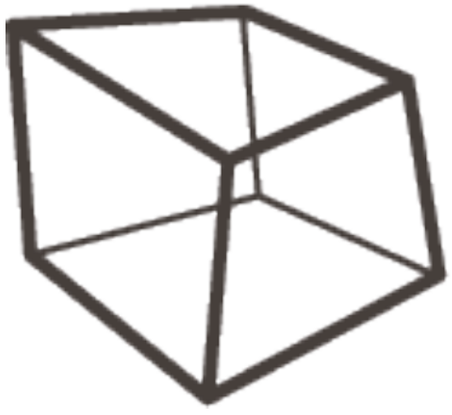
Willkommen im



MACH DEIN DING!

FABLAB ZÜRICH

zum Workshop Arduino Einführung

Von Null auf Arduino

in 4 Stunden

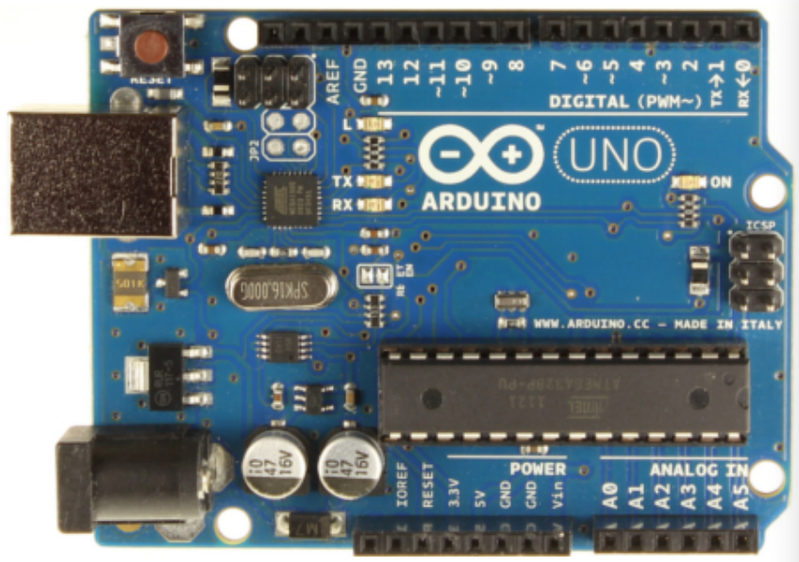
Marc Schaffer

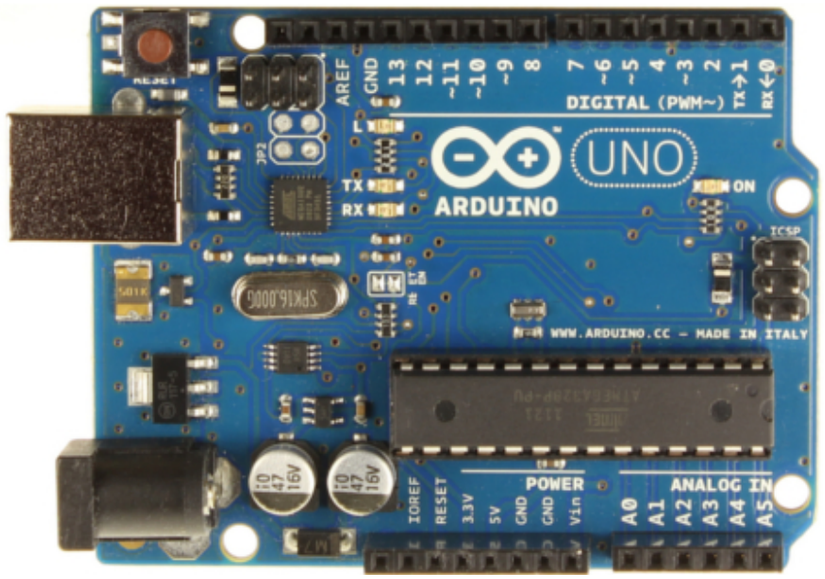
Agenda

1. Was ist ein Arduino ?
2. Sicherheitshinweise  
3. Auspacken und Inbetriebnahme
4. Was kann ein Arduino
5. Programmierung
6. Anwendungsbeispiele
7. Die grosse weite Welt der Shields
8. Elektrische Verbindungen und Aufbau
9. Fragen

1. Was ist ein Arduino ?

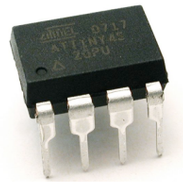
Die Verbindung von Hardware, Entwicklungssoftware mit grosser Bibliothek und einer grossen, aktiven Community.





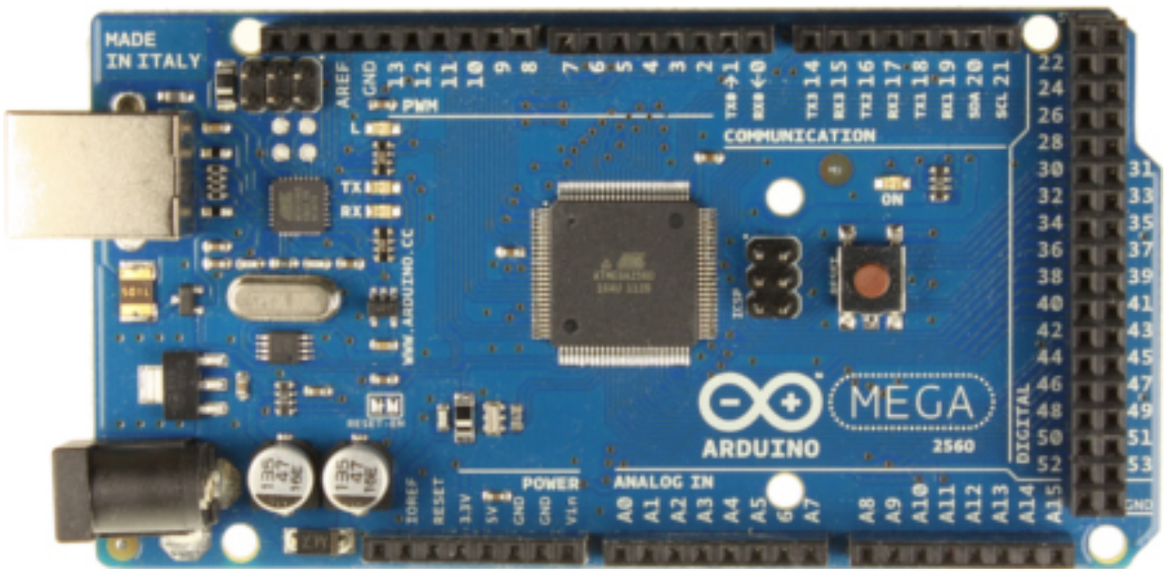
Uno

16 MHz
8 bit
Digital I/O 14



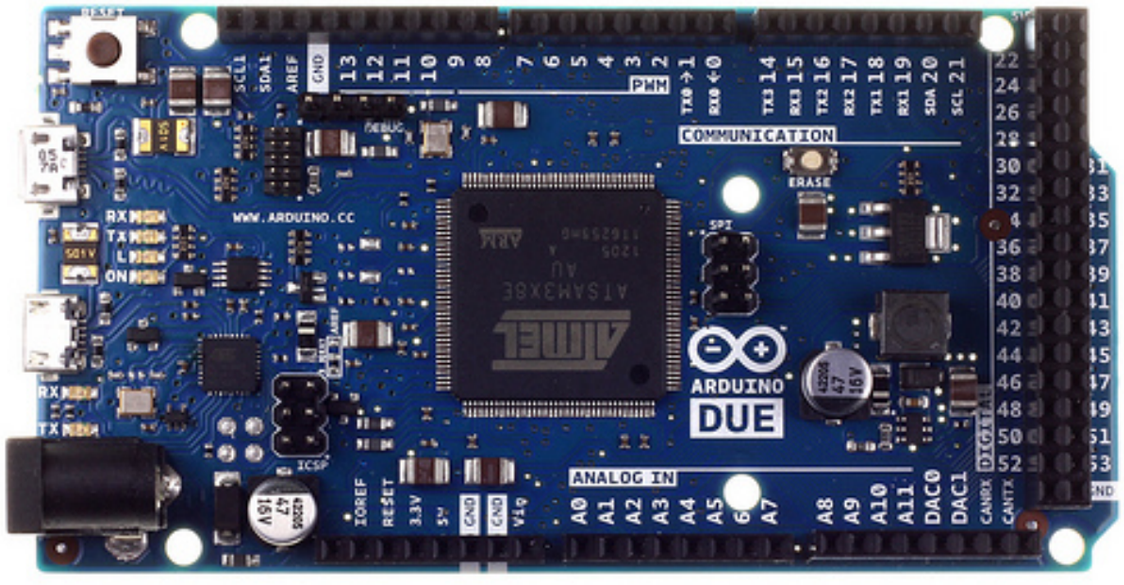
ATtiny

1 MHz ++
8 bit
Digital I/O 6



Mega

16 MHz
8 bit
Digital I/O 54



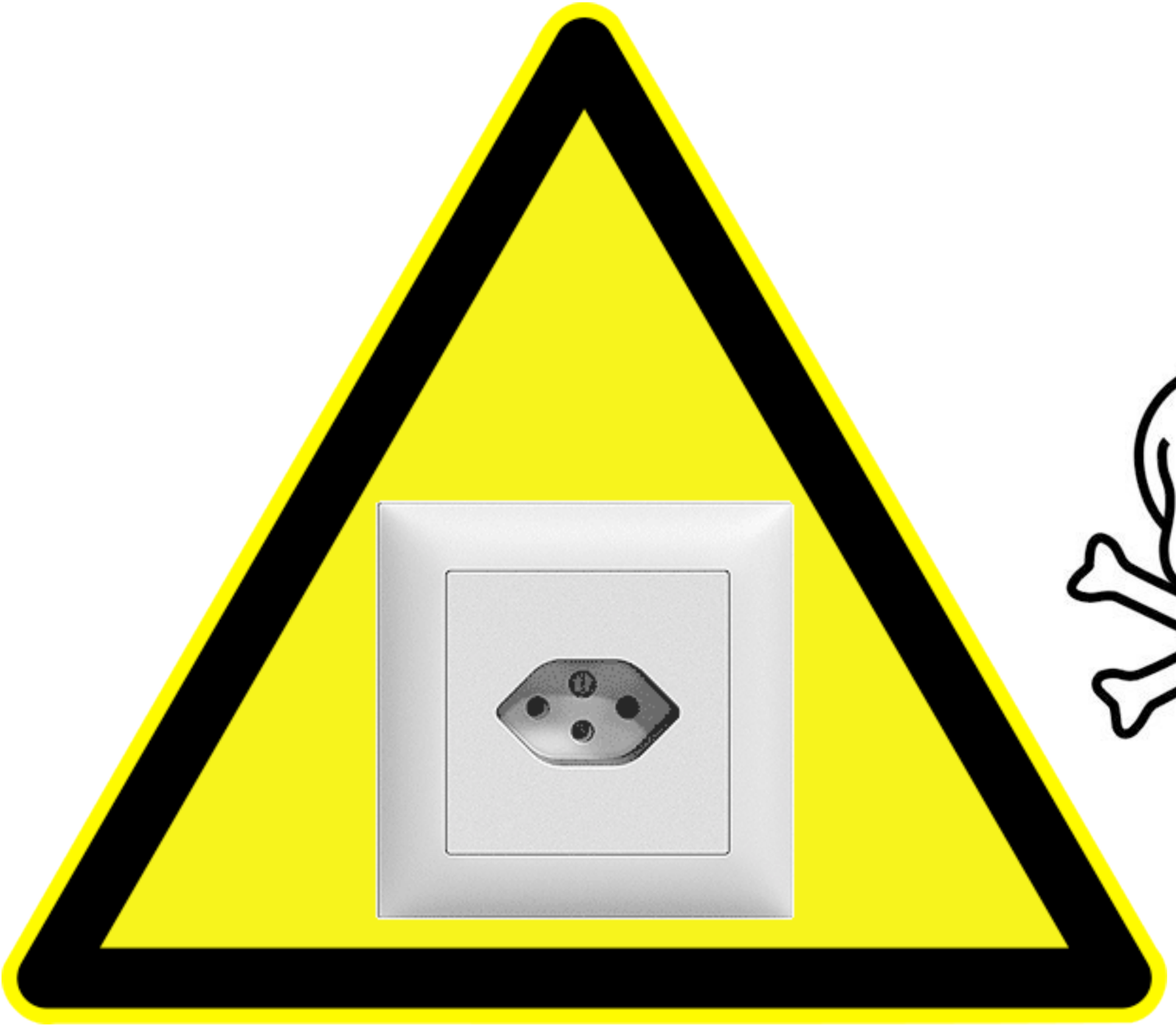
Due

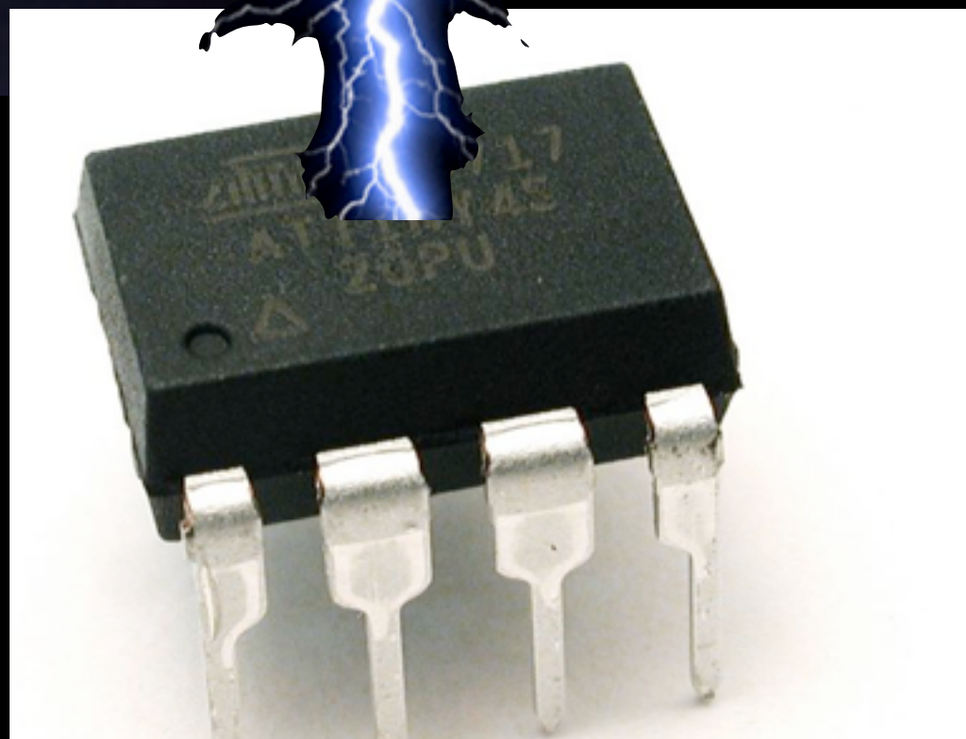
84 MHz
32 bit
Digital I/O 54

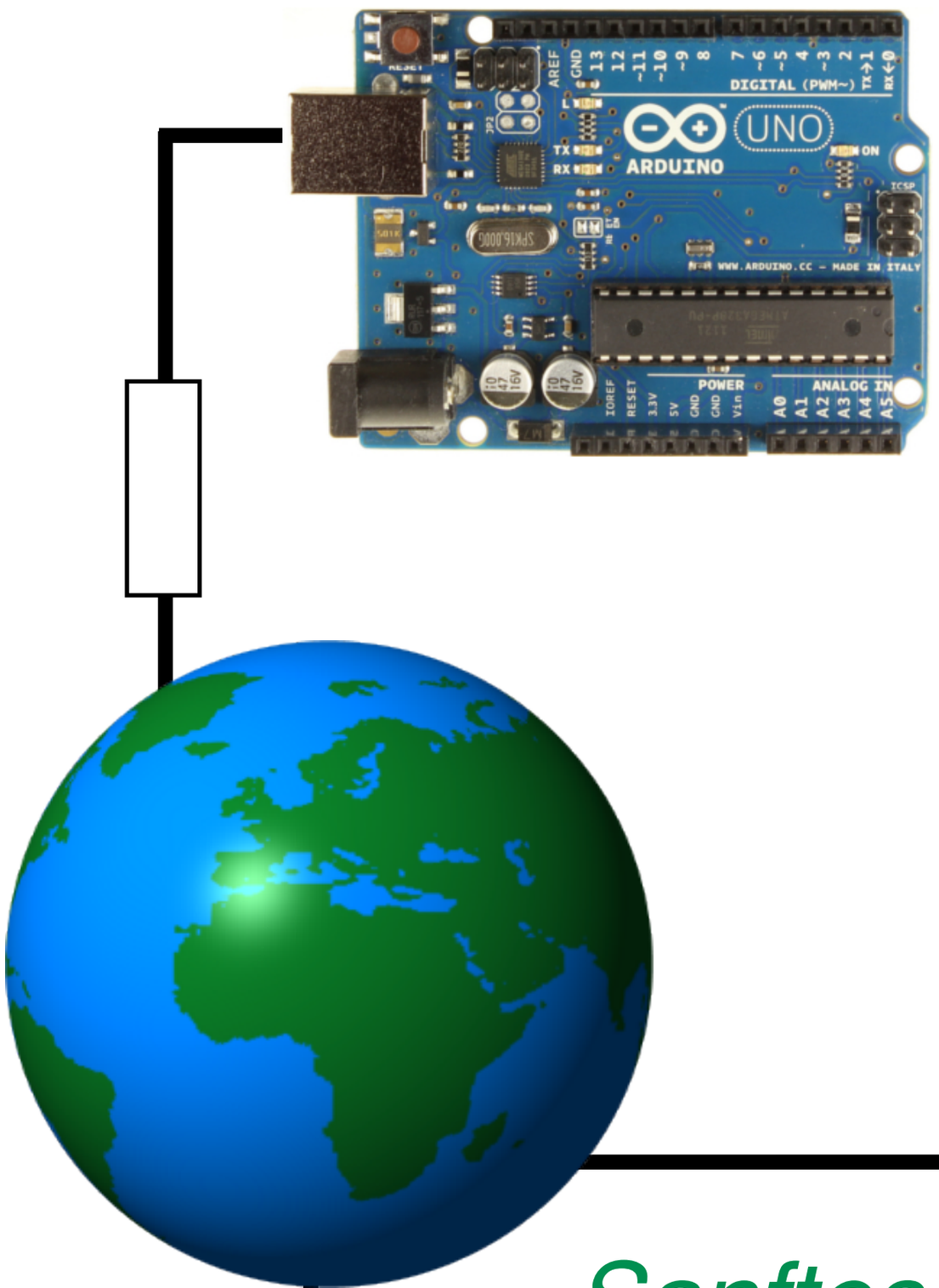
Entwickelt wurde es Ende 2005 von Prof. Massimo Banzi und David Cuartielles am *Interaction Design Institute Ivrea* (IDII) in Italien.

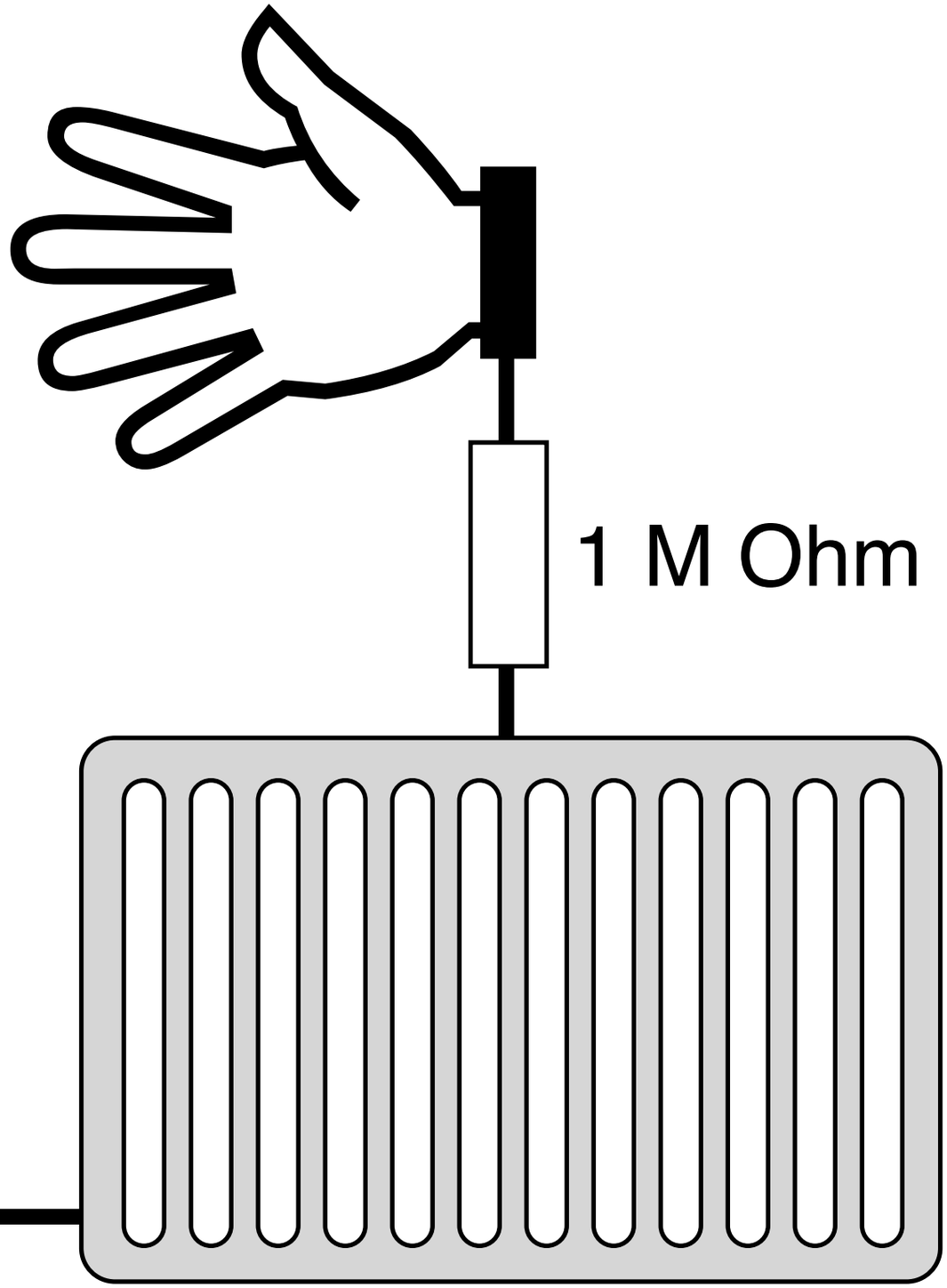
Namensgebend war ein Studentenlokal nahe des IDII, welche nach dem italienischen König Arduino (um 1000 n. Ch.) benannt wurde.

2. Sicherheitshinweise




ATMEGA16
16PU

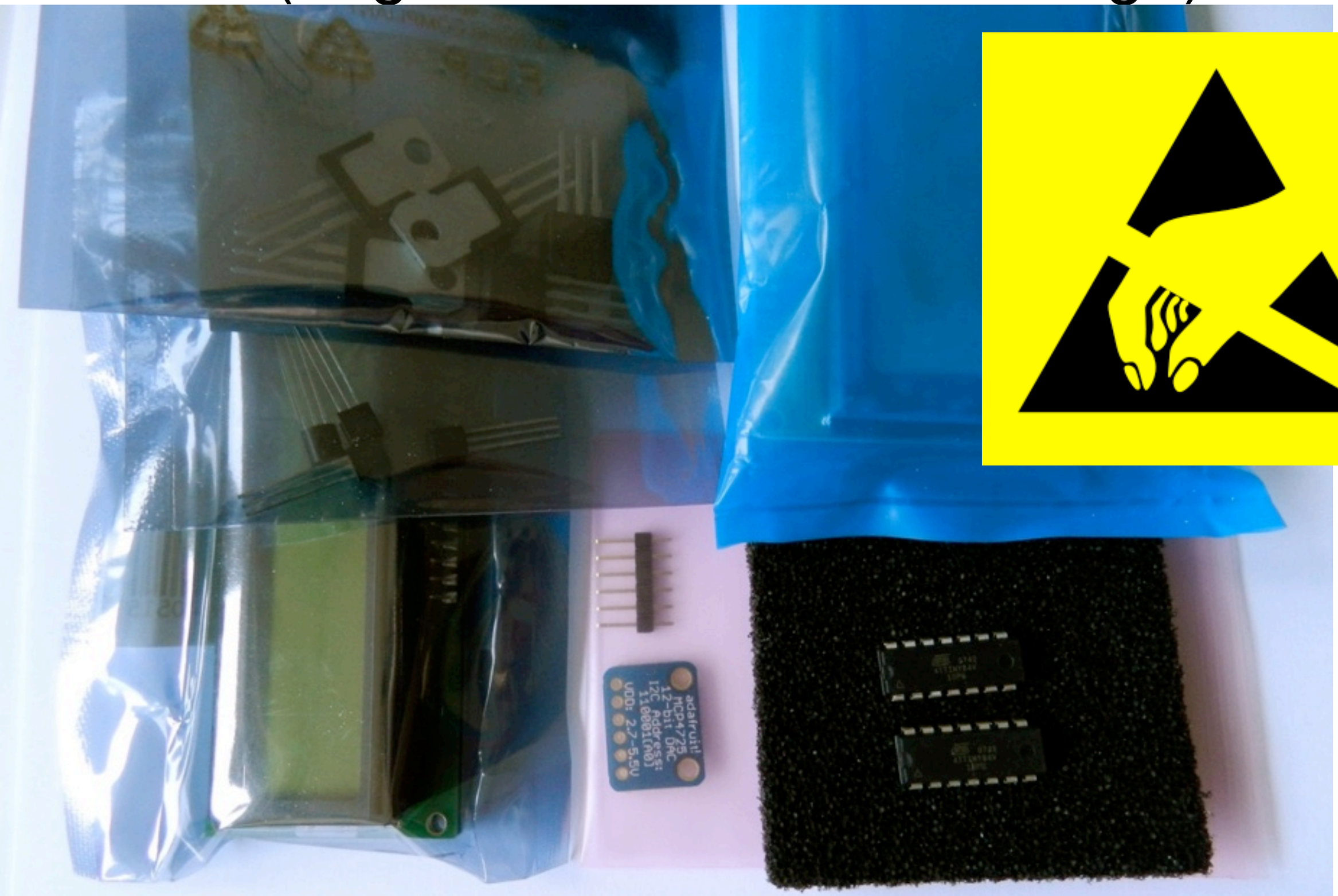




Sanftes Ableiten von Ladungen

GND / Ground / Erde

ESD Elektrostatische Entladung (engl. **E**lectro**s**tatic **D**ischarge)




DIGITAL (PWM~)
UNO
ON
RESET
POWER
GND
5V
Vin

3. Auspacken und Inbetriebnahme

Reset

GND

Digital

seriell

USB

ATmega328
- Flash 32 KB
- RAM 2 KB
- EEPROM 1 KB

7-12 V

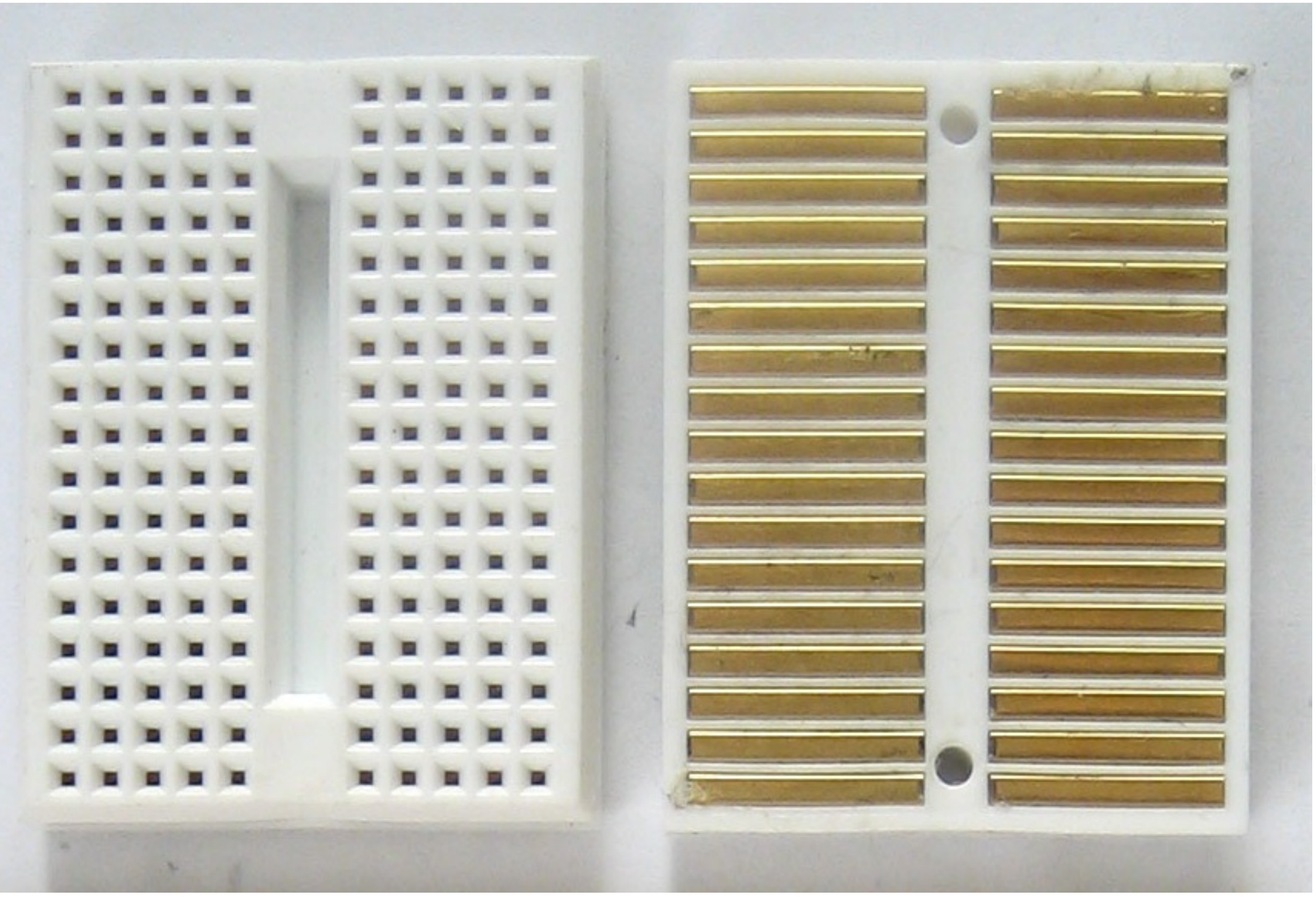
3,3 V

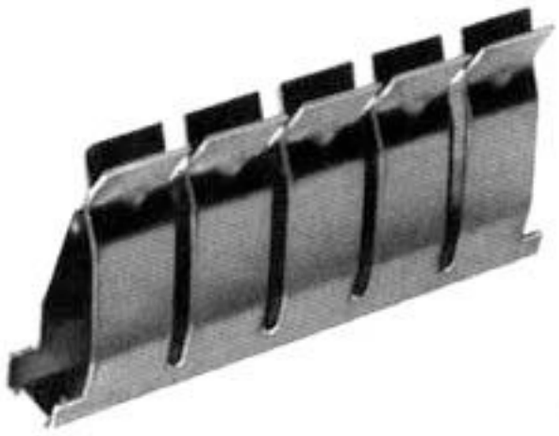
5 V

GND

Analog

Breadboard - Steckplatine





Potentiometer

Widerstände

Piezo

LED‘s

Taster

Kondensatoren

Dioden

LDR

Transistoren

Feuchtigkeitssensor

Software

http://arduino.cc/en

➔ Download

Code Editor

Blink | Arduino 1.0.4

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Kompilierung abgeschlossen.

Binäre Sketchgröße: 1.084 Bytes (von einem Maximum von 30.720 Bytes)

1 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A700fm

Kompilieren

Senden

Laden

Speichern

Status

Fehler

Anschluss

Kommentarblock

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */
```

Kommentarzeile

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
```

Wichtig !!!

Deklaration + Wertzuweisung

```
int led = 13;
```

Setup (läuft einmal)

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
} //setup
```

Loop (läuft immer)

```
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
} //loop
```

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT| INPUT_PULLUP
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due only

- analogReadResolution()
- analogWriteResolution()

4. Was kann ein Arduino ?

Problem lösen oder Arbeit abnehmen

Also Überwachen, Steuern und Regeln:

Überwachen, Steuern und Regeln:

Überwachen: Alarmanlage
(Warten auf Magnetschalter am Fenster)

Steuern: Treppenlicht
(Befehle abarbeiten nach Tastendruck)

RegeIn: Herdplatte
(Aktion+Reaktion
von Temperatursensor + Relais/SSR)

EVA

Eingabe

Verarbeitung

Ausgabe

(Engl. IPO Input-Processing-Output)

EVA

Eingabe: Taste, Schalter, Tastatur, Kontakt, Messung, Sensor, Signal, Telefonanruf usw.

Verarbeitung: Programm

Ausgabe: Leuchte, Anzeige, Ton, Schalter, Servo, Roboter, Drucker, Relais, Log (Speicher), Tweet, SMS usw.

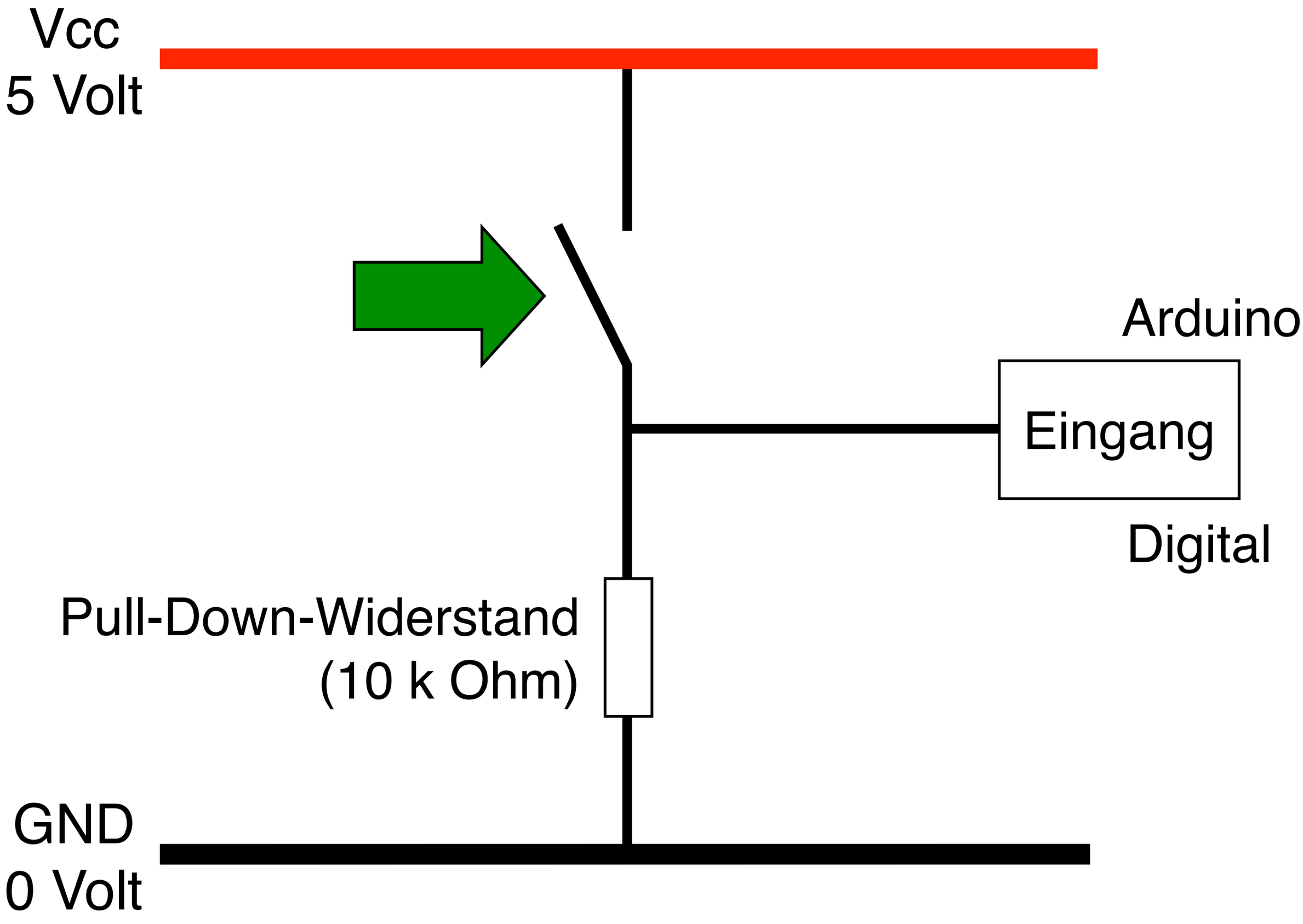
Vcc
5 Volt

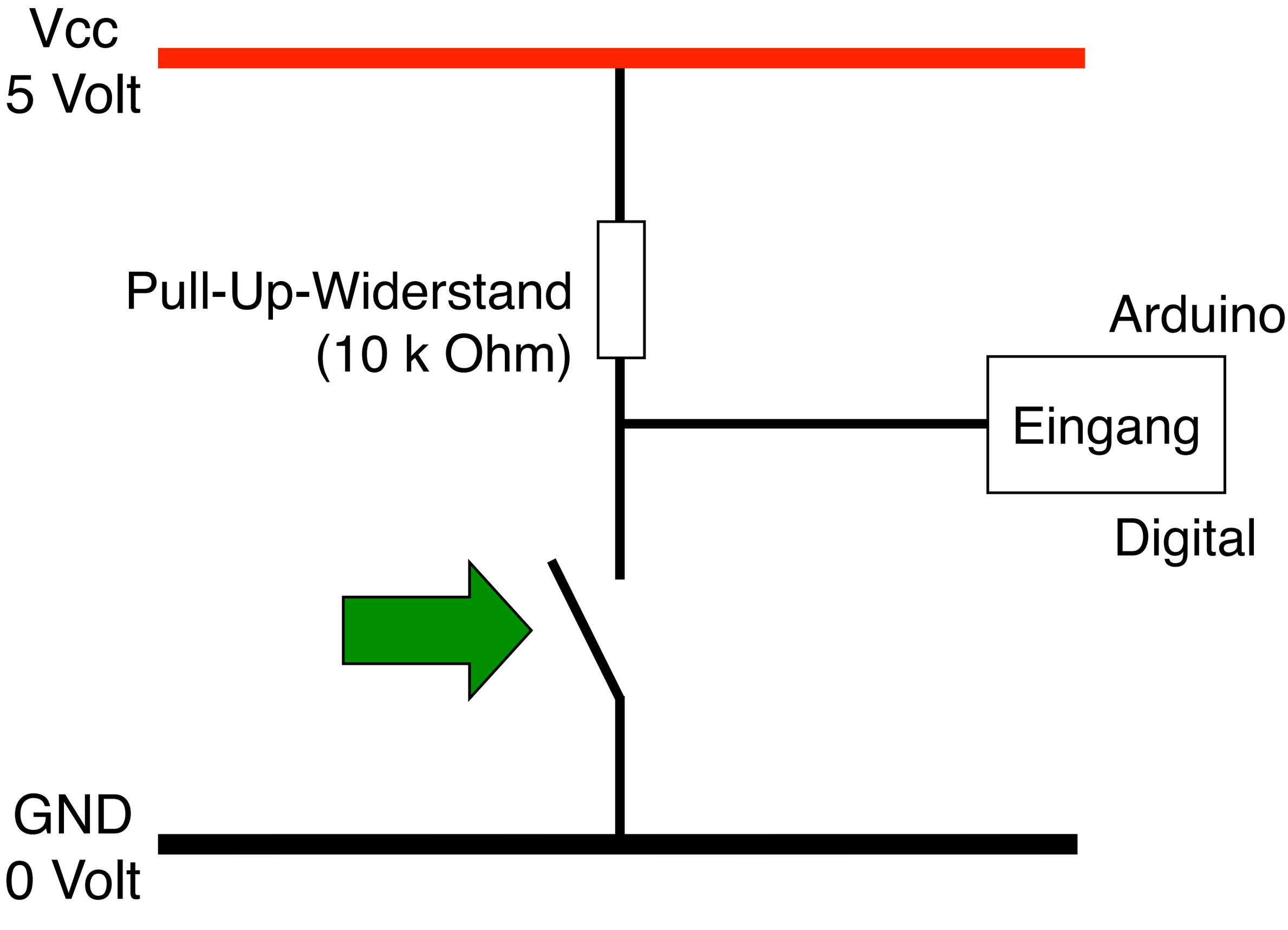
Arduino

Eingang

Digital

GND
0 Volt


Vcc
5 Volt
Arduino
Eingang
Digital
Pull-Down-Widerstand
(10 k Ohm)
GND
0 Volt


Vcc
5 Volt
Pull-Up-Widerstand
(10 k Ohm)
Arduino
Eingang
Digital
GND
0 Volt

Vcc
5 Volt

Integriert und zuschaltbar
Pull-Up-Widerstand
(20 k Ohm)

Arduino

Eingang

Digital

```
pinMode(2, INPUT_PULLUP);
```

Invertiert !

GND
0 Volt

Vcc
5 Volt

1023 Relativ !

Arduino

Eingang

Analog

Potentiometer
„Poti“

Schleif-Widerstand
(10 k Ohm)

0

GND
0 Volt

Vcc
5 Volt

Widerstand
(47 k Ohm)

1023 Relativ !

Arduino

Eingang

Analog

LDR

0

GND
0 Volt

Vcc
5 Volt

HIGH / 1 / TRUE

Arduino

Ausgang

13

Digital

Begrenzungs-Widerstand
(? Ohm)

GND
0 Volt

LOW / 0 / FALSE

Ohmsches Gesetz

$$R = \frac{U}{I} \qquad U = R * I \qquad I = \frac{U}{R}$$

R	linearer elektrischer Widerstand	Ohm
U	Spannung	Volt
I	Stromstärke	Ampere

Ohmsches Gesetz

$$R = \frac{U}{I} \qquad U = R * I \qquad I = \frac{U}{R}$$

$$5 = \frac{10}{2} \qquad 10 = 5 * 2 \qquad 2 = \frac{10}{5}$$

R	linearer elektrischer Widerstand	Ohm
U	Spannung	Volt
I	Stromstärke	Ampere

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09

28. Electrical Characteristics

28.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

28.2 DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5 -0.5		$0.2V_{CC}$(1) $0.3V_{CC}$(1)	V
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$(2) $0.6V_{CC}$(2)		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.1V_{CC}$(1)	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.8V_{CC}$(2) $0.7V_{CC}$(2)		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.1V_{CC}$(1)	V
V_{IH2}	Input High Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	$0.9V_{CC}$(2)		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5 -0.5		$0.2V_{CC}$(1) $0.3V_{CC}$(1)	V
V_{IH3}	Input High Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$(2) $0.6V_{CC}$(2)		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(3) except RESET pin	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.9 0.6	V
V_{OH}	Output High Voltage(4) except Reset pin	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.3			V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA

Microcontroller:
DC Current per I/O Pin ..... 40.0 mA
DC Current VCC and GND Pins..... 200.0 mA

LED:
DC Forward Current..... 20 mA

$$R = \frac{U \quad 5\ \text{Volt}_{\ (USB)}}{I \quad 10\ \text{mA}} = 500$$

R	linearer elektrischer Widerstand	Ohm
U	Spannung	Volt
I	Stromstärke	Ampere

Vcc
5 Volt

HIGH / 1 / TRUE

Arduino

Ausgang

13

Digital

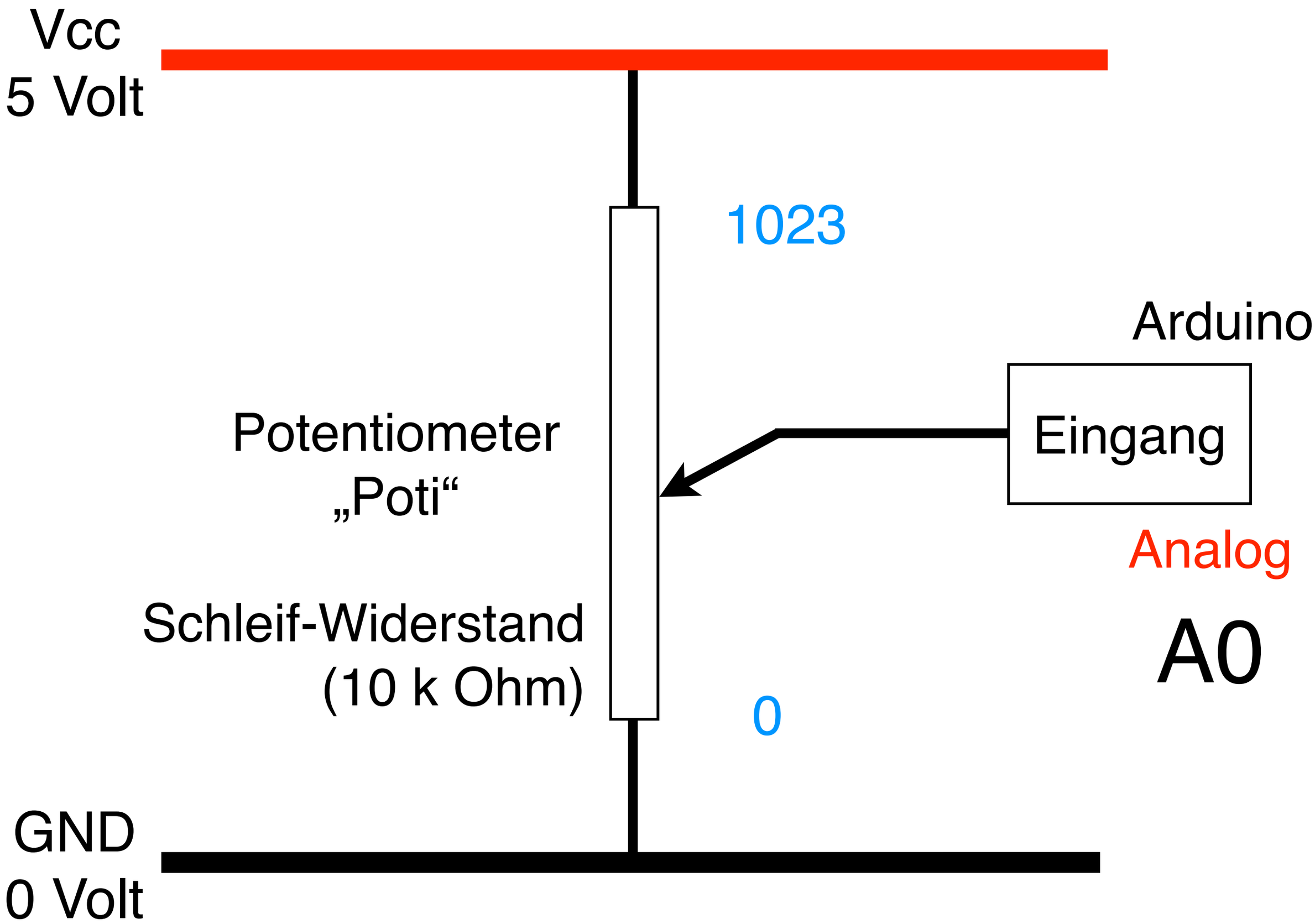
Begrenzungs-Widerstand
(470 Ohm)

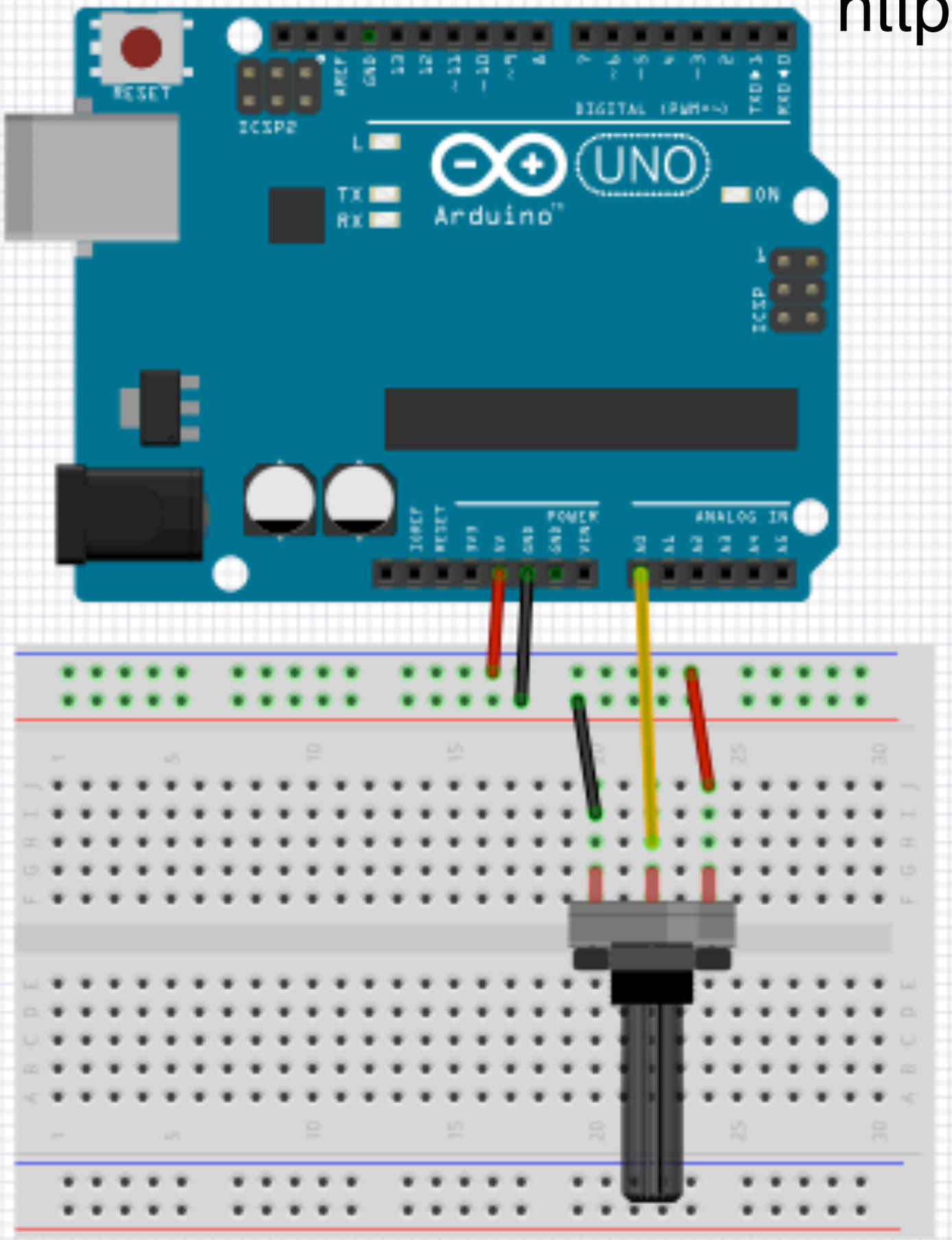
GND
0 Volt

LOW / 0 / FALSE

5. Programmierung

oder wie formalisiere ich ein Problem


Vcc
5 Volt
1023
Arduino
Eingang
Analog
A0
Potentiometer
„Poti“
Schleif-Widerstand
(10 k Ohm)
0
GND
0 Volt



```
Sensorwert = analogRead(A0);
```

```
Sensorwert = analogRead(A0);

if (Sensorwert > 500)
{
    digitalWrite(ledPin, HIGH);
} //if
else
{
    digitalWrite(ledPin,LOW);
} //else
```

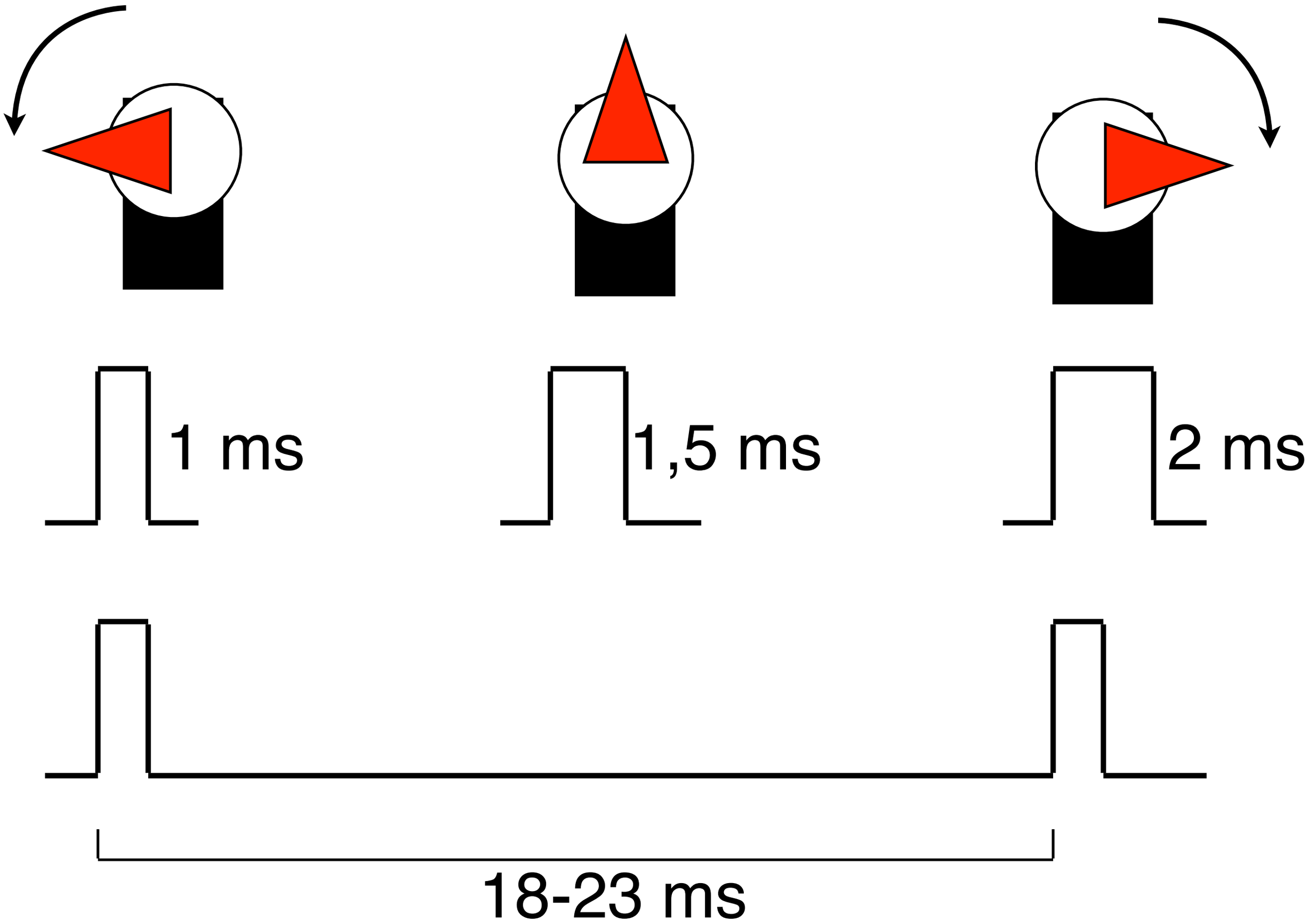
```
Sensorwert = analogRead(A0);

if(Sensorwert > 500)
{
    digitalWrite(ledPin, HIGH);
} //if
else
{
    digitalWrite(ledPin,LOW);
} //else
```

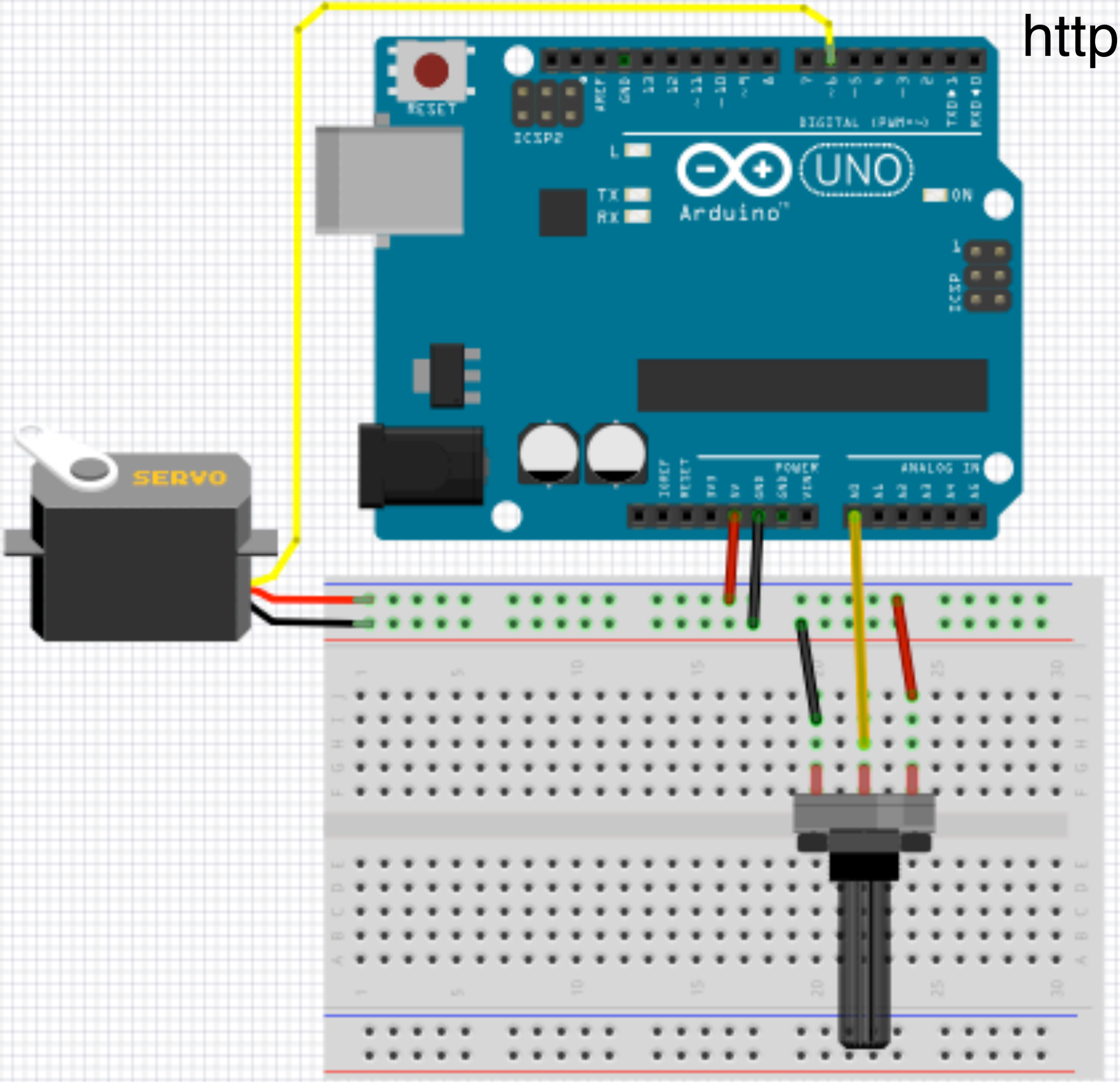
```
for (int hell = 0; hell < 255; hell ++)
{
    analogWrite(11, hell );
    delay(2);
} //for
```

```
for (int hell = 255; hell >= 0; (hell=hell-10)   )
{
    analogWrite(11, hell );
    delay(2);
} //for
```

6. Anwendungsbeispiele

Mechatronic Sourcebook,
Newton C. Braga,
Thomson Delmar Learning


1 ms
1,5 ms
2 ms
18-23 ms



Beispiele: Servo/Knob

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

void loop() {
        ...
        Serial.print("Wert = ");
        Serial.print(Variable);
        Serial.println("");
} //loop
```

AnalogInOutSerial - Serial Monitor

Serial → "Serial Monitor"

Win: PuTTY
HyperTerminal

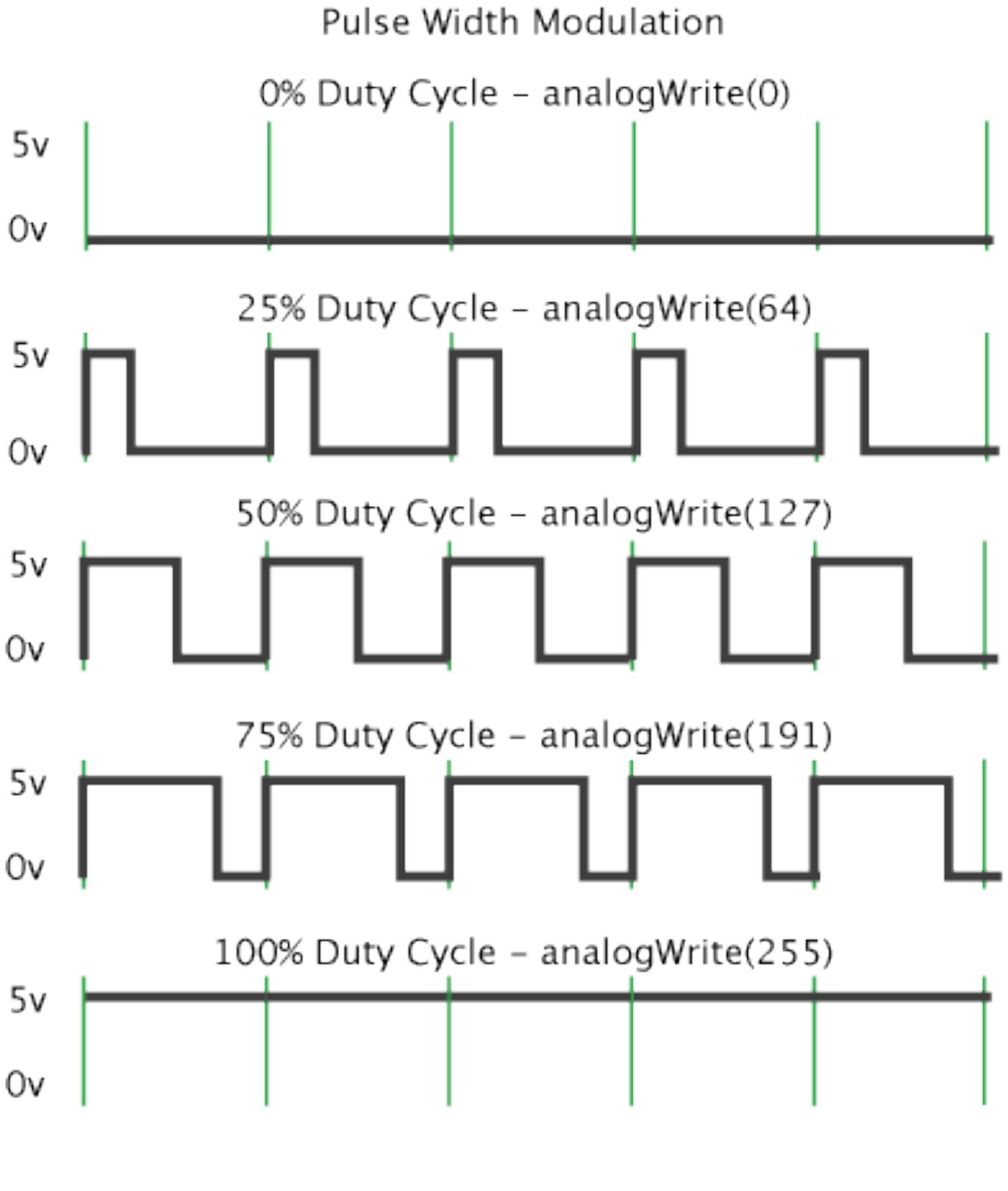
Mac: CoolTerm

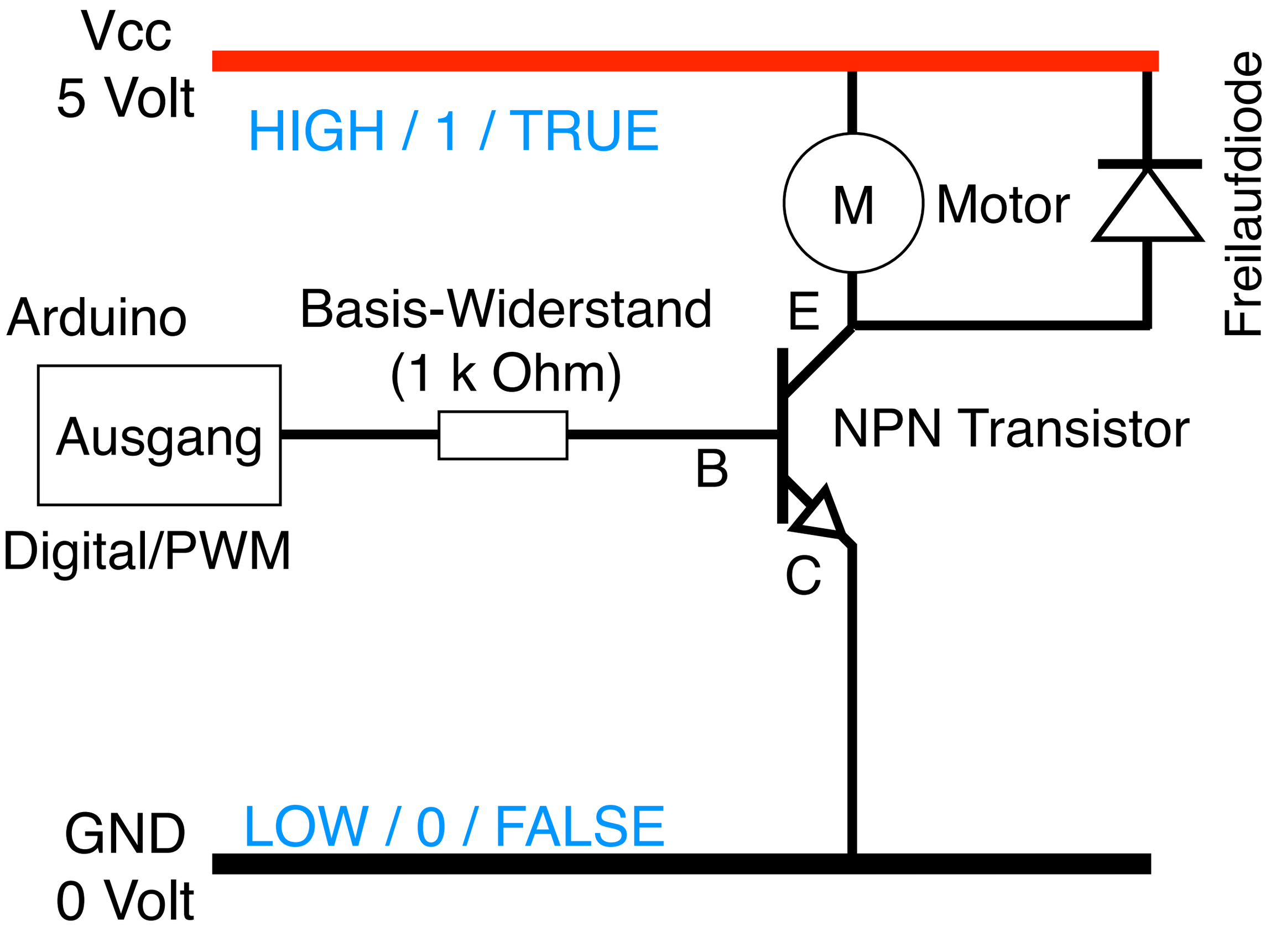
Linux: Minicom
Screen

```
String serialString = "";          // a string to hold incoming data
boolean stringComplete = false;    // whether the string is complete
long a=0;
char SerialChars[200];

void setup() {
  Serial.begin(9600);  // initialize serial:
  serialString.reserve(200);  // reserve 200 bytes for the inputString:
  SerialChars[0] = '\0';
  Serial.print("Arduino ready.\r\n$ ");
} //setup

void loop() {
    if (stringComplete) {  // Wenn Enter gedrueckt
        stringComplete = false;

        if (serialString == "AT\r\n") {
            Serial.println("OK");
        } //AT

        serialString.toCharArray(SerialChars, 200);
        if (sscanf(SerialChars, "ServoN=%d", &a) == true)
        {
            Serial.print("ServoN = ");
            Serial.print(a);
            Serial.println("");
        } //ServoN

        serialString = "";
    }//if stringComplete
}//loop

void serialEvent() {
    while (Serial.available()) {
        char inChar = (char)Serial.read();     // get the new byte:
        serialString += inChar;     // add it to the inputString:
        Serial.print(inChar); // =Echo
        if (inChar == '\n') {  //newline (Return != Enter)
            stringComplete = true;
        }
    }//while
}//serialEvent
```

```
String serialString = "";          // a string to hold incoming data
boolean stringComplete = false;    // whether the string is complete
long a=0;
char SerialChars[200];

void setup() {
  Serial.begin(9600);  // initialize serial:
  serialString.reserve(200);  // reserve 200 bytes for the inputString:
  SerialChars[0] = '\0';
  Serial.print("Arduino ready.\r\n$ ");
} //setup

void loop() {
    if (stringComplete) {  // Wenn Enter gedrueckt
        stringComplete = false;

        if (serialString == "AT\r\n") {
            Serial.println("OK");
        } //AT

        serialString.toCharArray(SerialChars, 200);
        if (sscanf(SerialChars, "ServoN=%d", &a) == true)
        {
            Serial.print("ServoN = ");
            Serial.print(a);
            Serial.println("");
        } //ServoN

        serialString = "";
    }//if stringComplete
}//loop

void serialEvent() {
    while (Serial.available()) {
        char inChar = (char)Serial.read();     // get the new byte:
        serialString += inChar;     // add it to the inputString:
        Serial.print(inChar); // =Echo
        if (inChar == '\n') {  //newline (Return != Enter)
            stringComplete = true;
        }
    }//while
}//serialEvent
```

IRQ


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v


Vcc
5 Volt
HIGH / 1 / TRUE
M
Motor
Freilaufdiode
Arduino
Basis-Widerstand
(1 k Ohm)
E
Ausgang
B
NPN Transistor
C
Digital/PWM
GND
0 Volt
LOW / 0 / FALSE

Vcc
5 Volt

HIGH / 1 / TRUE

Freilaufdiode

M Motor

Arduino

Basis-Widerstand
(1 k Ohm)

E

NPN Transistor

Ausgang

B

Digital/PWM

C

Lebenswichtig !

GND
0 Volt

LOW / 0 / FALSE

Vcc
5 Volt

HIGH / 1 / TRUE

M Motor

Freilaufdiode

D

Arduino

Ausgang

G

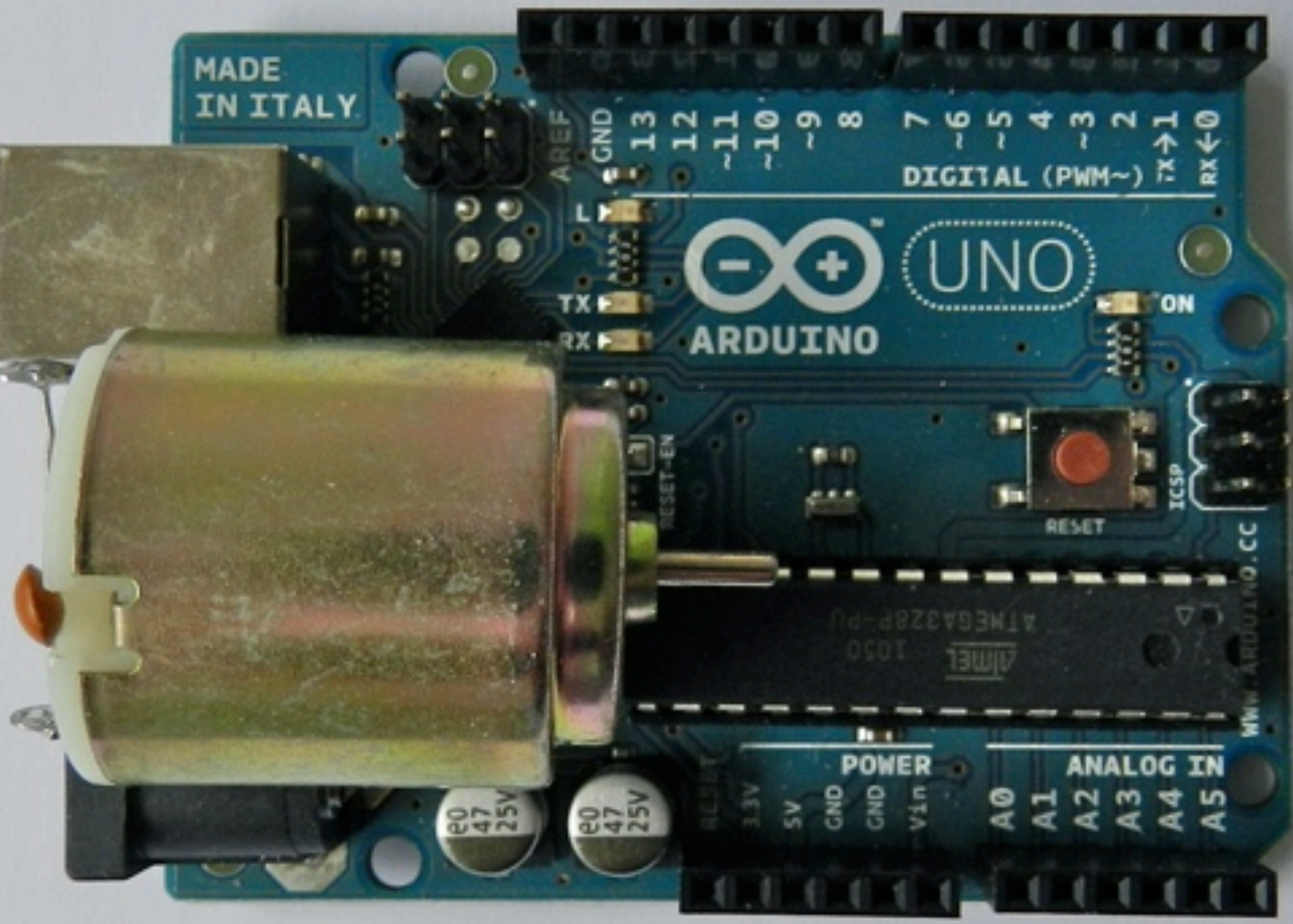
Feldeffekt Transistor
N-Kanal-MOSFET

S

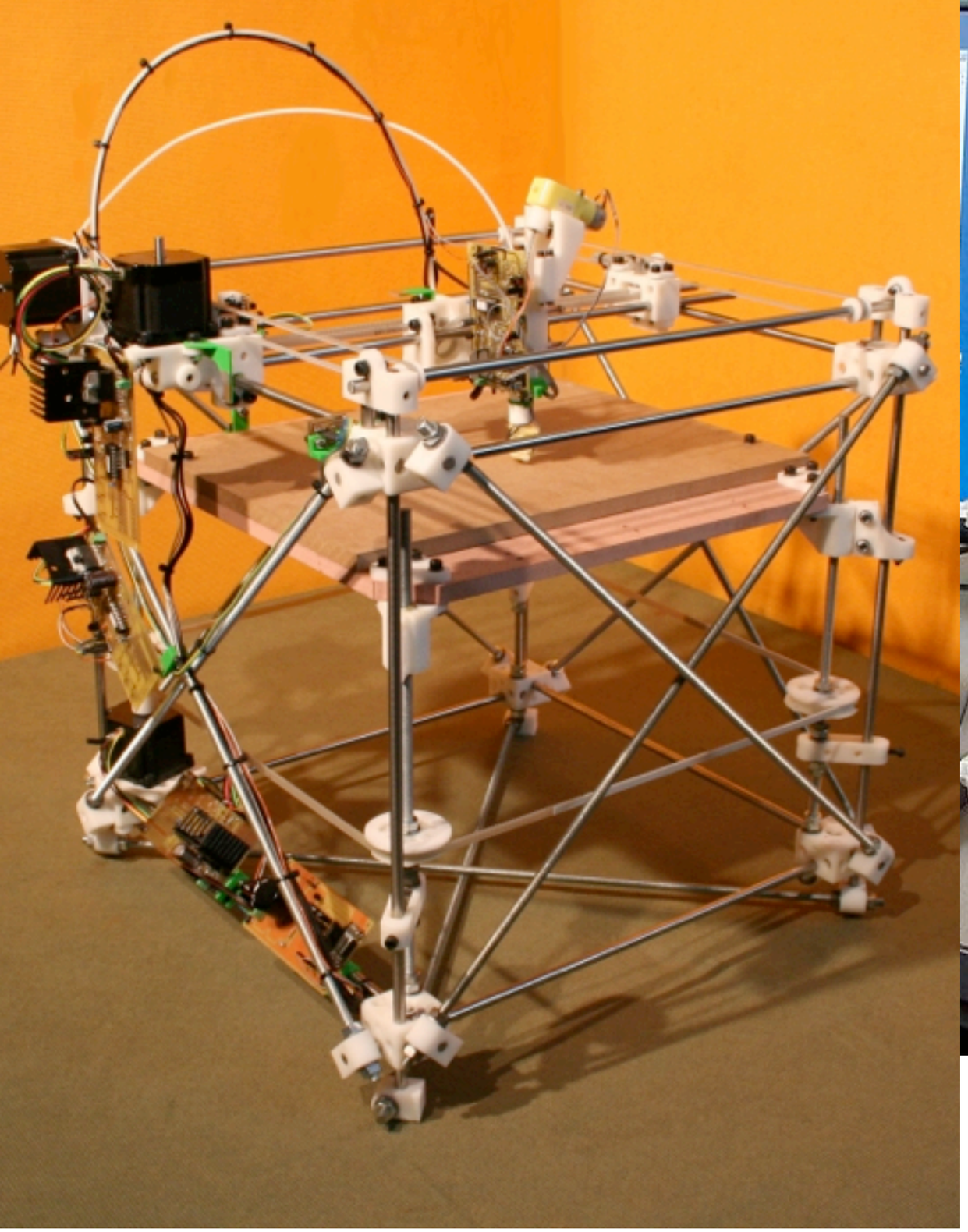
Digital/PWM

GND
0 Volt

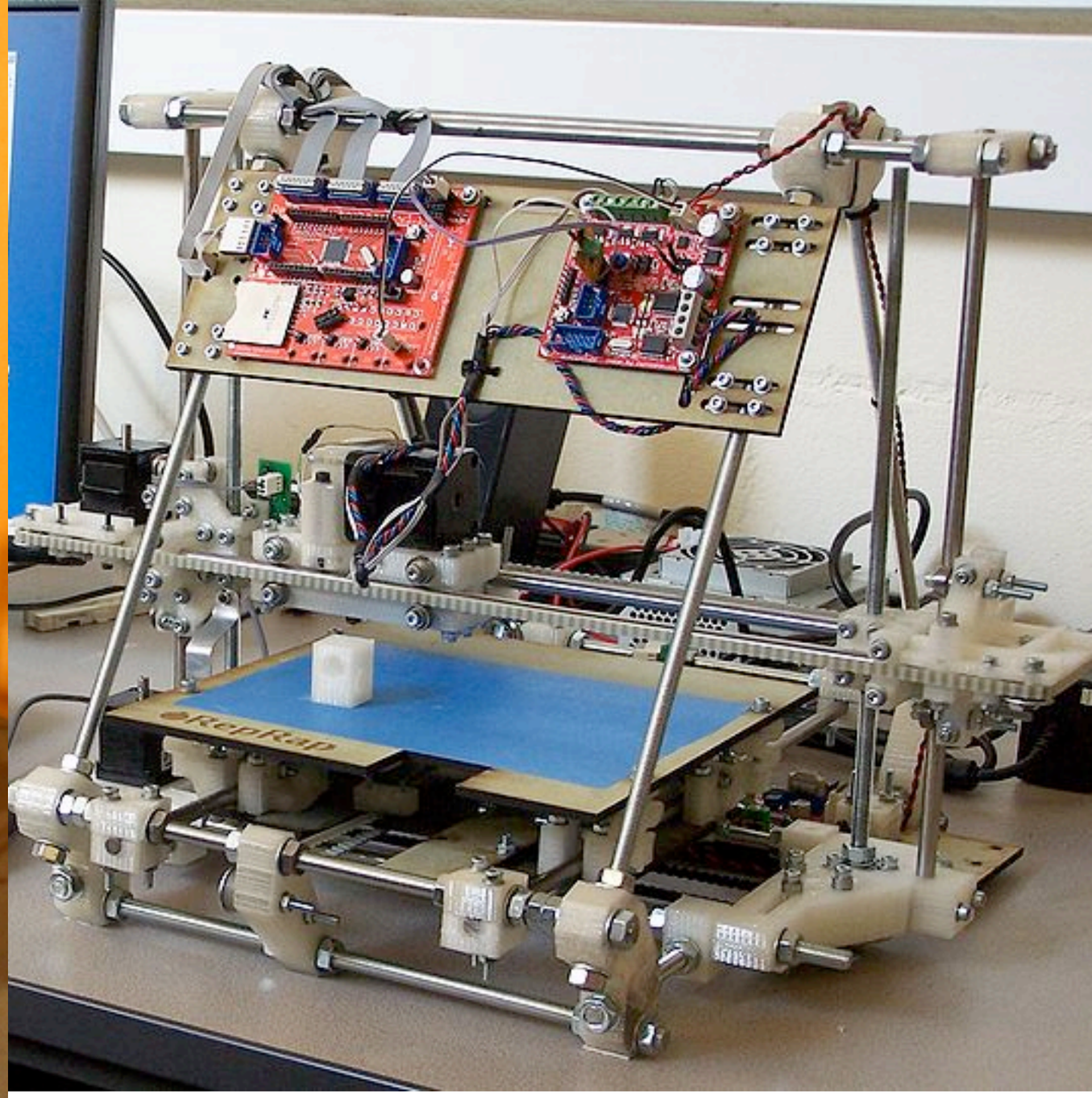
LOW / 0 / FALSE


MADE
IN ITALY
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
ICSP
RESET
POWER
ANALOG IN
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
ATMEGA328P-PU
ATMEL
1050
WWW.ARDUINO.CC
220
47
25V

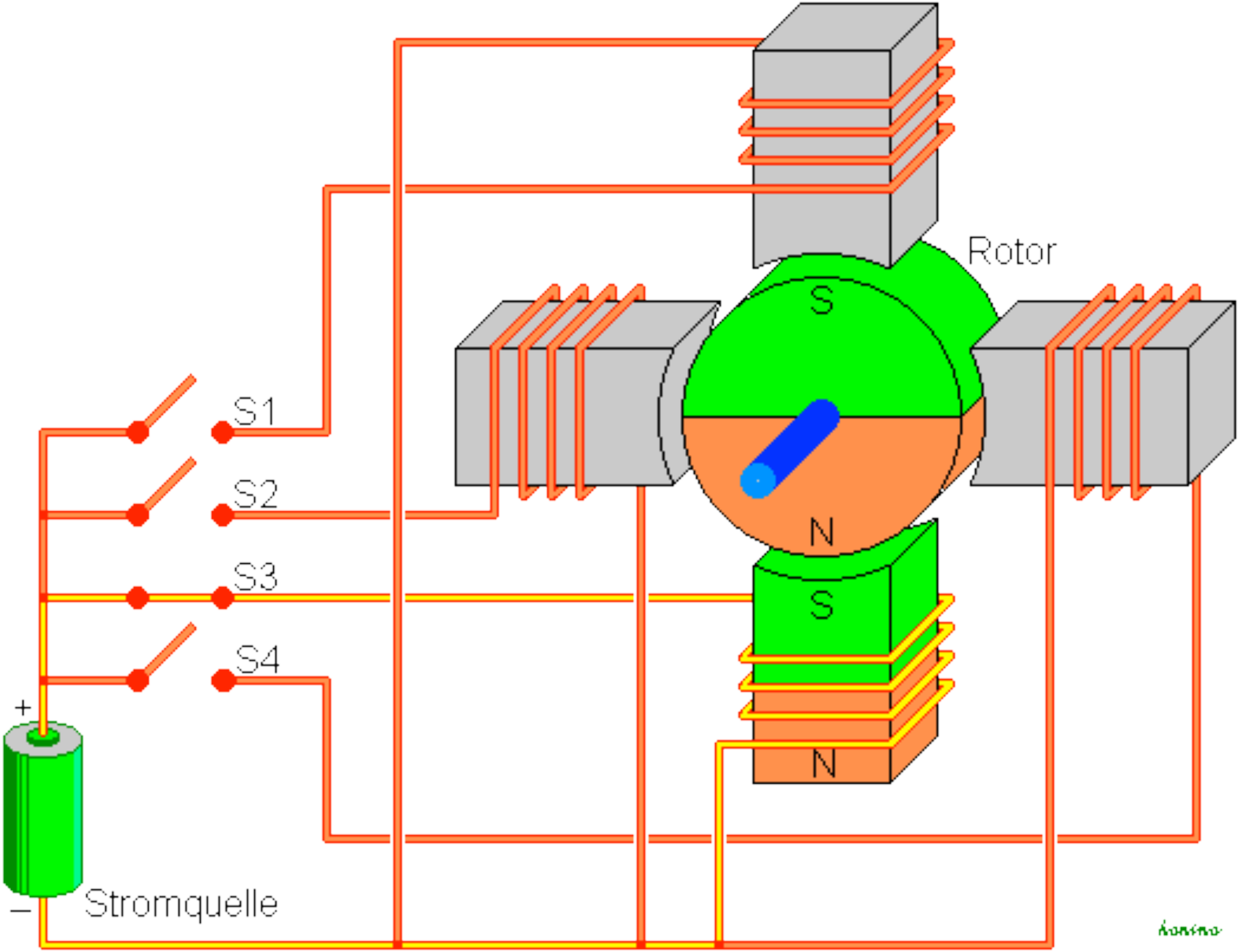
"Very Large Array" Foto: dpa



RepRap Version 1.0
(Darwin)



RepRap version 2.0
(Mendel)



RepRap Arduino Mega Pololu Shield 1.4

Extruder 1

Extruder 2

HBP Thermistor

Extruder Heater

Extruder Thermistor

Extruder Heater 2

Extruder Thermistor

HBP Heater

D10

D09

D08

11A

5A

SDRAMPS

X Axis

Y Axis

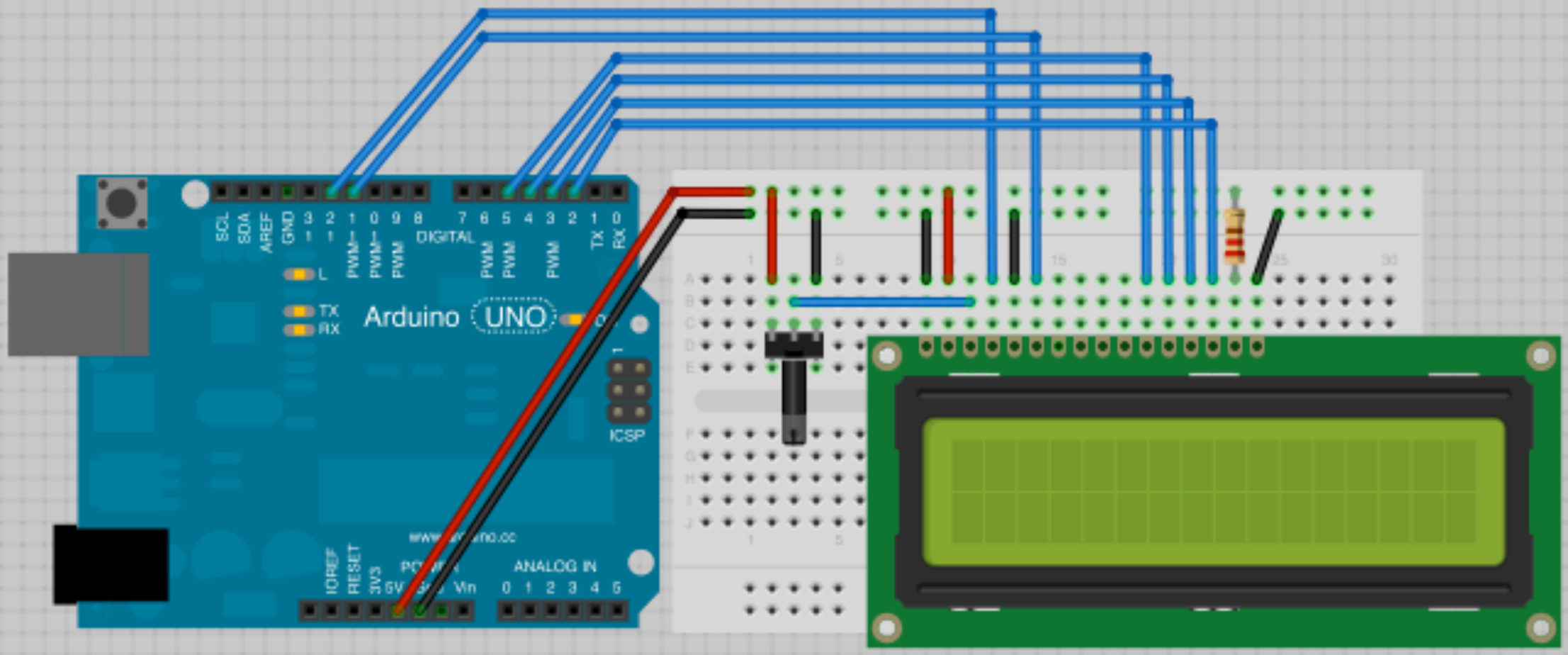
2 Z Axis Motors

L N G COM COM V+ V+

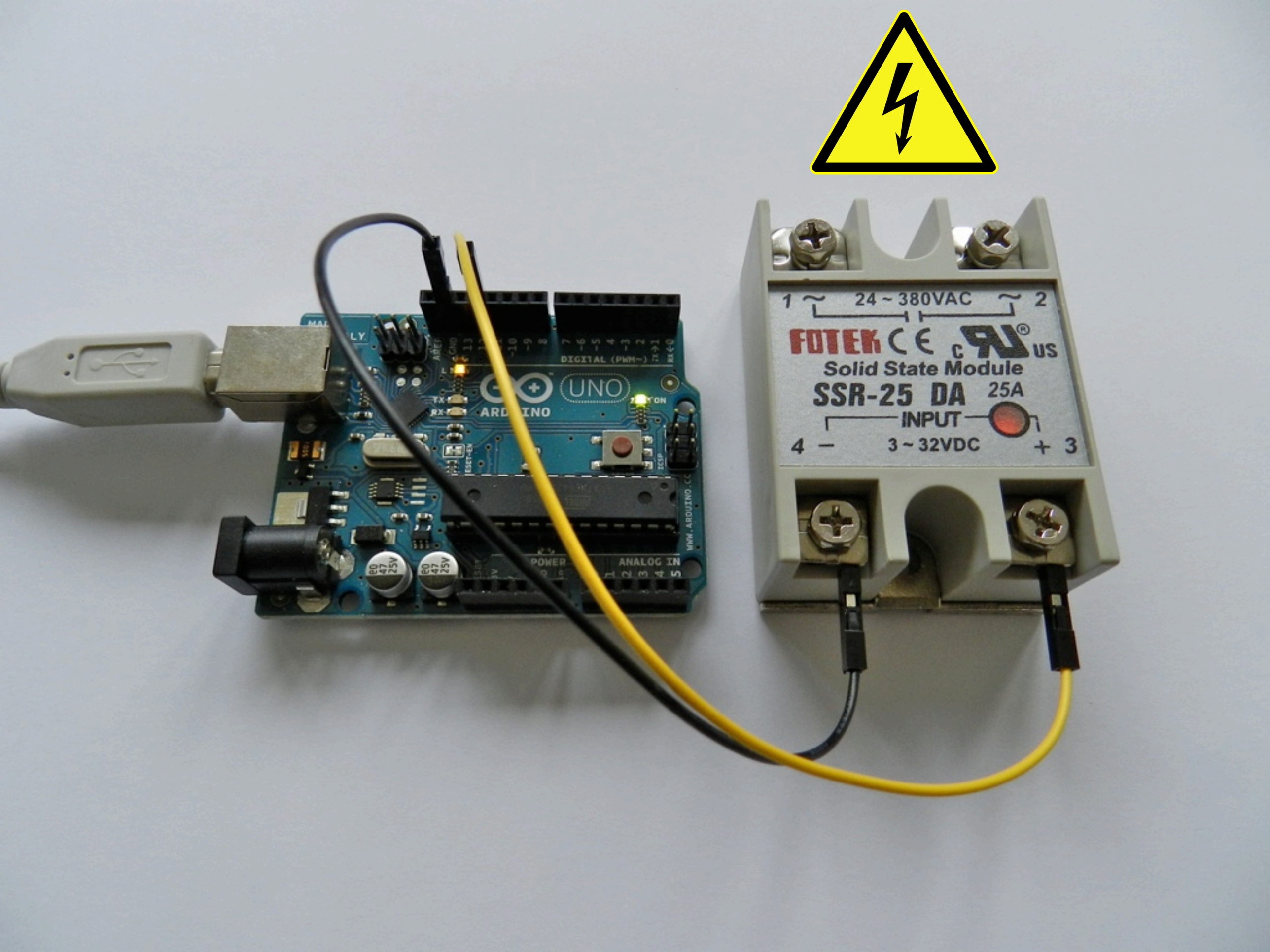
Power Supply

Created by Neil Underwood 5/28/2011

http://fritzing.org



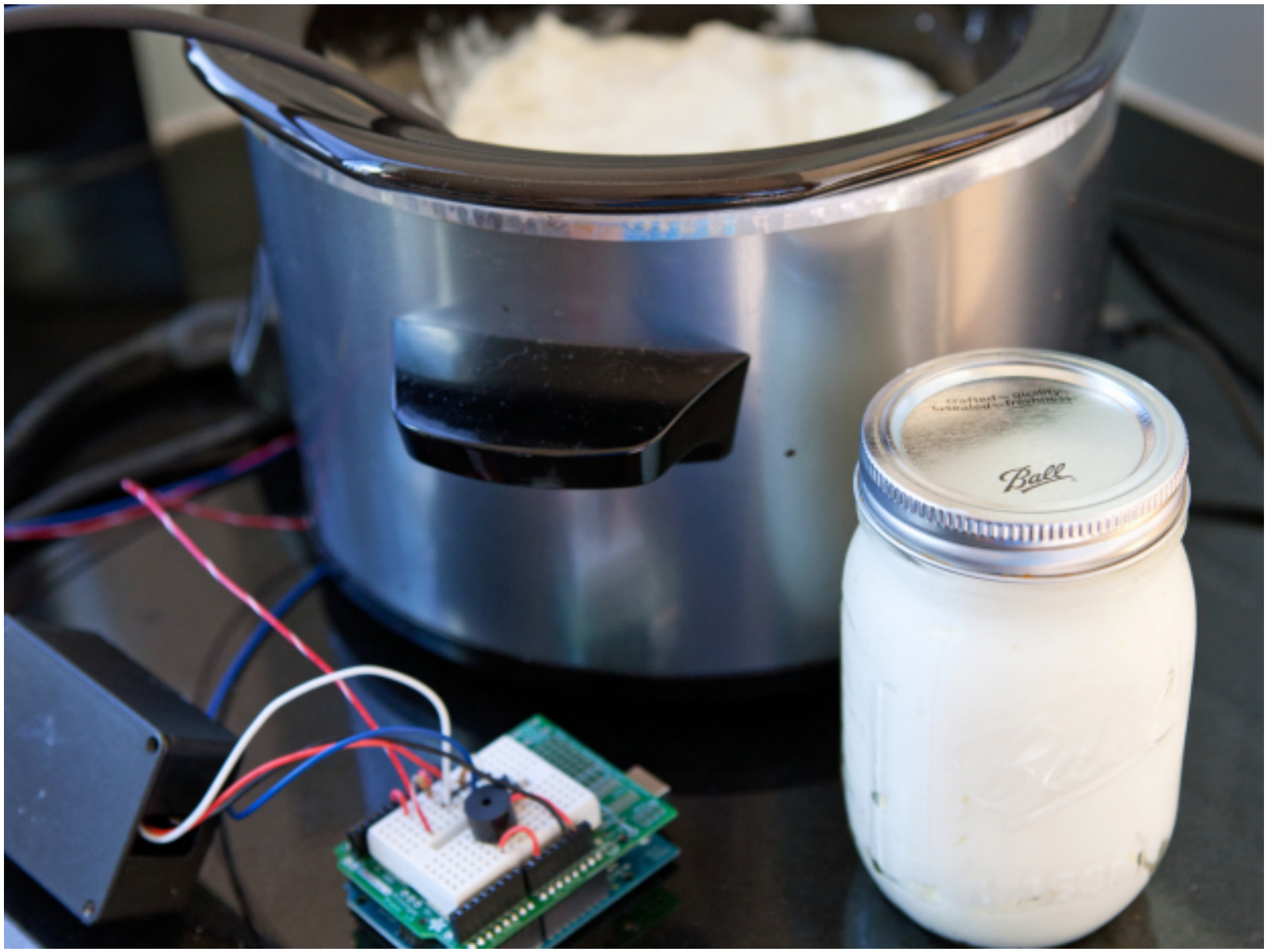
LCD : Liquid Crystal Display
Flüssigkristallanzeige


1 ~ 24 ~ 380VAC ~ 2
FOTEK
Solid State Module
SSR-25 DA 25A
INPUT
4 − 3 ~ 32VDC + 3
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
WWW.ARDUINO.CC

MAKE: PROJECTS

Yobot: Arduino Yogurt Maker

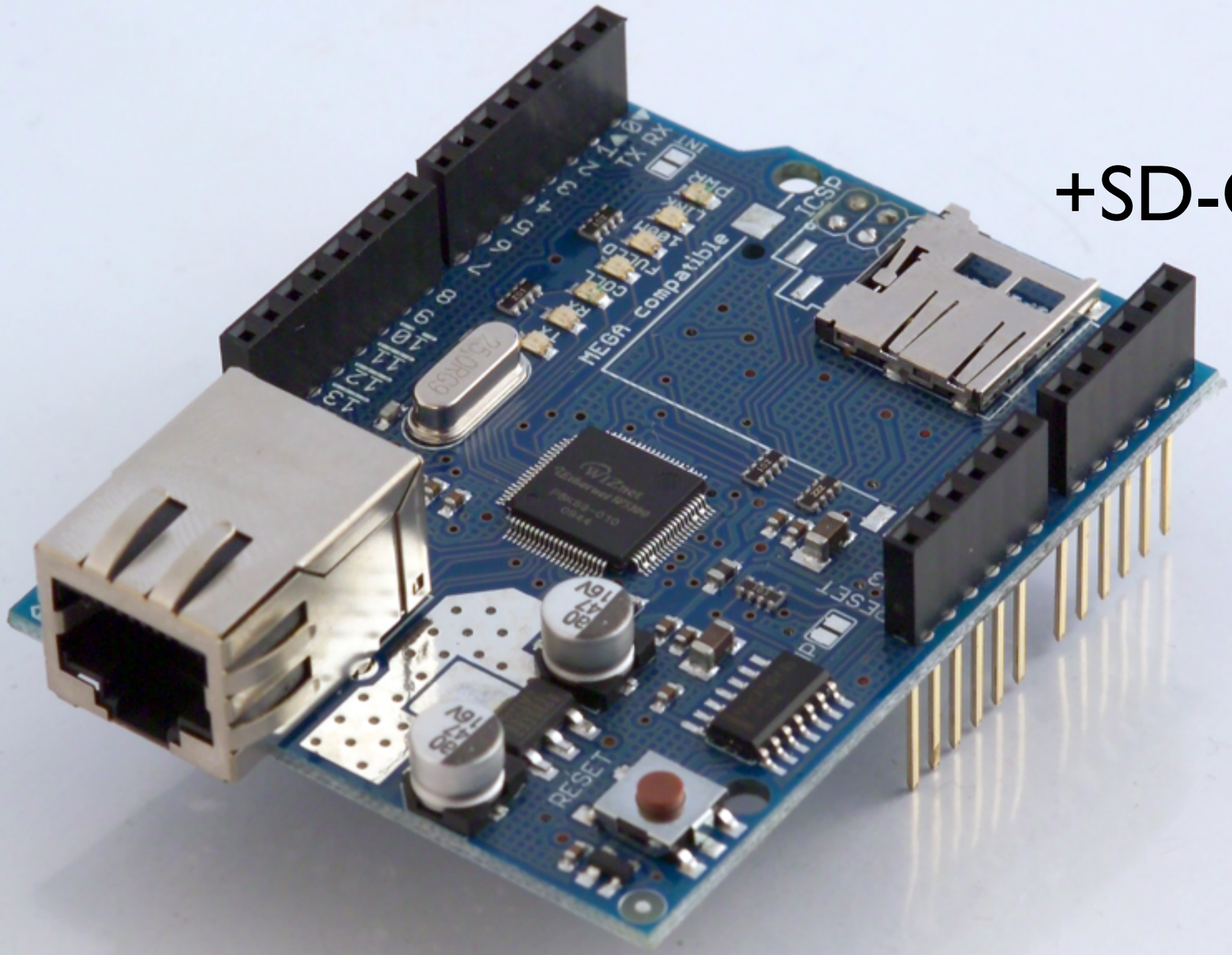
Build your own smart yogurt maker.



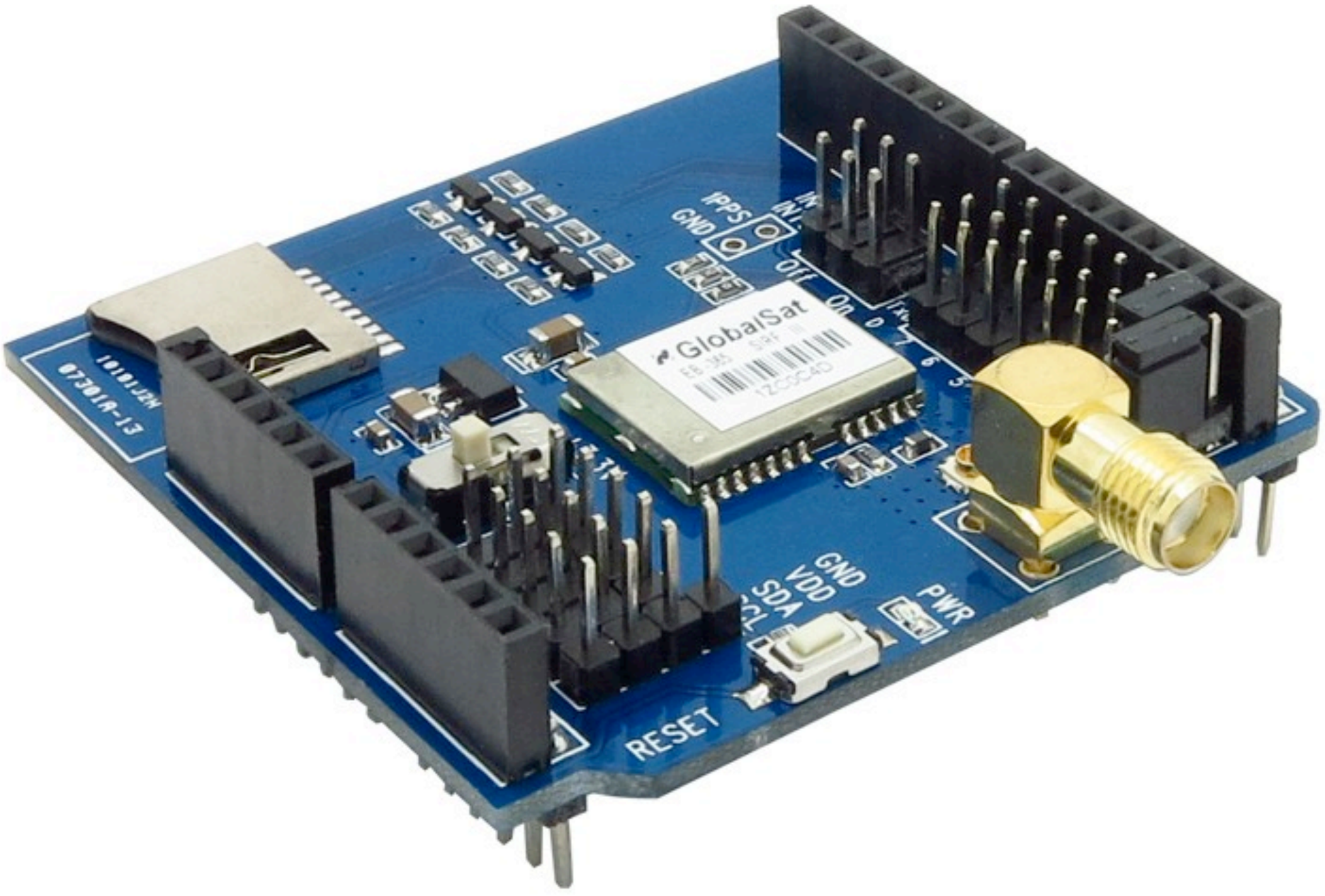
MAKE Magazine Volume 25

7. Die grosse weite Welt der Shield‘s und Breakout Board‘s

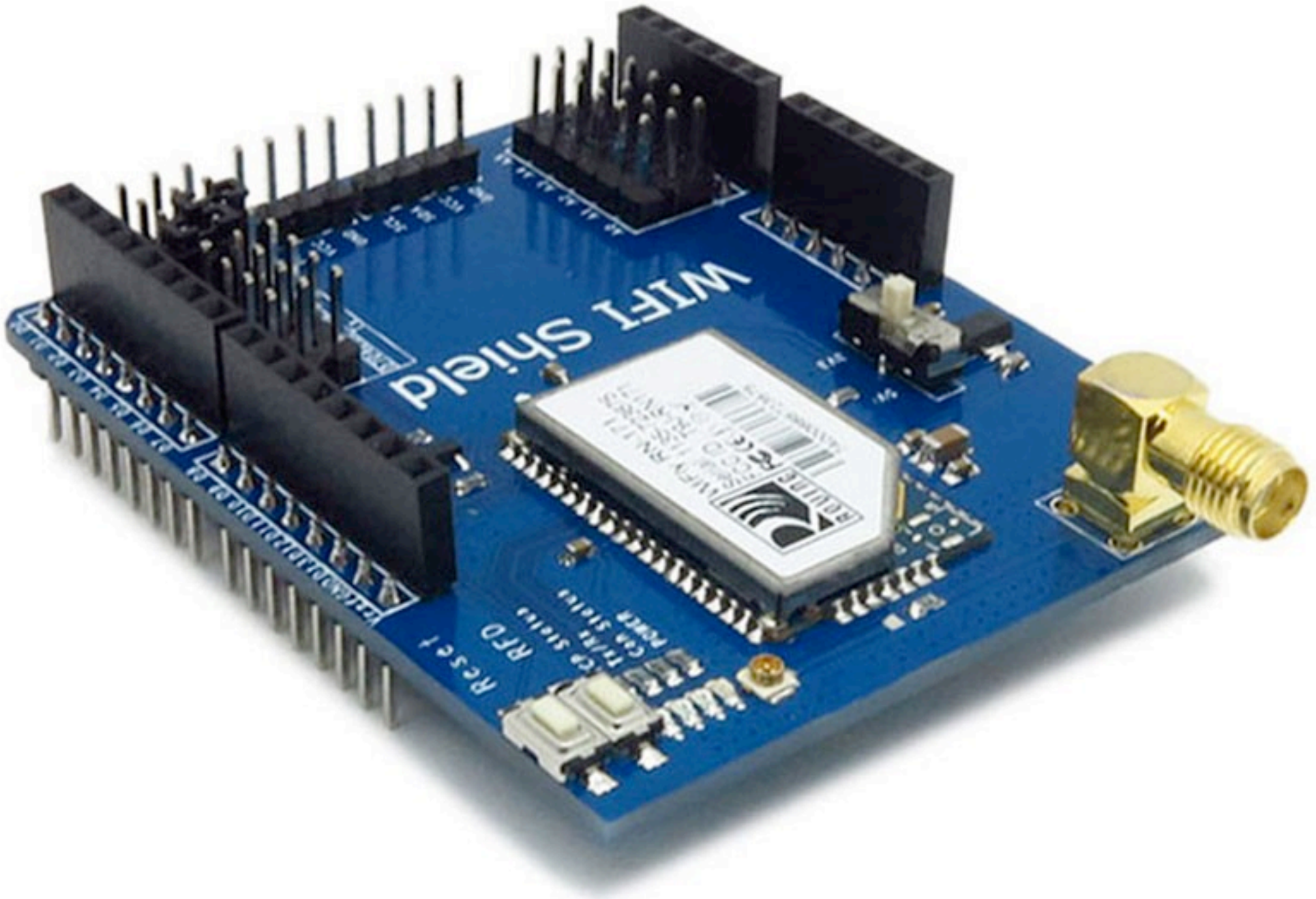
+SD-Card



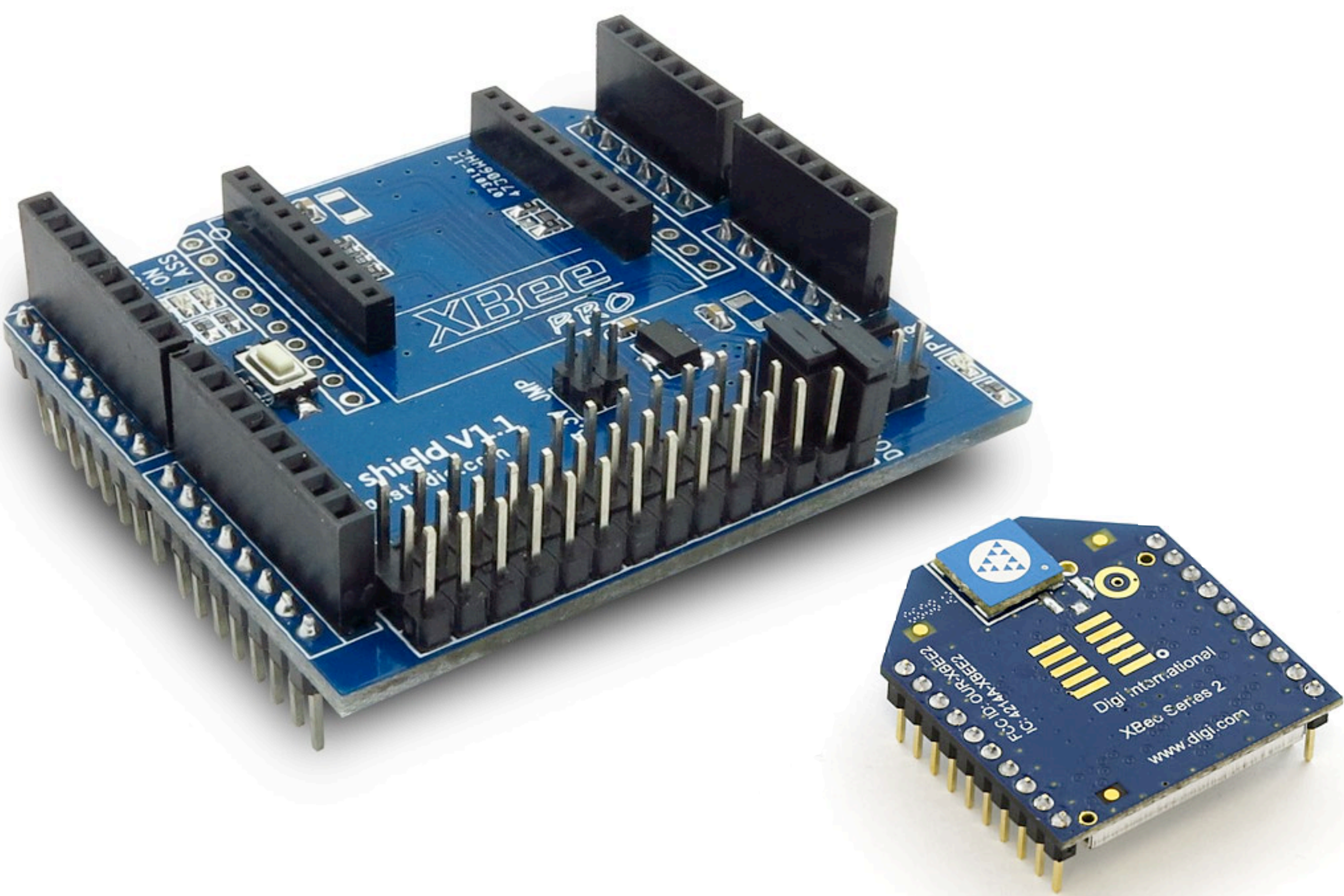
Ethernet Shield (Arduino)



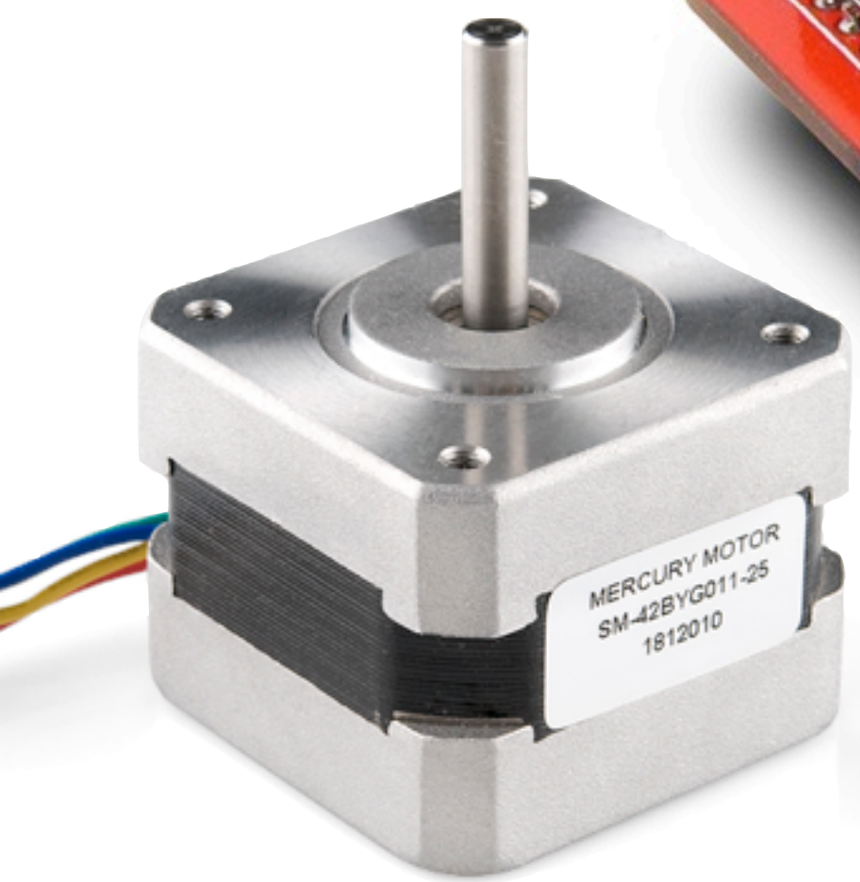
GPS Shield (ITEAD)

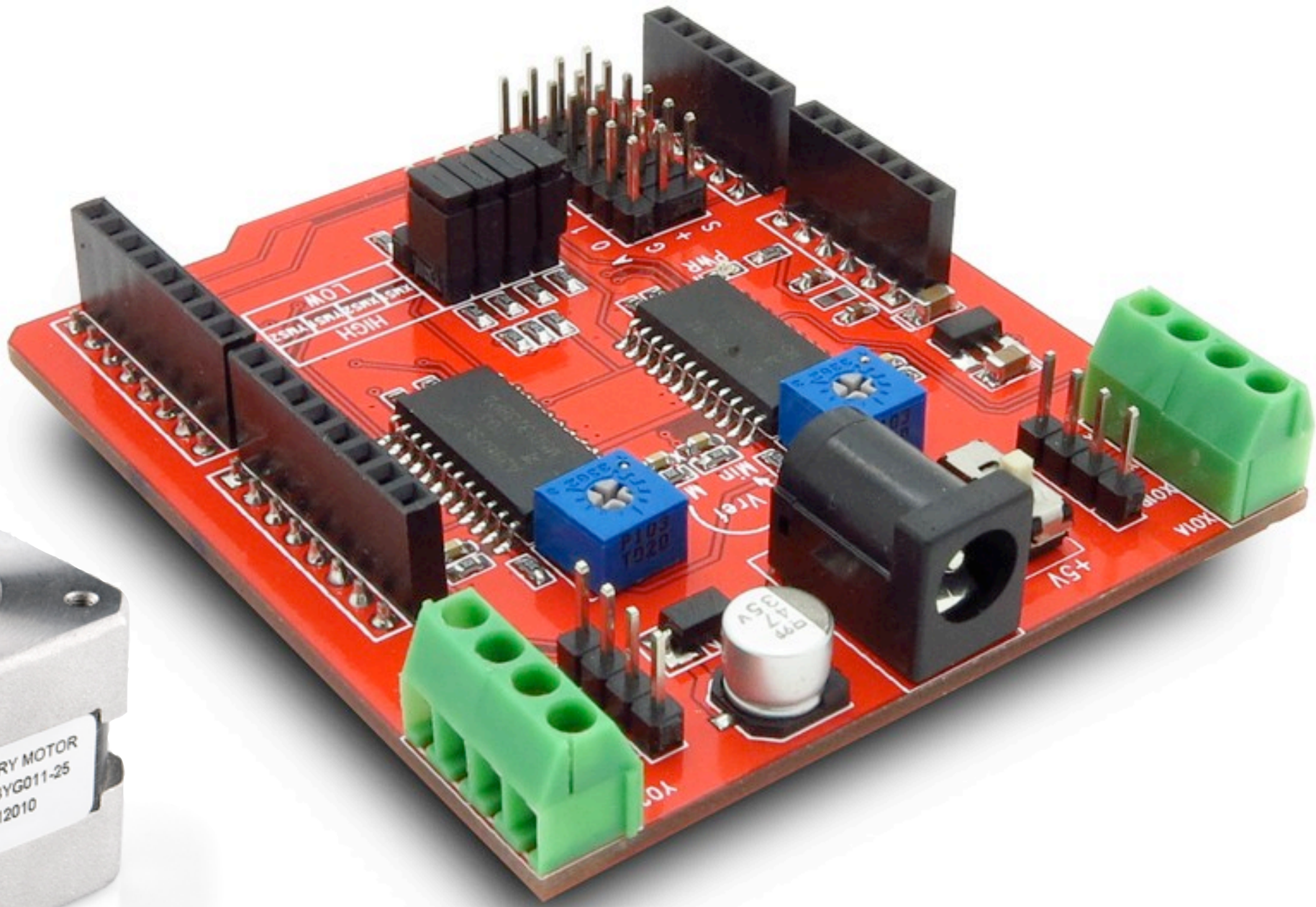


Wifi / WLAN Shield (ITEAD)

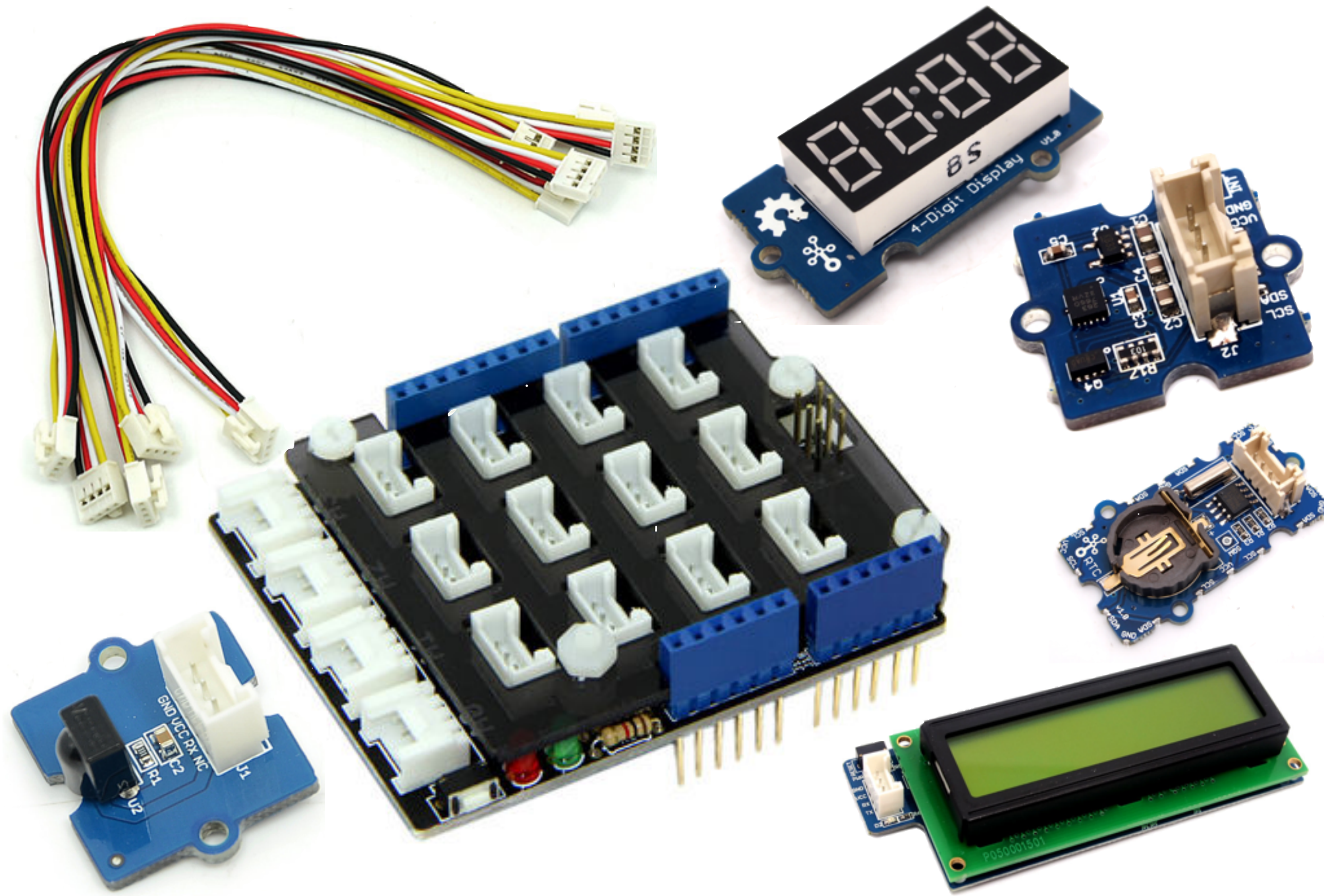


XBee Shield (ITEAD)

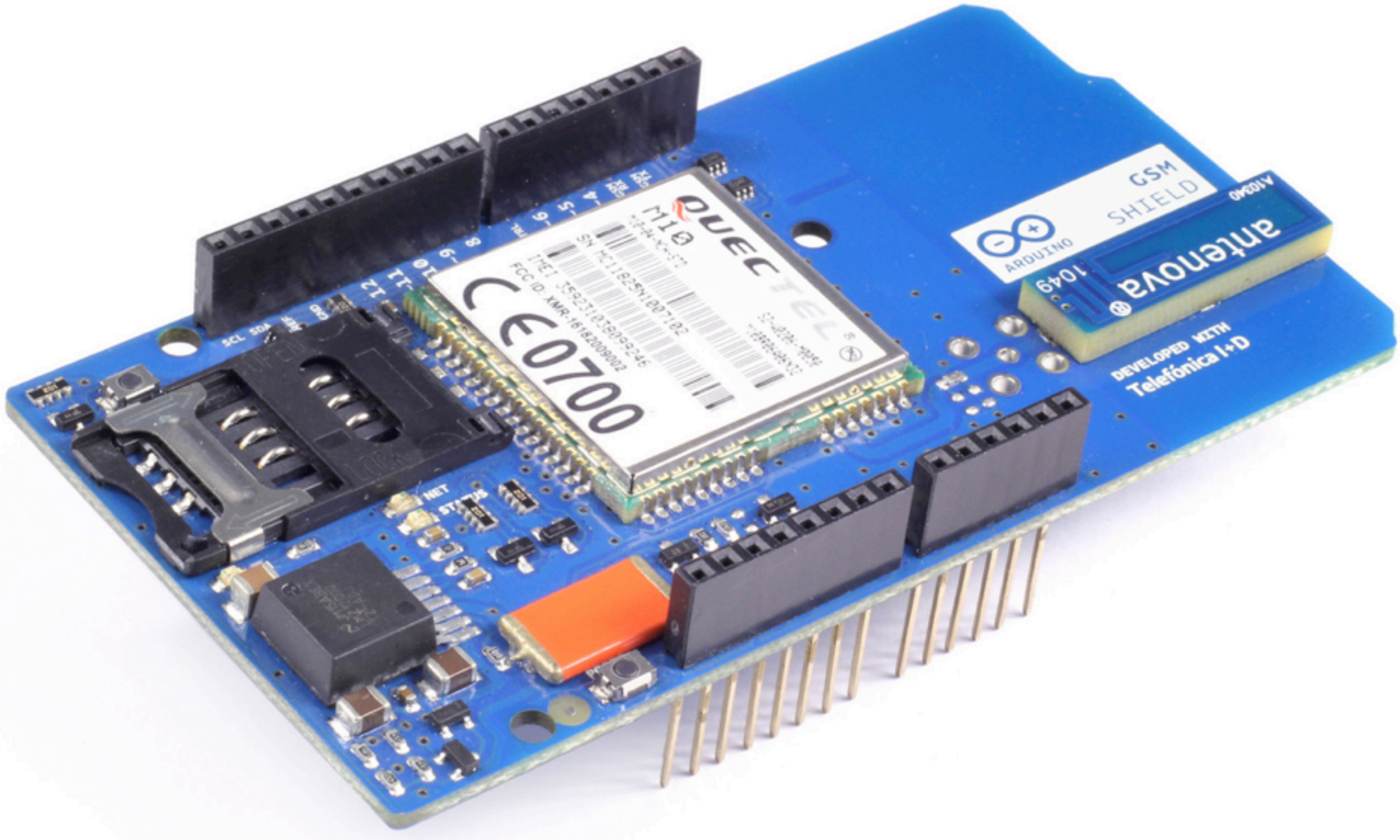




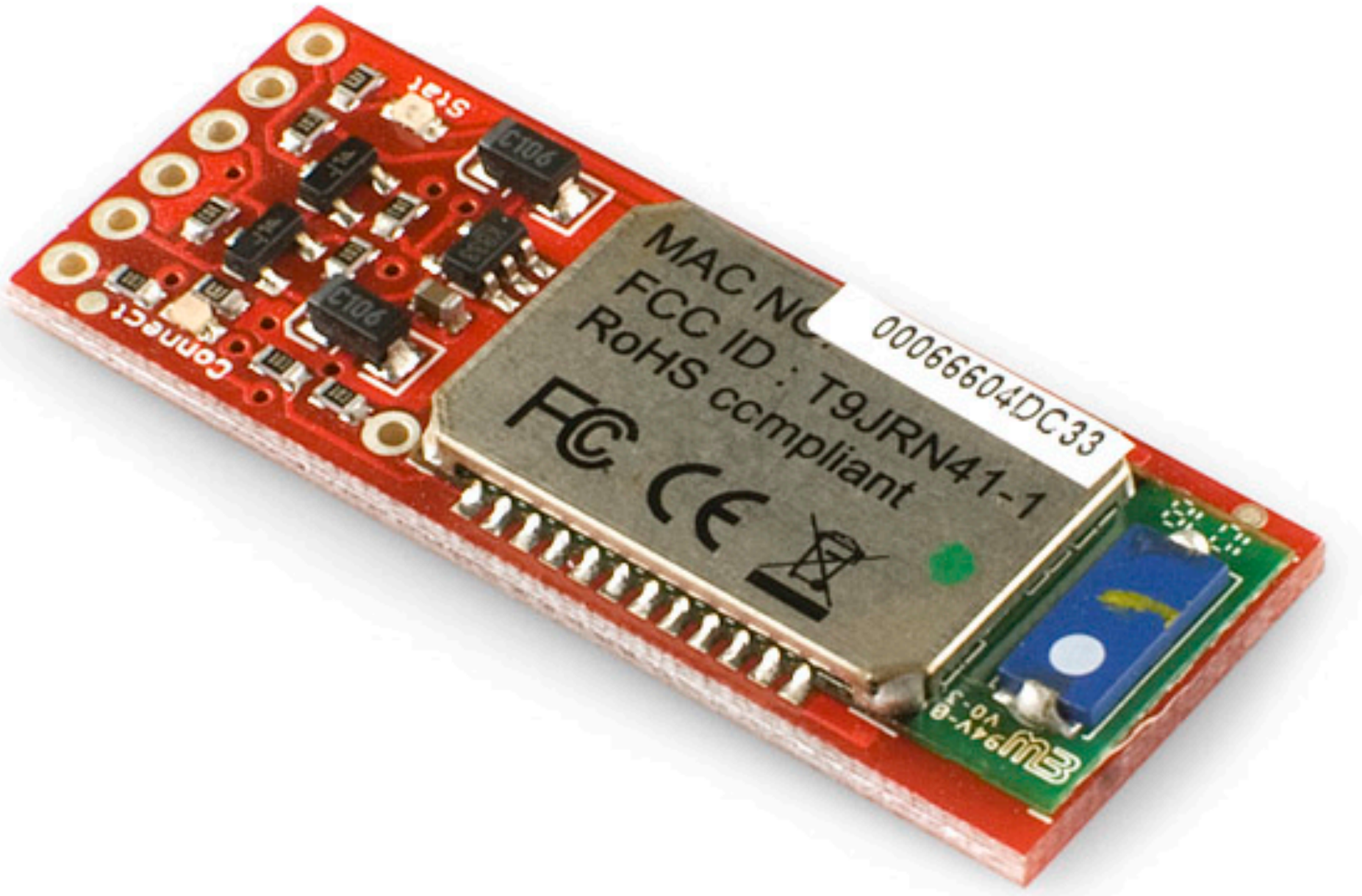
Dual Stepper Motor Driver Shield (ITEAD)



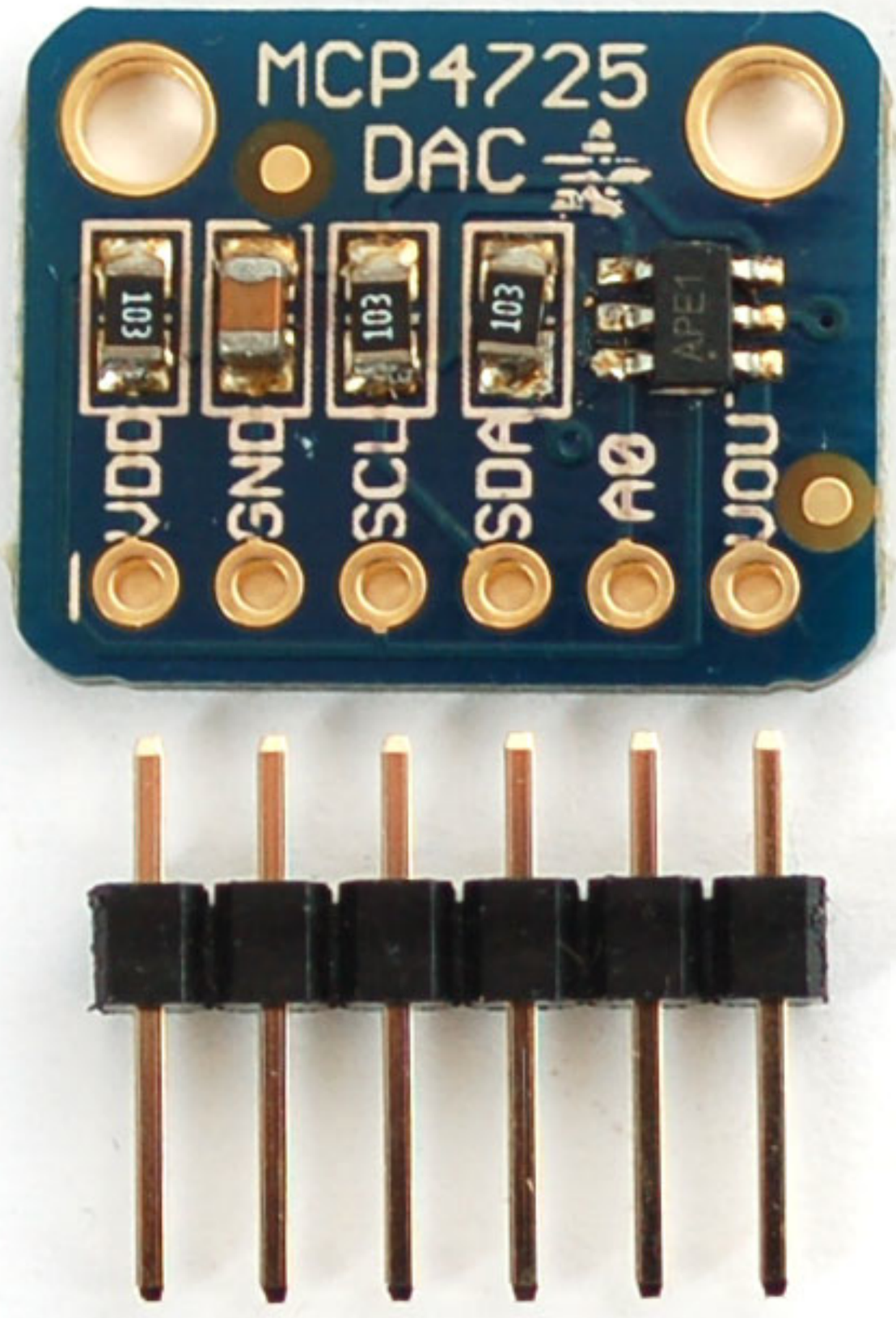
Grove-Base Shield (Seeed Studio)



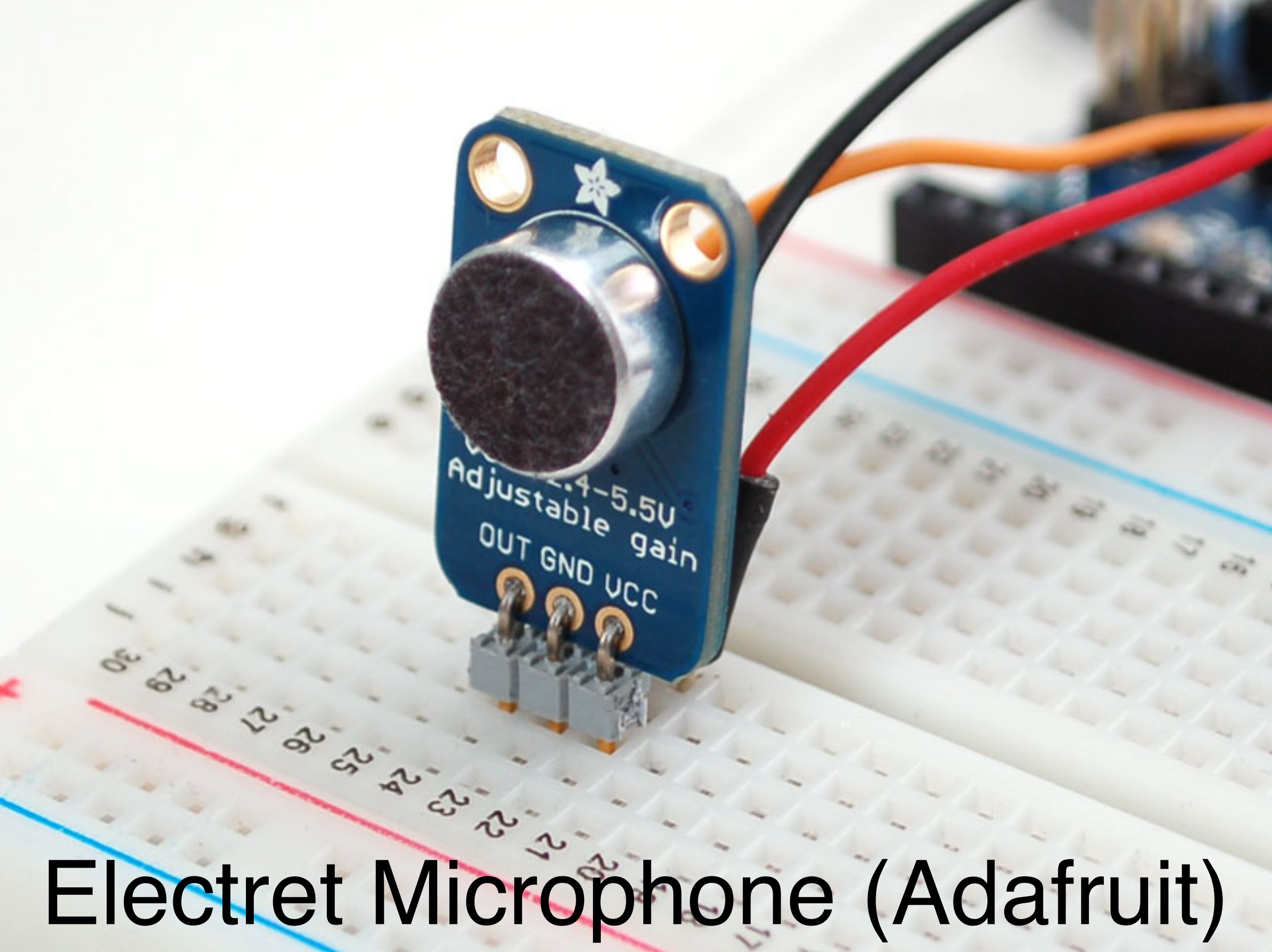
GSM Shield (Arduino)



Bluetooth Modem (SparkFun)

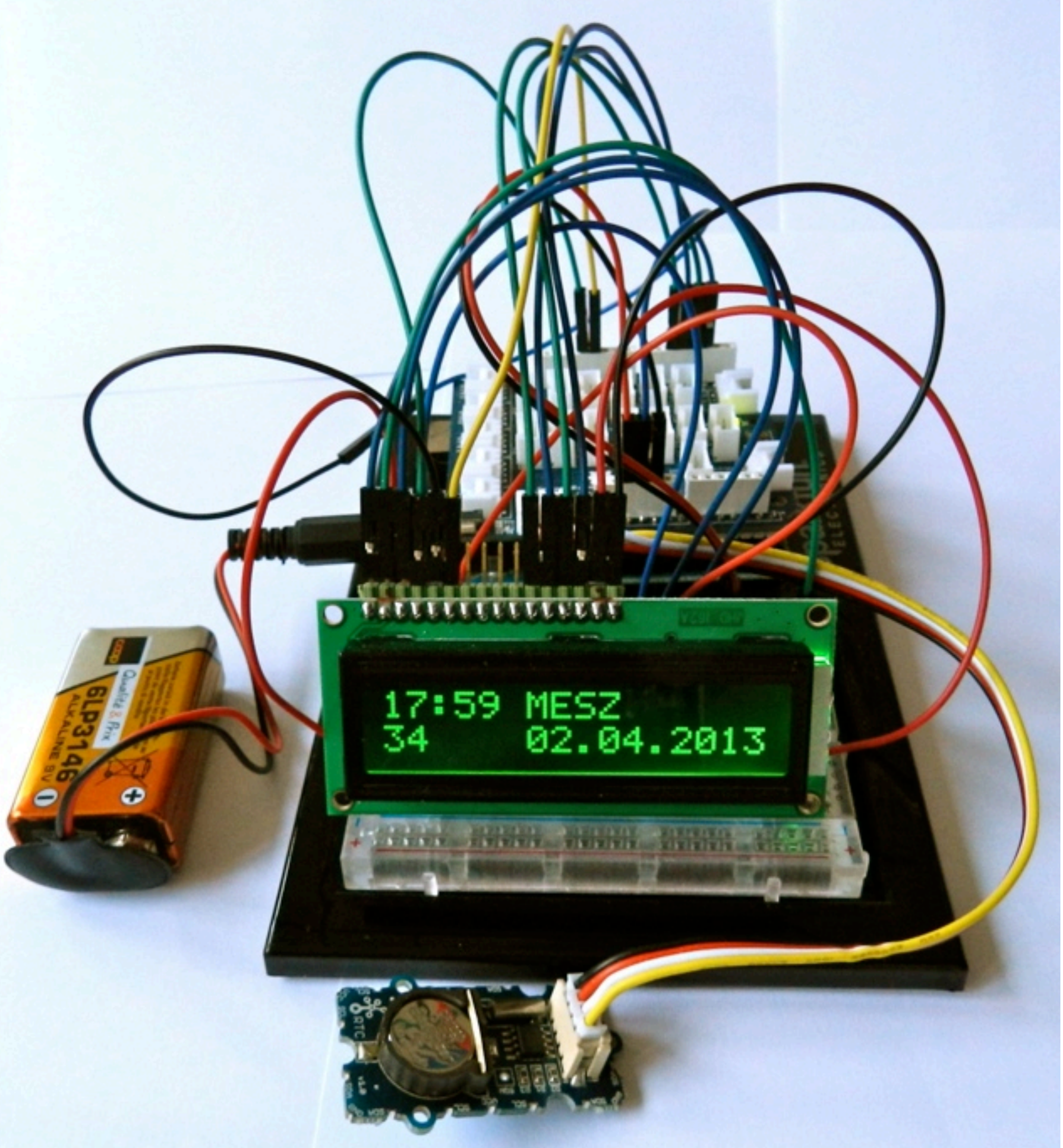


12-Bit DAC I2C (Adafruit)

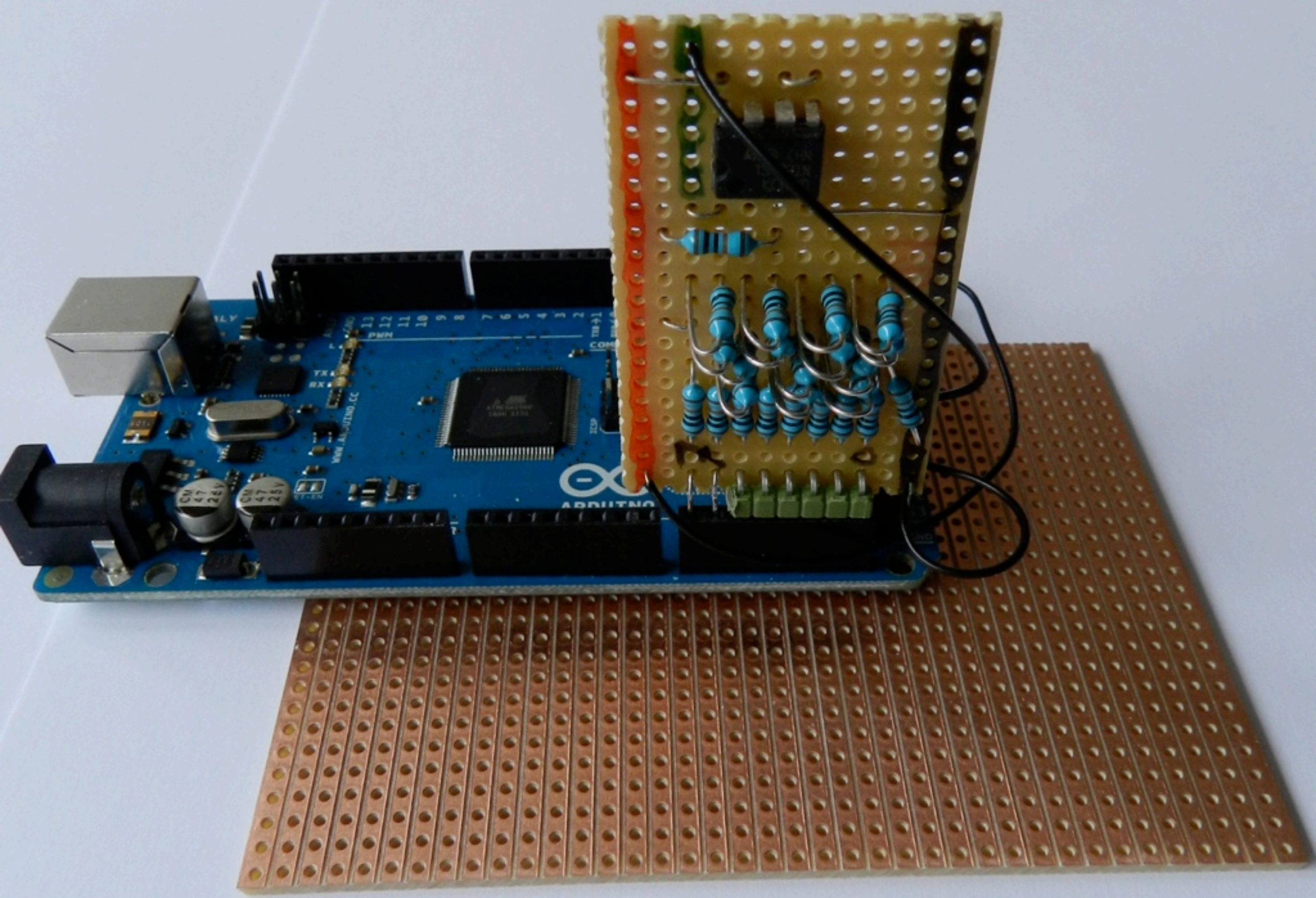


Electret Microphone (Adafruit)

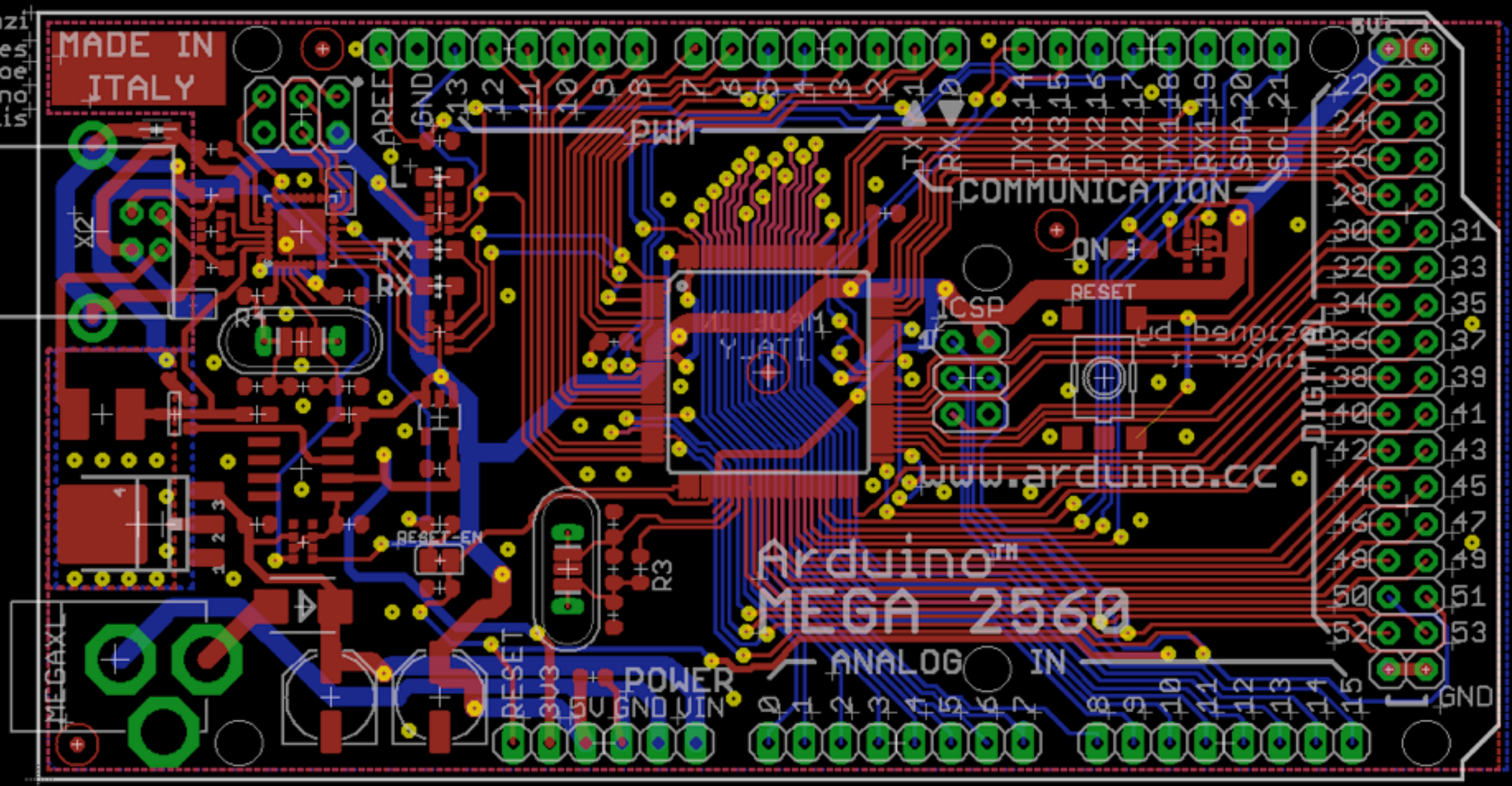
8. Elektrische Verbindungen und Aufbau von Schaltungen



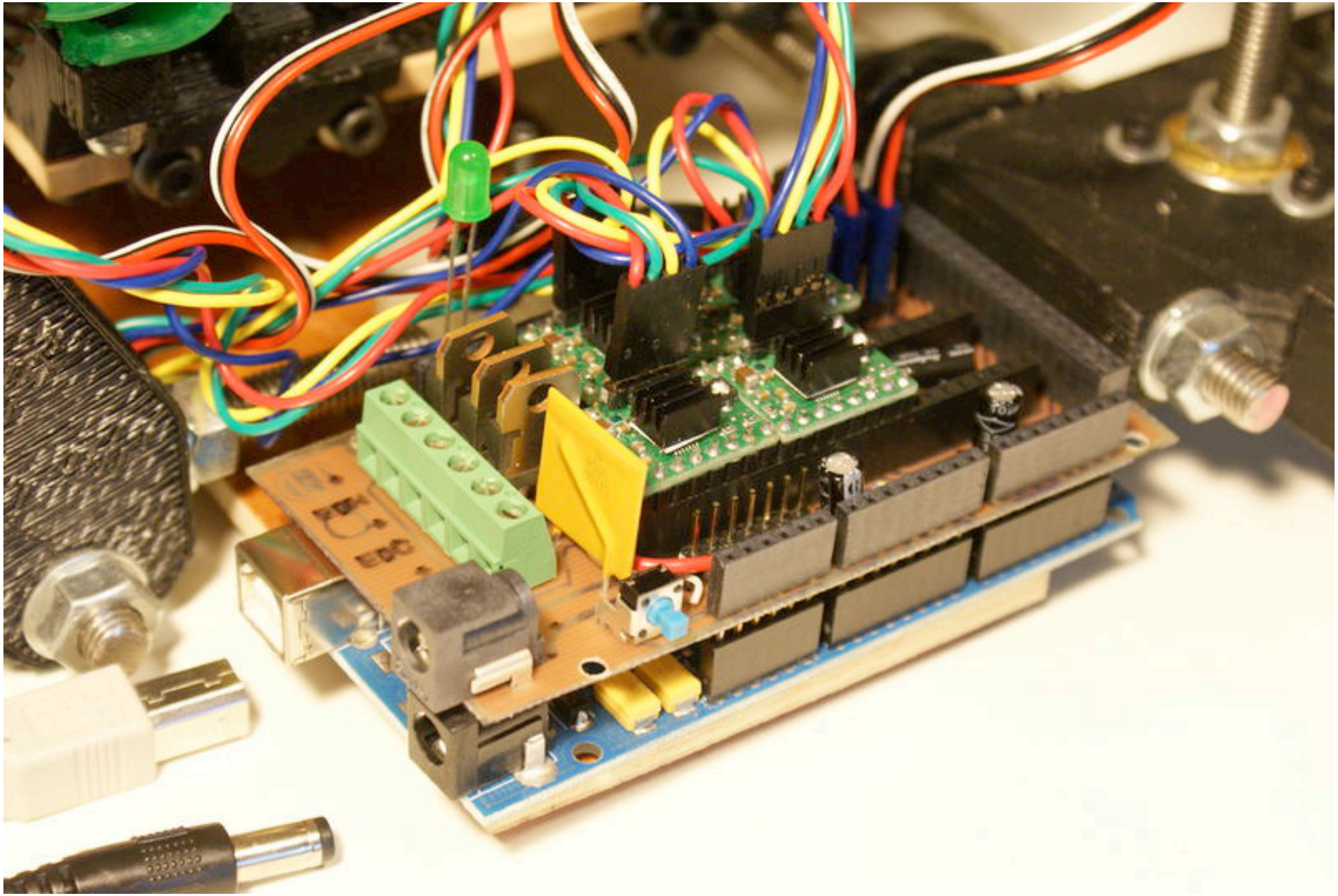
Breadboard
Steckplatine



Veroboard / Stripboard / Lochrasterplatine



Arduino Mega 2560 Reference Design



9. Fragen

und eigene Experimente

(Ende)