Plataforma de Monitoramento Esportivo

Arthur Simões Gonzaga e Victor Martins N. Luz

***Resumo*—Este projeto visa criar uma plataforma embarcada de monitoramento esportivo, que realize a aquisição de dados corporais e de movimentação do usuário em tempo real, de modo a efetuar uma análise qualitativa do desempenho do atleta alvo.**

***Keywords*—Sistemas Embarcados, Wearable, Antropometria, Raspberry Pi, Linux**

I. Introdução

Na última década, o dinamismo da evolução tecnológica pela qual a sociedade passa deixa de ser representado apenas por seus legados capitais, como o telefone, a internet, as impressoras, e ruma para nichos tidos até então como "indissociados"desse movimento de evolução, seguindo um processo contínuo de difusão. A busca pela otimização de processos e performances ganhou no desenvolvimento tecnológico um forte e versátil aliado, e o mundo esportivo, por exemplo, começa a experimentar o potencial da tecnologia como mais uma ferramenta para o aumento do rendimento dos atletas. Sem deixar de lado os tradicionais cuidados com o treino, condicionamento físico e alimentação, a tecnologia chega ao esporte para aprimorar o trabalho do atleta e equipe técnica, oferecendo novas possibilidades táticas e estratégicas capazes de agregar melhorias ao desempenho, seja qual for a modalidade.

No futebol não é diferente. A partir dos anos 60[1], a preocupação de tirar a sobrecarga de responsabilidades do técnico de futebol e melhorar o rendimento dos atletas resultou na formação das primeirtas comissões técnicas, que no decorrer das décadas viraram padrão para times de alto rendimento. Essas equipes, que de início focavam na figura do preparador físico, passaram a envolver profissionais de diversas áreas, da tática à psicologia, com o objetivo de atender a diversidade de demandas dos atletas e do esporte, contribuindo para o sucesso das equipes nas competições.

Atualmente, soma-se a todo esse trabalho especializado o uso de tecnologias para avaliações mais rápidas e eficazes do desempenho dos atletas e das táticas de equipe, contribuindo de forma mais ostensiva na prevenção de lesões, correção de posicionamentos táticos, avaliação das transições de jogos e revisão do comportamento geral do time. Vale salientar que o uso de meios tecnológicos não está restrito aos times, e hoje tem peso para alterar o resultado de um jogo. A arbitragem, além de dispor de um sistema de comunicação entre os componentes da equipe, tem agora a possibilidade de utilizar recurso de vídeo (e *software* que processa esse imageamento) para definir a ocorrência de um gol. Esse sistema foi utilizado em um jogo entre as equipes da França e da Espanha[2] , em que árbitro utilizou do recurso tecnológico para anular um gol da equipe francesa.

O uso do sistema de posicionamento global para fins de acompanhamento de movimentação localizada é tão recente quanto o uso geral do GPS. Apenas dois anos após a disponibilização da rede (em 1995), os primeiros estudos sobre a precisão do sistema para caracterizar a locomoção humana foram publicados.[3] O primeiro dispositivo comercial a utilizar a tecnologia foi lançado em 2003, e desde então vem sendo introduzido em esportes de campo aberto, como o futebol, o rugby, o hockey e o tênis.[4]

Atualmente, muitos times brasileiros de futebol já introduziram o uso de tecnologia GPS[5] para mapear a movimentação dos atletas durante os treinamentos e jogos, afim de melhor qualificar os treinamentos, proporcionando uma intervenção mais dinâmica e personalizada para o perfil e particularidades de cada jogador.

Figura 1. Utilização de GPS por jogadores de futebol

[1]"A Evolução das Comissões Técnicas no Futebol", Caderno de Campo, 2017. [Online]. Available: https://cadernodecampo.com/2008/09/11/a-evolucao-das-comissoes-tecnicas-no-futebol/. [Accessed: 03- Apr- 2017].

[2]"Vídeo-árbitro "trama"França por duas vezes na derrota frente a Espanha", O Jogo, 2017. [Online]. Available: http://www.ojogo.pt/internacional/noticias/interior/video-arbitro-anula-golo-a-griezmann-no-franca-espanha-5755641.html. [Accessed: 03- Apr- 2017].

[3]R. Aughey, "Applications of GPS Technologies to Field Sports", International Journal of Sports Physiology and Performance, vol. 6, no. 3, pp. 295-310, 2011.

[4]Julen Castellano; David Casamichana, "Deporte con dispositivos de posicionamiento global (GPS): Aplicaciones y limitaciones", Revista de Psicología del Deporte, vol. 23, no. 2, pp. 355-364, 2014.

[5]"Tecnologia em "top"de jogadores de futebol gera brincadeira, mas é coisa séria.", Canaltech, 2017. [Online]. Available: https://canaltech.com.br/noticia/gps/tecnologia-em-top-de-jogadores-de-futebol-gera-brincadeira-mas-e-coisa-seria-61281/. [Accessed: 03- Apr- 2017].

A. Revisão Bibliográfica

O monitoramento esportivo vem ganhando espaço e ainda não é uma tecnologia acessível a todos os atletas do time. Todavia, o tema vem sendo abordado em artigos científicos, sempre relacionados à utilização para medidas antropométricas.

Artigos de referência do assunto são: *Deporte con dispositivos de posicionamiento global (GPS): Aplicaciones y limitaciones*, desenvolvido por Julen Castellano e David Casamichana e; *Applications Of GPS Technologies to Field Sports*, de Robert J. Aughey. Porém, este artigos tratam somente da importância da utilização desta tecnologia no esporte e não na construção de um dispositivo/plataforma de aquisição de dados.

B. Objetivos e Requisitos

Este projeto almeja modelar e construir uma plataforma embarcada, que possa ser utilizada por um jogador de futebol ou qualquer outro esporte a céu aberto, com o intuito de mapear a sua posição, informando a uma terceira pessoa, dados qualitativos sobre o seu posicionamento no campo.

Os requisitos abrangem três etapas de funcionamento do protótipo: a coleta de dados do GPS que deve ser feita, afim de buscar dados sobre o posicionamento; o estabelecimento de uma comunicação com um computador externo, com o propósito de exibição de dados e avaliação em tempo real (a partir da geração de mapas de calor ou estatística de em qual local do campo o atleta mais se posicionou, por exemplo); e por fim, o dimensionamento do consumo energético da plataforma, tendo em vista a preocupação com a eficiência do consumo do dispositivo, que requer um tempo de utilização grande.

Além das estatísticas advindas diretamente do *tracking* GPS, é possível obter outros dados úteis para a avaliação de performance de um atleta, como um levantamento dos dados de velocidade ou aceleração desenvolvidas pelo jogador ao longo do treinamento.

O *hardware* a ser desenvolvido é caracterizado pela integração de sensores, incluídos neste o módulo GPS, e de um computador *Raspberry Pi* para o processamento de dados e gerenciamento da comunicação com o dispositivo em que serão exibidas as informações. Tanto o sensoriamento, quanto o *Rasberry* serão embarcados em um *wearable*, visando a portabilidade do sistema e o conforto do atleta sob monitoramento.

A difusão tecnológica, além de buscar novas áreas de atuação, também se expande atingindo novos públicos. Dispositivos que desempenham esse tipo de monitoramento, utilizados pelos clubes de futebol, envolvem tecnologia e fabricação estrangeiras, fato que agrega custos de tributação e transporte ao já elevado preço do equipamento. O desenvolvimento dessa plataforma visa obter um custo moderado de projeto, pelo qual clubes de pequeno porte ou até mesmo amadores possam pagar para dispor dessa tecnologia, tornando-a mais acessível.

Além disso, o time de futebol da Universidade de Brasília, tem um departamento de análise, o qual pode muito bem utilizar da tecnologia desenvolvida para qualificar suas análises e apontar as difuculdades a serem tratadas.

Em uma procura de como o mercado desta tecnologia se comporta, achou-se uma empresa brasileira, GPS Pro Soccer,

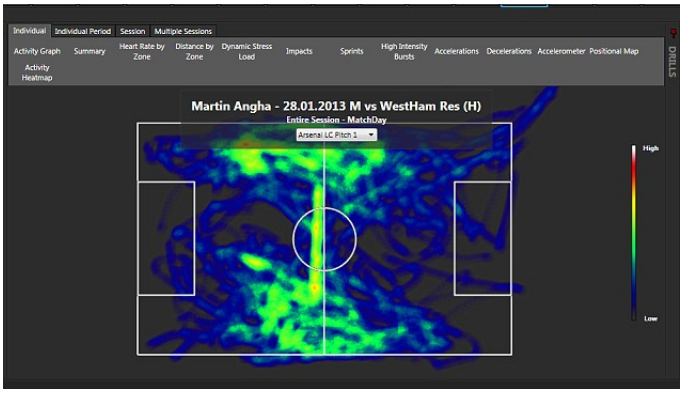

Figura 2. Exemplo de Mapa de Calor

que desenvolve produtos próximos ao qual irá se desenvolver, todavia, o custo dos equipamentos é elevado, ficando em torno de R$ 5000,00[6] para cada monitor, sendo de acesso local, somente. Algumas outras empresas internacionais também trabalham com plataformas parecidas. A SPT desenvolve a solução junto ao colete por um preço de 249,99 dólares[7], porém, o equipamento é importado (este preço não inclui taxas e impostos) e não gera os dados em tempo real, o que torna o nosso produto competitivo no mercado.

II. Modelo de Funcionamento

O *Raspberry Pi*, diferentemente de microcontroladores potencialmente utilizáveis para realizar esse projeto, permite o *multitasking* do *software* nele embarcado. Essa característica é importante para a coordenação de vários subsistemas necessários para solucionar o problema a que o projeto se propõe a lidar. Especificamente para este dispositivo de monitoramento, faz-se necessário gerenciar a comunicação serial com o módulo GPS, responsável por obter as informações de posicionamento, realizar a interpretação desses dados, lidar com conexões de *Internet* e enviar essas informações para um servidor, que, com o banco de dados de posicionamento, é capaz de disponibilizar remotamente as informações coletadas, gerando um ambiente de fácil entendimento para o usuário.

O usuário, através de um circuito de controle, poderá guiar o início e fim do registro de dados e, por consequência, do monitoramento do atleta. Essa ativação do *traking* se dará após a conexão serial *(UART)* entre o *Raspberry PI* e o módulo *GPS NEO-6M*, e a partir do momento que o módulo GPS fixar uma triangulação com os satélites. Os dados enviados pelo módulo são obtidos de *strings* de informações, enviadas a cada segundo para o *Rasberry*. As informações de latitude, longitude, velocidade e altitude são extraídas dessas *strings* e são transformadas em variáveis, tornando possível a realização

[6]P. Sistemas, "GPS PRO SOCCER-GPS Pro Soccer-Sistema de gerenciamento de atletas de futebol", Gpsprosoccer.com.br, 2017. [Online]. Available: http://gpsprosoccer.com.br/gps-pro-soccer/. [Accessed: 04- Apr- 2017].

[7]S. Vest, "SPT Pack (GPS + Vest) – SPT", Sports-performancetracking.com, 2017. [Online]. Available: https://www.sportsperformancetracking.com/product/spt-pack-gps-vest/#w3mIiem20adi2tVW.97. [Accessed: 04- Apr- 2017].

de operações matemáticas com esses valores. Após a discretização desses valores, eles são armazanados em um arquivo, que serve de base de dados para o servidor.

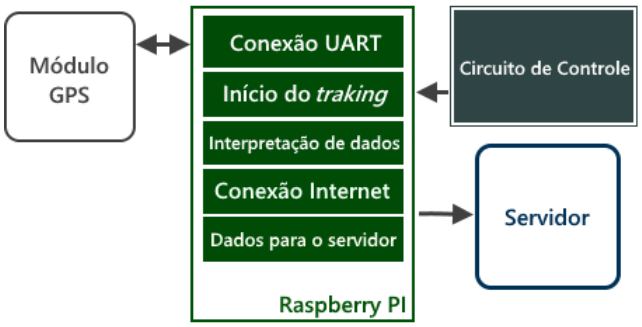

Figura 3. Fluxograma de funcionamento do sistema.

Ao utilizar uma solução já disponível para a interface de servidor, o *Flask*, pode-se tornar o *Raspberry* em um servidor *python*, possibilitando a criação de *backlogs* e o uso de *scripts* em *html*, *css* ou mesmo a própria linguagem *python* para desenvolver as interfaces gráficas. Um dos *scripts html* utlilizados tem seu código fonte disponibilizado gratuitamente pelo *Google* [8], sendo responsável por mapear, por incidência, latitudes e longitudes na plataforma do*Google Maps*, gerando mapas de calor de posicionamento, oferecendo uma interface de fácil compreensão e análise.

Dispondo desse ferramental, foi possivel criar uma página *web* que utiliza dos pacotes *RPi.GPIO* para leitura de valores do *Raspberry*. De modo a testar essa interação entre servidor e *hardware*, Fez-se a leitura de valores nas GPIO do Raspberry com sucesso, sendo essas informações passadas para a interface html, que exibe o valor atual no *site*. Além disso, a partir de um banco de dados de latitudes e longitudes, gerou-se um mapa de calor de posicionamento.

Figura 4. Testes de interface com servidor.

III. Descrição de Hardware

Para tornar operacionais as etapas de funcionamento propostas para o sistema, o equipamento embarcado precisa atuar em pelo menos duas grandes frentes: Obtenção de dados de localização e integração com um servidor, para a disponibilização dos dados registrados.

A obtenção dos parâmetros essenciais para o sucesso da operação do dispositivo proposto é possível com a utilização de um módulo GPS integrado ao *Raspberry*. O módulo *NEO-6M*[9], da empresa suiça *U-blox Holding AG* fornece, via comunicação serial, *strings* no padrão *NMEA 0183 (National Marine Electronics Association)*[10], a partir das quais é possível discretizar as informações de latitude, longitude, velocidade, altitude - entre outras.

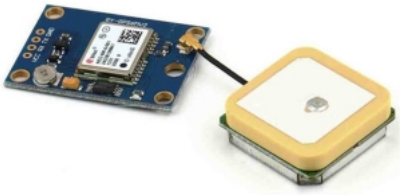

Figura 5. Módulo GPS NEO-6M.

Esse módulo exige entre 3.3V e 5V para a alimentação, e a sua comunicação UART tem nivel lógico alto baseado em 3.3V. Essa operação é ideal para sua integração com os principais microcontroladores e também com o *Raspberry Pi*, dispositivo utilizado como base para o projeto.

Com o objetivo de fornecer maior controle do usuário sobre o sistema, que operaria de forma autônoma a partir de sua ativação, foi inserido um sistema de controle, por meio do qual o usuário operará botões de ativação/desligamento do *tracking*. O botão é ligado a um GPIO do *Raspberry* na configuração de resistor *pull-up*.

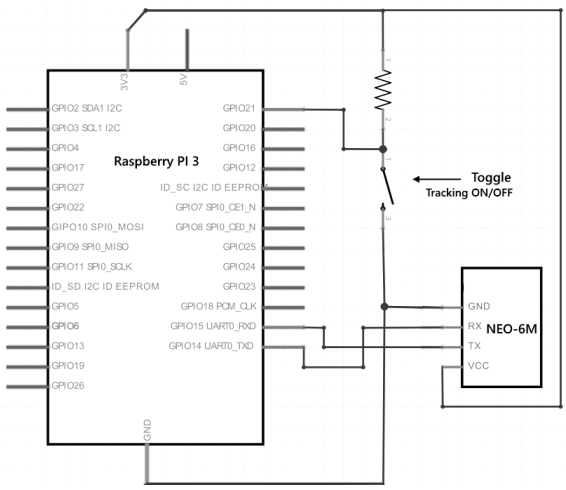

Figura 6. Esquemático inicial do sistema.

[8]"Google Maps API's - Heatmaps", Google, 2017. [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap?hl=pt-br. [Accessed: 06- May- 2017]

[9]"NEO-6 U-blox 6 GPS Modules DataSheet", U-blox, 2011. [Online]. Available: https://www.u-blox.com/sites/default/files/products/documents/NEO-6-DataSheet-%28GPS.G6-HW-09005%29.pdf. [Accessed: 05- May- 2017]

[10]"NMEA data". [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm. [Accessed: 01- Jun- 2017]

IV. Descrição de Software

As soluções por *software* utilizadAs foram bem específicas. Os requisitos de programação levantados para viabilizar o projeto foram:

- Realizar a aquisição e *'parse'* dos dados do GPS para gerar os mapas;
- Criar um servidor *back-end*, para que computadores externos pudessem acessar o dispositivo;
- Gerar uma interface gráfica simples, de utilização intuitiva e de fácil análise;

A. Configurações de UART

A conexão UART exigiu procedimentos prévios de configuração do *Raspberry*. Como padrão, a transmissão serial é dedicada para operar o console, e é preciso desativar essa funcionalidade para fazer uso dos pinos seriais. Esse procedimento é feito removendo qualquer referência a "console"feita no arquivo */boot/cmdline.txt*. Faz-se necessário fazer o *reboot* do sistema para que a mudança surta efeito.

Vale salientar que, no *Raspberry Pi 3*, a adição do *Bluetooth* alterou o local para comunicação UART de */dev/ttyAMA0* para */dev/ttyS0* (utilizada nos códigos desse dispositivo de forma alternativa, como */dev/serial0*).

A conexão UART estabelecida entre o módulo GPS o *Raspberry* tem *baud rate* de 9600 *bits* por segundo. O módulo GPS envia, a cada segundo, um pacote de *strings* no padrão NMEA contendo informações sobre horário, posicionamento e satélites compondo a triangulação. Para os primeiros testes de comunicação e obtenção de dados, foi utilizado o *software gpsd/cgps*[11].

Com essa aplicação instalada, dispõe-se no terminal do sistema operacional o status da conexão com o módulo, e as principais informações obtidas, utilizando os comandos a seguir:

```
$ sudo gpsd /dev/serial0 -F
/var/run/gpsd.sock
$ cgps -s
```

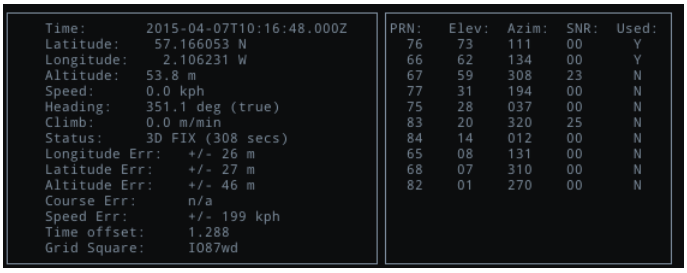

Figura 7. Exemplo de retorno da aplicação gpsd/cgps.

Obtendo retorno dessa aplicação, certifica-se que o módulo GPS e a comunicação estabelecida estão funcionando como previsto. A partir de então, o tratamento das *strings* NMEA enviadas deve ser feito dentro do programa da plataforma de monitoramento, para possibilitar a manipulação desses valores e sua compilação em um banco de dados.

[11]"cgps - Linux Man Page". [Online]. Available: https://www.systutorials.com/docs/linux/man/1-cgps/. [Accessed: 02- Jun- 2017]

B. Padrão NMEA e 'parse' dos valores

O padrão NMEA é um protocolo estabelecido para a transmissão de dados para dispositivos de georreferenciamento. Esse protocolo consiste no envio de *strings* de estrutura padronizada, cada uma com um conjunto de infrmações específicas, e se diferenciam entre si de acordo com o ID de origem da mensagem (primeiros 6 *bits*). Abaixo, está um exemplo de *string GPGGA (GPS Fix information)*. Desta string especificamente, é feito o *parse* (discretização) dos valores de posicionamento, velocidade e altitude, a serem utilizados no projeto.

```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,
1,8,1.03,61.7,M,55.2,M,,*76
```

Após os *bits* de identificação*(\$GPGGA)*, cada campo separado por vigula traz informações de GPS. Por exemplo, o primeiro campo informa o horário no fuso GMT, o segundo e terceiro campo trazem informações da latitude e o quarto e quinto campos, da longitude.

A biblioteca *gps.h*, a partir do arquivo nmea.c, recolhe esses valores de cada envio da string e os disponibiliza em uma struct, da qual e possível resgatar os valores no programa principal já como variáveis, do tipo *double*.

C. Servidor

Na criação do servidor, foi utilizada a solução supracitada, o *Flask* - um *framework* em *python* que faz toda a logística de servidor. Isto é, o Flask torna o *Raspberry* um ponto de acesso, o qual, a partir do seu IP, pode rodar *templates* em diversas linguagens web, tais como o PHP, HTML, CSS e até mesmo o *python*. Ele utiliza de um pacote para renderizar os templates e torna a interface independente do *back-end*. Outra facilidade que o mesmo permite é a manutenção em tempo real de todo o *website*. Caso seja necessário, basta apenas editar o arquivo que deseja e assim que salvar, o *Flask* atualiza a página web. Para a interface foram usados scripts com *APIs* para fazer o *HeatMap*. Toda a interface foi configurada em HTML.

D. Interface de controle

Utilizaremos dos GPIOs para fazer o controle do aparelho final. Haverão botões para inicialização das gravações e de liga e desliga do sistema. Esse botões acionam sinais, que acionam funções. O botão ON/OFF, por exemplo, a ser acionado, despertará uma função *toggle*, que fará a interrupção de um laço de aquisição e armazenamento dos dados.

V. Resultados

O servidor, até então, funciona como esperado. Após uma requisição, é gerado um *log* pelo *Raspberry* que pode retornar o IP da máquina que está requisitando acesso ao monitoramento.

A interface gráfica com a API do Google mostra-se como uma ferramenta bastante adequada para o projeto, exibindo os dados de forma intuitiva e simplificada, localizando as coordenadas no terreno do mapa.

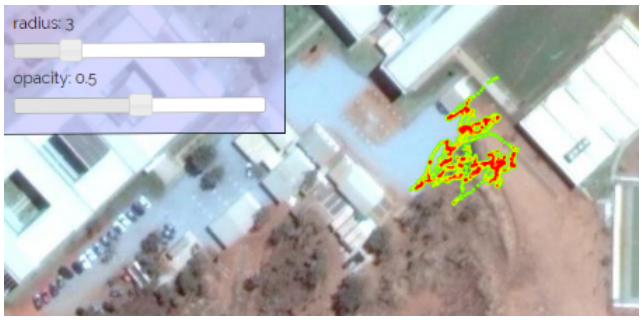

Figura 8. Interface de servidor exibindo dados armazenados.

O GPS funciona corretamente em campo aberto. Em lugares fechados, geralmente, ele tem dificuldade na obtenção de fixos (triangulação com satélites), custando para retornar valores de *tracking*. Foi possível, após uma quantidade de tempo maior que a observada em outros testes, obter fixos dentro da sala de aula, mas os valores obtidos não se mostravam precisos o suficiente para a geração dos mapas de calor, resultando em discrepâncias de deslocamento no mapa.

Apêndice A
Códigos principais do sistema

A. Código 1 - Aquisição de Dados com o GPS - V1.0

```
//compile usando: gcc -o
//tracking tracking.c -lgps -lm

#include <stdio.h>
#include <stdlib.h>
#include <gps.h>
#include <signal.h>

FILE *fp;

void fecha(){
        fclose(fp);
        exit(0);
}
int main() {
        int i,u=0;
    gps_init();
    loc_t data;
        signal(SIGINT, fecha);


    while(1){
        fp = fopen("data.csv", "a+");
        if(u==0){
                fprintf(fp,"\"w\";\"lat\"
;\"lon\"\n");
                u=1;}
        for(i=0;i<10;i++){
        gps_location(&data);
        sleep(1);
        fprintf(fp,"0;%lf;%lf\n",
data.latitude, data.longitude);
        printf("%lf %lf %lf %lf\n",
data.latitude, data.longitude, data.speed,
data.altitude);
        }
        fclose(fp);
    }
    return EXIT_SUCCESS;
}
```

B. Código 2 - Servidor Flask

```
from flask import Flask, render_template, g

app = Flask(__name__)

@app.route('/')
def index():
  return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

C. Código 3 - Web Page

```
<script src="./papaparse.min.js"></script>
<script src="https://maps.googleapis.com/
    maps/api/js?v=3.exp&libraries=
    visualization"></script>

<script src="./jquery-2.1.1.min.js"></script
    >
<script src="https://code.jquery.com/ui
    /1.11.0/jquery-ui.js"></script>
<link rel="stylesheet" href="https://code.
    jquery.com/ui/1.11.0/themes/smoothness/
    jquery-ui.css">

<script>
  var map, pointarray, heatmap;
  var csv = [];
  // http://stackoverflow.com/a
     /2901298/562440
  function numberWithCommas(x) {
      return x.toString().replace(/\B(?=(\d
         {3})+(?!\d))/g, ",");
  }
  function handleFileSelect(evt) {
    var file = evt.target.files[0];

    Papa.parse(file, {
      header: true,
      dynamicTyping: true,
      complete: function(results) {
        csv = [];
        if(results.meta.fields.indexOf("
            weight") == -1) {
          for(idx in results["data"]) {
            var row = results["data"][idx];
            csv.push(new google.maps.LatLng(
                row["lat"], row["lon"]))
          }
        } else {
          var max = results["data"][0]["
              weight"];
          for(idx in results["data"]) {
            var row = results["data"][idx];

            max = Math.max(max, row["weight"
                ]);
            csv.push({
              location: new google.maps.
                  LatLng(row["lat"], row["lon
                  "]),
              weight: row["weight"]
            });
          }
          $("#max-label").html("max: "+
              numberWithCommas(max));
          $("#max-slider").slider("option","
              max",max);
          $("#max-slider").slider("option","
              value",max);
        }
```

APÊNDICE B
CÓDIGOS QUE COMPÕEM A BIBLIOTECA *gps.h*

A. Código 4 - Comunicação via UART

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <inttypes.h>
#include <string.h>

#include "serial.h"
#define PORTNAME "/dev/serial0"
// porta /dev/ttySO do Raspberry PI 3

int uart0_filestream = -1;

void serial_init(void)
{
    uart0_filestream = open(PORTNAME, O_RDWR |
        O_NOCTTY | O_NDELAY);

    if (uart0_filestream == -1)
    {
        //TODO error handling...
    }
}

void serial_config(void)
{
    struct termios options;
    tcgetattr(uart0_filestream, &options);
    options.c_cflag = B9600 | CS8 | CLOCAL |
        CREAD;
    options.c_iflag = IGNPAR;
    options.c_oflag = 0;
    options.c_lflag = 0;
    tcflush(uart0_filestream, TCIFLUSH);
    tcsetattr(uart0_filestream, TCSANOW, &
        options);
}

void serial_println(const char *line, int len)
{
    if (uart0_filestream != -1) {
        char *cpstr = (char *)malloc((len+1) *
            sizeof(char));
        strcpy(cpstr, line);
        cpstr[len-1] = '\r';
        cpstr[len] = '\n';

        int count = write(uart0_filestream,
            cpstr, len+1);
        if (count < 0) {
            //TODO: handle errors...
        }
        free(cpstr);
    }
}

// Lendo uma linha recebida pela UART
```

```
void serial_readln(char *buffer, int len)
{
    char c;
    char *b = buffer;
    int rx_length = -1;
    while(1) {
        rx_length = read(uart0_filestream, (
            void*)(&c), 1);

        if (rx_length <= 0) {
            //espera por mensagens
            sleep(1);
        } else {
            if (c == '\n') {
                *b++ = '\0';
                break;
            }
            *b++ = c;
        }
    }
}

void serial_close(void)
{
    close(uart0_filestream);
}
```

B. Código 5 - Conversão do padrão NMEA para variáveis double

```
//Tratamento dos dados das strings NMEA
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#include <string.h>
#include <math.h>

#include "nmea.h"

void nmea_parse_gpgga(char *nmea, gpgga_t *loc
    )
{
    char *p = nmea;

    p = strchr(p, ',')+1;

    p = strchr(p, ',')+1;
    loc->latitude = atof(p);

    p = strchr(p, ',')+1;
    switch (p[0]) {
        case 'N':
            loc->lat = 'N';
            break;
        case 'S':
            loc->lat = 'S';
            break;
        case ',':
            loc->lat = '\0';
            break;
            //Tratamento de hemisferios de
                latitude (Norte e Sul)
    }

    p = strchr(p, ',')+1;
    loc->longitude = atof(p);

    p = strchr(p, ',')+1;
    switch (p[0]) {
        case 'W':
            loc->lon = 'W';
            break;
        case 'E':
            loc->lon = 'E';
            break;
        case ',':
            loc->lon = '\0';
            break;
            //Tratamento de hemisferios de
                latitude (Leste e Oeste)
    }

    p = strchr(p, ',')+1;
    loc->quality = (uint8_t)atoi(p); //
        Qualidade da recepcao de dados

    p = strchr(p, ',')+1;
    loc->satellites = (uint8_t)atoi(p);

    p = strchr(p, ',')+1;

    p = strchr(p, ',')+1;
    loc->altitude = atof(p);
}

void nmea_parse_gprmc(char *nmea, gprmc_t *loc
    )
{
    char *p = nmea;

    p = strchr(p, ',')+1;
    p = strchr(p, ',')+1;

    p = strchr(p, ',')+1;
    loc->latitude = atof(p);

    p = strchr(p, ',')+1;
    switch (p[0]) {
        case 'N':
            loc->lat = 'N';
            break;
        case 'S':
            loc->lat = 'S';
            break;
        case ',':
            loc->lat = '\0';
            break;
    }

    p = strchr(p, ',')+1;
    loc->longitude = atof(p);

    p = strchr(p, ',')+1;
    switch (p[0]) {
        case 'W':
            loc->lon = 'W';
            break;
```

```
        case 'E':
            loc->lon = 'E';
            break;
        case ',':
            loc->lon = '\0';
            break;
    }

    p = strchr(p, ',')+1;
    loc->speed = atof(p);

    p = strchr(p, ',')+1;
    loc->course = atof(p);
}

uint8_t nmea_get_message_type(const char *
    message)
{
    uint8_t checksum = 0;
    if ((checksum = nmea_valid_checksum(
        message)) != _EMPTY) {
        return checksum;
    }

    if (strstr(message, NMEA_GPGGA_STR) !=
        NULL) {
        return NMEA_GPGGA;
    }

    if (strstr(message, NMEA_GPRMC_STR) !=
        NULL) {
        return NMEA_GPRMC;
    }

    return NMEA_UNKNOWN;
}

uint8_t nmea_valid_checksum(const char *
    message) {
    uint8_t checksum= (uint8_t)strtol(strchr(
        message, '*')+1, NULL, 16);

    char p;
    uint8_t sum = 0;
    ++message;
    while ((p = *message++) != '*') {
        sum ^= p;
    }

    if (sum != checksum) {
        return NMEA_CHECKSUM_ERR;
    }

    return _EMPTY;
}
```

C. Código 6 - Funções para aquisição dos dados do GPS no programa principal

```
#include "gps.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "nmea.h"
#include "serial.h"

extern void gps_init(void) {
    serial_init();
    serial_config();

}

extern void gps_on(void) {
}

extern void gps_location(loc_t *coord) {
    uint8_t status = _EMPTY;
    while(status != _COMPLETED) {
        gpgga_t gpgga;
        gprmc_t gprmc;
        char buffer[256];

        serial_readln(buffer, 256);
        switch (nmea_get_message_type(buffer))
            {
            case NMEA_GPGGA:
                nmea_parse_gpgga(buffer, &
                    gpgga);

                gps_convert_deg_to_dec(&(gpgga
                    .latitude), gpgga.lat, &(
                    gpgga.longitude), gpgga.lon
                    );

                coord->latitude = gpgga.
                    latitude;
                coord->longitude = gpgga.
                    longitude;
                coord->altitude = gpgga.
                    altitude;

                status |= NMEA_GPGGA;
                break;
            case NMEA_GPRMC:
                nmea_parse_gprmc(buffer, &
                    gprmc);

                coord->speed = gprmc.speed;
                coord->course = gprmc.course;

                status |= NMEA_GPRMC;
                break;
        }
    }
}

extern void gps_off(void) {
    serial_close();
}

void gps_convert_deg_to_dec(double *latitude,
    char ns,  double *longitude, char we)
{
    double lat = (ns == 'N') ? *latitude : -1
```

```
        * (*latitude);
    double lon = (we == 'E') ? *longitude : -1
         * (*longitude);

    *latitude = gps_deg_dec(lat);
    *longitude = gps_deg_dec(lon);
}

double gps_deg_dec(double deg_point)
{
    double ddeg;
    double sec = modf(deg_point, &ddeg)*60;
    int deg = (int)(ddeg/100);
    int min = (int)(deg_point-(deg*100));

    double absdlat = round(deg * 1000000.);
    double absmlat = round(min * 1000000.);
    double absslat = round(sec * 1000000.);

    return round(absdlat + (absmlat/60) + (
        absslat/3600)) /1000000;
}
```