miBEAT ECG Monitor

miBEAT has been updated with SciChart's WPF charts which are DirectX powered. These replace the LabWindows charts used in the previous edition. The visual comparison is shown below.

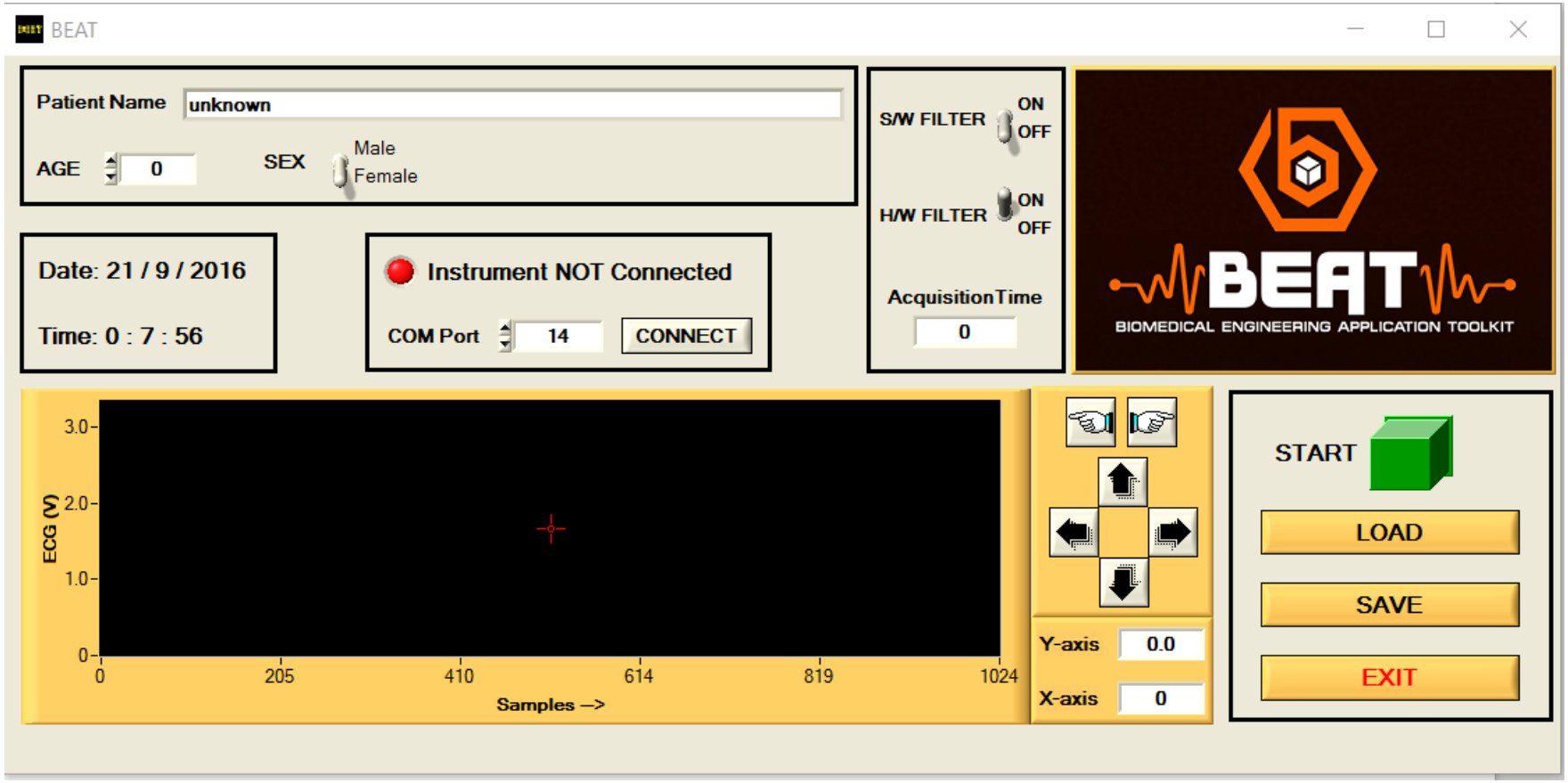


Figure 1: miBEAT (LabWindows)

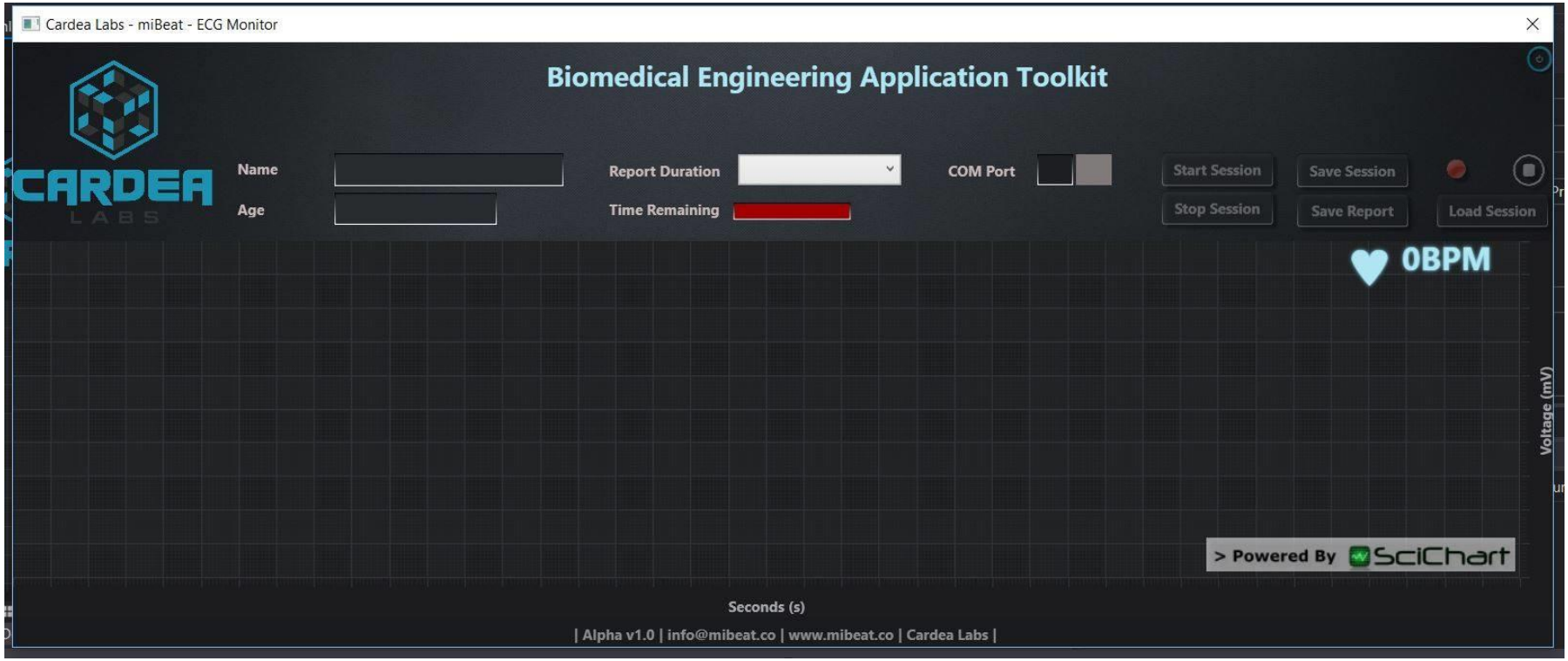


Figure 2: miBEAT Main Window (SciChart)

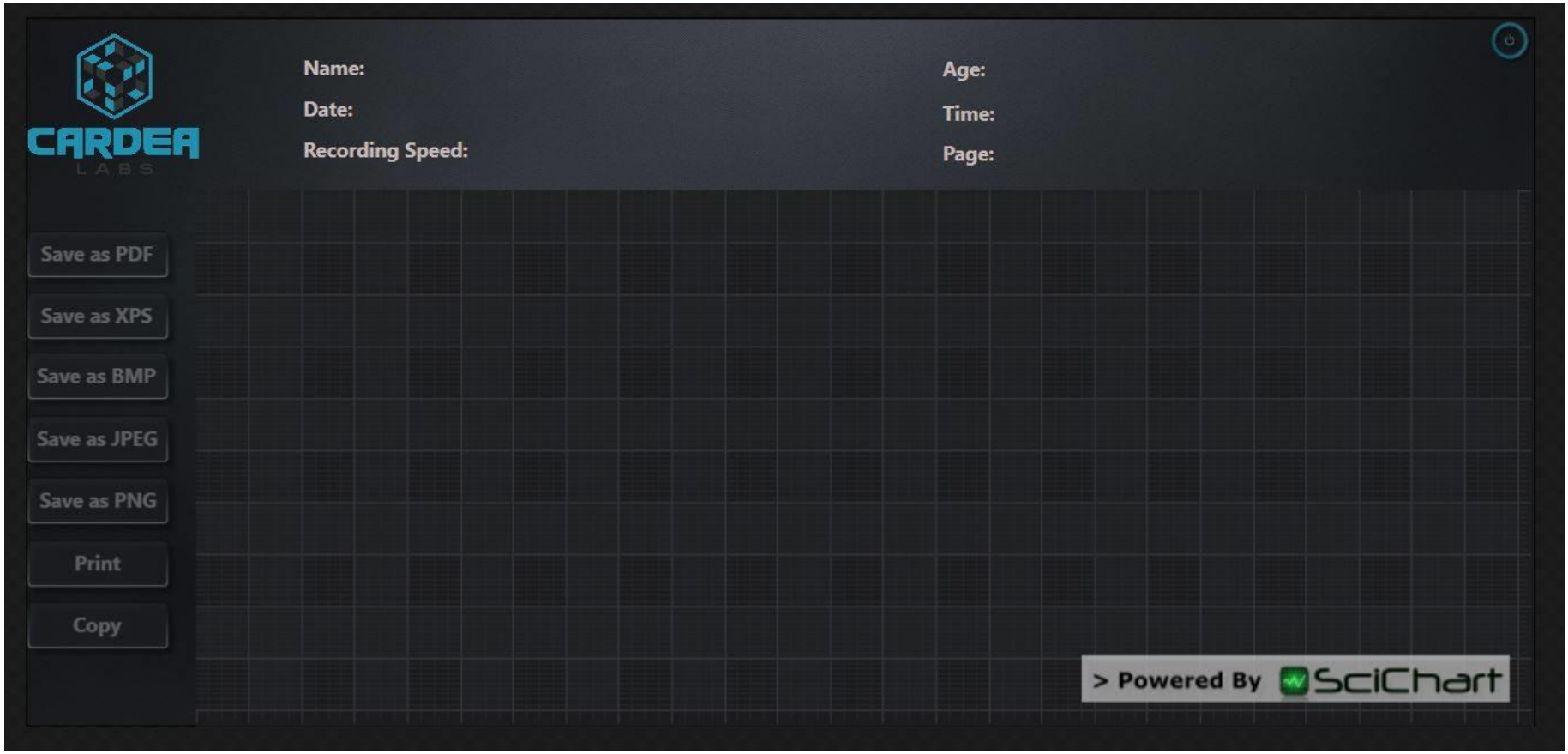


Figure 3: miBEAT Print Window (SciChart)

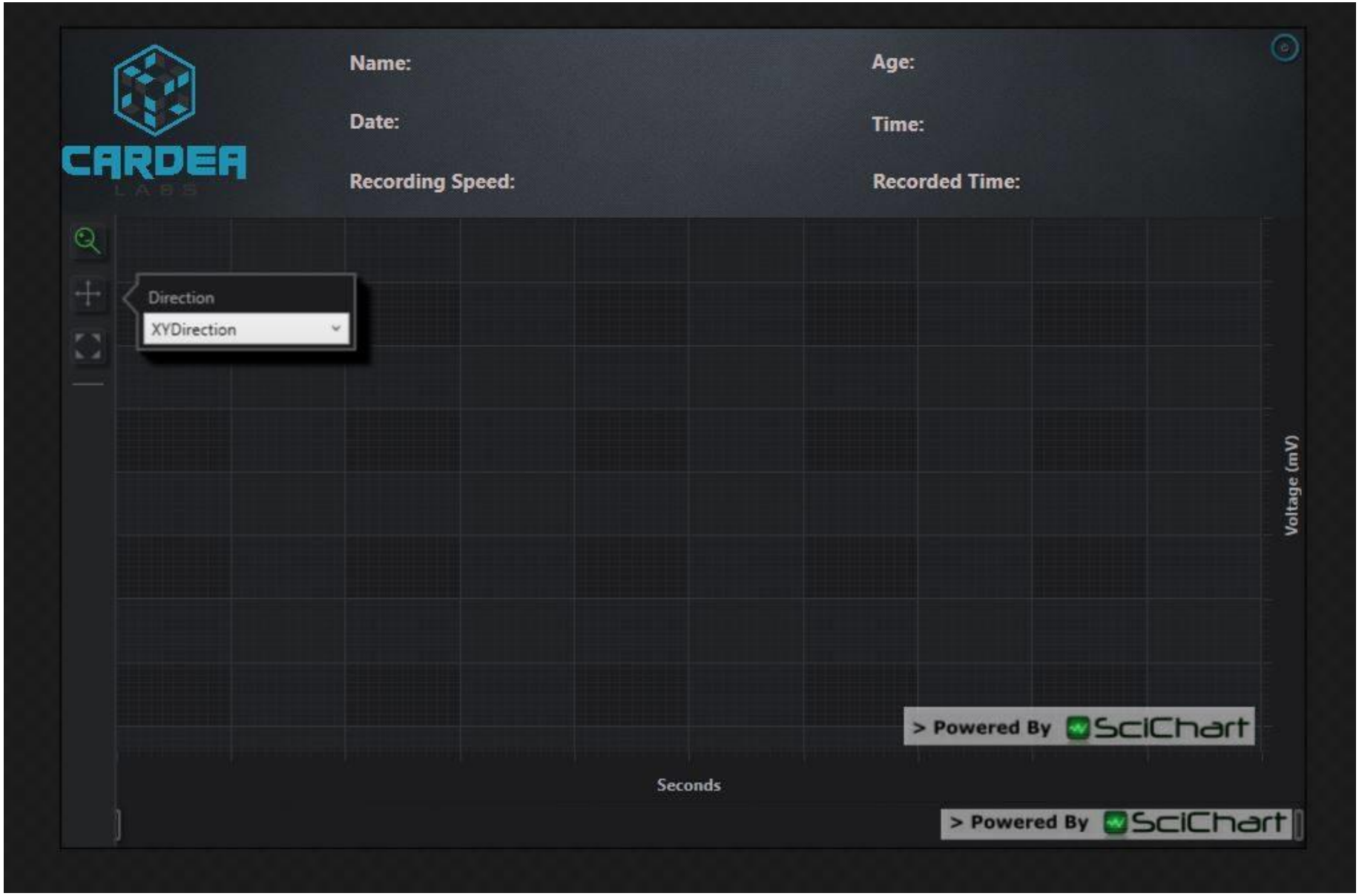

CARDEA
LABS
Name:
Age:
Date:
Time:
Recording Speed:
Recorded Time:
Direction
XYDirection
Voltage (mV)
> Powered By SciChart
Seconds
> Powered By SciChart

Figure 4: miBEAT Session Window (SciChart)

The main window makes use of the FIFO functionality of SciChart to generate a real time scrolling ECG monitor. All chart surfaces use FastLineRenderableSeries which is used to render a XyDataSeries.

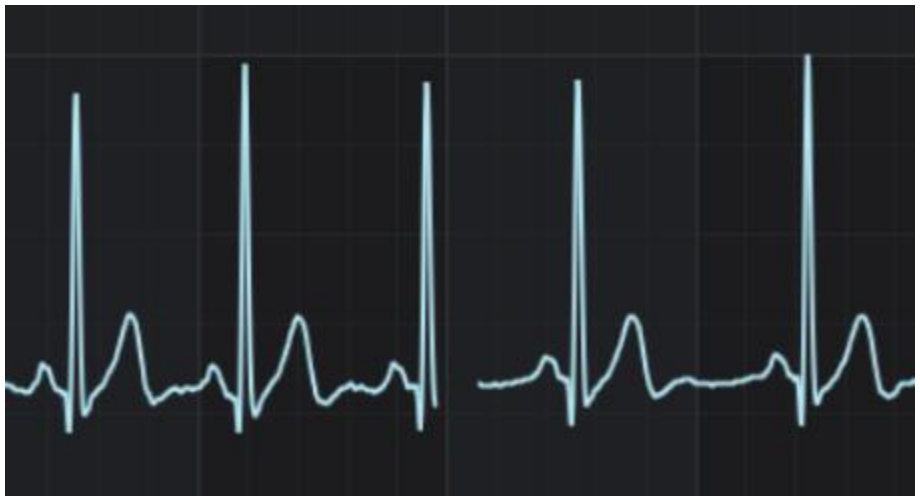


Figure 5: FIFO Scrolling Trace

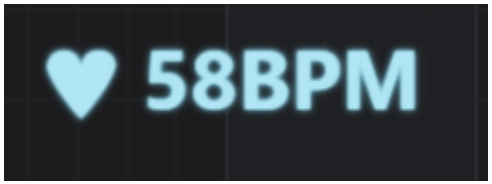


Figure 6: Heart Rate Calculator

The print window makes use of the line graph functionality and SciChart's inbuilt export functionality to export to XPS, JPG, BMP and PNG. This is then converted to a PDF file by using a third party extension, PDFSharp.

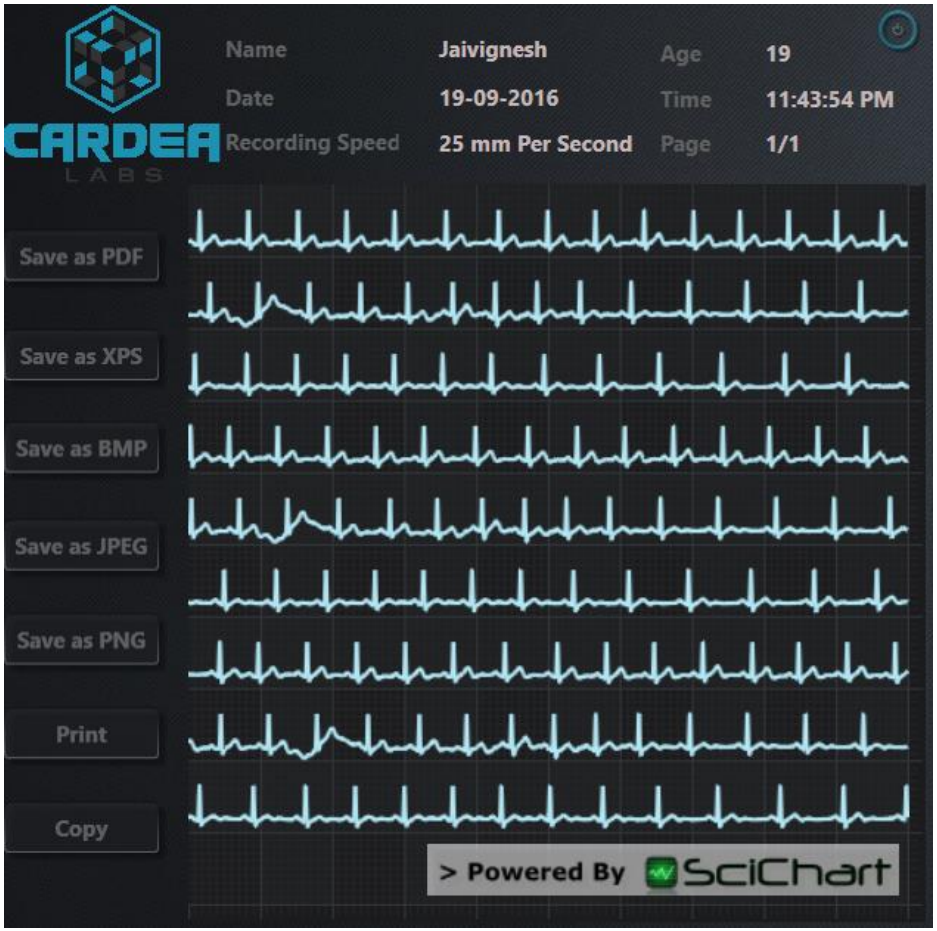
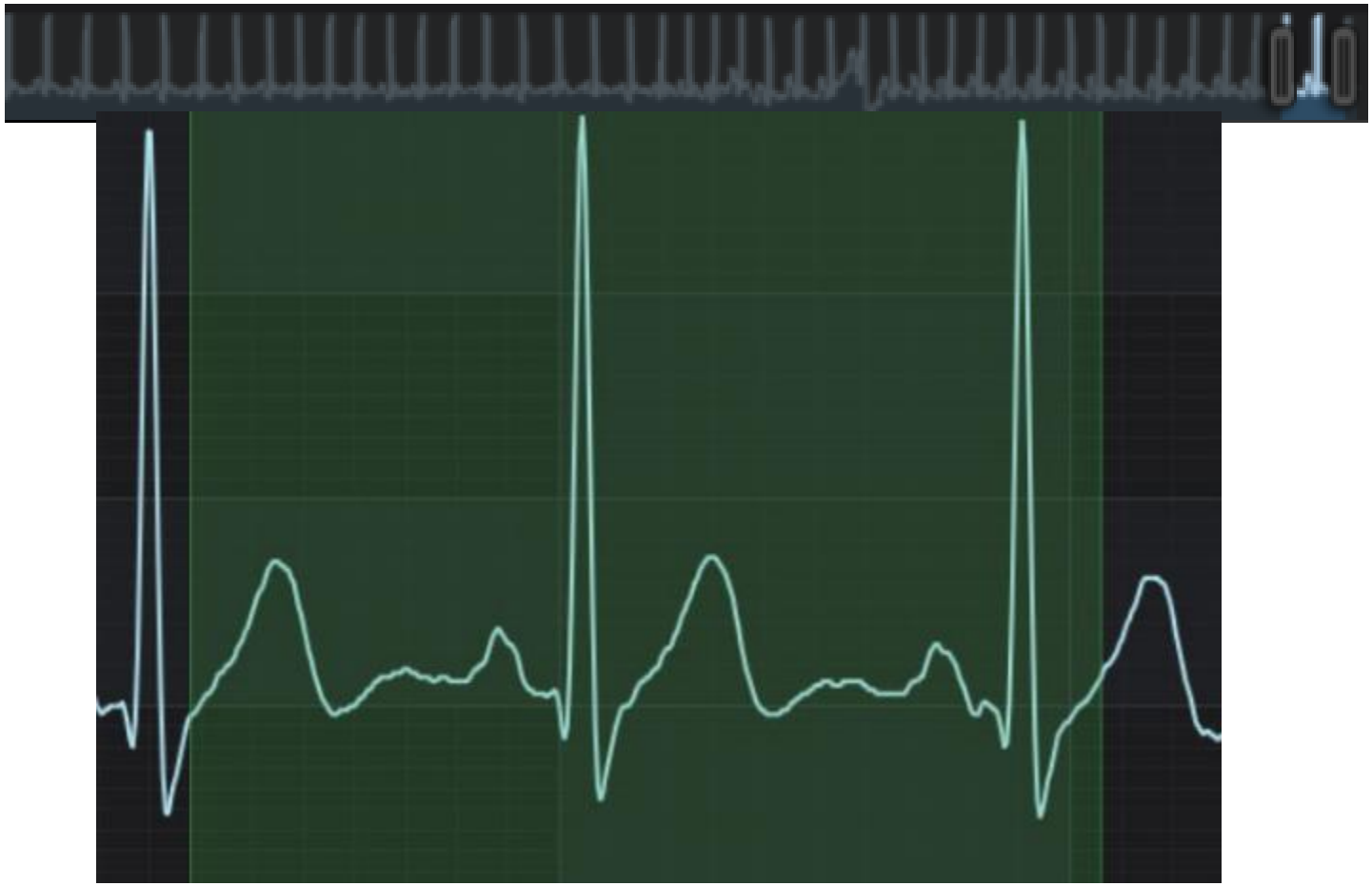


Figure 7: miBEAT Functional Print Window (SciChart)

The session window makes use of the Pan, Scroll and Zoom functionality of SciChart to simulate viewing of a recorded session. This also uses SciChart Overview. This control can be bound to a parent SciChartSurface and renders the first series on that surface in entirely. A reticule is overlaid on the SciChartOverview to show the portion of the parent data in the viewport. You can interact with the SciChartOverview by dragging the middle area to scroll, or the edges of the middle area to scale. Similarly as you pan the chart by dragging, the SciChartOverview updates to show the current section of the data in the viewport. This doesn't increase the memory burden by much as the SciChartOverview shares the same source data as the parent SciChartSurface.



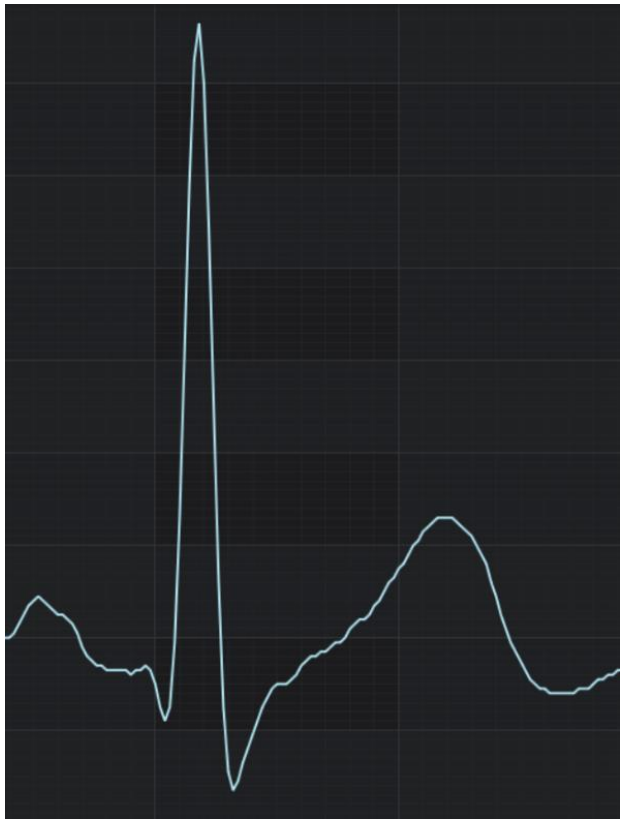


Figure 8: SciChart Overview Control (Scroll)

Figure 9: RubberBandXyZoomModifier (Zoom)

Figure 10: ZoomPanModifier (Pan)

All data sources for the session window use a double array whose source is a text file on the local disk. This is loaded using the StreamReader class.

```
System.IO.StreamReader file_Record =
   new System.IO.StreamReader("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\Test_\\report_test.txt");
```

For real-time data acquisition, data from the hardware is acquired from a HC-05 module (BlueTooth) using the SerialPort class. This is then written to a local file for later analysis.

```
public static void mySerialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    if (flag_Timer == true)
    {
        while (serialPortBT.BytesToRead > 0)
        {
            int data_Byte;
            double data_Double;
            data_Byte = serialPortBT.ReadByte();
            data_Double = Convert.ToDouble(data_Byte);
            _sourceData[counter] = data_Double;
            counter++;
            if(ECGMonitorView.timer_Record.IsEnabled==true)
            {
                _recordData[counter_Record] = data_Double;
                counter_Record++;
            }
        }
    }
}
```

For the print window, the entire recorded session (30/60/90 seconds) is split into 10 second sessions. These individual sessions are then stacked on top of each other to simulate an ECG report.

```
if (recording_Time == 30)
{
    while (counter >= 0 && counter < 2500)
    {
        if (counter == 0)
        {
            dataSeries.Append(counter, double.NaN);
        }
        else
        {
            dataSeries.Append(counter, (_sourceData1[counter - 0] + 700));
        }
        counter++;
    }

    while (counter >= 2500 && counter < 5000)
    {
        if (counter == 2500)
        {
            dataSeries.Append((counter - 2500), double.NaN);
        }
        else
        {
            dataSeries.Append((counter - 2500), (_sourceData2[counter - 2500] + 400));
        }
        counter++;
    }
```

The saving of the report is done using the inbuilt export functionality, which also supports printing to an external printer.

```
private void save_PDF_Click(object sender, RoutedEventArgs e)
{
    string ext = Convert.ToString(DateTime.Now);
    string filename_xps = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.xps", DateTime.Now);
    string filename_pdf = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.pdf", DateTime.Now);
    sciChart.ExportToFile(filename_xps, ExportType.Xps, useHighQualityOutput = true);
    using (PdfSharp.Xps.XpsModel.XpsDocument pdfXpsDoc = PdfSharp.Xps.XpsModel.XpsDocument.Open(filename_xps))
    {
        PdfSharp.Xps.XpsConverter.Convert(pdfXpsDoc, filename_pdf, 0);
    }
}

private void save_XPS_Click(object sender, RoutedEventArgs e)
{
    string ext = Convert.ToString(DateTime.Now);
    string filename_xps = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.xps", DateTime.Now);
    sciChart.ExportToFile(filename_xps, ExportType.Xps, useHighQualityOutput = true);
}

private void save_BMP_Click(object sender, RoutedEventArgs e)
{
    string ext = Convert.ToString(DateTime.Now);
    string filename_bmp = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.bmp", DateTime.Now);
    sciChart.ExportToFile(filename_bmp, ExportType.Bmp, useHighQualityOutput = true);
}

private void save_JPEG_Click(object sender, RoutedEventArgs e)
{
    string ext = Convert.ToString(DateTime.Now);
    string filename_jpeg = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.jpg", DateTime.Now);
    sciChart.ExportToFile(filename_jpeg, ExportType.Jpeg, useHighQualityOutput = true);
}

private void save_PNG_Click(object sender, RoutedEventArgs e)
{
    string ext = Convert.ToString(DateTime.Now);
    string filename_png = string.Format("C:\\Users\\jaivignesh\\Desktop\\VIT\\Internship_Cardea\\Data\\BEAT_Report-{0:yyyy-MM-dd_hh-mm-ss-tt}.png", DateTime.Now);
    sciChart.ExportToFile(filename_png, ExportType.Png, useHighQualityOutput = true);
}
```

We are extremely grateful to SciChart Ltd (http://www.scichart.com) for their continued support throughout the lifespan of the development phase.

This is the alpha version (v1.0) and would be released shortly. Keep visiting this page to view the latest updates to this application.

Cheers.

Author: Jaivignesh Jayakumar

Jaivignesh is a student at Vellore Institute of Technology, Vellore. He is also a part time .NET application developer for Cardea Labs and is currently working on the miBEAT project.