b

MIT

APP INVENTOR

Sumário

Sumário de Ilustrações

Notas sobre esta apostila

Este material, foi desenvolvido com intuito acadêmico para uma introdução sobre o maravilhoso mundo dos aplicativos para celulares.

Esta apostila virtual é gratuita e não pode ser revendida **apenas distribuída**. Não há necessidade de me contatar para distribuir, sinta-se à vontade!

Caso venha ter alguma dúvida sinta-se à vontade a para entrar em contato comigo pelo e-mail *brunomichel00@gmail.com*

1 Introdução

Segundo o site TechTudo, *Google App Inventor* é uma ferramenta desenvolvida pela Google que permite a criação de aplicativos para smartphones que rodam o sistema operacional *Android*, sem que seja necessário conhecimentos profundos de programação.

De acordo com Harold Abelson, um pesquisador do MIT que trabalhou nesse projeto, o objetivo de *Google App Inventor* é permitir que utilizadores sejam também criadores e não apenas consumidores.

Com uma interface simples e fácil de usar, o programa foge das linhas de programação normal e possibilita até mesmo usuários comuns lançarem seus aplicativos. Graças ao recurso *drag and drop* (arrastar e soltar), a programação das aplicações acontece de forma simples e intuitiva.

Essa apostila tem como objetivo introduzir o usuário ao mundo das aplicações *Android* com exemplos e exercícios. Não se deve tomar essa apostila como material único e definitivo, apenas como um material inicial. À venda deste conteúdo é proibido, mas sua distribuição gratuita não!

2 Interface

A interface do *App Inventor* é limpa e organizada de modo bastante semelhante às IDEs mais comuns, com algumas diferenças que visam uma otimização do trabalho e melhor entendimento pelo usuário. No lugar de uma árvore de arquivos do lado esquerdo da tela, por exemplo, a aplicação apresenta elementos que podem ser utilizados na criação das aplicações, bastando um simples comando de “clique e arraste” para inserir o item desejado no projeto.

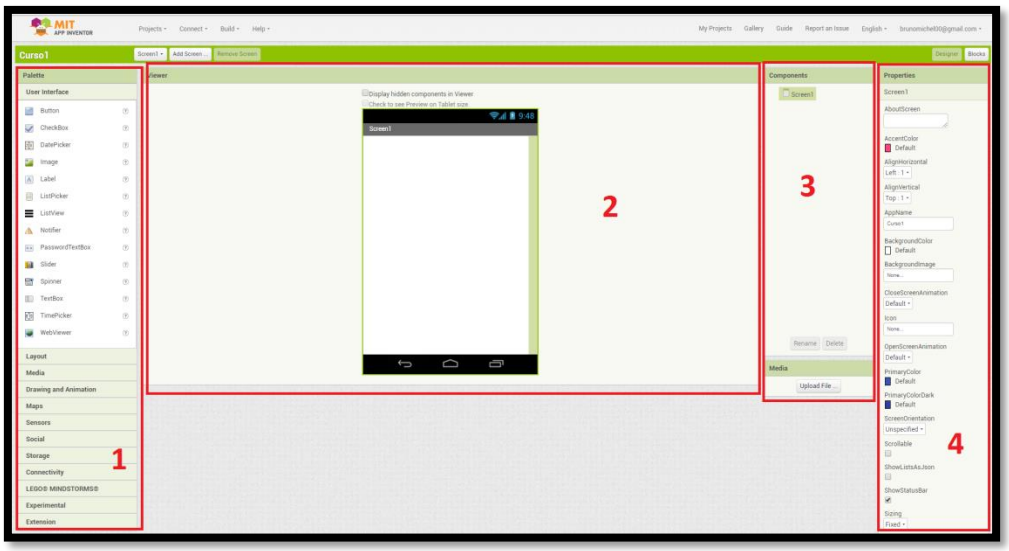


Figura 1 Interface do Google App Inventor

2.1 Item 1 *Palette*

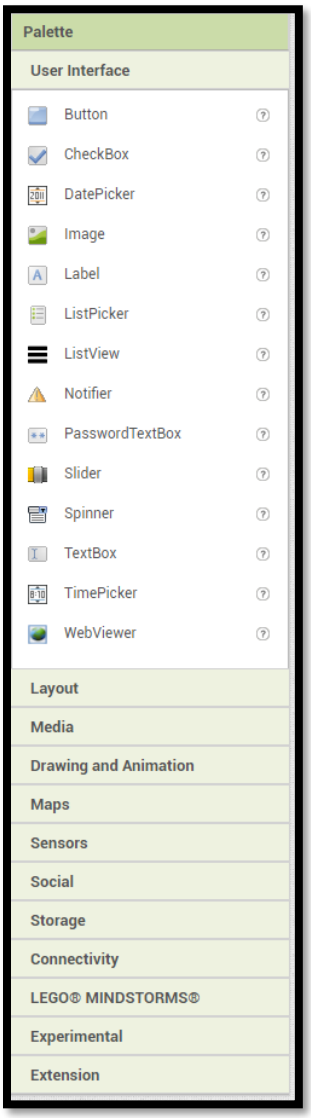


Figura 2 Palette

Como nas IDEs, no App Inventor existe a palheta de itens, essa palheta contém todos os objetos que você poderá usar em seu aplicativo, ele utiliza o estilo de organização chamado de *Drop Down menu* no quando ao clicar em outro item, o item superior irá se fechar e o debaixo irá se abrir.

2.2 Item 2 *Viewer*

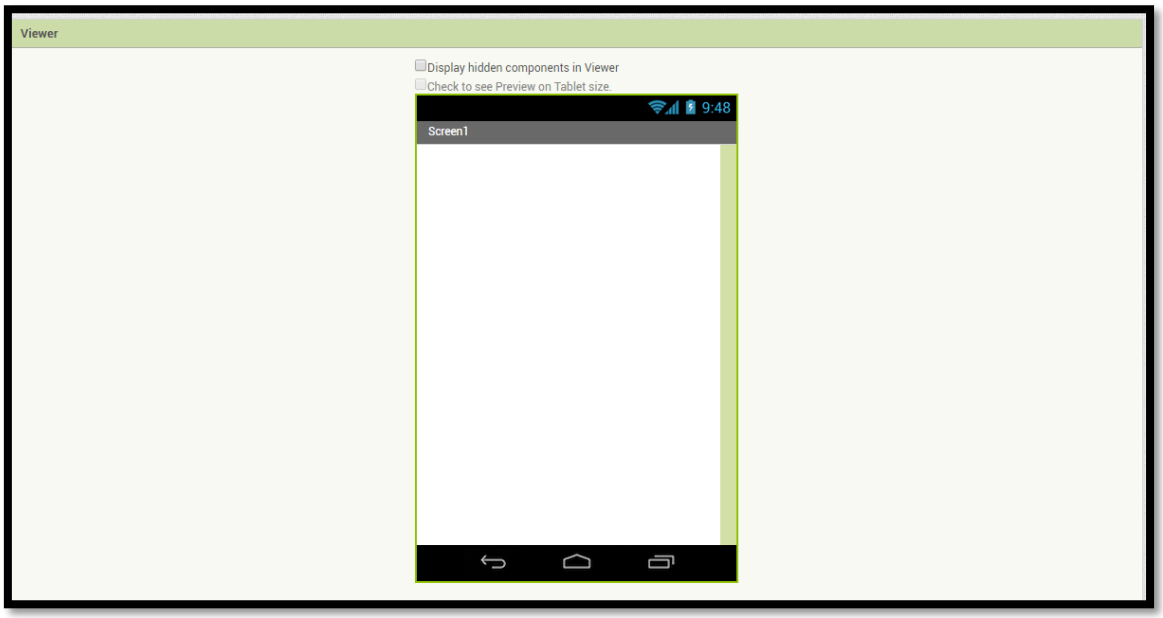


Figura 3 Viewer

O visualizador é literalmente aonde você irá acompanhar a interface do seu aplicativo todo os itens arrastados da palheta irão ficar visíveis nesta tela, a qual está simulando a tela de um celular com o sistema Android.

2.3 Item 3 Components e media

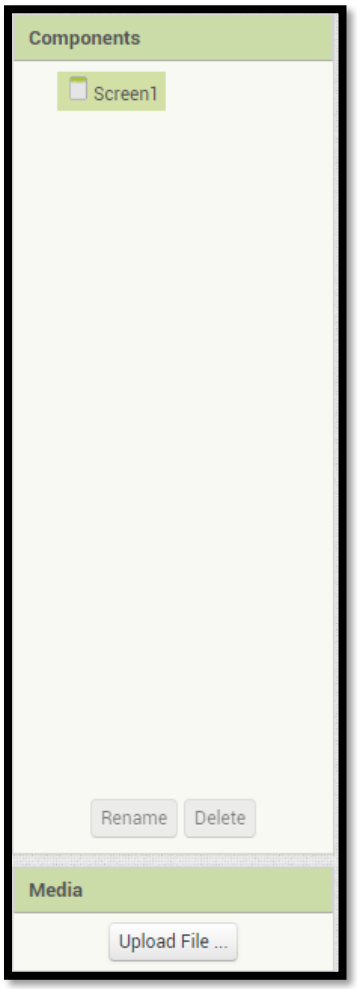


Figura 4 Components e Media

Nesta tela, também vai ficar armazenado os itens que você soltar no *Viewer*, porém eles irão ser dispostos em forma de listas no qual ao clicar em cima a propriedades deste item irá aparecer (próximo item). Também há o botão *Upload File...* logo abaixo do *Media*, nessa guia você poderá fazer o *upload* de imagens ou sons que pretende usar na aplicação.

2.4 Item 4 *Properties*

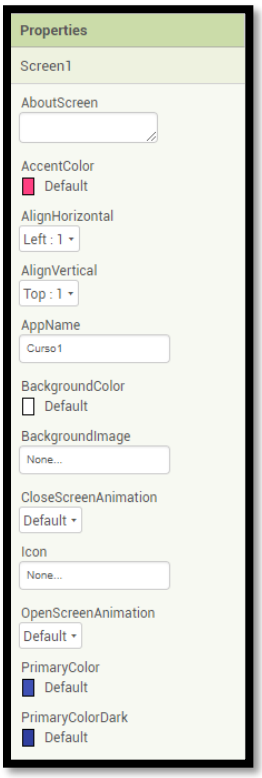


Figura 5 Properties

Como mencionado anteriormente, essa tela irá apresentar as propriedades do item que foi clicado, aqui você pode renomear o objeto para sua vontade, modificar altura, largura, texto dos botões, das *labels* entre outros itens.

3 Construindo o primeiro aplicativo!

O aplicativo que vamos desenvolver é bem simples, ao clicar em um botão, ele irá mostrar uma *label* chamada “Hello World”.

1º Passo: Vamos Começar um **novo projeto** clicando na opção *Projects -> Start New Project* conforme a imagem abaixo:

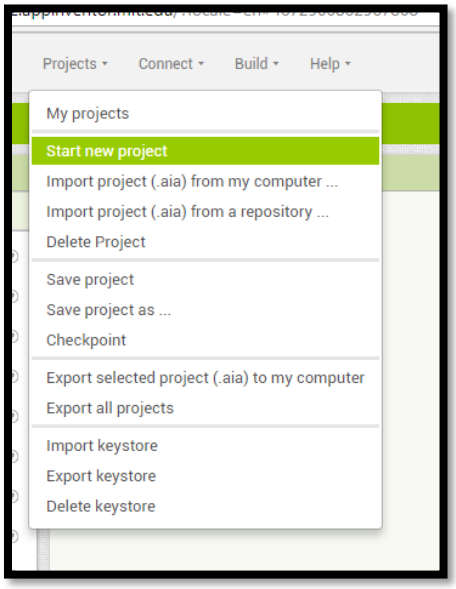


Figura 6 Novo Projeto

2º Passo: Agora arrastamos os itens *BUTTON* e *LABEL*, conforme ilustrado abaixo.

Figura 7 Label e Botão

3º Passo: Vamos modificar os textos tanto do botão quando da *label*, clicando respectivamente em cada um e indo para as *properties*.

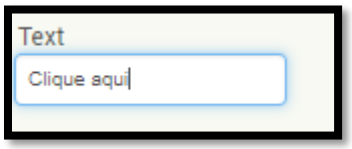


Figura 8 Opção text do botão

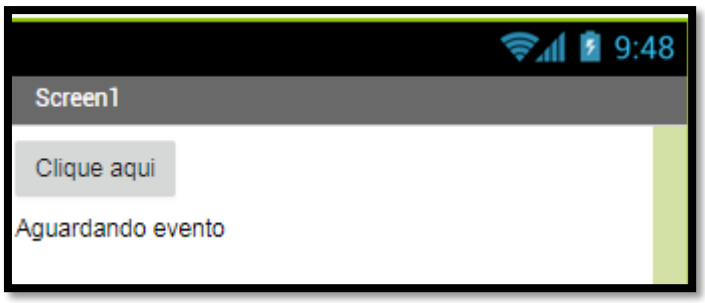


Figura 9 Ambas renomeadas

4º Passo: Agora iremos para a parte de blocos! Clique na opção *BLOCKS* no canto direito superior.

Figura 10 Botão Blocks

5º Passo: Vamos começar a nossa parte lógica!

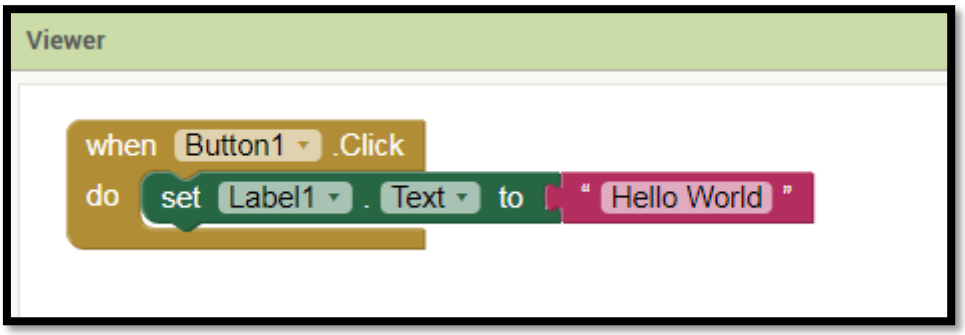


Figura 11 Lógica, controle e texto

Então, neste caso, usamos três estruturas, a primeira (dourada) é chama de estrutura de controle, a segunda (verde) é chamada de lógica e a terceira (roxa) é o texto que iremos apresentar.

6º Passo: Agora iremos fazer o *Build* do aplicativo e gerar o .apk, para isso vá: *Build-> App(save .apk to my computer) ou Build-> App(Provide QR code for .apk)* fica a seu critério!

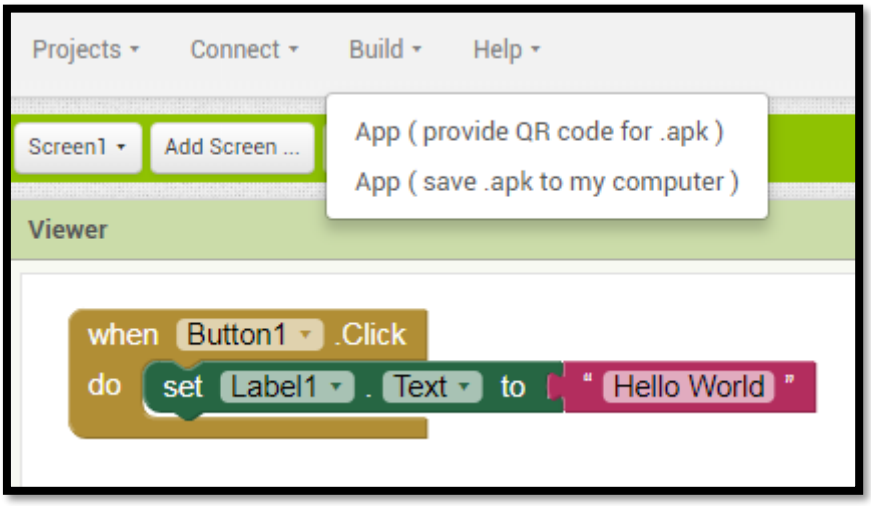


Figura 12 Build .apk

E tudo certo! Caso deseje ver o vídeo do aplicativo em funcionamento, acesse https://www.youtube.com/watch?v=jGAJ3OqXyMU

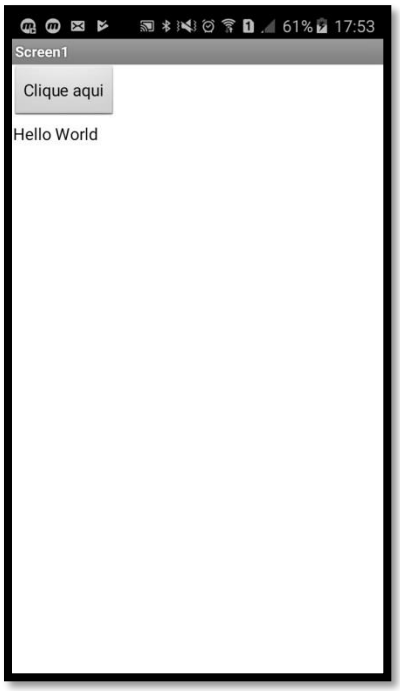


Figura 13 App Final

3.1 Construindo o segundo aplicativo!

O próximo aplicativo será o clássico calcula média.

1º Passo: Vamos Começar um **novo projeto** clicando na opção *Projects -> Start New Project* conforme a imagem abaixo:

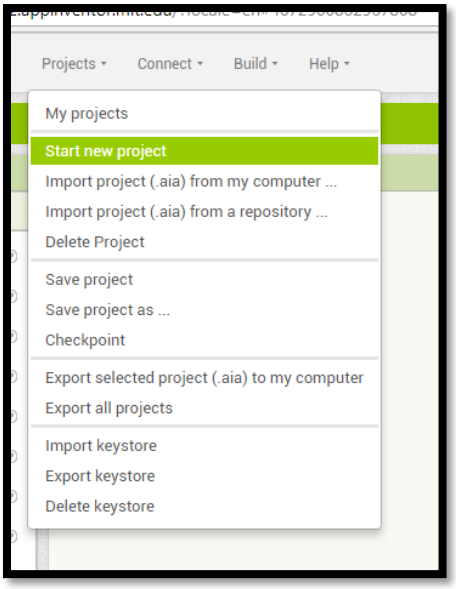


Figura 14 Novo Projeto2

2º Passo: Agora arrastamos os itens necessários.

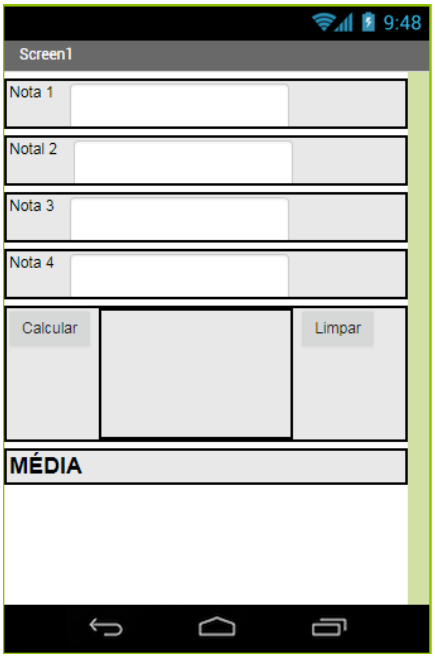


Figura 15 Design 2

4º Passo: Agora iremos para a parte de blocos! Clique na opção *BLOCKS* no canto direito superior.

Figura 16 Botão Blocks 2

5º Passo: Vamos começar a nossa parte lógica!

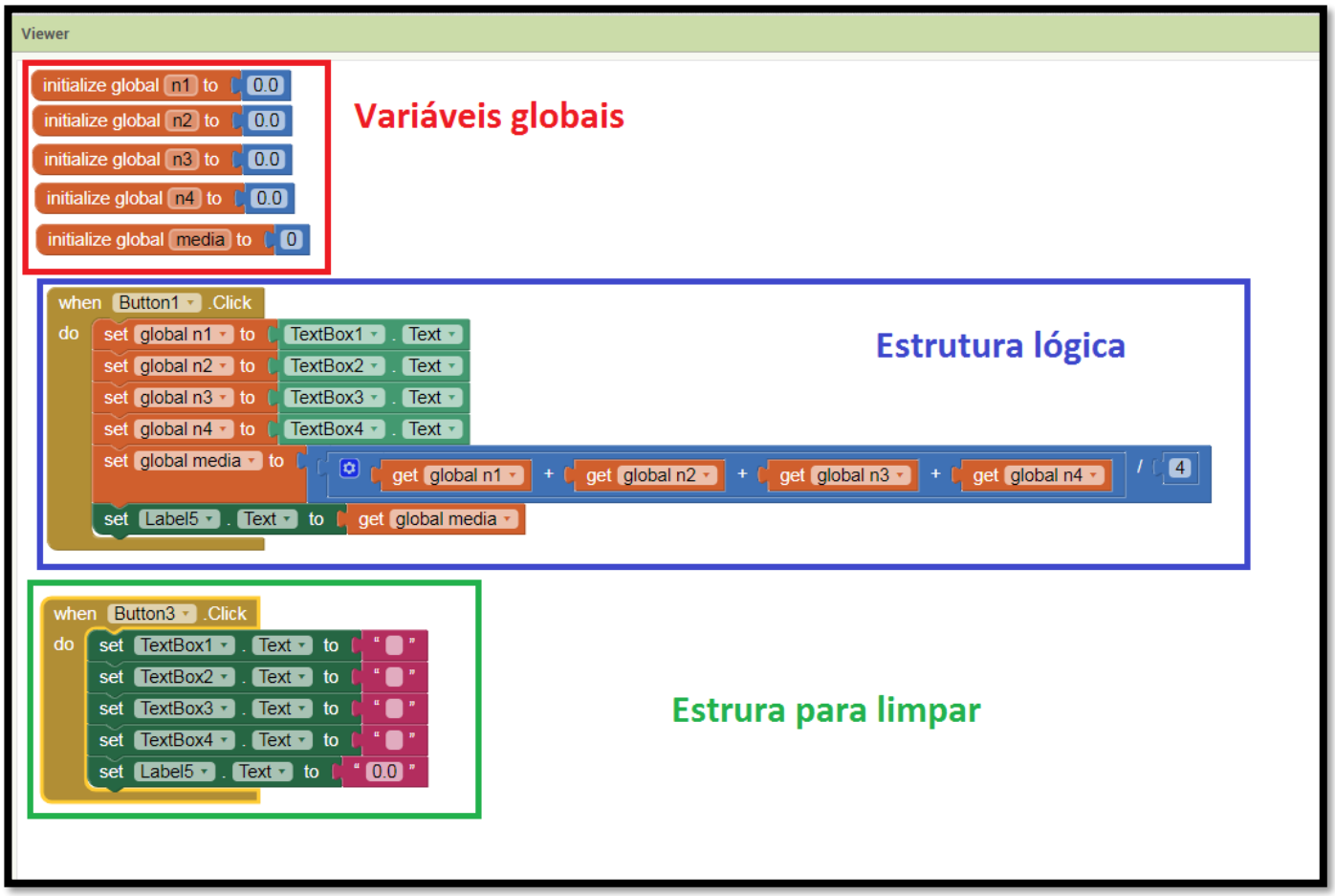


Figura 17 Codificação em blocos

6º Passo: Acesse https://www.youtube.com/watch?v=bZg7OLQMTtM para ver o aplicativo funcionando.

Figura 18 Aplicativo pronto

4 Exercícios - Básicos

1- Desenvolva um aplicativo que calcule a média de um aluno, caso a média seja maior que 6, deverá aparecer uma mensagem “Aluno Aprovado” e se for menor “Aluno reprovado”.

2- Faça um aplicativo que contenha a imagem de 4 animais, e ao pressionar a imagem do animal ele deverá emitir o som respectivo.

3-Dizemos que um número natural é triangular se ele é produto de três números naturais consecutivos. Exemplo: 120 é triangular, pois 4.5.6 = 120. Dado um inteiro não-negativo n, verificar se n é triangular.

4- Desenvolva um aplicativo que faça a conversão direta do bitcoin para real e dólar, adicione uma *webview* que irá carregar uma página que tenha a cotação do bitcoin.

5- Faça um tradutor de Português para Inglês – utilize o componente *YandexTranslate* localizado na paleta de *Media*.

6- Agora adicione um módulo que a leia a palavra traduzida – utilize o componente *TextToSpeech* localizado na paleta *Media*.

7- Adicione agora que ao invés de escrever, o usuário também terá a opção de falar a palavra – para isso utilize o componente *SpeechToText* localizado na paleta *Media.*

8-A entrada no mercado dos carros flex em 2003 ampliou o poder de escolha do consumidor, permitindo que ele migrasse do etanol para a gasolina e vice-versa conforme um ou outro combustível ficasse mais vantajoso para o bolso. No entanto, muitos motoristas ainda desconhecem quando devem optar pelo etanol e pela gasolina.

A resposta requer um cálculo simples. O uso do etanol é vantajoso se o litro custar até 70% do valor do litro da gasolina. Isso ocorre porque motores abastecidos com etanol consomem 30% a mais, em média, do que os abastecidos com gasolina.

A lógica principal deste APP consiste em comparar os valores do etanol e da gasolina usando uma estrutura condicional do tipo (if else). Podemos fazer o cálculo da porcentagem de forma muito simples multiplicando o valor da gasolina por 0.7 e para exibir o resultado vamos usar uma técnica que irá trocar a imagem indicando qual combustível é mais vantajoso.

9- Desenvolva um aplicativo que seja capaz de calcular a Lei de Ohm.

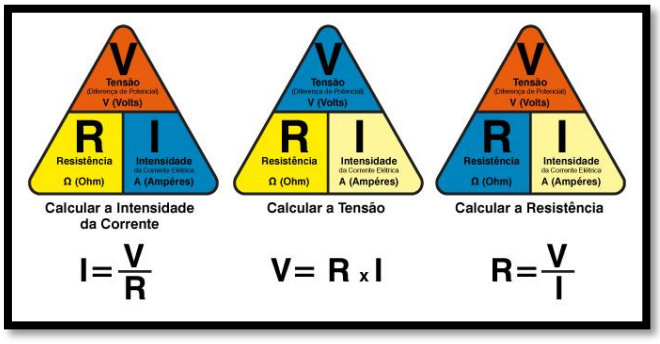
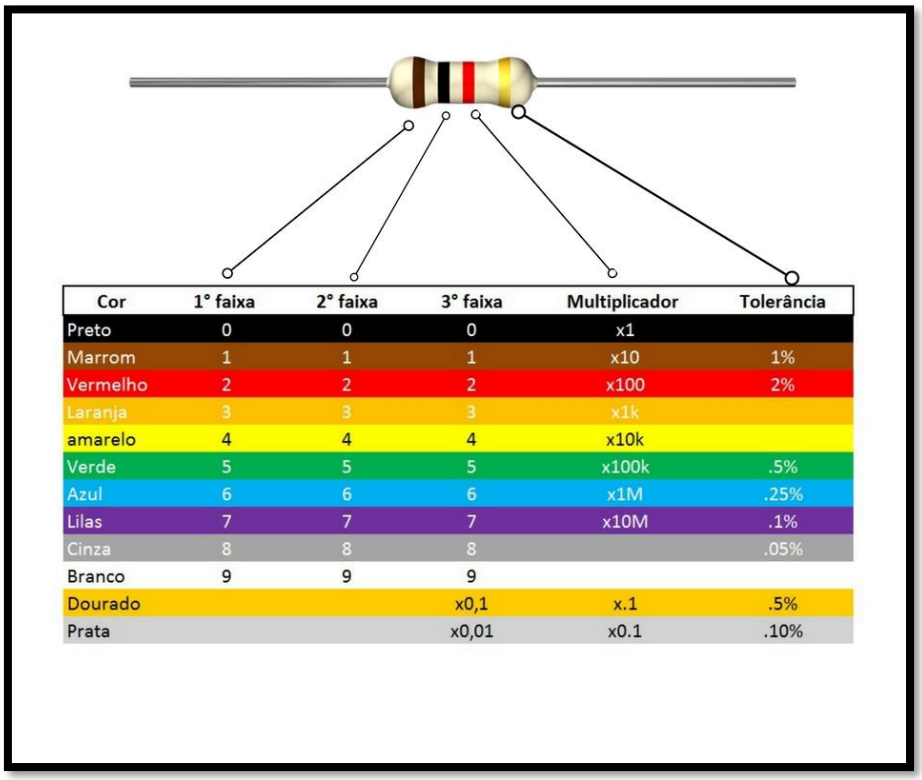


Figura 19 Lei de Ohm

10- Resistores são peças fundamentais na eletrônica, saber qual valor da resistência utilizar é essencial para que o seu projeto funcione perfeitamente, para isso estes componentes eletrônicos possuem faixas de cores que auxiliam na leitura, conforme a imagem abaixo:



Cor	1° faixa	2° faixa	3° faixa	Multiplicador	Tolerância
Preto	0	0	0	x1	
Marrom	1	1	1	x10	1%
Vermelho	2	2	2	x100	2%
Laranja	3	3	3	x1k	
amarelo	4	4	4	x10k	
Verde	5	5	5	x100k	.5%
Azul	6	6	6	x1M	.25%
Lilas	7	7	7	x10M	.1%
Cinza	8	8	8		.05%
Branco	9	9	9		
Dourado			x0,1	x.1	.5%
Prata			x0,01	x0.1	.10%

Figura 20 - Tabela de Cores

Monte um aplicativo que seja capaz de dar o valor da resistência correta, lembrando que resistores podem ter a terceira faixa para a leitura.

5 Projeto – Controle de led utilizando AppInventor e Arduino – HC-05 ou inferior

A plataforma *Android* é compatível com a pilha de rede *Bluetooth*, o que permite a um dispositivo permutar dados em comunicação sem fios com outros dispositivos *Bluetooth*. A estrutura de trabalho do aplicativo oferece acesso à funcionalidade do Bluetooth por meio *da Android Bluetooth API*. Essas APIs permitem a conexão sem fio de aplicativos a outros dispositivos Bluetooth, permitindo recursos sem fio ponto a ponto e multiponto.

Ao usar *Bluetooth* APIs, um aplicativo *Android* pode executar as seguintes atividades:

- Procurar outros dispositivos *Bluetooth*
- Consultar o adaptador *Bluetooth* local para verificar a existência de dispositivo *Bluetooth* pareada
- Estabelecer canais RFCOMM
- Conectar-se a outros dispositivos por meio da descoberta de serviços
- Transferir dados de e para outros dispositivos
- Gerenciar várias conexões

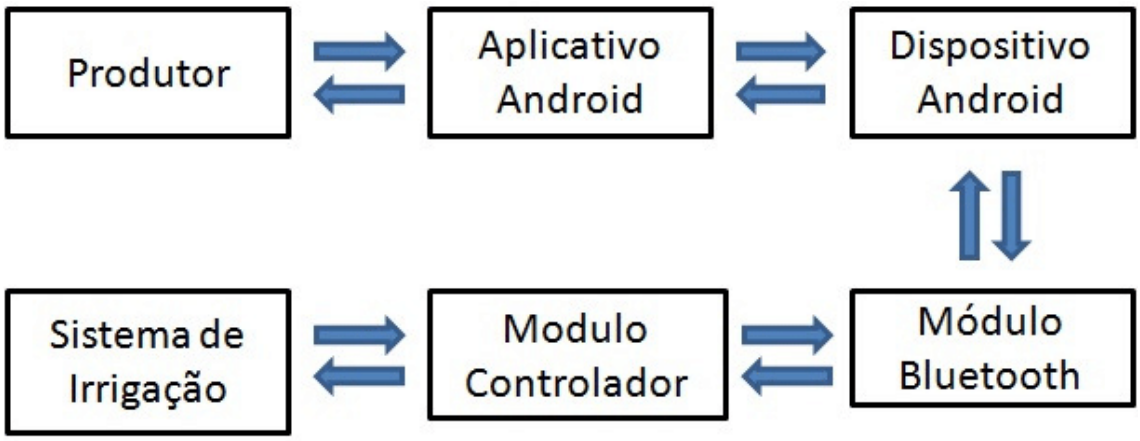


Figura 21 Fluxograma Bluetooth

5.1 Construindo o aplicativo para acender o LED

1º Passo: Vamos Começar um **novo projeto** clicando na opção *Projects -> Start New Project* conforme a imagem abaixo:

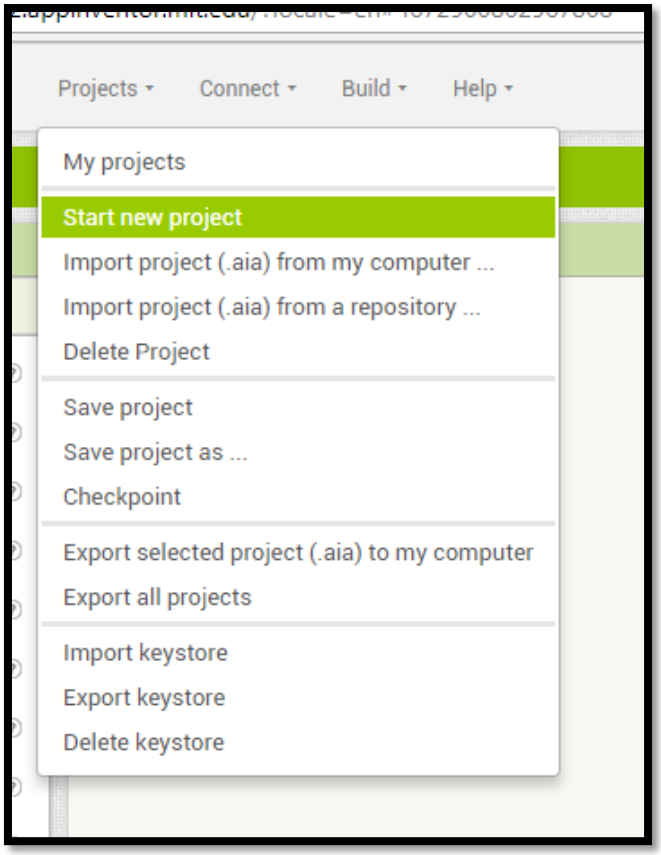


Figura 22 Aplicativo Bluetooth

2º Passo: Agora arrastamos os itens necessários.

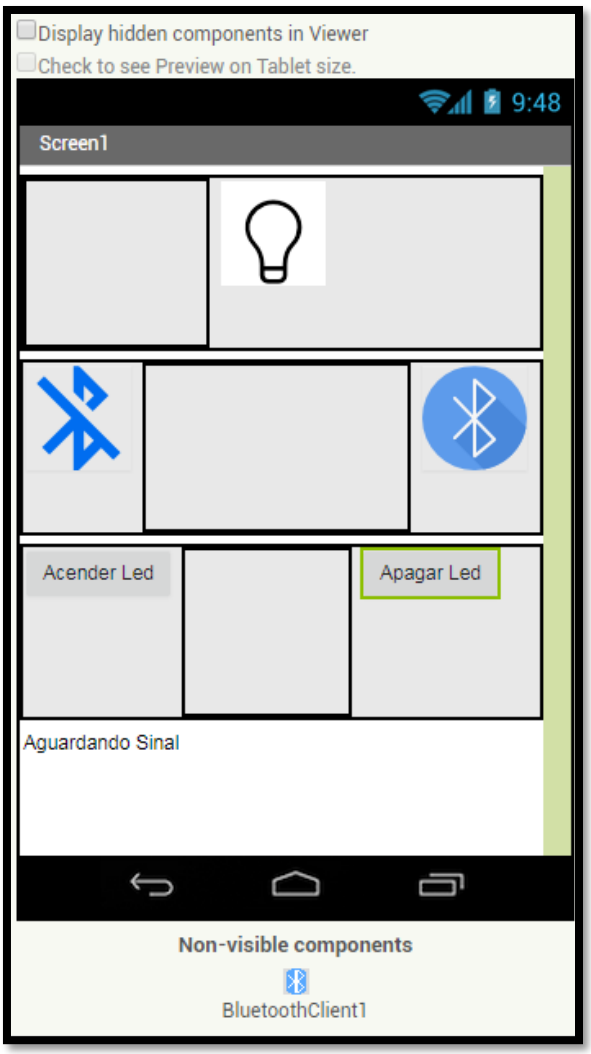


Figura 23 Design bluetooth

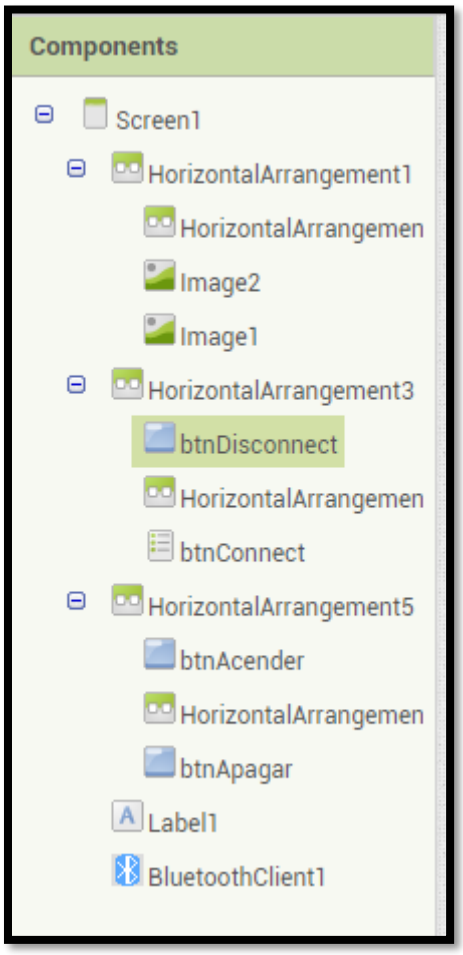


Figura 24 - Lista de Componentes

Lembrando que o componente *"btnConnect"* não é um botão comum e sim um *ListPicker*.

4º Passo: Agora iremos para a parte de blocos! Clique na opção *BLOCKS* no canto direito superior.

Figura 25 Botão Blocks Bluetooth

5º Passo: Vamos começar a nossa parte lógica!

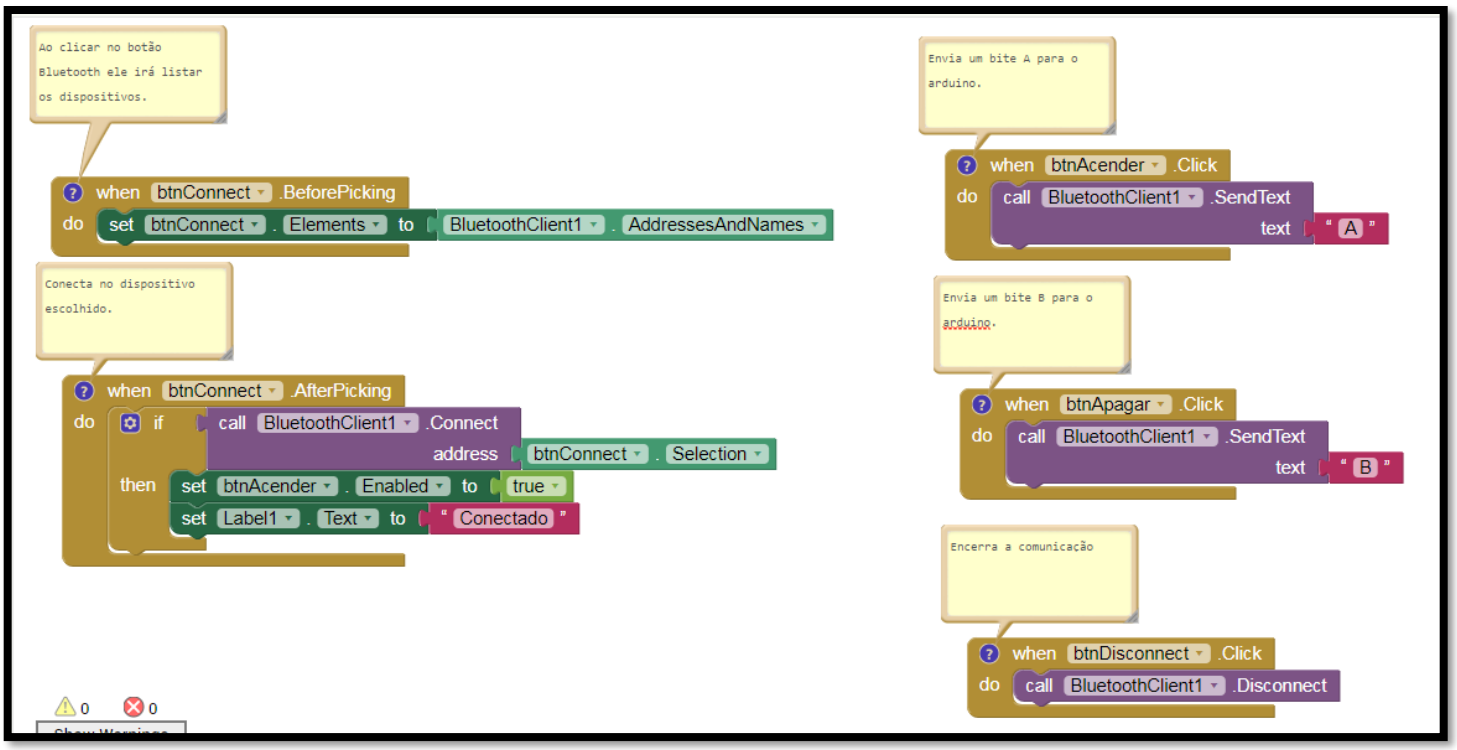


Figura 26 Blocks bluetooth

5.2 A parte do Arduino

No microcontrolador vamos como conectar o módulo *bluetooth*, para fazer isso siga o exemplo abaixo. Para ver o exemplo funcionando acesse:
https://youtu.be/OnfePZ0rwoU

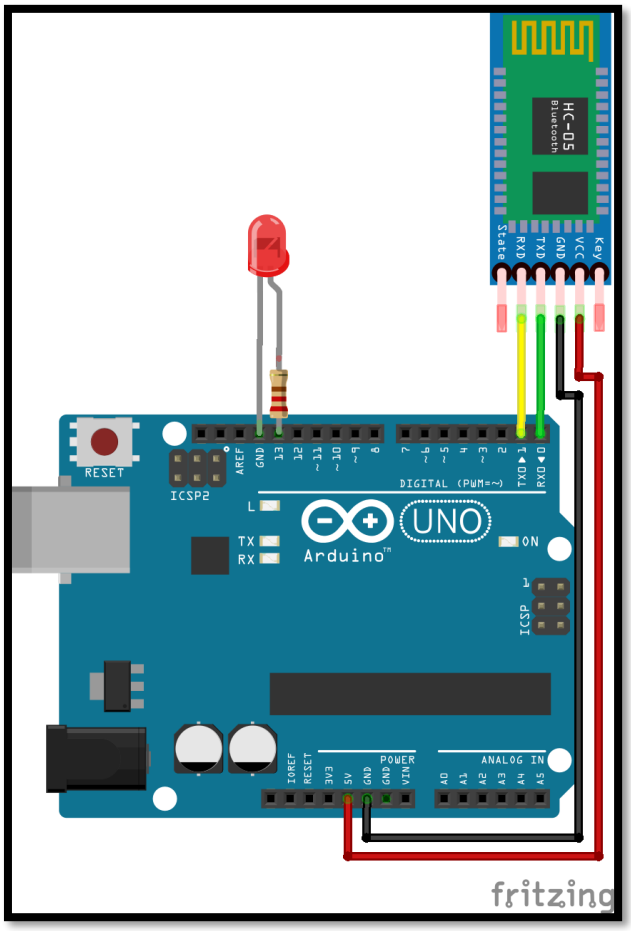


Figura 27 Conexão arduino

O código usado para este exemplo:

```
char data = 0;
void setup()
{
    Serial.begin(9600);
    pinMode(13, OUTPUT);
}
void loop()
{
   if(Serial.available() > 0)
   {
      data = Serial.read();
      Serial.print(data);
      Serial.print("\n");
      if( data=='1' || data=='A')
         digitalWrite(13, HIGH);
      else if( data== '0' || data=='B')
         digitalWrite(13, LOW);
```

```
    }
}
```

Tabela 1 Código Arduino

6 Projeto – Criando um aplicativo de chat com o Firebase

Neste tutorial você irá aprender a criar um aplicativo de mensagem instantânea aos moldes do *Whatsapp*. Este artigo foi adaptado do artigo em inglês de Meghra Singh Benwal, o qual seu site está nas referências.
Na nossa aplicação iremos usar o banco de dados na nuvem Firebase, disponibilizada gratuitamente pelo Google.

Vantagens do Firebase

1- *Smart* autenticação que fornece um acesso seguro.
2- Por ser na nuvem, pode ser acessível de qualquer aparelho.
3- É um banco de dados em tempo real ou seja, demonstra as modificações assim que for atualizado.
4- Usuários podem adicionar colaboradores.
5- Possui uma interface gráfica que permite manipular o banco de dados visualmente.
6- Não há código apenas configuração.

6.1 Criando e configurando um projeto no Firebase

1º Passo: Acesse *https://console.firebase.google.com/* , lembrando que o Firebase trabalha com contas Google assim como o App Inventor 2

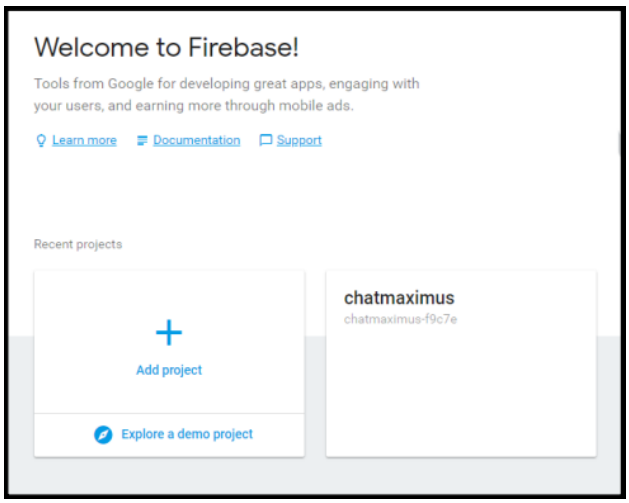


Figura 28 - Tela inicial Firebase

2º Passo: Clique na opção *Add Project*, dê um nome ao projeto e escolha o país.

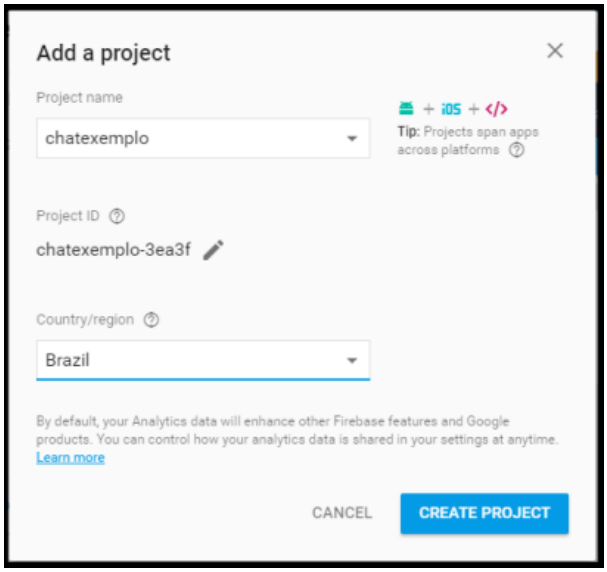


Figura 29 - Criando um Projeto

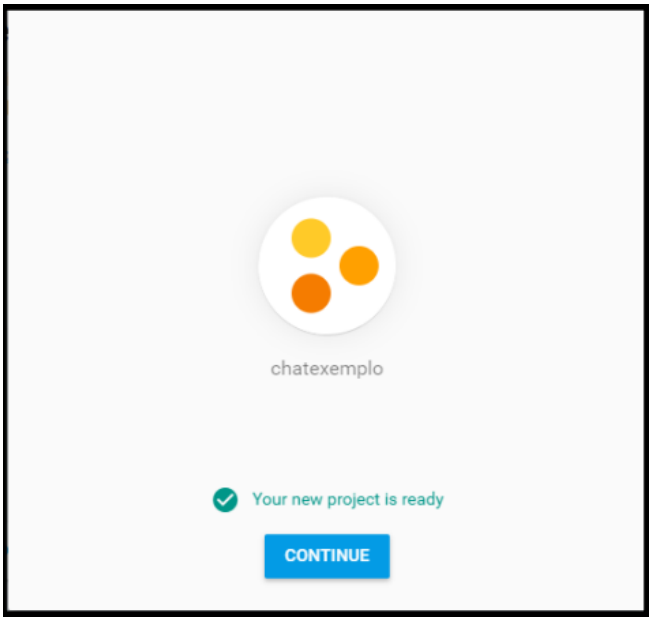


Figura 30 - Finalização do projeto

3º Passo: Nessa tela, olhe na palheta a esquerda e escolha a opção *database* e depois *"Real Time Database"*

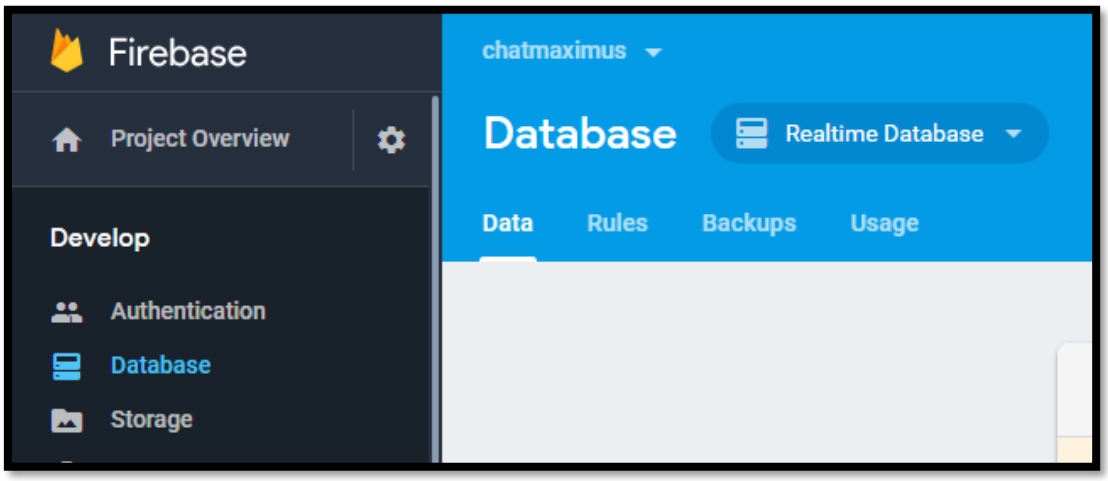


Figura 31 - Configurações do Projeto

4º Passo: Agora vá na opção "*rules*" e deixe exatamente igual a imagem abaixo:

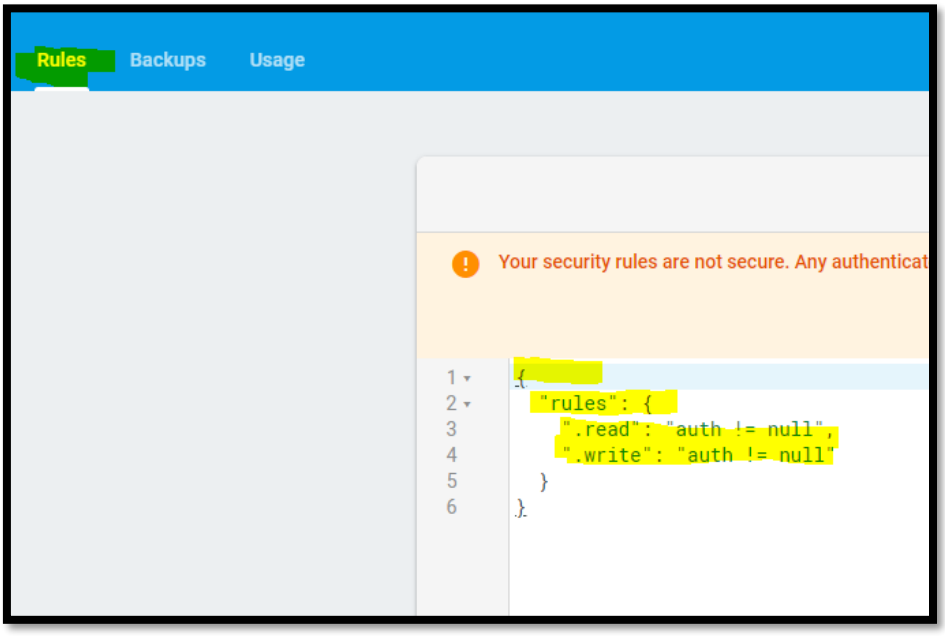


Figura 32 - Níveis de Segurança

6.2 App Inventor 2

1º Passo: Vá em http://ai2.appinventor.mit.edu/, crie um novo projeto e deixe-o similar a figura abaixo:

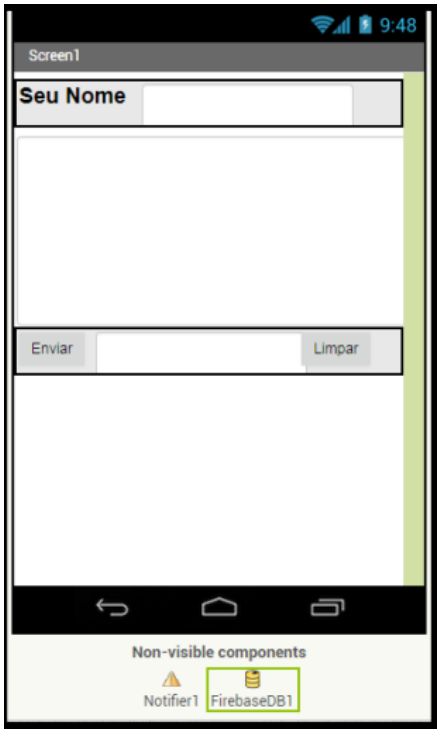


Figura 33 - Interface do usuário

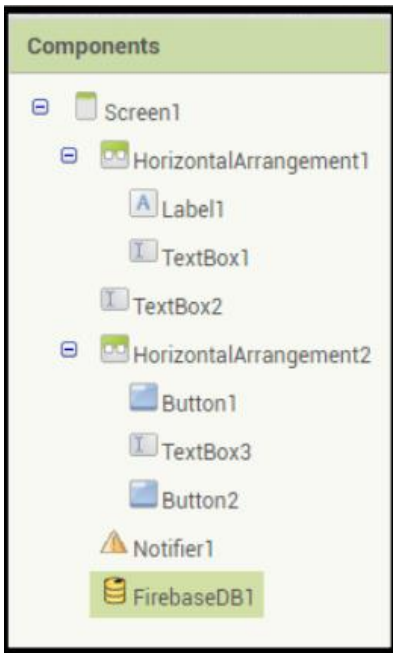


Figura 34 – Palheta de objetos

2º Passo: Configurando a opção “FirebaseDB1” do AppInventor2 – o objeto FirebaseDB1 vai estar localizado na opção de *EXPERIMENTAL* conforme a imagem:

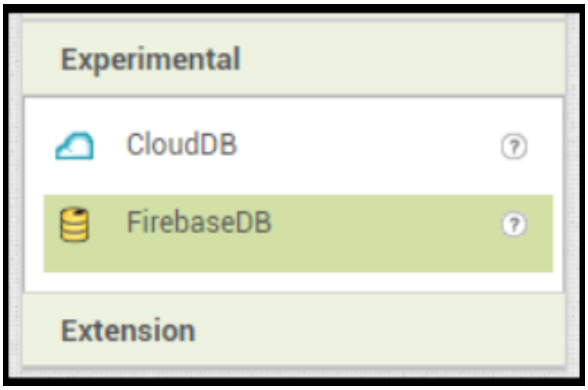


Figura 35- Localização Firebase

3º Passo: Agora vá nas propriedades do objeto FirebaseDB e configure a opção FirebaseURL.

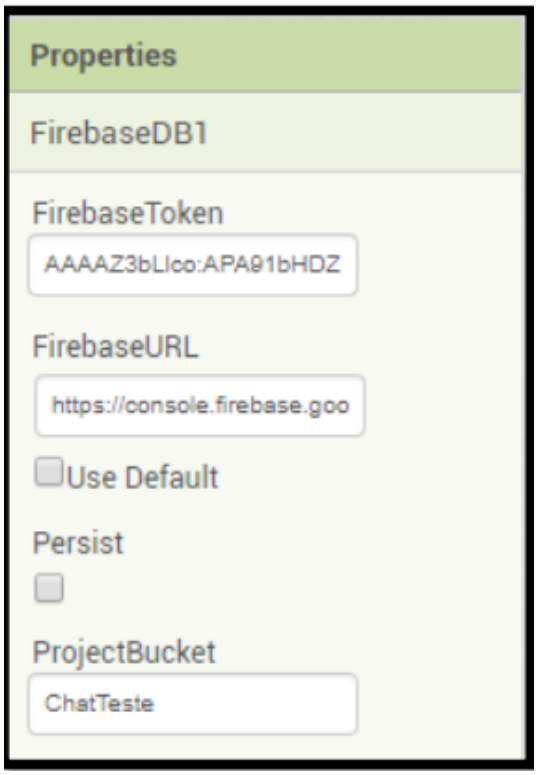


Figura 36 - Propriedades do FirebaseDB

4º Passo: Para conseguir a firebase URL, volte ao site Firebase, clique na opção “Database” e clique no link que fica localizado dentro da aba “DATA”

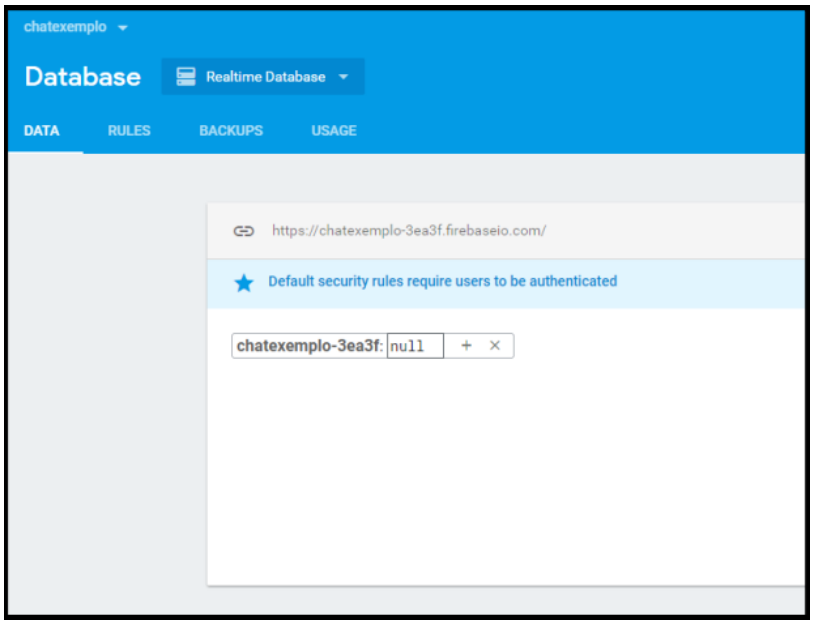


Figura 37 - Firebase URL

5º Passo: Vamos começar a nossa codificação em blocos

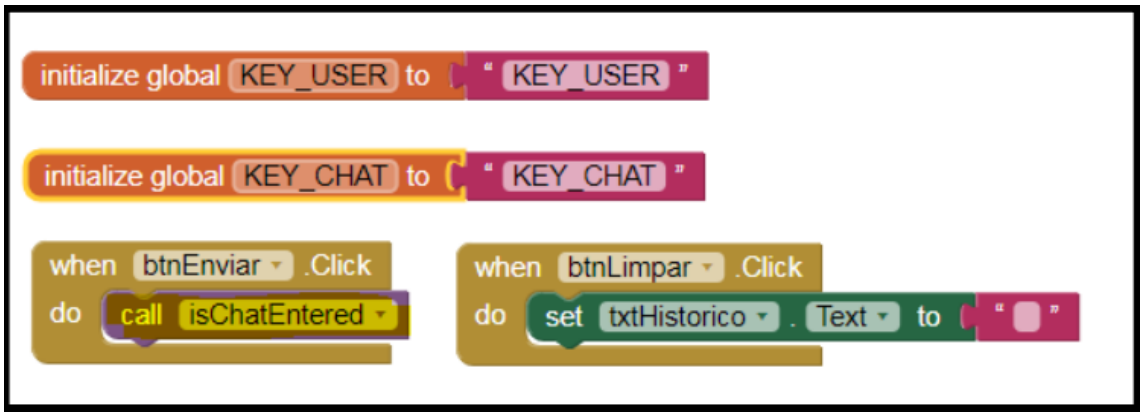


Figura 38 - Parte lógica

Primeiro criamos duas variáveis globais a “KEY_USER” e a “KEY_CHAT” ambas responsáveis por receber nosso usuário e mensagens.
Depois o botão limpar simplesmente apaga todas as mensagens que há no histórico. O botão Enviar está chamando uma função que vamos implementar mais a frente,
por enquanto apenas pegue uma *procedure* na palheta e renomeie para *isChatEntered*.

7º Passo: Procedimento para testar se há alguém conectado

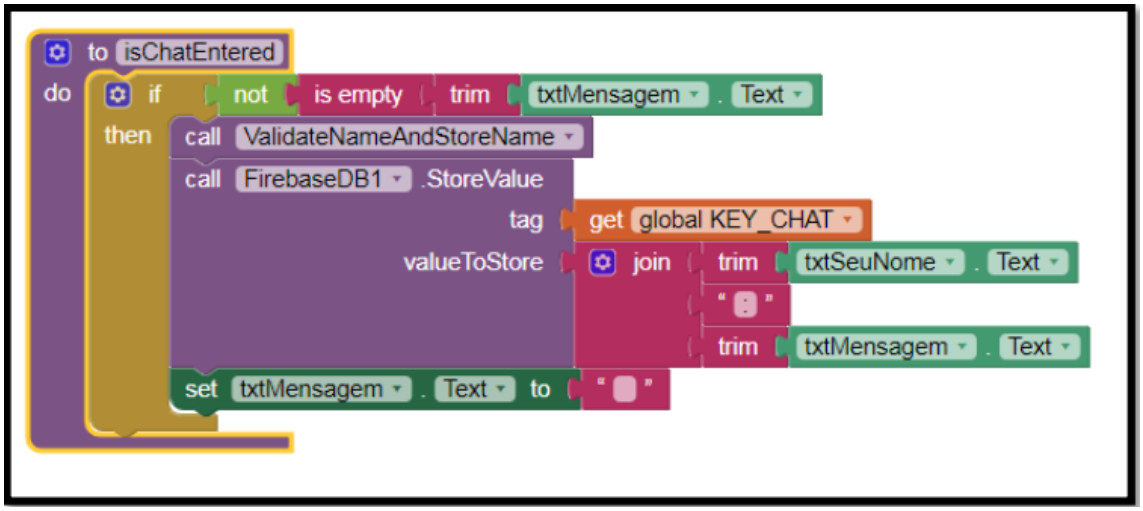


Figura 39 - Função para verificar se há alguém conectado

Apesar de ser grande esse bloco não é nada complexo, ele primeiro verifica se existe algo digitado na mensagem. Caso exista ele chama a função que conecta com o banco de dados e envia a mensagem com o nome concatenado. O mesmo se repete aqui, vamos criar posteriormente a função *ValidateNameAndStoreName*.

8º Passo: Validação do usuário;

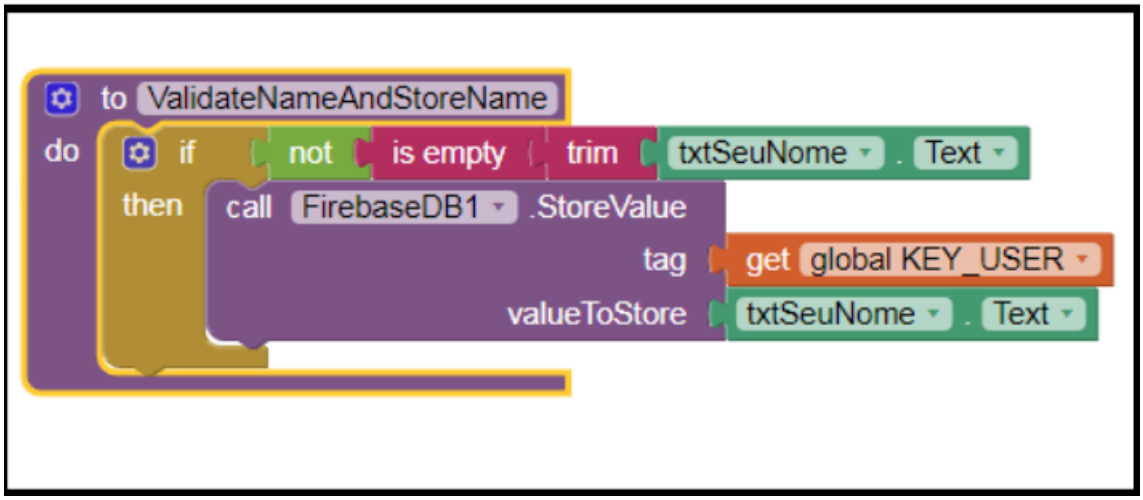


Figura 40- Validação do Usuário

9º Passo: Atualização do banco de dados

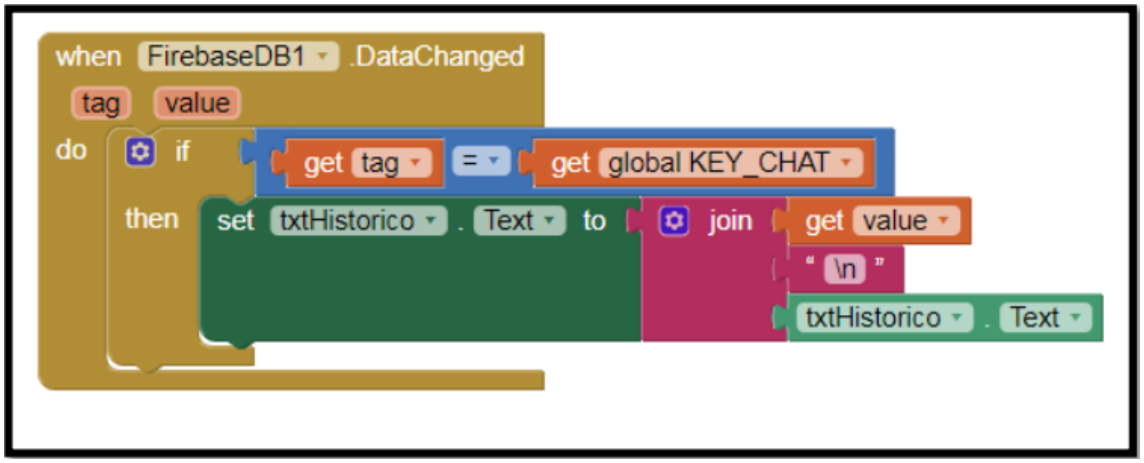


Figura 41 - Atualização do Banco de Dados

6.3 Conclusão

Esse aplicativo é capaz de mandar simples mensagens de texto de um celular ao outro, como melhoria futura será implementada a capacidade de enviar mídia, como por exemplo, uma foto ou áudio.

7 Projeto – Space Invaders

7.1 Introdução

Neste tutorial iremos aprender a desenvolver o clássico jogo Space Invaders utilizando a plataforma do MIT. Neste tutorial você irá aprender as seguintes habilidades:

- Usar o componente *clock*
- Utilizar *timer* para movimentar os *sprites*
- Detectar colisão
- Definir Visibilidade de *sprites*

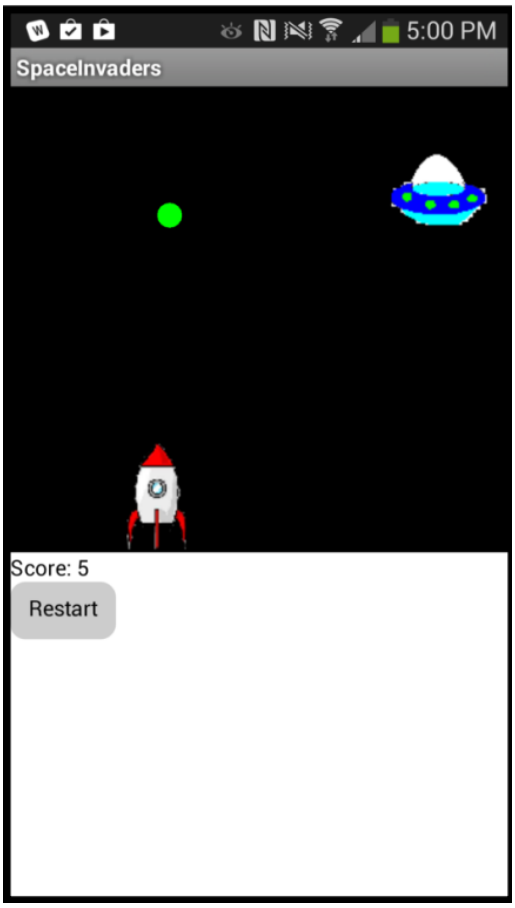


Figura 42 - Space Invaders

7.2 Metodologia

1º Passo: Vá ao site https://www.flaticon.com/ e escolha dois sprites (png) de uma nave amiga e uma nave inimiga.

Figura 43 - Exemplo de Sprite

2º Passo: Monte a interface do jogo conforme o modelo abaixo.

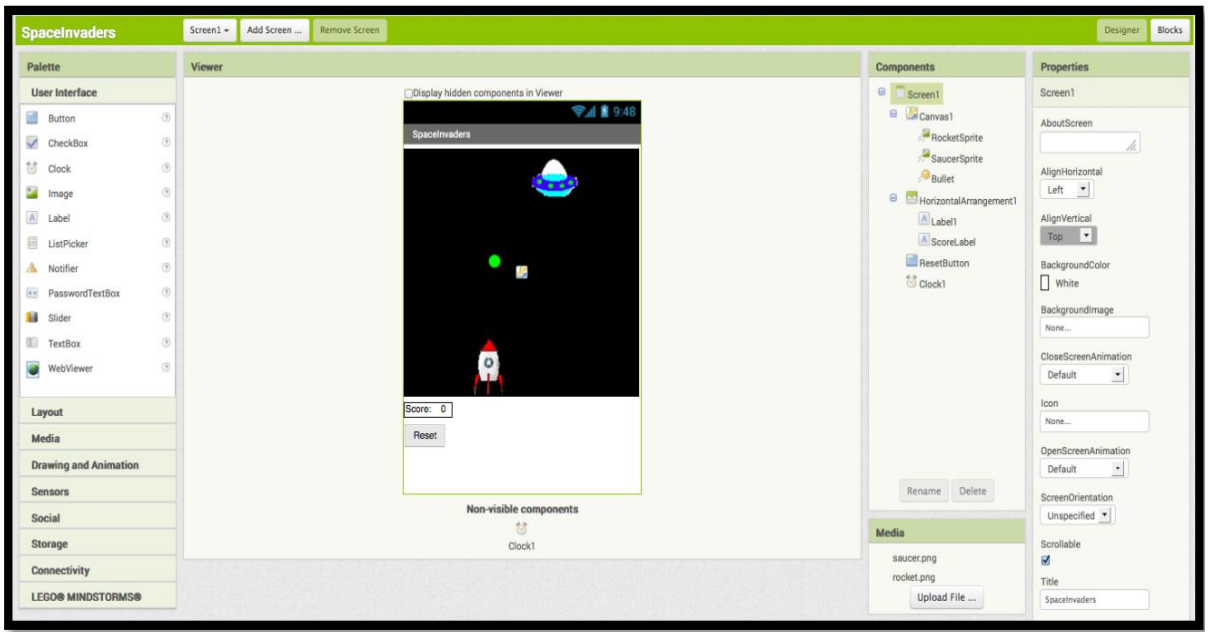


Figura 44 - Interface do jogo

Tipo do Componente	Pallet Group	Nomenclatura
Canvas	Drawing and Animation	Canvas1
ImageSprite	Drawing and Animation	RocketSprite
ImageSprite	Drawing and Animation	SaucerSprite
BallSprite	Drawing and Animation	Bullet
Clock	User Interface	Clock1
Horizontal Arrangement	Layout	HorizontalArrangement1
Label	User Interface	Label1
Label	User Interface	ScoreLabel
Button	User Interface	ResetButton

3º Passo: Codificação em blocos

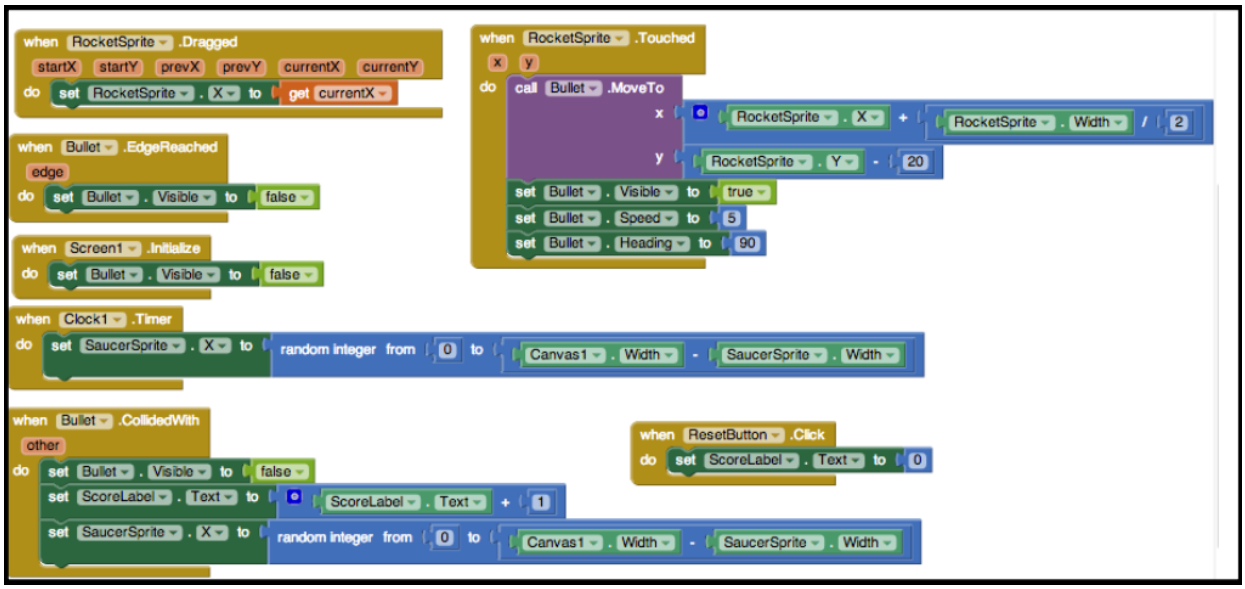


Figura 45- Blocos

8 Projeto - Bloco de Notas

Nesta seção iremos fazer um bloco de notas, no qual você poderá inserir uma anotação seguido da data e receberá uma notificação quando for salva.

Você irá aprender a utilizar:

- Banco de Dados *TinyDB*
- *Date Picker*
- Importar e utilizar uma extensão no seu projeto

8.1 Metodologia

1º Passo: Crie um projeto novo no App Inventor (http://ai2.appinventor.mit.edu) e arraste os seguintes componentes da palheta:

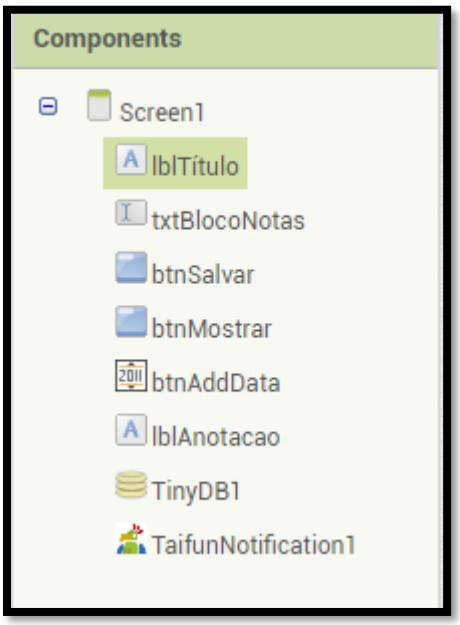


Figura 46 - Componentes

Por enquanto, pule o componente “TaifunNotification1” pois ele é uma extensão e ainda não adicionamos, logo ele não está disponível.
O componente “btnAddData” apesar de ser um botão, ele é localizado como “DATE PICKER”.

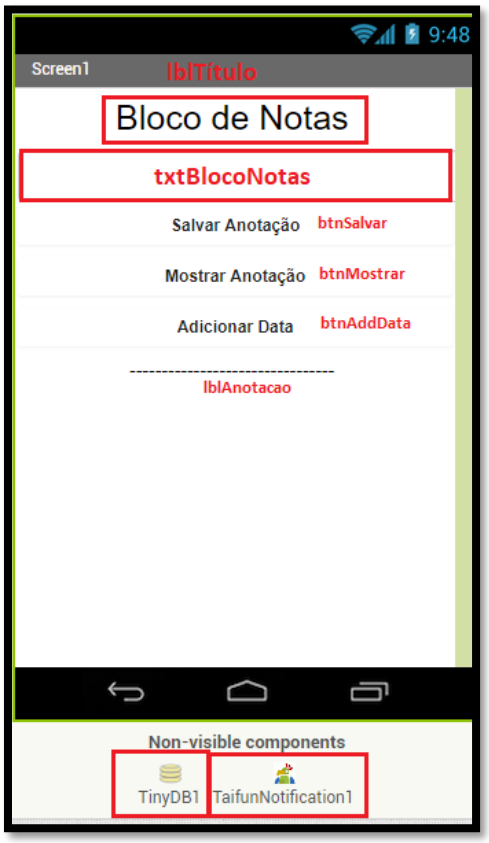


Figura 47 - Interface para o usuário

2º Passo: Adicione o método “when btnSalvar.Click” para área em branco, conforme a imagem:

Figura 48- Método Click

3º Passo: Adicione o método para inserir no banco de dados, para isso clique no componente “TinyDB1” e arraste “Call TinyDB1.StoreValue” e adicione dois componentes, um de texto chamado “data” e o texto do nosso campo “txtBlocoNotas”

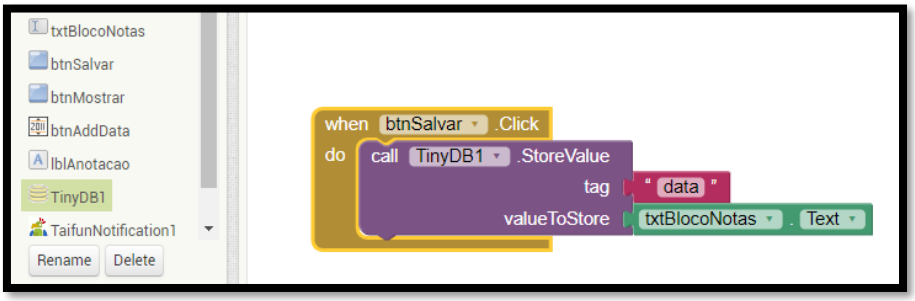


Figura 49 - Botão salvar finalizado

Neste caso, a “tag” é a nossa chave primária – o que vai identificar o dado para o banco de dados. E o “valuToStore” é o valor propriamente dito.

4º Passo: Vamos implementar o método para mostrar o valor salvo no banco, para isso:

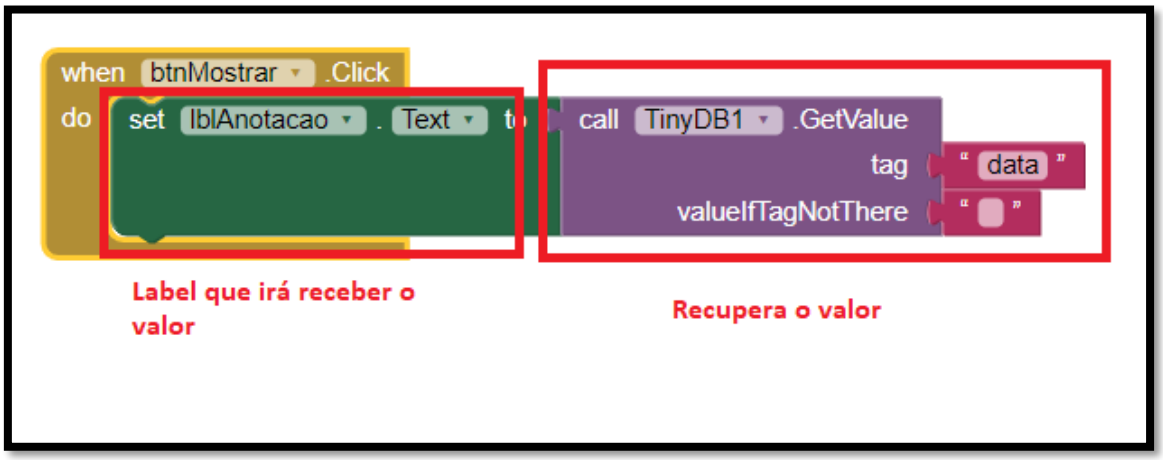


Figura 50 - Botão recuperar

O método “Click” do botão Mostrar irá receber o valor do banco de dados e posteriormente adiciona-lo como texto na label.

Ao executar o projeto, você irá perceber que está funcionando, porém ainda está faltando a implementação da data.

5º Passo: Vamos fazer o método para inserir a data no nosso campo de texto, para isso vá no objeto “btnAddData” e escolha a opção “When.btnAddData.TouchDown”

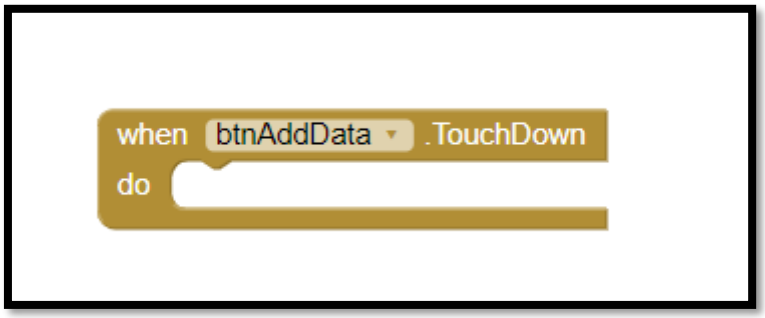


Figura 51 - Método TouchDown

6º Passo: Agora vem a formatação, a ideia é que a mensagem fique neste formato “30/12/2099 – Olá Mundo” para isso vamos formatar nosso campo de texto para que ele receba a mensagem:.

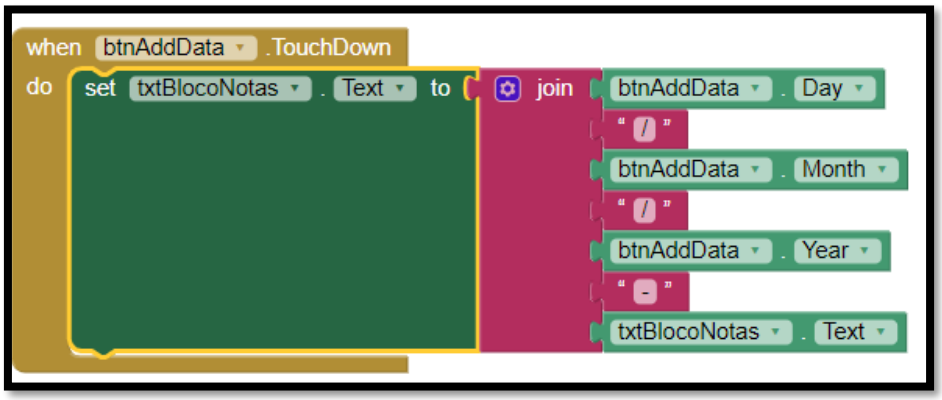


Figura 52 - Método addData completo

7º Passo: Enviar uma notificação para o usuário assim que a mensagem foi salva. Para que isso seja possível, vamos adicionar uma extensão ou biblioteca, como preferir chamar. O site que possui a maior gama de extensões é o Pura Vida Apps – a grande maioria é desenvolvida em JavaScript. Acesse: https://puravidaapps.com/extensions.php

Procure pela extensão “Simple Notification Extension”

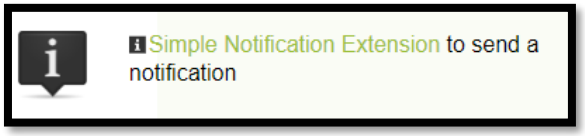


Figura 53 - Extensão

A próxima página, deverá sempre ser sua melhor amiga, é a documentação da biblioteca – e isso se aplica a todas as tecnologias – sempre olhe os manuais! Desça até o fim da página e clique na opção “Download TaifunNotification extension (aix file)”

Figura 54 - Página de Download

Após fazer o *download*, vá no app inventor e volte para a opção *designer*, na palheta de componentes vá na ultima opção *Extension* e clique em *import extension*.

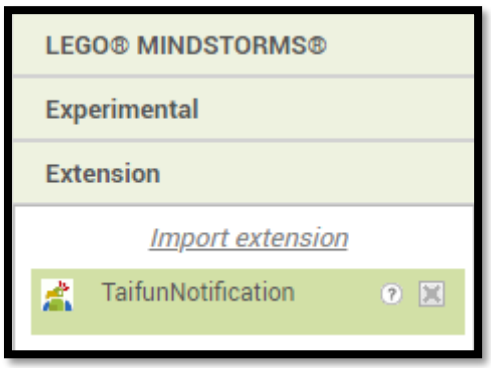


Figura 55 - Adicionar extensão

Uma vez adicionada a extensão, arraste-a para o aplicativo e comece a usa-la normalmente.

8º Passo: A mensagem ou notificação, será enviada 1 segundo após clicar no botão salvar, mas iremos adicionar uma imagem a essa notificação, então em “Designer” clique no componente ”TaifunNotification1” e vá na suas propriedades (última paleta a direita) e marque a opção “DisplayBigPicture” e faça o upload de uma imagem.

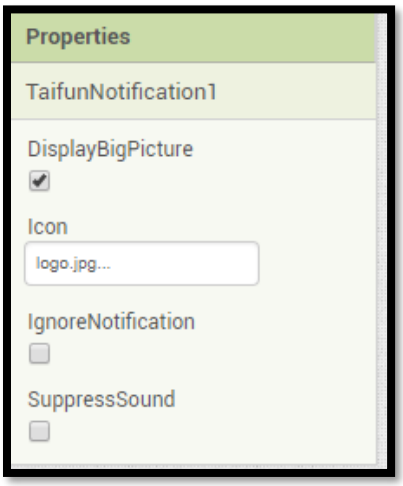


Figura 56 - Proprieades da extensão

9º Passo: Adicione o método “call TaifunNotification1.Send” e preencha de acordo com o que é solicitado,

Seconds: Tempo para enviar a notificação após clicar no botão salvar.

Title: Título da notificação.

Text: Texto que irá inserir.

StartText: Deixe com um texto vazio.

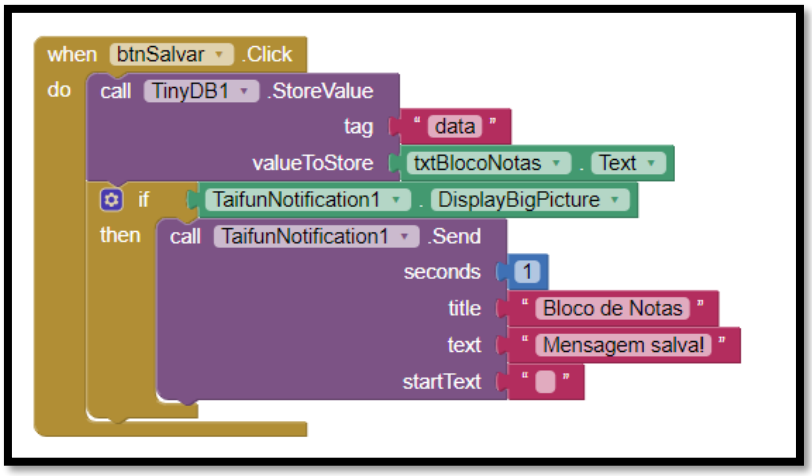


Figura 57 - Adicionando notificação

8.2 Conclusão

Agora basta gerar o código QR Code ou baixar o arquivo .apk e testar!

Figura 58 - Aplicativo Final

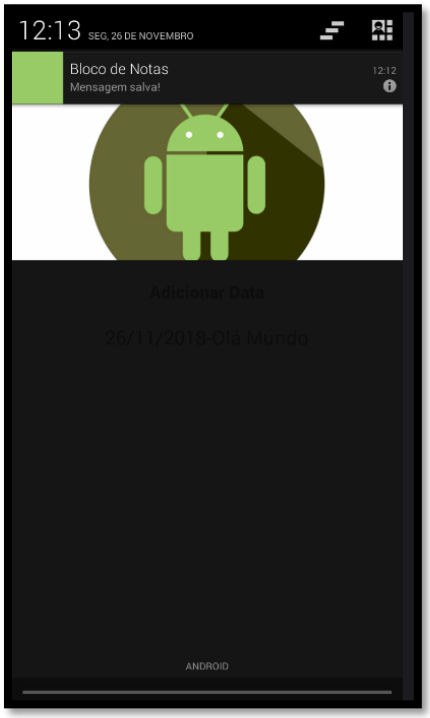


Figura 59 – Notificação

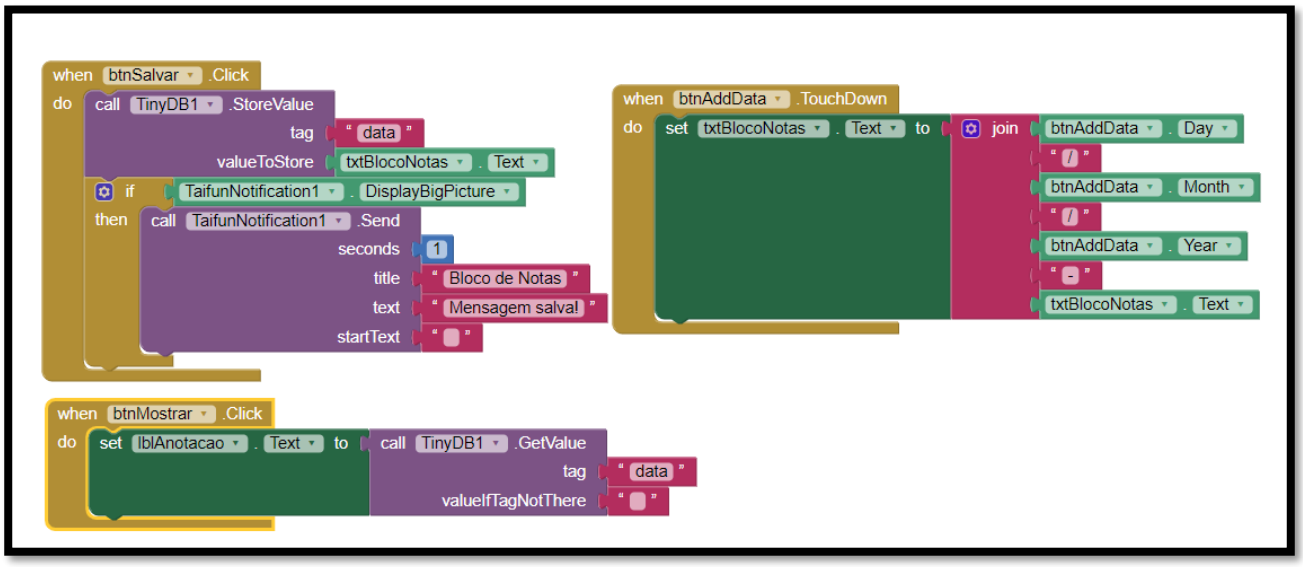


Figura 60 - Código completo

9 Projeto – Dicionário Interativo

Neste projeto você irá desenvolver um dicionário interativo, que ao buscar a palavra o aplicativo irá trazer a definição, a imagem, disparar o som que o objeto faz e o próprio aplicativo irá falar a definição.

- Manipular Listas (*arraylists*).
- Inserir sons e imagens dinamicamente.
- Manipular listas encadeadas e seus índices.

9.1 Design do aplicativo e blocos

1º Passo: Para o desenvolvimento do aplicativo vamos utilizar os seguintes componentes.

- 2 Labels
- 1 Campo de Texto
- 2 *botões*
- 1 Image
- 1 *Sound* (Não visível)
- 1 *Text to Speech* (Não visível)

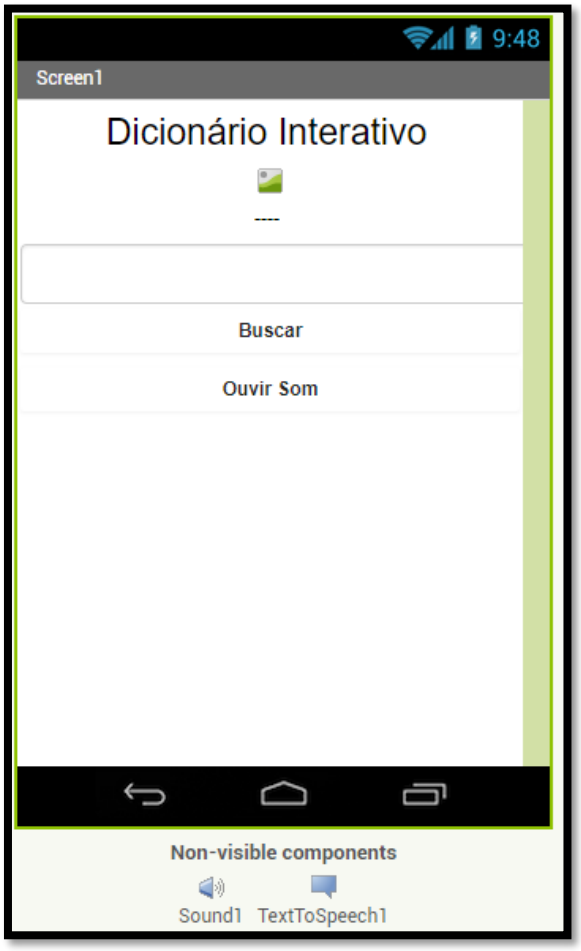


Figura 61 - Design Aplicativo

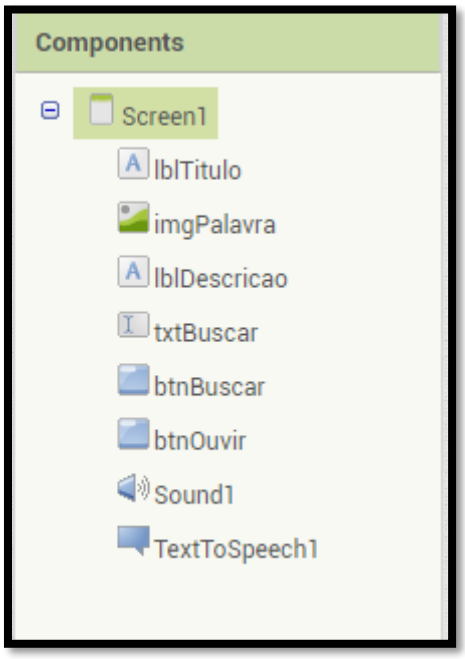


Figura 62 - Árvore de Componentes

2º Passo: Uma vez montado o design vamos para o algoritmo, crie duas variáveis globais uma chamada “listaDePalavras” e a segunda de “listaDeSignificados” e inicialize elas com os elementos da lista “make a list”, conforme abaixo:

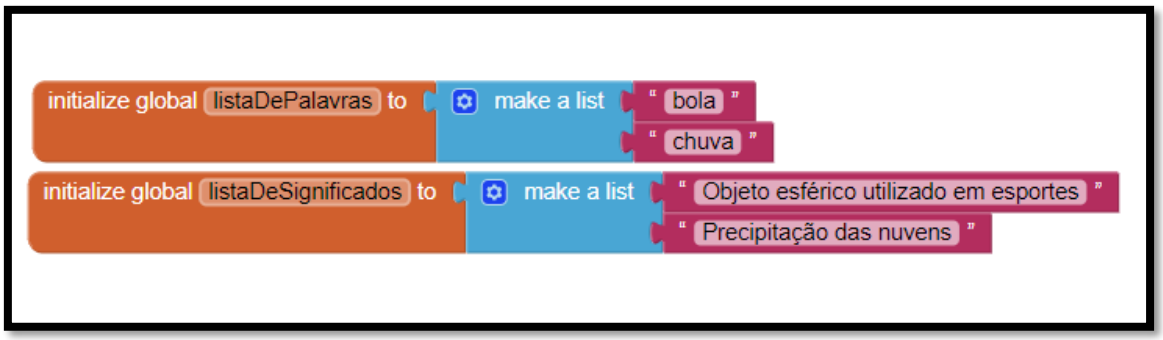


Figura 63 - Inicializando a lista

3º Passo: Arraste o método “*when bnBuscar.Click*” para área em branco.

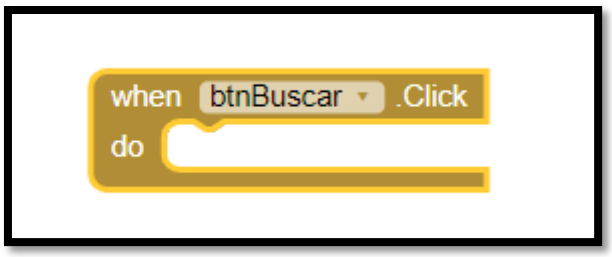


Figura 64 - Método para busca

4º Passo: Agora precisamos dizer para a lista buscar a palavra que foi digitada, para isso ela vai usar como referencia o que foi digitado no campo de busca, se encontrar a palavra, traz ela e a descrição, caso não, informa que a palavra não foi encontrada.

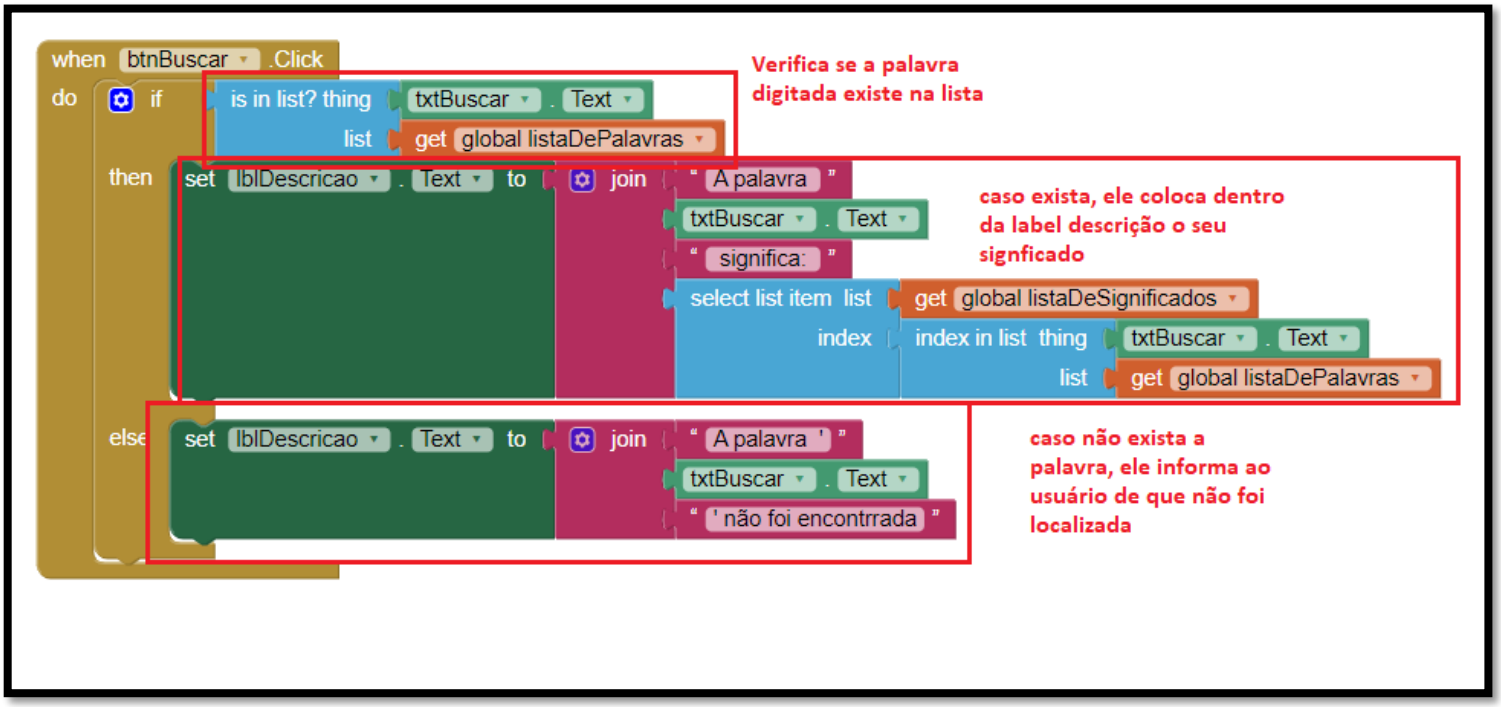


Figura 65 - Localização da palavra

Prestar atenção no segundo bloco, aonde se tem o item “*select list item list index*” neste caso, estamos passando como parâmetro (número de índice igual 1) quando buscamos o texto digitado com a palavra, ao invés de retornar a palavra, retorna o número do índice que é 1 – lembrando que as listas no AppInventor começam em 1 e não em 0.

5º Passo: Vamos carregar as imagens e os efeitos sonoros quando a palavra for buscada, então vamos fazer o upload de duas imagens, uma chuva e uma bola qualquer e para os sons pode se efetuar uma busca no youtube para “efeito sonoro chuva/bola” e utilizar o site https://ytmp3.cc/ para fazer a conversão do vídeo para .mp3

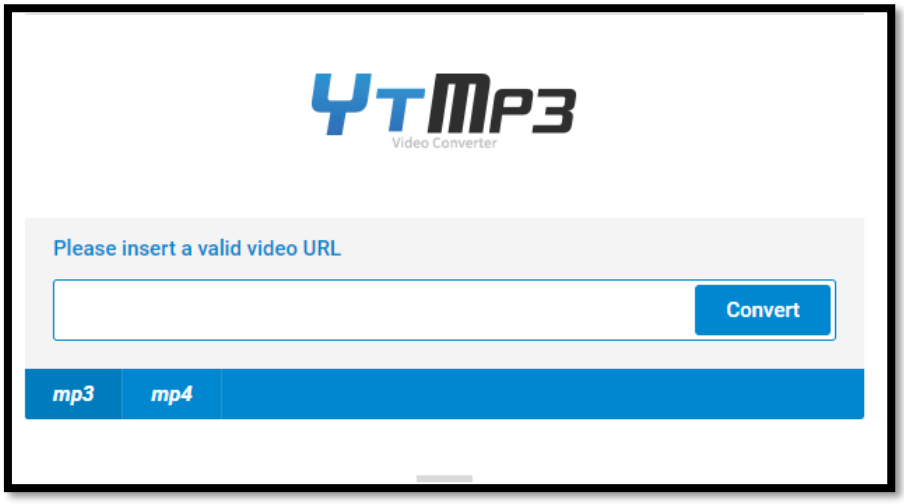


Figura 66 - Site para conversão

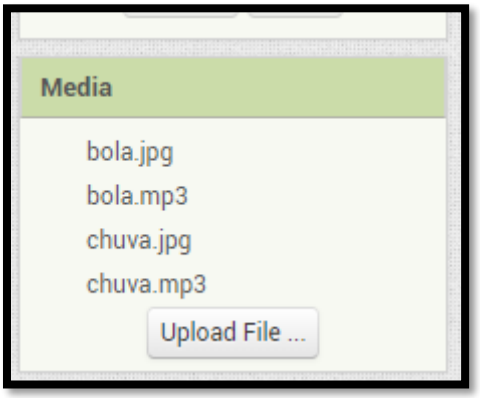


Figura 67 - Mídias carregadas

6º Passo: Para adicionar as imagens basta clicar no componente “imgPalavra” e escolher a função “*set imgPalavra.Picture to*” o segredo de mudar a imagem esta no campo de texto, para isso pegamos o que foi digitado no campo de texto e adicionamos a extensão, conforme abaixo:

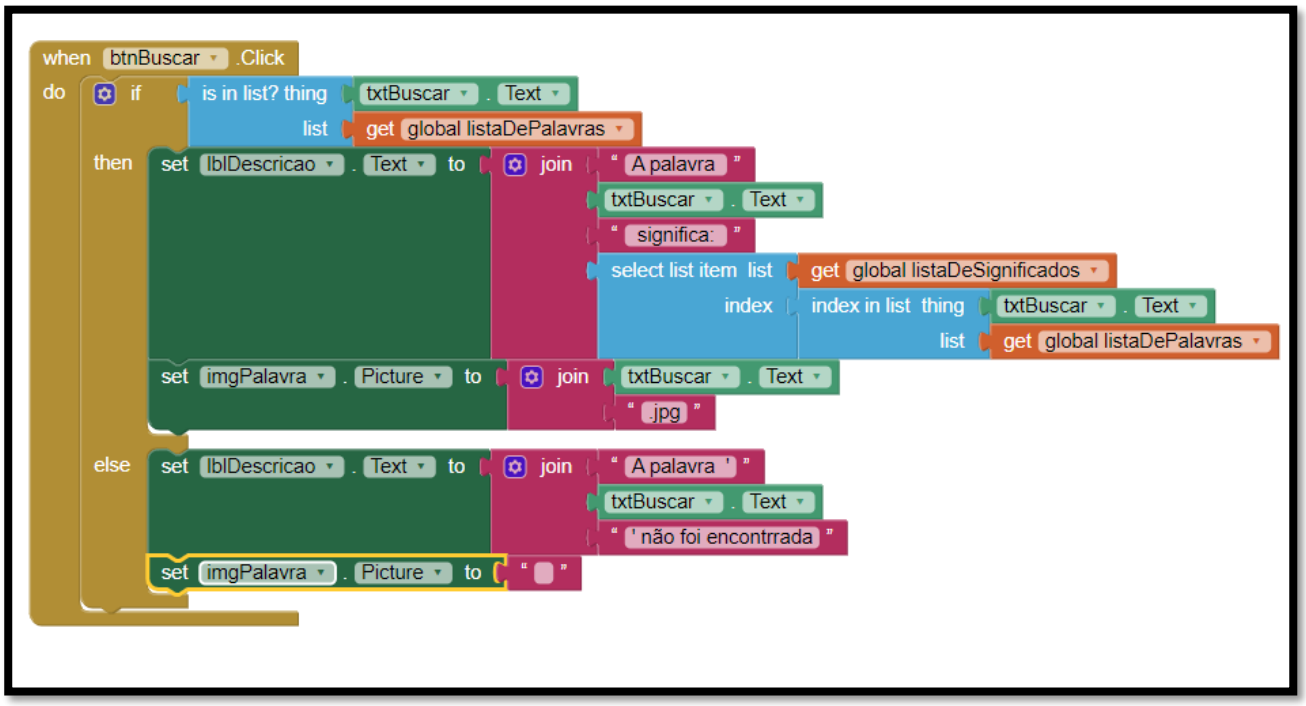


Figura 68 - Adicionando imagem a busca

7º Passo: Agora vamos fazer o aplicativo falar, então, clique no componente "*TextToSpeech*" e busque por "*set TextToSpeech1.Language to*" e adicione um componente de texto escrito "Portuguese" ou "pt", após isso basta chamar o método "*call TextToSpeech. Speak Message*" e adicionar os itens a serem falados.

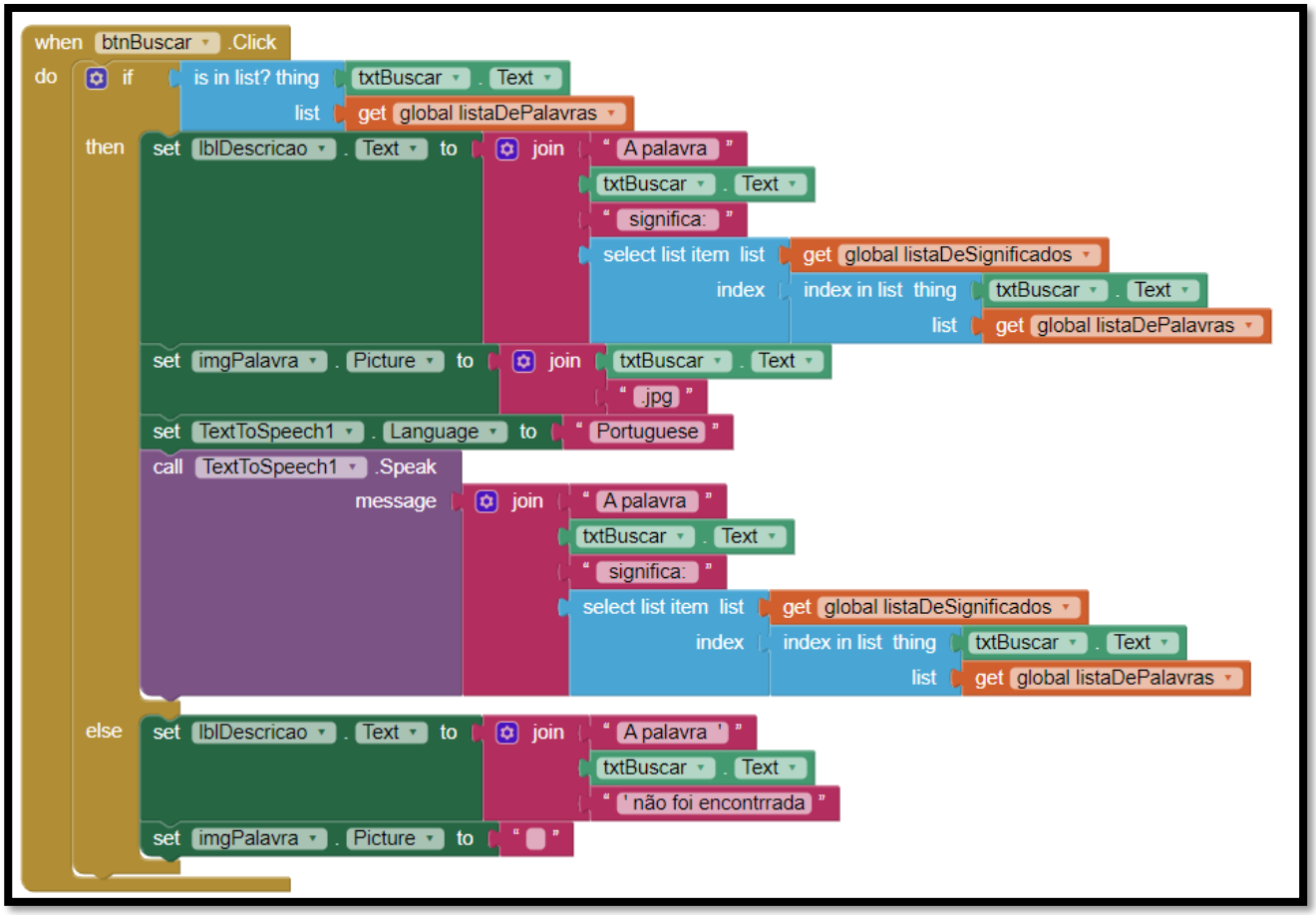


Figura 69 - Método para fala do texto

8º Passo: Para a finalização do projeto, puxe o método "*when btnOuvir.Click do*" e coloque a fonte do som e use o método para tocar.

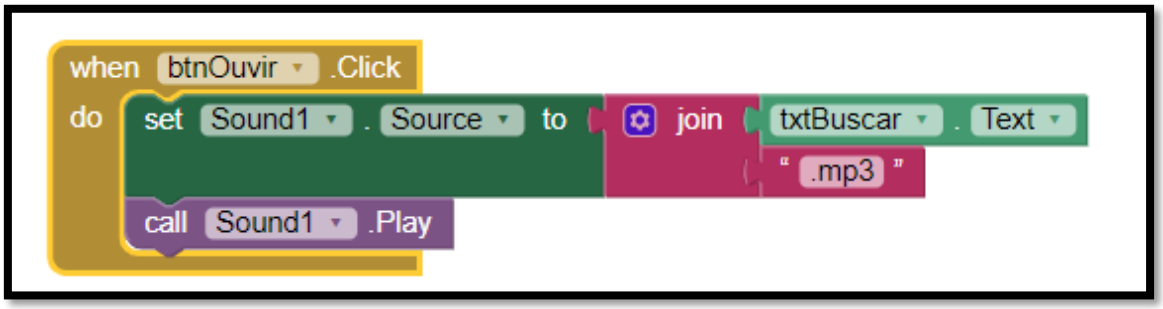


Figura 70 - Botão ouvir som

9.2 Conclusão

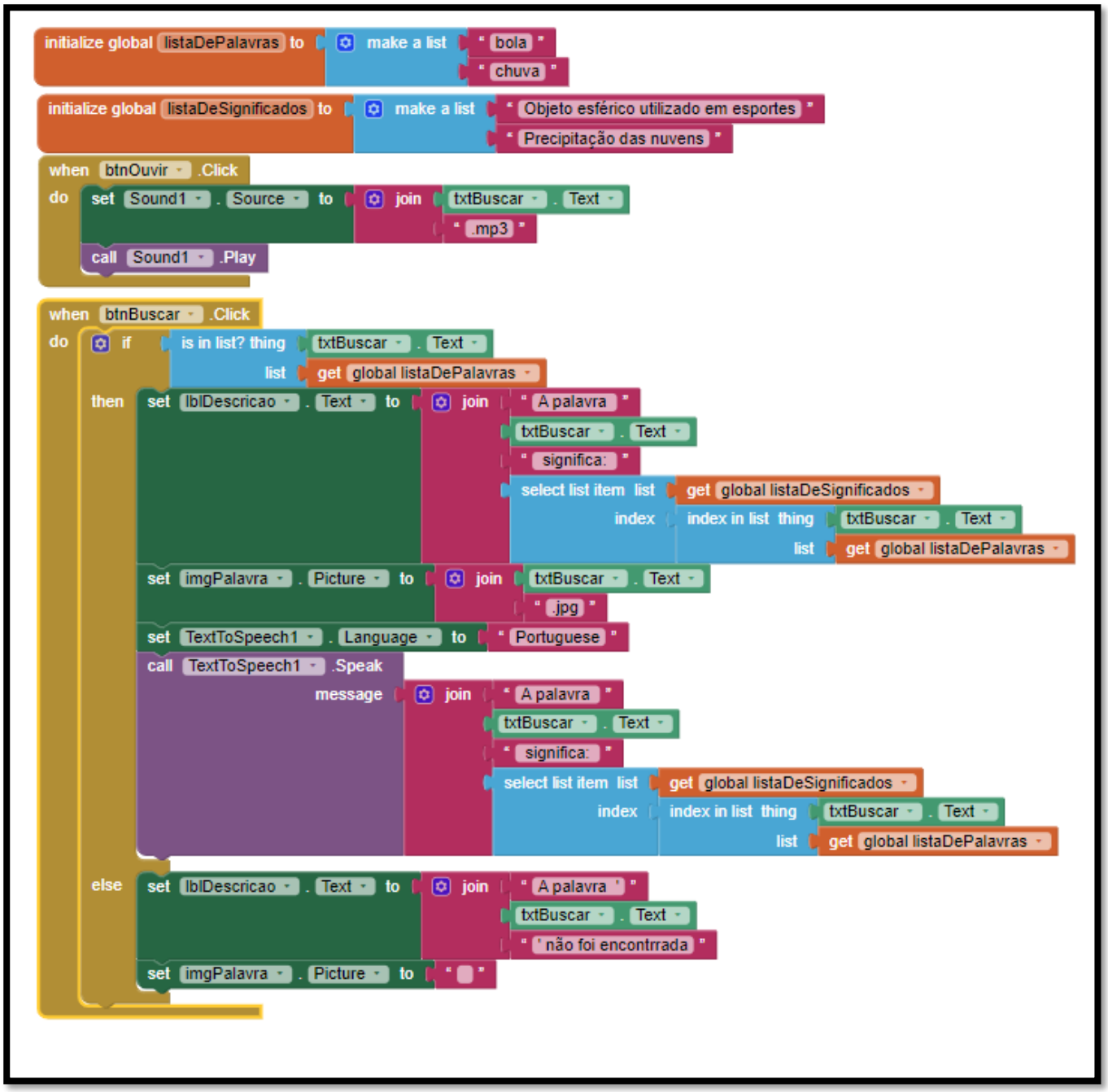


Figura 71 - Código completo

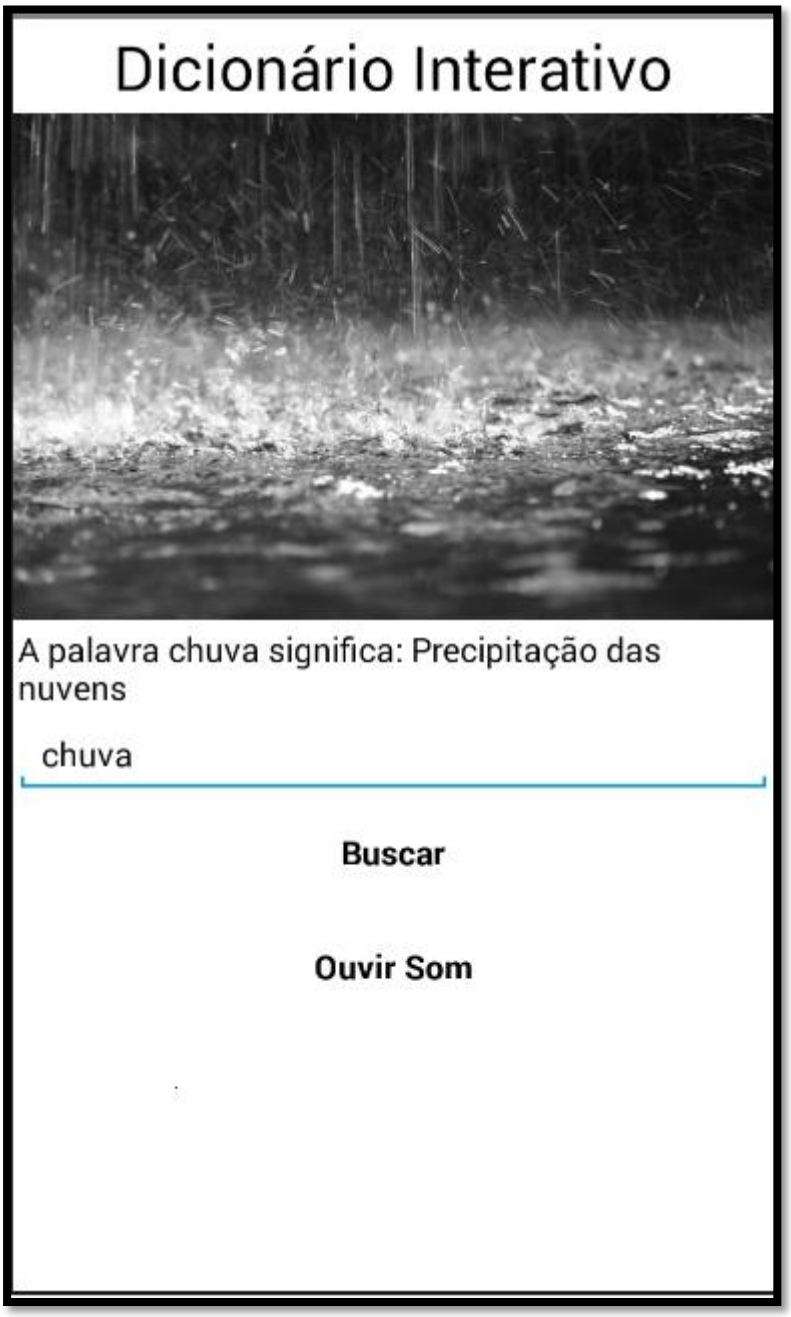


Figura 72 - Aplicativo

10 Projeto – Editor de Imagens

Neste projeto iremos criar um aplicativo que permite tirar uma foto, ou abrir da galeria e editar com o objeto *Canvas* e também irá salvar a foto.

10.1 Design do Aplicativo e Algoritmo

1º Passo: Neste aplicativo iremos utilizar os seguintes componentes:

- 6 Botões
- 1 *ImagePicker*
- 1 Campo de texto
- 1 *Label*
- 1 *Canvas*
- 1 *Camera* (não visível)

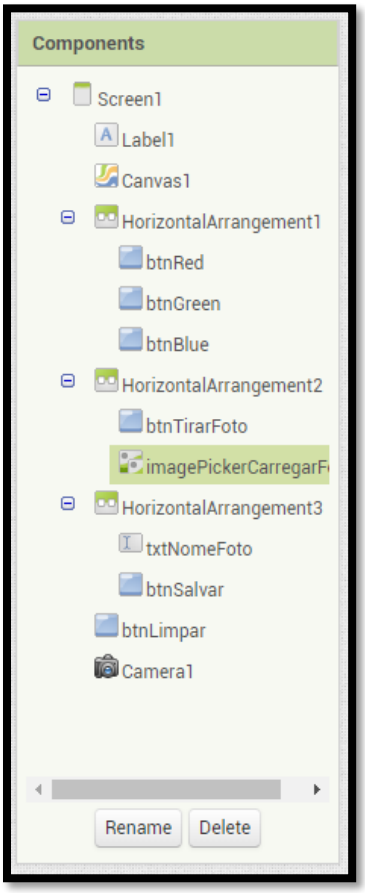


Figura 73 - Árvore de Componentes

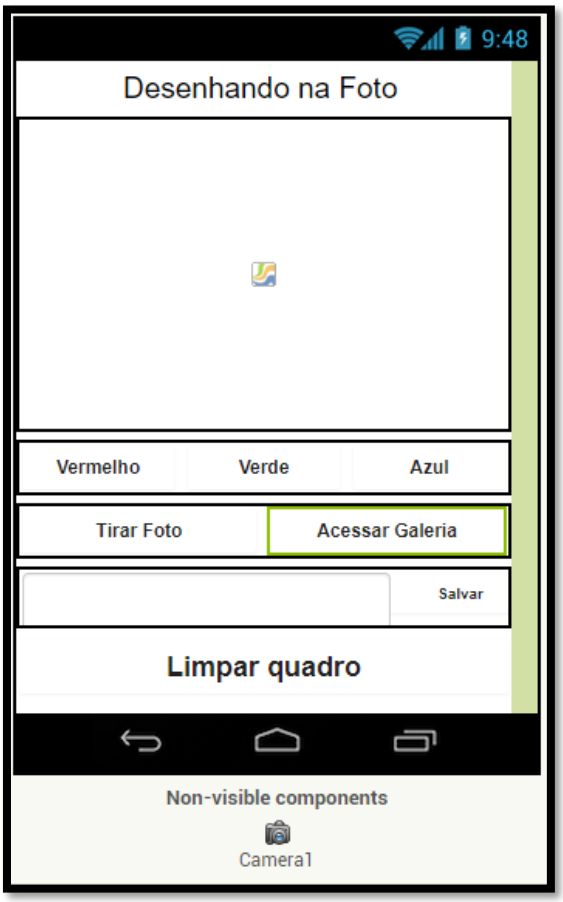


Figura 74 - *Layout* do aplicativo

2º Passo: Uma vez montada a interface, vamos aos blocos, primeiramente iremos habilitar a opção de desenhar círculos no canvas,para isso clique no componente Canvas e selecione a opção abaixo:

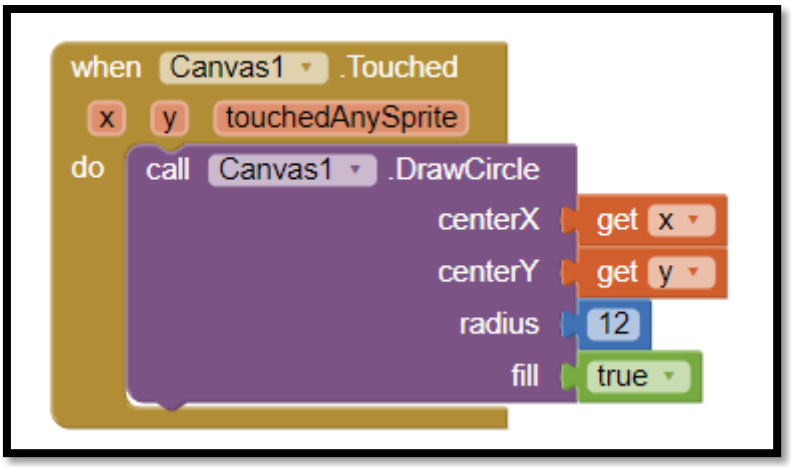


Figura 75 - Desenhando Círculos

3º Passo: Precisamos agora, habilitar a opção "*drag*" no canvas, para isso, faça como o bloco abaixo descrito.

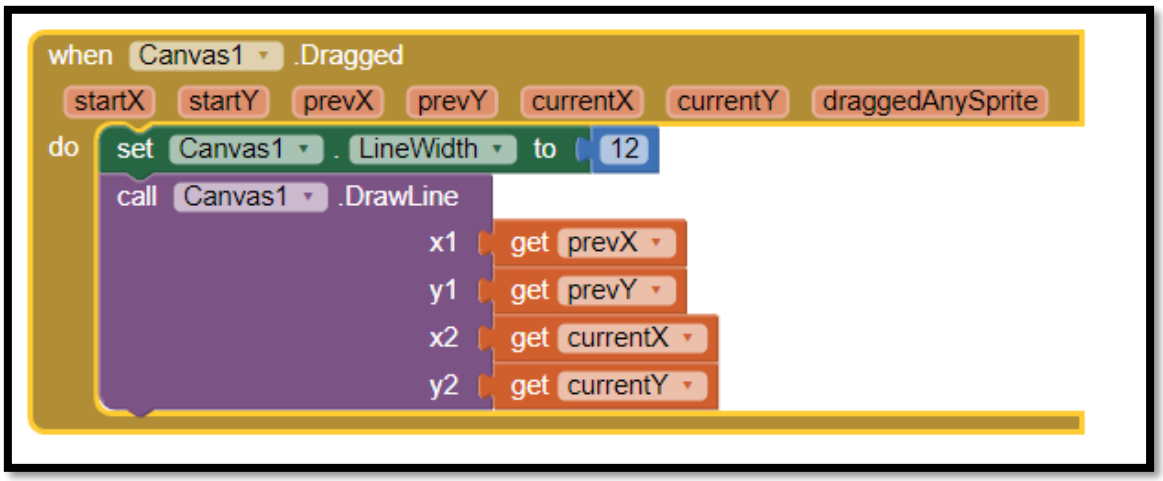


Figura 76 - Função *drag*

4º Passo: Para criar o botão de limpar, basta simplesmente utilizar o método "*call Canvas1.Clear*"

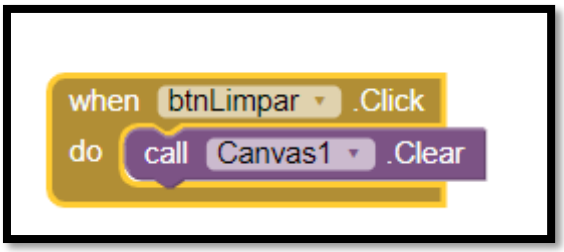


Figura 77 - Botão Limpar

5º Passo: Habilitando a mudança de cor do Canvas, para isso, arraste os itens conforme abaixo:

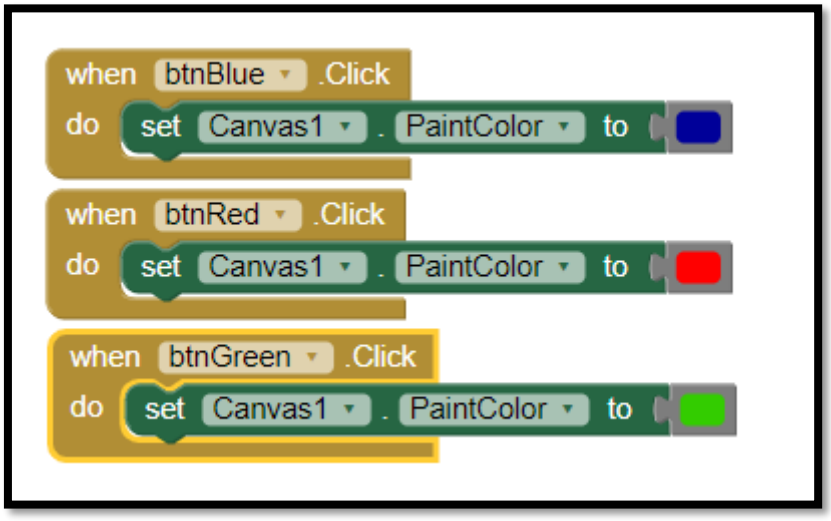


Figura 78 - Mudando cor

6º Passo: Para carregar a imagem da sua galeria para o canvas faça conforme a imagem abaixo:

Figura 79 - Carregar da galeria

7º Passo: Para carregar a imagem direto da câmera para o canvas faça conforme a imagem abaixo

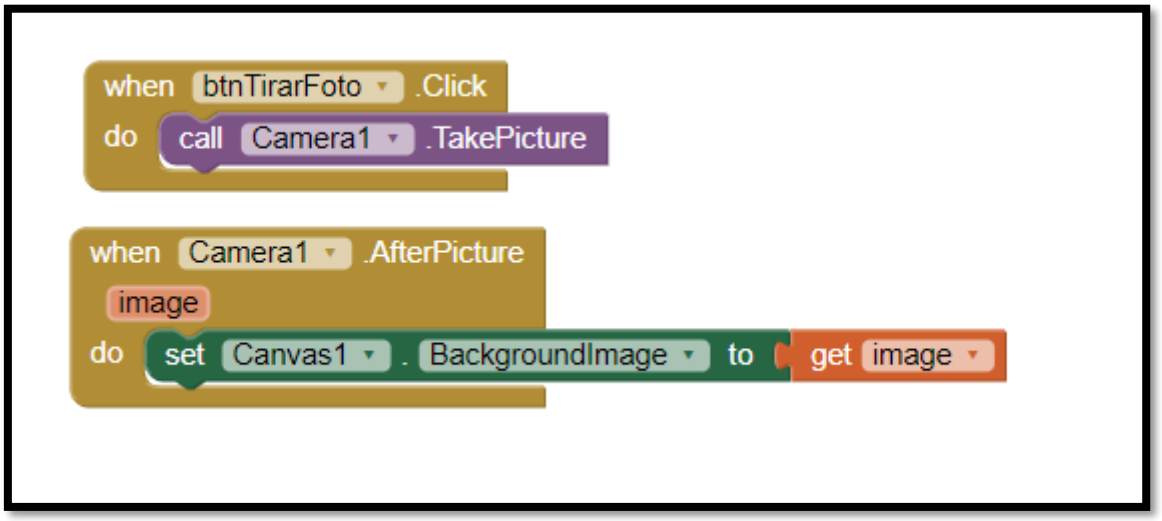


Figura 80 - Foto tirada da câmera

8º Passo: Finalizamos o projeto com o botão salvar, para salvar não esqueça de por a extensão.

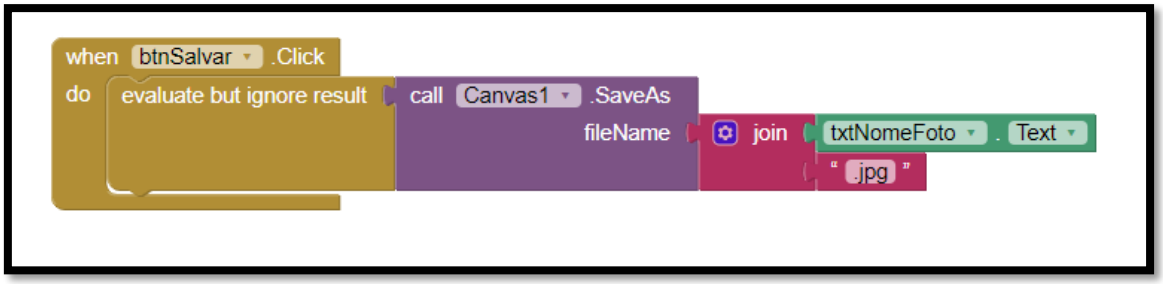


Figura 81 - Botão Salvar

10.2 Conclusão

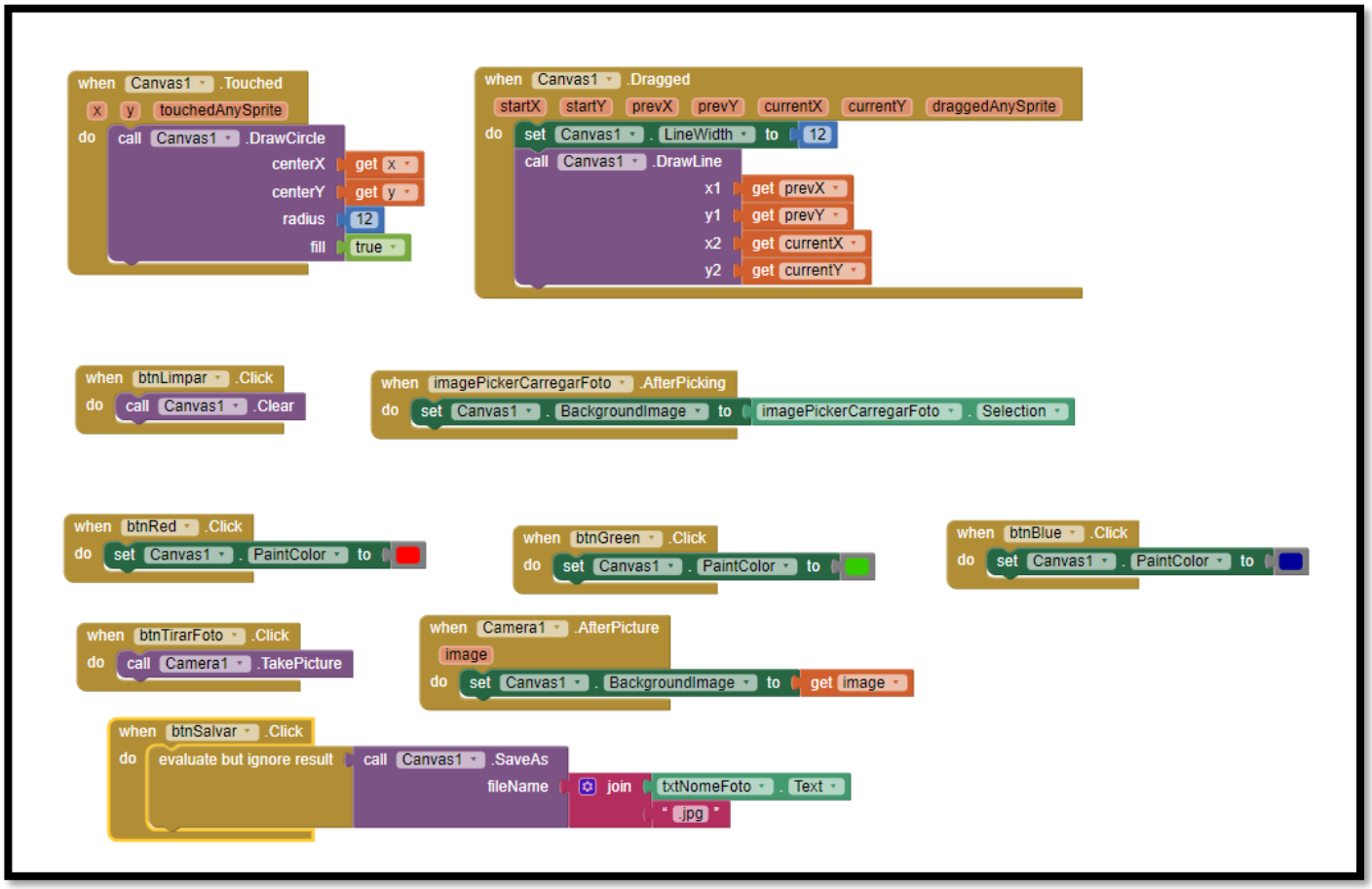


Figura 82 - Código Completo

Figura 83 - App Final 1

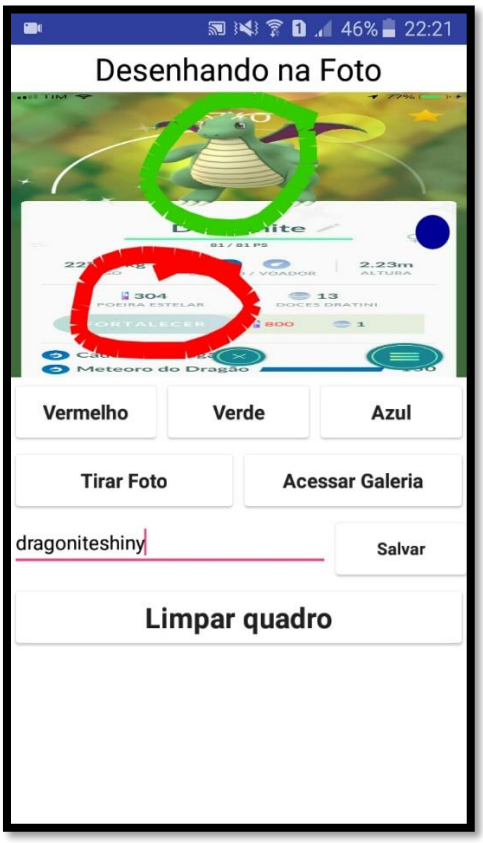


Figura 84- App Final 2

11 Exercícios – Intermediário

1- Desenvolva um aplicativo que calcule a média de um aluno, caso a média seja maior que 6, deverá aparecer uma mensagem “Aluno Aprovado” e se for menor “Aluno reprovado”. Porém nesse exercício você deverá aplicar o conceito de listas (*arrays*) então o professor deverá informar quantas notas ele irá aplicar e após isso, deverá ser incluído o aluno no banco de dados, com sua média e respectiva situação.

2- Acesse o site da Pura Vida Apps, e instale a extensão de *battery* - https://puravidaapps.com/battery.php - e crie um aplicativo que envia uma notificação ao usuário -para isso baixe extensão de notificação - https://puravidaapps.com/notification.php - quando a bateria estiver abaixo de 10% e quando estiver totalmente carregada.

3- Desenvolva uma aplicação que o usuário entre com a fórmula e a expressão matemática seja resolvida, para isso utilize a extensão https://puravidaapps.com/math.php

4- Crie uma aplicação que o usuário deverá entrar com as horas das do seu dia, como por exemplo: Estudos – 1h, Netflix – 3h, Trabalho – 8h e no final gerar um gráfico do dia. Para isso acesse o link https://github.com/MillsCS215AppInventorProj/chartmaker e baixe a extensão.

5- Faça um aplicativo capaz de fazer a fórmula de Bhaskara e *plotar* a função de segundo grau. Utilize as extensões que achar necessária.

6- A Equação de Drake é um argumento probabilístico usado para estimar o número de civilizações extraterrestres ativas em nossa galáxia Via Láctea com as quais poderíamos ter chances de estabelecer comunicação. Foi formulada por Frank Drake em 1961, não com o propósito de fornecer uma estimativa do número de civilizações, mas sim como um modo de estimular um diálogo científico.

$$N = R^* \times f_p \times n_e \times f_l \times f_i \times f_c \times L$$

Figura 85 - Equação de Drake

Onde:

N é o número de civilizações extraterrestres em nossa galáxia com as quais poderíamos ter chances de estabelecer comunicação.

R* é a taxa de formação de estrelas em nossa galáxia.

fp é a fração de tais estrelas que possuem planetas em órbita.

ne é o número médio de planetas que potencialmente permitem o desenvolvimento de vida por estrela que tem planetas.

fl é a fração dos planetas com potencial para vida que realmente desenvolvem vida.

fi é a fração dos planetas que desenvolvem vida inteligente.

fc é a fração dos planetas que desenvolvem vida inteligente e que têm o desejo e os meios necessários para estabelecer comunicação.

L é o tempo esperado de vida de tal civilização.

Com estas informações, desenvolva um aplicativo que gere o número de civilizações em nossa galáxia.

7- Probabilidade é o estudo das chances de obtenção de cada resultado de um experimento aleatório. A essas chances são atribuídos os números reais do intervalo entre 0 e 1. Resultados mais próximos de 1 têm mais chances de ocorrer. Além disso, a probabilidade também pode ser apresentada na forma percentual.

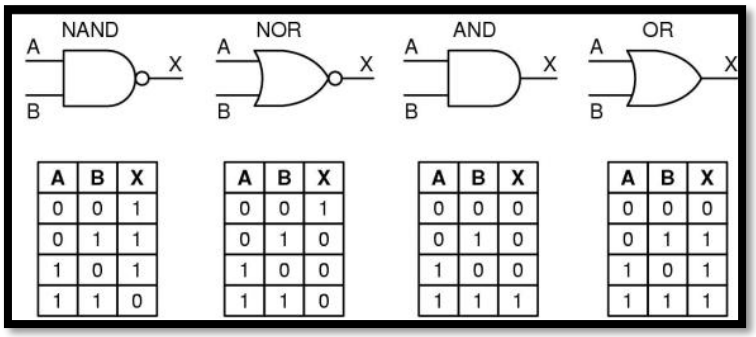
Desenvolva um aplicativo que calcule a probabilidade de cair um número X selecionado pelo usuário. Neste caso o usuário poderá entrar com a quantidade de dados que desejar.

$$P(E) = \frac{n(E)}{n(\Omega)}$$

Lembrando que se ele escolher as chances de cair o número X & Y deverá ser feita uma multiplicação caso seja X | Y deverá ser feita uma soma.

8- Portas lógicas são a essência da eletrônica moderna, fundamental em sistemas digitais e analógicos, somente com estes componentes os computadores são capazes de somar, multiplicar, dividir, subtrair e fazer as mais diversas operações, que são essências no nosso dia a dia.

Abaixo segue um exemplo de quatro portas lógicas, aonde 0 significa desligado e 1 significa ligado



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Figura 86 - Portas Lógicas

Utilizando a extensão https://community.appybuilder.com/t/led-view-extension/1869 desenvolva um aplicativo que recebe as entradas das portas lógicas (A e B) e acenda o LED quando a saída for 1 e apague quando for 0.

9- Crie um cronômetro, armazene os tempos salvos no banco de dados e plote gráfico em colunas dos tempos. Utilize as extensões que achar necessárias.

10- Crie um diário de saúde, aonde pessoa poderá anotar o peso dela todos os dias, plotar o gráfico e ter o IMC calculado. Utilize as extensões que achar necessárias.

12 Projeto – Criando um aplicativo que pisca LEDs com *BLUETOOTH LE 4.0(HM10)*

Diferentemente do já conhecido *bluetooth* HC-05 ou HC-06 o módulo HM10 trabalha com a versão 4.0 da tecnologia sem fio, e isso implica que é utilizado um modo de baixa energia (*Low Energy*) e trabalha na tensão de 3,3V.

Na prática a diferença vai ser que o responsável por fazer o pareamento da conexão é o aplicativo e não mais o hardware. O App Inventor não tem biblioteca nativa para BLE então é necessário baixar uma extensão.

Neste módulo você irá aprender a utilizar:

- *Bluetooth LE*
- *ListView*
- Enviar dados para o arduino

12.1 Aplicativo e Blocos

1º Passo: Acesse http://iot.appinventor.mit.edu/assets/resources/edu.mit.appinventor.ble-20181124.aix para fazer o *download* da extensão, caso queira saber mais o módulo acesse a documentação, disponível em http://iot.appinventor.mit.edu/assets/resources/edu.mit.appinventor.ble-20181124.aix

Author	Version	Download .aix File	Source Code
MIT App Inventor	20181124	BluetoothLE.aix	Via GitHub

Figura 87- Download da extensão

2º Passo: Para o desenvolvimento da interface do aplicativo vamos precisar de:

- 7 Botões
- 1 *ListView*
- 1 *Label*
- 1 *Bluetooth Cliente (Non-Visible)*
- 1 *Bluetooth Low Energy (Non-Visible)*

Figura 88 - *Layout* aplicativo

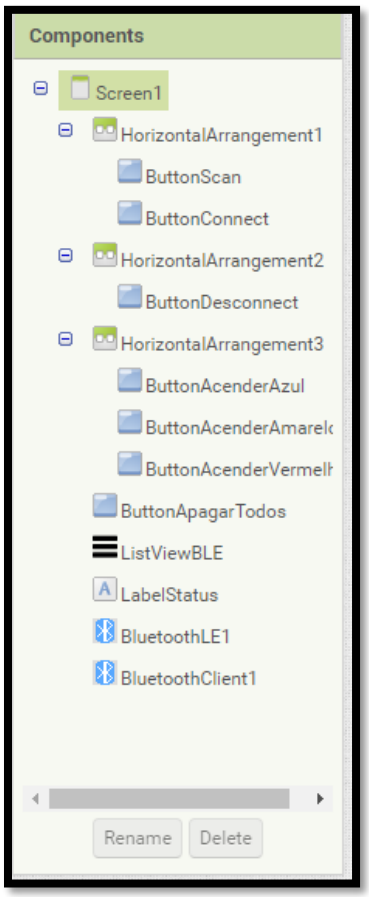


Figura 89 -Árvore de componentes

3º Passo: Agora vamos começar a codificação, primeiramente vamos criar o botão para escanear – lembre de usar a extensão BLE – conforme adicionada. Verificamos se o *bluetooth* esta ligado, caso esteja, habilitamos a *listview* e usamos o métodos para escanear.

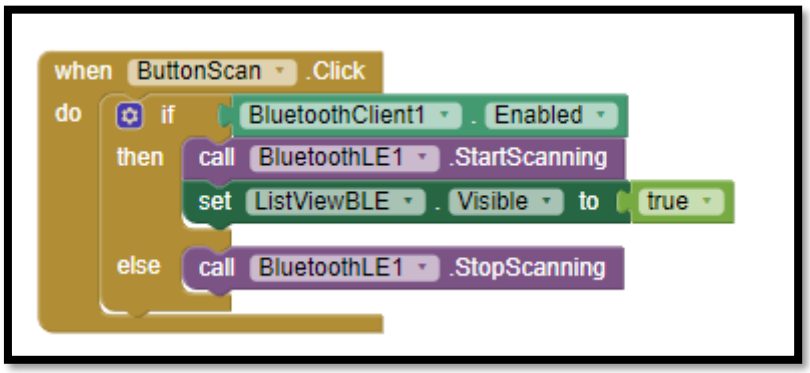


Figura 90 - Botão Scan

4º Passo: Quando o dispositivo for encontrado, ele irá habilitar a lista de equipamentos.

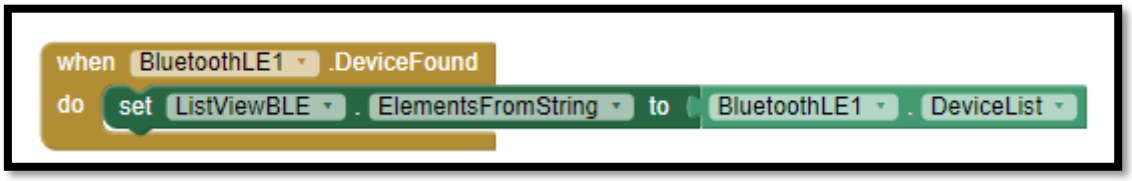


Figura 91 – ListView

5º Passo: É necessário criar duas variáveis globais que irão nos dar as permissões para o envio e o recebimento de dados. Essas definições são padrões para isso.

UUID: 0000FFE0-0000-1000-8000-00805F9B34FB

CHARACTERISTIC UUID: 0000FFE1-0000-1000-8000-00805F9B34FB

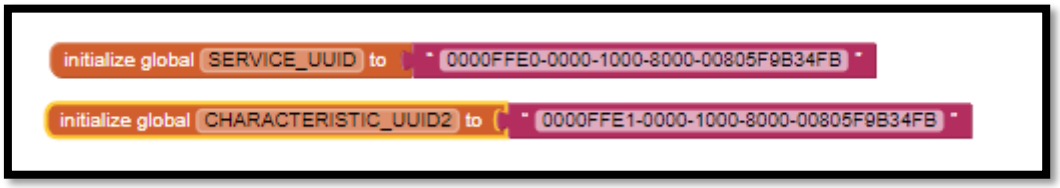


Figura 92 – Permisssões

6º Passo: Agora precisamos fazer que ele se conecte ao *bluetooth* até o momento ele apenas procura pelo dispositivo e lista o que há, para conectar precisamos verificar se o adaptador *bluetooth*

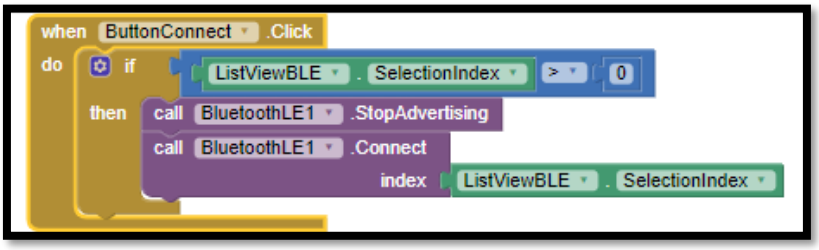


Figura 93 - Botão conectar

7º Passo: Uma vez que o dispositivo foi encontrado, habilitamos a visualização na lista.

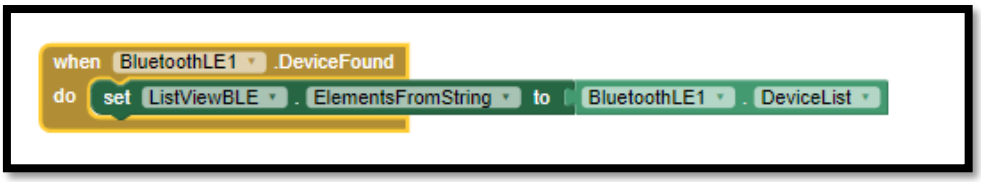


Figura 94 - Lista de dispositivos

8º Passo: Para encerrar a conexão vasta chamar o método "*call BluetoothLE1.Disconnect*"

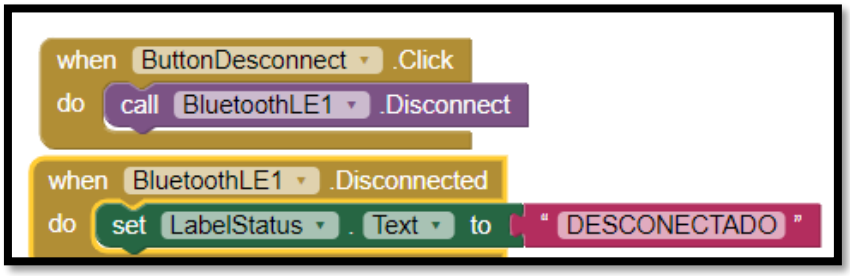


Figura 95 - Método Desconectar

9º Passo: Agora vamos criar o método que envia o comando de acender LED para o Arduino. Utilize o chamado "*WriteStrings*" conforme ilustrado abaixo.

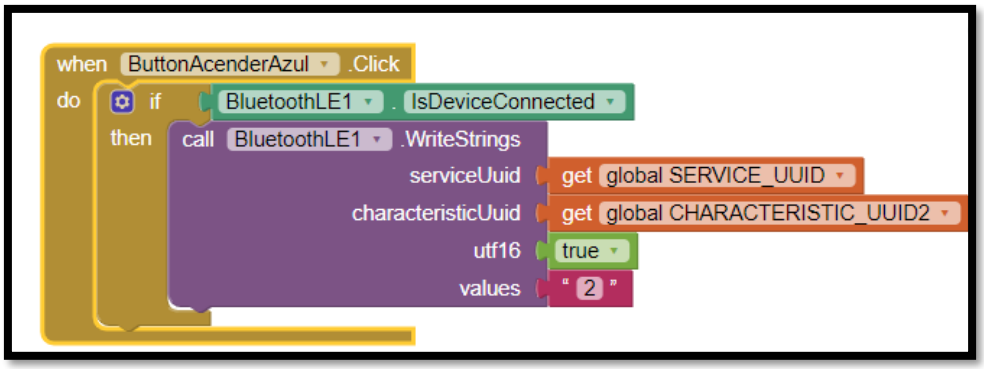


Figura 96- Acender Led Azul

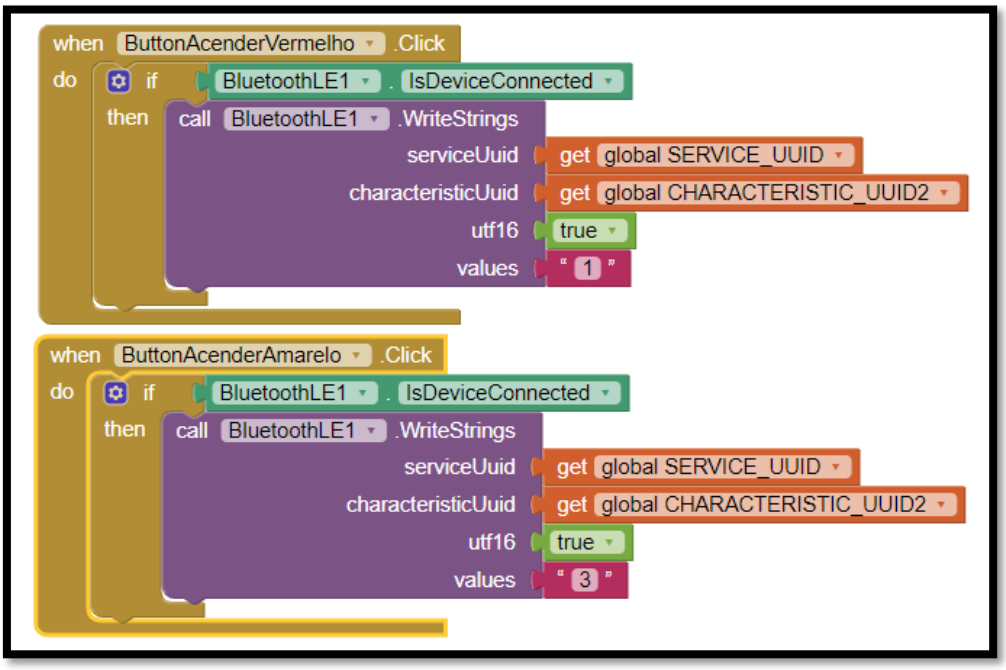


Figura 97 - Métodos para acender Vermelho e Amarelo

10º Passo: Caso queira mostrar o endereço hexa do seu dispositivo adicione o método "*FoundDeviceAdress*" e o coloque na label status.

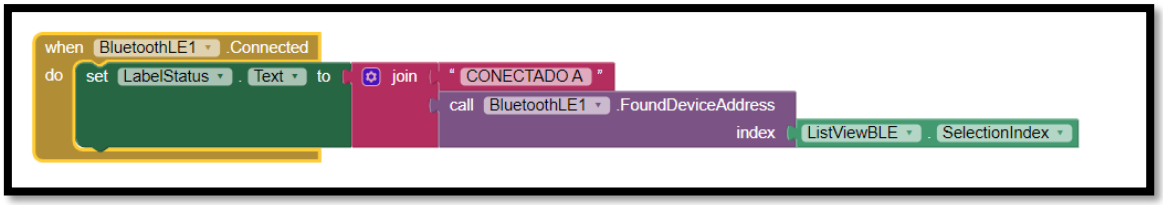


Figura 98 - Device Address

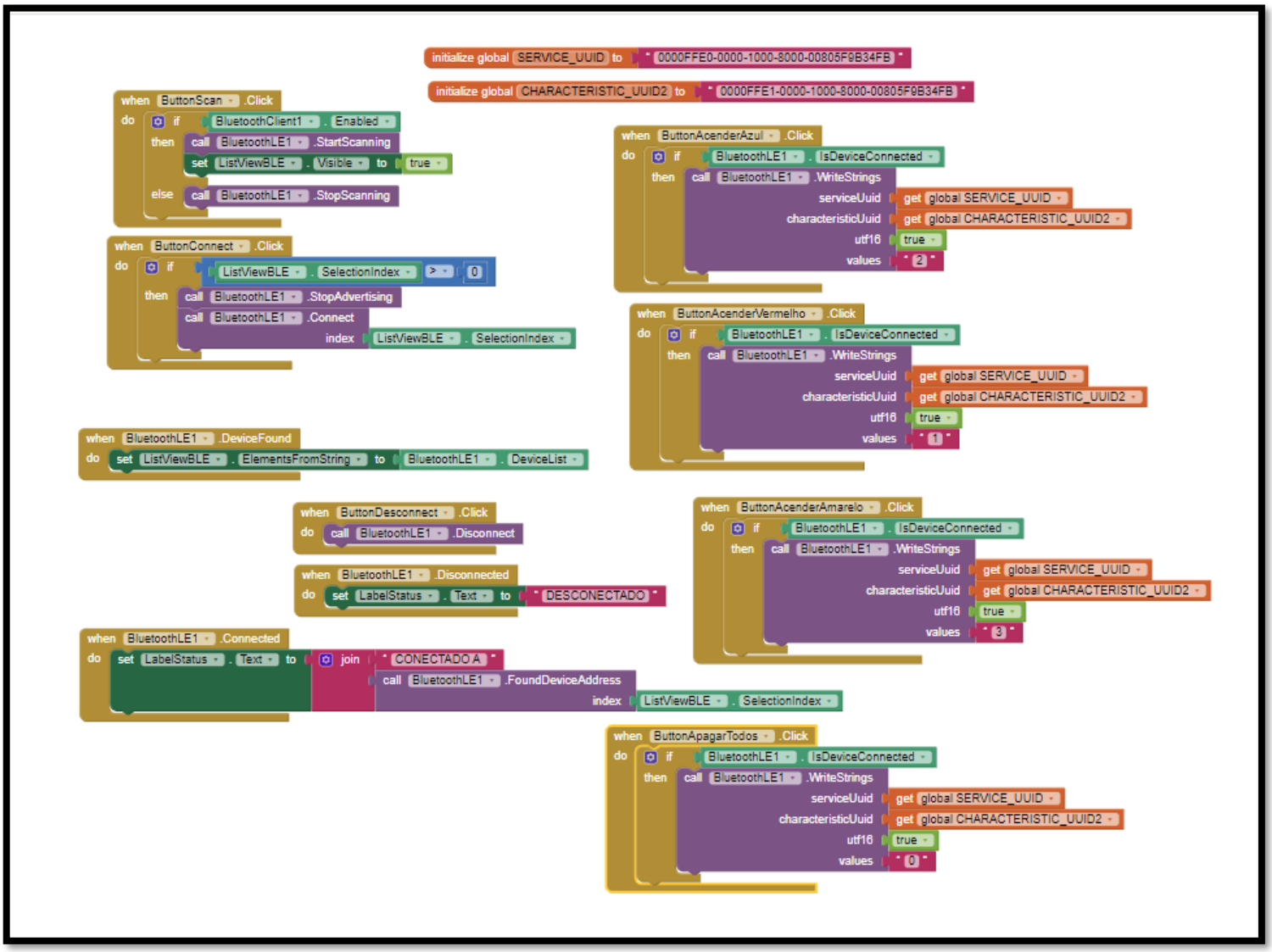


Figura 99 - Código Completo

12.2 Parte do Arduino

Nesta parte do projeto, se você não se sentir seguro com a eletrônica e a programação na linguagem C do Arduino, basta repetir os passos exatamente conforme descrito.

1º Passo: Monte o Arduino conforme a imagem abaixo.

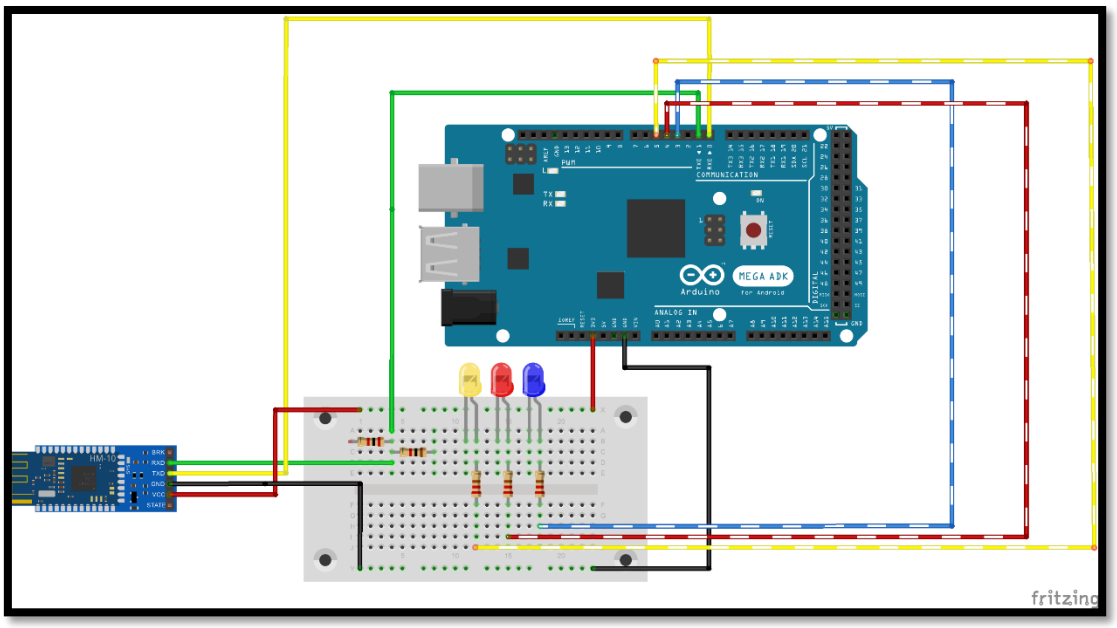


Figura 100 - Diagrama

Componente	Pino do Arduino
Led Azul	2
Led Vermelho	3
Led Amarelo	4
HM-10 (VCC)	VCC
HM-10 (RX)	Protoboard -> TX 0
HM-10 (TX)	RX0
HM10 (GND)	GND

Aonde HM-10 é o módulo *bluetooth*, também serão necessários 5 resistores, três de 220Ω, 1kΩ e um de 2kΩ.

2º Passo: Baixe um editor de código para Arduino - https://www.arduino.cc/en/Main/Software - e cole o seguinte código:

```
char data = 0;

void setup()
{
  Serial.begin(9600);

  pinMode(3, OUTPUT); //azul
  pinMode(4, OUTPUT); //vermelho
  pinMode(5, OUTPUT); //amarelo
}
void loop()
{


  if (Serial.available() > 0)
  {
    data = Serial.read();

    Serial.print(data);
    Serial.print("\n");
    if ( data == '1' || data == 'A')  {
      digitalWrite(3, HIGH);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);


    } else if ( data == '2' || data == 'V') {
      digitalWrite(3, LOW);
      digitalWrite(4, HIGH);
      digitalWrite(5, LOW);

    } else if ( data == '3' || data == 'M') {
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, HIGH);
    }

    else if ( data == '0' || data == 'B')     {
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      yep
    }
  }
}
```

13 Projeto – *Bluetooth Low Energy* recebendo dados HM10

1º Passo: Monte o Arduino conforme a imagem abaixo. Se você ainda tem o projeto anterior, basta remover os leds.

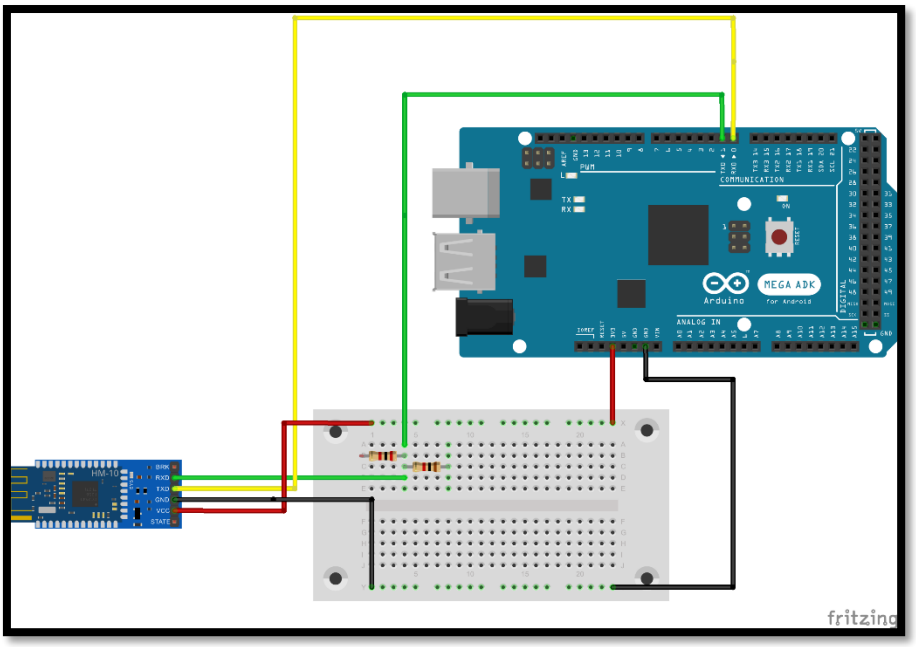


Figura 101 - Modelo BLE + Mega

2º Passo: Agora cole o código abaixo, lembrando que ele apenas envia uma sequência de caracteres “BBBBB” que irá ser visualizada em nosso aplicativo.

```
#include <SoftwareSerial.h>

#define txPin 1
#define rxPin 0
SoftwareSerial BLE(rxPin, txPin);// rx tx

void setup() {

  BLE.begin(9600);

}

void loop() {
  while (BLE.available()) {

    BLE.print("BBBBBBBBB");

    delay(1);
  }


    BLE.println("BBBBBBBBB");

    delay(500);

}
```

c aplicativo, para isso, deixe conforme a imagem abaixo

Figura 102 - Tela do Projeto

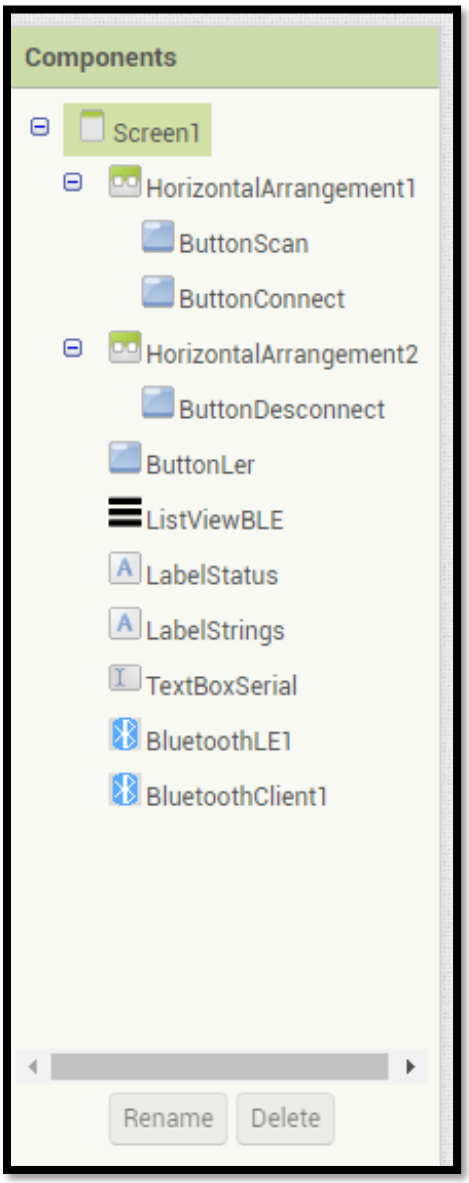


Figura 103 - Arvore de Componentes

4º Passo: Modificando do projeto anterior, remova todos os botões de acender LED, adicione um botão para começar a receber eventos do Arduino.

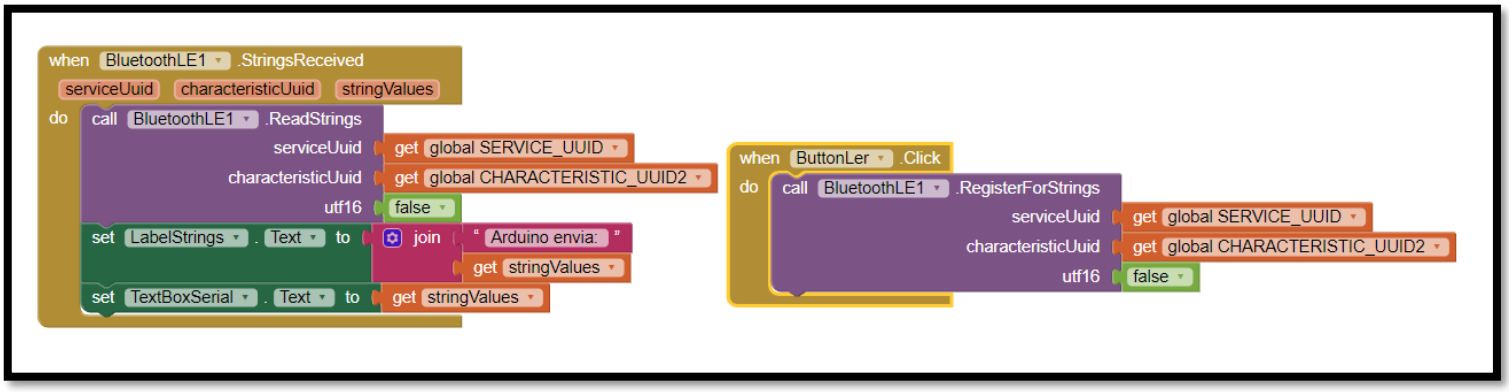


Figura 104 - Novos códigos

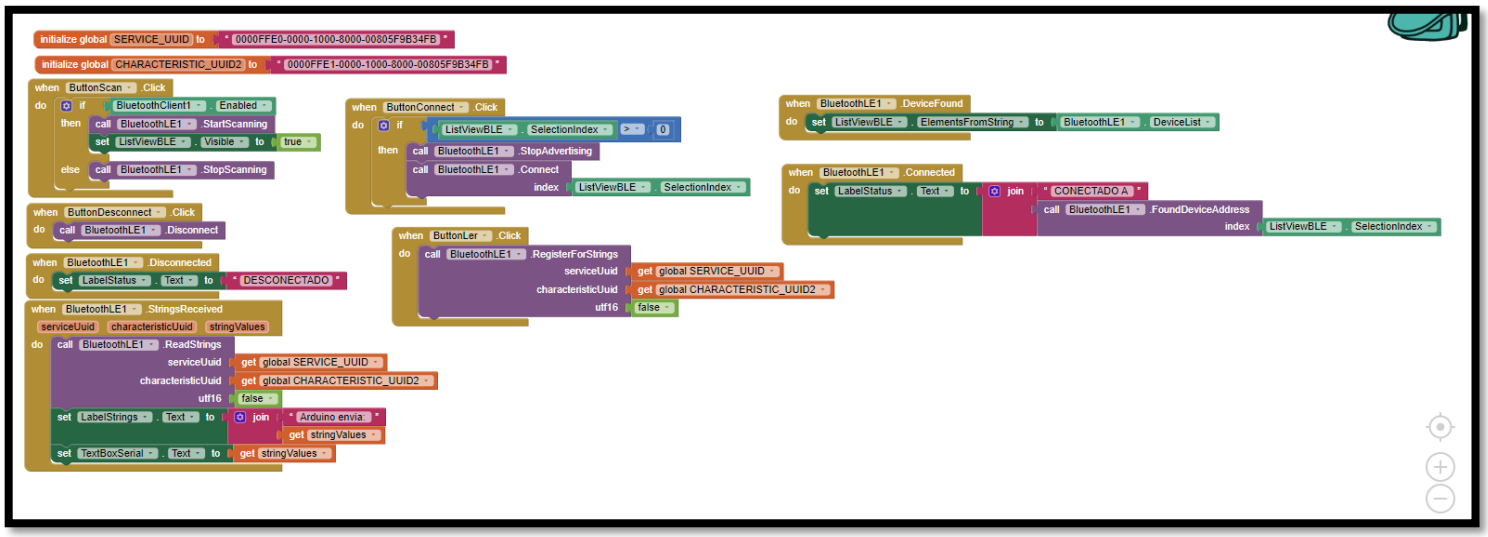


Figura 105 - Código Completo

Figura 106 - Tela do app

14 Exercícios – Avançado

1 – Utilize um sensor de proximidade ultrassônico, um módulo *bluetooth* e crie um aplicativo que receba a proximidade de algum objeto.

2- Faça um sistema de automatização residencial, aonde você poderá ligar diferentes objetos da sua casa, utilizando o módulo *bluetooth*, através do aplicativo, tais como:

- Lâmpada
- Trancas da Porta
- Ventilador
- Entre outros.

3- Faça um sistema que mostre no aplicativo a quantidade de batimentos cardíacos, e gere um gráfico desses batimentos. Utilize sensor de batimentos.

4- Faça um projeto de catraca que envie o código do RFID para o seu celular e você possui a opção de liberar ou não.

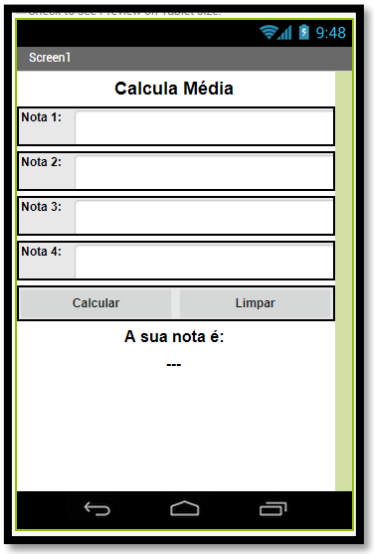
5- Faça um sistema que meça a temperatura e umidade do cômodo e envie via *Bluetooth*.

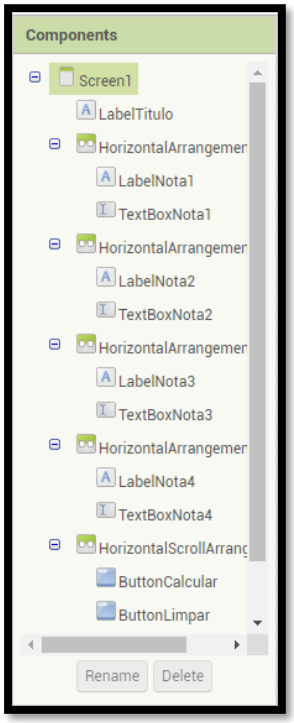
6- Desenvolva um sistema de irrigador automático, utilizando sensor de solo e controlado via celular.

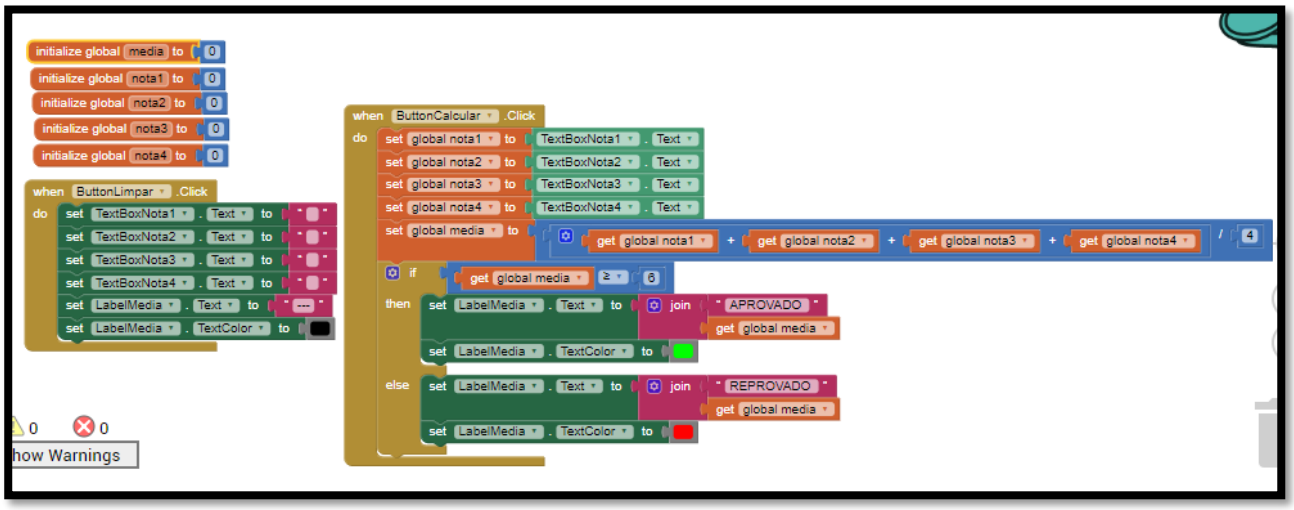
ANEXO – Repostas das Listas de Exercícios

Lista básica

Exercício 1- Calcula Média

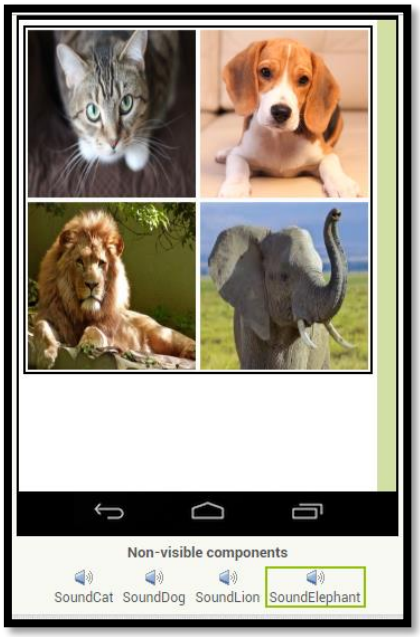


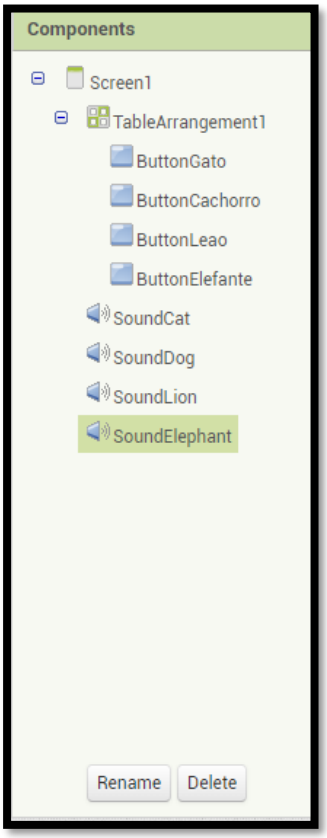


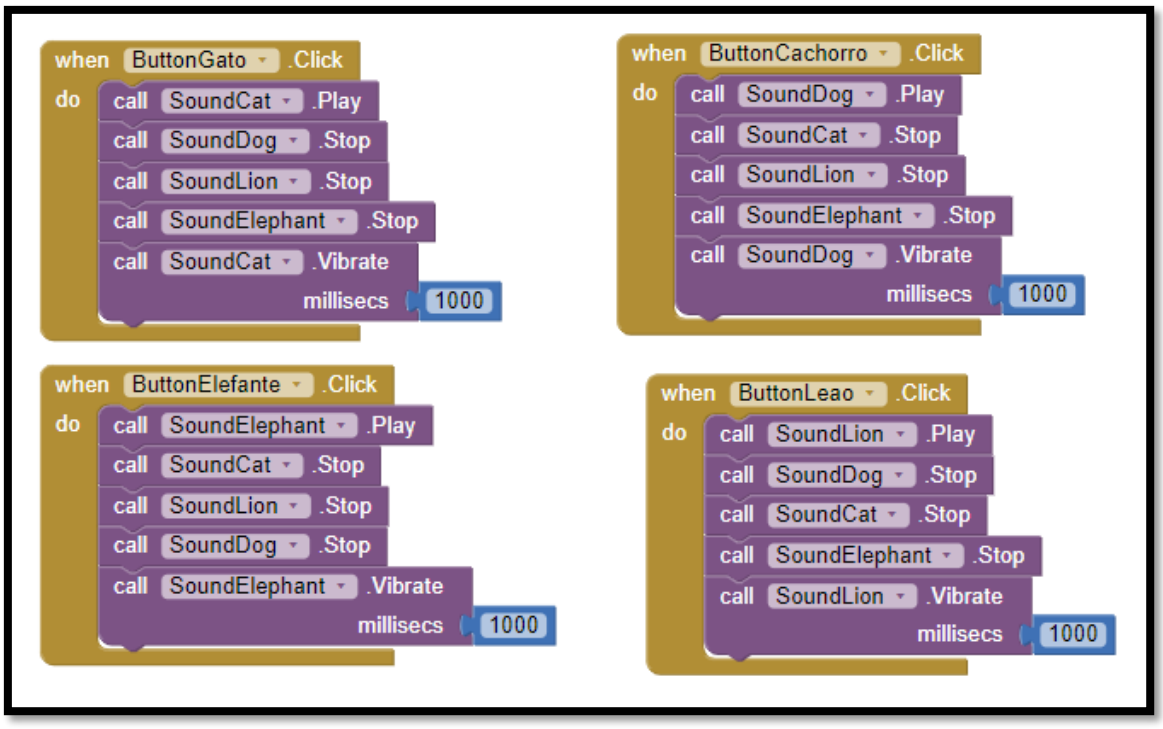


Vídeo: https://youtu.be/eSliG1Jv1pQ

Exercício 2 – Sons dos animais

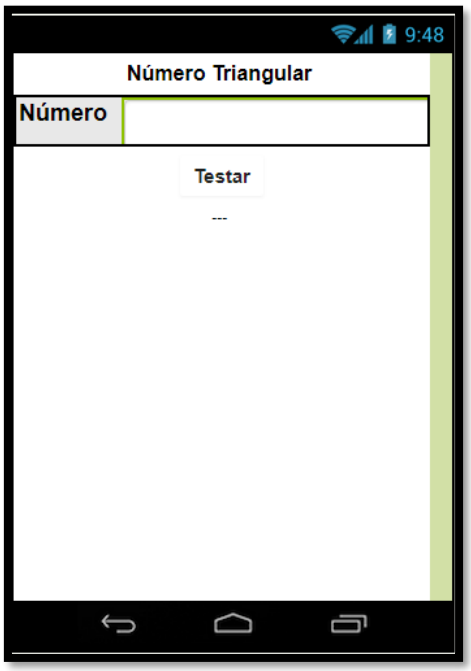


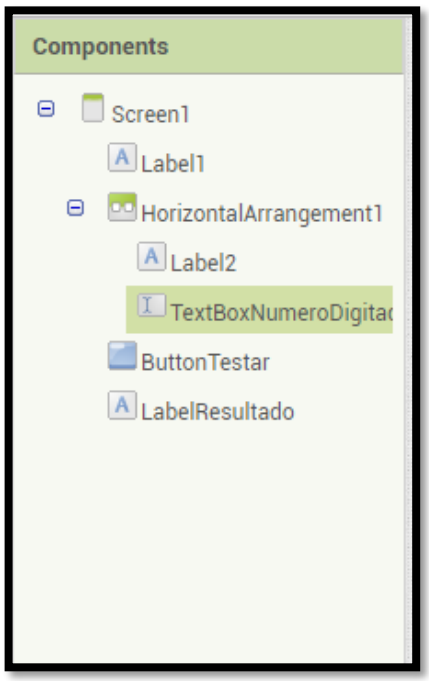


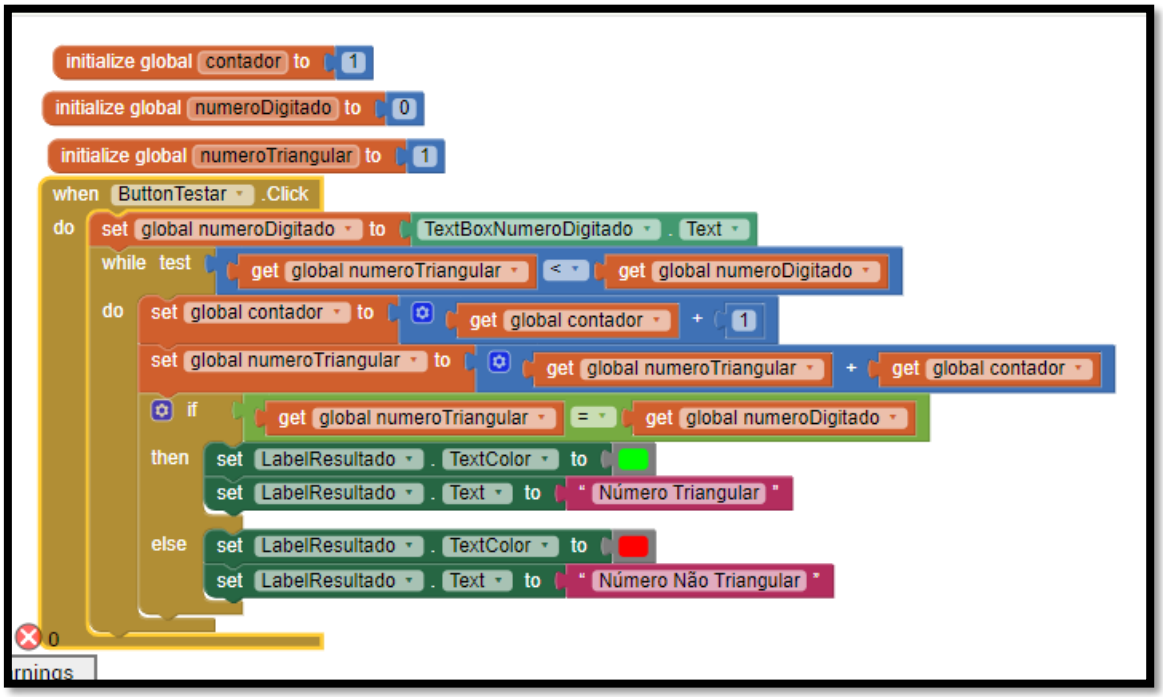


Vídeo: https://youtu.be/KoWMoMGDles

Exercício 3- Número Triangular

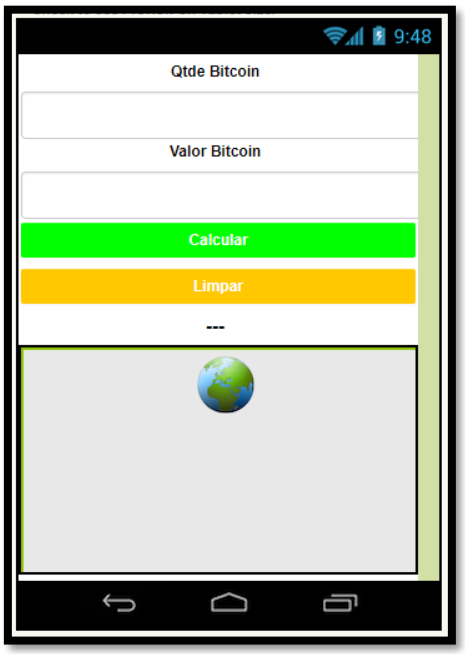


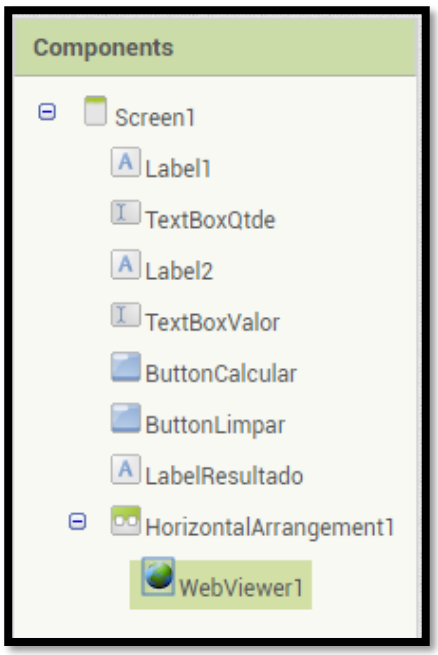


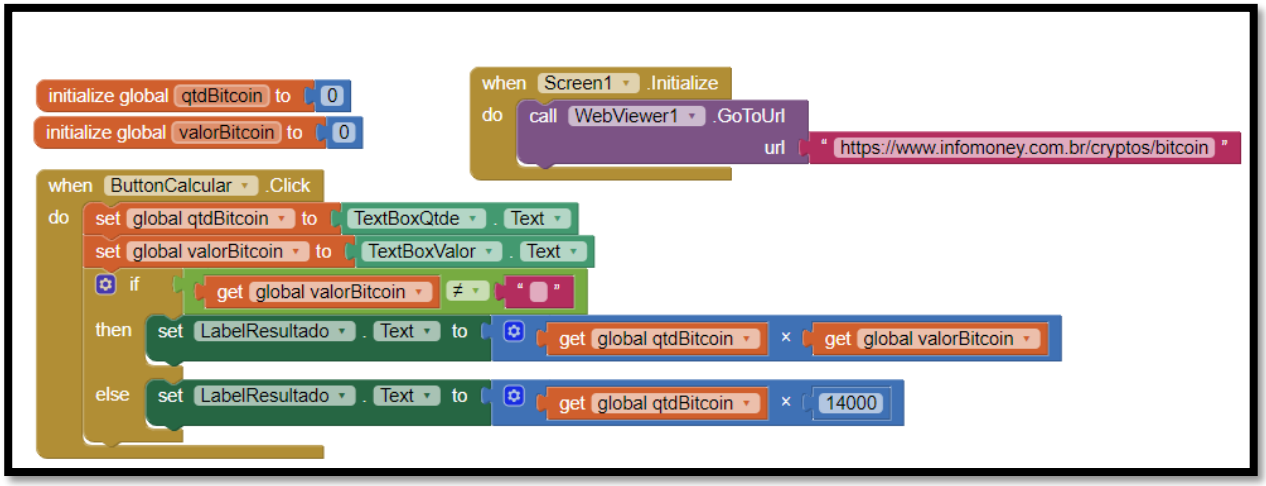


Vídeo: https://youtu.be/Wbj_y_6FDEM

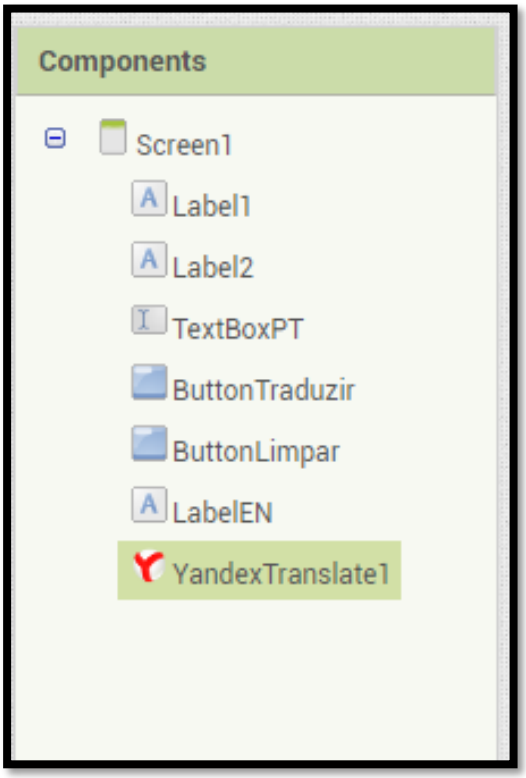
Exercício 4 – Bitcoin Conversor

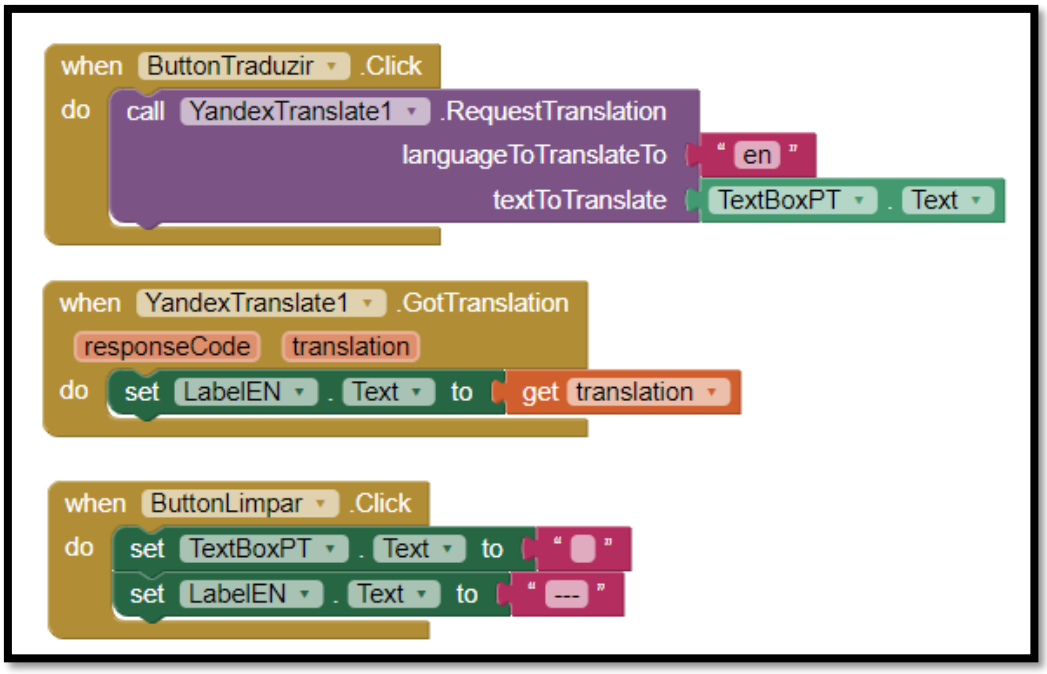






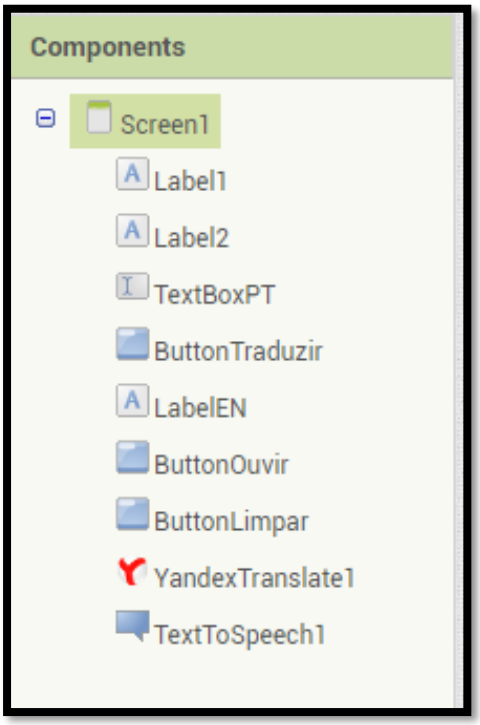
Vídeo: https://youtu.be/rtvO8_ftCqo

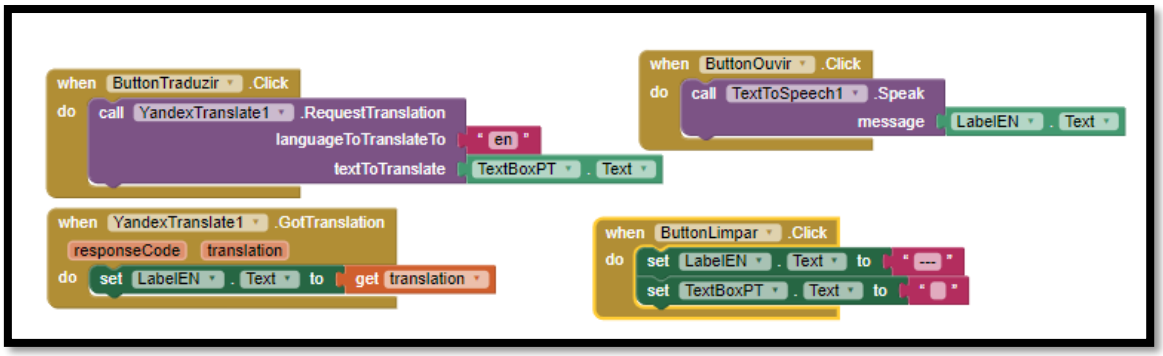




Vídeo: https://youtu.be/k5C36Skgi38

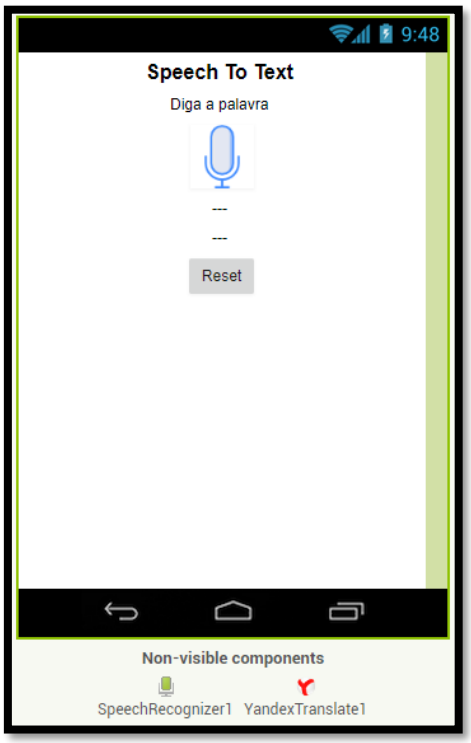
Exercício 6 – *Text to Speech*

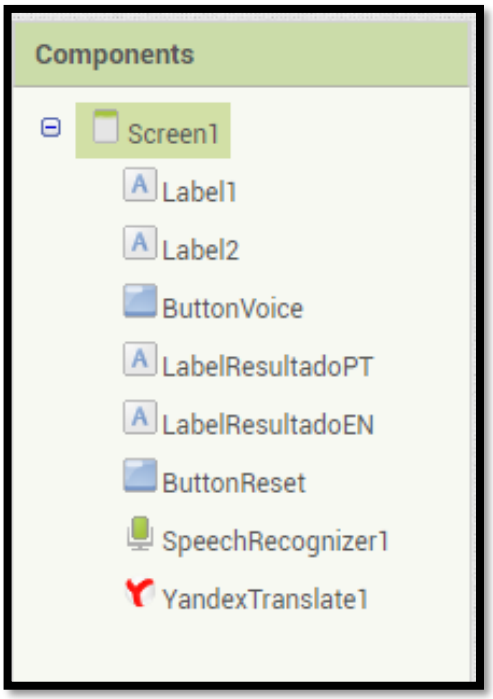


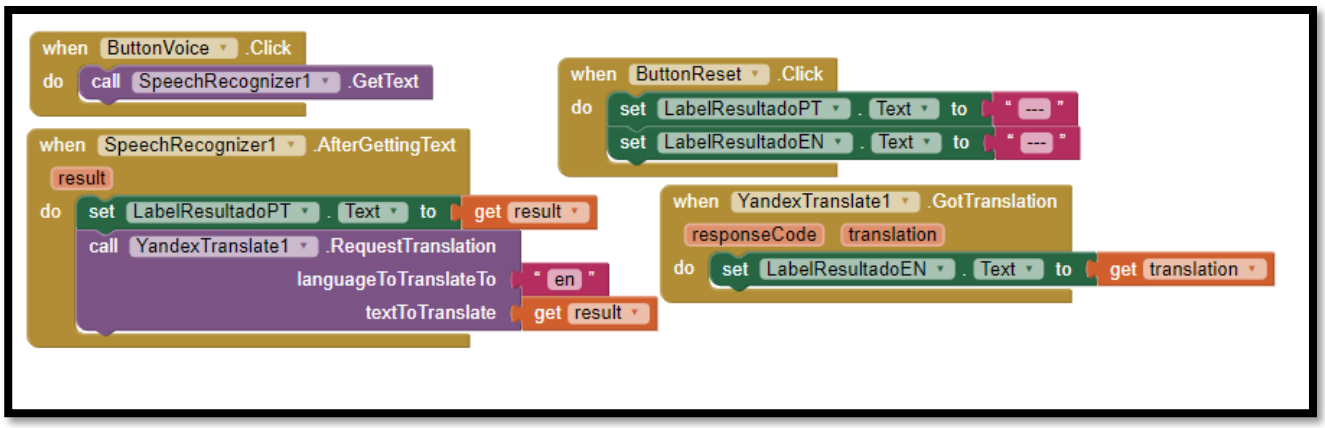


Vídeo: https://youtu.be/dX5zwJdLSdg

Exercício 7 – *Speech to Text*

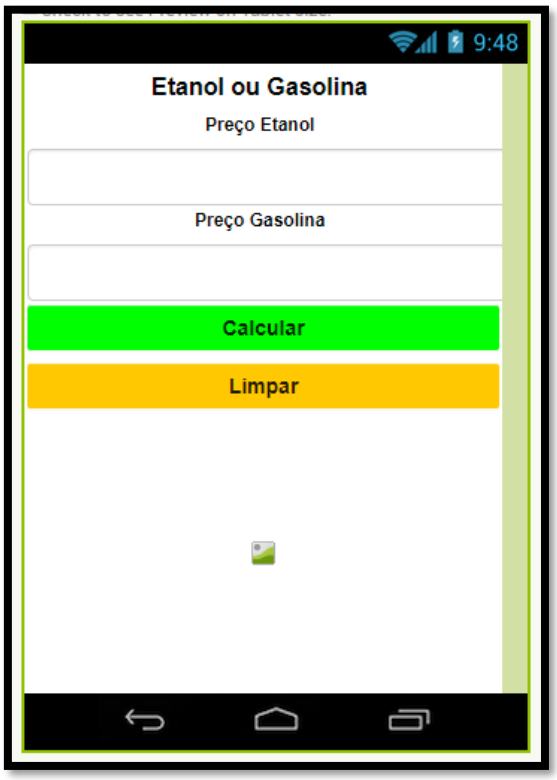


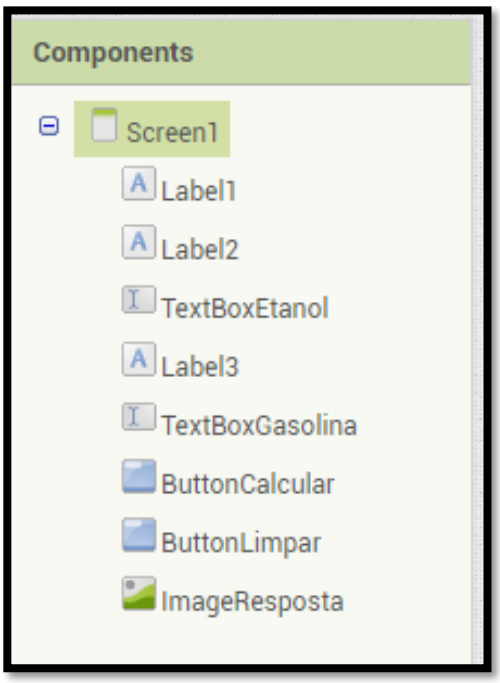


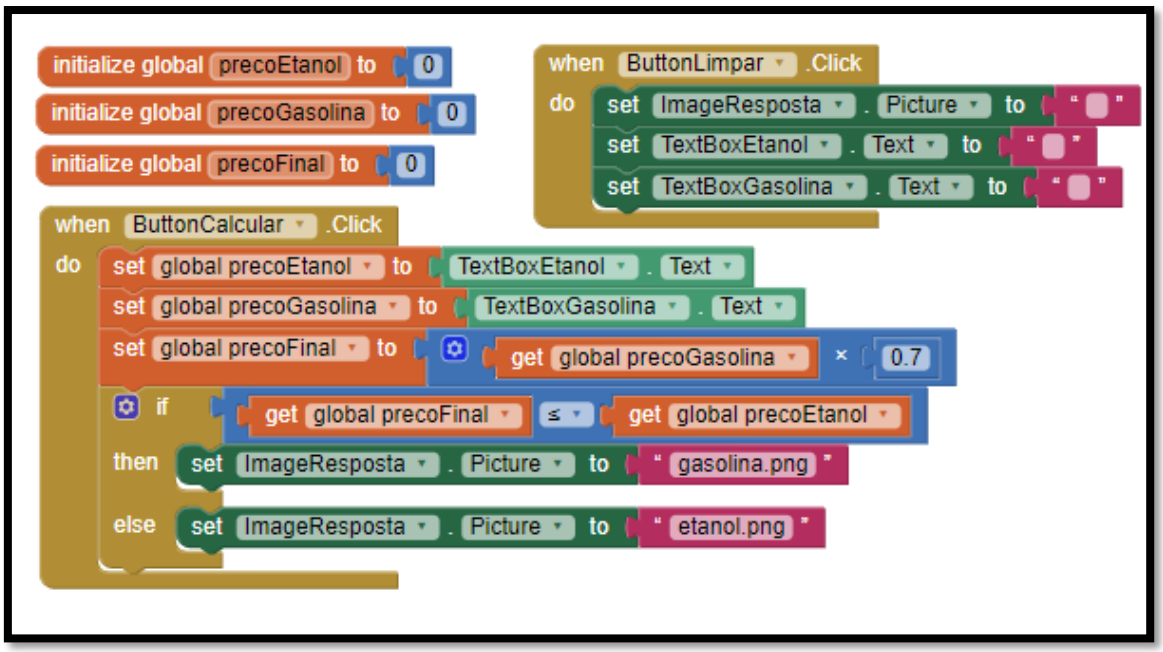


Vídeo: https://youtu.be/iPAMoGtwKw8

Exercício 8 – Etanol Ou Gasolina

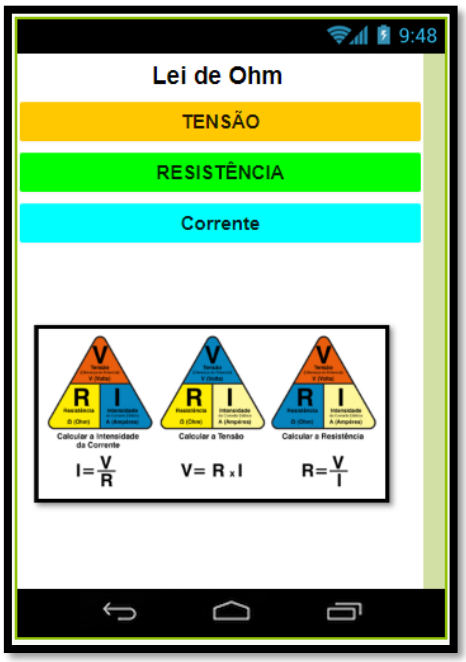


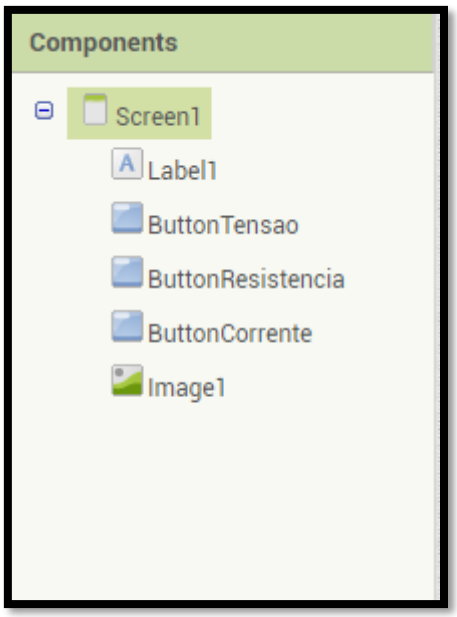


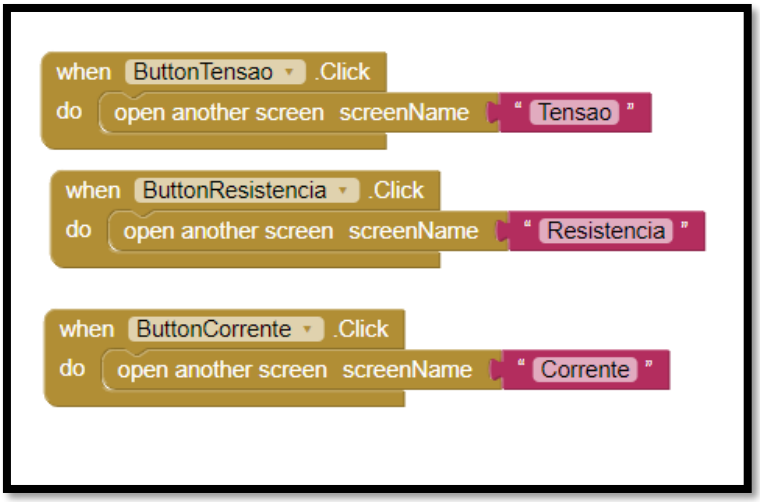


Vídeo: https://youtu.be/TaZBHa5QPtc

Exercício 9 – Lei de Ohm

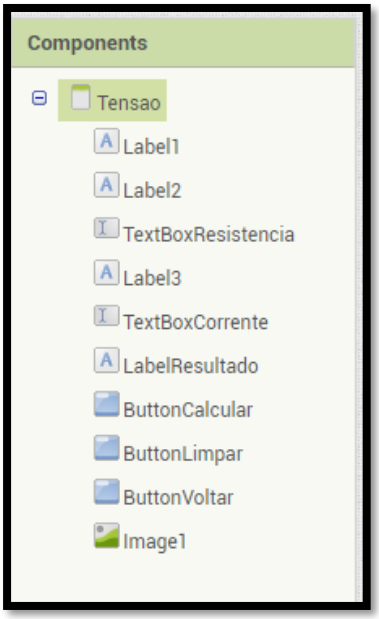


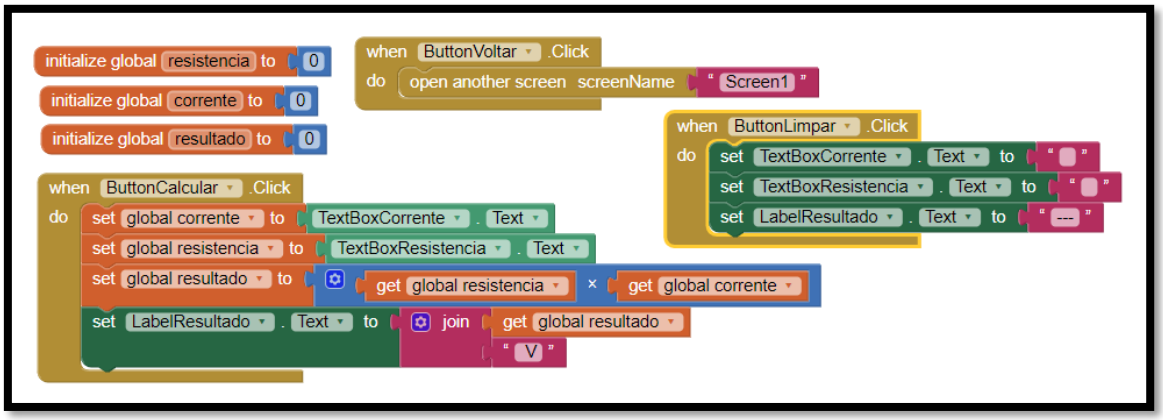


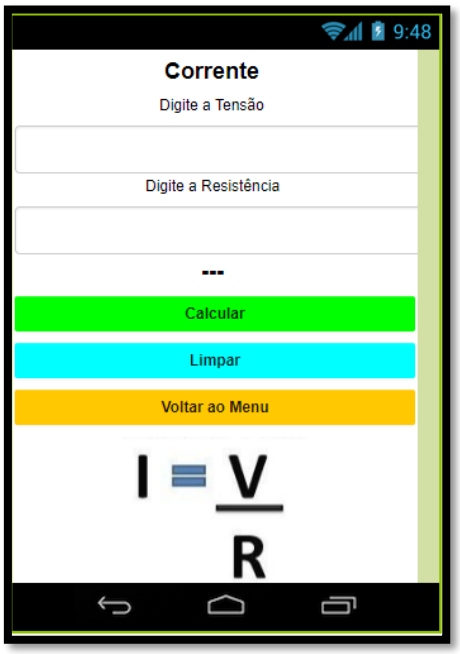



9:48
Tensão
Digite a Resisténcia
Digite a Corrente

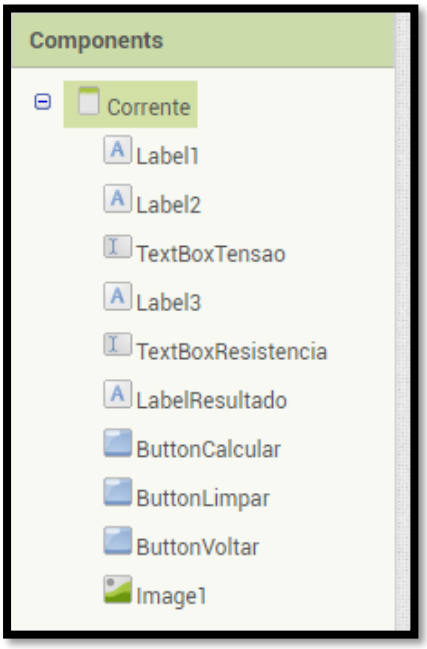
Calcular
Limpar
Voltar ao menu
V = I ✖ R

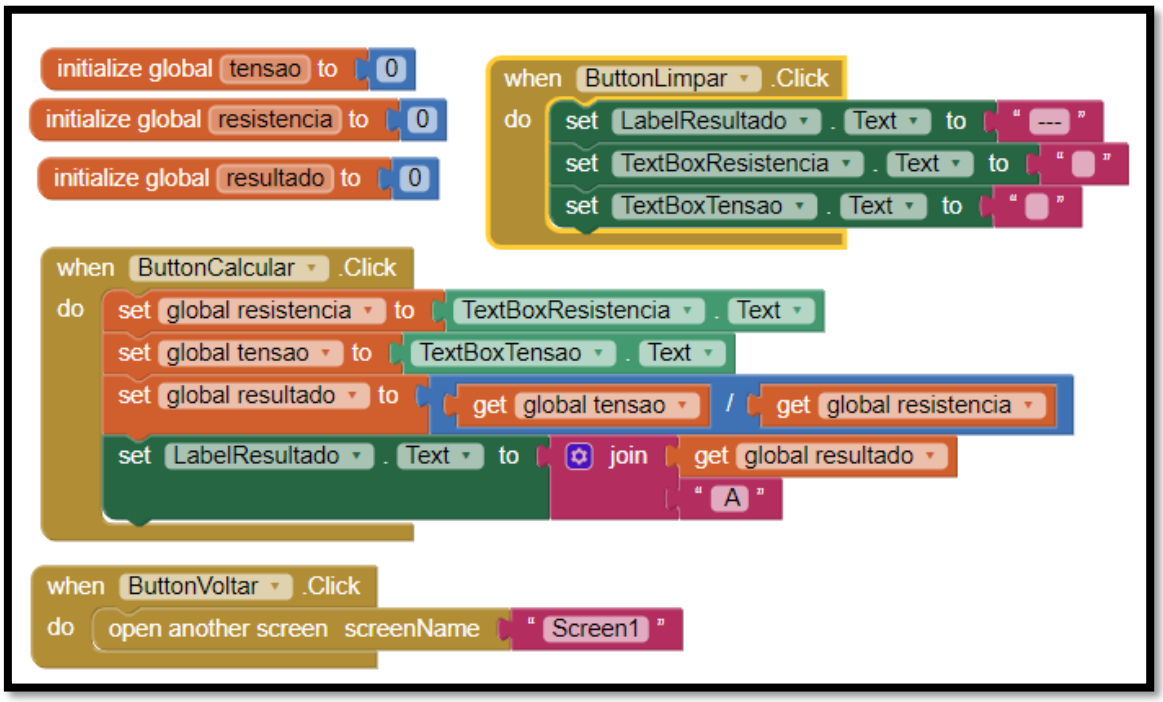

Components
Tensao
Label1
Label2
TextBoxResistencia
Label3
TextBoxCorrente
LabelResultado
ButtonCalcular
ButtonLimpar
ButtonVoltar
Image1

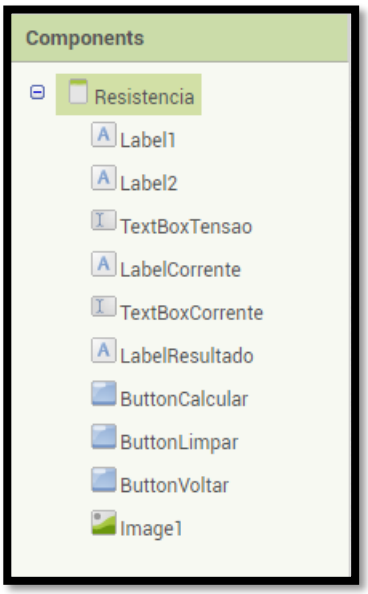

initialize global resistencia to 0
initialize global corrente to 0
initialize global resultado to 0
when ButtonVoltar .Click
do open another screen screenName " Screen1 "
when ButtonLimpar .Click
do set TextBoxCorrente . Text to " "
set TextBoxResistencia . Text to " "
set LabelResultado . Text to " --- "
when ButtonCalcular .Click
do set global corrente to TextBoxCorrente . Text
set global resistencia to TextBoxResistencia . Text
set global resultado to get global resistencia × get global corrente
set LabelResultado . Text to join get global resultado
" V "

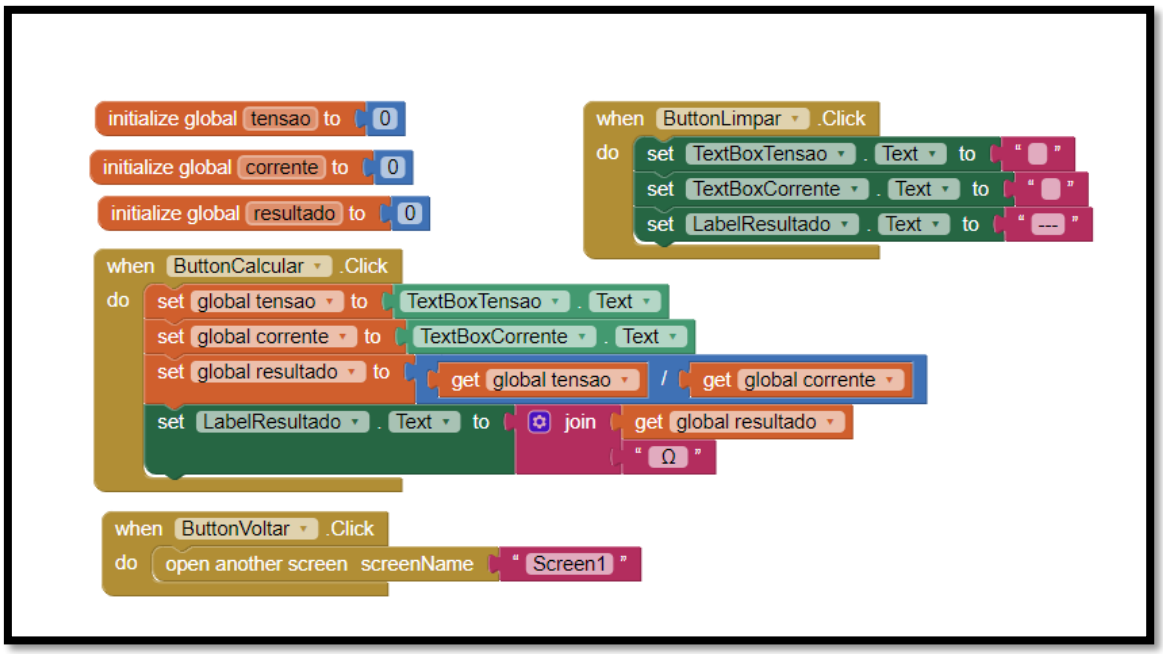

9:48
Corrente
Digite a Tensão
Digite a Resistência

Calcular
Limpar
Voltar ao Menu
I = V / R


Components
Corrente
Label1
Label2
TextBoxTensao
Label3
TextBoxResistencia
LabelResultado
ButtonCalcular
ButtonLimpar
ButtonVoltar
Image1

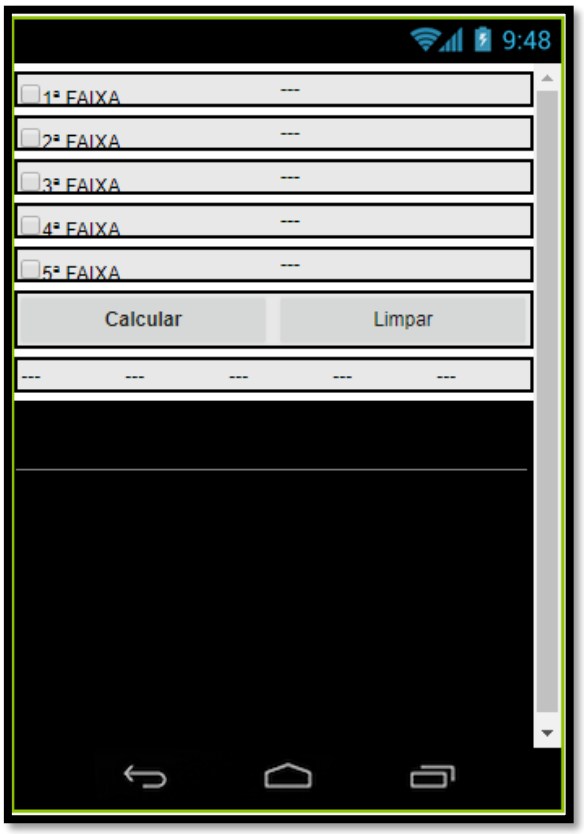

initialize global tensao to 0
initialize global resistencia to 0
initialize global resultado to 0
when ButtonLimpar .Click
do set LabelResultado . Text to "---"
set TextBoxResistencia . Text to ""
set TextBoxTensao . Text to ""
when ButtonCalcular .Click
do set global resistencia to TextBoxResistencia . Text
set global tensao to TextBoxTensao . Text
set global resultado to get global tensao / get global resistencia
set LabelResultado . Text to join get global resultado "A"
when ButtonVoltar .Click
do open another screen screenName "Screen1"

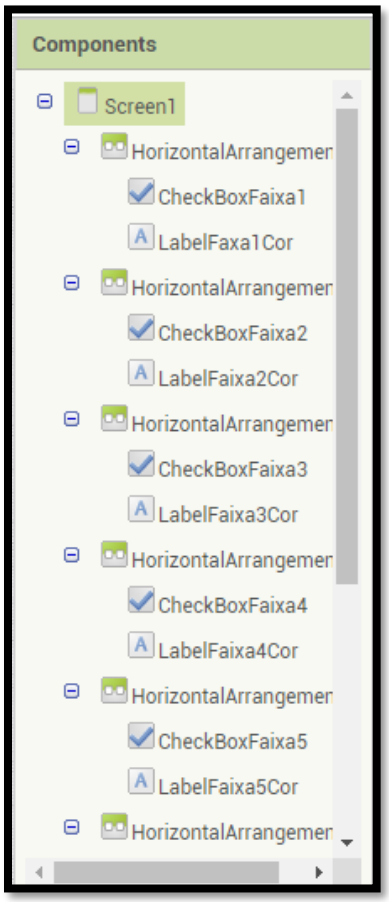


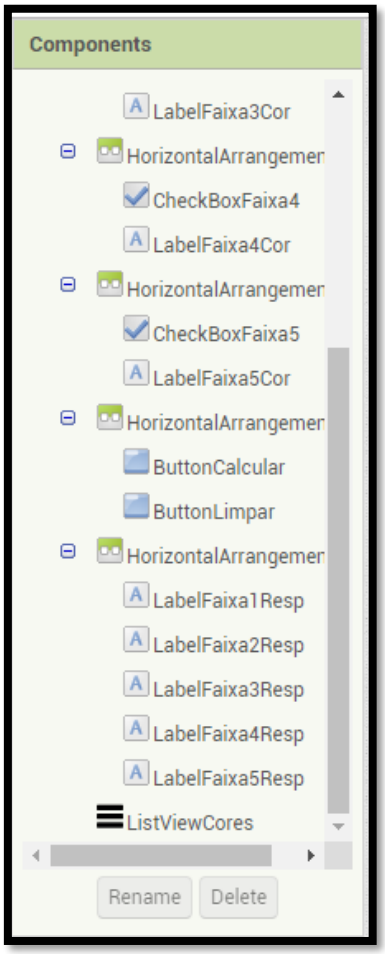


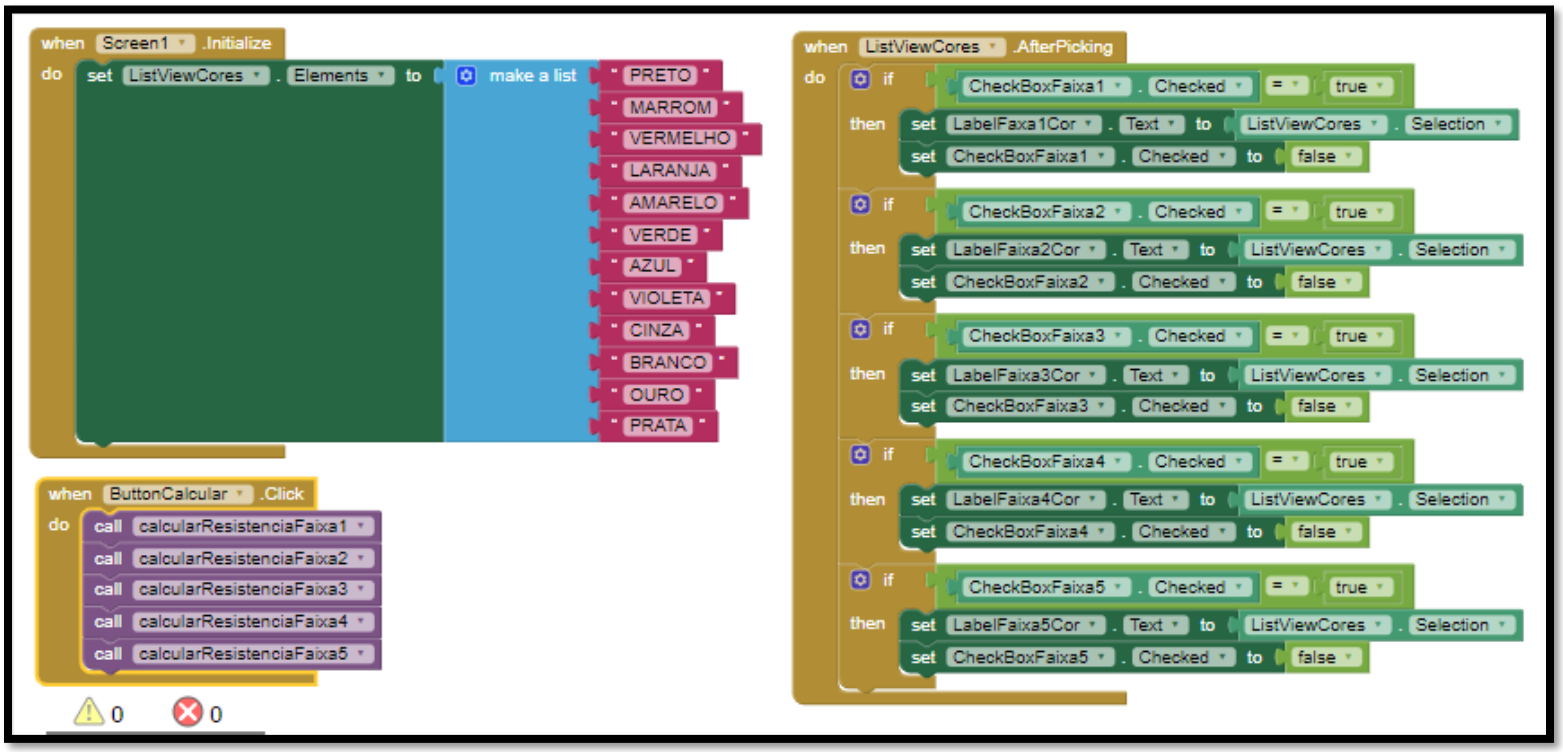
Vídeo: https://www.youtube.com/watch?v=Zc30dCy1H98&feature=youtu.be

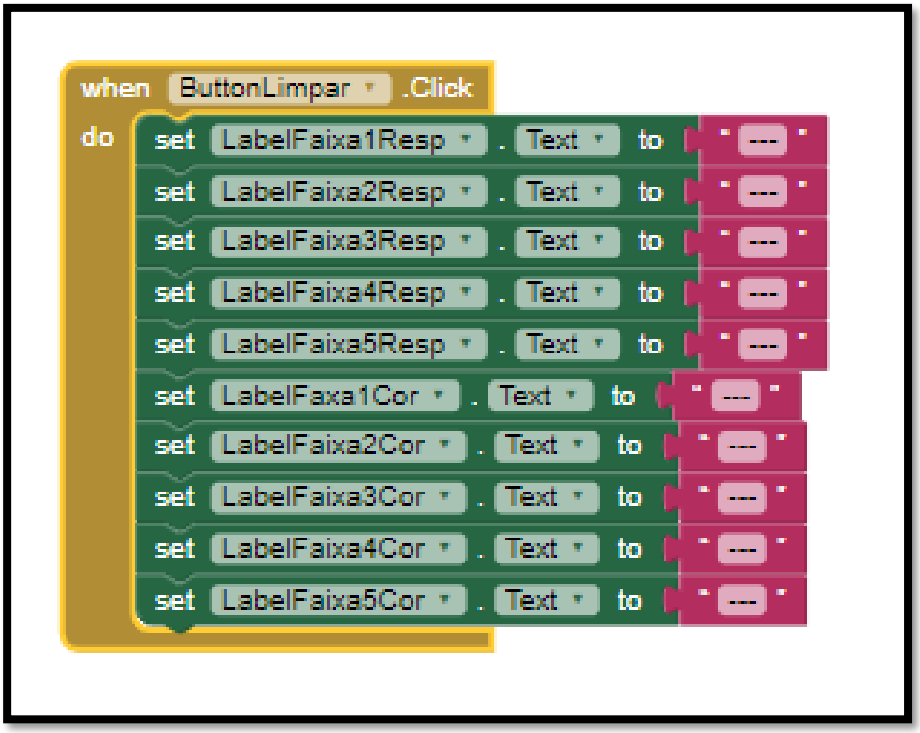
Exercício 10 – Resistor

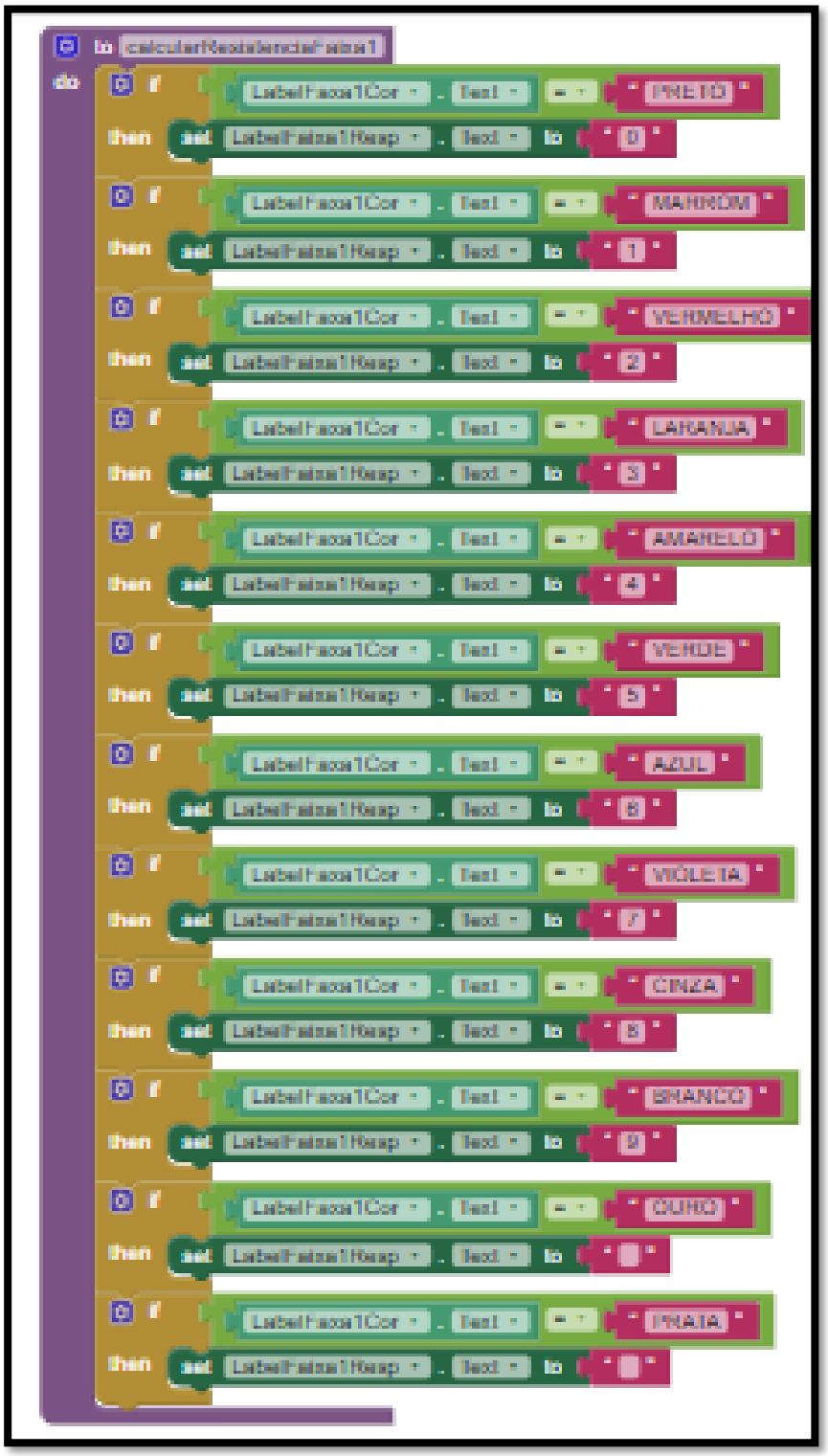


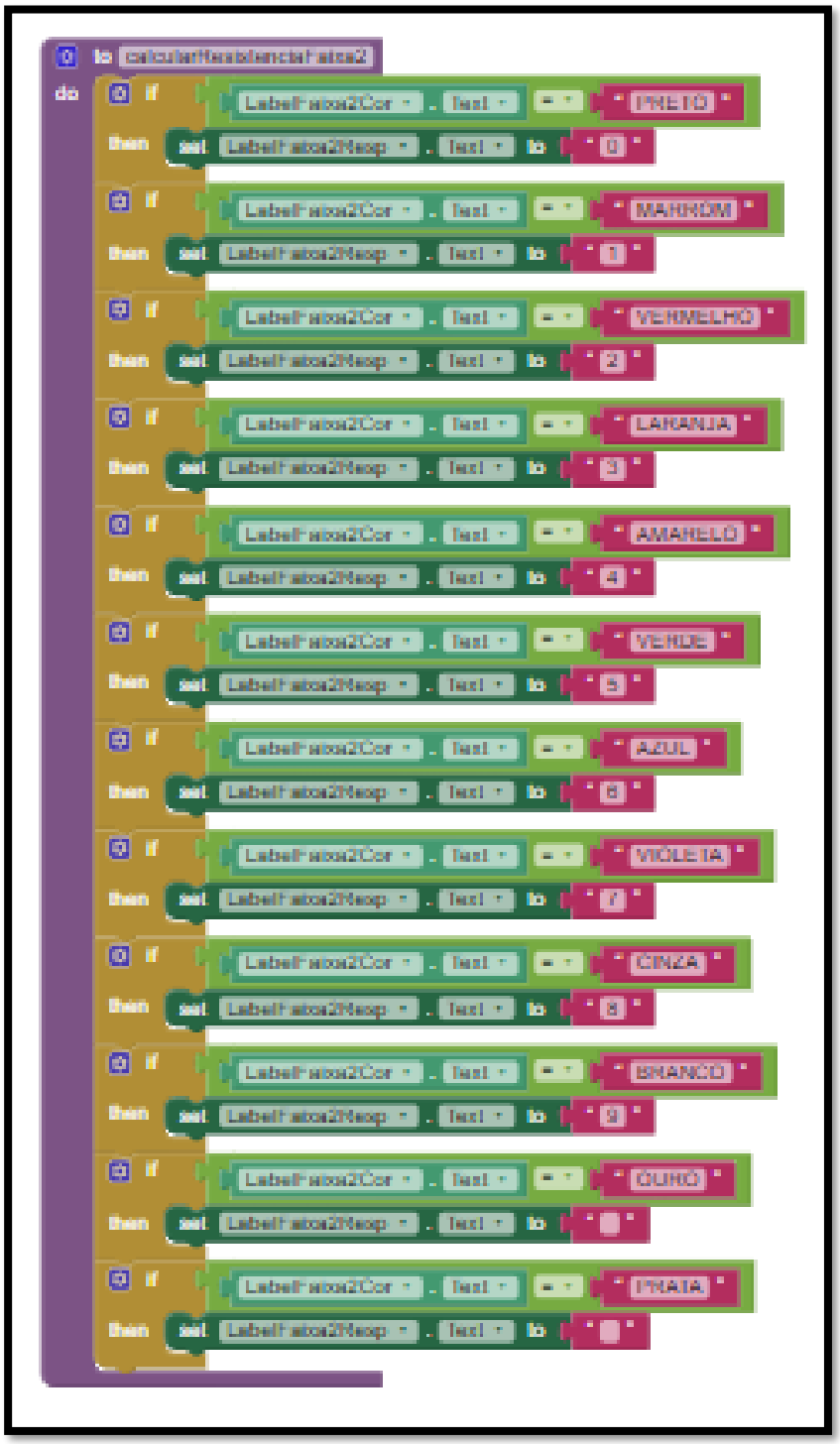


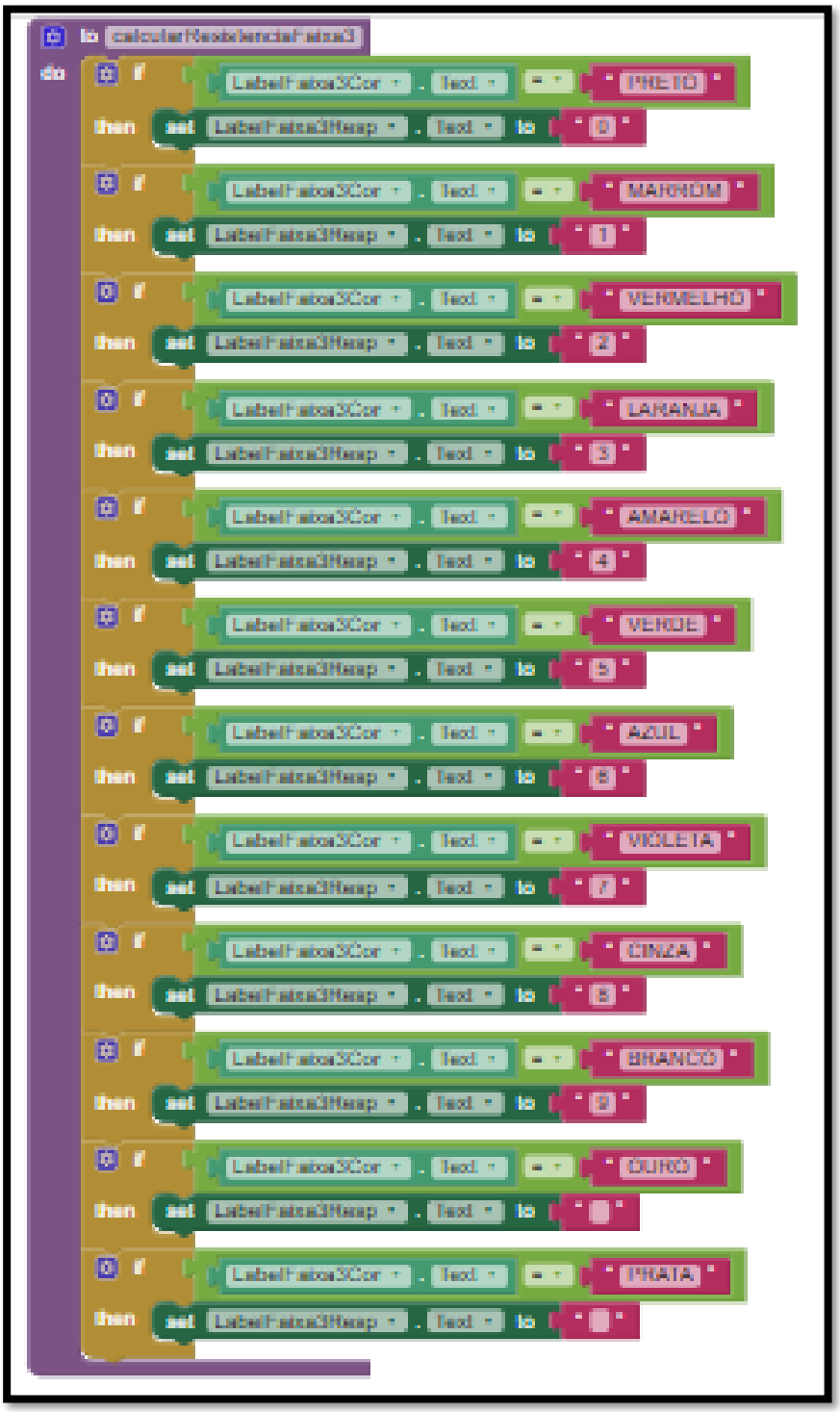


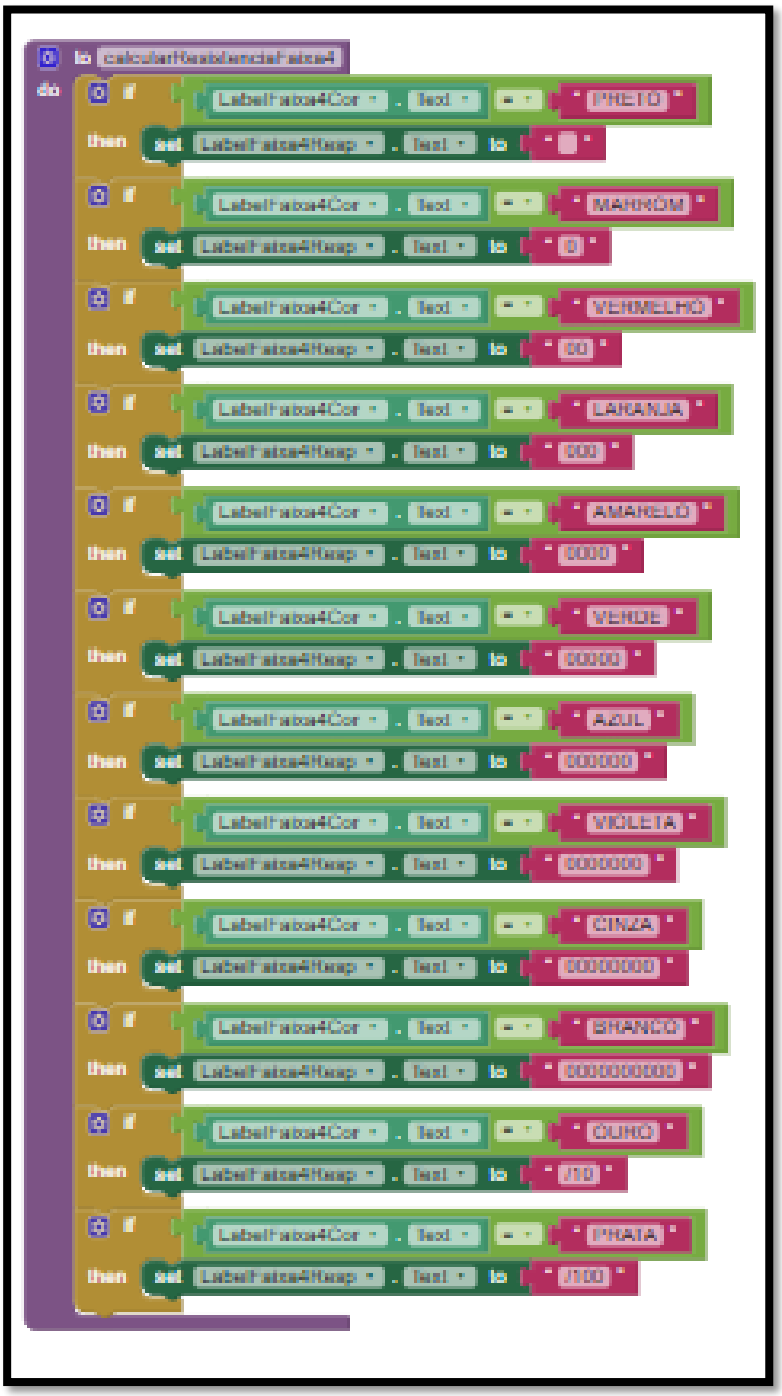

when Screen1 .Initialize
do set ListViewCores . Elements to make a list
PRETO
MARROM
VERMELHO
LARANJA
AMARELO
VERDE
AZUL
VIOLETA
CINZA
BRANCO
OURO
PRATA
when ButtonCalcular .Click
do call calcularResistenciaFaixa1
call calcularResistenciaFaixa2
call calcularResistenciaFaixa3
call calcularResistenciaFaixa4
call calcularResistenciaFaixa5
0
0
when ListViewCores .AfterPicking
do if CheckBoxFaixa1 . Checked = true
then set LabelFaxa1Cor . Text to ListViewCores . Selection
set CheckBoxFaixa1 . Checked to false
if CheckBoxFaixa2 . Checked = true
then set LabelFaixa2Cor . Text to ListViewCores . Selection
set CheckBoxFaixa2 . Checked to false
if CheckBoxFaixa3 . Checked = true
then set LabelFaixa3Cor . Text to ListViewCores . Selection
set CheckBoxFaixa3 . Checked to false
if CheckBoxFaixa4 . Checked = true
then set LabelFaixa4Cor . Text to ListViewCores . Selection
set CheckBoxFaixa4 . Checked to false
if CheckBoxFaixa5 . Checked = true
then set LabelFaixa5Cor . Text to ListViewCores . Selection
set CheckBoxFaixa5 . Checked to false

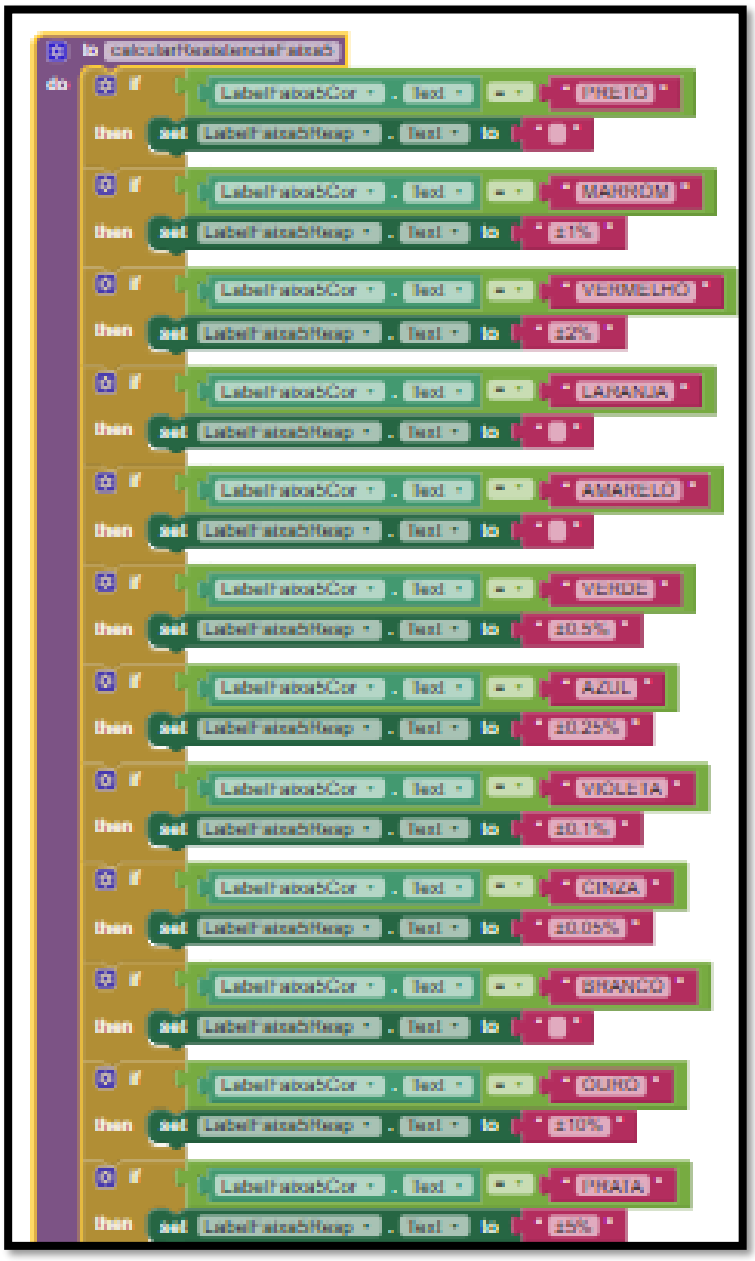

when ButtonLimpar .Click
do set LabelFaixa1Resp . Text to ---
set LabelFaixa2Resp . Text to ---
set LabelFaixa3Resp . Text to ---
set LabelFaixa4Resp . Text to ---
set LabelFaixa5Resp . Text to ---
set LabelFaxa1Cor . Text to ---
set LabelFaixa2Cor . Text to ---
set LabelFaixa3Cor . Text to ---
set LabelFaixa4Cor . Text to ---
set LabelFaixa5Cor . Text to ---


to calcularResistencia1faixa1
do
if LabelFaixa1Cor . Text = " PRETO "
then set LabelFaixa1Resp . Text to " 0 "
if LabelFaixa1Cor . Text = " MARROM "
then set LabelFaixa1Resp . Text to " 1 "
if LabelFaixa1Cor . Text = " VERMELHO "
then set LabelFaixa1Resp . Text to " 2 "
if LabelFaixa1Cor . Text = " LARANJA "
then set LabelFaixa1Resp . Text to " 3 "
if LabelFaixa1Cor . Text = " AMARELO "
then set LabelFaixa1Resp . Text to " 4 "
if LabelFaixa1Cor . Text = " VERDE "
then set LabelFaixa1Resp . Text to " 5 "
if LabelFaixa1Cor . Text = " AZUL "
then set LabelFaixa1Resp . Text to " 6 "
if LabelFaixa1Cor . Text = " VIOLETA "
then set LabelFaixa1Resp . Text to " 7 "
if LabelFaixa1Cor . Text = " CINZA "
then set LabelFaixa1Resp . Text to " 8 "
if LabelFaixa1Cor . Text = " BRANCO "
then set LabelFaixa1Resp . Text to " 9 "
if LabelFaixa1Cor . Text = " OURO "
then set LabelFaixa1Resp . Text to " "
if LabelFaixa1Cor . Text = " PRATA "
then set LabelFaixa1Resp . Text to " "


to calcularResistencia2
do
if Label2Cor . Text = "PRETO"
then set Label2Resp . Text to "0"
if Label2Cor . Text = "MARROM"
then set Label2Resp . Text to "1"
if Label2Cor . Text = "VERMELHO"
then set Label2Resp . Text to "2"
if Label2Cor . Text = "LARANJA"
then set Label2Resp . Text to "3"
if Label2Cor . Text = "AMARELO"
then set Label2Resp . Text to "4"
if Label2Cor . Text = "VERDE"
then set Label2Resp . Text to "5"
if Label2Cor . Text = "AZUL"
then set Label2Resp . Text to "6"
if Label2Cor . Text = "VIOLETA"
then set Label2Resp . Text to "7"
if Label2Cor . Text = "CINZA"
then set Label2Resp . Text to "8"
if Label2Cor . Text = "BRANCO"
then set Label2Resp . Text to "9"
if Label2Cor . Text = "OURO"
then set Label2Resp . Text to
if Label2Cor . Text = "PRATA"
then set Label2Resp . Text to


to calcularResistenciaFaixa3
do
if LabelFaixa3Cor . Text = PRETO
then set LabelFaixa3Resp . Text to 0
if LabelFaixa3Cor . Text = MARROM
then set LabelFaixa3Resp . Text to 1
if LabelFaixa3Cor . Text = VERMELHO
then set LabelFaixa3Resp . Text to 2
if LabelFaixa3Cor . Text = LARANJA
then set LabelFaixa3Resp . Text to 3
if LabelFaixa3Cor . Text = AMARELO
then set LabelFaixa3Resp . Text to 4
if LabelFaixa3Cor . Text = VERDE
then set LabelFaixa3Resp . Text to 5
if LabelFaixa3Cor . Text = AZUL
then set LabelFaixa3Resp . Text to 6
if LabelFaixa3Cor . Text = VIOLETA
then set LabelFaixa3Resp . Text to 7
if LabelFaixa3Cor . Text = CINZA
then set LabelFaixa3Resp . Text to 8
if LabelFaixa3Cor . Text = BRANCO
then set LabelFaixa3Resp . Text to 9
if LabelFaixa3Cor . Text = OURO
then set LabelFaixa3Resp . Text to
if LabelFaixa3Cor . Text = PRATA
then set LabelFaixa3Resp . Text to


to calcularfaixainicialfaixa4
do
if Label1faixa4Cor . Text = "PRETO"
then set Label1faixa4Resp . Text to "0"
if Label1faixa4Cor . Text = "MARROM"
then set Label1faixa4Resp . Text to "0"
if Label1faixa4Cor . Text = "VERMELHO"
then set Label1faixa4Resp . Text to "00"
if Label1faixa4Cor . Text = "LARANJA"
then set Label1faixa4Resp . Text to "000"
if Label1faixa4Cor . Text = "AMARELO"
then set Label1faixa4Resp . Text to "0000"
if Label1faixa4Cor . Text = "VERDE"
then set Label1faixa4Resp . Text to "00000"
if Label1faixa4Cor . Text = "AZUL"
then set Label1faixa4Resp . Text to "000000"
if Label1faixa4Cor . Text = "VIOLETA"
then set Label1faixa4Resp . Text to "0000000"
if Label1faixa4Cor . Text = "CINZA"
then set Label1faixa4Resp . Text to "00000000"
if Label1faixa4Cor . Text = "BRANCO"
then set Label1faixa4Resp . Text to "000000000"
if Label1faixa4Cor . Text = "OURO"
then set Label1faixa4Resp . Text to "0.1"
if Label1faixa4Cor . Text = "PRATA"
then set Label1faixa4Resp . Text to "0.01"



Vídeo: https://youtu.be/d2ay6fgzzCg