Conversão de base

Anderson Iago Merten[1]
Igor Fachini[1]

[1]Bacharelado em Sistemas de Informação – Católica de Santa Catarina (UNERJ)
R. dos Imigrantes, 500 – 89254-430 – Jaraguá do sul – SC – Brasil

igor.fachini@unerj.br, anderson.merten@unerj.br

Abstract. *This article aims to explain how to convert decimal base to binary base, as well as demonstrate how the communication was made between Android and Ardunino so that the conversion result can be displayed on LEDs.*

Resumo. *Este artigo tem como objetivo explicar como funciona a conversão de base decimal para base binária, bem como demonstrar como foi feita a comunicação entre Android e Ardunino, para que o resultado da conversão possa ser exibido em LEDs.*

1. Convertendo bases

1.1. O projeto

O experimento consiste em um aplicativo para celular android, com programação feita em java, onde o usuário vai inserir um número em base decimal, esse número é convertido em base binária, octal e hexadecimal, ao mesmo tempo que exibida na tela do smartphone. Em conjunto a isso, será enviado o número já convertido em binário via bluetooth para um microcontrolador Arduíno, que controlará uma sequencia de 8 LEDs, ligando os LEDs que representam os ”1”do número binário e apagando os que representam números ”0”.

1.2. As conversões

Para as conversões foram utilizadas funções pré definidas do java que serão melhor explicadas posteriormente. Essas funções são:

- Integer.toBinaryString()
- Integer.toOctalString()
- Integer.toHexString()

2. Android

A aplicação foi feita em android ,usando a plataforma de desenvolvimento android, ”Android Studio”disponível no site `https://developer.android.com/studio/index.html?hl=pt-br`.

2.1. Criando a aplicação

Com o Android Studio instalado, vá em File,New, New Project. De um nome para sua aplicação no campo ’Application name’, e na próxima janela, iremos compilar para telefones ou tablets, certifique que a opção minima de sdk(Versão do android) esteja selecionada 4.0.1, já que os recursos que iremos utilizar estão disponíveis dessa versão em

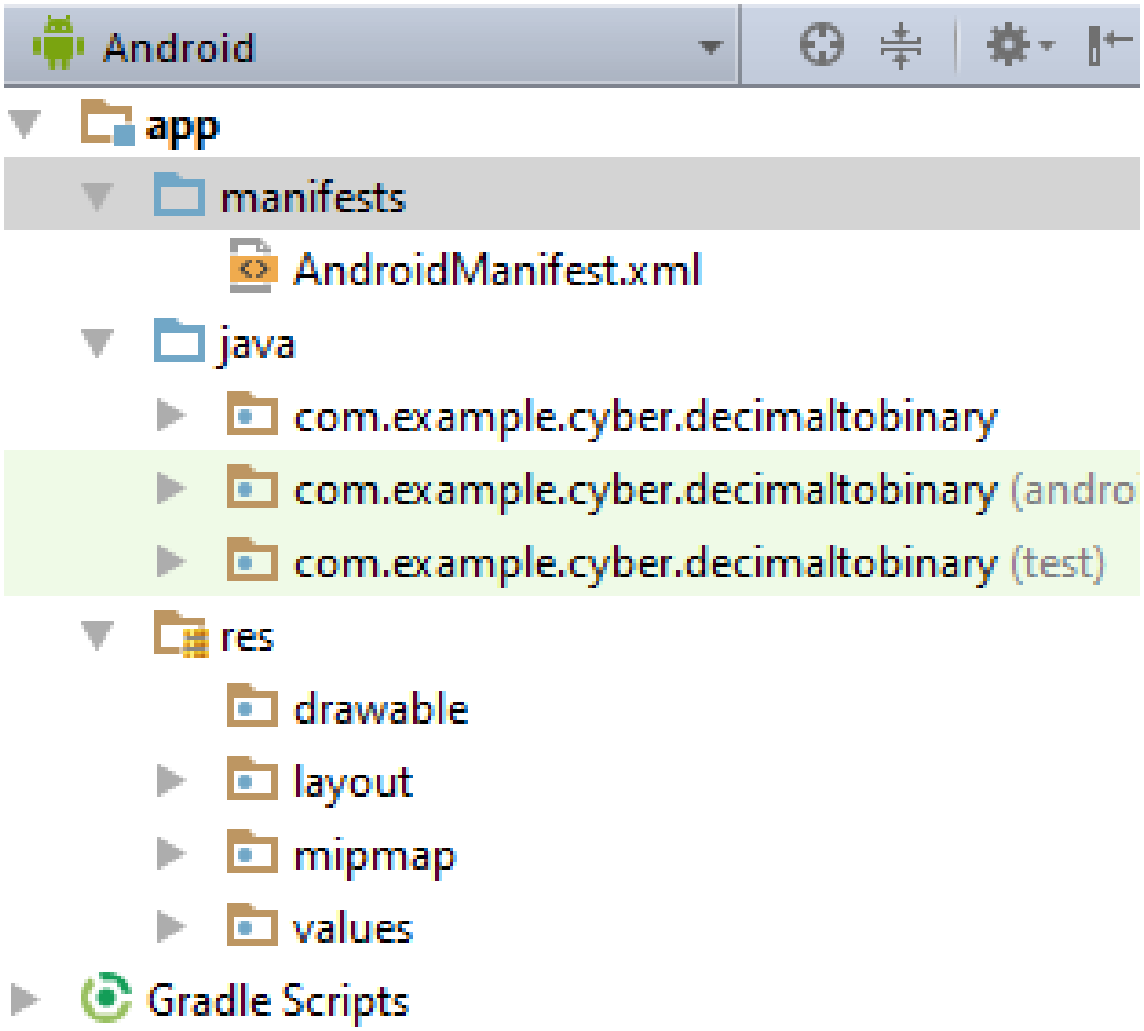

Figura 1. Estrutura inicial

diante. Para a próxima janela escolha Empty Activity(Tela vazia) e logo em seguida de um finish.

O android Studio devera ter criado um tipo de estrutura de pastas parecidas como a Figura 1.

Logo acima da pasta app, temos os tipos de visões, o tipo de visão Android, mostra exatamente os arquivos que são usados no android, a pasta app contem todo seu projeto.

Uma breve explicação sobre as pastas:

- manifest
  - Contem a configuração da aplicação, como caminho,permissões etc...
- java
  - Contem as pastas que armazenam os códigos escritos em java.
- res
  - Contem as pastas que são responsáveis pela parte visual da aplicação.

2.2. Criando o layout da tela

A pasta layout dentro da pasta 'res' contem as telas. Por default de uma aplicação vazia, deve um arquivo chamado

```
'activity_main.xml',
```

que é a nossa tela principal. Ao abri-lo, podemos ver a seguinte representação, de acordo com a figura 2.

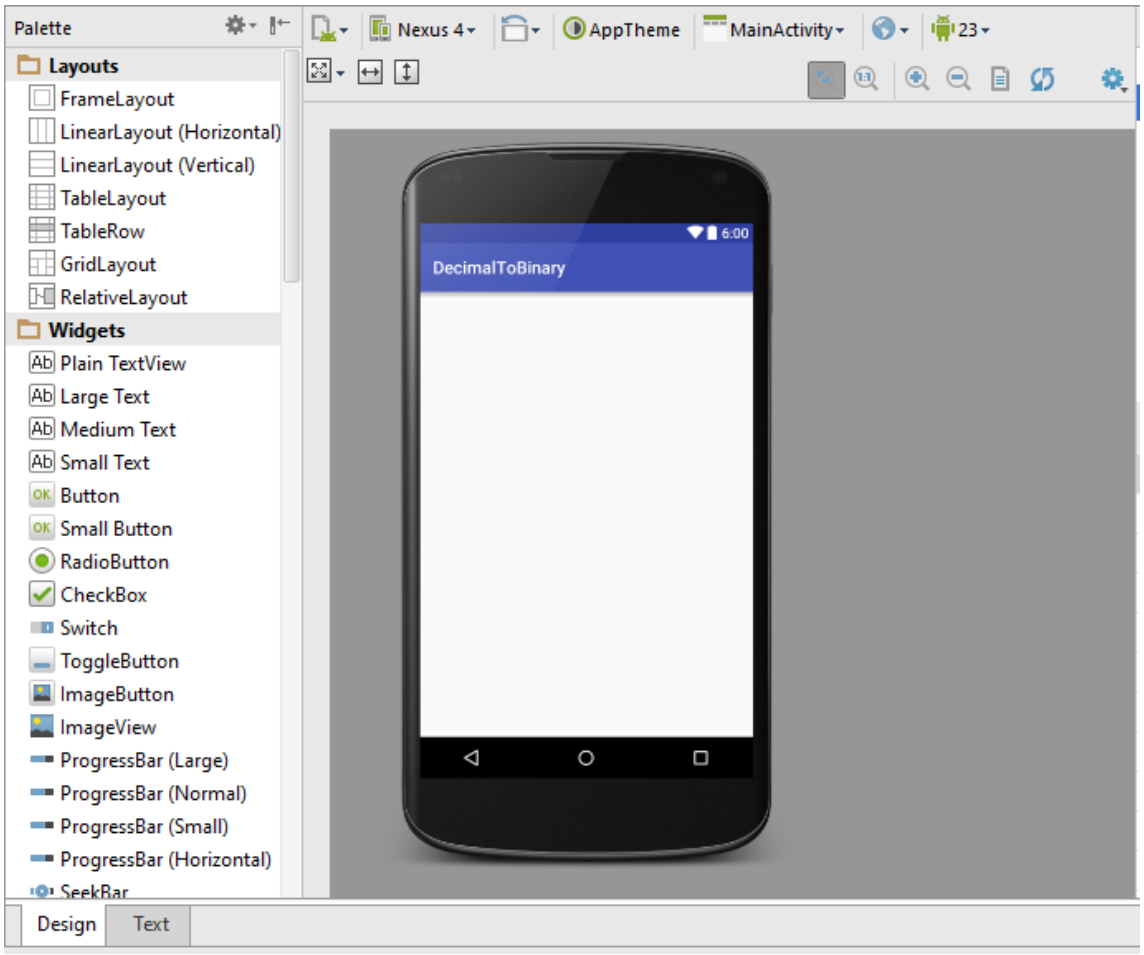

Figura 2. Tela Inicial

No lado esquerdo na aba design, temos todos os elementos que podemos adicionar na tela, podemos adiciona-los simplesmente fazendo um "drag and drop".

Na aba 'text', temos o código dos elementos da tela. Já que nossa aplicação consiste em converter um número decimal para binário,octal e hexadecimal, teremos uma caixa para inserir os números decimais e 3 labels(rótulos) mostrando as conversões para seus respectivos tipos.

Logo a baixo teremos um botão conectar, que irá fazer com que o aparelho conecte com o modulo bluetooth. Já que nosso objetivo é enviar apenas o binário para o Arduíno, teremos um outro botão para enviar, além de conter um checkBox para poder enviar em tempo de alteração do campo de decimais.

Para criar tal cenário, apague o código da aba 'text', e adicione o seguinte código.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.example.cyber.decimaltobinary.MainActivity">

   <LinearLayout
      android:id="@+id/linear"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:focusable="true"
      android:focusableInTouchMode="true"
      android:orientation="vertical">


      <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:gravity="center"
         android:text="Decimal"
         android:textAppearance="?android:attr/textAppearanceLarge"
            />

      <EditText
         android:id="@+id/inboxdec"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:digits="0123456789"
         android:inputType="number" />

      <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Binario:"
         android:id="@+id/txtBin"
         android:textAppearance="?android:attr/textAppearanceLarge"
            />
      <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Octal:"
         android:id="@+id/txtOct"
```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
            />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hexadecimal:"
        android:id="@+id/txtHex"
        android:textAppearance="?android:attr/textAppearanceLarge"
            />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar automaticamente"
        android:id="@+id/ckbEnviarAutomaticamente"
        android:layout_above="@+id/btnConectar"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />


</LinearLayout>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/linear"
    android:layout_centerHorizontal="true">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Conectar"
        android:id="@+id/btnConectar"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar Binario"
        android:id="@+id/btnEnviar"
        android:layout_alignTop="@+id/btnConectar"
        android:layout_alignRight="@+id/linear"
        android:layout_alignEnd="@+id/linear"
        android:enabled="false" />

</LinearLayout>
```

```
</RelativeLayout>
```

Irá gerar a seguinte tela, conforme a figura 3.

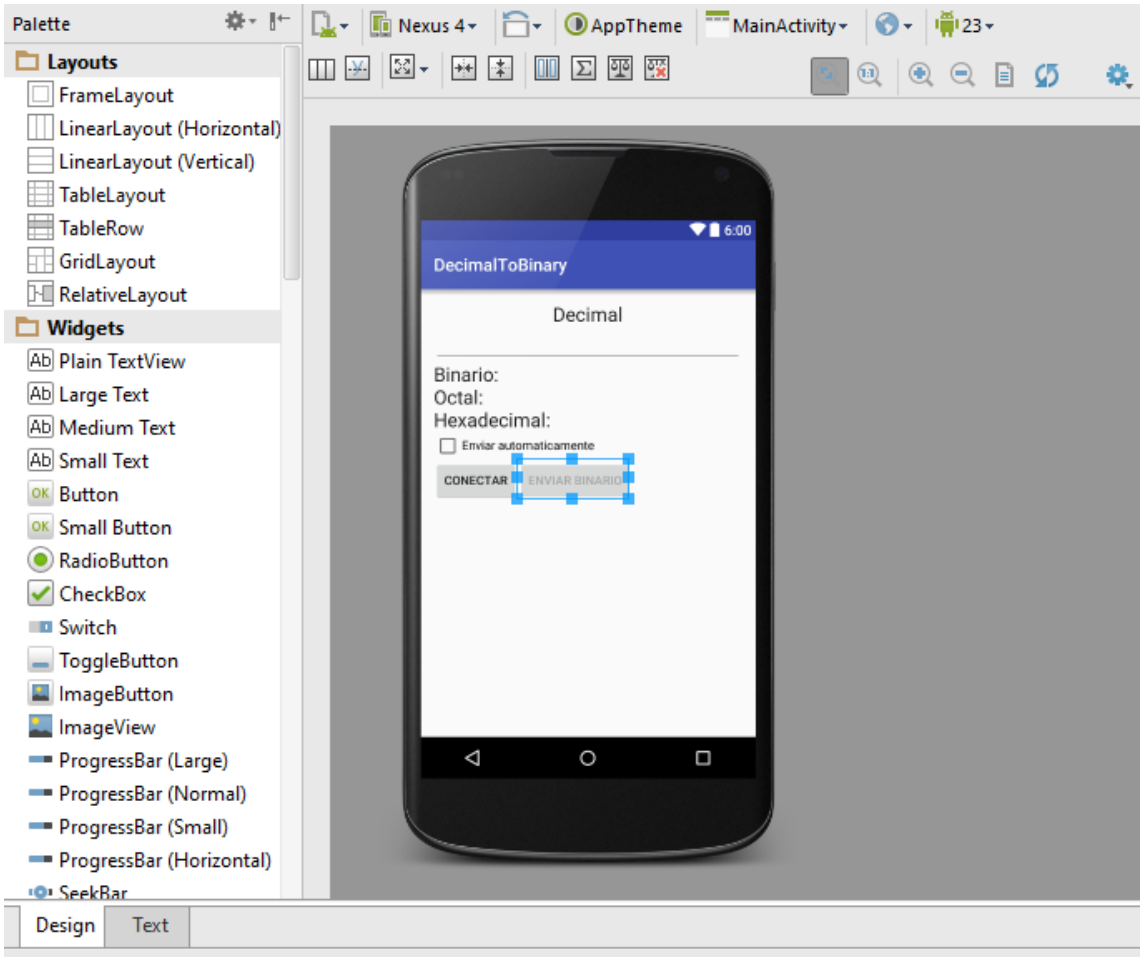

Figura 3. Tela pronta

Para cada elemento da tela, em seu código, podemos observar que existe,

```
android:id=""
```

ele é responsável por dar um nome ao elemento da tela que só o sistema android ira enxergar.

Para o campo 'EditText' do decimal, delimitamos que só haverá a possibilidade de inserir números de 0 a 9,o botão 'enviar binário' deixaremos desabilitado, só irá habilitar quando o aparelho estiver conectado com o modulo bluetooth.

Agora que criamos os elementos na tela, vamos construir o código que converte decimal para binário,octal e hexadecimal.

Observando nosso diretório de pastas, na pasta 'Java', e na pasta que contém o 'MainActivity' por default, podemos observar que existe um método nativo das aplicações chamado, onCreate que faz parte de um grupo chamado de ciclo de vida de uma activity,

como podemos ver na figura 4.

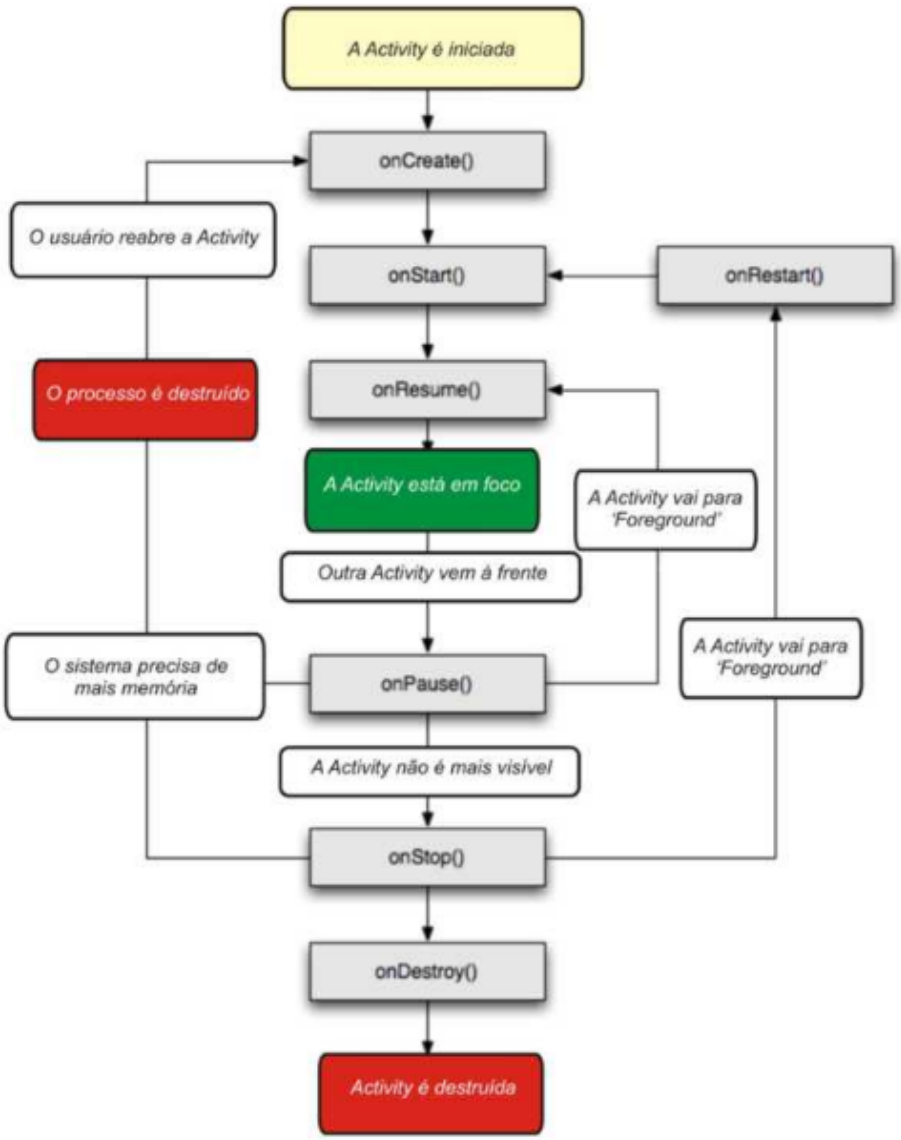

Figura 4. Ciclo de vida da activity

O método nativo onCreate é muita vezes usado quando se há necessidade de referenciar elementos da tela uma única vez, já que ele é apenas executado quando a aplicação é a aberta pela 1º vez.

Usaremos mais 2 métodos nativos, que serão, onResume que conterá os códigos para mudanças nas variáveis e outras alterações de estado, e o método onPause(), que terá o código que desconectara do bluetooth quando a tela não esta mais sendo visível.

2.3. Referenciando os elementos da tela

Nessa fase precisamos criar as variáveis no escopo para que possam ser visíveis para a classe inteira, depois teremos que referenciar suas variáveis com os elementos da tela,conforme vemos a seguir:

```
    private EditText editTxtDecimal;
    private TextView txtBin,txtOct,txtHex;
    private Button btnConectar,btnEnviar;
    private CheckBox enviarAutomaticamente;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTxtDecimal = (EditText) findViewById(R.id.inboxdec);
        editTxtDecimal.setFilters(new InputFilter[]{new
            InputFilterMinMax("0", "255")});

        txtBin = (TextView) findViewById(R.id.txtBin);
        txtOct = (TextView) findViewById(R.id.txtOct);
        txtHex = (TextView) findViewById(R.id.txtHex);

        btnConectar = (Button) findViewById(R.id.btnConectar);
        btnEnviar = (Button) findViewById(R.id.btnEnviar);
        enviarAutomaticamente =
            (CheckBox)findViewById(R.id.ckbEnviarAutomaticamente);
    }
```

Embaixo da referencia de editTxtDecimal, podemos observar um tipo de setFilters que filtra um minimo e máximo,já que nosso binário só poderá conter 8 dígitos para poder enviar por bluetooth, precisamos limitar nosso decimal que sera convertido para binário, a um valor máximo de 255 que seria o máximo de 8 dígitos em binário.

Na pasta java, crie uma classe java, com o nome 'InputFilterMinMax', e coloque o seguinte código contendo uma logica que delimita o editText.

```
    import android.text.InputFilter;
import android.text.Spanned;

public class InputFilterMinMax implements InputFilter {

    private int min, max;

    public InputFilterMinMax(int min, int max) {
        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }
```

```
    @Override
    public CharSequence filter(CharSequence source, int start,
        int end, Spanned dest, int dstart, int dend) {
        try {
            int input = Integer.parseInt(dest.toString() +
                source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}
```

2.4. Criando métodos java de conversão

Agora que referenciamos nossos elemento básicos, precisamos inserir algum tipo de código que converta decimal para 3 tipos de destinos finais ao mesmo tempo que é digitado o decimal, para isso usaremos um métodos de conversão prontos do java que se chamam, 'Integer.toBinaryString(), Integer.toOctalString() e Integer.toHexString()'.

O método android que captura as mudanças feitas no campo de texto do decimal se chama 'addTextChangedListener', e para seu uso é necessário que criemos 3 métodos de eventos obrigatórios, são eles,'beforeTextChanged, onTextChanged , afterTextChanged'. No nosso caso, colocaremos nosso código dentro do método 'onTextChanged', segue o código dessa explicação.

```
     @Override
    protected void onResume() {
        super.onResume();

        editTxtDecimal.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
                start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start,
                int before, int count) {
                if (editTxtDecimal.getText().toString().equals(""))
                    {
                    txtBin.setText("Binario: ");
                    txtOct.setText("Octal: ");
                    txtHex.setText("Hexadecimal:");
                }else{
```

```
                txtBin.setText("Binario: " +
                    Integer.toBinaryString(Integer.parseInt(
                editTxtDecimal.getText().toString())));
                txtOct.setText("Octal: " +
                    Integer.toOctalString(Integer.parseInt(
                editTxtDecimal.getText().toString())));
                txtHex.setText("Hexadecimal: " +
                    Integer.toHexString(Integer.parseInt(
                editTxtDecimal.getText().toString())));

                if(enviarAutomaticamente.isChecked()){
                    enviarBinario();
                }

            }

        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
}
```

O trecho acima serve para, quando houver alguma alteração no campo de texto destinado a receber o número decimal, seja convertida para as bases ou, caso o campo estiver vazio, apenas escrevemos os nomes das bases, sem realizar nenhuma conversão.

Logo abaixo temos uma opção que verifica se a opção de enviar automaticamente esta ativada, caso esteja, chamaremos o método que enviará somente o valor do binário por bluetooth.

Para poder enviar por bluetooth, precisamos nos conectar com um modulo bluetooth que já esteja pareado com o aparelho.

Antes de qualquer interação com o bluetooth, o Android precisa ter um código que receba tal permissão do aparelho para haver manipulação com bluetooth, estas requisições de permissão sempre aparecerá na hora da instalação de um aplicativo.

Na pasta 'manifest' e em androisManifest coloque o seguinte código antes da tag 'application' para garantir tal permissão na hora da instalação do aplicativo.

```
    <uses-permission
        android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH"
        />
```

Muito bem, temos nossa permissão, mas antes de escrevermos o código dessa classe precisamos explicar a seguinte situação.

Quando clicarmos no botão conectar, verificaremos se dispositivo possui adaptador bluetooth, se possuir, chamaremos nossa classe que exibe os dispositivos na tela, só que para chama-la, precisamos enviar um valor como se fosse um identificador, pois chamamos a classe pelo método 'startActivityForResult(classe)' que qualquer retorno a partir desse método, irá cair no método reservado do Android chamado 'onActivityResult', então resumindo, quando selecionarmos nosso dispositivo na tela, a classe ira retornar o endereço do dispositivo, junto com sua chave de in-dentificação para que o Android possa ver 'quem' esta retornando.

Primeiramente coloque essas variáveis que servirão como identificadores, e a variável Android para conseguirmos identificar se o dispositivo possui bluetooth.

```
// Requisicao para Activity de ativao do Bluetooth
// Se numero for maior > 0,este codigo sera devolvido em
   onActivityResult()
private static final int REQUEST_ENABLE_BT = 1;
// Requisicao para Activity para inciar tela do aplicativos
   pareados,
// que se houver ou nao aplicativo pareado retornara para
   onActivityResult()
// e realizara as devidas aes conforme a resposta
public static final int SELECT_PAIRED_DEVICE = 2;

// BluetoothAdapter  comando de entrada padro paras todas
   interaes com bluetooth
private BluetoothAdapter bluetoothPadrao = null;

//Usado para conectar com o dispositivo bluetooth
BluetoothThread btt;
Handler writeHandler;
```

Logo em seguida coloque esse trecho dentro do método onCreate().

```
// Obtem o bluetooth padrao do aparelho celular
   bluetoothPadrao = BluetoothAdapter.getDefaultAdapter();
```

2.5. Exibir e selecionar dispositivos pareados com o aparelho

Precisamos de uma tela que exiba esses dispositivos pareados, mas não criaremos toda essa tela, criaremos somente sua casca e deixaremos que métodos android crie automaticamente mais adiante.

Primeiramente crie um "layout resource file"com o nome 'textheader' dentro das pasta 'layout' que fica na pasta 'res'. Depois de criado adicione um 'TextView' vazio que sera populado por informações.

Segue o trecho de código que devera ser colocado na aba 'text' de 'textheader'.

```
<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="..."
    android:textAppearance="?android:attr/textAppearanceLarge"
       />
```

No próximo passo criaremos uma classe separada que sera responsável por exibir os dispositivos pareados.

Na pasta java, crie uma classe java com o nome 'PairedDevices' e coloque o código a seguir que gerencia essa parte de exibir os dispositivos pareados.

```
package com.example.cyber.decimaltobinary;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.app.ListActivity;
import android.view.View;
import android.view.WindowManager;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import java.util.Set;


public class PairedDevices extends ListActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
     /*
      * Esse trecho não é essencial, mas da um melhor visual a
         lista.
      * Adiciona um titulo a lista de dispositivos pareados
         utilizando o
      * layout text_header.xml.
      */
      ListView lv = getListView();
      LayoutInflater inflater = getLayoutInflater();
      View header = inflater.inflate(R.layout.text_header, lv,
         false);
      ((TextView)
         header.findViewById(R.id.textView)).setText("\nDispositivos
         pareados\n");
      lv.addHeaderView(header, null, false);
     /*
      * Usa o adaptador Bluetooth para obter uma lista de
```

```
      dispositivos
    * pareados.
    */
     BluetoothAdapter btAdapter =
        BluetoothAdapter.getDefaultAdapter();
     Set<BluetoothDevice> pairedDevices =
        btAdapter.getBondedDevices();

   /*
    * Cria um modelo para a lista e o adiciona a tela. Se
       houver
    * dispositivos pareados, adiciona cada um a lista.
    */
     ArrayAdapter<String> adapter = new
        ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1);
     setListAdapter(adapter);
     if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
           adapter.add(device.getName() + "\n" +
              device.getAddress());
        }
     }
}
/*
 * Este metodo é executado quando o usuario seleciona um
    elemento da lista.
 */
@Override
protected void onListItemClick(ListView l, View v, int
   position, long id) {
   /*
    * Extrai nome e endereço a partir do conteudo do elemento
       selecionado.
    * Nota: position-1 não utilizado pois adicionamos um
       titulo a lista e o valor
    * de position recebido pelo metodo é deslocado em uma
       unidade.
    */
     String item = (String) getListAdapter().getItem(position
        - 1);
     String devName = item.substring(0, item.indexOf("\n"));
     String devAddress = item.substring(item.indexOf("\n") +
        1, item.length());

   /*
    * Utiliza um Intent para encapsular as informações de nome
       e endereço.
    * Informa a Activity principal que tudo foi um sucesso!
```

```
            Finaliza e
        * retorna a Activity principal.
        */
         Intent returnIntent = new Intent();
         returnIntent.putExtra("btDevName", devName);
         returnIntent.putExtra("btDevAddress", devAddress);
         setResult(RESULT_OK, returnIntent);
         finish();
    }
}
```

Temos uma pequena observação, para toda e qualquer classe que esta ligada com uma tela, é necessário sua declaração dentro do 'AndroidManigest.xml'.

Declare esse pequeno trecho depois da ultima declaração de activity.

```
    <activity android:name=".PairedDevices"></activity>
```

Agora que criamos nossa classe que exibe e retorna o dispositivo selecionado, precisamos conectar com esse dispositivo.

2.6. Conectando com o dispositivo

A parte de conexão será realizada por uma classe externa que executará em segundo plano, crie uma classe com o nome 'BluetoothThread' e adicione o código a seguir.

```
package com.example.cyber.decimaltobinary;

       import android.annotation.SuppressLint;
       import android.bluetooth.BluetoothAdapter;
       import android.bluetooth.BluetoothDevice;
       import android.bluetooth.BluetoothSocket;

       import java.io.InputStream;
       import java.io.OutputStream;

       import android.os.Handler;
       import android.os.Message;

       import android.util.Log;

       import java.util.UUID;

@SuppressLint({ "DefaultLocale", "HandlerLeak" })
public class BluetoothThread extends Thread {

    // Tag for logging
    private static final String TAG = "BluetoothThread";

    // Delimiter used to separate messages
```

```
private static final char DELIMITER = ';';

// UUID especifica um protocolo para comunicacaoo serial
   Bluetooth genérica
private static final UUID uuid =
   UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

// Ira armazenar o endereço MAC do disposivo Bluetooth
private final String address;

// BluetoothSocket cria um ponto de conexão que permite
   trocar dados com outro
// disposivo bleutooth atraves do ImputStream() e
   OutputStream()
public BluetoothSocket socket;
private OutputStream outStream;

// Handlers usado para passar dados entre threads
private final Handler readHandler;
private final Handler writeHandler;
/**
 * Construtor da classe, recebe o endereço MAC do dispositivo
    bluetooth
 * e um Handler para as mensagens recebidas.
 *
 */
public BluetoothThread(String address, Handler handler) {

   this.address = address.toUpperCase();
   this.readHandler = handler;

   writeHandler = new Handler() {
      @Override
      public void handleMessage(Message message) {
         write((String) message.obj);
      }
   };
}
/**
 * Devolve o manipulador(Handler) de escrita para a conexão.
    As mensagens
 * recebidas por este manipulador(handler) sera escrito no
    Bluetooth socket.
 */
public Handler getWriteHandler() {
   return writeHandler;
}

/**
```

```
 * Conectar o Bluetooth socket, ou lançar uma exeção se ele
    falhar.
 */
private void connect() throws Exception {

   Log.i(TAG, "Attempting connection to " + address + "...");

   // busca o dispositivo padrao do aparelho
   BluetoothAdapter adapter =
      BluetoothAdapter.getDefaultAdapter();

   // Find the remote device
   BluetoothDevice remoteDevice =
      adapter.getRemoteDevice(address);

   // Create a socket with the remote device using this
      protocol
   socket =
      remoteDevice.createRfcommSocketToServiceRecord(uuid);

   // Make sure Bluetooth adapter is not in discovery mode
   adapter.cancelDiscovery();

   // Connect to the socket
   socket.connect();

   // Get input and output streams from the socket
   outStream = socket.getOutputStream();
   Log.i(TAG, "Connected successfully to " + address + ".");
}

/**
 * Disconnect the streams and socket.
 */
private void disconnect() {
   if (outStream != null) {
      try {
         outStream.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   if (socket != null) {
      try {
         socket.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

```
}
/**
 * Write data to the socket.
 */
private void write(String s) {

   try {
      // Add the delimiter
       s += DELIMITER;
      byte[] msgBuffer = s.getBytes();
      outStream.write(msgBuffer);
      Log.i(TAG, "[SENT] " + s);
   } catch (Exception e) {
      Log.e(TAG, "Write failed!", e);
   }
}

/**
 * Pass a message to the read handler.
 */
private void sendToReadHandler(String s) {
   Message msg = Message.obtain();
   msg.obj = s;
   readHandler.sendMessage(msg);
   Log.i(TAG, "[RECV] " + s);
}

/**
 * Entry point when thread.start() is called.
 */
public void run() {

   // Attempt to connect and exit the thread if it failed
   try {
      connect();
      sendToReadHandler("CONNECTED");
   } catch (Exception e) {
      Log.e(TAG, "Failed to connect!", e);
      sendToReadHandler("CONNECTION FAILED");
      disconnect();
      return;
   }

   // Loop continuously, reading data, until
      thread.interrupt() is called
   while (!this.isInterrupted()) {

      // Make sure things haven't gone wrong
      if (outStream == null) {
```

```
                Log.e(TAG, "Lost bluetooth connection!");
                break;
            }
        }
        // Se thread é interrompido, ele fecha a conexão
        disconnect();
        sendToReadHandler("DISCONNECTED");
    }
}
```

Criamos nossa classe que cuidará da conexão e comunicação, agora precisamos criar a lógica dos botoes de conexão e envio e demais métodos otimizadores.

Para poder usar a classe que criamos, precisamos adicionar uma instância dela e um tipo de variável que permitirá fazer comunicações de escritas entre elas, adicione as seguintes variáveis no escopo do projeto.

```
    BluetoothThread btt;
    Handler writeHandler;
```

2.7. Metodos dos botoes e ações do bluetooth

Precisamos também colocar agora as ações dos nossos botoes, coloque o código a seguir dentro do método 'onCreate'.

```
    btnConectar.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
             if (bluetoothPadrao == null) {
                Toast.makeText(getApplicationContext(),
                   "Dispostivo nÃ$o possui Bluetooth",
                   Toast.LENGTH_LONG).show();
             } else {
                if (!bluetoothPadrao.isEnabled()) {
                   Intent novoIntent = new
                      Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                   startActivityForResult(novoIntent,
                      REQUEST_ENABLE_BT);
                } else {
                   listaDeDispositivos();
                }
             }

          }
       });

       btnEnviar.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
```

```
            if (editTxtDecimal.getText().toString().equals(""))
               {
               Toast.makeText(getApplicationContext(), "NÃ$o ha
                  binario para enviar",
                  Toast.LENGTH_LONG).show();
            }else{
               enviarBinario();
            }
         }
      });
```

E o restante do nosso código é inserido fora do método 'onCreate', logo abaixo dele.

```
    public void enviarBinario(){
      if (btt != null) {
         Message msg = Message.obtain();
         msg.obj = Integer.toBinaryString(Integer.parseInt(
         editTxtDecimal.getText().toString()));
         writeHandler.sendMessage(msg);
      }
   }

   @Override
   public void onStop() {
      super.onStop();;
      interromperBluetooth();

   }

   public void interromperBluetooth(){
      if(btt != null){
         btnConectar.setText("Conectar");
         btt.interrupt();
         btt = null;
         btnConectar.setEnabled(true);
         btnEnviar.setEnabled(false);
      }

   }

    public void listaDeDispositivos() {
      if (bluetoothPadrao.isEnabled()) {
         if (btt == null) {
            Intent searchPairedDevicesIntent = new Intent(this,
               PairedDevices.class);
            startActivityForResult(searchPairedDevicesIntent,
               SELECT_PAIRED_DEVICE);
         } else {
```

```
            interromperBluetooth();
        }
    }
}

protected void onActivityResult(int requestCode, int
    resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {

        // Retrono do pedido de ativacao do Bluetooth
        case REQUEST_ENABLE_BT:
            if (resultCode == Activity.RESULT_OK) {
                Toast.makeText(getApplicationContext(),
                    "Bluetooth Ativado XD",
                    Toast.LENGTH_LONG).show();
                listaDeDispositivos();
            } else {
                Toast.makeText(getApplicationContext(), "Voce
                    precisa ativar o bluetooth ",
                    Toast.LENGTH_LONG).show();
            }
            break;
        case SELECT_PAIRED_DEVICE:
            if (resultCode == RESULT_OK) {
                if (btt == null) {
                    btt = new
                        BluetoothThread(data.getStringExtra("btDevAddress"),
                        new Handler() {
                        @Override
                        public void handleMessage(Message message)
                            {
                            String s = (String) message.obj;
                            //Em tempo de execucao compara as
                                mensagens recebidas pelo gerenciado
                                de conexao
                            if (s.equals("CONNECTED")) {
                                btnConectar.setText("Desconectar");
                                btnConectar.setEnabled(true);
                                Toast.makeText(getApplicationContext(),
                                    "Conectado",
                                    Toast.LENGTH_LONG).show();
                                btnEnviar.setEnabled(true);
                            } else if (s.equals("DISCONNECTED")) {
                                Toast.makeText(getApplicationContext(),
                                    "Desconectado",
                                    Toast.LENGTH_LONG).show();
                                interromperBluetooth();
                            } else if (s.equals("CONNECTION
```

```
                        FAILED")) {
                        Toast.makeText(getApplicationContext(),
                            "Falha na conexao",
                            Toast.LENGTH_LONG).show();
                        interromperBluetooth();
                    }
                }
            });
        }

        if (btt != null) {
            // Get the handler that is used to send
               messages
            writeHandler = btt.getWriteHandler();

            // Run the thread
            btt.start();

            btnConectar.setText("Conectando...");
            btnConectar.setEnabled(false);
        }
    } else {
        Toast.makeText(getApplicationContext(), "Nenhum
            dispositivo Selecionado",
            Toast.LENGTH_LONG).show();
    }
    break;

  }
};
```

Explicando por cima o que cada método faz...

enviarBinario()

Usado para enviar o binário por bluetooth, resgatamos o valor do binário convertendo o valor de decimal para binário, logo em seguida, convertemos esse valor para um formato de mensagem que sera enviado para 'BluetoothThread' por meio do

```
writeHandler.sendMessage(msg);
```

Que por sua vez, a classe 'BluetoothThread' enviara o valor em bytes por bluetooth, que ao chegar ao Arduíno, convertera esses bytes em algo entendível.

onStop()

Toda vez que a tela atual da aplicação não estiver mais visível, esse método sera chamado.

interromperBluetooth()

Esse método contem funções que desconectarão o bluetooth e limparão a instancia de 'BluetoothThread', mostrando assim, que o bluetooth foi desconectado.

listaDeDispositivos()

Esse método chama a tela que mostrara os dispositivos pareados na tela, independente se o usuário escolher ou não algum dispositivo da lista, retornara seu resultado para onActivityResult(...).

onActivityResult(....)

Esse método nativo do Android, é usado muita vezes para resultado de chamadas, por Exemplo no 1º case REQUEST ENABLE BT: Quando clicamos em conectar, se o bluetooth não estiver ativado, a aplicação requisitara a permissão da ativação do mesmo, caso aceito ou não, retornara para onActivityResult com seu identificador e com sua resposta.

Logo a baixo no 'case SELECT PAIRED DEVICE:' temos a mesma situação para tela dos dispositivos pareados, se o usuário selecionou algum dispositivo da lista, retornara com o endereço do bluetooth e mandara para 'BluetoothThread' tentar fazer tal conexão. De acordo com o resultado dessa tentativa de conexão, retornara seu resultado para a 'onActivityResult', que de acordo com a resposta realizara suas devidas ações.

2.8. Finalizando aplicação

Pronto, com oque temos já podemos fazer os teste com o Arduíno, para compilar a aplicação devemos ir em **Build, Generate Signed Apk**, e apenas seguir os passos dela. Apos isso devera ter gerando um apk, que devera enviado para o celular e ser feito a instalação do mesmo.

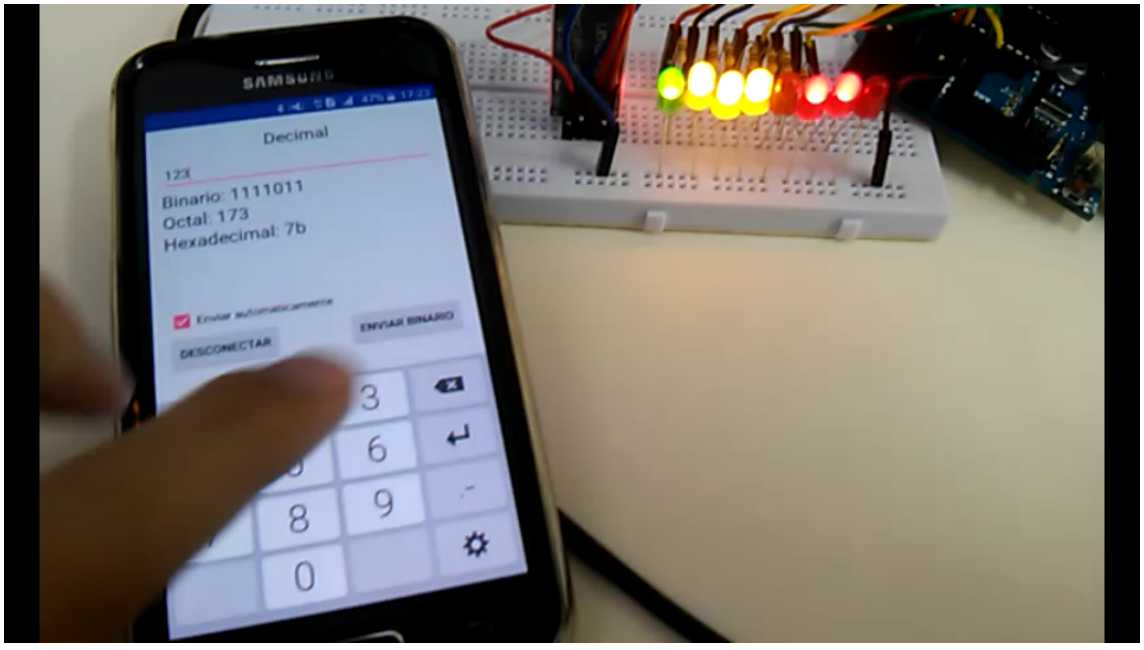

Figura 5. Aplicação

3. Arduino

3.1. Eletrônica

A parte eletrônica do projeto é bem simples,sendo composto por 8 LEDs, 8 resistores de 220 Ohms,1 protoboard 1 placa Arduino Uno e uma placa bluetooth HC-06, ligadas entre si conforme representa a figura 6.

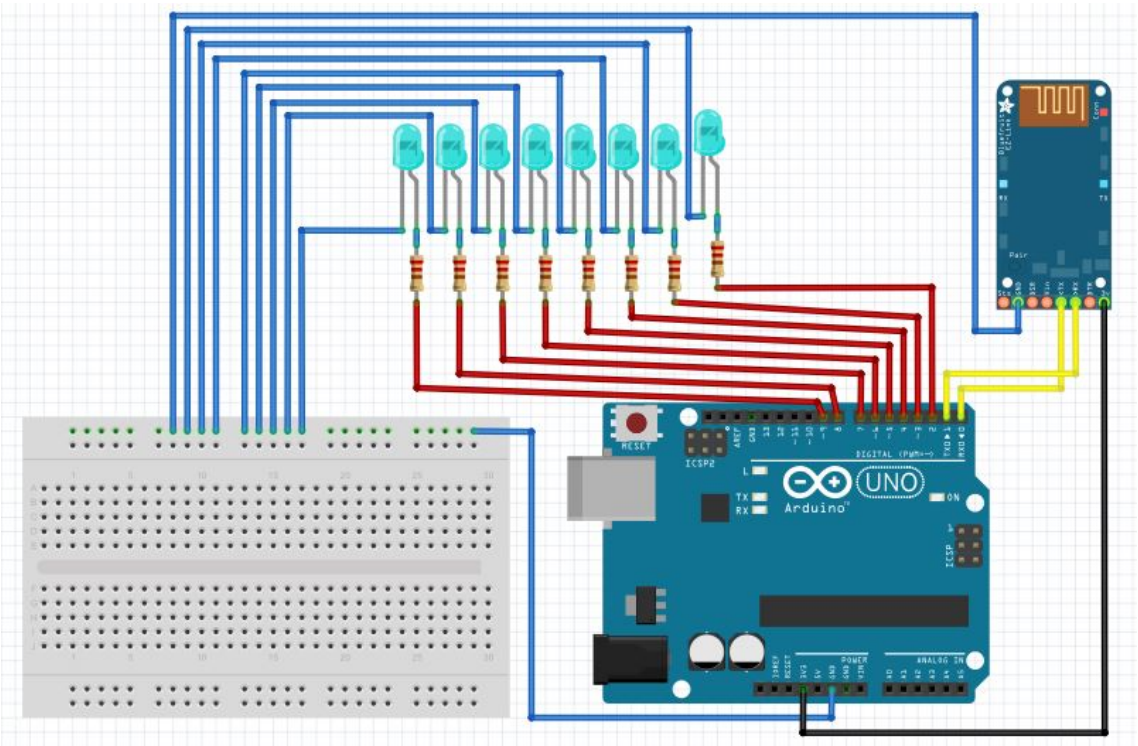

Figura 6. Ligações do projeto

Como visto na figura, cada LED é ligado em seu terminal positivo a um resistor de 220 Ohms, e esse a sua respectiva posição no intervalo de 2 a 9 das portas digitais do Arduino Uno, enquanto os terminais negativos estão todos ligados no GND da mesma placa. Já o módulo de Bluetooth está com suas portas TX e RX ligadas as portas RX e TX (respectivamente) do Arduino, e é alimentado através da saída de 3V.

3.2. Programação

O Programa desenvolvido no Arduino é igualmente simples, com uma função principal e três funções secundárias com o intuito de deixar o código mais limpo e mais fluído.

3.2.1. Programa principal

No programa principal é iniciada a variavel ”numeros”, que receberá a sequência de zeros e uns, e logo após isso, no Setup é iniciado a serial.

```
String numeros;

void setup() {
```

```
  Serial.begin(9600);
  Serial.setTimeout(9999999);

}
```

Já no loop temos a variável já iniciada recebendo o conteúdo da serial e trabalhamos em cima do comprimento de ”numeros”para fazer um loop varrendo-o completamente.

Antes de entrar propriamente no loop (for), apagamos todos os LEDs para podermos tratar apenas de acende-los quando necessário. No for criado fazemos uma variável Char de nome ”bi”receber cada posição da string recebida da serial, e logo após acendemos ou não o LED que representa a posição do vetor.

```
 void loop() {

   numeros = (String)Serial.readStringUntil(';');
   apagaLeds();
  for (int i = 0; i < numeros.length(); i++) {
   char bi = (char)numeros[i];

   switch (i) {
     case 0:
      if (bi == '1') {acendLed(2);}
       break;
     case 1:
     if (bi == '1') {acendLed(3);}
       break;
     case 2:
     if (bi == '1') {acendLed(4);
       break;
     case 3:
     if (bi == '1') {acendLed(5);}
       break;
     case 4:
     if (bi == '1') {acendLed(6);}
       break;
     case 5:
     if (bi == '1') {acendLed(7);}
       break;
     case 6:
     if (bi == '1') {acendLed(8);
       break;
     case 7:
     if (bi == '1') {acendLed(9);}
       break;

   }
```

```
  }
  numeros = "";
}
```

3.2.2. Funções

Criamos três funções auxiliares, uma para acender o LED recebido, onde apenas indicamos qual a porta lógica deve receber o sinal para acender o LED que representa a posição do digito 1 do número binário

```
void acendLed(int led){
  analogWrite(led, 255);
}
```

Essa função não foi utilizada na versão final do software, mas foi utilizada para testes.

```
void apagaLed(int led){
  analogWrite(led, 0);
}
```

Essa função foi criada para garantir que todos os LEDs estejam apagados ao iniciar uma conversão, ao mesmo tempo que garante que se o número for menor do que 8 digitos, os digitos que não serão utilizados permanecerão apagados.

```
void apagaLeds() {
  analogWrite(2, 0);
  analogWrite(3, 0);
  analogWrite(4, 0);
  analogWrite(5, 0);
  analogWrite(6, 0);
  analogWrite(7, 0);
  analogWrite(8, 0);
  analogWrite(9, 0);
}
```