THE NATIONAL INSTITUTE OF ENGINEERING
MYSURU – 570008

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

IOT Project [EC6E104] – VI Semester

Report on

GOD'S EYE

under the guidance of

Mr. Puneeth S
Assistant Professor, Dept. of ECE

Ms. Nandini S P
Assistant Professor, Dept. of ECE

Submitted By:

Manu M B	**4NI18EC040**
Muhammed Mustafa M C	**4NI18EC043**
Shujan Pannag J	**4NI18EC083**

ABSTRACT

God’s eye is an Internet of Things(IoT) project implemented to monitor the person at the doorstep. When a person arrives at the doorstep, the Raspberry Pi recognizes the person and sends the name of the recognized person, date, and time to the owner’s email. This will notify who is at the doorstep to the owner even when he or she is far from home. The logs can be viewed anytime.

CONTENTS

Chapter 1

INTRODUCTION

The home surveillance system is one of the Internet of Things(IoT) applications that attributes to a comfortable life. In this project, we are implementing a doorstep monitoring system using a credit-card-sized computer called Raspberry Pi. A camera module attached to the Pi is used to capture continuous frames of the person at the doorstep. It compares the frame with the model trained on images of family, friends and recognizes them using OpenCV. The name of the person, date, and time are collected and uploaded to the cloud. The data is stored in the database called PostgreSQL and the data is sent to the owner's email through an SMTP API built using Golang. The user can access the data log anytime.

Chapter 2

HARDWARE

2.1 Raspberry Pi 4B+

A Raspberry Pi is a tiny computer about the size of a deck of cards. It uses what is called a system on a chip(SoC), which integrates the CPU and GPU in a single integrated circuit, with the RAM, USB ports, and other components soldered onto the board for an all-in-one package. The Raspberry Pi 4 rocks a 1.5 GHz quad-core ARM CPU, a 500 MHz VideoCore VI GPU, and 4GB of RAM.

2.2 Raspberry Pi Camera v2

The Camera Module can be used to take high-definition video, as well as stills photographs. It has a Sony IMX219 8-megapixel sensor (compared to the 5 megapixel OmniVision OV5647 sensor of the original camera).

Chapter 3

SOFTWARE

3.1 Node Side

3.1.1 Python 3

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as scripting to connect existing components together.

3.1.2 OpenCV

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation.

3.2 Server Side

3.2.1 Golang

It is an open-source procedural programming language. Programs in Go are assembled by using packages, for efficient management of dependencies. This language also supports an environment adopting patterns alike to dynamic languages.

3.2.2 Gin – Web Framework

Gin is a web framework written in Go (Golang). It features a martini-like API with performance that is up to 40 times faster thanks to httprouter.

3.2.3 GORM – ORM library for Golang

The GORM is an ORM library for Golang and aims to be developer-friendly. It is an ORM library for dealing with relational databases. This GORM library is developed on the top of the database/sql package.

Chapter 4

INSTALLATION

4.1 Node Side

Python is installed on Raspberry Pi using the following command:

```
$ sudo apt install python3
```

OpenCV packages for Python are installed using the Python package manager called ‘pip’ using the following command:

```
$ pip3 install opencv-python
$ pip3 install opencv-contrib-python
```

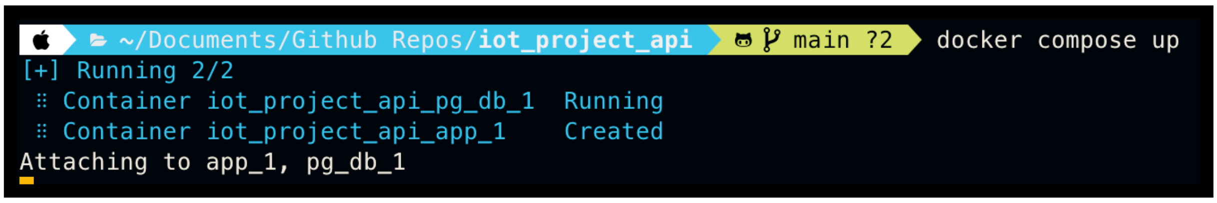
4.2 Server Side

Clone the API repository from GitHub

```
$ git clone https://github.com/shujanpannag/iot_project_api.git
```

Build the API service image and Database image using the following command:

```
$ sudo docker compose up
```



```
 ~/Documents/Github Repos/iot_project_api   main ?2  docker compose up
[+] Running 2/2
 ⠿ Container iot_project_api_pg_db_1  Running
 ⠿ Container iot_project_api_app_1    Created
Attaching to app_1, pg_db_1
```

Figure 4.1: Docker spawns API service at port 8080 and Database server at port 5432

Chapter 5

IMPLEMENTATION

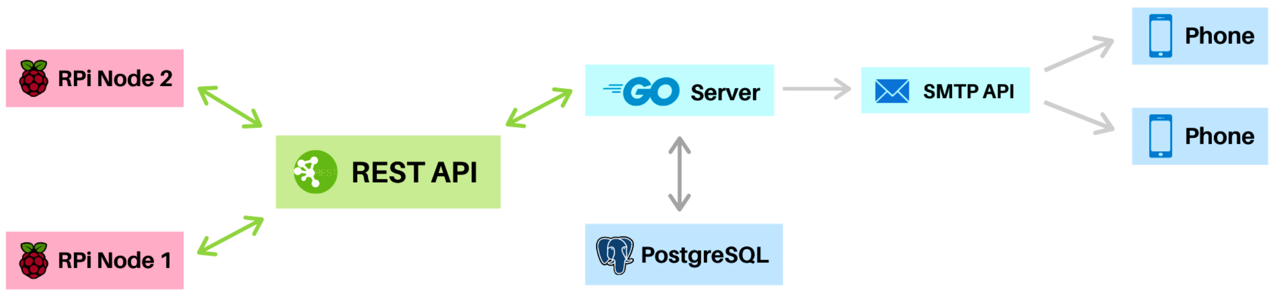


Figure 5.1: Architecture

The official Raspberry Pi OS is written to the MicroSD card using a software tool called balenaEtcher. The 'ssh' and 'wpa_supplicant.conf' file are added with the WiFi credentials mentioned. The MicroSD card is then inserted into the Pi and the camera module is attached to the CSI port. Pi is then powered up and updated with the necessary packages needed for the implementation.

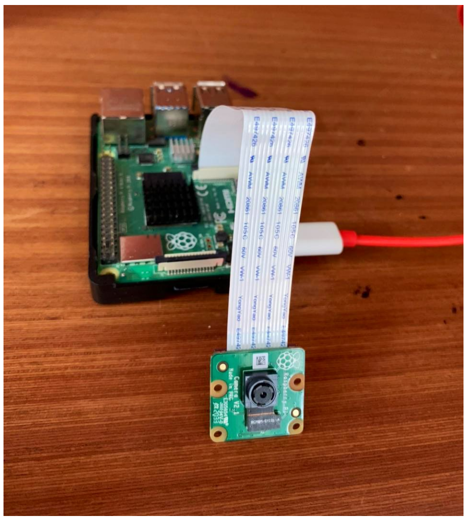
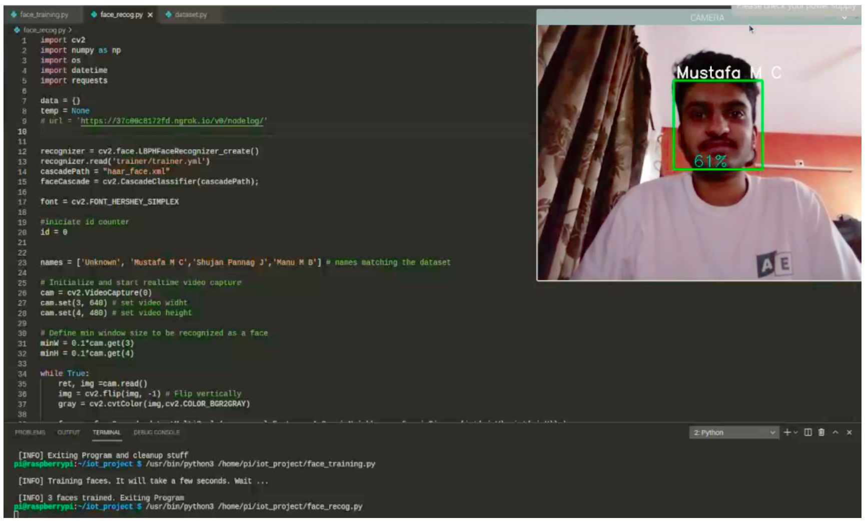


Figure 5.2: Node Setup.

To create the datasets, a python script is written which captures 60 images and stores them in a folder. The model is trained on datasets to recognize the person. Finally, by running face_recog.py, faces are recognized. The recognized person’s name, date, time, and IP address of the node are uploaded to the cloud using a REST API.



```python
import cv2
import numpy as np
import os
import datetime
import requests

data = {}
temp = None
# url = 'https://37c00c8172fd.ngrok.io/v0/nodelog/'


recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haar_face.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#iniciate id counter
id = 0


names = ['Unknown', 'Mustafa M C','Shujan Pannag J','Manu M B'] # names matching the dataset

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:
    ret, img =cam.read()
    img = cv2.flip(img, -1) # Flip vertically
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

```

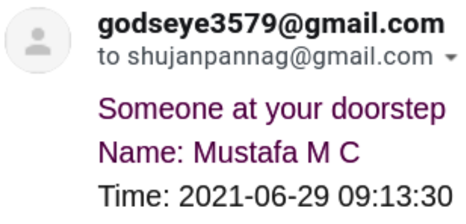
```
 [INFO] Exiting Program and cleanup stuff
pi@raspberrypi:~/iot_project $ /usr/bin/python3 /home/pi/iot_project/face_training.py

 [INFO] Training faces. It will take a few seconds. Wait ...

 [INFO] 3 faces trained. Exiting Program
pi@raspberrypi:~/iot_project $ /usr/bin/python3 /home/pi/iot_project/face_recog.py
```

Figure 5.3: Face Recognition.

The REST API server is implemented using the Go programming language and Gin web framework. Once the data is uploaded, the server unmarshals the HTTP body and retrieves the data. This data is then inserted into the Postgres database using the Go Object Relational Mapping (GORM). For every insertion to the database i.e., when a person is at the doorstep, an email is sent through SMTP client implemented using net/smtp package.



godseye3579@gmail.com
to shujanpannag@gmail.com

Someone at your doorstep
Name: Mustafa M C
Time: 2021-06-29 09:13:30

Figure 5.4: Email alert sent when someone is at the doorstep.

Sever logs and Database logs can be downloaded using a HTTP GET verb through API.

5.1 API Resource Policy

5.1.1 Node Log

Get Raspberry Pi Node Log of required <date>

Method	**GET** /v0/nodelog/:date
Request Body	curl -i -H 'Content-Type: application/json' http://localhost:8080/v0/nodelog/2000-06-28

Upload Raspberry Pi log data to the cloud

Method	**POST** `/v0/nodelog/`
Request Body	`curl -i -H 'Content-Type: application/json' -d '{"relname": " " , "ipaddr": " ", "datetime": " "}' http://localhost:8080/v0/nodelog/`

5.1.2 User Log

Get registered user information of required <rel>

Method	**GET** `/v0/userrel/:rel`
Request Body	`curl -i -H 'Content-Type: application/json' http://localhost:8080/v0/nodelog/name`

Register new user data on the server

Method	**POST** `/v0/userrel/`
Request Body	`curl -i -H 'Content-Type: application/json' -d '{"name": " ", "rel": " ", "email": " "}' http://localhost:8080/v0/userrel`

Update registered user information of <name>

Method	**PUT** `/v0/userrel/:name`
Request Body	`curl -i -H 'Content-Type: application/json' -d '{"name": " ", "rel": " ", "email": " "}' http://localhost:8080/v0/name`

Delete user from the database of <name>

Method	**DELETE** `/v0/userrel/:name`
Request Body	`curl -i -H 'Content-Type: application/json' http://localhost:8080/v0/name`

5.1.3 Server Log

Download Server and Database log

Method	**GET** `/v0/getlog`
Request Body	`curl http://localhost:8080/v0/getlog`

Chapter 6

FUTURE WORK

God's Eye is a prototype of the concept and not a production ready platform. Following are the features and improvements that can be added:

- Function to add new users and train the model on the node.
- Addition of a fool proof feature to differentiate between face and a photo.
- Enabling more features offered by the API on Node Side.
- A GUI or a web-based UI to monitor and perform analytics on log data.

Project Hosted on GitHub

- API SERVER: https://github.com/shujanpannag/iot_project_api
- NODE: https://github.com/mbmanu/IoT_Project

BIBLIOGRAPHY

[1] Raspberry Pi Foundation, https://www.raspberrypi.org/, July 2021.

[2] OpenCV team, https://opencv.org/, July 2021.

[3] Pratheek Joshi, “OpenCV with Python by Example”, Packt, 2015.

[4] Gin, https://pkg.go.dev/github.com/gin-gonic/gin, June 2021.

[5] GORM, https://pkg.go.dev/gorm.io/gorm, June 2021.

[6] Naren Yellavula, “Building RESTful Web services with Go”, Packt, 2017.

[7] PostgreSQL Global Development Group, https://www.postgresql.org/files/documentation/pdf/13/postgresql-13-A4.pdf, June 2021.