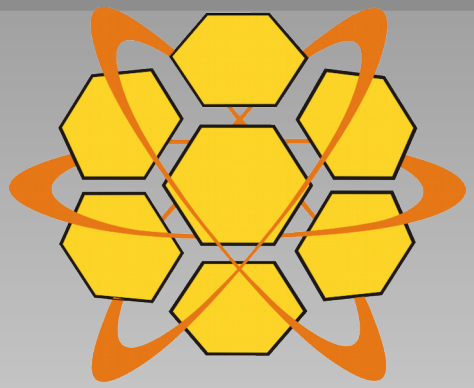
RX - TX com Arduino

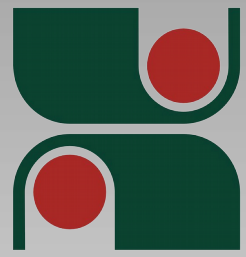
Palestrante:

DANIEL S. CAMARGO



COLMÉIA

Grupo de Pesquisa em
Software e Hardware Livre



UDESC
Joinville

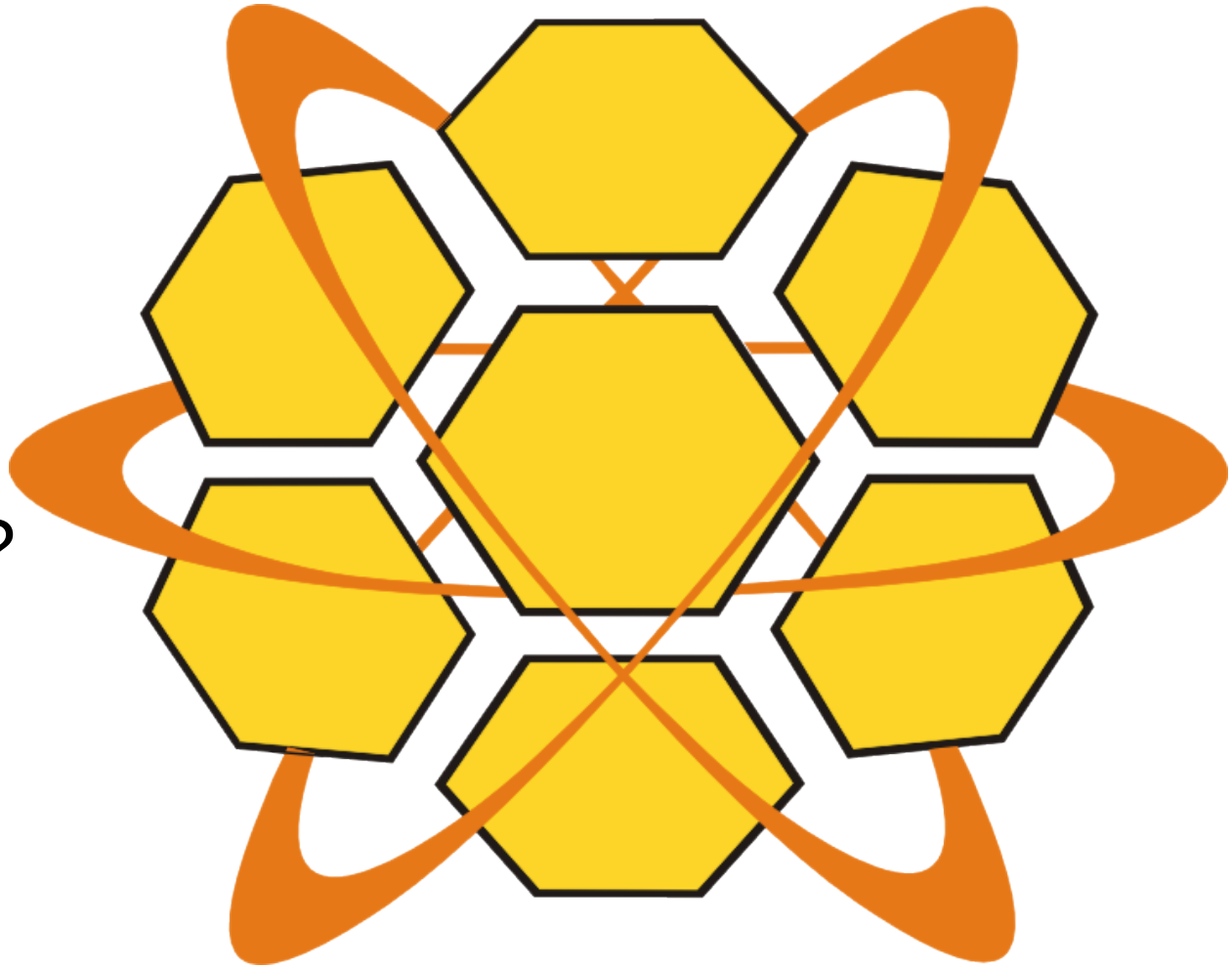
Material Disponível no site:
www.colmeia.udesc.br

Agenda

- O Colmeia;
- O que é RxTx;
- Aplicações;
- Exemplos com Arduino;
- Considerações finais;
- Referências.

O COLMÉIA

- Quem somos?
- O que fazemos ?



Informações: www.colmeia.udesc.br
e-mail: contato@colmeia.udesc.br

O que é Rx - Tx?

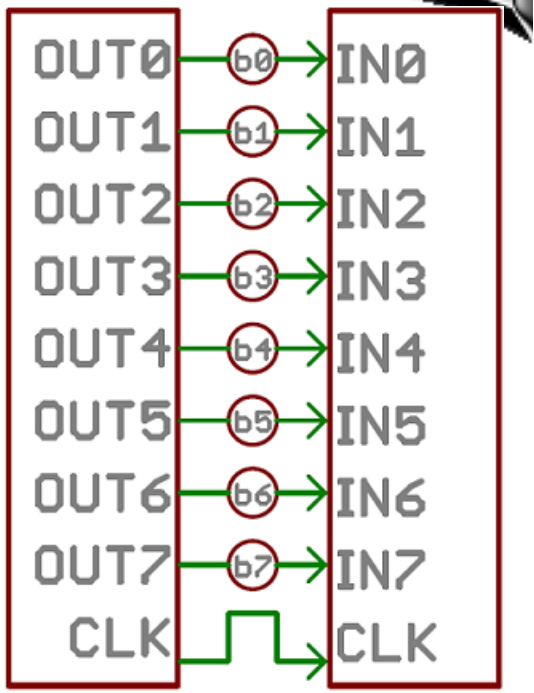
Não, não falaremos sobre Jets!

Sea Doo 2015 RXTX 260 Jet Ski

Rx - Tx

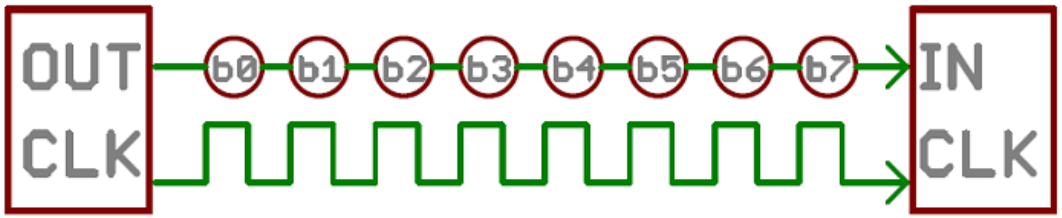
- No conceito de comunicação:
  Rx = Recepção e **Tx** = Transmissão de dados.
- Para haver comunicação entre dispositivos embarcados, é necessário ter o mesmo protocolo de comunicação em comum .
- Dois modos: **SERIAL** e **PARALELO**.

Rx - Tx – Paralelo/Serial



- **PARALELO**: Transfere múltiplos bits ao mesmo tempo. Requer barramentos de dados (acima de 8 fios);

- **SERIAL**: Transmite seus dados um bit por vez. Requerem de 1 a 4 fios.



Rx - Tx – Sync / Async

- **Sync**: Todos os dispositivos sincronizados por um relógio externo 'CLK'. Requer um fio extra, mas ganha-se com velocidade. Ex.: I²C e SPI.
- **Async**: Minimiza fios e portas I/O. Dispositivos sincronizados por sinais especiais ao longo da transmissão. Ex.: GPS, Bluetooth, XBee, LCDs...

I²C: Inter-Integrated Circuit
SPI: Serial Peripheral Interface

Rx - Tx – Async

- O protocolo serial assíncrono tem algumas regras internas que eliminam erros na transferência, como:
  - Bits de dados;
  - Bits de sincronização;
  - Bits de paridade; e
  - *Baud Rate* (bps).

Rx - Tx – Async (cont.)

- Mas é necessário assegurar que ambos dispositivos possuam a mesma configuração.
- Cada bloco possui normalmente 1 byte;
- Configuração mais comum é:
  9600 8N1: *9600 baud, 8 data bits, no parity, e 1 stop bit.*

=7975= OK

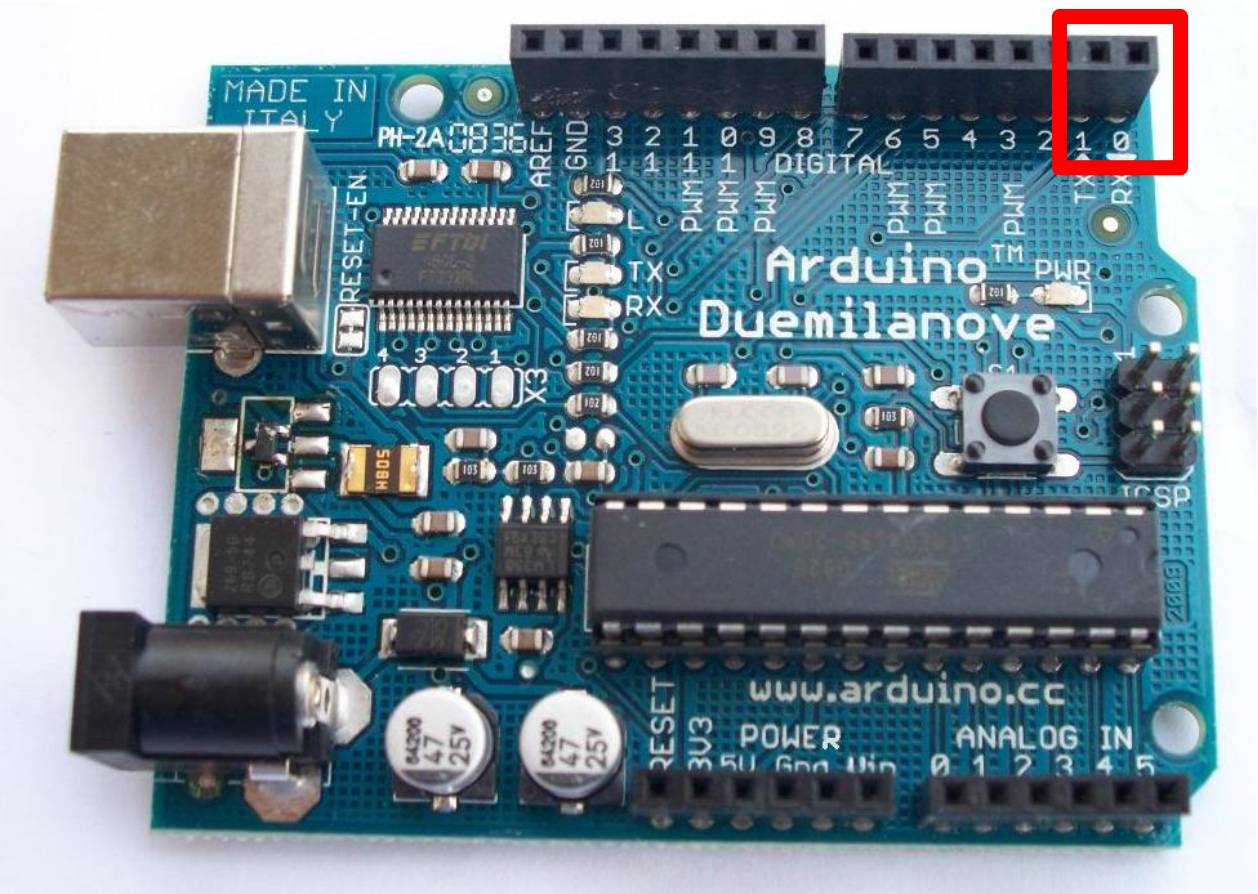
Rx - Tx no Arduino

- Os exemplos a seguir serão implementados em *simplex* onde apenas enviam ou recebem informação;

- *Full-duplex:* envia e recebe simultaneamente;
- *Half-duplex:* enviar depois de receber (*v-v*);

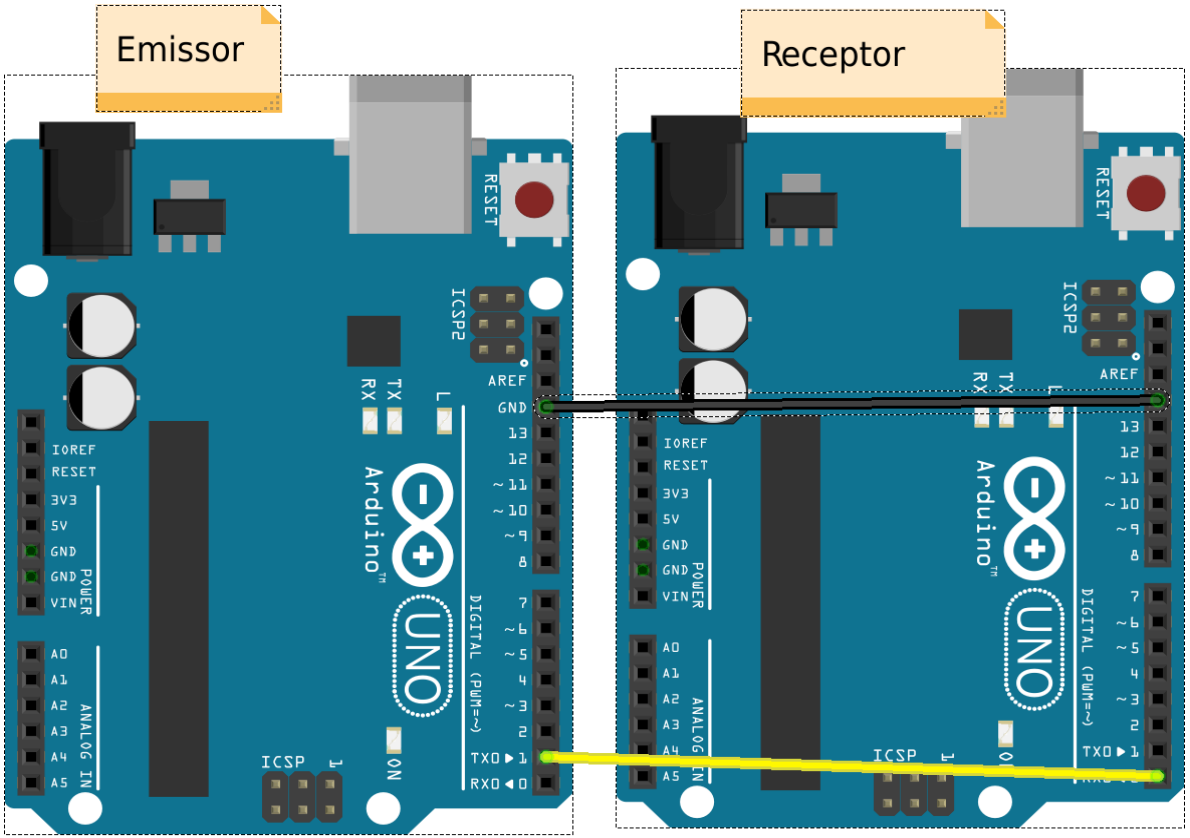
Disposição no Arduino

Entrada: D0 (Rx) e saída: D1 (Tx)



Fonte: http://arduino.cc/

Exemplo 1
Cabeado simples com dois nodos



Exemplo 1 – Cabeado Simples

Não utiliza qualquer biblioteca, fazendo apenas o tratamento nos Arduinos.

Utiliza apenas as portas seriais nativas (D0=RX e D1=TX) competindo com FTDI (Cuidado no upload do código).

Permite comunicação direta por meio de fios, conectando as portas padrão.

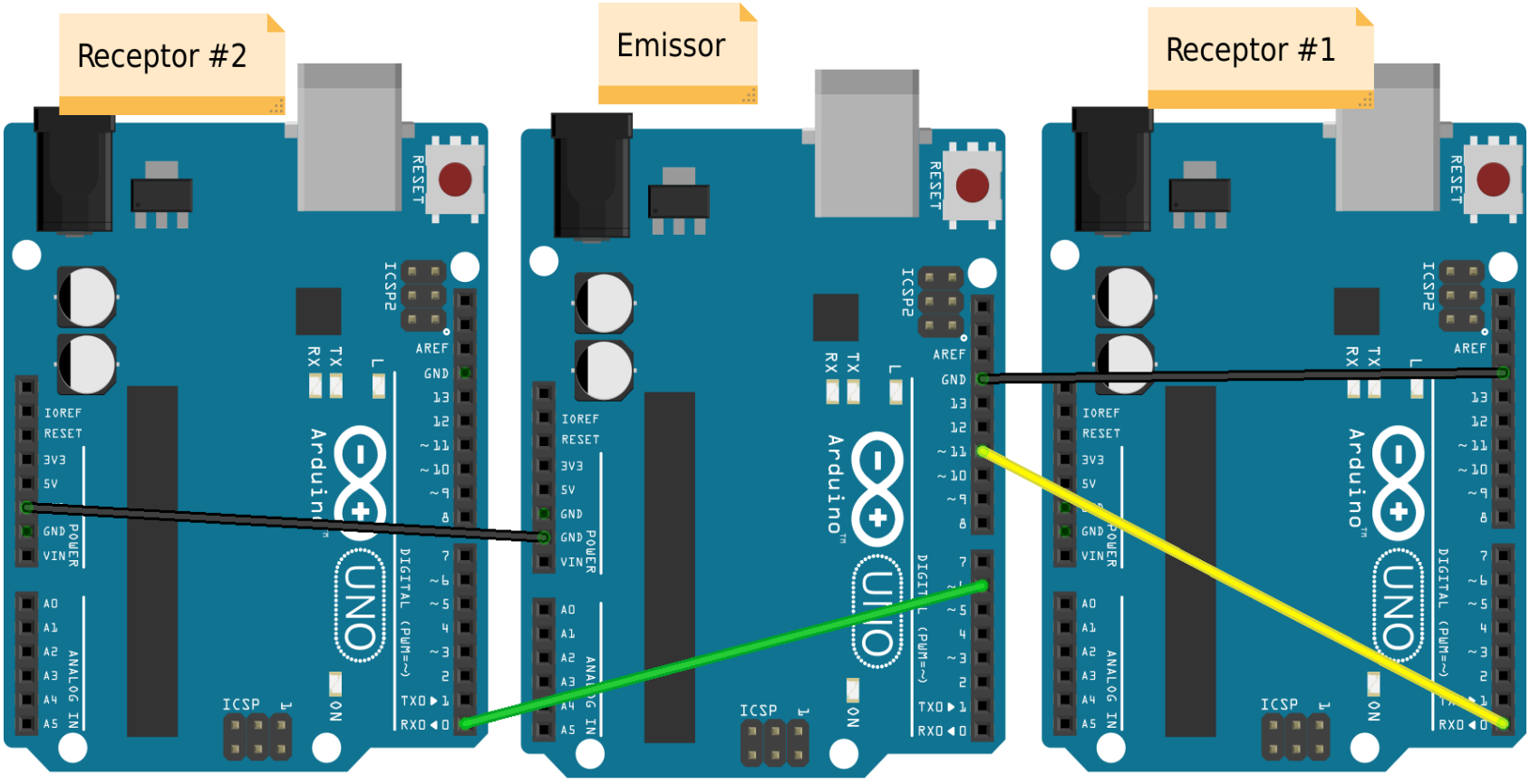
Nível de dificuldade baixo.

Obs.: Configurar serial para NL&CR

```
/*receptor*/

String inputString = "";

void setup(){
  Serial.begin(9600);
  Serial.println("Receptor");
  inputString.reserve(144);
}

void loop(){
   if (Serial.available() > 0) {
          char inChar = (char)Serial.read();
          inputString += inChar;
          if (inChar == '\n'){
            Serial.print("->: ");
            Serial.println(inputString);
            inputString = "";
          }
  }
}
```

```
/*emissor*/

void setup(){
  Serial.begin(9600);
  Serial.println("Emissor");
}

void loop(){
  if (Serial.available() > 0){
    int x = Serial.read();
    Serial.write(x);
  }
}
```

Exemplo 2
Cabeado com Biblioteca e *n* nodos



Exemplo 2 – Cabeado *n* Nodos

Utilizando a biblioteca nativa do Arduino *SoftwareSerial*.

Possibilita abrir várias portas seriais em pinos não nativos diferentes de (0 e 1).

Permite comunicação direta por meio de fios, conectando as portas configuradas.

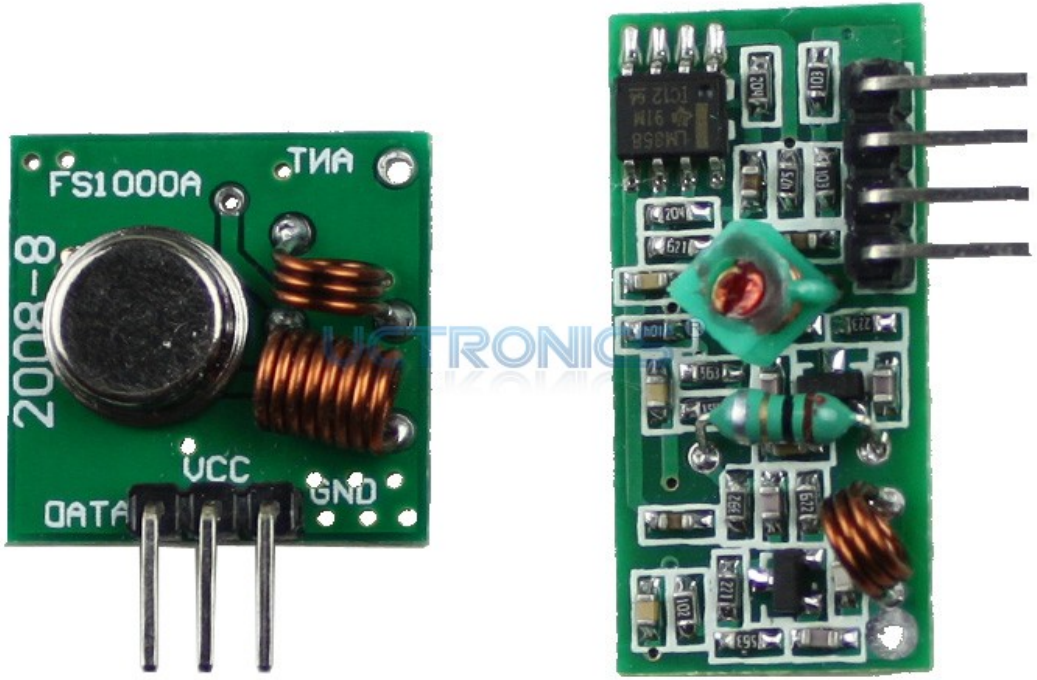
Código do receptor continua o mesmo.

Nível baixo++.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial1(10, 11); // RX, TX
SoftwareSerial mySerial2(5, 6); // RX, TX

bool enableHost1=false;
bool enableHost2=false;

void setup(){
  Serial.begin(9600);
  Serial.println("Ola, sou o Emissor!");
  mySerial1.begin(9600);
  mySerial1.println("Ola receptor1?");
  mySerial2.begin(9600);
  mySerial2.println("Ola receptor2?");
}

void loop(){
  if (Serial.available() > 0){
    int x = Serial.read(); //leia dado
```

Código Continua...

Exemplo 3
Rádio Frequência



Exemplo 3 – RF (Rx)

Utilizando a biblioteca *VirtualWire* não nativa do Arduino.

Possibilita comunicação serial por módulos RF ligado aos pinos de (D11 e D12).

Nível: médio.

```
// receiver {4 pins, bigger} Atach data to 11D port

#include "VirtualWire.h"

void setup(){
  Serial.begin(9600);

  vw_setup(500); // Bits per sec
  vw_rx_start();
}

void loop(){
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;

  if (vw_get_message(buf, &buflen)) {
    Serial.print("-> ");

    for (int i = 0; i < buflen; i++)
      Serial.write(buf[i]);
  }
}
```

Exemplo 3 - RF (Tx)

Utilizando a biblioteca *VirtualWire* não nativa do Arduino.

Possibilita comunicação serial por módulos RF ligado aos pinos de (D11 e D12).

Nível: médio.

```
// Transmitter (3 pins, smaller) Data 12D
#include "VirtualWire.h"

String inputString = "";
boolean stringComplete = false;

void setup(){
  Serial.begin(9600);
  Serial.println("Transmissor");
  vw_setup(500); // Bits per sec
  inputString.reserve(144);
}

void loop(){
  if (stringComplete) {
    Serial.println(inputString);
    const char *sent = inputString.c_str();
    vw_send((uint8_t*)sent, strlen(sent));
    vw_wait_tx(); // Wait message gone
    inputString = ""; // clear the string:
    stringComplete = false;
  }
}

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n')
      stringComplete = true;
  }
}
```

Exemplo 4
X-Bee
(Sem implementação)

XBee

Possui duas macro-versões: S1 e S2;
Protocolos S1: *IEEE 802.15.4* e S2: Zigbee+S1;
Permite formar redes complexas: Star, Tree, Mesh (S2);
Devem ser configurados pelo *software* x-ctu;
Possui três tipos de nodos: Coordenador, Roteador, Final;
Configurações de Segurança, Consumo, etc;
Permite redes com 65K nodos.

Exemplo 4 - XBee (Tx)

Utilizando a biblioteca XBee não nativa do Arduino.

Envia até 65 bytes por vez, alocado em vetor byte a byte.

Indicado o uso de Shield.

Nível: complexo

```
#include <XBee.h>
#include <SoftwareSerial.h>

XBee xbee = XBee(); // Cria objeto XBee
SoftwareSerial nss(2, 3); //Novo RxTx

uint8_t payload[] =  {0,0}; /** Vetor de carga **/
uint32_t msb = 0x0013a200;
uint32_t lsb = 0x40ae9a7f;
XBeeAddress64 addr64 = XBeeAddress64(msb,lsb); //Nodo Coordenador
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
XBeeResponse response = XBeeResponse();
int pin5 = 0, pin4 = 0, pin3 = 0;

void setup() {
  Serial.begin(9600);
  nss.begin(9600);
  xbee.setSerial(Serial);
}

void loop() {
  pin5 = analogRead(5);
  payload[0] = pin5 >> 8 & 0xff; // pega o valor da divisao por 0xff (2
  payload[1] = pin5 & 0xff; // pega o resto da divisao por 0xff (256)
  xbee.send(zbTx);
  if (xbee.readPacket(500)) {
    if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) {
      xbee.getResponse().getZBTxStatusResponse(txStatus);
      if (txStatus.getDeliveryStatus() == SUCCESS)
        nss.println("Sucesso!");
    else
      nss.println("Fracasso!");
  }
  }else if (xbee.getResponse().isError())
    nss.println("Erro ao ler pacote!");
   else
    nss.println("TimeOut");
  delay(1000);
}
```

Considerações Finais

- Pode-se perceber a simplicidade de uso das portas Rx/Tx do Arduino, com o uso de bibliotecas.
- A segurança da rede depende de abordagem mais complexa em sua implementação .
- Existem diversos tipos de comunicação que utilizam Rx/Tx do Arduino.

Bibliografia e Materiais para consulta

- ***http://www.arduino.cc*** [Página oficial do Arduino]
- ***http://vimeo.com/31389230*** [Documentário Arduino]
- ***http://fritzing.org/*** [Software de desenho de circuitos]
- ***https://learn.sparkfun.com/tutorials/serial-communication*** [Sparkfun]
- ***http://www.digi.com/xbee/*** [Site oficial Xbee]

Perguntas?

Alguma dúvida?

- E-mail: contato@colmeia.udesc.br

Obrigado!