Calibration System for Pressure Sensitive Cardiovascular Catheters

Danielle Ebben, Martin Grasse, Anthony Wampole, Erik Yusko, and Anita Zarebi

Submitted May 11, 2007

Blood pressures within the heart and surrounding vessels are often measured using catheters that are equipped with pressure transducers. The accuracy of blood pressure measurements from these catheters has been called into question as calibration data is not provided by the manufacturer. A product is needed to calibrate and verify the pressure transducer measurements while simulating in vivo conditions.

A product was developed to encase the catheter in saline solution while achieving pressures up to 200mmHg. A computer-controlled air pump increases pressure in an air-tight chamber in response to user input. Pressure in the chamber is monitored with an additional pressure transducer imbedded in the tank. A feedback loop between this pressure sensor and the computer program maintains the desired pressure.

Introduction and Theory

In order to measure internal pressures within the cardiovascular system, catheters that have several pressure transducers along their length are often used. During trials, the pressure transducers on the catheter are sometimes found to be out of calibration, and there is no good method to recalibrate them.

To effectively recalibrate the catheter, a system was designed that incorporates an acrylic pressure chamber, a control circuit with a data acquisition device, and a computer interface. The system uses feedback loops to control the pressure and temperature inside of the chamber allowing the user to calibrate within a wide range of both pressures and temperatures.

The calibration system was designed with several main design constraints in mind. First of all, the system must be able to reach and maintain pressures between 0 and 200 mmHg, and must be damage-resistant to 300 mmHg. Secondly, it must be able to maintain constant temperature (the system will most often be used at body temperature: 37˚ Celcius). Also, during the calibration process, the catheter's pressure transducers must be shielded from light, as excess light exposure will affect pressure measurements. Finally, the design must be completely safe for the user and bystanders.

Pressure Chamber

In order to maintain an environment at a constant pressure for catheter testing, an air-tight chamber was built out of 0.375 inch acrylic. The chamber is 18 inches long, 4 inches wide, and 2 inches high, with the catheter mounted 1 inch from the bottom. All sides are bound with ethylene dichloride.

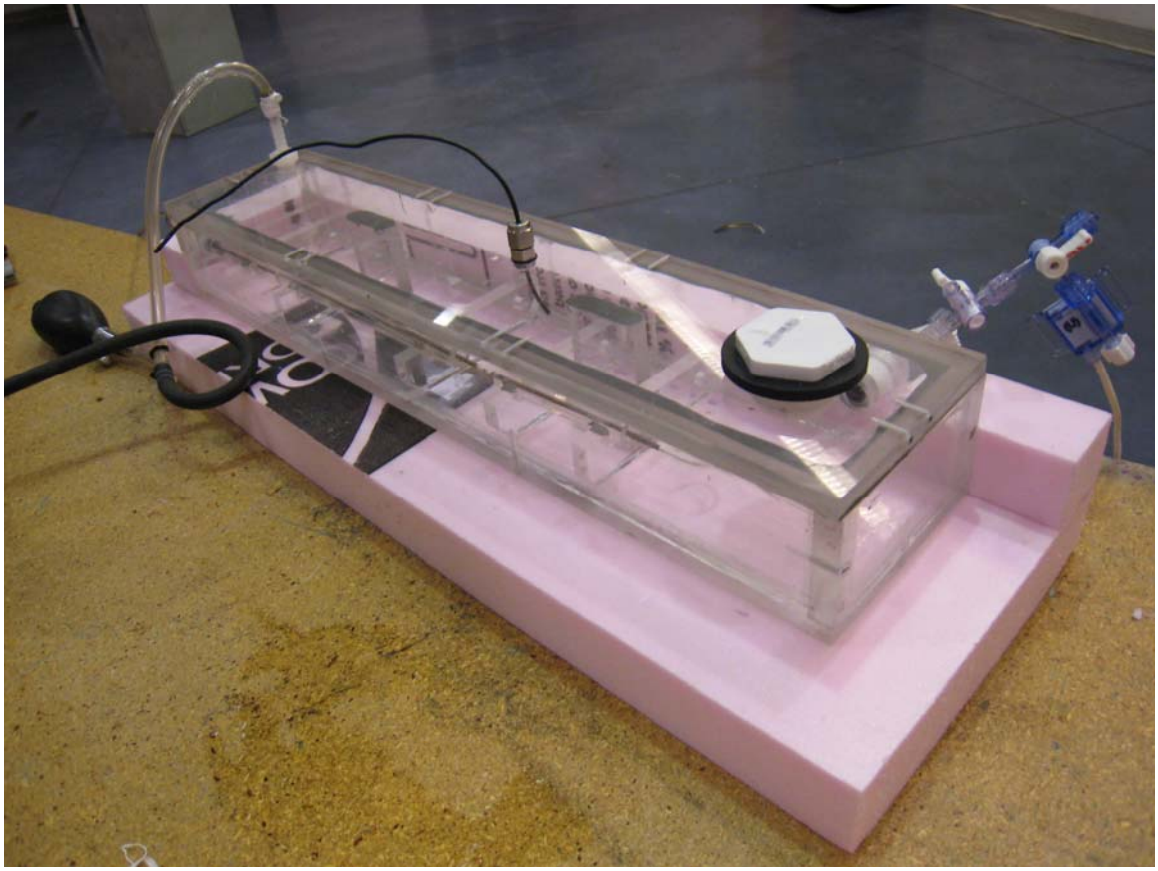

Figure 1: Pressure Chamber

To prevent air leaks, the edges of the chamber are sealed with silicone caulk and all inputs into the chamber are sealed with o-rings and/or Teflon tape. A preliminary prototype had a removable lid to facilitate placement of the catheter. In an attempt to seal the lid, both a rubber o-ring material and cork gasket were placed around the top. Both of these sealing methods failed at very low pressures and rapid air leaking was observed. Therefore, in the second generation prototype the top on the chamber is permanently sealed with epoxy.

A 1.156 inch hole (1 inch NPT) was drilled in the top of the chamber through which the saline can be added until the catheter is completely immersed. The hole is threaded so that it may be plugged using a 1 inch NPT plug. The top of the chamber also contains a barbed hose adaptor which connects to the air pump and a 0.125 inch diameter by 0.125 inch NPT tube fitting through which a thermistor is inserted to mediate temperature control.

The catheter is supported at the same level as a pressure sensor that is mounted in the wall of the chamber. The catheter supports are made out of blocks of acrylic and 0.5 inch outer diameter – 0.25 inch inner diameter acrylic tubing. Gaps which coincide with the

location of pressure sensors on the catheter are left between each of the supports so that the catheter sensors are fully exposed to pressurized saline. In order to guide the catheter through the mounts, a size 8 French sheath inducer is used. The sheath is inserted through a 0.185 inch inner diameter rotating hemostasis valve and passes into the first tubular mount in the interior of the box. It is then passed through the center of the three mounts until it spans the two gaps. The catheter is inserted into the sheath until the tip of the catheter is just visible at the end of the sheath. While the operator holds the catheter in place, the sheath is retracted and the catheter remains inside the container. Finally, the operator adjusts the catheter so that the pressure transducers are aligned with the gaps in the tubular mounts and tightens the hemostasis valve around the catheter to create a seal.

The chamber and its heating element are encased in a 1.5 inch thick polystyrene jacket, which helps facilitate more efficient temperature control. In addition to insulating the container, the case has the added benefit of blocking light from the photosensitive pressure transducers of the catheter.

Control Circuit

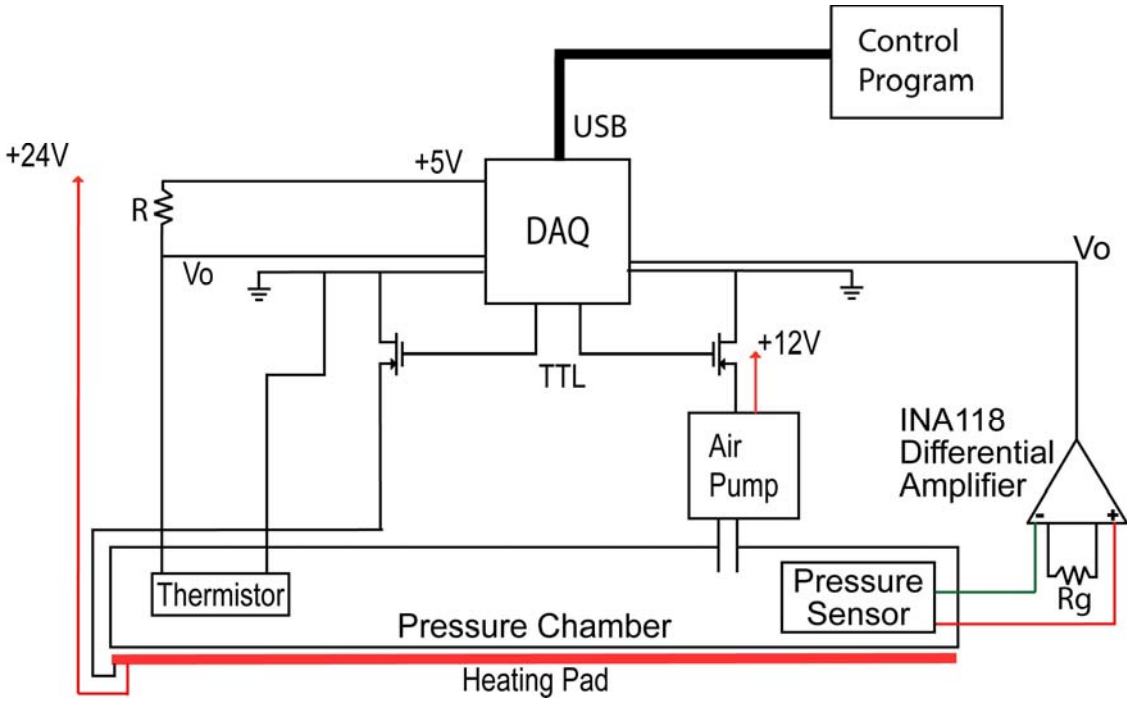

Figure 2: Control Circuit Diagram

The goal of the device is to create an environment that simulates *in vivo* conditions, and provide multiple pressure levels at which to calibrate the pressure transducers on the catheter. To do this, both pressure and temperature must be monitored and controlled inside the chamber.

Sensors

Two sensors are used inside the chamber, one measuring pressure and the other measuring temperature. The pressure sensor is simply a strain gauge in which a differential linear voltage proportional to pressure is achieved. We used an AD Instruments MLT0670 Disposable Blood Pressure Transducer, but any strain gauge or pressure transducer can be used if a voltage-pressure relationship is found. To monitor temperature, a thermistor, which changes resistance proportional to temperature, is used. A 5 volt excitation voltage from the DAQ (described below) is used for both sensors.

Signal Conditioning

In order to ensure the proper resolution is achieved, the voltage change per variable change must be substantial enough to be acquired by the DAQ's 12 bit AD converter. This required the use of an INA118 instrumentation differential amplifier to amplify the differential signal by a gain:

$$G = \frac{50K\Omega}{R_G} + 1$$

where Rg is a resistor added to select a gain. Electrical noise was assumed to be the same in both channels of the pressure sensor wires, thereby eliminated its total effect through differential amplification. Electrical noise was not an issue in the thermistor as temperature changes will not occur rapidly enough to make noise a concern.

Temperature and Pressure Control

To control the temperature inside the chamber, a Minco HR5183R15.6L12BU heating pad was used. The heating pad is turned on or off depending on whether the temperature inside the chamber is above or below the desired temperature. In much the same way, the pressure inside the chamber is controlled with a Sensodyne air pump. During a calibration run, the air pump is turned on or off depending on whether the pressure inside the chamber is above or below the desired pressure. This control mechanism is described in more detail in the software control section of this report.

Data Acquisition

The primary hardware component in communicating between the computer and external circuitry is a Measurement Computing DAQ (model USB-1208LS). Essentially, the DAQ serves as an analog to digital converter for the two analog signals monitored: voltage proportional to pressure and voltage proportional to temperature. The DAQ also provides a highly stable 5 volt power source for exciting various components of the control circuitry. Finally, the DAQ allows digital TTL pulses to be applied to external circuitry to turn items on and off. These TTL pulses are used to switch logic FET transistors on and off, which in turn switch the air pump and the heating pad on and off.

Software Control

To provide a user interface to monitor current pressure and temperature and to control the outputs to the air pump and heating pad, a software control program was developed using the Visual Basic .NET framework.

Menus

Two menus comprise the interface between the user and the calibration device. The first is the main menu, which allows for user input of values as well as displays information relevant to calibration. Through the main menu, it is possible to access the installation settings menu, which allows the user to input voltage relationships as well as set the temperature the device will be maintained at throughout the course of calibration.

Main Menu

Shown in Figure 3**,** the main menu allows the user to enter desired pressures at which the catheter can be calibrated. The main menu also displays the current temperature and pressure inside the chamber.

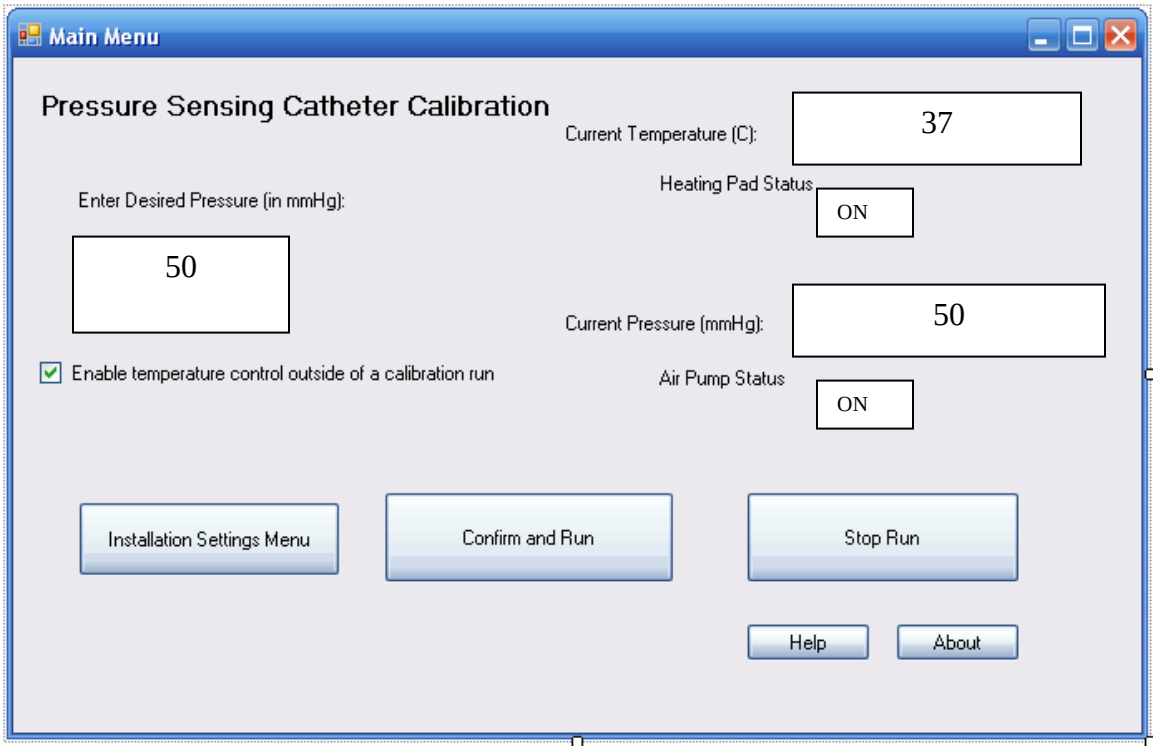

Figure 3: Main Menu of Software Interface

Installation Settings Menu

Accessible through the main menu, the installation settings menu allows the user to enter the voltage relationships specific to the pressure and temperature sensors in use, as shown in Figure 4. Algorithms that were developed using a linear relationship between the variable being monitored and the voltage generated from the sensor are then used to convert the voltage to a pressure or temperature value. Also in the installation settings menu is an entry box where the user can change the desired temperature of the chamber.

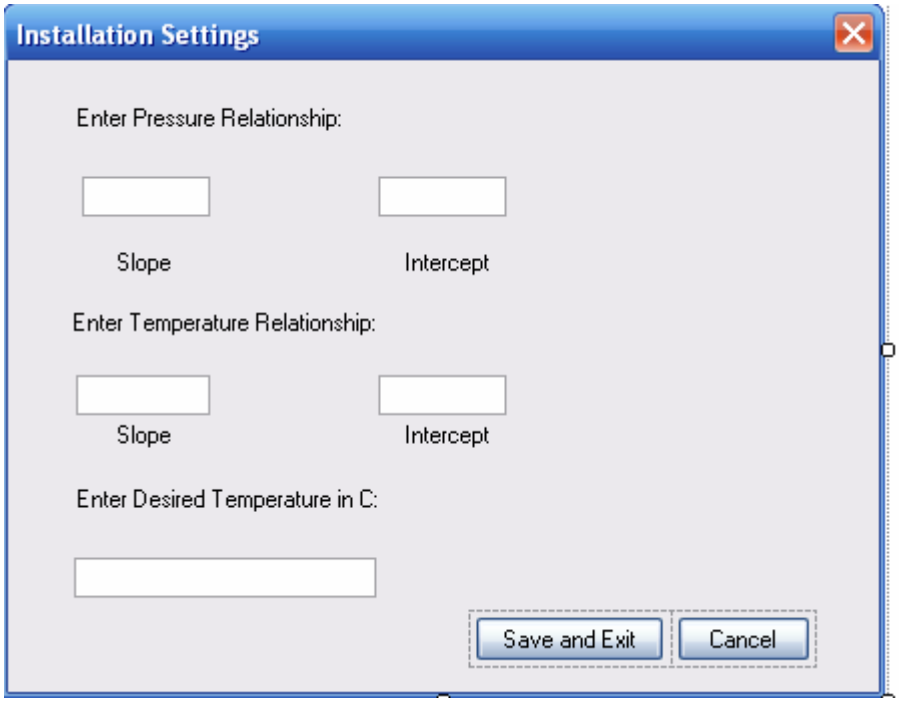

Figure 4: Installation Settings User Interface

Feedback Loop

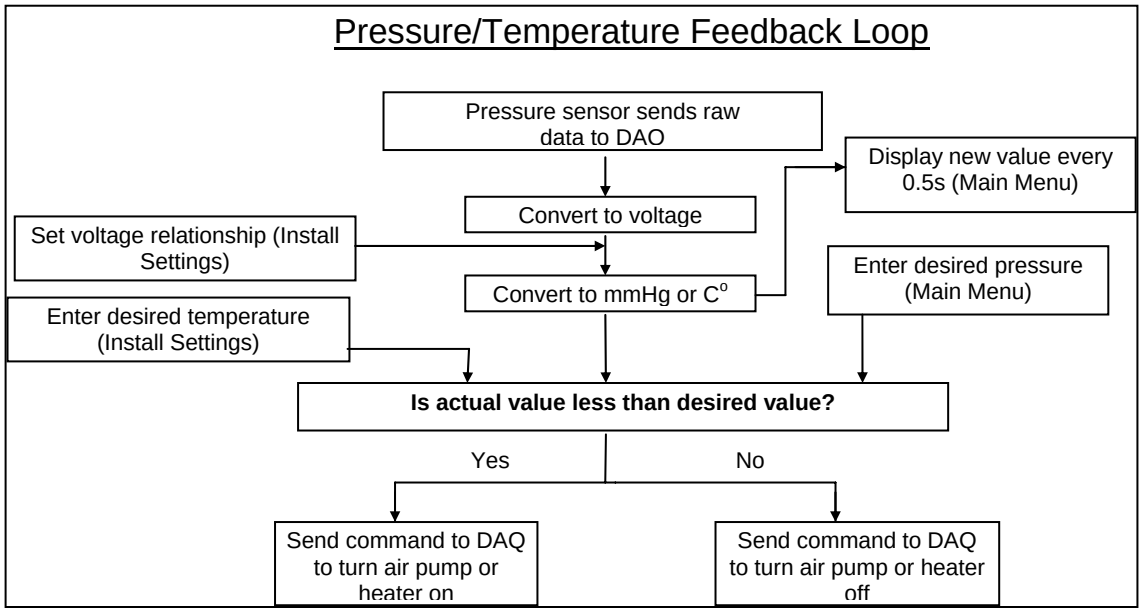

Figure 5: Logic Flowchart of Feedback Loop

The overall feedback loop is outlined in the logic flowchart in Figure 5. This outlines the basic logic behind the control software. See Appendix for the program code.

Testing

Efficacy of the total design depends heavily on the ability to create a virtually air-tight chamber. To test the chamber's air retention, it was connected to the hardware as it will be when the product is being used for calibration. The only difference was that instead of connecting the air pump into the barbed port where the hose would typically connect, the port was connected to the end of a manometer and the connection was secured with Teflon tape. Using the manometer bulb, the chamber was pressurized to four incremented pressures (50, 100, 150, and 200 mmHg). Pressure readings were then taken at 30 second intervals for a period of five minutes. Two trials were done at each of the four pressures and these results are shown graphically in Figure 6. All leak rates were less than 1.8 mmHg per minute, with the fastest leak rates occurring at higher pressures. These slow air leaks are easily within acceptable tolerance ranges and are expected to improve when the device is firmly connected to the air pump rather than the manometer.

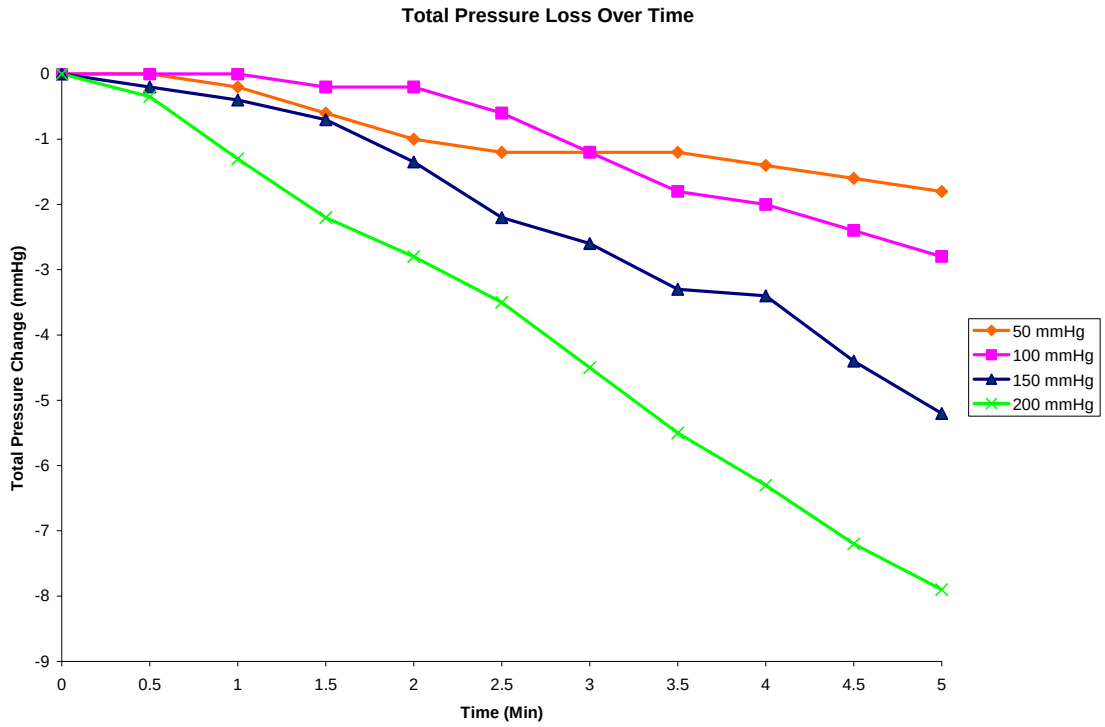

Figure 6: Pressure Loss Test Results

Preliminary testing of the entire design also took place, however these results are observational, not quantitative. The program, hardware, and chamber interface well. Currently, the device is functional, but not optimal. Input from the computer controls the pressure and temperature in the chamber. However, without impedance into the air pump, low pressures are bypassed because of a high flow rate from the pump. On the other hand, placing resistors in front of the pump compromises the power supplied to the pump. As a result, the pump is able to reach 200 mmHg starting from atmospheric

pressure, but is not able to restart when pressure falls beneath the desired value. It cannot restart because it is unable to overcome the load of the already high pressure combined with the added resistance. This problem is encountered at pressures above 60 mmHg.

Future Development

As previously stated, the current prototype is functional and could be used in a clinical setting, but is not yet optimal. To improve the design we need to optimize the air pump output so that all pressures are attainable and maintainable. Along the same line, we would like to incorporate an over-pressure valve that will automatically release air if the pressure exceeds the desired target. Adding this would help resolve the overshoot of low pressure targets that occurs when there is no resistance placed in front of the air pump. Also, the current pressure sensor is not registering pressures under 30 mmHg, although the transducer's specifications list a low pressure value of -30 mmHg. Either a relationship needs to be found to express this or a new transducer must be used. Due to a change in transducers toward the end of the semester, our current adaptation piece needs to be modified because it is currently made from a material that is not resistant to corrosion. Additionally, much more design testing needs to be done. This includes rigorous code validation, heating time tests, and full model testing.

Conclusion

A system was developed which can mimic *in vivo* cardiovascular physiological conditions through computer-monitored pressure and temperature control. The simulated environment created by this system may be used to test and calibrate pressure sensitive cardiovascular catheters. The fabricated calibration system takes input from a user, checks and verifies pressures inside of the chamber, and turns the air pump on if necessary. This is all done through a computer program. The thermal environment of the calibration system is held constant through the use of a thermistor, a heating pad, and the aforementioned software. To keep light from affecting pressure readings and to control the thermal gradient more effectively, a polystyrene outer-chamber was created to enclose the acrylic chamber during the calibration procedure. In this way, the current prototype has addressed the requirements of constant pressure, constant temperature, and light shielding that are necessary for pressure sensitive catheter calibration. While the calibration system still needs further development, the final product will be beneficial not only because of its application to catheters, but also because the device could have implications for a wide range of biomedical and physiological tests which require a simulated vascular environment.

```
'Under Pressure Main Menu
'BME 402 Spring 2007

'Written by     Marty Grasse mmgrasse@wisc.edu
'               Anita Zarebi alzarebi@wisc.edu



Public Class MainMenu

    Const PortNum As MccDaq.DigitalPortType = MccDaq.DigitalPortType.FirstPortA ' set port to use

    Const Direction As MccDaq.DigitalPortDirection = MccDaq.DigitalPortDirection.DigitalOut ' program digital port A for output

    Private DaqBoard As MccDaq.MccBoard = New MccDaq.MccBoard(0) 'Declare value of board

    'Declarations
    Dim CurrentPressureVolts As Single
    Dim CurrentPressuremmHg As Single
    Dim CurrentPressureRaw As UShort

    Dim CurrentTemperatureRaw As UShort
    Dim CurrentTemperatureVolts As Single
    Dim CurrentTemperatureDegrees As Single

    Dim OutputRun As Boolean

    Dim ULStat As MccDaq.ErrorInfo
    Dim DesiredPressure As Single
    Dim DesiredTemperature As Single
    Dim PressureSlope As Single
    Dim PressureIntercept As Single
    Dim TempSlope As Single
    Dim TempIntercept As Single
    Dim TempBitValue As MccDaq.DigitalLogicState
    Dim PressBitValue As MccDaq.DigitalLogicState
    Dim BitNumPressure As Short
    Dim BitNumTemperature As Short
    Dim PortType As MccDaq.DigitalPortType
    Dim Range As MccDaq.Range


    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

        PortType = PortNum

        ULStat = DaqBoard.DConfigPort(PortNum, Direction)
        'Sets the direction of the digital output port


        DesiredPressureBox.Text = 0
        DesiredPressure = Val(DesiredPressureBox.Text)

        Microsoft.VisualBasic.FileSystem.FileOpen(1, "C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects\Main Menu_Marty_4_26\Defaults.txt", OpenMode.Input)
        'Microsoft.VisualBasic.FileSystem.FileOpen(1, "C:\Documents and Settings\Anita Zarebi\My Documents\Visual Studio 2005\Projects\Main Menu_Marty_4_13\Defaults.txt", OpenMode.Input)
        DesiredTemperature = LineInput(1)
        PressureSlope = LineInput(1)
        PressureIntercept = LineInput(1)
        TempSlope = LineInput(1)
        TempIntercept = LineInput(1)
```

```
        FileClose(1)


        OutputRun = False
        'sets the boolean to false --> keeps outputs from turning on


        'DRIVE ALL OUTPUT BITS LOW UPON LOAD

        TempBitValue = MccDaq.DigitalLogicState.Low
        PressBitValue = MccDaq.DigitalLogicState.Low


        ULStat = DaqBoard.DBitOut(PortType, BitNumTemperature, TempBitValue)
        ULStat = DaqBoard.DBitOut(PortType, BitNumPressure, PressBitValue)



    End Sub
    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles DesiredPressureBox.TextChanged

    End Sub
    Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles CurrentTemperatureBox.Click

    End Sub


    Public Sub ConfirmRunButton_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles ConfirmRunButton.Click


        DesiredPressure = Val(DesiredPressureBox.Text)
        Microsoft.VisualBasic.FileSystem.FileOpen(1, "C:\Documents and Settings\
    Administrator\My Documents\Visual Studio 2005\Projects\Main Menu_Marty_4_26\Defaults.
    txt", OpenMode.Input)
        'Microsoft.VisualBasic.FileSystem.FileOpen(1, "C:\Documents and Settings\Anita
    Zarebi\My Documents\Visual Studio 2005\Projects\Main Menu_Marty_4_13\Defaults.txt",
    OpenMode.Input)
        DesiredTemperature = LineInput(1)
        PressureSlope = LineInput(1)
        PressureIntercept = LineInput(1)
        TempSlope = LineInput(1)
        TempIntercept = LineInput(1)
        FileClose(1)

        OutputRun = True


    End Sub

    'All loops in timer so that will update every half second and continually run loops
    'Note that the interval is set as a parameter of the timer (see form page); half a
    second
    'corresponds to 500 ms


    Public Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Timer1.Tick

        'PortType = PortNum
        'set this when load the form


        BitNumPressure = CShort(0)
        BitNumTemperature = CShort(1)
```

```
        'assign output bit numbers within the port to 0 and 1 for pressure and temperature
    , respectively

        Range = Range.Bip10Volts



        'Pressure Commands
        ULStat = DaqBoard.AIn(0, Range, CurrentPressureRaw) 'calling AIn for Pressure
        ULStat = DaqBoard.ToEngUnits(Range, CurrentPressureRaw, CurrentPressureVolts)
    'convert raw reading from DAQ to a voltage value

        'Parameters: BoardNumber of DAQ, Channel # of DAQ,
        'BIP10Volts = range -10 to 10 V, name for function's output
        'AIn will read in the voltage from that channel and output it as
        'a number called CurrentPressureVolts

        If ULStat.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then
            Stop
            'add code to pop up text box with error message
        End If


        CurrentPressuremmHg = PressureSlope * CurrentPressureVolts + PressureIntercept

        'CurrentPressuremmHg = 67.132 * CurrentPressureVolts + 558.63

        'This function converts CurrentPressureVolts to CurrentPressure in mmHg

        CurrentPressureBox.Text = CurrentPressuremmHg  'for real
        'CurrentPressureBox.Text = CurrentPressureVolts  'for testing
        'Note: This will be refreshed every half second according to timer parameters

        'TEMPERATURE INPUTS

        ULStat = DaqBoard.AIn(1, Range, CurrentTemperatureRaw) 'calling AIn for
    Temperature

        ULStat = DaqBoard.ToEngUnits(Range, CurrentTemperatureRaw,
    CurrentTemperatureVolts)

        If ULStat.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then
            Stop
            'add code to pop up text box with error message
        End If


        'CurrentTemperatureDegrees = 92.509 * CurrentTemperatureVolts + 144.52
        CurrentTemperatureDegrees = TempSlope * CurrentTemperatureVolts + TempIntercept

        CurrentTemperatureBox.Text = CurrentTemperatureDegrees  'use for real program
        'CurrentTemperatureBox.Text = CurrentTemperatureVolts    'use for troubleshooting


        'TEMPERATURE OUTPUTS

        'This will allow the user to toggle the temperature outputs on and off when not in
     a calibration run

        If TempControlToggle.Checked = False And OutputRun = False Then

            TempBitValue = MccDaq.DigitalLogicState.Low


            ULStat = DaqBoard.DBitOut(PortType, BitNumTemperature, TempBitValue)

        Else
```

```
            If CurrentTemperatureDegrees < DesiredTemperature Then
                TempBitValue = MccDaq.DigitalLogicState.High

            Else
                TempBitValue = MccDaq.DigitalLogicState.Low

            End If

            ULStat = DaqBoard.DBitOut(PortType, BitNumTemperature, TempBitValue)

        End If

        'Error handling for temperature outputs
        If ULStat.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then
            MsgBox(ULStat.Value)

            Stop
        End If


        If OutputRun = True Then
            'This code will only allow the digital outputs to be functional if the confirm↙
     and run button is depressed

            'Pressure Outputs
            If CurrentPressuremmHg < DesiredPressure Then
                PressBitValue = MccDaq.DigitalLogicState.High
                ' Parameters: BoardNumber of DAQ, Port Type (digital port programmable as ↙
    output or input
                ' 0 is the first pin in port A (or channel 1), BitValue (0 for off, 1 for ↙
    on)


            Else
                PressBitValue = MccDaq.DigitalLogicState.Low
            End If



            ULStat = DaqBoard.DBitOut(PortType, BitNumPressure, PressBitValue)




            If ULStat.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then
                'MsgBox(ULStat.Value)
                'code to break out of timer and display message

                Stop
            End If
        End If

        If OutputRun = False Then

            PressBitValue = MccDaq.DigitalLogicState.Low



            ULStat = DaqBoard.DBitOut(PortType, BitNumPressure, PressBitValue)

        End If

        If TempBitValue = MccDaq.DigitalLogicState.High Then
            HeaterStatusBox.Text = "The heating pad is ON."
        End If

        If TempBitValue = MccDaq.DigitalLogicState.Low Then
```

```
            HeaterStatusBox.Text = "The heating pad is OFF."
        End If

        If PressBitValue = MccDaq.DigitalLogicState.High Then
            AirPumpStatusBox.Text = "The air pump is ON."
        End If

        If PressBitValue = MccDaq.DigitalLogicState.Low Then
            AirPumpStatusBox.Text = "The air pump is OFF."
        End If

    End Sub

    Private Sub CurrentPressureBox_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles CurrentPressureBox.Click

    End Sub

    Private Sub InstallationSettingsButton_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles InstallationSettingsButton.Click
        If OutputRun = True Then
            InstallationSettingsButton.Enabled = False
            'can't change installation settings during calibration run
        Else
            InstallationSettingsButton.Enabled = True

            InstallSettings.Show() 'When clicked, this will show the Install Settings form

        End If

    End Sub

    Private Sub CurrentPressureLabel_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles CurrentPressureLabel.Click

    End Sub

    Private Sub CurrentTempLabel_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles CurrentTempLabel.Click

    End Sub



    Private Sub Label1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
     Handles HeaterStatusBox.Click


    End Sub

    Private Sub StopButton_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles StopButton.Click


        OutputRun = False
        'setting this to false will bring pressure output low to turn off both the air
    pump and the heating pad


        InstallationSettingsButton.Enabled = True  're-enable the Install Settings button
    when calibration is stopped


    End Sub

    Private Sub TempBoxControlToggle_CheckedChanged(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles TempControlToggle.CheckedChanged
```

```
    End Sub

    Private Sub HelpButton_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles HelpButton.Click
        Help.Show()

    End Sub

    Private Sub AboutButton_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles AboutButton.Click
        AboutBox.Show()

    End Sub


End Class
```

```
'Under Pressure Installation Settings Menu
'BME 402 Spring 2007

'Written by     Marty Grasse mmgrasse@wisc.edu
'               Anita Zarebi alzarebi@wisc.edu



Imports System.Windows.Forms

Public Class InstallSettings


    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles OK_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        FileOpen(1, "C:\Documents and Settings\Administrator\My Documents\Visual Studio
    2005\Projects\Main Menu_Marty_4_26\Defaults.txt", OpenMode.Output)
        'FileOpen(1, "C:\Documents and Settings\Anita Zarebi\My Documents\Visual Studio
    2005\Projects\Main Menu_Marty_4_13\Defaults.txt", OpenMode.Output)
        'save changes to file if button press = ok
        'Writes all current values in the boxes to the defaults file regardless of whether
     or not they are different
        'from old defaults
        PrintLine(1, DesiredTemperatureBox.Text)
        PrintLine(1, PressureSlopeBox.Text)
        PrintLine(1, PressInterceptBox.Text)
        PrintLine(1, TempSlopeBox.Text)
        PrintLine(1, TempInterceptBox.Text)
        FileClose(1)

        Me.Close() 'closes window
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles Cancel_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        'discard changes and keep old settings
        Me.Close()
    End Sub


    Private Sub InstallSettings_Load(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
        FileOpen(1, "C:\Documents and Settings\Administrator\My Documents\Visual Studio
    2005\Projects\Main Menu_Marty_4_26\Defaults.txt", OpenMode.Input)
        'FileOpen(1, "C:\Documents and Settings\Anita Zarebi\My Documents\Visual Studio
    2005\Projects\Main Menu_Marty_4_13\Defaults.txt", OpenMode.Input)
        DesiredTemperatureBox.Text = LineInput(1)
        PressureSlopeBox.Text = LineInput(1)
        PressInterceptBox.Text = LineInput(1)
        TempSlopeBox.Text = LineInput(1)
        TempInterceptBox.Text = LineInput(1)
        FileClose(1)

    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles PressureSlopeBox.TextChanged

    End Sub

    Private Sub TextBox3_TextChanged(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles DesiredTemperatureBox.TextChanged

    End Sub
```

```
    Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
    Handles PressureInterceptLabel.Click

    End Sub

    Private Sub PressInterceptBox_TextChanged(ByVal sender As System.Object, ByVal e As   ↙
    System.EventArgs) Handles PressInterceptBox.TextChanged

    End Sub
End Class
```