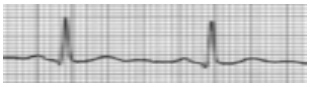


Bangladesh University of Engineering and Technology

CSE 408

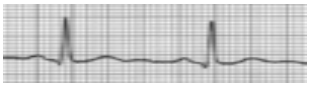
Computer Interfacing Sessional

"Wireless Electrocardiogram(ECG) System"

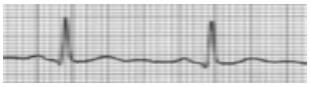
Date: 15th November, 2008

Group members:

1. Jakaria Mahmood (0405068)
2. Amit Kumar Dutta (0405071)
3. Fahian Ahmed (0405072)
4. Sayeed Safayet Alam (0405088)
5. Shahadat Hossain (0405089)



INDEX



Introduction

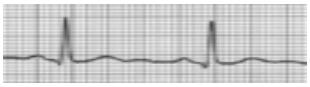
An **electrocardiogram** (**ECG** or **EKG**) is a medical diagnostic procedure produced by an electrocardiograph, which records the electrical activity of the heart over time. Its name is made of different parts: *electro*, because it is related to electrical activity, *cardio* for heart, *gram*, a Greek root meaning "to write".

Electrical impulses in the heart originate in the sinoatrial node and travel through the heart muscle where they impart electrical initiation of systole or contraction of the heart. The electrical waves can be measured at selectively placed electrodes (electrical contacts) on the skin. Electrodes on different sides of the heart measure the activity of different parts of the heart muscle. An EKG displays the voltage between pairs of these electrodes, and the muscle activity that they measure, from different directions, also understood as vectors. This display indicates the overall rhythm of the heart and weaknesses in different parts of the heart muscle. It is the best way to measure and diagnose abnormal rhythms of the heart, particularly abnormal rhythms caused by damage to the conductive tissue that carries electrical signals, or abnormal rhythms caused by levels of dissolved salts (electrolytes), such as potassium, that are too high or low.

Conventional ECG machines found at hospital & diagnostic centers are heavy weighted and need expert technician to operate. These machines consist of 12 electrodes which are attached to 12 different positions of body skin to take the heart signal. Though ECG with these machines are much accurate and very precise they have some problems like their availability, costs, technician's availability etc. heart patients need fast treatment and care. So, if it would possible to predict any possible malfunction of heart by a less sophisticated but light weighted, highly available and easy to operate heart signal measuring device then it could save many people from future heart malfunction by prediction any abnormality of heart in advance.

Fig: Conventional ECG machine

Considering these issues we attempt to make a smart ECG system which consists of 2 electrodes and a small circuit consisting of two 12 Volt power supply having features of wirelessly transmit heart signal of patients to nearby computer or cell phone with in 100ft radio frequency range.



Problem Description:

Electrical Voltage induced at different part of body by heart is about 1.5 mV at peak. Capturing this small signal introduce a lot of noise. Again heart signal frequency in bounded to a certain band. If we filter signal that band then we are likely to get exact heart signal. Modern ECG monitors offer multiple filters for signal processing.

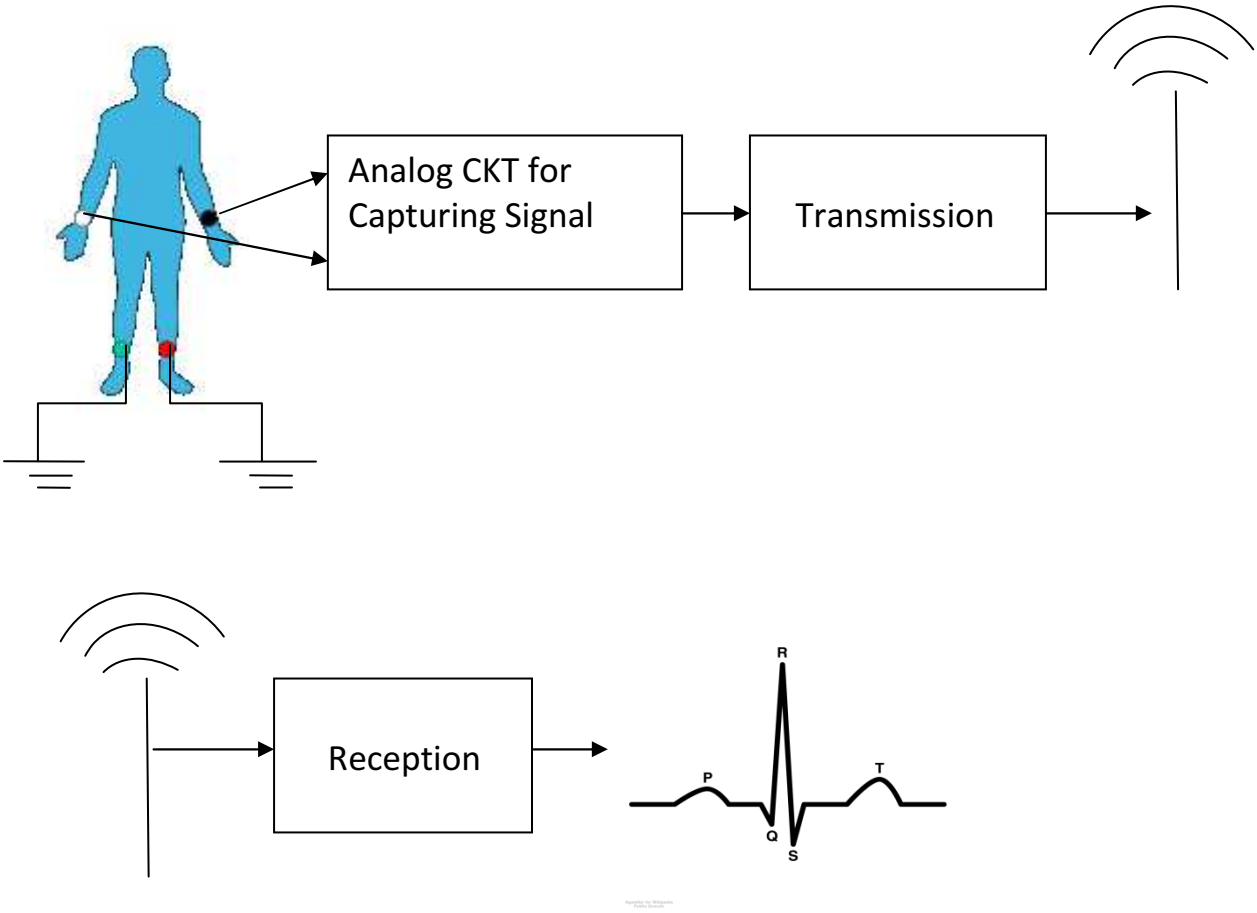
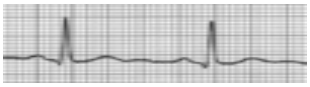


Fig: Display

We also need an amplifier to amplify the signal so that it can be easily visualized. For this purpose we used instrumentation amplifier which is made up of Op Amp and resistance.



System Block Diagram

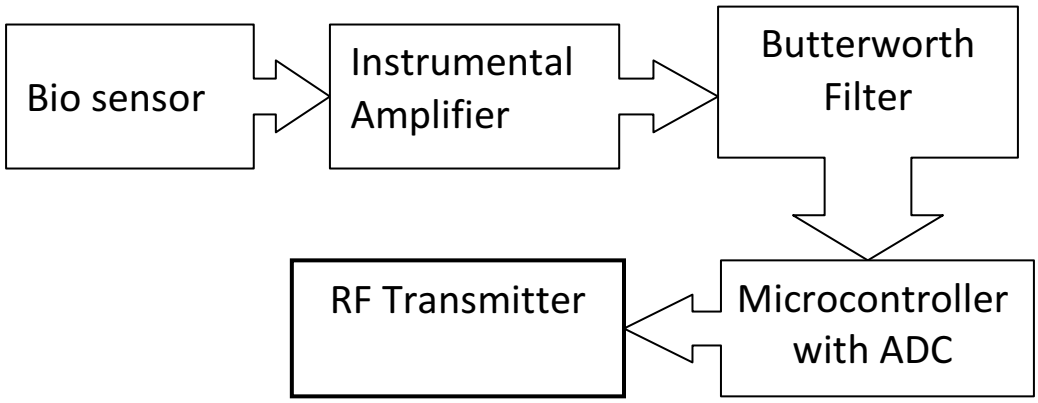


Fig: Transmission Side

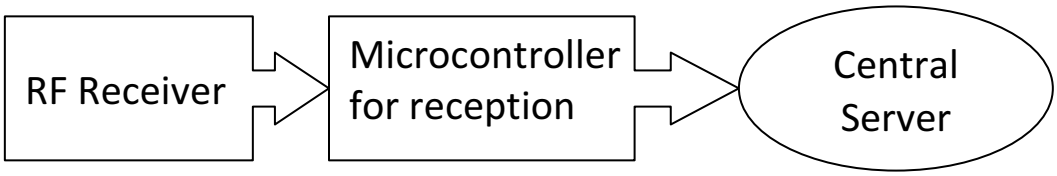


Fig: Reception Side

Analog and transmission Section:

Hardware description:

Bio Sensors: We used conventional ECG clips that are widely used in Hospital & diagnostic centers for ECG. These sensors are locally available at any renowned medical instrument stores. To get more conductivity we used Body Line gel.

Figure: **Bio sensor Clip**

Figure: Bio sensor Gel

Analog Circuit: Our analog circuit made up several complex analog circuit elements like Instrumental amplifier, 3rd order Butterworth filter.

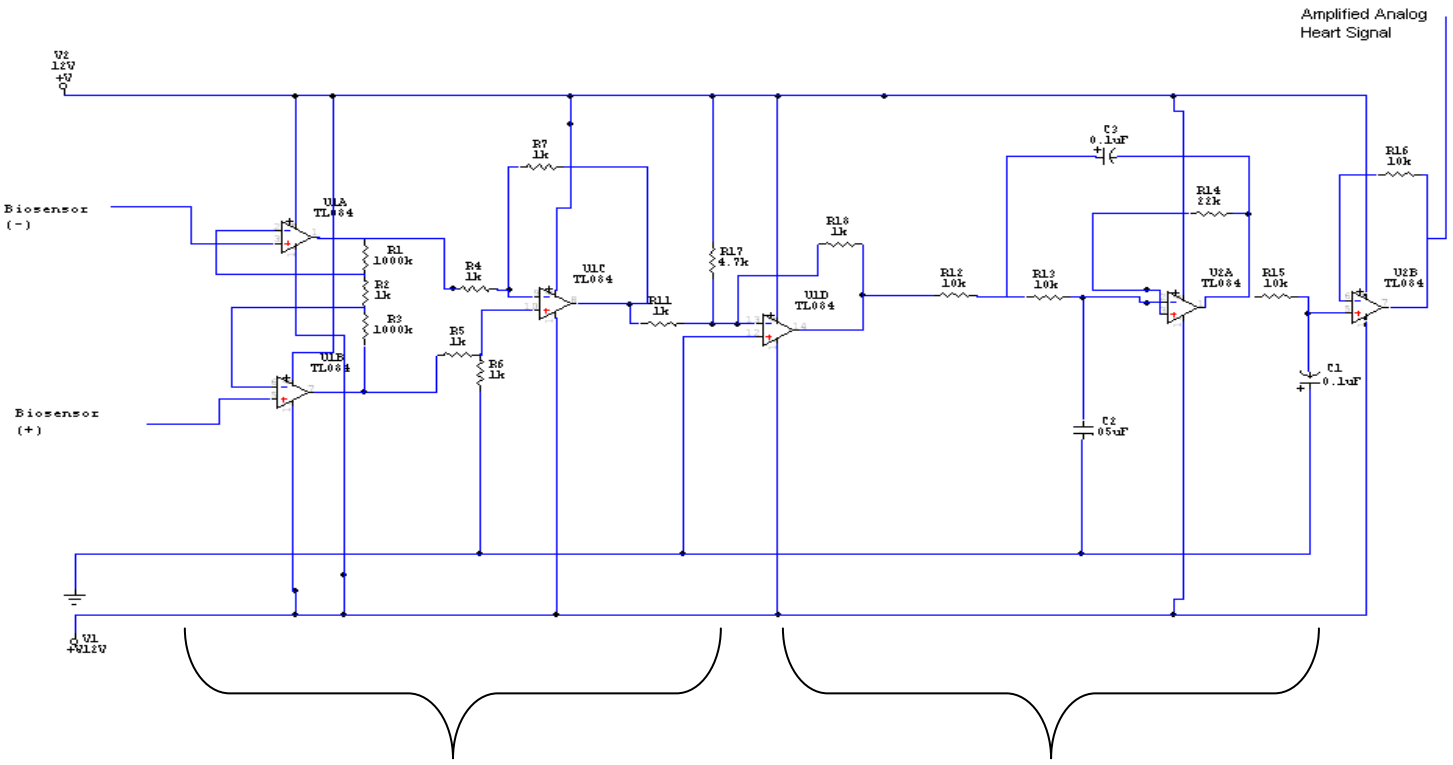


Fig: Instrumental Amplifier **Fig: 3rd Order Butterworth Filter**

Fig: Analog Circuit

In the analog circuit we have two parts. One is the amplifier part and the other is the filter part. Both are described below:

Amplifier part:

In the amplifier part we used differential amplifier or instrumentation amplifier. A differential amplifier usually takes 2 inputs, subtracts it and then amplifies it.

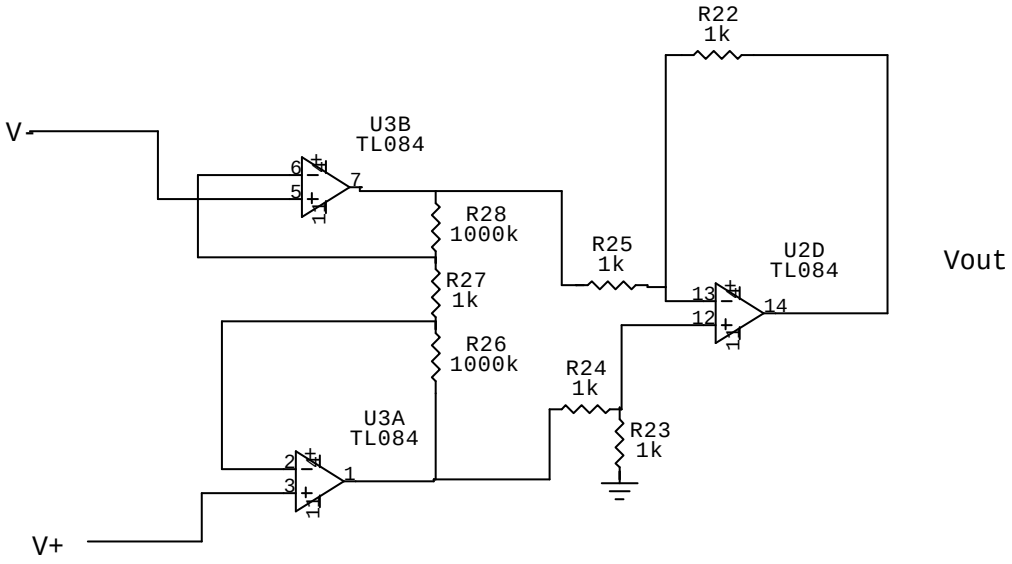


Fig: Instrumentation amplifier

In our amplifier the formula is

$$V_{out} = (1+2/a)\,(V^{+} - V)$$

Where a = (1k/1000k) = .001.

So it amplifies (1+2/.001) = 2001 times. Because there is a phase shifting in both hand signal. So the subtracted value gets doubled. So, V+-(V-) =2V.

Filter part:

In our filter part we used a Low pass 3rd order Butterworth Filter.

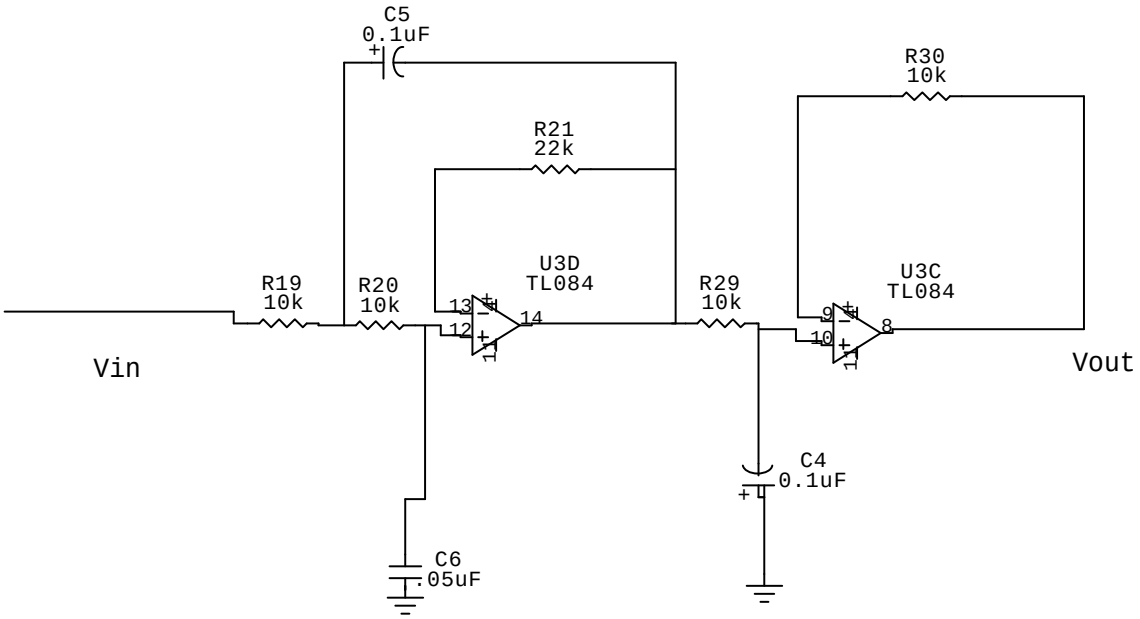


Fig: 3rd order Butterworth Filter.

The Butterworth filter of 3rd order gives 60dB decade. For heart signal the frequency stays from 50-300Hz. So we set the parameter as

R=10k

C=0.1uF.

So,

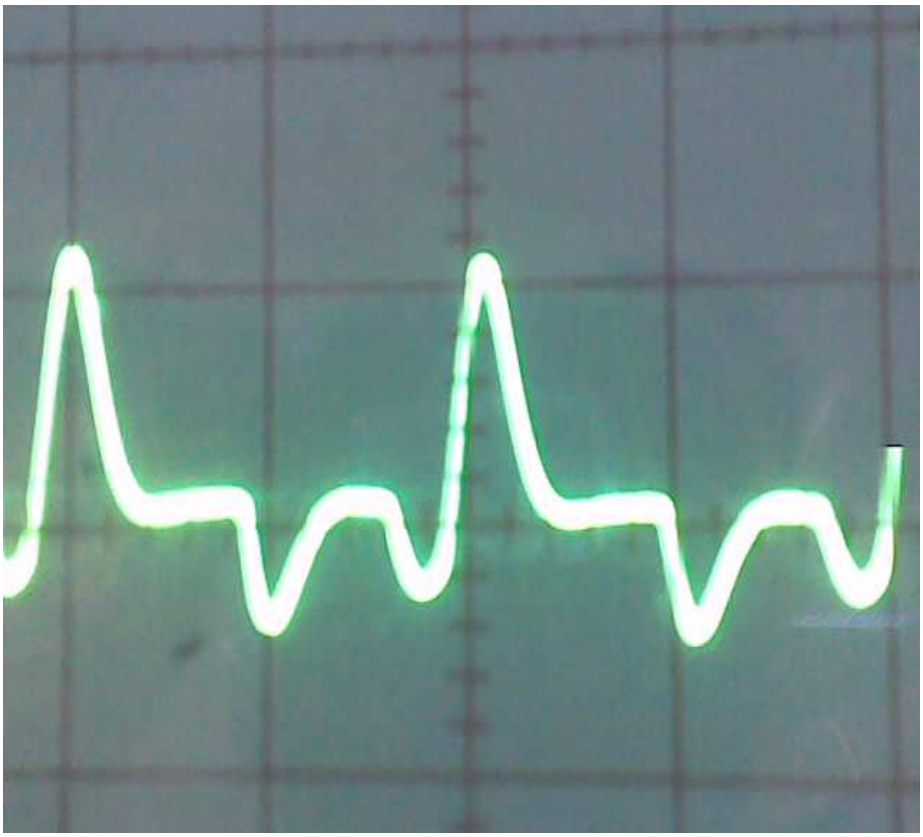
We get the cutoff frequency f.

f= 1/ (2πRC).

So, f= 1/ (2π* 10k * .1uF)

= 159.15Hz.

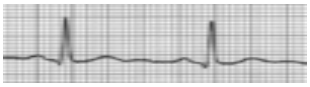
So, the filter keeps its range from 0-159.15Hz. Other Noises are cut down.



Amplified and filtered heart signal found in oscilloscope using our analog circuit. This photo was taken after the end of 8th week of running term. (Middle of the September 2008)

Converting Analog Heart signal to digital and Wireless Transmission

The ECG sensing circuit gives amplified and filtered heart signal in analog form. To transmit this signal, we had 2 options. First one is to transmit this analog signal in this analog form. When we first proposed our system block diagram, we decided to transmit the signal with some chip which can perform analog transmission and reception. But, later we found it very difficult to get such chip or wireless module. So, we decided to use an A/D converter to transform the signal into digital and then transmit the output bytes. We also decided to transmit these byes serially. So, we decided to use a microcontroller in the middle. There were plenty of choices for this, we choose ATMega8 manufactured by Atmel which has built in A/D converter and USART support. Using this microcontroller there was no need to use external ADC and serial communication devices so that PCB takes less space, easier to create programs-it saves both time and money. One thing is very important here. The transmission should be real time. So, we were looking for a wireless module which can be connected with USART of microcontroller and can transmit with a decent speed that fulfills our requirement. We got RCT-433-ASBR from Radiotronix™ which can perform this job.



Converting Analog Heart signal to Digital

Atmega8 microcontroller has up to 6 multiplexed single ended input channels with 10 bit resolution and ±2 LSB absolute accuracy. The A/D converter in Atmega8 is designed to work between ground and specified voltage reference. References may be taken from AREF pin which is connected with external voltage source.ADC unit is powered with separate power supply pins AVCC with AGND, but AVCC must not differ ±0.3 of VCC. Also ADC unit can have different voltage input sources selectable in ADMUX register. All ADC inputs are multiplexed via multiplexer. Each channel can be selected by changing 4 bits in ADMUX register. ADC unit can operate in two modes: Single conversion and free running modes. The ADC unit converts analog input at the selected channel to 10 bit digital values which is stored ad ADC internal register. After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result Registers (ADCL, ADCH). For single ended conversion, the result is

$$\text{ADC} = \frac{(Input\ Voltage) * 1024}{(Reference\ Voltage)}$$

To serve our purpose we apply analog heart signal to the ADC0 pin of the microcontroller. We operate ADC unit in single conversion mode in continuous manner. Once initialized this mode takes 13 ADC cycles for single conversion. In this mode ADC data register has to be read before new value is written. With the microcontroller operating with 1MHz clock and input clock of ADC division factor as 4, the single conversion mode took 13*4 =52ns for each conversion.

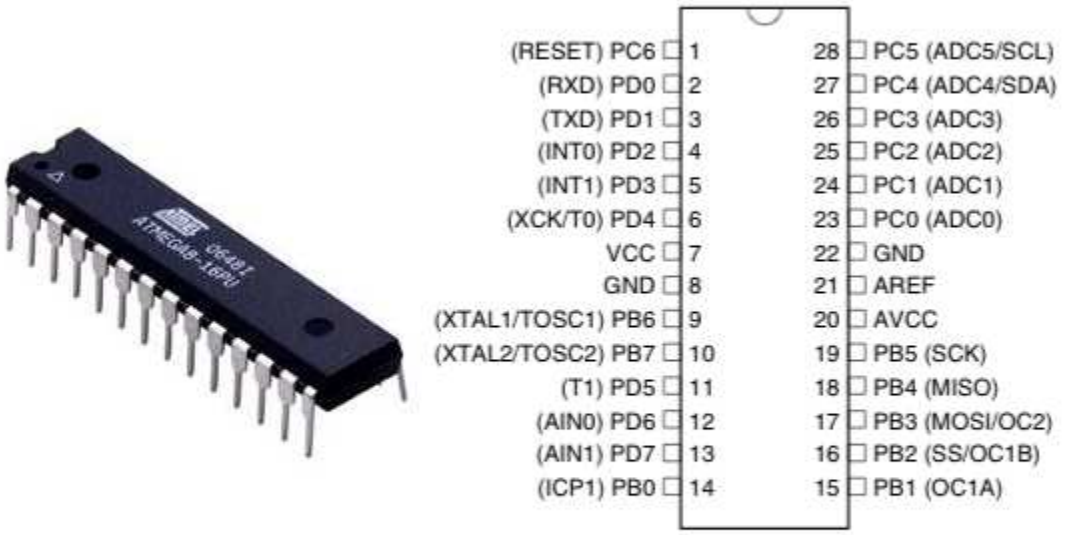


Figure : Atmega8 and it's DIP package

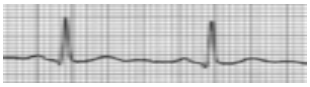
UART & Wireless Transmission

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is powerful and useful interface for serial communication. Microcontroller reads the analog signal periodically, converts it to digital by ADC and serially transmits the data using UART at pin 3. So, we connected the data pin of our Wireless Transmitter at pin 3 of microcontroller to transmit the data wirelessly.

We used RCT-433-ASBR for Wireless Transmission. The RCT-433-ASBR is ideal for remote control applications where low cost and longer range is required. The transmitter operates from a 1.5-12V supply, making it ideal for battery-powered applications. The transmitter employs a SAW-stabilized oscillator, ensuring accurate frequency control for best range performance.

Features of RCT-433-ASBR:

- 315/418/433.92 MHz Versions
- Low Cost



- 1.5-15V operation
- 5 ma Current Consumption at 3 V
- Small size
- 0 dBm output power at 3 V
- 4800 baud operation

Applications:

- Remote Keyless Entry (RKE)
- Remote Lighting Controls
- On-site Paging
- Asset Tracking
- Wireless Alarm and Security Systems
- Long Range RFID
- Automated Resource Management

Seeing these features and applications we have decided to use this Wireless transmitter. It's voltage level was s also very flexible. So, we found no problem while using it.

At microcontroller, UART transmitter was enabled by setting TXEN bit of UCSRB register.

For any serial communication we had to set up a baud rate which satisfies both end. We had to find a baud rate which is available at microcontroller and also supported by Wireless Transmitter. For Asynchronous normal mode the equation for calculating baud rate is:

$$\mathbf{BaudRate = f\_clk/(16(UBRR+1))}$$

$$\mathbf{UBRR = ((f\_clk/(BaudRate*16))-1)}$$

Figure : Wireless Transmitter (RCT-433-ASBR)

We decided to use baud rate 4800 bps which is the highest supported baud rate of our wireless transmitter module. As our microcontroller was operating at 1MHz clock frequency, calculated value for UBRR value was 12.

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. For our communication we choose 1 start bit, 8 data bits, 2 stop bits and no parity bits.

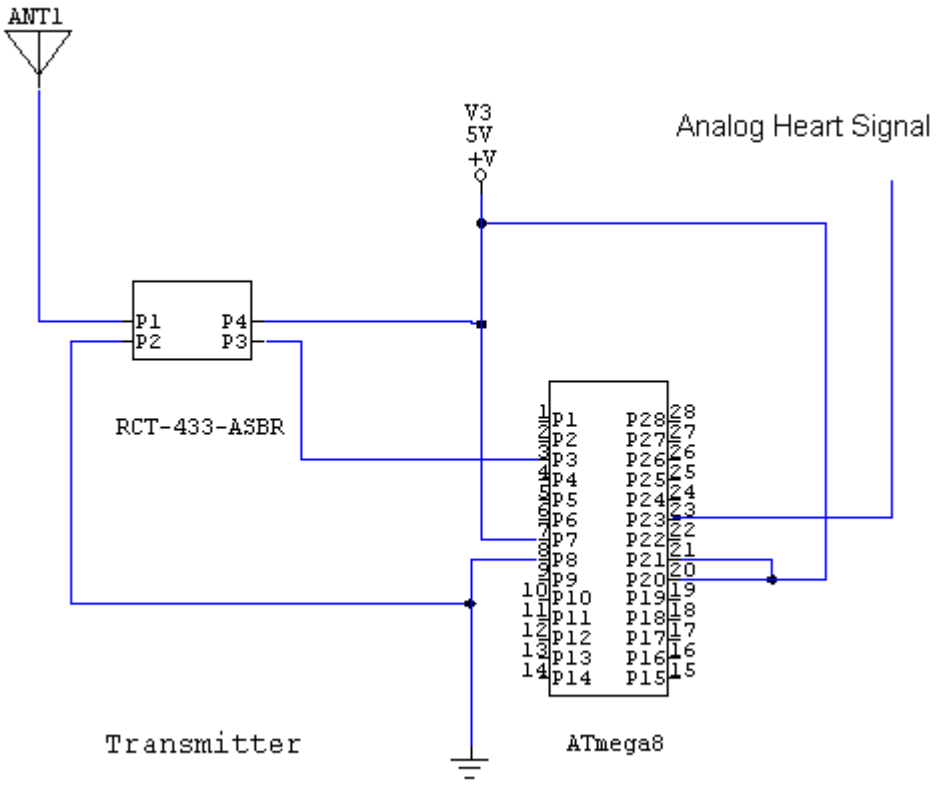


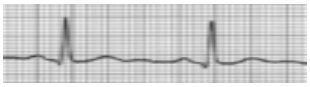
Figure : A/D conversion and Wireless Transmission

Receiving transmitted Data and Interfacing with Computer

RCR-433-ASBR is the ASK/OOK Receiver for RCT-433-ASBR. The RCR-433-ASBR is ideal for long range remote controlled applications where cost is a primary concern. The receiver module requires no external RF components except for the antenna. It generates virtually no emissions, making FCC and ETSI approvals easy. The super-regenerative design exhibits exceptional sensitivity at a very low cost. A SAW filter can be added to the antenna input to improve selectivity for applications that require robust performance.

Features of RCR-433-ASBR:

- Low Cost
- 3MHz receiving bandwidth – works with any LC or SAW based transmitter
- 5V operation



- 4.5mA current drain
- No External Parts are required
- Small Size: 1.76” x .43”
- 4800 baud operation

For these reasons, we found it very easy to use this Wireless Receiver. We used another Atmega8 at receiving side. Here, UART receiver was enabled by setting TXEN, RXEN bit and RXCIE (RX complete interrupt enable) of UCSRB register. Receiver’s data pin was directly connected with microcontrollers serial data reception pin (Pin 2). When one byte data is received at UART, an internal interrupt is occurred. At interrupt service routine we checked whether this is a data byte (as we had to send some flag and synchronization byte for reliable communication) and transmitted the data byte to Computer’s

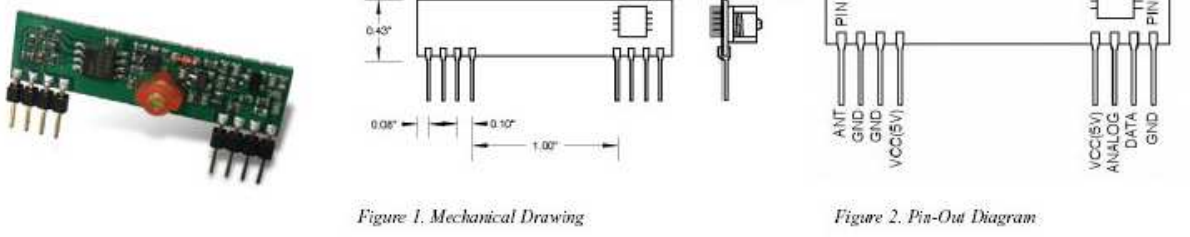


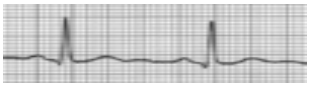
Figure 1. Mechanical Drawing *Figure 2. Pin-Out Diagram*

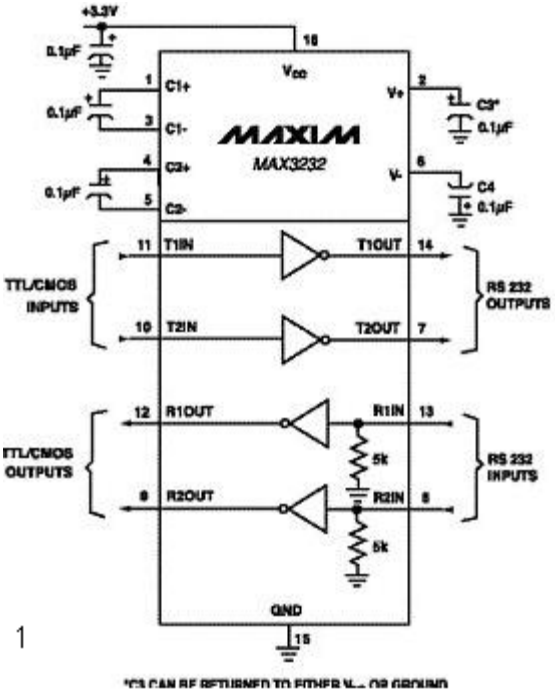
Figure :Wireless Receiver (RCR-433-ASBR)

serial port through MAX232. We had the same baud rate (4800 bps, UBRR = 12) and frame format (1 start bit, 8 data bit, 2 stop bit, no parity bit) in receiving end .

We need MAX232 to communicate with PC because microcontroller sends serial data in TTL level. PC’s serial port works in RS-232 logic level which is different from TTL level. AVR microcontroller’s logic level is: “0” – 0 volt,”1”- 5 volt while RS-232 logic levels are different: “0”- from +3v to 25v and “1”- from -3v to -25v. For level conversion TTL to RS-232 external level converter was used. MAX232 serves this purpose with the use of some external capacitors (1µF to 22µF) and 5V source. On the PC part we used DB9 connector to connect with the PC’s serial por.

Null modem communication configuration was used for this purpose. In order to make this work we have to “cross” some of the wires so when the microcontroller transmits some information on one end, the computer is able to detect and receive the same information.





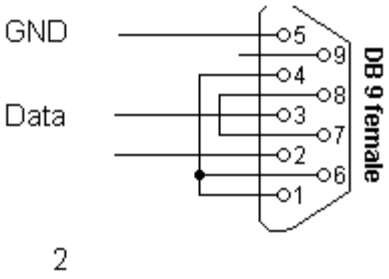


Figure : 1.Max232E
2.Null Modem Connection

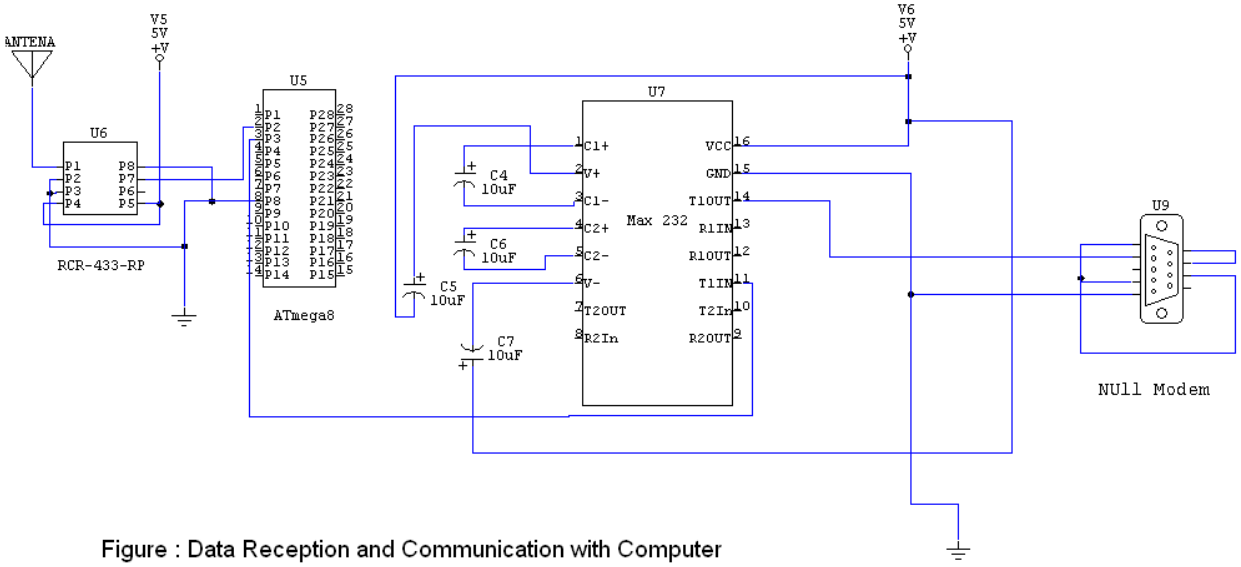
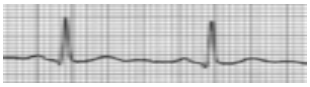


Figure : Data Reception and Communication with Computer



Software description

We built software which was used to show the ECG curve in PC. This software is written in Java. We used Java Communication API to communicate with PC's serial Port. Java Communication API is not available with the general JDK distribution. However, we included all resources about this API and it's installation instruction in the CD.We used Java Serial Port calss to read from serial port. When receiving data from serial port, it buffers them in a Vector. Another thread removes data from this buffer and updates the screen. Linear approximation was used to plot this curve. We also had to set the frame format to read from Serial Port. This is very important as if this frame format must match with previous formats we have used so far. Otherwise, synchronization will be lost. Frame format setting is easy using Java Communication API. We set it by this way:

```
serialPort.setSerialPortParams (4800,SerialPort.DATABITS_8,SerialPort.STOPBITS_2,SerialPort.PARITY_NONE);
```

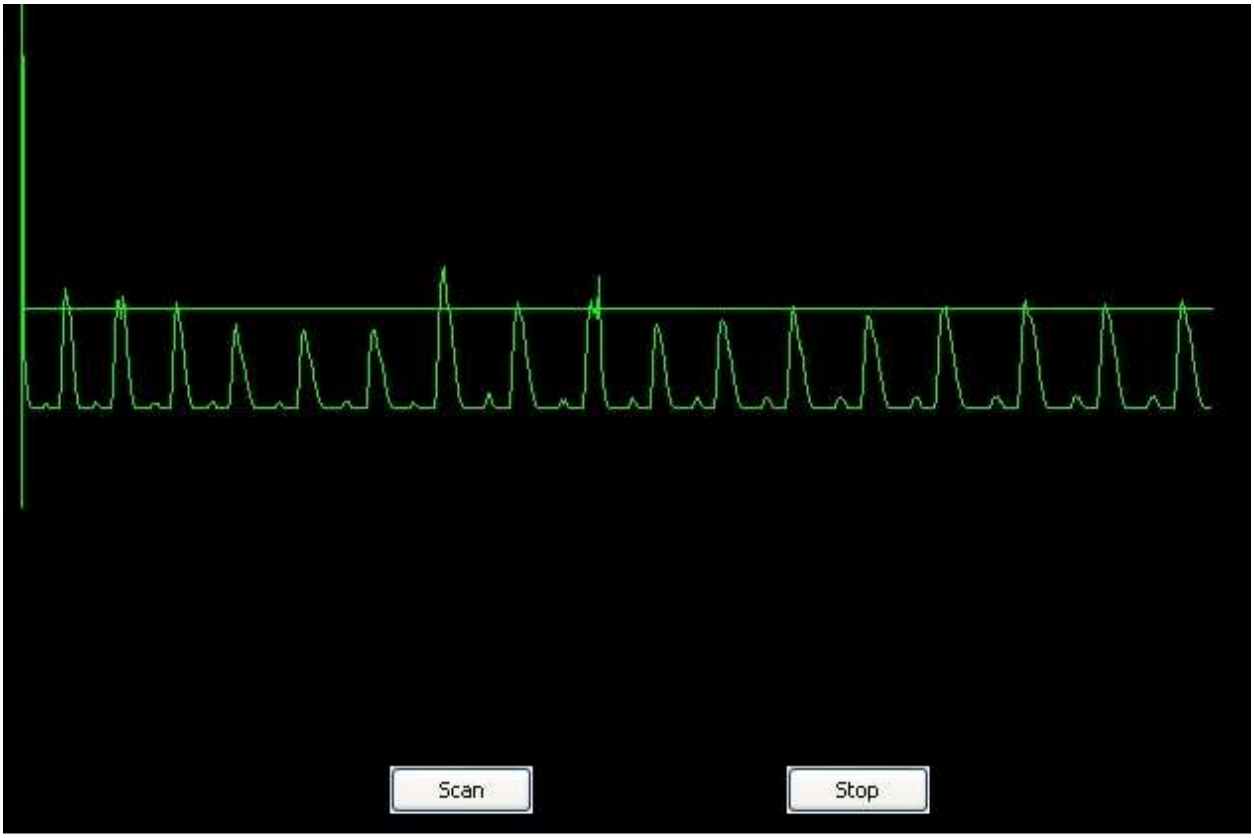
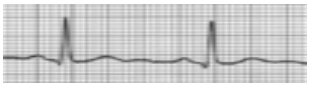


Figure : Final Output (ECG Curve) at Computer

(Full Code of this software and microcontroller is attached in Appendix and CD)

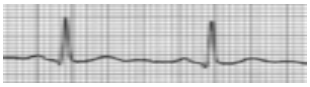


Some problems and Solutions:

1. ADC unit works between ground and specified voltage reference (at most 5 volt). Any signal out of this range is not properly reflected on output data. So, we had to modify the analog signal.
2. ADC of microcontroller gives digital result in 10 bit resolution. But for serial communication the usual UART frame can have at most 8 bits/ So, we have to discard last 2 bits to make sure continuous data transmission. Discarding last 2 bits looses very small amount of accuracy in case of 10 bit resolution.
3. Using a good and precise voltage source for microcontroller Vcc is important for good serial communication. We used 7805(Linear Regulator) which ensured constant 5V supply.
4. It is very hard to make the analog circuit stable. Because, as the heart signal is at some millivolt range, if we amplify it noise is amplified too. So, we had to reduce circuit component as much as possible. We did not used conventional OP-AMP(741). Instead, we used TL-084 which had 4 OP-AMPs inside it.
5. For ensuring smooth and error free wireless communication, it is very important to send some flag and synchronization byte with each data byte. We used 16 flag bytes and 1 synchronization byte. By this way, we ensured 100% error free, real time wireless transmission.

Future prospects

- In this project, we successfully built Wireless ECG system. Our next target will be to build this system for hospital. There will be a central server and the ECG sensing device will be attached to the body of the critical heart patients. The device attached to the body will continuously transmit patient data to the server. In the server the ECG waveform will be analyzed by pattern recognition. If something abnormal observed related doctor will be altered via his mobile phone.
- If everything works fine we can further enhance the system going beyond the limit of a hospital. For this patient location system can be tried. Our ECG sensing device can be equipped with GPS module which can readily communicate through wireless media to a nearby server. So patient location is always available to our device. When critical situation arise, our device will be able to send patient location to the server and ask for immediate help.



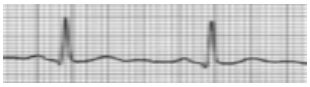
Conclusion

It is not easy at all to build a wireless electrocardiogram system. To finish the project in due time, we divided our group in 2 major parts. One group worked to build and stable Analog circuit and other worked for successful wireless communication and computer interfacing. As a result, we were able to build the system within 11th week of the running semester. It is also good to note that we participated in Project Show at CSE DAY'08. As our system is very cheap, if this project can be implemented in practically then it will bring a radical change in public health care system especially in heart care system.

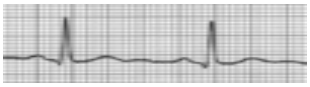
Appendix:

Microcontroller Code at Wireless Transmission part

```
#include <avr/io.h>
#include <util/delay.h>

void USART_Transmit( unsigned char data );
void USART_Init(unsigned int baud);
void InitADC();
long ADCresult(char n);

void USART_Transmit( unsigned char data )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );

        /* Put data into buffer, sends the data */
        UDR = data;
}

void InitADC()
{
        ADCSRA |= (1<<ADEN) | (1<<ADPS1);
}

void USART_Init( unsigned int baud )
{
```

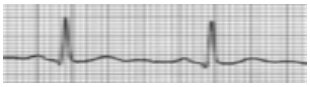


```
        /* Set baud rate */
        UBRRH = (unsigned char)(baud>>8);
        UBRRL = (unsigned char)baud;

        /* Enable transmitter */
        UCSRB = (1<<TXEN);

        /* Set frame format: 8data, 2stop bit */
        UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
long ADCresult(char n)
{
        ADMUX=n;
        ADCH &=0b00000011;
        ADCSRA |= (1<<ADSC); //start conversion

        while( (ADCSRA & 0x40) !=0 ) {};
        return ADC;
}
int main(void)


{
        unsigned char data;
        long temp;
 USART_Init(12); //4800 bps
        InitADC();
        while(1)
        {
                int i;
                /*get value from ADC*/
                temp=ADCresult(0);
                temp=temp>>2;
                data=temp;
                //send some flag bytes
                for(i=0;i<15;i++)
                {
                        USART_Transmit(0xAA);
                }
                //then send synchronization byte
                USART_Transmit(0xFF);
                USART_Transmit( data ); // now our main data
                _delay_ms(1000);
        }
        return 0;
}
```

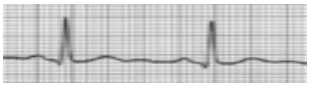


Microcontroller Code at Wireless Receiving part

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
unsigned char USART_Receive( void );
void USART_Init( unsigned int baud );
void USART_Transmit( unsigned char data );
unsigned char rx_data, prev1, prev2;


ISR(USART_RXC_vect)
{
        rx_data = USART_Receive();
        if   ( (prev1 == 0xAA) && (prev2 == 0xFF) )
        {
                PORTC = rx_data;//To See what we actually received
                USART_Transmit(rx_data);//Transmit the data to serial port
                _delay_ms(1000);
        }
  /* if this is not data update variables */
        prev1 = prev2;
        prev2 = rx_data;
}



unsigned char USART_Receive( void )
{
        /* Wait for data to be received */
        while ( !(UCSRA & (1<<RXC)) );
        /* Get and return received data from buffer */
        return UDR;
}


void USART_Transmit( unsigned char data )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );
        /* Put data into buffer, sends the data */
        UDR = data;
}
```

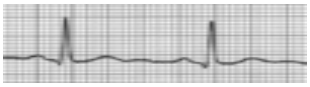


```
void USART_Init( unsigned int baud )
{
        /* Set baud rate */
        UBRRH = (unsigned char)(baud>>8);
        UBRRL = (unsigned char)baud;
        /* Enable receiver and receive interrupt*/
        //UCSRB = (1<<RXCIE)|(1<<RXEN);
        UCSRB = 0b10011000;//enable rxcie,rxen and txen

        /* Set frame format: 8data, 2stop bit */
        UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
        sei();     // Global interrupt enable
}
int main(void)
{
        DDRC = 0xFF;
        prev1 = 0x00;
        prev2 = 0x00;
         USART_Init(12); //4800 bps
        while(1)
        {
        }
}
```

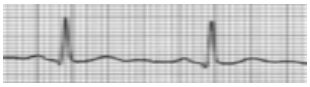


Software code to see ECG curve at computer

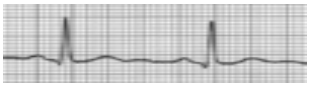
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.*;
import javax.comm.*; // for java comm API
/*
          @author : Amit
*/
 class ECG extends JFrame implements ActionListener
 {
   JButton startButton;
   JButton stopButton;

   private SerialComm sc = null;
   private int dx, dy, x, y;
   private Vector point;
   private final int TOTAL_X = 600;
   private Thread updateThread = null;

   public ECG()
   {
      ecgLayout customLayout = new ecgLayout();

      this.getContentPane().setBackground(Color.BLACK);
      getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));
      getContentPane().setLayout(customLayout);
     // this.getContentPane().setBackground(Color.LIGHT_GRAY);

      startButton = new JButton("Scan");

      getContentPane().add(startButton);
      startButton.addActionListener(this);

      stopButton = new JButton("Stop");
      getContentPane().add(stopButton);
      stopButton.addActionListener(this);

      setSize(getPreferredSize());
      setResizable(false);

      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            System.exit(0);
```



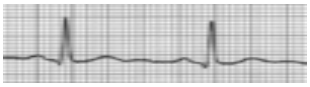
```
    }
  });

  dx = 25;
  dy = getHeight() / 2;

  point = new Vector();
  for (int i = 0; i < TOTAL_X; i++) {
    point.add(new Integer(0));
  }

}

public void actionPerformed(ActionEvent event) {
        if (event.getSource() == startButton) {
                startButtonActionPerformed();
        } else if (event.getSource() == stopButton) {
                stopButtonActionPerformed();
        }
}

private void startButtonActionPerformed() {
        if (updateThread == null) {

                updateThread = new Thread(new UpdateThread());
                sc = new SerialComm();
                point.clear();
                for (int i = 0; i < TOTAL_X; i++) {
              point.add(new Integer(0));
          }
                updateThread.start();
        }
}

private void stopButtonActionPerformed() {
        if (updateThread != null) {
                sc.stopReading();
                sc = null;
                        updateThread.stop();
                        updateThread = null;
                        point.clear();
                        for (int i = 0; i < TOTAL_X; i++) {
              point.add(new Integer(0));
          }

        }
}
```



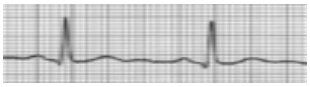
```
public void paint(Graphics g) {
   super.paint(g);
   //g.setColor(new Color(255, 0, 0));
   g.setColor(Color.GREEN);


   g.drawLine(dx,dy,dx + TOTAL_X, dy);
  g.drawLine(dx, dy - 200, dx, dy + 100);



   Integer tmp = (Integer) point.get(0);
   int y0 = tmp.intValue();
   int x0 = dx;

   for (int i = 1; i < TOTAL_X; i++) {
      x = i + dx;
      tmp = (Integer) point.get(i);
      y = dy - tmp.intValue();
      g.drawLine(x0,y0,x,y);
      x0 = x;
      y0 = y;
   }
}


private class UpdateThread implements Runnable {
   public void run() {
      while (true) {
         int v = sc.getData();
         if (v == -1) {
                  //System.out.println("dataList empty");
                  continue;
         }
         y = (int) (200.0 / 256.0 * v);
         y -= 50;
         point.add( new Integer(y) );
         point.remove(0);
         repaint();

         try {
            Thread.sleep(2);
         } catch (InterruptedException ex) {
            ex.printStackTrace();
         }
      }
```



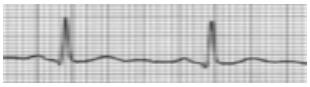
```
    }

  }

  public static void main(String args[])
  {
        try
    {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

    }
    catch (Exception ex)
    {
      System.out.println(ex.toString());
    }
    ECG window = new ECG();

    window.setTitle("ECG");
    window.pack();
    window.setVisible(true);
    window.setLocation(150, 150);
    window.setResizable(false);
    window.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);


  }
}


class SerialComm
{
        private CommPortIdentifier portId;
        SerialPort serialPort;
        //static CommPortIdentifier portId;
        //static CommPortIdentifier saveportId;
        static Enumeration      portList;
        InputStream         inputStream;
//      SerialPort          serialPort;
//      Thread          readThread;

//      static String       messageString = "Hello, world!";
//      static OutputStream     outputStream;
//      static boolean      outputBufferEmptyFlag = false;
        boolean         portFound = false;
  String        defaultPort;
 //  private FileOutputStream fos = null;
  //private DataOutputStream dos = null;
```



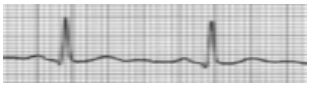
```
private final int DATA_MAX = 10000;                    // what should be this max value

        private Vector dataList;

        public SerialComm()
        {

                String osname = System.getProperty("os.name","").toLowerCase();
        if ( osname.startsWith("windows") )
        {
     // windows
     defaultPort = "COM4";
     System.out.println("Set default port to "+ defaultPort);
    }

                // parse ports and if the default port is found, initialized the reader
            portList = CommPortIdentifier.getPortIdentifiers();
            while (portList.hasMoreElements()) {
              portId = (CommPortIdentifier) portList.nextElement();
              if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL)
              {
                if (portId.getName().equalsIgnoreCase(defaultPort))
                {
                  System.out.println("Found port: "+defaultPort);
                  portFound = true;
                  break;
                  // init reader thread
                 // nulltest reader = new nulltest();
                }
              }

            }
            if (!portFound)
            {
              System.out.println("port " + defaultPort + " not found.");
              System.exit(1);
            }
            dataList = new Vector();
            try
            {
                  serialPort = (SerialPort)portId.open("ECG",2000);
                  inputStream = serialPort.getInputStream();
                  System.out.println("\nPL:Got inputStream");
        //        outputStream =  serialPort.getOutputStream();
          //      System.out.println("\nPL:Got outputStream");
                  serialPort.addEventListener(new SP_event_listener());
                  serialPort.notifyOnDataAvailable(true);
```



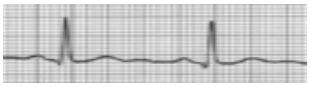
```
            //
        serialPort.setSerialPortParams(2400,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PA
RITY_NONE);


serialPort.setSerialPortParams(4800,SerialPort.DATABITS_8,SerialPort.STOPBITS_2,SerialPort.PARITY_N
ONE);

             }
             catch(Exception e)
             {
                   e.printStackTrace();

             }


       }


          public int getData() {
                if (dataList.size() > 0) {
                      Integer n = (Integer) dataList.remove(0);
                      return n.intValue();
                } else {
                      return -1;
                }
          }

          public void stopReading() {
                serialPort.close();
          }

                private class SP_event_listener implements SerialPortEventListener
                {
                      public void serialEvent(SerialPortEvent event)
                      {

                            switch(event.getEventType())
                            {
                                  case SerialPortEvent.BI:
                                  case SerialPortEvent.OE:
                                  case SerialPortEvent.FE:
                                  case SerialPortEvent.PE:
                                  case SerialPortEvent.CD:
                                  case SerialPortEvent.CTS:
                                  case SerialPortEvent.DSR:
                                  case SerialPortEvent.RI:
```



```
                                        case SerialPortEvent.OUTPUT_BUFFER_EMPTY: break;
                                        case SerialPortEvent.DATA_AVAILABLE:
                                                try
                                                {
                                                        while(inputStream.available()>0)
                                                        {
                                                                int data =(int) inputStream.read() ;
                                                                System.out.println("Receive "+ data);
                                                                if (dataList.size() <= DATA_MAX) {
                                                                        dataList.add(new
Integer(data));
                                                                } else {


                                                                }

                                                        }


                                                }

                                                catch(Exception es)
                                                {
                                                        es.printStackTrace()      ;

                                                }
                                                break;
                                }

                        }
                }

}

class ecgLayout implements LayoutManager {

   public ecgLayout() {
   }

   public void addLayoutComponent(String name, Component comp) {
   }

   public void removeLayoutComponent(Component comp) {
   }

   public Dimension preferredLayoutSize(Container parent) {
      Dimension dim = new Dimension(0, 0);
```



```
        Insets insets = parent.getInsets();
        dim.width = 692 + insets.left + insets.right;
        dim.height = 540 + insets.top + insets.bottom;

        return dim;
    }

    public Dimension minimumLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        return dim;
    }

    public void layoutContainer(Container parent) {
        Insets insets = parent.getInsets();

        Component c;
        c = parent.getComponent(0);
        if (c.isVisible()) {c.setBounds(insets.left+208,insets.top+472,72,24);}
        c = parent.getComponent(1);
        if (c.isVisible()) {c.setBounds(insets.left+408,insets.top+472,72,24);}
    }
}
```



Acknowledgement

Our project has benefited from support, guideline and sharing experience from a lot of people. It is indeed a great pleasure for us to express our gratitude to those people who helped directly and indirectly.

First of all, we want to thank our project supervisor Md. Abdullah Adnan and our Computer interfacing course teacher Chowdhury Sayeed Hyder. Both of them gave us continuous inspiration, helped with providing instrument support and thoroughly reviewed all the work which keep us busy to give a quality product.

Then, we want to thank our friend Ashik Zinnat Khan (0405092) who gave us the wireless modules and supported us always when we faced any problem in wireless transmission. Also,our friend Ahmed Mashfique Raihan (0405029) & Md. Ahsanur Rashid(0405075) always gave us full support.

Among seniors Nashid vai(CSE 03), Subrata vai(CSE 03), Taskinoor vai(CSE 03) and Rosi Vai(CSE 03) always helped us with all kinds of resources and advices. They told us the problems they faced while building Analog circuit and wireless transmission. These helped us a lot to face the challenge of building Wireless ECG and at last we succeeded in the project.