Relatório de Pesquisa de Iniciação Científica

Título do Projeto:

Rede de Embarcações Não Tripuladas

Título do Relatório:

Rede ZigBee e suas Limitações

UERJ

Faculdade de Engenharia – FEN

Período deste relatório: setembro de 2016 a novembro de 2016

Autor: Ian Henriques de Andrade – Matricula:201410078211

Professor Orientador: José Paulo Vilela Soares da Cunha

Local de desenvolvimento do projeto: Laboratório de Controle e Automação da UERJ

Local e data: Rio de Janeiro, 31 de Agosto de 2017

Resumo

Este relatório expõe resultados e conclusões sobre as limitações de redes ZigBee no controle de processos com redes sem fio, utilizando o Arduino.

1 Introdução

A proposta deste trabalho é o estudo de redes sem fio tendo em vista sua aplicação em pequenas embarcações não tripuladas (BUORO, 2013; SOUSA, 2016), que podem ser usadas futuramente para integrar sensores e atuadores, entre outros fins (ROSARIO, 2013). Em um primeiro momento, começamos estudando uma rede sem fio do tipo ZigBee.

Em trabalhos anteriores, em que se utilizaram a rede ZigBee e a mesma embarcação que será utilizada neste projeto, foi alcançada uma frequência de comunicação de 30 Hz. Para monitoração do barco "em campo" é utilizado um sistema de câmeras VICON que monitoram a trajetória do barco, e esse sistema tem capacidade de fazer leituras de posicionamento em uma frequência de até 1 kHz, ou seja, estava sendo aproveitado apenas 3% da capacidade desse sistema (ROSARIO e CUNHA, 2016, Seção 6).

1.1 Objetivo

O objetivo principal no estudo da rede sem fio do tipo ZigBee é verificar seus limites da velocidade de comunicação, assim como seu atraso de comunicação, para analisar se seria viável continuar utilizando uma rede ZigBee no projeto, ou se seria necessário utilizar outro tipo de rede sem fio com uma velocidade de comunicação maior, como por exemplo, o Wi-Fi.

2 Considerações Iniciais

Os módulos Xbee utilizados são da Série 2 e o software utilizado para configuração foi o X-CTU. As configurações adotadas dos módulos BASE e REMOTO, foram, respectivamente, como COORDENADOR e ROTEADOR. Vale ressaltar que o módulo Remoto foi configurado como ROTEADOR, por esta configuração permitir a opção de "*NO SLEEP*", diferentemente da configuração como *END DEVICE*, ou seja, para que o dispositivo remoto não hibernasse a cada período de tempo como modo de economia de energia, pois nosso objetivo principal no momento é aumentar a taxa e reduzir o atraso da comunicação.

Tendo em vista que a embarcação que será utilizada possui um total de três motores e que necessitará que cada comando possua "quatro parâmetros", sendo um deles para informar que o comando enviado será para mudar os estados dos motores e os outros três para informar o comando de cada um dos motores, o teste realizado se baseia em comandos com quatro caracteres. Os *softwares* utilizados tanto no dispositivo BASE quanto no REMOTO estão disponibilizados no Apêndice A.

Os comandos considerados são enviar um estado "LOW" ou "HIGH" para um pino de saída do Arduino, dessa forma seria possível observar, com o auxílio de um osciloscópio, tanto o envio (BASE) quanto a execução do comando (REMOTO).

A velocidade de comunicação serial utilizada *BAUDRATE* foi de 115200 bps, que é a mais alta permitida pelos módulos Xbee. Cada conjunto de dados enviado é do tipo 8N1N, o qual possui um *start bit*, seguido por 8 bits de dados, e um *stop bit*, contendo um total de 10 bits por caractere enviado, como mostra a Figura 1.

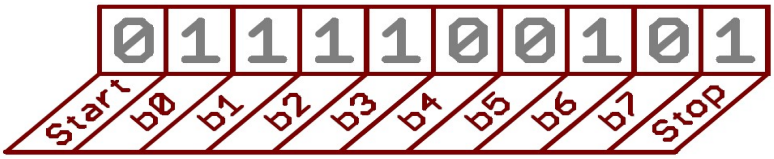


Figura 1 - Exemplo de um conjunto de dados utilizado.

Dessa maneira, é possível estimar um valor teórico para a máxima velocidade que poderá ser obtida nesta configuração, como detalhado na Tabela 1.

Tabela 1 - Cálculo Teórico da Frequência Máxima de comunicação.

Temos que:	1 Comando = 4 Caracteres
Sendo que:	1 Caractere = 10 Bits
Logo:	1 Comando = 40 bits
Como a *Baudrate* é 115200 bps, então:	
$\boldsymbol{Frequência\ máxima} = \frac{115200}{40} = \boldsymbol{2880\ Hz}$	

3 Testes Realizados

3.1 *Broadcast* x *Unicast*

Primeiramente, foi realizado um teste para comparar o desempenho da rede ZigBee, em modo AT, quando transmissão de dados era realizada por: ***Broadcast*** e ***Unicast***, ambas detalhadas na Figura 2. Isso é feito configurando os parâmetros DH e DL do módulo Coordenador no software X-CTU, no caso de transmissão *Unicast*, DH e DL devem ser o mesmo endereço de 64 bits do módulo destinatário, e no caso de transmissão *Broadcast*, da seguinte forma: DH = 0, DL = 0xFFFF, como mostra a Figura 3.

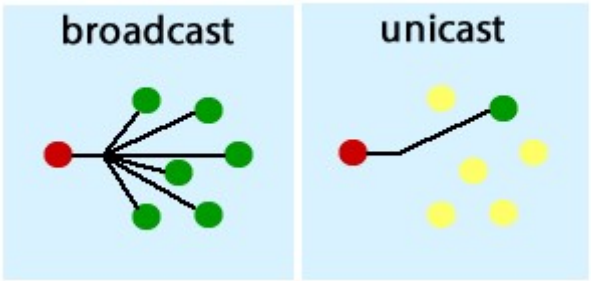


Figura 2-Transmissões UNICAST x BROADCAST.

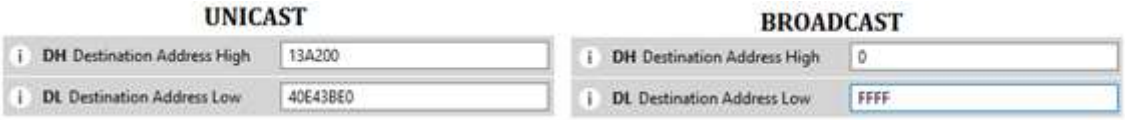
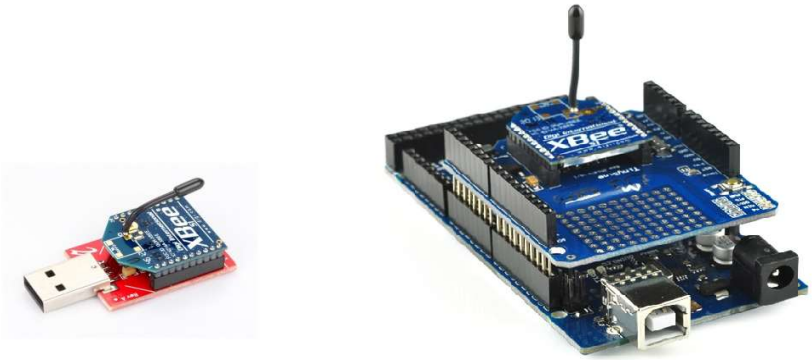


Figura 3 - Configuração do tipo de transmissão.

Para isso, foi utilizado a BASE, configurada coordenador em modo AT, conectada diretamente ao computador pelo *Xbee Explorer USB*, e o dispositivo remoto, também em modo AT, conectado por meio de um *Shield Xbee* à um Arduino UNO, como mostra a Figura 4.



DISPOSITIVO BASE (USB EXPLORER + XBEE) DISPOSITIVO REMOTO (ARDUÍNO + XBEE)

Figura 4 – Módulos utilizados nos experimentos.

O teste consistia no envio de 30 comandos consecutivos, espaçados por um período, e foi observado qual a limitação de ambas transmissões. O envio dos comandos foi feita por meio do *software* XCTU, dessa forma não foi possível monitorar o envio dos bytes no dispositivo BASE pelo osciloscópio.

Na Figura 5 observa-se que na transmissão ***Broadcast***, quando o tempo de envio entre cada comando era menor que 1 s, havia perda de pacotes, ou seja, nem todos comandos chegavam ao dispositivo remoto.

Já ao utilizar a transmissão ***Unicast***, como mostra a Figura 6, não houve perdas de comandos até um intervalo entre comandos de 14 ms, ou seja, uma frequência de aproximadamente 71 Hz.

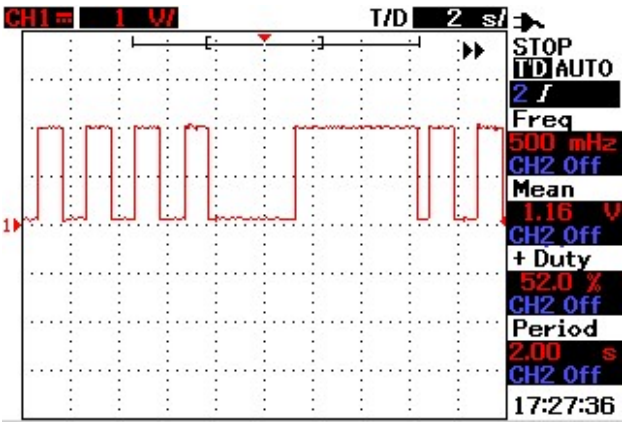


Figura 5 - Recepção de 30 comandos em modo BROADCAST.

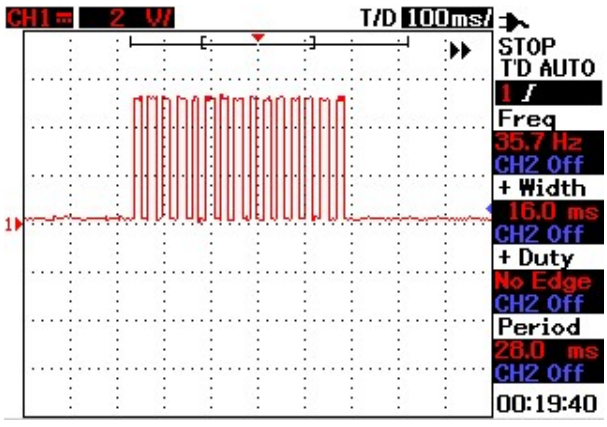


Figura 6 - Recepção de 30 comandos em modo UNICAST.

3.2 Velocidade de Comunicação e Tempo de Resposta dos Módulos ZigBee

Os objetivos deste teste são medir o atraso entre o envio do comando até a execução do mesmo, além de verificar o limite da velocidade de comunicação de uma rede Zigbee.

Para isso, foi utilizado tanto na base quanto no dispositivo remoto, o *Shield Xbee* acoplado ao Arduino UNO, como mostra a Figura 7. A transmissão entre os módulos ZigBee era realizada em modo *Unicast*. A Base enviava um comando a cada período e era possível verificar a resposta desse comando no dispositivo Remoto.

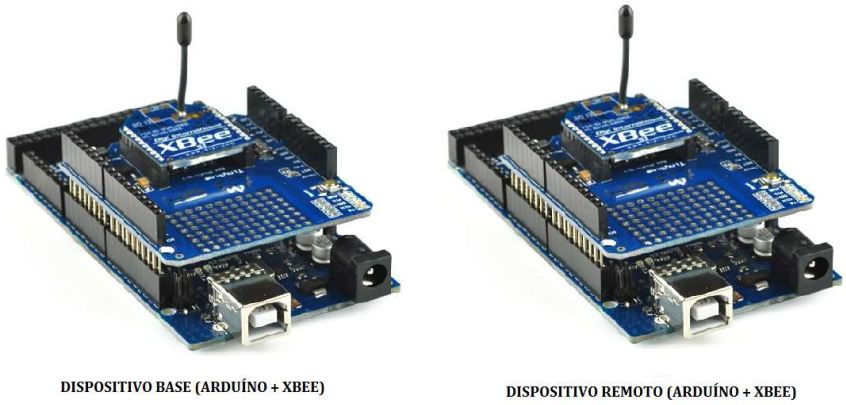


Figura 7 – Módulos utilizados no Experimento 2.

Observou-se que a máxima velocidade de comunicação alcançada, sem haver perda de comandos, foi de um comando a cada 15 ms, como mostra a Figura 8, ou seja uma frequência de aproximadamente 67 Hz. Além disso, verificou-se que, em todas as velocidades de comunicação medidas, o tempo de resposta era de 10 ms, como se vê na Figura 9.

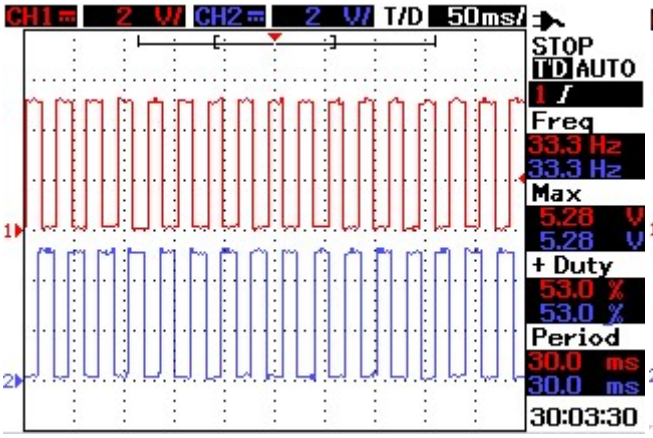


Figura 8 -Velocidade Máxima de Comunicação sem haver perdas. Canal 1: pulsos do disp. Base; Canal 2: pulsos do disp. Remoto.

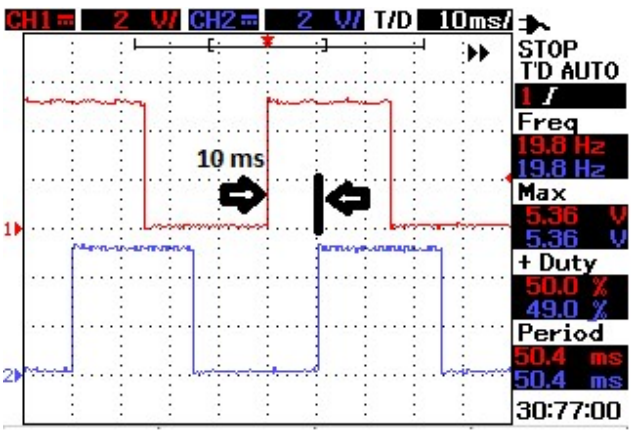


Figura 9 - Tempo de Resposta da Rede ZigBee. Canal 1: pulsos do disp. Base; Canal 2: pulsos do disp. Remoto.

3.3 Velocidade Máxima da Comunicação Serial com Fio

O objetivo deste teste é medir o limite de velocidade que pode ser alcançado utilizando os Arduínos ligados diretamente sem a utilização dos módulos XBee, além do tempo de resposta, para ser possível comparar o quanto uma rede ZigBee limita a velocidade da comunicação. Vale mencionar, que espera-se nesse experimento, ter uma medição da limitação máxima de comunicação entre arduinos, e ao utilizar qualquer rede sem fio, irá reduzir inevitavelmente o desempenho, porém o nosso objetivo será aproveitar ao máximo a essa capacidade do Arduino.

Para isso, foi utilizado os mesmos programas do experimento anterior, porém como não foram utilizados os módulos ZigBee, a comunicação entre a base e remoto foi feita por meio de três fios, que conectavam os pinos Tx_{BASE} ao Rx_{REMOTO} e o Rx_{BASE} ao Tx_{REMOTO} e GND_{BASE} ao GND_{REMOTO}.

Na Figura 10 observa-se que a velocidade máxima de transmissão sem a perda de comandos nesse teste foi de um intervalo de 340 µs entre comandos, ou seja, uma frequência de 2,94 kHz, mais do que 40 vezes maior do que a alcançada com a rede ZigBee. Vale mencionar ainda, que ao reduzir o intervalo entre comandos para valores menores que este, a velocidade real de envio não era alterada, podendo então este ser considerado a máxima velocidade de comunicação entre os Arduinos UNO utilizados. Além disso, o tempo de atraso observado neste experimento foi de cerca de 330 µs para todas as periodicidades de comando utilizadas, como mostra a Figura 11.

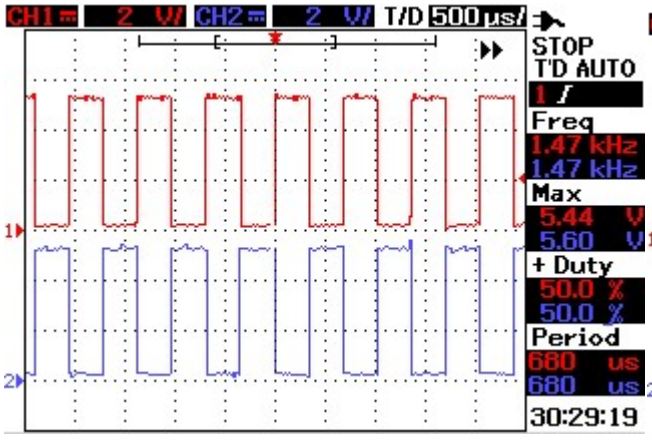


Figura 10 - Velocidade Máxima alcançada ligando diretamente os Arduinos.

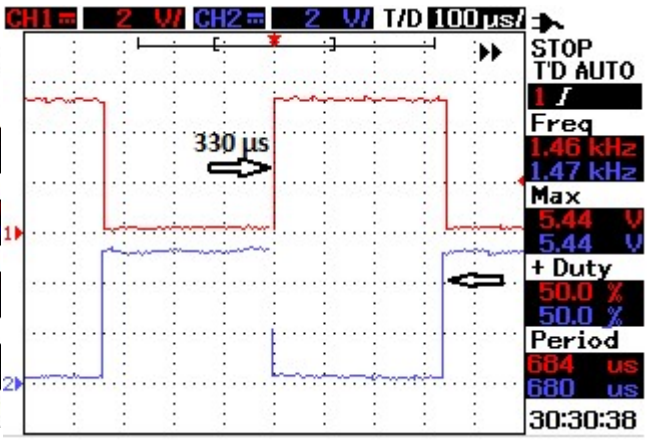


Figura 11 - Tempo de Resposta (delay) ligando diretamente os Arduinos.

Nota: Todas as Medições foram realizadas com um osciloscópio digital Agilent U1602B de dois canais

4 Discussão dos Resultados

Diante do exposto, infere-se que uma rede ZigBee é bem limitada quando se busca uma alta velocidade de comunicação. A frequência máxima de transmissão de comandos com quatro bytes obtida ao utiliza-lá foi de 67 Hz, que é 2,2% da frequência de comunicação alcançada ao conectar diretamente os Arduinos por meio de suas portas seriais (2,94 kHz), a qual pode ser considerada a maior frequência que pode ser alcançada utilizando essa configuração.

Além disso, vale mencionar o tempo de atraso da rede ZigBee foi de 10 ms, cerca de 30 vezes maior que aquele apresentado ao ligar diretamente os Arduínos.

Ainda é importante noticiar que o valor máximo de frequência obtido ao ligar diretamente os dois Arduinos ultrapassou o valor máximo teórico esperado, o qual foi calculado na Tabela 1.

5 Conclusão

Após os testes realizados, foi possível obter conhecimentos sobre uma rede ZigBee, além de verificar quantitativamente suas limitações. Mesmo que a frequência de comunicação alcançada nos testes (67 Hz), tenha sido mais que o dobro da que foi utilizada em trabalhos anteriores, 30 Hz (ROSÁRIO e CUNHA,2016), ainda assim não chega nem perto de aproveitar a total capacidade do sistema de câmeras VICON, que temos ao nosso dispor no laboratório de Controle e Automação da UERJ, que conseguiria realizar medições de posicionamento em taxas de até 1 kHz.

Além disso, após as experiências, inferiu-se que a limitação encontrada ao realizar o envio de comandos em (ROSARIO e CUNHA, 2016) era ocasionada diretamente pela Porta USB e o programa utilizado no trabalho, que era o MATLAB.

Dessa forma, para o decorrer do projeto resta duas opções:

(I) Começar o estudo de outros tipos de redes sem fio, que possuam maiores velocidades de comunicação do que a rede ZigBee;

(II) Aprimorar o trabalho já realizado, mantendo a rede ZigBee, reconhecendo sua utilidade quando se deseja um baixo consumo de energia e estudar métodos para aumentar o desempenho da embarcação, mesmo com a limitação de velocidade já conhecida.

Referências

BUORO, Angelo Sabbatini, 2013, **Controle dos Motores e Acionamento de uma pequena Embarcação**, Projeto de Graduação em Engenharia Eletrônica – UERJ.

RAMOS, Jadeilson de Santana bezerra, 2012, **Instrumentação eletrônica sem fio: transmitindo dados com módulos XBee Zigbee e PIC16F877A**, Editora ÉRICA Ltda.

ROSÁRIO, Rafael Vida de Castro, 2013, **Sistema para monitoração de uma embarcação não tripulada**, Projeto de Graduação em Engenharia Eletrônica - UERJ.

ROSARIO, R. V. C. ; CUNHA, J. P. V. S., 2016. **Experimentos de Rastreamento de Trajetória de uma Embarcação de Superfície Utilizando Linearização por Realimentação e Controle a Estrutura Variável**. In: XXI Congresso Brasileiro de Automática, 2016, Vitória. Anais do XXI Congresso Brasileiro de Automática, 2016. p. 1-6.

SOUZA, Lenielson Rodrigues, 2016, **Acionamento dos Motores CC de uma Embarcação Teleoperada**, Projeto de Graduação em Engenharia Eletrônica – UERJ.

APÊNDICE A – Programa utilizado no Arduino BASE

```
//Autor: Ian Henriques de Andrade
//Aluno de Graduação em Engenharia Elétrica com Enfâse em Sistemas Eletrônicos
//Bolsista de Iniciação Científica
//Orientado por: Prof. José Paulo Vilela
//Data: 21/11/2016
int led=13;
void setup() {
pinMode(led,OUTPUT);
Serial.begin(115200); //Inicia a comunicação serial à velocidade de 115000
}
void loop() {
Serial.write('1');
Serial.write('2');
Serial.write('3');
Serial.write('4');
digitalWrite(led, HIGH); //acende o led para indicar que o 1º comando foi enviado
delay(X); // espera X milisegundos
Serial.write('1');
Serial.write('5');
Serial.write('6');
Serial.write('7');
digitalWrite(led,LOW); // apaga o led para indicar que o 2º comando foi enviado
delay(X); // espera X milisegundos
}
```

APÊNDICE B – Programa utilizado no Arduino REMOTO

```
//Autor: Ian Henriques de Andrade
//Aluno de Graduação em Engenharia Elétrica com Enfâse em Sistemas Eletrônicos
//Bolsista de Iniciação Científica
//Orientado por: Prof. José Paulo Vilela
//Data: 21/11/2016
char valores[4]={}; //vetor onde será armazenado o comando recebido
char liga[4]= {'1','2','3','4'}; //vetor utilizado para comparar comandos recebidos
char desliga[4]={'1','5','6','7'}; //vetor utilizado para comparar comandos recebidos
void setup()
{
  pinMode(13, OUTPUT);
  Serial.begin(115200); //inicia a comunicação serial à velocidade de 115200
}
void loop()
{
  if (Serial.available()>3){ //Espera ter pelo menos 4 caracteres na entrada serial
    if(Serial.read()=='1'){ //garante que o primeiro caracter lido seja 1
    valores[0]='1';        //após isso armazena o comando no vetor 'valores'
      for (int i=1; i<4;i++){
      valores[i]=Serial.read();}
    }
     if (valores[1]==liga[1] and valores[2]==liga[2] and valores[3]==liga[3]){
      digitalWrite(13,HIGH);
     }
    if (valores[1]==desliga[1] and valores[2]==desliga[2] and valores[3]==desliga[3]){
      digitalWrite(13,LOW);}
    }
}
```