Centralized Monitoring System for Covid ICU at CGH

Presented by:
MD. Hedayetul Islam Chy.
ID No. : 17702013
Dept. of EEE, CU

Supervised by:
Dr. Mohammed Arif Iftakher Mahmood
Associate Professor
Dept. of EEE,CU

Outline

- Introduction
- Current status
- Motivation
- Methodology
- System Architecture
- Backend-work
  - Image Acquisition
  - Data Analysis
  - Server Process
  - Straightened Image
  - Segmentation
    - Color-based
    - Pixel-based
  - OCR
- 3D design
- Implementation at CGH
- Application
- Conclusion
- Future work

Introduction

- Inside covid-ICU, Hospital personnel has to stay close to the covid infected patients for continuous checkups.
- Bedside monitor in ICU shows data of
  - Heart rate,
  - Oxygen saturation,
  - blood pressure, etc. of a patient.
- Doctors and hospital staff must need to monitor this information consecutively.
- Due to covid, barely possible to inspect the patient in this condition.
- Lack of PPE and other protective equipment systems made the situation worse.

Current status

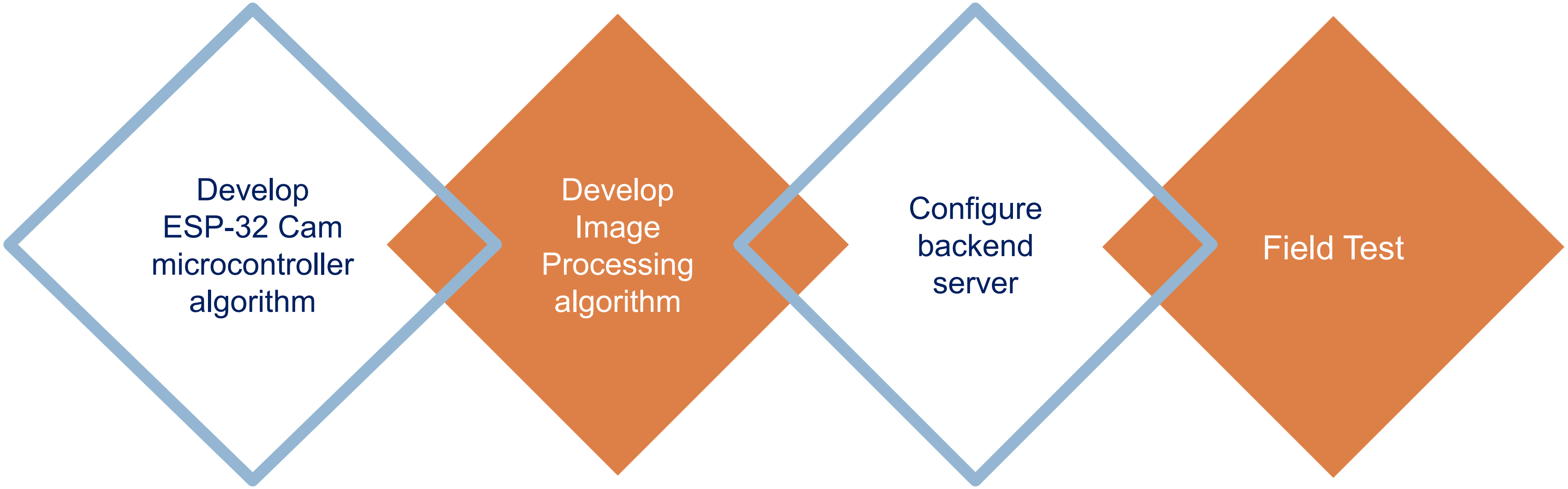
- Waste of Protective equipment
- Need to pay for central monitoring software and licensing (CMSL).
  - e.g. In CGH, for setup a 10 bed ICU CMSL fee of **~17,000 USD**
- Can’t take data from the outgoing sensor signal because-
  - No right to access those devices
  - Again need to implement a new infrastructure

Motivation

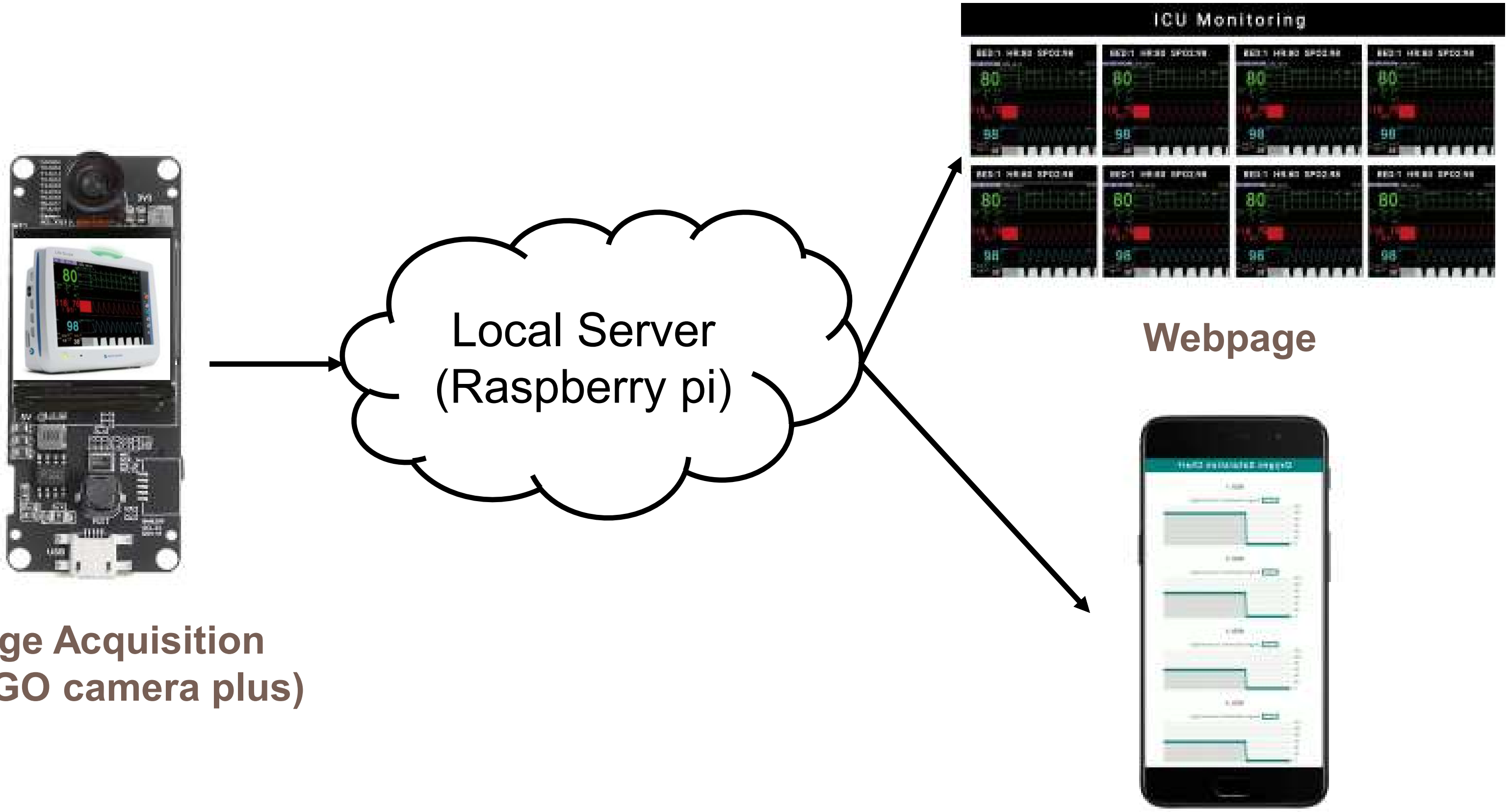
- Reduce virus exposure among the hospital personnel’s
- A system that allows images and data from ICU bedside monitor to be displayed on a web page
- Alert if the patient’s condition is critical
- A fully wireless system
- Easily deployable to any ICU
- Economical and easy to use

How?

- Develop a local server that shows all bedside monitors image on the webpage and reduce contact with covid infected patients and hospital personnel’s


Develop ESP-32 Cam microcontroller algorithm
Develop Image Processing algorithm
Configure backend server
Field Test

Methodology



Local Server (Raspberry pi)

Webpage

Image Acquisition (TTGO camera plus)

Android Application

System Architecture

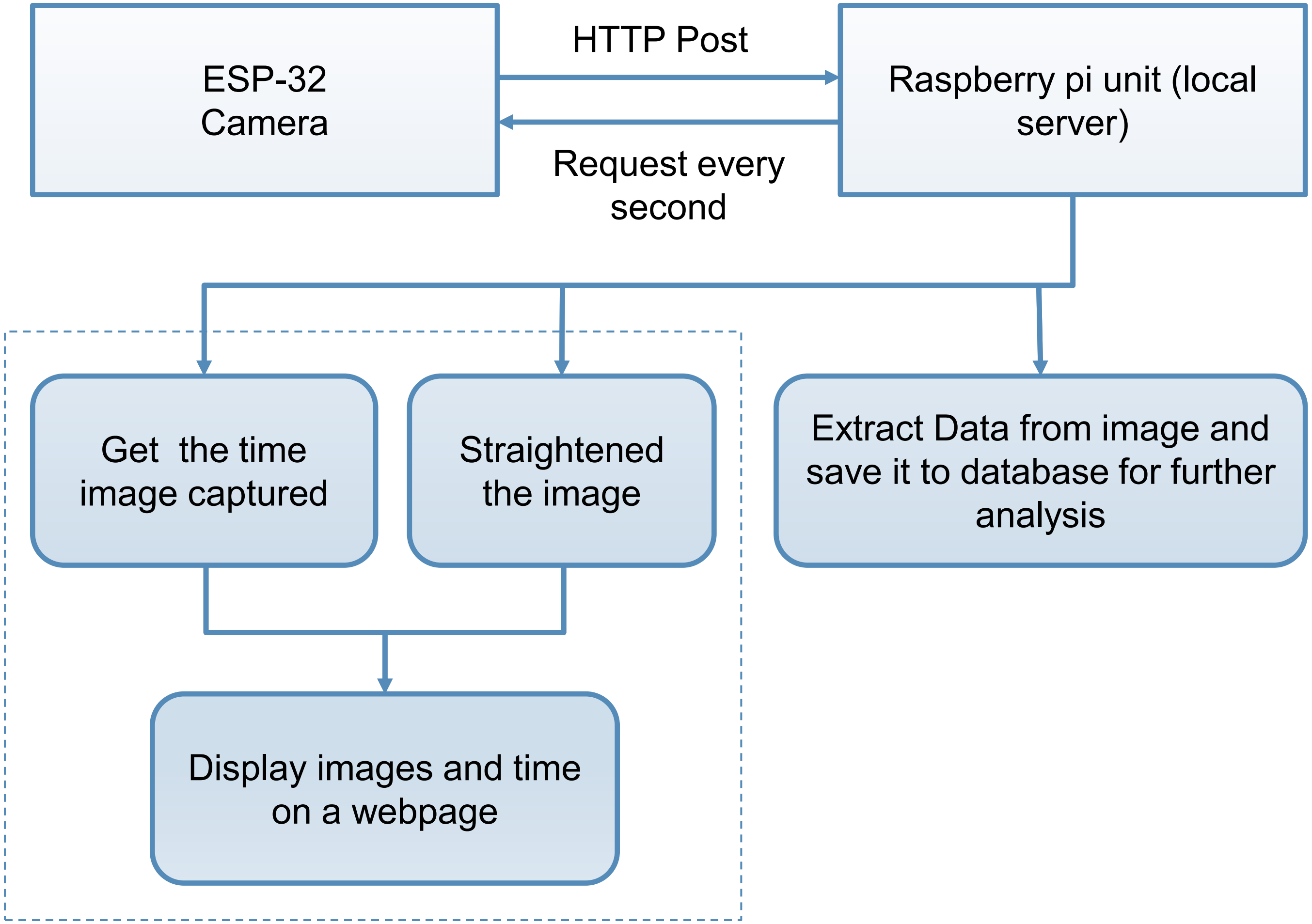

ESP-32
Camera
HTTP Post
Request every second
Raspberry pi unit (local server)
Get the time image captured
Straightened the image
Extract Data from image and save it to database for further analysis
Display images and time on a webpage

Figure: Overview of the system

Entire process

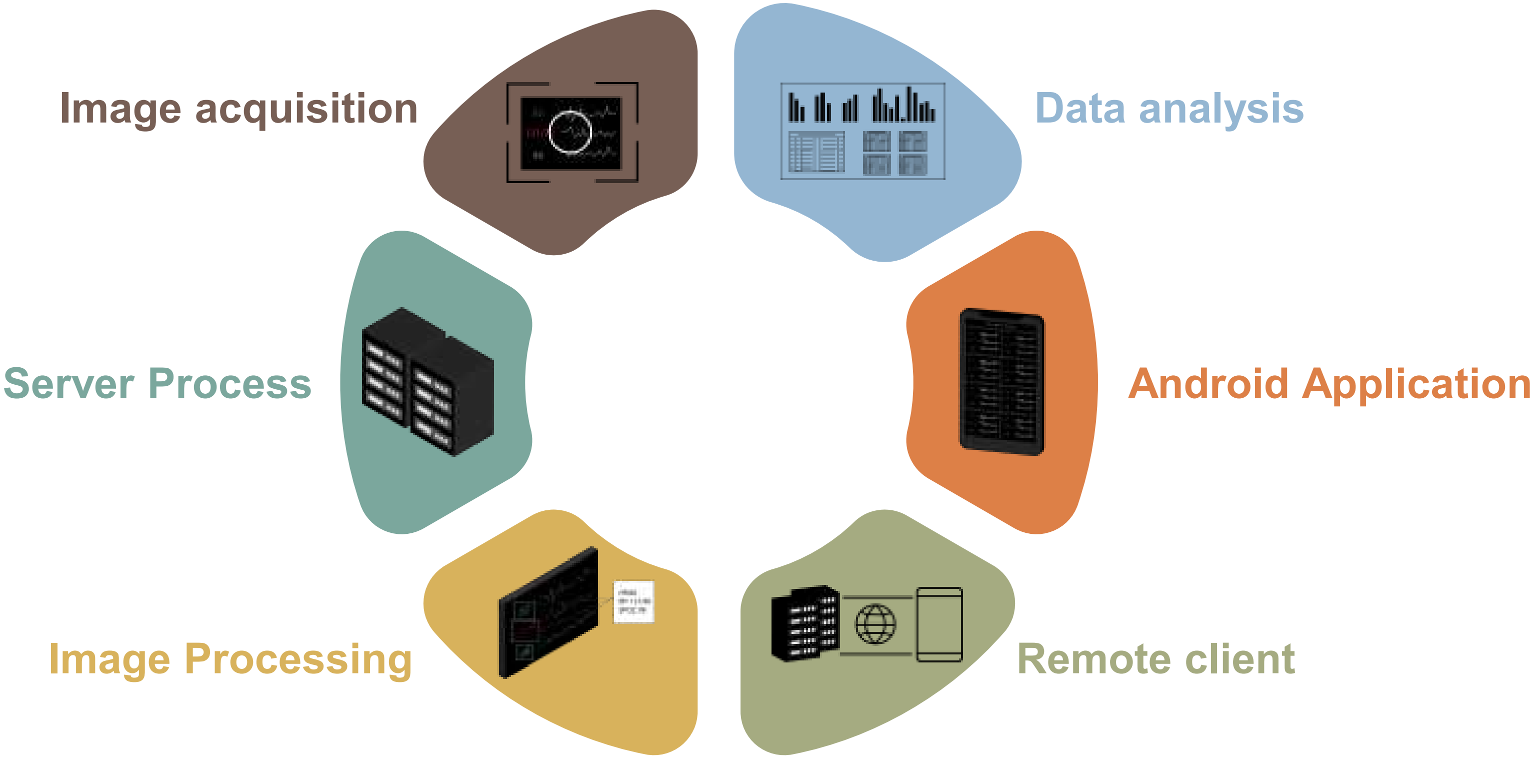


Image acquisition

Data analysis

Server Process

Android Application

Image Processing

Remote client

Backend: Image Acquisition

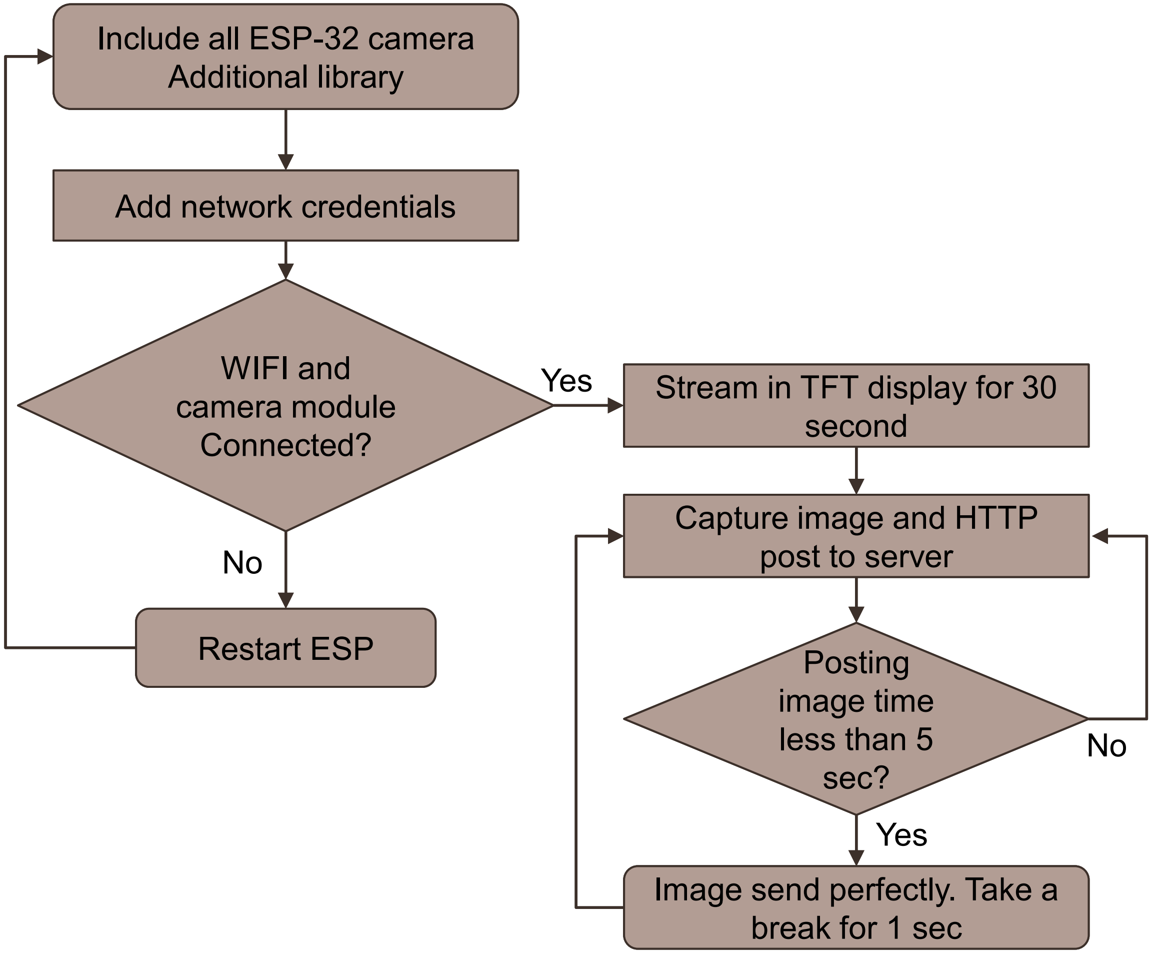
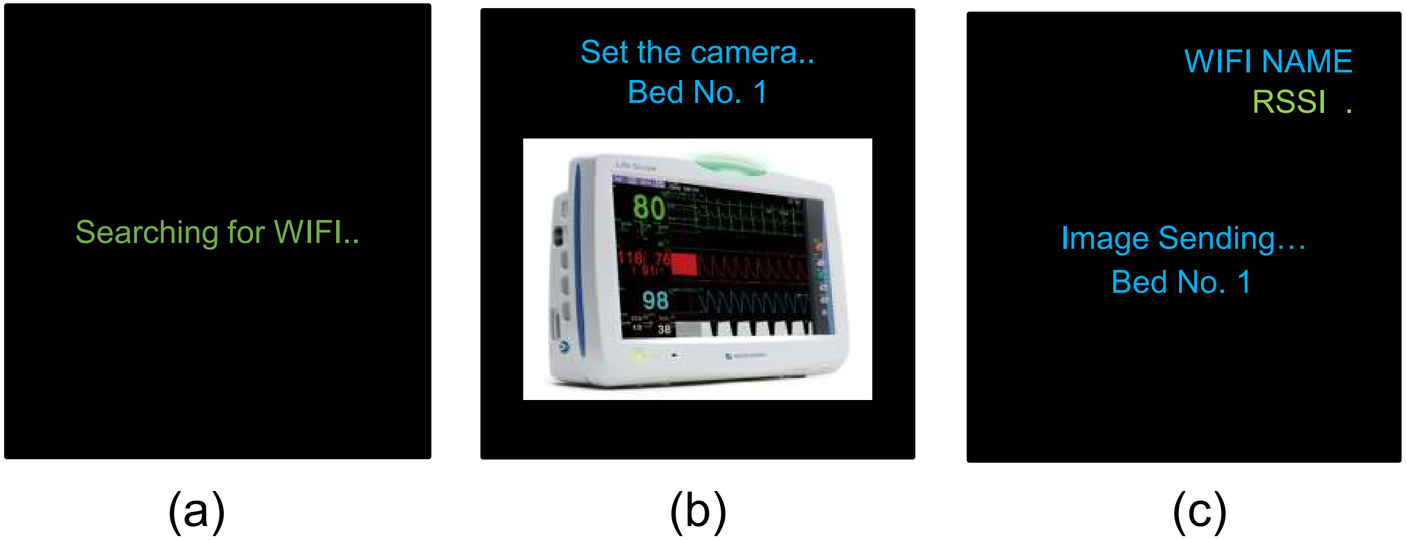

Include all ESP-32 camera Additional library
Add network credentials
WIFI and camera module Connected?
Yes
No
Restart ESP
Stream in TFT display for 30 second
Capture image and HTTP post to server
Posting image time less than 5 sec?
No
Yes
Image send perfectly. Take a break for 1 sec

Fig: ESP-32 cam (TTGO camera Plus)

Figure HTTP image post using Esp-32 cam flowchart

Backend: Image Acquisition



Searching for WIFI..

(a)

Set the camera..
Bed No. 1

(b)

WIFI NAME
RSSI .

Image Sending…
Bed No. 1

(c)

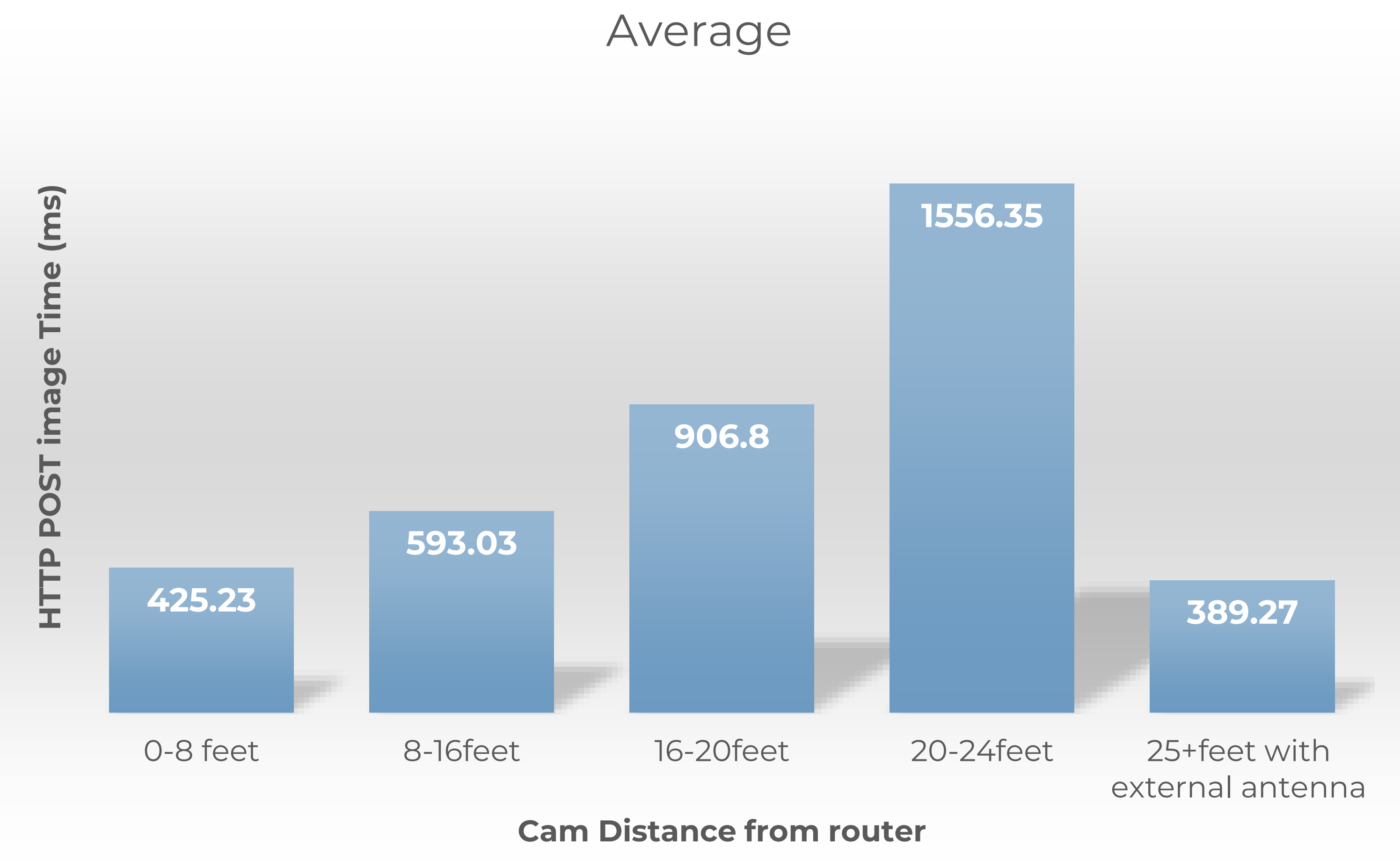
Figure: TTGO camera plus display (a)Searching network to connect, (b)Display camera for 30 seconds, (c)Start sending image to server after 30 sec.

Esp-32 cam: Data analysis

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

- Without antenna ~15,000 image
- With antenna ~26,000 image

Data analysis chart



Backend: Server process

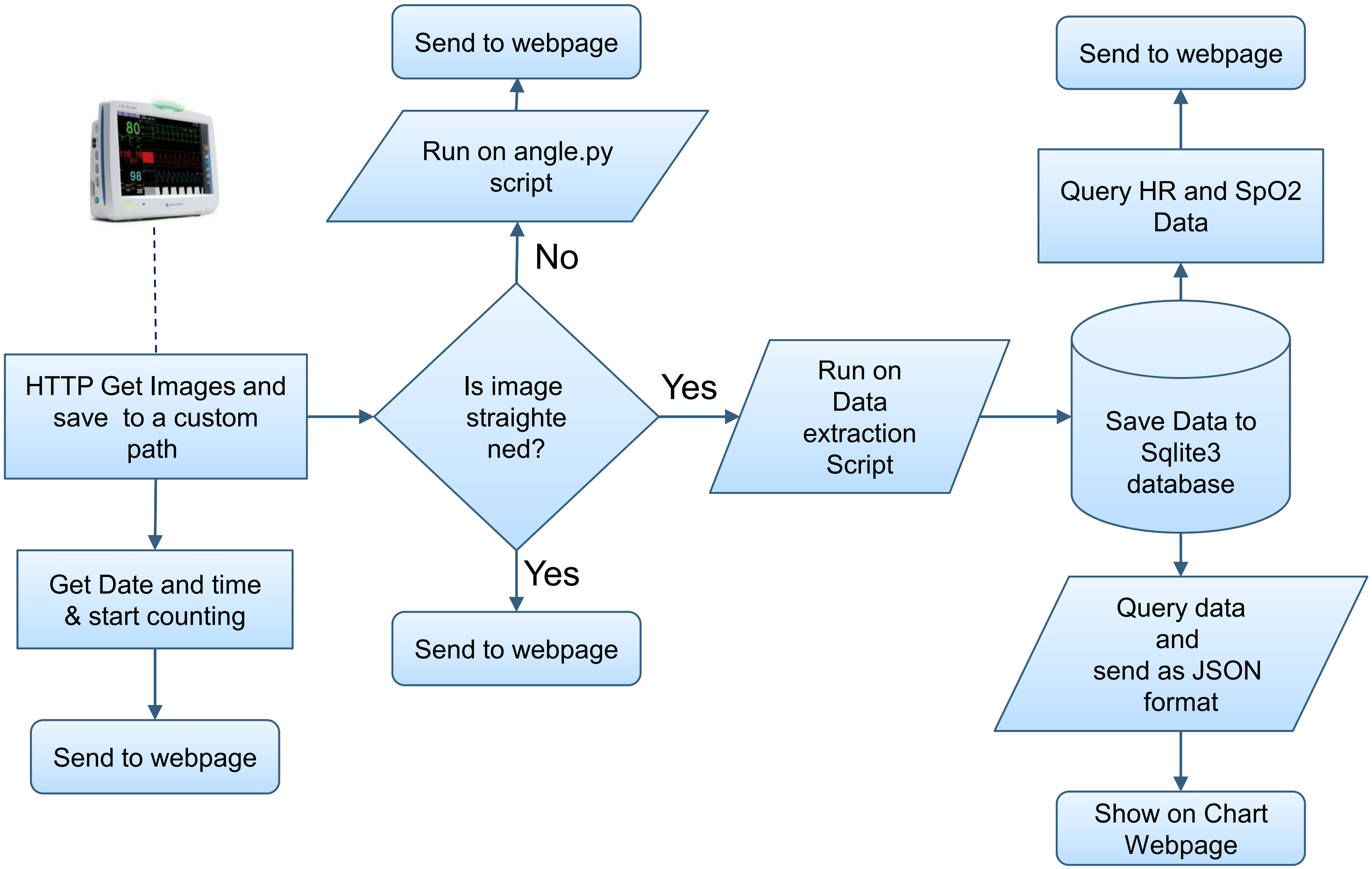
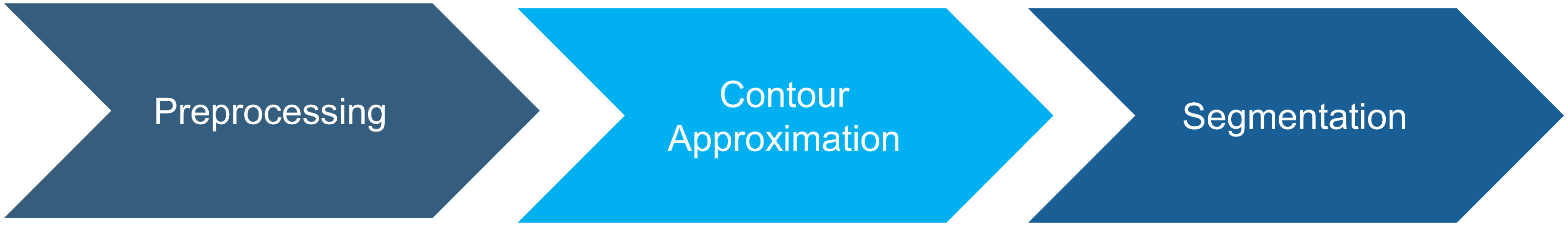

Send to webpage
Run on angle.py script
No
HTTP Get Images and save to a custom path
Is image straighte ned?
Yes
Run on Data extraction Script
Yes
Send to webpage
Get Date and time & start counting
Send to webpage
Send to webpage
Query HR and SpO2 Data
Save Data to Sqlite3 database
Query data and send as JSON format
Show on Chart Webpage

Figure: Flask Server flowchart

Backend: Straightened Image

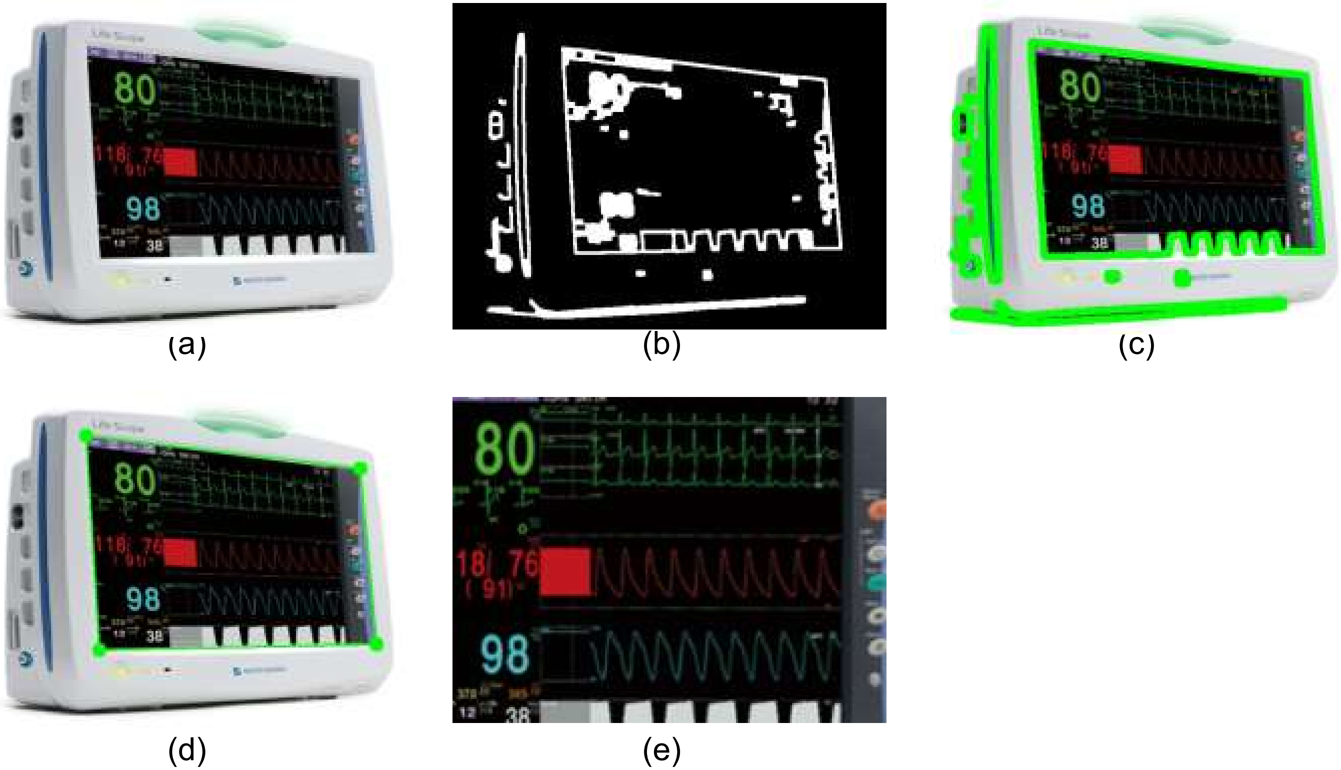


- Resize Image
- Grayscale Conversion
- Apply gaussian blur/ median blur
- Binary conversion

- Canny Edge Detection
- Morphological transformation
    - Erosion
    - Dilation
- Calculate the biggest contour
- Contour Approximation

- Retransform the Rectangle biggest contour
- Resize image

Backend: Straightened image



(a)

(b)

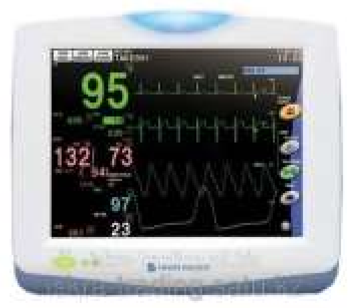
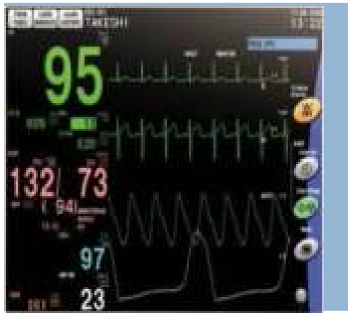
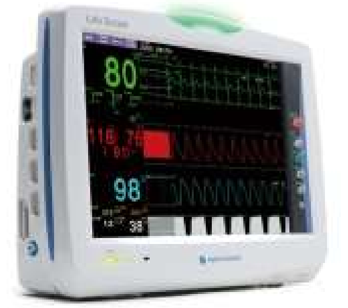
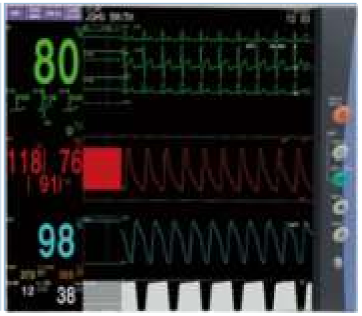
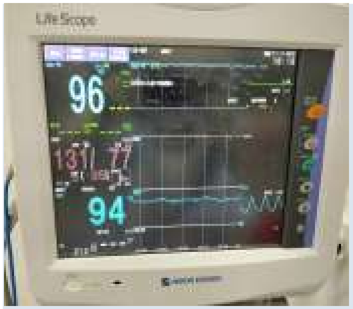
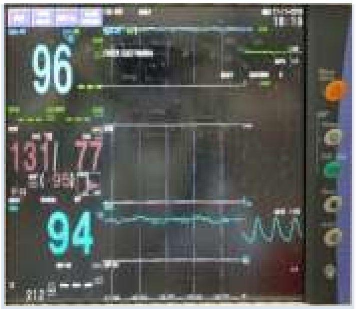
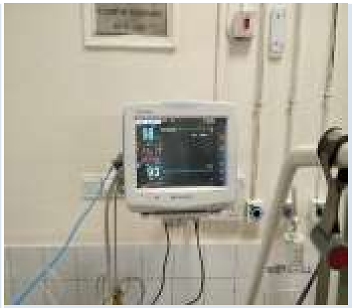
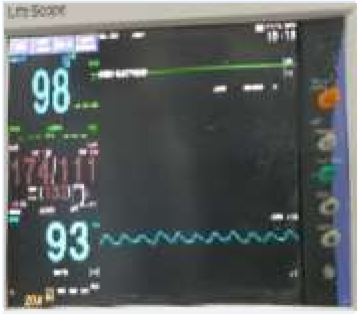
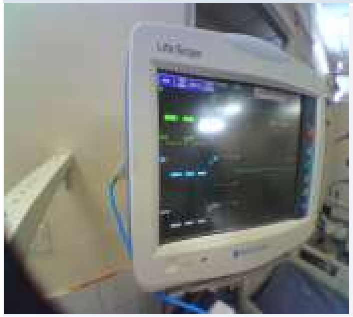
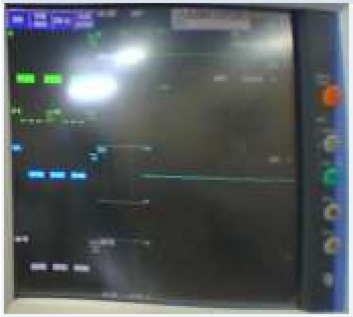
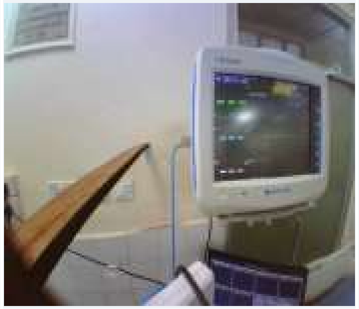
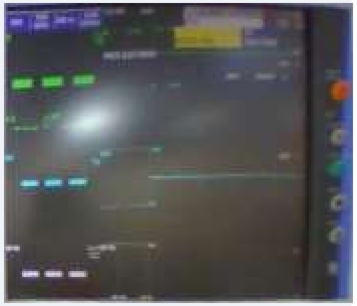
(c)

(d)

(e)

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

Visual performance of Straightened image

Input Image	Straightened Image	Input Image	Straightened Image
			
			
			

Backend: Segmentation (Pixel-based)

Pixel-based Segmentation

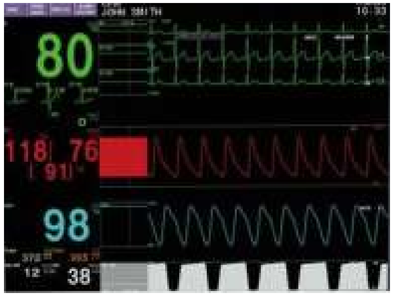
Preprocessing

- Resize Image
- BGR to Gray conversion
- Thresholding operation
- Blur filter

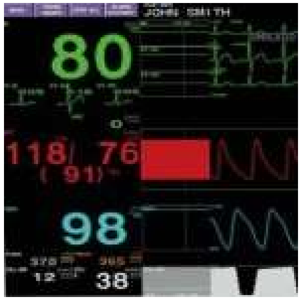
Contour Approximation

- Morphological Transformation
- Find Contours
- Remove Small pixel

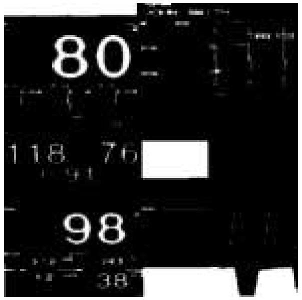
Pixel-Based



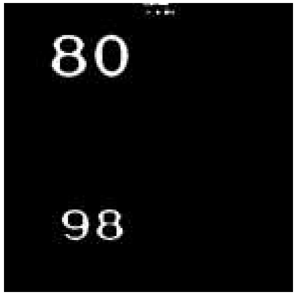
(a)Original Image



(b)Crop and resized image



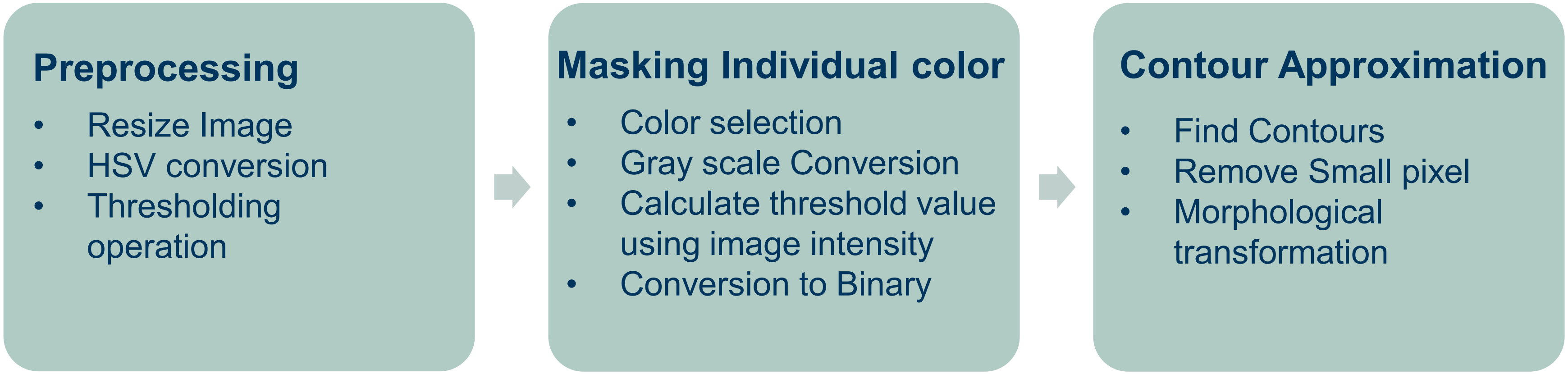
(c) Threshold image



(d) Final Image

Backend: Segmentation (Color-based)

Color-based Segmentation



Preprocessing

- Resize Image
- HSV conversion
- Thresholding operation

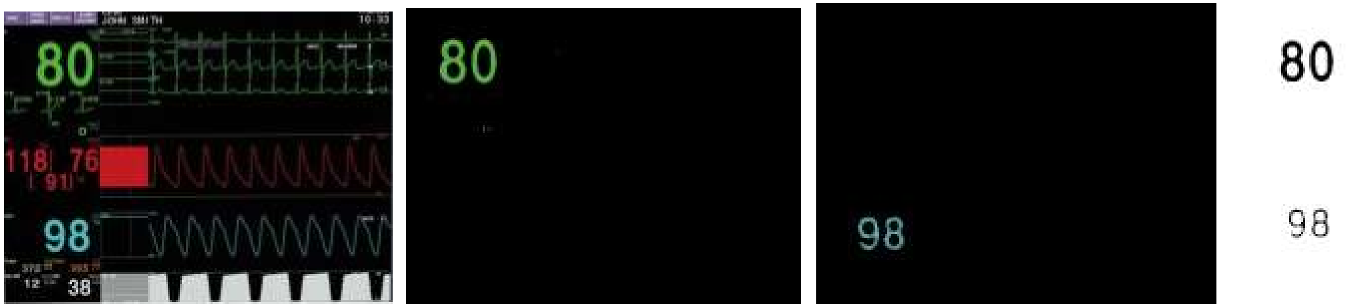
Masking Individual color

- Color selection
- Gray scale Conversion
- Calculate threshold value using image intensity
- Conversion to Binary

Contour Approximation

- Find Contours
- Remove Small pixel
- Morphological transformation

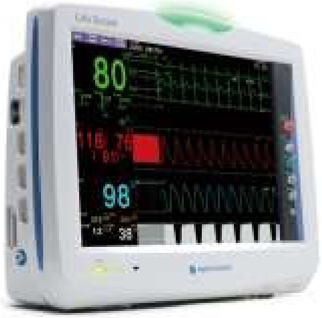
Color-Based



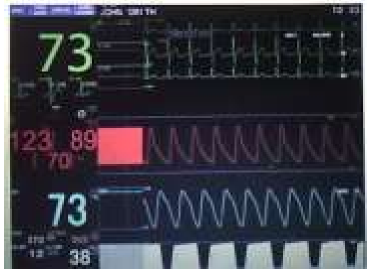
(a)Original Image (b)Masking green (c) Masking Blue (d) Final Image

OCR

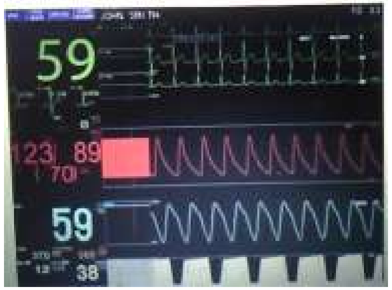
- Using pytesseract library to get the list of data from images



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



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



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

Web Interface

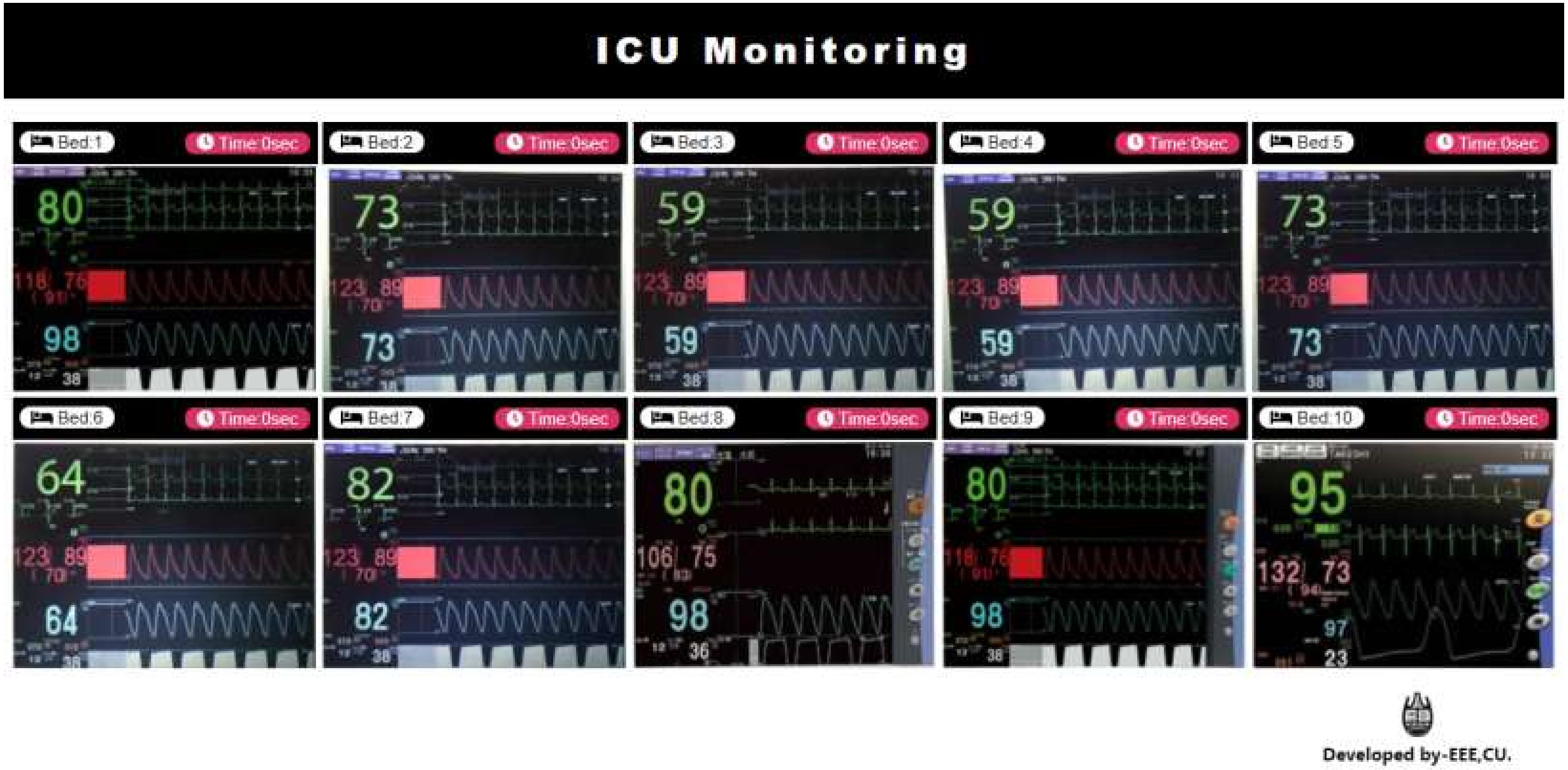
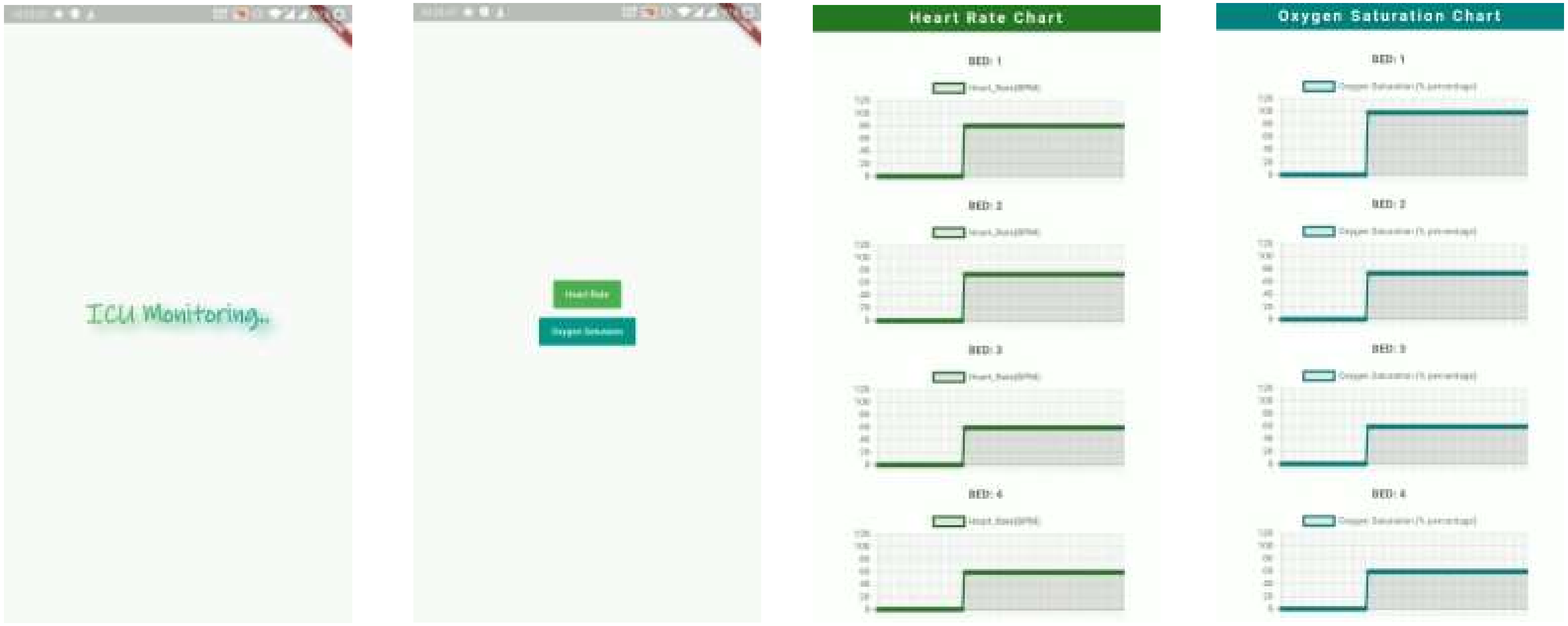


Figure: Web interface (Counting time)

Android App Interface



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

Figure: (a)Intro (b)Choose section, (c)Heart rate interface, (d)Oxygen Saturation Interface

3D design

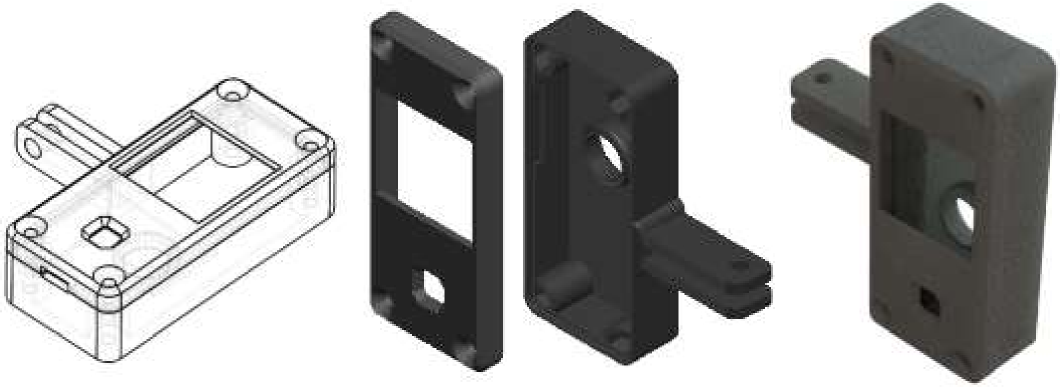
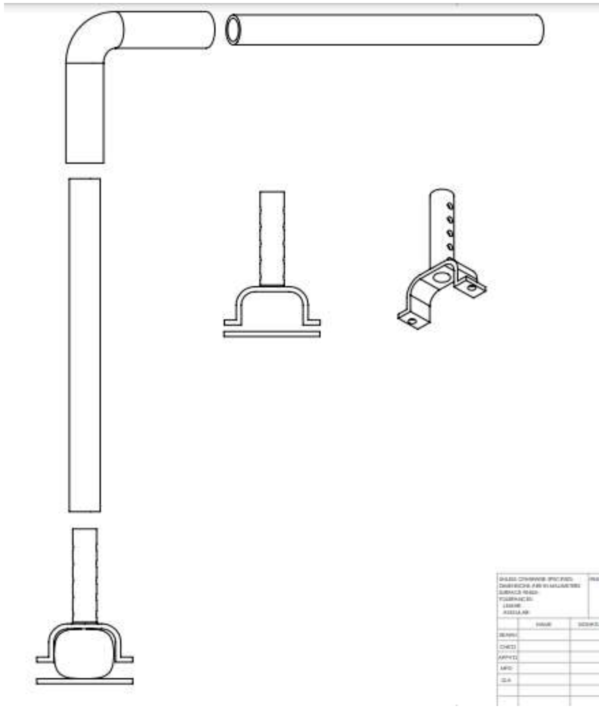


Figure: TTGO Camera Case



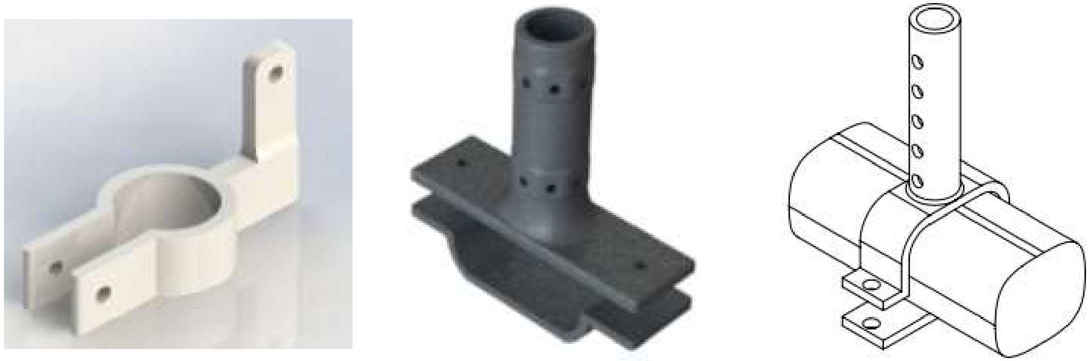
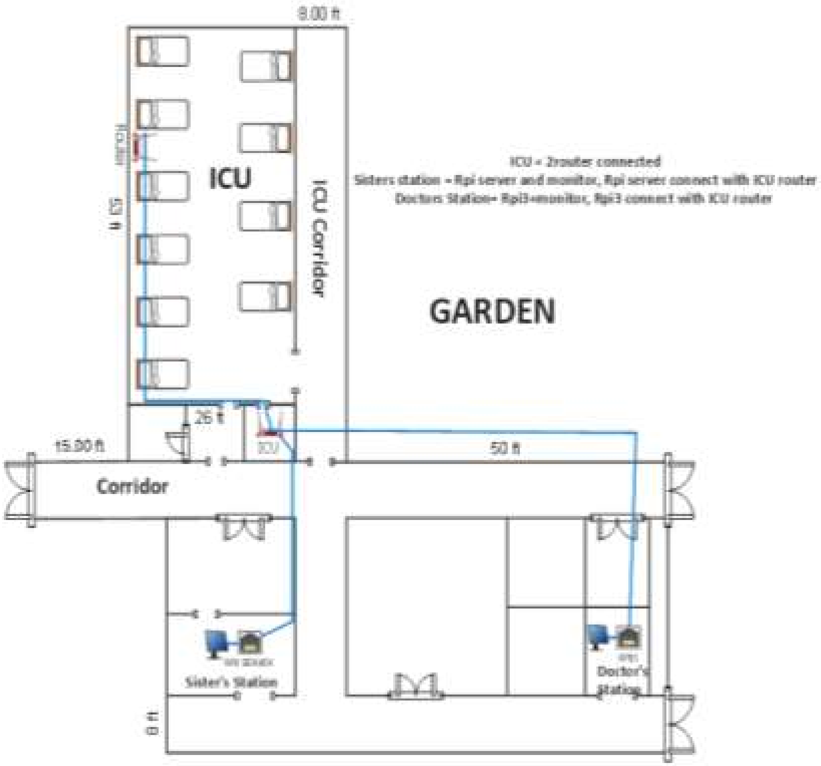


Figure: Clamp Design

Implementation at CGH



ICU - 2router connected
Sisters station - Rpi server and monitor, Rpi server connect with ICU router
Doctors Station- Rpi3+monitor, Rpi3 connect with ICU router

Floor plan of CGH covid ICU

Inside ICU room

Doctors Room*

** Image with Doctor Moumita Das, Junior Consultant, CGH*

Application

- Plug and play system, easy to operate.
- Can be operated without the help of an expert.
- Well suited for Covid-ICU to monitor their patients' conditions 24/7.

Conclusion

- Hospital personnel can monitor patients’ status from non-covid space.
- Wireless system
- Unconditional changes of the patient will warn via the system.

Future Work

- Remotely usable for the client using MQTT protocol
- Secure dynamic web hosting
- Creating pyGUI

Thank You!

For Your Attention