

# *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  
douglasddsa@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 biblioteca Open CV, cuja é livre para instalação. Diferentemente do ponto de controle anterior, mudou-se da fswebcam para a Open CV em virtude da garantia que a foto retirada pelo Sistema de Monitoramento ser efetivamente o rosto de uma pessoa.

Para isso, desenvolveu-se um código em Python para apenas retirar a foto da pessoa no momento em que sua face fosse reconhecida, evitando assim que objetos fossem capturados em fotos. Isso foi possível em virtude da presença de um identificador facial na biblioteca, denominando `haarcascade_frontalface_default.xml` que realiza o processamento digital para perceber a diferença de contraste nos pixels da imagem. Enquanto não for realizada a leitura, o valor armazenado é 0, e torna-se 1 quando é identificado a face. O código só é inicializado no momento em que há o reconhecimento da presença de face para, então, realizar a captura da foto pelo `command imwrite`. A transmissão em tempo real com delay de 2s é realizada pelo `command imshow`. Após a retirada da foto, o processo aguarda 5s para então, ser finalizado. É importante ressaltar que foi adicionada uma interrupção por sinal do teclado caso o programa não identifique face.

Além disso, é possível observar a webcam em tempo real através do navegador web. Para tanto, configurou-se o endereço de ip e a porta de transmissão. Para isso, 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. Segue em anexo o script em python comentado relacionado os processos para identificação de face.

### 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 utilizado por o

ssmtp, 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 mapck como meio de codificação da mensagem.

Verificou-se nesta etapa do projeto que o MTA que apresenta maior recursos e implementação por python é o SMTP.

Inicialmente, foi instalado as aplicações SMTP atualizando a versão e a biblioteca Python presente na Raspberry. Logo após, o MTA foi configurado por meio de linha de comando, 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 MIME Multipart, Sistema de mensagem multimedia, os anexos de texto, imagem e corpo da mensagem a ser enviada. Segue em anexo o script em Python onde foram comentadas todas as configuração necessárias para o envio de mensagem pelo Gmail.

### C. Código para execução

Para agregar todos os components de imagem, coordenada e Sistema de envio por email, para simples teste de funcionamento, optou-se por desenvolver um código que realizada a chamada por meio de função no terminal através de processos pai e filho. No código presente no anexo, cria-se um processo filho que será responsável por realizar a chamada de Sistema (system("")) do script executável (chmod +x take\_picture.py) em python para retirar a foto de uma pessoa apenas no momento em que é reconhecida a face. Enquanto a face não for reconhecida, o processo filho permanece no loop. Entretanto, no momento em que a face é reconhecida, a foto é retirada e salva no diretório (/home/pi/IMAGEMteste.png). O processo filho é finalizado através da função wait(& status), cuja só permite que o processo pai retorne a atividade seguinte, apenas no momento em que o filho é finalizado. Dentro do processo pai, é realizada outras 2 chamadas pelo Sistema, na primeira é executado o código para gravação das coordenadas de latitude e longitude no arquivo Latitude\_Longitude.txt. Após a finalização deste, é executado o código para envio para email, onde serão anexada a foto e a coordenada. Segue em anexo o código comentado.

### D. Módulo GPS

Nesse ponto de controle o objetivo era conectar o módulo GPS diretamente a Raspberry, porém o código para fazer essa comunicação não foi desenvolvido corretamente para o prazo estabelecido e será apresentado no próximo ponto de controle, dessa maneira a placa Arduino foi utilizada novamente para receber as coordenadas do módulo GPS. As coordenadas eram recebidas e enviadas para a Raspberry utilizando uma comunicação serial do tipo UART.

## VI DESCRIÇÃO DE HARDWARE

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

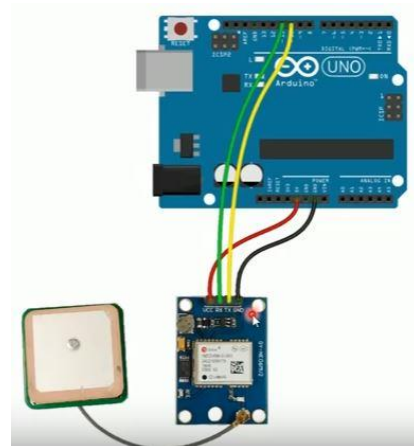


Figura1: Esquemático do circuito.

Foi montado um circuito para as comunicação do módulo GPS com o Arduino e um circuito para a comunicação UART do Arduino com a Raspberry.

No primeiro 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 modulo é do tipo serial, dessa forma o RX do modulo deveria ser ligado ao TX da placa e o TX no modulo ao RX da placa, porém, foram utilizadas 2 portas digitais do Arduino para fazer essa comunicação, pois as portas RX e TX seriam utilizadas para fazer a comunicação entre o Arduino e a Raspberry. 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. Transformar as portas digitais em seriais foi possível com a biblioteca SoftwareSerial.

Para fazer a comunicação UART entre o Arduino e a Raspberry foram utilizadas as portas RX e TX do Arduino e da Raspberry. Foi feito um circuito divisor de tensão, pois a porta TX do Arduino trabalha com uma tensão de 5V e a porta RX da Raspberry com 3.3V, dessa forma se as 2 fossem ligadas diretamente a Raspberry seria sobrealimentada e queimaria. O circuito utiliza 3 resistores de 390Ω. A figura 2 abaixo mostra o esquemático desse circuito

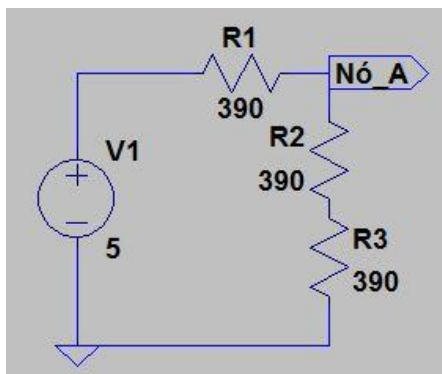


Figura 2: Circuito divisor de tensão.

Fazendo os cálculos para a tensão no Nó A, a tensão encontrada será de 3.33V, que é aceitável já que os pinos da Raspberry suportam até 3.5V. A porta TX da Raspberry foi ligada diretamente a porta RX do Arduino

## 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 conectar o módulo GPS as portas 10 e 11 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 a 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. As informações que o módulo passa são do tipo long, esse tipo de variável trabalha com 4 bytes. O comando Serial.write() envia apenas um byte por vez, dessa forma foi feito o seguinte trecho de código:

```
Serial.write(latitude);
Serial.write((latitude)>>8);
Serial.write((latitude)>>16);
Serial.write((latitude)>>24);
```

em que cada linha é feito um deslocamento de 8 bits para que sejam transmitidos os 4 bytes que compõem a informação completa. O mesmo foi feito para que a latitude também fosse enviada de forma correta.

No código da Raspberry até a linha 38 foi feito o procedimento padrão para fazer a configuração da comunicação UART e assim poder receber as coordenadas. Logo após foi definida a variável *i* que recebe o valor das coordenadas, as variáveis float *convert* e *convert2*, a variável *convertido* do tipo char e as variáveis *a*, *b*, *c* e *stop* que são variáveis de controle. Em seguida foi definido uma variável *ponteiro* do tipo FILE para que as coordenadas pudessem ser salvas em um arquivo .txt. Após a abertura do arquivo para escrita foi criado um laço *while* que era controlado pela variável *stop* até que ela fosse menor do que 9600, que era o Baud rate da comunicação, então foi feito um *if* para saber se haviam informações disponíveis, se houvessem informações disponíveis era criado uma variável *j* para fazer o controle de um laço *for* e *i* era iniciada como 0. Após esse procedimento eram feitas duas condições *if* utilizando as variáveis *a*, *b* e *c* para que fossem colocadas no arquivo 2 strings indicando qual dado era a Latitude e qual era a Longitude, mas não seus valores. Era executado então o laço *for* variando de 0 a 3, ele executava a seguinte operação:

```
i /= serialGetchar(uart0_fd) << 8*j;
```

a variável *i* foi declarada do tipo int pois em C essa variável trabalha com 4 bytes e as coordenadas enviadas pelo módulo GPS são do tipo long de 4 bytes também, como explicado no código do arduino. Como o Arduino só mandava 1 byte por vez eram necessário 4 ciclos para receber a mensagem completa e a cada ciclo fazer um deslocamento de múltiplos de 8 para receber os 32 bits da mensagem. Após o recebimento da mensagem ela era convertida para o tipo float na variável *convert*, dividida por 100000 e jogada na variável *convert2*, depois era convertida para string com a função *sprint()* e colocada na variável *convertido*, logo após era salva no arquivo, Esse procedimento era realizado 2 vezes, a primeira para pegar o valor da latitude e a segunda para a longitude.

## VIII RESULTADOS

Esperava-se nessa etapa do projeto visualizar o funcionamento integrado do reconhecimento da facial, 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 de reconhecimento facial funcionou dentro do previsto, pois foi possível retirar a foto do rosto 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 o Sistema só é ativado no momento em que é reconhecida a a face.

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.  
 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.

## 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.

## 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/MotionGuideBasicFeatures#Snapshots\\_45\\_The\\_Traditional\\_Periodic\\_Web\\_Camera](http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionGuideBasicFeatures#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/ssmtp-to-send-emails](http://www.raspberry-projects.com/pi/software_utilities/email/ssmtp-to-send-emails). Acessado em 08 de Maio de 2017.

## Anexo I – Código para reconhecimento facial

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6                                File: take picture.py

#!/usr/bin/env python

from __future__ import print_function
#Importando numpy as np
import cv2 #Importando a biblioteca Open CV
import time #Importando biblioteca com os comandos para funções de tempo
import sys #Importando biblioteca que possui as funções de chamada do sistema
import signal #Adicionando biblioteca que possui Interrupções no sistema

#configurando o arquivo para imagem

#def main(args):

#stream='http://pi@raspberrypi:rasp@10.42.0.1:8081/cgi/mjpg/mjpg.cgi?.njpg'
#cap.open("http://10.42.0.233/?action=stream?dummy=param.njpg")
#stream='http://10.42.0.233:8081/video?x.njpg'
#cap.open(stream)

#-----$

#Definindo interrupção por chamada de CTRL+C pelo terminal
def signal_handler(signal, frame):
    print("Saindo do programa")
    sys.exit(0)

#Importando módulo da biblioteca OpenCV para identificação de face presente na webcam
face_cascade = cv2.CascadeClassifier('/home/pi/opencv-3.2.0/data/haarcascades/haarcascade_frontalface_default.xml')
file= "/home/pi/IMAGEMteste.png" #Indicação de caminho para a captura de imagem pela webcam
cap = cv2.VideoCapture(0) #Inicializando a webcam
print("Iniciando")

AG Get Help      AO WriteOut      AR Read File     AV Prev Page     AK Cut Text      AC Cur Pos
AX Exit          AJ Justify       AW Where Is      AU Next Page     AU UnCut Text    AT To Spell
```

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6                                File: take picture.py

#Inicializando a webcam para capturar frames retangulares no rosto utilizando escala cinza

while (cap.isOpened()):
    ret,frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, flags=cv2.CASCADE_SCALE_IMAGE,minSize=(50, 50), maxSize=NS)

#definindo o tamanho do frame a ser colocado no rosto

    if len(faces) > 0:
        print("Pessoa detectada!")
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x - 10, y - 20), (x + w + 10, y + h + 10), (0, 255, 0), 2)
            roi_gray = frame[y-15:y + h+10, x-10:x + w+10]
#Comando para inicializar a Visualização da webcam
            cv2.imshow("Captura do Rosto", frame)
            cv2.startWindowThread()

#Comando para retirar foto
            print("Tirando a foto")
            cv2.imwrite(file, frame)

            print('Para cancelar, aperte Ctrl+c pelo terminal ')
            signal.signal(signal.SIGINT, signal_handler)

#
            if cv2.waitKey(2000) & 0xFF == ord('q'):
#
            if (cv2.waitKey(3000)) >=3000:
#
                break

#Após a captura, o processo aguarda 5 segundos para ser encerado

            cv2.waitKey(5000)

#Desativando a webcam
```

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6 File: take picture.py

#Comando para retirar foto
print("Tirando a foto")
cv2.imwrite(file, frame)

print('Para cancelar, aperte Ctrl+c pelo terminal ')
signal.signal(signal.SIGINT, signal_handler)

#
if cv2.waitKey(2000) & 0xFF == ord('q'):
if (cv2.waitKey(3000)) >=3000:
break

#Após a captura, o processo aguarda 5 segundos para ser encerrado
cv2.waitKey(5000)

#Desativando a webcam
cap.release()
print("saindo")

#Destruindo as janelas de visualização da webcam
cv2.destroyAllWindows()

^G Get Help      ^O WriteOut      ^R Read File     ^V Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^N Next Page     ^U UnCut Text    ^T To Spell
```

## Anexo II – Código para envio de mensagem pelo GMAIL

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6 File: mymsg2.py

#!/usr/bin/env python

import smtplib #importação da biblioteca SMTP
import os      #importação da biblioteca com as principais operações no sistema, como as funções POSIX
import sys     #Biblioteca que permite maior interação com o sistema, com as funções open, system

#import base64

from email.mime.multipart import MIMEMultipart #importante biblioteca para utilização de diferentes formatos de arquivos no email
from email.mime.text import MIMEText # importante biblioteca para configuração de texto para o email
from email.mime.image import MIMEImage # importante biblioteca para configuração de imagem no email
#from email.mime.application import MIMEApplication
#from email.mime.base import MIMEBase
#from email import Encoders

##Configuração da conta do Usuário
smtpUser = 
smtpPas = 

##Configurando o destinatário, remetente e corpo do email
toAdd = 'douglasdds@gmail.com'
fromAdd = smtpUser

subject = 'email sent from Pi'
header = 'To: ' + toAdd + '\n' + 'From: ' + fromAdd + '\n' + 'Subject: ' + subject + '\n'
body = 'sent from raspberry pi'

print header + '\n' + body

##Configurando o acesso ao servido SMTP para enviar email através do gmail
server=smtplib.SMTP('smtp.gmail.com',587) ##conexão com o servidor pela porta 587
server.ehlo() #Iniciando conexão

^G Get Help      ^O WriteOut      ^R Read File     ^V Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^N Next Page     ^U UnCut Text    ^T To Spell
```



```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6 File: mymsg2.py Modified

##Configurando o destinatário, remetente e corpo do email
toAdd = 'douglassdds@gmail.com'
fromAdd = smtpUser

subject = 'email sent from Pi'
header = 'To: ' + toAdd + '\n' + 'From: ' + fromAdd + '\n' + 'Subject: ' + subject + '\n'
body = 'sent from raspberry pi'

print header + '\n' + body

##Configurando o acesso ao servido SMTP para enviar email através do gmail
server=smtpplib.SMTP('smtp.gmail.com',587) ##conexão com o servidor pela porta 587

server.ehlo() #iniciando conexão
server.starttls() #Tipo de conexão TLS
server.ehlo()

server.login(smtpUser, smtpPass) # Login

#Corpo da mensagem
msg= MIMEMultipart()
msg["From"]= fromAdd
msg["To"]= toAdd
msg["Subject"]= header

#anexos da imagem da Webcam
imgFilename= 'teste de imagem'
with open('/home/pi/IMAGEMteste.png', 'rb') as f:
    msgImg= MIMEImage(f.read(), name=imgFilename)
msg.attach(msgImg)

#anexos da coordenada GPS
txtFilename= 'coordenadas gmail'
with open('/home/pi/Desktop/Latitude_Longitude.txt', 'rb') as f:
    msgTxt= MIMEText(f.read())
    #msgTxt= MIMEBase('multipart', 'plain')
    #msgTxt= MIMEBase('application', 'octet-stream')
    #msgTxt= MIMEApplication(f.read())
    #msgTxt.set_payload(f.read())
#Encoders.encode_base64(msgTxt)
#msgTxt.add_header('Content-Transfer-Encoding', 'base64', filename=txtFilename)
msgTxt.add_header('Content-Disposition', "attachment; filename= %s" % txtFilename)
msg.attach(msgTxt)

msgText= MIMEText('<b>{</b><br><br>'.format(body, imgFilename,txtFilename), 'html')
#msgText= MIMEText('<b>{</b><br><br>'.format(body,txtFilename), 'html')

msg.attach(msgText)

#Anexando remetente, destinatário, assunto e anexos

^G Get Help      ^O WriteOut      ^R Read File     ^V Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^V Next Page     ^U UnCut Text    ^T To Spell
```

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6 File: mymsg2.py Modified

    msgTxt= MIMEText(f.read())
    #msgTxt= MIMEBase('multipart', 'plain')
    #msgTxt= MIMEBase('application', 'octet-stream')
    #msgTxt= MIMEApplication(f.read())
    #msgTxt.set_payload(f.read())
#Encoders.encode_base64(msgTxt)
#msgTxt.add_header('Content-Transfer-Encoding', 'base64', filename=txtFilename)
msgTxt.add_header('Content-Disposition', "attachment; filename= %s "% txtFilename)
msg.attach(msgTxt)

msgText= MIMEText('<b>{}</b><br><br>'.format(body, imgFilename,txtFilename), 'html')
#msgText= MIMEText('<b>{}</b><br><br>'.format(body,txtFilename), 'html')
msg.attach(msgText)

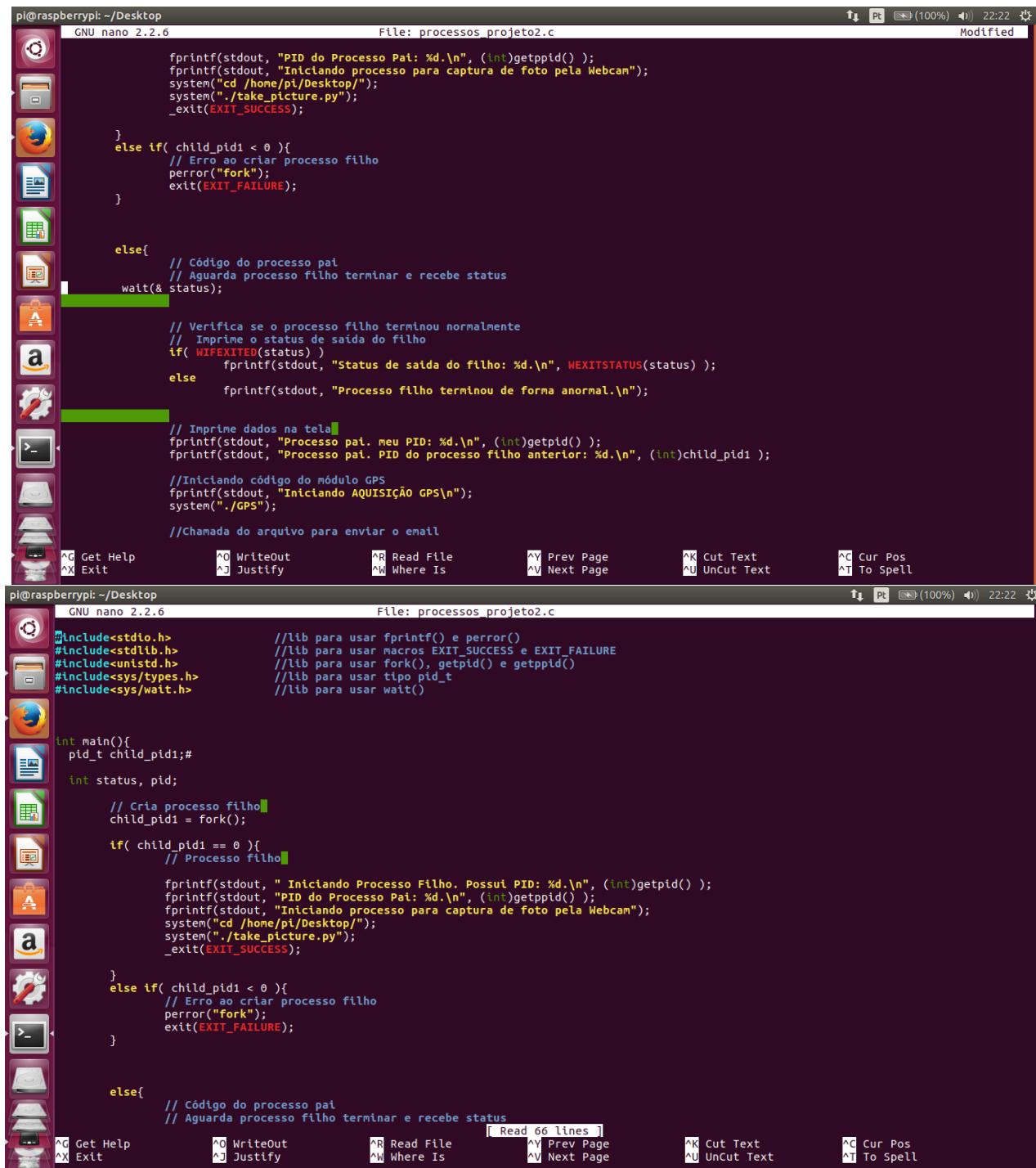
#Anexando remetente, destinatário, assunto e anexos
server.sendmail(fromAdd, toAdd, msg.as_string())
server.quit()

```

^G Get Help      ^O WriteOut      ^R Read File      ^V Prev Page      ^K Cut Text      ^C Cur Pos  
^X Exit          ^J Justify      ^W Where Is      ^N Next Page    ^U UnCut Text   ^T To Spell



### Anexo III – Código envolvendo processos



```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6                               File: processos projeto2.c                               Modified

    fprintf(stdout, "PID do Processo Pai: %d.\n", (int)getppid() );
    fprintf(stdout, "Iniciando processo para captura de foto pela Webcam");
    system("cd /home/pi/Desktop/");
    system("./take_picture.py");
    _exit(EXIT_SUCCESS);

}
else if( child_pid1 < 0 ){
    // Erro ao criar processo filho
    perror("fork");
    exit(EXIT_FAILURE);
}

else{
    // Código do processo pai
    // Aguarda processo filho terminar e recebe status
    wait(& status);

    // Verifica se o processo filho terminou normalmente
    // Imprime o status de saída do filho
    if( WIFEXITED(status) )
        fprintf(stdout, "Status de saída do filho: %d.\n", WEXITSTATUS(status) );
    else
        fprintf(stdout, "Processo filho terminou de forma anormal.\n");

    // Imprime dados na tela
    fprintf(stdout, "Processo pai. meu PID: %d.\n", (int)getpid() );
    fprintf(stdout, "Processo pai. PID do processo filho anterior: %d.\n", (int)child_pid1 );

    //Iniciando código do módulo GPS
    fprintf(stdout, "Iniciando AQUISIÇÃO GPS\n");
    system("./GPS");

    //Chamada do arquivo para enviar o email

^G Get Help      ^O WriteOut      ^R Read File      ^V Prev Page      ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is       ^N Next Page     ^U UnCut Text    ^T To Spell

pi@raspberrypi: ~/Desktop
GNU nano 2.2.6                               File: processos projeto2.c                               [ Read 66 lines ]

#include<stdio.h>                               //lib para usar fprintf() e perror()
#include<stdlib.h>                              //lib para usar macros EXIT_SUCCESS e EXIT_FAILURE
#include<unistd.h>                             //lib para usar fork(), getpid() e getppid()
#include<sys/types.h>                          //lib para usar tipo pid_t
#include<sys/wait.h>                           //lib para usar wait()

int main(){
    pid_t child_pid1;#
    int status, pid;

    // Cria processo filho
    child_pid1 = fork();

    if( child_pid1 == 0 ){
        // Processo filho

        fprintf(stdout, " Iniciando Processo Filho. Possui PID: %d.\n", (int)getpid() );
        fprintf(stdout, "PID do Processo Pai: %d.\n", (int)getppid() );
        fprintf(stdout, "Iniciando processo para captura de foto pela Webcam");
        system("cd /home/pi/Desktop/");
        system("./take_picture.py");
        _exit(EXIT_SUCCESS);

    }
    else if( child_pid1 < 0 ){
        // Erro ao criar processo filho
        perror("fork");
        exit(EXIT_FAILURE);
    }

    else{
        // Código do processo pai
        // Aguarda processo filho terminar e recebe status
    }
}
```

```
pi@raspberrypi: ~/Desktop
GNU nano 2.2.6 File: processos_projeto2.c Modified

// Verifica se o processo filho terminou normalmente
// Imprime o status de saída do filho
if( WIFEXITED(status) )
    fprintf(stdout, "Status de saída do filho: %d.\n", WEXITSTATUS(status) );
else
    fprintf(stdout, "Processo filho terminou de forma anormal.\n");

// Imprime dados na tela
fprintf(stdout, "Processo pai. meu PID: %d.\n", (int)getpid() );
fprintf(stdout, "Processo pai. PID do processo filho anterior: %d.\n", (int)child_pid1 );

//Iniciando código do módulo GPS
fprintf(stdout, "Iniciando AQUISIÇÃO GPS\n");
system("./GPS");

//Chamada do arquivo para enviar o email
fprintf(stdout, "Iniciando script para enviar a foto pelo e-mail\n");
system("cd /home/pi/Desktop");
system("./nmsg2.py");
fprintf(stdout, "Mensagem enviada com sucesso!!!!");
}
return 0;
```

#### Anexo IV -Código Arduino

```
#include<SoftwareSerial.h>
#include <TinyGPS.h>
SoftwareSerial serial1(10,11); //RX,TX
TinyGPS gps1;

void setup(){
    Serial.begin(9600);
    serial1.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){
        long latitude, longitude;
```

```

gps1.get_position(&latitude, &longitude);
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);
}

```

```

while(1){
    Serial.write(latitude);
    Serial.write((latitude)>>8);
    Serial.write((latitude)>>16);
    Serial.write((latitude)>>24);
    delay(1000);
    Serial.write(longitude);
    Serial.write((longitude)>>8);
    Serial.write((longitude)>>16);
    Serial.write((longitude)>>24);
    delay(1000);
}
}
}

```

Anexo V - Código Raspberry

```

#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <string.h>

#define TTY "/dev/ttyS0"

int uart0_fd;
void ctrl_c(int sig)
{
    puts(" Fechando " TTY " ...");
    close(uart0_fd);
    exit(-1);
}

int main(void)
{
    signal(SIGINT, ctrl_c);
    uart0_fd = serialOpen(TTY, 9600);
    if(uart0_fd==-1)
    {
        puts("Erro abrindo a UART. Garanta que ela");
        puts("nao esteja sendo usada por outra aplicacao.");
        return -1;
    }
    if(wiringPiSetup() == -1)
    {
        puts("Erro em wiringPiSetup().");
    }
}

```

```

        return -1;
    }
    puts("UART configurada:");
    system("stty -F " TTY);
    puts("");
    serialFlush(uart0_fd);
    //linha 38
    int i;
    float convert, convert2;
    char convertido[64];
    int a = 0, b = 0, c = 0, stop = 0;

    FILE *ponteiro;

    ponteiro = fopen("Latitude_Longitude.txt", "w+");

    while(stop < 9600){
        stop++;
        if(serialDataAvail(uart0_fd) != 0)
        {

            int j;
            i = 0;
            if(a == 0){
                fputs("Valor da latitude: \n", ponteiro);
                c++;
            }
            if(b == 4){
                fputs("Valor da longitude: \n", ponteiro);
                c++;
            }
            for(j = 0; j < 4; j++){
                i |= serialGetchar(uart0_fd) << 8*j;
                a++;
                b++;
            }
            convert = (float)i;
            convert2 = convert/100000;
            sprintf(convertido, "%f", convert2);
            fputs(convertido, ponteiro);
            putc('\n', ponteiro);

            if(c == 2){
                a = 0;
                b = 0;
                c = 0;
            }
        }
        delay(1);
    }
    serialClose(uart0_fd);
    fclose(ponteiro);
    return 0;
}

```

