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

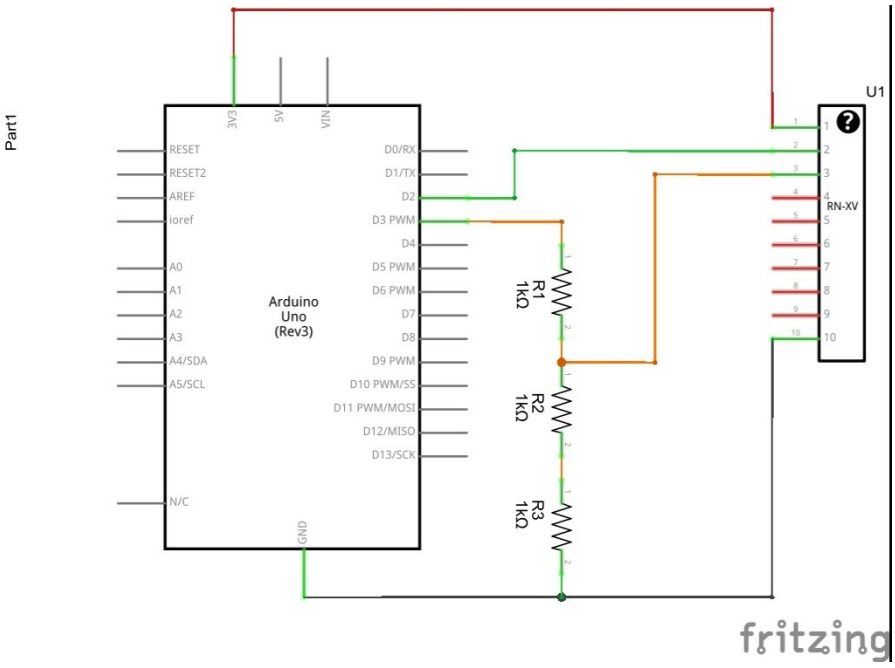

Fig. 2. Arduino and RN-XV module connection schema.

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

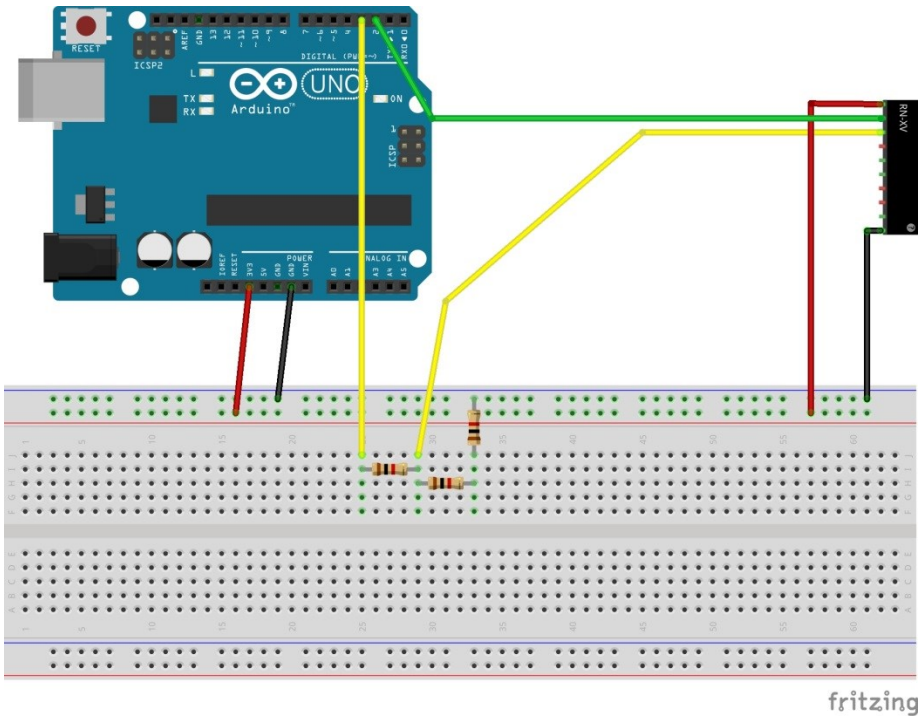

Fig. 3. Arduino and RN-XV connection

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.

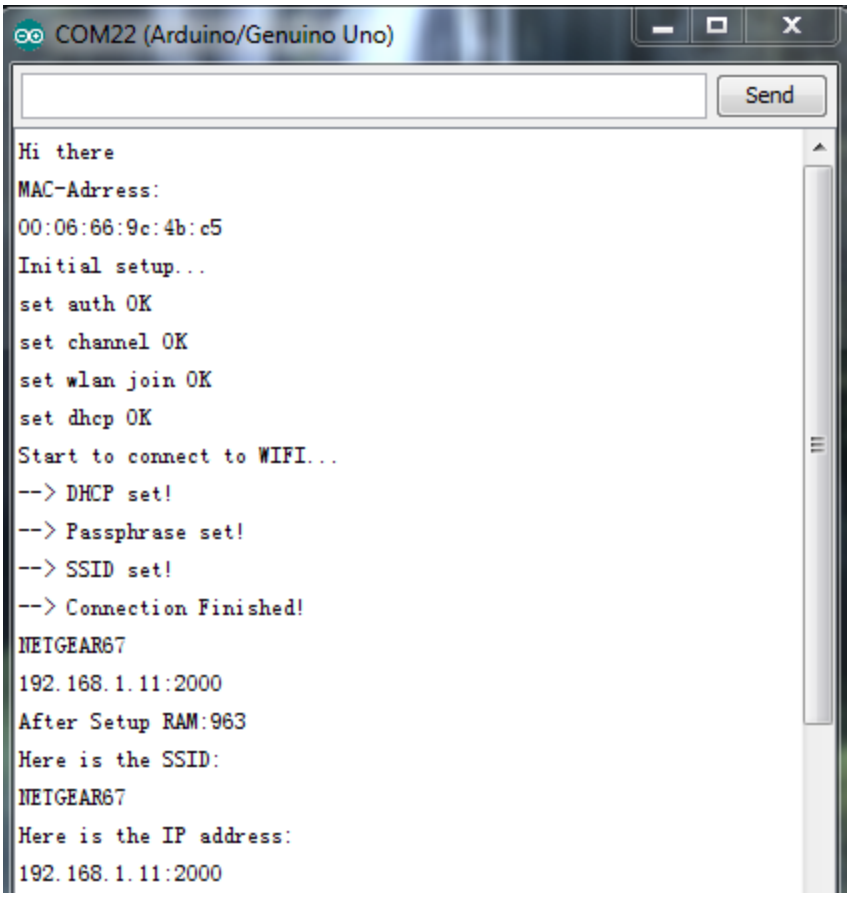

Fig. 5. **test_wifi_init** example output.

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 included 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.

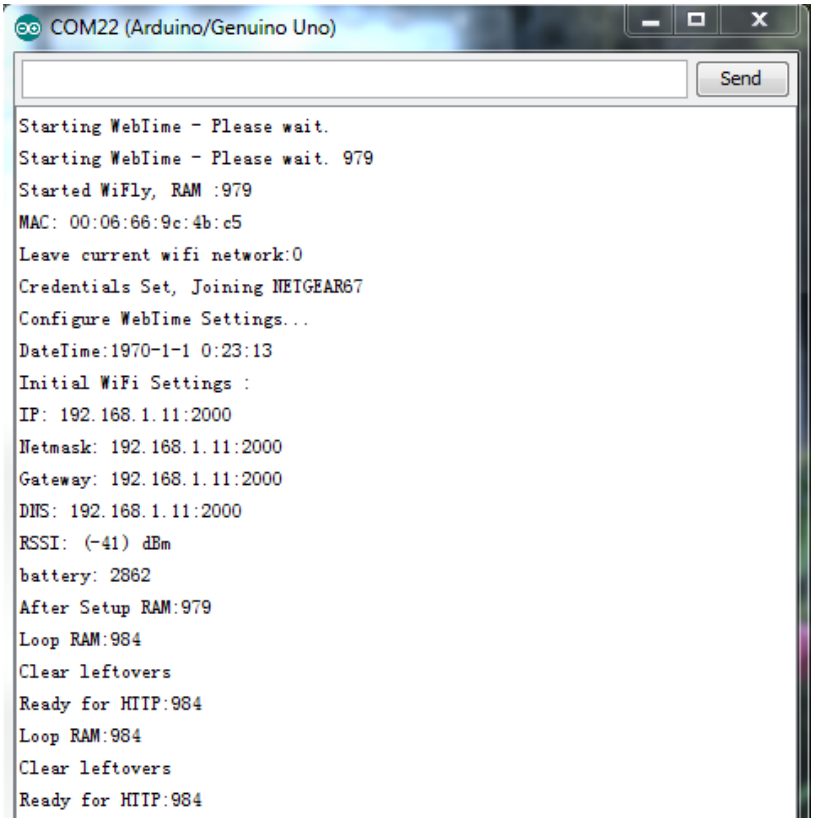

Fig. 6. WebTime example output.

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*.

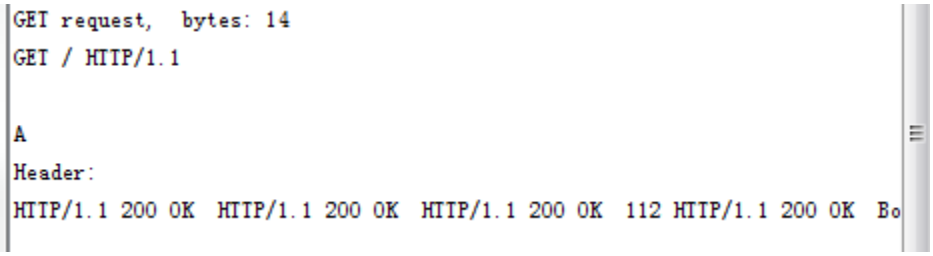

Fig. 7. WebTime example, Arduino received HTTP GET.

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

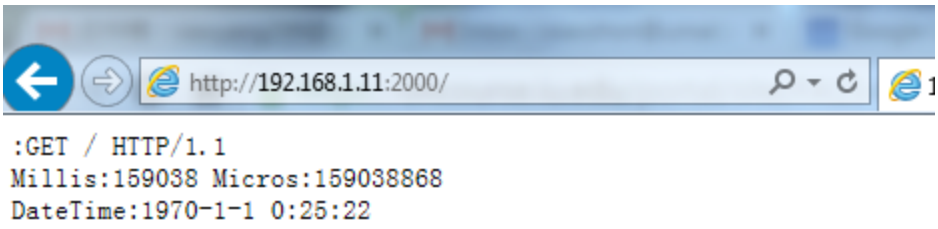

Fig. 8. WebTime example, time on browser.

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

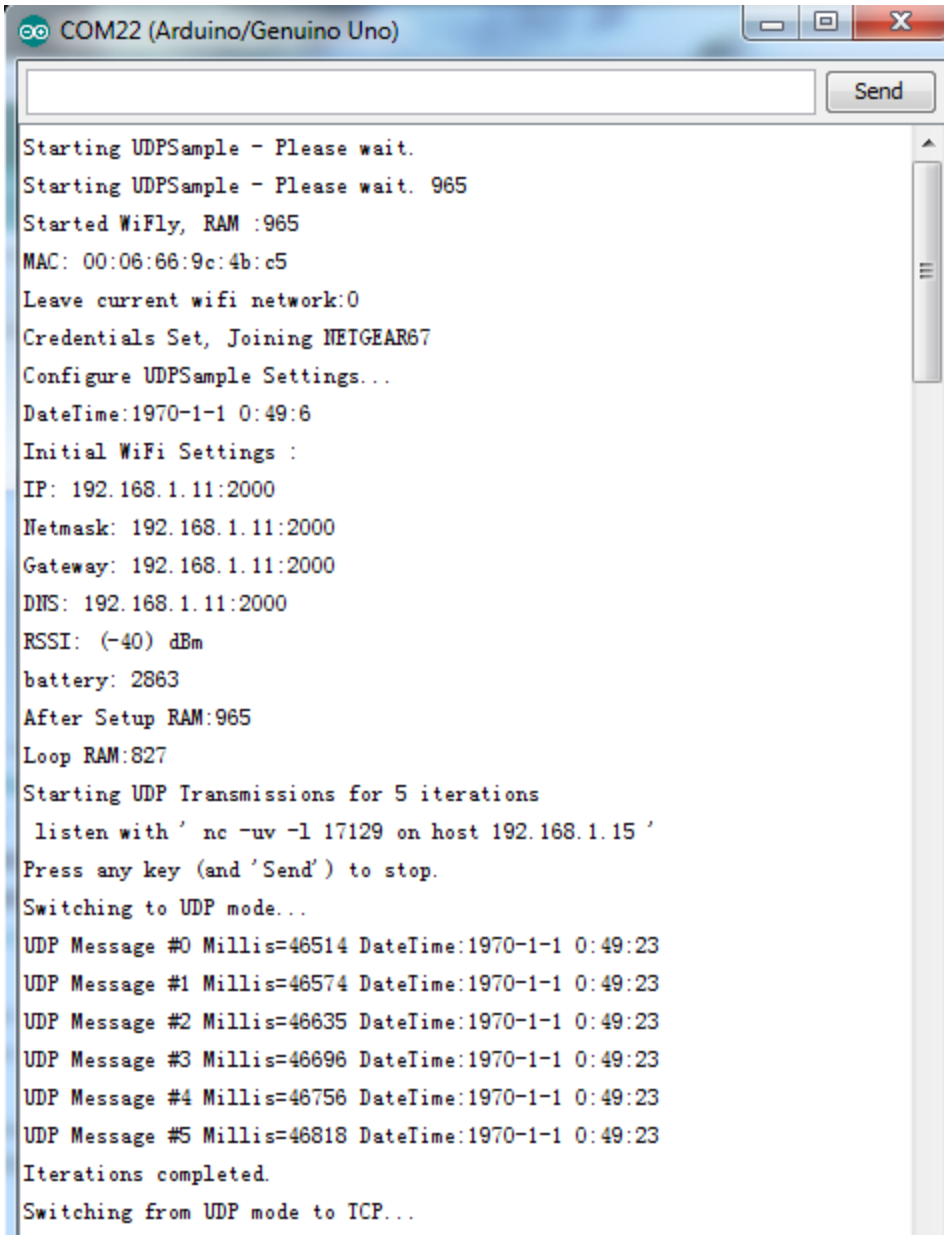

Fig. 10. UDPSample example, Arduino is now running as a UDP sender.

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

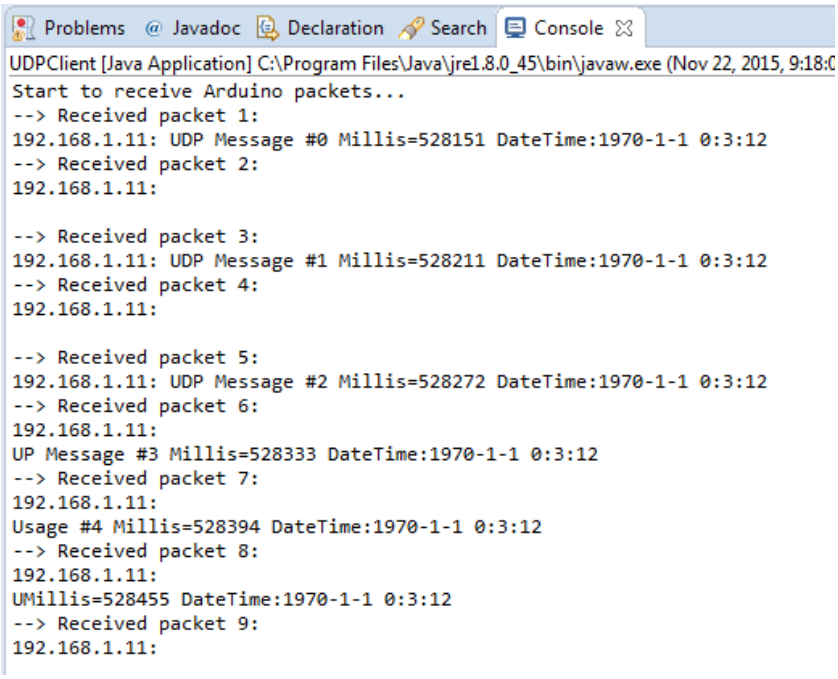

Fig. 11. UDPSample example, message received in the UDPClient.

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