IoT – Praktikum 2018/19

Team A:

AVR-Technologie

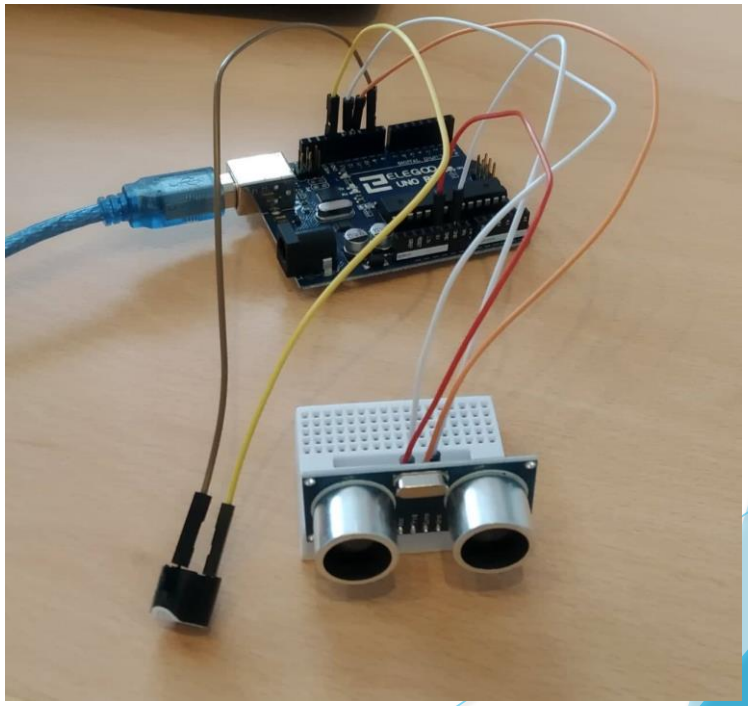
TI CC1101

Der Start …

- Grundverständnis
- Arbeiten mit Sensoren
  - Fernbedienung
  - Abstandsensor
  - Bewegungsmelder
- Aber auch mit
  - Motoren

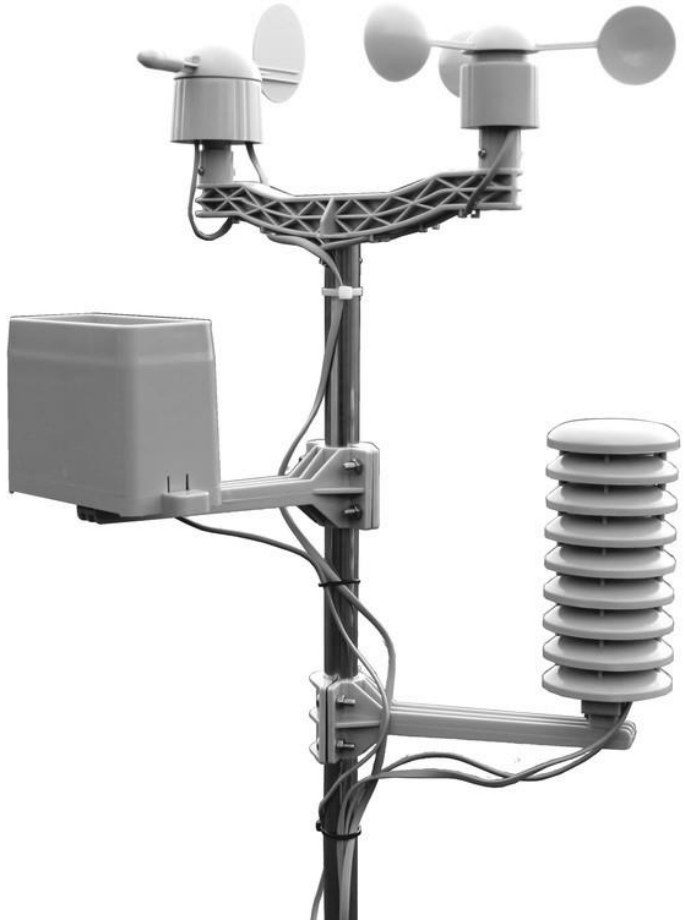
Einparksensor

```
#include "SR04.h"
SR04 sr04 = SR04(12,11); //Pin 11 & 12 werden verwendet
long a; //Speichert den Wert des Abstandes

void setup() {
  pinMode(9,OUTPUT); //den Pin für den aktiven Buzzer als Output definieren
}
void loop() {
   a=sr04.Distance();//gibt den Abstand in cm zurück
   if (a > 30){
    delay(500);
   }
   else {
    digitalWrite(9,HIGH);
    delay(100);
    digitalWrite(9,LOW);
    delay(a*10);//delay ist der Abstand in cm * 10
   }
}
```



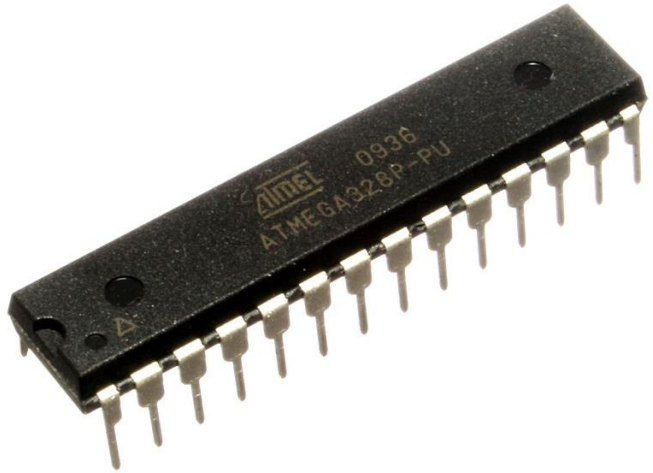
Idee

- DHT11
  - Temperatur
  - Luftfeuchtigkeit
- TI CC1101



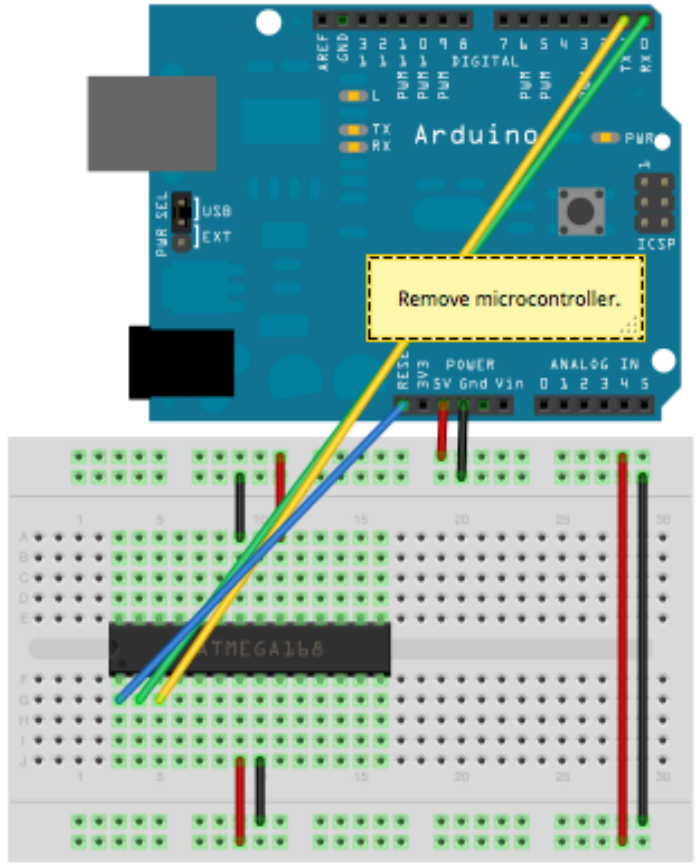
Atmel ATmega 328p

- tinyAVR, megaAVR und xmegaAVR
- 8-bit AVR-Microcontroller
- Spannung zwischen 1,8 – 5,5V
- Taktrate bis zu 20Mhz
- 32 KiB Flash-Speicher
- Stromsparende Architektur
  - Pico Power



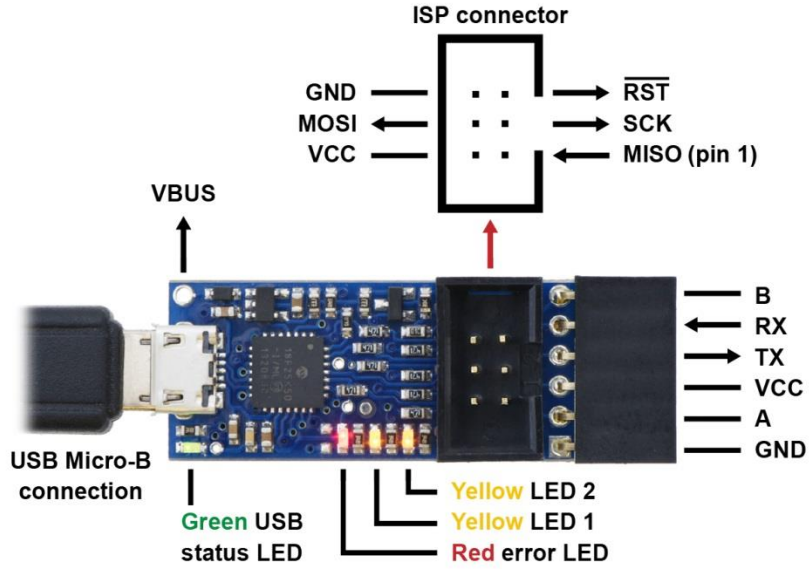
Vom Arduino zum Standalone

- Arduino als
  - Stromversorgung
  - Programmierschnittstelle

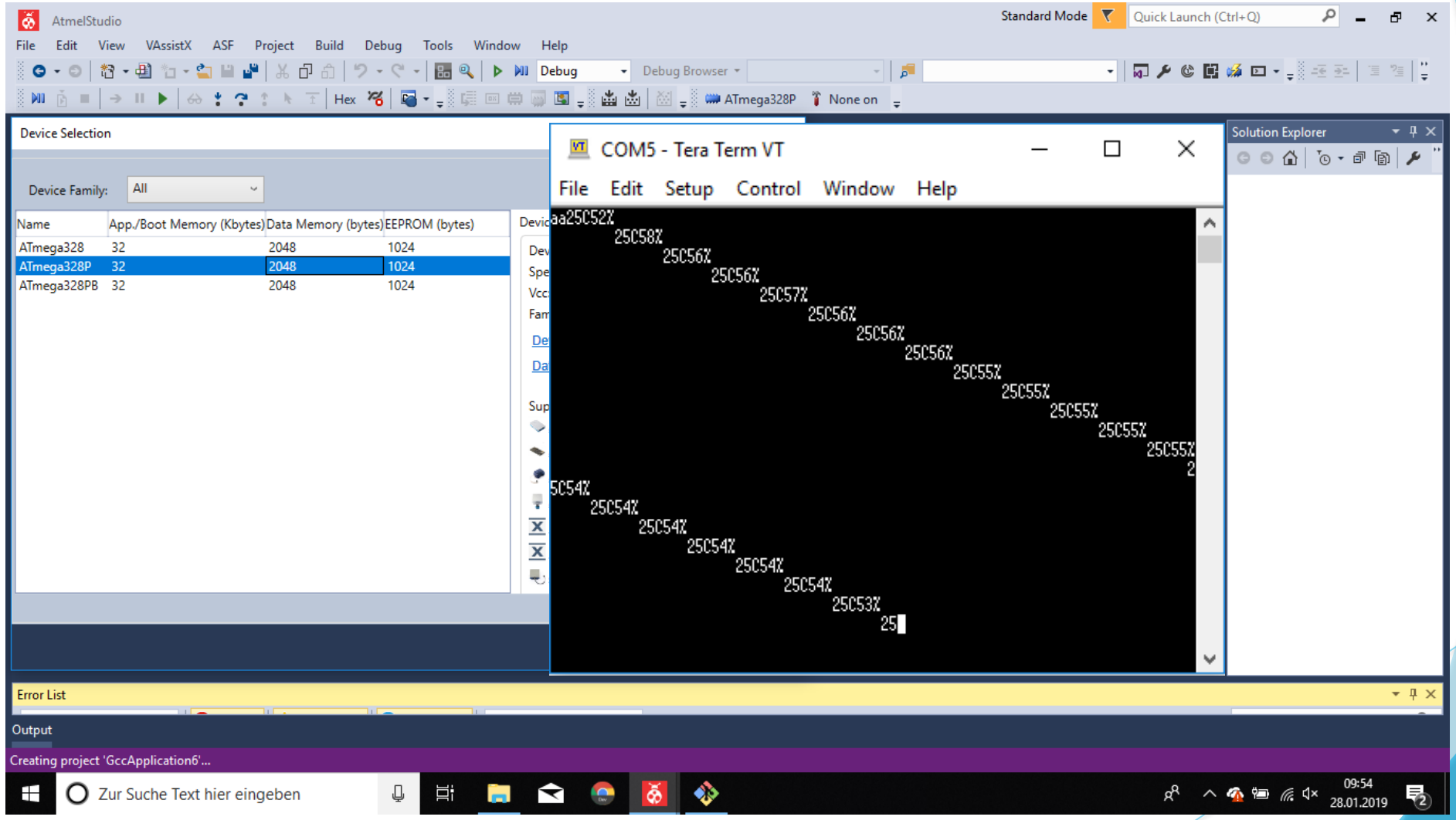


Pololu AVR-Programmer

- ISP (In-System Programmer)
- 2 Ports für
  - Programmieren
  - Serielle Ausgabe



Atmel Studio und Tera Term



Blink-Sketch

```c
#define F_CPU 8000000 //Definiert den Takt des internen Oscilllator
#include <avr/io.h> //Übergibt dem Compiler den Controllertyp
#include <util/delay.h> //benötigt für delay

int main(void)
{
    DDRD |= (1<<PORTD1); //setzt Port D1 als OUTPUT

    while (1)
    {
        PORTD |= (1<<PORTD1); //setzt Port D1 auf HIGH
        _delay_ms(500);
        PORTD &= ~(1<<PORTD1); //setzt Port D1 auf LOW
        _delay_ms(500);
    }
}
```

pinMode - Output

- Im Register definieren
- DDRD |= (1<<4)
- Nehmen wir an: DDRD = 00100101
- ausführen von DDRD |= (1<<4):
- 1 = 00000001 -> (1<<4) = 00010000
- 00100101 | 00010000 = 00110101 //4. Bit nun 1

pinMode - Input

- DDRD &= ~(1<<2)
- Wieder: DDRD = 00100101
- Ausführen von DDRD &= ~(1<<2):
- 1 = 00000001 -> (1<<2) = 00000100
- Negieren von (1<<2): ~(1<<2) = 11111011
- 00100101 & 11111011 = 00100001 //2. Bit nun 0

digitalWrite

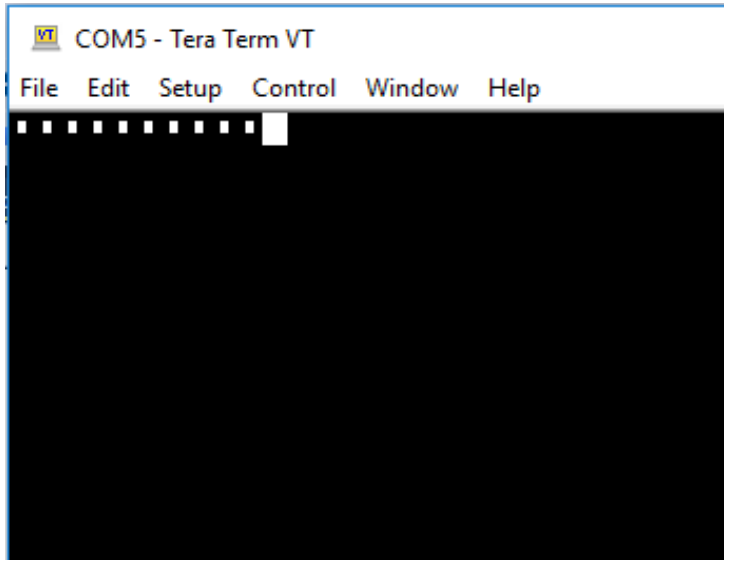
- Ins Port Register eintragen
- HIGH:
  - PORTD |= (1<<4);
- LOW:
  - PORTD &= ~(1<<4);

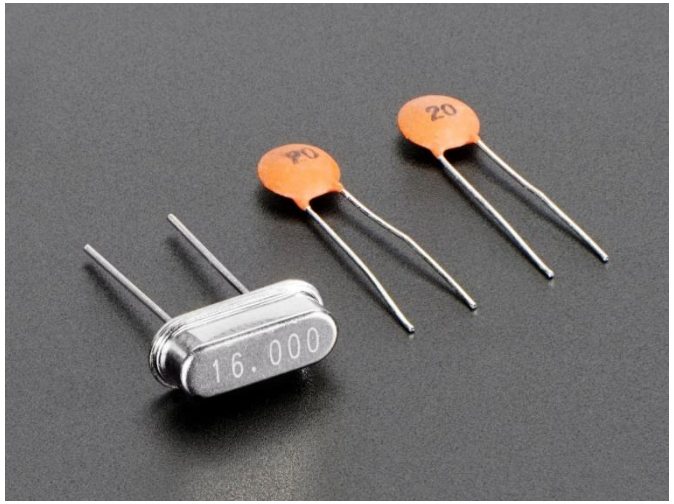
digitalRead

- (PIND & (1<<4))
- Nehmen wir an PortD4 ist High
- z.B. PIND = 10010001
- 1 = 00000001 -> (1<<4) = 00010000
- 10010001 & 00010000 = 00010000 //Das Ergebnis ist 16
- Da 16 > 0 erhalten wir true
- Wenn das Ergebnis 0 ist dann ist der Port LOW

Fehlerbehebung:

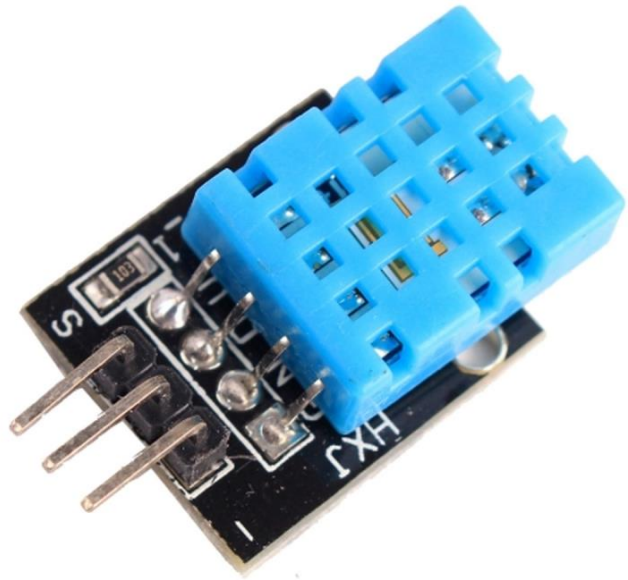
```
#include <util/delay.h>
#include <avr/io.h>

enum lh {LOW,HIGH};
enum io {INPUT,OUTPUT};
typedef uint8_t byte;
```



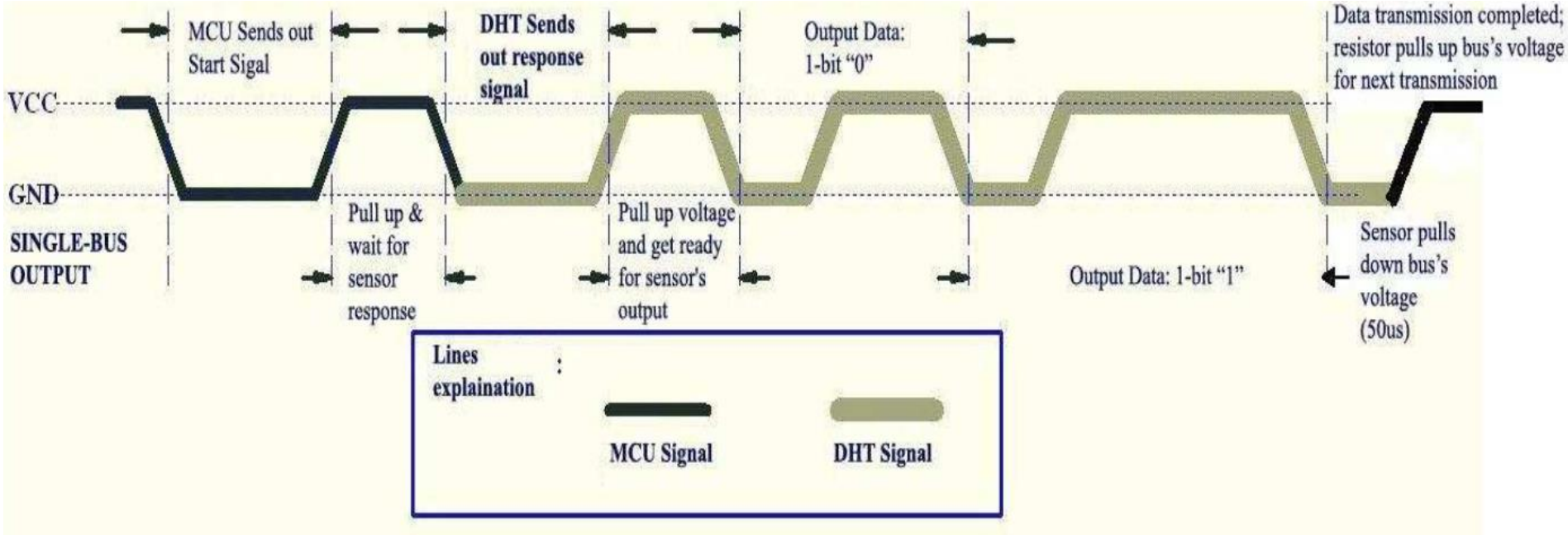


Temperatur- und Luftfeuchtigkeitssensor

- 3,3 – 5,5 Volt Versorgung
- Serial Interface(Single-Wire Two-Way)
- Kommunikationsprozess: ~4 ms
- 40bits Daten



Kommunikationsprozess:



Bit-Übertragung:

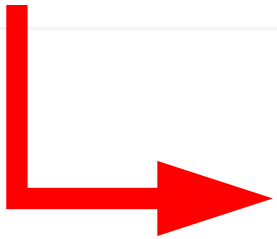
```
for (int j = 0; j < 40; j++) {
    if (confirm(pin, 50, LOW)) {
        return 102;
    }
    bool ok = false;
    int tick = 0;
    for (int i = 0; i < 8; i++, tick++) {
        if (digitalRead(pin) != HIGH) {
            ok = true;
            break;
        }
        _delay_us(10);
    }
    if (!ok) {
        return 103;
    }
    data[j] = (tick > 3? 1:0);
}
```

parse-Methode:

```
int parse(byte data[40], byte* ptemperature, byte* phumidity) {
    byte humidity = bits2byte(data);
    byte humidity2 = bits2byte(data + 8);
    byte temperature = bits2byte(data + 16);
    byte temperature2 = bits2byte(data + 24);
    byte check = bits2byte(data + 32);
    byte expect = humidity + humidity2 + temperature + temperature2;
    if (check != expect) {
        return 105;
    }
    *ptemperature = temperature;
    *phumidity = humidity;
    return 0;
}
```

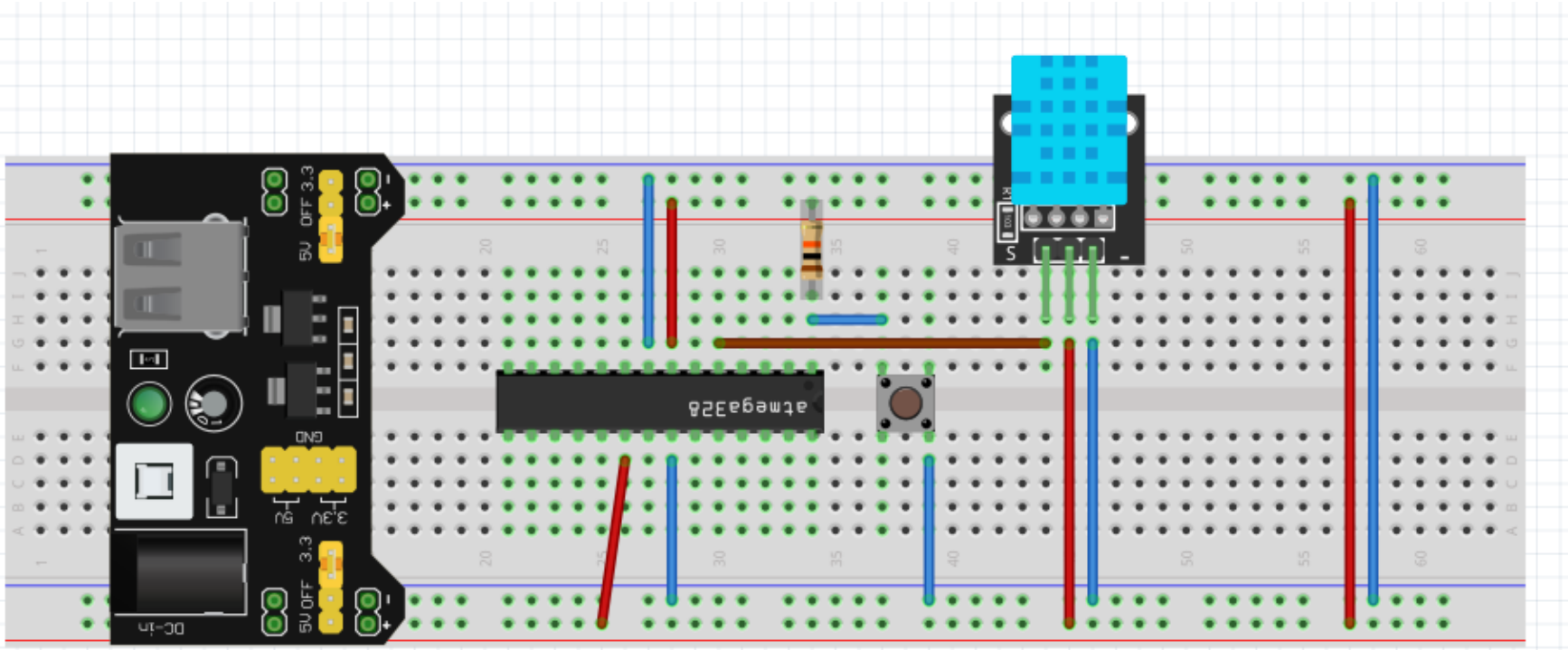
Fehlerbehebung:

```
void delay(int ms){
    for (int i = 0; i < ms;i++){
        _delay_ms(1);
    }
}
```



```
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);
_delay_ms(20);
digitalWrite(pin, HIGH);
_delay_us(30);
pinMode(pin, INPUT);
```

Derzeitiger Microcontroller



Future

- weiterer Sensor?
- Übertragung zur Station/Rasperry Pi
- Stromversorgung
- Platine