Note: These tutorials are for how you set up the non-coding portion of the project and are written as if you are attempting to run my code or have written something similar. If you have a question about coding, my code in the github repository is pretty thoroughly commented or you could check out my list of references.

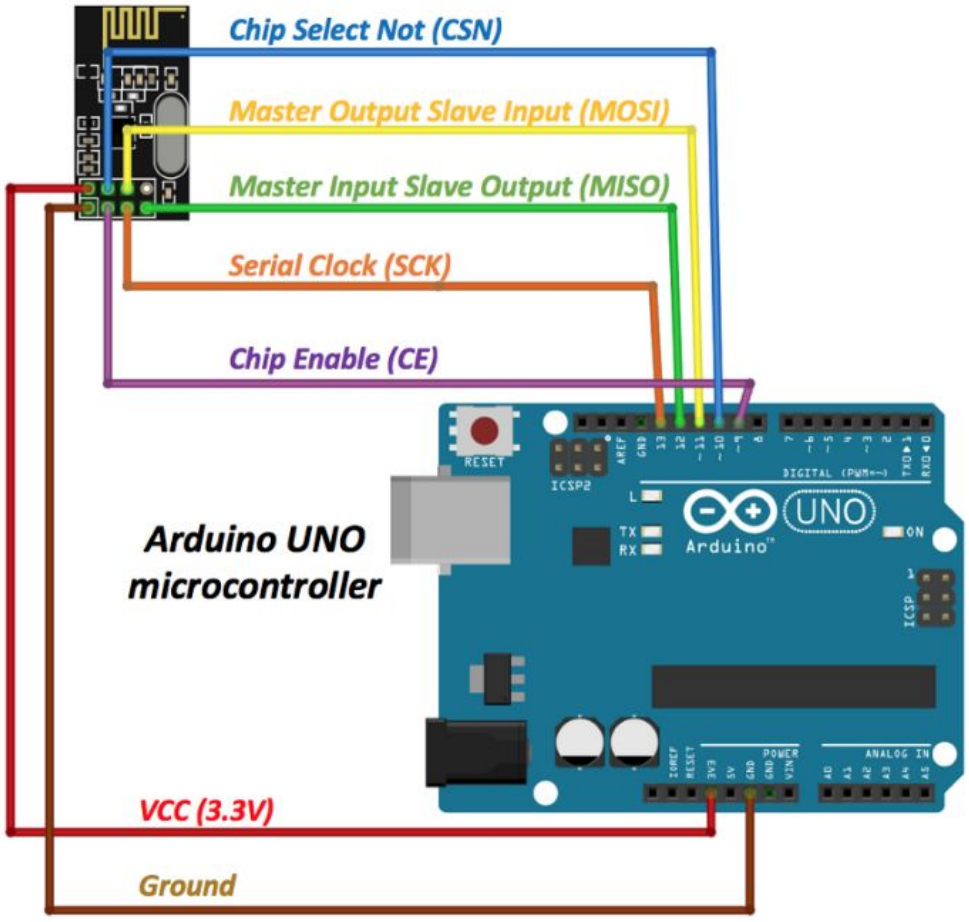
Setting up the Server(Arduino Ethernet Shield):

1. Place the Arduino Ethernet Shield on top of the Arduino, while making sure that all of the pins are actually socketed into the arduino. If not careful, they could become bent out of place instead, which would leave that port inoperable.
2. Open Windows Task Manager on a computer connected to the network you are planning on hooking the arduino up to.
3. Click more details.
4. Go to the performance tab.
5. Look at ipv4 under the wireless or ethernet section.
6. Record the first 3 numbers listed (Ex. 172 25 55).
7. Open up the project code
8. Input the three recorded numbers on the two lines IPAddress ip(); and IPAddress gateway(); each separated by a comma. (This address is specific to each network and is required for a device to be recognized by the network)
9. Insert a random number between 1 and 999 as a fourth number on these lines, also separated by a comma. (This will function as the device's specific address on the network. If the number is already taken, another will be assigned by the network. Whether your inputted number is used or not does not affect functionality.)
10. Plug the Arduino into the network using an ethernet cable
11. Open the Arduino's serial monitor.
12. Run the code.
13. The ip address for the arduino will appear on the serial monitor. (Ex. 172.25.55.66)
14. Record this address.
15. Open Windows Control Panel on the computer you wish to input commands to the arduino with.
16. Click the programs tab.
17. Click "turn windows features on and off." (You need administrator privileges to perform this step)
18. Check the box labeled "Telnet Client."
19. Open the windows console or "command window." on this computer
20. Enter "telnet 172.25.55.66" into the console and hit the enter key (replace the example ip address with your own)
21. The screen should go blank with only your cursor at the top.
22. Hit any key.
23. A few lines stating what commands you can perform are displayed.
24. Enter one of the commands shown to send a command to the transmitter Arduino.

25. Any other inputs besides those listed above will return the list of commands that can be performed.
26. The connection should last until you exit the control window or disconnect the arduino.

Transmitter Arduino and Receiver Arduino(RF24 Chips):

1. Use Male/Female wires to attach the transceiver chip to the Arduino as shown below.



https://medium.com/@benjamindavidfraser/arduino-nrf24l01-communications-947e1acb33fb

2. Remember that if you cheaped out on the transceiver chips, you'll need to place a 100 mf capacitor across the ground and 3.3 volts. This means that you'll need to use a breadboard as well since you'll need to put the ground and 3.3 volts into the board to be able to place the capacitor across them.
3. Import the RF24 library from my repository or from the repository at this link: https://github.com/nRF24/RF24

Connecting the Server Arduino and Transmitter Arduino:

Due to the overlapping digital input and output pins on the Arduino Ethernet Shield and the RF24 Transceiver Chips, they can't be hooked up to the same Arduino. However, any of the open digital pins can be used to send a high or low signal between the Arduinos. They can just simply be plugged in from port to port. For example, digital pin 5 on the Server Arduino can be wired straight into digital pin 5 on the Transmitter Arduino. For this project, I only had 2 commands: "on" and "off," so I only needed one wire because it could send a high or low signal depending on which command was being sent. However, if you wanted more commands, you could simply increase the amount of wires. It sounds inefficient, but if you had 4 wires, you could use them like a 4 bit decoder. This would allow you to have 16 different commands due to the different high and low combinations being sent between Arduinos.