

# *Monitoramento de Veículos utilizando o Raspberry Pi*

Divino Luiz Barbosa Moreira  
Universidade de Brasília – Campus Gama  
Brasília, Distrito Federal  
d-luiz@hotmail.com

Douglas da Silveira Alves  
Universidade de Brasília – Campus Gama  
Brasília, Distrito Federal  
douglasdds@gmail.com

**Resumo**— O seguinte trabalho visa a elaboração de um sistema de monitoramento de veículos utilizando a Raspberry Pi com o auxílio de uma câmera infravermelho que fará o reconhecimento facial do condutor do veículo, se constatado que o condutor não é o proprietário a câmera irá disparar uma fotografia e enviá-la junto com as coordenadas do veículo por GSP (Global System Position) para o e-mail do proprietário.

**Palavras-chave**—monitormento, veículos, Raspberry Pi, fotografia, GPS, e-mail.

## I. JUSTIFICATIVA

Com o aumento da criminalidade um dos crimes que tem maior ocorrência pelo país é o roubo de carros. No Distrito Federal dados da secretaria de segurança comprovaram que em 2016 1 carro era roubado a cada 40 minutos. Um dos métodos mais eficazes e que ajudam na recuperação e localização do veículo em caso de roubo é o monitoramento e rastreamento por GPS.

## II. OBJETIVO

O projeto tem por objetivo fazer o monitoramento e o rastreamento do veículo para proteger tanto os ocupantes como próprio automóvel. Isso é feito utilizando uma câmera infravermelho que faz o reconhecimento facial do condutor do veículo e compara com uma fotografia armazenada do proprietário do carro, caso não haja o reconhecimento facial a câmera instalada no veículo dispara uma foto, enviando para o e-mail do proprietário a foto do assaltante e as coordenadas para localização do automóvel, facilitando dessa forma a recuperação do automóvel.

## III. REQUISITOS

- Raspberry Pi 3.
- Câmera Infravermelho.
- Módulo GPS;
- Interface Web.
- Comunicação Wireless.

## IV. BENEFÍCIOS

Como o sistema opera em tempo real as chances de recuperação do veículo aumentam significativamente e com a foto tirada pela câmera as autoridades também tem maiores chances de encontrar e punir o assaltante.

## V. IMPLEMENTAÇÃO

Tem-se que o projeto pode ser definido em três partes: a captura da foto com a data e o horário em que o indivíduo tenta furtar o carro, o registro da localização do veículo e o envio destas informações para o e-mail do proprietário do veículo.

### A. Captura da foto:

Para conseguir capturar o rosto da pessoa que está invadindo o carro, optou-se inicialmente em se utilizar a aplicação fswebcam por ser uma aplicação de instalação rápida, leve e de simples utilização, comparadas a outras pesquisadas pela dupla, como o ffmpeg e o opencv. Entretanto, essa extensão para raspberry possui algumas limitações, por exemplo, a tentativa de reconhecimento facial pode ser implementada pela função motion, porém esta realiza de acordo com a variação de movimento do indivíduo, por meio da mudança significativa na imagem.

Inicialmente, é necessário realizar a instalação da aplicação na raspberry, usando o comando `sudo apt-get fswebcam`. Feito isso, basta utilizar o comando `fswebcam nome_imagem.jpg`; este comando retira a foto e a deixa armazenada na root da raspberry pi. A imagem salva por esse comando é 320x280, porém é possível aumentar sua resolução, dependendo da webcam utilizada, para isso basta digitar a resolução após char a aplicação: `fswebcam -r 640x480 nome_imagem.jpg`. Além disso, a aplicação permite retirar fotos por intervalos regulares, por exemplo, a cada 5 segundos: `fswebcam -loop 5 imagem_%T.jpg` e para encerra-lo será necessário interromper a aplicação usando `Control+C`.

Além disso, é possível observar a webcam em tempo real através do navegador web. Para tanto, pode-se utilizar a aplicação motion (Anexo I). Para instala-la, usa-se o comando: `sudo apt-get motion`. Após a instalação, é necessário configurar o arquivo que controla a aplicação `/etc/motion/motion.config`, alterando os campos relacionados a captura de frames para fotos e imagens, resolução e interface http. Realizadas as alterações, é necessário saber o ip onde a raspberry está conectada, usando o comando `ifconfig`, e digita-lo no navegador e ao final, a porta setada para tal aplicação, normalmente a 8081. Feito isso, é possível visualizar a pessoa em tempo real.

### B. Email:

Para enviar o email contendo a imagem e a localização como anexos, é necessário que a raspberry pi possua uma aplicação MTA (Message Transfer Agent), responsável por enviar o email no formato de cliente-servidor. O MTA utilizando o smtp, uma alternativa simples ao sendmail que possui interatividade com a raspberry pi; além disso, é necessário instalar outros dois pacotes de funcionalidades, como o mailutils que consiste em um conjunto de ferramentas e comandos para processar o email e o mpack como meio de codificação da mensagem.

Inicialmente, foi instalado as aplicações citadas acima usando o comando `apt-get install exim4`; `sudo apt-get install mailutils` e `sudo apt-get install mpack`. Logo após, o MTA foi configurado por meio de sua interface gráfica com o comando: `sudo dpkg-reconfigure exim4-config`, onde foram definidos o servidor, o nome do sistema de e-mail, o ip e o endereço hostname para envio de mensagem (gmail, hotmail, outlook, entre outros), verificar Anexo II. Em seguida, configurou-se por meio da root, o acesso do cliente por meio do comando `sudo nano /etc/exim4passwd.client` onde foram adicionadas as seguintes linhas no arquivo, considerando a utilização do gmail:

```
gmail-smtp.l.google.com:YOU@gmail.com:PASSWORD
*.google.com:YOU@gmail.com:PASSWORD
smtp.gmail.com:YOU@gmail.com:PASSWORD
```

Por fim, a aplicação de e-mail foi reiniciada pelos comandos:

```
sudo update-exim4.conf
sudo /etc/init.d/exim4 restart
```

Para enviar um mensagem contendo texto, basta utilizar o comando no terminal:

```
mail -s "tese" nomedoemail@exemplo.com
```

Para enviar o arquivo com anexo, basta digitar o comando no terminal:

```
mpack -s "teste" /home/pi/diretóriodesejado/ arquivo.extensão
nomedoemail@exemplo.com
```

### C. Módulo GPS

Nesse ponto de controle o objetivo ao se trabalhar com o módulo GPS era entender seu funcionamento básico e se ele seria capaz de atender as necessidades do projeto. Os testes realizados foram feitos utilizando a placa Arduino UNO, para o próximo ponto de controle o módulo será acoplado a Raspberry. Foi utilizado o modelo Gy-neo6mv2 que é compatível com o Arduino e com a Raspberry. Esse modelo se mostrou extremamente eficiente e fornece diversas informações como latitude, longitude, data, hora e etc.

O módulo GPS funciona com o sistema GPS que consiste de 24 satélites na órbita terrestre. Para localizar e calcular a posição do receptor GPS são necessários pelo menos 3 satélites. A função básica desse módulo é enviar sinais para

os satélites e calcular o tempo que eles demoram para chegar, depois é feito um cálculo de triangulação e então é possível descobrir a localização em terra. A descrição de hardware e software serão feitas nos tópicos a seguir.

## VI DESCRIÇÃO DE HARDWARE

Para o desenvolvimento desta etapa do projeto foram utilizados 2 notebooks com o Sistema operacional Linux, 1 Raspberry Pi 3 model B, 1 webcam Multilaser de 5 Mp WC040 e 1 módulo GPS.

O circuito montado para fazer a ligação entre o módulo GPS e a placa Arduino está esquematizado na figura 1 a seguir.

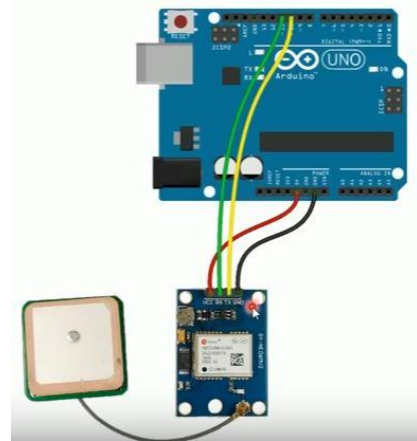


Figura1: Esquemático do circuito.

O receptor trabalha com uma alimentação que pode variar de 3.3V a 5V, dessa forma foi utilizada a saída de 5V do Arduino para alimentar o módulo, no esquema essa ligação é representada pelo fio vermelho. O terra do receptor foi ligado ao terra da placa representado pelo fio preto.

A comunicação entre o Arduino e o módulo é do tipo serial, dessa forma o RX do módulo deveria ser ligado ao TX da placa e o TX no módulo ao RX da placa, porém, foram utilizadas 2 portas digitais do Arduino para fazer essa comunicação, pois o Arduino apresenta uma limitação. Ao serem usadas as portas RX e TX, a porta serial de comunicação com o computador fica inabilitada e o Arduino não consegue mais enviar e receber informações do computador. Dessa maneira, a porta RX do módulo foi ligado no pino 11 da placa, conexão feita pelo fio verde, e a porta TX ligado no pino 10 em amarelo.

## VII DESCRIÇÃO DE SOFTWARE

O programa utilizado para fazer a comunicação entre o Arduino e o receptor GPS está no anexo 1. O módulo GPS manda a cada período de tempo um pacote de dados na forma de texto contendo várias informações. O programa busca nesse pacote de dados as informações que são necessárias e permite que elas sejam trabalhadas de forma mais fácil. No programa foram utilizadas duas bibliotecas que permitiram isso. A primeira foi a SoftwareSerial.h, que é uma biblioteca que transforma uma porta digital do Arduino em porta serial, com essa biblioteca foi possível resolver o problema da limitação de hardware do Arduino. A segunda biblioteca é a TinyGPS.h, que é biblioteca que pega o pacote de dados do módulo GPS.

No programa é feita à instanciação de dois objetos, para a classe `SoftwareSerial` o objeto é o `serial1` e para a classe `TinyGPS` o objeto é o `gps1`. Um é para fazer a comunicação e o outro para fazer a interpretação dos dados recebidos. Na comunicação serial foi definida a velocidade 9600, pois é nessa velocidade que o módulo funciona. O uso do módulo é feito de forma simples utilizando um laço `while` e com a função `available()` que verifica se há informações disponíveis. Se houverem informações disponíveis, elas são lidas pela função `read()` e colocadas em uma variável `char` que foi chamada de `cIn`. Após esse procedimento as informações contidas em `cIn` são enviadas para o objeto `gps1` caractere por caractere até o fim da informação. Quando o pacote de informação for totalmente lido e estiver completo será possível trabalhar com ele e utilizando funções como a `get_position()` é possível pegar a latitude e a longitude. A figura 2 mostra os dados obtidos pelo módulo GPS.

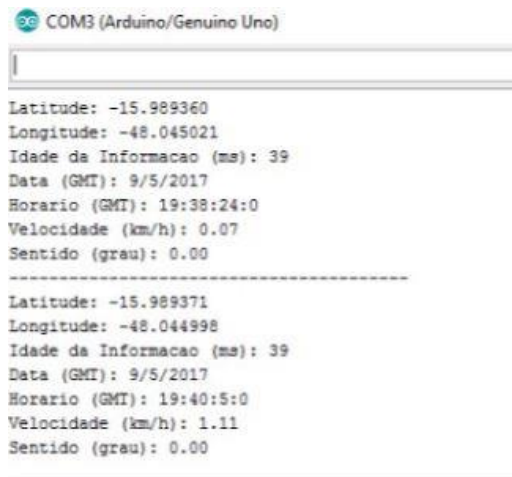


Figura 2: Dados obtidos pelo GPS.

## VIII RESULTADOS

Esperava-se nessa etapa do projeto visualizar o funcionamento modularizado do reconhecimento da webcam, do Sistema para enviar mensagens para o email e a recepção das coordenadas pelo módulo GPS (Global Position System). Pode-se observar que o sistema webcam funcionou dentro do previsto, pois foi possível retirar a foto da pessoa que está invadindo o carro, além de conseguir visualizar em tempo real o que acontece através de um endereço na página web. Assim, para assegurar que a foto registrada contém a face do suspeito, é necessário que aplicar o reconhecimento facial, aplicação que a dupla pretende implementar para a próxima etapa do projeto. Além disso, verificou-se que o Sistema de email enviado pela raspberry pi funcionou de modo satisfatório, pois foi possível enviar para o email cadastrado na raspberry a mensagem de texto e o arquivo em anexo contendo a fotografia e o arquivo `txt` contendo informações relativas a coordenadas. É importante ressaltar que nessa fase de testes, ainda não foi possível utilizar a raspberry pi junto ao módulo GPS devido a complicações relativas a conexão e leitura nas portas GPIO da raspberry pi 3. Por fim, verificou-se que o módulo GSP funcionou corretamente, quando foi testado no Arduino Uno. Este retornou a localização exata em todas as vezes em que foi testado, informando as coordenadas de latitude, longitude, dia e horário. A dupla pretende utilizar apenas a raspberry para

obter tais informações para a próxima etapa do projeto, porém em caso de mal funcionamento, estes valores serão fornecidos para raspberry através de comunicação UART com o Arduino.

## IX CONCLUSÃO

Conclui-se que o projeto é passível de ser realizado com êxito por parte da dupla, porém necessita-se realizar ajustes. Para a próxima etapa do projeto, pretende-se implementar o sistema de reconhecimento facial e agregar o módulo GPS à raspberry pi, de modo a aplicar os conceitos vistos em sala de aula relacionados a processos e threads, além da implementação de scripts de modo a otimizar o funcionamento do Sistema embarcado.

## REFERÊNCIAS

- [1] Disponível em: <http://g1.globo.com/distrito-federal/noticia/2016/07/df-registra-em-media-um-roubo-ou-furto-de-carro-cada-40-minutos.html>. Acesso em 02 de Abril. 2017.
- [2] Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2008-1-23-vf-leandrobeszczynski.pdf>. Acessado em 29 de Março. 2017.
- [3] Disponível em: [http://files.comunidades.net/mutcom/Monte\\_um\\_localizador\\_e\\_bloqueador\\_veicular\\_via\\_SMS.pdf](http://files.comunidades.net/mutcom/Monte_um_localizador_e_bloqueador_veicular_via_SMS.pdf). Acesso em 29 de Março. 2017.
- [4] Disponível em: <http://docplayer.com.br/19733841-Configurando-raspberry-pi-com-camera-em-modo-de-video-vigilancia.html>. Acesso em 29 de Março. 2017.
- [5] Disponível em: <https://roboott.wordpress.com/2016/01/07/raspberry-pi-servidor-de-webcam/>. Acessado em 06 de Maio de 2017.
- [6] Disponível em: <http://www.awesomeprojects.xyz/2015/09/beginners-guide-how-to-setup-usb-webcam.html>. Acessado em 06 de maio de 2017.
- [7] Disponível em: <http://ask.xmodulo.com/install-usb-webcam-raspberry-pi.html>. Acessado em 06 de maio de 2017.
- [8] Disponível em: <http://dqsoft.blogspot.com.br/2015/04/conectando-uma-webcam-ao-raspberry-pi.html>. Acessado em 07 de Maio de 2017.
- [9] Disponível em: [http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionGuideBasicFeatures#on\\_motion\\_detected](http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionGuideBasicFeatures#on_motion_detected). Acessado em 07 de Maio de 2017.
- [10] Disponível em: <http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionGuide>

[BasicFeatures#Snapshots 45 The Traditional Periodic Web Camera](#) . Acessado em 07 de maio de 2017.

[11] Disponível em : <http://tudosobreraspberry.info/2017/03/controle-os-sensores-ligados-ao-raspberry-por-interface-web-com-o-cayenne/>. Acessado em 07 de Maio de 2017.

[12] Disponível em : <https://pplware.sapo.pt/truques-dicas/tutorial-raspberry-pi-enviar-e-mails-via-gmail/>. Acessado em 07 de maio de 2017.

[13] Disponível em: <http://blog.andrecardoso.com/raspberry-pi-envio-de-e-mail-pelo-gmail/>. Acessado em 08 de Maio de 2017.

[14] Disponível em : <http://automatobr.blogspot.com.br/2014/09/enviando-email-do-seu-raspberry-pi.html>. Acessado em 08 de Maio de 2017.

[15] Disponível em : <https://blog.butecopensource.org/enviando-emails-com-o-python/>. Acessado em 08 de Maio de 2017.

[16] Disponível em : <https://docs.python.org/2.7/library/email.html>. Acessado em 08 de Maio de 2017.

[17] Disponível em: [http://www.raspberry-projects.com/pi/software\\_utilities/email/smtp-to-send-emails](http://www.raspberry-projects.com/pi/software_utilities/email/smtp-to-send-emails). Acessado em 08 de Maio de 2017.

## Anexo I – Alterações no arquivo Motion

```
#####  
# Daemon  
#####  
# Start in daemon (background) mode and release terminal (default: off)  
daemon on  
#####  
  
#####  
# Live Stream Server  
#####  
# Restrict stream connections to localhost only (default: on)  
stream_localhost off  
#####  
  
#####  
# HTTP Based Control  
#####  
# Restrict control connections to localhost only (default: on)  
webcontrol_localhost off  
#####
```

## Anexo II – Configuração do MTA exim4

sudo dpkg-reconfigure exim4-config

É necessário selecionar as seguintes opções quando a interface gráfica aparecer:

- a) Indicar o tipo de mail server que será usado. Selecionar a opção: "mail sent by smarthost; received via SMTP or fetchmail"
- b) Selecionar o nome do sistema de email: raspberrypi.
- c) Ip: 127.0.0.1::1
- d) Configuração de conta: raspberry.
- e) Machines to relay for: campo vazio.
- f) Endereço dp hostname para envio dos emails: para Google, smtp.gmail.com::587
- g) Hide local mail in outgoing email: selecionar a opção No.
- h) Keep number...: selecionar a opção No
- i) Delivery method for local mail: selecionar "Maildir format in home directory"
- j) Split configuration...: selecionar a opção No.

3) Configuração da conta de email do cliente . Como root, edite /etc/exim4/passwd.client:

sudo nano /etc/exim4/passwd.client

Adicionar as próximas linhas ao fim do arquivo:

```
gmail-smtp.l.google.com:YOU@gmail.com:PASSWORD
*.google.com:YOU@gmail.com:PASSWORD
smtp.gmail.com:YOU@gmail.com:PASSWORD
```

Logo após, é necessário atualizar e reiniciar a aplicação de email. Para isso:

```
sudo update-exim4.conf
sudo /etc/init.d/exim4 restart
```

### ANEXO III

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>

SoftwareSerial serial1(10, 11); // RX, TX
TinyGPS gps1;

void setup() {
  serial1.begin(9600);
  Serial.begin(9600);

  Serial.println("O GPS está aguardando pelo sinal dos satelites...");
}

void loop() {
  bool recebido = false;

  while (serial1.available()) {
    char cIn = serial1.read();
    recebido = gps1.encode(cIn);
  }

  if (recebido) {
    Serial.println("-----");

    //Latitude e Longitude
    long latitude, longitude;
    unsigned long idadeInfo;
    gps1.get_position(&latitude, &longitude, &idadeInfo);

    if (latitude != TinyGPS::GPS_INVALID_F_ANGLE) {
```

```

    Serial.print("Latitude: ");
    Serial.println(float(latitude) / 100000, 6);
}

if (longitude != TinyGPS::GPS_INVALID_F_ANGLE) {
    Serial.print("Longitude: ");
    Serial.println(float(longitude) / 100000, 6);
}

if (idadeInfo != TinyGPS::GPS_INVALID_AGE) {
    Serial.print("Idade da Informacao (ms): ");
    Serial.println(idadeInfo);
}

//Dia e Hora
int ano;
byte mes, dia, hora, minuto, segundo, centesimo;
gps1.crack_datetime(&ano, &mes, &dia, &hora, &minuto, &segundo, &centesimo, &idadeInfo);

Serial.print("Data (GMT): ");
Serial.print(dia);
Serial.print("/");
Serial.print(mes);
Serial.print("/");
Serial.println(ano);

Serial.print("Horario (GMT): ");
Serial.print(hora);
Serial.print(":");
Serial.print(minuto);
Serial.print(":");
Serial.print(segundo);
Serial.print(":");
Serial.println(centesimo);

//altitude
float altitudeGPS;
altitudeGPS = gps1.f_altitude();

if ((altitudeGPS != TinyGPS::GPS_INVALID_ALTITUDE) && (altitudeGPS != 1000000)) {
    Serial.print("Altitude (cm): ");
    Serial.println(altitudeGPS);
}

//velocidade
float velocidade;
//velocidade = gps1.speed();    //nós
velocidade = gps1.f_speed_kmph(); //km/h
//velocidade = gps1.f_speed_mph(); //milha/h
//velocidade = gps1.f_speed_mps(); //milha/segundo

Serial.print("Velocidade (km/h): ");
Serial.println(velocidade, 2); //Conversão de Nós para Km/h

```

```
//sentido (em centesima de graus)
unsigned long sentido;
sentido = gps1.course();

Serial.print("Sentido (grau): ");
Serial.println(float(sentido) / 100, 2);


//satelites e precisão
unsigned short satellites;
unsigned long precisao;
satellites = gps1.satellites();
precisao = gps1.hdop();

if (satellites != TinyGPS::GPS_INVALID_SATELLITES) {
    Serial.print("Satelites: ");
    Serial.println(satellites);
}

if (precisao != TinyGPS::GPS_INVALID_HDOP) {
    Serial.print("Precisao (centesimos de segundo): ");
    Serial.println(precisao);
}
delay(100000);

}
}
```