Making accurate ADC readings on the Arduino

There are many sensors out there which output a voltage as a function of the supply voltage as their sensed value. Temperature sensors, light sensors, all sorts.

Measuring that voltage, and converting it in to real figures for whatever is being sensed is not actually as simple as you might at f think.

There are many examples on the internet for converting an ADC value into a voltage, but basically it boils down to:

- Divide the ADC value by the ADC maximum value
- Multiply by the supply voltage

And that sounds simple enough, doesn't it?

```
unsigned int ADCValue;
double Voltage;

ADCValue = analogRead(0);
Voltage = (ADCValue / 1024.0) * 5.0;
```

Surely that looks OK, yes? You've got your Arduino plugged into the USB, which is supposedly 5 volts – after all, all the examples the web just say 5v.

Wrong!

What you have there is a rough approximation. Nothing more.

If you want to make ACCURATE readings you have to know exactly what your supply voltage is.

Measuring the 5V connection on my Arduino while plugged in to the USB is actually reading 5.12V. That makes a big difference to the results of the conversion from ADC to voltage value. And it fluctuates. Sometimes it's 5.12V, sometimes it's 5.14V. so, you real need to know the supply voltage at the time you are doing your ADC reading.

Sounds tricky, yes?

Yes.

However, if you have a known precise voltage you can measure using the ADC, then it is possible to calculate what your supply voltage is. Fortunately, some of the AVR chips used on Arduinos have just such a voltage available, and can be measured with the ADC. Any Arduino based on the 328 or 168 chips has this facility.

I came across this nice piece of code on the TinkerIt site. It measures this 1.1V reference voltage, and uses the resultant ADV value work out what the supply voltage must be.

```
long readVcc() {
  long result;
  // Read 1.1V reference against AVcc
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA,ADSC));
  result = ADCL;
  result |= ADCH<<8;
  result = 1125300L / result; // Back-calculate AVcc in mV
  return result;
}

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println( readVcc(), DEC );
  delay(1000);
}
```

Very nice. very elegant. And, more importantly, very useful.

So now, using that, your ADC code could now look like this:

```
unsigned int ADCValue;
double Voltage;
double Vcc;

Vcc = readVcc()/1000.0;
ADCValue = analogRead(0);
Voltage = (ADCValue / 1024.0) * Vcc;
```

And it will be a whole lot more accurate.

Addendum on calibration and accuracy

The internal band-gap, while *nominally* 1.1V can actually be anywhere between 1V and 1.2V. If you want super-accurate readings y may need to adjust the value 1125200 to a more accurate value to represent your band-gap. That value is calculated as the band- gap voltage (in mV) multiplied by 1023. You can do the opposite of the above system and manually measure your Vcc with a DMM then use that to measure and calculate the band-gap voltage in your chip. Multiply that voltage by 1000 for mV and then by 102 get the ADC division value, and Bob's your uncle. From then on, whatever your Vcc voltage does, you can get even more accurate ADC results.

This entry was posted in Uncategorized on February 1, 2016 [https://hackingmajenkoblog.wordpress.com/2016/02/01/making-accurate-adc-readings-on-the-arduino/] .

22 thoughts on “Making accurate ADC readings on the Arduino”

buy arduino kit
April 7, 2016 at 2:42 pm

Nice article. Doesn’t the onboard regulator fix it at 5v?

majenko Post author
April 7, 2016 at 2:46 pm

Around 5V, yes, though not always precisely. But of course that only has any effect when you are powering it through the barrel ja
Through USB the regulator is not in use, so it could be anywhere from 4.75 to 5.25V.

Norbert
June 5, 2016 at 10:05 am

Great! – Q: for a LED-Display (3 digits) as Digital-Voltmeter – do you purpose to scan "readVcc()" only once in the setup or continuously (1x per loop)?

majenko Post author
June 5, 2016 at 10:08 am

That all depends on what Vcc is. If it's a reasonably stable source that is unlikely to change, such as USB or an external power supp then just once, or maybe just "once in a while" is fine. If its a battery, though, where the voltage is likely to decrease over time (batteries have quite a sharp initial decrease, then a gentle slope down, followed by a rapid drop off) then you should check more often. How often really is up to you and depends on just how accurate you want your results. Doing it before every reading will be course, the most accurate, but will slow your readings down somewhat. For displaying on an LED display you don't need to be sampling that fast anyway.

Norbert
June 5, 2016 at 5:29 pm

Thanks so much! Your answer helped me out!
OK, so I understand if the Supply-Voltage changes I have to sample it. In my case I'll use it for a DIY-Power-Supply (60V/5A), wh a separate smaller 5V/1A Aux.-Supply is available. The hole thing is powered from normal 230VAC, so I think the deviation of the 5V-Power for the Arduino would be only small, (caused f.ex. by temperature-fluctuation). So I think to scan it every ~1min. with a "millis()"-counter (-Interrupt) would be in the right bandwidth.

Pingback: Using the ACS712 Current Sensor – Arduino++

Mingjian Lu
June 24, 2016 at 11:18 pm

Saved my life, ty

natong
August 24, 2016 at 9:51 am

Voltage = (ADCValue / 1023.0) * Vcc;

1023 or 1024 ?

---

majenko Post author
August 24, 2016 at 9:58 am

1024. As per the datasheet for the ATMega328P section 24.7. "...0x3FF represents the selected reference voltage minus one LSB."

Buzzy
January 23, 2017 at 8:31 pm

Correct value is 1023.0, since analogRead returns int (0 to 1023).
* 0 = 0.00 V
* 1023 = Vref
See https://www.arduino.cc/en/Reference/analogRead

---

majenko Post author
January 23, 2017 at 8:38 pm

No. Common misconception there. 1023 is *not* Vref. It is Vref **minus 1 LSB** (***READ THE DATASHEET***). So Vref would actually be 1024 not 1023 (1023 + 1 LSB = 1024).

Monitoração
September 12, 2016 at 8:43 pm

I put the piece of code on the TinkerIt site but the print shows me 5.14, not 1., can you tell me what is wrong?

---

majenko Post author
September 12, 2016 at 9:46 pm

What makes you think it would give you 1? 5.14 sounds perfectly reasonable to me.

---

mkvirtanen
October 5, 2016 at 6:18 pm

Using the above constant 1125300L my vcc was 5.14V as well. Two separate meters told me that it is 4.99 to 5.00 volts. This was, course, reflected in the A0 voltage measurements which were ~3% wrong as well. Therefore I made change:
result = 1095720L / result ;
and now the voltage measured was a bit closer to the truth.

---

aida
December 28, 2016 at 4:47 am

Hi, how about phidget interface kit 2/2/2? How to know its accurate ADC reading. Phidget is 10 bit resoultion same with arduino. I apply the same method?

---

majenko Post author
December 28, 2016 at 12:05 pm

Unless the phidget (whatever that is) is an Arduino with a suitable ATMega chip on board that you can program directly, no, absolutely not.

---

bwoo
July 27, 2017 at 6:59 pm

This makes no sense. If you want to measure voltages against the 1V1 reference voltage then use:
analogReference(INTERNAL);
But the 1V1 reference isn't anymore accurate than the 5v rail. I've measured voltages of the 1V1 reference as low as 1.04V and the datasheet says it has a tolerance of +/-0.1V so the accuracy can be as bad as +/-10%.

---

majenko Post author
July 27, 2017 at 7:50 pm

You have it all backwards, sorry. We're not measuring against the internal VREF, we are measuring the internal VREF using Vcc as reference. From that you work out what the Vcc must have been for the ADC to give the value it did when measuring 1.1V. Oh, and 0.11v (10% of 1.1V) is actually only a 2.2% error when translated up to the 5V we are actually measuring with. It's only a 10% error when you are measuring things against that 10% error. You may get an inaccuracy of ±22 LSB, not ±102 LSB as you imply.

---

bwoo
September 2, 2017 at 11:24 pm

OK, I got that you're trying to measure Vcc. What I still don't get is how you can work out an accurate value for Vcc just by reading the 1v1 against Vcc if the 1v1 isn't accurate. It seems to me that we have two unknowns (the 1v1 voltage and the Vcc voltage) and only one equation (the ratio of the two given by reading 1V1 against Vcc). How can this give us any conclusion without assuming t the 1V1 is accurate?
(And sorry for such a delay – I completely forgot about this)

---

majenko Post author
September 2, 2017 at 11:30 pm

That's just it: we're assuming that the 1.1v is more accurate – or rather, on the scale of 0-5V it's more accurate than the 5V is. Imagine a spring – or a "slinky". Stretch it wide. The whole length (5V) is quite long. One coil of it (1.1V) is very short. Now relax i little. The length changes a large amount, but the size of one coil only changes a small amount. It's all relative. Plus we're not just assuming that it is more accurate, we are assuming it's more *stable*. It's less likely to vary over time like the 5V is – and we can use that to our advantage: calibrate it. First off you do everything in reverse. You apply an accurate 5V to VREF and measure the 1. reference – then you know just what the 1.1V measures. You can then use that value to calculate what the 5V supply is when you running your board from a less stable supply (such as batteries). Yes, there are two "unknowns", but one of them is "less" unknov that the other.

bwoo
September 3, 2017 at 12:12 am

OK this makes more sense (although the need for calibration makes me sad)

ichoosedrone.us
September 4, 2017 at 10:43 pm

Great article! We will be linking to this particularly great content on our site. Keep up the good writing.
