Arduino-Based Optical Tachometer

by **CMPalmer** on September 16, 2007

Table of Contents

Intro: Arduino-Based Optical Tachometer

Over ten years ago, I put up a web page with detailed instructions on building a simple electric motor based on one from the Beakman's World TV show. I called it the "Beakman's Electric Motor" page and over the years it has had hundreds of thousands, if not millions, of hits. Realizing that just building a motor, no matter how cool, wasn't a good science fair project, I added suggestions for using the motor as a science fair project, such as experimenting with different magnets, batteries, and coil constructions and seeing how they affect performance. In order to do this, the speed of the motor should be measured, but I left that as an open-ended question.

I've had dozens of e-mails over the years asking how to measure the speed and I've always suggested using a broken light beam and a counter, but I've never built one myself. I even suggested to one person that they use a slot-car lap counter since I'd seen on one sale at Toys 'R Us for 99 cents and she said that it worked perfectly, but not everyone can find a good deal like that.

It just so happened that I got one of these "how do I measure the motor speed?" e-mails on the same day my new Arduino Diecimila microcontroller board arrived from the Make Store , so I thought that would make a great weekend project.

Here is the result, an optical tachometer for Beakman's Electric Motor using an IR emitter/detector pair and a Arduino board. With a few modifications to the programming, you can use this tachometer for measuring other things such as fan or propeller speed. Notes are included on what to change for different applications.

How to have fun with Arduino is a good source of basics on how to setup and use the Arduino board.

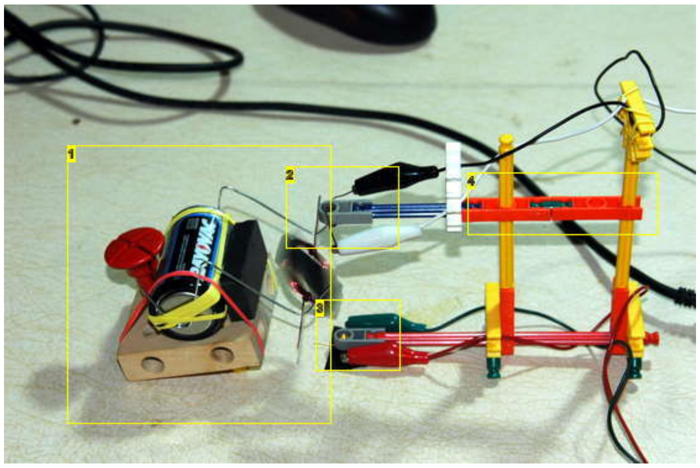


Image Notes
1. Beakman's Motor
2. IR LED
3. IR Detector (Phototransistor)
4. This part of the frame slides up and down to adjust the light beam gap

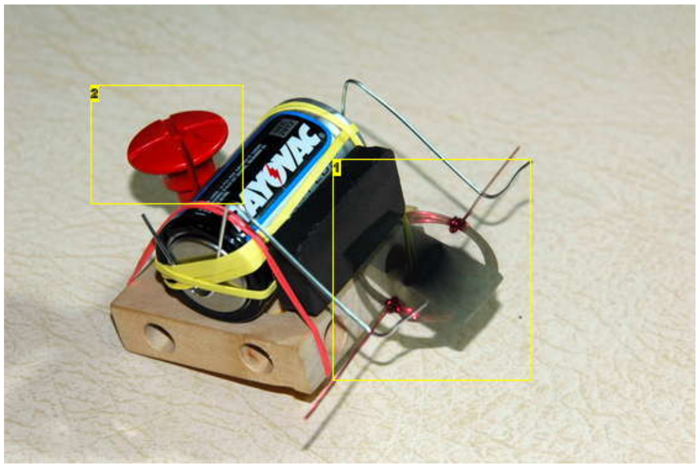


Image Notes
1. Spinning coil
2. This does nothing

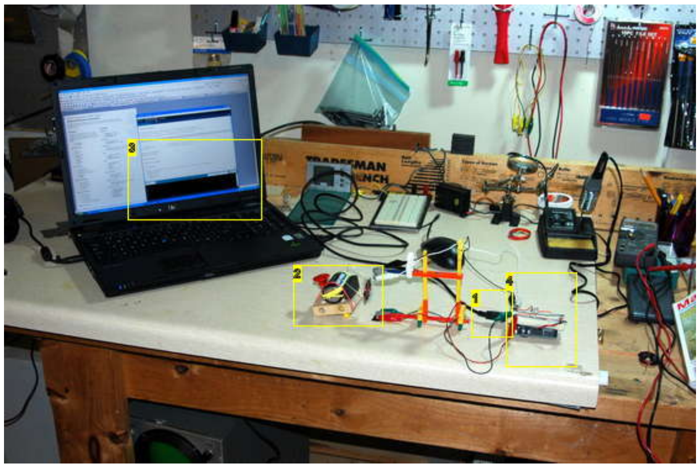


Image Notes
1. USB connection
2. Motor
3. Results
4. Arduino board and proto board.

Step 1: Materials Needed

Arduino Diecimila Board
Available from the Make Store or from several other online resources. Note however that the techniques of this Instructable could be adapted for other microcontrollers and circuits.

Computer with Arduino software and USB cable

IR LED and IR phototransistor
I used a Radio Shack #276-142, but that may be an old part number. Parts selection on this probably isn't too critical.

Visible light LED
I used a high-brightness red one that I had around. Actual selection not too critical.

10K Ohm resistor

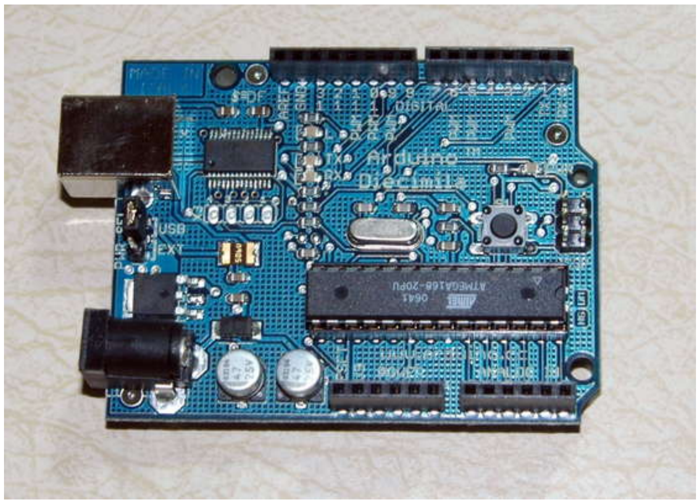
220 Ohm resistor

Breadboard (semi-optional), hookup wires, clips

Opaque tape, such as black electrical tape

Framework for holding LED and detector
Use your imagination, I used KNex pieces to build a frame.

Beakman's Electric Motor (or something else to measure)
Original instructions for building the motor are here: Beakman's Motor
Similar plans are available from other places, such as this Instructable:
Simple Electric Motor



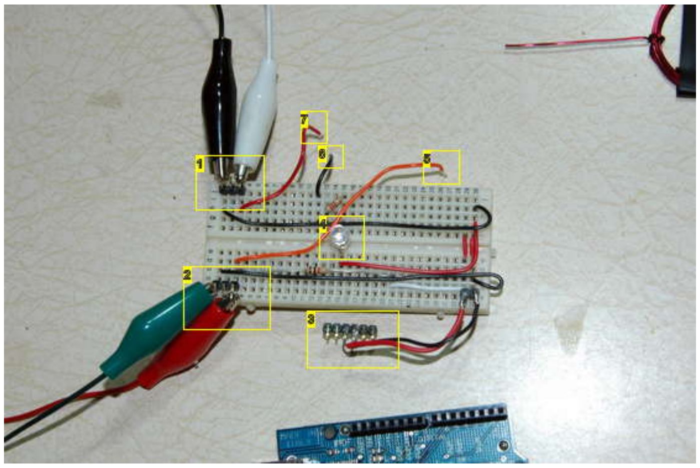
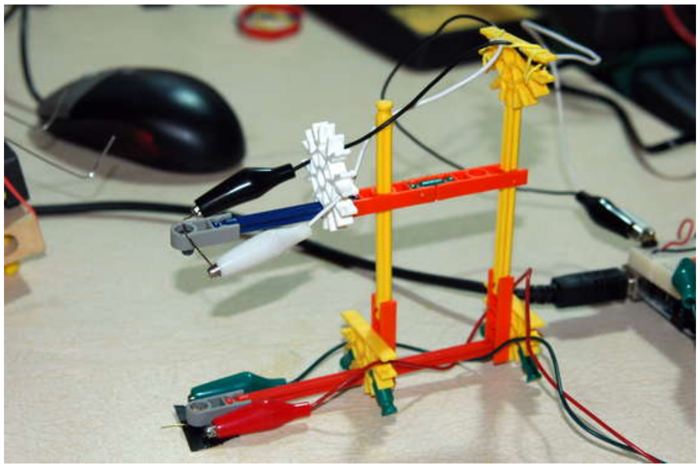
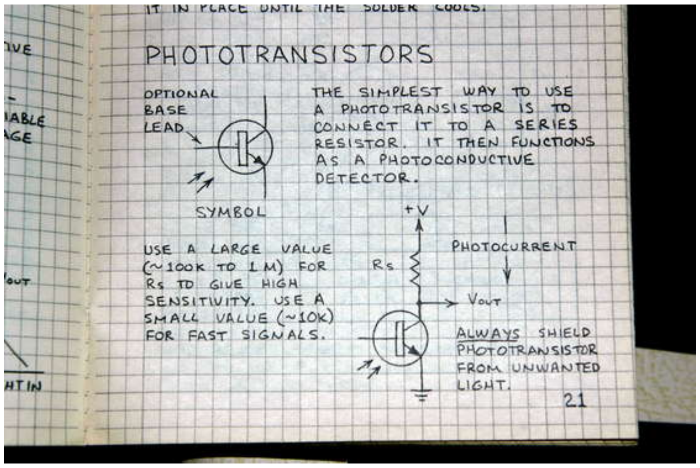


Image Notes
1. Connection to IR LED
2. Connection to phototransistor
3. Power and ground connectors
4. Status LED
5. to Arduino pin 2
6. to Arduino pin 12
7. to Arduino pin 13




IT IN PLACE UNTIL THE SOLDER COOLS.

PHOTOTRANSISTORS

THE SIMPLEST WAY TO USE A PHOTOTRANSISTOR IS TO CONNECT IT TO A SERIES RESISTOR. IT THEN FUNCTIONS AS A PHOTOCONDUCTIVE DETECTOR.

USE A LARGE VALUE (~100K TO 1 M) FOR Rs TO GIVE HIGH SENSITIVITY. USE A SMALL VALUE (~10K) FOR FAST SIGNALS.

ALWAYS SHIELD PHOTOTRANSISTOR FROM UNWANTED LIGHT.

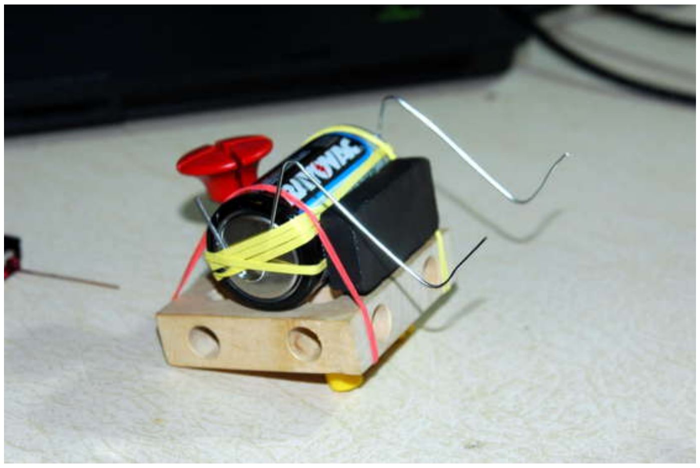
21

Step 2: Building and Preparing the Motor (if necessary)

Follow the instructions to build the Beakman's Electric motor. The motor should be mounted on a suitable base so that it can run freestanding a bit off the table. I used some Brio construction set pieces (I have a lot of toys laying around).

The most important thing to ensure reliable motor operation is to make sure the coil is balanced and the tails of the coil are very straight. I drew a diagram using my coil form and a ruler, then laid the completed coil on the paper to align the tails perpendicular to the coil and to make sure they were straight. I then used a drop of superglue where the tails were wrapped around the coil to make sure they couldn't slide.

To enable the coil to break the light-beam, place a piece of opaque tape (I used black electrical tape) across the coil. This might slow it down a tiny bit, but if all of your coils have the same sized piece of tape in the same position, your results between coils should be relatively consistent.



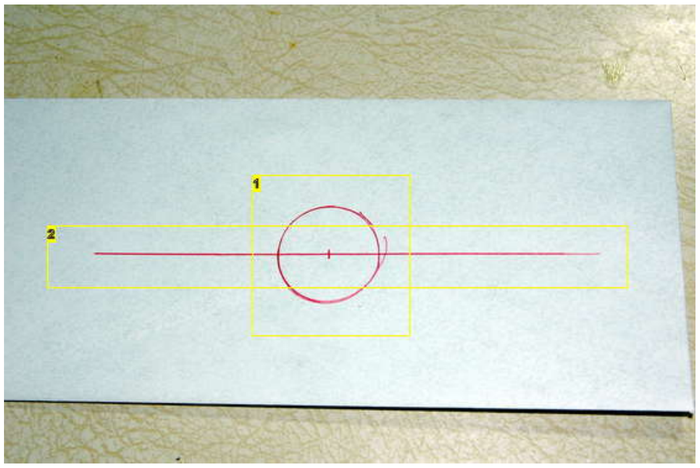


Image Notes
1. Same diameter as coil
2. Straight line for aligning coil tails

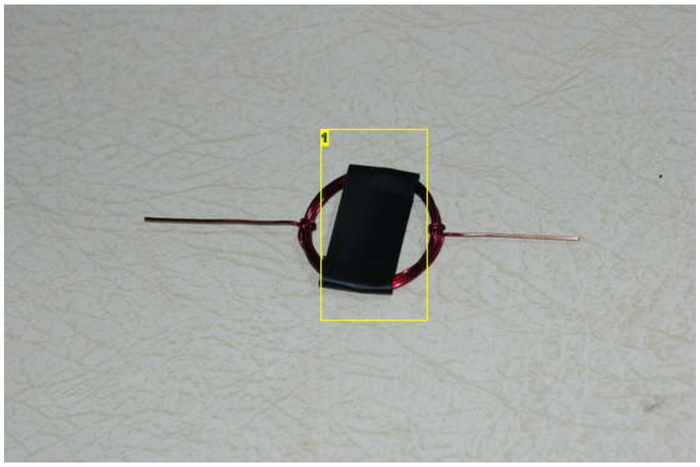


Image Notes
1. Black electrical tape across middle of coil

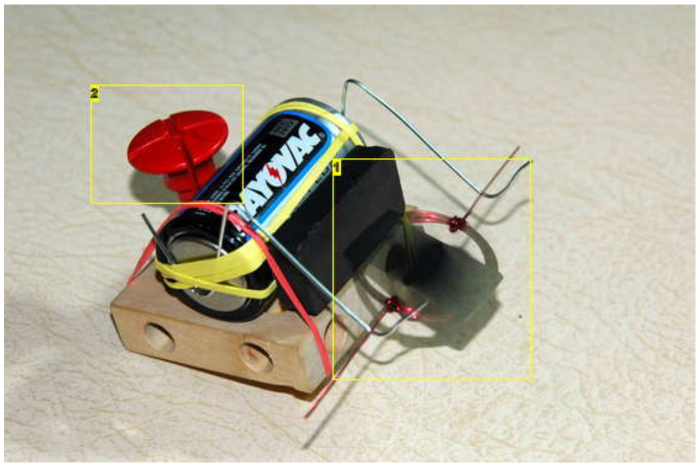


Image Notes
1. Spinning coil
2. This does nothing

Step 3: IR Detector Circuit

There are three separate circuits to connect to the Arduino. You can refer to my schematic and notes sketch in the images and to the pictures of the breadboard to see how I hooked it all up. Of course, the critical things to note are the anode/cathode orientation of the LEDs and transistors and the connections to power and ground. The whole circuit is powered from the Arduino board and since the program will communicate with the PC, I'm using the USB connector for power.

IR Detector Circuit
I figured the easiest way to detect breaks in the light path was to use an IR LED and IR phototransistor, configured so that the phototransistor is either "on" or "off" instead of using a photocell with an analog threshold.

I found the configuration in the Forrest Mims circuit notebook "Optoelectronic Circuits" - the excerpt is in the images below. These books are wonderful, by the way. I'm anxious to try many of the other sensors and circuits he describes in them with the Arduino.

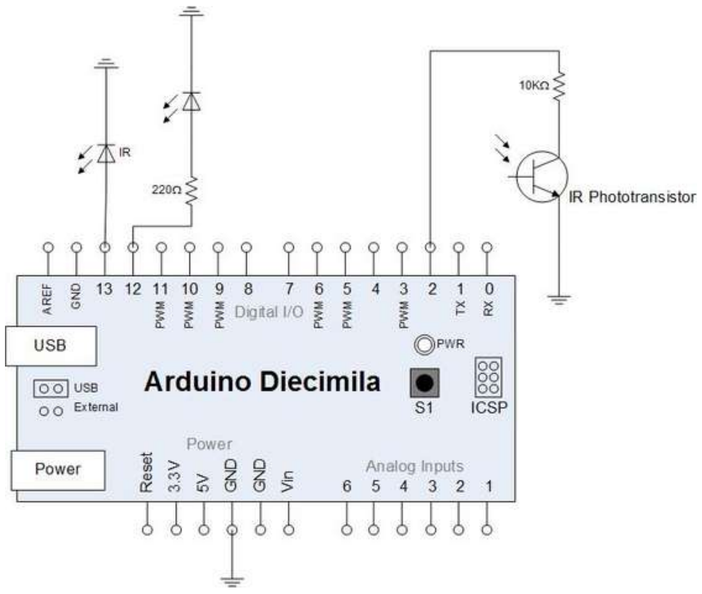
A 10K Ohm resistor goes from the +5V connector on the Arduino to the collector on the phototransistor (pin 2). The emitter of the phototransistor (pin 1) is connected to ground (again on the Arduino board). Pin 3 is unused and can be bent out of the way or clipped. The collector (pin 2) is also connected to the digital pin 2 on the Arduino. Input 2 on the Arduino corresponds to interrupt 0, which we will be using to count pulses.

I used alligator clip patch wires to connect the phototransistor and small angled pins on the breadboard to connect to the circuit. The hookups for this project are simple enough that you could probably do point-to-board wiring and just plug the hookup wires into the sockets on the Arduino. I used a small breadboard instead. Of course, if you have the Arduino shield, construction would be even easier.

IR LED Circuit

The Arduino Diecimila has a current limiting resistor on pin 13, so to reduce the parts count, I connected the IR LED to that pin. If you have an older Arduino board, remember to connect an appropriate series resistor to the output pin. The connections for this were also done with a right-angled header on the breadboard and alligator clip patch wires to the LED. Anode to digital pin 13, cathode (flat side) to ground.

Status LED
Since we're dealing with invisible IR light, interrupts, and serial communications for status, I wanted to see something that would tell me the circuit is working, so I connected a RED status LED to digital pin 12 through a 220 Ohm series resistor.



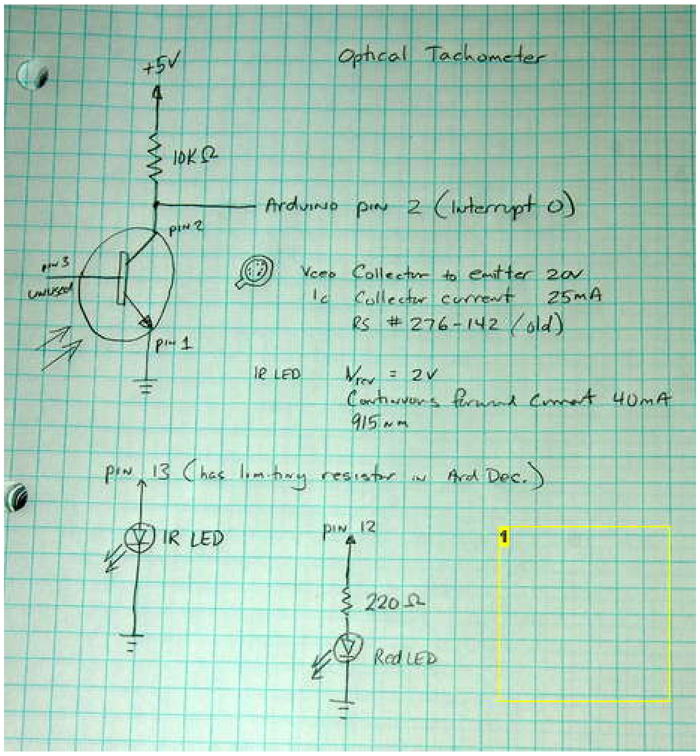
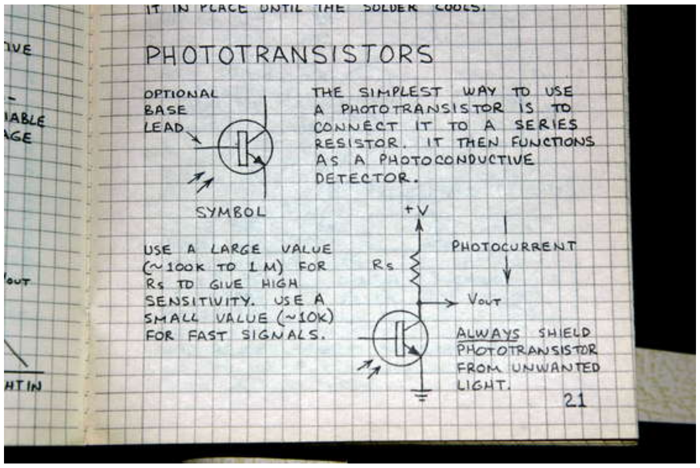

Optical Tachometer
+5V
10KΩ
Arduino pin 2 (Interrupt 0)
pin 2
pin 3
unused
pin 1
Vceo Collector to emitter 20V
Ic Collector current 25mA
RS # 276-142 (old)
IR LED
Vrev = 2V
Continuous forward current 40mA
915 nm
pin 13 (has limiting resistor in Ard Dec.)
IR LED
pin 12
220 Ω
Red LED
1

Image Notes
1. View full size (through "i" link) if you can't read the writing.


IT IN PLACE UNTIL THE SOLDER COOLS.

PHOTOTRANSISTORS

THE SIMPLEST WAY TO USE A PHOTOTRANSISTOR IS TO CONNECT IT TO A SERIES RESISTOR. IT THEN FUNCTIONS AS A PHOTOCONDUCTIVE DETECTOR.

USE A LARGE VALUE (~100K TO 1 M) FOR Rs TO GIVE HIGH SENSITIVITY. USE A SMALL VALUE (~10K) FOR FAST SIGNALS.

ALWAYS SHIELD PHOTOTRANSISTOR FROM UNWANTED LIGHT.

21
OPTIONAL BASE LEAD
SYMBOL
+V
Rs
PHOTOCURRENT
Vout

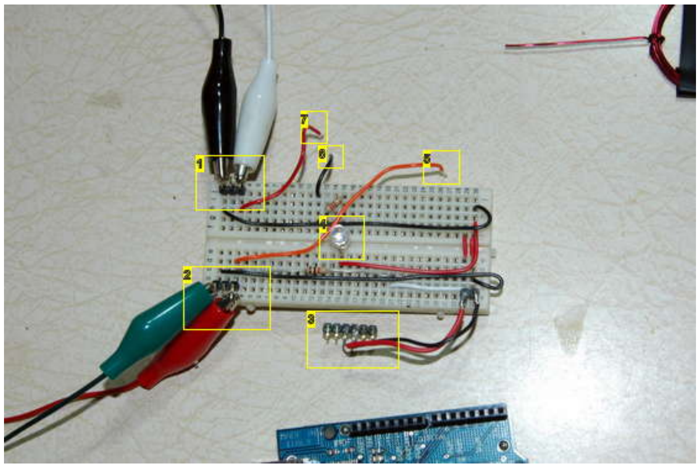


Image Notes
1. Connection to IR LED
2. Connection to phototransistor
3. Power and ground connectors
4. Status LED
5. to Arduino pin 2
6. to Arduino pin 12
7. to Arduino pin 13

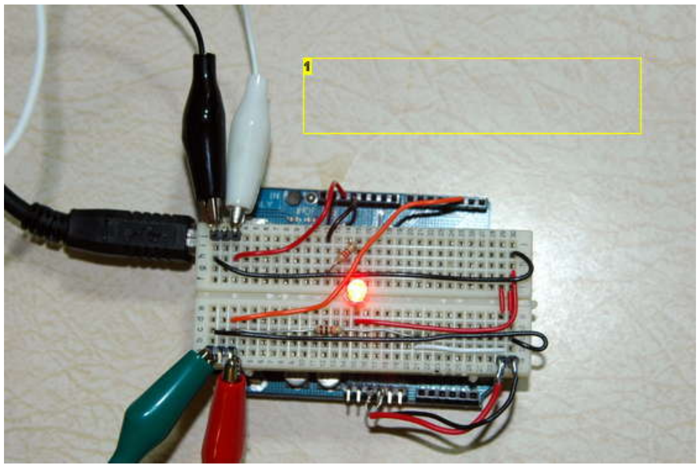


Image Notes
1. Completed circuit sitting on top of Arduino board

Step 4: Programming

The program for calculating the RPM of the motor is pretty simple. I adapted it from a Hall Effect motor speed calculator at the excellent Arduino Playground site.

You can download the "sketch" for the program below.

The rpm_fun function is the interrupt function that will be called whenever the data on pin 2 changes from HIGH to LOW (a FALLING pulse). It updates the global rpmcount, then toggles the status LED.

```
void rpm_fun() {   //Each rotation, this interrupt    //function is run twice, so take    //that into consideration for    //calculating RPM   //Updat
```

Setup initializes the variables, configures the serial parameters, sets the pin modes, and sets up the interrupt function.

```
void setup() {   Serial.begin(9600);   //Interrupt 0 is digital pin 2, so that is where    //the IR detector is connected   //Triggers on FALLING (cha
```

The loop function, as the name implies, is the main processing loop that "runs forever" while the board is powered up. The first statement delays for one second (1000 milliseconds), but note that the interrupt function will break in every time the value of pin 2 changes and run the rpm_fun function. After the 1 second delay, the interrupt is temporarily disabled (this may not be necessary, but seems safer) then the RPM is calculated based on the number of interrupts and the elapsed time between now and the last time the calculation occurred. The result is sent back to the computer over the serial port, then the interrupt is restored.

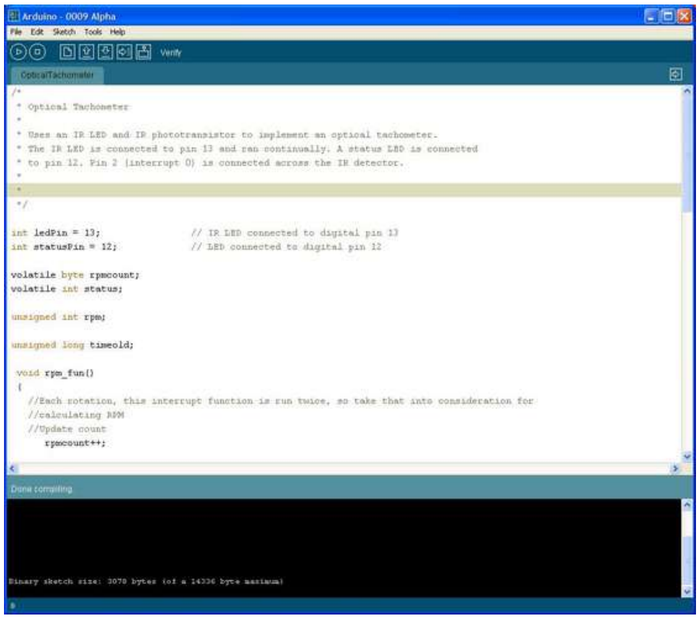
```
void loop() {   //Update RPM every second   delay(1000);   //Don't process interrupts during calculations   detachInterrupt(0);   rpm = 30*1000/(milli
```

Note that the way the motor and the IR detector is configured, each single turn of the coil will result in two transitions, so the calculation takes that into effect. The same would occur for a two bladed fan or propeller. If only one light break per revolution occurred, such as a swinging arm, the calculation would be:

```
  rpm = 60*1000/(millis() - timeold)*rpmcount;
```

For a three bladed fan, the calculation would be:

```
  rpm = 20*1000/(millis() - timeold)*rpmcount;
```



```
/*
 * Optical Tachometer
 *
 * Uses an IR LED and IR phototransistor to implement an optical tachometer.
 * The IR LED is connected to pin 13 and ran continually. A status LED is connected
 * to pin 12. Pin 2 (interrupt 0) is connected across the IR detector.
 *
 *
 */

int ledPin = 13;                // IR LED connected to digital pin 13
int statusPin = 12;             // LED connected to digital pin 12

volatile byte rpmcount;
volatile int status;

unsigned int rpm;

unsigned long timeold;

 void rpm_fun()
 {
   //Each rotation, this interrupt function is run twice, so take that into consideration for
   //calculating RPM
   //Update count
      rpmcount++;
```

```
Binary sketch size: 3078 bytes (of a 14336 byte maximum)
```

File Downloads

OpticalTachometer.pde (1 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'OpticalTachometer.pde']

Step 5: Putting it All Together

The IR LED should be pointed at the IR detector over a gap of a few inches. While you can tape the detector to the table and hold the IR LED in your hand (as I did during testing), this is pretty awkward. What you want to do is to build a makeshift frame that keeps the emitter and detector aligned, but leaves enough space that the motor coil can rotate between them and break the light path with the strip of tape.

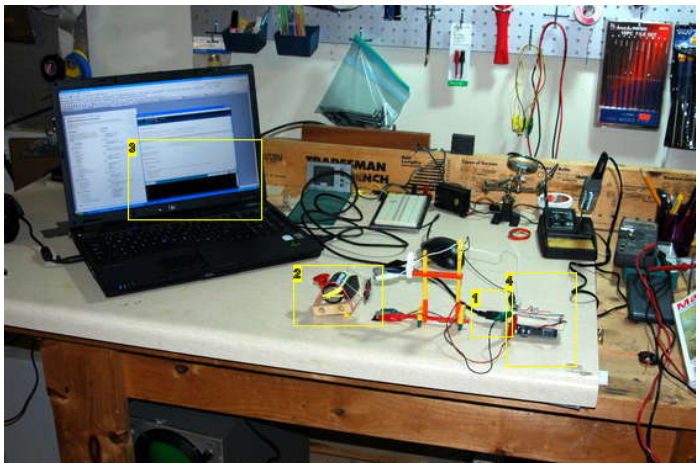
I built a KNex frame where the upper arm can slide up and down and used rubber bands to secure the wires that go to the LED and phototransistor, but you can whatever you can find - LEGO, Popsicle sticks, stiff wire, whatever.

You can test to see if the circuit and program is working by downloading the sketch to the Arudino and using your finger to break the light beam between the LED and photodetector. Every time you break the beam, the status LED should toggle on and off. You can turn on the serial monitor in the Arduino software and you should see a "0" being out every second. If you break the beam a few times, the "0" should change to a small number.

If that is working, you are ready to integrate the motor.

Get the motor spinning with the tape on the coil, make sure the detector circuit is working and the serial monitor is on, then slide the motor in between the LED and detector. If everything is working, the RPM of the motor should show up on the serial monitor on the PC.

The best performance I've got so far was around 1200 RPM. Leave comments if you do better and what you did differently!



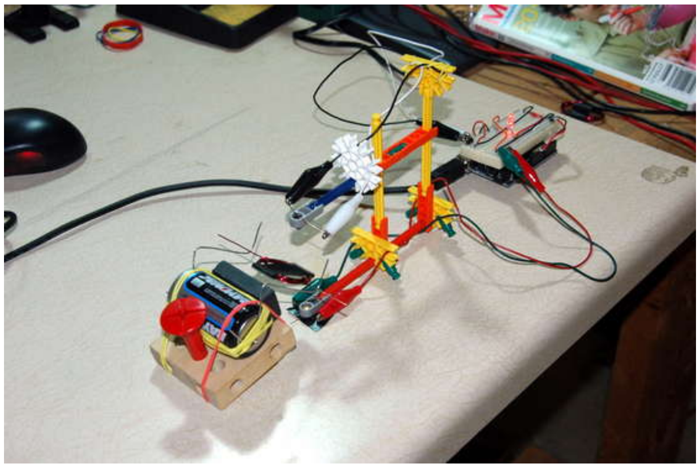
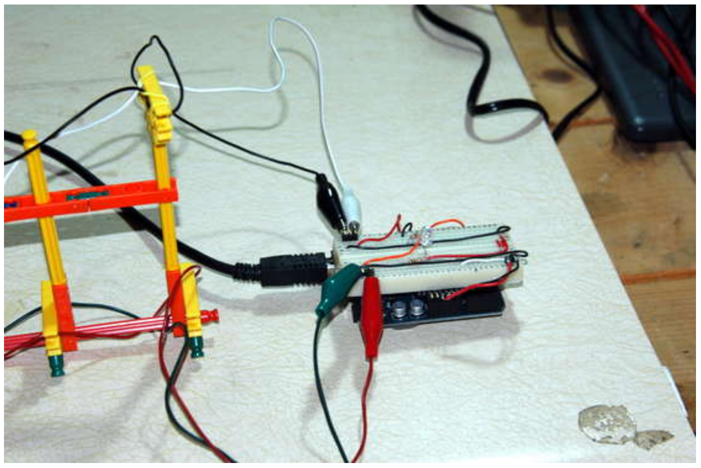


Image Notes
1. USB connection
2. Motor
3. Results
4. Arduino board and proto board.



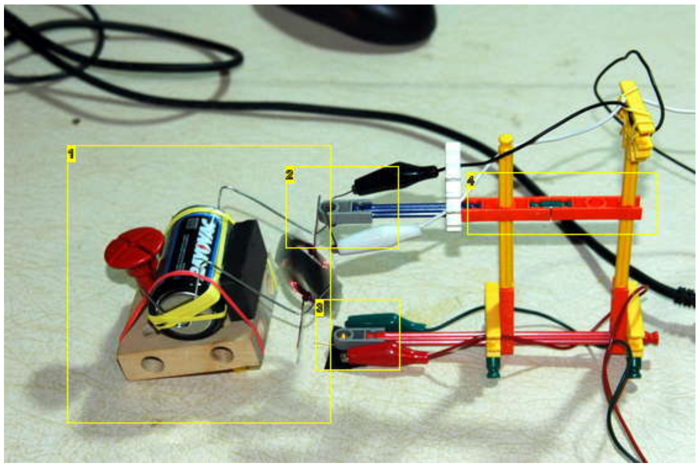


Image Notes
1. Beakman's Motor
2. IR LED
3. IR Detector (Phototransistor)
4. This part of the frame slides up and down to adjust the light beam gap

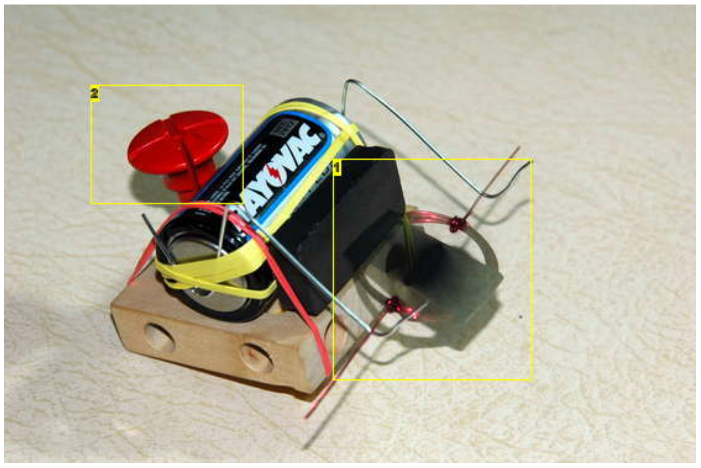


Image Notes
1. Spinning coil
2. This does nothing

Step 6: Future Steps

The whole point of this was to enable people to measure the motor speed of the Beakman's Motor so they could vary details and measure the effect the changes have on the motor speed and I think this succeeds in providing those measurements.

However two improvements come to mind already. The first would be to easily capture the RPM information so that it could be graphed and averaged. Since it is just numbers coming across a serial port, this shouldn't be hard to do in any language.

The second is to use the Processing language to create a graphical tachometer display, either a big analog dial or a digital display with a graph, to show the data instead of just reading numbers off the serial terminal.

Processing Home Page

Any other ideas?

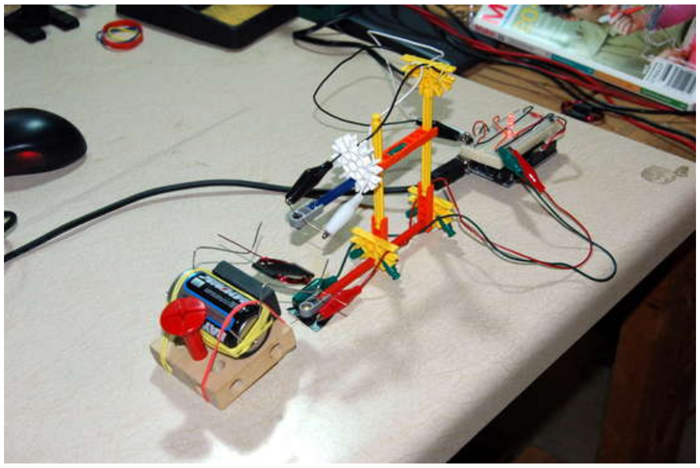
Here are a few more resources on the motor, including some EE lab handouts where the motor is one of their projects:

Fields and Waves Handout

Comments on Motor Design

A few more pictures are available on my Flickr page.

Another introduction to this project (and other random stuff) can be found on my blog, Avoidance Central.

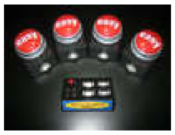


Related Instructables

The RRRRRRRRRRRBE a $3 Arduino by jackzylkin

Autonomous Paintball Sentry Gun by sentryGun53

iAndroidRemote - Control Android mobile using an Apple Remote by sudar


Quiz-O-Tron 3000: Arduino quiz contestant lockout system by RoysterBot

How to have fun with Arduino (and become a Geek in the process) by john otto

Simple Arduino Robotics Platform! by CalcProgrammer1

Comments

25 comments Add Comment

lalder says: Jun 4, 2008. 6:14 AM REPLY
OMG is that the Forest Mimns Book in the back ground? I totally loved them, and learned a bunch from them!

craigbic says: Sep 1, 2010. 2:30 PM REPLY
yes it is! I still have all of his "Engineer's Notebook" series he did for Radio Shack (back when they actually sold cool stuff for electronics hackers). The schematic and explanation of Phototransistors, as I recall (I'll have to dust off my collection) was from his fantastic reference book, Getting Started in Electronics which was also sold through Radio Shack. It's still available and I highly recommend it as well as any of his other electronics reference books to anyone interested in electronics at any skill level. They are terrific references and easy to follow.

RANDOMFISHYFACE says: May 30, 2010. 6:31 PM REPLY
for me when i down load the program it comes up as a tmp file so could you post it replying to this comment

Lenny24 says: Feb 6, 2010. 1:31 AM REPLY
Hello. Great Instrucrable; I need the kind of Script for an RC-Car Engine-RPM-Meter (OMG, Creepy Word ;))
And i Found out, That you can also Use a Reed Sensor for Measuring. Or an Hall-Sensor. I Tried out Both and the Hall sensor worked best for me.

wii552 says: Nov 29, 2009. 6:51 AM REPLY
could this be done with a cds cell/photocell/LDR/photoresistor(all the same thing)

zholy says: Dec 5, 2009. 12:53 PM REPLY
I've tried that ... but it is too slow ... LDR + laser ... you could break the beam very fast, an Arduino would not noticed.

EvilSpeeder says: Sep 18, 2007. 10:28 AM **REPLY**
What would the upper rpm limit of something like this be?

kikiclint says: Nov 24, 2009. 6:24 PM **REPLY**
That would depend on how fast the arduino can run through its loop of code. It also might depend on how fast the sensor can detect a change of light.

zq76 says: Jul 6, 2008. 7:34 PM **REPLY**
Besides the point, but another way to measure the rotor speed is to hook the two paper clips up to an oscilloscope and count the voltage spikes over a period of time.

CMPalmer says: Sep 18, 2007. 11:48 AM **REPLY**
According to this source , the fastest coils have a relatively small diameter with many turns of small gauge wire. My coil in the drawings above had 7 or 8 turns wound around a pill bottle and maxed out at 1200 RPM, but usually ran steadily at around 700-800 RPM. At high speeds, imbalances are magnified and the coil would jump around on the paper clip "brushes" and slow down. Using thinner wire would mean that the coil couldn't the rigid shape of the tails.

One alternative is to allow the coil to sit inside a loop of wire (like the design that uses a large safety pin instead of bent paperclips). This wouldn't ensure proper contact all the time, but would minimize jumping.

What I'm thinking of doing is using two straight, rigid pieces of large, bare copper wire or metal tubing with a non-conductive joint in the middle, then winding a coil with a large number of turns out of very thin magnet wire, then inserting the rod through the width of the coil and soldering the bared ends to the tails. Then use paint on one side of one tail to mimic the half-sanded insulation on the magnet wire. This coil should be heavy, very stable, and have a high electromagnetic density. When I get a chance to build it, I will add it to the Instructable in case these instructions aren't too clear.

I estimate that speeds around 2500 RPM shouldn't be impossible. Of course this motor design is inefficient and doesn't provide much torque, so there is no reason to super-optimize it except for fun.

Other things to remember are (a) use a fresh battery - the motor isn't too kind to them and (b) there is a bit of arcing at the contact points, so periodically you need to wipe off the coil tails and the "brush" points as carbon can build up.

The paper clip arms need to be the same height as well (but they are easily - maybe too easily - adjusted).

I've also thought about gluing small craft beads on the coil tails to keep it from "walking" side to side - something else that will slow it down or stop it.

If you look at the Beakman's Motor page, there are some links to other designs of simple motors and you can find a ton of them via Google (there is a cool brushless motor that uses a magnet reed switch where the motor shaft has permanent magnets attached and it rotates beside an electromagnetic coil - I think I'm going to build one of those some day).

Another good Science Fair type project would be to compare the speed and efficiency of the different types of simple motors.

EvilSpeeder says: Sep 18, 2007. 11:52 AM **REPLY**
I should have been more specific, I meant to ask about the tachometer. Thanks

CMPalmer says: Sep 18, 2007. 12:01 PM **REPLY**
To quote Emily Litella, "Never mind" :-) Someone with a little more knowledge of the ATmega chip in the Arduino would have to answer that - it would all depend on how fast the interrupts can be processed. Taking out the status LED toggle would probably be a good idea for high speed devices. You could also use an offset "flag" on the coil to only break the light beam once per revolution (and use tighter optics or maybe a laser for the emitter/detector). If someone can figure out the maximum measurable speed with this setup, I would like to know as well.

Cairie says: Nov 24, 2009. 5:28 PM **REPLY**
i added a Serial.println and the serial data is in very strange characters. ascii maybe? anyway, it's not in number form. do you have any suggestions on how to make it voltage change in integers? i want to bring it over to Pd and create some sounds.
thank you!

kikiclint says: Nov 24, 2009. 6:22 PM **REPLY**
You could use an op-amp as a comparator, so that the output would only be on, or off. It is really simple. By changing the voltage at the non inverting input voltage, you can change the sensitivity as well. http://web.telia.com/~u85920178/begin/opamp00.htm gives an example of how to do this. You might be able to leave off the negative voltage and have it still work.

DBeta says: Oct 6, 2009. 7:44 PM **REPLY**
Thanks for this. I know this is an old guide, but I just got an arduino and your code example worked perfectly for tapping into the digital tach output on an S2000(A Honda sports car built for racing). I actually didn't have to change it at all! The output of the car worked exactly like the photocell you used. I tested it up to 4,000 RPM, but no higher(just didn't test higher, not sure the limit). I'm working on attempting to get that info cleanly into the computer to be processed by python or the like, but that is just a mater of time.

teddanbren says: Aug 19, 2009. 4:24 PM **REPLY**
I need to be able to calculate angular acceleration (which will vary) for calculations after the is there a way to store each RPM reading into either a spreadsheet or .txt file so it can be analyzed in a graphing program?

robotkid249 says: Jun 9, 2009. 8:09 PM **REPLY**
You don't need a resistor to the led, arudino has them built in.

raykholo says: Jun 21, 2009. 10:43 AM **REPLY**
its only for one pin (like pin 13) and only on some models other pins need a resistor so that if something shorts out in the circuit, some of the output voltage will still be burned off and the board wont fry, then again, each component has a different voltage rating, so resistors might be necesary

sparkyhester says: May 21, 2009. 2:23 PM **REPLY**
Great instructable. I tried this with a small 'flag' on the shaft of a motor that ran through the slot of an opto-transistor. The motor was running at about 1200RPM I think but the figures being produced by the arduino were in the region of 30000 to 50000. Any ideas why?

adrenalynn says: May 16, 2009. 4:06 AM **REPLY**
I concur with sscanf: This won't work as detailed in your "pretty picture" - you need to go back to your hand-drawn schematic on graph paper. As detailed in the first picture the transistor will never see the voltage it needs...

yoghurtsniffer says: Jan 24, 2009. 5:38 AM **REPLY**
Do you think i could use this as a switch so that when i send my ir light signal the receiver will pick it up and read it as a HIGH then perform a sequence or motions etc etc. I had a system setup but it was set to the sony protocol and that meant it could only use my tv remote....but that was abit offt topic. Also i have some IR photodiodes and a few IR receivers lieing around seeing as you suggest 1 only uses to pins of the IR receiver does that mena i can use the Photodiode? (So use a photodiode instead of a receiver?) Thanks :)

sscanf says: Jan 4, 2009. 7:59 PM **REPLY**
I just used this howto as a reference for a similar application using the same IR components and Arduino Diecimila. Everything worked great but note that there is an important difference between the circuit diagrams on this page for connecting the phototransistor. I found that the hand drawn connection shown on the graph paper worked but the other does not. Your mileage may vary.

DavidRobertson says: Nov 5, 2008. 12:45 PM **REPLY**
Cool

os_sanches says: Sep 16, 2007. 5:20 PM **REPLY**
Hi! Can I use this tach to see another rpm measures? the aplication run under windows xp? can you send me this aplication? tks

CMPalmer says: Sep 16, 2007. 7:33 PM **REPLY**
Yes, you can use it for measuring different kinds of rotational motion. See the later steps. The Arduino application runs under Windows XP, but is compiled to object code for the Arduino. All of the source code and links to the application are included in the Instructable.