Ponto de Controle 2

Sistema de Segurança Aprimorado para Cofres Pessoais

Felipe Chermont Pereira
Faculdade do Gama (FGA)
Universidade de Brasília (UnB)
Brasília, Brasil
chernox0427@gmail.com

Guilherme Simões Dias
Faculdade do Gama (FGA)
Universidade de Brasília (UnB)
Brasília, Brasil
g.simoesdias@gmail.com

***Abstract*— This project is aimed at demonstrating an improved security system for personal safes, which in addition to protecting the property of the owner also allows remote monitoring of the safe. For this, a two-stage locking system was developed, with facial recognition and a numeric keypad with a password using a Raspberry Pi to perform the necessary processing. In this system, when an access occurs, a photo of the subject's face is captured and compared with those present in the database and, if necessary, a message will be sent to the main person responsible for the safe via telegram, informing that someone unknown is trying to access it. it.**

***Resumo*— Este projeto é voltado para a demonstração de um sistema aprimorado de segurança para cofres pessoais, que além de proteger os bens do proprietário também permite o monitoramento remoto do cofre. Para isso, foi desenvolvido um sistema de tranca de duas etapas, com reconhecimento facial e teclado numérico com senha utilizando uma Raspberry Pi para realizar os processamentos necessários. Neste sistema, quando ocorre um acesso, uma foto do rosto do sujeito é capturada e comparada com as presentes no banco de dados e caso necessário, será enviado uma mensagem para o principal responsável pelo cofre via telegrama, informando que alguém desconhecido está tentando acessa-lo.**

Keywords—Segurança; Reconhecimento Facial; Raspberry PI;

I. INTRODUÇÃO

O projeto visa aprimorar a segurança dos cofres pessoais convencionais ao implementar um sistema de reconhecimento facial com uso de técnicas de "*Machine Learning*" para identificar o usuário antes de garantir o acesso. Ademais, outra medida de segurança adotada é o alerta emitido caso uma tentativa fracassada ocorra, em que um bot do Telegram que está integrado ao sistema realiza capturas de imagens da câmera e envia diretamente para o celular do proprietário do cofre, permitindo que este tome as ações necessárias. Em conjunto com a verificação fácil, o sistema conta com um sistema de senha de dígitos que permite ao proprietário garantir acesso a pessoas não registradas em caso de necessidade ou emergência.

II. JUSTIFICATIVA

Dentro de residências ou estabelecimentos existem diversos objetos que precisam ser protegidos pelo seu valor ou que precisam ser mantidos fora de alcance de determinadas pessoas que frequentam o local por questões de segurança.

Em geral, os cofres pessoais convencionais cumprem com o seu papel de restringir o acesso ao seu conteúdo, no entanto estes não fazem distinção de quem está acessando, pois geralmente requerem apenas uma senha numérica. Fato que gera uma brecha no caso de algum indivíduo descobrir a senha.

Por este motivo, uma segunda etapa de verificação com um fator mais restrito e individual pode ser implementada para limitar o acesso.

Dentre os mecanismos de identificação que se adequam a esse requisito, temos biometria, reconhecimento de íris e fácil. No entanto, o reconhecimento de íris é um processo extremamente avançado e que requer sensores específicos que não podem ser obtidos para este projeto por limitações de verba.

Contudo, entre a autenticação por biometria e a identificação facial, optamos pela segunda opção pois ela nos dá a possibilidade de implementar um sistema de alerta mais completo e com informações mais relevantes para o proprietário, como uma foto do indivíduo que está tentando abrir o cofre.

III. OBJETIVOS

Desenvolver um protótipo de um cofre com um sistema de segurança aprimorado capaz de restringir o acesso ao seu conteúdo por meio de um sistema de reconhecimento facial auxiliado por aprendizado de máquina e proporcionar uma maior segurança e confiança ao proprietário.

Além de ser capaz de gerar alertas com informações relevantes sobre quem está acessando o cofre em tempo real via mensagem pelo aplicativo Telegram diretamente para o celular do proprietário.

IV. METODOLOGIA

O projeto será divido em pontos de controle, como objetivo de analisar e marca o desenvolvimento do mesmo. Serão 4 pontos de controle, cujos são definidos da seguinte forma:

- **PC1**: Proposta do projeto (justificativa, objetivos, requisitos, benefícios, revisão bibliográfica).
- **PC2**: Realização de testes nos componentes de forma individual, utilizando as ferramentas mais básicas da placa de desenvolvimento além de bibliotecas prontas

para demonstrar a viabilidade do projeto e a capacidade do grupo de trabalhar com o hardware presente.

- **PC3**: Refinamento do protótipo, acrescentando recursos básicos de sistema (múltiplos processos ethreads, pipes, sinais, semáforos, MUTEX etc.) e iniciando a integração entre os subsistemas.
- **PC4**: Refinamento do protótipo, montagem final do projeto físico e realização de adequações.

A cada ponto de controle será realizado um relatório contendo o avanço do projeto conforme o cronograma para registrar a evolução do mesmo.

Além disso, na agenda dos membros do projeto serão dedicadas um mínimo de três horas semanais para o desenvolvimento. O projeto terá parte de desenvolvimento em *Software* e parte em *Hardware*.

1) Software: A funcionalidade deste depende da capacidade de executar as seguintes ações:

Com um módulo de câmera ser capaz de identificar rostos de indivíduos e realizar o reconhecimento fácil, usando o banco de dados de referência, com uma acurácia de no mínimo 80%.

Realizar comunicação com smartphones via internet para enviar alertas na forma de imagens registradas pelo modulo de câmera com o menor intervalo possível.

Reconhecer a sequência numérica registrada como senha para desbloquear o cofre, além de realizar comandos básicos como limpeza de caracteres e realizar nova tentativa.

2) Hardware: Este deve atendender a demanda do software, além de ser construído de forma a ter uma interface amigável e intuitiva. Também deve ser integrado a estrutura garantido que a essência do produto não seja comprometida.

V. REQUISITOS

A. *Hardware*

- 1 processos ethreads pi 3B;
- 1 Módulo de câmera;
- 1 Teclado numérico matricial;
- 1 Protoboard;
- 1 Display LCD 16x2;
- 1 SD card 16GB;
- Diversos jumpers;
- 1 Cabo de alimentação tipo micro USB;
- 1 Transformador 12V 1.5A

B. *Software*

- Integração raspberry com bots de Telegram;
- Sistema de reconhecimento facial;
- Identificação de entradas no *keypad*;
- Sistema de chave e cadeado;
- Comunicação com display.

VI. BENEFÍCIOS

O projeto demonstrará uma opção mais confiável e segura para guardar bens pessoais sem correr o risco de serem tomados por pessoas indesejadas ou indevidas. Garantindo que o proprietário do cofre tenha informações de qualidade sobre a situação dos seus bens a todo momento pelo seu smartphone.

VII. REVISÃO BIBLIOGRÁFICA

A ideia do projeto surgiu através de pesquisas acerca da utilização da raspberry pi para sistemas de segurança e de seu potencial envolvendo IoT (*internet of things*). Foi então que encontramos o projeto[1] "Sistema de monitoramento com Raspberry Pi" publicado pelo Paulo Lucas, onde foi utilizado uma Raspberry Pi, uma câmera e um sensor de movimento PIR para identificar movimentação anormal em determinado local e aciona a câmera para tirar uma foto do local.

Outra inspiração para o projeto foi o TCC[2] do Alef Kaian, onde ele descreveu como técnicas de Machine Learning e reconhecimento facial são e tendem a ser cada vez mais utilizadas como biometria para diversas aplicações.

Os projetos acima mostram como é possível utilizar técnicas de reconhecimento facial para biometria e como é possível utilizar das imagens fornecidas pela câmera associada a Raspberry Pi para desenvolver um sistema seguro para o usuário, demonstrando a viabilidade do projeto proposto.

VIII. DESENVOLVIMENTO

A. *Componentes principais*

1) Raspberry Pi 3B+: Modelo de 2018 da Raspberry Pi Foundantion com conectividade com Ethernet.

- Processador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz e 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- 40-pin GPIO header
- Conector de interface HDMI
- 4 portas USB 2.0
- Conector CSI para câmera Raspberry Pi
- Conector DSI para display capacitivo Raspberry Pi
- Saída de vídeo composto e áudio através do plug P4
- Slot cartão micro SD para carregamento do sistema operacional e armazenamento de arquivos
- Entrada de fonte DC micro USB 5V/2.5A
- Suporte a Power-over-Ethernet (PoE) (requer HAT PoE separadamente)

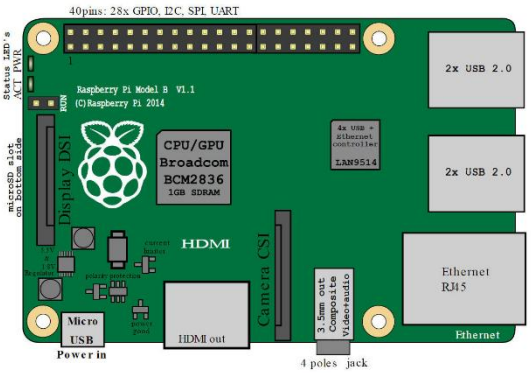

Figura 01 – Esquemático Raspberry Pi

2) Modulo de câmera: GOtech Webcam OFFICE HD 720p.

- Resolução 720P
- Ângulo da câmera 80°
- Tipo de interface USB 2.0
- Sensor 1/7" CMOS
- Vídeo chamada HD de 720p
- Correção automática de pouca luz
- Possui ajuste de foco automático
- Suporte aos sistemas operacionais Windows, Mac OS e Linux

Figura 02 – GOtech Webcam OFFICE

3) Teclado matricial

- Quantidade de Teclas: 16;
- Conector: 8 pinos (2,54mm);
- Limites de Operação: 35VDC, 100mA;
- Isolação: 100MΩ, 100V;
- Tempo de contato: <=5ms;
- Tamanho: 69,2 x 76,9 x 0,8mm;

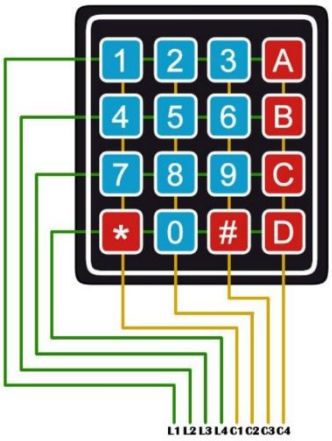

Figura 03 – Teclado matricial 4x4.

4) Display Cristal Líquido (LCD 16X02 - AZ/BR) com Módulo I2C.

- Interface: Serial I2C;
- Tensão de trabalho: 2,5V a 6V;
- Pinos de conexão: SDA, SCL, GND e VCC;
- Displays suportados: LCD 16x02 e LCD 20x04;
- Corrente em standby: 10uA;
- Tipo de controlador: Expansor de portas I/O de 8 bits I2C;
- Endereços: 0x20 a 0x27 (Padrão 0x27), pode ser alterado.

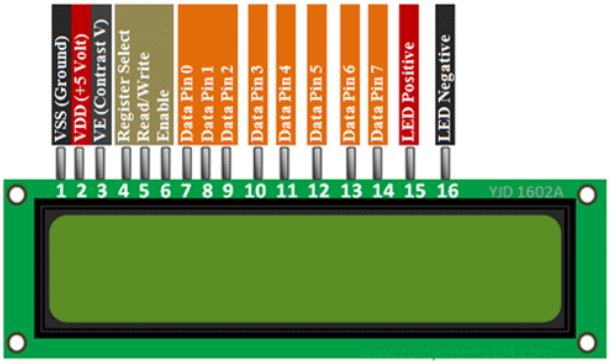

Figura 04 – *Datasheet* Display LCD 16x2.

5) Mini Fechadura Solenoide Trava Elétrica 12v.

- Tensão de Operação: 12V
- Corrente de Operação: 0.42A
- Força: 0.07KG
- Lingueta: 7mm
- Tempo máximo de acionamento: 10s
- Tempo de abertura: 1s

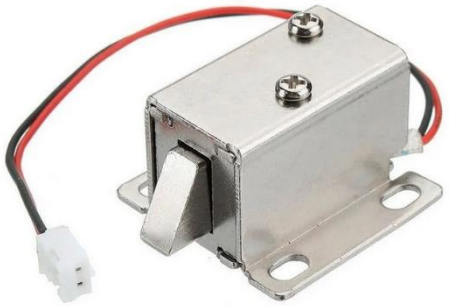

Figura 05 – Mini fechadura solenoide trava elétrica

B. Ponto de controle 2 B

A partir deste ponto de controle, o produto a ser desenvolvido foi alterado de uma fechadura de porta para um protótipo de cofre pessoal, mas o sistema de segurança e de acionamento foram mantidos e adaptados para se adequarem a nova aplicação.

Nesta etapa demos início a testagem dos dispositivos já obtidos utilizando bibliotecas prontas e códigos de projetos disponíveis de forma gratuita na internet, realizando modificações se necessário.

A) Reconhecimento facial:

Existem hoje, diversos modelos e formas de se realizar um reconhecimento facial, sendo as mais utilizadas as que utilizam técnicas de *Machine learning*, que seguem em geral etapas similares.

Primeiramente, para trabalharmos com uma máquina inteligente, é preciso que ela seja treinada para reconhecer o nosso rosto. Para isso, precisamos extrair o rosto de uma foto e armazená-lo para ser utilizado posteriormente no treinamento da máquina. Foi utilizado a biblioteca MTCNN do python para extração do nosso rosto de diversas fotos e

armazenada em uma biblioteca de faces, além de algumas extrações que serão utilizadas para validação de modelos de reconhecimento facial. Como pode ser visto:

Figura 06: faces extraídas

Após a extração e armazenamento dessas faces em uma pasta é preciso transformar cada imagem em um embedding, que transforma a imagem extraída em um vetor de 128 posições:

	0	1	2	3	4	5	6	7	8	9	...	118	119	120	121
0	0.277707	-1.283412	0.043247	1.389752	0.991944	0.428680	1.127639	-2.709962	-0.334266	1.021776	...	-1.876992	0.119716	0.296430	-0.174554
1	0.693574	-0.328871	0.447829	0.128441	0.581162	0.798301	1.496496	-0.018922	-0.235677	-0.082735	...	0.180743	-0.246003	-1.359578	1.417853
2	0.664913	-0.964368	0.408733	-0.161984	0.021216	0.074997	1.084596	0.181011	0.558638	-0.515139	...	0.726521	0.325165	0.763198	0.243011
3	1.033389	0.220684	0.237572	0.101300	0.174494	1.096997	-0.180970	-1.819630	0.141053	-0.226207	...	1.927083	-0.302668	-0.117569	1.383566
4	0.387302	-0.135145	0.348346	0.318273	-1.024203	2.037830	0.687630	-1.318440	-0.138175	-0.040490	...	-0.116166	0.381709	-0.452759	1.048633
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
77	-0.933698	-0.945418	2.151282	0.235214	-1.094787	0.131982	-0.389204	-0.364099	0.479617	-0.486738	...	-0.045312	-1.709893	-0.267989	-2.290354
78	-1.133278	-0.808195	2.549028	0.482774	-0.521533	1.421512	0.345399	-1.174166	-0.229309	0.944606	...	-1.279609	-1.041665	0.307865	-1.611843
79	-1.465724	-0.810393	3.134558	0.781290	-1.645653	0.052991	-0.466939	-0.851543	-0.302198	-0.775990	...	-1.123134	-1.223078	0.080220	-2.294381
80	-1.327975	0.183268	2.571831	0.109206	-0.398528	0.247603	-0.388200	-0.575436	-0.055788	-0.332925	...	-1.083344	-0.472297	-0.644847	-0.692956
81	-1.333094	-1.122412	2.940122	0.658895	-1.174600	0.125585	-0.190061	-0.698858	-0.324646	-0.659722	...	-0.871916	-1.385101	-0.187091	-2.125300

82 rows × 128 columns

Figura 07 - Embeddings das imagens

Tendo os *embeddings* de cada face e suas validações, podemos armazená-las em um arquivo .csv para realizar o treinamento.

Carregando as faces misturadas das pessoas a serem reconhecidas e suas validações, podemos realizar o processo de Encoder para classificar como 0 faces de uma pessoa e 1 a face da outra pessoa.

Agora, podemos realizar o processo de avaliação de Modelos para identificar o melhor modelo para o projeto, utilizando as faces misturadas, e as faces salvas na pasta de validação.

Verificamos primeiramente o modelo KNN (K-ésimo Vizinho mais Próximo) que teve 100% de reconhecimento das faces:

```
MODELO: KNN
Acuracia: 1.0000
Sensitividade: 1.0000
Especificidade: 1.0000
```

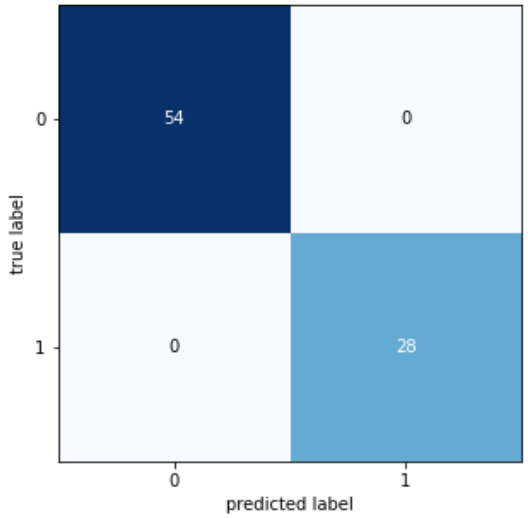

Figura 05: Matriz modelo KNN

O segundo modelo utilizado para verificação foi a uma rede neural utilizando a biblioteca Keras utilizando o TensorFlow como backend. O que também resultou em 100% de acurácia.

```
MODELO: Keras
Acuracia: 1.0000
Sensitividade: 1.0000
Especificidade: 1.0000
```

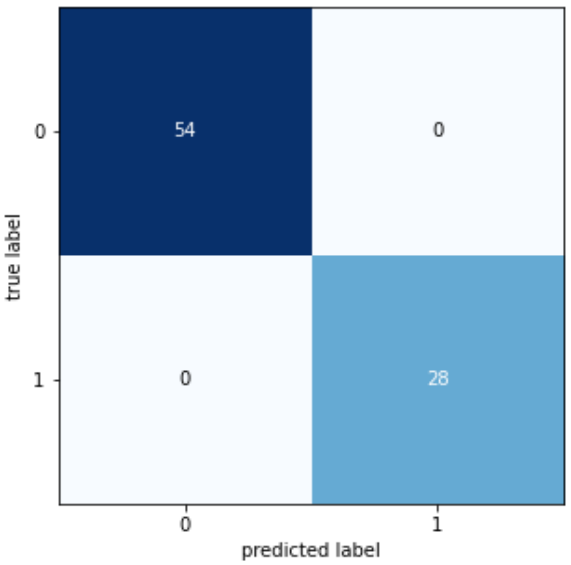

Figura 06: Matriz modelo de Rede Neural

A similaridade dos resultados entre os dois modelos se deve principalmente pela baixa quantidade de imagens utilizadas no treinamento. Apesar dos dois modelos provavelmente servirem para a aplicação, salvamos como

modelo treinado o que utiliza rede neural, por ter um processo mais eficaz à medida que o projeto tenha progresso.

Tendo o modelo treinado e a parte de reconhecimento estabelecida, é preciso apenas rodar o modelo utilizando frames da captura de vídeo de uma webcam para realização do teste.

Para isso, foi utilizado a biblioteca OpenCV do python, juntamente com as outras já mencionadas, para captura de vídeo. Após cada frame, a imagem é capturada como uma foto, é gerado o embedding dela e comparada com o modelo treinado e então retorna o resultado em tela. Como deveria ser demonstrado a partir do código em anexo [Anx01].

Porém, ainda não foi possível demonstrar os resultados, por questões de problemas com as bibliotecas do OpenCV e MTCNN para abertura da câmera e da imagem.

Apesar dos erros, o processo é bastante similar em qualquer sistema de reconhecimento facial, podendo ser adaptado ao utilizar a raspberry pi.

B) Telegram Bot API:

Para realizar a comunicação com o smartphone do proprietário encontramos uma biblioteca chamada Telepot que funciona em python 2.7 e 3 e faz uso de urllib3 para criar solicitações HTTP.

Esta biblioteca está repleta de funções como: criar e remover eventos no calendário, enviar mensagens de texto, encaminhar mensagens, enviar fotos, áudio, documentos, localização dentre muitas outras funções.

Contudo, neste projeto vamos fazer uso somente da função:

sendPhoto (*chat_id*, *photo*, *caption=None*, *parse_mode=None*, *disable_notification=None*, *reply_to_message_id=None*, *reply_markup=None*)

Que tem como parâmetros obrigatórios apenas o ID da conversa desejada e o caminho da foto, sendo que esta não pode exceder 10MB.

Essa abordagem é o suficiente para suprir a demanda do projeto. Um exemplo de código usando a biblioteca está presente no anexo [Anx02].

C) Mini Fechadura Solenoide

Esse componente não pode ser testado neste ponto de controle pois foi comprado online e sua entrega não chegou a tempo. No entanto, não serão serão usadas bibliotecas especificas para sua atuação.

REFERENCES

[1] "Sistema de monitoramento com Raspberry PI", 11 fev. Disponível em: https://www.filipeflop.com/blog/sistema-de-monitoramento-com-raspberry-pi/ Acesso em 25 Fev. 2021

[2] "Sistema de autenticação por Biometria Utilizando Técnicas de Machine Learning e processamento Digital de Sinais / Alef Kaian Feitosa Barbos

[3] "Connect a Raspberry Pi Keypad – Code Lock" Disponível em: <https://tutorials-raspberrypi.com/connecz-raspberry-pi-kecpad-code-lock/> Acesso em 25 Fev. 2020.

[4] HERTZ, Daniel. "How to Use a Keypad With a Raspberry Pi 4", 2019. Disponível em: <https://maker.pro/raspberry-pi/tutorial/how-to-use-a-keypad-with-a-raspberry-pi-4>. Acesso em 25 Fev. 2020.

[5] IDRIS. "Interface 4×4 Keypad With Raspberry Pi", 2019. Disponível em: <https://tutorial.cytron.io/2019/09/18/interface-4x4-keypad-with-raspberry-pi/>. Acesso em 25 Fev. 202

[6] "Python framework for Telegram Bot API", Disponível em: <https://telepot.readthedocs.io/en/latest/> Acesso em 25 Mar 2021

ANEXOS

Anx01 – Código Reconhecimento Facial

```
1-  import numpy as np
2-  from PIL import Image
3-  from mtcnn.mtcnn import MTCNN
4-  from tensorflow.keras.models import load_model
5-  import cv2
6-
7-  pessoa = ['LETICIA', 'FELIPE']
8-
9-  cap = cv2.VideoCapture(2)
10- num_classes = len(pessoa)
11-
12- detector = MTCNN()
13- facenet = load_model("facenet_keras.h5")
14- model = load_model("faces.h5")
15-
16- def extract_faces(image, box, required_size=(160, 160)):
17- pixels = np.asarray(image)
18-
19- x1, y1, width, height = box
20-
21- x1, y1 = abs(x1), abs(y1)
22- x2, y2 = x1 + width, y1 + height
23-
24- face = pixels[y1:y2, x1:x2]
25-
26- image = Image.fromarray(face)
27- image = image.resize(required_size)
28-
29- return np.asarray(image)
30-
31- def get_embedding(facenet, face_pixels):
32-
33- face_pixels = face_pixels.astype('float32')
34-
35- mean, std = face_pixels.mean(), face_pixels.std()
36- face_pixels = (face_pixels - mean) / std
37-
38- samples = np.expand_dims(face_pixels, axis=0)
39-
40- yhat = facenet.predict(samples)
41-
42- return yhat[0]
43-
44- while True:
45- _, frame = cap.read()
46-
47- faces = detector.detect_faces(frame)
48-
49- for face in faces:
50- confidence = face['confidence'*100]
51-
52- if confidence >= 90:
53- x1, y1, w, h = face['box']
54- face = extract_face(frame, face['box'])
55-
```

```
face = face.astype("float32")/255

emb = get_embedding(facenet, face)

tensor = np.expand_dims(emb, axis=0)

classe = model.predict_classes(tensor)[0]
prob  = model.predict_proba(tensor)
prob  = prob[0][classe]*100

user = str(pessoa[classe]).upper()

color = (192,255,119)

cv2.retangle(frame, (x1, y1), (x1+w, y1+h), color, 2)

font = cv2.FONT_HERSHEY_SIMPLEX
font_scale = 0.5

cv2.putText(frame, user, (x1, y1-10), font, fontScale =
font_scale, color=color, thickness = 1)

cv2.imshow("FACE RECOGNITTION", frame)

key = cv2.waitKey(1)

if key == 27:
break

cap.release()
cv2.destroyAllWindows()
```

Anx02 – Código Exemplo Bot Telegram

```
import sys
import time
import random
import datetime
import telepot

def handle(msg):
chat_id = msg['chat']['id']
command = msg['text']

print 'Got command: %s' % command

if command == 'command1':
bot.sendMessage(chat_id, ********)
elif command == 'command2':
bot.sendMessage(chat_id, *******)
elif command == 'photo':
bot.send_photo(chat_id, photo=open('path', 'rb'))

bot = telepot.Bot('*** INSERT TOKEN ***')
bot.message_loop(handle)

while 1:
time.sleep(10)
```