```python
# coding: utf-8
from bluepy import btle
import time
import logging
import globals
import struct
from multiconnect import Connector
from notification import Notification

class tp357():
    def __init__(self):
        self.name = 'tp357'
        self.ignoreRepeat = False

    def isvalid(self,name,manuf='',data='',mac=''):
        if name.lower() in [self.name]:
            return True
        if (mac.lower().startswith("10:76:36")):
            #broadcasted advertising data
            return True

    def parse(self,data,mac,name,manuf):
        action={}
        action['present'] = 1
        logging.debug('TP357 PARSE manufacturer: ' + manuf )
        # temperature
        logging.debug ('T_manuf: ' + manuf[2:6] )
        ts_manuf = '0x' + manuf[4:6] + manuf[2:4]
        t_manuf = int(ts_manuf, 16)/10.0
        logging.debug('TP357------ Advertising Data=> Temp: ' + str(t_manuf))
        action['temperature'] = t_manuf
        # moisture
        logging.debug ('M_manuf: ' + manuf[6:8] )
        ms_manuf = '0x' + manuf[6:8]
        m_manuf = int(ms_manuf, 16)
        logging.debug('TP357------ Advertising Data=> Moist: ' + str(m_manuf))
        action['moisture'] = m_manuf
        return action

    def read(self,mac):
        result={}
        try:
            conn = Connector(mac)
            conn.connect()
            if not conn.isconnected:
                conn.connect()
                if not conn.isconnected:
                    return
            Firm = bytearray(conn.readCharacteristic('0x24'))
            batt = bytearray(conn.readCharacteristic('0x18'))
            battery = batt[0]
            firmware = "".join(map(chr, Firm))
            notification = Notification(conn,tp357)
            conn.writeCharacteristic('0x10','0100',response=True)
            notification.subscribe(2)
            result['battery'] = battery
            result['firmware'] = firmware
            result['id'] = mac
            logging.debug('TP357------'+str(result))
            return result
        except Exception as e:
            logging.error(str(e))
        return result

    def handlenotification(self,conn,handle,data,action={}):
        result={}
        if hex(handle) == '0xe':
            received = bytearray(data)
            temp,hum = "".join(map(chr, received)).replace("T=", "").replace("H=", ""
            ).rstrip(' \t\r\n\0').split(" ")
            result['moisture'] = hum
            result['temperature'] = temp
```

```
72                result['id'] = conn.mac
73                result['source'] = globals.daemonname
74                globals.JEEDOM_COM.add_changes('devices::'+conn.mac,result)
75
76    globals.COMPATIBILITY.append(tp357)
77
```