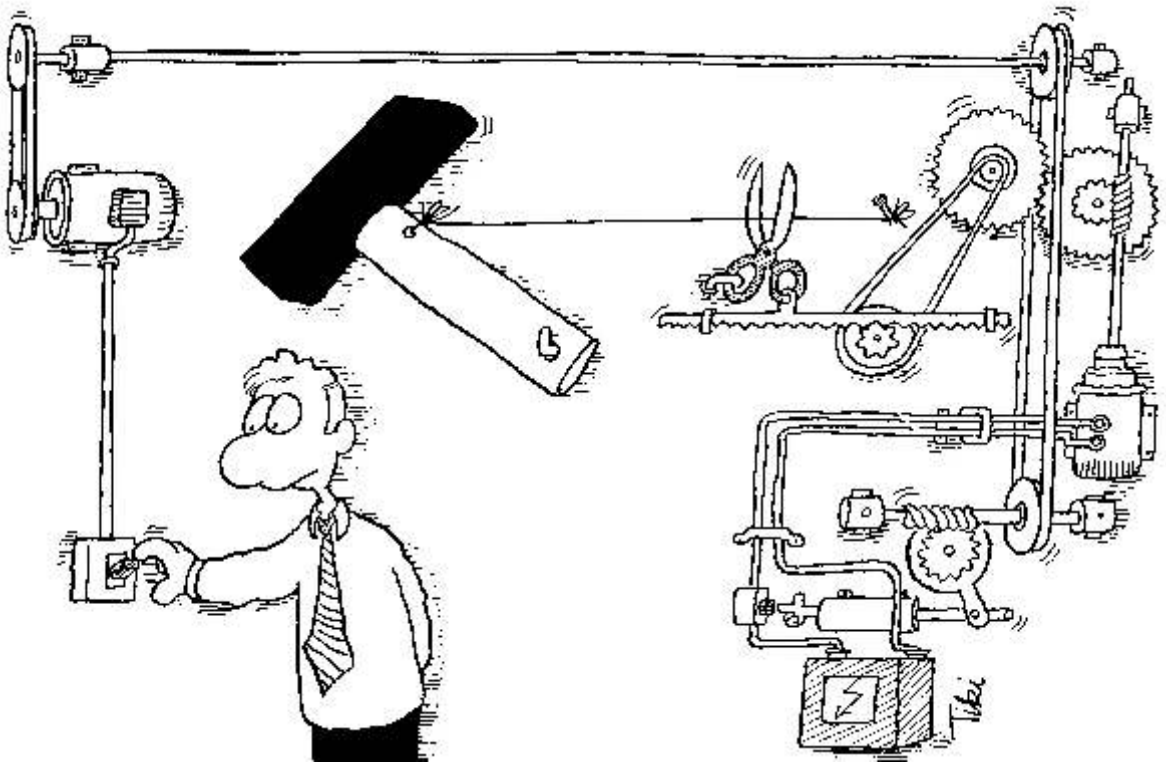
# Small microcontroller course

for beginners
with the Arduino Uno



What is the goal?

- Losing fear of technology

- sniff some physics

- train logical thinking

- Enjoy experimenting

April 2020
Martin

# content

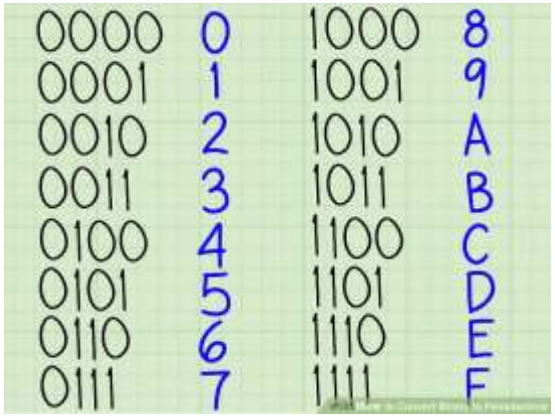
1.1

Welcome to the microcontroller course from Germany

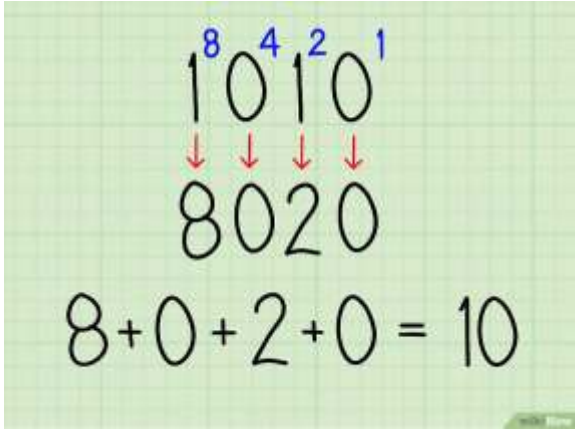
## mic-1.1 what does a microcontroller think?

Your computer does not know 3 or 4, it can only count 0 and 1.
But that very quickly.
If you need 5 seconds for an invoice, he did it in 1/100000 seconds.
You can think of something, but he can't think at all.
The computer can only process and calculate tasks.
If you don't give him tasks, he falls asleep immediately.

How can you calculate with 0 and 1?
Here is the conversion from binary to hexadecimal:




So if you want to convert its number: 1010 into your decimal system,
then each digit has a value (see picture).




The computer does not have to convert.
He can e.g.

add up directly.

```
   11      (3)
+ 100      (4)
---------------
= 111      (7)
```

more you find here: https://www.wikihow.com/Convert-Binary-to-Hexadecimal

1.2

try to calculate like the controller:

5 + 3 = 8

in binary:

**101**
**+ 11**
**--------**
**=?**

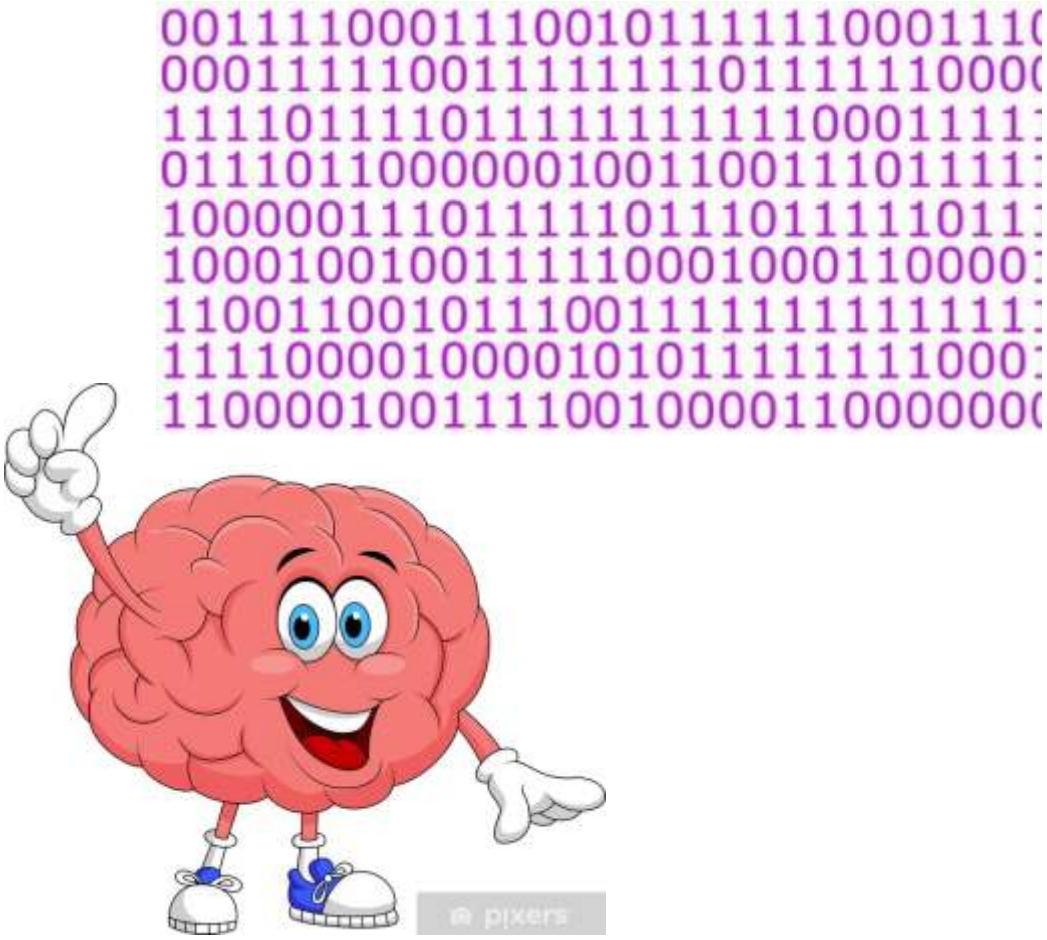
or

binary
**D + 10**

what does "D" mean?

if you could look into the brain of the controller, it would look like this.

terrible



only where does he have hands and feet?
How can he see, hear and speak?

We'll get to that later

The microcontroller is the treasure trove, its brain.
Its components
(Metal, crystal, plastic) come from a factory.
But what's more important is the program.
That was put into him by someone
who can think and design creatively,
e.g. from you.
That means you're the
most important part of it all..


Some people think that everything that
makes up their lives is in their brains,
and everything would come from this earth..

But who thought and made man is much bigger
and nothing works without our God.

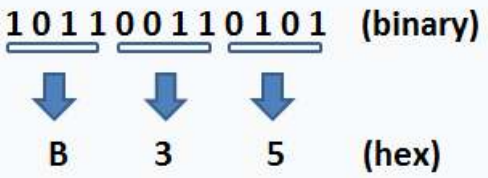
Do you like this song
https://www.youtube.com/watch?v=i5naQOpPDzU

# 1.4

We calculate with numbers in the decimal system and have it
10 characters (0, 1, 2, .... 9).
The binary system has only 2 characters (0, 1) a bit.
The computer cannot do more. He's basically stupid, but quick.
For us, this is not very clear, so 4 bits have been combined
to a hex number (0,1,2, ... 9, A, B, C, D, E, F). (just for info)

| Decimal | Hexadecimal | Binary |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 2 | 10 |
| 3 | 3 | 11 |
| 4 | 4 | 100 |
| 5 | 5 | 101 |
| 6 | 6 | 110 |
| 7 | 7 | 111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

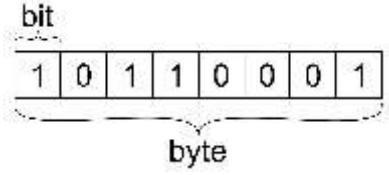
**To convert binary numbers into hexadecimals, you only have to make 4-bit groups and convert directly each group:**




In IT practice one speaks of byte (B).
8 bits are combined here.

You can count up to 255.

The double byte length is a word
which goes up to 65535.




Then there are the asci signs with that
one every character on the computer keyboard
can assign to a byte.

| Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |

2.1

# mic-2.1

Now you don't have to learn to calculate with 0 and 1.
You can tell the computer what to do in your language.


Just as there are rules in a text (letter or book)
for commas, periods, lower or upper case, you also have to pay attention to a few rules,
if you want to tell him something.

In a way, this is a separate language with English words and its own grammar.
In our case it is the programming language "C" for our small microcontroller Arduino.

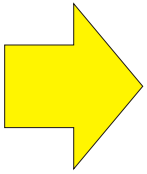
(I only found elaborate instructions on the Internet. Maybe there are also simple ones.
Therefore in the following only what we need)

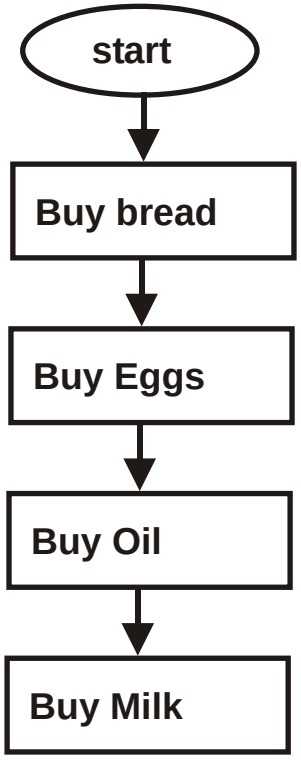
Was the microcontroller (we just call it "mic")
that it can work?

Like when mom sends you to shop.
Then you have a shopping list.
You start at the top and when you find
the bread this point is done.
(cross out)
Now the next one comes
until everyone is in the basket.

That's exactly what the mic does.

**and here is his program**






Mic is working step by step

the mic doesn't understand the word “shopping”, of course.
It knows very few words (commands), like in math.
Mic understands:

**+ add**
**- subtract**
*** multiply**
**: to divide**
**( ) what is in the parentheses belongs together**

In math it's an equation:
a + b = c

for mic "=" is a command:
save the result of **a + b**
in the variable **c**.
that's why we say we write in the program:

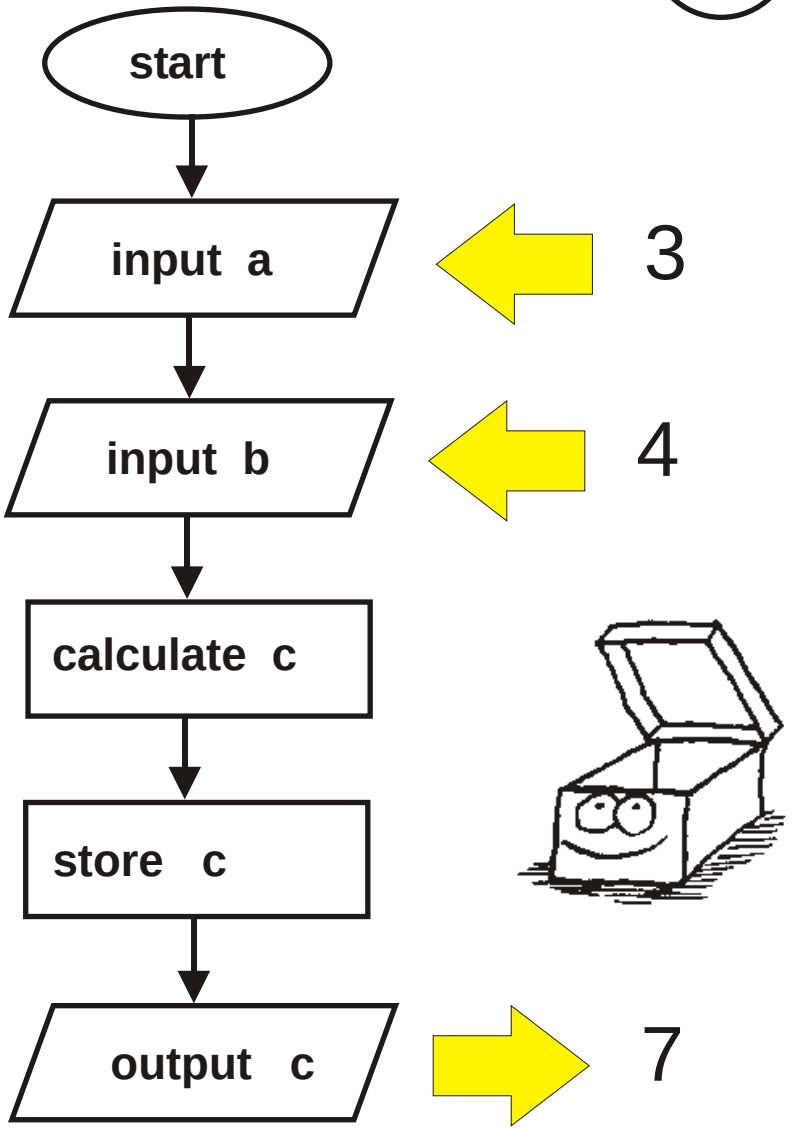
**c = a + b**

Each variable is stored in a memory location.

Either fixed in the program or read in from the outside
via an interface (we will see later)

You tell mic how big a is e.g. **a = 3** and **b = 4**,
then it calculates and comes to the result **c = 7**

**This is exactly how you do it with your calculator.**



so that mic can understand you, you have to use his grammar.
First, he needs to know what that variable is.
Are these numbers or letters?

Integers are your primary data-type for number storage.
The number range goes from -32,000 to 32,000

```
int a = 3;

int b = 4;

int c = a + b;
```

Mic understands that **a** is an integer variable and has the value 3.

Each line is a command and is marked with a **";"** completed.

**";"** is part of its grammar.

words written in blue, red or green are commands he knows.

„Es braucht
die richtigen Worte,
nicht derer viele."

It takes the right words, not many of them.

...and

Nice words are not worth much
if actions don't follow.

So that you can understand your program later, it is good to write an explanatory text behind the short commands.

**"//"** is used for this. Everything in the line behind it is only for you.

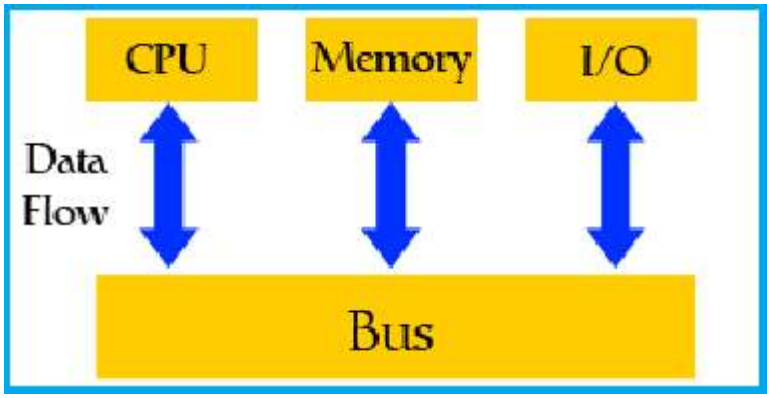
E.g. `int c = a + b;  // The result is saved in "c".`

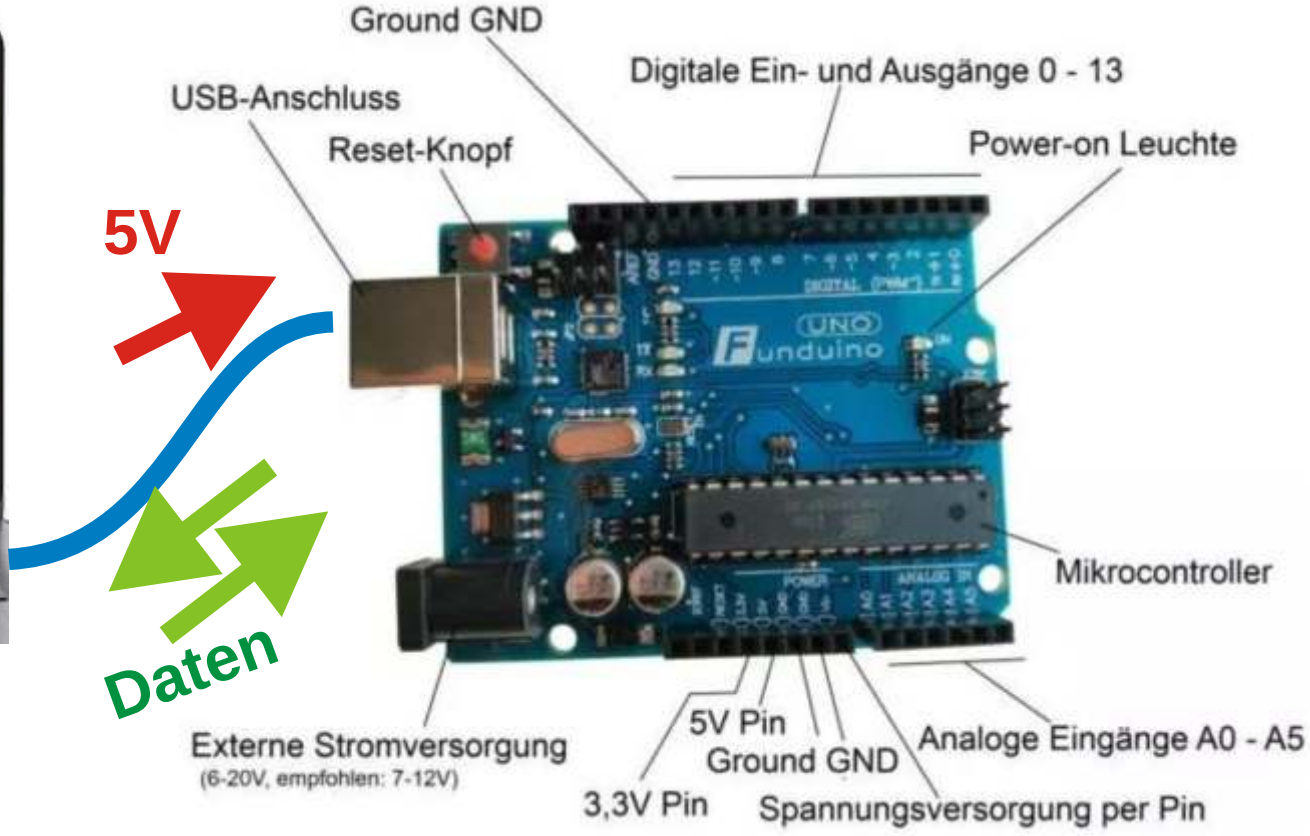
Before we write another program, let's take a look at mic hardware.

# Mic-3

3.1

What's going on in the mic:
It needs 5V supply, which it uses the blue USB cable from the laptop. And he can use this cable talk to the laptop (with you).
Inside is his brain (CPU) and his memory (Memory) and connections at which it reads signals and can output (I / O). All are connected (Bus)



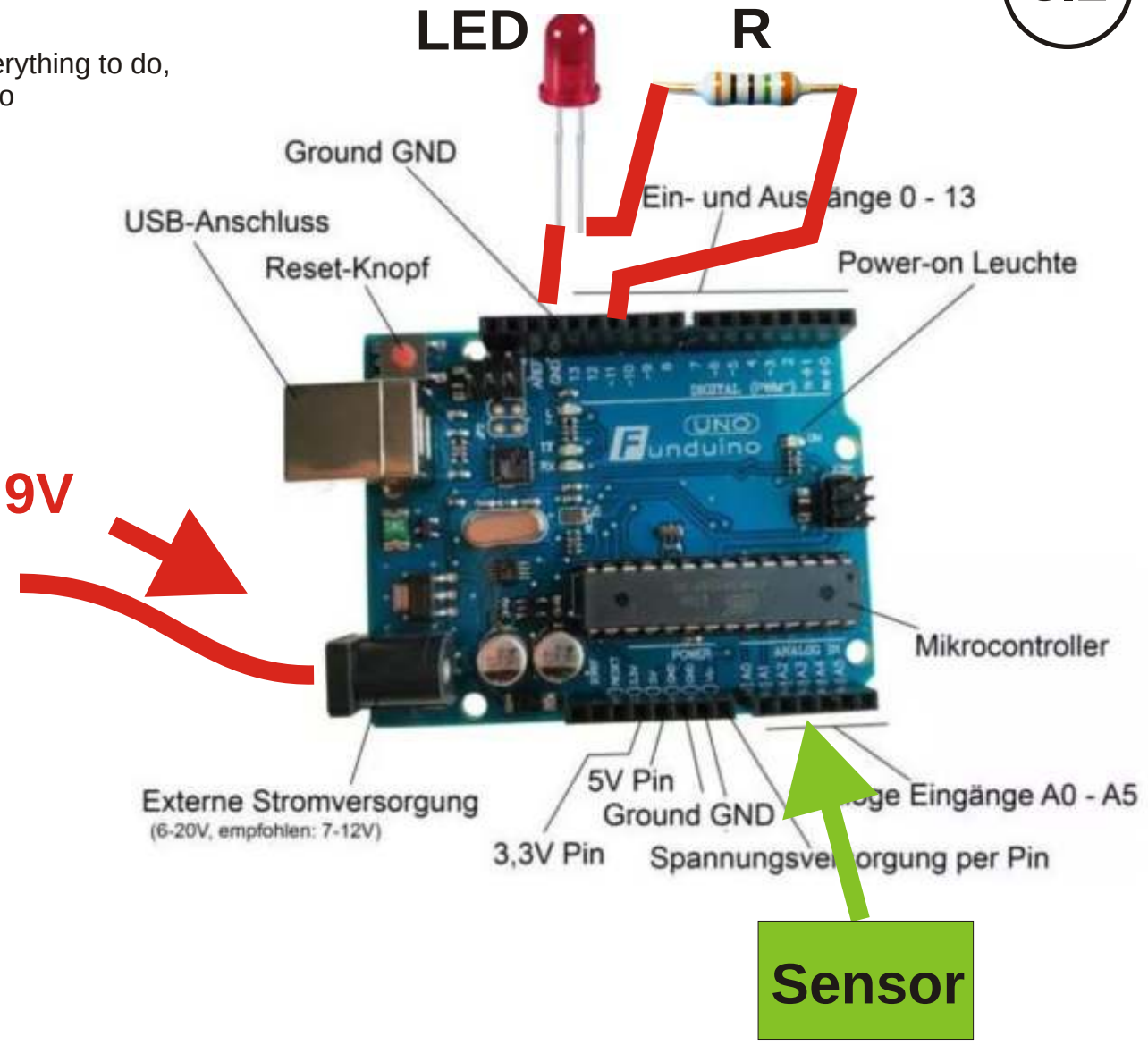



When the laptop has told him everything to do,
(Program uploaded) then he is too
satisfied with a 9V battery.

At the digital outputs it can
e.g. control an LED.
Always through a resistor R, so
the output is not overloaded.

At the analog inputs it can
Read signals from a sensor.

More on that later.

If you want to know more,
what is in there, you can see on the picture:

CPU: the central command and computing unit,
that controls everything
.
Timer: The delivers the exact clock (16 MHz) for each work step

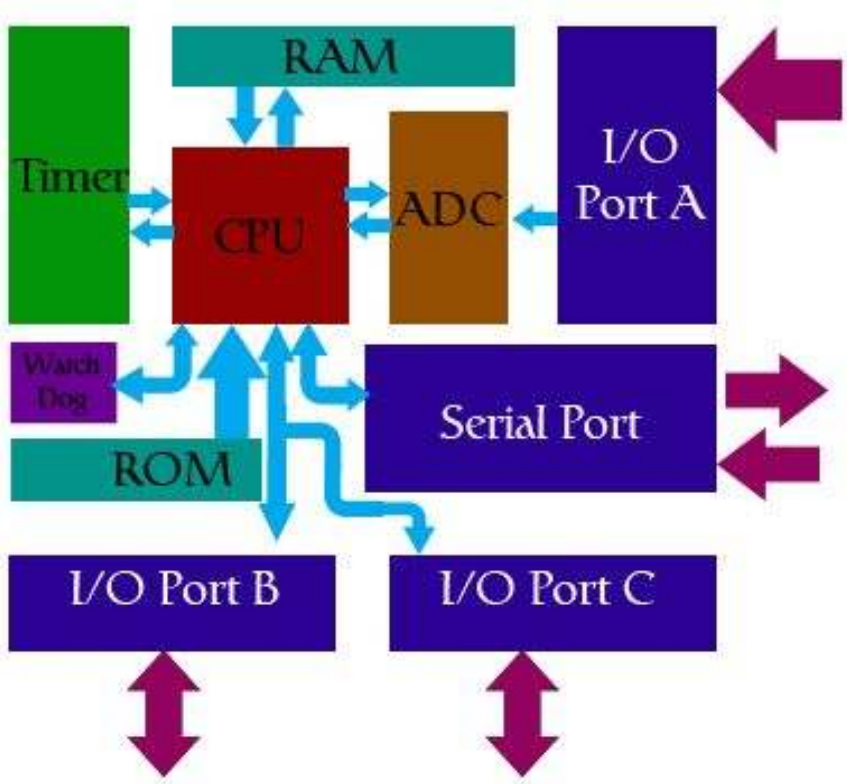
ROM: His long-term memory. Everything is saved there
what he needs to know anyway. Since you have no access.

RAM: You can write and erase in his short-term memory.
Then there is your program.

Serial Post: This is the interface (connection to the laptop)
Sometimes the mic is allowed to speak, then he has to listen,
as it should be with you.

I / O Port B and C: Mic
output digital signals (0V or 5V)
or read in (input).

I / O Port A: Analog signals (between 0-5V) can also be
measured and converted into digital values ??by the ADC.




What good is all that if life doesn't come in?

Like you too. You are like a light bulb. Everything is there and yet lifeless.
There is light and heat only when electricity flows.
Where Jesus lives, electricity flows and gives life.

**Mic-4**

4.1

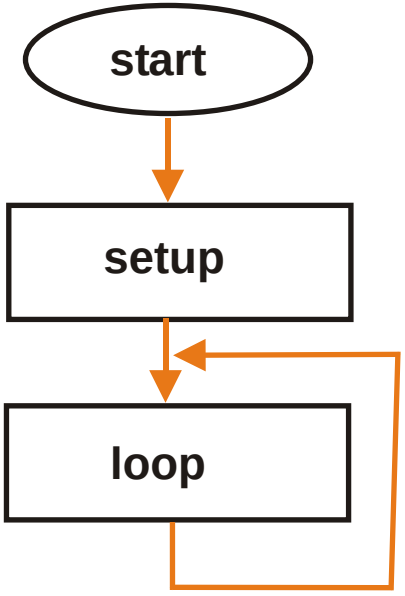
Now we download the software for our laptop.

https://www.chip.de/downloads/Arduino-Software-IDE_90789714.html
or
https://www.arduino.cc/en/main/software


If this tool is installed on the laptop it will look like this:

Everything listed in the setup is only processed once.

What is in the loop is constantly repeated.




# 4.2

Now we are writing our first program.
There is a yellow LED on pin 13 (connection 13) on the mic board, which should be in the Rhythm of 1 second (= 1000 ms, milliseconds) flash.

First we have to say in the setup that this pin should be an output.
With the command “pinMode” we define that. (Case sensitive)
The pin number is in the brackets, then the mode “OUTPUT” separated by commas.
Every instruction is always marked with “;” completed.
Everything in the setup is enclosed by the curly brackets {}

Signals are output in the loop with the command digitalWrite ().
HIGH = 5V
LOW = 0V
So that the LED too
1 second
remains on
Delay () command
second hand.

sketch-LED-blinkt | Arduino 1.8.11

Datei Bearbeiten Sketch Werkzeuge Hilfe

sketch-LED-blinkt §

```
void setup() {
                          // put your setup code here, to run once:
pinMode(13, OUTPUT);      // Pin 13 soll ein Ausgang sein.
}

void loop() {
                           // put your main code here, to run repeatedly:
digitalWrite(13, HIGH);   //Schalte die die Spannung an Pin13 ein (LED an).
delay(100);               //Warte 1000 Millisekunden (eine Sekunde)
digitalWrite(13, LOW);    //Schalte die die Spannung an Pin13 aus (LED aus).
delay(100);               //Warte 1000 Millisekunden (eine Sekunde).
}
```

In concern
lies dying
Spice

4.3

In order for the mic to understand us, a tool (compiler)
translate that into his language (0 and 1)

To do this, we click on the tick on the top left
(it will turn yellow and at the bottom right we will see a green bar)
After the compiler is finished, appears in the black box below
a text with information.


If you have forgotten something, e.g. ";"
then the error message comes in orange and
the line with the error is highlighted in pink.
(Sometimes the error is in one line
further up.)



Jetzt brauchen wir nur noch
das Mic-Board,
dann kann es los gehen.

## mic-5.1

5.1

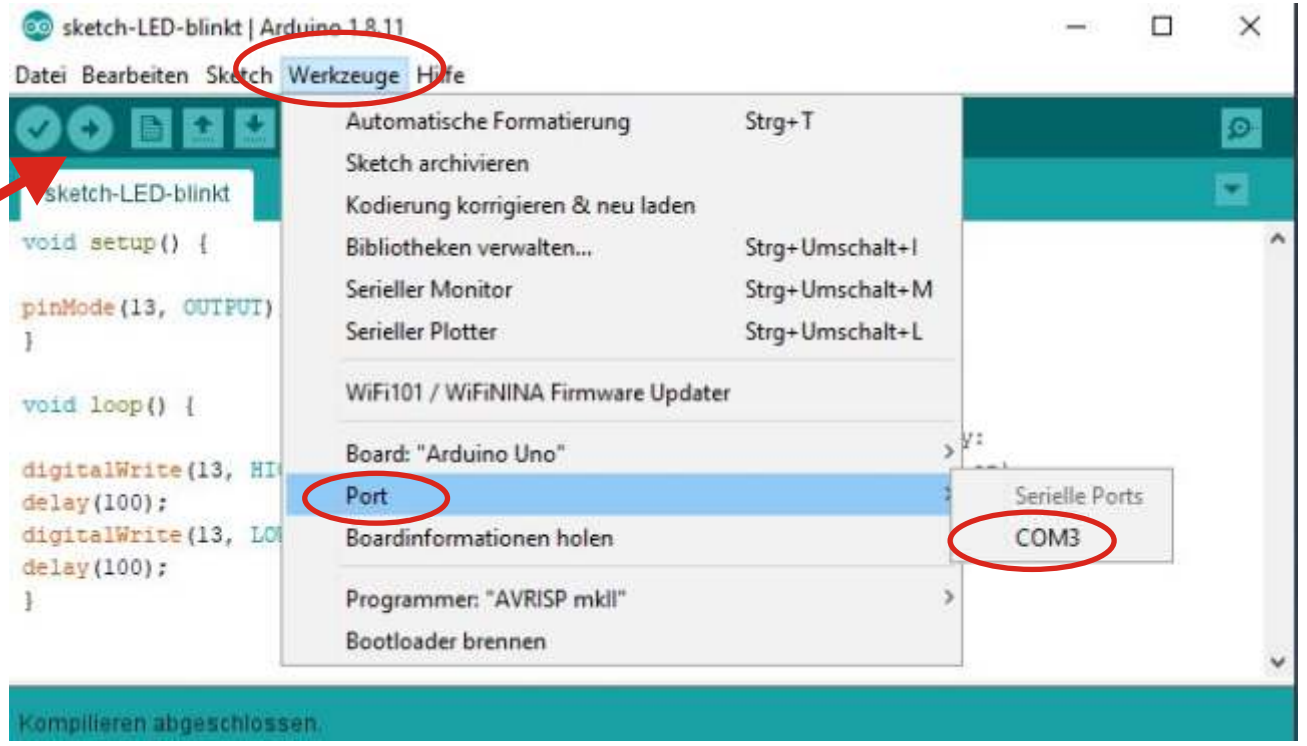
Hooray, the real mic is here!

plug in right away (see lesson mic-3)

A green LED lights up on the board,
that means mic has energy.
Maybe a yellow LED is already blinking
further left.

Now open your first mic-4 program,
Click on the compiler (tick left-top),

Now for uploading the program
press the arrow.
(Probably an error message is coming now)



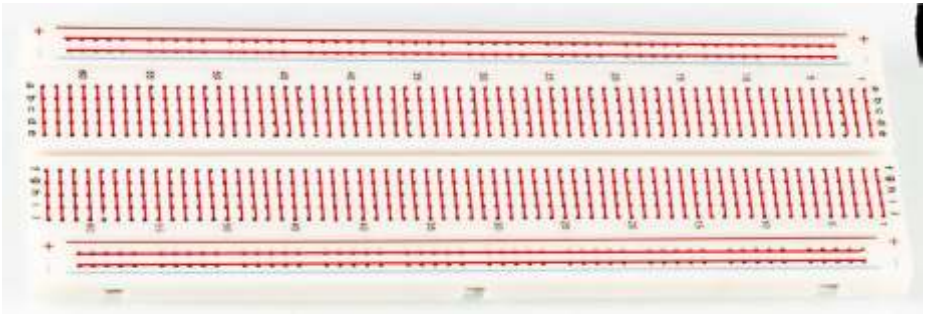

Therefore in the “Tools” menu
Go “port”. There on COM3 or COM6,
(what he is suggesting) click.
The PC now knows which door (port)
he can speak to mic.

Uploading works.
The yellow LED flashes.

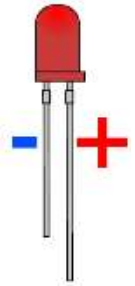
Try out what we are changing
can:
e.g. in the command “delay” in brackets
change the time. Instead of 1000
Enter 100.
Compile, upload and the flashes
yellow LED 10 times faster.

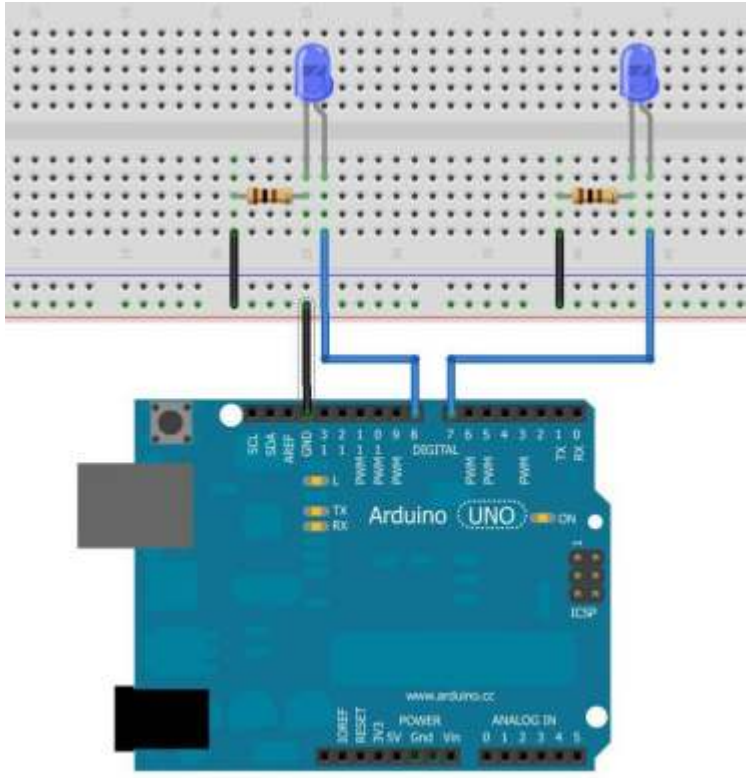
## 5.2


A breadboard or "breadboard" is a good tool for circuits
Concerns without soldering. There are always several contacts in
a breadboard belongs connected. Therefore, many places
can get these cables without being lost or screwed down.
The following picture shows in color which contacts are related.



Important ! Always a series resistor
with 330 ohms or more in series with
insert the LED.
The long wire of the LED is
the anode (+),
always to the mic output
and the
Resistance at GND (0V).
.






```
oid setup()
{// //Wir starten mit dem Setup
pinMode(7, OUTPUT); //Pin 7 ist ein Ausgang.
pinMode(8, OUTPUT); //Pin 8 ist ein Ausgang.
}
void loop()
{// Das Hauptprogramm beginnt.
digitalWrite(7, HIGH); //Schalte die LED an Pin7 an.
delay(1000); //Warte 1000 Millisekunden.
digitalWrite(7, LOW); // Schalte die LED an Pin7 aus.
digitalWrite(8, HIGH); //Schalte die LED an Pin8 ein.
delay(1000); //Warte 1000 Millisekunden.
digitalWrite(8, LOW); //Schalte die LED an Pin8 aus.
}//Hier am Ende springt das Programm an den Start des Loop-Teils. Also•c
/
```

## 5.3

For further experiments we download the manual.

German: http://funduino.de/wp-content/uploads/2016/11/Anleitungen-deutsch-12-2016.pdf

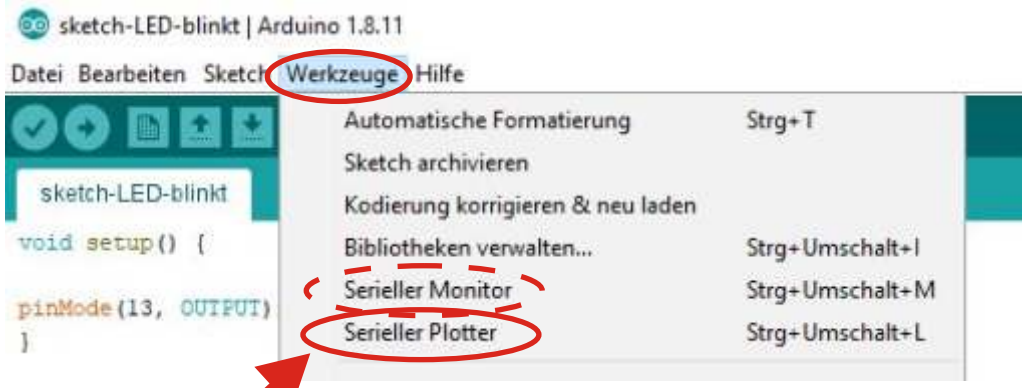
or English: http://www.funduino.de/Arduino-tutorials-08092014.pdf

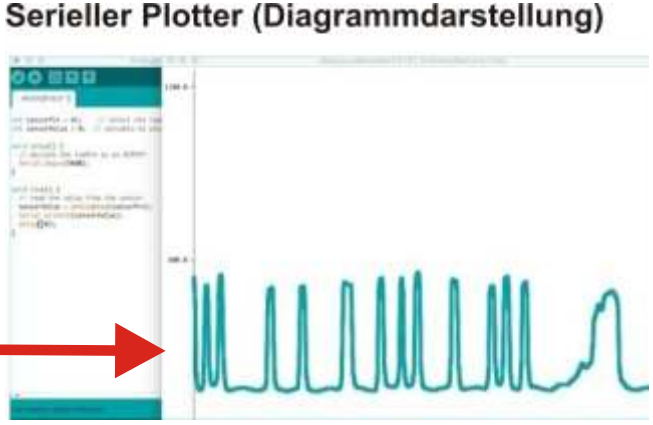
you can choose now
which circuit you want to do.

With an example I want to show
how sensor values ??are read in and
transferred to the PC via the serial interface
and be displayed there.
This works particularly well with the photo resistor.
So in the manual (German page 32) or
(English page 37) you will find the circuit and
the program.
To see the data on the PC screen,
let's click in the end
Tools menu on Serial Monitor and see
how the sensor values ??are displayed on the screen.

It gets even more interesting when we use "serial plotter"
click.

Now move your hand in front of the photo resistor
back and forth and watch what's going on in the diagram.



# mic-6.1

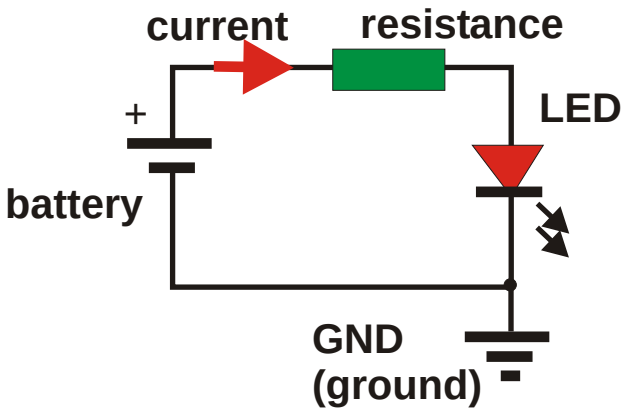
Need a little bit of physics
we still.

1. Current can only flow when the circuit is closed.
So from the battery (+) through the resistor and
through the light-emitting diode (LED) back to the battery.
The negative pole of the battery is our reference point (GND).

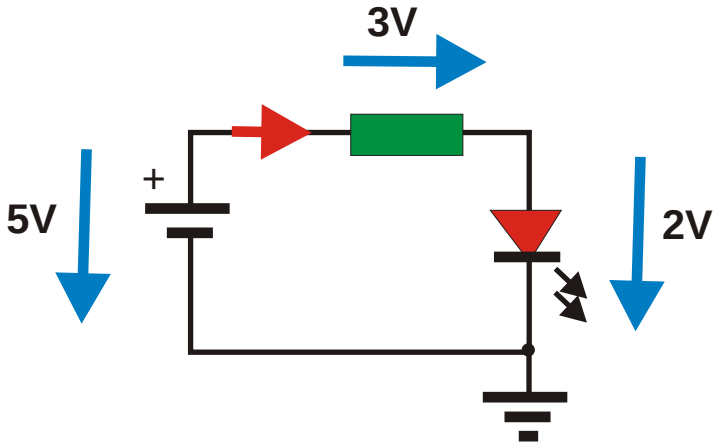
2. So that the current (red arrow) does not become too large
and destroys the LED, we need it
Resistor that limits the current.

We work with a battery voltage of 5V.
There will be about 2V on the LED when it is lit.
Then 3V remain for the resistor.

3. There is a simple formula (Ohm's Law)
with which we can calculate the value for the resistance.




**R = U / I**




U is the voltage in volts (V) across resistor R.
For the current I in amperes (A) we assume 5 mA (0.005 A),
so that the LED lights up but is not yet damaged.

R = 3V / 5 mA = 0.6 kOhm = 600 Ohm

The larger the resistance value, the smaller the current.

The set contains resistors with 330 ohms (also ok).

## 6.2

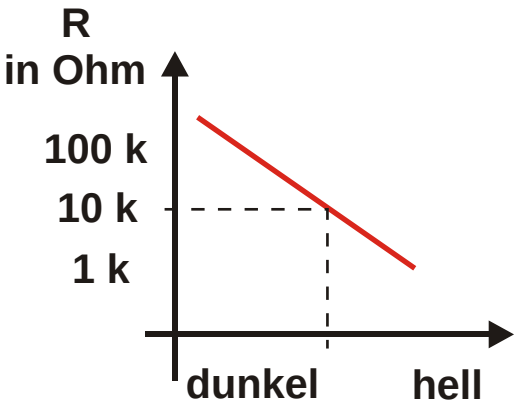
How is that with the example now
from Mic 5.3?

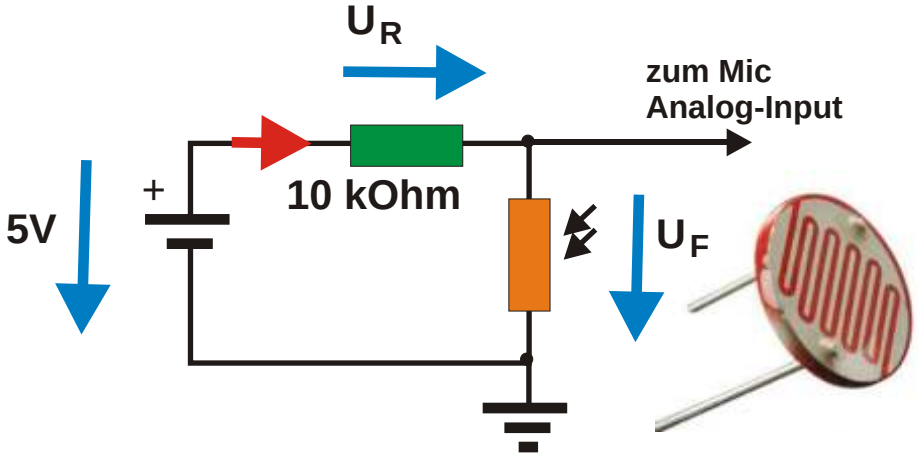
The photo resistor has one
large resistance value e.g. 100 kOhm.
If a lot of light falls on it, the value becomes smaller, e.g. 1 kOhm.

But the mic needs one
Measure the voltage signal
can. For this we leave again
to flow a current.
Of course with a series resistor
e.g. R = 10 kOhm (also included in the set).

If there is some light now
He has about photo resistance
half the voltage 5V / 2 = 2.5V

The mic can measure this voltage.
(Read in at the analog input, via the
Give serial interface to PC)

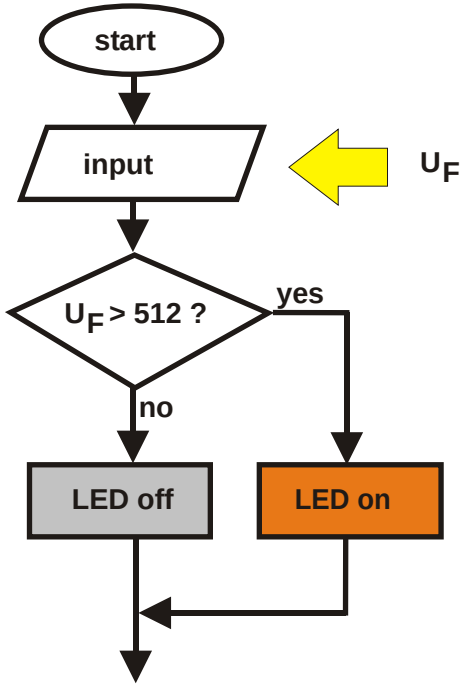


## 6.3

When the measured value has been read in, we can also control an LED.
This LED should light up if the measured value is greater than 512. That is just half from the maximum value 1024, which corresponds to the 5V.

For this we need a command that can distinguish between a smaller and a larger value.




That is the “if” command, that is the condition

where () met,
then the LED is turned on.

If not “else” switched off again.
Whatever Mic should do is always in the braces.

```
if (sensorwert > 512)          // Wenn der Sensorwert über 512 beträgt
{
digitalWrite(LED,HIGH);    // die LED soll leuchten
}

else
{
digitalWrite(LED,LOW);    // andernfalls soll sie nicht leuchten
}
```