UNIVERSIDADE COMUNITÁRIA DA REGIÃO DE CHAPECÓ

(UNOCHAPECÓ)

Área de Ciências Exatas e Ambientais Ciência da Computação

Jonas Rodrigeri

PROTÓTIPO DE SISTEMA PARA TELEMETRIA AUTOMOBILÍSTICA COM ARDUINO E PHP

Chapecó – SC, Jun. de 2015

JONAS RODRIGERI

PROTÓTIPO DE SISTEMA PARA TELEMETRIA AUTOMOBILÍSTICA COM ARDUINO E PHP

Trabalho de conclusão de curso, apresentado à UNOCHAPECO, como parte das exigências para a obtenção do título de Cientista da Computação.

Chapecó – SC, jun. 2015

FOLHA DE APROVAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

Esta Monografia foi julgada para obtenção do título de Bacharel em Ciência da Computação, no Curso de Graduação em Ciência da Computação da Universidade Comunitária da região de Chapecó - UNOCHAPECÓ – Campus Chapecó

Apresentação à Comissão Examinadora integrada pelos professores:

_____

Prof. Esp. Rodrigo Alceu Benedet
Orientador

_____

Prof. Esp. Marcos Moretto
Membro da banca

_____

Prof. Esp. Cezar Junior Souza
Membro da banca

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI	*American National Standards Institute*
API	*Interface de Programação de Aplicativos*
ASP	*Active Server Pages*
CBA	Confederação Brasileira de Automobilismo
CGI	*Common Gateway Interface*
ETAs	Estações de Tratamento de Águas
ETEs	Estações de Tratamento de Efluentes
FIA	*Federation Internationale de l'Automobile*
GPRS	Serviço de Rádio de Pacote Geral
GSM	*Global System for Mobile Communications*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IBM	*International Business Machines*
IDE	*Integrated Development Environment*
ISO	*International Organization for Standardization*
PHP	*Hypertext Pre-Processor*
SGBDOR	Sistema de Gerenciamento de Banco de Dados Objeto Relacional
SGDB	Sistema de Gerenciamento de Banco de Dados
SGML	*Standar Generalized Markup Language*
SO	*Operating System*
SQL	*Structured Query Language*
USB	*Universal Serial Bus*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente muitas modalidades de corridas automobilísticas são disputadas no Brasil, desde campeonatos amadores disputados em pistas de terra a campeonatos mundiais disputados em circuitos modernos com alto grau de investimento.

Apesar dessa modalidade de esporte ser muito praticada no país (são 43 corridas confirmadas para o ano de 2015, essas organizadas apenas pela Confederação Brasileira de Automobilismo), ainda são pouco aparadas por sistemas de monitoramento de informações referentes aos veículos durante a corrida por meio de telemetria.

Atualmente o mercado dispõe de alguns sistemas de monitoramento de dados por meio de telemetria, porém exigem altos custos de investimento, o que acaba por inviabilizar a sua utilização. Por exemplo, a Formula1 utiliza um sofisticado porem caro, sistema de telemetria, que transmite informações do veículo praticamente em tempo real (com incríveis apenas 2 ms de atraso) para a equipe de engenheiros analisar, por regra da FIA, só podem ser recebidos dados de um veículo e não podem ser enviados.

Com isso será apresentado um projeto para a implantação de um sistema de telemetria automobilística com Arduino e PHP. No qual será abordado sobre os eventos automobilísticos, quais as informações a serem coletadas através de telemetria, qual a forma de captação das informações com Arduino e como elas serão exibidas.

O protótipo do sistema terá como propósito facilitar o acesso das informações referente aos veículos, durante uma corrida, também exigirá um baixo investimento para assim também ser acessível a pequenos campeonatos automobilísticos.

O protótipo contará com um dispositivo de coleta de dados, que ficará localizado no veículo, transmitindo as informações necessárias via conexão de rádio frequência. Será leve e ficará fixado de modo a não oferecer riscos às pessoas envolvidas na corrida, tanto pilotos, quanto telespectadores e equipes. Além também de não interferir na condução do veículo.

1.1 Tema

Protótipo de Sistema para Telemetria Automobilística com Arduino e PHP.

1.2 Delimitação do Problema

Será desenvolvido um protótipo de sistema para a captação de informações automobilísticas através de telemetria, para isso serão usados sensores de velocidade e de posicionamento, tendo seu foco principal em corridas automobilísticas disputadas no Brasil.

Para exibir as informações captadas pela telemetria, será utilizada a linguagem de programação PHP, conectada ao banco de dados PostgreSQL e com HTML5, em um sistema *web*.

Será um protótipo barato e de fácil implantação, contará apenas com uma bateria embutida além dos sensores responsáveis pela transmissão das informações, que deverão ser transmitidas por meio módulos de rádio frequência.

A apresentação das informações se dará através de um sistema web com um mapa que mostrará as localizações dos veículos na pista e uma tabela com informações mais detalhadas dos veículos.

1.3 Hipóteses ou Questões de Pesquisa

1. Como é feita a captação de dados no automobilísticos?
2. Quais são as informações a serem coletadas através da telemetria?
3. Qual a forma de captação de informações com Arduino?
4. Dentre as principais tecnologias de apresentação de dados na web, qual é a melhor maneira de demonstrar estes dados de telemetria para o usuário?

2 OBJETIVOS

Este projeto para ser desenvolvido será dividido em objetivos, os quais podem ser gerais e específicos.

2.1 Objetivo geral

Desenvolver um protótipo de sistema para a utilização de Telemetria Automobilística com Arduino e PHP.

2.2 Objetivos Específicos

- Analisar as metodologias adotadas nos eventos automobilísticos no Brasil.
- Definir as informações a serem coletadas através da telemetria.
- Compreender a forma de captação de informações com Arduino.
- Dentre as principais tecnologias de apresentação de dados na *web*, especificar qual a melhor maneira de demonstrar os dados da telemetria para o usuário.

3 JUSTIFICATIVA

Atualmente no Brasil as corrida automobilísticas em sua grande maioria, não dispõem de um sistema de monitoramento das informações dos veículos durante uma corrida. Os poucos sistemas disponíveis são grandes e complexos, acarretando assim num valor muito elevado para os níveis financeiros dessas corridas.

Visando um fácil acesso as informações dos veículos durante uma corrida, tanto pelas equipes quanto pelo público, será desenvolvido um protótipo de sistema de telemetria com Arduino, capaz de captar informações e transportá-las via ondas de rádio para um computador, onde será armazenado num banco de dados e imediatamente replicado para um sistema *web* (desenvolvido com PHP) para que todas as pessoas com acesso a esse sistema possam ver.

O protótipo será capaz de captar informações de velocidade e posicionamento do veículo na pista, e fará isso sem ter nenhuma ligação com o mesmo, isso porque os sensores são independentes, só precisam estar posicionados dentro do veículo para que possam transmitir as informações.

Além de não interferir na condução do veículo, o protótipo de captação de informações será de baixo custo para as equipes de corrida. Podendo assim trazer uma grande quantidade de informações a elas, sem grandes gatos.

A apresentação se dará por um sistema web onde, terá um mapa com a localização do veículo durante a corrida, logo baixo desse mapa, haverá uma tabela com as informações detalhadas dos veículos.

4 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica é dividida em sete temas, Automobilismo, Telemetria, Arduino, Banco de Dados, HTML, PHP e Mapas Digitais. Segue abaixo o desenvolvimento.

4.1 Automobilismo

Segundo Scariot (2006), o automobilismo é um esporte de corrida de carros, que teve início no ano de 1894. Em 1904 criou-se a FIA (Federação Internacional Automobilística), que começou a criar autódromos em diversos países como, por exemplo, o Brooklands na Inglaterra, o de Indianápolis nos Estados Unidos e o Milanes na Itália.

Figura 1: Corrida automobilística.

Fonte: https://neitessari.files.wordpress.com/2011/03/stock.jpg.

4.1.1 Automobilismo no Brasil

No Brasil o automobilismo teve início em 1908, no estado de São Paulo, com o circuito de Itapecerica. Até então as corridas eram realizadas em rodovias das grandes capitais. Então no dia 12 de Maio de 1940, foi inaugurado o primeiro circuito fechado do país, o Autódromo de Interlagos (SCARIOT, 2006).

Figura 2: Autódromo de Interlagos.

Fonte: http://www.podcorrer.com/wp-content/uploads/2008/07/interlagos-sp.jpg.

A CBA (Confederação Brasileira de Automobilismo) é a associação máxima federal de administração de desporto do automobilismo brasileiro e foi fundada em 7 de Setembro de 1961, sendo a CBA filiada da FIA (CONFEDERAÇÃO BRASILEIRA DE AUTOMOBILISMO, 2015).

Entre os principais objetivos da CBA está o compromisso de dirigir, difundir e incentivar no país, a prática de todas as modalidades desportivas automobilísticas e desde que credenciada, desenvolver as atividades ligadas ao turismo, trânsito e transportes. Cabe a CBA coordenar o complexo técnico desportivo do automobilismo brasileiro, que é regulado por normas nacionais e internacionais através das regras de práticas desportivas, e essas regras deverão ser aceitas por todos que façam parte do sistema desportivo nacional do automobilismo (CONFEDERAÇÃO BRASILEIRA DE AUTOMOBILISMO, 2015).

A seguir apresenta-se a telemetria, que irá descrever um pouco sobre captação de dados por meio de conexões sem fio, essas informações podem ser utilizadas por várias áreas, neste trabalho é focado em telemetria automobilística.

4.2 Telemetria

Segundo Gpssat (2012), telemetria é uma tecnologia que nos permite obter, analisar e repassar dados ou informações remotamente, esses dados podem estar em um veículo, por exemplo, e ser transmitido por algum meio que não utiliza uma ligação por cabos, na maioria

das vezes se trata de rádio frequência. Para isso as frequências são criptografadas, de modo a evitar que as informações sejam acessadas por outras pessoas.

A telemetria é uma tecnologia focada em monitoramento, medição ou rastreamento de objetos através de dados enviados por meio de ondas de rádio ou via *wireless*. É um sistema de monitoramento com diversos sensores, muito falada e utilizada nas corridas de Formula 1, também é muito usada em industrias de monitoramentos e outras competições esportivas (TECNOCONTROL, 2015)

Com a telemetria pode-se capturar diversos tipos de informações, desde informações mais básicas como: velocidade do veículo, rotação do motor, quilômetros percorridos, número de voltas, temperatura do freio e temperatura do motor. Ou informações avançadas como: falhas mecânicas, excessos gerado pelo veículo, restrições de sensores, nível do liquido do arrefecimento do motor (GPSSAT, 2012).

Com o auxílio da telemetria é possível, por exemplo, ajustar melhor um veículo durante uma corrida, apesar das normas impedirem o envio de informações da equipe para o veículo, o recebimento de informações é essencial nas corridas de hoje. A equipe recebe em tempo real as informações do veículo e quando o piloto passa nos boxes para reabastecer ou trocar os pneus, os engenheiros podem aplicar alguma mudança com base nos dados coletados pela telemetria (GPSSAT, 2012).

4.2.1 Aplicações da Telemetria

Segundo Tecnocontrol (2015), a telemetria pode ser aplicada em diversas áreas, abaixo veremos alguns exemplos de sua utilização:

- **Telemetria na Meteorologia:** A telemetria é muito utilizada na medição de dados meteorológicos através de balcões com transmissão de dados desde 1920. Existem aplicações também na hidrografia, recolhendo dados de bacias hidrográficas e estações hidrométricas e os transmitindo aos centros de análises e controles (TECNOCONTROL, 2015).
- **Telemetria de água e esgoto:** Automação, monitoramento e controle, em tempo real, de reservatórios e elevatórias de água e esgoto, ETAs (Estações de Tratamento de Águas) e ETEs (Estações de Tratamento de Efluentes) via rádio. Em um município sem sistema de telemetria, é a população que avisa a companhia de água e esgoto quando ocorre uma falha no abastecimento. O sistema de telemetria permite monitorar em tempo real o funcionamento de estações

elevatórias, reservatórios, medidores de vazão e demais dispositivos elétricos e hidráulicos do sistema (TECNOCONTROL, 2015).

- **Telemetria no gerenciamento hídrico – Hidrografia:** A telemetria é importante na gestão da água, incluindo a qualidade da água e funções de aferição de fluxo. As principais aplicações incluem monitoramento de águas subterrâneas, detecção de vazamentos em tubulações de distribuição e equipamentos de vigilância. Ter os dados disponíveis quase em tempo real permite reações rápidas a eventos na área afetada (TECNOCONTROL, 2015).
- **Telemetria no monitoramento de energia:** Em fábricas, edifícios e casas, o consumo de energia dos sistemas é monitorado em vários locais; parâmetros relacionados (por exemplo, temperatura) são enviados via telemetria sem fio à um local central. As informações são coletadas e processadas, permitindo o uso mais eficiente da energia (TECNOCONTROL, 2015).
- **Telemetria na distribuição de recursos:** Muitos recursos precisam ser distribuídos por grandes áreas. Telemetria é útil nesses casos, uma vez que permite que o sistema de canalização de recursos em que eles são necessários, são exemplos as explorações de tanques em refinarias e fábricas de produtos químicos de gasolina (TECNOCONTROL, 2015).
- **Telemetria em Fornecedores de Energia – Usinas de Energia:** Em alguns países, a telemetria é usado para medir a quantidade de energia elétrica consumida. O medidor de energia elétrica se comunica com um concentrador, e este último envia a informação através de GPRS (Serviço de Rádio de Pacote Geral) ou GSM (*Global System for Mobile Communications*) para o servidor do fornecedor de energia (TECNOCONTROL, 2015).

4.2.2 Aplicação da Telemetria em Corridas de Formula 1

No início dos anos 80, os computadores se tornaram equipamentos obrigatórios nas corridas de Formula 1. No início eram utilizados apenas para as tarefas básicas como cronometragem do tempo, contagem de voltas e consumo de combustível, com o passar do tempo começaram a fazer um pouco de tudo, de dentro dos boxes a equipe usa computadores potentes para captar informações do veículo em tempo real durante a corrida, por meio da telemetria com comunicação via ondas de rádio, com isso a equipe pode saber onde o piloto acelera, freia ou troca de marcha. Com essas informações em mãos a equipe pode simular

uma volta perfeita baseada nas características do veículo, e tentar aplicá-la na prática (ENCICLOPÉDIA F1, 2015).

4.2.3 Captação de dados com Telemetria

Segundo Viva O Linux (2004), o sistema de telemetria básico é composto por quatro elementos chave do processo:

1. **Máquinas Inteligentes e Sensores:** Aparelhos que monitoram, controlam e medem algum tipo de atividade localmente. Podem existir vários sensores em um determinado local;

2. **Interface da Aplicação:** Interface entre os sensores e a rede de comunicação. É responsável pela comunicação entre os sensores de captação;

3. **Base de Comunicação:** O sistema pode ser por linhas fixas ou rádio, e transmitir informações dos sensores através da interface da aplicação, para um computador central de comando e um centro de controle;

4. **Centro de Controle e Comando:** Este é o ponto central que recebe os dados transmitidos pelos sensores. A informação é processada, podendo ser repassada para diferentes locações através da internet.

A coleta de informações por meio de telemetria, consiste em ter um sensor instalado em um determinado local repassando informações, essas informações podem variar de acordo com os sensores instalados. Também é necessário ter uma interface para conectar os sensores da aplicação (VIVA O LINUX, 2004).

Para repassar os dados é preciso escolher uma base de comunicação, que pode ser cabeada ou por meio de comunicação sem fio, por fim tem o centro de controle e comando, que recebe e processa as informações obtidas por meio da telemetria (VIVA O LINUX, 2004).

A seguir apresenta-se o Arduino, que será o responsável pala captação e transmissão dos dados, para esses sejam salvos e posteriormente usados pelo protótipo. O Arduino conta com diversos modelos de sensores, que são acoplados a placa Arduino, possibilitando assim a obtenção de dados do ambiente em seu redor.

4.3 Arduino

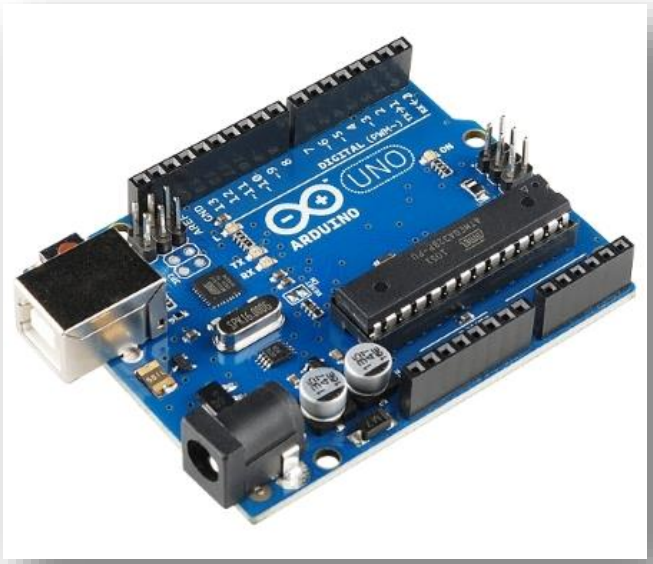
Segundo Arduino (2015), Arduino é uma plataforma eletrônica de código aberto baseado em hardware e software de fácil utilização. É destinado para que qualquer pessoa seja capaz de desenvolver projetos interativos.

Em termos práticos, um Arduino é um pequeno computador que pode ser programado para interagir com sensores de entrada e saída de dados, ou seja, um sistema que pode interagir com o ambiente por meio de *hardware* e *software* (MCROBERTS, 2011).

Segundo Filipeflop (2014), o Arduino foi criado em 2005 por um grupo de 5 pesquisadores: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além de possuir o conceito de hardware livre, permitindo que qualquer um possa montar, modificar, melhorar ou personalizar o Arduino, partindo do mesmo *hardware* base.

Assim, foi criada uma placa composta por um micro controlador Atmel, que é formada por circuitos de entrada e saída, que pode ser facilmente conectada à um computador, e ser programada via IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) utilizando uma linguagem baseada em C/C++, com apenas um cabo USB (*Universal Serial Bus*) (FILIPEFLOP, 2014).

Figura 3: Arduino UNO.



Fonte: http://cnd.blog.filipeflop.com/wp-content/uploads/2014/09/Arduino-Uno-R3.jpg.

Depois de programado, o micro controlador Arduino pode ser usado de forma independente, ou seja, pode se colocá-lo para controlar um robô, uma lixeira, um ventilador, as luzes de uma casa, a temperatura do ar condicionado, pode utilizá-lo como um aparelho de medição ou para muitas outras coisas (FILIPEFLOP, 2014).

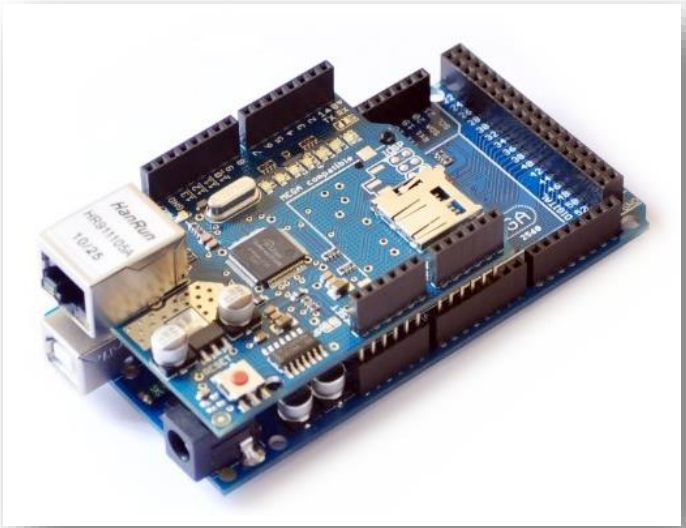
4.3.1 Potencialidades do Arduino

A lista de possibilidades é praticamente infinita. É possível automatizar praticamente tudo, uma casa, um carro, um escritório ou também criar um novo brinquedo ou equipamento, e até mesmo melhorar algo já existente (FILIPEFLOP, 2014).

Para isso, o Arduino possui uma quantidade enorme de sensores e componentes que podem ser utilizados nos projetos. Grande parte do material utilizado no Arduino são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores e *leds* (FILIPEFLOP, 2014).

Existem também as chamadas *shields*, que são placas auxiliares que são encaixadas na placa Arduino para expandir suas funcionalidades. Alguns modelos de placa Arduino já vem com os encaixes próprios para essas *shields*, outros não possuem esse suporte, então é necessário soldar os pinos (FILIPEFLOP, 2014).

Figura 4: *Shield* Arduino.

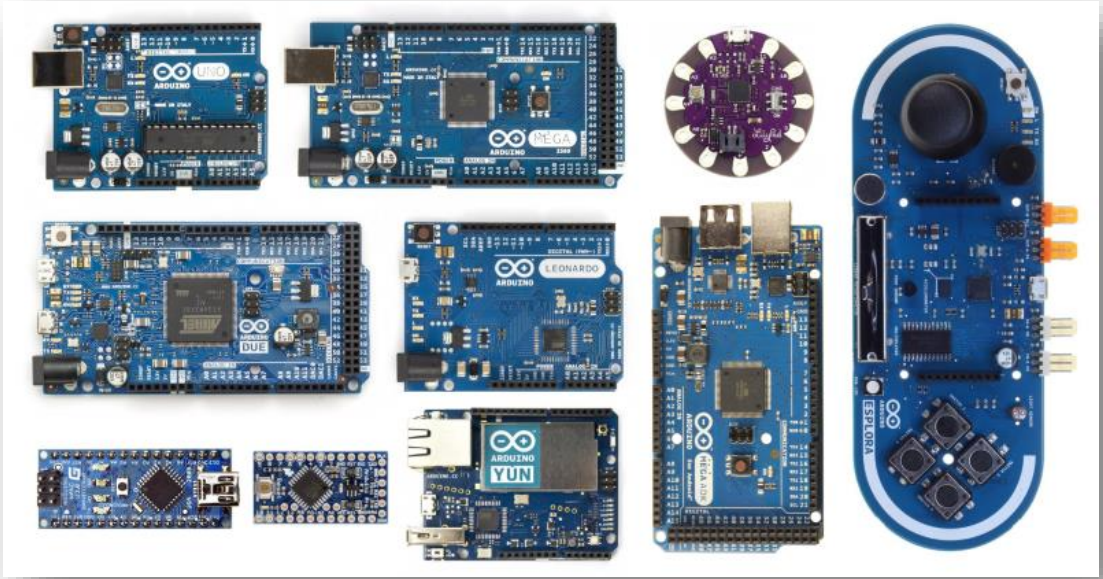


Fonte: http://cnd.blog.filipeflop.com/wp-content/uploads/2014/09/Arduino_Ethernet_Shield_W5100_Mega.jpg

4.3.2 Modelos de Placas Arduino

Segundo Filipeflop (2014), o tipo de placa a se utilizar depende muito do projeto a ser desenvolvido e do número de portas necessárias. As opções vão das mais comuns, como o Arduino Uno e suas 14 portas digitais e 6 analógicas, passando por placas com maior poder de processamento, como o Arduino Mega, com micro controlador ATmega2560 e 54 portas digitais, e o Arduino Due, baseado em processador ARM de 32 *bits* e 512 *Kbytes* de memória:

Figura 5: Modelos de Placas Arduino.



Fonte: http://cnd.blog.filipeflop.com/wp-content/uploads/2014/09/O-que-Arduino.png.

4.3.3 Estrutura de um programa para Arduino

Escrever um programa para um Arduino é muito simples. Tudo o que se precisa é conectar a placa Arduino a um computador por meio de um cabo USB e utilizar um ambiente de programação chamado IDE, onde digita-se o código do programa, faz os testes para encontrar eventuais erros, e transfere o programa para o Arduino (FILIPEFLOP, 2014).

Não é necessário ser expert em linguagem C para programar um Arduino. Pode-se começar um programa utilizando a estrutura básica, que é composta por duas partes, ou dois blocos (FILIPEFLOP, 2014):

> *setup*() – É nessa parte do programa que se configura as opções iniciais do programa: os valores iniciais de uma variável, se uma porta será utilizada como entrada ou saída, mensagens para o usuário, etc. (FILIPEFLOP, 2014).

> *loop*() – Essa parte do programa repete uma estrutura de comandos de forma contínua ou até que alguma comando de "parar" seja enviado ao Arduino. (FILIPEFLOP, 2014).

Levando em consideração o programa abaixo, que acende e apaga o *led* embutido na placa Arduino em intervalos de 1 segundo, pode-se ter um entendimento melhor dessa estrutura (FILIPEFLOP, 2014):

Figura 6: Exemplo de programação com Arduino.

```
//Programa : Pisca Led Arduino
//Autor : FILIPEFLOP

void setup()
{
   //Define a porta do led como saída
   pinMode(13, OUTPUT);
}

void loop()
{
   //Acende o led
   digitalWrite(13, HIGH);

   //Aguarda o intervalo especificado
   delay(1000);

   //Apaga o led
   digitalWrite(13, LOW);

   //Aguarda o intervalo especificado
   delay(1000);
}
```

Fonte: http://blog.filipeflop.com/arduino/o-que-e-arduino.html.

4.3.4 IDE do Arduino

Segundo Mcroberts (2011), a IDE do Arduino é dividida em três partes, quando abre-se a IDE, é encontrado o que pode-se ver na figura 7 abaixo:

Figura 7: IDE do Arduino.

Fonte: http://www.jayconsystems.com/timage/Arduino%20IDE%20-%20Software%20tutorial.jpg.

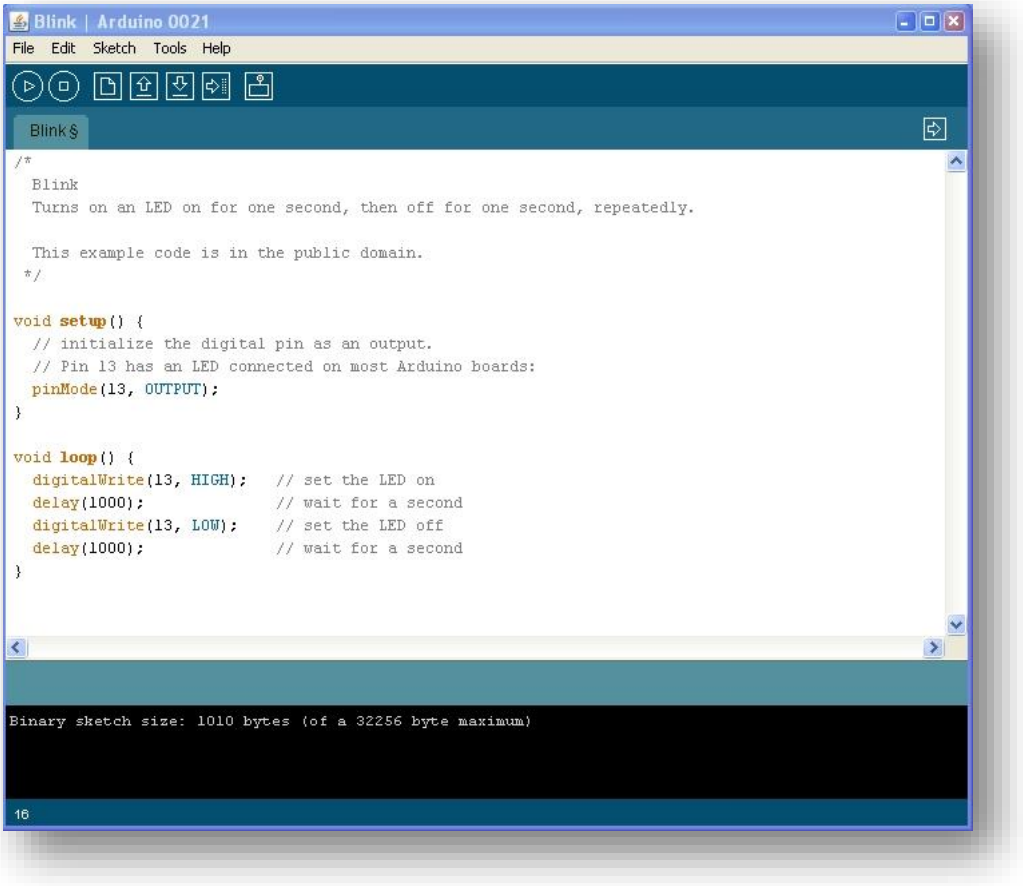
A seguir apresenta-se as três partes em que a IDE do Arduino é dividida.

- ***Toolbar*: o toolbar é onde ficam os botões, que ao todo são sete:
  - ***Verify/Compile*: (Verificar/Compilar)** é utilizado para verificar se o código está sem erros antes de fazer o *upload* para a placa do Arduino;
  - ***Stop*: (Parar)** para a execução do programa e também desmarca os botões marcados do toolbar;
  - ***New*: (Novo)** cria um novo *sketch* (arquivo) em branco, onde será inserido o programa para ser executado pelo Arduino;
  - ***Open*: (Abrir)** apresenta uma lista de *sketches* armazenados no computador ou uma lista de *sketches* que podem ser experimentados com diversos periféricos;

- ***Save*: (Salvar)** salva o código digitado na janela de *sketch* para o arquivo;
- ***Upload to I/O Board*: (Sobe para o Arduino)** sobe o código contido na janela para a sua placa Arduino. É essencial que seja salvo o *sketch* antes de enviar para a placa;
- ***Serial* Monitor: (Monitor *Serial*)** o monitor *serial* é uma ferramenta muito útil para a depuração do código. Também pode-se mandar dados de para a placa Arduino utilizando o monitor *serial*.

- **Código ou *Sketch Window*:** é onde fica o código fonte para ser compilado e executado, como mostra a figura 8 (MCROBERTS, 2011).

Figura 8: Exemplo de janela com código fonte.



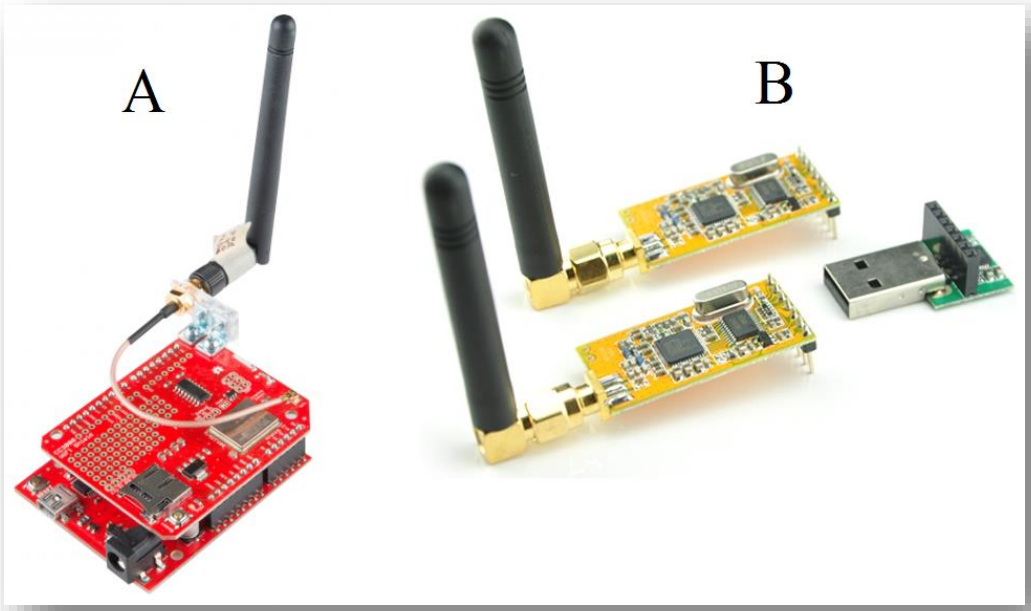
Fonte: http://www.hobbytronics.co.uk/image/data/tutorial/arduino-install/install_6.jpg.

- **Janela de mensagens:** em baixo do código fica a janela de mensagens, onde serão listadas, por exemplo, as mensagens de erro (MCROBERTS, 2011).

4.3.5 Utilizando comunicação sem fio

Segundo Filipeflop (2014), pode-se utilizar a comunicação via *wireless* na transmissão de dados, para isso existem várias formas de comunicação e vários modelos de placas para fazer esta comunicação. Podendo ser por meio de ondas de rádio, por *wi-fi*, por telefonia, por *bluetooth*, entre outras.

Figura 9: Exemplo de placas sem fio para Arduino.



Fonte: (A) http://cnd.blog.filipeflop.com/wp-content/uploads/2014/10/Wifi_Shield_Sparkfun-630x589.png e (B) http://cnd.blog.filipeflop.com/wp-content/uploads/2014/10/Modulo_Radio_Wireless_Apc220_Adaptador_Usb1.jpg.

Para trabalhar com uma *shield* sem fio é necessário um computador com porta USB para a programação do módulo, além de um Arduino e da própria *shield*, para este trabalho uma de ondas de rádio, como pode-se ver a placa “B” na figura 9 acima (FILIPEFLOP, 2014).

Com os componentes em mãos, é só instalar, configurar e usar. Passadas as etapas de instalação e configuração, as *shields* de rádio frequência, se comunicam numa distância de até 1 quilômetro, em ondas de 418 à 455 *Mhz*, podendo chegar a taxa de transferência de 19200 *bps*. Com essa forma de comunicação, pode-se obter informações de um local remoto, como exemplo, um carro, uma casa, uma máquina de indústria. Isso tudo sem a necessidade de estar conectados por um fio (FILIPEFLOP, 2014).

Os dados coletados por meio da telemetria com o auxílio do Arduino, serão salvos em um banco de dados, para assim poderem ser acessados pelas pessoas interessadas. A seguir apresenta-se alguns modelos de bancos de dados, com ênfase maior no PostgreSQL.

4.4 Banco de Dados

Segundo Heuser (1998), os SGDB (Sistemas de Gerência de Banco de Dados), surgiram por volta da década de 70, tendo como objetivo principal facilitar a programação de aplicações de banco de dados.

Inicialmente os bancos de dados eram concentrados nas próprias máquinas onde rodavam as aplicações, assim cada computador possuía seu próprio banco de dados. Mas atualmente com as melhoras na internet, tanto na velocidade quanto na confiabilidade das conexões, isso ficou para trás, agora apenas uma máquina é usada para servir de banco dados, para que muitas outras acessem seus dados. Uma das melhores formas de fazer essas interações é com aplicações *web*, por exemplo, com a utilização do PHP (*Hypertext Pre-Processor*) (NEVES, 2002).

O PHP permite a geração de páginas dinâmicas, sendo embutindo no HTML (*HyperText Markup Language*) que efetua diversas operações para gerar essas páginas, para que sejam dinâmicas, é necessário realizar acessos ao banco de dados. Para isso o PHP conta com diversas funções prontas para serem usadas e de fácil entendimento e aplicação (NEVES, 2002).

4.4.1 Banco de dados relacional e banco de dados orientado a objetos

Segundo Neves (2002), o banco de dados relacional é matematicamente perfeito, completo e consistente internamente, sendo muito mais utilizado comercialmente, podendo resolver muitos problemas. Um banco de dados relacional é um banco de dados que modela seus dados de uma forma que sejam percebidos pelo usuário como tabelas, ou mais formalmente como relações.

Segundo Neves (2002), o banco de dados orientado a objetos oferece modelos diretos e intuitivos para desenvolvimento de aplicações, é um banco de dados onde as informações são armazenadas na forma de objetos, que atualmente são muito utilizados nas linguagens de programação.

4.4.2 PostgreSQL

Segundo Pgdocptbr (2015), o PostgreSQL é um SGBDOR (Sistema de Gerenciamento de Banco de Dados Objeto Relacional) baseado no POSTGRES Versão 4.2, desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. O POSTGRES foi pioneiro em vários conceitos que somente se tornaram disponível muito mais tarde em alguns sistemas de banco de dados comerciais.

Segundo Neto (2003), o SQL (*Structured Query Language* ou Linguagem de Consulta Estruturada) foi criado no início dos anos 80, pela IBM (*International Business Machines*), e é utilizada para acessar dados em um sistema gerenciador de banco de dados relacional.

Em meados de 1980, o ANSI (*American National Standards Institute* ou Instituto Nacional Americano de Padrões) iniciou seu desenvolvimento para a padronização de uma linguagem para banco de dados relacionais. A primeira publicação de padronização do SQL se deu pelo ANZI e pela ISO (*International Organization for Standardization* ou Organização Internacional de Padrões) em 1986 (NETO, 2003).

O PostgreSQL passou a ser desenvolvido por volta de 1986, na Universidade Norte-Americana da Califórnia, a Universidade de Berkeley. Desde então o PostgreSQL vem sendo melhorado com a adição de funcionalidades e muitos outros benefícios (NETO, 2003).

Só por volta de 1995 o PostgreSQL passou a ter um interpretador SQL, em um trabalho realizado pelos profissionais Andrew Yu e Jolly Chen. Com isso teve um aumento de 25% de seu código fonte, mas mais do que apenas aumento no código fonte, o PostgreSQL passou a se igualar a seus concorrentes, tendo também um interpretador SQL (NETO, 2003).

Em meados de 1996 o PostgreSQL passou a ser um código aberto. E isso o ajudou a ter várias agregações com o objetivo de deixa-lo totalmente competitivo em nível funcional, com os outros disponíveis no mercado. A Partir daí o PostgreSQL foi ganhando espaço, e sendo utilizado em grandes escalas em sistemas web no mundo todo (NETO, 2003).

Segundo Pgdocptbr (2015), o PostgreSQL é um descendente do código fonte aberto, o código original de Berkeley, que suporta grande parte do padrão SQL e oferece muitas funcionalidades modernas, como:

- Comandos complexos;
- Chaves estrangeiras;
- Gatilhos;
- Visões;
- Integridade transacional;

- Controle de simulação multiversão.

Além disso, o PostgreSQL pode ser ampliado pelo usuário de muitas maneiras como, por exemplo, adicionar novos:

- Tipos de dados;
- Funções;
- Operadores;
- Funções de agregação;
- Métodos de índice;
- Linguagens procedurais.

Devido à sua licença liberal, o PostgreSQL pode ser utilizado, modificado e distribuído por qualquer pessoa para qualquer finalidade, seja particular, comercial ou acadêmica, livre de encargos (Pgdocptbr, 2015).

4.4.3 Recursos do PostgreSQL

Segundo Souza, Amaral e Lizardo (2011), o PostgreSQL se destaca por oferecer uma grande variedade de recursos comparado com sistemas similares com o MySQL, sendo inclusive comparado com o Oracle, um sistema pago de grande porte.

Ele possui muitas características de sistemas modernos, como: consultas complexas, *triggres* (disparadores), transações e controle de concorrência (SOUZA; AMARAL; LIZARDO, 2011).

O PostgreSQL implementa algumas características da Orientação a Objetos, como por exemplo, a herança. Esta característica faz com que ao criar uma tabela filha, essa herde as características da tabela pai (NETO, 2003).

Abaixo apresenta-se um exemplo de uso de herança com o PostgreSQL, temos a criação da tabela animais com os campos: id, nome, peso_medio, pais_origem e cor, na sequência a criação da tabela de mamíferos, que possui os campos: id e tipo. Como a tabela mamíferos utiliza o comando "*INHERITS* (animais);" no momento da criação, ela irá herdar os campos da tabela animais, ficando assim com: id, nome, peso_medio, pais_origem, cor e tipo. O campo id foi implementado na tabela filha (mamíferos), então fica sendo o id local e não o herdado da tabela pai (animais) (NETO, 2003).

Figura 10: Exemplo de herança com o PostgreSQL.

```
CREATE TABLE animais (
        id SERIAL NOT NULL,
        nome TEXT,
        peso_medio REAL,
        pais_origem TEXT,
        cor TEXT,
        CONSTRAINT id_animais PRIMARY KEY (id)
);

CREATE TABLE mamiferos (
        id SERIAL NOT NULL,
        tipo Text,
        CONSTRAINT id_mamiferos PRIMARY KEY (id)
) INHERITS (animais);
```

Fonte: O autor.

4.4.4 Porque usar o PostgreSQL

Segundo Neves (2002), o PostgreSQL é um dos melhores bancos de dados para sistema operacional LINUX, tanto para aplicações simples quanto para aplicações de maior complexidade com uma excepcional robustez, estando à altura de outros bancos de dados como Oracle, Sybase ou o Interbase.

Abaixo apresenta-se algumas comparações entre MySQL e PostgreSQL:

Tabela 1: MySQL X PostgreSQL.

	MySQL	PostgreSQL
Valida *user/password*	Sim	Sim
Segurança contra falhas	Não	Sim
Chaves primária/estrangeira	Não	Sim
Confirma ou cancelas operações realizadas	Não	Sim

Bloqueia apenas linhas a serem utilizadas	Não	Sim
Triggers	Sim	Sim
Custo	Pode ser cobrado se vier junto com algum pacote de alguma aplicação.	0

Fonte: Neves (2002).

Outra comparação foi publicada pelo "Linux in Brazil", para o artigo "Banco de Dados" em abril de 2002.

Tabela 2: MySQL X PostgreSQL.

	MySQL	PostgreSQL
Custo	0/3000	0
Transações	Não	Sim
Lock de Linha	Não	Sim
Constraints	Não	Parcial
Program	Parcial	Sim
Senhas	Sim	Sim
Failsafe	Não	Sim
Hotback	Não	Não

Fonte: Neves (2002).

- **Custo:** Se constarem dois preços, indica que o produto é cobrado apenas se for redistribuído como parte de outra aplicação;
- **Transações:** *Commit*, *rollback*, níveis de isolamento;
- ***Contraints*:** Chave primária e chave estrangeira, com capacidade de habilitar/desabilitar ou dropar/adicionar;
- **Programável:** Criar *triggers*, extensões da linguagem SQL;
- **Segurança:** Requer validação de usuário e senha;

- ***Fail safe*:** Após excluir 100 de 100000 linhas, realizar *commit*, desligar o sistema sem *shutdown*, e após religar deve haver as 999900 linhas corretas;
- ***Hot backup:*** Verificar se é possível realizar um *backup* consistente da base de dados enquanto ela está no ar e transações estiverem sendo executadas.

Um outro fato importante descrito neste artigo é que o Oracle tem mais recursos que os outros bancos de dados como: PostgreSQL, MySQL e Informix, mas vem seguido pelo PostgreSQL. Em comparativo de performance-on-line, o Oracle e o PostgreSQL aparecem como os que possuem mais recursos (NEVES, 2002).

Em questão de desempenho em operações on-line o MySQL é mais rápido e mais escalável. Oracle e PostgreSQL são parecidos em escalabilidade, mas o PostgreSQL começa a sentir a carga um pouco antes do Oracle (Neves, 2002).

- **Inconvenientes do PostgreSQL**
  - Consome mais recurso que o MySQL;
  - É mais lento;
  - Tem menos funções PHP.

Apesar de que alguns autores discordam (Neves, 2002).

Segundo Neves (2002), pode-se concluir que o PostgreSQL é recomendado para sistemas mais sérios e robustos, que a consistência dos dados seja fundamental, para isso o PostgreSQL é mais confiável e seguro. Já o MySQL seria recomendado para aplicações que necessitassem de velocidade e gerenciamento de acessos concorrentes, mas a segurança não fosse tão importante.

Com os dados salvos no banco, é hora de resgata-los para a sua visualização, para isso será utilizado o PHP, mas antes apresenta-se o HTML, que é o que realmente o usuário final tem maior contato.

4.5 HTML

O HTML abreviatura de *Hyper Text Markup Language*, é uma linguagem de marcação para documentos eletrônicos, e é projetada para ser repassada ao longo da internet (GRAHAM, 1998).

O HTML define um conjunto de estilos em comum para páginas *Web* como, por exemplo, cabeçalhos, parágrafos, listas, e tabelas. Essa linguagem de marcação define

também estilos de caracter, como negrito, e exemplos de código. Cada elemento tem um nome e está contido no que é chamado de *tag*. Quando cria-se uma página *Web* em HTML, atribui-se um rótulo para cada um dos elementos da sua página através dessas *tags*, que indicam “isso é um cabeçalho” ou “isso é um item de uma lista”. É como se alguém estivesse trabalhando para um jornal ou revista em que apenas escrevesse artigos, mas outra pessoa cuidasse do *layout*. Essa pessoa poderia explicar ao programador visual que determinada linha é o título, é a legenda de uma figura ou é um cabeçalho. O mesmo ocorre com o HTML (LEMAY, 1998).

O HTML se baseia no SGML (*Standar Generalized Markup Language* ou Linguagem Geral de Marcação Padrão), que faz parte de um sistema de processamento de documentos bem maior. Por ser uma herança da SGML, o HTML é uma linguagem que descreve a estrutura de um documento, e não sua apresentação real. A ideia é de que muitos documentos possuem elementos em comum, como por exemplo, parágrafos, títulos, campos ou listas. Assim quando for criado um documento, pode-se usar essa marcação para dar nomes apropriados a esses elementos (LEMAY, 1998).

Criar um documento com linguagem de marcação significa, usar *tags* dentro do texto, entre as palavras, essas *tags* de marcação são semelhantes às de outras linguagens de marcação, como a LaTeX e a troff. Essas *tags* produzem diferentes efeitos nos navegadores, elas já são pré-definidas e não podem ser adicionadas novas opções. Além disso, cada navegador pode interpreta as *tags* de marcação de maneira diferente (LEMAY, 1998).

Abaixo veremos um exemplo de utilização de HTML para título, parágrafo, negrito e uma tabela.

Figura 11: Exemplo de HTML.

```
<h1>título</h1>

<p>paragrafo</p>

<b>negrito</b>

<table>
    <tr>
        <td>Célula A</td>
        <td>Célula B</td>
    </tr>
    <tr>
        <td>Célula C</td>
        <td>Célula D</td>
    </tr>
</table>
```

Fonte: O autor.

4.5.1 HTML 5

Segundo Tecmundo (2015), o HTML 5 conta com algumas melhorias em relação as versões anteriores do HTML, dentre elas a principal diferença notada pelo usuário final, talvez seja a de não precisar instalar *plug-ins* para assistir vídeos de diferentes formatos, assim como para exibir outros elementos que necessitariam desses mesmos *plug-ins*, isso é um grande benefício, já que alguns celulares mini não permitem a instalação de *plug-ins*. Para os desenvolvedores, o HTML 5 deixa algumas tarefas mais simples e oferecem novas maneiras de fazer o que já estava feito.

O HTML 5, é uma tentativa de padronizar a maneira de como os navegadores interpretam as informações recebidas. Mas isso pode gerar algumas distorções, já que cada navegador interpreta os elementos de uma forma diferente (TECMUNDO, 2015).

O Tecmundo (2015), descreve algumas das novas *tags* do HTML 5:

- ***Tags Canvas*:** Especializadas em carregar imagens em bitmap, as *tags canvas* serão especificas para edição breve de imagens através de API's (Interface de Programação de Aplicativos) e JavaScript.

- ***Tags* de Vídeo:** Com ela será muito mais fácil incorporar vídeos às páginas, basta inserir a *tag* de vídeo como se fosse uma imagem. Nela será utilizada a mesma forma que *tag* "<*img*>", devendo ser informado o *src* (fonte), que é o caminho do arquivo de vídeo, e também pode-se informar os parâmetros, *width* (largura) e *height* (altura), que correspondem às dimensões da apresentação do vídeo.

Geolocalização: Este recurso de geolocalização se chama "*geottaging*", e permite saber qual é a localização do usuário. Essa localização não é importante apenas para o usuário ou pessoas da família, também é muito importantes para sites, que podem direcionar produtos ou informações de acordo com o local específico onde o usuário se localiza.

Tendo conhecimento da forma de apresentação do HTML, agora apresenta-se o PHP, que responsabiliza-se pelo resgate dos dados do banco, isso com o auxílio do PostgreSQL para depois gerar a página HTML dinâmica.

4.6 PHP

Segundo Php.net (2015), o PHP é uma linguagem de programação de ampla utilização, interpretada, que é especialmente interessante para desenvolvimento para a *web* e pode ser mesclada dentro do código HTML. A sintaxe da linguagem lembra a do C, Java e Perl, sendo ela fácil de se aprender. O objetivo principal da linguagem é permitir à desenvolvedores criar páginas de forma dinâmica e rápida, mas pode-se fazer muito mais com o PHP.

Castagnetto et al. (2001), afirma que o PHP é uma linguagem embutida em um documento HTML para proporcionar a capacidade de gerar instruções especificas. É possível transformar um site num aplicativo *web* e não apenas numa coleção de páginas estáticas que não podem ser alteradas com frequência.

O que distingue o PHP de linguagens como o JavaScript no lado do cliente, é que o código é executado no servidor, gerando o HTML que é então enviado para o cliente. O cliente recebe os resultados da execução desse *script*, mas não tem acesso ao código fonte. Pode-se inclusive configurar servidores *web* para processar todos os arquivos HTML com o PHP, e então não haverá realmente nenhum modo dos usuários descobrirem se é usada essa linguagem ou não (Php.net, 2015).

A melhor parte em usar o PHP, é que ele é extremamente simples para um iniciante, mas oferece muitos recursos para um programador profissional, por exemplo, dispõe de uma

imensa lista de funções. Apesar do desenvolvimento do PHP ser focado nos *scripts* do lado do servidor, é possível fazer muito mais com ele (Php.net, 2015).

4.6.1 História do PHP

Segundo Castagnetto et al. (2001), em 1994 quando Rasmus Lerdorf juntou alguns *scripts* Perl para monitorar quem estaria espionando seu resumo. Pouco a pouco as pessoas foram ficando interessadas nos *scripts*, que mais tarde foram lançados em um pacote de ferramentas chamado de *Page Home Page* (primeiro significado de PHP). Por volta de 1995 surgiu o PHP/FI ou PHP2, onde Rasmus Lerdorf criou um sistema de processamento de *scripts* e encorpou outra ferramenta para analisar a vinda do formulário HTML.

Logo essas ferramentas começaram a ser usadas pelas pessoas para desenvolver tarefas mais complicadas e o desenvolvimento passou de uma só pessoa para um grupo de desenvolvedores, que eram encarregados pelo projeto e sua organização. Assim teve início o PHP3. Esse grupo de desenvolvedores (Rasmus Lerdorf, Andi Gutmans, Zeev Suraski, Stig Bakken, Shane Acaraveo e Jim Winstead) aperfeiçoaram e aprimoraram o sistema de processamento de *scripts* e adicionaram uma API simples, que permitiria que outros desenvolvedores pudessem implantar novas funcionalidades à linguagem (CASTAGNETTO et al., 2001).

A versão 4 do PHP chamada de PHP4, se baseia no sistema de processamento de *scripts* Zend. Este sistema de processamento foi projetado do zero, para poder ser encorpados em outros projetos com facilidade. O PHP4 é o primeiro aplicativo usado em processamentos Zend (CASTAGNETTO et al., 2001).

Segundo Maseto (2006), o PHP4 foi lançado em Maio de 2000, além de melhorias na performance o PHP4 incluiu outras características chaves como o suporte para vários servidores *Web*, HTTP (*Hypertext Transfer Protocol*), *buffer* de saída, além de maneiras mais seguras de manipular *inputs* de usuários e muitas outras construções novas. Implementou principalmente as características de orientação a objetos.

O PHP5 tem como uma das principais características a implantação de orientação a objetos em seu projeto. Com isso o PHP passou a ser visto por programadores com outros olhos, na versão 4 o PHP não suportava todas as características de orientação a objetos, mas com a reescrita do núcleo da linguagem isso mudou (AMSETO, 2006).

O PHP6 não existe de fato, os livros que foram lançados com o nome de PHP6, na verdade referiam-se à versão 5.6. O PHP 7 será lançado em Junho de 2015 (PHP PR, 2014).

O PHP7 contará com muitas novidades em relação as versões anteriores como, por exemplo, seu desempenho em comparação com o PHP5, que deve aumentar cerca de 10 à 25 %, também contará com novas palavras reservadas além de remover os construtores do PHP4. Será adicionado o operador “Espaço-nave” <=>, que será o operador idêntico. Essas são apenas algumas das várias melhorias do PHP para a versão 7 (WILMS, 2015).

4.6.2 Tipos de dados PHP

Segundo Tipos (2015), o PHP possui oito tipos de dados nativos, os quais estão descritos abaixo:

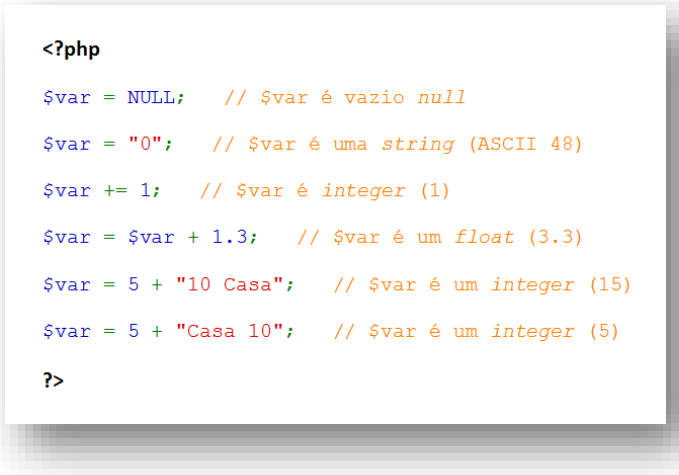
- **Tipos escalares:**
  - *boolean* (apenas *true*/*false* ou verdadeiro/falso);
  - *integer* (números inteiros);
  - *float* (números reais, também pode ser *double*);
  - *string* (série de caracteres).

- **Tipos compostos:**
  - *array* (armazena vários tipos de dados, com a estrutura de chave valor);
  - *object* (é a instancia de uma classe, possui atributos semelhante a um *array*).

- **Tipos especiais:**
  - *resource* (é uma referência a um recurso externo);
  - *NULL* (representa que uma variável não possui valor).

O PHP não requer e nem suporta a definição de tipo explicita na declaração da variável, o tipo da variável vai depender do tipo do valor atribuído a ela. Se uma variável receber um valor inteiro, ela assumirá como tipo *integer*, porém se logo em seguida receber um texto, ela será do tipo *string*, e assim ela irá variando de acordo com o tipo do valor (MANIPULAÇÃO... 2015).

Figura 12: Exemplo de tipos de dados PHP.



```php
<?php
$var = NULL;   // $var é vazio null
$var = "0";   // $var é uma string (ASCII 48)
$var += 1;   // $var é integer (1)
$var = $var + 1.3;   // $var é um float (3.3)
$var = 5 + "10 Casa";   // $var é um integer (15)
$var = 5 + "Casa 10";   // $var é um integer (5)
?>
```

Fonte: O autor.

4.6.3 PHP Orientado a Objetos

Ao trabalhar com orientação a objetos é necessário entender o conceito de classes e objetos, uma classe é uma estrutura que define um tipo de dados, podendo ter atributos (variáveis) e também funções (métodos), que manipulam esses atributos (DALLOGLIO, 2007).

Um objeto possui a mesma estrutura e propriedades de uma classe, sua estrutura é dinâmica e seus atributos podem mudar de valor durante a execução. Vários objetos derivados de uma classe podem ser declarados (DALLOGLIO, 2007).

Para instanciar um objeto, deve-se fazer o uso do operador *new*, seguido do nome da classe, por exemplo, para a classe Produto seria "*new* Produto();". Um objeto não foi feito para ser acessado diretamente, assim para obter os valores dos atributos devemos usar suas propriedades.

Figura 13: Exemplo de classe PHP.

```php
class Produto {

    private $preco;
    private $codigo;
    private $descricao;

    function getCodigo() {
        return $this->codigo;
    }

    function getPreco() {
        return $this->preco;
    }

    function getDescricao() {
        return $this->descricao;
    }

    function setCodigo($codigo) {
        $this->codigo = $codigo;
    }

    function setPreco($preco) {
        $this->preco = $preco;
    }

    function setDescricao($descricao) {
        $this->descricao = $descricao;
    }
}
```

Fonte: O autor.

Figura 14: Exemplo de instância de uma classe PHP.

```php
// instancia o objeto
$produto = new Produto();

// insere valores no objeto
$produto->setCodigo(1);
$produto->setPreco(10.99);
$produto->setDescricao("Café");

// imprime os valores do objeto
echo $produto->getCodigo();
echo $produto->getPreco();
echo $produto->getDescricao();
```

Fonte: O autor.

4.6.4 Potencialidades do PHP

Segundo Php.net (2015), o PHP é focado principalmente nos *scripts* do lado do servidor, portanto, pode-se fazer qualquer coisa que outro programa CGI (*Common Gateway Interface*) possa fazer, por exemplo, coletar dados de formulários, gerar páginas com conteúdo dinâmico ou enviar e receber *cookies*. Essas são apenas algumas das funcionalidades do PHP.

O PHP pode ser utilizado na maioria dos sistemas operacionais da atualidade, por exemplo, Microsoft Windows, Mac OS X, Linux e suas várias variações, e com certeza muitos outros (Php.net, 2015).

O PHP possibilita a liberdade de escolha entre vários sistemas e servidores *web*. Do mesmo modo pode-se escolher entre utilizar programação estruturada, programação orientada a objetos, ou até mesmo as duas juntas (Php.net, 2015).

O PHP não limita-se apenas a gerar HTML, suas habilidades incluem geração de imagens, arquivos PDF, entre outros. Pode-se facilmente criar qualquer padrão de texto, como XHTML (*eXtensible Hypertext Markup Language*) ou XML (*eXtensible Markup Language*). Uma das características mais fortes e mais significativas do PHP é seu suporte a uma ampla variedade de banco de dados. Desenvolver uma página web consultando um banco de dados é incrivelmente simples usando uma das extensões específicas do PHP (Php.net, 2015).

4.6.5 Vantagens do uso do PHP

Segundo Souza (2006), as principais vantagens do uso do PHP são:

- Licença gratuita;
- Plataforma gratuita para se rodar ele (Linux);
- Velocidade de processamento ótima;
- Eficiência de processamento ótima;
- Métodos de segurança ótimos;
- Roda em qualquer tipo de plataforma;
- Código fonte livre;
- *Exceptions* (Exceções para controle de fluxo) ótimas;
- Orientação a objetos ótima;
- É a linguagem *web* mais popular e que mais cresce;

- Possibilita a utilização dos maiores e mais utilizados bancos de dados no mercado sem necessitar de configuração externa;
- Está sempre em atualização e tendo corrigidas falhas e adicionados novos recursos;
- É estável;
- Exige e consome pouco recurso de hardware do servidor;
- Flexibilidade ótima;
- Componentes nativos, não dependendo de componentes externos para algumas funcionalidades básicas;
- Documentação, controle e reportamento de erros ótimos;
- Comunidade de desenvolvimento muito participativa e prestativa;
- Planos de hospedagem *web* baratos e sem nenhum custo extra para a utilização do MySQL em conjunto com o PHP;
- A programação em PHP é eficiente;
- O Apache (servidor *web* utilizado para rodar o PHP) é bem seguro.

Dentro da página HTML será inserido um mapa para apresentar as posições dos veículos, com isso dispõem-se de várias opções de mapas digitais, para continuidade do trabalho, foi feito o direcionamento à API do Google Maps, por ser de fácil acesso, oferecer um pacote gratuito à aplicações não comercializadas, e pela ampla disponibilidade de funções.

4.7 Mapas Digitais

Para a exibição das informações dos veículos, será usado um mapa digital, os mais conhecidos são o Google Maps (mapa da Google), o Bing Maps (mapa da Microsoft) e o OpenStreetMap (mapa colaborativo gratuito). O projeto é direcionado para o uso do Google Maps. A seguir apresenta-se mais informações sobre essa API.

4.7.1 Google Maps

Segundo Introdução... (2015), a API do Google Maps é um serviço público e qualquer pessoa pode usar, sendo apenas cobrado em caso do sistema onde o mapa é utilizado for um sistema remunerado, caso contrário existe a versão livre. O Google Maps possui várias API’s, que podem ser incorporadas em várias aplicações, abaixo apresenta-se uma tabela das API’s listadas no site oficial do Google Maps.

Tabela 3: Tabela de API's e descrições do Google Maps.

API	Descrição
Google Maps JavaScript API	Incorpore um mapa do Google em sua página da web usando JavaScript. Manipule o mapa e adicione conteúdo com a ajuda de vários serviços.
Google Maps API for Flash	Use essa API ActionScript para incorporar um mapa do Google na sua página da web ou aplicativo baseado em Flash. Manipule o mapa em três dimensões e adicione conteúdo com a ajuda de vários serviços.
Google Earth API	Incorpore um verdadeiro globo digital em 3D à sua página da web. Leve os seus visitantes a qualquer lugar da Terra (até mesmo nas profundezas dos oceanos) sem tirá-los de sua página da web.
Google Static Maps API	Incorpore uma imagem simples e rápida do Google Maps em sua página da web ou site para celular sem precisar de códigos JavaScript ou qualquer carregamento dinâmico de página.
Serviços da web	Use solicitações de URL para acessar informações de geocodificação, rotas, elevação e lugares dos aplicativos cliente e manipule os resultados em JSON ou XML.
Google Maps Data API	Visualize, armazene e atualize dados de mapa por meio de feeds da Google Data API, usado um modelo de elementos (marcadores, linhas e formas) e coleções de elementos.

Fonte: http://www.devmedia.com.br/introducao-a-google-maps-api/26967.

4.7.2 Potencialidades do Google Maps API

Com a API do Google Maps é possível desenvolver muitas aplicações, sendo assim muito difícil descrever todas aqui. Abaixo demonstra-se algumas dessas possibilidades que essa API fornece:

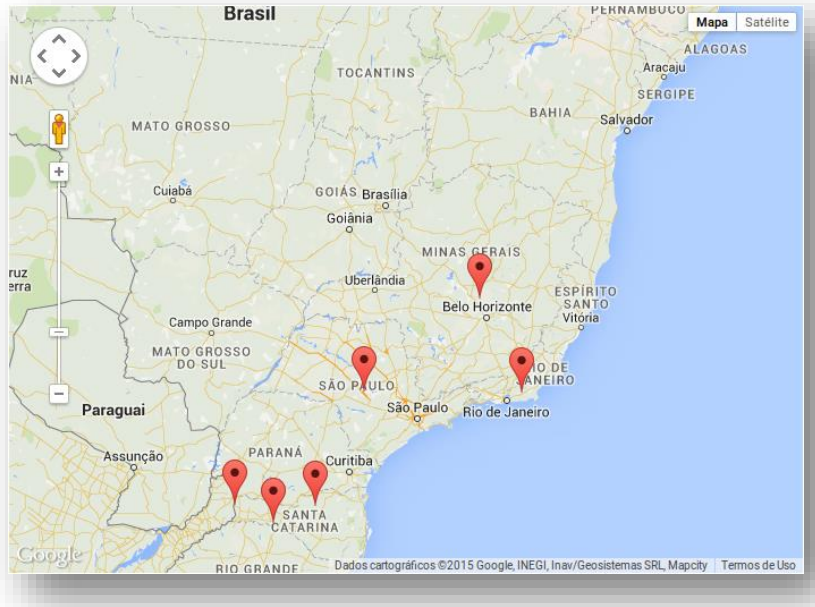
- **Desenhar o mapa:** A figura 15 mostra como é o mapa da API de Google, apenas o mapa sem nenhum elemento adicionado.

Figura 15: Mapa simples.

Fonte: O autor.

- **Inserir marcadores para os pontos:** A figura 16 mostra o mapa com marcadores padrões da API do Google. Esses pontos são usados para marcar um local onde tem algo importante, que pode ser uma casa, um local de trabalho, um veículo, etc.

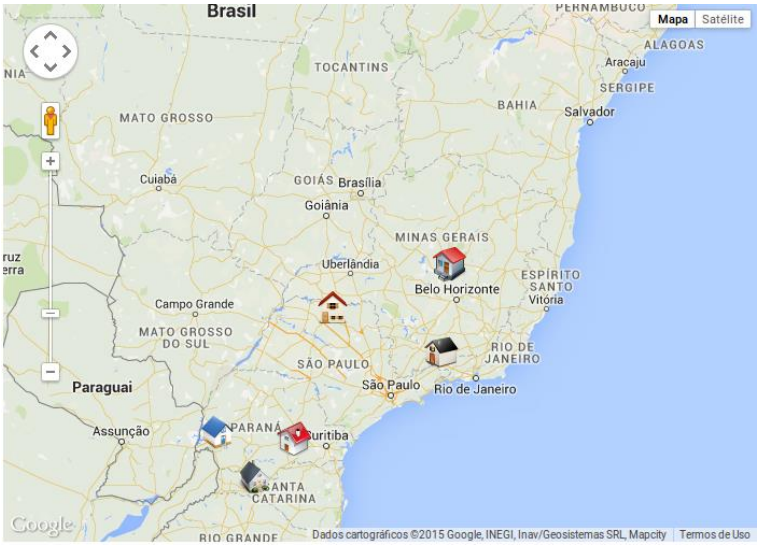
Figura 16: Mapa com marcadores.



Fonte: O autor.

- **Personalizar estes marcadores:** A figura 17 mostra o mapa com marcadores personalizados, os marcadores ou pontos, podem ser personalizados com imagens, para isso é só adicionar um parâmetro ao marcador com a imagem desejada.

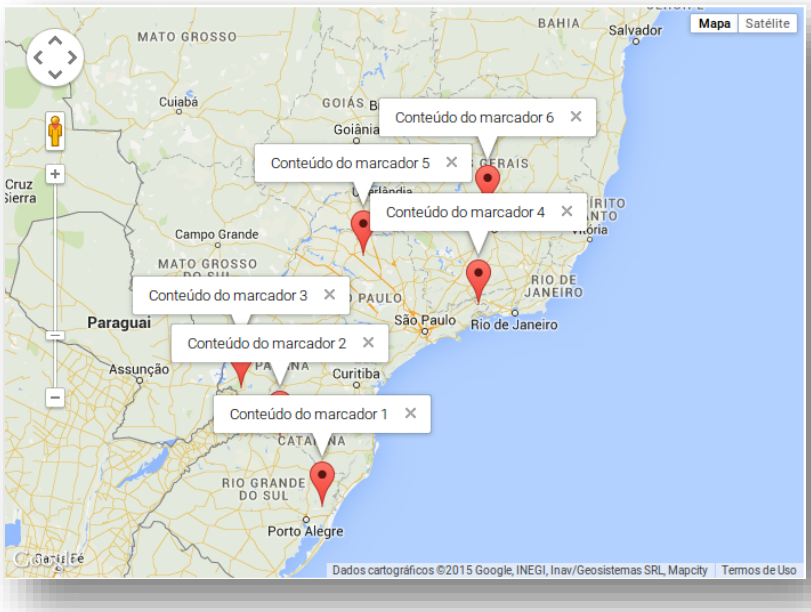
Figura 17: Mapa com marcadores personalizados.



Fonte: O autor.

- **Inserir caixa de informações para um marcador:** A figura 18 mostra os marcadores com caixas de informações, essas informações podem ser variadas, dependendo de quem as for manipular. É um espaço que pode-se inserir as informações necessárias para o marcador.

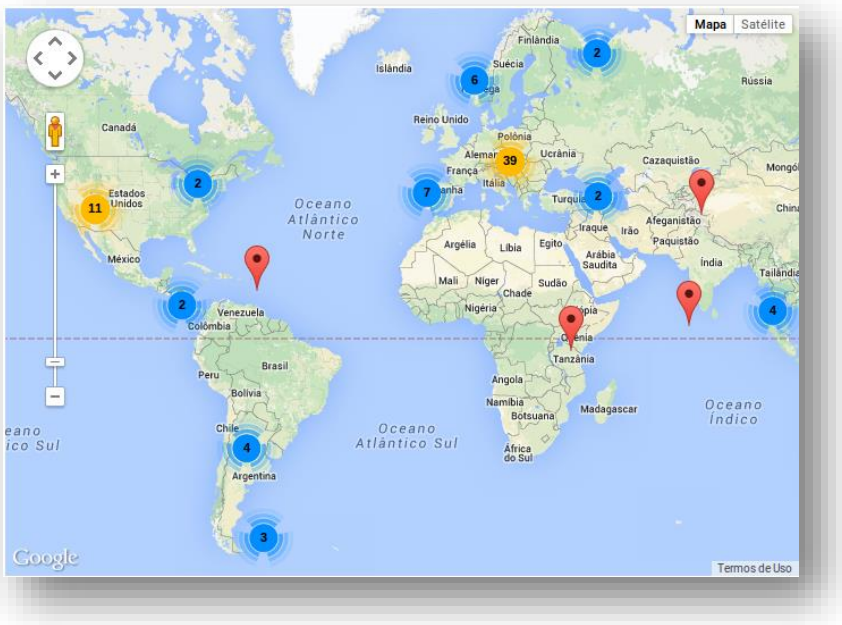
Figura 18: Marcadores com caixas de informação.



Fonte: O autor.

- **Personalizar as caixas de informações:** As caixas de informações tem suporte a HTML, com isso elas podem ser bem dinâmicas, inserindo as *tags* de marcação, é possível inserir parágrafos, tabelas, imagens, textos em negrito, entre outras personalizações que podem ser feitas.
- **Adicionar identificador único para caixas de informação:** Cada caixa de informação pode ter um marcador único, podendo assim ser diferenciada das demais, isso é mais útil a nível de programação, já que ao usuário final isso não será visível.
- **Agrupar pontos próximos:** A figura 19 mostra o mapa com os ícones mais próximos agrupados, isso acontece quando o zoom do mapa diminui, para evitar a sobreposição de um marcador com o outro. Quando o zoom é aproximado, eles se separam novamente para ficarem visíveis na sua posição original.

Figura 19: Marcadores agrupados.

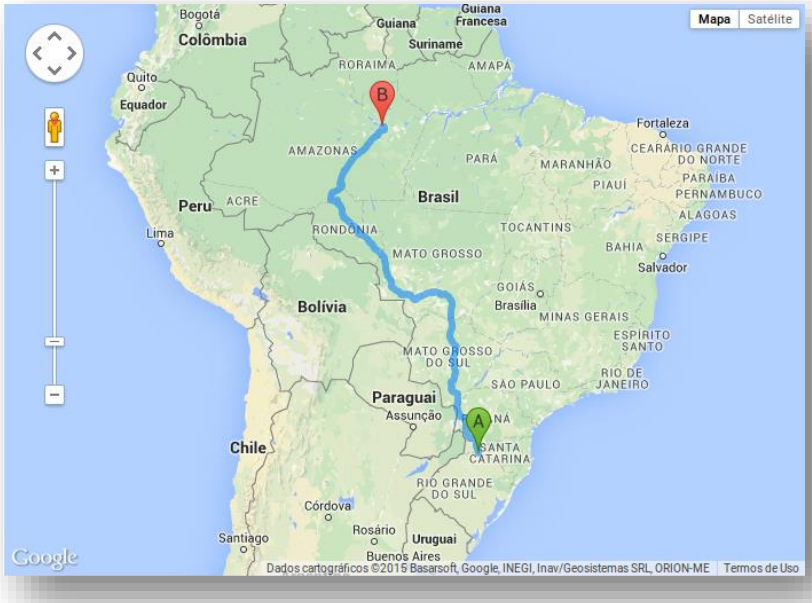


Fonte: O autor.

- **Criar rotas entre pontos:** A figura 20 mostra uma rota entre o ponto “A” e o ponto “B”, neste exemplo, a origem é Chapecó e o destino é Manaus. Para gerar uma rota é lavado em consideração o caminho mais curto entre os pontos. Além da distância, é possível escolher o meio de transporte que será usado para percorrer,

por exemplo, carro, ônibus ou a pé. De acordo com o meio de transporte escolhido será estimado o tempo para percorrer a rota.

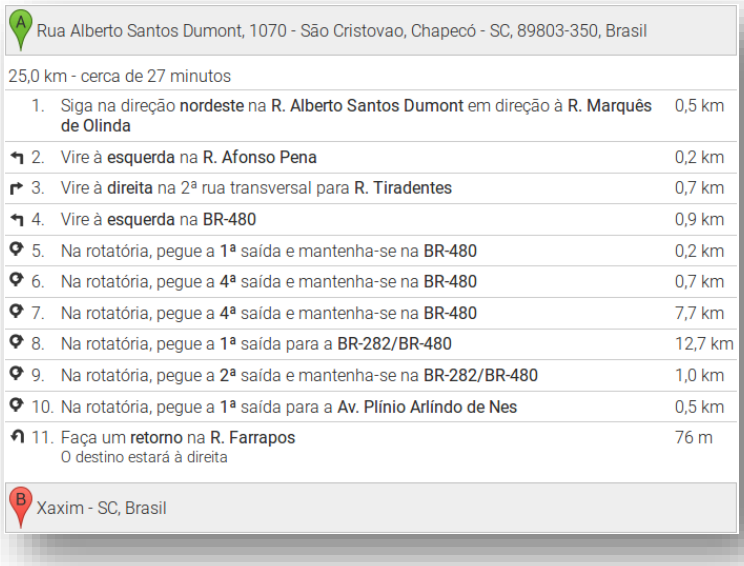
Figura 20: Mapa com rota entre dois pontos.



Fonte: O autor.

- **Calcular tempo para percorrer uma rota:** A figura 21 mostra as informações detalhadas de qual caminho seguir em uma rota, por exemplo, seguir 200 metros na rua de origem, dobrar a direita e seguir 800 metros, dobrar a esquerda e assim por diante até chegar ao local de destino.

Figura 21: Informações da rota.



Fonte: O autor.

- **Inserir figuras geométricas como círculos, triângulos e retângulos:** A figura 22 mostra um marcador com um círculo em seu entorno, mas poderia ser outra figura geométrica, como um retângulo ou uma área gerada entre dois ou mais marcadores, essas áreas ou figuras geométricas podem ser usadas para demarcar uma região especifica do mapa, podendo ser uma propriedade, uma área restrita, ou ainda ter outras finalidades.

Figura 22: Mapa com círculo em torno de um marcador.

Fonte: O autor.

- **Visualizar no formato de mapa ou na visão de imagens de satélite:** A figura 23 mostra o mapa com a visão por satélite, essa é uma opção que permite visualizar como é a aparência da superfície terrestre vista por fotografias tiradas com satélites, essas fotos são atualizadas constantemente, para evitar que o mapa fique muito desatualizado.

Figura 23: Mapa com visão por satélite.

Fonte: O autor.

- **Dar zoom:** O zoom é usado para enquadrar melhor o ponto do mapa desejado, ele pode ser ajustado manualmente pelo usuário ou pode ser automático, onde o sistema já aproxima ou distancia, o zoom automático pode ser percebido, por exemplo, quando é criada uma rota, onde o mapa aumenta ou diminui o zoom para enquadrar toda essa rota na tela, de forma a ocupar o máximo de espaço.

Figura 24: Mapa com zoom

5 PROCEDIMENTOS METODOLÓGICOS

A qualificação do projeto segundo a natureza é uma pesquisa aplicada, com objetivo de gerar conhecimentos para aplicação prática dirigidos a soluções de problemas específicos.

A forma de abordagem a ser utilizada no projeto será qualitativa, onde não requer o uso de métodos e técnicas estatísticas. O ambiente natural é a fonte direta para coleta de dados e o pesquisador é o instrumento-chave.

Os procedimentos técnicos utilizados no projeto são: pesquisa bibliográfica, que elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e atualmente com matérias disponibilizados ne internet, e pesquisa ação, que é concebida e realizada com uma estreita associação com uma ação ou com a resolução de um problema coletivo.

Para atingir os objetivos a pesquisa se caracteriza por explicativa, que visa identificar os fatores que determinam ou contribuem para a ocorrência dos fenômenos.

6 CRONOGRAMA

Tabela 4: Cronograma da Monografia I

Atividades/Mês	Jan	Fev	Mar	Abr	Mai	Jun
Levantamento do problema	**X**					
Estudo do tema		**X**	**X**			
Desenvolvimento dos objetivos		**X**	**X**			
Questões de pesquisa		**X**	**X**			
Justificativa		**X**	**X**	**X**	**X**	
Levantamento e realização da revisão bibliográfica			**X**	**X**	**X**	
Referências			**X**	**X**	**X**	**X**
Cronograma					**X**	**X**
Orçamento					**X**	**X**
Procedimentos metodológicos						**X**
Entrega Monografia I						**X**
Defesa						**X**
Correções						**X**

Fonte: O autor.

Tabela 5: Cronograma da Monografia II

Atividades/Mês	Jul	Ago	Set	Out	Nov	Dez
Resumo e Abstract	**X**	**X**				
Levantamento de informações	**X**	**X**				
Desenvolvimento e aplicação	**X**	**X**	**X**	**X**	**X**	**X**
Conclusão					**X**	**X**
Entrega Monografia II						**X**
Defesa						**X**
Correções						**X**

Fonte: O autor.

7 ORÇAMENTO

Tabela 6: Orçamento

Itens	Quantidade	Custo Unitário	Custo do Item
Placa Arduino Uno	2	R$ 47,89	R$ 95,78
Placa de Rádio Frequência	2	R$ 37,00	R$ 74.00
Placa GPS	1	R$ 98,50	R$ 98,50
		Custo Total	**R$ 368,28**

Fonte: O autor.

Os custos com livros, artigos, canetas, folhas, impressão e outros custos de desenvolvimento ficam por conta do acadêmico.

8 REFERÊNCIAS

ADUINO. **O que é o Arduino?** Disponível em: <http://www.arduino.cc/>. Acesso em: 12 abr. 2015.

CASTAGNETTO, Jesus M. et al. **PHP Programando.** São Paulo - Sp: Markon Books de Brasil, 2001. 770 p.

CONFEDERAÇÃO BRASILEIRA DE AUTOMOBILISMO. **Apresentação.** Disponível em: <http://www.cba.org.br/site/apresentacao.php>. Acesso em: 15 abr. 2015.

DALL'OGLIO, Pablo. **PHP Programando com Orientação a Objetos:** Orientação a Objetos. São Paulo - Sp: Novatec, 2007. 574 p.

ENCICLOPÉDIA F1. **Telemetria.** Disponível em: <http://www.enciclopediaf1.com.br/por_dentro_da_f1/telemetria>. Acesso em: 29 abr. 2015.

FILIPEFLOP. **O que é Arduino?** 2014. Disponível em: <http://blog.filipeflop.com/arduino/o-que-e-arduino.html>. Acesso em: 10 maio 2015.

GPSSAT. **Telemetria.** 2012. Disponível em: <http://www.gpssat.com.br/telemetria-completa>. Acesso em: 27 abr. 2015.

GRAHAM, Ian S.. **HTML.** 3. ed. Rio de Janeiro - Rj: Campus, 1998. 640 p.

HEUSER, Carlos Alberto. **Projeto de Banco de Dados.** 4. ed. Poto Alegre: Sagra Luzzatto, 1998. 64 p. Disponível em: <https://jalvesnicacio.files.wordpress.com/2010/03/carlos_alberto_heuser-projeto_de_banco_de_dados.pdf>. Acesso em: 26 abr. 2015.

INTRODUÇÃO a Google Maps API. Disponível em: <http://www.devmedia.com.br/introducao-a-google-maps-api/26967>. Acesso em: 3 maio 2015.

LEMAY, Laura. **Aprenda em 1 semana HTML4.** Rio de Janeiro - Rj: Campus, 1998. 631 p.

MANIPULAÇÃO de Tipos. Disponível em: <http://php.net/manual/pt_BR/language.types.type-juggling.php>. Acesso em: 21 abr. 2015.

MASETO, Jhony Maiki. **ANÁLISE AVALIATIVA ENTRE FRAMEWORKS DE PHP.** 2006. 103 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Ciência da Computação, Universidade Comunitária Regional de Chapecó - UnochapecÓ, Chapecó - Sc, 2006. Disponível em: <http://www5.unochapeco.edu.br/pergamum/biblioteca/php/imagens/0000b5/0000b5d1.pdf>. Acesso em: 16 abr. 2015.

MCROBERTS, Michael. **Arduino Básico:** O que exatamente é um Arduino?. São Paulo - Sp: Novatec, 2011. 453 p.

NEVES, Denise Lemes Fernandes. **PostgreSQL:** Conceitos e Aplicações. Tatuape - Sp: Érica, 2002. 188 p.

O QUE é o Arduino? Disponível em: <http://arduino.cc/>. Acesso em: 12 abr. 2015.

O QUE é o PostgreSQL? Disponível em: <http://pgdocptbr.sourceforge.net/pg80/preface.html#INTRO-WHATIS>. Acesso em: 11 abr. 2015.

PEREIRA NETO, Álvaro. **PostgreSQL.** São Paulo - Sp: Érica, 2003. 884 p.

PGDOCPTBR. **O que é o PostgreSQL?** Disponível em: <http://pgdocptbr.sourceforge.net/pg80/preface.html#INTRO-WHATIS>. Acesso em: 11 abr. 2015.

PHP PR. **PHP 6? Que nada, a próxima versão é a 7!** 2014. Disponível em: <http://www.phppr.net/php-6-que-nada-proxima-versao-e-7/>. Acesso em: 10 maio 2015.

PHP.NET. **O que é o PHP?** Disponível em: <http://php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 11 abr. 2015.

PHP.NET. **O que o PHP pode fazer?** Disponível em: <http://php.net/manual/pt_BR/intro-whatcando.php>. Acesso em: 11 abr. 2015.

SCARIOT, Patrícia. **UM ESTUDO PARA REQUALIFICAÇÃO URBANA.** 2006. 66 f. TCC (Graduação) - Curso de Arquitetura e Urbanismo, Centro TecnolÓgico – Cetec, Universidade ComunitÁria Regional de ChapecÓ – UnochapecÓ, Chapecó – Sc, 2006. Disponível em:

<http://www5.unochapeco.edu.br/pergamum/biblioteca/php/imagens/00007A/00007A69.pdf>. Acesso em: 15 abr. 2015.

SOUZA, Arthur Câmara; AMARAL, Hugo Richard; LIZARDO, Luis Eduardo O.. **PostgreSQL: uma alternativa para sistemas gerenciadores de banco de dados de código aberto.** 2011. 4 f. Tese (Doutorado) - Curso de Ciência da Computação, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais (ufmg), Belo Horizonte - Mg, 2011. Disponível em: <http://www.periodicos.letras.ufmg.br/index.php/ueadsl/article/viewFile/2896/2855>. Acesso em: 12 abr. 2015.

SOUZA, Sandro J. S.. **Vantagens do PHP 5.** 2006. Disponível em: <http://phpbrasil.com/articles/article.php/id/1173>. Acesso em: 16 abr. 2015.

TECMUNDO (Org.). **O que é HTML5?** Disponível em: <http://www.tecmundo.com.br/navegador/2254-o-que-e-html-5-.htm>. Acesso em: 10 maio 2015.

TECNOCONTROL. **O que é TELEMETRIA e no que é aplicada?:** Aplicações da telemetria. Disponível em: <http://tecnocontrol.com.br/blog/o-que-e-telemetria-e-que-e-aplicada/>. Acesso em: 29 abr. 2015.

TELEMETRIA. Disponível em: <http://www.magnetimarelli.com/pt/business_areas/motorsport/excelências-tecnológicas/telemetria>. Acesso em: 27 abr. 2015.

TIPOS: Introdução. Disponível em: <http://php.net/manual/pt_BR/language.types.intro.php>. Acesso em: 21 abr. 2015.

VIVA O LINUX. **O que compõem um sistema de telemetria.** 2004. Disponível em: <http://www.vivaolinux.com.br/artigo/Transmissao-de-dados-via-telemetria-uma-opcao-de-comunicacao-remota?pagina=4>. Acesso em: 24 maio 2015.

WILMS, Alan. **Principais Novidades do PHP 7.** 2015. Disponível em: <https://medium.com/@alanwillms/principais-novidades-do-php-7-6821683fc9a>. Acesso em: 10 maio 2015.