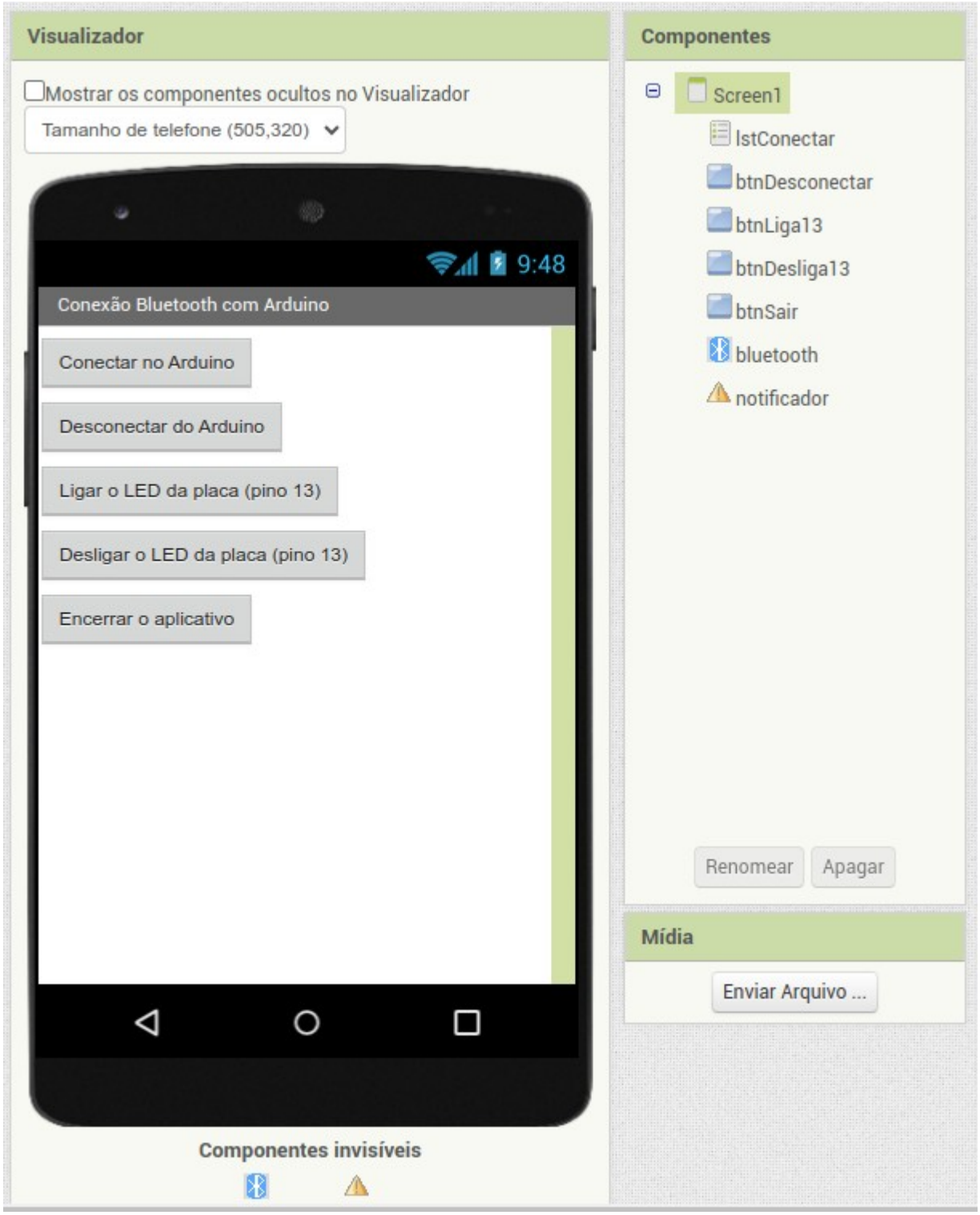
Conexão *Bluetooth* com Arduino

Desenho da aplicação no MIT App Inventor

1 – Conexão básica: o App envia comandos ao Arduino

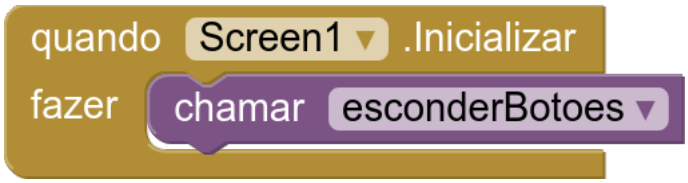


O desenho da tela do App está exibido acima e contém 4 botões (cujos ‘nomes’ podem ser vistos ao lado do visualizador, na seção de componentes), e dois componentes invisíveis: um cliente *bluetooth* e um notificador. Também há um componente ‘EscolheLista’ (lstConectar) o qual será responsável por exibir a lista de dispositivos *bluetooth* disponíveis.

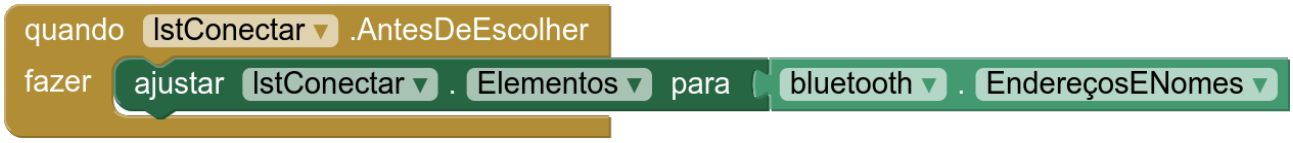
Vamos ver os blocos de código do App.

Vamos iniciar nosso App controlando as possibilidades que o usuário terá. Para isso, vamos 'desabilitar' os botões que ainda não podem ser utilizados. Para isso, vamos escrever um procedimento (se você não quer que o botão nem apareça quando não estiver habilitado ao uso, troque a propriedade '`Ativado`' pela '`Visível`'):

Depois de escrito o procedimento, vamos chamá-lo no momento em que a tela for iniciada:



Para que ocorra a conexão, deveremos selecionar o dispositivo *bluetooth* adequado. O bloco de código a seguir é capaz de exibir os dispositivos conhecidos e pareados:

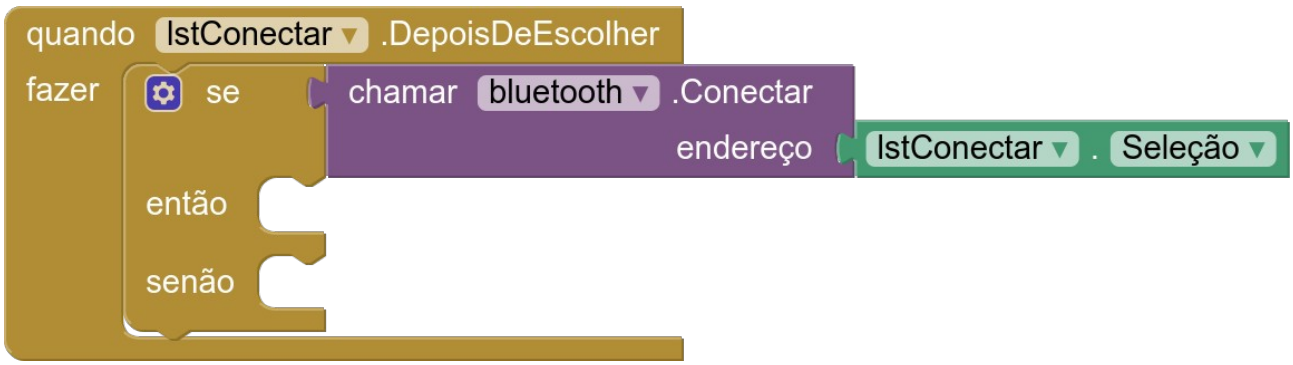


Note que:

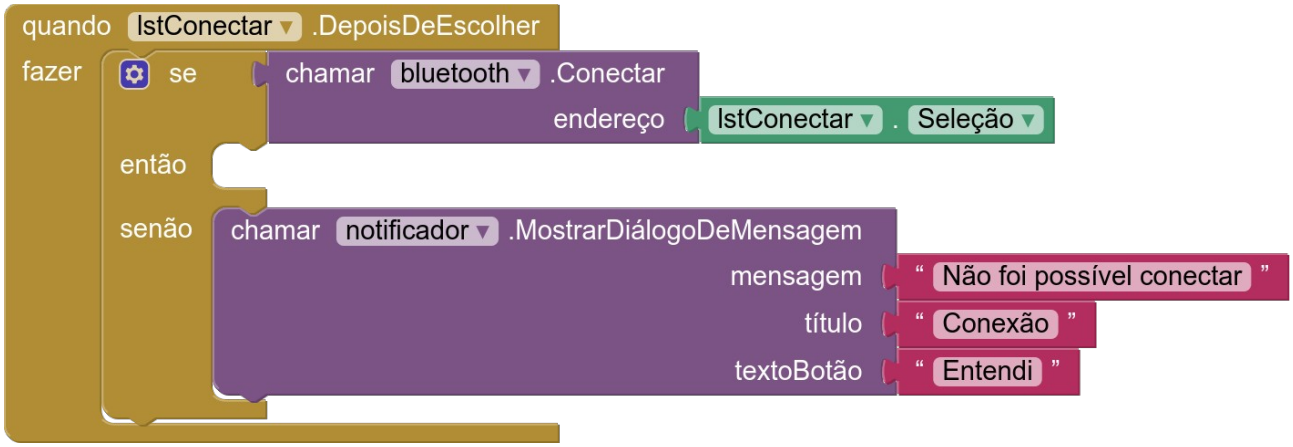
- não existe um 'clique'; o método escolhido ('`AntesDeEscolher`') será executado quando você clicar no botão de lista `lstConectar` (nome que eu escolhi, verifique em seu projeto o nome que você utilizou);
- o método escolhido ocorre ANTES que seja escolhido um dos itens da lista; ou seja: ele irá ajustar os elementos da lista a partir da lista de endereços e nomes que ele extrair do cliente *bluetooth* (para isso funcionar o bluetooth TEM QUE estar ativado e o módulo com o qual você quer se conectar TEM QUE estar pareado – nossa aplicação, pelo menos por enquanto, é simples, e não é capaz de fazer isso, ela usa o que o celular oferecer).

Agora, vejamos o que ocorre DEPOIS que uma conexão *bluetooth* for escolhida por meio do processo anterior:

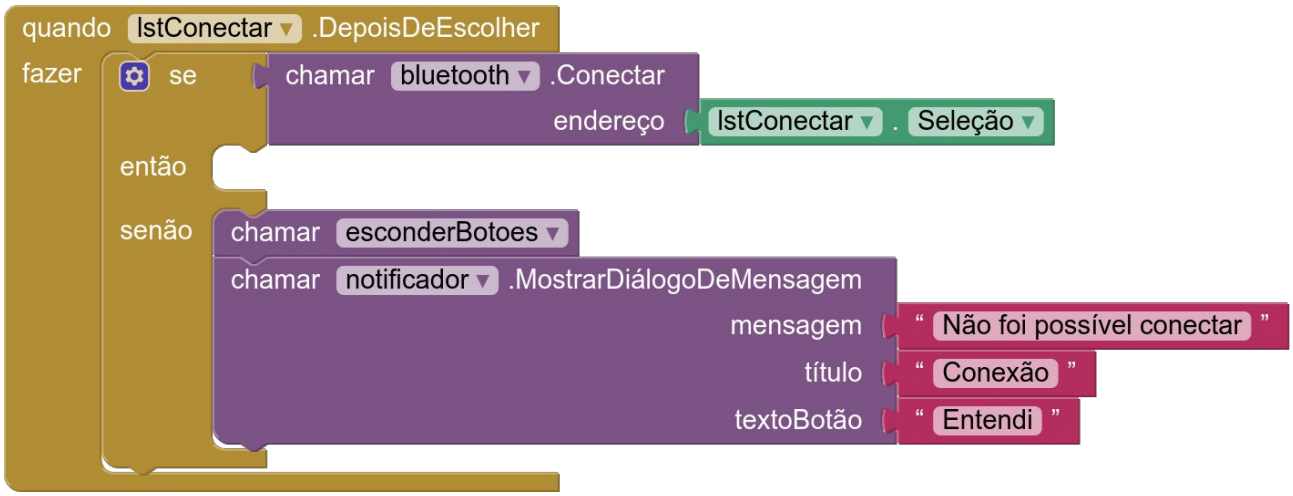
Primeiramente, note que a tentativa de conexão está inserida dentro de um bloco SE/ ENTÃO:



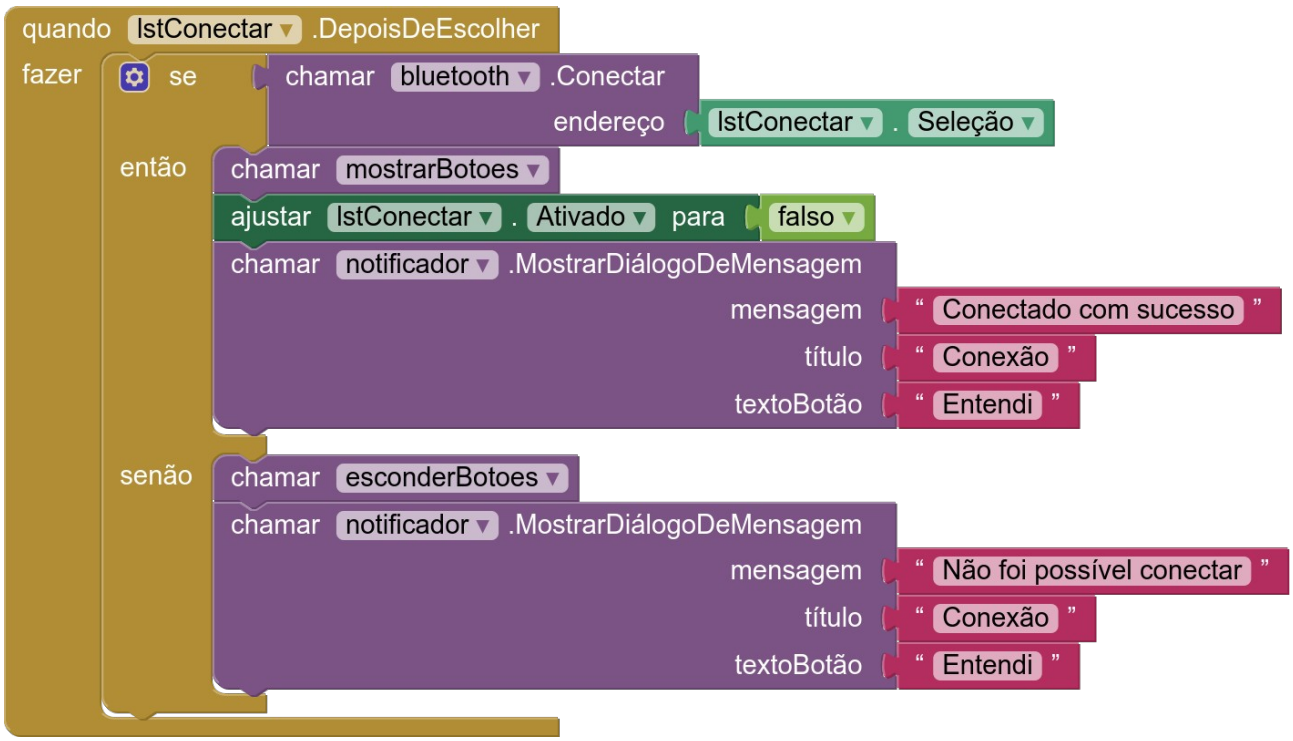
Este bloco será acionado depois (`DepoisDeEscolher`) que o usuário selecionar uma conexão *bluetooth* da lista. O que ocorre na sequência somente será executado SE ao ser chamado o cliente *bluetooth* ele conseguir se conectar com o enderço/ nome da seleção. Caso a conexão não seja possível, colocaremos uma mensagem de erro. Assim:



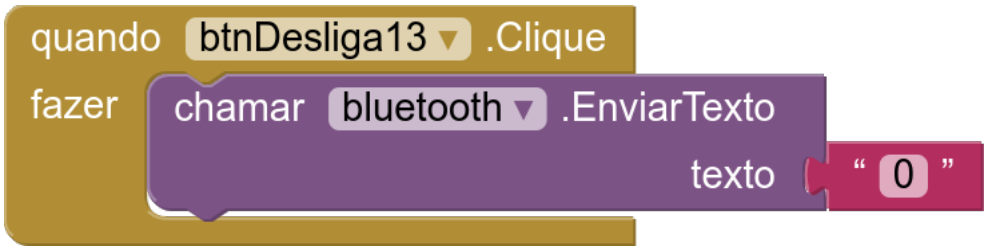
Mas, não podemos esquecer que esta conexão pode ter dado errado em uma segunda tentativa. Desta forma, é bom chamar a função de desabilitar os botões:

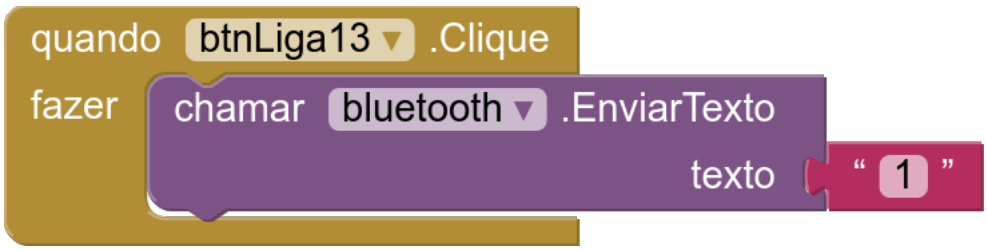


Quando ao conexão der certo, podemos realizar algumas atividades, tais como: habilitar os botões, desabilitar o botão de conexão (pois teremos que nos desconectar primeiro da conexão atual), e informar que estamos conectados:

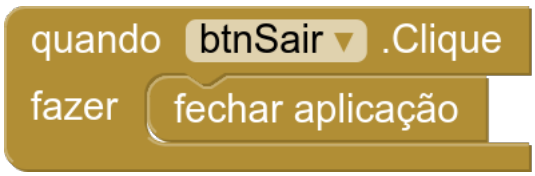


Já temos a conexão ativa, então podemos enviar comandos ao Arduino por meio dela. Tecnicamente, iremos transferir um caractere, o qual será lido pelo módulo *bluetooth* e passado ao código do Arduino para utilização. Para isso, teremos o envio de códigos a partir dos botões que controlam o LED da placa (pino 13, `LED_BUILTIN`); vamos usar o envio do caractere '`0`' para desligar o LED e o envio do caractere '`1`' para ligá-lo (o código no Arduino deve levar isso em consideração):

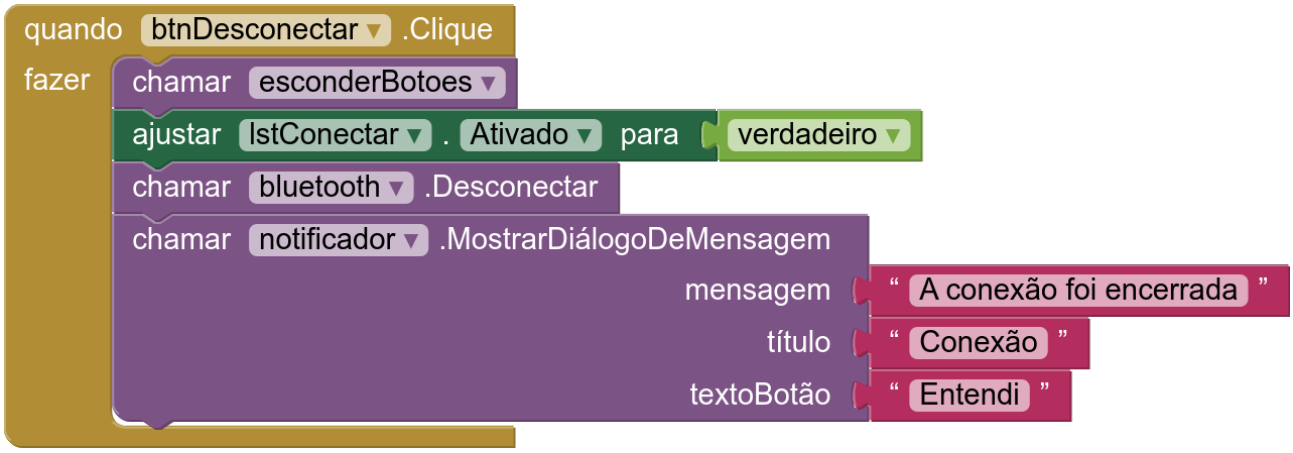




Para o botão que encerra a aplicação (`btnSair` em meu desenho), o código já é conhecido, e é:



Finalmente, vamos ver o botão de desconectar:



A função da desconexão (além da desconexão do *bluetooth*) é desabilitar os botões e informar o usuário que a conexão foi encerrada. Também será liberada ação sobre a lista de conexões conhecidas.

Código no Arduino.

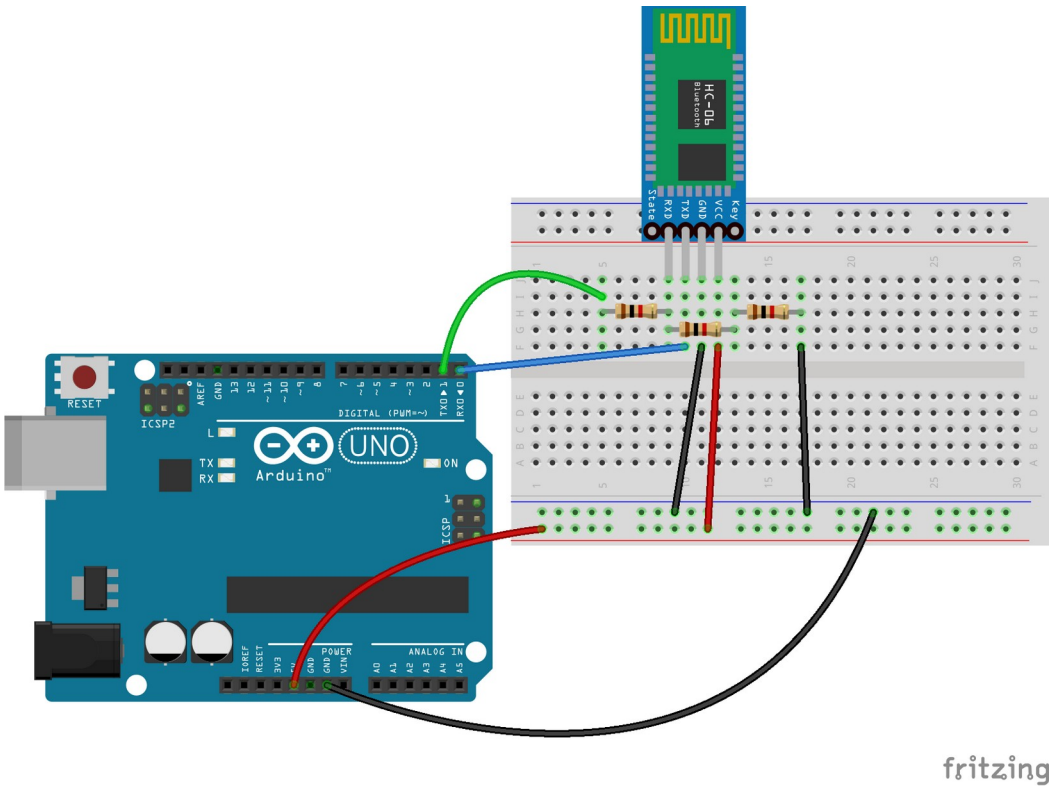
O código que segue é capaz de interpretar o caractere recebido pelo *bluetooth* e acionar o LED ligado ao pino 13 (`LED_BUILTIN`) em correspondência. Compile e envie o código ao Arduino para testar.

```
char controle = 0; //define uma variável tipo 'char' e inicializa com '0'

void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);     //este é o LED da placa, porta digital 13
  digitalWrite(LED_BUILTIN, LOW);   //inicia com o LED desligado
}

void loop() {
  if(Serial.available()) {          //se houver dados na interface serial
    controle = Serial.read();       //lê os dados da serial e coloca na variável 'controle'
  }
  if(controle == '0')               //controle é zero?
  {
    digitalWrite(LED_BUILTIN, LOW); //desliga o LED
  }
  if(controle == '1')               //controle é um?
  {
    digitalWrite(LED_BUILTIN, HIGH); //liga o LED
  }
}
```

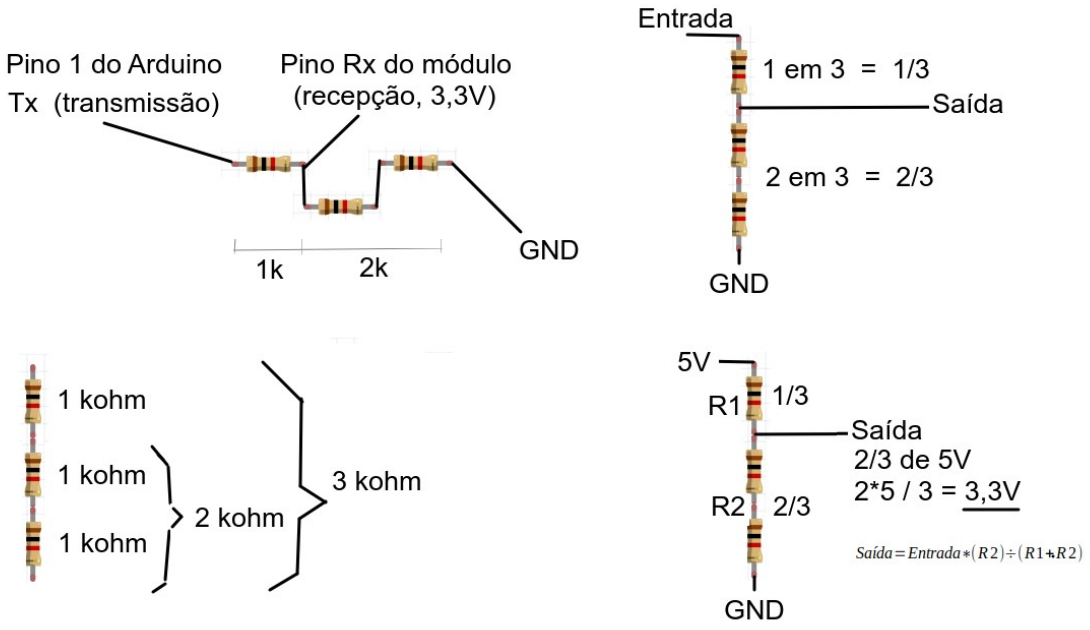
Após enviar o código ao Arduino, ligue o monitor serial e teste: digite os números `0` e `1` para desligar e ligar o LED da placa.

Desligue o cabo de conexão USB. Vamos usar a interface serial para conectar nosso módulo *bluetooth*:

- monte o circuito a seguir



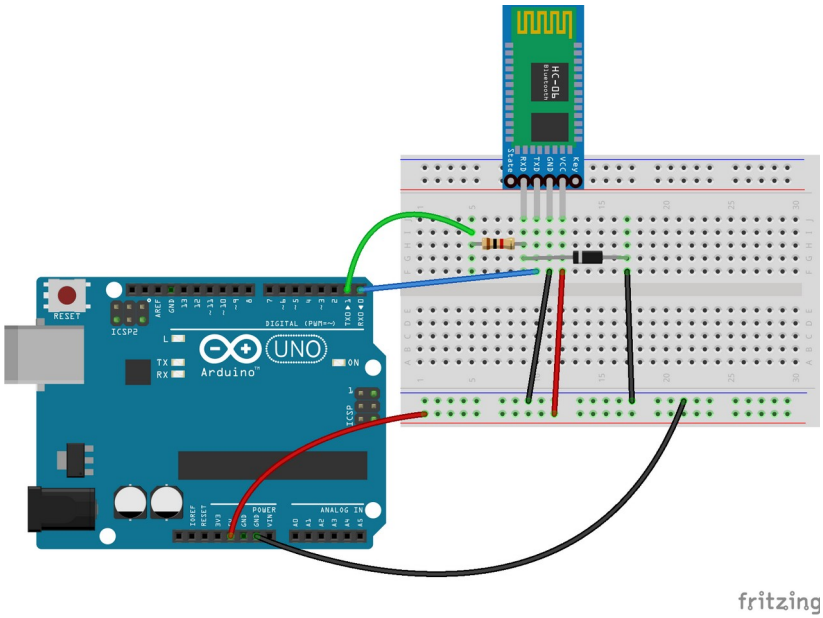
Note a existência de 3 resistores (no exemplo, de 1kohm, marrom, preto, vermelho). A finalidade é realizar um divisor de tensão: a entrada (o Tx do Arduino, pino 1) está conectada em série com o primeiro resistor (1 kohm); os outros dois resistores estão conectados em série (o que resultará em uma resistência de 1+1 = 2kohm), e ligados ao GND. A conexão com o módulo *bluetooth* em seu pino 'Rx' é realizada entre a conexão do resistor de 1kohm e o de '2 kohm', formando um divisor de tensão de razão 1/3 x 2/3:



Matematicamente, chamando o primeiro resistor de R1 e o segundo de R2, temos:

$$Saída = Entrada * (R2) \div (R1 + R2)$$

Outra forma de conectar oferecendo proteção ao módulo é usando um diodo Zener, de 3,3Volts (que nos meus módulos deve vir soldado no próprio módulo, para garantir…):



Neste caso o diodo Zener garante que a tensão não passe de 3,3Volts ao módulo (o resistor até pode ser dispensado, o Zener não).

Passadas as explicações da eletrônica, SE VOCÊ UTILIZAR ESTA CONEXÃO ELÉTRICA, UTILIZANDO OS PINOS Rx e Tx DO ARDUINO, NÃO VAI CONSEGUIR REALIZAR ATUALIZAÇÃO DO CÓDIGO ENQUANTO O MÓDULO *BLUETOOTH* ESTIVER CONECTADO. Se precisar atualizar o código, desconecte o módulo (pois a USB também usa o Tx/Rx e haverá conflito).

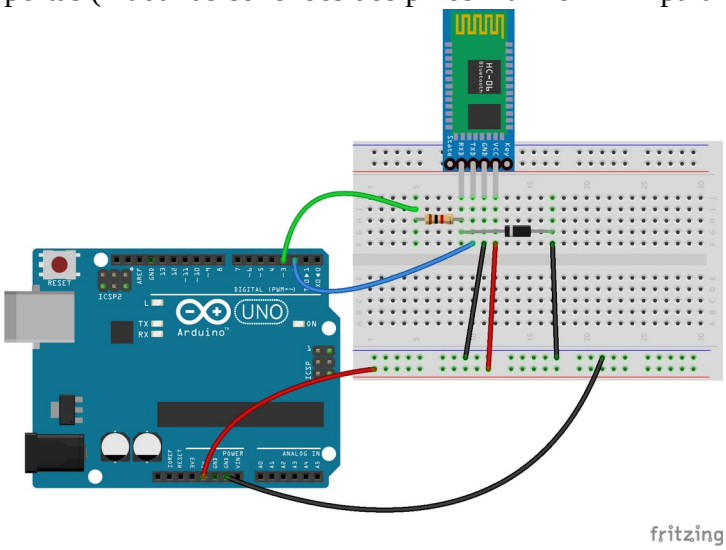
Teste:

- transfira o código ao Arduino;
- faça a conexão elétrica do módulo (NÃO esqueça dos resistores ou do Zener);
- conecte alimentação ao Arduino (pode ser a USB, mas não transmita código);
- pareie com o celular;
- inicie o App e teste suas funcionalidades.

Para não ter que desconectar a USB, você poderá criar uma instância de comunicação serial, assim:

```
1  #include <SoftwareSerial.h>          //biblioteca de comunicação serial
2  #define RX 2                         //pino 2 (será usado como RX pelo software)
3  #define TX 3                         //pino 3 (será usado como TX pelo software)
4  SoftwareSerial bluetooth(RX, TX);    //cria um objeto de comunicação serial
5  char controle = '0';                 //para armazenar o caractere recebido
6
7  void setup() {
8    Serial.begin(9600);                //inicializa o monitor serial a 9600bps
9    bluetooth.begin(9600);             //inicializa o objeto bluetooth a 9600bps
10   pinMode(LED_BUILTIN, OUTPUT);      //este é o LED da placa, porta digital 13
11   digitalWrite(LED_BUILTIN, LOW);    //inicia com o LED desligado
12 }
13
14 void loop() {
15   if(bluetooth.available()) {        //se houver dados na interface serial na qual está o bluetooth
16     controle = bluetooth.read();     //lê os dados vindos do bluetooth e coloca na variável 'controle'
17   }
18   if(controle == '0')                //controle é zero?
19   {
20     digitalWrite(LED_BUILTIN, LOW); //desliga o LED
21   }
22   if(controle == '1')                //controle é um?
23   {
24     digitalWrite(LED_BUILTIN, HIGH); //liga o LED
25   }
26 }
```

Neste caso, será necessário realizar nova conexão elétrica, ligando o módulo *bluetooth* em outras portas (mudar as conexões dos pinos `'0'` e `'1'` para `'2'` e `'3'`):



Para este exemplo, quem emula a comunicação serial é o objeto bluetooth criado na linha 4 do código, a partir da biblioteca de comunicação serial. Pode usar outros pinos (por exemplo, 7 e 8, 10 e 11, etc).

- vantagens: você pode ter mais de um objeto de comunicação serial; libera a comunicação USB para testar código sem ter que trabalhar no hardware.
- desvantagens: objetos ocupam memória; o desempenho pode não ser tão bom quanto o da interface nativa.

Teste:

- transfira o código ao Arduino;
- faça a conexão elétrica do módulo (NÃO esqueça dos resistores ou do Zener);
- conecte alimentação ao Arduino (pode usa a USB);
- pareie com o celular;
- inicie o App e teste suas funcionalidades.

Testando o módulo *bluetooth*.

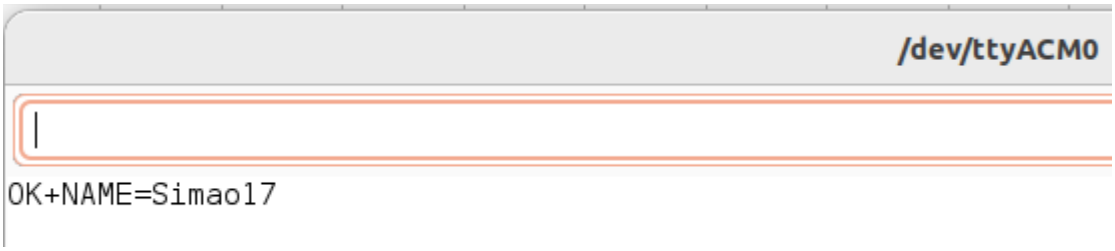
Digite, compile e envie para o Arduino o código que segue (note os pinos das ligações do módulo):

```
1  //Usa a interface serial para testar o bluetooth emulando uma serial
2
3  #include <SoftwareSerial.h> //biblioteca de comunicação serial
4
5  #define RX 2 //pino 2 (será usado como RX)
6  #define TX 3 //pino 3 (será usado como TX)
7
8  SoftwareSerial bluetooth(RX, TX); //cria um objeto de comunicação serial
9
10 void setup() {
11   Serial.begin(9600);
12   bluetooth.begin(9600);
13 }
14
15 void loop() {
16   //envia o que for lido em uma serial para a outra
17   if (bluetooth.available())
18     Serial.write(bluetooth.read());
19   if (Serial.available())
20     bluetooth.write(Serial.read());
21 }
```

Ligue o monitor serial e digite `AT` (em maiúsculo, seguido de 'enter'). Se tudo estiver certo, o módulo responderá com '`OK`'.

> NOTA: NEM TODOS os módulos aceitam todos os comandos, e existem pequenas variações entre eles. Infelizmente, você terá que testar cada um deles… Em alguns casos é necessário utilizar `AT+RESET` para que os comandos sejam realizados.

Digite `AT+NAME` (exatamente assim, `AT+NAME`, seguido de 'enter'). O módulo deve responder com o 'nome' dele (por exemplo `Simao17`, HC-05, HC-06...):



Digite `AT+PIN` para saber qual a senha. A resposta será `+PIN=0017`.

Para trocar a senha, digite `AT+PIN5678` (sem espaços, `5678` será a nova senha). A resposta deverá ser '`OksetPIN`' (e lembre-se de que esta nova senha será usada para parear, o que estiver armazenado em conexões anteriores não funcionará mais – **se trocou, retorne a senha à anterior**).

Digite `AT+PIN` para ver a nova senha. A resposta será `+PIN=5678`. O que você achou da segurança?

Pesquise os comandos '`AT`' para saber mais.

Exercícios.

1 – Elabore um App capaz de acender/ apagar um LED conectado a um dos pinos digitais (à sua escolha):

- elabore a conexão de *hardware* (lembre-se dos resistores ou Zener);
- elabore o código no Arduino;
- elabore um novo App;
- teste e comprove o funcionamento, eventualmente realizando ajustes.

2- Elabore um App capaz de enviar um caractere qualquer pelo *bluetooth*. Teste com um dos exemplos de *hardware + software* Arduino anteriores. Teste.