IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 2)
Available online at: www.ijariit.com

Walking stick with heart attack detection

Cheedella Sai Teja
saiteja.chedella@gmail.com
Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu

ABSTRACT

The Walking Stick with Heart Attack Detection is an experiment that can be used to help heart problem suffering human being, to detect heart attack and to call doctor's help. It was designed specially to help heart problem citizens and patients with heart issues. It consists of three basic sub schemes heart attack detection, the algorithm of analysis and Wireless Communication. The first scheme consists of the electro-cardio-grapy method. This method is used to capture the heartbeat signal on the user's wrist from the patient, and the rest two are kept in the stick. The stick consists of microcontroller that will process the heart attack algorithm. The microcontroller on the stick runs a heart attack algorithm. The indication about him/her of heart condition will be given in the form of alert message. Communication channel will connect the doctor/emergency for medical help of his/her heart attack condition alert. Each of the three sub-schemes mentioned responded undoubtedly.

Keywords— *EGC, Heart beat, Heart attack, Bluetooth, Stick*

1. INTRODUCTION:

The World Health Organisation (WHO) states that "About 6,10,000 people die of heart disease across the world every year–that's 1 in every 4 deaths. Heart disease is the leading cause of death for both men and women. More than half of the deaths due to heart disease were in men. About one-half of those who die to do so within 1 hour of the start of symptoms and before reaching the hospital. A heart attack happens if the flow of oxygen-rich blood to a section of heart muscle suddenly becomes blocked and the heart can't get oxygen. Most heart attacks occur as a result of ischemic heart disease. ... If the clot becomes large enough, it can mostly or completely block blood flow through a coronary artery. The World Health Organisation (WHO) recommended that "everyone should know the warning signs of a heart attack and how to get emergency help". Electro cardio grapy will represent or detects the symptoms of a heart attack.

An ECG is an electrical representation of the heartbeat and is used to display the indication of heart disease. The heart is beating in a regular sinus rhythm between 60 - 100 beats per minute (specifically 82 bpm). An electrocardiograph is a machine that is used to perform electrocardiography and produces the electrocardiogram. ECG records the electrical activity generated by heart muscle depolarizations, which propagate in pulsating electrical waves towards the skin. By the ECG waveform, the condition of the patient heart can be explained by doctors.

The ECG library will contain abnormal ECG waveform, and the data is retrieved mostly from aged people who are more than 55 years old. The risk of heart attack increases steadily for men after age 45, and the average age of a man having his first heart attack is about 66. Actually, this Experiment is designed to help the heart attack people who need walking aids by walking stick. The walking stick is used as alerting unit or detection unit. It will directly connect for emergency immediately.

Fig. 1: walking stick image with heart attack detection

1.1 Project design

The ECG unit on the handling part detects abnormal heartbeat signal from the person who holds the stick. The microcontroller is present in the stick runs a heart attack algorithm. The indication about him/her of heart condition will be displayed in LCD (if it is placed). The wireless communication like Bluetooth, Wi-Fi, Zigbee, GSM will communicate to emergency/doctors for medical help at the moment of a heart attack.

Consideration of this project to shorten the time between the moment of heart attack and the arrival of the doctor to the respective person. The indication before the call to the doctor will give the patient a chance to avoid a heart attack. Two electrodes (biosensors) are placed on the user's handling part of the stick will give the ECG signal. That signal will send the next field that will do the conversion/amplifies the analog ECG signal to digital signal and the transmitted to the respective unit present in a walking stick. The ECG unit, the ADC (analog to digital conversion), and the part that consists of transmitting data will wear on the user's wrists. The wireless connection between the transmitting unit present on the wrist and the main circuit unit on the walking stick gives the user to move freely by avoiding wire attachment between the wrist and the stick.

The digital ECG signal will be received from the receiver part on the stick, and a process of heart attack to detect heart attack symptoms will be processed in the microcontroller. Heart attack symptom is detected, the risk level rises. When the risk level reaches, it converted to the emergency mode, the GSM module will activate the patient's mobile phone to call/alert message to respective doctors for medical help. Smart mobile phones can include a GPS function.

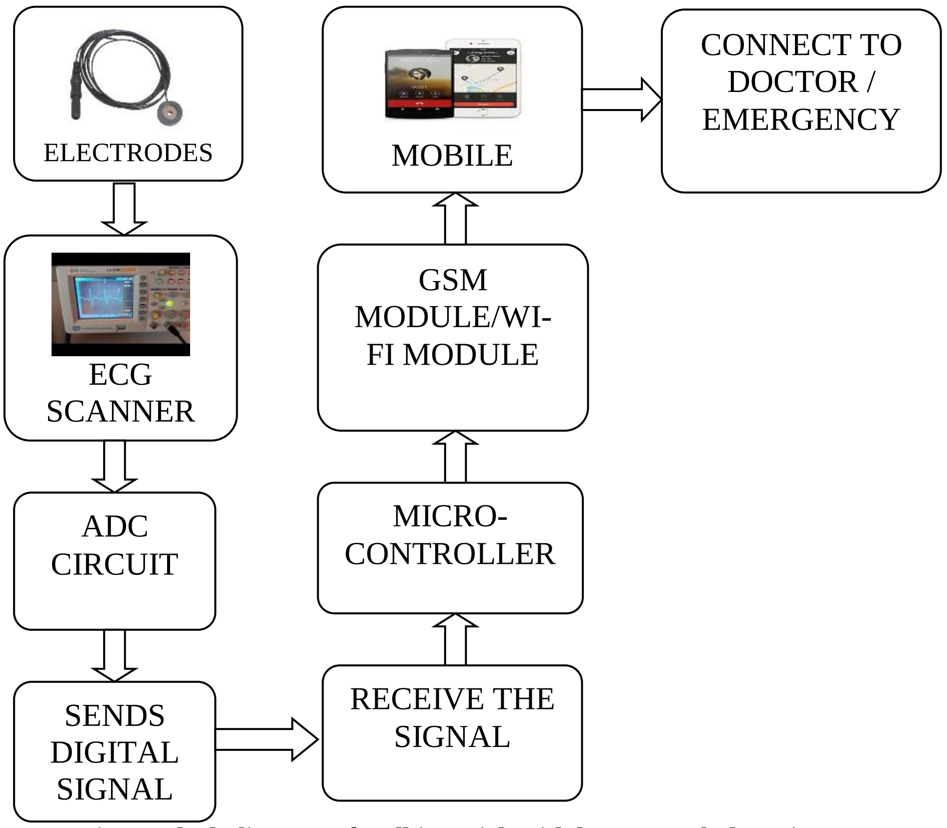

Fig. 2: Block diagram of walking stick with heart attack detection

1.2 Sub- strategizes

Block diagram consists of sub-strategizes: ECG unit, Analysis Circuit, Communication. The ECG circuit will be worn on the hand/wrist and the remaining part is placed in the stick. The ECG circuit unit captures and displays ECG graph from the wrists by electrodes. The circuit will amplify and filters the ECG signal from the ECG graph. Then, the waveform will be in analog form is converted to digital signal by an Analog to digital convertor and sends that digitalized signal to the stick. In the Analysis unit, the receiver will capture the digitalized ECG signal and sends it to the microcontroller. The microcontroller will run the specified algorithm to detect a heart attack or heart attack symptoms and to raise the risk level. When a heart attack symptoms are confirmed, a signal is sent to the Communication unit. In the Communication unit, the module (wi-fi, Bluetooth, GSM) activates the mobile phone connected to the module to process the call/emergency alert message to the doctor.

2. DESIGN ALGORITHM

2.1 Electrodes

As we used Ag-AgCl Electrodes to sense the heart signals. Why the Ag-AgCl ECG electrodes are used? Because the electrodes include good electrical contact with human skin, low motion artifacts and strong adhesive quality to the skin. Electrodes are used as our biosensors. One of the main advantages of using Ag/AgCl electrodes is the low noise level it generates during biological signals recording.

2.2 Analog ECG Circuitry

We consider circuit as three components: unity-gain buffers, a differential amplifier and a band-pass filter. The unity-gain buffers are needed for both wrists as transforms of impedances. An op-amp circuit is a circuit with a very high input impedance. This high input impedance is the reason unity gain buffers are used. This will now be explained. When a circuit has a very high input impedance, very little current is drawn from the circuit. If you know Ohm's law, you know that current, I=V/R. Thus, the greater the resistance, the less current is drawn from a power source. Thus, the power of the circuit isn't affected when current is feeding a high impedance load. As we know that skin has very high impedance, the input impedance of op-amps is infinity and the op-amps will be able to receive the biosignals from the two electrodes. A differential amplifier is a type of electronic amplifier that amplifies the difference between two input voltages but suppresses any voltage common to the two inputs. It is an analog circuit with two inputs and one output in which the output is ideally proportional to the difference between the two voltages. The differential amplifier will take the two biosignals from the electrodes and differentiate the signals with the gain to get the desired ECG waveform. A band-pass filter is a device that passes frequencies within a certain range and rejects frequencies outside that range. So, the filter will make sure that the noise of frequencies outside 0.5 Hz and 130 Hz are eliminated.

2.3 Transmission of Data between Wrist and the Walking Stick

Analog ECG waveform should transmit directly to the stick. But, we use the HP-3 transceiver to make the transmission of data in wireless mode from wrist to the stick. So, we can reduce the wiring in Walking stick in case of driving and walking and this will reduce the inconvenience between the wrist and the stick. The user can use the stick while driving a car and put his stick at the back seats without detaching any wire between his wrist and the stick. He also does not have to switch the device off. When the stick falls down, when the stick falls down, it won’t drag the user’s wrist to the ground.

After we browsed through the data sheet of the HP-3 transceiver, we discovered that the analog bandwidth of the transceiver pair is between 50 Hz and 28000 Hz. Normal ECG waveform has a frequency range between 50 Hz and 70 Hz. However, to cover all the possible scenarios like sleeping and fast walking, the lower and the upper cutoff frequencies were decided to be 0.5 Hz and 150 Hz. Since the range from 0.5 Hz and 50 Hz is outside the transmission bandwidth of the HP-3 transceiver, we thought about using mixer and oscillator to raise the lowest frequency of analog signal, which is 0.5 Hz, to 60 Hz. This would ensure correct transmission of the analog signal.

When we discussed this idea with our TA, we were introduced to the RS232 capability of PIC. Since the RS232 signal is digital, we can use the same transceiver to transmit a digital signal. We no longer have to worry about the lowest frequency of the analog signal along with the mixer and the oscillator.

2.4 Analog to digital conversion of ECG Signal

We are using the PIC for its RS232 feature, it would be best to use the Analog to Digital conversion feature with the PIC. We had to assume between using an 8-bit conversion or a 10-bit conversion. The 10-bit conversion would have a higher resolution to the digital ECG waveform. According to PIC-C, the final result was determined by the RS232 transmission.

2.5 Microcontroller

The microcontroller is a BASIC Stamp 2 which will run a real time program to constantly monitor the output of the A/D Converter, comparing current data samples against stored samples. It will include an algorithm to process both the amplitude and frequency of the heartbeat, to cover as many possible cases of a heart attack as reliably as possible. Once a heart attack is detected and confirmed, relevant data such as the time of occurrence will be collected, and a signal is sent to the Bluetooth/Wi-Fi Module to initiate the emergency dial up sequence via the cell phone.

When the microcontroller on the walking stick has received digital ECG data from the wrists, it will check for heart attack symptoms. We decided to use the heart attack algorithm developed by the previous project. We would like to indicate to the user his heart condition so that he can take proper action like slowing down or taking a rest before a heart attack really happens to him.

3. VERIFICATION /TESTING

3.1 Analog and Digital Conversion

To test the Analog to Digital conversion of the PIC, we would initialize the inputs with a known voltage level and check the corresponding digital values.

Here are a few samples of data.
Let u assume the known voltages as the 0.1V, 1V, 2V, 2.5V. The corresponding digital values are 5, 50, 100, 125.

3.2 RS232 of the PIC

We will test the RS232 pins are working or not by an oscilloscope. The samples or signals showed by the scope between 0V and 5V. The signal was showed in the oscilloscope is square wave because the signal considered as the digital.

We display data (digital) received in figure below by the Javelin microcontroller on the sheet and plotted the data points in Microsoft Excel.

3.3 Javelin’s Memory

We will be stored the ECG samples in the Javelin’s memory. The data will be modified in terms of amplitude, pulse rate and pulse width. The Javelin memory will store and programmed to run the amplitude, pulse rate and pulse width. It will provide the data positively. With the data modification, we can detect the heart attack of a human being.

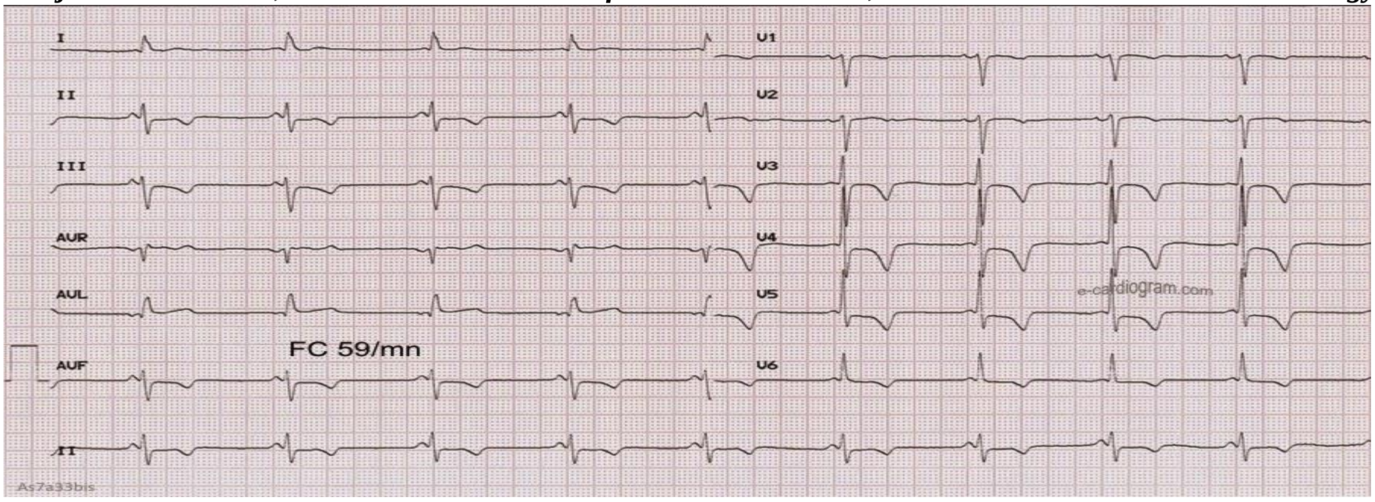

Fig. 3: A 63-year-old woman with 10 hours of chest pain and sweating

3.4 Caution level

To see and alert the user in case of heart attack, we consider the low-risk and the high-risk heart attacks with LED bulbs for caution. For those symptoms, we used the same modified data discussed above. We will display the caution in terms of the monitor and watched in the LEDs.

Alert level is considered into 4 to 6 and 7 to 9 conditions, as the 4 to 6 condition was monitored the low risk LED will shone. Similarly, the 7 to 9 condition was monitored the high risk will be shone.

3.5 Emergency Calling

Our project will execute emergency calling with just a Bluetooth module. Bluetooth communication is wireless. The user can put his cell phone anywhere he wants as long as it is within the range of the Bluetooth communication. We used the previously modified data to alert level to rise above 9. When the testing was done, we have programmed the phone number as 9003263127, which is the doctor phone's number. The Bluetooth/Wi-Fi module has started the process accordingly, my cell phone's screen showed the message "Dialing 9003263127" and the doctor phone range accordingly.

3.6 Power Consumption

We did not use a laptop as an intermediate between the Bluetooth module and the cell phone. The laptop is a very powerful machine that can store any programs like phone dialling program and can have hardware like Bluetooth and infra-red. Our project will never be marketable and useable if it required a laptop strapped to the walking stick. The elimination of the laptop greatly simplifies the hardware requirement of the previous project. This elimination will reduce power consumption and the project's executing time. Our project is completely portable. We have two main units. One main unit is worn on the wrists and the other is installed on the walking stick. The wrist unit with batteries is 3.4 oz and the stick unit with batteries is 6.4 oz.

4. ADVANTAGES

- We can save at least thousands of old people from death.
- Easily communicate with medical help centre by using GSM and GPS.
- Equipment is worn on a user's stick and wrist.

5. DISADVANTAGES

- Cost is high.
- Due to the lack of signal, call can't connect to the help centre.
- If medical centres don't react immediately, the maximum chance of such a person dies after several minutes.

6. MODIFICATION

- Using GSM there is a chance of lack of signal.
- Including with GSM, GPS is more efficient in performance. Using GPS we can locate the location of such person.

7. CONCLUSION

The Walking Stick with Heart Attack Detection functions as designed overall. ECG waves properly collected from analog circuitry unit. The transmitting and receiving of A/D converted waveform performed as expected. The most significant improvement was the emergency calling part. We successfully deleted the laptop between the Bluetooth module and the mobile phone when activating emergency calling. Possible future improvements are better packaging of the wrist circuitry, lower power consumption for main units, more common media rather than just walking sticks, the shorter delay between heart attack detection and emergency calling via cell phone, and more accurate and faster heart attack algorithm

8. REFERENCES

[1] ECG Library. 2002.
[2] Wikipedia, the Free Encyclopedia, "RS-232", April 30, 2005

[3] Hwang, Limsui, Zhao, “Wireless Heart Attack Detector with GPS”, ECE445.University Illinois, Fall 2004
[4] https://courses.ece.uiuc.edu/ece445/cgi-bin/view_project.pl?fall2004_24

APPENDIX

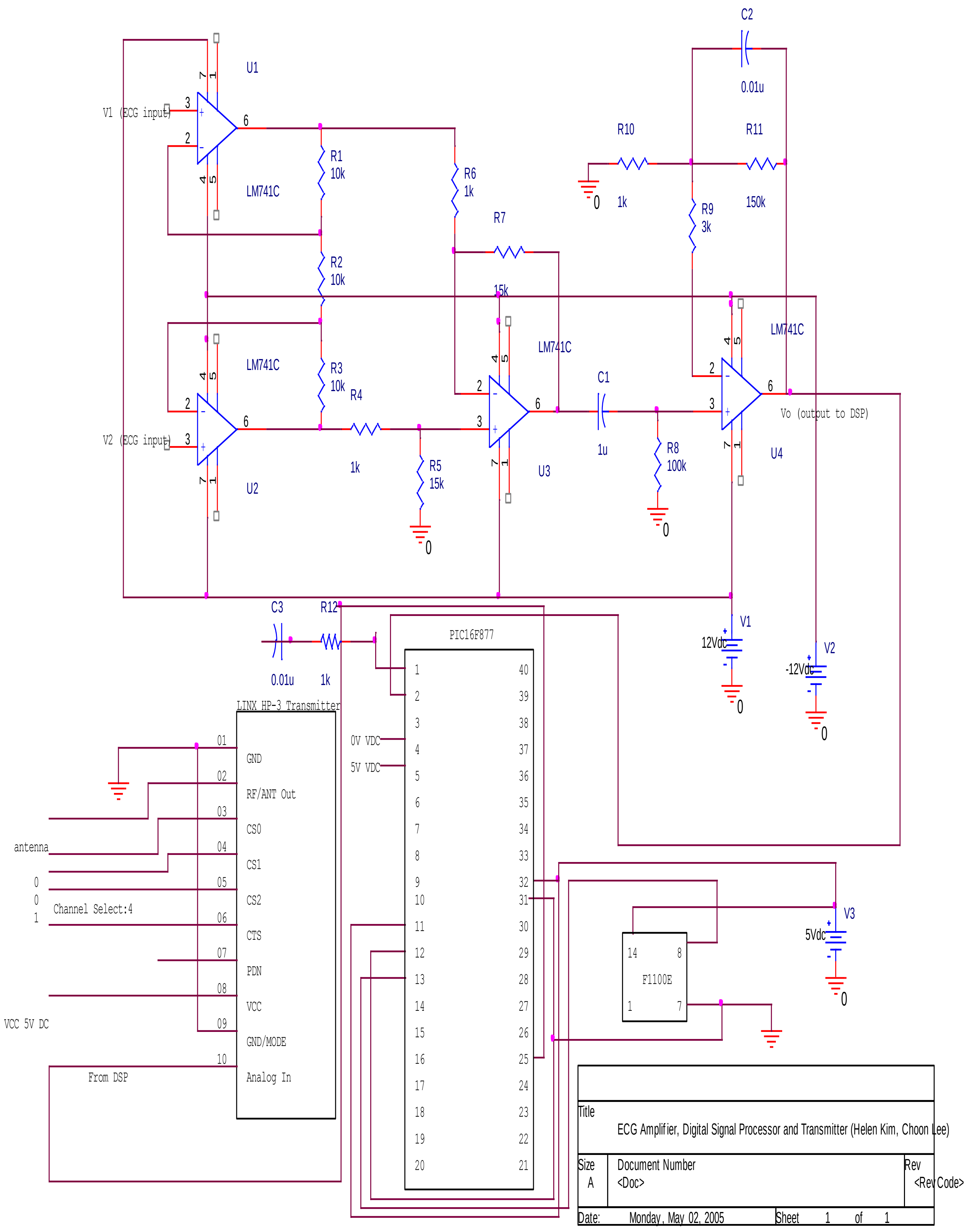

Fig. 4: ECG Analog Circuit, PIC16F877, HP-3 Transmitter

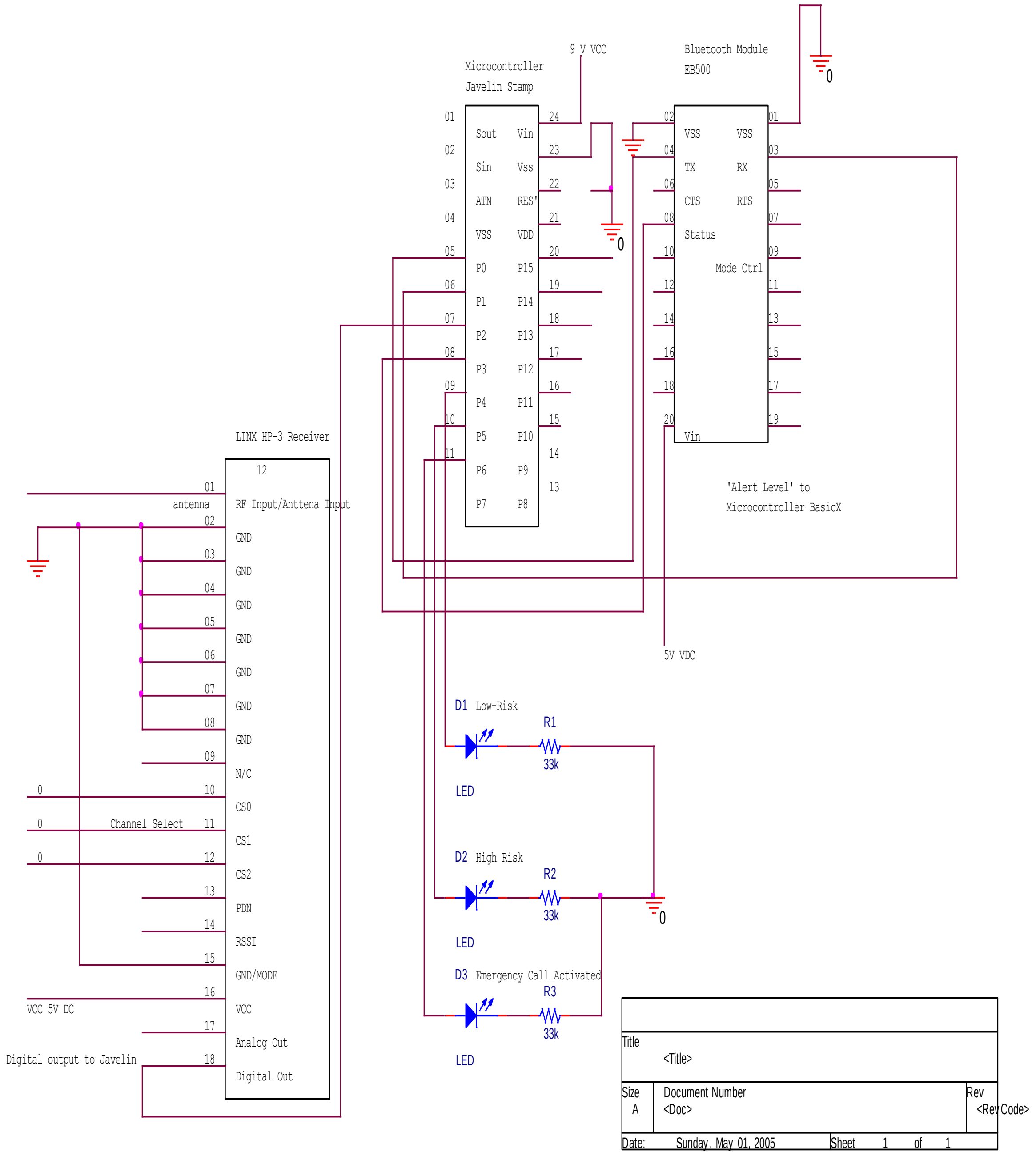

Fig. 5: HP-3 Receiver, Javelin Microcontroller, EB500 Bluetooth Module

PIC16F877 Programming Code:

```
#include <16F877.h>   // <-- This is the type of PIC you're using.
          //   Other supported PICs are in:
          //   C:\Program Files\PICC\Devices

#device *=16 ADC=8   // Use 16-bit pointers, use 8-bit ADC

#fuses HS        // You may or may not want some of these ....
#fuses NOWDT
#fuses NOPROTECT
#fuses NOLVP
#fuses NODEBUG
#fuses NOPUT
#fuses NOBROWNOUT

#use delay(clock=20000000)
#use rs232(baud=9600, xmit = PIN_C6, bits = 8)   //8 bits transmission

#include <stddef.h>
#include <string.h>
```

```
#include <stdlib.h>
#include "Include\Compiler.h"
#include "Include\Globals.h"

int8 value;

void main()
        {
                setup_adc_ports(RA0_ANALOG_RA3_RA2_REF); // A0 Ref=A2,A3
                setup_adc(ADC_CLOCK_INTERNAL);
                set_adc_channel( 0 );
                delay_us(100);

                while (TRUE)      //Always true to sample continuously
                {
                        value= read_adc();
                        putc(value);
                        delay_us(2500);    //This delay will make a 400Hz sampling frequency
                }

        }    // END MAIN ROUTINE
```

Javelin Stamp Programming Code:

```
import stamp.core.*;
import java.io.*;

//Most codes were created by previous project, especially heart attack algorithm
//Codes created by us are those about receiving digital ECG data, alert level, LEDs and emergency calling, //EB500

public class ReceiverProcessorEB500 {

 // SAMPLING FREQUENCY DEPENDENT CONSTANTS
   /* OFFSET adaptibility:
   1) Ath must be relative to baseline
   2) Wth must be relative to baseline
   3) Take account into the case where PRindex1 starts at the peak, would glitch occur?
   */
 final static int SAMPLING_FREQ = 400;  // Current Sampling Frequency in Hz.
 final static int ARRAY_SIZE = 2048;
 final static int MAX_VALUE  = 65535 >> 1;
                  // Frequency of 600 Hz
 final static int Ath = 200;       // Minimum Threshold for value, relative to BASELINE
 final static int Pmin = (60*SAMPLING_FREQ)/200;    // Absolute Min. period (pulse rate 200 bpm)
 final static int Pmax = (60*SAMPLING_FREQ)/35;    // Absolute Max. period (pulse rate 35 bpm)
 final static int Wmax_th = 20;   // Maximum allowable width of W at Wth
 final static int AlertThreshold = 10; // How many alerts to take before Calling EMS
 final static int AdecLimit = 50;  // peaks should not differ by more than 50%
 public static int Wth = 200;      // Level at which only the QRS pulse should exceed


 final static int Alert_Level_Low = CPU.pin4;
 final static int Alert_Level_High = CPU.pin5;
 final static int Emergency_call = CPU.pin6;


 final static int Receiver_RX_PIN = CPU.pin2; //The receiving port of RS232 from
                                   //the HP-3 Receiver


 final static int EB500_RX_PIN = CPU.pin0; //The receving port of RS232 from EB500
 final static int EB500_TX_PIN = CPU.pin1; //The transmitting port of RS232 to EB500
 final static int STS_PIN = CPU.pin3;    //Status of EB500, not connected or connected

static Uart Receiver = new Uart(Uart.dirReceive, Receiver_RX_PIN,
                Uart.dontInvert, Uart.speed9600,
                Uart.stop1);

 static Uart EB500RX = new Uart(Uart.dirReceive, EB500_RX_PIN,
                Uart.dontInvert, Uart.speed9600,
                Uart.stop1);

 static Uart EB500TX = new Uart(Uart.dirTransmit, EB500_TX_PIN,
                Uart.dontInvert, Uart.speed9600,
                Uart.stop1);

 //Variables of EB500
 static char d;
 static char m;
 static String b;

 //Variable of HP-3 Receiver
 public static int g;

 //Variables of Processor
 static StringBuffer buffer = new StringBuffer(80);
 static char c;

 public static int AlertLevel;
 public static int i;
 public static int temp;
 public static StringBuffer sbuf = new StringBuffer(8);
 public static String str;

 public static int [] sampleArray = new int[ARRAY_SIZE];

 public static void main()
 {
  System.out.println("================================");
  System.out.println("ECG Data Receiving, ECG Algorithm Running, 911 Call Ready");
  System.out.println("================================");

  // Main Instantiations
  int arrayCount = 0;
  AlertLevel = 0;
  int windowCount = 0;


  Terminal terminal = new Terminal();


  // Main Processing Loop begins ======================================
  arrayCount = 0;

  while(true)
  {
   // Delay to have a 250Hz cycle, which is a 4ms period

   addSample(arrayCount, sampleArray);

   arrayCount++;
     // ECG Window Check
   if (arrayCount >= ARRAY_SIZE)
   {
   arrayCount = 0; // reset arrayCount


   AlertLevel += check_filled_ECG(sampleArray);

   System.out.print("\nAlert Level is now: ");
   System.out.print(AlertLevel);
   if (windowCount > 30) // reset AlertLevel after 2 min
   {
   AlertLevel = 0;
   }

  windowCount++;
  } // ECG Window Check

  //Displaying Alert Level LEDs
  if (AlertLevel >= 7) //High Risk
  { CPU.writePin(CPU.pin5,true);
   CPU.writePin(CPU.pin4,false);
  }
  else        //Low Risk
  { CPU.writePin(CPU.pin4,true);
   CPU.writePin(CPU.pin5,false);
  }

  if (AlertLevel >= AlertThreshold)
  {
   System.out.println("Alert, Emergency Dial-up");
   CPU.writePin(CPU.pin6,true);
   Call911();
   AlertLevel = 0;
  }
  else
  {CPU.writePin(CPU.pin6,false);}

 } // end while(1) Main Processing Loop


 } // main ends

 // This function calls 911 via Bluetooth EB500
 static void Call911()
 {

  CPU.delay(25000);
  System.out.println("Emergency Call Started.\r");


  System.out.println("EB500's Info Started.\r");
  EB500TX.sendString("ver all\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("EB500's Info Completed.\r");


  System.out.println("Listing Visible Devices Started.\r");
  EB500TX.sendString("lst visible\r");
  CPU.delay(25000);
  getMessage();
  System.out.println("Listing Visible Devices Completed.\r");


  System.out.println("Getting the address of EB500 Started.\r");
  EB500TX.sendString("get addr\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Getting the address of EB500 Completed.\r");



  System.out.println("Setting the security mode of EB500 Started.\r");
  EB500TX.sendString("set security open\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Setting the security mode of EB500 Completed.\r");

  System.out.println("Setting the visible mode of EB500 Started.\r");
  EB500TX.sendString("set visible on\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Getting the visible mode of EB500 Completed.\r");

  System.out.println("Getting the security mode of EB500 Started.\r");
  EB500TX.sendString("get security\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Getting the security mode of EB500 Completed.\r");

  System.out.println("Getting the visible mode of EB500 Started.\r");
  EB500TX.sendString("get visible\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Getting the visible mode of EB500 Completed.\r");

  System.out.println("Setting the passkey of EB500 Started.\r");
  EB500TX.sendString("set passkey 0000\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Setting the passkey of EB500 Completed.\r");

  System.out.println("Setting the connectable mode of EB500 Started.\r");
  EB500TX.sendString("set connectable on\r");
  CPU.delay(5000);
  getMessage();
  System.out.println("Setting the connectable mode of EB500 Completed.\r");

  do{
  System.out.println("Connecting to cellphone Started.\r");
  EB500TX.sendString("con 00:0A:D9:84:F5:7B\r");//IEEE Address of our ERiCsson T610
  CPU.delay(5000);
  getMessage();
  System.out.println("Connecting to cellphone Completed.\r");
  if (CPU.readPin(CPU.pin3) == true)//Checking the connection status of the bluetooth module
  m='y';
  else
  m='n';
  System.out.println("\r");
  } while (m!= 'y');//Continue making connection until connection is successfully established

  System.out.println("Dialing Started.\r");
  EB500TX.sendString("ATD");      //AT Commands given by the Austrian Guy
  CPU.delay(5000);
  EB500TX.sendString("2173335257");
  CPU.delay(5000);
  EB500TX.sendString(";");
  CPU.delay(5000);
  EB500TX.sendString("\r");
  CPU.delay(5000);
  System.out.println("Dialing Completed.\r");

  System.out.println("Emergency Call Completed.");

 }

 static void getMessage()
 {
  while (EB500RX.byteAvailable() ==true)
  {
  d = (char) EB500RX.receiveByte();
```

```
   if (d != ' ')
   {
     System.out.print(d);
   }
  }

  System.out.print('\n');
 }


//=====================================================================
   static boolean addSample(int arrayCount, int [] samples)
   {
       boolean fail = false;

       g = Receiver.receiveByte();
       temp = g;

       samples[arrayCount] = temp;

       return fail;             // Refer to P.7 of manual for I/O.
   }
//=====================================================================

//=====================================================================
static int check_filled_ECG(int [] samples)
{
 // Array Parsing Flags
   int baseline = 0;

 // Amplitude flags
 int AmpPeak = 0;
 int Alert = 0;
 int AmpVar;

 // PulseRate flags
 int period_sample;
 boolean found_PR_peak1 = false;
 boolean found_PR_peak2 = false;
 int PRindex1 = 0;
 int PRindex2 = 0;
   boolean repolarized = false;

 // PulseWidth flags
 boolean found_PW_begin = false;
 boolean found_PW_end = false;
// Wth = Amax/4; // want Wth to be 25% Amax
 int W_sample;
 int PW_index_begin = 0;
 int PW_index_end  = 0;
 int BPM = 0;

    System.out.print("\n~4 sec ECG Check Window Filled....\n");  //DEBUG
    System.out.print("===== Main Parsing Loop Begins =====\n");  //DEBUG
    // >>> --- Establishing Baseline --------
    for (i = 0; i < ARRAY_SIZE; i++)
    {
        if (i == 0) baseline = samples[i];
        else
        {
            baseline = (baseline + samples[i]) / 2;  // baseline is always the average of all the data points
        }
    }

    System.out.print("Baseline = "); // DEBUG
    System.out.print(baseline);
    // <<< --- Establishing Baseline end -----


 // >>> ========= Main Parsing Loop Begins ========
 for (i = 0; i < ARRAY_SIZE; i++)
 {
  // >>> --- Checking PulseRate -----
  if (!found_PR_peak1) // if first QRS peak is not found yet
  {
   if (samples[i] > samples[PRindex1])  // amplitude relative to baseline is not important here
   {
    PRindex1 = i;
          System.out.print("\nPulseRate peak 1 (Abs) increased to "); // DEBUG
          System.out.print(samples[PRindex1]);
          System.out.print("\n @t= "); // DEBUG
          System.out.print(PRindex1);
   }
   else if (((samples[i] - baseline) < ((samples[PRindex1] / 2) - baseline)) && ((samples[PRindex1] - baseline ) > Ath))
       // Amp relative to baseline is key here
   { // if current sample is lower than 50% of the first peak relative to baseline, AND it breaks the Ath minimum limit
    found_PR_peak1 = true; // then first peak is found
        System.out.print("\nPulseRate peak 1 is Found: "); // DEBUG
        System.out.print(samples[PRindex1]);
          System.out.print("\n @t= "); // DEBUG
          System.out.print(PRindex1);
   }
  }
  else if (!found_PR_peak2)
  {
     // wait for repolarization, don't search for the second peak
     // until samples[i] has returned to baseline
        if ((samples[i] >= baseline) || repolarized)
        {
        repolarized = true;    // so that we know the ECG has repolarized
            if (samples[i] > samples[PRindex2]) // amplitude relative to baseline is not important here
            {
                PRindex2 = i;
                System.out.print(" \nPulseRate peak 2 (Abs) increased to "); //DEBUG
                System.out.print(samples[PRindex2]);
                System.out.print("\n @t= "); // DEBUG
                System.out.print(PRindex2);
            }
            else if ((samples[i] - baseline) < ((samples[PRindex2]/ 2) - baseline) && ((samples[PRindex2] - baseline) > Ath))
                        // Amp relative to baseline is key here
            { // if current sample is lower than 50% of the second peak relative to baseline, AND it breaks the Ath minimum limit
                found_PR_peak2 = true; // then second peak is found
                System.out.print("\nPulseRate peak 2 is Found: "); //DEBUG
                System.out.print(samples[PRindex2]);
                System.out.print("\n @t= "); // DEBUG
                System.out.print(PRindex2);
            }
        } // if samples[i] > baseline
  }
  // <<<--- Finished Checking PulseRate -----

  // >>> --- Checking PulseWidth -----
  if (((samples[i] - baseline) > Wth) && (!found_PW_begin))
  {
   PW_index_begin = i;
   found_PW_begin = true;
        System.out.print("\nPulseWidth begin ");  //DEBUG
        System.out.print(PW_index_begin);
        System.out.print(" Amp = ");
        System.out.print(samples[i]);
  }
  else if (((samples[i] - baseline) < Wth) && (!found_PW_end) && found_PW_begin)
```

```
  {
   PW_index_end = i;
   found_PW_end = true;
        System.out.print("\nPulseWidth ends ");  //DEBUG
        System.out.print(PW_index_end);
        System.out.print(" Amp = ");
        System.out.print(samples[i]);
  }
  // <<< --- Finished Checking PulseWidth -----

 }
 // <<<========== Main Parsing Loop ends ==============


    System.out.print("\n===== Main Parsing Loop ends =====\n");  //DEBUG


 // >>>-------- Setting Alert for Amplitude check Begins -------
 // Set historically max and min values and find % difference
 // See which of the two peaks are greater
    if (samples[PRindex1] > samples[PRindex2])
    {
       AmpPeak = samples[PRindex1] - baseline;
    }
    else
    {
       AmpPeak = samples[PRindex2] - baseline;
    }

//    if (AmpPeak > Amax) Amax = AmpPeak;
//      if (AmpPeak < Amin) Amin = AmpPeak;

    System.out.print("AmpPeak is (baseline adjusted) ");  //DEBUG
    System.out.print(AmpPeak);
//     System.out.print(" Amin is (baseline adjusted) ");
//     System.out.print(Amin);

//  AmpVar = ((Amax - Amin)*100)/Amax;
//    System.out.print(" \n%AmpChange is ");
//     System.out.print(AmpVar);
 if (/*(AmpVar > AdecLimit) || */ (AmpPeak < Ath)) // If percent decrease exceeds limit
 {
  Alert++;
    System.out.println("\nAlert Raised from Amplitude Check!"); // DEBUG
 }
   else
   {
   System.out.println("\nAmplitude Check Okay! ( > 200 relative to baseline)");
   }
 // <<<-------- Setting Alert for Amplitude check Ends-------


 // >>>-------- Setting Alert for PulseRate check Begins-------
 // Now peak1 and peak2 are set
 period_sample = PRindex2 - PRindex1;
   System.out.print("\nPeriod_sample is (Pmax=");
   System.out.print(Pmax);
   System.out.print(" Pmin=");
   System.out.print(Pmin);
   System.out.print("): ");
 System.out.print(period_sample);
   BPM = 10000/( (period_sample*(10000/SAMPLING_FREQ)) /60 );
   System.out.print("\nTranslating to ");
   System.out.print(BPM);
 System.out.print(" BPM (Beats Per Minute) at a Sampling Frequency of ");
 System.out.print(SAMPLING_FREQ);
```

```
  System.out.print(" Hz");

 // if !(Pmin < P < Pmax) Alert the main loop!
 if(period_sample < Pmin || period_sample > Pmax)
 {
  Alert++;
   System.out.println("\nAlert Raised from PulseRate Check!"); // DEBUG
 }
 // <<<-------- Setting Alert for PulseRate check Ends-------


 // >>>-------- Setting Alert for PulseWidth check Begins-------
 W_sample = PW_index_end - PW_index_begin;

 System.out.print("\nW_sample is (less than 20): ");
  System.out.print(W_sample);
 if (W_sample > Wmax_th)
 {
  Alert++;
   System.out.println("\nAlert Raised from PulseWidth Check!"); //DEBUG
 }
 // <<<-------- Setting Alert for PulseWidth check Ends-------


 System.out.print("\n===== End ECG Window Check =====\n");  //DEBUG
 return Alert;
}

} // class ReceiverProcessorEB500 ends
```