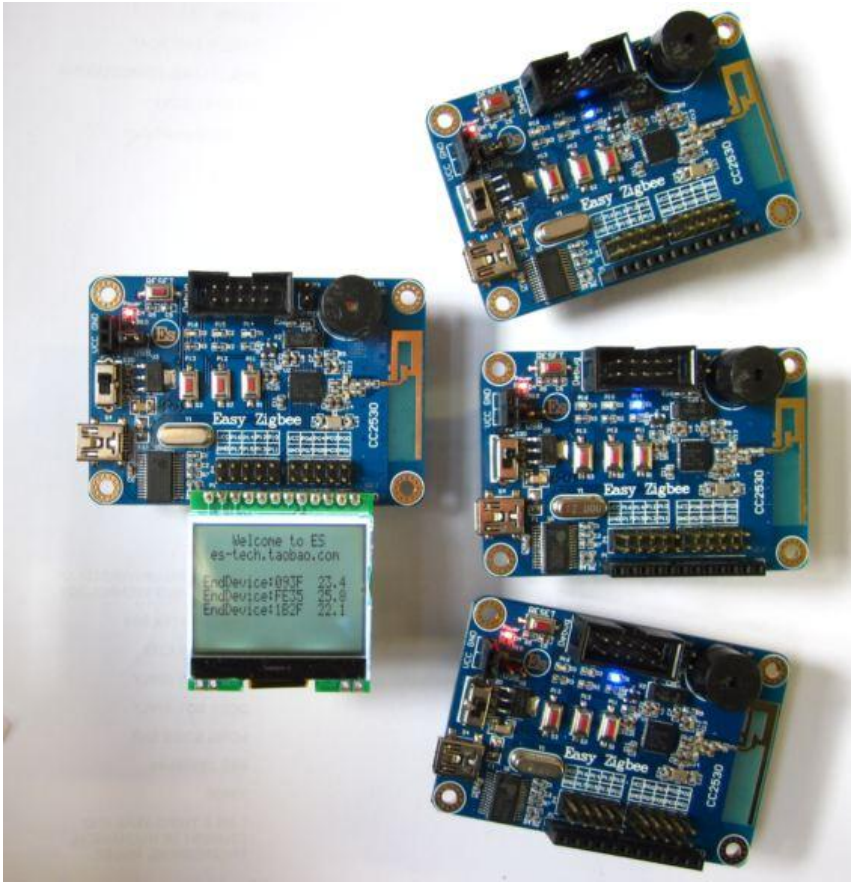
文件名：无线传感器网络（自组网与数据传输）



注意：

1. 先安装好 PL2302 USB 转串口驱动，转备好 USB 线，连接计算机与协调器模块模块。
2. 需要两个或两个以上模块。

- **实验目的：**

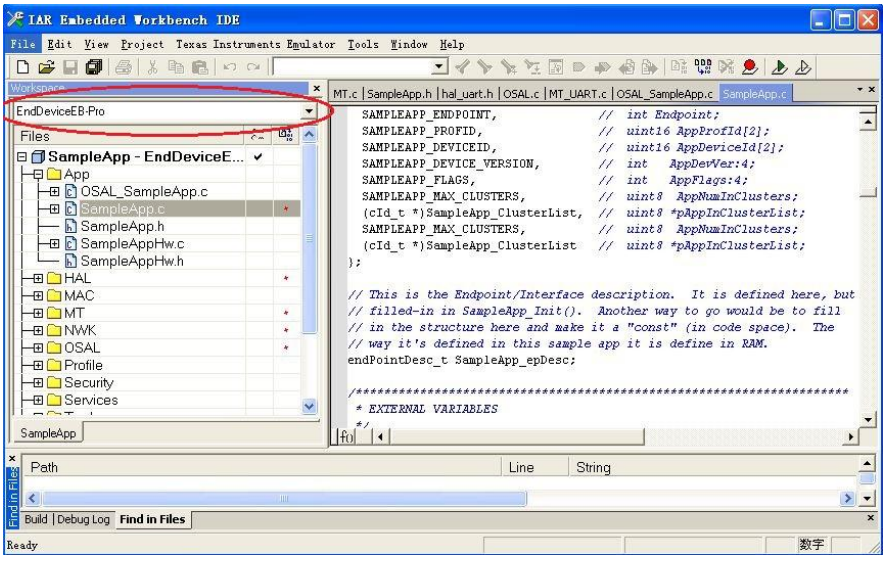
  学习如何使用 TI 提供的协议栈，组成无线传感器网络，并实现自组网与无线数据传输。其中一个模块为协调器，其他模块为无线终端。一个网络只有一个协调器，可以有多个无线终端。

- **实验现象：**

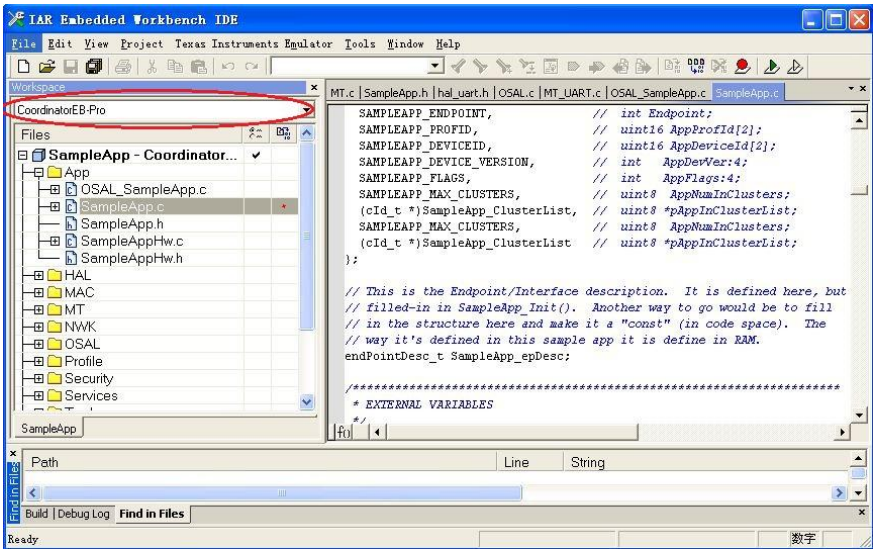
1. 无线终端上电以后，自动搜索附近是否有协调器，如果没有，过一段时间后重新搜索；如果有则向协调器申请加入网络，协调器自动向无线终端分配一个 16 位的地址，组网成功后，无线终端的蓝色 LED 常亮；
2. 其他无线终端上电以后，以同样的方式申请加入这个网络，协调器自动分配没一个无线终端一个不同的地址；
3. 无线终端每隔 3 秒向协调器发送一组数据；
4. 协调器接收所有无线终端发送过来的数据，并把每个无线终端的地址和发送过来的数据通过串口发送到电脑，显示出来。

- **实验步骤：**

1. 打开工程文件：Projects\zstack\Samples\无线传感器网络（自组网）\CC2530DB，选择 **EndDeviceEB-Pro,** 下载到每一个无线终端模块中；（作为终端设备无线发送数据给协调器），如下图所示。



2. 选择 **CoodinatorEB-Pro,** 下载到协调器模块中；（作为协调器串口跟电脑连接），并连接 USB 线。如下图所示。



3. 使用 USB 数据线连接协调器与计算机，打开串口调试助手，设置好参数，打开协调器电源。
4. 分别打开所有无线终端的电源。可以看到串口调试助手每隔 3 秒左右便收到一次数据，如下图所示：

具体实验：（发送部分）

1. 登记事件，设置编号、发送时间等

```
uint16 SampleApp_ProcessEvent( uint8 task_id, uint16 events )
{
  afIncomingMSGPacket_t *MSGpkt;
  (void)task_id;  // Intentionally unreferenced parameter

  if ( events & SYS_EVENT_MSG )
  {
    MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( SampleApp_TaskID );
    while ( MSGpkt )
    {
      switch ( MSGpkt->hdr.event )
      {

        case CMD_SERIAL_MSG:
        SampleApp_SerialCMD((mtOSALSerialData_t *)MSGpkt);
        break;

        // Received when a key is pressed
        case KEY_CHANGE:
          SampleApp_HandleKeys( ((keyChange_t *)MSGpkt)->state,
((keyChange_t *)MSGpkt)->keys );
          break;

        // Received when a messages is received (OTA) for this endpoint
        case AF_INCOMING_MSG_CMD:
          SampleApp_MessageMSGCB( MSGpkt );
          break;

        // Received whenever the device changes state in the network
        case ZDO_STATE_CHANGE:
          SampleApp_NwkState = (devStates_t)(MSGpkt->hdr.status);
          if ( (SampleApp_NwkState == DEV_ZB_COORD)
              || (SampleApp_NwkState == DEV_ROUTER)
              || (SampleApp_NwkState == DEV_END_DEVICE) )
          {
```

```
            // Start sending the periodic message in a regular interval.
            osal_start_timerEx( SampleApp_TaskID,
                                SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
                                SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT );
          }
          else
          {
            // Device is no longer in the network
          }
          break;

        default:
          break;
      }

      // Release the memory
      osal_msg_deallocate( (uint8 *)MSGpkt );

      // Next - if one is available
      MSGpkt = (afIncomingMSGPacket_t
*)osal_msg_receive( SampleApp_TaskID );
    }

    // return unprocessed events
    return (events ^ SYS_EVENT_MSG);
  }

  // Send a message out - This event is generated by a timer
  //   (setup in SampleApp_Init()).
  if ( events & SAMPLEAPP_SEND_PERIODIC_MSG_EVT )
  {
    // Send the periodic message
    SampleApp_SendPeriodicMessage();

    // Setup to send message again in normal period (+ a little jitter)
    osal_start_timerEx( SampleApp_TaskID,
SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
```

```
            (SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT + (osal_rand() & 0x00FF)) );
      // return unprocessed events
      return (events ^ SAMPLEAPP_SEND_PERIODIC_MSG_EVT);
   }
   // Discard unknown events
   return 0;
}
```

解释：

1.1 SampleApp_TaskID 为任务 ID, 函数开头定义了 **SampleApp_TaskID = task_id;**也就是 **SampleApp** 初始化的任务 **ID** 号。

1.2 SAMPLEAPP_AA_PERIODIC_MSG_EVT 为登记任务事件，同一个任务下可以有多个事件，这个是事件的号码。我们可以定义自己的事件，但是编号不能重复。文件定义了**#define SAMPLEAPP_SEND_PERIODIC_MSG_EVT 0x0001**

1.3 SAMPLEAPP_AA_PERIODIC_MSG_TIMEOUT 为事件重复执行的时间。这里可以你需要发送数据的时间间隔。文件中定义了**#define SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT 3000**，这里以毫秒为单位，所以是 3s, 也就是刚刚实验为什么隔约 3s 收到数据的原因。

2. 周期性发送数据函数

```
void SampleApp_SendPeriodicMessage( void )
{
   uint8 data[10]={178,201,45,56,46,58,77,32,88,19};
   if ( AF_DataRequest( &SampleApp_Periodic_DstAddr, &SampleApp_epDesc,
                        SAMPLEAPP_PERIODIC_CLUSTERID,
                        10,                    //一共 10 个数据
                        data,                  //装载要发送的数据
                        &SampleApp_TransID,
                        AF_DISCV_ROUTE,
                        AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
   {
   }
}
```

解析:

其中发送的数据储存在 data 数组里。

具体实验：（接收部分）

1. 读取数据并把发送到电脑函数

```
void SampleApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
  uint8 asc_16[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
  uint16 flashTime,temp;
  uchar receive_data[10];
  switch ( pkt->clusterId )
  {
    case SAMPLEAPP_POINT_TO_POINT_CLUSTERID:
      temp=pkt->srcAddr.addr.shortAddr; //读出数据包的 16 位短地址
      for(i=0;i<10,i++) receive_data[i]=pkt->cmd.Data[i] //读出数据(每次 10 个）

     HalUARTWrite(0,"ENDDEVICE ShortAddr:0x",22); //串口显示
      /****将短地址分解，通过串口显示出来*****/
      HalUARTWrite(0,&asc_16[temp/4096],1);
      HalUARTWrite(0,&asc_16[temp%4096/256],1);
      HalUARTWrite(0,&asc_16[temp%256/16],1);
      HalUARTWrite(0,&asc_16[temp%16],1);

      HalUARTWrite(0,"   Reveive data: ",22); //串口显示
      /****将接收到的第一个分解，通过串口显示出来*****/
      HalUARTWrite(0,&asc_16[temp%1000/100],1);
      HalUARTWrite(0,&asc_16[temp%100/10],1);
      HalUARTWrite(0,&asc_16[temp%10],1);

      HalUARTWrite(0,"\n",1);                    // 回车换行

      break;

    case SAMPLEAPP_FLASH_CLUSTERID:
      flashTime = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
      HalLedBlink( HAL_LED_4, 4, 50, (flashTime / 4) );
      break;
  }
}
```

解释：

所有的数据和信息都在函数传入来的 **afIncomingMSGPacket_t *pkt** 里面，进入 **afIncomingMSGPacket_t** 的定义，它是一个结构体，内容如下：

```
typedef struct
{
  osal_event_hdr_t hdr;      /* OSAL Message header */
  uint16 groupId;            /* Message's group ID - 0 if not set */
  uint16 clusterId;          /* Message's cluster ID */
  afAddrType_t srcAddr;      /* Source Address, if endpoint is
STUBAPS_INTER_PAN_EP,
                                it's an InterPAN message */
  uint16 macDestAddr;        /* MAC header destination short address */
  uint8 endPoint;            /* destination endpoint */
  uint8 wasBroadcast;        /* TRUE if network destination was a broadcast
address */
  uint8 LinkQuality;         /* The link quality of the received data frame
*/
  uint8 correlation;         /* The raw correlation value of the received
data frame */
  int8  rssi;                /* The received RF power in units dBm */
  uint8 SecurityUse;         /* deprecated */
  uint32 timestamp;          /* receipt timestamp from MAC */
  afMSGCommandFormat_t cmd; /* Application Data */
} afIncomingMSGPacket_t;
```

里面包含了数据包的所有东西，长地址、短地址、RSSI 等。其中洪湖数据在 afMSGCommandFormat_t cmd 里面。

```
typedef struct
{
  byte   TransSeqNumber;
  uint16 DataLength;              // Number of bytes in TransData
  byte  *Data;
} afMSGCommandFormat_t;
```