Connection:

Arduino NRF24L01

GND - GND

3.3V - VCC

9 - CE

10 - CSN

13 - CLK

MOSI - 11

MISO - 12

I did not connect the IRQ pin and it was fine.

Download the Library from Sketch-Include Library and search from RF24 and then selection RF24 library provided by tmhr20

<https://github.com/TMRh20>

Checking if there is problem in connection between Arduino and NRF24L01 - it can done using the following program.

// 18 Mar 2018 - simple program to verify connection between Arduino

// and nRF24L01+

// This program does NOT attempt any communication with another nRF24

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <printf.h>

#define CE\_PIN 9

#define CSN\_PIN 10

const byte thisSlaveAddress[5] = {'R','x','A','A','A'};

RF24 radio(CE\_PIN, CSN\_PIN);

char dataReceived[10]; // this must match dataToSend in the TX

bool newData = false;

void setup() {

Serial.begin(9600);

printf\_begin();

Serial.println("CheckConnection Starting");

Serial.println();

Serial.println("FIRST WITH THE DEFAULT ADDRESSES after power on");

Serial.println(" Note that RF24 does NOT reset when Arduino resets - only when power is removed");

Serial.println(" If the numbers are mostly 0x00 or 0xff it means that the Arduino is not");

Serial.println(" communicating with the nRF24");

Serial.println();

radio.begin();

radio.printDetails();

Serial.println();

Serial.println();

Serial.println("AND NOW WITH ADDRESS AAAxR 0x41 41 41 78 52 ON P1");

Serial.println(" and 250KBPS data rate");

Serial.println();

radio.openReadingPipe(1, thisSlaveAddress);

radio.setDataRate( RF24\_250KBPS );

radio.printDetails();

Serial.println();

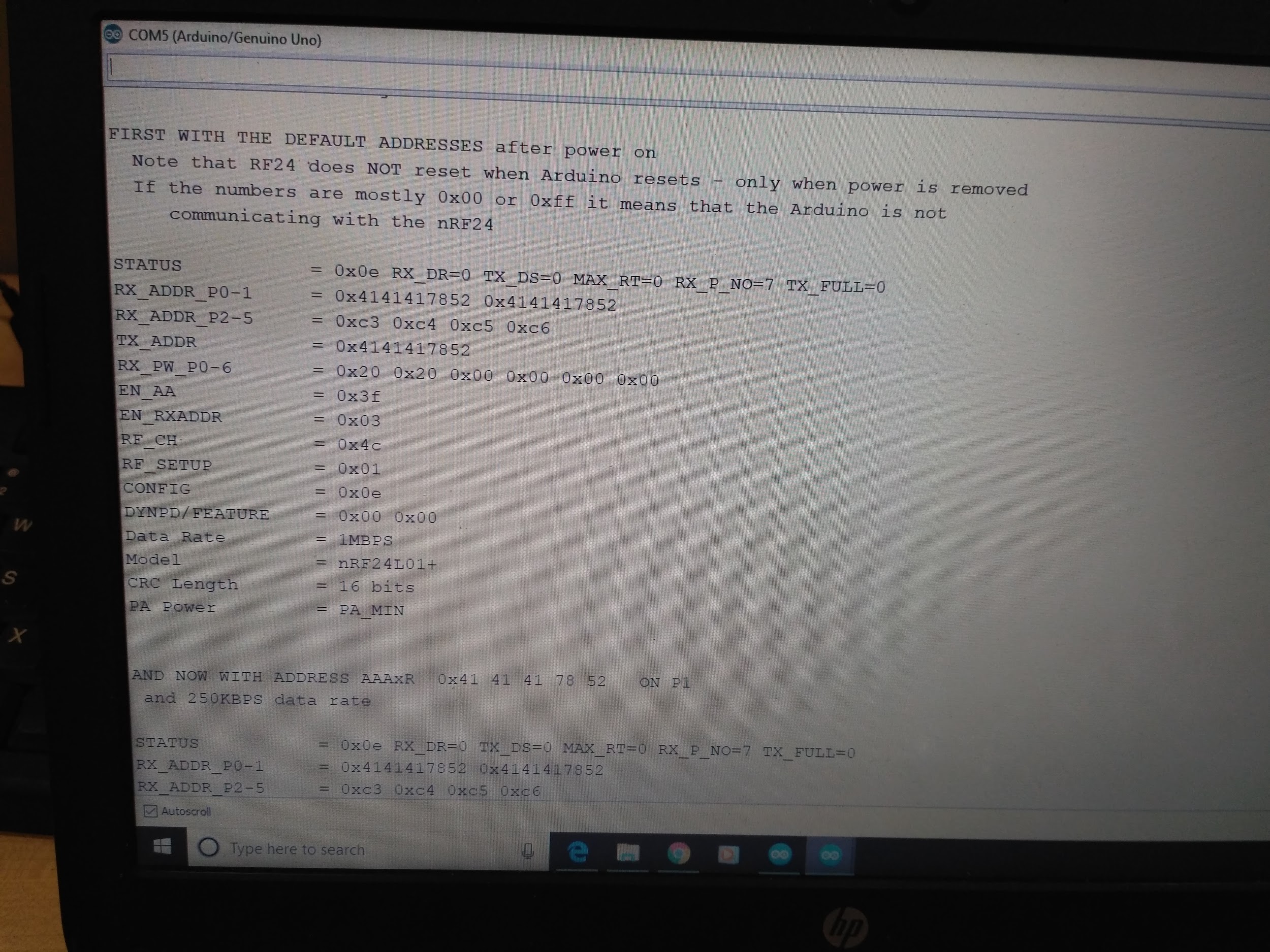
Serial.println();

}

void loop() {

}

When I ran the above program with some NRF24L01 addresses like 0x00 and 0xFF was getting printed. But when I ran the above program on Arduino with some other NRF24L01 module it was printing different addresses, data rate etc as follows



After checking connection I ran the following program which transmits and receives some data. It was working fine.

**Arduino to Arduino via NRF24L01**

**Transmitter**

// SimpleTx - the master or the transmitter

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#define CE\_PIN 9

#define CSN\_PIN 10

const byte slaveAddress[5] = {'R','x','A','A','A'};

RF24 radio(CE\_PIN, CSN\_PIN); // Create a Radio

char dataToSend[10] = "Message 0";

char txNum = '0';

unsigned long currentMillis;

unsigned long prevMillis;

unsigned long txIntervalMillis = 1000; // send once per second

void setup() {

Serial.begin(9600);

Serial.println("SimpleTx Starting");

radio.begin();

radio.setDataRate( RF24\_250KBPS );

radio.setRetries(3,5); // delay, count

radio.openWritingPipe(slaveAddress);

}

//====================

void loop() {

currentMillis = millis();

if (currentMillis - prevMillis >= txIntervalMillis) {

send();

prevMillis = millis();

}

}

//====================

void send() {

bool rslt;

rslt = radio.write( &dataToSend, sizeof(dataToSend) );

// Always use sizeof() as it gives the size as the number of bytes.

// For example if dataToSend was an int sizeof() would correctly return 2

Serial.print("Data Sent ");

Serial.print(dataToSend);

if (rslt) {

Serial.println(" Acknowledge received");

updateMessage();

}

else {

Serial.println(" Tx failed");

}

}

//================

void updateMessage() {

// so you can see that new data is being sent

txNum += 1;

if (txNum > '9') {

txNum = '0';

}

dataToSend[8] = txNum;

}

**Receiver**

// SimpleRx - the slave or the receiver

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#define CE\_PIN 9

#define CSN\_PIN 10

const byte thisSlaveAddress[5] = {'R','x','A','A','A'};

RF24 radio(CE\_PIN, CSN\_PIN);

char dataReceived[10]; // this must match dataToSend in the TX

bool newData = false;

//===========

void setup() {

Serial.begin(9600);

Serial.println("SimpleRx Starting");

radio.begin();

radio.setDataRate( RF24\_250KBPS );

radio.openReadingPipe(1, thisSlaveAddress);

radio.startListening();

}

//=============

void loop() {

getData();

showData();

}

//==============

void getData() {

if ( radio.available() ) {

radio.read( &dataReceived, sizeof(dataReceived) );

newData = true;

}

}

void showData() {

if (newData == true) {

Serial.print("Data received ");

Serial.println(dataReceived);

newData = false;

}

}

**Arduino to Raspberry Pi Communication via NRF24L01**

**Simple Transmitter**

#include<SPI.h>

#include<RF24.h>

// ce, csn pins

RF24 radio(9, 10);

void setup(void){

radio.begin();

radio.setPALevel(RF24\_PA\_MAX);

//radio.setDataRate( RF24\_1MBPS );

radio.setChannel(0x76); //It was creating problem while communicating with Raspberry Pi

radio.openWritingPipe(0xF0F0F0F0E1);//Previously the address was 0x65646f4e31

radio.enableDynamicPayloads(); //It was creating problem while communicating with Raspberry Pi

radio.powerUp();

}

void loop(void){

const char text[] = "Hello World is awesome";

radio.write(&text, sizeof(text));

delay(1000);

}

**Transmitter with Serial Print and Acknowledgement**

// SimpleTx - the master or the transmitter

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#define CE\_PIN 9

#define CSN\_PIN 10

const byte slaveAddress[5] = {'R','x','A','A','A'};

RF24 radio(CE\_PIN, CSN\_PIN); // Create a Radio

char dataToSend[10] = "Message 0";

char txNum = '0';

unsigned long currentMillis;

unsigned long prevMillis;

unsigned long txIntervalMillis = 1000; // send once per second

void setup() {

Serial.begin(9600);

Serial.println("SimpleTx Starting");

radio.begin();

radio.setDataRate(RF24\_1MBPS);

radio.setChannel(0x76);//Originally not present in this program

radio.setRetries(3,5); // delay, count

radio.openWritingPipe(0xF0F0F0F0E1);//Working with Node RED address 0x65646f4e31 //originally slave address

radio.enableDynamicPayloads();//Originally not present in this program

}

//====================

void loop() {

currentMillis = millis();

if (currentMillis - prevMillis >= txIntervalMillis) {

send();

prevMillis = millis();

}

}

//====================

void send() {

bool rslt;

rslt = radio.write( &dataToSend, sizeof(dataToSend) );

// Always use sizeof() as it gives the size as the number of bytes.

// For example if dataToSend was an int sizeof() would correctly return 2

Serial.print("Data Sent ");

Serial.print(dataToSend);

if (rslt) {

Serial.println(" Acknowledge received");

updateMessage();

}

else {

Serial.println(" Tx failed");

}

}

//================

void updateMessage() {

// so you can see that new data is being sent

txNum += 1;

if (txNum > '9') {

txNum = '0';

}

dataToSend[8] = txNum;

}

**Receiver**

import RPi.GPIO as GPIO  
from lib\_nrf24 import NRF24  
import time  
import spidev  
  
GPIO.setmode(GPIO.BCM)  
  
pipes = [[0xE8, 0xE8, 0xF0, 0xF0, 0xE1], [0xF0, 0xF0, 0xF0, 0xF0, 0xE1]]  
  
radio = NRF24(GPIO, spidev.SpiDev())  
radio.begin(0, 17)  
  
radio.setPayloadSize(32)  
radio.setChannel(0x76)  
radio.setDataRate(NRF24.BR\_1MBPS)  
radio.setPALevel(NRF24.PA\_MIN)  
  
radio.setAutoAck(True)  
radio.enableDynamicPayloads()  
radio.enableAckPayload()  
  
radio.openReadingPipe(1, pipes[1])  
radio.printDetails()  
radio.startListening()  
  
while(1):  
 # ackPL = [1]  
 while not radio.available(0):  
 time.sleep(1 / 100)  
 receivedMessage = []  
 radio.read(receivedMessage, radio.getDynamicPayloadSize())  
 print("Received: {}".format(receivedMessage))  
  
 print("Translating the receivedMessage into unicode characters")  
 string = ""  
 for n in receivedMessage:  
 # Decode into standard unicode set  
 if (n >= 32 and n <= 126):  
 string += chr(n)  
 print("Out received message decodes to: {}".format(string))

**Output for Simple Transmitter**

STATUS = 0x0e RX\_DR=0 TX\_DS=0 MAX\_RT=0 RX\_P\_NO=7 TX\_FULL=0  
RX\_ADDR\_P0-1 = 0xe7e7e7e7e7 0xf0f0f0f0e1  
RX\_ADDR\_P2-5 = 0xc3 0xc4 0xc5 0xc6   
TX\_ADDR = 0xe7e7e7e7e7  
RX\_PW\_P0-6 = 0x00 0x20 0x00 0x00 0x00 0x00   
EN\_AA = 0x3f   
EN\_RXADDR = 0x02   
RF\_CH = 0x76   
RF\_SETUP = 0x01   
CONFIG = 0x0f   
DYNPD/FEATURE = 0x3f 0x06   
Data Rate = 1MBPS  
Model = nRF24l01   
CRC Length = 16 bits  
PA Power = PA\_MIN  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115, 32, 97, 119, 101, 115, 111, 109, 101, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Hello World is awesome  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115, 32, 97, 119, 101, 115, 111, 109, 101, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Hello World is awesome  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115, 32, 97, 119, 101, 115, 111, 109, 101, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Hello World is awesome  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115, 32, 97, 119, 101, 115, 111, 109, 101, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Hello World is awesome  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115, 32, 97, 119, 101, 115, 111, 109, 101, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Hello World is awesome  
Received: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 32, 105, 115,

**Output for Transmitter with serial print with acknowledgement**

STATUS = 0x0e RX\_DR=0 TX\_DS=0 MAX\_RT=0 RX\_P\_NO=7 TX\_FULL=0  
RX\_ADDR\_P0-1 = 0xe7e7e7e7e7 0xf0f0f0f0e1  
RX\_ADDR\_P2-5 = 0xc3 0xc4 0xc5 0xc6   
TX\_ADDR = 0xe7e7e7e7e7  
RX\_PW\_P0-6 = 0x00 0x20 0x00 0x00 0x00 0x00   
EN\_AA = 0x3f   
EN\_RXADDR = 0x02   
RF\_CH = 0x76   
RF\_SETUP = 0x01   
CONFIG = 0x0f   
DYNPD/FEATURE = 0x3f 0x06   
Data Rate = 1MBPS  
Model = nRF24l01   
CRC Length = 16 bits  
PA Power = PA\_MIN  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 50, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 2  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 51, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 3  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 52, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 4  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 53, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 5  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 54, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 6  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 55, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 7  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 56, 0]  
Translating the receivedMessage into unicode characters  
Out received message decodes to: Message 8  
Received: [77, 101, 115, 115, 97, 103, 101, 32, 57, 0]  
Translating the receivedMessage into unicode characters

Reference

<https://github.com/nRF24/RF24/issues/367>

<http://forum.arduino.cc/index.php?topic=421081.0> Example of Sample Transmission.