We design and manufacture Raspberry Pi based industrial hardware

We have two systems built around Digi XBEE S8 devices

XBEE-COM
XBEE-IO

Each of these can we used together with a central XBEE unit, connected to the host via either an USB-to-UART (i.e. USB COM port) interfaced mPCIe card or as direct plug HAT style card connecting directly to the RPi UART.

This gives you a local and a remote end

The XBEE-COM unit allows you to use the setup in “transparent serial mode”, either as a 1:1 or a 'hub-and-spoke' one to many arrangement (the latter requiring a bit more complex configuration but is possible). This unit is designed to connect up to equipment via either Serial RS232 or 2-wire RS485. Note that the RS485 transceiver we use here has built in, transparent, auto-direction control making RS485 easy as no additional control signals are needed.

The XBEE-IO unit uses the XBEE in "API mode" this is where you send the remote XBEE a command which it interprets and transmits back its response to the sender. With this it’s possible to have multiple remote XBEE-IO modules and poll them each for their current analogue and digital data readings, the XBEE-IO module also has outputs so the remote XBEE unit can be told to switch a relay on or off.

In both modes the modules operate in "digi-mesh" comms mode, this is useful as it extends the reach of the modules meaning that the XBEE units can be far away from the central hub/host unit and the communication packets will be routed between the modules so long as a route can be established.

The XBEE-COM unit requires very minimal setup is required, other than baud rates just enough so the base unit knows the remote unit's address ID, and likewise the remote unit knows the base unit's ID). You just treat the local XBEE like it’s a standard serial port connected to the target equipment with a very, very long cable ;-)

Make sure when you reply you use the phrase ‘You asked for intelligent design, here’s my quote’ so I know you’ve read this

API usage requires a bit more time and effort as you have to write some python code to request data and decode the results using open-source libraries. The XBEE API model also allows for some fairly advanced tricks like configuring the device to "send me a value when x input goes above this threshold" etc

There are other neat features like being able to resolve an XBEE address from its user defined name

Have a look at these links for details

Flyer
https://www.digi.com/products/embedded-systems/rf-modules/sub-1-ghz-modules/xbee-sx-900
https://www.digi.com/pdf/ds-xbee-xbeepro-sx-modules.pdf

XBEE SX Unit Manual
https://www.digi.com/resources/documentation/digidocs/pdfs/90001538.pdf

Digi Overview documentation on API/Transparent modes:
https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_transparent_and_api_mode.htm
https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_transparent_mode_detailed.htm
https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_serial_communication.htm

Target XBEE Python Lib
https://github.com/niolabs/python-xbee

The problem we are facing is that the python library above is incomplete when using digimesh based protocols.

Attached are some scripts that we created that use the incomplete python lib, take a look at the bug report for the IS command in demo.py especially as this is missing the decode functionality.

The first part of the project is to fork and extend the exiting library to get the IS and other data reading functionality working correctly for digimesh format protocol and allow the demo scripts to be finished off

Further milestones are

1) Tidy up the demo scripts

2) Create new transparent serial mode scripts to setup multiple XBEE-COM endpoints in transparent serial comms mode so it is possible to demonstrate 1:Many serial comms

Using the mbpoll program and windows modbus slave simulators – note RS485 is not required here, basic RS232 will work fine:

https://embeddedpi.com/documentation/isolated-rs485/raspberry-pi-modbus-mbpoll-linux-installation

https://embeddedpi.com/documentation/isolated-rs485/mypi-industrial-raspberry-pi-modbus-rs485-read-write-command-line-demo

3) Create a basic python logging program to read values from multiple XBEE-IO modules logging to a file in semicolon delimited format