Smart Gate

Batch No: A-04

R.Hemalatha (164G1A0530)
K.Likith kumar (164G1A0549)
K.Keerthi Reddy (164G1A0537)
A.Ganesh (164G1A0526)

Project Guide:

Dr.T.Hithendra sarma, MTech., Ph.D.
Principal

Srinivasa Ramanujan Institute of Technology
Department of Computer Science & Engineering

Abstract:

- **Smart Gate Technology** is based on Embedded systems and Deep Learning. It involves face recognition and raspberry pi. Smart gate is used to allow only valid persons.

Literature survey:

- OpenCV tutorials point
- Raspberry pi tutorials point

https://www.researchgate.net/publication/220566092_Face_Recognition_A_Literature_Survey

https://pythonhosted.org/facereclib/references.html

Existing System:

Normal gate:

- It opens the gate without any authorization

Automatic gate:

- Just as sensor without validation

Limitations:

- No security
- Time Taking Process

Proposed System

- Raspberry pi with validation check
  - Checks the image with existing dataset
  - If it is exist then it opens the gate otherwise it does not open the gate
  - Scope of our project:
  - It is a supervised learning.
  - Only one person should stand Infront of the camera

Face recognition:

A **facial recognition system** is a technology capable of identifying or verifying a person from a digital image as a source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database.

Cntd…

- Face recognition is used to check that a person has permission to enter the organization or not. The output of facial recognition in raspberry pi is going to send power supply to stepper motor then it turns to 90 degrees that means door will open otherwise it gives sound by bursar.

Requirements:

Hardware Requirements:

RAM : 4GB

Processor : intel core i3

Raspberry pi model 3

USB camera

Stepper motor

32 GB memory card

Software requirements:

Raspberry pi OS

Python IDE

OpenCV

Planning:

Task	Date
Requirements (Abstract review)	20-12-2019
Analysis (Problem Definition , Planning , Literature survey)	25-01-2019
Design and Implementation (coding)	01-03-2020
Testing	15-03-2020
Output of project(Result)	20-03-202
Maintenance and (Deployment)	Never end process
Document submission	06-04-2020

Installation of Raspbian stretch and OpenCV

- By using below link install Raspbian stretch with OpenCV on memory card

- https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

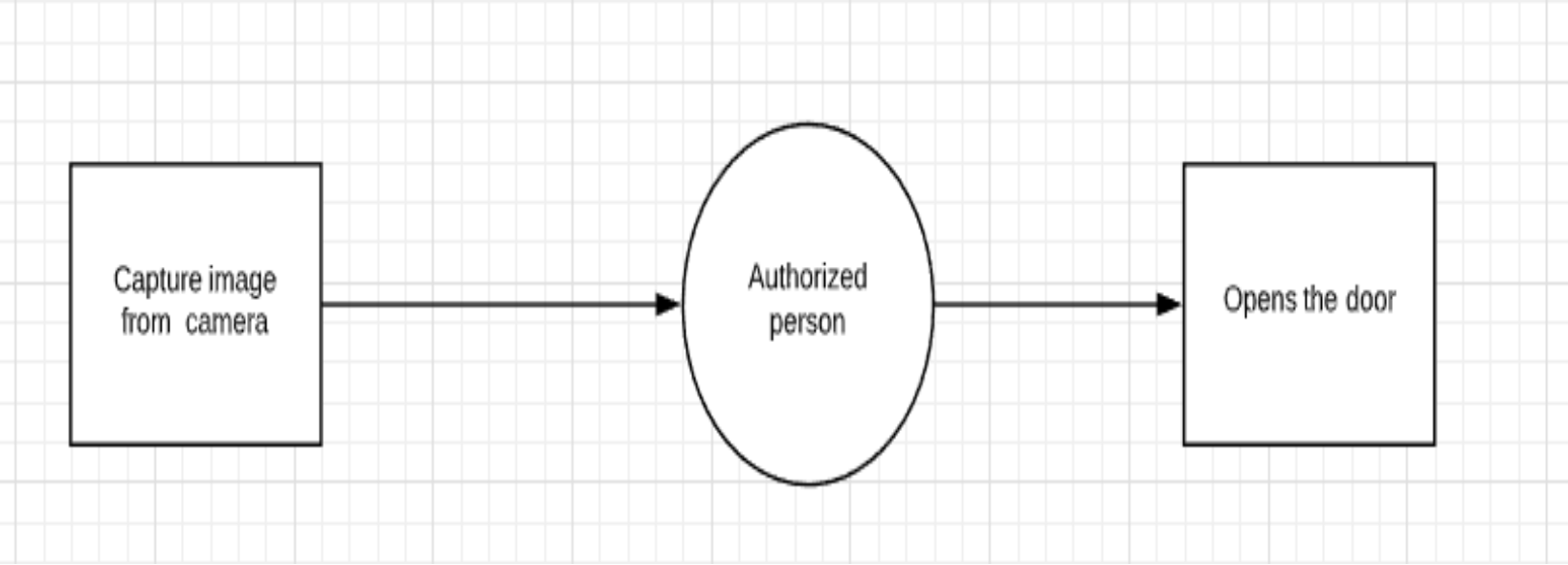
Open CV

- **OpenCV** (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision.

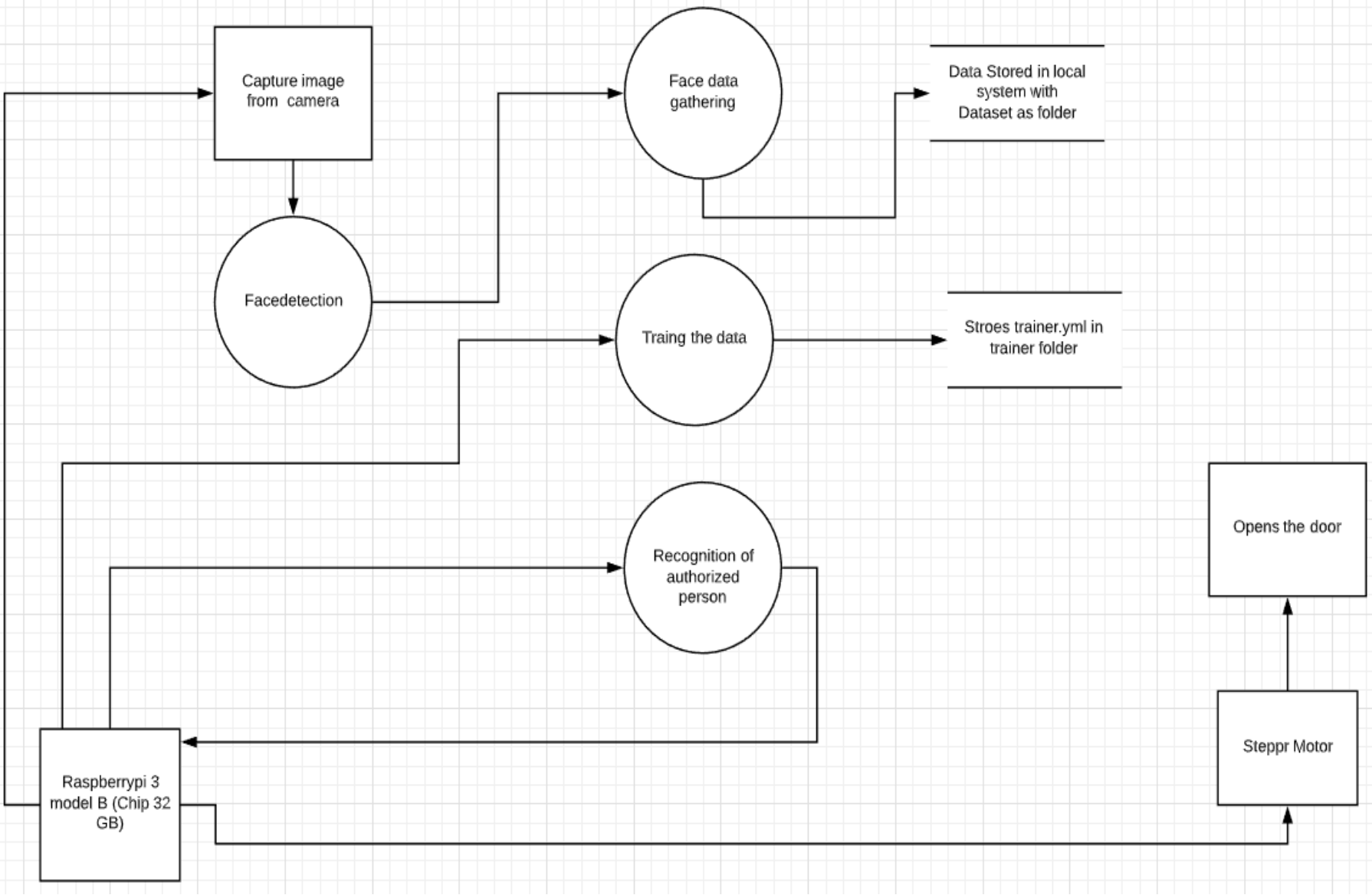
- Computer vision is a field of study which encompasses on how computer see and understand digital images and videos.

Data Flow Diagram:

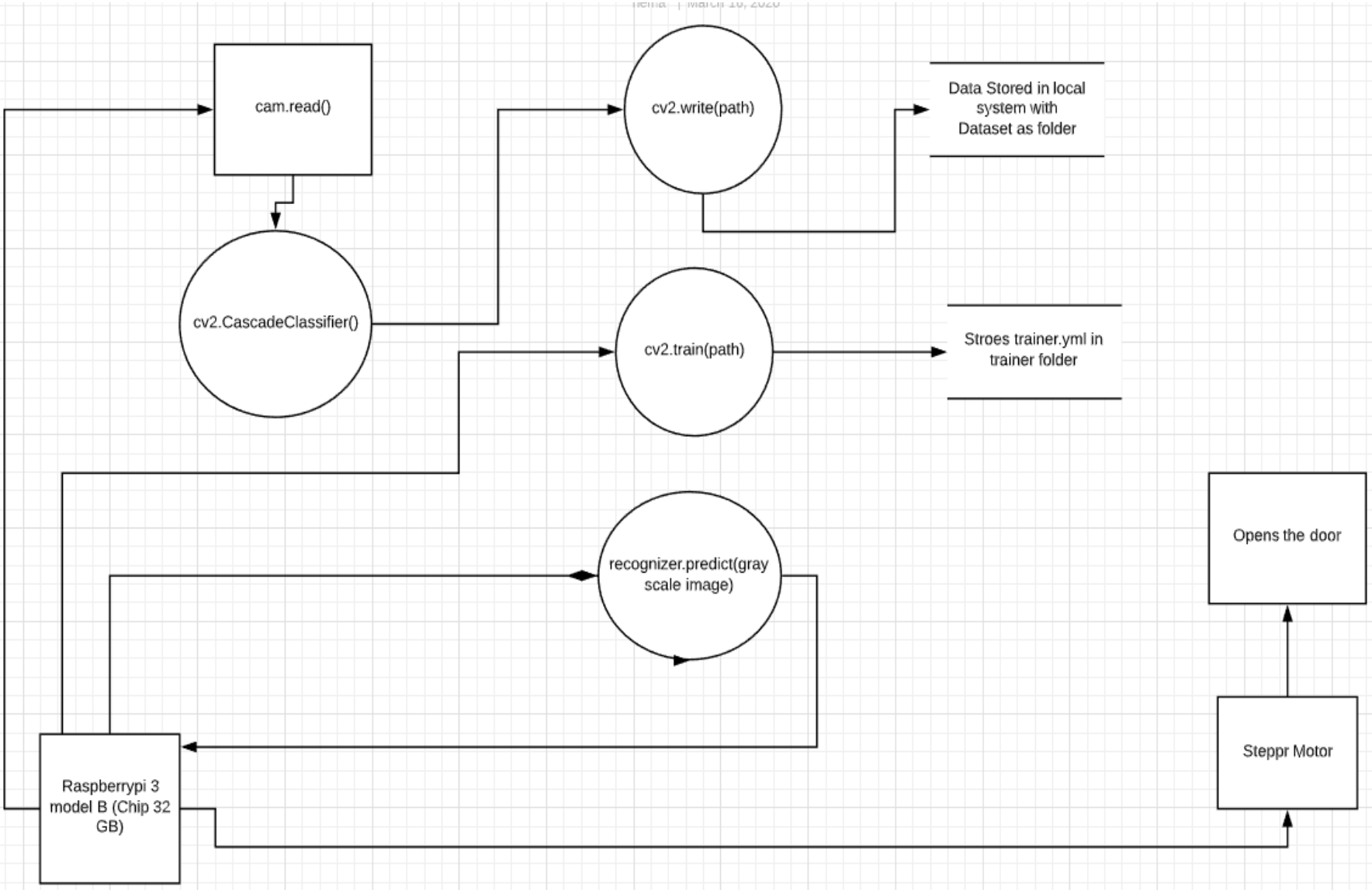
- Level-0:



Level 1:


Capture image from camera
Face data gathering
Data Stored in local system with Dataset as folder
Facedetection
Traing the data
Stroes trainer.yml in trainer folder
Recognition of authorized person
Opens the door
Steppr Motor
Raspberrypi 3 model B (Chip 32 GB)

Level 2:


cam.read()
cv2.write(path)
Data Stored in local system with Dataset as folder
cv2.CascadeClassifier()
cv2.train(path)
Stroes trainer.yml in trainer folder
recognizer.predict(gray scale image)
Opens the door
Steppr Motor
Raspberrypi 3 model B (Chip 32 GB)

Face Detection

- we need to import the **cv2** module, which will make available the functionalities needed to read the original image and to convert it to gray scale.
  - import cv2

- Syntax for xml file:
  - faceCascade = cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml')

Cntd...

- Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.

- The most common way to detect a face by using the **Haar Cascade** classifier.

- **Haar Cascade** is a machine learning object detection **algorithm used** to identify objects in an image or video(LBPH).

Cntd...

- To read the original image, We call the **imread** function of the **cv2** module, passing as input the path to the image, as a string.
- Image=cv2.imread(' C:\Users\R hemalatha/164G1A0530.jpg')
- To read the image through video,we call videocapture function.
- cap = cv2.VideoCapture(0)

Image colour

- To convert our original image from the **BGR** color space to **gray**, we use the code **COLOR_BGR2GRAY**.
  - gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

Detect Multi Scale:

- we must call our classifier function, passing it some very important parameters, as scale factor, number of neighbors and minimum size of the detected face.

```
faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    )
```

where facecascade is the xml file,

- **gray** is the input grayscale image

Cntd...

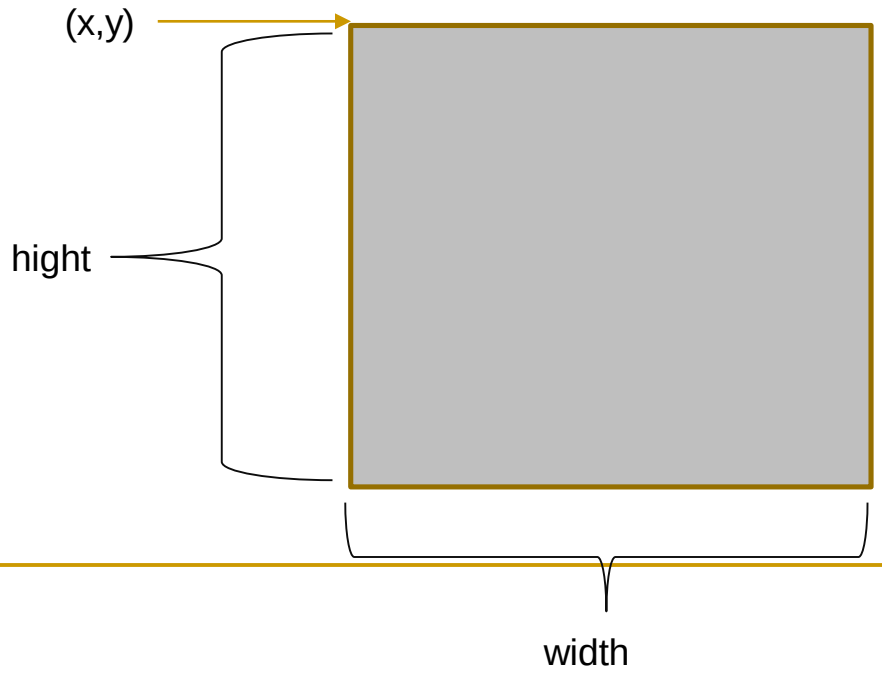
- **scaleFactor** is the parameter specifying how much the image size is reduced at each image scale. It is used to create the scale pyramid.

- **minNeighbors** is a parameter specifying how many neighbors each candidate rectangle should have, to retain it. A higher number gives lower false positives.

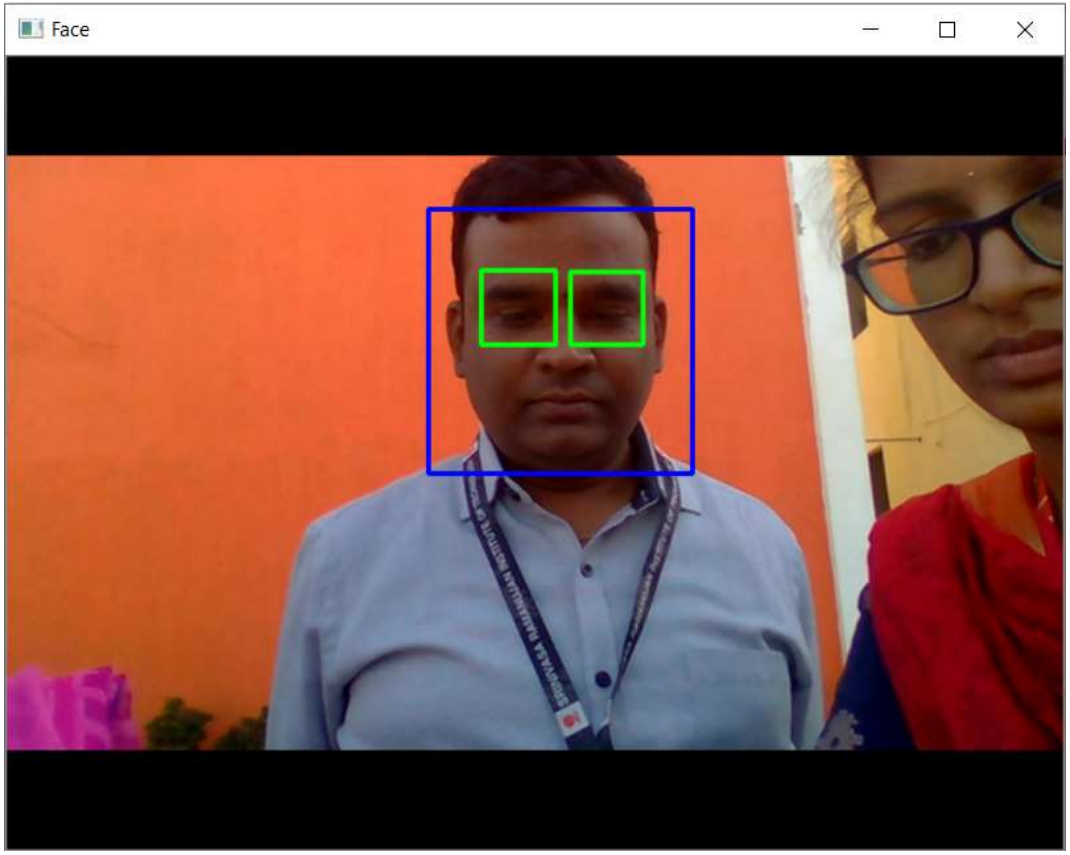
- **minSize** is the minimum rectangle size to be considered a face.

Image Shape

- To mark the faces in the image, using, for example, a blue rectangle.
  - cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

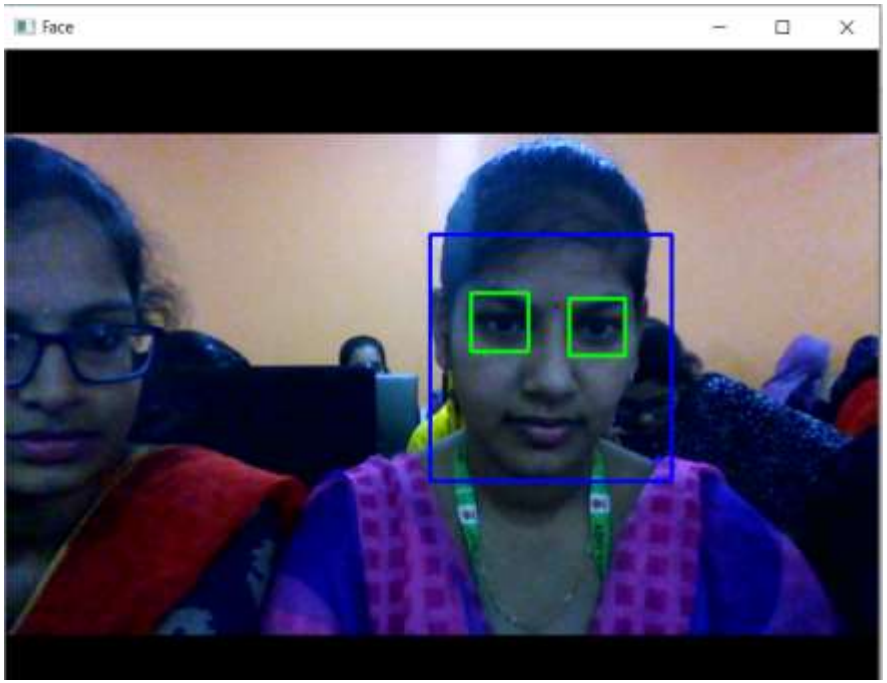


Output face with eye capture



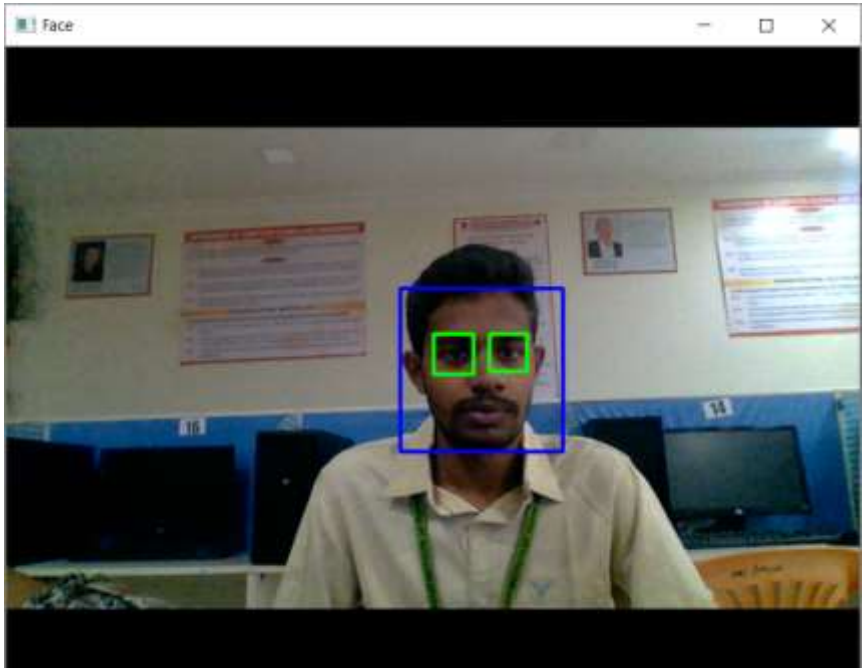
Outputs of :

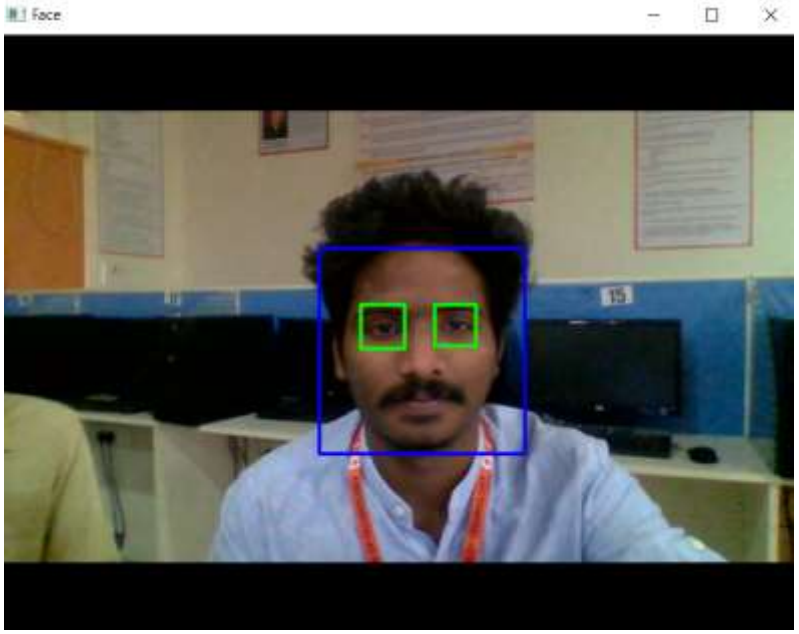
Face detect



Face and eyes detection

Cntd..

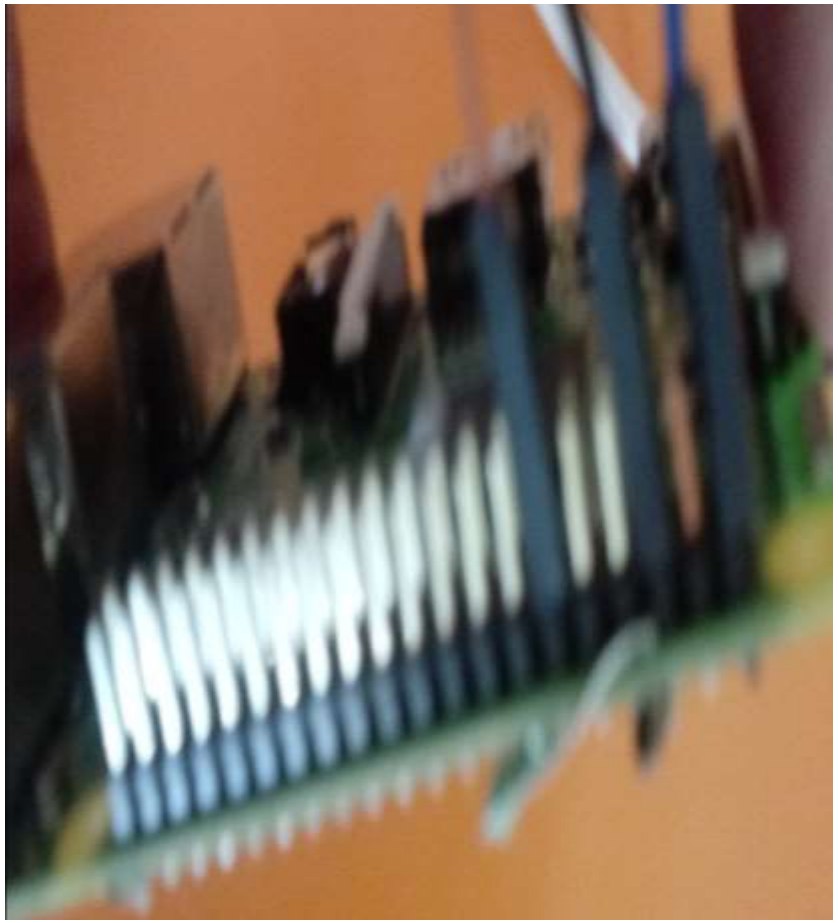




Raspberyy pi 3 Model B :

Specifications :

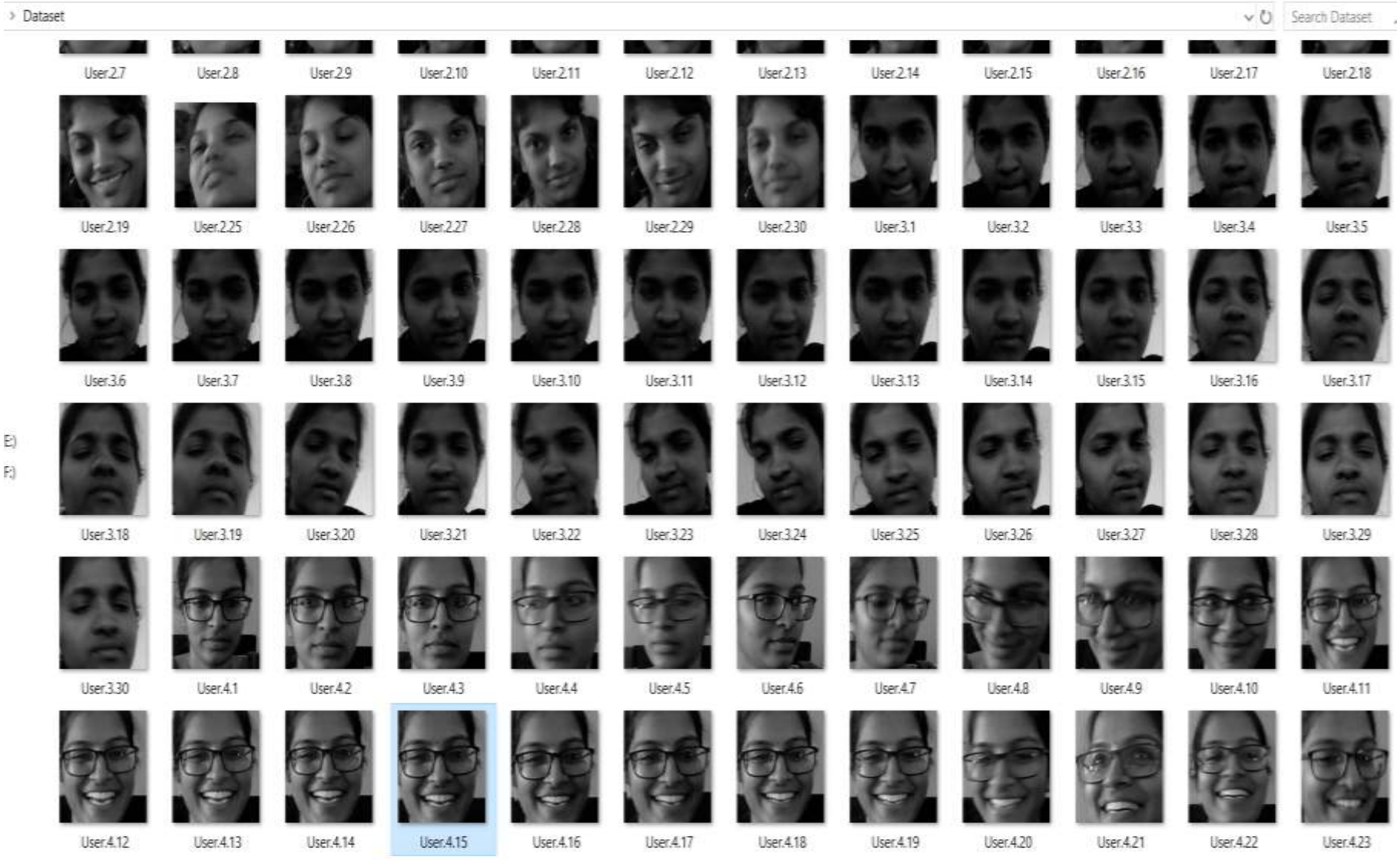
- Full size HDMI
- 1GB RAM
- 40-pin extended GPIO
- 4 USB 2 ports
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A



Data gathering:

- For each person, enter one numeric face id
  - face_id = input('\n enter user id end press <return> ==> ')
- Save the captured image into the datasets folder
  - cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
- Take 50 samples and quit

Dataset:


Dataset
Search Dataset
User.2.7
User.2.8
User.2.9
User.2.10
User.2.11
User.2.12
User.2.13
User.2.14
User.2.15
User.2.16
User.2.17
User.2.18
User.2.19
User.2.25
User.2.26
User.2.27
User.2.28
User.2.29
User.2.30
User.3.1
User.3.2
User.3.3
User.3.4
User.3.5
User.3.6
User.3.7
User.3.8
User.3.9
User.3.10
User.3.11
User.3.12
User.3.13
User.3.14
User.3.15
User.3.16
User.3.17
User.3.18
User.3.19
User.3.20
User.3.21
User.3.22
User.3.23
User.3.24
User.3.25
User.3.26
User.3.27
User.3.28
User.3.29
User.3.30
User.4.1
User.4.2
User.4.3
User.4.4
User.4.5
User.4.6
User.4.7
User.4.8
User.4.9
User.4.10
User.4.11
User.4.12
User.4.13
User.4.14
User.4.15
User.4.16
User.4.17
User.4.18
User.4.19
User.4.20
User.4.21
User.4.22
User.4.23

Training the data:

- Path for face image database
  path = 'dataset'

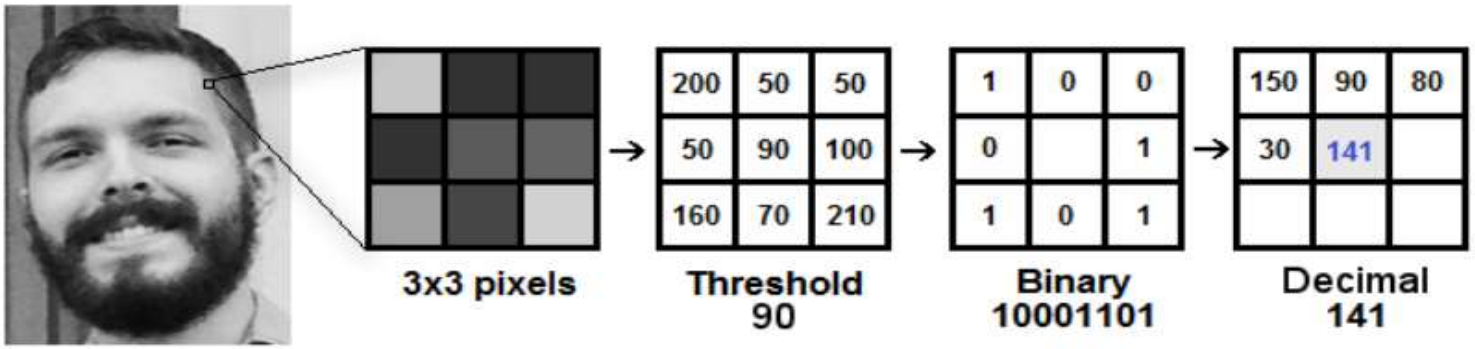
- Using LBPH algorithm
  recognizer = cv2.face.LBPHFaceRecognizer_create()

- Training the dataset
  recognizer.train(faces, np.array(ids))

- Save the model into trainer/trainer.yml
  recognizer.write('trainer/trainer.yml')

LBP:

- **Local Binary Pattern** (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.



Recognition of valid person:

- Raspberry pi interacts with GPIO pins
  - import RPi.GPIO as GPIO
- In GPIO.setup() has two parameters in that 12 pin represents the output pin of pi will interact with GPIO to show the output
  - GPIO.setup(12, GPIO.OUT)
- PWM (pulse with modulation) is a technique for controlling power. We use it here to control the amount of power going to the motor and hence how fast it spins.
  - p = GPIO.PWM(12, 50)

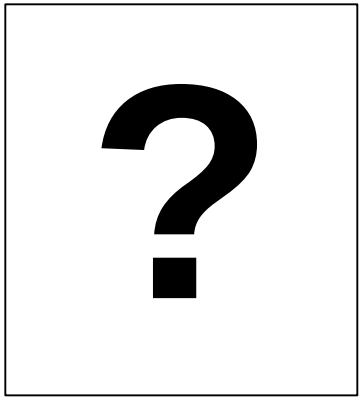
Cntd…

- The recognizer.predict (), will take as a parameter a capture portion of the face to be analyzed and will return its probable owner, indicating its id and how much confidence the recognize is in relation with this match. Recognize the face belongs to which ID
  - id = recognizer.predict(gray[y:y+h,x:x+w])
- If Id matches the existing data set then it turns to 90 degrees by using
  - p.changeDutyCycle(7.5)

References:

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html#face_detection
- https://www.superdatascience.com/blogs/opencv-face-detection
- https://www.instructables.com/id/real-time-face%20-recognition-an-end-to-end-project/

Queries



THANK YOU