File - C:\Users\Terry\PycharmProjects\adc_polling\main.py

```
import binascii
import time

import xbee
from machine import ADC, Pin

x = xbee.XBee()
#   set ADC reference to Vcc (+3.3v)
x.atcmd('AV', 2)

#   debug controls terminal output
debug = False

ADC_PIN_0 = "D0"
ADC_PIN_1 = "D1"
ADC_PIN_2 = "D3"
ADC_PIN_3 = "D2"
DIO_PIN_4 = "D4"
DIO_PIN_5 = "D5"
DIO_PIN_6 = "D6"
DIO_PIN_7 = "D7"
DIO_PIN_9 = "D9"
DIO_PIN_10 = "D10"
DIO_PIN_11 = "D11"
DIO_PIN_12 = "D12"
DIO_PIN_15 = "D15"
TARGET_64BIT_ADDR = b'\x00\x13\xA2\x00\x41\xB0\xB0\xD0'
xbupy = 0xB0D0
xbard = 0xF1B9
pri_pl = 77
sec_pl = 2
rcvck = 155

print(" +---------------------------------------+")
print(" | XBee MicroPython Joystick Transmitter |")
print(" +---------------------------------------+\n")

# Create an ADC object for pin DIO0/AD0.
adc_pin0 = ADC(ADC_PIN_0)
adc_pin1 = ADC(ADC_PIN_1)
adc_pin2 = ADC(ADC_PIN_2)
adc_pin3 = ADC(ADC_PIN_3)
D4 = Pin(DIO_PIN_4, Pin.IN, Pin.PULL_UP)
D5 = Pin(DIO_PIN_5, Pin.IN, Pin.PULL_UP)
D7 = Pin(DIO_PIN_7, Pin.IN, Pin.PULL_UP)
D9 = Pin(DIO_PIN_9, Pin.IN, Pin.PULL_UP)
D10 = Pin(DIO_PIN_10, Pin.OUT, Pin.PULL_UP)
D11 = Pin(DIO_PIN_11, Pin.OUT, Pin.PULL_UP)
D12 = Pin(DIO_PIN_12, Pin.OUT, Pin.PULL_UP)
D15 = Pin(DIO_PIN_15, Pin.OUT, Pin.PULL_UP)

adc0 = 128
adc1 = 128
adc2 = 128
adc3 = 128
adc0_Old = adc0
adc1_Old = adc1
adc2_Old = adc2
adc3_Old = adc3
errct = 0
ba = bytearray(8)
oldbap = bytearray(8)
oldbas = bytearray(8)
serial = False


def findrcv():
    receiver = 0
```

```
    try:
        xbee.transmit(xbupy, ba)
        receiver = xbupy
        print("Found XBee Python receiver")
    except Exception as er:
        print("Could not locate XBee Python receiver: %s" % str(er))

    try:
        xbee.transmit(xbard, ba)
        print("Found XBee Arduino receiver")
        if receiver == 0:
            receiver = xbard

    except Exception as er:
        print("Could not locate XBee Arduino receiver: %s" % str(er))

    if receiver == 0:
        print("FATAL ERROR - no receivers found")
    else:
        if receiver == xbupy:
            print("Using XBee Python receiver")
        else:
            print("Using XBee Arduino receiver")

    return receiver


def btndown(opt_str):
    if opt_str.find("0") != -1:
        return 0
    else:
        return 1


#   Scan the inputs for changes to transmit
#   two_channel controls transmission of data from the auxiliary port

two_channel = True
target = findrcv()

while True:
    adc0 = 255 - int(adc_pin0.read() / 16)
    diff = adc0 - adc0_Old
    if abs(diff) < 5:
        adc0 = adc0_Old
    else:
        adc0_Old = adc0
    adc1 = int(adc_pin1.read() / 16)
    diff = adc1 - adc1_Old
    if abs(diff) < 5:
        adc1 = adc1_Old
    else:
        adc1_Old = adc1
    #
    if two_channel:
        adc2 = 255 - int(adc_pin2.read() / 16)
        diff = adc2 - adc2_Old
        if abs(diff) < 5:
            adc2 = adc2_Old
        else:
            adc2_Old = adc2
        adc3 = int(adc_pin3.read() / 16)
        diff = adc3 - adc3_Old
        if abs(diff) < 5:
            adc3 = adc3_Old
        else:
            adc3_Old = adc3
```

```
    channel = pri_pl
    ba[0:1] = channel.to_bytes(1, 'big')
    ba[1:2] = adc1.to_bytes(1, 'big')
    ba[2:3] = adc0.to_bytes(1, 'big')
    ba[3:4] = (D4.value()).to_bytes(1, 'big')
    ba[4:5] = (D5.value()).to_bytes(1, 'big')
    ba[5:6] = rcvck.to_bytes(1, 'big')

    if ba != oldbap:
        try:
            if debug:
                print("T", binascii.hexlify(ba, "/"))
            xbee.transmit(target, ba)
        except Exception as e:
            errct = errct + 1
            print("Transmit failure: %s" % str(e))

        oldbap[0:8] = ba[0:8]

    channel = sec_pl
    ba[0:1] = channel.to_bytes(1, 'big')
    ba[1:2] = adc3.to_bytes(1, 'big')
    ba[2:3] = adc2.to_bytes(1, 'big')
    ba[3:4] = (D7.value()).to_bytes(1, 'big')
    ba[4:5] = (D9.value()).to_bytes(1, 'big')
    ba[5:6] = rcvck.to_bytes(1, 'big')
    if ba != oldbas:
        try:
            if debug:
                print("T", binascii.hexlify(ba, "/"))
            xbee.transmit(target, ba)
        except Exception as e:
            errct = errct + 1
            print("Transmit failure: %s" % str(e))
    oldbas[0:8] = ba[0:8]

    received_msg = xbee.receive()
    if received_msg:
        # Get the sender's 64-bit address and payload from the received message.
        payload = received_msg['payload']
        msg = payload.decode()
        if debug:
            print("R", binascii.hexlify(payload, "/"))
        pl_type = payload[0]
        if pl_type == pri_pl:
            D10.value(payload[6])
            D11.value(payload[7])
        else:
            D15.value(payload[6])
            D12.value(payload[7])

    time.sleep_ms(25)
```