**Code:**

#define USE\_ARDUINO\_INTERRUPTS true // Set-up low-level interrupts for most acurate BPM math.

#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.

#include<LiquidCrystal.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

// Variables

const int PulseWire = 0; // PulseSensor PURPLE WIRE connected to ANALOG PIN 0

const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.

int Threshold = 550; // Determine which Signal to "count as a beat" and which to ignore.

// Use the "Gettting Started Project" to fine-tune Threshold Value beyond default setting.

// Otherwise leave the default "550" value.

PulseSensorPlayground pulseSensor; // Creates an instance of the PulseSensorPlayground object called "pulseSensor"

void setup() {

Serial.begin(9600); // For Serial Monitor

lcd.begin(16,2);

// Configure the PulseSensor object, by assigning our variables to it.

pulseSensor.analogInput(PulseWire);

pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino's LED with heartbeat.

pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began" seeing a signal.

if (pulseSensor.begin()) {

Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino power-up, or on Arduino reset.

lcd.setCursor(0,0);

lcd.print(" Heart Rate Monitor");

}

}

void loop() {

int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that returns BPM as an "int".

// "myBPM" hold this BPM value now.

if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat happened".

Serial.println("♥ A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat happened".

Serial.print("BPM: "); // Print phrase "BPM: "

Serial.println(myBPM); // Print the value inside of myBPM.

lcd.setCursor(0,2);

lcd.print("HeartBeat Happened !"); // If test is "true", print a message "a heartbeat happened".

lcd.setCursor(5,3);

lcd.print("BPM: "); // Print phrase "BPM: "

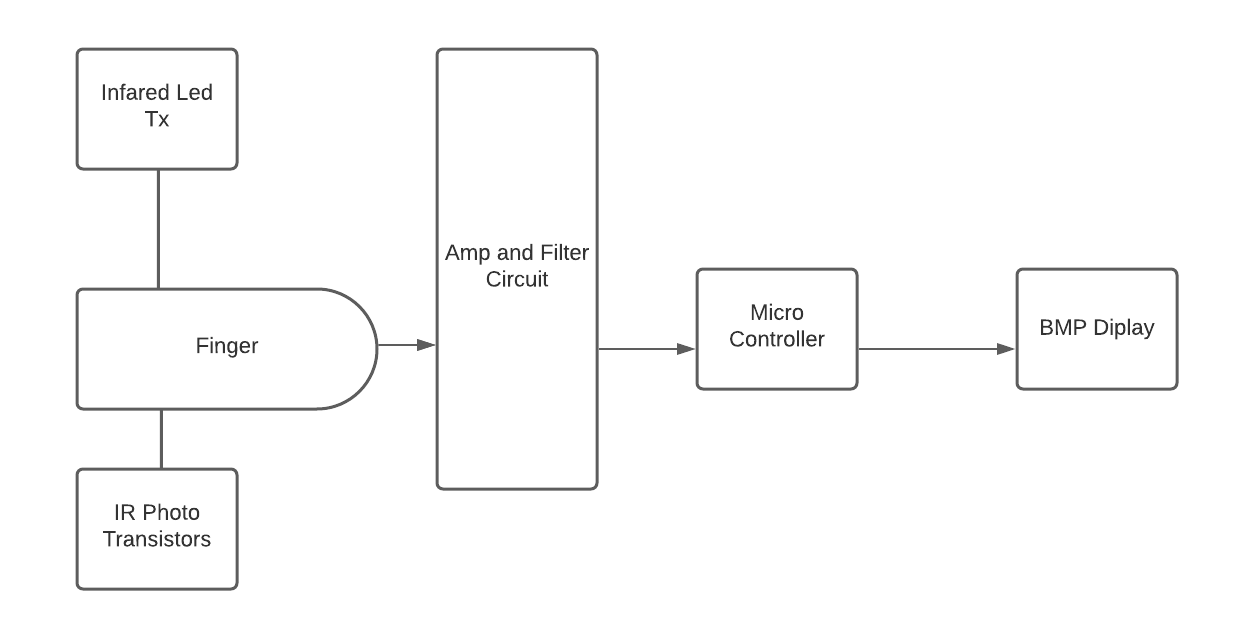
lcd.print(myBPM);

}

delay(20); // considered best practice in a simple sketch.

}

**Working of Pulse Sensor:**



The Principle of pulse sensor is very simple. The sensors which we are using have two surfaces, the first one is Infrared Led TX and the second one is IR Photo Transistors. The Led is located above the vein of a person like on fingertip. It is necessary to locate the Led on the top of the layer of human fingertip directly. When the Led is placed rightly on the vein then the Led will starts to emitting light. When the heart is start pumping, then there will be the flow of blood in the veins. Now for instance, if we know the flow of the blood then we will easily find the heart rates also.

If the flow of blood is sensed then the Amp Filter Circuit will receive the Led light that is produced by the blood flow. So, the small change that is within the light is now controlled by the micro controller and the result of our pulse is shown by the BPM Display.

**Testing of Heartbeat Sensor:**

**Test case:**

When I first put my finger on the sensor to check the pulse rate / heartbeat unfortunately, there was a big problem. The LED on the board blinking like crazy (four or five time faster than my real heartbeat). Processing app screen was not stable too. It always going to 200 BPM in 2 seconds and remain 200 BPM. I was not pushed my finger too soft or too hard. Also, when I tried to change BPM = min (BPM, 200): in the Processing to BPM = min (BPM, 300), heart rate goes up to 300 BPM out of the borders of the Processing screen in seconds.

I think our Arduino code is the reason. I tried to change delay in Arduino code, and I tried both 32 and 64 bit versions of Processing but still problem occurs.

I did some fine tuning on threshold like “=<650”. It was “<250” in original code.

**Solution**

There was surprisingly little code error because the sensor appears to have some analog filtering built in. The analog signal it produces is not the heart beat itself, it is simply a voltage which we see on a time line, which rises and falls according to the heartbeat. The code we have, which apparently analyses this voltage and determines a numerical heartbeat in beats per minute appears at first glance to be overly complex. In principle, we have to smooth the incoming signal and count the transitions between 'below thresholds' and 'above threshold' and divide by 2 to get the number of beats. This count divided by the sample period is beats per unit time, which we can scale up to beats per minute.

The pulse sensor module has a light that 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. This variation in light transmission and reflection can be obtained as a pulse from the output of the pulse sensor. This pulse can be then conditioned to measure heartbeat and then programmed accordingly to read as heartbeat count using Arduino.

**Testing**

In order to check the pulse rate by Arduino we develop a system / device which sense our pulse rate. As someone want to know his heartbeat then can put his finger on the sensor. The sensor detects your pulse rate and show on the LED with the help of signals coming from Microcontroller.

Each heartbeat will change the amount of blood in the finger and the light from the IR LED passing over the finger and thus detected by the Photo Diode will also differ.

Functioning of this development / project is pretty easy but a slight intention for conniving heart rate is essential. There are diverse procedures for scheming heart rate, but here we have read only ten pulses. Now there we are going to calculate total heart beat in a minute by relating the below **method**:

Ten\_pusle\_time=time\_2-time\_1;

1\_pulse\_time= Ten\_pusle\_time /10;

Rate=60000 / 1\_pulse\_time;

Where time\_1 is first pulse hostage value

Time\_2 is list pulse hostage value

Rate is final heart rate.

When first pulse comes, we start to note the time hostage functioning in arduino that is Millis (milliseconds) and take very first pulse hostage value form milliseconds Then we wait for 10 pulses. After writing down 10 pulses we again take hostage assessment in time2 and then we detract time\_1 from time\_2 to take original time taken by ten pulses. And then divide this time by 10 times for getting results for one pulse.

**Test case**

Ten\_pulse\_time1 = 7.29 sec

Next\_ten\_pulse\_time2 = 14.45 sec //combine with previous i.e. time1+time2

Final\_ten\_pulse\_time = time2-time1

Final\_ten\_pulse\_time = 14.45-7.29

= 7.16 sec

1\_pulse\_time = Final\_ten\_pulse\_time / 10

1\_pulse\_time = 7.16/10

= 0.716 sec

Pulse\_rate\_per\_minute = 60/0.716

Pulse\_rate\_per\_minute = 83.80 m-1

For adults 18 and older a normal resting heart rate is between 60 and 100 beats per minutes (bpm), depends upon the person’s physical condition and age. For children age 6 to 15 the normal resting heart rate is between 70 and 100 bpm.

**Conclusion:**

The Pulse sensor is an attachment and play pulse sensor for Arduino. It very well may be utilized by understudies, specialist, competition, creators, and game and portable engineers who need to effectively fuse live pulse information into their ventures. The embodiment is an incorporated optical enhancing circuit and commotion killing circuit sensor. Cut the pulse sensor to your ear cartilage or fingertip. Then, at that point, it into your Arduino, you are currently prepared to peruse pulse.