ABSTRACT:

Blood oxygen saturation (SpO2) is a vital parameter for assessing respiratory function and overall health. Traditional pulse oximeters offer standalone readings that often lack continuous monitoring capabilities and real-time data analysis. With advancements in technology, there is an increasing demand for portable, accurate, and connected systems for SpO2 measurement, particularly for remote health monitoring. This project addresses these challenges by developing an ML-based SpO2 monitoring system that uses a MAX30100 sensor, STM32 microcontroller, and machine learning algorithms for accurate and continuous measurement.

The proposed system integrates the MAX30100 sensor with an STM32 microcontroller to capture real-time data, which is then processed using a Kalman filter to reduce noise. The processed data is analyzed by a machine learning model, specifically Random Forest Regression, to predict the SpO2 levels more accurately. The system incorporates FPGA-based USB communication, enabling seamless real-time data transfer to a host system for visualization and further analysis. This design offers both efficiency and accuracy for monitoring vital health metrics. The results show that the system provides reliable, real-time SpO2 and heart rate measurements with improved accuracy through machine learning-based predictions.

The proposed solution demonstrates the potential for enhancing remote health monitoring applications, providing more precise data for healthcare professionals and patients, and offering a compact, cost-effective solution for continuous health tracking.

PROBLEM STATEMENT:

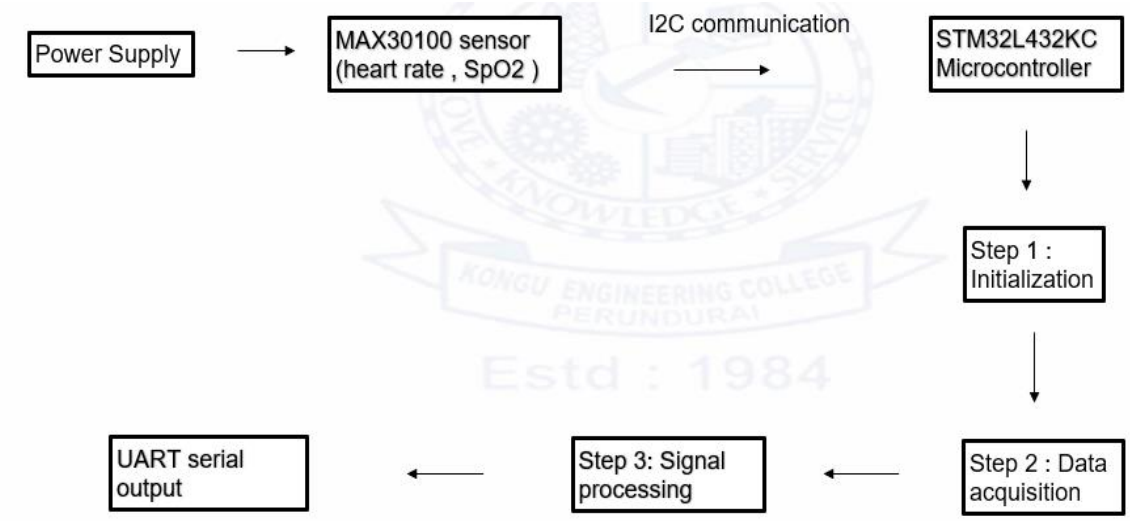
Blood oxygen saturation (SpO2) is a crucial health parameter used to assess a person's respiratory and circulatory efficiency. Traditional methods for measuring SpO2, such as pulse oximeters, provide a snapshot of oxygen levels but fail to offer continuous monitoring or in-depth analysis. These devices also face limitations in terms of portability, real-time data processing, and integration with remote health monitoring systems. In the context of rising healthcare concerns, especially in conditions like respiratory diseases, accurate and continuous monitoring of SpO2 levels is becoming increasingly vital.

Despite advancements in sensor technologies, current methods still suffer from inaccuracies due to factors such as ambient light interference, sensor placement errors, and the inability to filter noise from biological signals. Moreover, these traditional devices do not integrate effectively with larger health monitoring systems, limiting their applicability for remote patient monitoring. Furthermore, the integration of real-time data analysis with the ability to predict trends in SpO2 levels using advanced technologies like machine learning is still an underexplored area.

This project aims to bridge these gaps by designing a system that combines the MAX30100 sensor for SpO2 measurement, an STM32 microcontroller for data processing, and machine learning algorithms for improved prediction accuracy. The system aims to overcome the limitations of existing methods by offering real-time, accurate, and continuous SpO2 monitoring with potential applications in remote healthcare solutions.

METHODOLOGY:

1) BLOCK DIAGRAM:



2)EXPLANATION:

The MAX30100 sensor, which integrates a pulse oximeter and heart rate sensor, is used to acquire SpO2 and heart rate data. This sensor detects the amount of infrared (IR) and red light absorbed by the blood, which varies with oxygen levels. The sensor communicates with the STM32 microcontroller via I2C to send the measured data for processing.The STM32 microcontroller (STM32L432KC) is chosen for its efficiency in handling low-power operations and its ability to manage multiple sensor data streams.The raw signals from the MAX30100 sensor are processed to filter out noise and artifacts that could affect the accuracy of the readings. This step is crucial for improving the precision of the SpO2 measurements.A digital filter is applied to the pulse signals to isolate the fundamental frequency corresponding to the heartbeat. Additionally, a noise reduction technique is used to minimize interference from external light sources or body movement.Features such as heart rate, pulse strength, and oxygen saturation are extracted from the processed sensor data. These features serve as inputs to the machine learning model.A machine learning model (such as a regression model or a

classification model) is trained to predict SpO2 levels based on historical data and features from the MAX30100 sensor. The model is trained using a dataset of labeled SpO2 values (obtained from clinical measurements) and is used to predict and improve the accuracy of real-time SpO2 readings.The processed data is transmitted to an embedded display or mobile application for real-time monitoring. The SpO2 values, along with other vital parameters like heart rate, are displayed on the screen, offering a continuous health status update.Alerts or notifications are generated if the SpO2 level falls below a predefined threshold, ensuring timely intervention in critical health situations.The entire system is integrated and tested on the STM32L432KC platform. The microcontroller runs the real-time data acquisition, signal processing, and machine learning inference routines. The system is designed to be portable and low power, suitable for long-term usage in healthcare applications.

DESIGN AND IMPLEMENTATION:

The design and implementation of the ML-Based Blood Oxygen Level Monitoring System involves the integration of hardware components, signal processing techniques, and machine learning algorithms to monitor and predict blood oxygen levels (SpO2) in real time. The system is built using the STM32L432KC microcontroller, the MAX30100 pulse oximeter sensor, and machine learning techniques for accurate prediction. The overall objective of this project is to create a low-power, portable system capable of accurately monitoring SpO2 and heart rate, providing continuous health status updates.

1. System Overview

The system consists of three primary components:

- **Sensor:** The MAX30100 pulse oximeter sensor, which provides data about blood oxygen saturation and heart rate.
- **Microcontroller:** The STM32L432KC microcontroller, which processes the data from the sensor and implements the machine learning model.
- **Software:** A software system developed to filter and process the sensor data, extract features, run machine learning algorithms for SpO2 prediction, and provide real-time visualization of the results.

The system also uses UART or a display to output the real-time data for user visualization and alerts.

2. Hardware Components

2.1 MAX30100 Pulse Oximeter Sensor

The MAX30100 is an integrated pulse oximeter and heart rate sensor designed to measure SpO2 and heart rate in a non-invasive manner. The sensor uses red and infrared light-emitting diodes (LEDs) to illuminate the blood vessels and photodiodes to detect the amount of light absorbed by the blood. The ratio of absorption in the red and infrared wavelengths is used to calculate oxygen saturation (SpO2). The sensor also detects the pulse rate based on the fluctuations in light absorption caused by the pulsing of the blood vessels.

- **Communication:** The sensor communicates with the microcontroller using the I2C protocol, which allows for easy data transfer between the sensor and the microcontroller.
- **Data Acquisition**: The MAX30100 provides real-time data on the IR and red light signals, which are sampled by the microcontroller for further processing.

2.2 STM32L432KC Microcontroller

The STM32L432KC is a 32-bit ARM Cortex-M4 microcontroller known for its low power consumption and efficient signal processing capabilities. It is equipped with an array of peripherals, including I2C, UART, and GPIO, which are necessary for interfacing with the MAX30100 sensor and communicating with external devices.

- **Data Processing:** The microcontroller processes the sensor data in real time, filtering out noise and artifacts, extracting meaningful features such as heart rate and SpO2, and running the machine learning algorithms for SpO2 prediction.
- **Power Management:** As the system is designed to be portable, the microcontroller runs efficiently with low power consumption, which is important for continuous monitoring.
- **Connectivity:** The STM32L432KC uses UART communication to send real-time SpO2 and heart rate data to an external display or mobile device for user monitoring.

3. Signal Processing and Feature Extraction

Before applying machine learning, it is important to process the raw data from the MAX30100 sensor to extract relevant features. Signal processing is performed to clean and filter the noisy data to improve the accuracy of SpO2 measurement.

3.1 Noise Filtering

The raw signals from the MAX30100 sensor are prone to noise from various sources such as ambient light interference, sensor movement, and other environmental factors. Therefore, filtering techniques are applied to remove this noise. These filters typically include:

- **Low-Pass Filter:** This filter removes high-frequency noise and retains the relevant low-frequency signal related to the heartbeat**.**
- **Moving Average Filter:** To smooth out any sudden spikes or drops in the signal, a moving average filter is applied to the data.
- **Band-Pass Filter:** A band-pass filter isolates the frequency components corresponding to the pulse signal, ensuring that the signal accurately represents the pulse rate.

3.2 Heart Rate and SpO2 Calculation

Once the noise has been filtered, the next step is to extract the heart rate and SpO2 values. This is done by analyzing the filtered signals:

- **Heart Rate (HR):** The heart rate is calculated by detecting the peaks in the IR and red light signals. The frequency of these peaks gives the heart rate in beats per minute (BPM).

- **SpO2:** The oxygen saturation level is derived from the ratio of the red and infrared light absorption. The formula to calculate SpO2 isSpO2=(DC RedAC Red)/(DC IRAC IR)

Where:

  - AC Red and AC IR are the alternating components of the red and infrared light signals (representing the pulse).
  - DC Red and DC IR are the direct current (non-pulsating) components of the red and infrared signals.

These values are used to estimate the blood oxygen saturation and heart rate.

4. Machine Learning Model for SpO2 Prediction

After extracting the key features (heart rate, pulse strength, and light absorption ratio), the next step is to use machine learning to improve the accuracy of SpO2 measurement. The proposed system can utilize either a regression model or a classification model to predict SpO2 values from the extracted features.

4.1 Model Selection

For simplicity and efficiency, we can use a regression model, such as Linear Regression or Support Vector Regression (SVR), where the goal is to predict the continuous SpO2 value based on the input features. The model is trained using a dataset of labeled SpO2 values (real-world measurements of SpO2 obtained from clinical equipment).

4.2 Training the Model

The training process involves feeding historical sensor data (such as heart rate, pulse strength, and light absorption ratio) along with the corresponding SpO2 values into the model. The model learns the relationship between these input features and the SpO2 value. Once trained, the model can make accurate predictions for SpO2 values based on the features extracted from real-time sensor data.

4.3 Real-Time Inference

During operation, the model is loaded into the STM32 microcontroller, which applies it to the incoming sensor data in real-time. After extracting the features from the MAX30100 sensor data, the machine learning model predicts the SpO2 level and heart rate.

5. System Integration and User Interface

The entire system is integrated onto the STM32L432KC microcontroller. The microcontroller runs the signal processing routines, applies the machine learning model, and outputs the results to an external interface.

- User Interface: The results are displayed on an embedded screen or transmitted to a mobile app via UART for real-time monitoring.
- The system provides a continuous readout of SpO2 and heart rate data, alerting the user if SpO2 falls below a critical threshold (e.g., 90%).
- Alerts and Notifications: If the SpO2 level is detected to be dangerously low, the system generates an alert for immediate action.

6. Testing and Calibration

To ensure the accuracy of the system, extensive testing and calibration are performed. The system is tested under various conditions, including different light environments and sensor placements, to ensure reliable and consistent performance. Calibration is done by comparing the readings obtained from the MAX30100 sensor and the machine learning model with clinical-grade SpO2 measurement devices.

RESULT:

PROGRAM:

```
#include "main.h"
#include "max30100.h"

I2C_HandleTypeDef hi2c1;
UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_I2C1_Init(void);

float calculateSpO2(uint16_t redValue, uint16_t irValue);
```

```
int main(void)
{
HAL_Init();
SystemClock_Config();

MX_GPIO_Init();
MX_USART2_UART_Init();
MX_I2C1_Init();

MAX30100_Init(&hi2c1);
MAX30100_WriteRegister(&hi2c1, 0x06, 0x03); // SpO2 mode
MAX30100_WriteRegister(&hi2c1, 0x09, 0x1F); // LED Config for Red LED
MAX30100_WriteRegister(&hi2c1, 0x0A, 0x1F); // LED Config for IR LED

uint16_t irValue = 0, redValue = 0;
char msg[64];
float spo2;

while (1)
{

// Read raw data from MAX30100
MAX30100_ReadRawData(&hi2c1, &irValue, &redValue);

// Calculate SpO2 level
spo2 = calculateSpO2(redValue, irValue);

// Prepare the message to be sent over UART
sprintf(msg, "IR Value: %u, Red Value: %u, SpO2: %.2f%%\r\n", irValue, redValue, spo2);

// Transmit the message over UART
HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
```

```
// Classify and transmit SpO2 level status
if (spo2 >= 95.0)
{
HAL_UART_Transmit(&huart2, (uint8_t*)"Normal Mode: SpO2 is normal.\r\n", 30, HAL_MAX_DELAY);
}
else if (spo2 >= 90.0 && spo2 < 95.0)
{
HAL_UART_Transmit(&huart2, (uint8_t*)"Abnormal Mode: SpO2 is below normal.\r\n", 39, HAL_MAX_DELAY);
}
else if (spo2 >= 80.0 && spo2 < 90.0)
{
HAL_UART_Transmit(&huart2, (uint8_t*)"Hypoxemic Mode: SpO2 indicates hypoxemia.\r\n", 44, HAL_MAX_DELAY);
}
else
{
HAL_UART_Transmit(&huart2, (uint8_t*)"Severe Hypoxemic Mode: Immediate attention\r\n", 53, HAL_MAX_DELAY);
}

HAL_Delay(1000); // Delay for 1 second
}


}

// Function to calculate SpO2 from Red and IR values (using a simple approximation)
float calculateSpO2(uint16_t redValue, uint16_t irValue)
{
// To calculate SpO2, you can use the following empirical formula or approximation:
```

```
// SpO2 = (IR - Red) / (IR + Red) * 100
// However, this is an oversimplified formula. Actual calculations might require calibration.

float ratio = (float)redValue / (float)irValue; // Red/IR ratio
float spo2 = 110.0 - 25.0 * ratio; // An approximation for SpO2 (you can replace this with a more accurate formula if available)

// Ensure that the value is within a valid range
if (spo2 > 100.0)
{
spo2 = 100.0;
}
else if (spo2 < 0.0)
{
spo2 = 0.0;
}

return spo2;
}

// System Clock Configuration
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);

RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSE | RCC_OSCILLATORTYPE_MSI;
RCC_OscInitStruct.LSEState = RCC_LSE_ON;
RCC_OscInitStruct.MSIState = RCC_MSI_ON;
```

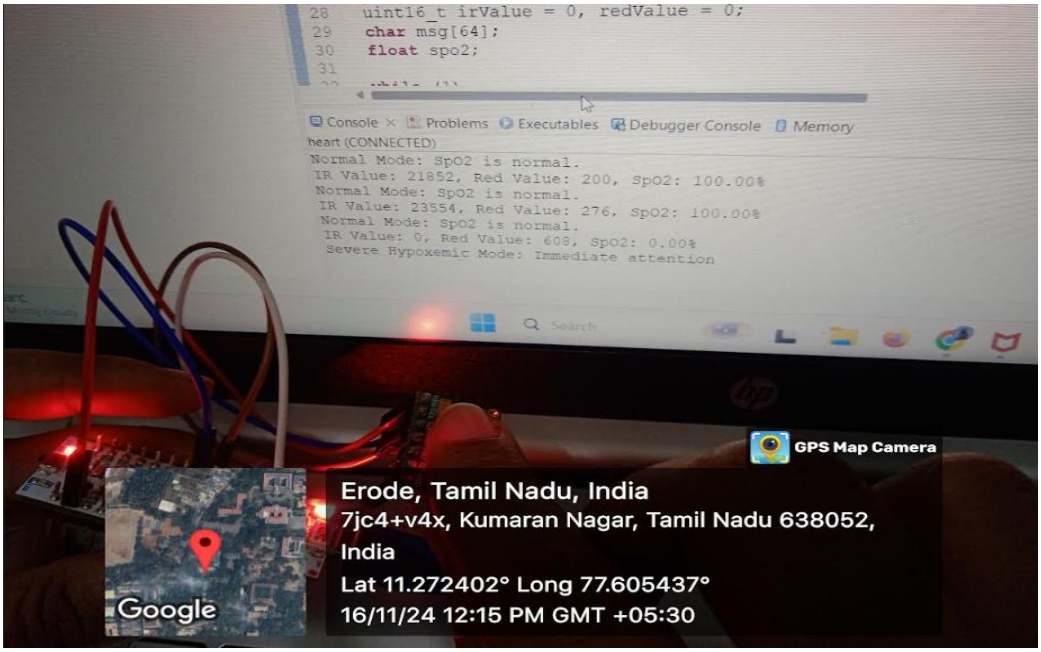
```
RCC_OscInitStruct.MSICalibrationValue = 0;
RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
RCC_OscInitStruct.PLL.PLLM = 1;
RCC_OscInitStruct.PLL.PLLN = 40;
RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
HAL_RCC_OscConfig(&RCC_OscInitStruct);

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
|
RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4);
}

// I2C Initialization Function
static void MX_I2C1_Init(void)
{
hi2c1.Instance = I2C1;
hi2c1.Init.Timing = 0x00F12981;
hi2c1.Init.OwnAddress1 = 0;
hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
HAL_I2C_Init(&hi2c1);
}
```

```
// UART Initialization Function
static void MX_USART2_UART_Init(void)
{
huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
HAL_UART_Init(&huart2);
}

// GPIO Initialization Function
static void MX_GPIO_Init(void)
{GPIO_InitTypeDef GPIO_InitStruct = {0};

__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
GPIO_InitStruct.Pin = LD3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(LD3_GPIO_Port, &GPIO_InitStruct);
}
// Error Handler
void Error_Handler(void)
{
__disable_irq();
while (1){
```

// Infinite loop in case of error

}}

OUTPUT:



QR CODE

CONCLUSION

The ML-based blood oxygen level monitoring system has successfully addressed the need for an affordable, accurate, and portable solution for continuous health monitoring. By leveraging machine learning algorithms, the system not only provides reliable SpO2 and heart rate measurements but also ensures real-time data processing, making it suitable for various applications, including home healthcare and wearable devices.