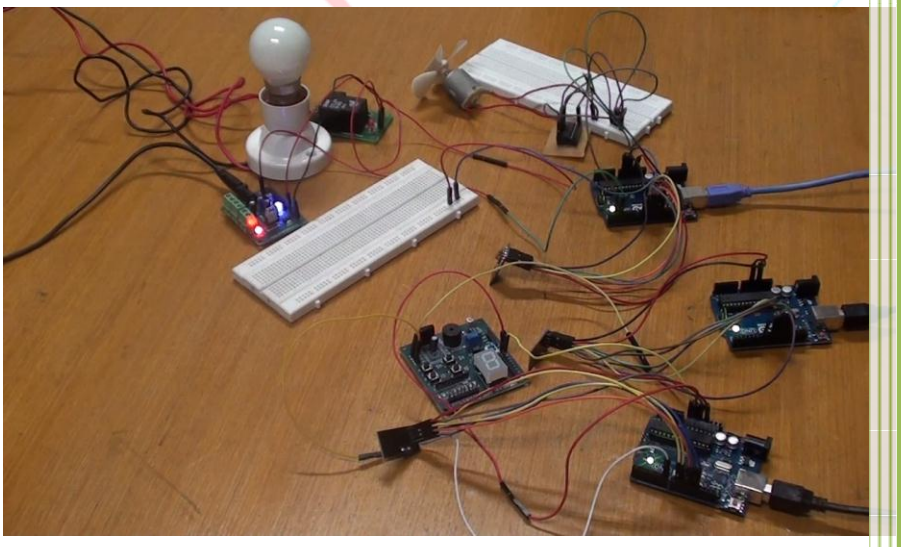
TENET TECHNETRONICS

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.

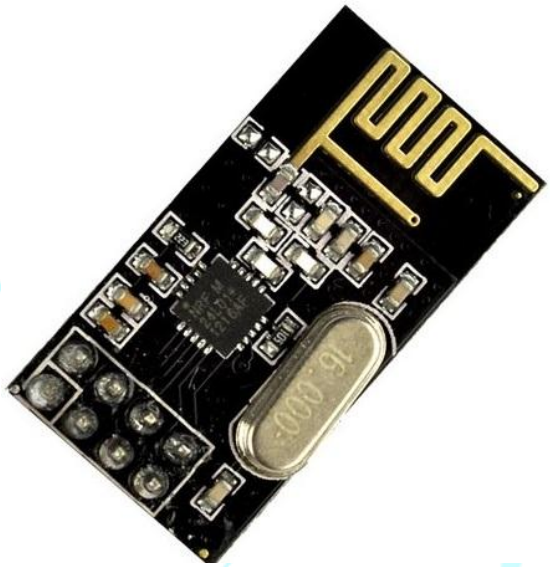


Figure 1 - nRF24L01 module

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:

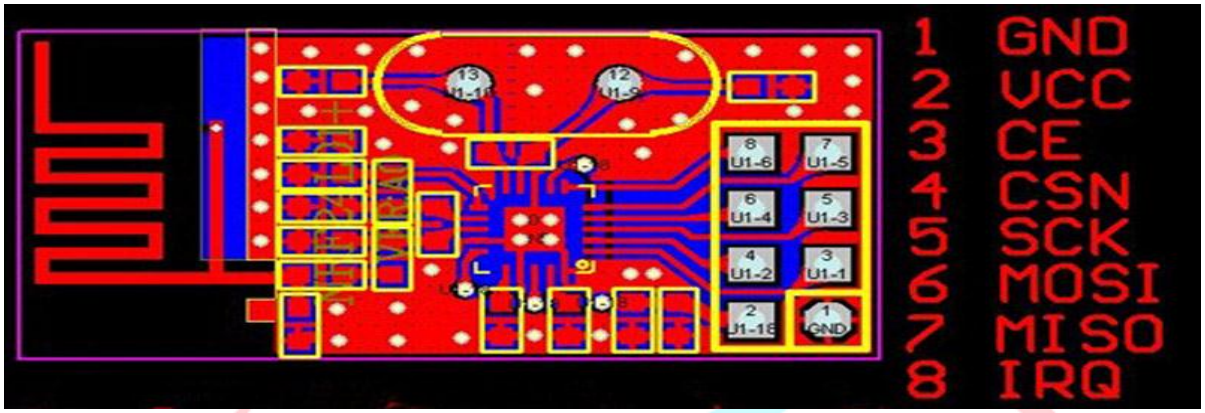


Figure 2 – nRF24L01 pin outs

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:

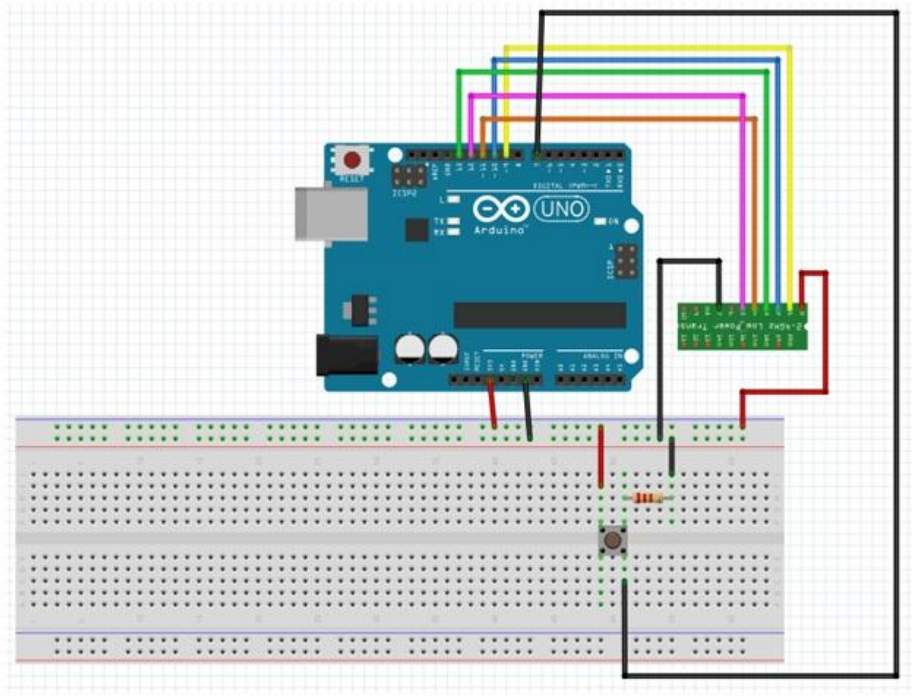


Figure 3 – Transmitter1 circuit diagram

Transmitter2 Block diagram:

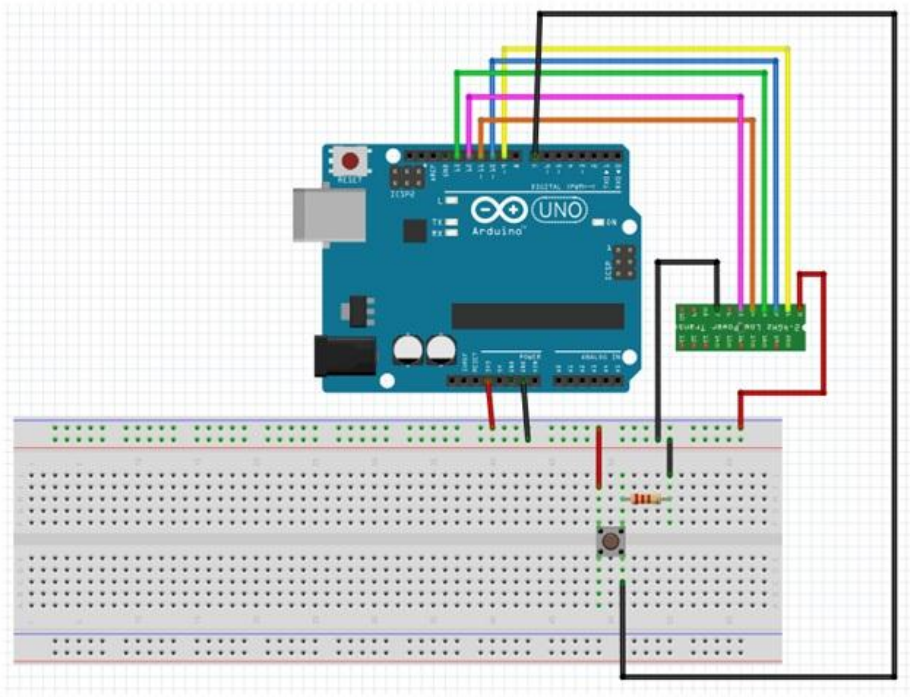


Figure 4 – Transmitter2 circuit diagram

Receiver:

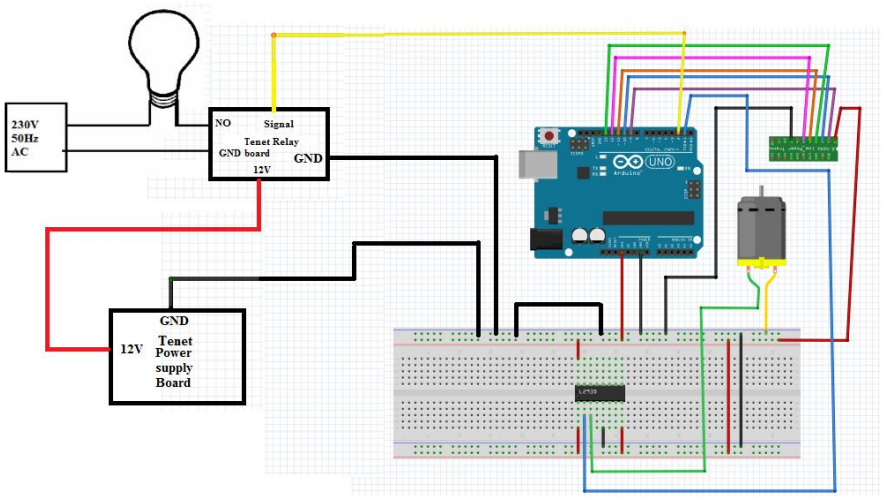


Figure 5 – Receiver circuit diagram

Code:

Transmitter1:

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
int transmitterId;

// Set up nRF24L01 radio on SPI bus plus pins 9 & 10
//Contacts from the radio to connect NRF24L01 pinamnam -> Arduino
```

```
//SCK -> 13
//MISO -> 12
//MOSI -> 11
//CSN -> 10
//CE -> 9

RF24 radio(9, 10);

// this is not the channel address, but the transmitter address
const uint64_t pipe = 0xE8E8F0F0E1LL;

//button connected to these pins
int buttonPin1 = 7;
void setup(void) {

    // CHANGE THIS PER EACH TRANSMITTER, from 0 to 4
    transmitterId = 1;
    radio.begin();

    // the following statements improve transmission range
    radio.setPayloadSize(2); // setting the payload size to the needed value
    radio.setDataRate(RF24_250KBPS); // reducing bandwidth
```

```
    radio.openWritingPipe(pipe); // set the transmitter address
}
void loop(void) {


    //until the button (buttonPin1) pressed send the package (id) to receiver Arduino
    if (digitalRead(buttonPin1) == HIGH) {
        // some implementations automatically shut down the radio after a transmission: this
        // ensures the radio is powered up before sending data
        radio.powerUp();
        // read and write expect a reference to the payload (& symbol)
        // second argument is the packet length in bytes (sizeof(int) == 2)
        radio.write(&transmitterId, 2);
    }
}
```

Transmitter2:

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
int transmitterId;
```

```
// Set up nRF24L01 radio on SPI bus plus pins 9 & 10
//Contacts from the radio to connect NRF24L01 pinamnam -> Arduino

//SCK -> 13
//MISO -> 12
//MOSI -> 11
//CSN -> 10
//CE -> 9

RF24 radio(9, 10);

// this is not the channel address, but the transmitter address
const uint64_t pipe = 0xE8E8F0F0E1LL;
//button connected to these pins
int buttonPin1 = 7;
void setup(void) {
        // CHANGE THIS PER EACH TRANSMITTER, from 0 to 4
        transmitterId = 1;
        radio.begin();
        // the following statements improve transmission range
        radio.setPayloadSize(2); // setting the payload size to the needed value
        radio.setDataRate(RF24_250KBPS); // reducing bandwidth
```

```
        radio.openWritingPipe(pipe); // set the transmitter address
}
void loop(void) {


        //until the button (buttonPin1) pressed send the package (id) to receiver Arduino
        if (digitalRead(buttonPin1) == HIGH) {


                // some implementations automatically shut down the radio after a transmission: this
                // ensures the radio is powered up before sending data
                radio.powerUp();


                // read and write expect a reference to the payload (& symbol)
                // second argument is the packet length in bytes (sizeof(int) == 2)
                radio.write(&transmitterId, 2);
        }
}
```

Receiver:

```
#include <SPI.h>

#include "nRF24L01.h"

#include "RF24.h"

int senderId;



// Set up nRF24L01 radio on SPI bus plus pins 9 & 10

//Contacts from the radio to connect NRF24L01 pinamnam -> Arduino


//SCK -> 13

//MISO -> 12

//MOSI -> 11

//CSN -> 10

//CE -> 9


RF24 radio(9, 10);

// this is not the channel address, but the transmitter address

const uint64_t pipe = 0xE8E8F0F0E1LL;


//LEDs connected to these pins

// ENSURE YOU HAVE THE RIGHT DIGITAL PINS HERE
```

```
int LEDpins[5] = { 2, 3, 4, 5, 6 };


void setup(void) {
        Serial.begin(9600);
        radio.begin();
        // the following statements improve transmission range
        radio.setPayloadSize(2); // setting the payload size to the needed value
        radio.setDataRate(RF24_250KBPS); // reducing bandwidth


        radio.openReadingPipe(1, pipe); // Open one of the 6 pipes for reception


        radio.startListening(); // begin to listen


        // Enable all the LED pins as output
        for (int i = 0; i < 5; i++) {
                pinMode(LEDpins[i], OUTPUT);
                digitalWrite(LEDpins[i], LOW); // this is unnecessary but good
practice nonetheless
        }
}
void loop(void) {


        // Turns off all the LEDs
```

```
        for (int i = 0; i < 5; i++) {
                digitalWrite(LEDpins[i], LOW);
        }
        if (radio.available()) {
                // this while is here to throw away all the packets but the last one
                bool done = false;
                while (!done) {
                        // read and write expect a reference to the payload (&
symbol)
                        // second argument is the packet length in bytes (sizeof(int)
== 2)
                        done = radio.read(&senderId, 2);
                }
                //Light up the correct LED for 50ms
                digitalWrite(LEDpins[senderId], HIGH);
                Serial.print("LED ");
                Serial.print(senderId);
                Serial.println(" On");
                delay(50);
        }
}
```

Libraries to be included:

https://github.com/maniacbug/RF24

Result:

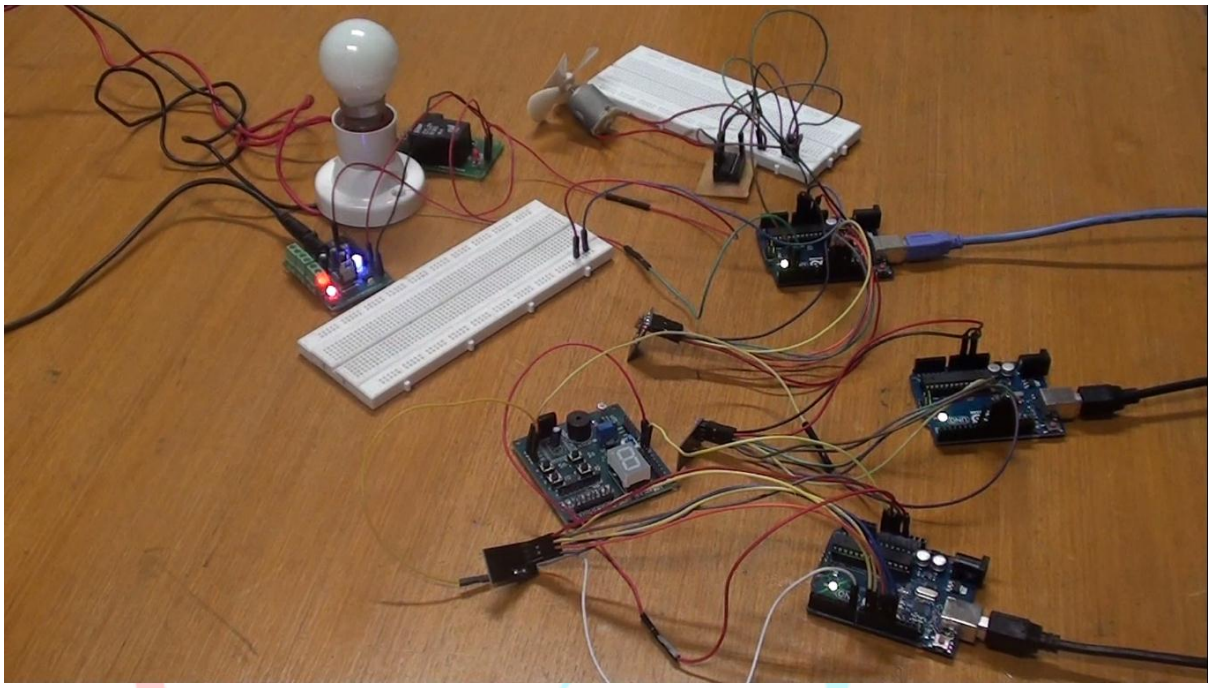


Figure 6 – No Button pressed

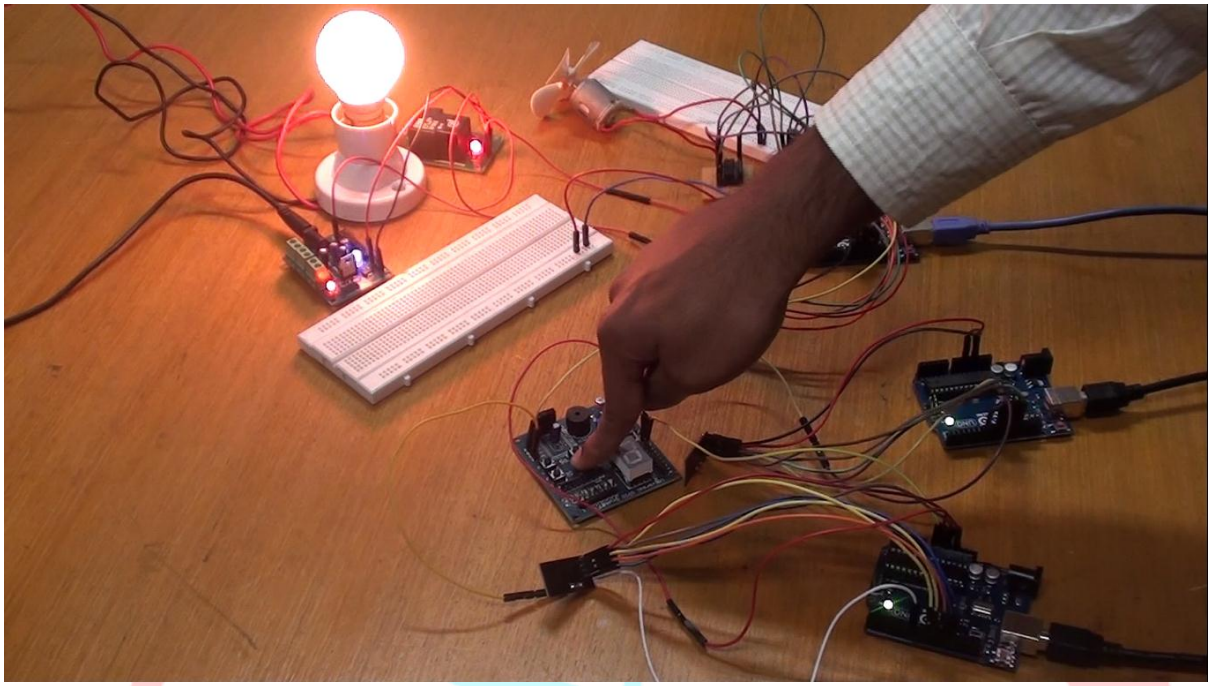


Figure 7 – Button1 is pressed

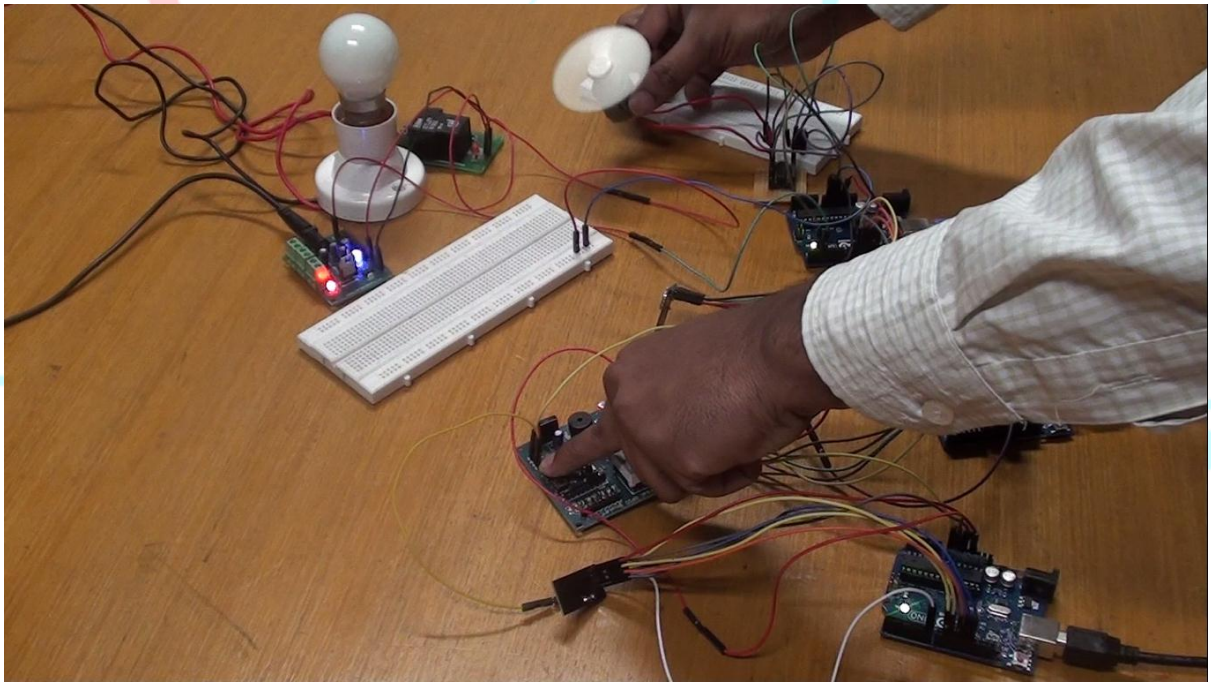


Figure 8 – Button2 is pressed

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