**BIOMEDICAL INSTRUMENTATION**

**MINI PROJECT**

**JULY 2019-NOV 2019**

**HEART BEAT SENSOR USING ARDUINO**

**DONE BY:**

**SHRUTHI RAJAGOPAL (2017505066)**

**HEART BEAT SENSOR USING ARDUINO**

**AIM:**

To design an Heart Beat sensor by interfacing Pulse sensor and arduino.

**SOFTWARE REQUIRED**

* Arduino IDE

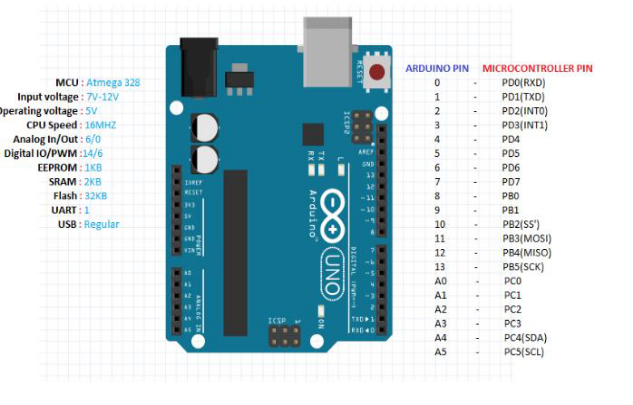
**COMPONENTS REQUIRED**

* Arduino UNO R3
* Cable for Arduino to computer
* Jumpers
* Pulse Sensor

**THEORY**

**ARDUINO UNO R3**

[Arduino](http://arduino.cc/) is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a [microcontroller](http://en.wikipedia.org/wiki/Microcontroller)) and a piece of [software](http://arduino.cc/en/Main/Software), or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.



**POWER (USB / BARREL JACK)**

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled and the barrel jack is labeled. The USB connection is also how you will load code onto your Arduino board.

**SOFTWARE**

A program for Arduino hardware may be written in any [programming language](https://en.wikipedia.org/wiki/Programming_language) with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit [AVR](https://en.wikipedia.org/wiki/Atmel_AVR) and 32-bit [ARM Cortex-M](https://en.wikipedia.org/wiki/ARM_Cortex-M) based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

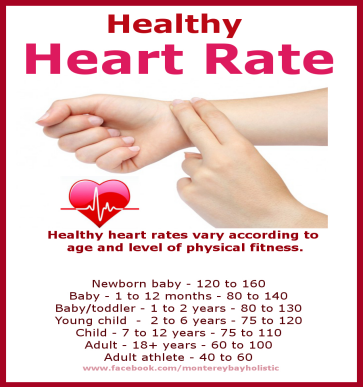
**ARDUINO IDE**

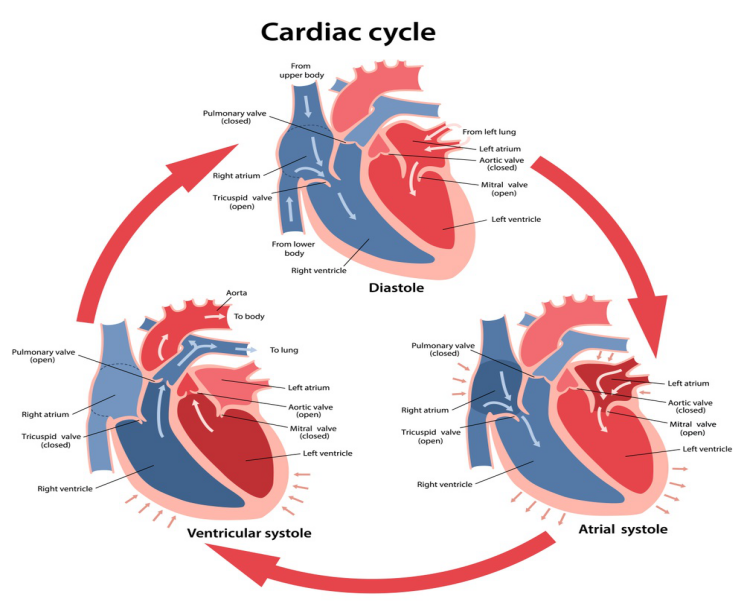
The Arduino [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It originated from the IDE for the languages [Processing](https://en.wikipedia.org/wiki/Processing_(programming_language)) and [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)). It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, [brace matching](https://en.wikipedia.org/wiki/Brace_matching), and [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), and provides simple one-click mechanisms to compile and upload programs to an Arduino board.

The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures.

**What is Heart Rate?**

Heart rate: The number of heartbeats per unit of time, usually per minute. The heart rate is based on the number of contractions of the [ventricles](https://www.medicinenet.com/script/main/art.asp?articlekey=26788) (the lower chambers of the heart). The heart rate may be too fast (tachycardia) or too slow ([bradycardia](https://www.medicinenet.com/script/main/art.asp?articlekey=2515)). The pulse is a bulge of an artery from waves of blood that course through the blood vessels each time the heart beats. The pulse is often taken at the wrist to estimate the heart rate.



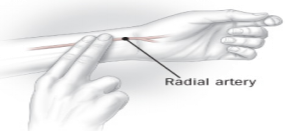


**Simple Ways to measure Heart Rate:**

**Method 1: Radial pulse**

To check your pulse using this method, you’ll be finding the [radial artery](https://www.healthline.com/human-body-maps/radial-artery).

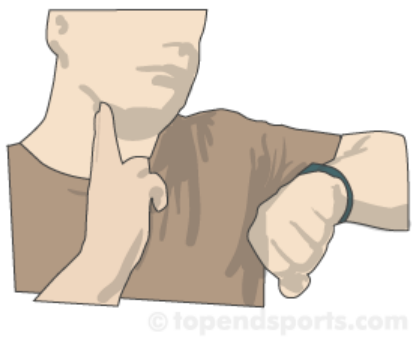
1. Place your pointer and middle fingers on the inside of your opposite wrist just below the thumb.
2. Don’t use your thumb to check your pulse, as the artery in your thumb can make it harder to count accurately.
3. Once you can feel your pulse, count how many beats you feel in 15 seconds.
4. Multiply this number by 4 to get your heart rate. For instance, 20 beats in 15 seconds equals a heart rate of 80 beats per minute (bpm).

****

**Method 2: Carotid pulse**

To check your pulse using this method, you’ll be finding the [carotid artery](https://www.healthline.com/human-body-maps/external-carotid-artery-branche).

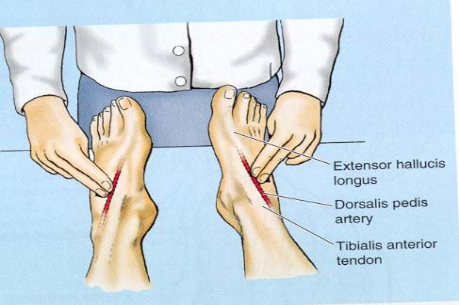
1. Place your pointer and middle fingers on the side of your windpipe just below the jawbone. You may need to shift your fingers until you can easily feel your heart beating.
2. Count the pulses you feel for 15 seconds.
3. Multiply this number by 4 to obtain your heart rate.



**Method 3: Pedal pulse**

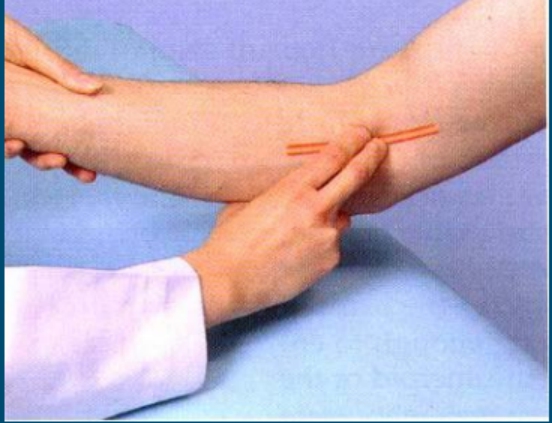
You can also find your pulse on the top of your [foot](https://www.healthline.com/human-body-maps/foot-vessels). This is called the pedal pulse.

1. Place your index and middle fingers above the highest point of the bone that runs along the top of your foot. You may have to move your fingers along the bone or slightly to either side to feel the pulse.
2. Once you have found your pulse, count the beats for 15 seconds.
3. Multiply by 4 to obtain your heart rate.



**Method 4: Brachial pulse**

1. Turn your arm so it’s slightly bent and your inner arm is facing up toward the ceiling.
2. Place your index and middle fingers along the side of your arm between the crook of your elbow on the top and the pointy part of your elbow bone on the bottom. Then move your fingers an inch up your arm. You may have to press quite firmly to feel your pulse.
3. Once you can feel the pulse, count how many beats occur in 15 seconds.
4. Multiply this number by 4 to obtain your heart rate.



**Method 5: Checking your heart rate with an assistive device:**

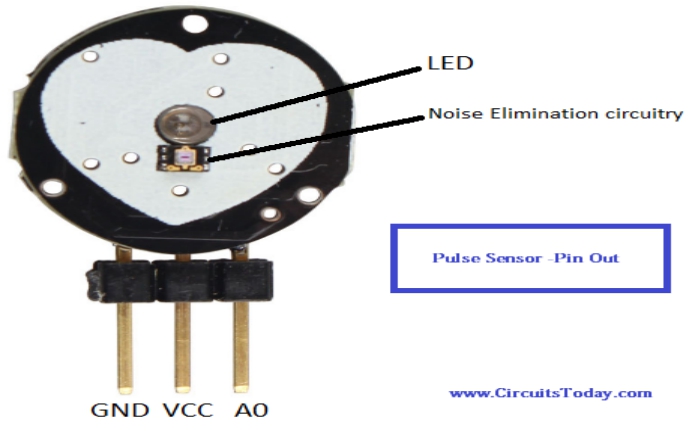
 

****

**Method 6: Using Pulse Sensor:**

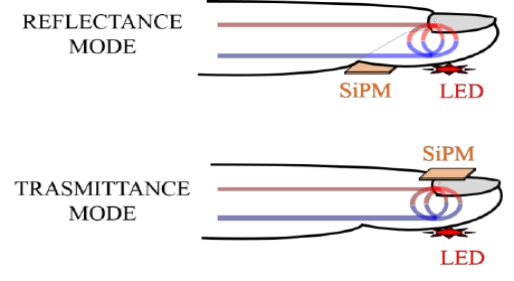
By interfacing the pulse sensor with Arduino UNO we can measure the heart rate in bpm and the measured value will be displayed in Serial Monitor of the Arduino IDE.

**Working – Pulse Sensor**



Pulse Sensor works on the principle of Photoplethysmography(PPG).

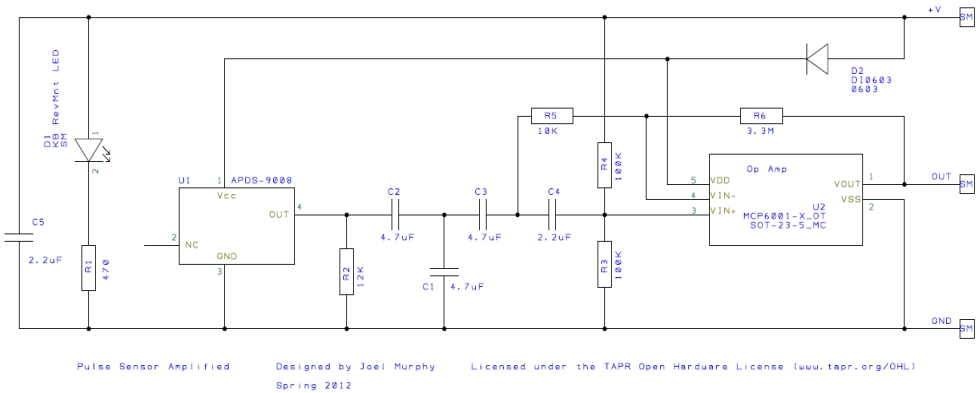
PPG makes uses of low-intensity infrared (IR) light. When light travels through biological tissues it is absorbed by bones, skin pigments and both venous and arterial blood. The voltage signal from PPG is proportional to the quantity of blood flowing through the blood vessels.



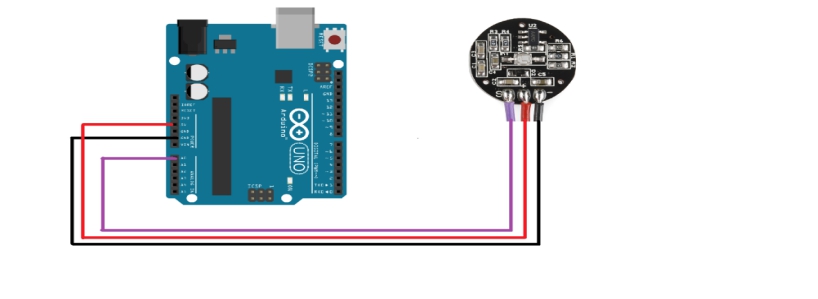
When a heartbeat occurs blood is pumped through the human body and gets squeezed into the capillary tissues. The volume of these capillary tissues increases as a result of the heartbeat. But in between the heartbeats (the time between two consecutive heartbeats,) this volume inside capillary tissues decreases. This change in volume between the heartbeats affects the amount of light that will transmit through these tissues. This change is very small but we can measure it with the help of Arduino.

The pulse sensor module has a light which helps in measuring the pulse rate. When we place the finger on the pulse sensor, the light reflected will change based on the volume of blood inside the capillary blood vessels. During a heartbeat, the volume inside the capillary blood vessels will be high. This affects the reflection of light and the light reflected at the time of a heartbeat will be less compared to that of the time during which there is no heartbeat (during the period of time when there is no heartbeat or the time period in between heartbeats, the volume inside the capillary vessels will be lesser. This will lead higher reflection of light). This variation in light transmission and reflection can be obtained as a pulse from the ouptput of pulse sensor. This pulse can be then conditioned to measure heartbeat and then programmed accordingly to read as heartbeat count.

**Pulse Sensor Schematic**



**CIRCUIT DIAGRAM**:



**ARDUINO CODE**:

int pulsePin = A0; // Pulse Sensor purple wire connected to analog pin A0

int blinkPin = 13; // pin to blink led at each beat

// Volatile Variables, used in the interrupt service routine!

volatile int BPM; // int that holds raw Analog in 0. updated every 2mS

volatile int Signal; // holds the incoming raw data

volatile int IBI = 600; // int that holds the time interval between beats! Must be seeded!

volatile boolean Pulse = false; // "True" when User's live heartbeat is detected. "False" when not a "live beat".

volatile boolean QS = false; // becomes true when Arduoino finds a beat.

static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse

volatile int rate[10]; // array to hold last ten IBI values

volatile unsigned long sampleCounter = 0; // used to determine pulse timing

volatile unsigned long lastBeatTime = 0; // used to find IBI

volatile int P = 512; // used to find peak in pulse wave, seeded

volatile int T = 512; // used to find trough in pulse wave, seeded

volatile int thresh = 525; // used to find instant moment of heart beat, seeded

volatile int amp = 100; // used to hold amplitude of pulse waveform, seeded

volatile boolean firstBeat = true; // used to seed rate array so we startup with reasonable BPM

volatile boolean secondBeat = false; // used to seed rate array so we startup with reasonable BPM

void setup()

{

pinMode(blinkPin,OUTPUT); // pin that will blink to your heartbeat!

Serial.begin(115200); // we agree to talk fast!

interruptSetup(); // sets up to read Pulse Sensor signal every 2mS

// IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD VOLTAGE,

// UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN

// analogReference(EXTERNAL);

}

// Where the Magic Happens

void loop()

{

serialOutput();

if (QS == true) // A Heartbeat Was Found

{

// BPM and IBI have been Determined

// Quantified Self "QS" true when arduino finds a heartbeat

serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.

QS = false; // reset the Quantified Self flag for next time

}

delay(20); // take a break

}

void interruptSetup()

{

// Initializes Timer2 to throw an interrupt every 2mS.

TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE

TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER

OCR2A = 0X7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE

TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A

sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED

}

void serialOutput()

{ // Decide How To Output Serial.

if (serialVisual == true)

{

arduinoSerialMonitorVisual('-', Signal); // goes to function that makes Serial Monitor Visualizer

}

else

{

sendDataToSerial('S', Signal); // goes to sendDataToSerial function

}

}

void serialOutputWhenBeatHappens()

{

if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work

{

Serial.print(" Heart-Beat Found "); //ASCII Art Madness

Serial.print("BPM: ");

Serial.println(BPM);

}

else

{

sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix

sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix

}

}

void arduinoSerialMonitorVisual(char symbol, int data )

{

const int sensorMin = 0; // sensor minimum, discovered through experiment

const int sensorMax = 1024; // sensor maximum, discovered through experiment

int sensorReading = data; // map the sensor range to a range of 12 options:

int range = map(sensorReading, sensorMin, sensorMax, 0, 11);

// do something different depending on the

// range value:

}

void sendDataToSerial(char symbol, int data )

{

Serial.print(symbol);

Serial.println(data);

}

ISR(TIMER2\_COMPA\_vect) //triggered when Timer2 counts to 124

{

cli(); // disable interrupts while we do this

Signal = analogRead(pulsePin); // read the Pulse Sensor

sampleCounter += 2; // keep track of the time in mS with this variable

int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise

// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)\*3) // avoid dichrotic noise by waiting 3/5 of last IBI

{

if (Signal < T) // T is the trough

{

T = Signal; // keep track of lowest point in pulse wave

}

}

if(Signal > thresh && Signal > P)

{ // thresh condition helps avoid noise

P = Signal; // P is the peak

} // keep track of highest point in pulse wave

// NOW IT'S TIME TO LOOK FOR THE HEART BEAT

// signal surges up in value every time there is a pulse

if (N > 250)

{ // avoid high frequency noise

if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)\*3) )

{

Pulse = true; // set the Pulse flag when we think there is a pulse

digitalWrite(blinkPin,HIGH); // turn on pin 13 LED

IBI = sampleCounter - lastBeatTime; // measure time between beats in mS

lastBeatTime = sampleCounter; // keep track of time for next pulse

if(secondBeat)

{ // if this is the second beat, if secondBeat == TRUE

secondBeat = false; // clear secondBeat flag

for(int i=0; i<=9; i++) // seed the running total to get a realisitic BPM at startup

{

rate[i] = IBI;

}

}

if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE

{

firstBeat = false; // clear firstBeat flag

secondBeat = true; // set the second beat flag

sei(); // enable interrupts again

return; // IBI value is unreliable so discard it

}

// keep a running total of the last 10 IBI values

word runningTotal = 0; // clear the runningTotal variable

for(int i=0; i<=8; i++)

{ // shift data in the rate array

rate[i] = rate[i+1]; // and drop the oldest IBI value

runningTotal += rate[i]; // add up the 9 oldest IBI values

}

rate[9] = IBI; // add the latest IBI to the rate array

runningTotal += rate[9]; // add the latest IBI to runningTotal

runningTotal /= 10; // average the last 10 IBI values

BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!

QS = true; // set Quantified Self flag

// QS FLAG IS NOT CLEARED INSIDE THIS ISR

}

}

if (Signal < thresh && Pulse == true)

{ // when the values are going down, the beat is over

digitalWrite(blinkPin,LOW); // turn off pin 13 LED

Pulse = false; // reset the Pulse flag so we can do it again

amp = P - T; // get amplitude of the pulse wave

thresh = amp/2 + T; // set thresh at 50% of the amplitude

P = thresh; // reset these for next time

T = thresh;

}

if (N > 2500)

{ // if 2.5 seconds go by without a beat

thresh = 512; // set thresh default

P = 512; // set P default

T = 512; // set T default

lastBeatTime = sampleCounter; // bring the lastBeatTime up to date

firstBeat = true; // set these to avoid noise

secondBeat = false; // when we get the heartbeat back

}

sei(); // enable interrupts when youre done!

}// end isr

**RESULT:**

Thus the Heart Beat Per Minute was implemented using ARDUINO by interfacing Pulse sensor.