I. Goal

We are aiming to provide each Ckbot module with wireless communication capability.
I the current settings, two wixels are used to connect a cluster of modules with the controlling computer.

II. Protocol

Each ckbot module of a cluster will have a wixel module attached, listening at a specified channel.

Master will use two wixels. One for transmitting commands to ckbot module, and anthoer for receiving query result from ckbot module.

Most of the commands will be setter commands, such as setting the position of a specified module.
Less frequently, there will be getter commands that will ask a specified module for data, such as getting position.

1. Sending setter commands

The wixel of master in tranciving mode will be broadcasting commands into the channel that belongs to a cluster. The radio channel is acting like a bus as in current settings.

Each command will contain a module ID, thus only the designated module will respond to a single command.

In practice, setter commands are sent in series. We assume the impact of losing a single command is not critical, as we care more about the most recent command.

2. Sending getter commands

III. Testing Metrics

1. Consecutive loss or not

In file, `send_only/send_only.c`,
`int sendPacket(int16 i) is a function that construct the packet.`
We used packet of length 6 bytes, which is 2 bytes of sequence number, plus 4 bytes of timestamp.

My experiment setup is described as follows:

1. Master sending packet with sequence number `i` to a receiver at channel 128.
2. Master sending packet with sequence number `i` (Same seq number) to a receiver at channel 228.
3. Increase seq number `i` by 1 until 1000.
4. Repeat 1-3.

Some example from the data I collected.

The first column is next seq number minus cur seq number
The second column is a count.
As we can see, most packet losses are just missing one packet.

----
file: `ch228/two_meter_228.log`
total packet sent: 16639

2: 998
3: 87
4: 8
5: 2
9: 1file: `ch128/five_meter_128.log`
----

total packet sent: 9430

0: 66 **/* next sequence number - cur sequence number == 0, there is a buffer flush issue */**
2: 550
3: 27
4: 3
6: 1
9: 1

----
file: `ch228/five_meter_228.log`
total packet sent: 8790

2: 865
3: 104

4: 11
5: 3
6: 1
7: 1

2. Loss related to packet length
Future work left is to test the loss rate for various length packets.

3. Loss related with channel or mode switching

Future work need to change the experiment setup. Such as using a master device just for sending, another master listener for receiving.

A slave device will
1. Just listen.
2. listen, and switch mode to send back to master. Then switch back to listen.

IV. About Wixel

Pololu-Wixel-SDK http://pololu.github.com/wixel-sdk/: The source code of the SDK is also abialable in github.

For windows, Wixel provides a IDE that includes most of what you will need.
Linux or Mac user need to install SDCC, with Pollolu-Wixel-SDK.

In default setup, you could simply go to the `pololu-wixel-sdk` directory. Here is the SDK structure, if your app is called `app_1`.

```
pololu-wixel-sdk
├── LICENSE.txt
├── Makefile
├── README.txt
├── apps
    ├── app_1
        ├── app_1.c
        ├── app_1.wxl
        ├── .
        ├── .
        ├── .
    ├── app_2
    ├── app_n
├── apps.mk
├── installer
├── libraries
├── make_all.bat
└── wixelcmd
```

Linux, Mac do:

```
pololu-wixel-sdk $ make
```

Windows do:

```
pololu-wixel-sdk $ make_all.bat
```

Then every app under `apps` directory will be compiled, the result file `app.wxl` will be created in the directory where you placed your source file `app.c`.

The next step is just use the loader, which is an easily installed GUI based program (for Windows, Mac, Linux), to load the `*.wxl` file into your wixel module.

The current radio testing code is using radio_queue.h, which provided the following functions:

`void`	`radioQueueInit (void)`
`uint8`	`radioQueueTxAvailable (void)`
`uint8`	`radioQueueTxQueued (void)`
`uint8 XDATA *`	`radioQueueTxCurrentPacket (void)`
`void`	`radioQueueTxSendPacket (void)`
`uint8 XDATA *`	`radioQueueRxCurrentPacket (void)`
`void`	`radioQueueRxDoneWithPacket (void)`

V. About cc2511f32, the core of Wixel

VI. Next Steps

1. Improve the draft protocol

2. radio_queue buffer flush issue

In previous testing settings. We used one sender and two receivers.
The sender will send packet with sequence number `i` ,where `i` range from `1` to `1000`, to both

receivers. We set the sender to change channel between 128 and 228 for each sequence `i` , which are listened by two receivers respectively. We found many packets were sent to only one channel, such channel 128 received two packets with `i=100`.