

Co-Simulação De Rede Para Internet Das Coisas

Lucas Barros Franson de Castilho

Dissertação apresentada na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança para obtenção do Grau de Mestre em Sistemas de Informação. No âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná

Trabalho realizado sob a orientação de:

Luisa M. G. Jorge

Paulo Melo

Augusto Foronda

Bragança

Outubro de 2018

Dedicatória

Ao meu Pai e minha Mãe.

Agradecimentos

Agradeço ao meu Pai Edson, pelo seu esforço desde criança para ajudar a família, vendendo comida da rua para sustentar a casa, indo para uma cidade grande sofrer, mas trazendo cicatrizes que mostram o quanto é guerreiro, sendo hoje um homem bem sucedido e que me orgulho de chamar de pai.

Minha Mãe Joelma, por sempre ter se esforçado ao máximo para aprender e conseguir enfrentar aquilo que sua carreira colocou a frente, sendo hoje essa mulher valente, que chefia um setor importante na empresa onde trabalha.

Aos meus professores que ajudaram na construção desse trabalho, Luísa Jorge que mesmo quando estava chateado com a tese, sempre estava me auxiliando em conjunto com o professor Paulo Melo. Ao professor Augusto Foronda que mesmo longe, me fez um guerreirinho em terras estrangeiras.

Quero agradecer também aqueles professores que passaram por minha vida acadêmica durante esses anos de faculdade. Sem vocês não seria nada.

Meus amigos que sempre pude contar e foram um sustento nos tempos difíceis que passei aqui, só quem faz um intercâmbio sabe como fica nossa cabeça e sem vocês eu não conseguira.

Um abraço especial para quem esteve comigo durante o tempo de construção da tese:

Murillo Henrique, Adriana Molina, Érika Vega, Jair García, Wesley Bernardo, Silvia Gaia, Maria Elizabeth Claudino, Ana Maria Antunes, Christian Barros e Felipe Franson

Resumo

A construção da rede co-simulada (rede física e rede simulada) dessa tese, será uma base para o desenvolvimento de técnicas de simulação usando a plataforma OMNeT++ (computacional) com dispositivos sensores (físicos). Uma vez estabelecida a plataforma, seria possível estender o funcionamento da rede para outros ambientes, como é o caso do desenvolvimento de redes físicas na mineração. Nesse sentido podemos construir uma rede física, e simular o comportamento de outros componentes e redes utilizando o OMNeT++.

Como esse trabalho utiliza IoT para o desenvolvimento da topologia proposta, esse trabalho acaba corroborando na utilização de aparelhos com baixa potência.

O objetivo desta dissertação é mostrar em um ambiente real a utilização de co-simulação com o OMNeT++ de forma a validar a plataforma criada para a produção de simulações de redes IoT. De tal forma será possível mostrar que a co-simulação funciona adequadamente no ambiente de simulações OMNeT++, e fornecer uma base para eventuais futuras expansões.

Basicamente temos um conjunto de componentes que estarão conectados entre si sem fios (criando uma RSSF), que estarão ligados (através de TCP/IP) a um computador executando OMNeT++, o qual suporta uma topologia de rede IP criada dentro do simulador. Deste modo é criada uma rede completa de dispositivos, tanto simulados, quanto físicos, e que pode ser usada para recolher resultados estatísticos do seu desempenho, nomeadamente dentro do OMNeT++.

Tivemos como resultado de recepção de pacotes em média 64,4%, demorando cerca de 1,20 segundos para a transmissão de cada pacote. Acreditamos que as ineficiências possam ter sido devidas tanto ao mau contato dos componentes, como à ineficiência do código projetado.

Palavras Chave: Internet of Things (IoT), Co-simulation, Internet das Coisas, Co-simulação HW/SW, Redes de Computadores.

Abstract

The construction of the co-simulated network (physical network and simulated network) of this thesis will be a basis for the development of simulation techniques using the OMNeT ++ (computational) platform with (physical) sensors. Once the platform is established, it would be possible to extend the operation of the network to other environments, such as the development of physical networks in mining. In this sense we can build a physical network, and simulate the behavior of other components and networks using OMNeT ++.

As this work uses IoT for the development of the proposed topology, this work ends up corroborating the use of low power appliances.

The objective of this dissertation is to show in a real environment the use of co-simulation with OMNeT ++ to validate the platform created for the production of simulations of IoT networks. In such a way it will be possible to show that the co-simulation works properly in the OMNeT ++ simulations environment, and to provide a basis for possible future expansions.

Basically we have a set of components that will be connected to each other wirelessly (creating an RSSF), which will be connected (via TCP / IP) to a computer running OMNeT ++, which supports an IP network topology created inside the simulator. In this way a complete network of devices, both simulated and physical, is created and can be used to collect statistical results of its performance, namely within OMNeT ++.

We had because of receiving packets on average 64.4%, taking about 1.20 seconds for the transmission of each packet. We believe that the inefficiencies may have been due to both the bad contact of the components and the inefficiency of the designed code.

Índice Geral

Dedicatória.....	iii
Agradecimentos	v
Resumo	vii
Abstract.....	ix
Índice Geral	xi
Índice de Figuras	xv
Índice de Tabelas	xvii
Índice de Listagens	xix
Capítulo 1 Introdução	1
1.1. Disposição do projeto	1
1.2. Motivação	2
1.3. Objetivos.....	2
1.4. Estruturação da tese	3
Capítulo 2 Integração ao tema.....	5
2.1. Internet das coisas	5
2.1.1. Contextualização	5
2.1.2. Meio de transmissão de dados.....	6
2.1.3. IoT no futuro.....	7
2.1.4. Desafios da IoT.....	7
2.2. Definição de Simulação	8
2.3. Co-simulação	9
Capítulo 3 Estado da arte.....	12
Capítulo 4 Metodologia.....	15
4.1. Introdução	15
4.2. Objetivo	16
4.3. Execução	18
4.3.1. ZigBee	19
4.3.1.1. <i>Command or Transparent and API mode</i>	20
4.4. Conclusão.....	22
Capítulo 5 Topologia.....	25
5.1. Introdução	25
5.2. Objetivo da topologia física.....	29
5.3. Objetivos da rede simulada.....	29
5.4. Dispositivos utilizados.....	30

5.4.1.	Arduíno.....	30
5.4.2.	xBee.....	30
5.4.3.	Raspberry Pi	30
5.5.	Software gerenciador da transmissão.....	30
Capítulo 6	Sistema no quesito hardware	33
Capítulo 7	Sistema no quesito software	41
7.1.	Introdução	41
7.2.	Software para parte física.....	41
7.2.1.	Definições dos pacotes	42
7.2.2.	Sensor de luminosidade	42
7.2.3.	Sensor de temperatura	43
7.2.4.	Explicação das atribuições das funções escritas para os Arduínos	44
7.2.1.1.	Transmissão por xBee	44
7.2.1.2.	Definições.....	44
7.2.1.3.	Resumo das funções utilizadas nos Arduínos	45
7.2.5.	Conexão entre um Arduíno com os outros.	46
7.2.6.	Raspberry Pi	46
7.3.	Software para a parte simulada	47
Capítulo 8	Dados obtidos e resultados	51
8.1.	Introdução	51
8.1.1.	Primeiro Teste	52
8.1.2.	Segundo Teste	56
8.1.3.	Terceiro teste	58
8.1.4.	Quarto teste.....	60
8.1.5.	Quinto teste.....	62
8.1.6.	Sexto teste.....	64
8.2.	Análise dos dados	66
8.2.1.	Arduíno 1	67
8.2.2.	Arduíno 2.....	67
8.2.3.	Arduíno coordenador	67
Capítulo 9	Conclusão e Trabalhos futuros	69
9.1.	Conclusão.....	69
9.2.	Análise dos resultados	70
9.3.	Trabalhos futuros	71
Bibliografia.....		72
Anexos.....		75
Anexo teste 1		75

Anexo teste 2	76
Anexo teste 3	77
Anexo teste 4	78
Anexo teste 5	79
Anexo teste 6	80

Índice de Figuras

Figura 1 Processo Hardware-in-loop Fonte: Youtube	10
Figura 2 Processo Hardware-in-loop com Hardware	10
Figura 3 Arquitetura de uma rede RSSF	16
Figura 4 Topologia completa.....	17
Figura 5 - Hardware-in-the-loop.....	19
Figura 6 ZigBee par, estrela, malha e árvore.....	20
Figura 7 Recebimento do dado pelo xBee.....	21
Figura 8 Transmissão do dado para o outro componente xBee.....	21
Figura 9 Chegada do dado e envio para o computador	21
Figura 10 Dado recebido	21
Figura 11 Transmissão API.....	22
Figura 12 - Topologia da rede física.....	26
Figura 13 Topologia de rede dentro do OMNeT++	28
Figura 14 Nó construído para a transmissão de dados.	33
Figura 15 TMP 36	34
Figura 16 Exemplo de instalação TMP36	34
Figura 17 Sensor LDR.....	35
Figura 18 configuração do sensor LDR.....	35
Figura 19 Configuração do xBee.....	36
Figura 20 Tipos de configuração do xBee.....	36
Figura 21 Shield xBee	38
Figura 22 Conexão do xBee com a placa Arduino.....	38
Figura 23 Conexão serial do Raspberry Pi com Arduino	39
Figura 24 Topologia física.....	39
Figura 25 Desbloqueio dos Arduínos via Xbee.....	46
Figura 26 Dispositivos montados para a transmissão.....	53
Figura 27 Dispositivo com barreira	53

Índice de Tabelas

Tabela 1 Dados da mensagem enviada pelos Arduínos	27
Tabela 2 Mensagem devolvida do OMNeT++ para Arduínos.	28
Tabela 3 Struct definida no Arduíno.	42
Tabela 4 Trecho de código para leitura do LDR	43
Tabela 5 Codificação do sensor de temperatura	43
Tabela 6 Serial para xBee	44
Tabela 7 Função Set interface	47
Tabela 8 Condições do teste 1	54
Tabela 9 Condição do teste 2.....	56
Tabela 10 Função de análise das mensagens	81
Tabela 11 Código Python	105
Tabela 12 dados teste 1.....	107
Tabela 13 Dados teste 2.....	112
Tabela 14 Dados teste 3.....	114
Tabela 15 Dados teste 4.....	118
Tabela 16 Dados teste 5.....	122

Índice de Listagens

Nenhuma entrada de índice de ilustrações foi encontrada.

Capítulo 1 Introdução

Este trabalho foi pensado para suprir uma necessidade da simulação computacional, que trabalha na integração de objetos físicos (reais) em redes simuladas. Este trabalho apresenta uma conexão de rede física (Real) ligada a um software de simulação com uma rede totalmente simulada, a qual foi criada para possibilitar testes para uma adequação do software OMNeT++. Esta dissertação foi desenvolvida no Instituto Politécnico de Bragança (IPB) num programa de dupla diplomação com a Universidade Tecnológica Federal do Paraná (UTFPR).

1.1. Disposição do projeto

A simulação pode servir, de grosso modo, para avaliar se o que se está a pretender é possível ou não de ser executado, o que evita custos adicionais ao projeto e pode viabilizar a sua construção.

Quando falamos de simulação geralmente traçamos uma ideia onde pensamos que tudo pode ser simulado por computadores, mas o fato é que nem sempre isso é possível, pois os equipamentos, muitas vezes, são complexos e é difícil serem totalmente simulados. Para isso surge a Co-simulação, que será tratada com mais detalhes nos capítulos abaixo.

A co-simulação é definida por (Steinbrink et al., 2017) como a união de dois ambientes diferentes que colaboram para suprir um propósito específico, portanto podemos dizer

que podemos utilizar a co-simulação para validação de um projeto com partes integradas ao software.

Esta dissertação tem como objetivo geral mostrar a possibilidade de integração do OMNeT++ com uma rede de dispositivos físicos geralmente associados à Internet das Coisas (Internet of Things - IoT).

1.2. Motivação

Esta tese foi criada com o objetivo de mostrar a conexão entre o OMNeT++ com dispositivos reais num ambiente de simulação IoT, de forma que os mesmos executem tarefas para gerar informação a ser analisada estatisticamente. A proposta foi estabelecida dada a importância da simulação por computadores para uma análise mais eficaz do comportamento do sistema, em que podemos construir modelos para simular os diferentes sistemas em vez de ter que recorrer à sua implementação. Porém dada a dificuldade na simulação de dispositivos complexos/recentes e porque para certos dispositivos físicos, apenas pretendemos analisar o seu comportamento quando integrado no modelo completo, e não individualmente, precisamos utilizar a co-simulação para garantir a integração de componentes simulados e componentes físicos.

1.3. Objetivos

Garantir um ambiente adequado para a construção da co-simulação e para que a mesma se execute de forma a permitir a recolha de dados sobre o comportamento do sistema IoT estudado.

Construir uma topologia capaz de representar um sistema IoT, criando uma implementação usando o OMNeT++, a qual será testada de forma a mostrar que a técnica utilizada é capaz de integrar o OMNeT++ e o Hardware físico, mesmo que ambos possuam parâmetros diferentes para execução.

A execução das duas etapas anteriores será importante para mostrar a possibilidade de conexão do OMNeT++ com equipamentos físicos, neste estudo equipamentos do tipo IoT.

1.4. Estruturação da tese

Baseado na discussão dos tópicos citados acima, a fim de facilitar a compreensão da tese, iremos descrever abaixo como os capítulos foram divididos.

Capítulo 1 Introdução. É feita uma introdução, mostrando sua motivação e os objetivos que devem ser atingidos no decorrer do trabalho.

Capítulo 2 Integração ao tema. Iremos dar uma contextualização dos conceitos empregues para a construção da dissertação, a fim de possibilitar um melhor entendimento do leitor.

Capítulo 3 Estado da arte. Discutiremos como outros autores resolverem problemas semelhantes no que se diz respeito a co-simulação, como implementaram essas ideias e os métodos utilizados.

Capítulo 4 Metodologia. Nesse capítulo iremos apresentar como foi definido o processo seguido para a implementação do código e como o mesmo será testado para verificar se os resultados pretendidos foram alcançados.

Capítulo 5 Topologia. Nesse capítulo explicaremos como foi planejada a topologia e explicaremos como a mesma funciona no ambiente teste.

Capítulo 6 Sistema no quesito Hardware. Nesse capítulo vamos explicar um pouco melhor sobre o Equipamento Físico usado.

Capítulo 7 Descrição do software. Nesse capítulo iremos mostrar o código desenvolvido no OMNeT++ para possibilitar uma conexão com sistemas reais, bem como a execução dos códigos a correr nos componentes físicos escolhidos.

Capítulo 8 Dados e resultados obtidos. Nesse capítulo serão analisados os dados obtidos na execução da topologia escolhida e mostraremos através de dados estatísticos o comportamento do sistema modelado.

Capítulo 9 Conclusão e Trabalhos Futuros. O que o trabalho trouxe de contribuição para ao meio acadêmico e a utilização do trabalho como base para desenvolvimentos futuros.

Capítulo 2 Integração ao tema

2.1. Internet das coisas

2.1.1. Contextualização

As mudanças do mundo, principalmente em relação a tecnologia, nos fazem pensar em um mundo totalmente conectado. Amparando essa ideia, foi criado o conceito de IoT, que basicamente é uma forma de possibilitar a conexão de objetos, anteriormente não conectados à Internet. “The basic idea of this concept is the pervasive presence around us of a variety of things or objects (Atzori, Iera, & Morabito, 2010) como explicado pelos autores os objetos inseridos no nosso cotidiano estarão realizando ações, para nos auxiliar em tarefas diárias, os quais irão comunicar entre si, a fim de atingir o objetivo desejado.

A IoT conta com uma vasta quantidade de objetos que podem ser utilizados para compor uma rede, os mesmos se tornarão inteligentes quando conectados através de um *Internet Service Provider* ISP, os quais irão partilhar dados para a produção massiva de informação. Cada dispositivo IoT possui um objetivo diferente, os quais podem ser definidos para cada ambiente que for utilizado, como indústria, hospitais, campos agrícolas, dentre outros (Cisco, 2016).

A IoT tem se expandido muito em diversos campos de serviço, pois os dispositivos são numerosos, podendo ser aplicados em várias áreas. Hoje a IoT tem sido aplicada como por exemplo na indústria 4.0, ligada à indústria inteligente (Wortmann & Flüchter, 2015). Como já dito a IoT tem vários campos de aceitação, sendo muito importante atualmente para o mundo moderno, eles se expandem em casas inteligente, sistema de segurança inteligente e até mesmo sistema de lixo inteligente. Por isso é necessário uma plataforma que execute testes consistentes de forma que quando aplicados na vida real esses sistemas

não possuam falhas que possam prejudicar a vida humana ao invés de ajudá-la, por exemplo durante o ano de 2001 um hacker mal intencionado, depois de trabalhar em uma empresa de gerenciamento de lixo inteligente, resolveu prejudicar a empresa, o mesmo conseguiu contaminar rios e lugares públicos (Smith, 2001).

Para que os dispositivos conectados à malha de rede funcionem, eles necessitam de sensores capazes de captar dados do ambiente e posteriormente executar alguma ação, as informações obtidas pelos dispositivos equipados com sensores podem ser armazenadas em um repositório de dados e então podem aprender com erros e acertos, para melhorar a forma de executar uma determinada tarefa no ambiente. Para a recolha/armazenamento de dados contamos com tecnologias como Fog e Cloud Computing. Essas tecnologias são utilizadas para partilhar dados que recebemos dos dispositivos para os utilizarmos de forma adequada nos nossos problemas, algumas empresas têm usado isso para colaboração industrial, a fim de aumentar os lucros (Cisco, 2016).

2.1.2. Meio de transmissão de dados.

A maioria das pessoas ligadas a computação quando pensam em algum sistema, já deduzem que a transição de dados ocorre totalmente via computador, pois estamos falando da obtenção de fatores do ambiente, no entanto não está completamente certo, pois existem vários tipos de transição os quais serão mostrados abaixo. De acordo com (Cisco, 2016) existem 3 tipos:

- M2M: Basicamente esta é uma definição que abrange informações trocadas entre computadores, Machine to Machine, portanto essa definição está baseada na captação de informação por sensores e posteriormente a partilha desses dados, totalmente via computadores.
- M2P: Apesar dessa definição também contar com a transição de dados utilizando computadores, a transmissão da informação ocorre pela ligação entre as pessoas e computadores. As pessoas possuem algum tipo de dado sobre o ambiente e transmitem os acontecimentos para os dispositivos, por outro lado o dispositivo transmite os dados recolhidos do ambiente para pessoa, Machine to People
- P2P: E quando tratamos de People to people estamos dizendo que a transmissão de dados ocorre das conversas e discussões das pessoas em torno de algum assunto.

Para termos uma conexão completa de IoT precisamos de dispositivos conectados entre eles, um repositório de dados para armazenar informação que possa servir para prever acontecimentos e capacitar decisões, feedback de pessoas para um sistema de correção e aceitação dos dados gerados, como também as três classes citadas acima, M2M, M2P e P2P para a transmissão geral dos dados.

2.1.3. IoT no futuro.

Com a IoT deverão existir bilhões de dispositivos conectados capazes de criar exabytes de dados, esses dados são armazenados e assim da união de vários componentes temos uma inteligência.

Poderíamos utilizar IoT, por exemplo, numa exploração agrícola. Numa exploração agrícola sem nenhum componente eletrônico, não é possível saber a humidade, temperatura, taxa de crescimento e a acidez do solo. Por outro lado, colocando alguns equipamentos para medir, através de sensores, o ambiente podemos saber diversas informações, que podemos partilhar entre os computadores para saber por exemplo que ação deverá ser tomada em função da informação recolhida. Esse é um dos tantos experimentos que podemos fazer utilizando IoT (Cisco, 2016).

2.1.4. Desafios da IoT

Mesmo sendo usados diferentes dispositivos na IoT deve ser possível trocar a informação entre eles, porém muitos desenvolvedores criam protocolos próprios os quais são inseguros, na maioria dos casos, e não são padronizados o que dificulta a comunicação. Atualmente o protocolo mais utilizado é o IP, porém nem todos os equipamentos o implementam.

O uso da IoT é extremamente vantajoso no mundo atual, o maior desafio que enfrentamos é implementá-la. Como a IoT trabalha com “coisas”, elas podem ser pequenas como é o caso de relógios, aparelhos auxiliares na saúde, controles, entre outros, isso dificulta a expansão da IoT pelo mundo, apesar do seu alto crescimento e aceitação pelas pessoas. São necessárias por exemplo baterias com dimensões extremamente reduzidas e com uma vida considerável.

2.2. Definição de Simulação

Para entendermos como funciona a simulação torna-se necessário entender o que é um modelo. Um modelo tenta reproduzir, uma aproximação, idealização ou estrutura do real, de forma a manter a essência do alvo a ser estudado (Batista, Salvi, & Lucas, 2011), porém o conceito de simulação se diferencia do modelo, em partes. Segundo (Costa & Almeida, 2015) a simulação utiliza o modelo, no entanto se difere, pois para a simulação é importante o conjunto de entradas, para então conseguir antecipar como o sistema irá agir e posteriormente ser estudado.

Na simulação temos o conceito de tempo e isso é muito importante, pois dependendo do tipo de simulação temos que utilizá-lo de maneira diferente, isso é, se um evento ocorre em 3 em 3 minutos, não é inteligente receber dados durante todo tempo, pois um evento só irá ocorrer nos próximos 3 minutos a partir do último evento.

A simulação pode ser dividida em três tipos segundo (Kopetz, 1999) no nível conceitual a principal diferença entre estes tipos é a natureza da base de tempo usada.

- Simulação em tempo contínuo, de acordo com (Kopetz, 1999) é quando temos uma base de tempo e sabemos que entre quaisquer dois instantes de tempo escolhidos, temos algum acontecimento entre eles. O estado do sistema muda continuamente ao longo do tempo.
- Simulação em tempo discreto, de acordo com (Kopetz, 1999) o estado do sistema só pode mudar em pontos discretos, em geral distribuídos uniformemente no tempo.
- Simulação em evento discreto (Kopetz, 1999) diferente da simulação por tempo discreto, esta somente se importa com o acontecimento em si, onde a simulação dá um salto para o próximo acontecimento que irá ocorrer, sendo que o tempo entre um acontecimento e outro não é o mesmo.

Para testes mais massivos são necessárias simulações para podermos entender como irá funcionar uma determinada topologia quando implementada em algum cenário real. Para isso, é utilizado algum programa de simulação que disponibiliza funções para implementar o comportamento dos componentes individuais.

O autor (Jacobson, Hall, & Swisher, 2006) explica que simulação é uma metodologia de modelagem e análise, na qual usuários finais podem verificar a eficiência de determinado produto. Isso é muito utilizado pois com esse tipo de teste, podemos não só simular ideias já existentes como também produzir diferentes situações no sistema.

A simulação pode ajudar também na diminuição de gastos, tendo por base que objetos simulados antes de implementados em algum ambiente podem possuir algum tipo de restrição, a simulação evita os problemas que derivam de ideias precipitadas e utilização de dispositivos errados.

2.3. Co-simulação

A Co-simulação é definida por (Steinbrink et al., 2017) como “*Co-simulation is defined as the coordinated execution of two or more models that differ in their representation as well as in their runtime environment*”. Assim, a Co-simulação é definida como a união de um ou mais ambientes, poderíamos falar que a Co-simulação é um processo de simbiose, onde dois ambientes distintos trabalham juntos como forma de alcançar um bem comum entre eles. Nas Aplicações que iremos citar neste trabalho trabalharemos com um ambiente simulado e outro real. Para um melhor entendimento serão explicados adiante os dois tipos mais comuns de co-simulação, utilizando um ambiente totalmente simulado por computador e um ambiente real.

- Hardware-in-loop (HiL) é uma técnica de simulação, a qual utiliza dispositivos reais. Segundo (Costa & Almeida, 2015) para tal é colocado um equipamento de hardware real dentro de uma simulação de computador para aumentar a confiabilidade da simulação, tornando-a mais realista, uma vez que é complicado a modelagem de um dispositivo muito complexo, dentro de uma simulação. A HiL costuma substituir alguma parte de um sistema simulado.

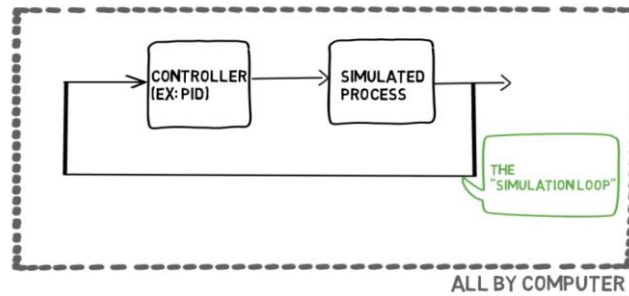


Figura 1 Processo Hardware-in-loop
Fonte: Youtube

Podemos verificar um exemplo de simulação que ocorre totalmente dentro de um computador, a qual possui um controlador também simulado, a diferença para o HiL é que uma parte não é simulado por computador, como pode ser visto da Figura 2 Processo Hardware-in-loop com Hardware.

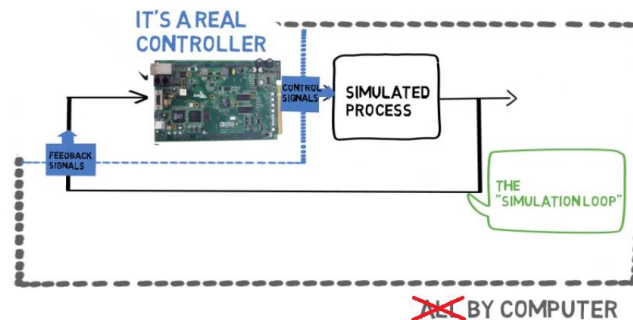


Figura 2 Processo Hardware-in-loop com Hardware

Fonte: YouTube

Tanto HiL como Robot-in-Loop (RiL) são importantes para entender uma simulação, porém cada uma é usada em tipos diferente de simulação. O RiL é utilizado para ambientes que são geralmente inóspitos, sendo que não possuímos o ambiente para testes, exemplo o fundo do mar, como é explicado por (Costa & Almeida, 2015). A HiL pode ser utilizada de forma a contornar o problema de projetar um hardware inteiro dentro de uma simulação, uma vez que alguns sistemas têm desempenhos muito complexos, e difíceis de serem modelados em software.

A Co-simulação será utilizado nesse trabalho para unir uma rede IoT (contendo um controlador IoT Raspberry Pi) com uma rede IP simulada no OMNeT++. Dessa forma poderemos simular uma rede completa, numa parte com componentes físicos obtendo/gerando dados reais e noutra parte uma rede (IP) simulada, modelando o funcionamento de um sistema complexo

Capítulo 3 Estado da arte

Nesse capítulo iremos tratar sobre trabalhos realizados anteriormente a este trabalho, mostrando assim o que já foi produzido durante os anos de estudo, tanto na área de co-simulação, quanto no que se diz respeito a produção de co-simulação ligada a IoT.

A fundamentação teórica deste trabalho tem como base auxiliar outros trabalhos na área de forma a possibilitar uma ideia dos trabalhos criados e a construção dos mesmos.

Algumas pesquisas mostram que é possível a criação de um ambiente Co-simulado, porém este é ainda pouco utilizado em trabalhos utilizando o simulador OMNeT++, proposto para construção desse trabalho. Este trabalho conta com o uso de tecnologias como Arduíno, xBee, Raspberry Pi e OMNeT++, para simular o comportamento de redes com dispositivos IoT.

Para (Rowson, 1994) a co-simulação envolvendo dispositivos físicos reais e dispositivos simulados, apesar de complicada, no ano de publicação do artigo, é válido, porém, o impedimento estava nos dispositivos reais, os quais trabalhavam com tempos diferente de processamento, podendo assim, como o autor diz, retardar a simulação. No tempo em que esse artigo foi publicado o desafio era o tempo de trabalho entre dispositivos diferentes, para exemplificar, podemos citar um aparelho que trabalha na escala de 2 GHz e um aparelho que trabalha na escala de 1 GHz, ambos executaram um conjunto de instruções em uma velocidade rápida para nós humanos, porém os dispositivos irão trabalhar com uma rapidez diferente, o que dificulta na construção de topologias Co-Simuladas. Diferente desse trabalho, nós iremos usar computador mais eficientes que da

época do artigo, isso nos ajuda, pois várias dificuldades encontradas pelos autores, como por exemplo o processamento de informações, acaba sendo facilitado pelos dispositivos existentes na época da criação desta dissertação. Nesse aspecto nosso trabalho pretende mostrar que atualmente o desenvolvimento de aplicações co-simuladas é facilitado pelos computadores atualmente disponíveis, fator que torna possível executar a co-simulação sem alguns dos anteriores problemas.

Para a criação do nosso trabalho é necessário que seja ajustado o formato de simulação do OMNeT++, de forma a podermos permitir a ligação entre o OMNeT++ e dispositivos físicos. Um trabalho parecido com esse foi realizado em (Filho, 2014), é possível ver a utilização da técnica de Hardware-in-the-Loop na Rede de Sensores Sem Fio (RSSF), o qual mostra que apesar das dificuldades encontradas no primeiro trabalho apresentado, é possível utilizar a co-simulação, mesmo com a limitação de tempo que encontramos hoje, como velocidade de processamento, uma vez que os dispositivos simulados não executam o processamento no mesmo tempo dos dispositivos reais, e como resultado, o autor conseguiu estabelecer uma alternativa para escalar testbeds. O autor utilizou uma técnica que será utilizada nos modelos desse trabalho para executar a união das plataformas aqui mencionadas o Hardware-in-Loop.

Os autores (Hofstein, Shore, & Kipnis, 2004) utilizam a co-simulação com o OMNeT++ justificando que os modelos comuns de simulação devem ser substituídos quando temos dispositivos complexos, principalmente considerando o alto nível de paralelismo que alguns possuem. Nesse trabalho, de acordo com o autor, as especificações feitas em tempo de simulação para processadores possuem um nível de complexidade muito elevado, pela alta taxa de paralelismo no nível de instrução, dentro de um processador, ou seja, dispositivos que tenham alto nível de complexidade no quesito físico de máquina seriam mais complexos de ser simulados. Para esclarecer o que é narrado no texto os autores utilizam para uma explicação mais clara um microprocessador simulado em um programa chamado MARS. Da mesma forma o nosso trabalho emprega co-simulação de forma a confirmar o comportamento de equipamentos Internet das coisas (IoT), em projetos complexos, pois muitas vezes teremos topologias complexas com equipamento difícil de simular empregadas em ambientes reais, como na agricultura, mineração ou casas inteligentes.

No trabalho de (JESUS, 2013) o mesmo mostra a utilização do OMNeT++ para simular os dados transmitidos entre carros reais autônomos em autoestradas, o trabalho não usa uma rede estática com um nó sorvedouro, mas uma rede dinâmica, com a aproximação de mais carros dentro do alcance de uma RSSF. No trabalho ele utiliza um programa para que haja a troca das posições dos carros, esse programa se chama MiXiM. Para as funcionalidades do trabalho construído por ele, esse programa tem a função de projetar os dados externos para o OMNeT++. O trabalho propôs descrever como seria a transmissão de dados em ambientes reais, com a ideia de encontrar em trabalhos futuros uma solução para diminuir o tráfego. A diferença do proposto para o nosso trabalho é basicamente a fórmula a qual construímos nossa topologia, primeramente escolhemos trabalhar com uma ambientação diferente, agrícola, em segundo nossa topologia é estática e não móvel, como ocorre com a simulação de (JESUS, 2013), porém o ponto principal do nosso trabalho é mostrar que existe a possibilidade que o trabalho aqui citado seja produzindo com dispositivos físicos reais e não simulados e conectados com o OMNeT++, pois apesar dos ambientes serem diferentes, os testes que serão produzidos nos nossos sensores poderá ser reproduzido com carros, caso isso seja necessário. Sendo assim como o trabalho dele mostra as diferentes situações se empregarmos diferentes condições para os sensores utilizados, podemos projetar problemas de distância na rede física da mesma forma que foi feita, porém não iremos fazer isso no nosso trabalho, apesar de possível.

No trabalho dos autores (Boehm & Kirsche, 2015), eles mostram a utilização em conjunto do OMNeT++ com o RoSeNet, de forma a gerir uma co-simulação, a ideia do projeto em desenvolvimento na publicação do artigo citado era criar uma API para facilitar a conexão de dispositivos a uma interface simulada OMNeT++. Porém durante a escrita desta dissertação não foram encontradas novas publicações do autor sobre o sucesso ou falha dos procedimentos aplicados, ou tão pouco a continuação do seu trabalho no tema. A aplicação aqui proposta tem uma ideia base parecida com a citada pelo autor do artigo, porém com um cenário diferente do mencionado no artigo, uma vez que não iremos usar a topologia que o mesmo propôs.

Capítulo 4 Metodologia

4.1. Introdução

Existem técnicas que são utilizadas para conectar sensores a algum tipo de base de dados (BD) que irá registrar esses dados para fins de análise. Uma das topologias que pode ser explorada é a RSSF, a qual é uma rede que possui sensores capazes de analisar um ambiente, monitorando um fenômeno. No nosso experimento iremos construir uma topologia a qual pode ser aplicada, por exemplo, na agricultura para levantamento das informações contidas no ambiente desejado.

As RSSF são geralmente um conjunto de nós de sensores (nodos sensores), de tamanho reduzido, distribuídos pela área a ser avaliada, como pode ser observado na Figura 3 Arquitetura de uma rede RSSF. Uma rede sem fios estruturada possui uma ligação a uma estação base, semelhante ao que ocorre em redes de celular, contudo redes do tipo ad hoc partilham informações de um nó ao outro até chegar à estação base e daí para a base de dados onde os dados serão armazenados para ser posteriormente analisados (Poderoso et al., 2015). Esses nós trabalham em conjunto para a transmissão de dados, através de uma rede sem fio e ocorre uma entrega desses dados ao que chamamos de nó sorvedouro (sink node), onde serão tratados e distribuídos para os outros componentes da topologia. para a tomada de decisões, essas decisões são tomadas com base de critérios pré-estabelecidos em alguma parte da topologia. A estação central, à qual os dados serão enviados deverá ser capaz de manipular os elementos que chegarão, através dos sensores, para ela, de forma a executar uma ação no ambiente, utilizando atuadores (Carvalho et al., 2015), como nosso ambiente é co-simulado deveremos então enviar os dados para um dispositivo (simulado ou não) dentro da nossa topologia. A avaliação dos dados deverá ser feita pelo OMNeT++, porém devemos separar a ideia de dados em duas concepções diferentes.

Primeiramente vamos diferenciar os dados que serão enviados dos dados que serão avaliados. Os dados que serão enviados são dados obtidos pelo sensores, os quais para título de diferenciação serão chamados nos seguintes parágrafos de mensagens, os mesmos serão importantes pois com eles é possível verificar se a transmissão das mensagens do nosso projeto estão sendo enviadas corretamente, porém os dados não irão interferir diretamente na cadeia de dados estatísticos que iremos gerar, portanto entramos no segunda diferenciação, dados estatísticos, os mesmo serão gerados a partir da forma que ocorre a transmissão das informações, independente de como isso ocorre.

Nosso trabalho será pensar em um método de utilizar IoT de tal forma construir uma topologia capaz de ser utilizada dentro de um ambiente como agricultura, apesar de podermos utilizar a IoT para vários outros problemas, escolhemos esse para testar as funcionalidades da topologia construída e obtermos dados estatísticos.

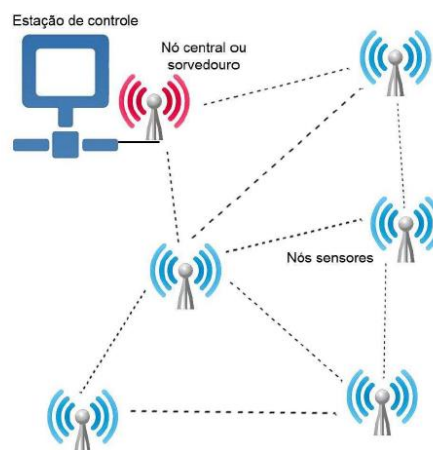


Figura 3 Arquitetura de uma rede RSSF

Na implementação do nosso trabalho não iremos utilizar um conjunto muito grande de nós sensores, pois o que se pretende é mostrar a aplicação do conceito de co-simulação. Como trabalho futuro poderá ser aplicado o conceito a redes mais complexas usando este tipo de topologia.

4.2. Objetivo

Nesta seção será discutido o plano que deverá ser traçado, por exemplo para descobrir as necessidades de uma determinada região de plantio. Nesta etapa será mostrado como será montada a tecnologia para receber dados e aproveitá-los de forma a estabelecer as

diretrizes corretas para o manejo do solo. A intenção foi montar uma topologia (melhor definida no capítulo referente a topologia), de forma obter dados para serem analisados dentro do OMNeT++ através das ferramentas do programa, isso foi feito com o objetivo de notar a performance da rede, como *throughput*, *delay*, taxa de erro, dentre outros e assim projetar as estatísticas referentes aos cenários estudados.

Portanto tendo em vista tudo que foi discutido anteriormente nesse ponto conseguimos ver como ocorre o comportamento de uma RSSF, por isso podemos analisar o modelo criado nessa dissertação. Como pode ser visto na Figura 4 Topologia completa a topologia possui duas vertentes, a rede simulada e a rede física, sendo que na parte física contamos com dispositivos que são capazes de transmitir dados por rádio frequência, sendo assim temos uma rede RSSF com dois dispositivos transmitindo dados para um *sink node* e recebendo mensagens com comandos pelo *sink node*.

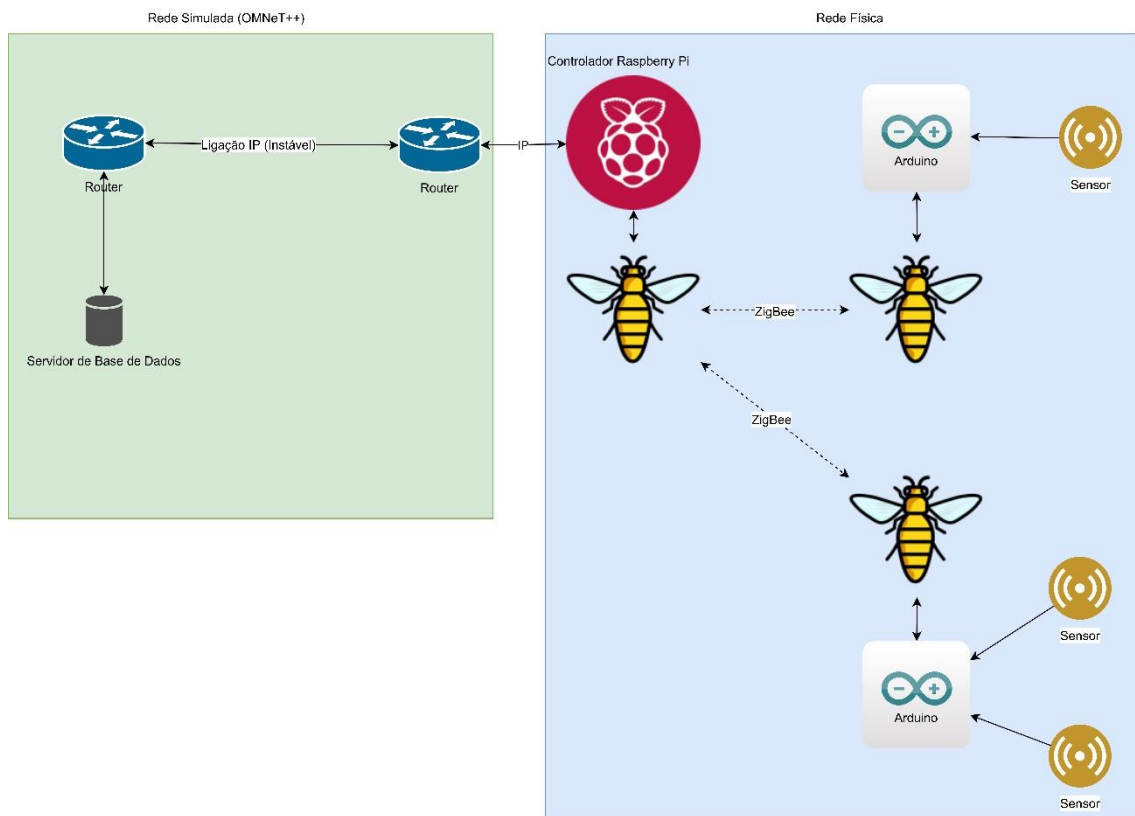


Figura 4 Topologia completa

Essa rede tem como objetivo o teste da interface de conexão para dispositivos IoT, de tal forma foi otimizada para fins acadêmicos, porém os mesmos devem ser fortemente testados para a obtenção de resultados.

Os dispositivos reais devem ser dispostos de forma a manter a operacionalidade sem prejuízo, tanto na qualidade do sinal da rede RSSF, quanto na transmissão dos dados. Como temos vários tipos de plantações, testaremos essa topologia em um número limitado de ambientes, porém com diferenças entre um e outro, justamente para simular os diferentes tipos de plantação, principalmente relação aos fenótipos das plantas escolhidas, em razão de termos diferentes condições de transmissão, com diferentes tipos de lavoura, essa topologia será testada em algumas condições ajustadas por nós que poderão ser implementadas nas produções agrícolas.

4.3. Execução

Para os testes referentes a esse projeto foi necessário criar a topologia proposta para analisar como está ocorrendo a transmissão e verificar os possíveis problemas que podem ocorrer em um modelo de rede. Dessa forma começamos por estudar se já havia trabalhos feitos em co-simulação, para darmos início a produção da nossa proposta, de forma a criar uma topologia diferente das existentes utilizando o OMNeT++.

A construção da topologia começou com o estudo mais aprofundado do OMNeT++, dessa forma obtivemos uma maneira de conectar dispositivos físicos ao OMNeT++, tal fato nos proporcionou que pudéssemos conectar qualquer dispositivo que trabalhe com protocolos do tipo TCP ou UDP ao OMNeT++.

A ligação entre os nossos dispositivos IoT e o computador central executando uma topologia simulada no OMNeT++, sugerida nesse trabalho, ocorre por intermédio do Hard-in-the-Loop.

A Figura 5, mostra onde iremos aplicar a técnica de Hardware-in-the-loop.

Basicamente teremos um computador executando uma topologia dentro do OMNeT++ conectado com um Raspberry Pi, o qual irá transmitir as informações adquiridas no meio, dessa forma utilizaremos a mesma técnica do artigo de (Filho, 2014), já citado nessa dissertação, mas com uma topologia distinta.



Figura 5 - Hardware-in-the-loop

Nesse sentido começamos a montar a topologia em relação a rede física, para podermos executar os testes na rede de forma adequada as nossas propostas. Para a realização do trabalho resolvemos optar pelas tecnologias narradas nas seguintes seções.

4.3.1. ZigBee

Como as escalas de um campo agrícola são notavelmente grandes, optamos por utilizar um padrão de comunicação que suporte tecnologias sem fio, para a conexão entre dispositivos. A grande maioria dos sensores, sem fio, utilizam radiofrequência (RF), para comunicação, com isso a distância entre os nós é variável, o que corrobora a utilização de ZigBee para o nosso projeto, pois é um padrão bastante usado em soluções sem cabeamento, se utilizado saltos múltiplos como é explicado por (Poderoso et al., 2015). O padrão ZigBee é baseado no IEEE 802.15.4, operando na grande maioria das vezes na faixa não licenciada de 2,4 GHz, embora isso mude em alguns países do mundo, porém ao implementar tal formato é necessário um cuidado com a propagação e degradação do sinal, Carvalho et al. (2015).

O Zigbee fornece funções de roteamento e multi-hop para o protocolo de rádio baseado em pacotes, como pode ser visto em (Digi International, 2018). Implementando o Zigbee para a transmissão temos o xBee, tal hardware é poderoso em sua construção, pois possui modos de transmissão de dados que podem construir vários tipos de topologia de rede que pode ser vista na Figura 6

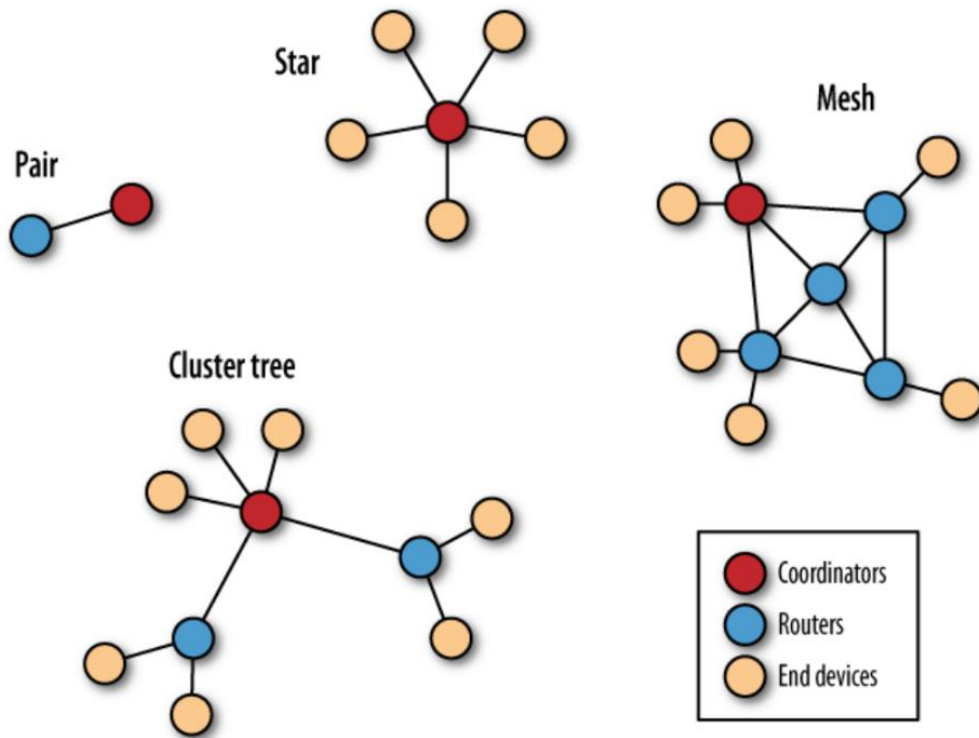


Figura 6 ZigBee par, estrela, malha e árvore

Fonte: (Faludi, 2010)

Para a configuração do xBee é necessário a utilização da ferramenta xCTU, ela possibilita a configuração do xBee, permitindo ao xBee pode executar as tarefas conforme o planejado.

4.3.1.1. *Command or Transparent and API mode*

O xBee contém dois modos de atuação, *application mode* ou *transparent mode* (AT) ou ainda *application programming interface* (API).

No modo AT temos uma transmissão direta dos dados recebidos, ou seja ocorre a transmissão pelos rádios da mesma forma que chegaram nos dispositivos, portanto se ocorreu a chegada de uma letra, a mesma será transmitida daquela forma. Isso pode ser visto com mais detalhes nas Figura 7, Figura 8, Figura 9, Figura 10 e Figura 11

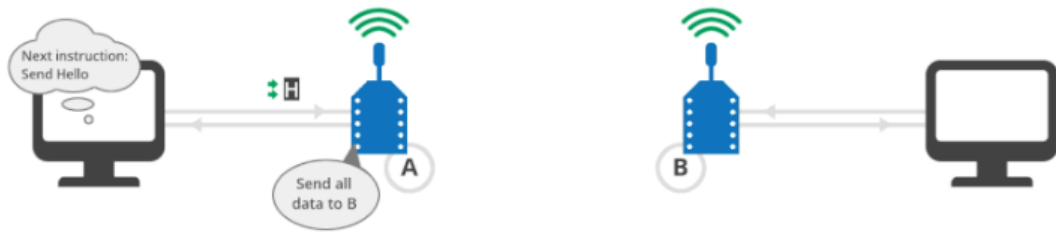


Figura 7 Recebimento do dado pelo xBee

Fonte: (DigiInternational, 2018)



Figura 8 Transmissão do dado para o outro componente xBee

Fonte: (DigiInternational, 2018)



Figura 9 Chegada do dado e envio para o computador

Fonte: (DigiInternational, 2018)



Figura 10 Dado recebido

Fonte: (DigiInternational, 2018)

No modo API o xBee possui a função de interação de um software com outro, através de uma interface padrão, feita para melhorar a comunicação entre computadores. Deve-se ressaltar que a API funciona para facilitar a comunicação entre computadores, não sendo tão intuitiva para nós humanos, portanto podemos dizer que não foi projetada para interação humana direta, porém funciona muito bem para o envio de pacotes, comumente

chamados de quadros da API, esse modo é diferenciado do AT pois encapsula as informações necessárias antes do envio. Para uma visualização do cenário no modo API podemos ver na Figura 11 Transmissão API como ocorre a transmissão dos dados, como funciona de forma semelhante ao modo AT, uma vez que é uma modificação do mesmo, optamos por simplificar a imagem para o leitor desse texto.



Figura 11 Transmissão API

Fonte: (DigiInternational, 2018)

Segundo (Faludi, 2010):

- Modo transparente, por definição apenas recebe dados e enviá-los para plataforma gerenciadora, sem alteração/modificação.
- Modo API, necessita que a plataforma gerenciadora interfira e administre os comandos do xBee. Dessa forma podemos trabalhar com dados completos entregues em formato de pacotes

Por não ter sido possível colocar a funcionar o modo API, para a elaboração dessa tese optamos por executar o xBee em modo AT, e corrigir erros de transmissão nos Arduínos receptores e transmissores de dados.

4.4. Conclusão

A construção da rede co-simulada (rede física e rede simulada) dessa tese, será uma base para o desenvolvimento de técnicas de simulação usando a plataforma OMNeT++ (computacional) com dispositivos sensores (físicos). Uma vez estabelecida a plataforma, seria possível estender o funcionamento da rede para outros ambientes, como é o caso do desenvolvimento de redes físicas na mineração. Nesse sentido podemos construir uma

rede física, e simular o comportamento de outros componentes e redes utilizando o OMNeT++.

Como esse trabalho utiliza IoT para o desenvolvimento da topologia proposta, esse trabalho acaba corroborando na utilização de aparelhos com baixa potência, mas muito úteis para os exemplos que serão simulados.

O objetivo desta dissertação é mostrar em um ambiente real a utilização de co-simulação com o OMNeT++ de forma a validar a plataforma criada para a produção de simulações de redes IoT. De tal forma será possível mostrar que a co-simulação funciona adequadamente no ambiente de simulações OMNeT++, e fornecer uma base para eventuais futuras expansões.

O trabalho aplica a topologia que será descrita mais detalhadamente no capítulo com o nome “Topologia”.

Basicamente temos um conjunto de componentes que estarão conectados entre si sem fios (criando uma RSSF), que estarão ligados (através de TCP/IP) a um computador executando OMNeT++, o qual suporta uma topologia de rede IP criada dentro do simulador. Deste modo é criada uma rede completa de dispositivos, tanto simulados, quanto físicos, e que pode ser usada para recolher resultados estatísticos do seu desempenho, nomeadamente dentro do OMNeT++.

Capítulo 5 Topologia

5.1. Introdução

Essa topologia foi criada com o objetivo de testar o desempenho base e perante possíveis alterações tanto na rede física, quanto na rede simulada dentro do OMNeT++. O caso de uso preferencial inclui dispositivos pensados para o uso na agricultura, por isso foi construído uma topologia que possibilitasse a transmissão não cabeada de dados, pois desta forma conseguimos aplicar os dispositivos em qualquer tipo de área.

A topologia que irá ser apresentada é constituída por uma parte física e outra simulada pelo OMNeT++, onde podem ser observados na parte da rede física quatro componentes principais sendo eles três Arduínos e um Raspberry Pi, esses dispositivos tem como função obter dados dos sensores e transmiti-los para o servidor dentro do OMNeT++, isso deve ocorrer para que consigamos trabalhar com os dados recebidos, e posteriormente enviar comandos para os Arduínos para que os atuadores conectados neles possam executar uma tarefa. Para a nossa rede utilizamos leds para simular um tipo de atuador, podendo ser substituídos mais para frente por atuadores mais complexos, como irrigadores, aquecedores, entre outros.

A transmissão dos dados sentido pelos sensores não serão de extrema importância no que diz respeito ao seu valor, no entanto darão bases sólidas para o estudo de problemas ligados a conexão das redes, tal como *delay*, perda de pacotes, dentre outros.

Basicamente a topologia é construída da forma mostrada na Figura 12 - Topologia da rede física, nessa imagem podemos verificar a existência de 3 Arduínos, os quais serão

equipados, cada um, com uma Xbee (placas de transmissão de dados por radiofrequência) e dois sensores, um de luminosidade e outro de temperatura.

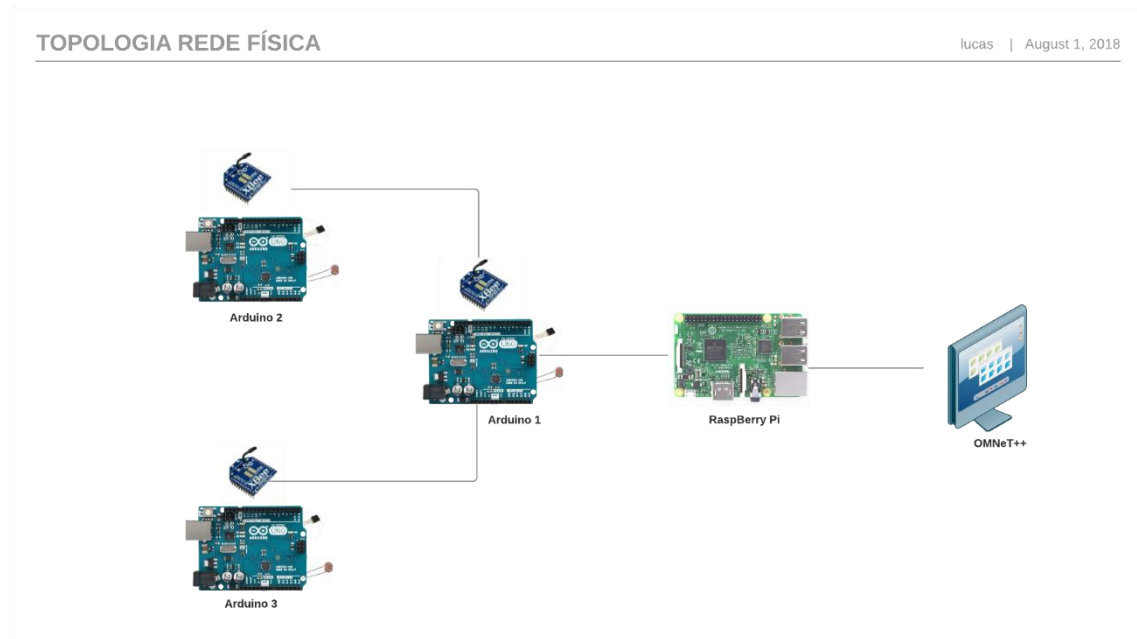


Figura 12 - Topologia da rede física

O Arduíno 1 mostrado na Figura 12 conectado ao Raspberry Pi através de uma conexão serial, dessa forma podemos dizer que o Arduíno irá ser uma ponte entre o nosso Raspberry e os outros Arduínos conectados a ele, por intermédio das xBee, portanto essa topologia irá funcionar como uma estrela (embora na prática use ligações ponto-a-ponto em modo AT).

O sensor de Luminosidade (LDR) utiliza uma foto-resistência para medir o nível de luz que chega até ele e o conversor analógico-digital do Arduíno converte os sinais de tensão do LDR em valores numéricos entre 0 e 1023, esses números são essenciais para sabermos quando podemos acender luzes no ambiente que estamos trabalhando.

O sensor de temperatura (TMP36) é utilizado para medir a temperatura do ambiente, como o nome já diz, e para que possamos analisar os dados dele obtido precisamos calcular quantos milivolts são retornados para a aplicação no Arduíno, dessa forma podemos saber a temperatura do ambiente (e eventualmente usar essa informação de forma adequada ao plantio de sementes e agricultura).

Com todos os Arduínos já montados e executando as funções planejadas, o Raspberry Pi fica responsável por enviar os dados de todos os Arduínos para o simulador OMNeT++

utilizando sockets TCP, dessa forma o mesmo fica responsável por compactar os dados e transmiti-los de forma adequada para o processamento da informação pelo OMNeT++. Portanto para a transmissão dos dados utilizaremos uma forma adequada de transmissão das mensagens, usando um padrão pré-definido. Nesse sentido todos os dados obtidos pelos Arduínos serão encapsulados em mensagens que seguem o padrão de mensagem descrito na Tabela 1.

Tabela 1 Dados da mensagem enviada pelos Arduínos

Nome do Campo	Descrição do conteúdo	Gama de valores
ID	Identificador do Nó	Nome do nó
NumSeq	Número de Sequência da mensagem	Valor inteiro (incrementado por cada mensagem)
Entrada/Saída	Indicação se é entrada ou saída	I (representando entrada)
IdSensorLuz	Identificador de Sensor de Luminosidade (SiL)	L
ValorSensorLuz	Valor do sensor de Luminosidade	0 - 1023
IdSensorTemp	Identificador de Sensor de Temperatura (SiT)	T
ValorSensorTemp	Valor do sensor de Temperatura	0 - 46

Já na rede simulada dentro do OMNeT++ utilizaremos um modulo externo para nos ligarmos aos componentes físicos, esse módulo externo se conectará através de uma nuvem simulada (essa nuvem será simulada por uma conexão ponto a ponto), a um servidor também simulado no OMNeT++, dessa forma todos os dados recebidos pela rede física seriam (em teoria) guardados no servidor dentro do OMNeT++, para ser analisados e verificados. O servidor enviará também respostas (simuladas) correspondentes a ordens para os atuadores ligados aos Arduíno.

Depois da verificação executada pelo OMNeT++ dentro do método criado para esse fim, a parte simulada da topologia irá transmitir outra mensagem, em resposta às informações obtidas para que os Arduínos executem alguma tarefa. A tarefa ficou limitada a leds instalados da protoboard, os leds serão nossos atuadores.

A mensagem que o OMNeT++ envia a parte física da topologia é mostrada seguindo a Tabela 2 Mensagem devolvida do OMNeT++ para Arduínos.

Tabela 2 Mensagem devolvida do OMNeT++ para Arduínos.

Nome do Campo	Descrição do conteúdo	Gama de valores
ID	Identificador do Nó	Nome do nó
NumSeq	Número de Sequência da mensagem	Valor inteiro (será repetido o valor recebido da mensagem construída nos Arduínos)
Entrada/Saída	Indicação se é entrada ou saída	O (representando saída)
IdSensorLuz	Identificador de Sensor de Luminosidade (SiL)	L
ValorSensorLuz	Identificador de ação para o led	I para ligar led O para deligar led
IdSensorTemp	Identificador de Sensor de Temperatura (SiT)	T
ValorSensorTemp	Identificar da ação para o led	I para ligar led O para deligar led

A topologia criada no OMNeT++ pode ser verificada na Figura 13 com mais detalhes. Também é possível verificar a topologia como um todo na Figura 4

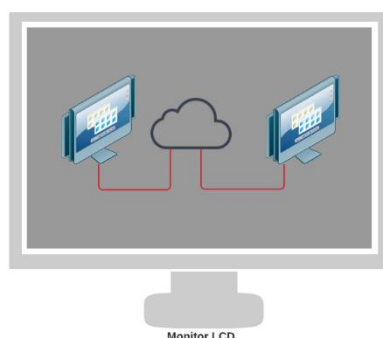


Figura 13 Topologia de rede dentro do OMNeT++

De um lado encontra o dispositivo que fará a conexão com o python da parte física e o outro o servidor simulado no OMNeT++, sendo um ligado ao outro através de uma nuvem, como já mencionado.

5.2. Objetivo da topologia física

Essa topologia será utilizada de modo a testar o xBee trabalhando em conjunto com o OMNeT++, de tal forma podemos aplicar barreiras dentro do sistema físico para analisar o comportamento do sistema elaborado como um todo.

A rede física é importante para demonstrarmos a conexão com o OMNeT++, uma vez que o mesmo não pode simular estruturas muito complexas, é necessário fazer uma ponte entre os dispositivos e o OMNeT++, de forma que o mesmo consiga através dos dados transmitidos, pelos dispositivos físico, fazer as análises necessárias para cada rede e assim verificando o desempenho da rede. As vantagens de utilizar componentes físicos vão desde complexidade e custo benefício, em muitos casos é mais viável desenvolver aspectos físicos de uma rede do que simularmos para fazermos testes nos componentes. Dessa forma a rede física da nossa topologia foi criada para simularmos eventos que seriam mais difíceis de suportar numa simulação por software, tais como a existências de estruturas que impeçam a transmissão dos sinais, efeitos de colisões de pacotes wireless, entre outros.

5.3. Objetivos da rede simulada.

A rede simulada foi criada tanto para mostrar a possibilidade de conexão entre o OMNeT++ e as componentes físicas e recolher resultados simples. Originalmente pretendia-se também possibilitar que os dados que irão ser transmitidos para o OMNeT++ fossem analisados pelo mesmo, permitindo a possibilidade de estudos referentes ao efeito das distâncias nos dispositivos físicos e também como eles se comportam com alguns tipos de interferências impostas dentro do OMNeT++, no entanto este estudo não chegou a ser desenvolvido.

5.4. Dispositivos utilizados.

5.4.1. Arduíno.

O Arduíno Uno foi utilizado como dispositivo final para o desenvolvimento desse trabalho. Ele foi utilizado para suprir as necessidades de pinos analógicos e da mesma forma buscar um baixo custo na implementação da rede. O seu impacto na construção desse trabalho é relevante pois o mesmo é responsável por obter as informações dos sensores e passá-las para o servidor fazer a análise.

5.4.2. xBee

O xBee foi utilizado pois utiliza a tecnologia ZigBee para transmissão de dados, tal fator nos fez acreditar que essa seria uma boa opção para ser utilizado no campo, principalmente que o xBee conta com uma tecnologia *sleep* que permite que ele repouse durante o tempo que não está sendo utilizado, assim economizando bateria e evitando custos.

5.4.3. Raspberry Pi

Foi utilizado pois é o mais próximo de um computador real, possibilitando assim um fator de utilização para programação do código Python, que são usados para transmissão de dados para o OMNeT++. Como vamos ver nos capítulos que mostram o desenvolvimento do Software o Raspberry Pi terá não só a função de administrar os dados recebidos pelos Arduíno, como também a missão de os enviar para o servidor simulado no OMNeT++, desta forma a capacidade computacional do Raspberry Pi vem assegurar a qualidade da topologia proposta.

5.5. Software gerenciador da transmissão.

Para o gerenciamento da transmissão de dados dos dispositivos basicamente utilizamos os dispositivos Arduíno executando códigos Arduíno, como já era o esperado, já para a

recepção desses valores utilizamos código Python dentro do Raspberry Pi, isso para que seja facilitada a recepção das informações, pois o Python possui bibliotecas capazes de receber dados por serial, através de um cabo USB, como temos um Arduíno transmitindo por serial para o Raspberry Pi a comunicação se torna fácil.

Para que a recepção dos dados ocorra de maneira adequada, o Raspberry Pi também será responsável por enviar as informações para o computador executando OMNeT++. Tal é conseguido através de código Python criado para transmitir os dados entregues ao Raspberry Pi (esses dados serão enviados através de socket para o nosso servidor).

Capítulo 6 Sistema no quesito hardware

Como dito anteriormente a topologia como um todo contará com dispositivos físicos e dispositivos simulados, desta forma para a realização da parte física foram escolhidos alguns dispositivos, dentre eles, Arduino Uno 3.0 e a placa xBee.

O funcionamento dos mesmos será explicado com detalhes nessa parte do trabalho. A Figura 14 mostra como foi montada um dos dispositivos utilizados para a transmissão dos dados captados no ambiente.

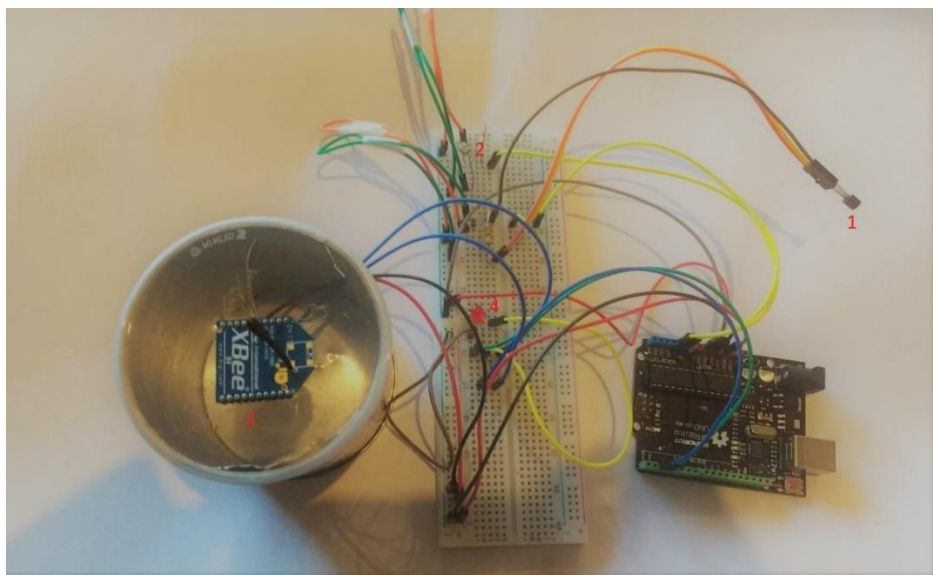


Figura 14 Nó construído para a transmissão de dados.

É possível observar na Figura 14 números em algumas partes específicas dessa imagem, o objetivo desses números é explicar detalhadamente a função de cada objeto utilizado neste dispositivo de transmissão.

No número 1 da imagem podemos verificar um sensor de temperatura. Esse sensor é um TMP 36, ele foi utilizado para captar as temperaturas do ambiente, dessa forma saberemos quantos graus Celsius o local, onde foi instalado os dispositivos, terá. O sensor pode ser visto com mais detalhes na Figura 15 e as definições mais detalhadas do mesmo podem ser verificadas no capítulo de topologia.



Figura 15 TMP 36

Apesar da disposição dos cabos serem diferentes do que foi implementado do nosso sistema, o TMP36 foi instalado utilizando a ideia da Figura 16, a qual podemos utilizar como base para entender seu funcionamento de transmissão dos dados. Na sua extremidade estão os pinos machos que receberão a eletricidade, um estará ligado a saída de 5V e o outro no GND do Arduíno, já o pino macho do meio estará ligado a porta A1 do Arduíno enviando assim os dados captados pelo sensor ao Arduíno.

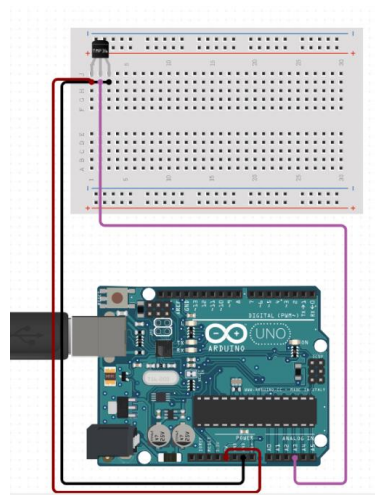


Figura 16 Exemplo de instalação TMP36

Para a construção desse trabalho também foi utilizado um sensor de luz, número 2 da imagem, um LDR para medir a intensidade de luminosidade do ambiente, dessa forma pudemos perceber como o sistema se comportaria em diferentes condições de luminosidade. O sensor utilizado na realização desse trabalho pode ser verificado na Figura 17.



Figura 17 Sensor LDR

Para a construção do circuito integrado ao Arduino montamos um sistema parecido com o da Figura 18, o qual não é idêntico ao construído, mas usa característica semelhantes. No nosso exemplo a leitura dos dados ocorre na porta A0 do Arduino, sendo utilizado uma resistência 10K Ohm dentro do circuito, para estabelecer um divisor resistivo onde a tensão de saída vai ser função da intensidade de luz que incide no sensor LDR.

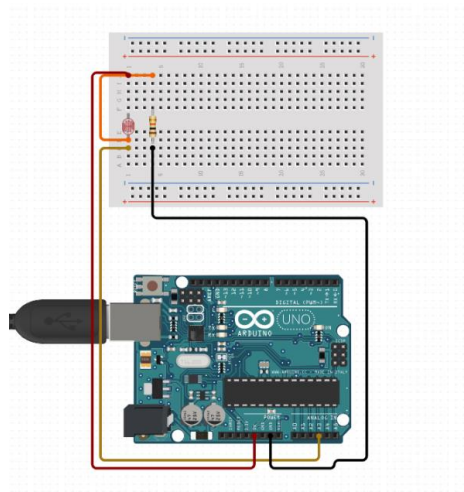


Figura 18 configuração do sensor LDR

No número 3 da Figura 14 podemos observar o xBee, talvez o elemento mais importante da nossa topologia uma vez que o mesmo vai possibilitar a transmissão de dados via radiofrequência nos locais onde forem implementados os dispositivos. Para que o xBee funcionasse adequadamente precisamos configurar algumas informações que irão ser gravadas no mesmo, como por exemplo, qual será a rede que o mesmo irá transmitir, para tais fins utilizamos um programa criado pela própria empresa fornecedora dos xBee, o mesmo se chama XCTU.

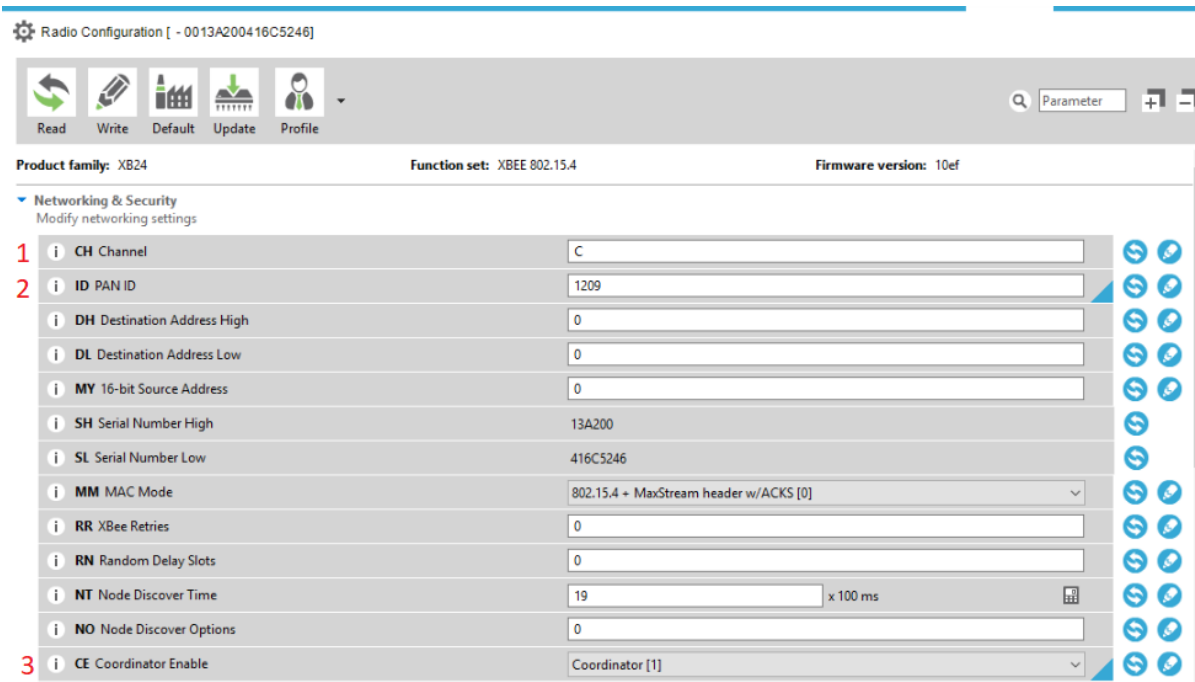


Figura 19 Configuração do xBee

Para os nossos testes configuramos 3 campos no XCTU, conforme a Figura 19, no primeiro campo salientado na imagem configuramos o canal, que será igual a em todos os 3 xBees utilizados, pois o canal define a frequência em que os xBee irão transmitir. Posteriormente configuramos o 2 campo Pan ID que também será o mesmo para todos os xBees, esses campos são utilizados para a troca de mensagens e se algum dos mesmos for diferente não ocorre a transmissão dos dados. O Pan ID pode ser configurado de forma diferente, caso haja a intenção de não transmitir dados. Já o campo número 3 é configurado de forma não análoga, ou seja, para cada tipo de dispositivo teremos uma configuração. Podemos escolher entre 3 configurações. um exemplo é mostrado na Figura 20, o xBee deve ser configurado conforme a necessidade do projeto.

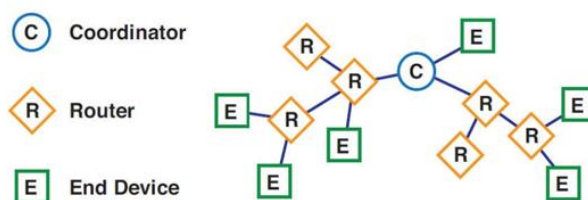


Figura 20 Tipos de configuração do xBee

Para esclarecer mais sobre os modos de configuração do xBee iremos explicar os três modos citados anteriormente.

Iremos começar citando o modo coordenador, o mesmo é responsável por gerir a rede que estaremos configurando, dessa forma para o funcionamento adequado da rede necessitamos de um coordenador; como a Figura 20 mostra, depois disso o xBee configurado como coordenador fica responsável pela verificação e seleção do PAN ID da rede, que será imposto no xBee pelo XCTU, como também o mesmo irá começar a rede verificando os dispositivos que compõe a mesma, além disso é capaz de enviar e receber as informações aos demais nós, como é explicado pela (DigiInternational, 2018)

Em segundo plano, iremos explicar como ocorre o funcionamento do modo roteador, o mesmo tem a função de intermediário, ele deve se conectar a uma rede já estabelecida e posteriormente terá a função de conectar os outros dispositivos ingressantes à rede a topologia total, os xBees configurados como roteador deverão rotear os pacotes, dessa forma os dados serão transportado para topologia total (DigiInternational, 2018).

E por fim os xBee configurados como dispositivos finais. Esses também devem ingressar em uma rede válida para a transmissão de dados, porém o que diferem eles dos routers é a capacidade dos mesmos serem conectados em um ambiente isolado e com uma bateria, dessa forma evitam o gasto de energia utilizando o modo *sleep* do xBee, portanto poupando gastos excessivos de carga, para isso, quando são configurados como dispositivos finais não conseguem identificar outros dispositivos que venham ingressar na rede(DigiInternational, 2018).

Nossa topologia não possui router, pois disponibilizamos os Arduínos no formato estrela, isso aconteceu por que o xBee que utilizamos para a execução da tese não possibilitava a configuração de rede em modo mesh. Sendo nossa ideia original impossível, a distribuição em mesh foi substituída por uma construção em estrela como mostra a Figura 12 do capítulo da topologia.

Para a conexão do Arduíno com o xBee tentamos utilizar a Xbee shield V1.4, porém a mesma não funcionava de maneira adequada, podemos dizer que não funcionava, em nenhum sentido, pois mesmo com o software do fabricante os xBee não eram identificados, até quando utilizávamos todas as configurações da shield. Para contornar os problemas criados pela placa decidimos optar pela conexão direta.

Para fins de conhecimento a Figura 21 mostra a shield que não funcionou em nosso projeto.

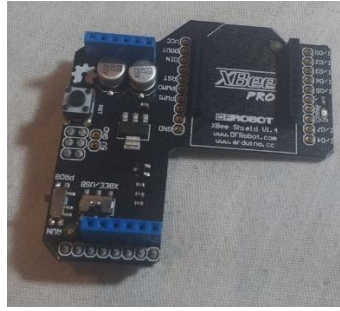


Figura 21 Shield xBee

Tal fator fez com que fizéssemos uma conexão direta nos nossos dispositivos, como pode ser visto na Figura 14, o xbee está sobre uma plataforma que possibilita e conectado diretamente as portas do Arduino.

Para conectarmos o xBee diretamente com o Arduino utilizamos as configurações mostradas na Figura 22, a qual mostra como ocorreu a distribuição dos fios na placa.

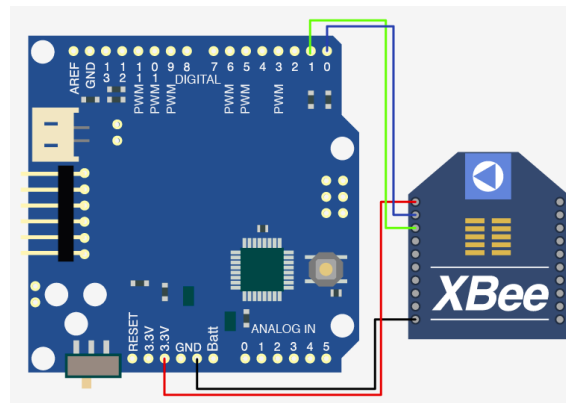


Figura 22 Conexão do xBee com a placa Arduino.

Primeiramente o xBee tem que ser alimentado com energia e isso acontece nos fios vermelho e preto, sendo ligados ao 5 V e no GND, respectivamente, posteriormente temos a entrada e saída de dados, o fio azul representa a entrada de dados na porta RX(0) e o fio verde a saída de dados no TX(1), porém essa distribuição, dos fios de entrada e saída de dados. é necessária para a configuração do xBee na XCTU, todavia isso é modificado e os fios de comunicação de dados serão postos nas portas 2 e 3, quando elaborado o código, isso terá um motivo e será explicado a frente.

Como mencionado no capítulo, os componentes Arduínos iriam ser distribuídos no formato estrela e foi isso que aconteceu. Nesse sentido é necessário um xBee configurado como coordenador, como já explicado. Sendo o coordenador responsável pelo envio e recebimento de dados temos que possibilitar uma ponte entre o Arduino, equipado com o

xBee coordenador, e nossa plataforma receptora dos dados. Tal fator nos fez unir o Arduíno com o Raspberry Pi serialmente.



Figura 23 Conexão serial do Raspberry Pi com Arduíno

A Figura 23 mostra a conexão feita entre o Raspberry Pi e o Arduíno, através do cabo serial (USB) preto mostrado na imagem. Como citado anteriormente, aqui podemos observar porque foi decidido usar as portas 2 e 3 para transmissão e recepção de dados por xBee. Tal ocorreu porque o cabo serial já utiliza a entrada e saída padrão do Arduíno, o que fez com que tivéssemos que modificar o código para a utilização do xBee.

Dessa forma foi padronizado que a utilização em todos os Arduínos para as placas xBee seriam nas portas digitais 2 e 3 do nosso Arduíno. Assim temos a topologia externa ao OMNeT++ apresentada simplificada como mostrado na Figura 24.

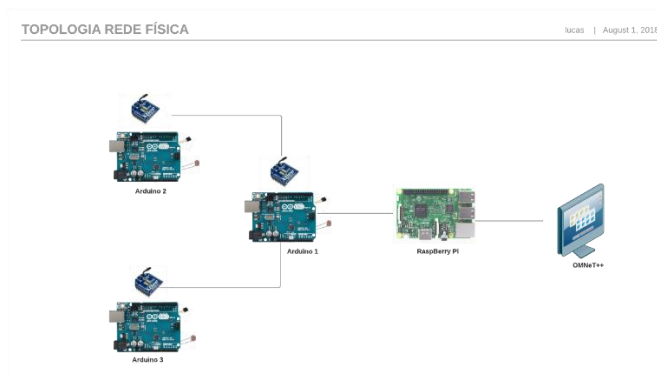


Figura 24 Topologia física

Capítulo 7 Sistema no quesito software

7.1. Introdução

O software foi desenvolvido em camadas, pois contamos com duas topologias que ao fim se tornam uma. Primeiro foi desenvolvido o software para a topologia física onde contamos com 3 Arduínos se comunicando através de xBee, com um dos Arduínos conectado através de um cabo serial com o Raspberry Pi. Dessa forma obedecemos parcialmente ao que é mostrado na Figura 4 (ver capítulo 4), com a adição de um Arduíno adicional ligado serialmente ao Raspberry Pi.

Para que a topologia funcionasse é necessário a parte de codificação dos componentes, por isso será narrado de forma adequada em dois parágrafos abaixo os métodos utilizados para a criação dos códigos para os dispositivos.

No primeiro parágrafo será explicado o funcionamento da parte física e o segundo da parte integrada ao OMNeT++.

7.2. Software para parte física

O software gerenciador deverá trabalhar de formas receber dados dos sensores e transferi-los para os Arduínos, possibilitando que trafeguem pela rede criada. Portanto é necessário a criação de algum algoritmo capaz de dimensionar aquilo que está sendo captado pelos sensores.

Como alguns códigos já estão prontos, utilizamos os mesmos dando os devidos créditos aos seus autores.

7.2.1. Definições dos pacotes

As definições dos pacotes seguirão os padrões estabelecidos na struct dentro dos Arduínos e podem ser vistos na Tabela 3, essa estrutura obedece os padrões estabelecidos anteriormente e seguem, de forma total, o que foi explicado na Tabela 1 e Tabela 2. De tal forma os campos dessa estrutura serão modificados no decorrer do código. O código está em sua íntegra nos anexos desse documento.

Tabela 3 Struct definida no Arduíno.

```

1.  typedef struct package{
2.      String ID;
3.      unsigned long int count;
4.      char IO;
5.      char sI1;
6.      int vL;
7.      char sI2;
8.      int vT;
9.      char endof;
10. }package;
```

Algumas funções do código construído, usado no Arduíno, utilizam essa estrutura pelo papel fundamental dentro do codificação. Ela mereceu ser destacada neste capítulo, pois será extremamente importante para o entendimento de várias partes do código.

7.2.2. Sensor de luminosidade

Os sensores de luminosidade enviaram as informações retiradas do ambiente para o Arduíno, logo é indispensável a utilização de uma função para receber esses dados. Dessa

forma utilizamos o código disponível por (Mota, 2017) para a realização do nosso trabalho, o mesmo disponibiliza informações adicionais sobre o funcionamento do LDR.

O código pode ser verificado na Tabela 4.

Tabela 4 Trecho de código para leitura do LDR

```
1. int takeDataSensorLDR(package *pack){
2. int ldrValue = analogRead(ldrPin); //O valor lido será
   entre 0 e 1023;
3. pack->vL = ldrValue;
4. return 1;
5. }
```

7.2.3. Sensor de temperatura

Para o recebimento dos dados produzidos pelo sensor de temperatura é necessário ajustar os dados produzidos pelos sensores com algumas fórmulas matemáticas, foi isso que (Ada, 2017) fez produzindo o código (Tabela 5) utilizado nessa dissertação.

Tabela 5 Codificação do sensor de temperatura

```
1. int takeDataSensorTMP(package *pack){
2. int tmpValue = analogRead(tmpPin);
3. float voltage = tmpValue * 5.0;
4. voltage = voltage / 1024.0;
5. pack->vT = ((voltage - 0.5) * 10)/2 ;
6. return 1;
7. }
```

7.2.4. Explicação das atribuições das funções escritas para os Arduínos

Para a transmissão de dados de um Arduino ao outro, foram necessárias cerca de 600 linhas de código, dessa forma fica quase impossível explicar todo o conteúdo respeitando o limite imposto para essa redação. Dessa forma explicaremos os pontos principais da codificação.

7.2.1.1. Transmissão por xBee

Vale ressaltar que a comunicação é feita de um Arduino ao outro utilizando apenas o xBee, ocorre uma exceção quando a transmissão é feita do Arduino ao Raspberry Pi.

Para a transmissão de dados entre um Arduino e outro o xBee utiliza a transmissão serial, padrão, do Arduino. Como o capítulo de Hardware deixa especificado esse padrão foi modificado em nosso projeto, especificamente para o xBee. Para a elaboração disso utilizamos uma biblioteca que nos permite a abertura de um caminho serial, segue a ideia na Tabela 6

Tabela 6 Serial para xBee

```
#include <SoftwareSerial.h>

SoftwareSerial XBee (3,2);
```

Dessa forma conseguimos projetar as portas de transmissão e recepção de dados para outras. No nosso caso utilizamos as portas digitais 3 e 2.

Isso é extremamente importante, pois o Arduino coordenador terá duas entradas seriais, uma para o xBee e outra para a ponte com o Raspberry Pi.

7.2.1.2. Definições

As mensagens estão padronizadas de forma que cada campo da mensagem é separado por uma vírgula. Essa vírgula serve para que seja possível uma quebra dos campos, os quais serão verificados para a produção de alguma ação dentro do código. Sendo assim é importante que as mensagens produzidas sejam distribuídas com integridade.

A ideia inicial era utilizar um *Checksum*, porém esse método foi reavaliado por problemas decorrentes ao gerenciamento do xBee na suas configurações de modo API. O problema ocorreu porque mesmo implementando todas as etapas citadas no (DigiInternational, 2018), como parâmetros de início e fim da mensagem, o xBee não executava nenhuma ação de resposta, nem quando testado no software do fabricante. Logo por essa razão decidimos optar em utilizar o modo AT de transmissão de dados, apesar de não ser o mais indicado.

O modo AT não foi desenvolvido com o propósito de grandes redes, mas foi uma forma de contornar o problema. Os testes em modo AT, podem são de utilidade, pois trechos das redes são construídas usando o modo AT pela facilidade de compreensão humana.

7.2.1.3. Resumo das funções utilizadas nos Arduínos

Iremos comentar sobre as funções secundárias, porém não menos importantes, pois são essenciais para a boa execução do código. São elas:

1. `int str_union_xbee(String * data);`

Responsável pela união dos caracteres que serão transmitidos pelo xBee, transformando-os em strings.

2. `int str_union(String * data);`

Responsável pela união de caracteres, porém o corpo dessa função é escrita especialmente para tratar a união serial entre o Arduíno e o Raspberry Pi

3. `String getValue(String data, char separator, int index);`

Responsável por executar o *split*. Ela foi criada com o objetivo de separar os campos das mensagens de forma que cada campo seja único.

4. `int howMany(String data, char count);`

Mostra a quantidade dentro da string de algum elemento especificado em seus parâmetros.

Esses são os protótipos feitos dentro do próprio código, executado nos Arduínos. Essas funções são utilizadas pelo autor do texto para auxiliar na construção e manutenção das mensagens transmitidas ao longo do código.

7.2.5. Conexão entre um Arduino com os outros.

O Arduino coordenador é quem vai definir as diretrizes e comportamentos dos outros ligados a rede. Ou seja, com o auxílio desse Arduino vamos dar comandos que deverão ser obedecidos pelos demais Arduínos. Para executar isso projetamos uma espécie de semáforo, o mesmo foi necessário, pois durante os testes os xBees paravam de transmitir quando acontecia alguma colisão.

Esse semáforo funciona como um bate e volta, um laço *while* será desbloqueado no Arduino dispositivo final quando o coordenador solicitar. A Figura 25 mostra como ocorre o desbloqueio dos Arduínos.

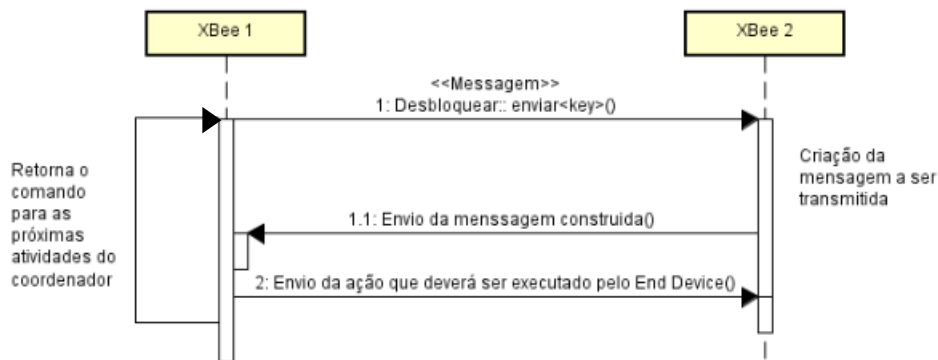


Figura 25 Desbloqueio dos Arduínos via Xbee

Se acontecer da transmissão parar durante essa tarefa, foi colocado dentro do código uma espécie de *thread*, que é capaz de parar a execução caso isso ocorra.

7.2.6. Raspberry Pi

O Raspberry será um dispositivo que irá intermediar os Arduínos e a topologia simulada no OMNeT++. Ele fará isso utilizando um programa feito em Python que implementa uma biblioteca “serial” para a conexão via serial e uma biblioteca Python “socket” para a transmissão via protocolos TCP e UDP.

Dessa forma o código Python espera os comandos feitos pelo coordenador e transmite os mesmos para o OMNeT++ fazer as devidas configurações nas mensagens.

O código Python possui duas funções importantes para executar a conexão com o OMNeT++.

A primeira é a de envio, essa é responsável por pegar os dados captados pelo coordenador e enviar para o OMNeT++, essa função é executada por uma thread, dessa forma o programa se torna independente da função de envio, evitando que as tarefas parem de ser executadas.

A segunda é a função de recebimento dos dados do OMNeT++, essa é utilizada para receber as ações referentes a cada Arduíno e então enviar as ações respectivas a cada Arduíno, para serem ativados os atuadores. Ela também é executada por uma *thread*.

Para o coordenador informar o Raspberry Pi sobre a existência de uma mensagem o mesmo deve enviar “0,#” como conteúdo serial, dessa forma ocorre a liberação das linhas seguintes.

O código poderá ser verificado nos anexos dessa dissertação.

7.3. Software para a parte simulada

Na parte simulada modificamos um exemplo fornecido pelo próprio OMNeT++ durante sua instalação. A conexão foi efetivamente feita utilizando o protocolo TCP, porém o protocolo UDP também foi implementado no algoritmo que foi escrito. Essa parte foi uma das mais problemáticas do projeto pelas dificuldades sentidas pelo autor deste documento.

Para a construção da conexão foi re-implementado as funções com nome *SetInterface* e *receiveWithTimeout* na biblioteca *SocketRTScheduler.h*, possibilitando o HiL. Por isso temos a conexão com os dispositivos físicos executando adequadamente.

A função *SetInterfaceModule* é a nova implementação criada por nós de forma a possibilitar a conectividade, ela possui os atributos que devem ser transmitido dentro do OMNeT++, quanto informações relevantes a conexão TCP.

Tabela 7 Função Set interface

```
1. void cSocketRTScheduler::setInterfaceModule(cModule *mod,
    cMessage *notifMsg, char *buf, int bufSize, int
    *nBytesPtr){
```

```

2. if (module)

3. throw      cRuntimeError      (      "cSocketRTScheduler:
    setInterfaceModule() already called" );

4. if (!mod || !notifMsg || !buf || !bufSize || !nBytesPtr)

5. throw      cRuntimeError      (      "cSocketRTScheduler:
    setInterfaceModule(): arguments must

6. be non-nullptr" );

7. module = mod;

8. notificationMsg = notifMsg;

9. recvBuffer = buf;

10.    recvBufferSize = bufSize;

11.    numBytesPtr = nBytesPtr;

12.    *numBytesPtr = 0;

13.    connected = true;

14.    }

```

Depois de ter estabelecido a conexão partimos para o método de análise das informações, dessa forma construímos uma classe que realiza o serviço de análise dos dados, como também re-implementamos o servidor da topologia que exemplificava o HiL.

Na re-codificação do servidor a função mais importante é a responsável pelo estudo dos campos das mensagens para produção da resposta. Essa função foi trabalhada como se houvesse um pseudo banco de dados e pode ser vista na Tabela 10 em anexo na página 81.

Já a classe para recolher os dados para análise estatística está a disposição nos anexos e se chama *LostPack*.

Capítulo 8 Dados obtidos e resultados

8.1. Introdução

Esse capítulo foi construído com o objetivo de mostrar os resultados obtidos na transmissão dos dados entre os dispositivos simulados e os dispositivos físicos.

Possuímos 3 Arduínos para a criação de ambientes físicos, como também o simulador OMNeT++ para produção de testes. Para os testes da topologia, tal como a análise dos dados, em relação a perda das mensagens, foram criados ambientes controlados de simulação, dentro do ambiente real e do simulado.

Foram analisados a taxa de perda para a verificação do modo AT no xBee, dessa forma avaliar se esse modo é um componente viável ou não para aplicação. Devido a alguns problemas encontrados durante a produção desse trabalho, os testes foram limitados no tempo, o que impediu a sua validação estatística. Como já mencionado implementar a conexão dos dispositivos físicos e o OMNeT++ demorou um tempo relativamente grande, em conjunto com os desafios enfrentados pela existência de maus contactos com o xBee, dado o não funcionamento da shield.

Para todos os testes feito foi calculada a perda de pacotes durante a execução do código como um todo, e assim podemos identificar as condições de execução do modo AT, dessa forma saber se é viável utilizar o modo AT para transmissão.

O teste dos códigos foram construídos simulando um funcionamento normal da rede do OMNeT++, ou seja, sem projetar condições anormais no simulador de software, somente foi projetado problemas no ambiente.

Para as definições desses testes definimos um conjunto de condições:

Cada dispositivo terá um nome específico, dessa forma conseguimos calcular a taxa de perda para cada um.

Os gráficos que serão mostrados abaixo indicando a quantidade de perda de mensagem para cada dispositivo implementado na rede, de forma que o eixo horizontal do gráfico é o número de vezes que não ocorreu a transmissão de um pacote entre as mensagens e o eixo vertical é a percentagem de ocorrência dos mesmos.

Para exemplificar se recebermos uma mensagem com o número 1 no campo referente a a contagem de mensagens e posteriormente recebermos o número 2 no próximo pacote que chegou ao OMNeT++, a perda de pacotes se totaliza como zero, logo a coluna com o zero ganhará mais um número. Diferentemente se ocorrer de chegar um número 3 e número 5, no qual ocorreu a perda de um pacote, portanto teremos o acréscimo de mais um na coluna 1 e assim sucessivamente.

8.1.1. Primeiro Teste

Em relação a ambientação física, foi distribuído os dispositivos durante o período da tarde, até o anoitecer da mesma noite. Portanto os dispositivos receberam luz do sol em metade do teste e no outro não.

Também foi colocado uma barreira plástica no dispositivo nomeado “AR2” com um pano preto, por isso, o mesmo, possui características noturnas em todo tempo de simulação, e pode ser verificado na Figura 27, porém o pano não foi evidenciado, pela qualidade da foto. Outros testes utilizaram a mesma ideia, por isso só irá ser destacado no decorrer do testes que foi utilizado.

Os dispositivos foram distribuídos como pode ser visto na Figura 26 e o OMNeT++ recebeu os dados da Tabela 12 nos anexos desse trabalho. As definições dos dispositivos estão definidas por tabelas em cada um dos testes, desta forma a Figura 26 é meramente figurativa e não deve ser considerada fidedigna com esse ou outros testes.

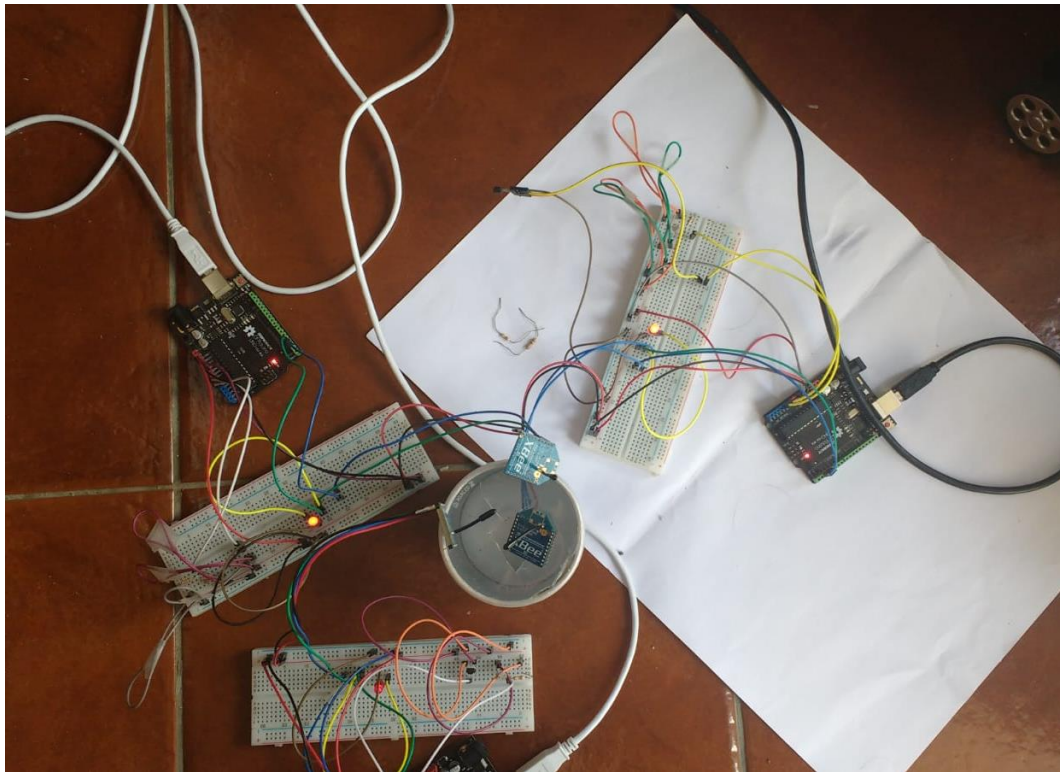


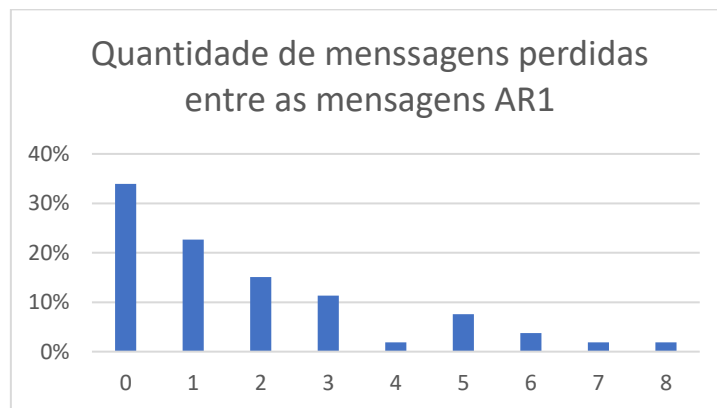
Figura 26 Dispositivos montados para a transmissão



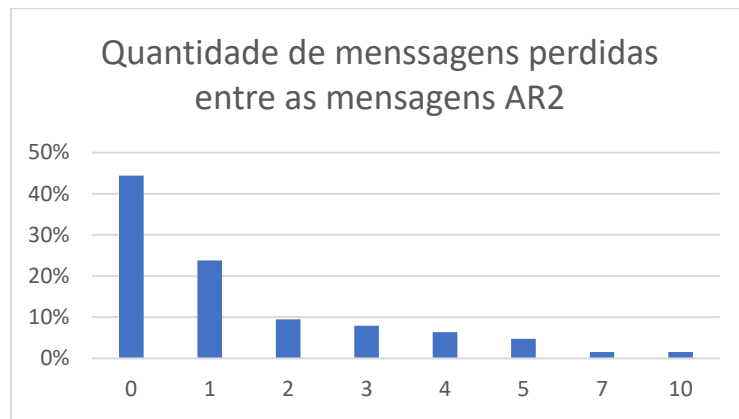
Figura 27 Dispositivo com barreira

Tabela 8 Condições do teste 1

	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 10 cm do Arduíno 2 e 1 metro do Arduíno coordenador.	O dispositivo ficou a uma distância de 10 cm em relação ao Arduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 4 horas		
Antena:	Sem nenhum tipo de bloqueio induzido.	Bloqueio feito com uma caixa plástica coberta por pano.	Sem nenhum tipo de bloqueio induzido.
Testado:	Taxa de perda de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	No início claro, penumbrando até o período da noite quando ocorre uma escuridão total.		
Inserção de algum dado anormal ao sistema:	Não.		

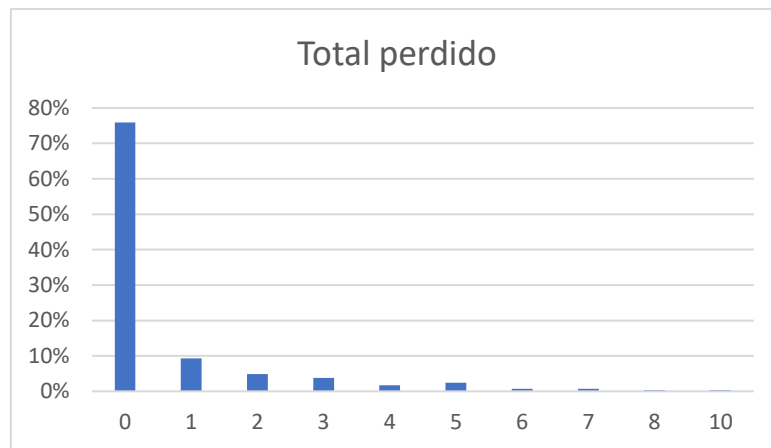


Teste 1a: Pacotes perdidos entre mensagens no Arduíno 1



Teste 2a: Pacotes perdidos entre mensagens no Arduíno 2

No respeitante ao Arduíno coordenador, optamos por não representar o gráfico, pois o mesmo apresentou recepção de todas as mensagens no servidor OMNeT++.



Teste 3a: Pacotes perdidos entre mensagens total

Dessa forma segue o total de mensagens recebidas e perdidas durante esse teste.

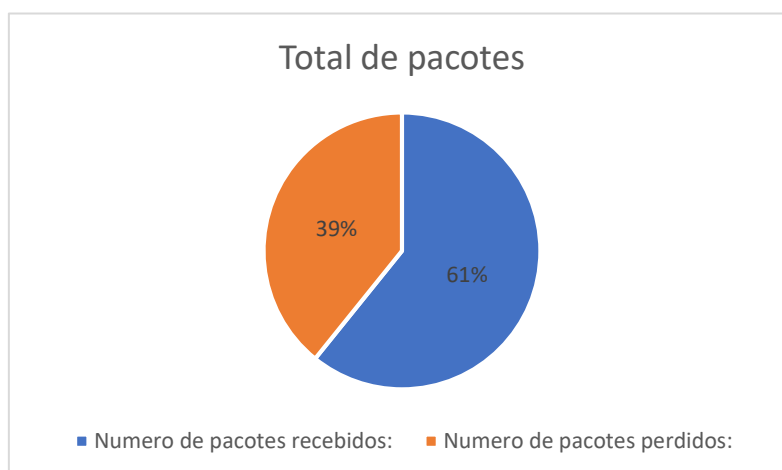


Gráfico 1: Pacotes perdidos entre mensagens valor total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 1 é possível ver os dados no Anexo teste 1.

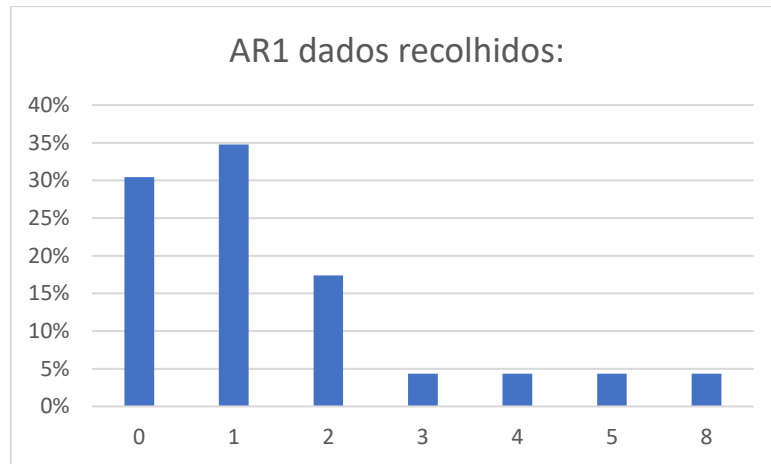
8.1.2. Segundo Teste

Como no primeiro teste inferimos uma barreira para a transmissão de dados. Fizemos um utilizando as condições normal de teste, para verificar a execução dos dispositivos no ambiente sem interferências, porém com um cenário parecido com o anterior, portanto fizemos o experimento utilizando as características da Tabela 9.

Tabela 9 Condição do teste 2

	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 1 metro do Arduíno 2 e 1 metro do Arduíno coordenador	O dispositivo ficou a uma distância de 1 metro em relação ao Aduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 2 horas		
Antena:	Sem nenhum tipo de bloqueio induzido.	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido.
Testado:	Taxa de perca de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	Todo o tempo baixa luminosidade.		
Inserção de algum dado anormal ao sistema:	Não.		

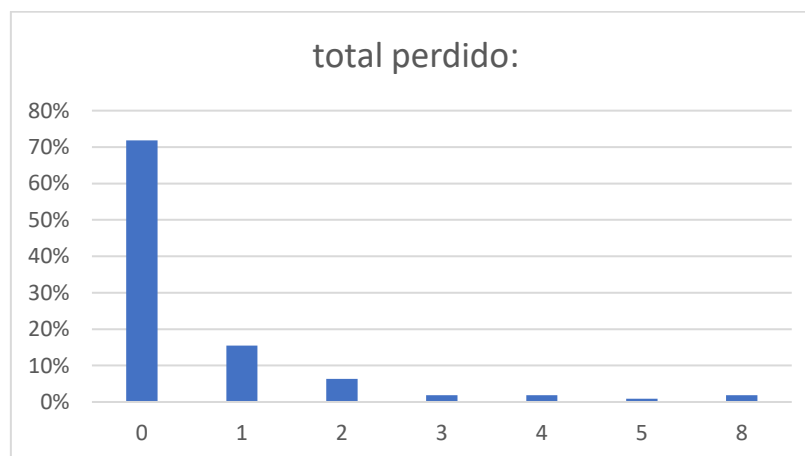
Os gráficos como já mencionado na descrição do teste 1 explicam a taxa de perda de pacote entre uma mensagem e outra.



Teste 4b: Pacotes perdidos entre mensagens no Arduino 1



Teste 5b: Ocorrência de mensagens perdidas entre mensagem Arduino 2



Teste 6b: Pacotes perdidos entre mensagens total

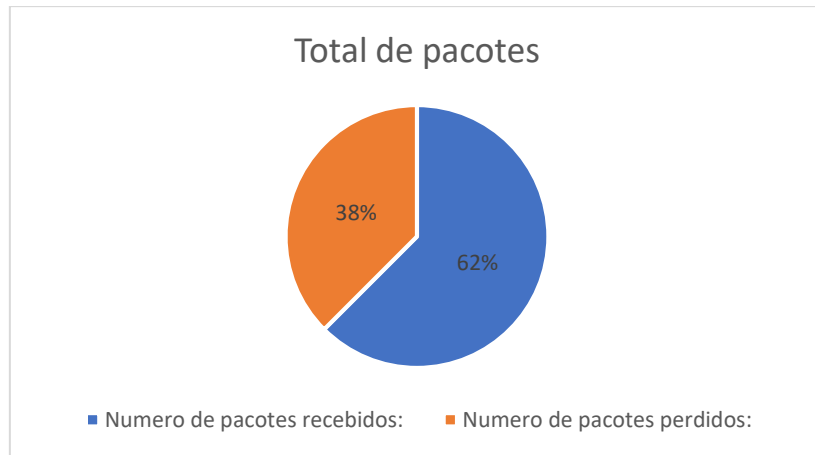


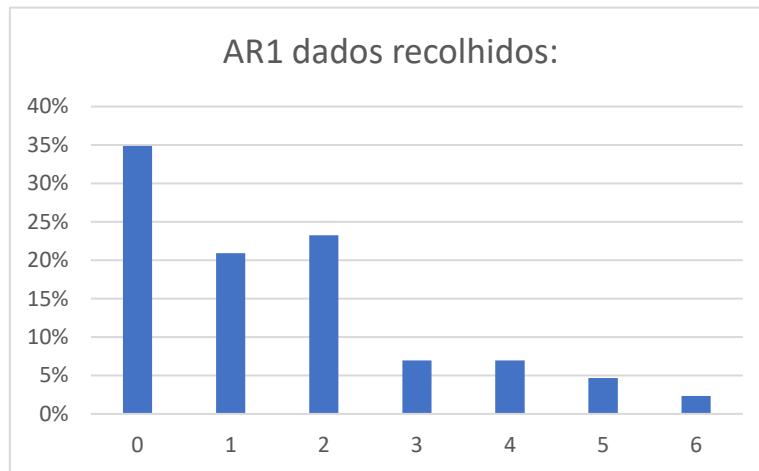
Gráfico 2: Pacotes perdidos entre mensagens valor total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 2 é possível ver os dados no Anexo teste 2.

8.1.3. Terceiro teste

Esse teste foi feito tentando explorar o Arduino 1, já que no primeiro fizemos um teste focando o Arduino intitulado com o número 2.

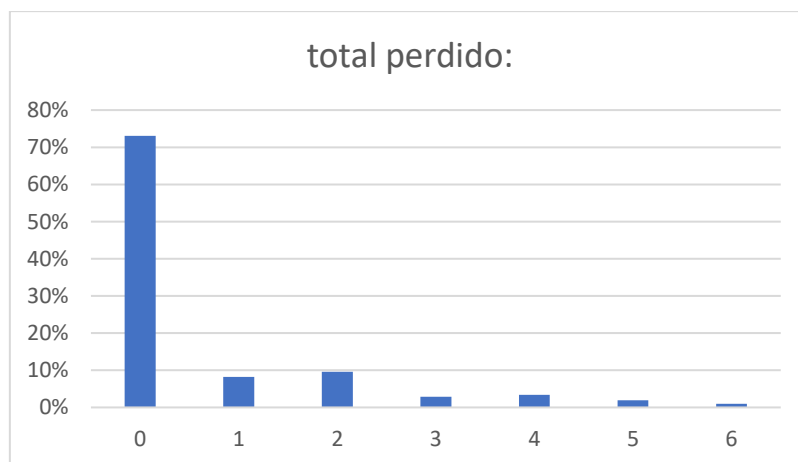
	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 10 cm do Arduino 2 e 1 metro do Arduino coordenador.	O dispositivo ficou a uma distância de 10 cm em relação ao Aduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 3 horas		
Antena:	Bloqueio feito com uma caixa plástica coberta por pano.	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido.
Testado:	Taxa de perca de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	Teste sobre influência de iluminação sintética.		
Inserção de algum dado anormal ao sistema:	Não.		



Teste 7c: Pacotes perdidos entre mensagens no Arduíno 1



Teste 8c: Pacotes perdidos entre mensagens no Arduíno 2



Teste 9c: Pacotes perdidos entre mensagens total

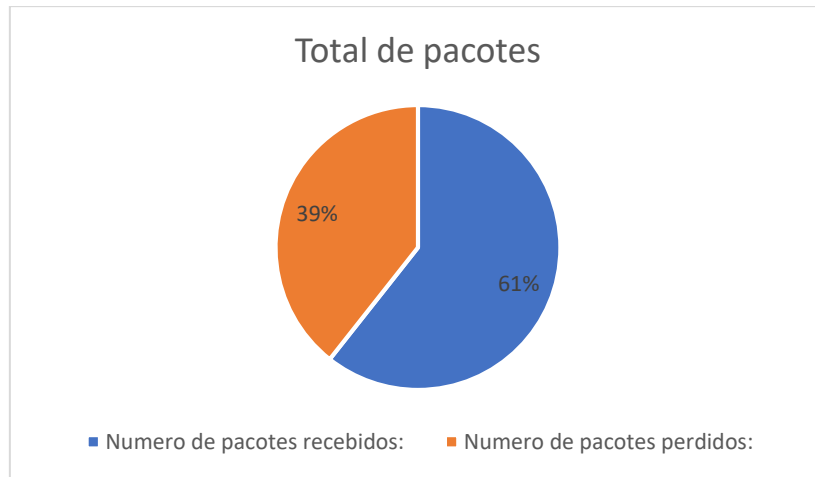


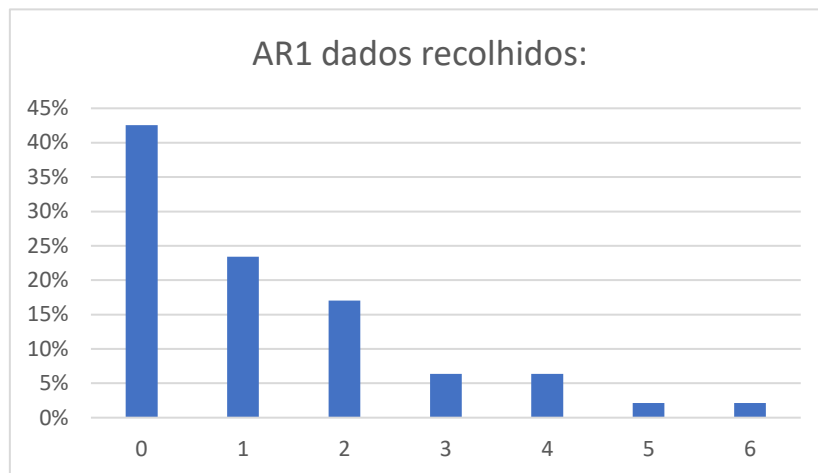
Gráfico 3: Pacotes perdidos entre mensagens valor total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 3 é possível ver os dados no Anexo teste 3.

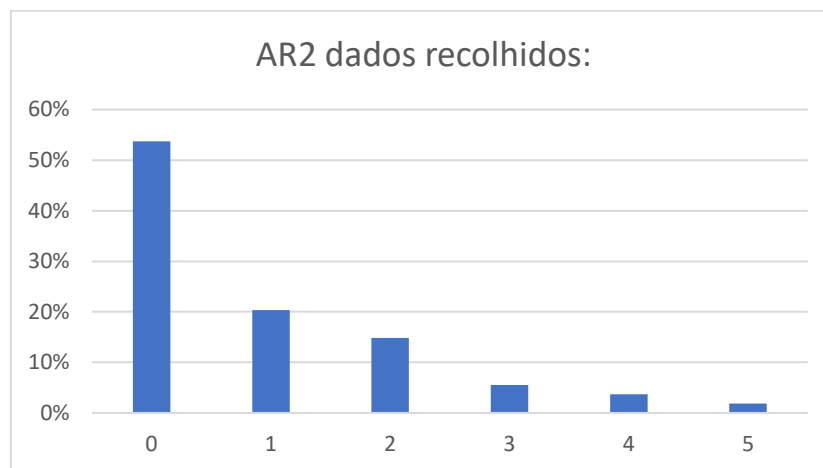
8.1.4. Quarto teste

O quarto teste foi feito colocando impedimento no Arduino coordenador.

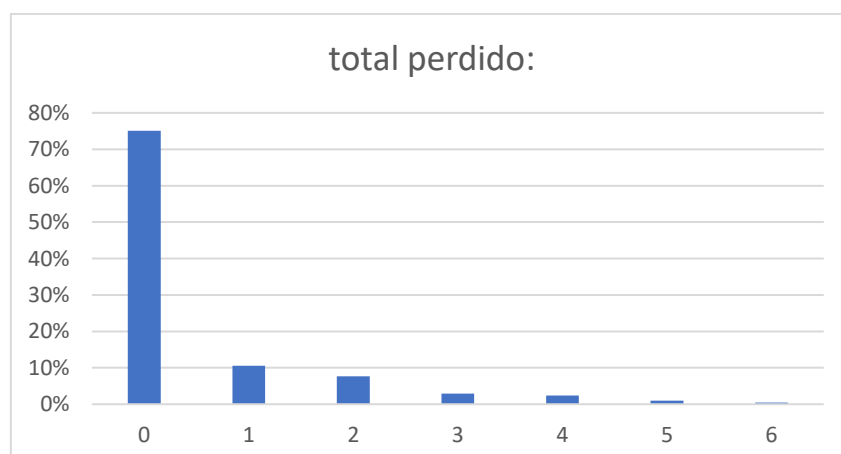
	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 1 metro do Arduino 2 e 1 metro do Arduino coordenador	O dispositivo ficou a uma distância de 1 metro em relação ao Aduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 3 horas		
Antena:	Sem nenhum tipo de bloqueio induzido.	Sem nenhum tipo de bloqueio induzido	Bloqueio feito com uma caixa plástica coberta por pano.
Testado:	Taxa de perca de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	Teste sobre influência de iluminação sintética.		
Inserção de algum dado anormal ao sistema:	Não.		



Teste 10d: Pacotes perdidos entre mensagens no Arduíno 1



Teste 11d: Pacotes perdidos entre mensagens no Arduíno 2



Teste 12d: Pacotes perdidos entre mensagens total

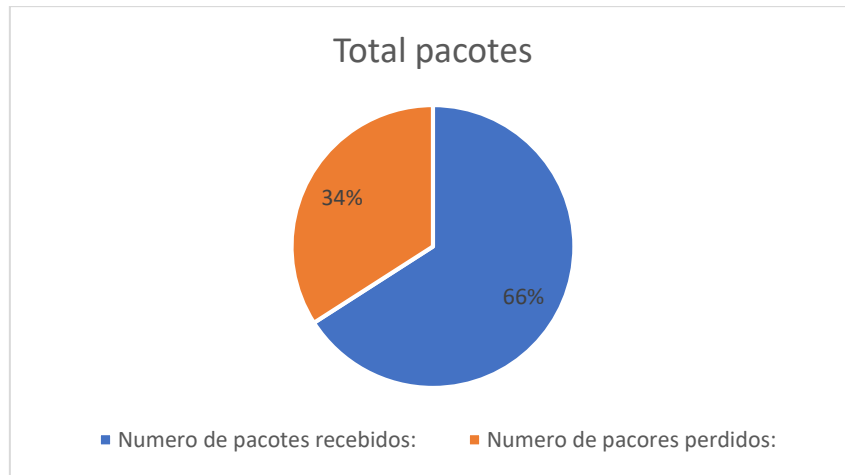


Gráfico 4: Pacotes perdidos entre mensagens valor total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 4 é possível ver os dados no Anexo teste 4.

8.1.5. Quinto teste

Esse teste foi feito no modo normal de execução do xBee.

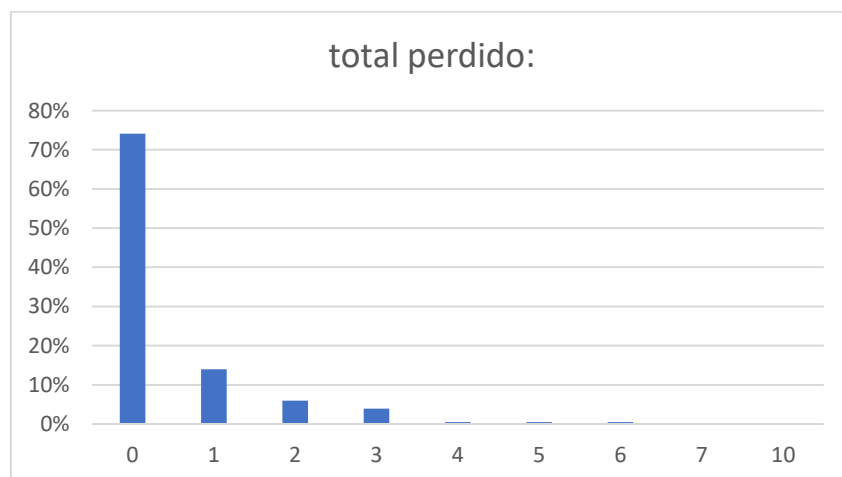
	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 10 cm do Arduíno 2 e 1 metro do Arduíno coordenador.	O dispositivo ficou a uma distância de 10 cm em relação ao Arduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 4 horas		
Antena:	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido
Testado:	Taxa de perda de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	Teste sob influência de iluminação sintética.		
Inserção de algum dado anormal ao sistema:	Não.		



Teste 13e: Pacotes perdidos entre mensagens no Arduíno 1



Teste 14e: Pacotes perdidos entre mensagens no Arduíno 2



Teste 15e: Pacotes perdidos entre mensagens total

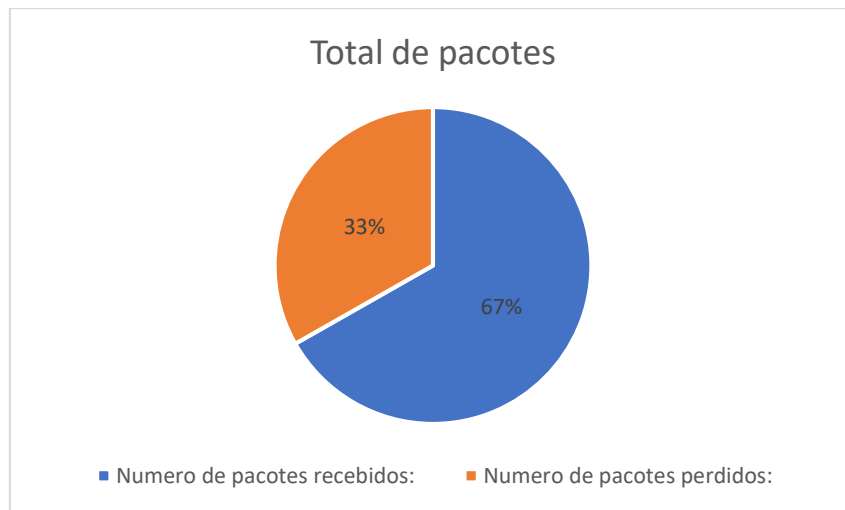


Gráfico 5: Pacotes perdidos entre mensagens valor total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 5 é possível ver os dados no Anexo teste 5.

8.1.6. Sexto teste

Esse teste foi criado de forma a colocar as antenas mais próximas o possível sem bloqueio.

	ARDUÍNO 1 (AR1)	ARDUÍNO 2 (AR2)	ARDUÍNO COORDENADOR (ARC)
Definição:	O dispositivo ficou a uma distância de 10 cm do Arduíno 2 e 1 metro do Arduíno coordenador.	O dispositivo ficou a uma distância de 10 cm em relação ao Arduíno 1 e 1 metro em relação ao coordenador	O dispositivo ficou a 1 metro de todos os outros dispositivos.
Duração:	Duração de 2:30 horas		
Antena:	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido	Sem nenhum tipo de bloqueio induzido
Testado:	Taxa de perda de pacotes.		
Temperatura:	Se manteve igual em todos os testes		
Luminosidade:	Teste sob influência de iluminação sintética.		
Inserção de algum dado anormal ao sistema:	Não.		

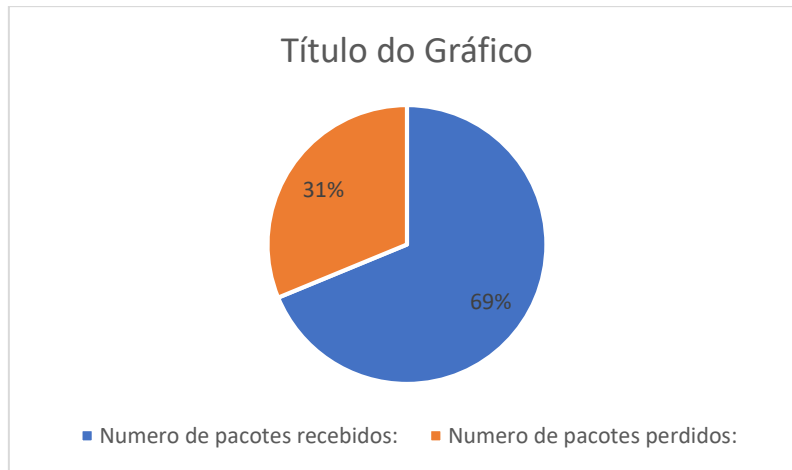
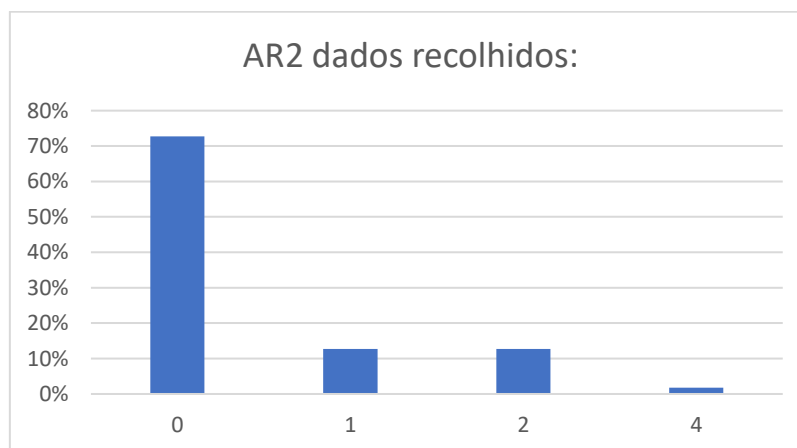


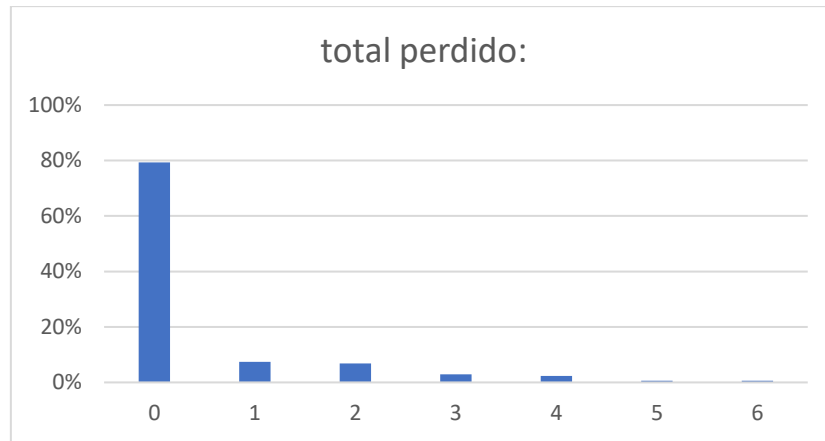
Gráfico 6: Pacotes perdidos entre mensagens valor total



Teste 16f: Pacotes perdidos entre mensagens no Arduino 1



Teste 17f: Pacotes perdidos entre mensagens no Arduino 2



Teste 18f: Pacotes perdidos entre mensagens total

Os dados obtidos para geração dos gráficos são apresentados nos anexos em formato de tabela. Para o teste 6 é possível ver os dados no Anexo teste 6.

8.2. Análise dos dados

Podemos perceber com os dados recolhidos que a média de ocorrência de transmissão, analisando somente os dados totais, é cerca de 64,4% de todos os pacotes enviados, isso é um pouco mais do que a metade. Em relação ao tempo de envio tivemos cerca de 1,20 segundos para a transmissão de cada pacote. Acreditamos que as ineficiências possam ter sido devidas tanto ao mau contato dos componentes, como à ineficiência do código projetado

Se observamos o tempo decorrente da simulação e compararmos com a duração dos testes, iremos perceber que as mensagens demoram muito para chegar ao destino, dessa forma não podemos executar em um ambiente prático IoT.

É importante dizer que para fins de testes da conexão com o OMNeT++ que mesmo a transmissão não sendo a melhor possível, a comunicação funciona com sucesso.

No respeitante aos Arduino, porém, se cada dispositivo for analisado separadamente, os resultados em média não são significativamente melhores:

8.2.1. Arduíno 1

Para o Arduíno 1 temos a recepção de 52% dos pacotes enviados, nesse cenário é percebido que quase metade dos pacotes enviados são efetivamente guardados pelo OMNeT++.

8.2.2. Arduíno 2

Para o Arduíno dois os resultados das médias são melhores pois possuímos cerca de 57,5% dos pacotes guardados no OMNeT++, dessa forma temos quase 60% dos pacotes sendo recebidos

8.2.3. Arduíno coordenador

Esse Arduíno tem a maior taxa de transmissão para o servidor simulado, nesse cenário cerca de 100% dos pacotes transmitidos são guardados pelo servidores simulado. Isso ocorre porque o dispositivo durante toda a simulação encontra-se conectado com o Raspberry Pi utilizando um cabo serial, logo não apresenta a perda de pacotes motivada pelo xBee.

Capítulo 9 Conclusão e Trabalhos futuros

9.1. Conclusão

Nesse trabalho podemos ver a integração dos dispositivos reais com o software de simulação OMNeT++, construindo assim o que chamamos de uma co-simulação. Nesse trabalho foi mostrado um exemplo de conexão do OMNeT++ com hardware real. Como já discutido durante a execução do trabalho essa união é muito importante quando não podemos simular o ambiente como um todo, devido várias problemáticas que vão desde o custo à complexidade dos sistemas.

Esse trabalho apresenta como diferencial o fato de proporcionar uma abertura na utilização do OMNeT++ em futuras construções de topologias, aumentando a eficiência dos testes propostos em ambientes complexos. Isso acontece porque o OMNeT++ proporciona a construção de módulos reutilizáveis de fácil construção e manutenção.

Os módulos são criados em C++ o que facilita a utilização do simulador. Utilizar o OMNeT++ em co-simulação é muito vantajoso pois possibilita a integração entre o hardware e o software.

Nos trabalhos de investigação citados na construção desse trabalho foi possível aprender os conceitos básicos sobre a co-simulação, embora nos mesmos sejam utilizados outros softwares para suporte à co-simulação.

A construção dessa dissertação traz como benefício a integração do OMNeT++ com hardware real num software modular. O OMNeT++ é grátis, vantagem inegável perante

outro software alternativo para este fim, como Matlab, que na sua maioria é complexo e não gratuito.

Esse trabalho demonstra através de testes a perda de pacotes devida ao xBee (por causa da utilização do modo AT na simulação), e como isso afeta o hardware real.

Importante salientar que um objetivo primário da utilização do OMNeT++ foi sabermos se era simples na prática a integração com hardware em simulações, o que foi constatado que foi possível conseguir. No entanto deve ser referido que sendo OMNeT++ uma ferramenta poderosa, porém é muito complexo encontrar material fora do site do próprio software. Desta forma foi uma tarefa complexa achar algum artigo que desse suporte na criação de co-simulação.

9.2. Análise dos resultados

Os resultados apresentaram uma perda significativa, de cerca de 40% dos pacotes recebidos, devido à ineficiência para a transmissão de pacotes do xBee no modo AT. Pelas análises do autor, isso pode ter ocorrido porque o xBee não possui mecanismos de gestão de tráfego em modo AT quando necessita de ler e escrever dados de várias fontes simultaneamente. Para contornar os erros derivados disso foi implementado um semáforo, que talvez não tenha sido a solução mais adequada a esse problema.

Levando em conta os aparelhos para a conexão do xBee com o Arduino, que demonstraram tiveram problemas no seu funcionamento e a alta perda de pacotes, faz com que suponhamos que o xBee não é eficiente se for utilizado em modo AT com a implementação construída.

Nosso objetivo no trabalho era executar uma conexão entre o OMNeT++ e um Hardware físico, nesse aspecto o OMNeT++ realizou a tarefa com sucesso, uma vez que apesar dos dados de transmissão sem fios não serem eficientes, o OMNeT++ distribuiu os pacotes TCP com sucesso na execução dos testes.

Portanto a criação de co-simulação utilizando o OMNeT++ é possível e pode ser utilizada em casos de necessidade de testes utilizando dispositivos físicos e simulados.

9.3. Trabalhos futuros

Em trabalhos futuros seria possível utilizar a ponte criada, para transmissão TCP entre OMNeT++ e hardware não simulado, para a construção de simulações mais densas, bem como em ambientes mais próximos da realidade.

Se neste trabalho se utilizar o xBee, deverá ser experimentado um código em modo API que facilite a conexão dos dispositivos e assim melhorando o desempenho do projeto. Se a execução em modo API for mais eficiente que a construção em modo AT, poderá ser criada uma malha mais densa, onde poderemos usar xBee na função de roteador, dispositivo final e coordenador.

Bibliografia

- Ada, Lady. (2017). Using a Temp Sensor. Retrieved May 17, 2018, from <https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor>
- Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805. <https://doi.org/10.1016/j.comnet.2010.05.010>
- Batista, I. de L., Salvi, R. F., & Lucas, L. B. (2011). *MODELOS CIENTÍFICOS E SUAS RELAÇÕES COM A EPISTEMOLOGIA DA CIÊNCIA E A EDUCAÇÃO CIENTÍFICA*. Atas do ENPEC. Campinas.
- Cisco. (2016). Introduction to the Internet of Things. Retrieved November 5, 2017, from <https://static-course-assets.s3.amazonaws.com/I2IoT13/en/index.html#0>
- Costa, L. F. S., & Almeida, L. M. P. de. (2015). *Aspectos de sincronização em co-simulação de equipas de robôs*. Porto.
- DigiInternational. (2018). ZigBee RF Modules User Guide. *Digi International*, 195.
- Faludi, R. (2010). *A Practical Guide to the ZigBee Mesh Networking Protocol. Building Wireless Sensor Networks*. O’ Reilly. <https://doi.org/10.1007/s13398-014-0173-7.2>
- Jacobson, S. H., Hall, S. N., & Swisher, J. R. (2006). Discrete-Event Simulation of Health Care Systems. In *International Series in Operations Research and Management Science* (Vol. 206, pp. 211–252). https://doi.org/10.1007/978-0-387-33636-7_8
- Kopetz, H. (1999). Cluster Simulation in Time-Triggered Real-Time Systems. *Informatik, T*, (9025658), 150.
- Mota, A. (2017). SENSOR DE LUZ COM LDR. Retrieved May 17, 2018, from <https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/>
- Smith, T. (2001). https://www.theregister.co.uk/2001/10/31/hacker_jailed_for_revenge_sewage/. *Theregister*.

- Steinbrink, C., Lehnhoff, S., Rohjans, S., Strasser, T. I., Widl, E., Moyo, C., ... Degefa, M. Z. (2017). Simulation-based validation of smart grids – Status quo and future research trends. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10444 LNAI, 171–185. https://doi.org/10.1007/978-3-319-64635-0_13
- Wortmann, F., & Flüchter, K. (2015). Internet of Things: Technology and Value Added. *Business and Information Systems Engineering*, 57(3), 221–224. <https://doi.org/10.1007/s12599-015-0383-3>

Anexos

Anexo teste 1

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	33,96%	18
1	22,64%	12
2	15,09%	8
3	11,32%	6
4	1,88%	1
5	7,54%	4
6	3,77%	2
7	1,88%	1
8	1,88%	1

Anexo 1 - Teste 1 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	44,44%	28
1	23,80%	15
2	9,52%	6
3	7,93%	5
4	6,34%	4
5	4,76%	3
7	1,58%	1
10	1,58%	1

Anexo 2 - Teste 1 - AR2 - Dados de pacotes perdidos

O Arduíno coordenador enviou 174 mensagens e todas chegaram com sucesso.

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	75,86%	220
1	9,31%	27
2	4,82%	14
3	3,79%	11
4	1,72%	5
5	2,41%	7

6	0,68%	2
7	0,68%	2
8	0,34%	1
10	0,34%	1

Anexo 3 - Teste 1 - TOTAL - Dados de pacotes perdidos

	Total de pacotes
Numero de pacotes recebidos:	290
Numero de pacotes perdidos:	187

Anexo 4 Total por perdidos

Anexo teste 2

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	30,43%	7
1	34,78%	8
2	17,39%	4
3	4,34%	1
4	4,34%	1
5	4,34%	1
8	4,34%	1

Anexo 5 - Teste 2 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	46,42%	13
1	32,14%	9
2	10,71%	3
3	3,57%	1
4	3,57%	1
8	3,57%	1

Anexo 6 - Teste 2 - AR2 - Dados de pacotes perdidos

ARC	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	100%	59

Anexo 7 - Teste 2 - ARC - Dados de pacotes perdidos

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
--------------	---	---

0	71,81%	79
1	15,45%	17
2	6,36%	7
3	1,81%	2
4	1,81%	2
5	0,90%	1
8	1,81%	2

Anexo 8 - Teste 2 - Total - Dados de pacotes perdidos

Numero de pacotes recebidos:	110
Numero de pacotes perdidos:	66

Anexo 9 - Total recebidos por perdidos

Anexo teste 3

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	34,88%	15
1	20,93%	9
2	23,25%	10
3	6,97%	3
4	6,97%	3
5	4,65%	2
6	2,32%	1

Anexo 10 - Teste 3 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	39,13%	18
1	17,39%	8
2	21,73%	10
3	6,52%	3
4	8,69%	4
5	4,34%	2
6	2,17%	1

Anexo 11 - Teste 3 – AR2 - Dados de pacotes perdidos

ARC	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	100%	119

Anexo 12 - Teste 3 - ARC - Dados de pacotes perdidos

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	73,07%	152
1	8,17%	17
2	9,61%	20
3	2,88%	6
4	3,36%	7
5	1,92%	4
6	0,96%	2

Anexo 13 - Teste 3 - Total - Dados de pacotes perdidos

Numero de pacotes recebidos:	208
Numero de pacotes perdidos:	135

Anexo teste 4

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	42,55%	20
1	23,40%	11
2	17,02%	8
3	6,38%	3
4	6,38%	3
5	2,12%	1
6	2,12%	1

Anexo 14 - Teste 4 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	53,70%	29
1	20,37%	11
2	14,81%	8
3	5,55%	3
4	3,70%	2
5	1,85%	1

Anexo 15 - Teste 4 – AR2 - Dados de pacotes perdidos

ARC	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	100%	108

Anexo 16 - Teste 4 - ARC - Dados de pacotes perdidos

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	75,11%	157
1	10,52%	22
2	7,65%	16
3	2,87%	6
4	2,39%	5
5	0,95%	2
6	0,47%	1

Anexo 17 - Teste 4 - Total - Dados de pacotes perdidos

Numero de pacotes recebidos:	209
Numero de pacotes perdidos:	108

Anexo teste 5

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	41,75%	38
1	31,86%	29
2	12,08%	11
3	8,79%	8
4	2,19%	2
5	1,09%	1
6	2,19%	2

Anexo 18 - Teste 5 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	52,52%	52
1	25,25%	25
2	12,12%	12

3	7,07%	7
5	1,01%	1
7	1,01%	1
10	1,01%	1

Anexo 19 - Teste 5 – AR2 - Dados de pacotes perdidos

ARC	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	100%	196

Anexo 20 - Teste 5 - ARC - Dados de pacotes perdidos

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	74,09%	286
1	13,98%	54
2	5,95%	23
3	3,88%	15
4	0,51%	2
5	0,51%	2
6	0,51%	2
7	0,25%	1
10	0,25%	1

Anexo 21 - Teste 5 - total - Dados de pacotes perdidos

Numero de pacotes recebidos:	386
Numero de pacotes perdidos:	192

Anexo teste 6

AR1	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	34,37%	11
1	18,75%	6
2	15,62%	5
3	15,62%	5
4	9,37%	3
5	3,12%	1
6	3,12%	1

Anexo 22 - Teste 6 - AR1 - Dados de pacotes perdidos

AR2	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	72,72%	40
1	12,72%	7
2	12,72%	7
4	1,81%	1

Anexo 23 - Teste 6 – AR2 - Dados de pacotes perdidos

ARC	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	100	87

Anexo 24 - Teste 6 - ARC - Dados de pacotes perdidos

Total	Percentagem de pacotes perdidos entre as mensagens	Nº de pacotes em que ocorreu esta situação
0	79,31	138
1	7,47	13
2	6,89	12
3	2,87	5
4	2,29	4
5	0,57	1
6	0,57	1

Anexo 25 - Teste 6 - Total - Dados de pacotes perdidos

Numero de pacotes recebidos:	174
Numero de pacotes perdidos:	79

Tabela 10 Função de análise das mensagens

```

1. std::string TelnetServer::splitData( const char *chars){
2. std::string payload = chars;
3. std::string send;
4. string key = getValue(payload.c_str(), ',', 0);
5. messageContent.push((payload+"\n").c_str());
6. if(dispositivos.find(key) != dispositivos.end()){
7. this->numRecv++;
8. string count = getValue(payload.c_str(), ',', 1);

```

```

9. dispositivos[key]->setlostBtwMsg(stod(count) -
    dispositivos[key]->getNumLastMsg() - 1);

10.    if(!dispositivos[key]->find(dispositivos[key]-
        >getlostBtwMsg())){

11.        dispositivos[key]->inputDataLostOfPack(
            dispositivos[key]->getlostBtwMsg() , 0);

12.        dispositivos[key]->inputDataLostOfPack(
            dispositivos[key]->getlostBtwMsg() , (1 +
            dispositivos[key]->getLostOfpack(dispositivos[key]-
            >getlostBtwMsg())));

13.        lostOfPack[dispositivos[key]->getlostBtwMsg()] = 0;

14.        lostOfPack[dispositivos[key]->getlostBtwMsg()]++;

15.    }else{

16.        lostOfPack[dispositivos[key]->getlostBtwMsg()]++;

17.        dispositivos[key]->inputDataLostOfPack(
            dispositivos[key]->getlostBtwMsg() ,
            (1+dispositivos[key]->getLostOfpack(dispositivos[key]-
            >getlostBtwMsg())));

18.    }

19.    this->totalLost = this->totalLost +
        dispositivos[key]->getlostBtwMsg();

20.    dispositivos[key]->setNumLastMsg(stod(count));

21.    dispositivos[key]-
        >getHistogram().collect(dispositivos[key]-
        >getlostBtwMsg());

```

```

22.     countTotalLost.collect(dispositivos[key]-
    >getlostBtwMsg());

23.     send = key+', '+count+', ';

24.     if(getValue(payload.c_str(), ', ', 2).compare("I") ==
    0){

25.         send = send + "O,";

26.         if((getValue(payload.c_str(), ', ', 3).compare("L")
    == 0) && (stoi(getValue(payload.c_str(), ', ', 4))) <= 400){

27.             send = send + "L,I,";

28.         }else         if((getValue(payload.c_str(),          ', ',
    3).compare("L") == 0) && ((stoi(getValue(payload.c_str(),
    ', ', 4))) > 400)){

29.             send = send + "L,O,";

30.         }

31.         if((getValue(payload.c_str(), ', ', 5).compare("T")
    == 0) && (stoi(getValue(payload.c_str(), ', ', 6))) >= 0){

32.             send = send + "T,I,#";

33.             return send;

34.         }

35.         return "ERROR";

36.     }else         if(getValue(payload.c_str(),          ', ',
    2).compare("O") == 0){

37.         return "ERROR";

38.     }

39. }

```

```

40.      return "ERROR: Don't match the ifs payload";

                                41.      }

```

Anexo 26 Código Arduino Cordenador

```

1.  //Sensor de luz com LD

2.  #include <Printers.h>
3.  #include <XBee.h>
4.  #include <SoftwareSerial.h>
5.  #include <stdio.h>
6.  #include <stdlib.h>
7.  #include <string.h>
8.  #include <assert.h>

9.  SoftwareSerial XBee(3,2);

10. //#####
    #####

11. /**
12.  A0 define the port A0 of Arduino;
13.  */
14. #define A0 0
15. /**
16.  A1 define the port A1 of Arduino;
17.  */
18. #define A1 1
19. /**
20.  led define the port 13 of Arduino;
21.  */
22. #define LedRed 12
23. /**
24.  Define one identificator to Arduino;
25.  */
26. #define Arduino "ARC"
27. /**
28.  Define one letter to sensor;
29.  */
30. #define SensorLuminosidade 'L'
31. #define SensorTemperatura 'T'
32. /**
33.  Define one delimitator to end of file;

```

```

34. */
35. #define endoffile '#'

36. #define THREAD_READ_TIME 1000

37. #define THREAD_WRITE_TIME 2000

38. #define STOP_MSN 30000

39. #define itemInStruct 8 //ARC,1,I,L,123,T,3,#

40. //#####
    #####

41. //Define structer of send and recived package
42. typedef struct package{
43. String ID;
44. unsigned long int count;
45. char IO;
46. char sI1;
47. int vL;
48. char sI2;
49. int vT;
50. char endof;
51. }package;

52. //#####
    #####

53. //Global variables
54. const int ldrPin = A0; //nLDR no pino analítico 8
55. const int tmpPin = A1;
56. const int ledred = LedRed;
57. unsigned long int msgEnvCount;

58. unsigned long long int lastExecuteTimeWriteS = 0;
59. unsigned long long int lastExecuteTimeReadS = 0;
60. unsigned long long int lastExecuteTimeWriteAction = 0;
61. unsigned long long int lastExecuteTimeReadAction = 0;
62. unsigned long long int lastExecuteTimeWriteT = 0;

63. String ArduínoUno[] = {"AR1", "AR2"};

64. //#####
    #####
65. /*Define fuctions*/

```

```

66. void conect_xbee1();

67. int defineStructure(package *pack);

68. int takeDataSensorTMP(package *pack);

69. int takeDataSensorLDR(package *pack);

70. int actionSensorL(String action);

71. int actionSensorT(String action);

72. void ldrOn();

73. void ldrOff();

74. int str_union_xbee(String * data);

75. int str_union(String * data);

76. String getValue(String data, char separator, int index);

77. int howMany(String data, char count);

78. void conect_xbee2();

79. //#####
    #####

80. void setup() {
81.   msgEnvCount = 0;
82.   pinMode(LED_BUILTIN, OUTPUT);
83.   pinMode(ledred, OUTPUT);
84.   Serial.begin(9600); //Inicia a comunicação serial
85.   XBee.begin(9600);
86. }

87. void loop() {

88.   connect_serial();
89.   connect_xbee1();
90.   connect_xbee2();
91. }

```

```

92. #####
    #####

93. void connect_serial(){
94. unsigned long long int timee = 0;
95. unsigned long long int lastExecuteTimeOfWhile = 0;
96. String action;

97. Serial.print("0,#");
98. delay(1000);

99. msgEnvCount += 1;
100.package pack;
101.defineStructure(&pack);
102.takeDataSensorLDR(&pack);
103.takeDataSensorTMP(&pack);
104.pack.IO = 'T';
105.String message = pack.ID + "," + String(pack.count) + "," + pack.IO + "," + pack.sI1 + "," +
    String(pack.vL) + "," + pack.sI2 + "," + String(pack.vT) + "," + pack.endof;

106.String tam = String(message.length() * sizeof(char));
107.Serial.print( tam + ",#");

108.delay(500);

109.Serial.print(message);

110.//lastExecuteTimeOfWhile = millis();
111./*
112.while(1){

113.if(Serial.read() == '0'){
114.Serial.print(message);
115.}else if(Serial.read() == '1'){
116.Serial.print("1,#");
117.break;
118.}
119.delay(100);
120.// timee= (unsigned long long int)(millis());
121.//if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}

122.}*/
123.delay(1000);
124.Serial.flush();
125.action = "";

```

```

126.while (str_union(&action));
127.//Serial.print(action);
128.actionSensorL(getValue(action, ',', 4));
129.actionSensorT(getValue(action, ',', 6));
130.action = "";
131.//Serial.flush();
132.delay(500);

133.}

134./*
135.Conect xbee with server;
136.*/

137.void connect_xbee1(){
138.String message = "";
139.String action;
140.String tamanhoS = "";
141.int tamanhoI = 0;
142.unsigned long long int timee = 0;
143.String sizeMen;
144.unsigned long long int lastExecuteTimeOfWhile = 0;
145.String buf = "22,#";

146.lastExecuteTimeWriteS = millis();
147.lastExecuteTimeReadS = millis();
148.lastExecuteTimeWriteAction = millis();
149.lastExecuteTimeReadAction = millis();
150.lastExecuteTimeWriteT = millis();

151.//Serial.println("0,#");

152.lastExecuteTimeOfWhile = millis();

153./* First while */
154.while(1){
155.if (getValue(buf, ',', 0) != "ok1"){
156.if((millis() - lastExecuteTimeWriteS) >= THREAD_WRITE_TIME){
157.XBee.print("A1,#");
158.lastExecuteTimeWriteS = millis();

159.}
160.if((millis() - lastExecuteTimeReadS) >= THREAD_READ_TIME){
161.buf = " ";
162.str_union_xbee(&buf);
163.//Serial.print(buf);
164.lastExecuteTimeReadS = millis();
165.}

166.}else{
167.break;
168.}

```

```

169.timee= (unsigned long long int)(millis());
170.//Serial.println(timee);
171.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
172.}

173.tamanhoS = getValue(buf, ',', 1);
174.tamanhoI = tamanhoS.toInt();
175.int flag = 0;
176.// Serial.print("second ");
177.// Serial.println(buf);
178.lastExecuteTimeOfWhile = millis();

179./*Second while*/
180.while(1){
181.message = "";
182.str_union_xbee(&message);
183.//message += ",#";
184.int tam = message.length() * sizeof(char);
185.if(getValue(message, ',', 0) == "ARD" ) continue;
186.if(getValue(message, ',', 0) == "ok1"){
187.if(flag == 0){
188.continue;
189.}else{
190.tamanhoS = getValue(message, ',', 1);
191.tamanhoI = tamanhoS.toInt();
192.flag = 0;
193.}
194.}else if(getValue(message, ',', 0) == "AR1" ){
195.//Serial.println(message);
196.if(tam != 0 && tamanhoI == tam && howMany(message, ',') == itemInStruct - 1 &&
    getValue(message, ',', 3) == "L" && getValue(message, ',', 5) == "T"){
197.sizeMen = String(message.length() * sizeof(char));
198.Serial.print(message);
199.break;
200.}else {
201.flag = 1;
202.}
203.}
204.timee= (unsigned long)(millis());
205.//Serial.println(timee);
206.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
207.delay(1000);
208.}
209.//Serial.print( tam + ",#");

210.Serial.print("0,#");
211.delay(100);
212.Serial.print( sizeMen + ",#");
213.delay(500);
214.Serial.print(message);
215.delay(500);
216.//Serial.print("terc ");
217.//Serial.println(message);
218./*

```

```

219.while(1){
220.if(Serial.read() == '0'){
221.Serial.print(message);
222.}else if(Serial.read() == '1'){
223.Serial.print("1,#");
224.break;
225.}

226.}
227.*/
228./*Third while*/
229.delay(1000);
230.//Serial.flush();
231.action = "";
232.while (str_union(&action));
233.//Serial.print(action);

234.String tam = String(action.length() * sizeof(char));
235.//if(15 == action.length() * sizeof(char)){ldrOn();} //delay(1000); ldrOff();}
236.buf = "";

237.lastExecuteTimeOfWhile = millis();

238./*quarto while*/
239.while(1){
240.if((millis() - lastExecuteTimeWriteAction) >= THREAD_WRITE_TIME){
241.if(getValue(action, ',', 0) == "AR1" && howMany(action, ',') == (itemInStruct - 1) &&
    getValue(action, ',', 3) == "L" && getValue(action, ',', 5) == "T")
242.XBee.print(action);
243.lastExecuteTimeWriteS = millis() - 500; //-500
244.}
245.if((millis() - lastExecuteTimeReadAction) >= THREAD_READ_TIME){
246.buf = "";
247.str_union_xbee(&buf);

248.if(getValue(buf, ',', 0) == "1N0"){
249.break;
250.}
251.lastExecuteTimeReadS = millis();
252.}
253.if((millis() - lastExecuteTimeWriteT) >= THREAD_WRITE_TIME){
254.XBee.print( "tr1,"+tam+"#");
255.lastExecuteTimeWriteT = millis();
256.}
257.timee= (unsigned long)(millis());
258.//Serial.println(timee);
259.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
260.}
261.}

262.void connect_xbee2(){
263.String message = "";
264.String action;

```

```

265.String tamanhoS = "";
266.int tamanhoI = 0;
267.unsigned long long int timee = 0;
268.String sizeMen;
269.unsigned long long int lastExecuteTimeOfWhile = 0;
270.String buf = "22,#";

271.lastExecuteTimeWriteS = millis();
272.lastExecuteTimeReadS = millis();
273.lastExecuteTimeWriteAction = millis();
274.lastExecuteTimeReadAction = millis();
275.lastExecuteTimeWriteT = millis();

276.//Serial.println("0,#");

277.lastExecuteTimeOfWhile = millis();

278./* First while */
279.while(1){
280.if (getValue(buf, ',', 0) != "ok2"){

281.if((millis() - lastExecuteTimeWriteS) >= THREAD_WRITE_TIME){
282.XBee.print("A2,#");
283.lastExecuteTimeWriteS = millis();

284.}
285.if((millis() - lastExecuteTimeReadS) >= THREAD_READ_TIME){
286.buf = " ";
287.str_union_xbee(&buf);
288.//Serial.print(buf);
289.lastExecuteTimeReadS = millis();
290.}

291.}else{
292.break;
293.}
294.timee= (unsigned long long int)(millis());
295.//Serial.println(timee);
296.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
297.}

298.tamanhoS = getValue(buf, ',', 1);
299.tamanhoI = tamanhoS.toInt();
300.int flag = 0;
301.// Serial.print("second ");
302.// Serial.println(buf);
303.lastExecuteTimeOfWhile = millis();

304./*Second while*/
305.while(1){

```

```

306.message = "";
307.str_union_xbee(&message);
308.//message += ",#";
309.int tam = message.length() * sizeof(char);
310.if(getValue(message, ',', 0) == "ARD" ) continue;
311.if(getValue(message, ',', 0) == "ok2"){
312.if(flag == 0){
313.continue;
314.}else{
315.tamanhoS = getValue(message, ',', 1);
316.tamanhoI = tamanhoS.toInt();
317.flag = 0;
318.}
319.}else if(getValue(message, ',', 0) == "AR2" ){
320.//Serial.println(message);
321.if(tam != 0 && tamanhoI == tam && howMany(message, ',') == itemInStruct - 1 &&
    getValue(message, ',', 3) == "L" && getValue(message, ',', 5) == "T"){
322.sizeMen = String(message.length() * sizeof(char));
323.Serial.print(message);
324.break;
325.}else {
326.flag = 1;
327.}
328.}
329.timee= (unsigned long)(millis());
330.//Serial.println(timee);
331.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
332.delay(1000);
333.}
334.//Serial.print( tam + ",#");

335.Serial.print("0,#");
336.delay(100);
337.Serial.print( sizeMen + ",#");
338.delay(500);
339.Serial.print(message);
340.delay(500);
341.//Serial.print("terc ");
342.//Serial.println(message);
343./*
344.while(1){
345.if(Serial.read() == '0'){
346.Serial.print(message);
347.}else if(Serial.read() == '1'){
348.Serial.print("1,#");
349.break;
350.}

351.}
352.*/
353./*Third while*/
354.delay(1000);
355.//Serial.flush();
356.action = "";
357.while (str_union(&action));
358.//Serial.print(action);

```

```

359.String tam = String(action.length() * sizeof(char));
360.//if(15 == action.length() * sizeof(char)){ldrOn();} //delay(1000); ldrOff();}
361.buf = "";

362.lastExecuteTimeOfWhile = millis();
363.while(1){
364.if((millis() - lastExecuteTimeWriteAction) >= THREAD_WRITE_TIME){
365.if(getValue(action, ',', 0) == "AR2" && howMany(action, ',') == (itemInStruct - 1) &&
    getValue(action, ',', 3) == "L" && getValue(action, ',', 5) == "T")
366.XBee.print(action);
367.lastExecuteTimeWriteS = millis() -500;//-500
368.}
369.if((millis() - lastExecuteTimeReadAction) >= THREAD_READ_TIME){
370.buf = "";
371.str_union_xbee(&buf);

372.if(getValue(buf, ',', 0) == "N02"){
373.break;
374.}
375.lastExecuteTimeReadS = millis();
376.}
377.if((millis() - lastExecuteTimeWriteT) >= THREAD_WRITE_TIME){
378.XBee.print( "tr2,"+tam+"#");
379.lastExecuteTimeWriteT =millis();
380.}
381.timee= (unsigned long)(millis());
382.//Serial.println(timee);
383.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
384.}
385.}

386.//#####
    #####

387./**
388.Define the structure of packege from each Arduíno in the place;
389.*/
390.int defineStructure(package *pack){
391.pack->ID = Arduíno;
392.pack->count = msgEnvCount;
393.pack->sI1 = SensorLuminosidade;
394.pack->sI2 = SensorTemperatura;
395.pack->endof = endoffile;
396.}

397./**
398.Take datas from sensor temperature
399.*/
400.int takeDataSensorTMP(package *pack){
401.int tmpValue = analogRead(tmpPin);

```

```

402.// converting that reading to voltage, for 3.3v Arduino use 3.3
403.float voltage = tmpValue * 5.0;
404.voltage =voltage / 1024.0;

405.pack->vT= ((voltage - 0.5) * 10)/2 ;//converting from 10 mv per degree wit 500 mV offset
406.//to degrees ((voltage - 500mV) times 100)

407.return 1;

408.}

409./**
410.Take datas from sensorLDR
411.*/
412.int takeDataSensorLDR(package *pack){
413.int ldrValue = analogRead(ldrPin); //O valor lido será entre 0 e 1023;
414.pack->vL = ldrValue;
415.return 1;

416.}

417./**
418.Function to check the light sensor and send the information acquired to the LED.
419.*/
420.int actionSensorL(String action){
421.// int incomingByte = Serial.read();
422.if (action == "I"){
423.ldrOn();
424.return 0;
425.}else if (action == "O"){
426.ldrOff();
427.return 0;
428.}else return 1;

429.}

430./**
431.Function used to activate the actuators connected to the temperature sensor;
432.*/
433.int actionSensorT(String action){
434.// int incomingByte = Serial.read();
435.if (action == "I"){
436.digitalWrite(ledred, HIGH);
437.delay(10);
438.digitalWrite(ledred, LOW);
439.return 0;
440.}else return 1;
441.}

```

```

442./**
443.Function use to turn on the led
444.*/
445.void ldrOn(){
446.digitalWrite(LED_BUILTIN, HIGH); //Low light turns on
447.}

448./**
449.Function use to turn off the led
450.*/

451.void ldrOff(){
452.digitalWrite(LED_BUILTIN, LOW); //Up light turns off
453.}

454.//#####
#####

455.int str_union_xbee(String * data){
456.String resul;
457.int re = 0;
458.char c = "";

459.while(XBee.available() > 0 && c != '#'){
460.int incomingByte = XBee.read();
461.c = char(incomingByte);
462.resul = resul + c;
463.re++;
464.}

465.if(re != 0){
466.*data = resul;
467.return 0;
468.}else return 1;

469.}
470.int str_union(String * data){
471.String resul;
472.int re = 0;
473.char c = "";
474.//Serial.flush();
475.while(Serial.available() > 0 && c != '#'){
476.int incomingByte = Serial.read();
477.c = char(incomingByte);
478.resul = resul + c;
479.re++;
480.}
481.if(re != 0){
482.*data = resul;
483.return 0;
484.}else return 1;

```

```

485.}

486.String getValue(String data, char separator, int index)
487.{
488.int found = 0;
489.int strIndex[] = { 0, -1 };
490.int maxIndex = data.length() - 1;

491.for (int i = 0; i <= maxIndex && found <= index; i++) {
492.if (data.charAt(i) == separator || i == maxIndex) {
493.found++;
494.strIndex[0] = strIndex[1] + 1;
495.strIndex[1] = (i == maxIndex) ? i+1 : i;
496.}
497.}
498.return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

499.}

500.int howMany(String data, char count)
501.{
502.int found = 0;
503.int maxIndex = data.length() - 1;

504.for (int i = 0; i <= maxIndex; i++) {
505.if (data.charAt(i) == count) {
506.found++;
507.}
508.}
509.return found;

510.}
511.//=====ANEXO=====
512./* memset(&ldrValue, 0, sizeof(ldrValue));
513.Serial.print("Primeiro laco ok, buf: ");
514.Serial.println(buf);
515.Serial.print("segundo laco ok, message: ");
516.Serial.println(message);
517.Serial.print("terceiro laco ok, buf: ");
518.Serial.println(buf);
519.*/

```

Anexo 27 Código aduino end device

```

1. //Sensor de luz com LDR

2. #include <Printers.h>
3. #include <XBee.h>
4. #include <SoftwareSerial.h>
5. #include <stdio.h>

```

```

6. #include <stdlib.h>
7. #include <string.h>
8. #include <assert.h>

9. //"ARC,1,O,L,I,T,I,#"
10. SoftwareSerial XBee(3,2);

11. #####
    #####

12. /**
13. A0 define the port A0 of Arduino;
14. */
15. #define A0 0
16. /**
17. A1 define the port A1 of Arduino;
18. */
19. #define A1 1
20. /**
21. led define the port 13 of Arduino;
22. */
23. #define LedRed 12
24. /**
25. Define one identifier to Arduino;
26. */
27. #define Arduino "AR2"
28. /**
29. Define one letter to sensor;
30. */
31. #define SensorLuminosidade 'L'
32. #define SensorTemperatura 'T'
33. /**
34. Define one delimiter to end of file;
35. */
36. #define endoffile '#'

37. #define THREAD_READ_TIME 1000

38. #define THREAD_WRITE_TIME 2000

39. #define THREAD_WRITE_OK 300

40. #define STOP_MSN 30000

41. #define itemInStruct 8

42. #####
    #####

43. //Define structer of send and recived package
44. typedef struct package{

```

```

45. String ID;
46. unsigned long int count;
47. char IO;
48. char sI1;
49. int vL;
50. char sI2;
51. int vT;
52. char endof;
53. }package;

54. #####
#####

55. //Global variables
56. const int ldrPin = A0; //nLDR no pino analítico 8
57. const int tmpPin = A1;
58. const int ledred = LedRed;
59. unsigned long int msgEnvCount;

60. unsigned long long int lastExecuteTimeWriteS = 0;
61. unsigned long long int lastExecuteTimeWriteT = 0;
62. unsigned long long int lastExecuteTimeWriteAction = 0;
63. unsigned long long int lastExecuteTimeWriteUnlock = 0;
64. unsigned long long int lastExecuteTimeReadS = 0;
65. unsigned long long int lastExecuteTimeReadAction = 0;
66. unsigned long long int lastExecuteTimeReadUnlock = 0;

67. #####
#####

68. void setup() {
69. msgEnvCount = 0;
70. pinMode(LED_BUILTIN, OUTPUT);
71. pinMode(ledred, OUTPUT);
72. Serial.begin(9600); //Inicia a comunicação serial
73. XBee.begin(9600);
74. }

75. void loop() {
76. connect_xbee();
77. }

78. void connect_xbee(){
79. String buf = "2,#";
80. String action;
81. String tamanho;

```

```

82. String tamanhoS = "";
83. unsigned long long int timee = 0;
84. unsigned long long int lastExecuteTimeOfWhile = 0;
85. int tamanhoI = 0;

86. int flag = 0;

87. /*First while*/
88. lastExecuteTimeWriteUnlock = millis();
89. lastExecuteTimeReadUnlock = millis();

90. /*Second while*/
91. lastExecuteTimeWriteS = millis() - 500;//-500
92. lastExecuteTimeReadS = millis();
93. lastExecuteTimeWriteT = millis();

94. /*third while*/
95. lastExecuteTimeWriteAction = millis();
96. lastExecuteTimeReadAction = millis();

97. /*First while for*/

98. lastExecuteTimeOfWhile = millis();
99. //Serial.print("oi");
100.while(1){
101.if (getValue(buf, ',', 0) != "A2"){
102.Bee.print("N02,#");
103.f((millis() - lastExecuteTimeWriteUnlock) >= THREAD_WRITE_OK){
104.Bee.print("N02,#");
105.astExecuteTimeWriteUnlock = millis();
106.}
107.if((millis() - lastExecuteTimeReadUnlock) >= THREAD_READ_TIME){
108.uf = "";
109.tr_union_xbee(&buf);
110.astExecuteTimeReadUnlock = millis();
111.} }else{
112.break;
113.}
114.timee= (unsigned long)(millis());
115.//Serial.println(timee);
116.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}

117.delay(1000);
118.}
119.Serial.print("Primeiro laco ok, buf: ");
120.Serial.println(buf);

121.msgEnvCount +=1;
122.package pack;
123.defineStructure(&pack);
124.takeDataSensorLDR(&pack);

```

```

125.takeDataSensorTMP(&pack);
126.pack.IO = 'T';
127.String message = pack.ID + "," + String(pack.count)+ "," + pack.IO + "," + pack.sI1 + "," +
    String(pack.vL) + "," + pack.sI2 + "," + String(pack.vT) + "," + pack.endof;
128.String tam;
129.tam = String(message.length() * sizeof(char));

130.buf = "";

131./*second while*/
132.lastExecuteTimeOfWhile = millis();

133.while(1){
134.if((millis() - lastExecuteTimeWriteS) >= THREAD_WRITE_TIME){
135.Bee.print(message);
136.astExecuteTimeWriteS = millis() - 500;//-500
137.}

138.if((millis() - lastExecuteTimeReadS) >= THREAD_READ_TIME/2){
139.uf = "";
140.tr_union_xbee(&buf);
141.f(getValue(buf, ',', 0) == "tr2"){
142.break;

143.astExecuteTimeReadS = millis();
144.}

145.if((millis() - lastExecuteTimeWriteT) >= THREAD_WRITE_TIME){
146.Bee.print( "ok2,"+tam+"#");
147.astExecuteTimeWriteT =millis();
148.}
149.timee= (unsigned long)(millis());
150.//Serial.println(timee);
151.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
152.}

153.Serial.print("Segundo laco ok, buf: ");
154.Serial.println(buf);

155.tamanhoS = getValue(buf, ',', 1);
156.tamanhoI = tamanhoS.toInt();
157.flag = 0;

158./*third while*/
159.lastExecuteTimeOfWhile = millis();
160.while(1){
161.action = "";
162.str_union_xbee(&action);
163.int tam = action.length() * sizeof(char);
164.if(getValue(action, ',', 0) == "ok2"){
165.f(flag == 0){
166.continue;

```

```

167.else if(getValue(action, ',', 0) == "tr2"){
168.tamanhoS = getValue(action, ',', 1);
169.tamanhoI = tamanhoS.toInt();
170.flag = 0;

171.}else if(getValue(action, ',', 0) == "AR2" && howMany(message, ',') == (itemInStruct - 1)
    && getValue(message, ',', 3) == "L" && getValue(message, ',', 5) == "T"){
172.if(tam != 0 && tamanhoI == tam){
173.actionSensorL(getValue(action, ',', 4));
174.actionSensorT(getValue(action, ',', 6));
175.break;
176.}else {
177.flag = 1;
178.}
179.}
180.//Serial.print(action);
181.timee= (unsigned long)(millis());
182.//Serial.println(timee);
183.if((timee - lastExecuteTimeOfWhile) >= STOP_MSN){return;}
184.delay(1000);}

185.Serial.print("Terceiro laco ok, action: ");
186.Serial.println(action);

187.}

188.//=====
    =====

189./**
190.Define the structure of package from each Arduíno in the place;
191.*/
192.int defineStructure(package *pack){
193.pack->count = msgEnvCount;
194.pack->ID = Arduíno;
195.pack->sI1 = SensorLuminosidade;
196.pack->sI2 = SensorTemperatura;
197.pack->endof = endoffile;
198.}

199./**
200.Take datas from sensor temperature
201.*/
202.int takeDataSensorTMP(package *pack){
203.int tmpValue = analogRead(tmpPin);
204./ converting that reading to voltage, for 3.3v Arduíno use 3.3
205.loat voltage = tmpValue * 5.0;

```

```

206.oltage =voltage / 1024.0;

207.ack->vT= (voltage - 0.5) * 10 ;//converting from 10 mv per degree wit 500 mV offset
208./to degrees ((voltage - 500mV) times 100)

209.eturn 1;
210./memset(&temp, 0, sizeof(temp));

211.}

212./**
213.Take datas from sensorLDR
214.*/
215.int takeDataSensorLDR(package *pack){
216.int ldrValue = analogRead(ldrPin); //O valor lido será entre 0 e 1023;
217.pack->vL = ldrValue;
218.return 1;

219.// memset(&ldrValue, 0, sizeof(ldrValue));
220.}

221./**
222.Function to check the light sensor and send the information acquired to the LED.
223.*/
224.int actionSensorL(String action){
225.if (action == "I"){
226.ldrOn();
227.return 0;
228.}else if (action == "O"){
229.ldrOff();
230.return 0;
231.}else return 1;

232.}

233./**
234.Function used to activate the actuators connected to the temperature sensor;
235.*/
236.int actionSensorT(String action){
237.if (action == "I"){
238.digitalWrite(ledred, HIGH);
239.delay(10);

```

```

240.digitalWrite(ledred, LOW);
241.return 0;
242.}else return 1;
243.}

244./**
245.Function use to turn on the led
246.*/
247.void ldrOn(){
248.digitalWrite(LED_BUILTIN, HIGH); //Low light turns on
249.}

250./**
251.Function use to turn off the led
252.*/

253.void ldrOff(){
254.igitalWrite(LED_BUILTIN, LOW); //Up light turns off
255.}

256.//#####
#####

257.int str_union_xbee(String * data){
258.String resul;
259.int re = 0;
260.char c = "";

261.while(XBee.available() > 0 && c != '#'){
262.nt incomingByte = XBee.read();
263.= char(incomingByte);
264.esul = resul + c;
265.e++;
266.}

267.if(re != 0){
268.*data = resul;
269.return 0;
270.else return 1;
271.}

272.int str_union(String * data){
273.String resul;
274.int re = 0;
275.char c = "";

```

```

276.while(XBee.available() > 0 && c != '#'){
277.nt incomingByte = XBee.read();
278.= char(incomingByte);
279.esul = resul + c;
280.e++;
281.}
282.if(re != 0){
283.*data = resul;
284.return 0;
285.else return 1;
286.}

287.String getValue(String data, char separator, int index)
288.{
289.int found = 0;
290.int strIndex[] = { 0, -1 };
291.int maxIndex = data.length() - 1;

292.for (int i = 0; i <= maxIndex && found <= index; i++) {
293.if (data.charAt(i) == separator || i == maxIndex) {
294.ound++;
295.trIndex[0] = strIndex[1] + 1;
296.trIndex[1] = (i == maxIndex) ? i+1 : i;
297.}
298.}
299.return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

300.}

301.int howMany(String data, char count)
302.{
303.int found = 0;
304.int maxIndex = data.length() - 1;

305.for (int i = 0; i <= maxIndex; i++) {
306.if (data.charAt(i) == count) {
307.ound++;
308.}
309.}
310.return found;

311.}

```

Tabela 11 Código Python

```

1. import threading
2. from socket import *
3. import time
4. import serial
5. import time
6. import string

7. # Cria o nome do host
8. meuHost = '193.136.195.14'
9. # Utiliza este número de porto
10. minhaPort = 4242
11. sockobj = socket(AF_INET, SOCK_STREAM)
12. sockobj.connect((meuHost, minhaPort))
13. #sockobj.bind((meuHost, minhaPort))
14. ser = serial.Serial("/dev/ttyUSB0", 9600, timeout=0)
15. # O socket começa a esperar por clientes limitando a
16. # 5 conexões por vez
17. #sockobj.listen(1)

18. text = ""
19. enviarThread = 0
20. receberThread = 0
21. enviarArd = 0
22. dadoEnviado = 0
23. dataRecv = ""

24. def connectOmnetEnviar():
25. global enviarThread
26. global dadoEnviado
27. global text
28. global sockobj
29. while True:
30. if( enviarThread == 1):
31. sockobj.sendall(text.encode('utf-8'))
32. dadoEnviado = 1
33. time.sleep(0.03)
34. conexão.close()
35. return

36. def connectOmnetReceber():
37. global receberThread
38. global enviarArd
39. global dataRecv
40. global text
41. global sockobj
42. while True:

43. if( receberThread == 1):
44. while True:
45. Recebe data enviada pelo cliente
46. dataRecv = sockobj.recv(1024).decode("ascii")
47. receberThread = 0

```

```

48. # Se não receber nada paramos o loop
49. if not dataRecv:
50.     reak
51. else:
52.     nviarArd = 1
53.     conexão.close()
54.     return
55. def takeBuffer():
56.     character = "
57.     line = "
58.     while( character != '#'):
59.         character = ser.read().decode("ascii")
60.         line = line + str(character)
61.     return line

62. te = threading.Thread(target=connectOmnetEnviar)
63. tr = threading.Thread(target=connectOmnetReceber)
64. te.start()
65. tr.start()

66. while ser.inWaiting:
67.     try:
68.         i = 0
69.         nVer = 1
70.         print(".")
71.         buffer = takeBuffer()
72.         print(buffer+"first")
73.         buffer.split(',')
74.         if (buffer[0] != '0'): continue
75.         tamanho = takeBuffer()
76.         tamanho = tamanho.split(',')
77.         aux = str(tamanho[0])
78.         tam = int(aux)

79.         while(nVer):
80.             uffer = takeBuffer()
81.             tr(buffer)
82.             rint(buffer+"second")
83.             es = buffer.split(',')
84.             f(tes[0] == '0'): nVer = 0
85.             f(len(buffer) != tam and tam != 0):
86.                 ser.write(str.encode('0'))
87.             lse:break
88.             flag = takeBuffer()

89.             if(str(flag[0]) == '1'): break
90.             time.sleep(0.1)
91.             if not buffer: continue
92.             if(nVer == 0): continue
93.             text = buffer
94.             print("=====>>>")
95.             enviarThread = 1
96.             while(dadoEnviado == 0): continue
97.             enviarThread = 0
98.             dadoEnviado = 0
99.             receberThread = 1
100. while(enviarArd == 0): continue

```

```

101.receberThread = 0
102.enviarArd = 0
103.print("<<<=====")
104.if(receberThread == 0):
105.print(dataRecv)
106.ser.write(str.encode(dataRecv))
107.time.sleep(2)

108.except ser.SerialTimeoutException:
109.print('Data could not be read')

```

Tabela 12 dados teste 1

Dados da mensagens recebidas:

```

ARC,1,I,L,660,T,16,#
AR1,1,I,L,719,T,17,#
ARC,2,I,L,668,T,16,#
ARC,3,I,L,656,T,17,#
AR2,3,I,L,28,T,18,#
ARC,4,I,L,644,T,16,#
AR2,4,I,L,28,T,17,#
ARC,5,I,L,645,T,18,#
ARC,6,I,L,659,T,16,#
AR2,5,I,L,28,T,17,#
ARC,7,I,L,646,T,18,#
ARC,8,I,L,643,T,16,#
AR1,7,I,L,696,T,17,#
ARC,9,I,L,641,T,16,#
ARC,10,I,L,642,T,16,#
ARC,11,I,L,641,T,16,#
AR2,9,I,L,27,T,16,#
ARC,12,I,L,661,T,17,#
AR2,10,I,L,28,T,17,#
ARC,13,I,L,645,T,16,#
AR2,11,I,L,27,T,16,#
ARC,14,I,L,661,T,16,#
ARC,15,I,L,660,T,17,#
AR1,13,I,L,694,T,16,#
ARC,16,I,L,657,T,16,#
AR1,14,I,L,707,T,17,#
AR2,13,I,L,28,T,16,#
ARC,17,I,L,662,T,17,#
ARC,18,I,L,640,T,16,#
ARC,19,I,L,639,T,16,#
AR1,17,I,L,699,T,16,#
ARC,20,I,L,659,T,18,#
AR2,17,I,L,28,T,16,#
ARC,21,I,L,656,T,17,#
AR2,18,I,L,28,T,16,#
ARC,22,I,L,658,T,16,#
AR2,19,I,L,28,T,16,#
ARC,23,I,L,652,T,17,#
AR2,20,I,L,27,T,16,#
ARC,24,I,L,650,T,16,#
ARC,25,I,L,655,T,17,#

```

```

AR2,21,I,L,28,T,16,#
ARC,26,I,L,633,T,17,#
ARC,27,I,L,588,T,16,#
AR1,24,I,L,691,T,17,#
ARC,28,I,L,636,T,16,#
ARC,29,I,L,592,T,16,#
AR1,26,I,L,687,T,17,#
AR2,26,I,L,28,T,16,#
ARC,30,I,L,619,T,16,#
AR1,27,I,L,707,T,17,#
ARC,31,I,L,609,T,16,#
AR2,28,I,L,28,T,16,#
ARC,32,I,L,618,T,16,#
AR1,29,I,L,701,T,17,#
ARC,33,I,L,616,T,16,#
AR2,30,I,L,28,T,15,#
ARC,34,I,L,621,T,16,#
ARC,35,I,L,621,T,16,#
ARC,36,I,L,486,T,16,#
ARC,37,I,L,615,T,15,#
AR2,33,I,L,28,T,16,#
ARC,38,I,L,610,T,16,#
AR2,34,I,L,28,T,16,#
ARC,39,I,L,618,T,16,#
AR1,35,I,L,681,T,15,#
ARC,40,I,L,612,T,16,#
AR2,36,I,L,27,T,16,#
ARC,41,I,L,614,T,16,#
AR1,37,I,L,702,T,16,#
ARC,42,I,L,620,T,15,#
AR2,38,I,L,28,T,16,#
ARC,43,I,L,610,T,16,#
AR2,39,I,L,28,T,16,#
ARC,44,I,L,624,T,15,#
AR1,40,I,L,698,T,16,#
ARC,45,I,L,625,T,16,#
AR2,41,I,L,28,T,16,#
ARC,46,I,L,619,T,15,#
AR1,42,I,L,698,T,16,#
ARC,47,I,L,610,T,17,#
AR1,43,I,L,693,T,16,#
AR2,43,I,L,28,T,16,#
ARC,48,I,L,611,T,16,#
AR2,44,I,L,27,T,16,#
ARC,49,I,L,626,T,16,#
AR2,45,I,L,28,T,15,#
ARC,50,I,L,614,T,16,#
ARC,51,I,L,625,T,16,#
AR2,46,I,L,27,T,16,#
ARC,52,I,L,613,T,15,#
AR2,47,I,L,28,T,16,#
ARC,53,I,L,610,T,1,#
ARC,54,I,L,609,T,1,#
ARC,55,I,L,627,T,1,#
AR1,50,I,L,704,T,4,#
ARC,56,I,L,621,T,1,#

```

```

ARC, 57, I, L, 624, T, 1, #
ARC, 58, I, L, 613, T, 1, #
ARC, 59, I, L, 627, T, 1, #
ARC, 60, I, L, 610, T, 1, #
AR1, 54, I, L, 696, T, 4, #
AR2, 55, I, L, 27, T, 1, #
ARC, 61, I, L, 627, T, 1, #
AR2, 56, I, L, 28, T, 1, #
ARC, 62, I, L, 629, T, 1, #
AR1, 56, I, L, 707, T, 4, #
AR2, 57, I, L, 27, T, 1, #
ARC, 63, I, L, 616, T, 1, #
AR1, 57, I, L, 706, T, 4, #
ARC, 64, I, L, 623, T, 1, #
AR2, 59, I, L, 27, T, 1, #
ARC, 65, I, L, 630, T, 1, #
ARC, 66, I, L, 614, T, 1, #
ARC, 67, I, L, 623, T, 1, #
ARC, 68, I, L, 612, T, 1, #
AR1, 60, I, L, 705, T, 4, #
ARC, 69, I, L, 624, T, 1, #
AR2, 64, I, L, 28, T, 1, #
ARC, 70, I, L, 611, T, 1, #
AR2, 65, I, L, 28, T, 1, #
ARC, 71, I, L, 621, T, 1, #
ARC, 72, I, L, 628, T, 1, #
AR1, 64, I, L, 682, T, 4, #
ARC, 73, I, L, 629, T, 1, #
ARC, 74, I, L, 611, T, 1, #
AR1, 65, I, L, 692, T, 4, #
ARC, 75, I, L, 623, T, 1, #
AR1, 66, I, L, 693, T, 4, #
ARC, 76, I, L, 624, T, 1, #
AR1, 67, I, L, 698, T, 4, #
ARC, 77, I, L, 612, T, 1, #
AR2, 71, I, L, 27, T, 1, #
ARC, 78, I, L, 616, T, 1, #
ARC, 79, I, L, 9, T, 1, #
AR2, 72, I, L, 22, T, 1, #
ARC, 80, I, L, 9, T, 1, #
AR1, 70, I, L, 80, T, 4, #
ARC, 81, I, L, 9, T, 1, #
ARC, 82, I, L, 8, T, 1, #
ARC, 83, I, L, 9, T, 1, #
AR2, 75, I, L, 22, T, 1, #
ARC, 84, I, L, 9, T, 1, #
ARC, 85, I, L, 9, T, 1, #
AR1, 74, I, L, 83, T, 4, #
ARC, 86, I, L, 9, T, 1, #
ARC, 87, I, L, 8, T, 1, #
AR2, 79, I, L, 22, T, 1, #
ARC, 88, I, L, 9, T, 1, #
ARC, 89, I, L, 9, T, 1, #
ARC, 90, I, L, 9, T, 1, #
ARC, 91, I, L, 11, T, 1, #
AR1, 79, I, L, 83, T, 4, #

```

```

AR2,83,I,L,22,T,1,#
ARC,92,I,L,9,T,1,#
ARC,93,I,L,9,T,1,#
AR1,81,I,L,81,T,4,#
ARC,94,I,L,9,T,1,#
AR2,86,I,L,22,T,1,#
ARC,95,I,L,9,T,1,#
AR2,87,I,L,22,T,1,#
ARC,96,I,L,9,T,1,#
AR1,84,I,L,82,T,4,#
ARC,97,I,L,9,T,1,#
AR2,89,I,L,22,T,1,#
ARC,98,I,L,9,T,1,#
ARC,99,I,L,8,T,1,#
AR2,90,I,L,22,T,1,#
ARC,100,I,L,9,T,1,#
AR1,87,I,L,80,T,4,#
ARC,101,I,L,9,T,1,#
AR1,88,I,L,80,T,4,#
ARC,102,I,L,9,T,1,#
AR1,89,I,L,80,T,4,#
ARC,103,I,L,9,T,1,#
ARC,104,I,L,10,T,1,#
ARC,105,I,L,9,T,1,#
AR2,96,I,L,22,T,1,#
ARC,106,I,L,9,T,1,#
AR1,92,I,L,80,T,4,#
ARC,107,I,L,9,T,1,#
ARC,108,I,L,9,T,1,#
AR1,94,I,L,80,T,4,#
ARC,109,I,L,9,T,1,#
ARC,110,I,L,9,T,1,#
ARC,111,I,L,9,T,1,#
ARC,112,I,L,9,T,1,#
AR2,102,I,L,23,T,1,#
ARC,113,I,L,9,T,1,#
ARC,114,I,L,9,T,1,#
AR2,104,I,L,22,T,1,#
ARC,115,I,L,9,T,1,#
AR2,105,I,L,22,T,1,#
ARC,116,I,L,9,T,1,#
ARC,117,I,L,9,T,1,#
AR1,102,I,L,80,T,4,#
ARC,118,I,L,9,T,1,#
ARC,119,I,L,9,T,1,#
AR2,108,I,L,21,T,1,#
ARC,120,I,L,9,T,1,#
ARC,121,I,L,9,T,1,#
AR1,106,I,L,80,T,4,#
ARC,122,I,L,8,T,1,#
AR2,110,I,L,22,T,1,#
ARC,123,I,L,9,T,1,#
ARC,124,I,L,9,T,1,#
AR1,109,I,L,81,T,4,#
ARC,125,I,L,9,T,1,#
ARC,126,I,L,9,T,1,#

```

```

AR2,114,I,L,22,T,1,#
ARC,127,I,L,9,T,1,#
AR2,115,I,L,22,T,1,#
ARC,128,I,L,9,T,1,#
AR2,116,I,L,22,T,1,#
ARC,129,I,L,9,T,1,#
ARC,130,I,L,9,T,1,#
AR1,115,I,L,80,T,4,#
ARC,131,I,L,9,T,1,#
ARC,132,I,L,9,T,1,#
AR1,117,I,L,80,T,4,#
ARC,133,I,L,9,T,1,#
AR2,121,I,L,23,T,1,#
ARC,134,I,L,8,T,1,#
ARC,135,I,L,9,T,1,#
ARC,136,I,L,9,T,1,#
AR2,122,I,L,22,T,1,#
ARC,137,I,L,9,T,1,#
AR2,123,I,L,22,T,1,#
ARC,138,I,L,9,T,1,#
ARC,139,I,L,9,T,1,#
AR2,125,I,L,22,T,1,#
ARC,140,I,L,9,T,1,#
AR2,126,I,L,22,T,1,#
ARC,141,I,L,9,T,1,#
AR2,127,I,L,21,T,1,#
ARC,142,I,L,9,T,1,#
ARC,143,I,L,9,T,1,#
AR1,126,I,L,79,T,4,#
ARC,144,I,L,8,T,1,#
AR2,129,I,L,22,T,1,#
ARC,145,I,L,9,T,1,#
ARC,146,I,L,9,T,1,#
AR2,131,I,L,24,T,1,#
ARC,147,I,L,9,T,1,#
ARC,148,I,L,10,T,1,#
AR1,130,I,L,79,T,4,#
AR2,133,I,L,22,T,1,#
ARC,149,I,L,9,T,1,#
ARC,150,I,L,9,T,1,#
AR1,131,I,L,80,T,4,#
ARC,151,I,L,9,T,1,#
AR1,132,I,L,80,T,4,#
ARC,152,I,L,9,T,1,#
ARC,153,I,L,9,T,1,#
AR2,138,I,L,22,T,1,#
ARC,154,I,L,10,T,1,#
AR1,134,I,L,80,T,4,#
ARC,155,I,L,9,T,1,#
ARC,156,I,L,9,T,1,#
AR1,136,I,L,80,T,4,#
ARC,157,I,L,9,T,1,#
AR1,137,I,L,80,T,4,#
ARC,158,I,L,9,T,1,#
AR1,138,I,L,80,T,4,#
ARC,159,I,L,10,T,1,#

```

```

AR1,139,I,L,80,T,4,#
ARC,160,I,L,9,T,1,#
ARC,161,I,L,9,T,1,#
AR1,141,I,L,80,T,4,#
ARC,162,I,L,9,T,1,#
ARC,163,I,L,9,T,1,#
ARC,164,I,L,9,T,1,#
AR2,149,I,L,22,T,1,#
ARC,165,I,L,9,T,1,#
ARC,166,I,L,9,T,1,#
ARC,167,I,L,9,T,1,#
AR1,145,I,L,81,T,4,#
AR2,152,I,L,22,T,1,#
ARC,168,I,L,9,T,1,#
AR1,146,I,L,80,T,4,#
AR2,153,I,L,22,T,1,#
ARC,169,I,L,9,T,1,#
AR1,148,I,L,79,T,4,#
ARC,170,I,L,9,T,1,#
ARC,171,I,L,9,T,1,#
AR1,149,I,L,79,T,4,#
ARC,172,I,L,9,T,1,#
AR1,150,I,L,78,T,4,#
ARC,173,I,L,9,T,1,#
ARC,174,I,L,9,T,1,#

```

Tabela 13 Dados teste 2

Dados da mensagens recebidas:

```

ARC,1,I,L,560,T,0,#
AR2,3,I,L,685,T,2,#
ARC,2,I,L,553,T,0,#
AR2,4,I,L,684,T,2,#
ARC,3,I,L,550,T,0,#
AR2,5,I,L,682,T,2,#
ARC,4,I,L,546,T,0,#
ARC,5,I,L,547,T,0,#
ARC,6,I,L,551,T,0,#
AR2,7,I,L,703,T,2,#
ARC,7,I,L,569,T,0,#
AR1,9,I,L,954,T,4,#
ARC,8,I,L,581,T,0,#
AR2,9,I,L,751,T,1,#
ARC,9,I,L,571,T,0,#
ARC,10,I,L,626,T,0,#
AR2,11,I,L,792,T,1,#
ARC,11,I,L,644,T,0,#
ARC,12,I,L,648,T,0,#
AR1,14,I,L,972,T,4,#
ARC,13,I,L,633,T,0,#
AR1,15,I,L,973,T,4,#
AR2,14,I,L,756,T,1,#
ARC,14,I,L,609,T,0,#

```

```

ARC,15,I,L,608,T,0,#
ARC,16,I,L,576,T,0,#
ARC,17,I,L,572,T,0,#
AR1,19,I,L,950,T,4,#
ARC,18,I,L,570,T,0,#
ARC,19,I,L,536,T,0,#
AR1,21,I,L,937,T,4,#
ARC,20,I,L,519,T,0,#
ARC,21,I,L,510,T,0,#
ARC,22,I,L,498,T,0,#
AR1,23,I,L,932,T,4,#
AR2,23,I,L,614,T,1,#
ARC,23,I,L,485,T,0,#
ARC,24,I,L,479,T,0,#
AR1,24,I,L,931,T,4,#
AR2,25,I,L,600,T,1,#
ARC,25,I,L,475,T,0,#
AR2,26,I,L,594,T,1,#
ARC,26,I,L,475,T,0,#
AR1,26,I,L,925,T,4,#
AR2,27,I,L,586,T,2,#
ARC,27,I,L,479,T,0,#
AR2,28,I,L,581,T,1,#
ARC,28,I,L,480,T,0,#
ARC,29,I,L,474,T,0,#
AR1,29,I,L,909,T,4,#
AR2,30,I,L,554,T,1,#
ARC,30,I,L,458,T,0,#
ARC,31,I,L,441,T,0,#
AR1,31,I,L,898,T,4,#
ARC,32,I,L,424,T,0,#
AR2,33,I,L,514,T,1,#
ARC,33,I,L,416,T,0,#
ARC,34,I,L,410,T,0,#
AR1,34,I,L,896,T,4,#
AR2,35,I,L,506,T,1,#
ARC,35,I,L,402,T,0,#
AR1,35,I,L,896,T,4,#
ARC,36,I,L,396,T,0,#
AR1,36,I,L,895,T,4,#
ARC,37,I,L,391,T,0,#
AR1,37,I,L,892,T,4,#
ARC,38,I,L,383,T,0,#
AR2,39,I,L,467,T,1,#
ARC,39,I,L,372,T,0,#
ARC,40,I,L,366,T,0,#
AR1,40,I,L,862,T,4,#
AR2,41,I,L,440,T,1,#
ARC,41,I,L,354,T,0,#
ARC,42,I,L,345,T,0,#
AR2,43,I,L,410,T,1,#
ARC,43,I,L,332,T,0,#
AR2,44,I,L,400,T,1,#
ARC,44,I,L,320,T,0,#
ARC,45,I,L,309,T,0,#
AR2,46,I,L,369,T,1,#

```

```

ARC,46,I,L,296,T,0,#
AR1,46,I,L,805,T,4,#
ARC,47,I,L,284,T,0,#
ARC,48,I,L,274,T,0,#
ARC,49,I,L,259,T,0,#
AR1,49,I,L,777,T,4,#
ARC,50,I,L,246,T,0,#
ARC,51,I,L,233,T,0,#
AR1,51,I,L,760,T,4,#
AR2,51,I,L,269,T,1,#
ARC,52,I,L,220,T,0,#
AR1,52,I,L,750,T,4,#
AR2,52,I,L,251,T,1,#
ARC,53,I,L,205,T,0,#
AR1,53,I,L,737,T,4,#
AR2,53,I,L,228,T,1,#
ARC,54,I,L,190,T,0,#
AR2,54,I,L,213,T,1,#
ARC,55,I,L,176,T,0,#
AR1,55,I,L,707,T,4,#
AR2,55,I,L,193,T,1,#
ARC,56,I,L,161,T,0,#
AR2,56,I,L,182,T,1,#
ARC,57,I,L,148,T,0,#
AR1,57,I,L,665,T,4,#
AR2,57,I,L,162,T,1,#
ARC,58,I,L,138,T,0,#
AR2,58,I,L,493,T,1,#
ARC,59,I,L,329,T,0,#
AR1,59,I,L,758,T,4,#

```

Tabela 14 Dados teste 3

Dados da mensagens recebidas:

```

ARC,1,I,L,391,T,0,#
AR2,1,I,L,452,T,1,#
ARC,2,I,L,394,T,0,#
ARC,3,I,L,383,T,0,#
ARC,4,I,L,396,T,0,#
ARC,5,I,L,388,T,0,#
AR1,5,I,L,716,T,4,#
AR2,5,I,L,463,T,1,#
ARC,6,I,L,393,T,0,#
AR2,6,I,L,460,T,1,#
ARC,7,I,L,391,T,0,#
AR2,7,I,L,454,T,1,#
ARC,8,I,L,394,T,0,#
AR2,8,I,L,445,T,1,#
ARC,9,I,L,386,T,0,#
AR1,9,I,L,711,T,4,#
ARC,10,I,L,391,T,0,#
ARC,11,I,L,385,T,0,#
AR1,10,I,L,738,T,4,#
ARC,12,I,L,380,T,0,#

```

```

AR1,11,I,L,737,T,4,#
ARC,13,I,L,383,T,0,#
AR1,12,I,L,728,T,4,#
ARC,14,I,L,393,T,0,#
ARC,15,I,L,382,T,0,#
AR2,15,I,L,466,T,1,#
ARC,16,I,L,401,T,0,#
AR1,15,I,L,728,T,4,#
ARC,17,I,L,397,T,0,#
AR2,17,I,L,462,T,1,#
ARC,18,I,L,398,T,0,#
ARC,19,I,L,386,T,0,#
AR1,18,I,L,703,T,4,#
ARC,20,I,L,387,T,0,#
AR2,20,I,L,471,T,1,#
ARC,21,I,L,383,T,0,#
AR2,21,I,L,467,T,1,#
ARC,22,I,L,385,T,3,#
AR1,21,I,L,706,T,4,#
AR2,22,I,L,462,T,1,#
ARC,23,I,L,397,T,4,#
ARC,24,I,L,386,T,3,#
ARC,25,I,L,397,T,3,#
ARC,26,I,L,394,T,4,#
AR1,24,I,L,738,T,4,#
AR2,25,I,L,453,T,1,#
ARC,27,I,L,395,T,4,#
AR2,26,I,L,467,T,1,#
ARC,28,I,L,393,T,3,#
ARC,29,I,L,395,T,3,#
ARC,30,I,L,400,T,3,#
AR2,29,I,L,464,T,1,#
ARC,31,I,L,402,T,3,#
ARC,32,I,L,392,T,2,#
ARC,33,I,L,388,T,2,#
ARC,34,I,L,393,T,2,#
AR1,31,I,L,725,T,4,#
ARC,35,I,L,394,T,2,#
AR1,32,I,L,739,T,4,#
AR2,35,I,L,441,T,1,#
ARC,36,I,L,390,T,2,#
AR2,36,I,L,440,T,1,#
ARC,37,I,L,399,T,2,#
AR1,34,I,L,708,T,4,#
ARC,38,I,L,389,T,2,#
AR2,38,I,L,459,T,1,#
ARC,39,I,L,395,T,3,#
AR1,36,I,L,707,T,4,#
ARC,40,I,L,393,T,3,#
AR2,40,I,L,441,T,1,#
ARC,41,I,L,391,T,3,#
ARC,42,I,L,390,T,3,#
AR2,42,I,L,448,T,1,#
ARC,43,I,L,397,T,3,#
AR1,40,I,L,706,T,4,#
AR2,43,I,L,459,T,1,#

```

```

ARC,44,I,L,385,T,3,#
AR2,44,I,L,444,T,1,#
ARC,45,I,L,384,T,3,#
ARC,46,I,L,382,T,3,#
ARC,47,I,L,386,T,3,#
AR2,47,I,L,452,T,2,#
ARC,48,I,L,392,T,3,#
ARC,49,I,L,395,T,3,#
AR1,46,I,L,738,T,4,#
ARC,50,I,L,389,T,3,#
AR1,47,I,L,702,T,4,#
ARC,51,I,L,394,T,3,#
AR1,48,I,L,739,T,4,#
ARC,52,I,L,397,T,3,#
AR1,49,I,L,712,T,4,#
AR2,52,I,L,455,T,1,#
ARC,53,I,L,390,T,3,#
ARC,54,I,L,394,T,3,#
AR1,51,I,L,736,T,4,#
ARC,55,I,L,394,T,3,#
AR2,55,I,L,462,T,1,#
ARC,56,I,L,393,T,3,#
ARC,57,I,L,384,T,3,#
AR1,54,I,L,714,T,4,#
AR2,57,I,L,449,T,1,#
ARC,58,I,L,388,T,3,#
ARC,59,I,L,387,T,3,#
ARC,60,I,L,398,T,3,#
AR1,57,I,L,717,T,4,#
AR2,60,I,L,467,T,1,#
ARC,61,I,L,389,T,3,#
ARC,62,I,L,398,T,3,#
AR1,59,I,L,703,T,4,#
AR2,62,I,L,447,T,1,#
ARC,63,I,L,390,T,3,#
AR1,60,I,L,715,T,4,#
AR2,63,I,L,449,T,1,#
ARC,64,I,L,396,T,3,#
AR1,61,I,L,732,T,4,#
AR2,64,I,L,463,T,1,#
ARC,65,I,L,386,T,2,#
ARC,66,I,L,399,T,3,#
ARC,67,I,L,399,T,0,#
ARC,68,I,L,404,T,4,#
AR1,64,I,L,237,T,2,#
AR2,68,I,L,470,T,1,#
ARC,69,I,L,399,T,3,#
ARC,70,I,L,408,T,4,#
AR1,66,I,L,240,T,2,#
ARC,71,I,L,405,T,4,#
AR1,67,I,L,240,T,2,#
ARC,72,I,L,409,T,4,#
ARC,73,I,L,394,T,4,#
AR2,73,I,L,462,T,1,#
ARC,74,I,L,402,T,3,#
AR1,70,I,L,240,T,3,#

```

```

ARC,75,I,L,384,T,3,#
ARC,76,I,L,390,T,3,#
ARC,77,I,L,391,T,3,#
AR1,73,I,L,244,T,2,#
ARC,78,I,L,397,T,3,#
AR1,74,I,L,246,T,2,#
AR2,78,I,L,467,T,1,#
ARC,79,I,L,399,T,3,#
AR2,79,I,L,442,T,1,#
ARC,80,I,L,388,T,4,#
AR1,76,I,L,247,T,2,#
ARC,81,I,L,393,T,3,#
AR2,81,I,L,453,T,2,#
ARC,82,I,L,401,T,4,#
AR2,82,I,L,442,T,2,#
ARC,83,I,L,406,T,4,#
ARC,84,I,L,404,T,3,#
ARC,85,I,L,392,T,3,#
AR1,81,I,L,247,T,2,#
AR2,85,I,L,469,T,2,#
ARC,86,I,L,406,T,3,#
AR2,86,I,L,452,T,2,#
ARC,87,I,L,391,T,2,#
ARC,88,I,L,400,T,3,#
ARC,89,I,L,393,T,3,#
ARC,90,I,L,403,T,3,#
AR2,90,I,L,464,T,2,#
ARC,91,I,L,394,T,2,#
AR1,86,I,L,252,T,2,#
AR2,91,I,L,456,T,2,#
ARC,92,I,L,389,T,3,#
AR1,87,I,L,254,T,2,#
ARC,93,I,L,401,T,3,#
ARC,94,I,L,393,T,3,#
AR1,89,I,L,255,T,2,#
ARC,95,I,L,395,T,2,#
ARC,96,I,L,400,T,0,#
ARC,97,I,L,396,T,0,#
AR2,97,I,L,460,T,1,#
ARC,98,I,L,401,T,0,#
ARC,99,I,L,400,T,0,#
ARC,100,I,L,396,T,0,#
ARC,101,I,L,396,T,0,#
AR1,95,I,L,255,T,3,#
ARC,102,I,L,401,T,0,#
AR2,102,I,L,468,T,1,#
ARC,103,I,L,393,T,0,#
AR1,96,I,L,258,T,3,#
ARC,104,I,L,398,T,0,#
ARC,105,I,L,399,T,0,#
ARC,106,I,L,390,T,0,#
AR1,98,I,L,256,T,3,#
AR2,105,I,L,456,T,2,#
ARC,107,I,L,388,T,0,#
AR2,106,I,L,468,T,1,#
ARC,108,I,L,402,T,0,#

```

```

AR2,107,I,L,448,T,2,#
ARC,109,I,L,395,T,0,#
AR1,101,I,L,274,T,4,#
ARC,110,I,L,403,T,0,#
ARC,111,I,L,399,T,0,#
AR1,103,I,L,271,T,3,#
AR2,110,I,L,464,T,1,#
ARC,112,I,L,405,T,0,#
AR1,104,I,L,270,T,4,#
ARC,113,I,L,389,T,0,#
AR2,112,I,L,468,T,2,#
ARC,114,I,L,401,T,0,#
ARC,115,I,L,400,T,0,#
ARC,116,I,L,406,T,0,#
AR1,108,I,L,271,T,3,#
AR2,115,I,L,466,T,2,#
ARC,117,I,L,390,T,0,#
AR1,109,I,L,272,T,4,#
ARC,118,I,L,402,T,0,#
ARC,119,I,L,393,T,0,#

```

Tabela 15 Dados teste 4

Dados da mensagens recebidas:

```

ARC,1,I,L,942,T,9,#
AR1,1,I,L,956,T,5,#
AR2,1,I,L,973,T,2,#
ARC,2,I,L,939,T,8,#
AR2,2,I,L,972,T,2,#
ARC,3,I,L,940,T,9,#
AR1,3,I,L,957,T,5,#
ARC,4,I,L,931,T,9,#
AR1,4,I,L,936,T,3,#
AR2,5,I,L,965,T,2,#
ARC,5,I,L,945,T,9,#
AR1,5,I,L,955,T,3,#
ARC,6,I,L,926,T,8,#
AR1,6,I,L,950,T,4,#
ARC,7,I,L,926,T,8,#
AR2,8,I,L,963,T,2,#
ARC,8,I,L,931,T,9,#
ARC,9,I,L,936,T,8,#
AR1,9,I,L,949,T,3,#
ARC,10,I,L,926,T,8,#
AR1,10,I,L,938,T,3,#
ARC,11,I,L,905,T,8,#
AR1,11,I,L,925,T,3,#
ARC,12,I,L,887,T,8,#
AR2,13,I,L,930,T,2,#
ARC,13,I,L,875,T,8,#
ARC,14,I,L,876,T,8,#
ARC,15,I,L,881,T,8,#
AR1,15,I,L,914,T,3,#
ARC,16,I,L,897,T,8,#

```

AR1,16,I,L,928,T,3,#
 ARC,17,I,L,902,T,8,#
 AR1,17,I,L,935,T,3,#
 ARC,18,I,L,907,T,8,#
 AR2,19,I,L,958,T,2,#
 ARC,19,I,L,904,T,7,#
 AR2,20,I,L,960,T,2,#
 ARC,20,I,L,902,T,7,#
 AR2,21,I,L,958,T,1,#
 ARC,21,I,L,901,T,7,#
 ARC,22,I,L,904,T,7,#
 AR1,22,I,L,937,T,3,#
 ARC,23,I,L,912,T,8,#
 ARC,24,I,L,916,T,8,#
 AR1,24,I,L,937,T,3,#
 AR2,24,I,L,960,T,1,#
 ARC,25,I,L,929,T,8,#
 AR2,25,I,L,965,T,1,#
 ARC,26,I,L,922,T,8,#
 ARC,27,I,L,940,T,8,#
 AR1,27,I,L,945,T,3,#
 ARC,28,I,L,945,T,8,#
 AR1,28,I,L,951,T,3,#
 ARC,29,I,L,946,T,8,#
 AR2,29,I,L,960,T,2,#
 ARC,30,I,L,914,T,8,#
 AR1,30,I,L,934,T,3,#
 ARC,31,I,L,918,T,8,#
 AR1,31,I,L,941,T,4,#
 AR2,31,I,L,958,T,1,#
 ARC,32,I,L,902,T,7,#
 AR2,32,I,L,954,T,2,#
 ARC,33,I,L,923,T,9,#
 ARC,34,I,L,902,T,8,#
 AR2,34,I,L,940,T,1,#
 ARC,35,I,L,903,T,8,#
 AR1,35,I,L,929,T,4,#
 ARC,36,I,L,873,T,7,#
 AR2,36,I,L,930,T,2,#
 ARC,37,I,L,876,T,8,#
 ARC,38,I,L,878,T,8,#
 AR1,38,I,L,915,T,3,#
 AR2,38,I,L,935,T,1,#
 ARC,39,I,L,879,T,8,#
 AR2,39,I,L,940,T,1,#
 ARC,40,I,L,880,T,8,#
 ARC,41,I,L,883,T,8,#
 ARC,42,I,L,889,T,8,#
 AR1,43,I,L,929,T,3,#
 AR2,42,I,L,954,T,2,#
 ARC,43,I,L,895,T,8,#
 AR2,43,I,L,959,T,1,#
 ARC,44,I,L,901,T,7,#
 AR2,44,I,L,970,T,1,#
 ARC,45,I,L,911,T,7,#
 AR1,46,I,L,937,T,3,#

```

ARC,46,I,L,902,T,8,#
AR1,47,I,L,936,T,3,#
ARC,47,I,L,898,T,8,#
AR2,47,I,L,958,T,1,#
ARC,48,I,L,931,T,8,#
AR2,48,I,L,985,T,1,#
ARC,49,I,L,953,T,8,#
ARC,50,I,L,953,T,8,#
AR1,51,I,L,966,T,3,#
ARC,51,I,L,952,T,8,#
AR1,52,I,L,965,T,3,#
ARC,52,I,L,954,T,9,#
AR2,52,I,L,983,T,2,#
ARC,53,I,L,953,T,8,#
AR1,54,I,L,966,T,3,#
AR2,53,I,L,984,T,2,#
ARC,54,I,L,953,T,8,#
ARC,55,I,L,952,T,8,#
AR1,56,I,L,966,T,3,#
ARC,56,I,L,953,T,8,#
AR2,56,I,L,983,T,2,#
ARC,57,I,L,955,T,8,#
AR2,57,I,L,982,T,2,#
ARC,58,I,L,955,T,8,#
AR1,59,I,L,963,T,3,#
ARC,59,I,L,955,T,9,#
ARC,60,I,L,954,T,8,#
ARC,61,I,L,954,T,8,#
AR1,61,I,L,966,T,3,#
AR2,60,I,L,981,T,2,#
ARC,62,I,L,953,T,8,#
ARC,63,I,L,953,T,8,#
ARC,64,I,L,952,T,8,#
AR1,64,I,L,963,T,3,#
AR2,62,I,L,980,T,2,#
ARC,65,I,L,953,T,8,#
AR1,66,I,L,963,T,3,#
AR2,63,I,L,979,T,2,#
ARC,66,I,L,953,T,9,#
ARC,67,I,L,951,T,8,#
AR2,65,I,L,979,T,2,#
ARC,68,I,L,951,T,8,#
ARC,69,I,L,891,T,8,#
ARC,70,I,L,893,T,8,#
ARC,71,I,L,940,T,8,#
AR2,69,I,L,978,T,2,#
ARC,72,I,L,940,T,8,#
AR1,73,I,L,960,T,3,#
ARC,73,I,L,947,T,9,#
AR1,74,I,L,953,T,3,#
AR2,71,I,L,976,T,2,#
ARC,74,I,L,946,T,8,#
AR1,75,I,L,959,T,3,#
AR2,72,I,L,976,T,2,#
ARC,75,I,L,913,T,9,#
AR1,76,I,L,944,T,3,#

```

```

ARC,76,I,L,943,T,9,#
AR1,77,I,L,957,T,3,#
ARC,77,I,L,943,T,9,#
AR2,75,I,L,972,T,2,#
ARC,78,I,L,941,T,9,#
AR2,76,I,L,972,T,2,#
ARC,79,I,L,940,T,9,#
AR2,77,I,L,969,T,2,#
ARC,80,I,L,937,T,9,#
AR2,78,I,L,967,T,2,#
ARC,81,I,L,935,T,9,#
AR1,82,I,L,949,T,3,#
AR2,79,I,L,964,T,2,#
ARC,82,I,L,931,T,8,#
AR1,83,I,L,945,T,3,#
AR2,80,I,L,962,T,2,#
ARC,83,I,L,931,T,8,#
AR2,81,I,L,962,T,2,#
ARC,84,I,L,928,T,8,#
AR2,82,I,L,957,T,2,#
ARC,85,I,L,928,T,8,#
AR1,86,I,L,944,T,3,#
ARC,86,I,L,928,T,8,#
AR2,84,I,L,958,T,2,#
ARC,87,I,L,926,T,8,#
AR1,88,I,L,940,T,3,#
AR2,85,I,L,958,T,2,#
ARC,88,I,L,876,T,7,#
ARC,89,I,L,812,T,7,#
ARC,90,I,L,801,T,7,#
AR1,90,I,L,862,T,3,#
AR2,86,I,L,876,T,2,#
ARC,91,I,L,795,T,7,#
AR1,91,I,L,858,T,2,#
ARC,92,I,L,795,T,7,#
AR2,88,I,L,876,T,2,#
ARC,93,I,L,794,T,7,#
AR2,89,I,L,865,T,2,#
ARC,94,I,L,795,T,7,#
AR2,90,I,L,865,T,2,#
ARC,95,I,L,797,T,6,#
AR2,91,I,L,863,T,2,#
ARC,96,I,L,797,T,6,#
AR2,92,I,L,874,T,2,#
ARC,97,I,L,799,T,7,#
AR1,97,I,L,852,T,3,#
AR2,93,I,L,862,T,2,#
ARC,98,I,L,797,T,6,#
ARC,99,I,L,795,T,7,#
AR1,99,I,L,850,T,3,#
AR2,95,I,L,857,T,2,#
ARC,100,I,L,789,T,7,#
ARC,101,I,L,797,T,7,#
AR1,101,I,L,858,T,3,#
AR2,97,I,L,860,T,2,#
ARC,102,I,L,799,T,7,#

```

```

AR1,102,I,L,857,T,3,#
AR2,98,I,L,854,T,2,#
ARC,103,I,L,758,T,7,#
AR1,103,I,L,824,T,3,#
ARC,104,I,L,744,T,7,#
ARC,105,I,L,732,T,7,#
ARC,106,I,L,717,T,7,#
ARC,107,I,L,709,T,7,#
AR1,106,I,L,788,T,3,#
ARC,108,I,L,691,T,7,#
AR2,103,I,L,767,T,2,#

```

Tabela 16 Dados teste 5

Dados da mensagens recebidas:

```

ARC,1,I,L,629,T,4,#
AR2,1,I,L,673,T,2,#
ARC,2,I,L,623,T,3,#
AR1,2,I,L,57,T,3,#
ARC,3,I,L,9,T,0,#
ARC,4,I,L,7,T,0,#
AR1,4,I,L,54,T,3,#
ARC,5,I,L,30,T,0,#
ARC,6,I,L,60,T,0,#
AR1,6,I,L,139,T,3,#
ARC,7,I,L,37,T,0,#
AR1,7,I,L,72,T,3,#
ARC,8,I,L,42,T,0,#
ARC,9,I,L,9,T,0,#
AR2,9,I,L,9,T,2,#
ARC,10,I,L,8,T,0,#
AR2,10,I,L,9,T,2,#
ARC,11,I,L,9,T,0,#
AR1,11,I,L,56,T,3,#
ARC,12,I,L,8,T,0,#
ARC,13,I,L,8,T,0,#
ARC,14,I,L,9,T,0,#
AR1,14,I,L,56,T,3,#
ARC,15,I,L,8,T,0,#
ARC,16,I,L,8,T,0,#
AR1,16,I,L,58,T,3,#
ARC,17,I,L,9,T,0,#
AR1,17,I,L,58,T,3,#
ARC,18,I,L,9,T,0,#
AR1,18,I,L,58,T,3,#
ARC,19,I,L,9,T,0,#
ARC,20,I,L,9,T,0,#
ARC,21,I,L,9,T,0,#
AR1,21,I,L,57,T,3,#
ARC,22,I,L,9,T,0,#
AR2,21,I,L,10,T,2,#
ARC,23,I,L,8,T,0,#
AR2,22,I,L,10,T,2,#
ARC,24,I,L,8,T,0,#
ARC,25,I,L,8,T,0,#

```

```

ARC,26,I,L,8,T,0,#
AR2,25,I,L,10,T,2,#
ARC,27,I,L,8,T,0,#
AR1,26,I,L,58,T,3,#
AR2,26,I,L,10,T,2,#
ARC,28,I,L,9,T,0,#
AR1,27,I,L,58,T,3,#
ARC,29,I,L,9,T,0,#
AR1,28,I,L,59,T,3,#
AR2,28,I,L,10,T,2,#
ARC,30,I,L,8,T,0,#
AR2,29,I,L,10,T,2,#
ARC,31,I,L,8,T,0,#
AR2,30,I,L,12,T,2,#
ARC,32,I,L,8,T,0,#
AR2,31,I,L,10,T,2,#
ARC,33,I,L,9,T,0,#
AR1,32,I,L,58,T,3,#
AR2,32,I,L,10,T,2,#
ARC,34,I,L,9,T,0,#
ARC,35,I,L,8,T,0,#
AR1,34,I,L,58,T,3,#
AR2,34,I,L,10,T,2,#
ARC,36,I,L,8,T,0,#
ARC,37,I,L,9,T,0,#
ARC,38,I,L,8,T,0,#
AR2,36,I,L,10,T,2,#
ARC,39,I,L,8,T,0,#
AR2,37,I,L,9,T,2,#
ARC,40,I,L,8,T,0,#
AR1,39,I,L,60,T,3,#
ARC,41,I,L,8,T,0,#
ARC,42,I,L,8,T,0,#
ARC,43,I,L,8,T,0,#
AR1,42,I,L,61,T,3,#
AR2,41,I,L,10,T,2,#
ARC,44,I,L,8,T,0,#
ARC,45,I,L,9,T,0,#
AR1,44,I,L,62,T,3,#
AR2,43,I,L,10,T,2,#
ARC,46,I,L,8,T,0,#
AR1,45,I,L,61,T,3,#
ARC,47,I,L,8,T,0,#
ARC,48,I,L,9,T,0,#
AR1,47,I,L,62,T,3,#
AR2,46,I,L,10,T,2,#
ARC,49,I,L,8,T,0,#
AR1,48,I,L,61,T,3,#
AR2,47,I,L,10,T,2,#
ARC,50,I,L,8,T,0,#
AR2,48,I,L,10,T,2,#
ARC,51,I,L,8,T,0,#
AR1,50,I,L,64,T,3,#
AR2,49,I,L,10,T,2,#
ARC,52,I,L,8,T,0,#
AR1,51,I,L,63,T,3,#

```

```

AR2,50,I,L,10,T,2,#
ARC,53,I,L,8,T,0,#
AR1,52,I,L,63,T,3,#
ARC,54,I,L,8,T,0,#
ARC,55,I,L,8,T,0,#
AR2,53,I,L,10,T,2,#
ARC,56,I,L,8,T,0,#
ARC,57,I,L,8,T,0,#
AR2,55,I,L,10,T,2,#
ARC,58,I,L,8,T,0,#
ARC,59,I,L,8,T,0,#
AR1,58,I,L,63,T,3,#
ARC,60,I,L,8,T,0,#
AR2,58,I,L,10,T,2,#
ARC,61,I,L,8,T,0,#
AR1,60,I,L,63,T,3,#
ARC,62,I,L,9,T,0,#
AR1,61,I,L,64,T,3,#
AR2,60,I,L,10,T,2,#
ARC,63,I,L,8,T,0,#
AR2,61,I,L,10,T,2,#
ARC,64,I,L,8,T,0,#
ARC,65,I,L,9,T,0,#
AR2,63,I,L,10,T,2,#
ARC,66,I,L,8,T,0,#
AR1,65,I,L,64,T,3,#
AR2,64,I,L,10,T,2,#
ARC,67,I,L,8,T,0,#
AR2,65,I,L,10,T,2,#
ARC,68,I,L,9,T,0,#
AR1,67,I,L,66,T,3,#
AR2,66,I,L,10,T,2,#
ARC,69,I,L,9,T,0,#
ARC,70,I,L,9,T,0,#
AR1,69,I,L,66,T,3,#
AR2,68,I,L,9,T,2,#
ARC,71,I,L,9,T,0,#
AR1,70,I,L,65,T,3,#
ARC,72,I,L,9,T,0,#
AR1,71,I,L,67,T,3,#
ARC,73,I,L,9,T,0,#
ARC,74,I,L,9,T,0,#
AR2,72,I,L,10,T,2,#
ARC,75,I,L,8,T,0,#
AR1,74,I,L,66,T,3,#
ARC,76,I,L,9,T,0,#
ARC,77,I,L,9,T,0,#
AR1,76,I,L,66,T,3,#
AR2,75,I,L,10,T,2,#
ARC,78,I,L,9,T,0,#
ARC,79,I,L,8,T,0,#
AR1,78,I,L,68,T,3,#
AR2,77,I,L,10,T,2,#
ARC,80,I,L,8,T,0,#
ARC,81,I,L,8,T,0,#
ARC,82,I,L,8,T,0,#

```

```

AR1,80,I,L,67,T,3,#
ARC,83,I,L,8,T,0,#
ARC,84,I,L,8,T,0,#
ARC,85,I,L,8,T,0,#
AR1,82,I,L,70,T,3,#
AR2,83,I,L,10,T,2,#
ARC,86,I,L,8,T,0,#
AR1,83,I,L,69,T,3,#
ARC,87,I,L,8,T,0,#
AR1,84,I,L,69,T,3,#
ARC,88,I,L,11,T,0,#
AR2,86,I,L,9,T,2,#
ARC,89,I,L,8,T,0,#
ARC,90,I,L,9,T,0,#
AR2,88,I,L,10,T,2,#
ARC,91,I,L,8,T,0,#
AR2,89,I,L,9,T,2,#
ARC,92,I,L,7,T,0,#
ARC,93,I,L,8,T,0,#
ARC,94,I,L,9,T,0,#
ARC,95,I,L,8,T,0,#
AR1,91,I,L,71,T,3,#
AR2,93,I,L,10,T,2,#
ARC,96,I,L,9,T,0,#
AR1,92,I,L,72,T,3,#
ARC,97,I,L,8,T,0,#
ARC,98,I,L,8,T,0,#
ARC,99,I,L,9,T,0,#
AR1,94,I,L,73,T,3,#
AR2,97,I,L,9,T,2,#
ARC,100,I,L,9,T,0,#
AR2,98,I,L,10,T,2,#
ARC,101,I,L,9,T,0,#
ARC,102,I,L,9,T,0,#
AR1,98,I,L,74,T,3,#
ARC,103,I,L,9,T,0,#
AR2,101,I,L,10,T,2,#
ARC,104,I,L,8,T,0,#
AR2,102,I,L,10,T,2,#
ARC,105,I,L,8,T,0,#
AR1,101,I,L,74,T,3,#
AR2,103,I,L,10,T,2,#
ARC,106,I,L,9,T,0,#
AR1,102,I,L,75,T,3,#
ARC,107,I,L,8,T,0,#
ARC,108,I,L,9,T,0,#
ARC,109,I,L,8,T,0,#
AR1,105,I,L,75,T,3,#
ARC,110,I,L,8,T,0,#
AR1,106,I,L,76,T,3,#
AR2,107,I,L,10,T,2,#
ARC,111,I,L,9,T,0,#
AR1,107,I,L,76,T,3,#
AR2,108,I,L,10,T,2,#
ARC,112,I,L,9,T,0,#
ARC,113,I,L,10,T,0,#

```

```

AR1,109,I,L,79,T,3,#
ARC,114,I,L,9,T,0,#
AR2,111,I,L,10,T,2,#
ARC,115,I,L,9,T,0,#
ARC,116,I,L,9,T,0,#
AR1,112,I,L,78,T,3,#
AR2,113,I,L,10,T,2,#
ARC,117,I,L,9,T,0,#
AR1,113,I,L,78,T,3,#
AR2,114,I,L,10,T,2,#
ARC,118,I,L,9,T,0,#
AR1,114,I,L,78,T,3,#
AR2,115,I,L,10,T,2,#
ARC,119,I,L,9,T,0,#
AR2,116,I,L,10,T,2,#
ARC,120,I,L,9,T,0,#
ARC,121,I,L,9,T,0,#
AR1,117,I,L,80,T,3,#
ARC,122,I,L,9,T,0,#
AR1,118,I,L,82,T,3,#
AR2,119,I,L,10,T,2,#
ARC,123,I,L,8,T,0,#
AR2,120,I,L,10,T,2,#
ARC,124,I,L,9,T,0,#
ARC,125,I,L,9,T,0,#
AR2,122,I,L,9,T,2,#
ARC,126,I,L,9,T,0,#
AR1,122,I,L,80,T,3,#
AR2,123,I,L,10,T,2,#
ARC,127,I,L,8,T,0,#
AR2,124,I,L,10,T,2,#
ARC,128,I,L,9,T,0,#
AR1,124,I,L,79,T,3,#
AR2,125,I,L,10,T,2,#
ARC,129,I,L,9,T,0,#
ARC,130,I,L,9,T,0,#
AR1,126,I,L,79,T,3,#
AR2,127,I,L,10,T,2,#
ARC,131,I,L,9,T,0,#
AR2,128,I,L,9,T,2,#
ARC,132,I,L,9,T,0,#
AR1,128,I,L,81,T,3,#
AR2,129,I,L,10,T,2,#
ARC,133,I,L,9,T,0,#
AR2,130,I,L,10,T,2,#
ARC,134,I,L,9,T,0,#
AR2,131,I,L,10,T,2,#
ARC,135,I,L,8,T,0,#
AR1,131,I,L,81,T,3,#
AR2,132,I,L,9,T,2,#
ARC,136,I,L,8,T,0,#
ARC,137,I,L,9,T,0,#
AR1,133,I,L,81,T,3,#
AR2,134,I,L,10,T,2,#
ARC,138,I,L,8,T,0,#
AR2,135,I,L,10,T,2,#

```

```

ARC,139,I,L,8,T,0,#
AR1,135,I,L,81,T,3,#
AR2,136,I,L,10,T,2,#
ARC,140,I,L,9,T,0,#
AR1,136,I,L,82,T,3,#
ARC,141,I,L,8,T,0,#
AR1,137,I,L,82,T,3,#
AR2,138,I,L,10,T,2,#
ARC,142,I,L,8,T,0,#
AR1,138,I,L,82,T,3,#
AR2,139,I,L,10,T,2,#
ARC,143,I,L,8,T,0,#
ARC,144,I,L,8,T,0,#
AR1,140,I,L,82,T,3,#
AR2,141,I,L,10,T,2,#
ARC,145,I,L,8,T,0,#
AR1,141,I,L,82,T,3,#
AR2,142,I,L,9,T,2,#
ARC,146,I,L,8,T,0,#
AR1,142,I,L,82,T,3,#
ARC,147,I,L,8,T,0,#
AR1,143,I,L,83,T,3,#
ARC,148,I,L,8,T,0,#
AR1,144,I,L,82,T,3,#
AR2,145,I,L,9,T,2,#
ARC,149,I,L,8,T,0,#
ARC,150,I,L,8,T,0,#
AR2,146,I,L,10,T,2,#
ARC,151,I,L,8,T,0,#
ARC,152,I,L,9,T,0,#
AR1,148,I,L,83,T,3,#
ARC,153,I,L,9,T,0,#
AR1,149,I,L,83,T,3,#
AR2,149,I,L,9,T,2,#
ARC,154,I,L,9,T,0,#
AR1,150,I,L,83,T,3,#
ARC,155,I,L,8,T,0,#
AR1,151,I,L,84,T,3,#
AR2,151,I,L,10,T,2,#
ARC,156,I,L,8,T,0,#
ARC,157,I,L,8,T,0,#
ARC,158,I,L,8,T,0,#
ARC,159,I,L,8,T,0,#
AR2,155,I,L,9,T,2,#
ARC,160,I,L,8,T,0,#
AR2,156,I,L,9,T,2,#
ARC,161,I,L,8,T,0,#
AR2,157,I,L,9,T,2,#
ARC,162,I,L,8,T,0,#
ARC,163,I,L,8,T,0,#
AR1,158,I,L,83,T,3,#
AR2,159,I,L,10,T,2,#
ARC,164,I,L,9,T,0,#
AR1,159,I,L,83,T,3,#
ARC,165,I,L,8,T,0,#
ARC,166,I,L,8,T,0,#

```

```

AR1,161,I,L,83,T,3,#
ARC,167,I,L,9,T,0,#
AR1,162,I,L,84,T,3,#
AR2,163,I,L,10,T,2,#
ARC,168,I,L,8,T,0,#
AR2,164,I,L,8,T,2,#
ARC,169,I,L,8,T,0,#
AR1,164,I,L,84,T,3,#
AR2,165,I,L,9,T,2,#
ARC,170,I,L,8,T,0,#
AR1,165,I,L,83,T,4,#
AR2,166,I,L,10,T,2,#
ARC,171,I,L,9,T,0,#
AR1,166,I,L,85,T,3,#
ARC,172,I,L,8,T,0,#
AR1,167,I,L,85,T,3,#
ARC,173,I,L,8,T,0,#
AR2,169,I,L,10,T,2,#
ARC,174,I,L,8,T,0,#
AR1,169,I,L,85,T,3,#
AR2,170,I,L,10,T,2,#
ARC,175,I,L,8,T,0,#
AR1,170,I,L,85,T,3,#
AR2,171,I,L,9,T,2,#
ARC,176,I,L,8,T,0,#
AR2,172,I,L,9,T,2,#
ARC,177,I,L,8,T,0,#
AR1,173,I,L,85,T,3,#
ARC,178,I,L,8,T,0,#
AR2,174,I,L,9,T,2,#
ARC,179,I,L,8,T,0,#
ARC,180,I,L,8,T,0,#
AR2,176,I,L,9,T,2,#
ARC,181,I,L,8,T,0,#
AR1,177,I,L,85,T,3,#
AR2,177,I,L,9,T,2,#
ARC,182,I,L,9,T,0,#
AR2,178,I,L,9,T,2,#
ARC,183,I,L,8,T,0,#
AR2,179,I,L,9,T,2,#
ARC,184,I,L,8,T,0,#
AR2,180,I,L,9,T,2,#
ARC,185,I,L,8,T,0,#
AR1,181,I,L,85,T,3,#
ARC,186,I,L,8,T,0,#
AR2,182,I,L,8,T,2,#
ARC,187,I,L,8,T,0,#
AR1,183,I,L,86,T,4,#
ARC,188,I,L,8,T,0,#
AR2,184,I,L,10,T,2,#
ARC,189,I,L,8,T,0,#
AR1,185,I,L,86,T,3,#
AR2,185,I,L,9,T,2,#
ARC,190,I,L,8,T,0,#
ARC,191,I,L,9,T,0,#
ARC,192,I,L,9,T,0,#

```

AR1,188,I,L,88,T,3,#
AR2,187,I,L,10,T,2,#
ARC,193,I,L,9,T,0,#
ARC,194,I,L,9,T,0,#
AR1,190,I,L,88,T,3,#
AR2,189,I,L,10,T,2,#
ARC,195,I,L,9,T,0,#
AR1,191,I,L,88,T,3,#
ARC,196,I,L,9,T,0,#
AR2,191,I,L,11,T,2,#