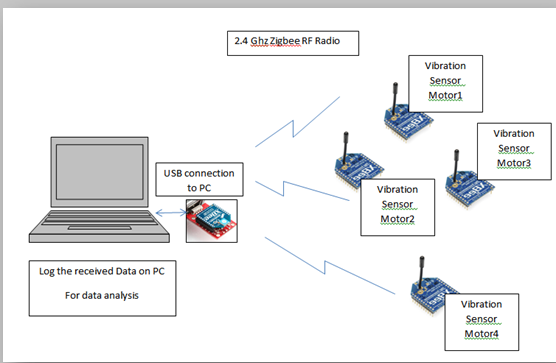
**ZigbeePRO Prototype documentation**

**Scope**: The scope of this document covers the details about the need for setting up a prototype to test WSN performance using zigbee-PRO Mest protocol and some sensors that feed data to a main computer over the serial port. The data analysis on the computer is not a part of the scope.

**Prototype setup requirements:**

1. Four XBEE-PRO S2B series ( XBP24-BZ7WIT-004 ) – 4nos
2. 4 Arduino UNO-R3 (or any similar controller)
3. 4 Vibration sensors (analog or digital)
4. 1amp power supply units to power the Arduinos
5. 4 XBEE Explorer dongles to program the XBEE-PRO S2B using Digi’s XCTU software.

**Prototype Setup block-diagram:**

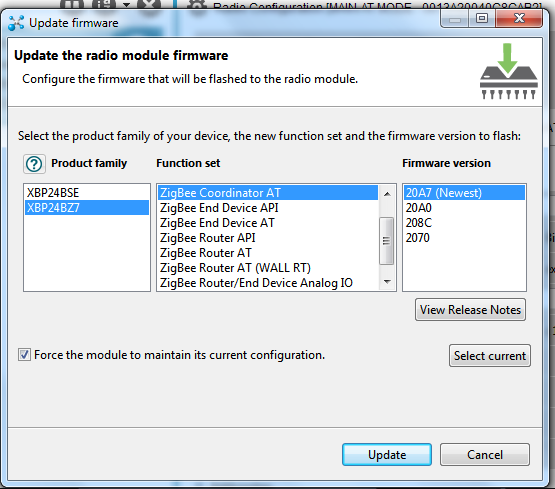


**Configuring the XBEE-PRO (Series 2):**

XBEE-PRO (Series-2) is a EM357 Silabs (previously ember) based Zigbee microcontroller with radio on the same SoC. XBEE-PRO can be used for a variety of wireless communication with a wide variety of wireless protocols like 802.15.4, Peer-to-peer wireless communication, ZigbeePRO-Mesh etc. In this prototype we will be using ZigbeePRO mesh protocol.

XCTU is a software provided by Digi (manufacturer of XBEE-PRO series modules). We are using Version: 6.3.1, Build ID: 20160318-3 of XCTU. Each XBEE-PRO module has be programmed using the below steps individually. To program the XBEE-PRO module, insert the XBEE-PRO into the usb explorer dongle. It gets detected by the XCTU software.

* XBEE-PRO modules can be used in AT mode (referred as transparent mode) or in API mode. In our prototype, we will be using all our XBEE-PROs in API mode.
* All XBEE-PRO modules must be configured to the same PAN ID. This is done via the XCTU software. After setting the PAN ID to ‘25’ ensure to click on “Write” button to save the settings to the XBEE-PRO>
* Only one XBEE-PRO module will act as the Coordinator while the other three XBEE-PROS will be configured as Router. (all XBEE-PROs should be updated to API mode).
* XBEE-PRO module comes in factory default state as Zigbee-PRO Router-AT mode. Hence the firmware needs to be changed on each of the XBEE-PRO module to its respective API firmware.
* To update the firmware of the XBEE-PRO, use the XCTU software and click on the update button.

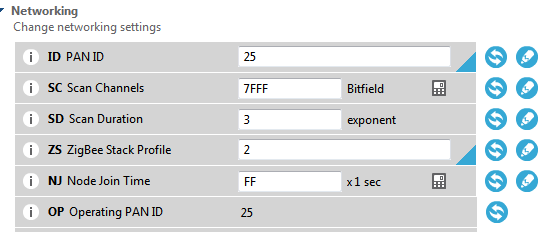


1. Select the product family as “XBP24BZ7” and for the coordinator node select “Zigbee CoordinatorAPI” function set and flash it with the latest firmware version provided. ***(If there is a new release from Digi then it is automatically updated in the XCTU menu.)***
2. Repeat the above operation for Router node by selecting the “ZigbeeRouterAPI” function set. The product family for the router node also is “XBP24BZ7” for our selected XBEE-PRO part number.

Note: Ensure to click the “Force the module to maintain its current configuration” checkbox. This ensures the settings of the module stay the same even after updating/changing the firmware. This ensures the XBEE-PRO module maintains the same PAN ID, Encryption status settings etc.

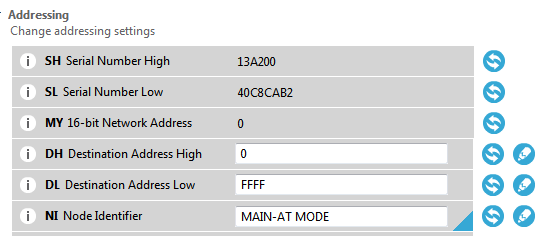
**Important Configuration settings for Coordinator Node:**

* Coordinator node initiates the mesh. In order to ensure all the nodes connect to this coordinator the PAN ID must stay the same on all the nodes. In this prototype, we have selected our PAN ID as 25.
* We have left the scan channel to factory default as 0x7FFF.
* Ensure to change the ZS Zigbee Stack Profile to 2. This ensures that the node uses the ZigbeePRO mesh protocol.
* After setting the parameters, ensure to click on the “Write” button on the XCTU and later “Read” button to check if all your settings are intact in the flash of the XBEE-PRO module.
* On the coordinator node the OP Operating PAN ID value *(read-only)* will show the same value as we set as our PAN ID.



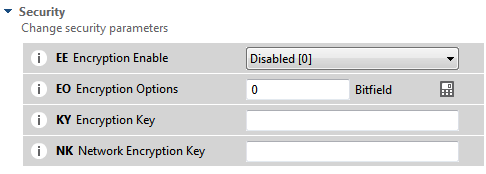
* Ensure to set the destination address (High) to 0x00 and destination address (Low) to 0xFFFF. 0x00000000 0000FFFF is the broadcast address. Setting this address on any node (whether coordinator or router or end node) will make that node to broadcast its message to all nodes in its network.
* The 16-bit network address (MY) of the coordinator is always 0 while for other nodes like router node or end node this would any 16 bit hex value but not 0. See the figure below to ensure our coordinator’s MY value is 0, DL value is 0xFFFF.
* **The SH and SL gives the MAC id for the XBEE-PRO module. The same is also printed on the module.** This id is unique to each XBEE-PRO module and no two MACs in the world will be same**.** In the API mode, the node will use this MAC id address to send the message to another specific node instead of broadcasting to all whenever it wants to send/receive message from a specific node only. (*In our prototype we have used broadcast mode only).*
* Give your Coordinator a node identifier name as “MAIN-API-2”. This name is only for viewing purpose and not used in our prototype code. This name is only to identify the node by a verbose name for human readability. ***(This has no significance in the coordinator’s firmware operations)***

**Note:** In our prototype, we have configured our coordinator to broadcast its message to all nodes. This way the coordinator tells all the nodes to which it is connected about its intent to receive data from a specific node. For eg, Coordinator send a message calling out node N1 to send message to it. This message will be listened by all other nodes and only the node named “N1*” (in the code)* will pass its analog channel (A0) value back to the coordinator.



* You can encrypt the network using Zigbee-PRO’s inbuilt encryption method. To use it enable the encryption by setting the EE bit. Ensure to use the same encryption key on all the nodes (coordinator + router + end nodes on this PAN ID). Leave the network encryption key blank (default state).

Note: In our prototype code, we have kept the encryption disabled.



* The API mode-2 is mandatory for our prototype to work. Ensure to select AP as ‘2’ as per the below figure. The API output mode (AO) should be native.



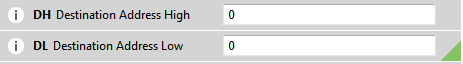
* Leave all other settings to factory default. **Ensure to click on “Write” button on the XCTU after you make your settings and read back to verify.**

**Important Configuration settings for Router Node(s):**

* Nodes connecting to the coordinator node may be either a router node or an end-node. End-node sleeps off and wakes up intermittently. If you are building a battery based zigbee node then it is important to update its firmware as End node (in API mode)

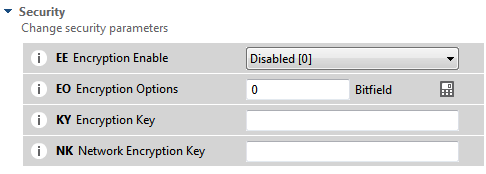
**Note:** In our prototype, we have kept all our nodes as router node. Even router nodes conserve power as per zigbee’s mesh protocol architecture at hardware level but it is less efficient as compared to an end node in terms of power consumption.

* In order to ensure all the nodes connect to our configured coordinator node the PAN ID must stay the same as set for the coordinator. In this prototype, we have selected our PAN ID as ‘25’. Our prototype uses 3 nodes as Router nodes.
* We have left the scan channel to factory default as 0x7FFF.
* Ensure to change the ZS Zigbee Stack Profile to 2. This ensures that the node uses the ZigbeePRO mesh protocol.
* After setting the parameters, ensure to click on the “Write” button on the XCTU and later “Read” button to check if all your settings are intact in the flash of the XBEE-PRO module.
* On all the router nodes check the OP Operating PAN ID value *(read-only)* should show the same value as set for the coordinator’s PAN ID.
* Ensure to set the destination address (High) to 0x00 and destination address (Low) to 0x00. Setting the DH (Destination address high) and DL (Destination address low) to 0x00 on all the router node ensures that this router node will send its message to the coordinator. ***(0x0000000000000000 can be used to address the Pan Coordinator.)Ensure to set the DH and DL on all our router nodes as 0x00 in our prototype.***



* The 16-bit network address (MY) of the router would be any 16 bit hex value but not 0.
* Give your router node identifier name as “RTR-1 API-2 mode”. This name is only for viewing purpose and not used in our prototype code. This name is only to identify the node by a verbose name for human readability. ***(This has no significance in the router’s firmware operations)***
* You can encrypt the network using Zigbee-PRO’s inbuilt encryption method. To use it enable the encryption by setting the EE bit. Ensure to use the same encryption key on all the nodes (coordinator + router + end nodes on this PAN ID). Leave the network encryption key blank (default state).

Note: In our prototype code, we have kept the encryption disabled.



* The API mode-2 is mandatory for our prototype to work. Ensure to select AP as ‘2’ as per the below figure. The API output mode (AO) should be native on all router nodes and all router nodes are to be set to API mode (AP) ‘2’.



* Leave all other settings to factory default. **Ensure to click on “Write” button on the XCTU after you make your settings and read back to verify.**

**Wiring setup for the prototype:**

1. Wiring of the XBEE-PRO to the Arduino board is the same for all router and coordinator nodes in our prototype.
2. On all the XBEE-PRO modules the pins are numbered on the PCB. **(Warning: XBEE-PRO modules work on 3.3V technology. Ensure to connect the 3.3V VCC from Arduino to power the XBEE-PRO as shown)**



1. Connect the following wires between Arduino and XBEE-PRO module for all nodes (both coordinator and router nodes):



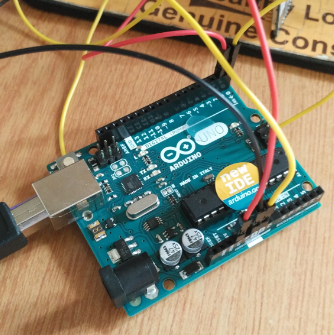
**XBEE-PRO side**  **Arduino side**

1. **3.3V VCC**
2. Pin-2
3. Pin-3

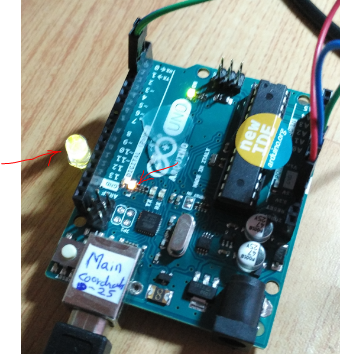
9(DTR) GND

10(GND) GND





1. The 5mm GREEN LED connected on the side of the Arduino board is the transmission/reception activity happening between Arduino code and the XBEE-PRO module. This LED blinks based on the activity. This GREEN LED is connected between PIN-12 and GND on the Arduino board in our prototype.
2. The onboard orange LED (SMD type) connected between PIN-13 and GND on Arduino blinks at periodic (1 sec) interval to indicate heart-beat of the Arduino board in our prototype.



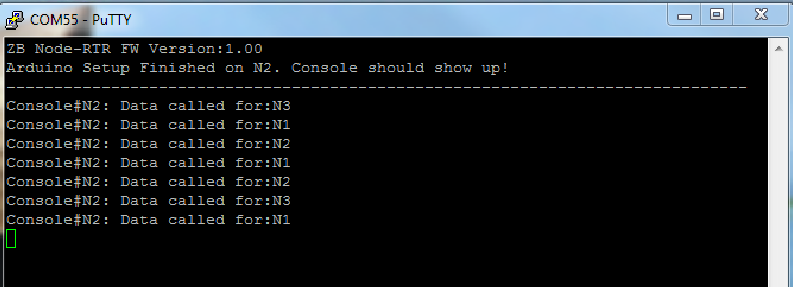
Heart-beat indicator

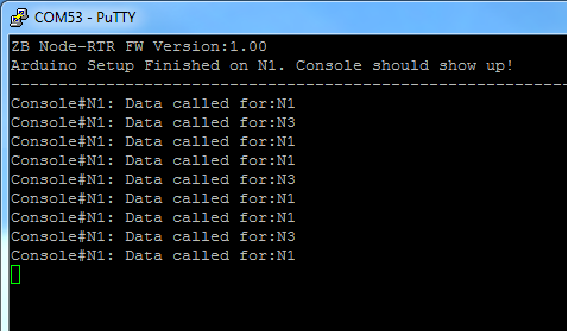
Activity indicator

**Switching on prototype and setting up:**

1. Connect the USB cable attached to the Arduino board connected to the Coordinator XBEE-PRO to a laptop. Open up the console based on the COM-port assigned by the PC to see the activity. (Use putty.exe software or an terminal/console viewing software).
2. Connect the vibration sensor (there are no polarity on this sensor so any of two wires can be connected to the GND while the other can be connected to the A0 pin marked on the Arduino). The data from the vibration sensor will be sampled by a 10 bit ADC on the Arduino-UNO board and sent via XBEE-PRO module to the coordinator’s console.
3. Vibration sensor voltage is checked and analysed for vibrations. Each vibration sensor has its own characteristics and based on the type of vibration the vibration frequency of the piezo between (1Hz to 16hz) is measured and converted to its appropriate voltage level by Arduino. (Note: Instead of analog sensor, digital accelerometer can also be used. Any sensor processing is part of the Arduino code and that does not affect the Zigbee-PRO wireless operation)
4. Record the console activity to store it as a txt file which can be later uploaded for analysis. Uploading to web is not a part of the scope in this prototype.
5. Console of each Router node will show up as shown. It will print the broadcasted message from the coordinator. This gives an indication to use as to which Node’s data has been requested by the coordinator. The coordinator node will request every sec a new node’s value. (This value is the analog channel-A0’s value to which the vibration sensor is attached.)

**Note:** Each routing node is labelled as N1, N2, N3 respectively in the Router firmware flashed to the Arduino. All the Arduinos connected to the Router type XBEE-PRO are flashed with RoutingNode\_Source while the Arduino connected to the XBEE-PRO configured as the coordinator is flashed with CoordinatorNode\_Source.

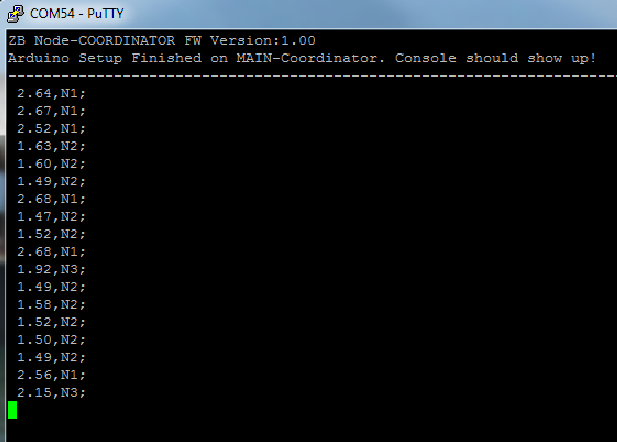




1. Console of coordinator node will show up as shown. It will print the received message from different router nodes. Only one router node will talk to the Coordinator at any point in time. The coordinator calls the respective router node (as N1, N2, N3) to send its data. According to this call the respective router node responds with a message containing the vibration sensor’s value back to the coordinator.

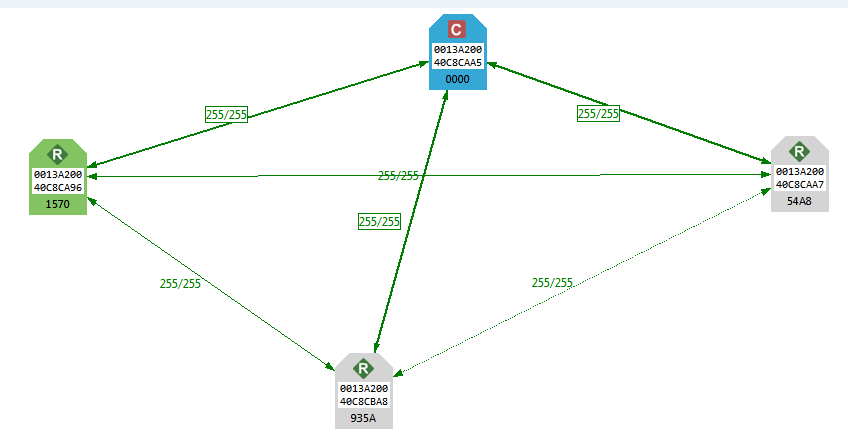
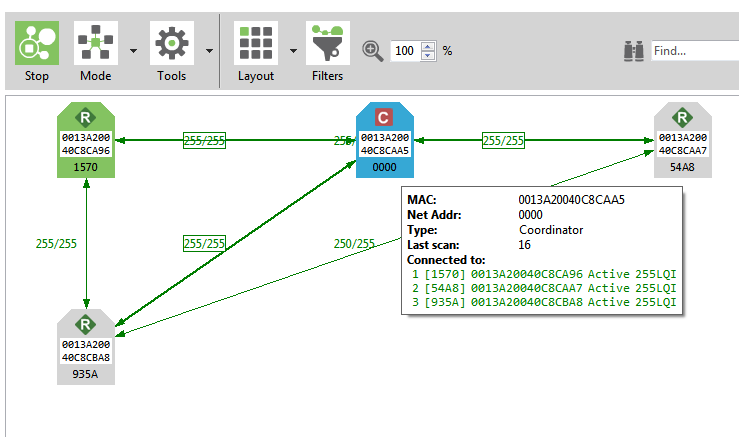
The message received at the Coordinator’s console is specially framed for processing and separating each node’s respective value for offline processing.

**Assume each router node (N1, N2, N3…) is measuring a different motor’s vibration and sending the values to the laptop where offline processing of this data will provide health status of each motor.**



**XCTU node scan after all nodes are connected/paired.**

* The figure below is generated by the Digi’s XCTU software to show in real-time the connected nodes. The Node labelled as ‘C’ is the coordinator while the one labelled as ‘R’ is the router node.
* Each node’s MAC address is displayed apart from the number of connections made by each node. The numbers like 1570, 54A8, 935A denote the 16-bit short address of respective nodes.



**Folder Setup**

In the folder ZB\_VibrationPrototype, there are three folders:

1. Folder “CoordinatorNode\_Source” contains the firmware code for the Arduino that is to be connected to the coordinator node XBEE-PRO. This folder contains the source code in txt, in .ino (Arduino file format) and in pdf format.
2. Folder “RoutingNode\_Source” contains the firmware code for the Arduino that is to be connected to the router node XBEE-PRO. This folder contains the source code in txt, in .ino (Arduino file format) and in pdf format.
3. Folder “Tools” contains the Arduino IDE setup file, putty.exe console utility, Digi’s XCTU software that was used to build this prototype.

**Trouble-shooting Guide**

Upon power-on everything should work normally. Allow around a 1 minute for nodes to connect upon power on. If they don’t connect there will no message transfer between the coordinator and respective router nodes. Here are a few things to try out to get the system working.

1. Check the API output mode is native and AO value is set to 0. To check this you need to plug the respective XBEE-PRO module into the USB explorer dongle to check in XCTU software.
2. Check the PAN id is set to the same value on all nodes.
3. Wiring connections are not lose and connected correctly as per the pin assignments described in this document.
4. The XBEE-PRO module firmware is a closed source from DIGI and we don’t get access to it. If you have XCTU software then you can flash it. Ensure you are using API mode firmware for the XBEE-PRO. (The default API mode is always ‘1’ so ensure you set it to ‘2’)
5. Sometimes the console may be in use by the PC and not properly released. So try to reset the console or laptop if you notice such a problem. It may happen that the hardware setup of Arduino and XBEE-PRO are working fine but you don’t get any data on the console attached to that Arduino. If this problem occurs you can be assured that the Arduino and XBEE-PRO are communicating with other nodes by watching the intermittent blinking of GREEN LED.

**Arduino Firmware flashing Guide for (Routing nodes)**

The firmware to be flashed on the Arduino connected to the router nodes needs to prepared for flashing as some minor modifications like labelling the Routing firmware for each node is to be done (like for eg N1, N2, N3 prior to flashing the respective routing Arduino firmware).

* Change the NODE\_SR to ‘1’ or ‘2’ or ‘3’ as per the Routing firmware you are trying to flash to the respective Arduino.
* Change the NODE\_NAME to “N1” or “N2” or “N3” as per the routing firmware you are trying to flash to the respective Arduino.
* ***This prototype firmware supports maximum of three router nodes but there is nothing in the firmware that limits this number. You may increase the nodes if you wish.***



* If you wish to see the frame-level details, then enable the DEBUG\_FRAME\_LEVEL by setting it to 1. This gives information such as frame type, node’s 64 bit mac ID and its 16 bit short address. (Eg: #define DEBUG\_FRAME\_LEVEL 1)
* DEBUG\_INDEPTH is for debugging the firmware code. Generally not necessary to enable unless you are changing the firmware logic.



**Arduino Firmware flashing Guide for (Coordinator node)**

The firmware to be flashed on the Arduino connected to the coordinator nodes needs to prepare for flashing if you wish to increase the number of routing nodes. Currently this prototype supports maximum of three routing nodes. If you increase the routing nodes ensure to let the Coordinator firmware know the max nodes by setting this pre-processor directive:



**The rest of the firmwares of Arduino for Router and Coordinator are well commented in the code for easy understanding of the code. It uses XBEE API library, Soft-serial library and timerOne libraries. Soft-serial library enables using any digital pins as Tx and Rx pins while the timerOne library is used for setting up 16-bit timer-1 in our prototyping Arduino firmware.**