

Arduino with RN-XV WiFly Module

Xiaoyang Zhong

11/20/2015

This tutorial describes how to connect a RN-XV WiFly module with Arduino directly, without the support of the XBee Shield.

0. Download and Install Related Libraries

Download the *WiFlySerial*, *PString*, *Time*, and *Streaming* libraries through **Oncourse/Resources/Projects/Project 3/Arduino_wifi_libraries.zip**.

Unzip the above files, and copy all the files to **[arduino-ide]/libraries/**. Restart Arduino IDE, you would be able to use the libraries.

Download the **Arduino_wifi_examples.zip** from **Oncourse/Resources/Projects/Project 3/**, put the examples to your working folder.

1. Connections

Connect the RN-XV to Arduino through wires.

Arduino	RN-XV WiFly
Pin 2	Pin 2 (TX)
Pin 3	Pin 3 (RX)
3.3 V (voltage out)	Pin 1 (3.3 V voltage in)
GND	Pin 10 (GND)

NOTE: Arduino Pin 3 must **NOT** connect to RN-XV directly, because the output voltage does not match. Arduino Pin 3 output voltage is 5.0 V, whereas the RN-XV module can only tolerant 3.3 V. Thus, a simple voltage divider is implemented to the output of Arduino Pin 3, which is shown below.

The real picture is shown below. The schema is also provided below. (Pay attention to the correct pins)

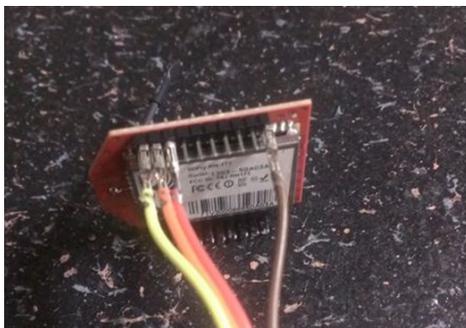


Fig. 1. RN-XV wire connection

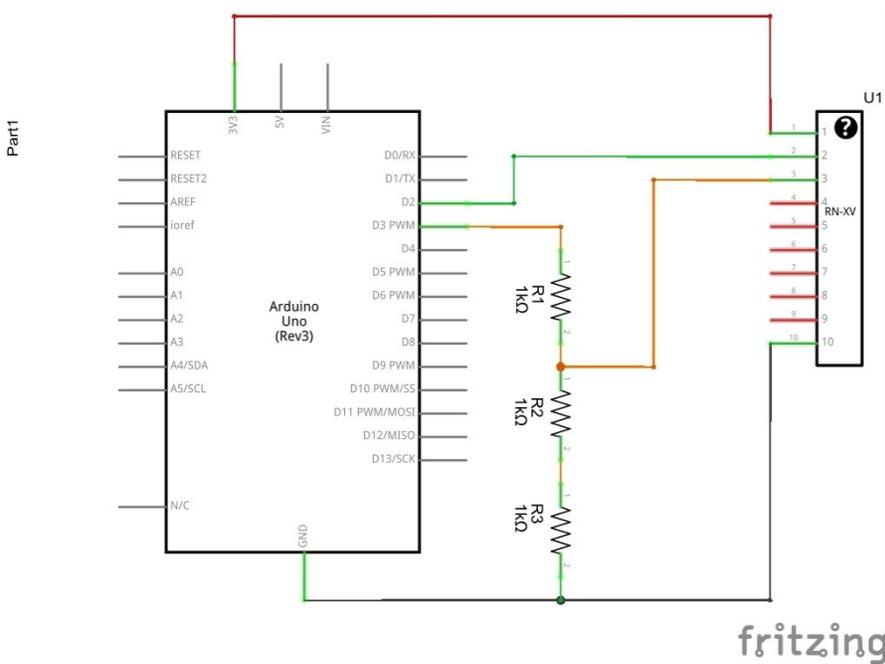


Fig. 2. Arduino and RN-XV module connection schema.

The output of Arduino Pin 3 is divided using three 1-k resistors, then to RN-XV ping 3.

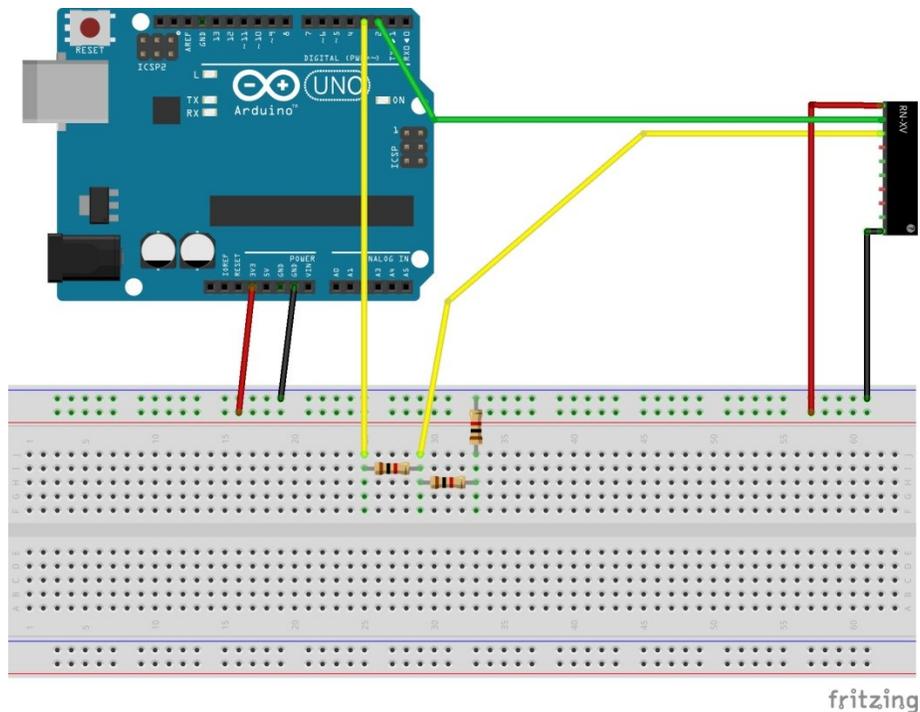


Fig. 3. Arduino and RN-XV connection

2. Run Initialization Application

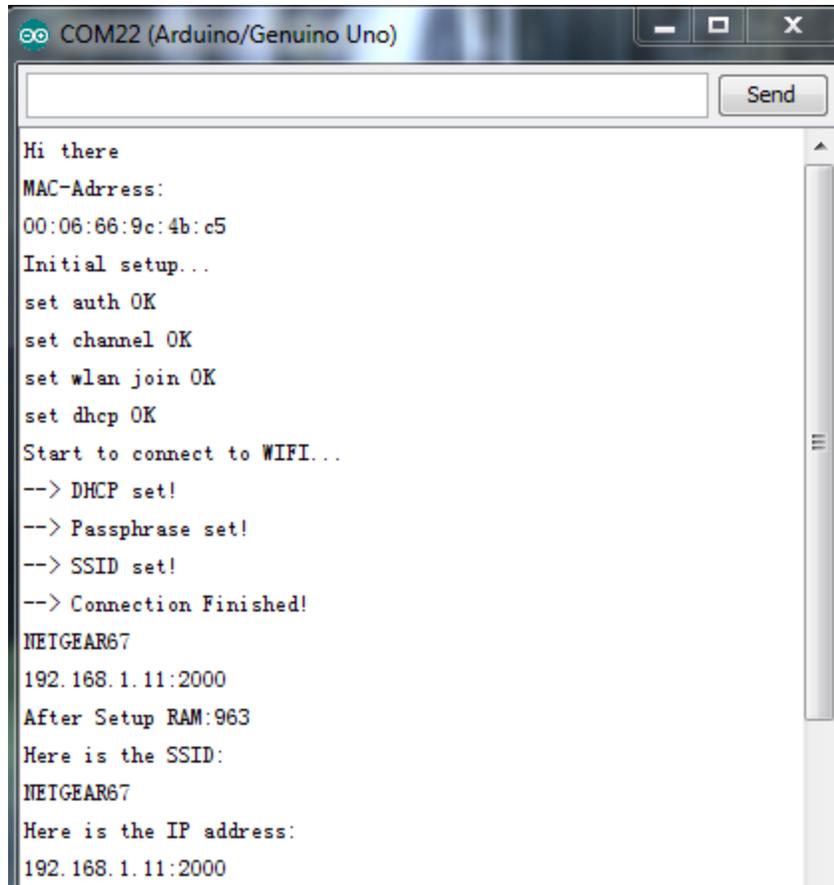
The RN-VX WiFly module needs some initial setup for the first time use. When the wires are correctly connected, the **RED** and **GREEN** leds on the RN-XV module would blink very fast.

Open the `test_wifi_init` example; modify lines 36 and 37 to include your wifi network ssid and password.

```
// Wifi parameters
char ssid[] = "yourSSID";
char passphrase[] = "yourPassword";
```

Fig. 4. WiFi parameters

Then upload the example to Arduino, you would see similar output from the *Serial Monitor* as the following figure. In the meantime, the **RED** led would go off, and **GREEN** led would blink slower, indicating the WiFi connection is established. Then **GREEN** led would blink fast in sometime, because it is transmitting data to Arduino. When the transmission is done, it would blink slower again. **Yellow** led would blink occasionally, indicating WiFly is transmitting data to the router.

The image shows a screenshot of an Arduino IDE serial monitor window. The window title is "COM22 (Arduino/Genuino Uno)". The output text is as follows:

```
Hi there
MAC-Address:
00:06:66:9c:4b:c5
Initial setup...
set auth OK
set channel OK
set wlan join OK
set dhcp OK
Start to connect to WIFI...
--> DHCP set!
--> Passphrase set!
--> SSID set!
--> Connection Finished!
HEIGEAR67
192.168.1.11:2000
After Setup RAM:963
Here is the SSID:
HEIGEAR67
Here is the IP address:
192.168.1.11:2000
```

Fig. 5. `test_wifi_init` example output.

When you see this, the WiFly module is correctly initialized. You are ready to go to the next step.

3. HTTP Example

This example shows how to setup a simple HTTP server on Arduino. This example is also included in the WiFlySerial library.

Open the **WebTime** example. This example shows how to read the time on Arduino (start from 1970 00:00:00) and send an HTTP response when you access Arduino from browser.

Modify lines 37 and 38 to include your WiFi ssid and password. Then upload the example to Arduino.

When you see the following output, your Arduino is ready to accept HTTP request.



Fig. 6. WebTime example output.

Open your browser (if Chrome does not work, use IE or Firefox), and type in the IP address and port number of Arduino:

`http://192.168.1.11:2000/`

You would see the following information in *Serial Monitor*.

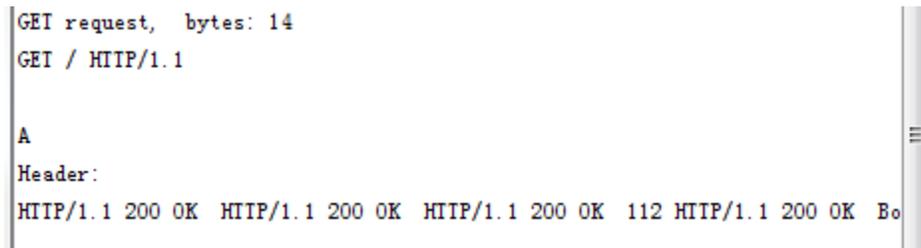


Fig. 7. WebTime example, Arduino received HTTP GET.

In the meantime, you would see the following information from your browser:

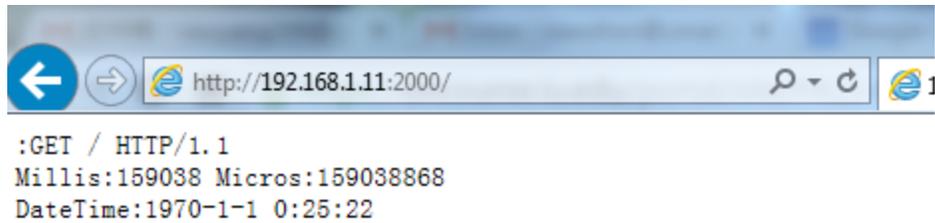


Fig. 8. WebTime example, time on browser.

If you have sensors attached, and want to post the sensor readings to the HTTP response, you would modify the function **MakeResponseBody()** to include your sensor readings.

4. UDP Example

This example shows how to send a packet using UDP protocol.

Open the **UDPSample** example. This example is based on WebTime, instead of using HTTP, it now uses UDP to **periodically** send the local time to any UDP listener.

In the meantime, run the **UDPClient.java** (inside the **UDPSample** example) using Eclipse/terminal/other java IDE.

Modify lines 67, 68 and 69 to include your WiFi ssid, password, and the IP address of your laptop, which would run a UDP listener. You can also change the HOST_PORT is you want.

```
// Set these to your local values
#define MY_WIFI_SSID "yourSSID"
#define MY_WIFI_PASSPHRASE "yourPassword"
#define MY_UDP_RECEIVING_HOST_IP "IP address of your laptop, the UDP listener"
#define MY_UDP_RECEIVING_HOST_PORT "17129"
```

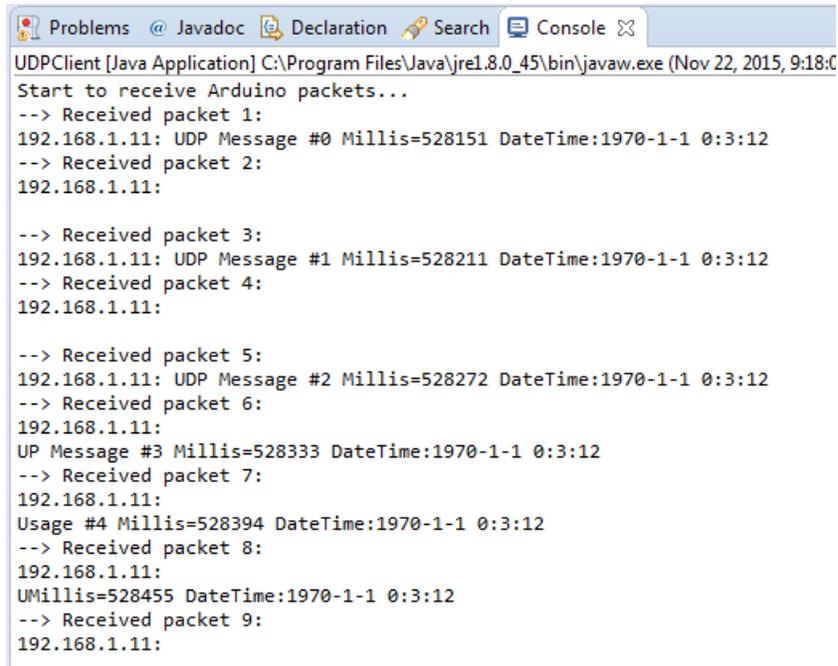
Fig. 9. UDPSample example configuration

Upload the example to Arduino, you would see the following output when the code is successfully running.

```
COM22 (Arduino/Genuino Uno)
Starting UDPSample - Please wait.
Starting UDPSample - Please wait. 965
Started WiFly, RAM :965
MAC: 00:06:66:9c:4b:c5
Leave current wifi network:0
Credentials Set, Joining NETGEAR67
Configure UDPSample Settings...
DateTime:1970-1-1 0:49:6
Initial WiFi Settings :
IP: 192.168.1.11:2000
Netmask: 192.168.1.11:2000
Gateway: 192.168.1.11:2000
DNS: 192.168.1.11:2000
RSSI: (-40) dBm
battery: 2863
After Setup RAM:965
Loop RAM:827
Starting UDP Transmissions for 5 iterations
listen with ' nc -uv -l 17129 on host 192.168.1.15 '
Press any key (and 'Send') to stop.
Switching to UDP mode...
UDP Message #0 Millis=46514 DateTime:1970-1-1 0:49:23
UDP Message #1 Millis=46574 DateTime:1970-1-1 0:49:23
UDP Message #2 Millis=46635 DateTime:1970-1-1 0:49:23
UDP Message #3 Millis=46696 DateTime:1970-1-1 0:49:23
UDP Message #4 Millis=46756 DateTime:1970-1-1 0:49:23
UDP Message #5 Millis=46818 DateTime:1970-1-1 0:49:23
Iterations completed.
Switching from UDP mode to ICP...
```

Fig. 10. UDPSample example, Arduino is now running as a UDP sender.

You can see that Arduino has sent several UDP packets in a row. Now check these packets in the **UDPCliant**. The following shows the output of Eclipse:

A screenshot of a Java IDE's console window. The title bar shows 'Problems @ Javadoc Declaration Search Console'. The main text in the console is as follows:

```
UDPCient [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (Nov 22, 2015, 9:18:00)
Start to receive Arduino packets...
--> Received packet 1:
192.168.1.11: UDP Message #0 Millis=528151 DateTime:1970-1-1 0:3:12
--> Received packet 2:
192.168.1.11:

--> Received packet 3:
192.168.1.11: UDP Message #1 Millis=528211 DateTime:1970-1-1 0:3:12
--> Received packet 4:
192.168.1.11:

--> Received packet 5:
192.168.1.11: UDP Message #2 Millis=528272 DateTime:1970-1-1 0:3:12
--> Received packet 6:
192.168.1.11:
UP Message #3 Millis=528333 DateTime:1970-1-1 0:3:12
--> Received packet 7:
192.168.1.11:
Usage #4 Millis=528394 DateTime:1970-1-1 0:3:12
--> Received packet 8:
192.168.1.11:
UMillis=528455 DateTime:1970-1-1 0:3:12
--> Received packet 9:
192.168.1.11:
```

Fig. 11. UDPSample example, message received in the UDPCient.

Reference:

<http://cairohackerspace.blogspot.com/2011/05/beginners-guide-to-connecting-and.html>

<https://roboticsgiesing.wordpress.com/2012/08/01/roving-rn-xv-wlan-shield-connected-to-arduino/>

<https://www.sparkfun.com/products/10822>

<http://www.java2s.com/Code/Java/Network-Protocol/ReceiveUDPpockets.htm>