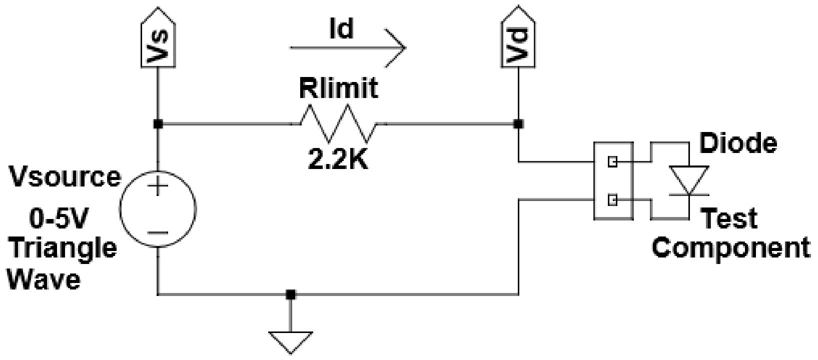
Curve Circuit Tracer Theory:

A curve tracer sweeps the voltage across a component while measuring the current. Voltage is displayed along the x-axis and current along the y-axis. The slope of the curve is I/V or conduction.

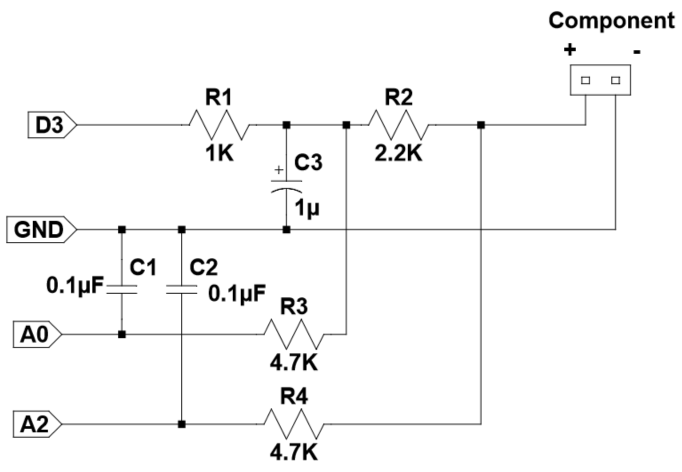
It's not a good idea to apply a voltage directly to a component because high current may flow and damage the component. Current is limited by adding a series resistor to the voltage source. Using Ohm's law, the limit resistor provides a means for calculating the current in the component.



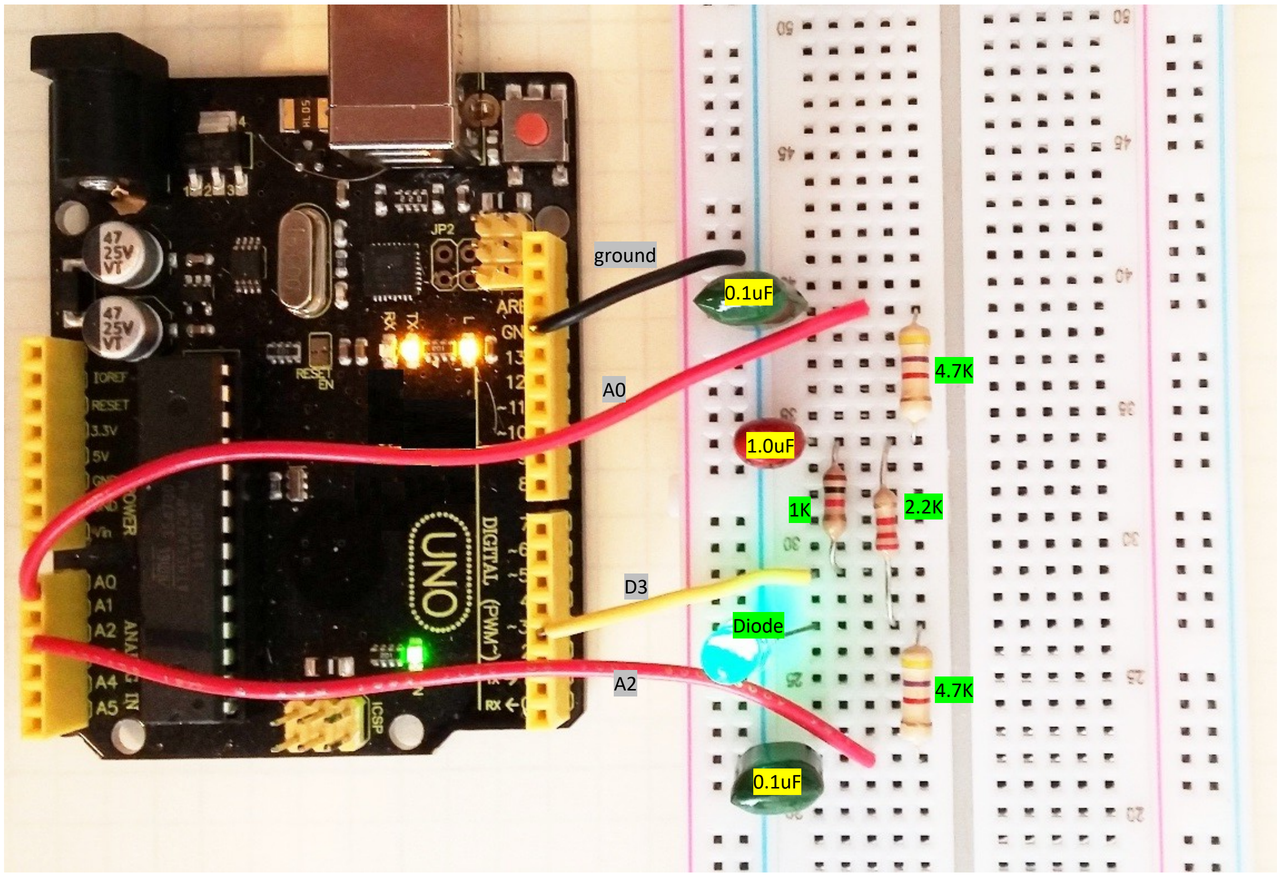
In the above example circuit, the current through the diode is (Vs-Vd)/2.2K and the voltage on the diode is Vd.

This project uses and Arduino Uno board. The voltage source and measurements are limited by the boards specifications. That is, the voltage range is limited at zero to 5V. Negative voltages are not possible without additional circuitry. However, reversing the test component leads in the socket applies a negative voltage to the part.

Our circuit uses pin-3 as the voltage source, A0 to measure Vs and A2 to measure Vd. The full circuit diagram is shown below. For more information on the design theory, refer the next section.



How to Breadboard it:



How the PWM Curve Tracer Works:

The Uno board does not have a built in digital to analog converter. To obtain voltage waveforms, the 0 to 5V digital signal from a pulse width modulated (PWM) output pin must be filtered. For pin-3, the nominal frequency is around 490Hz. The Uno has a built in 10-bit analog to digital converter (ADC). Let's strive for a filtered PWM signal variation that is no more than one ADC bit. That voltage is $5V/2^{10}$ = 5V/1024 or about 5mV. This is 1000 times smaller than the 5V amplitude from the PWM pin.

Simple resistor-capacitor (RC) low pass filters decrease the signal amplitude by 10 for every decade change in frequency. Five volts at 490Hz becomes 0.5V at 49H, then 0.05V at 4.9Hz and finally 5mV at 0.49Hz. We are talking about sine wave and not the square wave that the PWM pin generates. So our filter would have to go a little lower than 0.49Hz.

RC filters also affect the time it takes a signal to rise or fall to a new value. The rise or fall signals are exponential and have a time constant (the time it takes to change to 63% of the final value) that is equal to R times C. As a general rule, we would like to wait five time constants to keep the signal within 1% of actual.

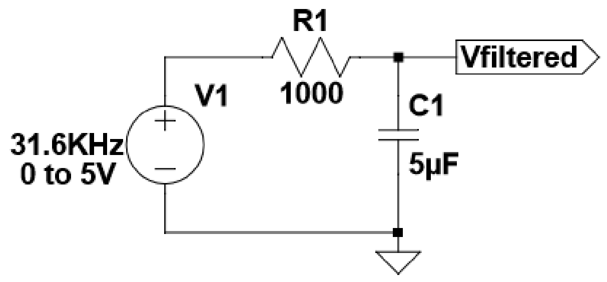
The formula for and RC low pass filter cutoff frequency is $f_{LP} = 1/(2\pi RC)$

If we pick R = 1000Ω and f_c=0.5Hz, then C = 319uF

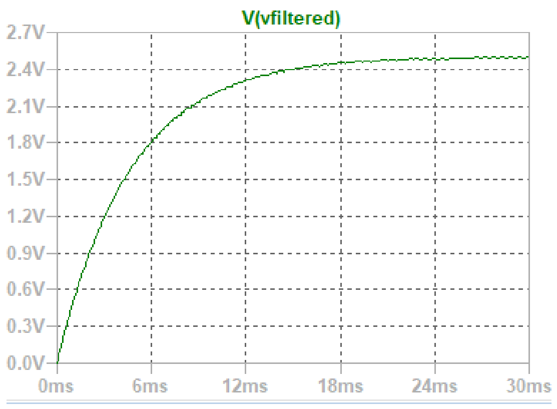
This gives us a time constant t_c = 1000 * 319e-6 = 0.32 seconds

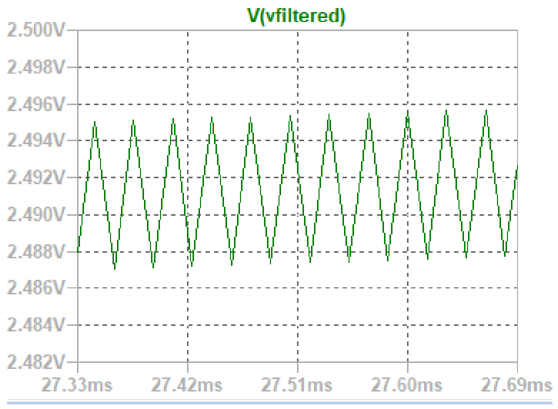
It would be safe to measure the voltage after five time constants or about 1.6 seconds. A waveform signal with hundreds of steps may take minutes to generate.

Fortunately, the Uno PWM frequency can be increased to around 31KHz which is 64 times as high. This means the filter cutoff frequency can be increased 64 times to around 32Hz. With a 1000 ohm filter resistor the capacitor value drops to 5uF. The time constant is 5ms and five times this is 25ms. A 128 step waveform could be generated in 3.2 seconds.



Simulating a 50% duty cycle PWM signal, using LTspice, reveals the rise time of the overall step. Zooming in on the waveform after 27 seconds reveals the average value is around 2.491V (less than 1% error from 2.5V). The ripple is about 8mVpp which is a little more than the 5mV we wanted.

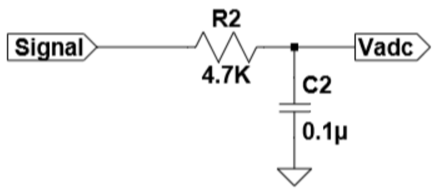




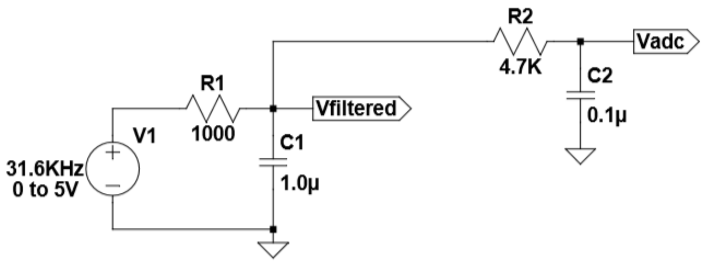
Analog to digital pins on the Uno board have an input range of zero to +5V. If an incorrect voltage is applied to the input it could possible damage the processor. We can't protect the input from every hazard but a simple current limiting resistor can go a long way in protection.

Let's say we charged a large electrolytic capacitor to 5V and accidentally reversed the polarity when it was connected to the analog pin. Negative 5V from a large capacitor has enough energy to damage the input. If the input had a series 4.7K resistor, the current would be limited to about 1mA. This would save the input pin from damage.

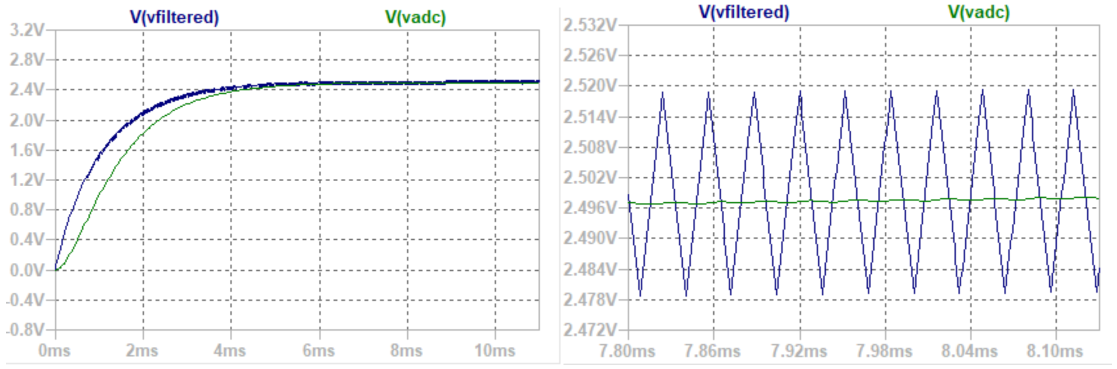
By adding a capacitor, the input protection can also add a second layer of low pass filtering. A common 0.1uF supply filter capacitor would make the cut off frequency 339Hz.

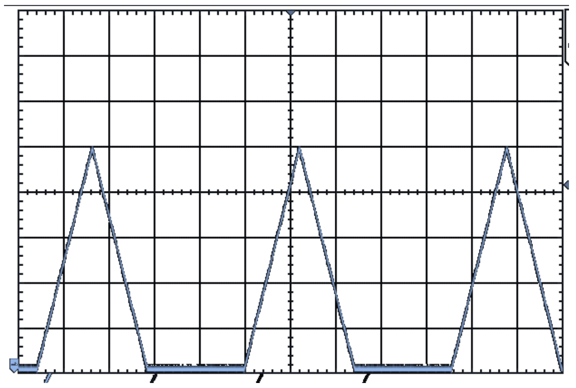


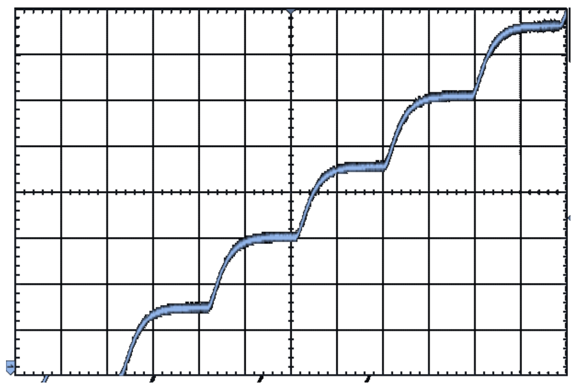
Combining the PWM circuit with the input protection circuit allows us to increase in the PWM cut off frequency by a factor of five. With a higher filter frequency we get a faster rise time and faster measurements.



Simulating a 50% duty cycle PWM signal, using LTspice, reveals the rise time at Vadc (the input to the analog pin) is around 1.5ms. Zooming in on the waveform after 8ms reveals the average value is around 2.497V with almost no ripple voltage. This design will give better performance than the single RC filter.




The waveform generated by the Uno measured at A0.   1V/Div and 0.5s/Div



A zoomed portion of the waveform generated by the Uno measured at A0.   50mV/Div and 5ms/Div
The exponential rise time of each voltage step is visible. The steps are approximately 10ms and 78mV.

<u>Software:</u>

The curve tracer requires a way to plot data in an x-y format. Our program will use a graphing serial terminal program called simple.exe. Place the program on your desktop where it can easily be launched. Information on how to use the serial terminal can be found in the simple_term_use.pdf file.

Rather than cluttering the program with a bunch of code to plot data, simply place the Crt folder into your Arduino Libraries folder (the Arduino folder is usually found in the Documents folder). The #include statement will load the needed subroutines for plotting.

```
#include<Crt.h>
```

Voltages read from the analog input pins are stored in variables x and y. They are defined as global variables.

```
volatile int x, y;
```

The setup section of the code sets digital pin 3 as an output and initializes it in a PWM mode with the analogWrite. The PWM clock divider is set to divide by 64. By setting the first three bits of the TCCR2B register to B001, the clock divider is set to divide by 1.

Serial data is sent at a 115200 bit rate or 11520 characters per second. The serial terminal must be set to the same bit rate (Baud). For x-y plotting, each line drawing operation requires 8 characters. So it takes 8/11520 = 0.69ms to transmit each line command.

```
void setup() {
 pinMode(3, OUTPUT);  //pin-3 is the PWM output 0 to 5V
 analogWrite(3,128);
 TCCR2B =TCCR2B & B11111000 | B00000001;  //set the clock divider to 1
 Serial.begin(115200);
}
```

When the main loop is entered, the serial terminal is cleared, a 10x10 grid is drawn and the scales are printed below the graph.

```
void loop() {
  int i;
  crtClear();
  delay(10);
  crtGrid(9,9);
  crtCursor(1,25);
  Serial.print("X = 0.5V/div   Y = 0.15mA/div");
  delay(10);
```

The positive ramp is generated by sending 64 steps that increment by 4. The PWM is scaled at 0 to 256 so each step = 5V*4/256 = 78.1mV. After each step settles (the delay of 30ms), analog input A0 (the voltage source) is read, then analog input A2 (the component voltage). Analog A0 is read again and averaged (added and divided by 2).

When a new display is generated, a starting point is plotted first. This is when the index counter I is zero. The if statement uses a Moveto call for the starting point. The x-axis or voltage direction is scaled to 720 pixels. The analog to digital converter is 10 bits or 1024 values so the scale factor is 720/1024 = 0.70313.

The y-axis or current direction has a pixel height of 400. To obtain current, the A2 value is subtracted from the A0 value then divided by the limit resistance of 2200 ohms. Since the filter has a 1K series resistor it drops the maximum voltage across the 2200 ohm resistor by about 2/3. To scale for 1.5mA full scale, we use 1.5mA * 2200 ohms = 3.3V.   1024*3.3V/5V = 676 ADC counts per 400 pixels. The scale factor is 400/676 = 0.592.

Each plot axis zero is offset by 1 as not to draw over the grid line. The y-axis starts with a zero at the top. Thus, the pixel value is subtracted from 399 to move zero one pixel from the bottom.

```
//ramp the PWM value 0 to 255 in steps of 4
  for (i = 0; i<255; i= i+4){
   analogWrite(3,i);          //wite the PWM value
   delay(30);
   y = analogRead(A0);
   x = analogRead(A2);
   y = (y+analogRead(A0))>>1;
   if (i == 0)  crtMoveto((int)(x*0.70313+1),(int)(399-(y-x)*0.5919),'R');
   else crtLineto((int)(x*0.70313+1),(int)(399-(y-x)*0.5919),'R');
  }
```

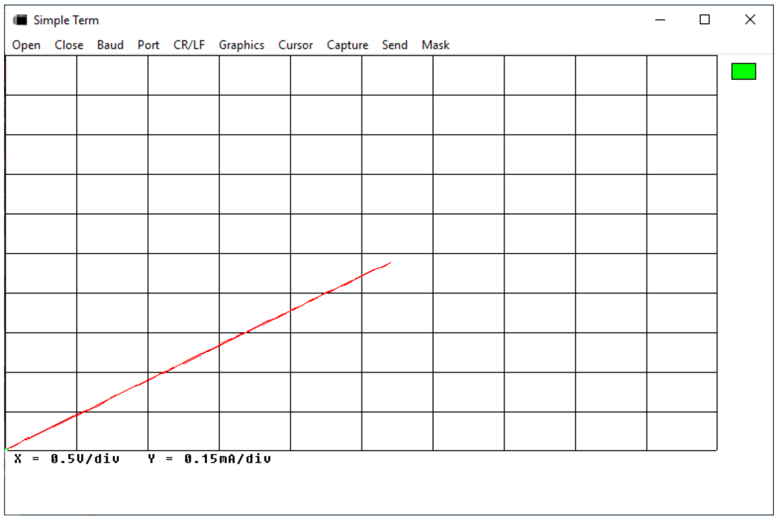
The ramp down operation is identical to the ramp up except: The starting point does not require plotting and the index is decremented by 4.

```
//ramp the PWM value from 254 to zero in steps of 4
  for (i = 254; i>=0;i = i-4){
   analogWrite(3,i);
   delay(30);
   y = analogRead(A0);
   x = analogRead(A2);
   y = (y+analogRead(A0))>>1;
   crtLineto((int)(x*0.70313+1),(int)(399-(y-x)*0.5919),'R');
  }
```

After plotting the curve trace, the program displays it for 0.5 seconds before erasing and starting over. This value should be adjusted where you like it. Also, the 30ms delays in the ramp up and ramp down loops are the settling times after the PWM step changes. These values can be adjusted to your circuit. Values of 9ms give a step time of around 9.7ms or 128*9.7ms = 1.24 seconds for ramp up and down.

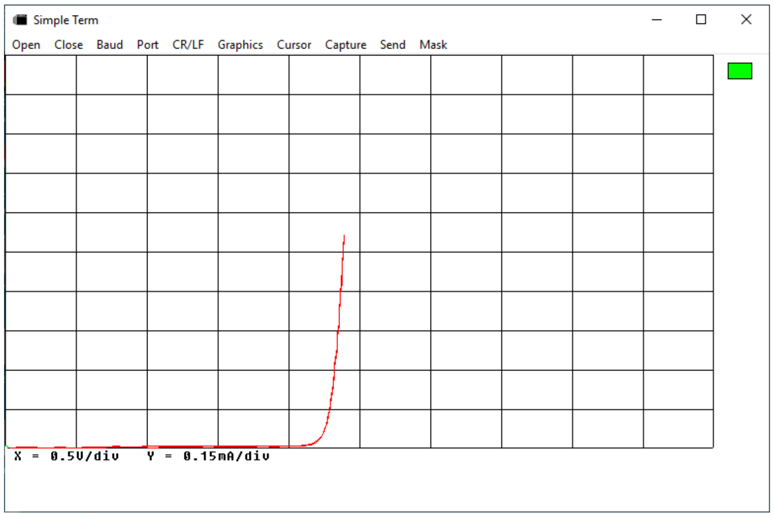
```
  delay(500);
}
```

Component Test Examples (with loop delays set at 30):

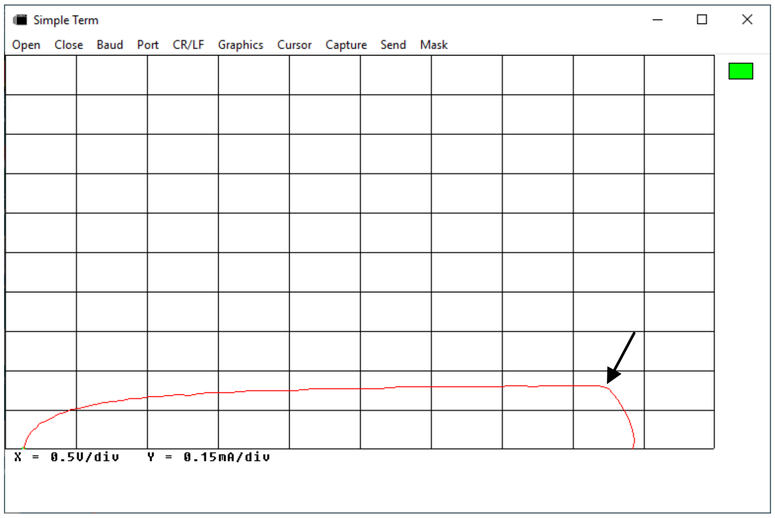

Simple Term
Open Close Baud Port CR/LF Graphics Cursor Capture Send Mask

X = 0.5V/div   Y = 0.15mA/div

A resistor generates a linear curve. The peak voltage is about 2.7V and the peak current is about 0.72mA. The resistor was a 3.9K. 2.7V/0.72mA = 3.75K within the 5% resistor tolerance.


Simple Term
Open Close Baud Port CR/LF Graphics Cursor Capture Send Mask
X = 0.5V/div Y = 0.15mA/div

A green LED diode draws no current until about 2.25V.


Simple Term
Open Close Baud Port CR/LF Graphics Cursor Capture Send Mask
X = 0.5V/div Y = 0.15mA/div

A 100uF capacitor generates a charge and discharge loop. The ramp down produces a negative current that is off the plot screen. Since the voltage applied to the capacitor is approximately a ramp, the capacitance can be calculated from the maximum current on the plot. Formula: I = C*ΔV/Δt. The ramp up voltage is almost exactly two seconds. V at the max current of 0.25mA is about 4.25V. So, C = 0.25mA *2sec / 4.25V = 0.000118F or 118uF.