

**Campus Quissamã**

**Curso Integrado Informática**

**Professor: Renato**

**Turma: 2º ano informática**

**Aluno: Arthur França Freitas e Vitória Silva Nascimento Cabral**

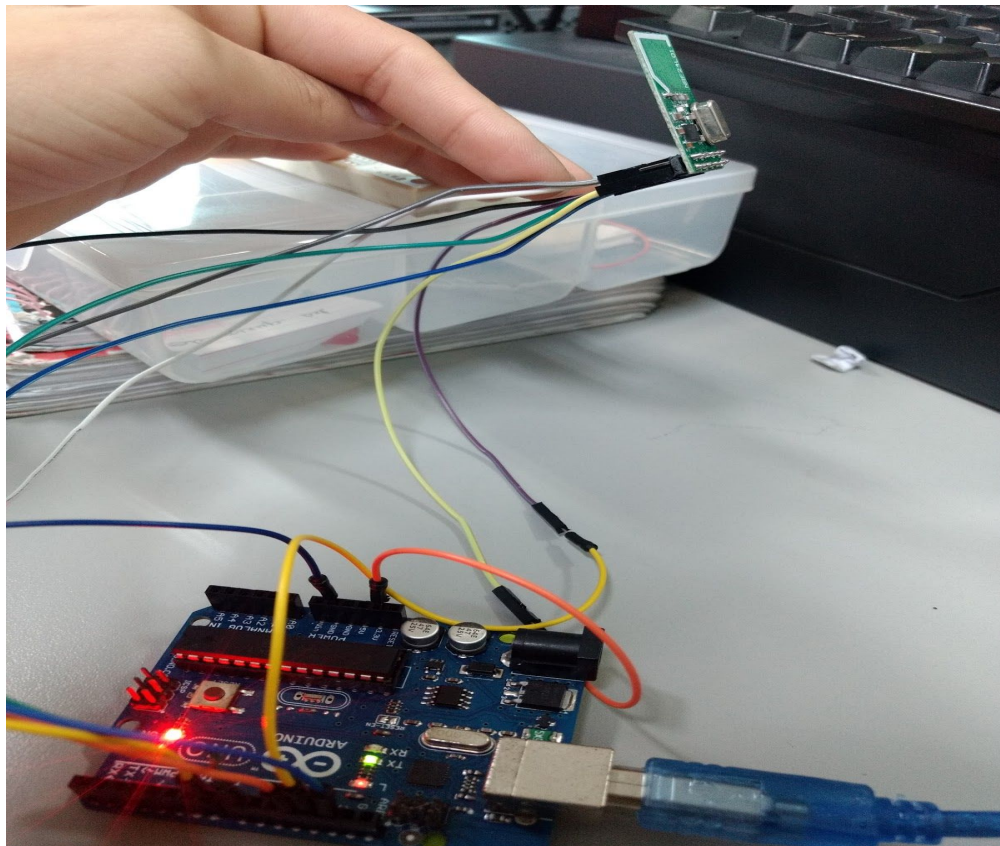
**Assunto: Relatório**

- **Relatório - 02/07/2019**

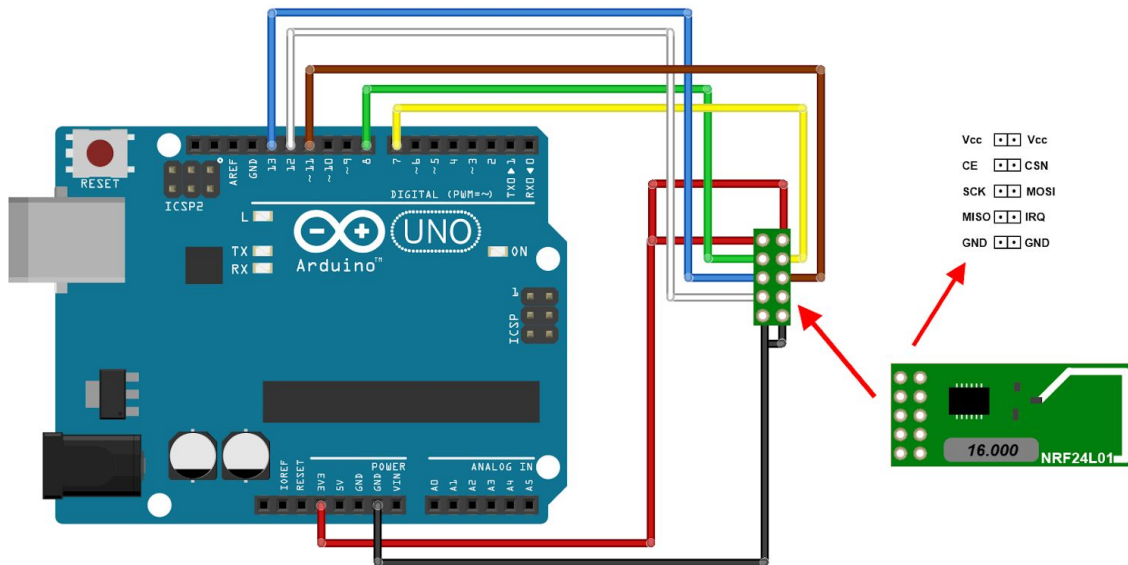
Neste relatório utilizamos o sensor de comunicação, mais conhecido como wireless NRF24L01. Utilizamos o mesmo juntamente com o Arduino Uno.

Fomos pedidos a utilizar o sensor, mediante a falta do sensor "ultrassom" e nos foi dado a oportunidade de aprender a utilizar o mesmo. E foi visto que, é bastante interessante mediante ter a possibilidade em se comunicar entre máquinas (computadores).

- **Foto da conclusão do projeto:**



- **Modelo proposto pelo site:**



- **Código:**

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

//
// Hardware configuration
//

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

RF24 radio(9,10);

//
// Topology
//

// Radio pipe addresses for the 2 nodes to communicate.
const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

//
// Role management
//
// Set up role. This sketch uses the same software for all the nodes
// in this system. Doing so greatly simplifies testing.
//

// The various roles supported by this sketch
typedef enum { role_ping_out = 1, role_pong_back } role_e;
```

```

// The debug-friendly names of those roles
const char* role_friendly_name[] = { "invalid", "Ping out", "Pong back"};

// The role of the current running sketch
role_e role = role_pong_back;

void setup(void)
{
  //
  // Print preamble
  //

  Serial.begin(57600);
  printf_begin();
  printf("nrRF24/examples/GettingStarted/nr");
  printf("ROLE: %snr",role_friendly_name[role]);
  printf("**** PRESS 'T' to begin transmitting to the other nodenr");

  //
  // Setup and configure rf radio
  //

  radio.begin();

  // optionally, increase the delay between retries & # of retries
  radio.setRetries(15,15);

  // optionally, reduce the payload size. seems to
  // improve reliability
  //radio.setPayloadSize(8);

  //
  // Open pipes to other nodes for communication
  //

  // This simple sketch opens two pipes for these two nodes to communicate
  // back and forth.
  // Open 'our' pipe for writing
  // Open the 'other' pipe for reading, in position #1

  //(we can have up to 5 pipes open for reading)

  //if ( role == role_ping_out )
  {
    //radio.openWritingPipe(pipes[0]);
    radio.openReadingPipe(1,pipes[1]);
  }
  //else
  {
    //radio.openWritingPipe(pipes[1]);
    //radio.openReadingPipe(1,pipes[0]);
  }
}

```

```

//
// Start listening
//

radio.startListening();

//
// Dump the configuration of the rf unit for debugging
//

radio.printDetails();
}

void loop(void)
{
//
// Ping out role. Repeatedly send the current time
//

if (role == role_ping_out)
{
// First, stop listening so we can talk.
radio.stopListening();

// Take the time, and send it. This will block until complete
unsigned long time = millis();
printf("Now sending %lu...",time);
bool ok = radio.write( &time, sizeof(unsigned long) );

if (ok)
printf("ok...");
else
printf("failed.nr");

// Now, continue listening
radio.startListening();

// Wait here until we get a response, or timeout (250ms)
unsigned long started_waiting_at = millis();
bool timeout = false;
while ( ! radio.available() && ! timeout )
if (millis() - started_waiting_at > 200 )
timeout = true;

// Describe the results
if ( timeout )
{
printf("Failed, response timed out.nr");
}
else
{
// Grab the response, compare, and send to debugging spew
unsigned long got_time;

```

```

radio.read( &got_time, sizeof(unsigned long) );

// Spew it
printf("Got response %lu,round-trip delay: %lu\n",got_time,millis()-got_time);
}

// Try again 1s later
delay(1000);
}

//
// Pong back role. Receive each packet, dump it out, and send it back
//

if ( role == role_pong_back )
{
// if there is data ready
if ( radio.available() )
{
// Dump the payloads until we've gotten everything
unsigned long got_time;
bool done = false;
while (!done)
{
// Fetch the payload, and see if this was the last one.
done = radio.read( &got_time, sizeof(unsigned long) );

// Spew it
printf("Got payload %lu...",got_time);

// Delay just a little bit to let the other unit
// make the transition to receiver
delay(20);
}

// First, stop listening so we can talk
radio.stopListening();

// Send the final one back.
radio.write( &got_time, sizeof(unsigned long) );
printf("Sent response.\n");

// Now, resume listening so we catch the next packets.
radio.startListening();
}
}

//
// Change roles
//

if ( Serial.available() )
{

```

```

char c = toupper(Serial.read());
if ( c == 'T' && role == role_pong_back )
{
printf("*** CHANGING TO TRANSMIT ROLE -- PRESS 'R' TO SWITCH BACKnr");

// Become the primary transmitter (ping out)
role = role_ping_out;
radio.openWritingPipe(pipes[0]);
radio.openReadingPipe(1,pipes[1]);
}
else if ( c == 'R' && role == role_ping_out )
{
printf("*** CHANGING TO RECEIVE ROLE -- PRESS 'T' TO SWITCH BACKnr");

// Become the primary receiver (pong back)
role = role_pong_back;
radio.openWritingPipe(pipes[1]);
radio.openReadingPipe(1,pipes[0]);
}
}
}
// vim:cin:ai:sts=2 sw=2 ft=cpp

```

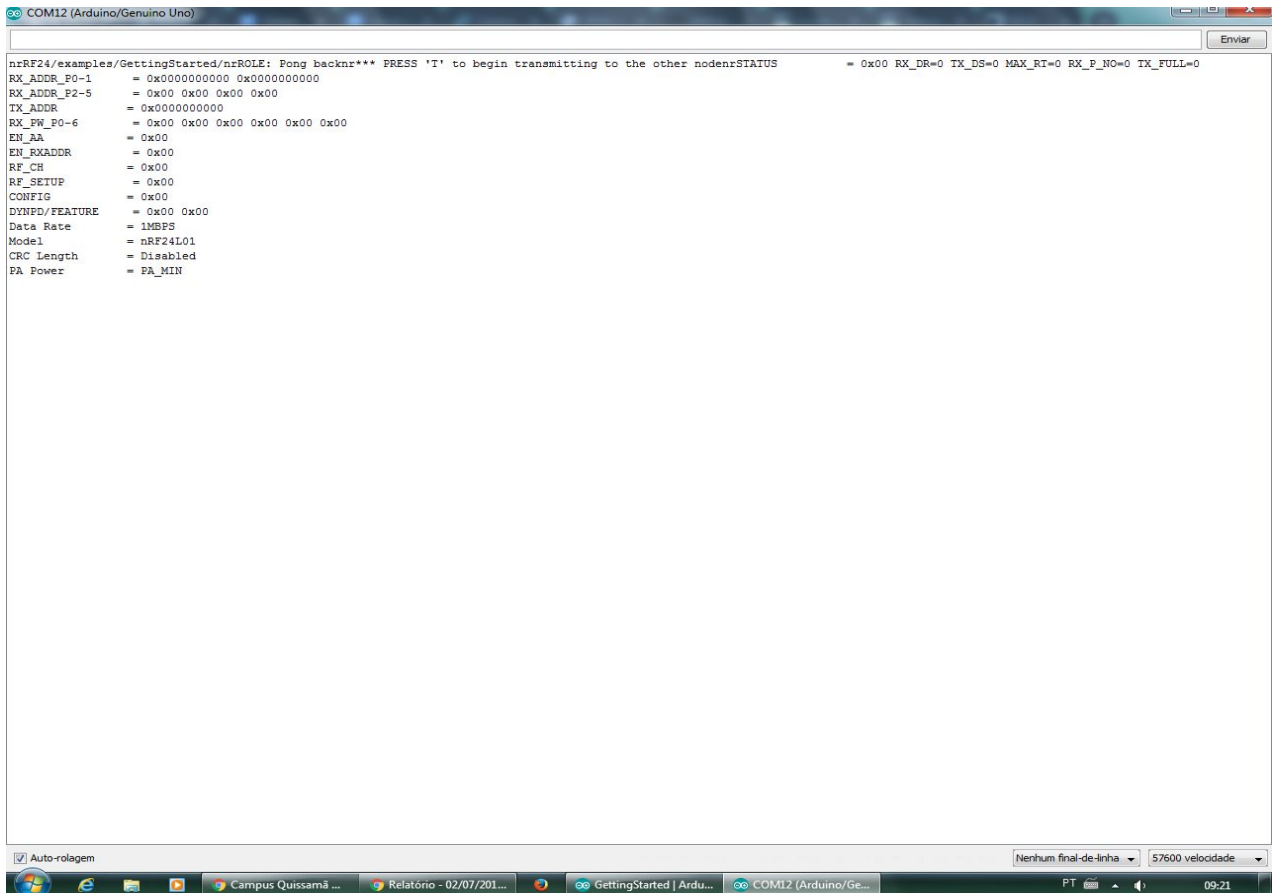
- **Componentes utilizados:**

- Arduino Uno;
- Sensor de Comunicação;
- 7 cabos fêmeas/macho;

- **Tabela utilizada:**

| Pino | Nome | Função                | Ligação Arduino |
|------|------|-----------------------|-----------------|
| 1    | Vcc  | Alimentação           | 3.3 V           |
| 2    | Vcc  | Alimentação           | 3.3 V           |
| 3    | CE   | Chip Enable RX / TX   | Pino 8          |
| 4    | CSN  | SPI Chip Select       | Pino 7          |
| 5    | SCK  | SPI Clock             | Pino 13         |
| 6    | MOSI | SPI Slave Data Input  | Pino 11         |
| 7    | MISO | SPI Slave Data Output | Pino 12         |
| 8    | IRQ  | Interrupção           | Não utilizado   |
| 9    | GND  | Terra                 | GND             |
| 10   | GND  | Terra                 | GND             |

- **Foto do Monitor:**



```
COM12 (Arduino/Genuino Uno)
Enviar

nRF24/examples/GettingStarted/nrROLE: Pong backnr*** PRESS 'I' to begin transmitting to the other node
RX_ADDR_P0-1 = 0x00000000 0x00000000
RX_ADDR_P2-5 = 0x00 0x00 0x00 0x00
TX_ADDR = 0x00000000
RX_PW_P0-6 = 0x00 0x00 0x00 0x00 0x00 0x00
EN_AA = 0x00
EN_RXADDR = 0x00
RF_CH = 0x00
RF_SETUP = 0x00
CONFIG = 0x00
DYNPD/FEATURE = 0x00 0x00
Data Rate = 1MBPS
Model = nRF24L01
CRC Length = Disabled
PA Power = PA_MIN

= 0x00 RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=0 TX_FULL=0
```

- **Conclusão:**

Chegamos a conclusão que, esse sensor wireless, é um sensor que utilizamos para nos comunicarmos com outra máquina, tendo em vista que é necessário dois computadores para realizar o processo, onde usaremos para nos comunicar o “monitor” arduino.

Na finalização desse projeto, foi visto que era necessário a instalação da biblioteca “RF24”, pois apenas com a instalação da mesma que o código irá funcionar. Visto isso, instalamos a biblioteca, e conseguimos concluir nosso projeto com sucesso.

- **Sites utilizados:**

<https://www.filipeflop.com/blog/arduino-modulo-nrf24l01-tutorial/> (site explicativo)  
<https://github.com/maniacbug/RF24> (link de download da biblioteca)