Arduino code the signal quality assessment algorithm implemented in Arduino due platform in real-time:

```
#include <LiquidCrystal.h>

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

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int len=500;

const int samples=500;

double max_val1;

double temp_var2 ;

const uint16_t Smooth_Window=5;

float X1[len];

float X2[len];

int gate[500];

float Pt[500];

float X3[len];

float X[len];  //signal after base line removal

float xnew[len];

float xx[500];

float xx1[500];

float noise2[len-
1]={0.365257344,0.821042837,0.385456344,0.172799924,0.26316963,0.712609807,0.9726620
8,0.381173258,0.350483402,0.548157168,0.066693058,0.763368558,0.163129827,0.434025134
,0.837910848,0.591062427,0.046326611,0.510617042,0.829463267,0.448402306,0.90063149,0.
07212821,0.968109781,0.594885029,0.570797034,0.752952024,0.953424444,0.649768269,0.50
9420111,0.925890606,0.335985697,0.942380286,0.540903089,0.637997468,0.791092927,0.040
877132,0.586145064,0.626056788,0.240051892,0.860745331,0.333708325,0.519768168,0.5408
96215,0.249217616,0.715121023,0.555422908,0.923399269,0.532272387,0.168532717,0.08781
9456,0.302456341,0.398624684,0.832978142,0.615457028,0.905585486,0.782754716,0.414828
111,0.178711438,0.951970899,0.849751872,0.285327522,0.842963085,0.323300181,0.6222638
88,0.119631283,0.488217985,0.362073478,0.229218268,0.111342182,0.045331346,0.65144462
9,0.649090654,0.140272967,0.694810372,0.005468558,0.750253102,0.617044086,0.434489125
,0.883925704,1,0.132934287,0.861936342,0.684923963,0.458745821,0.609600171,0.07102097
9,0.656408173,0.256446106,0.376076331,0.956382554,0.767358487,0.369461897,0.276122113
```

,0.844077087,0.645533636,0.467673132,0.313260856,0.246352331,0.99542461,0.334442188,0.72435537,0.081653168,0.189231143,0.53155406,0.789215282,0.806369727,0.132192835,0.630437822,0.933716857,0.006684889,0.776644539,0.621965897,0.961863562,0.488137786,0.479933073,0.132509111,0.914106125,0.066510228,0.968053753,0.156915862,0.467057473,0.455594717,0.967586288,0.180038472,0.917590456,0.293342451,0.210312173,0.553343043,0.381251503,0.397350819,0.497595036,0.513928553,0.219377665,0.585552866,0.28408062,0.515526027,0.007823467,0.148881618,0.15619391,0.328741494,0.438662568,0.442063752,0.04646976,0.765028582,0.022448605,0.622895715,0.379995792,0.747660111,0.312793181,0.779729831,0.429911412,0.417434661,0.764413607,0.963982272,0.514774266,0.875234868,0.353535481,0.512598176,0.392725959,0.626854802,0.433246129,0.421969473,0.092569679,0.115278709,0.813183 18,0.172151679,0.077735277,0.515692382,0.400411833,0.6671033,0.41072699,0.40800552,0.46562385,0.031916032,0.443169438,0.237819485,0.946155853,0.133565907,0.063665824,0.919891425,0.238489477,0.601876528,0.086713885,0.980141489,0.758511008,0.759731677,0.482473769,0.128259852,0.369939816,0.667289832,0.081329918,0.484837873,0.375006379,0.289323508,0.787472656,0.379431817,0.55972924,0.127139004,0.040880003,0.542653408,0.267200292,0.705299696,0.723297181,0.951355831,0.819192932,0.802712691,0.105106559,0.763545446,0.673819311,0.394480538,0.172407688,0.904452059,0.829902652,0.931245094,0.972343551,0.981806749,0.754622996,0.569254186,0.991573772,0.994827388,0.034052869,0.914312294,0.664526022,0.449980885,0.6599534,0.75975286,0.029126823,0.961361429,0.126695558,0.592098046,0.898940895,0.290991926,0.279413722,0.248324708,0.685910507,0.210378013,0.940978087,0.617933516,0.340296683,0.397438578,0.538777707,0.170212691,0.145550192,0.857643145,0.402350543,0.109548,0.342775753,0.249027958,0.001553505,0.834210697,0.703026093,0.956630824,0.149215201,0.605591726,0.075560487,0.14200149,0.739239314,0.867236378,0.235507274,0.059794417,0.918327783,0.288463686,0.902303937,0.133694563,0.872199442,0.500326048,0.648726066,0.136737525,0.554740171,0.825304405,0.625865741,0.648907902,0.465476803,0.845226206,0.460017688,0.627555111,0.65867875,0.390492945,0.575943734,0.422763753,0.38386654,0.105361665,0.689442141,0.434831153,0.203734171,0.111823017,0.962110305,0.432646626,0.635890635,0.959385324,0.824329854,0.512406882,0.752063563,0.228081945,0.64712569,0.895639038,0.010569943,0.962684176,0.236774032,0.555497522,0.972162646,0.874183065,0.195933614,0.173559829,0.261501122,0.363550308,0.221762426,0.020011683,0.697359716,0.631610189,0.808813049,0.94380535,0.266061994,0.79707738,0.873739962,0.34285224,0.437705094,0.626933431,0.939902642,0.679430501,0.504229968,0.109310085,0.50649948,0.954918794,0.798302865,0.630935876,0.648087036,0.427500875,0.791009198,0.436692955,0.210820464,0.292013405,0.552391396,0.057415878,0.465124981,0.654999978,0.573 3996,0.072038152,0.221265764,0.772501322,0.966097509,0.10368123,0.659719847,0.813973824,0.82870726,0.353426723,0.479581418,0.637711029,0.586550396,0.396365002,0.451252876,0.018522078,0.039054149,0.147299398,0.055513705,0.457217573,0.568709507,0.810511647,0.845183805,0.15359121,0.313499498,0.988148251,0.405142299,0.331981101,0.20314804,0.323531791,0.452969908,0.342132439,0.045495817,0.484188234,0.216811327,0.701545698,0.536449652,0.091139806,0.787537171,0.261359863,0.118347039,0.579433733,0.090151321,0.258347509,0.696602989,0.065696205,0.83537958,0.882126622,0.782416224,0.339765394,0.72141392,0.891769043,0.641549016,0.703953478,0.619297849,0.087547294,0.251609526,0.265338639,0.18384179,0.042180437,0.001547875,0.663281883,0.755813644,0.792833252,0.410142605

```
,0.6911845,0.905415426,0.713880366,0.88208216,0.356068593,0.193237149,0.548363604,0.38
9949285,0.144650739,0.29600002,0.666283924,0.898302556,0.961092149,0.47053545,0.61075
0973,0.062580335,0.847112873,0.780559536,0.722502075,0.472755214,0.663344955,0.408430
267,0.577589413,0.565780481,0.55623802,0.507566959,0.799783041,0.129393445,0.84314270
6,0.720046915,0.756792585,0.194217937,0.753177888,0.529132177,0.255347725,0.075915207
,0.140373742,0.314509411,0.496676742,0.17857702,0.748112152,0.50604295,0.662197967,0.3
50895001,0.512222816,0.348333128,0.724792017,0.870414625,0.523852951,0.377936041,0.06
2764742,0.30925843,0.701014634,0.370672134,0.308325853,0.211161875,0.409559624,0.9542
68403,0.761857589,0.546796404,0.100043455,0.049217508,0.077156657,0.425214266,0.21252
2481,0.396932467,0.825183524,0.402597928,0.257371835,0.962113842,0.271148084,0.619619
126,0.57400094,0.32022254,0.4663285,0.053368498,0.636856848,0.856534644,0.351573824,0.
856989647,0.881025526,0.097673277,0.070655606,0.36799412,0.140449997,0.11583343,0.756
52705,0.689522612,0.527978829,0.410563257,0.5028569,0.799674452,0.37853262,0.59013195
7,0.109428998,0.040175067,0.112160971,0.533763507

};   // normalized random noise of 500 samples

float XN[len-1];

float dX[len-1];

float AA[4];

float BB[4];

float r[4];

float sum;

float P,P1,P2;

double timep,time1, timep1,timep21,timep22;


void setup()

{

 for (int i1 = 0; i1 < samples; i1++)

  {

//    ppgsignal[i1] = 0;

    X[i1]=0;

    X1[i1]=0;

    X2[i1]=0;

    X3[i1]=0;
```

```
    dX[i1]=0;
//    tempvec[i1] = 0;
    XN[i1]=0;
  }
  Serial.begin(9600);  // baud rate for the communication between the compulter and the board via serial port
  pinMode(A0,INPUT);
  Serial1.begin(9600); // baud rate for the trasmitting the signal via blue tooth.
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.setCursor(0,0);
  lcd.print("SIGNAL QUALITY:");

}

void loop()
{

int i2 = 0;
  while (i2 < 500)
  {

     xx[i2] = analogRead(A0);  // to read the sensor data from the pulse sensor which is connencted to the analog pin A0 of arduion due board
    delay(10);                // 10 milliseconds given because required sampling frequency (fs) =100 hz;
     i2=i2+1;

  }
```

```
      // ------------------------- """sigal smoothing stage""" -------------------------

for (int i3=0; i3<samples-Smooth_Window+1; i3++)
  {
    double temp_var1=0;
     for (int i4=i3; i4<i3+Smooth_Window; i4++)
     {
      temp_var1=temp_var1+xx[i4]/Smooth_Window;
     }
     xx[i3] = temp_var1;
  }

// ------------------------- """sigal mean reamoval stage"""

 temp_var2 = 0;
  for (int i5 = 0; i5 < samples; i5++)           // mean caluculation
  {
    temp_var2 = temp_var2 + xx[i5] / samples;
    //Serial.println(temp_var2);
  }

  for (int i6 = 0; i6 < samples; i6++)         // mean reamoval
  {
    xx[i6] = xx[i6] - temp_var2;

  }
```

```
// "Baseline removal from the signal start here"

BB[0]=0.9850; BB[1]=(-2.9550); BB[2]=2.9550; BB[3]=(-0.9850); // coefficients of the chebyshew high pass filter with cutt of frequency 0.3 hz of order 3

AA[0]=1.0000; AA[1]=(-2.9698); AA[2]=2.9400; AA[3]=(-0.9702); //coefficients  of the chebyshew high pass filter with cutt of frequency 0.3 hz of order 3

X1[0]=BB[0]*xx[0];

X1[1]=BB[0]*xx[1]+BB[1]*xx[0]-AA[1]*X1[0];

X1[2]=BB[0]*xx[2]+BB[1]*xx[1]+BB[2]*xx[0]-AA[1]*X1[1]-AA[2]*X1[0];


  for(int n=3; n<len; n++)


  {


     X1[n] = BB[0]*xx[n]+BB[1]*xx[n-1] + BB[2]*xx[n-2]+BB[3]*xx[n-3]- AA[1]*X1[n-1]-AA[2]*X1[n-2]- AA[3]*X1[n-3];


 }


 // "Baseline removal from the signal end here"


//  smooth the baseline removal signal star here


 int k1=len-1;

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

{

  X2[k1]=X1[i];

  k1--;
```

```
}

X3[0]=BB[0]*X2[0];
X3[1]=BB[0]*X2[1]+BB[1]*X2[0]-AA[1]*X3[0];
X3[2]=BB[0]*X2[2]+BB[1]*X2[1]+BB[2]*X2[0]-AA[1]*X3[1]-AA[2]*X3[0];

  for(int n=3; n<len; n++)

  {

     X3[n] = BB[0]*X2[n]+BB[1]*X2[n-1] + BB[2]*X2[n-2]+BB[3]*X2[n-3]- AA[1]*X3[n-1]-
AA[2]*X3[n-2]- AA[3]*X3[n-3];

  }



 int k2=len-1;
for(int i=0;i<len;i++)
{
  X[k2]=X3[i];
 k2--;
}

////  smooth the baseline removal signal end here
```

```
// difference  operation start here

  for(int k=1; k<len; k++)

 {
  dX[k-1]=X[k]-X[k-1];

 }

 // difference  operation end  here

  // ------------------------- """Signal normalization stage start here"""

   max_val1 = 0;
  for (int i7 = 0; i7 < samples-1; i7++)           // maximum value caluculation
  {
    max_val1 = max(max_val1, abs(dX[i7]));
    //Serial.print(ppgsignal[i7]);Serial.print(',');Serial.println(max_val1);
  }

  for (int i8 = 0; i8 < samples-1; i8++)         // Signal Normalization
  {
    dX[i8] = dX[i8] / max_val1;

  }

// ------------------------- """Signal normalization stage end here """
```

```
for(int i=0; i<len-1; i++)

{

  XN[i]=dX[i]+0.1*noise2[i];  // adding random noise of amplitude 0.1 to the normalized difference signal.

}

//Levinson-Durbin Algorithm to find the the  predictor coefficients calculations star here
      for(int i=0; i<3; i++)
      {
        sum=0;
     for(int j=0 ; j<500-i-1; j++)
     {
      sum=sum+XN[j]*XN[j+i];
     }
      r[i]=sum/(len-i-1);
     }
  P=r[1]/r[0];   //P= predictor coefficients of first order.

//Levinson-Durbin Algorithm to find the the  predictor coefficients calculations end here

// thresold settings for Noise Free, Motion Artifact and Pulse Free signal

if(P>0.93)

{
```

```
for(int i=0; i<499; i++)
{
gate[i]=1;  // if signal is NOISE FREE  gate =1 is generated
Pt[i]=P;
    lcd.setCursor(0,1);
    lcd.print("NOISE FREE      ");
    Serial.print(xx[i]*10+1200);
  Serial.print(',');
  Serial.print(X[i]*10+900);
  Serial.print(',');
  Serial.print(dX[i]*300+600);
  Serial.print(',');
  Serial.print(XN[i]*300+500);
  Serial.print(',');
  Serial.print(Pt[i]);
  Serial.print(',');
  Serial.println(gate[i]*100);
  Serial1.println(X[i]);  // to trasmit the signal via blue tooth

}

}

else if(P<0.93 && P>0)

{
```

```
for(int i=0; i<499; i++)
{
gate[i]=0;  // if signal is MOTION ARTIFACT  gate =0 is generated
Pt[i]=P;
  lcd.setCursor(0,1);
  lcd.print("MOTION ARTIFACT");
  Serial.print(xx[i]*10+1200);
  Serial.print(',');
  Serial.print(X[i]*10+900);
  Serial.print(',');
  Serial.print(dX[i]*300+600);
  Serial.print(',');
  Serial.print(XN[i]*300+500);
  Serial.print(',');
  Serial.print(Pt[i]);
  Serial.print(',');
  Serial.println(gate[i]*100);
}

}

else

{

 for(int i=0; i<499; i++)
{
gate[i]=-1;   // if signal is PULSE FREE gate =-1 is generated
```

```
Pt[i]=P;
  lcd.setCursor(0,1);
lcd.print("PULSE FREE      ");
  Serial.print(xx[i]*10+1200);
  Serial.print(',');
  Serial.print(X[i]*10+900);
  Serial.print(',');
  Serial.print(dX[i]*300+600);
  Serial.print(',');
  Serial.print(XN[i]*300+500);
  Serial.print(',');
  Serial.print(Pt[i]);
  Serial.print(',');
  Serial.println(gate[i]*100);
}

}



}
```