了e﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽



**课程实验报告**

**课程名称： 物联网通信技术**

**专业班级：**

**学 号： U2016XXXXX**

**姓 名： XXX**

**指导教师： 徐海银**

**报告日期： 2018.12**

**计算机科学与技术学院**

目录

实验一 3

1 实验目的与要求 3

2 实验内容 3

3 实验过程与结果 3

4实验结果分析 5

5心得体会与总结 6

实验二 7

1 实验目的与要求 7

2 实验内容 7

3 实验过程与结果 7

4实验结果分析 11

5心得体会与总结 13

实验三 15

1 实验目的与要求 15

2 实验内容 15

3 实验过程与结果 15

4实验结果分析 17

5心得体会与总结 18

# 实验一

## 1 实验目的与要求

本实验介绍了如何在 TinyOS 上进行节点与节点之间的无线通信。通过这个实验，熟悉通信相关的组件及接口以及如何以单播的方式发送和接收消息。

## 2 实验内容

实现类似跑马灯的效果，两个节点都维护计数值 counter，初始值为 1。节点 1 每隔１秒发送计数值到节点２，节点２每隔１秒发送自身计数值＋１到节点１。节点 1 和节点 2 均接收对方发送的计数值，收到后更新 counter，用 LED 灯显示 counter 的末三位。

效果：节点 1，节点 2 都开着的时候，节点１和节点 2 的 LED 灯显示的值每隔１秒递增１。此时按住其中一个节点的 RESET，另外一个节点的显示将不会停住不再变化，放开之后，又重新从１开始计数。提示：需要分辨节点的编号以发送不同的计数值。

## 3 实验过程与结果

首先我们先打开系统提供的BlinkToRadio，位于下图位置：

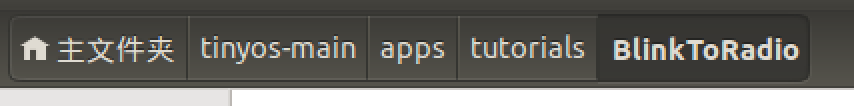


图1.1 BlinkToRadio位置

BlinkToRadio可以实现以无线方式收发包。一个节点，每隔一秒会将自身的counter值加1，然后通过无线传输的方式发给别的节点。其他节点收到该counter值，会通过节点的led灯显示该值。

下面我们试着修改信道，通过修改信道，可以防止干扰。如下图，在BlinkToRadio的makefile中添加一行：

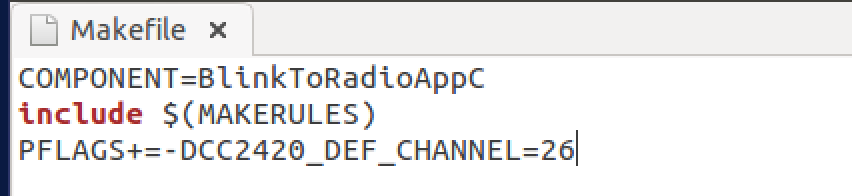


图1.2 添加信道

然后我们去修改时延，以便观察实验现象。修改BlinkToRadio.h，如下：

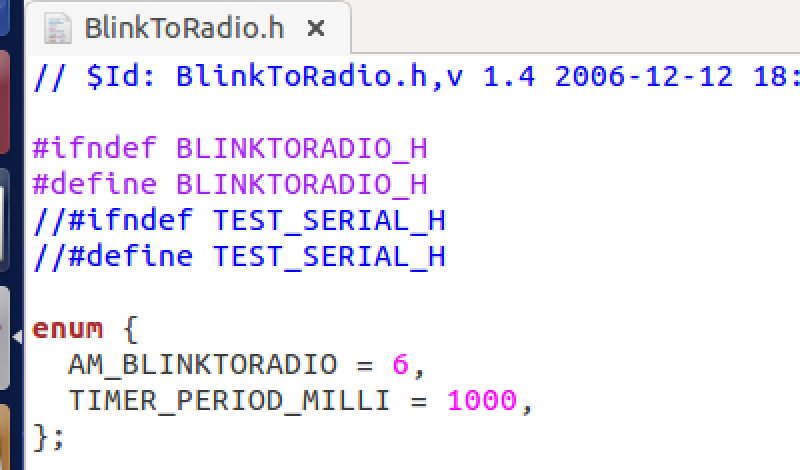


图1.3 修改时延

最后我们将数据传输方式由广播改为单播。我们可以通过为两个节点烧写不同的程序，使得一个节点负责发送，一个负责收。在代码中为这两个节点赋不同的节点号。其中一个节点5会把自己的counter值发送出去（此时是以广播形式），如下：

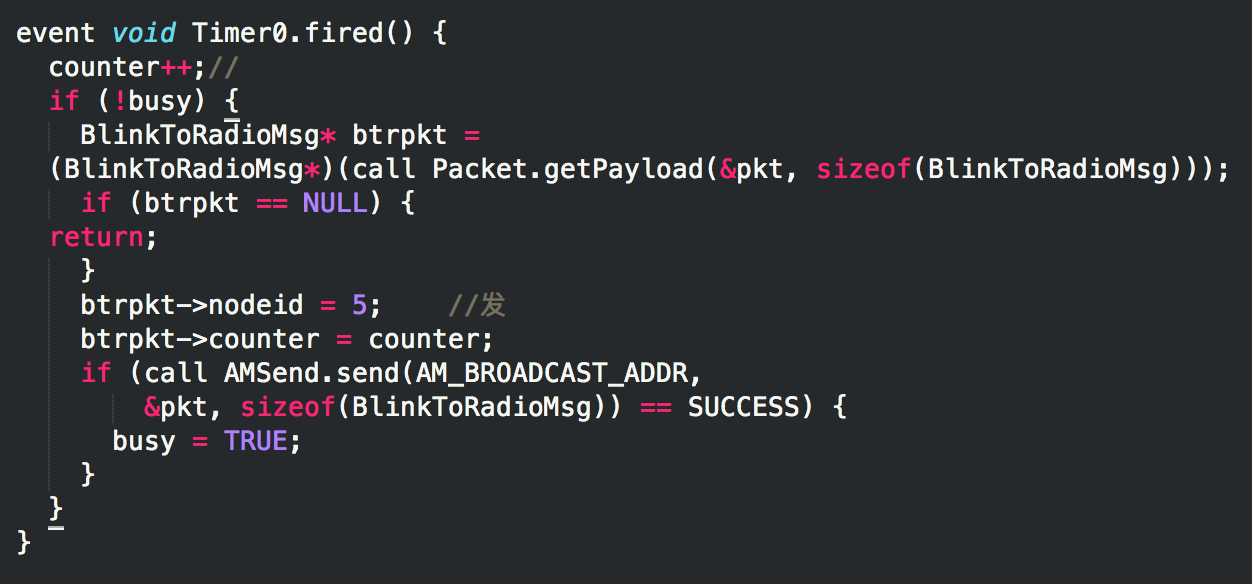


图1.4 发送部分

然后在该节点的接收部分，我们限定只有收到4号节点的包，才会亮对应的led灯，如下：

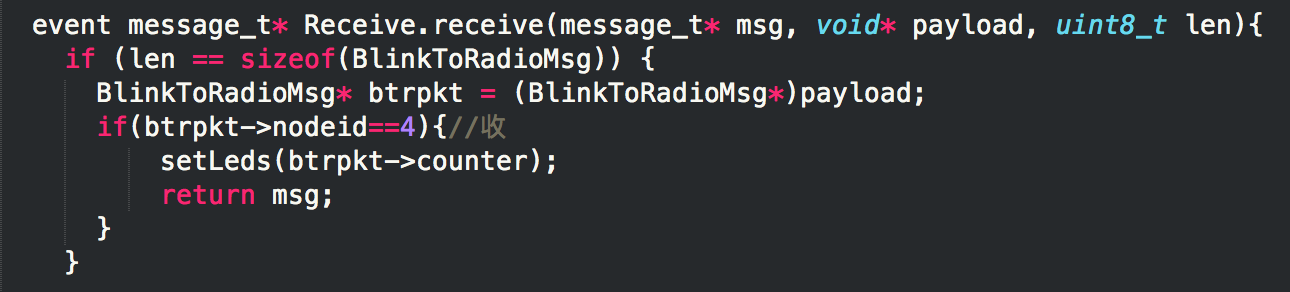


图1.5 接收部分

## 4实验结果分析

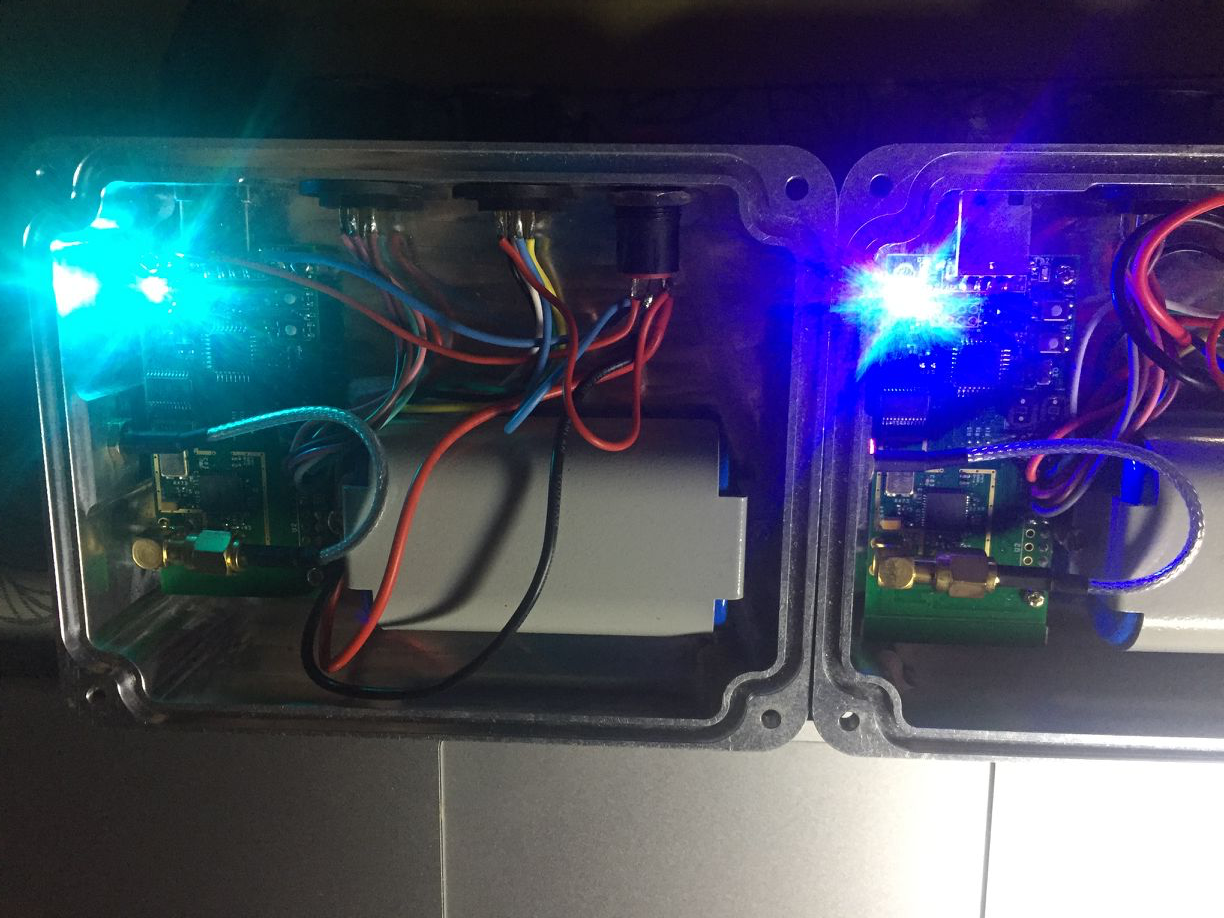


图1.6 实验现象

## 5心得体会与总结

本次实验是物联网通信技术实验的第一次实验。实验主要是对BlinkToRadio程序的熟悉，并且通过适当的程序修改实现了广播转为单播。但是在实现的时候还是有一些问题。

首先是节点发送信息的方式还是广播形式，只是接受部分改为只接收对应节点的包，所以这些节点发送包的时候，还是会对其他节点产生影响。

# 实验二

## 1 实验目的与要求

本实验的目的是实现节点和PC间的串口双向通讯，通过串口连接，PC可以从网络收集其他节点的数据，也可以发送数据或者命令到节点，因此，串口通信编程是无线传感器网络中的重要内容。

## 2 实验内容

1.修改 BlinkToRadio 程序实现简单的基站程序 RadioAndSerial．使得当RadioAndSerial 接收到无线数据包时 Led2 闪烁，并将数据包转发到串口；当串口接收到数据包时，Led0 闪烁，并将数据包转发到无线模块；

2.使用 mig 创建 BlinkToRadioMsg 的 java 对象，BlinkToRadio 发送的消息由 RadioAndSerial 接收并转发到串口，然后使用 MsgReader 读取 BlinkToRadioMsg 对象，展示接收到的数据。

## 3 实验过程与结果

## 4实验结果分析

## 5心得体会与总结

# 实验三

## 1 实验目的与要求

本实验的目的是实现节点和节点间的无线通讯和路由转发，可以使用串口发送消息到基站(指定编号为1的节点为基站)，然后转发到指定节点，或者从某个节点定时发送数据，经过固定路径可以转发到基站节点，通过简单的静态路由实验了解无线传感网络的数据路由过程和传感器数据采集过程。

## 2 实验内容

修改BlinkToRadio和RadioAndSerial，实线以下功能:

1.节点之间通过单播的方式收发数据;

2.节点路由方式为:目标节点编号大于自身节点编号的，将数据转发到编号为自身节点编号+1的节点，目标节点编号小于自身节点编号的，转发到编号自身节点编号-1的节点，节点只能直接与相邻编号的节点通讯，即数据只能在相邻节点之间转发。

3.中继节点的 LED 显示目标节点的编号，停留时间为1s, 目标节点LED 显示发送过来，停留时间为 3s, 数据包只转发一次，不进行重复发送，数据到达目标节点后计数值改为原来的计数值加上该节点的节点编号，沿原路径返回到 基站;

4.指定编号为1的节点为基站，基站首先显示目标节点编号，最后显示节点3的counter值加上发送过去的值基站进行串口读写为广播方式,如果基站节点收到数据，目标节点为自身时将数据发送到串口，否则发送到无线模块;

## 3 实验过程与结果

## 4实验结果分析

## 5心得体会与总结

附录：(各人实验程序代码)

第一次实验：

#include <Timer.h>

#include "BlinkToRadio.h"

module BlinkToRadioC {

uses interface Boot;

uses interface Leds;

uses interface Timer<TMilli> as Timer0;

uses interface Packet;

uses interface AMPacket;

uses interface AMSend;

uses interface Receive;

uses interface SplitControl as AMControl;

}

implementation {

uint16\_t counter;

message\_t pkt;

bool busy = FALSE;

void setLeds(uint16\_t val) {

if (val & 0x01)

call Leds.led0On();

else

call Leds.led0Off();

if (val & 0x02)

call Leds.led1On();

else

call Leds.led1Off();

if (val & 0x04)

call Leds.led2On();

else

call Leds.led2Off();

}

第二次实验：

BlinkToRadio部分：

#include <Timer.h>

#include "BlinkToRadio.h"

module BlinkToRadioC {

uses interface Boot;

uses interface Leds;

uses interface Timer<TMilli> as Timer0;

uses interface Packet;

uses interface AMPacket;

uses interface AMSend;

uses interface Receive;

uses interface SplitControl as AMControl;

}

event void Boot.booted() {

call AMControl.start();

}

event void AMControl.startDone(error\_t err) {

if (err == SUCCESS) {

call Timer0.startPeriodic(TIMER\_PERIOD\_MILLI);

}

else {

call AMControl.start();

}

}

event void AMControl.stopDone(error\_t err) {

}

event void Timer0.fired() {

counter++;

if (!busy) {

BlinkToRadioMsg\* btrpkt =

(BlinkToRadioMsg\*)(call Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));

if (btrpkt == NULL) {

return;

}

btrpkt->nodeid = TOS\_NODE\_ID;

btrpkt->counter = counter;

if (call AMSend.send(AM\_BROADCAST\_ADDR,

&pkt, sizeof(BlinkToRadioMsg)) == SUCCESS) {

busy = TRUE;

}

}

}

RadioAndSerial部分：

#include <Timer.h>

#include "BlinkToRadio.h"

module BlinkToRadioC {

uses interface Boot;

uses interface Leds;

uses interface Timer<TMilli> as Timer0;

uses interface Packet as RadioPacket;

uses interface AMPacket as RadioAMPacket;

uses interface AMSend as RadioAMSend;

uses interface Receive as RadioReceive;

uses interface SplitControl as RadioAMControl;

uses interface Packet as SerialPacket;

uses interface Receive as SerialReceive;

uses interface AMSend as SerialAMSend;

uses interface SplitControl as SerialControl;

}

implementation {

message\_t packet;

uint16\_t counter;

message\_t pkt;

bool busy = FALSE;

void setLeds(uint16\_t val) {

if (val & 0x01)

call Leds.led0On();

else

call Leds.led0Off();

if (val & 0x02)

call Leds.led1On();

else

call Leds.led1Off();

if (val & 0x04)

call Leds.led2On();

else

call Leds.led2Off();

}

event void Boot.booted() {

call RadioAMControl.start();

call SerialControl.start();

}

event void RadioAMControl.startDone(error\_t err) {

if (err == SUCCESS) {

call Timer0.startPeriodic(TIMER\_PERIOD\_MILLI);

}

else {

call RadioAMControl.start();

}

}

event void SerialControl.startDone(error\_t err) {

if (err == SUCCESS) {

call Timer0.startPeriodic(1000);

}

}

第三次实验：

BlinkToRadio部分：

#include <Timer.h>

#include "BlinkToRadio.h"

module BlinkToRadioC {

uses interface Boot;

uses interface Leds;

uses interface Timer<TMilli> as Timer0;

uses interface Packet;

uses interface AMPacket;

uses interface AMSend;

uses interface Receive;

uses interface SplitControl as AMControl;

}

implementation {

uint16\_t counter;

message\_t pkt;

bool busy = FALSE;

uint16\_t aim\_node = 1;

uint16\_t nodeid = 1;

void setLeds(uint16\_t val) {

if (val & 0x01)

call Leds.led0On();

else

call Leds.led0Off();

if (val & 0x02)

call Leds.led1On();

else

call Leds.led1Off();

if (val & 0x04)

call Leds.led2On();

else

call Leds.led2Off();

}