第 4 代温湿度记录仪二次开发 UDP 通讯 python demo

版本 V1.0"""

```
def send_out(udp_socket, recv_ip, sn, functionCode):
    """回复数据"""
    dLen = bytes([0,1]) #数据长度
    state = bytes([0]) #  状态
    oOptions = bytes([0,0])  #  操作选项
    currentTime = bytes(list(map(int, time.strftime("%y,%m,%d,%H,%M,%S", time.localtime()).split(","))))  #  实时时间
    HDReporT = bytes([0, 0])  #  历史数据上报时刻

    RTReporInterval = 1  #  实时数据上报间隔(分钟,1-1440)
    RecorInterval = 1  #  数据记录间隔(分钟,1-1440)
    HDRepor = 0  #  历史数据上报间隔(小时,0 代表随实时数据上报)
    HTAlarm = 35  #  高温告警值(℃)
    LTAlarm = 0  #  低温告警值(℃)
    TBuffer = 0.1  #  温度缓冲值(℃)
    HHAlarm = 75  #  高湿告警值(rh%)
    LHAlarm = 35  #  低湿告警值(rh%)
    HBuffer = 0.1  #  湿度缓冲值(rh%)

    #  根据 SN 号判断设备类型回复配置数据,SN 号 C0 开头为,80 开头为单温度
    if sn[0:1] == b'\xC0':
        cdata = (oOptions + currentTime + (RTReporInterval).to_bytes(2, byteorder = 'big') + (RecorInterval).to_bytes(2, byteorder = 'big') + HDReporT +
                 (HDRepor).to_bytes(1, byteorder = 'big') + struct.pack('>h',int(HTAlarm*10)) + struct.pack('>h',int(LTAlarm*10)) + (int(TBuffer*10)).to_bytes(2, byteorder = 'big') +
                 (int(HHAlarm*10)).to_bytes(2, byteorder = 'big') + (int(LHAlarm*10)).to_bytes(2, byteorder = 'big') + (int(HBuffer*10)).to_bytes(2, byteorder = 'big'))
    elif sn[0:1] == b'\x80':
        cdata = (oOptions + currentTime + (RTReporInterval).to_bytes(2, byteorder = 'big') + (RecorInterval).to_bytes(2,byteorder = 'big') + HDReporT +
                 (HDRepor).to_bytes(1, byteorder='big') + struct.pack('>h',int(HTAlarm*10)) + struct.pack('>h',int(LTAlarm*10)) + (int(TBuffer * 10)).to_bytes(2,byteorder='big'))
    #  判断功能码,如果是实时数据带配置参数回复
    if functionCode == b'\x01' or functionCode == b'\x0B':
        cLen = bytes([0, len(cdata)])  #  配置长度
        CRCM = bytearray.fromhex((calc_crc((b'\x7e' + sn + functionCode + dLen + state + cLen + cdata).hex()))[2:6]) #计算 CRC 校验码
        sendBytes = b'\x7e'+ sn + functionCode + dLen + state + cLen + cdata + CRCM
+ b'\x0D'
    else:
        cLen = bytes([0, 0]) #  配置长度
        CRCM = bytearray.fromhex((calc_crc((b'\x7e' + sn + functionCode + dLen +
state + cLen).hex()))[2:6]) #计算 CRC 校验码
        sendBytes = b'\x7e' + sn + functionCode + dLen + state + cLen + CRCM +
b'\x0D'
    dest_ip = recv_ip[0]
    dest_port = recv_ip[1]
    udp_socket.sendto(sendBytes, (dest_ip, dest_port))
    print("回复数据:",sendBytes.hex())

def recv_data(udp_socket):
    """接收数据"""
    while True:
        recv_data = udp_socket.recvfrom(1024)
        recv_ip = recv_data[1] #设备 IP  端口
        data = recv_data[0] #设备上报的数据
        if data[0:1] == b'\x7e' and data[len(data)-1:len(data)] == b'\x0D': #  判断是否
包头为 7E,包尾为 0D
            print("接收数据:", data.hex())
            sn = data[1:7] #设备 SN 号
            functionCode = data[7:8] #功能码
            #  功能码 01 和 0B 为实时数据
            if functionCode == b'\x01' or functionCode == b'\x0B':
                temperature = struct.unpack('>h',data[10:12])[0] / 10 #温度值
                #  根据 SN 号判断设备类型,SN 号 C0 开头为,80 开头为单温度
                if sn[0:1] == b'\xC0':
                    humidity = int.from_bytes(data[12:14], byteorder='big') / 10   #  湿
度值
                    print("接收解析:","SN 号:"+sn.hex(), "温度:"+str(temperature), "湿
度:"+str(humidity))
                elif sn[0:1] == b'\x80':
                    print("接收解析:","SN 号:"+sn.hex(), "温度:"+str(temperature))
            #  功能码 02 和 0C 为历史数据
            elif functionCode == b'\x02' or functionCode == b'\x0C':
                dLen = int.from_bytes(data[8:10], byteorder='big') #  数据项长度
                #  温湿度记录解析
                if sn[0:1] == b'\xC0':
                    for i in range(10,dLen+1,10):
                        HTemperature = struct.unpack('>h', data[i:i+2])[0] / 10   #  记录
温度值
                        HHumidity = struct.unpack('>h', data[i+2:i+4])[0] / 10  #  记录
```

```
湿度值
                        #HTime = data[i+4:i+10].hex()   #  记录时间
                        #  记录时间
                                       HTime = ("20"+str(ord(data[i+4:i+5])).zfill(2) +
str(ord(data[i+5:i+6])).zfill(2) + str(ord(data[i+6:i+7])).zfill(2) + " " +
                                                     str(ord(data[i+7:i+8])).zfill(2)+ ":" +
str(ord(data[i+8:i+9])).zfill(2) + ":" + str(ord(data[i+9:i+10])).zfill(2))
                                                                  print(" 历 史 记 录
"+str(int(i/10)),HTemperature,HHumidity,HTime)
                #  单温度记录解析
                elif sn[0:1] == b'\x80':
                    for i in range(10,dLen+1,8):
                        HTemperature = struct.unpack('>h', data[i:i+2])[0] / 10   #  记录
温度值
                        #HHumidity = struct.unpack('>h', data[i+2:i+4])[0] / 10   #  记录
湿度值
                        #  记录时间
                                       HTime = ("20"+str(ord(data[i+2:i+3])).zfill(2) +
str(ord(data[i+3:i+4])).zfill(2) + str(ord(data[i+4:i+5])).zfill(2) + " " +
                                                     str(ord(data[i+5:i+6])).zfill(2)+ ":" +
str(ord(data[i+6:i+7])).zfill(2) + ":" + str(ord(data[i+7:i+8])).zfill(2))
                        print("历史记录"+str(int(i/10)),HTemperature,HTime)
            #  功能码 03 和 0D 为告警数据
            elif functionCode == b'\x03' or functionCode == b'\x0D':
                print("告警记录")
            send_out(udp_socket, recv_ip, sn, functionCode) #回复设备

def calc_crc(string):
    """CRCM 计算"""
    data = bytearray.fromhex(string)
    crc = 0xFFFF
    for pos in data:
        crc ^= pos
        for i in range(8):
            if ((crc & 1) != 0):
                crc >>= 1
                crc ^= 0xA001
            else:
                crc >>= 1
    return hex(((crc & 0xff) << 8) + (crc >> 8) + 4660)

def main():
    # 1.  创建套接字
    udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # 2.  绑定本地信息
    udp_socket.bind(("", 6666))
    # 3.  创建一个子线程用来接收数据
    t = threading.Thread(target=recv_data, args=(udp_socket,))
    t.start()

if __name__ == "__main__":
```