

# **CHILDREN'S EYE SAFETY MONITORING SYSTEM FOR TVs USING RASPBERRY PI**

A project report submitted in partial fulfillment of the requirements

for the award of credits to

**Bachelor of Technology**

In

**ELECTRONICS & COMMUNICATION ENGINEERING**

By

A.Surya Mohan Kiran (20BQ1A0408)

B.Naveen (20BQ1A0424)

U.Bharath Kumar (20BQ1A0418)

B.Pavan Kalyan (20BQ1A0427)

B.Chandrashekar (20BQ1A0414)

**IOT Tools And Applications  
(Skill Advanced Course)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING  
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY  
(AUTONOMOUS)**

(Approved by AICTE and permanently affiliated to JNTUK, Accredited by NBA and NAAC)

**NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508**

**OCTOBER 2023**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBURU**  
**(AUTONOMOUS)**



**CERTIFICATE**

This is to certify that the project titled **CHILDREN'S EYE SAFETY MONITORING SYSTEM FOR TVs USING RASPBERRY PI** is a bonafied record of work done by **A.SuryaMohanKiran** (20BQ1A0408), **B.Naveen** (20BQ1A0424), **U.BharatKumar**(20BQ1A0418),**B.PavanKalyan**(20BQ1A0427),**B.ChandraShekar**(20BQ1A0414) under the guidance of Sri.M.PARDHA SARADHI, Associate Professor in partial fulfillment of the requirement of the degree for Bachelor of Technology in Electronics and Communication Engineering during the academic year 2023-2024.

**Mr. S. NAGA RAJU**  
**COURSE INSTRUCTOR**

**PROF. M.Y. BHANU MURTHY**  
**HEAD OF THE DEPARTMENT**

## DECLARATION

We, **A.Surya Mohan Kiran (20BQ1A0408),B.Naveen (20BQ1A0424), U.BharatKumar(20BQ1A0418)B.Pava Kalyan(20BQ1A0427),B.Chandrashekar (20BQ1A0414)**, hereby declare that the project report entitled “**CHILDREN’S EYE SAFETY MONITORING SYSTEM FOR TVs USING RASPBERRY PI**” is done by us under the guidance of **Sri.M.Pardha Saradhi, Associate Professor** as part of the **IOT Tools And Applications (Skill Advanced Course)** in partial fulfillment of the requirement of the degree for Bachelor of Technology in Electronics and Communication Engineering during the academic year 2023–2024.

**Place :-** VVIT, Namburu

**Date :-** 12-10-2023

**Signature of the candidates**

**A.Surya Mohan Kiran,**

**B.Naveen,**

**U.Bharat Kumar,**

**B.Pavan Kalyan,**

**B.Chandra Shekar.**

## **ACKNOWLEDGEMENT**

We express our sincere thanks wherever it is due

We express our sincere thanks to the Chairman, Vasireddy Venkatadri Institute of Technology, Sri Vasireddy VidyaSagar for providing us well equipped infrastructure and environment.

We thank Dr. Y. Mallikarjuna Reddy, Principal, Vasireddy Venkatadri Institute of Technology, Nambur, for providing us the resources for carrying out the project.

Our sincere thanks to Dr. K. Giri babu, Dean of Studies for providing support and stimulating environment for developing the project.

Our heartfelt thanks to Dr. M. Y. Bhanu Murthy, Head of the Department, Department of ECE, for his co-operation and guidance which help us to make our project successful and complete in all aspects.

We also express our sincere thanks and are grateful to our guide Sri.M.Pardha Saradhi, Associate Professor, Department of ECE, for motivating us to make our project successful and fully complete. We are grateful for his precious guidance and suggestions.

Also, it our duty to extend heartfelt thanks to our course instructor Mr. S. Nagaraju, Assistant Professor for his valuable instructions to conclude this work.

We also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

### **NAME OF THE CANDIDATE:**

**A.Surya Mohan Kiran(20BQ1A0408)**

**B.Naveen(20BQ1A0424)**

**U.Bharat Kumar(20BQ1A0418)**

**B.Pavan Kalyan(20BQ1A0427)**

**B.Chandra Shekar(20BQ1A0414)**

# TABLE OF CONTENTS

	Pg.no
<b>ABSTRACT</b>	<b>iii</b>
<b>1.INTRODUCTION AND INSTALLATION</b>	
INTRODUCTION	1
1.1 SETTING UP RASPBERRY PI	
1.1.1 INSTALLING THE OS IMAGE FOR RASPBERRY PI	2
1.1.2 FLASHING THE OS IMAGE IN THE MEMORY CARD	3
1.1.3 BOOTING THE RASPBERRY PI	5
1.1.4 INSTALLATION OF VNC VIEWER IN DESKTOP	6
1.1.5 CONNECTING RASPI REMOTELY TO VNC VIEWER	7
1.2 LIBRARIES INSTALLATION	9
<b>2.HARDWARE AND SOFTWARE DESCRIPTION</b>	
2.1 HARDWARE DESCRIPTION	12
2.2 SOFTWARE DESCRIPTION	14
<b>3.IMPLEMENTATION</b>	
3.1 SOFTWARE IMPLEMENTATION	
3.1.1 Face detection and Eye detection	16
3.1.2 Distance detection	19
3.1.3 Mail alert	21
3.1.4 Accessing google spread sheets	22
3.1.5 Time based actions( calculating duration)	24
3.2 HARDWARE IMPLEMENTATION	25
<b>4.WORKING AND RESULTS</b>	
4.1 WORKING	27
4.2 RESULTS	27
<b>5.CONCLUSION AND FUTURESCOPE</b>	29
<b>REFERENCES</b>	30
<b>APPENDIX</b>	31

# LIST OF FIGURES

<b>Fig. No.</b>	<b>Figure Name</b>	<b>Page. No.</b>
1.1	Safe Distance	1
1.2	Raspberry Pi OS image	3
1.3	Writing the OS	4
1.4	Raspberry pi Window	4
1.5	VNC Viewer website	5
1.6	Direct Enabling of VNC	6
1.7	Enabling VNC through terminal	7
1.8	VNC viewer	8
2.1	Raspberry Pi 3B+	11
2.2	32GB memory card	11
2.3	Ultrasonic sensor	12
2.4	Raspberry pi 3 camera module	12
2.5	Buzzer	13
3.1	Theoretical face model	17
3.2	Face detection	18
3.3	Interfacing ultrasonic with raspi	19
3.4	Ultrasonic sensor output	21
3.5	Raspi camera interface with raspi	26
3.6	PIN configuration of Raspi 3b+	26
3.7	Hardware setup	27
4.1	Mail alert	29
4.2	Google spread sheet data	30

# ABSTRACT

The most neglected issue that is of the child eyes safety while watching the television from a closer distance eventually causing degradation of the retina. Due to which most of the children get opted to specs in small age. So, our system which is a model based on the concepts of Internet of Things and Sensors can reduce the rate of retina degradation at some extent. If some critical distance is maintained from the television screen then it will reduce the rate of strain on eyes while watching television. The system is totally controlled by the Raspberry Pi 3 Model B/B+. The Sensors like proximity sensor, raspberry pi camera module with computer vision library are used for distance measuring and child detection respectively. This system if developed further can also be used for the child tracking using the camera and sensors.

The safe distance for watching the television is around 2 to 3 meters from the television screen, so we are going to detect the distance using the ultrasonic sensor and also the camera is used for detecting the object i.e., children whether it is looking towards the television or any other area. If the system finds the critical distance crossed it will wait timer of 15 seconds to finish and then the Raspberry Pi will trigger the buzzer, if the same warnings are done for two times, the raspberry pi will trigger a SMTP email protocol, which will send the image of the child and a Text alert to parents and after sending the Email, the tv will be Turned off and the duration, time and date all the data is updated in Google spread sheets, so that parents can know how much time kids are watching Tv in Day.

# CHAPTER 1

## INTRODUCTION AND INSTALLATION

We're thrilled to introduce you to our comprehensive project report, which dives deep into the development and implementation of our groundbreaking solution - a Children's Eye Safety Monitoring System designed for television. This project was born out of our deep concern for the eye health of children in today's digital era. As screens have become an inseparable part of our daily lives, it's crucial to strike a balance between the enjoyment of entertainment and the preservation of well-being, especially for our young ones. With this mission as our guiding light, our dedicated team embarked on a journey to create a solution that not only safeguards children's eye health but also cultivates responsible screen time habits.

We will meticulously guide you through the intricate workings of our Children's Eye Safety Monitoring System. We'll shine a light on the technology and methodology behind it, unveiling the significant impact it has on nurturing healthy screen habits among children. Our system is more than just a watchful eye; it actively detects eye activity, encourages regular breaks, and even sends timely alerts to parents or guardians when necessary. Furthermore, we'll delve into the technical intricacies of our project. From harnessing facial recognition technology to seamlessly integrating email alerts, we've left no stone unturned to ensure the system operates efficiently and effectively. We'll also share the invaluable data and insights acquired during the testing phase, providing tangible evidence of the system's real-world effectiveness.

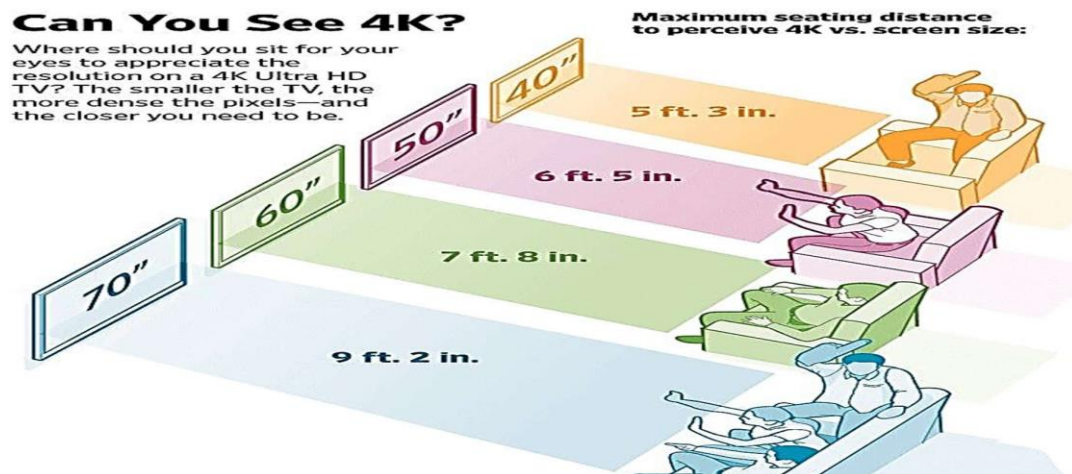


Fig 1.1: Safe Distance



## **HARDWARE REQUIREMENTS:**

1. Raspberry pi 3B+ board
2. Ultrasonic sensor
3. Raspberry pi 3 camera module
4. Buzzer
5. Led and resistor
6. 32GB memory card
7. Bread board and connecting wires

## **REQUIRED LIBRARIES**

1. OPENCV(CV2)
2. REQUESTS
3. GSPREAD
4. OAUTH2CIENT
5. SMTP LIB
6. RPIO.GPIO
7. TIME
8. DATE TIME

To do all this installations in raspberry pi and connecting all the sensors and modules, First we have to set up the raspberry pi board.

### **1.1 To setup the raspberry pi board the following steps involved:**

Setting up a Raspberry Pi board involves a series of steps. First, you need to install the OS imager designed for Raspberry Pi. Once that's done, you'll proceed to flash the OS image onto the memory card. After successfully flashing the image, the next step is to boot up the Raspberry Pi. To enable remote access and control, you'll need to install a VNC viewer on your desktop. Finally, you can establish a remote connection to the Raspberry Pi through the VNC viewer, allowing for convenient and seamless interaction with the board. These steps ensure a smooth and efficient setup process for your Raspberry Pi.

#### **1.1.1 Installing the OS imager for raspberry pi:**

Raspberry Pi Imager is available on the official website. They are pushing this solution, so it'll be easy to find. To install the Raspberry pi Imager go to this link  
: <https://www.raspberrypi.com/software/>.

When you go to the above link then it will show the webpage like this click download for windows

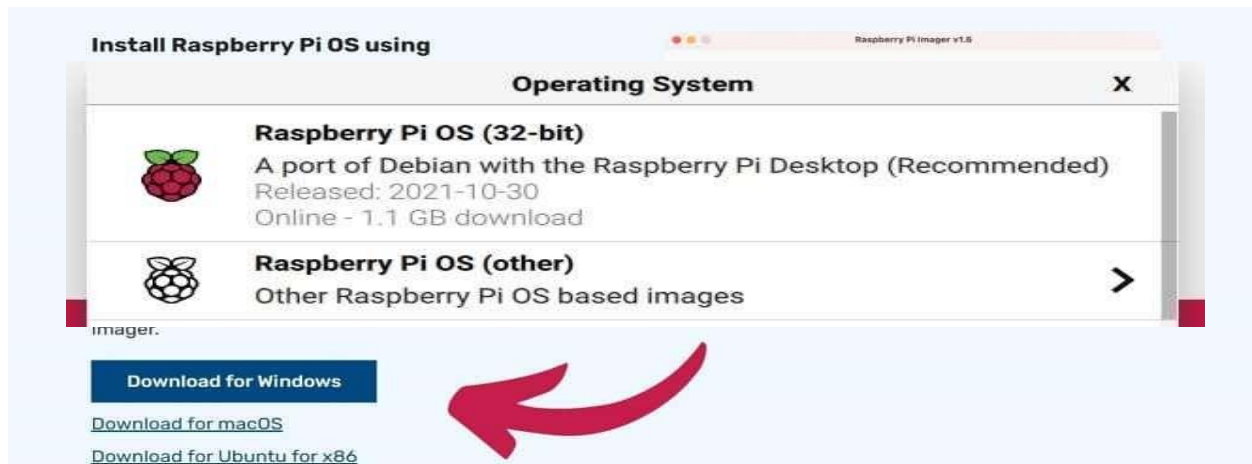


Fig 1.2: Raspberry Pi OS Image

## Install Raspberry Pi Imager on your Computer

Once downloaded on your computer, install it like any other application. For Windows users, double-click on the downloaded file and follow the instructions to install it. Keep the default values if you have a choice. The tool will then be available in the main menu (or in your applications for other operating systems). Start it and you'll be ready to move to the next step.

### 1.1.2 Flash Raspberry Pi OS on your SD card

Raspberry Pi Imager has an interface that is pretty easy to use. There are three main steps:

- Choose the operating system you want to install.
  - Choose the SD card or USB device to use.
  - Start the writing process.

Three steps to create SD card and Writing the OS in to the memory card:

- 1) Click on “Choose OS” and chose the version you want to install on your Raspberry Pi. The first option is the Desktop version, and you'll find the alternatives by clicking on the second element in the list: Raspberry Pi (other). Select RASPBERRY PI OS (64-

- BIT) compatible for OPENCV.
- 2) Once done, you can choose the SD card (in general, you'll have only one choice).
  - 3) Click on Write to start the installation:

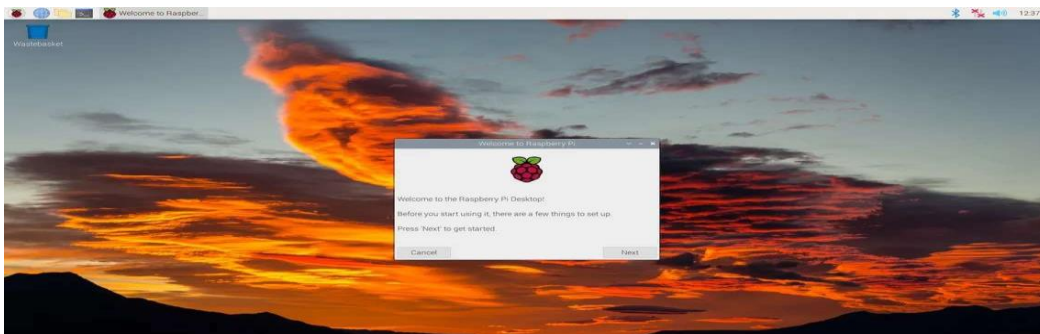


**Fig 1.3: Writing the OS**

After a few minutes SD card will be ready. If your operating system opens empty drives or format questions, you must ignore everything. Imager will do everything. You have nothing else to do. After the writing process. We can see on the picture, there will be a verifying process to ensure your SD card is not corrupted and that everything will work correctly.

### **1.1.3 First boot on Raspberry Pi OS:**

- Insert the SD card: Get your SD card and insert it into your Raspberry Pi. Then start the Raspberry Pi, with a screen and a keyboard plugged. Your device may reboot a few times, don't panic, this is normal.
- When you start on the desktop version for the first time, there is nothing to do. The system automatically logs in and introduces you to a welcome wizard:



**Fig 1.4: Raspberry pi Window**

Follow the wizard to configure the main settings :

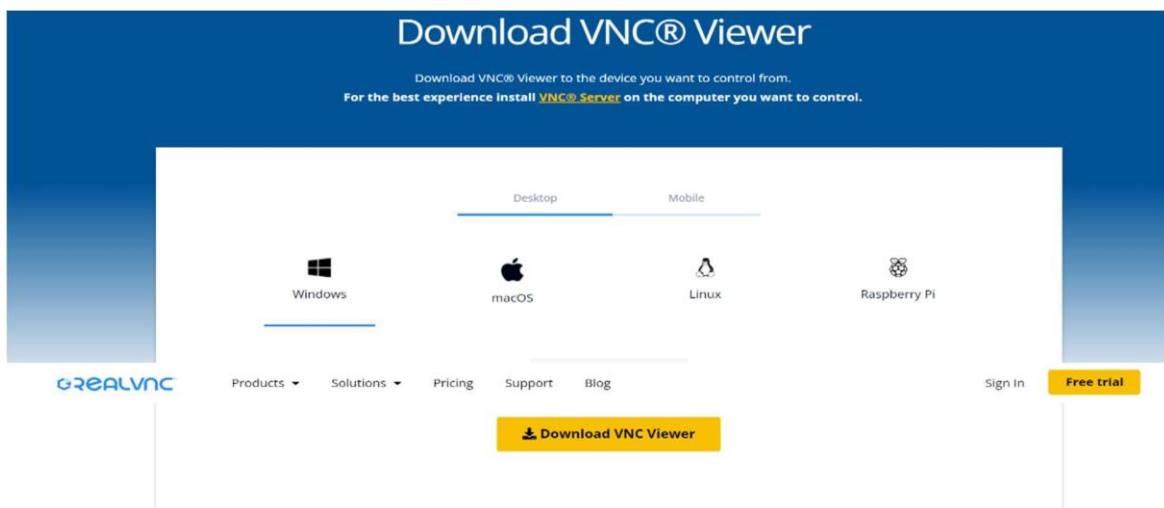
- Select your country and change the default language.
  - Change the password.
  - Connect to the Wi-Fi network if needed.
  - Start system updates.
  - Wait for the updates to finish and restart the Raspberry Pi.
- ☐ **Note:** With the latest versions of Raspberry Pi OS, the welcome wizard is now showing up before opening the session. You need to create the first user and password before anything else:
- So do remember the user name and password.

Change the Default Password on Raspberry Pi OS. It is therefore essential to change it to keep your device safe. Also, the welcome wizard has already allowed you to do it.

- Go to the main menu.
- **Go to Preferences.**
- Launch **Raspberry Pi Configuration**.
- In the System tab, click on “**Change Password**”.

### 1.1.4 INSTALLATION OF VNC VIEWER IN DESKTOP:

- To connect the raspberry pi remotely with laptop, install VNC viewer for laptop. To install it.

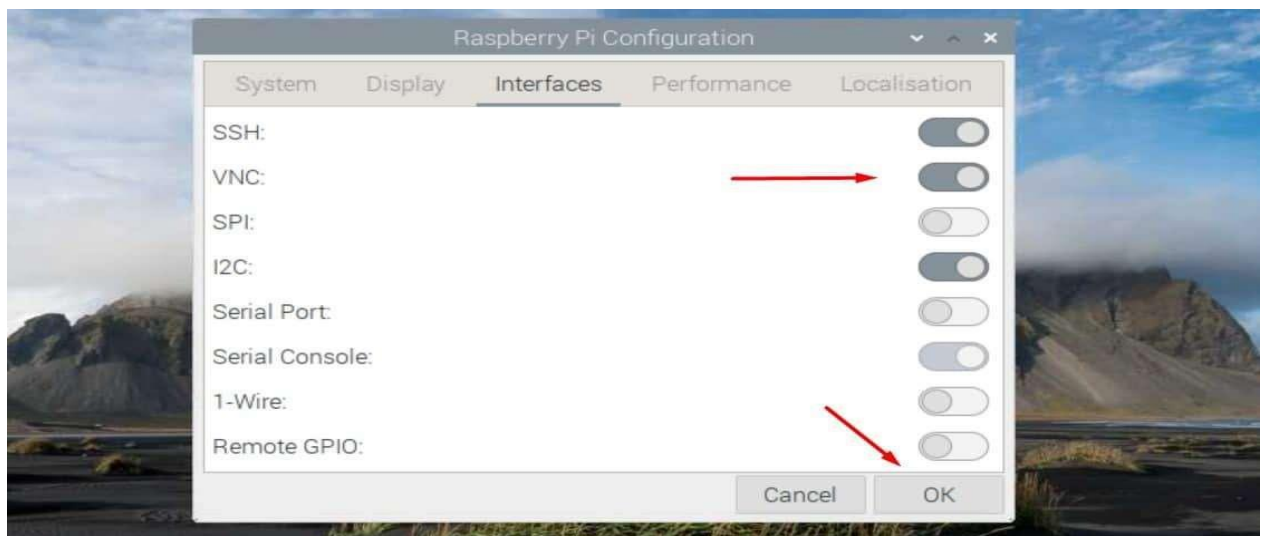


**Fig 1.5: VNC Viewer website**

- Go to the **website**  
<https://www.realvnc.com/en/connect/download/viewer/windows/> and click  
download for window and then install it in your laptop.

### 1.1.5 Connecting raspberry pi remotely to VNC viewer :

- The first step is to enable VNC on Raspberry Pi OS (via the system configuration or raspi-config), then install the client on a computer, and type the IP address of the Raspberry Pi to get connected to it.
- Enable the VNC server on your Raspberry Pi
- Depending on the operating system and version you use, enabling VNC on your Raspberry Pi might be slightly different. Let's start by making sure it's enabled on your system.
- If you have Raspberry Pi OS installed on your Pi, it will be straightforward, as VNC is pre-installed on any version so you'll have to enable it to use it (it's disabled by default, for security reasons).
- If you can get access to the desktop environment of your Raspberry Pi, here are the steps to enable VNC:
  - Open the main menu.
  - Go to Preferences > Raspberry Pi Configuration.
  - In the Interfaces tab, find the line about VNC:



**Fig 1.6: Direct Enabling of VNC**

- The interface can be slightly different depending on your version, but basically, it's just a checkbox to enable it.

- Once done, click on “OK” to apply the changes.

A few seconds later, VNC is enabled (you’ll see a VNC icon in the top-right corner) and you can move to the next part to install VNC on your computer.

- Open a session on the Raspberry Pi, and get to the command line.

You can also use the **terminal app** on the desktop if you want, even if there is no real reason to do this instead of the previous method.

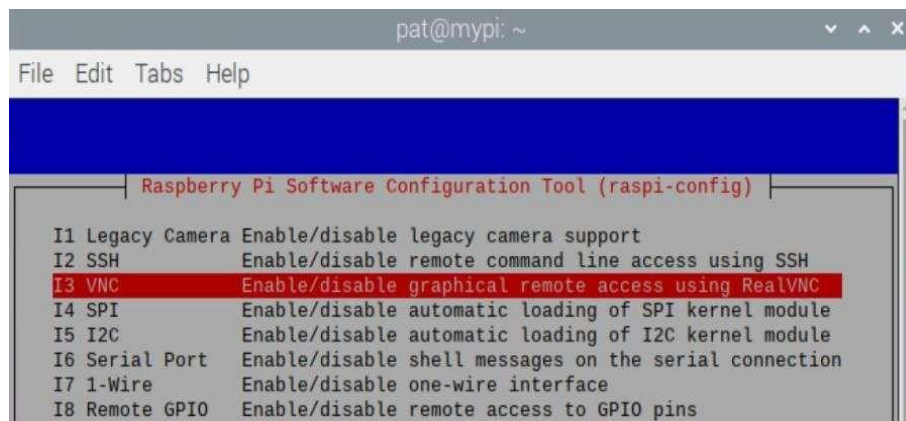
- Type the command:

**sudo raspi-config**

- Go to the Interfaces submenu and choose VNC:
- Confirm that you want to enable it.
- Quit raspi-config.
- After doing this, VNC is enabled right away, and you can start using it.

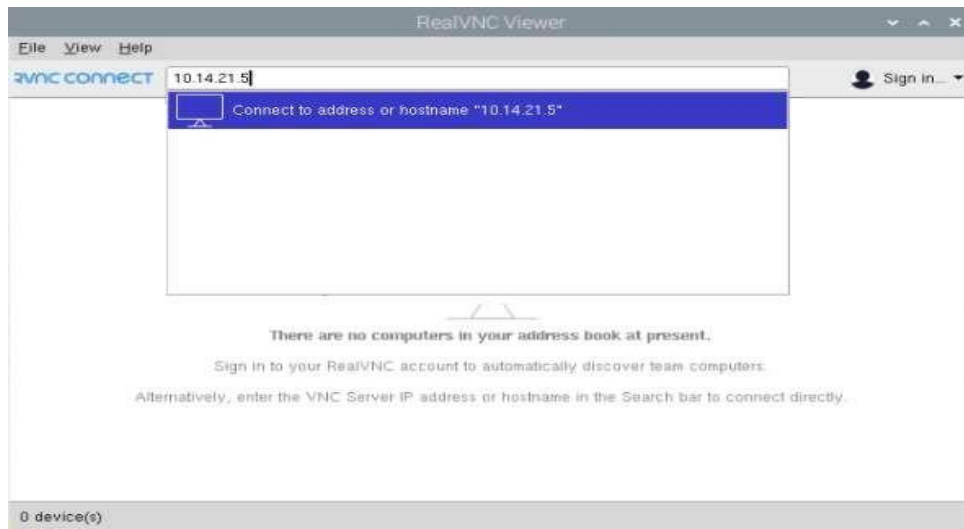
>Then on desktop a icon will be shown like this:

The IP address will be changed, if we change the wi-fi network. So ensure that you always connect raspberry pi with the same wi-fi device.



**Fig1.7: Enabling VNC through terminal**

On the device you will use to take control, run **RealVNC** Viewer and enter the IP address in the search bar and then it will show a window and enter the username and password for raspberry pi. Then the window of the raspberry pi will be start streaming in your laptop.



**Fig 1.8: VNC Viewer**

## **1.2 LIBRARIES INSTALLTION:**

### **1. OpenCV (cv2):**

- OpenCV is a library for computer vision tasks.
- You can install it using pip:

**pip install opencv-python**

### **2. Requests:**

- Requests is used for making HTTP requests.
- It's a common library, and you can install it using pip:

**pip install requests**

### **3. gspread:**

- Gspread is used for accessing Google Sheets.
- Install it using pip:

**pip install gspread**

### **4. oauth2client:**

- oauth2client is used for handling OAuth 2.0 authentication, necessary for accessing Google services.
- Install it using pip:

**pip install oauth2client**

### **5. datetime:**

- datetime is a standard Python library for working with dates and times.

- There's no need to install it separately; it's part of the Python standard library.

#### **6. smtplib:**

- smtplib is used for sending email alerts.
- It's also part of the Python standard library, so there's no need to install it separately.

#### **7. email.mime:**

- email.mime is used for creating email messages.
- It's also part of the Python standard library.

#### **8. RPi.GPIO (if using Raspberry Pi):**

- If you're running this code on a Raspberry Pi and want to use the GPIO pins, you'll need the RPi.GPIO library.
- To install it on a Raspberry Pi, you can use pip:

Copy code

**pip install RPi.GPIO**

#### **9. Face Cascade XML Files (haarcascades):**

- The code uses Haar Cascade classifiers for face and eye detection. Make sure you have the XML files for these classifiers.
- You can download the required XML files from the OpenCV GitHub repository or other reliable sources.

To enable the camera and GPIO pins on a Raspberry Pi, you'll need to follow specific setup procedures. Here's a brief overview:

##### **Enabling the Camera:**

1. Ensure your Raspberry Pi is powered off.
2. Connect the camera module to the camera port on the Raspberry Pi.
3. Power on the Raspberry Pi.
4. Open the Raspberry Pi Configuration tool using the command:

**sudo raspi-config**

5. Navigate to "Interfacing Options" and select "Camera."
6. Enable the camera module and follow the prompts to save the changes.
7. Reboot your Raspberry Pi.

##### **Enabling GPIO Pins:**

1. If not already installed, install the RPi.GPIO library as mentioned above.
2. Ensure your code has appropriate GPIO pin setup and usage, as you have already done in your provided code.



3. Make sure your Raspberry Pi is running the Raspbian OS or a compatible distribution. Please note that enabling and using the GPIO pins on a Raspberry Pi may require superuser privileges (using **sudo**). Be careful when working with hardware components to avoid damaging them or the Raspberry Pi.

## CHAPTER-2

### HARDWARE AND SOFTWARE DESCRIPTION

#### 2.1 Hardware description:

**1. Raspberry Pi 3B+:** The Raspberry Pi 3B+ is a compact, versatile single-board computer (SBC) renowned for its exceptional capabilities. Powered by a quad-core ARM Cortex-A53 processor running at up to 1.4 GHz and equipped with 1 GB of LPDDR2 SDRAM, it handles diverse computing tasks effectively. Integrated dual-band 2.4 GHz and 5 GHz Wi-Fi, along with Bluetooth 4.2/BLE support, facilitates wireless connectivity and IoT applications. Its connectivity options include four USB 2.0 ports, HDMI for video and audio output, a 40-pin GPIO header, and Gigabit Ethernet. Featuring a CSI camera connector and DSI display connector, it excels in visual applications. Despite lacking built-in storage, it relies on microSD cards and supports multiple operating systems, including Raspberry Pi OS. The compact, credit-card-sized form factor ensures portability and ease of integration, making it a favored choice for education, DIY electronics, robotics, and more.



**Fig 2.1: Raspberry Pi 3b+**

**2. 32 GB Memory Card:** The 32 GB memory card is a small, removable storage device commonly used with single-board computers like the Raspberry Pi. It stores the operating system, software, and data required for the computer to function. In the context of a Raspberry Pi, it typically contains the Raspberry Pi OS (or another compatible operating system) and serves as the primary storage medium for the system.



**Fig 2.2: 32 GB memory card**

**3. Ultrasonic Sensor:** An ultrasonic sensor is a hardware component that uses ultrasonic sound waves to measure distances. It typically consists of a transmitter that sends out ultrasonic pulses and a receiver that listens for the echoes. By measuring the time it takes for the sound waves to bounce back, the sensor can calculate the distance to an object. Ultrasonic sensors are commonly used in robotics, automation, and various DIY projects for tasks such as obstacle avoidance, distance measurement, and object detection.



**Fig 2.3: Ultrasonic sensor**

**4. Raspberry Pi Camera Module :** The Raspberry Pi Camera Module is a small and lightweight camera accessory designed specifically for Raspberry Pi boards. It allows the Raspberry Pi to capture high-quality still images and video. There are different versions of the camera module available, including the standard Camera Module and the Camera Module v2. These modules are widely used for photography, video recording, surveillance, computer vision, and other visual applications in Raspberry Pi projects.



**Fig 2.4: Raspberry pi 3 camera module**

**5. Buzzer:** A buzzer is an electromechanical component that produces sound or an audible signal when an electric current is applied to it. Buzzer modules are often used in electronics projects for generating alerts, notifications, or alarms. They can emit different types of sounds, such as beeps or tones, depending on the specific application.



**Fig 2.5: Buzzer**

## **2.2 SOFTWARE DESCRIPTION:**

### **1. OpenCV (cv2):**

- Description: OpenCV, or Open Source Computer Vision Library, is a powerful open-source computer vision and machine learning software library. It provides a wide range of tools and functions for image and video analysis, object detection, and machine learning tasks.
- Usage in Code: In the code, OpenCV (cv2) is used for face and eye detection within the camera feed.

### **2. Requests:**

- Description: The requests library is a popular Python library for making HTTP requests. It allows you to send HTTP GET and POST requests, making it useful for interacting with web services and APIs.
- Usage in Code: Requests is used to send email alerts through SMTP by connecting to a mail server.

### **3. gspread:**

- Description: gspread is a Python library for accessing Google Sheets, allowing you to interact with Google Sheets documents programmatically. It provides methods to read, write, and manipulate Google Sheets data.
- Usage in Code: In the code, gspread is used to interact with a Google Sheets document named 'kidswatch' to record data related to children's TV-watching duration.

#### **4. oauth2client:**

- Description: The oauth2client library is used for handling OAuth 2.0 authentication, which is required for securely accessing Google services and APIs.
- Usage in Code: oauth2client is used to authorize access to Google Sheets, enabling the code to interact with the specified Google Sheets document.

#### **5. datetime:**

- Description: The datetime module is a standard Python library used for working with dates and times. It provides classes and functions to manipulate and format date and time data.
- Usage in Code: In the code, datetime is used for timestamping and formatting time-related information.

#### **6. smtplib:**

- Description: The smtplib module is part of the Python standard library and is used for sending email messages using the Simple Mail Transfer Protocol (SMTP).
- Usage in Code: smtplib is used to send email alerts when certain conditions are met, such as exceeding the allowed TV-watching time.

#### **7. email.mime:**

- Description: The email.mime module is also part of the Python standard library and is used for creating email messages with MIME (Multipurpose Internet Mail Extensions) content types.
- Usage in Code: It is used in combination with smtplib to format and send email alerts.

#### **8. RPi.GPIO (if using Raspberry Pi):**

- Description: RPi.GPIO is a Python library for Raspberry Pi GPIO (General Purpose Input/Output) pins. It provides functions to control and interact with hardware components connected to the GPIO pins.
- Usage in Code: If you are running the code on a Raspberry Pi, RPi.GPIO is used to control GPIO pins, which can be utilized for hardware interactions, such as triggering a buzzer.

#### **9. Face Cascade XML Files (haarcascades):**

- Description: These XML files contain pre-trained Haar Cascade classifiers used for object detection tasks, such as face and eye detection. They are a part of OpenCV's data distribution.
- Usage in Code: The Haar Cascade XML files are used in OpenCV for face and eye detection within the camera feed. They help identify specific facial features.

#### **10. Camera Module (if using Raspberry Pi):**

- Description: The camera module is a hardware component that can be connected to a Raspberry Pi. It allows the Raspberry Pi to capture images and video, making it useful for computer vision and surveillance applications.
- Usage in Code: If you're using a Raspberry Pi, the camera module is used to capture the video feed for face and eye detection.

## **CHAPTER-3**

### **IMPLEMENTATION**

#### **3.1 SOFTWARE IMPLEMENTATION:**

In the software implementation we divided the project into five parts:

- Face detection and Eye detection
- Distance detection
- Mail alert
- Accessing google spread sheets
- Time based actions( calculating duration)

##### **3.1.1 Face detection and Eye detection:**

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not.

We are going to use Haar Cascade. A Haar Cascade is basically a classifier which is used to detect the object for which it has been trained for, from the source.

The methodology of face detection can be applied to landmark (e.g., eyes, nose tip, and mouth) localization, which can then be utilized to face geometrical normalization.

Environment Setup required :

1. Python — OpenCV, numpy
2. Insert haarcascade\_eye.xml & haarcascade\_frontalface\_default.xml files in the same folder.

Features:

Recognize and locate facial features: Get the coordinates of the eyes, ears, cheeks, nose, and mouth of every face detected.

Get the contours of facial features: Get the contours of detected faces and their eyes, eyebrows, lips, and nose.

Recognize facial expressions: Determine whether a person is smiling or has their eyes closed.

Track faces across video frames: Get an identifier for each individual person's face that is detected. This identifier is consistent across invocations, so you can, for example, perform image manipulation on a particular person in a video stream.

Process video frames in real-time: Face detection is performed on the device, and is fast enough to be used in real-time applications, such as video manipulation.

Algorithm:

The **haar-like algorithm** is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

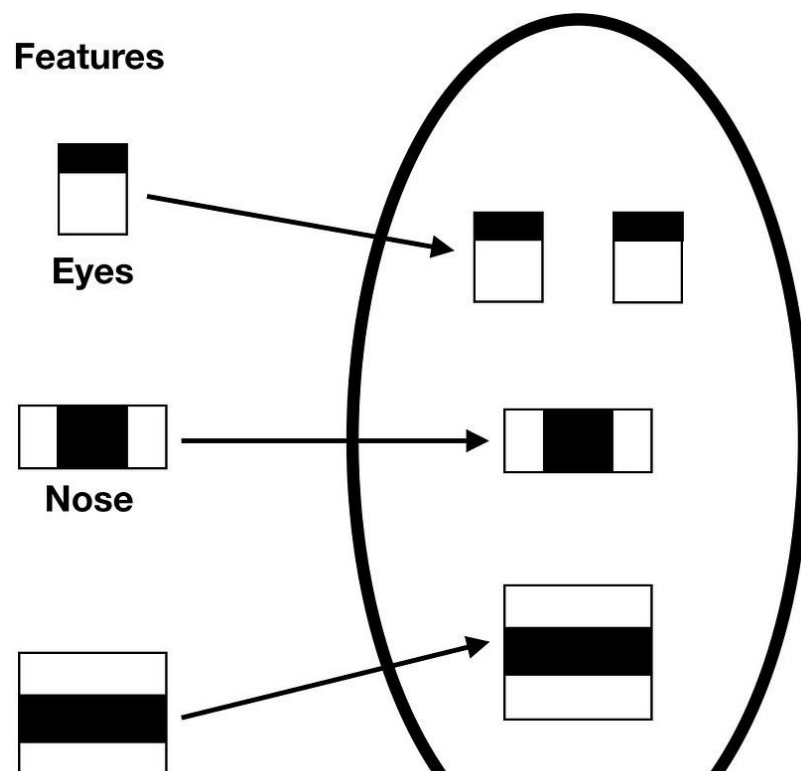


Fig 3.1: Theoretical face model



```

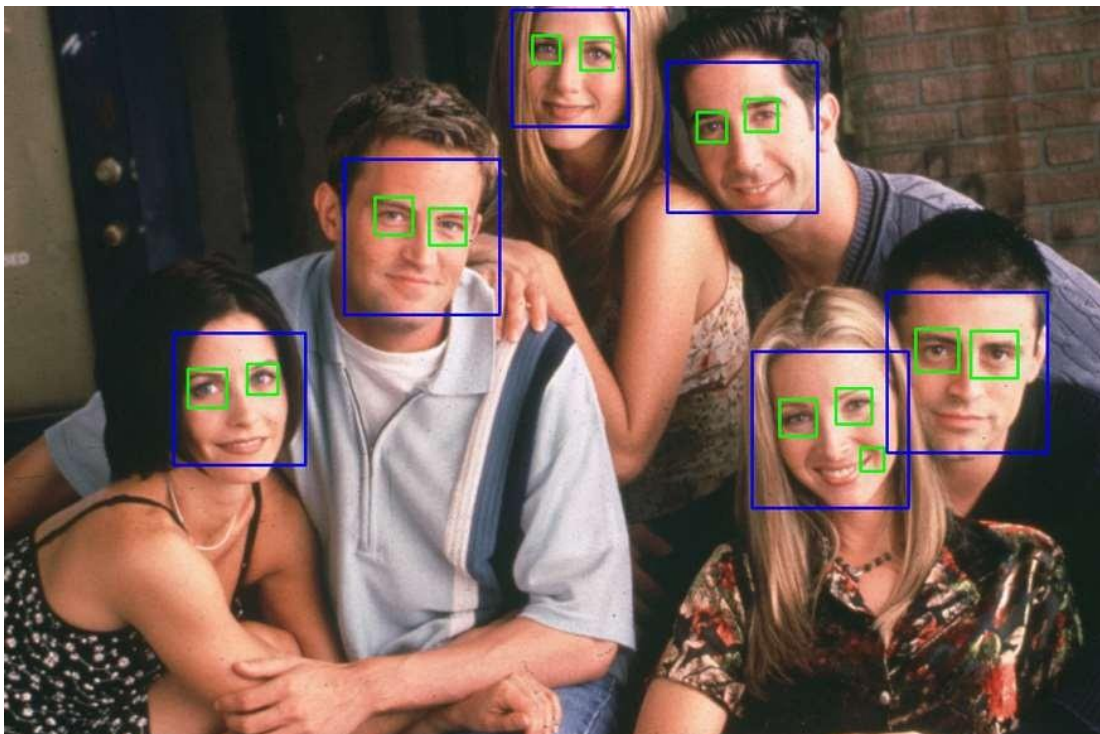
import numpy as np
import cv2

face_cascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")#save the image(i) in the same
directoryimg = cv2.imread("friends.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)faces =
face_cascade.detectMultiScale(gray, 1.3, 5)for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)

for (ex,ey,ew,eh) in eyes:
    cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv2.imshow('img',img)cv2.waitKey(0)
cv2.destroyAllWindows()

```

Output :



**Fig 3.2: Face Detection**

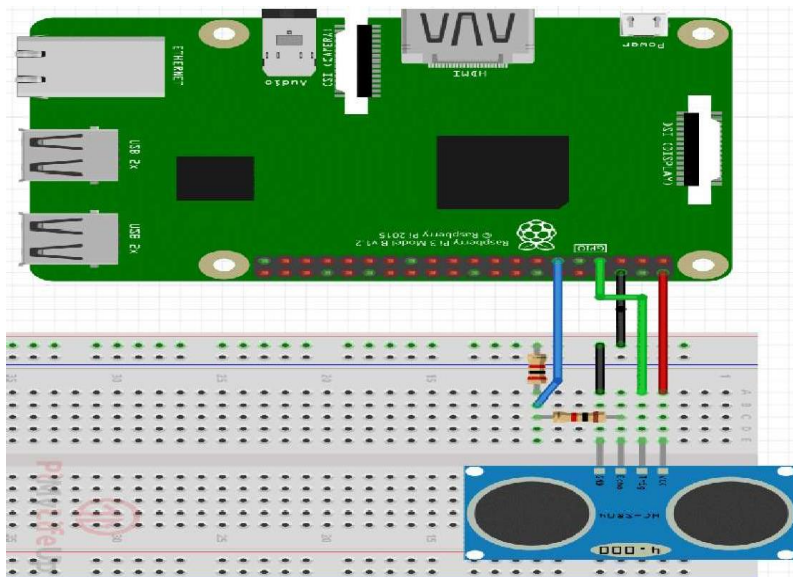
There is as such no restriction to the number of faces this algorithm can detect. The code can detect faces, but it would still require verification from the user. This is therefore not a fully intelligent system since it requires interaction from the user. With this algorithm we can determine whether someone is in front of camera or not, and If present their eyes opened or not.

### 3.1.2 DISTANCE DETECTION:(ULTRASONIC SENSOR)

The reason for adding a voltage divider is to drop the voltage going to the GPIO pins down to **3.3v** from **5v**. In this guide, we will be utilizing a **1k  $\Omega$**  resistor and a **2k  $\Omega$**  resistor to achieve this. If you want to understand how we calculate the values of a voltage divider, you can check out the guide featured at the end of this book.

Follow the guides below to see how to wire your HC-SR04 distance sensor to your Raspberry Pi.

- VCC Connects to Pin 2 (5v)
- Trig Connects to Pin 7 (GPIO 4)
- Echo Connects to R1 (1k  $\Omega$ )
- R2 (2k  $\Omega$ ) Connects from R1 to Ground



**Fig 3.3: Interfacing ultrasonic with raspi**

To construct the circuit just do the following.

Don't forget you can use the diagram above to give yourself an idea of what you need to do.

1. Run a wire from the **ground pin (Pin 6)** to the **ground/negative rail** on the **breadboard**.
2. Run a wire from 5v **pin (Pin 2)** to the **VCC** pin on the **HC-SR04** distance sensor.
3. Run a wire from **pin 7** to the **TRIG** pin on the distance sensor.
4. Run a **1k  $\Omega$**  resistor from the **ECHO** pin on the distance sensor to a spot on the breadboard.
5. Run a **2k  $\Omega$**  resistor from the **1k  $\Omega$**  resistor to the **ground/negative rail** on the Raspberry Pi.
6. Run a wire from between the **1k  $\Omega$**  resistor and the **2k  $\Omega$**  resistor to **pin 11** on the Raspberry Pi.

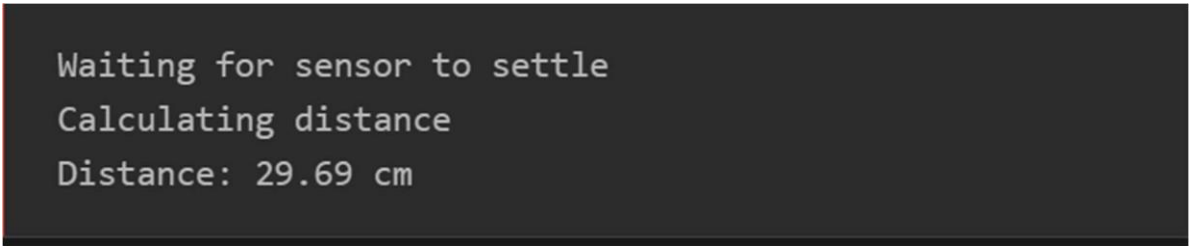
Wire from R1 and R2 connects to Pin 11 and GND connects to Pin 6 (Ground)

#### **CODE:**

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
try:
    GPIO.setmode(GPIO.BOARD)
    PIN_TRIGGER = 7
    PIN_ECHO = 11
    GPIO.setup(PIN_TRIGGER, GPIO.OUT)
    GPIO.setup(PIN_ECHO, GPIO.IN)
    GPIO.output(PIN_TRIGGER, GPIO.LOW)
    print "Waiting for sensor to settle"
    time.sleep(2)
    print "Calculating distance"
    GPIO.output(PIN_TRIGGER, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(PIN_TRIGGER, GPIO.LOW)
    while GPIO.input(PIN_ECHO)==0:
        pulse_start_time = time.time()
```

```
while GPIO.input(PIN_ECHO)==1:  
    pulse_end_time = time.time()  
pulse_duration = pulse_end_time - pulse_start_time  
distance = round(pulse_duration * 17150, 2)  
print "Distance:",distance,"cm"
```

**OUTPUT:**



```
Waiting for sensor to settle  
Calculating distance  
Distance: 29.69 cm
```

**Fig 3.4: Ultrasonic sensor output**

### **3.1.3 Mail alert:**

To send an image in an email using Python and SMTP (Simple Mail Transfer Protocol), you can use the `smtplib` library for sending emails and the `email` library for composing the email with attachments. Here's a step-by-step guide on how to do it:

#### **Import the required libraries:**

```
import smtplib

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.base import MIMEBase
from email import encoders
```

#### **Set up your email and SMTP server details:**

```
smtp_server = 'smtp.example.com' # Replace with your SMTP server
smtp_port = 587 # Replace with your SMTP port (587 is common for TLS)
smtp_username = 'your_email@example.com' # Replace with your email address
smtp_password = 'your_password' # Replace with your email password
sender_email = 'your_email@example.com' # Replace with your email address
receiver_email = 'recipient@example.com' # Replace with the recipient's email address
```

#### **Create a MIMEMultipart object to compose the email:**

```
msg = MIMEMultipart()
msg['From'] = sender_email
msg['To'] = receiver_email
msg['Subject'] = 'Image Attachment'
```

#### **Attach the image to the email:**

```
with open('image.jpg', 'rb') as img_file: # Replace 'image.jpg' with your image file name
    img = MIMEImage(img_file.read(), name='image.jpg')
msg.attach(img)
```

### **Connect to the SMTP server and send the email:**

try:

```
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls() # Use TLS (Transport Layer Security) encryption
server.login(smtp_username, smtp_password)
server.sendmail(sender_email, receiver_email, msg.as_string())
print('Email with image sent successfully!')
```

except Exception as e:

```
print('Email could not be sent. Error:', str(e))
```

finally:

```
server.quit()
```

Make sure to replace the placeholders with your actual email and server details. Also, replace 'image.jpg' with the path to the image file you want to attach to the email. This code sends an email with the specified image as an attachment to the recipient's email address.

### **3.1.4 ACCESSING GOOGLE SPREAD SHEETS:**

#### **1. Set Up a Google Cloud Project:**

- Go to the Google Cloud Console.
- Create a new project or use an existing one.
- Enable the Google Sheets API for your project:
  - Click on the "APIs & Services" > "Library" in the left sidebar.
  - Search for "Google Sheets API" and enable it.
- Create credentials for your project:
  - Click on "APIs & Services" > "Credentials."
  - Click on "Create credentials" and select "Service Account Key."
  - Follow the setup wizard to create a service account. You'll download a JSON file with your credentials. Keep this file safe.

#### **2. Install the Required Python Libraries:**

You'll need the `google-auth`, `google-auth-oauthlib`, `google-auth-httplib2`, and `google-api-python-client` libraries to work with the Google Sheets API. You can install them using `pip`:

```
pip install --upgrade google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client
```

### 3. Authorize Your Application:

You need to authorize your Python application to access your Google Sheets. To do this, use the credentials JSON file you downloaded in step 1.

```
from google.oauth2 import service_account credentials =
service_account.Credentials.from_service_account_file( 'your-credentials.json',
scopes=['https://www.googleapis.com/auth/spreadsheets'])
```

### 4. Access and Modify Google Sheets:

Now that you have the necessary credentials, you can access and modify Google Sheets. Here's an example of how to do this:

```
from googleapiclient.discovery import build

# Create a Google Sheets API client

service = build('sheets', 'v4', credentials=credentials)

# Define the spreadsheet ID and range

spreadsheet_id = 'your-spreadsheet-id'

range_name = 'Sheet1!A1:B2'

# Example: Read values from a range

result = service.spreadsheets().values().get(spreadsheetId=spreadsheet_id,
range=range_name).execute()

values = result.get('values', [])

if not values:

    print('No data found.')

else:
```

```

print('Data:')

for row in values:

    print(row)

# Example: Write values to a range

values = [

    ["Value1", "Value2"],

    ["Value3", "Value4"]

]

body = {

    'values': values

}

result = service.spreadsheets().values().update(spreadsheetId=spreadsheet_id,
range=range_name, valueInputOption="RAW", body=body).execute()

print(f'Updated {result.get("updatedCells")} cells')

```

After writing your Python script, execute it. Ensure you have the necessary permissions for the Google Sheets you want to access.

Remember to replace 'your-credentials.json' with the path to your credentials JSON file and 'your-spreadsheet-id' with the ID of the Google Sheet you want to work with. Also, make sure that the Google Sheets API is enabled for your Google Cloud project.

**3.1.5 Time calculation:** If we are succeed with all the above software things, then by combining all this we can build a child monitoring system. By giving some conditions and using time library we can actually calculate the time. By the time library we can only calculate the duration but can build a timer with it. So we implanted a logic based on the iteration speed of a while loop, we incremented the value accordingly. In raspberry pi board, the while taking approximately 0.5 seconds to complete all the lines in our code, based on the iteration speed it can be adjusted. So finally we can calculate the Duration of kids watching tv and we set timer which is used for distance warning from all this software implementations.



### 3.2 HARDWARE IMPLEMENTATION:

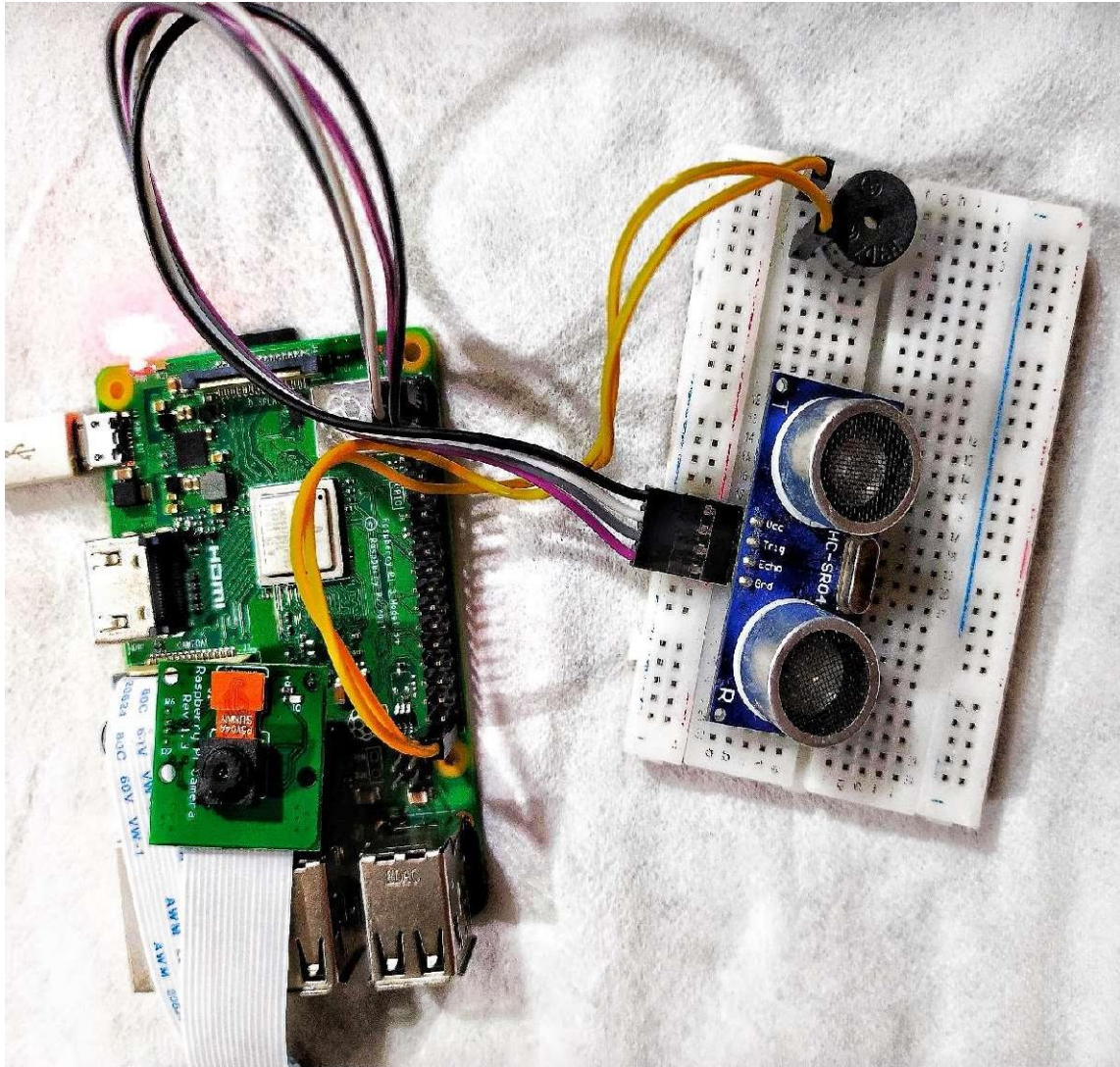


Fig 3.5: Raspi camera interface with raspi board

Pi Model B+	
3V3 Power	1
GPIO2 (BCM120)	2
GPIO3 (BCM120)	3
GPIO4 (BCM120)	4
Ground	5
GPIO17	6
GPIO27	7
GPIO22	8
3V3 Power	9
GPIO10 (SPI0_MISO)	10
GPIO9 (SPI0_MISO)	11
GPIO11 (SPI0_SCLK)	12
Ground	13
ID_SD (SD IO CS0/CS1)	14
GPIO5	15
GPIO6	16
GPIO13	17
GPIO18	18
GPIO26	19
Ground	20
GPIO14 (UART5_TXD)	21
GPIO15 (UART5_RXD)	22
GPIO18 (PCB_CLK)	23
Ground	24
ID_SC (SD IO CS0/CS1)	25
GPIO5	26
GPIO6	27
GPIO13	28
GPIO18	29
GPIO26	30
Ground	31
GPIO14 (UART5_TXD)	32
GPIO15 (UART5_RXD)	33
GPIO18 (PCB_CLK)	34
Ground	35
ID_SD (SD IO CS0/CS1)	36
GPIO5	37
GPIO6	38
GPIO13	39
GPIO18	40
GPIO26	41
Ground	42
ID_SC (SD IO CS0/CS1)	43
GPIO5	44
GPIO6	45
GPIO13	46
GPIO18	47
GPIO26	48
Ground	49
ID_SD (SD IO CS0/CS1)	50
GPIO5	51
GPIO6	52
GPIO13	53
GPIO18	54
GPIO26	55
Ground	56
ID_SC (SD IO CS0/CS1)	57
GPIO5	58
GPIO6	59
GPIO13	60
GPIO18	61
GPIO26	62
Ground	63
ID_SD (SD IO CS0/CS1)	64
GPIO5	65
GPIO6	66
GPIO13	67
GPIO18	68
GPIO26	69
Ground	70
ID_SC (SD IO CS0/CS1)	71
GPIO5	72
GPIO6	73
GPIO13	74
GPIO18	75
GPIO26	76
Ground	77
ID_SD (SD IO CS0/CS1)	78
GPIO5	79
GPIO6	80
GPIO13	81
GPIO18	82
GPIO26	83
Ground	84
ID_SC (SD IO CS0/CS1)	85
GPIO5	86
GPIO6	87
GPIO13	88
GPIO18	89
GPIO26	90
Ground	91
ID_SD (SD IO CS0/CS1)	92
GPIO5	93
GPIO6	94
GPIO13	95
GPIO18	96
GPIO26	97
Ground	98
ID_SC (SD IO CS0/CS1)	99
GPIO5	100
GPIO6	101
GPIO13	102
GPIO18	103
GPIO26	104
Ground	105
ID_SD (SD IO CS0/CS1)	106
GPIO5	107
GPIO6	108
GPIO13	109
GPIO18	110
GPIO26	111
Ground	112
ID_SC (SD IO CS0/CS1)	113
GPIO5	114
GPIO6	115
GPIO13	116
GPIO18	117
GPIO26	118
Ground	119
ID_SD (SD IO CS0/CS1)	120
GPIO5	121
GPIO6	122
GPIO13	123
GPIO18	124
GPIO26	125
Ground	126
ID_SC (SD IO CS0/CS1)	127
GPIO5	128
GPIO6	129
GPIO13	130
GPIO18	131
GPIO26	132
Ground	133
ID_SD (SD IO CS0/CS1)	134
GPIO5	135
GPIO6	136
GPIO13	137
GPIO18	138
GPIO26	139
Ground	140
ID_SC (SD IO CS0/CS1)	141
GPIO5	142
GPIO6	143
GPIO13	144
GPIO18	145
GPIO26	146
Ground	147
ID_SD (SD IO CS0/CS1)	148
GPIO5	149
GPIO6	150
GPIO13	151
GPIO18	152
GPIO26	153
Ground	154
ID_SC (SD IO CS0/CS1)	155
GPIO5	156
GPIO6	157
GPIO13	158
GPIO18	159
GPIO26	160
Ground	161
ID_SD (SD IO CS0/CS1)	162
GPIO5	163
GPIO6	164
GPIO13	165
GPIO18	166
GPIO26	167
Ground	168
ID_SC (SD IO CS0/CS1)	169
GPIO5	170
GPIO6	171
GPIO13	172
GPIO18	173
GPIO26	174
Ground	175
ID_SD (SD IO CS0/CS1)	176
GPIO5	177
GPIO6	178
GPIO13	179
GPIO18	180
GPIO26	181
Ground	182
ID_SC (SD IO CS0/CS1)	183
GPIO5	184
GPIO6	185
GPIO13	186
GPIO18	187
GPIO26	188
Ground	189
ID_SD (SD IO CS0/CS1)	190
GPIO5	191
GPIO6	192
GPIO13	193
GPIO18	194
GPIO26	195
Ground	196
ID_SC (SD IO CS0/CS1)	197
GPIO5	198
GPIO6	199
GPIO13	200
GPIO18	201
GPIO26	202
Ground	203
ID_SD (SD IO CS0/CS1)	204
GPIO5	205
GPIO6	206
GPIO13	207
GPIO18	208
GPIO26	209
Ground	210
ID_SC (SD IO CS0/CS1)	211
GPIO5	212
GPIO6	213
GPIO13	214
GPIO18	215
GPIO26	216
Ground	217
ID_SD (SD IO CS0/CS1)	218
GPIO5	219
GPIO6	220
GPIO13	221
GPIO18	222
GPIO26	223
Ground	224
ID_SC (SD IO CS0/CS1)	225
GPIO5	226
GPIO6	227
GPIO13	228
GPIO18	229
GPIO26	230
Ground	231
ID_SD (SD IO CS0/CS1)	232
GPIO5	233
GPIO6	234
GPIO13	235
GPIO18	236
GPIO26	237
Ground	238
ID_SC (SD IO CS0/CS1)	239
GPIO5	240
GPIO6	241
GPIO13	242
GPIO18	243
GPIO26	244
Ground	245
ID_SD (SD IO CS0/CS1)	246
GPIO5	247
GPIO6	248
GPIO13	249
GPIO18	250
GPIO26	251
Ground	252
ID_SC (SD IO CS0/CS1)	253
GPIO5	254
GPIO6	255
GPIO13	256
GPIO18	257
GPIO26	258
Ground	259
ID_SD (SD IO CS0/CS1)	260
GPIO5	261
GPIO6	262
GPIO13	263
GPIO18	264
GPIO26	265
Ground	266
ID_SC (SD IO CS0/CS1)	267
GPIO5	268
GPIO6	269
GPIO13	270
GPIO18	271
GPIO26	272
Ground	273
ID_SD (SD IO CS0/CS1)	274
GPIO5	275
GPIO6	276
GPIO13	277
GPIO18	278
GPIO26	279
Ground	280
ID_SC (SD IO CS0/CS1)	281
GPIO5	282
GPIO6	283
GPIO13	284
GPIO18	285
GPIO26	286
Ground	287
ID_SD (SD IO CS0/CS1)	288
GPIO5	289
GPIO6	290
GPIO13	291
GPIO18	292
GPIO26	293
Ground	294
ID_SC (SD IO CS0/CS1)	295
GPIO5	296
GPIO6	297
GPIO13	298
GPIO18	299
GPIO26	300
Ground	301
ID_SD (SD IO CS0/CS1)	302
GPIO5	303
GPIO6	304
GPIO13	305
GPIO18	306
GPIO26	307
Ground	308
ID_SC (SD IO CS0/CS1)	309
GPIO5	310
GPIO6	311
GPIO13	312
GPIO18	313
GPIO26	314
Ground	315
ID_SD (SD IO CS0/CS1)	316
GPIO5	317
GPIO6	318
GPIO13	319
GPIO18	320
GPIO26	321
Ground	322
ID_SC (SD IO CS0/CS1)	323
GPIO5	324
GPIO6	325
GPIO13	326
GPIO18	327
GPIO26	328
Ground	329
ID_SD (SD IO CS0/CS1)	330
GPIO5	331
GPIO6	332
GPIO13	333
GPIO18	334
GPIO26	335
Ground	336
ID_SC (SD IO CS0/CS1)	337
GPIO5	338
GPIO6	339
GPIO13	340
GPIO18	341
GPIO26	342
Ground	343
ID_SD (SD IO CS0/CS1)	344
GPIO5	345
GPIO6	346
GPIO13	347
GPIO18	348
GPIO26	349
Ground	350
ID_SC (SD IO CS0/CS1)	351
GPIO5	352
GPIO6	353
GPIO13	354
GPIO18	355
GPIO26	356
Ground	357
ID_SD (SD IO CS0/CS1)	358
GPIO5	359
GPIO6	360
GPIO13	361
GPIO18	362
GPIO26	363
Ground	364
ID_SC (SD IO CS0/CS1)	365
GPIO5	366
GPIO6	367
GPIO13	368
GPIO18	369
GPIO26	370
Ground	371
ID_SD (SD IO CS0/CS1)	372
GPIO5	373
GPIO6	374
GPIO13	375
GPIO18	376
GPIO26	377
Ground	378
ID_SC (SD IO CS0/CS1)	379
GPIO5	380
GPIO6	381
GPIO13	382
GPIO18	383
GPIO26	384
Ground	385
ID_SD (SD IO CS0/CS1)	386
GPIO5	387
GPIO6	388
GPIO13	389
GPIO18	390
GPIO26	391
Ground	392
ID_SC (SD IO CS0/CS1)	393
GPIO5	394
GPIO6	395
GPIO13	396
GPIO18	397
GPIO26	398
Ground	399
ID_SD (SD IO CS0/CS1)	400
GPIO5	401
GPIO6	402
GPIO13	403
GPIO18	404
GPIO26	405
Ground	406
ID_SC (SD IO CS0/CS1)	407
GPIO5	408
GPIO6	409
GPIO13	410
GPIO18	411
GPIO26	412
Ground	413
ID_SD (SD IO CS0/CS1)	414
GPIO5	415
GPIO6	416
GPIO13	417
GPIO18	418
GPIO26	419
Ground	420
ID_SC (SD IO CS0/CS1)	421
GPIO5	422
GPIO6	423
GPIO13	424
GPIO18	425
GPIO26	426
Ground	427
ID_SD (SD IO CS0/CS1)	428
GPIO5	429
GPIO6	430
GPIO13	431
GPIO18	432
GPIO26	433
Ground	434
ID_SC (SD IO CS0/CS1)	435
GPIO5	436
GPIO6	437
GPIO13	438
GPIO18	439
GPIO26	440
Ground	441
ID_SD (SD IO CS0/CS1)	442
GPIO5	443
GPIO6	444
GPIO13	445
GPIO18	446
GPIO26	447
Ground	448
ID_SC (SD IO CS0/CS1)	449
GPIO5	450
GPIO6	451
GPIO13	452
GPIO18	453
GPIO26	454
Ground	455
ID_SD (SD IO CS0/CS1)	456
GPIO5	457
GPIO6	458
GPIO13	459
GPIO18	460
GPIO26	461
Ground	462
ID_SC (SD IO CS0/CS1)	463
GPIO5	464
GPIO6	465
GPIO13	466
GPIO18	467
GPIO26	468
Ground	469
ID_SD (SD IO CS0/CS1)	470
GPIO5	471
GPIO6	472
GPIO13	473
GPIO18	474
GPIO26	475
Ground	476
ID_SC (SD IO CS0/CS1)	477
GPIO5	478
GPIO6	479
GPIO13	480
GPIO18	481
GPIO26	482
Ground	483
ID_SD (SD IO CS0/CS1)	484
GPIO5	485
GPIO6	486
GPIO13	487
GPIO18	488
GPIO26	489
Ground	490
ID_SC (SD IO CS0/CS1)	491
GPIO5	492
GPIO6	493
GPIO13	494
GPIO18	495
GPIO26	496
Ground	497
ID_SD (SD IO CS0/CS1)	498
GPIO5	499
GPIO6	500
GPIO13	501
GPIO18	502
GPIO26	503
Ground	504
ID_SC (SD IO CS0/CS1)	505
GPIO5	506
GPIO6	507
GPIO13	508
GPIO18	509
GPIO26	510
Ground	511
ID_SD (SD IO CS0/CS1)	512
GPIO5	513
GPIO6	514
GPIO13	515
GPIO18	516
GPIO26	517
Ground	518
ID_SC (SD IO CS0/CS1)	519
GPIO5	520
GPIO6	521
GPIO13	522
GPIO18	523
GPIO26	524
Ground	525
ID_SD (SD IO CS0/CS1)	526
GPIO5	527
GPIO6	528
GPIO13	529
GPIO18	530
GPIO26	531
Ground	532
ID_SC (SD IO CS0/CS1)	533
GPIO5	534
GPIO6	535
GPIO13	536
GPIO18	537
GPIO26	538
Ground	539
ID_SD (SD IO CS0/CS1)	540
GPIO5	541
GPIO6	542
GPIO13	543
GPIO18	544
GPIO26	545
Ground	546
ID_SC (SD IO CS0/CS1)	547
GPIO5	548
GPIO6	549

Buzzer Positive (Longer Leg) -> GPIO Pin (e.g., GPIO21 on Pin 40)

Buzzer Negative (Shorter Leg) -> GND (Raspberry Pi)



**Fig 3.7: Hardware setup**

## CHAPTER-4

### WORKING AND RESULT

#### 4.1 WORKING:

1. Main Monitoring Function - kidswatch:

- The kidswatch function is the main monitoring process.
- It uses a loop to continuously capture frames from the webcam and analyze them.
- It detects faces and eyes in the frames using OpenCV's cascade classifiers.
- If eyes are detected as open, it increments the time counter (t) by 0.5 seconds.
- If eyes are closed for 10 or more seconds ( $t \geq 10$ ), it increments the warning count (warn\_count) and potentially activates a buzzer (if hardware is set up, but commented out).
- If either the duration exceeds 30 seconds or the warning count exceeds 4, it breaks out of the loop.

2. Monitoring Loop and Alerts:

- The script continues to monitor if the duration is less than 30 seconds and the warning count is less than 4.
- If either condition is met (duration  $\geq 30$  seconds or warning count  $\geq 4$ ), the script sends an email alert indicating either that the duration limit was reached or that the warning limit was exceeded.

3. Calculating and Formatting Duration:

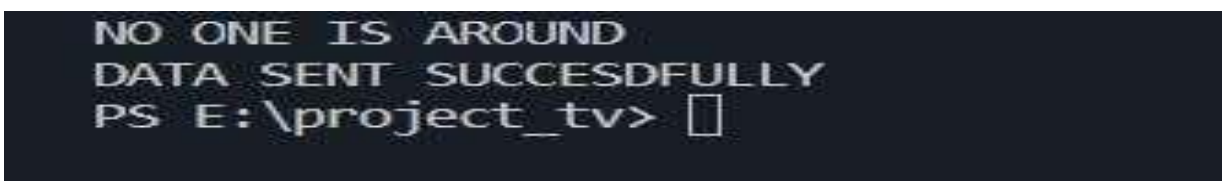
- The script calculates and formats the duration based on hours, minutes, and seconds.

4. Google Sheets Data Logging:

- The script accesses Google Sheets using the Google Sheets API and inserts a new row with data, including the date, day of the week, **start time, end time, and duration, and and also the reasons why tv turned off.**

#### 4.2 RESULT :

First condition: If no one is watching the tv. In command prompt and the data is sent to google sheets:



Second condition: if the kid is watching in a safer distance then the tv will be off when duration completed:

```
duration: 30
Email alert sent successfully
Duration Completed
Data sent successfully
```

Third condition: If the kid watching tv but crossed the safe distance and watch the tv overall 15 seconds for two times an email alert will be send to the parent with an image in it:

```
Waiting for sensor to settle
Calculating distance
Distance: 29.69 cm
```

The email alert will be like this:

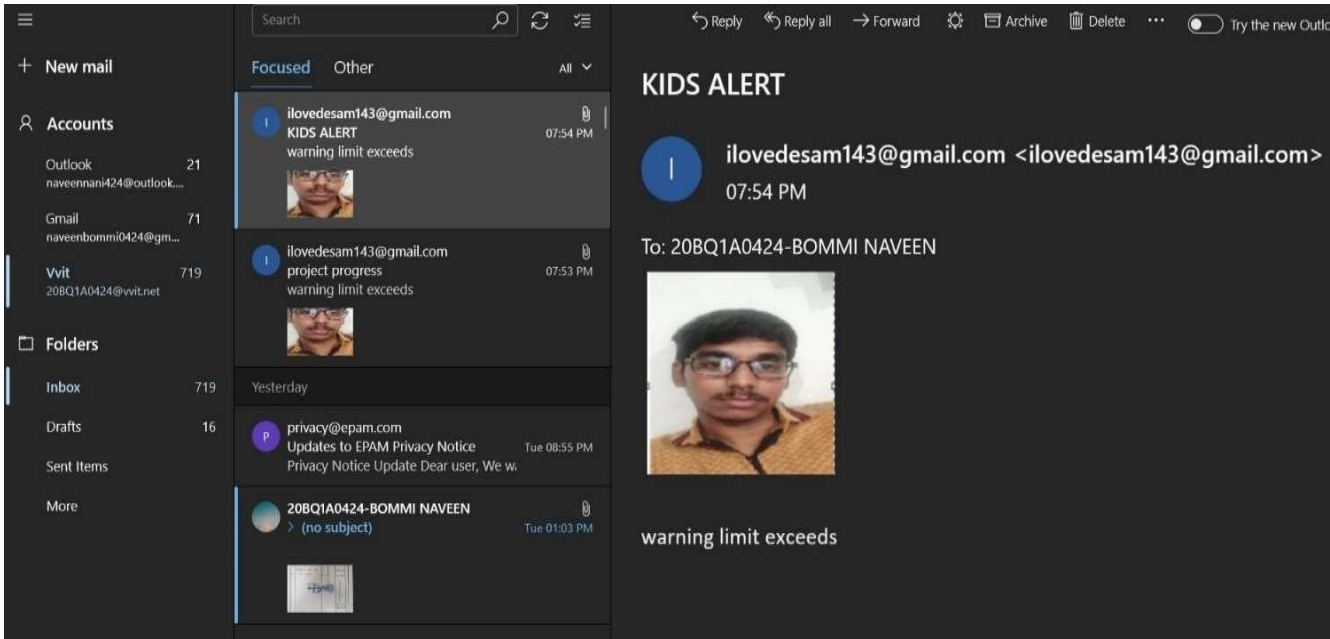


Fig 4.1:mail Alert



In the google spread sheets:

	A	B	C	D	E	F	G	H	I	J	K
1	DATE	DAY	STARTING TIME	ENDING TIME	DURATION	STATUS					
2	2023-09-23	Saturday	09:28:45 PM	09:29:04 PM	18sec	CROSSED THE DISTANCE LIMIT					
3	2023-09-23	Saturday	09:33:43 PM	09:33:49 PM	5sec	NO ONE IS AROUND					
4	2023-09-23	Saturday	09:36:39 PM	09:37:08 PM	28sec	NO ONE IS AROUND					
5	2023-09-23	Saturday	09:37:53 PM	09:38:24 PM	31sec	DURATION COMPLETED					
6	2023-09-23	Saturday	09:38:34 PM	09:38:43 PM	8sec	NO ONE IS AROUND					
7	2023-09-23	Saturday	09:38:51 PM	09:38:57 PM	6sec	NO ONE IS AROUND					
8	2023-09-23	Saturday	09:40:29 PM	09:40:56 PM	26sec	CROSSED THE DISTANCE LIMIT					
9	2023-09-23	Saturday	09:42:07 PM	09:42:35 PM	27sec	CROSSED THE DISTANCE LIMIT					
10	2023-09-23	Saturday	09:52:06 PM	09:52:25 PM	19sec	CROSSED THE DISTANCE LIMIT					
11	2023-09-23	Saturday	09:59:04 PM	09:59:37 PM	33sec	CROSSED THE DISTANCE LIMIT					
12	2023-09-23	Saturday	10:01:59 PM	10:02:21 PM	22sec	CROSSED THE DISTANCE LIMIT					
13	2023-09-23	Saturday	10:03:18 PM	10:03:54 PM	36sec	CROSSED THE DISTANCE LIMIT					
14	2023-09-23	Saturday	10:05:12 PM	10:05:40 PM	28sec	CROSSED THE DISTANCE LIMIT					
15	2023-09-23	Saturday	10:07:30 PM	10:07:36 PM	6sec	NO ONE IS AROUND					
16	2023-09-23	Saturday	10:18:54 PM	10:18:59 PM	4sec	NO ONE IS AROUND					
17	2023-09-25	Monday	08:43:54 AM	08:43:58 AM	3sec	NO ONE IS AROUND					
18	2023-09-25	Monday	08:44:33 AM	08:44:58 AM	25sec	CROSSED THE DISTANCE LIMIT					
19	2023-09-25	Monday	08:50:41 AM	08:51:07 AM	26sec	NO ONE IS AROUND					
20	2023-09-25	Monday	08:52:56 AM	08:53:05 AM	8sec	NO ONE IS AROUND					
21	2023-09-25	Monday	08:55:32 AM	08:56:03 AM	31sec	DURATION COMPLETED					
22	2023-10-01	Sunday	03:13:02 PM	03:13:05 PM	3sec	NO ONE IS AROUND					
23	2023-10-01	Sunday	03:15:46 PM	03:15:51 PM	5sec	NO ONE IS AROUND					
24	2023-10-01	Sunday	03:16:05 PM	03:16:24 PM	19sec	NO ONE IS AROUND					
25	2023-10-01	Sunday	03:17:00 PM	03:17:21 PM	20sec	NO ONE IS AROUND					

**Fig 4.2: google spread sheet data**

## **CHAPTER-5**

### **CONCLUSION**

This system utilizes computer vision techniques and hardware components like cameras to assess whether a child's eyes are open or closed while watching TV. It measures the duration of time the child's eyes remain closed and provides warnings if the child's eyes remain closed for an extended period, with the ability to send email alerts as notifications to the parents or caregivers. The system also logs important data about TV viewing sessions, such as start and end times, the duration of viewing, and the day of the week, which can be recorded in a Google Sheets document for later analysis. Furthermore, this code demonstrates the integration of various technologies, including OpenCV for image processing, Google Sheets API for data storage, and email alerts for timely notifications, all aimed at enhancing child safety during screen time.

Overall, this system represents a proactive approach to child safety and well-being in the digital age. By continuously monitoring and analyzing a child's eye activity during TV viewing, it empowers parents and caregivers with valuable insights into their children's screen time habits. This information can help in promoting responsible screen time and ensuring that children take regular breaks to protect their eye health. Additionally, the integration of email alerts ensures that parents can be promptly informed of any concerning situations, further enhancing their ability to provide a safe and nurturing environment for their children's development.

#### **FUTURE SCOPE:**

**Machine Learning Integration:** Extending our project to monitor employees in the office for the purpose of ensuring efficiency demands a thorough consideration of ethical and legal implications, prioritizing employee privacy, and promoting transparency by making small changes in the code. Allow parents to set custom alert thresholds based on their preferences. This could include specifying the maximum allowable screen time or distance from the TV.

Develop a mobile app or web interface that allows parents to remotely monitor their child's screen time and receive real-time alerts. Provide options to pause or limit screen time remotely and implement behaviour analytics to detect changes in a child's screen time habits over time. Recognize trends or patterns that could indicate excessive screen time or other issues.

## REFERENCES

- [1] <https://docs.opencv.org/4.x/index.html>
- [2] <https://docs.python.org/3/>
- [3] <https://developers.google.com/sheets/api/guides/concepts>
- [4] <https://developers.google.com/identity/protocols/oauth2>
- [5] <https://realpython.com/python-send-email/>
- [6] [https://docs.opencv.org/4.x/da/d5b/tutorial\\_py\\_video\\_display.html](https://docs.opencv.org/4.x/da/d5b/tutorial_py_video_display.html)
- [7] <https://docs.python.org/3/library/time.html>
- [8] [https://docs.opencv.org/4.x/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html)
- [9] <https://www.raspberrypi.com/documentation//>

## APPENDIX

```
import cv2
import time
import requests
import gspread
from oauth2client.service_account import ServiceAccountCredentials
import datetime
import RPi.GPIO as GPIO
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
import os
try:
    # Set GPIO mode to BCM
    GPIO.setmode(GPIO.BCM)

    # Define GPIO pins for Trig, Echo, and Buzzer
    trig_pin = 17
    echo_pin = 18
    buzzer_pin = 21
    led_pin = 11

    GPIO.setwarnings(False)

    # Set up the GPIO pins
    GPIO.setup(trig_pin, GPIO.OUT)
    GPIO.setup(echo_pin, GPIO.IN)
    GPIO.setup(buzzer_pin, GPIO.OUT)
    GPIO.setup(led_pin, GPIO.OUT)

    # Email configuration
```



```

smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = 'ilovedesam143@gmail.com'
smtp_password = 'zkhjanzkensnbjbj'
receiver_email = '20bq1a0424@vvit.net'

# Google Sheets API authentication

warn_count = 0

# Other initializations
sst = time.time()

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')

distance = 0
cap = cv2.VideoCapture(0)

def send_email_alert(subject, message, image_path=None):
    try:
        # Set up the SMTP server
        server = smtplib.SMTP(smtp_server, smtp_port)
        server.starttls()
        server.login(smtp_username, smtp_password)

        # Create a MIMEText message
        msg = MIMEMultipart()
        msg['From'] = smtp_username
        msg['To'] = receiver_email
        msg['Subject'] = subject
        msg.attach(MIMEText(message, 'plain'))

```

```

# If an image path is provided, attach the image to the email
if image_path:
    with open(image_path, 'rb') as image_file:
        image = MIMEImage(image_file.read(), name='captured_image.jpg')
        msg.attach(image)

# Send the email
server.sendmail(smtp_username, receiver_email, msg.as_string())

# Close the SMTP server
server.quit()
print("Email alert sent successfully")

# Delete the image file after sending
if image_path and os.path.exists(image_path):
    os.remove(image_path)
    print("Image deleted successfully")
cv2.destroyAllWindows()
except Exception as e:
    print("Email alert failed to send:", str(e))

def get_distance():
    try:
        # Trigger the sensor by sending a 10us pulse on the Trig pin
        GPIO.output(trig_pin, True)
        time.sleep(0.00001)
        GPIO.output(trig_pin, False)

        pulse_start_time = time.time()
        pulse_end_time = time.time()

        # Wait for the Echo pin to go HIGH

```

```

while GPIO.input(echo_pin) == 0:
    pulse_start_time = time.time()

    # Wait for the Echo pin to go LOW
    while GPIO.input(echo_pin) == 1:
        pulse_end_time = time.time()

    # Calculate the pulse duration
    pulse_duration = pulse_end_time - pulse_start_time

    # Calculate distance (in centimeters)
    distance = (pulse_duration * 34300) / 2
    return distance

except Exception as e:
    print("Error in get_distance:", str(e))
    return 0

def kidswatch():
    try:
        t = 0
        no_face = 0
        GPIO.output(led_pin, GPIO.HIGH)
        print("hi")
        global warn_count
        global status
        while True:
            ret, frame = cap.read()

            gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.3,
minNeighbors=2)

            distance = int(get_distance())
            print("distance", distance)

```

```

if len(faces) == 0:
    no_face += 0.5
    print("no faces detected", no_face)
else:
    no_face = 0

for (x, y, w, h) in faces:
    roi_gray = gray_frame[y:y + h, x:x + w]
    eyes = eye_cascade.detectMultiScale(roi_gray)

    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(frame, (x + ex, y + ey), (x + ex + ew, y + ey + eh), (0,
255, 0), 2)

    if len(eyes) >= 1 and distance < 24:
        cv2.putText(frame, "Eyes Open", (x, y - 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
        t += 0.5
        print("time", t)
    else:
        cv2.putText(frame, "Eyes Closed", (x, y - 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    if t >= 10:
        warn_count += 1
        print("warning distance")

        GPIO.output(buzzer_pin, GPIO.HIGH)
        time.sleep(2)
        GPIO.output(buzzer_pin, GPIO.LOW)
        break

elif no_face >= 10:

```

```

        print("NO ONE IS AROUND")
        GPIO.output(buzzer_pin, GPIO.HIGH)
        time.sleep(1)
        GPIO.output(buzzer_pin, GPIO.LOW)
        status = "NO ONE IS AROUND"
        return True
    else:
        eft = time.time()
        duration = int(eft - sst)
        if duration < 30 and warn_count < 2:
            pass
        else:
            break
    cv2.imshow('Kidswatch', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    eft = time.time()
    duration = int(eft - sst)
    print("duration:", duration)

    if duration < 30 and warn_count < 2:
        GPIO.output(led_pin, GPIO.LOW)
        return kidswatch()
    elif duration < 30 and warn_count >= 2:
        message = "Warning limit exceeded in Kidswatch!"
        ret, frame = cap.read()
        image_path = 'captured_image.jpg'
        cv2.imwrite(image_path, frame)
        send_email_alert("Kidswatch Alert", message, image_path)
        print("ALL THE WARNINGS ARE COMPLETED")
        GPIO.output(led_pin, GPIO.LOW)
        GPIO.output(buzzer_pin, GPIO.HIGH)

```

```
        time.sleep(1)
    except Exception as e:
        print("An error occurred:", str(e))
        print("Error type:", type(e))
    finally:
        # Release the camera and close OpenCV windows, cleanup GPIO pins, and handle
        any other necessary cleanup tasks.
        cap.release()
        cv2.destroyAllWindows()
        GPIO.output(led_pin, GPIO.LOW)
```