Arduino Wind Chill Machine

by **msuzuki777** on February 8, 2011

Table of Contents


Author:msuzuki777
Lazy Old Geek

Intro: Arduino Wind Chill Machine

Problem: So I've been taking walks with my dog, Marcus and my Arduino pedometer.
http://www.instructables.com/id/Arduino-Pedometer/
Out here in the high desert in January, it gets cold and a little wind creates a cold wind chill factor. I'm a LAZY OLD GEEK and would like to know what the wind chill factor is.

Solution: I've always wanted to build a weather station so I decided to make an Arduino wind anemometer to measures wind speed and a thermometer. An instrument that measures wind speed is called an anemometer. Then the Arduino can calculate wind chill.

Wikipedia Definition: Wind chill (often popularly called the wind chill factor) is the felt air temperature on exposed skin due to wind. It measures the effect of wind on air temperature.

LAZY OLD GEEK 'facts': Any object, e.g., a car is not affected by wind chill. If the temperature is 10F, it doesn't matter if the wind chill is 10F or -40F. Wind chill is the apparent temperature felt by humans and animals. Or I should say some animals. My dog, Marcus has a nice fur coat and is part husky. He doesn't feel it. He doesn't care if it's -40F wind chill and seems to prefer it. Now, I, on the other hand won't venture out if the wind chill is -40F. But if it's -10F, I might wear my full face mask (see picture) and venture out. I don't like to wear it because it's uncomfortable after a while so it's helpful to know what the wind chill is.

So that's not a great reason to build a wind chill machine but I've always wanted a weather station and this is a good starting point. My weather station is designed for the U.S., so I use Fahrenheit and MPH.

Attention Readers: If you want to duplicate this project, you will need a laptop with a USB port and a motor vehicle to calibrate the wind speed. Another problem is finding the cups for the anemometer. Many Instructables readers seem to be good at improvising. Aluminum is preferred but plastic should work also. For the innovative readers, I've provided some hardware and software tips on how to design your own anemometer. For example, the software could be adapted to an LED or LCD display for a standalone instrument.

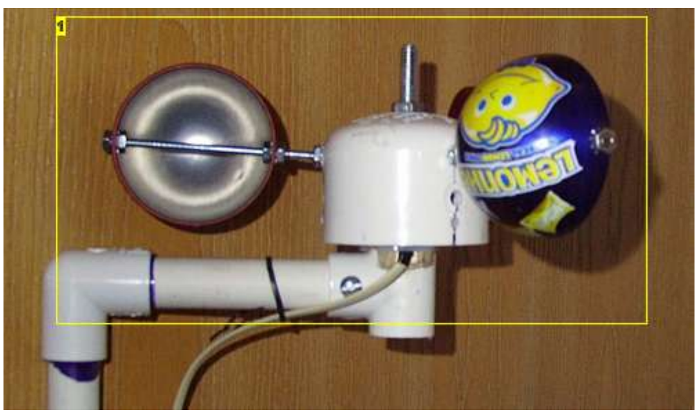


Image Notes
1. My Anemometer

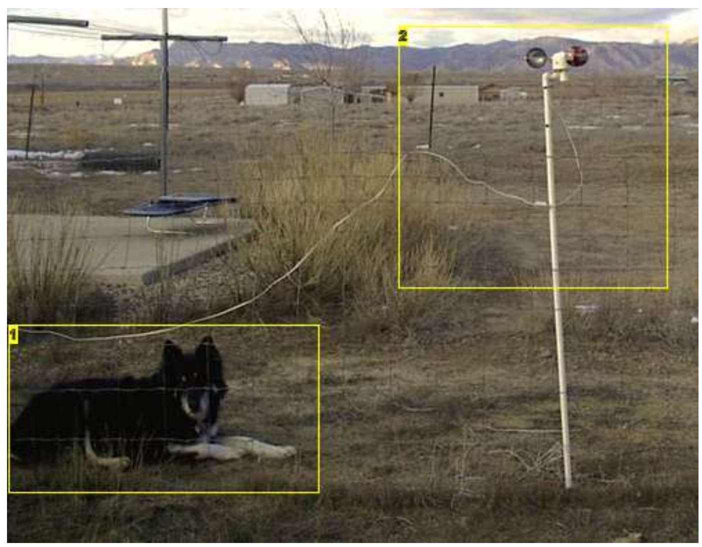


Image Notes
1. My dog, Marcus
2. Anemometer in action

Image Notes
1. Full face ski mask

Step 1: Parts List

Here’s a list of the parts I used:

USB Freeduino kit (Arduino-clone) $22.50
AdaFruit DIY shield $6.00

PVC
1/2” right angle
1/2” T
10 feet 1/2” PVC
2” PVC end cap

Hardware
5-16 x 3” bolt
5-16 nuts and washers
8-32 x 4” bolts
8-32 nuts and lock washers

2 small magnets (10 for $1 at Harbor Freight)
1 Inline skate bearing(Size 608Z)($1.36@)
3 Aluminum balls SuperBubble, LemonHead
Honeywell SS461C Hall Effect IC($1.86 Digikey)
50 feet Telephone cable ($3.00 4 or 6 wire)
1 MCP9701 Temperature sensor($0.30 DigiKey)
2 telephone couplers ($1.00@ dollar store)

Prices US dollars February 2011
The total is about $15 plus Arduino stuff.

Tools:
Soldering tools
DMM Recommended
Drill/drill press
Socket wrenches

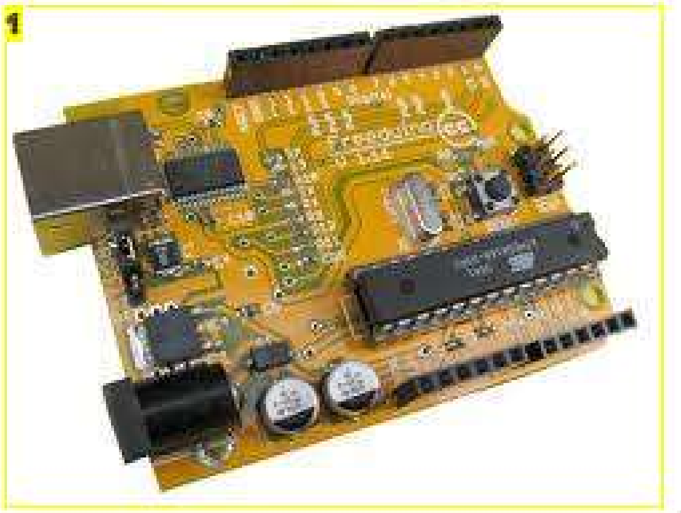


Image Notes
1. Freeduino, an Arduino-clone

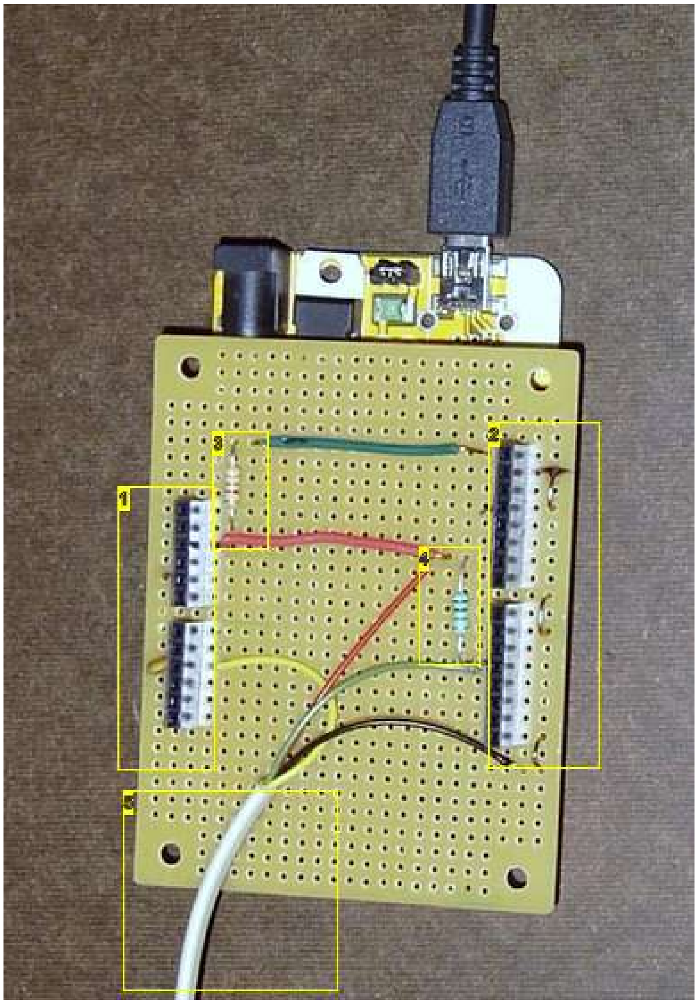


Image Notes
1. Arduino shield pins
2. Arduino shield pins
3. 3.3K resistor
4. 10K resistor
5. Phone cable to anemometer sensors

Image Notes
1. In line skate bearing, size 608Z

Image Notes
1. Candy balls, made of aluminum

Image Notes
1. Honeywell Hall-effect sensor, SS461C
2. Pin out of SS461C

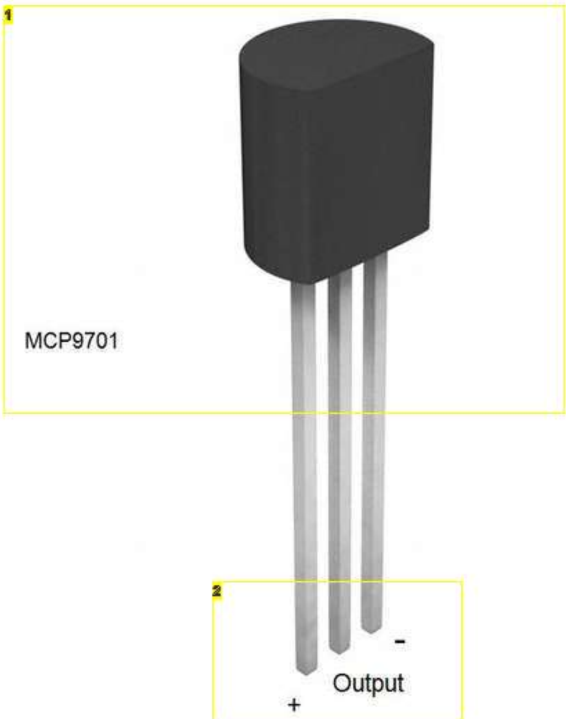


Image Notes
1. Temperature sensor, MCP9701
2. Pin out of MCP9701

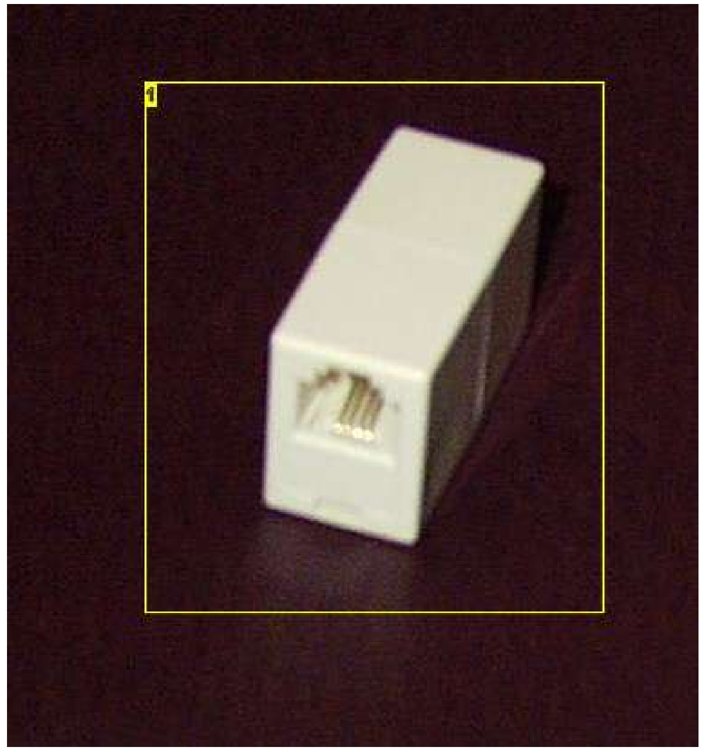


Image Notes
1. Phone coupler, make sure there is four or six connectors

Step 2: Selecting Parts

Problem: For me, the hardest part was finding some cups to use to collect the wind.
Solution: One suggestion was plastic Easter eggs. Another was metal soup ladles. Plastic is fragile and deteriorates in heat. Soup ladles would be hard to work with. I looked around and asked around. My cousin came up with these aluminum balls that had LemonHead\AtomicFireball candy in them. They're about 2.7" in diameter. Unfortunately, this may have been a Christmas thing. My cousin bought them at a Rite Aid. They're made by ferrapan.com. You may have to find a substitute.

Problem: Next I wanted a bearing to make it spin easier.
Solution: One of the easiest to get bearings is for inline skates. The most common size is 608Z. One option is to search your local thrift stores to find a used pair of inline skates and pull out the bearings. I just bought mine from Amazon.com.

Problem: Being LAZY and not too mechanically inclined, I needed a way to attach the bearing.
Solution: One internet site used two skate bearing and scrapped out the insides of a ½" PVC 'T' for fit. I actually tried two bearing in a 'T' but decided there was too much friction so cut one off. Instead of having to scrape the insides I just took a bolt and some big washers and a bearing on one side of the T. Make sure the washers are bigger than the outside of the bearing. Roughly center the assembly, then tighten the nut to pull the bearing into the 'T' (See picture).

Image Notes
1. Candy balls, made of aluminum

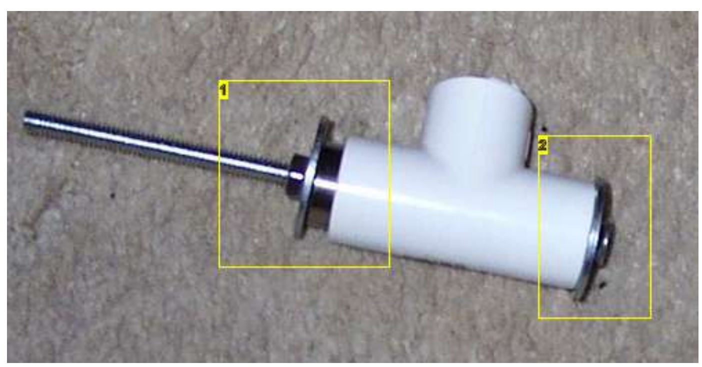


Image Notes
1. Nut, big washer and skate bearing
2. Big washer and bolt head

Step 3: Assembling the Anemometer

My design is based on a 2” PVC end cap. Some internet searching suggested that three cups worked better than four. Probably either would work okay but I decided on three.

Procedure:
The first step is to mark the end cap for drilling the center and the three arms, 120 degrees apart. And two holes, 180 degrees across for the two magnets.

I recently got a drill press from Harbor Freight.
Safety: When using a drill press, it is advisable to use safety goggles and clamp your work to the table.

I didn’t use the all of the following but think it would work pretty well.
Place the end cap on a piece of paper and drew around it (see picture). By taking the diameter and dividing by two you can find the center of the circle. Then I used a compass to mark out the 120 degree points and also straight across for the two magnets. Then I placed the end cap back on the paper and marked the bottom of the end cap with a Sharpie. Since I had a laser line, I traced the line up the end cap. I marked the arm points up about 1.4”. I clamped the end cap and drilled 5/32” holes for the arms.

Finding the center: I made a little V bracket (see picture). Put in a 5/16” drill bit since I used 5-16 /bolts. Place the paper template under the V bracket aligning the edge of the circle with the edges of the V bracket. Fold or cut to make it fit under the bolts. Adjust the V bracket and template so that the drill tip is over the center of the paper circle(see picture). Place the end cap in the V and marked the ‘center’ with a Sharpie. Now if you rotate the end cap, the center should stay directly under the drill bit. Keep fiddling with the bracket and the mark until the drill tip stays centered. Clamp and drill (see picture, the clamps are not shown). I assume the more mechanically adept readers have a better solution but this was adequate.

Image Notes
1. 2'PVC end cap

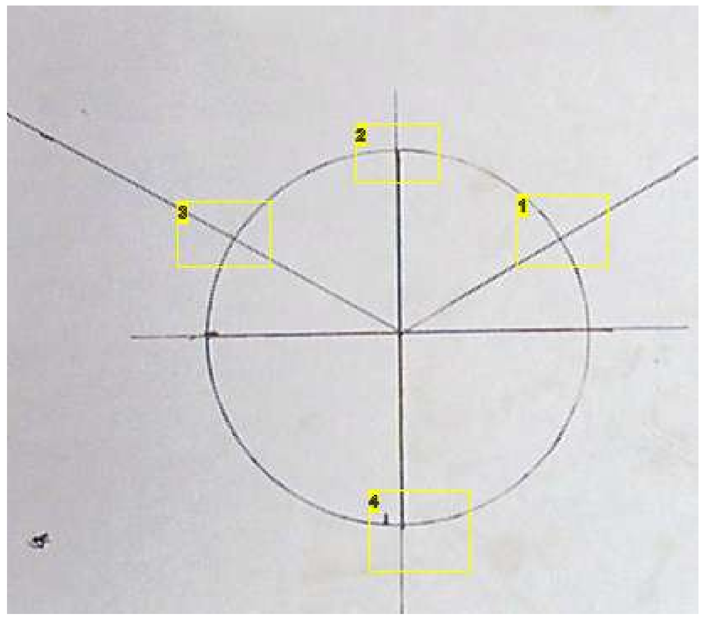


Image Notes
1. 120 location
2. One magnet location
3. 120 location
4. 120 location and other magnet

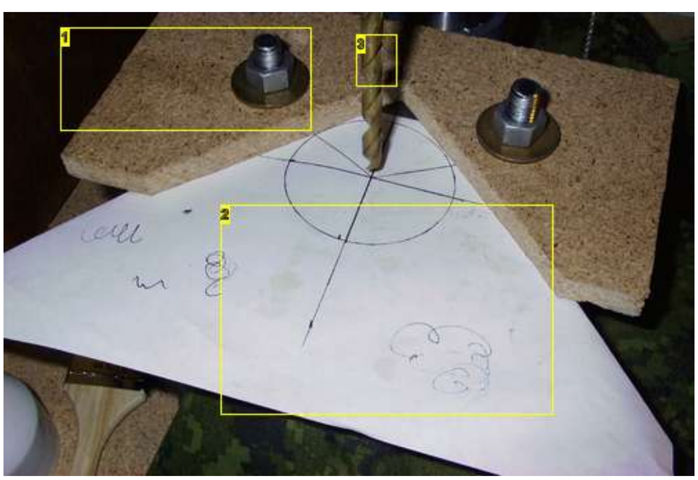


Image Notes
1. Custom made V bracket
2. Paper template
3. 5/16" drill

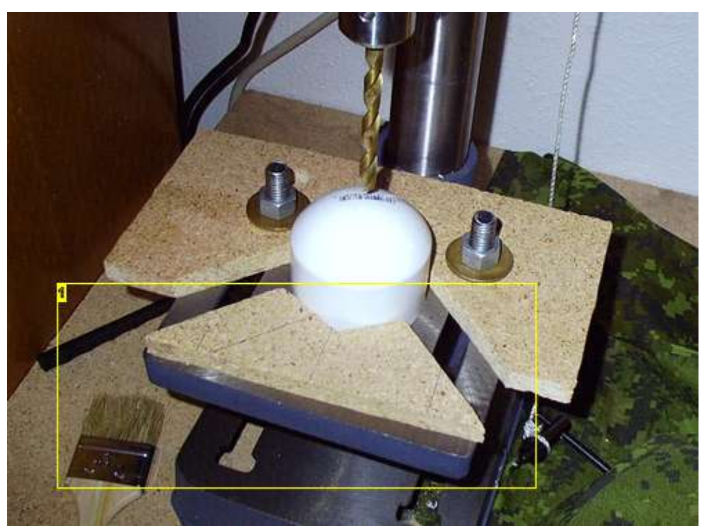


Image Notes
1. Another custom made bracket for clamping end cap

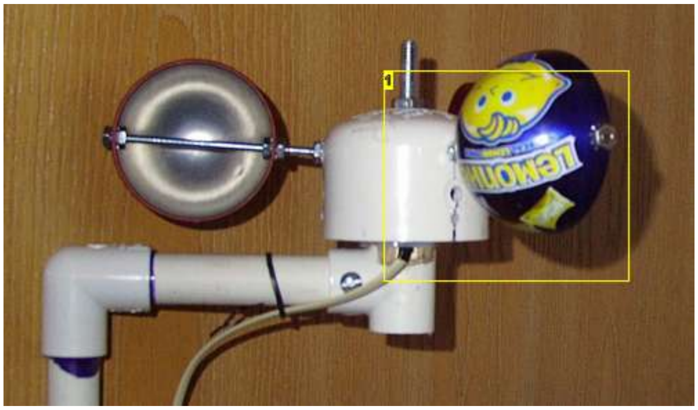


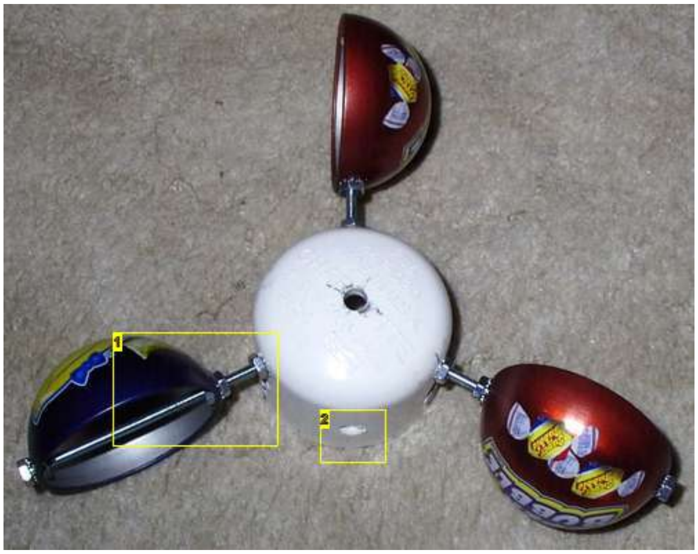
Image Notes
1. 120 marker line for one cup and one magnet

Step 4: Attaching the cups

Procedure:
I used a blue cup that was a different color from two red ones. This is so it's easier to see when the assembly makes one revolution. Using similar techniques from above, mark the cups 180 degrees across about ¼" from the rim for the mounting holes. Drill 5/32" holes. The metal around the hole may deform a little with the drill.
I couldn't find 8-32 x 4" bolts so I bought a 12" piece of 8-32 threaded stock and cut it into three pieces. Using four 8-32 nuts and lock washers, attach the cups to the threaded stock. When you tighten it make sure the cups are still fairly round. With two more nuts and lock washers, attach the cups to the end cap. The holes are pretty snug, so you may actually have to screw the cup assembly into the end cap. You might want to use Loctite Threadlocker.

For easier testing and usage, I made an L assembly out of ½" PVC and glued it together. To attach the T, I didn't glue it. I drilled a hole for an 8-32 bolt and nut so that the anemometer can be replaced easier (see picture).

Put a little oil on the bearing. Put the 5-16 x 3" bolt through the bottom of the PVC T and the bearing. Put on a small 5-16 washer and two 5-16 nuts. Tighten the lower nut so that the bolt is tight on the bearing. Hold that nut with a wrench and tighten the second nut to lock the two together. The bolt will have a little bit of play because of the bearing (see picture).
Add two more nuts and a washer to the bolt. Adjust to approximately where you want the cup assembly to ride.
Slip the cup assembly on the bolt and give it a spin. Make sure the cups and end cap clears the PVC T. When you're happy with the height take off the cups and tighten the two nuts together so they won't move. You can keep the bolt from turning by using a ½" socket with an extender sticking through the bottom of the T on the bolt.



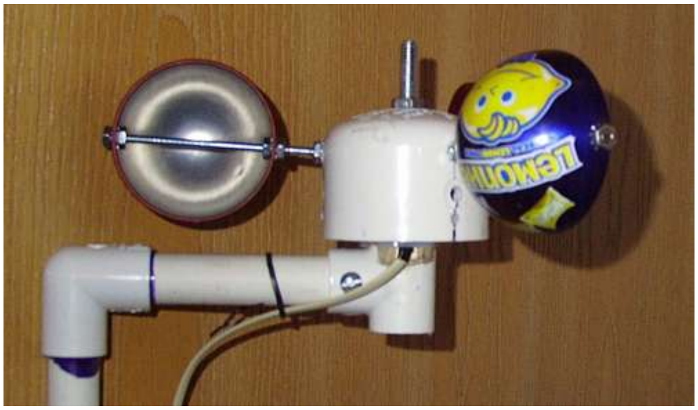
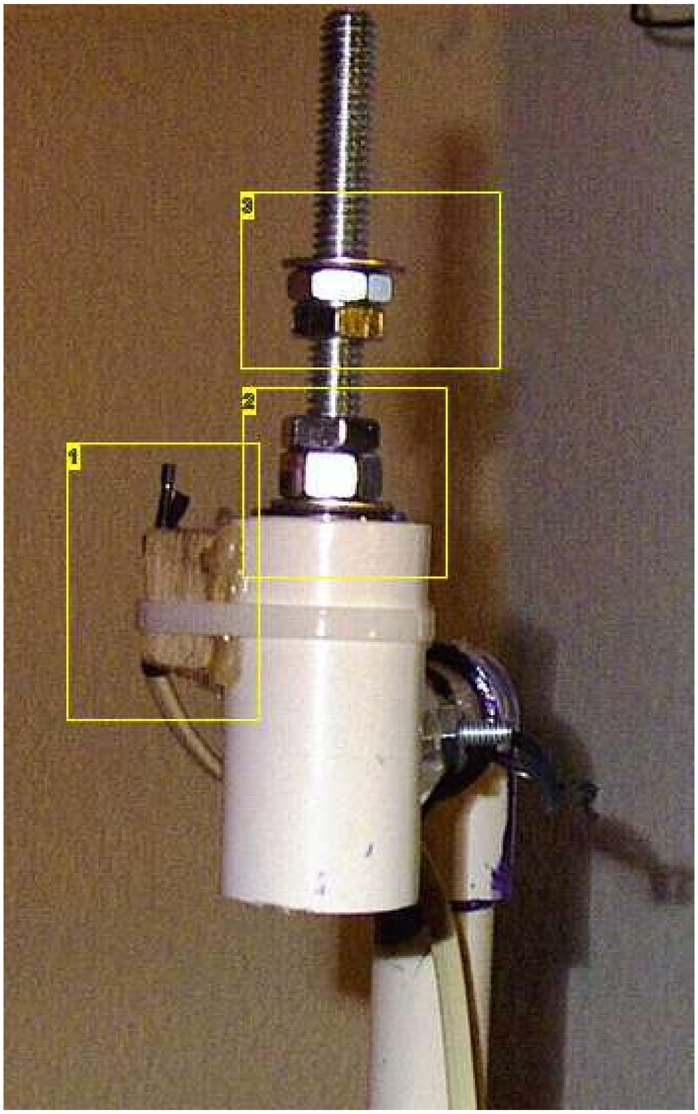


Image Notes
1. 8-32 hardware to attach the cups
2. The other magnet



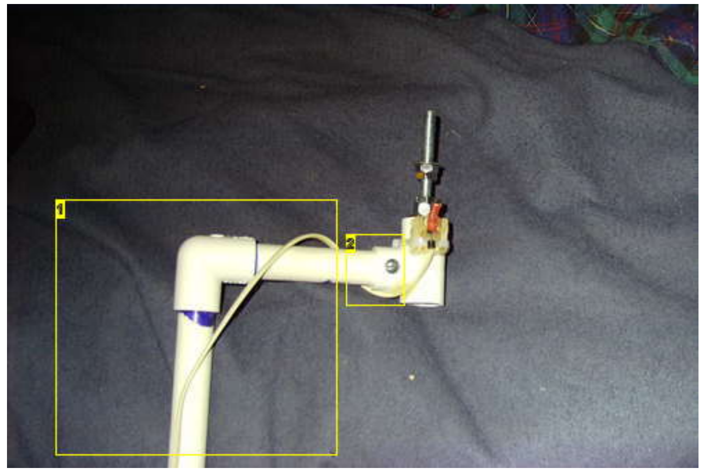


Image Notes
1. PVC L assembly, glued
2. 8-32 bolt and nut, not glued

Image Notes
1. Wood block
2. Bearing, washer and nuts
3. Nuts to position end cap

Step 5: Hall-effect and Temperature sensors

Theory of Operation:
I hadn't used these Hall-effect sensors before so I had to do some experimenting. The Honeywell SS461C is a bipolar latching sensor. As many of you know magnets have a North and a South pole. The SS461 is connected to 5Vdc and ground and has a digital output. One magnet pole latches the output low and the other pole latches it back high. For this application, you don't care which one is North and which is South, just as long as they alternate. One of the concerns is how close the magnet has to be to trigger the device. With the magnets, I have, it's about ½", but closer works fine.

Wiring SS461C: I used a short 4 wire telephone cable with a standard phone connector on the end. Connect the SS461C + pin to the red wire. Connect the – pin to the black wire. Connect the output to the green wire. If you don't use phone connectors, just wire according to the schematic.

Wiring MCP9701 Temperature sensor: I placed this right below the SS461C. Connect the + pin to red wire of the same phone cable and the – pin to black wire. The output pin goes to the yellow wire. I then wrapped the connections with electrical tape and hot glued it. (See schematic)

Problem: So you may ask why did I put the sensors on a cable with a phone connector? The anemometer has to be calibrated. I calibrated it by setting it up in my car. Once calibrated, it will need to be setup outside of my house and connected to the Arduino inside the house. How do I accommodate these different configurations?
Solution: So I put a short phone cord with connector on the sensors and another one on the Arduino shield. Then I put phone couplers on both of these connectors and put a standard phone extension cable between. For the car calibration, I used a short six foot extension. For the house, I used a 50 foot extension with one connector passing through my exterior wall.

If you try this, four cautions:
1. Make sure the couplers and cables are four or six wires. There are some two wire cables that won't work.
2. Be very careful that you use the same couplers for calibration and usage. Some couplers go straight through and some cross over pins. It really doesn't matter which ones you use, just as long as you always use the same ones.
3. Make sure the two different extender cords work the same. All of the cords I have seen cross over where pin 1 on one end is not pin 1 on the other. I know this is complicated but there is an easy solution coming up.
4. Make sure the sensors are connected to the correct places on the shield. (See schematic) The four connectors on the outsides are the shield to Arduino connectors. I used an Adafruit DIY shield. The SS461C +, - and Output pins are connected as shown.
The red wire from the sensors needs to get to +5V on the shield, the black wire goes to a ground. The green wire from the SS461C hall-effect sensor output goes to Digital 4. The yellow wire from the MCP9701 temperature sensor output goes to Analog 2 on the Arduino/shield. On the shield connect a resistor (10K works) from 5Vdc

to Digital 4. Also connect a 3.3K resistor between 3.3V and AREF (explained later).

Easy solution: Connect everything up without power to the Arduino with the short extender cable. Take your DMM, set it for ohms and measure from the +5 on the sensors to the +5 on the Arduino/shield, it should be shorted. Do the same for the grounds. Measure from the SS461C hall-effect sensor output to D4 on the shield and measure from the MCP9701 temperature sensor output to A4. These should all be shorted.
Then, switch the phone extender cable to the long one and repeat. If all are correct, then you are good to go.

If you have a DMM but don't know how to use it, I wrote a little tutorial in:
http://www.instructables.com/id/Arudino-No-Blinky/

Attaching magnets:
The magnets I have are 5/16" diameter so I clamped and drilled two 5/16" holes in the end cap 180 degree apart about 0.7" up from the flat. Now if you take two magnets and they couple together, the two inner surfaces are opposite poles by the rule of opposites attract. One is North and one is South. It doesn't matter which one is which. So if you separate the magnets and put them into the two holes so that one North and one South is facing inward, you are good to go. Mine fit pretty snugly but I added hot glue to hold them in. (I also filled some extra holes in my end cap with hot glue.

Image Notes
1. Honeywell Hall-effect sensor, SS461C
2. Pin out of SS461C

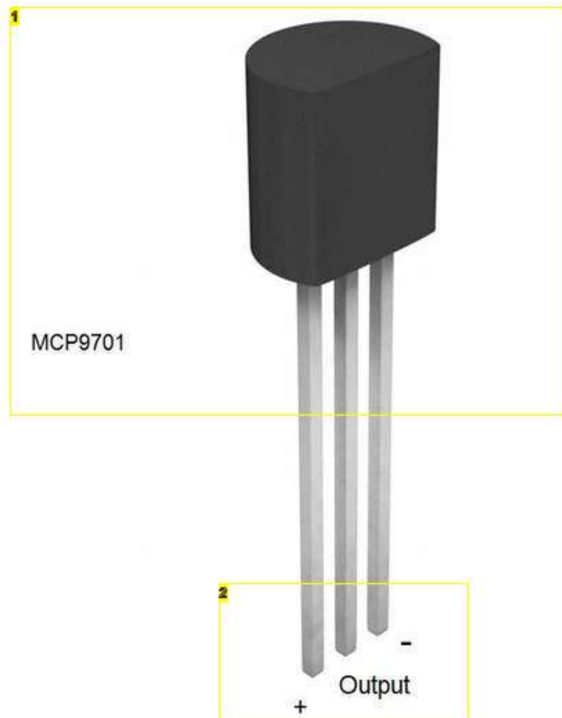


Image Notes
1. Temperature sensor, MCP9701
2. Pin out of MCP9701

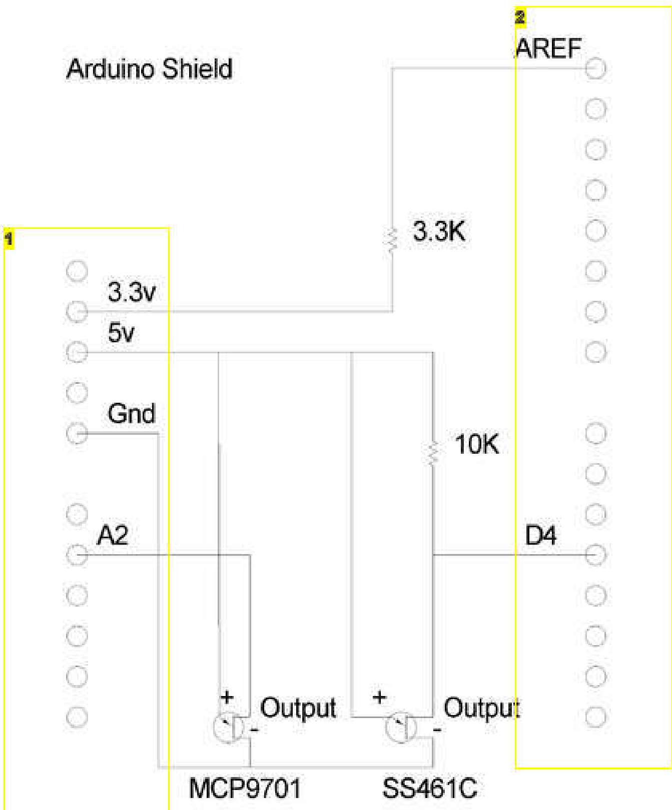


Image Notes
1. Arduino shield pins
2. Arduino shield pins

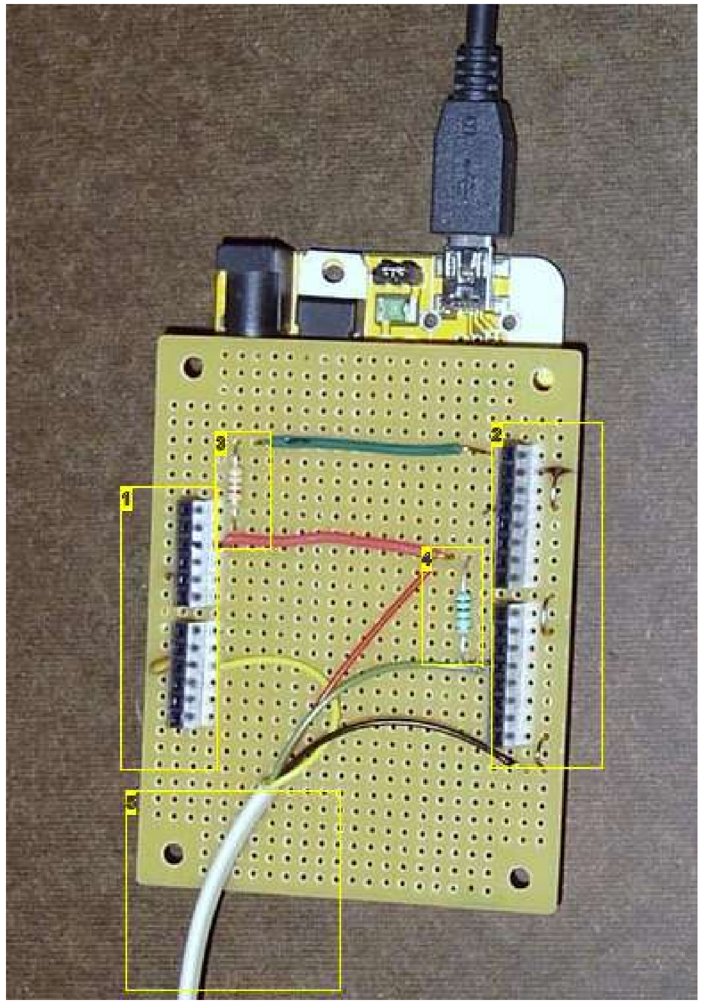


Image Notes
1. Arduino shield pins
2. Arduino shield pins
3. 3.3K resistor
4. 10K resistor
5. Phone cable to anemometer sensors

Step 6: Mounting the switch and temp sensor

I carved a little block of wood to fit the curve of the 1/2" T about 0.3" deep. I hot-glued the block to the T near the bearing. I carved a little groove in the block for the sensors. Position the SS461C so that it's even with where the magnets will pass but doesn't touch. You can use the little wire wraps while you get it adjusted then I used a plastic tie wrap. (See picture)

Testing: Plug in the Arduino either to USB or a 5Vdc supply. If you have a DMM, attach it to ground and Digital 4. With the cup assembly on and spinning, the voltage should toggle from 5Vdc to 0V, once per revolution.

Problem: When I was testing the assembly, the Hall switch didn't always work.
Solution: The Arduinos, Arduino clones and shields are made by different manufacturers of differing qualities. I tend to buy the cheapest, hence the Freeduino clone. All of these manufacturers use different quality and types of components. My AdaFruit shield pins didn't seem to fit snugly in the Freeduino. This could be because of slightly different size pins and headers and/or the header pins wearing out from use. What I did was take a pair of needle nose pliers and slightly bend the shield pins a little this way and that so they made good contact with the Freeduino. This is not an ideal solution but adequate for my needs.

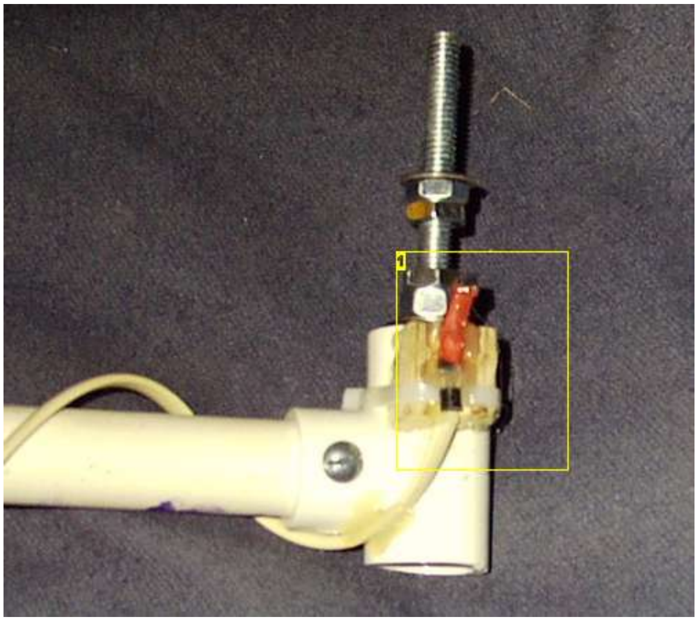


Image Notes
1. SS461C on top, MCP9701 below

Step 7: Software Design and Calibration

Software Design:

The attached Arduino software(MTSAnemometer.zip) has to look for High/Low transitions on Digital 4, count them and then log it to the serial port of the computer. The software will determine how often to log data to the SD card. This software is set for 2 second samples. This can be changed by changing the value of SampInterval.

Here is the significant portions of the software loop.

```
// Wind calculator Look for High
if (digitalRead(WindPin)==HIGH)
{
 SeeHigh=true;
 digitalWrite(greenLEDpin, LOW);
}
 //Look for Low thus a High to Low transition
 else if (SeeHigh==true)
{
 //Increment counter
 CntRPM++;
 SeeHigh=false; //Start over
 digitalWrite(greenLEDpin,HIGH);
}


//After two seconds have elapsed
RPM=CntRPM*(60.0/SampInterval);
```

Code explanation: First I look for a high on the anemometer Digital 4 (Windpin). When it goes high, I set SeeHigh true. For troubleshooting purposes, I turn off a green LED. Then when Digital 4 goes low and SeeHigh has been set then I know that the anemometer had transitioned from high to low and one revolution (RPM) has occurred. So I increment the CntRPM and also turn on the green LED so that I can tell that it is working.
After two seconds have elapsed, I multiply CntRPM by 30 to get RPM (revolutions per minute).

Calibration procedure:

Problem: So how do you determine how anemometer RPM relates to MPH? There are several solutions depending on what equipment you have available. I will list three. All of these require that you have a way of displaying RPMs coming from the Arduino. My Arduino code will display data to a computer/laptop connected to the Arduino with USB. If you don’t have a laptop, one alternative would be to rewrite the code to display data on an LCD shield.

Hardware requirements: Your laptop will need a USB port or two. My old laptop only has only one USB 1.0 port but works fine with a USB hub.
Arduino Requirements: Install my Arduino code into the Arduino. Connect the Arduino to the laptop USB port.
Anemometer Setup: I stuck the anemometer assembly in the opened window of my car’s back seat door. The bottom of the PVC was pinched between the seat and the door. Or you might be able to put it in the door slot that you close the door with. My seats have overhead handles for all seats to help get in and out. I took a string, looped it around the handle and the PVC to keep the anemometer upright. If the PVC is touching some vulnerable part of the car, put something between like a soft cloth. Carefully close the window partially. (See picture) Plug the sensors into the Arduino.

Solution1: If you have a laptop and a car, run the software with serial terminal at 9600 baud. (See sample output screen)
Safety first: Ideally, this should be done with a second person to help. On a windless day, pick a straight, flat, deserted road and set cruise control for a known MPH. One typical problem is that most speedometers are off. You can check this by finding some highway mile markers and finding out how long it actually takes to go a mile at some supposed speed.
See what RPM on the Arduino serial terminal is. (The serial terminal also displays MPH and Twc but these values are based on my anemometer calibration.) Take the car’s known MPH and divide it by the RPM and plug it into the Arduino code. Here’s the Arduino code. Replace .054 with your calculation.

```
MPH=RPM*.054; //Calibration value
```

Solution2: If you have a laptop and a car and a standalone GPS, run the software with a serial terminal.
Safety first: Ideally, this should be done with a second person to help. On a windless day, pick a straight, flat, deserted road and set cruise control. Take the MPH from your GPS and divide it by the RPM and plug it into the Arduino code as above. By the way, you can also determine how far off your car speedometer is. If your GPS reads in knots per hour, multiply it by 1.151 to get MPH.

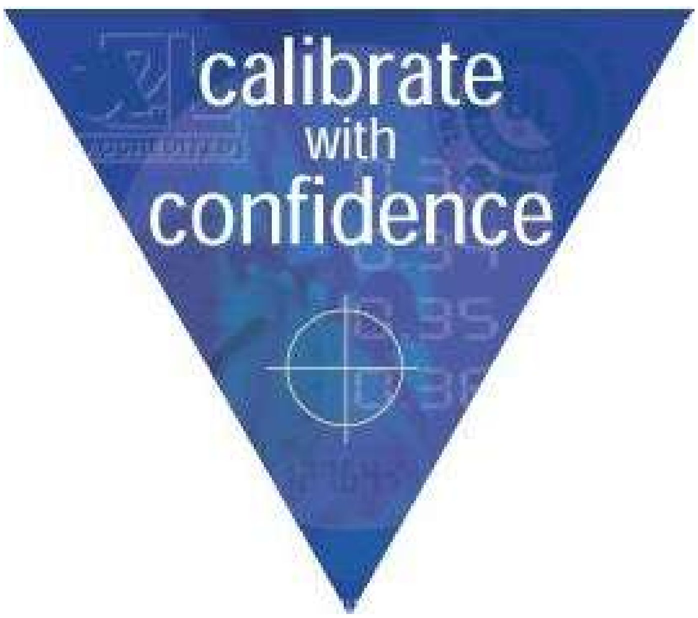
Solution3: I have a laptop and a car and a NMEA GPS that connects to my laptop USB. I connected the Arduino and the GPS to my laptop, and ran my VB Express software. See attached. This program should work with any Arduino and a USB NMEA-compatible GPS. Mine is a USGlobalSat BU-353.

To install my calibration software, copy cpublish.zip to a directory on the laptop, unzip it, make sure you have an Internet connection and run setup.
You'll also have to install the USB driver for the GPS.
My program setup requires the USB/com port numbers for the GPS and the Arduino. These are listed under Device Manager, under Ports. My GPS is called Prolific. My Arduino comes up as USB Serial Port.
If you have some trouble, my GPS Instructable has some more tips.
http://www.instructables.com/id/GPS-for-Lazy-Old-Geeks/

Start my program (see picture), plug in the correct com port numbers. Click the UPDATE button. You should see the GPS data displaying and the Arduino data at the bottom.
Safety first: Ideally, this should be done with a second person to help. On a windless day, pick a straight, flat, deserted road and set cruise control. Now my program is automated so it gets the MPH (converted from NMEA knots per hour) from the GPS and the RPM from the Arduino. I actually, tried several different speeds. Write down the times when you were driving with cruise control set.

Go back to my program, click on File Save (ignore any messages). Type in a filename and note the file location. My program will save the data to a .CSV file so I plotted the data to an Excel chart (see picture). This shows a fairly linear relationship. Take a sample MPH from your GPS and divide it by the RPM and plug it into the Arduino code as above. I used this to determine the .054 multiplier shown in the program.
If you don't have Excel, you can also use free Google Docs or Open Office. The plotted graph of some different MPHs looks pretty ragged but it is fairly linear. (see graph)

To verify your calibration, load the Arduino with your calibration and rerun the above tests. The Arduino MPH should be close to 'actual' MPH but will probably jump above and below it.



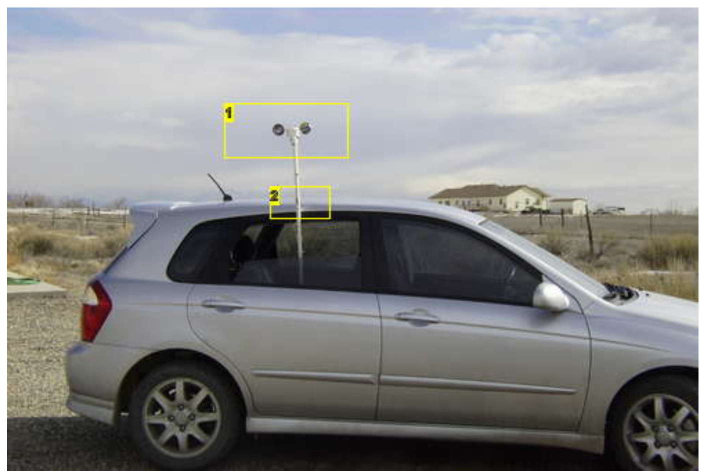


Image Notes
1. Anemometer
2. String to hold assembly

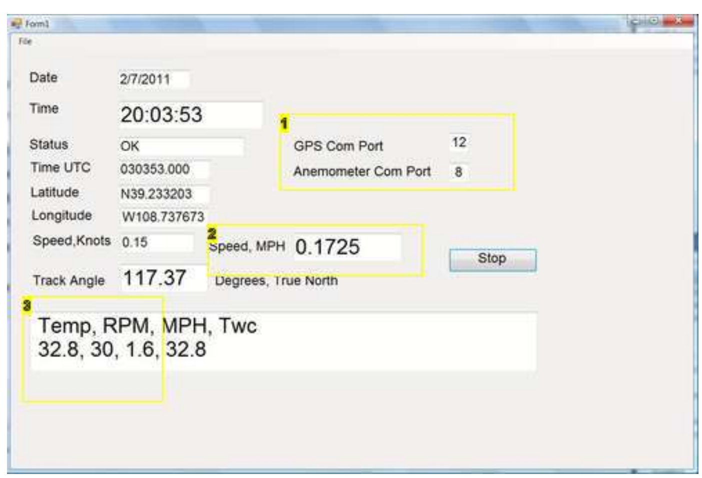
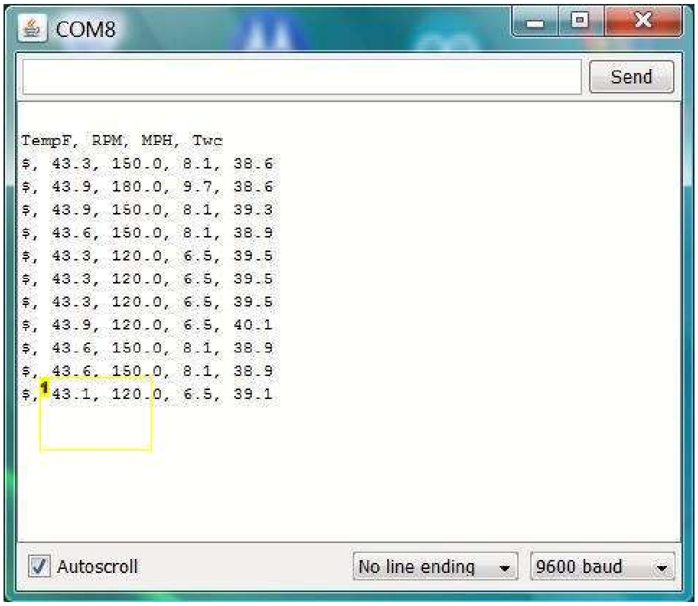
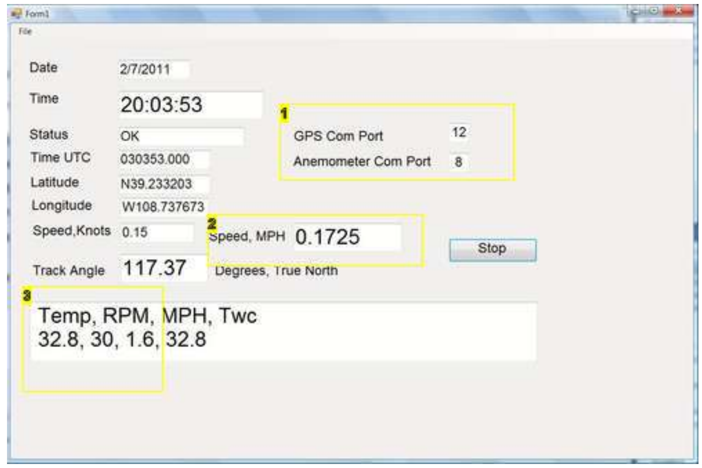


Image Notes
1. Com port numbers must be entered
2. Speed taken from GPS





3. Temperature and RPM from Arduino and Anemometer

Image Notes
1. Temp and RPM, the rest are not accurate until calibrated

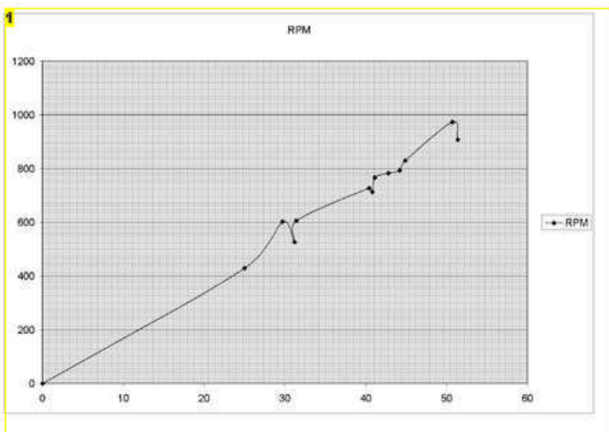


Image Notes
1. RPM and MPH are actually fairly linear

File Downloads

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

Cpublish.zip (213 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Cpublish.zip']

Step 8: Using MCP9701 Temperature sensor

The MCP9701 puts out an analog voltage related to the temperature.
Problem: The maximum output voltage is 3.0 volts and the standard Arduino uses the 5 volt supply often from the computer/laptop USB. This voltage can vary causing the readings to vary.

Solution: To improve accuracy, I decided to use the 3.3Vdc from the Arduino connected to the Arduino AREF pin through a 3.3K resistor. The 3.3Vdc source is much more precise than the 5Vdc. For the techno-geeks, Internal resistance is 32K so AREF = 3.3/(32K/(32K+3.3K) =3.0 Vdc.
To use the AREF pin, the following line was added to the Arduino code.

```
analogReference(EXTERNAL);
```

Software Design:
Here is the essential code:

```
// Get temperature readings Average nsamps
for (byte j=0;j<nsamp;j++)
{
  ThermValue += analogRead(ThermPin);
}
  ThermValueAvg=ThermValue/nsamp;

  mVout=(float) ThermValueAvg*3000.0/1023.0; //3.0V = 3000mV
  //TempC=(mVout-400.0)/19.5; //Ta = (Vout-400mV)/19.5mV //Original
  TempC=(mVout-490.0)/19.5; //Ta = (Vout-400mV)/19.5mV //Modified
  TempF=TempC*(9.0/5.0)+32;
```

Code explanation: The analog 2 ThermPin is sampled ten times then averaged. The averaged analog count is converted to mVolts(mVout). The mVout is converted to TempC (Centigrade). The first TempC calculation is commented out. This is the suggested conversion. For the MCP9701 I was using, I found that this was off about 8 degrees, so I adjusted (trial and error) the formula to the second formula. You may want to adjust the 490 value, if you're using an MCP9701 and find that it is not reporting the correct temperature. Then next line converts TempC to TempF (Fahrenheit).

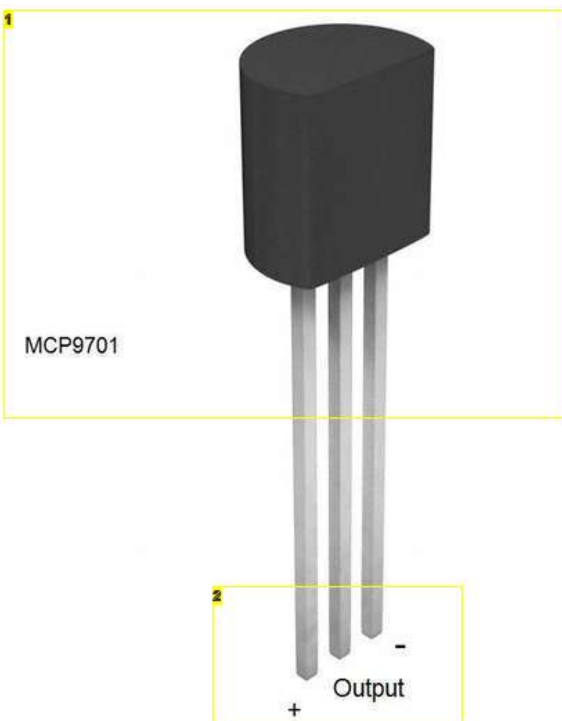


Image Notes
1. Temperature sensor, MCP9701
2. Pin out of MCP9701

Step 9: Wind Chill

There are a few variations on calculating Wind Chill. This is the one chosen by the National Weather Service.
http://en.wikipedia.org/wiki/Wind_chill

The equivalent formula in US customary units in computer terms:

Twc=35.74+0.6215*Tf-35.75MPH^0.16 +0.4275*Tf*MPH^0.16

Twc is wind chill in Fahrenheit
Tf is temperature in Fahrenheit
MPH is wind speed in MPH

Wind chill Temperature is only defined for temperatures at or below 10 °C (50 °F) and wind speeds above 4.8 kilometres per hour (3.0 mph).

So I incorporated all of this into the Arduino code:

```
if ((TempF <50.0) && (MPH > 3.0))
{
  Twc=35.74+0.6215*TempF-
      35.75*pow(MPH,0.16)+0.4275*TempF*pow(MPH,0.16);
}
else
{
  Twc=TempF;
}
```

Step 10: Using Anemometer

The Arduino can be connected with USB to a computer and the temp, wind speed and wind chill will be displayed on the serial monitor. So if you plugged in your calibration number and adjusted temperature offset if needed, then your wind speed and wind chill should be correct.

GEEK comments: The picture below shows the anemometer stuck in the fence. It's about 50 feet from my house. This is not ideal. If the wind is coming from the west, my house will block the wind and skew the data. The ideal spot would be on my roof so the wind wouldn't be blocked by any buildings. For now, I want to see how it works over some time. My next project will be to add a weather vane to indicate wind direction and it will likely be attached to the same pole.

One concern, I have is if the temperature sensor will be affected if it's in the hot sun. I have another thermometer that is on the north side of my house always out of the sun so I can do some comparisons.
Also, I need to tie down the phone lines better and insulate the connectors from rain and snow. But then I am LAZY and it's cold out there.

In the works, I am also writing a VB Express program to collect the data so that I can find highs and lows. I've already noticed that the data has anomalies such as temperatures that are way too high or low. Outside temperature does not have sudden jumps so there are some problems somewhere in the path. It could be hardware and/or software.

Conclusion: So as it is, this system gives me a pretty good indication of wind chill when we start our early morning walks and gives me a clue on how to dress and whether to go out now or wait for less wind chill.

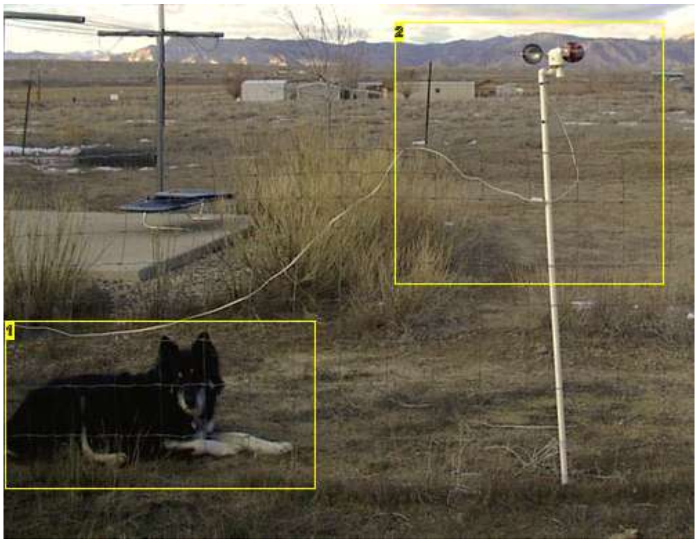
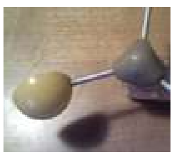
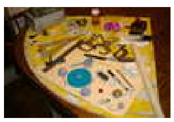


Image Notes
1. My dog, Marcus
2. Anemometer in action

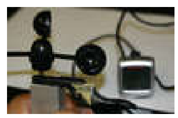
Related Instructables

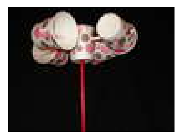

Make it yourself anemometer for under $30 by pashanoid


Easter Egg Anemometer (Wind Speed Meter) by robbtoberfest


Anemometer from CDROM motor, and plastic Easter egg halves by outlawmws

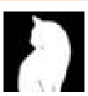

Anemometer by mrigsby

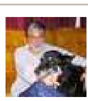

Digital Anemometer (wind meter) by dan


Paper Cup Anemometer by nkrauel

Comments

4 comments **Add Comment**

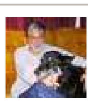

ChrysN says: Feb 10, 2011. 6:55 PM **REPLY**
Cool project!


msuzuki777 says: Feb 18, 2011. 3:27 PM **REPLY**
Thanks, I enjoyed making it.

Lazy Old Geek


rimar2000 says: Feb 10, 2011. 7:43 PM **REPLY**
¡Good work!

Here in Argentina the summer has been a little hot. 43º of thermic sensation in my zone, that is not one of the hotter. And the meteorological service threaten us with more heat during frebruary!


msuzuki777 says: Feb 18, 2011. 3:26 PM **REPLY**
Sorry, my response is so slow. My ISP won't connect to Instructables.com so I'm can't connect unless I use my cousins.

There's actually another thing called heat index which is the counter part of wind chill. I was thinking about adding that feature but our summers are pretty dry so don't know if it would be worth it.

Lazy Old Geek