编程挑战赛决赛赛题补充说明及常见问题解答

第一部分：赛题 A

一、代码格式问题

赛题 A 采用软件评分的方式来进行，参赛队仅可提交一份代码。该代码需满足以下要求：

1. 符合 TinyOS-2.1.1 标准，在该环境下能编译通过。

2. 该代码需包含 0-49 号节点的所有逻辑。

3. 该代码中需在发射功率为 1 的前提下进行书写，评分时，将强制设定功率为 1。

二、图片数据的发送速率问题

PC 端往 0 号节点发送图片数据时，由于受 0 号节点本身代码的影响，现做出如下说明：

在评分时，所有参赛队发送图片数据的程序一致，但接收的效果需参赛队自己优化。可从以下方面优化：

a）0 号节点不可存在往串口发送的代码块。

b）0 号节点代码逻辑必须严谨，不可过多占用节点资源，如尽量避免使用死循环、task 里面套 post 形成的死循环等。

c）其他节点需合理的进行数据包的传递，避免无节制的随意发送数据包，造成信道资源的占用和大量的发送开销。

三、最终结果的判定

评分软件会一直对 49 号节点的信息进行监听，最终计算结果时，将根据数据包对应位来对比像素点是否正确，数据包的格式（即在 49 号点监听到的信息）必须满足：

头信息（8 字节）+ 数据包负载（77 字节）

核对结果时会将第 9 和 10 个字节当作数据包序号，从而解析出最后的 75 个字节的 RGB 信息。

PC 发送给 0 号点的数据包是按照序号(即 0-399)的顺序发送的，在核对正确性时，要求参赛队按照顺序输出包的信息，即输出的包按照 0，1，2，……，398，399 的包序号输出。

a）如输出为 0，1，3，2，8，5，6，7，9,…则只会将其中 0，1，3，8，9 算作正确的输出包；

b）重复的包只算最开始的那个；

c）不满足长度的包直接忽略

四、评分软件的评分流程

评分软件流程如下：

1.参赛队抽签，确定评分顺序。

2.将参赛代码和评分顺序导入软件。

3.显示参赛队参赛信息（含学校 logo，学校名称，参赛人员）。

4.编译参赛代码（按参赛队编号识别该队代码）。

5.烧录 null 程序（清除上一队烧录的代码）。

6.烧录参赛队程序（50 个点批量烧录，烧录不成功将再次烧录）。

7. 后台监听开始（1-48 号点的开销，49 号点的监听信息）

8. 开始发送图片数据

9.90 秒时间到，停止后台监听，计算最终得分。

10.进行下一组评分

五、常见问题解答

下面是某些参赛队提出的问题，希望下面的回答对参赛队能有所帮助。

1. 测试平台与实际的比赛平台是否一致？

答：测试云平台是给参赛队测试使用，需要借助网络。而评分软件是直接电脑对实验床，专为赛题 A 评分设计，更加的稳定。另外测试平台上并未对功率、信道进行限定，参赛队使用时可随意的在 Makefile 里定义。其节点的位置也是按照顺序依次摆放的，与比赛时的节点位置随机有所不同，参赛队需注意。

2.图片数据的发送速率是多少？在 90 秒之前能发送完成吗？没发送完怎么处理？

答：由于图片数据的发送和 0 号节点的代码质量有关，所以速率无法确定，如果代码正常一般情况下在 70S 内是能够发送完毕的。没发送完说明代码还不够完善，评分结果也按照实际的结果来计算。

3.0 号节点不能存在串口发送的代码块，49 号节点又需要往串口发送，而又共用一份代码，这不是矛盾的吗？

答：虽然 0 号节点和 49 号节点共用一份代码，但每个节点需要具备的功能不一样，例如 1-48 号节点不需要串口收发模块一样（它们只承担无线转发的任务）。当我们知道每个节点的功能时，是可以通过很多种方式在代码里将不同身份的节点逻辑进行区分的。当节点烧录上电后，其对应的 TOS_NODE_ID 就是本节点的 ID 号。即每个节点烧录的代码一样，但其代码里 TOS_NODE_ID 值是不一样的，只和节点 ID 一致。

4.节点烧录完毕后，多少秒图片数据才开始发送？时间太短会不会影响路由的建立？

答：在评分流程里我们可以看到烧录完毕后,软件会执行后台监听，再发送图片数据。可以看出，这个时间非常短，基本上在 2 秒内就开始发送了。对于建立路由的影响，那就要看代码的设计了。

5.50 个点同时烧录，那么他们的时钟是一致的吗？

答：虽然是同时烧录，但完成还是有细微的先后，另外节点本身计时就不一定精确。

第二部分：赛题 B

一、标签信息收集节点代码问题（重要）

在之前公布的赛题 B 的中，规定标签信息收集节点不由参赛队编程，现有不少参赛队希望能对该节点进行编程，从而更好的使用网络协议来完整本赛题。现做出如下说明：

1.标签信息收集节点代码参赛队可自行编写代码，也可默认使用官方提供的代码进行比赛，但其数据包发送格式必须满足：

```
//数据包格式
typedef nx_struct TagMsg{
    nx_uint16_t tagid;      //标签id
    nx_uint16_t tagdata;    //标签数值
    nx_uint16_t hint;       //参赛队标志项，用于区分其他队伍
}TagMsg;
```

此节点在烧录代码后，需交由大赛方放于固定位置，然后进行监听，从而获取到最终结果，其数据包格式如下：

8 字节数据监听头信息+2 字节标签 ID+2 字节标签数据+2 字节队伍信息

例如：

00 FF FF 00 00 06 00 89 00 64 00 AB 00 01

00 FF FF 00 00 06 00 89：Listen 包头

00 64：标签 ID 为 0x64 （即 100）

00 AB：标签数据

00 01：参赛队编号（此项比赛时根据实际情况来定）

大赛方再根据获取到的数据包，拿到对应的位来核对结果。

2. 标签信息收集节点代码可仿照给出的程序进行编写。

3. 参赛队需自行确保数据的安全加密工作，确保数据的格式和准确性。

4. 参赛队也可默认使用之前提供的代码来完成本题。

二、 比赛细节

1.比赛场地：16m*15m 外围会用 10cm 高度的泡棉围住，防止小车跑出。

2.Tag 位置：Tag 会用三脚架固定在 1.2m 左右的高度，三脚架会放在 1m*1m 泡棉上，防止小车撞到。

3.Tag 信息：ID 编号从 100-114，共 15 个，离三脚架 1.5m 以内一般都能读取到数值，放置 Tag 的地方会有指示牌指明 Tag 的 ID 号。

4.节点信息：共 11 个=1 个小车车载节点+1 个 PC 端节点+8 个组网节点+1 个标签信息收集节点。

5.参赛队需考虑移动中的智能车网络传输质量，传输成功率，控制实时性等情况。

三、常见问题解答

1.怎么发送数据给 PC 端节点？

答：PC 与节点通信可参考赛题 A 中的 SendPkt 程序，该程序就是典型的 PC 往节点发送数据。当然参赛队也可通过 PC 端图形化界面来操控下车，控制小车的命令也可自行定义。请仔细分析 PC、PC 端节点、车载节点、辅助节点、信息收集节点各自扮演的角色。

2.比赛中能否用我们自己的电脑对节点烧写程序和控制小车？

答：参赛队可用自己的电脑控制小车，但节点的烧录需要在大赛方的设备上完成。

3.小车是用我们自己的吗？小车上节点的程序和布置用于组网的节点的程序，还有与 PC 相连的节点的程序是否需要分开写还是写在一个程序中？即比赛中是分开提交 3 个程序，还是只提交一个程序？

答：在比赛现场我们会提供小车，参赛队可以使用自己的小车，也可以由我们提供（组多车少）。另外我们没有规定程序必须是一份，只要能自己操控小车获取到 Tag 信息并在信息收集节点输出就行，至于你的代码是什么样，我们并不规定。

4.标签信息收集节点放在什么位置？

答：由于要对信息收集节点的输出信息进行监听，故该点将放在赛场某侧，具体要以实际的比赛赛场来决定。

5.标签对应的 ID 会事先告知吗？

答：放置 Tag 的地方会有指示牌指明 Tag 的 ID 号，直接读取即可。

6.参赛的 hint 位是什么？

答：hint 位是一个常量，作为一个参考，并不会影响比赛结果，比赛时会提前告知参赛队。

7.辅助节点必须放置 8 个吗？

答：辅助节点个数没有要求，只要能完成本题，摆放的位置、个数随意，但最多不超过 8 个。

另外，参赛队可将大赛赛题方面的疑问发送到大赛技术支持邮箱，问题的描述应当详细清晰，可适当加入图片、代码进行说明。

大赛技术支持邮箱：tech@iotcompetition.org