December 15, 2003

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: Post-Mortem Report for an Emergency Response System

Dear Dr. Rawicz

The attached document, *Post-Mortem Report for an Emergency Response System*, outlines the process our team went through when designing and implementing our product for our ENSC 340 project. We are developing a device that will help people with medical conditions obtain assistance under emergency situations.

This document details the current state of the device, deviations from our original plans, and our future plans for the device. In addition, we outline some of the budget details and time constraints we encountered and explain the inter-personal and technical experience gained from working on the project.

GiveLife Systems was formed in September of 2003 by four motivated, innovative, and talented engineering students: Tristen Georgiou, Yang Pan, Hashina Parveen, and Melody Guo. If you have any questions or concerns about our functional specification, please feel free to contact me by phone at (604) 771- 6089 or by e-mail at givelife-systems@sfu.ca.

Sincerely,

Y Pan

Yang Pan
President and CEO
GiveLife Systems, Inc

Enclosure: *Post-Mortem Report for an Emergency Response System*

GIVELIFE
Systems

Post-Mortem Report for an

Emergency Response System

Project Team: Tristen Georgiou
Yang Pan
Hashina Parveen
Melody Guo

Submitted to: Dr. Andrew Rawicz – ENSC 340
Mr. Steve Whitmore – ENSC 305
School of Engineering Science
Simon Fraser University

Issued date: December 15, 2003

Revision: 1.0

TABLE OF CONTENTS

1. INTRODUCTION

GiveLife Systems (GLS) use off-the-shelf sensors, microcontrollers, and wireless technology to develop *Intelli-Alert* ™. The system improves upon existing technology by monitoring and acting upon pulse rates of the user. This report re-examines the process that took this dream from concept to reality and documents the earth-shattering experiences of each of the four members.

2. CURRENT STATE OF THE DEVICE

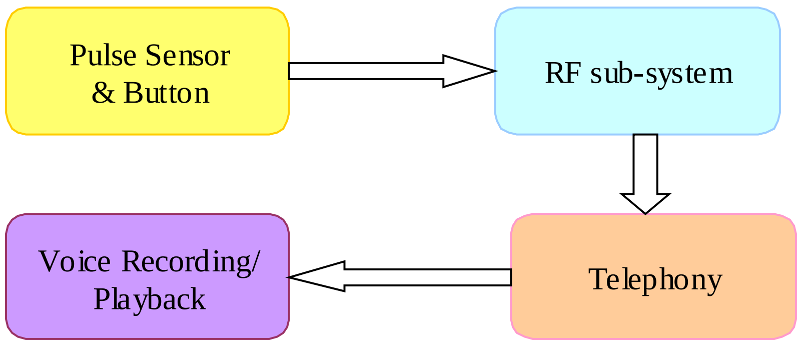

Figure 1 System Block Diagram

Figure 1 above shows the system block diagram of the major components of *Intelli-Alert* ™. When the pulse sensor detects the user's irregular pulse beat or the panic buttons are pressed, the RF transmitter sends a radio signal to the RF receiver at the base station located in the house. The telephony system dials the pre-programmed numbers for help. Once the call is answered, the base station will play a pre-recorded message stating the address where help is needed.

The IR LED and phototransistor pairs are placed on opposite sides of finger to detect a human pulse from the fingertips. A circuit which consists of signal acquisition, filtering, and amplification is used to detect and amplify the small signal caused by blood flow.

Two SPST-NO switches are placed in series and connected to the microcontroller. A simple and efficient firmware is designed for the OOPic II microcontroller evaluation kit. The microcontroller monitors a few different conditions and act upon those conditions. The code simply handles the following two conditions: both buttons depressed simultaneously or a 0 pulse rate. If either of these conditions is detected, one of the digital I/O pins will go high, and the emergency signal will be transmitted.

Both the pulse sensor and push buttons are interfaced directly into the microcontroller. Each is connected to one of the microcontroller's digital I/O lines.

The Master Evaluation Kit model # MDEV-900-HP produced by Linx Technologies is used to implement the RF link between the remote device and the base station. The data input pin on the Linx MDB with the transmitter module is directly connected to the digital output from the microcontroller. The Linx MDB with the receiver module is connected to a PC via a serial connection.

A TAPI-compliant voice modem is used to handle the telephony portion of our product. TAPI sets up the path for call but does not interpret the data on the path. It requires appropriate media stream API to handle the media stream.

Microsoft Visual Basic 6.0 is used to write an application which functions as 1) an interface for the user to record message and set phone numbers to be dialled under emergency situation and 2) make emergency phone calls when radio trigger is received.

3. DEVIATION FROM PLAN

3.1 Overall System

In terms of functionality, we achieved the minimum requirement that we planned. Due to time and budget constraints, we were unable to package the device to make it water proof.

3.2 Pulse Sensor & Button

The final prototype of pulse sensor device is made for the user to wear it around his finger rather than on his wrist. This is due to the difficulties we faced in getting out electrical circuit to detect the pulse via the wrist. The idea we had on using infrared sensors to detect the pulse instead worked better through the finger, as differences in voltage were much higher and more detectable.

3.3 RF Transmitter/Receiver

Everything worked out well with the RF modules. With its default functions and easy instructions, using the modules was a breeze. Only thing is that the RF transmitter is really big and does not fit onto the wrist. As a prototype, this is fine but for future development, an inbuilt model will be designed.

3.4 Base Station – Telephony & Voice Recording/Playback

Instead of designing a telephone answering machine from scratch to ring for help, we designed a software interface to ring for help via the modem telephone line. Using Microsoft Visual Basic 6.0, the interface was designed as shown below.

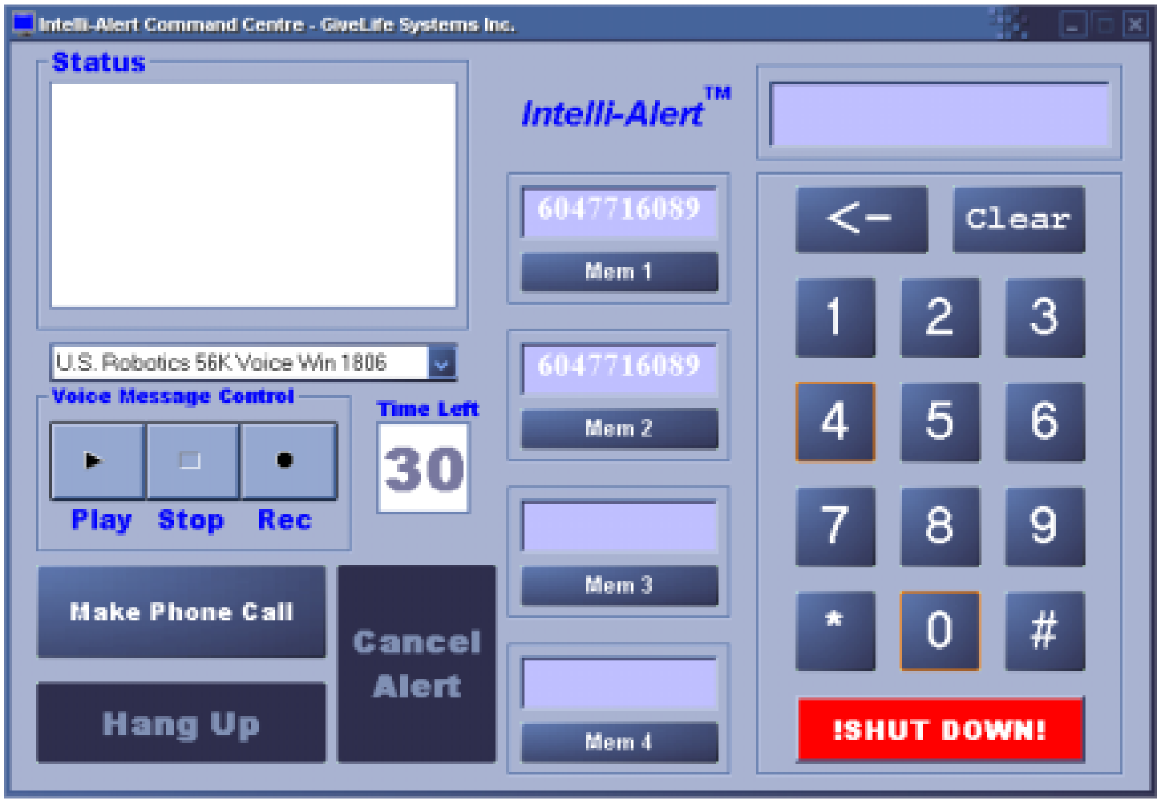

Figure 2 The Final GUI

This is a more efficient way of doing so as nowadays almost everyone has a computer in his or her home. With easy installation of the interface to the computer, the system is up and ready for action.

The voice recording and playback functionality is embedded into our control program, instead of using any third party program. We have created this option to let the user personalize his/her own voice message for help using this embedded recording function. This way the user can edit the voice message in such a way that the user's address, telephone or even the next of kin to call can be played out. This simplifies the process of using our product.

4. FUTURE PLANS

4.1 Overall System

Intelli-Alert ™ has great potential for further research and development. As we re-examine the development of the device, we have the following suggestions for future development.

Placing the pulse sensor on the finger can interfere with everyday activities, so future plans include research for placing the sensor in unobtrusive areas of the body, such as the wrist or earlobe

Although, there are many similar emergency response devices out in the market there is none the monitors pulse, and we feel strongly that our device caters to the elderly. Thus, there is a need for improvement on our prototype in terms of making it more user-friendly to the elderly and ensuring its durability since it will be worn all the time.

4.2 Pulse Sensor & Button

The current pulse sensor works by letting the user wear it on his finger. This can be uncomfortable and inconvenient at times since most of us use our hands to do many activities. A more pleasant way of keeping the device on the body is wearing the device like a watch. It is still there but does not come in your way during your everyday activities. Thus, a future development of this device is to re-design it to be worn on the wrist where the infrared sensors can effectively detect one’s pulse.

4.3 RF Transmitter/Receiver

Right now in our prototype, the RF transmitter and receiver are separate from rest of the components of the system. For future development of this product, the RF transmitter can be integrated with the pulse sensor on the wrist and the RF receiver to a PC or laptop that has the installed version of the *Intelli-Alert* ™ interface system.

4.4 Base Station – Telephony & Voice Recording / Playback

The interface designed for the *Intelli-Alert* ™ system is adequate enough to handle most of the functions required of the system. A possible improvement could be to develop a matching telephone answering system specific to the *Intelli-Alert* ™ emergency response system, catering to the needs of the users. An embedded base station version is also planned.

5. BUDGET DETAILS AND TIME CONSTRAINTS

5.1 Budget

Table 1 contains the estimated cost and the cost of the project up to December 15, 2003.

Table 1: Table of Estimated Cost vs. Actual Cost

Required Materials	Estimated Cost	Actual Cost
Sensors	60.00	55.00
OOPic Microcontroller	Donated	Donated
RF Transmitter / Receiver	80.00	Donated
LCD	Donated	Donated
Miscellaneous Items	40.00	10.00
Contingency	27.00	-
Total	**207.00**	**65.00**

The cost listed above is the amount spent working on the prototype of the *Intelli-Alert* ™ system. We were fortunate enough to borrow certain items, such as the RF Transmitter and Receiver from Dr. Andrew Rawicz. Some of the items came from our “resource-guy” Tristen too. Because our major devices were borrowed, our estimated cost is lower than the actual cost of the project. If we had to purchase everything, the total cost would have been approximately $600.00. We estimate the mass produced version to cost approximately $100 per unit.

5.2 Time

A Gantt chart is shown below in which the expected times to completion are indicated in blue and the actual time taken to complete are indicated in pink.

It is amazing that we managed to stick to our schedule most of the time. This is because we realised early that time management is important in order to achieve the expected deadline of the 2nd week of December (after our exams). Only our implementation and integration took longer than expected due to some problems we face in getting all the different components working together.

Despite this slippage, we had the first successful trial on December 2nd. Since then, more modifications have been taking place to improve robustness of the device. We also used this time to fix any major software and firmware bugs.

Our group was the second ENSC 340 group to demonstrate our device on Monday, December 15, 2003. By adhering to the schedule, we were able to distribute the frustration and pressure throughout the entire semester.

ID	**Task Name**	**Weeks of Sept.**				**Weeks of Oct.**				**Weeks of Nov.**				**Weeks of Dec.**			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Research	🟦	🟦	🟦	🟦	🟦	🟦	🟦									
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪								
2	Proposal	🟦	🟦	🟦													
		🟪	🟪	🟪													
3	Functional Specification			🟦	🟦	🟦											
				🟪	🟪	🟪											
4	Design Specification					🟦	🟦	🟦	🟦								
							🟪	🟪	🟪								
5	Implementation / Integration				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
						🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
6	Debugging Modification					🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦			
						🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪		
7	Documentation / Website	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪				

Figure 3: Gantt chart

6. INTER-PERSONAL AND TECHNICAL EXPERIENCES

6.1 Yang Pan

Working on this project has been a great experience. I not only learned a lot of technical skills, but also gained insights into group dynamics and the documentation process.

On the technical side, I managed to learn and program in visual basic. Since we want our program to be as integrative as possible, I have to learn how to use VB ActiveX control objects to perform certain tasks. Given that I have never done any similar work before, this task has been a challenging one. I am glad I managed to beat the learning curve and get the program working in the end.

As an engineer, I always dislike writing documentations. However, completing this project has taught me the importance of documentation. For instance, by writing the functional specification, I had a much better idea of where this project is going and what I need to do next. Without proper documentation, the whole process would not have been as smooth as it has been.

Working in a group is not easy as well. Each group member has his/her own schedule, strengths/weaknesses. This project taught me how to best manage time in a group setting and how to distribute tasks according to each group member's strengths and weaknesses.

6.2 Tristen Georgiou

The experience gained from this project has been invaluable. As a team we pulled through well, and as an individual I gained much technical experience.

The most important lesson I have learned is that the theory we learn in class can only take you a certain distance, and from there experience is the most important. Many times when I was working on the pulse sensor device, the expected theoretical results were non-existent, and at other times favourable results occurred when theory said it was impossible. This being said, most times a reasonable explanation could be formulated, and the project could go on. For example, I built a circuit using my theoretical skills from my education, and found that the circuit would not work as I expected, but when I moved the probes to a different point on the circuit, favourable results would arise. The reason behind this was due to the fact that my circuit's impedance was so high, that the probe's impedance could not be ignored; the simple act of observation changed that which was being observed (think that is a famous quote, but not sure where I heard it).

I also learned much about product documentation, and I am sure that the functional and design specification knowledge I gained will be invaluable if I ever design my own product.

Finally, I think my teamwork skills have improved. It was challenging working with a group with very different backgrounds, but we all learned how to overcome such minor problems and work out any differences we might have. Overall I think we worked very well as a group, and we have all gained better personal skills because of it.

6.3 Hashina Parveen

Working on the *Intelli-Alert* ™ emergency response system for GiveLife Systems, Inc. has taught me a lot in terms of interpersonal relations and teamwork. The four of us came from very different backgrounds. Yet, we somehow managed to work together as a team getting this great product, *Intelli-Alert* ™ into fruition.

I was in charge of working on the interface of the *Intelli-Alert* ™ system. Being a person who is not into computer software, learning a new programming language, Microsoft Visual Basic 6.0 (VB) was a challenge itself. To include TAPI programming into VB was even a tougher one. At times I almost gave up. I am thankful that Yang Pan helped me out towards the end, or else you would not have an interface for the system.

In terms of interpersonal relations, Melody and I were good friends from back home in Singapore before we worked together for ENSC 340. To do a project module with people you did not really know was tough especially when you want to communicate something important with them. Having a shy personality with guys did not help much either. Yang and Tristen are wonderful guys who did most of the work and did not bother us gals much. At times, it felt that they were leaving us out of the loop but I understand that they work better by their own and only want to inform us when things are working well rather than all the time.

I am glad that things turned out well in the end. The four of us are friends now and the prototype turned out great. Thanks to the three of you, for all you have taught me (Yang, Tristen and Melody) and also to the professors, Andrew and Steve and TAs who guided us along the way. If I was a pain at times, I am sorry but if I was an inspiration to you, I am happy to be of serve.

6.4 Melody Guo

This project has been an extremely challenging but exciting experience for me. This is the first project-based course that I have taken. At the beginning of this semester, I heard that this course was going to be super tough. Well, it turns out quite true. Being an engineering third year student with little practical experience, especially in hardware is obviously disadvantageous in this kind of project. Thanks to other team members. Their hard working really inspired me.

From a technical perspective, I increased my knowledge of programming in Visual Basic and learned how to integrate software with hardware. To me, Visual Basic is a new programming language. Doing voice recording and play back was not an easy job initially. After going through a tough trial and failure process, I eventually got it done. It taught me how to deal with frustration too.

Interpersonally, I have learned to be understanding. Working in a group is much more challenging than individually, especially when 4 of us come from different backgrounds. I feel extremely lucky to have worked with my team mates. It’s a quite eye-opening experience.

Overall, It has been a very precious experience and I enjoyed working very much with my group.

APPENDIX A: SOURCE CODE

Visual Basic Code (PC Program)

```
Option Explicit

Public SerialIn As String
Public i%
Public TimerCounter As Integer
Public counter% 'counter used to record which phone # to call
Public timesPlayed%
Public msgPath As String
Public emptyCount As Integer '# of empty tel# memory
Public AlertNum As Integer

Private Type MCI_WAVE_DELETE_PARMS  'parameter structure needed for MCI_DELETE
   Callback As Long
   From As Long
   To As Long
End Type

Private Declare Function mciSendCommand Lib "winmm.dll" Alias "mciSendCommandW" ( _
                              ByVal wDeviceID As Long, _
                              ByVal uMessage As Long, _
                              ByVal dwParam1 As Long, _
                              ByVal dwParam2 As Any) As Long 'dwParam2 is the addr of the param struct

Private Declare Function mciGetErrorString Lib "winmm.dll" Alias "mciGetErrorStringW" ( _
                              ByVal dwError As Long, _
                              ByVal lpstrBuffer As String, _
                              ByVal uLength As Long) As Long

Private Declare Function GetAddrOf Lib "kernel32" Alias "MulDiv" ( _
                              nNumber As Any, _
                              Optional ByVal nNumerator As Long = 1, _
                              Optional ByVal nDenominator As Long = 1) As Long

Private Declare Function timeGetTime Lib "winmm.dll" () As Long

'declare a TAPI variable
Private currCall As ITAPICall

Private Sub Form_Load()

   Move (Screen.Width - Width) \ 2, (Screen.Height - Height) \ 2

   '-------------initialize some global variables
   TimerCounter = 0
   counter = 0
   timesPlayed = 0
   emptyCount = 0
   msgPath = App.path & "\msg.wav"
   AlertNum = 0

   '-----------read stored phone number from file to memory-----------
   Dim phonenumber$(3)
   Dim path$
   path = App.path & "\number.dat"

   Open path For Input As #1
   i = 0
   Do While Not EOF(1) 'if file not empty, load the numbers
      Input #1, phonenumber$(i)
      i = i + 1
   Loop
   Close #1
   If i <> 0 Then
      i = i - 1
   End If

   Dim MemCounter%
   For MemCounter = 0 To i
         lblMem(MemCounter).Caption = phonenumber$(MemCounter)
   Next
   '==================================================================


   '------------------initialize serial port----------------------
   'Use COM1.
   SerialPort.CommPort = 1
   '4800 baud, no parity, 8 data, and 1 stop bit.
   SerialPort.Settings = "4800,N,8,1"
   'Tell the control to read entire buffer when Input is used.
   SerialPort.InputLen = 0
   'Open the port.
   SerialPort.PortOpen = True
   '===================================================================


   '----------------initialize MMcontrol--------------------------------
   With MMControl
      .Notify = True
      .Wait = True
      .FileName = App.path & "\msg.wav"
      .TimeFormat = 8     'in bytes
      .RecordMode = mciRecordOverwrite
      .Command = "Open"  'must do "open" after setting the properties!!! (spent 2 hrs to figure this shit out!)
   End With
   '======================================================================

   Dim line As ITAPILine
   mTAPIEx.Initialize   'Initialize
   For Each line In mTAPIEx.Lines  'enumerate the line devices
      If line.Caps.Media_Modes And TAPIMEDIATYPE_AUDIO Then
         cboLineSel.AddItem line.DeviceName 'check the device audio caps
         cboLineSel.ItemData(cboLineSel.ListCount - 1) = line.DeviceID
      End If
   Next
   cboLineSel.ListIndex = 0 'make the first device to be the default TAPI device

End Sub

Private Sub Form_Unload(Cancel As Integer)
   MMControl.Command = "Close"
   mTAPIEx.UnInitialize
End Sub

Private Sub cmdExit_Click()
   Dim pintReturn As Integer
   pintReturn = MsgBox("Do you really want to quit?", vbOKCancel + vbQuestion + _
                  vbDefaultButton2, "IACC")
   If pintReturn = vbOK Then
      Unload Me
   End If
End Sub

Private Sub cmdKeypad_Click(Index As Integer)
' Responds to mouse click on keypad buttons on screen
```

```
    If Index >= 0 And Index <= 9 Then
        If Len(lblDisplay.Caption) < 11 Then
            lblDisplay.Caption = lblDisplay.Caption & Index
        End If
    ElseIf Index = 10 Then
        If Len(lblDisplay.Caption) < 11 Then
            lblDisplay.Caption = lblDisplay.Caption & "*"
        End If
    ElseIf Index = 11 Then
        If Len(lblDisplay.Caption) < 11 Then
            lblDisplay.Caption = lblDisplay.Caption & "#"
        End If
    ElseIf Index = 12 Then
        If Len(lblDisplay.Caption) > 0 Then
            lblDisplay.Caption = Left(lblDisplay.Caption, Len(lblDisplay.Caption) - 1)
        Else
        End If
    Else    'index = 13 the clear key
        lblDisplay.Caption = ""
    End If

End Sub

Private Sub cmdKeypad_MouseDown(Index As Integer, Button As Integer, _
                        Shift As Integer, x As Single, Y As Single)
    cmdKeypad(Index).BackColor = &H80FFFF
End Sub

Private Sub cmdKeypad_MouseUp(Index As Integer, Button As Integer, _
                        Shift As Integer, x As Single, Y As Single)
    cmdKeypad(Index).BackColor = &HC0FFC0
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
' responds to keyboard press. it relates keyboard keys 0-9, *, # and backspace to
' corresponding keys on the keypad on screen. all other key presses are ignored

    Select Case KeyAscii
    Case 48
        cmdKeypad_Click (0)
    Case 49
        cmdKeypad_Click (1)
    Case 50
        cmdKeypad_Click (2)
    Case 51
        cmdKeypad_Click (3)
    Case 52
        cmdKeypad_Click (4)
    Case 53
        cmdKeypad_Click (5)
    Case 54
        cmdKeypad_Click (6)
    Case 55
        cmdKeypad_Click (7)
    Case 56
        cmdKeypad_Click (8)
    Case 57
        cmdKeypad_Click (9)
    Case 42
        cmdKeypad_Click (10)
    Case 35
        cmdKeypad_Click (11)
    Case 8
        cmdKeypad_Click (12)
    Case Else
        KeyAscii = 0
    End Select

End Sub
```

```
Private Sub cmdMem_Click(Index As Integer)

    lblMem(Index).Caption = lblDisplay.Caption

    lblDisplay.Caption = ""

    Dim path$
    path = App.path & "\number.dat"

    Open path For Output As #1
    Dim counter%
    For counter = 0 To 3
        Print #1, lblMem(counter).Caption
    Next
    Close #1

End Sub


Private Sub MMControl_PlayClick(Cancel As Integer)

    With MMControl
        .Command = "Prev"   'rewind to the beginning of the sound clip
        .Command = "Play"
    End With

End Sub

Private Sub MMControl_RecordClick(Cancel As Integer)

    MMControl.Command = "Prev"

    '1st, the old wave file has to be erased
    Const MCI_DELETE As Long = &H856
    Const MCI_WAIT As Long = &H2&

    Dim del_param As MCI_WAVE_DELETE_PARMS
    del_param.Callback = 0
    del_param.From = 0
    del_param.To = MMControl.Length
    Debug.Print "wave clip length = " & MMControl.Length

    Dim Addr As Long: Addr = GetAddrOf(del_param)   'get the address of del_param

    Dim retval As Long
    retval = mciSendCommand(MMControl.DeviceID, MCI_DELETE, MCI_WAIT, Addr)
    'Debug.Print MMControl.DeviceID

    'used to get and print the error msg====================
    Dim ErrorMsg As String
    ErrorMsg = String(1024, vbNullChar)       ' Reserve space for message
    mciGetErrorString retval, ErrorMsg, 1024    ' Get message
    ErrorMsg = Left(ErrorMsg, InStr(1, ErrorMsg, vbNullChar) - 1)  'trim away remaing null chars
    Debug.Print ErrorMsg   ' print message
    '======================================================

    MMControl.Command = "Save"
    MMControl.Command = "Record"

End Sub

Private Sub MMControl_StopClick(Cancel As Integer)
    With MMControl
        .Command = "Stop"
        .Command = "Save"
        .Command = "Prev"
    End With
    Debug.Print "wave clip length = " & MMControl.Length
```

```
End Sub


Private Sub SerialPort_oncomm()

   Dim cominput As String


   Select Case SerialPort.CommEvent

  ' Errors
      Case comEventBreak   ' A Break was received.
         'ListStatus.AddItem "Break Error!"
      Case comEventFrame   ' Framing Error
         'ListStatus.AddItem "Framing Error!"
      Case comEventOverrun   ' Data Lost.
         'ListStatus.AddItem "Data Lost!"
      Case comEventRxOver   ' Receive buffer overflow.
         'ListStatus.AddItem "Receive buffer overflow!"
      Case comEventRxParity   ' Parity Error.
         'ListStatus.AddItem "Parity Error!"
      Case comEventTxFull   ' Transmit buffer full.
      Case comEventDCB   ' Unexpected error retrieving DCB
         'ListStatus.AddItem "Unexpected error retrieving DCB!"

  ' Events
      Case comEvReceive   ' Received RThreshold # of chars.
         cominput = SerialPort.Input
         'ListStatus.AddItem cominput
         'Dim askii As Integer
         If (cominput = "0") Then
            If AlertNum = 0 Then
               AlertNum = 1
               ListStatus.Clear
               ListStatus.AddItem "Emergency alert received! Start counting down!"
               txtTimeLeft.Enabled = True
               Timer1.Enabled = True
               cmdCancel.Enabled = True
            ElseIf AlertNum = 1 Then
               AlertNum = 0
               ListStatus.AddItem "Emergency+ alert received!"
               txtTimeLeft.Enabled = False
               txtTimeLeft.Text = 30
               Timer1.Enabled = False
               cmdCancel.Enabled = False
               cmdCall_Click
            End If
         ElseIf (cominput = "1") Then
            ListStatus.AddItem "Emergency+ alert received!"
            txtTimeLeft.Enabled = False
            txtTimeLeft.Text = 30
            Timer1.Enabled = False
            cmdCancel.Enabled = False
            cmdCall_Click
         End If
      Case comEvSend   ' There are SThreshold number of characters in the transmit buffer.
      Case comEvEOF   ' An EOF charater was found in the input stream
  End Select

End Sub



Private Sub Timer1_Timer()

   Dim TimeLeft As Integer
   TimeLeft = 28 - 2 * TimerCounter

   If TimerCounter < 15 Then 'not 30s yet
```

```
        Beep
        txtTimeLeft.Text = TimeLeft
    Else    '30s has passed
        ListStatus.AddItem ("making calls.....")
        Timer1.Enabled = False
        ListStatus.AddItem ("timer disabled.")
        txtTimeLeft.Enabled = False
        txtTimeLeft.Text = 30
        TimerCounter = 0
        cmdCall_Click   'make the call
    End If

    TimerCounter = TimerCounter + 1

End Sub

Private Sub cmdCancel_Click()

    Timer1.Enabled = False
    cmdFake.Enabled = True

    txtTimeLeft.Enabled = False
    txtTimeLeft.Text = 30
    TimerCounter = 0
    ListStatus.AddItem ("Alert Cancelled by user.. Timer disabled")

End Sub

Private Sub cmdFake_Click()
    cmdFake.Enabled = False
    cmdCancel.Enabled = True

    ListStatus.Clear
    ListStatus.AddItem ("alert received")
    txtTimeLeft.Enabled = True
    Timer1.Enabled = True
End Sub


'--------------------------------------------------------------------------------
' TAPI related code start from here------------------------------------------------
'--------------------------------------------------------------------------------

Private Sub cmdCall_Click()

    Dim pintReturn As Integer
    Dim line As ITAPILine
    Dim ret As Integer

    cmdCancel.Enabled = False

    If cboLineSel.ListIndex >= 0 Then
        Set line = mTAPIEx.GetLineFromDeviceID(cboLineSel.ItemData(cboLineSel.ListIndex))
        If line.Open Then
            If lblMem(counter).Caption <> "" Then 'if there is number stored
                Set currCall = line.MakeCall(lblMem(counter).Caption)
                'Open the line device and make the call
                ListStatus.AddItem ("Making a call: call#" & counter + 1)
                If counter < 4 Then counter = counter + 1 'roll up counter #
                cmdHangUp.Enabled = True
                cmdCall.Enabled = False
            Else    'else skip to next number
                emptyCount = emptyCount + 1
                If counter < 3 Then
                    ListStatus.AddItem ("Number " & counter + 1 & " doesn't exist")
                    counter = counter + 1 'roll up counter #
                    ListStatus.AddItem "next call #" & counter + 1
                    ListStatus.AddItem "=============="
                    cmdCall_Click
                Else
                    ListStatus.AddItem ("Number " & counter + 1 & " doesn't exist")
                    If emptyCount = 4 Then
                        ret = MsgBox("You didn't save any number, dumb ass! No one's gonna come!", vbOKOnly, "Dumb Ass!")
                        If ret = vbOK Then
                            ListStatus.Clear
                            cmdFake.Enabled = True
                        End If
                    Else
                        ret = MsgBox("All available numbers called! Help is on its way!", vbOKOnly, "Hold on Granny!")
                        If ret = vbOK Then
                            ListStatus.Clear
                            cmdFake.Enabled = True
                        End If
                    End If
                    Debug.Print emptyCount
                    emptyCount = 0
                    counter = 0
                End If
            End If
        End If
    Else
        pintReturn = MsgBox("Please pick a device", vbOKOnly)
    End If

End Sub

Private Sub HangUp()
    cmdHangUp.Enabled = False
    cmdCall.Enabled = True
    cmdCancel.Enabled = True
    currCall.Drop
    ListStatus.AddItem "Call Ended!" & counter
End Sub
Private Sub cmdHangUp_Click()

    Call HangUp
    'reset counters
    counter = 0
    timesPlayed = 0
    emptyCount = 0

    ListStatus.Clear
    ListStatus.AddItem "Call Ended by user!"

End Sub

Private Sub mTAPIEx_OnConnected(ByVal m_Call As TAPIEXLibCtl.ITAPICall, _
                    ByVal ConnectedMode As TAPIEXLibCtl.LINECONNECTEDMODE)

    ListStatus.AddItem ("Connected and waiting, call#" & counter)
    Call Pause(10000)    'wait 10s for connection to be mad and the otherend to pick up the phone
    ListStatus.AddItem ("Connected & paused 10s, call#" & counter)
    'ListStatus.AddItem msgPath

    Dim rt As Boolean
    rt = currCall.PlaybackFile(msgPath)
    Debug.Print rt
    If rt Then ListStatus.AddItem "Playing message"
    timesPlayed = 1

End Sub

Private Sub mTAPIEx_OnPlayBackComplete(ByVal m_Call As TAPIEXLibCtl.ITAPICall)

    Dim ret As Integer
    'ListStatus.AddItem "playcompleted: " & timesPlayed

    If (currCall.ActivePlayBackFile = msgPath And timesPlayed < 3) Then
```

```
        Call Pause(3000, True)
        currCall.PlaybackFile msgPath
        timesPlayed = timesPlayed + 1
        'ListStatus.AddItem "Play again: " & timesPlayed
    Else 'msg has been played for 3 times
        ListStatus.AddItem ("going to hang up call#" & counter)
        HangUp     'hang up the phone
        Call Pause(5000)    'pause for 5s
        ListStatus.AddItem "Call Ended and 5s paused! call#" & counter
        ListStatus.AddItem "=============="

        If counter < 4 Then
            cmdCall_Click   'if not all 4 numbers have been called yet, call again
        Else
            ret = MsgBox("All available numbers called! Help is on its way!", vbOKOnly, "Hold on Granny!")
            If ret = vbOK Then
                ListStatus.Clear
                cmdFake.Enabled = True
            End If
            counter = 0 'reset counter
            emptyCount = 0 'reset counter
        End If
    End If

End Sub

Private Sub Pause(ByVal mSecs As Long, Optional bYield As Boolean = True)
    Dim startTime As Long

    startTime = timeGetTime()
    Do While timeGetTime < startTime + mSecs
        If bYield Then
            DoEvents
        End If
    Loop

End Sub
```

Firmware (C Code)

```
// GiveLife Systems
// Microcontroller Source Code
// ---------------------------
// Version History
// Version 1.0
//

oSerial LCD = new oSerial; // LCD serial line
oDIO1 Pulse = new oDIO1; // Sensor input line
oOneShot RiseEdge = new oOneShot; // detects rising edge from sensor
oEvent GetPulse = new oEvent; // event is triggered by rising edge
oCounter Counts = new oCounter; // counter that increments at 60 Hz
oByte NumCounts = new oByte; // number of 60 Hz counts between rising edges
oByte BPM = new oByte; // beats per minute
oEvent TimeOut = new oEvent; // if counter times out (after 2 seconds)
oWire TOLink = new oWire;
oNibble TOCounter = new oNibble; // Counts number of TimeOuts

oDIO1 button = new oDIO1; // Button IO Line
oWire ButLink = new oWire; // links the button status to the interrupt
oEvent ButPressed = new oEvent; // button pressed interrupt
oSerialX ToComp = new oSerialX; // serial line to wireless device

Sub void main(void)
{
  Init; // initializations
}

Sub void Init(void)
{
  OOPic.PullUp = cvOn; // activate pullup resistors on I/O lines 8-15
  NumCounts = 0; // reset counter
  LCDInit;  // initialize LCD
  SensorInit;  // initialize sensor
  CounterInit;  // initialize objects for counting on rising edge
  ButtonInit; // initalize button IO Line
  ToComp.IOLineS = 13; // IOLine for serial line
  ToComp.IOLineF = 0; // disables flow control
  ToComp.Baud = cv4800; // sets baud rate
  ClrScn; // clear LCD screen
  GoToStart;
  LCD.String = "Insert Finger...";
}

Sub void SensorInit(void)
{
  Pulse.IOLine = 15;
  Pulse.Direction = cvInput;
}

Sub void CounterInit(void)
{
  Counts.Output.Link(NumCounts); // counter output to NumCounts variable
  Counts.ClockIn1.Link(OOPic.Hz60); // set counter to count at 60 Hz
  Counts.Mode = 0; // no clock ANDed with ClockIn1
  Counts.Direction = cvPositive; // set counter to increment
  Counts.Operate = cvFalse; // don't operate counter yet
  RiseEdge.Input.Link(Pulse); // link pulse to input of edge detector
  RiseEdge.InvertIn = cvTrue; // invert the pulse input
  RiseEdge.Output.Link(GetPulse.Operate); // link to interrupt code
  RiseEdge.Operate = cvTrue; // start detecting rising edge
  TOLink.Input.Link(NumCounts.MSB);
  TOLink.Output.Link(TimeOut.Operate); // Links MSB of NumCounts to TimeOut event
  TOLink.Operate = cvTrue;
}

Sub void ButtonInit(void)
{
  button.IOLine = 14;
  button.Direction = cvInput; // make it an input
  ButLink.Input.Link(button);
  ButLink.Output.Link(ButPressed.Operate);
  ButLink.InvertIn = cvTrue; // invert input, input goes low when button pressed
  ButLink.Operate = cvTrue;
}

// interrupt code once a rising edge is detected
Sub void GetPulse_CODE(void)
{
  TOCounter = 0; // reset time out counter
  if(Counts.Operate == 1) // if timing
  {
    if(NumCounts < 30) // detected noise, ignore
      return;
    Counts.Operate = 0; // stop timer
    BPM = 3600/NumCounts; // calculate BPM
    GoToStart; // put LCD cursor at first position
    LCD.String = "Pulse ";
    LCD.String = Str$(BPM.Value); // display beats per minute
    LCD.String = "/min ";
    GoToBot;
    LCD.String = "                "; // clear bottom line
    NumCounts = 0; // reset counter
    Counts.Operate = 1; // restart timer
  }
  else // not timing, start timer
  {
    NumCounts = 0; // reset counter
    Counts.Operate = 1; // start timer
  }
}

Sub void TimeOut_CODE(void)
{
  GoToStart; // put LCD cursor at first position
  LCD.String = "Pulse 00000/min"; // display lack of heart sensing
  TOCounter++;
  GoToBot;
  LCD.String = "Help in = ";
  LCD.String = Str$(10-(2*TOCounter.Value));
  if(TOCounter > 4) // no pulse for 10 seconds
  {
    RiseEdge.Operate = cvFalse; // disable interrupts
    Counts.Operate = cvFalse;
    ButLink.Operate = cvFalse; // disble button
    ToComp.Value = 49; // send an "1"
    OOPic.Delay = 50; // Delay for half a second;
    ClrScn;
    GoToStart;
    LCD.String = " HEART FAILURE";
    GoToBot;
    LCD.String = "  Calling help.";
  }
}

Sub void ButPressed_CODE(void) // Button interrupt code
{
  RiseEdge.Operate = cvFalse; // disable interrupts
  Counts.Operate = cvFalse;
  ClrScn;
  GoToStart;
  LCD.String = " Button Pressed";
  GoToBot;
  LCD.String = "  Calling help.";
  ToComp.Value = 48; // send an "0"
  OOPic.Delay = 50; // Delay for half a second;
  ClrScn;
}

// LCD code from here to bottom
Sub void LCDInit(void)
{
  LCD.Baud = cv9600;
  LCD.Operate = cvTrue;
}

Sub void ClrScn(void)
{
  LCD.Value = 254; // Send Prefix
  LCD.Value = 88;
}
Sub void GoToStart(void)
{
  LCD.Value = 254; // Send Prefix
  LCD.Value = 72;
}
Sub void GoToBot(void)
{
  LCD.Value = 254; // Send Prefix
  LCD.Value = 71;
  LCD.Value = 1; // column
  LCD.Value = 2; // row
}
```