**HEART RATE MONITORING SYSTEM USING**

**FINGERTIP THROUGH IOT**

**Abstract.**

This paper describes real-life applications for getting the heart rate of a person by getting the Beats Per Minute (BPM), ECG (electrocardiogram), and IBI (Inter-Beat Interval) of the person using Arduino. The Heart Rate Pulse Sensor Module, a key sensor, is included in the model. The changes in heart rate are monitored by the heart rate pulse sensor module. It keeps track of the BPM as the beats per minute. The heart rate sensor also depicts the person's ECG and IBI. The interval between heart beats is used to calculate the IBI. The heart pulse sensor modules are secured to the Arduino uno board and are analyzed by the Arduino IDE. The pulse sensor module measures the heart rate and outputs it to the serial plotter and processing development environment, which creates 2D drawings of the heart rate graphics. In order to obtain an accurate 500Hz pulse sensor signal and a high-quality BPM readout, Arduino employs a hardware time interrupt. The serial baud rate is set to

115200 to get a clear graph.

**Keywords:** Arduino, Heartbeat Sensors, Health Monitoring System.

# Introduction

As we can see, nowadays, about a thousand individuals pass away each month as a result of neglecting their health owing to excessive workloads. The Internet of Things is something we have decided to incorporate. We all know that IOT makes life easier, and our initiative will use it to benefit everyone.

Because of its cutting-edge technology, heart rate monitoring systems are one of the significant advancements in the worldwide healthcare program. Due to the high rate of heart attack deaths among hospitalized ill patients, this technological innovation is desperately needed. Heart disease, including heart attacks, coronary disease, congestive heart failure, and congenital heart disease, is the leading cause of death for both men and women globally.

This work provides the potential for continuous heart rate monitoring for persons who live alone and have no one to check on their condition. It was created to provide patients with prompt and appropriate heart health.

# Literature survey

This section contains research that was done by numerous researchers. The authors of [1] used Arduino-based heartbeat and temperature monitoring systems for patients who were treated remotely. However, this literature's drawback was the project's extremely intricate design. The technique in [2] makes use of technology to track bodily indicators like heart rate and send the appropriate data to an expert via SMS. An Arduino Uno, a GSM module, and an IR-based cardiac sensor make up the framework. This equipment will be able to measure heart rate in anyone, from an elderly person to a newborn baby. Following a successful check-in system, the device's convenience will allow for the proper commands. A strategy used to assist medical personnel in working more effectively is presented by the authors in [3]. This study suggests developing a flexible, dependable, and private heart rate tracking and supervision system using sensor networks and Internet of Things technology. The technique in [4] makes it difficult to comprehend the entire monitoring system because it doesn't provide precise information about the temperature sensor that is being used. Since the project's validation procedure and the reliability of heartbeat monitoring are not covered, it is challenging to assess the system's dependability. The system described by the authors in [5] consists of sensors that measure a patient's heart rate and body temperature and is controlled by a central unit. The processed data from these sensors is transmitted via the GSM module to another location, where it is presented on mobile devices. The authors in [6] processed this signal and calculated the heart rhythm in beats per minute (BPM) using the PIC16F87 microcontroller. An SMS alert is sent to the mobile phones of medical staff, patients, or their loved ones. Doctors are able to continuously assess, diagnose, and counsel patients to take earlier precautions as a result. Additionally, the family will be informed so they may quickly see the patient. According to the authors of [7], wearable sensors are used for patient monitoring with modern wireless technologies like Bluetooth and Zigbee due to the advantages of mobility and low system power being consumed. The method in [8] is intended for use at home by patients who are not in severe condition but require ongoing or infrequent clinical or family monitoring. According to the method in [9], a microcontroller-based wireless temperature and cardiac monitoring system that is efficient, reasonably priced, and designed to enable remote patient monitoring is created. An IOT-based system for heart attack and heart rate detection and identification has been created by the authors of [10]. Patients participating in this study will use sensor-equipped equipment and an Android application to evaluate and transmit their heart measurements online.

# Proposed Work

Objectives:

The goal:

* To monitor the patient's heart rate
* Obtain the patient's ECG.
* To predict a patient's BPM.

Coverage was defined as primarily patient customers or customers in need of help track your daily heart rate. This limitation is related to software programming that only measures heart rate. The sensor element made from eco-friendly fabrics and materials.

**SYSTEM HARDWARE:**

All of these elements make up our project:

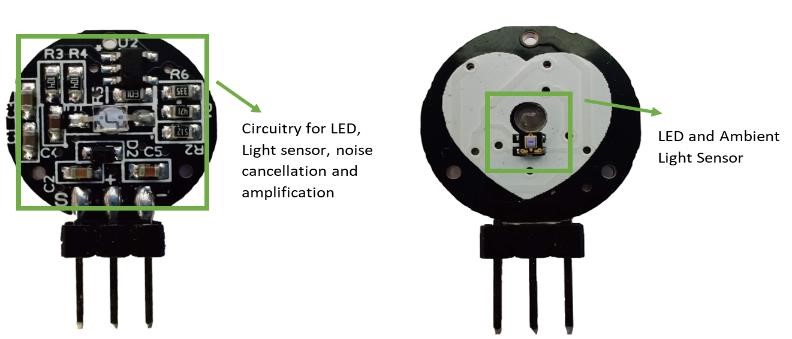
* Pulse sensor
* Bread Board
* Arduino

Pulse Sensor:

The Pulse Sensor is an excellently built, low in power heart-rate sensor that is plugand-play compatible with the Arduino. The sensor easily clips onto a fingertip or earlobe and connects directly to Arduino, which is the nicest part. It also features holes for stitching into the cloth and is quite tiny (button-shaped).

Hardware review:

On the front of the sensor, next to the heart logo, you place your finger. A tiny circular opening is also visible, through which the Kingbright's green LED along with the reversed mount glows. Just below the oblong hole, there is a tiny Avago APDS-9008 natural light picture sensor.



**Fig. 1.** Pulse sensor Pinout

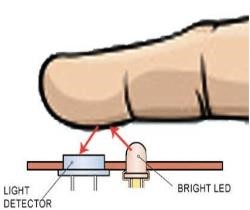
This sensor functions similarly to the ones used in laptops, tablets, and smartphones to adjust the screen's brightness in reaction to ambient light. On the back of the module, a Micro Chip MCP6001 Op-Amp, a couple of resistors, and capacitors make up the R/C filter network. A reverse protection diode is also present to prevent damage in the event that the power lines are accidentally reversed.

The module requires a DC source of the power of around 3.3 and 5V and uses a maximum of 4 mA of electricity.

Pulse sensor working:

Optical heart rate sensors work using simple math. The idea of How an optical heart rate pulse sensor works are easy to understand for anyone who has researched it. As I put the flashlight through your finger, I could see your heart pounding. The pulse sensor operates similarly to other visual heart rate monitors by flashing the green light (550 nm) attached to a finger and measuring the quantity of reflected light with a photosensor.

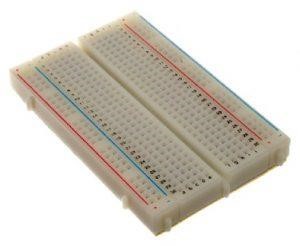
The term "photoplethysmography" refers to this method of light pulse detection. Greenlight can be absorbed by arterial blood's oxygenated hemoglobin. Redness enhances the blood's capacity to absorb green light (elevated hemoglobin content). Heart rate changes the amount of reflected light blood running through a finger causing a waveform on the output of the photosensor. Soon you'll be able to read your heartbeat while keeping the light on get light sensor readings.



**Fig. 2.** Pulse sensor Working

Breadboard:

The word "breadboard" is created by fusing the words "bread" and "board." Previously, this was used to cut the bread into pieces. Around 1970, it was also referred to as a breadboard and used in electronic projects and devices.



**Fig. 3.** Breadboard

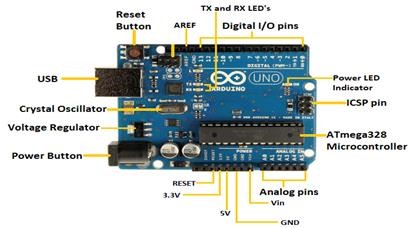
The breadboard is also known as a solder-less board as the component can be attached to the board without the need for additional soldering, permitting it to easily be reused. Although the connection on the breadboard is temporary, connecting the components without soldering is still possible.

Arduino:

Arduino is a basic hardware and software open-source electronics platform. An Arduino board can be used to take inputs like light from a sensor, a user's finger on a button, or tweets. This allows anything to be started, an LED to be turned on, and any information to be published publicly. This can be done using the Wiring-based Arduino Programming Language and the Arduino Software (IDE) built-on Processing.

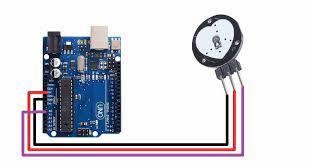
Numerous projects throughout the years, from basic domestic goods to complex scientific apparatus, have employed Arduino as their brain. Due to their efforts, a staggering amount of information is now freely accessible and could be very beneficial to both novices and experts. Arduino was developed at the Ivrea Interaction Design Institute as a straightforward tool for speedy prototyping intended for students with no prior training in electronic gadgets or programming.

The Arduino board started to grow as its user base increased, moving beyond simple 8-bit boards to include products for wearable electronics, 3D printing, embedded systems, and Internet of Things (IoT) applications.



**Fig. 4.** Arduino Uno Pinout

Circuit diagram:



**Fig. 5.** Circuit Diagram for interfacing the Pulse sensor

# CODE FOR THE HEART RATE RUNNING ON THE PROCESSING SOFTWARE:

import processing.serial.\*;

PFont font;

PFont portsFont;

Scrollbar scaleBar;

Serial port;

int Sensor;

int IBI;

int BPM;

int[] RawY;

int[] ScaledY;

int[] rate;

float zoom;

float offset;

color eggshell = color(255, 253, 248);

int heart = 0;

int PulseWindowWidth = 490;

int PulseWindowHeight = 512;

int BPMWindowWidth = 180;

int BPMWindowHeight = 340;

boolean beat = false;

String serialPort;

String[] serialPorts = new String[Serial.list().length];

boolean serialPortFound = false;

Radio[] button = new Radio[Serial.list().length\*2];

int numPorts = serialPorts.length;

boolean refreshPorts = false;

void setup() {

size(700, 600);

frameRate(100);

font = loadFont("Arial-BoldMT-24.vlw");

textFont(font);

textAlign(CENTER);

rectMode(CENTER);

ellipseMode(CENTER);

scaleBar = new Scrollbar (400, 575, 180, 12, 0.5, 1.0);

RawY = new int[PulseWindowWidth];

ScaledY = new int[PulseWindowWidth];

rate = new int [BPMWindowWidth];

zoom = 0.75;

resetDataTraces();

background(0);

drawDataWindows();

drawHeart();

fill(eggshell);

text("Select Your Serial Port",245,30);

listAvailablePorts();

}

void draw() {

if(serialPortFound){

background(0);

noStroke();

drawDataWindows();

drawPulseWaveform();

drawBPMwaveform();

drawHeart();

fill(eggshell);

text("Pulse Sensor Amped Visualizer v1.5",245,30);

text("IBI " + IBI + "mS",600,585);

text(BPM + " BPM",600,200);

text("Pulse Window Scale " + nf(zoom,1,2), 150, 585);

scaleBar.update (mouseX, mouseY);

scaleBar.display();

} else {

autoScanPorts();

if(refreshPorts){

refreshPorts = false

drawDataWindows();

drawHeart();

listAvailablePorts();

}

for(int i=0; i<numPorts+1; i++){

button[i].overRadio(mouseX,mouseY);

button[i].displayRadio();

}

}

}

void drawDataWindows(){

noStroke();

fill(eggshell);

rect(255,height/2,PulseWindowWidth,PulseWindowHeight);

rect(600,385,BPMWindowWidth,BPMWindowHeight);

}

void drawPulseWaveform(){

RawY[RawY.length-1] = (1023 - Sensor) - 212;

zoom = scaleBar.getPos();

offset = map(zoom,0.5,1,150,0);

for (int i = 0; i < RawY.length-1; i++) {

RawY[i] = RawY[i+1];

float dummy = RawY[i] \* zoom + offset;

ScaledY[i] = constrain(int(dummy),44,556);

}

stroke(250,0,0);

noFill();

beginShape();

for (int x = 1; x < ScaledY.length-1; x++) {

vertex(x+10, ScaledY[x]);

}

endShape();

}

void drawBPMwaveform(){

if (beat == true){

beat = false;

for (int i=0; i<rate.length-1; i++){

rate[i] = rate[i+1];

}

BPM = min(BPM,200);

float dummy = map(BPM,0,200,555,215);

rate[rate.length-1] = int(dummy);

}

stroke(250,0,0);

strokeWeight(2);

noFill();

beginShape();

for (int i=0; i < rate.length-1; i++){

vertex(i+510, rate[i]);

}

endShape();

}

void drawHeart(){

fill(250,0,0);

stroke(250,0,0);

heart--;

heart = max(heart,0);

if (heart > 0){

strokeWeight(8);

}

smooth();

bezier(width-100,50, width-20,-20, width,140, width-100,150);

bezier(width-100,50, width-190,-20, width-200,140, width-100,150);

strokeWeight(1);

}

void listAvailablePorts(){

println(Serial.list());

serialPorts = Serial.list();

fill(0);

textFont(font,16);

textAlign(LEFT)

int yPos = 0;

int xPos = 35;

for(int i=serialPorts.length-1; i>=0; i--){

button[i] = new Radio(xPos, 95+(yPos\*20),12,color(180),color(80),color(255),i,button);

text(serialPorts[i],xPos+15, 100+(yPos\*20));

yPos++;

if(yPos > height-30){

yPos = 0; xPos+=200;

}

}

int p = numPorts;

fill(233,0,0);

button[p] = new Radio(35, 95+(yPos\*20),12,color(180),color(80),color(255),p,button);

text("Refresh Serial Ports List",50, 100+(yPos\*20));

textFont(font);

textAlign(CENTER);

}

void autoScanPorts(){

if(Serial.list().length != numPorts){

if(Serial.list().length > numPorts){

println("New Ports Opened!");

int diff = Serial.list().length - numPorts;

serialPorts = expand(serialPorts,diff);

numPorts = Serial.list().length;

}else if(Serial.list().length < numPorts){

println("Some Ports Closed!");

numPorts = Serial.list().length;

}

refreshPorts = true;

return;

}

}

void resetDataTraces(){

for (int i=0; i<rate.length; i++){

rate[i] = 555;

}

for (int i=0; i<RawY.length; i++){

RawY[i] = height/2;

}

}

**BPM USING SERIAL MONITOR**

#define USE\_ARDUINO\_INTERRUPTS true

#include <PulseSensorPlayground.h>

const int PulseWire = 3;

const int LED13 = 13;

int Threshold = 550;

PulseSensorPlayground pulseSensor;

void setup() {

Serial.begin(9600);

if (pulseSensor.begin()) {

Serial.println("PulseSensor object created!");

}

}

void loop() {

int myBPM = pulseSensor.getBeatsPerMinute()

if (pulseSensor.sawStartOfBeat()) {

Serial.println("♥ A HeartBeat Happened ! ");

Serial.print("BPM: ");

Serial.println(myBPM);

}

delay(20);

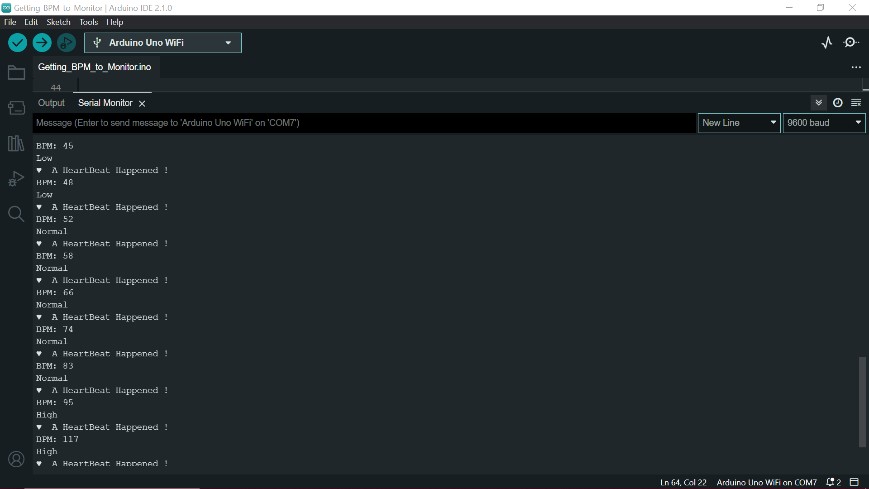
}

# Results

To get the bpm:

Firstly, you have to install the pulse sensor library. This library includes several example sketches. You must load the Getting\_BPM\_to\_Monitor into your IDE for Arduino from those sketches. This sketch computes the heart rate by measuring the interval between pulses and outputs the outcome via a serial monitor.

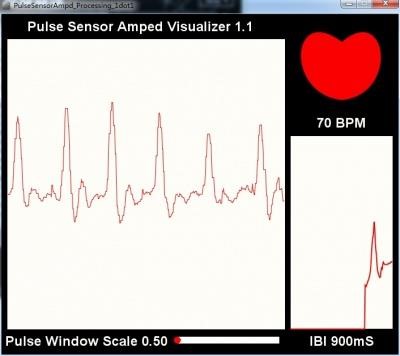
Here in our project, we’ve also depicted if the BPM is low, normal, or high by setting limits up to a certain BPM.



**Fig. 6.** Output of Beats Per Minute (BPM)

To get ECG:

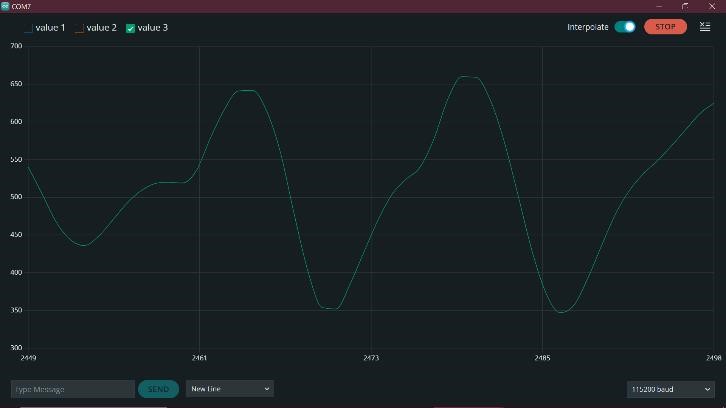
You can access ECG through the PulseSensorAmpd\_Processing\_Visualizer. After choosing the PulseSensorAmpd\_Processing\_Visualizer, You'll get a chance to choose an USB port that your board for Arduino is attached to as soon as the Sketch launches. If your Arduino isn't listed in the ports list, click the Refresh Serial Ports button in order to update the list. you'll start to observe the ECG, once you select the correct port.



**Fig. 7.** Output of Electrocardiogram (ECG)

To get IBI:

The interbeat interval is plotted by using the PulseSensor playground library. All of the information that the Arduino receives from the pulse sensor is shown here by the software. It displays a real-time heart rate graph for the user. It exhibits the IBI over time. The processing sketch reads the serial port and visualizes the IBI.



**Fig. 8.** Output of Interbeat Interval (IBI)

# Conclusion

Many patients die at this moment has become indispensable. The medical team and the patient's family should keep a record of the patient's medical condition data. The device is not bulky and can be manufactured and sold at a low cost. It satisfies the requirements of inexpensive, manufacturable, and efficient devices. Large quantity at low price. We are embedded systems, IoT, and we will design the circuit through this project. Acquired knowledge about embedded systems and the practical use of microcontrollers. This device can be used with Arduino to monitor a person's heartbeat.

In summary, using an Arduino to monitor heartbeats is a convenient and efficient option. Real-time monitoring of vital signs is possible thanks to the reliable platform Arduino provides for the collection and analysis of physiological data and application. This technology includes remote patient monitoring, fitness tracking, and more health care. Combining sensors with Arduino programming enables accurate measurements early detection of anomalies and quick intervention.

Using an Arduino to monitor heartbeats is a reliable and convenient strategy. Arduino's powerful infrastructure enables collection and interpretation Acquire physiological data in real time and enable monitoring of vital signs. This technology provides accurate readings for early diagnosis of abnormalities and is used for Remote patient monitoring, fitness tracking, and healthcare. provide efficient service Improving medical outcomes through monitoring and rapid intervention through the integration of sensors into Arduino programming.

# References

1. 1. Globally advanced engineering research and science. The authors of this book are Vikram Singh R. Parihar, and Rakesh y. tonge, and pooja d. ganorkar. https://www.researchgate.
2. P. A. Pawar, "Heart rate monitoring system using an infrared base sensor and an Arduino Uno," in CSIBIG 2014: An International Conference on Big Data, Proceedings of the 2014 Conference on IT in Business, Industry, and Government, 2014. DOI: 10.1109/CSIBIG.2014.7057005
3. "An IOT-based smart framework for a human heartbeat rate monitoring and control system," Presented at the 6th International Electronic Conference on Sensors and Applications, 15 November 2019–30 November 2019, by Sani Abba and Abubakar Mohammed Garba, https://doi.org/10.3390/ecsa-606543.
4. VG M Pandey and colleagues (2019). Using Arduino and GSM, a heartbeat monitoring and alerting system. DOI:10.32628/CSEIT195230
5. Mohammed Waji Alam, Tanin Sultana, Mohammed Sami Alam, "One Heartbeat" Temperature measurement system for remote health monitoring "Wireless Body Area Networks", International Journal of Bio-Science and BioTechnology Vol.8, No.1 (2016), pp. 171-190,

http://dx.doi.org/10.14257/ijbsbt.2016.8.1.16

1. Heartbeat monitoring alerts according to SMS 2009 IEEE Symposium on Industrial Electronics and Applications 4.–6. Kuala Lumpur, Malaysia, October 2009. Warzarina Matt Jubadi, Siti Faridatul Aisha Mohud Sahak Abt. University of Electronic Engineering, Tun Hussain Ong Malaysia Batu Pahat, Johor, Malaysia, DOI: 10.1109/ISIEA.2009.5356491
2. Harshavardhan B.Patil, Prof.V.M.Umale, “Arduino Based Wireless Biomedical Parameter Monitoring System Using Zigbee”, International Journal of Engineering Trends and Technology (IJETT) – Band 28 Number 7 – Oct.

2015.DOI:10.15/22315381/IJETT-V28P261

1. Manisha Sherar, Jaikaran Singh, Mukesh Tiwari, "Drahtlose Patientgesundheit" “Monitoring Systems”, International Journal of Computer Applications

(0975 – 8887) Band 62 – Nr. January 6, 2013. DOI: 10.5120/10081-4697

1. C.K. Das, M.W. Aram, M. I. Hoque, “Wireless heart rate and temperature monitoring system for remote patients,” ICMERE2013-PI-197. DOI: 10.13140/2.1.2568.9124
2. Giri, Suchitra and Kumar, Ujwar and Sharma, Varsha and Kumar, Satish and Kumari, Shika, Kumar Lawani, Rahul, Pande, Plastia, Bhadra, Beas, an IoTbased heart attack detection and heart rate monitoring system (July 16, 2020). International Conference on Artificial Intelligence, IoT and Smart Cities and

Applications (ICAISC 2020), https://dx.doi.org/10.2139/ssrn.3652736