

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 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—monitoramento, 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 que faz o reconhecimento facial do condutor do veículo, 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 .
- 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 realizar a captura de foto, desenvolveu-se um código em Python para apenas realizar a captura da foto 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, denominado `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 carrega todas as características para reconhecimento facial através do pacote `face_cascade` por meio do atributo `cv2.CascadeClassifier`. Para que estes atributos sejam capturados em um frame, definiu-se o tipo de arquivo de imagem e o diretório por meio da variável `file`. Para condicionar a integração da webcam e a biblioteca `open CV`, utiliza-se a função `cv2.VideoCapture(0)` por meio do atributo `Cap`. É dentro deste atributo em que será definido as variáveis utilizadas pela biblioteca para determinar a escala de conversão de cor, o frame a ser capturado, além de determinar a escala de qualidade para conversão da imagem, os pixels de vizinhança e seus respectivos tamanhos, tudo isso dentro da condição `while` para realizar a leitura e gravação das características do rosto. No momento em que o rosto é capturado, por meio do laço `for`, cria-se um retângulo com as dimensões da face detectada e, então, realiza-se a captura da foto pelo comando `imwrite`. A transmissão em tempo real com delay de 2s é realizada pelo comando `imshow`. Após a retirada da foto, o processo aguarda 5s, por meio da função `WaitKey` para então, ser finalizado. É importante ressaltar que se o programa não reconhece a face, este fica preso dentro da estrutura de repetição, impedindo que um objeto seja confundido com a face humana.

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 pe 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 componentes de imagem, coordenada e Sistema de envio por email, para simples teste de funcionamento, optou-se por desenvolver um código que realiza a chamada por meio de função no terminal através de processos pai e filho. No código presente no anexo II, 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 do email, onde serão anexadas a foto e as coordenadas. Segue em anexo o código comentado.

VI DESCRIÇÃO DE HARDWARE

A primeira parte do hardware desenvolvida foi o circuito de ligação entre o módulo GPS e a Raspberry está esquematizado na figura 1 abaixo.

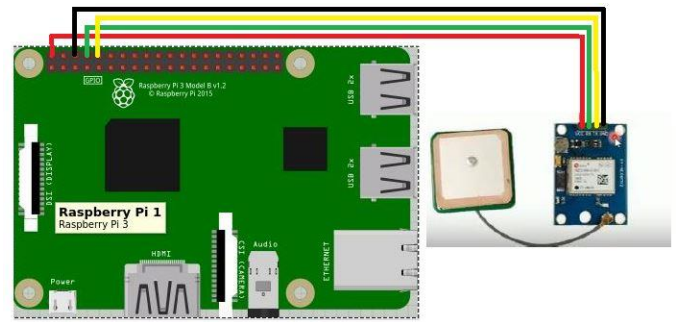


Figura 1: Ligação Raspberry módulo GPS.

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.

A segunda parte do hardware desenvolvida está no esquema da figura 2.

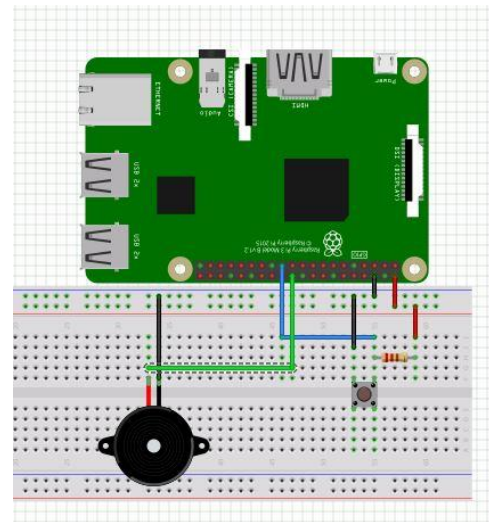


Figura 2: Circuito de alerta para desligar a Raspberry.

Após ser dada a partida no carro a Raspberry leva aproximadamente um minuto para tirar a foto e mandar as coordenadas, esse tempo foi estabelecido pois é o tempo médio que o módulo GPS leva para conseguir sinal. Antes que seja atingido esse tempo um buzzer (conectado na porta 22) é disparado para alertar o proprietário que a Raspberry deve ser desligada e impedir que ela realize suas funções, esse buzzer fica ativo por 3 segundos. Para desligar o Sistema foi utilizado um push button e um resistor de PULL UP, como mostrado na figura, que com recursos de software identifica a borda de descida e desliga a Raspberry. Se após o apito de buzzer o condutor não pressionar o botão a Raspberry executa o

programa mandando o e-mail, caso o botão seja pressionado após o apito ela é desativada.

VII DESCRIÇÃO DE SOFTWARE

O Sistema utiliza 3 scripts atividades que serão executados por processos na raspberry e 2 códigos que serão executados no boot desta. O primeiro script em python refere-se ao Sistema para retirar a foto, já o segundo script em C é usado para a leitura e escrita das coordenadas do GPS em um arquivo LATITUDE_LONGITUDE.txt e o último script refere-se ao Sistema para realizar o envio de email para o motorista. Em relação aos códigos que serão implementados no boot da rasp, o primeiro trata-se de um arquivo em C responsável por realizar o desligamento da raspberry assim que o usuário aperta o botão. Por fim, o Segundo arquivo é responsável por ativar o buzzer, por 3 segundos, assim que a raspberry estiver ligada de modo a alertar o motorista que o Sistema deve ser desligado, uma vez que este só precisa estar ligado quando um indivíduo estranho está no carro. O código em linguagem C resultante destes 3 scripts são colocados no cronograma de processos da raspberry para serem executados a cada 1 minuto, por meio do arquivo crontab. Já os arquivos para boot são acrescentados no ficheiro /etc/rc.local, cujo é executado logo após a inicialização da raspberry por meio da simples colocação do diretório do programa e inicialização deste como super-usuário, com & no final do comando de modo a permitir sua inclusão no boot, como verifica-se no ANEXO VII.

O primeiro arquivo para boot foi chamado de desl_rasp. O código em python é responsável por realizar a configuração de entrada, saída das portas digitais da raspberry, além de setar o tipo de interrupção desejada, por borda de descida ou subida. Como verificado no item anterior, utilizou-se um resistor de Pull-up para verificar a interrupção de borda de descida do botão. Assim, no momento em que o usuário aperta o botão, a borda de descida é identificada e ativa-se o subprocesso Call que por meio do comando “Halt” desliga a raspberry. Para setar os comandos das portas GPIO, usou-se a biblioteca python RPi.GPIO, onde definiu-se o pino BCM23 da rasp como pino de entrada e com resistor de Pull up associado. Dentro de um loop infinito, utilizou-se o commando GPIO.eait_for_edge para que o pino capturasse a borda de descida do botão, para assim ativar o subprocesso Call para desligar a raspberry.

O Segundo arquivo para boot na raspberry foi chamado de buzzer. O código em linguagem C utiliza chamadas de Sistema, o comando Linux “echo” para ativar o pino 22 como saída digital, seu valor de nível lógico alto e a sua direção pelo interval de 3 segundos. Este intervalo refere-se a notificação que o motorista recebe, informando-lhe que a raspberry está ligada e que deve ser desligada. Após esse intervalo de tempo, inicia-se os processos com os scripts para execução da captura de foto, captura das coordenadas GPS e envio destas informações para o email.

O primeiro processo descrito no código é o processo filho. Este é responsável por realizar o processo de captura da foto com o

reconhecimento facial (toda a metodologia de implementação, está descrita no item IV tópico A) cuja é salva com permissão de acesso. Este é acessado pelo processo filho por meio da chamada de Sistema, pela função System. Este processo só retorna ao processo pai no momento em que ocorre a identificação da face do indivíduo, caso contrário, fica retida no loop. Ao sair desta condição, o programa executa o processo pai, onde são realizados os próximos scripts.

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, separar e depois 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 *sprint* 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. Este é chamado pelo processo pai por outra chamada de Sistema, denominada pelo executável *./GPS*.

Por fim, realizada a escrita das coordenadas GPS, é executado o script, por chamada de sistema, *mymsg.py*. Este contém estrutura python necessária para conecta-se ao MTA e os pacotes de configuração de corpo de email de arquivos multimedia, como pode-se observar no ANEXO II.

Realizada as considerações acima, o código foi salvo, compilado e criado um objeto executável denominado de *processos_projeto2*. Este arquivo é executado dentro do agenciador de tarefas da raspberry, chamado Crontab. É uma daemon que quando ligada, permite que certos arquivos ou scripts permaneçam em stand by e, em determinado horário, as executam em background. Esta pode ser configure em 5 itens: minute, hora, dia do mês, mês e usuário em que o Sistema realizará a tarefa. Como deseja-se saber o trajeto realizado pela pessoa que roubou o carro, configurou-se a daemon para o usuário da raspberry, todos os dias da semana, todos os meses, todos os dias e a cada 1 minuto. Sua implementação é bem

simples, requer apenas a inclusão das configurações citadas anteriormente e o comando para executar o script.

VIII RESULTADOS

Esperava-se nessa etapa do projeto desenvolver o circuito de alerta para o motorista onde seria possível desabilitar o funcionamento do Sistema, fazer o programa ficar em loop mandando a cada 1 minuto para o e-mail a foto e as coordenadas do GPS, esta etapa é extrema importância, pois caso o veículo seja roubado o proprietário receberá a localização atualizada do veículo a cada 1 minuto, e por último colocar a função para execução automática do programa pela Raspberry. Todos os objetivos foram alcançados com sucesso e todos os sistemas integrados funcionaram de forma correta.

IX CONCLUSÃO

O projeto proposto obteve êxito na sua construção e funcionamento. Porém, ainda não foi possível implementar o Sistema no veículo, pois a corrente fornecida pela tomada de 12V do carro não é suficiente para o funcionamento da Raspberry com a câmera e o GPS. Uma possível solução para resolver esse problema seria conectar diretamente a alimentação do Sistema na bateria do carro ou trocar o fusível da tomada por outro que permite a passagem de uma corrente mais elevada, porém fazer essas modificações necessitariam de um conhecimento mais aprofundado em sistemas automotivos e de maior tempo para aperfeiçoar o projeto.

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/smtp-to-send-emails. Acessado em 08 de Maio de 2017.
- [18] Disponível em : <http://cadernodelaboratorio.com.br/2015/06/10/inicializando-um-programa-automaticamente-no-raspberrypi/>. Acessado em 21 de Junho de 2017.

- [19] Disponível em : <https://stackoverflow.com/questions/42326042/face-recognition-using-python-and-opencv>. Acessado em 21 de Junho de 2017.
- [20] Disponível em: <http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html> . Acessado em 21 de Junho de 2017.
- [21] Disponível em: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html. Acessado em 30 de Junho de 2017.
- [22] Disponível em: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html. Acessado em 30 de Junho de 2017.
- [23] Disponível em: <http://neylsoncrepalde.github.io/2017-02-21-reconhecimento-facial/>. Acessado em 30 de Junho de 2017.
- [24] Disponível em: <https://updatedcode.wordpress.com/2016/04/01/deteccao-facial-com-python-e-opencv/>. Acessado em 30 de Junho de 2017.
- [25] <http://www.helviojunior.com.br/it/raspberry-pi-autologin-auto-start-uma-aplicacao-e-dessabilitar-screen-blanking/>. Acessado em 30 de Junho de 2017.
- [26] Disponível em: <http://www.sistemasembarcados.org/2016/02/18/como-executar-um-comando-na-inicializacao-do-raspberry-pi-orange-pi-banana-pi/>. Acessado em 30 de Junho de 2017.
- [27] Disponível em: <https://0jln.wordpress.com/2013/05/21/executando-scripts-na-inicializacao-raspberry-pi/>. Acessado em 30 de Junho de 2017.
- [28] Disponível em: [78055/1/adding-a-shutdown-button-to-the-raspberry-pi-b](https://www.element14.com/community/docs/DOC-78055/1/adding-a-shutdown-button-to-the-raspberry-pi-b). Acessado em 30 de Junho de 2017.
- [29] Disponível em : <http://www.vitorbritto.com.br/blog/agendando-tarefas-com-crontab/>. Acessado em 30 de Junho 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
import 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?.mjpg'
#cap.open("http://10.42.0.233/?action=stream?dummy=param.mjpg")
#stream='http://10.42.0.233:8081/video?x.mjpeg'
#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")

Get Help      WriteOut      Read File      Prev Page      Cut Text      Cur Pos
Exit          Justify       Where Is       Next Page     UnCut Text    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 encerrado

cv2.waitKey(5000)

#Desativando a webcam

Get Help      WriteOut      Read File      Prev Page      Cut Text      Cur Pos
Exit          Justify       Where Is       Next Page     UnCut Text    To Spell
```



```
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("saíndo")

#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 = 
smtpPass = 

##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 = '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=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)

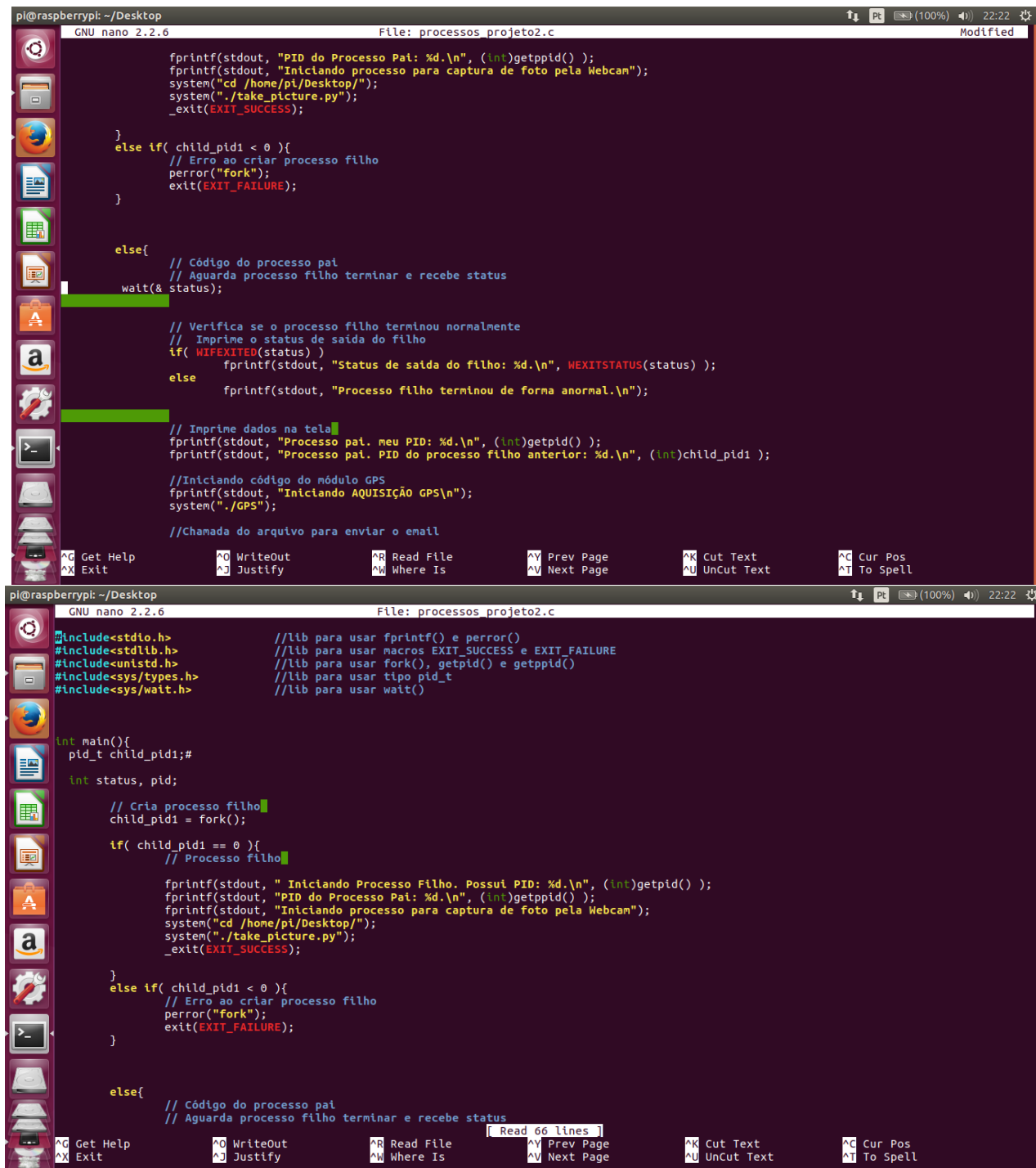
msgText= MIMEText('<b>{}</b><br><br>'.format(body, imgFilename,txtFilename), 'html')
#msgText= MIMEText('<b>{}</b><br><br>'.format(body,txtFilename), 'html')
msg.attach(msgText)

#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                               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("./nmsg2.py");
fprintf(stdout, "Mensagem enviada com sucesso!!!!");
}
return 0;
```

Anexo IV- Código para desligar a raspberry pelo botão

```
pi@raspberrypi: ~/Desktop/Douglas/Sistemas_Embarcados/Projeto
GNU nano 2.2.6 File: desl_rasp.py

#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Input teste com detecção de borda usando try

import RPi.GPIO as GPIO
import subprocess

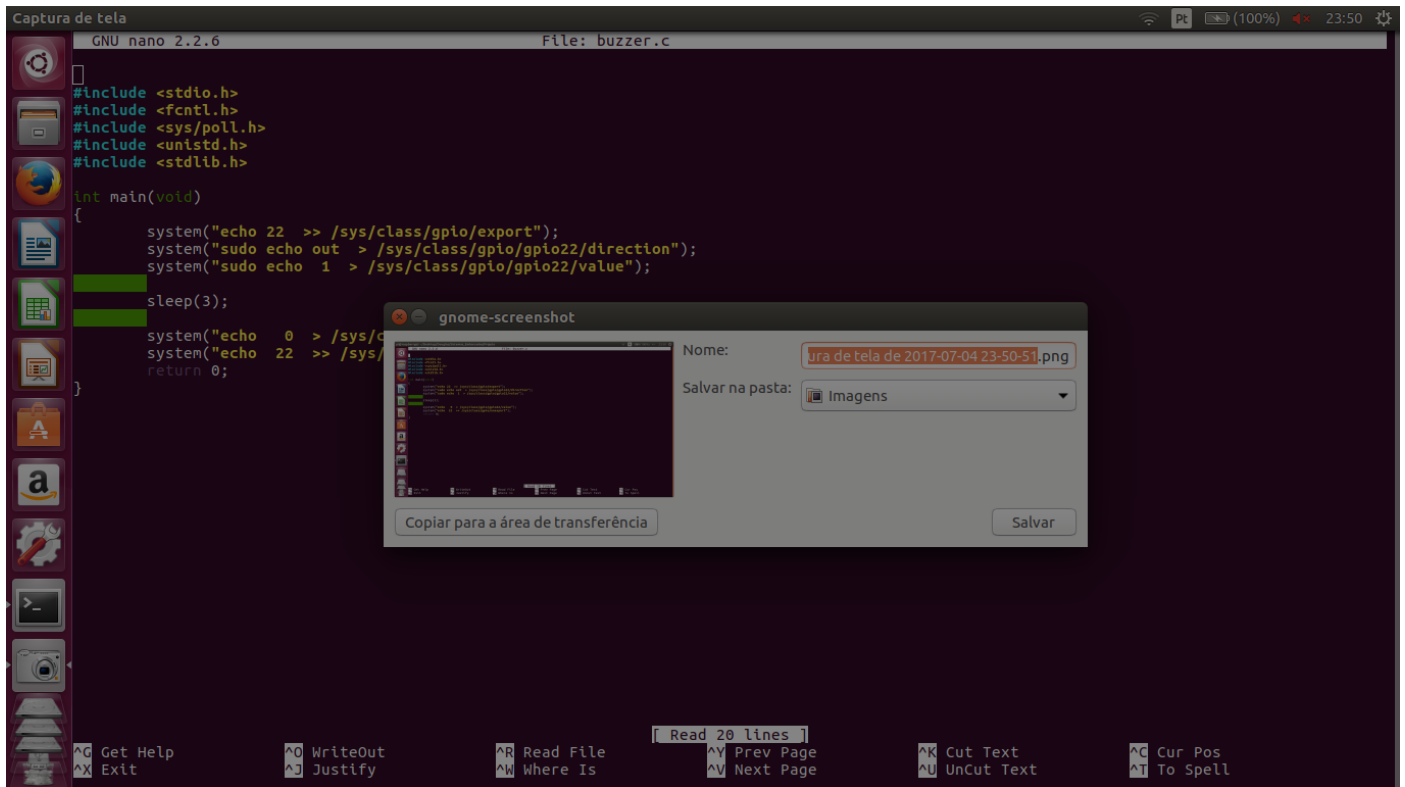
GPIO.setmode(GPIO.BCM)

SWITCH = 23

GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_UP) #com pull-down via sw

try:
    GPIO.wait_for_edge(SWITCH, GPIO.FALLING) #detecta, escreve e sai do programa
    print("Entrada em nível BAIXO")
    subprocess.call(["halt"])
except KeyboardInterrupt: #sai do programa com CTRL+c
    GPIO.cleanup()
```

Anexo V- Código para ativar o buzzer



```
GNU nano 2.2.6 File: buzzer.c
#include <stdio.h>
#include <fcntl.h>
#include <sys/poll.h>
#include <unistd.h>
#include <stdlib.h>

int main(void)
{
    system("echo 22 >> /sys/class/gpio/export");
    system("sudo echo out > /sys/class/gpio/gpio22/direction");
    system("sudo echo 1 > /sys/class/gpio/gpio22/value");

    sleep(3);

    system("echo 0 > /sys/class/gpio/gpio22/value");
    system("echo 22 >> /sys/class/gpio/unexport");
    return 0;
}
```

Nome: Jra de tela de 2017-07-04 23:50:51.png
Salvar na pasta: Imagens
Copiar para a área de transferência Salvar

Get Help WriteOut Read File Read 20 lines Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell

Anexo Vi- Arquivo de configuração do agendamento de tarefas

```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /tmp/crontab.hiTEVu/crontab Modified
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow   command
* * * * * sudo /home/pi/Desktop/./processos_projeto2 &
```

Anexo VII – Código da Raspberry

```
#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("%lf %lf \n", data.latitude, data.longitude);
        sprintf(converte_lat, "%lf", data.latitude);
        sprintf(converte_long, "%lf", 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;
}
```