CSE 2006 - microprocessor and interfacing

PROJECT REPORT

HEART ATTACK DETECTOR

GROUP MEMBERS:

- 16BCE0223- SR NAVYA SREE
- 16BCE2305-SHIVAM DIXIT
- 16BCE0468-ARCHIT KUMAR
- 16BCE2144-TRIPTI
- 16BCI0122-ASHUMAN PARASHAR

AIM

The aim of this project is to create a heart attack detecting device which can perceive the symptoms of an heart attack in a human and can function accordingly

This system Heart Attack Detection by Heart Rate Monitoring Project helps to inform if a person is about to have a heart attack. This system does this by detecting the heart beat level and informs as soon as the heart beat level does not fall within the permissible limit. Thus this system can be used to save life of many people as this system alerts the doctor about the patient’s heart beat level. For this the system uses two circuits. One is the transmitting circuit which is with the patient and the other is the receiver circuit which is being supervised by the doctor or nurse. The system makes use of heart beat sensor to find out the current heart beat level and display it on the LCD screen.

ABSTRACT

- A **piezoelectric sensor** is a device that uses the **piezoelectric** effect, to measure changes in pressure, acceleration, temperature, strain, or force by converting them to an electrical charge.
- The sensor will detect higher alterations in pulse rate and when there are greater alterations motor will start vibrating.
- If the person stops the piezoelectric sensor within a specific buffer time for eg .15 seconds, the vibrations will stop, else signal will be produced by the sensor.

The signal produced by the piezoelectric will undergo signal conditioning and then converted into digital data by Arduino. Using the JSS Module, the nearest hospital will be informed about the emergency and the data will be sent to a computer for display and further analysis

Piezoelectric sensor

- A piezoelectric sensor is a device that uses the piezoelectric effect, to measure changes in pressure, acceleration, temperature, strain, or force by converting them to an electrical charge. The prefix piezo- is Greek for 'press' or 'squeeze'.

Method:

The pulse sensor we are going to use is a plug and play **heart rate sensor.** 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

COMPONENTS:

The main components in the project will be:

- Heart Sensor
- Arduino
- Motor
- PeasoElectric Sensor

Pin Out – Pulse Sensor

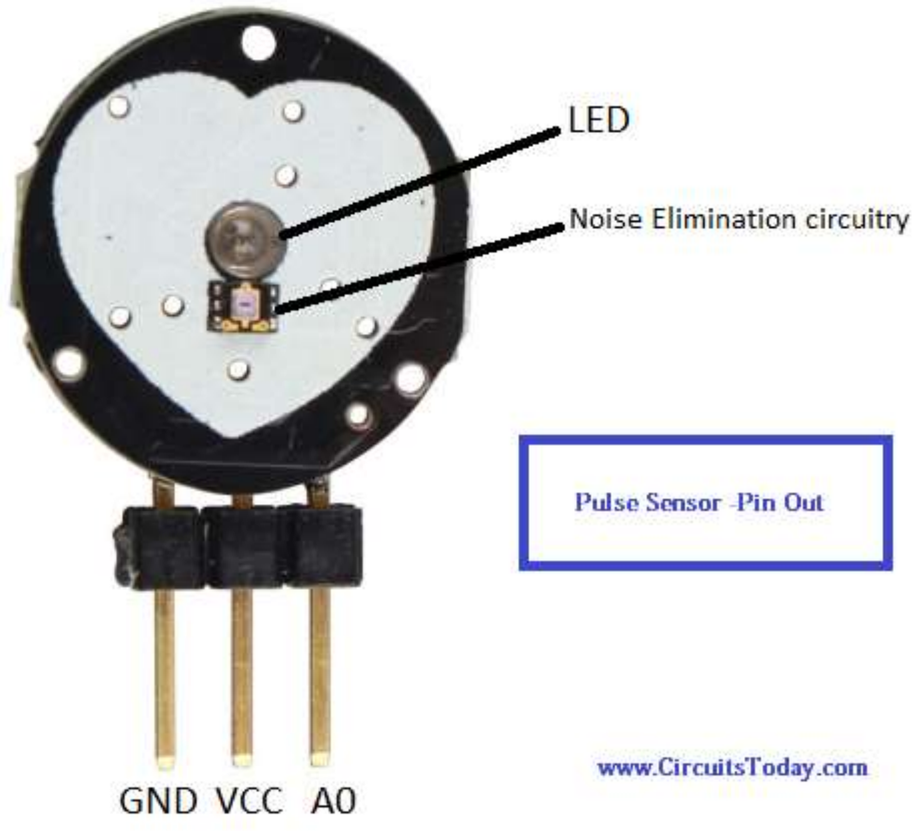
The pulse sensor has three pins which are as described below:

GND: Ground Pin

VCC: 5V or 3V Pin

A0: Analog Pin

There is also a LED in the center of this sensor module which helps in detecting the heartbeat. Below the LED, there is a noise elimination circuitry which is supposed to keep away the noise from affecting the readings.

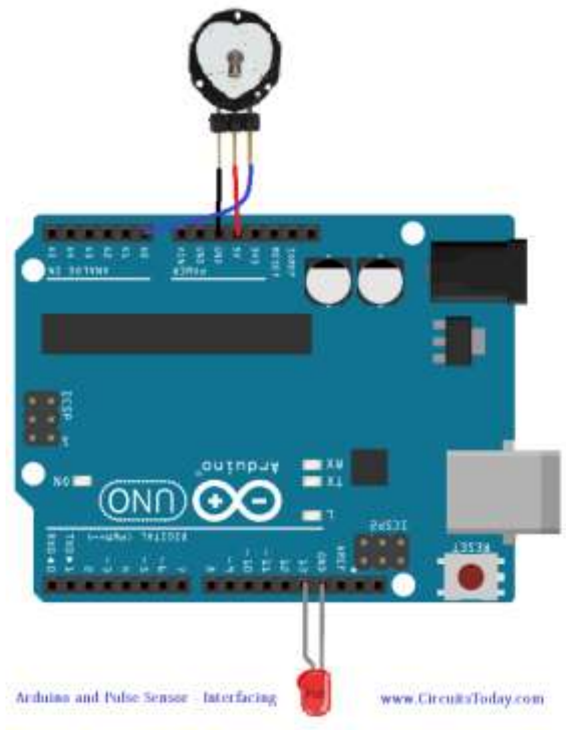


Working – Pulse Sensor

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 coditioned to measure heartbeat and then programmed accordingly to read as heartbeat count.

Circuit Diagram – Pulse Sensor to Arduino



Connect the pulse sensor with Arduino as follows

- GND pin of pulses sensor to GND of Arduino
- VCC of pulse sensor to 5V of Arduino
- A0 of pulse sensor to A0 of Arduino

After that, connect the LED to pin 13 and GND of Arduino as shown in the figure below. The LED will blink according to the heart beat.

Code/Program

```
int sensor_pin = 0;

int led_pin = 13;

volatile int heart_rate;

volatile int analog_data;

volatile int time_between_beats = 600;

volatile boolean pulse_signal = false;
volatile int beat[10];        //heartbeat values will be sotred in this array

volatile int peak_value = 512;

volatile int trough_value = 512;

volatile int thresh = 525;

volatile int amplitude = 100;

volatile boolean first_heartpulse = true;

volatile boolean second_heartpulse = false;

volatile unsigned long samplecounter = 0;   //This counter will tell us the pulse timing

volatile unsigned long lastBeatTime = 0;


void setup()

{

  pinMode(led_pin,OUTPUT);

  Serial.begin(115200);

  interruptSetup();

}


void loop()

{

     Serial.print("BPM: ");

     Serial.println(heart_rate);

     delay(200); //  take a break

}


void interruptSetup()

{
  TCCR2A = 0x02;  // This will disable the PWM on pin 3 and 11

  OCR2A = 0X7C;   // This will set the top of count to 124 for the 500Hz sample rate

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

  TIMSK2 = 0x02;  // This will enable interrupt on match between OCR2A and Timer

  sei();          // This will make sure that the global interrupts are enable

}

ISR(TIMER2_COMPA_vect)

{

  cli();

  analog_data = analogRead(sensor_pin);

  samplecounter += 2;

  int N = samplecounter - lastBeatTime;


  if(analog_data < thresh && N > (time_between_beats/5)*3)

    {

      if (analog_data < trough_value)

      {

        trough_value = analog_data;

      }

    }


  if(analog_data > thresh && analog_data > peak_value)

    {

      peak_value = analog_data;

    }
```

```
 if (N > 250)
{
  if ( (analog_data > thresh) && (pulse_signal == false) && (N > (time_between_beats/5)*3) )
    {
      pulse_signal = true;
      digitalWrite(led_pin,HIGH);
      time_between_beats = samplecounter - lastBeatTime;
      lastBeatTime = samplecounter;

     if(second_heartpulse)
      {
        second_heartpulse = false;
        for(int i=0; i<=9; i++)
        {
          beat[i] = time_between_beats; //Filling the array with the heart beat values
        }
      }

      if(first_heartpulse)
      {
        first_heartpulse = false;
        second_heartpulse = true;
        sei();
        return;
      }
    word runningTotal = 0;

    for(int i=0; i<=8; i++)
      {
        beat[i] = beat[i+1];
        runningTotal += beat[i];
      }

    beat[9] = time_between_beats;
    runningTotal += beat[9];
    runningTotal /= 10;
    heart_rate = 60000/runningTotal;
  }
}



if (analog_data < thresh && pulse_signal == true)
  {
    digitalWrite(led_pin,LOW);
    pulse_signal = false;
    amplitude = peak_value - trough_value;
    thresh = amplitude/2 + trough_value;
    peak_value = thresh;
    trough_value = thresh;
  }

if (N > 2500)
  {
      thresh = 512;

      peak_value = 512;

      trough_value = 512;

      lastBeatTime = samplecounter;

      first_heartpulse = true;

      second_heartpulse = false;

    }


  sei();

}
```

Code Explanation

In the interrupt function, we have set up a timer that will throw an interrupt every other millisecond which gives us a sample rate of 500Hz and a beat to beat timing resolution of 2mS. This will disable the PWM output on the pin 3 and 11 and also will disable the tone() function. sei() ensures that the global interrupts are enabled.

```
TCCR2A = 0x02;

 TCCR2B = 0x06;

 OCR2A = 0X7C;

 TIMSK2 = 0x02;

 sei();
```

The following function runs after every 2mS. It takes reading from the pulse sensor every 2mS and increments the sample counter. The sample counter is used to keep track of the time and the N variable is used to avoid the noise.

```
ISR(TIMER2_COMPA_vect)
```

```
{   cli();

  analog_data = analogRead(sensor_pin);

  samplitudeleCounter += 2;

  int N = sampleCounter - lastBeatTime;
```

The following two loops will keep track of the highest and lowest values. The threshvariable is initialized at 512 which is the middle point of the analog point. This variable is used to keep track of the middle point.

```
if(analog_data < thresh && N > (time_between_beats/5)*3)

   {         if (analog_data < trough_value)

     {

       trough_value = analog_data;

    }   }
```

The following function will look for a heartbeat. If the output reading has passed the thresh value and 3/5 of the last time between the beats has passed, then the pulse signal will become true and the LED will become high. Then we update the variable ‘lastbeattime’ by calculating the time since the last beat.

```
if (N > 250)
  {

    if ( (analog_data > thresh) && (pulse_signal == false) && (N > (time_between_beats/5)*3) )

     {             pulse_signal = true;

       digitalWrite(led_pin,HIGH);

       time_between_beats = sampleCounter - lastBeatTime;

       lastBeatTime = sampleCounter;
```

In the start, we have initialized the first beat as true and the second beat as false. So, if we have the first reading, then it gets kicked by return. While in the second reading, we seed the rate[] array which will help us in calculating the BPM. The BPM is actually derived from the average of last 10 time between beat values.

```
if(second_heartpulse)

      {    second_heartpulse = false;

        for(int i=0; i<=9; i++)

        {      beat[i] = time_between_beats; //Filling the array with the heart beat values

        }       }

      if(first_heartpulse)

      {

        first_heartpulse = false;

        second_heartpulse = true;

        sei();

        return;

      }
```

Then we shifted the data from the rate[] array to the ‘runningtotal’ variable. The old value will fall out and the fresh value will keep coming in every time the function will run. Then we averaged the array and calculated the BPM.

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

      {

        beat[i] = beat[i+1];

        runningTotal += beat[i];

      }

     beat[9] = time_between_beats;

     runningTotal += beat[9];
```

```
    runningTotal /= 10;

    heart_rate = 60000/runningTotal;
```

We calculated the beat when the pulse value was greater than the thresh value. When the pulse value is less than the thresh value, then we assume that the pulse is over and the LED will go down. After that, we update the new 50% mark for the thresh variable.

```
if (analog_data < thresh && pulse_signal == true)

  {

    digitalWrite(led_pin,LOW);

    pulse_signal = false;

    amplitude = peak_value - trough_value;

    thresh = amplitude/2 + trough_value;

    peak_value = thresh;

    trough_value = thresh;

  }
```

If we found no beat for about 2.5mS, then the variables that we used to calculate the heart beat will be given the initial values.

```
if (N > 2500)

  {

    thresh = 512;

    peak_value = 512;

    trough_value = 512;
    lastBeatTime = sampleCounter;

    first_heartpulse = true;

    second_heartpulse = false;

  }
```

REFERENCES:

- WWW.CIRCUITSTODAY.COM/PULSE-SENSOR-ARDUINO
- http://nevonprojects.com/heart-attack-detection-by-heart-beat-sensing/