ABSTRACT

With advancements in Technology and people being more conscious about their health than ever, technical advancements in the field of medicine are inevitable. People are switching towards wearable devices to keep track of their health and fitness related data.This project looks into the development of a heart rate monitor to keep track of the user’s heart rate. The report presents the development of a heart rate monitor. The heart rate monitor developed uses the photoplethysmography (PPG) principle to calculate the heart beats per minute

.

The PPG module developed is then interfaced with an Arduino Uno board responsible for calculating the beats per minute. It then transfers the serial data to a serial monitor. The results show that the heart rate is successfully displayed on the serial monitor and could be helpful in emergency or monitoring situations.

**Table of contents**

**Abstract**

**1 : INTRODUCTION**  **2 : OBJECTIVE**

**3: REQUIREMENTS**

**3.1 HARDWARE:**

**3.1.1 ARDUINO UNO**

**3.1.2 PULSE SENSOR**

**3.2 SOFTWARE:**

**3.2.1 ARDUINO IDE**

**4: CIRCUIT CONFIGURATION**

**5: SIMULATION**   **6: RESULTS**

**7:APPLICATION**

**8: FUTURE SCOPE**

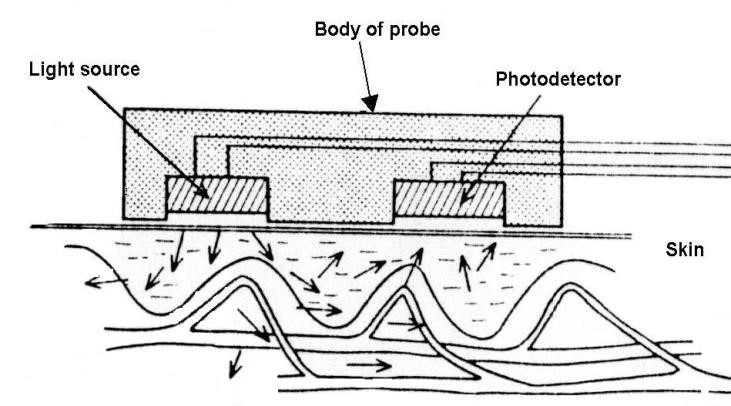
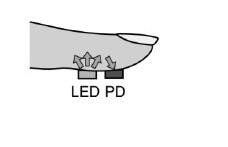
**9: DRAWBACKS**

# INTRODUCTION

Photoplethysmography (PPG) is a low cost optical technique that is able to detect volumetric changes in blood flowing through capillaries from the skin surface. Photoplethysmography was developed in the late 1800s where scientists observed real-time blood flow using light bulbs. It was in the late 1930s that the term ‘photoplethysmography’ was coined by scientists. With advancement in technology, PPG developments now focus on consumer applications using wearable devices. These wearable devices are usually connected to a peripheral device to interpret the results obtained. Today, those peripheral devices have been replaced by smartphones to deliver data to consumers in a user friendly manner.

**Photoplethysmography:**

Photoplethysmography is a derivative of plethysmography technique, where in a simple optical setup can help detect volume changes in peripheral blood circulation. John Allen describes this technique as a non-invasive method since it makes measurements at the skin surface. The technology uses optoelectronic components, such as a red or near infrared light source, to illuminate skin and a photodetector to observe the variations in light intensity within the observed area. Usually, a red or near infrared source of light is used to illuminate skin. This light then travels through tissues and is absorbed by pigments, bones, and blood. The PPG sensors optically observe changes in blood flow volume by detecting changes in light intensity.



**Fig 1:** **PPG placement.LED and photodiode placement for transmission and**

**reflectance based photoplethysmography**

## OBJECTIVE

The objectives of the project are:

1. To design and fabricate patient monitoring system for monitoring Hearth rate signal.
2. Analysis of heartbeat signals for increasing accuracy of BPM.
3. Using built in serial monitor to display data.

**REQUIREMENTS**

**Hardware Requirements**



**1. Arduino uno 2. Heartbeat sensor.**

* + 1. **Jumper wires**
    2. **USB cable**

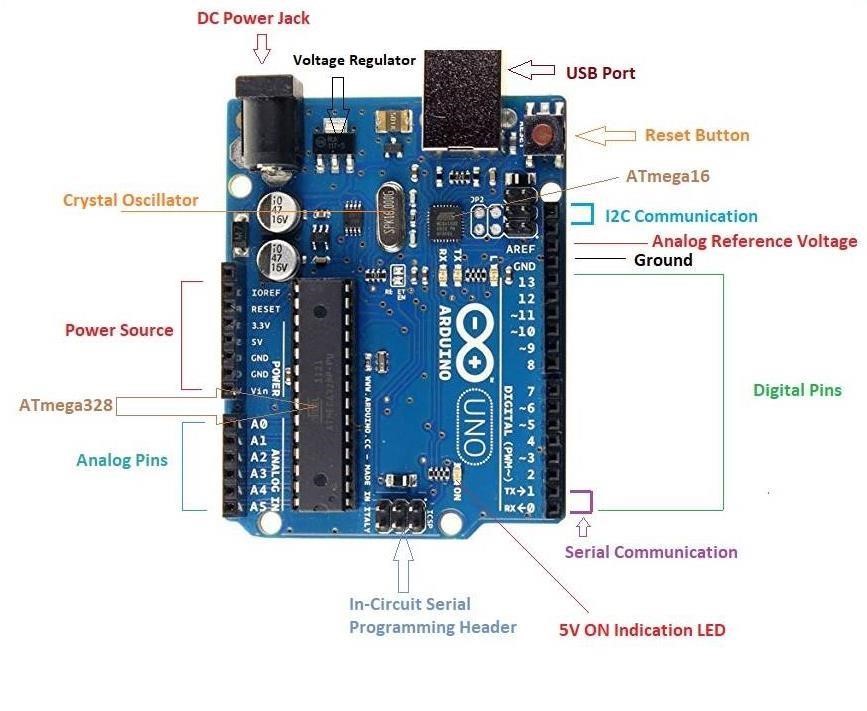
**Software Requirements**



**1.Arduino IDE**

* 1. **Hardware Requirement:**

**3.1.1 ARDUINO UNO BOARD**



**Figure 2: Arduino Uno Board**

**Arduino Uno** is a microcontroller board based on the ATmega328P ([datasheet).](http://www.atmel.com/Images/doc8161.pdf) It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again."Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

**GENERAL PIN FUNCTIONS**

 **LED**: There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

 **VIN**: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

 **5V**: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

 **3V3**: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

**GND**: Ground pins.



**IOREF**: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF

pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

 **Reset**: Typically used to add a reset button to shields which block the one on the board

**Special Pin Functions**

Each of the 14 digital pins and 6 Analog pins on the Uno can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

In addition, some pins have specialized functions:

**Serial**: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

 **External Interrupts**: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

 **PWM**(**P**ulse **W**idth **M**odulation) 3, 5, 6, 9, 10, and 11 Can provide 8-bit PWM output with the analogWrite() function.

 **SPI**(**S**erial **P**eripheral **I**nterface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

 **TWI**(**T**wo **W**ire **I**nterface): A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

 **AREF**(**A**nalog **REF**erence): Reference voltage for the analog inputs

**TECHNICAL SPECIFICATION OF ARDUINO BOARD:**

|  |  |
| --- | --- |
| Microcontroller | [ATmega328P](http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf) |
| Input Voltage (recommended) | 7-12V |
| Operating voltage | 5v |
| Input Voltage(limit) | 6-20V |
| Digital I/O Pins | 6-20v14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 632 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20mA |
| DC Current for 3.3V Pin | 50mA |
| Flash Memory | KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED\_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4mm |
| Weight | 25 g |

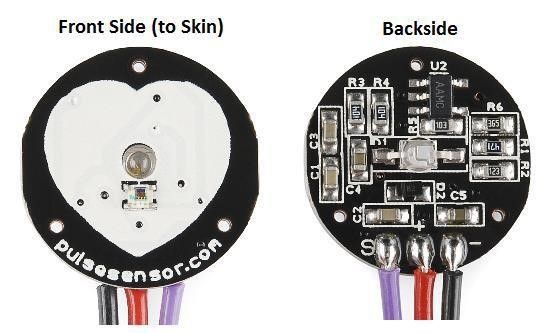
**3.1.2 PULSE SENSOR SENSOR**

It is an Open Source heart rate monitor which considered as a PPG device used to monitor the non-invasive heart rate. It measures the real-time heart beats and calculates BPM with the aid of algorithms implemented by Arduino.This sensor has two sides, the front one which has a heart shape is the side to be attached to the skin. The pins of the pulse sensors are three .If the frond side is facing you, then the most left pin is the GND while the middle one is the input voltage which will be connected to the +5v of the Arduino. The last one for outputting the electrical and will be wired with the analog bins of the Arduino.

The Pulse sensor converts the physical PPG into electrical signals. The sensor outputs a raw signal of analog voltage fluctuations, amplifies it and normalize the wave at V/2.

With every beat of the heart, a pulse wave travel along all arteries to the tissues where the Pulse Sensor is attached. When this pulse wave goes under the sensor, the signal experiences a rapid upward rise in its value. It falls back down toward the normal point and before the next pulse sensor goes under the sensor, the signal stabilizes to the ambient noise.

Due to the repetitive characteristic of the pulse wave, the peak is chosen as a reference point because it’s recognizable. By applying calculation algorithm on the time between each two successive peaks the heart rate is measured. Ideally we want to find the instantaneous moment of the heart beat for accurate measurements. According to heart researchers, the instantaneous moment is when the signal gets 25% or 50% of its amplitude. This pulse sensor first measures the IBI when the signal gets 50% of the amplitude, which from the BPM is derived from average of 10 IBI times.



**Figure 3: Pulse Sensor**

**3.2 SOFTWARE REQIREMENT**

**3.2.1 Arduino IDE:**

Is the required software environment to program the Arduino by writing a code and upload it to the Arduino. It also outputs the results for analysis using both serial monitor and serial plotter. The version used in this project is 1.8.3 (Genuino) which supports both serial monitor to print the HR wave. The Arduino IDE used to write a code The Arduino IDE supports the languages [C a](https://en.wikipedia.org/wiki/C_(programming_language))nd [C++ u](https://en.wikipedia.org/wiki/C%2B%2B)sing special rules of code structuring. The Arduino IDE supplies a [software library f](https://en.wikipedia.org/wiki/Software_library)rom the [Wiring p](https://en.wikipedia.org/wiki/Wiring_(development_platform))roject, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable [cyclic executive p](https://en.wikipedia.org/wiki/Cyclic_executive)rogram with the [GNU toolchain,](https://en.wikipedia.org/wiki/GNU_toolchain) also included with the IDE distribution. The Arduino IDE employs the program a to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware

.

**A algorithm for Measuring BPM:**

The algorithm for measuring BPM and IBI is:

1. Initialize Arduino configurations.

-Baud Rate = 9600.

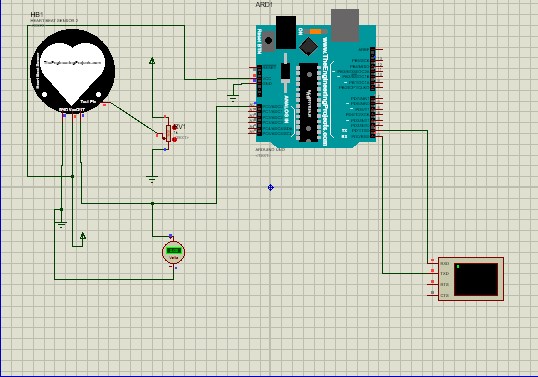
1. Apply input signals to A1.
2. Obtain data from reading A1.
3. Perform Calculations on the data.
4. Print it to the Serial Plotter.
5. Wait 1000ms until the next reading.
6. Repeat from step 3.

**OTHER REQUIREMENTS**

 **Juper wires:**

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with [breadboards a](https://blog.sparkfuneducation.com/what-is-a-breadboard)nd other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn’t get much more basic than jumper wires. Though jumper wires come in a variety of colors, the colors don’t actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power.

**4.1 CIRCUIT CONFIGURATION:**



**PROCEDURE:**

1. Upload the code to Arduino UNO and Power on the system.

* + 1. Displays user's live and changing BPM, Beats Per Minute, in Arduino's native Serial Monitor.
    2. Print: "♥ A HeartBeat Happened !" when a beat is detected, live.

2) Learn about using a PulseSensor Library "Object".

4) Blinks LED on PIN 13 with user's Heartbeat.

--------------------------------------------------------------------\*/

#define USE\_ARDUINO\_INTERRUPTS true // Set-up low-level interrupts for acurate bpm #include <PulseSensorPlayground.h>

// 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.

// Determine which Signal to "count as a beat" and which to ignore.

// Use the "Gettting Started Project" to fine-tune Threshold beyond

int Threshold = 550;

PulseSensorPlayground pulseSensor; // Creates an object of the PulseSensorPlayground

void setup() {

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

// 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.

}

}

void loop() { int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object

// "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” prints "a heartbeat is happened”

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

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

}

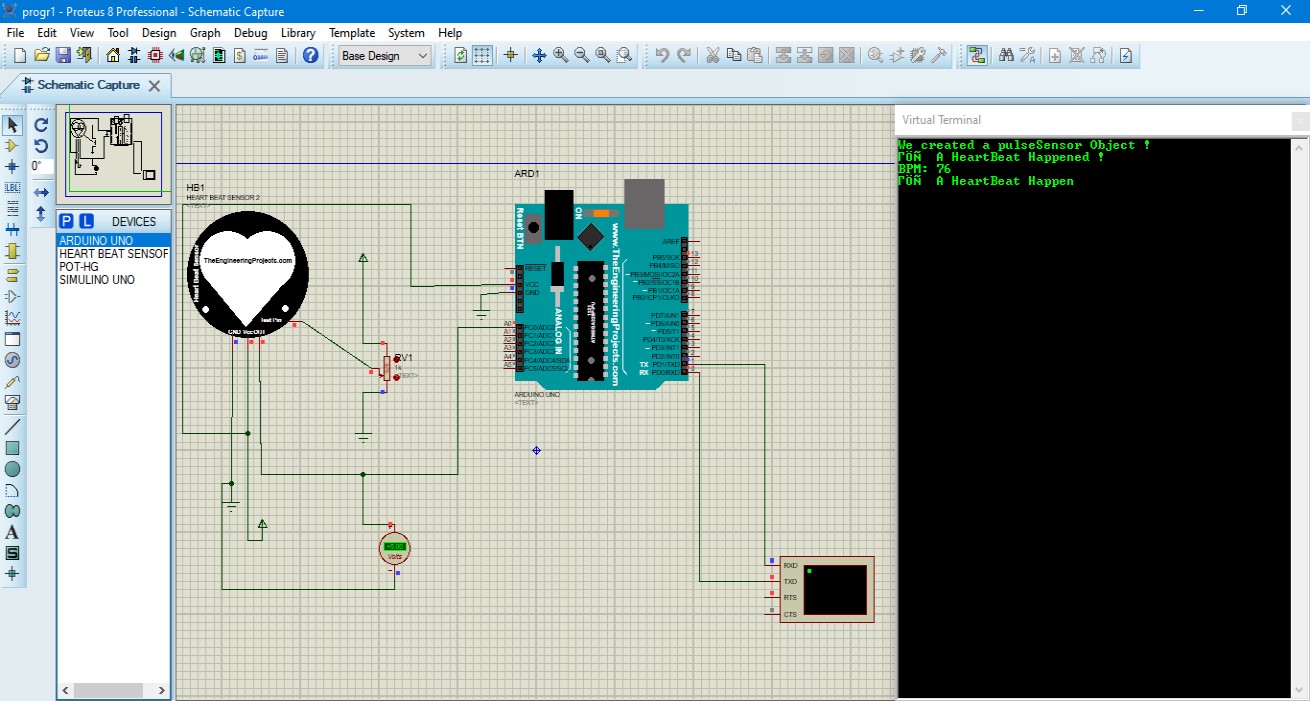
delay(20); // considered best practice in a simple sketch.

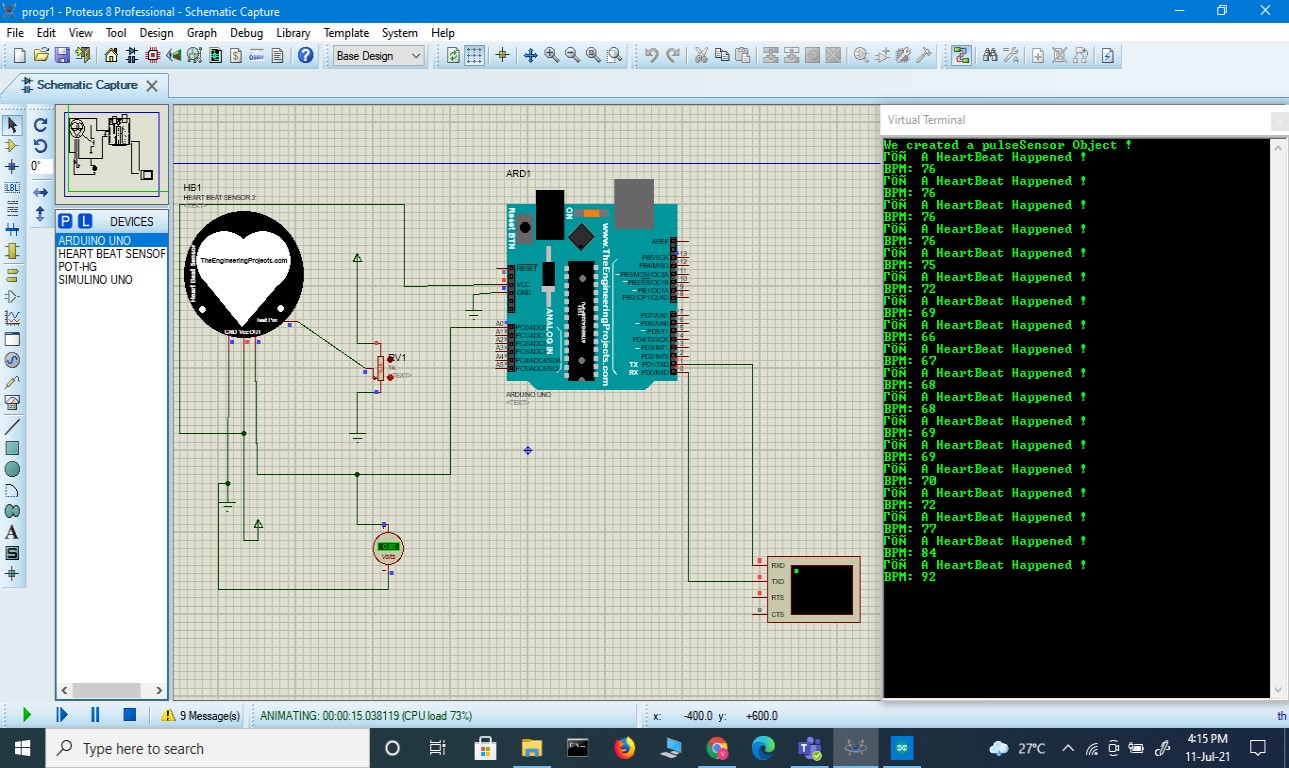
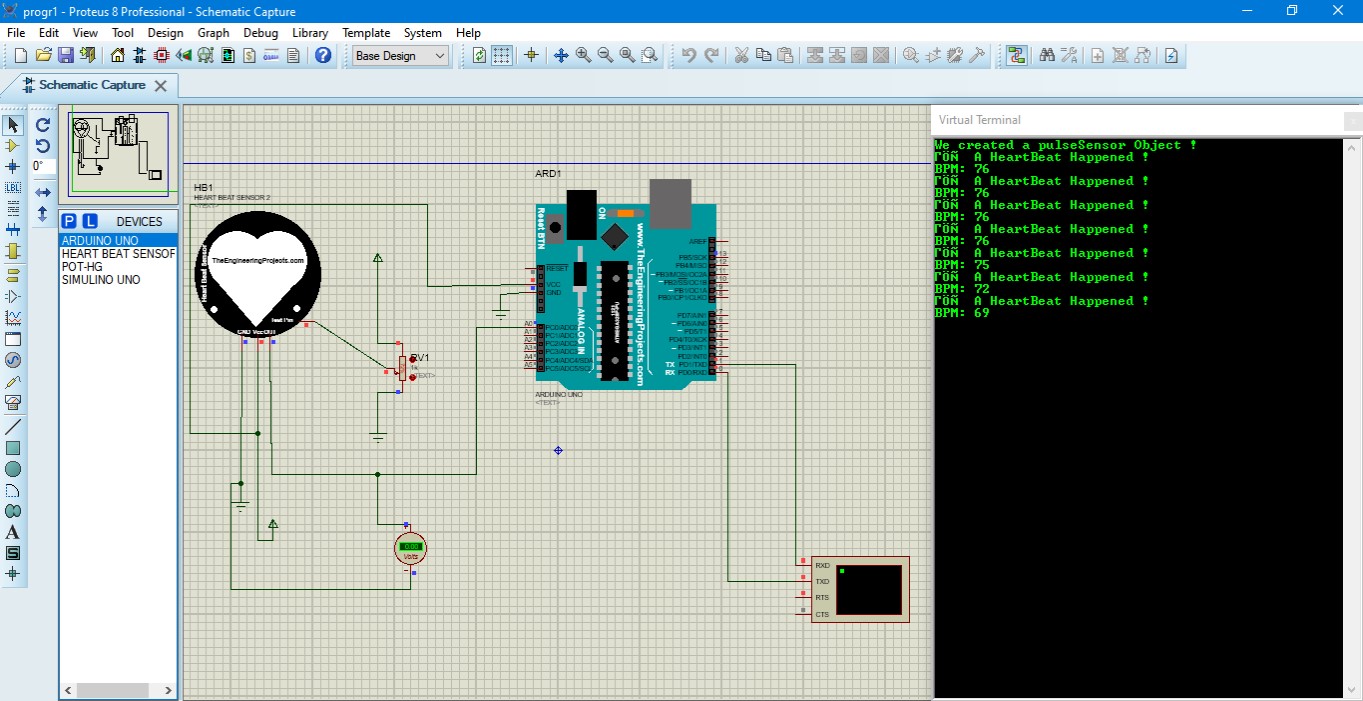
}

1. The Arduino asks us to place our finger in the sensor and press the switch.

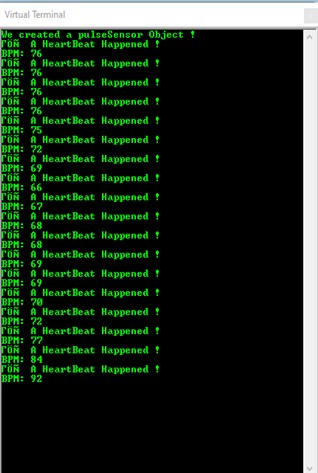
1. Place any finger (except the Thumb) in the sensor clip and push the switch (button). Based on the data from the sensor. Arduino calculates the heart rate and displays the heartbeat in bpm.While the sensor is collecting the data, sit down and relax and do not shake the wire as it might result in a faulty values..

**SIMULATION :**





**Result on serial plotter:**



**Applications**

* This project can be used as an inexpensive alternative to other heart rate monitors
* A set point can determine wheather a person is healthy or not checking his/her heart beat and comparing it with set point

# Future scope

1. Design robust system to improve measuring efficiency even in the presence of noise.
2. An IOT based system.
3. Application to send an SMS containing the measured data and the patient’s location to the medical emergency and emergency contacts of the patient in order to get an ambulance and notify his relatives.

## Challenges and limitation:-

* It may give inconsistent reading which reduces the efficiency of the project
* Logic used is very simple . Therefore, results may vary if a sophisticated instrument is used for the same purpose
* Pulse sensor might not detect accurate readings if it was placed with excessive or loose pressure on the body

REFERENCES

1. J. Allen, “Photoplethysmography and its application in clinical physiological measurement,”

*Physiol. Meas*, vol. 28, pp. R1 – R39, 2007.

1. T. Tamura, Y. Meada, M. Sekine and M. Yoshida, “Wearable photoplethysmographic sensors – past and present,” *Electronics*, vol. 3, pp. 282 – 302, 2014.

P. A. Kyriacou, “Investigation of new electro-optical techniques for monitoring patients,” Ph.D. dissertation, Dept. Med. Electronics and Physics, University of London, London, England, April 2001.

1. J. G. Webster, *Design of Pulse Oximeters*, 1st ed. Bristol: Institute of Physics Publishing, 1997.
2. J. T. B. Moyle, “Uses and abuses of pulse oximetry,” *Archives of Disease in Childhood,* vol.

74, pp. 77-80, 1996. .

1. Texas Instruments. (2014). 2.4 GHz Bluetooth Low Energy System-on-Chip. [Online]. Available: www.ti.com.

Texas Instruments. (2010). LMx24-N, LM2902-N Low Power, Quad-Operational Amplifiers. [Online]. Available: www.ti.com.

**THANK YOU**

VAISHNAVI K

SEM 4 ‘ISE’

NMIT