UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**TUTORIAL DE INSTALAÇÃO E UTILIZAÇÃO DO SIMULADOR OPENWSN EM ANBIENTE WINDOWS**

**Vinicius Pozzobon Borin**

Santa Maria, Agosto de 2014

ÍNDICE

[1. DOWNLOAD DO OPENWSN 2](#_Toc396483928)

[2. PREPARANDO PARA A SIMULAR 3](#_Toc396483929)

[2.1. Compilando o firmware como uma extensão do Python 3](#_Toc396483930)

[2.2. Instalando e Configurando as interfaces de rede virtuais 6](#_Toc396483931)

[3. SIMULANDO 9](#_Toc396483932)

[3.1. Executando a Simulação 9](#_Toc396483933)

[3.2. Criando *links* 10](#_Toc396483936)

[3.3. Pingando um nó 11](#_Toc396483939)

[4. DEBUGANDO 12](#_Toc396483943)

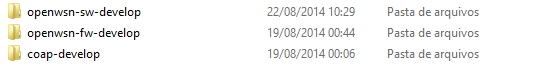
[4.1. Instalando o *Wireshark* 12](#_Toc396483944)

1. DOWNLOAD DO OPENWSN

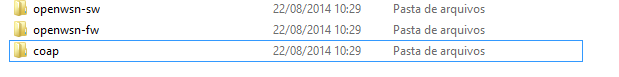
Baixe os arquivos do OpenWSN no Github e deixe todos na mesma pasta:

* <https://github.com/openwsn-berkeley/openwsn-fw> - firmware do OpenWSN
* <https://github.com/openwsn-berkeley/openwsn-sw> - Software do Simulador do OpenWSN
* <https://github.com/openwsn-berkeley/coap> - Módulo de Python que implementa o CoAP

Todos as pastas baixadas virão com os seguintes nomes:



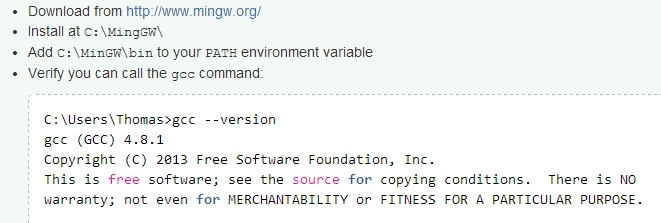
Neste momento, renomeie todas as pastas e tire o “-develop”, pois caso não faça isso, posteriormente ao executar o simulador ocorrerá um erro pois ele não achará as pastas. Deve ficar assim:



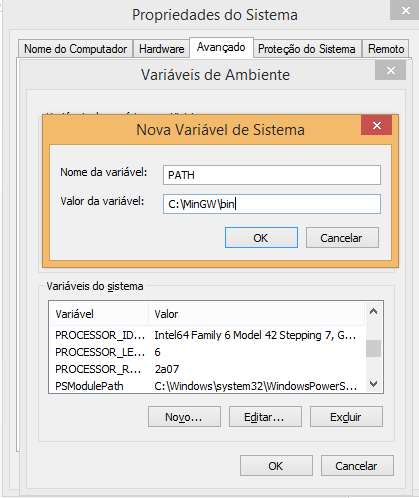
1. PREPARANDO PARA A SIMULAR

## 2.1. Compilando o firmware como uma extensão do Python

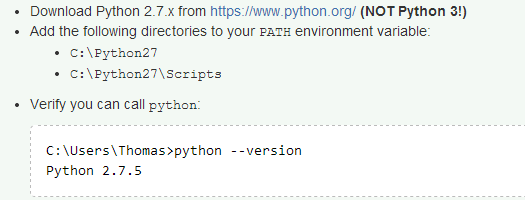
*2.1.1 Instale o mingw*



ATENÇÃO: conforme descrito acima, não esqueça de adicionar a variável *PATH* no sistema. Para isto, aperte com o botão direito em “Computador” e vá em propriedades. Depois vá na aba “Avançado” e em seguida em “Variáveis de Ambiente”. Uma nova janela abrirá, então onde diz “Variáveis do Sistema” insira uma nova variável do sistema, conforme a imagem abaixo:



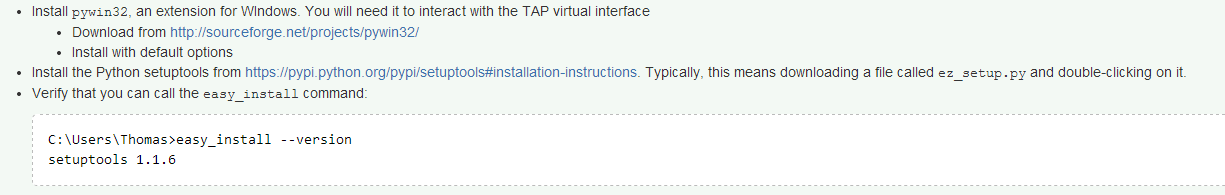
*2.1.2 Instale o Python:*



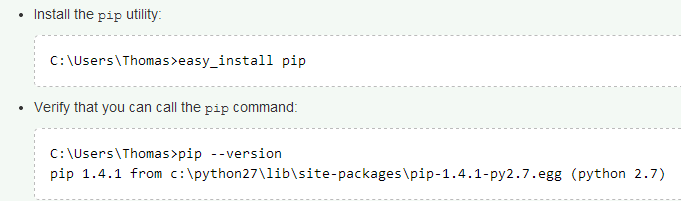
ATENÇÃO: conforme descrito acima, não instale o Python 3, mas sim o 2.7.x.

*2.1.3 Instale o pywin32*

*Use* este *link* aqui para baixá-lo, e escolha a versão correta para seu sistema operacional: <http://sourceforge.net/projects/pywin32/files/pywin32/Build%20219/>

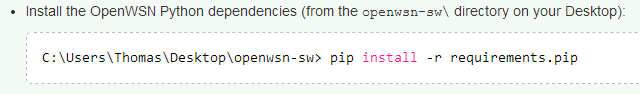


*2.1.4 Instale o pip*



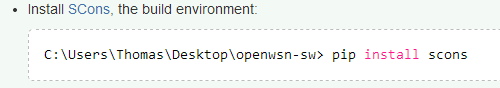
*2.1.5 Instale algumas dependências do Python*

ATENÇÃO: Você deve reproduzir este comando dentro do diretório *openwsn-sw* e não no diretório padrão que estava realizando os outros comandos.



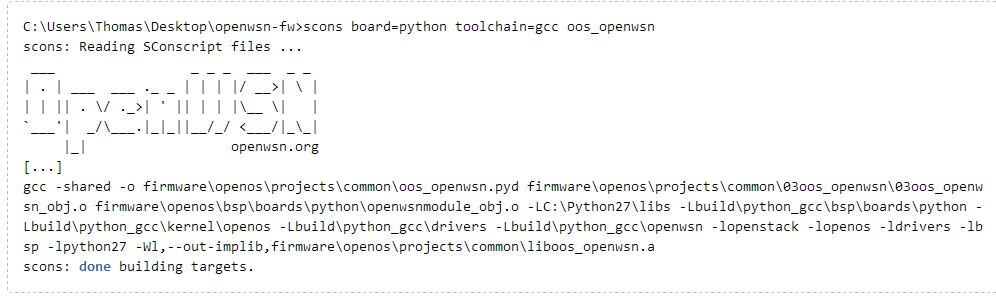
*2.1.6 Instale o scons*

ATENÇÃO: Continue dentro do diretório *openwsn-sw* que você executou o comando anterior.



*2.1.7 Compilando o firmware como uma extensão do Python (FINALMENTE)*

ATENÇÃO: Troque de diretório para execução esta etapa, entre no diretório *openswn-fw* e faça o comando descrito abaixo:



## 2.2. Instalando e Configurando as interfaces de rede virtuais

Agora é necessário criar uma rede virtual que será utilizada posteriormente pelo simulador para emular a rede de sensores. Para esta parte foi utilizado as seguintes placas de rede:

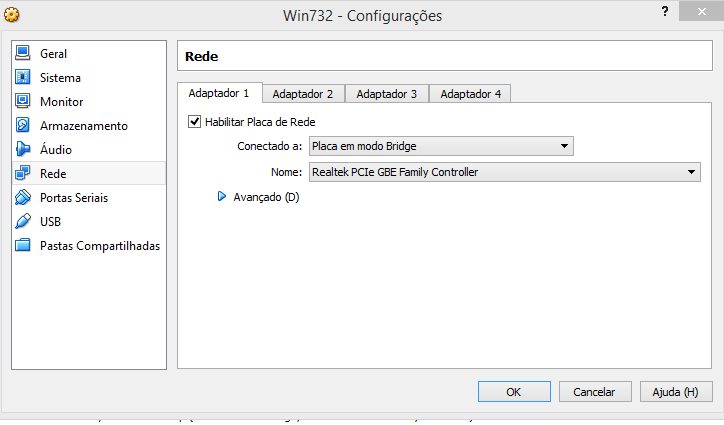
1. Adaptador *Ethernet* do computador;
2. Adaptador *Ethernet* virtual *tun/tap* (será mostrado como criar posteriormente);
3. Adaptador *Ethernet* em uma máquina virtual rodando *Windows* (ou *Linux*).

*2.2.1 Instalação da interface tun/tap*

Primeiramente, instale a interface *tun/tap* através da descrição desta página: <https://openwsn.atlassian.net/wiki/display/OW/Install+tun>. Atenção: renomeie o adaptador de rede *tun/tap* no seu computador para “OpenWSN” para que ele funcione posteriormente.

*2.2.2 Configuração da máquina virtual*

Utilizou-se o *software* VirtualBox para emular o *Windows* virtualmente. Para configurar o adaptador de rede do VirtualBox, selecione a máquina virtual com seu Windows virtual e aperte em “Configurações”. Na janela que abrir, vá em “Rede” e configure o “Adaptador 1” para o modo *Bridge*, conforme a figura:



A configuração *Bridge* deve ser utilizada pois assim a máquina virtual é reconhecida como sendo um computador independente de da rede. Caso deixe a configuração padrão, que é do tipo NAT, sua rede não funcionará

*2.2.3 Configurando e pingando as placas de rede*

As placas de redes devem ser configuradas de modo semelhante ao mostrado no esquema abaixo.



Após estas configurações você já pode testar o *ping*. Para isto vá na máquina virtual e “pingue” a placa de rede física, e vice-versa:

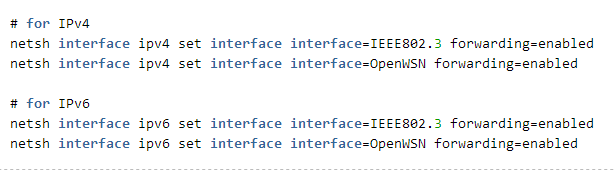
|  |  |
| --- | --- |
| Na máquina virtual | Na máquina física |
| ping 192.168.1.151 | ping 192.168.1.150 |
| ping aaaa::2 | ping aaaa::1 |

Todos os 4 comandos de *ping* deverão ser executados com sucesso, caso contrário temos algum problema. **ATENÇÃO: caso não consiga pingar, especialmente nos comandos de IPv6, o problema poderá estar no *firewall* do *Windows*, DESATIVE-O, tanto na sua máquina física quanto na virtual, e tudo deve funcionar.**

Caso tente pingar entre a interface *tun/tap* e alguma outra placa de rede não será possível, mas não se assuste, só será possível quando o simulador estiver funcionando.

*2.2.4 Habilitando o encaminhando entre a placa física e a tun/tap*

É necessário realizar os comandos abaixo para permitir que ambas interfaces na sua máquina real consigam encaminhar os dados entre elas. Não esqueça de substituir os nomes das suas interfaces nos comandos abaixo.



*2.2.5 Criando uma rota para o IPv6*

Adiciona uma rota para o IPv6 funcionar. Sem isto nada funcionará posteriormente.



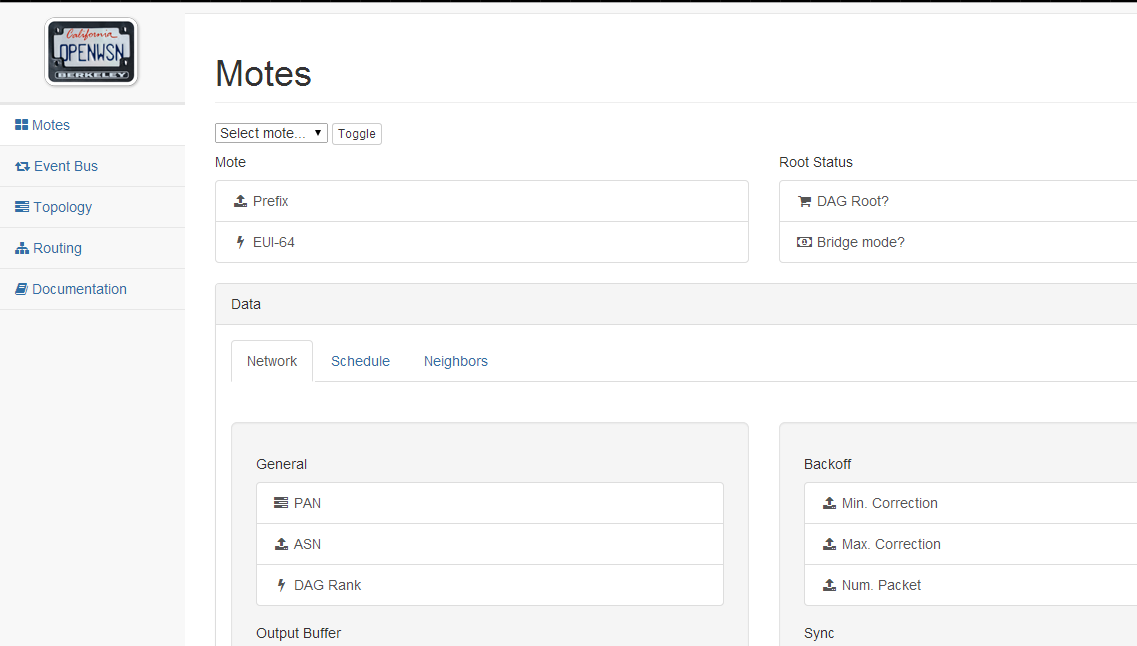
1. SIMULANDO

## Executando a Simulação

Caso todos os passos anteriores tenham sido concluídos com sucesso, você já pode iniciar o simulador. Para isto, acesse a pasta “openwsn-sw\software\openvisualizer” e execute o comando abaixo:

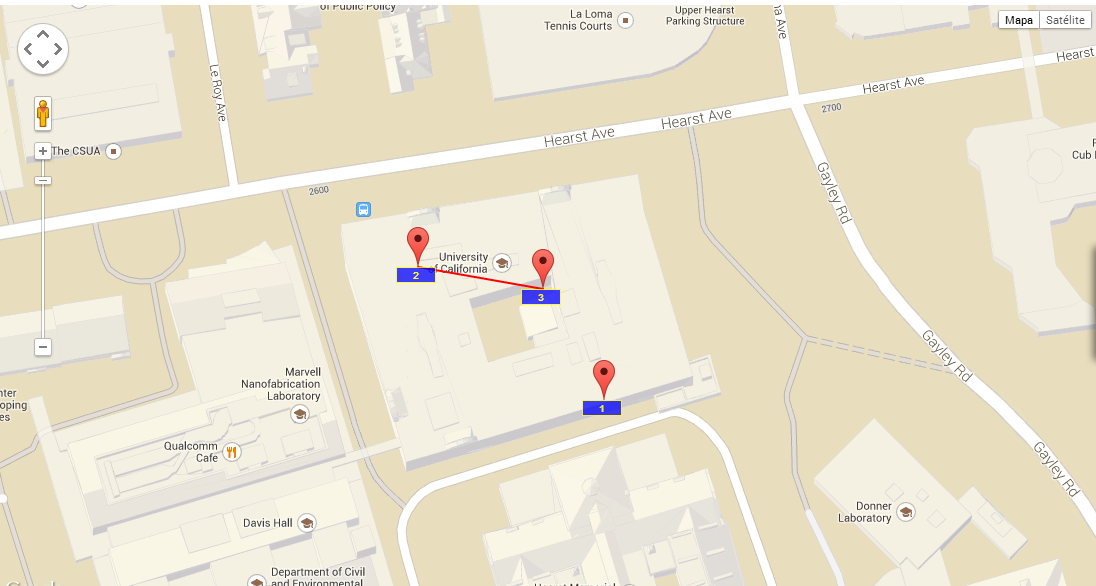


Para acessar o terminal do simulador via *browser* da rede, acesse [**http://127.0.0.1:8080**](http://127.0.0.1:8080/). Abrirá a seguinte janela:

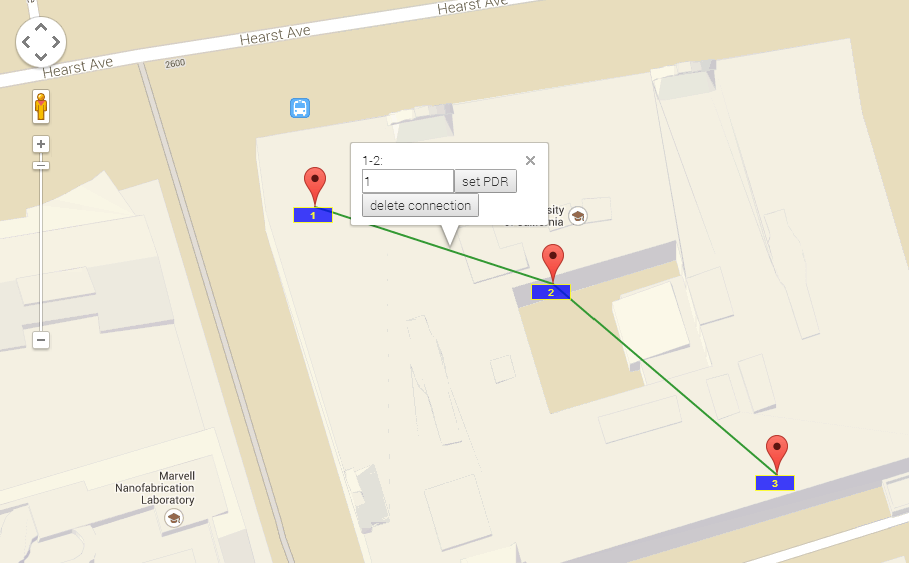


## 3.2. Criando *links*

No simulador, vá na aba “Topology”. Deverá abrir um mapa semelhante a este:



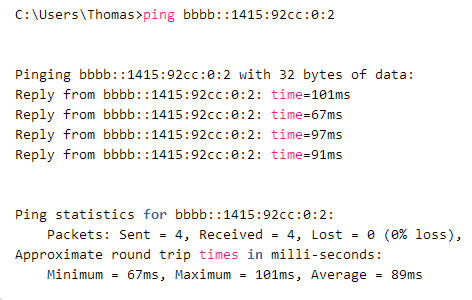
Para adicionar um *link*, clique com o botão direito em um marcador vermelho e depois novamente com o botão direito em outro marcador. Uma linha irá surgir entre eles representando um *link*. Crie um *link* entre 1-2 e 2-3. Depois, dê um duplo clique na linha e coloque o valor do PDR para 1, representando um *link* ideal. Deverá ficar assim:

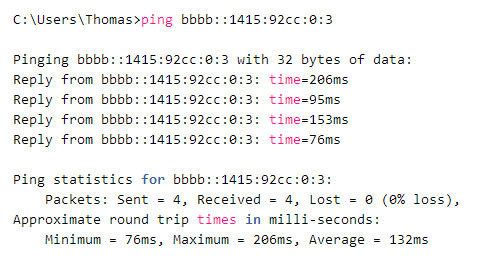


## 3.3. Pingando um nó

No simulador, vá na aba “*Motes*”. Selecione o nó 0001 e aperte em “*Toggle”*. Aperte uma única vez, pois mais de uma resultará em uma falha geral na simulação. Agora você poderá pingar os outros nós da rede. Pingue os nós 2 e 3. caso tenha configurado os IPs conforme este tutorial, os valores a serem pingados são os mesmos abaixo.

Para ver o endereço de IP de cada nó, selecione os *Motes* 0002 e 0003 no simulador e encontre-o. **ATENÇÃO: abra um novo *prompt* de comando para realizar os pings, nunca feche o *prompt* da simulação, pois isto irá encerrá-la.**



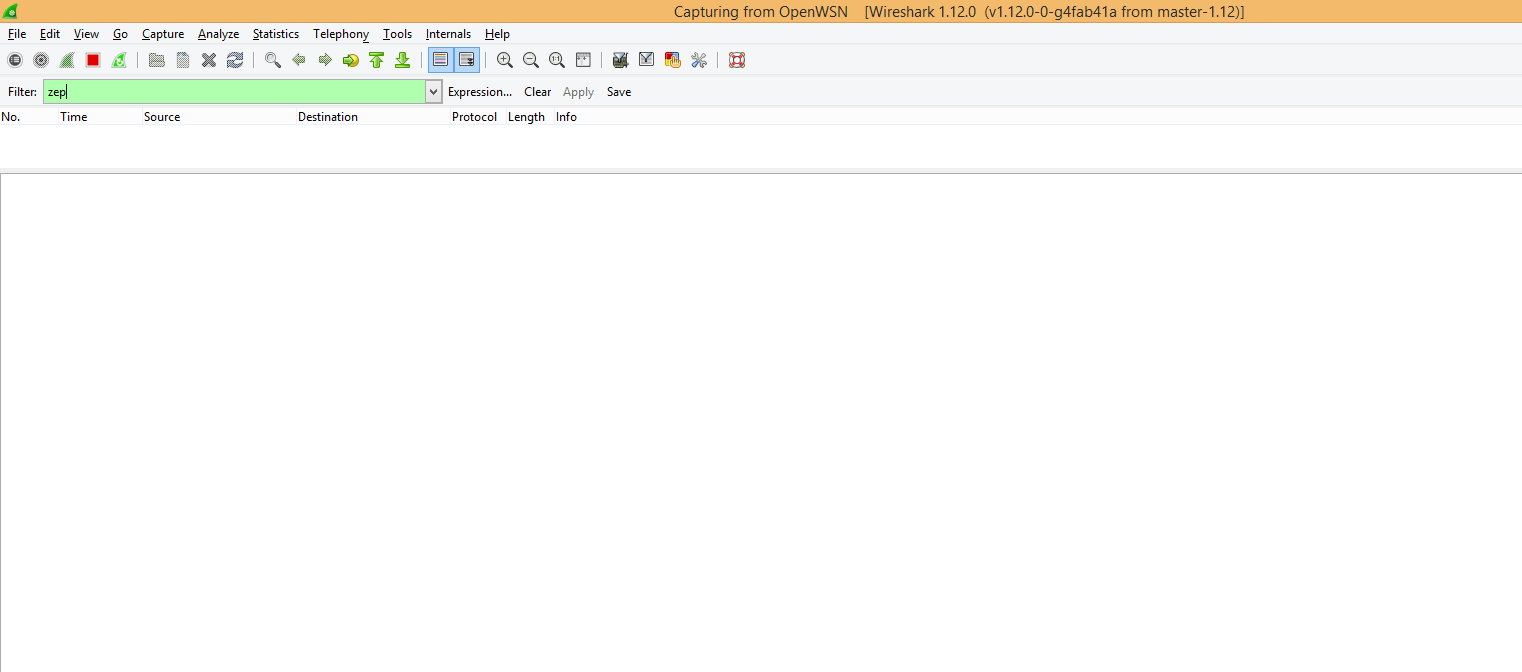


1. DEBUGANDO

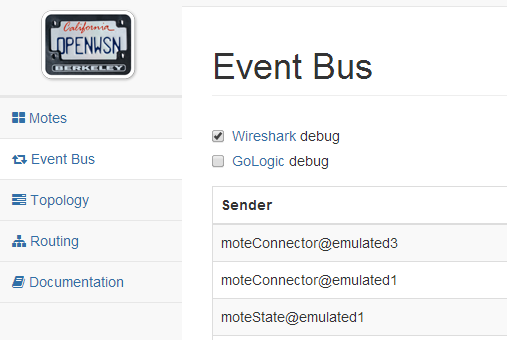
## 4.1. Instalando o *Wireshark*

O Wireshark mostra o comportamento do ambiente do radio. Baixe e instale o programa com as configurações padrão: <http://www.wireshark.org/>

Inicie o programa e selecione o adaptador de rede do OpenWSN e depois aperte em *Start*. Na janela que abrir, digite no buscador a palavra “*zep*”, conforme a imagem abaixo. *Zep* representa o protocolo de encapsulamento *Zigbee.* Ao digitar, nada irá aparecer, mas isto é normal neste momento.



Agora, volte ao simulador. Não feche o *Wireshark*. Vá na aba “*Event Bus*” do simulador e marque a caixa “*Wireshark debug*”:



Novamente, volte ao Wireshark. Agora deverá estar aparecendo os pacotes trocados no ambiente do rádio simulado.

