**Arduino Tachometer**

**Jump To:**

* Part 1: Introduction
  + **Part 1: Introduction**
  + [Part 2: Parts List](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/parts.html)
  + [Part 3: Schematic](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/schematic.html)
  + [Part 4: Theory of Operation](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/theory.html)
  + [Part 5: Hardware](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/hardware.html)
  + [Part 6: Software](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/software.html)
  + [Part 7: Data & Observations](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/data.html)
  + [Part 8: Conclusion](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/conclusion.html)

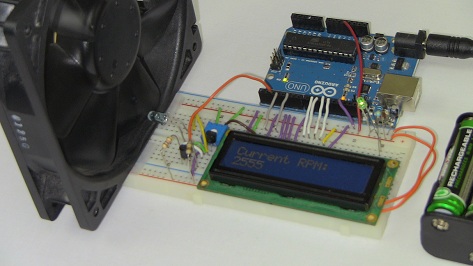
**Project Info**  
Author: Chris  
Difficulty: Medium  
Time Invested: 3 Hours  
  
Prerequisites:

* [Intro To Arduino](http://www.pyroelectro.com/tutorials/arduino_led_blinky_intro/)
* [Arduino LCD Interface](http://www.pyroelectro.com/tutorials/16x2_lcd_arduino/)
* [PIC Tachometer](http://www.pyroelectro.com/tutorials/digital_tachometer_rpm/)

Take a look at the above  
articles before continuing  
to read this article.

           A tachometer is a useful tool for counting the RPM (rotations per minute) of a wheel or basically anything that spins. The easiest way to build a tachometer is using a [transmitter and receiver](http://www.pyroelectro.com/tutorials/wireless_infrared_communication/). When the link between them is broken, you know that something is spinning and can execute some code that calculates the current RPM of whatever is spinning to break the transmitter/receiver link.  
           In this article we will explore how to use an IR transmitter and receiver break-beam pair similar to the [PIC Tachometer](http://www.pyroelectro.com/tutorials/digital_tachometer_rpm/) project I built a few months ago, but because of popular demand, the [Arduino](http://www.pyroelectro.com/tutorials/arduino_led_blinky_intro/) system will be used for all the processing and break-beam interruption counting. The end result will be a 16x2 LCD displaying the RPM of some computer fans.

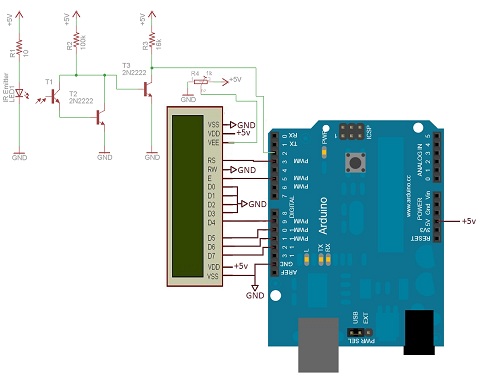
**Arduino Tachometer - Demonstration**

**Arduino Tachometer - Project Setup**  
[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer.jpg)

**Purpose & Overview Of This Project**  
           The purpose of this project is to build a single input, single output system. The input will come in the form of a signal state change from high (+5v) to low (+0v) which will occur when the IR break-beam is interrupted and the Arduino will then increment an internal counter. As time goes on, additional processing and calculation will occur as interrupts are trigger and the LCD will output the calculated RPM.  
           To create the IR break-beam, we will used an IR LED with a low value resistor so that it shines very bright. The receiver will be a phototransistor which biases 'on' whenever the IR LED's light is detected. A computer fan will be placed between the IR link and turned on so as to continuously generate an interrupt through some additional transistor logic circuitry. For output, the [Arduino LCD interface](http://www.pyroelectro.com/tutorials/16x2_lcd_arduino/) that we saw last week will be used so that we can output the final RPM value to the LCD.

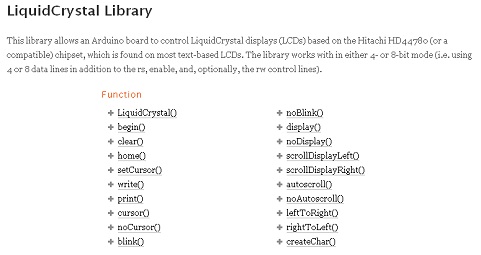
**Parts List Details**  
           The parts used in this project are all listed out above, but the more interesting and necessary parts are listed out below with a little more detail to describe their function.  
  
[**Arduino UNO**](http://www.pyroelectro.com/parts/arduino/)  
           This is the Arduino board that we will be using to process the IR break-beam pulses that tell us when the CPU fan has moved. The Arduino will use these pulses along with a timer to figure out what the current RPM of the fan is.  
  
[**16x2 LCD**](http://www.pyroelectro.com/parts/16x2_LCD/)  
           After the Arduino has figured out what the curent RPM is, it will be displayed on this LCD so that it's obvious to the user.  
  
[**5kΩ Trimpot**](http://www.pyroelectro.com/parts/5k_Trimmer/)  
           This trimpot will be used for setting the contrast of the 16x2 LCD. It's gives an anaolg output varying from +5v to 0, which the LCD translates to a brightness setting.  
  
[**IR Emitter Diode**](http://www.pyroelectro.com/parts/IR_Emitter_Diode/) and [**Phototransistor**](http://www.pyroelectro.com/parts/IR_Detector/)  
           The photo transistor turns on whenever intense Infrared light shines on it. So whenever the IR LED is on and shining, it keeps the phototransistor biased 'on', but if the IR LED is blocked, by...for example a CPU fan blade, the phototransistor is biased 'off'.  
  
[**2n3904**](http://www.pyroelectro.com/parts/2n3904/) and [**2n3906**](http://www.pyroelectro.com/parts/2n3906/)  
           These transistors will mainly be used as level shifters to ensure the pulses output from the IR break-beam to the Arduino come in the form of +0v to +5v and nothing in-between.

**Schematic Overview**  
           The circuit diagram for this project is a little more complicated than last week's [16x2 LCD Interface](http://www.pyroelectro.com/tutorials/16x2_lcd_arduino/). For starters, the LCD interface is simplified to have only 2 control lines and 4 data lines. Then the tachometer IR break-beam circuit is added on the side to make things a little more complex.

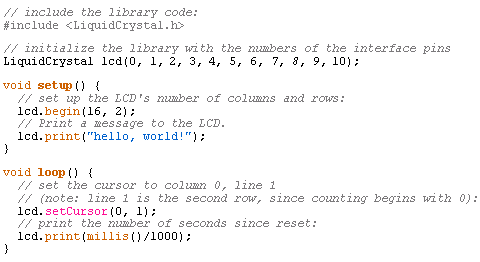
[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/schematic.png)  
[View Full Schematic](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/schematic.png)

**Schematic Specifics**  
  
**16x2 LCD Interface**  
           2 control pins and 4 data pins are connected from the Arduino to the LCD. These are what will tell the LCD what to do and when.  
  
**IR Break-Beam Circuit**  
           The break-beam circuit's signal goes to the digital input pin #2 on the Arduino. This will interrupt the Arduino so it can count that a pulse has just been registered and the tachometer is reading data.

**Arduino LCD Library**  
           For this project we'll be using the same Arduino LCD library as before to control and output to the LCD. Mainly we will just be udpated the 2nd row with the newly calculated RPM value. But we'll still use the same theory as seen here.

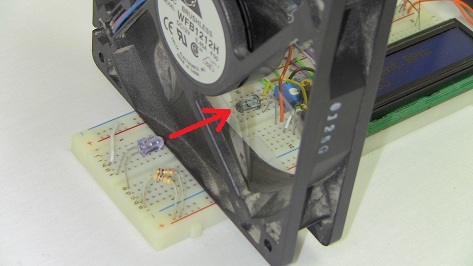
[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/lcd_lib.jpg)

           Just as a refresher, the basic way to setup the Arduino LCD interface uses the 'Hello, World!' code seen below. We'll be using code very similar to this, especially the: "lcd.print(millis()/1000);" for our tachometer.

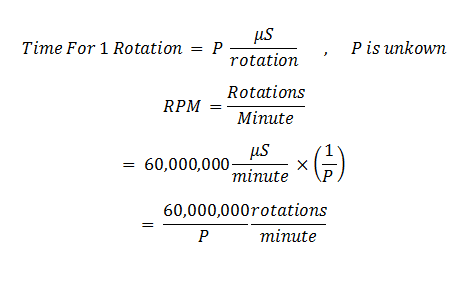


           So understand these LCD library functions as best you can before continuing. They're not overly sophisticated and are well documented on Arduino's website.

**Arduino RPM Counting**  
           Since we're going to be counting the RPM for a CPU fan, first we need to realize that we're using an IR break-beam that counts every interruption. This is great, except we do need to realize that the CPU fan has 7 blades. This means 7 interrupts is 1 RPM.

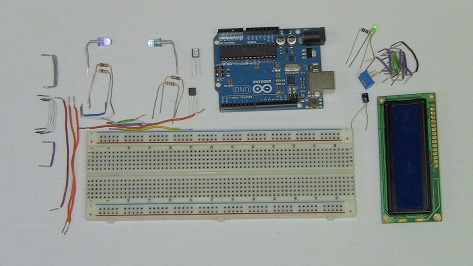
[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/break_beam.jpg)

           If we keep track of the interrupt count, we can know that every 7th interruption means 1 full rotation has just occured. Similarly, if we keep track of the time it takes for every full rotation to occur, we can then easily calculated the full RPM.

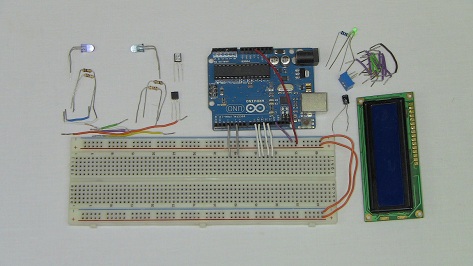


           We'll use the formula seen above in our code to calculate the RPM. The formula is rock solid, the accuracy of our tachometer will now depend upon how well our Arduino can keep track of the time between interrupts and full rotation counts.

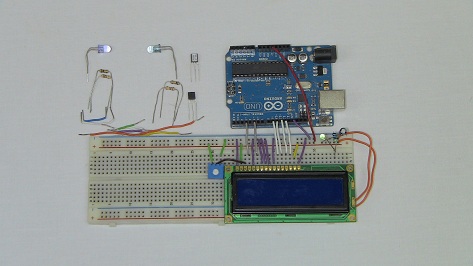
**Hardware Design**  
           There are two main portions of the hardware design here. The first is connecting the LCD to the Arduino like we did in the [16x2 LCD Arduino Interface](http://www.pyroelectro.com/tutorials/16x2_lcd_arduino/). The second is building the IR break-beam circuit. Still there's not too many parts, so it shouldn't take very long to build.  
  
**Building The Complex Circuit**  
          Below you can see all the parts necessary to get started wiring the circuit up exactly as you saw in the [schematic](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/schematic.html) details.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer_0.jpg)

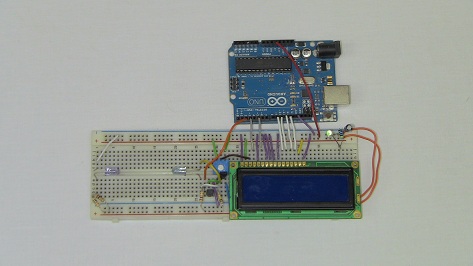
**·**First the +5v power and LCD data/control connections are made.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer_1.jpg)

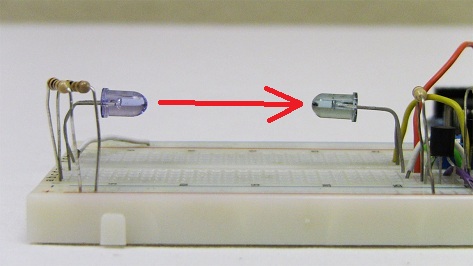
**·**The LED is connected, along with contrast control and power LED.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer_2.jpg)

**·**The IR break beam circuit is now assembled. Try to keep a good distance between the IR LED and phototransistor.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer_3.jpg)

**·**Here you can see a decent distance between the IR transmitter/receiver, where I'll put the CPU Fan.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/break__beam.jpg)

**·**Enough hardware talk! Let's get started with some firmware/software so we can see this thing in The main loop seen below is where the RPM is calcuated and the LCD is updated. Since the main loop is gigantic while(1) loop, which means it runs over and over forever, the RPM is being calcualted and LCD updated many times a second. The interrupt function is counting the time between interrupts so that in the main loop the RPM calculation can be done.  
  
**16x2 Static LCD Text**

**------------« Begin Code »------------**

#include <LiquidCrystal.h>

LiquidCrystal lcd(3, 5, 9, 10, 11, 12);

volatile float time = 0;

volatile float time\_last = 0;

volatile int rpm\_array[5] = {0,0,0,0,0};

void setup()

{

//Digital Pin 2 Set As An Interrupt

attachInterrupt(0, fan\_interrupt, FALLING);

// set up the LCD's number of columns and rows:

lcd.begin(16, 2);

// Print a message to the LCD.

lcd.print("Current RPM:");

}

//Main Loop To Calculate RPM and Update LCD Display

void loop()

{

int rpm = 0;

while(1){

//Slow Down The LCD Display Updates

delay(400);

//Clear The Bottom Row

lcd.setCursor(0, 1);

lcd.print(" ");

//Update The Rpm Count

lcd.setCursor(0, 1);

lcd.print(rpm);

////lcd.setCursor(4, 1);

////lcd.print(time);

//Update The RPM

if(time > 0)

{

//5 Sample Moving Average To Smooth Out The Data

rpm\_array[0] = rpm\_array[1];

rpm\_array[1] = rpm\_array[2];

rpm\_array[2] = rpm\_array[3];

rpm\_array[3] = rpm\_array[4];

rpm\_array[4] = 60\*(1000000/(time\*7));

//Last 5 Average RPM Counts Eqauls....

rpm = (rpm\_array[0] + rpm\_array[1] + rpm\_array[2] + rpm\_array[3] + rpm\_array[4]) / 5;

}

}

}

//Capture The IR Break-Beam Interrupt

void fan\_interrupt()

{

time = (micros() - time\_last);

time\_last = micros();

}

**------------« End Code »------------**

           Remember that the CPU fan has 7 blades, so this tachometer is only meant to work with fans like that. If your fan or encoder only sends 4 pulses per rotation, you'll want to change that in the c**Data & Observations**  
           Here is the demonstration video of how the system works given input and resulting output. A fan is placed inbetween the IR break-beam link and things go from there. go watch:

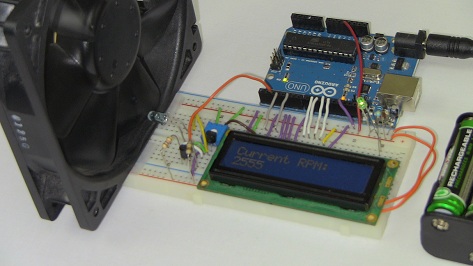
           The two fans operate at around 3000 RPM and 2600 RPM which was reflected with an error of about +/-100 RPM. I attribute that mostly to round-off errors in the calculation process. Either way hopefully you are convinced that the system works as designed.ode so that "(time\*4)".action.

**Data & Observations**  
           Here is the demonstration video of how the system works given input and resulting output. A fan is placed inbetween the IR break-beam link and things go from there. go watch:

<http://www.youtube.com/watch?v=L6QhLbdpVjY&feature=player_embedded>

           The two fans operate at around 3000 RPM and 2600 RPM which was reflected with an error of about +/-100 RPM. I attribute that mostly to round-off errors in the calculation process. Either way hopefully you are convinced that the system works as designed.

**An Overview Of The Arduino Tachometer**  
           The system we designed here is another good example of given some input, what will you output? The CPU Fan generated interrupt pulses and our output system reacted accordingly telling us how many RPM the fan spins at. Although it wasn't 100% precise, I will take 95% precise for the $10 in parts that it takes to build this tachometer with an Arduino.

[](http://www.pyroelectro.com/tutorials/tachometer_rpm_arduino/img/arduino_tachometer.jpg)

**What To Do Now**  
           Break-beam systems are useful not just for tachometer readings but also for reaction based systems. Perhaps you want to know when a door is opened or shut. Maybe you want to be sure nothing passed under your robot. There are many uses for a break-beam and while the circuits used here are only introductory, there's many routes you can take to build bigger and better IR break-beams for tachometers and other awesome devices.