CHAPTER – 1

ABSTRACT

ABSTRACT

Home surveillance system is becoming one of the key factors of the home security systems. A fully remotely controlled surveillance of home is going to be the future. Everyone is worried about the security of the home when they are not at home, also they want to be sure that the children or old ones are safe.

This paper presents a novel solution that makes the surveillance of home from anywhere with the help of internet. This software is programmed in python language and raspberry pi is used as a hardware. The photo is clicked with the help of raspberry pi camera and is send to the authorized owner of the house.

This Raspberry Pi based Smart Surveillance System presents the idea of monitoring a particular place in a remote area. The proposed solution offers a cost effective ubiquitous surveillance solution, efficient and easy to implement.

This project will also present the idea of motion detection and tracking using image processing. This type of technology is of great importance when it comes to surveillance and security. Live video streams will therefore be used to show how objects can be detected then tracked. The detection and tracking process will be based on pixel threshold.

We can even add more than one authorized person to whom the message will be forwarded. The database is maintained with the help of google firebase. We can even extend the project by adding more modules for providing more security.

This surveillance system can be used from anywhere in the world either by using mobile application designed in Flutter or by accessing web portal.

We can automate our home through IOT, which means we can able to control our appliances, devices and electrical switches from anywhere. Also all the detected images can be accessed through web portal or mobile application. These images are stored in Firebase Storage which is cloud storage.

Mobile application is built in flutter, which allows to develop applications for Android and iOS both with single codebase.

CHAPTER – 2

INTRODUCTION

2. Introduction

In the world of Internet of Things, when we have all the technologies to revolutionize our life, it's a great idea to develop a system which can be controlled and monitored from anywhere. There are many types of good security systems and cameras out there for home security but they are much expensive so today we will build a low cost simple **Raspberry Pi based Intruder Alert System**, which not only alert you through an email but also sends the picture of Intruder when it detects any.

In this IOT based paper, we will build a **Home Security System using Raspberry PI Camera**. This system will detect the presence of Intruder and quickly alert the user by sending him a alert mail. This mail will also contain the Picture of the Intruder, captured by Pi camera. Raspberry Pi is used to control the whole system. This system can be installed at the main door of your home or office and you can monitor it from anywhere in the world using your Email over internet.

With the increasing penetration of the digital world into smart objects, the basic and conventional methods of various home appliances like refrigerator, Air Conditioner, etc. are taking a whole new, revamped alteration. Many a times, a problem arises that parents are not at home with their kids and a visitor arrives. That person may be the courier guy or the home delivery boy or a person from your friend. This is where the system can come into use.

The problem with conventional systems is that it is either always on or it gives the information only when the bell is pressed. But with the system presented here the owner or the parent will get the info even when the person is close by to the gate or the bell. This gives additional security features to the security system and gives the residents more time to prepared if there is a threat.

Though IOT has abundant benefits, there are flaws in the IOT governance and implementation level. The key observations are that, there is no standard definition worldwide also universal standardization is required at the architectural level and technology changes from vendor to vendor and hence requires interoperability.

Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

The problem of computer vision appears simple because it is trivially solved by people, even very young children. Nevertheless, it largely remains an unsolved problem based both on the limited understanding of biological vision and because of the complexity of vision perception in a dynamic and nearly infinitely varying physical world.

Using software to parse the world’s visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.
Computer Vision is the process of using machines to understand and analyse imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960’s, recent advances in Machine Learning, as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

- In **object classification**, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
- For **object identification**, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.

A classical application of computer vision is handwriting recognition for digitizing handwritten content (we'll explore more use cases below). Outside of just recognition, other methods of analysis include:

- Video **motion analysis** uses computer vision to estimate the velocity of objects in a video, or the camera itself.
- In **image segmentation**, algorithms partition images into multiple sets of views.
- **Scene reconstruction** creates a 3D model of a scene inputted through images or video.
- In **image restoration**, noise such as blurring is removed from photos using Machine Learning based filters.

Any other application that involves understanding pixels through software can safely be labelled as computer vision.

Case Study

Day by day the work and life of human being's are increasingly busy and complicated with the rapid growth in communications and information technology. As the economic expansion is growing rapidly, the standard of living also keeps on rising up and the people are requiring more living functions. The concept of smart home has focused the attention of re-searchers. A lot of efforts have been done for the development of home automation to control that remotely. Smart Home is an advanced technology to make a house to become intelligent and automated. Usually, that technology has automation systems for lighting, temperature control, security and many other functions. Here in this paper a comparative study has done on different types of home automation system. We have gone through different techniques for the implementation of smart home such as; phone-based remote controller for home and office automation, PC remote control of appliances by using telephone lines, Blue-tooth wireless technology based home automation, internet based wireless home automation system, remote home automation monitoring using mobile through spoken commands, GSM-based remote sensing and control system using FPGA, GSM-Bluetooth based remote monitoring and control system with automatic light controller.

CHAPTER – 3

EXISTING SECURITY TECHNOLOGY

3. Existing security technology

Closed-circuit television (CCTV) Security System

Video surveillance is the use of video cameras to transmit a signal to a specific place, on a limited set of monitors. It differs from broadcast television in that the signal is not openly transmitted, though it may employ point to point (P2P), point to multipoint, or mesh wireless links. In the U.S. the first commercial closed-circuit television system became available in 1949, called Vericon.

Fig. CCTV Camera

Operation of a CCTV Security System

The simplest system is a camera connected directly to a monitor by a coaxial cable with the power for the camera being provided from the monitor. The outdoor or indoor camera take several images per second and thus cannot be differentiated by human eye. The images are then transferred via a coaxial cable or optic fiber to a computer placed in a secure location. This computers are monitored by security personnel and responds to any improper behaviors'. These systems have been incorporated with alarm systems so as to send out an alert in case of a security bridge.

Two types of CCTV storage exist; VCR and DVR. The DVR system is more superior as it can be able to transmit digitized video signals over the data networks and thus can allow for remote control and monitoring of the system.

Remote Surveillance IP System

IP surveillance is a digitized and networked version of closed-circuit television (CCTV). In an IP surveillance system, an IP camera records video footage and the resulting content is distributed over an IP (Internet protocol) network. Adding networking capability to digital CCTV provides additional benefits, including:

- Improved ability for remote viewing and control. Anyone on the network can potentially see video from any camera connected to the network.
- IP storage makes it possible to store data in any geographic location.
- Greater ease of distribution. An image of a crime suspect, for example, can be immediately distributed to officials.
- The ability to connect to email and other communications systems so that alerts can be sent automatically.

Disadvantage of existing technology

1. The view of the CCTV is limited:

This means that once these are installed, they are capable of keeping track of a certain angle and certain area of that particular area. The area is certainly very limited. If the vandals plan to disrupt the condition of a particular place, they might just change the direction of the camera or simply can put something like a chewing gum on the camera to hide the view so that they can continue disrupting the normal state of the place.

2. The wireless technology:

The technology of this wireless CCTV is such that they can subject to distortion due to the other powerful devices that are installed alongside. The camera might not send proper signal to the output devices which might lead to the security people fail to understand the story behind any kind of vandalism.

3. The wired technology:

The disadvantage of using a wired CCTV camera is that it cannot be ported to any other place but has to be installed in a particular part of an area and has to be left it that way for the rest of the time. This causes limitations to the view as well.

4. Hackers:

There are hackers all over the world and CCTV cameras are very much keen on becoming the victim of such misdeeds which again leads to a corrupted security system. There are people who can easily get information that are preloaded in these CCTV software by just hacking the entire device thus getting a lead on the footage that might be very important in terms of security.

5. Haunts the privacy at times:

There are certain places where people need to have some privacy, be it the changing rooms or the toilets. The CCTV cameras are certainly not installed in these Places but still people get uncomfortable because of the corruption which always is a part under such circumstances.

6. The monitoring system:

There might be problems related to the monitoring system as well. The CCTV cameras are not always capable of sending the right feedback which again can lead to the monitoring system to misjudge the entire scenario. There are situation when simply hacking the monitoring area, the vandals can get information of whatever they need to know, thus corrupting the security system altogether.

7. The quality of the material:

These CCTV cameras are certainly made of some very poor quality materials so are very much keen on getting destroyed, if the vandals want to destroy these.
In spite of these disadvantages, installing CCTV cameras has become one of the most important ways of maintaining the security of any particular public place.

- Wireless Technology
- Limited Camera View
- Easy to Hack
- No Real Time Alert
- No proper assurance of Complete Security

CHAPTER – 4

IMPLEMENTING TECHNOLOGIES

4. Implementing technology

The Raspberry Pi

The Raspberry Pi is a Linux based microcomputer based on ARM architecture. It was built mainly to aid in developing open source game. The device is estimated to cost about $35 depending on the model.

The Raspberry Pi Models

This part describes the models of Raspberry Pi available. This report will not attempt to provide full specifications but an overview in order to help in making a decision as to which device it is required to accomplish the objectives in question. Currently, five Raspberry Pi models do exist. They are: Model B+, Model A+, Model B, Model A and the Compute Module (currently only available as part of the Compute Module development kit). All these models use the same SoC (System on Chip - combined CPU & GPU), the BCM2835, but other hardware features differ.

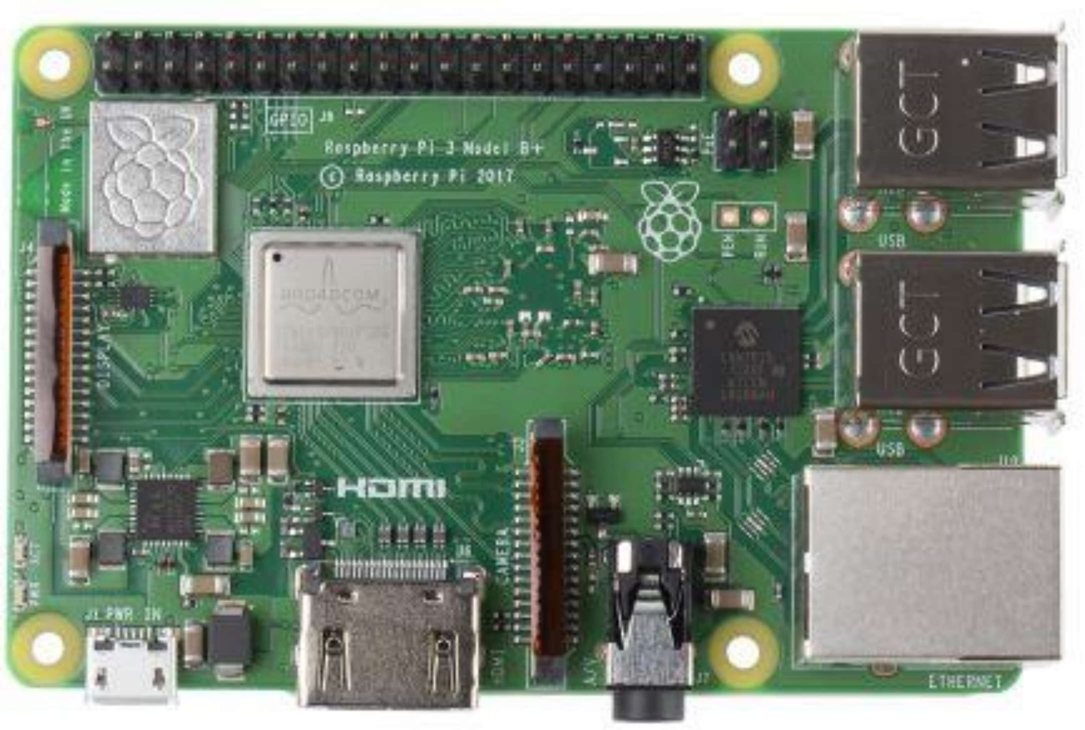


Fig. Raspberry Pi

Model B+/B

First release was made in July 2014. This Model is an upgrade of the Model B. It has the following characteristics: 4 USB ports, 40 pins on the GPIO header , Improved power circuitry which allows higher powered USB devices to be attached and now hot-plugged. The full size composite video connector of Model B has been removed and the functionality moved to the 3.5mm audio/video jack and the full size SD card slot of Model B has also been replaced with a much more robust microSD slot. The following details some of the improvements over the Model B:

- Current monitors on the USB ports mean the B+ now supports hot-plugging
- Current limiter on the 5V for HDMI means HDMI cable-powered VGA converters work in all cases.
- 14 more GPIO pins
- EEPROM readout support for the new HAT expansion boards
- Higher drive capacity for analog audio out, from a separate regulator, which means a better audio DAC quality
- No more back powering problems, due to the USB current limiters which also inhibit back flow, together with the "ideal power diode"
- Composite output moved to 3.5mm jack
- Connectors now moved to two sides of the board rather than the four of the original device
- Ethernet LEDs moved to the Ethernet connector
- 4 squarely-positioned mounting holes for more rigid attachment to cases.

Model A/A+

This is the basic device, with a single USB port and 256MB of SDRAM. Onboard ports include: Full size SD card, HDMI output port, Composite video output, One USB port, 26 pin expansion header exposing GPIO, 3.5mm audio jack, Camera interface port (CSI-2), LCD display interface port (DSI) and One microUSB power connector for powering the device.

Programming the Raspberry Pi

To enable communication with the outside world, the Raspberry Pi has to be programmed with a suitable programming language. These languages include Java, FOTRAN, Pascal, Python, C, C++ etc. Each language has its own syntax and semantics. RPI can be programmed using any of these languages but for purposes of this project, Python will be of great importance to study. It is provided by default through and thus optimum operation of the Pi can be achieved.

Python is a wonderful and powerful programming language that's easy to use (easy to read and write) and with Raspberry Pi lets you connect your project to the real world.

Python syntax is very clean, with an emphasis on readability and uses standard English keywords. Start by opening IDLE from the desktop.

IDLE

The easiest introduction to Python is through IDLE, a Python development environment. Open IDLE from the Desktop or applications menu:

Fig. Python Logo

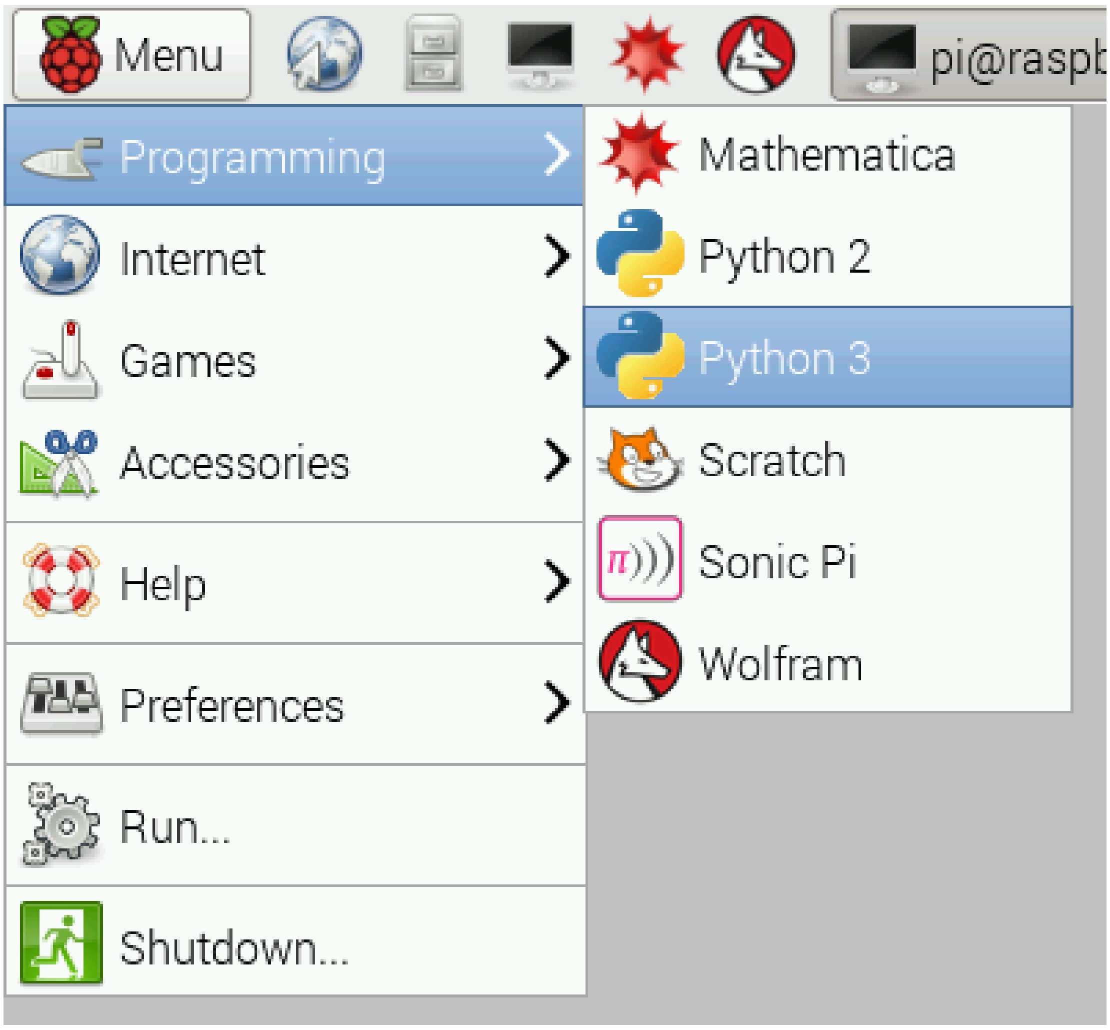


Fig. Raspberry Pi Menu

IDLE gives you a REPL (Read-Evaluate-Print-Loop), which is a prompt you can enter Python commands into. Because it's a REPL, you even get the output of commands printed to the screen without using print.

Note: two versions of Python are available — Python 2 and Python 3. Python 3 has was first released in 2008 and Python 2 development ended with 2.7, which was released in 2010. Python 3 is recommended, but Python 2 is available for legacy applications which do not support Python 3 yet.

You can use variables if you need to but you can even use it like a calculator. For example:

```
>>> 1 + 2
```

```
3
>>> name = "Sarah"
>>> "Hello " + name
'Hello Sarah'
```

IDLE also has syntax highlighting built in and some support for autocompletion. You can look back on the history of the commands you've entered in the REPL with Alt + P (previous) and Alt + N (next).

Basic Python usage

Hello world in Python:

```
print("Hello world")
```

Simple as that!

Indentation

Some languages use curly braces { and } to wrap around lines of code which belong together, and leave it to the writer to indent these lines to appear visually nested. However, Python does not use curly braces but instead requires indentation for nesting. For example a for loop in Python:

```
for i in range(10):
    print("Hello")
```

The indentation is necessary here. A second line indented would be a part of the loop, and a second line not indented would be outside of the loop.

For example:

```
for i in range(2):
    print("A")
    print("B")
```

would print:

```
A
B
```

```
A
B
```

whereas the following:

```
for i in range(2):
    print("A")
print("B")
```

would print:

```
A
A
B
```

Variables

To save a value to a variable, assign it like so:

```
name = "Bob"
age = 15
```

Note that data types were not specified with these variables, as types are inferred, and can be changed later.

```
age = 15
age += 1  # increment age by 1
print(age)
```

This time comments are used beside the increment command.

Comments

Comments are ignored in the program but there for you to leave notes, and are denoted by the hash # symbol. Multi-line comments use triple quotes like so:

```
"""
This is a very simple Python program that prints
"Hello".
That's all it does.
```

```
"""
print("Hello")
```

Lists

Python also has lists (called arrays in some languages) which are collections of data of any type:

```
numbers = [1, 2, 3]
```

Lists are denoted by the use of square brackets [] and each item is separated by a comma.

Iteration

Some data types are iterable, which means you can loop over the values they contain. For example a list:

```
numbers = [1, 2, 3]

for number in numbers:
    print(number)
```

This takes each item in the list numbers and prints out the item:

```
1
2
3
```

Note the word number to denote each item. This is merely the word I chose for this - it's recommended you choose descriptive words for variables - using plurals for lists, and singular for each item makes sense. It makes it easier to understand when reading.

Other data types are iterable, for example the string:

```
dog_name = "BINGO"

for char in dog_name:
    print(char)
```

This loops over each character and prints them out:

```
B
I
N
G
O
```

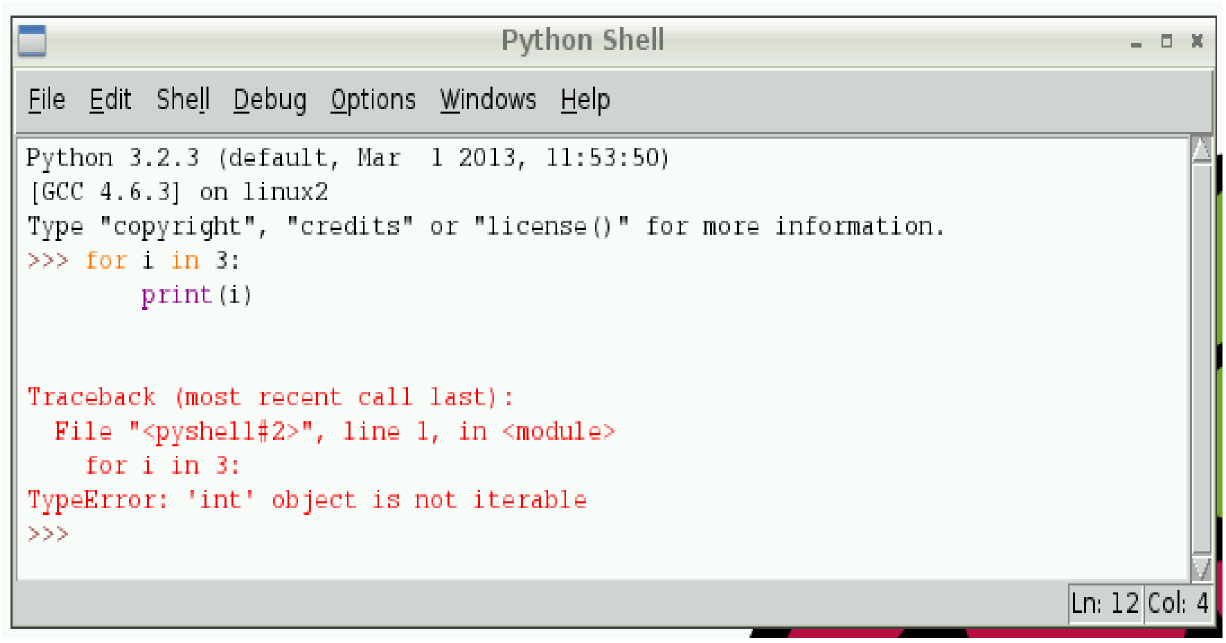
Range

The integer data type is not iterable and trying to iterate over it will produce an error. For example:

```
for i in 3:
    print(i)
```

will produce:

```
TypeError: 'int' object is not iterable
```



```
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> for i in 3:
        print(i)


Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    for i in 3:
TypeError: 'int' object is not iterable
>>>
```

Fig. Python Shell

However you can make an iterable object using the range function:

```
for i in range(3):
    print(i)
```

`range(5)` contains the numbers 0, 1, 2, 3 and 4 (five numbers in total). To get the numbers 1 to 5 (inclusive) use range(1, 6).

Length

You can use functions like len to find the length of a string or a list:

```
name = "Jamie"
print(len(name))  # 5

names = ["Bob", "Jane", "James", "Alice"]
print(len(names))  # 4
```

If statements

You can use if statements for control flow:

```
name = "Joe"

if len(name) > 3:
    print("Nice name,")
    print(name)
else:
    print("That's a short name,")
    print(name)
```

Python files in IDLE

To create a Python file in IDLE, click File > New File and you'll be given a blank window. This is an empty file, not a Python prompt. You write a Python file in this window, save it, then run it and you'll see the output in the other window.

For example, in the new window, type:

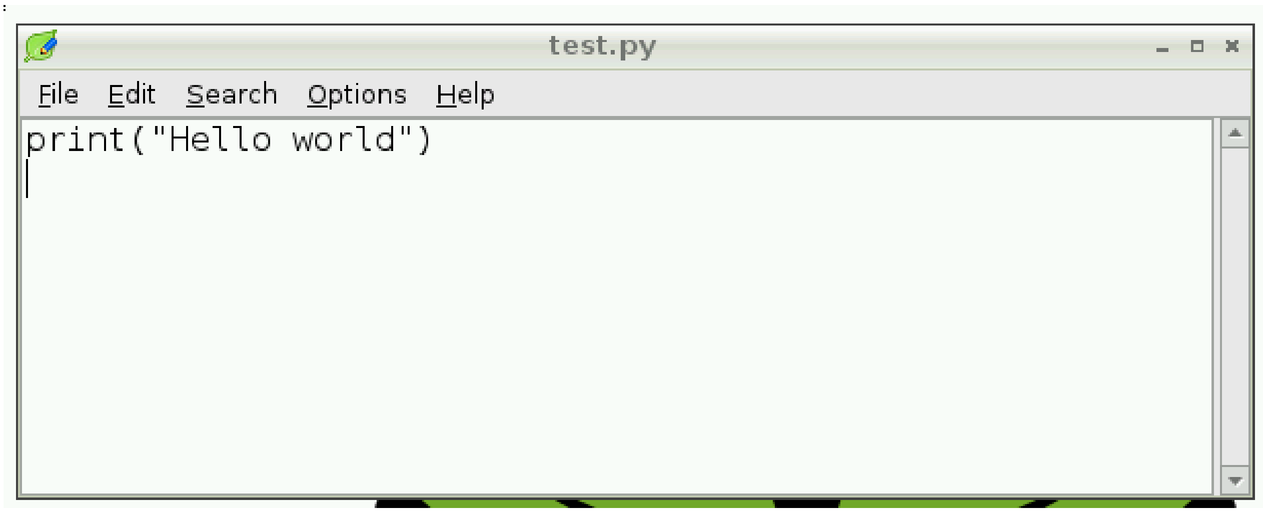
```
n = 0
for i in range(1, 101):
    n += i
```

```
print("The sum of the numbers 1 to 100 is:")
print(n)
```

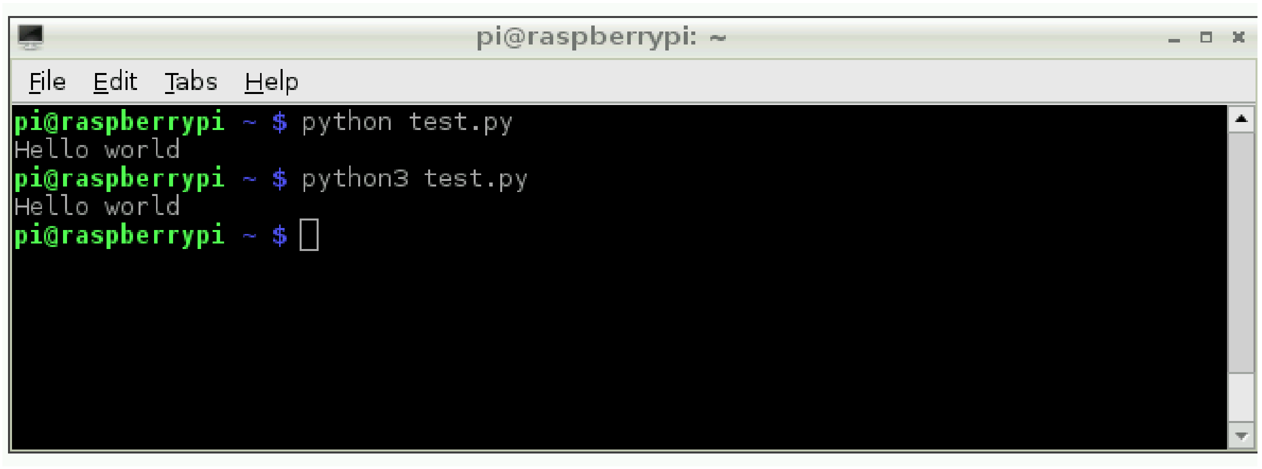
Then save this file (File > Save or Ctrl + S) and run (Run > Run Module or hit F5) and you'll see the output in your original Python window.

Executing Python files from the command line

You can write a Python file in a standard editor like Vim, Nano, or LeafPad, and run it as a Python script from the command line. Just navigate to the directory the file is saved in (use cd and ls for guidance) and run with python3, e.g. python3 hello.py.



```
print("Hello world")
```



```
pi@raspberrypi ~ $ python test.py
Hello world
pi@raspberrypi ~ $ python3 test.py
Hello world
pi@raspberrypi ~ $ 
```

Fig. Hello World in Python

Other ways of using Python

<u>Command Line</u>

The standard built-in Python shell is accessed by typing python3 in the terminal. This shell is a prompt ready for Python commands to be entered. You can use this in the same way as IDLE, but it does not have syntax highlighting or autocompletion. You can look back on the history of the commands you've entered in the REPL by using the Up/Down keys. Use Ctrl + D to exit.

IPython

IPython is an interactive Python shell with syntax highlighting, autocompletion, pretty printing, built-in documentation, and more. IPython is not installed by default. Install with:

sudo pip3 install ipython

Then run with ipython from the command line. It works like the standard python3, but has more features. Try typing len? and hitting Enter. You're shown information including the docstring for the len function:

```
Type:        builtin_function_or_method
String Form:<built-in function len>
Namespace:  Python builtin
Docstring:
len(object) -> integer

Return the number of items of a sequence or mapping.
```

Try the following dictionary comprehension:

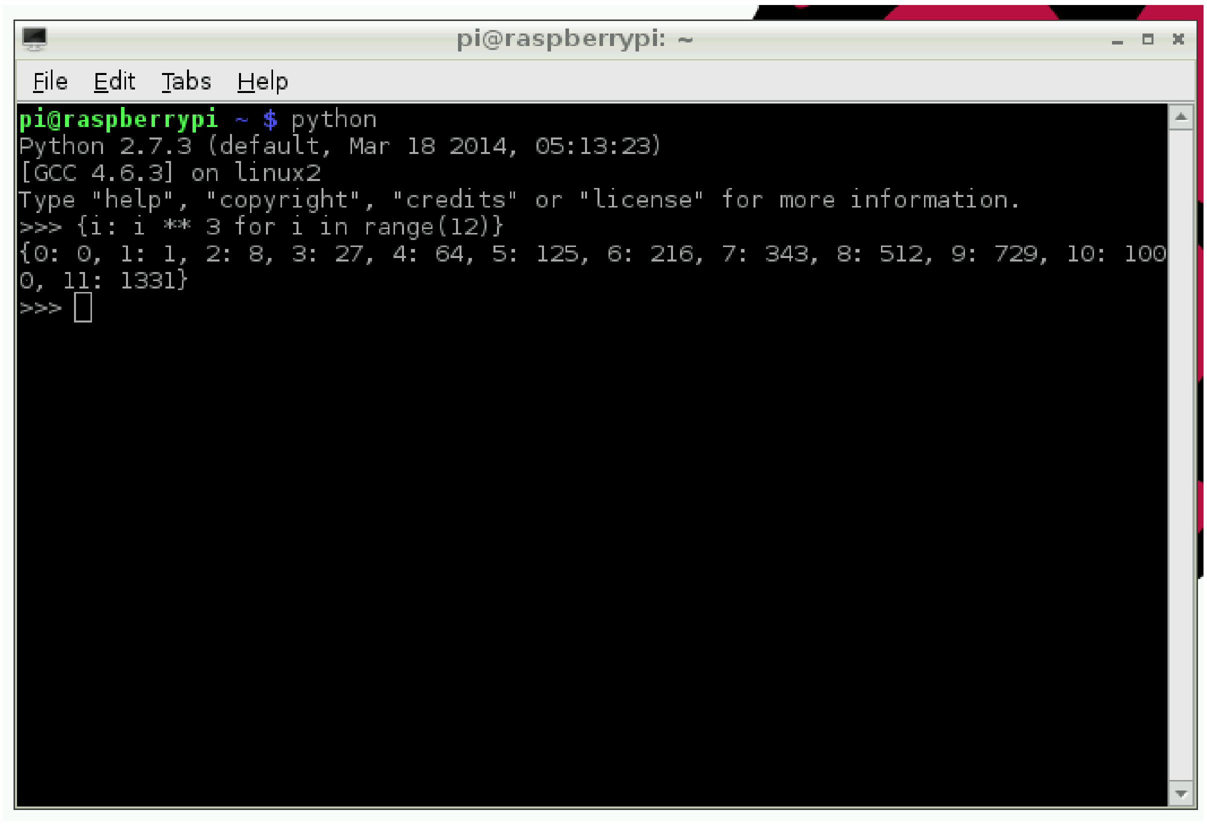
```
{i: i ** 3 for i in range(12)}
```

This will pretty print the following:

```
{0: 0,
 1: 1,
 2: 8,
 3: 27,
 4: 64,
 5: 125,
 6: 216,
 7: 343,
 8: 512,
 9: 729,
 10: 1000,
 11: 1331}
```

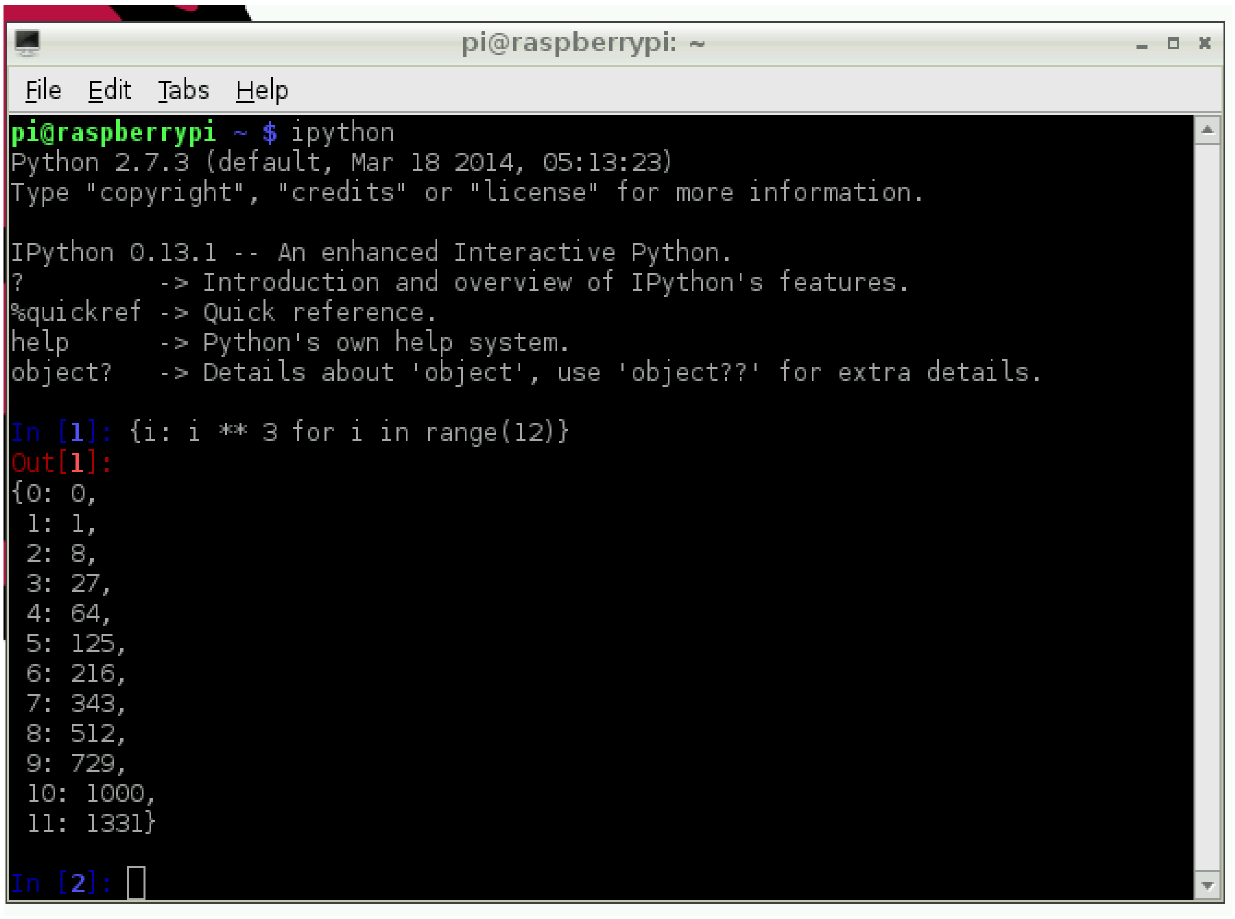
In the standard Python shell, this would have printed on one line:

```
{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8:
512, 9: 729, 10: 1000, 11: 1331}
```



```
pi@raspberrypi ~ $ python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> {i: i ** 3 for i in range(12)}
{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729, 10: 100
0, 11: 1331}
>>> 
```

Fig. Python in Raspberry Pi



```
pi@raspberrypi ~ $ ipython
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: {i: i ** 3 for i in range(12)}
Out[1]:
{0: 0,
 1: 1,
 2: 8,
 3: 27,
 4: 64,
 5: 125,
 6: 216,
 7: 343,
 8: 512,
 9: 729,
 10: 1000,
 11: 1331}

In [2]:
```

Fig. IPython in Raspberry Pi

You can look back on the history of the commands you've entered in the REPL by using the Up/Down keys like in python. The history also persists to the next session, so you can exit ipython and return (or switch between v2/3) and the history remains. Use Ctrl + D to exit.

Installing Python libraries

apt

Some Python packages can be found in the Raspbian archives, and can be installed using apt, for example:

```
sudo apt update
sudo apt install python-picamera
```

This is a preferable method of installing, as it means that the modules you install can be kept up to date easily with the usual sudo apt update and sudo apt upgrade commands.

pip

Not all Python packages are available in the Raspbian archives, and those that are can sometimes be out of date. If you can't find a suitable version in the Raspbian archives, you can install packages from the Python Package Index (known as PyPI).

To do so, install pip:

```
sudo apt install python3-pip
Then install Python packages (e.g. simplejson)
with pip3:
sudo pip3 install simplejson
```

piwheels

The official Python Package Index (PyPI) hosts files uploaded by package maintainers. Some packages require compilation (compiling C/C++ or similar code) in order to install them, which can be a time-consuming task, particlarly on the single-core Raspberry Pi 1 or Pi Zero.
piwheels is a service providing pre-compiled packages (called *Python wheels*) ready for use on the Raspberry Pi. Raspbian Stretch is pre-configired to use piwheels for pip. Read more about the piwheels project at www.piwheels.org.

Raspberry Pi Operating Systems

An operating system makes Raspberry Pi run. Since Raspberry Pi is a credit sized computer that is based on Linux, optimum performance of RPI can be achieved if it is therefore operated in this environment. Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on RPI. Important to note is that the

Raspberry Pi does not operate in a Windows environment. To get access to Pi from windows we require Putty Software. Putty is an SSH and TelNet client.

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian was created by Mike Thompson and Peter Green as an independent project.[4] The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

Raspbian uses PIXEL, **P**i **I**mproved **X**-Window **E**nvironment, **L**ightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

The Pi Camera Module

The Camera Board on the Raspberry Pi is a small printed circuit board with a camera on it. The PCB is connected to a ribbon cable which connects to the Pi itself on its own port. The ribbon can be extendable. The camera being used in this project is Raspberry Pi camera module Version 1. This module is capable of capturing high definition video and still photographs. 1080p30, 720p60 and VGA90 video and still capture modes are available on this module. The connection from the module to the board is done by the CSI port. The camera is accessed through the MMAL and V4L APIs.

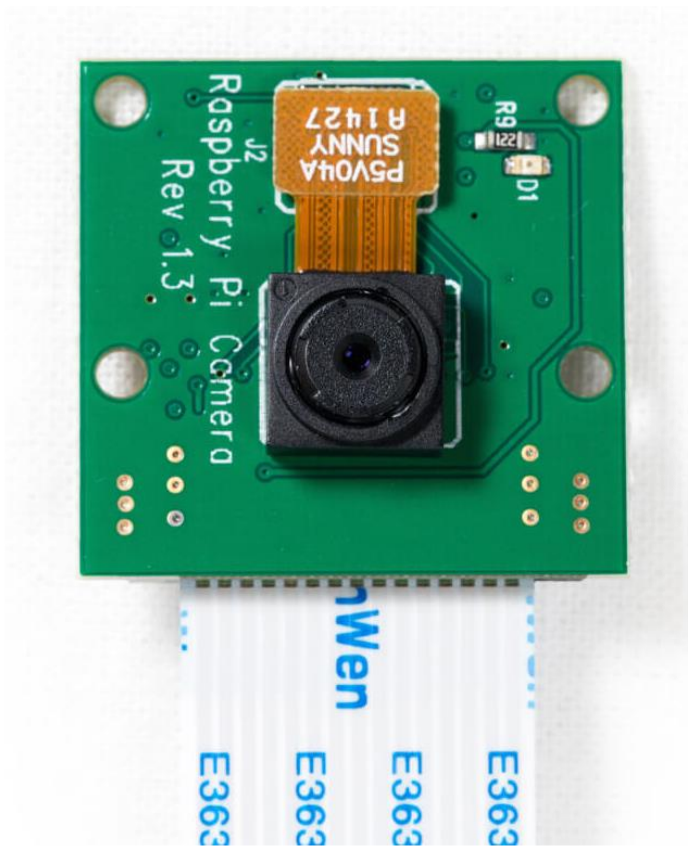


Fig. Pi Camera

GPIO, Interfacing and other Ports

Raspberry Pi has got 40 general purpose input/output pins also known as GPIO. They are the means by which a Raspberry Pi can interact with other devices. These pins can be programmed in ways to suit the device it is been connected to. Each pin has its own designated work. The details of their functions are shown in Fig. below.

There are 4 USB 2.0 ports for connecting USB devices. More than 4 USB devices can also be connected using USB HUB. An Ethernet port (RJ45 10/100 MBit/s) is fabricated on the board as well. It can be used to connect to the internet instead of using USB Wi-Fi dongle which results in faster internet speed. Other devices which have Ethernet connectivity can also be connected via this port.For video input, there is a 15-pin MIPI camera interface (CSI) connector through which a Raspberry Pi Camera can be connected. For video output, there are numerous ports. HDMI port is used to connect to HDMI supported monitors, TRRS jack for composite video output and MIPI display interface (DSI) for raw LCD panel displays. Raspberry Pi does not have any audio line-in option. However, for audio output it has an independent analog output via a 3.5mm phone jack. The HDMI port also serves as an audio output for cables supporting both audio and video. There is a micro-SD card slot which when loaded with a memory card acts as a storage medium for the Pi and also the operating system needed to run the Raspberry Pi is stored in that card.

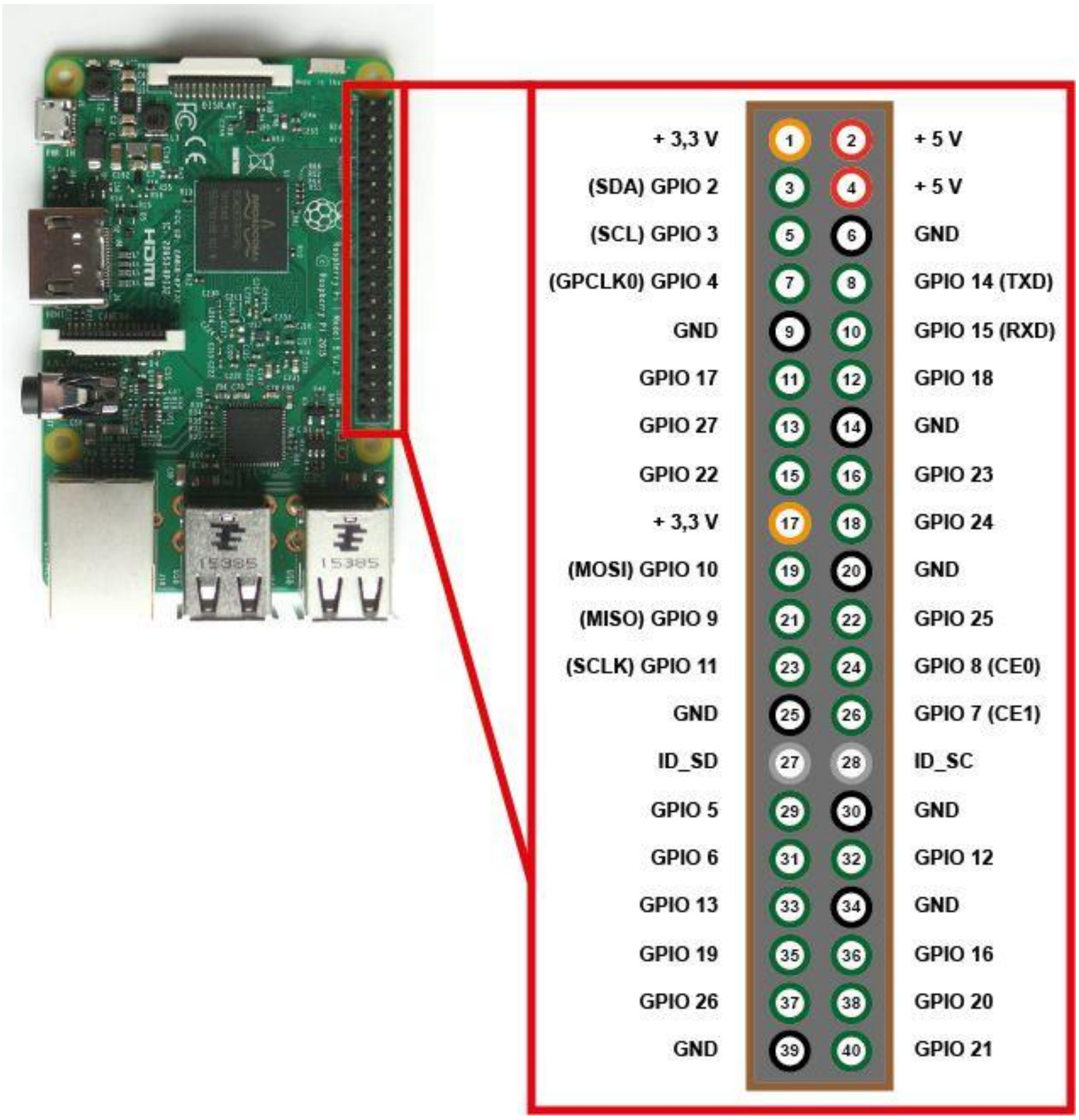


Fig. Pin Configuration of module

Power

Raspberry Pi needs 5V, and 700mA to 2A current to operate. This is supplied by a micro USB cable or GPIO power connector. The input voltage is divided into portions of 3.3V, 2.5V and 1.8V to operate the different components. The B model typically uses 700-1000mA depending on the peripherals in use. An individual GPIO pin can tolerate up to 16mA safely while they can take 50mA safely while distributed across all the 26 GPIO pins on the board. The HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take any amount in between 100mA or over 1000mA. The dimensions of this device are 85.60 mm × 56.5 mm and weigh about 45g.

CHAPTER – 5

IMAGE PROCESSING

5. Image Processing

Image Processing is a technique to enhance raw images received from cameras/sensors placed on satellites, space probes and aircrafts or pictures taken in normal day-to-day life for various applications. Most importantly, this technology is used in surveillance.

Grayscalling

Grayscale is a range of shades of gray without apparent color. The darkest possible shade is black, which is the total absence of transmitted or reflected light. The lightest possible shade is white, the total transmission or reflection of light at all visible wavelengths. Intermediate shades of gray are represented by equal brightness levels of the three primary colors (red, green and blue) for transmitted light, or equal amounts of the three primary pigments (cyan, magenta and yellow) for reflected light.

In the case of transmitted light (for example, the image on a computer display), the brightness levels of the red (R), green (G) and blue (B) components are each represented as a number from decimal 0 to 255, or binary 00000000 to 11111111. For every pixel in a red-green-blue (RGB) grayscale image, R = G = B. The lightness of the gray is directly proportional to the number representing the brightness levels of the primary colors. Black is represented by R = G = B = 0 or R = G = B = 00000000, and white is represented by R = G = B = 255 or R = G = B = 11111111. Because there are 8 bits in the binary representation of the gray level, this imaging method is called 8-bit grayscale.

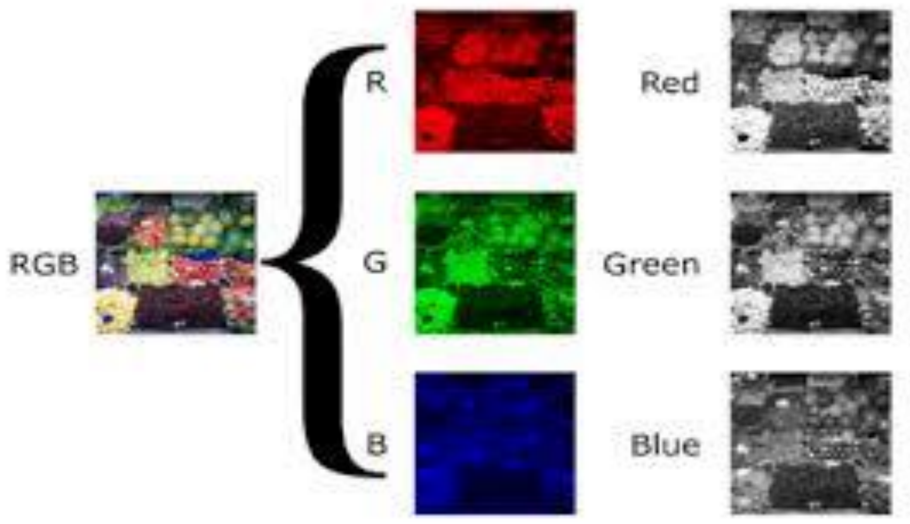


Fig. Original and gray images

Object Detection

Object detection is the process of finding instances of real-world objects such as faces, bicycles, and buildings in images or videos. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

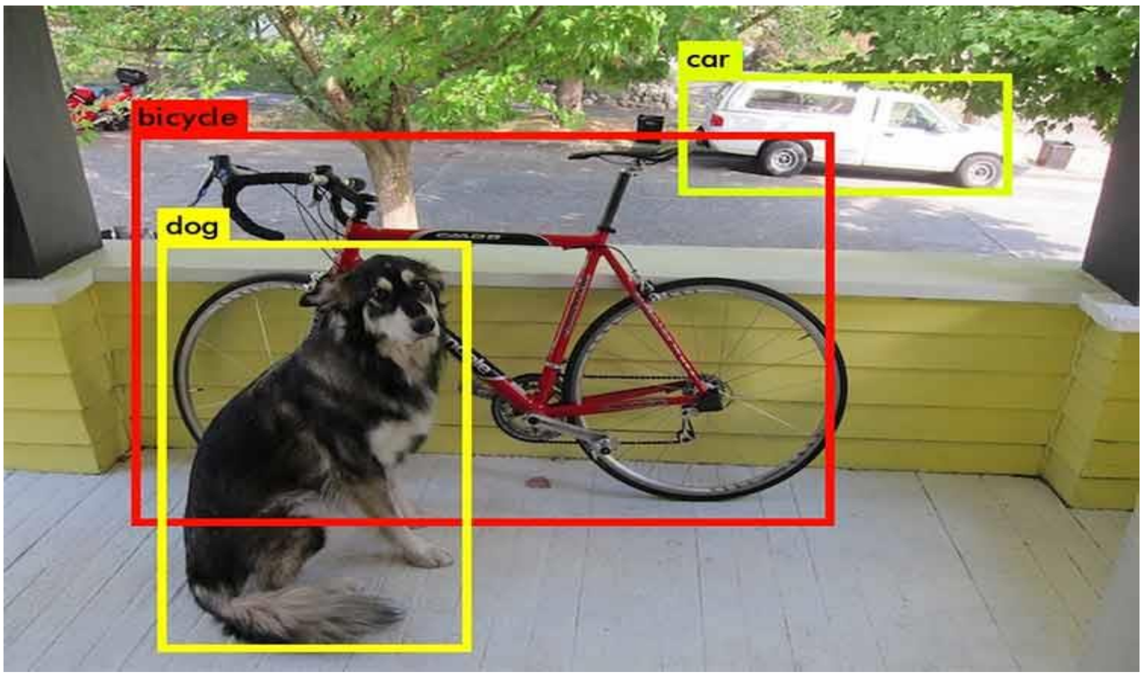
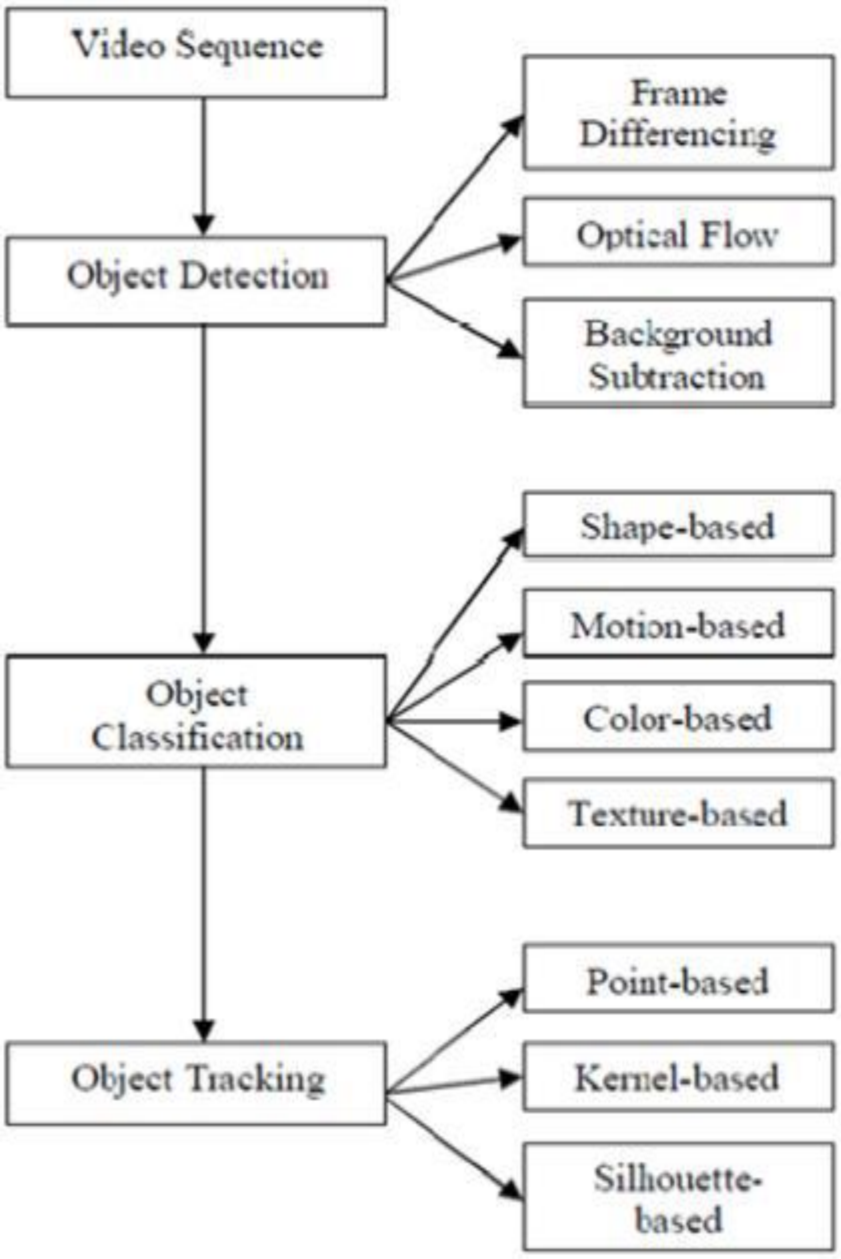


Fig. Object detection

The above can be achieved through the implementation of the algorithm below:



CHAPTER – 6

HOME AUTOMATION

6. Home Automation

A home automation system is a technological solution that enables automating the bulk of electronic, electrical and technology-based tasks within a home.

It uses a combination of hardware and software technologies that enable control and management over appliances and devices within a home.

Home automation is also known as domotics, and a home with an automation system is also known as a smart home.

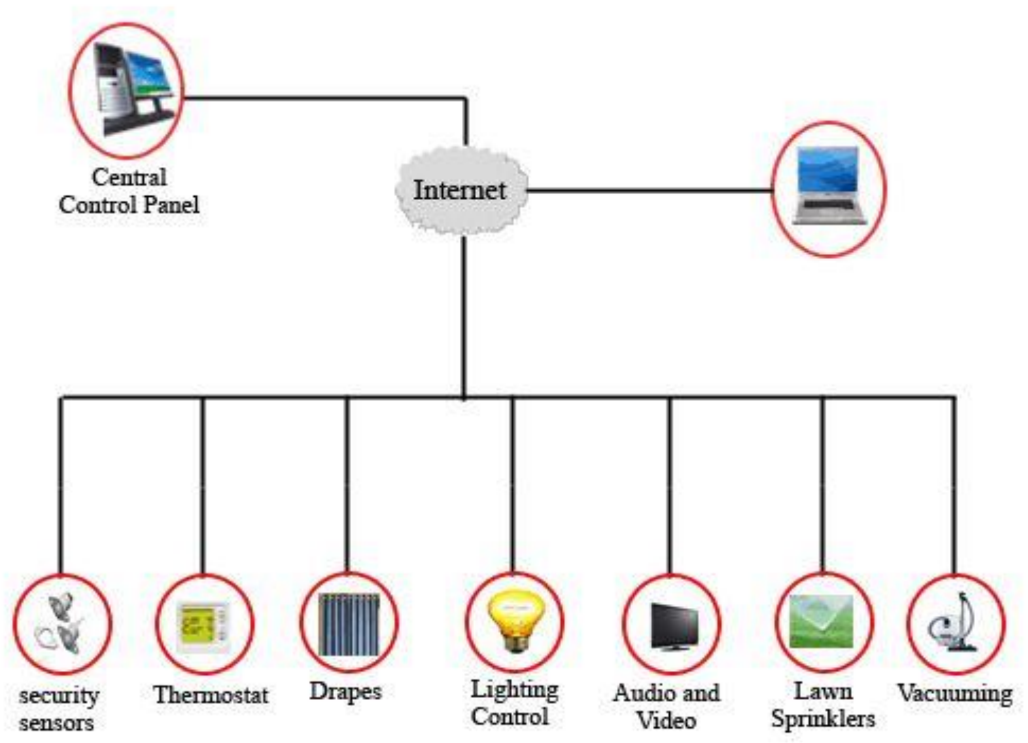


Fig. An overview of home automation system

Some of the processes within home automation system design, implementation and maintenance include:

- Installation of computer control for heating, ventilation and air conditioning systems
- Internet/remote/network access to all installed components and equipment
- Installation and maintenance of network-enabled surveillance cameras and physical security systems

- Central control and management capabilities over electrical fixtures and electronic appliances

Devices within the home automation system connect and communicate with each over a local wired or wireless network. The entire home automation system usually requires system management software, installation of device/appliance controllers, motion and temperature sensors and other components.

Automation

Automation is, unsurprisingly, one of the two main characteristics of home automation. Automation refers to the ability to program and schedule events for the devices on the network. The programming may include time-related commands, such as having your lights turn on or off at specific times each day. It can also include non-scheduled events, such as turning on all the lights in your home when your security system alarm is triggered. Once you start to understand the possibilities of home automation scheduling, you can come up with any number of useful and creative solutions to make your life better. Is that west-facing window letting in too much light? Plug your motorized blinds into a “smart” outlet and program it to close at noon each day. Do you have someone come by at the same time each day to walk the dog? Program your home automation system to unlock the front door for them, and lock it up again when they’re done.

Remote Control

The other main characteristic of cutting-edge home automation is remote monitoring and access. While a limited amount of one-way remote monitoring has been possible for some time, it’s only since the rise in smartphones and tablets that we’ve had the ability to truly connect to our home networks while we’re away. With the right home automation system, you can use any Internet-connected device to view and control the system itself and any attached devices.

Monitoring apps can provide a wealth of information about your home, from the status of the current moment to a detailed history of what has happened up to now. You can check your security system’s status, whether the lights are on, whether the doors are locked, what the current temperature of your home is and much more. With cameras as part of your home automation system, you can even pull up real-time video feeds and literally see what’s going on in your home while you’re away.

Even simple notifications can be used to perform many important tasks. You can program your system to send you a text message or email whenever your security system registers a potential problem, from severe weather alerts to motion detector warnings to fire alarms. You can also get notified for more mundane events, such as programming your "smart" front door lock to let you know when your child returns home from school.

The real hands-on control comes in when you start interacting with the home automation system from your remote app. In addition to arming and disarming your security system, you can reprogram the scheduling, lock and unlock doors, reset the thermostat and adjust the lights all from your phone, from anywhere in the world. As manufacturers are creating more and more "smart" devices and appliances all the time, the possibilities for home automation are virtually limitless.

Home Automation Components

What kinds of things can be part of a home automation system? Ideally, anything that can be connected to a network can be automated and controlled remotely. In the real world (outside of research labs and the homes of the rich and famous), home automation most commonly connects simple binary devices. This includes "on and off" devices such as lights, power outlets and electronic locks, but also devices such as security sensors which have only two states, open and closed.

Where home automation becomes truly "smart" is in the Internet-enabled devices that attach to this network and control it. The classic control unit is the home computer, for which many of the earlier home automation systems were designed. Today's home automation systems are more likely to distribute programming and monitoring control between a dedicated device in the home, like the control panel of a security system, and a user-friendly app interface that can be accessed via an Internet-enabled PC, smartphone or tablet.

Manufacturers have produced a wide variety of "smart" devices, many of which are full of innovative features but few of which offer the kind of integration needed to be part of a complete home automation system. Much of the problem has been that each manufacturer has a different idea of how these devices should be connected and controlled. So while you may have a "smart" TV, washing machine, refrigerator, thermostat, coffee maker or any of the other Internet-ready household devices on the market, the end result is usually a separate control scheme for each device.

In the near future, home automation may be standardized to let us truly take advantage of all of these additional possibilities. For the time being, the home security providers that specialize in home automation have focused on the most critical and useful parts of a connected home. At a basic level, this means the doors and windows and environmental devices (thermostat, smoke detectors, temperature, humidity, fire and carbon dioxide sensors) that keep you safe and comfortable. For additional real-time security, convenience and control, home automation systems from security providers should also include options for video cameras. With the best systems, you'll also be able to include lights and individual electrical outlets into your home automation package.

Energy Efficiency

One clear advantage of home automation is the unmatched potential for energy savings, and therefore cost savings. Your thermostat is already "smart" in the sense that it uses a temperature threshold to govern the home's heating and cooling system. In most cases, thermostats can also be programmed with different target temperatures in order to keep energy usage at a minimum during the hours when you're least likely to benefit from the heating and cooling.

At the most basic level, home automation extends that scheduled programmability to lighting, so that you can suit your energy usage to your usual daily schedule. With more flexible home automation systems, electrical outlets or even individual devices can also be automatically powered down during hours of the day when they're not needed. As with isolated devices like thermostats and sprinkler systems, the scheduling can be further broken down to distinguish between weekends and even seasons of the year, in some cases.
Set schedules are helpful, but many of us keep different hours from day to day. Energy costs can be even further reduced by programming "macros" into the system and controlling it remotely whenever needed. In other words, you could set up a "coming home" event that turns on lights and heating as you're driving home after work, for example, and activate it all with one tap on your smartphone. An opposite "leaving home" event could save you from wasting energy on forgotten lights and appliances once you've left for the day.

CHAPTER – 7

DATABASE

7. Database

Google Firebase

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products which are used by 1.5 million apps.

Firebase is a Backend-as-a-Service—BaaS—that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform.

Firebase frees developers to focus crafting fantastic user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you'll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can't be everything to everybody. But it gets pretty close.

Development Services

i. Firebase Cloud Messaging

Formerly known as Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost.

ii. Firebase Auth

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

iii. Realtime Database

Real-time data is the way of the future. Nothing compares to it. Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it.

When you connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a WebSocket. WebSockets are much, much faster than HTTP. You don’t have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client’s network can carry it.

Firebase provides a realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swiftand Node.js applications.

The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the realtime database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Realtime Database was released for beta use.

iv. Firebase Storage

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage

Firebase Storage provides a simple way to save binary files—most often images, but it could be anything—to Google Cloud Storage directly from the client.

Firebase Storage has it’s own system of security rules to protect your GCloud bucket from the masses, while granting detailed write privileges to your authenticated clients.

v. Firebase Hosting

Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layerencryption (SSL). Firebase partners with Fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

vi. ML Kit

ML Kit is a mobile machine learning system for developers launched on May 8, 2018 in beta during the Google I/O 2018. ML Kit API's feature a variety of features including text recognition, detecting faces, scanning barcodes, labelling images and recognising landmarks. It is currently available for iOS or Android developers. You may also import your own TensorFlow Lite models, if the given API's aren't enough. The API's can be used on-device or on cloud.

Crashlytics

Crash Reporting creates detailed reports of the errors in the app. Errors are grouped into clusters of similar stack traces and triaged by the severity of impact on app users. In addition to automatic reports, the developer can log custom events to help capture the steps leading up to a crash. Before acquiring Crashlytics, Firebase was using its own Firebase Crash Reporting.

Performance

Firebase Performance provides insights into an app's performance and the latencies the app's users experience.

CHAPTER – 8

HARDWARE

8. Hardware

Raspberry Pi Model 3 B+

First release was made in July 2014. This Model is an upgrade of the Model B. It has the following characteristics: 4 USB ports, 40 pins on the GPIO header , Improved power circuitry which allows higher powered USB devices to be attached and now hot-plugged. The full size composite video connector of Model B has been removed and the functionality moved to the 3.5mm audio/video jack and the full size SD card slot of Model B has also been replaced with a much more robust microSD slot. The following details some of the improvements over the Model B:

- Current monitors on the USB ports mean the B+ now supports hot-plugging
- Current limiter on the 5V for HDMI means HDMI cable-powered VGA converters work in all cases.
- 14 more GPIO pins
- EEPROM readout support for the new HAT expansion boards
- Higher drive capacity for analog audio out, from a separate regulator, which means a better audio DAC quality
- No more back powering problems, due to the USB current limiters which also inhibit back flow, together with the "ideal power diode"
- Composite output moved to 3.5mm jack

The Camera Board on the Raspberry Pi is a small printed circuit board with a camera on it. The PCB is connected to a ribbon cable which connects to the Pi itself on its own port. The ribbon can be extendable. The camera being used in this project is Raspberry Pi camera module Version 1. This module is capable of capturing high definition video and still photographs. 1080p30, 720p60 and VGA90 video and still capture modes are available on this module. The connection from the module to the board is done by the CSI port. The camera is accessed through the MMAL and V4L APIs.

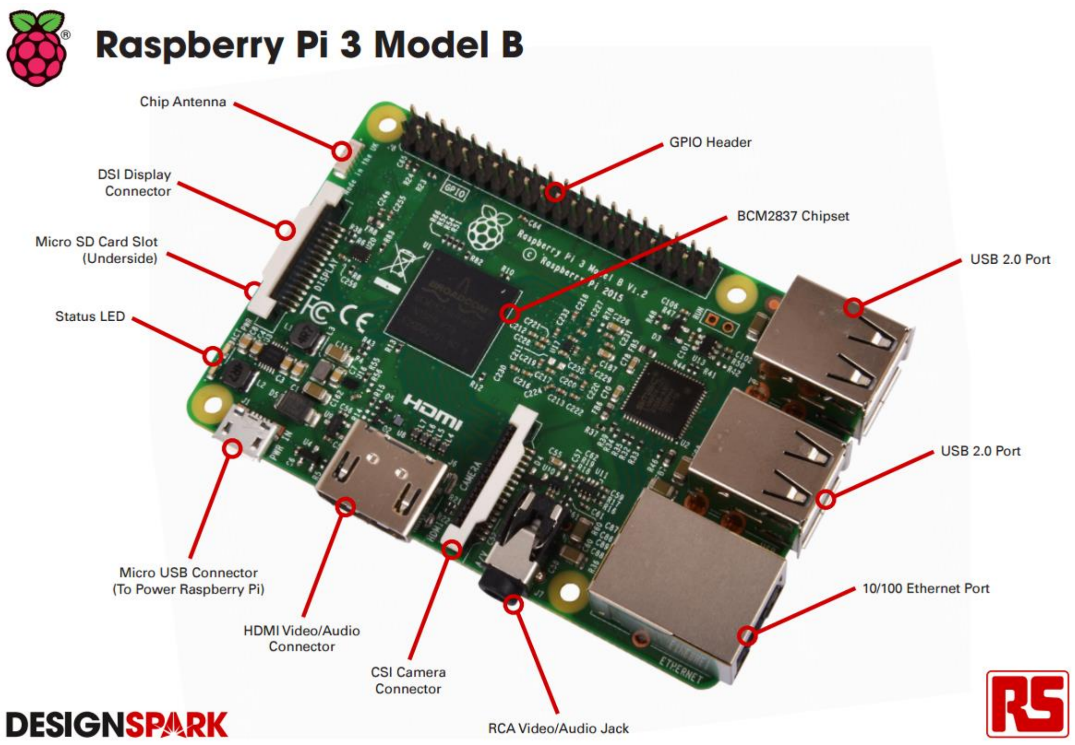


Fig. The Raspberry Pi

Pi Camera Module

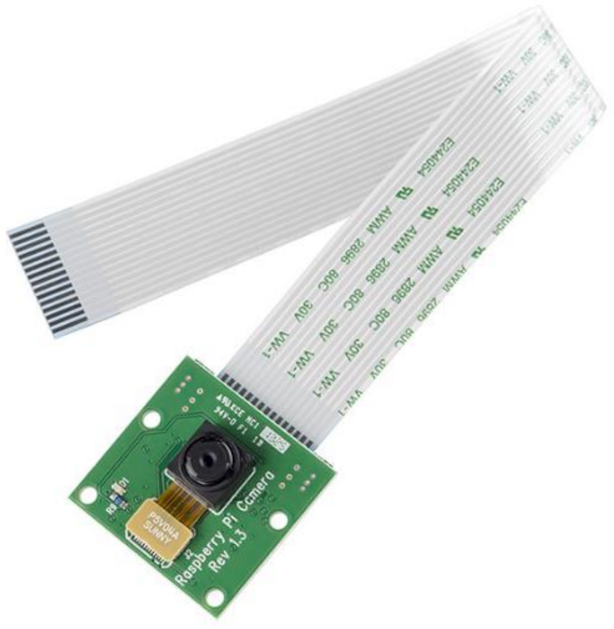


Fig. Pi Camera Module

CHAPTER – 9

SOFTWARE

9. Software

The software components of this project are an indispensable part for our complete idea to work. To develop our application, we have used Sublime Text, which is popular text editor for programming languages provides various plugins. For the connection between the Application and the Raspberry Pi we have used Python as our connection script language, Google Firebase to serve as our database and Python Server provided by Flask. The software configuration and characteristics are explained in detail in the coming sub-sections.

Python

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt.sources newsgroup in 1991, and version 1.0 was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do.

One of the amazing features of Python is the fact that it is actually one person's work. Usually, new programming languages are developed and published by large companies employing lots of professionals, and due to copyright rules, it is very hard to name any of the people involved in the project. Python is an exception.
There are not many languages whose authors are known by name.

Python was created by Guido van Rossum, born in 1956 in Haarlem, the Netherlands.

Python isn't a young language. It is mature and trustworthy. It's not a one-hit wonder. It's a bright star in the programming firmament, and time spent learning Python is a very good investment.

What makes Python so special? How does it happen that programmers, young and old, experienced and novice, want to use it? How did it happen that large companies adopted Python and implemented their flagship products using it?
There are many reasons – we've listed some of them already, but let's enumerate them again in a more practical manner:

- it's easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it's possible to start the actual programming faster;
- it's easy to teach – the teaching workload is smaller than that needed by other languages; this means that the teacher can put more emphasis on general (language-independent) programming techniques, not wasting energy on exotic tricks, strange exceptions and incomprehensible rules;
- it's easy to use for writing new software – it's often possible to write code faster when using Python;
- it's easy to understand – it's also often easier to understand someone else's code faster if it is written in Python;
- it's easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.

Flask (Python web Framework)

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. Flask is commonly used with MongoDB, which gives it more control over databases and history.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

Flask is a small and powerful web framework for Python. It's easy to learn and simple to use, enabling you to build your web app in a short amount of time.

Installing Flask

Before getting started, we need to install Flask. Because systems vary, things can sporadically go wrong during these steps. If they do, like we all do, just Google the error message or leave a comment describing the problem.

Install virtualenv

Virtualenv is a useful tool that creates isolated Python development environments where you can do all your development work.

Use virtualenv to install Flask. Virtualenv is a useful tool that creates isolated Python development environments where you can do all your development work. Suppose you come across a new Python library that you'd like to try. If you install it system-wide, there is the risk of messing up other libraries that you might have installed. Instead, use virtualenv to create a sandbox, where you can install and use the library without affecting the rest of your system. You can keep using this sandbox for ongoing development work, or you can simply delete it once you've finished using it. Either way, your system remains organized and clutter-free.

Generating and sending e-mail

After configuring the system to send an alert to the predefined subscriber, it was then necessary to generate and send the mail. Multipurpose Internet Mail Extension (MIME) package was then called and used to generate the attachment. MIME supports characters other than ASCII, non –text attachments (audio, video and application programs) etc. It thus extends the format of an email. Simple Mail Transfer Protocol (SMTP) program was then used to deliver the email from the Raspberry Pi to the configured mail hub. This can be summarized using the blocks below.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

```
class smtplib.SMTP([host[, port[, local_hostname[,timeout
]]]])
```

An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations. If the optional host and port parameters are given, the SMTP connect() method is called with those parameters during initialization. If specified, local_hostname is used as the FQDN of the local host in the HELO/EHLO command. Otherwise, the local hostname is found using socket.getfqdn(). If the connect() call returns anything other than a success code, an SMTPConnectError is raised. The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). If the timeout expires, socket.timeout is raised.

For normal use, you should only require the initialization/connect, sendmail(), and SMTP.quit() methods. An example is included below.

```
class smtplib.SMTP_SSL([host[, port[, local_hostname[, keyfile[, certfile[, timeout]]]]]])
```

An SMTP_SSL instance behaves exactly the same as instances of SMTP. SMTP_SSL should be used for situations where SSL is required from the beginning of the connection and using starttls() is not appropriate. If host is not specified, the local host is used. If port is omitted, the standard SMTP-over-SSL port (465) is used. local_hostname has the same meaning as it does for the SMTP class. keyfile and certfile are also optional, and can contain a PEM formatted private key and certificate chain file for the SSL connection. The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). If the timeout expires, socket.timeout is raised.

```
class smtplib.LMTP([host[, port[, local_hostname]]])
```

The LMTP protocol, which is very similar to ESMTP, is heavily based on the standard SMTP client. It’s common to use Unix sockets for LMTP, so our connect() method must support that as well as a regular host:port server. local_hostname has the same meaning as it does for the SMTP class. To specify a Unix socket, you must use an absolute path for host, starting with a ‘/’.

Authentication is supported, using the regular SMTP mechanism. When using a Unix socket, LMTP generally don’t support or require any authentication, but your mileage might vary.

A nice selection of exceptions is defined as well:

```
exception smtplib.SMTPException
```

The base exception class for all the other exceptions provided by this module.

```
exception smtplib.SMTPServerDisconnected
```

This exception is raised when the server unexpectedly disconnects, or when an attempt is made to use the SMTP instance before connecting it to a server.

```
exception smtplib.SMTPResponseException
```

Base class for all exceptions that include an SMTP error code. These exceptions are generated in some instances when the SMTP server returns an error code. The error code is stored in the smtp_code attribute of the error, and the smtp_error attribute is set to the error message.

```
exception smtplib.SMTPSenderRefused
```

Sender address refused. In addition to the attributes set by on all SMTPResponseException exceptions, this sets 'sender' to the string that the SMTP server refused.

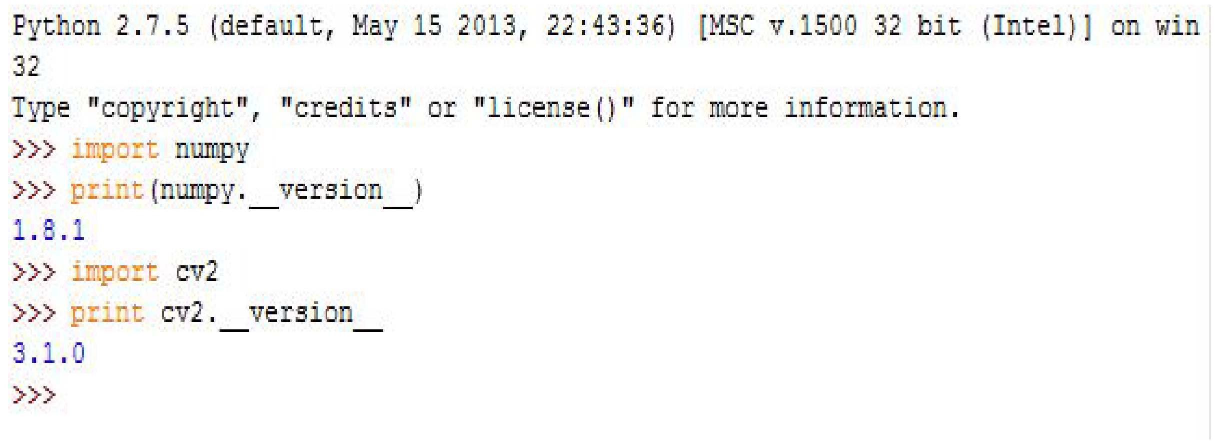
Open CV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.



```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import numpy
>>> print(numpy.__version__)
1.8.1
>>> import cv2
>>> print cv2.__version__
3.1.0
>>>
```

Fig. : Importing NumPy and OpenCV

The very first process in this system is object identification. The camera module is to detect a face from a running feed and capture a photograph in that instant. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images which is then used to detect objects in other images. For detecting a face, the program needs images with faces in them (positive images) and images without faces (negative images). The program then extracts the features from them. Each feature

A single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

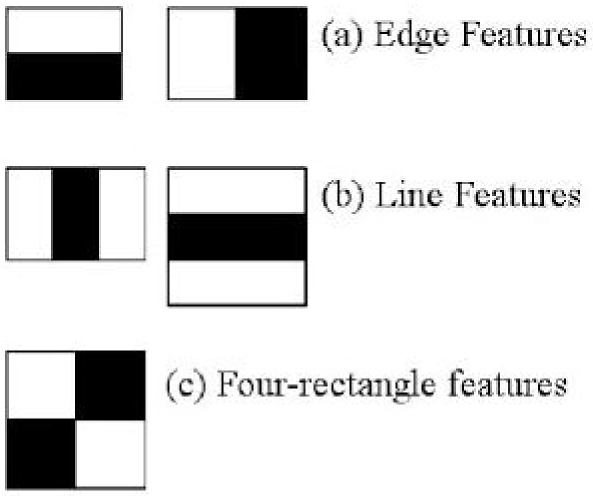


Fig. Extractable Features

Plenty of features are calculated by using the possible sizes and location of each kernel. This is done by using integral images which simplifies the calculation of pixels to an operation that requires only four pixels. Among the huge number of features that can be calculated, not all of them are useful for detection of objects (faces) in every case. The most relevant features need to be extracted from the collection. This is done by applying all the features to all of the training images. The features with the least error rate are chosen which means they are the features that best classifies the face and non-face images. Even after further classification, the final number of features that is needed to detect a face in an image is still very large which makes the process time consuming. However, a solution to this problem is possible because in an image, most of the image region is non-face region. For this, the concept of Cascade of Classifiers has been introduced. The features are grouped into different stages of classifiers and applied one-by-one on the window. If the first stage is a failure, the window is discarded entirely. Remaining 20

features on it are not taken into consideration. If it passes, the remaining stages of features are applied and the process is completed. The window which passes all stages is a face region. The steps to complete this process in Open CV are simplified in the following block diagram:

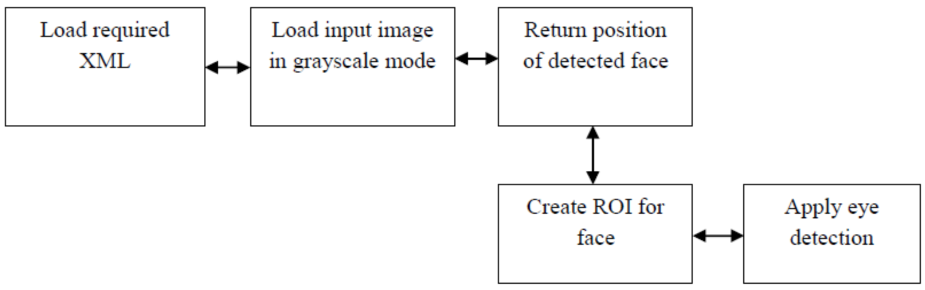

Load required XML
Load input image in grayscale mode
Return position of detected face
Create ROI for face
Apply eye detection

Fig. Haar Cascade process

Once a face region is detected, the image is immediately passed on to the next step of the process to identify if the face detected is that of a permissible individual or an intruder.

Facial Recognition is a computer application or process which is capable of identifying a person from an image or video feed. This is achieved by cross examining a given picture with an already available image on a database. The FaceRecognizer class for face recognition that comes with OpenCV includes the following algorithms:

- Eigenfaces (see createEigenFaceRecognizer())
- Fisherfaces (see createFisherFaceRecognizer())
- Local Binary Patterns Histograms (see createLBPHFaceRecognizer())

Web Designing

1 HTML

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively **easy to learn**, with the basics being accessible to most people in one sitting; and quite **powerful** in what it allows you to create. It is constantly undergoing revision and evolution to meet the demands and requirements of the growing Internet audience under the direction of the **» W3C**, the organisation charged with designing and maintaining the language.

The definition of HTML is **HyperText Markup Language**.

- *HyperText* is the method by which you move around on the web — by clicking on special text called **hyperlinks** which bring you to the next page. The fact that it is *hyper* just means it is not linear — i.e. you can go to any place on the Internet whenever you want by clicking on links — there is no set order to do things in.
- *Markup* is what **HTML tags** do to the text inside them. They mark it as a certain type of text (*italicised* text, for example).
- HTML is a *Language*, as it has code-words and syntax like any other language.

HTML consists of a series of short **codes** typed into a text-file by the site author — these are the tags. The text is then **saved as a html file**, and **viewed through a browser**, like *Internet Explorer* or *Netscape Navigator*. This browser reads the file and translates the text into a visible form, hopefully rendering the page as the author had intended. Writing your own HTML entails using tags correctly to create your vision. You can use anything from a rudimentary text-editor to a powerful graphical editor to create HTML pages.

The tags are what separate normal text from HTML code. You might know them as the words between the <angle-brackets>. They allow all the cool stuff like images and tables and stuff, just by telling your browser what to render on the page. Different tags will perform different functions. The tags themselves don't appear when you view your page through a browser, but their effects do. The simplest tags do nothing more than apply formatting to some text, like this:

```
<b>These words will be bold</b>, and these will not.
```

In the example above, the <b> tags were wrapped around some text, and their effect will be that the contained text will be bolded when viewed through an ordinary web browser.

If you want to see a list of a load of tags to see what's ahead of you, look at this tag reference. Learning the tags themselves is dealt with in the next section of this website, My First Site.

HTML VERSION	YEAR
HTML 1.0	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML 5 2014	

Elements and Tag:

HTML uses predefined tags and elements which tells the browser about content display property. If a tag is not closed then browser apples that effect till end of page.

CSS

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages of CSS

CSS saves time − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.

Pages load faster − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

Superior styles to HTML − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Multiple Device Compatibility − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

Global web standards − Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Who Creates and Maintains CSS?

CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by the W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE − The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

Cascading Style Sheets level 1 (CSS1) came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

CSS Syntax

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.
A style rule set consists of a selector and declaration block.

```
Selector => h1

Declaration => {color:blue;font size:12px;}
```

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.

  For Example:

  -> color is property and blue is value.

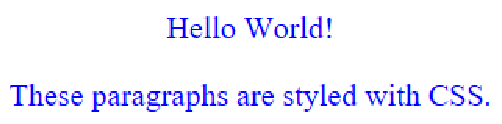
  -> font size is property and 12px is value.

- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example :

In the following example all p elements will be center-aligned, with a blue text color:

```
p {
    color: blue;
    text-align: center;
}
```



Hello World!

These paragraphs are styled with CSS.

Flutter

Flutter is an open-source mobile application development framework created by Google. It is used to develop applications for Android and iOS, as well as being the primary method of creating applications for Google Fuchsia.

The first version of Flutter was known as "Sky" and ran on the Android operating system. It was unveiled at the 2015 Dart developer summit, with the stated intent of being able to render consistently at 120 frames per second. During the keynote of Google Developer Days in Shanghai, Google announced Flutter Release Preview 2 which is the last big release before Flutter 1.0. On December 4th, 2018, Flutter 1.0 was released at the Flutter Live event, denoting the first "stable" version of the Framework.

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.[9]

On Android, and on Windows, macOS and Linux via the semi-official *Flutter Desktop Embedding* project, Flutter runs in the Dart virtual machine which features a just-in-time execution engine. Due to App Store restrictions on dynamic code execution, Flutter apps use ahead-of-time (AOT) compilation on iOS.

A notable feature of the Dart platform is its support for "hot reload" where modifications to source files can be injected into a running application. Flutter extends this with support for stateful hot reload, where in most cases changes to source code can be reflected immediately in the running app without requiring a restart or any loss of state. This feature as implemented in Flutter has received widespread praise.

CHAPTER – 10

CODING

10. Coding

Back End Coding

i. Main.py

```
import cv2
import sys
import databaseConfig
from mail import sendEmail
from flask import Flask, render_template, Response
from camera import VideoCamera
from flask_basicauth import BasicAuth
import time
import threading

email_update_interval = 60 # sends an email only once in this time interval
video_camera = VideoCamera(flip=True) # creates a camera object, flip vertically
object_classifier = cv2.CascadeClassifier("models/facial_recognition_model.xml")
# an opencv classifier

# App Globals (do not edit)
app = Flask(__name__)
#app.config['BASIC_AUTH_USERNAME'] = 'MINOR'
#app.config['BASIC_AUTH_PASSWORD'] = 'PROJECT'
#app.config['BASIC_AUTH_FORCE'] = True

#basic_auth = BasicAuth(app)
last_epoch = 0

def check_for_objects():

    if 'uid' not in session:
        return redirect('/')
    uid= session['uid']

        global last_epoch
        while True:
                try:
                        frame, found_obj , disp =
video_camera.get_object(object_classifier)
                        #print(time.time() - last_epoch)
```

```
                    if found_obj and (time.time() - last_epoch) >
email_update_interval:
                        last_epoch = time.time()
                        #cv2.imshow("get",frame)
                        print( "Sending email...")
                        sendEmail(frame,uid)
                        #cv2.imshow("get",disp)
                        print ("done!")
            except:
                    print ("Error sending email: ", sys.exc_info()[0])

#@app.route('/')
#@basic_auth.required
#def index():
 #   return render_template('index.html')


app = Flask(__name__)
app.secret_key = "TheFourthEye"

@app.route('/', methods = ['GET', 'POST'])
def index():
    if 'uid' in session :
        return redirect('dashboard')
    return render_template('index.html')

@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        passw = request.form['password']
        try:
            user = auth.sign_in_with_email_and_password(email,passw)
            info = auth.get_account_info(user['idToken'])['users'][0]
            # print(info)
            userDataHolder = {'id' : info['localId'], 'email' : info['email'], 'isVerified' :
info['emailVerified'], 'lastLogin' : info['lastLoginAt'], 'createdAt' : info['createdAt'],
'passUpdatedAt':info['passwordUpdatedAt'] }

            if userDataHolder['isVerified'] == False :
                auth.send_email_verification(user['idToken'])
                return """<div style="margin:auto;text-align:center;padding:30px;font-
size:22px;">
                <p>Please verify your email and <a href="/"> try again</a>
                </p>
                </div>"""
            session['uid']= userDataHolder['id']

        except :
            return """<div style="margin:auto;text-align:center;padding:30px;font-
size:22px;">
                <p>
                Error in signing in. Check your credentials and <a href="/"> try
again</a>
                </p>
                </div>"""

        print('Login ')
        return redirect('/dashboard')
    else:
        return redirect('/')

@app.route('/logout')
def logout():
    if 'uid' in session :
        session.pop('uid', None)
        return redirect('/')
    else:
        return redirect('/')

@app.route('/signup',methods=['POST','GET'])
def signup():
    if request.method == 'POST':
        name = request.form['signUpName']
        email = request.form['registerEmail']
        passw = request.form['registerPassword']

        profileInfo ={'name':name, 'email': email, 'contact':'', 'city':'', 'country':'',
'address':''}
        try:
            print(name,email,passw)
            user = auth.create_user_with_email_and_password(email,passw)

            info = auth.get_account_info(user['idToken'])['users'][0]
            # print(info)
            userDataHolder = {'id' : info['localId'], 'email' : info['email'], 'isVerified' :
info['emailVerified'], 'lastLogin' : info['lastLoginAt'], 'createdAt' : info['createdAt'],
'passUpdatedAt':info['passwordUpdatedAt'] }
            print(1)
            auth.send_email_verification(user['idToken'])
            print(2)
            db.child('users').child(userDataHolder['id']).set(profileInfo)
            print(3)
        except:
            return """<div style="margin:auto;text-align:center;padding:30px;font-
size:22px;">
                <p>
                    Error signnig up please <a href="/">try agin</a>.
                    </p>
                </div>"""
        print('Logout')
        return """<div style="margin:auto;text-align:center;padding:30px;font-
size:22px;">
            <p>
                Please verify your email and then <a href="/">login</a> to edit profile
and continue.
                </p>
            </div>"""
    else:
        return redirect('/')

@app.route('/live_feed')
def live_feed():
    if 'uid' not in session:
        return redirect('/')
    return render_template('live_feed.html')

@app.route('/dashboard')
def dashboard():
    if 'uid' not in session:
        return redirect('/')
    userId = session['uid']
    peopleName = None
    res = db.child('users').child(userId).child('people').get().val()
    if res :
        res = db.child('users').child(userId).child('people').get().val().values()
        peopleName=[]
        for r in res:
          peopleName.append(r['name'])
    print(peopleName)
    return render_template('dashboard.html',peopleName=peopleName)

@app.route('/switch_control')
def switch_control():
    if 'uid' not in session:
        return redirect('/')
    return render_template('switch_control.html')
```

```
@app.route('/profile')
def profile():
   if 'uid' not in session:
      return redirect('/')

   return render_template('profile.html')

@app.route('/profile/<name>')
def named_profile(name):
   if 'uid' not in session:
      return redirect('/')
   return render_template('namedProfile.html',name=name)

@app.route('/people')
def people():
   if 'uid' not in session:
      return redirect('/')
   return render_template('people.html')


def gen(camera):
   while True:
      frame = camera.get_frame(object_classifier)
      yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
   if 'uid' not in session:
      return redirect('/')
   return Response(gen(video_camera),
               mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
   t = threading.Thread(target=check_for_objects, args=())
   t.daemon = True
   t.start()
   app.run(host='0.0.0.0', debug=False)
```

ii. databaseConfig.py

```
import pyrebase
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

mailInfo ={
   'email':'mail@gmail.com',
   'password':'PASSWORD',
   'subject':'FourthEye '
   }

def notifyPeople(user,toMail,name):
   server = smtplib.SMTP('smtp.gmail.com',587)
   server.starttls()
   server.login(mailInfo['email'],mailInfo['password'])
   msg = MIMEMultipart()
   msg['From'] = mailInfo['email']
   msg['To'] = toMail
   msg['Subject'] = mailInfo['subject'] + "- "+user+" added you"
   body = "Dear "+name+",<br> &nbsp; &nbsp; The FourthEye user
<strong>"+user+"</strong>, added you as a trusted person. Now you will also
receive security alert from <strong>The
FourthEye</strong><br><em>Congratulations</em><br><br><strong>Thank
you</strong><br><em>The FourthEye Team</em><br>GEC Bilaspur"
   msg.attach(MIMEText(body,'html'))
   text = msg.as_string()
   try:
      server.sendmail(mailInfo['email'],toMail,text)
   except Exception as e:
      print("Error in sending mail. Error : "+ e)
   finally:
      server.quit()

def sendAlertToAll(uid):
        email =db.child('users').child(uid).child('people').get().val()
        sendList = [email]
        res = db.child('users').child(uid).child('people').get().val()
        if res:
                peopleData =
db.child('users').child(uid).child('people').get().val().values()
                # print(umail)
                emailList=[ data['email'] for data in peopleData ]
                emailsString = ','.join(emailList)
                sendList = sendList+emailList
        return sendList

cred = {
   'apiKey': "<firebase apiKey>",
   'authDomain': "app_domain.firebaseapp.com",
   'databaseURL': "https://app_domain.firebaseapp.com",
   'projectId': "fourtheye",
   'storageBucket': "fourtheye.appspot.com",
   'messagingSenderId': "8798798779"
}


firebase = pyrebase.initialize_app(cred)
auth = firebase.auth()

db = firebase.database()
```

iii. camera.py

```
import cv2
from imutils.video.pivideostream import PiVideoStream
import imutils
import time
import numpy as np

class VideoCamera(object):
   def __init__(self, flip = False):
      self.vs = PiVideoStream(resolution=(480 , 320),framerate=32).start()
#resolution and framerate
      self.flip = flip
      time.sleep(0.1)

   def __del__(self):
      self.vs.stop()

   def flip_if_needed(self, frame):
      if self.flip:
         return np.flip(frame, 0)
      return frame

   def get_frame(self, classifier):
      frame = self.flip_if_needed(self.vs.read()).copy()
      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      objects = classifier.detectMultiScale(
         gray,
         scaleFactor=1.5,
         minNeighbors=4,
         minSize=(30, 30),
         flags=cv2.CASCADE_SCALE_IMAGE
```

```
        )
        for (x, y, w, h) in objects:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        ret, jpeg = cv2.imencode('.jpg', frame)
        return jpeg.tobytes()

    def get_object(self, classifier):
        found_objects = False
        frame = self.flip_if_needed(self.vs.read()).copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        objects = classifier.detectMultiScale(
            gray,
            scaleFactor=1.5,
            minNeighbors=4,
            minSize=(30, 30),
            flags=cv2.CASCADE_SCALE_IMAGE
        )

        if len(objects) > 0:
            found_objects = True

        # Draw a rectangle around the objects
        for (x, y, w, h) in objects:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        ret, jpeg = cv2.imencode('.jpg', frame)
        #print('called = ', found_objects )
        return (jpeg.tobytes(), found_objects,frame)
```

iv. mail.py

```
import smtplib
#from email.MIMEMultipart import MIMEMultipart
#from email.MIMEText import MIMEText
#from email.MIMEImage import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
import databaseConfig

# Email (only works with gmail)
fromEmail = '<EMAIL>'
```

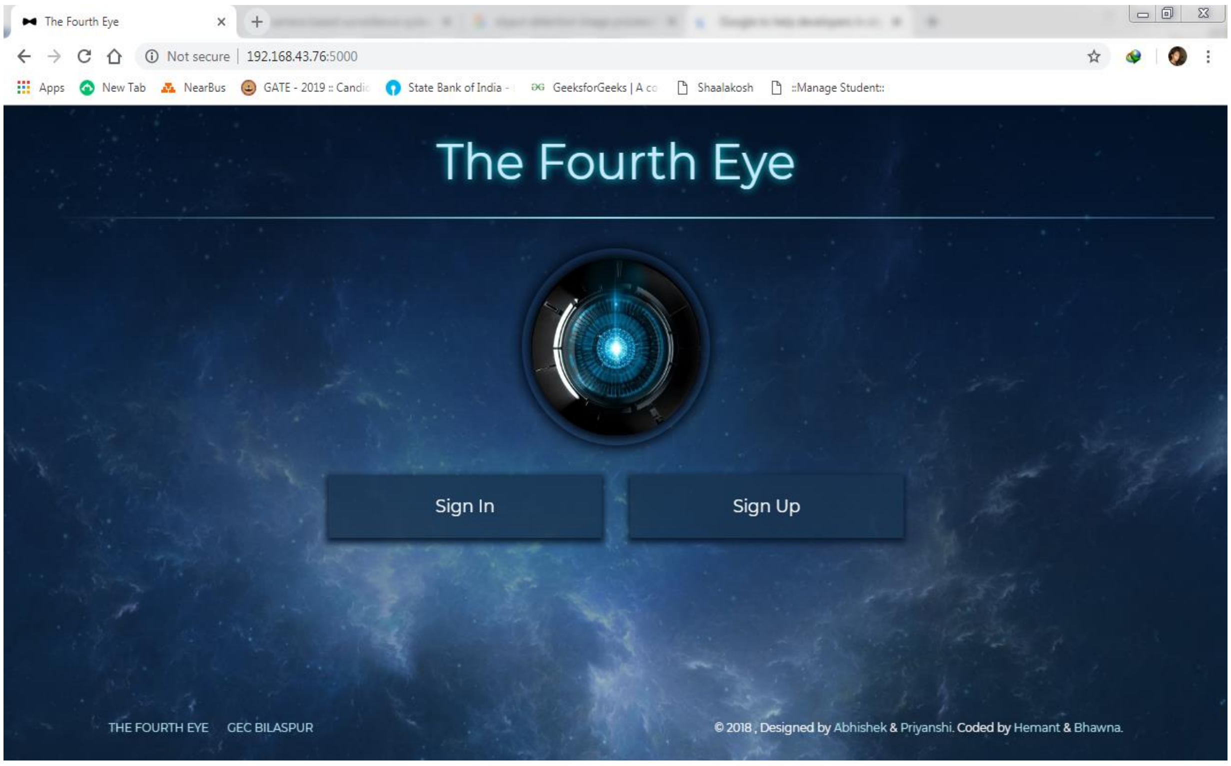
```
# You can generate an app password here to avoid storing your password in plain
text
# https://support.google.com/accounts/answer/185833?hl=en
fromEmailPassword = '<PASSWORD>'

# Email you want to send the update to
def sendEmail(image,uid):
        toMail = sendAlertToAll(uid)
        msgRoot = MIMEMultipart('related')
        msgRoot['Subject'] = 'The Fourth Eye - Security Update'
        msgRoot['From'] = fromEmail
        # msgRoot['To'] = toEmail
        msgRoot.preamble = 'Raspberry pi security camera update'

        msgAlternative = MIMEMultipart('alternative')
        msgRoot.attach(msgAlternative)
        msgText = MIMEText('The Fourth Eye cam detected Somone')
        msgAlternative.attach(msgText)

        text = '<strong>Some suspecious activity detected. Take a look.</strong>
<br><img src="cid:image1"> <br> <em>The FourthEye</em>'
        msgText = MIMEText(text, 'html')
        msgAlternative.attach(msgText)

        msgImage = MIMEImage(image)
        msgImage.add_header('Content-ID', '<image1>')
        msgRoot.attach(msgImage)

        smtp = smtplib.SMTP('smtp.gmail.com', 587)
        smtp.starttls()
        smtp.login(fromEmail, fromEmailPassword)
        smtp.sendmail(fromEmail, toEmail, msgRoot.as_string())
        smtp.quit()
```

CHAPTER – 11

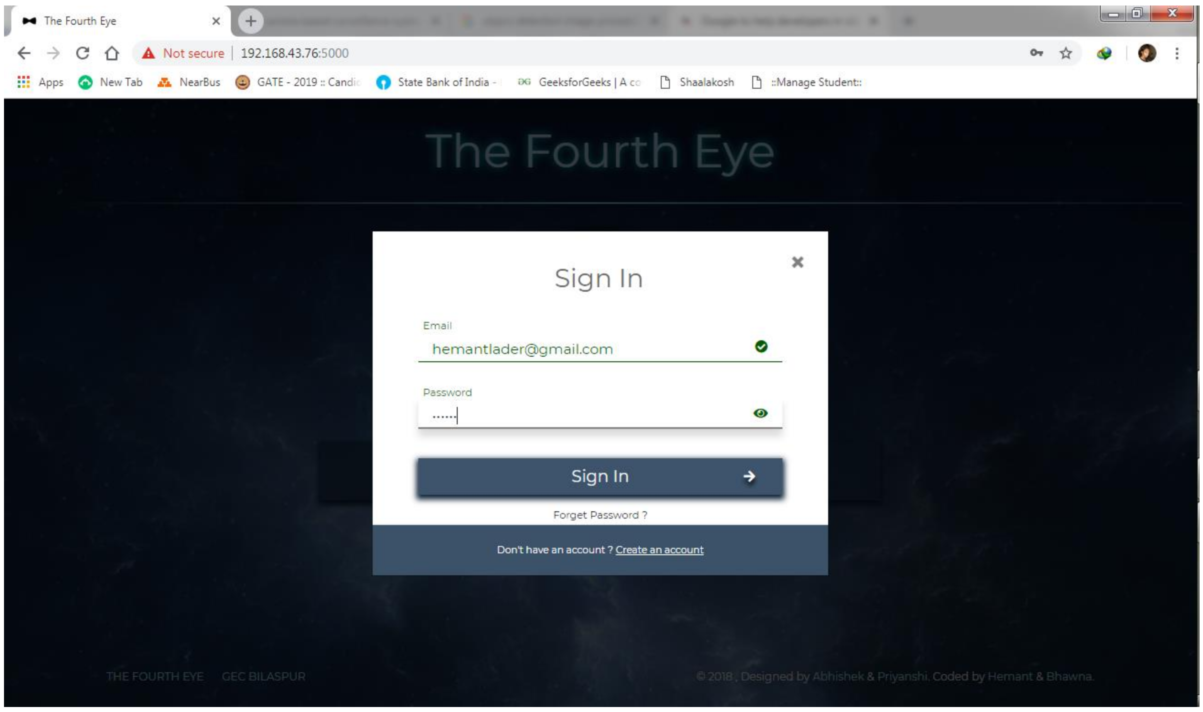
SCREENSHOTS

11. Screenshots

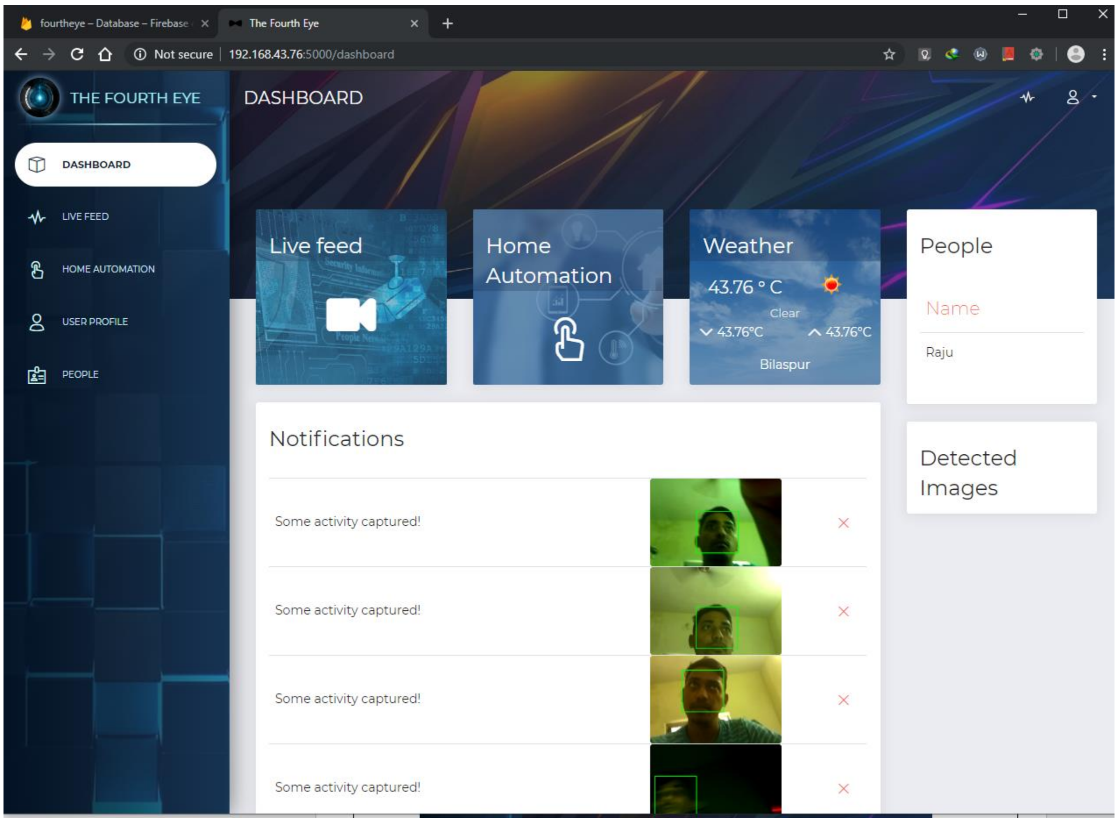
Home Page



Sign In Page



Dashboard



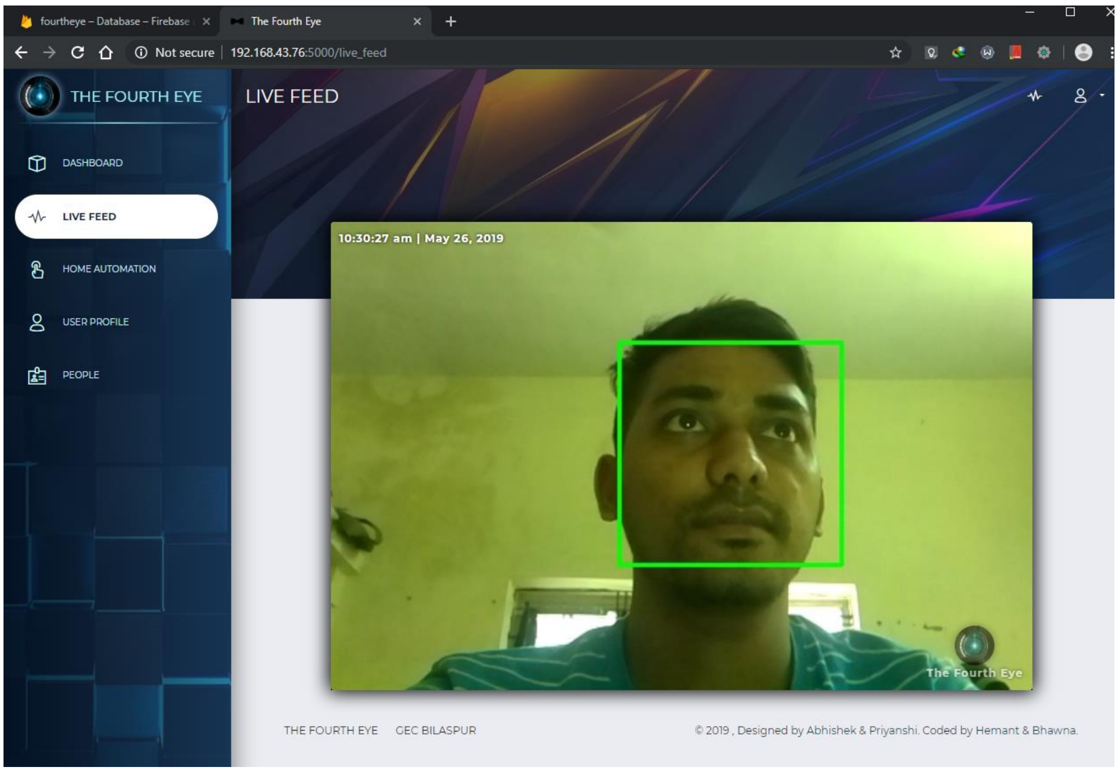
Some activity captured!

Some activity captured!

Some activity captured!

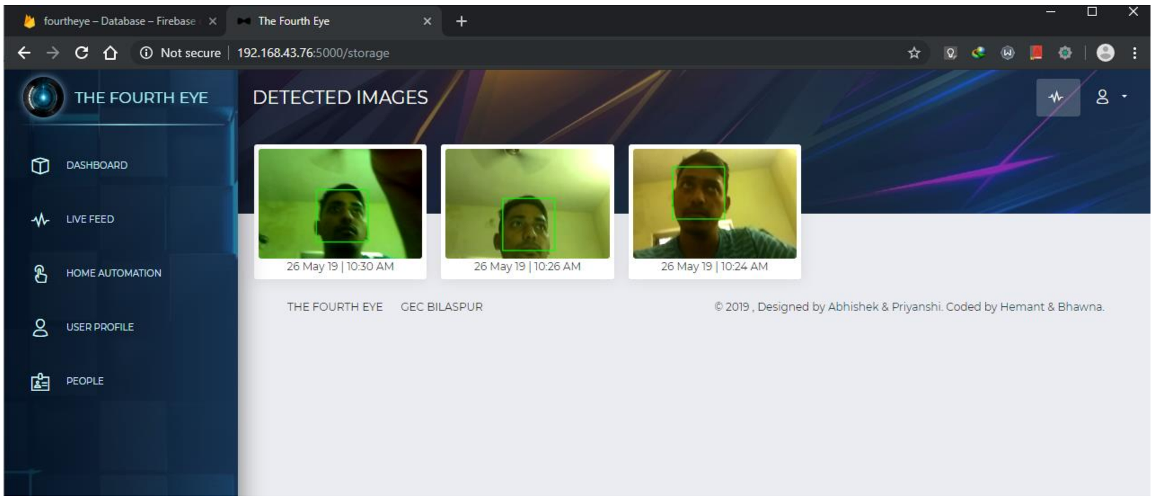
Some activity captured!

Live Feed

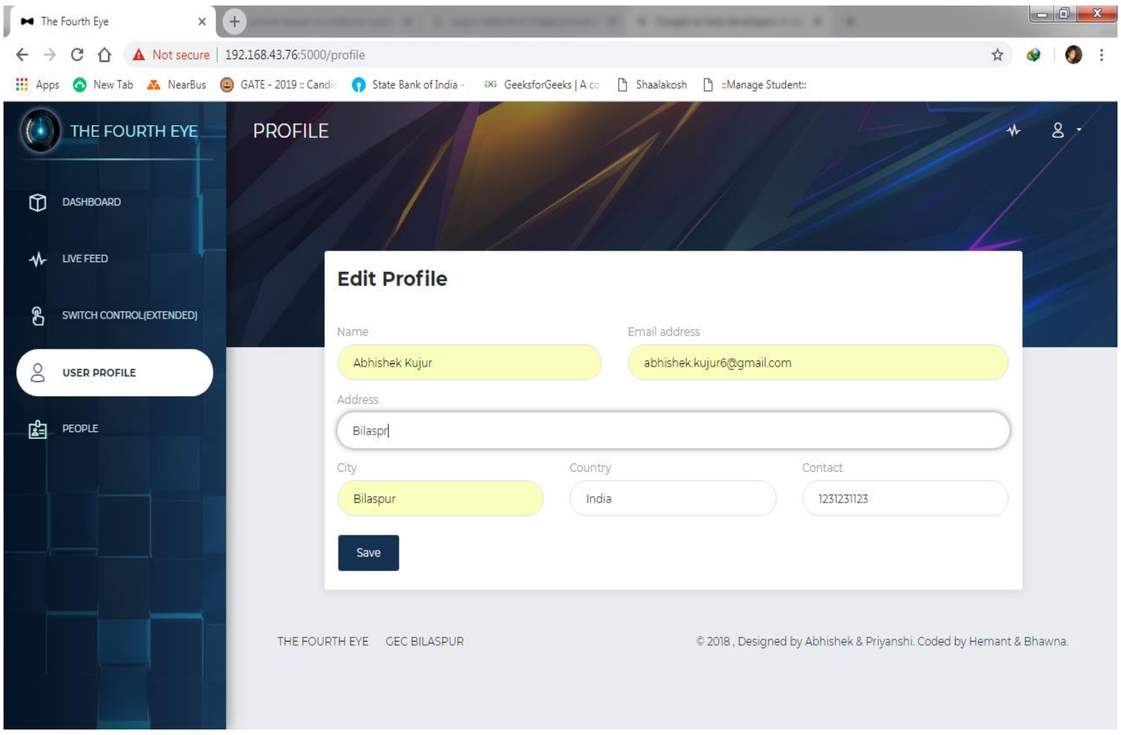


© 2019 , Designed by Abhishek & Priyanshi. Coded by Hemant & Bhawna.

Detected Images



User Profile

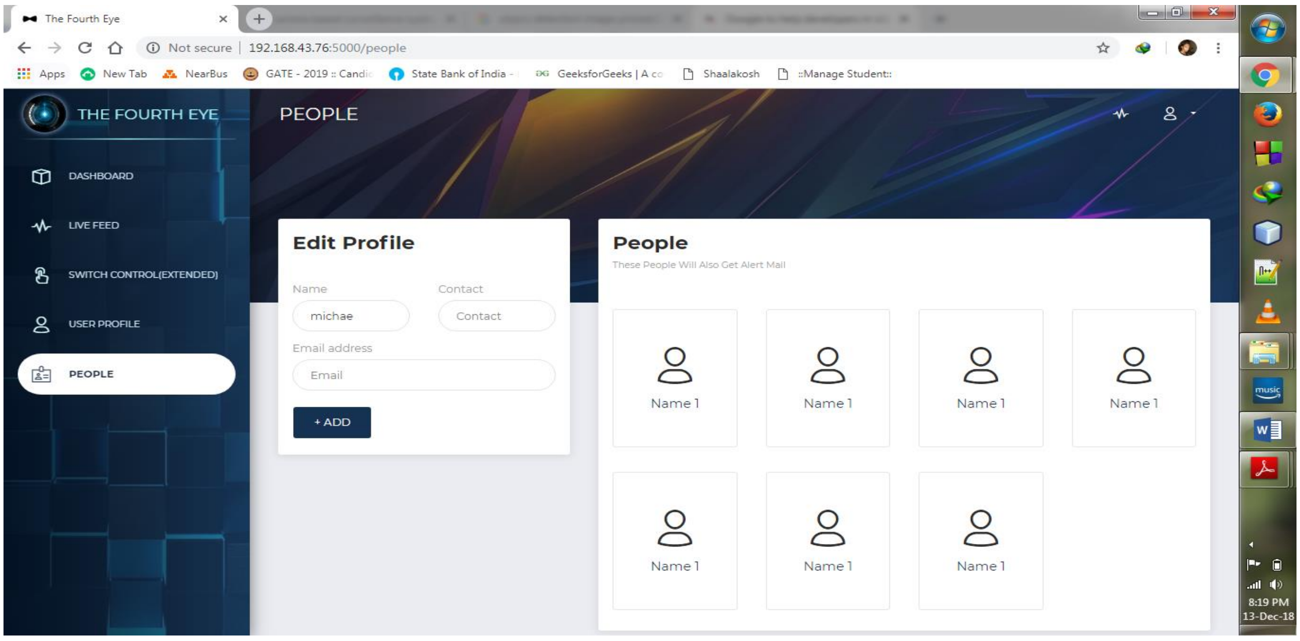


Edit Profile

Name: Abhishek Kujur
Email address: abhishek.kujur6@gmail.com
Address: Bilaspr
City: Bilaspur
Country: India
Contact: 1231231123

Save
THE FOURTH EYE
GEC BILASPUR
© 2018 , Designed by Abhishek & Priyanshi. Coded by Hemant & Bhawna.

Trusted People (They'll also get notified)


The Fourth Eye
Not secure | 192.168.43.76:5000/people
Apps New Tab NearBus GATE - 2019 :: Candi State Bank of India GeeksforGeeks | A co Shaalakosh ::Manage Student::
THE FOURTH EYE
PEOPLE
DASHBOARD
LIVE FEED
SWITCH CONTROL(EXTENDED)
USER PROFILE
PEOPLE

Edit Profile

Name: michae

Contact: Contact

Email address: Email

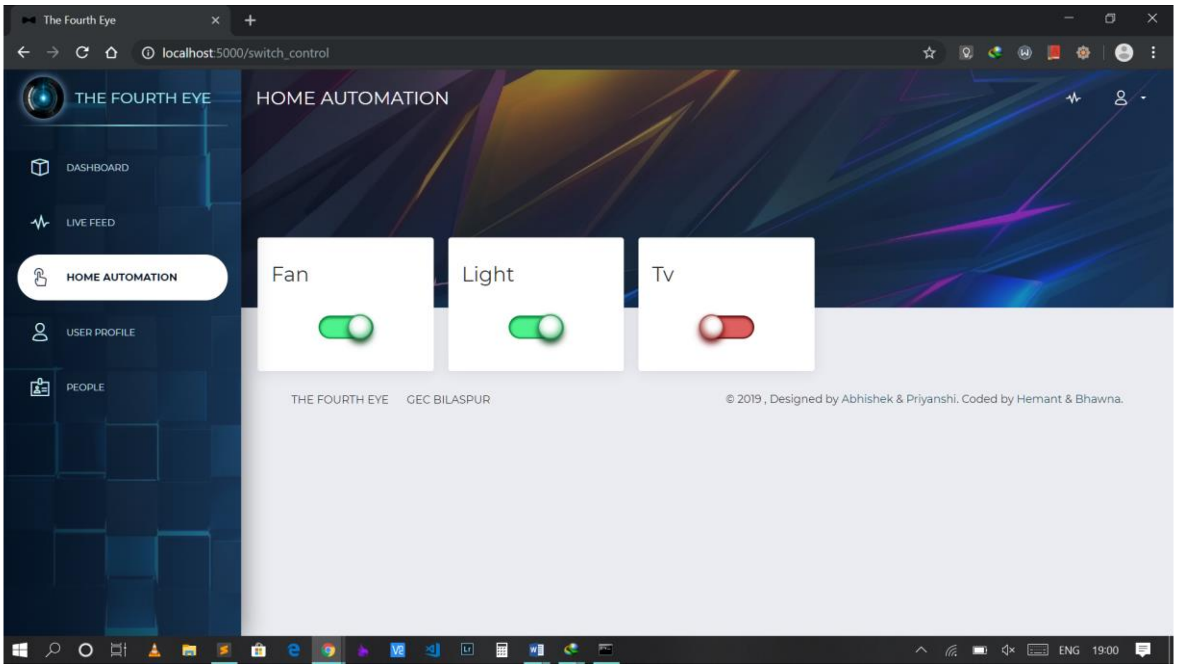
+ ADD

People

These People Will Also Get Alert Mail

Name 1
Name 1
Name 1
Name 1
Name 1
Name 1
Name 1

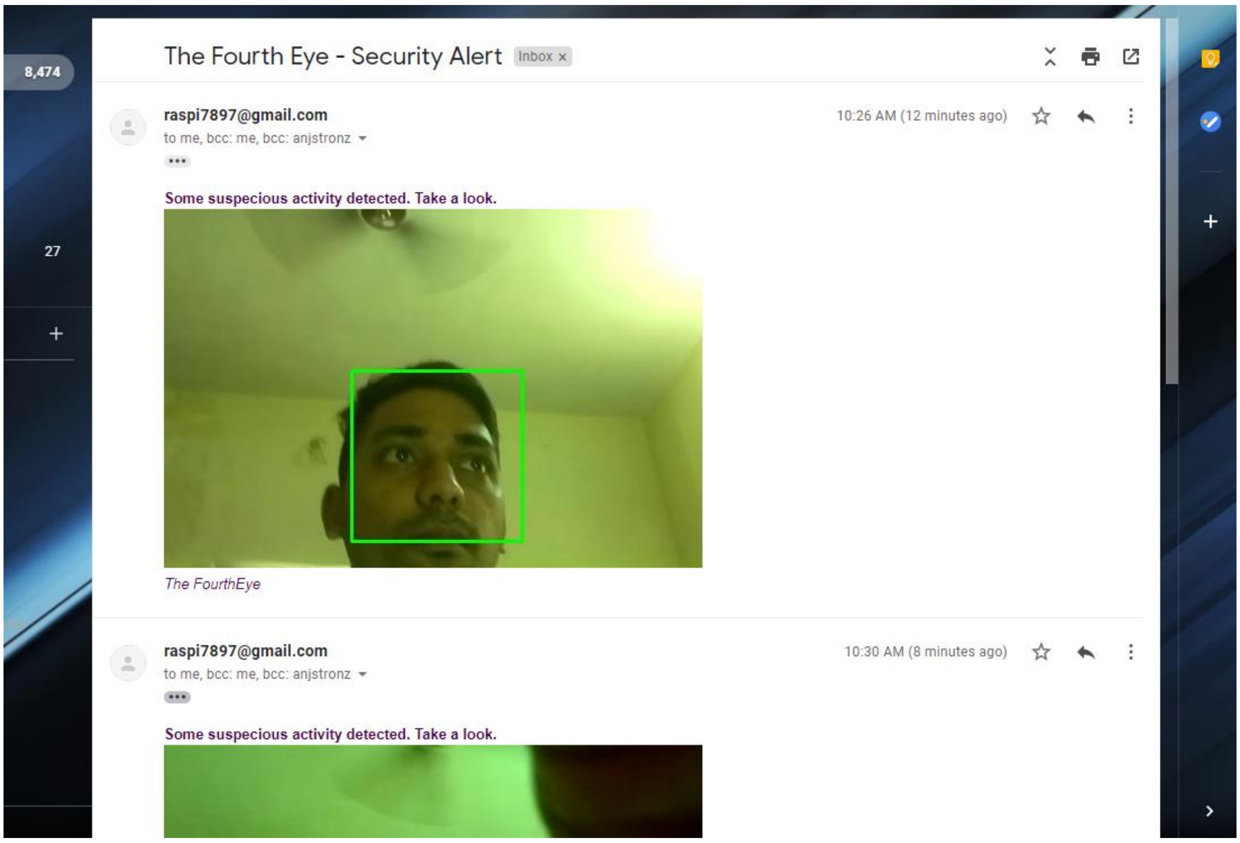
Home Automation



THE FOURTH EYE GEC BILASPUR

© 2019 , Designed by Abhishek & Priyanshi. Coded by Hemant & Bhawna.

Security alert (e-mail notification)


The Fourth Eye - Security Alert Inbox ×

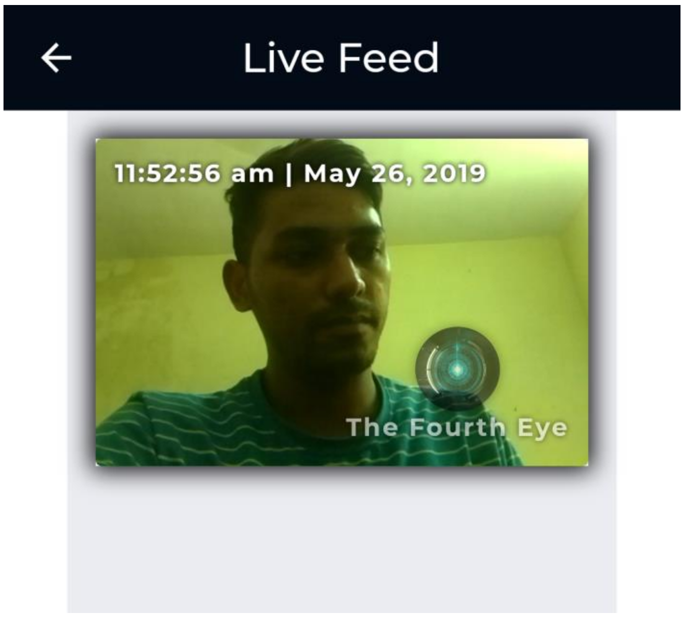
raspi7897@gmail.com
to me, bcc: me, bcc: anjstronz
10:26 AM (12 minutes ago)

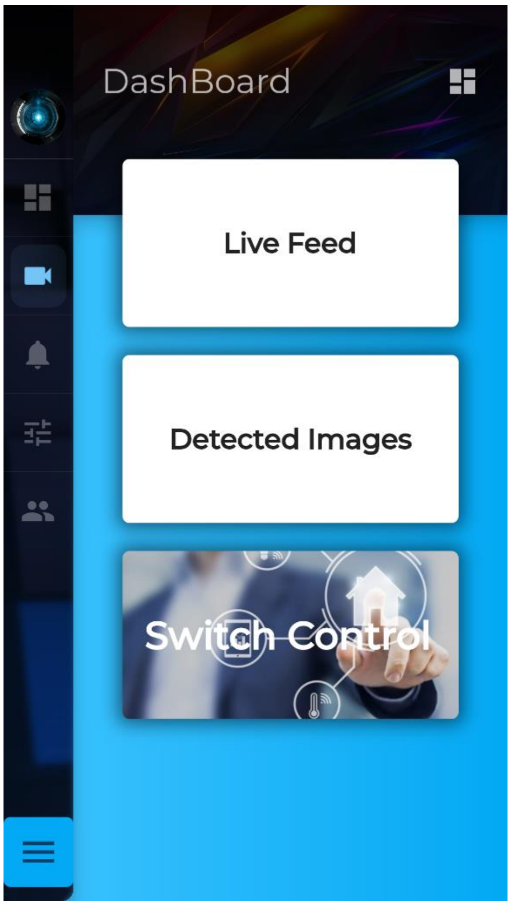
Some suspecious activity detected. Take a look.

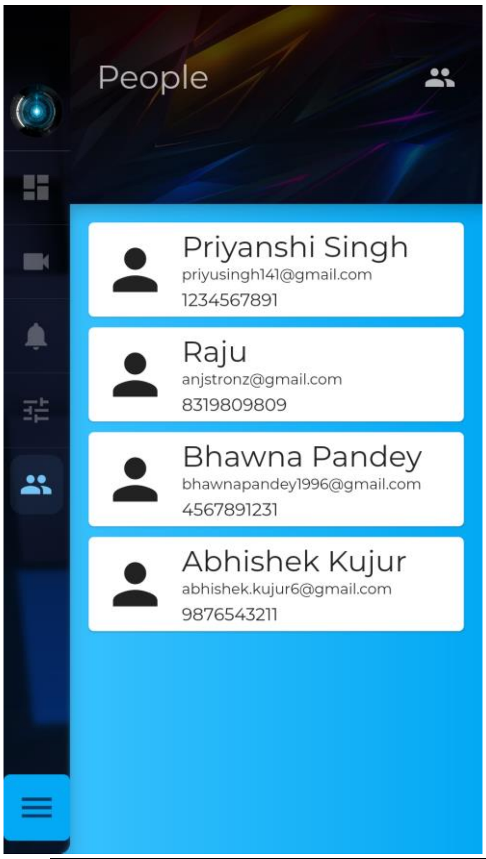
The FourthEye

raspi7897@gmail.com
to me, bcc: me, bcc: anjstronz
10:30 AM (8 minutes ago)

Some suspecious activity detected. Take a look.





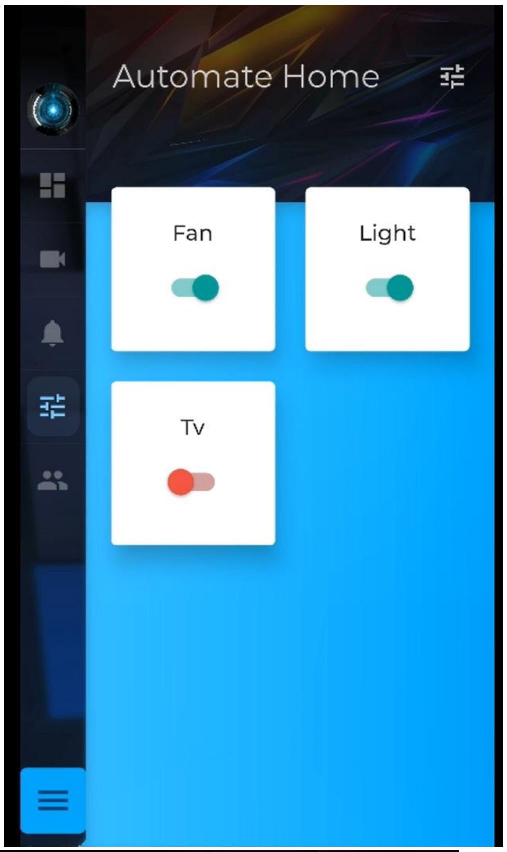


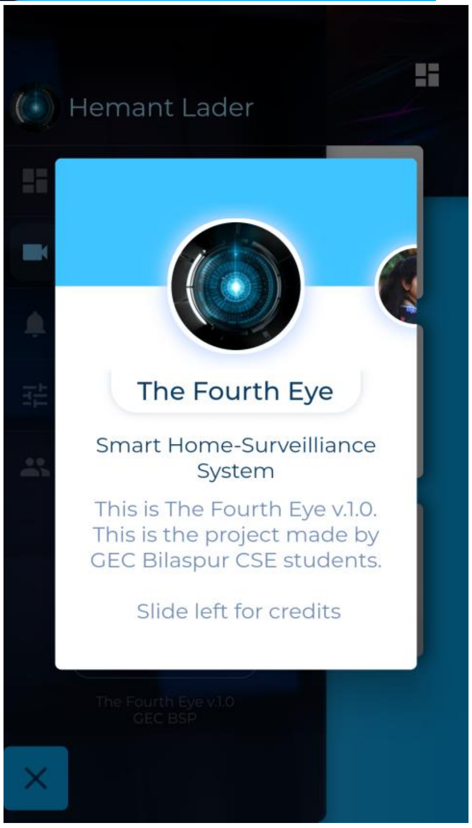
Priyanshi Singh
priyusingh141@gmail.com
1234567891

Raju
anjstronz@gmail.com
8319809809

Bhawna Pandey
bhawnapandey1996@gmail.com
4567891231

Abhishek Kujur
abhishek.kujur6@gmail.com
9876543211



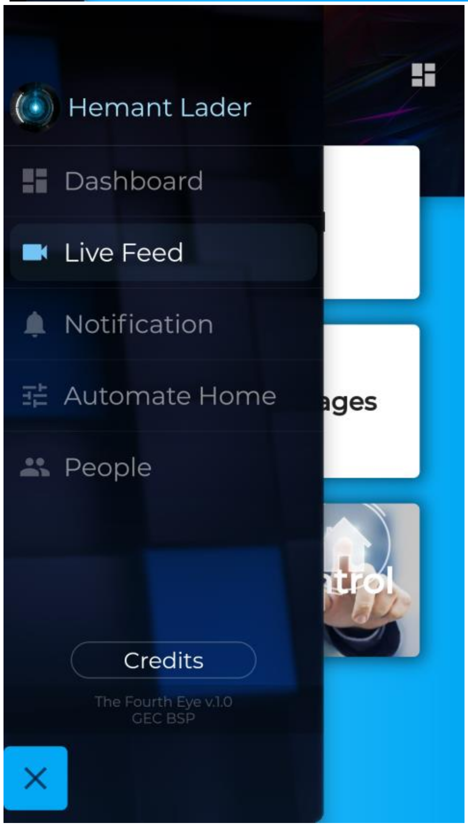


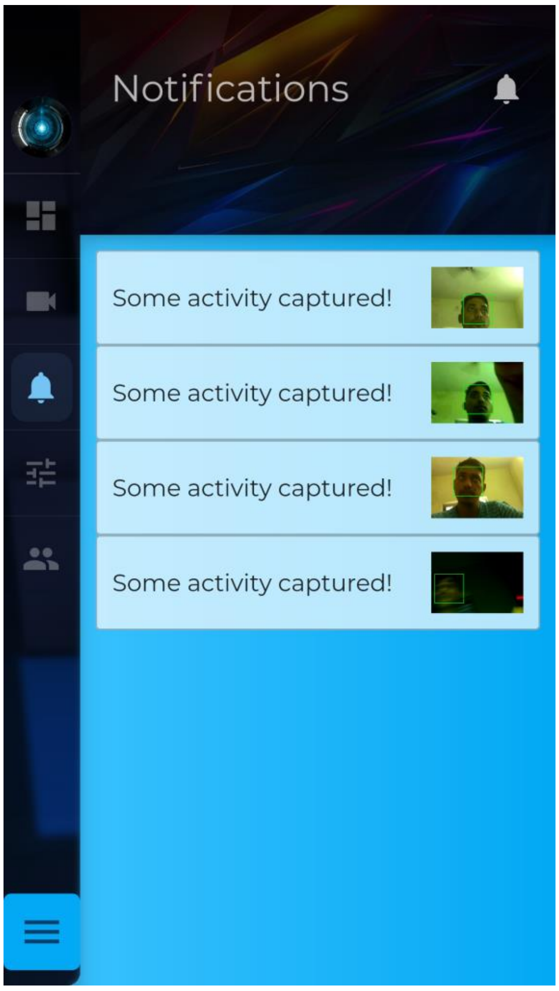
The Fourth Eye

Smart Home-Surveilliance System

This is The Fourth Eye v.1.0. This is the project made by GEC Bilaspur CSE students.

Slide left for credits



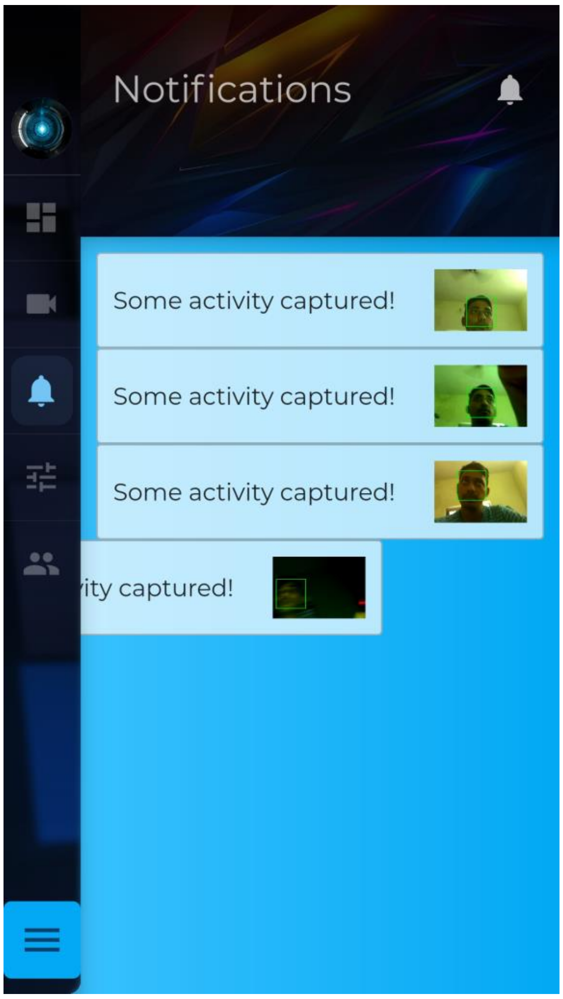

Notifications

Some activity captured!

Some activity captured!

Some activity captured!

Some activity captured!


Notifications

Some activity captured!

Some activity captured!

Some activity captured!

ity captured!

CHAPTER – 12

CONCLUSION

12. Conclusion

In our project, we have tried to come up with a prototype of an intelligent security system which has reliability and mobility. With our project and paper, we have tried to integrate Raspberry Pi and IOT based to achieve a real-time security system. There are existing models of security but none of those let the user receive real time image of the intruder especially in Bangladesh. The other applications are also very much required and practical for the context of educational institutions and work places to reduce the hassle everyone is facing now days. In this era of extreme chaos, we need one less thing to worry about and that is about our security. It is time to move forward and include more features and mobility to our security systems so we can get updates on the go. This security system is only a prototype which can be developed more for commercial use and also has immense application especially in the context of Bangladesh in recent times. Though our project is now seen to be a bit expensive but which more time the costs can be reduced much more so that this can be a practical system so more people can have access to an advanced security model.

CHAPTER – 13

BIBLIOGRAPHY

13. <u>Bibliography</u>

- Mrutyunjaya Sahani, Chiranjiv Nanda, Abhijeet Kumar Sahu and Biswajeet Pattnaik, “Web-Based Online Embedded Door Access Control and Home Security System Based on Face Recognition” 2015 International Conference on Circuit, Power and Computing Technologies [ICCPCT]
- Rhythm Haji,Arjun Trivedi, Hitarth Mehta, ProfA.B.Upadhyay “Implementation of Web-Surveillance using Raspberry Pi”International Journal of Engineering Research & Technology (IJERT) Vol. 3 Issue 10, October- 2014, IJERT.
- Jinsoo Han; Chang-Sic Choi; Ilwoo Lee, "More efficient home energy management system based on ZigBee communication and infrared remote controls," Consumer Electronics, IEEE Transactions on ,vol.57, no.1, pp.85,89, February 2011.
- Erdem, H.; Uner, A., "A multi-channel remote controller for home and office appliances," Consumer ElectronicsConsumer Electronics, IEEE Transactions on , vol.52, no.3, pp.837,843, Aug. 2006.
- https://raspberrypi.org
- https://firebase.google.com/
- https://en.wikipedia.org/wiki/Python_(programming_language)
- http://www.numpy.org/
- http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html
- http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#face-recognition-with-opencv