

System for Remote Health Management of Quarantined Patients

ECE 445 Design Document

Team 23 — Arnav Ahluwalia, Rohit Kumar and Ishaan Datta

TA: Yichi Zhang

Spring 2021

Table of Contents

Para	Topic	Page No.
1	Introduction	2
1.1	Background	2
1.2	High Level requirements	2
2	Design	2
2.1	Block Diagram	3
2.2	Physical Design	4
2.3	Block Descriptions	5
2.3.1	Subsystem 1- Control Unit and Power Supply	5
2.3.2	Subsystem 2- Sensors for measuring patient health data	7
2.3.3	Subsystem 3- WiFi Module	8
2.3.4	Subsystem 4- Patient Management Web Portal	9
3	Requirements and Verification	11
3.1	Requirements Verification for ATmega386 Microcontroller	11
3.2	Requirements Verification for power supply	12
3.3	Requirements verification for TI LM35 Temperature sensor	12
3.4	Requirements verification for Sunrom Blood Pressure and Pulse rate sensor	12
3.5	Requirements verification for Max30100 Blood Oxygen sensor	13
3.6	Requirements verification for ESP8266 WiFi subsystem	12
4	Tolerance Analysis	14
4.1	Tolerance Analysis for microcontroller	14
4.2	Tolerance analysis for power supply	14
4.3	Tolerance Analysis for LM35 Temperature sensor	14
4.4	Tolerance Analysis for Sunrom Blood Pressure and Pulse rate sensor	15
4.5	Tolerance Analysis of Max30100 Blood Oxygen sensor	15
4.6	Tolerance Analysis of ESP 8266 WiFi module	15
5	Safety and Ethics	16
5.1	Cost and Schedule	17
5.2	Cost Analysis	17
6	Schedule	18
7	References	19

1. Introduction

1.1. Objective:

The COVID-19 pandemic has brought to light a wide variety of infrastructure shortcomings in the healthcare industry. Hospitals have been unable to consistently provide services to the high influx of patients, a majority of whose condition isn't critical enough to be admitted. 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. Generally speaking, this problem can be broken down to a naturally existing low physician-to-population ratio. 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 how overwhelming the situations would become for the medical infrastructure [1]. Our objective is 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.

1.2. Background:

During the pandemic, monitoring the health of home quarantined patients at scale enables provision of timely medical support. 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. 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. This is particularly true for developing countries (places where the medical infrastructure isn't expansive enough to cover all patients).

1.3. High-level requirements:

1.3.1. The device should be compact, affordable and be able to accurately measure patient vitals like pulse rate, blood oxygen, temperature and blood pressure with an accuracy greater than 95%.

1.3.2. The device should communicate with an internet-based backend service via a Home WiFi network to transmit the measured readings with zero data loss.

1.3.3. The backend service should store patient data securely and help medical personnel in monitoring patient status on near-Realtime basis.

2. Design:

Our design for the patient monitoring system is composed of four overall subsystems. The first comprises of a microcontroller and power supply. We'll be using the ATmega328 microcontroller as it has a low cost, is easy to program and has sufficient digital and analog I/O pins to interface with the sensors. All other components have likewise been selected to be cost-efficient and compatible with our microcontroller. The second subsystem are three sensors for measuring patient health parameter. The third subsystem is our WiFi module, used by our controller to connect to the patient's home WiFi network. It would enable transmission of sensor data over the internet. The last subsystem consists of a cloud-based backend service intended for receiving and storing health data from the sensors. It would also provide a front-end for medical personal to monitor the health of patients remotely.

2.1. Block Diagram:

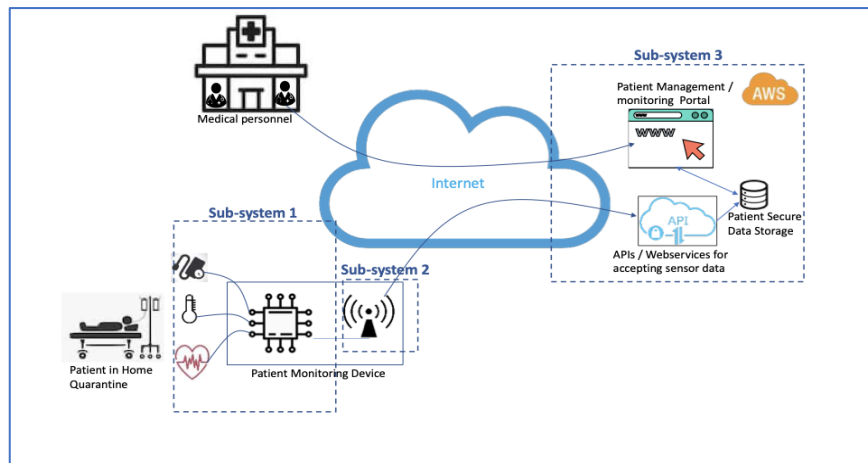


Fig. 1. Overall System Deployment

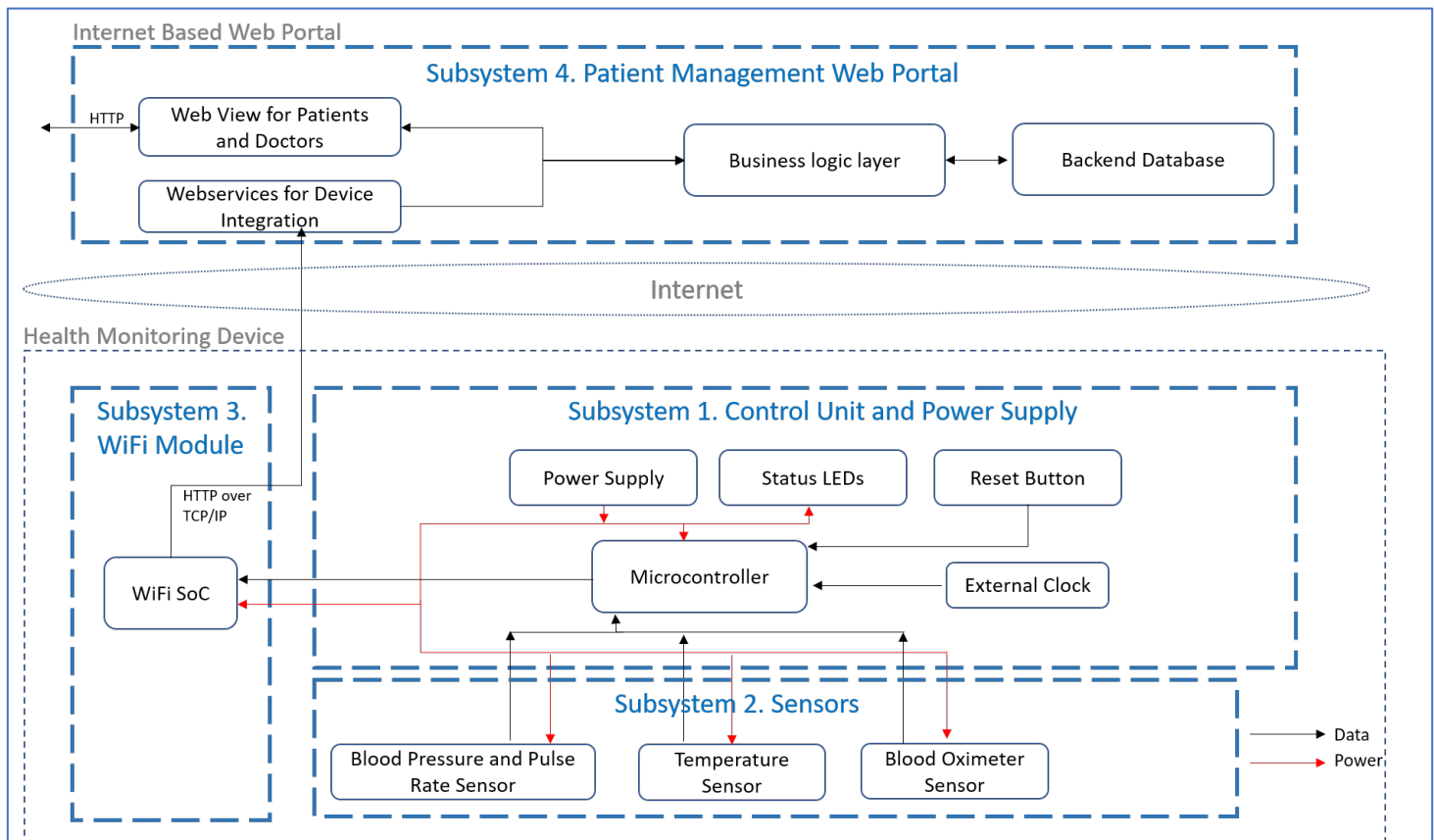


Fig. 2. Overall Block Diagram

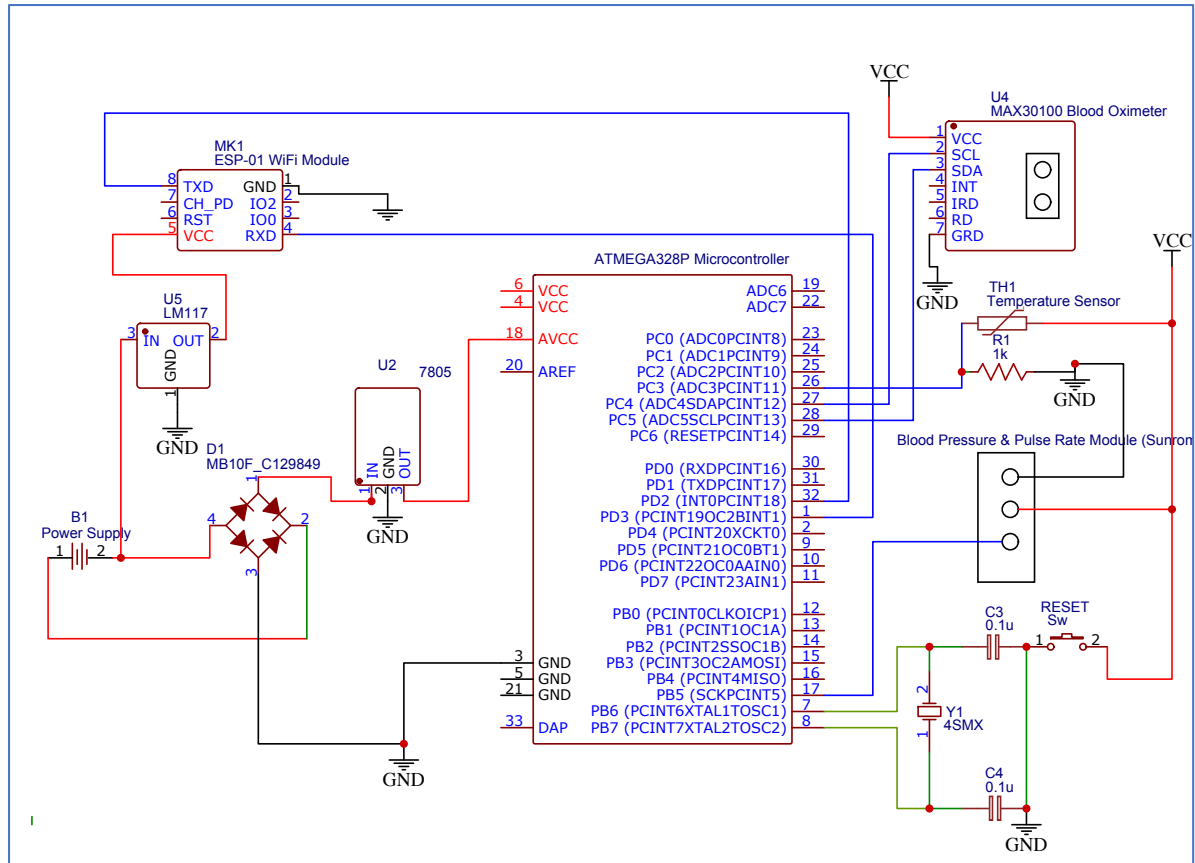


Figure 3: Subsystem 1 & 2 Schematic

2.2. Physical design:

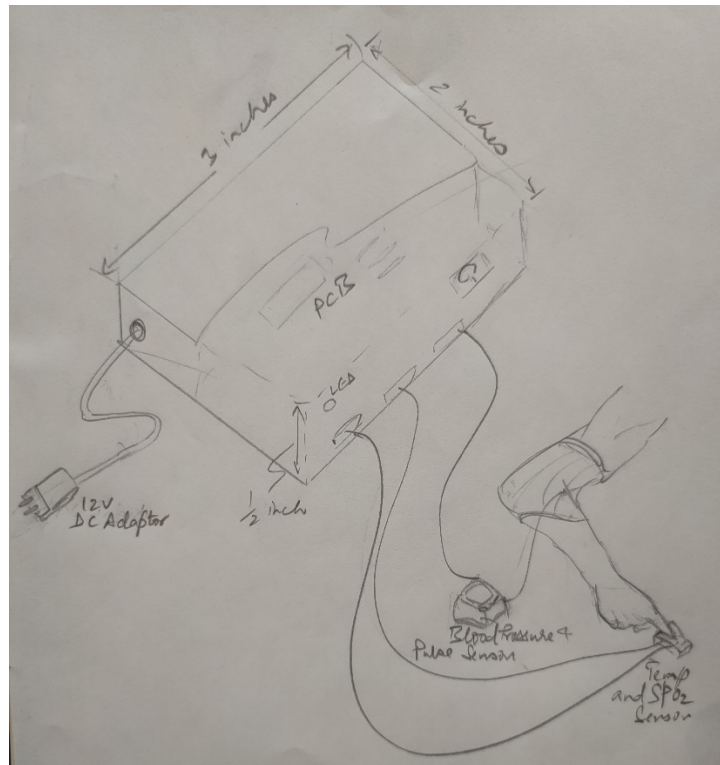


Figure 4: Physical Design (not to scale)

2.3. Block Descriptions:

2.3.1. Subsystem 1- Control Unit and Power Supply

(a) Control Unit (ATMEGA 328 PU) [2]: The microcontroller 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, Temperature sensor- analog input (0 to 10mV), Blood Pressure and Pulse sensor- digital input (3V serial data), Blood Oximeter (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, LED output.

Once the device is powered on and reset button is pressed, the microcontroller polls the I/O pins for sensor data, captures the same when available and sends the data onwards to the WiFi module.

A series of values of the temperature sensor output are detected at intervals of 0.1 secs on an Analog GPIO pin (PC3) of the microprocessor for at least 30 secs (till the temperature stabilizes), and converted into centigrade via division by a scaling factor. The BP and Pulse sensor output (3V, UART serial data at 9600 bps) is captured by a microcontroller Digital IO pin (PB5) once every 2 secs. Systolic, Diastolic and Pulse Rate data are received from the sensor as ASCII integer value.

The Blood Oximeter sensor data is captured via two Atmega I2C pins, A5 configured as I2C clock (SCL) and A4 for the data (SDA), configured at 9600 bps. Thereafter the captured sensor data is combined with a unique device 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.

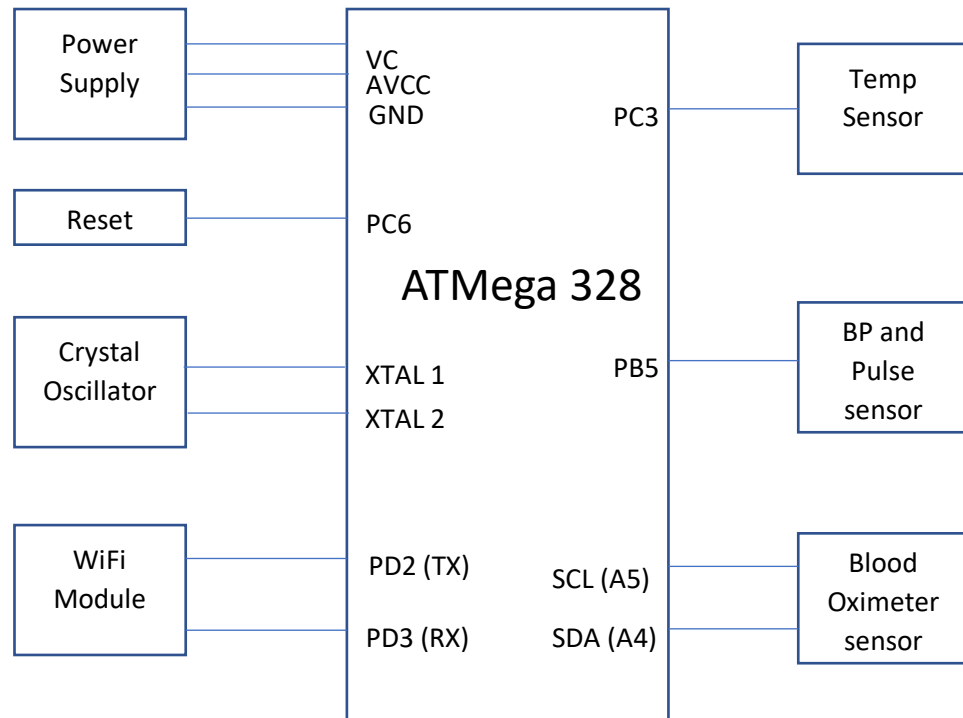


Figure 5: ATmega328 interface with other components

ATmega Pin	Function	Connection
1 (PC6)	RESET signal	Reset button
4 (PD2)	Serial send	To WiFi Module RxD
5 (PD3)	Serial receive	To WiFi Module TxD
9	Crystal Oscillator Ext Clock input 1- XTAL1	To oscillator
10	Crystal Oscillator Ext Clock input 2- XTAL2	To oscillator
7, 20	Power Supply VCC/AVCC – Digital & Analog	To +5V regulated supply
8,22	Ground	To ground of power supply
26	Temperature Sensor	To Temp sensor
28	Blood Oximeter Sensor SCL – IC2 Clock	To Oximeter SCL
27	Blood Oximeter Sensor SDA – IC2 Data	To Oximeter SDA
19	Blood Pressure and Pulse Sensor	To BP/ Pulse sensor

Table 1: Pin Connection on ATmega328

(b) Power Supply Unit: The power supply unit 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 KA7805 [3] regulator can provide up to 1 Amps of current (with heat sink) at 5V DC. The LM 1117 [4] regulator provides up to 800mA at 3.3V DC with heat sink. Power requirements of all the modules being fed by the two regulators are as below:

To Device	Power Requirements
ATMega386 microcontroller	1.8V to 5.5V DC, Max 12mA/ Typical 5.2mA
ESP8266 WiFi module	Max 3.3V DC, 80mA
MAX30100 Pulse Oximeter	3.3V DC, 20mA
Blood Pressure/ Pulse Rate Sensor	5V DC, 200mA
Temperature Sensor	5V DC, 60 μ A
TOTAL power needed	215mA at +5V DC, 100mA at +3.3V

Table 2: Power requirements from all modules

Block diagram of the power module is as below:

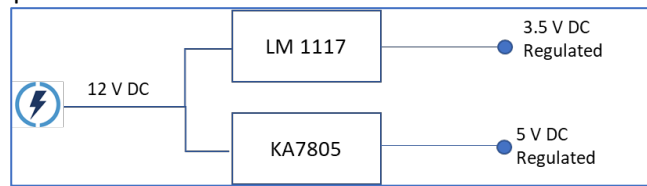


Figure 6: Block diagram of the power module

Schematic diagram of the regulated power supply is as below:

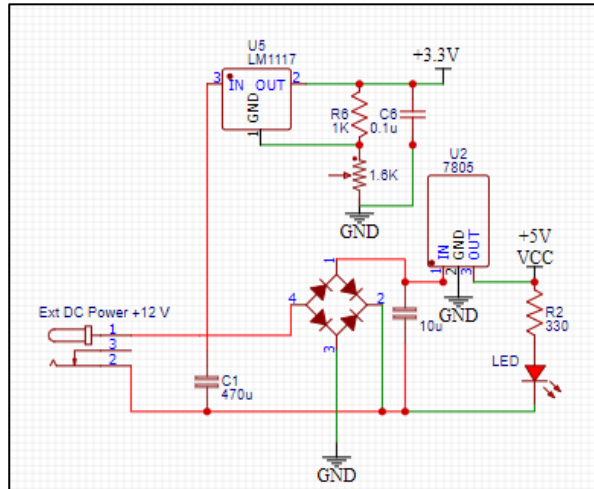


Figure 7: Power supply module, regulated 5V and 3.5V DC

2.3.2 Subsystem 2- Sensors for measuring patient health data: We propose to use BP/ Pulse Rate, Blood pulse oximeter and Temperature measurement sensors in our project.

(a) Temperature Sensor: We've chosen to use the TI- LM35 [5] temperature sensor, which 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.

Block Diagram:

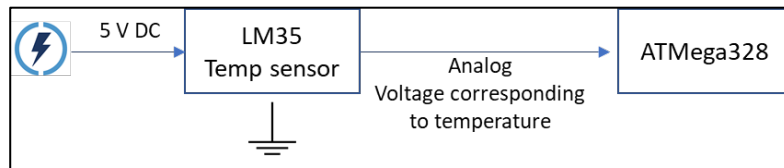


Figure 8: Block diagram of Temperature sensor

(b) Blood Pressure and Pulse Rate Sensor: An over-the-counter Sunrom blood pressure sensor [6] would be utilized to measure patient's systolic/diastolic blood pressure and pulse rate. The patient would need 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: +3.3V 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.

Block Diagram:

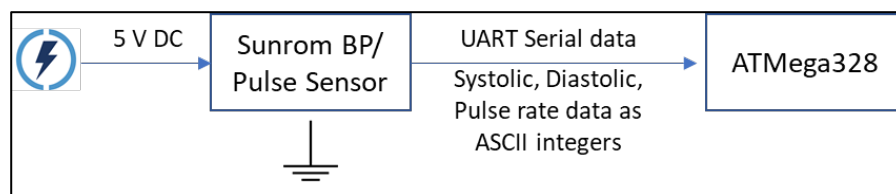


Figure 9: Block diagram of Blood Pressure/ Pulse rate sensor

(c) Blood Oxygen Sensor: We shall use the MAX30100 based Pulse Oximeter Integrated Circuit [7] 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:

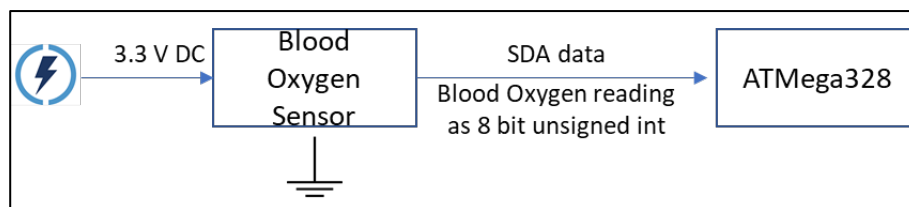


Figure 10: Block diagram of Blood Oxygen sensor

2.3.3 Subsystem 3- WiFi Module: Desired minimum characteristics of the WiFi subsystem are as follows:

- a) It should be capable of working both as a WiFi Access Point as well as WiFi client concurrently. The AP functionality is needed to set the home WiFi SSID and

password. The WiFi client function is needed connect to the Internet via the home WiFi.

- b) It should support 802.11 b/g/n protocols that are prevalent in home WiFi routers.
- c) It should a full-fledged TCP/IP stack with support for HTTP(s)/ DNS.
- d) It should support interfacing with our ATmega328 module.
- e) Should be low cost with a small form factor.

All the above characteristics are met by the chosen ESP8266 ESP-01 [8] module. For enabling connectivity to the internet, our microcontroller would interface with the ESP8266 microchip-equipped with a TCP/IP stack, this would be utilized to transmit the readings from the microcontroller to the backend service via a home WiFi network.

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 HTTP over the Internet to a cloud-based backend web service.

Block Diagram:

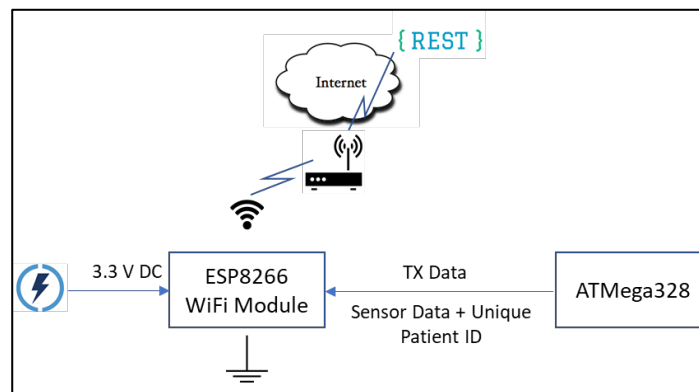


Figure 11: Block diagram of WiFi Module

Subsystem 4- Patient Management Web Portal: We will implement an application with a cloud-based Backend which would carry out user provisioning (for patients as well as health professionals), storing, capturing and the display of patient data via a website- this application would essentially store the patient data, and display it on a website for monitoring by a medical doctor. The platform should also have an email-based alerting mechanism that gets triggered by tunable health parameters obtained from the device, allowing it to indicate if/as soon as a patient needs medical attention. The Self-service web portal on the cloud to register patients and link their personal details with the unique device ID present with them. Web portal on the cloud for authorized medical personnel to view patient data. The portal would have sufficient security controls – authenticated access and encrypted data storage to keep patient data secure. The backend will run inside a docker container within an AWS EC2 instance.

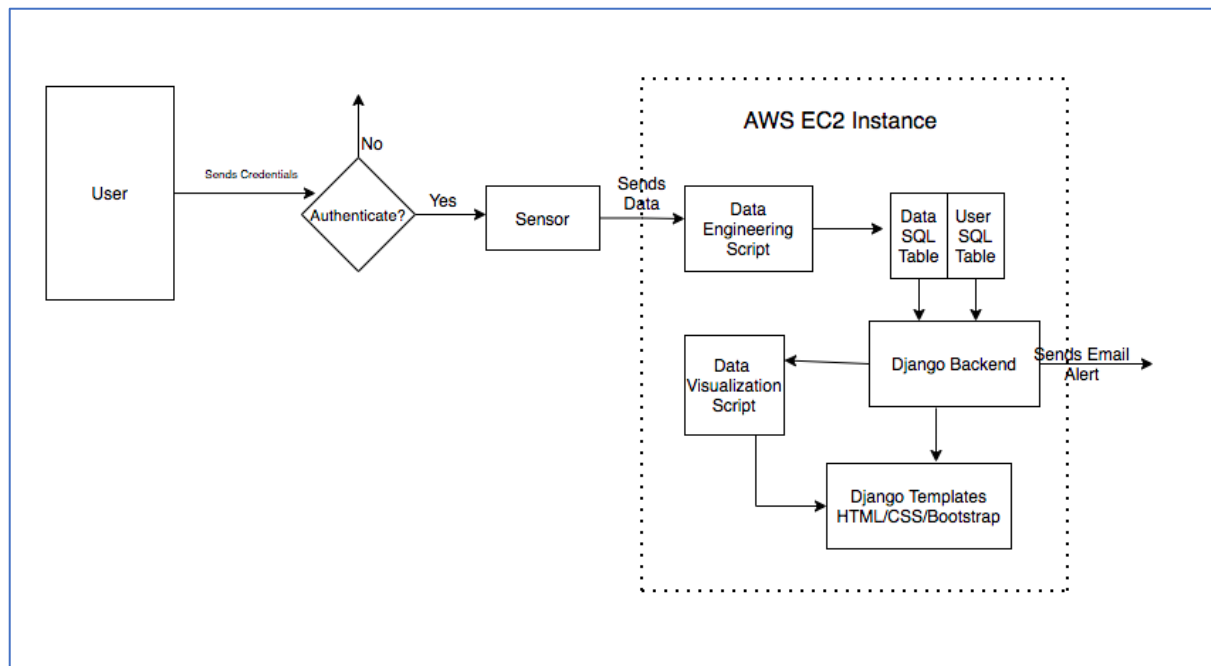
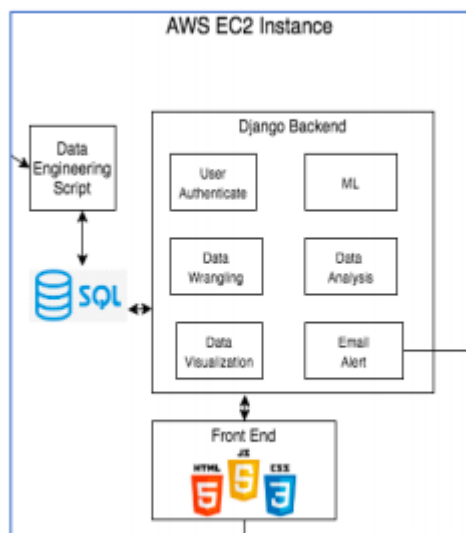
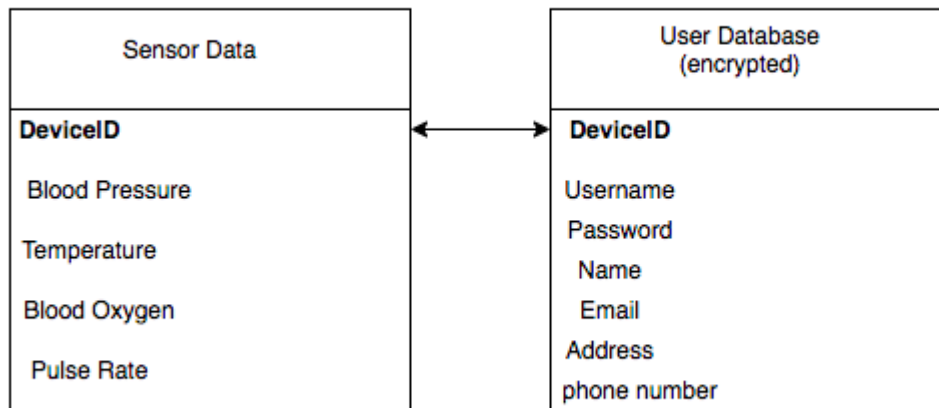


Figure 3: Subsystem 3



Data Engineering: We will have a python script to receive the sensor data over HTTP and store it in a MySQL database. Patient's personally identifiable information entered into the system by medical personnel shall be stored using AES 256 bit symmetric encryption. There will be 2 MySQL databases with a shared key, deviceID. One table will be for all the sensor data, including heart rate, blood pressure, etc. The other table will be user data, strictly seen by only the admin.



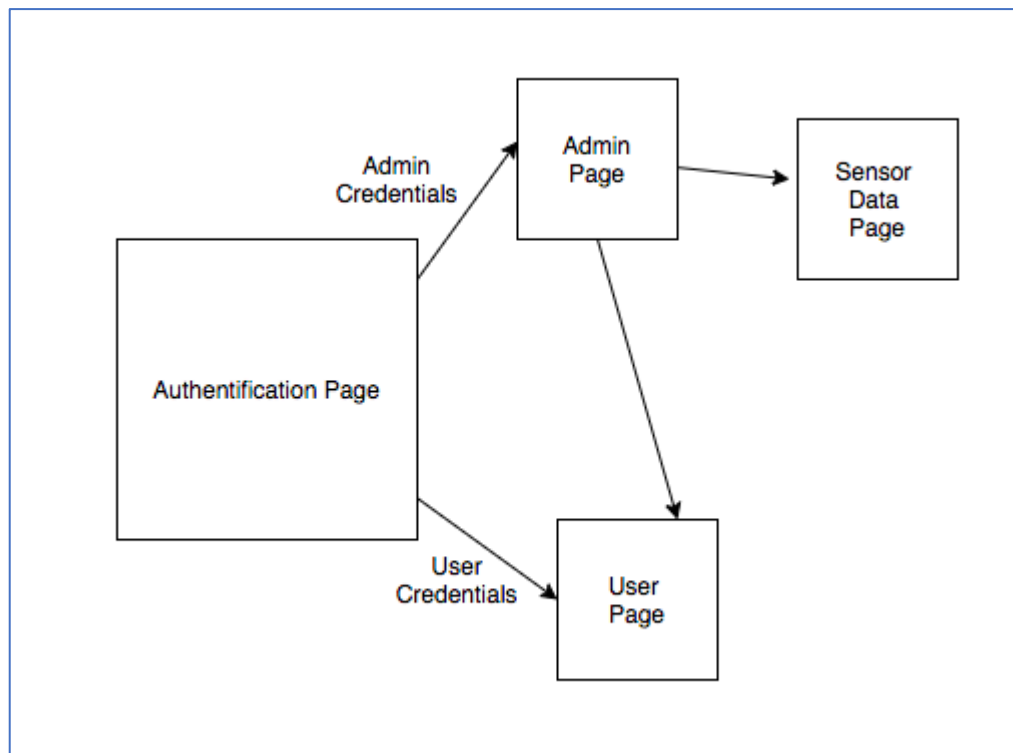
Django Core: The core of our backend will be run on a Django web application. This is where we have our portal, user authentication, data visualization & analytics, a script that performs an email alert system whenever medical attention is needed. Django has a built-in user authentication system to check for username and password and a built-in admin page.

Using the Django.Core.Mail API, I will send email notifications whenever the temperature, blood pressure, blood oxygen or pulse rate cross a new worse threshold. If any of the sensor readings are in a different category from the last time they were recorded, we send an email alert. Here are the different categories for the alert system.

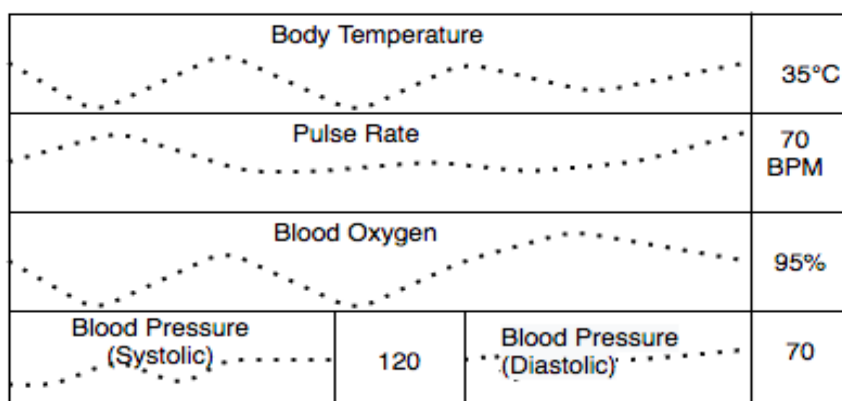
	Pulse Rate	Blood Pressure	Temperature	Blood Oxygen
Well Below Average	Less than 60	Less than 60 (SYSTOLIC) Less than 40 (DIASTOLIC)	Less than 35	Less than 85%
Bellow Average	60-70	60-90 (SYSTOLIC) 40-60 (DIASTOLIC)	35-36.5	85-95%
Average	70-75	90-120 (SYSTOLIC) 60-80 (DIASTOLIC)	36.5-47.5	95-100%
Above Average	75-83	120-140 (SYSTOLIC) 80-90 (DIASTOLIC)	37.5-40	N/A
Well Above Average	83+	140+ (SYSTOLIC) 90+ (DIASTOLIC)	40+	N/A

Front End: There will be 4 different types of webpage. The first will be the authentication page. This will be the builtin authentication page used by django. Through the authentication page, you log your credentials. If you log in with admin credentials, you can each get a view of each user page or you can click on the sensor data page. The sensor data page should just be one table with all the

data given by each sensor. This includes the encrypted user information. Admin also has the ability to create a user and log data. These pages will be using bootstrap, HTML, CSS and Javascript on top of Django Templates.



The visualization shown on the user page will be 6 different line graphs representing the 5 different sensor readings (2 for blood pressure). The line graphs will change colors depending on what category the last reading was. If the reading is in the healthy range, it will be green, slightly unhealthy will be yellow, really unhealthy will be red. Next to each line graph the last reading taken will be shown.



3. Requirements and Verification:

3.1 Requirements Verification for ATmega386 Microcontroller:

Requirement	Verification
1. The microcontroller gets input from the Temperature sensor	Connect a voltmeter to microcontroller PIN 26 and GND. Hold the temperature sensor against a warm object. Check for voltage increase at the rate of 10mV per °C. Note: The final readout can be calibrated using an external clinical thermometer.
2. The microcontroller gets input from the BP/ Pulse sensor	- Connect a pair wires from PIN 17 and GND of the microcontroller to the serial (COM port) of a laptop/PC. - Use a Terminal program to check for UART signals at the microcontroller pin.
3. The microcontroller gets input from the Blood Oximeter sensor	SCL (Clock signals) check: Connect an oscilloscope to PIN 27 and GND of the microcontroller and check for clock signals at 400kHz. SDA (Data) check: Place a finger on the Oximeter sensor. With an oscilloscope connected to PIN 28 and GND of the microcontroller, check for received data signals.
4. The microcontroller gets input VCC and GND from the power supply	Check +5 V VCC and GND via connecting a voltmeter to Pins 18 and 3 of the microcontroller.
5. The microcontroller sends sensor data to the WiFi module.	- Connect a pair of wires from PIN 32 and GND of the microcontroller to the serial (COM port) of a laptop/PC. - Use a Terminal program to check for outgoing UART signals at the microcontroller pin.[9] - Reset the microcontroller and take a reading from each connected sensor in sequence. The outgoing data should be visible on the Terminal program.
6. The program (sketch) loaded on the microcontroller is working as desired.	Add a debugging routine to the sketch to output sensor data to a locally connected LCD panel besides sending it to the WiFi module.

Table 3: Requirement verification for microcontroller

3.2 Requirements Verification for power supply:

Requirement	Verification
1. Stable +3.3V $\pm 5\%$ DC supply at up to 200mA is being provided.	Connect a voltmeter to PIN 2 and GND of the LM1117 chip and check voltage/ current drawn at full load (all sensors connected and active)
2. Stable +5V $\pm 5\%$ supply at up to 0.5A is being provided.	Connect a voltmeter to PIN 3 and GND of the KA7805 chip ¹ and check voltage/ current drawn at full load (all sensors connected and active)

Table 4: Requirements verification for power supply

3.3 Requirements verification for TI LM35 Temperature sensor:

Requirement	Verification
1. The temperature sensor accurately measures patient temperature compared to a clinical sensor with ± 0.5 °C error.	Measure body temperature using a clinical temperature. Compare the same reading against our device.
2. The sensor voltage output increases linearly with rise in temperature.	Use a voltmeter to measure the sensor output at pin 26 of the microcontroller. Take a series of readings with the temperature steadily increasing (as measured by a clinical thermometer). Plot the reading of Temp vs sensor Output voltage on a graph

Table 5: Requirements verification of temperature sensor

3.4 Requirements verification for Sunrom Blood Pressure and Pulse rate sensor:

Requirement	Verification
1. The BP/ Pulse rate sensor accurately measures patient readings compared to a clinical sensor with less than 5% error.	Measure BP/ Pulse rate using a clinical gadget. Compare the same reading against our device.
2. The sensor data is available at its output as ASCII integers.	Connect a pair wires from output and GND of the sensor to the serial (COM port) of a laptop/PC. Use a Terminal program to check for received signals.

Table 6: Requirements verification of BP/ Pulse rate sensor

3.5 Requirements verification for Max30100 Blood Oxygen sensor:

Requirement	Verification
1. The Max30100 Blood Oxygen sensor accurately measures patient readings compared to a clinical sensor with less than 5% error.	Measure SpO2 using a clinical gadget. Compare the same reading against our device.
2. The sensor data is available at its output as 8-bit unsigned integers.	Connect an Arduino Uno to the sensor using 3.3 V VDD, SCL and SDA connectors. Use the Max30100 libraries to read I2C SDA data off the sensor. Display the SpO2 data on a serial terminal.

Table 7: Requirements verification of Blood Oxygen sensor

3.6 Requirements verification for ESP8266 WiFi subsystem:

Requirement	Verification
1. Capture patient sensor data and patient ID from the microcontroller.	Connect the ESP8266 RX pin 4 with to the serial (COM port) of a laptop/PC. Use a Terminal program to check for received signals.
2. Transmit captured data over the WiFi module to a destination IP address and port.	Start a TCP listener on a local PC/ laptop on the same WiFi. Set its IP address and port in the ESP8266 sketch as destination IP. Check for received data.
3. Run a mini-HTTP server (in an Access Point role) using which a patient can set SSID and password.	Latch on to the local WiFi AP created by the ESP8266 using a laptop. Connect to its webpage.
4. Allow the end user to log on to the device using a browser, and set the home SSID and password	Connect to the webpage hosted on ESP8266 after Step 3 and set the home WiFi SSID and password.
3. Ensure that the ESP8266 device can work in both WiFi Access Point (to set home SSID/ password), as well as WiFi client mode (to send patient sensor data to the backend cloud over the home WiFi network).	Check both Step 4 and Step 2 above concurrently.

Table 8: Requirements verification of the WiFi subsystem

3.7 Requirements verification for Patient Management Web Portal:

Requirement	Verification
1. Sensor data received from the web-application will be parsed and stored into the database.	Print Statements
2. Patient personally identifiable information (names, addresses, telephone numbers, email addresses) will be encrypted prior to storage in the database.	View the MySQL Database and see if it follows the encryption.
3. The portal for medical personnel will have authentication, searching, visualization and alerting features.	Test the system as fake user.
4. The portal should be capable of being rendered in Chrome/ Internet Explorer browsers.	Attempt to render all 4 webpages in every browser.

Table 9: Requirements verification of Patient Management Web Portal

4. Tolerance Analysis:

4.1 Tolerance Analysis for microcontroller:

The ATmega328 IC is a robust microcontroller and is capable of handling fluctuations in input voltage (1.8 to 5.5V) and temperatures (-40°C to 85° C). As it is software driven, the probability of errors being introduced by the microcontroller itself is negligible. However, the microcontroller can introduce errors in case:

- Extensive computations (stack and heap size too large, too many variables stored in memory etc.) are performed in memory (2KB SRAM).
- Sketch size is too large (32KB Flash memory).
- Data is sent to the WiFi module at an exceedingly high rate (limited SPI and I2C memory).

In our case, the patient is expected to use the device less than 5 times a day. The sensor data is *never stored on the device*. Each time, *only a few bytes of data would be captured and sent by the device to the backend service over the internet*. Hence neither speed of data transmission nor onboard storage is a critical factor in design of this project.

4.2 Tolerance analysis for power supply:

The power supply module uses any external 230V AC to 12V DC commercially available adaptor as the input source. Such adaptors may not be completely reliable and hence we need to analyze the impact on the device, of a power surge, underload or ripple, at the input of our power supply module. Besides this, the impact of a thermal overload on the out of the power supply module also needs to be checked.

The KA7805 IC provides a rated output voltage V_o of minimum 4.75V to max 5.25 V for an input variation of $V_i = 7V$ to 20V from the adaptor (current varying from 0.5 to 1A). We therefore observe that the regulated output is within our requirement of $\pm 5\%$ of 5V DC supply. Similarly, datasheet for KA7805 indicate typical ripple filtering capability (PSRR – Pulse rate Rejection Ratio) of 65dB to cater for transient ac ripple in cheap AC to DC adaptors.

The LM1117 IC has a output voltage variation V_o of 3.235V to 3.365 V for an input variation of $V_i = 4.75V$ to 15V from the adaptor (current load 1A). It is observed that the regulated output is within our requirement of $\pm 5\%$ of 5V DC supply. Similarly, datasheet for LM1117 indicate typical ripple filtering capability of 60 to 120dB to cater for transient ac ripple in cheap AC to DC adaptors.

4.3 Tolerance Analysis for LM35 Temperature sensor:

It is worth noting that clinical thermometers, the gold standard for measuring patient temperature, are usually placed orally, in the armpit or rectum of a patients for an accurate reading. The sensor used in our project measures temperature of patient's skin, similar to a non-contact IR based temperature sensors used these days in almost public areas, buildings and offices.

However, it is worthwhile using the LM35 chip as opposed to an IR non-contact sensor or an oral digital commercial thermometer for our project due to the following reasons:

- The has a high accuracy ($\pm 0.5^\circ C$).
- It is low cost of \$1.3, as compared to \$15 for an IR sensor and \$4 for a digital thermometer on average.

The same LM35 sensor can also be placed orally if encased in a suitable waterproof plastic encasing- however this is beyond the scope of our project.

4.4 Tolerance Analysis for Sunrom Blood Pressure and Pulse rate sensor:

The primary risk of incorrect BP readings may be due to incorrect usage by the patient. BP measurement typically requires the patient to sit idle, place the cuff properly with the air pressure tube facing the inner arm, pump in air for about 7 to 10 seconds and then release to get the measurement. It is envisaged that patients will need to be trained to handle the BP/ Pulse rate sensor correctly.

As we are using a commercially available external sensor for this purpose, its technical tolerance aspects are out of scope. The overall accuracy of the sensor is however important considering its implications on managing a patient's health parameters.

4.5 Tolerance Analysis of Max30100 Blood Oxygen sensor:

As is the case for other sensors used in the project, the primary risk of incorrect SpO2 readings may be due to incorrect usage by the patient. Patients will need to be trained to place a finger on the sensor correctly for about 10 secs.

The sensor uses Red and IR LEDs internally to measure SpO2 levels. Any errors in the LED sensor data received in the Max30100 IC, and thereafter converted to digital data, will be subject to errors due to interference etc. However internal technical characteristics of the module are beyond the scope of our project.

4.6 Tolerance Analysis of ESP 8266 WiFi module:

The WiFi module is important for our project as it allows internet connectivity to our device, a critical requirement for medical personnel to remotely monitor patients. However, as the amount of data sent by the module is in a few hundred bytes three to four times a day, we do not envisage any bandwidth or latency related constraints. Given that the WiFi module's lowest sensitivity is -95dBm, and that commonly used home WiFi routers have a gain of 20dBm, we assume no problems in achieving connectivity within a house for our negligible data rates. The only requirement is that the device should stay connected to the home WiFi router.

5. Safety and Ethics:

- 5.1 **Safety:** Safety considerations for the project basically include the standard set of precautions taken while working on electrical engineering projects. The device would not be working above 5V DC for any of its components. Precautions would be taken as per our lab guidelines for soldering the different components. Due to the ongoing pandemic, team members would ensure maintaining appropriate social distancing and sanitary measures while working together or in the lab.
- 5.2 **Ethics, Security & Privacy:** One of the biggest concerns with recording, transmitting and storing patient health data is maintaining privacy and security. The US HIPAA act, EU's GDPR and other such privacy regulations treat patient health data as "sensitive personal information", needing the highest level of security and privacy. This includes ensuring-

- 5.3 **Security of Data in Transit:** We plan on anonymizing the transmitted data such that it does not contain any personal details (like names, addresses, phone number, etc.). During device provisioning stage, the patient/ assistant would need to enter personal details on a provisioning (secure) website and link the device unique number with it (the unique number can be printed on the side of each device). Thereafter, all outbound data has just this number along with sensor metrics - no personal details would be sent.
- 5.4 **Security of Data at Rest:** At the cloud end, the data would be stored in an encrypted state once it is linked with personal details. We will use AES 256-bit symmetric encryption.
- 5.5 In addition, our project follows **IEEE codes of ethics** [10] as following:
- To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.
 - To be honest and realistic in stating claims or estimates based on available data.
 - To improve the understanding of technology; its appropriate application, and potential consequences.
 - To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.
 - To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.
 - To avoid injuring others, their property, reputation, or employment by false or malicious action.
 - To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

6. Cost and Schedule:

6.1 Cost Analysis

6.1.1 Labor

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
Total Cost of Labor:			\$28,000

Table 9: Cost of labor**6.1.2 Cost of device components**

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 5.5×2.1mm 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
Total		\$48.69

Table 10: Cost of parts**6.1.3 Grand Total**

Section	Total
Cost of labor	\$28,000
Cost of parts	\$48.69
Grand total	\$28,048.69

Table 11: Grand Total**6.2 Schedule**

Week	Task	Responsibility
2/8/2020	Project Approval	Ishaan
	Finalize project proposal	Ishaan
	Review proposal	Arnav and Rohit
2/15/2021	Mock Design review	
	Research and select device components	Ishaan
	Research and backend webservice components	Arnav and Rohit
2/22/2021	Prepare design document	
	Research and prepare design document for device microcontroller, power, WiFi	Ishaan

	Research and prepare design document for three sensors	Arnav and Ishaan
	Research backend webservice, database design	Rohit
	Procure device hardware	Ishaan
3/1/2021	Design Document Due	
	Finalize initial draft of design document	All
	Research hardware datasheets and circuit diagrams	Ishaan
	Research and prepare PCB design [11]	Ishaan and Arnav
	Research backend webservice, database design	Rohit
3/8/2021	Design Document Review	
	Finalize and submit design document	All
	Prepare draft circuit diagrams	Ishaan
	Research microcontroller and WiFi chip firmware sketches	Ishaan
	Research connectivity for sensors	Arnav
	Prepare sample backend web service databases and data storage encryption logic	Arnav
	Prepare sample web portal Medical personnel	Rohit
3/15/2021	Simulation and Soldering	
	Simulate device on breadboard	Ishaan
	Simulate Webservice for capturing sensor data	Rohit
	Simulate Web portal for medical personnel	Arnav
3/22/2021	PCB design and ordering	
	Design PCB and place order	Ishaan
	Test backend service	Rohit and Arnav
4/5/2021	Implementation	
	Fabricate device on PCB	Ishaan
	Implement web services, portals and databases	Rohit and Arnav
4/12/2021	Testing	
	Test Device and sensors	Ishaan and Arnav
	Test webservice, portal implementation	Rohit and Arnav
4/19/2021	Mock demo and final demo	
	Prepare device demo, presentation and final paper	Ishaan and Arnav
	Prepare Web portal demo, presentation and final paper	Rohit
5/3/2021	Final Paper	
	Prepare and review final paper	All
	Prepare and review final presentation	All

Table 12: Schedule of work

7. 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

[3] KA7805 Voltage Regulator. Available:

<https://components101.com/ics/7805-voltage-regulator-ic-pinout-datasheet>

[4] LM1117 voltage regular. Available:

<https://www.ti.com/lit/ds/symlink/lm1117.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-ww&ts=1614320716275>

[5] LM35 Temperature sensor. Available:

<https://www.ti.com/lit/ds/symlink/lm35.pdf>

[6] Sunrom Blood Pressure sensor. Available:

<https://www.sunrom.com/p/blood-pressure-sensor-serial-output>

[7] Interfacing Max30100 SpO2 sensor. Available:

<https://www.teachmicro.com/max30100-arduino-heart-rate-sensor/>

[8] ESP01 datasheet. Available:

https://www.alldatasheet.com/view.jsp?Searchword=Esp-01%20datasheet&gclid=Cj0KCQiAj9iBBhCJARIsAE9qRtAZfm8ZTEikxCaUMmkeJxIMS5LMiw3fOd6m0IKxS8MyVGBVyPJBpf4aAlcAEALw_wcB
<http://www.microchip.ua/wireless/esp01.pdf>

[9] Terminal software for testing UART interfaces. Available:

<https://hw-server.com/terminal-terminal-emulation-program-rs-232>

[10] IEEE Code of Ethics. Retrieved February, 2015. Available:

<http://www.ieee.org/about/corporate/governance/p7-8.html>

[11] EasyEDA circuit design tool. Available:

<https://easyeda.com/editor#id=16e896f3789e462592edc0b9aee25a68|!8a04d92177e14a799b14e839e7963013|da3cc5d37c654936bc88171ba976a14c>