A Project Report on

Health Monitor

submitted in partial fulfillment of the requirements for the degree of

B. Tech

IN

ELECTRONICS AND CONTROL SYSTEM

BY

RAGHIB SHAMS

SUBHANKAR SAMANTA

RAHIL NASIR

Under the guidance of

Prof. NAGESWARA RAO BUDIPI

SCHOOL OF ELECTRONICS ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

(Deemed to be University)

BHUBANESHWAR

NOVEMBER 2021

ACKNOWLEDGEMENTS

We would like to convey our heartfelt thanks to Prof. NAGESWARA RAO BUDIPI for his invaluable direction and support in the development of our project work ideas. Without the help of our renowned supervisor and teaching colleagues from the School of Electronics, we would not have been able to complete this project. Their efforts are much appreciated and heartily recognised, and our group would like to use this occasion to convey our heartfelt gratitude and debt to them all.

Prof. NAGESWARA RAO BUDIPI has been really helpful in slowly and thoroughly editing all of our articles and evaluating any other documents whenever we have approached him.

ABSTRACT

Pulse oximeters are frequently used to determine blood oxygen saturation in a non-invasive manner. The motive of this project is to design and develop a affordable ppg based health monitoring kit to investigate the precise oxygen saturation level and pulse rate to predict the patients' health state using machine learning. SpO2 levels in healthy adults range from 95 to 100 percent and the heart rate ranges from . The pulse oximeter will be able to gather SpO2 and heart rate data with the help of the MAX30100 module. Using the Blynk program and the ESP32 Wi-Fi and Bluetooth module, this design monitors patient data and feed it on the cloud platform. Cloud-based machine learning techniques will be used to forecast the patient's health.

The various feeds of the health parameters would also be monitored, logged and analyzed using IOT Could Platforms like Blynk IoT app .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

PPG : Photoplethysmography

ECG : Electrocardiogram

IR : Infrared

I^2C : Inter Integrated Circuit

BPM : Beats per minute

SpO2 : Peripheral capillary oxygen saturation

MCU : Micro-Controller Unit

SOC : System on chip

OLED: Organic light emitting diode

ML : Machine learning

HTTP: Hypertext Tranfer Protocol

HTML: Hypertext Markup Language

TCP : Transmission Control Protocol

IoT : Internet of Things

CHAPTER 1

INTRODUCTION

The Project is basically a health monitor presently based on photoplethysmography for measurement of health parameters like pulse rate heart rate

1.1 Introduction to pulse oximeter:

Pulse oximeter is a gadget that works on the pulse oximetry concept. Pulse oximetry is a technique for measuring people's oxygen saturation (SpO2) without causing damage to their bodies. The normal level of SpO2 is between 95 and 100 percent. The level of SpO2 is mostly determined by the R (ratio) value. The R value is the ratio of red and infrared light. Red has a wavelength of 660nm, whereas IR (infrared) has a wavelength of 940nm. The pulse oximeter determines whether or not a person's blood oxygen level is within a healthy range. Hypoxemia occurs when your oxygen level falls below the usual range.

1.2 Problem Description:

The objective of this study through our project was to develop determine whether a machine learning predictive model could be trained on a set of clinical data and weather will it can be used to predict transcutaneous hemoglobin oxygen($Sp0_2$) level of a patient and by judging all the acquired values predict and show the levels for the past days.

1.3 Issues and Challenges:

- **Calibration and filtering of data asper different categories of health conditions :** Couple of initial of data for BPM measure were error so had to be neglected by the MCU itself .
- **Sensor error :** Although the sensor accuracy wasn't very high but was good enough to get health conditions and obtain logs for analysis. The sensor precision in case of Pulse rate was an issue and really challenging to be made stable.

- **I2C Address determination :** The I2C address specially for OLED wasn't well documented mainly as it had many variants and ours was purchased from the local market . Therefore a special c++ program had to be run connecting the I2C device through the bus of the mcu.
- **I2C pull configuration :** The I2C pull up for the two devices was very well configured and therefore had to be configured in custom to make communication more stable at the device logic level .

  The regular available module of Max30100 had the I2C pull-up resistors connected to the onboard LDO of 1.8V which made it difficult to establish communication with general MCUs having logic voltage level above 1.8V (3.3 in Esp32) to fix this communication problem the 4.7k resistors' copper track on the PCB , connected to 1.8V LDO had to be etched and soldered to the output terminal of the 3.3V LDO . As shown in the figure below .

Figure 1

- **Exporting SPO2 and BPM data for ML data analysis in Python :** Although data could be logged and saved in csv using UART serial communication of PuTTY terminal as well as the Blynk cloud but a perfect format to simply analyze data from csv couldn't to attained . Hence some filtering , dividing of data had to be done using python tools itself .

CHAPTER 2

PROJECT COMPONENTS

This chapter basically provides the background information regarding the components which we are using in our project. Starting with IoT network Protocols we have :

2.1 IoT Protocols :

WiFi 802.11 b/g/n :

IEEE 802.11 is a set of MAC and PHY protocols for establishing WLAN computer communication and is part of the IEEE 802 series of LAN technical standards. The standard and amendments form the foundation for wifi-branded wireless network equipment and are the most extensively utilised wireless computer networking protocols in the world. Esp32 runs WiFi at 2.4GHz giving it compatibility with all WiFi bands of standard b, g and n .

HTTP :

Hypertext Transfer Protocol (HTTP) is the application layer of TCP Internet Protocol for transmission of hypermedia documents such as HTML . Designed mainly for communication between server and browsers but works for other applications as well . This protocol works on server - client concept working with request and response .

2.2 IOT SOC Dev. Board :

Wifi, BLE, and Lora modules for IoT are divided into two categories:

- In a single chip, the MCU executes both the IoT network stack and the host application.
- A host processor + IOT module solution in which the wifi stack is contained in the wireless connectivity solution and the host application is operated by a separate processor.

ESP32:

Espressif Systems designed the ESP32 SOC chip. This SOC supports embedded devices with WiFi dual mode Bluetooth connection. Tensilica Xtensa LX6 microprocessors with a dual-core variant and a clock rate of over 240 MHz are used in the ESP32 chip.The soc features 2 I2C controllers and multi ADC on Gpios as well . The development board of Esp32 is based on Esp wroom 32 dev module which comes with a on module 4mb flash .

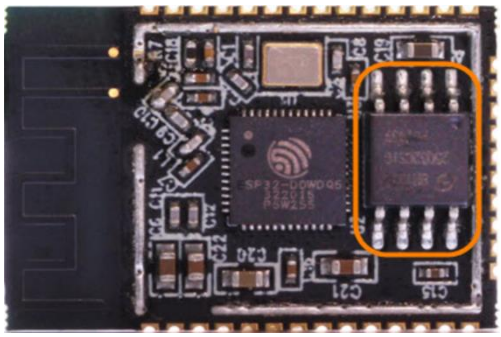


Figure 2.1

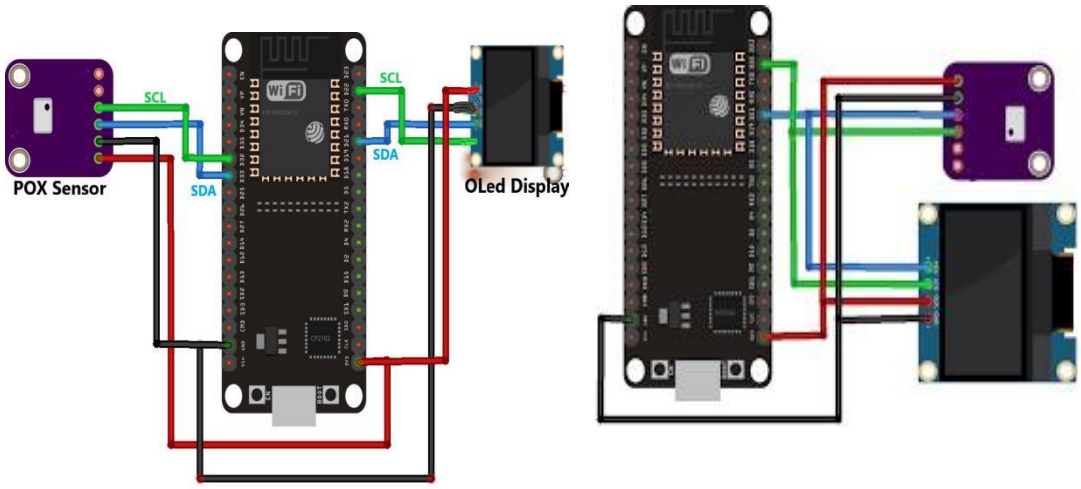
MCU Communication:

A multipoint Control Unit (MCU) is a device that is frequently used to connect video conferencing systems. The multipoint control unit is a LAN endpoint that allows three or more terminals and gateways to join a multipoint conference. A necessary multipoint controller (MC) plus an optional multipoint processor make up the MCU (MPs).

I2C Communication Protocol:

Inter-Integrated Circuit is the abbreviation for Inter-Integrated Circuit. It's a serial communication bus interface connection protocol that's built into devices. Philips Semiconductor created it in 1982. It has recently become popular for short-distance communication. Two Wired Interface is another name for it (TWI). There are two types of operation for I2C: master and slave.

For integrating the Pulse oximeter ic (with controller) with the Oled controller, we use I2C in one master multi-slave mode. We linked both our devices with separate I2C addresses in both the same I2C bus and the Multiple I2C bus interface as the ESP32 has two I2C controllers.



SAME I2C BUS **Different I2C BUS**

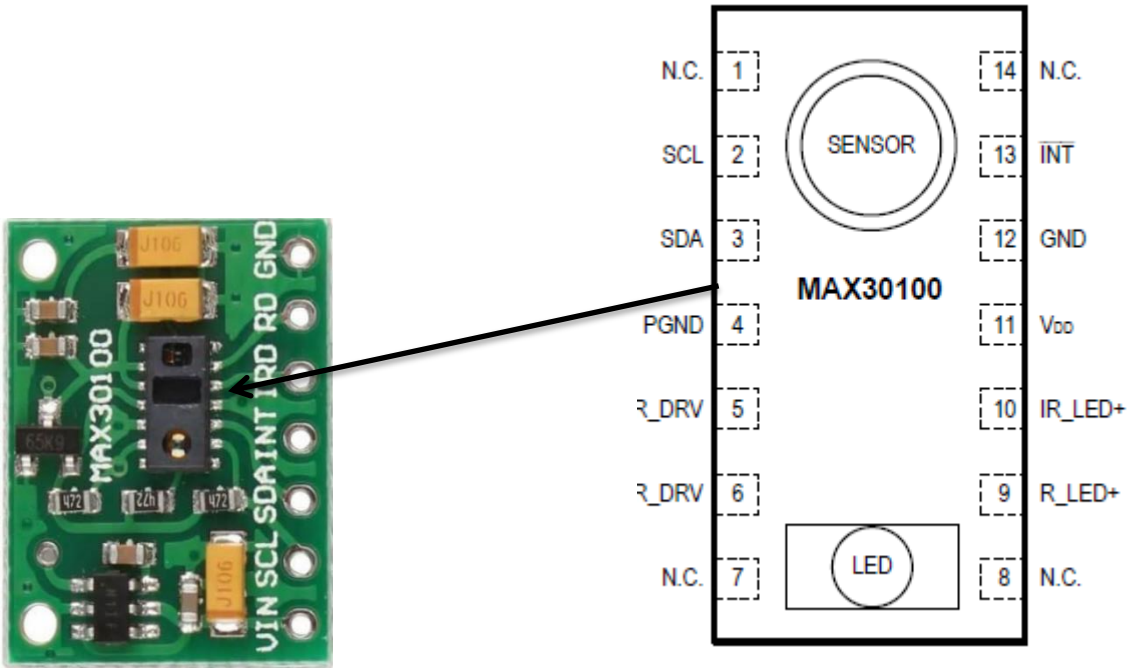
Fig 2.2 I^2C Configurations

2.3 SENSORS

A sensor is a device that detects changes in the environment and reacts to some other system's output. A sensor turns a physical phenomena into a measurable analogue voltage, which can then be displayed or communicated for further processing. It turns the physical activity to be measured into an electrical counterpart and then processes it so that electrical signals can be sent and processed quickly. The sensor can report whether or not an object is present (binary) or what measurement value was attained.**:**

MAX30100:

The MAX30100 is a sensor that combines pulse oximetry and a heart rate monitor. It detects pulse oximetry and heart rate signals using two LEDs, a photodetector, improved optics, and low noise analogue signal processing. The MAX30100 is powered by 1.8 V (for control and register unit) and 3.3 V (for LEDs and I2C communication with higher logic level MCU) power sources, and it may be turned off via software with a very low standby current, allowing the power supply to stay connected at all times.



Senor Module

Sensor IC

Fig 2.3 MAX30100

The main components of sensing in the IC are the red LED , IR LED , photo diode, ambient temperature sensor , ADC and the control and register units . The Red led here has Wavelength of about 650nm to 670nm and that of IR led is 870nm to 900nm , making the sensor suitable for measuring the absorption by blood in presense of Hb and HbO2 .

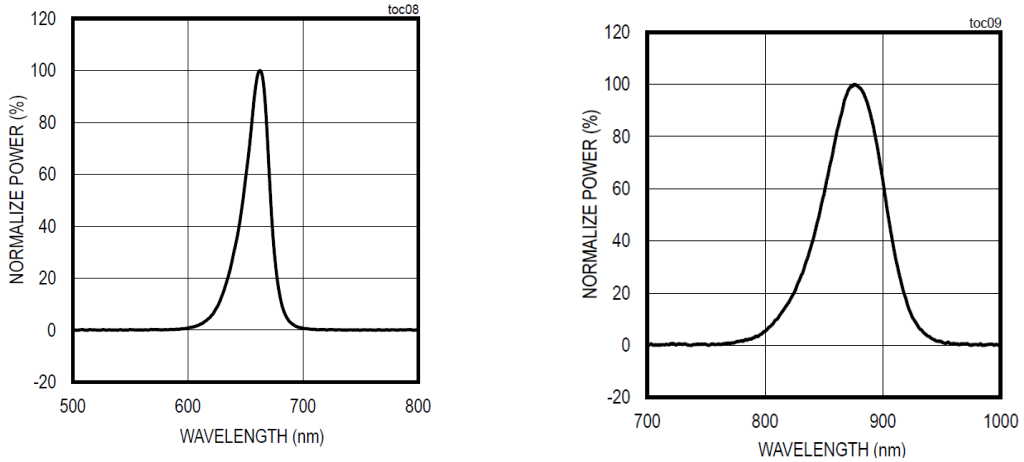


Fig 2.4

The on chip temperature sensor is meant for optional calibration of the SpO2 subsystem on the basis of ambient temperature change as the SpO2 calculation algorithm is critically correct to the interpretaion of the Red light but relatively insensitive to the IR lights wavelength .

The following are Software Programming Components :

2.4 Embedded C

It is a set of digital Machine level language extensions for C programming language set by the C standards committee to make C extensions compatible with various embedded devices . Here in our project we have used embedded c programming in a simplified way with arduino compatible libraries .

Libraries Used :

Arduino C/C++

I. **Wire :** An I2c interfacing Library with functions , syntax , registers and address calls of I2C
II. **Adafruit_SH110X :** Library by Adafruit for Oled display controller enabling it for selectable I2C and SPI interface with its register instructions and simplifying programming functions .
III. **Adafruit_gfx :** Library by Adafruit with set of functions and syntax for simplified graphics illustrations on Oled ,lcds and matrix displays .
IV. **MAX30100_PulseOximeter :** Maxim MAX30100 pulse oximetry chip library. This library is made to work with the MAXIM MAX30100 heart rate and oxygen saturation sensor chip. The library's current state allows for the reading of IR values for determining Heart Rate and the manipulation of any register.

2.5 Python :

Numpy : It is a Python library which provides a simple yet a powerful data structure: the n dimensional array. This is the foundation on which almost all the power of Python’s data science toolkit is built, and learning NumPy is also the first step towards learning and knowing Python. It basically stands for Numerical Python. It aims to provide an array object that is up to 50x faster than traditional Python lists.

Pandas :

Pandas is a NumPy-based open source library. It's a Python library with a variety of data structures and methods for handling numbers and data, as well as time series. It's most well-known for making data import and analysis significantly simpler. It is quick and provides users with a high level of performance and productivity. It is simple to load data from a variety of file objects. Missing data in both floating points and non-floating data can be handled easily. Columns can be inserted and deleted from DataFrame and higher dimensional objects, and size mutuality is also available. Data sets can be combined and connected, and data sets can be reshaped and pivoted. It also includes tidbits of information.

Matplotlib :

It's a fantastic Python visualisation toolkit for 2D array graphs. Matplotlib is a multi-function data visualisation toolkit based on NumPy arrays and designed to operate with the SciPy stack as a whole. One of the most important advantages of visualisation is that it provides us with visual access to massive volumes of data in simply understandable graphics.

Seaborn :

Seaborn is a fantastic Python visualisation package for plotting statistical graphs. It comes with attractive default styles and colour palettes to make statistics graphs more appealing. It is based on the matplotlib software and is tightly connected with pandas data structures. Its main goal is to make visualisation a key component of data exploration and comprehension. It also has dataset-oriented APIs, which allow us to move between several visual representations for the same variables to better comprehend the dataset.

2.6 A basic Introduction to Machine Learning:

Machine learning is a technique for teaching algorithms to learn from their previous experiences. AI includes machine learning (artificial intelligence). On sample data, Machine Learning algorithms are created from a Joblib file. On the sample data, partitioning for train and test data should be possible so that the algorithms' correctness may be measured later. Spyder, pycharm, Jupyter notebook, and other tools are examples of machine learning algorithms.Machine learning can be divided into three categories, as shown below:

Machine learning that is supervised
Machine learning without supervision
Machine learning with reinforcement

Machine learning that is supervised Data is connected to labels. Unsupervised Machine Learning does not associate labels with data. Machine learning without supervision It is decided to employ a clustering-based method. The feedback-based Reinforcement Machine Learning technique is used.

.Machine learning algorithms are divided into two types based on the data:

- Classification
- Regression

The programmer determines whether to use classification or regression based on the data. Normally, classification algorithms are used when we want to classify specific values, and regression techniques are used when we want to predict continuous variables like temperature.

2.7 Output and Actuator:

OLED DISPLAY Board:

Organic Light-Emitting Diode (OLED) is a technology that employs LEDs to produce light that is produced by organic molecules. A series of organic thin films are pierced between two conductors to create these. A brilliant light is emitted when an electrical current is applied. OLEDs allow for emissive displays, in which each pixel is individually controlled and generates its own light.

2.8 IOT Cloud Platform

Cloud is a very common need for IoT devices today enabling the remote access of Infrastructural facilities and analytics , ex - storage , Api services , security , etc

Blynk

Blynk is an IOT development Platform with app and widgets services mainly for mobile devices . Its app operates on TCP/IP Http protocol for both local and global hosting , the app and development boards connect through a central cloud using a provided API key (authorization token) . Its app has multiple widgets for data streaming , monitoring ,representing, managing and logging, it also has widgets with features of triggering and notifying asper the algorithm of the program and various types or actuators . One of the best part of blynk is that extra hardware for switches , sensor and actuators is not always important to possess as Blynk also provides a virtual pin feature which can be used to program and connect all its virtual widgets .

CHAPTER 3

WORKING OF PROJECT MODEL

3.1 Max30100 IC Working :

The IC sensing setup has Red Led , IR Led and photo diode, the re .As a finger or artery with significant blood flow / concentration is brought near the sensing region over the glass slab the emitted red and Ir Light enters the blood say up to nail and reflect back to the photo diode after some absorption (as illustrated in the figure below), this absorption is dependent on blood oxygen saturation level due to spontaneous hemoglobin oxygenation .

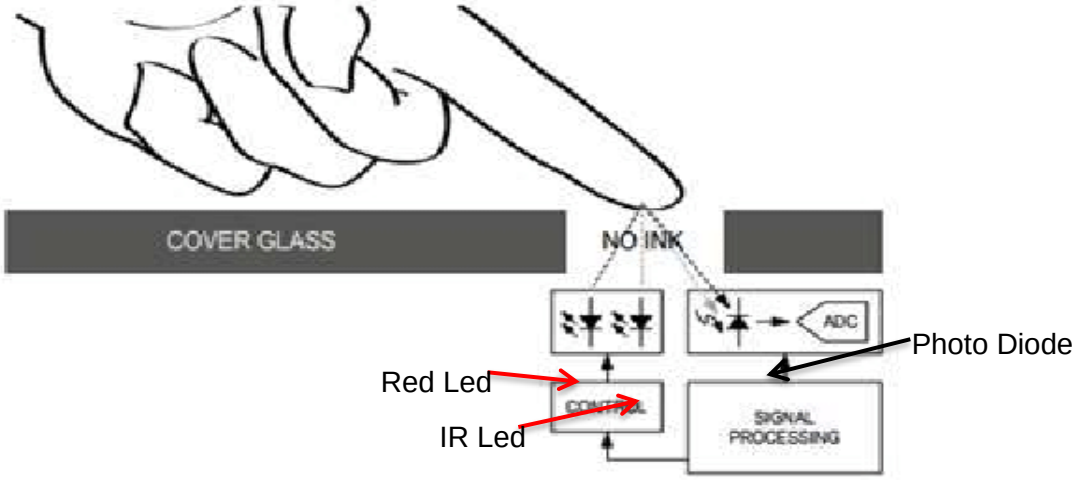


Fig 3.1

The wavelength of the Red(660nm) and IR (880nm) light makes them suitable for measuring the absorption by the presence of Hb and HbO_2 in the flowing blood.

The amount of reflected light signal measured is also dependent on pulse pressure rate causing volumetric change in presence of arterial blood with hemoglobin, which can further help in determining ppg curve with diastole and systole

In this arterial blood the deoxygenated hemoglobin absorbs more emissions of red light and allows most of the infrared light to pass through . Where as oxygenated

hemoglobin absorbs more emissions of infrared light and allows most of the red light to passing through .

Calculation of Heart rate and SpO2 :

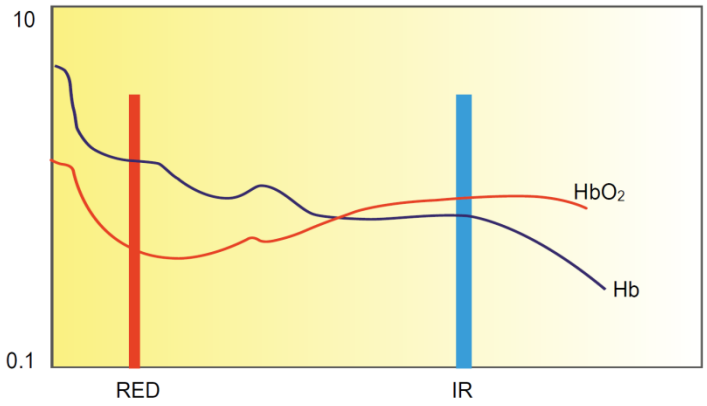


Fig 3.2

Therefore, at the photodiode or phototransistor output from the reflected light more voltage is generated at the event of deoxygenated blood detection due to infrared light received at about 880nm and at the of event oxygenated blood detection signal voltage is generated due to red light at about 660nm .

The plotted ppg signal wave mainly fluctuates with time mainly due to the increase in amount of increase in blood with each heart beat . The peak transmitted light in each wavelength is subtracted with the minimum transmitted light, by which the effects of other tissues is corrected for allowing for measurement of only the arterial blood .

The ratio of the red light received to the infrared light received is then calculated by the processor , representing the ratio of oxygenated hemoglobin to deoxygenated hemoglobin. This ratio is then used to calculate the blood oxygen level (Spo2%) by the processor using a lookup table based on the Beer-Lambert law .

3.2 Working Circuitry:

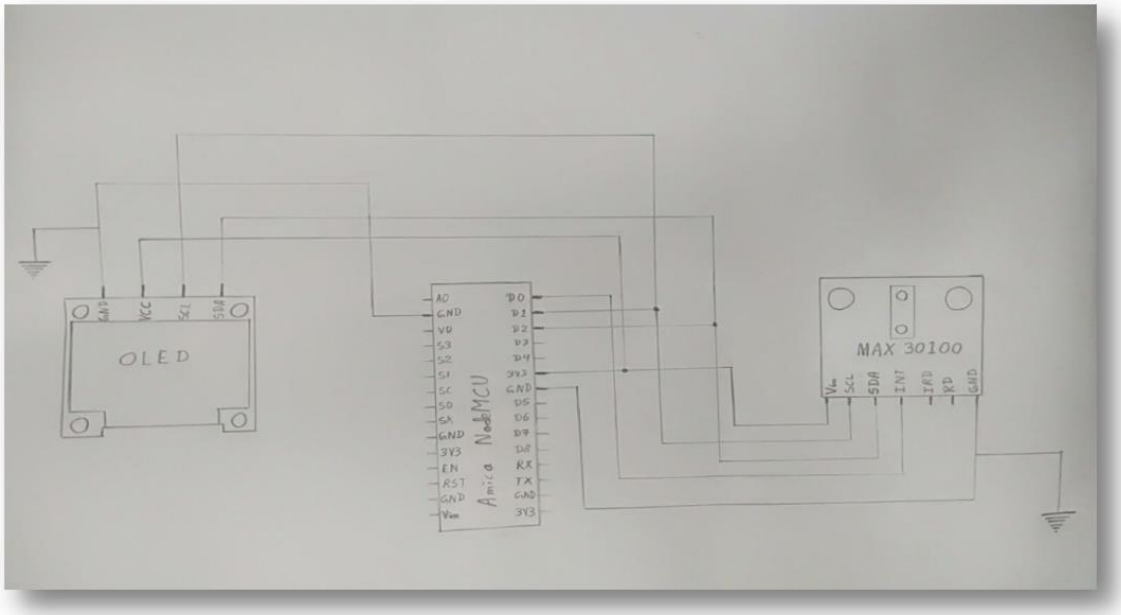


Fig 3.3

3.3 Program code :

Arduino :

Including the libraries for Esp32 libraries for WiFi, Blynk IoT cloud platform, I2C , Max30100 pulse oximeter , OLED , display graphics and fonts .

```
#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include<Wire.h>

#include "MAX30100_PulseOximeter.h">

#include<Adafruit_GFX.h>

#include<gfxfont.h>

#include <Adafruit_SH110X.h>
```

Defining I2C controller addresses for our pulse oximeter sensor and Oled , the defined oled parameters such as resolution height ,width and reset value .

```
#define MAX30i2c_Addr 0x57

#define Oledi2c_Addr 0x3C

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define OLED_RESET -1
```

Defined reporting period as 1 sec (1000ms)

```
#define REPORTING_PERIOD_MS 1000
```

```
char ssid[] = "**********";            // WiFi credentials.

char pass[] = "**********";

char auth[] = "UxxDkZFUX4Axh_ZwmCKSDXc6MYDBRgGR"; //Blynk Auth token
```

Constants c and n are defined in int to keep a count of reading a person taken , variables BPM and SpO2 are define in float to keep records of the sensed parameters .

```
int c=0 , n=1;

float BPM, SpO2;

uint32_t tsLastReport = 0;
```

A variable for noting the last sensing time was defined above to keep note of updates in value
Pulseoximeters library functions is allotted with an object pox
Object display of OLED is allotted with screen resolution , I2c controller and its reset value.

```
PulseOximeter pox;

Adafruit_SH1106G display = Adafruit_SH1106G(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);
```

'Graphics Bitmap Array', 128x64px in hexadecimal bytes for display themes and UI being added to the program memory

```
const unsigned char myBitmap [] PROGMEM = {

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f, 0xfb, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe1,
0xe3, 0x3f, 0xf3, 0xff, 0xc0, 0xff, 0xff, 0xff, 0xff, 0xff , 0xff,
0xff, 0xff, 0xff, 0xff, 0xf1, 0xe7, 0xff, 0xff, 0xff, 0xe7, 0x3f,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf5, 0xd7,
0xba, 0x7b, 0xff, 0xe7, 0x3a, 0xe3, 0xc7, 0xa6, 0x7f, 0xff, 0xff,
0xff, 0xff, 0xff, 0xf4, 0xd6, 0x20, 0x23, 0xff, 0xe7, 0x80, 0x59,
0xb2, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf6, 0xf7, 0x33,
0x33, 0xff, 0xe7, 0x93, 0xdd, 0xf3, 0x33, 0x3f, 0xff, 0xff, 0xff,
0xff, 0xff, 0xf6, 0xb7, 0x33, 0x33, 0xff, 0xe7, 0x93, 0x81, 0xc3,
0x33, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf6, 0x37, 0x33, 0x33,
0xff, 0xe7, 0x33, 0x9f, 0xb3, 0x33, 0x3f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xf7, 0x77, 0x33, 0x33, 0xff, 0xe7, 0x33, 0xcf, 0x33, 0x33,
0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe3, 0x62, 0x01, 0x01, 0xff,
0xc0, 0xf1, 0xe1, 0x80, 0x11, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```

```
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xfd, 0xff, 0xff, 0xff, 0xb0, 0x3f,
0xff, 0xff, 0xff, 0xf7, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0,
0x07, 0xff, 0xff, 0x30, 0x13, 0xff, 0xff, 0xff, 0xef, 0xff, 0xff,
0xff, 0x3f, 0xff, 0xff, 0xf7, 0xf7, 0xff, 0xff, 0x3f, 0xf3, 0xff,
0xff, 0xff, 0xff, 0xff, 0xfe, 0x7e, 0x1f, 0xff, 0xff, 0x20, 0x02,
0xff, 0xff, 0x3f, 0xf3, 0xff, 0xff, 0xff, 0xef, 0xff, 0xfc, 0x38,
0x07, 0xff, 0xff, 0x20, 0x02, 0x7f, 0xff, 0x3d, 0xf3, 0xff, 0xff,
0xfe, 0xdf, 0xff, 0xfc, 0x38, 0x07, 0xff, 0xff, 0x20, 0x02, 0x7f,
0xff, 0x3d, 0xf3, 0xff, 0xff, 0xfd, 0xbf, 0xff, 0xfc, 0x3f, 0x3f,
0xff, 0xff, 0x21, 0x82, 0x7f, 0xff, 0x3d, 0xf3, 0xff, 0xff, 0xff,
0xff, 0xff, 0xfe, 0x3f, 0x3f, 0xff, 0xff, 0x23, 0xe2, 0x7f, 0xff,
0x3c, 0xb3, 0xff, 0xff, 0xfd, 0xff, 0xff, 0xfe, 0x1f, 0xff, 0xff,
0xff, 0x23, 0xe2, 0x7f, 0xff, 0x3a, 0xf3, 0xff, 0xff, 0xdb, 0xff,
0xff, 0xff, 0x0f, 0xff, 0xff, 0xff, 0x21, 0x82, 0x7f, 0xff, 0x3f,
0x73, 0xff, 0xfe, 0x07, 0xff, 0xff, 0xff, 0x87, 0x3f, 0xff, 0xff,
0x20, 0x02, 0x7f, 0xff, 0x3f, 0x73, 0xff, 0xfc, 0x0f, 0xff, 0xff,
0xff, 0xe0, 0x1f, 0xff, 0xff, 0x20, 0x00, 0x7f, 0xff, 0x3f, 0xf3,
0xff, 0xfc, 0x0f, 0xff, 0xff, 0xff, 0xf0, 0x1f, 0xff, 0xff, 0x20,
0x02, 0xff, 0xff, 0x3f, 0xf3, 0xff, 0xfe, 0x1f, 0xff, 0xff, 0xff,
0xfc, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x03, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x1f, 0xff, 0xff, 0xff, 0xfb,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc,
0x1f, 0xff, 0xff, 0xff, 0xfd, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xcf, 0xff, 0xff, 0xff, 0xdc, 0x19, 0xff, 0xff, 0xff, 0xb5, 0xff,
```

```
0xff, 0xdf, 0x9f, 0xff, 0xff, 0x87, 0xff, 0xff, 0xff, 0x80, 0x00,
0xff, 0xff, 0xff, 0xf2, 0xff, 0xfe, 0x06, 0x07, 0xff, 0xff, 0xcf,
0xff, 0xff, 0xff, 0x01, 0x80, 0xff, 0xff, 0xff, 0xbb, 0xff, 0xfc,
0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x7f,
0xff, 0xff, 0x75, 0xff, 0xfc, 0x00, 0x03, 0xff, 0xff, 0xcf, 0xff,
0xff, 0xff, 0x80, 0x00, 0xff, 0xff, 0xfd, 0xff, 0xff, 0xfc, 0x00,
0x03, 0xff, 0xff, 0xcf, 0xff, 0xff, 0xff, 0xe0, 0x83, 0xff, 0xff,
0xfb, 0xbf, 0xff, 0xfe, 0x00, 0x03, 0xff, 0xfe, 0x40, 0xff, 0xff,
0xff, 0xc0, 0x81, 0xff, 0xff, 0xf7, 0x7f, 0xff, 0xff, 0x02, 0x0f,
0xff, 0xfc, 0xcf, 0xff, 0xff, 0xff, 0x00, 0x00, 0xff, 0xff, 0xee,
0xff, 0xff, 0xff, 0x82, 0x1f, 0xff, 0xfc, 0xcf, 0xff, 0xff, 0xff,
0x00, 0x00, 0x7f, 0xff, 0xe3, 0xff, 0xff, 0xff, 0xe0, 0x3f, 0xff,
0xfd, 0xc0, 0x3f, 0xff, 0xff, 0x00, 0x80, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xf0, 0x7f, 0xff, 0xfc, 0xff, 0x8f, 0xff, 0xff, 0x8c,
0x11, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf9, 0xff, 0xff, 0xfe,
0xfc, 0xcf, 0xff, 0xff, 0xfc, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0x01, 0xff, 0xff, 0xff, 0xfc, 0x1f,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xcf,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff };
```

```
void setup() {
```

Setup function is a part of the program that is defined to run only once in the beginning , it is used mainly for initializing program in the function , parameters , etc of the microcontroller and devices connected to it .

Serial terminal is initialized with baudrate 115200 , and the WiFi with HTTP protocol is initiazed with blynks library using credentials and the cloud apps API key.

```
  Serial.begin(115200);

  Blynk.begin(auth, ssid, pass);
```

OLED Display is initialized over I2C , allocating all the addresses and the register and pin values the library . Also keeping a check if the I2c initialization was successful and printing debug statements to the serial terminal accordingly .

```
display.begin(Oledi2c_Addr, true);

 if(!display.begin(Oledi2c_Addr,true)) {

   Serial.println(F("SH1106G allocation failed"));

   delay(200);

   for(;;)

   break;

 }
```

Clearing all the display values if any and putting theme graphics to display to the complete screen from the bitmap array loaded in program memory , and displaying it as per need .

```
 display.clearDisplay();

 display.drawBitmap(0, 0,  myBitmap,128, 64, SH110X_WHITE);

 display.display();

 delay(800);

 display.invertDisplay(1);

 delay(4000);

 display.clearDisplay();
```

Initializing Max301100 pulse Oximeter and appending debug statements to the serial terminal if not initialised , also displaying wait of sensor parameter value on the screen .

```
Serial.print("Initializing Pulse Oximeter.. ");

pox.begin();

   if (!pox.begin()) {
```

```
Serial.println("Failed to allocate i2c to POX");

displayWait();

delay(200);

for (;;)

break;

}

    else {

Serial.println("SUCCESS!");

Serial.println(" time , Description , BPM , SPO2;");

} }
```

```
void testdrawChar(void) {

  display.setTextSize(1);

  display.setTextColor(SH110X_WHITE);

  display.setCursor(0, 0);

   for (uint8_t i = 0; i < 168; i++) {

    if (i == '\n') continue;

    display.write(i);

    if ((i > 0) && (i % 21 == 0))

      display.println();

  }

  display.display();

  delay(1);

}
```

Loop() is used for running program instructions on the microcontroller in an infinite loop .

```
void loop() {
```

Pulse oximeter is instructed for update , Bylnk server is initialized and values are fed to the Iot android cloud platform. Function parameters for calculating pulse rate and level of SpO2 is allocated to defined variables of the sensor parameters .

```
pox.update();

Blynk.run();

BPM = pox.getHeartRate();

SpO2 = pox.getSpO2();
```

Conditional statement is being added to check if some change in sensor parameters is being sensed i.e a finger is being placed on the sensor . If sensed with with a period of more than 1second ,

the measured values are being logged to the serial terminal

```
    if((BPM != 0) && (SpO2 != 0)){

     if (millis() - tsLastReport > REPORTING_PERIOD_MS)

    {

      Serial.print(",Heart rate & SpO2 , ");

      Serial.print(BPM);

      Serial.print(" , ");

      Serial.println(SpO2);

      c++ ;
```

The sensor measured values are being fed to the widgets of the blynk cloud through there allotted virtual pins.

```
        Blynk.virtualWrite(V7, BPM);
```

```
        Blynk.virtualWrite(V8, SpO2);
```

The OLED is being set for display and the text font size and colour is selected. Also, the cursor is set to the starting of the display .

```
  display.clearDisplay();

  display.invertDisplay(0);

  display.setTextSize(1);

  display.setTextColor(SH110X_WHITE);

  display.setCursor(0, 0);
```

Instructions for appropriate spacing , character , measure sensor parameter printing are being called with update at regular intervals .

```
  display.println(" ");

  display.display();

  delay(100);

  display.println(" ");

  display.display();

  delay(100);

  display.setTextSize(2);

  display.setTextColor(SH110X_WHITE);

  display.print("BPM");

  display.display();

  delay(100);

  display.print(" ");

  display.display();

  delay(100);
```

```
  display.print(":");
  display.display();
  delay(100);
  display.print(" ");
  display.display();
  delay(100);
  display.println((int)BPM);
  display.display();
  delay(500);
  display.print("SPO2");
  display.display();
  delay(100);
  display.print(":");
  display.display();
  delay(100);
  display.print(" ");
  display.display();
  delay(100);
  display.print((int)SpO2);
  display.display();
  delay(200);
  display.println("%");
  display.display();
  delay(500);
//  display.setTextColor(SH110X_BLACK, SH110X_WHITE);
```

```
  display.display();

  delay(4000);

  tsLastReport = millis();

    }
```

On completion of recording the required number of sensed values of an individual the recording session is ended and counter is refreshed .

```
if (c >= 1100){

        Serial.print(n );

        Serial.println("Person's session complete.");

        c = c - 1100;

        n++;

        }

}
```

The condition of both sensor parameters , spo2 level and pulse rate is being declared as no finger being placed for sensing .

```
    if((BPM = 0) && (SpO2 = 0)){

      Serial.println("No Finger");

      delay(800);

     pox.update();

     BPM = pox.getHeartRate();

      SpO2 = pox.getSpO2();

      for(;;)

      break;

    }
```

```
}
```

A function with instructions for displaying a wait state for measured values to displayed is defined as displayWait()

```
void displayWait(void){

  display.clearDisplay();

  display.invertDisplay(0);

  display.setTextSize(1);

  display.setTextColor(SH110X_WHITE);

  display.setCursor(0, 0);

  display.println(" ");

  display.display();

  delay(100);

  display.println(" ");

  display.display();

  delay(100);

  display.setTextSize(2);

  display.setTextColor(SH110X_WHITE);

  display.print("BPM");

  display.display();

  delay(100);

  display.print(" ");

  display.display();

  delay(100);

  display.print(":");

  display.display();
```

```
  delay(100);

  display.print(" ");

  display.display();

  delay(100);

  display.print("_");

  display.display();

  delay(500);

  display.print("_");

  display.display();

  delay(500);

  display.println("_");

  display.display();

  delay(500);

  display.print("SPO2");

  display.display();

  delay(100);

  display.print(":");

  display.display();

  delay(100);

  display.print(" ");

  display.display();

  delay(100);

  display.print("_");

  display.display();

  delay(500);
```

```
  display.print("_");

  display.display();

  delay(500);

  display.println("_");

  display.display();

  delay(500);

//  display.setTextColor(SH110X_BLACK, SH110X_WHITE); // 'inverted' text

  display.display();

  delay(5000);

  for (;;)

break;

}
```

Python Code for ML :

Importing libraries in order to use different features in ML

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
pd.pandas.set_option('display.max_rows',None)
import seaborn as sns
from sklearn import preprocessing

data = pd.read_csv(r'C:\Users\WINDOWS HP\Desktop\New folder\PPG_
&h_edit.csv')
data.head()

data.columns = ['Time','Description','BPM','SPO2']
data.head()
```

```
data.drop('Description',axis=1,inplace=True)
data.head()
```

Converting time to specific min,hour and seconds

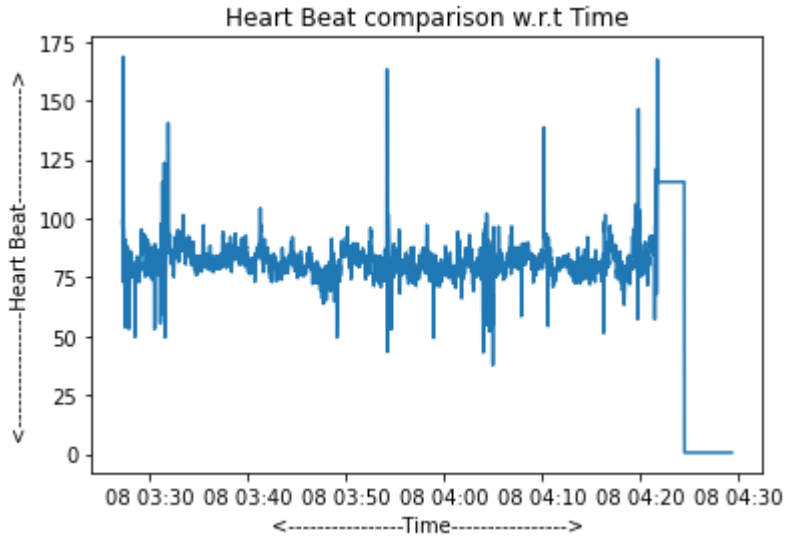
```
data['Time'] = pd.to_datetime(data['Time'])
data['Minute'] = data['Time'].dt.minute
data['Hour'] = data['Time'].dt.hour
data['Second'] = data['Time'].dt.second
data.head(10)
data['SPO2'] = data['SPO2'].astype('float64')
data.info()
```

Conditions for predicting the condition of the person

```
data.loc[(data['BPM'] == 0.00), 'Condition'] = 'Dead'
data.loc[(data['BPM'] < 70) & (data['SPO2'] < 95), 'Condition'] =
'Critical'
data.loc[((data['BPM'] >= 70)&(data['BPM'] <= 90)) | ((data['SPO2']
>= 95)&(data['SPO2'] <= 99)), 'Condition'] = 'Normal'
data.loc[(data['BPM'] > 90) & (data['SPO2'] > 100), 'Condition'] =
'Abnormal'
data.head()
data.shape
```

Plotting heart rate vs time graph:

```
plt.plot_date(data['Time'], data['BPM'], linestyle='solid')
plt.title('Heart Beat comparison w.r.t Time')
plt.xlabel('<----------------Time---------------->')
plt.ylabel('<----------------Heart Beat----------------
```



Plotting SPO2 vs time graph:

```
fig, ax = plt.subplots()
ax.plot_date(data['Time'], data['BPM'], linestyle='solid', marker='')
ax
plt.title('Heart Beat comparison w.r.t Time')
plt.xlabel('<----------------Time---------------->')
plt.ylabel('<----------------Heart Beat---------------->')
fig, ax = plt.subplots()
ax.plot_date(data['Time'], data['SPO2'], ydate=True,
linestyle='solid')

fig, ax = plt.subplot
ax.plot_date(data['Time'], data['SPO2'], ydate=True,
linestyle='solid', marker = '')
```

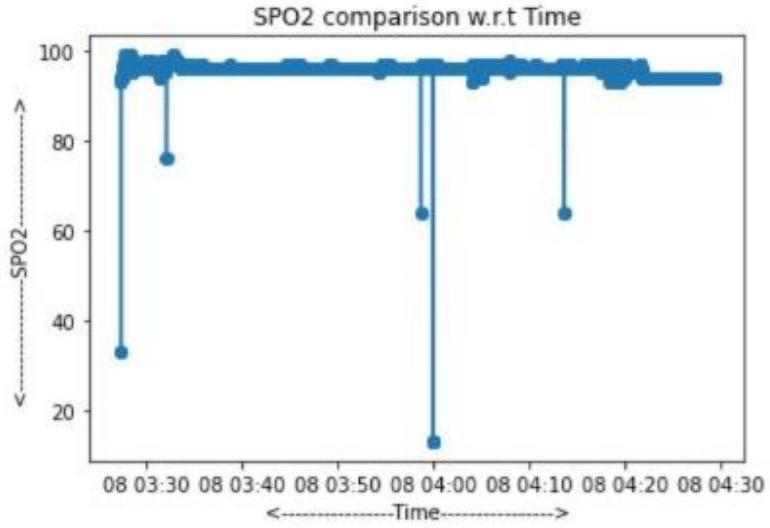


FIG 3.5

```
#ax.xaxis.set_major_formatter(DateFormatter('%H:%M'))
#ax.yaxis.set_major_formatter(DateFormatter('%H:%M'))
plt.title('SPO2 comparison w.r.t Time')
plt.xlabel('<----------------Time---------------->')
plt.ylabel('<----------------SPO2---------------->')
#df = data.groupby(['Minute']).agg(BPM_mean = ('BPM', 'mean'))
#df
data

data_length = len(data['Minute'])-1
```

New mean time:

```
#New Time

i=0
time = []

while(i<data_length):
    a=data['Minute'][i]
    j=i
```

```
    count = 0
    s = 0
    mean = 0
    while(j<data_length):
        if(a == data['Minute'][j]):
            count = count + 1
            j=j+1
        else:
            break

    time.append(data_copy['Time'][j-1])
    i=i+count
```

#New Hour

```
i=0
hour = []

while(i<data_length):
    a=data['Minute'][i]
    j=i
    count = 0
    s = 0
    mean = 0
    while(j<data_length):
        if(a == data['Minute'][j]):
            count = count + 1
 j=j+1
        else:
            break

    hour.append(data['Hour'][j-1])
    i=i+count
```

#New Minute

```
i=0
minute = []

while(i<data_length):
    a=data['Minute'][i]
    j=i
    count = 0
    s = 0
    mean = 0
    while(j<data_length):
        if(a == data['Minute'][j]):
            count = count + 1
            j=j+1
        else:
            break

    minute.append(data['Minute'][j-1])  i=i+count
```

#New Second

```
i=0
second = []

while(i<data_length):
    a=data['Minute'][i]
    j=i
    count = 0
    s = 0
    mean = 0
    while(j<data_length):
if(a == data['Minute'][j]):
            count = count + 1
            j=j+1
        else:
            break

    second.append(data['Second'][j-1])
    i=i+count
```

```
print(len(time))
print(len(hour))
print(len(minute))
print(len(second))
```

#Mean BPM & SPO2

```
def mean(data,feature):
    i=0
    spo2_mean=[]
    while(i<data_length):
        a=data['Minute'][i]
        j=i
        count = 0
        s = 0
        mean = 0
        while(j<data_length):
            if(a == data['Minute'][j]):
                s = data[feature][j] + s
                count = count + 1
                #spo2_mean = (s/count).astype(int)
                #spo2_mean.append(s/count)
                j=j+1
            else:
                break

        mean = round(s/count,2)
        spo2_mean.append(mean)
        i=i+count
    return(spo2_mean)

#spo2_mean
```

```
data_new = pd.DataFrame()

data_new =pd.DataFrame({'Time' : time,
```

```
                        'Hour' : hour,
                        'Minute' : minute,
                        'Second' : second })
data_new.head()

for feature in ['BPM','SPO2']:
    xy = mean(data,feature)
    mn = pd.DataFrame(xy)
    data_new = pd.concat([data_new,mn],axis=1)


data_new.columns =
['Time','Hour','Minute','Second','BPM_mean','SPO2_mean']
data_new.head()
```

```
data_new.loc[(data_new['BPM_mean'] == 0.00), 'Condition'] = 'Dead'

data_new.loc[((data_new['BPM_mean'] < 60)|(data_new['BPM_mean'] >
100)) |
             ((data_new['SPO2_mean'] < 94)|(data_new['SPO2_mean'] >
100)), 'Condition'] = 'Critical'

data_new.loc[((data_new['BPM_mean'] >= 60)&(data_new['BPM_mean'] <=
100)) &
             ((data_new['SPO2_mean'] >= 94)&(data_new['SPO2_mean'] <=
99)), 'Condition'] = 'Normal'

data_new.head()

#data_new.to_csv(r'E:\Bio Sensing\PPG_work.csv')

data_new = pd.read_csv(r'C:\Users\WINDOWS HP\Desktop\New
folder\PPG_work.csv')
data_new.drop('Unnamed: 0',axis=1,inplace=True)
data_new['Time'] = pd.to_datetime(data_new['Time'])
sns.countplot(x = data_new['Condition'])
plt.title('Count of number of Target Feature')
```

```
data_new['Condition'].value_counts().plot(kind='pie',autopct='%.2f')
plt.title('Percentage share of Target Feature')


sns.scatterplot(x = data_new['Time'], y = data_new['BPM_mean'],hue =
data_new['Condition'])

sns.scatterplot(x = data_new['Time'], y = data_new['SPO2_mean'],hue =
data_new['Condition'])
```

```
le=preprocessing.LabelEncoder()
le.fit(['Critical','Normal'])
y=pd.DataFrame()
data_new['Condition'] = le.transform(data_new['Condition'])

data_new.head()

corr = data_new.corr()
plt.figure(figsize=(10,7))
sns.heatmap(corr, annot=True, cmap='coolwarm')

data_new.info()

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report

x = data_new[['Hour','Minute','Second','BPM_mean','SPO2_mean']]

y = data_new['Condition']
```

Dividing data into test and train inorder to train the model

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.20, random_state = 42)

print('Tota train rows {}'.format(len(x_train)))
print('Tota test rows {}'.format(len(x_test)))
```

```
# Training model using Decision Tree Classifier

def training_model():
    from sklearn.tree import DecisionTreeClassifier
    classifier = DecisionTreeClassifier(random_state=42)
    classifier.fit(x_train, y_train)
    return classifier

model = training_model()
```

#Visualizing the performace of the model

```
y_pred_train = model.predict(x_train)

#Printing the classification_report
print(classification_report(y_train, y_pred_train,
                            target_names = ['Critical','Normal']))
```

#Printing the confusion_matrix

```
cm = confusion_matrix(y_train , y_pred_train)
print(cm)
```

#Printing the accuracy score

```
acc_score = accuracy_score(y_pred_train , y_train)
acc_score = acc_score*100
print('\nAccuracy of test set {}%'.format(acc_score))
```

#Visualizing the performace of the model

```
y_pred_test = model.predict(x_test)

#Printing the classification_report
print(classification_report(y_test, y_pred_test,
                            target_names = ['Critical','Normal']))
```

#Printing the confusion_matrix

```
cm = confusion_matrix(y_test , y_pred_test)
print(cm)
```

#Printing the accuracy score

```
acc_score = accuracy_score(y_pred_test , y_test)
acc_score = acc_score*100
print('\nAccuracy of test set {}%'.format(acc_score))
```

```
x.head(0)

def check_data():
    hour = int(input('Enter the current hour : '))
    minute = int(input('Enter the current minute : '))
    second = int(input('Enter the current second : '))
    bpm_mean = float(input('What is your current BPM : '))
    spo2_mean = float(input('What is your current SPO2 : '))
    d = [[hour,minute,second,bpm_mean,spo2_mean]]
    pd.DataFrame(d)
```

Checking data and testing the model:

```
    prediction = model.predict(d)
    normal = 'Your health condition is normal'
    critical = 'Yor health condition is critical'

    if prediction == 0:
            return critical
    elif prediction == 1:
            return normal

check_data()

check_data()
check_data()
check_data()
data_new[data_new['Condition']=='Critical']
```

```
#asthma
#normal
#breathing
#avg. 10 diff persons
#per person 100mins.
data_new
```

Results :

Hardware Setup :

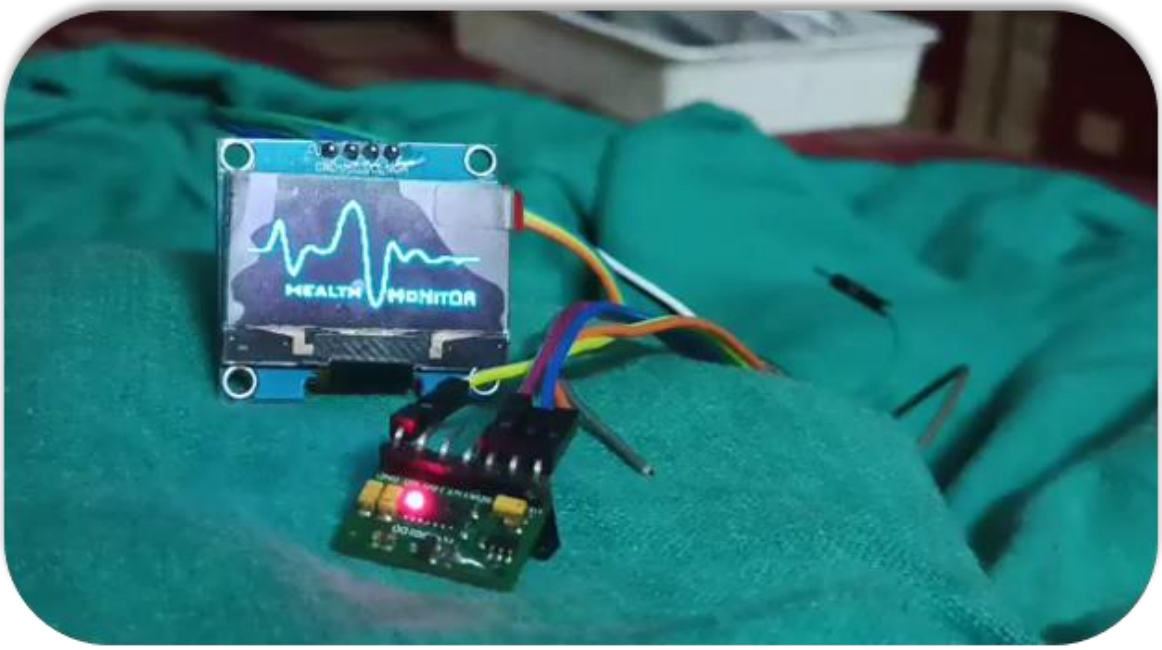


Fig 3.6

CLOUD APP SETUP :

FIG 3.7

CHAPTER 4
PROJECT SUSTAINABILITY AND MARKET SCOPE

4.1 Patients continuous health monitoring on AI could:

Cloud computing will be substantially more effective as a result of AI. It makes AI more accessible by democratising it. It fosters AI-powered transformation for businesses by cutting adoption costs and encouraging co-creation and innovation. The cloud expands AI's breadth and sphere of effect substantially, first within the user organisation and later throughout the marketplace. In fact, AI and the cloud will mutually benefit, allowing AI's true potential to bloom through the cloud.

- Low power consumption:

During continuous operation, commercial pulse oximeter sensors use roughly 20-60 mW of electricity. Other researchers have demonstrated that wireless pulse oximeter sensors can be built to operate with as low as 1.5 milliwatts of power. In pulse oximeter sensors, LEDs occupy the majority of the power budget.

- **Graphical View**
- **IOT Enabled**

- **Can be deployed Even as regular wearable device**

Pulse oximeters can be made into wearables and that data can be used in very different ways.

It started with measuring blood oxygen levels when you placed your finger on the sensor on the back of the device. But it can also be taken more easily from the wrist.You can also take spot readings too, by cycling through the watch menu on the device itself.This is presented as a true blood oxygen percentage score.

- **Cost Effective**: Very low cost devices and circuits were used to build the setup leading to estimated cost of production of about Rs. 700 -800 for the PPG setup.

CHAPTER 5
FUTURE SCOPE OF ENHANCEMENT

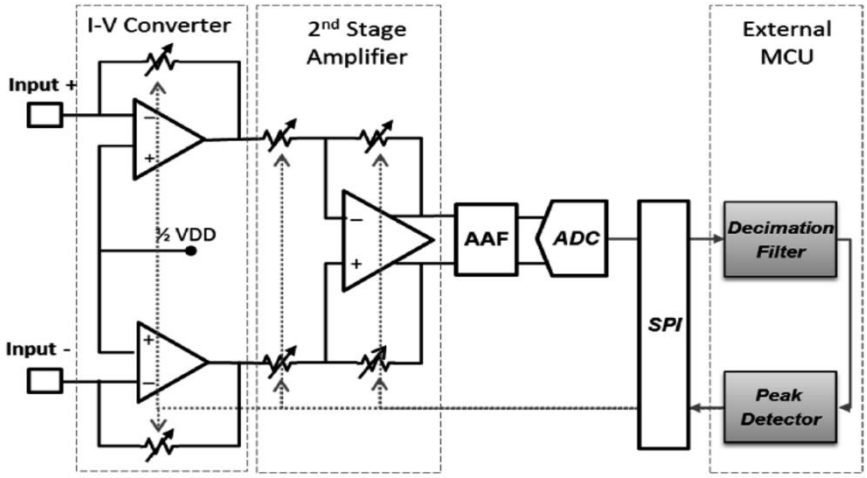
5.1 Prediction ML Analysed:

The dataset (SPO2, Heart Rate, Age, and Gender) collected by various people's pulse oximeters can be utilised to train and test the model. Based on this, a decision can be made about which machine learning algorithm to use or which machine learning algorithm produces the best results on the dataset. There are three aspects to the Machine Learning implementation here.

(1) Prediction of SPO2 and/or HR based on other factors (Supervised Machine Learning)
(2) SPO2 prediction and/or time prediction (Supervised Machine Learning)
(3) Prediction of a person's health based on COVID -19 potential.
HR at Transit time of the Ecg Pulse,Adaptation of calculated blood pressure to the user's routine health metrics, deviations, and identification of important health problems.

5.2 Ecg developments:

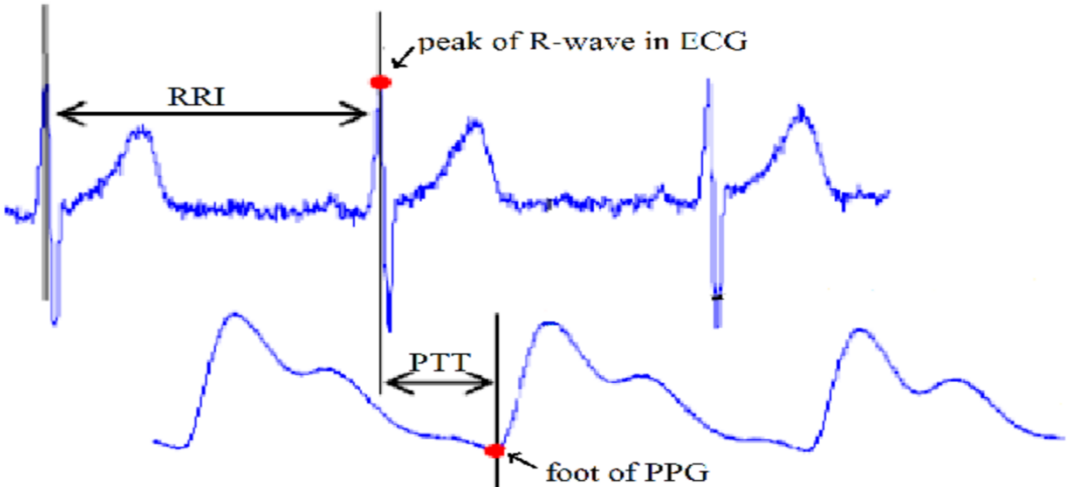
To make workflow easier, ECG systems have been streamlined. Simplified step-by-step operation, the advent of touch-screen systems, and improved interface with ECG management systems, electronic medical records (EMRs), and cardiovascular information systems are just a few examples (CVIS). As healthcare reform and Meaningful Use requirements for information technology (IT) systems call for tight integration and easy interoperability to enable paperless healthcare and the free flow of all patient data between departments, hospitals, and referring physicians, the interoperability component will become increasingly important.



5.1 Schematic diagram of the ECG Sensor

5.3 Pulse transit time calculated bp:

PTT is the time delay for the pressure wave to travel between two arterial sites and can be estimated simply from the relative timing between proximal and distal arterial waveforms.



5.2 Pulse transit time calculation

CHAPTER 6
CONCLUSION

Pulse oximetry is truly vital to medical care, but this prominence carries a responsibility. Too often, pulse oximetry readings are misinterpreted or are erroneous for various reasons (see sidebar). The tendency to accept the percentage at face value is high. Many expensive probes continue to be discarded prematurely as defective. Direct patient interventions can also occur based on erroneous results; worse, patients can be inappropriately monitored, giving caregivers a false sense of security. Pulse oximetry has come a long way since its inception in the early 1800s and with future technology, pulse oximeters will continue to improve patient care more effectively and efficiently**.**

Measuring your heart rate and oxygen saturation in your blood was not as simple as we had hoped. However, with perseverance, we were able to gain a sufficient understanding of the DSP involved as well as the theory underlying measuring SpO2 to implement it from the ground up. Not only is all of this specific to the MAX30100, but comparable approaches and calculations should be performed on either your own self-built sensor or a sensor made by a company other than Maxim. The MAX30100 allows you to integrate a relatively complex analogue circuit into a very small package. However, based on our preliminary studies, we must conclude that monitoring heart rate from the wrist using a sensor is quite challenging.With the existing technique for recognising peaks, it's practically impossible. It's also worth noting that when measuring oxygen saturation in the paper, I didn't properly calibrate the sensor; instead, I simply altered a standard model to fit what I thought was correct. If you intend to use this sensor to measure SpO2, it is critical that you calibrate it properly.

REFERENCES :

[1]. Assessment and Monitoring of Respiratory Function *Emily L. Dobyns,* Chapter: 39 *in Pediatric Critical Care (Fourth Edition), 2011*

[2]. Iowa Head and Neck Protocols ,Pulse Oximetry Basic Principles and Interpretation .March ,2017

[3]. Websites- Stack Overflow
- Random Nerd
- Adarfruit
- Maxim Integrated
- Element 14