HEADER FILE TO BE INCLUDED-

```
#include <Arduino.h>
#include <Streaming.h>
#include <Flash.h>
#include "CurveTracer.h"

CurveTracer curveTracer; // dynamic memory allocation is not allowed

void setup()
{ Serial.begin(9600);
  CurveTracer(); // run constructor here
}

void loop()
{ // Fill in the preferences here:
  bool printCurve = 1; // 0 = without curve, 1 = with curve
  int averaging = 100; // 1 = no averaging (filter out 50Hz on lamp)
  int stepTime_ms = 0; // for the cell capacity effect
  int mode = 1; // 1 = run, 2 = U calibration, 3 = I calibration

  switch(mode)
  { case 1: curveTracer.trace(printCurve, averaging, stepTime_ms);
break;
    case 2: curveTracer.calibrateUmeasurement(); break;
    case 3: curveTracer.calibrateImeasurement(); break;
  }
}
```

HEADER FILE TO BE INCLUDED-

```
#include <Dac.h>

Dac dac(4, 0); // dynamic memory allocation is not allowed

/*
                      SOLAR
          I |
            |       < measurement direction
            |  * * * * *
            |                *
            |                  *
            |                    *
            |                     *
            |                      *
            |                       *
            |
```

```
            --------------------------
              ^                       U
            mVstop

*/

CurveTracer::CurveTracer():
UpinADC(1), IpinADC(2), mAoffsetError(0), mVstart(0), averaging(1)
{ dacOut(0); // turn BUZ10 off
}

int CurveTracer::trace(bool printCurve, int _averaging, int
stepTime_ms)
{ const int mVstep=5;
  averaging = _averaging;
  int Wire_mOhm = 2;
  int mV=0, mA=0, mW=0, mWpeak=0, mVnext=0;
  //if(printCurve) Serial << F("\n\nmV mA mW ");

  for(int i=0; i<=1023; i++)
  { measure(i, stepTime_ms, mV, mA);
    correctForWire(Wire_mOhm, mV, mA);
    if(i==0)
    { mAoffsetError = mA; // I should be 0 at start but isn't
      mVstart = mV;
      mVnext = mV;
    }
    repairOffsetError(mV, mA);
    calcPower(mV, mA, mW, mWpeak);

    if(mV <= mVnext) // measurement direction 80 75 70mV, reduce
amount of measurement data: 100, 105 instead of 100, 101, 102, 103,
104, 105 mV
    { if(printCurve) dataOut(mV, mA, mW);
      //testOut(i, mVnext); // only for test
      mVnext = mV - mVstep;
    }
  }
  //Serial << F("\nmWpeak: ") << mWpeak;
  delay(1); // pause time to read from screen
  return mWpeak;
}

void inline CurveTracer::measure(int i, int stepTime_ms, int &mV, int
&mA)
{ dacOut(i);
  delay (stepTime_ms);
  measureUmV(mV);
  measureImA(mA);
}

bool inline CurveTracer::measureUmV(int &mV)
```

```
{ const float aU = 0.00074356; // from calibration
  const float bU = 0.00074969; // from calibration
  int adcVal=0;

  adcIn(adcVal, UpinADC, averaging);
  mV = (aU * adcVal + bU) * 1000; // eliminate offset voltage
  return true;
}

bool inline CurveTracer::measureImA(int &mA)
{ const float aI = 0.00658427; // from calibration
  const float bI = 0.06617978; // from calibration
  int adcVal=0;

  adcIn(adcVal, IpinADC, averaging);
  mA = (aI * adcVal + bI) * 1000; // eliminate offset voltage
  return true;
}

/*
Formula: UmV = (aU * adc + bU) * 1000
Run calibrateUmeasurement()
Determine aU and bU:
1) Put a voltage (U1) of about 0.1V to the solarcell + connector.
Measure U1 and read the adc value (=adc1).
2) Put a voltage (U2) of about 0.7V to the solarcell + connector.
Measure U2 and read the adc value (=adc2).
3) Use Solarcell curvetracer calibration.xls.
   Calculate aU and bU and add the values to the lidfunction
measureUmV():
   aU = (U1 - U2)/(adc1 - adc2)
   bU = U2 - aU * adc2
4) Run again and check the accuracy and linearity
*/

void inline CurveTracer::calibrateUmeasurement()
{ int adc=0, mV=0;
  dacOut(0); // BUZ10 is off
  delay(500);
  if(!adcIn(adc, UpinADC, 32)) return;
  measureUmV(mV);
  Serial << F("Calibration U adc: ") << adc << " UmV: " << mV << endl;
}

/*
Formula: ImA = (aI * adc + bI) * 1000
Run calibrateImeasurement()
Determine aI and bI:
1) Put a current (I1) of about 0.5A to the solarcell + connector.
Measure I1 and read the adc value (=adc1).
2) Put a current (I2) of about 5A to the solarcell + connector.
Measure I2 and read the adc value (=adc2).
```

```
3) Use Solarcell curvetracer calibration.xls.
   Calculate aI and bI and add the values to the lidfunction
measureImA():
   aI = (I1 - I2)/(adc1 - adc2)
   bI = I2 - aI * adc2
4) Run again and check the accuracy and linearity
*/

void inline CurveTracer::calibrateImeasurement()
{ int adc=0, mA=0;
  dacOut(1023); // BUZ10 is on
  delay(500);
  if(!adcIn(adc, IpinADC, 32)) return;
  measureImA(mA);
  Serial << F("Calibration I adc: ") << adc << " ImA: " << mA << endl;
}

bool inline CurveTracer::checkAdcVal(int val)
{ if(val >= 1023)
  { //Serial << F("\nADC overflow\n");
    return false;
  }
  return true;
}

void inline CurveTracer::dacOut(int val)
{ //if(!dac.write(val)) Serial << F("\nDac error ") << val;
}

void inline CurveTracer::dataOut(int mV, int mA, int mW)
{ Serial << endl << mV << " " << mA << " " << mW;
}

void inline CurveTracer::correctForWire(int mOhm, int &mV, int mA)
{ mV = mV + mA * mOhm / 1000;
}

void inline CurveTracer::repairOffsetError(int &mV, int &mA)
{ if(mA <= mAoffsetError)
  { mA = 0;
    mV = mVstart;
  }
}

void inline CurveTracer::calcPower(int mV, int mA, int &mW, int
&mWpeak)
{ mW = (float)mA * mV / 1000;
  if(mW > mWpeak) mWpeak = mW;
}

bool inline CurveTracer::adcIn(int &adcVal, int adcPin, int adcCount)
{ long adcValLong = 0;
```

```
  for(int i=0; i<adcCount; i++)
  { int temp = analogRead(adcPin);
    if(!checkAdcVal(temp)) return false;
    adcValLong += temp;
  }
  adcValLong /= adcCount;
  adcVal = (int)adcValLong;
}

/*
void inline CurveTracer::testOut(int i, int mVnew)
{ Serial << " i: " << i << " mVnew: " << mVnew << " mAoffsetError: "
<< mAoffsetError << " mVstart: " <<  mVstart;
}
*/
```

MAIN PROGRAM FOR THE ARDUINO BOARD-

```
#ifndef CURVETRACER_H
#define CURVETRACER_H

class CurveTracer
{
public:
  CurveTracer();
  int trace(bool printCurve, int _adcCount, int stepTime_ms);
  void inline calibrateUmeasurement();
  void inline calibrateImeasurement();

private:
  void inline measure(int i, int stepTime_ms, int &mV, int &mA);
  bool inline measureUmV(int &mV);
  bool inline measureImA(int &mA);
  bool inline checkAdcVal(int adc);
  bool inline adcIn(int &adcVal, int adcPin, int adcCount=1);
  void inline dacOut(int val);
  void inline dataOut(int mV, int mA, int mW);
  void inline correctForWire(int mOhm, int &mV, int mA);
  void inline repairOffsetError(int &mV, int &mA);
  void inline calcPower(int mV, int mA, int &mW, int &mWpeak);
  /*void inline testOut(int i, int mVnew);*/

  int UpinADC, IpinADC, mAoffsetError, mVstart, averaging;
};

#endif
```