multi-Spin v2.0 Cheatsheet

Author: Maurizio Bocca (maurizio.bocca@utah.edu)
Date: 01/15/2014

multi-Spin is a multi-frequency, TDMA communication protocol specifically designed for RF sensor networks applications. *multi-Spin* defines the order of transmission of the RF sensors composing the system and synchronizes their switching on different frequency channels without the need of a central command or coordination from a central unit (which we call *listen node*).

In multi-Spin, time is divided into *slots*, *cycles* and *rounds*:

- A round includes *CHANNELS_NUMBER* TDMA cycles. The constant *CHANNELS_NUMBER* is defined in channels.h and represents the number of frequency channels the RF sensors use for communicating among them. These channels are also defined in channels.h and are included in the *channel_sequence* array. In the interval of time required to complete a round, the system measures the RSS of each link on each selected frequency channel.
- A cycle is composed of (*MAX_NUM_NODES*+3) slots. The constant *MAX_NUM_NODES* is defined in spin_multichannel.h and represents the number of RF sensors composing the network. In the interval of time required to complete a cycle, the system measures the RSS of each link on a single (i.e., the current one) frequency channel. The additional 3 slots can be used by the application developer in different ways: for instance, they can be used to broadcast specific commands from the listen node to the RF sensors (e.g., *stop*, *suspend*, *skip channel*, etc.), or to reconfigure the values of the parameters of the application (such as the frequency channels to be measured, the transmitting power of the nodes, etc.).
- In each slot, only one RF sensor is allowed to broadcast a packet, while all the other nodes are in receiving mode. The order of transmission of the RF sensors is based on their built-in ID number, which is defined as *THIS_NODE_ID* in the *main.c* file located inside the *RF_sensor* folder. The length, in terms of ticks of the clock counter, of a slot id defined by the constant *SLOT_LENGTH*.

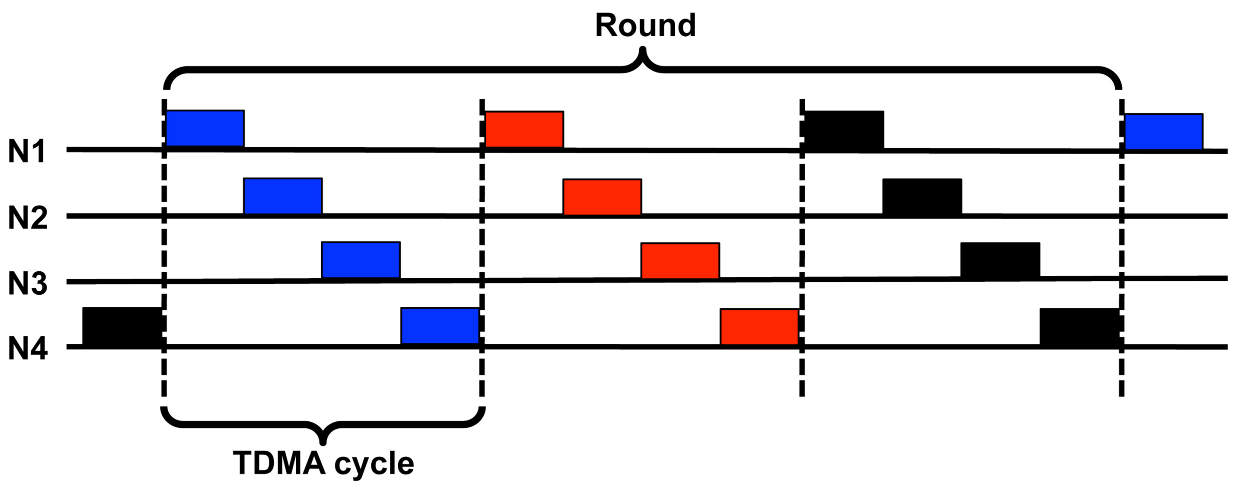

Fig. 1. *multi-Spin* division of time into slots (solid blocks), cycles, and rounds.

Packet Format

The packets broadcasted by the RF sensors include:

- A 16 bit integer packet counter which is increased by one at each broadcasted packet;
- The ID number of the RF sensor broadcasting the packet;
- An array of *MAX_NUM_NODES* RSS measurements. This array is used to store the most recent RSS measurements of the packets received from the other nodes. The *n*-th element of the array has the RSS of the last packet broadcasted by the RF sensor having ID number *n.* Consequently, as shown in Table 1, the elements of the RSS array preceding the *THIS_NODE_ID*-th element refer to the frequency channel used in the current cycle, while the elements following the *THIS_NODE_ID*-th element refer to the frequency channel used in the previous cycle, i.e., the frequency channel preceding the current one in the *channel_sequence* array.
- The frequency channel on which the packet was broadcasted;

The packet type *spinPacket_t* is defined in spin_multichannel.h.

TX ID NUMBER	RSS MEASUREMENTS VECTOR	FREQUENCY CHANNEL
1	**7f c3 c8 c5**	**26**
2	**c4 7f c6 b**	**26**
3	**cb bd 7f c3**	**26**
4	**c6 7f c4 7f**	**26**
1	**7f c1 ca c4**	**15**
2	**c1 7f be c3**	**15**
3	**c3 c6 7f c3**	**15**
4	**c1 c6 c9 7f**	**15**
1	**7f bf c3 bf**	**20**
2	**c3 7f c6 c5**	**20**
3	**ca c4 7f c8**	**20**
4	**c3 c7 c0 7f**	**20**
1	**7f c0 7f c1**	**26**
2	**c4 7f c5 c7**	**26**
3	**ca c6 7f bf**	**26**
4	**c6 bb c4 7f**	**26**

Table 1. The payload of the packets transmitted by four RF sensors communicating on three different frequency channels (15, 20, 26). The different colors are used to indicate to which frequency channel the RSS measurements of each packet correspond. In this case, RSS measurements are in hex format. A 7f in the *p*-th element of the RSS measurements vector indicates that the last packet broadcasted by the RF sensor having ID number *p* was not received by the transmitting RF sensor.

Listen Node Operation:

The listen node is normally connected to a USB port of a standard laptop or computer. It collects all the RSS measurements by overhearing the packets broadcasted by the RF sensors. To do so, the listen node has to maintain synchronicity with the RF sensors.
The functioning of the listen node is based on two interrupt service routines (ISR):

- The *channel_hoppingISR*, which fires whenever the timer associated to *next_channel_time* and indicating the time remaining to the next channel switching expires. At the reception of a packet, the listen node updates *next_channel_time* as:

  next_channel_time = (*MAX_NUM_NODES* - *int_TX_id* + 2) * *SLOT_LENGTH*

  where *int_TX_id* is the ID number of the RF sensor that broadcasted the received packet.
- The *reset_radio_channelISR*, which fires whenever the timer associated to *next_reset_radio_time* expires. *next_reset_radio_time* indicates an interval of time after which, if no packets were received, the listen node switches to the *default* frequency channel, i.e., the first frequency channel of the *channel_sequence* array defined in channels.h. The listen node remains on the default frequency channel until a new packet is received.

The listen node transfers the content (packet counter and RSS measurements) of the received packets through the serial port of the laptop or computer to which is connected.

RF Sensor Operation:

In normal conditions, RF sensors have to *a*) broadcast packets at the right time (or slot), *b*) receive the packets broadcasted by the neighboring RF sensors, measure the RSS, and properly store it in the RSS measurements vector, and *c*) switch frequency channel synchronously.
The functioning of the RF sensors is based on two ISRs:

- The *channel_hoppingISR*, which fires in exactly the same way as for the listen node. However, the RF sensors exploit an additional safety mechanism to avoid keeping on switching frequency channel after having fallen out-of-sync. If an RF sensor does not receive a packet in the interval of time corresponding to *RESET_LIMIT* cycles, it switches back to the default frequency channel and then starts broadcasting packets at a pseudo-random interval in order to look for another RF sensor in the area. The reception of a packet allows the RF sensors to synchronize themselves and communicate on the defined frequency channels.
- The *next_TX_timeISR*, which fires whenever the timer associated to *next_TX_time* and indicating the time left to the transmission of a packet expires. At the reception of a packet, the RF sensor updates *next_TX_time* as:

```
if (int_TX_id > THIS_NODE_ID)
{
   next_TX_time = (MAX_NUM_NODES - int_TX_id + THIS_NODE_ID + 2) * SLOT_LENGTH;
}
else
{
   next_TX_time = (THIS_NODE_ID - 1 - int_TX_id) * SLOT_LENGTH;
}
```

The *if* branch considers the case in which the RF sensor received a packet broadcasted by a RF sensor having a higher ID number. In this case, *next_TX_time* includes:

- (*MAX_NUM_NODES* - *int_TX_id*) slots, which are required by the RF sensors that have not yet broadcasted their packet in the current cycle;
- 3 slots, that are left to the application developer;
- (*THIS_NODE_ID -1*) slots, required by the preceding RF sensors to broadcast their packets.

The *else* branch considers the case in which the RF sensor received a packet broadcasted by a RF sensor having a lower ID number. In this case, *next_TX_time* includes the slots required by the RF sensors following *int_TX_id* and preceding *THIS_NODE_ID*.