4. nRF24 on ATTiny84 - Conversation

Purpose: Having successfully demonstrated ability to have nRF24 that is connected to the ATTiny84 continuously send out data, and for the chip on the Raspberry Pi to see it (See note titled "**nRF24 on ATTiny84 - Blind Data Send**."); now I want to get a back-and-forth 'conversation' going between the two. So the goal here is to have the ATTiny84 send out data packets; the RPi to receive them and reply back with an acknowledgement packet that the ATTiny84 sees in return, and then have the ATTiny84 make some update to it's next data packet and send that updated packet out to the RPi. What I want to see on the RPi side is the revised data packets coming in from the ATTiny84 to "prove" the two chips are in a conversation. (See note titled "**nRF24 on Raspberry Pi**.")

Goal: From the monitor on the Raspberry Pi, see that a back-and-forth conversation between the ATTiny84 and RPi is occurring.

Key References --

What I finally discovered on 3/25 is the below site is the key resource for the nRF24:

nRF24 / RF24 github repo @ https://github.com/nRF24/RF24

See the pin connection table at the bottom of the following page (which also appears to be the home of the documentation that goes with the above github repo):

https://rf24.readthedocs.io/en/latest/

Another key reference is the so-called Arduino Core for the ATTiny family. Above reference says the code there is based on the 'SpenceKonde' core. The home for this is @ https://github.com/SpenceKonde/ATTinyCore. See the readme file there for documentation, including how to reference pins inside the code that match up to the processor physical pins. It says to use the Port and Number as reference. E.g., ATTiny84 physical pin 12 would be referenced as 'PA-1' in the code. As in: `const int ledPin = PIN_PA1; // the pin that the led is wired to`

"Addressing" with the RF24's is confusing to me. So called 'pipes' are opened for reading and writing; and there are notes in the Arduino RF24.h file about addresses associated with these pipes. So far I have not been able to work out what's really going on. In RF24.h there was a reference to the below web page re addressing:

Improve RF24 Radio Performance With Proper Addressing Schemes @ http://maniacalbits.blogspot.com/2013/04/rf24-addressing-nrf24l01-radios-require.html

05/29/2022 --

Initial Success!

ATTiny Software: AckPayloads_jrrMod. In Arduino IDE I modified the **`acknowledgementPayloads.cpp`** code for the ATTiny to simplify it by removing all the serial-monitor 'print' statements, and by removing the Receive mode option. The Arduino sketch is named: AckPayloads_jrrMod. So for the ATTiny84 side the code creates a device that is transmit only.

> **Use `acknowledgementPayloads` on Raspberry Pi Side**. NOTE: This is the code that uses the 'auto-ack with payload.' Don't confuse this with the manualAcknowledgements.cpp code. So on the RPi side this pairs with `acknowledgementPayloads`.
>
> **AckPayloads_jrrMod_2**. In reconstructing this Milestone after the failure from Milestone #5 I created a tweaked tiny84 side sketch from the original AckPayloads_jrrMod - I named it AckPayloads_jrrMod_2. Did this to use the LEDs better and my now standard errorLED routine.

I also added a RED led to be able to signal errors detected in the code.

When I run the ATTiny84 device with the `acknowledgementPayloads` app on the RPi side - with no code mods by me on this first run - I get a continuous stream of:

```
Received 8 bytes on pipe 1: Hello 1 Sent: World 0
Received 8 bytes on pipe 1: Hello 3 Sent: World 2
Received 8 bytes on pipe 1: Hello 5 Sent: World 4
Received 8 bytes on pipe 1: Hello 7 Sent: World 6
Received 8 bytes on pipe 1: yyyyy 7 Sent: World 8
Received 8 bytes on pipe 1: Hello 9 Sent: World 8
Received 8 bytes on pipe 1: Hello 9 Sent: World 10
Received 8 bytes on pipe 1: Hello 11 Sent: World 10
Received 8 bytes on pipe 1: Hello 11 Sent: World 12
Received 8 bytes on pipe 1: Hello 13 Sent: World 12
Received 8 bytes on pipe 1: Hello 13 Sent: World 14
Received 8 bytes on pipe 1: Hello 15 Sent: World 14
Received 8 bytes on pipe 1: Hello 15 Sent: World 16
Received 8 bytes on pipe 1: Hello 17 Sent: World 16
Received 8 bytes on pipe 1: yyyyy 17 Sent: World 18
Received 8 bytes on pipe 1: xxxxx 17 Sent: World 18
.....
```

The "yyyyy" and "xxxxx" lines are showing that occasionally the RPi doesn't respond back properly, or in a timely manner; and/or the ATTiny84 side isn't picking up the carrier of the RPi response. Those occasional glitches are rare on my bench. The two chips screamed through to increment 255 in a a second or so with only about 5 mis-hits.

SO - OK, now I can exchange data between the two devices. Next - I think it's time to connect an analog input on the ATTiny84 side and see that varying data on the RPi side.