Rssi Demo

From TinyOS Wiki

This lesson is to give an example on how to get Rssi readings from incoming packets. It consists of two applications:

- a simple application that periodically sends messages over the radio, and
- a modified version of BaseStation that includes the Rssi data on the payload of messages received over the radio before forwarding them over to the serial link.

It will also be shown how to get RSSI noise floor readings, to estimate the channel background noise when no node is transmitting.

Currently, this demo works only for motes that use CC1000 or CC2420 radio transceivers, but it should contain enough information for porting it to other chips that provide RSSI in a similar manner.

Contents

Introduction

RSSI (http://en.wikipedia.org/wiki/Received_Signal_Strength_Indication) is an acronym for Received Signal Strength Indication. It is a measure of the signal power on the radio link, usually in the units of dBm (http://en.wikipedia.org/wiki/DBm) , while a message is being received. It can be used for estimating node connectivity and node distance (although the relation between distance and RSSI is noisy and not straightforward), among other things. Another usage of RSSI is to sample the channel power when no node is transmitting to estimate the background noise, also known as noise floor.

According to TinyOS' HAA <ref name="tep2">

```
TEP 2: Hardware Abstraction Architecture (http://www.tinyos.net/tinyos-2.x/doc/html/tep2.html)
```

</ref> the RSSI data is not provided by the standard platform independent Hardware Interface Layer (represented interfaces `Packet` and `AMPacket`, as explained in TEP 116 <ref name="tep116">

```
TEP 116: Packet Protocols (http://www.tinyos.net/tinyos-2.x/doc/html/tep116.html)
```

</ref>). RSSI must be accessed by a platform-specific HAL interface, as in the case of the CC2420 or by a specific field of the `message_t` struct as defined in the `platform_message.h` <ref name="tep111">

```
TEP 111: message_t (http://www.tinyos.net/tinyos-2.x/doc/html/tep111.html)
```

</ref> like in the case of the CC1000.

The RSSI values given by TinyOS are usually not in dBm units, and should be converted by the platform-specific relation to get meaningful data out of it.

Demo Application

Since RSSI is very platform-specific it will now be shown a hands-on demo to explain how to get RSSI readings from incoming packets. This demo is located in $TOSROOT/apps/tutorials/RssiDemo .

Intercept Base

InterceptBase is a modified version of the `BaseStation` component that provides the `Intercept` interface to allow the user application to inspect and modify radio and serial messages before they are forwarded to the other link, and to decide whether they shall be forwarded or not.

Sending Mote

SendingMote is the application to be put in the mote that sends the message whose RSSI will be read by the base. It contains a simple logic to periodically send a RssiMsg, as defined bellow:

```
typedef nx_struct RssiMsg{
  nx_uint16_t rssi;
} RssiMsg;
```

The RssiMsg is sent empty, it is the base that will include the RSSI value in the message.

Rssi Base

RssiBase is the application that will be put in a node connected to the serial port and will effectively read the RSSI. It uses the InterceptBase component to forward the messages over the radio but intercept the RssiMsg before it is forwarded and include the RSSI value in it. It contains some cryptic macros so it can work correctly with chips that use either the CC2420 or CC1000 radio. First, let us analyse the forward event:

```
event bool RssiMsgIntercept.forward(message_t *msg, void *payload, uint8_t len) {
  RssiMsg *rssiMsg = (RssiMsg*) payload;
  rssiMsg->rssi = getRssi(msg);

  return TRUE;
}
```

It always forwards the messages (always returns true) but modifies the payload by including the rssi. Let us now analyse the `getRssi(message_t *)` method:

```
#ifdef __CC2420_H__
uint16_t getRssi(message_t *msg){
  return (uint16_t) call CC2420Packet.getRssi(msg);
}
```

```
#elif defined(CC1K_RADIO_MSG_H)
   uint16_t getRssi(message_t *msg){
   cc1000_metadata_t *md =(cc1000_metadata_t*) msg->metadata;
   return md->strength_or_preamble;
 }
#else
 #error Radio chip not supported! This demo currently works only \
        for motes with CC1000 or CC2420 radios.
#endif //__CC2420_H__
```

That code, for platforms with the CC2420 radio, gets the RSSI from the `getRssi(message_t *)` method of the `CC2420Packet` interface, provided by the `CC2420ActiveMessageC` HAL component. For platforms with the CC1000 radio the RSSI is extracted from the packet metadata.

Java Application

A java application was called RssiDemo.java was created to see the results on the computer. For more explanations on how it works, look in the Mote-PC serial communication and SerialForwarder tutorial lesson, on which the file was based.

To build it simply give the `make` command (no platform needed) in the $TOSROOT/apps/tutorials/RssiDemo directory and RssiDemo.class will be built. To run the application first install SendingMote in a mote, RssiBase in another which is connected to the computer's serial port and run it by issuing the `java RssiDemo` command. Be sure to set the MOTECOM variable before, or using the -comm option. The output should be something like this:

```
Rssi Message received from node 1: Rssi = -14
Rssi Message received from node 1: Rssi = -1
Rssi Message received from node 1: Rssi = 2
Rssi Message received from node 1: Rssi = -2
Rssi Message received from node 1: Rssi = 2
Rssi Message received from node 1: Rssi = 0
Rssi Message received from node 1: Rssi = -2
Rssi Message received from node 1: Rssi = 2
Rssi Message received from node 1: Rssi = 2
Rssi Message received from node 1: Rssi = -10
Rssi Message received from node 1: Rssi = 1
Rssi Message received from node 1: Rssi = -6
Rssi Message received from node 1: Rssi = -4
Rssi Message received from node 1: Rssi = 0
Rssi Message received from node 1: Rssi = 0
```

The above test was performed with a tmote mini plus development board as base and a tmote invent as sending mote. It should be noted that this app prints only the raw RSSI values. The readings should be converted to dBm for more meaningful information.

Noise Floor Reading

To perform RSSI noise floor reading on platforms with the CC2420 radio, the HAL component CC2420ControlP should be used. On platforms which use the CC1000 radio, the CC1000RssiP component should be used. For detailed information on specifics, look up the module's documentation.

Related Documentation

- Simple One dimensional tracking with RSSI (http://mythicalcomputer.blogspot.com/2008/11/tracking-using-rssi-application-in.html)

References

<references/>