Faculdade de Tecnologia de Americana – Ministro Ralph Biasi

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Danilo Henrique Costa Souza

Aplicativo mobile para triciclo de pintura viária automatizado

Americana, SP

2019

Faculdade de Tecnologia de Americana – Ministro Ralph Biasi

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Danilo Henrique Costa Souza

Aplicativo Mobile para triciclo de pintura viária automatizado

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do (a) Prof. Me. Clerivaldo José Roccia

Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, SP.

2019

S714a SOUZA, Danilo Henrique Costa.

Aplicativo Mobile para triciclo de pintura viária automatizado. / Danilo Henrique Costa Souza. – Americana, 2019.

62f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Clerivaldo José Roccia

1 Android – aplicativos 2. Dispositivos móveis - aplicativo I. ROCCIA, Clerivaldo José II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.519

Fatec Americana Ministro Ralph Biasi

Centro Paula Souza 50 anos 1969 • 2019

São Paulo Governo do Estado

Faculdade de Tecnologia de Americana

Danilo Henrique Costa Souza

APLICATIVO MOBILE PARA TRICICLO DE PINTURA VIÁRIA AUTOMATIZADO

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana.
Área de concentração: Análise e Desenvolvimento de Sistemas

Americana, 10 de dezembro de 2019.

Banca Examinadora:

Prof. Me. Clerivaldo José Roccia
Mestre
Faculdade de Tecnologia de Americana

Prof. Me. Alberto Martins Júnior
Mestre
Faculdade de Tecnologia de Americana

Prof. Me. Murilo Fujita
Mestre
Faculdade de Tecnologia de Americana

AGRADECIMENTOS

Agradeço primeiramente a Deus, minha esposa Valquiria e minha mãe que nunca me deixaram desistir de concluir minha graduação. Agradeço a bibliotecária da Faculdade Anhanguera de Santa Bárbara D' Oeste Mirele Victoriano com referências bibliográficas. Também agradeço imensamente ao Prof. Clerivaldo José Roccia por orientar na pesquisa e apresentação desse trabalho, e que não me deixou desistir.

DEDICATÓRIA

Dedico esse trabalho aos meu pais e esposa, pelo apoio e paciência na jornada dessa graduação, aos professores e amigos que aqui fiz e que eternamente estarão em minhas lembranças.

RESUMO

Esse trabalho de conclusão de curso apresenta o desenvolvimento de um aplicativo mobile para um triciclo de pintura viária automatizado, esse aplicativo tem como objetivo principal facilitar a iteração do usuário com o sistema automatizado de pintura do triciclo, substituindo o atual painel eletrônico do sistema automatizado por um *tablet* ou *smartphone*. Por meio das telas do aplicativo o usuário irá escolher qual tipo de faixa deseja pintar e assim inserir as distâncias que as pistolas de pintura do triciclo ficarão ativadas e desativadas conforme o mesmo se desloca. As distâncias inseridas pelo usuário são enviadas pelo aplicativo via comunicação *bluetooth* para uma placa controladora Arduino que está instalada no triciclo, essa placa controladora recebe essas informações por meio de um módulo *bluetooth* onde a programação contida na placa faz o acionamento e desacionamento das pistolas de pintura por meio de relés.

Palavras Chave: Android; Arduino; Bluetooth

ABSTRACT

This course completion paper introduces the development of a mobile application for an automated road painting tricycle, the main purpose of which is to facilitate user iteration with the automated tricycle painting system by replacing the current automated system electronic panel with a tablet or smartphone. Through the application screens the user will choose which type of lane to paint and thus enter the distances that the tricycle spray guns will be activated and deactivated as it moves. The distances entered by the user are sent by the application via bluetooth communication to an Arduino controller board that is installed in the tricycle, this controller board receives this information through a bluetooth module where the programming contained in the board activates and deactivates the spray guns through relays.

Keywords: *Android; Arduino; Bluetooth*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APP – Aplicativo

IDE - Ambiente de Desenvolvimento Integrado

PWM – Pulse Width Modulation

RX – Pino da placa Arduino no qual receber dados seriais

TX – Pino da placa Arduino no qual transmitir dados seriais

TCC – Trabalho de Conclusão de Curso

XML – Extensible Markup Language

1 INTRODUÇÃO

Atualmente é notório que os *smartphones, tablets* e seus aplicativos estão cada vez mais inseridos nas atividades do cotidiano, mudando a forma que as pessoas compram, se relacionam e interagem com equipamentos ao seu redor. Um bom exemplo de como os aplicativos são utilizados para interagir com equipamentos ao nosso redor são as casas inteligentes que podem ser controladas por meio de *smartphones e tablets*, podendo o usuário ligar e desligar uma lâmpada ou simplesmente verificar se a mesma está acesa ou apagada através de um aplicativo. Segundo Sergio Luiz e Felipe Alberto.

> A ideia de controlar dispositivos de forma inteligente já não é nova. Podemos citar que, em 1894, o sérvio naturalizado estadunidense Nikola Tesla demonstrou os primeiros experimentos de comunicação sem fio. Em 1898, ele apresentou um pequeno barco controlado por rádio, equipamento que deu origem ao primeiro controle remoto (JR. SERGIO; FARINELLI, 2019, p.16).

Baseando-se na crescente utilização de aplicativos mobile e sua fácil iteração com o usuário para controlar, enviar e receber informações para diversos equipamentos sejam eles residenciais, industriais e comerciais, esse trabalho de conclusão de curso visa mostrar o desenvolvimento de um aplicativo para um triciclo de pintura de viária automatizado e seu impacto na iteração do usuário com o sistema.

É importante salientar que o aplicativo foi solicitado por uma empresa do ramo de fornecimento de equipamentos para pintura viária. Essa empresa não autorizou a inclusão total do código fonte do aplicativo e da placa Arduino nesse trabalho de conclusão de curso. Nos anexos desse trabalho encontra-se parte do código fonte da placa Arduino e do aplicativo, em especial a parte do código que faz o envio dos dados do aplicativo para placa Arduino via comunicação *bluetooth* e parte do código da placa Arduino que recebe os dados do aplicativo.

2 PROBLEMA

Atualmente o triciclo de pintura viária automatizado é fornecido com duas pistolas de pintura de alta pressão, uma moto bomba para pintura de alta pressão, uma placa Arduino Mega 2560 e seus periféricos de entrada e saída. A placa Arduino é o componente central de processamento automatizado do triciclo.

Figura 1 – Triciclo de pintura viária automatizado.

Fonte: Elaborado pelo autor

O triciclo está programado para pintar seis diferentes tipos de faixas que são: faixa contínua dupla; faixa contínua simples; faixa dupla seccionada; faixa simples seccionada; faixa seccionada a esquerda mais faixa contínua a direita; faixa contínua a esquerda mais faixa seccionada a direita. Para escolher o tipo de faixa que o triciclo irá pintar o operador por meio de um painel ligado a placa Arduino insere as distâncias desejadas que as pistolas de pintura irão ser acionadas e desacionadas durante o processo de pintura, o operador também habilita e desabilita as pistolas de pintura por meio de duas chaves fixadas nesse painel. Por meio da combinação das quatro distâncias inseridas e a ativação dessas duas

chaves é definido o tipo de faixa a ser pintada pelo triciclo automatizado. O problema em questão é a escolha do operador referente ao tipo de faixa a ser pintada, sendo essa escolha é definida por meio da combinação das distâncias inseridas no painel mais a ativação ou não das duas chaves, causando assim muita dificuldade em operar o triciclo de pintura automatizado. Para minimizar esse problema foi criado um manual onde o mesmo contém instruções sobre a escolha do tipo de faixa a ser pintada e as combinações entre as distâncias inseridas mais o estado das duas chaves. Esse manual foi utilizado em alguns testes, porém não obteve sucesso, onde o operador alegou ainda ter dificuldades em escolher o tipo de faixa a ser pintada.

Vale salientar que os botões do tipo *push button* responsáveis pela navegação no menu do display LCD e a inserção das distâncias apresentam mal contato quando pressionados, sendo necessário precisar o mesmo botão diversas vezes para surgir efeito.

O painel que o operador do triciclo interage é composto por um display de LCD16x2, cinco botões do tipo *push button* e duas chaves.

Figura 2 – Painel do triciclo de pintura viária automatizado.

Fonte: Elaborado pelo autor

3 OBJETIVO

O objetivo desse trabalho é mostrar o desenvolvimento de um aplicativo mobile para um triciclo de pintura viária automatizado, facilitando a iteração do usuário e substituindo o atual painel controlador de inserção das distâncias por um tablet ou smartphone. O aplicativo foi desenvolvido para sistema Android já que o cliente tem intenção de comercializar o triciclo juntamente com um *tablet* e com o aplicativo já instalado, ficando agregados ao valor final do triciclo.

3.1 Objetivo Específico

Desenvolvimento de um aplicativo para que o usuário escolha por meio de uma tela principal qual tipo de faixa o triciclo automatizado irá pintar, assim que o usuário escolher o tipo de faixa através de um botão em formato de imagem outra tela será carregada tendo como parâmetros de entrada as distâncias desejadas e o acionamento das pistolas de pintura por meio de um botão. Desse modo as pistolas de pintura um e dois serão acionados durante o processo de pintura enquanto o triciclo se descola. No capítulo três desse trabalho serão mostradas a tela principal e as telas referente a cada tipo de faixa. Logo após o usuário inserir as distâncias de acionamento e desacionamento o mesmo pressionará um botão denominado confirmar distâncias. Quando esta situação ocorrer o aplicativo enviará por comunicação *bluetooth* as distâncias inseridas para placa Arduino que tem um módulo *bluetooth* conectado para receber essas informações. Com essas etapas concluídas a placa Arduino se encarregará de acionar e desacionar as pistolas de pintura enquanto o triciclo de pintura se desloca, ficando a cargo do operador do triciclo guiar o mesmo, caso ocorra algum problema durante o processo de pintura o operador poderá comandar o desacionamento das pistolas de pintura utilizando um botão denominado parar pintura, interrompendo o processo instantaneamente.

4 JUSTIFICATIVA

O desenvolvimento desse aplicativo surgiu da necessidade em substituir o atual painel eletrônico do triciclo de pintura viária automatizado por um tablet ou smartphone tornando assim mais fácil a escolha do tipo de faixa e inserção das distâncias pelo operador do triciclo, o aplicativo apresenta um *design* minimalista e de fácil iteração do usuário. O desenvolvimento desse aplicativo também se justifica na crescente utilização de tablets e smartphones em controlar diversos equipamentos.

As tecnologias e ferramentas utilizadas para desenvolvimento desse aplicativo foram escolhidas levando em consideração seu baixo custo e facilidade de implementação ao sistema automatizado do triciclo.

4.1 Sistemas Similares

Antes de desenvolver o aplicativo, foi verificado por aplicativos similares e gratuitos disponíveis na internet. Essa busca foi realizada na plataforma Google Play, porém nenhum dos aplicativos tinha a funcionalidade de customização de telas que o cliente solicitou. Levando esses aspectos em consideração foi elaborada uma tabela com as funcionalidades dos aplicativos similares e o desenvolvido:

- S1: Arduino Bluetooth RC 4 Channel
- S2: Arduino Bluetooth Controller
- S3: Bluetooth Control for Arduino
- S4: App - Desenvolvido

Tabela 1 – Sistemas similares, comparativo de funcionalidades entre aplicativos.

Funcionalidade	**S1**	**S2**	**S3**	**S4**
Aplicativo Pago	X			X
Multiplataforma				
Gera relatório				
Embarcado	X	X	X	X
Manual de utilização	X	X	X	X
Design Minimalista	X			X

Fonte: Elaborado pelo autor

Conforme a Tabela 1 mostra os sistemas S1 e S4 apresentaram as mesmas funcionalidades, porém o sistema S4 foi escolhido por conta da customização.

5 REFERENCIAL BIBLIOGRÁFICO

Nesse capítulo serão abordadas todas as referências de pesquisa para o desenvolvimento do aplicativo, apresentando a metodologia e tecnologias utilizadas nesse TCC (Trabalho de conclusão de curso), explicando e detalhando suas principais características.

5.1 Metodologia de Desenvolvimento Utilizada

A metodologia utilizada para o desenvolvimento do aplicativo foi a metodologia de prototipação, pois foi a que melhor se encaixou com o projeto. Depois de definido os requisitos funcionais e não funcionais do aplicativo e finalizado a criação das telas de iteração com o usuário, foi dado início a implementação do código referente a funcionalidade do aplicativo. Segundo Ian Sommerville:

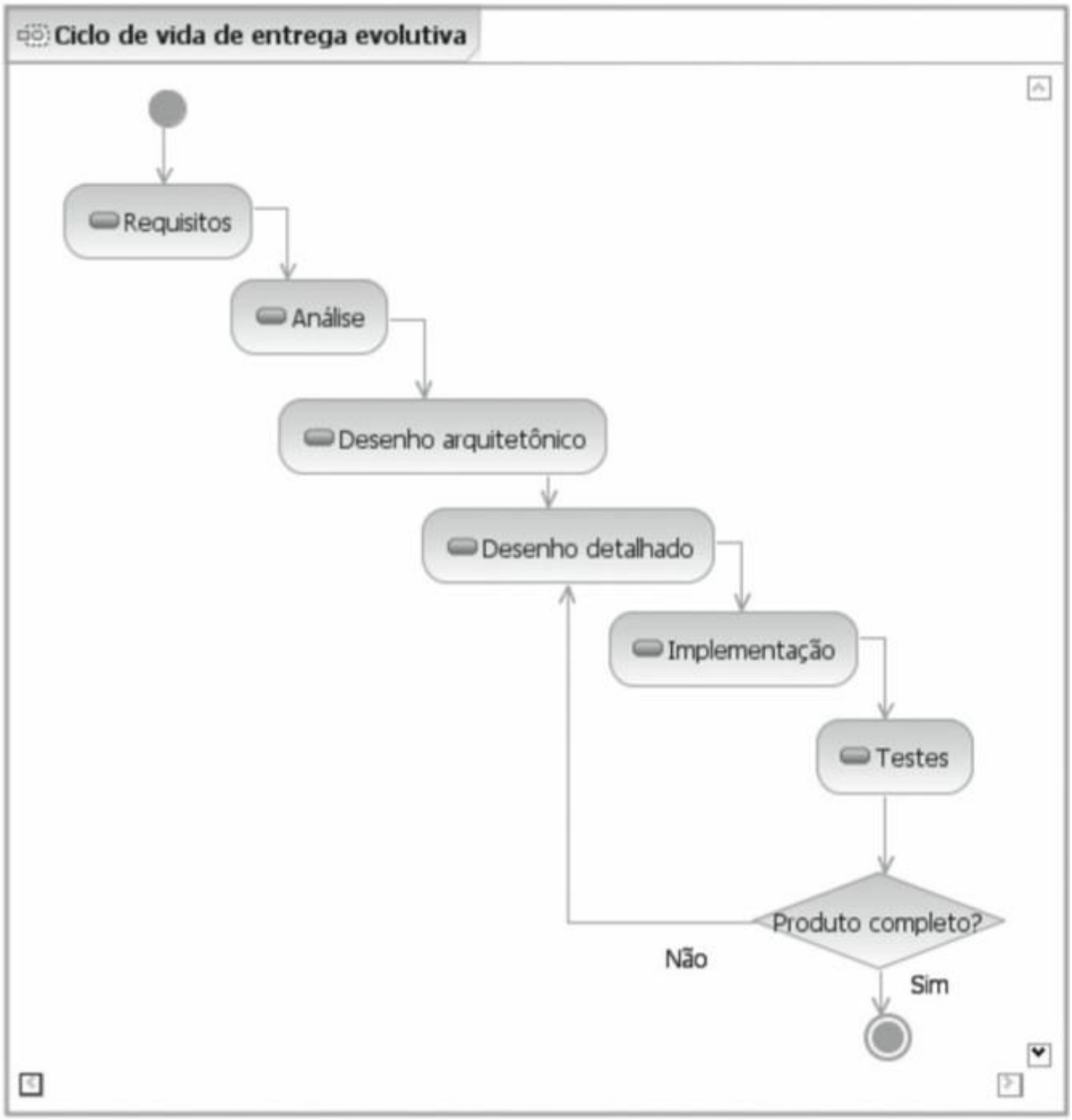
> Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções. O desenvolvimento rápido e iterativo do protótipo é essencial para que os custos sejam controlados e os Stakeholders do sistema possam experimentá-lo no início do processo de software (SOMMERVILLE, 2011, p. 30).

Para Wilson de Paula a metodologia de prototipagem é uma variação do modelo espiral, segundo o autor é:

> Uma variante do modelo em espiral é modelo de Prototipagem evolutiva. Nesse modelo, a espiral é usada não para desenvolver o produto completo, mas para construir uma série de versões provisórias que são chamadas de protótipos. Os protótipos cobrem cada vez mais requisitos, até que se atinja o produto desejado. Na prototipagem evolutiva, entretanto, o que se obtém ao final de cada iteração não é um produto completamente funcional, ao contrário do modelo em espiral verdadeiro (PADUA, 2009, p. 95).

Esse ciclo de vida de entrega evolutiva se mostrou bem aderente e funcional para o desenvolvimento do aplicativo. Esse modelo é melhor compreendido na figura abaixo.

Figura 3 – Ciclo de vida de entrega evolutiva.



Fonte: PADUA, 2009, p. 95.

Essa metodologia de prototipagem será melhor abordada no capítulo seis desse trabalho, onde será possível observar sua aplicação e importância em projetos que tem um prazo curto para serem entregues, e precisam ser constantemente testados.

5.2 A Placa Arduino

A placa Arduino surgiu dada a necessidade de estudantes de Design trabalhar com tecnologia, sendo um meio barato e de fácil utilização. Seu início foi na cidade de Ivrea na Itália no ano de 2005. O professor Massimo Banzi discutiu a necessidade da criação de tal tecnologia com um pesquisador visitante da Universidade de Malmö da Suécia, que também procurava uma solução semelhante. Assim nasceu o Arduino. (EVANS, 2013).

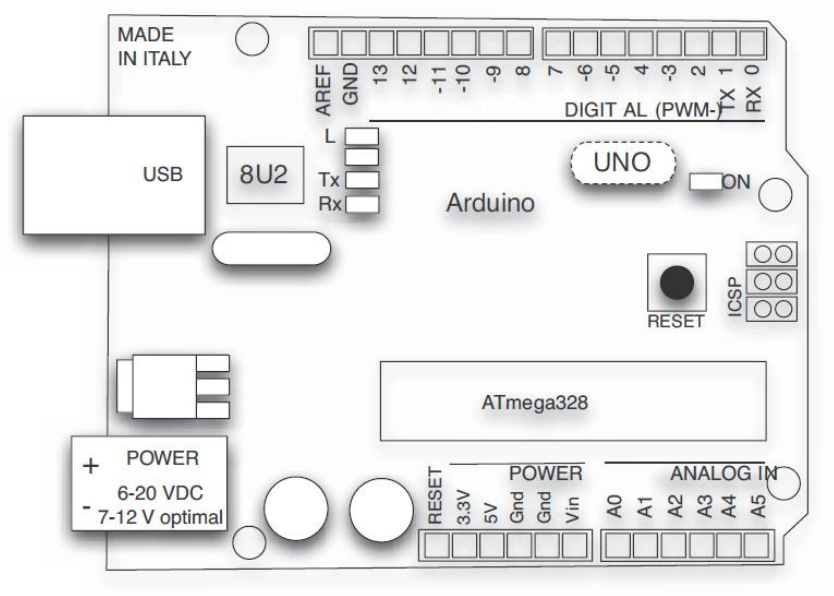
“A placa Arduino é basicamente um microcontrolador que administra todas as etapas envolvidas na medida e armazenagem dos dados.” (CARVALHO; AMORIM, 2014. p. 2).

Na placa de *hardware* Arduino é utilizado um microprocessador sendo esse o componente principal da placa, disposto na placa de circuito impresso, que por sua vez contém uma série de outros componentes eletrônicos para seu funcionamento e comunicação com outros dispositivos eletrônicos. Atualmente existem várias versões da placa Arduino cada uma com suas particularidades, uma das placas Arduino mais populares é o modelo UNO R3 que utiliza como base um microprocessador de 8 bits Atmel AVR. Segundo Denys Nicolosi um microprocessador é definido como:

> Um microprocessador é um elemento eletrônico, desenvolvido para executar tarefas específicas, com linguagem de comando específica. Ele se utiliza de uma Memória de Programa (Code Memory – ROM) para ler as instruções que deve executar e se utiliza de uma Memória de Dados (Data Memory - RAM) para armazenar temporariamente informações de uso próprio das instruções, enquanto essas informações devem ser armazenadas (Nicolosi, 2007, p.60).

A placa Arduino pode ser programada para executar instruções como ler um sinal de um sensor, enviar um comando para um motor, enviar dados para um computador entre outras inúmeras tarefas que podem ser executadas quando devidamente programada. A Figura 4 a seguir mostra o layout da placa ARDUINO modelo UNO.

Figura 4 – Layout e pinos da placa Arduino uno.



Fonte: EVANS, Martin, 2013, p. 27.

Atualmente são comercializados trinta e dois tipos de placa Arduino no site oficial. A que foi utilizada para o desenvolvimento desse projeto foi a Arduino Mega 2560, no capítulo seis na subseção 6.1 desse trabalho será detalhada melhor as características desse modelo de placa.

5.3. LINGUAGEM DE PROGRAMAÇÃO JAVA

A linguagem de programação Java se tornou uma das mais populares para programar e desenvolver aplicativos baseados na Internet e *softwares* para dispositivos que se comunicam por rede. Equipamentos de som e outros dispositivos utilizam a tecnologia Java para se conectarem em rede. Um dos principais dispositivos compatíveis com Java são os aparelhos celulares (DEITEL, 2010).

5.3.1. BREVE HISTÓRIA DA LINGUAGEM JAVA

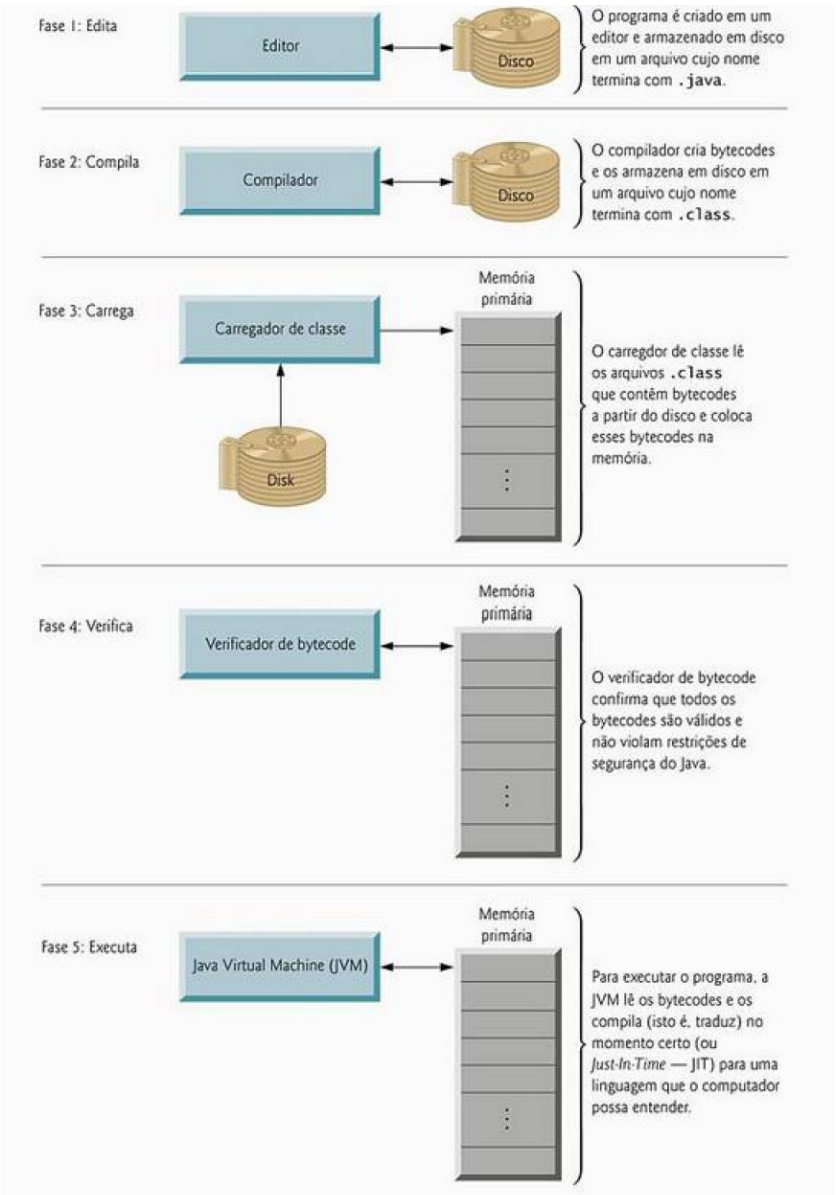
Em 1991 a Sun Microsystems financiou um projeto de pesquisa interna que foi resultado de uma linguagem baseada em C++, o criador dessa nova linguagem foi James Gosling, atribuindo o nome para nova linguagem como Oak em homenagem a uma árvore vista de sua janela da Sun, porém já havia uma outra linguagem de computador com esse nome. A ideia de chamar a nova linguagem de Java veio quando uma equipe da Sun foi a uma cafeteria local, o nome Java foi sugerido pois é o nome de uma cidade de origem de um tipo de café importado, e assim a nova

linguagem foi chamada. O Java foi oficialmente anunciado pela Sun em 1995 em uma conferência do setor (DEITEL, 2010).

5.3.2. ESTRUTURA DA LINGUAGEM JAVA

A sintaxe da linguagem Java é bem semelhante as linguagens C++ e C#, porém o modo como essa linguagem é compilada verificada e interpretada é o que faz se diferenciar das demais e tem como vantagem a portabilidade (DEITEL, 2010). A figura a seguir demonstra quais as fazes de um programa Java, desde sua edição até a JVM (*Java Virtual Machine*) interpretar e executar os *bytecodes*.

Figura 5 – Ambiente de desenvolvimento Java.



Fonte: DEITEL, 2010, p. 9.

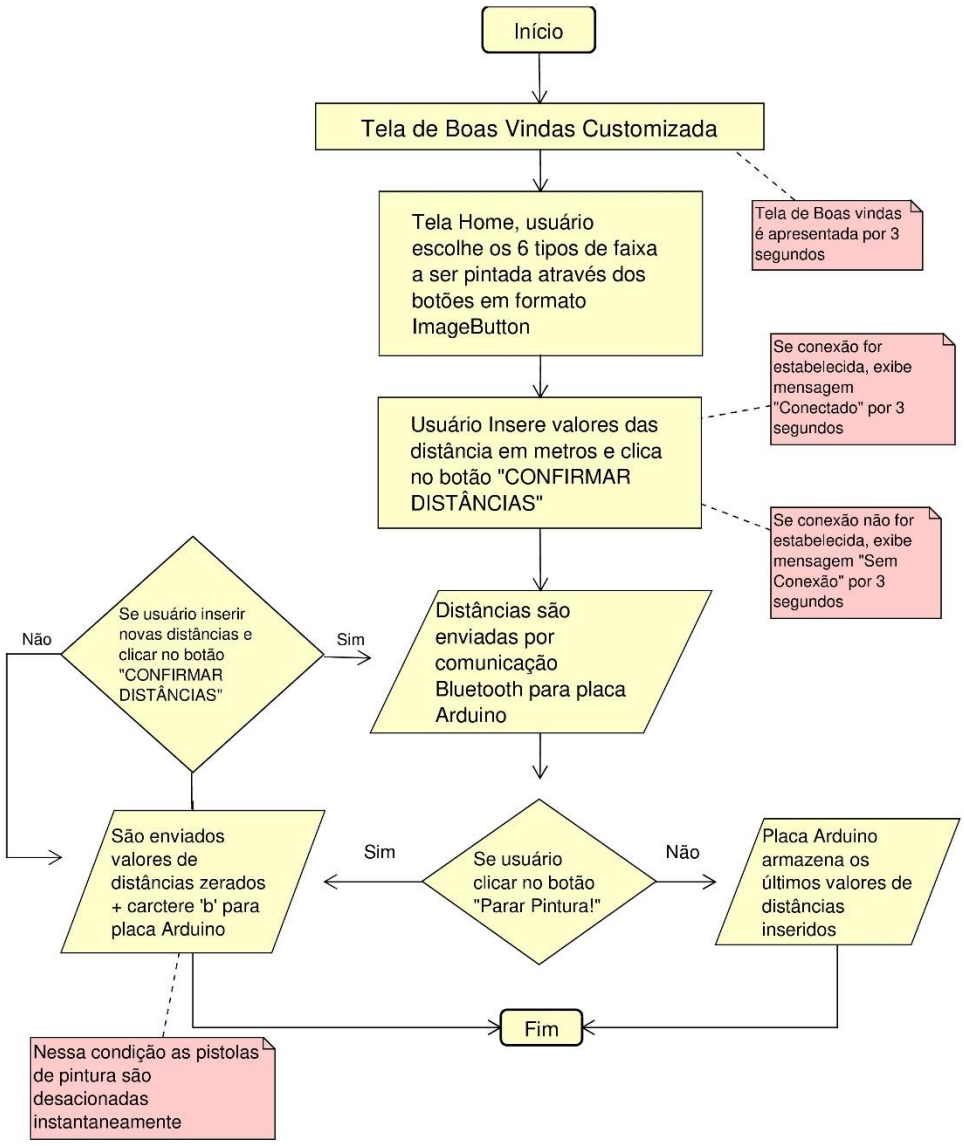
6 DESENVOLVIMENTO

Nesse capítulo será mostrado o desenvolvimento da programação do aplicativo, da placa Arduino e os recursos e ferramentas utilizadas.

Para o desenvolvimento do *layout* do aplicativo foram consideradas algumas das dez heurísticas de Nielsen, que presa por um *design* minimalista e funcional.

Antes de iniciar o desenvolvimento das telas do aplicativo e a programação necessária para o envio das distâncias via comunicação *bluetooth*, foi elaborado um fluxograma a fim de facilitar o desenvolvimento conforme Figura 6 abaixo.

Figura 6 – Fluxograma do Aplicativo.



Fonte: Elaborado pelo autor.

6.1 RECURSOS E FERRAMENTAS

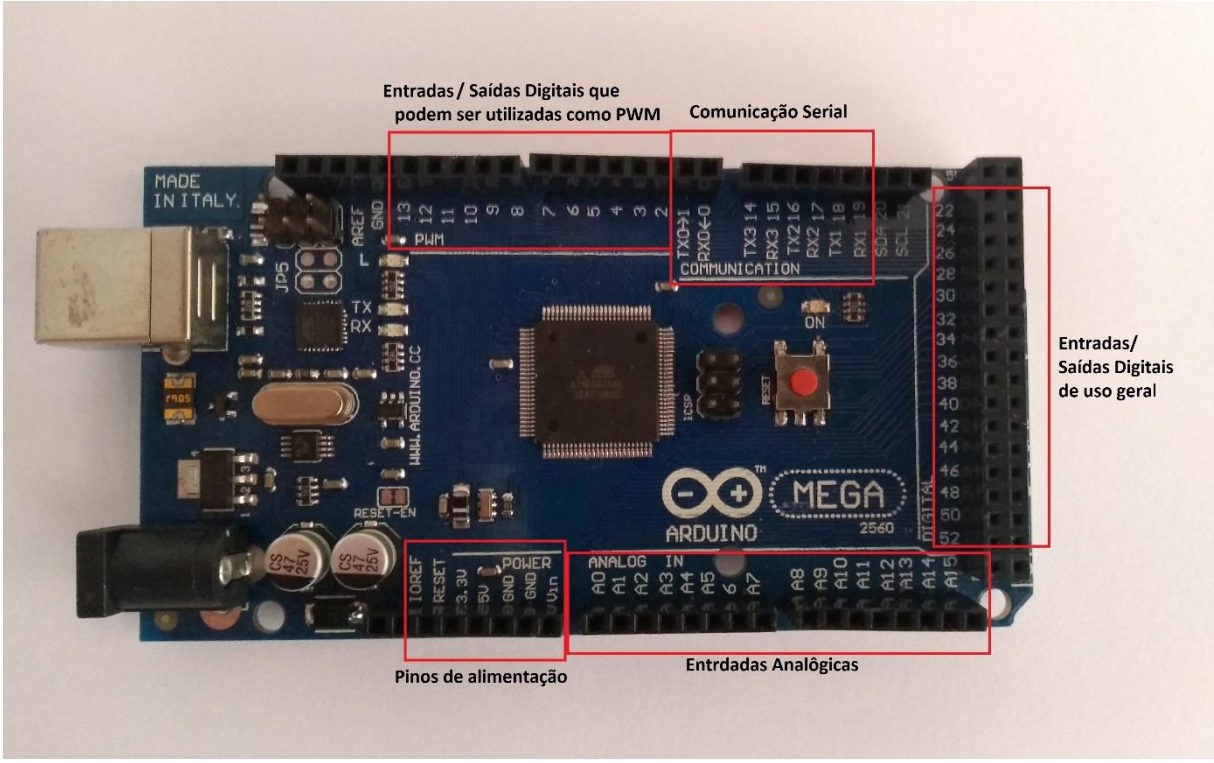
Nessa subseção será apresentada as ferramentas de programação e recursos necessários para o desenvolvimento do sistema:

- Android Studio: O Android Studio é uma IDE oficial para o desenvolvimento de aplicativos Android. Atualmente está na sua versão estável 3.4.1.
- Arduino IDE: É um ambiente de desenvolvimento integrado usado para programação em placas Arduino, essa IDE é a oficial e disponibiliza constates atualizações de bibliotecas e placas. Com essa IDE também é possível programar outras placas. Atualmente sua versão estável é a 1.8.10.
- Astah Professional: É uma ferramenta que suporta os requerimentos da UML 2.x para construção de diagramas de classe, caso de uso, sequência, máquina de estado, atividade, componente, dentre outros (ASTAH, 2018). É a ferramenta utilizada para a construção dos diagramas de classe e caso de uso que posteriormente será utilizada pelos desenvolvedores de *software*, sua versão utilizada foi a 8.2.0.

A placa Arduino utilizada para controle automatizado das pistolas do triciclo de pintura foi a Arduino Mega 2560, a escolha desse modelo de placa se deu, pois, a mesma contém cinquenta e quatro portas digitais que podem ser utilizadas como entrada e saída, quinze portas analógicas e quinze que podem ser utilizadas como PWM (*Pulse Width Modulation*). Outro fator interessante nesse modelo de placa é a quantidade de portas destinadas a comunicação serial, nesse modelo são disponibilizados 4 canais de comunicação serial, sendo que desses 4 canais os pinos digitais 0 e 1 correspondem ao canal de comunicação utilizado para descarregar os programas na placa Arduino, restando ainda 3 canais de comunicação serial. Esse fator foi de grande ajuda pois o módulo *bluetooth* HC-06 também trabalha com comunicação serial. Desse modo toda vez que fosse necessário carregar um novo programa para placa Arduino o módulo *bluetooth* precisaria ser desconectado para não interferir na gravação do programa, caso o mesmo estivesse conectado nas portas 0 e 1, dessa maneira o módulo *bluetooth* foi conectado nas portas 16 e 17 da placa correspondente a TX e RX respectivamente. Para que o módulo *bluetooth* funcionasse corretamente e também fosse possível

monitorar o recebimento de dados na fase de testes realizados com o aplicativo, foi necessária a inclusão da biblioteca *SoftwareSerial.h*. Abaixo imagem da placa Arduino modelo 2560.

Figura 7 – Placa Arduino Mega 2560

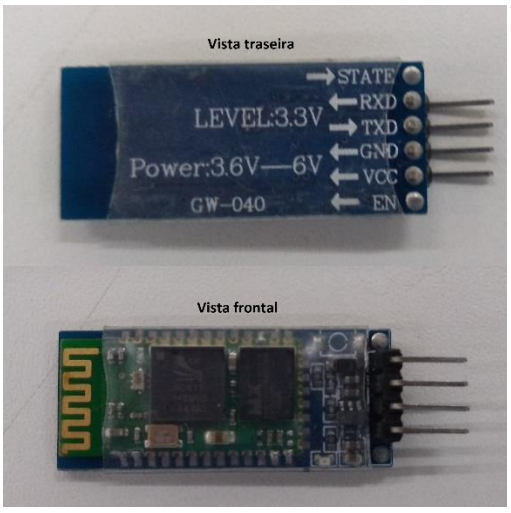


Fonte: Elaborado pelo autor.

O fluxograma da programação da placa Arduino e detalhamento das funções será tratada na subseção 6.4 desse capítulo.

Na Figura 8 abaixo pode-se observar os pinos de alimentação e comunicação do módulo *bluetooth* modelo HC-06 utilizado na placa Arduino, esse módulo trabalha com tensão de alimentação entre 3.6Volts a 6Volts.

Figura 8 – Módulo bluetooth HC-06



Fonte: Elaborado pelo autor.

6.2 DESENVOLVIMENTO DAS TELAS DO APLICATIVO

Para o desenvolvimento das telas do aplicativo foi utilizado a IDE Android Studio, a linguagem de marcação utilizada para criação das telas foi a XML (*Extensible Markup Language*), sendo que a própria IDE Android Studio indica se há algum erro na sintaxe. Mesmo a IDE Android Studio tendo recursos visuais do tipo clicar e arrastar para o desenvolvimento de telas, a edição da linguagem XML no modo texto se mostrou bem eficiente do ponto de vista de ajustes no *layout* para cada tela. Para tela principal foram adicionados seis botões do tipo *ImageButton* conforme pode-se observar na Figura 9, que ao clicar em cima da imagem correspondente ao tipo de faixa uma nova tela será apresentada ao usuário.

Figura 9 – Tela inicial do aplicativo

Fonte: Elaborado pelo autor

O código XML da tela principal e das demais telas do aplicativo estão disponíveis para consulta nos anexos desse trabalho. A seguir serão apresentadas as telas desenvolvidas e suas versões. Nas primeiras versões de tela desenvolvidas conforme Figura 10, quando o usuário clicar no *ImageButtom* referente a pintura de faixa dupla seccionada é possível notar o excesso de parâmetros que o mesmo deverá inserir antes de habilitar as pistolas de pintura um e dois.

Figura 10 – Tela da primeira versão para faixa dupla seccionada.

Fonte: Elaborado pelo autor.

Nessa primeira versão de tela para o tipo de faixa dupla seccionada é possível notar o excesso de parâmetros em que o usuário terá que inserir, e ainda habilitar as pistolas um e dois por meio das chaves virtuais denominadas “Habilita Pistola 1” e “Habilita Pistola 2", essas chaves virtuais são como interruptores utilizados para ligar e desligar uma lâmpada. Depois de alguns testes realizados com o aplicativo foi desenvolvida uma segunda versão dessa tela, mais simples e intuitiva, onde o usuário só precisa inserir duas distâncias referentes as pistolas acionadas e desaciondas, e por fim pressionar o botão confirmar distâncias para que as mesmas sejam enviadas ao módulo *bluetooth* da placa Arduino.

Um adendo importante nessa etapa do desenvolvimento é que a programação da placa Arduino ainda continua recebendo quatro parâmetros de distância, mesmo que o usuário insira no aplicativo somente dois, sendo facilmente implementado atribuindo o mesmo valor inserido na “Distância Pistola 1 aberta” para “Distância Pistola 2 aberta”, e seguindo a mesma lógica para o valor inserido na “Distância Pistola 1 Fechada” ser igual para “Distância Pistola 2 Fechada” já que se trata de uma faixa dupla seccionada e simétrica, na Figura 11 está a imagem da segunda versão da tela para faixa seccionada dupla contínua.

Figura 11 – Tela da segunda versão para faixa dupla seccionada.

Fonte: Elaborado pelo autor.

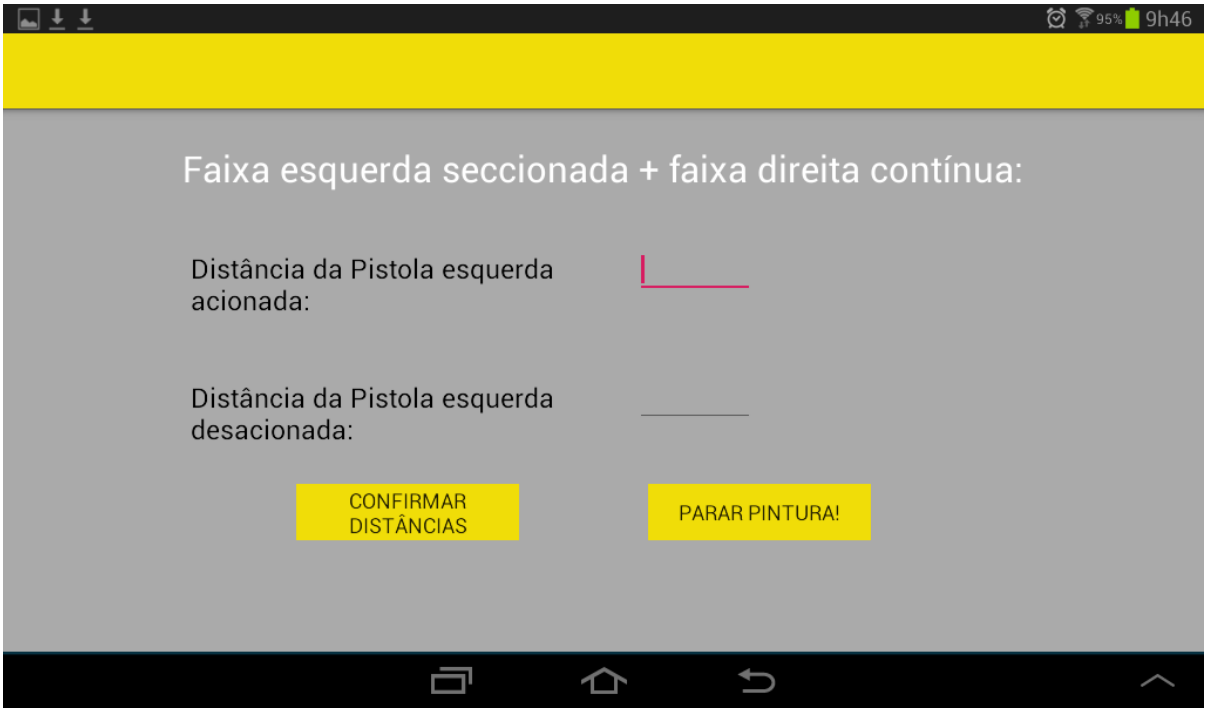
Mantendo o mesmo conceito minimalista foram desenvolvidas as demais telas referentes a cada tipo de faixa a ser pintada. Na Figura 12 pode-se observar a tela desenvolvida para pintura de faixa dupla contínua, sem a necessidade da inserção de parâmetros.

Figura 12 – Tela para faixa dupla contínua.

Fonte: Elaborado pelo autor.

Na Figura 13 abaixo pode-se observar a tela desenvolvida para o tipo de pintura de faixa esquerda seccionada mais faixa direita contínua.

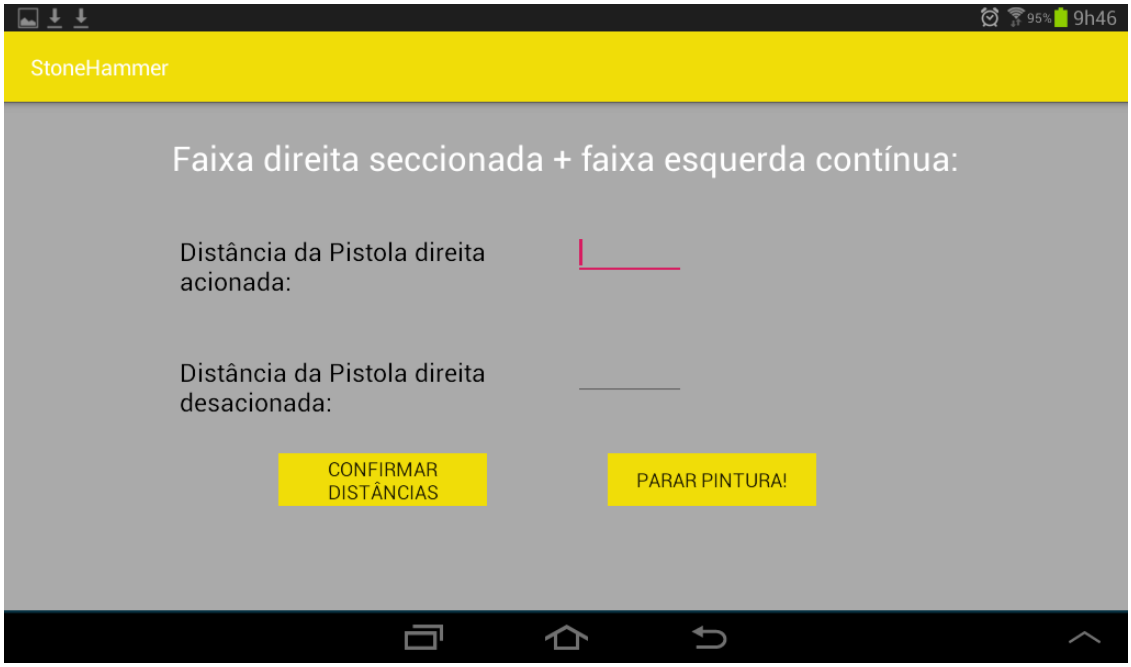
Figura 13 – Tela para faixa esquerda seccionada mais faixa direita contínua.



Fonte: Elaborado pelo autor.

Na Figura 14 abaixo é apresentada a tela desenvolvida para o tipo de pintura de faixa direita seccionada mais faixa esquerda contínua.

Figura 14 – Tela para faixa direita seccionada mais faixa esquerda contínua.



Fonte: Elaborado pelo autor.

A Figura 15 abaixo mostra a tela desenvolvida para o tipo de faixa simples seccionada.

Figura 15 – Tela para faixa simples seccionada.

Fonte: Elaborado pelo autor.

Na Figura 16 abaixo é apresentada a tela desenvolvida para o tipo de pintura de faixa simples contínua.

Figura 16 – Tela para faixa simples contínua.

Fonte: Elaborado pelo autor.

6.3 DESENVOLVIMENTO DA PROGRAMAÇÃO DO APLICATIVO

Nessa subseção será mostrado o desenvolvimento das partes mais relevantes da programação do aplicativo, em especial a parte do código responsável por enviar as distâncias inseridas pelo usuário para placa Arduino. A linguagem de programação utilizada na IDE Android Studio para desenvolver as funcionalidades do aplicativo foi a Java.

Primeiramente para que a comunicação *bluetooth* seja estabelecida entre o App (Aplicativo) e a placa Arduino, foi necessário inserir algumas linhas de código no arquivo “AndroidManifest.xml”, esse arquivo é criado automaticamente quando se inicia um novo projeto na IDE Android Studio, abaixo na Figura 17 está às três linhas de código necessárias para que o sistema operacional Android instalado em Tablet ou Smartphone permita a comunicação *bluetooth* entre o aplicativo e o módulo HC-06.

Figura 17 – Instruções que permitem a comunicação bluettoth do App.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="com.google.android.things.permission.USE_PERIPHERAL_IO" />
```

Fonte: Elaborado pelo autor.

Outra parte importe do código contido no arquivo “AndroidManifest.xml” são as linhas de instruções responsáveis por habilitar as telas toda vez que o usuário clicar no *ImageButton* correspondente à faixa desejada para pintura, também foi adicionado a instrução para que as telas do aplicativo sejam somente exibidas na posição horizontal, já que o tablet ficará fixado em um suporte no triciclo na posição horizontal.

Figura 18 – Instruções que habilita as telas e executa-as horizontalmente.

```
<activity android:name=".simplescontinua" android:screenOrientation="landscape"/>
<activity android:name=".simplesseccionada" android:screenOrientation="landscape"/>
<activity android:name=".direitasecciesquercont" android:screenOrientation="landscape"/>
<activity android:name=".esquerdracejadireitacont" android:screenOrientation="landscape"/>
<activity android:name=".Duplacontinua" android:screenOrientation="landscape"/>
<activity android:name=".Tracejadadupla" android:screenOrientation="landscape"/>
<activity android:name=".HomeActivity" android:screenOrientation="landscape"/>
<activity android:name=".MainActivity" android:screenOrientation="landscape">
```

Fonte: Elaborado pelo autor.

Nos anexos desse TCC está disponível o código completo do arquivo “AndroidManifest.xml” para consulta.

Como já mencionado anteriormente cada tela é responsável por enviar os parâmetros de distância via comunicação *bluetooth* para o módulo HC-06 conectado a placa Arduino de acordo com o tipo de faixa, assim que o usuário insere os valores das distâncias em metros e clica no botão denominado “*CONFIRMAR DISTÂNCIAS”* a linha de código responsável por enviar esses parâmetros cria um canal de comunicação entre o aplicativo e o módulo *bluetooth* enviando os valores inseridos nas *TextBox*, essa linha de código pode ser melhor compreendida na imagem abaixo.

Figura 19 – Linha de código responsável pelo envio dos parâmetros.

```
btSocket.getOutputStream().write(editText.getText().toString().getBytes());
```

Fonte: Elaborado pelo autor.

Pode-se observar que essa linha de código utiliza a função *btSocket* responsável por criar um canal de comunicação, capturar o valor contido na *TextBox* denominada “editText” em vermelho conforme Figura 19, converte o valor para o tipo *String* e envia os dados via *Bluetooth*.

Nas primeiras versões do App foi utilizada a função “*btSocket*” dentro da função *try*, desse modo esperava-se que os valores inseridos nas quatro *TextBox* das primeiras versões de tela fossem capturados pela programação da placa Arduino, o que não aconteceu, isso se deu pela falta de sincronia entre o App e a parte do código da placa Arduino responsável por capturar esses quatros valores, onde o código da placa Arduino capturava apenas o primeiro valor corretamente e os outros três valores restantes eram atribuídos valor zero, a imagem abaixo mostra essa parte do código:

Figura 20 – Primeira versão do código App utilizando a função btSocket

```
if(event.getAction() == MotionEvent.ACTION_DOWN)
  button1.setBackgroundColor(colorOn);
    try {
         btSocket.getOutputStream().write(editText.getText().toString().getBytes());
         btSocket.getOutputStream().write(editText2.getText().toString().getBytes());
         btSocket.getOutputStream().write(editText3.getText().toString().getBytes());
         btSocket.getOutputStream().write(editText4.getText().toString().getBytes());
        }
    catch (IOException e) {
    e.printStackTrace();
    }
```

Fonte: Elaborado pelo autor.

Para corrigir esse problema de sincronia entre o envio e captura dos parâmetros, foram feitas diversas tentativas de inclusão de tempo de espera entre o envio de cada parâmetro, uma dessas tentativas pode ser vista na imagem abaixo.

Figura 21 – Código App utilizando a função btSocket e a função wait

```
if(event.getAction() == MotionEvent.ACTION_DOWN)
  button1.setBackgroundColor(colorOn);
    try {
         btSocket.getOutputStream().write(editText.getText().toString().getBytes());
         wait(3000);//Tempo de espera em milissegundos
         btSocket.getOutputStream().write(editText2.getText().toString().getBytes());
         wait(3000);//Tempo de espera em milissegundos
         btSocket.getOutputStream().write(editText3.getText().toString().getBytes());
         wait(3000); //Tempo de espera em milissegundos
         btSocket.getOutputStream().write(editText4.getText().toString().getBytes());
        } catch (IOException e) {
        e.printStackTrace();
    }
```

Fonte: Elaborado pelo autor.

Foram feitas diversas tentativas utilizando a função *Wait* com valores de tempo diferentes, porem sem sucesso, onde ainda a programação da placa Arduino capturava e armazenava somente o primeiro valor corretamente e atribuía valor zero para os outros três parâmetros restantes.

Outra tentativa para sincronizar o envio e captura entre o App e a placa Arduino respectivamente, foi a alteração da parte do código da placa Arduino responsável por capturar e armazenar os valores recebidos pelo módulo *bluetooth* HC-06, assim como foi utilizada a função *Wait* no código do App, onde essa função aguarda um período em milissegundos para que o programa continue sua execução passando para próxima linha, desse mesmo modo foi utilizada a função *delay* na programação da placa Arduino, essa função assim como a função *Wait* trabalha com valores em milissegundos, nos testes de envio e captura foram utilizados os mesmos valores em milissegundos na função *Wait* do App e na função *delay* da placa Arduino. Vale

salientar que nessa fase de testes foi utilizada a ferramenta Monitor Serial da IDE Arduino, essa ferramenta foi de grande ajuda pois não foi necessário deslocar o triciclo para verificar se cada alteração feita nos códigos surgiram efeito, essa fase de testes pode ser melhor compreendida na Figura 22.

Figura 22 – Teste de sincronia código App e código placa Arduino

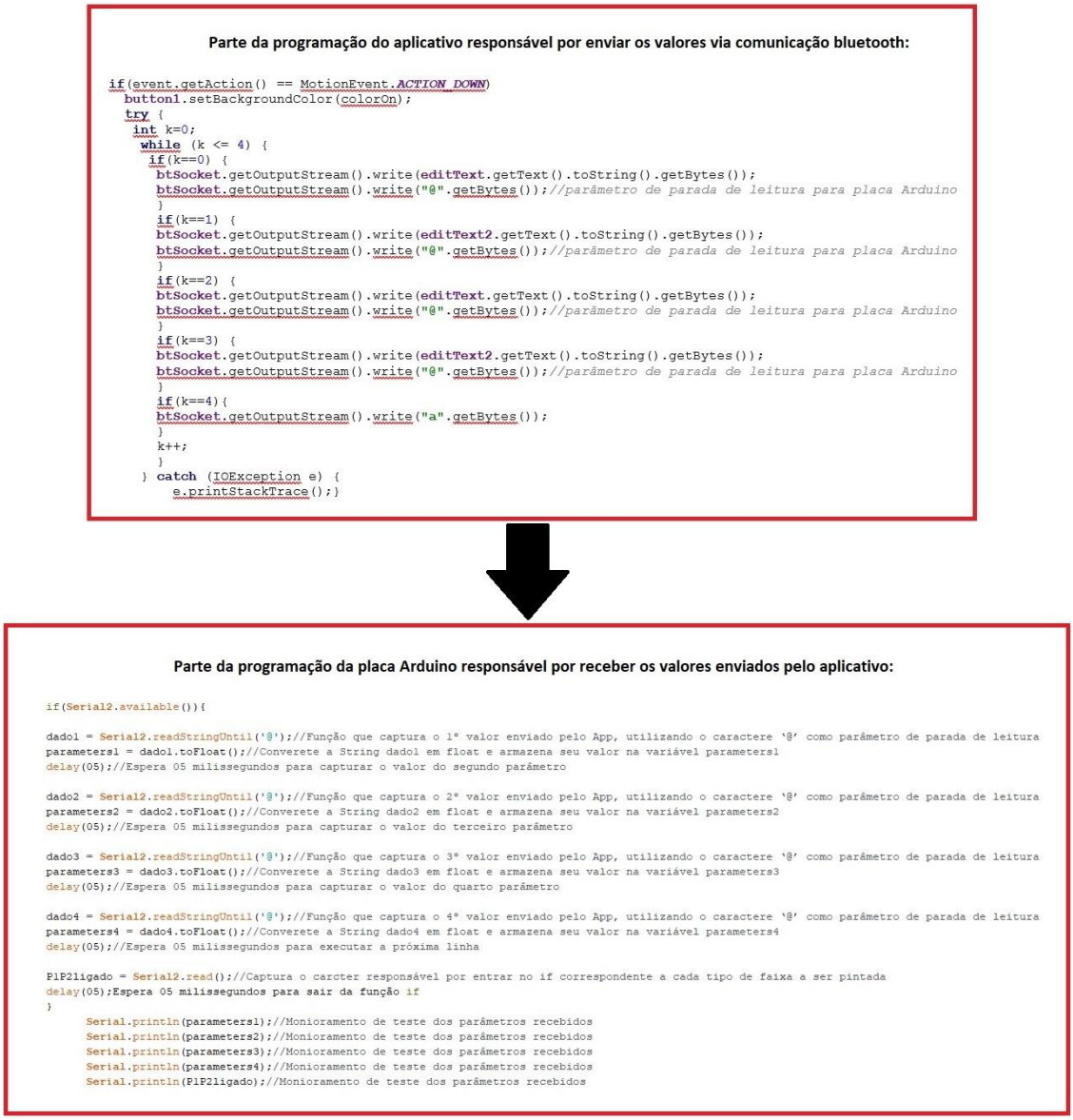
Fonte: Elaborado pelo autor.

Na figura acima pode-se observar a tentativa de sincronizar o envio e recebimento dos parâmetros de distância, porem mesmo atribuindo valores iguais em milissegundos para função *Wait* e função *delay* na programação do App e para placa Arduino respectivamente, o mesmo problema continuou ocorrendo, desse modo foi feita uma breve pesquisa para saber o que poderia estar ocorrendo. Uma das soluções encontras para evitar alterar a taxa de velocidade de comunicação de envio e recebimento o que poderia continuar gerando problemas de sincronização e sendo necessário implementar mais linhas de código, foi utilizar a função *Serial.readStringUntil( );* na programação da placa Arduino, essa função captura os valores recebidos pelo módulo *bluetooth* HC-06 da placa Arduino em formato *String*, porem seu diferencial está no parâmetro de parada de leitura, onde esse parâmetro

pode ser incluído entre parênteses no final da função, desse modo foi inserido o caractere “@” como parâmetro de parada, sendo escrita desse modo *Serial.readStringUntil(‘@’);* na subsecção 6.4 desse capítulo será abordado mais detalhadamente essa parte da programação da placa Arduino.

Utilizando o caractere “@” como parâmetro de parada de leitura na programação da placa Arduino, o código de programação do App também precisou ser alterado para enviar o caractere “@” logo após o envio de cada parâmetro de distância, também foi inserida a estrutura de repetição *While* a fim de garantir o envio dos quatros parâmetros de distância para cada variável correspondente no programa da placa Arduino, na Figura 23 isso pode ser melhor compreendido.

Figura 23 – Código do App e placa Arduino, código funcional.



Parte da programação do aplicativo responsável por enviar os valores via comunicação bluetooth:

```
if(event.getAction() == MotionEvent.ACTION_DOWN)
  button1.setBackgroundColor(colorOn);
  try {
   int k=0;
    while (k <= 4) {
     if(k==0) {
      btSocket.getOutputStream().write(editText.getText().toString().getBytes());
      btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada de leitura para placa Arduino
      }
      if(k==1) {
      btSocket.getOutputStream().write(editText2.getText().toString().getBytes());
      btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada de leitura para placa Arduino
      }
      if(k==2) {
      btSocket.getOutputStream().write(editText.getText().toString().getBytes());
      btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada de leitura para placa Arduino
      }
      if(k==3) {
      btSocket.getOutputStream().write(editText2.getText().toString().getBytes());
      btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada de leitura para placa Arduino
      }
      if(k==4){
      btSocket.getOutputStream().write("a".getBytes());
      }
      k++;
      }
    } catch (IOException e) {
       e.printStackTrace();}
```

Parte da programação da placa Arduino responsável por receber os valores enviados pelo aplicativo:

```
if(Serial2.available()){

dado1 = Serial2.readStringUntil('@');//Função que captura o 1° valor enviado pelo App, utilizando o caractere ‘@’ como parâmetro de parada de leitura
parameters1 = dado1.toFloat();//Converete a String dado1 em float e armazena seu valor na variável parameters1
delay(05);//Espera 05 milissegundos para capturar o valor do segundo parâmetro

dado2 = Serial2.readStringUntil('@');//Função que captura o 2° valor enviado pelo App, utilizando o caractere ‘@’ como parâmetro de parada de leitura
parameters2 = dado2.toFloat();//Converete a String dado2 em float e armazena seu valor na variável parameters2
delay(05);//Espera 05 milissegundos para capturar o valor do terceiro parâmetro

dado3 = Serial2.readStringUntil('@');//Função que captura o 3° valor enviado pelo App, utilizando o caractere ‘@’ como parâmetro de parada de leitura
parameters3 = dado3.toFloat();//Converete a String dado3 em float e armazena seu valor na variável parameters3
delay(05);//Espera 05 milissegundos para capturar o valor do quarto parâmetro

dado4 = Serial2.readStringUntil('@');//Função que captura o 4° valor enviado pelo App, utilizando o caractere ‘@’ como parâmetro de parada de leitura
parameters4 = dado4.toFloat();//Converete a String dado4 em float e armazena seu valor na variável parameters4
delay(05);//Espera 05 milissegundos para executar a próxima linha

P1P2ligado = Serial2.read();//Captura o carcter responsável por entrar no if correspondente a cada tipo de faixa a ser pintada
delay(05);Espera 05 milissegundos para sair da função if
}
      Serial.println(parameters1);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters2);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters3);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters4);//Monioramento de teste dos parâmetros recebidos
      Serial.println(P1P2ligado);//Monioramento de teste dos parâmetros recebidos
```

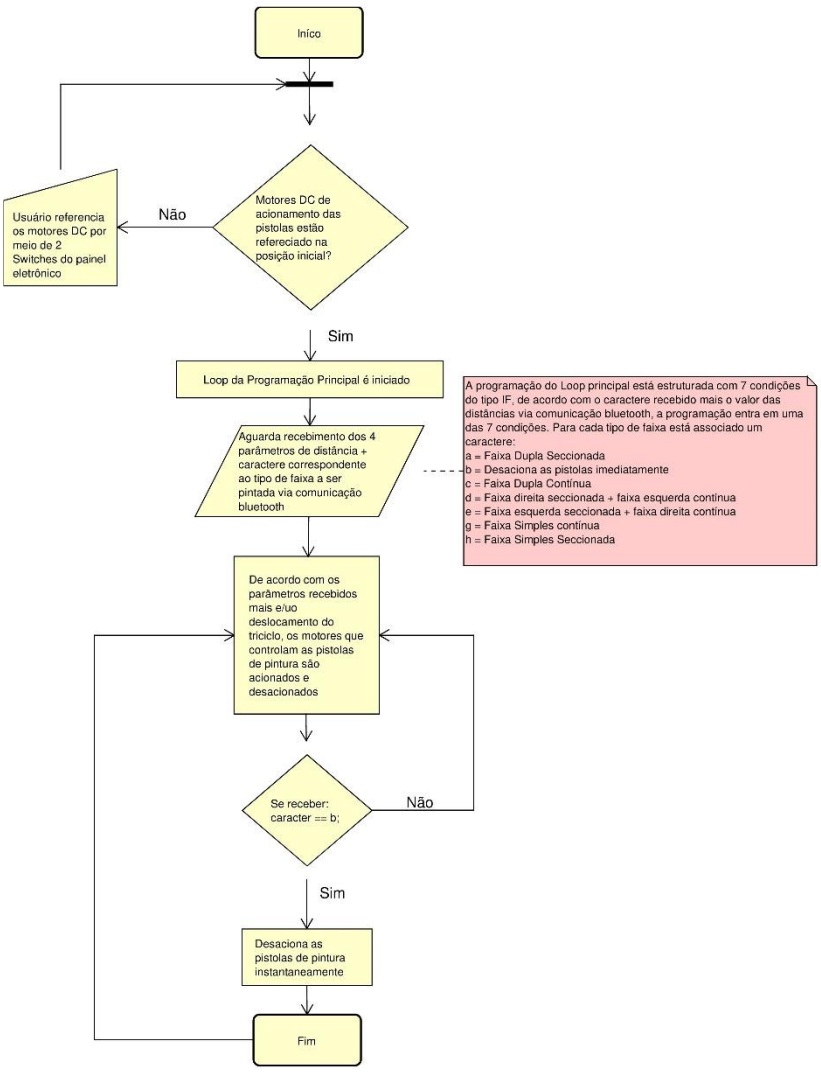
Fonte: Elaborado pelo autor.

6.4 DESENVOLVIMENTO DA PROGRAMAÇÃO PARA PLACA ARDUINO

O desenvolvimento da programação para placa Arduino foi mais simples e fácil do que o desenvolvimento da programação do aplicativo, pois a programação necessária para placa Arduino já estava desenvolvida para funcionar com o painel eletrônico do triciclo, sendo necessário somente substituir os parâmetros inseridos pelo usuário no painel eletrônico por parâmetros que serão recebidos via comunicação *bluetooth* pelo aplicativo.

O algoritmo de programação da placa Arduino foi estruturado de forma que cada ação referente ao acionamento e desacionamento das pistolas de pintura do triciclo fosse executada por meio de funções, abaixo a figura mostra o fluxograma utilizado para o desenvolvimento da programação.

Figura 24 – Fluxograma algoritmo de placa Arduino.



Fonte: Elaborado pelo autor.

Desse modo o código do loop principal ficou mais simples e fácil de alterar caso fosse necessário, tendo em sua estrutura o recebimento dos parâmetros de distância mais o caractere que corresponde ao tipo de faixa a ser pintada, a tabela abaixo ilustra essas combinações.

Tabela 2 – Combinação de parâmetros para cada tipo de faixa

Tipo de faixa a ser pintada	Caractere recebido	Parâmetro de distância (Pistola 1)	Parâmetros de distância (Pistola 2)
Faixa dupla seccionada	**a**	**> 0**	**> 0**
Desaciona as pistolas instantaneamente	**b**	**-**	**-**
Faixa Dupla contínua	**c**	**0.00**	**0.00**
Faixa Direita seccionada + esquerda contínua	**d**	**> 0**	**> 0**
Faixa esquerda seccionada + direita contínua	**e**	**> 0**	**> 0**
Faixa simples contínua	**g**	**0.00**	**0.00**
Faixa simples seccionada	**h**	**> 0**	**> 0**

Fonte: Elaborado pelo autor.

Em um primeiro momento foi feita a tentativa de capturar os dados recebido pelo módulo *bluetooth* HC-06 utilizando a função *Serial.readString( )* porem a mesma só capturava o primeiro conjunto dos cinco dados enviados pelo aplicativo. A função *Serial.readString( );* é herdada da classe *Stream*, essa classe não é chamada diretamente nas linhas de código da programação da placa Arduino, mas sim invocada quando usa uma função que depende dela, algumas das bibliotecas que dependem dessa classe são Serial, Wire, Ethernet e SD.

A lógica de programação do loop principal da placa Arduino está estruturada para receber cinco parâmetros sendo eles: distância da pistola de pintura Nº1 acionada; distância da pistola de pintura Nº2 acionada; distância da pistola de pintura Nº1 desacionada; distância da pistola de pintura Nº 2 desacionada; caractere para entrar na lógica *if* correspondente a cada tipo de faixa a ser pintada.

7 CONSIDERAÇÕES FINAIS

O Aplicativo desenvolvido apresentou-se funcional e de fácil iteração do usuário. Seu *design* minimalista foi crucial para atingir seu objetivo que foi a substituição do painel eletrônico por um *tablet* melhorando a iteração do usuário com o triciclo automatizado de pintura viária. Foram realizados inúmeros testes antes da entrega do aplicativo para o cliente, todos os tipos de pintura de faixas foram testadas simulando a pintura real de uma via. Vale salientar que o código fonte não foi entregue para o cliente como forma de entrega do produto final, a entrega foi do arquivo instalador do aplicativo em extensão “.apk”.

Uma consideração importante é em relação a atualização do aplicativo e melhorias contínuas, um aprimoramento que foi sinalizado tanto pelo cliente quanto pelo usuário do triciclo, foi a inserção de um indicativo de que a comunicação *bluetooth* entre o aplicativo e a placa Arduino está de fato acontecendo, essa funcionalidade poderá facilmente ser implementada por meio de um Led virtual.

Outro aprimoramento seria a inclusão de uma funcionalidade em que o usuário inseriria a distância a ser pintada pelo triciclo. Essa funcionalidade eliminaria a necessidade do operador apertar o botão parar pintura, uma vez que o mesmo irá inserir a distância em metros que o triciclo irá pintar, assim que o triciclo percorrer a quantidade em metros inseridas pelo usuário o mesmo irá parar de pintar quando essa distância for atingida. Por meio de um Encoder incremental já instalado no triciclo e funcionando com o aplicativo desenvolvido, a programação na placa Arduino teria uma rotina de programação que saberia quanto o triciclo se deslocou, desacionando as pistolas de pintura assim que fosse atingida a distância que foi inserida pelo usuário.

De modo geral tanto o cliente quanto o operador do triciclo ficaram satisfeitos com o desempenho do aplicativo, o aplicativo se mostrou funcional e estável durante os testes de pintura.

O desenvolvimento desse aplicativo foi enriquecedor, pois abrangeu diversas disciplinas do curso, como algoritmos e lógica de programação, linguagem de programação, interação humano computador, engenharia de *software* e dentre outras que foram tão importantes para que esse projeto fosse desenvolvido com sucesso.

REFERÊNCIAS BIBLIOGRÁFICAS

Arduino CC Reference **Communication**, Disponível em: <https://www.arduino.cc/reference/pt/language/functions/communication/serial/readstring/>, Acesso em 27 de nov. 2019.

CARVALHO, Luiz Raimundo Moreira de; AMORIM, Helio Salim de. Observando as marés atmosféricas: uma aplicação da placa Arduino com sensores de pressão barométrica e temperatura. **Rev. Bras. Ensino Fís.**, São Paulo, v. 36, n. 3, p. 1-7, set. 2014. Disponível em: <http://www.sbfisica.org.br/rbef/pdf/363501.pdf>. Acesso em: 25 de nov. 2019, às 12h34min.

DEITEL, Harvey M.; DEITEL, Paul. J. **Java**: Como Programar. Trad. Edson Furmankiewicz. 8ª Edição. São Paulo: Pearson Education, 2010.

Eu sou Dev, As 10 Heurísticas de Nielsen, Disponível em: < https://eusoudev.com.br/heuristicas-de-nielsen/>, Acesso em: 26 nov. 2019.

EVANS, Martin; NOBLE, Joshua; HOCHENBAUM, Jordan. **Arduino em ação**. Trad. Camila Paduan. São Paulo: Novatec, 2013. 424p

JR. Sergio L. S.; FARINELLI. Felipe A. **Dmótica:** Automação residencial e casas inteligentes com Arduino e ESP8266. São Paulo: Érica, 2019.

MONK, Simon**. Projetos com arduino e android**: use seu smartphone ou tablet para controlar o arduino. Porto Alegre: Bookman, 2014.

Nicolosi, Denys E. C.; **Microcontrolador 8051 Detalhado.** 8ª Ed. São Paulo: Érica, 2007.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software**: fundamentos, métodos e padrões. 3ª ed. Rio Janeiro, Editora: LTC, 2009.

PEREIRA, L. C. O.; SILVA, M. L. da. **Android para desenvolvedores**. 2. ed. Rio de Janeiro: Brasport, 2012.

SOMMERVILLE, I. **Engenharia de Software**. 9ª Edição: Pearson Prentice Hall. São Paulo, 2011.

The Android Developers, **App Manifest Overview**, Disponível em: <https://developer.android.com/guide/topics/manifest/manifest-intro?hl=eng>, Acesso em 27 nov. 2019.

The Android Developers, **Bluetooth**, Disponível em: <https://developer.android.com/guide/topics/connectivity/bluetooth#java>, Acesso em: 18 nov. 2019.

The Android Developers, **BluetoothSocket**, Disponível em: <https://developer.android.com/reference/android/bluetooth/BluetoothSocket>, Acesso em: 18 nov. 2019.

The Android Developers, **Layouts**, Disponível em: <https://developer.android.com/guide/topics/ui/declaring-layout>, Acesso em 27 nov. 2019.

The Android Developers, **OutputStream**, Disponível em: <https://developer.android.com/reference/java/io/OutputStream.html>, Acesso em 27 nov. 2019.

ANEXOS

Programação XML do arquivo “AndroidManifest.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:dist="http://schemas.android.com/apk/distribution"
package="com.example.stonehammer4">

<dist:module dist:instant="true" />

<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission
android:name="com.google.android.things.permission.USE_PERIPHERAL_IO" />

<application
 android:allowBackup="true"
    android:icon="@mipmap/stonehammer"
    android:label="StoneHammer"
    android:roundIcon="@mipmap/stonehammer"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
<activity android:name=".simplescontinua"
android:screenOrientation="landscape"/>
    <activity android:name=".simplesseccionada"
android:screenOrientation="landscape"/>
    <activity android:name=".direitasecciesquercont"
android:screenOrientation="landscape"/>
    <activity android:name=".esquerdracejadireitacont"
android:screenOrientation="landscape"/>
    <activity android:name=".Duplacontinua"
android:screenOrientation="landscape"/>
    <activity android:name=".Tracejadadupla"
android:screenOrientation="landscape"/>
    <activity android:name=".HomeActivity"
android:screenOrientation="landscape"/>
    <activity android:name=".MainActivity"
android:screenOrientation="landscape">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>
```

Programação XML das telas do Aplicativo.

XML referente a tela de boas vindas ao usuário “activity_main.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="683dp"
        android:layout_height="587dp"
        android:layout_centerInParent="true"
        android:rotation="0"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/stonehammer2" />
</RelativeLayout>
```

XML referente a tela de principal do aplicativo “activity_home.xml”

```
<?xml version="1.0" encoding="utf-8"?>

<HorizontalScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:background="@android:color/darker_gray">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray">


        <RelativeLayout
            android:layout_width="975dp"
            android:layout_height="600dp"

            android:background="@android:color/background_light"
            tools:context=".HomeActivity">

            <ImageButton
                android:id="@+id/imageButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="40dp"
                android:layout_marginLeft="40dp"
                android:layout_marginTop="100dp"
                android:layout_marginEnd="437dp"
```

```
        android:layout_marginRight="437dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/faixaduplatrac2" />

    <ImageButton
        android:id="@+id/imageButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="180dp"
        android:layout_marginLeft="180dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="900dp"
        android:layout_marginRight="900dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imageButton"

        app:srcCompat="@drawable/faixaduplacont2" />

    <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="320dp"
        android:layout_marginLeft="320dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="437dp"
        android:layout_marginRight="437dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imageButton"
        app:srcCompat="@drawable/faixaesqtradireitcont2" />

    <ImageButton
        android:id="@+id/imageButton4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="470dp"
        android:layout_marginLeft="470dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="800dp"
        android:layout_marginRight="900dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imageButton"
        app:srcCompat="@drawable/faixaesqcontdiretrac2" />

    <ImageButton
        android:id="@+id/imageButton5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="620dp"
        android:layout_marginLeft="620dp"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="900dp"
        android:layout_marginRight="900dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imageButton"
        app:srcCompat="@drawable/faixasimplessec" />

    <ImageButton
        android:id="@+id/imageButton6"
        android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:layout_marginStart="770dp"
                android:layout_marginLeft="770dp"
                android:layout_marginTop="100dp"
                android:layout_marginEnd="900dp"
                android:layout_marginRight="900dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toEndOf="@+id/imageButton"
                app:srcCompat="@drawable/faixasimplescont" />

            <TextView
                android:id="@+id/textView5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="350dp"
                android:layout_marginLeft="350dp"
                android:layout_marginTop="30dp"
                android:layout_marginEnd="900dp"
                android:layout_marginRight="900dp"
                android:text="Escolha o tipo de faixa abaixo:"
                android:textColor="@android:color/background_dark"
                android:textSize="24sp"
                app:layout_constraintEnd_toEndOf="parent" />

        </RelativeLayout>
    </ScrollView>
</HorizontalScrollView>
```

XML referente a tela faixa dupla seccionada “activity_tracejadadupla.xmll”

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        android:fillViewport="true">
        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="30dp"
            android:text="Faixa Dupla Seccionada:"
            android:textColor="@android:color/background_light"
            android:textSize="24sp" />
        <TextView
            android:id="@+id/textView"
            android:layout_width="330dp"
            android:layout_height="60dp"
            android:layout_marginLeft="140dp"
            android:layout_marginTop="120dp"
            android:layout_marginBottom="500dp"
            android:text="Distância das Pistolas acionadas:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />
```

```
        <TextView
            android:id="@+id/textView2"
            android:layout_width="330dp"
            android:layout_height="60dp"
            android:layout_marginLeft="140dp"
            android:layout_marginTop="230dp"
            android:layout_marginBottom="400dp"
            android:text="Distância das Pistolas desacionadas:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editText"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="520dp"
            android:layout_marginTop="115dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3"/>
        <EditText
            android:id="@+id/editText2"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="520dp"
            android:layout_marginTop="225dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3"/>
        <Button
            android:id="@+id/button1"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="250dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Confirmar Distâncias" />
        <Button
            android:id="@+id/button2"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="550dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Parar Pintura!" />

    </RelativeLayout>
</ScrollView>
```

XML referente a tela faixa dupla contínua “activity_duplacontinua.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray"
    tools:context=".Duplacontinua">
    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:text="Faixa Dupla Contínua:"
        android:textColor="@android:color/background_light"
        android:textSize="24sp" />
    <TextView
    android:id="@+id/textView7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="90dp"
    android:text="Atenção!!!"
    android:textColor="@android:color/holo_red_dark"
    android:textSize="30sp" />
    <TextView
        android:id="@+id/textView8"
        android:layout_width="360dp"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:text="Somente acione a chave abaixo se o triciclo de
pintura estiver posicionado e preparado para pintar."
        android:textAlignment="center"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="22dp" />
    <Switch
        android:id="@+id/switch1"
        android:layout_width="325dp"
        android:layout_height="70dp"
        android:layout_marginLeft="320dp"
        android:layout_marginTop="300dp"
        android:layout_marginBottom="500dp"
        android:background="@color/colorPrimary"
        android:text="Acionamento das Pistolas de Pintura"
        android:textColor="@android:color/background_dark"
        android:textSize="16sp" />
</RelativeLayout>
</ScrollView>
```

XML referente a tela faixa esquerda seccionada mais faixa direita contínua “activity_esquerdracejadireitacont.xml”

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        android:fillViewport="true">

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="30dp"
            android:text="Faixa esquerda seccionada + faixa direita
contínua:"
            android:textColor="@android:color/background_light"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/textView"
            android:layout_width="330dp"
            android:layout_height="60dp"
            android:layout_marginLeft="160dp"
            android:layout_marginTop="120dp"
            android:layout_marginBottom="500dp"
            android:text="Distância da Pistola esquerda acionada:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="380dp"
            android:layout_height="60dp"
            android:layout_marginLeft="160dp"
            android:layout_marginTop="230dp"
            android:layout_marginBottom="400dp"
            android:text="Distância da Pistola esquerda desacionada:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editText"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="540dp"
            android:layout_marginTop="115dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3" />
```

```
        <EditText
            android:id="@+id/editText2"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="540dp"
            android:layout_marginTop="225dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3" />


        <Button
            android:id="@+id/button1"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="250dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Confirmar Distâncias" />

        <Button
            android:id="@+id/button2"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="550dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Parar Pintura!" />
    </RelativeLayout>
```

XML referente a tela faixa direita seccionada mais faixa esquerda contínua “activity_direitasecciesquercont.xml”

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        android:fillViewport="true">

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="30dp"
            android:text="Faixa direita seccionada + faixa esquerda
contínua:"
```

```
        android:textColor="@android:color/background_light"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="330dp"
        android:layout_height="60dp"
        android:layout_marginLeft="160dp"
        android:layout_marginTop="120dp"
        android:layout_marginBottom="500dp"
        android:text="Distância da Pistola direita acionada:"
        android:textColor="@android:color/background_dark"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="380dp"
        android:layout_height="60dp"
        android:layout_marginLeft="160dp"
        android:layout_marginTop="230dp"
        android:layout_marginBottom="400dp"
        android:text="Distância da Pistola direita desacionada:"
        android:textColor="@android:color/background_dark"
        android:textSize="18sp" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="100dp"
        android:layout_height="45dp"
        android:layout_marginLeft="520dp"
        android:layout_marginTop="115dp"
        android:layout_marginBottom="400dp"
        android:ems="10"
        android:inputType="numberDecimal"
        android:maxLength="3"/>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="100dp"
        android:layout_height="45dp"
        android:layout_marginLeft="520dp"
        android:layout_marginTop="225dp"
        android:layout_marginBottom="400dp"
        android:ems="10"
        android:inputType="numberDecimal"
        android:maxLength="3"/>


    <Button
        android:id="@+id/button1"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="250dp"
        android:layout_marginTop="320dp"
        android:layout_marginBottom="500dp"
        android:background="@color/colorPrimary"
        android:text="Confirmar Distâncias" />

    <Button
        android:id="@+id/button2"
        android:layout_width="190dp"
```

```
            android:layout_height="wrap_content"
            android:layout_marginLeft="550dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Parar Pintura!" />

    </RelativeLayout>
</ScrollView>
```

XML referente a tela faixa simples contínua “activity_simplescontinua.xml”

```
<?xml version="1.0" encoding="utf-8"?>

    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray">
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray"
    tools:context=".simplescontinua">


    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:text="Faixa Simples Contínua:"
        android:textColor="@android:color/background_light"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="90dp"
        android:text="Atenção!!!"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/textView8"
        android:layout_width="360dp"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:text="Somente acione a chave abaixo se o triciclo de
pintura estiver posicionado e preparado para pintar."
        android:textAlignment="center"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="22dp" />
```

```
    <Switch
        android:id="@+id/switch1"
        android:layout_width="325dp"
        android:layout_height="70dp"
        android:layout_marginLeft="320dp"
        android:layout_marginTop="300dp"
        android:layout_marginBottom="500dp"
        android:background="@color/colorPrimary"
        android:text="Acionamento da Pistola de Pintura"
        android:textColor="@android:color/background_dark"
        android:textSize="16sp" />

</RelativeLayout>
</ScrollView>
```

XML referente a tela faixa simples seccionada “activity_simplesseccionada.xml”

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        android:fillViewport="true">

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="30dp"
            android:text="Faixa simples seccionada:"
            android:textColor="@android:color/background_light"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/textView"
            android:layout_width="330dp"
            android:layout_height="60dp"
            android:layout_marginLeft="160dp"
            android:layout_marginTop="120dp"
            android:layout_marginBottom="500dp"
            android:text="Distância da Pistola acionada:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="380dp"
            android:layout_height="60dp"
            android:layout_marginLeft="160dp"
            android:layout_marginTop="230dp"
            android:layout_marginBottom="400dp"
```

```
            android:text="Distância da Pistola desacionada:"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editText"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="540dp"
            android:layout_marginTop="115dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3" />

        <EditText
            android:id="@+id/editText2"
            android:layout_width="100dp"
            android:layout_height="45dp"
            android:layout_marginLeft="540dp"
            android:layout_marginTop="225dp"
            android:layout_marginBottom="400dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:maxLength="3" />


        <Button
            android:id="@+id/button1"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="250dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Confirmar Distâncias" />

        <Button
            android:id="@+id/button2"
            android:layout_width="190dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="550dp"
            android:layout_marginTop="320dp"
            android:layout_marginBottom="500dp"
            android:background="@color/colorPrimary"
            android:text="Parar Pintura!" />

    </RelativeLayout>
</ScrollView>
```

Código fonte parcial em linguagem Java das telas do Aplicativo.

“MainActivity.java”

```
package com.example.stonehammer4;

        import androidx.appcompat.app.AppCompatActivity;

        import android.os.Bundle;
        import android.content.Intent;
        import android.os.Handler;


public class MainActivity extends AppCompatActivity {
    private static int SPLASH_TIME_OUT = 3000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        new Handler().postDelayed(new Runnable(){
            @Override
            public void run(){
                Intent homeIntent = new Intent(MainActivity.this,
HomeActivity.class);
                startActivity(homeIntent);
                finish();

            }
        },SPLASH_TIME_OUT);
    }
}
```

Código das ações da tela principal “HomeActivity.java”

```
package com.example.stonehammer4;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;

public class HomeActivity extends AppCompatActivity {

    private ImageButton button;
    private ImageButton button2;
    private ImageButton button3;
    private ImageButton button4;
    private ImageButton button5;
    private ImageButton button6;

    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
```

```
        button = (ImageButton) findViewById(R.id.imageButton);
        button2 = (ImageButton) findViewById(R.id.imageButton2);
        button3 = (ImageButton) findViewById(R.id.imageButton3);
        button4 = (ImageButton) findViewById(R.id.imageButton4);
        button5 = (ImageButton) findViewById(R.id.imageButton5);
        button6 = (ImageButton) findViewById(R.id.imageButton6);

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Tracejadadupla();
            }
        });
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Duplacontinua();
            }
        });
        button3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Esquerdracejadireitacont();
            }
        });
        button4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Direitasecciesquercont();
            }
        });
        button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Simplesseccionada();
            }
        });
        button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Simplescontinua();
            }
        });

    }
    public void Tracejadadupla(){ //Abre tela para pintura tracejada dupla
        Intent intent = new Intent(this, Tracejadadupla.class);
        startActivity(intent);
    }

    public void Duplacontinua(){ //Abre tela de pintura faixa dupla
contínua
        Intent intent = new Intent(this, Duplacontinua.class);
        startActivity(intent);
    }

    public void Esquerdracejadireitacont(){ //Abre tela Esquerda tracejada
+ Direita Contínua
        Intent intent = new Intent(this, esquerdracejadireitacont.class);
        startActivity(intent);
```

```
    }

    public void Direitasecciesquercont(){ //Abre tela Esquerda tracejada +
Direita Contínua
        Intent intent = new Intent(this, direitasecciesquercont.class);
        startActivity(intent);
    }

    public void Simplesseccionada(){ //Abre tela Esquerda tracejada +
Direita Contínua
        Intent intent = new Intent(this, simplesseccionada.class);
        startActivity(intent);
    }

    public void Simplescontinua(){ //Abre tela Esquerda tracejada + Direita
Contínua
        Intent intent = new Intent(this, simplescontinua.class);
        startActivity(intent);
    }
}
```

Código fonte das ações da tela referente a faixa dupla seccionada

“Tracejadadupla.java”

```
package com.example.stonehammer4;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import java.io.IOException;
import java.util.Set;
import java.util.UUID;
import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


//FAIXA DUPLA SECCIONADA

public class Tracejadadupla extends AppCompatActivity implements
OnClickListener {

    Button i1;
    TextView t1;


    EditText editText;
    EditText editText2;

    Button button1;
```

```
    Button button2;

    String address = null , name=null;


    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    Set<BluetoothDevice> pairedDevices;

    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");

    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tracejadadupla);
        try {setw();} catch (Exception e) {}

        editText = (EditText) findViewById(R.id.editText);
        editText2 = (EditText) findViewById(R.id.editText2);
        button1 = (Button)findViewById(R.id.button1);
        button2 = (Button)findViewById(R.id.button2);


        button1.setOnTouchListener(new View.OnTouchListener()//Ação que
ocorrerá quando botão for acionado
        {   @Override
        public boolean onTouch(View v, MotionEvent event){
 int colorOn = 0xFF20B2AA;
 int coloroff = 0xFFFFFF00;
 if(event.getAction() == MotionEvent.ACTION_DOWN)
 button1.setBackgroundColor(colorOn);
 try {
   int k=0;
     while (k <= 4) {
         if(k==0) {

btSocket.getOutputStream().write(editText.getText().toString().getBytes());

btSocket.getOutputStream().write("@".toString().getBytes());//parâmetro de
parada para o Arduino
                        }
if(k==1) {

btSocket.getOutputStream().write(editText2.getText().toString().getBytes())
;

btSocket.getOutputStream().write("@".toString().getBytes());//parâmetro de
parada para o Arduino
                        }
if(k==2) {

btSocket.getOutputStream().write(editText.getText().toString().getBytes());

btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada para
o Arduino
                        }
if(k==3) {

btSocket.getOutputStream().write(editText2.getText().toString().getBytes())
```

```
;

btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada para
o Arduino
                        }
                        if(k==4){
                          btSocket.getOutputStream().write("a".getBytes());

                        }
                        k++;
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),e.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            if(event.getAction() == MotionEvent.ACTION_UP)
            button1.setBackgroundColor(coloroff);
            return true;}
        });

        button2.setOnTouchListener(new View.OnTouchListener()
        {   @Override
        public boolean onTouch(View v, MotionEvent event){
            int colorOn = 0xFF20B2AA;
            int coloroff = 0xFFFFFF00;
            if(event.getAction() == MotionEvent.ACTION_DOWN)

                button2.setBackgroundColor(colorOn);

            try {
                int j=0;
                while (j <= 4) {
                    if(j==0) {
                        editText.setText("");

btSocket.getOutputStream().write("0.00".getBytes());
btSocket.getOutputStream().write("@".toString().getBytes());//parâmetro de
parada para o Arduino
                    }
                    if(j==1) {
                        editText2.setText("");

btSocket.getOutputStream().write("0.00".getBytes());
btSocket.getOutputStream().write("@".toString().getBytes());//parâmetro de
parada para o Arduino
                    }
                    if(j==2) {
                        editText.setText("");

btSocket.getOutputStream().write("0.00".getBytes());
btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada para
o Arduino
                    }
                    if(j==3) {
                        editText2.setText("");

btSocket.getOutputStream().write("0.00".getBytes());
```

```
btSocket.getOutputStream().write("@".getBytes());//parâmetro de parada para
o Arduino
                    }
                    if(j==4){
                        btSocket.getOutputStream().write("b".getBytes());
                    }
                    j++;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            catch (Exception e)
            {
                Toast.makeText(getApplicationContext(),e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
            if(event.getAction() == MotionEvent.ACTION_UP)
                button2.setBackgroundColor(coloroff);
            return true;}
        });

        }//Fechamento loop programação principal
```

Código fonte parcial da placa Arduino:

```
// Loop principal da programação
void loop() {

float parameters1;
float parameters2;
float parameters3;
float parameters4;

String dado1;
String dado2;
String dado3;
String dado4;

char P1P2ligado;

if(Serial2.available()){

   dado1 =   Serial2.readStringUntil('@');
   parameters1 = dado1.toFloat();
   delay(05);

   dado2 = Serial2.readStringUntil('@');
   parameters2 = dado2.toFloat();
   delay(05);

   dado3 = Serial2.readStringUntil('@');
   parameters3 = dado3.toFloat();
   delay(05);
```

```
    dado4 = Serial2.readStringUntil('@');
    parameters4 = dado4.toFloat();
    delay(05);

    P1P2ligado = Serial2.read();

    delay(05);
}



      Serial.println(parameters1);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters2);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters3);//Monioramento de teste dos parâmetros recebidos
      Serial.println(parameters4);//Monioramento de teste dos parâmetros recebidos
      Serial.println(P1P2ligado);//Monioramento de teste dos parâmetros recebidos


  setInputFlags();
  resolveInputFlags();
  //Conversão do pulso em metros
  dist = pulse * 0.54024/1000.0; //Calculo da Distancia em metros 0,4502
  P1_ligado = digitalRead(39);
  P2_ligado = digitalRead(41);

//nova Lógica
if(P1P2ligado == 'b')//Desliga as Pistolas
{
fecha_P1_P2();
//fecha_P1();
//fecha_P2();
}


//Faixa Contínua Dupla
if(P1P2ligado == 'c')//Recebe caracter do App via Bluetooch
   {
     if ((parameters1 == 0.00 and parameters2 == 0.00) and (parameters3 == 0.00
and parameters4==0.00))
       {
       //Modo Continuo Distancias iguas a 0
       //Abre a Pistola P1 e P2 quando a chave estiver ligada

       abre_P1_P2();

       }
   }

//Faixa Dupla tracejda
```

```
      if ((P1P2ligado == 'a') and ((parameters1 > 0.01 and parameters2 > 0.01) and
(parameters3 > 0.01 and parameters4 > 0.01)))
        {

           if ( (parameters1 == 0.00 and parameters2 == 0.00) and (parameters3 ==
0.00 and parameters4==0.00))// if para cair no laço de pulso do

            {

             }
      // Se sensor estiver aberto e a distancia for menor
    else if (((digitalRead(40) == 0) and ((parameters2 - dist) < 0.01))  and
((digitalRead(34) == 0) and ((parameters4 - dist) < 0.01))){
       abre_P1_P2();

       pulse=0;
       // Se sensor estiver fechado e a distancia for menor
    }else if ( ((digitalRead(38) == 0) and ((parameters1 - dist) < 0.01)) and
((digitalRead(36) == 0) and ((parameters3 - dist) < 0.01))){
        fecha_P1_P2();
        pulse=0;

    }

      }



//Faixa esquerda contínua + Faixa direita tracejada
 if( (P1P2ligado == 'd') and ((parameters1 == 0.00 and parameters2 == 0.00) and
(parameters3 > 0.01 and parameters4 > 0.01))) {
  //Fecha a Pistola P1 quando a chave estiver desligada



   // Se o botao estiver ligado
    if (parameters3 == 0.00 and parameters4==0.00)// if para cair no laço de pulso do
encoder
    {

    }
    else if ((digitalRead(34) == 0) and ((parameters4 - dist) < 0.01)){
       abre_P1();
       abre_P2();
       pulse=0;
       // Se sensor estiver fechado e a distancia for menor
    }else if ((digitalRead(36) == 0) and ((parameters3 - dist) < 0.01)){
        abre_P1();
        fecha_P2();
        pulse=0;
```

```
    }
   }

//Faixa esquerda Seccionada + Faixa Direita Contínua
 if( (P1P2ligado == 'e') and  ((parameters1 > 0.01 and parameters2 > 0.01) and
(parameters3 == 0.00 and parameters4 == 0.00) )) {


    if (parameters1 == 0.00 and parameters2 == 0.00)// if para cair no laço de pulso
do encoder
    {

    }
    else if ((digitalRead(40) == 0) and ((parameters2 - dist) < 0.01))// Se sensor
estiver aberto e a distancia for menor
    {
       abre_P2();
       abre_P1();
       pulse=0;
       // Se sensor estiver fechado e a distancia for menor
    }else if ((digitalRead(38) == 0) and ((parameters1 - dist) < 0.01)){
        abre_P2();
        fecha_P1();
        pulse=0;

    }
   }



//Faixa Simples Direita Contínua
  if(P1P2ligado == 'g'){
    if ((parameters1 == 0.00 and parameters2 == 0.00) and (parameters3 == 0.00 and
parameters4==0.00))
       {

  abre_P1();
  pulse=0;
       }
  }

//Faixa Esquerda Simples Seccionada
  if(P1P2ligado == 'h'){

  if((parameters1 > 0.01 and parameters2 > 0.01) and (parameters3 == 0.00 and
parameters4==0.00)){
    if (parameters1 == 0.00 and parameters2==0.00)// if para cair no laço de pulso do
encoder
    {
```

```
    }
    else if ((digitalRead(40) == 0) and ((parameters2 - dist) < 0.01))// Se sensor
estiver aberto e a distancia for menor
    {
       abre_P1();
       pulse=0;
       // Se sensor estiver fechado e a distancia for menor
    }else if ((digitalRead(38) == 0) and ((parameters1 - dist) < 0.01)){
        fecha_P1();
        pulse=0;

    }
  }
  }


}//fecha loop void programação principal
```