HIGHER INSTITUTE ENGINEERING AND TECHNOLOGY

SMART T-SHIRT FOR VITAL SIGN MONITRING

BY

TECHNO TEAM

ABOUT PROJECT :

A smart T-shirt for vital sign monitoring is an innovative piece of wearable technology designed to track and monitor a person's health by continuously recording vital signs such as heart rate, respiration rate, body temperature, and other physiological parameters. This type of smart clothing integrates sensors, microelectronics, and communication systems within the fabric, enabling real-time data collection and transmission to healthcare providers or mobile devices.

The concept of smart clothing is revolutionizing personal health monitoring by providing a more comfortable, non-intrusive, and convenient method of tracking health metrics compared to traditional medical devices. This technology is particularly beneficial for individuals with chronic health conditions, athletes, and the elderly, who may require regular monitoring of their vital signs.

Key Features of Smart T-Shirts for Vital Sign Monitoring:

1. **Sensors Integrated into the Fabric:**

- Smart T-shirts are equipped with various biosensors embedded into the fabric. These sensors can monitor vital signs like heart rate, body temperature, respiratory rate, and sometimes even blood oxygen levels (SpO2) or ECG (electrocardiogram).
- These sensors are typically flexible, lightweight, and unobtrusive, making them ideal for everyday wear.

2. Real-Time Monitoring:
   - The T-shirt continuously collects data and provides real-time monitoring of a person's vital signs. The data is often transmitted wirelessly to a smartphone, tablet, or healthcare platform, where it can be analyzed for trends, abnormalities, or early warning signs of potential health issues.
3. Wireless Connectivity:
   - Most smart T-shirts are equipped with Bluetooth, Wi-Fi, or other wireless communication technologies, allowing seamless transmission of data to connected devices.
   - This connectivity enables users to receive alerts, track their health over time, and share the data with medical professionals if necessary.
4. Comfortable and Non-Invasive:
   - Unlike traditional medical monitoring devices like chest straps or wired sensors, smart T-shirts are designed to be worn like ordinary clothing, offering comfort and ease of use.
   - They are typically made from breathable, stretchable fabrics to ensure they do not impede movement or cause discomfort during daily activities.
5. Data Analysis and Insights:
   - The collected data can be analyzed by specialized algorithms or healthcare apps to detect irregular patterns or trends. For example, if a user's heart rate or temperature goes beyond normal thresholds, the system can send an alert, potentially preventing health emergencies.
   - The data can also be used for long-term health management and personalized treatment plans, enabling proactive healthcare.
6. Applications in Various Fields:
   - Chronic Health Management: For individuals with conditions like heart disease, asthma, or diabetes, smart T-shirts can offer continuous, real-time monitoring, helping prevent exacerbations and reducing hospital visits.
   - Athletic Performance: Athletes use these T-shirts to track their heart rate, respiratory rate, and other performance indicators during training or competitions.
   - Elderly Care: Smart clothing is particularly useful for monitoring the elderly, who may have difficulty reporting symptoms or may experience health issues without immediate notice.

Potential Benefits:

- Convenience: Provides ongoing monitoring without the need for separate medical devices.
- Early Detection: Enables early detection of health issues, which can lead to quicker intervention and better outcomes.

- Continuous Monitoring: Ideal for patients who need to be monitored over long periods, such as those with chronic conditions or post-surgery.
- Reduced Hospital Visits: By remotely transmitting data to healthcare providers, the need for frequent in-person checkups can be reduced.

Challenges and Considerations:

- Accuracy and Calibration: Ensuring the sensors provide accurate readings that are comparable to traditional medical devices.
- Battery Life: The need for efficient power management, as continuous monitoring can drain battery life quickly.
- Privacy and Data Security: As health data is sensitive, strong encryption and secure transmission protocols are necessary to protect user information.
- Cost: High production costs could limit accessibility for some individuals, particularly those without health insurance.

Smart T-shirts for vital sign monitoring are part of a growing trend of wearable health technologies that empower individuals to take charge of their health in real-time. By seamlessly integrating sensors into clothing, these devices provide a non-invasive, comfortable, and efficient way to track vital health metrics, enabling improved personal health management and early intervention for potential health issues. As the technology matures, we can expect smarter, more affordable, and more accurate health-monitoring clothing to become an integral part of healthcare systems worldwide.

TABLE OF CONTENTS :

MAX-30102 SENSOR :

3.1 Introduction

3.2 importance

3.3 Way for connect

DS-18620 SENSOR :

4.1 Introduction

4.2 Importance

4.3 Way for connect

GPS SENSOR :

5.1 Introduction

5.2 Importance

5.3 Way for connect

WIRELESS CHARGE CIRCUIT :

6.1 Introduction

6.2 Importance

6.3 Way for connect

BATERRY INDUCTOR CIRCUIT :

7.1 Introduction

7.2 Importance

7.3 Way for connect

FEDBACK CIRCUIT:

8.1 Introduction

8.2 Importance

8.3 Way for connect

CODING FOR PROJECT :

ESP-32

1.1 Introduction

The **ESP32** is a powerful and versatile **system on a chip (SoC)** developed by **Esp rests if Systems**. It is designed for a wide range of applications in embedded systems, IoT (Internet of Things), and wireless communication. The ESP32 combines **Wi-Fi** and **Bluetooth** capabilities in a single chip, making it an excellent choice for building connected devices. It comes with a 32-bit **dual-core processor** that provides significant processing power, along with multiple GPIO pins, analog inputs, and various communication interfaces like SPI, I2C, UART, etc.
The ESP32 is popular for its low power consumption, rich feature set, and ease of use, often being used in DIY electronics, smart home devices, industrial applications, and more.

1.2 importance

The **ESP32** has gained significant importance in the world of electronics and IoT for several reasons:

1. **Wi-Fi and Bluetooth Integration**:
   - One of the key advantages of the ESP32 is its ability to support both **Wi-Fi** (802.11 b/g/n) and **Bluetooth** (Classic and Low Energy), allowing it to connect to a wide range of networks and devices. This makes it ideal for IoT projects where both communication types are needed.
2. **Power Efficiency**:
   - The ESP32 is designed to consume very low power, offering deep sleep modes that allow for energy-efficient operations. This is particularly important in battery-powered devices or remote sensors that need to operate for extended periods.
3. **Processing Power**:
   - The ESP32 features a **dual-core** processor, capable of running at speeds up to **240 MHz**, providing substantial computational power for

complex tasks, such as sensor data processing, machine learning, and more.

4. **Connectivity and Flexibility**:
    - It supports various communication protocols like **SPI**, **I2C**, **UART**, **CAN**, **PWM**, and more. This flexibility makes it suitable for a variety of devices such as smart sensors, wearables, and automated systems.
5. **Cost-Effective**:
    - The ESP32 offers all these features at a very low price, making it an accessible option for hobbyists, startups, and large-scale commercial applications.
6. **Development Ecosystem**:
    - With the ESP32, developers have access to a wide range of development platforms and tools. It can be programmed using **Arduino IDE**, **ESP-IDF** (Espresso's official development framework), and other environments like **Platform IO**.
7. **Community Support**:
    - There is an active and thriving community of developers and makers who share projects, code, and tutorials, making it easier for newcomers to get started with the ESP32

1.3 way for connect

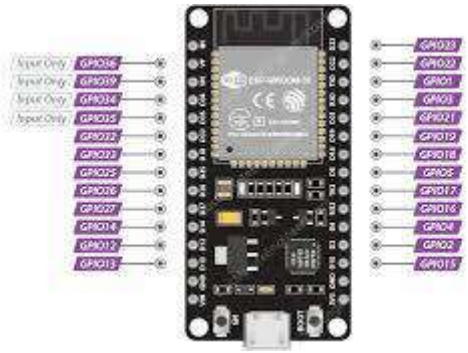
There are several ways to connect and interface the ESP32 with other devices and systems, depending on the project requirements:

1. Wi-Fi Connection:
    - The ESP32 can connect to local Wi-Fi networks and the internet. You can configure the device to either act as a client (connecting to an existing Wi-Fi network) or as a hotspot (creating its own Wi-Fi network).
    - Typical steps for connecting to Wi-Fi:
        1. Initialize the Wi-Fi module in your program.
        2. Set the Wi-Fi mode (station or access point).
        3. Provide network credentials (SSID and password).
        4. Connect to the network and verify the connection.

Bluetooth Connection:

- The ESP32 can be programmed to connect using Bluetooth Classic or Bluetooth Low Energy (BLE). Bluetooth enables communication with other Bluetooth devices, such as smartphones, tablets, or other Bluetooth-enabled modules.
- For BLE, you can use libraries like BLEDevice.h in the Arduino IDE, while for Bluetooth Classic, you might use Bluetooth Serial.h.

- ESP32 has a variety of **General Purpose Input/Output (GPIO)** pins that can be used for **digital input** and **output**, **PWM control**, **analog readings**, and interfacing with other sensors or actuators.
- The pins can be configured for different roles, like **UART** for serial communication, **I2C** for interfacing with sensors, or **SPI** for high-speed data transfer.



MPU-6050 SENSOR

2.1 introduction

The MPU-6050 is a widely-used 6-axis motion tracking sensor, combining a 3-axis gyroscope and a 3-axis accelerometer into a single device. It is designed and manufactured by InvenSense, and it is primarily used to measure and track acceleration, angular velocity, and orientation. This sensor is commonly found in a variety of applications, including robotics, gaming, wearable devices, and navigation systems.

- Accelerometer: Measures acceleration along three axes (X, Y, Z), typically ranging from ±2g to ±16g.
- Gyroscope: Measures angular velocity or the rate of rotation along the same three axes, typically ranging from ±250°/s to ±2000°/s.

The MPU-6050 is used in conjunction with a microcontroller, such as an Arduino, to interface with the sensor and collect data for motion detection and analysis. Its small form factor, high precision, and low cost make it an ideal choice for a wide range of consumer electronics, robotics, and motion-tracking applications.

2.2 Importance of MPU-6050

The MPU-6050 plays a crucial role in various applications due to its combined functionality, cost-effectiveness, and ease of use. Some of its key features and importance include:

1. Compact Design:

- The MPU-6050 combines both the accelerometer and gyroscope into a single compact chip, reducing the size and complexity of designs where motion detection is required.

2. Motion Tracking and Orientation Detection:
   - The sensor allows for precise tracking of acceleration and angular velocity, which is essential for applications like motion-controlled gaming, wearable devices, drone stabilization, and robotics.
3. Cost-Effective:
   - The MPU-6050 is an affordable sensor that provides significant functionality, making it accessible for both hobbyists and professionals working on cost-sensitive projects.
4. Low Power Consumption:
   - It is designed to operate with low power consumption, making it suitable for battery-powered devices and applications where energy efficiency is critical.
5. Versatile Applications:
   - The sensor is widely used in a variety of fields, such as:
     - Robotics: To track movement and balance robots.
     - Inertial Measurement Units (IMUs): Used in navigation systems for detecting orientation changes.
     - Sports Technology: For tracking movement and analyzing performance in sports.
     - Healthcare Devices: For monitoring body movement in wearable health monitoring systems.
     - Drones: To stabilize flight and measure orientation.
6. Easy Integration with Microcontrollers:
   - The MPU-6050 uses the I2C communication protocol, which simplifies its integration with microcontrollers like Arduino, Raspberry Pi, and others, making it easy to incorporate into DIY projects.
7. Sensor Fusion Capability:
   - The MPU-6050 can be paired with software algorithms (e.g., Kalman Filter) for sensor fusion, enabling the combination of accelerometer and gyroscope data for more accurate orientation and motion tracking.
   -

2.3 How to Connect and Use the MPU-6050

Connecting the MPU-6050 to a microcontroller is straightforward. It uses the I2C (Inter-Integrated Circuit) communication protocol, which only requires two signal lines: SDA (Serial Data Line) and SCL (Serial Clock Line).
Here's a step-by-step guide on how to connect and use the MPU-6050 with a microcontroller, such as an Arduino:
1. Wiring the MPU-6050:

The sensor is typically connected using the following pins:

- VCC: Connect to 3.3V or 5V (depending on your microcontroller).
- GND: Connect to Ground.
- SDA: Connect to the SDA pin of the microcontroller (e.g., A4 on Arduino Uno).
- SCL: Connect to the SCL pin of the microcontroller (e.g., A5 on Arduino Uno).
- INT (Interrupt): This pin can be used to trigger interrupts, but it's optional for basic applications.

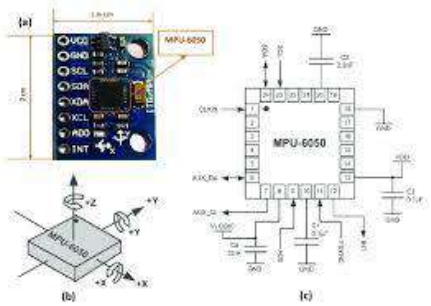
2. Library Installation:

In the Arduino IDE, you need to install the appropriate library to communicate with the MPU-6050.

- Open Arduino IDE, go to Sketch > Include Library > Manage Libraries.
- In the Library Manager, search for MPU6050 or I2Cdev and install the library.

3. Basic Example Code:

Once the sensor is connected and the library is installed, you can use the following code to read data from the MPU-6050 and display it in the Serial Monitor.



MAX-30102 SENSOR

3.1 Introduction to MAX-30102

The **MAX-30102** is an advanced **optical sensor** designed by **Maxim Integrated**. It is a **pulse oximeter** and **heart-rate sensor** integrated into a small, low-power device. The MAX-30102 is widely used for measuring **heart rate (HR)** and **blood oxygen saturation (SpO2)** in wearable devices, health-monitoring systems, fitness trackers, and medical applications.

The sensor works by using infrared light to measure the amount of oxygen in the blood and the heart rate by detecting the changes in blood volume that occur with each heartbeat. The MAX-30102 integrates multiple key components into one module, including a **photodetector**, **LEDs** (for emitting light), and a **proximity sensor**, all designed to work together for precise measurements.

The MAX-30102 is an improvement over its predecessor, the **MAX-30100**, providing more accurate readings, faster performance, and a better signal-to-noise ratio, making it an excellent choice for health and wellness applications.

3.2 Importance of MAX-30102 Sensor

The **MAX-30102** plays a crucial role in various health monitoring applications due to its compact size, precision, and low power consumption. Below are some reasons why it is important:

1. **Accurate Health Monitoring**:
    - The MAX-30102 allows for real-time, non-invasive measurement of vital signs, including heart rate and blood oxygen saturation (SpO2), which are critical indicators of a person's health.
    - It can be used to track **oxygen levels** in patients with respiratory issues, monitor **heart rate** during exercise, or keep track of overall health in wellness applications.
2. **Small Size and Low Power Consumption**:
    - This sensor is compact and lightweight, making it ideal for portable and wearable devices like **fitness trackers**, **smartwatches**, and **health monitoring bands**.
    - Its low power consumption ensures long battery life in battery-operated devices, which is essential for continuous health monitoring in wearable applications.
3. **Ease of Integration**:
    - The MAX-30102 uses the **I2C** interface, which simplifies its integration with microcontrollers like **Arduino**, **Raspberry Pi**, and other embedded systems. This allows for easy development of custom health-monitoring applications or IoT devices.
    - Its simple communication protocol makes it accessible for both beginners and professionals.
4. **Versatile Applications**:
    - **Wearable Devices**: The MAX-30102 is commonly used in smartwatches and fitness bands to monitor heart rate and SpO2 levels during workouts.
    - **Medical Devices**: The sensor is ideal for medical-grade equipment that monitors blood oxygen levels in hospitals or for home care.
    - **Fitness and Wellness**: It enables users to track their heart rate during exercise or measure SpO2 during high-altitude activities.
5. **Integrated with Multiple Sensors**:
    - The MAX-30102 contains both a **Red** and **IR LED** that are used to detect the absorption of light by blood vessels, which makes it capable

of calculating SpO2 accurately. It also includes an **ambient light sensor** for better performance under varying light conditions.

6. **Affordable**:
    - The MAX-30102 offers great functionality at a relatively low cost, which makes it affordable for both developers and consumers. This affordability enables broader use in consumer health devices, especially in the wearables market.
7. **Improved Signal Quality**:
    - Compared to earlier models, the MAX-30102 provides a better signal-to-noise ratio, meaning it produces more accurate readings even in environments with a lot of movement or noise.
    -

3.3 How to Connect the MAX-30102 Sensor

The **MAX-30102** sensor communicates with microcontrollers via the **I2C** protocol. It is very easy to connect and interface with microcontrollers like **Arduino**, **Raspberry Pi**, and other platforms that support I2C communication. Here's a step-by-step guide on how to connect and use the MAX-30102 with an Arduino:

1. Wiring the MAX-30102:

- The sensor uses I2C communication, so it requires only **4 wires** for connection: **VCC**, **GND**, **SDA**, and **SCL**.

The wiring for connecting the MAX-30102 to an Arduino is as follows:

- **VCC**: Connect to the 3.3V or 5V pin of the Arduino (based on the sensor module's specifications).
- **GND**: Connect to Ground (GND) on the Arduino.
- **SDA**: Connect to the **SDA** pin on the Arduino (on Arduino Uno, this is **A4**).
- **SCL**: Connect to the **SCL** pin on the Arduino (on Arduino Uno, this is **A5**).
- **INT** (optional): This is the interrupt pin, which is often not used for basic applications.

2. Library Installation:

- To simplify working with the MAX-30102 on Arduino, you'll need to install the necessary libraries for communication and data reading. One of the popular libraries is the **SparkFun MAX3010x Library**.

To install the library:

1. Open the **Arduino IDE**.
2. Go to **Sketch > Include Library > Manage Libraries**.
3. Search for **SparkFun MAX3010x** and install it.

DS-18620 SENSOR

4.1 Introduction to DS-18620 Sensor

The **DS-18620** sensor is a **temperature and humidity sensor** designed by **Maxim Integrated**. It is part of the **DS18B20 family** of sensors but is typically more advanced, often providing additional features for better environmental measurements. The **DS-18620** sensor is used to measure both **temperature** and **humidity** in various applications such as home automation, weather stations, HVAC (heating, ventilation, and air conditioning) systems, industrial environments, and agricultural systems.
The sensor operates on the **1-Wire communication protocol**, making it simple to integrate with microcontrollers such as **Arduino** and **Raspberry Pi**. It's known for its **low power consumption**, **accurate readings**, and ease of use in different IoT-based applications.

4.2 Importance of DS-18620 Sensor

The **DS-18620** sensor is important due to its multiple features and advantages:

1. **Accurate Temperature and Humidity Measurement**:
    - The sensor can measure **temperature** with high accuracy, typically ranging from -40°C to 125°C, with a resolution of **0.5°C**.
    - **Humidity** measurement is also precise, typically ranging from **0% to 100% RH** (Relative Humidity) with high accuracy, making it ideal for controlling environments where temperature and humidity are critical factors.
2. **Low Power Consumption**:
    - The DS-18620 operates with **very low power consumption**, making it ideal for battery-powered devices and IoT applications where energy efficiency is important.
    - This feature is particularly useful in remote applications or systems that need to operate over extended periods without frequent battery replacements.
3. **Simple 1-Wire Interface**:
    - The **1-Wire communication protocol** allows multiple sensors to be connected to a single data line, reducing the number of pins required on a microcontroller and simplifying wiring.
    - It allows for easy communication with microcontrollers like **Arduino**, **Raspberry Pi**, and other platforms without needing complex circuitry.
4. **Compact and Durable**:

- The small, compact form factor of the DS-18620 makes it easy to integrate into tight spaces, such as in embedded systems or wearable devices.
- The sensor is also designed to be durable and reliable, making it suitable for both indoor and outdoor applications, even in harsh environments.

5. **Real-Time Monitoring**:
   - The sensor allows for real-time monitoring of environmental conditions, enabling automatic adjustments in systems such as **HVAC**, **smart home systems**, or **agriculture**.
   - It's widely used in applications where precise and consistent environmental data is crucial, such as in **greenhouses**, **smart homes**, or **weather stations**.
6. **Multiple Applications**:
   - **Weather Stations**: Provides real-time temperature and humidity data to weather monitoring systems.
   - **Smart Homes**: Helps control home climate conditions by integrating into home automation systems.
   - **Industrial Systems**: Monitors temperature and humidity to ensure proper conditions in factories or warehouses.
   - **Agriculture**: Used to maintain optimal growing conditions for crops in controlled environments like greenhouses.
7. **Affordable and Easy to Use**:
   - The DS-18620 sensor is affordable, making it accessible for both hobbyists and professionals. It is easy to use with various microcontroller platforms, supported by numerous libraries and resources.
   -

4.3 How to Connect the DS-18620 Sensor

The **DS-18620** sensor uses the **1-Wire** communication protocol, which simplifies its connection and wiring. The following steps show how to connect and use the DS-18620 sensor with an **Arduino** or a **Raspberry Pi**:

1. Wiring the DS-18620 Sensor:

The DS-18620 typically has three pins for connection:

- **VCC**: Connect to the **3.3V** or **5V** pin of the microcontroller.
- **GND**: Connect to **Ground (GND)**.
- **Data**: The data pin is used for **1-Wire** communication. This pin connects to a digital pin on the microcontroller (e.g., **D2** on Arduino).

Additionally, a **pull-up resistor** (typically **4.7kΩ**) is required between the **VCC** and **Data** pins to ensure proper data transmission.

For an **Arduino** connection:

- **VCC**: 3.3V or 5V pin on the Arduino board.
- **GND**: GND pin on the Arduino.
- **Data**: Digital pin (e.g., D2 or any digital pin).

For a **Raspberry Pi** connection:

- **VCC**: 3.3V (Raspberry Pi's 3.3V pin).
- **GND**: GND pin.
- **Data**: GPIO pin (e.g., GPIO4 or another GPIO pin).

2. Library Installation:

To use the **DS-18620** sensor with an **Arduino**, the **OneWire** library is typically required for communication. If you're using the sensor for **temperature and humidity measurements**, you can use the **DallasTemperature** library, which is built on top of the OneWire library.

To install the libraries:

- Open **Arduino IDE**.
- Go to **Sketch > Include Library > Manage Libraries**.
- Search for **OneWire** and install it.
- Also, search for **DallasTemperature** and install it.

GPS SENSOR

5.1 Introduction to GPS Sensor

A **GPS sensor** is a device that allows you to receive signals from the **Global Positioning System (GPS)** satellites to determine the precise geographical location of the sensor on Earth. These sensors are integral to a wide variety of applications that require location data, such as in **navigation systems**, **tracking devices**, and **location-based services**.

The GPS sensor works by receiving signals from a network of satellites orbiting the Earth. These satellites transmit information regarding their position and the exact time at which the signal was sent. The GPS receiver uses this data to triangulate its own position by comparing the time delay between when the signal was sent and

when it was received. This allows it to calculate the distance from multiple satellites and compute the device's exact coordinates (latitude, longitude, and altitude).
The GPS sensor module can be connected to a microcontroller (like **Arduino**, **Raspberry Pi**, or other embedded systems) to provide real-time location information. Many GPS sensors also feature additional functionalities, such as **velocity** and **time** information.

5.2 Importance of GPS Sensor

The **GPS sensor** is important for various reasons, as it provides precise and reliable location data that is essential for several modern applications:

1. **Navigation and Mapping**:
    - GPS sensors are crucial for modern **navigation systems** in vehicles, smartphones, and other portable devices. They help users determine their exact location on a map and provide real-time directions to reach a destination.
    - GPS is used in **GPS navigation systems** found in cars, trucks, boats, planes, and even in walking directions, helping users find the most efficient routes.
2. **Tracking**:
    - **GPS sensors** are widely used in **asset tracking**, including tracking vehicles, shipments, pets, and even individuals. They are often used in **fleet management systems** for logistics companies and also in **wearable devices** for real-time tracking of people, such as children or elderly family members.
    - It's also used in **geofencing** technology, where a virtual boundary is set, and notifications are sent when the tracked object enters or exits the boundary.
3. **Location-Based Services**:
    - Many mobile apps, like **weather apps**, **fitness trackers**, and **ride-sharing apps** (e.g., Uber, Lyft), rely on GPS sensors to offer location-based services. This allows users to interact with the environment, such as locating nearby points of interest or sharing their real-time location.
    - **Social media apps** also use GPS for **location tagging**, allowing users to share where they are or to post photos with geographic data.
4. **Surveying and Geodesy**:
    - GPS sensors are widely used in **surveying** and **geodesy** for accurate measurement of land, construction projects, and even geological studies. High-precision GPS is used for **topographic surveys**, **land marking**, and even for creating high-resolution maps.
5. **Timing and Synchronization**:

    - GPS signals are used for time synchronization in various systems, such as **telecommunication networks**, **power grids**, and **financial systems**, where precise time is critical.
    - GPS provides **atomic clock-level accuracy** for timing applications, which is vital in many industrial, scientific, and military systems.
6. **Aviation and Maritime**:
    - GPS is essential for **aircraft navigation** in both civil and military aviation, ensuring that aircraft can safely travel between destinations without requiring ground-based navigation infrastructure.
    - In the **maritime industry**, GPS helps with **marine navigation**, helping ships and boats to plot their courses in the open sea.
7. **Agriculture**:
    - In **precision agriculture**, GPS sensors enable **automated farming machinery** to work accurately, such as in **autonomous tractors** or **drones** for monitoring crops or fields.
8. **Search and Rescue Operations**:
    - GPS sensors play a crucial role in **search and rescue operations** by helping rescuers locate missing people or stranded vehicles in remote areas, often in combination with **drones** or **satellite phones**.
    -

5.3 How to Connect GPS Sensor

The GPS sensor typically communicates with microcontrollers and embedded systems using **serial communication** via **UART (Universal Asynchronous Receiver-Transmitter)** or **I2C** protocols. The most commonly used GPS modules include the **u-blox NEO-6M**, **NEO-7M**, and **NEO-M8N**, which communicate using UART (TX/RX pins).
Here's how to connect a common GPS sensor (e.g., **u-blox NEO-6M**) to an **Arduino** or **Raspberry Pi**:

1. Wiring the GPS Sensor:

- **GPS Module Pins**:
    - **VCC**: Power input pin (usually 3.3V or 5V, depending on the sensor).
    - **GND**: Ground.
    - **TX**: Transmit pin, which sends data to the microcontroller (Arduino/Raspberry Pi).
    - **RX**: Receive pin, which gets data from the microcontroller (optional, depending on if you need bidirectional communication).
    - **ANT**: Antenna pin (if the module has an external antenna for better reception).

Connecting to an Arduino (Using UART):

1. **VCC**: Connect to **5V** (for Arduino boards that support 5V) or **3.3V** (for boards that only support 3.3V).

2. **GND**: Connect to **GND** on Arduino.
3. **TX** (GPS Module): Connect to **RX** pin on the Arduino (e.g., **Pin 4**).
4. **RX** (GPS Module): Connect to **TX** pin on the Arduino (e.g., **Pin 3**).

If using an **Arduino Uno**, you may want to use **software serial** to communicate with the GPS module on other pins since the default hardware serial is used for programming and debugging.

Connecting to Raspberry Pi:

1. **VCC**: Connect to **3.3V** pin on the Raspberry Pi (ensure the GPS module can work with 3.3V).
2. **GND**: Connect to **GND** pin.
3. **TX** (GPS Module): Connect to **RX** pin (GPIO 15 or other UART pins).
4. **RX** (GPS Module): Connect to **TX** pin (GPIO 14 or other UART pins).

2. Library Installation:

For Arduino, you will need a library that allows you to communicate with the GPS module and process the data. A popular library for this is the **TinyGPS++** library.

To install it:

- Open **Arduino IDE**.
- Go to **Sketch > Include Library > Manage Libraries**.
- Search for **TinyGPS++** and install it.

For Raspberry Pi, you can use **PySerial** in Python to communicate with the GPS module over UART.

WIRELESS CHARGE CIRCUIT

6.1 Introduction to Wireless Charging Circuit

A **wireless charging circuit** is a technology that allows the transfer of electrical energy from a power source to a device without the use of physical connectors or cables. The technology behind wireless charging is based on **inductive charging**, which uses electromagnetic fields to transfer energy between two coils: a transmitter

coil (in the charging station) and a receiver coil (in the device being charged). The transmitter coil generates an electromagnetic field, and the receiver coil in the device captures this field, converting it back into electrical energy to charge the battery.
The most common standard for wireless charging is the **Qi (pronounced "chee") standard**, which is widely used in smartphones, wearables, and other small electronics. The wireless charging circuit can be designed to handle different power levels for devices ranging from low-power gadgets (like smartwatches) to high-power devices (like electric vehicles).

6.2 Importance of Wireless Charging Circuit

1. **Convenience**:
    - Wireless charging eliminates the need for physical cables and connectors, making it more convenient to charge devices. Simply placing the device on a charging pad or station is enough to begin charging, which is particularly useful in public places like airports, cafes, or offices where charging cables might not be available.
    - **Multiple devices** can be charged simultaneously using wireless charging stations with multiple pads, reducing cable clutter.
2. **Durability and Safety**:
    - Since there are no exposed wires or connectors, wireless charging circuits reduce wear and tear on charging ports and cables. This can lead to a longer lifespan for both the device and the charging equipment.
    - It also improves **safety**, as there is no risk of exposed or frayed wires that could cause electrical shorts or fires.
3. **Aesthetics**:
    - Wireless charging provides a clean and minimalist charging solution. Devices can be charged without the need for unsightly cords, making it aesthetically pleasing for users who prefer a sleek and wire-free environment.
4. **Faster Charging**:
    - Wireless charging technology, especially with recent advancements like **Fast Charging (Qi Fast Charging)**, allows devices to charge at a relatively fast rate, often comparable to wired charging speeds.
5. **Compatibility and Flexibility**:
    - Many wireless chargers are built to be compatible with a variety of devices, including smartphones, wireless headphones, and other electronic gadgets. As more devices integrate wireless charging support, this technology will continue to be a standard feature in consumer electronics.
6. **Smart Integration**:

- Wireless charging can be integrated with smart devices, allowing for features like **automatic charging**, where the device starts charging as soon as it is placed on the charger. It can also allow for energy-saving features like **standby mode**, where the charger only supplies power when the device is placed on it.

7. **Health and Comfort**:
   - In the medical field, wireless charging circuits are increasingly being used in medical devices, like **implants**, **hearing aids**, or **prosthetics**, where direct physical contact for charging is not ideal.
8. **Environmental Benefits**:
   - Wireless charging reduces the need for disposable cables, which can have an environmental impact if not recycled properly. It also helps reduce the wear and tear of connectors, meaning fewer devices may need repairs due to broken charging ports.

How to Connect a Wireless Charging Circuit

The wireless charging system consists of two main parts: the **transmitter circuit** (charging station) and the **receiver circuit** (in the device being charged). Below is a breakdown of how each part connects and functions.

Components of Wireless Charging Circuit:

1. **Transmitter Circuit**:
   - **AC Power Source**: Provides the input power for the transmitter.
   - **Rectifier and Voltage Regulator**: Converts AC to DC voltage suitable for the charging coil.
   - **Oscillator Circuit**: Generates a high-frequency AC signal to drive the transmitter coil.
   - **Transmitter Coil**: Creates an electromagnetic field that transfers energy wirelessly.
2. **Receiver Circuit**:
   - **Receiver Coil**: Captures the electromagnetic field from the transmitter and converts it into electrical energy.
   - **Rectifier and Voltage Regulator**: Converts the energy back to a suitable DC voltage to charge the battery of the device.
   - **Battery Management System**: Ensures safe and efficient charging of the device's battery.
   -

6.3 How to Connect the Transmitter (Charger) Circuit:

1. **Power Supply**:
   - The transmitter circuit requires a power supply, typically from **AC mains voltage** (e.g., 120V or 240V AC). This is connected to the input of the circuit.

    - A **rectifier circuit** will convert this AC power to DC voltage to power the rest of the transmitter circuit.
2. **Oscillator and High-Frequency Generation**:
    - The oscillator circuit, usually composed of **transistor-based circuits**, generates a high-frequency AC signal (typically in the range of 100-200 kHz). This signal is essential to excite the transmitter coil.
3. **Transmitter Coil**:
    - The high-frequency AC signal is fed to the transmitter coil, which generates an alternating electromagnetic field. This coil is usually positioned at the surface of the wireless charger pad.
4. **Connection to Device (Receiver Circuit)**:
    - When the device is placed over the transmitter coil, the receiver coil in the device picks up the electromagnetic energy. This is a wireless power transfer process where energy is transferred through the air.

How to Connect the Receiver (Device) Circuit:

1. **Receiver Coil**:
    - The receiver coil in the device is placed close to or aligned with the transmitter coil in the charging station. It receives the electromagnetic field generated by the transmitter coil.
2. **Rectification and Voltage Regulation**:
    - The electromagnetic field is converted back into DC power by the receiver coil, which is then rectified and regulated by the rectifier and voltage regulator circuit.
3. **Battery Charging**:
    - The regulated DC voltage is then used to charge the battery in the device. The **battery management system** ensures that the battery is charged safely and efficiently by monitoring charging voltage, current, and temperature.

Example of Wireless Charging Circuit (Transmitter Side):

- **Transmitter Side**:
    - **AC Input → Rectifier Circuit → DC Power Supply → Oscillator Circuit → Transmitter Coil**.
- **Receiver Side**:
    - **Receiver Coil → Rectifier and Voltage Regulator → Battery Charging Circuit → Battery**.

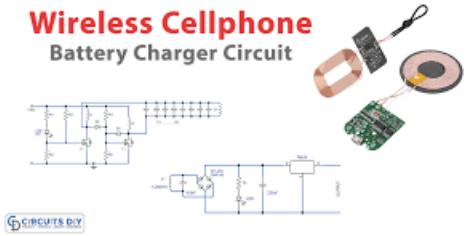
Connecting Wireless Charging to Devices (e.g., Smartphone):

1. **Transmitter**:
    - In a wireless charging pad, the **transmitter circuit** is embedded into the pad, where users place their devices.
2. **Receiver**:

    - The device, such as a **smartphone**, contains the **receiver coil** built inside its body or case. Upon alignment with the transmitter, the device receives power wirelessly to charge its battery.
3. **Inductive Power Transfer**:
    - The electromagnetic field is transferred wirelessly between the transmitter and receiver, completing the charging process.

Safety Considerations:

- **Overcharging Protection**: The receiver circuit should have **overcharge protection** to prevent damage to the device's battery.
- **Foreign Object Detection (FOD)**: Some advanced wireless charging systems feature **FOD** that detects any foreign objects (e.g., metal) between the charging coils, preventing potential damage or safety hazards.



BATERRY INDUCTOR CIRCUIT

7.1 Introduction to Battery Inductor Circuit

A **battery inductor circuit** refers to an electrical circuit that combines **batteries** and **inductors** to store and manage energy for various applications. Inductors are passive components that store energy in a magnetic field when electrical current passes through them. When used with a battery, inductors can smooth out fluctuations in voltage and current, store energy temporarily, or be part of a more complex system like **boost converters**, **buck converters**, or **flyback circuits** for energy conversion and regulation.

Battery inductor circuits are commonly found in power supply systems, **energy harvesting systems**, **wireless charging systems**, and **DC-DC converters**. They can also be seen in **electrical motors**, where inductors work in conjunction with batteries to create rotational motion or in **power management circuits** for optimizing battery usage and extending the battery life.

The combination of batteries and inductors is essential for **efficient energy conversion**, **stabilization**, and **voltage regulation** in a variety of electrical devices and systems.

7.2 Importance of Battery Inductor Circuit

1. **Energy Storage and Conversion**:
    - Inductors are important in circuits for converting and stabilizing the energy from batteries. They can store energy temporarily and release it when needed, smoothing out the voltage or current for more stable operation.
    - In energy conversion applications, inductors are crucial in **boosting or bucking** voltage levels to meet the power requirements of specific components in the system, like **microcontrollers**, **LEDs**, or **motors**.
2. **Voltage Regulation**:
    - In circuits that power sensitive electronics, maintaining a stable voltage is critical. An inductor can help smooth out voltage ripples caused by battery discharge or fluctuations. Inductor-based filters or regulators ensure consistent performance of connected devices, even as the battery voltage decreases.
3. **Power Efficiency**:
    - Battery inductor circuits are used in **DC-DC converters**, which help achieve efficient power conversion. For instance, when converting a higher voltage from the battery to a lower one for certain devices (or vice versa), inductors play a key role in reducing energy loss, making the system more efficient.
    - Inductors can help optimize the use of battery power by managing how energy is transferred and stored.
4. **Reducing Power Loss**:
    - Inductors help reduce energy loss in systems by smoothing out electrical currents. This is particularly beneficial when dealing with high currents, where inductors can minimize **voltage spikes** and **current surges** that could potentially damage the circuit.
5. **Inductive Charging and Wireless Power**:
    - In wireless charging systems, inductors are used in both the transmitter and receiver circuits. They play an essential role in transferring energy between the charging pad and the device battery wirelessly, facilitating the wireless charging process.
6. **Smooth Operation of Motors**:
    - In electric motor circuits powered by batteries, inductors help regulate the current and smooth the operation of the motor by reducing the chances of current spikes that can lead to inefficient motor performance.

7. **Battery Life Extension**:
    - By stabilizing the power supply to the device, battery inductor circuits can extend the overall lifespan of the battery. Inductors help to reduce the load on the battery, ensuring that it operates more efficiently and lasts longer.
    -

7.3 How to Connect a Battery Inductor Circuit

To connect a **battery inductor circuit**, the following components are typically required:

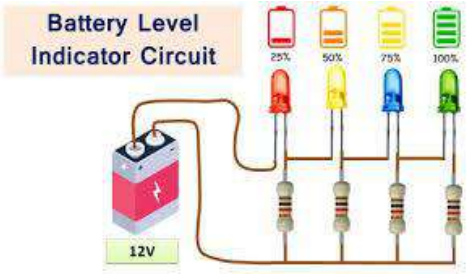
1. **Battery**: A battery (e.g., **Li-ion**, **Li-Po**, or **AA**) will provide the necessary power source for the circuit.
2. **Inductor**: A coil of wire wound around a core, the inductor is used to store energy in a magnetic field. The choice of inductor depends on the desired frequency and power requirements of the circuit.
3. **Load**: This could be any component or device that consumes power, such as a motor, LED, microcontroller, or sensor.
4. **Switching Element**: A switching element such as a **transistor** (MOSFET, BJT, etc.) is used to control the energy transfer between the inductor and the load.
5. **Control Circuit**: The control circuit, such as a **DC-DC converter** or **pulse-width modulation (PWM)** controller, manages the energy transfer and ensures the battery and inductor work together effectively.
6. **Capacitors**: In many circuits, capacitors are used in conjunction with inductors to smooth voltage and reduce ripple effects.

Steps to Connect a Battery Inductor Circuit

1. **Battery Connection**:
    - Connect the **positive terminal** of the battery to the input side of the inductor. The negative terminal should be grounded.
2. **Inductor Integration**:
    - Place the **inductor** in series with the battery and the load. The inductor's primary role is to store energy when the current is flowing through it and release it when needed.
3. **Switching Element**:
    - If using a **DC-DC converter** circuit, place a **transistor or MOSFET** as the switching element between the battery, inductor, and load. The switching element will regulate how much energy is transferred by periodically opening and closing.
    - The frequency of the switching will determine how much energy the inductor stores and releases.
4. **Control Circuit**:

- The **control circuit** should be connected to the switching element (MOSFET or transistor). This control circuit is responsible for adjusting the duty cycle of the switch to ensure the desired voltage is supplied to the load.

5. **Load Connection**:
   - Connect the **load** to the output side of the inductor (after the energy has passed through the inductor and regulator). This load could be any electronic component like a motor, sensor, or microcontroller that requires a stable supply of power.
6. **Capacitors for Filtering**:
   - Capacitors are typically connected in parallel with the load or in series with the inductor to reduce ripple and stabilize the voltage across the circuit.
7. **Testing**:
   - After connecting the circuit, test it by measuring the voltage at the load and ensuring the inductor is properly smoothing the current. The load should receive a stable voltage even as the battery's charge decreases.



FEDBACK CIRCUIT

8.1 Introduction to Feedback Circuit

A **feedback circuit** is a fundamental concept in electrical engineering, widely used in control systems, amplifiers, oscillators, and various signal-processing applications. It involves taking a portion of the output signal from a system and returning it to the input. The feedback can either be **positive** or **negative**, depending on the desired effect on the system.

- **Negative feedback** occurs when the feedback signal is inverted, leading to a reduction in the output, which stabilizes the system and improves its performance.

- **Positive feedback** occurs when the feedback signal is in phase with the input, amplifying the output. This can lead to system instability, but it is useful in applications like oscillators and certain amplifiers.

Feedback circuits are typically used in systems that require a high degree of stability, precision, or adaptability, such as **operational amplifiers (op-amps)**, **audio amplifiers**, **control systems**, and **voltage regulators**.

8.2 Importance of Feedback Circuit

1. **Stability and Control**:
   - **Negative feedback** is widely used to stabilize a system. In amplifiers, for example, it ensures that the output remains constant and proportional to the input, preventing distortion or oscillation.
   - In control systems, feedback helps maintain the desired output by continually adjusting the input, improving the system's stability and responsiveness.
2. **Accuracy and Precision**:
   - Feedback helps **correct errors** in the output signal. In an operational amplifier, for example, negative feedback ensures that the output is a precise and accurate amplification of the input signal.
   - This is crucial in systems that require highly accurate responses, such as **measurement instruments** and **sensor systems**.
3. **Gain Control**:
   - Feedback circuits control the gain (amplification) in various applications. In amplifiers, negative feedback is used to reduce the gain to a more manageable level, avoiding clipping and distortion. In contrast, positive feedback can be used to increase gain, often in applications requiring sharp transitions like oscillators.
4. **Improved Bandwidth**:
   - Negative feedback in amplifiers increases **bandwidth** by reducing the gain at high frequencies. This results in more uniform performance across a wide range of signal frequencies, making the system more versatile.
5. **Noise Reduction**:
   - Negative feedback helps reduce **noise** in systems. It filters out unwanted components and ensures that the output closely follows the intended input, leading to higher **signal-to-noise ratios**.
6. **Adaptability**:
   - Feedback circuits make systems **adaptive**. By continuously monitoring the output and adjusting the input accordingly, feedback allows systems to maintain performance even when external conditions change.
7. **Oscillators and Signal Generation**:

- Positive feedback is used in **oscillator circuits** to generate periodic waveforms. The feedback ensures that the system reaches a point where it starts oscillating, which is essential in creating sine waves, square waves, or other signals used in communication, clock generation, and RF circuits.

8. **Regulation**:
   - In power supplies, **feedback** ensures that the output voltage remains stable despite changes in the input voltage or load conditions. This is particularly important in **voltage regulators** and **DC-DC converters** to maintain consistent output.
   -

8.3 How to Connect a Feedback Circuit

To connect a feedback circuit, several key components are involved, such as amplifiers (often operational amplifiers or transistors), resistors, capacitors, and sometimes inductors, depending on the application. Below are the basic steps and methods to integrate a feedback loop into an electronic circuit.

Components of a Feedback Circuit

1. **Amplifier**:
   - The core of most feedback circuits is an **amplifier** (e.g., **op-amp**, **transistor**, or **operational amplifier**). The amplifier amplifies the input signal, and part of the output is fed back into the input.
2. **Feedback Path**:
   - The **feedback path** is where the output signal is returned to the input. This can be achieved using a **resistor** or **capacitor** to control the amount of feedback. The path may be direct or include additional components to modify the feedback characteristics.
3. **Feedback Network**:
   - This consists of the components (resistors, capacitors) that define the feedback ratio or control the behavior of the feedback loop. In **negative feedback**, the network will reduce the output in a controlled manner. In **positive feedback**, the network amplifies the output.
4. **Input**:
   - The input signal is applied to the system, and the feedback mechanism adjusts the output based on this signal.
5. **Output**:
   - The output is the result of the feedback process, which is either amplified, regulated, or oscillated, depending on the circuit design.

Connecting Negative Feedback Circuit (Op-Amp Example)

One of the most common applications of feedback circuits is in **operational amplifiers** (op-amps). Here's how to connect a negative feedback circuit with an op-amp:

1. **Op-Amp Setup**:
    - Connect the **non-inverting input** of the op-amp to the signal input (the source of the signal you want to amplify or process).
    - The **inverting input** is connected to the feedback path.
2. **Feedback Path**:
    - Place a **resistor** (or combination of resistors) between the op-amp output and the inverting input. This resistor creates the feedback loop.
    - If a **capacitor** is added in parallel with the resistor, it will affect the frequency response and behavior of the feedback loop.
3. **Power Supply**:
    - Connect the op-amp to a **dual power supply** (positive and negative voltages) or a **single supply** depending on the op-amp's requirements.
4. **Load**:
    - The output from the op-amp goes to the **load**, which could be a speaker, sensor, or any other system component that receives the processed signal.

Example: Non-Inverting Amplifier with Feedback

In a **non-inverting amplifier** configuration (using an op-amp), the feedback path is used to control the gain:

1. **Input**: The input signal is applied to the non-inverting input.
2. **Feedback Resistor**: A resistor is placed between the output and the inverting input of the op-amp, creating a feedback loop. The amount of feedback determines the gain.
3. **Output**: The op-amp amplifies the input signal with the feedback controlling how much amplification occurs.

The gain of a non-inverting amplifier with negative feedback is determined by the resistor values in the feedback loop:

Gain=1+RfRin\text{Gain} = 1 + \frac{R_f}{R_{in}}Gain=1+RinRf

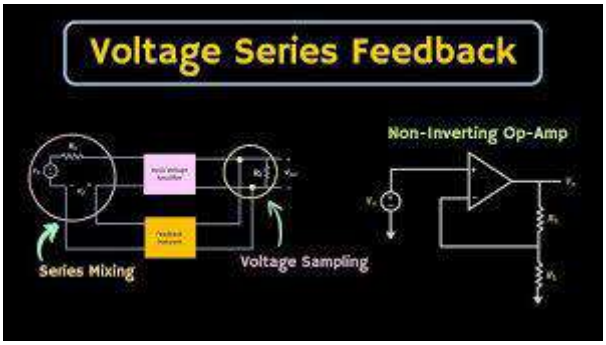
Where:

- RfR_fRf is the feedback resistor.
- RinR_{in}Rin is the resistor between the inverting input and ground.

Example: Positive Feedback (Oscillator Circuit)

In an **oscillator circuit**, positive feedback is used to generate periodic signals:

1. **Oscillator Setup**: Use components such as resistors, capacitors, and an amplifier (often an op-amp or transistor).
2. **Feedback Network**: Set up a feedback network to ensure that the feedback is in phase with the input (positive feedback). This causes the circuit to continuously oscillate once started.
3. **Output**: The output of the oscillator is a periodic waveform, such as a sine wave or square wave, depending on the design.



CODING FOR PROJECT

9.1 code

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"

const char* ssid = "Wasef";
const char* password = "m4346525";
String url = "http://192.168.1.5/techno/index.php";

// DS18B20 - إعداد مستشعر الحرارة
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// MPU6050 - إعداد مستشعر الحركة
Adafruit_MPU6050 mpu;
int stepCount = 0;
float threshold = 1.2;
bool stepDetected = false;
```

```
// MAX30102 - إعداد مستشعر الأوكسجين ومعدل ضربات القلب
MAX30105 particleSensor;
#define BUFFER_SIZE 100
uint32_t irBuffer[BUFFER_SIZE];
uint32_t redBuffer[BUFFER_SIZE];

// المتغيرات
float temperature = 0.0;
int32_t heartRate = 0;
int32_t spo2 = 0;
float caloriesBurned = 0.0;
float breathingRate = 0;

unsigned long lastTime = 0;
unsigned long breathingInterval = 0;
int breathingCount = 0;

// تعريف دبابيس SDA و SCL
#define SDA_PIN 5
#define SCL_PIN 18

void connectWifi() {
    WiFi.mode(WIFI_OFF);
    delay(1000);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to WiFi");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
}

void sendDataToServer(String postData) {
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(url);
        http.addHeader("Content-Type", "application/x-www-form-urlencoded");
        int httpCode = http.POST(postData);
        if (httpCode > 0) {
            String response = http.getString();
            Serial.println("Server Response: " + response);
```

```
        } else {
            Serial.println("Error in POST: " + String(httpCode));
        }
        http.end();
    } else {
        Serial.println("WiFi not connected!");
    }
}

void setup() {
    Serial.begin(9600);
    connectWifi();
    sensors.begin();

    if (!mpu.begin()) {
        Serial.println("Failed to connect to MPU6050");
        while (1);
    }
    if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
        Serial.println("MAX30102 not found");
        while (1);
    }
    particleSensor.setup();
    particleSensor.setPulseAmplitudeRed(0x0A);
    particleSensor.setPulseAmplitudeIR(0x0A);
}

void loop() {
    // قراءة درجة الحرارة
    sensors.requestTemperatures();
    temperature = sensors.getTempCByIndex(0);

    // قراءة بيانات MPU6050
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
    float accX = a.acceleration.x;
    float accY = a.acceleration.y;
    float accZ = a.acceleration.z;
    float acceleration = sqrt(accX * accX + accY * accY + accZ * accZ);
    static float prevAcceleration = 0;
    float deltaAcc = abs(acceleration - prevAcceleration);
    if (deltaAcc > threshold && !stepDetected) {
        stepCount++;
        stepDetected = true;
```

```
    }
    if (deltaAcc < threshold) {
        stepDetected = false;
    }
    prevAcceleration = acceleration;

    // قراءة MAX30102
    for (int i = 0; i < BUFFER_SIZE; i++) {
        while (particleSensor.check() == false);
        redBuffer[i] = particleSensor.getRed();
        irBuffer[i] = particleSensor.getIR();
    }
    int8_t hrValid, spo2Valid;
    maxim_heart_rate_and_oxygen_saturation(irBuffer, BUFFER_SIZE, redBuffer,
&spo2, &spo2Valid, &heartRate, &hrValid);

    // حساب السعرات الحرارية
    caloriesBurned = (stepCount * 0.045) + ((0.071 * heartRate) - 4.02);

    // حساب معدل التنفس
    breathingInterval = millis() - lastTime;
    if (breathingInterval > 1000) {
        breathingRate = breathingCount;
        breathingCount = 0;
        lastTime = millis();
    }

    // طباعة البيانات
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" °C");

    Serial.print("Steps: ");
    Serial.println(stepCount);

    Serial.print("Heart Rate: ");
    Serial.println(hrValid ? heartRate : 0);

    Serial.print("SpO2: ");
    Serial.println(spo2Valid ? spo2 : 0);

    Serial.print("Calories Burned: ");
    Serial.println(caloriesBurned);
```

```
    Serial.print("Breathing Rate: ");
    Serial.println(breathingRate);

    // إرسال البيانات للسيرفر
    String postData = "temperature=" + String(temperature) + "&steps=" +
String(stepCount) +
                      "&heart=" + String(heartRate) + "&oxygin=" + String(spo2) +
                      "&calories=" + String(caloriesBurned) + "&respiratory=" +
String(breathingRate);
    sendDataToServer(postData);

    delay(100);
}
```

9.2 Explanation of this code

This code is a program for the **ESP32** microcontroller that collects data from multiple sensors, including a **DS18B20** temperature sensor, an **MPU6050** motion sensor, and a **MAX30102** sensor for heart rate and SpO2 (oxygen saturation). It processes the sensor data, computes additional metrics like **breathing rate** and **calories burned**, and sends the collected information to a server over Wi-Fi using **HTTP POST**.

Code Breakdown:

1. Libraries Used:

- **WiFi.h**: Library to enable Wi-Fi connectivity.
- **HTTPClient.h**: Library for sending and receiving data via HTTP protocol.
- **OneWire.h**: Library to interface with **DS18B20** sensors that use the **OneWire** protocol.
- **DallasTemperature.h**: Library to communicate with **DS18B20** temperature sensors.
- **Adafruit_Sensor.h**: Adafruit sensor library for general sensor support.
- **Wire.h**: Library for I2C communication.
- **Adafruit_MPU6050.h**: Library to interface with the **MPU6050** motion sensor.
- **MAX30105.h**: Library for reading data from the **MAX30102** (heart rate and SpO2) sensor.

- **spo2_algorithm.h**: Library for calculating **heart rate** and **SpO2** based on data from **MAX30102**.

2. **Initial Setup:**
   - **Wi-Fi Network**: The network's **SSID** and **password** are defined to allow the ESP32 to connect to a Wi-Fi network.
   - **URL**: A **URL** is specified where the data will be sent via **HTTP POST**.
3. **Sensor Setup:**
   - **DS18B20 (Temperature)**: The **OneWire** protocol is used to read the temperature data.
   - **MPU6050 (Motion)**: This sensor measures acceleration along the **X**, **Y**, and **Z** axes to detect movement or steps based on acceleration.
   - **MAX30102 (Heart Rate & SpO2)**: This sensor is used to measure **heart rate** and **SpO2**. The **maxim_heart_rate_and_oxygen_saturation** algorithm calculates these metrics from the sensor's data.
   - Additional calculations are done to compute **breathing rate** and **calories burned**.
4. **Wi-Fi Connection:**
   - **connectWifi() function**: This function connects the ESP32 to the Wi-Fi network using the defined **SSID** and **password**. If the connection is successful, the device's **IP address** is printed to the Serial Monitor.
5. **Sensor Data Reading:**
   - **Temperature**: The temperature is read using the **DS18B20** sensor.
   - **Motion (MPU6050)**: Acceleration data is captured from the **MPU6050** sensor. If there is a significant change in acceleration (above a threshold), it is considered a step. The step count is incremented based on these movements.
   - **Heart Rate and SpO2 (MAX30102)**: The **MAX30102** sensor reads the infrared (IR) and red light signals to calculate **heart rate** and **SpO2**. If the data is valid, the heart rate and SpO2 values are extracted.
   - **Calories Burned**: This is calculated based on the number of steps taken and the heart rate using the formula:
     - caloriesBurned = (stepCount * 0.045) + ((0.071 * heartRate) - 4.02)

- **Breathing Rate**: The breathing rate is calculated based on detected breathing events over time.

6. Sending Data to the Server:

- **sendDataToServer() function**: This function sends the sensor data to the server via **HTTP POST**. The data is formatted as a URL-encoded string, and the POST request is made using the **HTTPClient** library. If the request is successful, the server response is printed to the Serial Monitor.

7. The loop function:

- The **loop function** continuously reads the sensors, computes the necessary values (temperature, heart rate, SpO2, calories burned, and breathing rate), and sends the data to the server every cycle. After each data transmission, there is a small delay of **100 milliseconds** before the next iteration.

Key Variables:

- **temperature**: Stores the temperature value read from the **DS18B20** sensor.
- **stepCount**: The number of steps detected based on the acceleration data from the **MPU6050** sensor.
- **heartRate**: The heart rate value calculated from the **MAX30102** sensor data.
- **spo2**: The SpO2 (oxygen saturation) level calculated from the **MAX30102** sensor.
- **caloriesBurned**: The total calories burned based on steps and heart rate.
- **breathingRate**: The rate of breaths calculated over a specific time period.

Flow of the Program:

1. The program begins by connecting to the Wi-Fi network.
2. It then initializes the sensors and starts reading data.
3. For each cycle, the program reads the temperature, motion data, and vital signs (heart rate and SpO2).
4. It calculates **calories burned** and **breathing rate** based on the sensor data.
5. The program sends all the data to the server using an **HTTP POST** request.
6. The process repeats continuously.

Key Points:

- **Wi-Fi Connectivity**: This code uses **Wi-Fi** to send the collected sensor data to a server for further processing or storage.
- **Multiple Sensors**: The code integrates data from **temperature**, **motion**, and **vital sign sensors** to monitor the user's health metrics.
- **Health Metrics**: The program calculates **heart rate**, **SpO2**, **calories burned**, and **breathing rate** based on the sensor readings.
- **Server Communication**: The data is transmitted to a server using HTTP POST requests, making it possible to store or process the data remotely.

Conclusion:

This code integrates multiple sensors with an **ESP32** to track various health metrics such as temperature, steps, heart rate, SpO2, calories burned, and breathing rate. It then sends the collected data to a server over Wi-Fi, allowing for real-time health monitoring.

4o mini