Lab 2 - Tópicos Avançados em Redes de Computadores

Soares, Diogo*
Mota, Edjair

November 2019

1 Introdução

À medida que a Internet das Coisas (IoT) avança, fica cada vez mais evidente que novos modelos de arquitetura de rede são necessários para comportar o modelo de negócio e dispositivos envolvidos, tais como sensores, roteadores e servidores. Um destes modelos de rede [14] é demonstrado na Figura 1, no qual podemos separa a comunicação em 3 etapas: a comunicação entre dispositivos (sensores) e *gateways* usando comunicação LoRa, a comunicação entre *gateways* e servidores em nuvem ou locais usando WiFi, Ethernet ou 3G/4G e a etapa de disponibilização de acesso à usuários ou dispositivos finais como PCs, *dashboards* de aplicação, celulares, etc.

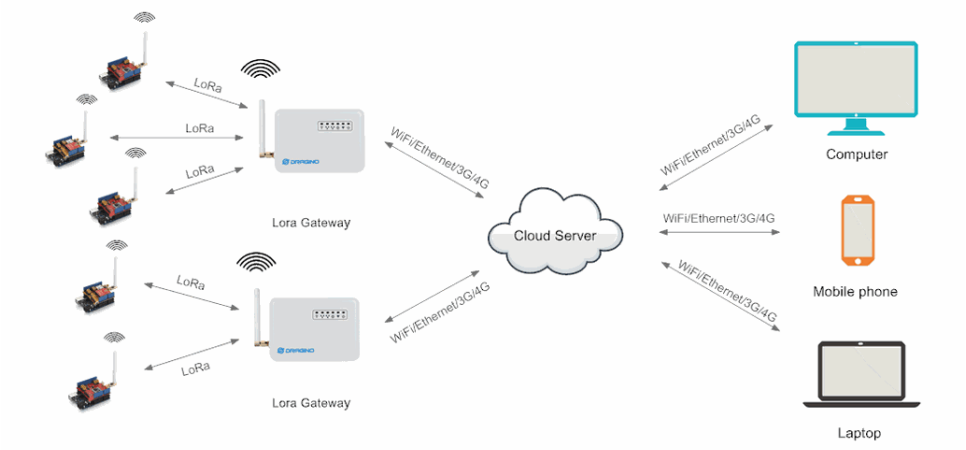

Figura 1: Etapas da comunicação com LoRa e sensores

Neste trabalho, iremos realizar uma investigação da primeira etapa de comunicação usando LoRa entre um dispositivo sensor utilizando arduíno e um dispositivo *gateway*, utilizando um raspberry pi. O objetivo é verificar questões relacionados as parametrizações da comunicação LoRa, efeitos na potência de sinal recebida, a relação sinal-ruído (SNR), tempo de comunicação, taxa de perda e distância de alcance entre os nodos. Para isso, serão necessários automatizar seus testes pois os mesmos serão realizados em diferentes ambientes e distâncias da UFAM.

*para tirar dúvidas: diogosoaresm@ufam.edu.br ou me procure no GRCM

2 O que deverá ser feito

2.1 Arduíno + Sensor + RTC

Você precisará realizar uma leitura do sensor utilizado na sua placa arduíno, leitura do tempo atual utilizando um módulo RTC [1] e enviando utilizando o LoRa. Neste trabalho recomendamos o uso da biblioteca RadioHead [8] pois ela possui uma boa aceitação na comunidade e implementar diversas funções que serão úteis neste trabalho.

Um exemplo básico de configuração e envio de dados usando a biblioteca RadioHead pode ser vista no exemplo abaixo:

Realiza configuração do LoRa (Usando RadioHead!!)

```
#include <RH_RF95.h>

RH_RF95 rf95;
#define DEBUG true

void setup() {
  Serial.begin(9600);
  while (!Serial) ; // Wait for serial port to be available

  if (!rf95.init())
    Serial.println("RF95 init failed");

  rf95.setTxPower(20, false);       //seta potencia de transmissao
  rf95.setFrequency(915.0);         //seta freq
  rf95.setSignalBandwidth(500000);  //seta BW (em Hz)
  rf95.setSpreadingFactor(8);       //seta SF
  rf95.setCodingRate4(6);           //seta CR
}

void loop() {
  float value = getSensorValue();   //funcao que pega dados de um sensor (exemplo)

  // Envia uma messagem para um rf95_server
  uint8_t valueToSend[10];
  dtostrf(value, 0, 10, valueToSend);
  rf95.send(valueToSend, sizeof(valueToSend));
  rf95.waitPacketSent();

#if defined(DEBUG)
  Serial.print("Msg enviada = ");
  Serial.println((char*)valueToSend);
#endif

  delay(1000);
}

float getSensorValue() {
    //ler dados...
    return 0.0;
}
```

Sobre as configurações de LoRa, serão necessárias as seguintes configurações:

1. Fator de espalhamento (*Spreading Factor* - SF): define o uso da técnica de espalhamento na comunicação, assim, grandes sequências de bits são codificadas em um único símbolo, visando diminuir a SNR. Desse modo, quanto maior SF, aumenta-se a SNR, a sensibilidade e o alcance. $SF \in \{SF7, SF8, SF9, SF10, SF11, SF12\}$.

2. Largura de banda (*Bandwidth* - BW): define a largura de banda utilizada na comunicação. Se a largura de banda for baixa, a sensibilidade será alta, mas a taxa de transmissão será mais baixa; a potência do sinal que o receptor vai detectar pode estar por baixo do nível de ruído, requerendo um maior ganho de processamento na recepção para decodificação do sinal. Uma largura de banda alta

leva a uma alta taxa de transferência, mas com maior tempo de transmissão e menor sensibilidade. $BW \in \{125kHz, 250kHz, 500kHz\}$.

3. Taxa de codificação (*Coding Rate* - CR): define a relação de correção adiantada de erros para redundância na mensagem, a fim de realizar a recuperação de erros. Um CR maior oferece maior proteção, no entanto, incrementa o tempo no ar de um pacote (tempo que o pacote consegue durar no ambiente até alcançar um destinatário). $CR \in \{\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}\}$.

Lembrando que estes parâmetros precisam ser configurados também no *gateway*. Assim, cada pacote deve ser definido de tal modo que o *payload* de dados deve ser do seguinte modo **num-xxx.xx-rtcTime**, no qual **num** corresponde ao id do pacote gerado (entre 1 e 100), **xxx.xx** corresponde ao valor coletado pelo sensor que estará utilizando (caso não seja um *float*, não precisa utilizar os valores após a vírgula) separados por um traco (-) e **rtcTime** é o tempo definido pelo RTC. Pode utilizar qualquer formato de tempo de tempo que deixe sua equipe mais confortável para trabalhar.

Para definir o RTC você pode sincronizar o mesmo no seu computador também, conforme descrito em [1] e no código disponibilizado abaixo. Caso utilize um RTC diferente do DS3231, consulte outro tutorial.

```
#include <DS3232RTC.h>        // https://github.com/JChristensen/DS3232RTC
#include <Streaming.h>        // http://arduiniana.org/libraries/streaming/

void setup()
{
    Serial.begin(9600);

    // setSyncProvider() causes the Time library to synchronize with the
    // external RTC by calling RTC.get() every five minutes by default.
    setSyncProvider(RTC.get);
    if (timeStatus() != timeSet)
      Serial.println("FAIL!");

    Serial.println("Setup finished");

    /** use o bloco abaixo
        apenas para testar o horario do RTC
    **/
    Serial.print("Verificando o tempo: ");
    testaTempo();
    Serial.println("");
    return 0;
    /** end **/
}

void loop()
{
    static time_t tLast;
    time_t t;
    tmElements_t tm;

    // check for input to set the RTC, minimum length is 12, i.e. yy,m,d,h,m,s
    if (Serial.available() >= 12) {
        // note that the tmElements_t Year member is an offset from 1970,
        // but the RTC wants the last two digits of the calendar year.
        // use the convenience macros from the Time Library to do the conversions.
        int y = Serial.parseInt();
        if (y >= 100 && y < 1000)
            Serial << F("Error: Year must be two digits or four digits!") << endl;
        else {
            if (y >= 1000)
                tm.Year = CalendarYrToTm(y);
            else    // (y < 100)
                tm.Year = y2kYearToTm(y);
            tm.Month = Serial.parseInt();
            tm.Day = Serial.parseInt();
            tm.Hour = Serial.parseInt();
            tm.Minute = Serial.parseInt();
            tm.Second = Serial.parseInt();
```

```
            t = makeTime(tm);
            RTC.set(t);         // use the time_t value to ensure correct weekday is set
            setTime(t);
            Serial << F("RTC set to: ");
            printDateTime(t);
            Serial << endl;
            // dump any extraneous input
            while (Serial.available() > 0) Serial.read();
        }
    }

    t = now();
    if (t != tLast) {
        tLast = t;
        printDateTime(t);
        if (second(t) == 0) {
            float c = RTC.temperature() / 4.;
            float f = c * 9. / 5. + 32.;
            Serial << F("  ") << c << F(" C  ") << f << F(" F");
        }
        Serial << endl;
    }
}

void testaTempo(){
  time_t t;

  t = now();
  printDateTime(t);
}

// print date and time to Serial
void printDateTime(time_t t)
{
    printDate(t);
    Serial << ' ';
    printTime(t);
}

// print time to Serial
void printTime(time_t t)
{
    printI00(hour(t), ':');
    printI00(minute(t), ':');
    printI00(second(t), ' ');
}

// print date to Serial
void printDate(time_t t)
{
    printI00(day(t), 0);
    Serial << monthShortStr(month(t)) << _DEC(year(t));
}

// Print an integer in "00" format (with leading zero),
// followed by a delimiter character to Serial.
// Input value assumed to be between 0 and 99.
void printI00(int val, char delim)
{
    if (val < 10) Serial << '0';
    Serial << _DEC(val);
    if (delim > 0) Serial << delim;
    return;
}
```

Listing 1: RTC *set* para DS3231

2.2 *Gateway* raspberry

Você precisará de uma rotina de escuta dos dados enviados. Aqui indicamos novamente a biblioteca RadioHead ou a biblioteca pyRadioHead [7], além de outras disponíveis pela *Internet*. Materiais relacionados à instalação e uso podem ser encontrados nas referências citadas e outras disponibilizadas em tutoriais online [2, 3, 4, 5, 10], o uso de bibliotecas é de livre escolha. Dado o fato de que disponibilizamos 3 raspberries com *setups* diferentes, tais como um raspberry conectado a um arduíno com LoRa *shield* e dois raspberries com LoRa *hat*, então a solução usada por uma equipe pode divergir da usada por outras.

Pede-se que implemente as seguintes rotinas para tratar cada situação abaixo. Lembre-se, consulte apenas o experimento relacionado à sua equipe.

- **Equipe 1 - SF**

  1. Analise o pacote gerado no arduino. Qual o tamanho dos pacotes gerados. Qual o tamanho usado para *header* e qual o tamanho usado para *payload*? Verifique a documentação da RadioHead e documente o que você encontrou.
  2. Usando $BW = 500kHz$ e $CR = \frac{4}{5}$ varie o SF entre todos os valores possíveis. Utilize o estacionamento da FT para realizar um experimento com visagem e visagem parcial conforme Figura 2 e Figura 3. Considere que, no mapa mostrado nas figuras você precisa dividir a distância total em 5 amostras (ex.: 240m do estacionamento da FT dividido por 5 = 48. Então utilize saltos de 48 metros no experimento, 0-48m, 0-96m, e assim por diante). Tenha em mente que para cada experimento serão gerados 100 pacotes enumerados.

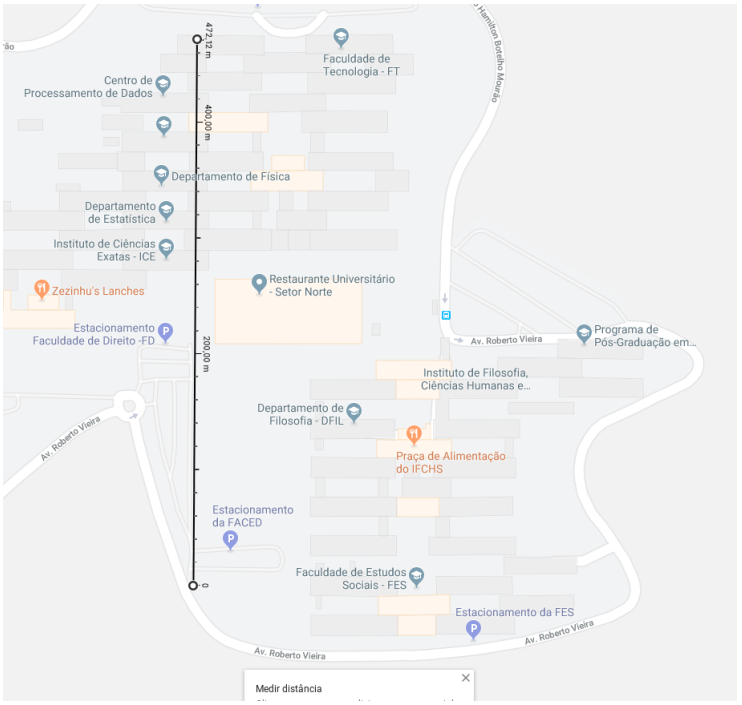

Figura 2: Cenário com visagem parcial

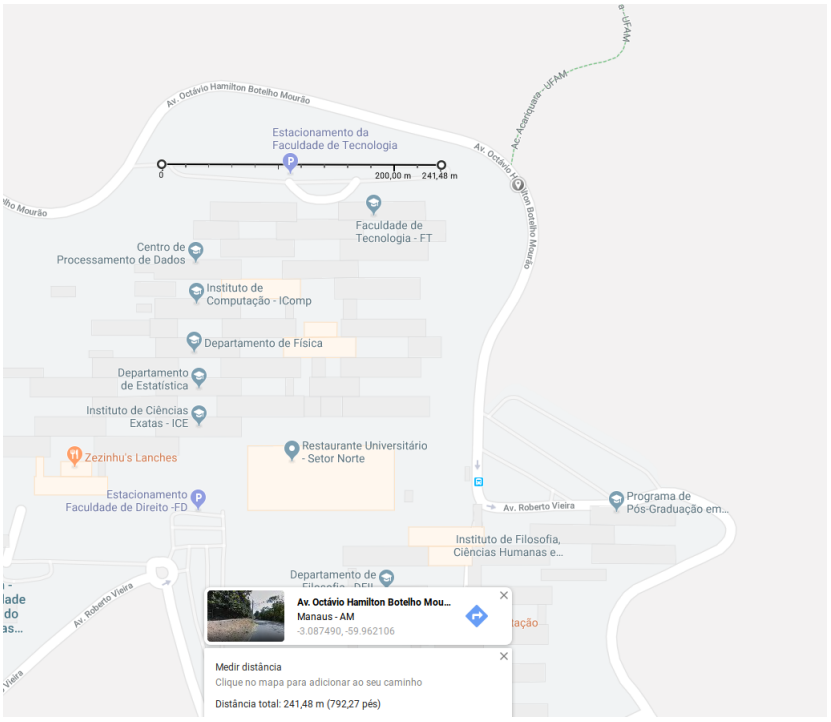

Figura 3: Cenário com visagem

3. Analise as seguintes métricas para cada distância utilizada e para cada valor de SF utilizado:
   - Perda de pacote - definida pela quantidade de pacotes perdidos dividido por 100. $\frac{Pacotes_{Perdidos}}{100} * 100$ (%).
   - Tempo de recepção - definido pela diferença de tempo do último pacote recebido pelo primeiro pacote recebido na unidade receptora (raspberry).
   - Potência do sinal - definida pelo RSSI na unidade receptora.
   - Relação sinal-ruído - definida pelo SNR em dBm na unidade receptora.
4. Pondere sobre as dificuldades encontradas durante a comunicação e responda:
   - Qual o melhor valor de SF para distâncias curtas? E para longas?
   - Qual a influência de SF no tempo de recepção?
   - Defina a importância de SF na comunicação entre dispositivos em um cenário de sensoriamento (sensores lendo dados e enviando via LoRa).

- **Equipe 2 - CR**

1. Analise o pacote gerado no arduino. Qual o tamanho dos pacotes gerados. Qual o tamanho usado para *header* e qual o tamanho usado para *payload*? Verifique a documentação da RadioHead e documente o que você encontrou.
2. Usando $BW = 500kHz$ e $SF = 12$ varie o CR entre todos os valores possíveis. Utilize o estacionamento da FT para realizar um experimento com visagem e visagem parcial conforme Figura 2 e Figura 3. Considere que, no mapa mostrado nas figuras você precisa dividir a distância total em 5 amostras (ex.: 240m do estacionamento da FT dividido por 5 = 48. Então utilize saltos de 48 metros no experimento, 0-48m, 0-96m, e assim por diante). Tenha em mente que para cada experimento serão gerados 100 pacotes enumerados.
3. Analise as seguintes métricas para cada distância utilizada e para cada valor de CR utilizado:
   - Perda de pacote - definida pela quantidade de pacotes perdidos dividido por 100. $\frac{Pacotes_{Perdidos}}{100} * 100$ (%).

- Tempo de recepção - definido pela diferença de tempo do último pacote recebido pelo primeiro pacote recebido na unidade receptora (raspberry).
- Potência do sinal - definida pelo RSSI na unidade receptora.
- Relação sinal-ruído - definida pelo SNR em dBm na unidade receptora.

4. Pondere sobre as dificuldades encontradas durante a comunicação e responda:
   - Qual o tempo de vida médio de um pacote no ar quando exposto a diferentes valores de taxa de codificação? Analise isso sob a ótica de seus resultados do item anterior.
   - Explique com suas palavras a importância do CR numa comunicação LoRa.
   - Qual a influência de CR no tamanho dos pacotes LoRa? Dica: consulte a documentação da RadioHead para verificar como avaliar o tamanho dos pacotes LoRa.

- **Equipe 3 - BW**

1. Analise o pacote gerado no arduino. Qual o tamanho dos pacotes gerados. Qual o tamanho usado para *header* e qual o tamanho usado para *payload*? Verifique a documentação da RadioHead e documente o que você encontrou.
2. Usando $SF = 12$ e $CR = \frac{4}{5}$ varie o BW entre todos os valores possíveis. Utilize o estacionamento da FT para realizar um experimento com visagem e visagem parcial conforme Figura 2 e Figura 3. Considere que, no mapa mostrado nas figuras você precisa dividir a distância total em 5 amostras (ex.: 240m do estacionamento da FT dividido por 5 = 48. Então utilize saltos de 48 metros no experimento, 0-48m, 0-96m, e assim por diante). Tenha em mente que para cada experimento serão gerados 100 pacotes enumerados.
3. Analise as seguintes métricas para cada distância utilizada e para cada valor de BW utilizado:
   - Perda de pacote - definida pela quantidade de pacotes perdidos dividido por 100. $\frac{Pacotes_{Perdidos}}{100} *$ 100 (%).
   - Tempo de recepção - definido pela diferença de tempo do último pacote recebido pelo primeiro pacote recebido na unidade receptora (raspberry).
   - Potência do sinal - definida pelo RSSI na unidade receptora.
   - Relação sinal-ruído - definida pelo SNR em dBm na unidade receptora.
4. Pondere sobre as dificuldades encontradas durante a comunicação e responda:
   - Explique com suas palavras o que é *Bandwidth* em uma comunicação LoRa e como ela pode ser útil em um cenário com ou sem visagem.
   - Qual a influência de BW na potência de sinal recebida e na relação sinal-ruído.
   - Qual largura de banda seria melhor para aumentar a distância de alcance da comunicação na sua opinião e porque?

Ao final, gere um relatório conforme o que é pedido para sua equipe. Todos os códigos utilizados deverão ser colocados em um repositório[1] *git* para correção. Lembre-se de realizar o *fork* do projeto, realizar os *commits* na pasta correspondente a de sua equipe (ex.: lab2/equipe1) e realizar o *pull request* na entrega, junto com o *pdf* do seu relatório. Descreva quaisquer dificuldades encontradas durante o experimento e lembre-se sempre de consultar qualquer referência antes de tirar qualquer dúvida.

A correção será baseado no resultado dos experimentos e da escrita do relatório.

Referências

[1] (2019). Como usar com arduino módulo real time clock - rtc. `https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-real-time-clock-rtc-ds3231/`. Accessed: 2019-11-06.

[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: 2019-11-06.

[1] https://github.com/diogosm/TERC-2019

[3] (2019a). Lora com raspberry e arduíno. https://www.dobitaobyte.com.br/lora-com-raspberry-e-arduino/. Accessed: 2019-11-06.

[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: 2019-11-06.

[5] (2019c). Lora tester for raspberry - a useful repository. https://github.com/lupyuen/LoRaArduino. Accessed: 2019-11-06.

[6] (2019d). Manual do lora shield. http://wiki.dragino.com/index.php?title=Lora_Shield. Accessed: 2019-11-06.

[7] (2019). pyradiohead github page. https://github.com/exmorse/pyRadioHeadRF95. Accessed: 2019-11-06.

[8] (2019). Radiohead main page. https://www.airspayce.com/mikem/arduino/RadioHead/index.html. Accessed: 2019-11-06.

[9] Bor, M. C., Roedig, U., Voigt, T., and Alonso, J. M. (2016). Do lora low-power wide-area networks scale? In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 59–67. ACM.

[10] Embarcados (2019). Lora shield dra gino+ raspberry. https://www.embarcados.com.br/lora-arduino-raspberry-pi-shield-dragino/. Accessed: 2019-11-06.

[Griese and Kleinschmidt] Griese, M. G. and Kleinschmidt, J. H. Aplicação da tecnologia lora em arquitetura para fiscalização eletrônica de veículos.

[12] KAROLESKI, J. A. (2018). Utilização de lora em redes para melhorar parâmetros de qualidade de distribuição de energia em zonas rurais.

[13] Pereira, P. H. M. (2019). Desenvolvimento de dispositivo de sensoreamento para cidades inteligentes usando o padrão lorawan.

[14] Raza, U., Kulkarni, P., and Sooriyabandara, M. (2017). Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873.

[15] Robinson, S. (2019). Long distance tracking andmonitoring with lora. https://www.dropbox.com/sh/wvspej3l4x4bq1e/AABJLE83HlteZzIjbx_zMvB9a?dl=0&preview=Long+Distance+Tracking+and+Monitoring+with+LoRa+-+Introduction+-+April+2016.pdf. Accessed: 2019-11-06.