

Starting Electronics Needs Your Help!

It is that time of the year when we need to pay for web hosting and buy new components and equipment for new tutorials. You can help by making a donation. Contribute to this website by clicking the **Donate** button. The total will be updated once daily. (You may need to clear your browser cache to see the updates.)

Donate



Target Amount: \$2000

Amount Raised: \$688

Thanks M.B. for your \$5 donation

Top Donor: C.C. \$100

X

[Home](#)

[Beginners](#)

[Projects](#)

[Tutorials](#)

[Articles](#)

[Reviews](#)

[Software](#)

Search...

STARTING ELECTRONICS

...resources for Beginners and Beyond

[Facebook](#)

[Google+](#)

[Twitter](#)

[Blog](#)

[YouTube](#)

[Donate](#)

[Home](#) ▶ [Articles](#) ▶ [Arduino](#) ▶ [Measuring Voltage With Arduino](#)

Measuring DC Voltage using Arduino

Created on: 23 May 2013

Arduino analog inputs can be used to measure DC voltage between 0 and 5V (on 5V Arduinos such as the Arduino Uno when using the standard 5V analog reference voltage).

The range over which the Arduino can measure voltage can be increased by using two resistors to create a voltage divider. The voltage divider decreases the voltage being measured to within the range of the Arduino analog inputs. Code in the Arduino sketch is then used to calculate the actual voltage being measured.

This allows voltages greater than 5V to be measured.

This video shows the Arduino being used to measure various voltages:

Arduino Articles

Articles

Arduino Articles

Arduino Ethernet Board Programming and Using

Arduino Ethernet Shield – Connecting & Testing

Arduino Ethernet Shield

Measuring Voltage using Arduino



Can't see the video? [View on YouTube →](#)

– SD Card
Testing

Battery
Powering
Arduino Uno

Connecting a
Buzzer to an
Arduino Uno

Measuring
Voltage with
Arduino

PCF8563 (RTC) /
Arduino Testing

Arduino Web
Page Button and
Push Button
LED Control

Differences
Between
Arduino
Revision 2 and 3

Difference
Between
Arduino and
ATmega328

Choosing an
Arduino for
Beginners

Principle of Operation

Input Impedance

A digital multimeter set to measure DC voltage will typically have an input impedance of $10\text{M}\Omega$ or greater.

What this means is that the resistance between the two multimeter probes is $10\text{M}\Omega$ or more.

A high input impedance for a voltmeter (or multimeter on the voltage scale) is desirable as the higher the input impedance, the less likely the multimeter will influence or change the circuit being measured.

Measuring voltage across a component in a circuit with a multimeter that has an input impedance of $10\text{M}\Omega$, is the same as connecting a $10\text{M}\Omega$ resistor across the component.

If a voltmeter has a low input impedance, say $10\text{k}\Omega$ and a voltage across a $10\text{k}\Omega$ resistor is being measured, the multimeter is effectively changing the resistor value to $5\text{k}\Omega$ (two $10\text{k}\Omega$ resistors in parallel = $5\text{k}\Omega$ resistance). The multimeter therefore has changed the circuit and possibly the voltage being measured.



So a high input impedance is desirable in our voltage divider circuit if the impedance of this "voltmeter" is going to influence the circuit being measured.

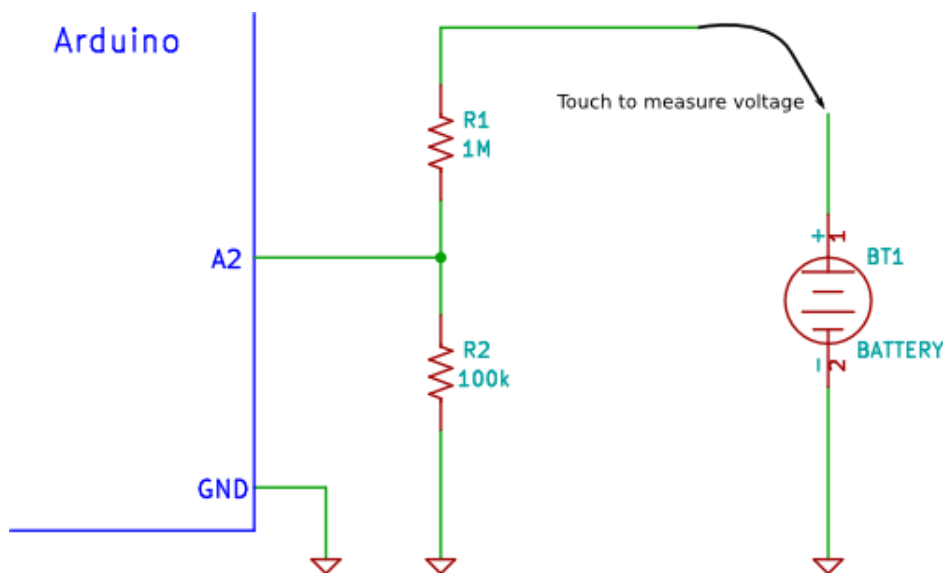
As a general rule though, a high input impedance device will generally pick up more noise or interference (EMI) than a low input impedance device.

Voltage Divider Circuit

A voltage divider circuit consisting of two resistors in series will divide the input voltage to bring it within the range of the Arduino analog inputs.

The circuit shown below will divide the input voltage by 11 (from the battery as the example input voltage being measured).

The circuit with the particular values shown has an input impedance of $1\text{M}\Omega + 100\text{k}\Omega = 1.1\text{M}\Omega$ and is suitable for measuring DC voltages up to about 50V.



Arduino Voltage Divider Circuit Diagram

Precautions

Common Ground

If the Arduino is powered from an external power supply or a USB cable (i.e. not powered by a isolated battery or other isolated power supply) the circuit may share a common ground or 0V connection with the circuit under test.

If the GND connection of the Arduino is connected to any other part of the circuit under test except GND, then this is the same as

shorting that part of the circuit to GND.

The GND of the Arduino is like the negative or common (COM) lead of a multimeter, except that it should be considered to be permanently connected to the GND of the circuit under test for safety, unless the Arduino or the circuit under test is completely isolated and "floating".

Input Protection

The resistor values in the circuit diagram above provide some over-voltage protection when measuring low voltages such as 5V, 9V or 12V. So if a voltage of say 30V is accidentally measured, it will not blow the Arduino analog input pin.

Any voltage higher than about 55V could damage the Arduino. The point on the resistor divider network connected to the the Arduino analog pin is equivalent to the input voltage divided by 11, so $55V \div 11 = 5V$. In other words, when measuring 55V, the Arduino analog pin will be at its maximum voltage of 5V.

Providing this basic over-voltage protection is at the expense of not using the full 10-bit range of the analog input ADC if only lower voltages are to be measured, but changes of about 0.054V can still be measured.

No other protection for voltage spikes, reverse voltage or voltages higher than 55V is shown in the circuit.

Arduino Voltage Measuring Sketch

The sketch below reads the value on pin A2 of the Arduino. It provides some simple filtering by adding up 10 analog values from pin A2 sampled at 10ms intervals.

After taking 10 samples, the voltage is calculated using the average of the 10 sample values and sent out of the serial port for display on the Arduino Serial Monitor window.

Note that calibrated values are used in this sketch – these will need to be changed for your particular reference voltage and actual resistor division factor, explained below.

```

/*-----
Program:      volt_measure

Description:   Reads value on analog input A2 and calcul
               the voltage assuming that a voltage divid
               network on the pin divides by 11.

Hardware:      Arduino Uno with voltage divider on A2.

Software:      Developed using Arduino 1.0.5 software
               Should be compatible with Arduino 1.0 +

Date:          22 May 2013

Author:        W.A. Smith, http://startingelectronics.or
-----

// number of analog samples to take per reading
#define NUM_SAMPLES 10

int sum = 0; // sum of samples taken

```

How the Code Works

Ten analog samples are taken using the following code:

```

while (sample_count < NUM_SAMPLES) {
    sum += analogRead(A2);
    sample_count++;
    delay(10);
}

```

The sum or total of all 10 values added together are stored in **sum**. The variable **sample_count** keeps track of the number of samples.

Both variables are reset after calculating and displaying the voltage:

```
sample_count = 0;  
sum = 0;
```

The number of samples taken can be changed at the top of the sketch:

```
#define NUM_SAMPLES 10
```

Don't make this value too high or the sum of all the samples will be too big to fit in the sum variable.

The voltage is calculated using:

```
voltage = ((float)sum / (float)NUM_SAMPLES * 5.0) / 1024.0;
```



A calibrated value is used in place of 5.0 in the above sketch – calibration is explained later.

The 10 samples (sum) is divided by 10 (NUM_SAMPLES) to get the average value read.

5.0 is the 5V ADC reference voltage. 1024.0 is the maximum value that the ADC can have plus 1 (1023 + 1 or 2 to the power of 10 plus 1) 1023.0 can also be used here.

This calculates the divided voltage – i.e. the voltage on the A3 pin.

The actual voltage is calculated by multiplying the divided voltage by the division factor from the voltage divider network:

```
Serial.print(voltage * 11.0);
```

The above line of code calculates the actual voltage and then sends it out the serial port for display in the serial monitor window.

The sketch uses a calibrated value instead of 11.0 as shown here.

Calibrating the Arduino and Circuit

A more accurate voltage could be obtained by using a precision reference voltage for the ADC and using 1% tolerance resistors or better.

Another way to obtain a more accurate reading is to calibrate the circuit. Calibration can be done by measuring the actual value of

the reference voltage and the actual values of the voltage divider resistors. These values can then be used in the calculations in the Arduino sketch code.

Each Arduino and set of resistors would need to be individually calibrated because the voltage from the 5V regulator and the resistance of the voltage divider resistors will probably be slightly different for each Arduino and set of resistors.

Performing the Calibration

To perform the calibration, you will need a multimeter.

Calibrating the ADC Reference Voltage

First measure the 5V on the Arduino from the regulator (found on the Arduino 5V pin). This voltage is used for the Arduino ADC reference voltage by default.

Now put the measured value into the sketch as follows.

```
voltage = ((float)sum / (float)NUM_SAMPLES * 5.015) / 1024.
```



In the above example, the voltage measured on the 5V Arduino pin was 5.015V.

Calibrating the Resistor Network

Connect a stable power supply, such as a 9V battery across the resistor network. Measure the voltage across both resistors together i.e. measure the battery voltage.

Now measure the voltage across the 100k resistor (R2) i.e. between Arduino pin A3 and GND.

The voltage divider factor is calculated by dividing the first voltage by the second voltage or:

dividing factor = input voltage ÷ output voltage


For example, if the first or input voltage measured is 10.02V and the second or output voltage is 0.9V, then the division factor is:



$$10.02 \div 0.9 = 11.133$$


Now use this value in the Arduino sketch code:

```
Serial.print(voltage * 11.133);
```

If calibration is used, then 5% tolerance resistors can be used for the voltage divider.

Comments **Community**  **Login** ▾

 **Recommend**  **Share** **Sort by Oldest** ▾

LOG IN WITH **OR SIGN UP WITH DISQUS** 

Peter • 2 years ago

HI

I have checked everything and got the following values.
(Arduino Uno)

Arduino 5v 5.02 v and 0.447 giving a factor of 11,23. These entered into the sketch gives a value of 4.96v on the serial monitor.

Regulated DC voltage 13.76 and 1.227 giving a factor of 11.21. These entered into the sketch gives values between 37.54 and 38.22

Any help will be greatly appreciated

^ | v • Reply • Share ›

TypeFasterOwner ➔ Peter • 2 years ago

I know this is old, but it might help someone else.
 If the measured Arduino voltage on the 5v pin is 5.02v (the reference voltage), and the voltage being measured on the analog read pin is 1.227 (which is the output of the voltage divider) then the analog read function will return 250.
 $(1.227 / 5.02) * 1024 = 250$

To get the voltage sensed on the pin we need to multiply the value returned by the Analog Read (250) by minimum resolution value of the ADC ($5.02 / 1024 = 0.0049$) which in this case is ($250 * 0.0049 = 1.227$)

To get the voltage dividers input voltage we multiply the voltage calculated above by the voltage divider factor, which is 11.21. ($11.21 * 1.227 = 13.76$)

To do the maths in one step:
 $\text{AnalogRead} * (\text{Ref_Volts} / 1024) * \text{Volt_Divider_Factor}$ or
 $250 * (5.02 / 1024) * 11.21 = 13.76$

As always, there's multiple ways of doing the maths. You should try and use the least number of intermediate steps. but don't crunch it down so much that you can't follow your own code. Also be careful about what variable types you use.

^ | v • Reply • Share ›

NoobieSnax ➔ TypeFasterOwner • a year ago

Thank you TypeFasterOwner
 Following the math on paper gives exact results but the serial monitor knocks out a fluctuating result pretty much always up to 2% out. Is there a reason for this? Is this just the tolerances or is it something within the code?
 Cheers everyone

^ | v • Reply • Share ›

startingelectronics Mod ➔

NoobieSnax • a year ago

The reference voltage is taken from the power supply, so is not accurate. If you are powering the board from USB, the power comes through a resetable fuse which has a small resistance. As the board current draw changes, the 5V will vary slightly because of the fuse

Measuring Voltage with Arduino
 will vary slightly because of the load.
 Accurate voltage measurement
 requires a good stable reference and
 filtered power supply.
 ^ | v • Reply • Share ›



dave • 2 years ago

Hi, Im in the same boat, I followed the instructions VERY CAREFULLY and I am getting similar results to The individual below me.

^ | v • Reply • Share ›



alessandro → dave • 2 years ago

I had the same "misleading" result of a calculated voltage around 38 V.

My Arduino was powered through USB and Vin read of 4.75V. My external power source to be monitored were 4 AAA batteries of 1.5V each, of total 6V. I've realized that if you don't connect the GND of the external source (the batteries) to the Arduino GND the result is wrong. After having connected the grounds, the reading is fine!

^ | v • Reply • Share ›



aniket mhetar • 2 years ago

if i apply 5v to arduino uno r3, then what is the output voltage

^ | v • Reply • Share ›

NoobieSnax • a year ago

Hi

Thanks for the great info provided thus far and in advance. I'm pretty new to this malarkey. What I can't work out is how the 1.1M ohm resistors divide the voltage by 11? Unless the current is 0.00005A and if it is how did you know it would be? Cheers

OK OK

I think I've got it. You used an arbitrary maximum for the voltage range i.e. 55V, selected a high resistance for R2 i.e 100K ohms then used the ol' $V_o = (V_{in} * R_2) / (R_1 + R_2)$ to establish the value for R1 - yeah? Current doesn't come into

