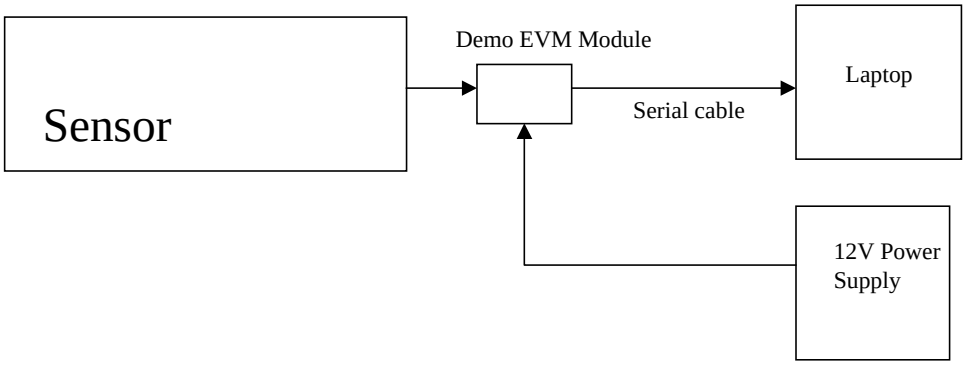
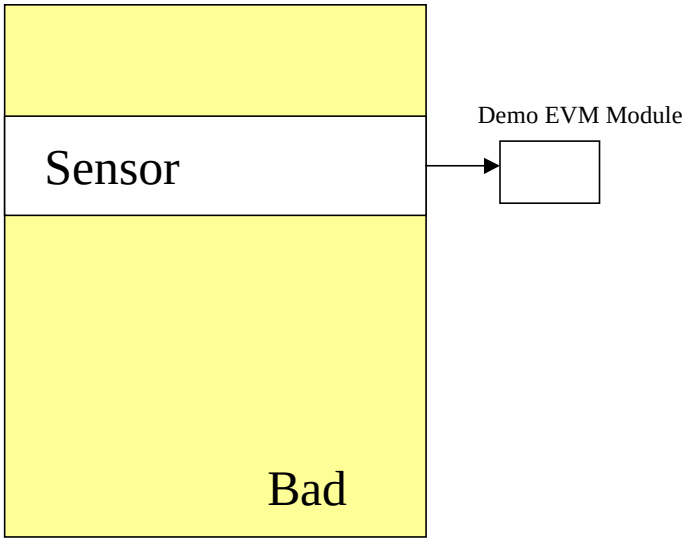
EF3415 – Sleep'O'Meter

The purpose of this application is to track the movements of the subject while is sleeping. If the subject has a good sleep the it will not move to many times during the night. In the cases when the subject is disturbed by external factors like noise/light/vibrations frequent moves will occur. Also in the case when the subject has some health problems frequent moves will occur.
By tracking the moves and the presence of the subject in bad other tasks like counting the number of sleep hours can be performed.

The project uses a sensor that is placed in bed, on the mattress and the EVM demo module KIT33794DWBEVM based on MC33794 and 908QY4 CPU. The main application is reading the data from the demo module thru the serial port and is displaying it on the screen. The data can be recorded and saved and a Plot showing the active sensor for each minute can be created using the collected data.

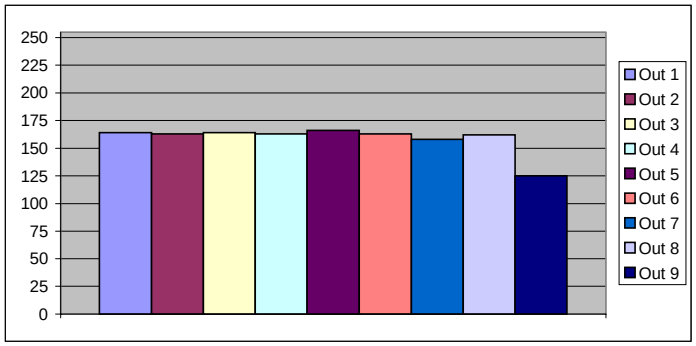




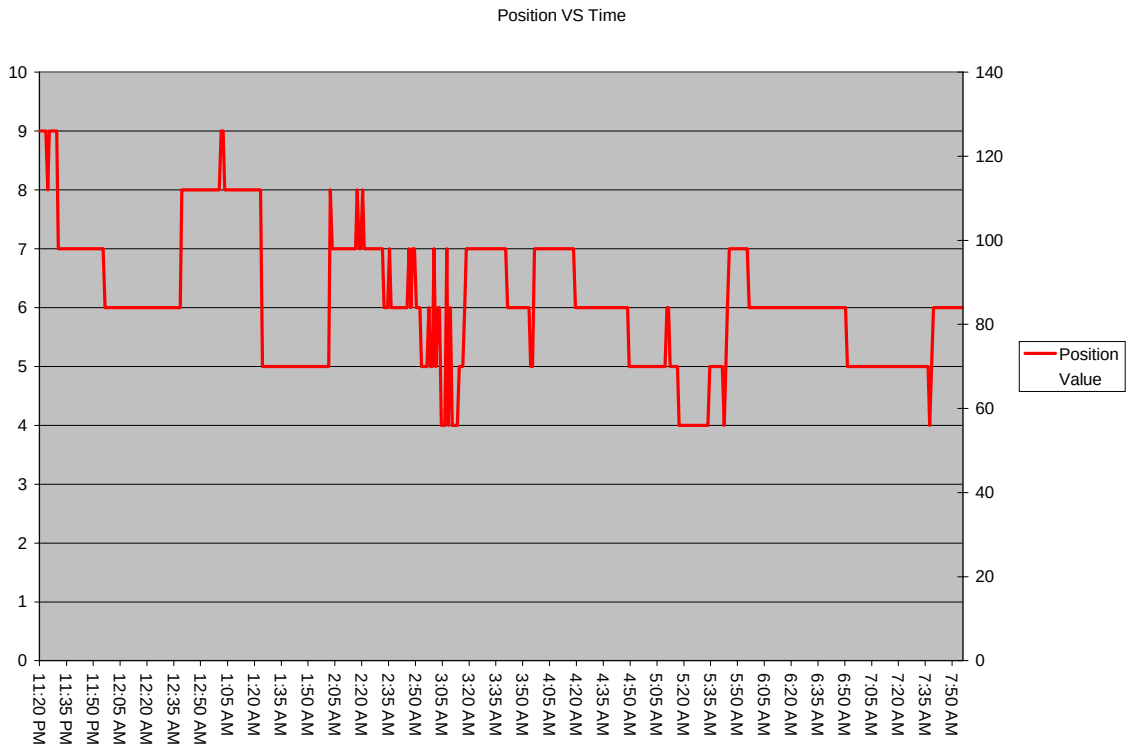
The setup is quite simple – place the sensor on the mattress, connect the sensor, the power and the serial cable to the EVM module. Connect the serial cable to the computer and start the EVM_log Excel program.
Press the start Button and the readings from the sensors will show up on the screen. This screen can be used for the initial debug of the sensor – place your hand in different regions of the sensor and check the result on the screen.
Once the recording is done press the Compute Average Button in order to get the results – the Average sheet will be populated with the average readings for each minute and the Position versus Time Chart will be created.
The Chart will show the position of the active sensor over time – in this case sensor no. 9 was placed at the edge of the bed and sensor no. 1 was at the opposite end.
As you can see from this chart something disturbed the subject about 2.35AM – in this case some noisy neighbors coming back from a party.
By analyzing the Charts obtained for different nights a clear connection can be made between the number of movements in bed, the disturbing factors and the quality of the sleep.

Out 1	Out 2	Out 3	Out 4	Out 5	Out 6	Out 7	Out 8	Out 9		Log	Log Position	Log at 1 Second
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	On/Off	TRUE	30975	TRUE
164	163	164	163	166	163	158	162	125	Value			

Compute Average

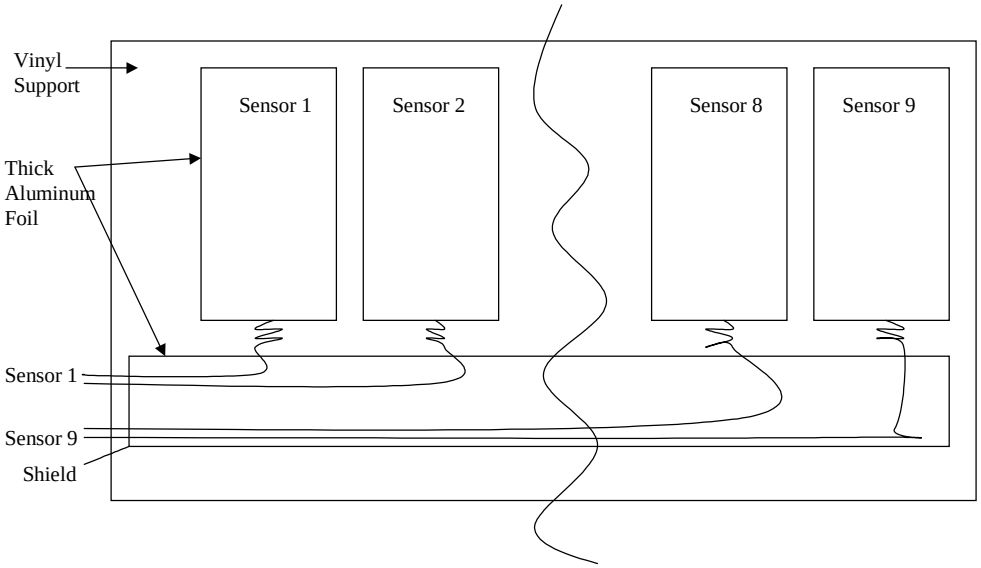


Start

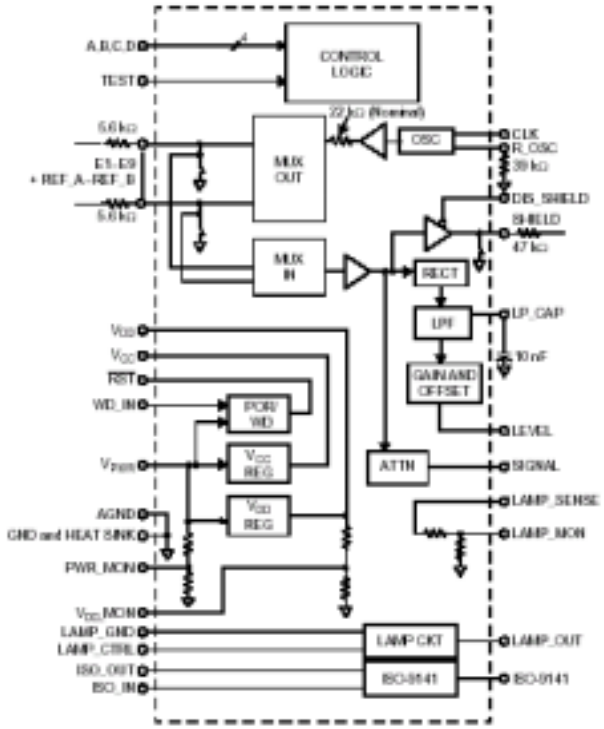

Position VS Time
10
9
8
7
6
5
4
3
2
1
0
140
120
100
80
60
40
20
0
11:20 PM
11:35 PM
11:50 PM
12:05 AM
12:20 AM
12:35 AM
12:50 AM
1:05 AM
1:20 AM
1:35 AM
1:50 AM
2:05 AM
2:20 AM
2:35 AM
2:50 AM
3:05 AM
3:20 AM
3:35 AM
3:50 AM
4:05 AM
4:20 AM
4:35 AM
4:50 AM
5:05 AM
5:20 AM
5:35 AM
5:50 AM
6:05 AM
6:20 AM
6:35 AM
6:50 AM
7:05 AM
7:20 AM
7:35 AM
7:50 AM
Position Value

Hardware

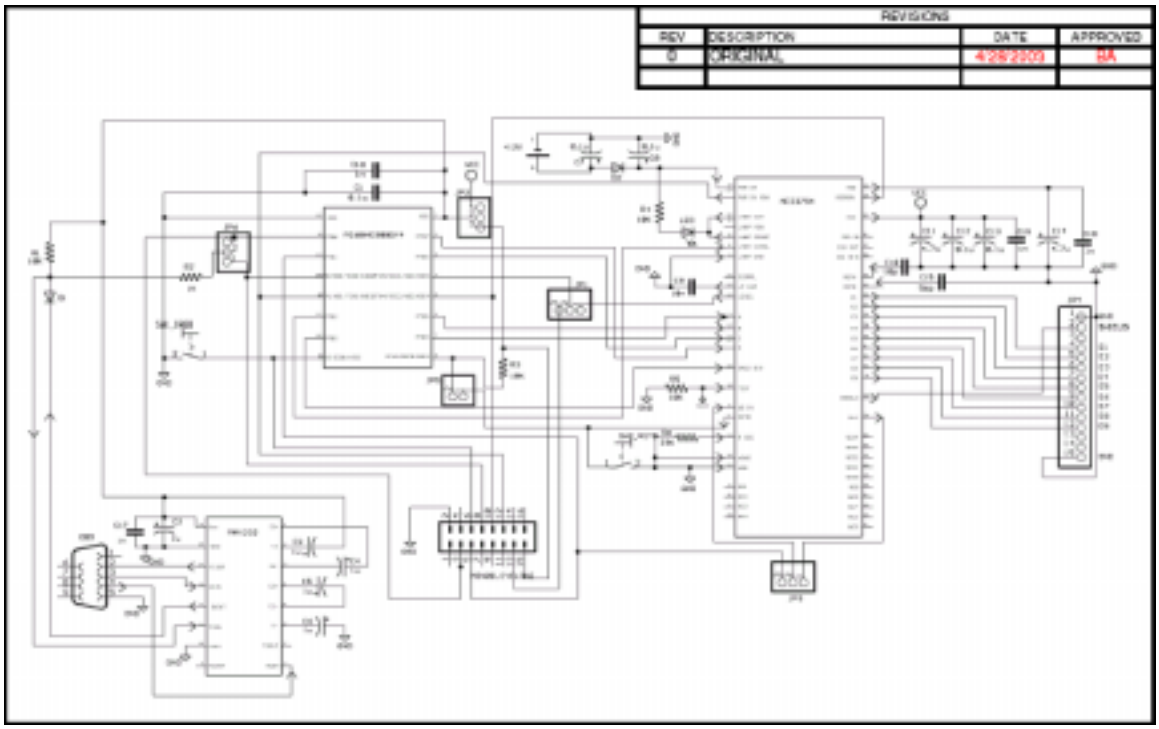
The hardware consist in a Sensor built on a piece of Vinyl/Textile material and the EVM Module. Vinyl/Textile material is flexible and is available for sale at Wal-Mart Stores.
The sensor elements are built from thick aluminium foil.
Because the size of the sensor is the size of the bed shielding was a must.
Each sensor is connected to a thin copper wire using a piece of cardboard and a couple of stapples. In order to release the stress the wires are bended in an S-shape and then are routed to the shield area. The shield area is another piece of thick aluminium foil connected to the shield output of the EVM module. Everything is fixed with tape.
So the whole sensor has 10 connections – sensor1…sensor 9 and shield.
All the connections are routed to a flat connector towards the EVM – like that the Sensor-EVM assembly caan be disconnected/connected very easy.



The EVM module is built around the MC 33794 Electric Field Imaging Device.
The MC 33794 allows the sensing of objects in an electric field it generates by driving a voltage through an internal resistance to electrically conductive electrodes and measures the voltage at the electrode pin. An object moving into the field with a dielectric constant different than the medium it is in will change the capacitance between the electrodes creating the field. The amount of capacitance between the electrodes and conductive elements connected to the circuit ground return will effect the measured voltage because the voltage drop across the resistor is proportional to the current through it, and the current through it is proportional to the capacitance.



The schematic diagram of the EVM module is a tipycal application for the MC 33794 IC. The Sensor IC is connected to the 908QY4 CPU – the Sensor in IC is reading the Sensor 1 to Sensor 9 inputs and is delivering the analog signal to the CPU. The CPU is measuring the signal using the integrated 8 bit AD and is delivering the signal level to the Serial Port.



Software

The data collection software is the main part of this project.
Excel was easy to use and is offering support for data storage/charts and VBA modules.
The LogEVM.xls file contain 4 worksheets and a VBA module.
The Results worksheet present the collected data in real time for immediate examination.
Also three Buttons are present:

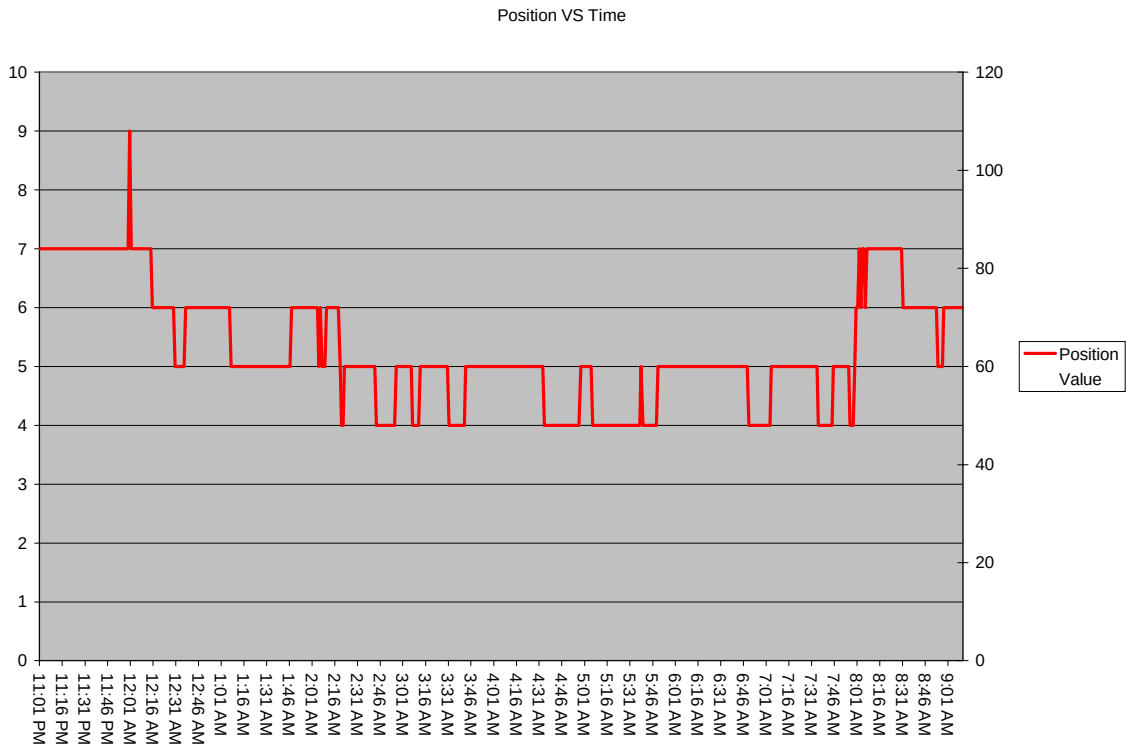
- Start Button to start data collection
- Compute Average Button to compute the average readings (at one minute) and display the Position vs. Time Chart
- Delete Log! Button used to delete All the collected data. This Button is hidden in the right side of the worksheet.

The Log worksheet contains the time in seconds from midnight, the time in AM/PM format and 9 columns for the readings from the sensors. A line is added to this table for each second when recording is on.

83999.25	11:19:59 PM	42	163	165	166	165	164	158	162	49
84000.3	11:20:00 PM	165	165	165	167	165	163	159	161	46
84001.29	11:20:01 PM	166	163	165	166	166	163	159	160	47
84002.28	11:20:02 PM	165	164	166	166	167	163	159	161	40
84004.64	11:20:04 PM	166	164	166	166	0	163	157	161	45
84005.3	11:20:05 PM	166	163	166	166	166	164	157	161	46
84005.9	11:20:05 PM	165	163	165	167	165	164	158	161	45
84006.56	11:20:06 PM	165	164	166	166	165	164	159	162	40
84007.27	11:20:07 PM	166	163	165	166	165	163	158	160	47
84008.26	11:20:08 PM	166	164	166	166	166	163	159	161	39
84009.31	11:20:09 PM	166	164	164	167	165	164	157	160	40
84010.29	11:20:10 PM	165	164	166	165	165	163	158	161	40
84011.28	11:20:11 PM	165	163	166	166	167	163	158	160	49
84012.27	11:20:12 PM	166	163	166	167	166	163	158	163	161
84013.26	11:20:13 PM	166	164	165	167	166	163	159	163	163
84014.3	11:20:14 PM	165	165	166	165	165	164	158	165	161

The Average worksheet contains the time in seconds from midnight, the time in AM/PM format, 9 columns with the average readings from the sensors, the number of seconds for the average period and minimum value + minimum value sensor number. This table is generated from the Log table by averaging the sensor reading values for each minute. The Average1() Subroutine create this table and the Position vs. Time Chart. The chart uses the Time and minimum value sensor number columns to create the final result for the collected data.

84058.3	11:20:58 PM	166.45	165.9	169.05	170.25	166.2333	166.75	128.8833	155.6333	104.6667	60	104.6667	9
84118.22	11:21:58 PM	171.2667	168.4	171.7167	173.4833	171.4167	168.6167	126.2667	109.0833	87.58333	60	87.58333	9
84178.26	11:22:58 PM	171.2167	168.2167	171.4167	173.2667	171.3	168.6833	119.5667	106.95	88.3	60	88.3	9
84239.22	11:23:59 PM	170.9344	167.9344	171.2787	173.0328	170.9344	168.1967	140.4262	102.4918	100.1475	61	100.1475	9
84298.27	11:24:58 PM	170.5932	167.9661	170.9831	173.0508	170.9153	167.9322	144.5593	99.16949	100.1525	59	99.16949	8
84358.3	11:25:58 PM	170.7667	167.7333	170.95	172.7667	170.8667	167.8333	143.85	98.5	97.98333	60	97.98333	9
84418.28	11:26:58 PM	170.4667	167.1833	170.95	172.55	170.65	167.6333	143.7	98.81667	97.01667	60	97.01667	9
84478.26	11:27:58 PM	170.3833	167.3167	170.6667	172.7667	170.3	167.7	143.2333	98.28333	96.95	60	96.95	9
84538.29	11:28:58 PM	170.2667	167.3833	170.5167	172.5167	170.5167	167.6833	143.05	97.53333	96.4	60	96.4	9
84598.27	11:29:58 PM	170.1333	167.3	170.55	172.3	170.5833	167.6167	138.15	98.63333	96.08333	60	96.08333	9
84658.31	11:30:58 PM	168.7667	166.5667	169.35	169.6833	124.2833	103.2167	59.63333	145.9833	131.4	60	59.63333	7



The VBA Module contains subroutines for Data Collection, a Form that is presenting the data in real time and a subroutine to compute the average values of the data for each minute and display the chart.
Serial port COM1:9600/N/8/1 is used for communication with the EVM module.
The MSComm object is used for serial communication.
The StartSerial() subroutine open the serial connection and initialize the MSComm object.

```
Set MSCommXX = New MSComm
With MSCommXX
   .CommPort = 1
   .Handshaking = 0
   .RThreshold = 1
   .RTSEnable = True
   .Settings = "9600,n,8,1"
   .SThreshold = 1
   .PortOpen = True
   .InputLen = 0   'every charcter in the buffer is returned
```

The protocol used By the EVM over the serial line is:

```
>Driver 1       ‘set shield driver on
>S 1            ‘ select electrode nr. 1
>X              ‘ Read the value for electrode nr. 1
>122            ‘ EVM return the value for electrode nr. 1
>S 2            ‘ Select electrode nr. 2
```

>X
>126
….
>S 9
>X
>144 ‘EVM return the value for electrode nr. 9

The subroutine ReadElectrode(electrode As Byte) is reading the value for each electrode. The routine interrogates the EVM and if the answer from the module is not coming in 1 second a timeout is generated.
This subroutine uses the ParseResult sub to parse the string returned by EVM for each electrode reading.

```
Private Function ParseResult(text As String) As Byte
' parse a 8 char text string returned by EVM serial interface
' return 0 in case of error
' X 0D 0A hexdig hexdig 0D 0A >
If Len(text) < 7 Then
   ParseResult = 0
   Exit Function
End If
If Mid(text, 1, 1) <> "X" Then
   ParseResult = 0
   Exit Function
End If
If Mid(text, 2, 1) <> Chr$(13) Then
   ParseResult = 0
   Exit Function
End If
If Mid(text, 3, 1) <> Chr$(10) Then
   ParseResult = 0
   Exit Function
End If
ParseResult = Val("&H" & Mid(text, 4, 2))   ' return value of 2 digit hex reading
End Function
```

The Main Loop of the program uses the ReadElectrode sub to read all the electrodes sequentially. The results are stored in the Log worksheet if Log is true. Each electrode is active if OUT1..OUT9 are TRUE. The starttime variable is used to generate an accurate 1 second interval for the readings (if Log at one Second is True). The values obtained are stored in the Log worksheet.

```
   Do While keepongoin
      DoEvents
      If serialon Then
         ' log at 1 second interval
         If firsttime Then
            starttime = Timer + 1
            firsttime = False
         Else
            starttime = starttime + 1
         End If

         If Worksheets("Results").Cells(2, 1).value Then ReadElectrode (1)
         If Worksheets("Results").Cells(2, 2).value Then ReadElectrode (2)
         If Worksheets("Results").Cells(2, 3).value Then ReadElectrode (3)
         If Worksheets("Results").Cells(2, 4).value Then ReadElectrode (4)
```

```
        If Worksheets("Results").Cells(2, 5).value Then ReadElectrode (5)
        If Worksheets("Results").Cells(2, 6).value Then ReadElectrode (6)
        If Worksheets("Results").Cells(2, 7).value Then ReadElectrode (7)
        If Worksheets("Results").Cells(2, 8).value Then ReadElectrode (8)
        If Worksheets("Results").Cells(2, 9).value Then ReadElectrode (9)

        ' Log the values if logging is active
        If Worksheets("Results").Cells(2, 11).value Then
           logpos = Worksheets("Results").Cells(2, 12).value
           ' seconds since midnight
           Worksheets("Log").Cells(logpos, 1).value = Timer
           ' system time
           Worksheets("Log").Cells(logpos, 2).value = Time
           ' values 1-9
           Worksheets("Log").Cells(logpos, 3).value = Worksheets("Results").Cells(3, 1).value
           Worksheets("Log").Cells(logpos, 4).value = Worksheets("Results").Cells(3, 2).value
           Worksheets("Log").Cells(logpos, 5).value = Worksheets("Results").Cells(3, 3).value
           Worksheets("Log").Cells(logpos, 6).value = Worksheets("Results").Cells(3, 4).value
           Worksheets("Log").Cells(logpos, 7).value = Worksheets("Results").Cells(3, 5).value
           Worksheets("Log").Cells(logpos, 8).value = Worksheets("Results").Cells(3, 6).value
           Worksheets("Log").Cells(logpos, 9).value = Worksheets("Results").Cells(3, 7).value
           Worksheets("Log").Cells(logpos, 10).value = Worksheets("Results").Cells(3, 8).value
           Worksheets("Log").Cells(logpos, 11).value = Worksheets("Results").Cells(3, 9).value
           ' increment log position
           If logpos >= 65535 Then logpos = 65535 ' max nr.of rows in excell
           Worksheets("Results").Cells(2, 12).value = logpos + 1
        End If

        ' wait until next second
        If Worksheets("Results").Cells(2, 13).value Then
           Do While starttime >= Timer
           DoEvents
           If Timer + 3600 < starttime Then starttime = starttime - 86400 ' skip midnight
           Loop
        End If

     End If
  Loop
```

The code is easy to understand and contain some comments, a part that is more difficult is related to the “skip Midnight” feature - the timer function used returns the number of seconds from midnight. The number will increase from 0 to 86399 and the will reset. Some conditional statements will take care of that – so the program will work correct even over midnight.

Conclusion

The Sleep’O’Meter is a fun project, but it can be a useful tool for examining the sleep behavior of individuals in different conditions.

Resources

MC33794 reference sheet –www.motorola.com
KIT33794DWBEVM.PDF – kit documentation
MS VBA Help

VBA Module Listing

```
' Under Tools/References Set: Microsoft Comm Control 6.0
Public MSCommXX As Variant
Public keepongoin As Boolean    'TRUE as long as the program is running
Public serialon As Boolean      'TRUE as long as a serial port is open and assigned to
MSCommXX
Public serialform As Boolean    'TRUE as long as SerialMonitor Form is active
Public Waittime As Single
Public sensorread As Byte


Sub MAIN()
Dim intext As String
keepongoin = True
serialon = False
serialform = False


SerialFormOpen
Waittime = Timer + 1
If Waittime > 86400 Then Waittime = Waittime - 86400 ' skip midnight

'test for parseresult a = ParseResult("X" & Chr$(13) & Chr$(10) & "FF67")
firsttime = True

   Do While keepongoin
      DoEvents
      If serialon Then
         ' log at 1 second interval
         If firsttime Then
            starttime = Timer + 1
            firsttime = False
         Else
            starttime = starttime + 1
         End If

         If Worksheets("Results").Cells(2, 1).value Then ReadElectrode (1)
         If Worksheets("Results").Cells(2, 2).value Then ReadElectrode (2)
         If Worksheets("Results").Cells(2, 3).value Then ReadElectrode (3)
         If Worksheets("Results").Cells(2, 4).value Then ReadElectrode (4)
         If Worksheets("Results").Cells(2, 5).value Then ReadElectrode (5)
         If Worksheets("Results").Cells(2, 6).value Then ReadElectrode (6)
         If Worksheets("Results").Cells(2, 7).value Then ReadElectrode (7)
         If Worksheets("Results").Cells(2, 8).value Then ReadElectrode (8)
```

```
        If Worksheets("Results").Cells(2, 9).value Then ReadElectrode (9)

        ' Log the values if logging is active
        If Worksheets("Results").Cells(2, 11).value Then
           logpos = Worksheets("Results").Cells(2, 12).value
           ' seconds since midnight
           Worksheets("Log").Cells(logpos, 1).value = Timer
           ' system time
           Worksheets("Log").Cells(logpos, 2).value = Time
           ' values 1-9
           Worksheets("Log").Cells(logpos, 3).value = Worksheets("Results").Cells(3,
1).value
           Worksheets("Log").Cells(logpos, 4).value = Worksheets("Results").Cells(3,
2).value
           Worksheets("Log").Cells(logpos, 5).value = Worksheets("Results").Cells(3,
3).value
           Worksheets("Log").Cells(logpos, 6).value = Worksheets("Results").Cells(3,
4).value
           Worksheets("Log").Cells(logpos, 7).value = Worksheets("Results").Cells(3,
5).value
           Worksheets("Log").Cells(logpos, 8).value = Worksheets("Results").Cells(3,
6).value
           Worksheets("Log").Cells(logpos, 9).value = Worksheets("Results").Cells(3,
7).value
           Worksheets("Log").Cells(logpos, 10).value = Worksheets("Results").Cells(3,
8).value
           Worksheets("Log").Cells(logpos, 11).value = Worksheets("Results").Cells(3,
9).value
           ' increment log position
           If logpos >= 65535 Then logpos = 65535 ' max nr.of rows in excell
           Worksheets("Results").Cells(2, 12).value = logpos + 1
        End If

        ' wait until next second
        If Worksheets("Results").Cells(2, 13).value Then
           Do While starttime >= Timer
           DoEvents
           If Timer + 3600 < starttime Then starttime = starttime - 86400 ' skip midnight
           Loop
        End If

     End If
  Loop

If serialon Then MSCommXX.PortOpen = False
Set MSCommXX = Nothing  ' release serialObject before exit
```

```
End Sub
Private Sub ReadElectrode(electrode As Byte)
Dim intext As String


        MSCommXX.InBufferCount = 0  ' reset input buffer
        OutText ("S " & Trim(Str(electrode)) & Chr$(13))
        ' S 1, S 2 select active electrode
        TypeText (Chr$(13) & "Elecrode " & Str(electrode) & "=")
        Waittime = Timer + 1        ' timeout to read result
        Do While Timer < Waittime
           DoEvents
           If MSCommXX.InBufferCount Then
              intext = MSCommXX.Input
              'MSCommXX.InBufferCount = 0  ' reset in buffer
              'TypeText (intext & Chr$(13))
              incount = Len(intext)
              For a = 1 To incount
                 'TypeText (Mid(intext, a, 1))
                 If Mid(intext, a, 1) = ">" Then Exit Do
              Next a
           End If
           If Timer + 3600 < Waittime Then Waittime = Waittime - 86400 ' skip midnight
        Loop


        MSCommXX.InBufferCount = 0  ' reset input buffer
        OutText ("X" & Chr$(13))    ' request a new reading
        Waittime = Timer + 1        ' timeout to read result
                           ' loop to wait the next reading
        Do While Timer < Waittime
           DoEvents
           If MSCommXX.InBufferCount >= 8 Then
              intext = MSCommXX.Input
              MSCommXX.InBufferCount = 0  ' reset in buffer

              'TypeText (intext & Chr$(13))
              'incount = Len(intext)
              'For a = 1 To incount
              '    TypeText (Mid(intext, a, 1) & "  " & Hex(Asc(Mid(intext, a, 1))) &
Chr$(13))
              'Next a
              result = ParseResult(intext)

              If result <> 0 Then
                 TypeText (Str(result))
```

```
            'Worksheets("Results").Cells(3, electrode).Value = result
          Else
            'TypeText (intext & Chr$(13))
            TypeText (Chr$(13))
            incount = Len(intext)
            For a = 1 To incount
            TypeText ("  " & Hex(Asc(Mid(intext, a, 1))))
            Next a
            TypeText (Chr$(13))
          End If
          Worksheets("Results").Cells(3, electrode).value = result
          Exit Do     ' go for the next value
        End If
        If Timer + 3600 < Waittime Then Waittime = Waittime - 86400 ' skip midnight
      Loop
      ' if timeout then result = 0
      If Timer >= Waittime Then Worksheets("Results").Cells(3, electrode).value = 0



End Sub
Sub SerialFormOpen()
SerialMonitor.Show (vbModeless)
'serialform = True
End Sub
Public Sub StartSerial()
   Waittime = Timer + 1
   Set MSCommXX = Nothing
   Set MSCommXX = New MSComm
   With MSCommXX
      .CommPort = 1
      .Handshaking = 0
      .RThreshold = 1
      .RTSEnable = True
      .Settings = "9600,n,8,1"
      .SThreshold = 1
      .PortOpen = True
      .InputLen = 0   'every charcter in the buffer is returned
                  'at any reading
      '.Output = "test 1234567890"
   End With
      SerialMonitor.TextBox1.text = "Port COM1 open, 9600,n,8,1 "
      '& MSCommXX.CommEvent & Chr$(13)
      OutText ("DRIVER 1")
      'OutText ("X")
      serialon = True
```

```
End Sub
Sub OutText(text As String)
   MSCommXX.Output = text

   'SerialMonitor.TextBox1.text = SerialMonitor.TextBox1.text & Chr$(13) + text
   'If serialform Then SerialMonitor.TextBox1.SetFocus

   End Sub
Sub TypeText(text As String)
   SerialMonitor.TextBox1.text = SerialMonitor.TextBox1.text & text
   If Len(SerialMonitor.TextBox1.text) >= 512 Then SerialMonitor.TextBox1.text = text
   'If serialform Then SerialMonitor.TextBox1.SetFocus
End Sub


Private Function ParseResult(text As String) As Byte

' parse a 8 char text string returned by EVM serial interface
' return 0 in case of error
' X 0D 0A hexdig hexdig 0D 0A >

If Len(text) < 7 Then
   ParseResult = 0
   Exit Function
End If

If Mid(text, 1, 1) <> "X" Then
   ParseResult = 0
   Exit Function
End If

If Mid(text, 2, 1) <> Chr$(13) Then
   ParseResult = 0
   Exit Function
End If

If Mid(text, 3, 1) <> Chr$(10) Then
   ParseResult = 0
   Exit Function
End If

ParseResult = Val("&H" & Mid(text, 4, 2))   ' return value of 2 digit hex reading

End Function
```

```
Sub deletelog()
'
' deletelog Macro
' Macro recorded 9/6/2003
'

'
    Sheets("Results").Select
    Range("L2").Select
    ActiveCell.FormulaR1C1 = "1"
    Sheets("Log").Select
    Cells.Select
    Selection.ClearContents
    Range("A1").Select
    Sheets("Results").Select
    Range("L3").Select
End Sub

Sub Average1()
' Average the recorded data to 1 minute interval / establish the position ( minimum
reading )
Dim logpos, avepos As Integer
Dim value(1 To 9) As Integer
Dim nrvalues As Integer
Dim k As Byte
Dim a As Date


' delete old values first
Sheets("Average").Select
    Cells.Select
    Selection.ClearContents
    Range("A1").Select

logpos = 1
avepos = 1
lasttime = Worksheets("Log").Cells(logpos, 1).value ' read initial value
lasttime = lasttime + 60 ' add 1 minute interval
If lasttime > 86400 Then
    lasttime = lasttime - 86400 ' skip midnight
    Do While ((lasttime <= Worksheets("Log").Cells(logpos, 1).value) And
(Worksheets("Log").Cells(logpos, 1).value <> ""))
    logpos = logpos + 1
    Loop
End If
```

```
' get initial values
nrvalues = 0
For k = 1 To 9
   value(k) = 0
Next k
 'look for 1 minute intervals
Do While Worksheets("Log").Cells(logpos, 1).value <> ""
      'DoEvents
      If Worksheets("Log").Cells(logpos, 1).value < lasttime Then
         For k = 1 To 9
            value(k) = value(k) + Worksheets("Log").Cells(logpos, 2 + k).value
         Next k
         nrvalues = nrvalues + 1
         logpos = logpos + 1
      Else
         ' compute the average, store the average in the average table and go to next minute
         ' logpos  - do not change
         Worksheets("Average").Cells(avepos, 1).value = Worksheets("Log").Cells(logpos
- 1, 1).value
         a = Worksheets("Log").Cells(logpos - 1, 2).value
         Worksheets("Average").Cells(avepos, 2).value = a
         ' start with position 1 as reference
         minimum = value(1) / nrvalues
         minpos = 1
         For k = 1 To 9
         ' avoid divide by 0
            If nrvalues <> 0 Then
               Worksheets("Average").Cells(avepos, 2 + k).value = value(k) / nrvalues
               If (value(k) / nrvalues) < minimum Then
               minimum = (value(k) / nrvalues)
               minpos = k
               End If
            Else
               Worksheets("Average").Cells(avepos, 2 + k).value = 0
            End If
         Next k
         Worksheets("Average").Cells(avepos, 12).value = nrvalues
         Worksheets("Average").Cells(avepos, 13).value = minimum
         Worksheets("Average").Cells(avepos, 14).value = minpos

         avepos = avepos + 1

         ' reinit average
         nrvalues = 0
         For k = 1 To 9
            value(k) = 0
```

```
            Next k
            'lasttime = Worksheets("Log").Cells(logpos, 1).value
            lasttime = lasttime + 60 '  next 1 minute interval
            If lasttime > 86400 Then
               lasttime = lasttime - 86400 ' skip midnight
               ' skip values until the next day

               Do While ((lasttime <= Worksheets("Log").Cells(logpos, 1).value) And
(Worksheets("Log").Cells(logpos, 1).value <> ""))
               logpos = logpos + 1
               Loop

            End If
         End If
Loop
         ' rebuild chart
         avepos = avepos - 1
         If avepos <> 0 Then  ' empty table
            Sheets("Position VS Time").Select
            ActiveChart.SeriesCollection(1).Values = "=Average!R1C14:R" &
Trim(Str(avepos)) & "C14"
            ActiveChart.SeriesCollection(2).Values = "=Average!R1C13:R" &
Trim(Str(avepos)) & "C13"
            ActiveChart.SeriesCollection(1).XValues = "=Average!R1C2:R" &
Trim(Str(avepos)) & "C2"
         End If
End Sub
```