```
! ------------------------------------------------------------------------
! DIFFERENTIAL PULSE VOLTAMMETRY (DPV)
! This is a test program for UWED to make basic DPV recordings
! Written by Alar Ainla, 2016-2018. Whitesides Group, Harvard.
! ------------------------------------------------------------------------

! ------------------------------------------------------------------------
! OPERATION PARAMETERS
! ------------------------------------------------------------------------

InitV=0.5           ! [V] This is the starting potential
VoltageRange=-1     ! [V] This is the voltage change during the scan
! For example, if InitV=0.5V and VoltageRange=-1V, then scan starts at 0.5V
! and will end at 0.5V-1V=-0.5V
pulsePeriod=100     ! [ms] Period of DPV pulse
pulseWidth=50       ! [ms] Pulse width
StepE=5             ! [mV] Step size. Potential change in each DPV cycle
pulseAmp=25         ! [mV] DPV amplitude. DPV amplitude is different from SWV.
! If SWV is performed with DPV program, pulsePeriod=2*pulseWidth,
! but pulseAmp in DPV = 2* pulseAmp in SWV

! ------------------------------------------------------------------------
! PROGRAM CODE
! ------------------------------------------------------------------------

! Helping variables
DIM timeStep1 AS INTEGER
DIM timeStep2 AS INTEGER
timeStep1=pulseWidth/2
timeStep2=(pulsePeriod-pulseWidth)/2

! Here we lock working electrode potential W[d]=22500 W[V]=1.5031
! And only varied is Reference electrode potential
! Constants then
! ADC[d]=(ADC[V])*2.01513E4+2.38491E2
! ADC[V]=(ADC[d]-2.38491E2)/2.01513E4
! Curr[uA]=0.00608*(dADC-30694)-0.04 ! Working electrode current
! (W-R)[V]=R[d]*(-6.64898E-5)+1.50310 ! Working electrode potential vs ref
Rslope=1.5040E4 ! This is number of digits per volt when Ref is changed

! Calculate program data structure
DIM pulseAmpD AS INTEGER
pulseAmpD=(pulseAmp/1000)*Rslope ! This is pulse amplitude, odd -1, even +1
DIM initW AS LONG
initW=22500 ! This is now constant and not calculated
DIM initR AS LONG
initR=22606-Rslope*InitV ! This should start as close to zero as possible, when InitV=0
nrScans=1

DIM StepHeightAbs AS INTEGER
nrSteps1=2*1000*ABS(VoltageRange)/ABS(StepE)
stepHeightAbs=ABS(StepE)*Rslope/1000
DIM stepR(nrScans) AS INTEGER
DIM nrSteps(nrScans) AS INTEGER
DIM nrDataPoints AS LONG
nrDataPoints=nrSteps1+2
IF(VoltageRange>0)THEN
    stepR(1)=(-1)*stepHeightAbs
```

```
ELSE
    stepR(1)=stepHeightAbs
END IF
    nrSteps(1)=nrSteps1
! Output these to console for feedback
PRINT "InitW"
PRINT initW
PRINT "InitR"
PRINT initR
PRINT "stepR"
PRINT stepR
PRINT "nrSteps"
PRINT nrSteps
PRINT "nrScans"
PRINT nrScans
PRINT "nrDataPoints"
PRINT nrDataPoints

! Resulting data storage
DIM Vw(nrDataPoints) AS DOUBLE ! Potential in volts
DIM IuA(nrDataPoint) AS DOUBLE ! Electrode current in uA
DIM dWe(nrDataPoints) AS LONG ! Working electrode as set
DIM dRe(nrDataPoints) AS LONG ! Reference electrode as set
DIM dADC(nrDataPoints) AS LONG ! ADC value
DIM Ts(nrDataPoints) AS LONG ! Time step
DIM Ni(nrDataPoints) AS INTEGER ! Index of point in scan
DIM DPi AS INTEGER ! Data point counter
DIM SCi AS INTEGER ! Scan counter
DIM ScanRunning AS INTEGER ! if 0 does not run, if 1 running
ScanRunning=0
DIM lastIndex AS INTEGER ! Last receiveing Index, to avoid double reading
lastIndex=0

! -- GUI --
! - Build GUI -
DIM startButton AS Button ! This button starts the measurement
DIM myMess AS Label ! This is text message area
DIM p1 AS Plot ! This is the result plot
DIM pCV AS PlotPoint ! This is one dataset
DIM myCV(nrDataPoints,2)
DIM myCV2(nrDataPoints,2)
FOR i=1 TO nrDataPoints
    myCV(i,1)=0
    myCV(i,2)=0
    myCV2(i,1)=0
    myCV2(i,2)=0
NEXT i
p1=Graphics.newPlot
pCV=p1.newPlot(myCV2)
p1.setView(-1,-100,1,100,0)
p1.setXaxisLabel("Potential Working vs Ref (V)")
p1.setYaxisLabel("Current working electrode (µA)")
p1.setTitle("Electrochemical DPV with UWED")
startButton=Graphics.newButton(10,10)
startButton.setTitle("Start")
myMess=Graphics.newLabel(110,10,200,20)
myMess.setText("....")
System.ShowGraphics(1)
```

```
! -- BLE --
print "Hello BLE"
DIM bm AS BLEPeripheral
ble.startble
DIM uuid(0) AS STRING
DIM services(1) AS STRING
DIM bleWriteChar AS BLECharacteristic
ble.startscan(uuid)

END

! ------------------------------------------------------------------------
! FUNCTIONS
! ------------------------------------------------------------------------

SUB InitializeUWED()
    DPi=0 ! No data points yet
    SCi=0 ! Scan counter
    ! Make graph empty
    FOR i=1 TO nrDataPoints
        myCV(i,1)=0
        myCV(i,2)=0
        myCV2(i,1)=0
        myCV2(i,2)=0
    NEXT i
    ! Start setting up
    sendToUWED("A("&STR(initR)&")")
    sendToUWED("B("&STR(initW)&")")
    sendToUWED("D(3)")
    sendToUWED("C(1)")
END SUB

SUB InitializeScan()
    SCi=SCi+1 ! Increase scan counter
    IF SCi>nrScans THEN ! If increase scan counter is larger, DONE
     sendToUWED("C(0)") ! Put UWED to potentiometric (off state)
     sendToUWED("O(0)") ! Put DPV mode off
        ! Save data
        OPEN "DPVData.txt" FOR OUTPUT AS #1
        FOR i=1 TO DPi
            PRINT #1, STR(dRe(i))&" "&STR(DWe(i))&" "&STR(dADC(i))&_
                " "&STR(Ts(i))&" "&STR(Ni(i))
        NEXT i
        CLOSE #1
        ! Send now email
        DIM em AS Email
        DIM em_mess AS STRING
        em_mess="This is data from DPV measurement from iPhone"
        em=System.newEmail("email@emailserver.com","DPV Data", em_mess)
        em.addAttachment("DPVData.txt", "text/plain")
        em.send
        myMess.setText("DONE")
        ScanRunning=0
    ELSE ! Initialize a new scan
        ! Overlay Square wave and stair function. Step is made in every cycle
        sendToUWED("E("&STR(timeStep1)&")") ! High step
        sendToUWED("O("&STR(timeStep2)&")") ! Low step
```

```
        sendToUWED("G("&STR(stepR(SCi)-pulseAmpD)&")") ! Low step
        sendToUWED("I("&STR(pulseAmpD)&")") ! High step
        sendToUWED("L("&STR(nrSteps(SCi))&")")
        sendToUWED("M()") ! Run the sequence
        ScanRunning=1
    END IF
END SUB

! - Process incoming data
SUB procData(input_array() AS INTEGER)
    PRINT "Index: "&CHR(input_array(1)) ! Feedback, so we can see continuity
    ! First check that there is no dublication
    IF(input_array(1)=lastIndex) THEN
        GOTO OutSub ! If same as last time do nothing
    END IF
    lastIndex=input_array(1) ! Otherwise put it same
    ! Now look if new data came M character on second position
    IF(input_array(2)=ASC("M"))THEN ! New data came
        IF ScanRunning=1 THEN ! Only if scan is running there is something to come
            DPi=DPi+1
            dRe(DPi)=ConvertToNumberBin(input_array,3,4)
            dADC(DPi)=ConvertToNumberBin(input_array,5,6)
            DPi=DPi+1
            dRe(DPi)=ConvertToNumberBin(input_array,7,8)
            dADC(DPi)=ConvertToNumberBin(input_array,9,10)
            Ts(DPi)=ConvertToNumberBin(input_array,11,14)
            Ts(DPi-1)=Ts(DPi)
            dWe(DPi)=ConvertToNumberBin(input_array,15,16)
            dWe(DPi-1)=dWe(DPi)
            Ni(DPi)=ConvertToNumberBin(input_array,17,18)
            Ni(DPi-1)=Ni(DPi)
            ! Do the math for visualization
            ! ODD
            VoltW=dRe(DPi-1)*(-6.64898E-5)+1.50310  ! Working electrode pot vs ref
            Curr=0.00608*(dADC(DPi-1)-30694)-0.04 ! Current in uA
            Curr=0.0063329*(dADC(DPi-1)-34732)
            myCV(DPi-1,1)=VoltW
            myCV(DPi-1,2)=Curr
            ! EVEN
            VoltW=dRe(DPi)*(-6.64898E-5)+1.50310  ! Working electrode pot vs ref
            Curr=0.00608*(dADC(DPi)-30694)-0.04 ! Current in uA
            Curr=0.0063329*(dADC(DPi)-34732)
            myCV(DPi,1)=VoltW
            myCV(DPi,2)=Curr
            ! Calculate pulse
            myCV2(DPi/2,1)=(myCV(DPi,1)+myCV(DPi-1,1))/2
            myCV2(DPi/2,2)=myCV(DPi,2)-myCV(DPi-1,2)
            ! Show graph
            pCV.setPoints(myCV2)
            p1.setView(-1,minY(myCV2),1,maxY(myCV2),0)
        END IF
    END IF
    ! If end of the scan start new
    IF (input_array(2)=ASC("m")) THEN ! End of scan
        InitializeScan ! Start next scan
    END IF
    ! In other cases do currently nothing
OutSub:
```

```
END SUB

! - Convert to Number from String source - currently not used
FUNCTION ConvertToNumberStr(input_array() AS INTEGER, _
                            begi AS INTEGER, _
                            endi AS INTEGER) AS LONG
    DIM nets AS STRING
    nets=""
    FOR i=begi TO endi
        nets=nets & chr(input_array(i))
    NEXT i
    ConvertToNumberStr=VAL(nets)
END FUNCTION

! - Convert to Number from Binary source - used in data receiving
FUNCTION ConvertToNumberBin(input_array() AS INTEGER, _
                            begi AS INTEGER, _
                            endi AS INTEGER) AS LONG
    DIM sum AS LONG
    sum=input_array(begi)
    FOR i=begi+1 TO endi
        sum=sum*256
        sum=sum+input_array(i)
    NEXT i
    ConvertToNumberBin=sum
END FUNCTION

! ------------------------------------------------------------------------
! EVENTS
! ------------------------------------------------------------------------

! When button is clicked
SUB touchUpInside(crtl AS Button, when AS DOUBLE)
    IF crtl=startButton THEN ! Send now
        myMess.setText("Start the scan")
        InitializeUWED
        InitializeScan
    END IF
END SUB

! BLE EVENTS and FUNCTIONS
! - Send command to UWED
SUB sendToUWED(inp AS STRING)
    DIM ax(LEN(inp)) AS INTEGER
    FOR i=1 TO LEN(INP)
        ax(i)=ASC(MID(inp,i,1))
    NEXT
    bm.writeCharacteristic(bleWriteChar,ax)
END SUB

! - First when device is discovered
SUB BLEDiscoveredPeripheral (   time AS DOUBLE, _
                                peripheral AS BLEPeripheral, _
                                services() AS STRING, _
                                advertisements(,) AS STRING, _
                                rssi)
    PRINT "Device Found";
    IF peripheral.bleName="UWED" THEN ! Right device found
```

```
        bm=peripheral
        ble.connect(bm)
        ble.stopScan
        print "Device is UWED"
    END IF
END SUB

! - When information about the device is fetched
SUB BLEPeripheralInfo(  time AS DOUBLE, _
                        peripheral AS BLEPeripheral, _
                        kind AS INTEGER, _
                        message AS STRING, _
                        err AS LONG)
    IF kind=1 THEN ! Connection completed
        peripheral.discoverServices(uuid)
    ELSE IF kind=2 OR kind=3 THEN ! Connection lost
        ble.connect(bm)
    ELSE IF kind=4 THEN ! Service dound
        DIM avServ(1) AS bleservice
        avServ=peripheral.services
        FOR a=1 TO UBOUND(services,1)
            FOR a=1 TO UBOUND(services,1)
                IF avServ(a).uuid="2220" THEN ! If right service id has been found
                    peripheral.discoverCharacteristics(uuid, avServ(a))
                    print "Service discovered"
                END IF
            NEXT
        NEXT
    END IF
END SUB

! - When information about service is obtained
SUB BLEServiceInfo( time AS DOUBLE, _
                    peripheral AS BLEPeripheral, _
                    service AS BLEService, _
                    kind AS INTEGER, _
                    message AS STRING, _
                    err AS LONG)
    IF kind=1 THEN
        DIM chx(1) AS blecharacteristic
        chx=service.characteristics
        FOR i=1 TO UBOUND(chx,1)
            IF service.uuid="2220" THEN
                IF chx(i).uuid="2221" THEN
                    peripheral.setNotify(chx(i),1) ! Notify if changes
                    print "Read characteristic found"
                ELSE IF chx(i).uuid="2222" THEN
                    bleWriteChar=chx(i)
                    print "Write characteristic found"
                END IF
                    myMess.setText("BLE CONNECTED!")
            END IF
        NEXT
    END IF
END SUB

! - If BLE Characteristic info comes
SUB BLECharacteristicInfo(  time AS DOUBLE, _
```

```
                            peripheral AS BLEPeripheral, _
                            characteristic AS BLECharacteristic, _
                            kind AS INTEGER, _
                            message AS STRING, _
                            err AS LONG)
    IF kind=2 THEN
        IF characteristic.uuid="2221" THEN
            DIM value(1) AS INTEGER
            value=characteristic.value
            procData(value)
        END IF
    END IF
END SUB

! ------------------------------------------------------------------------
! HELPING FUNCTIONS
! ------------------------------------------------------------------------

FUNCTION minY(inarray() as DOUBLE) AS DOUBLE
    a=inarray(1,2)
    for i=1 to DPi
        if a>inarray(i,2) then
            a=inarray(i,2)
        end if
    next
    minY=a
end function

function maxY(inarray() as DOUBLE) AS DOUBLE
    a=inarray(1,2)
    for i=1 to DPi
        if a<inarray(i,2) then
            a=inarray(i,2)
        end if
    next
    maxY=a
end function

! - END OF THE CODE -
```