
FINAL YEAR PROJECT - SECOND REVIEW

25-03-2022

DESIGN AND DEVELOPMENT OF IOT BASED HEALTH CARE MONITORING DEVICE FOR EARLY DIAGNOSIS OF HEART DISEASE USING AI



PROJECT GUIDE - DR. GEETHA C

GROUP 9B

TEAM MEMBERS

HEMANGANI - 110118031

AKSHYAH - 110118006

Timelines And Goalposts Achieved

Review – 0:

January

1. Formulation of problem statement
2. Literature review
3. Study of heart diseases
4. Framed solution timeline
5. Dataset collection

Review – 1:

February

1. Sensor selection and purchase
2. Fabrication, testing, calibration, and thresholding of sensors
3. Circuit design and implementation
4. C programming for data acquisition from sensors and display
5. Pre-processing, and analysis of data and feature engineering
6. Design and development of machine learning algorithms
7. Comparison with existing methods
8. Training, testing, and debugging of the ML model

Review – 2:

March

1. Addition of blood pressure using BPT algorithm
2. Developed fall detection algorithm
3. Incorporation of Wi-Fi module using NodeMCU ESP8266
4. Configuring and programming NodeMCU
5. Successful implementation of wireless communication and display in browser
6. Integrating python and C codes
7. Design of the wearable hand sensor glove
8. Documentation

Final Review Tasks:

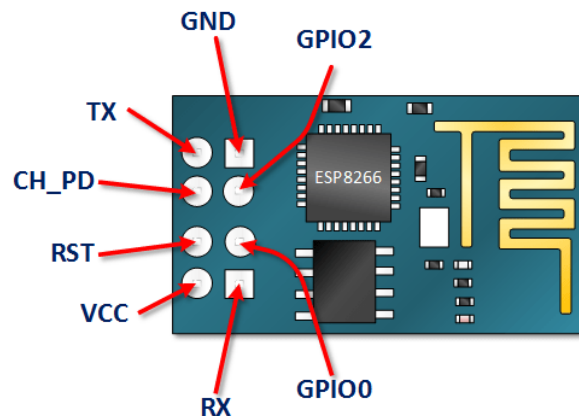
April

1. Integration of GPS and RFID
2. Designing of PCB
3. Compact fabrication with inbuilt power supply
4. Performance Analysis and verification of results
5. Testing and submission

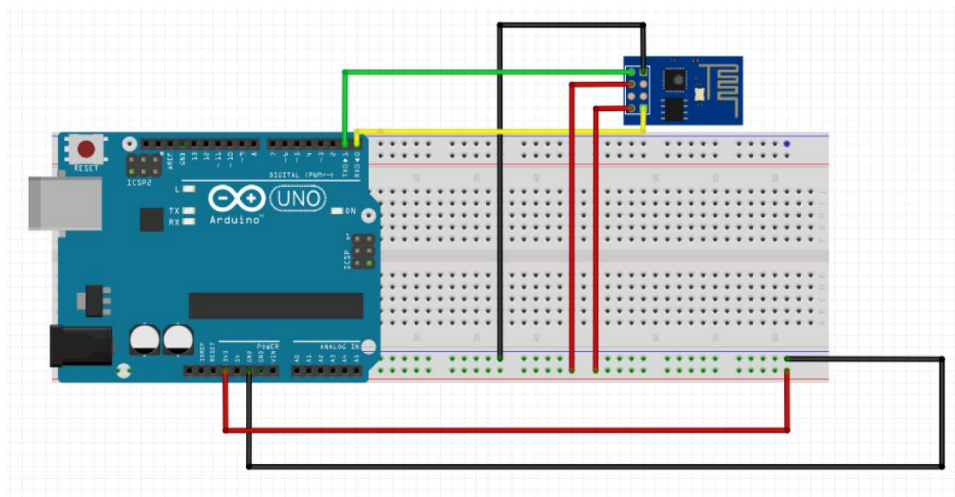
ESP8266

The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability. We can use ESP8266 in the projects for communication. It has 3 different modes. Now we will connect ESP8266 to Arduino Uno R3 and then we will set up our microchip with AT commands.

First, I did use ESP8266 and Arduino UNO R3. ESP8266 can be used in 2 different styles. In the first option we can use it as a receiver or transmitter. Second option is using ESP8266 as a microcontroller.



We will use just 5 of these 8 pins. RX and TX are our transmitter and receiver pins. We'll use them for serial communication. Last 3 pins are about the power system of ESP8266 which are GND, VCC, and CH_PD (CH_EN). As we know that GND and VCC pins are ground and voltage pins. We will use CH_PD (CH_EN) pins for enabling our microchip. Also, we can understand the right direction of the microchip by using the golden line at the right side of it.



We have to use a serial monitor at 115200 baud rate because this rate is the default option for ESP8266. Secondly, in the line options we should use Both NL & CR. If we do these steps correctly, we are ready to start to set up.

As the first step of setup, we should type "AT" in the serial monitor screen. Then we should get an "OK" response from this screen. If we get this, we will have an available ESP8266. Then we will type "AT+GMR" command to reach our version information.

Basic	
Command	Description
AT	Test AT startup
AT+RST	Restart module
AT+GMR	View version info
AT+GSLP	Enter deep-sleep mode
ATE	AT commands echo or not
AT+RESTORE	Factory Reset
AT+UART	UART configuration, [<i>@deprecated</i>]
AT+UART_CUR	UART current configuration
AT+UART_DEF	UART default configuration, save to flash
AT+SLEEP	Sleep mode
AT+RFPOWER	Set maximum value of RF TX Power
AT+RFVDD	Set RF TX Power according to VDD33

AT Commands Table for **ESP8266**

We can see that we all work at 115200 baud rate. This baud rate is too fast for us and we might want to change it to 9600 which is the optimum baud rate that is used by other components. For this change we will use the "AT+UART_DEF" command which can be seen on the table. But we will have some changes on this command. This command changes default UART configurations of our device. So, we should give some configuration parameters. "AT+UART_DEF=9600,8,1,0,0" we will use this command to change baud rate of ESP8266.

Commands	Description	Type
AT+RST	restart module	basic
AT+CWMODE	wifi mode	wifi
AT+CWJAP	join AP	wifi
AT+CWLAP	list AP	wif
AT+CWQAP	quit AP	wifi
AT+CIPSTATUS	get status	TCP/IP
AT+CIPSTART	set up TCP or UDP	TCP/IP
AT+CIPSEND	send data	TCP/IP
AT+CIPCLOSE	close TCP or UDP	TCP/IP
AT+CIFSR	get IP	TCP/IP
AT+CIPMUX	set multiple connections	TCP/IP
AT+CIPSERVER	set as server	TCP/IP

AT Commands for WiFi Configuration

The last step of the ESP8266 setup is about WiFi connection. In the table we can see AT commands for WiFi configuration. Usage of ESP8266 we need to know these commands for WiFi communication. The first command "AT+RST" is resetting our device. It is not changing settings, just reset the device.

"AT+CWMODE" this command is allowing us to change our device's usage mode. This command takes 1 parameter which should be integer.

"AT+CWMODE=mode" mode is representing the parameters and it looks like:

Modes:

1- Station mode (client)

2- AP mode (host)

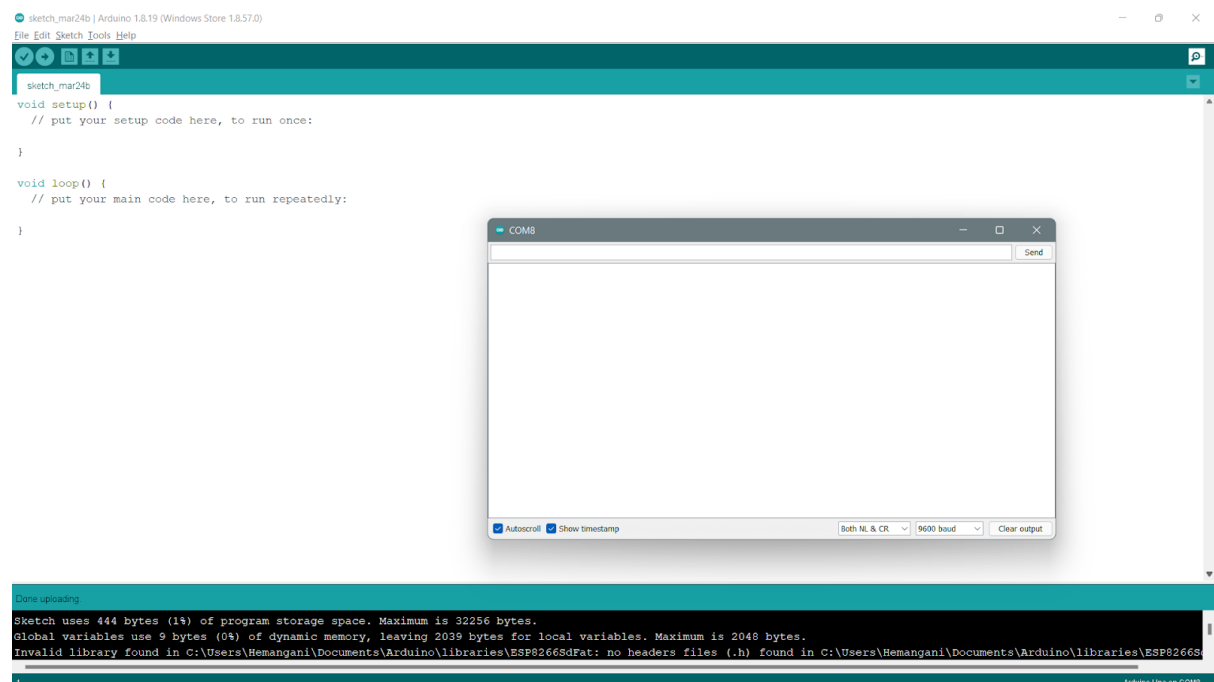
3- AP and Station mode

That means if we want to use ESP8266 as a client of our system, we should use the "AT+CWMODE=1" command for it.

Third command is about WiFi spots. With this command we can scan WiFi spots which are located nearby. "AT+CWLAP" can list all available WiFi spots for us.

The next step of setup is joining a WiFi. I did set my device's mode as station mode then I listed all available WiFi spots. Now we will join WiFi without password. For this step we will use the "AT+CWJAP" command. But we know that for joining a WiFi we need SSID and Password. We can see all available SSIDs when we scan WiFi. Now we should change this command to "AT+CWJAP="YOUR_SSID_NAME","YOUR_PASSWORD".

We will use the "AT+CIFSR" command to find our IP. We can use this IP to communicate to our device from another system. Also, this command returns us the MAC Address of ESP8266.



COM8

Send

X

17:58:46.231 -> AT

17:58:46.231 ->

17:58:46.231 -> OK

17:59:17.519 -> AT+GMR

17:59:17.519 -> AT version:0.40.0.0 (Aug 8 2015 14:45:58)

17:59:17.599 -> SDK version:1.3.0

17:59:17.599 -> Ai-Thinker Technology Co., Ltd.

17:59:17.639 -> Build:1.3.0.2 Sep 11 2015 11:48:04

17:59:17.679 -> OK

17:59:52.109 -> AT+RST

17:59:52.109 ->

17:59:52.109 -> OK

17:59:52.189 -> I(CFHMFSBZSYCSfCHRObIbJcBfKIDf

17:59:52.669 -> Ai-Thinker Technology Co., Ltd.

17:59:52.709 ->

17:59:52.709 -> invalid

17:59:54.789 -> WIFI DISCONNECT

18:00:11.897 -> AT+CMODE=1

18:00:11.937 ->

18:00:11.937 -> OK

18:00:22.760 -> AT+CWJAP

18:00:24.920 -> +CWLAP:(0,"HP",-84,"a0:d3:c1:b1:74:50",1,66)

18:00:24.960 -> +CWLAP:(0,"HP",-71,"a0:d3:c1:a5:0b:10",6,69)

18:00:25.000 -> +CWLAP:(3,"Crackers",-44,"82:1a:06:0a:65:4d",11,113)

18:00:25.040 -> +CWLAP:(0,"HP",-82,"a0:d3:c1:a5:07:30",11,72)

18:00:25.120 -> +CWLAP:(3,"DIRECT-RZSUBIRKSHA-PCmsNC",-87,"92:91:33:7d:4f:b3",1,67)

18:00:25.160 ->

18:00:25.160 -> OK

18:00:51.851 -> AT+CWJAP="Crackers","asdqwe123"

18:01:06.861 -> +CWLAP:1

18:01:06.861 ->

18:01:06.861 -> FAIL

18:01:45.914 -> AT+CWJAP="Crackers","asdqwe123"

18:01:49.114 -> WIFI CONNECTED

18:01:49.794 -> WIFI GOT IP

18:01:51.034 ->

18:01:51.034 -> OK

18:03:05.417 -> AT+CIFSR

18:03:05.417 -> +CIFSR:STAIP,"192.168.165.1"

18:03:05.457 -> +CIFSR:STAMAC,"e8:db:84:e4:b7:1f"

18:03:05.497 ->

18:03:05.497 -> OK

Autoscroll

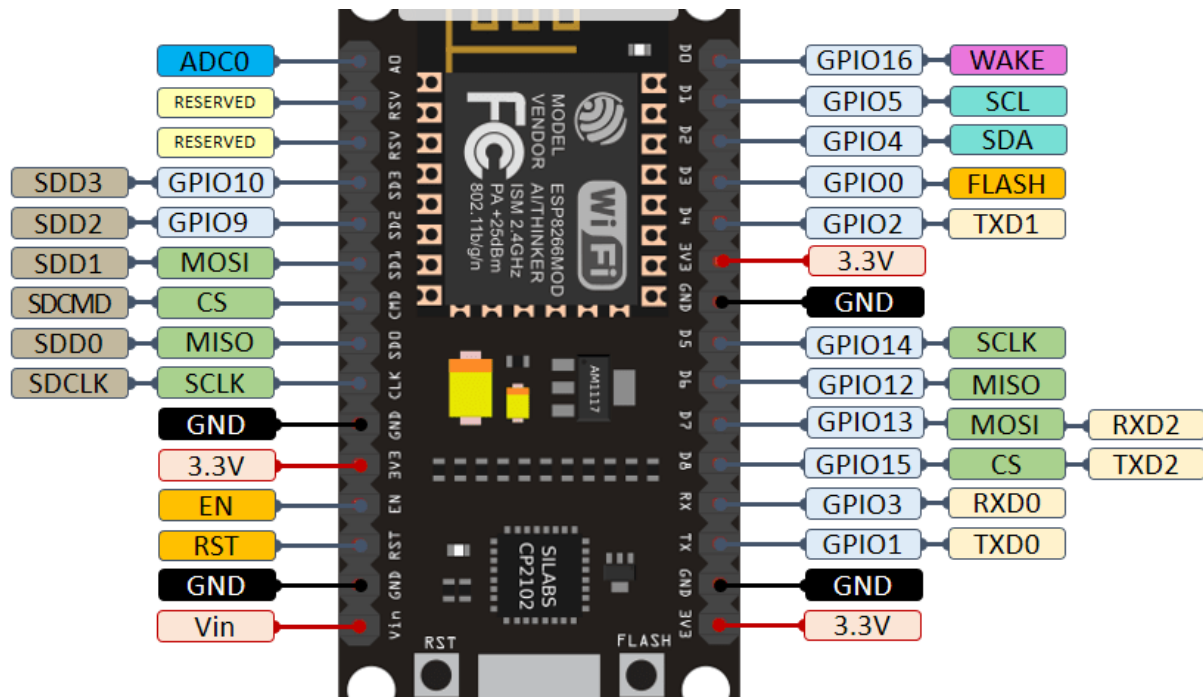
Show timestamp

Both NL & CR

9600 baud

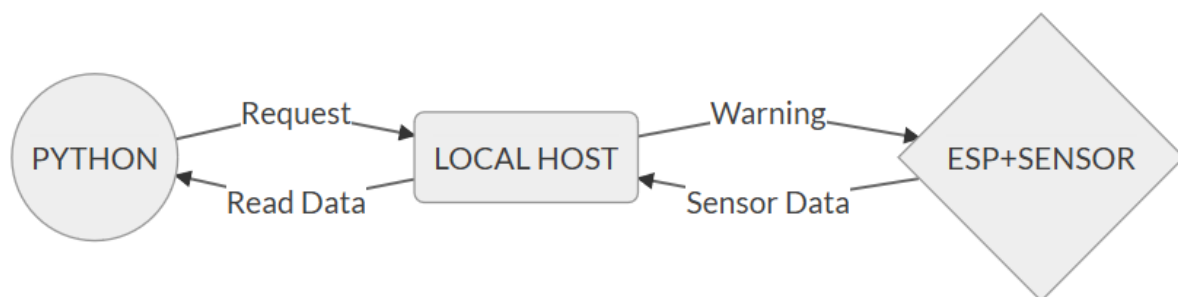
Clear output

Installing ESP8266 Into Arduino IDE And, Python Communication



ESP8266 12-E NodeMCU Pinout Diagram

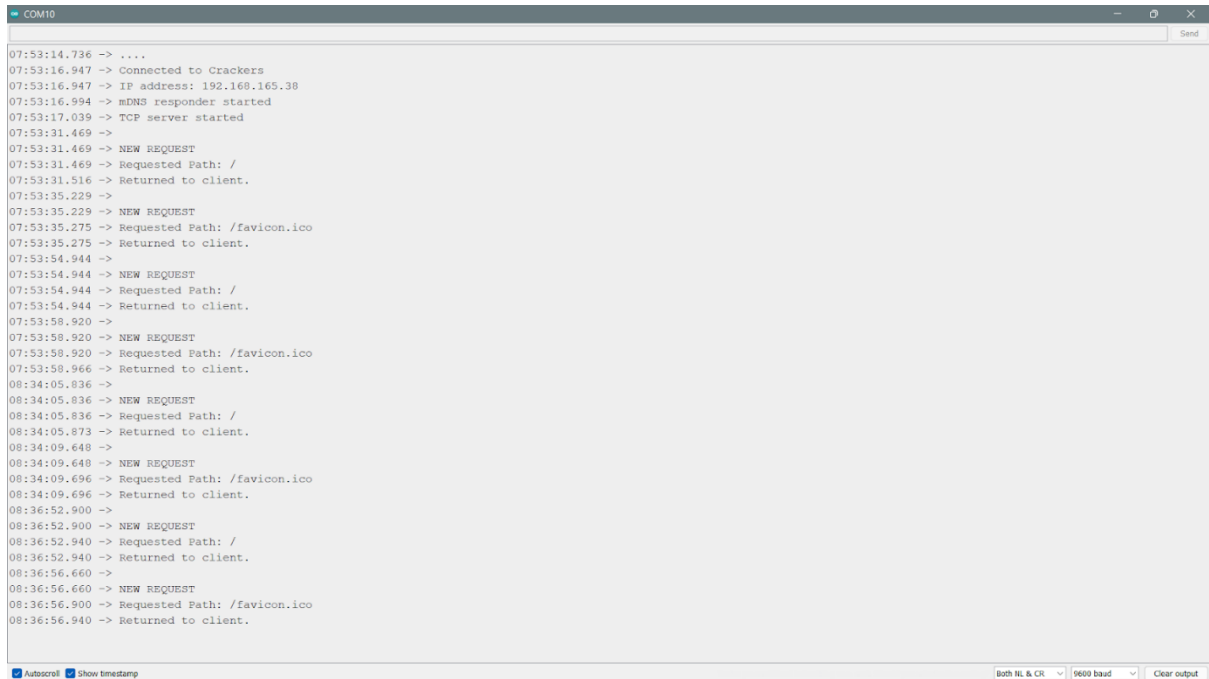
Created a simple mDNS communication system. Our esp connects to our wifi and creates a localhost server and starts to wait for a request. Every time our python sends a request to that localhost, esp runs the desired code and then returns the result as an http request. Finally, python reads that returned data as http request and grab those variables from it. With this, esp can return strings, datas and arrays. Python code will understand their datatype.



While code is being uploaded, we can open the serial port, the details are printed when uploading is done. Learn the IP of esp and note that. ESP's IP on local; changes by wifi to wifi, not session to session, so when we close and open it later, it will not be changed.

We can also view the raw data with a browser while pasting the ip of ESP on a browser. Or we can make an application to read it, or can even use another language.

Serial monitor window



```
07:53:14.736 -> ....
07:53:16.947 -> Connected to Crackers
07:53:16.947 -> IP address: 192.168.165.38
07:53:16.994 -> mDNS responder started
07:53:17.039 -> TCP server started
07:53:31.469 ->
07:53:31.469 -> NEW REQUEST
07:53:31.469 -> Requested Path: /
07:53:31.516 -> Returned to client.
07:53:35.229 ->
07:53:35.229 -> NEW REQUEST
07:53:35.275 -> Requested Path: /favicon.ico
07:53:35.275 -> Returned to client.
07:53:54.944 ->
07:53:54.944 -> NEW REQUEST
07:53:54.944 -> Requested Path: /
07:53:54.944 -> Returned to client.
07:53:58.920 ->
07:53:58.920 -> NEW REQUEST
07:53:58.920 -> Requested Path: /favicon.ico
07:53:58.966 -> Returned to client.
08:34:05.836 ->
08:34:05.836 -> NEW REQUEST
08:34:05.836 -> Requested Path: /
08:34:05.873 -> Returned to client.
08:34:09.648 ->
08:34:09.648 -> NEW REQUEST
08:34:09.696 -> Requested Path: /favicon.ico
08:34:09.696 -> Returned to client.
08:36:52.900 ->
08:36:52.900 -> NEW REQUEST
08:36:52.940 -> Requested Path: /
08:36:52.940 -> Returned to client.
08:36:56.660 ->
08:36:56.660 -> NEW REQUEST
08:36:56.900 -> Requested Path: /favicon.ico
08:36:56.940 -> Returned to client.
```

Browser window



Measure of SpO₂, Heart Rate and BP Trends (BPT)

High physiological stress and high altitudes can be a reason for varying levels of oxygen. The human body is generally capable of adapting itself to such extreme conditions but hypoxemia is always a possibility.

This is a unique and easy way to measure PPG and calculate spO₂, heart rate and Blood Pressure trending (BPT) with surprisingly high accuracy.

There are several Pulse boards available with different form factors and applications. ProtoCentral also provides the popular MAX30102-based breakout board in a variety of versions, but Pulse Express stands out by integrating the MAX32664D Biometric sensor hub.

Typically, PPG is the output from the optical sensors and it is up to the user to calculate other vitals such as Heart rate, spO₂ etc, but the MAX32664D sensor hub does all the vitals calculations and gives us the final output.

- Built in the shape and size of a finger, it is ideally suited to measuring vitals with a Velcro strap hole to connect the finger on board.
- Internal algorithm for measuring Pulse Heart Rate, Pulse Blood Oxygen Saturation (SpO₂), Estimated Blood Pressure trending.
- Integrates a high-sensitivity pulse oximeter and heart rate sensor (MAX30102) and biometric sensor hub (MAX32664D).
- In-built accelerometer for robust detection and compensation of motion artifacts.

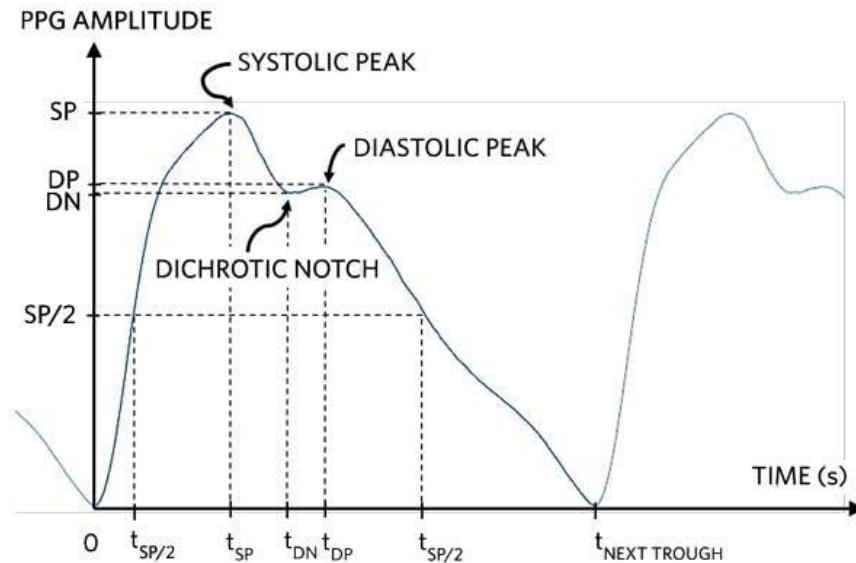
Pulse oximetry (percentage of SpO₂ concentration in blood) has been used as a key health indicator for many decades. Although the original academic development was made in 1935, the modern basis for determining the SpO₂ concentration using light sources and photosensor(s) was developed by Takuo Aoyagi and Michio Kishi in 1972.

When commercially feasible, SpO₂ concentration measurement devices have made huge gains in medical applications. Since 1987, the Standard of Care (SoC) for the administration of a general anaesthetic has included pulse oximetry. All modern hospital bedside equipment include an SpO₂ module based on the same fundamentals, albeit with minor modifications.

Pulse oximetry is used to measure the level of oxygen (oxygen saturation) in the blood. It's a simple, painless measure of how well oxygen is being sent out from our heart to parts of our body, such as our arms and legs. It can be used to check if there is enough oxygen in the blood and to check the health of a person with any condition that affects blood oxygen levels.

Blood Pressure Trend is NOT the same as absolute blood pressure measurement. BPT uses an algorithm to look at changes in the shape of the PPG signal and correlate them to changes in BP from a given calibrated baseline BP. This is still useful because it is not possible to take continuous BP recordings from traditional cuff-based sphygmomanometer devices.

Changes in BP or the BP trend (BPT) prove to be valuable in cardiovascular care and monitoring of high-risk patients.



Protocentral Pulse Express comes with all the basic things for us to get started with reading PPG signal and get basic measurements. It consists of a MAX32664-D Biometric Sensor Hub along with a MAX30102 pulse sensor and logic level converters. We simply have to hook it up with Arduino as shown in the following table.

Max32664 pin label	Arduino Nano Connection	Pin Function
SDA	A4	Serial Data
SCL	A5	Serial Clock
Vin	3.3V	Power
GND	Gnd	Gnd
MFIO Pin	D5	MFIO
RESET Pin	D4	Reset

We have designed the ProtoCentral Pulse Express with integrated high-sensitivity optical sensors (MAX30102) and MAX32664D biometric sensor hub to read PPG signals and perform finger based heart rate, spo2 and blood oxygen saturation measurements.

The board is connected to the MC using a standard I2C interface. Using the ProtoCentral Pulse Express Arduino Library uploaded to the NANO interfaced with the Pulse Express Pulse Oximeter and the Heart Rate Monitor makes it much easier to read the PPG and the measured SPO2, the heart rate and the estimated BP. The output is seen on the serial monitor and the serial plotter.

This code configures the sensor in algorithm mode to enable the sensor to start calculating the Spo2, BP systolic and diastolic and HR values.

The board will start in calibration mode once we upload the library, we will have to keep our finger on the sensor until the calibration progress reaches 100%. The board will switch to estimation mode once the calibration process is completed. It takes 10 to 20 seconds for the algorithm to gather enough samples to give good values.

Circuit Images

