PULSE RATE SENSOR

PROJECT REPORT

Presented To
22BEC502

Presented By
22BEC508

Acknowledgement

➢Theorecal knowledge is never sufficient It should be practically demonstrated and understood by a person, as we were able to understand during our project. I am falling short of words to express my gratude to all who partcipated in this project and to complete it on me

➢Here I take this opportunity to thank my friend who was in this project with me and we are very thankful to our Project coordinator Dr. Prakash gajjar and H.O.D of E.C department Dr. Usha Mehta for providing us their valuable guidance in every aspect regarding with

our project and helping us to complete it in me without any delay or problem.

Yours faithfully Prit barot- 22bec502

Darshil mavadiya- 22bec508

Overview:

In this project, we will design Heartbeat/Pulse/BPM RateMonitor using Arduino & Pulse Sensor. You can interface the pulse sensor with Arduino for monitoring Heartbeat/Pulse/BPM Rate. I used a 204 LCD panel to display

the pulse rate in BPM. You can even use a 162 LCD display.

This sensor is quite easy to use and operate. Place your finger on top of the sensor and it will sense the heartbeat by measuring the change in light from the expansion of capillary blood vessels. You can use this sensor with ESP8266 to upload the BPM data to the Internet

Pulse Sensor:

Introduction:

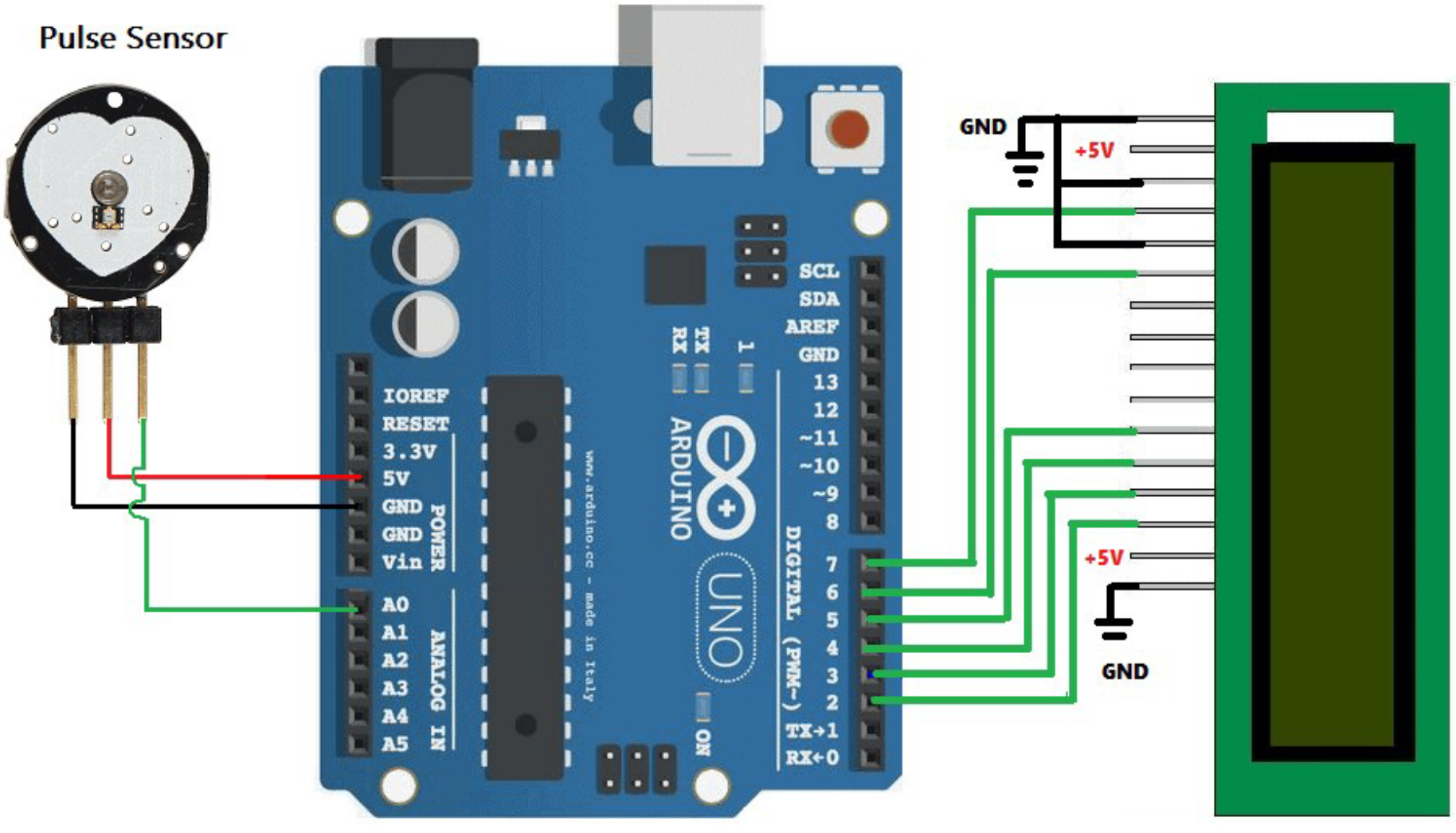
The Pulse Sensor is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects. The essence is an integrated optical amplifying circuit and noise eliminating circuit sensor. Clip the Pulse Sensor to your earlobe or fingertip. Then it into your Arduino, you are now ready to read heart rate.

The front of the sensor comes with the heart logo. This is where you place your finger. On the front side, you will see a small round hole, from where the green LED shines. Just below the LED is a small ambient light photosensor APDS9008 which adjust the brightness in different light conditions.
On the back of the module you will find MCP6001 Op-Amp IC, a few resistors, and capacitors. This makes up the R/C filter network. There is also a reverse protection diode to prevent damage if you connect the power leads reverse.

PinOut – Pulse Sensor:

The pulse sensor has three pins: VCC, GND & Analog Pin. The module operates from a 3.3 to 5V DC Voltage supply with an operating current of < 4mA.

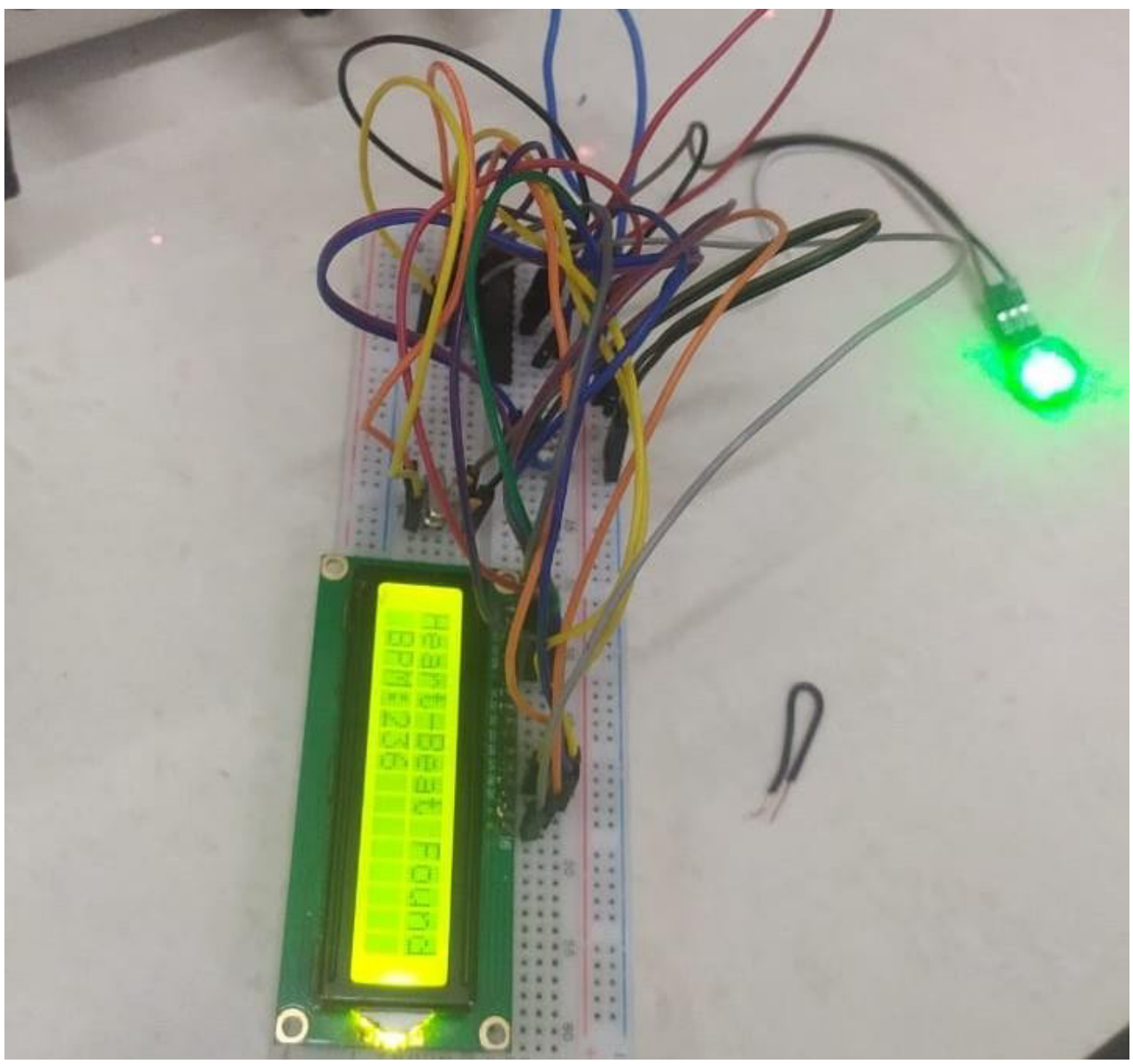
Pulse Rate (BPM) Monitor using Arduino



Working of the Project:

When a heartbeat occurs, blood is pumped through the human body and gets squeezed into the capillary tissues. Consequently, the volume of these capillary tissues increases. But in between the 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 can be measured with the help of a microcontroller.

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.



Source code:

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int pulsePin = A0;                 // Pulse Sensor purple wire connected to analog pin A0
int blinkPin = 13;                 // pin to blink led at each beat

volatile int BPM
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
                                   lcd.begin(16, 2);
 lcd.clear();
}


// 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);
     lcd.print("Heart-Beat Found ");
     lcd.setCursor(1,1);
     lcd.print("BPM: ");
     lcd.setCursor(5,1);
     lcd.print(BPM);
```

```
    delay(300);
    lcd.clear();
  }
 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
```

```
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
```

```
    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
    firstBeat = true
    secondBeat = false
  }

 sei();
}// end isr
```

Conclusion

In conclusion, we have learned about a simple pulse sensor that detected pulse on the basis of light. Through various example sketches from the PulseSensor Playground library, we saw its different features. We monitored the user's pulse through blinking the onboard LED as well as plotting it in the serial monitor. The heart rate (BPM) was also demonstrated through another example sketch. Additionally, we also used the pulse processing visualizer app to show the BPM, IBI, and pulse in real time.