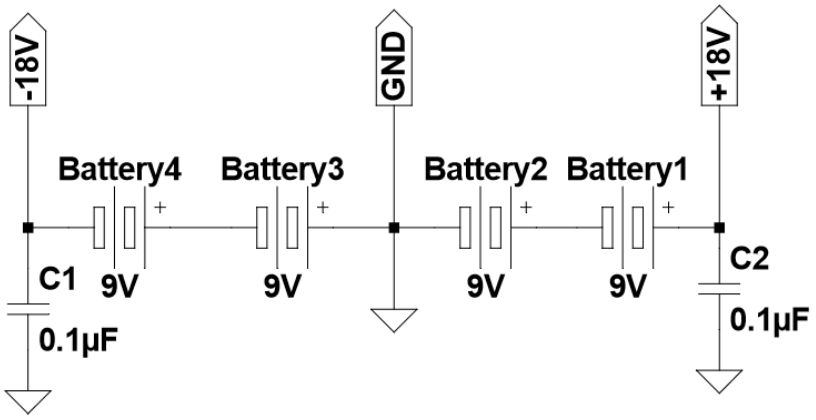
An Uno based ±12V Curve Tracer by simple-circuit 12-8-2019

Circuit Design Theory:

The previous project's curve tracer's waveform generator used a PWM output to generate a 0 to 5V voltage swing. To obtain higher voltages and negative swings, we need amplification and a dual voltage power supply.

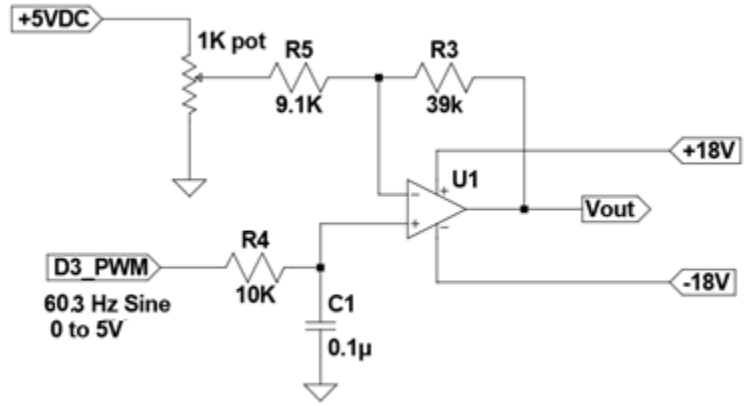
A dual power supply is easily constructed from 9V batteries. It takes four batteries to get plus and minus 18V. We need a minimum of plus and minus 15V in our design. Fifteen volts can be generated from the Uno 5VDC pin. However, you will need a DC to DC converter such as a muRata NKA0515SC or equivalent.

The battery power supply has enough current and voltage to destroy the Uno and our amplifier circuit. Keep this in mind when connecting the circuit. A simple miswire can be catastrophic.



If your supply circuit does not use a power switch (DPDT). You will have to remove two batteries, #1 and #3 to turn off the supply. The 0.1uF capacitors help to short high frequencies to ground. They are needed to keep the amplifier stable.

The plus and minus sine wave generator circuit:

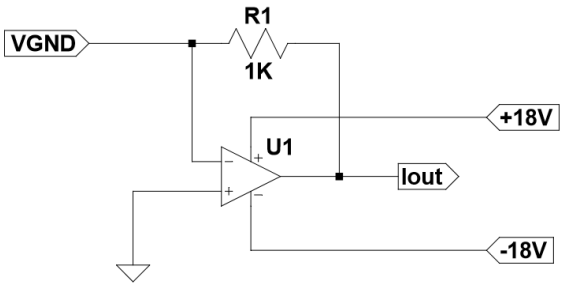


An RC filter with a cut off frequency of 159 Hz is used to filter a PWM modulated sine wave. This is a good cutoff frequency for filtering the 31Khz PWM signal. However, it will reduce the 60.2Hz sine amplitude slightly. The filtered PWM sine will swing ±2.31V about a mid level of 2.5V.

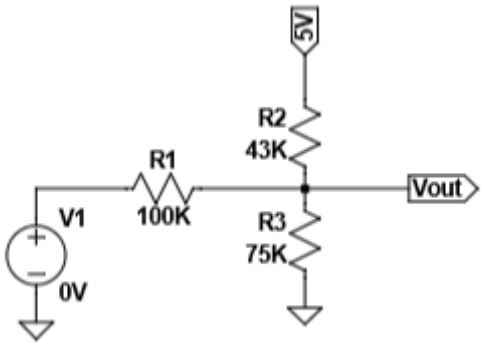
The amplifier is a non-inverting op amp configuration. The gain is approximately R3/R5 + 1. The potentiometer impedance adds about 200 ohms to the R5 value. So the gain is around 39K/9.3K + 1 = 5.194. We intend to use half of the 5V signal for the plus swing and half for the negative swing. Therefore each swing will be 2.31V * 5.194 = 12.0V.

The 1K potentiometer is used to adjust the offset of the circuit to zero. If we didn't apply a midpoint voltage, the output would only swing from zero to 24V. The signal generator zero is set by programming the PWM signal to half (128) then adjusting the potentiometer until the amplifier output is 0V.

To simplify current measurements, we use a dual op amp (gaining our needed amplifier for free). A virtual ground is used for the negative pin the curve tracer. When current is applied to VGND, the Iout voltage changes at -1V/mA. In other words, the circuit forces the current through R1 to equal the current injected into VGND. This keeps the inverting input at the same potential as the non-inverting input (ground). The advantage to this circuit is that the current is a direct voltage measurement.



Now we attack the second problem associated with bipolar measurement. The analog input voltages are limited from zero to 5V. We can use simple resistor networks to scale and offset the inputs. For current, we will use a -5V to +5V input range and a -5mA to +5mA current sense scale. For voltage, we will use a -12V to +12V input range scaled to 0 to 5V.

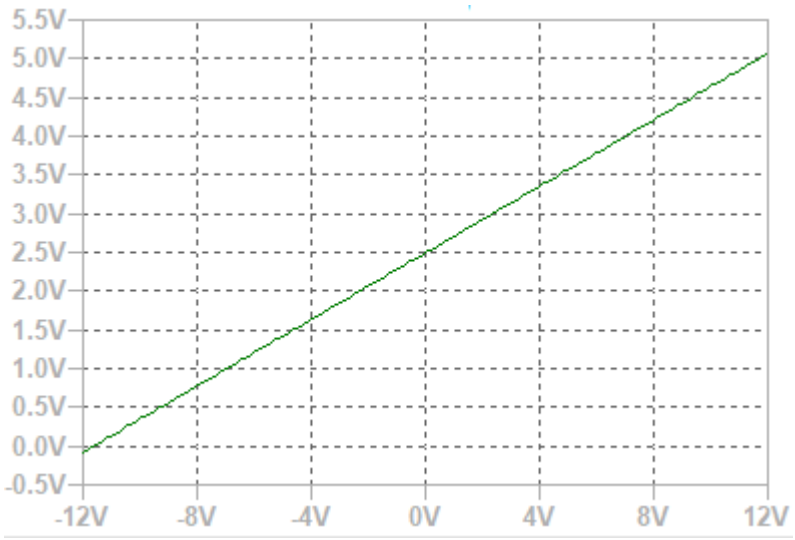


In the voltage scaling circuit (above), resistor R1 forms a voltage divider with the parallel combination of resistors R2 and R3: Vout = [R2//R3 / (R1 + R2//R3)] * V1, R2//R3 = 27.33K and Vout = 0.215 * V1. This ratio is good for an input range of 23.26V or ±11.63V. It's not ±12V because available 5% tolerance resistors values were chosen. Note: Some gain adjustment may be needed to reduce the waveform generator output level to ±11.6V or clipping of input data will occur.

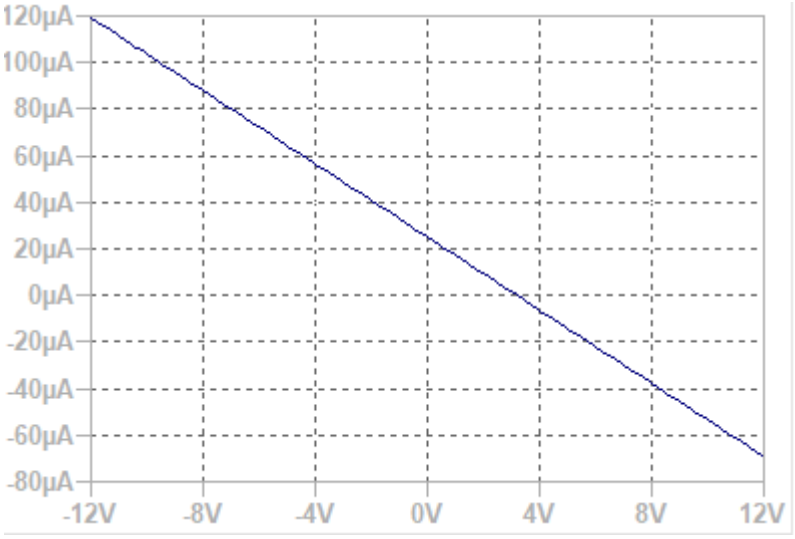
To obtain a midscale zero, Vout must be 2.5V with V1=0V. Grounding the input resistor R1 forms a voltage divider. The equation for the divider circuit is: Vout = [R1//R3 / (R2 + R1//R3)] * 5V, R1//R3 = 42.86K and Vout = 0.499 * 5V = 2.496V. Resistor R2 must equal the parallel combination of R1 and R3 to divide 5V by two.

You can scale the input to a different range by using the following process:

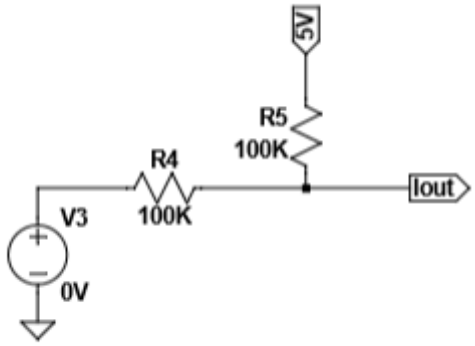
Pick R1, for example R1 = 100K
Pick a full scale value, e.g. fs = 30V (this will be plus and minus 15V)
Let R3 = R1/(fs/10 – 1) = 100K/(30/10 -1) = 50K
Let R2 = 1 / (1/R1 + 1/R3) = 1 / (1/100K + 1/50K) = 33.33K



This input vs output curve of the voltage scaling circuit shows the ±11.63V input converted to the needed zero to 5V range.



The voltage scaling circuit uses 5V to pull the zero point up. This results in a current being injected into the voltage node under test. It does not affect the measured voltage in a curve tracer circuit but it will add current to a device under test. At zero volts, over 20uA will be injected into the test device. Fortunately, the current is linear to the measured voltage and it can be mathematically removed from the current reading.



The current scaling circuit is easier to implement. By making R4=R5, the voltage divider has a ratio of 1/2. The mid scale point is 2.5V and the V3 input has a range of 10V. With the 2.5V offset, the scale is ±5V. The current sense voltage is the output of an amplifier which has a very low output resistance. The small injected current from the scaling circuit does not affect the measurement.

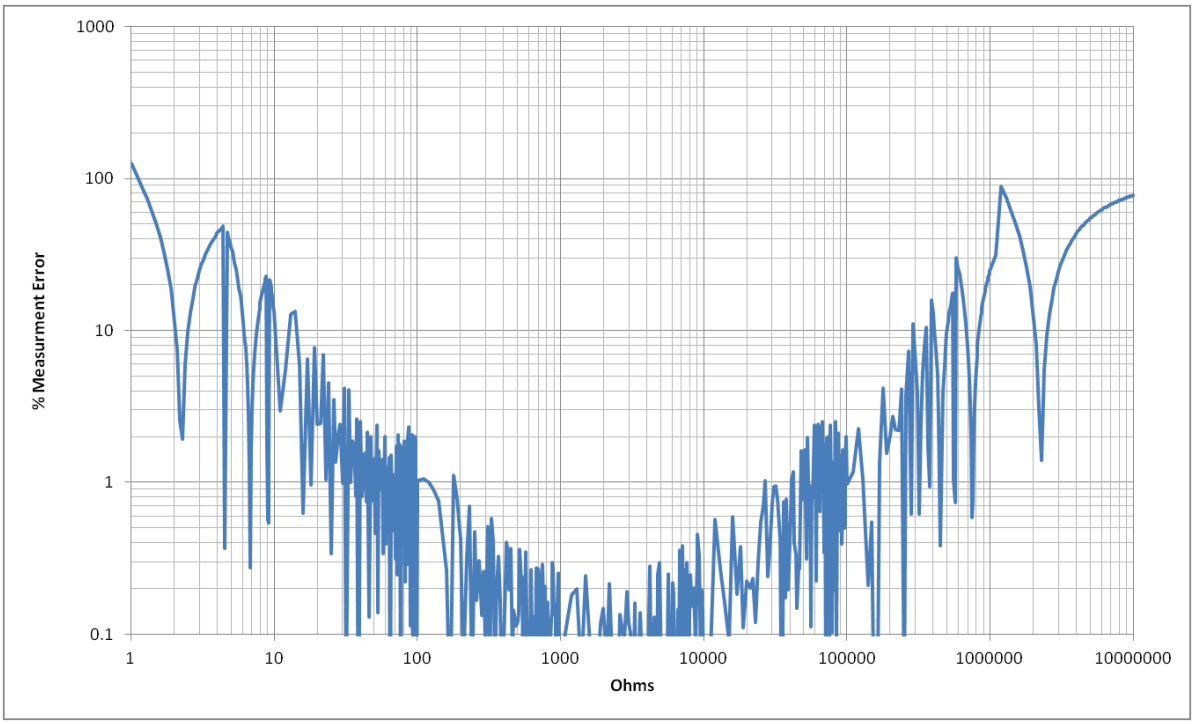
Some Notes on Measurement Accuracy:

The PWM signal generator has 256 values and the analog input 1024 values. With a voltage output of ±11.5V, the signal generator is limited to steps of 23V/256 or 89.9mV per step. The voltage sense circuit has a resolution of 23.26V/1024 or 22.7mV. The current sense circuit has a resolution of 10mA/1024 or 9.8uA.

So how do all these values work together? When the device is high impedance, generator steps dominate the voltage measurement. When the device is low impedance, the analog input dominates the voltage measurement. For example: The reverse voltage of a diode will jump in about 90mV increments while the forward diode voltage may change in millivolt increments, the curve tracer can only detect about 23mV changes.

Current is the opposite. Low impedances cause current changes dependent on the series limit resistor (2.2K ohms) and the generator steps. The steps will be around 90mV/2.2K or 41uA. For high impedances, the current measurement circuit dominates with 10uA increments.

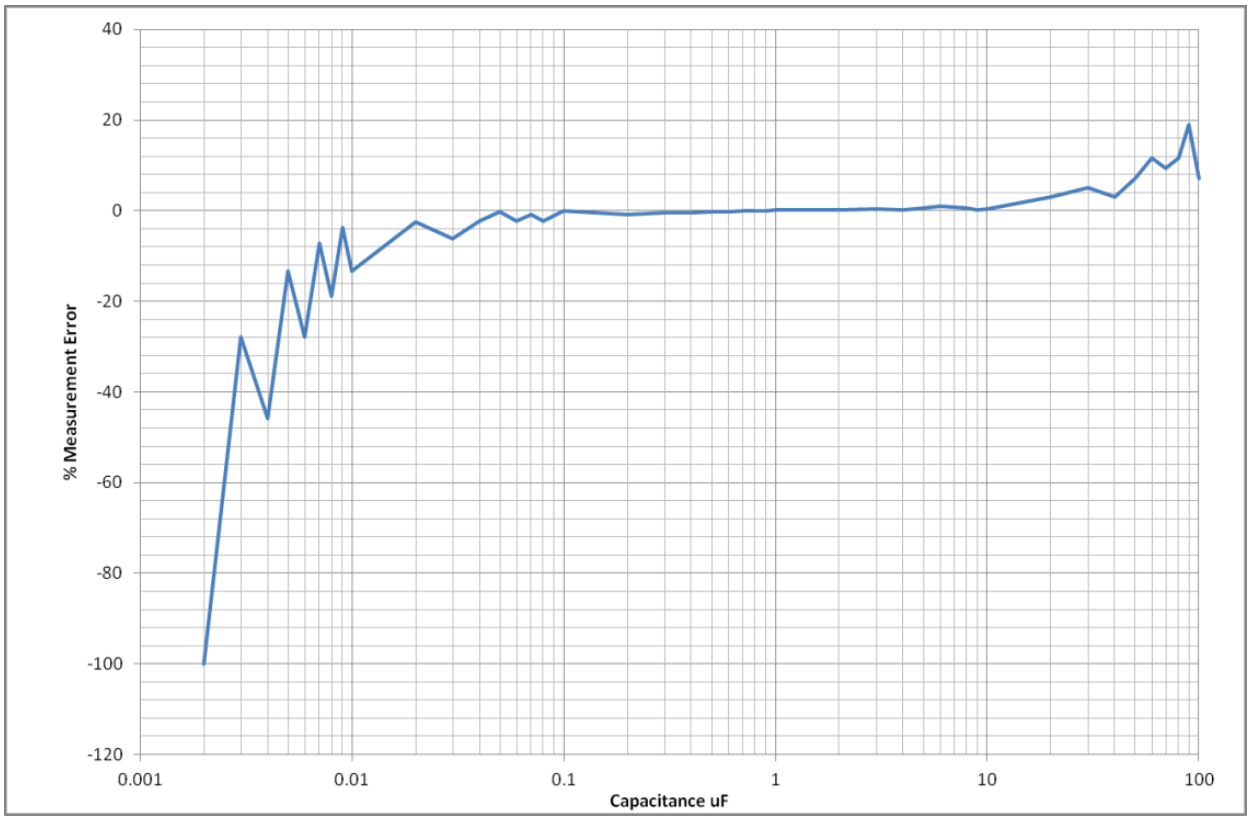
A curve tracer displays conductance curves or slopes of I/V. The current and voltage changes along a curve can be used to measure resistance. (V2-V1)/(I2-I1) = R. Quantization by the 10-bit ADC (1024 values) limits the accuracy of resistance measurement. Our system uses fixed scales, so we are stuck with the limitation of the design. The graph below shows the resistance measurement error caused by quantization. Keep in mind there are additional errors from non-linearities in the ADC.



The graphs shows the curve tracer performs well for resistances of 100 ohms to 40K ohms. It is useful for 10 ohms to about 400K ohms with a 10% error.

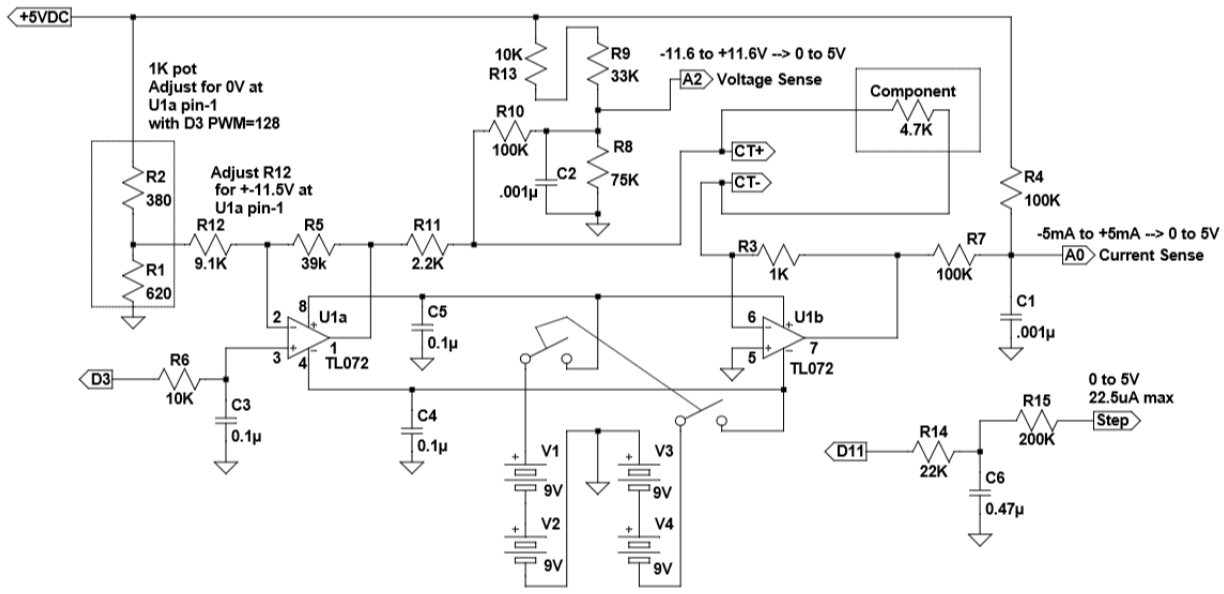
Capacitance can be measured using a sine wave excitation voltage. C = Ipp/(Vpp * 2 * pi * f)

Quantization by the 10-bit ADC again limits the range of capacitance. The graph below shows capacitance measurement errors caused by quantization.



The curve tracer performs well for values of 0.02uF to 50uF. When measuring capacitors larger than 10uF, this technique may result in the peak current exceeding 5mA. The voltage amplitude should be reduced or the frequency increased to prevent clipping by the current measurement circuit.

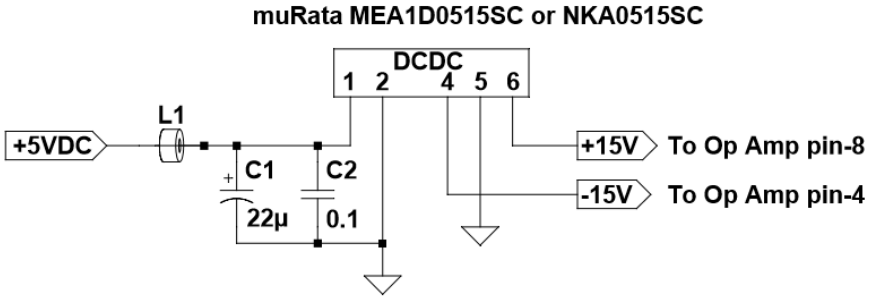
The Curve Tracer Schematic:



A Fritzing file is provided should you want to make some changes to the circuit.

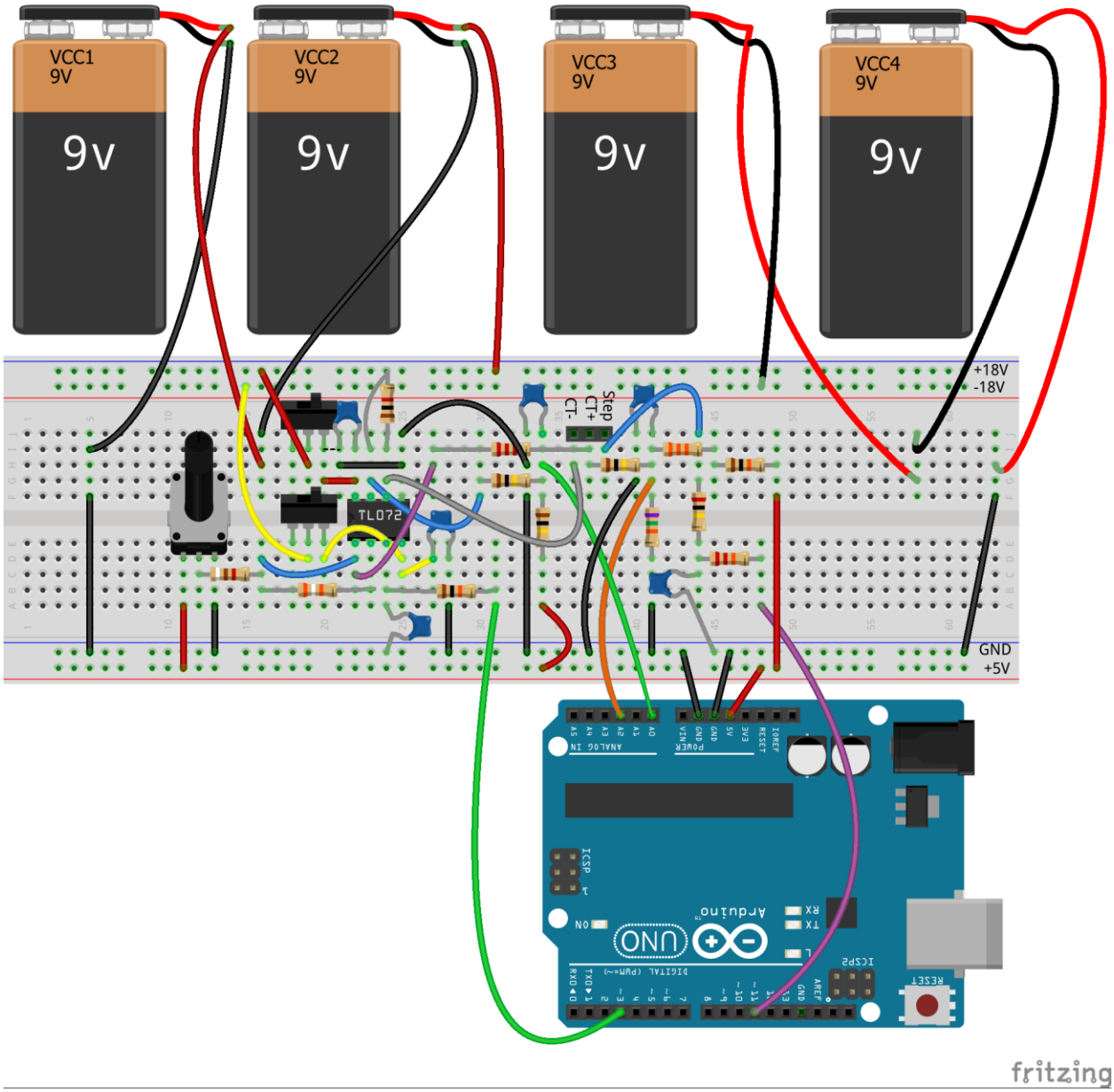
Resistors R1 and R2 represent a potentiometer set to the above value. Use a 1K potentiometer with the end terminations connected to 5V and ground and the center tap to R12. The value of R12 might need adjusting to keep the waveform within the measurement range. To see if the sine wave is clipped, use the X-t viewing mode in the uno_pwm_ct.exe application.

Replacing the 9V batteries with a 5VDC to ±15VDC converter is a modification that simplifies operation (see schematic below).



Inductor L1 is a ferrite bead. The DC to DC converter is powered by the Uno's +5VDC. The power switches are not needed if you use a converter.

How to Breadboard it:



The Arduino Sketch:

```
//simple-circuit 12-8-2019

//A Component Curve Tracer
//Uses a general purpose serial port driven I/O control program for UNO boards
//baud = 115200 with one stop bit, 8-data bits and no parity
//Works with uno_pwm_ct.exe application

//Serial Commads:

 // @adcNNNN Read ADC pin
 //   Analog pins are NNNN = 0 for A0 to NNNN = 5 for A5.
 //   Returns ad?MMMM\n where ? = A,B,C,D,E,F for channels A0 to A5
 //   and NNNN = 0000 to 1023,  V = 5*NNNN/1024

 // @meaNNNN Generates a minus to plus ramp voltage then returns to zero
 //   Reads Voltage and Current for 256 samples.
 //   Returns meaNNNNMMMM\n where NNNN and MMMM are V then I
 //   V = (NNNN-512)*0.02461 Volts, I = -(NNNN-512)*0.00977 mA

 // @posNNNN Generates a zero to plus ramp voltage, has same return as mea command
 // @negNNNN Generates a zero to minus ramp voltage, has same return as mea command

 // @cctNNNN Generates a continous sine voltage
 //   Reads Voltage and Current for 256 samples.
 //   Returns cctNNNNMMMM\n where NNNN and MMMM are V then I 0000 to 1023

 // @frqNNNN Sets Sine Oscillator frequency constant 4 through 50
 //   Returns nothing
 //   f = 963.234/NNNN

 // @magNNNN Sets Sine Oscillator magnitude constant 25 through 127
 //   Returns nothing
 //   Vpeak is approximately 12.5*NNNN/127

 // @dacNNNN Set PWM value NNNN = 0 to 255 on pin D11
 //   Returns dac\n
 //   PWM Voltage = 5*NNNN/256

//Interface program must sent serial string then parse returned character string for
//the returned value. Returned string is up to 13 characters including the
//cariage return and linefeed. The returned MMMM value begins after the echoed command string.

#include "TimerOne.h"

// global variables

  int c_count = 0;            //command buffer counter
  char cmd_buf[10];           //serial port input buffer to hold command string
  volatile int cmd_val = 0;   //integer value from command string
  volatile int v0[256],i0[256];  //analog read values
  volatile int a,b;             //oscillator variables
  volatile int oscON, idac;     //oscillator on/off flag
  int fs = 16; //oscillator frequency constant 15 == 60Hz
  int mag = 127*256; //oscillator magnitude 0 to 127
  int analogmux;

//setup initiallizes the i/o pin directions, the serial interface and the ADC
void setup() {
  int j;
```

```
  pinMode(3, OUTPUT);   //pwm waveform output
  pinMode(11, OUTPUT);  //step generator output
  analogWrite(3,128);  //set the output to mid scale or 0V
  analogWrite(11,0);  //set the output to 0V
  TCCR2B =TCCR2B & B11111000 | B00000001;  //set the PWM frequency to 31Khz
  _SFR_BYTE(ADCSRA) &= 0xFE; //double the ADC clock frequency (~20KHz sample rate)
  Serial.begin(115200);
  a=0;
  b=mag;
  oscON = 0;
  idac = 0;
  Timer1.initialize(165);         // initialize timer1, and set a 1/2 second period
  Timer1.attachInterrupt(callback);  // attaches callback() as a timer overflow interrupt

}

//main read, display and control loop 5Hz
void loop() {
 char c;

//This section scans the serial input string for a command string
//The three letter command is saved in the global variable
//cmd_buf[] as a null terminated string
 while (Serial.available() != 0) {  //if a character is available, process it
  c = Serial.read();                //get one character
  if (c == '@') {
    c_count = 0;                    //reset to command start
    cmd_val = 0;
    cmd_buf[0] = 0;
    break;
  }
  cmd_buf[c_count] = c;             //put the character in the command array
  c_count++;
  if ((c == 13) || (c == 10)){      //if end of line characters go
    fndCmd();                       //look for a command string
                                    //and process it
    c_count = 0;                    //reset the command string buffer
    cmd_val = 0;
    cmd_buf[0] = 0;
    break;
  }
  if (c_count <= 3) {               //after reading three command characters
   if (c_count == 3) {              //read up to four integer characters
    cmd_buf[c_count] = 0;           //initiallize command integer to zero
    cmd_val = 0;
    c_count++;
   }
  }
 //This section extracts 4-digit integer from input string
 //The value is available in global variable cmd_val
  if (c_count > 4){
     cmd_val = cmd_val * 10 + int(c-'0');
    if (c_count >= 8) {                 //only read 3 char then 4 integers
     fndCmd();
     c_count = 0;
     cmd_val = 0;
     cmd_buf[0] = 0;
    }
  }
 }
}
```

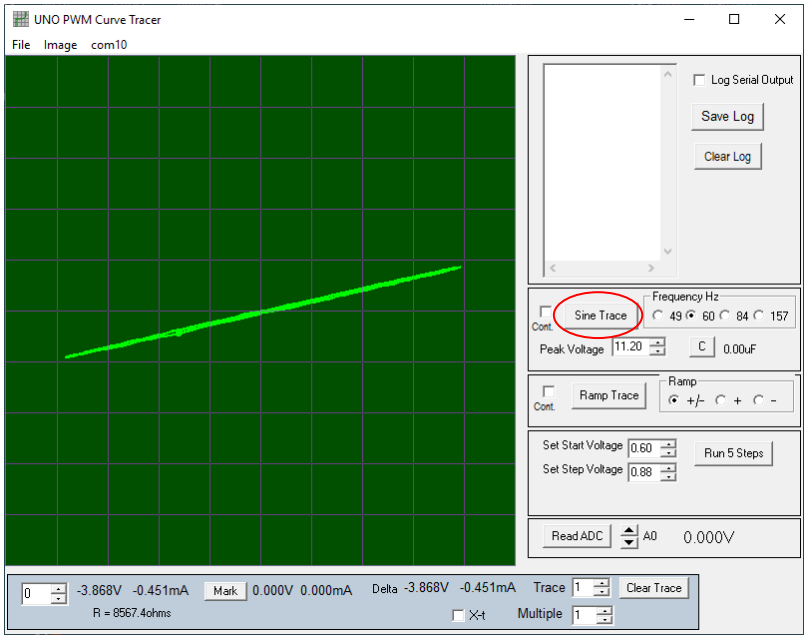
```
//routine to search input string for measure command
//other 3-character commands with integer input can be added
//for additional functions
void fndCmd() {
  int i;

   //process dac which is dac command followed by 0000 to 4095 12-bit value
   if ((cmd_buf[0] == 'd') && (cmd_buf[1] == 'a') && (cmd_buf[2] == 'c')) {
    if (cmd_val > 256) cmd_val = 256;     //dac limits 0-4095
    analogWrite(11,cmd_val);
    Serial.println("dac");
   }

  //Process ADC measure for ADC channel A0 through A5
  //Output is integer 0000 to 1023 (0 to 5V)
  if ((cmd_buf[0] == 'a') && (cmd_buf[1] == 'd') && (cmd_buf[2] == 'c')) {
     if (cmd_val >5) cmd_val = 5;     //adc ports are 0 to 11
     Serial.print("ad");
     Serial.print(char(cmd_val + 65));   //return A-F for channels 0-5
     if (cmd_val == 0) analogmux = A0;
     if (cmd_val == 1) analogmux = A1;
     if (cmd_val == 2) analogmux = A2;
     if (cmd_val == 3) analogmux = A3;
     if (cmd_val == 4) analogmux = A4;
     if (cmd_val == 5) analogmux = A5;
     v0[0] = analogRead(analogmux);
     inttostr(v0[0]);
     Serial.println();
   }

 //Process component curve trace with negative to positive ramp
 //Output is integer 0000 to 1023 (-12.6 to 12.6V) and 0000 to 1023 (-5 to 5mA)
  if ((cmd_buf[0] == 'm') && (cmd_buf[1] == 'e') && (cmd_buf[2] == 'a')) {
      oscON = 0;
      analogWrite(3,0);
      delay(100);
      analogRead(0);
      for (i=0;i<=255;i++){
       analogWrite(3,i);
       delay(1);
       v0[i] = analogRead(A2);
       i0[i] = analogRead(A0);
      }
      analogWrite(3,128);
      for (i=0;i<=255;i++){
       Serial.write("mea");
       inttostr(v0[i]);
       inttostr(i0[i]);
       Serial.println();
      }
   }

  //Process component curve trace with continous Sine wave
  //Output is integer 0000 to 1023 (-12.6 to 12.6V) and 0000 to 1023 (-5 to 5mA)
  if ((cmd_buf[0] == 'c') && (cmd_buf[1] == 'c') && (cmd_buf[2] == 't')) {
      if (oscON == 0){
        oscON = 1;
       delay(100);
      }
      while (b < 0);     //sync on waveform generator
      while (b >= 0);
      for (i=0;i<=255;i++){    //get 256 readings
```

```
      i0[i] = analogRead(A0);
      v0[i] = analogRead(A2);
     }

    for (i=0;i<=255;i++){    //send 256 data pairs
     Serial.print("cct");
     inttostr(v0[i]);
     inttostr(i0[i]);
     Serial.println();
     }

  }

  //process oscillator on/off
  if ((cmd_buf[0] == 'o') && (cmd_buf[1] == 's') && (cmd_buf[2] == 'c')) {
  i = 0;
  oscON = cmd_val;
  }

//Process component curve trace with zero to positive ramp
//Output is integer 0000 to 1023 (-12.6 to 12.6V) and 0000 to 1023 (-5 to 5mA)
 if ((cmd_buf[0] == 'p') && (cmd_buf[1] == 'o') && (cmd_buf[2] == 's')) {
     oscON = 0;
     analogWrite(3,128);
     delay(100);
     analogRead(0);
     for (i=0;i<=255;i++){
      analogWrite(3,i/2 + 128);
      delay(1);
      v0[i] = analogRead(A2);
      i0[i] = analogRead(A0);
     }
     analogWrite(3,128);
     for (i=0;i<=255;i++){
      Serial.write("mea");
      inttostr(v0[i]);
      inttostr(i0[i]);
      Serial.println();
     }
 }

//Process component curve trace with zero to negative ramp
//Output is integer 0000 to 1023 (-12.6 to 12.6V) and 0000 to 1023 (-5 to 5mA)
 if ((cmd_buf[0] == 'n') && (cmd_buf[1] == 'e') && (cmd_buf[2] == 'g')) {
     oscON = 0;
     analogWrite(3,128);
     delay(100);
     analogRead(0);
     for (i=0;i<=255;i++){
      analogWrite(3,128 - i/2);
      delay(1);
      v0[i] = analogRead(A2);
      i0[i] = analogRead(A0);
     }
     analogWrite(3,128);
     for (i=0;i<=255;i++){
      Serial.write("mea");
      inttostr(v0[i]);
      inttostr(i0[i]);
      Serial.println();
     }
 }
```

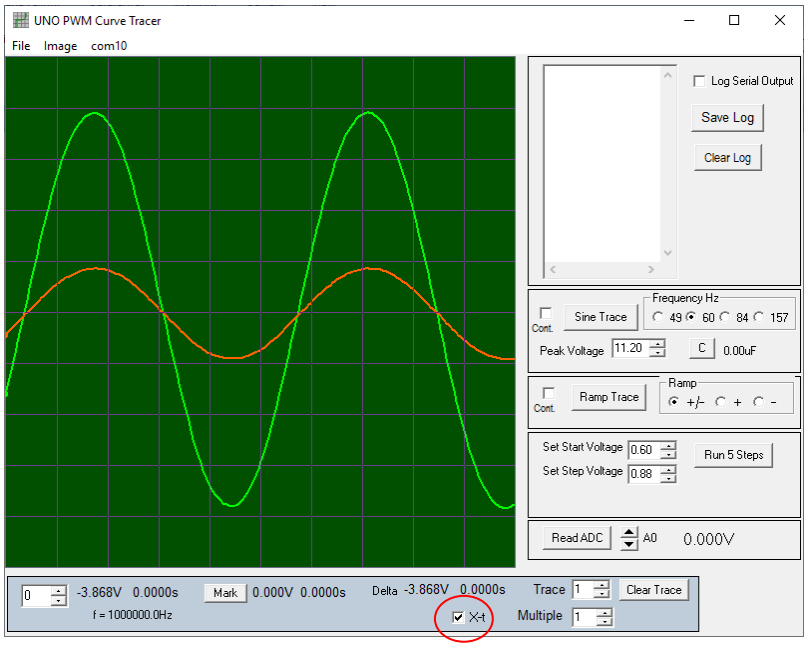
```
   //set oscillator frequency constant 4 to 50
   if ((cmd_buf[0] == 'f') && (cmd_buf[1] == 'r') && (cmd_buf[2] == 'q')) {
    if (cmd_val < 4) cmd_val = 4;
    if (cmd_val > 50) cmd_val = 50;
    oscON = 0;
    a=0;
    b=mag;
    fs = cmd_val;
    oscON = 1;
   }

   //set oscillator magnitude 25 to 127 (about 2.4V peak to 12.5V peak)
   if ((cmd_buf[0] == 'm') && (cmd_buf[1] == 'a') && (cmd_buf[2] == 'g')) {
    if (cmd_val < 25) cmd_val = 25;
    if (cmd_val >127) cmd_val = 127;
    oscON = 0;
    a=0;
    b=cmd_val*256;
    mag = cmd_val*256;
    oscON = 1;
   }
}

//routine to convert integer to four ascii characters 0000-9999
//easier to generate leading zeros for fixed ouput format
void inttostr(int iv) {
   char c[4];
     c[3] = (iv % 10) + '0';
         iv = iv / 10;
         c[2] = (iv % 10) + '0';
         iv = iv / 10;
         c[1] = (iv % 10) + '0';
         iv = iv / 10;
         c[0] = (iv % 10) + '0';
         iv = iv / 10;
     Serial.print(c[0]);
     Serial.print(c[1]);
     Serial.print(c[2]);
     Serial.print(c[3]);
}

void callback()   //sine oscillator interupt routine
{
  if (oscON != 0){
    a = a + b/fs;         //dual integrator oscillator
    b = b - a/fs;
   analogWrite(3,a/256+128); //write the PWM sine value 0-255
  }
}
```

The uno_pwm_ct.exe Application:

To simplify operation, a Windows® application written using Lazarus (FPC compiler), is provided. The application will try to connect to the Uno board when it starts. If not, you need to click on the com port assigned for your Uno.

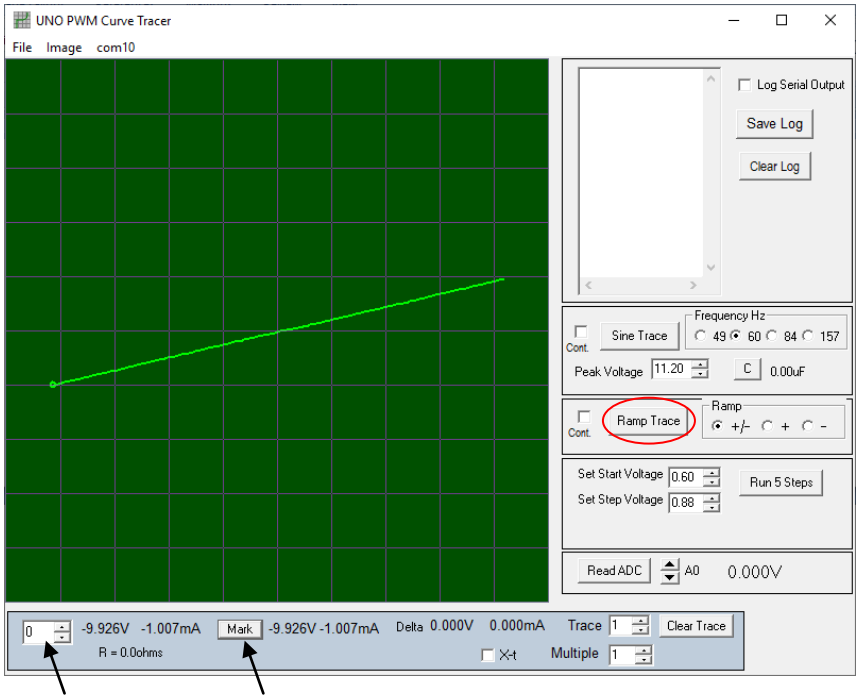


Insert a 10K resistor in the +CT and –CT socket pins. Click on the Sine Trace button to generate a curve.



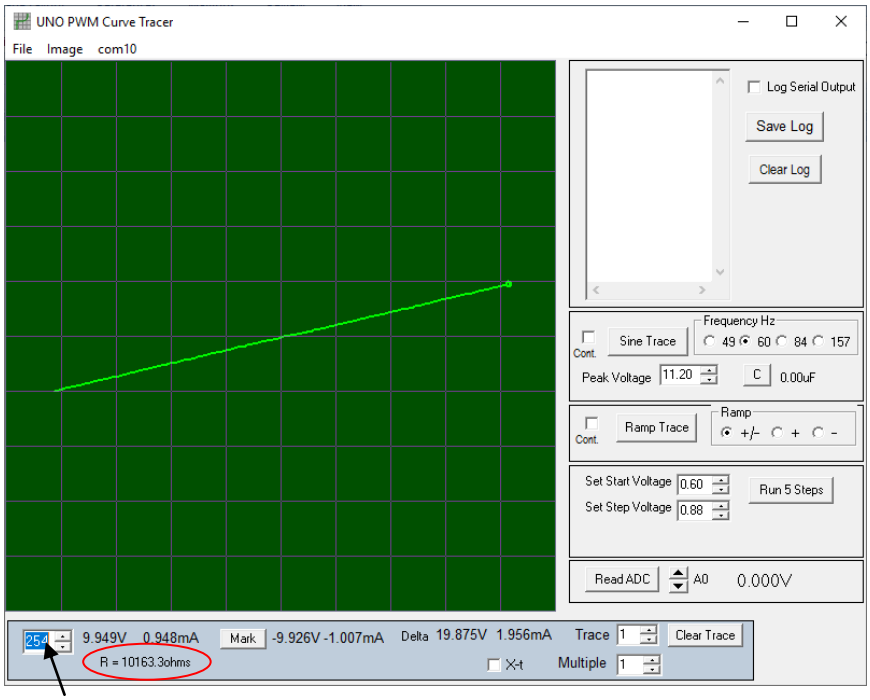
Checking the X-t check box turns on the amplitude versus time display mode.

The voltage is in green and the current in orange. The sample rate is around 8.2KHz and there are 256 samples.



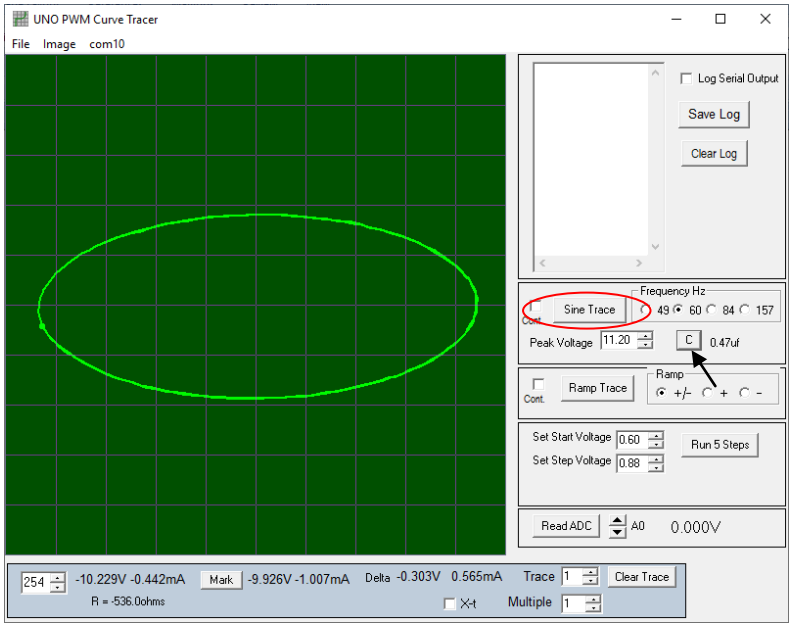
Click on the Ramp Trace button to obtain a higher resolution sweep. The ramp generates all 256 steps in sequence.

Set the cursor index to zero then click on the Mark button. This saves the voltage and current at sample zero.



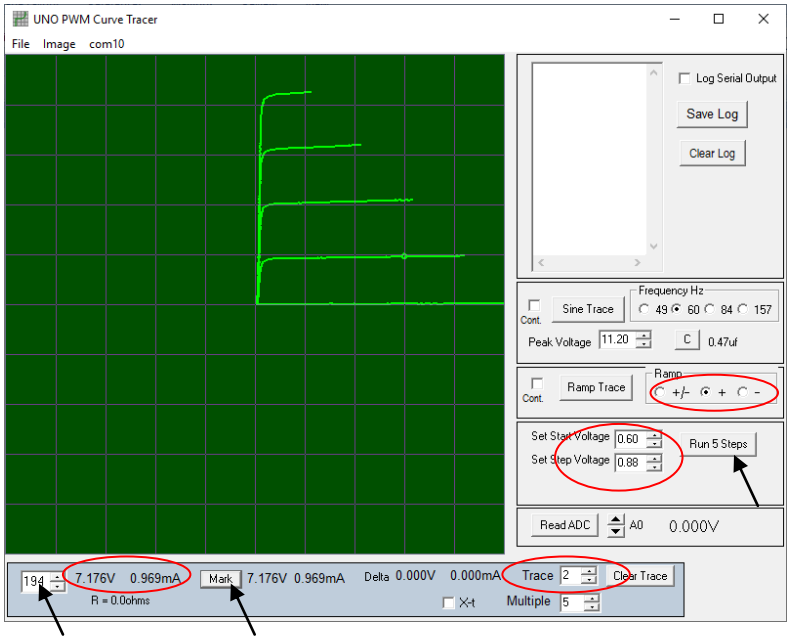
Adjust the cursor index to a high value over 250. The marked point is subtracted from the cursor values and displayed as the Delta values.

Resistance is calculated from the delta values and displayed under the cursor values.



Insert a 0.47uF capacitor in the +CT and –CT socket pins. Click on the Sine Trace button to generate a curve.

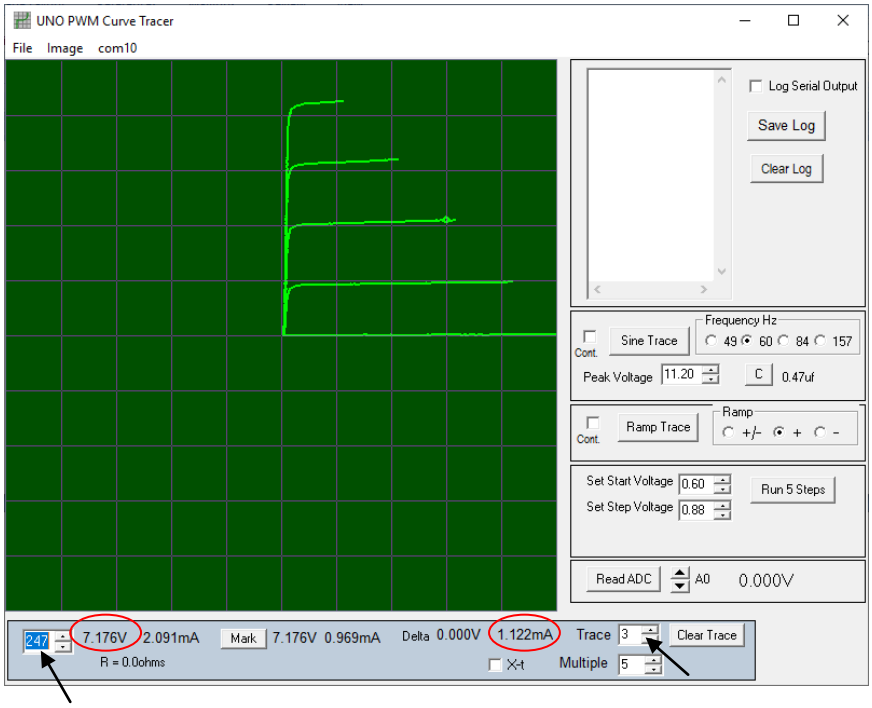
Next, click the C button to calculate capacitance.



Insert a 2n2222A or a 2n2904 NPN transistor. +CT is the Collector, –CT is the Emitter and Step is the Base.

Set Ramp to +, Start Voltage to 0.6, and Step Voltage to 0.88. Click the Run 5 Steps button.

Select Trace number 2. Then adjust the cursor index for a voltage of about 7V. Click the Mark button to record the collector current.

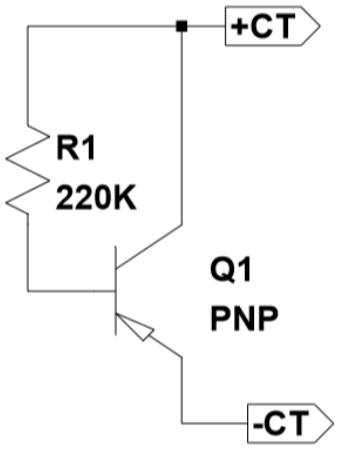


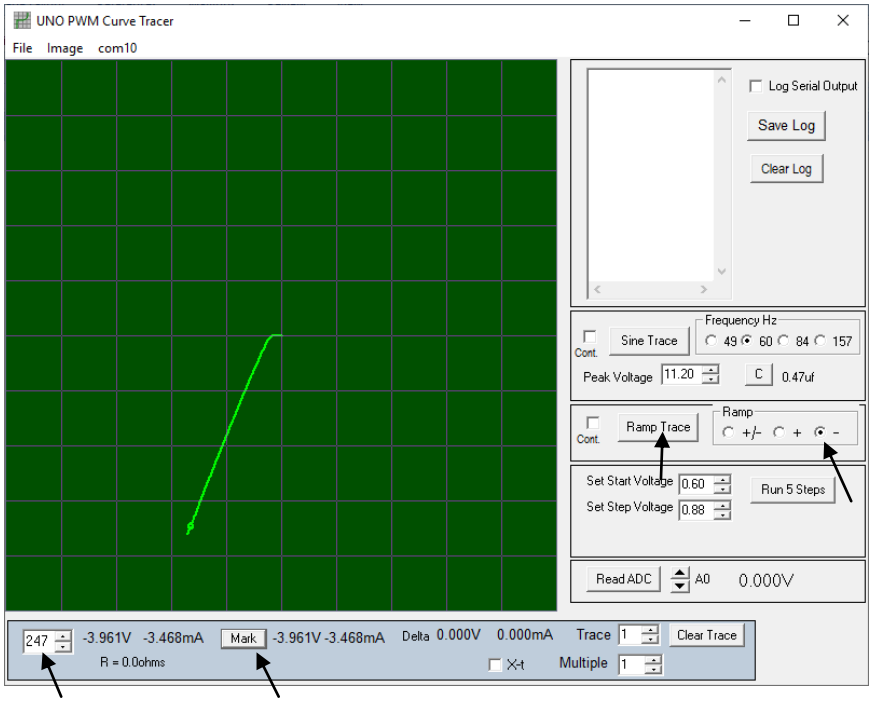
Select Trace number 3. Then adjust the cursor index for a voltage of about 7V.

The Delta collector current is 1.122mA. The base step was 0.88V and the current step is 0.88V/222K or 3.96uA.

Calculate hfe = 1.122mA/3.96uA
hfe = 283.

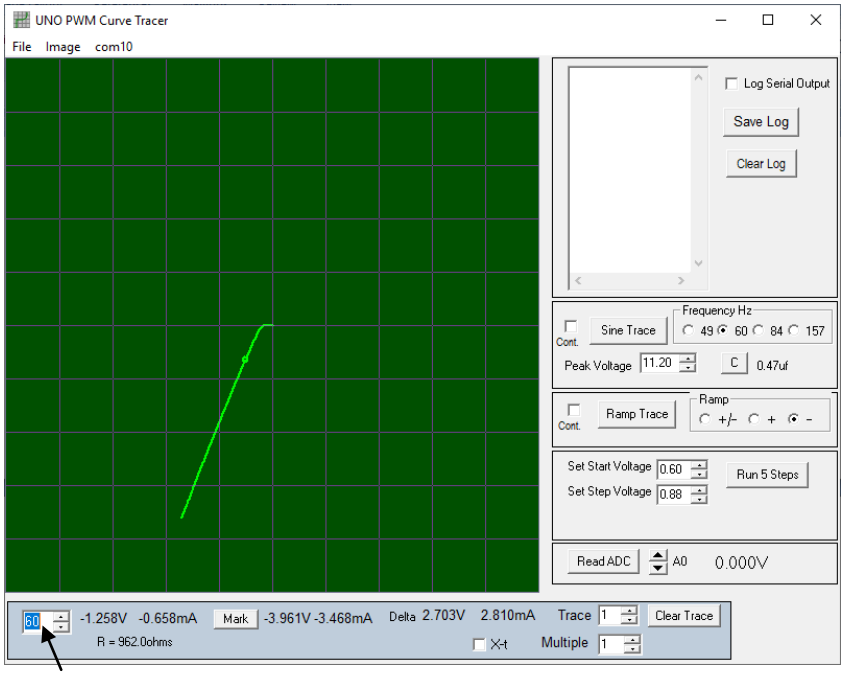
The step generator can only make positive voltages and currents. So it is not possible to generate curves for PNP transistors. However, we can use a resistor amplifier configuration to measure HFE. Using the circuit below, we simply measure the resistance slope and divide it by the collector to base resistor.





Connect the PNP transistor and the 220K base to collector resistor.

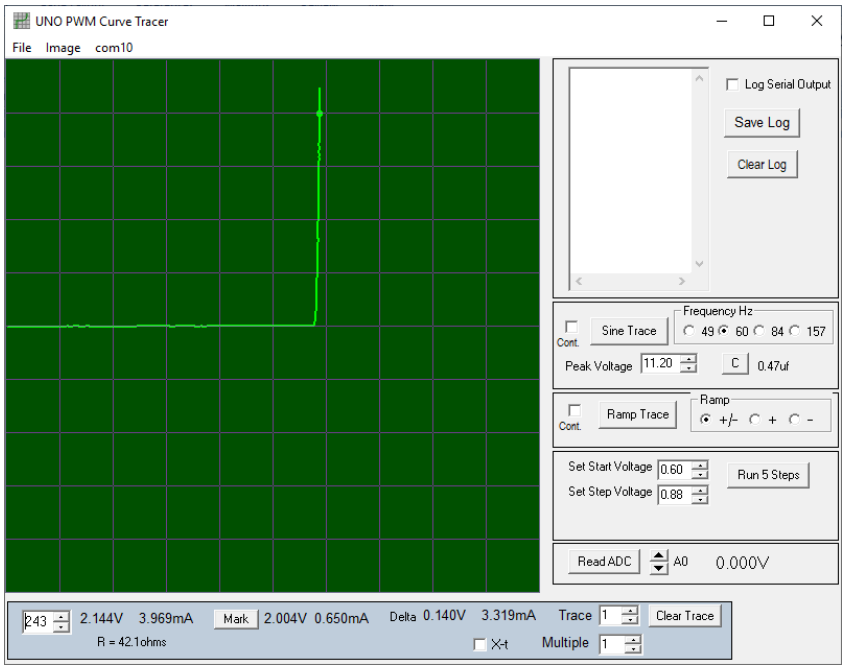
Select Ramp -, click the Ramp Trace button, change the cursor index to a high value and click the Mark button.



Move the cursor to a low value keeping the voltage below the -0.6V knee.

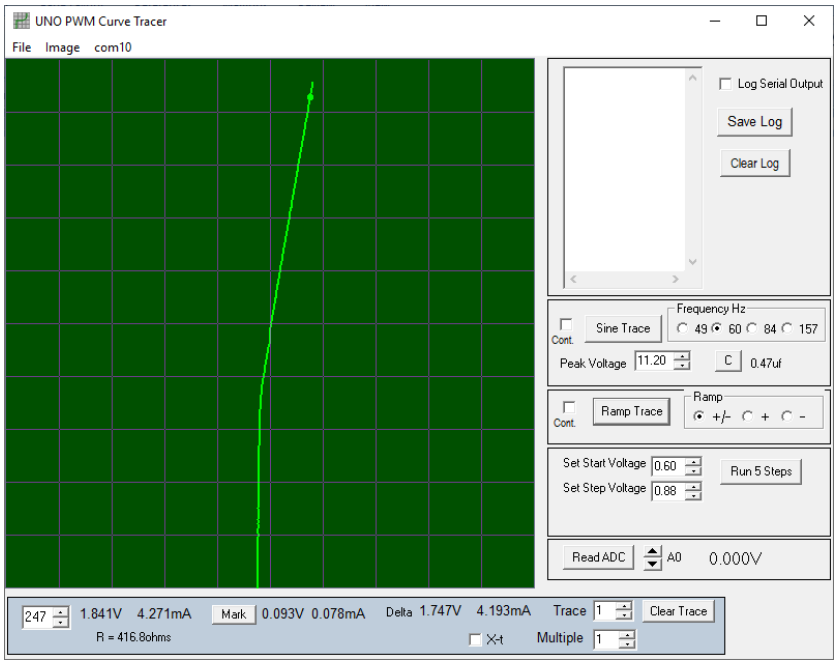
Read the resistance. (962 ohms)

HFE = 222000/962 =229



Insert a Green LED diode with the anode in +CT and cathode in –CT.

Perform a Ramp Trace. Move the cursor to measure the forward voltage at 4mA.



This is a Ramp trace of a DIP reed relay. The curve tracer does not have enough current to activate the contacts. However, the connection polarity can be checked.

The positive voltage direction shows a 416 ohm coil resistance while the negative direction shows the built-in clamp diode.