In episode #80 we covered the basics of measuring current. Today, we will continue and measure small currents. In a next video, we will concentrate on measuring high currents.

Let’s start.

For the first example we need the bench power supply and an Arduino Uno. I use an Arduino Uno with a TFT shied to later use it as a current meter and display our measurements.

If we want to measure the current used by this Uno, we just connect our multi-meter in-between the power supply and the Arduino. As discussed in video # 80, we can do this on the “low” or on the “high” side. Let’s first use the “low side”. If we do low side current measurements, we connect the multi-meter between the negative of the battery and the negative of the module. As expected, everything works fine and, if the polarization of the multimeter is ok, we measure around 50 mA at 7 volts.

But in many projects, we want to measure the current with the Arduino itself because we want to use the results in our sketch. How can we do that? Measuring voltages was quite simple, because the Arduinos usually have 8 analog inputs. So, the obvious is to use one of these pins. So, according the last video, we have to replace the Multi-Meter with a resistor and measure the voltage across this “shunt”. Then, we can calculate the current and display it on the TFT. The sketch is simple and fast done. The only question is how we calculate the current if we have the reading of the analog pin.

As usual, this is done using Ohm’s law. Let’s assume we want, that we want to be able to measure a maximum of 500 mA. Then, 500 mA has to equal 5 volts. Then, the resistor has to be 10 ohms.

This is truly not ideal, because, if our Arduino would use 500 mA, we would lose 5 volts across the resistor. This has two disadvantages: We would not have enough voltage to power the Arduino anymore and resistor would heat up. The power to dissipate can be calculated and, in this case, would be 2.5 watts. But anyway, because we have only small currents and a voltage regulator, we can live with the 10 Ohms. Let’s continue.

Because Arduinos have a 10-bit ADC, they read 1024 with 5 volts on an analog pin. To calculate the current, we therefore can use the following formula. Easy.

As always, we do not want to destroy our Arduino. So, we quickly check, if everything is ok. The datasheet of the processor used says, that we must not go below -0.5 or above 5.5 volts on any pin.

We use our volt meter to check and measure nearly 500 mV, as expected. But wait a minute! The voltage is negative. Why? Because the current flows from the Arduino back to the battery, the voltage is negative compared to ground of the Arduino. But the Analog to digital converter of the Arduino can only measure positive voltages. And if the current gets a little bigger, we destroy or Uno. Not a good solution! Normally, in this case, we just exchange the cables of our multimeter and get a positive reading. But, as we learned in the last video, if we would do that with the Arduino, we would shorten the shunt and always get a zero reading.

But fortunately, I have a very precise 16-bit ADC, the ADS1115. Let’s check this one. It costed nearly as much as the whole Arduino, so it must be better than the built-in ADCs. But a short check of the data sheet shows, that it has exactly the same problem. It is only usable for positive voltages. So, we are stuck!

As a last resort, we consult our “bible for electronics” purchased in one of the last mailbags. And really, we find a solution: To use a simple Opamp. Here, they propose a general purpose LM358A. But because these Opamps are not optimized for our case, it need some compensation circuitry to zero the reading if no current flows. Luckily, we get much better suited Opamps for our purpose. If we use the tiny AD8028 Opamp, for example, we can get with of nearly half the components. Why did I choose the AD8028? There are a few reasons:

1. It can be used with a single supply voltage of 5 volts. The early Opamps were designed for split voltage and higher voltages of e.g. +/-12 volt. Which means 24 volts in total
2. It is “rail-to-rail”, which means, its output can go to both “rails”, zero and 5 volts. Many Opamps can only go close to the rails, e.g. from 0.2 to 4.8 volts
3. It is “zero drift”, which means, that we can avoid the components to adjust the zero point

Because we have to use an Opamp to solve our problem of negative voltage, we can use it also as an amplifier of let’s say factor of around 100. With this amplification, we only need 5 volts / 100 = 50 mV across the shunt. Very good, because we can reduce the shunt to only 0.5 ohm! Which will improve our design considerably. So, with the 50 mA, we expect a voltage of 2.5 volt. Let’s check. Luckily, everything works as expected (or, as the engineer would say: As planned). Now, we can connect this output to our Arduino A0 pin, adjust the formula and voilà, we have a working low side current meter with our Arduino.

Now let’s do a little blinking with the LED. Now, the display starts to display instable values. If we really want to know what happens, we either have to increase the speed of our program, or we have to use an oscilloscope. But you know how to remove a delay() statement in a sketch and I save the usage of the oscilloscope for a later experiment. So, we can go on.

We are now able to measure mA. Now, we want to go on and measure smaller currents. For example, I have a Polar Heart rate watch with two different sensors, one with the proprietary transmission protocol, and one with additional Bluetooth transmission. I want to understand, why the one with Bluetooth discharges the battery much faster than the one with the Polar protocol. So, let’s check both.

As we learned in our last video, measuring small currents is not a domain of multimeters, even not of the expensive ones. This is, why Dave Jones from eevblog created a small device: the µcurrent Gold. Gold in this case not only means quality, it means also price: It costs including shipping around 80$. But for me, it was obviously worth the money…

It includes three shunt resistors and a fixed 100x amplification with a maximum output voltage of +/- 1.5 volts. Because it uses a coin cell battery, it has no “behind the scenes” ground connection. The three shunts result in three ranges: mA, µA, and nA. The respective shunt resistors are shown in this table. It starts with a stunning small 10 mΩ and ends with 10kΩ for the nA range.

So, the maximum currents are: +/-1.5 A in the first, +/-1.5 mA in the second, and +/-1.5 µA in the most sensible range and the voltage drop is always only a stunning 15 mV. Please be aware, that the ranges differ by a factor of 1000. This is completely different than a cheap multimeter, where your ranges differ usually only a factor of 10. I do not know, if you can destroy anything if you select the wrong range, but it fooled me, because in overload situations, the output voltage starts to oscillate. Before I discovered my error I thought, the device was defective.

So, we use the low side method and feed the pulse sensors with the same battery, just removed from the case. The µcurrent is connected like a multimeter, and, because I suspect varying currents, I connect its output to my oscilloscope. We immediately see, that the pulse sensor has lots of current spikes of 1.4 mA. They are short, but they do not switch off if the device is not used. The device should start when there is skin contact between the two poles and should sleep, if this contact is no more there.

After a while, the sensor enters real deep sleep and we have to switch the range of the µcurrent. No, we see, that it consumes a constant 1.2 µA.

Now let’s compare this number with the sensor without Bluetooth. During activity the peaks are only 0.45 mA. This is roughly one third of the Bluetooth sensor than the 1.4 mA. But, more importantly, it stops nearly immediately if the skin contact is no more there. Then, it enters into deep sleep mode and also here, we can measure a lower current consumption: About 0.6 µA which is also half of the Bluetooth sensor. But I think, the huge difference in battery live does not come from there, I think, it comes from the fact, that the Bluetooth sensor sometimes does things which are not necessary, but use a lot of energy if another Bluetooth device is in its proximity.

So, you see, with Dave’s amplifier, we can measure quite small currents. And if we connect it to the oscilloscope, we are even able to see changing currents. You see, I did not do any precautions to reduce noise, which would be necessary, if we want to measure lower currents. For my experiments, measuring one µA is sufficient.

So, we started on the low side with a normal shunt and a multimeter. But quickly, we discovered, that, we needed an amplifier to reduce the size of the shunt and change the polarity of the output. We were able to build a simple amplifier using a well suited Opamp.

For measuring smaller currents, we could go on with our own amplifier, but I choose to use the µcurrent gold. With this device, we were able to measure deep sleep currents of two different heart rate sensors to find out, that the main difference between the two sensors is the behavior if the sensors are not used.

Measuring on the low side is the preferred method for small currents. However, we always need a resistor between the two grounds (battery and device). And sometimes, this is not, what we want. Then, we have to go to the “High side”. But this and also the hall effect sensors, will be the topic of a future video.

I hope, this episode was useful or at least interesting for you. Bye

#84 Measuring small currents (incl. building of an amplifier and the µCurrent Gold)

In this video we measure small currents down to µA. We also use the Arduino itself for that purpose, discover and solve some of the problems associated with that.

We only work on the low side and discuss also the effect of the shunt resistor and how to reduce it.

Link to the µcurrent Gold:

https://www.eevblog.com/projects/ucurrent/