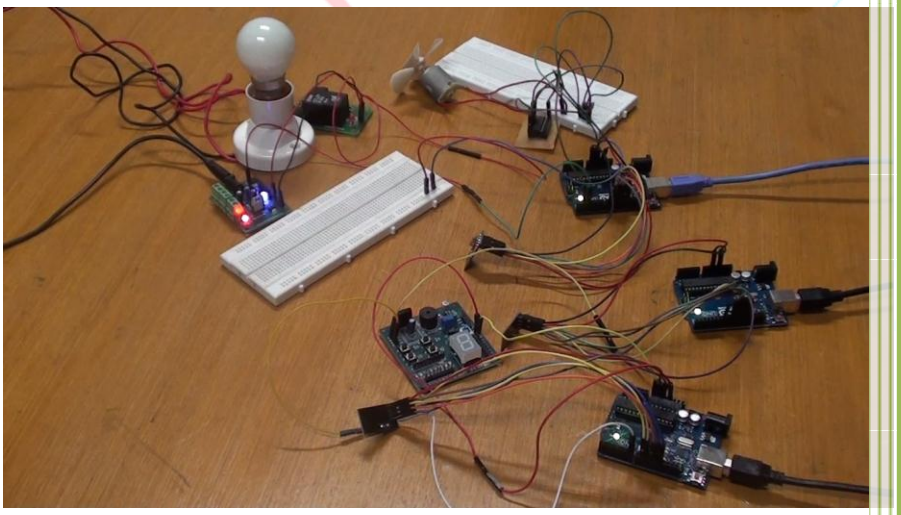
TENET TECHNETRONICS

2015

Interfacing NRF24L01 Transceiver with Arduino UNO



Author: Vivek g s

Transmitting data through wireless can be done by various hardware modules like XBEE, HC-05 Bluetooth module, RF ASK module etc. These modules operate on different frequency band and they are very much costlier, if you are looking out for a wireless transceiver device which uses ultralow power and with less cost and it can also transmit and receive data up to 1Km range NRF24L01 is the device.

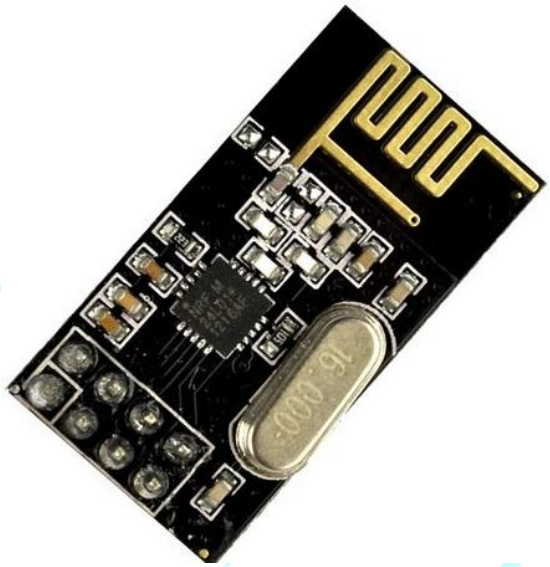


Figure 1 - nRF24L01 module

The nRF24L01+ is a single chip 2.4GHz transceiver with an Enhanced shock burst protocol embedded in it, which operates on a very low power (i.e. 3.3V). In this application note we are interfacing nRF24L01+ module with Arduino UNO to turn ON an LED, whenever the LDR value reaches certain limit on the transmitter end a LED on the receiver side will glow.

PIN Configuration:

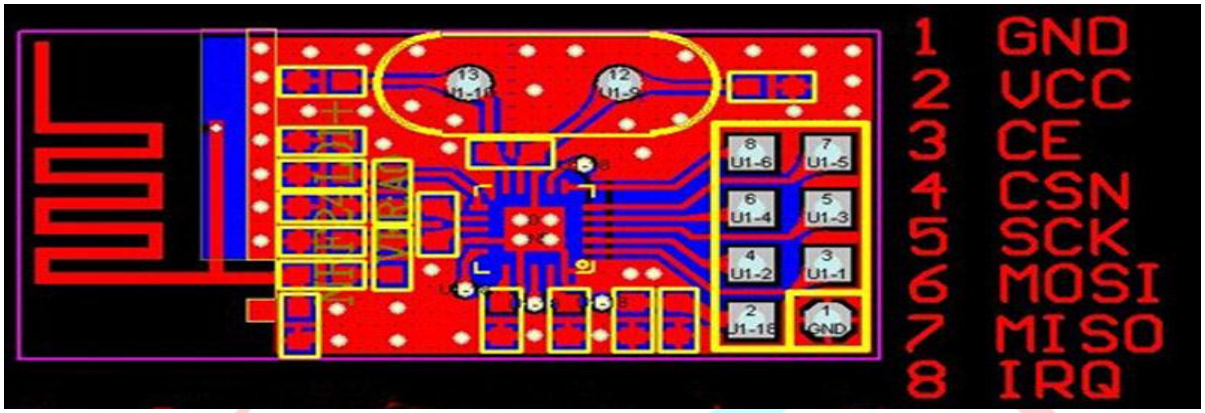


Figure 2 – nRF24L01 pin outs

Pin Description:

GND	Connects to System Ground
IRQ	Maskable interrupt pin. Active Low
MISO	SPI Slave Data Output
MOSI	SPI Slave Data Input
SCK	SPI Slave Data Input
CSN	SPI Chip Select
CE	Chip Enable Activates RX or TX mode. CE = 0 makes the chip to go into Stand-by
VCC	Connects to Power Supply (3.3V).

Pin connections:

Arduino UNO	**nRF24L01**
GND	GND
3.3V	VCC
Pin-9	CE
Pin-10	CSN
Pin-13	SCK
Pin-11	MOSI
Pin-12	MISO
No connection	IRQ

Interfacing nRF24L01+ module with Arduino UNO

Transmitter1 Block diagram:

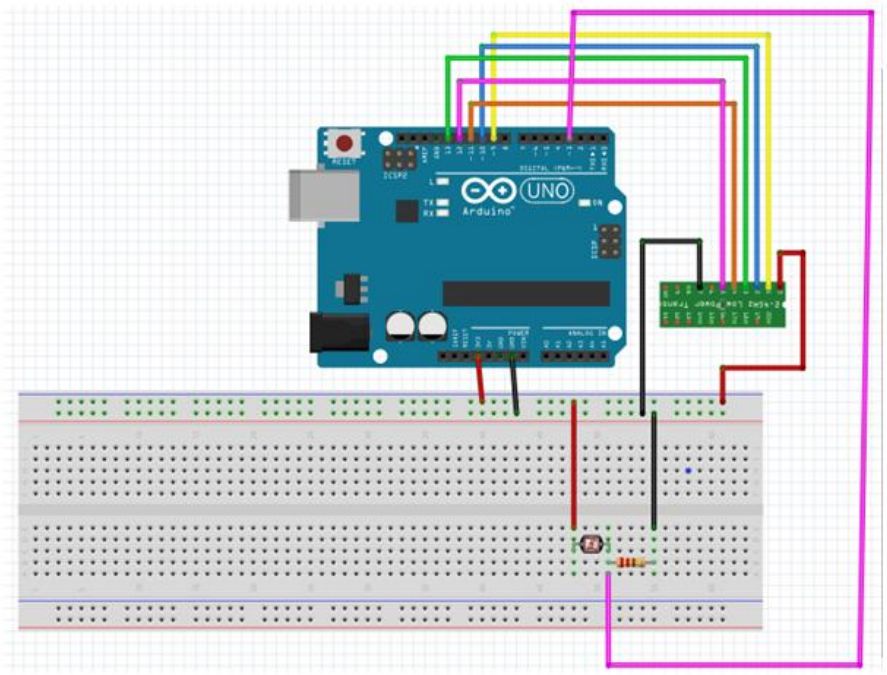


Figure 3 – Transmitter1 circuit diagram

Transmitter2 Block diagram:

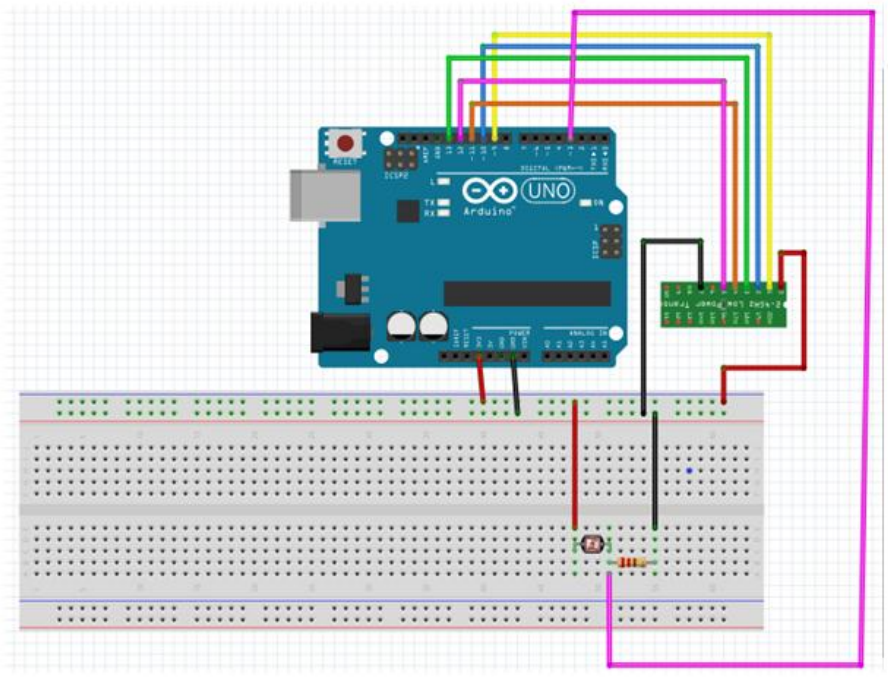


Figure 4 – Transmitter2 circuit diagram

Receiver:

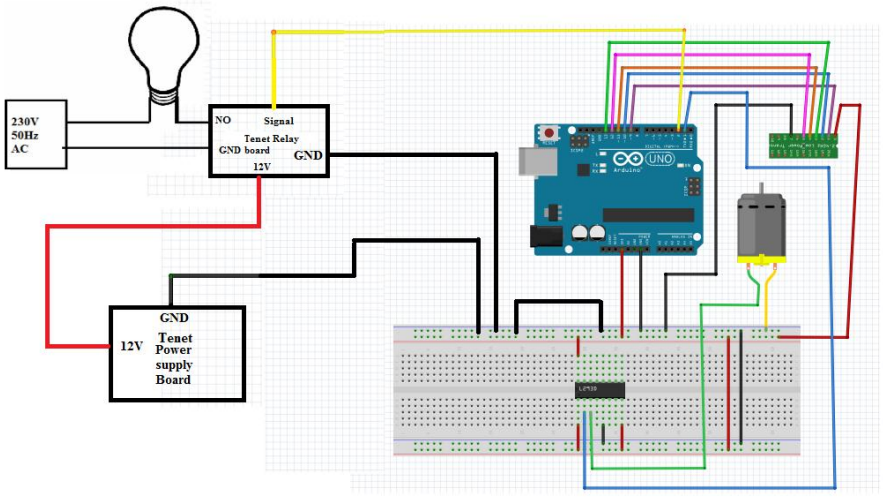


Figure 5 – Receiver circuit diagram

Code:

Transmitter1:

#include <SPI.h> //Call SPI library so you can communicate with the nRF24L01+

#include <nRF24L01.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/

#include <RF24.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/

const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or active mode (1)

```
const int pinCSN = 10; //This pin is used to tell the nRF24 whether the SPI communication is a command or message to send out

bool done = false; //used to know when to stop sending guesses

RF24 wirelessSPI(pinCE, pinCSN); // Create your nRF24 object or wireless SPI connection

const uint64_t wAddress = 0xB00B1E50C3LL;  //pipe for writing or transmitting data

const uint64_t rAddress = 0xB00B1E50A4LL;  //pipe for reading or recieving data


void setup()

{

  Serial.begin(57600);   //start serial to communicate process

  randomSeed(analogRead(0)); //create unique seed value for random number generation

  wirelessSPI.begin();           //Start the nRF24 module

  wirelessSPI.openWritingPipe(wAddress);    // setup pipe to transmit over

  wirelessSPI.openReadingPipe(1,rAddress);  //set up pipe to recieve data

  wirelessSPI.stopListening();  //turn off recieve capability so you can transmit

}


void loop()

{

  if(!done) { //true once you guess the right number
```

```
    byte randNumber = (byte)random(11); //generate random guess between 0 and 10


   if (!wirelessSPI.write( &randNumber, 1 )){  //if the write fails let the user know over serial monitor
        Serial.println("Guess delivery failed");
    }
    else { //if the write was successful
        Serial.print("Success sending guess: ");
        Serial.println(randNumber);


       wirelessSPI.startListening(); //switch to recieve mode to see if the guess was right
       unsigned long startTimer = millis(); //start timer, we will wait 200ms
       bool timeout = false;
       while ( !wirelessSPI.available() && !timeout ) { //run while no recieve data and not timed out
         if (millis() - startTimer > 200 ) timeout = true; //timed out
       }

       if (timeout) Serial.println("Last guess was wrong, try again"); //no data to recieve guess must have been wrong
       else  { //we recieved something so guess must have been right
         byte daNumber; //variable to store recived value
```

```
        wirelessSPI.read( &daNumber,1); //read value

        if(daNumber == randNumber) { //make sure it equals value we just sent, if so we are done

          Serial.println("You guessed right so you are done");

          done = true; //signal to loop that we are done guessing

        }

        else Serial.println("Something went wrong, keep guessing"); //this should never be true, but just in case

      }

      wirelessSPI.stopListening(); //go back to transmit mode


    }

  }

   delay(1000);

}
```

Transmitter2:

```
#include <SPI.h> //Call SPI library so you can communicate with the nRF24L01+

#include <nRF24L01.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/

#include <RF24.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/
```

```
const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or active mode (1)

const int pinCSN = 10; //This pin is used to tell the nRF24 whether the SPI communication is a command or message to send out

byte counter = 1; //used to count the packets sent

bool done = false; //used to know when to stop sending packets

RF24 wirelessSPI(pinCE, pinCSN); // Create your nRF24 object or wireless SPI connection

const uint64_t wAddress = 0xB00B1E50D2LL;            // Pipe to write or transmit on

const uint64_t rAddress = 0xB00B1E50B1LL;  //pipe to recive data on


void setup()

{

  Serial.begin(57600);   //start serial to communicate process

  randomSeed(analogRead(0)); //create unique seed value for random number generation

  wirelessSPI.begin();          //Start the nRF24 module

  wirelessSPI.openWritingPipe(wAddress);       //open writing or transmit pipe

  wirelessSPI.openReadingPipe(1,rAddress);  //open reading or recieve pipe

  wirelessSPI.stopListening(); //go into transmit mode

}
```

```
void loop()
{
  if(!done) { //true once you guess the right number
    byte randNumber = (byte)random(11); //generate random guess between 0 and 10


   if (!wirelessSPI.write( &randNumber, 1 )){  //if the write fails let the user know over serial monitor
       Serial.println("Guess delivery failed");
    }
    else { //if the write was successful
       Serial.print("Success sending guess: ");
       Serial.println(randNumber);


      wirelessSPI.startListening(); //switch to recieve mode to see if the guess was right
      unsigned long startTimer = millis(); //start timer, we will wait 200ms
      bool timeout = false;
      while ( !wirelessSPI.available() && !timeout ) { //run while no recieve data and not timed out
        if (millis() - startTimer > 200 ) timeout = true; //timed out
      }
```

```
      if (timeout) Serial.println("Last guess was wrong, try again"); //no data to recieve guess must have been wrong

      else  { //we recieved something so guess must have been right

        byte daNumber; //variable to store recived value

        wirelessSPI.read( &daNumber,1); //read value

        if(daNumber == randNumber) { //make sure it equals value we just sent, if so we are done

          Serial.println("You guessed right so you are done");

          done = true; //signal to loop that we are done guessing

        }

        else Serial.println("Something went wrong, keep guessing"); //this should never be true, but just in case

      }

      wirelessSPI.stopListening(); //go back to transmit mode


    }

  }

  delay(1000);

}
```

Receiver:

```
#include <SPI.h> //Call SPI library so you can communicate with the nRF24L01+

#include <nRF24L01.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/
```

```
#include <RF24.h> //nRF2401 libarary found at https://github.com/tmrh20/RF24/


const int pinCE = 9; //This pin is used to set the nRF24 to standby (0) or active mode (1)

const int pinCSN = 10; //This pin is used to tell the nRF24 whether the SPI communication is a command or message to send out

byte daNumber = 0; //The number that the transmitters are trying to guess

RF24 wirelessSPI(pinCE, pinCSN); // Declare object from nRF24 library (Create your wireless SPI)

const uint64_t rAddress[] = {0xB00B1E50D2LL, 0xB00B1E50C3LL}; //Create pipe addresses for the 2 nodes to recieve data, the "LL" is for LongLong type

const uint64_t wAddress[] = {0xB00B1E50B1LL, 0xB00B1E50A4LL}; //Create pipe addresses for the 2 nodes to transmit data, the "LL" is for LongLong type


void setup()

{

  randomSeed(analogRead(0)); //create unique seed value for random number generation

  daNumber = (byte)random(11); //Create random number that transmitters have to guess

  Serial.begin(57600);  //start serial to communication

  Serial.print("The number they are trying to guess is: ");

  Serial.println(daNumber); //print the number that they have to guess
```

```
  Serial.println();
  wirelessSPI.begin();  //Start the nRF24 module
  wirelessSPI.openReadingPipe(1,rAddress[0]);      //open pipe o for recieving meassages with pipe address
  wirelessSPI.openReadingPipe(2,rAddress[1]);      //open pipe o for recieving meassages with pipe address
  wirelessSPI.startListening();                 // Start listening for messages
}

void loop()
{
    byte pipeNum = 0; //variable to hold which reading pipe sent data
    byte gotByte = 0; //used to store payload from transmit module

    while(wirelessSPI.available(&pipeNum)){ //Check if recieved data
     wirelessSPI.read( &gotByte, 1 ); //read one byte of data and store it in gotByte variable
     Serial.print("Recieved guess from transmitter: ");
     Serial.println(pipeNum); //print which pipe or transmitter this is from
     Serial.print("They guess number: ");
     Serial.println(gotByte); //print payload or the number the transmitter guessed
     if(gotByte != daNumber) { //if true they guessed wrong
      Serial.println("Fail!! Try again.");
     }
```

```
    else { //if this is true they guessed right
     if(sendCorrectNumber(pipeNum)) Serial.println("Correct! You're done.");
//if true we successfully responded
     else Serial.println("Write failed"); //if true we failed responding
    }
    Serial.println();
   }


  delay(200);
}


//This function turns the reciever into a transmitter briefly to tell one of the
nRF24s
//in the network that it guessed the right number. Returns true if write to module
was
//successful
bool sendCorrectNumber(byte xMitter) {
   bool worked; //variable to track if write was successful
   wirelessSPI.stopListening(); //Stop listening, stop recieving data.
   wirelessSPI.openWritingPipe(wAddress[xMitter-1]); //Open writing pipe to
the nRF24 that guessed the right number
   if(!wirelessSPI.write(&daNumber, 1))  worked = false; //write the correct
number to the nRF24 module, and check that it was recieved
   else worked = true; //it was recieved
```

```
    wirelessSPI.startListening(); //Switch back to a reciever
    return worked;  //return whether write was successful
}
```

Libraries to be included:

https://github.com/maniacbug/RF24

Result:

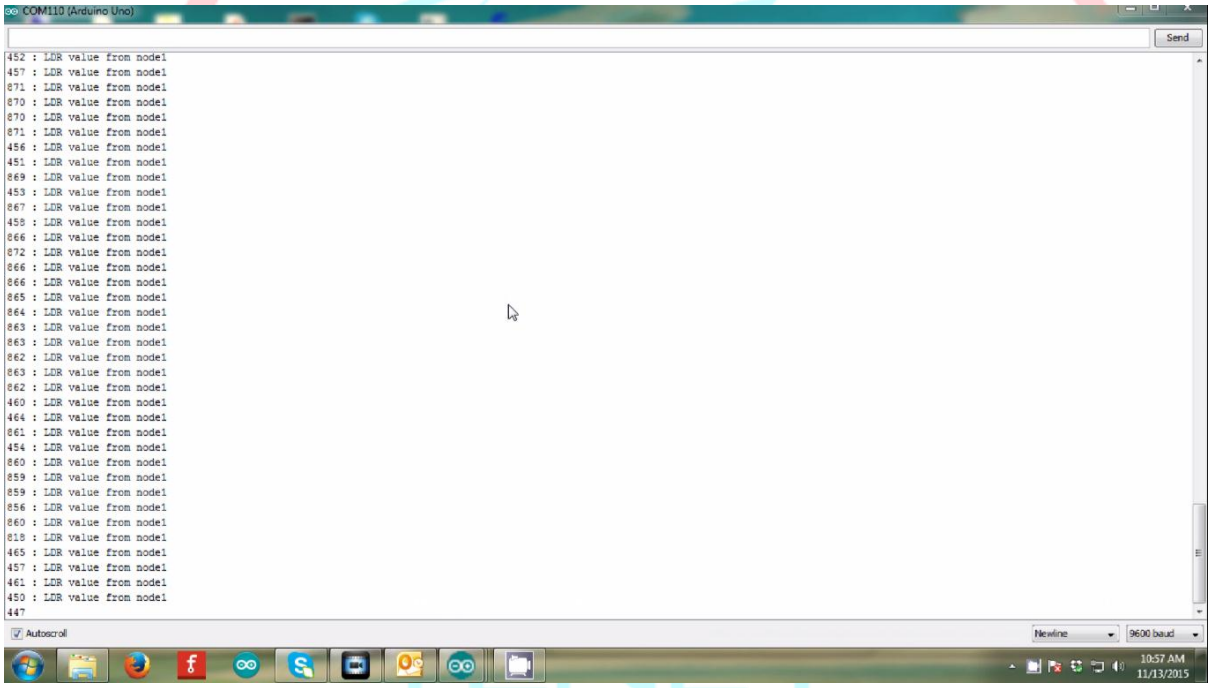


Figure 6 – Analog values read from node1

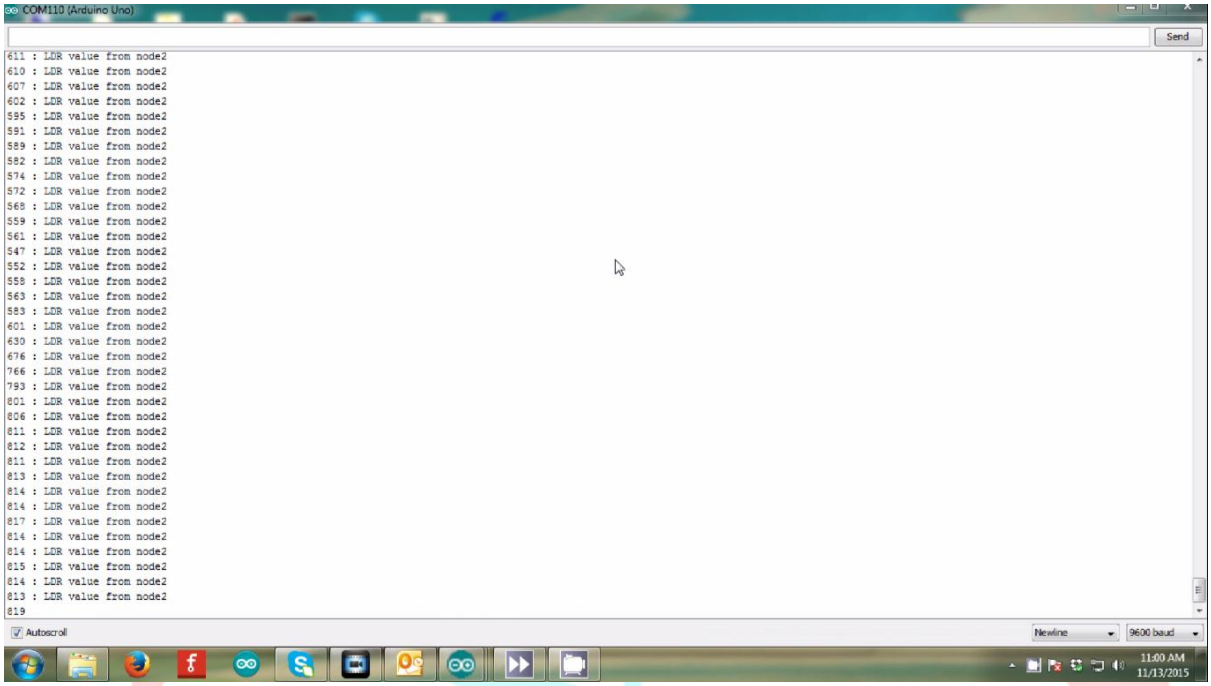


Figure 6 – Analog values read from node2

For product information:

1. http://www.tenettech.com/product/5294/nrf24l01module
2. HTTP://WWW.TENETTECH.COM/PRODUCT/202/ARDUINO-UNO-ARDUINO-UNO-R3
3. HTTP://WWW.TENETTECH.COM/PRODUCT/2609/BASIC-BREADBOARD

For more information please visit: www.tenettech.com

For technical query please send an e-mail: info@tenettech.com

9/3, 2nd floor, SreeLaksmi Complex, opp, to Vivekananda Park, Girinagar, Bangalore - 560085,

Email: info@tenettech.com, Phone: 080 - 26722726