XBee configuraton

1. Download Digi XBee Studio and use 802.15.4 firmware in temporary. Also, remember to change the drone's SYSID in Mission Planner.

2. Make the XBee modules in the same CH(Channel) and ID(Pan ID).Change NI to identify the modules. Set the BD(Baud Rate) to 57600.

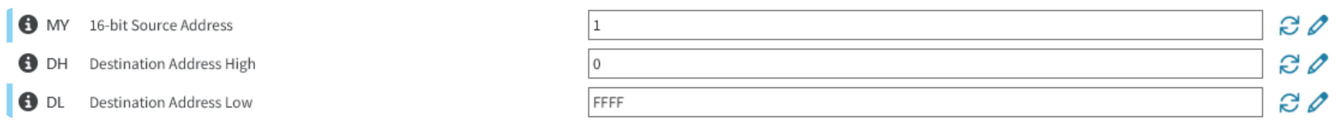
3. Coordinator: Set CE(Device Role=1) to coordinator and enable API(AP=1). Endpoint: Set CE(Device Role=0) to endpoint and set mode(AP=0) to transparent mode(AT mode).

802.15.4 only have coordinator and endpoints. And we set them in the different mode: API and AT mode.

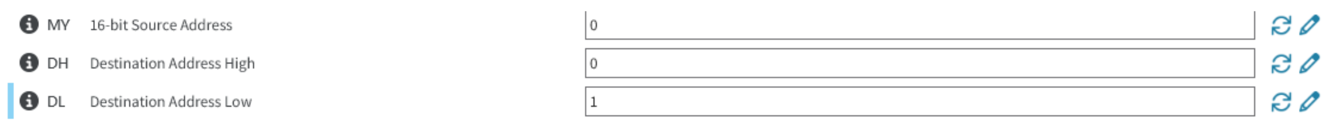
4. Coordinator: Set DL(Destination Address Low) to FFFF in broadcast mode.

5. There are two ways to address parameters for endpoints. Set DH、DL to host's SH、SL, or set the host's MY(16-bit Source Address). For example, set coordinator's MY=1. Then we can set endpoints' DL=1.

Coordinator


MY 16-bit Source Address 1
DH Destination Address High 0
DL Destination Address Low FFFF

Endpoint


MY 16-bit Source Address 0
DH Destination Address High 0
DL Destination Address Low 1

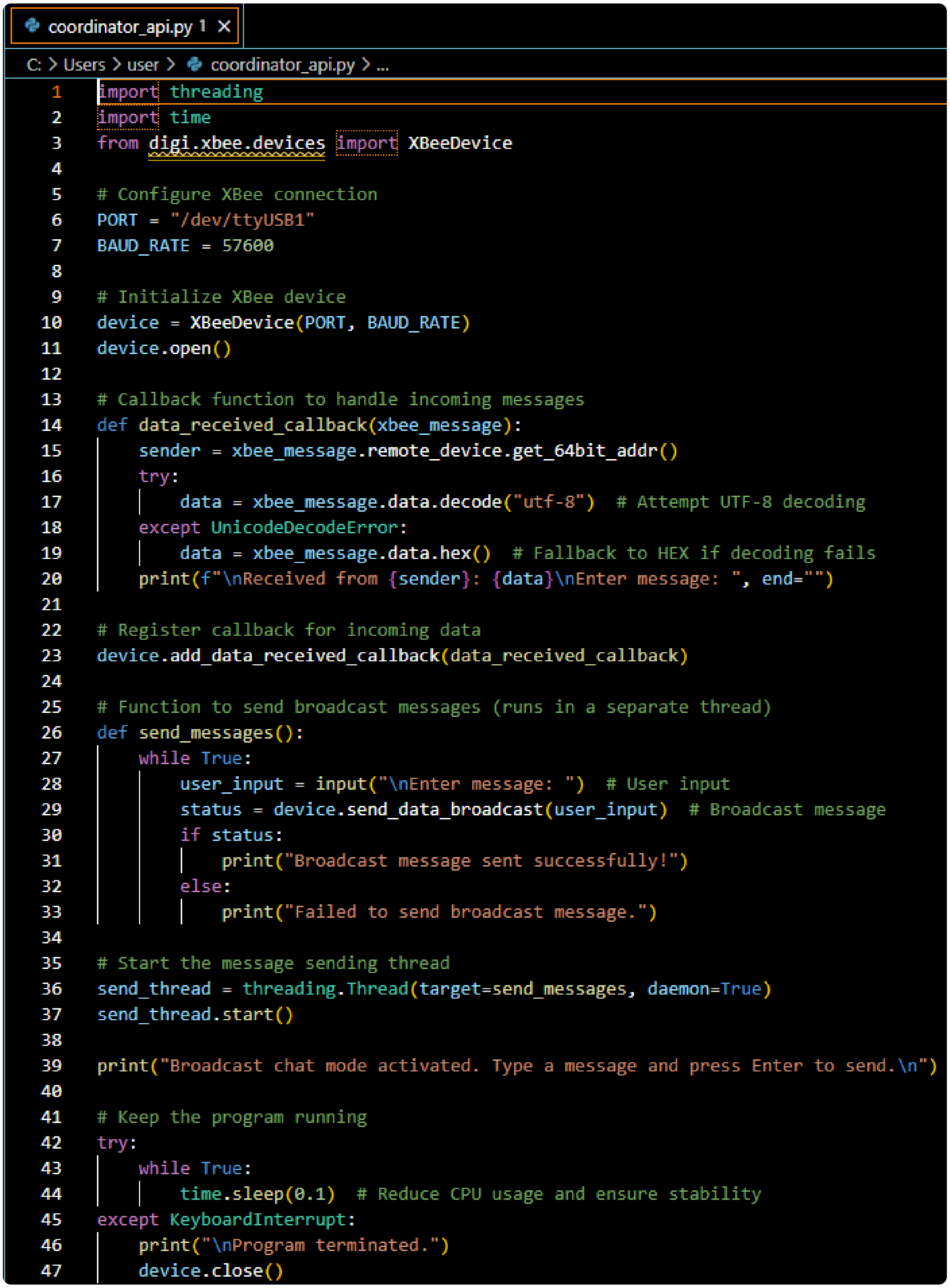
6. Some people recommend Ardupilot's parameters should be changed: TELEM_DELAY = 10、BRD_SER1_RTSCTS = 0

Others: Use zigbee3.0 or DigiMesh. It includes Router.

Coordinator	Router	Endpoint
JV: Disabled	JV: Enabled	JV: Enabled
CE: Enabled	CE: Disabled	CE: Disabled

Python codes work like XBee console in XCTU, here are the codes I test:

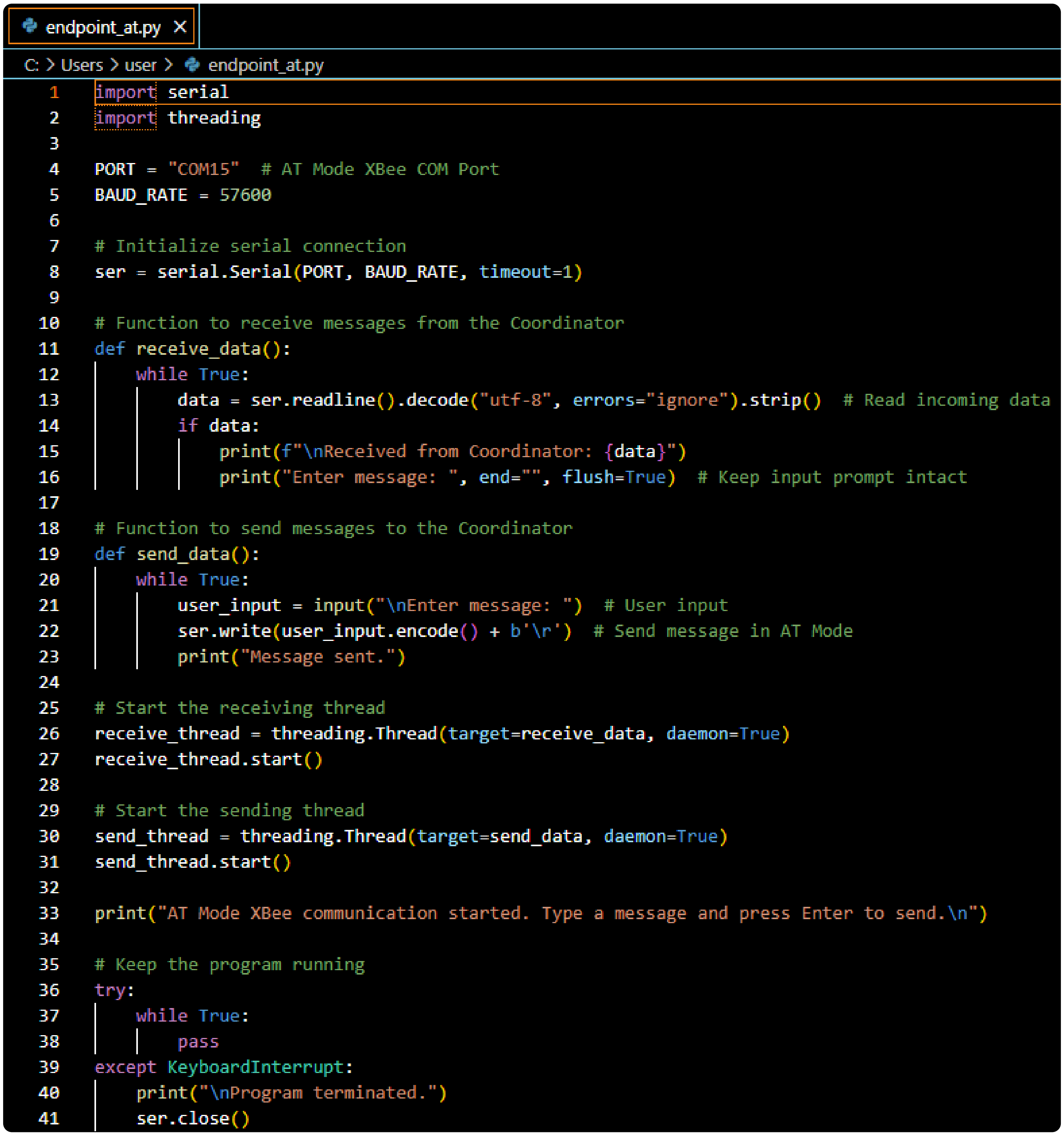
Coordinator


coordinator_api.py 1

C: > Users > user > coordinator_api.py > ...

```python
import threading
import time
from digi.xbee.devices import XBeeDevice

# Configure XBee connection
PORT = "/dev/ttyUSB1"
BAUD_RATE = 57600

# Initialize XBee device
device = XBeeDevice(PORT, BAUD_RATE)
device.open()

# Callback function to handle incoming messages
def data_received_callback(xbee_message):
    sender = xbee_message.remote_device.get_64bit_addr()
    try:
        data = xbee_message.data.decode("utf-8")  # Attempt UTF-8 decoding
    except UnicodeDecodeError:
        data = xbee_message.data.hex()  # Fallback to HEX if decoding fails
    print(f"\nReceived from {sender}: {data}\nEnter message: ", end="")

# Register callback for incoming data
device.add_data_received_callback(data_received_callback)

# Function to send broadcast messages (runs in a separate thread)
def send_messages():
    while True:
        user_input = input("\nEnter message: ")  # User input
        status = device.send_data_broadcast(user_input)  # Broadcast message
        if status:
            print("Broadcast message sent successfully!")
        else:
            print("Failed to send broadcast message.")

# Start the message sending thread
send_thread = threading.Thread(target=send_messages, daemon=True)
send_thread.start()

print("Broadcast chat mode activated. Type a message and press Enter to send.\n")

# Keep the program running
try:
    while True:
        time.sleep(0.1)  # Reduce CPU usage and ensure stability
except KeyboardInterrupt:
    print("\nProgram terminated.")
    device.close()
```

Endpoint



endpoint_at.py

C: > Users > user > endpoint_at.py

```python
import serial
import threading

PORT = "COM15"  # AT Mode XBee COM Port
BAUD_RATE = 57600

# Initialize serial connection
ser = serial.Serial(PORT, BAUD_RATE, timeout=1)

# Function to receive messages from the Coordinator
def receive_data():
    while True:
        data = ser.readline().decode("utf-8", errors="ignore").strip()  # Read incoming data
        if data:
            print(f"\nReceived from Coordinator: {data}")
            print("Enter message: ", end="", flush=True)  # Keep input prompt intact

# Function to send messages to the Coordinator
def send_data():
    while True:
        user_input = input("\nEnter message: ")  # User input
        ser.write(user_input.encode() + b'\r')  # Send message in AT Mode
        print("Message sent.")

# Start the receiving thread
receive_thread = threading.Thread(target=receive_data, daemon=True)
receive_thread.start()

# Start the sending thread
send_thread = threading.Thread(target=send_data, daemon=True)
send_thread.start()

print("AT Mode XBee communication started. Type a message and press Enter to send.\n")

# Keep the program running
try:
    while True:
        pass
except KeyboardInterrupt:
    print("\nProgram terminated.")
    ser.close()
```