Sumario

SUMARIO ____________________________________________________ 1
A. CÓDIGO USB SERIAL PYTHON ________________________________ 3
B. CÓDIGO USB SERIAL KIVY __________________________________ 4
C. CÓDIGO USB SERIAL B4A ________________________________ 6
D. CÓDIGO OSCILÓSCOPIO ___________________________________ 11
A. Código USB Serial Python

```python
import serial
import time

# configure the serial connections (the parameters differs on the device you are connecting to)
ser = serial.Serial(
    port='COM5',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)

print(ser.isOpen())

print('Enter your commands below.\r\nInsert "exit" to leave the application.\n')

ser.write('\n')
ser.write('\n
for i in range(1,21):
    ser.write([0x00])

ser.write('\n')
time.sleep(0.1)
ser.write([0x15])
#ser.write('\n')
time.sleep(0.1)
while 1 :
    if ser.inWaiting() > 0:
        print(hex(ord(ser.read(1))))
        #print(ser.read(1))
```

B. Código USB Serial Kivy

```python
# -*- coding: utf8 -*-

from kivy.app import App
from kivy.uix.floatlayout import FloatLayout
from kivy.graphics import Line
from kivy.uix.label import Label
from kivy.core.window import Window
from kivy.clock import Clock
import serial

# Basic class Float Layout
class Test1(FloatLayout):
    def __init__(self, **kwargs):
        super(Test1, self).__init__(**kwargs)
        # Set the timer for redrawing the screen
        refresh_time = 0.5
        Clock.schedule_interval(self.timer, refresh_time)

        with self.canvas:
            self.centered_circle = Line(circle = (self.center_x, self.center_y, 50), width = 2)

    def timer(self, dt):
        # Get data from serial port
        value = arduino.readline()

        # Draw the circle according to the data
        self.centered_circle.circle = (self.center_x, self.center_y, value)

    # More about drawing in Kivy here: http://kivy.org/docs/api-kivy.graphics.html

# Main App class
class SerialDataApp(App):
    def build(self):
        return Test1()

# Main program
if __name__ == '__main__':
    # Connect to serial port first
    try:
        arduino = serial.Serial('/dev/bus/usb/001/002', 115200)
        print "Connected"
    except:
        print "Failed to connect"
```
exit()

# Launch the app
SerialDataApp().run()

# Close serial communication
arduino.close()
C. Código USB Serial B4A

#Region Project Attributes
    #ApplicationLabel: USB Serial Example
    #VersionCode: 1
    #VersionName:
    #SupportedOrientations: unspecified
    #CanInstallToExternalStorage: false
#End Region

#Region Activity Attributes
    #FullScreen: False
    #IncludeTitle: True
#End Region

Sub Process_Globals
    ' Demo has been changed to support Ver_2.4 JeanLC
    Dim usb1 As UsbSerial
    Dim astreams1 As AsyncStreams
    Dim zbyte(1) As Byte
    Dim data(2) As Byte
    Dim USBValue As Int
    Dim flag As Boolean
    Dim Counter As Int
    Dim Timer1 As Timer
    Dim bc As ByteConverter
End Sub

Sub Globals
    Dim btnSend, btnOpen, btnClose, btnReset As Button
    Private adc As Label
End Sub

Sub Activity_Create(FirstTime As Boolean)
    Activity.LoadLayout("1")
    btnClose.Enabled = False
    btnSend.Enabled = False
    btnReset.Enabled = False
    flag = False
    Counter = 0
    data(0) = 2
    data(1) = 255
    Timer1.Initialize("Timer1", 500) ' 1000 = 1 segundo
End Sub

Sub btnOpen_Click
    If usb1.UsbPresent(1) = usb1.USB_NONE Then ' Ver_2.4
        Log("Msgbox - no device")
        MsgBox("No USB device or accessory detected!", "Error")
        Log("Msgbox - returned")
        Return
    End If
    Log("Checking permission 1").
    If (usb1.HasPermission(1)) Then ' Ver_2.4
        'Msgbox(usb1DeviceInfo(1), "Device Information 1") ' Ver_2.4
        Dim dev As Int
        dev = usb1.Open(115200, 1) ' Ver_2.4
        'dev = usb1.Open(9600, 1) ' Ver_2.4
        If dev <> usb1.USB_NONE Then
            Log("Connected successfully! 1")
            btnOpen.Enabled = False
            btnClose.Enabled = True
            btnSend.Enabled = True
            btnReset.Enabled = True
            astreams1.Initialize(usb1.GetInputStream, usb1.GetOutputStream, "astreams1")
            zbyte (0) = 0x0D
            astreams1.Write(zbyte)
            zbyte (0) = 0x0D
            astreams1.Write(zbyte)
            Delay(1000)
            zbyte (0) = 0x00
            For i = 1 To 20
                astreams1.Write(zbyte)
            Next
            zbyte(0)=0x0D
            'astreams1.Write(zbyte)
            Timer1.Enabled = True
            Delay(1000)
            zbyte(0)=0x14
            astreams1.Write(zbyte)
            flag=True
            'zbyte(0)=0x0D
            'astreams1.Write(zbyte)
        Else
            Log("Error opening USB port 1")
        End If
    Else
        usb1.RequestPermission(1) ' Ver_2.4
    End If
End Sub

Sub btnReset_Click
    zbyte (0) = 0x0D
    astreams1.Write(zbyte)
    Delay(1000)
    zbyte (0) = 0x0F
    astreams1.Write(zbyte)
    astreams1.Close
    usb1.Close
End Sub

Sub Astreams1_NewData (Buffer() As Byte)
    ' You must check for DeviceInfo or analyze Buffer data to know what is connected
to the USB
    ' The order of the USB could change as you plug them and could change when
changing the hub port they are connected to
    If flag=True Then
        data(Counter)=Buffer(0)
        If Buffer.Length > 1 Then
            Counter= 1
            data(Counter)=Buffer(1)
        End If

        Counter= Counter + 1
        If Counter == 2 Then
            USBValue= data(0)
            USBValue= USBValue*256
            USBValue= USBValue + data(1)
            USBValue= (USBValue/1023)*6.6
            Counter=0
        End If
        Log(USBValue)
    End If
End Sub

Sub Timer1_Tick
    zbyte(0)=0x14
    astreams1.Write(zbyte)
End Sub
Sub btnClose_Click
    streams1.Close
    usb1.Close
    btnOpen.Enabled = True
    btnClose.Enabled = False
    btnSend.Enabled = False
End Sub

Sub btnSend_Click
    streams1.Write("abcde".getBytes("UTF8"))
End Sub

Sub btnExit_Click
    ExitApplication
End Sub

Sub AStreams1_Error
    Log("Error: ", LastException)
    streams1.Close
End Sub

Sub AStreams1_Terminated
    Log("Terminated")
    streams1.Close
End Sub

Sub Delay(nMilliSecond As Long)
    Dim nBeginTime As Long
    Dim nEndTime As Long
    nEndTime = DateTime.Now + nMilliSecond
    nBeginTime = DateTime.Now
    Do While nBeginTime < nEndTime
        nBeginTime = DateTime.Now
        'Log(nBeginTime)
        If nEndTIme < nBeginTime Then
            Return
        End If
        DoEvents
    Loop
End Sub
Sub Activity_Resume

End Sub

Sub Activity_Pause (UserClosed As Boolean)

End Sub
D. Código Osciloscopio

#Region Module Attributes
    #FullScreen: False
    #IncludeTitle: True
    #ApplicationLabel: Oscilloscope
    #VersionCode: 1
    #VersionName: 
    #SupportedOrientations: landscape
    #CanInstallToExternalStorage: False
#End Region

'Activity module
Sub Process_Globals
    'These global variables will be declared once when the application starts.
    'These variables can be accessed from all modules.
    Dim ProgName As String     : ProgName = "Oscilloscope"
    Dim ProgVersion As String    : ProgVersion = "V BP"
    Dim Timer1 As Timer
    Type Curves (Name As String, Color As Int, Width As Float, Scale As Double, Offset As Double, Draw As Boolean)
        Dim CurveVal(4, 105) As Double
        Dim ScreenX0, ScreenX1, ScreenY0, ScreenY1, ScreenW, ScreenH, Border As Int
        Dim GridX0, GridX1, GridY0, GridY1, GridYm, GridW, GridH, Div As Int
        Dim NbDivX, NbDivY As Int
    Dim ScreenCol, GridLineCol As Int
    Dim t, dt As Double
    Dim dx, xx, cx As Float
    Dim xxx(105) As Float
    Dim ii As Int
    Dim CurveI As Int        ' curve index
        Dim Curve(4) As Curves
        Dim CurvesI(4) As Int
        Dim CurvesNb As Int : CurvesNb = 0
        Dim CurvesNbDisp As Int
        Dim y1(4) As Float
        Dim y2(4) As Float
        Dim SingleShot As Boolean : SingleShot = False
        Dim Stopped As Boolean     : Stopped = True
    ' Dim ScopeMode As String     : ScopeMode = "MEM"
Dim ScopeMode As String : ScopeMode = "SCOPE"
Dim ScopeMode As String : ScopeMode = "ROLL"
Dim ScopeRolling As Boolean : ScopeRolling = False
Dim w(4) As Double
Dim a(4) As Double
Dim TimeScale(10) As Double
Dim SignalScale(10) As Double
Dim usb1 As UsbSerial
Dim astreams1 As AsyncStreams
Dim zbyte(1) As Byte
Dim data(2) As Byte
Dim USBValue As Float
Dim flag As Boolean
Dim Counter As Int
Dim Timer2 As Timer
Dim bc As ByteConverter

End Sub

Sub Globals
    'These global variables will be redeclared each time the activity is created.
    'These variables can only be accessed from this module.
    Dim btnStart, btnStop, btnSingleShot, btnOpen, btnReset, btnSend, btnExit As Button
    Dim pnlOcilloscope, pnlScreen, pnlGraph, pnlCursor, pnlControl As Panel
    Dim pnlCurveTools, pnlDispValues As Panel
    Dim cvsScreen, cvsGraph, cvsCursor As Canvas
    Dim rectScreen, rectGrid As Rect
    Dim spnTimeScale As Spinner
    Dim spnScale0 As Spinner
    Dim lblScale0 As Label
    Dim lblValue0, lblValue1, lblValue2, lblValue3, lblusb, lblvolt As Label
    Dim lblOffset0, lblOffset1, lblOffset2, lblOffset3 As Label
    Dim edtOffset0, edtOffset1, edtOffset2, edtOffset3 As EditText
    Dim rbtScopeScope, rbtScopeMEM, rbtScopeROLL As RadioButton
    Dim scvControl As ScrollView
    Dim bmpRoll As Bitmap

End Sub

Sub Activity_Create(FirstTime As Boolean)
    Activity.Title = ProgName & " " & ProgVersion
    dt = 0.01
    t = 0
    Timer1.Initialize("Timer1", dt * 1000)
End Sub
'Start USB Serial Initialization
flag = False
Counter = 0
Timer2.Initialize("Timer2", 2)
data(0) = 2
data(1) = 255
USBValue = data(0)
    USBValue = USBValue * 256
    USBValue = USBValue + data(1)

'btnOpen.Enabled = True
'btnSend.Enabled = False
'btnReset.Enabled = False
'btnExit.Enabled = True

'Finish USB Serial Initialization

NbDivX = 10
NbDivY = 8

Div = Floor(80%y / NbDivY)
xx = 0
xxx(0) = 0
'dx = Div / 10
dx = 2
GridW = Div * NbDivX
GridH = Div * NbDivY
Border = 6dip

ScreenY0 = 0
ScreenH = GridH + 2 * Border
ScreenY1 = ScreenY0 + ScreenH

ScreenX0 = 0
ScreenW = GridW + 2 * Border
ScreenX1 = ScreenX0 + ScreenW

GridY0 = Border
GridY1 = GridY0 + GridH
GridX0 = Border
GridX1 = GridX0 + GridW
GridYm = GridH / 2

rectGrid.Initialize(0, 0, GridW, GridH)
rectScreen.Initialize(0, 0, ScreenW, ScreenH)
ScreenCol = Colors.White
GridLineCol = Colors.Gray

pnlOcilloscope.Initialize(""")
Activity.AddView(pnlOcilloscope, 0, 0, 100%x, 100%y)

pnlScreen.Initialize(""")
pnlOcilloscope.AddView(pnlScreen, ScreenX0, ScreenY0, ScreenW, ScreenH)
cvsScreen.Initialize(pnlScreen)

pnlGraph.Initialize(""")
pnlOcilloscope.AddView(pnlGraph, GridX0, GridY0, GridW, GridH)
cvsGraph.Initialize(pnlGraph)

pnlCursor.Initialize("pnlCursor")
pnlOcilloscope.AddView(pnlCursor, GridX0, GridY0, GridW, GridH)
cvsCursor.Initialize(pnlCursor)

pnlControl.Initialize(""")
Activity.AddView(pnlControl, ScreenX1, 0, 100%x - ScreenX1, 54dip)
Dim cbg As ColorDrawable
cbg.Initialize(Colors.RGB(255, 196, 196), 0)
pnlControl.Background = cbg

Dim w1, w2, w3, t1, h1 As Int
w1 = 4dip
w2 = (pnlControl.Width - 4 * w1) / 3
w3 = w1 + w2
t1 = 4dip
h1 = 52dip
btnStart.Initialize("btnStart")
pnlControl.AddView(btnStart, w1, t1, w2, h1)
btnStart.Text = "Start"

btnStop.Initialize("btnStop")
pnlControl.AddView(btnStop, w1 + w3, t1, w2, h1)
btnStop.Text = "Stop"

btnSingleShot.Initialize("btnSingleShot")
pnlControl.AddView(btnSingleShot, w1 + 2 * w3, t1, w2, h1)
btnSingleShot.Text = "Single Shot"

scvControl.Initialize(480dip)
pnlOcilloscope.AddView(scvControl, ScreenX1, pnlControl.Height, 100%x - ScreenX1, 100%y - pnlControl.Height)
scvControl.Panel.LoadLayout("controls")
scvControl.Panel.Width = scvControl.Width
Dim cbg As ColorDrawable
  cbg.Initialize(Colors.RGB(196, 196, 255), 0)
  scvControl.Panel.Background = cbg
  scvControl.Color = Colors.RGB(196, 196, 255)

  pnlCurveTools.Initialize(""")
  pnlOcilloscope.AddView(pnlCurveTools, 0, ScreenY1, ScreenX1, 100%y - ScreenY1)

Dim cbg As ColorDrawable
  cbg.Initialize(Colors.RGB(255, 196, 196), 0)
  pnlCurveTools.Background = cbg

  pnlDispValues.Initialize(""")
  pnlOcilloscope.AddView(pnlDispValues, 0, ScreenY1, ScreenX1, 100%y - ScreenY1)

Dim cbg As ColorDrawable
  cbg.Initialize(Colors.RGB(255, 236, 153), 0)
  pnlDispValues.Background = cbg
  pnlDispValues.Visible = False

InitCurves
InitCalcCurves
InitSpinners

Dim ww As Float
  ww = pnlDispValues.Width / 4
  lblValue0.Initialize(""")
  pnlDispValues.AddView(lblValue0, 0, 0, ww, pnlDispValues.Height)
  lblValue0.TextColor = Curve(0).Color
  lblValue0.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
  ' lblValue1.Initialize(""")
  ' pnlDispValues.AddView(lblValue1, ww, 0, ww, pnlDispValues.Height)
  ' lblValue1.TextColor = Curve(1).Color
  ' lblValue1.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
  ' ' lblValue2.Initialize(""")
  ' pnlDispValues.AddView(lblValue2, 2 * ww, 0, ww, pnlDispValues.Height)
  ' lblValue2.TextColor = Curve(2).Color
  ' lblValue2.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
  ' ' lblValue3.Initialize(""")
  ' pnlDispValues.AddView(lblValue3, 3 * ww, 0, ww, pnlDispValues.Height)
  ' lblValue3.TextColor = Curve(3).Color
'   lblValue3.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL

    For i = 0 To 0
        Dim cbx As CheckBox
        cbx.Initialize("cbxDrawCurve")
        pnlCurveTools.AddView(cbx, 6dip + i * 66dip, 0, 60dip, 50dip)
        cbx.Tag = i
        cbx.Text = " " & (i + 1)
        cbx.Typeface = Typeface.DEFAULT_BOLD
        cbx.Color = Curve(i).Color
        cbx.Checked = True
    Next

    lblScale0.Color = Curve(0).Color

    lblOffset0.Color = Curve(0).Color

    edtOffset0.Text = Curve(0).Offset

    Select ScopeMode
        Case "SCOPE"
            rbtScopeScope.Checked = True
        Case "MEM"
            rbtScopeMEM.Checked = True
        Case "ROLL"
            rbtScopeROLL.Checked = True
    End Select
End Sub

Sub Activity_Resume
    InitGrid
End Sub

Sub Activity_Pause (UserClosed As Boolean)
    btnStop_Click
End Sub

Sub InitGrid
    Dim i As Int
    Dim x, y As Float

    cvsScreen.DrawRect(rectScreen, ScreenCol, True, 1)
    For i = 0 To NbDivY
        y = GridX0 + i * Div
Experimentación con tarjeta BusPirate y migración a una tablet

Sub cvsScreen.DrawLine(GridX0, y, GridX1, y, GridLineCol, 1dip)
Next
For i = 0 To NbDivX
    x = GridY0 + i * Div
cvsScreen.DrawLine(x, GridY0, x, GridY1, GridLineCol, 1dip)
Next
pnlScreen.Invalidate

cvsGraph.DrawRect(rectGrid, Colors.Transparent, True, 1)
pnlGraph.Invalidate

cvsCursor.DrawRect(rectGrid, Colors.Transparent, True, 1)
pnlCursor.Invalidate
End Sub

Sub btnStart_Click
    'Timer1.Enabled = True
    SingleShot = False
    Stopped = False
    ii = 0
    xx = -dx
    EraseCurves
End Sub

Sub btnStop_Click
    Timer1.Enabled = False
    SingleShot = False
    Stopped = True
End Sub

Sub btnSingleShot_Click
    Timer1.Enabled = False
    SingleShot = True
    Stopped = False
    ii = 0
    xx = 0
    EraseCurves
End Sub

Sub spnTimeScale_ItemClick (Position As Int, Value As Object)
    dt = Value / 10
    t = 0
    Timer1.Initialize("Timer1", dt * 1000)
End Sub
Sub spnScale_ItemClick (Position As Int, Value As Object)
    Dim spn As Spinner

    spn = Sender
    Curve(spn.Tag).Scale = Div / Value
End Sub

Sub edtOffset_FocusChanged (HasFocus As Boolean)
    Dim edt As EditText
    Dim val As Double

    If HasFocus = False Then
        edt = Sender
        Curve(edt.Tag).Offset = edt.Text
    End If
End Sub

Sub rbtScope_CheckedChange(Checked As Boolean)
    Dim rbt As RadioButton

    btnStop_Click
    rbt = Sender
    ScopeMode = rbt.Tag
    btnStart_Click
End Sub

Sub Timer1_Tick
    Dim i, j As Int

    t = t + dt
    xx = xx + dx
    ii = ii + 1
    If ii > 100 Then
        If SingleShot = True Then
            Timer1.Enabled = False
            SingleShot = False
            Stopped = True
            Return
        Else
            Select ScopeMode
                Case "MEM"
                    xx = 0
                    ii = 0
                Case "SCOPE"
                    'GetValues
                    DrawCurves
            End Select
        End If
    End If
End Sub
EraseCurves
xx = 0
ii = 0
'GetValues
DrawCurves
Case "ROLL"
ii = 100
xx = 100 * dx
For i = 0 To 3
    For j = 0 To 99
        CurveVal(i, j) = CurveVal(i, j + 1)
    Next
Next
ScopeRolling = True
'GetValues
DrawCurves
End Select
Return
End If
End If
'GetValues
DrawCurves
End Sub
Sub DrawCurves2
For i = 0 To ii - 1
    y2(0) = GridYm + (-Curve(0).Offset - CurveVal(0, i)) * Curve(0).Scale
    If ii > 0 Then
        cvsGraph.DrawLine(xxx(i) - dx, y1(0), xxx(i), y2(0), Curve(0).Color, Curve(0).Width)
        End If
    y1(0) = y2(0)
Next
SingleShot = False
Log("Draw")
End Sub
Sub DrawCurves
Dim i, j As Int
Dim r1, r2 As Rect
Dim x, yy1(4), yy2(4) As Float
Log("drawcurves")
If SingleShot = False Then
    Select ScopeMode
        Case "MEM"
            r1.Initialize(xx, 0, xx + dx, GridH)
End If
End Sub
cvsGraph.DrawRect(r1, Colors.Transparent, True, 1)

Case "ROLL"
    If ScopeRolling = True Then
        cvsGraph.DrawRect(rectGrid, Colors.Transparent, True, 1)
        For i = 0 To CurvesNb
            If Curve(i).Draw = True Then
                yy1(i) = GridYm + (-Curve(i).Offset - CurveVal(i, 0)) * Curve(i).Scale
                For j = 1 To 99
                    x = j * dx
                    yy2(i) = GridYm + (-Curve(i).Offset - CurveVal(i, j)) * Curve(i).Scale
                    cvsGraph.DrawLine(x - dx, yy1(i), x, yy2(i), Curve(i).Color, Curve(i).Width)
                    yy1(i) = yy2(i)
                Next
            End If
        Next
    End If
End Select
End If
For i = 0 To CurvesNb
    If Curve(i).Draw = True Then
        y2(i) = GridYm + (-Curve(i).Offset - CurveVal(i, ii)) * Curve(i).Scale
        If ii > 0 Then
            cvsGraph.DrawLine(xx - dx, y1(i), xx, y2(i), Curve(i).Color, Curve(i).Width)
        End If
        y1(i) = y2(i)
    End If
Next
pnlGraph.Invalidate
DoEvents
End Sub

Sub GetValues
    Dim i As Int

    For i = 0 To CurvesNb
        CurveVal(i, ii) = USBValue / 0xFF00
    Next
End Sub

Sub EraseCurves
    cvsGraph.DrawRect(rectGrid, Colors.Transparent, True, 1)
End Sub
Sub cbxDrawCurve_CheckedChange(Checked As Boolean)
    Dim cbx As CheckBox
    cbx = Sender
    Curve(cbx.Tag).Draw = Checked
End Sub

Sub pnlCursor_Touch (Action As Int, X As Float, Y As Float) As Boolean 'Return True to consume the event
    If Stopped = False Then
        Return
    End If

    Select Action
        Case Activity.ACTION_DOWN
            pnlDispValues.Visible = True
            cx = X
            If X >= 0 And X <= GridW Then
                DrawCursor(X)
                DispValues(X)
            End If
        Case Activity.ACTION_MOVE
            If X >= 0 And X <= GridW Then
                DrawCursor(X)
                DispValues(X)
            End If
        Case Activity.ACTION_UP
            cvsCursor.DrawLine(cx, 0, cx, GridH, Colors.Transparent, 1)
            pnlCursor.Invalidate
            pnlDispValues.Visible = False
    End Select
    Return True
End Sub

Sub DrawCursor(x As Float)
    cvsCursor.DrawLine(cx, 0, cx, GridH, Colors.Transparent, 1)
    cx = x
    cvsCursor.DrawLine(cx, 0, cx, GridH, Colors.Red, 1)
    pnlCursor.Invalidate
End Sub

Sub DispValues(x As Int)
    Dim i As Int

    i = 100 / GridW * x
    lblValue0.Text = NumberFormat(CurveVal(0, i), 1, 6)
lblValue1.Text = NumberFormat(CurveVal(1, i), 1, 6)
lblValue2.Text = NumberFormat(CurveVal(2, i), 1, 6)
lblValue3.Text = NumberFormat(CurveVal(3, i), 1, 6)

End Sub

Sub InitCurves
Curve(0).Color = Colors.Red
Curve(1).Color = Colors.Blue
Curve(2).Color = Colors.Black
Curve(3).Color = Colors.RGB(64, 192, 0)

Curve(0).Width = 1dip
Curve(1).Width = 1dip
Curve(2).Width = 1dip
Curve(3).Width = 1dip

Curve(0).Scale = 20
Curve(1).Scale = 20
Curve(2).Scale = 20
Curve(3).Scale = 20

Curve(0).Offset = 0
Curve(1).Offset = 1
Curve(2).Offset = -1
Curve(3).Offset = 2

Curve(0).Draw = True
Curve(1).Draw = False
Curve(2).Draw = False
Curve(3).Draw = False

End Sub

Sub InitCalcCurves
w(0) = 2 * cPI * 2.1
w(1) = 2 * cPI * 3.7
w(2) = 2 * cPI * 4.3
w(3) = 2 * cPI * 5.7

a(0) = 1.0
a(1) = 2.0
a(2) = -1.0
a(3) = 1.5

End Sub

Sub InitSpinners
Dim i As Int
TimeScale(0) = 10
TimeScale(1) = 5
TimeScale(2) = 2
TimeScale(3) = 1
TimeScale(4) = 0.5
TimeScale(5) = 0.2
TimeScale(6) = 0.1
TimeScale(7) = 0.05
TimeScale(8) = 0.02
TimeScale(9) = 0.01

SignalScale(0) = 10
SignalScale(1) = 5
SignalScale(2) = 2
SignalScale(3) = 1
SignalScale(4) = .5
SignalScale(5) = .2
SignalScale(6) = .1
SignalScale(7) = .05
SignalScale(8) = .02
SignalScale(9) = .01

For i = 0 To 9
    spnTimeScale.Add(TimeScale(i))
    spnScale0.Add(SignalScale(i))
Next
spnTimeScale.SelectedIndex = 6
spnScale0.SelectedIndex = 3
Curve(0).Scale = Div / spnScale0.SelectedItem
End Sub

Sub btnExit_Click
    ExitApplication
End Sub

Sub btnOpen_Click
    If usb1.UsbPresent(1) = usb1.USB_NONE Then ' Ver_2.4
        Log("Msgbox - no device")
        MsgBox("No USB device or accessory detected!", "Error")
        Log("Msgbox - returned")
        Return
    End If
    Log("Checking permission 1")
If (usb1.HasPermission(1)) Then ' Ver_2.4
' MsgBox(usb1DeviceInfo(1), "Device Information 1") ' Ver_2.4
Dim dev As Int
dev = usb1.Open(115200, 1) ' Ver_2.4

If dev <> usb1.USB_NONE Then
  Log("Connected successfully! 1")
  btnOpen.Enabled = False
  btnSend.Enabled = True
  btnReset.Enabled = True
  astreams1.Initialize(usb1.GetInputStream, usb1.GetOutputStream, "astreams1")
  zbyte (0) = 0x0D
  astreams1.Write(zbyte)
  zbyte (0) = 0x0D
  astreams1.Write(zbyte)
  Delay(1000)
  zbyte (0) = 0x00
  For i=1 To 20
    astreams1.Write(zbyte)
  Next
  zbyte (0) = 0x0D
  astreams1.Write(zbyte)
  Timer2.Enabled = True
  zbyte (0) = 0x14
  astreams1.Write(zbyte)
  flag = True
  zbyte (0) = 0x0D
  astreams1.Write(zbyte)
Else
  Log("Error opening USB port 1")
End If
Else
  usb1.RequestPermission(1) ' Ver_2.4
End If
End Sub

Sub btnReset_Click
  btnOpen.Enabled = True
  btnClose.Enabled = False
  btnSend.Enabled = False
  zbyte (0) = 0x0D
  astreams1.Write(zbyte)
  Delay(1000)
  zbyte (0) = 0x0F
  astreams1.Write(zbyte)
  astreams1.Close
End Sub

Sub Astreams1_NewData (Buffer() As Byte)
' You must check for DeviceInfo or analyze Buffer data to know what is connected to the USB
' The order of the USB could change as you plug them and could change when changing the hub port they are connected to
'Log("NewData 1")
'Log(BytesToString(Buffer, 0, Buffer.Length, "UTF8"))
If flag=True Then
    'adc.Text=BytesToString(Buffer, 0, Buffer.Length, "UTF8")
    'adc.Text= Counter
data(Counter)=Buffer(0)
    If Buffer.Length > 1 Then
        Counter= 1
        data(Counter)=Buffer(1)
    End If
End If

'Log(Chr(data(Counter)))
Counter= Counter + 1
If Counter == 2 Then
    Counter=0
    USBValue= data(0)
    USBValue= USBValue*256
    USBValue= USBValue + data(1)
    If SingleShot== True Then
        'Log(USBValue)
        CurveVal(0, ii) = (USBValue/1023)*6.6
        'Log(CurveVal(0,ii))
        't= t+dt
        'xx= xx+ dx
        xxx (ii+1)= xxx(ii)+ dx
        ii= ii+1
        'DrawCurves
        If ii > 100 Then
            DrawCurves2
        End If
    End If
End If
End If
End Sub
Sub Timer2_Tick
    'Log(bc.HexFromBytes(data))
    zbyte(0)=0x14
    astreams1.Write(zbyte)
End Sub

Sub AStreams1_Error
    Log("Error: " & LastException)
    astreams1.Close
End Sub
Sub AStreams1_Terminated
    Log("Terminated")
    astreams1.Close
End Sub

Sub Delay(nMilliSecond As Long)
    Dim nBeginTime As Long
    Dim nEndTime As Long
    nEndTime = DateTime.Now + nMilliSecond
    nBeginTime = DateTime.Now
    Do While nBeginTime < nEndTime
        nBeginTime = DateTime.Now
        'Log(nBeginTime)
        If nEndTime < nBeginTime Then
            Return
        End If
        DoEvents
        Loop
End Sub