Table of Contents

GitHub Username: hammerox

Acenda o Farol Baixo

("*Turn On the Low Beams"* in Portuguese)

Description

Avoid getting traffic tickets by turning your car's low beams on when daylight.

This app sends a simple reminder every time you get on a vehicle, allowing you to turn the lights on even before getting into a highway.

The law demands drivers to use low beams on highways even when daylight. Whoever fails to comply are committing a medium level infringement and shall be fined by R$85,15 and 4 points on your driver's license. In November, this value will be readjusted to R$130,16.

The Brazilian law 13.290/2016 has entered into force on July 08 2016 and since then thousands of tickets were issued throughout the country.

"It is better to prepare and prevent than it is to repair and repent."

Intended User

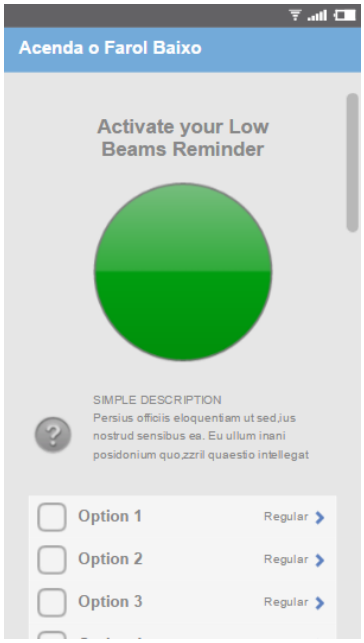
This app is developed for every Brazilian driver, especially the ones that drives frequently through highways.

Features

- Detects if user is on a vehicle
- Sends a warning to turn on the low beam
- Warnings just when daylight
- User can customize the warning's configurations, such as:
    - Notifications;
    - Sounds;
    - Work time;
    - Detector's precision vs Battery consumption

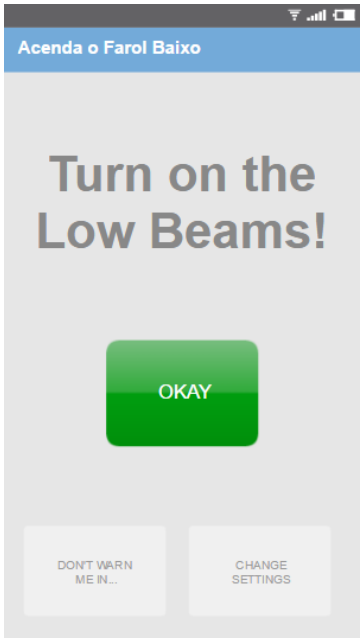
User Interface Mocks

Screen 1: Main Screen



This is the main screen. It contains a button for activating and deactivating the alarm of the vehicle detector. A simple description of the app is present, as well as a button which shows detailed information about how the app works. Below it will contain all of the app's settings so the user can customize its behavior. At the very end of the screen the user will have access to a list all times the alarm was triggered.

Screen 2: Alarm Screen



This is the alarm screen. It will be launched when the app is triggered up by the detector and will emit a sound to warn the user. "Okay" button will close the app and maintain the configurations unchanged. "Don't Warn Me In…" will put the detector to sleep for an X amount of time, defined by the user. "Change Settings" will send the user to the main screen where it will be able to reconfigure the app. This screen can be disabled by the configurations.

Key Considerations

How will your app handle data persistence?

Data will be stored on internal storage using SharedPreferences as all the customization will be done through a PreferenceFragment.

Describe any corner cases in the UX.

When the detector triggers the alarm, the app will:

- Send a notification, registering the time it was triggered;
- Emit a sound;
- Show the alarm screen;

All of the options above can be disabled by the user's configuration.

Describe any libraries you'll be using and share your reasoning for including them.

Smart Location Library – To detect whether the user's device is on a vehicle or not.
https://github.com/mrmans0n/smart-location-lib

Describe how you will implement Google Play Services.

The app will use:

- Google's Location and Activity Recognition, which is handled entirely by the Smart Location Library;
- Google's Analytics, which will be added as a dependency to the app using Gradle;

Next Steps: Required Tasks

Task 1: Project Setup

- Configure Smart Location Library (SLLib) on Gradle;
- Check if SLLib detector is running properly:
    - Setup a layout to show SLLib's output;
    - Build app on a real device;
    - Drive a car with the app running;

Task 2: Detector's Setup

- Create tools/logs to better understand how SLLib is working;
- Store SLLib's temporary data on SharedPreferences;
- Setup SLLib to work on the background;
- Setup an algorithm to trigger a sound when certain conditions are satisfied:
    - Setup system to emit a sound;
    - Define and develop the alarm trigger requirements;
    - Test app on a car;
    - Adjust algorithm using the tools/logs;

Task 3: UX Design

- Design main and alarm screens' layout;
- Wire the trigger to show the alarm screen;
- Setup code to show a notification on trigger;

Task 4: Setting's Customization

- Define on paper all options to be added to the setting's list:
    - Sound to emit on trigger;
    - Show notification or not;
    - Show alarm screen or not;
    - Period to listen;
    - SLLib's update rate;
    - Others;
- Create a PreferenceFragment and place it on the main screen;
- Code all options and add them to show on the PreferenceFragment;
- Check if all options work as expected;
- Create a sleep mode to be used on the alarm screen;

Task 5: Data and Widget

- Configure a database to store time and date of alarm triggers;
- Develop a ContentProvider for accessing the database;
- Develop a Loader to retrieve data from the database to the main screen;
- Create a widget to display:
    - The detectors activation/deactivation button;
    - A list with the database content;

Task 6: Last Details

- Set and check all button's actions;
- Enter description on main screen;
- Create an AlertDialog and enter all detailed information about how the detector works;
- Add Google Analytics;
- Include support for accessibility;

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"