Spo2 & BPM detection

Objective:

The primary aim of this project is to create a heart rate and SpO2 tracking system, leveraging the ESP8266 microcontroller, MAX30100 pulse oximeter sensor, and the Thing Speak IoT cloud platform. This device will be responsible for collecting and transmitting user heart rate and SpO2 data through Wi-Fi to Thing Speak. Subsequently, users can conveniently access and monitor this data remotely via a web browser or smartphone application

Components:

ESP8266:

The ESP8266 is a versatile and affordable microcontroller with built-in Wi-Fi, ideal for IoT projects. It features GPIO pins for hardware interfacing, onboard flash memory for program storage, and operates at 3.3V. It supports programming in various languages like Arduino and Micro Python and has strong community support with libraries available. Its applications include home automation, sensor networks, and remote monitoring, making it a popular choice for IoT projects due to its cost-effectiveness and small form factor.

MAX30100:

The MAX30100 is a compact optical sensor module by Maxim Integrated, designed for measuring heart rate and blood oxygen saturation (SpO2). It combines red and infrared LEDs with a photodetector and signal processing in one package. With low power consumption, it's ideal for wearable fitness trackers, pulse oximeters, and health monitoring. It communicates via I2C, making it easy to integrate with microcontrollers for precise and reliable vital sign measurements.

0.96-inch 7-pin OLED:

The 0.96-inch 7-pin OLED is a compact and efficient display with high contrast and brightness. It's commonly used in small electronic devices and projects, offering vibrant colours and low power consumption. With a simplified 7-pin interface, it's suitable for wearables, IoT applications, and handheld devices, displaying text, graphics, and custom content.

Jumper wires:

Jumper wires are short, insulated conductive cables used to connect electronic components, like sensors and microcontrollers. They come in various types, lengths, and colours and are essential for prototyping and connecting components on boards.

Breadboard:

A breadboard is a versatile, solderless prototyping tool used for building and testing electronic circuits. It features interconnected metal clips in a grid pattern, allowing components to be inserted and connected quickly. This facilitates easy experimentation and design changes, making it ideal for beginners and professionals alike.

Thing Speak:

Thing Speak is a cloud-based IoT platform by MathWorks for collecting, analysing, and visualizing data from connected devices. It supports data integration, analytics, visualization, and alerts, making it versatile for IoT applications.

Implemented Attributes:

Heart Rate & SpO2 monitoring:

Sensor data is collected every 15 seconds, and the values are retrieved using the functions `pox. getHeartRate() and `pox.getSpO2()`. This 15-second interval aligns with the minimum data update frequency required by Thing Speak's cloud platform.

Internet Connectivity:

Establishing an internet connection is vital for transmitting data from the ESP8266 to the cloud and accessing real-time data on the local web server's host. This internet connectivity is enabled through the inclusion of the library `#include <ESP8266WebServer.h>`. Additionally, we provide the SSID and password for the Wi-Fi network to which the ESP8266 should connect.

Iot Cloud Integration:

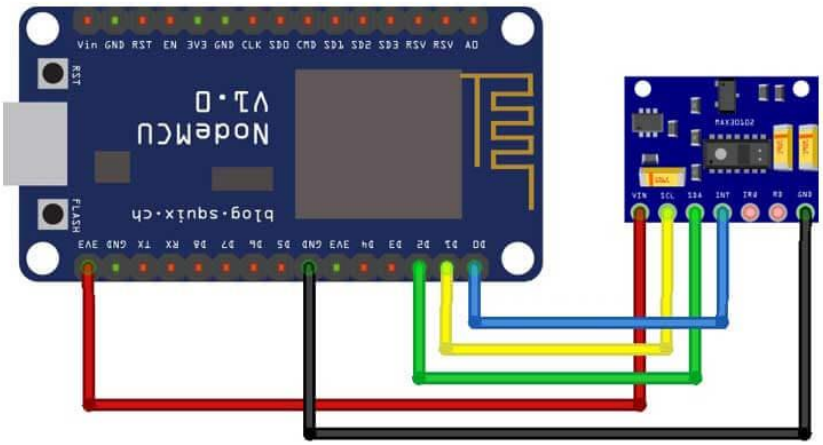
The IoT cloud serves as a repository for storing previously analysed data, which will later be utilized for data visualization. To establish a connection with this cloud platform, we include the library `#include <ThingSpeak.h>`. In addition to this library, we must provide the Channel ID and API Key for the specific channel and profile where the data should be recorded. Before executing the code, it's essential to create the necessary channel on the Thing Speak website, ensuring it meets the required specifications. To input data into the fields, we use the function ThingSpeak.setField(1, SpO2). Subsequently, the function `ThingSpeak.writeFields(channelNumber, Api Key) is employed to write these values to the designated channel.

Display:

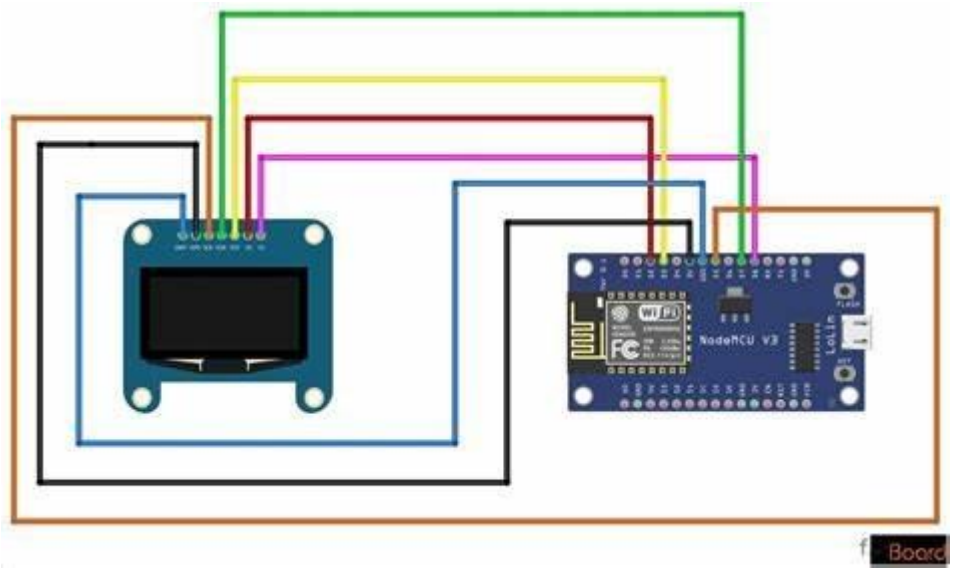
The sensed data is being shown on a 0.96-inch OLED display by utilizing the Adafruit_GFX.h and Adafruit_SSD1306.h libraries. During initialization, specific GPIO pins of the ESP8266 are connected to corresponding pins on the OLED display as follows: OLED_MOSI to 13, OLED_CLK to 14, OLED_DC to 2, OLED_CS to 15, and OLED_RESET to 0. To display the data, the display.println() function is employed, and when it's time to clear the displayed data, the display.clearData() function is called. The data is transferred from the ESP8266 to the OLED display using the Serial Peripheral Interface (SPI) mechanism.

Configuration Diagrams:

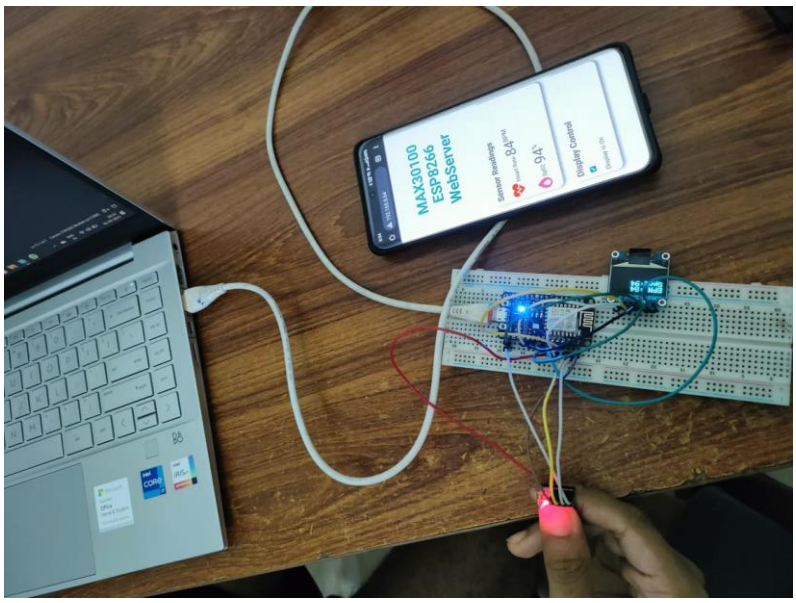
Connection of ESP8266 and MAX30100:



Connection of ESP8366 and OLED:



Circuit Diagram :



Code URl: https:/github.com/Kota04/IoT_project

Code Explanation:

- Void setup() -> The setup includes defining the necessary baud rate for data transmission, setting up pin modes, establishing a connection to Wi-Fi with the ESP8266, and initializing a web server for data viewing. Additionally, it includes initializing a pulse oximeter and commencing the display module.

- The `loop()` function is a control loop that continuously runs and performs several tasks: handling web server requests, reading pulse oximeter data, updating and displaying data on an OLED display, reporting data to ThingSpeak, and triggering email alerts based on health-related conditions. It follows a periodic reporting mechanism based on the elapsed time since the last report.

- Void loop() ->

  `server.handleClient()`: This line handles incoming HTTP client requests. It suggests that the microcontroller is hosting a web server, and this function is used to manage client interactions.

  `pox.update()`: This corresponds to updating or reading data from a pulse oximeter sensor. It's common for sensors to require periodic updates to obtain fresh data.

  ```
  if (displayOn) {
    displaytext(BPM, SpO2);
  }
  ```
  This conditional statement checks if a variable `displayOn` is set to `true`. If it is, it calls a function `displaytext(BPM, SpO2)` to display information on the OLED display.

  ```
  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
    // ...
  }
  ```
  This conditional statement checks if a certain period of time, defined by `REPORTING_PERIOD_MS`, has passed since the last data report. If the condition is met, it proceeds to update and report sensor data.

  Data Acquisition and Reporting:

  - `BPM = pox.getHeartRate() ->` fetch the heart rate (BPM)
  - `SpO2 = pox.getSpO2() ->` blood oxygen saturation (SpO2) data from the pulse oximeter sensor.
  - `display.clearDisplay() ->` clears the display to prepare it for new data.
  - `Serial.print()` and `Serial.println()` statements are used to print the sensor data and messages to the serial monitor for debugging and monitoring purposes.

- `ThingSpeak.setField(1, SpO2) ;` and `ThingSpeak.setField(2, BPM);` set the data fields for uploading to ThingSpeak, which is an IoT data platform.

- The code sends the data to ThingSpeak with an HTTP POST request and checks the response code for success or failure.

Alerting:

- A conditional statement checks if the SpO2 is below 80 or BPM is above 120. If this condition is met, it constructs an alert message and calls a function **`sendEmailAlert()`** to send an email alert with the message. This is likely for health monitoring purposes.

`tsLastReport = millis()` -> This line updates the timestamp `tsLastReport` to the current time using the `millis()` function. It's used to track the last time data was reported.

- The `displaytext` function clears the OLED display, sets text size and color, prints BPM and Spo2 values along with labels, displays the updated content on the OLED screen, and introduces a small delay before returning.

display.clearDisplay() -> This line clears the contents of the OLED display. It prepares the screen for the new information that will be displayed

display.setTextSize(2) -> It sets the text size to 2X-scale, making the text larger than the default size. This is done to increase readability.

display.setTextColor(WHITE) -> This line sets the text color to white. On an OLED display, specifying the text color is important because OLEDs emit light themselves, and setting the text to white makes it visible against a dark background

display.setCursor(0, 0) -> This sets the cursor position at the top-left corner of the display (coordinates 0, 0), indicating where the text will begin.

display.print(F("BPM :")) -> This line prints the text "BPM :" on the display. The F() macro is often used with Arduino and similar platforms to store text strings in flash memory (instead of RAM) to save space.

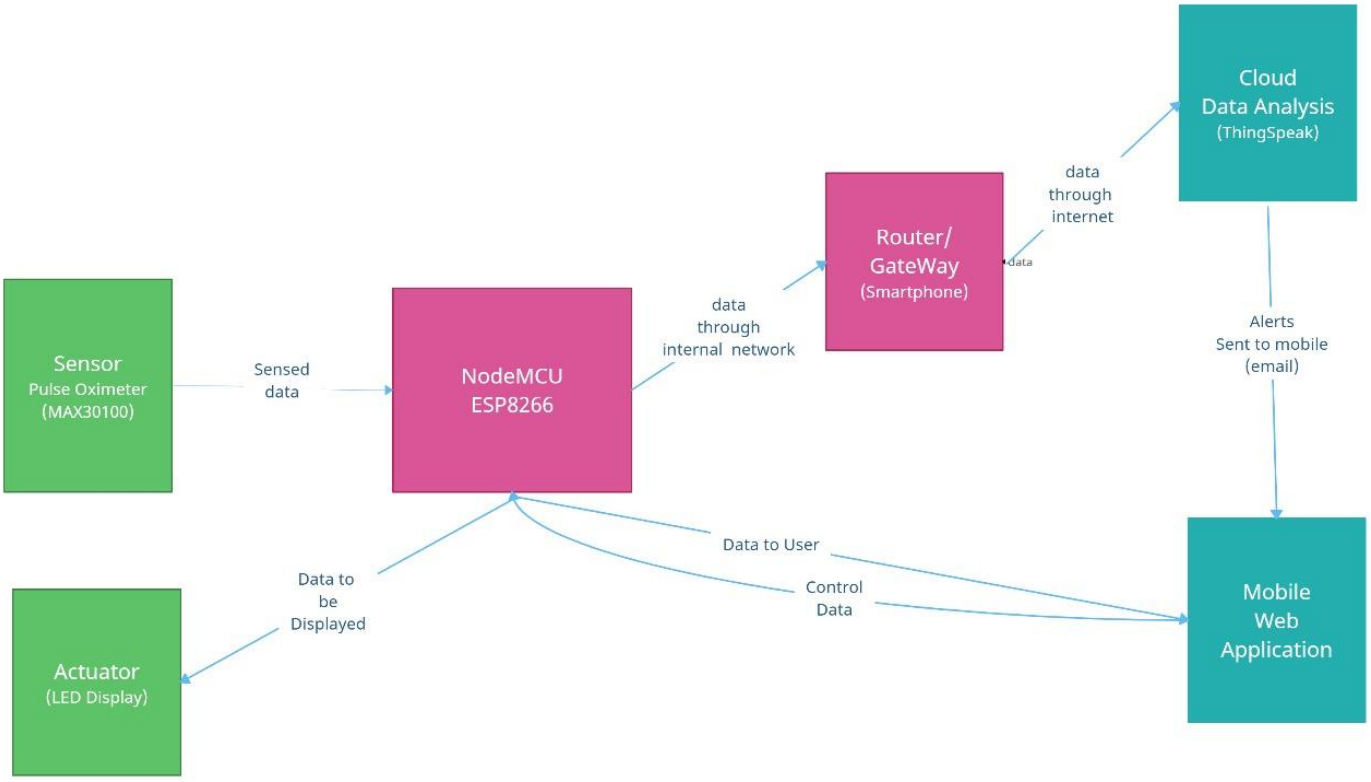
- void handle_OnConnect() -> Responds with an HTML document containing BPM and SpO2 data when a client connects to the web server (HTTP status 200).

- void handle_NotFound() -> Sends an HTTP response with a status code of 404 ("Not found") when a requested resource is not found on the server.

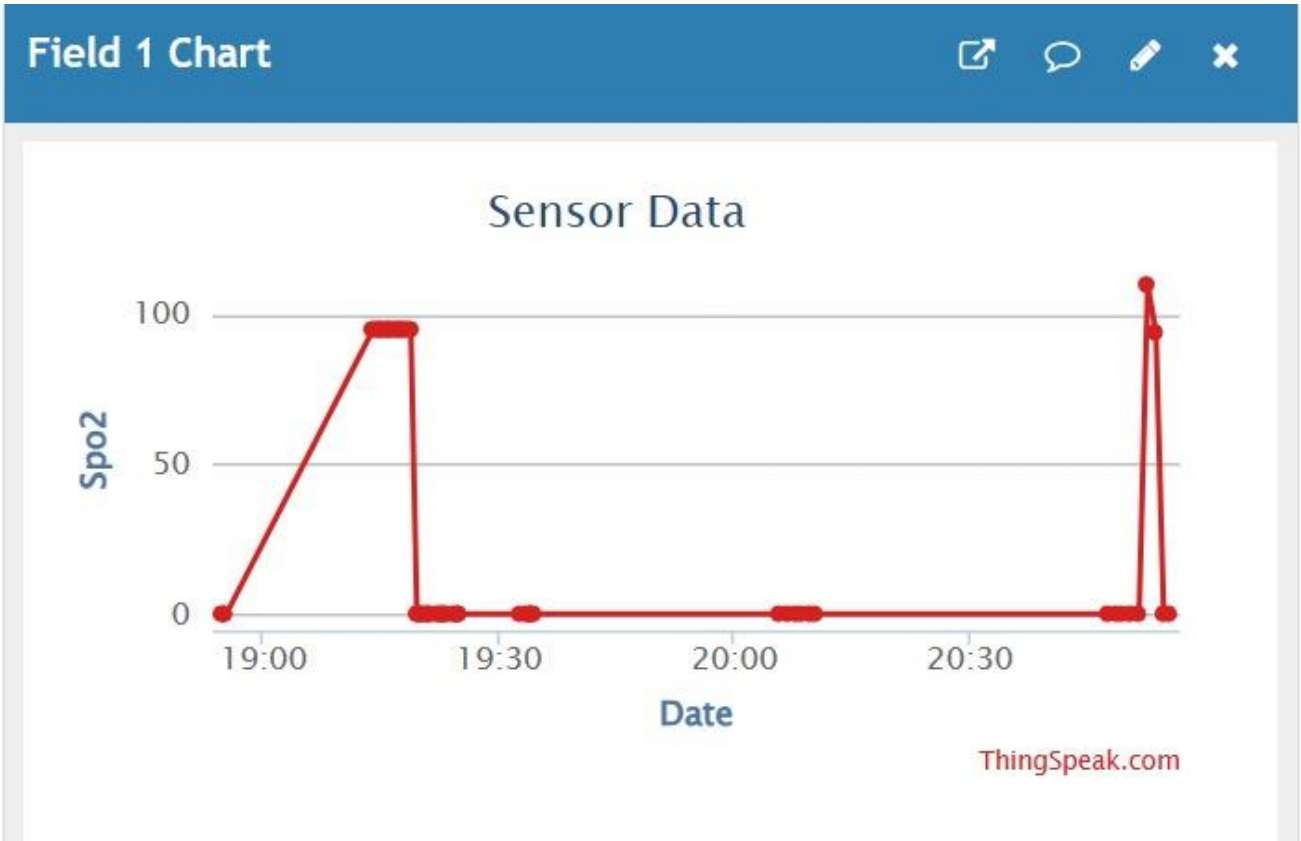
- String SendHTML(float BPM, float SpO2) -> These functions are employed to build a user interface (UI) for real-time data monitoring on smart devices connected to the same Wi-Fi network as the ESP8266. The web page, created using AJAX, updates data without requiring a page refresh. It also includes a toggle button for

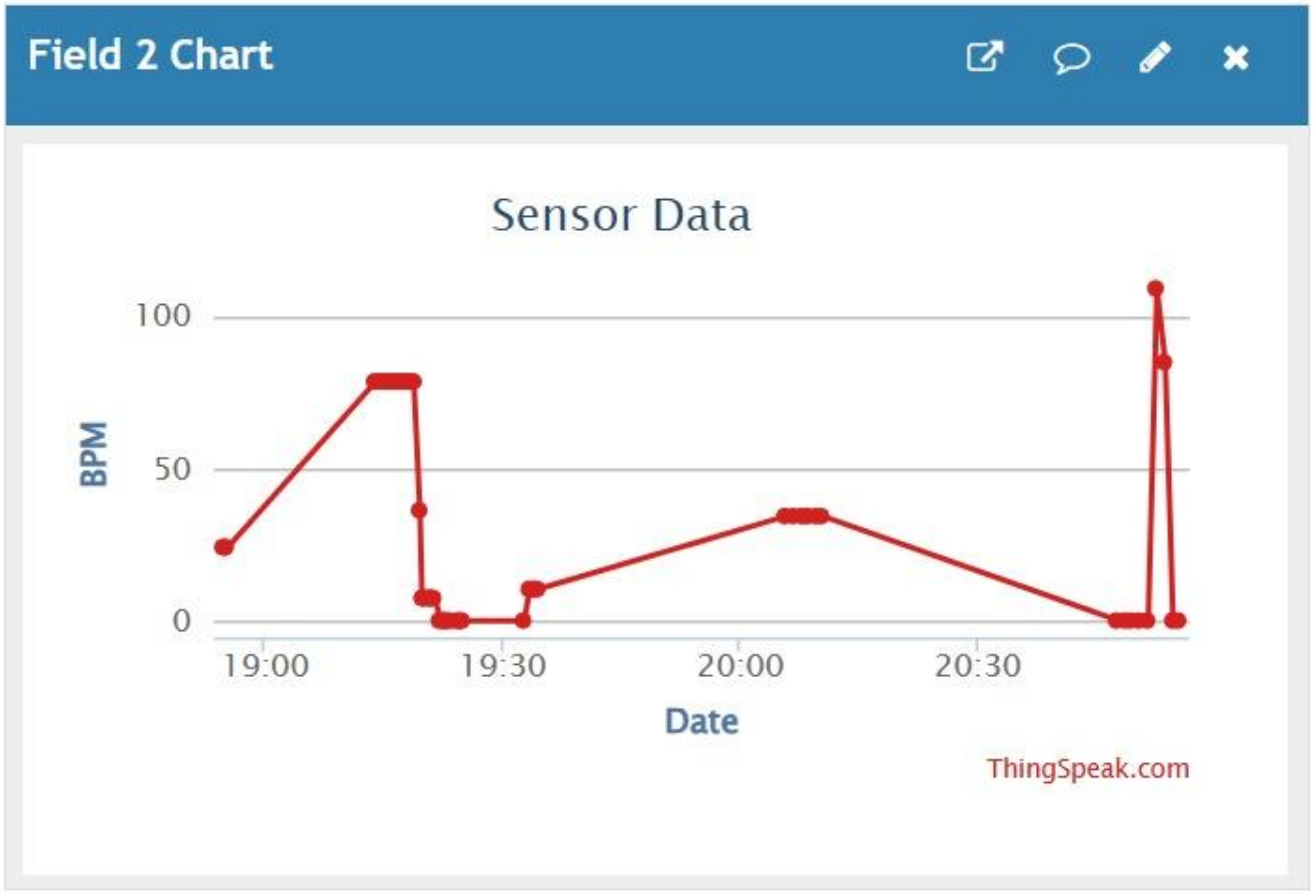
controlling an LED display. To access this web page, users need to enter the IP address displayed in the serial monitor into their web browser.

Data Flow Diagram:



Output Data:





Demo Video Link: https://photos.app.goo.gl/1KTKhD7hgdVx3zLS9

Done By:

Janapati Sai Ajay (210101052)

Kota Satya Kiran (210101058)

Nitish Kumar Pinneti (210101125)