Ação: Programação com Arduinos

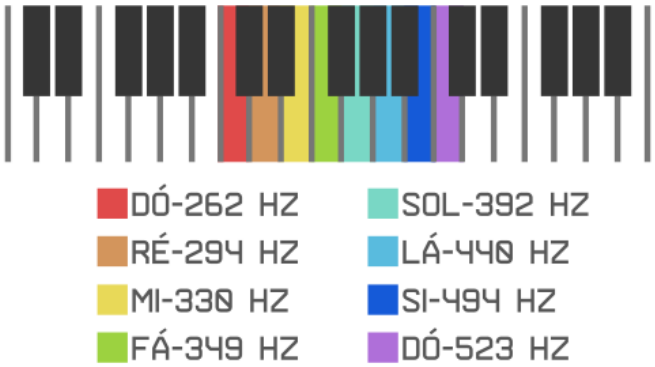
4ª sessão - Introdução à programação com Arduino

Abril, 2021

Atividade 8 – Piano (Buzzer)

Descrição: Nesta atividade pretendem-se reproduzir notas musicais, utilizando um buzzer. Utilizando botões de pressão iremos simular um piano, em que cada botão permite acionar uma nota musical diferente.

Na música, cada nota musical possui uma frequência específica em hertz (Hz). Na figura seguinte, é possível observar a frequência de cada uma das notas de uma escala musical.



Buzzer

O Buzzer é um componente sonoro, frequentemente utilizado em dispositivos eletrónicos. Geralmente os kits Arduino trazem dois buzzers quase iguais na aparência, mas na realidade são diferentes. O buzzer ativo, vem muitas vezes, com um autocolante no topo que pode ser removido e é fechado na parte inferior. O buzzer passivo não tem o autocolante e tem os fios condutores a descoberto, na parte inferior.

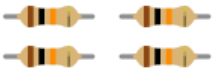
O buzzer ativo contem um oscilador interno e emite um som, desde que seja alimentado com energia. Só consegue produzir um som com uma frequência especifica.

O buzzer passivo necessita de um oscilador externo para que se consiga emitir som, mas pode ser controlado para produzir sons com diferentes frequências.

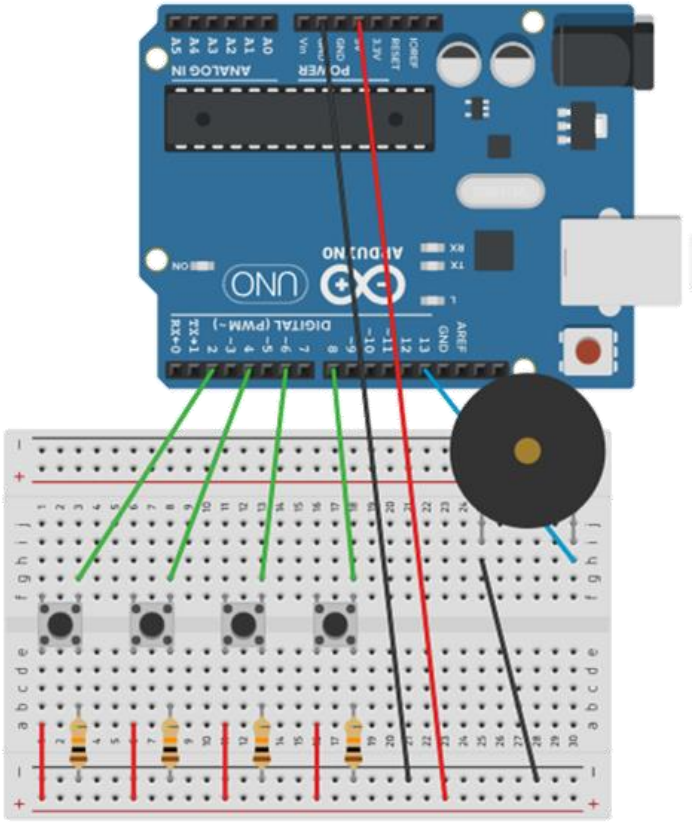
Quando proceder à montagem do circuito, tenha em atenção que este componente possui polaridade.

Comece por selecionar os componentes de acordo com a lista de materiais. Monte o circuito conforme representado na figura seguinte e efetue a respetiva programação.

Material necessário:

- **4 Botões de pressão** 
- **1 Buzzer** 
- **4 Resistências 10kΩ** 
- **Fios** 

Esquema do circuito:



Programação:

1º - Vamos começar por definir as variáveis constantes para cada pino digital do circuito. Para tal, sugere-se a criação de um array para representar os pinos digitais referentes aos botões de pressão e uma variável inteira para o pino digital referente ao buzzer.

```
const int btnPin[] = {2, 4, 6, 8};
const int buzzerPin = 13;
```

Para além disso, será necessário definir um array de 4 posições com os valores de frequência (notas musicais) associados ao som de cada botão:

```
const int tones[] = {262, 294, 330, 349};
```

2º - Na função **"setup"** será necessário escrever o código que permita configurar os pinos onde estão ligados os botões de pressão, como pinos de entrada (INPUT) e o pino do buzzer, como pino de saída (OUTPUT):

```
for (i = 0; i < 4; i++) {
  pinMode(btnPin[i], INPUT);
}
pinMode(buzzerPin, OUTPUT);
```

3º - Na função **"loop"**, será necessário obter o estado de cada botão, utilizando a função **digitalRead**. A função **tone** permite ativar um som no buzzer e a função **noTone** permite desativar o som do buzzer. Através do estado do botão (premido 1; não premido 0) e utilizando a estrutura condicional **if** é possível determinar qual o botão que foi pressionado e assim ativar o som no buzzer, de acordo com a frequência correspondente:

```
for (i = 0; i < 4; i++) {
  if (digitalRead(btnPin[i])) {
    tone(buzzerPin, tones[i]);
    delay(150);
    noTone(buzzerPin);
  }
}
```

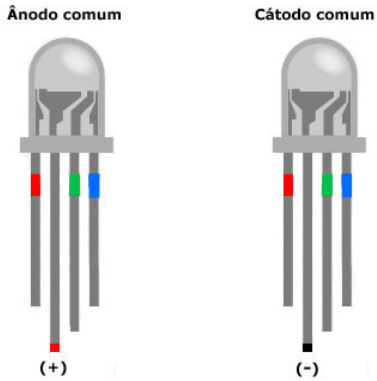
Código Final:

```
const int btnPin[] = {2, 3, 4, 5};
const int buzzerPin = 8;

const int tones[] = {262, 294, 330, 349};
int i;

void setup() {
  // put your setup code here, to run once:
  for (i = 0; i < 4; i++) {
    pinMode(btnPin[i], INPUT);
  }
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  for (i = 0; i < 4; i++) {
    if (digitalRead(btnPin[i])) {
      tone(buzzerPin, tones[i]);
      delay(150);
      noTone(buzzerPin);
    }
  }
}
```

Atividade 9 – Efeito de cores com Led RGB

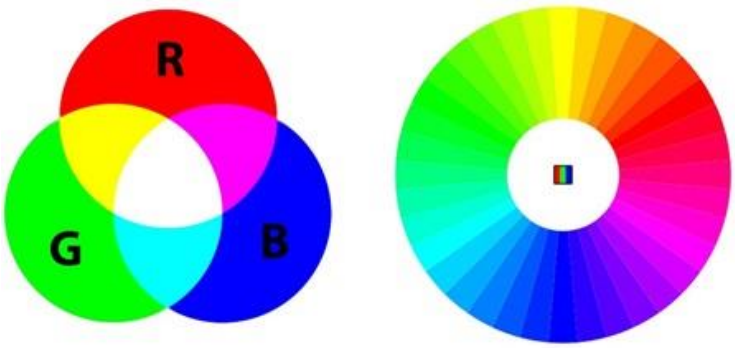
Descrição: Nesta atividade pretende-se construir um circuito que permita controlar um led RGB. As cores do led RGB serão geradas aleatoriamente, de meio em meio segundo.

O led RGB tem 3 leds integrados que podem emitir luz vermelha, verde e azul, respetivamente. Como estão muito próximos uns dos outros, os nossos olhos veem o resultado da combinação das cores em vez das três cores individuais. Possui quatro terminais, em que o terminal mais longo é a porta comum. Existem 2 tipos de leds RGB, o ânodo comum e o cátodo comum. Para cada uma das cores existe um terminal e o terminal mais longo deverá ser conectado ao polo positivo (ânodo comum) e ao polo negativo (cátodo comum).



Para se distinguir um led RGB de cátodo comum de um led RGB de ânodo comum, podemos utilizar um multímetro em modo de tensão contínua. Ao colocarmos a ponta vermelha do multímetro no terminal do led mais longo e a ponta preta num dos outros terminais, se o led acender, significa que este é um led de ânodo comum. Ao colocarmos a ponta preta do multímetro no terminal do led mais longo e a ponta vermelha num dos outros terminais, se o led acender, significa que este é um led de cátodo comum.

Quando se combinam as três cores primárias (vermelha, verde e azul) de luz, com brilho diferente, é possível produzir quase toda a luz visível.



O Arduino permite controlar o LED para emitir um total de 256 (0-255) brilhos diferentes por PWM. Assim, através da combinação das três cores diferentes, o led RGB pode emitir 256^3 = 16777216 cores, cerca de 16 milhões de cores.

Comece por selecionar os componentes de acordo com a lista de materiais. Monte o circuito conforme representado na figura seguinte e efetue a respetiva programação. Sempre que possível utilize as cores sugeridas para os fios.

Material necessário:

- **1 Led RGB (Ânodo comum)** 
- **3 Resistências**  **220 Ω**
- **Fios** 

Esquema do circuito:

Programação:

1º - Será necessário definir três variáveis constantes correspondentes aos terminais do led RGB que estarão ligados às portas digitais PWM:

```
#define ledPinR 11   //Vermelho
#define ledPinG 9   //Verde
#define ledPinB 10    //Azul
```

2º - Na função **"setup"** será necessário escrever o código que permita configurar os pinos onde estão ligados os terminais do led RGB, como pinos de saída (OUTPUT):

```
pinMode(ledPinR, OUTPUT);
pinMode(ledPinG, OUTPUT);
pinMode(ledPinB, OUTPUT);
```

3º - Na função **"setup"** adicionar a função randomSeed para inicializar o gerador de números pseudoaleatórios, de forma a começar num ponto arbitrário de uma sequência aleatória. Esta sequência, embora seja muito longa e aleatória, é sempre a mesma.

Se é importante que uma sequência de valores gerados pela função random() seja diferente em execuções subsequentes de um código, então podemos utilizar a função randomSeed() para inicializar o gerador de números aleatórios, com uma entrada significantemente aleatória, como por exemplo através da função analogRead() num pino desconectado.

```
randomSeed(analogRead(A0));
```

4º - Na função **"loop",** utilizando a função **analogWrite**, é necessário atribuir um valor entre 0 e 255 a cada uma das cores do led RGB. Utilizando a função random é possível atribuir um valor aleatório entre um valor mínimo (0 - inclusive) e um valor máximo (256 - exclusive).

```
analogWrite(ledPinR, random(0, 256));
analogWrite(ledPinG, random(0, 256));
analogWrite(ledPinB, random(0, 256));
```

Código Final:

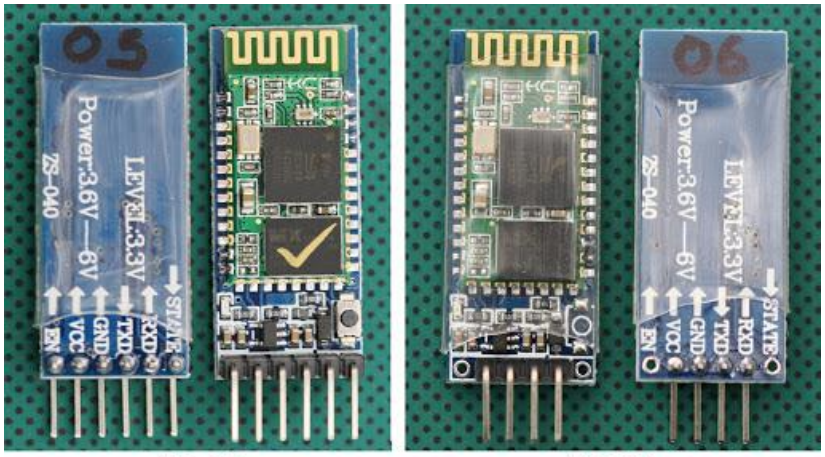
```
#define ledPinR 11  //Vermelho
#define ledPinG 9  //Verde
#define ledPinB 10   //Azul

void setup() {
  pinMode(ledPinR, OUTPUT);
  pinMode(ledPinG, OUTPUT);
  pinMode(ledPinB, OUTPUT);
  randomSeed(analogRead(A0));
}

void loop() {
  analogWrite(ledPinR, random(0, 256));
  analogWrite(ledPinG, random(0, 256));
  analogWrite(ledPinB, random(0, 256));
  delay(500);
}
```

Atividade 10 – Controlo de Led RGB por Bluetooth

Descrição: Nesta atividade pretende-se construir um circuito que permita modificar a cor de um led RGB, utilizando uma aplicação móvel e um módulo Bluetooth.

Módulo Bluetooth

Os módulos Bluetooth HC-05 e o HC-06 são os módulos mais comuns, utilizados para realizar a comunicação Bluetooth com um Arduino. São baratos e fáceis de utilizar.



HC-05 HC-06

A tecnologia Bluetooth consiste num protocolo para comunicação de rádio para utilização pessoal. É uma especificação de rede sem fios classificada como PAN (Personal Area Network). Foi desenvolvida inicialmente em 1994, pela Ericsson, como uma alternativa sem fios ao protocolo RS-232. Têm como alcance máximo, cerca de 60 metros.

O módulo Bluetooth HC-05 funciona tanto em modo master, como em modo slave, o que significa que, além de poder receber conexões, também é capaz de gerar conexões com outros dispositivos Bluetooth. Já o módulo HC-06 apenas funciona em modo slave, ou seja, apenas pode receber uma conexão Bluetooth.

Para a montagem do circuito, apenas vamos utilizar os pinos RX e TX, +5V e GND. Os pinos RX e TX são os canais de comunicação do módulo. O RX do módulo deve ser ligado ao TX do Arduino e o TX do módulo deve ser ligado ao RX do Arduino.

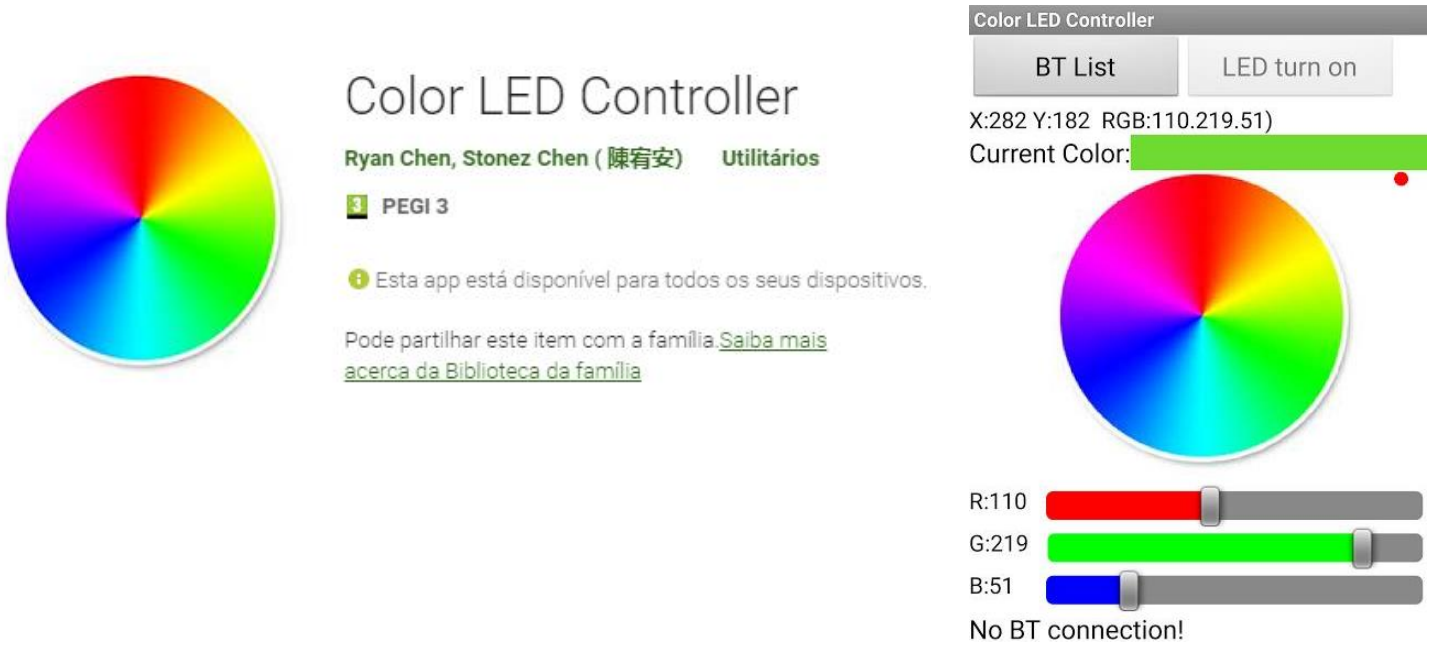
A tensão de comunicação nos módulos Bluetooth é de 3.3 V, ou seja, apesar de a alimentação ser 5V, os pinos RX e TX utilizam sinais de 3.3V para a comunicação. Desta forma, é necessário utilizarmos divisor de tensão para obter 3.3 V a partir da saída de 5V do TX do Arduino.

O pino EN/Key é utilizado para acionar o modo de comandos AT do módulo Bluetooth. Os comandos AT são um conjunto de comandos, utilizados para configuração do módulo, que podem ser enviados para o módulo, através do monitor série. O pino State está conectado ao led integrado e pode ser utilizado para verificar se o Bluetooth está a funcionar corretamente.

Aplicação Móvel (Android)

A aplicação móvel utilizada para a realização deste projeto apenas funciona em Android e está disponível em:

https://play.google.com/store/apps/details?id=appinventor.ai_yuanryan_chen.BT_LED



Antes de se começar a utilizar a aplicação, após a respetiva instalação, é necessário emparelhar o módulo Bluetooth com o smartphone. O código pin solicitado por defeito é 1234. Este código pode ser modificado através de comandos AT. Depois de emparelhado, é necessário escolher o módulo Bluetooth ao qual se pretende efetuar uma conexão. Esta configuração é efetuada através do botão BT List.

Sempre que se selecionar uma cor na aplicação, será enviada para o Arduino uma sequência idêntica à seguinte, através de Bluetooth:

252.74.255)

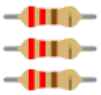
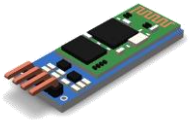
Cada cor, correspondente ao led RGB, estará separada por um ponto final:

Vermelho.Verde.Azul)

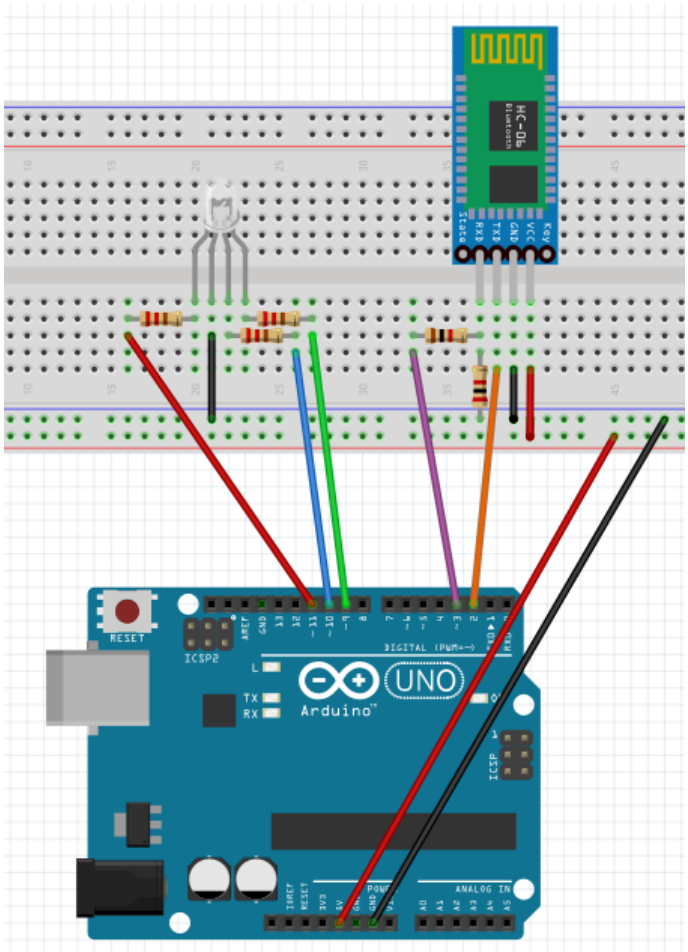
O parêntese) representa o terminador de cada uma das sequências de cores enviadas pela aplicação. A aplicação envia cerca de cinco a seis sequências, para cada cor selecionada pelo utilizador, pelo que devemos considerar apenas a última.

Comece por selecionar os componentes de acordo com a lista de materiais, monte o circuito conforme representado na figura seguinte e, por fim, efetue a respetiva programação.

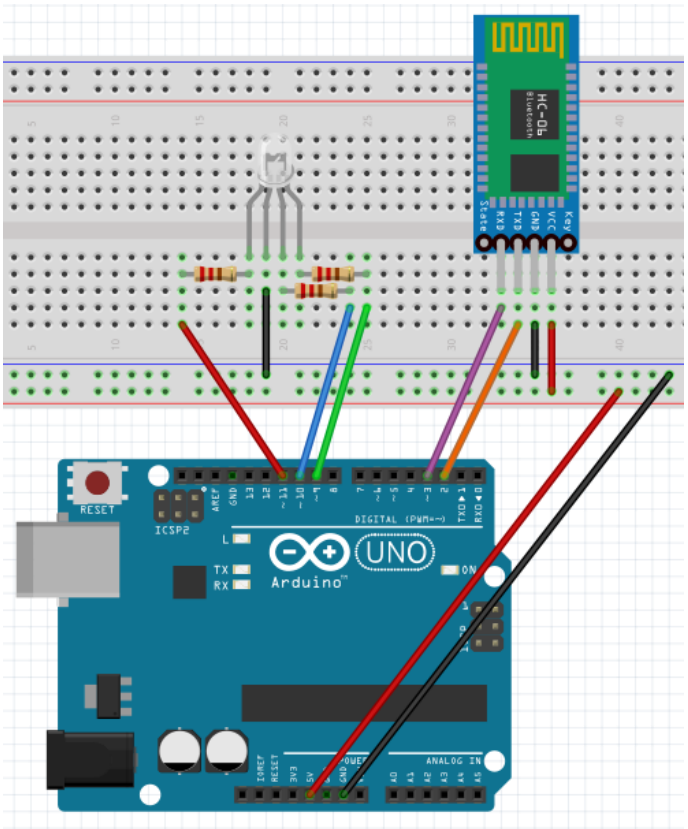
Material necessário:

- **1 Led RGB (Ânodo comum)** 
- **3 Resistências 220 Ω** 
- **1 Resistência 1kΩ** 
- **1 Resistência de 2kΩ** 
- **1 Módulo Bluetooth** 
- **Fios** 

Esquema do circuito:



Circuito alternativo:



Obs: Caso não tenham as resistências de 1kΩ e 2kΩ efetuem a montagem do circuito alternativo, embora seja recomendável aplicar um divisor de tensão à saída Tx do Arduino.

Programação:

1º - Incluir a biblioteca SoftwareSerial:

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

2º - Criar o objeto mySerial com a configuração da porta Rx do Arduino (2) e da porta Tx do Arduino (3).

```
SoftwareSerial mySerial(2, 3); // RX, TX
```

3º - Definir três variáveis constantes correspondentes aos terminais do led RGB que estarão ligados às portas digitais PWM:

```
#define redPin 11
#define greenPin 9
#define bluePin 10
```

4º - Declarar três variáveis inteiras para guardar os valores correspondentes às cores (vermelho, verde e azul) enviadas pela aplicação:

```
int red, green, blue;
```

5º - Na função **"setup"** será necessário escrever o código que permita configurar os pinos onde estão ligados os terminais do led RGB, como pinos de saída (OUTPUT):

```
pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);
```

6º - Na função **"setup"**, inicializar a comunicação série entre o Monitor Série e o Arduino e inicializar a comunicação série entre o Arduino e o módulo Bluetooth:

```
Serial.begin(9600);
mySerial.begin(9600);
```

7º - Na função **"loop"**, obter a sequência de cores enviada pela aplicação móvel. Cada sequência de cores vem no formato **Vermelho.Verde.Azul)** pelo que é necessário extrair os valores correspondentes às cores:

```
while (mySerial.available() > 0) {
  red = mySerial.readStringUntil('.').toInt();
  green = mySerial.readStringUntil('.').toInt();
  blue = mySerial.readStringUntil(')').toInt();
}
```

Como a aplicação envia cerca de cinco a seis sequências, para cada cor selecionada pelo utilizador, devemos considerar apenas a última. Isto é conseguido através do ciclo de repetição **While**. Através da função readStringUntil(terminator) é possível extrair os valores correspondentes às cores. No entanto, estes são devolvidos sob o formato de String, pelo que é necessário convertê-los para números inteiros, através da função toInt().

8º - Atribuir as cores extraídas da aplicação, ao led RGB. Não esquecer que as portas onde estão ligados os terminais do led RGB são portas PWM, pelo que podemos usar a função analogWrite para lhes atribuir um valor entre 0 e 255:

```
analogWrite(redPin, red);
analogWrite(greenPin, green);
analogWrite(bluePin, blue);
```

Estas linhas de código podem estar dentro de uma função que recebe como parâmetros, os valores das cores recebidas pela aplicação móvel e que as atribui ao led RGB:

```
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Se pretenderem, podem ir imprimindo os valores das cores, obtidos através da aplicação móvel, no monitor série:

```
Serial.print("Red: ");
Serial.print(red);
Serial.print(" Green: ");
Serial.print(green);
Serial.print(" Blue: ");
Serial.println(blue);
```

Código Final:

```
#include <SoftwareSerial.h>
#define redPin 11
#define greenPin 9
#define bluePin 10

SoftwareSerial mySerial(2, 3); // RX, TX
int red, green, blue;

void setup() {

  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  Serial.begin(9600);
  mySerial.begin(9600);
}

void loop() {
  while (mySerial.available() > 0) {
    red = mySerial.readStringUntil('.').toInt();
    green = mySerial.readStringUntil('.').toInt();
    blue = mySerial.readStringUntil(')').toInt();
  }
  setColor(red, green, blue);
  Serial.print("Red: ");
  Serial.print(red);
  Serial.print(" Green: ");
  Serial.print(green);
  Serial.print(" Blue: ");
  Serial.println(blue);
}
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```