

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 mapack 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 fazer a conexão direta entre o módulo GPS e a Raspberry. O objetivo foi alcançado e os resultados obtidos foram satisfatórios e atenderam os requisitos do projeto.

VI DESCRIÇÃO DE HARDWARE

O circuito de ligação entre o módulo GPS e a Raspberry está esquematizado na figura 1 abaixo.

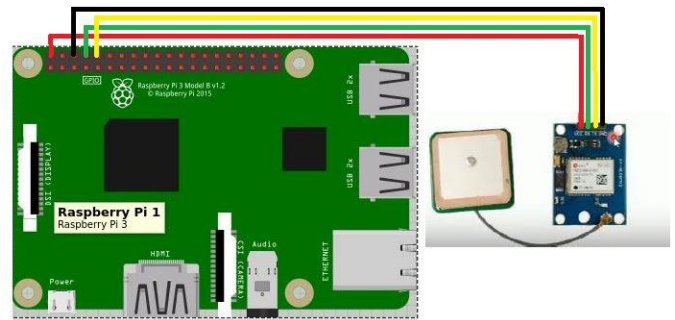


Figura 1: Ligação Raspberry módulo GPS.

A comunicação entre a Raspberry e o módulo é do tipo UART, dessa forma a porta RX do módulo (fio verde) foi ligada a porta TX da Raspberry (pino 8), a porta TX do módulo (fio amarelo) foi ligada a porta RX da Raspberry (pino 10), o VCC do módulo (fio vermelho) ligado ao pino 2 da Raspberry e por fim o GND do módulo (fio preto) ao GND da Raspberry (pino 6). Nesse esquema de conexão não foi necessário fazer um circuito divisor de tensão pois a porta TX do módulo opera com 3.3V, que é a mesma tensão de entrada que os pinos da Raspberry suportam.

VII DESCRIÇÃO DE SOFTWARE

Com a ligação direta entre o módulo e a Raspberry, o código implementado ficou mais simples e menor, como pode ser visto no Anexo IV. Para fazer a leitura das coordenadas do receptor GPS foi utilizada a biblioteca *gps.h*. Essa biblioteca trabalha da seguinte forma; a cada período de tempo o receptor recebe um pacote de dados contendo várias informações, essa biblioteca tem o papel de pegar esse pacote de dados e separar e transformar em informações que possam ser utilizadas, como a latitude e a longitude. Essa biblioteca também é capaz de fazer a comunicação UART com o módulo.

Na função *main()* o código começa com a inicialização da biblioteca com a função *gps_init()*, que também inicia a comunicação UART. Logo depois é criada uma variável *data* do tipo *loc_t* que é específico da biblioteca *gps.h*, em seguida uma variável do tipo *int* que irá definir quantos *loops* o programa vai fazer, duas variáveis do tipo *char* que guardam as conversões das coordenadas, isso é necessário pois a coordenadas são do tipo *float* e para serem salvas em arquivo *.txt* precisam ser do tipo *char*. Em seguida é criado um ponteiro para arquivo que vai salvar a latitude e a longitude, esse arquivo é aberto e então o programa espera 3 segundos até iniciar o loop *while*, foi dado esse tempo para garantir que o receptor receba os dados corretamente. O laço *while* será executado 2 vezes, no laço é chamada a função *gps_location()* que busca todas as informações disponíveis e coloca na variável *data*. A função *sprintf* é então usada para converter a latitude (*data.latitude*) do tipo *float* para o tipo *char* na variável *converte_lat*, o mesmo é feito para a longitude, após esse procedimento os valores da latitude e da longitude são colocados no arquivo, isso é feito 2 vezes para garantir a confiabilidade dos dados recebidos. Terminado o laço *while* o arquivo é fechado e o programa encerrado.

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 o Sistema só é ativado no momento em que é reconhecida 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 as coordenadas. Por fim, verificou-se que o modulo GSP funcionou corretamente na Raspberry. Este retornou a localização exata em todas as vezes em que foi testado, informando as coordenadas de latitude e longitude.

IX CONCLUSÃO

Com a integração da Raspberry com o módulo GPS e os testes realizados, o projeto entra em sua fase final sendo que a próxima etapa a ser realizada é a implementação de todo o sistema no automóvel. Será feito um circuito que habilitará ou não o sistema, onde o ocupante do veículo pressiona um botão para manter o sistema desligado, caso esse botão não seja pressionado o sistema é habilitado após um determinado período de tempo e então é tirada a foto do ocupante do veículo e as coordenadas começam a ser enviadas para o e-mail do proprietário. É importante salientar que os objetivos estabelecidos no ponto de controle anterior foram alcançados neste.

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. 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. 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. 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. 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
```

```
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= MIME multipart()
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      ^N 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)

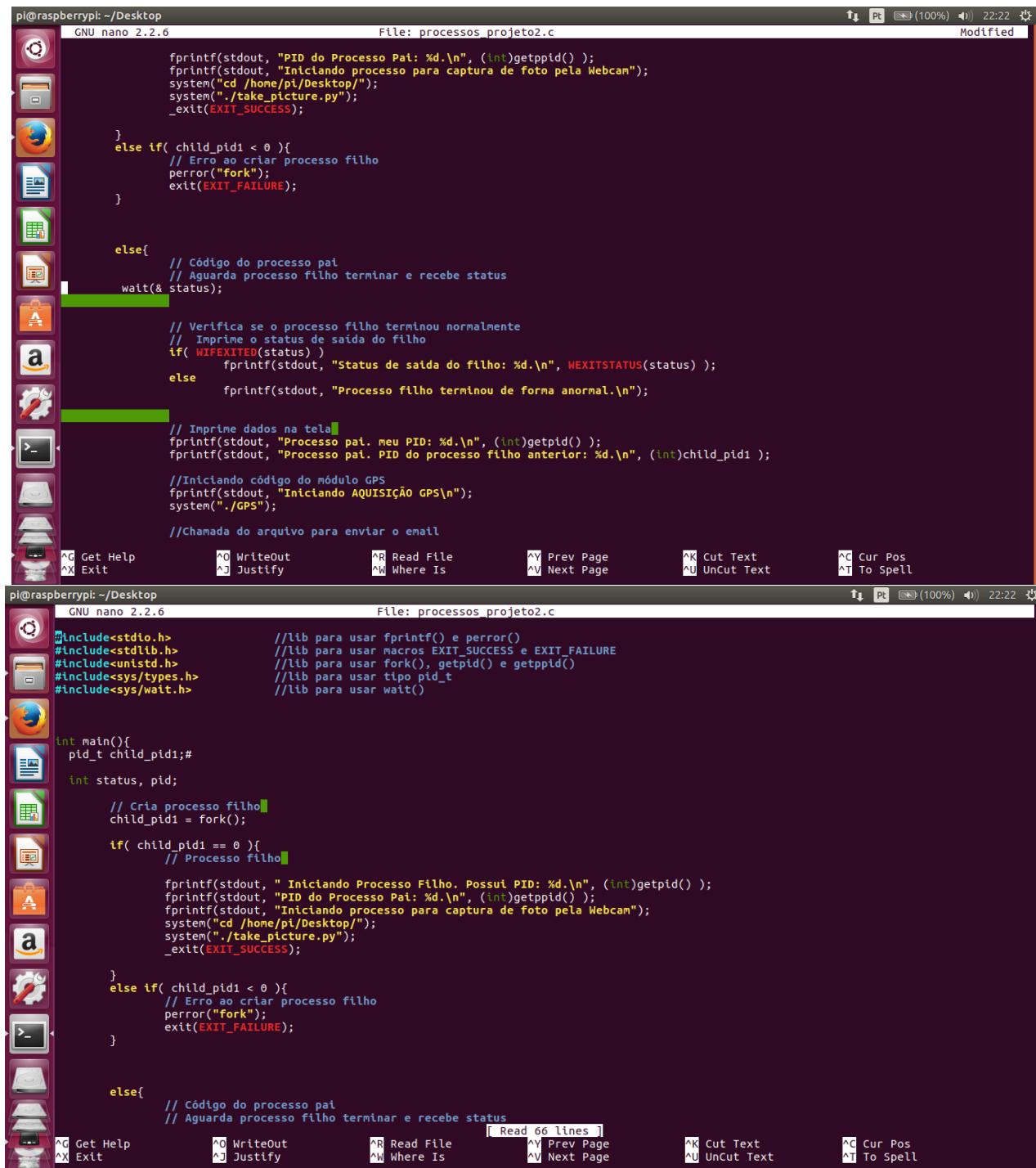
msgText= MIMEText('<b>{}</b><br><br>'.format(body, imgFilename,txtFilename), 'html')
#msgText= MIMEText('<b>{}</b><br><br>'.format(body,txtFilename), 'html')
msg.attach(msgText)

#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                               Modified

#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
        Read 66 lines
        ^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 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("./mymsg2.py");
fprintf(stdout, "Mensagem enviada com sucesso!!!!");
}
return 0;
```

Shortcuts:

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

```
#include <gps.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){

    gps_init();
    loc_t data;
    int stop = 0;
    char converte_lat[64];
    char converte_long[64];
    FILE *coordenadas;

    coordenadas = fopen("latitude_longitude.txt", "w+");

    sleep(3);

    while(stop < 2){

        gps_location(&data);
        printf("%f %f\n", data.latitude, data.longitude);
        sprintf(converte_lat, "%f", data.latitude);
        sprintf(converte_long, "%f", data.longitude);
        fputs("Valor da latitude: ", coordenadas);
        fputs(converte_lat, coordenadas);
        fputc('\n', coordenadas);
        fputs("Valor da longitude: ", coordenadas);
        fputs(converte_long, coordenadas);
        fputc('\n', coordenadas);
        stop++;
        sleep(1);
    }
    fclose(coordenadas);
    return 0;
}
```

