

# **System for Remote Health Management of Quarantined Patients**

---

**By**

**Arnav Ahluwalia**

**Ishaan Datta**

**Rohit Kumar**

**Final Report for ECE 445, Senior Design, Spring 2021**

**TA: Yichi Zhang**

**03 May 2021**

**Team 23**

## Abstract

We have developed a system for remotely monitoring health parameters of patients in home quarantine. Three of subsystems are on the device and the fourth is a backend service for capturing data from the device and assisting medical personal in monitoring health of the patients.

Within the system, a control module is programmed to receive data from three sensors that measure Blood Pressure, Pulse Rate, Temperature and Blood Oxygen of a home quarantined patient. A WiFi module thereafter transmits the measured data via a Home WiFi or Smartphone hotspot to an Internet based web service. Authorized medical personnel can logon to a web portal frontend for remotely monitoring patient health parameters.

We have successfully implemented the entire system end to end and met all targeted requirement.

## Table of Contents

1. Introduction .....	1
2. Design .....	2
2.1 Design Procedure: .....	2
2.2 Design Details .....	4
2.2.1    Subsystem 1: Control and Power Supply Unit .....	5
2.2.2    Subsystem 2- Sensors for measuring patient health data .....	6
2.2.3    Subsystem 3- WiFi Module .....	7
2.2.4    Subsystem 4 - Patient Management Web Portal .....	8
2.3 Design Verification.....	9
2.3.1    Subsystem-1 and 2 (Control Unit and Sensors) verification: .....	9
2.3.2    Subsystem- 3 (WiFi Module) verification:.....	11
2.3.3    Subsystem- 4 (Patient Management Web Portal) verification: .....	12
3. Costs .....	15
4. Conclusion .....	16
5. References .....	17

## 1. Introduction

During the pandemic, monitoring the health of home quarantined patients at scale enables provisioning of timely medical support.

It's been an ongoing struggle to prioritize individuals who are at-risk, leading to a lack of equal access to medical staff and facilities. This is particularly relevant for places where the outbreak has progressed to an extent that not enough beds/medical facilities are available to cater to every patient and triaging is being carried out, i.e., medical personnel must tend to higher-risk/seriously ill patients [1].

Whether we consider first-world regions such as New York/Chicago, or third-world countries such as India, high rates of COVID concurrent with high population densities are the major factors that influence capability of medical infrastructure to handle a pandemic.

Enabling medical authorities to remotely monitor the health of COVID-19 patients would lessen the load on hospitals and help divert medical resources well in time to people who need it the most.

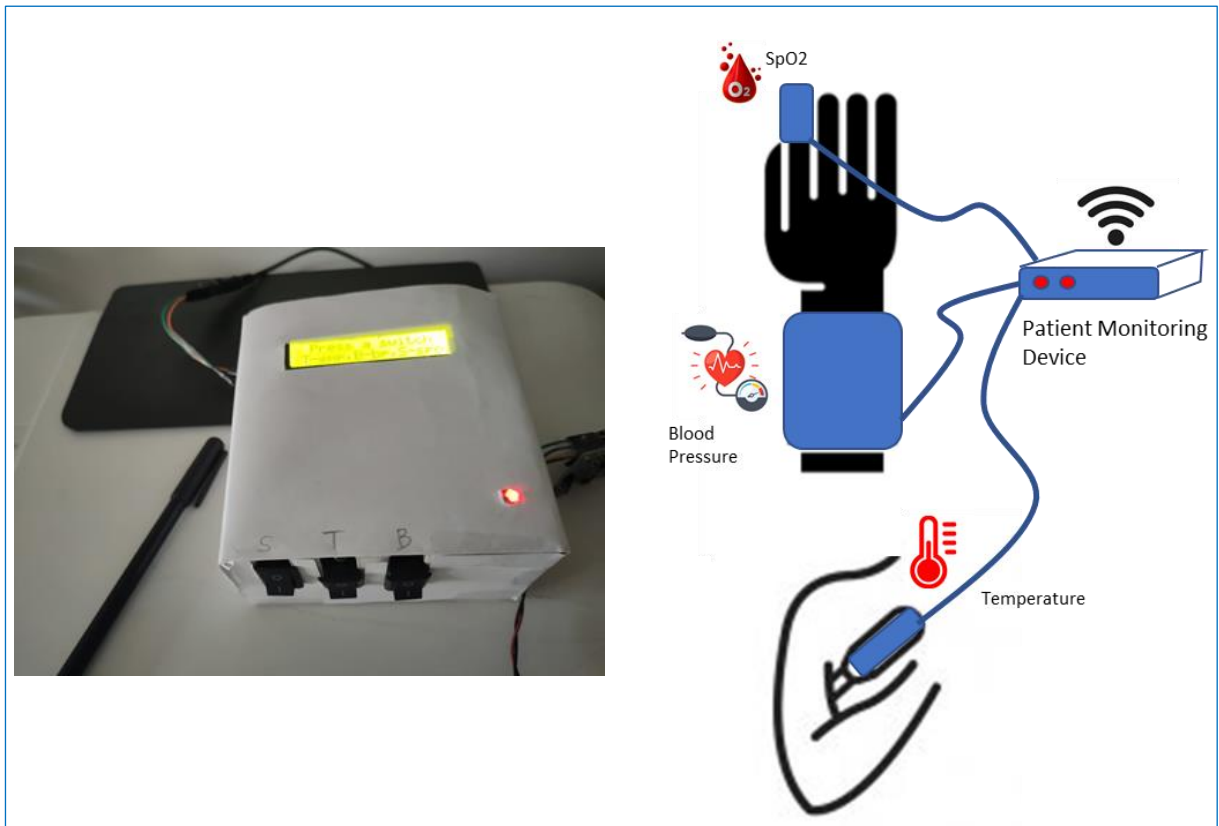
Our objective was to develop a low-cost medical device coupled with a scalable backend service which is capable of monitoring the status of patients while they're at home, allowing for medical professionals to remotely and efficiently keep track of their vital parameters.

## 2. Design

### 2.1 Design Procedure:

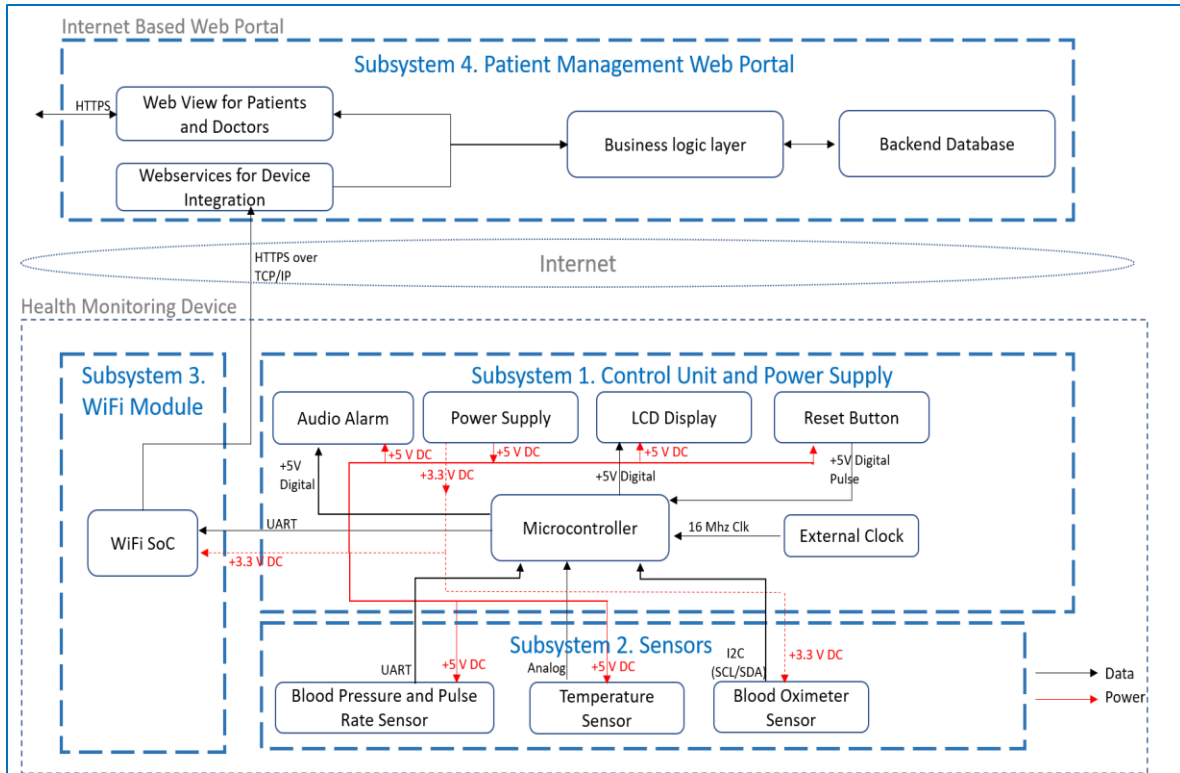
The project has been designed to satisfy three primary requirements:

1. The device (Fig.1) should be compact (7cm width x 7cm height x 4cm height), affordable (less than \$50 per device) and be able to measure patient vitals like pulse rate, blood oxygen, temperature and blood pressure with an accuracy greater than 95%.
2. The device should communicate with an internet-based backend service using HTTPS transport protocol via a Home WiFi (802.11 a/b/g) network to transmit the measured readings.
3. The backend service should accept and store patient data securely (AES-256 bit encrypted) and provide password based authenticated access to authorized medical personnel in monitoring patient status.



**Fig. 1. Patient Monitoring Device**

The overall system comprises four major subsystems (Fig. 2)- the control unit and power supply, sensors for measuring patient health data, the WiFi module and the patient management web portal.



**Fig. 2. Overall Block Diagram**

The first subsystem on the device comprises of a power supply and an ATmega386 microcontroller. The microcontroller is programmed to receive data from the second subsystem, i.e., three sensors for measuring Blood Pressure, Pulse Rate, Temperature and Blood Oxygen of the patient.

The third subsystem is an ESP8266 based WiFi module that receives data from the microcontroller and transmits the same via a Home WiFi or Smartphone hotspot to the fourth subsystem, an Internet based web portal. Medical personnel can logon to the web portal for remotely monitoring the received health parameters of patients.

The system has additional features for registering patients and medical personnel, providing authenticated access to the web portal, securing the data during transit and storage, visual display of patient parameters on the device and raising alerts in case of abnormal data.

We have added additional features and changes in our initial design basis feedback from our staff. The specific requirements added are use of HTTPS instead of HTTP for data transfer, Visual display at the device, Audio alarm in case of abnormal readings, and the capability to permit a patient in setting a client WiFi SID/ password on the WiFi module via a mini web portal on the WiFi chip.

The most important factors while selecting components for the device were affordability, accuracy and compactness.

While selecting a microcontroller, the ATmega328 [2] was used as:

- It allows interfacing with each of our sensors (supports UART and I<sup>2</sup>C).
- Is affordable at a price of \$2.
- Is relatively easy to program and allows convenient integration with Arduino-compatible sensors.
- No complex computations are necessary, thus a small 2KB SRAM is sufficient.
- 32KB flash memory sufficient for our small sketch sizes.

The ESP8266 SOC was selected for Internet access via a Home WiFi. This device meets our criteria for affordability, compactness and availability in addition to capability to integrate with an Atmega microcontroller via a UART interface.

Similarly, the three health sensors for Blood Pressure, Pulse Rate, Temperature and Blood Oxygen were chosen for their accuracy, compactness, capability to integrate with the Atmega microcontroller via standard UART/ I2C protocols, cost and ease of availability.

## 2.2 Design Details:

The overall schematic (Fig. 3) was prepared using EasyEDA designer [3] and is as under:

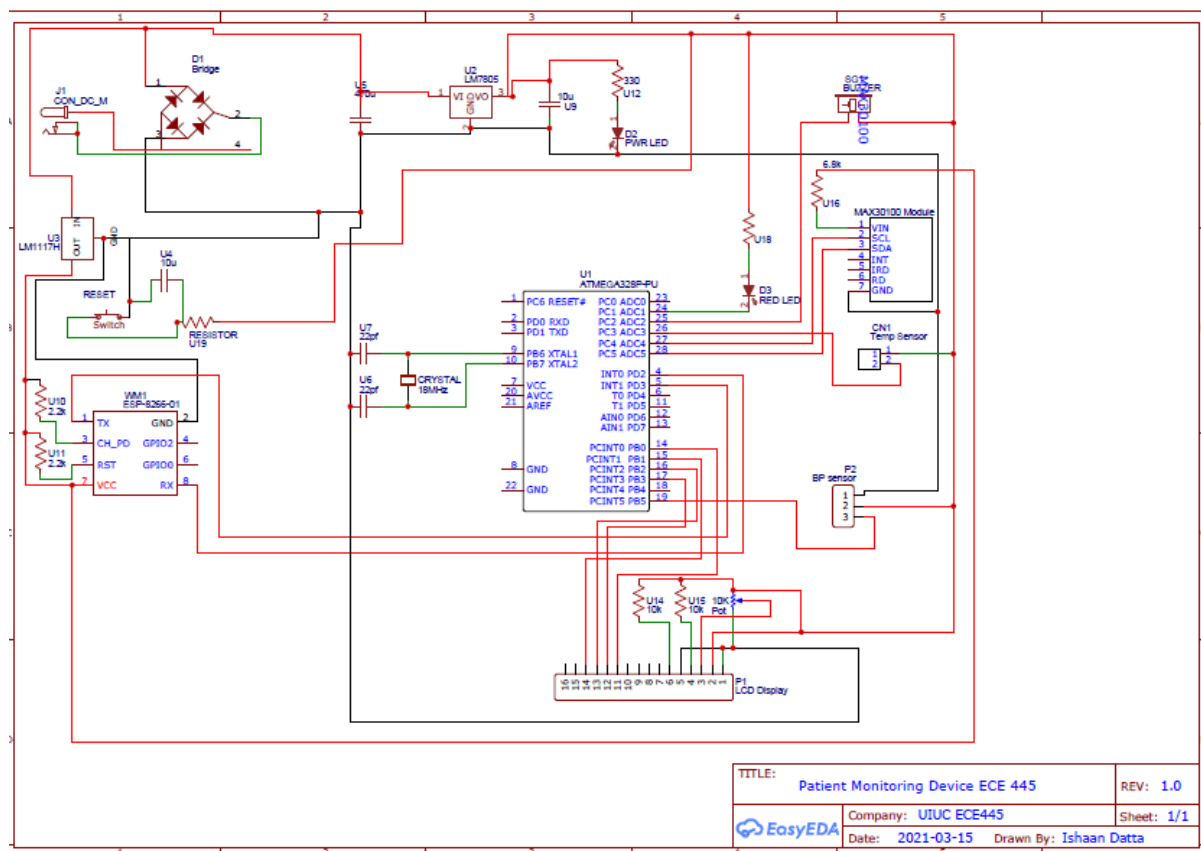


Fig. 3: Schematic

### 2.2.1 Subsystem 1: Control and Power Supply Unit

**Control Unit (ATMEGA 328 PU) [2]:** The microcontroller (Fig. 4) would form the heart of the system that captures sensor (Subsystem-2) data and does suitable protocol conversions to send the data periodically to the backend application (Subsystem-4) via the onboard WiFi Module (Subsystem-3).

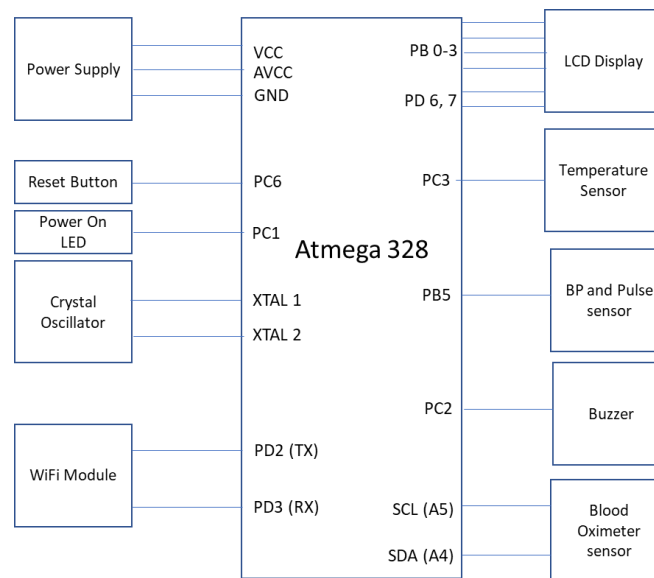
**Inputs:** 5V DC Power supply at up-to 12mA, Temperature sensor- Analog input (0 to 10mV), Blood Pressure and Pulse sensor- Digital UART input at 9000 Bauds, Blood Oximeter- I2C interface (SCL/SDA), external oscillator for frequency of operation (16 Mhz), reset button.

**Outputs:** UART output (0 to 5V PWM) to WiFi module (patient ID, processed sensor data) working at 9600 bps, LCD Display output, Audio Alarm.

Once the device is powered on and reset button is pressed, the microcontroller polls the I/O pins for sensor data at delay of every 2 secs and captures the data when available.

- Displays the readings on a local LCD display and sends the data onwards to the WiFi module via UART interface.
- In case the readings are not within normal ranges, a local audio alarm (buzzer) goes off.

Thereafter the captured sensor data is combined with a unique device Patient Identity number hardcoded in the microcontroller firmware, and transmitted to the WiFi module via a UART interface at 9600 bps. Digital I/O pins PD2 and PD3 are used for TX and RX respectively.



**Fig. 4: ATmega328 interface with other components**

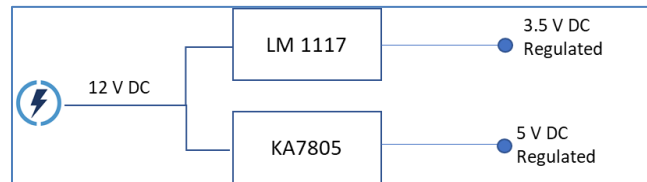
**Power Supply Unit:** The power supply unit (Fig. 5) provides both 5V and 3.5V DC regulated power supply to the circuit components.

**Inputs:** 12V DC power supply from an over the shelf 230V AC to 12V DC convertor adaptor.



**Outputs:** Regulated +5V DC  $\pm 5\%$  supply to ATmega386 microcontroller, BP sensor and Temperature sensor. Regulated +3.3V DC  $\pm 5\%$  supply to ESP8266 WiFi module, and MAX30100 Pulse Oximeter IC.

The 5V DC regulator is able to provide minimum 216mA at 5V in our circuit. In our case, the KA7805 [4] regulator can provide up to 1 Amps of current (with heat sink) at 5V DC and hence meets requirements. The +3.3 V DC regulator is able to supply 100mA at 3.3V in our circuit. The LM 1117 [5] regulator provides up to 800mA at 3.3V DC with heat sink



**Fig. 5: Block diagram of the power module**

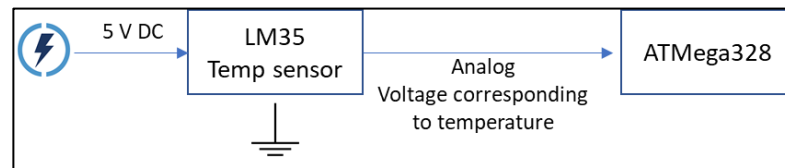
### 2.2.2 Subsystem 2- Sensors for measuring patient health data

We have used BP/ Pulse Rate, Blood pulse oximeter and Temperature measurement sensors in our project.

**Temperature Sensor:** We use the TI- LM35 [6] temperature sensor (Fig. 6), that provides analog readings within an error margin of 0.5 °C (at 25°C). It provides a linear rise in its output of 10mV/°C.

**Inputs:** +5V DC VCC and GND from power supply unit.

**Outputs:** 0 to 1500mV linear rise at the rate of 10mV per 1°C rise in temperature.

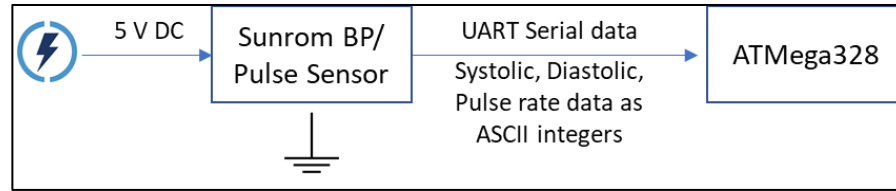


**Fig. 6: Block diagram of Temperature sensor**

**Blood Pressure and Pulse Rate Sensor (Fig. 7):** An over-the-counter Sunrom blood pressure sensor [7] has been utilized to measure patient's systolic/diastolic blood pressure and pulse rate. The patient needs to place the BP sensor cuff around the elbow and measure the BP as per usual practice. These readings would be output to the microcontroller using a UART interface.

**Inputs:** +5V DC VCC and GND from power supply unit.

**Outputs:** UART Serial data readings (ASCII integer values of Systolic pressure, Diastolic pressure, Pulse rate) to PB5 pin 19 of the microcontroller.



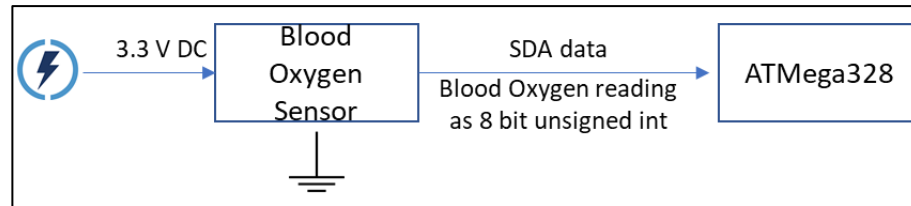
**Fig. 7: Block diagram of Blood Pressure/ Pulse rate sensor**

**Blood Oxygen Sensor (Fig. 8):** We use the MAX30100 based Pulse Oximeter Integrated Circuit [8] to measure the patient's blood-oxygen saturation (also called **SpO2**, Saturation of Peripheral Oxygen).

**Inputs:** +3.3V VDD (for Max30100 IC) DC and GND from power supply unit.

**Outputs:** IC2 SDA outputs SpO2 reading to microcontroller.

**Block Diagram:**



**Fig. 8: Block diagram of Blood Oxygen sensor**

### 2.2.3 Subsystem 3- WiFi Module

Characteristics of the WiFi subsystem (Fig. 9) are as follows:

- It is capable of working both as a WiFi Access Point as well as WiFi client concurrently.
- It supports 802.11 b/g/n protocols that are prevalent in home WiFi routers.
- It has a full-fledged TCP/IP stack with support for HTTPS/ DNS.
- It supports interfacing with our ATMega328 module.
- It has a low cost with a small form factor.

All the above characteristics are met by the chosen ESP8266 ESP-01 [9] module. For enabling connectivity to the internet, our microcontroller interfaces with the ESP8266 microchip- equipped with a TCP/IP stack, that is utilized to transmit the readings from the microcontroller to the backend service via a home WiFi network.

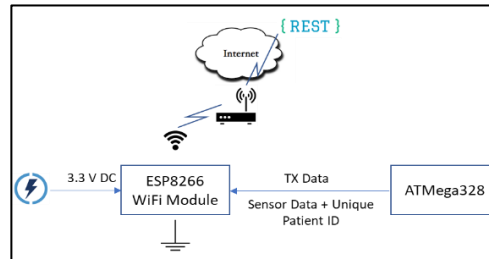
The ESP8266 works in two modes:

- Access Point Mode. In this mode, the module permits local WiFi connectivity to a patient's Smartphone/ Laptop and allows browser access to a local website running on the ESP8266 microcontroller. The website allows setting up an SSID/ Password that is used by the ESP8266 to connect to the Home WiFi network or the patient's Smartphone WiFi Hotspot.

- WiFi Client Mode: Once the patient connects to the local website running on the ESP chip and sets the SSID/ password, the ESP8266 works as a WiFi client and uses the credentials to connect to Internet using the Home WiFi/ Smartphone hotspot.

**Inputs:** Maximum +3.3V DC and GND from power supply unit, sensor data from microcontroller, SSID and password from patient (via an http request).

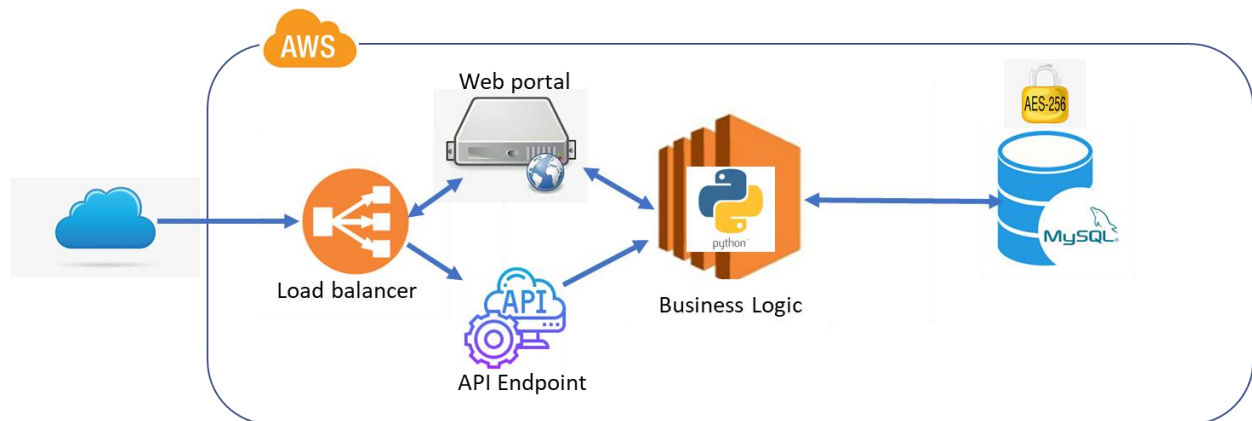
**Outputs:** Sensor data sent via HTTPS over the Internet to a cloud-based backend web service.



**Fig. 9: Block diagram of WiFi Module**

#### 2.2.4 Subsystem 4 - Patient Management Web Portal

A patient management web portal as well as API endpoint comprise the 4<sup>th</sup> Subsystem. Both are hosted on AWS cloud as depicted in Fig. 10 below.



**Fig. 10: Block diagram of Backend service**

**Load balancer:** An AWS ELB (Elastic Load Balancer) [11] has been used for providing SSL offload as well as reverse proxy function.

**API endpoint:** This python-based module provides an HTTP interface to receive sensor data from our device.

**Web Portal:** The portal has modules for user provisioning (for patients as well as health professionals), storing, capturing and the display of patient data. This application essentially displays patient vitals for monitoring by a medical professional.

The API and Web Portal modules are implemented using Python, with Crypto, Pandas, PyMySQL and Bottle libraries for ciphering, data manipulation, MySQL and web server functionality respectively. A backend MySQL database is intended for storing patient as well as user's data.

We experimented using a Django backend, but later switched to Python Bottle framework [10] due to better understanding of this framework and its capability to easily implement all needed routine (e.g. crypto, database integration, data manipulation etc.)

The web frontend visualization of sensor data as 2D plots has been implemented using the Graph.js [12] JavaScript libraries. This library was chosen as it fully met our needs and is fairly easy to implement.

## 2.3 Design Verification

Verification results of each of the above subsystems are as below:

### 2.3.1 Subsystem-1 and 2 (Control Unit and Sensors) verification:

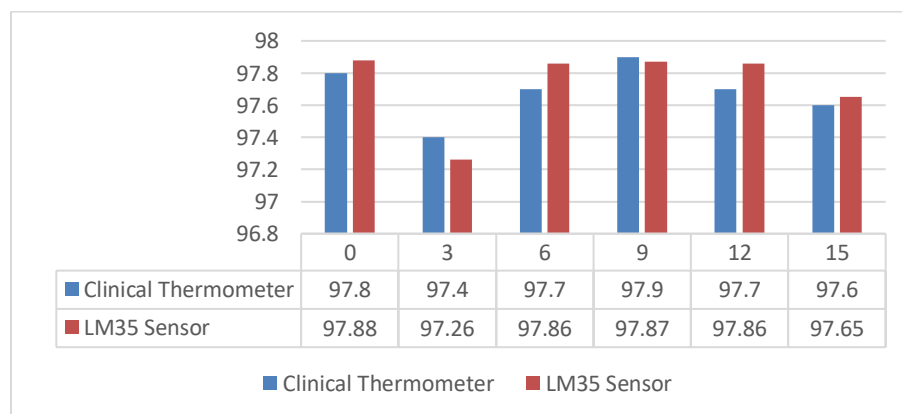
The Atmega328 micro-controller has been tested with the sensors and is performing as per design.

**The LM35 Temperature** sensor measurements are accurate and received correctly at the ATmega.

Verification: The temperature sensor was connected to an ATmega analog port as per design and its output was redirected to a laptop using the Atmega RX/TX port using a UART to USB converter. An Arduino Serial Interface Monitor software running on a PC displayed the output of the sensor as integer values proportional to the analog voltage input from the sensor. The accuracy of the readings was compared with a clinical thermometer.

Success Criteria: There was an increase in voltage readout (integer values) noted with increasing temperature and after scaling and conversion, the difference between LM35 Fahrenheit readings and the clinical thermometer have an average variation of 0.1%.

The output from the LM35 temperature sensor (Fig 11) received from the ATmega vis a vis a Clinical thermometer is as below:



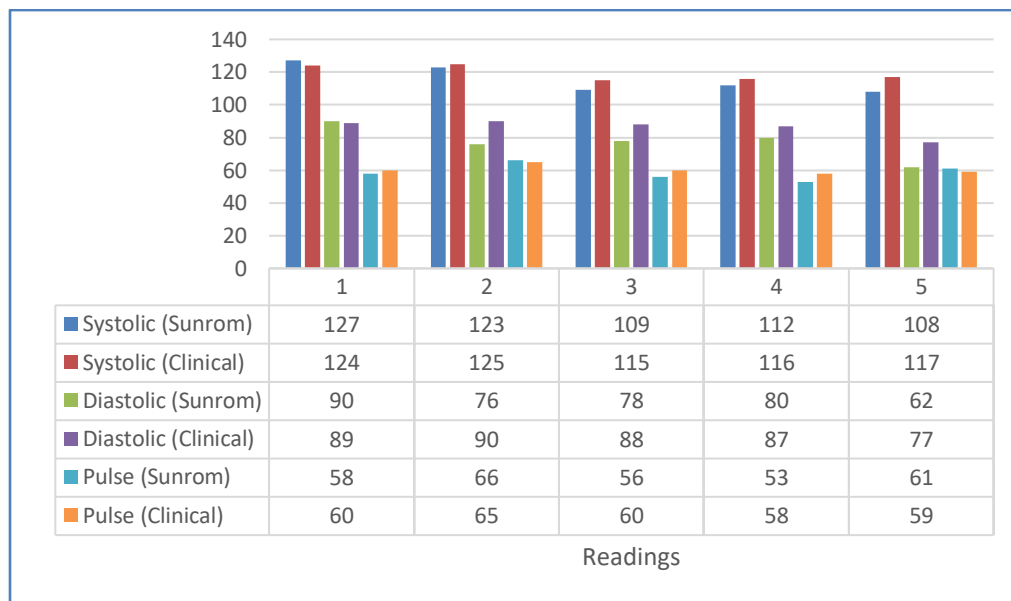
**Fig. 11: LM35 vs Clinical thermometer reading variation**

The **Sunrom BP/ Pulse sensor** measurements are accurate and received correctly at the ATmega.

Verification: The BP/ Pulse sensor UART output was received correctly in the Serial Interface Monitor software connected to the ATmega RX/TX pins and the readings were compared with a clinical BP/ Pulse device.

Success Criteria: Value of Systolic, Diastolic and Pulse rate were correctly received at the ATmega and had a minimal variation (Systolic: 0.18%, Diastolic: 1.28%, Pulse: 1.35%) compared to the clinical device.

The output (Fig. 12) from the Sunrom sensor received from the ATmega vis a vis Clinical BP device is as below:



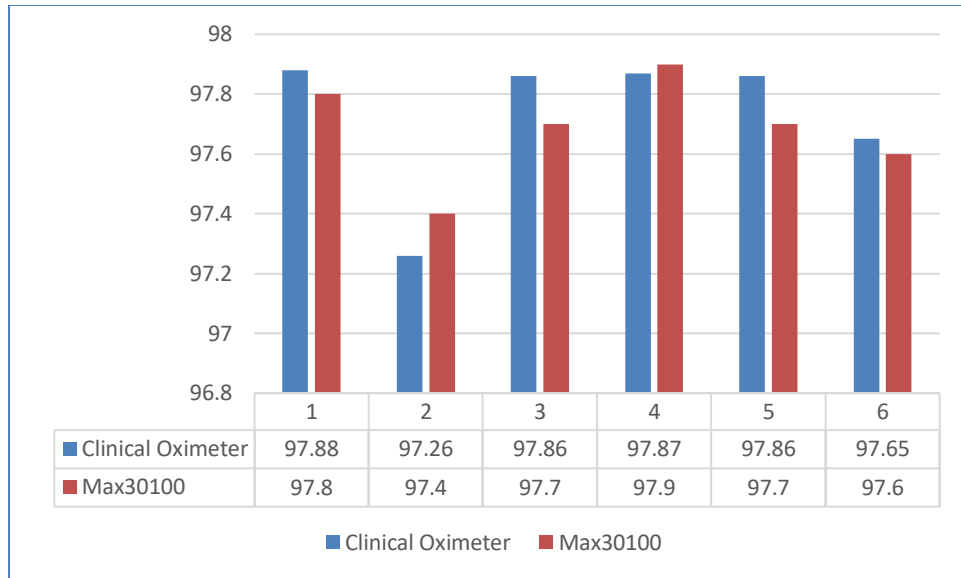
**Fig. 12: Sunrom vs Clinical BP/ Pulse sensor reading variation**

**The Max30100 Blood Oxygen (SpO2) sensor** measurements are accurate and received correctly at the ATmega.

Verification: The Max30100 sensor I2C output was received correctly in the Serial Interface Monitor software connected to the ATmega RX/TX pins and the readings were compared with a clinical SpO2 device.

Success Criteria: The SpO2 reading were correctly received at the ATmega and had a minimal variation (1.69%) compared to the clinical device.

The output (Fig. 13) from the Max30100 sensor received from the ATmega vis a vis a Clinical Oximeter is as below:



**Fig. 13: Max30100 vs Clinical Oximeter sensor reading variation**

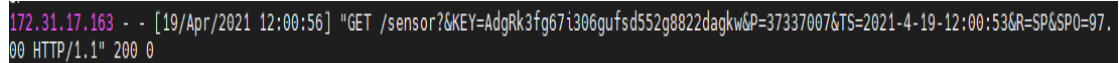
### 2.3.2 Subsystem- 3 (WiFi Module) verification:

The ESP8266 WiFi module was programmed, connected to the ATmega microcontroller via UART interface to receive data, and the output was sent to our AWS cloud-based API endpoint via the home WiFi network.

Verification:

- The Atmega output (sensor data) was received correctly at the ESP WiFi module and verified using the Serial Interface Monitor software running on a PC connected to the ESP module via a UART to USB convertor.
- Configuring the Home WiFi SSID and Password on the ESP8266:
  - On first time startup, the ESP Module tested if it could connect to any home WiFi network. On failure (as it did not know which WiFi to connect to), it created a local Access Point with SSID "ECE445Team23" /password "password" and started a local HTTP listener with a web portal to permit the patient to set the home WiFi credential.
  - We could connect to the ECE445Team223 WiFi Access Point via a mobile handset, and the mobile handset browser opened automatically and redirected up to the ESP hosted website.
  - We could thereafter successfully select and set the Home WiFi SSID and password.
  - The ESP stored the Home WiFi SSID/ password in its EEPROM memory and connected to the Home WiFi.
  - On next startup, the ESP automatically retrieved the credentials from its EEPROM and connected to the Home WiFi using the stored credentials.

- During ESP firmware development, all the above steps were tested gradually as the development proceeded via operating the ESP in a debug mode and watching the output on our Serial Interface Monitor, and finally tested and demonstrated in a real-world test case.
- Connecting to the API endpoint using HTTPS:
  - After successfully connecting to the Home WiFi, the ESP established TLS connectivity to our API endpoint hosted on the AWS cloud at URL <https://covidlb-1830690526.us-east-2.elb.amazonaws.com/sensor> TCP port 443.
  - The embedded SHA2 fingerprint of the API endpoint in the ESP firmware was used to validate the self-signed API endpoint (AWS Elastic Load Balancer) server certificate.
- Sending the sensor data to the portal:
  - The sensor data was correctly sent to the API endpoint using a HTTP GET request. Format of the request, for SpO2 readings for instance, was as follows: *https:// covidlb-1830690526.us-east2.elb.amazonaws.com/sensor?KEY=<APIkey>&P=<patientID>&TS=<timestamp>&R=<sensor type>&SPO=<dd.dd>*
  - The http request was received correctly at the API endpoint (python “bottle framework” based webserver) as displayed in the webserver logs (Fig 14).



```
172.31.17.163 - - [19/Apr/2021 12:00:56] "GET /sensor?KEY=AdgRk3fg67i306gufsd552g8822dagkw&P=37337007&TS=2021-4-19-12:00:53&R=SP&SPO=97.00 HTTP/1.1" 200 0
```

**Fig. 14: Sensor data received at AWS hosted API endpoint**

Success Criteria: The data sent by the sensors via the ATmega and ESP8266 was correctly and instantaneously (< 150ms delay) received at the API endpoint.

### 2.3.3 Subsystem- 4 (Patient Management Web Portal) verification:

The API endpoint and patient management web portal comprise the backend Subsystem-4 and are hosted on the AWS cloud. The web portal is for permitting secure login to medical personnel, provisioning patient data and viewing patient sensor data (health parameters) for taking appropriate action. The API endpoint receives sensor data from our devices and populates a backend MySQL database, such that the data can be viewed by medical personnel via the web portal.

Verification:

- API endpoint
  - TLS (HTTPS) endpoint was successfully set up using an AWS ELB (Elastic Load balancer). The ELB also acts as a reverse proxy to our python-based web application hosted on an AWS VM. A self-signed certificate was generated and used for provisioning the ELB correctly using Amazon Certificate Manager (ACM) [13] services.

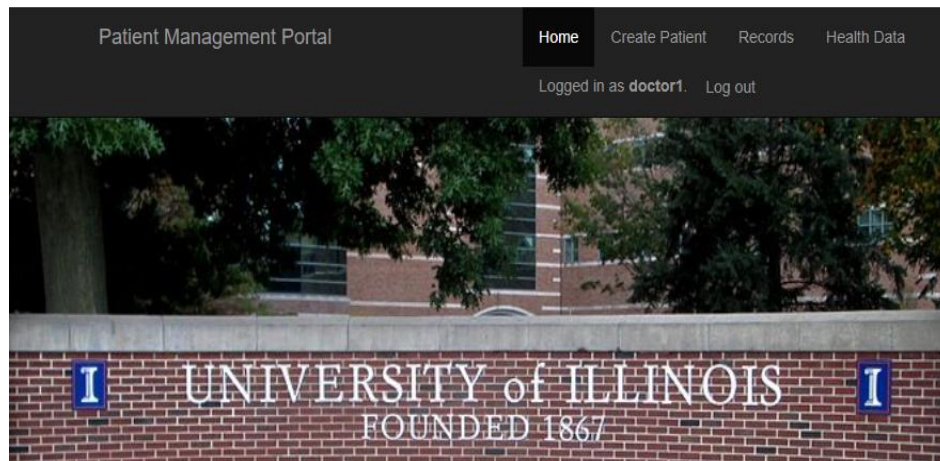
- The web application received sensor data sent by the device, authenticated the device via the API Key parameter, extracted the data from the HTTP GET request, and then stored the data in our MySQL database into the appropriate tables, one for each sensor (Fig. 15 below).

```
mysql> select * from sensor_b;
```

ptid	tstamp	sys	dia	pul
37337007	2021-4-17-16:43:09	94	121	72
37337007	2021-4-18-10:13:09	185	128	63
37337007	2021-4-18-10:14:12	147	106	58
37337007	2021-4-18-10:15:15	150	122	57
37337007	2021-4-18-10:16:47	127	90	58
37337007	2021-4-18-10:17:50	123	76	66
37337007	2021-4-18-10:19:03	109	78	56

**Fig. 15: Patient sensor data in the Mysql database**

- Web Portal (Fig. 16)
  - Medical personnel authenticated access: Username/password based authenticated access was implemented for medical personnel. HTTP Session Cookies were used to maintain persistence. The password was saved to the database after creating a SHA2 Salted hash.



**Fig. 16: Patient management web portal**

- A form was implemented to enable medical personnel to create patient records. The patient data was store encrypted (Fig. 17 below) using AES256 to ensure security and privacy of patient's Personally Identifiable Information (PII).

```
mysql> select * from patient;
```

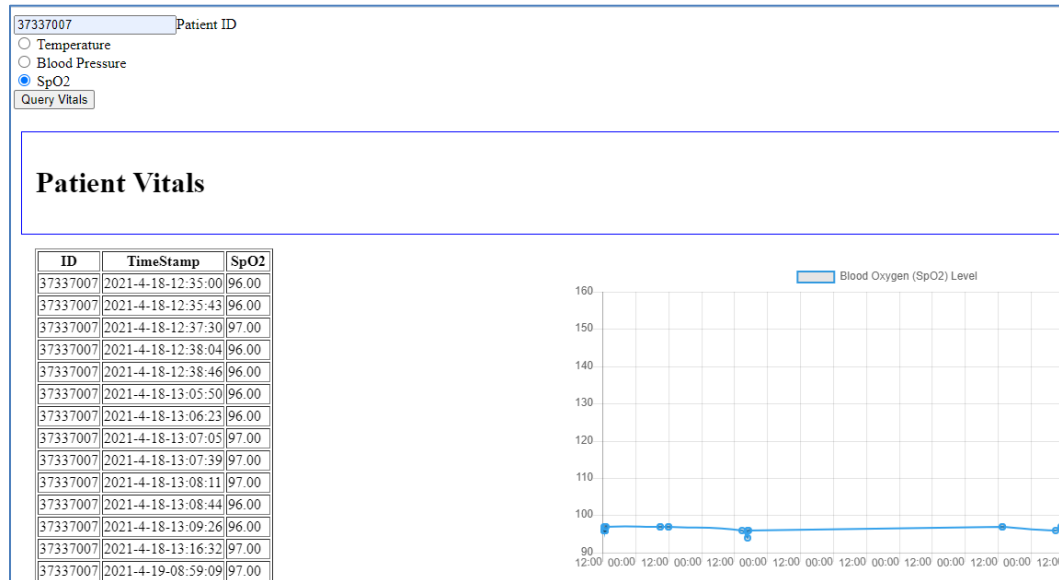
ptid	ptname	ptaddr
37337007	jHxeU9YBsFK3i/KGnDxFDQ==	59+c5dzPuFBC4hqzukJJAnEaiehgTUIZLNUw4sJ87lM=

1 row in set (0.00 sec)

**Fig. 17: Patient PII data stored after AES256-bit encryption**



- Patient health parameters were displayed for information of authenticated medical personnel via a web page. The data was displayed in a tabular as well as graphical form (Fig. 18 below) for ease of visualization.



**Fig. 18: Patient data visualization for medical personnel**

### 3. Costs

Cost of Labor: Cost of labor of the three team members is as per Table 1 below.

Name	Hourly Rate	Hours	Total
Ishaan Datta	\$30	320 (@20 hours a week)	\$9600
Arnav Ahluwalia	\$30	320 (@20 hours a week)	\$9600
Rohit Kumar	\$30	320 (@20 hours a week)	\$9600
<b>Total Cost of Labor:</b>		Total x 2.5	<b>\$70,000</b>

**Table 1: Cost of parts**

Cost of device components: Cost of the device components is as per Table 2 below.

No.	Name	Total Price
1	Amtel ATmega328P	\$2
2	Crystal Oscillator	\$0.14
3	Capacitors 0.1uF	\$0.08
4	Two pin tactile Micro switch	\$0.15
5	HatchnHack Two pin LED	\$0.01
6	Texas Instruments LM1117	\$0.38
7	Robu KA7805	\$0.27
8	Capacitors 10uF and 470uF	\$0.20
9	Robu DC-005 5.5x2.1mm Female DC Power Jack Supply Socket	\$0.14
10	TI- LM35 temperature sensor	\$1.3
11	Sunrom Blood Pressure and Pulse Rate Sensor	\$40
12	MAX30100 based Pulse Oximeter	\$2.47
13	ESP8266 ESP01	\$1.55
14	LCD-016M002B	\$1.1
<b>Total</b>		<b>\$49.79</b>

**Table 2: Cost of parts**

Grand Total: The overall cost of the project is as per Table 3 below.

Section	Total
Cost of labor	\$70,000
Cost of parts	\$48.69
<b>Grand total</b>	<b>\$70,048.69</b>

**Table 3: Grand Total**

## 4. Conclusion

The patient remote monitoring device and backend system has been implemented end-to-end and has met all the planned requirement.

Considering the re-occurrence of pandemic in many parts of the world, this system has immense potential to reduce the crushing load on hospitals.

Our original intended purpose of implementing the system was to allow patients to be monitored at their homes. Considering the current humanitarian crises situation in India and Brazil [14] that are facing a massive second wave of Covid-19 infections, we can now foresee a new use case for this device and system. Even within hospitals, we are *now observing an acute shortage of nursing staff [15] within hospitals* to the extent that many patients even within hospitals are not attended to for hours. Besides home usage, this device can also be used *within hospitals* to monitor health of all admitted patients from a central console or nursing station, such that scarce resources like Oxygen Cylinders can be given to patients the moment they need it.

## 5. References

- [1] Home Quarantine patients being monitored on phone. Available: <https://www.thehindu.com/news/national/covid-19-316-lakh-people-in-quarantine-across-india-highest-number-in-up/article32114808.ece>
- [2] ATmega328P datasheets. Available: [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf)
- [3] EasyEDA circuit design tool. Available: <https://easyeda.com/editor#id=16e896f3789e462592edc0b9aee25a68|!8a04d92177e14a799b14e839e7963013|da3cc5d37c654936bc88171ba976a14c>
- [4] KA7805 datasheet. Available: <https://www.alldatasheet.com/datasheet-pdf/pdf/99445/FAIRCHILD/KA7805.html>
- [5] LM1117 datasheet. Available: <https://www.onsemi.com/pdf/datasheet/lm1117-d.pdf>
- [6] LM35 Temperature sensor. Available: <https://www.ti.com/lit/ds/symlink/lm35.pdf>
- [7] Sunrom Blood Pressure sensor. Available: <https://www.sunrom.com/p/blood-pressure-sensor-serial-output>
- [8] Interfacing Max30100 SpO2 sensor. Available: <https://www.teachmemicro.com/max30100-arduino-heart-rate-sensor/>
- [9] ESP01 datasheet. Available: <http://www.microchip.ua/wireless/esp01.pdf>
- [10] Python Bottle Framework. Available: <https://bottlepy.org/docs/dev/>
- [11] AWS Elastic Load Balancing. Available: <https://aws.amazon.com/elasticloadbalancing/?whats-new-cards-elb.sort-by=item.additionalFields.postDateTime&whats-new-cards-elb.sort-order=desc>
- [12] JavaScript Plotting library Graph.js. Available : <https://www.chartjs.org/>
- [13] AWS Amazon Certificate Manager. Available: <https://aws.amazon.com/certificate-manager/>
- [14] India and Brazil Covid Crisis. Available: <https://theprint.in/world/brazil-hunts-for-vaccines-as-indias-covid-crisis-slows-deliveries-globally/648716/>
- [15] Shortage of nursing staff in India. Available: <https://indianexpress.com/article/india/lack-of-doctors-and-nurses-to-treat-covid-patients-will-be-the-next-big-crisis-dr-devi-shetty-7298066/>