ST SMART

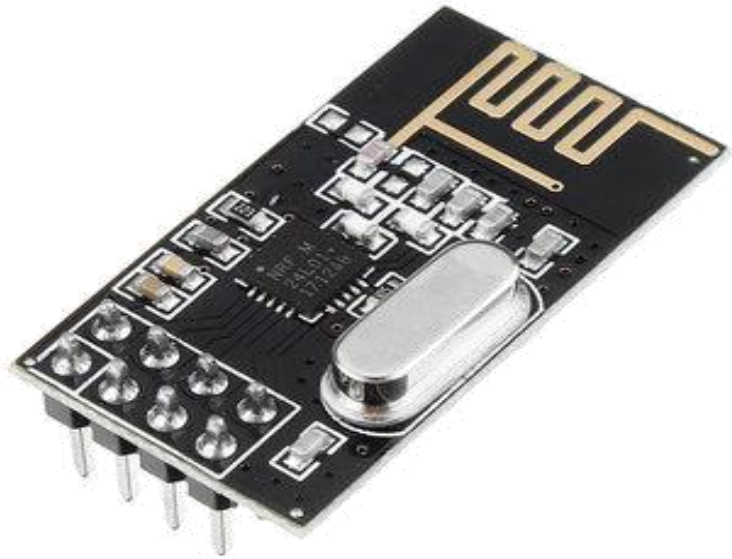
MAKE EVERYTHING SMART

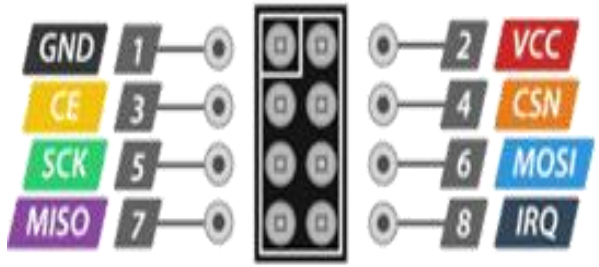
SMART TECHNOLOGT

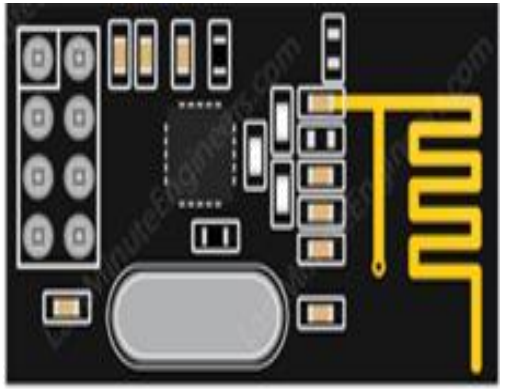
LECTURE 8

• nRF 24L01

Frequency Range	2.4 GHz ISM Band
Maximum Air Data Rate	2 Mb/s
Modulation Format	GFSK
Max. Output Power	0 dBm
Operating Supply Voltage	1.9 V to 3.6 V
Max. Operating Current	13.5mA
Min. Current(Standby Mode)	26µA
Logic Inputs	5V Tolerant
Communication Range	800+ m (line of sight)

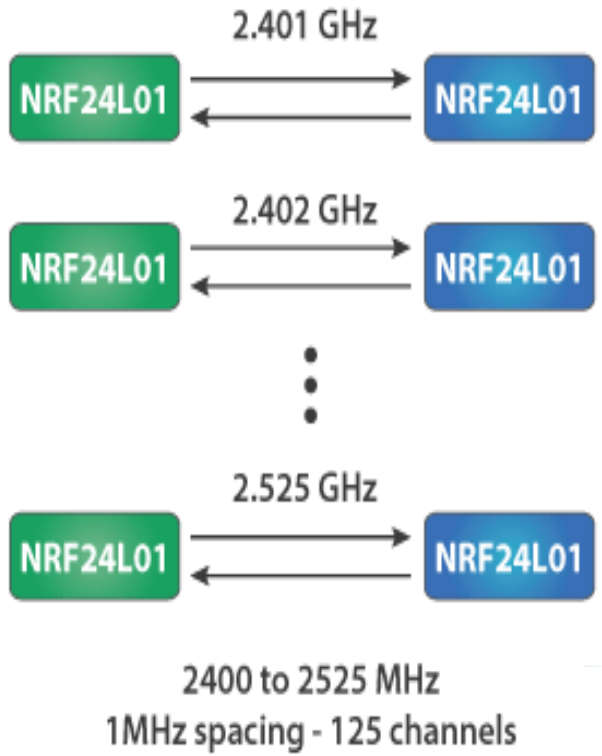


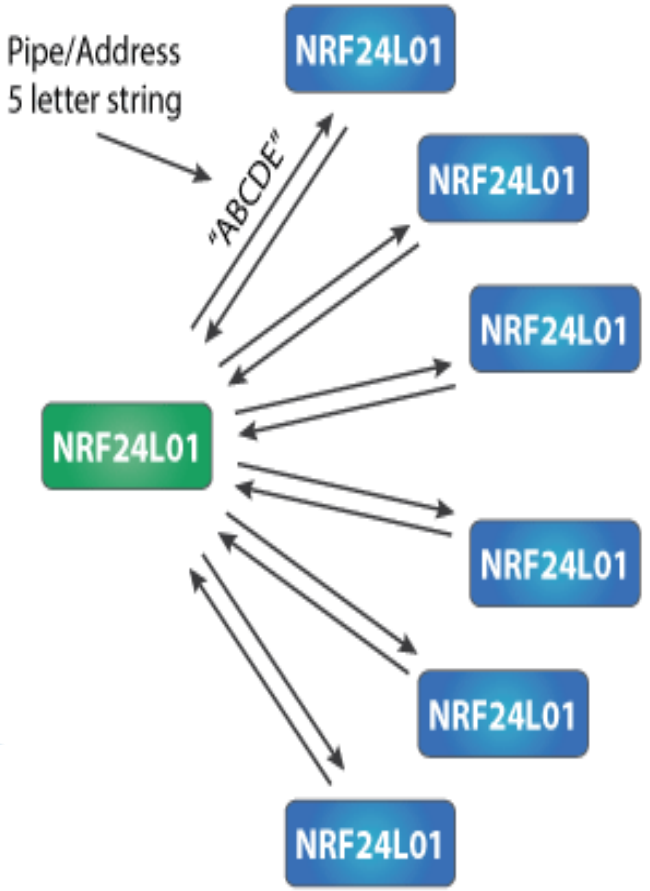




• nRF 24L01

- The module can use 125 different channels which gives a possibility to have a network of 125 independently working modems in one place. Each channel can have up to 6 addresses, or each unit

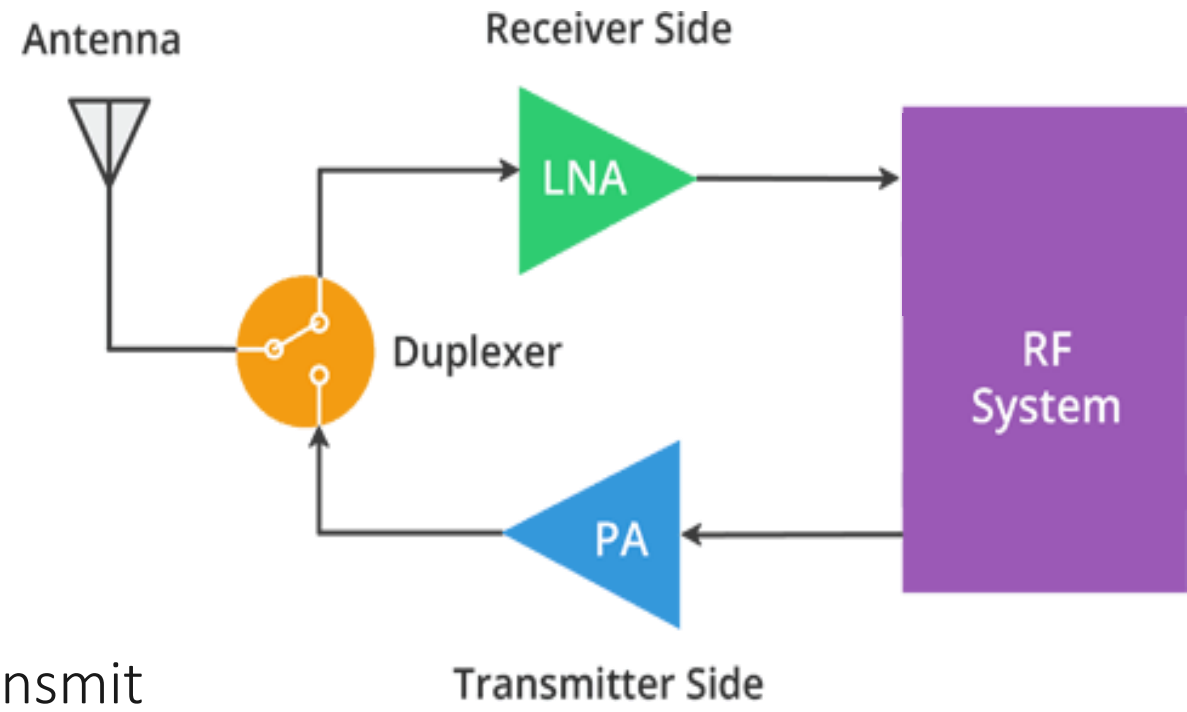




• PA and LNA?

What is PA and LNA?

PA stands for Power Amplifier. It only amplifies the signal strength being transmitted from the nRF24L01+ chip. Whereas LNA stands for Low-Noise Amplifier whose function is to take an extremely weak signal from the antenna (usually below microvolts or -100 dBm) and amplify it to a more useful level (usually around 0.5 to 1 V).

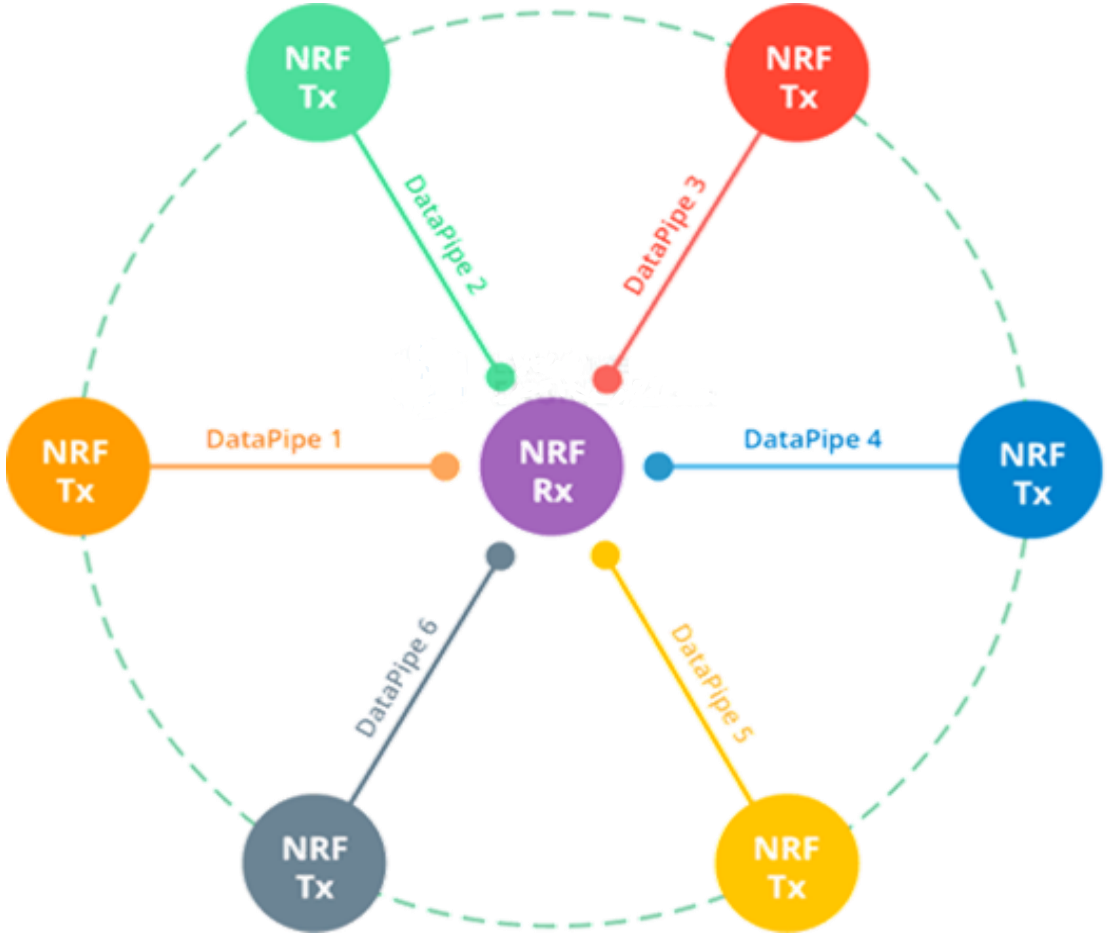


The (LNA) of the receive path and the (PA) of the transmit path connect to the antenna through a duplexer, which isolates the two signals and prevents the relatively powerful PA output from overloading the sensitive LNA input.

• PA and LNA?

The nRF24L01+ has a feature called Multiceiver. It is an abbreviation for Multiple Transmitter Single Receiver

In a multiceiver network each RF channel is logically divided into 6 parallel data channels called data pipes. In other words, the data pipe is one of six logical channels in a single physical RF channel. Each data pipe has its own unique address called a data pipe address. Only one data pipe can receive a packet at a time.



• nRF 24L01

there are three scenarios of how the two nRF24L01 modules interact with each other.

1. Transaction with an acknowledgment:

the transmitter initiates the communication by sending a data packet to the receiver. Once the packet is transmitted, it waits approximately 130 µs to receive the acknowledgment packet (ACK packet). When the receiver receives the packet it sends the ACK packet to the transmitter. The transaction ends when the transmitter receives the ACK packet.

• nRF 24L01

there are three scenarios of how the two nRF24L01 modules interact with each other.

2. Transaction with a lost data packet:

retransmission is required due to the loss of the transmitted packet. After the packet is transmitted, the transmitter waits to receive the ACK packet.

If the transmitter does not receive it within the auto-retransmit-delay (ARD) time it retransmits the packet. When the receiver receives the retransmitted packet, it transmits the ACK packet thereby terminating the transaction.

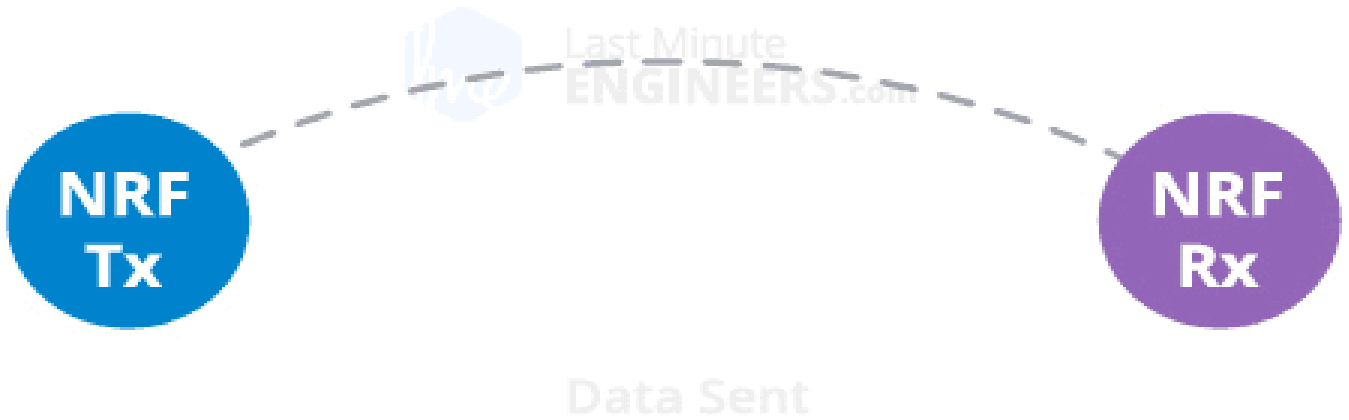
• nRF 24L01

there are three scenarios of how the two nRF24L01 modules interact with each other.

3. Transaction with a lost acknowledgment:

retransmission is required due to loss of ACK packets. Here even though the receiver has received the packet in the first attempt itself, due to the loss of the ACK packet, the transmitter thinks that the receiver has not received the packet.

So after the Auto-Retransmit-Delay timeout, the transmitter retransmits the packet. Now when the receiver receives the packet with the same ID as before, it discards it and sends the ACK packet again. The transaction ends when the transmitter receives the ACK packet.

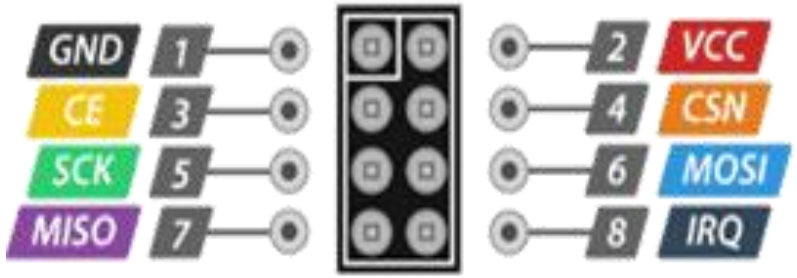


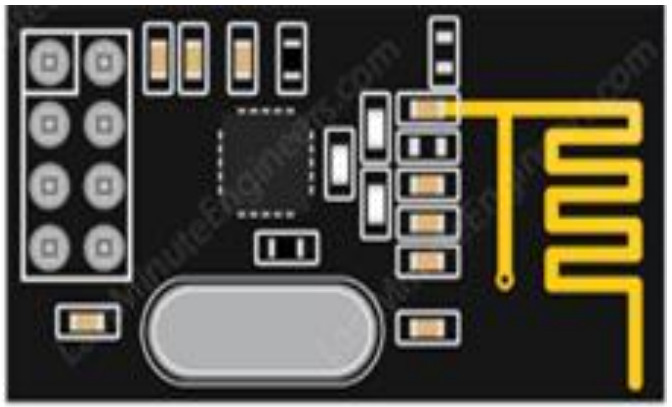
CE (Chip Enable) is an active-high pin. When selected, the nRF24L01 will either transmit or receive, depending on which mode it is currently in.

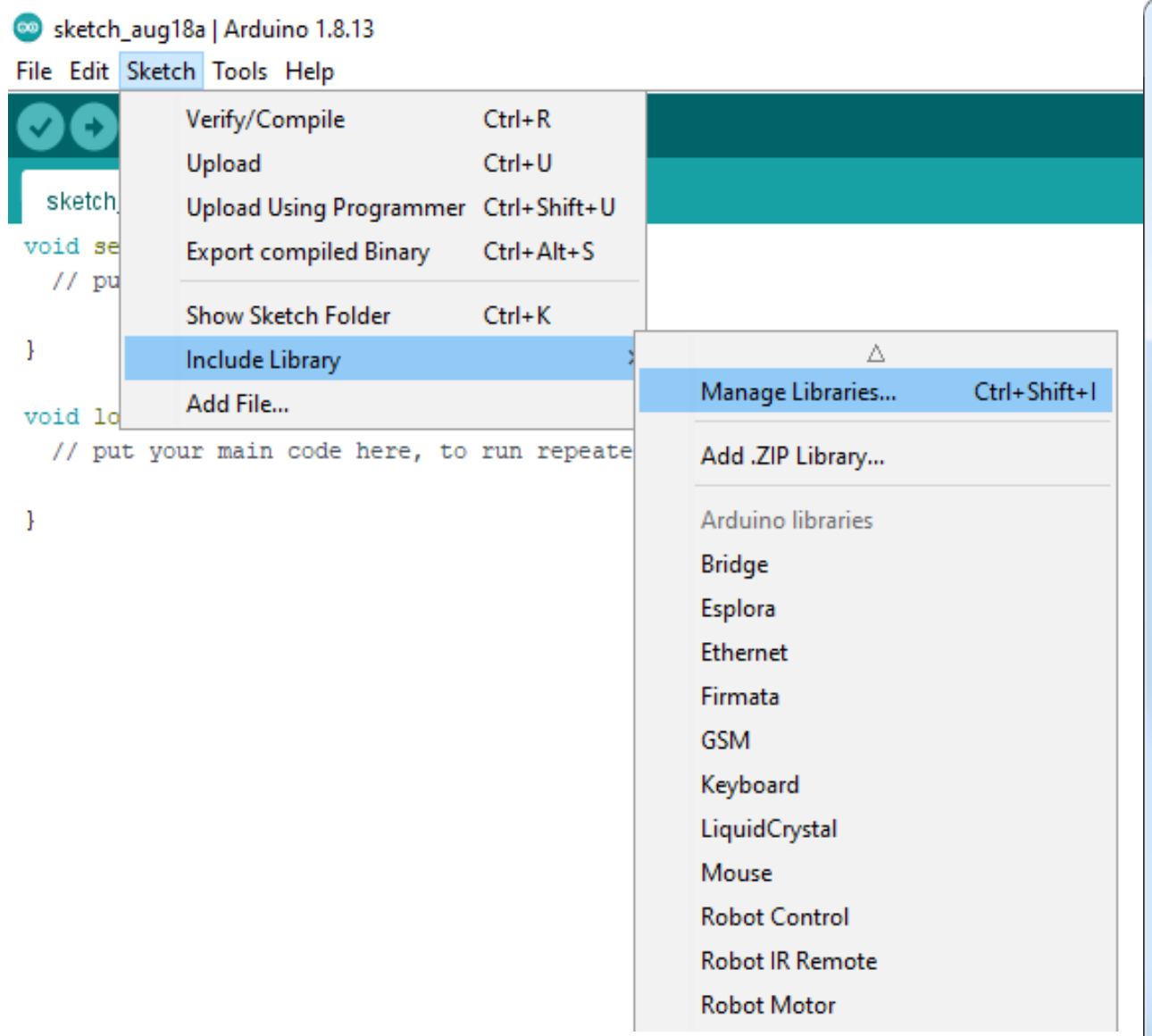
CSN (Chip Select Not) is an active-low pin and is normally kept HIGH. When this pin goes low, the nRF24L01 starts listening on its SPI port for data and processes it accordingly.

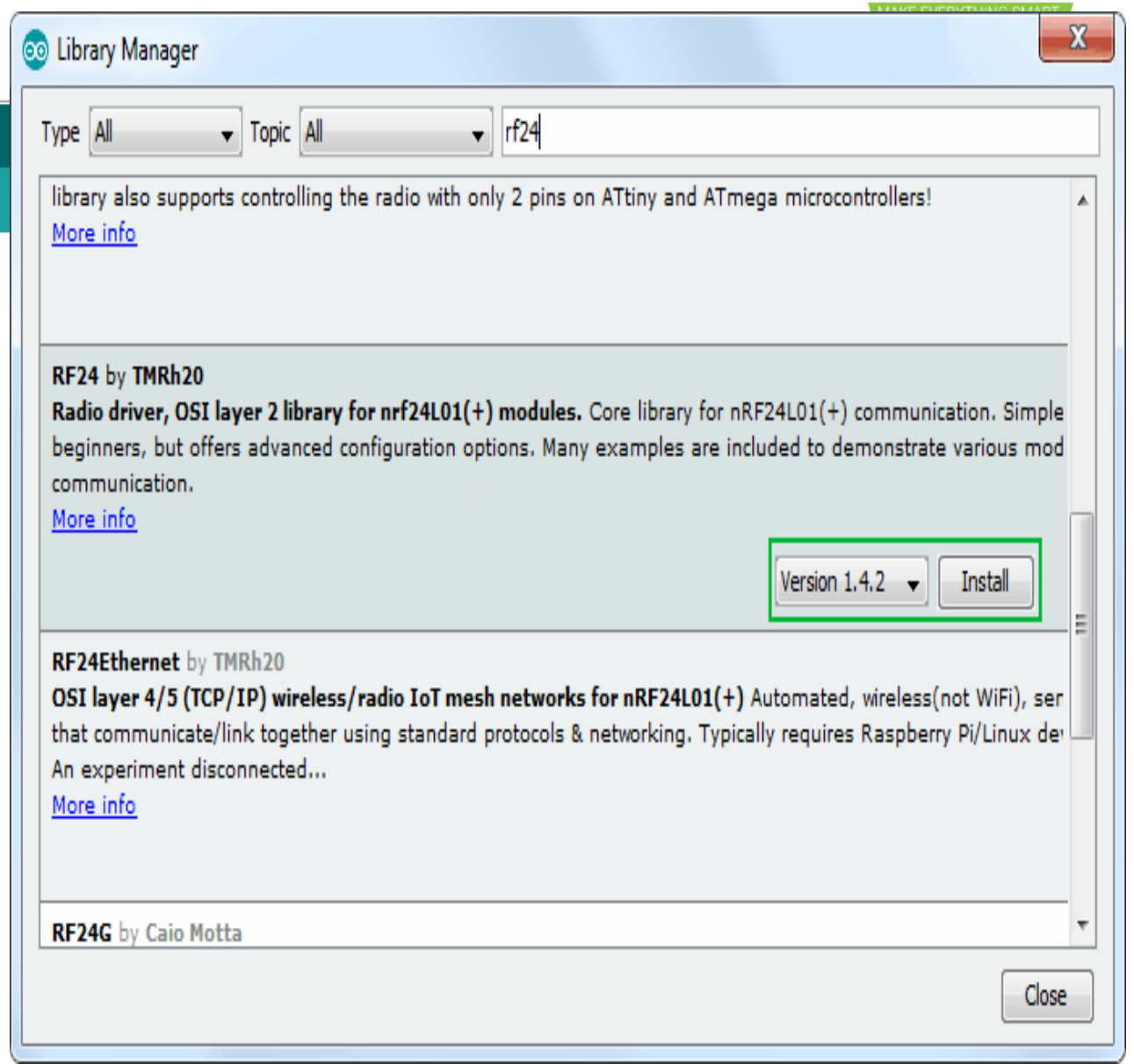
Arduino SPI pins

Arduino	SCK	MISO	MOSI	SS
Uno	13	12	11	10
Nano	13	12	11	10
Mega	52	50	51	53






sketch_aug18a | Arduino 1.8.13
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
Esplora
Ethernet
Firmata
GSM
Keyboard
LiquidCrystal
Mouse
Robot Control
Robot IR Remote
Robot Motor
void se
// pu
}
void lo
// put your main code here, to run repeate
}


Library Manager
Type All Topic All rf24
library also supports controlling the radio with only 2 pins on ATtiny and ATmega microcontrollers!
More info
RF24 by TMRh20
Radio driver, OSI layer 2 library for nrf24L01(+) modules. Core library for nRF24L01(+) communication. Simple
beginners, but offers advanced configuration options. Many examples are included to demonstrate various mod
communication.
More info
Version 1.4.2
Install
RF24Ethernet by TMRh20
OSI layer 4/5 (TCP/IP) wireless/radio IoT mesh networks for nRF24L01(+) Automated, wireless(not WiFi), ser
that communicate/link together using standard protocols & networking. Typically requires Raspberry Pi/Linux de
An experiment disconnected...
More info
RF24G by Caio Motta
Close

• Inspection Code

If your serial output has these values then Your nrf module is in working condition :

```
  EN_AA          = 0x3f
  EN_RXADDR      = 0x02
  RF_CH          = 0x4c
  RF_SETUP       = 0x03
  CONFIG         = 0x0f
```

```
#include <SPI.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 8);   //  CE, CSN pins  & create an RF24 object

byte addresses[][6] = {"1Node", "2Node"};

void setup() {
  radio.begin(); //begin operation of the chip

  radio.setPALevel(RF24_PA_LOW); //power gain “higher amplitude longer distance
```

Set Power Amplifier (PA) level and Low Noise Amplifier (LNA) state

(decibel-milliwatts) dBm is a unit of level used to indicate that a power level is expressed in decibels (dB) with reference to one milliwatt (mW). It is used in radio, microwave and fiber-optical communication networks

level (enum value)	nRF24L01 description
RF24_PA_MIN (0)	-18 dBm
RF24_PA_LOW (1)	-12 dBm
RF24_PA_HIGH (2)	-6 dBm
RF24_PA_MAX (3)	0 dBm

```
radio.openWritingPipe(addresses[0]);
```

Open a pipe for writing via byte array. Old addressing format retained for compatibility.
Only one writing pipe can be opened at once

```
radio.openReadingPipe(1, addresses[1]);
```

Which pipe to open. Only pipe numbers 0-5 are available

address of the pipe to open.

Open a pipe for reading
Up to 6 pipes can be open for reading at once. Open all the required reading pipes, and then call startListening().

• Inspection Code

```
radio.startListening(); //Start listening on the pipes opened for reading.

Serial.begin(9600);
  printf_begin();
  radio.printDetails(); //does nothing if printf is not defined “work with
#include <printf.h>

}

  void loop() {
  //  empty

  }
```

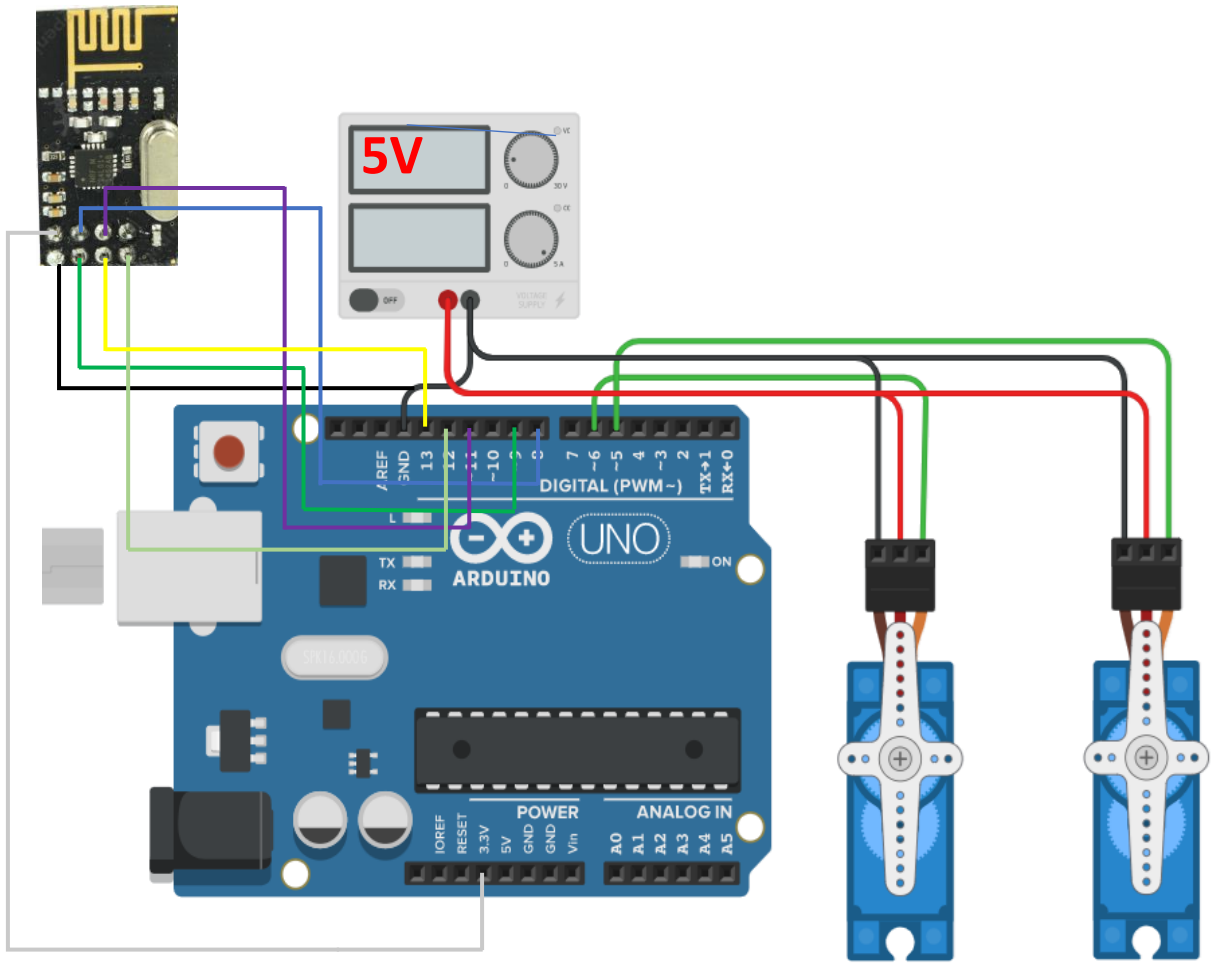
• Transmitter code

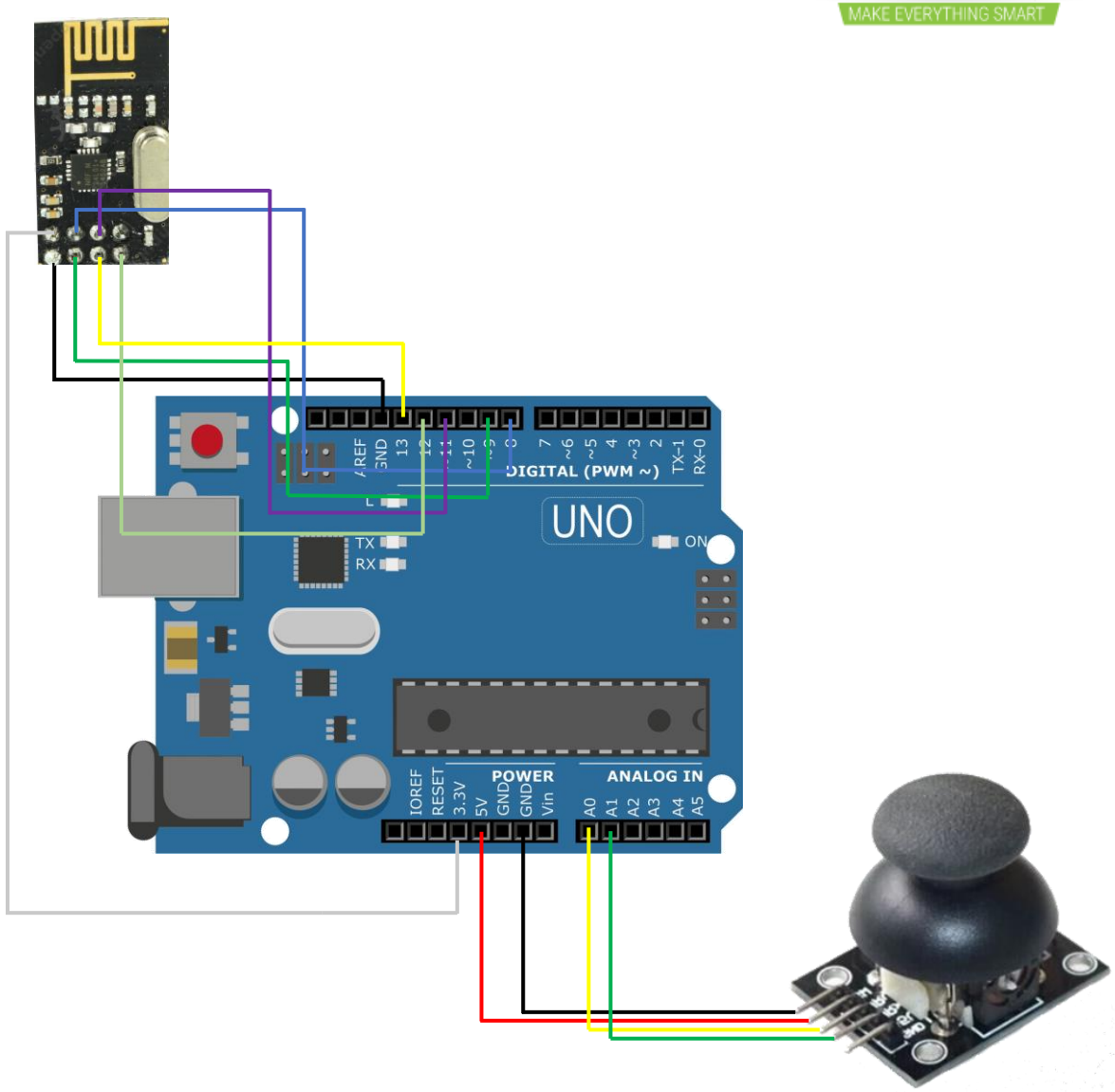
```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8); // CE, CSN
const byte address[6] = "00001"; //address through which two modules communicate.
void setup() {
  radio.begin();
  radio.openWritingPipe(address); // set the transmitter address
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening(); //Set module as transmitter
}
void loop() {
  const char text[] = "ST Smart";
  radio.write(&text, sizeof(text));
  delay(1000);}
```

Note that you can send messages up to 32 bytes at a time, as this is the maximum size of a packet that the nRF24L01+ can handle.
If you need confirmation that the receiver has received the data, the method write() returns a bool value. It returns TRUE when the data reaches the receiver or it returns FALSE if the data is lost

• Receiver code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8);
const byte address[6] = "00001";//address through which two modules communicate.
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address); //set the address
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();  //Set module as receiver
}
void loop() {
  if (radio.available()) { //Read the data if available in buffer
    char text[32] = "";
    radio.read(&text, sizeof(text));
    Serial.println(text);
  }}
```


5V
DIGITAL (PWM~)
UNO
ARDUINO
TX
RX
ON
POWER
ANALOG IN


DIGITAL (PWM ~)
UNO
TX
RX
ON
POWER
ANALOG IN

• Transmitter

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8);
int joystick[2];
const byte address[6] = "00001";
void setup() {
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}
void loop() {
  joystick[0]=analogRead(A0);
  joystick[1]=analogRead(A1);
  joystick[0]=map(joystick[0],0,1023,0,180);
  joystick[1]=map(joystick[1],0,1023,0,180);
  radio.write(joystick,sizeof(joystick));}
```

• Receiver

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>
RF24 radio(9, 8);

Servo myservo1,myservo2;

const byte address[6] = "00001";
int joystick[2]={90,90};
void setup() {
Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
  myservo1.attach(6);
  myservo2.attach(5);}
```

```
void loop() {
if ( radio.available()) {
  while (radio.available()) {
   radio.read(joystick,sizeof(joystick));
      Serial.print("VRX= ");
      Serial.print(joystick[0]);
      Serial.print("    VRY= ");
      Serial.println(joystick[1]);
      myservo1.write(joystick[0]);
      myservo2.write(joystick[1]);
}
}
}
```

THANKS FOR COMING