PULSE RATE DETECTION USING ARDUINO

A report submitted in partial fulfilment of the requirements

for the award of

Bachelor of Technology

in

Electronics and Communication Engineering

By

TIRUMALASETTY.JAHNAVI
(22BQ1A04E3)

VELAGALETI.LAHARI
(22BQ1A04F1)

SWARNA.RISHITHA
(22BQ1A04D9)

VUYYURU.MADHURI
(22BQ1A04F4)

Open-Source Hardware Tools for Electronics Engineers

(Skill Oriented Course)

UNDER THE GUIDANCE OF

Mrs.Sandhya Rani, M.Tech

Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Approved by AICTE and permanently affiliated to JNTUK)

Accredited by NBA and NAAC with 'A' Grade

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508

JUNE 2024

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBUR
(AUTONOMOUS)

CERTIFICATE

This is to certify that the Mini Project titled **"PULSE RATE DETECTION USING ARDUINO"** is a bonafide record of work done by **T. JAHNAVI (22BQ1A04E3), V. LAHARI (22BQ1A04F1), V. MADHURI (22BQ1A04F4), S. RISHITHA (22BQ1A04D9)** as part of the Skill Oriented Course Open-Source Hardware Tools for Electronics Engineers in partial fulfilment of the requirement of the degree for Bachelor of Technology in Electronics and Communication Engineering during the academic year 2023–24.

Mrs. K. Sandhya Rani	**DR. M. Y. BHANU MURTHY**
Course Coordinator	**Head of Department**

Signature of the External

DECLARATION

We, **T. JAHNAVI (22BQ1A04E3), V. LAHARI (22BQ1A04F1), V. MADHURI (22BQ1A04F4), S. RISHITHA (22BQ1A04D9)** hereby declare that the report **PULSE RATE DETECTION USING ARDUINO** entitled is done by us as part of the Skill Oriented Course Open-Source Hardware Tools for Electronics Engineers in partial fulfilment of the requirement of the degree for **Bachelor of Technology** in **Electronics and Communication Engineering** during the academic year 2023–24.

DATE:

PLACE: VVIT, NAMBUR

SIGNATURE OF THE CANDIDATES:

T. JAHNAVI (22BQ1A04E3),

V.SIVA LAHARI (22BQ1A04F1),

V.MADHURI (22BQ1A04F4),

S. RISHITHA (22BQ1A04D9).

ACKNOWLEDGEMENT

We express our sincere thanks to the Chairman, Vasireddy Venkatadri Institute of Technology. Sri Vasireddy Vidya Sagar for providing us well equipped infrastructure and environment.

We thank Dr. Y. Mallikarjuna Reddy, Principal, Vasireddy Venkatadri Institute of Technology, Nambur, for providing us the resources for carrying out the project.

Our sincere thanks to Dr. M. Y. Bhanu Murthy, Head, Department of ECE, for his co- operation and guidance which helped us to make our project successful and complete in all aspects.

We also express our sincere thanks and are grateful to our course coordinators
Mrs.K. Sandhya Rani, Assistant Professor of Department of ECE, for motivating us to make our project successful and fully complete. We are grateful for their precious guidance and suggestions.

We also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

NAME OF THE CANDIDATES:

T. JAHNAVI (22BQ1A04E3),

V.SIVA LAHARI (22BQ1A04F1),

V.MADHURI (22BQ1A04F4),

S. RISHITHA (22BQ1A04D9).

LIST OF CONTENTS

FIGURES

ABSTARCT

In modern healthcare, monitoring vital signs like pulse rate and oxygen saturation is crucial for timely diagnosis and treatment of various medical conditions. This project presents a cost-effective and efficient solution for real-time monitoring of pulse rate and oxygen saturation levels using Arduino microcontroller.

The components used in building up this project are MAX30105 sensor, Arduino microcontroller, Jumper wires, bread board, USB cables. The MAX30105 sensor module is the core component of the pulse oximeter project. It is a versatile and compact sensor capable of measuring both heart rate and oxygen saturation levels (SpO2) through non-invasive means. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse Oximetry and heart-rate signals.

The max30105 sensor is a Pulse Oximetry and heart rate monitor which is used to check the health of a person with any condition that affects blood oxygen levels, such as Heart attack, Heart failure, Lungs Cancer, Asthma. Overall, the MAX30105 pulse oximeter Arduino project offers a wide range of applications across healthcare, education, research, and technology development, demonstrating the versatility and potential impact of DIY medical device projects in addressing various societal needs and challenges.

It combines a pulse oximeter, heart-rate monitor, and particle sensor in a single package. The module features integrated LEDs (red, green, and infrared) and a photodetector, allowing it to measure reflected light from the skin for accurate readings. With internal signal processing, it enhances measurement accuracy and reduces noise. Its low power consumption makes it ideal for battery-powered devices. The MAX30105 uses an I2C interface for easy integration with microcontrollers and digital systems. It supports various measurement modes, including pulse oximetry for blood oxygen levels, heart rate monitoring, and particle detection, useful for applications like smoke detection and air quality monitoring. This versatility makes it a key component in smartwatches, fitness trackers, portable medical devices, and environmental sensors.

Overall, the MAX30105 Pulse Oximeter Arduino project exemplifies the transformative potential of DIY health monitoring solutions in improving healthcare accessibility, promoting health literacy, and driving innovation in the healthcare industry.

CHAPTER 1

INTRODUCTION

1.1 PULSE AND OXYGEN RATE DETECTION:

Pulse and oxygen rate detection is a critical aspect of monitoring vital signs in medical and fitness contexts. The pulse, or heart rate, indicates how many times the heart beats per minute, while the oxygen rate, commonly measured as blood oxygen saturation (SpO2), indicates the percentage of oxygenated hemoglobin in the blood. Together, these metrics provide essential information about cardiovascular health and respiratory efficiency.

***Fig -1.1*: Oximeter**

An oximeter is a non-invasive medical device used to measure the oxygen saturation (SpO2) level in the blood. This measurement indicates how well oxygen is being transported to parts of the body furthest from the heart, such as the arms and legs. Oximeters are widely used in various healthcare settings, from hospitals to home care, and have become essential tools for monitoring respiratory and cardiovascular health.

Oximeters are invaluable tools in modern healthcare, providing critical information about a patient's oxygenation status and cardiovascular health. Their non-invasive nature, ease of use, and reliability make them suitable for a wide range of applications, from clinical settings to personal health monitoring. As technology advances, oximeters are becoming more accurate, user-friendly, and integrated into various health monitoring systems, enhancing their role in preventative health care and chronic disease management.

1.2 IMPORTANCE

Oximeters have become an essential tool in both clinical and home settings due to their ability to provide critical information about a person's oxygenation status and overall respiratory and cardiovascular health. Here are several detailed reasons highlighting the importance of oximeters:

1. Early Detection of Hypoxemia

Hypoxemia is a condition characterized by low levels of oxygen in the blood. Oximeters can detect hypoxemia early, often before symptoms become severe. Early detection allows for timely intervention, which is crucial in preventing complications and ensuring adequate oxygen delivery to vital organs.

2. Monitoring Chronic Respiratory Diseases

For individuals with chronic respiratory diseases such as Chronic Obstructive Pulmonary Disease (COPD), asthma, and interstitial lung disease, regular monitoring of blood oxygen levels is essential. Oximeters help these patients and their healthcare providers manage their conditions more effectively by providing real-time data on oxygen saturation levels. This allows for adjustments in treatment plans and the use of supplemental oxygen when necessary.

3. Managing Acute Respiratory Conditions

During acute respiratory conditions, such as pneumonia or during an exacerbation of chronic diseases, monitoring SpO2 can guide treatment decisions. Infections like COVID-19, which directly impact lung function, can be closely monitored using oximeters to ensure that patients receive appropriate and timely care.

4. Use in Anesthesia and Surgery

Oximeters are critical during surgery and anesthesia. They provide continuous monitoring of a patient's oxygen saturation and heart rate, helping anesthesiologists ensure that patients remain stable throughout the procedure. Any drop in oxygen levels can be immediately addressed to prevent hypoxia-related complications.

5. Postoperative Care

After surgery, patients are often monitored using oximeters to ensure they are recovering well and that their oxygen levels remain stable. This is particularly important for patients with a history of respiratory issues or those who have undergone major surgeries.

6. Exercise and Fitness

Athletes and fitness enthusiasts use oximeters to monitor their SpO2 levels during physical activities, ensuring they are training within safe limits.

1.3 HOW OXIMETERS WORKS:

An oximeter works by using the principles of photoplethysmography (PPG) and spectrophotometry to measure blood oxygen levels (SpO2). It emits light from LEDs (usually red and infrared) into the skin, typically at a fingertip or earlobe. Hemoglobin in the blood absorbs different wavelengths of light to varying degrees depending on its oxygenation state.

Oxygenated hemoglobin absorbs more infrared light and allows more red light to pass through, while deoxygenated hemoglobin absorbs more red light and allows more infrared light to pass through. A photodetector on the other side of the skin measures the amount of light that has passed through or reflected back. By comparing the absorption of red and infrared light, the oximeter can calculate the ratio of oxygenated to deoxygenated hemoglobin, thereby determining the blood oxygen saturation level. This non-invasive method provides a quick and continuous way to monitor a person's respiratory function.
Oximeters operate based on the principles of spectrophotometry, which involves measuring light absorption in the blood. Here's a breakdown of the process:

Light Emission and Detection: The device has a light source that emits two wavelengths of light (typically red and infrared) through a translucent part of the body, usually a fingertip or earlobe.

Absorption Measurement: A photodetector on the opposite side of the emitter measures the amount of light that passes through the tissue.

Oxygen Saturation Calculation: Oxygenated and deoxygenated hemoglobin absorb light differently at these wavelengths. By analyzing the changes in light absorption, the device calculates the ratio of oxygenated hemoglobin to total hemoglobin, providing the SpO2 reading.

Pulse Rate: Additionally, the device detects the pulsatile nature of blood flow, allowing it to measure the heart rate concurrently.

Remote Monitoring: With advancements in technology, some oximeters are equipped with wireless connectivity, enabling remote monitoring of patients' respiratory parameters. This is particularly useful for telemedicine, home healthcare, and remote patient monitoring applications.

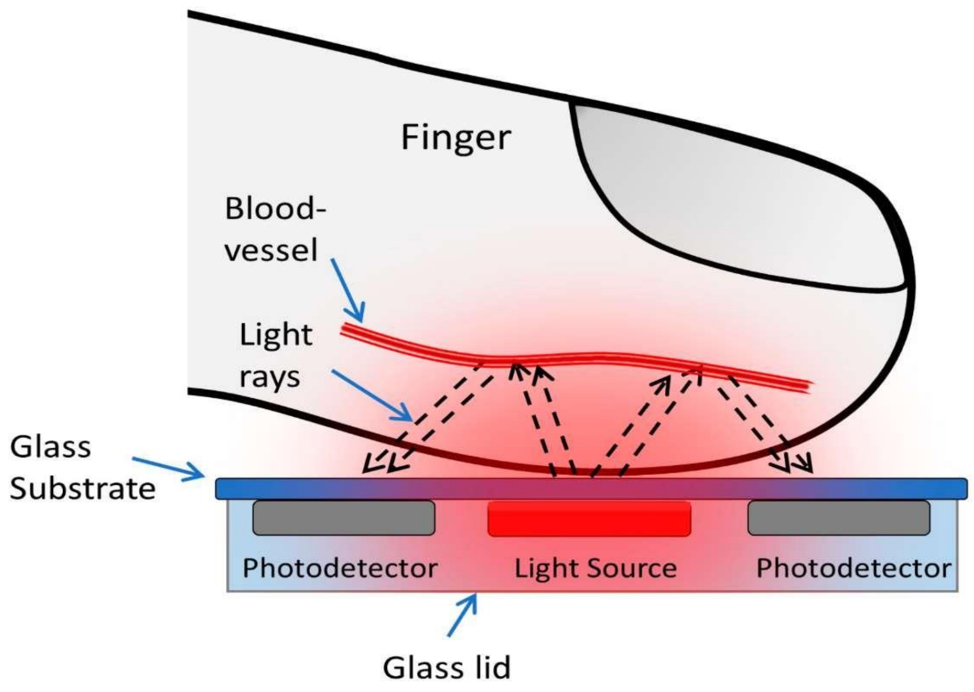


Fig-1.2:**Working**

CHAPTER 2

HARDWARE AND SOFTWARE COMPONENTS

2.1 HARDWARE COMPONENTS

2.1.1 ARDUINO UNO

Arduino Uno is an open-source microcontroller board based on the processor ATmega328P. There are 14 digital I/O pins,6 analog inputs, a USB connection, a power jack, an ICSP header, and a reset button. It contains all the necessary modules needed to support the microcontroller. Just plug it into a computer with a USB cable or power it with an adapter to get started. You can experiment with your Arduino without worrying too much about it. In the event of a worst-case scenario, you could buy a new one as the Uno is very economical compared to other boards like raspberry pi, STM, etc.

THE HARDWARE STRUCTURE OFARDUINO UNO :

The Arduino Uno is built around the ATmega328P microcontroller, which acts as the brain of the board. It features 14 digital input/output pins, of which 6 can be used as PWM outputs, and 6 analog inputs for various sensor integrations. The board operates at a clock speed of 16 MHz, facilitated by a ceramic resonator. It includes a USB connection for programming and serial communication, a power jack for external power supply, and an ICSP (In-Circuit Serial Programming) header for direct programming of the microcontroller. The onboard voltage regulator allows the board to be powered by a USB or an external power source ranging from 7 to 12 volts. Additionally, the Arduino Uno has a built-in reset button for easy restarting of the program. The simplicity of its hardware design, along with extensive pin configurations and power management options, makes it a versatile and popular choice for both beginners and experienced developers in creating various electronic projects.

- **Micro Controller :** It is the central processing unit of Arduino Uno.
- **Digital Pins :** There are 14 digital pins on Arduino Uno which can be connected to components like LED,LCD, etc.
- **Analog Pins :** There are 6 analog pins on the Uno. These pins are generally used to connect sensors because all the sensors generally have analog values. Most of the input components are connected here.

- **Power Supply :** The power supply pins are IOREF, GND, 3.3V, 5V, Vin are used to connection sensors because all the sensors generally have analog values. Most of the input components are connected here.
- **Power Jack :** Uno board can be powered both by external supply and via USB cable.
- **USB Port :** This port function is to program the board or to upload the program. The program can be uploaded to the board with the help of Arduino IDE and USB cable.
- **Reset Button :** This is used to restart the uploaded program.

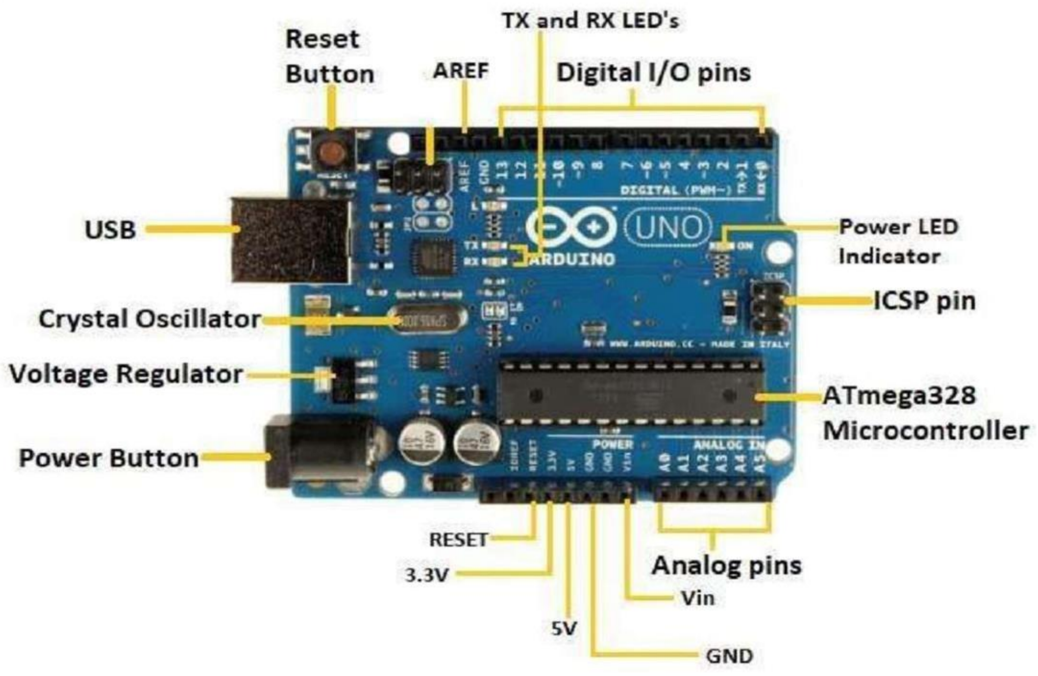


***Fig 2.1*: Arduino Uno**

The Arduino Uno is a versatile and powerful microcontroller board that has become a cornerstone of the maker movement. Its ease of use, extensive community support, and flexibility make it an excellent choice for a wide range of applications, from education and prototyping to robotics and home automation. Whether you are a beginner looking to learn about electronics and programming or an experienced developer working on an innovative project, the Arduino Uno provides a solid foundation for your creative endeavors.

2.1.2 OXIMETERY SENSOR

The MAX30105 is a highly integrated optical sensor module developed by Maxim Integrated, specifically designed for applications in pulse oximetry and heart-rate monitoring. This compact sensor module combines multiple optical components, including

LEDs (Light-Emitting Diodes), photodetectors, and optical elements, within a single package.

At its core, the MAX30105 utilizes a process called photoplethysmography (PPG) to measure blood oxygen saturation (SpO2) and heart rate. This involves emitting light from the integrated LEDs into the skin and detecting the amount of light that is absorbed or reflected by the blood vessels underneath. By analyzing the differences in light absorption at various wavelengths, the MAX30105 can determine the oxygen saturation level of the blood and the heart rate of the individual.

Oximetry sensors are devices used to measure blood oxygen saturation (SpO2) and, often, pulse rate. These sensors are integral to pulse oximeters, which are widely used in medical, fitness, and home health monitoring applications. The non-invasive nature of oximetry sensors, combined with their ability to provide real-time data, makes them essential tools in various health-related fields.

The MAX30105 module offers several key features that make it suitable for a wide range of applications. These include:

Key Features

Non-Invasive: Oximetry sensors measure oxygen levels without the need for blood samples.

Real-Time Monitoring: Provides immediate feedback on blood oxygen saturation and pulse rate.

Compact and Portable: Often small and lightweight, making them suitable for both stationary and mobile applications.

Low Power Consumption: Designed to operate efficiently, which is crucial for wearable and portable devices.

Highly Integrated Design: The MAX30105 integrates multiple optical components into a compact package, simplifying the design and implementation of pulse oximeters and heart-rate monitors.

Flexible Operation Modes: The module supports various operation modes and adjustable parameters, allowing for customization based on specific application requirements.

Real-Time Data Processing: The MAX30105 includes internal signal processing capabilities, which enhance the accuracy of measurements and reduce the need for external processing circuitry.

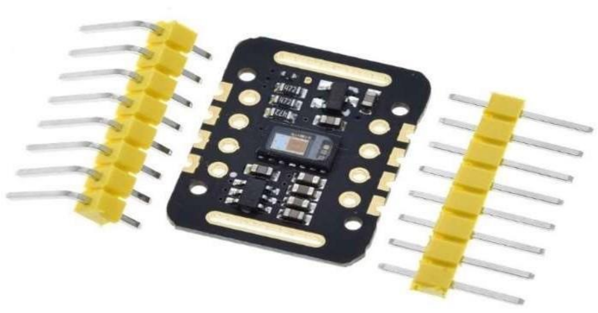


***Fig-2.2*: Oximeter Sensor**

2.1.3 16x2 LCD DISPLAY

A 16x2 LCD (Liquid Crystal Display) is a widely used alphanumeric display module that can show 16 characters per line on two lines. These displays are commonly used in various electronics projects and products due to their simplicity, versatility, and ease of use. They provide a straightforward way to add visual output to microcontroller-based systems, including those built with Arduino, Raspberry Pi, and other development platforms.

Key Features

Character Display: Displays 16 characters on each of the two lines.

Character Resolution: Each character is typically displayed in a 5x8 or 5x10 dot matrix.

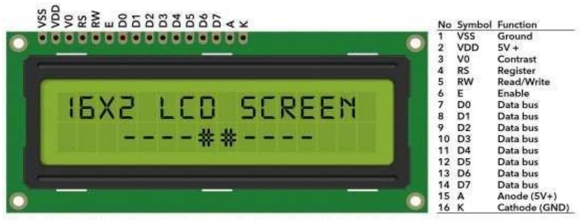
Backlight: Often includes an LED backlight for visibility in low-light conditions.

Controller: Usually driven by the HD44780 or a compatible controller, which handles the display control and communication.

Interface: Commonly uses a parallel interface with 4-bit or 8-bit data lines, along with control lines.

Power and Initialization: The display requires a power supply (usually 5V) and needs to be initialized through a specific sequence of commands to set the display mode, cursor, and other settings.

How a 16x2 LCD Display Work



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

***Fig-2.3*: 16x2 LCD Display**

Data Communication: The microcontroller communicates with the LCD using either a 4-bit or 8-bit parallel interface. In 4-bit mode, data is sent in two 4-bit nibbles.

Control Signals: The RS (Register Select) pin determines whether the data being sent is command (instruction) or character data. The RW (Read/Write) pin selects read or write mode, and the E (Enable) pin triggers the data read/write process.

Character Display: Each character is defined by a dot matrix and the corresponding ASCII value. The display controller translates the data sent from the microcontroller into the characters shown on the screen.

Backlight Control: The backlight can be controlled via a separate power connection, allowing the display to be visible in different lighting conditions.

2.1.4 BREAD BOARD

A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike a perf board or stripboard, breadboards do not require soldering or destruction of tracks and are hence reusable. For this reason, breadboards are also popular with students and in technological education.

A variety of electronic systems may be proto typed by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).Compared to more permanent circuit connection methods, modern breadboards have high parasitic capacitance, relatively high resistance, and less reliable connections, which are subject to jostle and physical degradation.

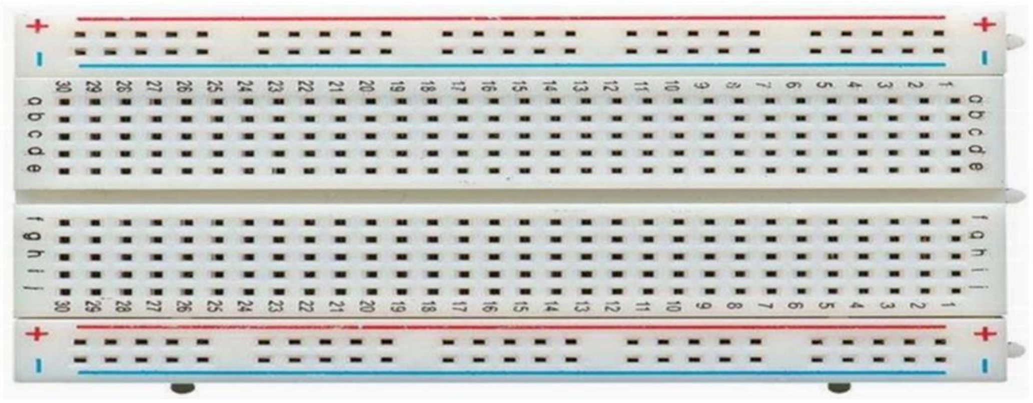


Fig-2.4: **Bread Board**

ADVANTAGES

1. It is adjustable

Every designer should try out a PCB design on a breadboard because it is easy to adjust a breadboard in case you realize that there may be a bug within the system of your design. Debugging a rock solid design may be hard but by using a breadboard, adjustable jumper leads makes it for you to debug your design. You therefore end up spending less since you don't not have to replace the entire expensive electronic equipment.

2. It acts as a shield

A breadboard ensures that your design remains intact by acting as a shield in case it suffers an accident, therefore holding your circuit firmly for it not to get destroyed before reaching its target destination. It is strong enough for it to hold your project in place.

3. It is flexible

A breadboard is flexible in that you can reuse it by playing with connections to create temporary prototypes. The nodes on the breadboards is what makes it easy to use again. This gives you an opportunity to experiment with circuit designs making you feel less limited on only one option since they do not require soldering.

4. Makes it easy for testing

The nodes in a breadboard make it easy for you to test your connection. You can thus easily create circuits by removing and placing wires on different nodes while testing if the components are effective on certain nodes or not. It gives you a chance to rearrange the wires until you are finally satisfied with what you intend to achieve.

DISADVANTAGES

1. **Cost:** High-quality beardboard materials can be expensive, and professional installation can further increase costs.
2. **Maintenance:** Painted beardboard requires periodic repainting or touch-ups, and the grooves can accumulate dust and dirt, necessitating regular cleaning.
3. **Installation Challenges**: Proper wall surface preparation can be time-consuming, and achieving precise alignment is crucial for a neat finish, which might be challenging for novice installers.
4. **Susceptibility to Damage:** Wood beardboard can swell or warp if exposed to high humidity without proper sealing, and it can also be damaged by heavy impacts or sharp objects.
5. **Style Limitations:** While versatile, the distinctive look of beardboard may not suit modern or contemporary interiors.

2.1.5 JUMPERWIRES

Jumper wires are short electrical wires with connector pins at each end, used to create electrical connections between components on a breadboard or other prototyping tool without soldering. They are essential for prototyping and testing circuits in electronics. The theory behind jumper wires revolves around their fundamental role in creating electrical pathways within a circuit. By connecting different points on a breadboard or electronic components on a PCB (Printed Circuit Board), jumper wires facilitate the flow of electrical signals between components, enabling them to communicate and interact effectively.

Types of Jumper Wires

Male-to-Male: Pins at both ends, connects two female headers.
Male-to-Female: Pin at one end, socket at the other, connects male header pins to components.
Female-to-Female: Sockets at both ends, connects male header pins on different modules.

Uses

Prototyping: Quick and easy connections on a breadboard.
Educational Project: Ideal for learning and teaching electronics.

Repairs and Adjustments: Allows frequent changes without permanent connections.

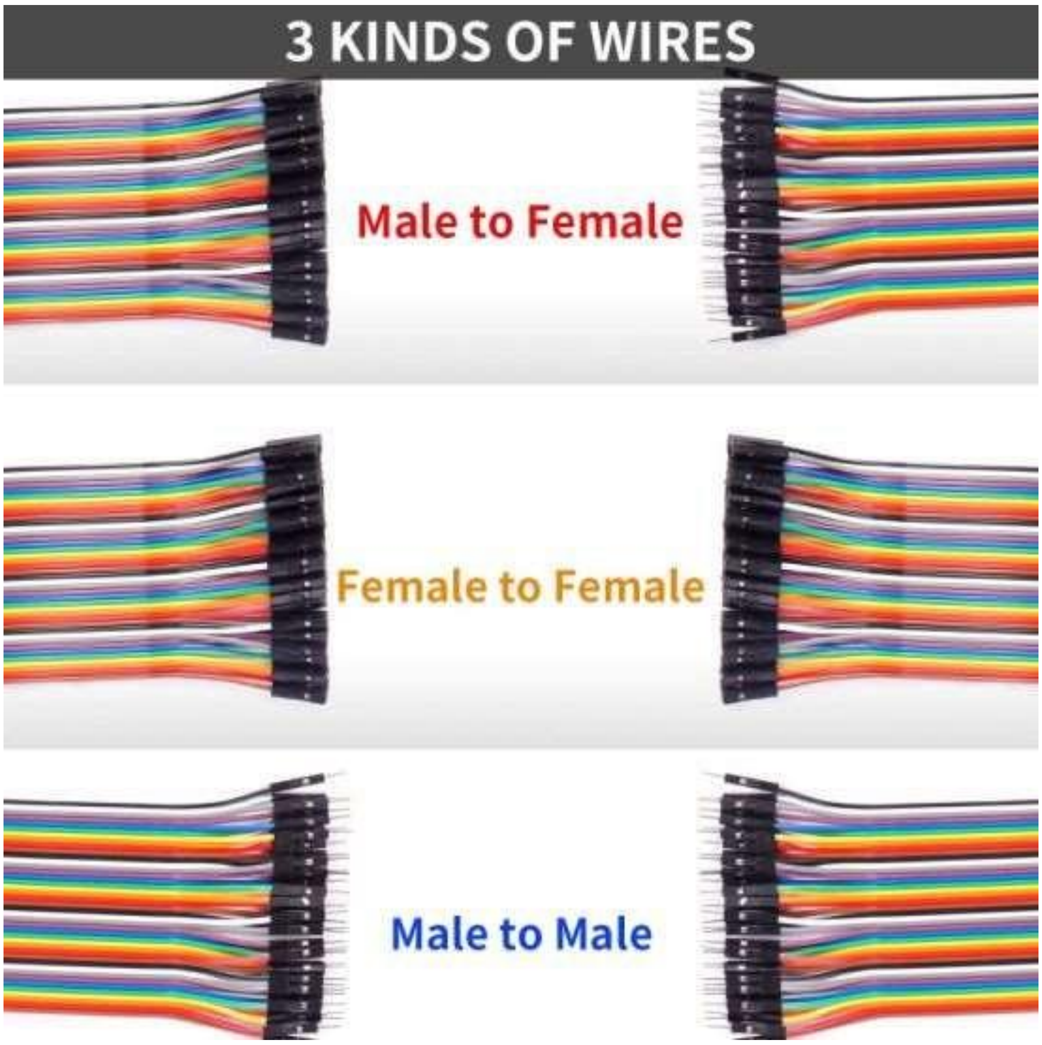


Fig-2.5: **Jumper Wires**

Advantages

Versatile Connectivity: Suitable for both temporary and permanent connections.

Ease of Use: Simple to plug and unplug, no soldering required.

Variety of Types: Available in different configurations, lengths, and colors.

Cost-Effective: Inexpensive, widely available, reusable.

Educational Value: Perfect for learning, essential for educational projects.

Disadvantages

Durability: Connectors can wear out, wires may break with rough handling.

Signal Integrity: Not ideal for high-frequency signals, potential interference.

Space Consumption: Can create a cluttered workspace in complex circuits.

Connection Reliability: Connections can become loose, less reliable than soldering.

Current Limitations: Not suitable for high-current applications, risk of overheating.

2.2 SOFTWARE REQUIREMENTS

2.2.1 SOFTWARE USED :

We used the following software in this project

• ARDUINO IDE SOFTWARE

Arduino IDE :-

The Arduino Integrated Development Environment (IDE) is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Programs written using Arduino Software (IDE) are called sketches. We need to connect the Arduino Uno board with the IDE to upload the sketch written in the Arduino IDE software.
These sketches are written in the text editor and are saved with the extension '.ino'.

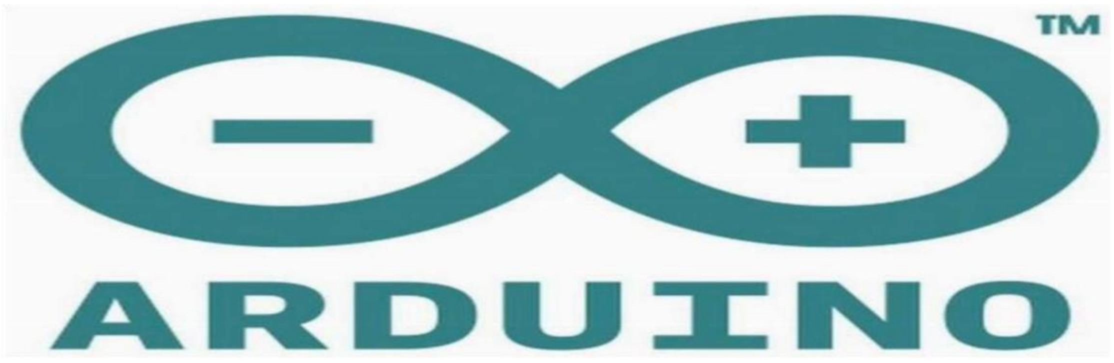


Fig 2.6: **Arduino IDE Logo**

The Arduino IDE simplifies the process of developing and deploying code to Arduino boards, making it accessible for both beginners and experienced developers to create a wide range of electronic projects. The IDE features a simple yet powerful code editor where users can write and edit Arduino sketches (programs). The editor provides syntax highlighting, auto-indentation, and other helpful features to enhance the coding experience. Arduino IDE comes with a built-in library manager that allows users to easily install, manage, and include external libraries. These libraries provide pre-written code for various functions, such as interfacing with sensors, controlling actuators, and communicating with external devices.
The IDE includes a built-in serial monitor tool that enables bidirectional communication between the Arduino board and the computer. This tool is invaluable for debugging code and monitoring sensor data in real-time.

2.2.2 DOWNLOADING ARDUINO IDE SOFTWARE

STEP 1: First up all, open the browser and search Arduino software on the Google search engine. You will find the official software link to download. Or you can directly visit link https://www.arduino.cc.

Click on the software menu option.

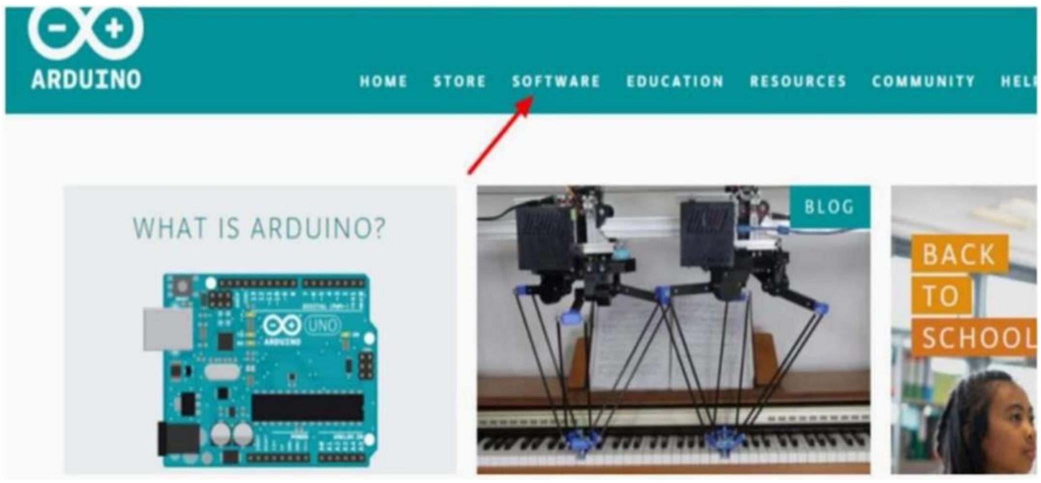
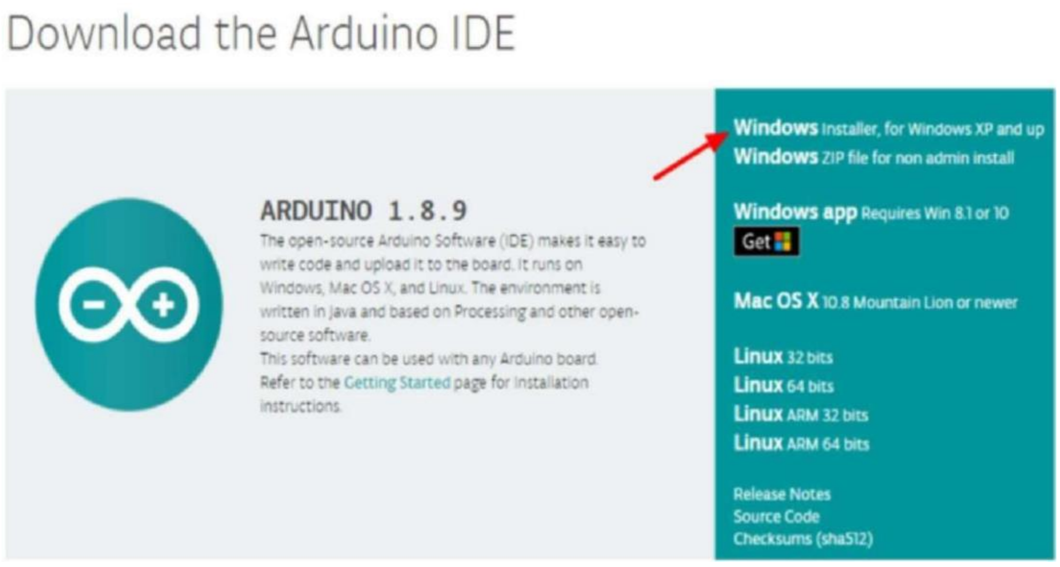


Fig 2.7: **What is Arduino**

STEP 2 : Select the Download submenu option under the software menu. Based on your operating system, choose the installer.



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

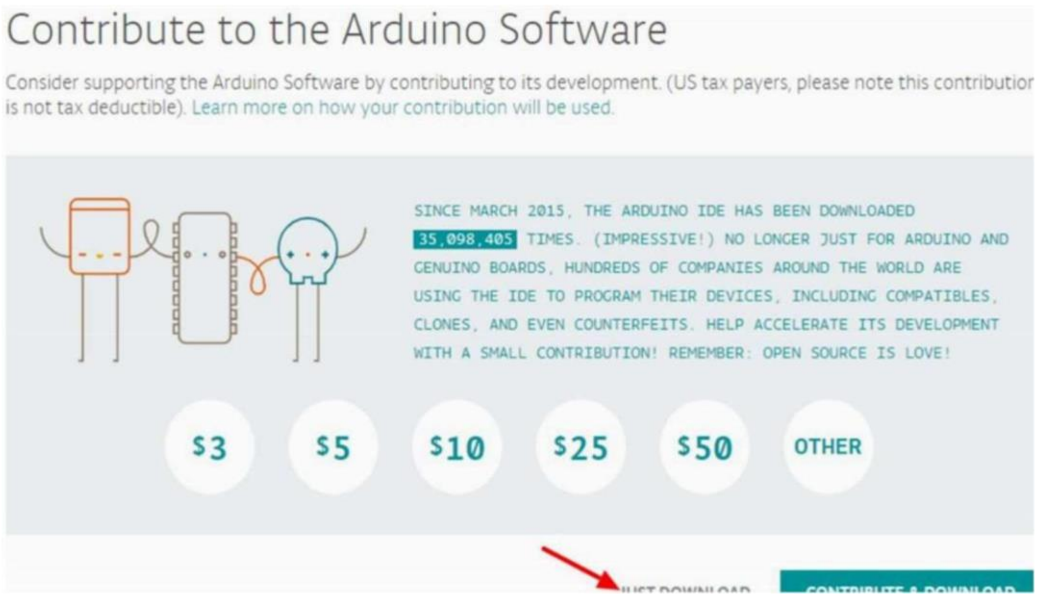
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

***FIG- 2.8*:** **Download Arduino**

This Arduino software is completely FREE. Though it's free, you can donate any amount if you want to contribute and support the Arduino software foundation. If you are making a donation, click on CONTRIBUTE & DOWNLOAD.

If you want to use it for FREE, click on JUST DOWNLOAD option.



Contribute to the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 35,098,405 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

***Fig-2.9*: Contribution to Arduino**

This will start downloading Arduino software on your system. After downloading, you can see the file '.exe' saved in your system.

STEP 3 : To install Arduino, double click on installed software (.exe file). It might take a few minutes to install. Awesome! You are all ready to use Arduino.

2.2.3 LIBRARY INSTALLATION

1.LiquidCrystal_I2C

The LiquidCrystal library in Arduino simplifies the interface with liquid crystal displays (LCDs), providing an easy-to-use way to display text and control the display properties. It offers a range of functions to initialize the display, write text, position the cursor, and control display properties such as backlight brightness and contrast. Using the LiquidCrystal library begins with connecting the LCD to the Arduino board, ensuring the correct wiring of the data pins and control pins. Once the hardware setup is complete, the library functions can be called in the Arduino sketch to interact with the LCD. This includes initializing the display with the begin() function, setting the cursor position with setCurser(), writing characters or strings with print(),and controlling display properties using functions like clear() and noDisplay().

The LiquidCrystal library abstracts away much of the complexity of interfacing with LCDs, allowing developers to focus on the functionality they want to implement rather than the intricacies of the hardware communication protocol. This makes it particularly useful for projects involving user interfaces, data visualization, or status displays where an LCD is employed as the output medium. Whether for educational purposes, hobbyist projects, or practical applications, the LiquidCrystal library enhances the accessibility and usability of LCD displays with Arduino.

2. MAX30105.h

The MAX30105.h library in Arduino facilitates the integration of the MAX30105 sensor module into Arduino projects, streamlining the process of interfacing with the sensor and extracting data. This library abstracts away the complexity of communicating with the sensor by providing a set of functions and methods that allow users to easily initialize the sensor, configure its settings, and retrieve raw or processed data.

Using the MAX30105.h library typically involves including the library header file at the beginning of the Arduino sketch. Once included, users can create an instance of the MAX30105 object, which serves as a handle for interacting with the sensor. Through this object, various functions can be called to perform tasks such as initializing the sensor, configuring its operating parameters (such as sample rate, LED brightness, and pulse width), and reading raw or processed data from the sensor's registers.

The library abstracts the low-level register manipulation required to communicate with the MAX30105 sensor, making it easier for developers to focus on their application logic rather than the intricacies of the sensor's communication protocol. This simplification is particularly valuable for projects involving pulse oximetry, heart rate monitoring, or other biomedical sensing applications where the MAX30105 sensor is employed. By providing a standardized interface, the MAX30105.h library enhances the accessibility and usability of the MAX30105 sensor within the Arduino ecosystem.

Steps to include library:

1. Go to the menu bar, click on "Sketch".
2. In the dropdown menu, select "Include Library".
3. Click on "Manage Libraries…". This will open the Library Manager window.
4. Click on the library you want to install to highlight it.
5. Click the "Install" button.
6. Libraries are installed.

CHAPTER 3
WORKING AND IMPLEMENTATION

3.1 INTRODUCTION

To create an oximeter project with an Arduino, you'll need an Arduino board (such as the Uno, Nano, or Mega), a pulse oximeter sensor module (like the MAX30100 or MAX30102), an optional LED display for visual output, jumper wires, a breadboard, a USB cable for programming, and any necessary resistors. Start by connecting the pulse oximeter sensor to the Arduino: VCC to 5V (or 3.3V, depending on the sensor), GND to GND, SCL to A5, and SDA to A4. If using an LED display, connect its VCC to 3.3V or 5V, GND to GND, SCL to A5, and SDA to A4. Next, open the Arduino IDE and install the required libraries by navigating to Sketch > Include Library > Manage Libraries, then search for and install the libraries specific to your sensor (such as "MAX30105" or "MAX30102") and the LED display libraries.

The Arduino code begins by including the necessary libraries and setting up the LED display and pulse oximeter sensor. In the setup function, initialize serial communication, the pulse oximeter sensor, and the LED display. The loop function continuously updates the sensor readings and displays the heart rate levels on the serial monitor and LED display. The code also includes a callback function to detect heartbeats. With everything connected and programmed, upload the code to the Arduino, and the device will measure and display the heart rate and oxygen saturation, providing a functional oximeter. This project is a practical application of integrating sensors with microcontrollers, offering real-time health monitoring capabilities.

In practice, once everything is connected and the code is uploaded to the Arduino, the device will measure and display heart rate levels. This setup provides real-time health monitoring capabilities. The LED display, if used, offers a visual representation of the data, making the oximeter more user-friendly. The project demonstrates how sensors can be integrated with microcontrollers to create practical health monitoring devices. It also showcases the versatility of the Arduino platform in handling various sensor inputs and providing real-time data processing and display. This kind of project is beneficial for learning about sensor integration, signal processing, and the basics of health monitoring technology.This real-time data is essential for monitoring patients with respiratory or cardiovascular conditions, during surgery, or in high-altitude environments. Its ease of use, quick readings, and portability make the oximeter an invaluable tool in both clinical settings and home healthcare.

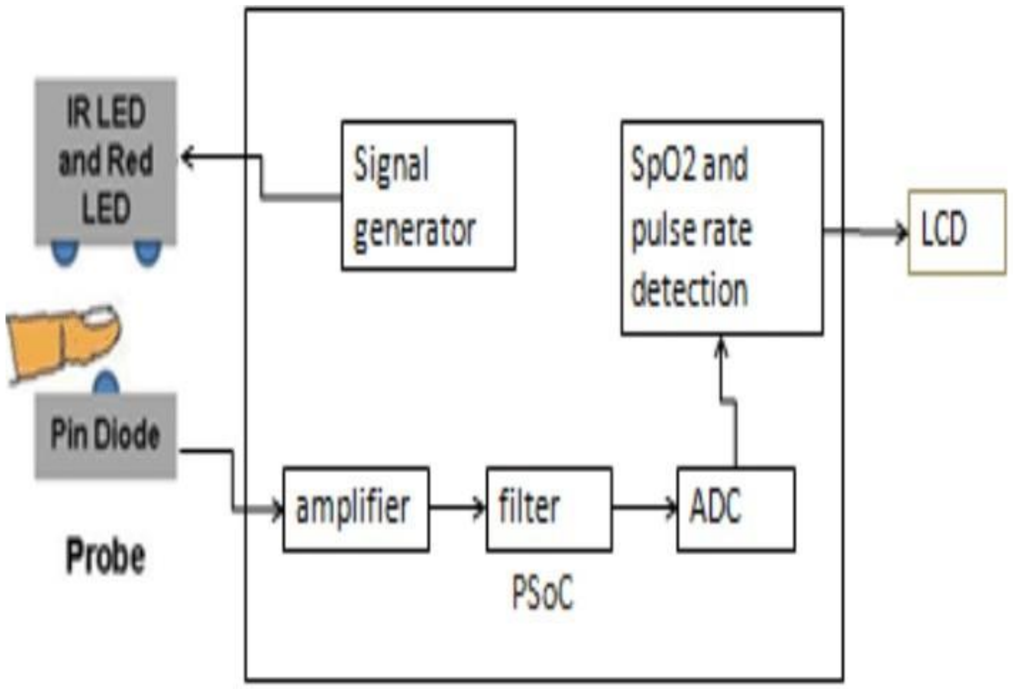


Fig-3.1: **Schematic diagram**

3.2 SYSTEM CIRCUIT DESIGN

Pulse Oximeter Sensor Module:

VCC: Connected to the 5V pin on the Arduino (or 3.3V if specified by the sensor).

GND: Connected to the GND pin on the Arduino.

SCL: Connected to the A5 pin on the Arduino (for I2C communication).

SDA: Connected to the A4 pin on the Arduino (for I2C communication).

Optional LED Display:

VCC: Connected to 3.3V or 5V (depending on the display's voltage requirements).

GND: Connected to the GND pin on the Arduino.

SCL: Connected to the A5 pin on the Arduino.

SDA: Connected to the A4 pin on the Arduino.

Jumper Wires:

Use jumper wires to establish connections between the sensor module, LED display (if used), and the Arduino.

Ensure proper polarity and alignment while making connections to avoid short circuits or incorrect connections.

Breadboard: A breadboard can be used to organize connections and distribute power and ground lines efficiently.

It helps in creating a neat and structured layout for the project, especially when dealing with multiple components and connections.

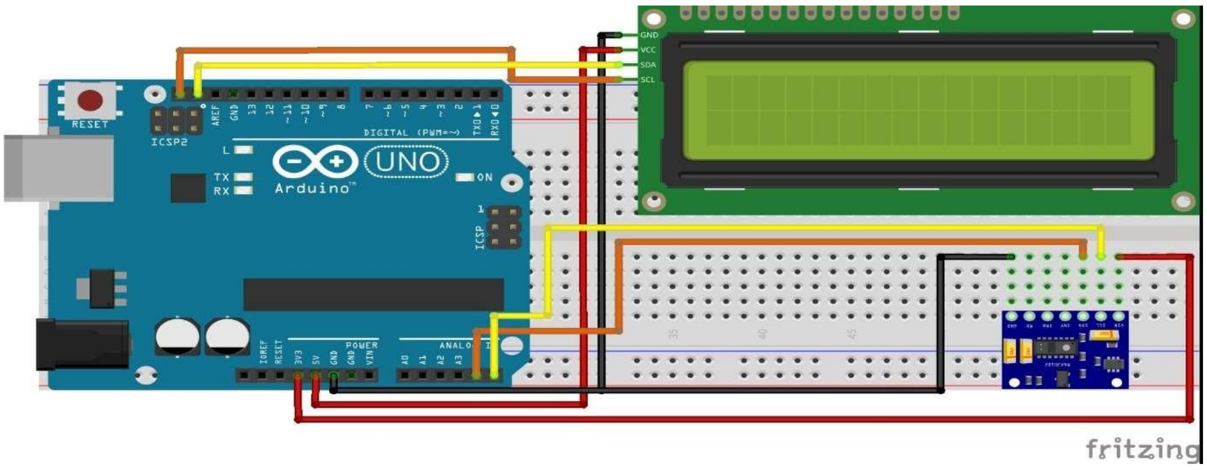


***Fig-3.2*: Circuit Diagram**

3.3 HOW TO CONNECT ARDUINO BOARD WITH LAPTOP

To connect an Arduino board to a laptop, follow these steps:

Using USB Cable:

Select a USB Cable: Ensure you have a USB cable compatible with your Arduino board. Most Arduino boards use a standard USB Type-B connector.

Connect Arduino to Laptop: Plug one end of the USB cable into the USB port on your laptop.

Plug the other end (the USB Type-B connector) into the USB port on your Arduino board.

Power On Arduino: If your Arduino board has an onboard power LED, it should light up once connected to the laptop.

Some boards may have a power switch; ensure it's turned on.

Driver Installation (if needed): In most cases, modern operating systems (such as Windows 10, macOS, or Linux) will automatically detect and install the necessary drivers for the Arduino board.

If drivers are not automatically installed, you may need to download and install them from the Arduino website or the manufacturer's website.

Check Connection: Once connected, you should see the Arduino board listed under the COM (Windows) or /dev (macOS/Linux) ports in the Arduino IDE (under Tools > Port).

If the port is not detected, try restarting the Arduino IDE or your laptop.

3.4 SYSTEM CIRCUIT IMPLEMENTATION

System circuit implementation involves designing and assembling the electronic components to achieve the desired functionality of a device. This process typically starts with defining the system requirements and creating a schematic diagram that outlines how each component is connected. For example, in designing a heart-rate monitoring device using the MAX30105 sensor and an Arduino Uno, the schematic would show the connections between the sensor module, the microcontroller, and any additional components such as resistors, capacitors, and display modules.

Implementing a system circuit for an oximeter involves integrating the pulse oximeter sensor module, Arduino board, and optional components like an LED display.

The Arduino serves as the central control unit, facilitating communication between the sensor and display components. The pulse oximeter sensor module, typically connected via I2C communication, measures vital parameters like heart rate (pulse). It requires power and ground connections to the Arduino for operation. Optionally, an LED display can be added to visually present the measured data.

This display also requires power and ground connections, as well as connections for I2C communication with the Arduino. Implementation includes physical connection of components using jumper wires on a breadboard, installation of necessary libraries in the Arduino IDE, coding to initialize the sensor, read data, and display it on the LED screen (if used), and finally, testing to ensure accurate measurement and display pulse rate.

Through this process, a functional oximeter system circuit is established, capable of monitoring vital signs in real-time.

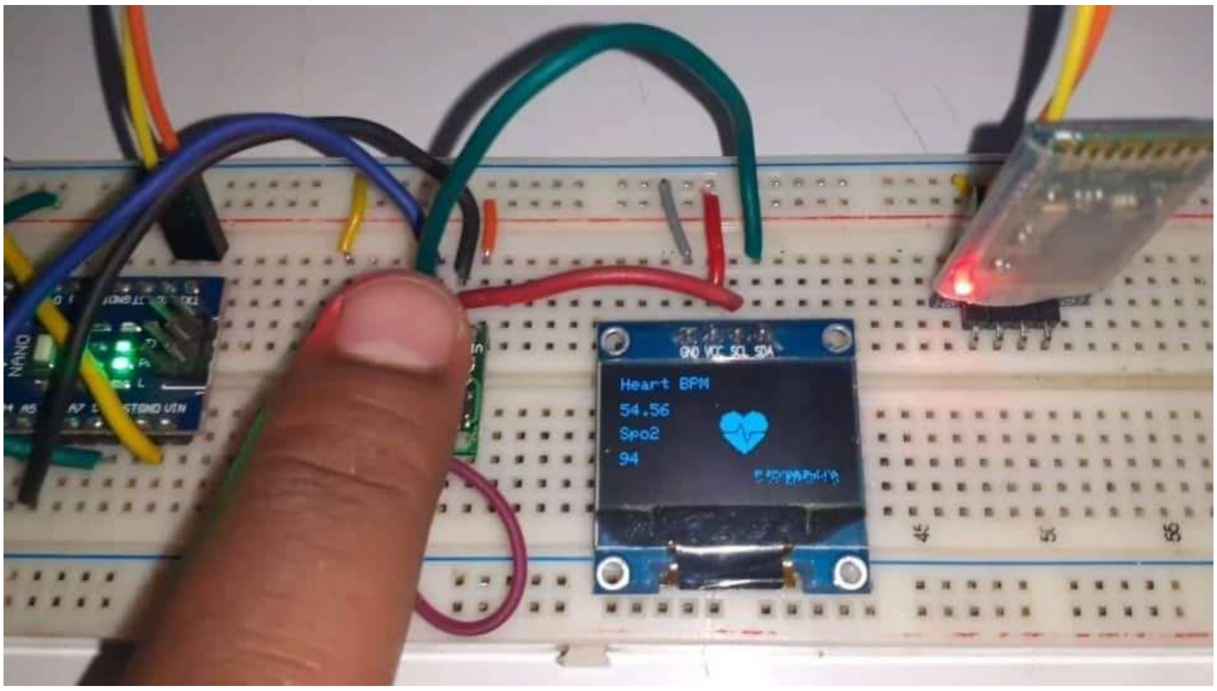


Fig-3.3: **System Circuit Implementation**

3.5 WORKING

PRACTICAL IMPLEMENTATION

1. Making on Arduino board:

Creating an oximeter using an Arduino board involves several steps, from gathering the necessary components to programming the device for accurate readings. Assemble the required components. You'll need an Arduino board, a pulse oximeter sensor module, and optionally, an LED display for visual output. The pulse oximeter sensor module is the core component responsible for measuring blood oxygen saturation (SpO2) and heart rate. It typically comes with connections for power, ground, and I2C communication, which will need to be connected to corresponding pins on the Arduino board. Additionally, if using an LED display, ensure it's compatible with the Arduino and has the necessary connections for power and communication. Creating an oximeter on an Arduino board involves assembling the necessary components, setting up the hardware connections, writing and uploading the Arduino code, and testing the device for accurate readings. With careful implementation and testing, you can develop a functional oximeter capable of monitoring vital signs in real-time.

2. Connecting Oximeter sensor

Connecting an oximeter sensor, such as the MAX30100 or MAX30105, to an Arduino board involves several steps to ensure proper communication and functionality.The oximeter sensor module typically has pins labeled for power (VCC), ground (GND), and data communication (SCL and SDA for I2C communication). Connect the VCC pin of the oximeter sensor module to a 5V pin on the Arduino board. Some modules may operate at 3.3V, so check the module's specifications and connect accordingly.Connect the GND pin of the oximeter sensor module to any ground (GND) pin on the Arduino board. The oximeter sensor module usually communicates with the Arduino using the I2C protocol. This requires two data lines: SCL (clock) and SDA (data).

Connect the SCL pin of the oximeter sensor module to the Arduino's A5 pin. This is the clock line for I2C communication. Connect the SDA pin of the oximeter sensor module to the Arduino's A4 pin. This is the data line for I2C communication. oximeter sensor module should be properly connected to the Arduino board. You can then proceed to write and upload code to the Arduino to initialize the sensor, read data from it, and perform any desired operations, such as displaying the readings on an LED display or transmitting them wirelessly to another device. Testing the sensor readings and ensuring accuracy is crucial before deploying the oximeter for practical use.

3. Connecting LCD Display

Connecting a 16x2 LCD display to an Arduino board involves a straightforward process that ensures proper communication between the two components. To begin, gather the necessary components: an Arduino board, a 16x2 LCD display, a potentiometer for contrast adjustment, jumper wires, and a breadboard for organizing connections. The LCD display typically has pins for power, ground, control signals (such as RS, RW, and EN), data lines (D4 to D7), and backlight control (A for anode and K for cathode). Understanding these pins is crucial for establishing the correct connections.

Start by connecting the power and ground pins of the LCD display to the corresponding 5V and GND pins on the Arduino board, respectively. Next, connect the VO pin (used for contrast adjustment) to the middle pin of the potentiometer. The other two pins of the potentiometer should be connected to 5V and GND for setting the contrast level. Control pins such as RS (Register Select), RW (Read/Write), and EN (Enable) need to be connected to digital pins on the Arduino board, allowing the Arduino to send commands and data to the LCD display.

After connecting the control pins, establish connections for the data lines (D4 to D7), which carry information from the Arduino to the LCD display. These pins should be connected to digital pins on the Arduino, enabling the transmission of characters and commands. Additionally, ensure that the backlight of the LCD display is appropriately controlled by connecting the A (anode) pin to 5V and the K (cathode) pin to GND. With all the connections in place, write and upload code to the Arduino using the Arduino IDE. The code should initialize the LCD display and specify the desired text or information to be displayed. During testing, power on the circuit and adjust the potentiometer if necessary to achieve optimal contrast on the LCD display. By following these steps and testing the display functionality, you can successfully connect a 16x2 LCD display to an Arduino board and display information as desired.

4. Overall working of the circuit

The oximeter circuit, incorporating an Arduino board and a pulse oximeter sensor module, functions as a vital sign monitoring system capable of measuring heart rate (pulse) in real-time. The pulse oximeter sensor module is connected to the Arduino board, typically via I2C communication, allowing the Arduino to receive data from the sensor regarding the oxygen levels and pulse rate of the user. Once the sensor readings are obtained, the Arduino processes this data using programmed algorithms to calculate accurate heart rate values. These values can then be displayed on an optional LED display or transmitted to a computer or mobile device for visualization and analysis. The LED display, if included in the circuit, provides a convenient interface for users to view their vital signs directly. The oximeter circuit's functionality relies on the precise integration of hardware components, including proper connections between the sensor module and the Arduino, as well as the implementation of robust software algorithms for data processing and display. Overall, the oximeter circuit serves as a valuable tool for health monitoring applications, offering users insights into their physiological parameters for better health management.

CHAPTER 4
RESULT AND CONCLUSION

4.1 RESULT

The result obtained on lcd display when finger is placed and not placed on sensor.

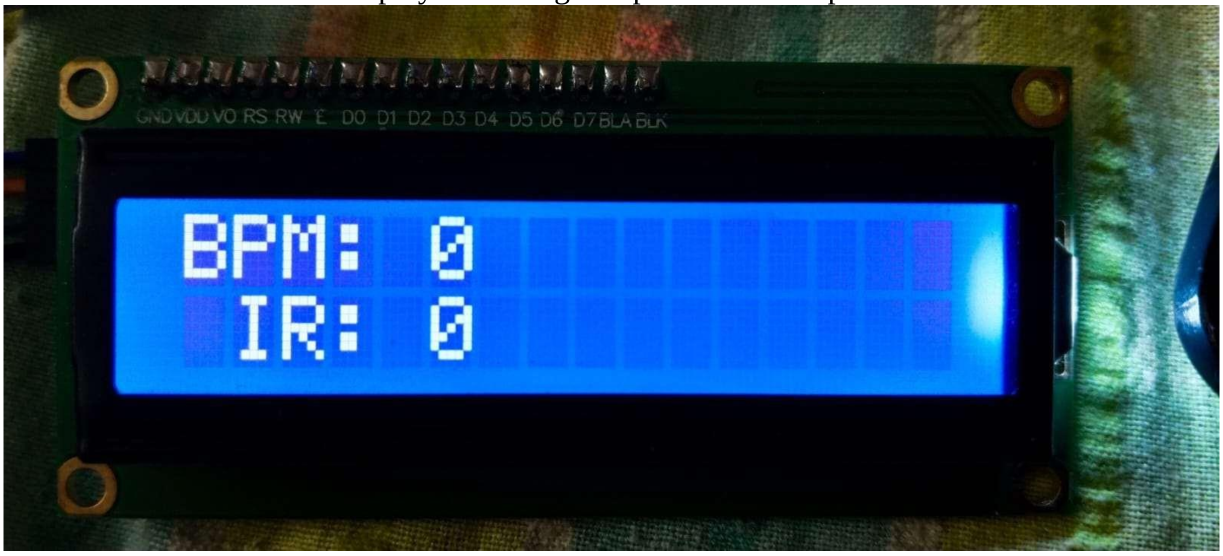


***Fig-4.1*:Result when finger is not placed**

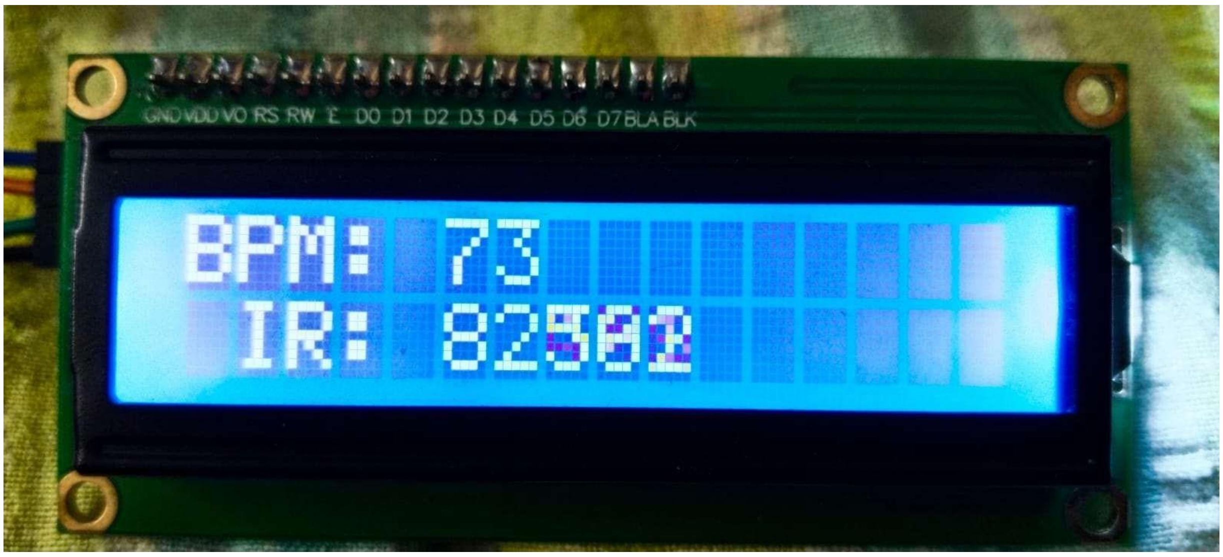


***Fig-4.2*: Output when finger placed**

4.2 CONCLUSION

The project begins with the detailed hardware setup, connecting the MAX30105 sensor to the Arduino Uno via the I2C interface, ensuring accurate power supply and grounding. The software development phase involves programming the Arduino to read raw data from the sensor, process this data to derive meaningful IR values and SpO2 levels, and display the results on an LCD or serial monitor. Extensive calibration and testing phases are critical to ensure the accuracy and reliability of the measurements, taking into account potential sources of error and refining the algorithm as needed.

The final implementation provides real-time monitoring of oxygen saturation and pulse rate, crucial metrics for assessing an individual's respiratory and cardiovascular health. This project showcases the potential for creating affordable, portable, and non-invasive health monitoring devices. It also opens the door for future enhancements, such as wireless data transmission, integration with mobile applications, and the addition of more sensors for comprehensive health tracking.

The development of an oximeter project using an Arduino board presents a versatile and accessible solution for monitoring vital signs such as heart rate. By integrating components like the pulse oximeter sensor module and optional LED display, this project provides a user-friendly interface for real-time health monitoring. Through careful hardware setup and software programming, the oximeter circuit accurately measures and processes physiological data, enabling users to track their health status conveniently. The project's flexibility allows for customization and expansion, with potential enhancements including wireless data transmission, data logging, and integration with mobile or web applications for remote monitoring. Overall, the oximeter project demonstrates the potential of Arduino-based solutions in healthcare, offering an affordable and accessible means of health monitoring for individuals and communities alike. As technology continues to evolve, such projects pave the way for innovative approaches to personal health management and wellness.

In addition to its immediate utility in health monitoring, the oximeter project holds significant educational value, serving as a practical demonstration of sensor integration, data processing, and display techniques using the Arduino platform. It provides an engaging hands-on experience for students, hobbyists, and enthusiasts to learn about electronics, programming, and healthcare technology in a tangible and meaningful way. Furthermore, the open-source nature of Arduino fosters collaboration and knowledge sharing, enabling individuals to contribute to the project's development and improve its functionality over time.

In conclusion, the oximeter project successfully demonstrates the ability to measure and display both infrared (IR) values and oxygen saturation levels (SpO2) using the MAX30105 sensor and an Arduino Uno. The project highlights the integration of hardware and software components, starting from the connection of the sensor module to the microcontroller, followed by coding to process and interpret the sensor data. Through careful calibration and testing, the system accurately reads and outputs IR values, which are crucial for determining the SpO2 levels. The results are displayed in real-time, providing vital health metrics that are essential for monitoring respiratory and cardiovascular conditions. This project not only underscores the practical applications of biomedical sensing technology but also serves as a foundational example for further advancements in wearable health monitoring systems. The successful implementation demonstrates the feasibility and effectiveness of creating low-cost, portable oximeters for both clinical and personal health monitoring.

CHAPTER 5
APPLICATIONS AND FUTURE SCOPE

5.1 FUTURE SCOPE

The future scope for oximeter technology holds immense promise, driven by ongoing advancements in sensor technology, data analytics, and connectivity. As we look ahead, several exciting possibilities emerge for the evolution and expansion of oximeter applications.

Firstly, the miniaturization and integration of oximeter sensors into wearable devices offer the potential for continuous and unobtrusive health monitoring. Wearable oximeters could provide valuable insights into an individual's health status throughout the day, enabling proactive management of chronic conditions and early detection of health issues.

Furthermore, the integration of oximeter data with artificial intelligence (AI) and machine learning algorithms holds the promise of personalized health insights and predictive analytics. By analyzing trends and patterns in oximeter readings over time, AI-driven systems could identify subtle changes in health parameters and provide personalized recommendations for preventive care and lifestyle modifications.

In addition to individual health monitoring, oximeter technology has the potential to revolutionize healthcare delivery in remote and underserved areas. Portable oximeters equipped with wireless connectivity could facilitate telemedicine consultations and remote patient monitoring, enabling healthcare providers to remotely assess patients' vital signs and intervene when necessary.

Moreover, the emergence of Internet of Things (IoT) platforms and cloud-based healthcare solutions opens up new avenues for data aggregation, analysis, and population health management. By aggregating anonymized oximeter data from a large population, healthcare organizations and policymakers can gain insights into community health trends, allocate resources more efficiently, and implement targeted interventions to improve public health outcomes.

As technology continues to advance, the oximeter's role in healthcare is likely to expand beyond traditional clinical settings, becoming an integral part of everyday health management. Whether through wearable devices, AI-driven analytics, or remote monitoring solutions, oximeter technology has the potential to empower individuals to take proactive control of their health and well-being, ushering in a new era of personalized and preventive healthcare.

5.2 APPLICATIONS

Oximeters have a wide range of applications across various industries and settings due to their ability to measure blood oxygen saturation (SpO2) and pulse rate accurately. Some of the key applications of oximeters include:

Medical Monitoring: In hospitals, clinics, and healthcare facilities, oximeters are used for continuous monitoring of patients' vital signs during surgeries, postoperative care, and in critical care units. They provide healthcare professionals with real-time data on pulse rate, allowing for timely intervention in cases of hypoxemia or bradycardia.

Home Healthcare: Oximeters are increasingly being used in home healthcare settings for monitoring chronic conditions such as chronic obstructive pulmonary disease (COPD), asthma, and sleep apnea. Patients with respiratory disorders can use oximeters to track their oxygen levels and pulse rate regularly, helping them manage their condition and detect any deterioration early on.

Sports and Fitness: Athletes and fitness enthusiasts use oximeters to monitor their oxygen saturation levels and pulse rate during exercise and training sessions. This information can help optimize workout intensity, track fitness progress, and prevent overexertion or dehydration.

REFERENCES

[1] https://chatgpt.com/

[2] https://www.arduinolibraries.info/libraries/liquid-crystal

[3] https://github.com/oxullo/Arduino-MAX30100

[4]https://en.wikipedia.org/wiki/Pulse_oximetry#:~:text=A%20pulse%20oximeter%20is%20a%20processed%20into%20other%20measurements.

[5] https://en.wikipedia.org/wiki/Arduino

[6] https://robomaterial.com/wp-content/uploads/2020/08/robomaterial-arduino-uno-r3-2.jpg

[7] https://www.tinytronics.nl/shop/image/cache/data/product-3136/Hartslagsensor-en-Pulsoximeter-module-Heart-Rate-Sensor-and-Pulse-Oximeter-MAX30102-Module-Met-Headers-With-Headers-1000x1000.jpg

[8] https://how2electronics.com/wp-content/uploads/2019/06/img1.jpg

[9]https://robokits.co.in/bmz_cache/b/b777fd6806faba5308f358e1351a8188.image.1066x800.jpg

APPENDEX

ARDUINO CODE:

```
#include <Wire.h>
#include "MAX30105.h"
#include <LiquidCrystal_I2C.h>

#include "heartRate.h"

MAX30105 particleSensor;
LiquidCrystal_I2C lcd(0x27,16,2);

const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;
int beatAvg;

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

  lcd.init();
  Serial.println("Initializing...");
  lcd.backlight();


  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port,
400kHz speed
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady
pressure.");

  particleSensor.setup(); //Configure sensor with default settings
  particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate
sensor is running
  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
}
```

```
void loop()
{
  long irValue = particleSensor.getIR();

  if (checkForBeat(irValue) == true)
  {
    //We sensed a beat!
    long delta = millis() - lastBeat;
    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the
array
      rateSpot %= RATE_SIZE; //Wrap variable

      //Take average of readings
      beatAvg = 0;
      for (byte x = 0 ; x < RATE_SIZE ; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }

  Serial.print("IR=");
  Serial.print(irValue);
  Serial.print(", BPM=");
  Serial.print(beatsPerMinute);
  Serial.print(", Avg BPM=");
  Serial.print(beatAvg);

  if (irValue < 50000)
    Serial.print(" No finger?");

  Serial.println();

  lcd.setCursor(0,0);
  lcd.print("BPM: ");
  lcd.print(beatAvg);

  lcd.setCursor(0,1);
  lcd.print(" IR: ");
  lcd.print(irValue);
}
```