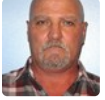




## Arduino DIY Geiger Counter

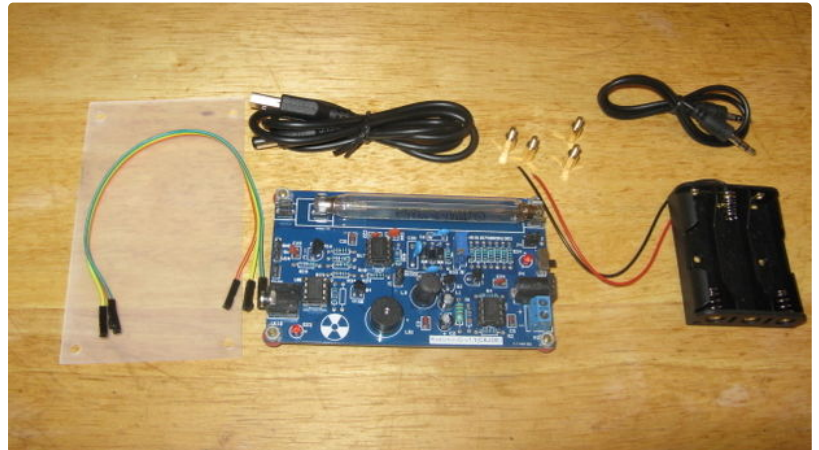
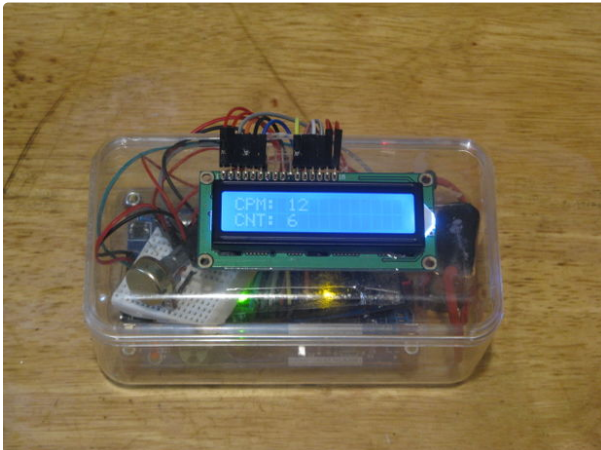


by Joseph Murchison

So you have ordered a DIY Geiger counter and you want to connect it to your Arduino. You go on line and try to duplicate how others have connected their Geiger counter to Arduino only to find something is wrong. Although your Geiger counter seems to work nothing works as described in the DIY you are following when you connect your Geiger counter to your Arduino.

In this Instructable I will be covering how to troubleshoot some of these glitches.

Remember; assemble and code Arduino one step at a time, if you go straight to a finished project and there is a missed wire or line of code it could take you forever to find the problem.



## Step 1: Tools and Parts

Prototype box I used a Ferrero Rocher candy box.

Small breadboard

16x2 LCD

Arduino board ether a UNO or Nano

220 ■ resistor

Pot 10 k■ adjustable resistor.

DIY Geiger Counter Kit

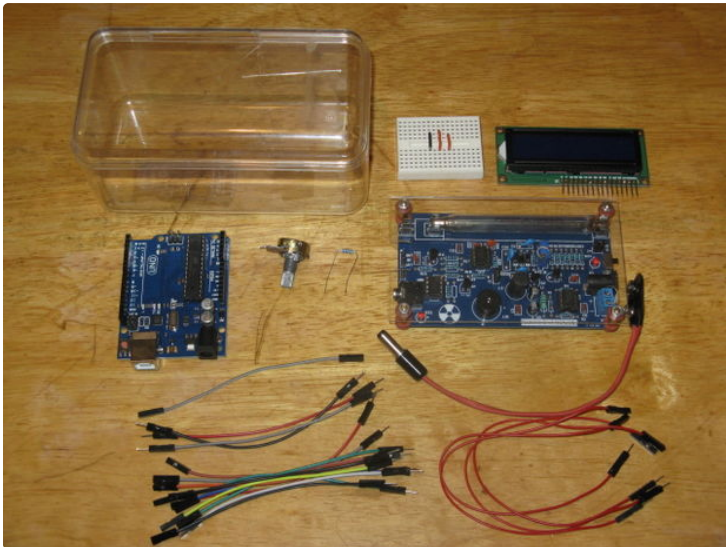
Jumper Wires

Battery connector or harness

Oscilloscope

Fine Nose Pliers

Small Standard Screwdriver



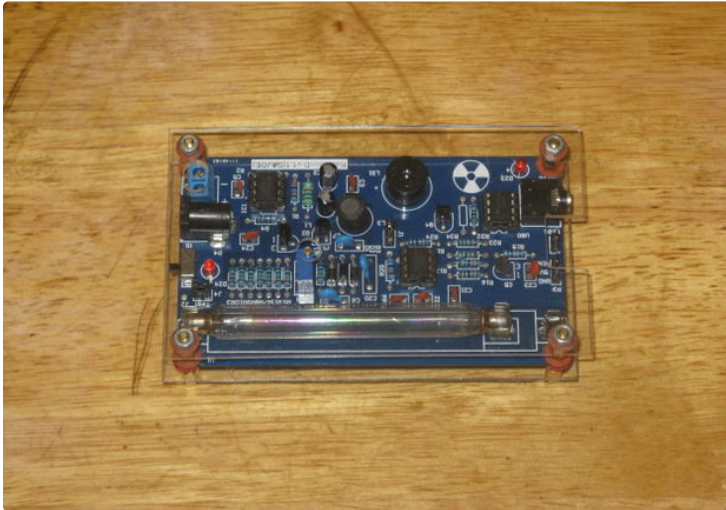
## Step 2: Assemble Your Geiger Counter

Any damage to your Geiger Tube; and your Geiger counter won't work, so use the protective acrylic cover to prevent damage to your Geiger tube.

This Instructable is on how I repaired the same Geiger counter with a broken Geiger tube and fitted

the protective acrylic cover to prevent breakage in the future.

<https://www.instructables.com/id/Repairing-a-DIY-G...>



---

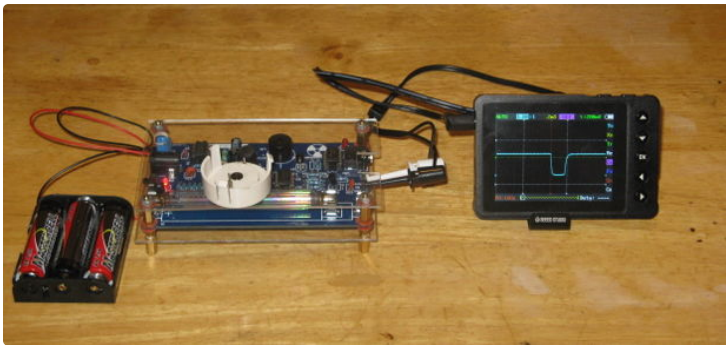
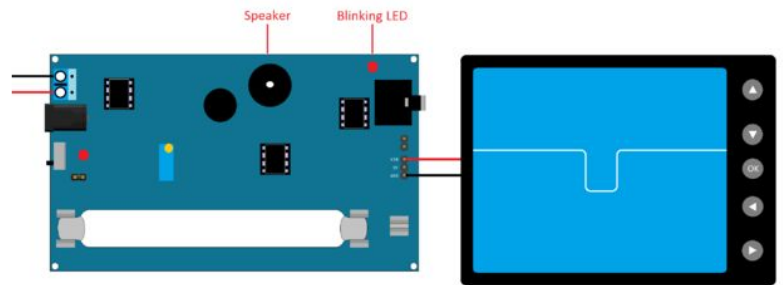
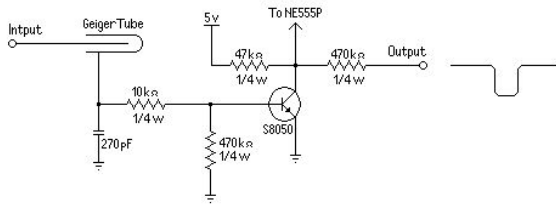
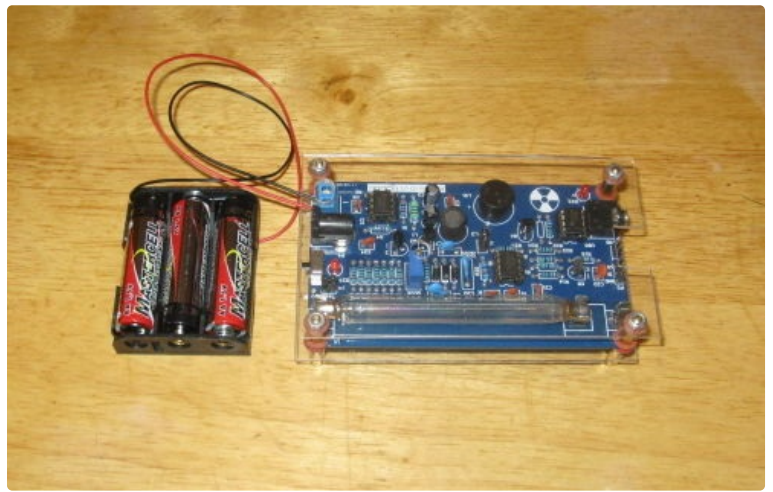
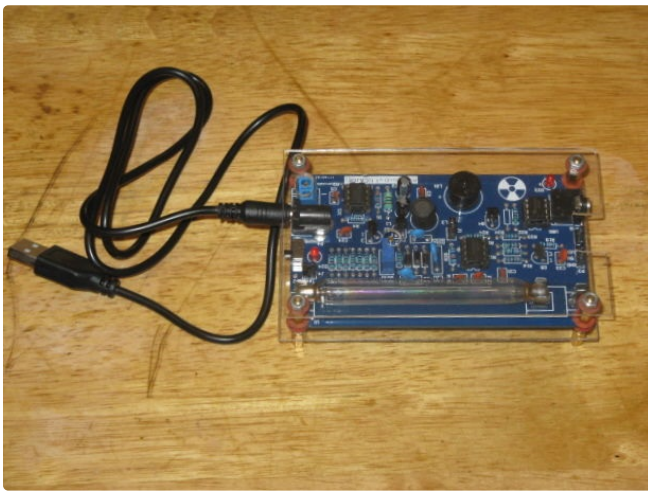
## Step 3: Electrically Testing the Geiger Counter

First use the right voltage for the power supply; the USB cord supplies 5 volts DC right from your computer, however the 3 AA battery holder is for 1.5 volt alkaline batteries making a total voltage of 4.5 volts. If you use 1.2 volt rechargeable NI-Cd or NI-MH batteries you will need a 4 AA battery holder for a total voltage of 4.8 volts. If you use less than 4.5 volts the Geiger counter may not act as it should.

There is very little circuitry on the Geiger counters output; so as long as the speaker makes a ticking sound, and the LED blinks, you should get a signal on the VIN pin.

To be sure of the output signal; connect an oscilloscope to the output by connecting the positive side of the oscilloscope probe to the VIN and the negative side of the oscilloscope probe to the ground.

Rather than just waiting on background radiation to trigger the Geiger counter I used americium-241 from a smoke detectors ion chamber to increase the Geiger counters reactions. The output of the Geiger counter started at +3 volts and dropped to 0 volts every time the Geiger tube reacted to the alpha particles and returning to +3 volts a moment later. This is the signal you will be recording with Arduino.



## Step 4: Wiring

There are two ways you can connect the Geiger counter to you Arduino and your computer.

Connect the GND on Arduino to the GND on the Geiger counter.

Connect the 5V on Arduino to the 5V on the Geiger counter.

Connect the VIN on the Geiger counter to the D2 on Arduino.

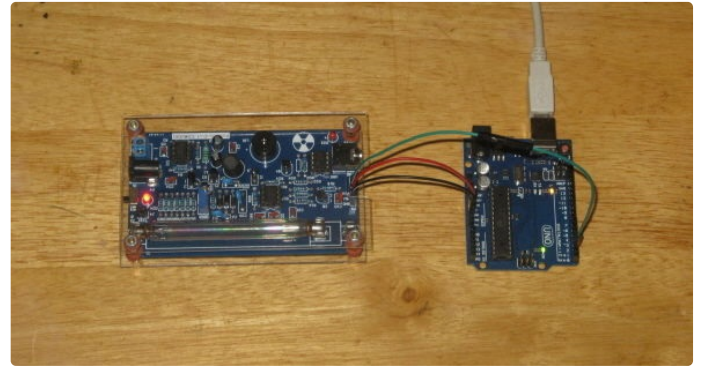
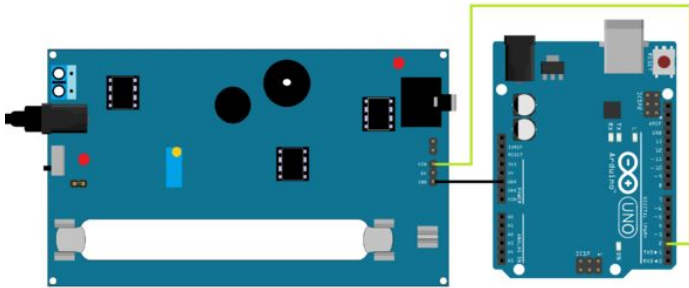
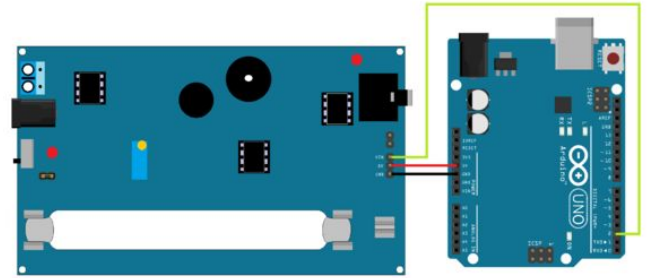
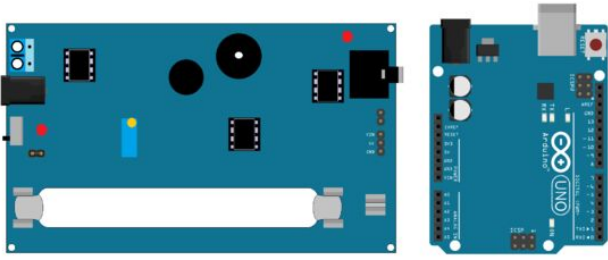
With independent power connected to the Geiger counter.

Connect the GND on Arduino to the GND on the Geiger counter.

Connect the VIN on the Geiger counter to the D2 on Arduino.

Connect Arduino to your computer.





## Step 5: Code

Open Arduino IDE and load the code.

// This Sketch counts the number of pulses a minute.

// Connect the GND on Arduino to the GND on the Geiger counter.

// Connect the 5V on Arduino to the 5V on the Geiger counter.

// Connect the VIN on the Geiger counter to the D2 on Arduino.

unsigned long counts; //variable for GM Tube events

unsigned long previousMillis; //variable for measuring time

void impulse() { // dipanggil setiap ada sinyal FALLING di pin 2

counts++;

}

#define LOG\_PERIOD 60000 // count rate

void setup() { //setup

counts = 0;

Serial.begin(9600);

pinMode(2, INPUT);

attachInterrupt(digitalPinToInterrupt(2), impulse,

FALLING); //define external interrupts

Serial.println("Start counter");

}

void loop() { //main cycle

unsigned long currentMillis = millis();

if (currentMillis - previousMillis > LOG\_PERIOD) {

previousMillis = currentMillis;

Serial.println(counts);

counts = 0;

}

}

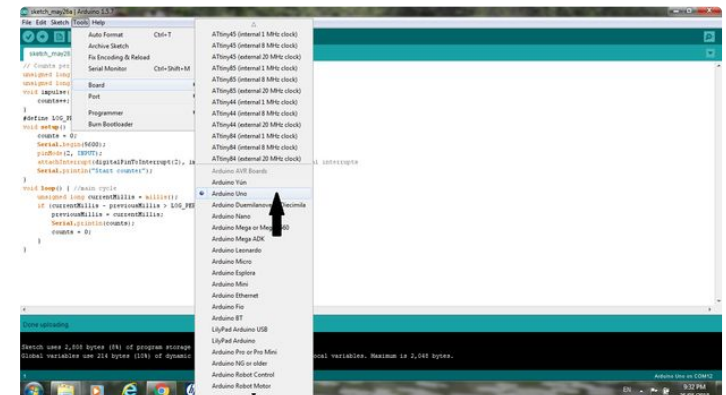
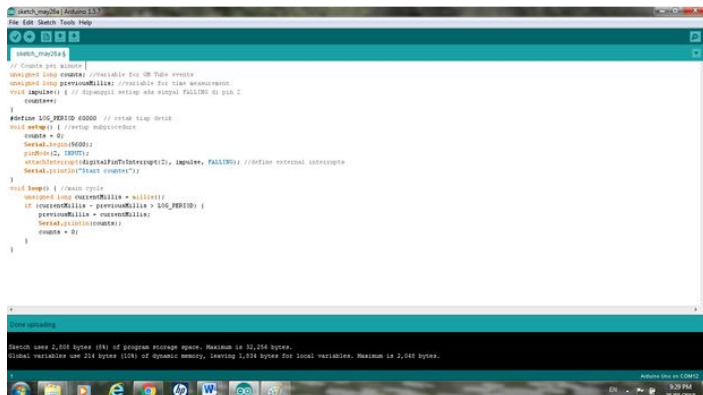
In Tools select the Arduino or other board you are using.

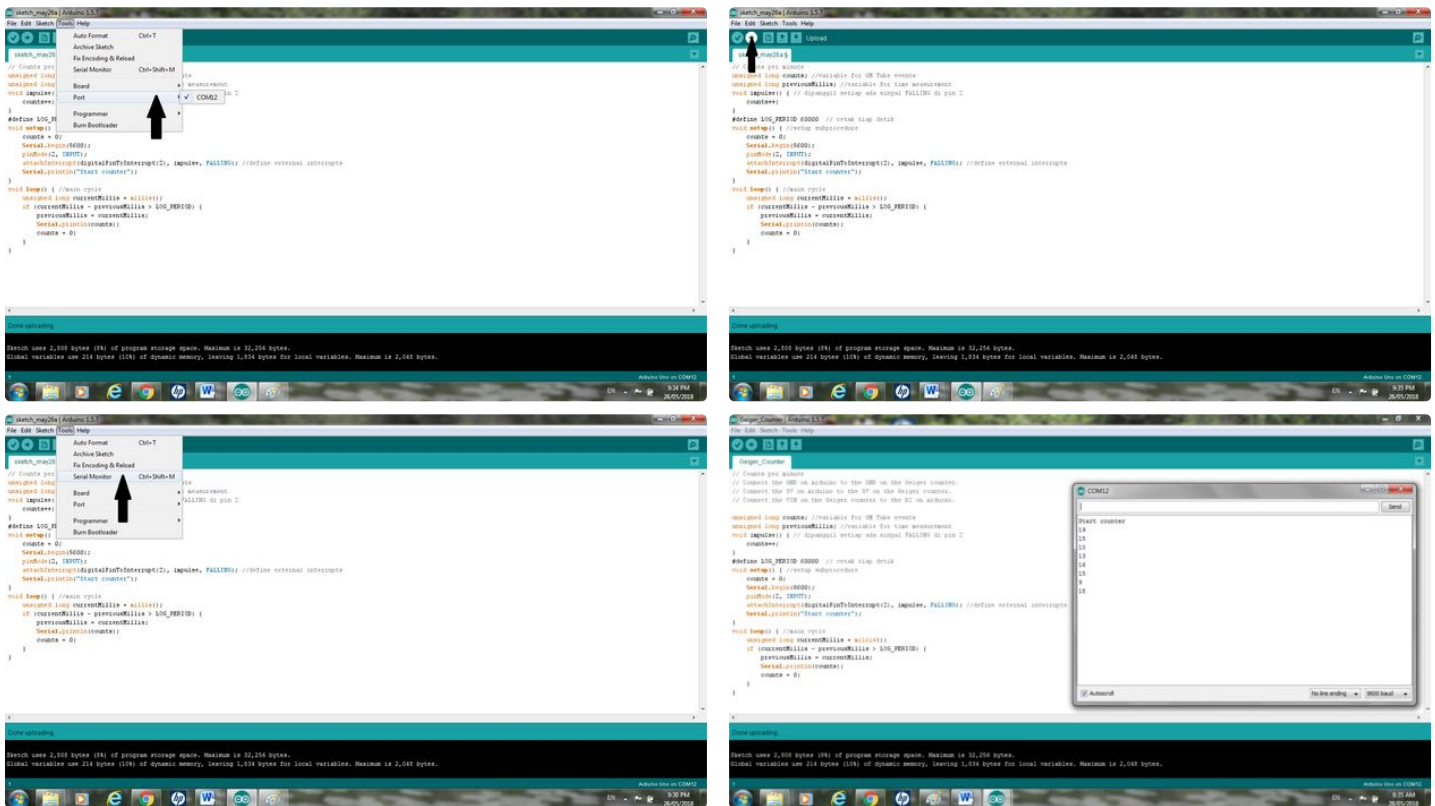
In Tools select the Port and Com

Upload the code.

Once the code is uploaded in Tools select Serial Monitor and watch your Geiger counter work.

Look for glitches. The only thing about this code is it is a bit tedious you must wait 1 minute for every count.



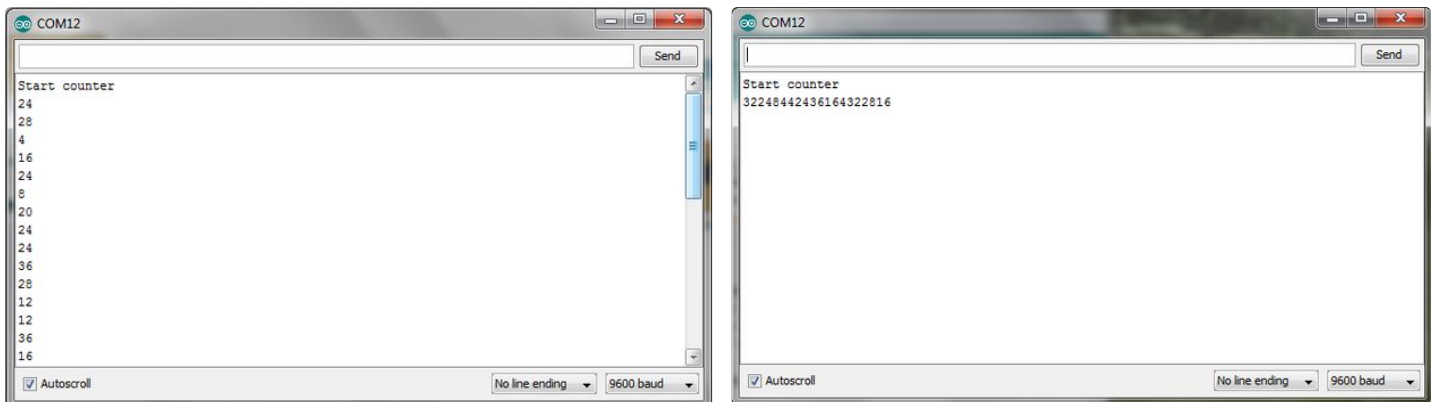


## Step 6: Serial.println Vs Serial.print

This is one of the first glitches I found in the code; so watch for it in your code, “Serial.println(cpm);” and “Serial.print(cpm);”.

Serial.println(cpm); will print each count on its own line.

Serial.print(cpm); will look like one big number printing each count on the same line making it impossible to tell what the count is.



## Step 7: J305 Background Radiation Measurement

First is the measurement of background radiation, the natural radiation that already exists naturally. The listed number is the CPM (count per minute), which is a total of measured radioactive particles every minute.

The J305 background average count was 15.6 CPM.



### Step 8: J305 Measurement of Smoke Sensor Radiation

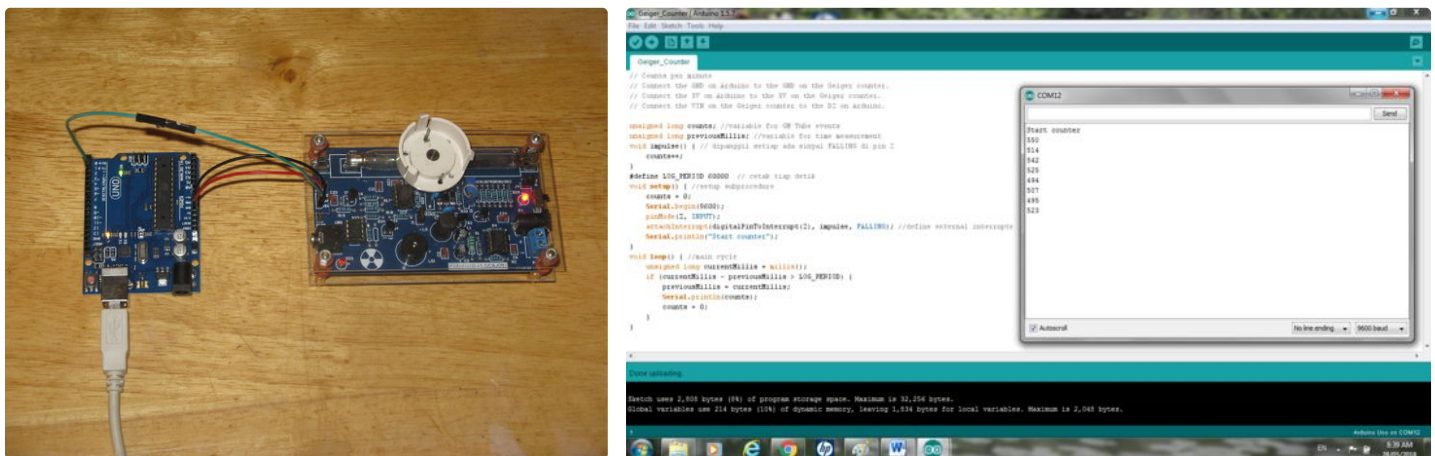
It is not uncommon for a Geiger counter to give you the same count repeatedly so check it with a radiation source. I used the radiation measurement from Americium an ion chamber from a smoke detector.

The smoke sensor utilizes Americium as a source of alpha particles that ionize smoke particles in the air. I removed the metal cap on the sensor so the alpha and beta particles can get to the Geiger tube along

with the gamma particles.

If everything is all right the counts should change.

Americium-241 from a smoke detectors ion chamber average count was 519 CPM.





## Step 9: SBM-20

This Arduino sketch is modified version written by Alex Boguslavsky.

This Sketch counts the number of pulses in 15 seconds and converts it to counts per minute making it less tedious.

Code I added "Serial.println("Start counter");".

Code I changed; "Serial.print(cpm);" to "Serial.println(cpm);".

"#define LOG\_PERIOD 15000"; sets the count time to 15 seconds, I changed it to "#define LOG\_PERIOD 5000" or 5 seconds. I found no appreciable difference in the average between counting for 1 minute, or 15 seconds and 5 seconds.

#include

#define LOG\_PERIOD 15000 //Logging period in milliseconds, recommended value 15000-60000.

#define MAX\_PERIOD 60000 //Maximum logging period without modifying this sketch

unsigned long counts; //variable for GM Tube events

unsigned long cpm; //variable for CPM

unsigned int multiplier; //variable for calculation CPM in this sketch

unsigned long previousMillis; //variable for time measurement

void tube\_impulse(){ //subprocedure for capturing events from Geiger Kit

counts++;

}

void setup(){ //setup subprocedure

counts = 0;

cpm = 0;

multiplier = MAX\_PERIOD / LOG\_PERIOD;  
//calculating multiplier, depend on your log period

Serial.begin(9600);

attachInterrupt(0, tube\_impulse, FALLING); //define external interrupts

Serial.println("Start counter"); // code I added

}

void loop(){ //main cycle

unsigned long currentMillis = millis();

if(currentMillis - previousMillis > LOG\_PERIOD){

previousMillis = currentMillis;

cpm = counts \* multiplier;

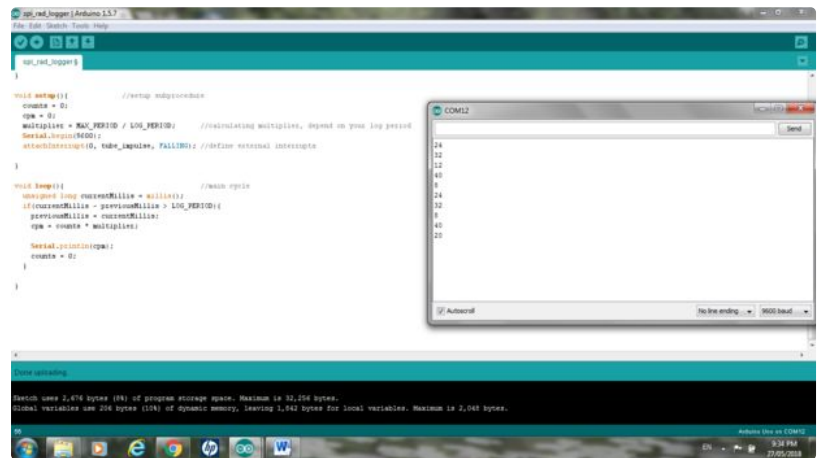
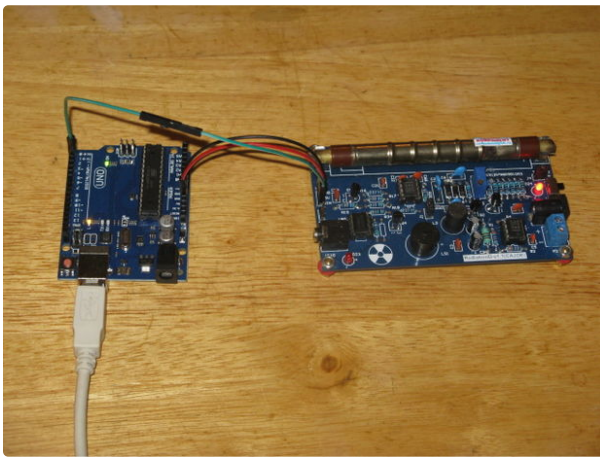
Serial.println(cpm); // code I changed

counts = 0;

}

}

The SBM-20 background average count was 23.4 CPM.



## Step 10: Wiring the Geiger Counter With an LCD

LCD Connection:

LCD K pin to GND

LCD A pin to 220  $\Omega$  resistor to Vcc

LCD D7 pin to digital pin 3

LCD D6 pin to digital pin 5

LCD D5 pin to digital pin 6

LCD D4 pin to digital pin 7

LCD Enable pin to digital pin 8

LCD R/W pin to ground

LCD RS pin to digital pin 9

LCD VO pin to adjust of 10 k $\Omega$  pot

LCD Vcc pin to Vcc

LCD Vdd pin to GND

Pot 10 k $\Omega$  adjustable resistor.

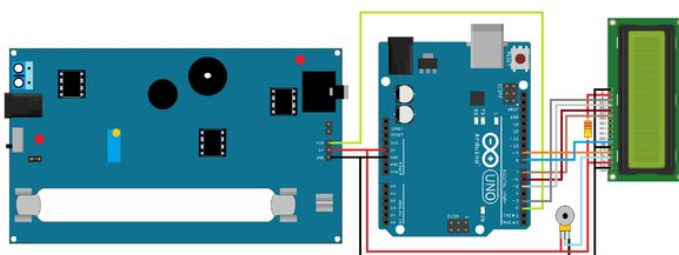
Vcc, Vo, Vdd

Geiger Counter

VIN to digital pin 2

5 V to +5V

GND to ground



## Step 11: Geiger Counter With LCD

// include the library code:	attachInterrupt(0,GetEvent,FALLING); // Event on pin 2
#include	}
#include	
#define LOG_PERIOD 15000 //Logging period in milliseconds, recommended value 15000-60000.	void loop() {
#define MAX_PERIOD 60000 //Maximum logging period without modifying this sketch	lcd.setCursor(0,0); // print text and CNT on the LCD
#define PERIOD 60000.0 // (60 sec) one minute measure period	lcd.print("CPM:");
volatile unsigned long CNT; // variable for counting interrupts from dosimeter	lcd.setCursor(0,1);
unsigned long counts; //variable for GM Tube events	lcd.print("CNT:");
unsigned long cpm; //variable for CPM	lcd.setCursor(5,1);
unsigned int multiplier; //variable for calculation CPM in this sketch	lcd.print(CNT);
unsigned long previousMillis; //variable for time measurement	if (millis() >=dispPeriod + PERIOD) { // If one minute is over
unsigned long dispPeriod; // variable for measuring time	cleanDisplay(); // Clear LCD
unsigned long CPM; // variable for measuring CPM	// Do something about accumulated CNT events....
// initialize the library with the numbers of the interface pins	lcd.setCursor(5, 0);
LiquidCrystal lcd(9, 8, 7, 6, 5, 3);	CPM = CNT;
void setup() { // setup	lcd.print(CPM); //Display CPM
lcd.begin(16, 2);	CNT = 0;
CNT = 0;	dispPeriod = millis();
CPM = 0;	}
dispPeriod = 0;	}
lcd.setCursor(0,0);	void GetEvent(){ // Get Event from Device
	CNT++;
	}
	void cleanDisplay (){ // Clear LCD routine
	lcd.clear();

```

lcd.print(" RH Electronics ");

lcd.setCursor(0,1);

lcd.print(" Geiger Counter ");

delay(2000);

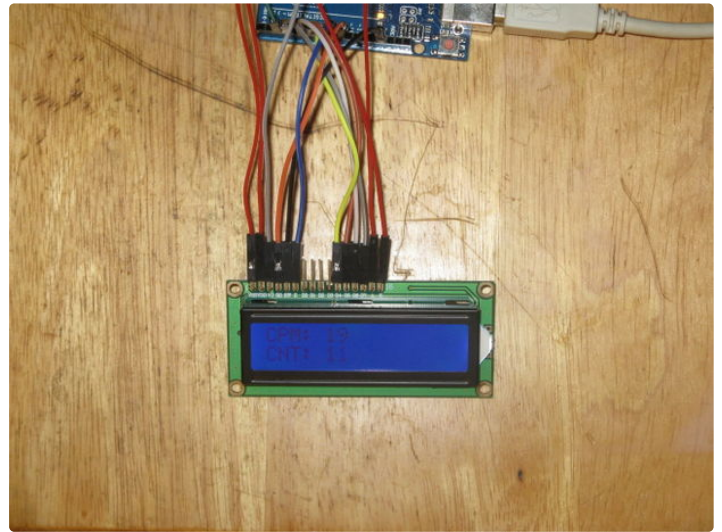
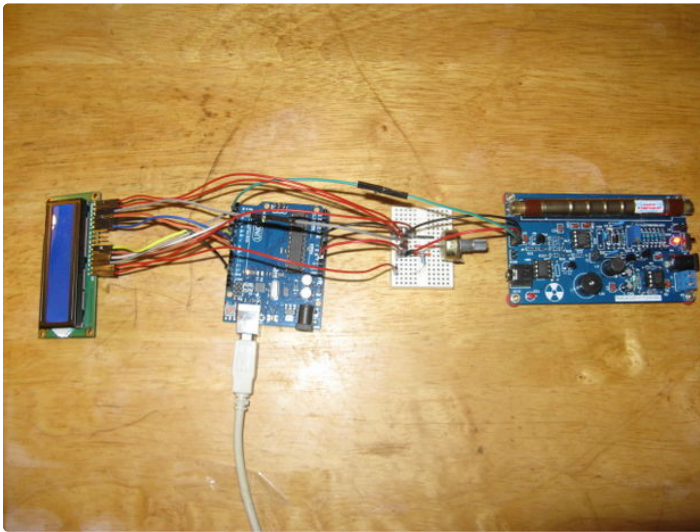
cleanDisplay();

                                lcd.setCursor(0,0);

                                lcd.setCursor(0,0);

                                }

```






## Step 12: Files

Download and install these files to your Arduino.

Place each .ino file in a folder by the same name.



	<a href="http://www.instructable...">http://www.instructable...</a>	<a href="#">Download</a>
	<a href="http://www.instructable...">http://www.instructable...</a>	<a href="#">Download</a>
	<a href="http://www.instructable...">http://www.instructable...</a>	<a href="#">Download</a>
	<a href="http://www.instructable...">http://www.instructable...</a>	<a href="#">Download</a>