Project for Bachelor of Science in Electrical and Electronic Engineering

Centralized Monitoring System for Covid ICU at CGH

by

Md. Hedayetul Islam Chy.

ID NO: 17702013

Department of Electrical and Electronic Engineering (EEE)

UNIVERSITY OF CHITTAGONG

Chittagong-4331, Bangladesh

March, 2022

Centralized Monitoring System for Covid ICU at CGH

by

Md. Hedayetul Islam Chy.

ID NO: 17702013

A project submitted for the partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering (EEE)
UNIVERSITY OF CHITTAGONG
Chittagong-4331, Bangladesh

Centralized Monitoring System for Covid ICU at CGH

Supervised by

Dr. Mohammed Arif Iftakher Mahmood

Associate Professor

Department of Electrical & Electronic Engineering

University of Chittagong

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering (EEE)

UNIVERSITY OF CHITTAGONG

Chittagong-4331, Bangladesh

APPROVAL OF SUBMISSION

The project titled **“Centralized Monitoring system for Covid ICU at CGH”,** is submitted by **Md. Hedayetul Islam Chy.** ID No **17702013** session **2016-17** has been carried out under my supervision as satisfactory in partial fulfillment of the requirement for the degree of BACHELOR OF SCIENCE IN ELECTRICAL & ELECTRONIC ENGINEERING on **27/03/2022**

Supervisor

Department of Electrical and Electronic Engineering (EEE)
UNIVERSITY OF CHITTAGONG
Chittagong-4331, Bangladesh

CANDIDATE'S DECLARATION

I, **Md. Hedayetul Islam Chy.** ID No. **17702013** declare that this project titled, **"Centralized Monitoring System for Covid ICU at CGH"** and the works presented in it are my original works. I hereby certify that:

- The works presented in this project are done mainly while I am studying for a B. Sc. Engineering at this University.
- The contents of this project are based on my own original works (e.g., published/unpublished).
- I am the lead author of all my published/unpublished works, from which materials are used to compose this project.
- The published/unpublished works of my own and the published work of others are clearly cited and/or discussed, wherever it is needed.
- Where I have quoted from the work of others, the source is always given.
- All main sources of help are clearly acknowledged.

Signature of the candidate with date

Md. Hedayetul Islam Chy.

Author Name

ACKNOWLEDGEMENTS

First of all, praises be to the Almighty Allah, whose blessing has enabled us to complete our project successfully.

My heartiest gratitude and deep appreciation goes to my project supervisor Dr. Md. Arif Iftakher Mahmood, Associate Professor, Department of Electrical and Electronic Engineering, University of Chittagong; for his untiring guidance, valuable suggestions, corrective advice, and assistance in accomplishing the work as well as manuscript writing. We consider ourselves very fortunate to be given an opportunity to work with them.

I also thank all the teachers of our department for their valuable opinions. Finally, I must thank all my closest friends and family members who support us throughout the way and gave me all the chances to get where I am right now.

ABSTRACT

Patients who have severe disease and injuries are shifted into ICU. Many patients in the ICU are so infected that doctors and nurses cannot approach them whenever they want. Normally all important data of patients that doctors and nurses need to observe are shown on the bedside monitor. ICU bedside monitor, which seems like a tv screen or computer display, provides ICU staff with some necessary information of patients in real-time such as heart rate and rhythms, body temperature, oxygen saturation(spo2), blood pressure, and many others. According to our system, using the Esp-32 camera, by capturing the image of the bedside monitor and sending it to the server through WIFI, doctors or nurses, or specialists can able to see the bedside monitor images to a webpage both locally and cloudly. Again using a computer vision tool, extracting data from the bedside monitor image and alert to the webpage if any patient's health condition isn't good.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ICU	Intensive Care Unit
HR	Heart rate
SpO_2	Oxygen Saturation
CLAHE	Contrast Limited Adaptive Histogram Equalization
GUI	Graphical User Interface
LAN	Local Area Network
PPE	Personal Protective Equipment
NIBP	Non-invasive Blood Pressure
LPF	Low Pass Filter

CHAPTER 1

1. INTRODUCTION

ICU is the place where a hospital provides its highest facilities. Mainly severe stages patients are shifted into this unit so that hospital can provide their best. Many patients in ICU are so infected that doctors and nurses cannot approach them whenever they need. Unlike traditional ICU, in covid ICU, one cannot enter without protective equipment. So, hospital personnel needs more protective equipment and it creates a deficit. Normally all-important data of patients that hospital personnel need to observe are shown on the bedside monitor. This monitor provides necessary information of patients in real-time such as heart rate and rhythms, body temperature, oxygen saturation, blood pressure, and many others. Hospital staffs need to check this consecutively. But in this infectious covid-19 situation, an inspection of the patient is barely possible.

1.1 PROBLEM DESCRIPTION

Chittagong General Hospital didn’t have an ICU to start with. With the emergence of Covid19, a 10 beds COVID ICU was set up in a couple of weeks. The monitor that uses for monitoring or showing data is Nihon kohden BSM-3562. However, the ICU monitors (Nihon kohden BSM-3562) didn’t come with central monitoring software and the licensing fee of ~17,000 USD was way over their budget. This feature was essential as they were trying to reduce virus exposure of the hospital personnel who otherwise had to frequently get inside the ICU to record the patients’ status.
In this covid-19 period, doctors and hospital personnel need to maintain many different protocols. Healthcare workers in the covid ICU need more protection. They must wear Gowns, Gloves, N95respirators and simple masks, etc. There are some limitations according to time and patient number. According to Kaiser Data [1]. hospital personnel must change their:

For ICU,

- Gloves: 2 gloves for each of 170 changes per patient per day
- Gowns: 20 changes per patient per day
- Simple masks: 10 changes per patient per day
- N95 respirators: 6 changes per patient per day

For Non-ICU,

- Gloves: 2 gloves for each of 80 changes per patient per day
- Gowns: 20 changes per patient per day
- Simple masks: 10 changes per patient per day
- N95 respirators: 2-6 changes per patient per day

World health organizations also add some rules for covid fighters. Hence, health workers need to follow these rules strictly. Hence, the Lack of PPE and other protective systems made the situation worse.

1.2 OBJECTIVE

- The main objective is to reduce virus exposure of the hospital personnel who otherwise had to frequently get inside the ICU to record the patient's status.
- To develop an IP-based camera system inside CGH covid-ICU unit.
- To show the bedside monitor image to hospital personnel in non-covid areas.
- Provide automated detection of heart rate and oxygen saturation data from the bedside monitor image.
- Creating an alarming system if the HR and SPO2 data are in an abnormal condition.
- Creating an app infrastructure system that helps doctors investigate patient's condition

CHAPTER 2

2. LITERATURE REVIEW

2.1 COVID ICU

ICU is the specialist hospital ward that provides treatment and monitoring for very ill people. This ward is staffed with professional healthcare facilities and has sophisticated monitoring equipment. There will be some rules and regulations for the hospital staff of ICU. The rule inside Covid ICU is stricter. Hospital staff strictly maintain those.

To continuously monitor, a bedside monitor hospital personnel use the bedside monitor. There are many bedside monitors available in the market. In Chittagong General Hospital aka CGH covid, ICU word uses the bedside monitor brand named NIHON KOHDEN. The model of the monitor is BSM-3562.

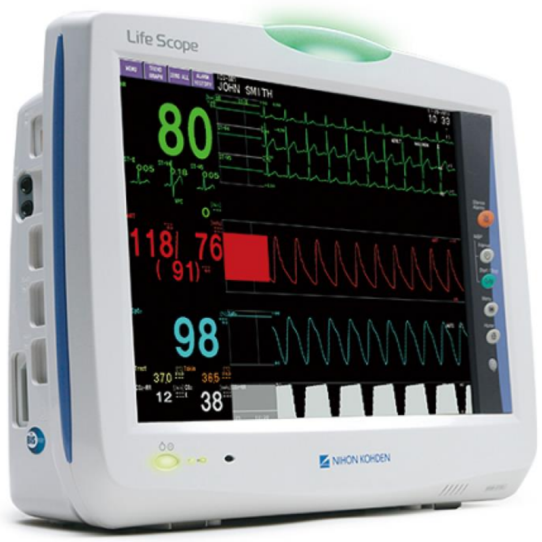


Figure 2-1	Bedside monitor Nihon kohden (BSM-3562) [2]

2.1.1 BSM-3562 monitor

Various data of the patients are shown in the monitor (BSM-3562). There is an IP-based system to centralized monitor the data. The basic parameters shown in the display are 3,6,10 lead ECG, respiration, SpO2, NIBP (non-invasive blood pressure), and dual temperature with dedicated cable and connector. To centralized this system there is an IP-based solution. This Ip-based solution must be bought from the company. In case this centralized is so expensive that almost 17k USD need for 10 displays.

2.2 IMAGE PROCESSING

It is a technique for performing operations on an image in order to get or extract relevant information. Digital image processing is the use of computer algorithms to perform image processing on digital images in computer science. Digital image processing, as a subsection or discipline of digital signal processing, has a number of advantages over analog image processing. It enables a considerably broader range of algorithms to be applied to the input data, as well as the avoidance of issues like noise accumulation and signal distortion during processing. Digital image processing may be described as multidimensional systems since images are defined in two dimensions (or more). [3].
Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods that would be impossible by analog means.

Digital image processing is a concrete application of, and a practical technology based on:

- Classification
- Feature extraction
- Multi-scale signal analysis
- Pattern recognition
- Projection
- Edge detection [3]

2.2.1 OpenCV

OpenCV (Open-Source Computer Vision Library) is a free and open-source software library for computer vision and machine learning. OpenCV was created to provide a common infrastructure for computer vision applications and to help commercial goods incorporate machine perception more quickly. OpenCV is a BSD-licensed product, making it simple for businesses to use and change the code. [4] [5].

The library has in excess of 2500 upgraded calculations, which incorporates an exhaustive arrangement of both work of art and cutting-edge PC vision and AI calculations. These calculations can be utilized to distinguish and perceive faces, recognize objects, characterize human activities in recordings, track camera developments, track moving articles, separate 3D models of items, produce 3D point mists from sound system cameras, join pictures together to create a high-goal picture of a whole scene, observe comparable pictures from a picture information base, eliminate red eyes from pictures taken utilizing streak, follow eye developments, perceive landscape and lay out markers to overlay it with increased reality, and so on. OpenCV has more than 47 thousand people of user communities and an estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups, and governmental bodies [4] [5].

- **Open source**: OpenCV is open source and released under the BSD 3-Clause License. It is free for commercial use.
- **Optimized:** OpenCV is a highly optimized library with a focus on real-time applications.
- **Cross-Platform**: C++, Python, and Java interfaces support Linux, macOS, Windows, iOS, and Android.

2.2.1.1 Canny edge detection:

The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a *computational theory of edge* *detection* explaining why the technique works. The Process of the Canny edge detection algorithm can be broken down into five different steps:

1. Smooth the image with a Gaussian filter to remove the noise.
2. Locate the image's intensity gradients.
3. Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection
4. Determine probable edges using a twofold threshold.
5. Hysteresis edge tracking: Finish the edge identification by suppressing all other edges that are weak and not coupled to strong edges. [6] [7].

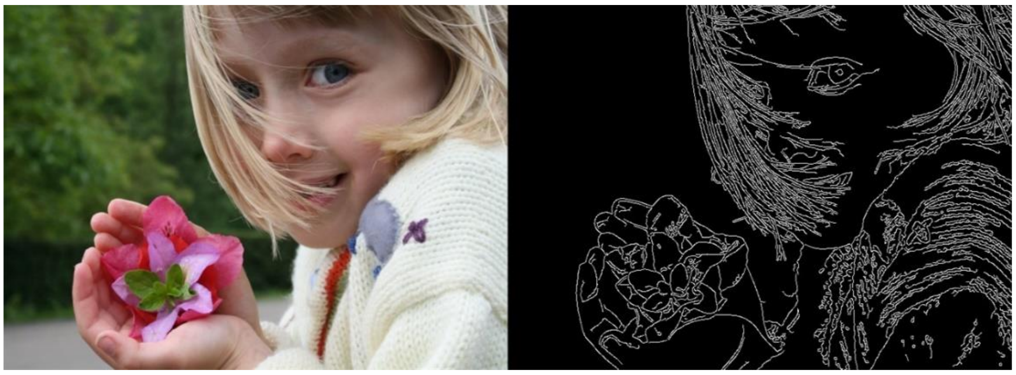


Figure 2-2 Canny edge detection (a) Original Image (b) Canny-edged image [8]

2.2.1.2 Image Thresholding:

a) Simple Thresholding:

Here, the matter is straightforward. If the pixel value is greater than a threshold value, it is assigned one value (maybe white), else it is assigned another value (maybe black). The function used is **cv2.threshold**. The first argument is the source image, which **should be grayscale**. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maximal which represents the value to be given if the pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function. Different types are: [8] [9]

- cv2.THRESH_BINARY

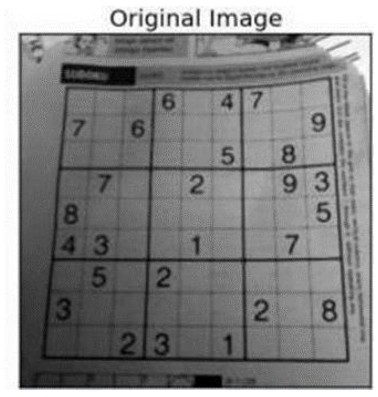
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

b) Adaptive Thresholding

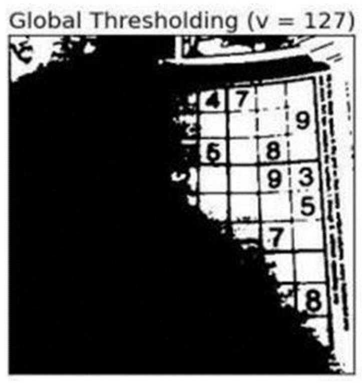
In Simple Thresholding, we used a global value as the threshold value. But it may not be good in all the conditions where the image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. In this, the algorithm calculates the threshold for a small region of the image. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination [9].

c) Otsu's Binarization

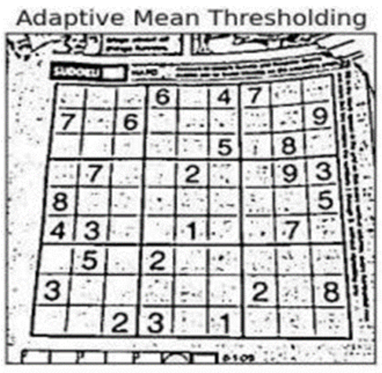
In global thresholding, we used an arbitrary value for the threshold value. By trial and error method, we can know whether the threshold value is good or not. But consider a **bimodal image** (In simple words, a bimodal image is an image whose histogram has two peaks). For that image, we can approximately take a value in the middle of those peaks as the threshold value. That is what Otsu binarization does. So in simple words,



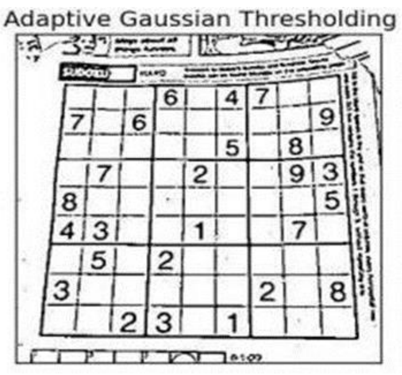
(a)



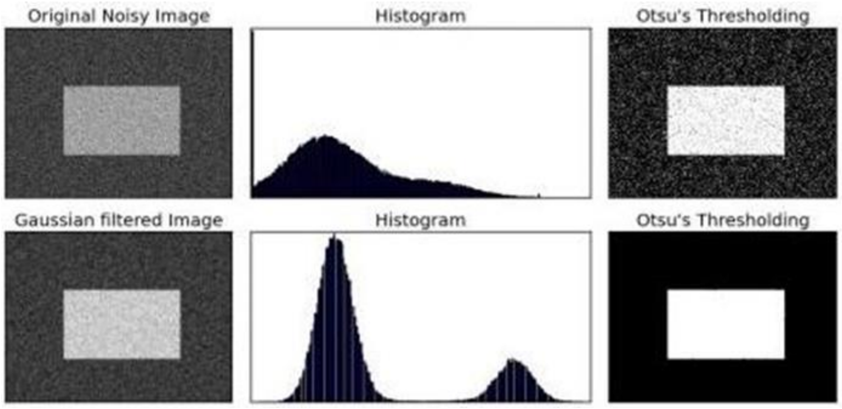
(b)



(c)



(d)



(e)

Figure 2-3Image Thresholding (A) Original Image, (B) Global Thresholding, (C) Adaptive Mean Thresholding, (D) Adaptive Gaussian Thresholding, (E) Ostu's Thresholding

it automatically calculates a threshold value from an image histogram for a bimodal image [8] [9].

For this, our cv2.threshold() function is used but passes an extra flag, *cv2.THRESH_OTSU*. **For threshold value, simply pass zero**. Then the algorithm finds the optimal threshold value and returns you as the second output, retVal. If Otsu thresholding is not used, retVal is the same as the threshold value we used [9] [8].

2.2.1.3 Smoothing Images

a) 2D Convolution (Image Filtering)

Images can be filtered with various low-pass filters (LPF), high-pass filters (HPF), etc. An LPF helps in removing noise or blurring the image. An HPF filter helps in finding edges in an image.

OpenCV provides a function, cv2.filter2D(), to convolve a kernel with an image. A 5x5 averaging filter kernel can be defined as follows:

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Filtering with the above kernel results in the following being performed: for each pixel, a 5x5 window is centered on this pixel, all pixels falling within this window are summed up, and the result is then divided by 25. This equates to computing the average of the pixel values inside that window. This operation is performed for all the pixels in the image to produce the output filtered image [10].

b) Gaussian Filtering

In this approach, instead of a box filter consisting of equal filter coefficients, a Gaussian kernel is used. It is done with the function, **cv2.GaussianBlur()**. We need to specify the width and height of the kernel which should be positive and odd. We also have to specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as equal to sigmaX. If both are given as zeros, they are calculated from the kernel size. Gaussian filtering is highly effective in removing Gaussian noise from the image [10] [11].

c) Median Filtering

Here, the function **cv2.medianBlur()** computes the median of all the pixels under the kernel window and the central pixel is replaced with this median value. This is highly effective in removing salt-and-pepper noise. One interesting thing to note is that, in the Gaussian and box filters, the filtered value for the central element can be a value that may not exist in the original image. However, this is not the case in median filtering, since the central element is always replaced by some pixel value in the image. This reduces the noise effectively. The kernel size must be a positive odd integer [10] [11].

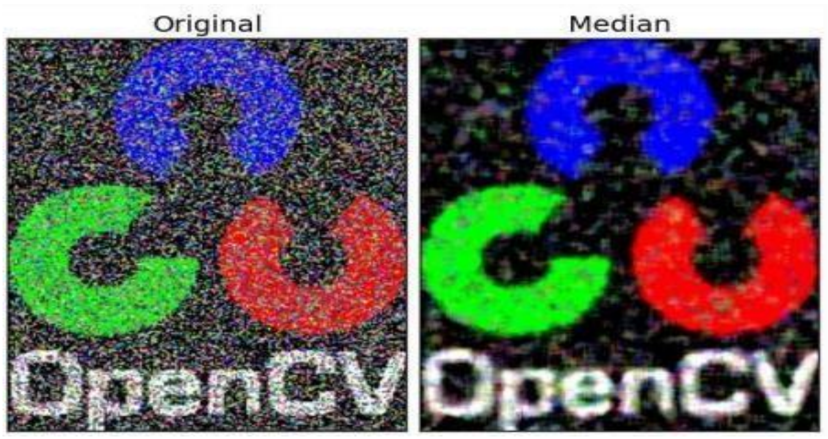
d) Bilateral Filtering

Bilateral filter, **cv2.bilateralFilter()**, is highly effective at noise removal while preserving edges. But the operation is slower compared to other filters.

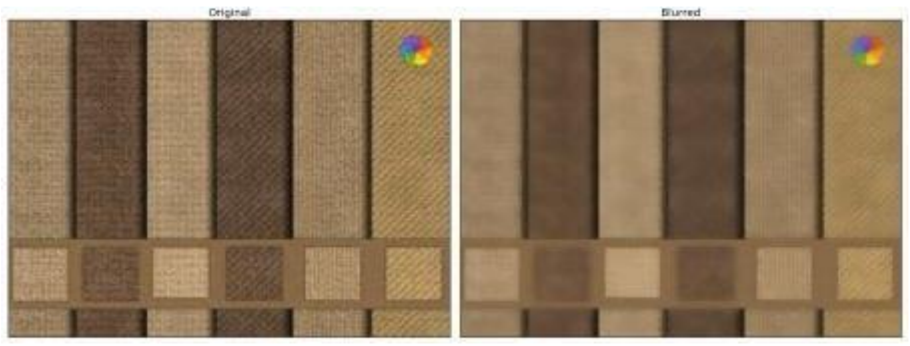
The bilateral filter also uses a Gaussian filter in the space domain, but it also uses one more (multiplicative) Gaussian filter component which is a function of pixel intensity differences. The Gaussian function of space makes sure that only pixels are 'spatial neighbors' are considered for filtering, while the Gaussian component applied in the intensity domain (a Gaussian function of intensity differences) ensures that only those pixels with intensities similar to that of the central pixel ('intensity neighbors') are included to compute the blurred intensity value. As a result, this method preserves edges, since for pixels lying near edges, neighboring pixels placed on the other side of the edge, and therefore exhibiting large intensity variations when compared to the central pixel, will not be included for blurring [11] [10].

OpenCV
OpenCV
OpenCV

(a) (b) (c)



(d)



(e)

Figure 2-4 Smoothing Images (a) Original Image, (b) 2D Convolution (Image Filtering), (c) Gaussian Filtering, (d) Mean Filtering, (e) Bilateral Filtering

2.2.1.4 Histogram Equalization

Consider an image whose pixel values are confined to some specific range of values only. For eg, a brighter image will have all pixels confined to high values. But a good image will have pixels from all regions of the image. So you need to stretch this histogram to either end and that is what Histogram Equalization does (in simple words). This normally improves the contrast of the image [8].

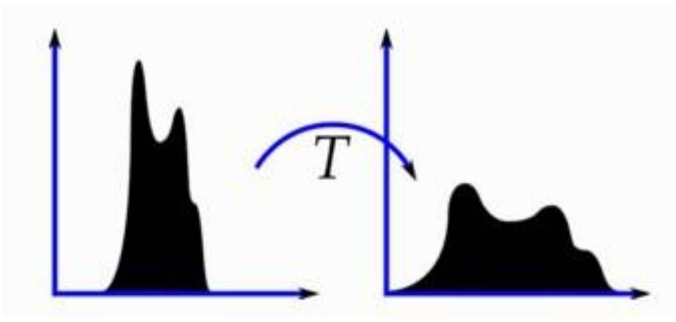
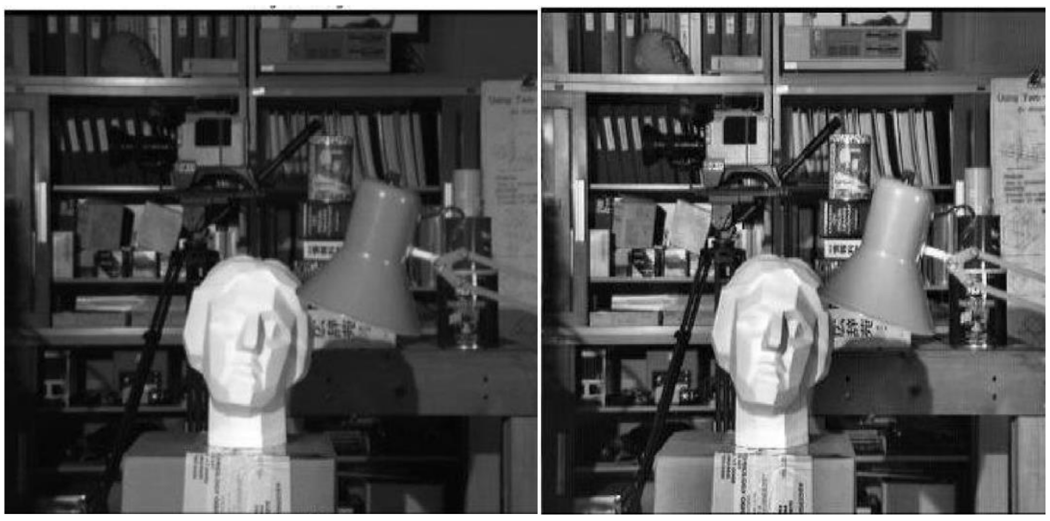


Figure 2-5 Histogram Equalization [8]

a) Contrast Limited Adaptive Histogram Equalization (CLAHE)

In adaptive histogram equalization, the image is divided into small blocks called "tiles" (tile size is 8x8 by default in OpenCV). Then each of these blocks is histogram equalized as usual. So in a small area, a histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, **contrast limiting** is applied. If any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied [8].



(a) (b)

Figure 2-6 Adaptive Histogram Equalization (a) Original Image (b) After applying CLAHE

2.2.1.5 Morphological transformation:

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, the second one is called structuring element or kernel which decides the nature of the operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient, etc also come into play. We will see them one-by-one with help of the following image [13] [14] in figure 2-7:

a) Erosion

The basic concept of erosion is similar to that of soil erosion, except that it erodes the foreground object's boundaries (Always try to keep foreground in white). So, what exactly does it do? The picture is slid through by the kernel (as in 2D convolution). If all the pixels under the kernel are 1, a pixel in the original image (either 1 or 0) will be regarded 1, else it will be degraded (made to zero). As a result, depending on the size of the kernel, all pixels along the boundary will be discarded. As a result, the foreground object's thickness or size shrinks, or the image's white region shrinks. It's useful for getting rid of little white sounds (as we've seen in the past)., etc [14]. Figure 2-7 (b) shows the morphological Erosion.

b) Dilation

It is just the opposite of erosion. f at least one pixel under the kernel is '1', a pixel element is '1'. As a result, the image's white zone expands, or the size of the foreground object expands. In most circumstances, such as noise reduction, erosion is followed by dilatation. Erosion reduces the size of our objects by removing white noise. As a result, we dilate it. They won't return now that the noise is gone, but our object area grows. It can also be used to reassemble damaged components of an object. [14]. . Figure 2-7 (c) shows the Dilation.

c) Opening

The opening is just another name of erosion followed by dilation. It is useful in removing noise, as we explained above. Here we use the function, cv.morphologyEx(). . Figure 2-7 (d) shows the opening property.

d) Closing

Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object. Figure 2-7 (e) shows the closing property

e) Morphological gradient

It is the difference between dilation and erosion of an image. Figure 2-7 (f) shows the gradient property

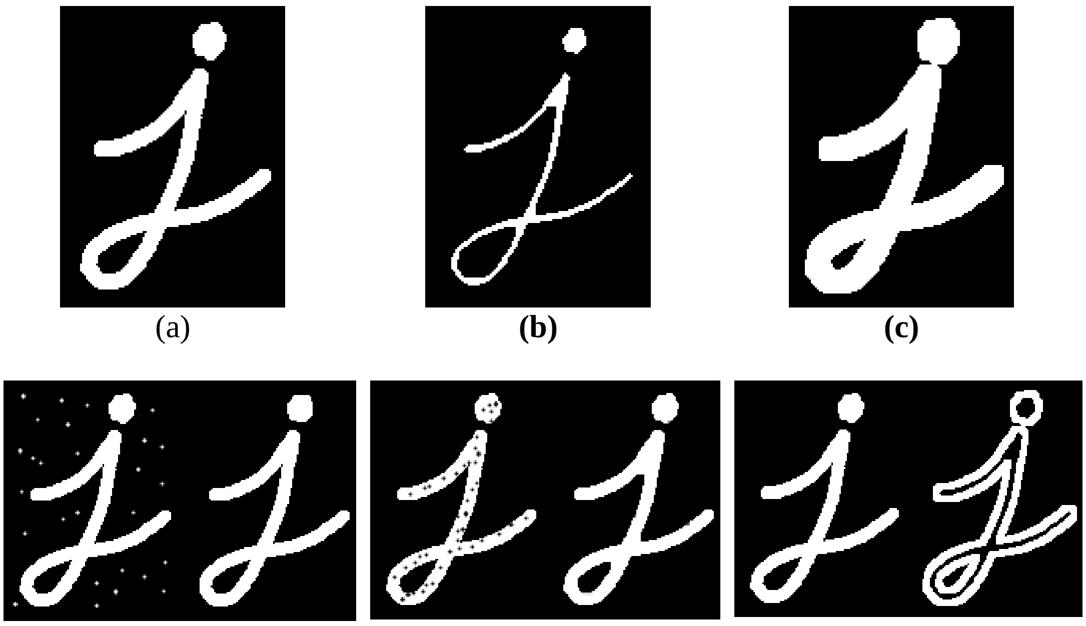


Figure 2-7 Morphological Operations (a) Original Image, (b) Erosion, (c) Dilation, (d)Opening (e)Closing (f)Morphological Gradient

2.3 RASPBERRY PI

Ever since its invention, Computer technology has progressed gradually and now became an integral part of everyday life. Application of more and more computer technology in the years Future will make the role of computer increasingly important. Everything from processing top-secret classified documents to watching movies, from rendering indefinitely complex 3D models to the recreation of the universe since the big bang via simulation – Computers have done what was thought to be impossible mere even a decade ago. Raspberry Pi (also known as RasPi) is an SBC (Single Board Computer) which is the size of a credit card developed by the Raspberry Pi Foundation in the UK (United Kingdom) to provide low-cost, high-performance computers that people can use to learn, solve problems and have fun. Raspberry Pi is a computer that can work as a full-featured desktop or notebook computer to create documents, process data with spreadsheets, watch movies, play games, and of course Programming.

The Raspberry Pi is perhaps the most inspiring computer available today. Although most of the computing devices we use (including phones, tablets, and games consoles) are designed to stop us from tinkering with them, the Raspberry Pi is exactly the opposite. From the moment first glance the Raspberry Pi makes it clear that it was built to be prodded with. It comes with the tools that are that will help to create unique electronic inventions using the Raspberry Pi. It is cheap and modular so it can be replaced easily if broken. There are several models of Raspberry Pi, among them the currently available models are:

- Raspberry Pi 1 Model A+
- Raspberry Pi 1 Model B+
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B
- Raspberry Pi ZERO
- Raspberry pi 4

For this project, the Raspberry Pi 3 Model B and Raspberry pi 4 were chosen because of the built-in wireless LAN connectivity. Other models can also be used with a Wireless adapter module.

2.3.1 Raspberry pi 3 model B

Raspberry Pi 3 Model B uses a system on a chip (SoC) from Broadcom – the BCM2837, which uses 64-bit architecture. The complete hardware specification is as follows:

SoC — Broadcom BCM2837
CPU — 4× ARM Cortex-A53, 1.2GHz
GPU — Broadcom VideoCore IV
RAM — 1GB LPDDR2 (900 MHz)
Networking — 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth — Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: — MicroSD
GPIO — 40-pin header, populated
Ports — HDMI, 3.5mm analog audio-video jack, 4× USB 2.0, Ethernet,
Camera Serial Interface (CSI), Display Serial Interface (DSI)

2.3.2 Raspberry pi 4

Raspberry Pi 3 Model B uses a system on a chip (SoC) from Broadcom – the BCM2837, which uses 64-bit architecture. The complete hardware specification is as follows:

SoC — Broadcom BCM2711
CPU — 4× ARM Cortex-A53, 1.5GHz quad-core 64-bit ARM Cortex-A72 CPU (~3x performance)
GPU — Broadcom VideoCore IV
RAM — 1GB, 2GB or 4GB of LPDDR4 SDRAM
Networking — Full-throughput Gigabit Ethernet, Dual-band 802.11ac wireless
Bluetooth — Bluetooth 5.0, Bluetooth Low Energy
Storage: — MicroSD
GPIO — 40-pin header, populated
Ports — HDMI, 3.5mm analog audio-video jack, 2× USB 2.0, Ethernet, 2x USB 3.0 ports Camera Serial Interface (CSI), Display Serial Interface (DSI)
Monitor — support Dual monitor, at resolutions up to 4k

Video — VideoCore VI graphics, supporting OpenGL ES 3. x,4Kp60 hardware decode of HEVC video

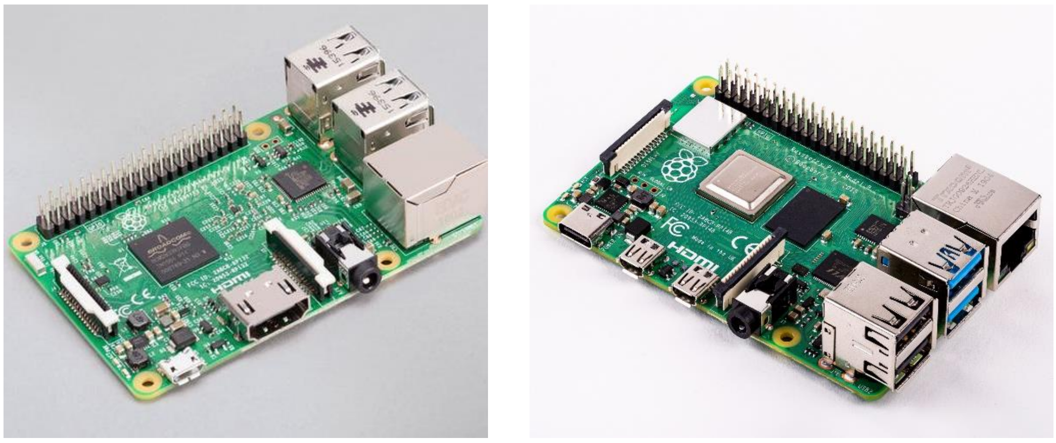

(a) (b)

Figure 2-8 (a) Raspberry pi 3 Model B [15] (b) Raspberry pi 4 [16]

2.4 ESP-32 CAM

ESP-WROOM-32 is a powerful, generic Wi-Fi, Bluetooth, and Bluetooth Low Energy MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming, and MP3 decoding. [17]

The ESP32-CAM is a full-featured microcontroller with a built-in video camera and a microSD card reader. It's low-cost and simple to use, making it ideal for IoT devices that require complex functions like image tracking and identification. Espressif's example software includes a sketch that enables you to create a web-based camera with a complex control panel. After you get the hang of programming the device you’ll find that it is very easy to use. [17]

2.4.1 TTGO camera plus

This module is based on ESPRESSIF-ESP32-DOWDQ6 240MHz Xtensa® dual-core 32-bit LX6 microprocessor chipset. It has both external and internal antennal solutions. The specification of this module:

Table 2-1 TTGO camera plus details [18] [19]

Chipset:	ESPRESSIF-ESP32-DOWDQ6 240MHz Xtensa® dual-core 32-bit LX6 microprocessor
FLASH:	QSPI 4M flash / 8M SRAM
SRAM:	520 kB SRAM + 8MBytes External SPRAM
KEY:	Reset
Display:	IPS Panel ST7789/1.3 Inch
Camera:	OV2640 2Megapixel
Working voltage:	2.3V-3.6V
Working current:	about 160mA
Working temperature range:	-40℃ to +85℃
Size:	69.13*28.41*8.45mm
SDCard:	Integrated external SD card slot
Sensor:	BME280/humidity/pressure/temperature
Power management chip:	IP5306
Power Supply:	USB 5V/1A
Charging current:	1A
Battery:	3.7V lithium battery
frequency range:	2.4GHz~2.5GHz(2400M~2483.5M)
Transmit Power:	22dBm
communication distance:	300m
Networking protocol:	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT

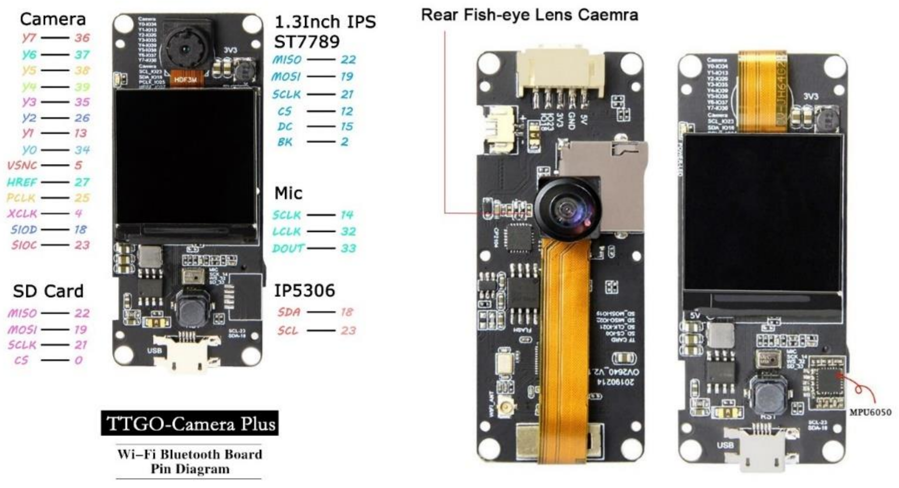


Figure 2-9 TTGO Camera plus module. [19]

2.5 FLASK SERVER

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[2] 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. [20]

Components are:

- Werkzeug
- Jinja
- MarkupSafe
- ItsDangerous

Features:

- Development server and debugger
- Integrated support for unit testing

- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired [20]

2.6 SQLITE3 DATABASE

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle. [22] [23]

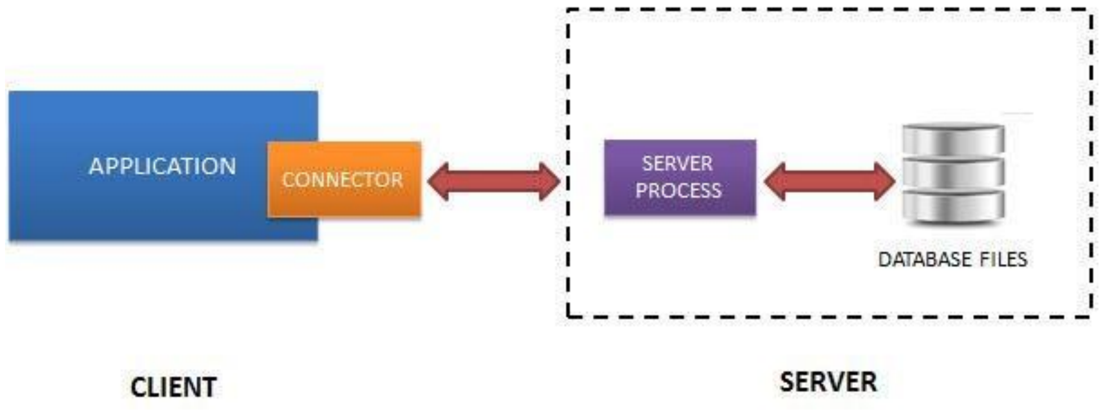


Figure 2-10 Overview of SQLite [23]

2.6.1 Uses of SQLITE3

- Embedded devices and the internet of things
- Application file format
- Websites
- Data analysis

- Cache for enterprise data
- Server-side database
- Data transfer format
- File archive and/or data container Literature Review
- Replacement for ad hoc disk files
- Internal or temporary databases
- Stand-in for an enterprise database during demos or testing
- Education and Training
- Experimental SQL language extensions [24].

2.7 FLUTTER

Flutter is an open-source framework by google for building beautiful, natively compiled, multi-platform applications from a single code base. It is mainly based on widgets. Widgets are generally defined in three basic types:

- Stateful widgets,
- Stateless widgets
- Inherited widgets

Being the central class hierarchy in the Flutter framework the three basic types of widgets are used in the construction of every Flutter application. Although all the instances of a widget are immutable, the Stateful widget allows the interaction between user and application. By giving access to the method **setState**, the state can be maintained in separate state objects. Alternatively, the Stateless widget acts as a constant, and before anything displayed can be changed, the widget must be recreated. The Inherited widget works by allowing another widget to subscribe to the Inherited widget's state allowing the state to be passed down to its children.

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

Figure 2-11 Flutter [25]

Flutter maintains official support for the following IDEs and editors via plugins:

- IntelliJ IDEA
- Android Studio
- Visual Studio Code

2.8 SOLIDWORKS

Dassault Systems' SolidWorks is a solid modeling computer-aided design (CAD) and computer-aided engineering (CAE) application. From start to finish, SOLIDWORKS is used to build mechatronic systems. At the initial stage, the software is used for planning, visual ideation, modeling, feasibility assessment, prototyping, and project management. After that, the program is used to design and create mechanical, electrical, and software components. Finally, the program can be used to manage devices, analytics, data automation, and cloud services.

2.9 CONCEPTDRAW

ConceptDraw DIAGRAM (formerly ConceptDraw PRO) is a proprietary diagramming software that may be used to produce business graphics such as diagrams, flowcharts, Infographics, data visualization for business process models, data presentation, and project management documentation. It can also be used to make professional and technical diagrams, such as topology diagrams for computer networks, engineering schemes, floor layouts, and other technical graphic forms.

CHAPTER 3

3. METHOD

The main purpose of this system is to reduce covid exposure inside the ICU. So, the system provides a quick and easy way for hospital staff to constantly monitor bedside monitor images in non-covid space. The complete system must be inexpensive and can easily be operated so it can be situated by an untrained person. Figure 3-1 shows the basic block diagram of our system:

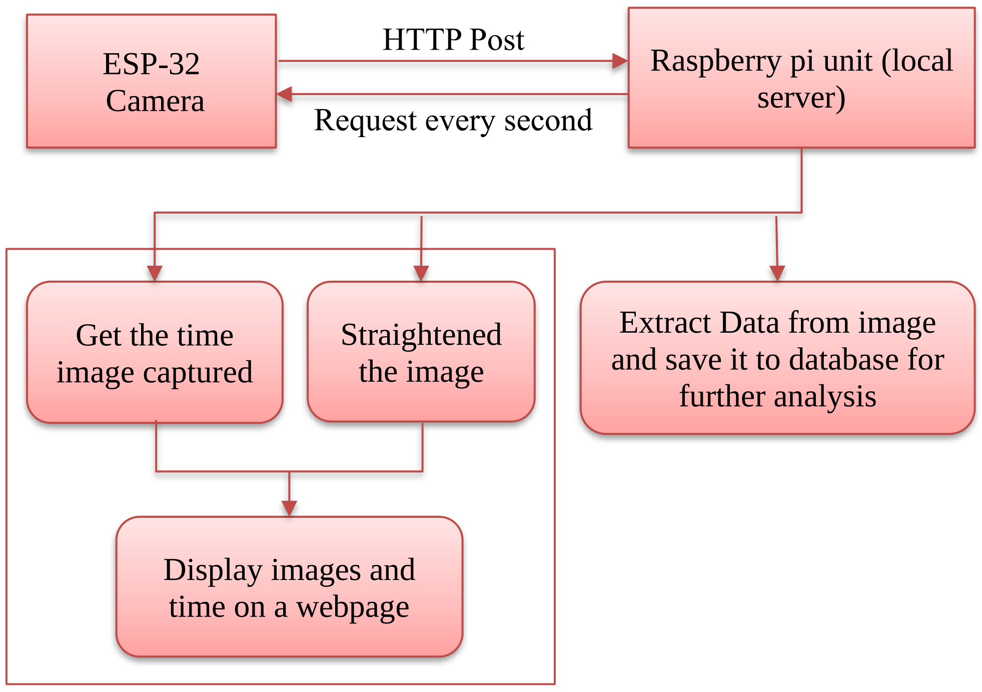

ESP-32 Camera
HTTP Post
Request every second
Raspberry pi unit (local server)
Get the time image captured
Straightened the image
Extract Data from image and save it to database for further analysis
Display images and time on a webpage

Figure 3-1 Overview of the entire process of ICU monitoring device

ESP32-cam captures the image of a bedside monitor and HTTP POST it to the raspberry pi local server. The image will post every second which means 1 frame per second. As a local server, we choose flask as the backend. It's a micro web framework based on python. In other words, it doesn't require any libraries, and also it is lightweight. Therefore, we choose this server for raspberry pi. After HTTP gets the image to the server, it stored the image in the RPI folder. The original image was not straightened yet. Next, the image follows a computer vision script that will straighten the image which will show only the image of the bedside monitor and save it to another folder. Again, for data read, the straightened image will send to another image processing script where we extract the Heart rate data and oxygen saturation data. For further analysis, we save those data to the sqlite3 database according to id, time, and image name. These data send to the HTML page through the socket and displayed on the Webpage. For showing the webpage over the internet, we capture a screenshot of the local webpage of the raspberry pi server unit and send it to another hosting server. We used MQTT as it is fast reliable and works with low bandwidth and low latency connection. It runs on TCP/IP and connects with WIFI or cellular internet with subscribe-publish architecture.

3.1 IMAGE ACQUISITION

It is an ESP32 development board that contains an ESP32 microcontroller, a camera (0v2640)/ ESP fisheye, a display (tft_ESPi/Ips panel ST7789), etc. Its code is based on Arduino IDE. First, include all the additional libraries. Download the st7789 library and set up USER.h for the display. Then add network credentials in the following variable named SSID and password. Make sure to select the right camera module. As we use ESP-eye for testing. Define all variables needed as server name(server_ip), server path, server port, etc. Invoke ST7789 library as TFT. In the setup function first, connect ESP with the router using network credentials by using the WIFI.h library. Sometimes, ESP will stack if it never finds the WIFI. So, create a time loop that while ESP doesn't connect with the router, it will automatically restart ESP. Then initialize the camera. Again, create another loop if the camera is not found, show an error in display and restart ESP. After that, set the brightness, contrast, saturation, sharpness, denoise, horizontal flip, vertical flip, and quality of the camera sensor. OV2640 camera can capture images of the following resolution:

1. UXGA (1600x1200)
2. SXGA (1280x1024)
3. XGA (1024x768)
4. SVGA (800x600)
5. VGA (640x480)
6. CIF (400x296)
7. QVGA (320x240)
8. HQVGA (240x176)

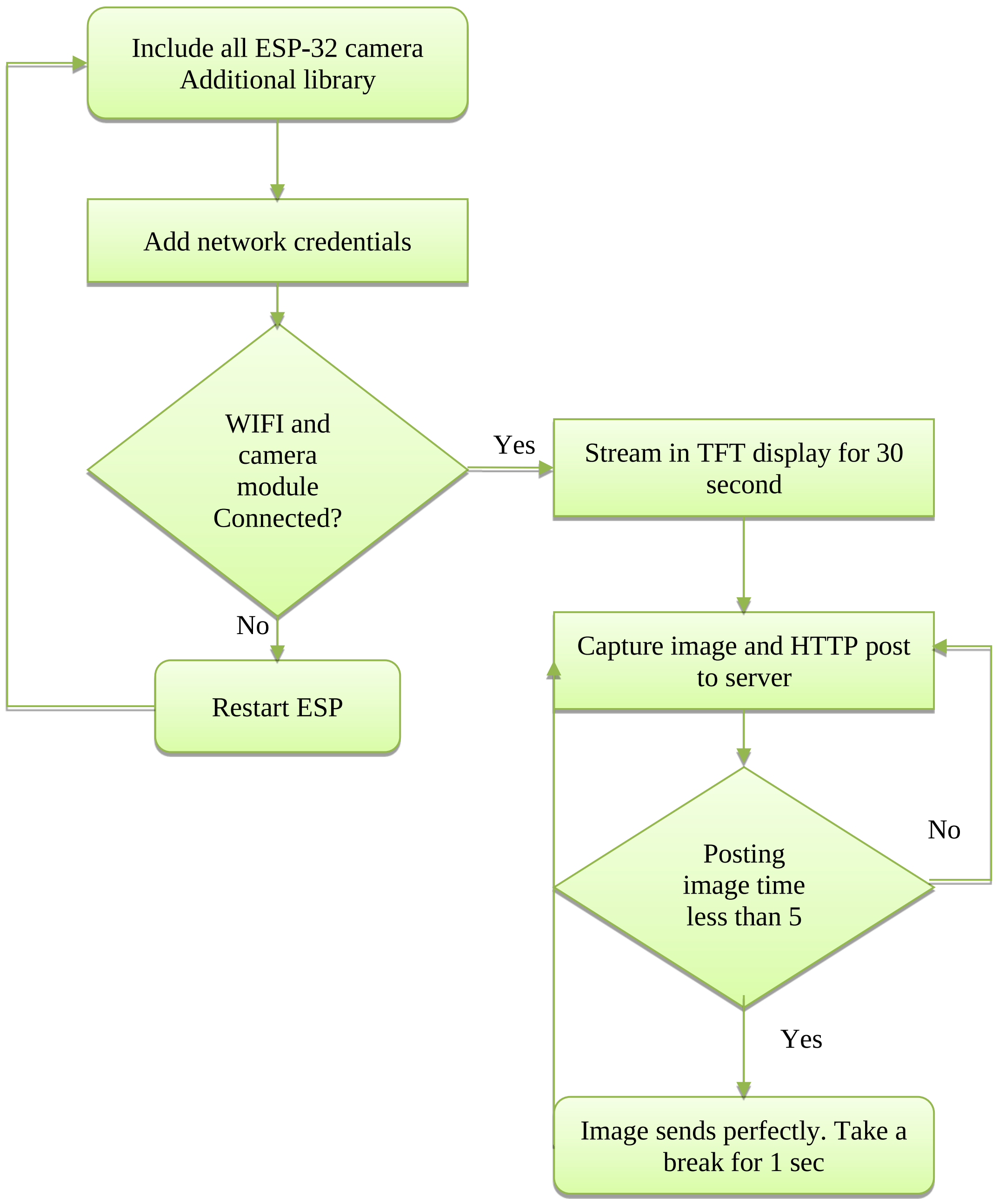
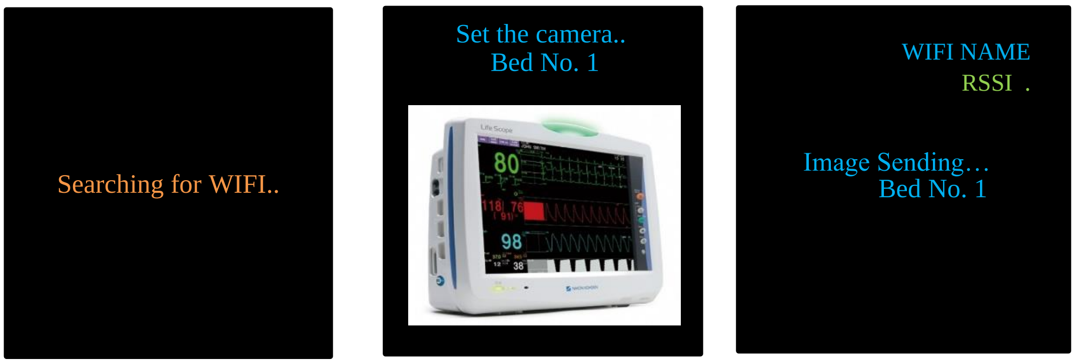

Include all ESP-32 camera Additional library
Add network credentials
WIFI and camera module Connected?
Yes
Stream in TFT display for 30 second
No
Restart ESP
Capture image and HTTP post to server
Posting image time less than 5
No
Yes
Image sends perfectly. Take a break for 1 sec

Figure 3-2 Flow Chart of HTTP post an image from ESP-32 Camera

We use images of UXGA resolution and the quality level is 15. In the loop function, create a loop that if Wi-Fi is connected with ESP-32, show image into the display for 30 seconds, and after 30-second camera will start posting an image to the server. If one image post to the server from the camera it will take 2 seconds break and post another image.

The process will run until the image stack. Sometimes posting an image from the camera takes too much time. So, we create another loop if the image won't post in 10 seconds after capture, then it will recapture another image and post it to the server.



(a) (b) (c)

Figure 3-3 TTGO camera plus display (a)Searching network to connect, (b)Display camera for 30 seconds, (c)Start sending image to the server after 30 sec.

3.2 SERVER PROCESS

To execute Python Code concerning the HTTP Request we need a uWSGI Server between our Web Server and Python Web Application. We whole process of creating a flask server in raspberry pi is shown below:

1. Update and upgrade RPI packages.
2. Install Nginx and start the Nginx service.
3. Install Flask and uWSGI.
4. First create a sample flask app.
5. Test Flask app with python and uWSGI.
6. Create and test the uWSGI initialization File.
7. Configure uWSGI to auto start after Reboot raspberry pi.
8. Reverse proxy(configure NGINX to redirect web traffic to uWSGI)

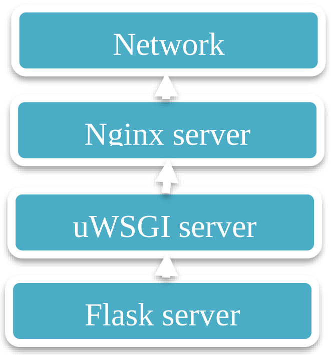


Figure 3-4 Creating a Flask server with Nginx and uWSGI backend [29]

After successfully installing the flask in raspberry pi, import all additional libraries in the flask app. First, create a route for uploading images from esp32 cam. All esp32 cam will hit this route and post image. This upload route will always be busy because every time camera will hit this route to upload images. After getting an image, we save those images to a fixed folder. To straighten those images, copy all that images and move to another folder where straighten function script will run. In the dashboard route, we show the straighten folder images according to the bed number of the ICU. To get data from images create a function in flask script from where call the data read function. It will extract the Heart rate data and Oxygen saturation data. Save those data to the database.

Here we create a Sqlite3 database. It helps to query in the future. In the database, create a column of id, monitor name, heart rate, oxygen saturation, and date posted of image. ID column will be autoincremented integer. In the monitor name column, save the image name which esp32 was posted in the upload route. Get the time when the image was posted on the server and save this at the database as in the date posted column.

To show data on the webpage with images, we pass those data through flask-socket. First, initiate the database and query the heart rate and oxygen saturation data according to descending order of monitor name and id. The socket will pass those data to the JavaScript file. It will break for 1 second every time. When a client is connected with the URL socket is on a pass data to the JavaScript page continuously. And when the URL is closed socket will disconnect from the client and stop the process.

Another route is for showing the Chart. First query Heartrate data and oxygen saturation data from database and Jsonify those data to a fixed URL. We prefer AJAX to get a request in that case. It is getting or retrieving some data from the server to the JavaScript page. It is a slow process but, in that case, we can ignore it.

3.3 IMAGE TRANSFORMATION

After posting an image to the local server (raspberry pi server) our image processing program will start. Here are two steps to complete the image processing part:

1. Straighten the image
2. Read data correctly from the image

3.3.1 Straighten the image

First of all, the image which we will get from the HTTP post is the whole environment that includes the bedside monitor and surroundings. For fixing the angle resize the image to our desired size. Convert the image BGR to Grayscale. Add some blur to the grayscale image for noise reduction. It can be used median or gaussian blur. To get the edges here we use the canny edge algorithm. In canny edge, there are two threshold parameters. By changing this threshold value, we check which threshold value we get the perfect edges. To get the perfect threshold values we check with various angled images over and over again. Then dilate and erode the edges to connect the edges. Calculate the contours of the dilated image. From the contours, we need to find out the perfect rectangle shape of the display image. For this, use the contour approximation method. This method is based on the Douglas-Peucker algorithm. After getting our desired rectangle contour, we need to re transform it. For this, from the rectangle contour, we calculate the x-coordinate and y-coordinate of the rectangle corner. Using the corner data in perspective transform we get our desired fixed angle output image which is directly shown on the webpage.

- Resize is one of the biggest factors.
- We check over and over again with the different threshold values with different images to find out the perfect edges for all images.

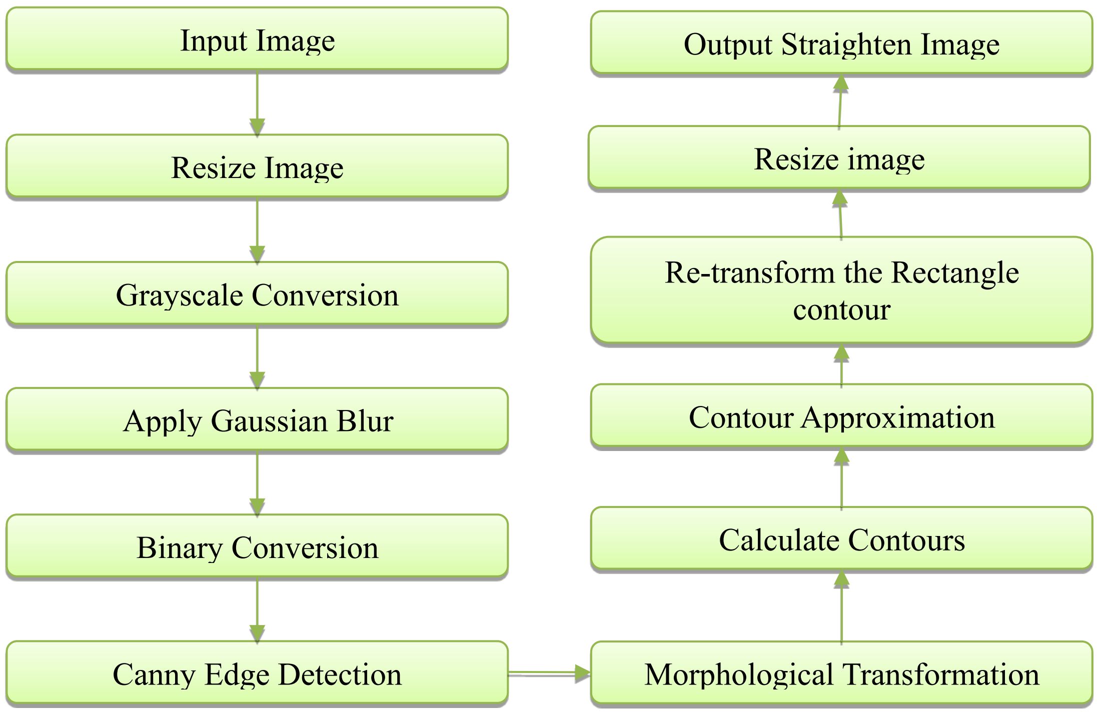


Figure 3-5 Algorithm of the Straightened image

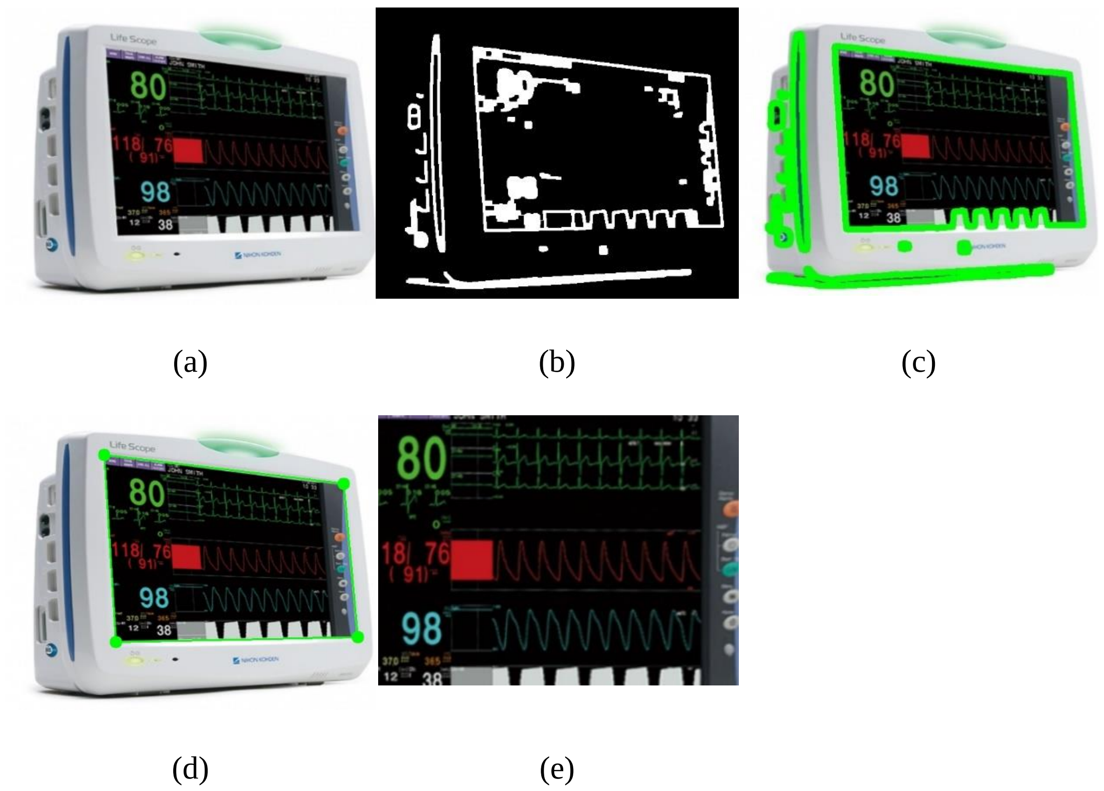


Figure 3-6 (a) Original Image, (b)Threshold Image, (c) Canny edge detection, (d)Get the big rectangle contour from the canny edged image, (e) Final Image

3.3.2 Read data from the image:

After straightening the image, we start our next task of the project which is to read data from the straightened image. Reading data from the images was one of the toughest parts of this project. We divide this read data process into two-part.

- Segmentation
- Tesseract-OCR

3.3.2.1 Segmentation

a) Color-based segmentation:

The image was first resized to 800x400 pixels. Convert the color channel to BGR to HSV. We can convert change BGR to RGB also. We choose HSV because here we get the perfect range for a specific color. The hue channel is our main target. Like, for green color our hue range is (30-75). After fixing the hue we fix the saturation and value for the specific color. How do we get this range? -> First, we create a trackbar. We split the image which we need to find the color range. We create six trackbars for the upper and lower range for hue, saturation, and value using cv2.createTrackbar. For an image, we get a range of HSV. Check the same process with other images and continue this process until the perfect range for color is found. Then repeatedly try the process for another color also. By using this process, we get the color range for green paste and red color. <- Create an array for the green color range. There must be a lower range array and an upper range array. Masking out the green color. Using the bitwise operation function we get only our desired green color which is the value of heart rate. Then convert the image into greyscale and apply a normal threshold to the greyscale image. Calculate the contour area and remove the small pixels. In the green mask image, there must be small pixels and noise. After calculating the contour area, we can remove those small pixels. Apply morphological erosion/opening/dilation because when we remove small pixels some pixels are also removed from the big contour. If those big contours are not perfectly connected the OCR can’t detect character perfectly. This helps the OCR engine to read data correctly from the image. Apply the same process for red color (Blood pressure) and paste color (Oxygen saturation). Then mask all colors into a single image.

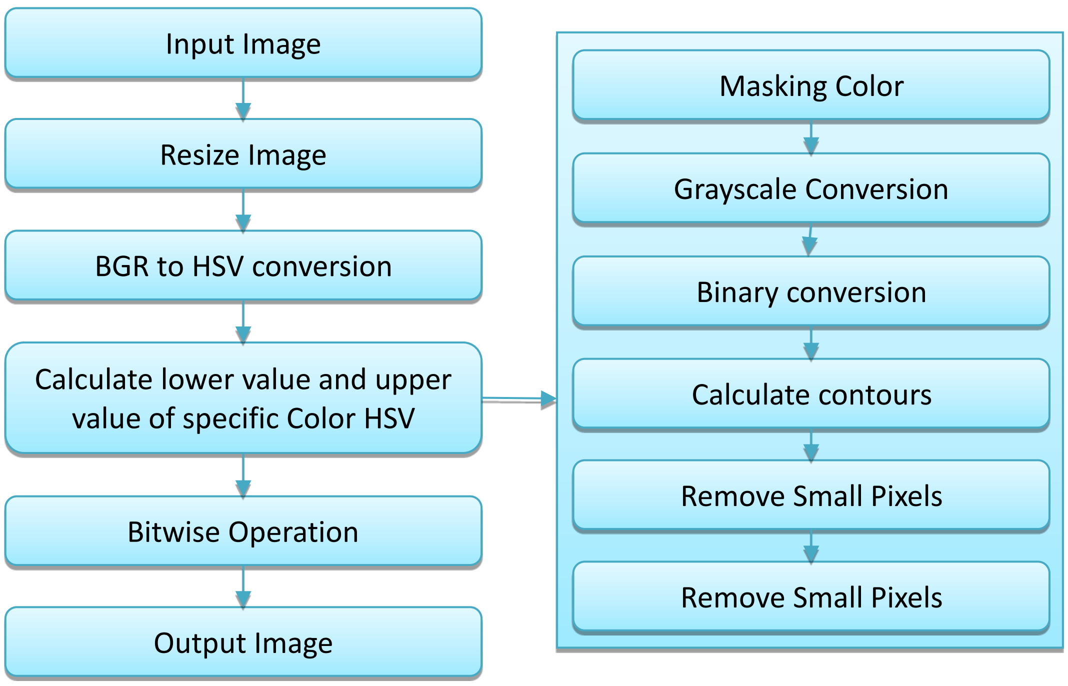


Figure 3-7 Algorithm for color-based segmentation

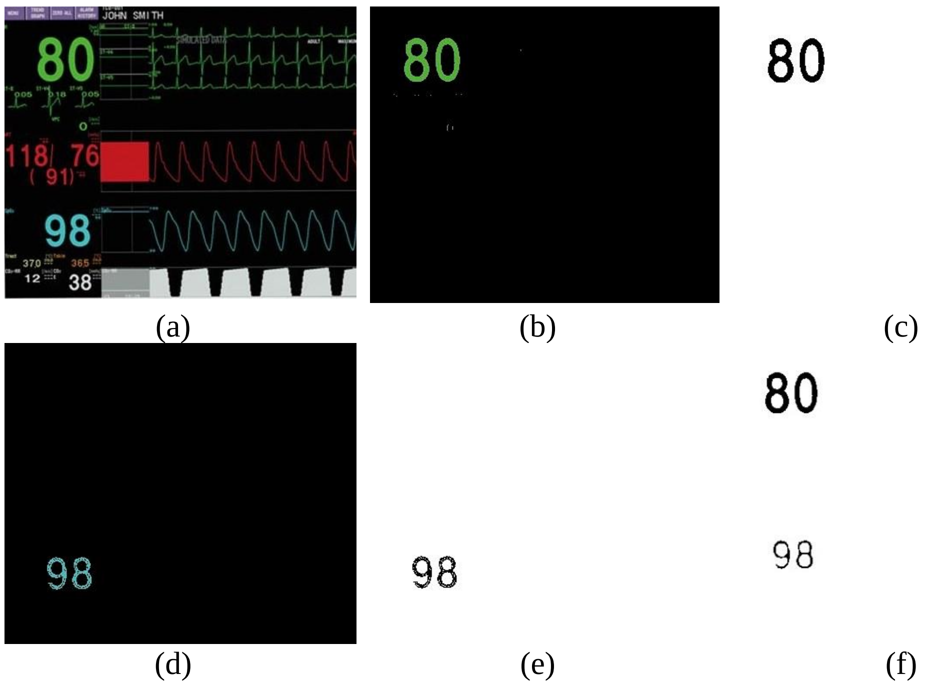
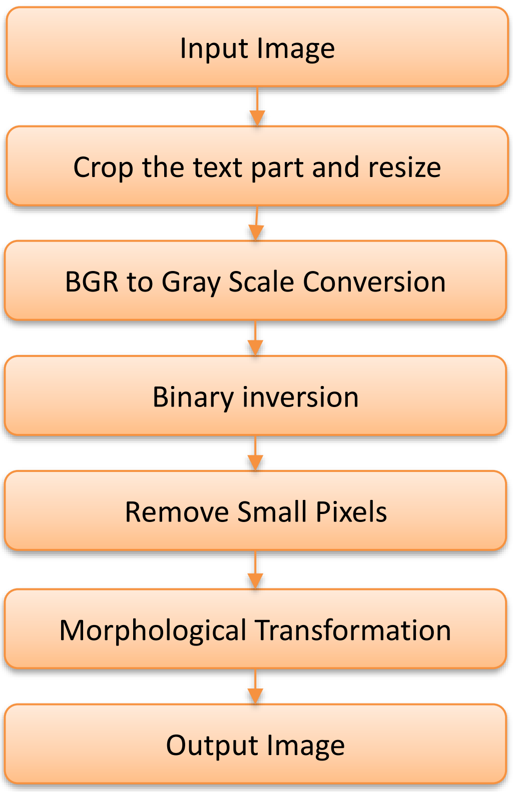


Figure 3-8 (a)Original Image, (b)Masking Green, (c)Bitwise operation and inary inversion, (d)Masking Blue, (e)Bitwise operation and binary inversion, (f)Add d and f images(bitwise)

b) Pixel-based segmentation:

First Resize the image and crop the part of the image where the values are situated. Then again resize the cropped image. Turn the image into a grayscale and calculate the threshold value. Apply normal threshold/Binary inversion to the greyscale image. Calculate the contour area and remove the small pixels from the inverted binarized image. Then we get our desired character which looks bolder. So, apply morphological erosion and dilation to perfect the character.



Input Image
Crop the text part and resize
BGR to Gray Scale Conversion
Binary inversion
Remove Small Pixels
Morphological Transformation
Output Image

Figure 3-9 Algorithm of Pixel-based segmentation

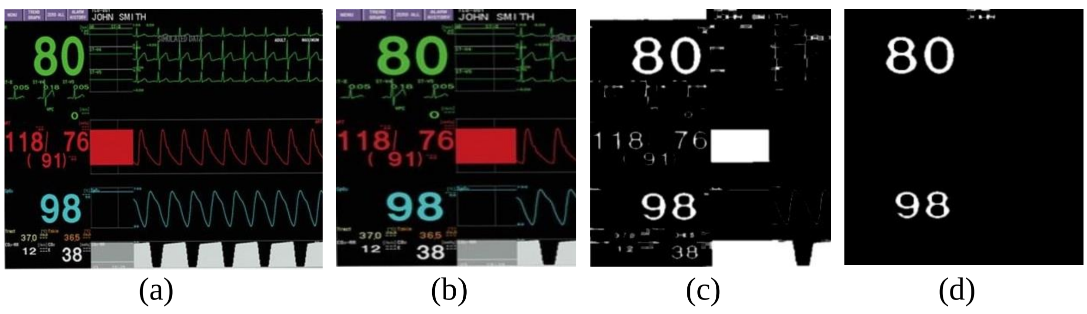


Figure 3-10 (a)Original Image, (b)Cropped and resized Image, (c)Binary Inversion, (d)Final Image

3.3.2.2 Tesseract OCR

We use the PYTESSERACT library to read image data as a string. Here we use the image_to_string function. We set the –OEM number 3 and –psm number 11. Using PSM 11 find out the character without any order as much as possible. Again, the set output base is a digit. It helps to find only the digits from the image. We get this data as a list that returns with proper extension.

3.4 ANDROID APPLICATION PROCESS

To monitor patient data in real-time we developed an application for android smartphones. We choose flutter SDK which is created by Google. It is an open-source UI SDK and uses Dart language as a backend. Flutter is so handful for both IOS and Android applications. We run the app locally. That means when the raspberry pi server and the mobile wife are running under the same router the apps will work. We use a dart package named URL launcher to hit directly to the chart URL. By using this app doctors can see the previous Heartrate and oxygen saturation data.

3.5 REMOTE ACCESS PROCESS

By using the MQTT protocol we publish our screenshot from the RPI server to the broker. According to MQTT architecture, the hosting server plays the broker role, and RPI, APPs are clients. When clients/doctors tap on the button of the app it subscribes and RPI publishes a screenshot which would be shown on doctors' end.

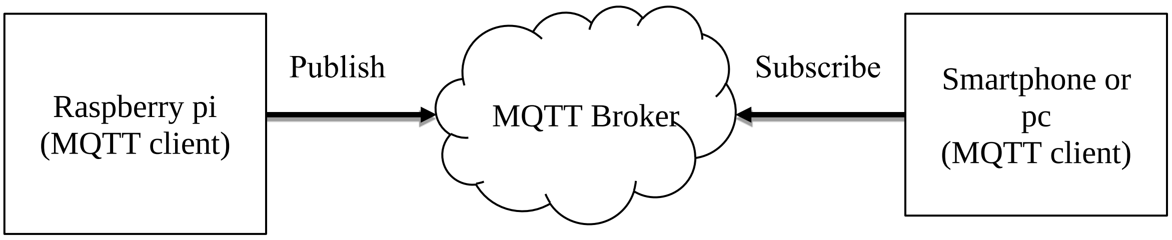


Figure 3-11 MQTT Algorithm

CHAPTER 4

4. RESULT

This project research has explored the development of continuously HTTP POST images to the local server and automatic detection of the data from the image and show the image and data to the server. In the data detection process, the images are the vital point. Sufficient good resolution images are required to develop and evaluate the algorithm. The performance of the developed systems is discussed in this section.

4.1 ESP32-CAM PROCESS

We collect images of the bedside monitors using an ESP-32 cam known as TTGO camera plus. We run our analysis on 8 esp-32 cameras and collect more than 26,000 images. Here, for all cameras and distances, the minimum, maximum, average, and standard deviation values are shown below:

Table 4-1 Data collected from TTGO camera plus

Distance From CAM	**Min(ms)**	**Average(ms)**	**S.D.(ms)**
0-8 feet	322.38	425.23	176.4997
8-16feet	345.125	593.03	415.9217
16-20feet	382.5	906.8	743.87
20-24feet	333.13	1556.35	1306.12
25+feet with external antenna	111	389.27	684.5

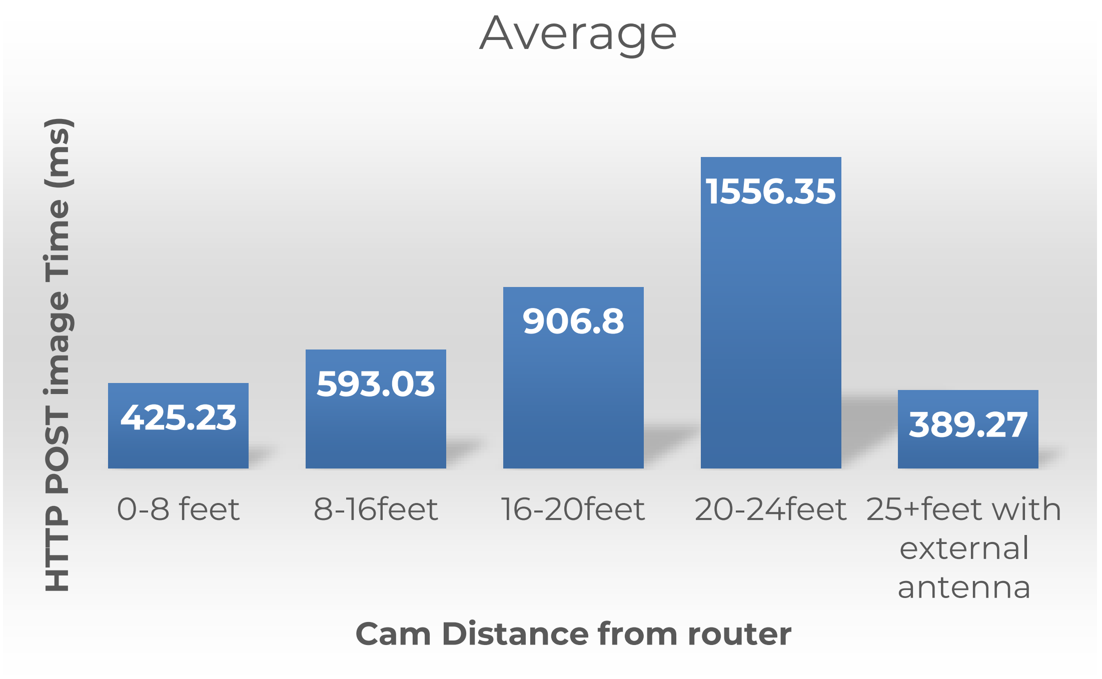
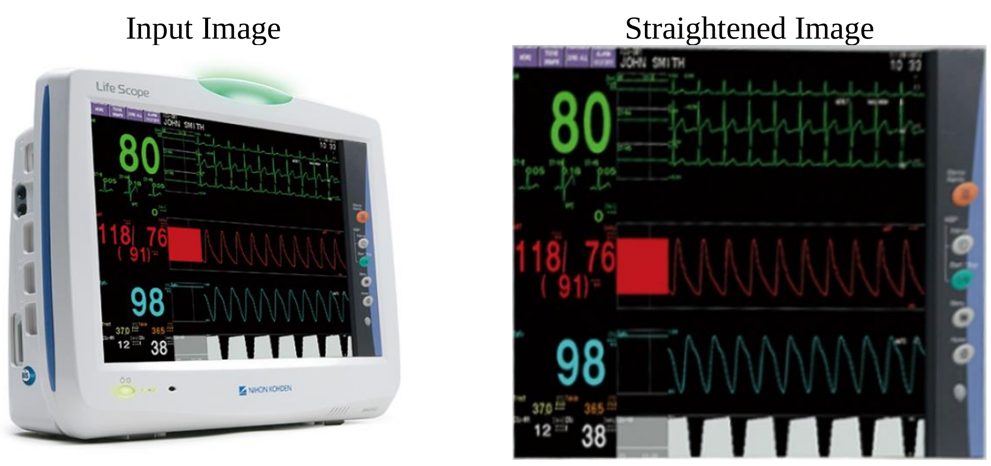


Figure 4-1 Average time of HTTP post image)

4.2 STRAIGHTENED IMAGE

After collecting the image from ESP-32, the Flask server receives it. Then it sends it to the angle.py function which helps to straighten the image. Execution time in Raspberry pi 3 and Raspberry pi 4 is pretty similar. The difference is milliseconds per image. Visual performance is shown in the figure:



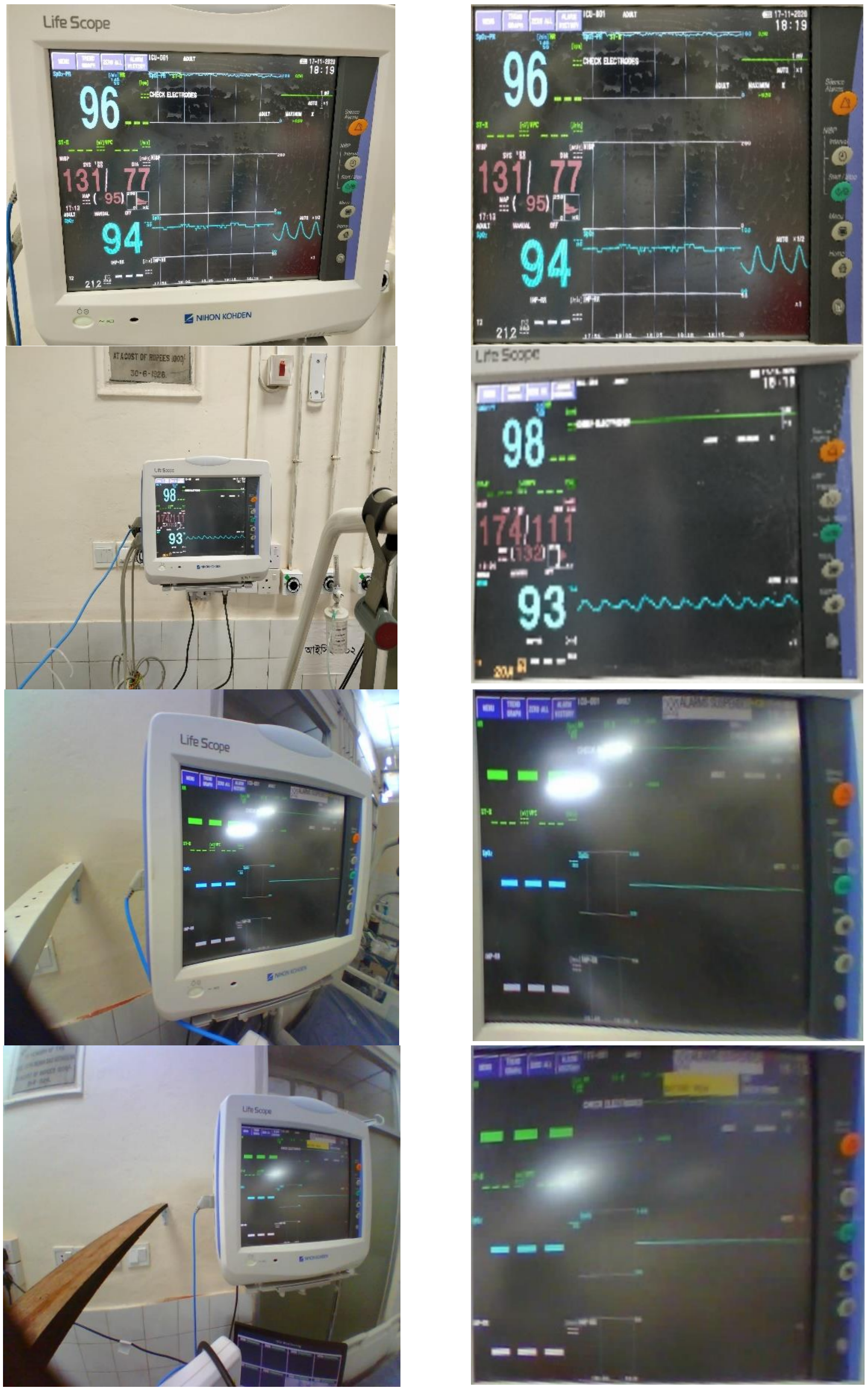


Figure 4-2 Straightened image visual performance

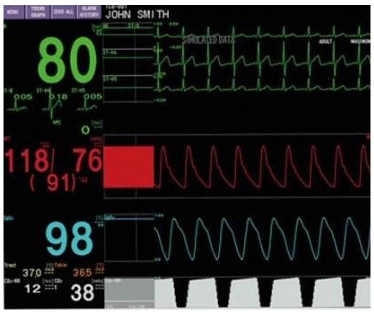
4.3 SEGMENTATION IMAGE

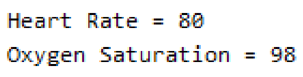
The segmentation of images plays a vital role in extracting data from the image. The performance of the color-based segmentation is more accurate than the pixel-based segmentation. Because in color-based segmentation we research with different colors individually e.g., Masking green for the Heartrate and Masking Blue for the Oxygen saturation. But in the case of pixel-based segmentation, we only threshold apply a Threshold operation to get the black and white version of the images. Then using the morphological transformation and contour approximation method we remove the small pixel and finalized the image. Here both HR and SPO2 segmentate at the same time.

The time to execute the code is more for color-based segmentation but the accuracy is far better. Because we research with different color values and approximate a certain color level for individual colors.

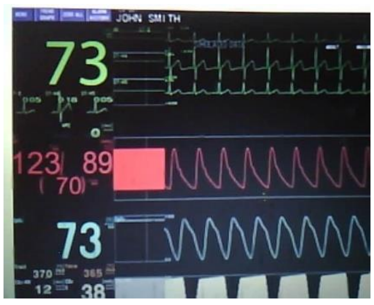
The execution time is less for the pixel-based segmentation model but not so accurate. Visual performance is shown in the figure.

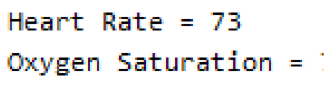
Input image **Output Image** **OCR**



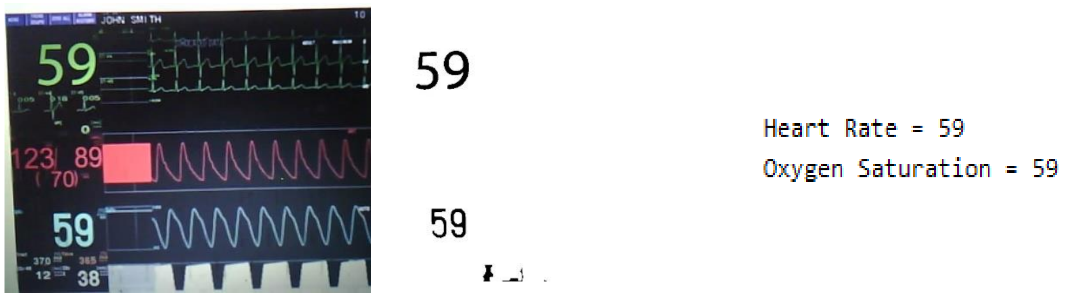


```
Heart Rate = 80
Oxygen Saturation = 98
```





```
Heart Rate = 73
Oxygen Saturation =
```



```
Heart Rate = 59
Oxygen Saturation = 59
```

Figure 4-3 Segmentation Image

4.4 OCR

Tesseract OCR runs using the Google OCR engine. Execution time is more for the raspberry pi server than the windows web server. To use OCR we need graphics which is not available for the raspberry pi that’s why it is slow for Raspberry. We have calculated the execution time among raspberry pi 3 model B, Raspberry pi 4, and windows server. It is shown in table 4-2

Table 4-2 Average Execution time of different Operating System

OS	Execution time (for 8 images) (seconds)
Windows (8GB)	2
Raspberry pi 3 model B (1GB)	10
Raspberry pi 4 (2 GB)	7

4.5 WEB SERVER INTERFACE

There are two services running background in the raspberry. One is for running the server named **"ICU.service"** and the other one is the automated show of the web page using the chromium named **"kiosk.service"**. In the **kiosk.service** it hit the server URL for showing the **"/images"** webpage. The service will start when the raspberry pi boot up and automatically calls it from **systemctl.**

We show the time that image posting into the particular bed in the web interface. After calculating the average receive rate, we set the image changing rate to 1 second. We can show 8/10 images from individual ESP-32 cameras at a time. All images will show the bed number. The camera will be fixed for the fixed bed. It was assigned with the ESP-32 cam. Again, on the webpage, there is a box for HR and SPO2 data. The **segmentation** and **Straightened** code continuously run in the background. We set an asynchronous process to run it. It continuously extracts the value from the images. The change of data depends on the segmentation and straightened script execution time.

We pass all data e.g., HR, SPO2, time difference through the flask socket to the HTML webpage. Thus doctors can easily see this. The image of the web interface is shown in the figure:

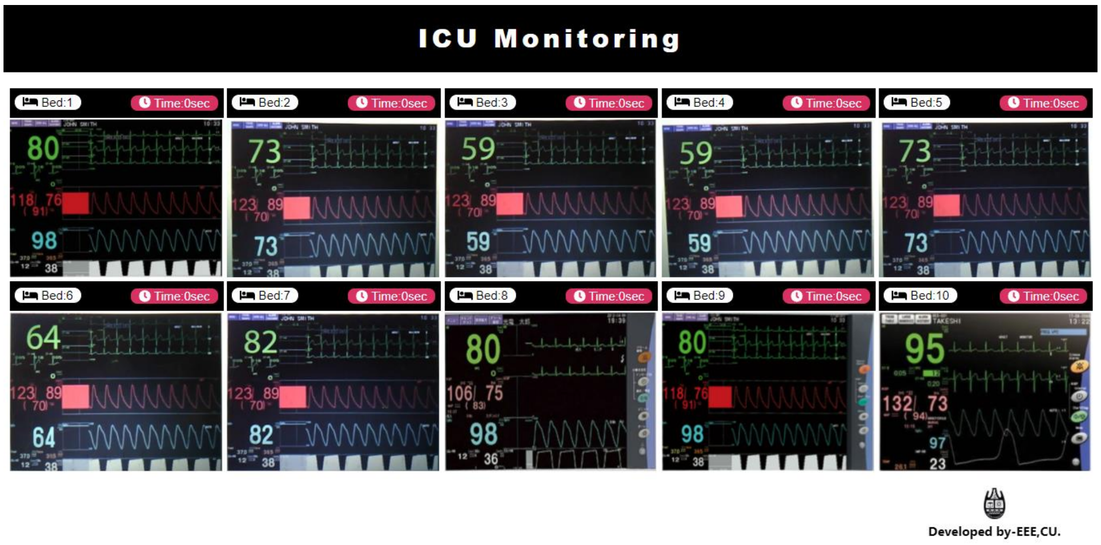
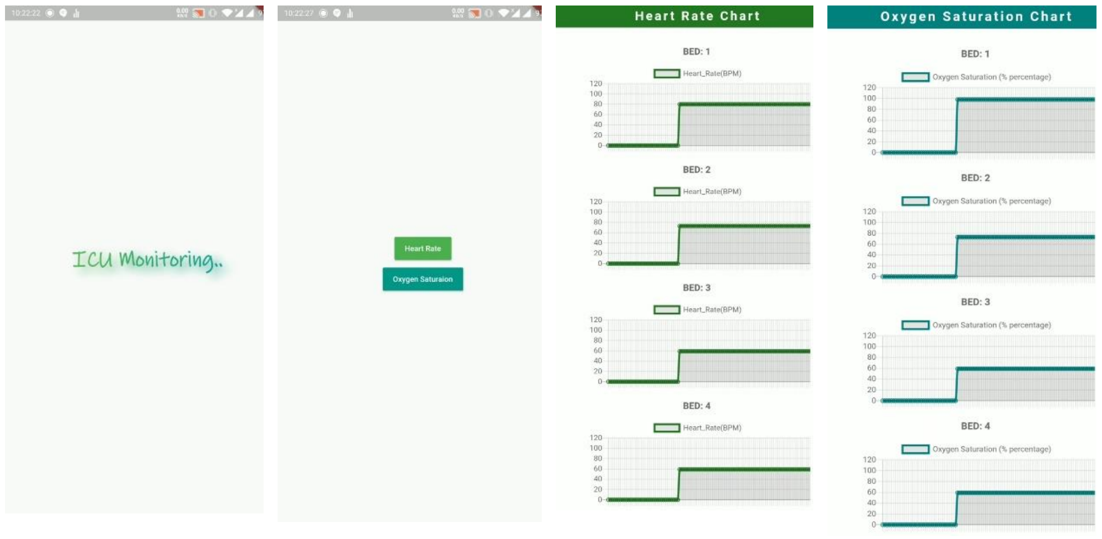


Figure 4-4Web interface

4.6 ANDROID APP INTERFACE

In the backend flask server, we create multiple routes to show HR data and SPO2 data through Flutter App. We hit the route using the **“url_Launcher”** package. This helps to show directly to the HR page and SPO2 page. This will introduce individual pages for both data. Every server change will automatically show in the application. Currently, it works locally. That means android will always connect with the router. When the server is down the app doesn’t show data. This app will start working instantly with the server.



(a) (b) (c) (d)

Figure 4-5 Android Application interface (a)Intro (b)Choose section, (c)Heart rate interface, (d)Oxygen Saturation Interface

4.7 3D DESIGN

We choose the software Solidworks to design our desired tools for setting up our project inside CGH ICU. We design a case for the TTGO camera plus. We have designed some clamps which are set up on the backside of the bedside monitor.

For perfection, we design a floorplan. In the floor plan, we show how we design our connection inside ICU, Nurse room, and Doctors room. Some of the designs shown in the figure below:

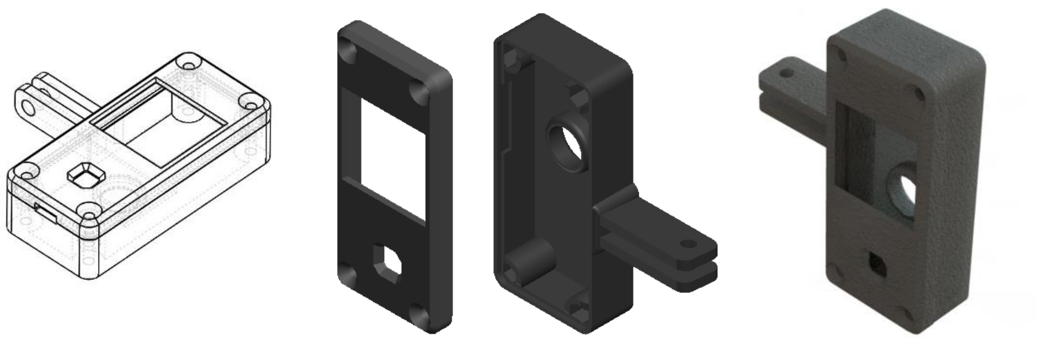


Figure 4-6 TTGO Camera Case

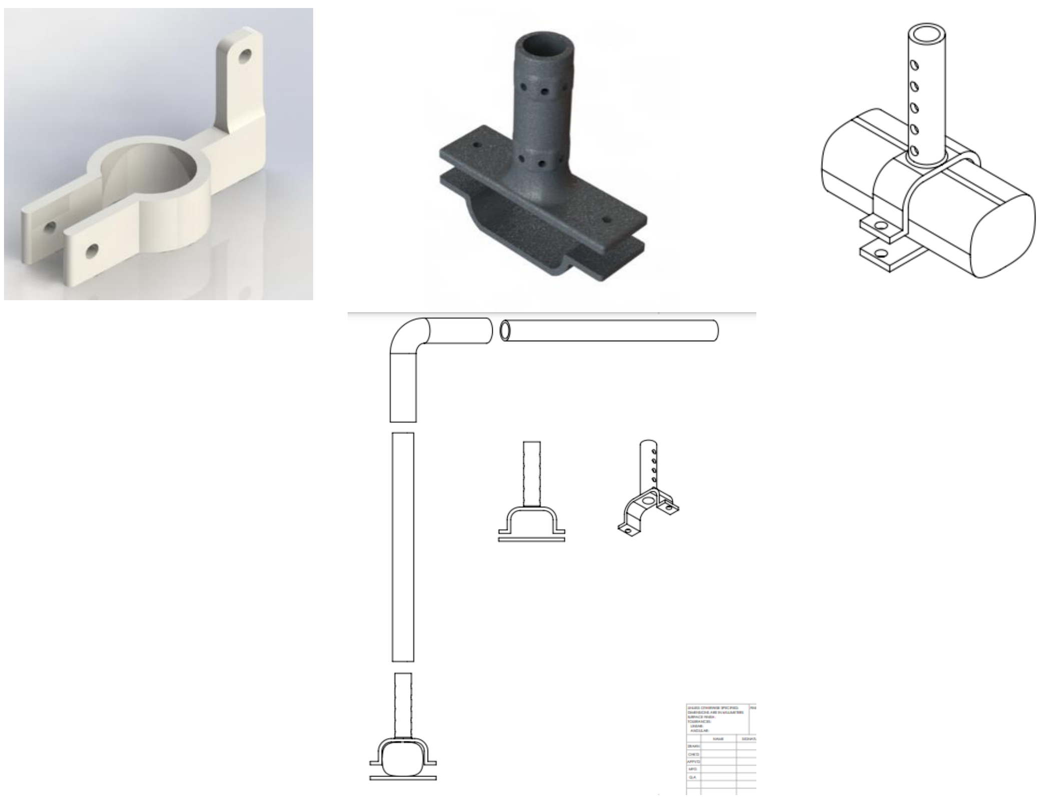
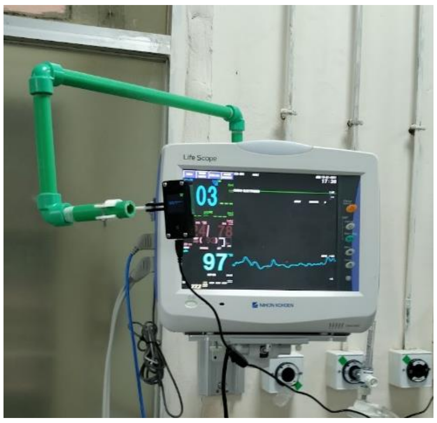


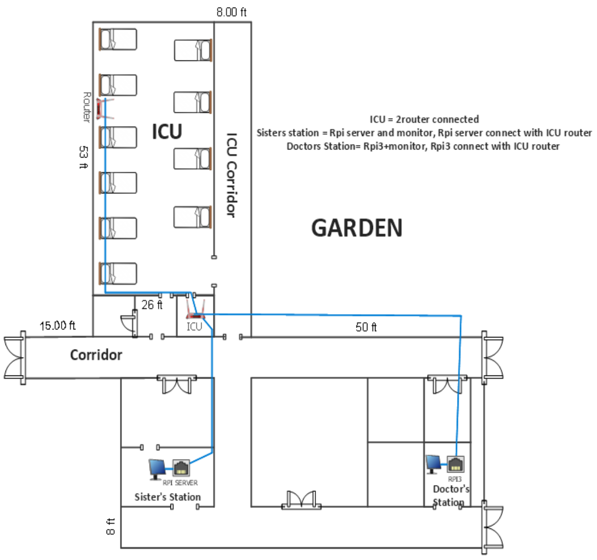
Figure 4-7 Clamp Design

4.8 IMPLEMENTATION IN CGH

To reduce the estimation cost we create a design structure with PVC pipe. The whole connection is connected under a local server. The network will cover the whole outside area. Thus doctors can easily access it from their meeting room.



(a) Inside ICU room



(c) Floor Plan

(b) Doctors Room

Figure 4-8 Project implementation in Chittagong General Hospital

CHAPTER 5

5. CONCLUSION

Our central monitoring system helps hospital staff to continue monitoring the patient from the safe space. It will show the continuous image of the bedside monitor and track the monitor data. Any certain change in patients' health means if there will be any change in abnormal monitor data, our system will alarm the duty doctors or nurses. Again, distance monitoring will help hospital personnel safe from infectious diseases like covid-19.

We have done our project to help the doctor, nurses, and other staff of ICU from exposure to the virus. In this covid-19 situation, many doctors and hospital personnel are dead due to the virus exposure. Every staff of the hospital needs to monitor the patient. So, our project will be helpful for them.

5.1 FUTURE SCOPE

Though our main target is continuously showing the bedside image to a non-covid space, there are still some room for improvement. Some are:

1. Our whole system design is under LAN. Thus, hospital staff whose are not connected with our network cannot access this feature. So, remote network establishment is very important. But there will happen same issue. The main problem would be the HTTP posting to the remote server.
2. There will no manually adding patient monitor system in GUI. It will be possible to design develop a system that hospital staff can easily adding or removing more bed system. Thus, they will get more benefit.

Many projects can be done using our project algorithm. e.g., the number plate detection system. Our image segmentation and OCR system can easily extract data from the number plate. Besides, Our Straightened image algorithm helps to develop scanner app. Where user can easily straight and crop or cut their image.

6. BIBLIOGRAPHY

[1] "PPE-Assumptions," [Online]. Available: https://www.centerforhealthsecurity.org/resources/COVID-19/PPE/PPE-assumptions. [Accessed 18 03 2022].

[2] "Life Scope VS BSM-3000 series," [Online]. Available: https://ae.nihonkohden.com/en/products/patientmonitoring/bsm3000.html. [Accessed 17 03 2022].

[3] "Digital image processing," [Online]. Available: https://en.wikipedia.org/wiki/Digital_image_processing. [Accessed 18 03 2022].

[4] "About," [Online]. Available: https://opencv.org/about/. [Accessed 03 18 2022].

[5] "OpenCV," [Online]. Available: https://en.wikipedia.org/wiki/OpenCV. [Accessed 03 18 2022].

[6] "Canny edge detector," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector. [Accessed 11 03 2022].

[7] "Canny Edge Detection," OpenCV, [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html. [Accessed 11 03 2022].

[8] "Histograms - 2: Histogram Equalization," [Online]. Available: https://docs.opencv.org/3.4/d5/daf/tutorial_py_histogram_equalization.html. [Accessed 18 03 2022].

[9] "Image Thresholding," [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html. [Accessed 18 03 2022].

[10] R. C. Gonzalez, Digital image processing, Pearson education India, 2009.

[11] "2D Convolution (Image Filtering),Image Blurring(Image Smoothing)," [Online]. Available:

https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. [Accessed 18 03 2022].

[12] S. Paris, "A gentle introduction to bilateral filtering and its applications," in *ACM SIGGRAPH 2007 courses*, 2007, pp. 3--es.

[13] "OpenCV-Python Tutorials," [Online]. Available: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. [Accessed 18 03 2022].

[14] "Morphological Transformations," [Online]. Available: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html#:~:text=Morphological%20transformations%20are%20some%20simple,decides%20the%20nature%20of%20operation.. [Accessed 18 03 2022].

[15] "Raspberry Pi 3 Model B," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/. [Accessed 22 03 2022].

[16] "Raspberry Pi 4," [Online]. Available: https://www.raspberrypi.com/news/raspberry-pi-4-on-sale-now-from-35/. [Accessed 22 03 2022].

[17] "ESP32," [Online]. Available: https://www.espressif.com/en/products/socs/esp32. [Accessed 22 03 2022].

[18] "LILYGO® TTGO T-Camera Plus ESP32-DOWDQ6 8MB SPRAM OV2640 Camera Module 1.3 Inch Display With WiFi Bluetooth Board - Normal," LILYGO, [Online]. Available: https://www.banggood.in/LILYGO-TTGO-T-Camera-Plus-ESP32-DOWDQ6-8MB-SPRAM-OV2640-Camera-Module-1_3-Inch-Display-With-WiFi-bluetooth-Board-p-1426498.html?cur_warehouse=CN. [Accessed 22 03 2022].

[19] "LILYGO® TTGO T-Camera Plus ESP32-DOWDQ6 8MB SPRAM Camera Module OV2640 1.3 Inch Display Rear Camera," LILYGO, [Online]. Available: http://www.lilygo.cn/prod_view.aspx?TypeId=50067&Id=1272&FId=t3:50067:3. [Accessed 22 03 2022].

[20] "Flask Web Development," [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/. [Accessed 22 03 2022].

[21] "Flask (web framework)," [Online]. Available: https://en.wikipedia.org/wiki/Flask_(web_framework). [Accessed 22 03 2022].

[22] "sqlite3 — DB-API 2.0 interface for SQLite databases," [Online]. Available: https://docs.python.org/3/library/sqlite3.html. [Accessed 22 03 2022].

[23] "SQLite," [Online]. Available: https://en.wikipedia.org/wiki/SQLite. [Accessed 22 03 2022].

[24] "Appropriate Uses For SQLite," [Online]. Available: https://www.sqlite.org/whentouse.html. [Accessed 22 03 2022].

[25] "Flutter," [Online]. Available: https://flutter.dev/. [Accessed 22 03 2022].

[26] "Digital image processing," [Online]. Available: https://en.wikipedia.org/wiki/Digital_image_processing. [Accessed 11 03 2022].

[27] "ESP32-CAM Post Images to Local or Cloud Server using PHP (Photo Manager)," 2021. [Online]. Available: https://randomnerdtutorials.com/esp32-cam-post-image-photo-server/.

[28] "National Health and Safety ," [Online]. Available: https://www.chippingnortonhealthcentre.nhs.uk/conditions/intensive-care/. [Accessed 2022 03 15].

[29] "Python Flask Web Application on Raspberry Pi with NGINX and uWSGI," [Online]. Available: https://iotbytes.wordpress.com/python-flask-web-application-on-raspberry-pi-with-nginx-and-uwsgi/. [Accessed 22 03 2022].