Módulo Bluetooth

Uma das formas de comunicação mais utilizadas em IOT para a troca de informações é o bluetooth. Ele é um tipo de sinal de rádio que permite a criação de uma rede de comunicação entre diversos dispositivos presentes dentro da área de cobertura do sinal (um raio de aproximadamente 100m). Por ser relativamente comum e muito utilizado por smartphones é uma alternativa viável e segura para estabelecer uma comunicação entre um smartphone e o nosso Arduino. Porém, diferentemente do Esp32 e de algumas outras placas especiais, o Arduino Uno (e as variantes mais comus como Mega, Nano, Leonardo, Due, etc) não conta com um bluetooth nativo e necessita de um módulo adicional que agregue a ele este recurso. Entre as diversas opções existentes no mercado, as mais conhecidas e utilizadas são as da família HC, principalmente por sua simplicidade e baixo custo. Os módulos dessa família possuem pequenas variações em seus recursos (podem funcionar em modo escravo ou modo misto, podem ter comunicação otimizada, etc), mas todas as versões possuem a implementação da comunicação básica necessária para a troca de informações e isso é o que diretamente nos interessa aqui. A comunicação bluetooth é na verdade uma comunicação Serial, porém mais segura dada a necessidade de um pareamento e de um endereçamento que identifique o dispositivo ao qual desejamos conectar. Neste livro, não é nosso objetivo o aprofundamento no funcionamento do bluetooth e no protocolo de comunicação que o sustenta e nem como a comunicação serial é implementada em outras linguagens de programação, mas sim na sua utilização prática com o Arduino. Por isso, veremos como descobrir o MacAddress de um módulo HC, realizar o pareamento e implementar o envio e recepção de mensagens via Serial no Arduino, comunicando com um app Android mobile desenvolvido através de blocos (sem programação com o AppInventor 2).

Principais métodos para trabalhar com o módulo HC

Como o módulo HC trabalha com bluetooth, baseado em comunicação Serial (na conexão estabelecida entre o módulo e o dispositivo ao qual deseja se comunicar) podemos fazer uso da biblioteca "SoftwareSerial", nativa do Arduino e que auxilia na troca de mensagens utilizando esta tecnologia. Ela nos fornece um objeto do tipo SoftwareSerial através do qual conseguimos acessar métodos para ler informações externas recebidas através da Serial ou para enviar informações através da Serial. A instanciação de um objeto desta classe é realizada como segue abaixo, informando as portas do Arduino onde o pino RX e o pino TX do módulo estão conectados:

```
SoftwareSerial <nome_objeto_serial>(<porta_pino_RX>, <porta_pino_TX>);
```

Criado o objeto, o mesmo deve ser inicializado atribuindo uma velocidade de trabalho pois trata-se de um objeto serial:

```
<nome_objeto_serial>.begin( <velocidade> );
```

Inicializado o objeto, para enviar uma String via serial basta dar um print:

```
<nome_objeto_serial>.print( <string_desejada> );
```

Já para ler dados externos que cheguem através do bluetooth (via serial), devemos criar um laço que deve ser executado enquanto o objeto perceber que ainda possui dados recebidos pendentes, concatenando-os em uma variável o resultado. Para testar se ainda existem dados recebidos (bytes) não lidos, utilizamos o método available:

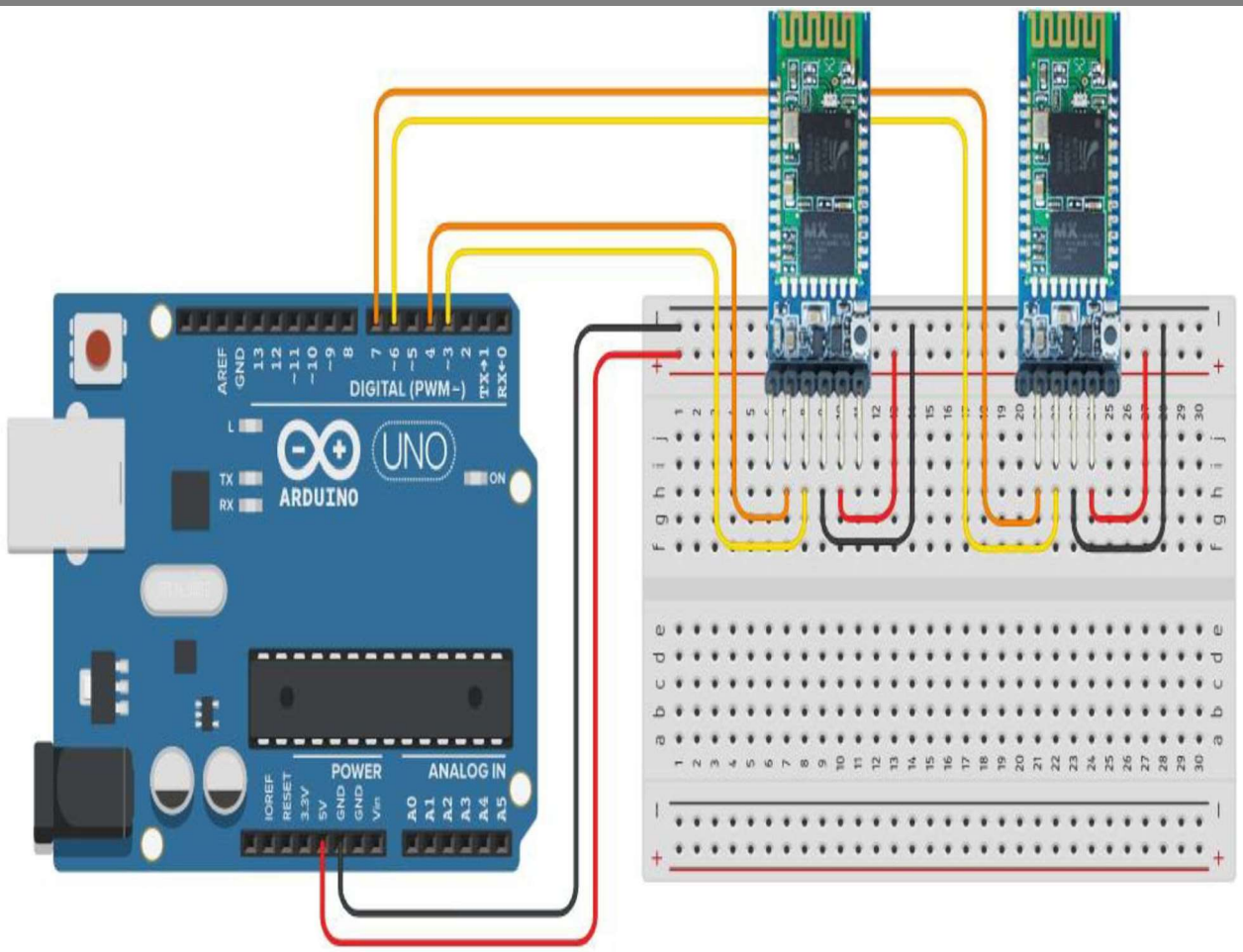
```
<nome_objeto_serial>.available()
```

Para pegar cada um destes bytes não lidos, utilizamos o método read, convertendo o byte em um caractere:

```
char(<nome_objeto_serial>.read())
```

Ligação eletrônica

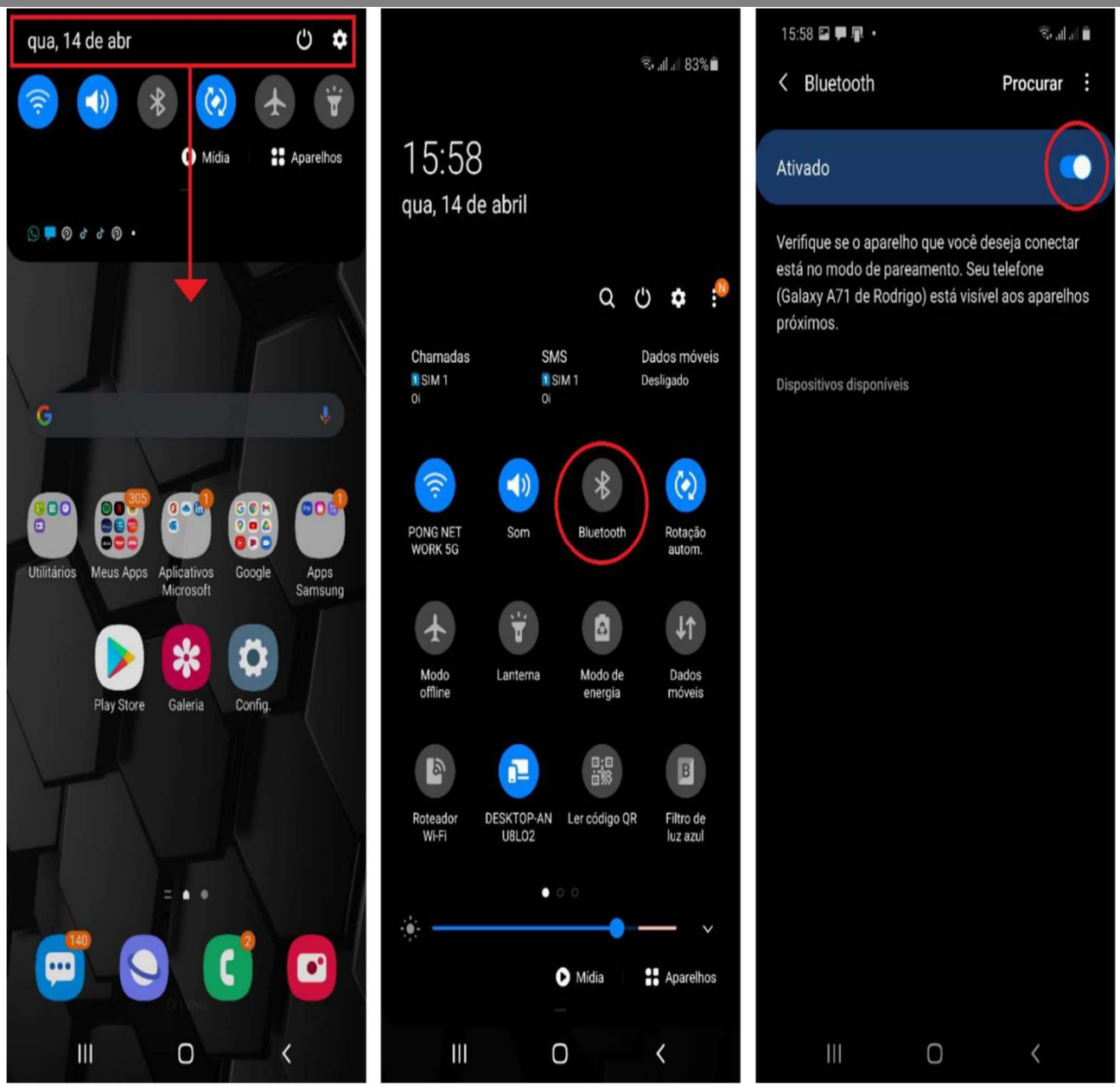
A ligação eletrônica dos módulos da família HC é bastante simples. Não importa se o modelo é do tipo mestre/escravo (6 pinos, como o HC05) ou somente do tipo escravo (4 pinos, como o HC06), os 4 pinos que nos interessam para o estabelecimento de uma comunicação são: VCC, GND, TX e RX (sendo estes dois últimos ligados a qualquer porta digital do Arduino). Assim, o nosso esquema eletrônico de ligação será similar ao apresentado abaixo, de acordo com o número de pinos do módulo (6 como no à esquerda e 4 como no à direita):



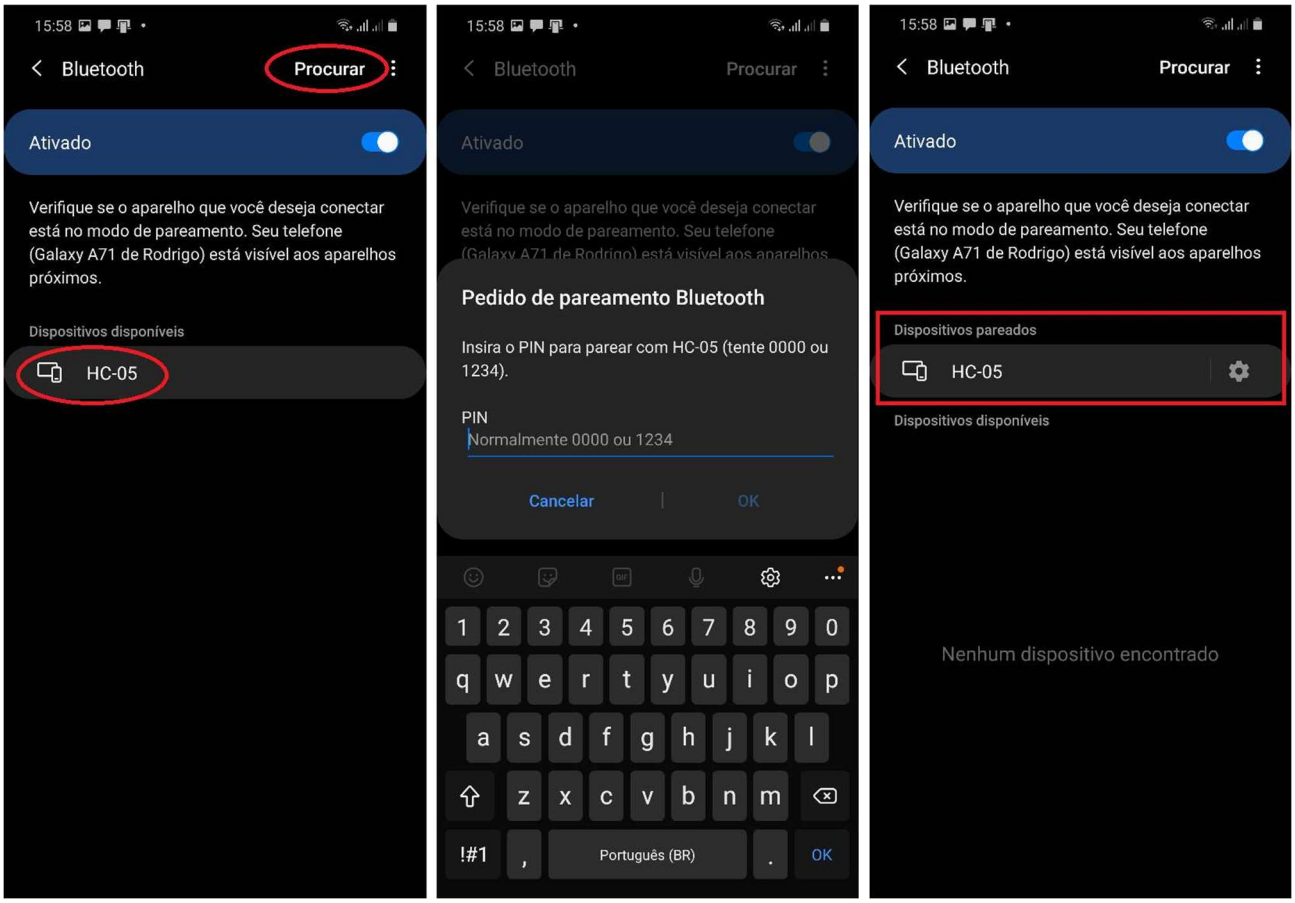
Pareamento do HC com um Smartphone Android

Assim como todo dispositivo bluetooth, antes de ser acessado e utilizado em conjunto com um smartphone, o módulo HC deve ser descoberto e pareado, criando assim uma conexão segura. O processo para essa descoberta e o pareamento, pode diferir um pouco de smartphone para smartphone, de acordo com a versão do sistema operacional (Android no nosso caso) e das mudanças de interface da fabricante. Porém, apesar de pequenas diferenças, o processo será sempre muito similar ao mostrado a seguir.

Primeiramente, devemos tocar na barra superior do Android, segurar e arrastar para baixo, para acessar os atalhos rápi dos. Depois, ao encontrar a opção "Bluetooth", é necessário pressionar e segurar por alguns segundos para que as opções de configuração sejam abertas. Ao abrir, caso o bluetooth esteja desativado, devemos ativar.

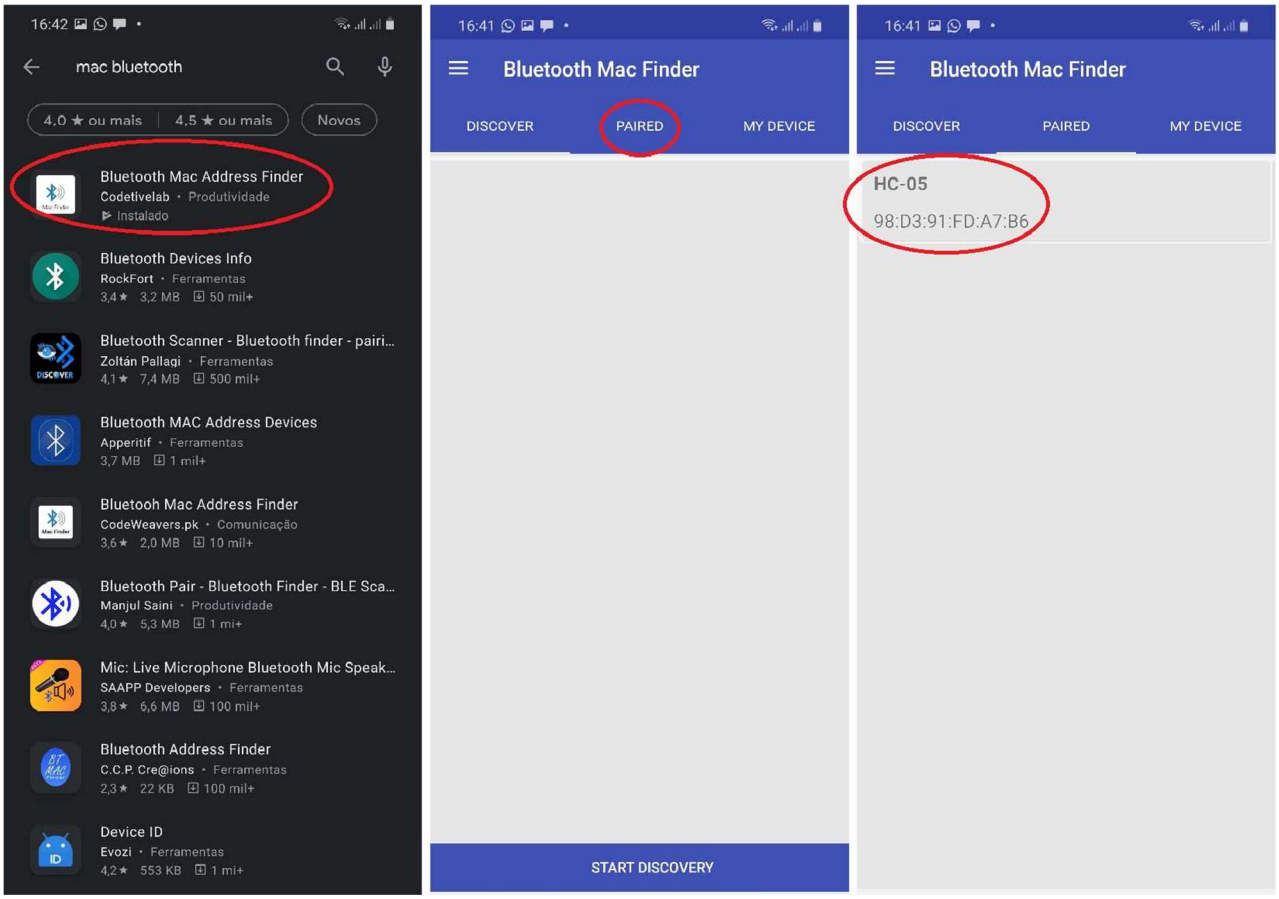


Após, um clique na opção "Procurar" fará com que o smartphone identifique todos os dispositivos bluetooth próximos a si e caso o HC estiver ligado, aparecerá na lista de dispositivos disponíveis. Basta agora clicar sobre o HC e será aberta uma caixa de diálogo solicitando uma senha de pareamento. Devemos então digitar a senha default (1234) e o dispositivo realizará o pareamento, estabelecerá a conexão e passará a ser exibido na lista de dispositivos pareados.



Descoberta do Mac Address do módulo HC

Em diversas linguagens de programação, ao tentar estabelecer uma conexão bluetooth com um dispositivo devemos obrigatoriamente informar o MAC Address do mesmo. O Mac Address é um endereço físico que identifica cada dispositivo em uma rede de forma única, sendo necessário para direcionar a troca de mensagens. Existem diversos apps para Android que auxil iam nessa tarefa. Um deles é o "Bluetooth Mac Address Finder", que pode ser baixado e instalado gratuitamente. Ao abrir o app, percebemos 3 guias na parte superior, sendo que a inicial é a "DISCOVER". Como nosso módulo HC já está pareado (não precisa ser descoberto), basta clicar em "PAIRED", localizar o nosso HC na lista de dispositivos pareados (no exemplo abaixo, só existe ele na lista) e o endereço Mac estará visível abaixo do seu nome.



Exemplo prático: criando app em Android integrado com Arduino

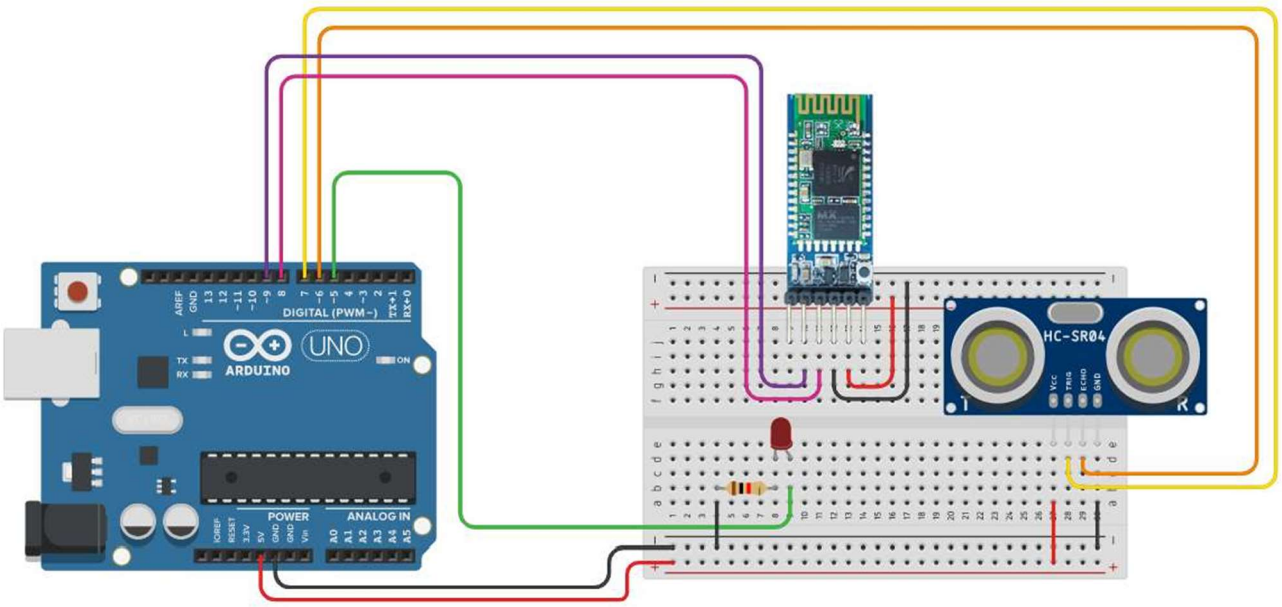
Para demonstrar o funcionamento prático da comunicação bluetooth entre o Arduino (através do módulo HC) e outro dispositivo, iremos desenvolver um aplicativo Android para smartphone que possa receber o valor de distância lida por um sensor ultrassônico do Arduino e mostrar na tela em tempo real (comunicação HC/Arduino → Android) e que possua na mesma tela um botão através do qual possa ordenar ao Arduino o ligamento e desligamento de um led quando pressionado (comunicação Android → HC/Arduino). O fundamental para nós neste livro é o entendimento do código do lado do Arduino e não outras linguagens de programação, como já citado anteriormente. Então para facilitar o desenvolvimento do lado do Android, iremos criar o App através do App Inventor 2, serviço online do MIT que permite o desenvolvimento através do arrastar de componentes e de blocos de função (sem a necessidade de programação). Caso você não conheça o App Inventor e nunca tenha desenvolvido aplicativos com linguagens de blocos (como por exemplo o Scratch), recomendamos que previamente assista a série de vídeos do excelente curso de App Inventor para Android, produzido pelo professor Flávio

Guimarães no canal "Brincando com ideias". O curso completo consiste de 36 aulas que vão do básico até a integração com banco de dados e o desenvolvimento de um sistema de automação residencial completo. Porém, os 5 primeiros vídeos da série (links abaixo) trazem todos os conceitos básicos necessários para compreender a ferramenta e começar a produzir os primeiros aplicativos Android, auxiliando no entendimento da combinação de blocos que será criada neste nosso exemplo.

Links dos 5 primeiros vídeos da série:

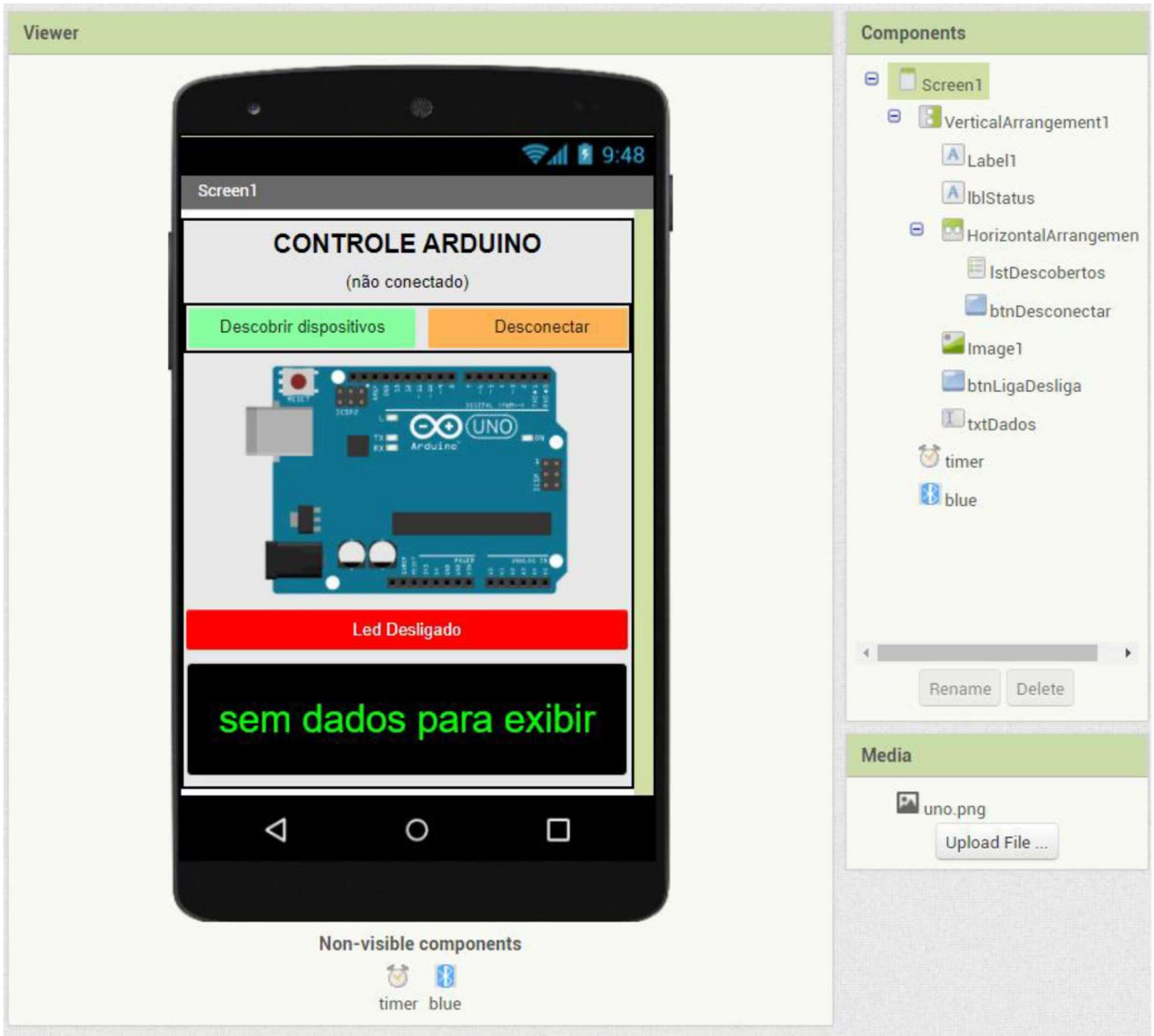
https://youtu.be/TKPXS7V1YLo
https://youtu.be/1gOx1HoAkkY
https://youtu.be/ePyUC23z0gc
https://youtu.be/DWRGfa21DBk
https://youtu.be/cOVu015orxQ

Vamos iniciar o projeto pelo seu esquema eletrônico. Utilizaremos um módulo HC-05 (lembrando que o modelo não importa pois para comunicação todos possuem os protocolos desejados) com seus pinos TX e RX ligados, respectivamente às portas 8 e 9 do Arduino. Para testar o envio de dados do Arduino para exibição no app Android, um sensor ultrassônico será incluído, com seus pinos Trigger e Echo ligados, respectivamente, às portas 6 e 7 do Arduino. Já para testar o envio de comandos do app Android para o Arduino, um led é suficiente e o seu pino positivo será conectado à porta 5 da placa. Deste modo, teremos o seguinte esquemático eletrônico:



No App Inventor, o layout do aplicativo de nosso exemplo (que pode ser visto abaixo, com a configuração visual à esquerda e a árvore de componentes à direita) contará com 5 elementos visuais importantes:

- Um Label (lblStatus) que informará o status da conexão bluetooth;
- um ListPicker (lstDescobertos) que mostrará uma lista de todos os dispositivos bluetooth descobertos pelo smartphone para que possamos selecionar o HC;
- um Button (btnDesconectar) que encerrará qualquer conexão bluetooth ativa;
- um Button (btnLigaDesliga) que quando pressionado enviará ao Arduino, de forma alternada, um aviso para ligar ou desligar o led;
- um TextBox (txtDados) que mostrará em tempo real a distância do ultrassônico recebida do Arduino através da conexão bluetooth;

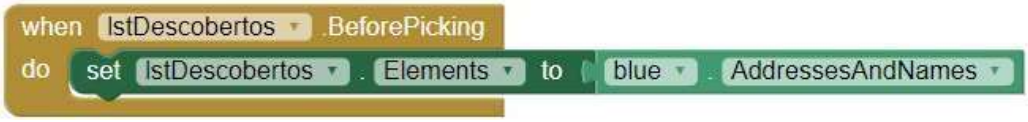


Além deles, outros dois elementos não visuais são necessários para a implementação desejada do aplicativo:

- um BluetoothClient (blue) responsável por acessar o serviço bluetooth do Android, estabelecer e controlar a conexão;
- um Clock (timer) que a cada um segundo se encarregará de receber e mostrar na tela o valor da distância recebida do Arduino através da conexão bluetooth.

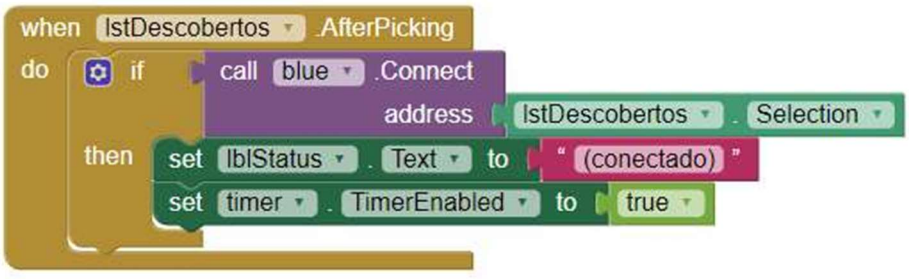
Quanto a programação dos blocos, vamos dividi-la em cinco partes independentes para um melhor entendimento.

Blocos parte 1) Listando os dispositivos bluetooth descobertos pelo smartphone



Ao clicarmos no ListPicker *lstDescobertos*, antes da lista ser exibida (bloco when/BeforePicking) devemos setar os seus itens. Isso é feito através do bloco set/Elements da própria lista, que deve receber por parâmetro uma lista de itens a exibir. O que desejamos mostrar são todos os dispositivos bluetooth descobertos pelo smartphone. Para isso, utilizamos o bloco AdressesAndNames do nosso BluetoothClient *blue* que retorna o MacAdress e o nome de identificação de cada um desses dispositivos.

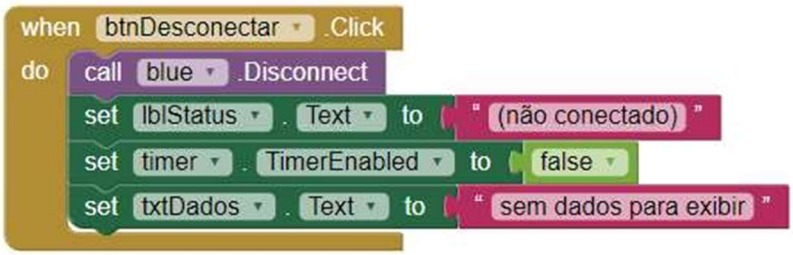
Blocos parte 2) Estabelecendo uma conexão bluetooth com o dispositivo selecionado



Ao fecharmos o ListPicker *lstDescobertos* (bloco when/AfterPicking), pegamos o dispositivo selecionado na lista (através do seu bloco Selection) e passamos o mesmo como parâmetro do bloco call/ConectAddress, para que o BluetoothClient *blue* tente estabelecer uma conexão com ele. Caso a conexão seja estabelecida com sucesso é retornado verdadeiro e assim entramos no bloco then de nosso condicional if. Nele, modificamos o

texto do Label *lblStatus* para "(conectado)" (através do seu bloco set/Text) e habilitamos o Clock *timer* para que a cada 1 segundo receba o valor da distância do ultrassônico vinda do Arduino. Para habilitar, basta utilizar o bloco set/TimerEnabled passando por parâmetro o valor true.

Blocos parte 3) Desconectando do dispositivo bluetooth



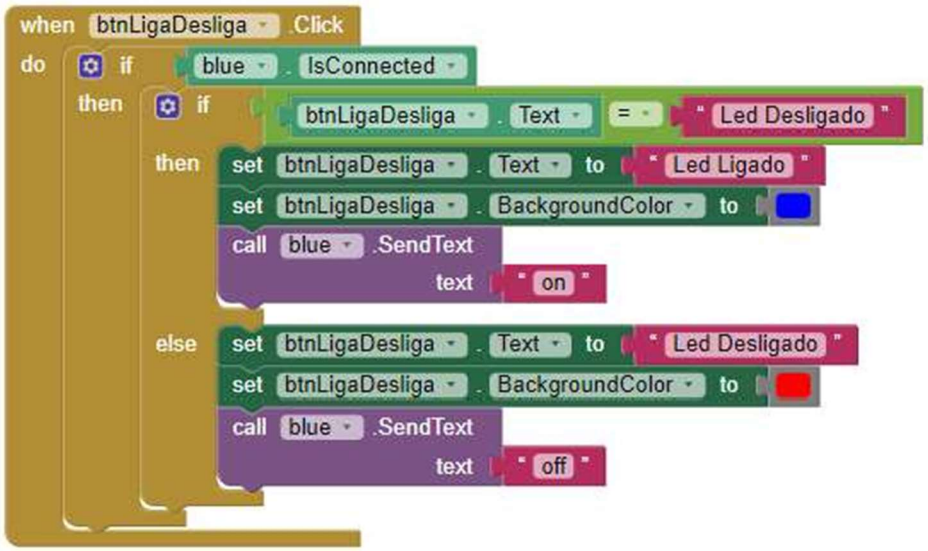
Ao clicarmos no Button *bntDesconectar* (bloco when/Click), encerramos a conexão através do bloco call/Disconect de nosso BluetoothCliente *blue*. Após encerrada, modificamos o texto do Label *lblStatus* para "(não conectado)" e o texto do TextBox *txtDados* para "sem dados para exibir" (através dos seus respectivos blocos set/Text), voltando ao estado inicial do aplicativo antes da conexão. Além disso, o Clock *timer* deve ser desativado pois como não estamos mais conectados ele não receberá mais dados de distância do ultrassônico para exibir. Fazemos isso através de seu bloco set/TimerEnabled passando por parâmetro true.

Blocos parte 4) Exibindo o valor recebido do Arduino a cada 1 segundo (distância)

Ao inserir o Clock *timer*, seu TimerInterval pode ser definido e em nosso exemplo, utilizamos o valor 1000 (1 segundo). Assim, a cada ciclo de tempo (bloco when/Timer) devemos verificar se recebemos alguma coisa do Arduino através da conexão. O componente BluetoothClient *blue*, através do bloco call/BytesAvailableToReceive nos informa o número de bytes recebidos através da conexão e não utilizados. Então, dentro de um bloco if/then, podemos testar se esse número de bytes recebidos e não usados é maior do que zero, o que significa que temos uma nova informação de distância do ultrassônico. Então, neste caso, retiramos do buffer de recebimento de informações do bluetooth *blue* (através do bloco call/ReceiveTextNumberOfBytes) o mesmo número de

bytes que descobrimos ter recebido anteriormente (bloco call/BytesAvailableToReceive) e alteramos o texto do TextBox txtDados para essa sequência de bytes capturada (bloco set/text), mostrando o valor da distância recebida.

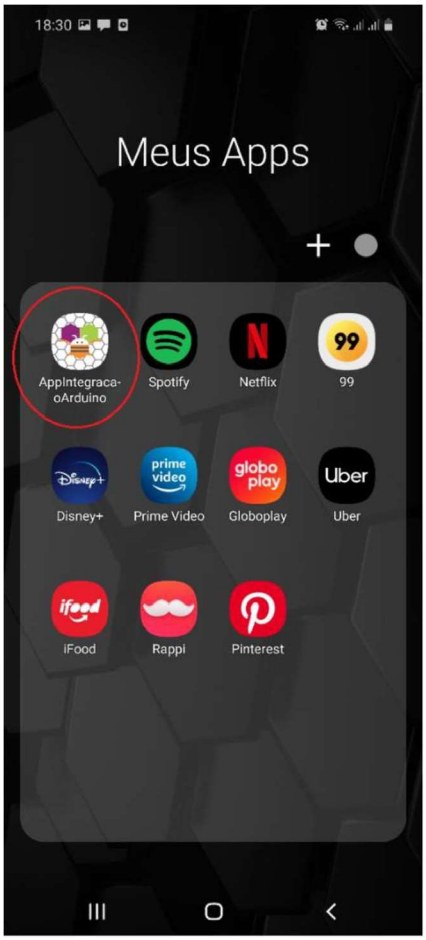
Blocos parte 5) Enviando uma mensagem ao Arduino para ligar ou desligar o Led

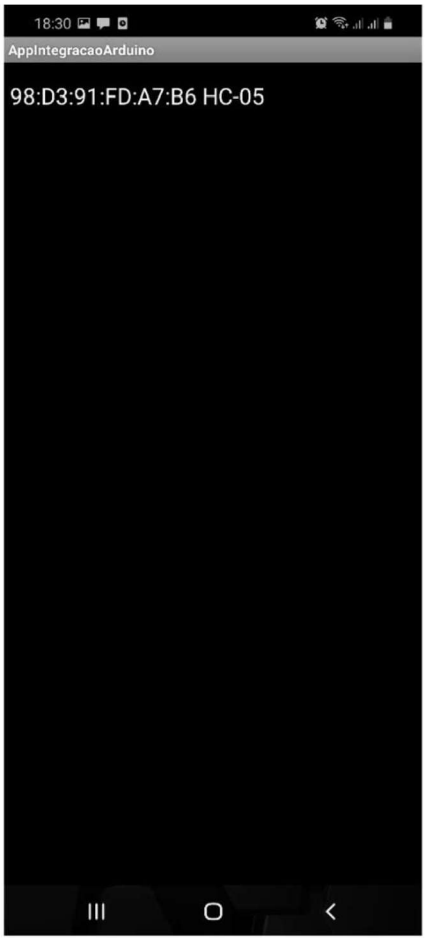


Ao clicar no Button btnLigaDesliga (bloco when/Click), primeiramente testamos em um bloco if/then se o nosso BluetoothClient *blue* está conectado, através do seu bloco IsConnected (pois caso não esteja, não faz sentido tentar enviar alguma coisa ao Arduino). Caso esteja, devemos verificar qual o estado do Button *btnLigaDesliga*, pois o mesmo botão será utilizado para duas ordens diferentes, enviadas de forma alternada. Se o seu texto for igual à "Led Desligado" (bloco Text) quando clicado, significa que devemos ligar o Led. Então, modificamos o texto do botão (bloco set/Text) para "Led Ligado", sua cor para azul (bloco set/BackgroundColor) e enviamos através do BluetoothCliente *blue* (bloco call/SendText) a String "on" para o Arduino, indicando ao mesmo que o led deve ser ligado. Já se texto do *btnLigaDesliga* não for igual à "Led Desligado" (bloco Text) quando clicado é porque é igual a "Led Ligado", o que significa que devemos desligar o Led. Então, modificamos o texto do botão (bloco set/Text) para "Led Desligado", sua cor para vermelho (bloco set/BackgroundColor) e enviamos através do BluetoothCliente *blue* (bloco call/SendText) a String "off" para o Arduino, indicando ao mesmo que o led deve ser ligado.

Concluído o aplicativo, o mesmo já pode ser gerado e instalado no smartphone da forma mais conveniente (lembrando que, conforme os vídeos do uso básico do App Inventor, temos a opção de compilação USB direta com instalação automática, geração de apk com download e instalação manual ou geração de QR code com download e instalação

manual). Abaixo, três imagens que mostram, em sequência, o ícone do app instalado no Smartphone, a tela do app após sua abertura e a lista de dispositivos bluetooth descobertos ao clicar no ListPicker (neste caso, apenas o módulo HC-05 utilizado no nosso exemplo):





Agora, finalmente vamos realizar a programação do nosso Arduino Uno para que ele funcione em conjunto com o app Android criado, enviando a ele através da conexão bluetooth as distâncias lidas pelo sensor ultrassônico e recebendo dela, também através da conexão bluetooth, indicativo de quando ligar ou desligar o led (através das palavras "on" e "off"). Além da importação das bibliotecas (Ultrasonic.h e SoftwareSerial.h), da criação dos objetos indicando as portas (conforme explicado anteriormente no esquema eletrônico) e a inicialização das Seriais (entre elas o bluetooth), a lógica é extremamente simples.

No Loop, a cada 1 segundo, testamos dentro de um while se existe algo a receber pela conexão bluetooh (objeto blue). Se sim, enquanto houverem bytes recebidos, os mesmos são concatenados em uma variável de texto (a String *recebido*), um a um. Ao final,

quando saímos do laço, temos dentro desta variável o texto recebido byte a byte de forma serial. Então, basta testar qual é este texto. Em nosso exemplo, se ele for a palavra "off" (enviada pelo aplicativo Android) o led deve ser desligado através da mudança de estado da porta 5 para LOW. Se a palavra for "on", o led deve ser ligado através da mudança de estado da porta 5 para HIGH. Além disso, a cada 1 segundo, descobrimos a distância do objeto mais próximo através da chamada do método Ranging (CM) do objeto ultrassom, armazenando este valor na variável *distancia*. Em seguida, através do objeto blue, enviamos de forma serial com o método println o valor desta mesma variável, através da conexão bluetooth, para que chegue ao app Android.

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

Ultrasonic ultrassom(7,6);
SoftwareSerial blue(8,9);

void setup(){
  Serial.begin(9600);
  blue.begin(9600);
  pinMode(5, OUTPUT);
  digitalWrite(5, LOW);
}

void loop() {

  delay(1000);

  //Testando o recebimento de dados através do bluetooth
  String recebido = "";
  while(blue.available()){
    recebido += char(blue.read());
  }

  //Executando o que foi recebido (caso algo tenha sido recebido)
  if(recebido == "on"){
    digitalWrite(5, HIGH);
  }
  if(recebido == "off"){
    digitalWrite(5, LOW);
  }

  //Obtendo e enviando informação da distância através do bluetooh
  float distancia = ultrassom.Ranging(CM);
  blue.println(distancia);

}
```

Com tudo pronto, conseguimos agora através do app Android selecionar o módulo HC, estabelecer uma conexão bluetooth entre o smartphone o Arduino, visualizar na tela a distância medida pelo ultrassônico e ordenar o ligamento/desligamento do led.

Abaixo, o print da tela do aplicativo funcional (recebendo uma distância de 73cm do ultrassônico) e uma foto do circuito montado com o led ligado através do clique do botão: