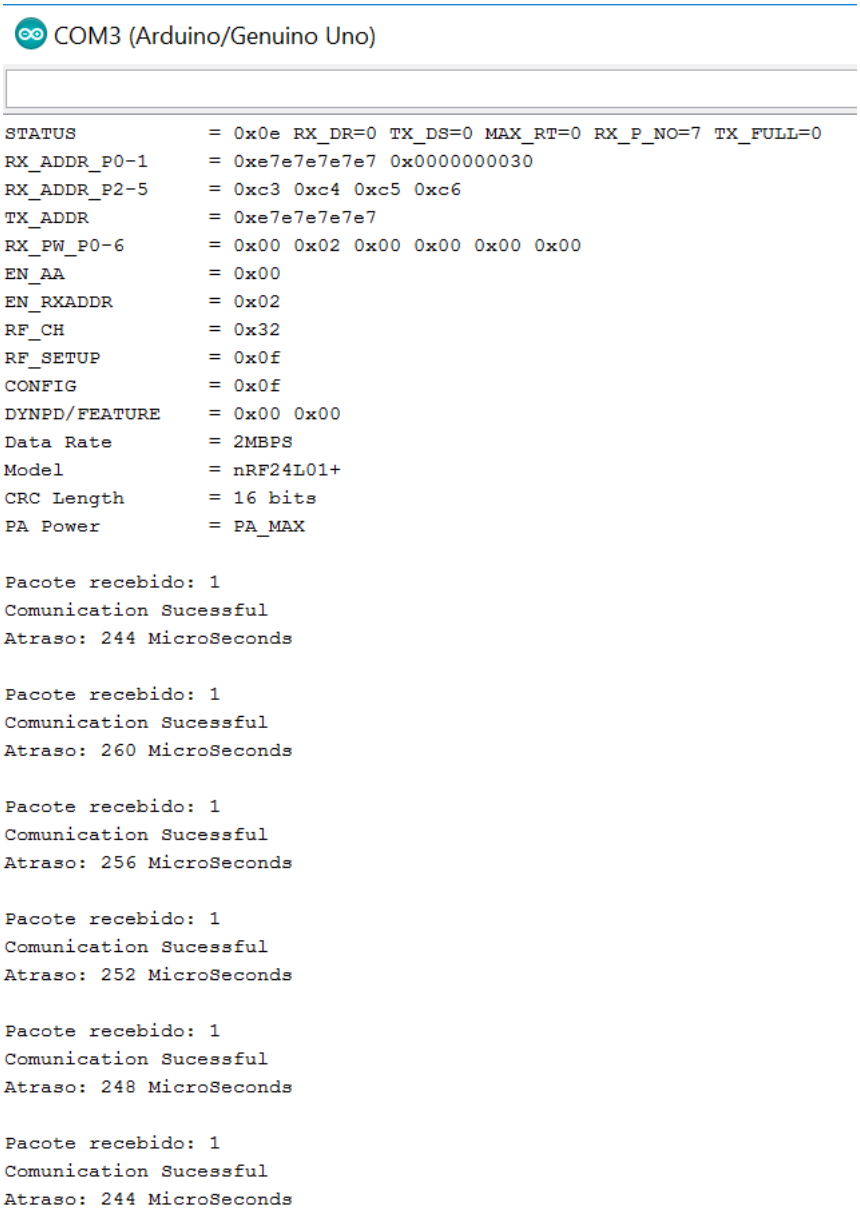


Figura 35 - Atraso para um Data Rate de 2 Mbps.

Figura 36 - Atraso para um Data Rate de 1 Mbps.

4.4.2. Análise teórica do Atraso

Para os calcular os valores teóricos do Atraso irá ser considerado:

- O tempo que demora a transmitir um pacote;
- Pacote terá tamanho de 10 bytes, denominado de L;
- Taxas de transmissão de 250 Kbps, 1 Mbps e 2 Mbps, denominadas de R.
- A soma do tempo de processamento com o tempo de tratamento da interrupção;
- O tempo de comunicação da trama entre o Arduino e os módulos por SPI, com taxa de transmissão de 10Mbps;

$$t_{atraso} = t_{transmissão} + (t_{tratamento} + t_{processamento}) + t_{transmissão\_spi}$$

Sabendo que:

$$\mathbf{t_{transmissão} = L / R}$$

Então com L = 10 bytes e para:

- R = 250 Kbps, tem-se que $\mathbf{t_{transmissão}} = 256$ µs
- R = 1 Mbps, tem-se que $\mathbf{t_{transmissão}} = 80$ µs
- R = 2 Mbps, tem-se que $\mathbf{t_{transmissão}} = 40$ µs
- R = 10 Mbps (para a transmissão SPI), tem-se que $\mathbf{t_{transmissão\_spi}} = \mathbf{8}$ µs

O Tempo de Atraso seria então:

- $\mathbf{t_{atraso\_250kpbs}} = 256$ µs + ($\mathbf{t_{tratamento}}$ + $\mathbf{t_{processamento}}$) + 8 µs
- $\mathbf{t_{atraso\_1Mpbs}} = 80$ µs + ($\mathbf{t_{tratamento}}$ + $\mathbf{t_{processamento}}$) + 8 µs
- $\mathbf{t_{atraso\_2Mpbs}} = 40$ µs + ($\mathbf{t_{tratamento}}$ + $\mathbf{t_{processamento}}$) + 8 µs

Comparando estes valores com os resultados práticos verificar-se-ia que a soma do tempo de tratamento com o tempo de processamento aumentava significativamente com o aumento da velocidade de transmissão, o que não é verdade. Tanto o tempo de transmissão de dados entre o Arduino e o módulo RF como o tempo de processamento manter-se-iam praticamente invariáveis entre a mudança das taxas de transmissão.

Tendo em conta valores de processamento obtidos anteriormente, com códigos mais extensos, serem menores que o 100µs conclui-se então que no momento em que os testes práticos apresentados foram efetuados ocorreram interferências de grande magnitude. Pode-se também inferir que os *bitrates* especificados são na realidade abaixo.

Pode-se verificar que o Tempo de Propagação é insignificante ao constatar que:

- Para uma distância de 10 cm;
- Velocidade de Onda de aproximadamente 2.9 * 10 ^ 8

$$\mathbf{T_{propagação}} = 0.01 / (2.9 * 10 \wedge 8) = 3{,}55 * 10 \wedge -11$$

Conclui-se então que o Tempo de Propagação não irá ser relevante. Assim como o tempo de transmissão do sinal da interrupção.

O Atraso é definido como o tempo entre o momento em que o transmissor envia o primeiro bit de um pacote e entre o momento em que o recetor recebe o último bit desse pacote.

As variantes que vão influenciar o Atraso serão a velocidade de transmissão, o tamanho do pacote a ser enviado, o tempo de processamento e o tempo que demora a enviar, do Arduino até ao módulo RF, a trama. O tempo de propagação dos sinais serão ignorados uma vez que os testes foram efetuados a distâncias mínimas.

4.4.3. Comunicação com recurso a ligação direta entre os módulos Arduino utilizando a mesma interface série

Para efetuar a comunicação direta entre os dois módulos Arduino recorreu-se ao uso do protocolo SPI (*Serial Peripheral Interface*). Este protocolo usa o modo *Full-Duplex*, onde é implementada a arquitetura "*Master-Slave*", onde um dos periféricos é definido como "Master" e outro como "*Slave*".

Para haver comunicação são necessários 4 sinais, que são:

- SCLK (Serial *Clock*) – que é o sinal de saída para o Master, correspondendo ao pino 13;
- MOSI (*Master Output Slave Input*) – que é sinal de entrada do *Slave*, correspondendo ao pino 11;
- MISO (*Master Input Slave Output*) – que é o sinal de saída do *Slave*, correspondendo ao pino 12;
- SS (*Slave Select*) – que é o sinal que seleciona o *Slave*, correspondendo ao pino 10;

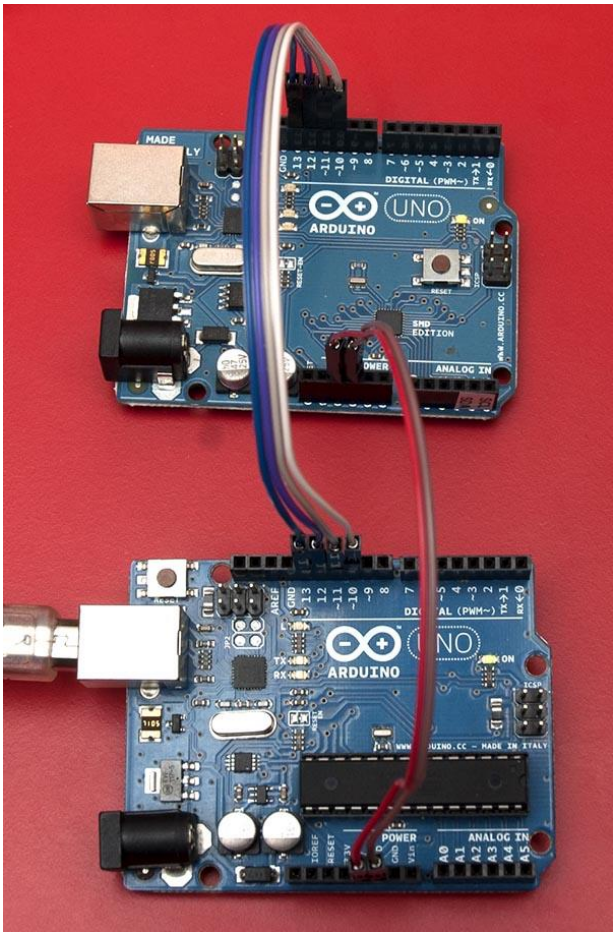


*Figura 37 - Montagem para efetuar ligação direta entre os módulos (*1).*

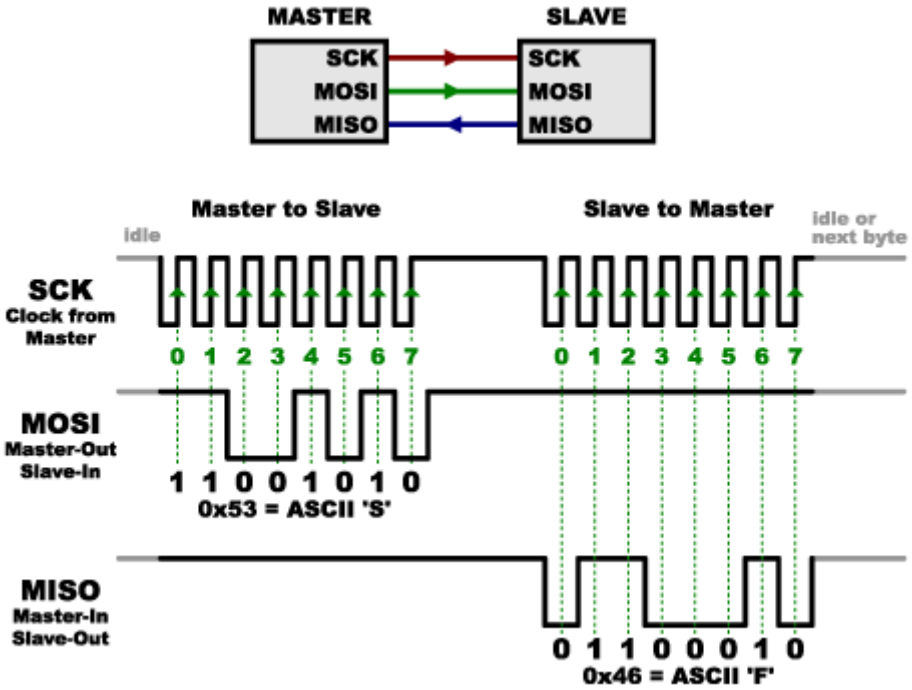


Figura 38 - Receção de dados em SPI.

4.4.4. Análise prática do Round-trip time por ligação direta entre os módulos

Para calcular o Round-Trip Time, criamos um programa onde enviamos um Byte no programa *Master*. No programa *Slave* ocorre uma interrupção onde adicionamos o valor 10 ao que recebeu e enviamos de volta esse valor.

Assim sendo os valores apresentados do Round-Trip Time foram os seguintes:

```
Resultado: 11
RTT: 24 microssegundos
Resultado: 12
RTT: 20 microssegundos
Resultado: 13
RTT: 24 microssegundos
Resultado: 14
RTT: 24 microssegundos
Resultado: 15
RTT: 24 microssegundos
Resultado: 16
RTT: 24 microssegundos
Resultado: 17
RTT: 20 microssegundos
Resultado: 18
RTT: 24 microssegundos
Resultado: 19
RTT: 24 microssegundos
Resultado: 20
RTT: 24 microssegundos
```

Figura 39 - Resultados comunicação com recurso a ligação direta entre os módulos Arduino.

4.4.5. Análise Teórica do Round-trip time por ligação direta entre os módulos

Considerando as várias etapas ao longo da comunicação, o tempo de round-trip teórico é dado pela seguinte expressão:

$t_{round\text{-}trip\_time} = t_{processamento1} + t_{transmissão1} + 2\ p_{ropagação} + t_{transmissão2} + t_{processamento2}$

Como o tempo de propagação é desprezável em relação ao restantes tempo, obteu-se que:

$t_{round\text{-}trip\_time} = 2 * (\frac{1*8}{10^7}) + t_{processamento1} + t_{processamento2}$;

$t_{round\text{-}trip\_time} = 1{,}6\mu s + t_{processamento1} + t_{processamento2}$;

Podemos agora então concluir que o tempo de processamento é cerca de 22µs, tendo uma grande interferência para o resultado final. Comparando o RTT por SPI ou por RF, reparamos que, no primeiro caso, o $t_{processamento}$ tem um impacto maior no resultado final do que na troca por rádio frequência.

5. Conclusão

Nesta primeira fase do projeto os objetivos eram efetuar a planificação do projeto, estudar a camada OSI e fazer com que os módulos RF comunicassem entre si, efetuando uma série de testes para avaliar essa comunicação.

É importante mencionar que alguns dos testes efetuados não foram feitos da maneira inicialmente, mas que o grupo trabalhou para perceber como melhor os realizar embora não a tempo da entrega do presente relatório.

Para este momento de avaliação o grupo reuniu-se frequentemente, mas irá necessitar de mais reuniões nos momentos seguintes para poder responder ao escalamento do trabalho necessário efetuar.

Para terminar é importante mencionar a importância desta UC no desenvolvimento da capacidade de aplicação de conhecimentos adquiridos no curso até ao momento. Os alunos são desafiados a entrelaçar conhecimentos variados que irão culminar num projeto funcional e útil, utilizando e desenvolvendo as suas competências críticas dos conteúdos que são aprendidos. Tal desenvolvimento é um ponto fulcral da formação dos alunos.

6. Autoavaliação

6.1. André Machado

Eu nesta fase estive inicialmente responsável por planificar o projeto e elaborar os diagramas Gantt, assim como dar inicio ao relatório. Trabalhei ainda de forma a conhecer mais sobre os diferentes módulos de modo a fazer as ligações necessária para o seu funcionamento. Elaborei diversos textos para o relatório assim como fui responsável por grande parte da sua formatação e ainda revisão do documento. Ajudei em conjunto com os outros elementos a elaborar código e na sua fase de testes.

6.2. Carlos Carvalho

Nesta fase do projeto estive envolvido principalmente na investigação do modo de funcionamento do Arduino e do RF24, e na elaboração de código para a realização de testes. Fiz a análise de alguns dos resultados obtidos, e ainda contribui para a elaboração do relatório.

6.3. Rui Silva

Nesta fase do projeto trabalhei na elaboração, em conjunto, dos programas utilizados para efetuar testes e comunicação entre os módulos RF. Trabalhei na análise da biblioteca a utilizar para o manuseamento dos mesmos e no planeamento dos recursos e ferramentas a utilizar. Redigi alguns dos textos presentes no relatório assim como editei alguns dos textos dos meus colegas.

7. Referências

(1) http://searchnetworking.techtarget.com/definition/OSI

(2) https://pplware.sapo.pt/tutoriais/networking/redes-sabe-o-que-e-o-modelo-osi/

(3) http://homepage.ufp.pt/lmbg/textos/norma_osi.html

(4)https://www.digi.com/resources/standards-and-technologies/rfmodems/rf-basics

(5)https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

(6)https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

(7)https://www.arduino.cc/en/Guide/Introduction

(8)https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo

(9)https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

(10)http://blog.simtronyx.de/en/briefly-introduced-the-nrf24l01-module-an-inexpensive-and-network-ready-2-4ghz-transceiver-development-board/