Montain Wolves Airsoft Bomb

Project

Uma imagem com emblema, símbolo, logótipo, Marca registada

Descrição gerada automaticamente

# Introdução

No âmbito da unidade curricular, é necessário desenvolver e entregar dois projetos eletrônicos que atendam a diferentes requisitos mínimos. Para cumprir essas exigências, o tema escolhido foi um projeto que permite expansões e atende a todos os critérios propostos.

Como praticante de Airsoft e membro da equipe Montain Wolves, decidi criar um projeto pessoal que pudesse ser integrado na disciplina. Este projeto consiste em uma Fake Bomb para ser utilizada em ambientes de jogo de Airsoft.

O projeto possui múltiplas possibilidades de atualização, tais como:

Operar como um sistema simples de efeito de bomba,

Integrar-se a um protocolo de comunicação para interação com outros dispositivos durante o jogo,

Conectar-se a um servidor, permitindo, através de sensores de geolocalização, o monitoramento em tempo real de sua posição, criando assim uma espécie de "streaming" do jogo.

O dispositivo, em sua versão final, poderá apresentar as seguintes funcionalidades:

Configuração via Bluetooth,

Configuração manual,

Geolocalização do dispositivo,

Integração em uma rede LoRa.

Para a unidade curricular é necessário entregar dois projetos eletrônicos em que os mesmos têm de apresentar requisitos mínimos diferentes, contudo o tema escolhido é um projeto que permite o upgrade e que abrange todos os requisitos pedidos.

Sendo eu um jogador de Airsoft na equipa dos Montain Wolves decidi criar um projeto pessoal em que poderia enquadrar na unidade curricular. Projeto este que é uma Fake Bomb para ser utilizada em ambiente de jogo de Airsoft.

Este projeto contém muitas vertentes de atualização como:

- Utilizar o seu simples sistema de efeito bomba

- Utilizar a mesma para enquadrar num protocolo de comunicação e comunicar com outros dispositivos em jogo

- Enquadrar a mesma num servidor e assim através dos seus sensores de glocalização saber sempre em tempo real onde a mesma se encontra e assim criar uma espécie de stream do jogo em questão.

O dispositivo na sua entrega final poderá contar com as seguintes funcionalidades:

- Configuração através de Bluetooth,

- Configuração através de manual,

- Geolocalização do dispositivo,

- Inserção numa rede Lora.

Índice

[Introdução 3](#_Toc182941366)

[Tema do projeto 8](#_Toc182941367)

[Linguagem de programação 9](#_Toc182941368)

[C++ 9](#_Toc182941369)

[Java 9](#_Toc182941370)

[Microcontroladores 10](#_Toc182941371)

[Softwares utilizados 11](#_Toc182941372)

[Arduino IDE 11](#_Toc182941373)

[Android Studio 12](#_Toc182941374)

[Git 12](#_Toc182941375)

[Componentes Utilizados no Projeto 14](#_Toc182941376)

[Esp32 Dev Kit V4 14](#_Toc182941377)

[Protoboard 15](#_Toc182941378)

[Placa PCB perfurada 15](#_Toc182941379)

[Display OLED 16](#_Toc182941380)

[Display Liquid Crystal I2C 16](#_Toc182941381)

[Módulo GPS 16](#_Toc182941382)

[Teclado 4x4 16](#_Toc182941383)

[Buzzer 17](#_Toc182941384)

[Atomizador piezoelétrico 18](#_Toc182941385)

[Leds WS2812b 18](#_Toc182941386)

[Esquemas e layouts 18](#_Toc182941387)

[Esquema elétrico 19](#_Toc182941388)

[Layout para PCB 20](#_Toc182941389)

[Modelo 3D 22](#_Toc182941390)

[Tampa 22](#_Toc182941391)

[Original 22](#_Toc182941392)

[Modelado 23](#_Toc182941393)

[Corpo 24](#_Toc182941394)

[Original 24](#_Toc182941395)

[Modelado 25](#_Toc182941396)

[Aplicação Android 25](#_Toc182941397)

[Estrutura e design 26](#_Toc182941398)

[Código Java 30](#_Toc182941399)

[Código Arduino 36](#_Toc182941400)

[Montain\_Wolves\_Bomb 36](#_Toc182941401)

[Leds\_WS2812b 36](#_Toc182941402)

[Gps 36](#_Toc182941403)

[Bomb 36](#_Toc182941404)

[Class 36](#_Toc182941405)

[Implementação 37](#_Toc182941406)

[Display\_liquidCrystal 40](#_Toc182941407)

[Display\_oled 49](#_Toc182941408)

[Paletas 49](#_Toc182941409)

[Bluetooth 53](#_Toc182941410)

[Keypad 59](#_Toc182941411)

[ManuallyConfig 59](#_Toc182941412)

[Setup 59](#_Toc182941413)

[Beep 60](#_Toc182941414)

[Core0 63](#_Toc182941415)

[Core1 63](#_Toc182941416)

[Bibliografia 63](#_Toc182941417)

[Figura 1 - Protejo Inicial 7](#_Toc182936415)

[Figura 2 - Arduino Nano 9](#_Toc182936416)

[Figura 3 - Arduino Uno R4 10](#_Toc182936417)

[Figura 4 - Arduino Mega 10](#_Toc182936418)

# Tema do projeto

Para a construção física do projeto, foi utilizado como base o modelo descrito a seguir. Os links presentes na bibliografia apresentam todos os componentes do projeto original, incluindo peças 3D e código. O código do projeto original é amplamente utilizado em diversos projetos semelhantes no universo do Airsoft. No entanto, a ideia central deste trabalho foi reutilizar a estrutura física com o propósito de adaptá-la às nossas necessidades específicas. Assim, a estrutura serviu apenas como um ponto de partida e precisou ser redesenhada. Para esse processo, foi utilizado o software de modelagem 3D SolidWorks.

Para a construção do projeto fisicamente foi utilizado o modelo abaixo como base. Os links na bibliografia iram apresentar todos os componentes do projeto original desde peças 3D como código. O projeto original o seu código é um código muito utilizado em amplos projetos do mesmo género no Airsoft, contudo a ideia era reutilizar a estrutura física para assim a mesma ser alterada consoante a nossa necessidade, logo o objetivo é apenas servir de base, logo as mesmas terão de ser redesenhadas, para isso será utilizado o software de modelagem 3D SolidWorks.



Figura 1 - Protejo Inicial

# Linguagem de programação

No desenvolvimento do projeto, foram empregadas diferentes linguagens de programação, de acordo com as necessidades específicas de cada ambiente. O C++ foi utilizado para a programação do microcontrolador, garantindo eficiência e controle detalhado dos recursos embarcados. Já a linguagem Java foi aplicada no desenvolvimento do aplicativo Android, proporcionando uma interface amigável e uma integração robusta com o sistema móvel.

## C++

C++ é uma linguagem de programação de propósito geral conhecida por ser poderosa e versátil, derivada do C. Ela oferece suporte à programação orientada a objetos, programação genérica e programação procedural. Seu principal diferencial é a combinação de eficiência de baixo nível (próxima ao hardware) com recursos de alto nível, o que a torna ideal para sistemas de software complexos, como sistemas operacionais, drivers de dispositivos, aplicações em tempo real, jogos e aplicações de alta performance.

C++ oferece suporte a encapsulamento, herança e polimorfismo, permitindo modularidade e reuso de código, além de controle detalhado de recursos, como memória, por meio de ponteiros e gerenciamento manual. Sua flexibilidade também se estende ao desenvolvimento de sistemas embarcados, como o ESP32, onde é amplamente utilizado para criar aplicações eficientes e com recursos avançados, tirando proveito das capacidades de conectividade Wi-Fi e Bluetooth do microcontrolador.

## Java

Java é uma linguagem de programação orientada a objetos, conhecida por sua portabilidade, robustez e facilidade de uso. Desenvolvida para ser independente de plataforma através da Máquina Virtual Java (JVM), ela é amplamente utilizada no desenvolvimento de aplicativos, especialmente para dispositivos Android.

No contexto do projeto, Java está sendo utilizado para o desenvolvimento de um aplicativo Android. Essa escolha permite a criação de interfaces interativas e a integração de funcionalidades específicas para o controle e monitoramento do dispositivo desenvolvido. A flexibilidade da linguagem, aliada a um vasto ecossistema de bibliotecas e ferramentas, facilita o desenvolvimento de aplicações móveis robustas e escaláveis, garantindo uma experiência de usuário fluida.

# Microcontroladores

Os microcontroladores desempenham o papel de "cérebro" neste projeto, pois são responsáveis, por meio da programação em C++, pelo controle da lógica do sistema. Eles permitem a leitura de sinais de entrada e a gestão das saídas, viabilizando o funcionamento desejado do dispositivo. Atualmente, há diversos tipos de microcontroladores disponíveis no mercado, dos quais alguns dos principais serão mencionados mais adiante neste trabalho.

Os microcontroladores serão o cérebro neste projeto pois é o mesmo que é responsável de através da linguagem de programação C++ controlar a parte lógica e assim podermos ler sinais de entrada e gerir as nossas saídas para assim obtermos o pretendido.

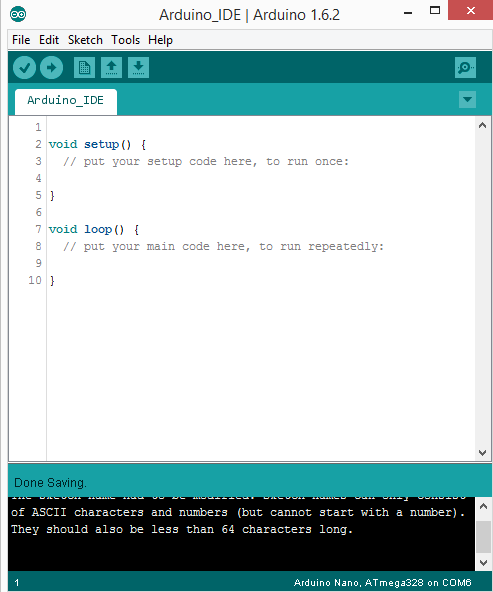
Existem muitos tipos de microcontroladores no mercado ao qual irei mencionar alguns principais mais abaixo.

# Softwares utilizados

## Arduino IDE

O Arduino IDE (Integrated Development Environment) é um ambiente de desenvolvimento integrado utilizado para programar placas Arduino e outros microcontroladores compatíveis. Ele oferece uma interface simples e acessível, projetada para facilitar a programação e a prototipagem de dispositivos eletrônicos, mesmo para iniciantes. O Arduino IDE suporta a linguagem de programação baseada em C/C++, tornando o desenvolvimento prático e intuitivo por meio de funções predefinidas e bibliotecas amplamente utilizadas pela comunidade.

Com o Arduino IDE, é possível escrever, compilar e carregar código para microcontroladores através de uma conexão USB. A ferramenta oferece recursos como detecção automática de portas seriais, um monitor serial para depuração em tempo real, além de um vasto repositório de exemplos e bibliotecas que ajudam a acelerar o desenvolvimento de projetos. Sua versatilidade e compatibilidade com uma ampla gama de placas, incluindo módulos baseados no ESP32, tornam o Arduino IDE uma escolha popular para prototipagem de sistemas embarcados, projetos IoT e automação em geral.



## Android Studio

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android. Baseado no IntelliJ IDEA, o Android Studio foi projetado pela Google para oferecer uma experiência de desenvolvimento poderosa e flexível, fornecendo todas as ferramentas necessárias para criar, depurar e testar aplicativos móveis para dispositivos Android.

O Android Studio oferece um editor de código inteligente com sugestões e autocompletar, suporte integrado ao desenvolvimento com a linguagem Java, Kotlin e C++, além de ferramentas gráficas de design para construir interfaces de usuário de forma visual. Possui também emuladores para testar aplicativos em uma ampla gama de dispositivos e configurações, bem como um sistema de compilação baseado no Gradle, que facilita a automação de tarefas e gerência de dependências.

O ambiente ainda traz ferramentas para otimização de desempenho, depuração de código, testes automatizados, suporte a desenvolvimento de aplicações para diferentes tamanhos de tela e dispositivos, além de integração com o Firebase para fornecer funcionalidades como autenticação, armazenamento de dados em tempo real, entre outros. Tudo isso faz do Android Studio a solução mais robusta e completa para o desenvolvimento de aplicativos Android.

## Git

Git é um sistema de controle de versão distribuído amplamente utilizado no desenvolvimento de software. Ele permite que desenvolvedores rastreiem alterações em seu código-fonte, colaborem de maneira eficiente em equipes e revertam para versões anteriores do projeto quando necessário. Criado por Linus Torvalds em 2005, o Git é conhecido por sua rapidez, confiabilidade e flexibilidade.

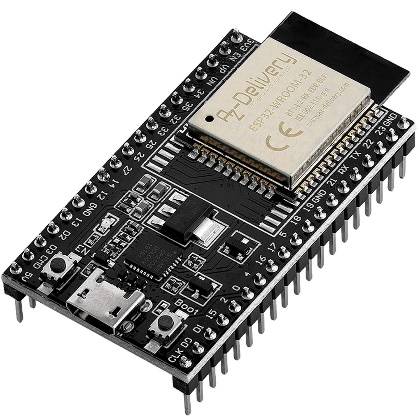
Com Git, as mudanças são registradas em um repositório que armazena todo o histórico de modificações, possibilitando o controle detalhado das versões do código. Cada desenvolvedor possui uma cópia local do repositório, permitindo o trabalho offline e a realização de experimentos sem impactar a versão principal do projeto. Ele utiliza conceitos como branches (ramificações) para que múltiplos desenvolvedores possam trabalhar em diferentes funcionalidades ou correções de forma isolada antes de mesclá-las ao projeto principal.

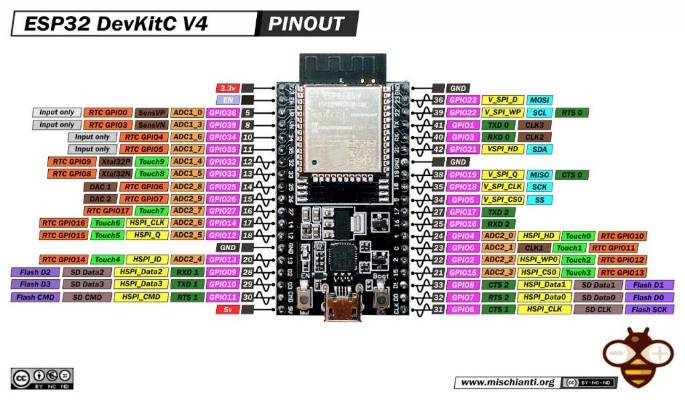
Além disso, Git facilita a colaboração por meio de plataformas de hospedagem como GitHub, GitLab e Bitbucket, que oferecem recursos adicionais como gerenciamento de issues, revisões de código, e integração contínua (CI/CD). Essas funcionalidades tornam Git uma ferramenta essencial para equipes de desenvolvimento modernas, ajudando a gerenciar o ciclo de vida dos projetos de forma eficiente e segura.



# Componentes Utilizados no Projeto

### Esp32 Dev Kit V4

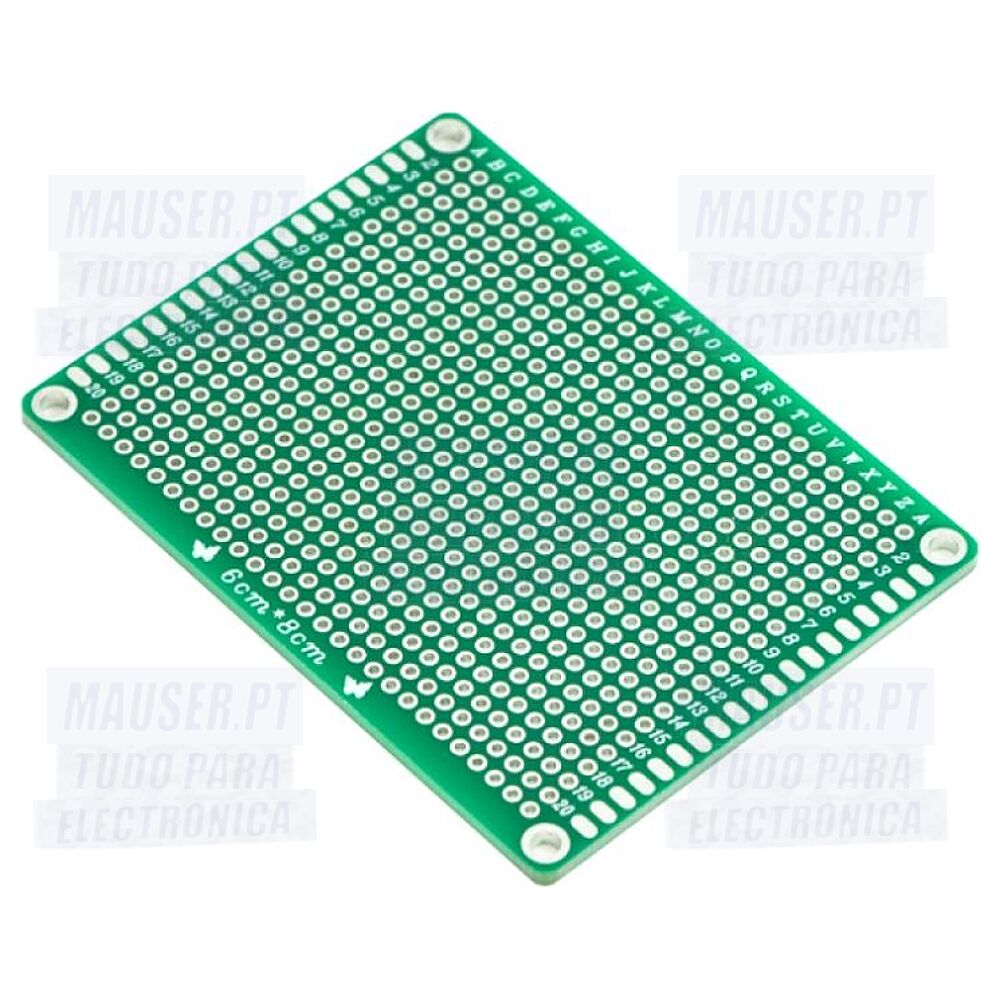




### Protoboard



### Placa PCB perfurada



### Display OLED



### Display Liquid Crystal I2C



### Módulo GPS



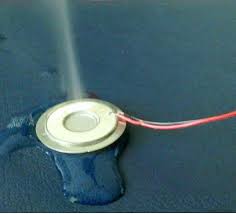
### Teclado 4x4



### Buzzer



### Atomizador piezoelétrico





### Leds WS2812b

# Esquemas e layouts

## Esquema elétrico

Uma imagem com texto, diagrama, Esquema, file

Descrição gerada automaticamente

## Layout para PCB

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Uma imagem com texto, eletrónica, Engenharia eletrónica, circuito

Descrição gerada automaticamente

Uma imagem com captura de ecrã, Retângulo, Azul elétrico, Azul majorelle

Descrição gerada automaticamente

# Modelo 3D

## Tampa

### Original

Uma imagem com dados

Descrição gerada automaticamente com confiança baixa

### Modelado

Uma imagem com Retângulo, entrada, design

Descrição gerada automaticamente

Uma imagem com círculo, Retângulo, design

Descrição gerada automaticamente

## Corpo

### Original

### Modelado

# Aplicação Android

## Estrutura e design

Uma imagem com texto, captura de ecrã, póster, logótipo

Descrição gerada automaticamente

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 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:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <ImageView  
 android:id="@+id/teamLogo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/montain\_wolves\_logo"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <ImageButton  
 android:id="@+id/play\_pause\_intro"  
 android:layout\_width="24dp"  
 android:layout\_height="24dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginEnd="5dp"  
 android:background="@null"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_pause" />  
  
 <TextView  
 android:id="@+id/editText\_inOut"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginStart="90dp"  
 android:text="@string/in\_out\_layout"  
 android:textColor="@color/green"  
 android:textSize="30dp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
 <Switch  
 android:id="@+id/sw\_gps"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="50dp"  
 android:background="@color/white"  
 android:text="@string/switch\_gps\_off"  
 android:textSize="25dp"  
 android:textStyle="bold"  
 android:textColor="@color/blue"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <Switch  
 android:id="@+id/sw\_sound"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="36dp"  
 android:background="@color/white"  
 android:text="@string/switch\_sound\_off"  
 android:textSize="25dp"  
 android:textStyle="bold"  
 android:textColor="@color/blue"  
 app:layout\_constraintBottom\_toBottomOf="@+id/sw\_gps"  
 app:layout\_constraintStart\_toEndOf="@+id/sw\_gps"  
 app:layout\_constraintTop\_toTopOf="@+id/sw\_gps"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <Switch  
 android:id="@+id/sw\_leds"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:text="@string/switch\_leds\_off"  
 android:background="@color/white"  
 android:textSize="25dp"  
 android:textColor="@color/blue"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/sw\_gps"  
 app:layout\_constraintStart\_toStartOf="@+id/sw\_gps"  
 app:layout\_constraintTop\_toBottomOf="@+id/sw\_gps" />  
  
 <Switch  
 android:id="@+id/sw\_smoke"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="25dp"  
 android:textColor="@color/blue"  
 android:background="@color/white"  
 android:text="@string/switch\_smoke\_off"  
 android:textSize="25dp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/sw\_sound"  
 app:layout\_constraintStart\_toStartOf="@+id/sw\_sound"  
 app:layout\_constraintTop\_toBottomOf="@+id/sw\_sound" />  
  
 <TextView  
 android:id="@+id/editText\_dataGame"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="128dp"  
 android:layout\_marginTop="20dp"  
 android:text="@string/game\_data\_layout"  
 android:textColor="@color/green"  
 android:textSize="30dp"  
 android:textStyle="bold"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/sw\_smoke" />  
  
 <EditText  
 android:id="@+id/edit\_text\_time\_game"  
 android:layout\_width="150dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="30dp"  
 android:background="@color/white"  
 android:ems="10"  
 android:hint="@string/time\_of\_game"  
 android:inputType="number"  
 android:textAlignment="center"  
 android:textSize="30dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_dataGame" />  
  
 <EditText  
 android:id="@+id/edit\_text\_players"  
 android:layout\_width="150dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="30dp"  
 android:background="@color/white"  
 android:ems="10"  
 android:hint="@string/num\_of\_players"  
 android:textSize="30dp"  
 android:inputType="number"  
 android:textAlignment="center"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/edit\_text\_time\_game" />  
  
 <EditText  
 android:id="@+id/edit\_text\_bomb\_Code"  
 android:layout\_width="200dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="30dp"  
 android:background="@color/white"  
 android:ems="10"  
 android:hint="@string/bomb\_code"  
 android:textSize="30dp"  
 android:inputType="number"  
 android:textAlignment="center"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/edit\_text\_players" />  
  
 <Button  
 android:id="@+id/btn\_send\_config"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="140dp"  
 android:layout\_marginTop="40dp"  
 android:background="@color/red"  
 android:text="@string/send\_config"  
 android:textSize="30dp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/edit\_text\_bomb\_Code" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

## Código Java

package com.example.montainwolvesapp;  
  
import static android.content.ContentValues.*TAG*;  
  
import android.bluetooth.BluetoothAdapter;  
import android.bluetooth.BluetoothDevice;  
import android.bluetooth.BluetoothSocket;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.content.pm.PackageManager;  
import android.media.MediaPlayer;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.Switch;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.util.UUID;  
  
public class MainActivity extends AppCompatActivity {  
  
 Switch gps, leds, sound, smoke;  
 Button sendConfig;  
 ImageButton playPause;  
 ImageView teamLogo;  
 MediaPlayer mediaPlayer = new MediaPlayer();  
 EditText timeGame, numPlayers, bombCode;  
 boolean musicSound;  
  
 private BluetoothAdapter bluetoothAdapter;  
 private BluetoothSocket bluetoothSocket;  
 private BluetoothDevice bluetoothDevice;  
 private OutputStream outputStream;  
  
 private final String DEVICE\_ADDRESS = "08:A6:F7:20:B3:1E"; // Coloque o endereço MAC do seu módulo Bluetooth  
 private final UUID MY\_UUID = UUID.*fromString*("00001101-0000-1000-8000-00805F9B34FB");  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 bluetoothAdapter = BluetoothAdapter.*getDefaultAdapter*();  
 bluetoothDevice = bluetoothAdapter.getRemoteDevice(DEVICE\_ADDRESS);  
  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*intro*);  
 mediaPlayer.start();  
 playPause = (ImageButton) findViewById(R.id.*play\_pause\_intro*);  
 gps = (Switch) findViewById(R.id.*sw\_gps*);  
 leds = (Switch) findViewById(R.id.*sw\_leds*);  
 smoke = (Switch) findViewById(R.id.*sw\_smoke*);  
 sound = (Switch) findViewById(R.id.*sw\_sound*);  
 sendConfig = (Button) findViewById(R.id.*btn\_send\_config*);  
 timeGame = (EditText) findViewById(R.id.*edit\_text\_time\_game*);  
 numPlayers = (EditText) findViewById(R.id.*edit\_text\_players*);  
 bombCode = (EditText) findViewById(R.id.*edit\_text\_bomb\_Code*);  
  
 timeGame.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message = "TIME" + timeGame.getText().toString() + "\n";  
 sendData(message);  
 }  
 });  
  
 numPlayers.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message = "TEAM" + numPlayers.getText().toString() + "\n";  
 sendData(message);  
 }  
 });  
  
 bombCode.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (bombCode.length() == Integer.*parseInt*(numPlayers.getText().toString())){  
 String message = "CODE" + bombCode.getText().toString() + "\n";  
 sendData(message);  
 }else{  
 Toast.*makeText*(MainActivity.this,  
 "Bomb Code Size let be the same of Players Number",  
 Toast.*LENGTH\_LONG*).show();  
 }  
 }  
 });  
  
  
 playPause.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 introSound();  
 }  
 });  
  
 gps.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (gps.isChecked()) {  
 String message = "gpsON\n";  
 sendData(message);  
 } else {  
 String message = "gpsOFF\n";  
 sendData(message);  
 }  
 }  
 });  
  
 leds.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message;  
 if (leds.isChecked()) {  
 leds.setText(R.string.*switch\_leds\_on*);  
 message = "ledsON\n";  
 } else {  
 leds.setText(R.string.*switch\_leds\_off*);  
 message = "ledsOFF\n";  
  
 }  
 sendData(message);  
 }  
 });  
  
 smoke.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message;  
 if (smoke.isChecked()) {  
 smoke.setText(R.string.*switch\_smoke\_on*);  
 message = "smokeON\n";  
 } else {  
 smoke.setText(R.string.*switch\_smoke\_off*);  
 message = "smokeOFF\n";  
 }  
 sendData(message);  
 }  
 });  
  
 sound.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message;  
 if (sound.isChecked()) {  
 sound.setText(R.string.*switch\_sound\_on*);  
 message = "soundON\n";  
 } else {  
 sound.setText(R.string.*switch\_sound\_off*);  
 message = "soundOFF\n";  
 }  
 sendData(message);  
 }  
 });  
  
 sendConfig.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String message = "sendConfig";  
 sendData(message);  
 }  
 });  
  
 connectBluetooth();  
  
 }  
  
 public void introSound(){  
 if(musicSound){  
 musicSound = false;  
 playPause.setImageResource(R.drawable.*ic\_play*);  
 }else{  
 musicSound = true;  
 playPause.setImageResource(R.drawable.*ic\_pause*);  
 }  
  
 if(musicSound) {  
 mediaPlayer.start();  
 }else{  
 mediaPlayer.pause();  
 }  
 }  
  
 private void connectBluetooth() {  
 try {  
 bluetoothSocket = bluetoothDevice.createRfcommSocketToServiceRecord(MY\_UUID);  
 bluetoothSocket.connect();  
 outputStream = bluetoothSocket.getOutputStream();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void sendData(String message) {  
 try {  
 outputStream.write(message.getBytes());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 try {  
 bluetoothSocket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

# Código Arduino

## Montain\_Wolves\_Bomb

## Leds\_WS2812b

## Gps

## Bomb

### Class

#pragma once

enum BombStatus {

  initialize,

  configuration,

  readyToArm,

  armed,

  disarm,

  explode

};

class Bomb{

  public:

    int time;

    int tries;

    int tryArming;

    int players;

    int gameTime;

    int speedLight = 20;

    const int speedSound[4] = {250, 500, 1000, 2000};

    BombStatus bombStatus = initialize;

    double latZone;

    double longZone;

    double maxDistance;

    boolean codeDiscovered;

    String code;

    boolean gps;

    boolean sound;

    boolean leds;

    boolean smoke;

    String checkGPS();

    String checkLeds();

    String checkSound();

    String checkSmoke();

    Bomb(int, String, int, int, int);

    boolean checkCode(String code);

    boolean isExplode();

    boolean finishGame(unsigned long);

    boolean isTimeOut(unsigned long);

    int getSize();

    boolean isValidZone();

  private:

};

### Implementação

#include"Bomb.h"

extern double getDistance(double, double);

Bomb::Bomb(int time, String code, int tries, int tryArming, int gameTime){

  this->time = time;

  this->tries = tries;

  this->tryArming = tryArming;

  this->code = code;

  this->players = 0;

  this->gameTime = gameTime \* 60000;

  this->gps = false;

  this->sound = true;

  this->leds = true;

  this->smoke = false;

  this->latZone = 40.769726;

  this->longZone = -8.027784;

  this->maxDistance = 10.0;

  this->codeDiscovered = false;

};

String Bomb::checkGPS(){

  if (this->gps){

    return "Enable";

  }else {return "Disable";}

}

String Bomb::checkLeds() {

  if (this->leds){

    return "Enable";

  }else {return "Disable";}

}

String Bomb::checkSound() {

  if (this->sound){

    return "Enable";

  }else {return "Disable";}

}

String Bomb::checkSmoke() {

  if (this->smoke){

    return "Enable";

  }else {return "Disable";}

}

boolean Bomb::checkCode(String trycode){

  if (trycode == this->code){

    return true;

  }else{

    return false;

  }

};

boolean Bomb::isExplode(){

  if (this->tries < 0){

    return true;

  }else{

    return false;

  }

};

boolean Bomb::finishGame(unsigned long timeInit) {

  if (millis() - timeInit >= this->gameTime){

    return true;

  }

  return false;

}

boolean Bomb::isTimeOut(unsigned long timeInit) {

  if (millis() - timeInit >= this->time){

    return true;

  }

  return false;

}

int Bomb::getSize(){

  int size = this->code.length();

  return size;

};

boolean Bomb::isValidZone(){

  double distance = getDistance(this->latZone, this->longZone);

  if(distance <= bomb.maxDistance){

    return true;

  }else{

    return false;

  }

};

## Display\_liquidCrystal

extern boolean setupFinish;

extern String code;

void printBluetoothConfig();

void printPlayers();

void printTimeGame();

void printTimeBomb();

void printBombCode();

void printSmoke(String);

void printSound(String);

void printLeds(String);

void printGPS(String);

void WaitSat();

void printCoordinate();

void printConfirmConfig();

void bombIsReady();

void availableZone();

void invalidZone();

void insertCode();

void teamIntro();

void printDigit(String);

void printCode(String);

void correctCode();

void wrongCode();

void tryAgain();

void bombExploded();

void bombExplodedToArming();

void youWin();

void bombArmed();

void printClock(int, int);

void restart();

void printBarChar();

uint8\_t barChar[8] = {

  0b11111,

  0b11111,

  0b11111,

  0b11111,

  0b11111,

  0b11111,

  0b11111,

  0b11111,

  };

void printBluetoothChoice() {

  lcd.setCursor(3,0);

  lcd.print("Bluetooth");

  lcd.setCursor(0,1);

  lcd.print("A - Yes   D - No");

  }

void printBluetoothConfig(){

  lcd.setCursor(3,0);

  lcd.print("Bluetooth");

  lcd.setCursor(1,1);

  lcd.print("Configuration!");

}

void printPlayers() {

  lcd.setCursor(0,0);

  lcd.print("Players Number");

}

void printTimeGame() {

  lcd.setCursor(0,0);

  lcd.print("Time Game");

}

void printTimeBomb() {

  lcd.setCursor(0,0);

  lcd.print("Time Bomb");

}

void printBombCode() {

  lcd.setCursor(0,0);

  lcd.print("Bomb Code");

}

void printGPS(String status){

  lcd.setCursor(0,0);

  lcd.print("GPS?");

  lcd.setCursor(9,0);

  lcd.print(status);

  lcd.setCursor(0,1);

  lcd.print("A - Yes   B - No");

}

void WaitSat() {

  lcd.setCursor(0,0);

  lcd.print("\*\*\*\*\*Wait\*\*\*\*\*\*\*");

  lcd.setCursor(0, 1);

  lcd.print("For more Sats!!\*");

}

void printCoordinate() {

  lcd.setCursor(0, 0);

  lcd.print("Lat: "); lcd.print(gps.location.lat(), 6);

  lcd.setCursor(0, 1);

  lcd.print("Long: "); lcd.print(gps.location.lng(), 6);

}

void setNewCoordinates() {

  lcd.setCursor(2, 0);

  lcd.print("coordenates");

  lcd.setCursor(3, 1);

  lcd.print("Configured");

}

void printLeds(String status){

  lcd.setCursor(0,0);

  lcd.print("Leds?");

   lcd.setCursor(9,0);

  lcd.print(status);

  lcd.setCursor(0,1);

  lcd.print("A - Yes   B - No");

}

void printSound(String status){

  lcd.setCursor(0,0);

  lcd.print("Sound?");

   lcd.setCursor(9,0);

  lcd.print(status);

  lcd.setCursor(0,1);

  lcd.print("A - Yes   B - No");

}

void printSmoke(String status){

  lcd.setCursor(0,0);

  lcd.print("Smoke?");

   lcd.setCursor(9,0);

  lcd.print(status);

  lcd.setCursor(0,1);

  lcd.print("A - Yes   B - No");

}

void printConfirmConfig(){

  lcd.setCursor(4,0);

  lcd.print("Confirm?");

  lcd.setCursor(0,1);

  lcd.print("D - Yes   C - No");

}

void bombIsReady(){

  lcd.setCursor(6,0);

  lcd.print("Bomb");

  lcd.setCursor(5, 1);

  lcd.print("READY!");

}

void availableZone(){

  lcd.backlight();

  lcd.setCursor(6, 0);

  lcd.print("Zone");

  lcd.setCursor(3, 1);

  lcd.print("Autorized");

}

void invalidZone(){

  lcd.backlightOff();

  lcd.setCursor(6, 0);

  lcd.print("Zone");

  lcd.setCursor(4, 1);

  lcd.print("Invalid");

}

void teamIntro(){

  lcd.setCursor(1,0);

  lcd.print("Montain Wolves");

  lcd.setCursor(4,1);

  lcd.print(F("Press D!!"));

}

void insertCode(){

  lcd.setCursor(0,0);

  lcd.print("Insert Bomb Code");

}

void bombArmed(){

  lcd.setCursor(0,0);

  lcd.print("Armed");

}

void printClock(int timeMin, int timeSec){

  lcd.setCursor(11, 0); // Or setting the cursor in the desired position.

  if (timeMin < 10){

    lcd.print("0");

  }

  lcd.print(timeMin);

  lcd.print(":");

  if (timeSec < 10){

    lcd.print("0");

  }

  lcd.print(timeSec);

}

void printDigit(String digit){

  lcd.setCursor(0, 1);

  lcd.print(digit);

}

void correctCode(){

  lcd.setCursor(1,0);

  lcd.print("Correct Code!");

}

void wrongCode(){

  lcd.setCursor(2,0);

  lcd.print("Wrong Code!");

}

void printCode(String code){

  lcd.setCursor(0, 1);

  lcd.print(code);

}

void tryAgain(){

  lcd.setCursor(3,1);

  lcd.print("Try Again!!");

}

void bombExploded(){

  lcd.setCursor(1,0);

  lcd.print("Bomb Exploded");

}

void bombExplodedToArming(){

  lcd.setCursor(1,0);

  lcd.print("Bomb Exploded");

  lcd.setCursor(3, 1);

  lcd.print("To Arming");

}

void youWin(){

  lcd.setCursor(4,0);

  lcd.print("YOU WIN!");

}

void restart() {

  lcd.setCursor(5,0);

  lcd.print("Restart");

  lcd.setCursor(4,1);

  lcd.print("Press D");

}

void printBarChar(){

  lcd.createChar(0, barChar);

  lcd.write(0);

}

## Paletas

#define NUM\_PALETTES 11

DEFINE\_GRADIENT\_PALETTE( browngreen\_gp ) {

  0,    6,  255,    0,     //green

  71,    0,  255,  153,     //bluegreen

  122,  200,  200,  200,     //gray

  181,  110,   61,    6,     //brown

  255,    6,  255,    0      //green

};

CRGBPalette16 brownGreenPalette = browngreen\_gp;

DEFINE\_GRADIENT\_PALETTE( heatmap\_gp ) {

  0,      0,    0,    0,  // black

  128,  255,    0,    0,  // red

  200,  255,  255,    0,  // bright yellow

  255,  255,  255,  255,  // full white

};

CRGBPalette16 heatPalette = heatmap\_gp;

DEFINE\_GRADIENT\_PALETTE( greenblue\_gp ) {

  0,   0,  255, 245,

  46,  0,  21,  255,

  179, 12, 250, 0,

  255, 0,  255, 245

};

CRGBPalette16 greenbluePalette = greenblue\_gp;

DEFINE\_GRADIENT\_PALETTE( Sunset\_Real\_gp ) {

  0, 120,  0,  0,

  22, 179, 22,  0,

  51, 255, 104,  0,

  85, 167, 22, 18,

  135, 100,  0, 103,

  198,  16,  0, 130,

  255,   0,  0, 160

};

CRGBPalette16 sunsetPalette = Sunset\_Real\_gp;

CRGBPalette16 purplePalette = CRGBPalette16 (

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::Magenta,

                                CRGB::Magenta,

                                CRGB::Linen,

                                CRGB::Linen,

                                CRGB::Magenta,

                                CRGB::Magenta,

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::DarkViolet,

                                CRGB::Linen,

                                CRGB::Linen

                              );

DEFINE\_GRADIENT\_PALETTE( fireandice\_gp ) {

  0,  80,  2,  1,

  51, 206, 15,  1,

  101, 242, 34,  1,

  153,  16, 67, 128,

  204,   2, 21, 69,

  255,   1,  2,  4

};

CRGBPalette16 fireandicePalette = fireandice\_gp;

DEFINE\_GRADIENT\_PALETTE( bhw2\_turq\_gp ) {

  0,   1, 33, 95,

  38,   1, 107, 37,

  76,  42, 255, 45,

  127, 255, 255, 45,

  178,  42, 255, 45,

  216,   1, 107, 37,

  255,   1, 33, 95

};

CRGBPalette16 turqPalette = bhw2\_turq\_gp;

DEFINE\_GRADIENT\_PALETTE( autumnrose\_gp ) {

  0,  71,  3,  1,

  45, 128,  5,  2,

  84, 186, 11,  3,

  127, 215, 27,  8,

  153, 224, 69, 13,

  188, 229, 84,  6,

  226, 242, 135, 17,

  255, 247, 161, 79

};

CRGBPalette16 autumnrosePalette = autumnrose\_gp;

DEFINE\_GRADIENT\_PALETTE( bhw1\_06\_gp ) {

  0, 184,  1, 128,

  160,   1, 193, 182,

  219, 153, 227, 190,

  255, 255, 255, 255

};

CRGBPalette16 bhw1\_06Palette = bhw1\_06\_gp;

DEFINE\_GRADIENT\_PALETTE( bhw2\_xmas\_gp ) {

  0,   0, 12,  0,

  40,   0, 55,  0,

  66,   1, 117,  2,

  77,   1, 84,  1,

  81,   0, 55,  0,

  119,   0, 12,  0,

  153,  42,  0,  0,

  181, 121,  0,  0,

  204, 255, 12,  8,

  224, 121,  0,  0,

  244,  42,  0,  0,

  255,  42,  0,  0

};

CRGBPalette16 xmasPalette = bhw2\_xmas\_gp;

DEFINE\_GRADIENT\_PALETTE( bhw1\_justducky\_gp ) {

  0,  47, 28,  2,

  76, 229, 73,  1,

  163, 255, 255,  0,

  255, 229, 73,  1

};

CRGBPalette16 justduckyPalette = bhw1\_justducky\_gp;

## Bluetooth

extern boolean bluetoothConfigured;

extern void bombIsReady();

extern void beepingTimes(int, int);

void bluetoothConfig(BluetoothSerial);

void bluetoothConfig(BluetoothSerial BT){

  String message = "";

  while(!bluetoothConfigured){

    if (BT.available()) {

      char inComingChar = BT.read();

    if (inComingChar != '\n'){

      message += String (inComingChar);

    }

    else{

      message = "";

    }

    Serial.write(inComingChar);

  if (message.substring(0, 4) == "TEAM")

  {

    beepingTimes(5, 50);

    bomb.players = message.substring(4).toInt();

    Serial.print("\nPlayers: "); Serial.println(bomb.players);

    lcd.clear();

    lcd.setCursor(4,0);

    lcd.print("Players");

    lcd.setCursor(7, 1);

    lcd.print(String(bomb.players));

  }

  if (message.substring(0,4) == "TIME")

  {

    beepingTimes(5, 50);

    bomb.time = message.substring(4).toInt() \* 60000;

    Serial.print("\nGame Time: "); Serial.println(bomb.time);

    lcd.clear();

    lcd.setCursor(3,0);

    lcd.print("Bomb Time");

    lcd.setCursor(2, 1);

    lcd.print(String(bomb.time / 60000));

    lcd.setCursor(5, 1);

    lcd.print("Minutes");

  }

  if (message.substring(0,4) == "CODE")

  {

    beepingTimes(5, 50);

    bomb.code = message.substring(4);

    Serial.print("Bomb Code: "); Serial.println(bomb.code);

    lcd.clear();

    lcd.setCursor(3,0);

    lcd.print("Bomb Code");

    lcd.setCursor(4, 1);

    lcd.print("Defined");

  }

  if (message == "gpsON")

  {

    beepingTimes(5, 50);

    bomb.gps = true;

    Serial.println("\nGPS is ON!");

    lcd.clear();

    lcd.setCursor(6,0);

    lcd.print("GPS");

    lcd.setCursor(7, 1);

    lcd.print("ON");

  }

  if (message == "gpsOFF")

  {

    beepingTimes(5, 50);

    bomb.gps = false;

    Serial.println("\nGPS is OFF!");

    lcd.clear();

    lcd.setCursor(6,0);

    lcd.print("GPS");

    lcd.setCursor(6, 1);

    lcd.print("OFF");

  }

  if (message == "ledsON")

  {

    beepingTimes(5, 50);

    bomb.leds = true;

    Serial.println("\nLeds are ON!");

    lcd.clear();

    lcd.setCursor(6,0);

    lcd.print("LEDS");

    lcd.setCursor(7, 1);

    lcd.print("ON");

  }

  if (message == "ledsOFF")

  {

    beepingTimes(5, 50);

    bomb.leds = false;

    Serial.println("\nLeds are OFF!");

    lcd.clear();

    lcd.setCursor(6,0);

    lcd.print("LEDS");

    lcd.setCursor(6, 1);

    lcd.print("OFF");

  }

  if (message == "soundON")

  {

    beepingTimes(5, 50);

    bomb.sound = true;

    Serial.println("\nSound is ON!");

    lcd.clear();

    lcd.setCursor(5,0);

    lcd.print("SOUND");

    lcd.setCursor(7, 1);

    lcd.print("ON");

  }

  if (message == "soundOFF")

  {

    beepingTimes(5, 50);

    bomb.sound = false;

    Serial.println("\nSound is OFF!");

    lcd.clear();

    lcd.setCursor(6,0);

    lcd.print("SOUND");

    lcd.setCursor(6, 1);

    lcd.print("OFF");

  }

  if (message == "smokeON")

  {

    beepingTimes(5, 50);

    bomb.smoke = false;

    Serial.println("\nSmoke is ON!");

    lcd.clear();

    lcd.setCursor(5,0);

    lcd.print("SMOKE");

    lcd.setCursor(7, 1);

    lcd.print("ON");

  }

  if (message == "smokeOFF")

  {

    beepingTimes(5, 50);

    bomb.smoke = false;

    Serial.println("\nSmoke is OFF!");

    lcd.clear();

    lcd.setCursor(5,0);

    lcd.print("SMOKE");

    lcd.setCursor(6, 1);

    lcd.print("OFF");

  }

  if (message == "sendConfig")

  {

    beepingTimes(1, 2000);

    bluetoothConfigured = true;

    Serial.println("\nGame READY!!");

    lcd.clear();

    bombIsReady();

    delay(2000);

    lcd.clear();

  }

  delay(50);

  }

  }

}

## Keypad

boolean isNum(char);

boolean isNum(char \_key){

  switch (\_key) {

    case '0': {

      return true;

      break;

    }

    case '1':  {

      return true;

      break;

    }

    case '2': {

      return true;

      break;

    }

    case '3': {

      return true;

      break;

    }

    case '4': {

      return true;

      break;

    }

    case '5':  {

      return true;

      break;

    }

    case '6': {

      return true;

      break;

    }

    case '7': {

      return true;

      break;

    }

    case '8': {

      return true;

      break;

    }

    case '9': {

      return true;

      break;

    }

    default :{

      return false;

    }

  }

}

## ManuallyConfig

boolean manuallyConfigured();

extern boolean isNum(char);

extern void printPlayers();

extern void printTimeGame();

extern void printTimeBomb();

extern void printBombCode();

extern void printDigit(String);

extern void printSmoke(String);

extern void printSound(String);

extern void printLeds(String);

extern void printGPS(String);

extern void printConfirmConfig();

extern void beepingTimes(int, int);

enum Config {

  Players,

  TimeGame,

  TimeBomb,

  BombCode,

  GPS,

  SetNewCoordinate,

  Leds,

  Sound,

  Smoke,

  Confirm

};

boolean manuallyConfigured(){

  Config config = Players;

  String \_players = "";

  String \_timeGame = "30";

  String \_timeBomb = "15";

  String \_code = "";

  char key;

  while (!Configured) {

    switch(config){

      case Players: {

        printPlayers();

        printDigit(\_players);

        key = keypad.getKey();

        if (key) {

          beepingTimes(1, 50);

          if (isNum(key)) {

            //lcd.clear();

            \_players += key;

            printDigit(\_players);

            }

          if (key == 'C' && \_players.length() > 0) {

            lcd.clear();

            \_players = \_players.substring(0, \_players.length() - 1);

            printDigit(\_players);

            }

          if (key == 'D' && \_players.length() > 0) {

            lcd.clear();

            config = TimeGame;

            bomb.players = \_players.toInt();

            }

          }

        break;

        }

        case TimeGame: {

          printTimeGame();

          printDigit(\_timeGame);

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (isNum(key)) {

              //lcd.clear();

              \_timeGame += key;

              printDigit(\_timeGame);

            }

            if (key == 'C' && \_timeGame.length() > 0) {

              lcd.clear();

              \_timeGame = \_timeGame.substring(0, \_timeGame.length() - 1);

              printDigit(\_timeGame);

            }

            if (key == 'D' && \_timeGame.length() > 0) {

              bomb.gameTime = \_timeGame.toInt();

              bomb.gameTime \*= 60000;

              config = TimeBomb;

              lcd.clear();

            }

          }

          break;

        }

        case TimeBomb : {

        printTimeBomb();

        printDigit(\_timeBomb);

        key = keypad.getKey();

        if (key) {

          beepingTimes(1, 50);

          if (isNum(key)) {

            //lcd.clear();

            \_timeBomb += key;

            printDigit(\_timeBomb);

          }

          if (key == 'C' && \_timeBomb.length() > 0) {

            lcd.clear();

            \_timeBomb = \_timeBomb.substring(0, \_timeBomb.length() - 1);

            printDigit(\_timeBomb);

          }

          if (key == 'D' && \_timeBomb.length() > 0) {

            bomb.time = \_timeBomb.toInt() \* 60000;

            config = BombCode;

            lcd.clear();

          }

        }

        break;

        }

        case BombCode: {

        printBombCode();

        printDigit(\_code);

        key = keypad.getKey();

        if (key) {

          beepingTimes(1, 50);

          if (isNum(key)) {

            //lcd.clear();

            \_code += key;

            printDigit(\_code);

            }

          if (key == 'C' && \_code.length() > 0) {

            lcd.clear();

            \_code = \_code.substring(0, \_code.length() - 1);

            printDigit(\_code);

            }

          if (key == 'D' && \_code.length() > 0 && \_code.length() == \_players.toInt()) {

            bomb.code = \_code;

            config = GPS;

            lcd.clear();

            }

          }

          break;

        }

        case GPS: {

          printGPS(bomb.checkGPS());

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'B' && bomb.gps) {

              lcd.clear();

              bomb.gps = false;

            }

            if (key == 'A' && !bomb.gps) {

              lcd.clear();

              bomb.gps = true;

            }

            if (key == 'D') {

              bomb.gps ? config = SetNewCoordinate : config = Leds;

              lcd.clear();

            }

          }

          break;

        }

        case SetNewCoordinate: {

          gps.satellites.value() < 6 ? WaitSat() : printCoordinate();

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'A') {

              lcd.clear();

              setNewCoordinates();

              bomb.latZone = gps.location.lat();

              bomb.longZone = gps.location.lng();

              delay(2000);

              }

            if (key == 'D') {

              config = Leds;

              lcd.clear();

            }

          }

          break;

        }

        case Leds: {

          printLeds(bomb.checkLeds());

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'B' && bomb.leds) {

              lcd.clear();

              bomb.leds = false;

            }

            if (key == 'A' && !bomb.leds) {

              lcd.clear();

              bomb.leds = true;

            }

            if (key == 'D') {

              config = Sound;

              lcd.clear();

            }

          }

          break;

        }

        case Sound: {

          printSound(bomb.checkSound());

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'B' && bomb.sound) {

              lcd.clear();

              bomb.sound = false;

            }

            if (key == 'A' && !bomb.sound) {

              lcd.clear();

              bomb.sound = true;

            }

            if (key == 'D') {

              config = Confirm;

              //config = Smoke;

              lcd.clear();

            }

          }

          break;

        }

        case Smoke: {

          printSmoke(bomb.checkSmoke());

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'B' && bomb.smoke) {

              lcd.clear();

              bomb.smoke = false;

            }

            if (key == 'A' && !bomb.smoke) {

              lcd.clear();

              bomb.smoke = true;

            }

            if (key == 'D') {

              config = Confirm;

              lcd.clear();

            }

          }

          break;

        }

        case Confirm: {

          printConfirmConfig();

          key = keypad.getKey();

          if (key) {

            beepingTimes(1, 50);

            if (key == 'C') {

              lcd.clear();

              config = Players;

            }

            if (key == 'D') {

              lcd.clear();

              Configured = true;

            }

          }

          break;

        }

      }

    }

    return true;

  }

## Setup

extern void beepingTimes(int, int);

void setup\_ori() {

  bomb.bombStatus = initialize;

  pinMode(beep, OUTPUT);

  beepingTimes(3, 250);

  Serial.begin(115200);

    // SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally

    if(!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {

      Serial.println(F("SSD1306 allocation failed"));

      for(;;); // Don't proceed, loop forever

      }

  Wire.begin();

  lcd.begin(&Wire);

  lcd.display();

  lcd.backlight();

  display.clearDisplay();

  display.display();

  neogps.begin(9600, SERIAL\_8N1, RXD2, TXD2);

  BT.begin(device\_name);

  FastLED.addLeds<WS2812B, PINO\_FITA, GRB>(fita, NUM\_LEDS\_FITA);

  FastLED.setBrightness(LED\_BRIGHTNESS);

  delay(2000);

  display.display();

  Serial.printf("The device with name \"%s\" is started \nNow you can pai it with Bluetooth\n", device\_name.c\_str());

  Serial.print("MAC Address: "); Serial.println(BT.getBtAddressString());

  xTaskCreatePinnedToCore(

    gpsTracker,

    "Task2",

    10000,

    NULL,

    1,

    &Task2,

    0);

};

## Beep

void beepingTimes(int, int);

void beepBomb();

void beepOn(boolean);

extern unsigned long lastBeep;

extern int bombTime;

void beepingTimes(int times, int time){

  if (bomb.sound) {

    for (int i = 0; i < times; i++) {

      digitalWrite(beep, HIGH);

      delay(time);

      digitalWrite(beep, LOW);

      delay(time);

    }

  }

}

void beepBomb() {

  int bombState = 0;

  if (bomb.time > bombTime / 2){

    //bombState = 2000;

    bombState = bomb.speedSound[3];

  }else if (bomb.time < bombTime / 2 && bomb.time > bombTime / 4) {

    bomb.speedLight = 50;

    //bombState = 1000;

    bombState = bomb.speedSound[2];

  } else if (bomb.time < bombTime / 4 && bomb.time > 20000) {

    bomb.speedLight = 100;

    //bombState = 500;

    bombState = bomb.speedSound[1];

  } else if (bomb.time < 20000) {

    bomb.speedLight = 255;

    //bombState = 250;

    bombState = bomb.speedSound[0];

  }

  if (millis() - lastBeep >= bombState && bomb.sound) {

    if (digitalRead(beep) == HIGH) {

      lastBeep = millis();

      digitalWrite(beep, LOW);

    }else {

      digitalWrite(beep, HIGH);

      lastBeep = millis();

    }

  } else if (millis() - lastBeep < bombState && millis() - lastBeep >= 50 && bomb.sound) {

      int stateBeeper = digitalRead(beep);

      if (stateBeeper == HIGH) {

        digitalWrite(beep, LOW);

      }

  }

}

void beepOn(boolean beeping) {

  if (beeping) {

    delay(10);

    digitalWrite(beep, HIGH);

  } else {

    digitalWrite(beep, LOW);

  }

}

## Core0

## Core1

extern void teamIntro();

extern void printBluetoothChoice();

extern void printBluetoothConfig();

extern void bluetoothConfig(BluetoothSerial);

extern boolean manuallyConfigured();

extern void invalidZone();

extern void availableZone();

extern void insertCode();

extern void correctCode();

extern void wrongCode();

extern void tryAgain();

extern void bombClock(unsigned long, int, int);

extern void youWin();

extern void bombArmed();

extern void printDigit(String);

extern void printCode(String);

extern void bombExploded();

extern void bombExplodedToArming();

extern void restart();

extern void beepingTimes(int, int);

extern void beepBomb();

extern void beepOn(boolean);

void core\_1();

unsigned long lastBeep;

int bombTime;

String code = "";

String secondCode = "";

int codeSize = 0;

int secondCodeSize = 0;

int minutesAnterior = bomb.time / 1000;

int secondsAnterior = 59;

unsigned long int bombTime\_millisAnterior = 0;

unsigned long int clock\_millisAnterior = 0;

unsigned long int gameTimeLast = 0;

int count = 0;

int timeMin;

int timeSec;

enum GameStatus {

  Intro,

  Configuration,

  Prepared,

  ReadyToArm,

  TryCode,

  VerifyCode,

  Disarm,

  Explode,

  ExplodeTryArming,

  Restart

};

GameStatus gameStatus = Intro;

void core\_1(){

  switch(gameStatus){

    case Intro: {

      char key = keypad.getKey();

      key ? beepingTimes(1, 50): void();

      teamIntro();

      if (key == 'D') {

        lcd.clear();

        gameStatus = Configuration;

      }

      break;

    }

    case Configuration: {

      bomb.bombStatus = configuration;

      printBluetoothChoice();

      char key = keypad.getKey();

      if (key == 'A'){

        beepingTimes(1, 50);

        lcd.clear();

        String message = "";

        printBluetoothConfig();

        bluetoothConfig(BT);

        lcd.clear();

        gameStatus = Prepared;

        gameTimeLast = millis();

      }

      if (key == 'D'){

        beepingTimes(1, 50);

        lcd.clear();

        if (manuallyConfigured()) {

          gameStatus = Prepared;

        }

      }

      break;

    }

    case Prepared: {

      if(bomb.gps) {

        if(bomb.isValidZone()){

          availableZone();

          char key = keypad.getKey();

          if(key == 'D') {

            gameStatus = ReadyToArm;

          }

        }else{

          lcd.setCursor(0,0);

          invalidZone();

        }

      } else {

        gameStatus = ReadyToArm;

      }

      break;

    }

    case ReadyToArm: {

      bomb.bombStatus = readyToArm;

      if(!bomb.isValidZone() && bomb.gps) {

        gameStatus = Prepared;

      }

      if (millis() - gameTimeLast >= bomb.gameTime) {

        bomb.bombStatus = explode;

        gameStatus = Explode;

        lcd.clear();

        break;

      }

      insertCode();

      char key = keypad.getKey();

      if(key){

        beepingTimes(1, 50);

        if(key == 'D' && bomb.tryArming > 0 && codeSize > 0){

          if(code == bomb.code){

            gameStatus = TryCode;

            //Serial.println("https://www.google.com/maps/place/40.769838,-8.026673");

            lastBeep = millis();

            timeMin = bomb.time / 60000;

            bombTime = bomb.time;

            timeSec = 0;

            lcd.clear();

            bomb.bombStatus = armed;

          }else{

            lcd.clear();

            wrongCode();

            tryAgain();

            delay(500);

            bomb.tryArming--;

            code = "";

            codeSize = 0;

            delay(500);

            lcd.clear();

            if (bomb.tryArming == 0){

              lcd.clear();

              gameStatus = ExplodeTryArming;

            }

          }

        } else if (key == 'C' && codeSize > 0){

          lcd.clear();

          codeSize--;

          if (codeSize > 0) {

            code = code.substring(0, code.length() - 1);

            printCode(code);

          }else {

            code = "";

          }

        }else  if (isNum(key)) {

          code += key;

          codeSize++;

          printDigit(code);

        }

      }

      break;

    }

    case TryCode: {

      bomb.leds ? respiracao(22, bomb.speedLight) : fillSolidColor(CRGB::Black);

      if (millis() - gameTimeLast >= bomb.gameTime) {

        bomb.bombStatus = explode;

        gameStatus = Explode;

        lcd.clear();

        break;

      }

      beepBomb();

      printDigit(secondCode);

      bombArmed();

      if (bomb.time > 0) {

        if (timeSec == 0) {

          bombTime\_millisAnterior = millis();

          timeMin--;

          timeSec = 59;

          lcd.clear();

        } else if (millis() - bombTime\_millisAnterior >= 1000) {

          bombTime\_millisAnterior = millis();

          timeSec--;

          bomb.time -= 1000;

          lcd.clear();

        }

        if (timeMin > 0 || timeSec > 0){

          printClock(timeMin, timeSec);

        }

      }else {

        lcd.clear();

        gameStatus = Explode;

      }

      char key = keypad.getKey();

      if (key){

        beepingTimes(1, 50);

        if (key == 'D' && secondCodeSize > 0){

          gameStatus = VerifyCode;

          lcd.clear();

        }else if (key == 'C' && secondCodeSize > 0){

          lcd.clear();

          secondCodeSize--;

          if (secondCodeSize > 0) {

            secondCode = secondCode.substring(0, secondCode.length() - 1);

            printCode(secondCode);

          }else{

            secondCode = "";

          }

        }else if (isNum(key)){

          secondCode += key;

          secondCodeSize++;

          printDigit(secondCode);

        }

      }

      break;

    }

    case VerifyCode: {

      if(bomb.checkCode(secondCode)){

        lcd.clear();

        correctCode();

        bomb.codeDiscovered = true;

        gameStatus = Disarm;

        lcd.clear();

      } else if (bomb.tries == 0){

        lcd.clear();

        bomb.codeDiscovered = false;

        gameStatus = Explode;

      }else{

        lcd.clear();

        bomb.codeDiscovered = false;

        wrongCode();

        tryAgain();

        bomb.tries--;

        bomb.leds ? fillSolidColor(CRGB::Red) : fillSolidColor(CRGB::Black);

        (bomb.tries > 0) ? delay(5 \* 1000) : delay(0);

        fillSolidColor(CRGB::Black);

        secondCodeSize = 0;

        secondCode = "";

        lcd.clear();

        gameStatus = TryCode;

      }

      break;

    }

    case Disarm: {

      bomb.leds ? fillSolidColor(CRGB::Green) : fillSolidColor(CRGB::Black);

      bomb.bombStatus = disarm;

      youWin();

      beepOn(true);

      char key = keypad.getKey();

      if (key == 'D') {

        gameStatus = Restart;

        lcd.clear();

        fillSolidColor(CRGB::Black);

        delay(200);

      }

      break;

    }

    case Explode: {

      bomb.leds ? fillSolidColor(CRGB::Red) : fillSolidColor(CRGB::Black);

      bomb.bombStatus = explode;

      beepOn(true);

      bombExploded();

      char key = keypad.getKey();

      if (key == 'D') {

        gameStatus = Restart;

        lcd.clear();

        fillSolidColor(CRGB::Black);

        delay(200);

      }

      break;

    }

    case ExplodeTryArming: {

      bomb.leds ? fillSolidColor(CRGB::Red) : fillSolidColor(CRGB::Black);

      beepOn(true);

      bombExplodedToArming();

      char key = keypad.getKey();

      if (key == 'D') {

        lcd.clear();

        gameStatus = Restart;

        fillSolidColor(CRGB::Black);

        delay(200);

      }

      break;

    }

    case Restart: {

      char key = keypad.getKey();

      restart();

      if (key) {

        beepingTimes(1, 50);

        if (key == 'D') {

          fillSolidColor(CRGB::Black);

          beepOn(false);

          ESP.restart();

        }

      }

      break;

    }

    default: {

      Serial.println("Something wrong!!");

      break;

    }

  }

}

# Bibliografia

https://www.printables.com/model/146410-airsoft-arduino-bomb-replica-kajaki-airsoft

<https://github.com/yinbot/Airsoft-BombPro>

https://github.com/JorgeFilipePinto/Airsoft\_Bomb\_Montain\_Wolves\_Team.git