XBee UART Python functions

Chris Vincent Densing

October 28, 2019

Part I
Packet encode functions: Generic

Includes functions for formation of packets that conform to the XBee API format. Included in file *packet_encode.py*.

1 AT Command Query (atcom_query)

Assemble a byte stream for querying AT parameter values. Use on local XBee device connected to serial.

1.1 Returns:

1. *bytestr* - (type: Bytes) Formatted XBee API frame (0x08: AT Command frame).

1.2 Arguments:

1. *param* - (type: String) Two-character AT parameter.

2 AT Command Set (atcom_set)

Assemble a byte stream for setting AT parameter values. Use on local XBee device connected to serial.

2.1 Returns:

1. *bytestr* - (type: Bytes) Formatted XBee API frame (0x08: AT Command frame).

2.2 Arguments:

1. *param* - (type: String) Two-character AT parameter.
2. *value* - (type: Bytes) New AT parameter value.

3 Transmit request (gen_txreq)

Assemble a byte stream for a transmit request payload. Use on local XBee device connected to serial. Returned stream still needs to have headers and checksum appended (use in conjunction with gen_headtail()). Refer to XBee documentation for details on arguments.

3.1 Returns:

1. *bytestr* - (type: Bytes) Semi-formatted XBee API frame (0x10: Transmit request frame).

3.2 Arguments:

1. *fid* - (type: String) 1-byte Frame ID in hexadecimal.
2. *dest* - (type: String) 8-byte Destination address in hexadecimal.
3. *brad* - (type: String) 1-byte Broadcast radius in hexadecimal.
4. *opts* - (type: String) 1-byte Transmit options in hexadecimal.
5. *data* - (type: String) Variable length payload in hexadecimal.

4 Generate Header and Checksum (gen_headtail)

Append headers (0x7e and 2-byte length) and checksum to a generic payload. Use on local XBee device connected to serial.

4.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

4.2 Arguments:

1. *bytestr* - (type: Bytes) Assembled payload without headers and checksum.

5 Transmit request arbitrary payload (msgformer)

Generates a transmit request API frame with an arbitrary message. Other required fields are set to the following:

- Frame ID - 0x01
- Broadcast radius - 0x00
- Transmit options - 0x00

Assembled byte stream already contains appropriate headers and checksum.

5.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

5.2 Arguments:

1. *msg* - (type: String) Variable-length arbitrary message in hexadecimal.
2. *dest* - (type: Bytes) 8-byte address of destination node.

Part II
Packet encode functions: Specific

Includes functions for formation of command packets specific to the RESE2NSE node 1 firmware. Included in file *packet_encode.py*.

6 Remote node unicast (debug_unicast)

Assembles command packet for issuing a unicast command to a remote node in standby/debug mode.

6.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

6.2 Arguments:

1. *n* - (type: Integer) Number of unicast packets remote node will transmit back.
2. *dest* - (type: Bytes) 8-byte address of remote node.

7 Remote node set channel (debug_channel)

Assembles command packet for changing the radio channel of a remote node. Command is processed by the MSP430.

7.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

7.2 Arguments:

1. *ch* - (type: Bytes) 1-byte new channel.
2. *dest* - (type: Bytes) 8-byte address of remote node.

8 Remote node set power level (debug_power)

Assembles command packet for changing the radio transmit power of a remote node. Command is processed by the MSP430.

8.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

8.2 Arguments:

1. *pow* - (type: Bytes) 1-byte new power level.
2. *dest* - (type: Bytes) 8-byte address of remote node.

9 Remote node start sensing (start_sensing)

Assembles command packet for transitioning from stanby/debug mode into sensing mode of a remote node.

9.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

9.2 Arguments:

1. *period* - (type: Integer) 1-byte sampling period.
2. *dest* - (type: Bytes) 8-byte address of remote node.

10 Remote node stop sensing (stop_sensing)

Assembles command packet for transitioning from sensing mode to stanby/debug mode of a remote node.

10.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

10.2 Arguments:

1. *dest* - (type: Bytes) 8-byte address of remote node.

11 Remote node query parameter (debug_query)

Assembles command packet for querying remote node parameters. This includes AT parameters (must be implemented in remote node firmware) and MSP430 parameters.

11.1 Returns:

1. *bytestr* - (type: Bytes) XBee API frame.

11.2 Arguments:

1. *atcom* - (type: String) Parameter to query.
   Supported parameters:
   (a) PL - power level
   (b) CH - channel
   (c) A - aggregator address
   (d) T - sampling period
2. *dest* - (type: Bytes) 8-byte address of remote node.

Part III

Packet decode functions

Includes functions for receiving and decoding packets from XBee through UART.

12 Receive packet (rxpacket)

Read the UART buffer for an XBee API frame. Checks the frame for length and checksum and returns the payload sans headers and checksum.

12.1 Returns:

1. success - (type: Integer) Returns a 1 for a successful read, 0 otherwise.
2. payload - (type: Bytes) XBee API frame payload.

12.2 Arguments:

1. ser - Serial interface returned from serial.Serial() function of pyserial.

13 Decode AT Command Response (decode_atcomres)

Decodes AT Command Response (frame 0x88)payload from rxpacket(). Prints the following decoded fields to the console:

- Frame ID
- AT Command
- Command Status
- Command Data (if available)

13.1 Returns:

1. error - (type: Integer) Returns a 0 when the status field of the received packet is 0 ('OK'). Returns 1 otherwise.

13.2 Arguments:

1. payload - (type: Bytes) XBee frame sans headers and checksum.

14 Decode Transmit Status (decode_txstat)

Decodes Transmit Status (frame 0x8b) payload from rxpacket(). Prints the following decoded fields to the console:

- Frame ID
- 16-bit Destination Address
- Transmit Retry count
- Delivery status
- Discovery status

14.1 Returns:

1. error - (type: Integer) Returns a 0 when the delivery status field of the received packet is 0 ('OK'). Returns 1 otherwise.

14.2 Arguments:

1. payload - (type: Bytes) XBee frame sans headers and checksum.

15 Decode Received Packet (decode_rxpacket)

Decodes a received packet (frame 0x90) payload from rxpacket(). Prints the following decoded fields to the console:

- 64-bit source address
- Reserved field
- Receive Options
- Received Data

15.1 Returns:

None

15.2 Arguments:

1. payload - (type: Bytes) XBee frame sans headers and checksum.

16 Decode Payload (decode_payload)

Parses the first byte of a payload and calls the appropriate decoding function. Currently supports the following XBee API frame types:

- AT Command Response (0x88)
- Transmit Status (0x8b)
- Received Packet (0x90)

16.1 Returns:

- status - (type: Integer) forwards returns of decode_atcomres() and decode_txstat(). Returns 0 otherwise.

16.2 Arguments:

- payload - (type: Bytes) XBee frame sans headers and checksum

Part IV
Packet Decode with logging

Decode functions which log to a file.

17 Decode Generic Payload (decodelog_payload)

Decodes a generic payload and logs decoded fields to a file. Currently supports the following frames:

- Received packet (0x90)

The function prints a line (to a file) in the format "[f1] Raw: 0x [f2]" with the following fields:

1. Timestamp [f1]
2. Raw payload in bytes [f2]

The function then calls the corresponding decoding function that may print further parsed fields from the raw packet.

17.1 Returns:

None

17.2 Arguments:

1. fp - (type: File pointer) Output log file.
2. payload - (type: Bytes) XBee frame sans headers and checksum.

18 Decode Receive Packet (decodelog_rxpacket)

Decodes a receive packet frame (0x90). The function prints a line (to a file) in the format ”—- 90, [f1], [f2], [f3], [f4]” with the following fields:

1. 64-bit source address in hexadecimal [f1]
2. Reserved field in hexadecimal [f2]
3. Receive options in hexadecimal [f3]
4. Received data in hexadecimal [f4]

18.1 Returns:

None

18.2 Arguments:

1. fp - (type: File pointer) Output log file.
2. payload - (type: Bytes) XBee frame sans headers and checksum.