Proof that Tony Stark has a heart

Problem statement

We will use an Arduino to read the photoplethysmography (PPG) waveform obtained through an infrared LED, red LED, and photo-detector placed on the finger to measure heart rate and SpO2 (oxygen saturation). The weak IR signal will be amplified and filtered through an Op-Amp-based analog circuit. The project will also include an alert system that is triggered for abnormal readings, to create a Smart Heart Monitor.

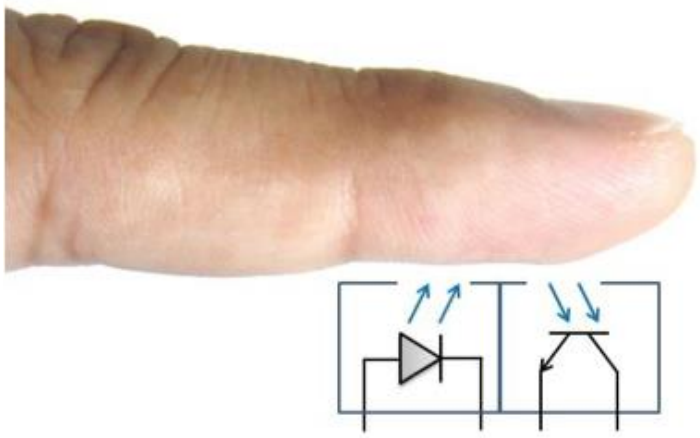
Team

Vaishnav V. Rao
Harshda Saxena
Manan Seth

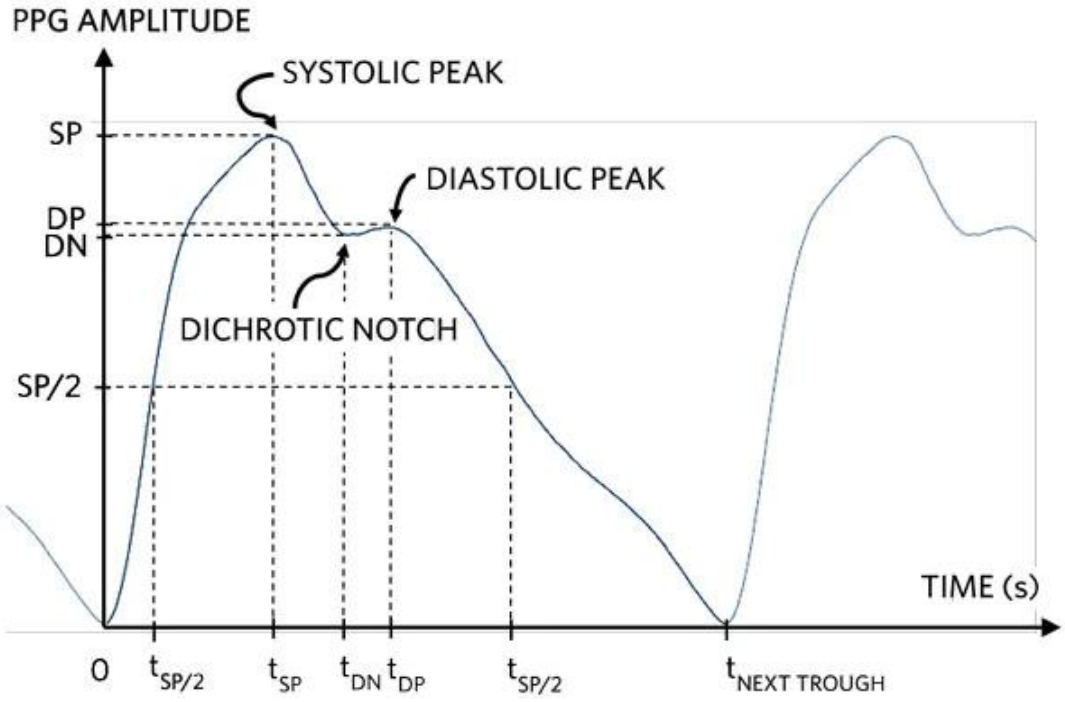
Description

Optical techniques of measuring these characteristics (photoplethysmography or PPG) use the fact that tiny capillaries in any patch of skin (fingertips for our purposes) furnished with a good blood supply, alternately expand and contract in time with the heartbeat. An ordinary infrared LED/phototransistor pair can sense this rhythmic change as small but detectable variations in skin contrast, using reflectance. It is a non-invasive method of finding heart rate.

The reflected IR signal detected by the photodiode is fed to a circuit that filters the unwanted signals and amplifies the desired pulse signal. This will involve the use of low pass filters to remove noise, the use of op-amps to amplify the weak pulsating signal into a TTL pulse, and capacitors to block DC components in the signal. The processed PPG signal is then fed into the Arduino for further analysis.



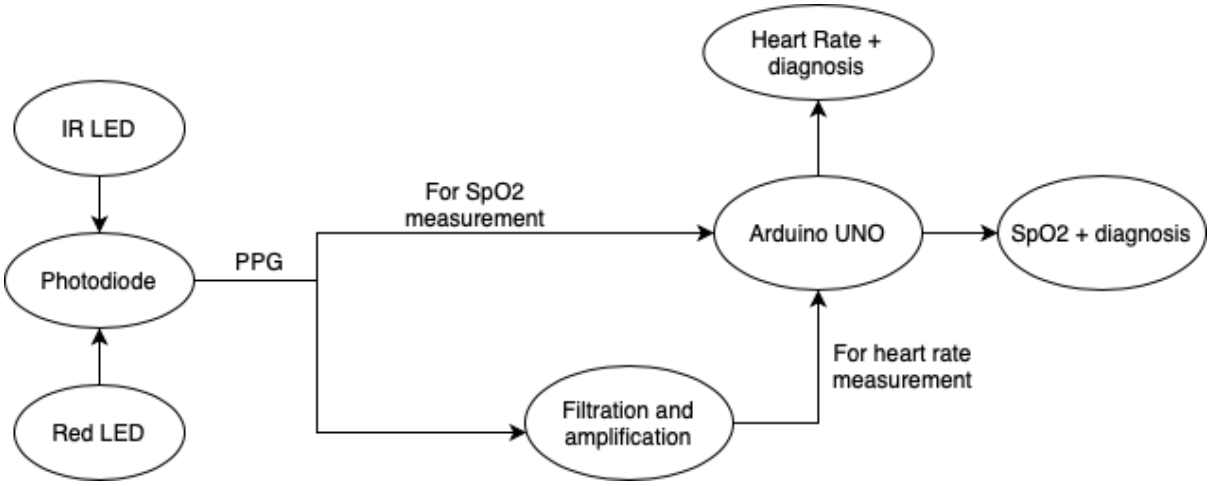
Reflective photoplethysmography



Representative features of a PPG signal

We will use a peak detection algorithm to find the heart rate and use the AC and DC components of the PPG signals corresponding to both the red and infrared LEDs to compute SpO2. The method for computing SpO2 is based on the principle that hemoglobin carrying more oxygen absorbs red light and IR light differently. The alert system will consist of an LCD display. This setup also alerts if one or more of these cardiac parameters are in the abnormal range.

Block diagram



Design details

Our project can be divided into 2 subparts or intermediate goals: the Heart Rate and SpO2 measurements, respectively. The final hurdle was to bring these together and to remove the ambient light variation from our circuit.

Let's focus on the Heart rate first.
Our main detector circuitry consists of an IR emitting diode that is forward biased, and an IR sensor that is reverse biased. They are placed next to each other, facing upwards, since we will be using reflectance.

The signal from the IR sensor (which is the PPG!) is then passed through a 220uF capacitor and fed using a 100 ohm resistor to an inverting Op-Amp with a gain of 1000 (that is, 100k ohm is connected in the feedback loop). This essentially functions as a differentiator (High pass filter), with a threshold frequency of around $2.2*10^{-3}$ Hz. Our pulse usually has a frequency of 1.2 Hz, so this passes nicely through the capacitor, whereas the DC component from the PPG is blocked due to the high impedance. We use a 100n capacitor in the feedback loop, which acts as an integrator (Low pass filter) with a threshold frequency of around 16 Hz, which functions to block the 50Hz ambient noise that the detector picks up from the surrounding.

TLDR, we have a nice signal without the DC component and the 50 Hz noise. This has very small amplitude, and hence we use an inverting amplifier, with V_+ set to 1.6V and gain 1000 to increase the amplitude. To set the V_+, we use a 3 resistor potentiometer, which has another 220uF connected in parallel across 2 of the resistors to remove any noise from the source voltage (here being the Arduino).

This V_+ sets the reference voltage of the Op-Amp, somewhere in between the 0-5V range accessible to us, around which the AC component of the amplified signal oscillates. We feed this into the analog pin of the Arduino and display it using the nice Serial plotter feature. (Note that this is inverted; hence we will need to plot 1023 - analogRead(pin) to get the correct waveform.)

To get the heart rate, we employ a peak detection algorithm. We used a variable (rise_count) which counts the times the recent value of the waveform is greater than the value 10ms before it. If the waveform just starts falling, and if rise_count is greater than some threshold (implying the waveform has been rising for some time continuously, compared to random noisy peaks), a maxima is detected and its time is returned. Finally, the time between 2 consecutive maximas are stored, and an averaging is performed to only count those time periods within 10% of each other, to finally print the pulse.

A few IF conditions checking LOW, NORM or HIGH BPM and printing those on the LCD later, voila! We have a very accurate Heart Rate Monitor, giving us values within +-5 BPM of that from a store bought oximeter.

Now, onto the SpO2 calculation.
To find SpO2, we need the reflected signals of both an IR and a RED LED. Note that since we have just 1 sensor both cannot be on at the same time, and the time difference in the RED and IR measurement does not matter over short enough durations. We first find R given by –

R =((REDmax - REDmin)/(REDmin))/((IRmax - IRmin)/(IRmin))

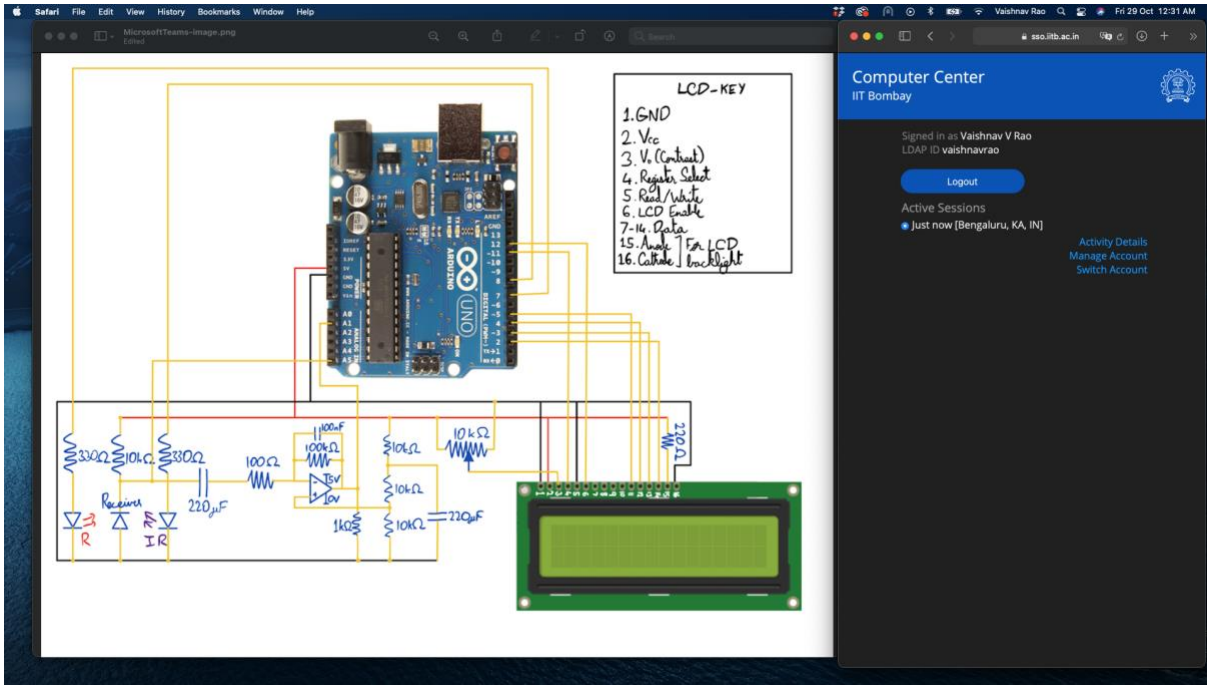
Using this R, we calibrate with a store bought oximeter using a linear relation to get SpO2 as SpO2 = m*R + b. The constants are found by testing on different family members and getting multiple readings.

Note that this requires both the unaffected DC and AC parts of the PPG, and hence all the analog processing we did to make the signal nice cannot be applied. Thus, we directly take the sensor into another analogpin, and carry out the necessary signal processing to get an accurate SpO2 via our code.
The first step would be to remove the 50Hz noise picked up by the sensor. Thus, whenever any of the LED's are on, we average over readings in a period of 20ms to remove the 50Hz noise.

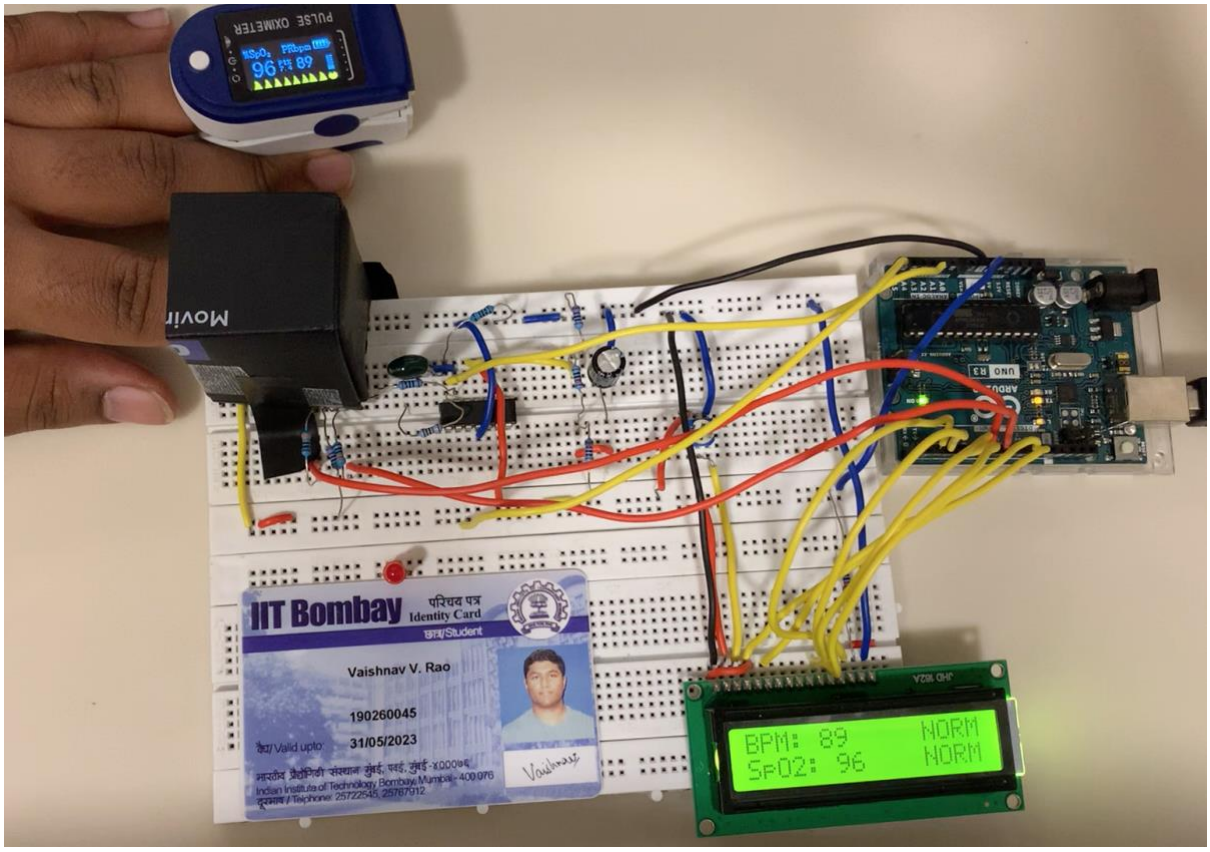
We then subtract the previous average reading from the sum over the last four average readings, add the new average reading, and divide by four to get a "moving average". The result of this process is a clean PPG waveform, albeit of low amplitude. Using these cleaner values collected over a time period of the pulse rate (which we know through the previous heart rate calculation section), we can get the maximum and minimum of the IR and Red signals needed to calculate R.

A bit of calibration and some IF conditions (to indicate if SpO2 is LOW or NORMAL) later, we have a device that reports the SpO2 to within 1% accuracy of the store-bought pulse oximeter

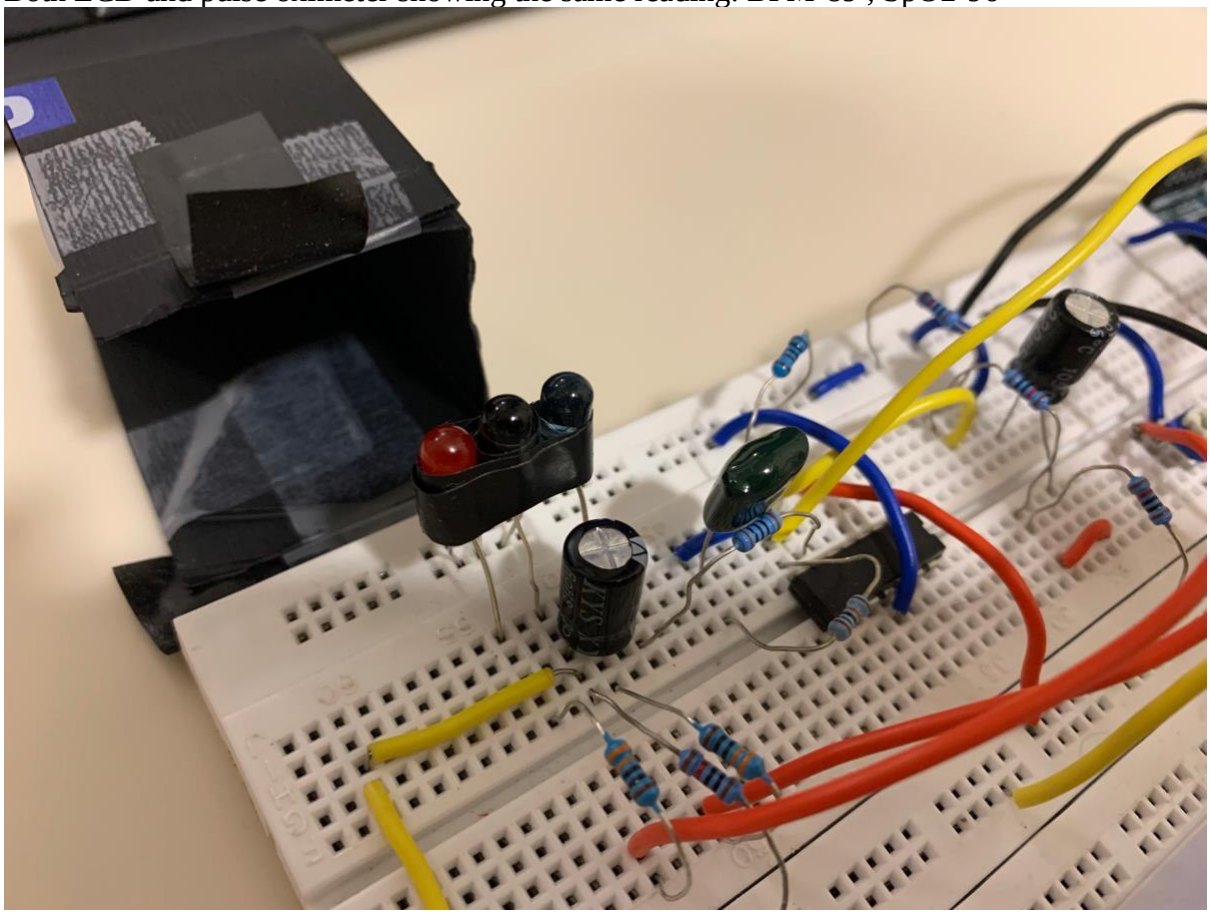
Circuit diagram and image



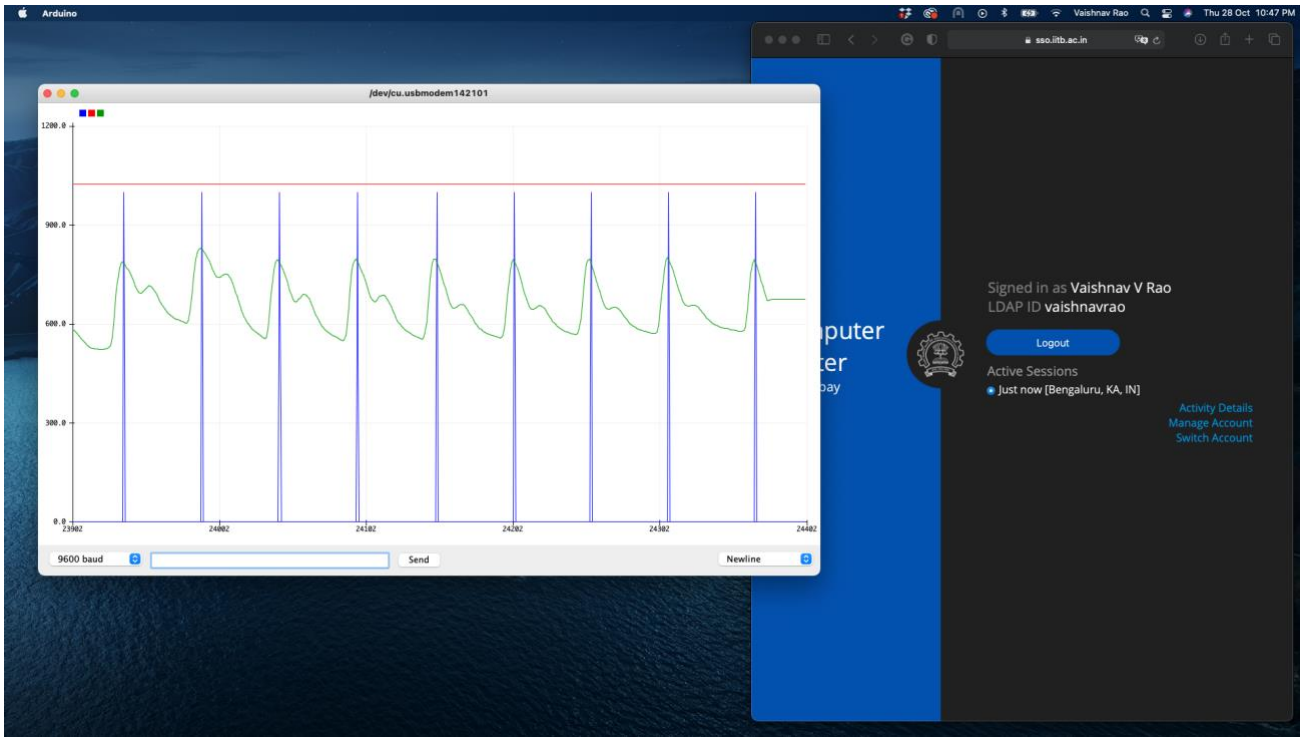
Circuit Diagram



Both LCD and pulse oximeter showing the same reading: BPM-89 ; SpO2-96



Red LED, Photodiode, and IR LED arrangement



The peak detection algorithm in action

Demo Link:

https://drive.google.com/file/d/1xcLlK2SzP-rKNCYRobJora69EYPmAEnm/view?usp=sharing

Bill of materials

Components	Number
Arduino UNO R3	1
JHD162A LCD Display	1
Infrared Emitter	1
Red LED	1
Infrared detector	1
LM324L Op-Amp	1
220μF capacitor	2
100nF capacitor	1
330Ω resistor	2

10kΩ resistor	4
100Ω resistor	1
100kΩ resistor	1
1kΩ resistor	1
220Ω resistor	1
10kΩ potentiometer	1
Wires	Numerous

Status and conclusion

We have achieved all the goals that we set out to accomplish. Our circuit accurately calculates the pulse rate (in beats per minute-BPM) as compared to a store-bought pulse oximeter or Apple watch. Once calibrated, it also calculates the SpO2 within ~1% accuracy of the store-bought pulse oximeter (tested on other individuals!). Based on these readings the device also indicates if the values are HIGH, LOW, or NORMAL.

Some of the major challenges encountered/solutions were:

- Difficulty in signal conditioning, setting of the reference voltage, and amplification of the PPG signal: We studied and used the single supply Op-Amp LM324L in a differentiator cum integrator-like setup to properly filter the signal, and amplify it.
- Peak detection code: Had difficulties differentiating between local maxima due to small variations and the actual systolic peaks. Incorporated a “rise threshold” to overcome this issue.
- Preserving the DC component for SpO2 calculation: Used a second channel to directly take an analog reading from the photodiode and used the “moving average” method (which sampled at specific intervals to eliminate 50Hz noise) to get cleaner readings for these small-amplitude waveforms.
- Maintaining R-SpO2 calibration: We covered the sensor-diode setup to remove variability of ambient light, and once calibrated, it functions perfectly.

Limitations:

- It takes ~20s for the circuit to start printing the waveform (on serial plotter) and make accurate calculations. This can result in erroneous readings initially. This is presumably due to the large capacitance value of the 220μF capacitor.
- Store-bought pulse oximeters are calibrated with R by using best-fit techniques on multiple data points obtained from multiple volunteers. Here, due to physical constraints, we had to find the slope and intercept parameters of the linear relation (between R and SpO2) by using two readings of R and matching them with the readings shown on the pulse-oximeter.
- It takes a long time for the circuit to indicate if no finger has been placed on the detector. This may result in inaccurate readings displayed on LCD if no finger has been kept on the detector.

Lessons learned:

- Signals from natural systems are difficult to measure, process, quantify, and analyze due to lots of variabilities.
- But, these are the most fun systems to study and play with!

Source codes

```
#include <LiquidCrystal.h>
const int maxperiod_siz=80;     // max number of samples in a
period
const int samp_siz=4;

const int sensorPin=A5;
const int analogpin=A1;
const int Threshold1 = 500;
const int Threshold2=900;
volatile int counter=0;
const int Delay=10;
byte beat=LOW;
float oldPulse = 0.0;
const int rise_threshold=7;   //Ref AnalogMaxima()
volatile unsigned long lastNoFin, lastPulse, now;

const int REDLed = 7;
const int IRLed = 8;

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

  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(REDLed,OUTPUT);
  pinMode(IRLed,OUTPUT);

  pinMode(analogpin, INPUT);
  pinMode(sensorPin, INPUT);

  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print("Testing...");
}

int AnalogMaxima() {    //Function to find peak in pulse
  int rise_count=0;
  int check1;
  int check2;
  while(!beat) {
    check1 = 1023-analogRead(analogpin);
    delay(Delay);
    check2 = 1023-analogRead(analogpin);
```

```
    if(check2>check1){
      rise_count++;
    }

    else if(check2<check1) {
      if(rise_count> rise_threshold){ //Waveform should be rising
for at least
        beat=HIGH;                    //rise_threshold number of
times to be counted as maxima
        Serial.print(100);
        lastPulse = millis();
        return millis();
      }
      rise_count=0;
    }
  Serial.print("0 1023 ");            //Prints reference lines
  Serial.println(1023-analogRead(analogpin));  //Prints Pulse
Waveform
  now = millis();
  if(now - lastNoFin > 15000 && now - lastPulse> 3000)    //If it
detects no peaks for 3 second
  {                              //and 15 seconds have passed for
it to have stabilized
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("NO FINGER");

    lastNoFin = millis();
  }
  }
}

void loop() {
  digitalWrite(REDLed,LOW);
  digitalWrite(IRLed,HIGH);     //IR ON Red OFF
  int last_beat, current_beat=0;
  int period_sum=0;
  int period[10];
  int index, samples, samplesCounter;
  index=0;
  samples=0;
  samplesCounter=0;
  last_beat= AnalogMaxima();
  beat=LOW;
  for(int i=0; i<10; i++){    //Storing multiple pulses for
calculation
    current_beat= AnalogMaxima();
    beat=LOW;
    period[i]= current_beat-last_beat;
    last_beat=current_beat;
  }
  int c=0;
  for(int i=1; i<10; i++) {
```

```
    if ( (period[i] <  period[i-1] * 1.1)  &&
                    (period[i] >  period[i-1] / 1.1)  )
{   //Adjacent pulses should have periods within 10% of each
other
       c++;
       period_sum+= period[i];
     }
  }
  samples=period_sum/(samp_siz*Delay*c);      //Number of samples
in one time period
  float pulse= 60000.0/(period_sum/c);

  pulseprint(pulse);    //Function to print pulse on LCD

  float IRreading[samp_siz], Redreading[samp_siz];
  long int now, ptr = 0;
  float IRlast, Redlast, reader, start_time;
  int n;
  for (int i = 0; i < samp_siz; i++){IRreading[i] = 0;
Redreading[i]=0;}  //Initialization
  float IRdata[maxperiod_siz],REDdata[maxperiod_siz];
  int Rptr =0;
  for (int i = 0; i < maxperiod_siz; i++) { IRdata[i] =
0;REDdata[i]=0;}  //Initialization
  float IRmax=0, IRmin=0, REDmax=0, REDmin=0, IRsum = 0, Redsum =
0;
  double R=0;

  while(true){   //Calculating SpO2
     //Red OFF IR ON
     digitalWrite(REDLed,LOW);
     digitalWrite(IRLed,HIGH);
     n = 0;
     start_time = millis();
     reader = 0.0;
     do                 //Taking readings over a fixed time
period
     {
       reader += analogRead (sensorPin);
       n++;
       now = millis();
     }
     while (now < start_time + 20);
     reader /= n;               // Average reading in 20 ms
     IRsum -= IRreading[ptr];   //Removing last measurement of
moving average
     IRsum += reader;           //Adding latest measurement to
moving average
     IRreading[ptr] = reader;   //Adding latest measurement
     IRlast = IRsum / samp_siz; //Moving average reading
     //Serial.print("0 1023 ");
     //Serial.println(IRlast);
```

```
     //IR OFF, Red ON
     digitalWrite(IRLed,LOW);
     digitalWrite(REDLed,HIGH);

     n = 0;
     start_time = millis();
     reader = 0.0;
     do
     {
       reader += analogRead (sensorPin);
       n++;
       now = millis();
     }
     while (now < start_time + 20);
     reader /= n;
     Redsum -= Redreading[ptr];
     Redsum += reader;
     Redreading[ptr] = reader;
     Redlast = Redsum / samp_siz;
     //Serial.print("0 1023 ");
     //Serial.println(Redlast);

     // R CALCULATION
     IRdata[Rptr]=IRlast;
     REDdata[Rptr]=Redlast;
     Rptr++;
     Rptr %= maxperiod_siz;
     samplesCounter++;
     if(samplesCounter>=samples){   //Finding min and max of IR
and RED LED DC values
       samplesCounter =0;
       IRmax = 0; IRmin=1023; REDmax = 0; REDmin=1023;
       for(int i=0;i<maxperiod_siz;i++) {
         if( IRdata[i]> IRmax) IRmax = IRdata[i];
         if( IRdata[i]>0 && IRdata[i]< IRmin ) IRmin = IRdata[i];
         IRdata[i] =0;
         if( REDdata[i]> REDmax) REDmax = REDdata[i];
         if( REDdata[i]>0 && REDdata[i]< REDmin ) REDmin =
REDdata[i];
         REDdata[i] =0;
       }
       R =  ( (REDmax-REDmin) / REDmin) / ( (IRmax-IRmin) / IRmin
) ;
       break;
     }
     ptr++;
     ptr %= samp_siz;
  }
  int spo2= 50*R+49.5;
  spo2print(spo2);
}

void pulseprint(float pulse){   //Function to print pulse on LCD
```

```
  if(pulse>40.0 && pulse<220.0) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("BPM: "+ String(int(pulse)));
  lcd.setCursor(0,1);
  lcd.print("         ");
  }
  else
  {
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("CALIBRATING");
  }
  if(pulse <=50 && pulse >40)
  {
    lcd.setCursor(12,0);
    lcd.print("LOW");
  }
  else if(pulse >=120 && pulse <220){
    lcd.setCursor(12,0);
    lcd.print("HIGH");
  }
  else if(pulse <=120 && pulse >50){
    lcd.setCursor(12,0);
    lcd.print("NORM");
  }
}

void spo2print (int spo2){
  if(spo2 <90 && spo2 > 70)
  {
     lcd.setCursor(0,1);
     lcd.print("SpO2: "+String(spo2));
     lcd.setCursor(12,1);
     lcd.print("LOW");
  }
  else if(spo2>=90 && spo2<=100){
    lcd.setCursor(0,1);
    lcd.print("SpO2: "+String(spo2));
    lcd.setCursor(12,1);
    lcd.print("NORM");
  }
  else if(spo2>100){
    lcd.setCursor(0,16);
    lcd.print("W.P.");
  }
}
```