

Sistemas de comunicação sem fios com ligação ao *Arduino* ou *LilyPad*

Existem duas formas principais de estabelecer uma comunicação sem fios entre estes micro-controladores, o *Xbee* e o *Bluetooth*.

Xbee

O *Xbee* permite colocar dois *Arduinos* ou *LilyPad* a comunicar entre si sem fios, cada um desempenhando o papel de receptor e emissor, através do protocolo ZigBee.

Existem duas versões principais deste hardware, o *Xbee*¹ e o *Xbee Pro*¹, sendo que a antena poder estar integrada no circuito ou ser saliente. O *Xbee* comunica a uma distância de aproximadamente 122 metros sem obstáculos e o *Xbee Pro* atinge os 1600 metros.

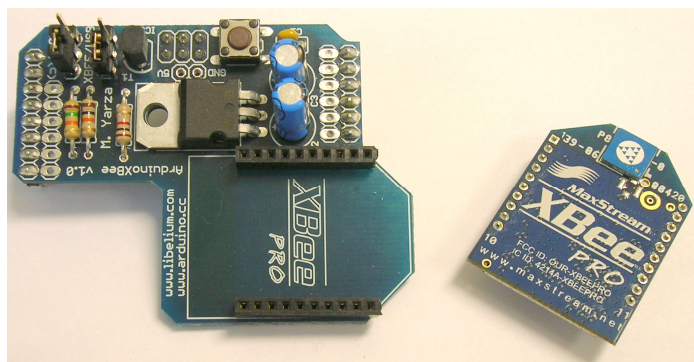


Figura 1 – Xbee Shield e Xbee Pro.

Para que o *Xbee* seja simples de ligar ao *Arduino*, foi criada uma placa própria (Figura 1), *Xbee Shield*, que encaixa perfeitamente no *Arduino*, replicando os contactos que cobre, para que continuem a poder utilizar-se. Qualquer uma das versões do *Xbee* encaixa posteriormente na placa. Assim, basta colocar a placa no modo USB para que seja possível fazer upload de código para o *Arduino*, alterando-se de seguida para o modo *Xbee*, de forma que os dois dispositivos comuniquem entre si.

¹ Digi: Making Wireless M2M Easy – XBee® & XBee-PRO® ZB ZigBee® PRO RF Modules. In: <http://www.digi.com/products/wireless/zigbee-mesh/xbee-zb-module.jsp> (2008-12-15; 1h);

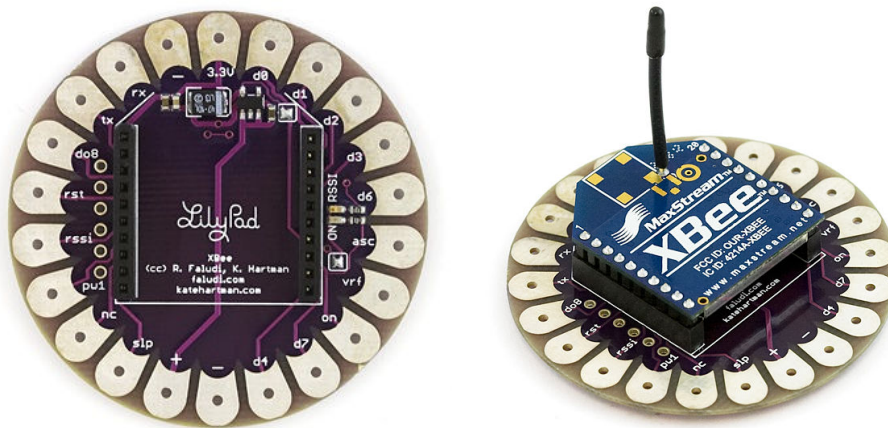


Figura 2 – LilyPad Xbee sem e com um Xbee ligado.

Para o *LilyPad* foi também criada, muito recentemente, uma versão especial própria para encaixar um *Xbee* na placa do *LilyPad*, o *LilyPad Xbee*. Usando esta nova placa, não se perdem os contactos próprios do *LilyPad*, ao contrário do que aconteceria se se ligasse um *Xbee* como se liga qualquer outro dispositivo ao *LilyPad*, que podem então ser usados para outros fins. Contudo, os contactos mudam um pouco de configuração e algumas funções são também alteradas, como podemos ver na Tabela 1.

Salienta-se que o *LilyPad Xbee* não substitui o *LilyPad Arduino*, já que não possui qualquer chip, não sendo possível programá-lo. Assim pode-se optar entre programar directamente o *Xbee* numa linguagem própria ou usar o *LilyPad Xbee* como um *shield* para ligar o *Xbee* ao *LilyPad Arduino*².

Tabela 1 – Funções de cada pin do *LilyPad Xbee*.³

Label	Description	Xbee Pin
3.3V	connect regulated power only, nominally limited to 3.4 Volts	1
—	ground	10
rx	data in	2
tx	data out	3
d08	digital output 8 (not supported yet)	4
rst	reset	5
rssi	received signal strength indicator / PWM output 0	6
pw1	PWM (pulse-width modulation) output 1	7

² Para mais informações sobre como ligar o *LilyPad Xbee* ao *LilyPad* como *shield* ver: Problemas encontrados – LilyPad Xbee;

³ FALUDI, Rob e HARTMAN, Kate – LilyPad Xbee. In: <http://lilypadxbee.katehartman.com/about-board/> (2008-11-25; 18h);

nc	no connection	8
slp	Sleep / digital input 8 / DTR	9
+	connect unregulated Voltage here	
—	ground (alternate connection)	10
d4	analog input 4 / digital input/output 4	11
d7	digital input/output 7 / CTS	12
on	ON light output	13
vrf	Voltage Reference for analog / digital inputs	14
asc	association light output / analog input 5 / digital i/o 5	15
d6	analog input 6 / digital i/o 6 / RTS	16
d3	analog input 3 / digital i/o 3	17
d2	analog input 2 / digital i/o 2	18
d1	analog input 1 / digital i/o 1	19
d0	analog input 0 / digital i/o 0	20

O código mais simples para colocar dois Xbee a comunicar entre si faz com que um deles acenda e apague um *LED* por ordem do outro, como se vê de seguida⁴:

Num dos Xbee

```
void setup()
{
  Serial.begin(9600); //velocidade de comunicação, a mesma entre os
                      //dois Xbees para que comuniquem entre si.
}

void loop()
{
  Serial.print('H'); //escreve H para a porta série.
  delay(1000); //espera 1 segundo.
  Serial.print('L'); //escreve H para a porta série.
  delay(1000); //espera 1 segundo.
}
```

No outro Xbee

```
int outputPin = 13; //variável outputPin do tipo int (inteiro) com o
                   //valor 13 atribuído, correspondente ao pin da placa
                   //Arduino.
int val; //declaração da variável val do tipo int.

void setup()
{
  Serial.begin(9600); //velocidade de comunicação, a mesma entre os
                      //dois Xbees para que comuniquem entre si.

  pinMode(outputPin, OUTPUT); // atribuição do modo output ao pin 13
                              //da placa do Arduino, definido pela
                              //variável outputPin.
}

void loop()
```

⁴ Arduino – Arduino Xbee Shield. In: <http://arduino.cc/en/Guide/ArduinoXbeeShield> (2008-11-23; 14h);

```

{
  if (Serial.available()) { //verifica se existe informação disponível
                            na porta série e se existir:
    val = Serial.read(); //lê essa informação e atribui-a à variável
                            val.
    if (val == 'H') { //verifica se a informação é igual a H e se for:
        digitalWrite(outputPin, HIGH); //envia corrente para o pin 13,
                                        acendendo o LED.
    }
    if (val == 'L') { //verifica se a informação é igual a L e se for:
        digitalWrite(outputPin, LOW); //não envia corrente para o pin
                                        13, apagando o LED.
    }
  }
}
}

```

Ambos os *Xbee* comunicam à mesma velocidade (9600 bits por segundo). Um deles envia alternadamente duas letras. Essas letras são lidas pelo outro, que executa uma acção correspondente a cada letra.

Os comandos mais usuais da comunicação em série são⁵:

`Serial.begin(speed);` //define a velocidade a que o Arduino vai enviar e receber informação. Geralmente coloca-se 9600 bits por segundo.

`Serial.print(data);` // envia informação para a porta série.

`Serial.print(data, encoding);` // o mesmo que a anterior, mas codifica a informação enviada. A informação a colocar no *encoding* é a de qual o código de codificação que se pretende usar.

`Serial.println(data);` // envia informação para a porta série, mas acrescenta “\r\n”, como se se tivesse pressionado *Return* ou *Enter* depois de escrever a informação.

`Serial.available();` // permite saber se há informação à espera de ser lida na porta série.

`Serial.read();` // lê a informação da porta série.

`Serial.flush();` // limpa a informação em espera na porta série.

⁵ BANZI, Massimo – Getting Started with Arduino. USA: O’Reilly, 2008;

Bluetooth

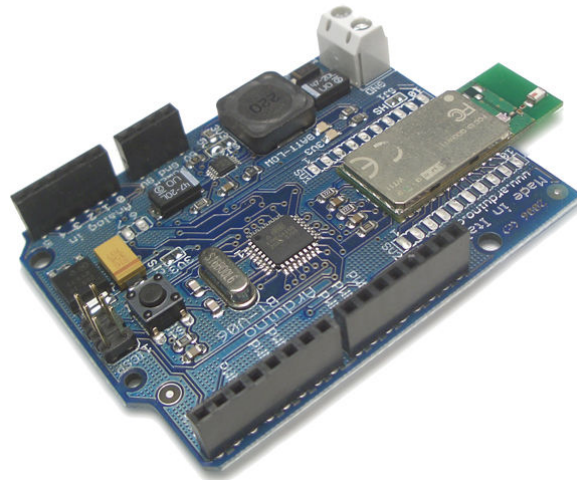


Figura 3 – Arduino BT.

Esta é a outra forma de ligar dois *Arduinos* ou *LilyPad* sem fios. Existe um *Arduino* que já possui um módulo *Bluetooth* incorporado, o *Arduino BT*⁶, que tem algumas modificações em relação ao *Arduino* normal, em particular a voltagem permitida que é reduzida a 5.5V no máximo. Possui um *surface-mounted ATmega168*, o que oferece o dobro do espaço de memória para os ficheiros criados pelo utilizador. Tem mais três *Pins PWM*, dois *Pins Analógicos* e o *pin 7* está ligado ao *reset* do dispositivo *Bluetooth*. Apenas usa a comunicação de série a *115200 baud*, sendo essa a velocidade para que foi pré-programado. O módulo Bluetooth usado é o Bluegiga WT11, versão iWrap.

Para conectar o *LilyPad* a um dispositivo *Bluetooth*, devem seguir-se os passos referidos na secção do *LiLypad*.



Figura 4 – Bluetooth Modem - BlueSMiRF Gold.

⁶ Arduino – Arduino BT. In: <http://arduino.cc/en/Main/ArduinoBoardBluetooth> (2008-12-15; 4h);

O dispositivo aconselhado por Leah Buechley⁷ é Bluetooth Modem - BlueSMiRF Gold⁸. As suas características podem ver-se na Tabela 2.

Tabela 2 – Características de Bluetooth Modem - BlueSMiRF Gold.

Tamanho	0.38 x 1.52 x 4.83 cm
Distância de transmissão	100m
Frequência	2.4~2.524 GHz
Comunicações em série	2400-115200bps
Opera em ambientes com outros sinais de RF	WiFi, 802.11g e Zigbee
Conexão encriptada	Sim
Antena incorporada	Sim
Consumo de potência	25 mA avg
Voltagem de Operação	3.3V-6V
Temperatura de Operação	-40~ +70C

⁷ BUECHLEY, Leah – LilyPad + Bluetooth. In:
<http://web.media.mit.edu/~leah/LilyPad/extend.html> (2008-12-15; 4h);

⁸ SparkFun Electronics - Bluetooth Modem - BlueSMiRF Gold. In:
http://www.sparkfun.com/commerce/product_info.php?products_id=582 (2008-12-15; 4h);