1204

【ラズパイ】
1.2345(センサ値) → ［〇｜〇｜〇｜〇］の配列のように 1 バイトずつに分けて NXP 送信
（下記のラズパイコードを参照）

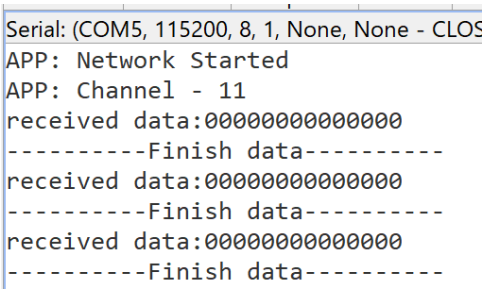
【NXP―Enddevice】
ラズパイから UART 通信で Enddevice は受信する．その際，1 バイトずつ読み取っていく．
最初を表す 0xAA，コマンドを表す 0xBB は，確認するだけで捨てる．

その後のセンサ値は tmpString 配列に格納する．この tmpString 配列は uint8_t 型にする．
前回この配列が char 型にしていた．

センサ値である 4 バイトを 4 回のループで tmpString 配列に格納したら，
PUBLIC uint8_t tmpString[64]　にすることで，他のファイルから呼び出せるようにし，
遠隔の MONOSTICK に送信するようにした．
(下記の Ustils コード, Enddevice コードを参照)

【NXP―Coordinator】
受信する側である Coordinator では
DBG_vPrintf(TRACE_APP, "%c", u8TempPayload);であった．
tmpString は uint8_t で，中身のセンサ値は 16 進数のため, %u や %x に変更して
(下記の Coordinator コードを参照)
実行したが，空白ではないものの, 0 の羅列が続いて表示している．
またセンサデータを格納する tmpString[64]　→　[16]にしたが，変化なし．

以下は Coordinator の表示内容である．



【ラズパイコード】

```
33
34  startTime = time.time()
35
36  SerialObj.baudrate = 115200  # set Baud rate to 115200
37  SerialObj.bytesize = 8     # Number of data bits = 8
38  SerialObj.parity   ='N'    # No parity
39  SerialObj.stopbits = 1     # Number of Stop bits = 1
40  SerialObj.timeout  = None  # Setting timeouts here No timeouts,waits forever
41
42  def uint8ToBytes (dataArray,decimalValue):
43      bytes = decimalValue.to_bytes(1,byteorder="little")
44      dataArray.extend(bytes)
45  def uint16ToBytes (dataArray,decimalValue):
46      bytes = decimalValue.to_bytes(2,byteorder="little")
47      dataArray.extend(bytes)
48  def uint32ToBytes (dataArray,decimalValue):
49      bytes = decimalValue.to_bytes(4,byteorder="little")
50      dataArray.extend(bytes)
51  def uint64ToBytes (dataArray,decimalValue):
52      bytes = decimalValue.to_bytes(8,byteorder="little")
53      dataArray.extend(bytes)
54  def MlmeScanRequest(channelList,scanType,scanDuration):
55      dataBytes = bytearray()
56      uint8ToBytes(dataBytes, 170) # Begin of data 0xAA
57      uint8ToBytes(dataBytes,6) # primitive type
58      uint32ToBytes(dataBytes,channelList)
59      uint8ToBytes(dataBytes,scanType)
60      uint8ToBytes(dataBytes,scanDuration)
61      uint32ToBytes(dataBytes,0) # Security data
62      SerialObj.write(dataBytes)
63      #print(dataBytes)
64
65  def SendSensor(data_sensor):
66
67      dataBytes = bytearray()
68      uint8ToBytes(dataBytes, 170) # Begin of data 0xAA
69      uint16ToBytes(dataBytes,187) # primitive type 0xBB
70      #uint8ToBytes(dataBytes,3)
71      dataBytes_sensor = bytearray(struct.pack("f", data_sensor))
72      dataBytes.extend(dataBytes_sensor)
73      SerialObj.write(dataBytes)
74
```

```
75
76  while True:
77      data = 0
78      sum_data = 0
79      volt = 0
80      sum_volt = 0
81      count =0
82      avg_data = 0
83      avg_volt = 0
84
85
86      #ADS1015からデータを取得
87      data = ads.read_adc(ads1015_pin, gain=GAIN)
88      sum_data += data
89      volt = data * UNIT
90      sum_volt += volt
91      count = count + 1
92      nextTime = time.time() - startTime
93
94
95      if nextTime > 5:
96          avg_data = sum_data / count
97          avg_volt = sum_volt / count
98          print("受信データ：" + str(avg_data))
99          print("電圧      :" + "{:.3f}".format(avg_volt))
100
101         SendSensor(avg_volt)
102
103         startTime = time.time()
104         nextTime = 0
105         count = 0
106
107         ReceivedString = SerialObj.readline()
108         print(ReceivedString)
109
110 SerialObj.close()
111
```

【ラズパイと Enddevice の表示内容】

```
電圧       : 0.668
b'data: 0 \r\n'
受信データ : 402.0
電圧       : 0.804
b'Sensor Data Catch \r\n'
受信データ : 395.0
電圧       : 0.790
b'data: 27 \r\n'
受信データ : 348.0
電圧       : 0.696
b'Sensor Data Catch \r\n'
受信データ : 386.0
電圧       : 0.772
b'data: 31 \r\n'
受信データ : 351.0
電圧       : 0.702
b'Sensor Data Catch \r\n'
受信データ : 402.0
電圧       : 0.804
b'data: 48 \r\n'
^Z
[1]+  停止                 python al.py
```

【Ustils コード】

```
uint8_t rxByte =  u8AHI_UartReadData (DBG_E_UART_0);

switch(state){
 case START: //最初を表す文字
     if(rxByte = 0xAA)
     {
       state = PRIMITIVE_TYPE;
       DBG_vPrintf(TRUE,"In start state\n");
       break;
     }
     break;
 case PRIMITIVE_TYPE: //コマンド処理
     DBG_vPrintf(TRUE,"In primitive type1 state \n");
     if(rxByte == 0xBB){
      DBG_vPrintf(TRUE,"In primitive type2 state \n");
      cmd = 3;
      state = WAIT_DATA;
      break;
     }
     break;
 case WAIT_DATA: //センサデータ(4回ループ 1biteごとに呼ばれる 4bite)
     DBG_vPrintf(TRUE,"Sensor Data Catch \n");
     DBG_vPrintf(TRUE,"data: %x \n", rxByte);
     tmpString[indexCount] = rxByte;
     indexCount++;
     if(indexCount == 4){
         DBG_vPrintf(TRUE,"Sensor Data Send \n");
         indexCount = 0;
         memset(tmpString,0,sizeof(tmpString));
         state = START;
     }
     break;
 default:
    break;
```

【Enddevice コード】

```
    uint8 u8TransactionSequenceNumber;
    ZPS_tsNwkNib * thisNib;
    thisNib = ZPS_psNwkNibGetHandle(ZPS_pvAplZdoGetNwkHandle());

    PDUM_thAPduInstance hAPduInst;
    hAPduInst = PDUM_hAPduAllocateAPduInstance(apduZDP);

    uint16 u16Offset = 0;
    uint16 i;
    u16Offset = 0;
```

```
    for (i = 0; i < 5; i++) {
            u16Offset += PDUM_u16APduInstanceWriteNBO(hAPduInst, u16Offset,"b", *(tmpString + i));
        }

    PDUM_eAPduInstanceSetPayloadSize(hAPduInst, u16Offset);
    DBG_vPrintf(TRUE, "Size : %d\nSending : ", PDUM_u16APduInstanceGetPayloadSize(hAPduInst));
    for (i = 0; i < 14; i++) {
        DBG_vPrintf(TRUE, "%c", *(tmpString + i));
    }
    DBG_vPrintf(TRUE, "\n");


    if (hAPduInst == PDUM_INVALID_HANDLE)
    {
        DBG_vPrintf(TRUE, "PDUM_INVALID_HANDLE\n");
    } else {

        ZPS_teStatus eStatus;
        ZPS_teAplAfSecurityMode  eSecurityMode = (ZPS_E_APL_AF_UNSECURE);//セキュリティ無効

        uint64 unicastMacAddr  = 0x001BC50122016BD5;
        eStatus=ZPS_eAplAfUnicastIeeeDataReq( //ユニキャスト通信
                hAPduInst,
                0x1337,
                0x01,
                0x01,
                unicastMacAddr,  //Dest: 64bit-coordinator
                eSecurityMode,
                0,
                &u8TransactionSequenceNumber
        );
```

【Coordinator コード】

```
   if (ZPS_EVENT_NONE != sStackEvent.eType)
   {
       switch (sStackEvent.eType)
       {
           case ZPS_EVENT_APS_DATA_INDICATION: //何かしら他端末からのデータを受信した
           {
              // DBG_vPrintf(TRACE_APP, "APP: APP_taskEndPoint: ZPS_EVENT_AF_DATA_INDICATION\n");

               /* Process incoming cluster messages for this endpoint... */
              // DBG_vPrintf(TRACE_APP, "    Data Indication:\r\n");
              // DBG_vPrintf(TRACE_APP, "        Profile :%x\r\n",sStackEvent.uEvent.sApsDataIndEvent.u16ProfileId);
              // DBG_vPrintf(TRACE_APP, "        Cluster :%x\r\n",sStackEvent.uEvent.sApsDataIndEvent.u16ClusterId);
              // DBG_vPrintf(TRACE_APP, "        EndPoint:%x\r\n",sStackEvent.uEvent.sApsDataIndEvent.u8DstEndpoint);

 /*追加部分ここから*/
               uint8 u8TempPayload;
               uint16 u16bytesread;
               uint16 i;
               DBG_vPrintf(TRACE_APP, "received data:");
               for(i = 0; i < 14; i++){
                u16bytesread = PDUM_u16APduInstanceReadNBO(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst,i,"b",&u8TempPayload);
                DBG_vPrintf(TRACE_APP, "%c", u8TempPayload);
               }
                DBG_vPrintf(TRACE_APP, "\n");


                DBG_vPrintf(TRACE_APP, "----------Finish data----------\n");
 /*ここまで*/

               /* free the application protocol data unit (APDU) once it has been dealt with */
               PDUM_eAPduFreeAPduInstance(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst);
           }
           break;
```