Universidade Federal do Amazonas – UFAM
Instituto de Computação
[ICC – 304] Tópicos Especiais em Redes de Computadores
Prof.: Edjair Mota
Relatório - Aula Prática 4
Alunos:
Arlinton José, Joao Alberto, Yuri Assayag , Marcos Rebelo, Mateus Medeiros, Rodrigo Leão

1. Introdução

O trabalho prático 4 consiste na elaboração de uma comunicação entre dispositivos (sensores) utilizando o LoRa. A comunicação se dá através de um dispositivo sensor utilizando arduíno e um dispositivo gateway utilizando um raspberry pi. Este trabalho é focado na construção de uma arquitetura voltada para o sensoriamento remoto com a utilização da comunicação Lora, no qual teremos um cenário completo com informações que cheguem ao usuário final e alguma ação seja realizada.

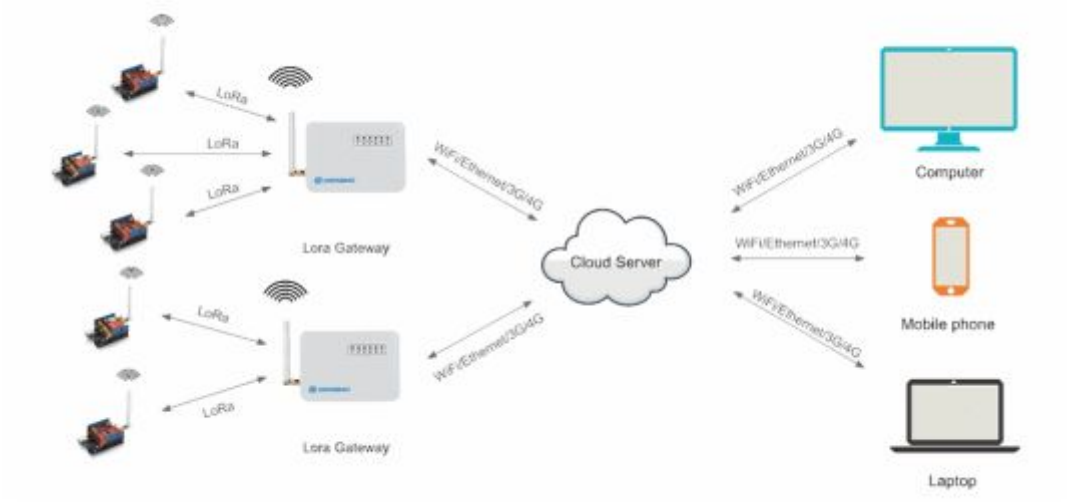

Figura 1. Etapas de comunicação com Lora.

2. Objetivo

Objetivo geral: Nossa equipe tem como objetivo desenvolver um código para dispositivos Lora com características de pacotes criptografados aumentando o grau de segurança de dispositivos IoT, realizando atividades relacionadas aos dispositivos finais, que serão representados por dispositivos arduinos integrados ao módulo Lora Shield.

Objetivos específicos:

1. Coletar as informações do sensor de pH e enviar para o gateway.
2. Realizar ações através do dispositivo atuador (Acender Led) de acordo com requisições do usuário que são dadas através do gateway.
3. Implementar um protocolo de comunicação segura através da criptografia.

3. Materiais Utilizados

Sender	Gateway
2 x Arduíno UNO	Arduíno UNO
2 x Lora Shied RF95	Lora Shied RF95
Led	-
Carregador Portatil 10000 mah Bateria Externa 5v	-
Sensor PH4502C PH 4502C Liquid PH Value Detection Detect Sensor Module Monitoring Control For Arduino	-

Tabela 1.Materiais utilizados.

4. Configurações dos nós.

Sender (Arduino com LoraShield): É o responsável por fazer o envio dos pacotes. Realiza uma leitura do sensor de pH acoplado ao arduino. A implementação foi feita utilizando a biblioteca RadioHead conforme a especificação do trabalho. Adicionalmente foi adicionada a biblioteca RHEncryptedDriver e Speck para envolver a encriptação dos dados enviados.

No Setup() as seguintes configurações são necessárias:

- *setTxPower(20):* Seta a potência de transmissão para 20db.
- *setFrequency(915)*: Seta a frequência para 915MHz.
- *setSignalBandwidth(500000)*: Fixa a largura de banda em 500kHz.
- *setSpreadingFactor(X)*: Seta o fator de espalhamento. (No nosso caso setamos os codigos para 9).
- *setCodingRate(6)*: Taxa de codificação fixa em 4/5.
- - encryptKey = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}.

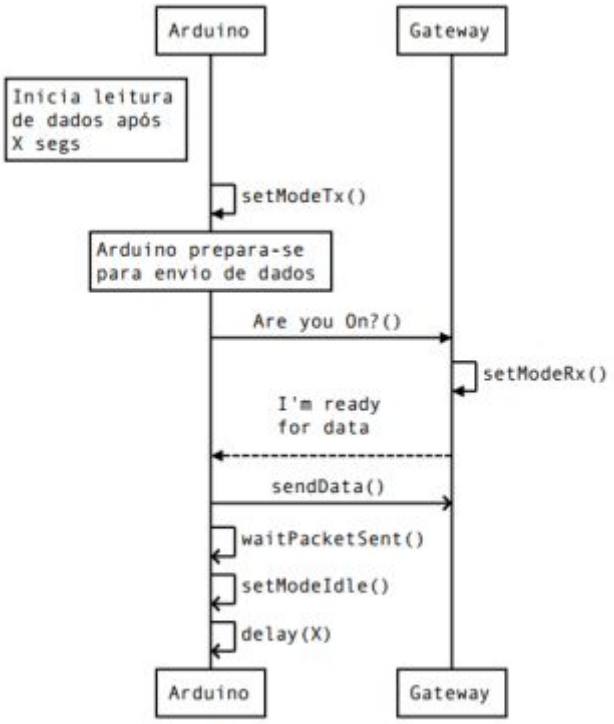

Figura 2. Diagrama de sequência do modo de transmissão de dados para o gateway

Inicialmente a comunicação implementada foi feita de acordo com o diagrama de sequência da Figura 2, no qual o arduino espera um tempo X para realizar o envio dos dados (estado esperando) e verifica se o gateway está pronto para recebê-los, enviando pacotes de "*are you on?"*. Quando a confirmação é dada, o sender realiza o envio dos dados do sensor (estado enviando) e volta a entrar no modo de espera.

O outro arduino é o responsável por realizar uma ação, que neste caso, desligar ou acender um Led. A requisição para fazer a ação é enviada pelo gateway, que recebe essa autorização através do usuário final, seguindo o diagrama da Figura 3. Para isso, o arduino espera também um tempo X e entra no modo de espera para que seja contactado pelo gateway quando alguma ação está sendo requisitada. Quando a mensagem de check é recebida, o arduino informa que está disponível ao gateway, que então envia a mensagem para a realização da ação e então realiza a ação conforme a requisição (seja para acender ou apagar o led). Após a realização da ação, o arduino volta a entrar no modo de espera.

Embora não faça parte da responsabilidade da equipe 1 de implementar o gateway (responsável por receber os dados do sensor de pH e de enviar a requisição da realização da ação), implementamos um gateway para realizar um teste da nossa implementação.

Toda a comunicação se dá através da criptografia, então caso cheguem mensagens que não sejam criptografadas (com a mesma chave de segurança), serão ignoradas pelo sistema e o

arduino continuará no modo de espera.

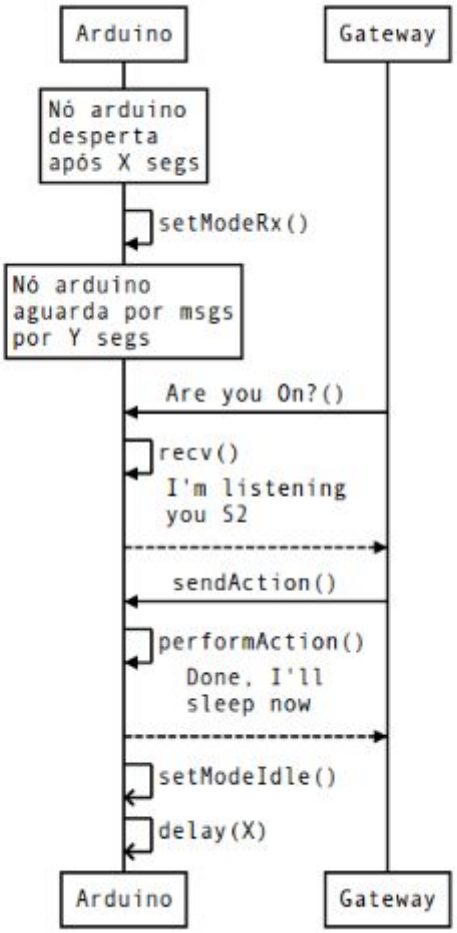

Figura 3. Diagrama de sequência do modo de recepção de dados vindos do gateway.

5. Análises

Um dos primeiros problemas acontecidos durante o desenvolvimento, foi os módulos arduinos "travarem" durante a execução dos testes, pois os mesmos não suportaram, conforme análise do código "muitos" pontos de inspeção no código, inseridos como "Serial.println".

Outro problema encontrado foi devido a utilização dos módulos de criptografia que devem ser cuidadosamente preparados pois além de os dois lados estarem codificados, há a possibilidade de conflito com outras bibliotecas, como por exemplo a biblioteca do RTC (disponível no código do trabalho anterior que serviu como base para este novo trabalho). Após apagar a importação da biblioteca, o erro foi solucionado.

Um terceiro aspecto aconteceu de os algoritmos não se sincroniza entre as máquinas, devido ao tempo de transmissão, tempo de setup e inicialização dos dispositivos. Para resolver este problema foi necessário a inserção de comandos delay() para que o processo de transmissão e recepção acontecesse de forma planejada.

Um aspecto crucial deste trabalho foi a utilização de máquinas de estados para “organizar” a sincronização dos pacotes entre os dispositivos.

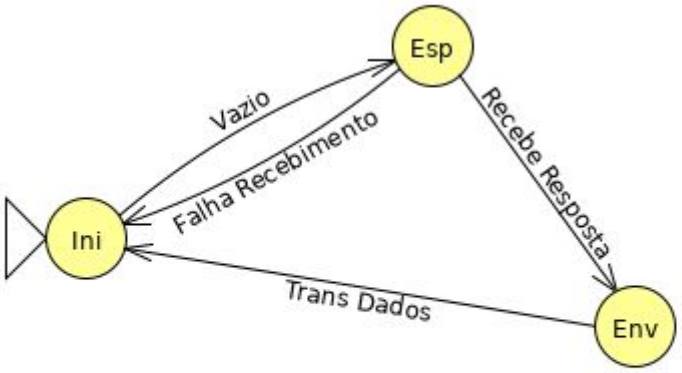

Figura 4. Diagrama de estado do arduino que envia os dados.

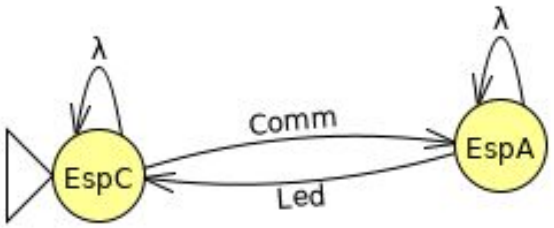

Figura 5. Diagrama de estado do arduino espera a solicitação de uma ação.

6.0 Conclusão

Conforme o experimento realizado podemos observar o comportamento do LoRa em ambientes de nós criptografados e aplicação de uma máquina de estados para a realização da comunicação.

Foi possível observar todas as etapas de comunicação Lora, assim como os detalhes de cada divisão da transmissão e os cuidados necessários para que a comunicação seja feita de forma plena, através da emissão e recepção dos dados.

7.0 Referências

[2] (2019). Iotlabs: Exploring lora technology. http://tet.pub.ro/pages/altele/Docs/Shield%20Dragino%20Lora/Lora%20Shield%20-%20Wiki%20for%20Dragino%20Project.pdf. Accessed: 2019-12-12

[3] (2019a). Lora com raspberry e arduíno. lora-com-raspberry-e-arduino/. Accessed: 2019-12-12. https://www.dobitaobyte.com.br/.

[4] (2019b). Lora shield and rpi to build a lorawan gateway. https://www.instructables.com/id/Use-Lora-Shield-and-RPi-to-Build-a-LoRaWAN-Gateway/. Accessed: 2019-12-12.

[5] (2019c). Lora tester for raspberry - a useful repository. https://github.com/lupyuen/LoRaArduino. Accessed: 2019-12-12.

[6] (2019d). Manual do lora shield. http://wiki.dragino.com/index.php?title=Lora_Shield. Accessed: 2019-12-12.

[7] (2019). pyradiohead github page. https://github.com/exmorse/pyRadioHeadRF95. Accessed: 2019- 12-12.

[8] (2019). Radiohead main page. https://www.airspayce.com/mikem/arduino/RadioHead/index.html. Accessed: 2019-12-12.

[10] Embarcados (2019). Lora shield dra gino+ raspberry. https://www.embarcados.com.br/lora-arduino-raspberry-pi-shield-dragino/. Accessed: 2019-12-12.

8.0 Anexos