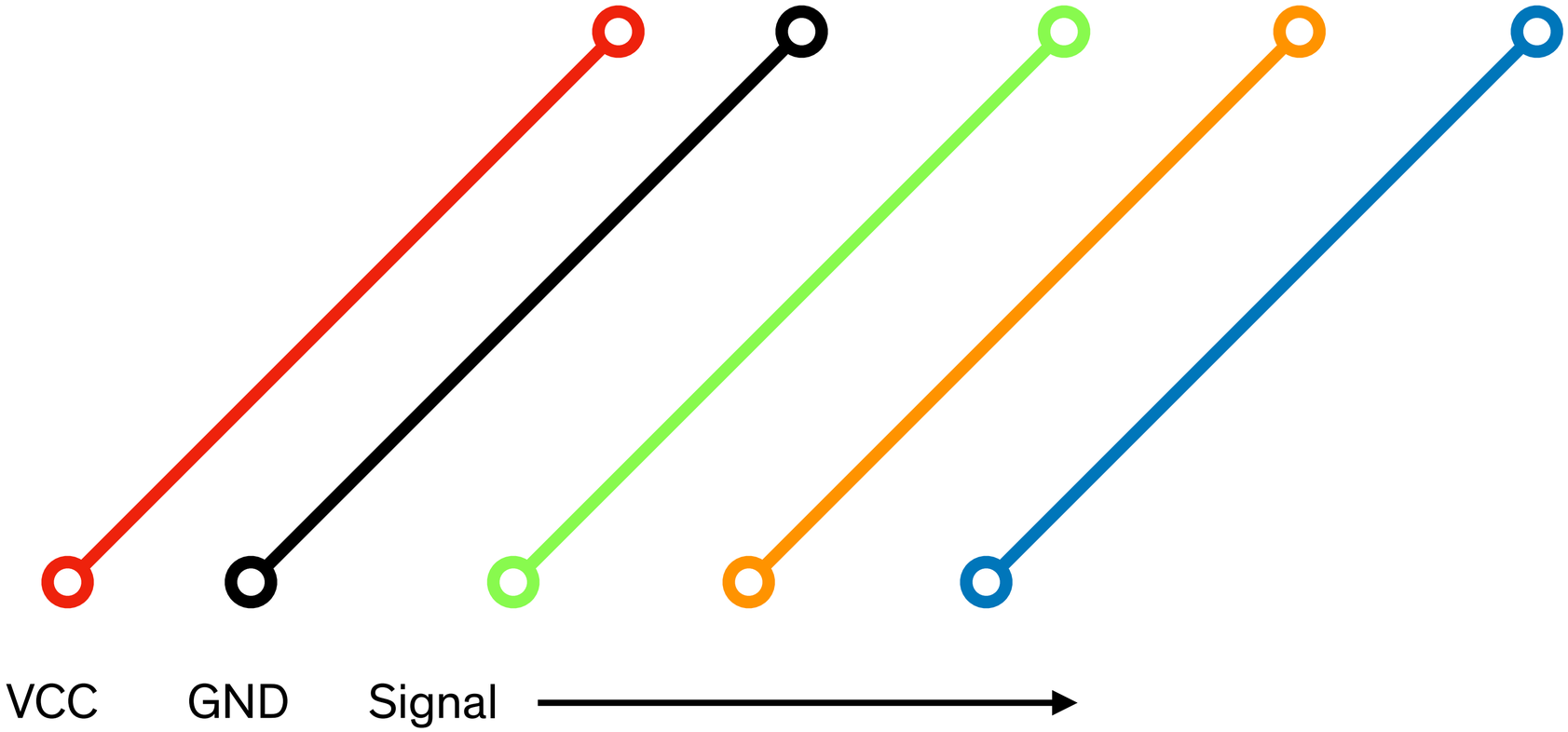
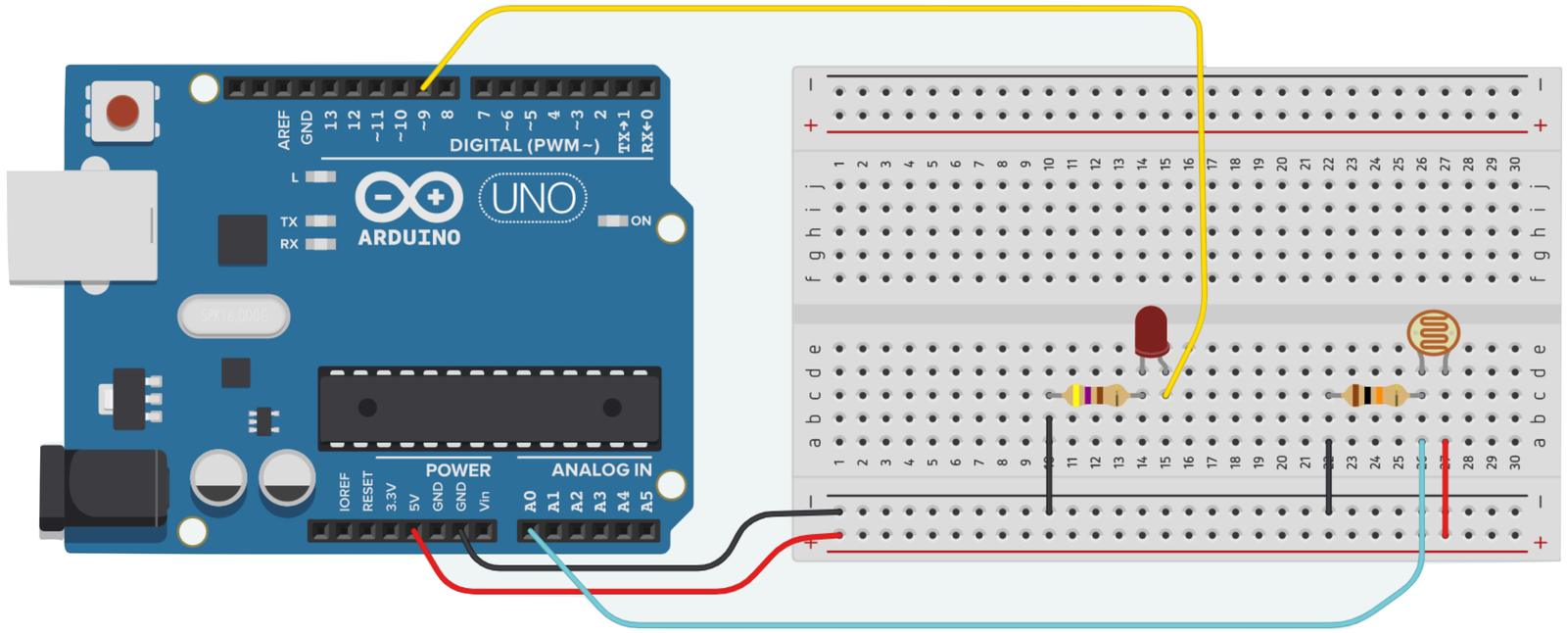
Physical Computing

WiSe 2019/20 – Grundlagen

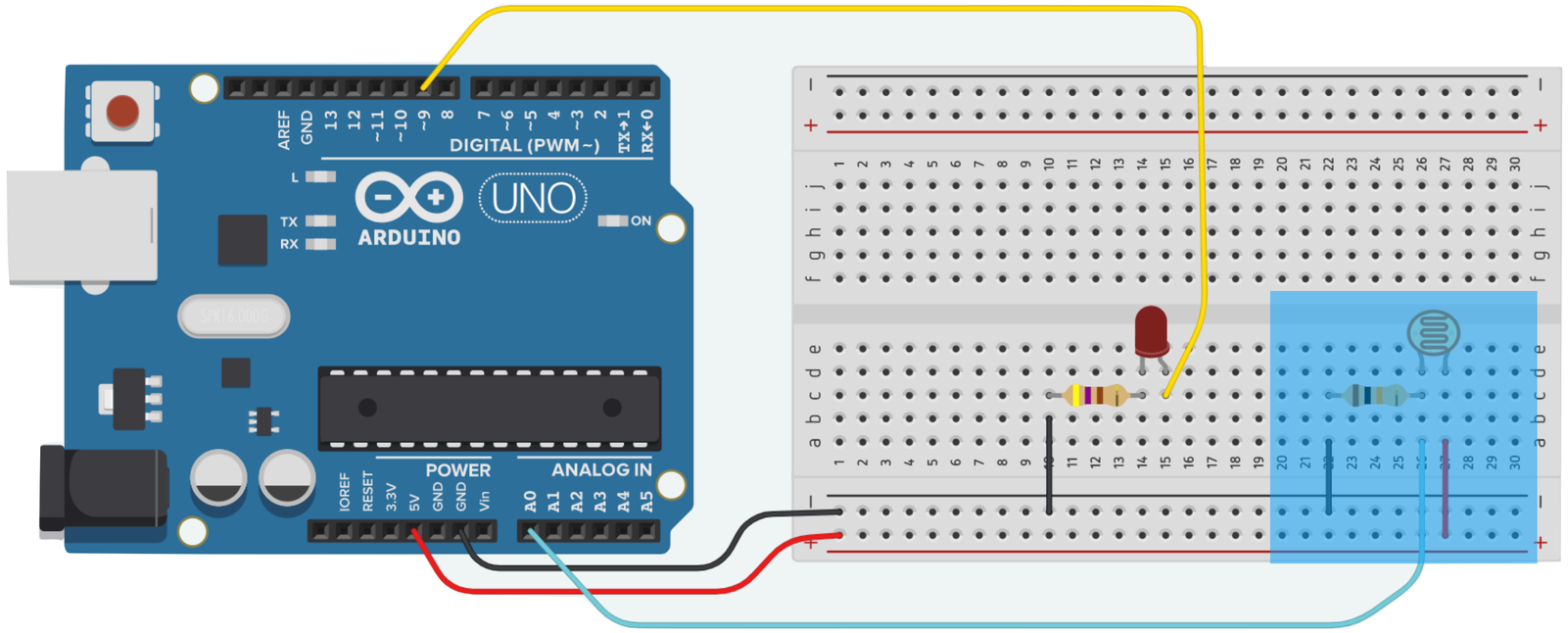
Farbkonventionen
Aufgabe
Multimeter
Motoren
Non-blocking
Endlicher Zustandsautomat
Bibliotheken
Logikgatter



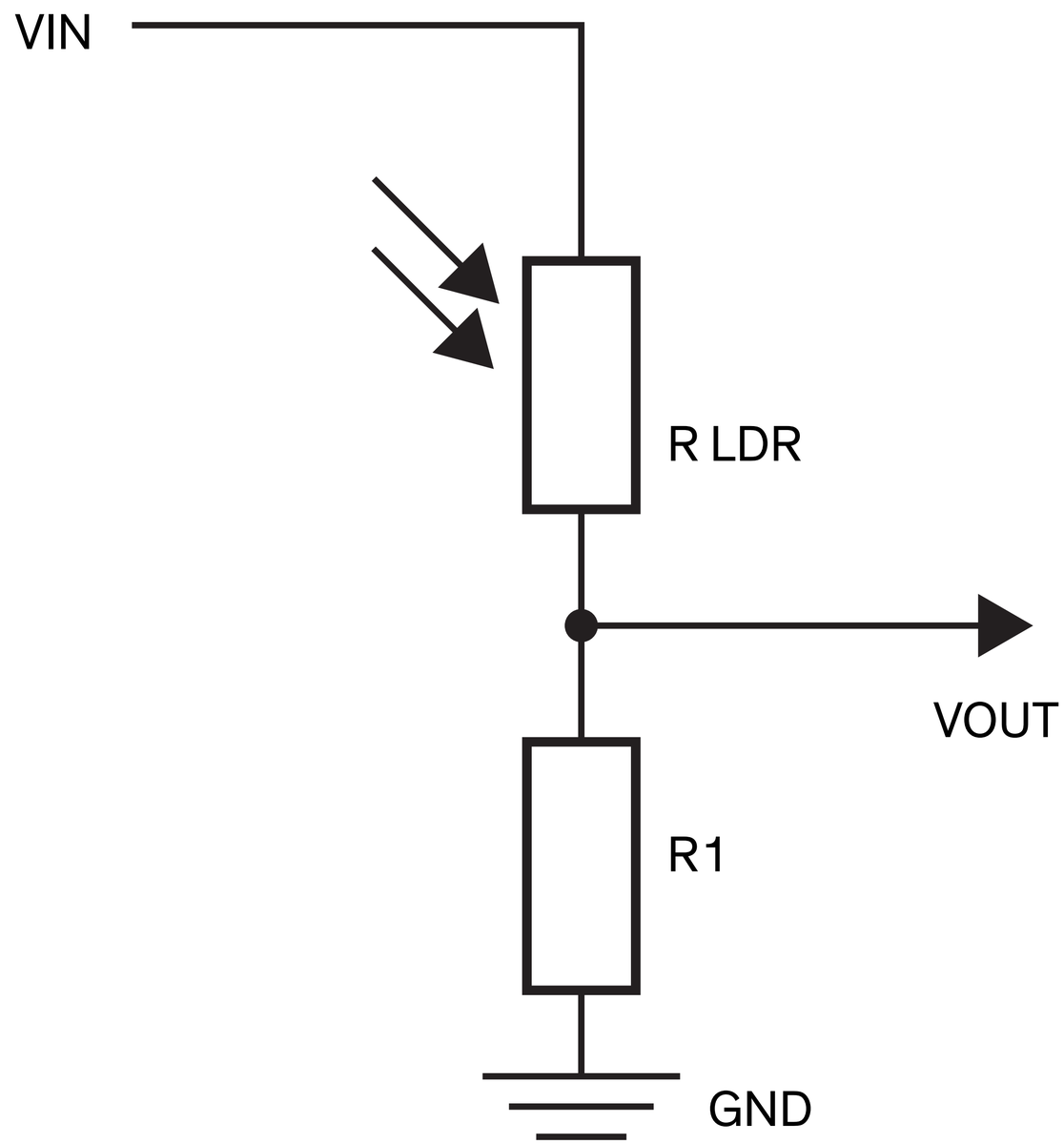
Farbkonventionen



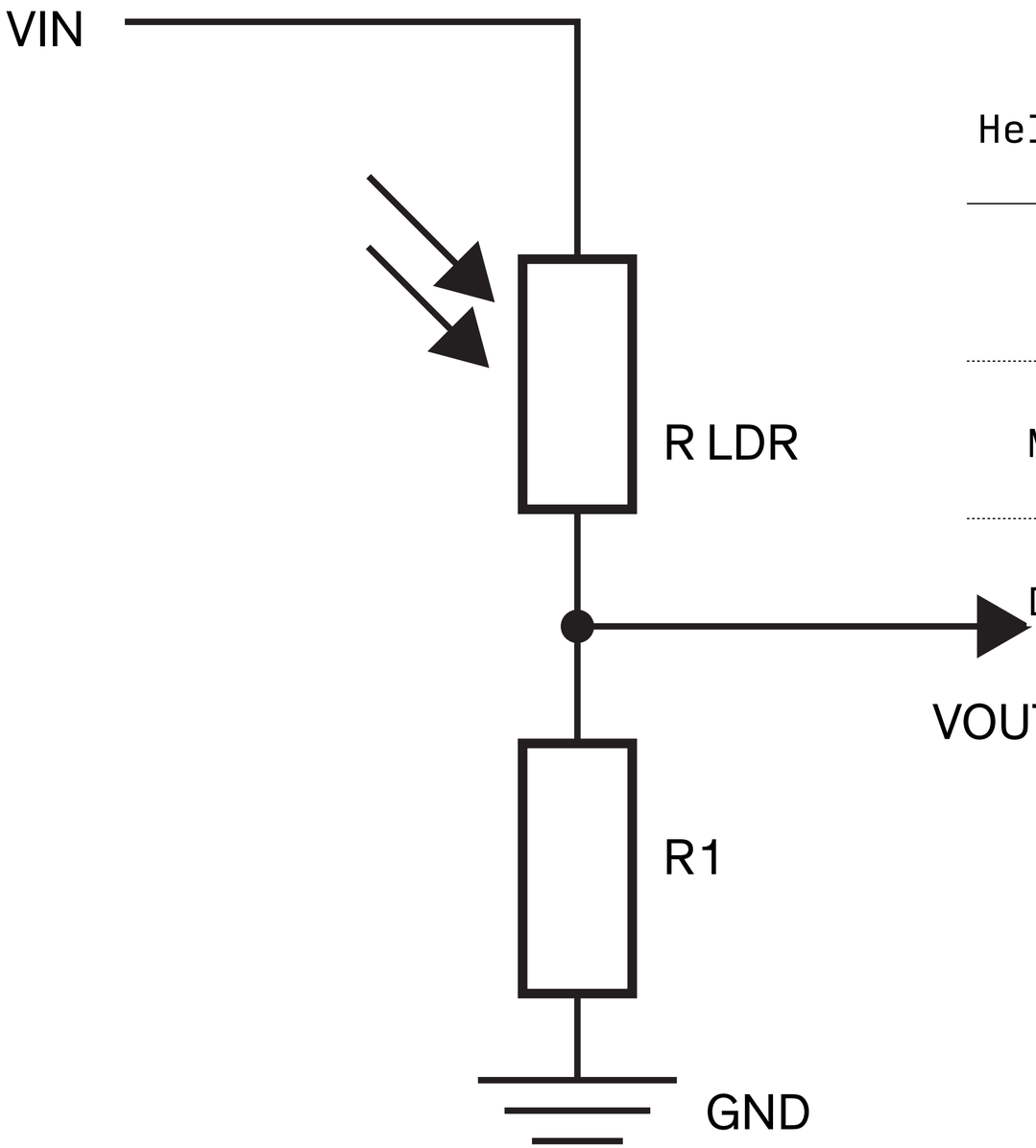
Aufgabe vom 08.11.19



Aufgabe vom 08.11.19

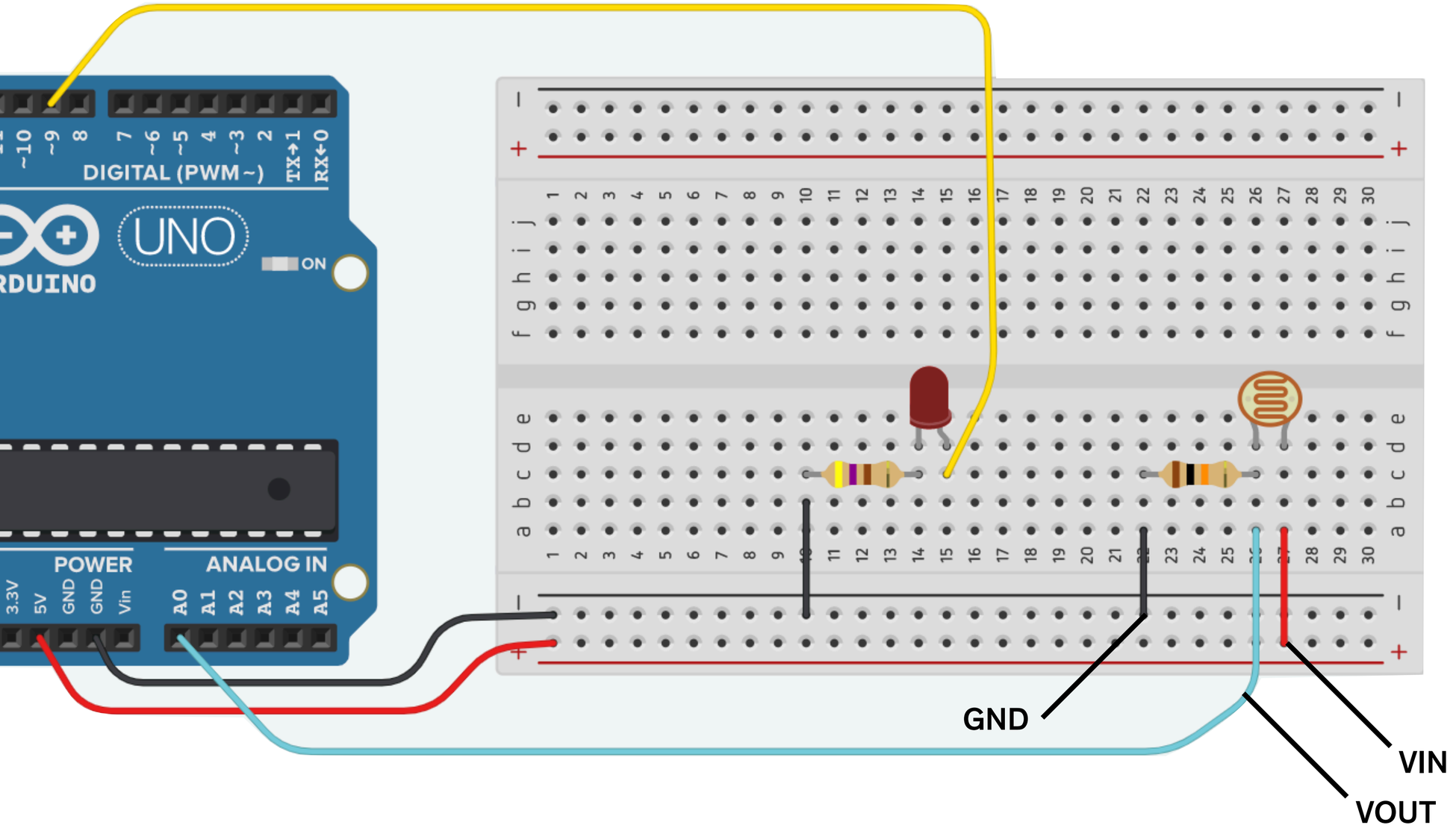


Spannungsteiler



Helligkeit	R1	R LDR	$\frac{R\ LDR}{(R1 + R\ LDR)}$	VOUT
Hell	10kΩ	1kΩ	0,09	4.55V
Mittel	10kΩ	5kΩ	0,33	3.52V
Dunkel	10kΩ	10kΩ	0,5	2.5V

Spannungsteiler

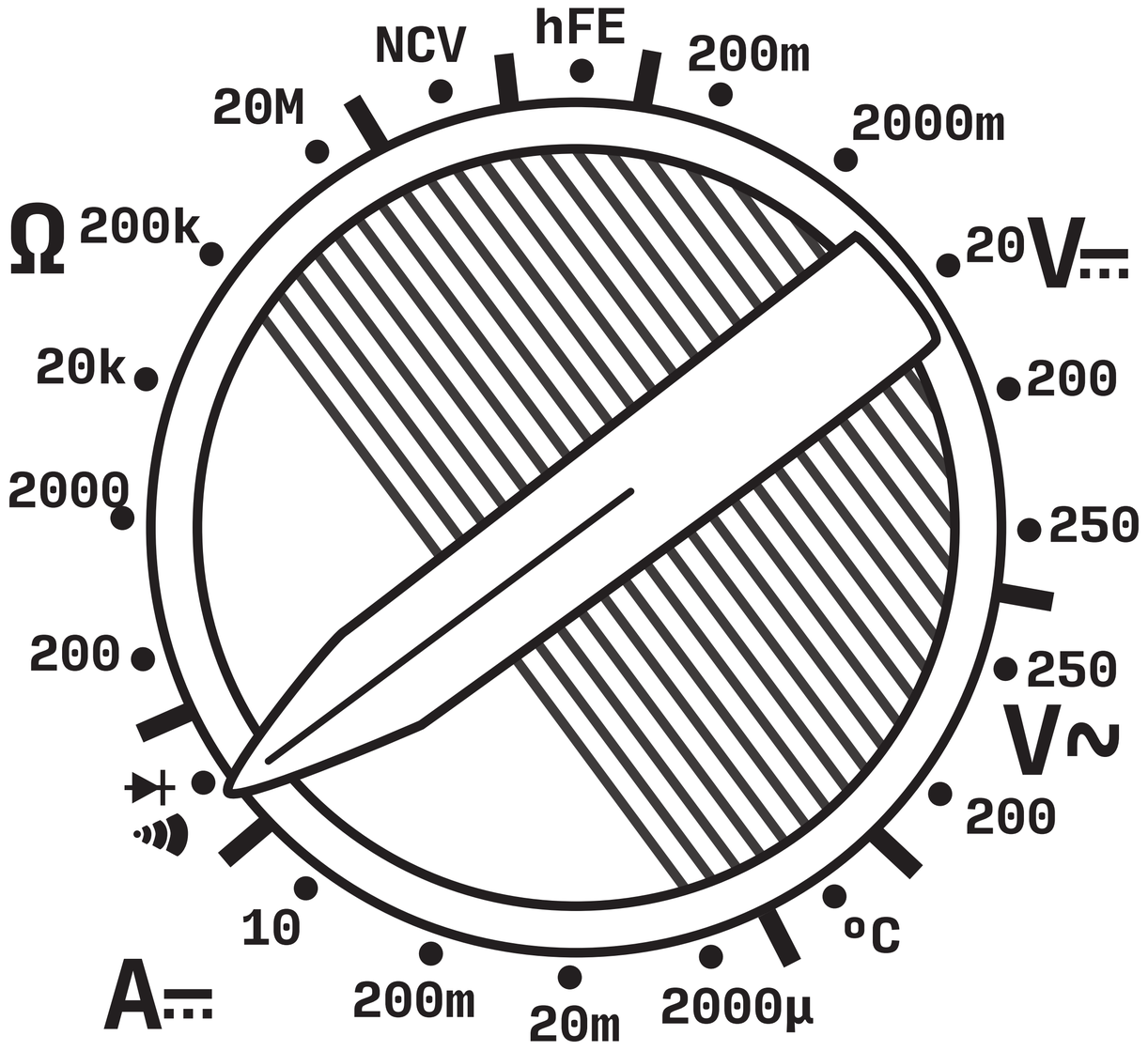


Spannungsteiler

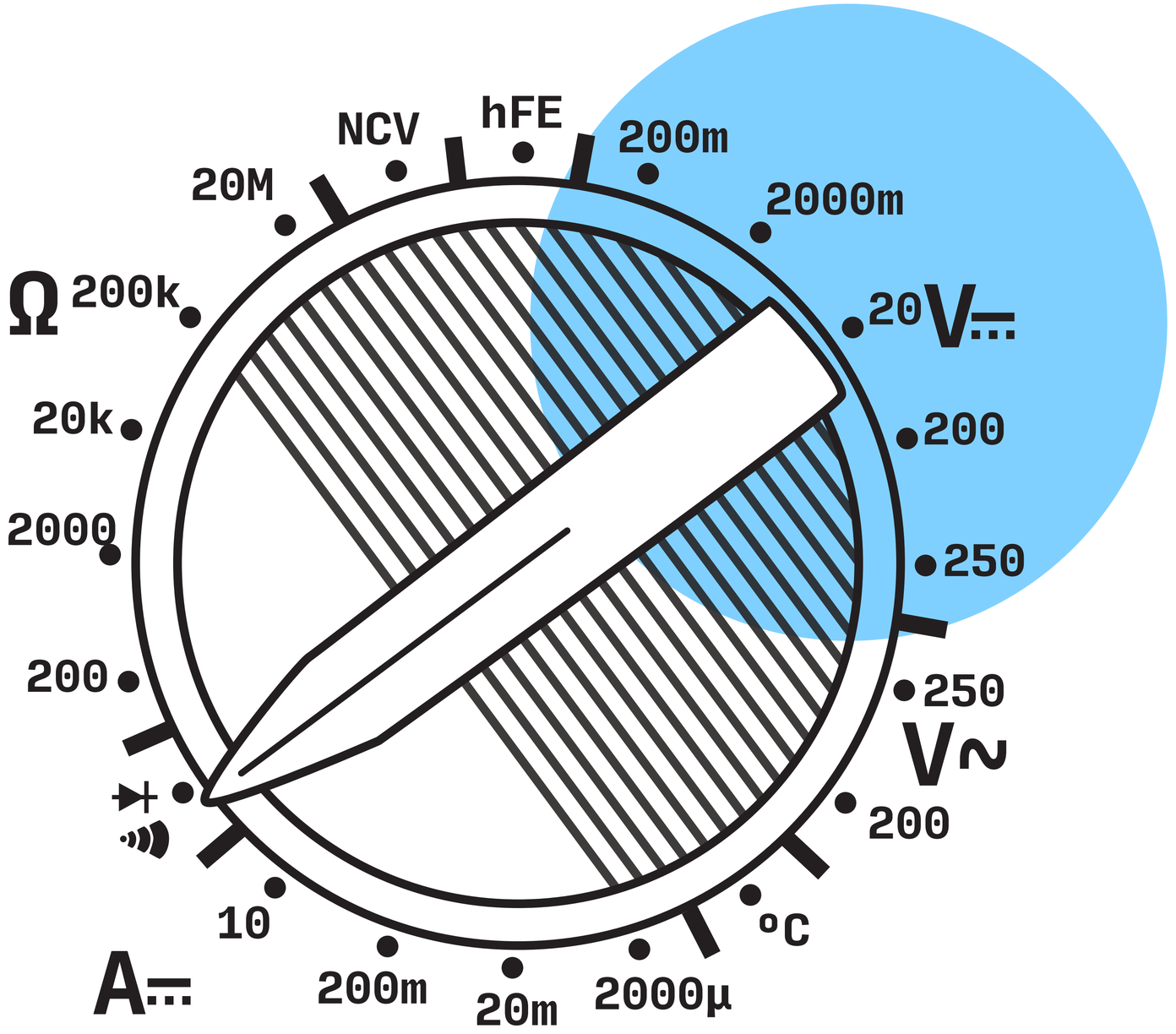
Multimeter

Multimeter

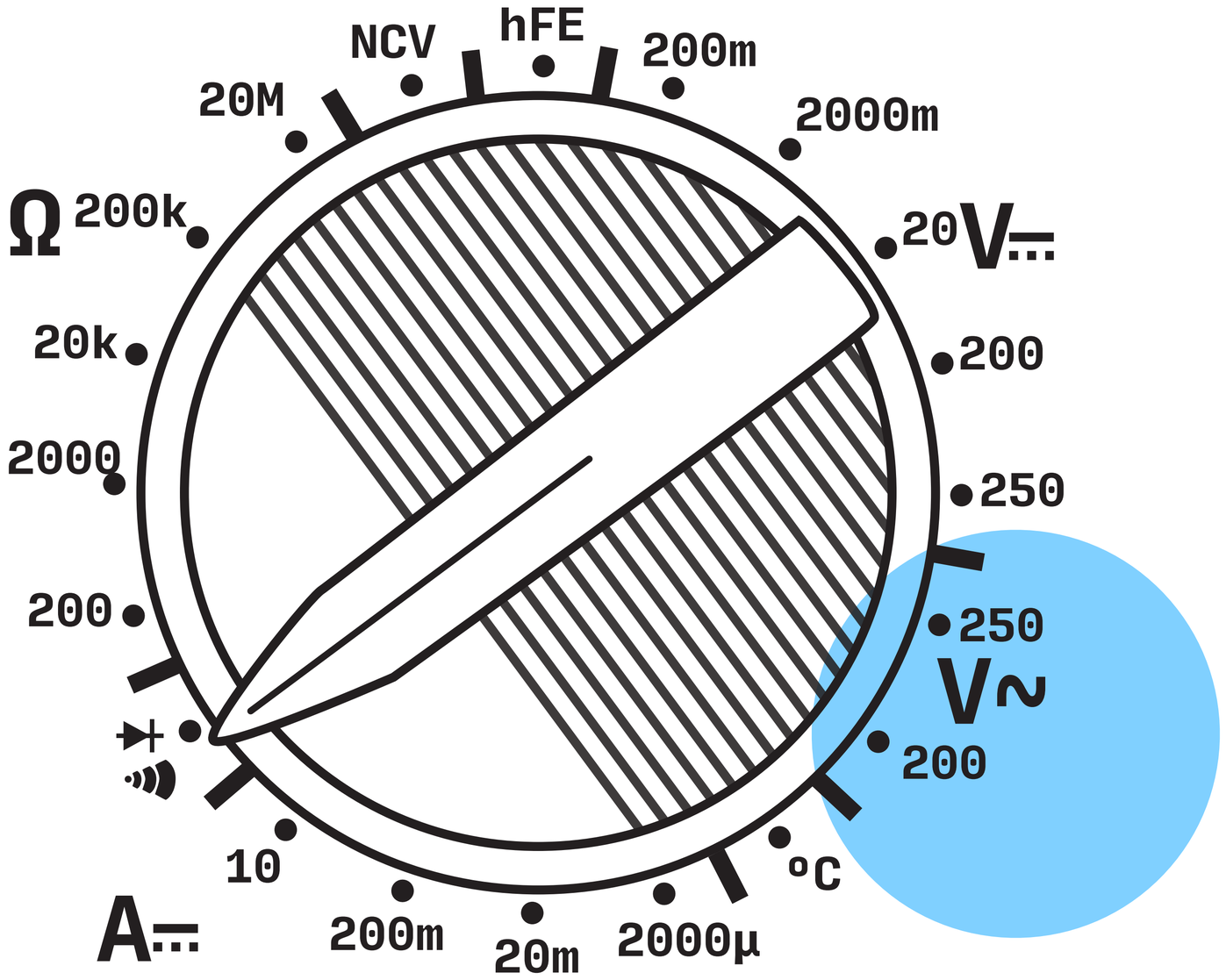

VOLTCRAFT
POWER
HOLD
20M
NCV
hFE
200m
2000m
Ω 200k
20k
2000
200
20 V⎓
200
250
250
V~
200
°C
10
A⎓
200m
20m
2000µ
V
CAT III 250V
FUSED
10A
hFE
°C
mAΩ


NCV
hFE
200m
20M
2000m
Ω
200k
20
V⎓
20k
200
2000
250
200
250
V≂
200
10
°C
A⎓
200m
20m
2000µ

Multimeter

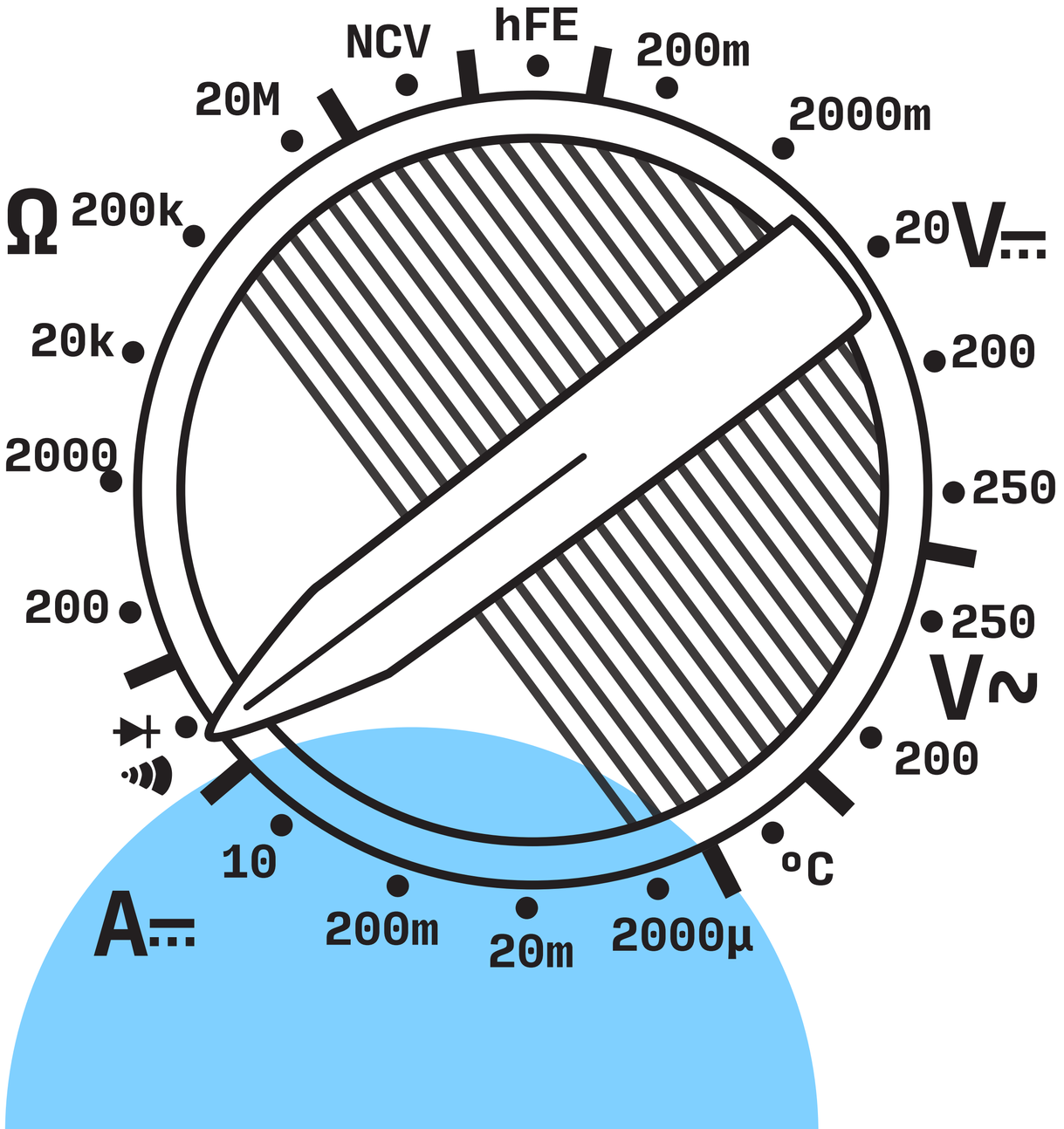

NCV
hFE
200m
20M
2000m
Ω
200k
20
V⎓
20k
200
2000
250
200
250
V≂
200
10
°C
A⎓
200m
20m
2000µ

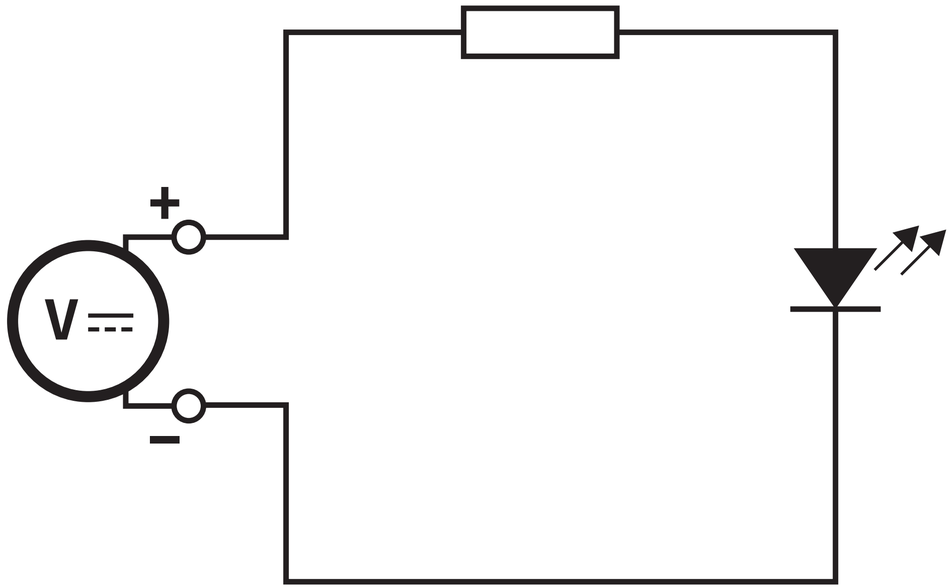
Multimeter


NCV
hFE
200m
2000m
20M
20
200k
200
20k
250
2000
250
200
200
10
°C
200m
20m
2000µ

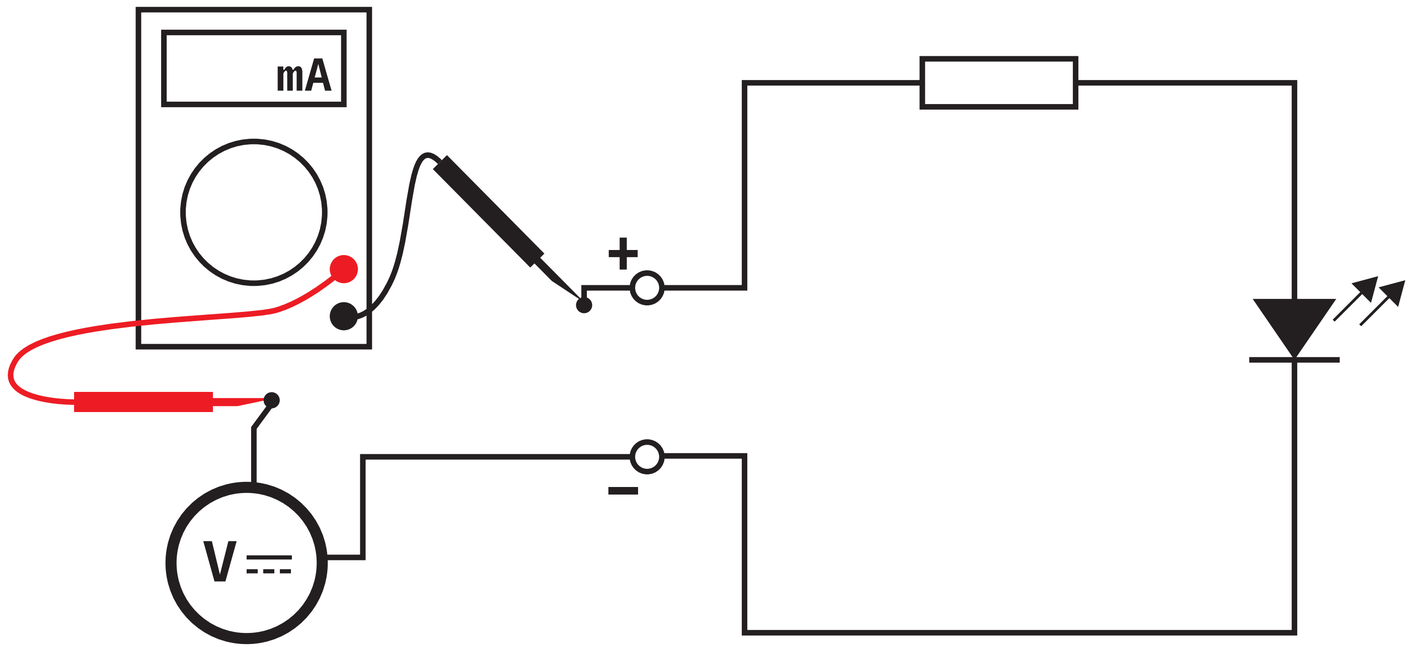
Multimeter

Multimeter


NCV
hFE
200m
2000m
20M
Ω
200k
20
V⎓
20k
200
2000
250
200
250
V~
200
10
°C
A⎓
200m
20m
2000µ

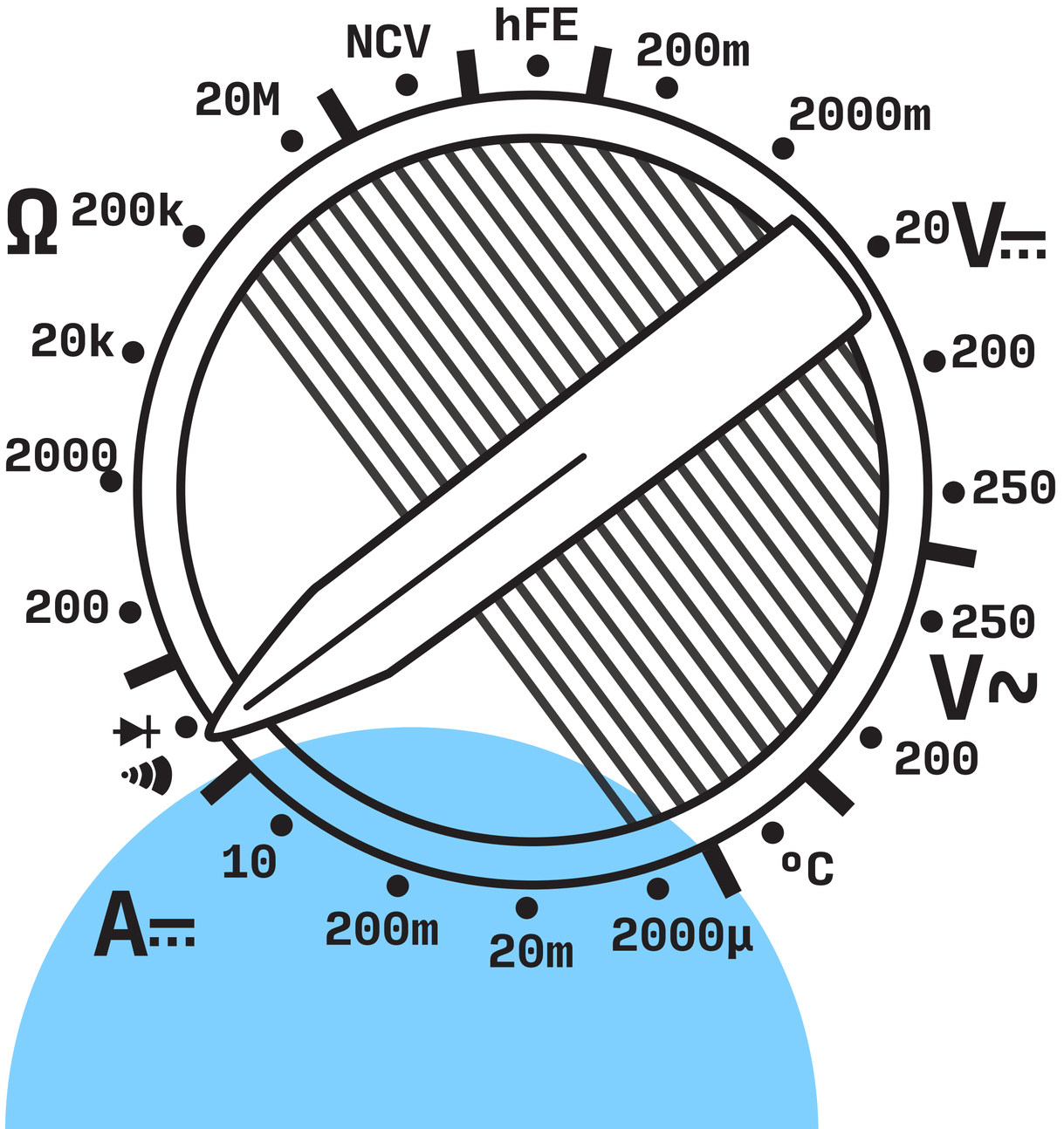


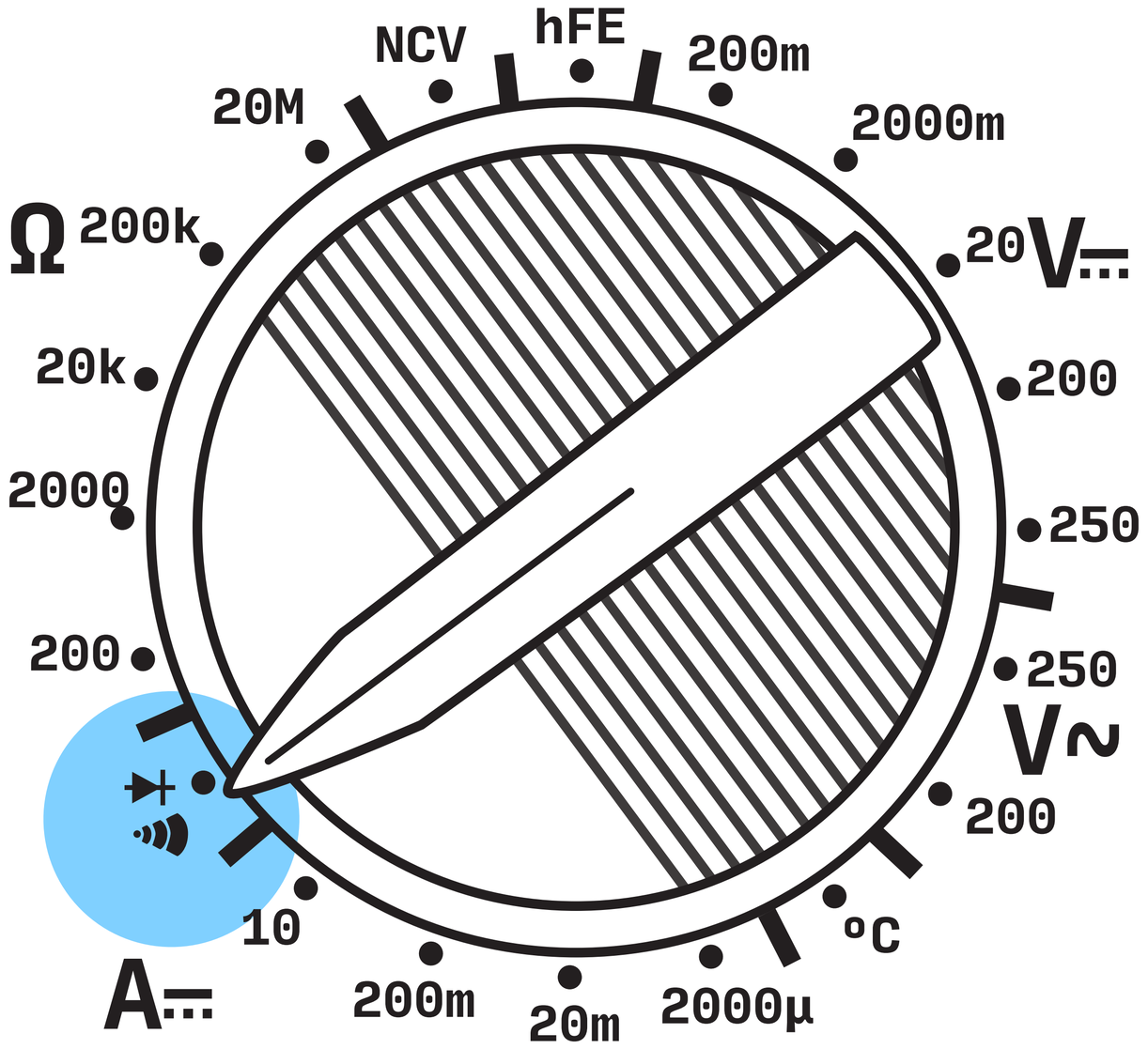
Multimeter



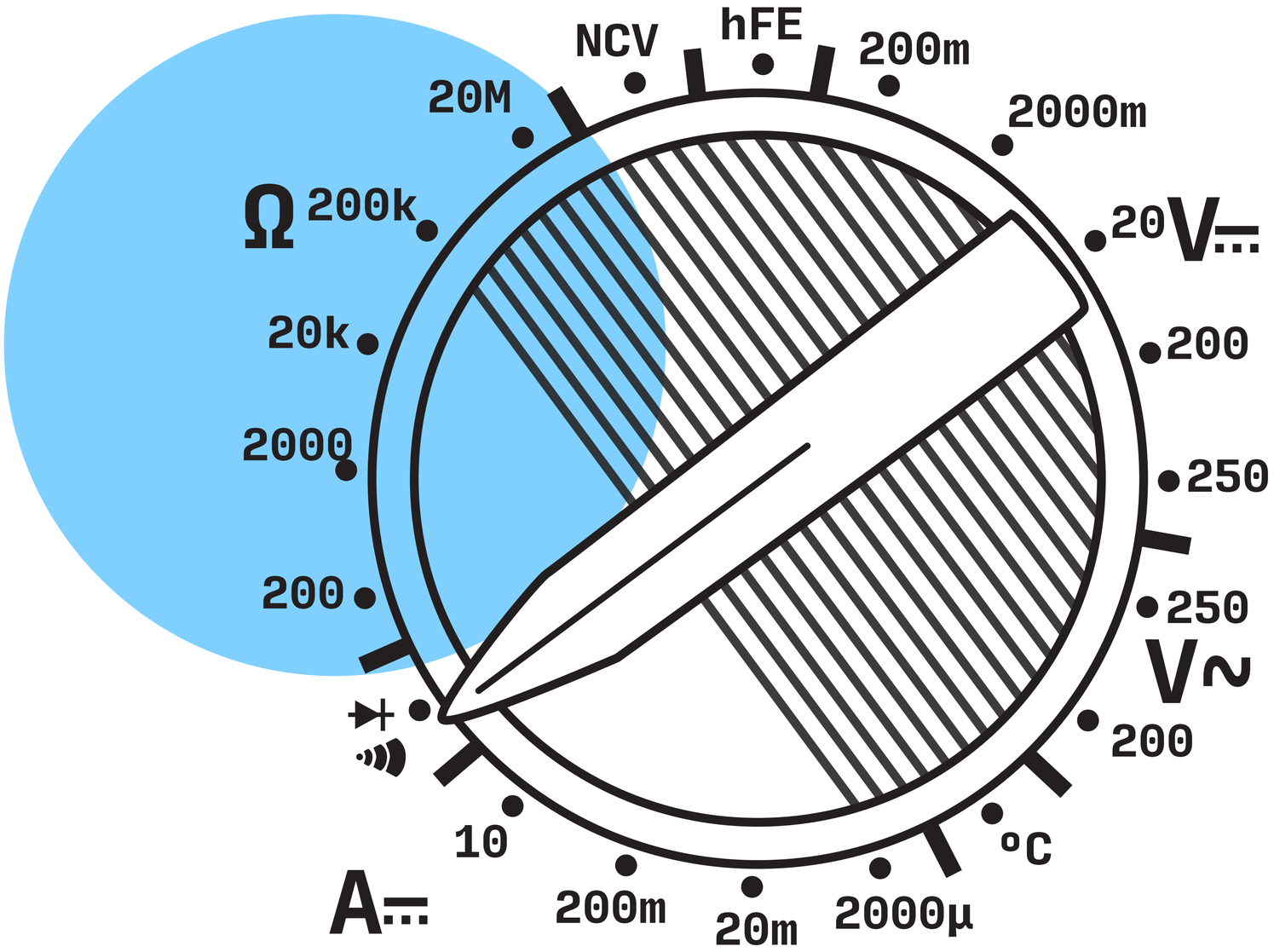
Multimeter

Multimeter

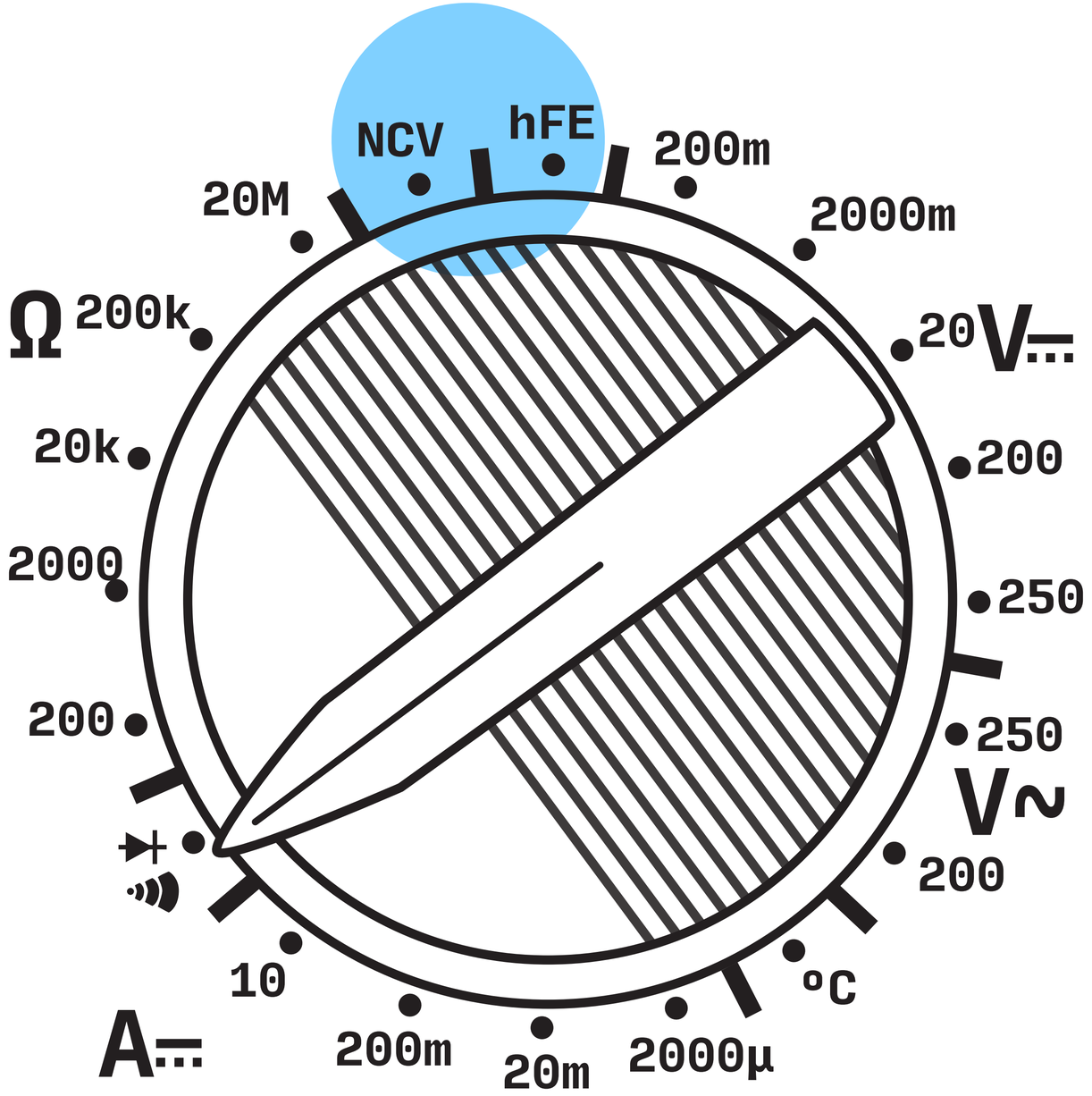

NCV
hFE
200m
2000m
20M
Ω
200k
20
V⎓
20k
200
2000
250
200
250
V~
200
10
°C
A⎓
200m
20m
2000µ


NCV
hFE
200m
2000m
20M
Ω
200k
20
V⎓
20k
200
2000
250
200
250
V~
200
10
°C
A⎓
200m
20m
2000µ

Multimeter


NCV
hFE
200m
2000m
20M
Ω
200k
20k
2000
200
20
V⎓
200
250
250
V~
200
°C
2000µ
20m
200m
10
A⎓

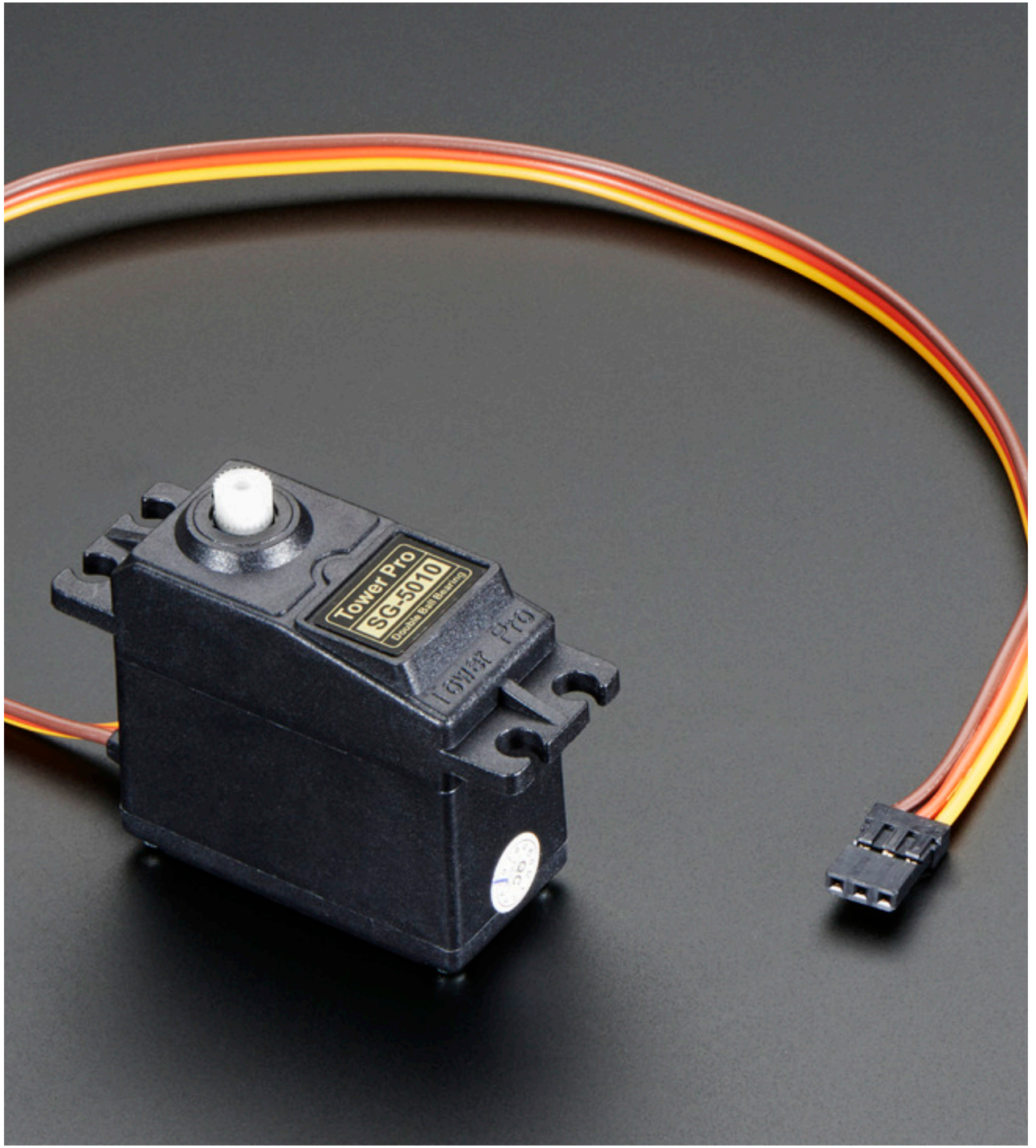
Multimeter


NCV
hFE
200m
2000m
20V⎓
200
250
250
V≂
200
°C
2000µ
20m
200m
10
A⎓
200
2000
20k
200k
Ω
20M

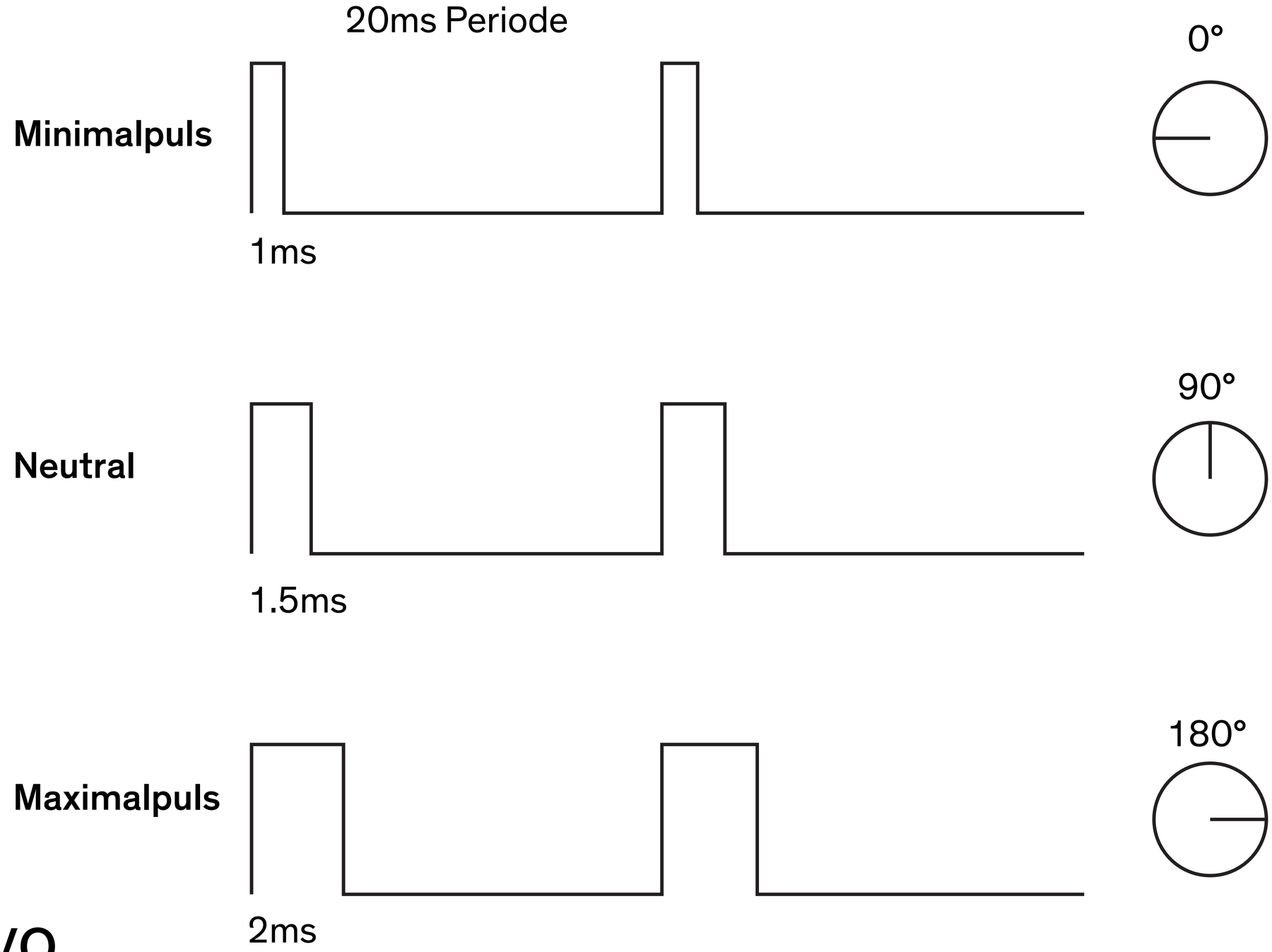
Multimeter

Motoren

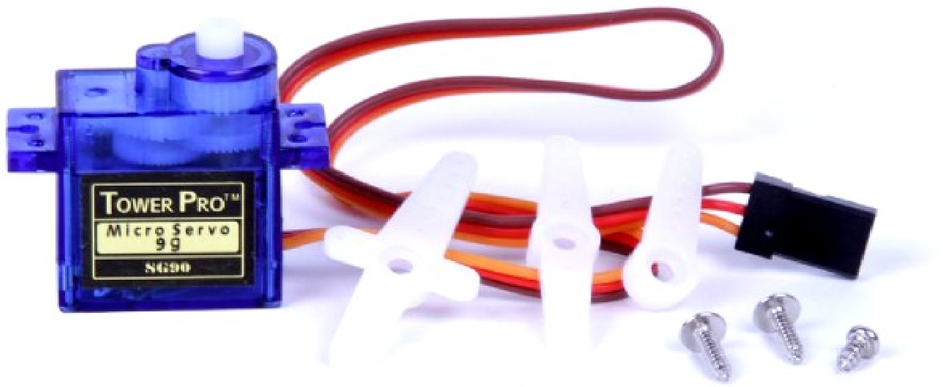
Servo



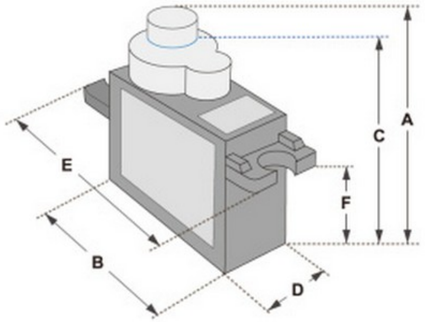
Servo


20ms Periode
Minimalpuls
1ms
0°
Neutral
1.5ms
90°
Maximalpuls
2ms
180°

SERVO MOTOR SG90 DATA SHEET

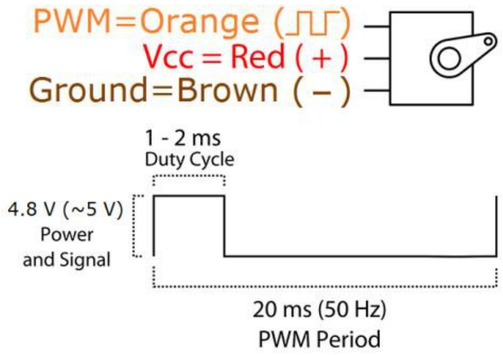


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

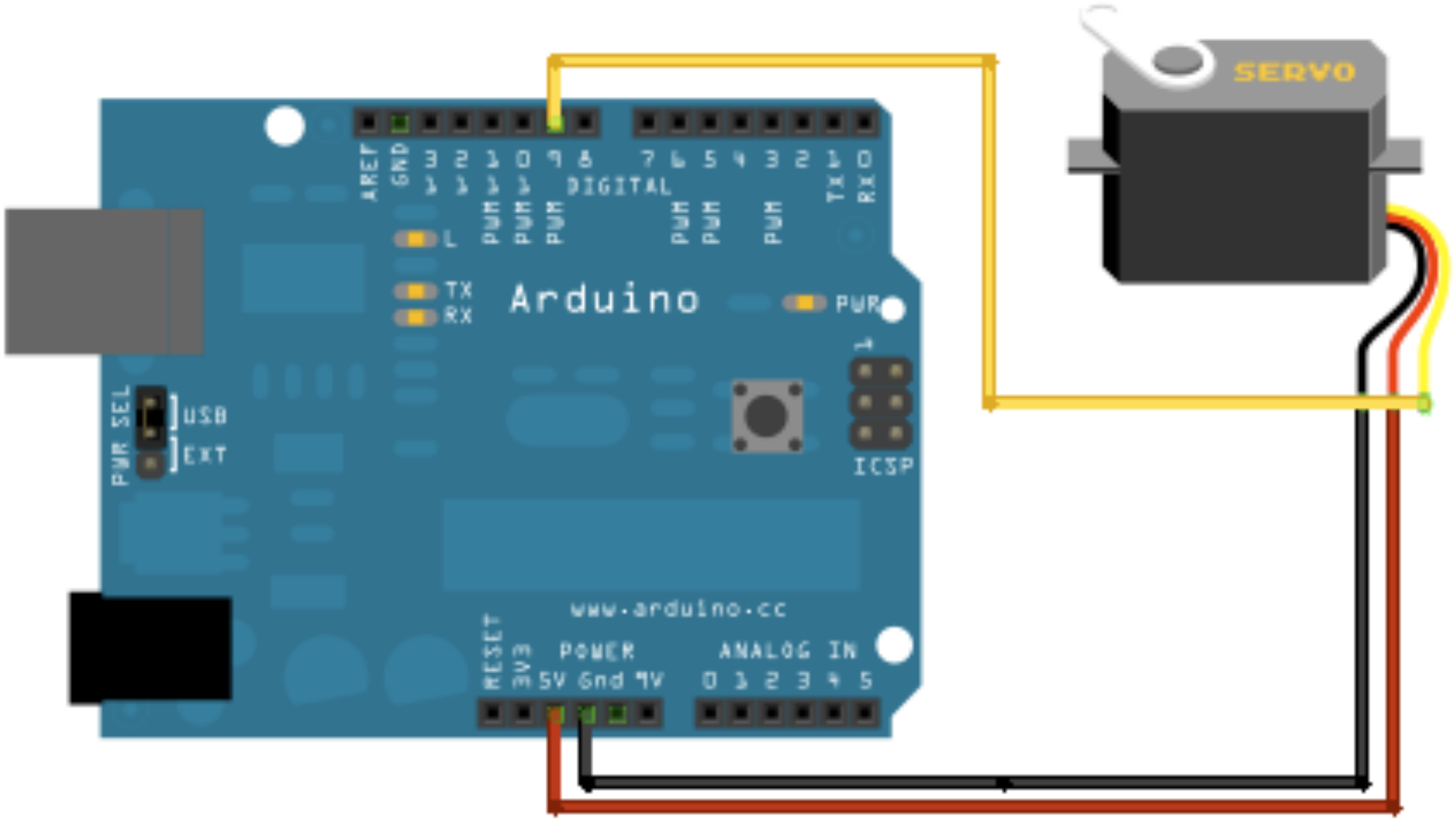


Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



Servo


SERVO
AREF
GND
DIGITAL
PWM
L
TX
RX
Arduino
PWR
PWR SEL
USB
EXT
ICSP
www.arduino.cc
RESET
3V3
POWER
5V
Gnd
Vin
ANALOG IN
0 1 2 3 4 5

Servo

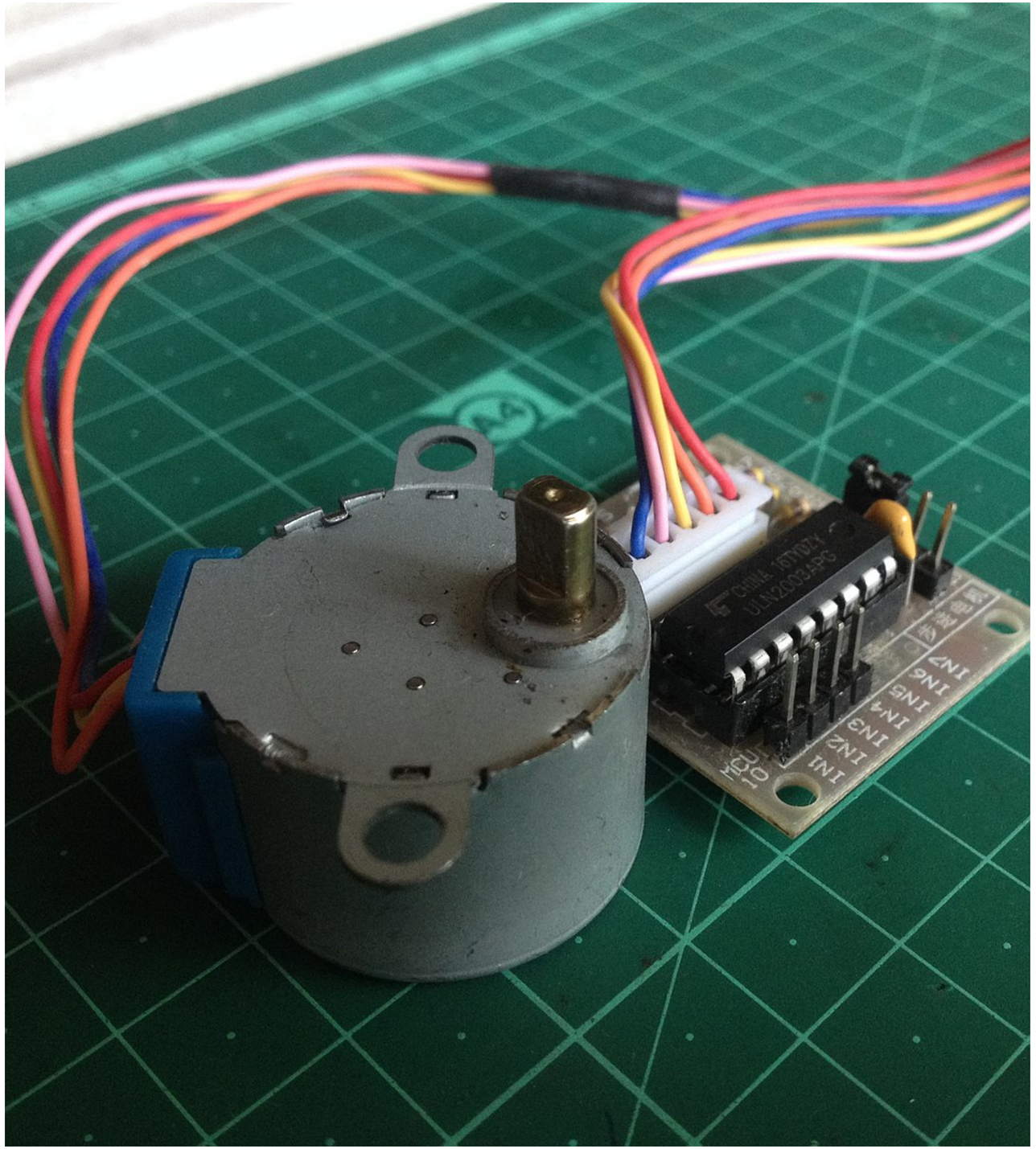
Servo

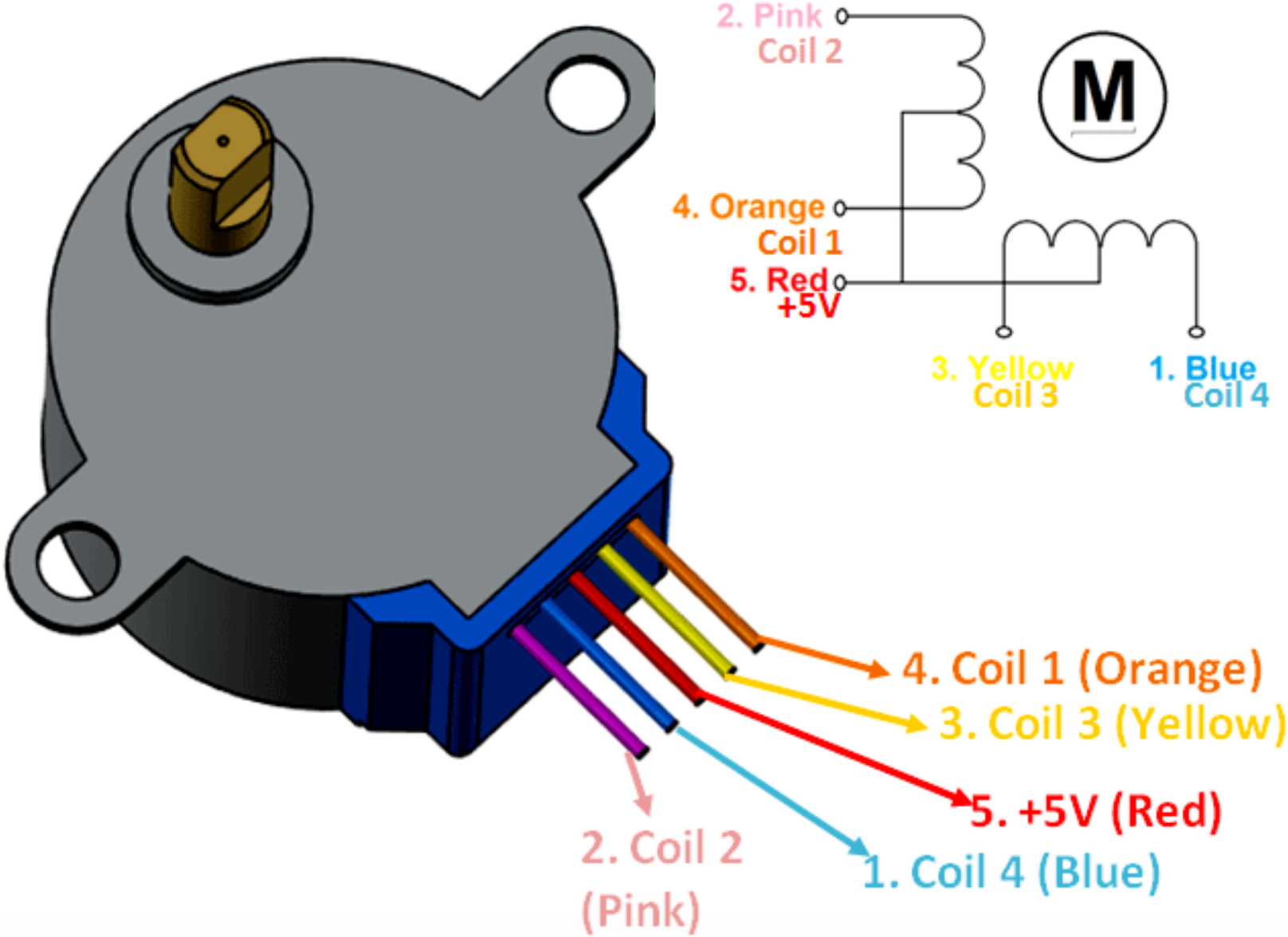

sketch_nov14a | Arduino 1.8.9
sketch_nov14a §
#include <Servo.h>
Servo motor;
void setup() {
motor.attach(9); // attaches the servo on pin 9 to the servo object
}
void loop() {
myservo.write(0);
delay(1000);
myservo.write(180);
delay(1000);
}

Pin Nummer	Signaltyp	Schema 1	Schema 2	Schema 3
1	GND (Erde)	Schwarz	Braun	Schwarz
2	VCC	Rot	Rot	Rot oder Braun
3	Steuerung	Weiß	Orange	Gelb oder Weiß

Servo

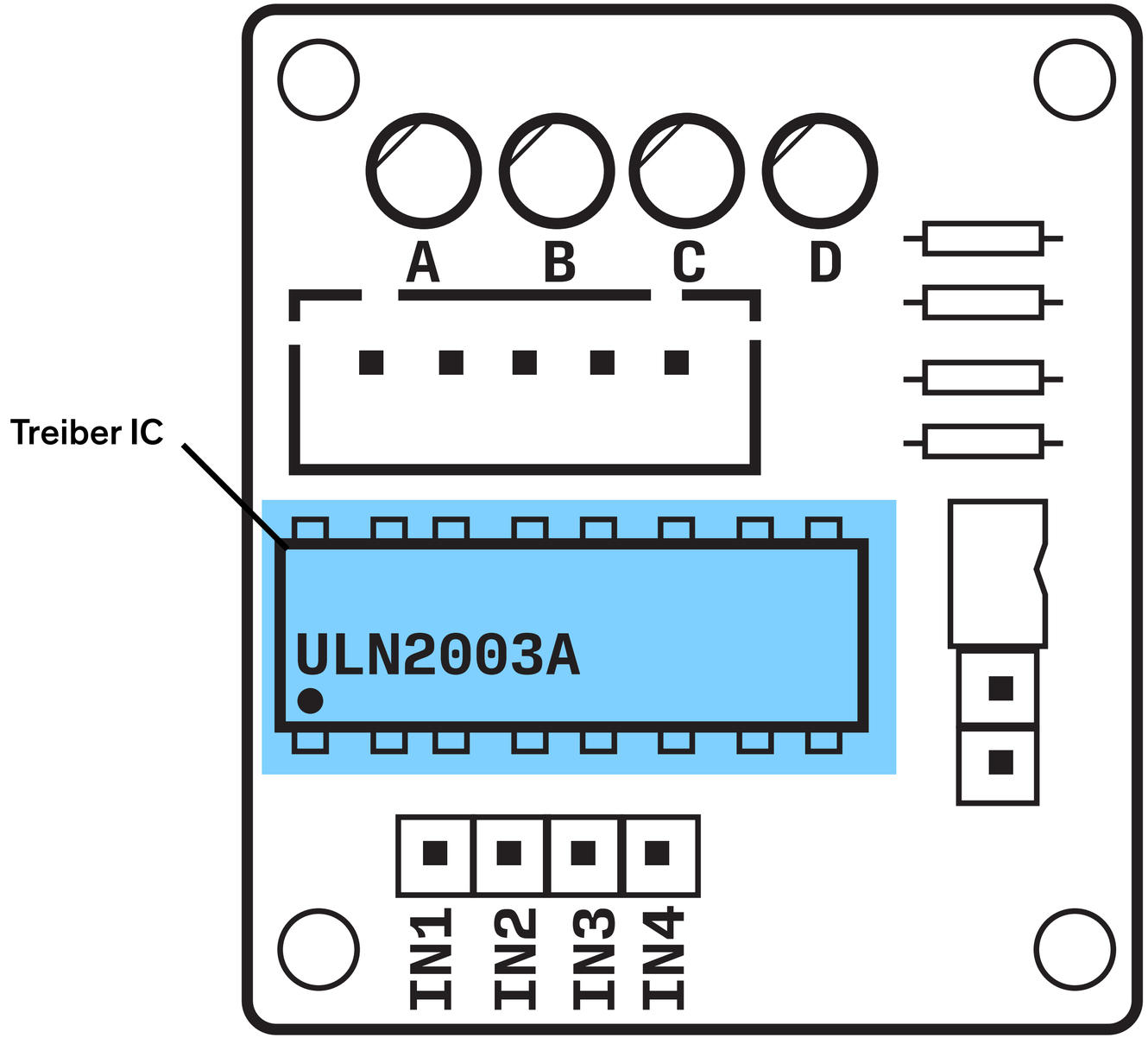
Schrittmotor



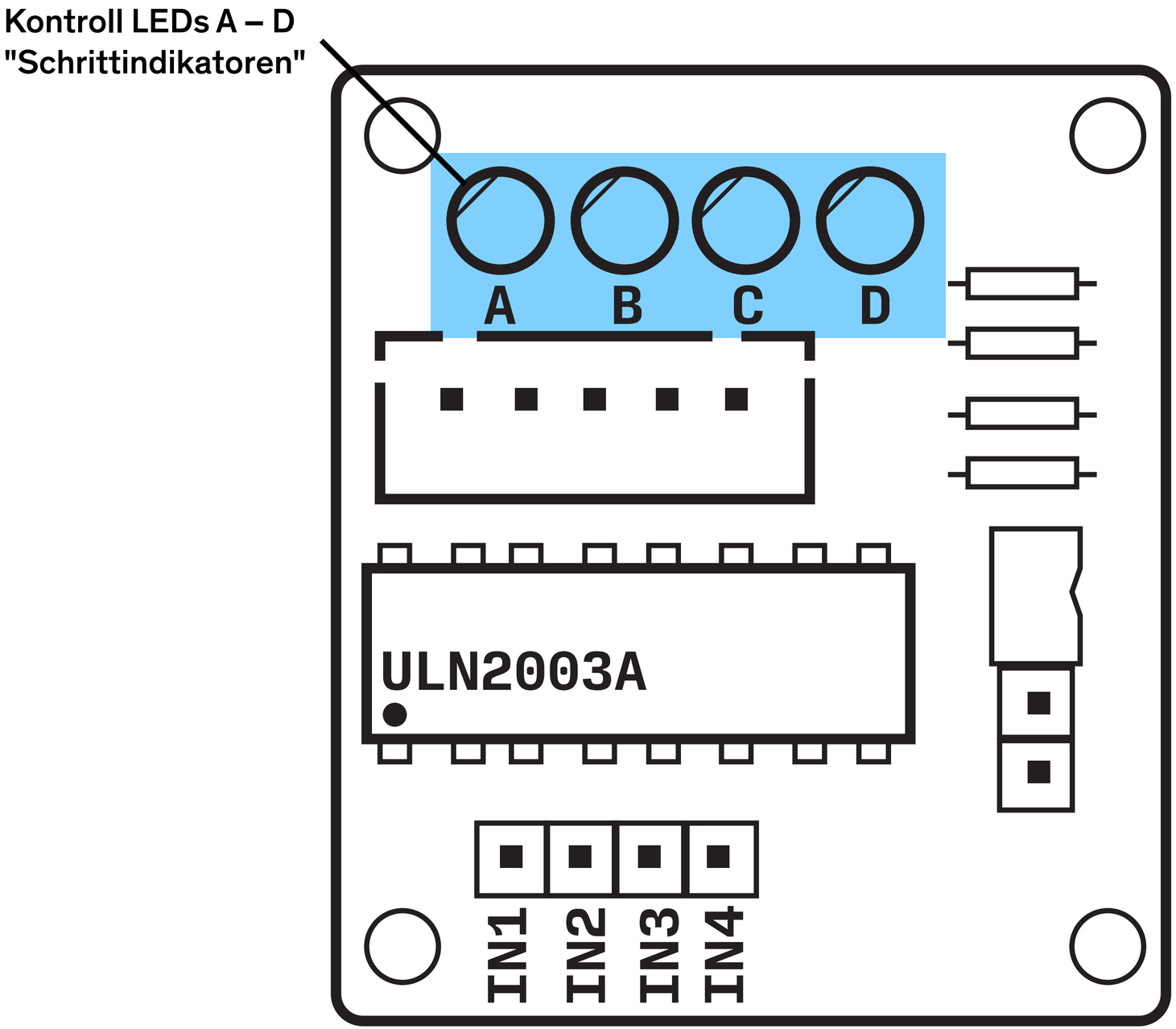


Schrittmotor

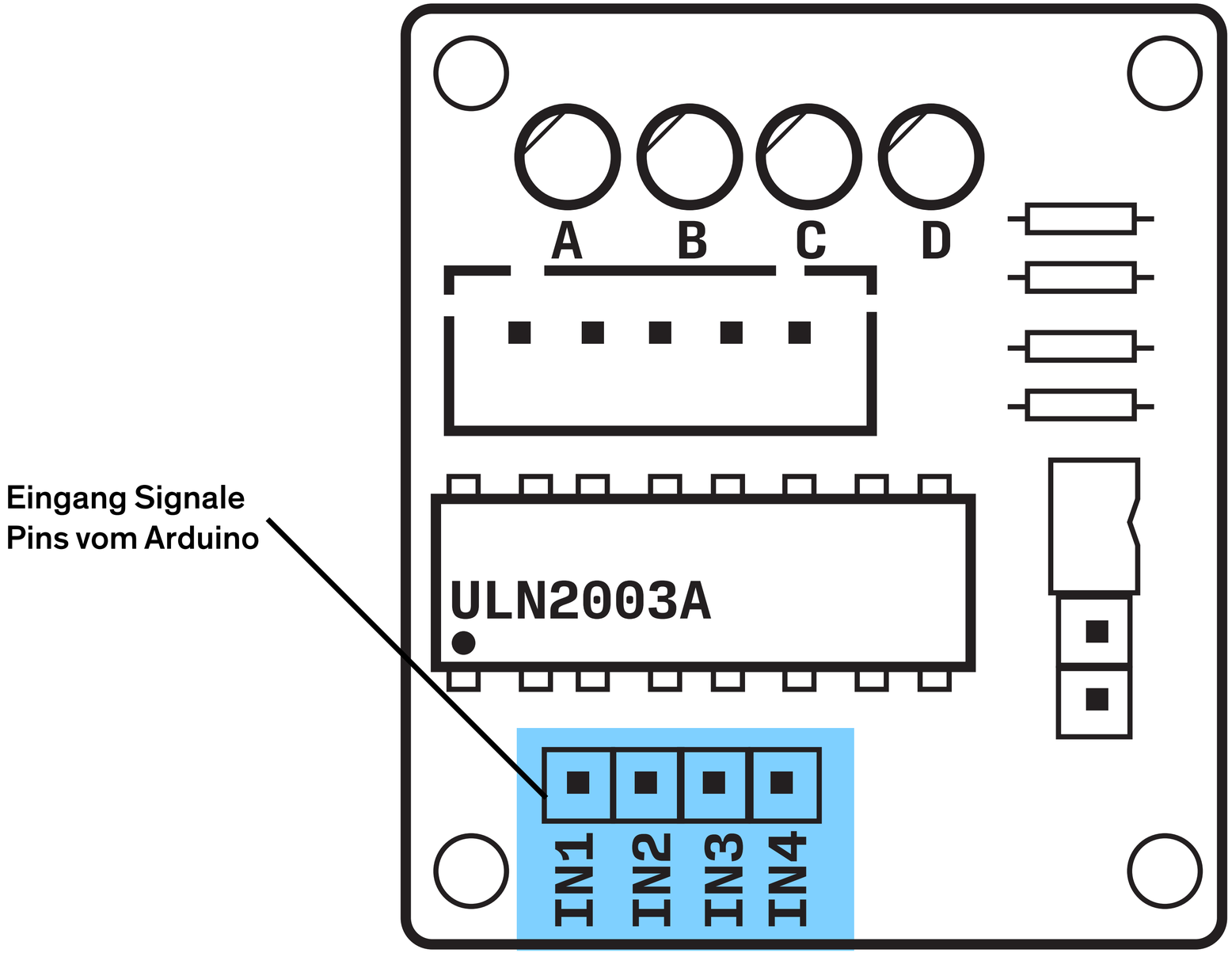
Schrittmotor



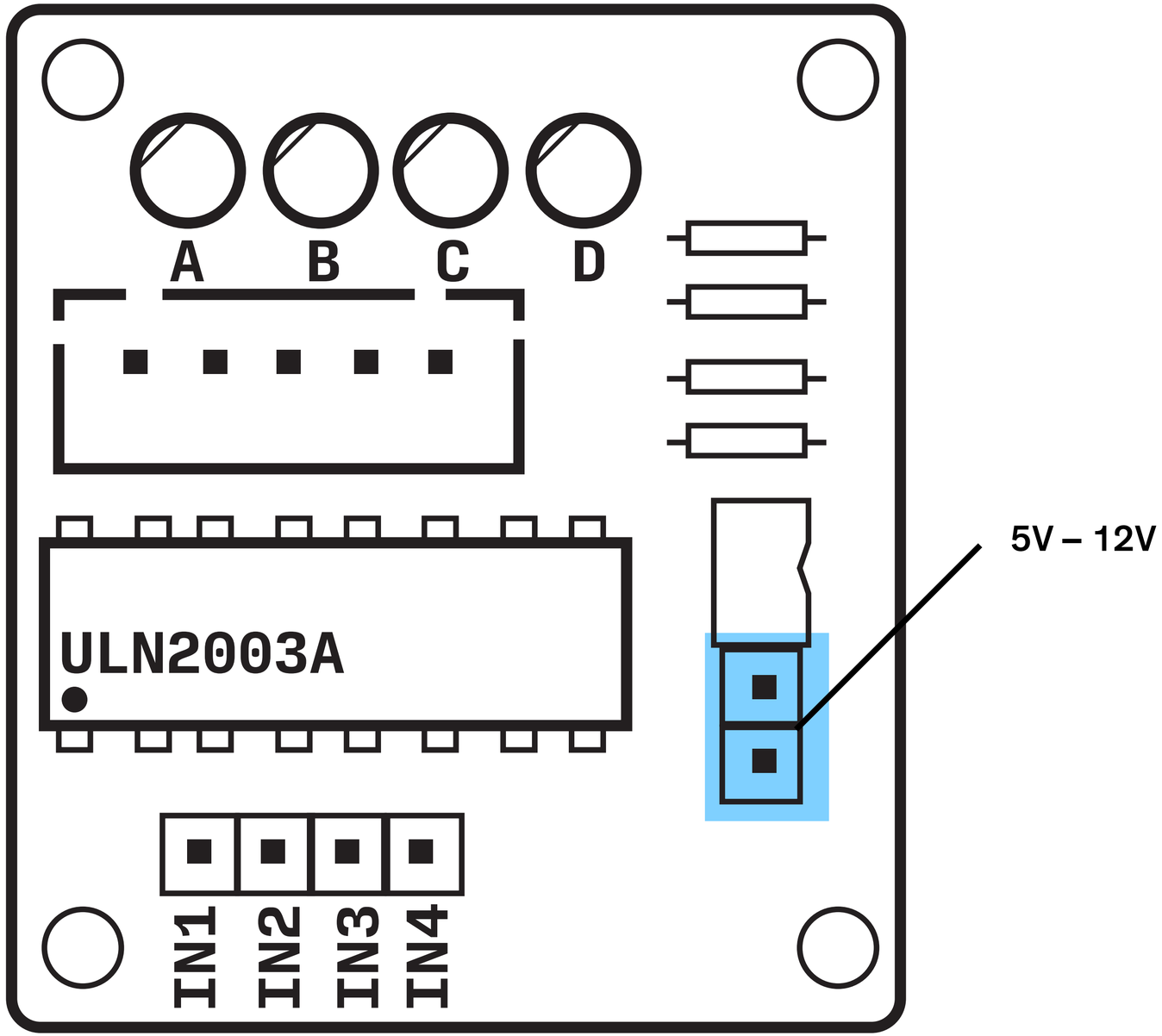
Schrittmotor



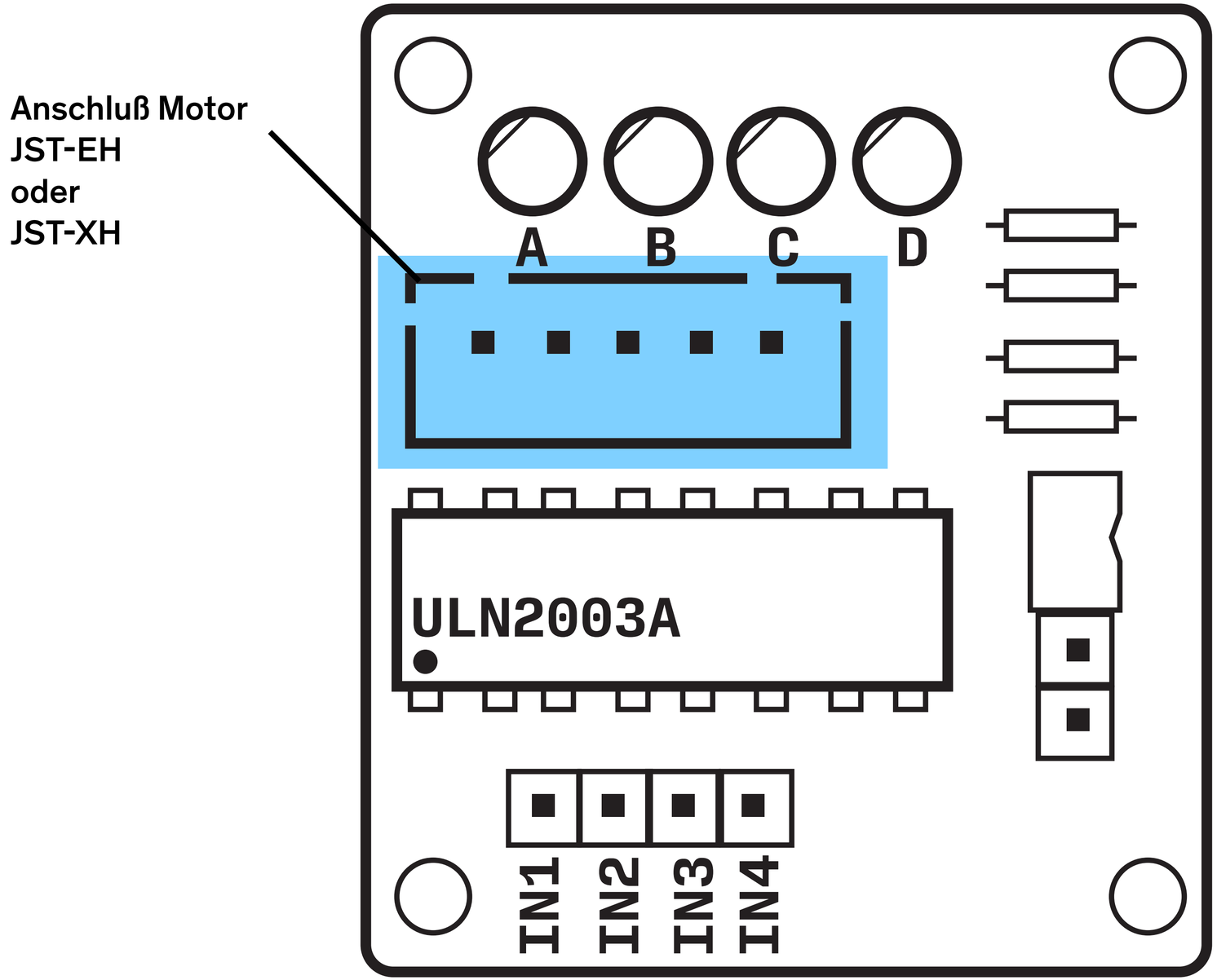
Schrittmotor



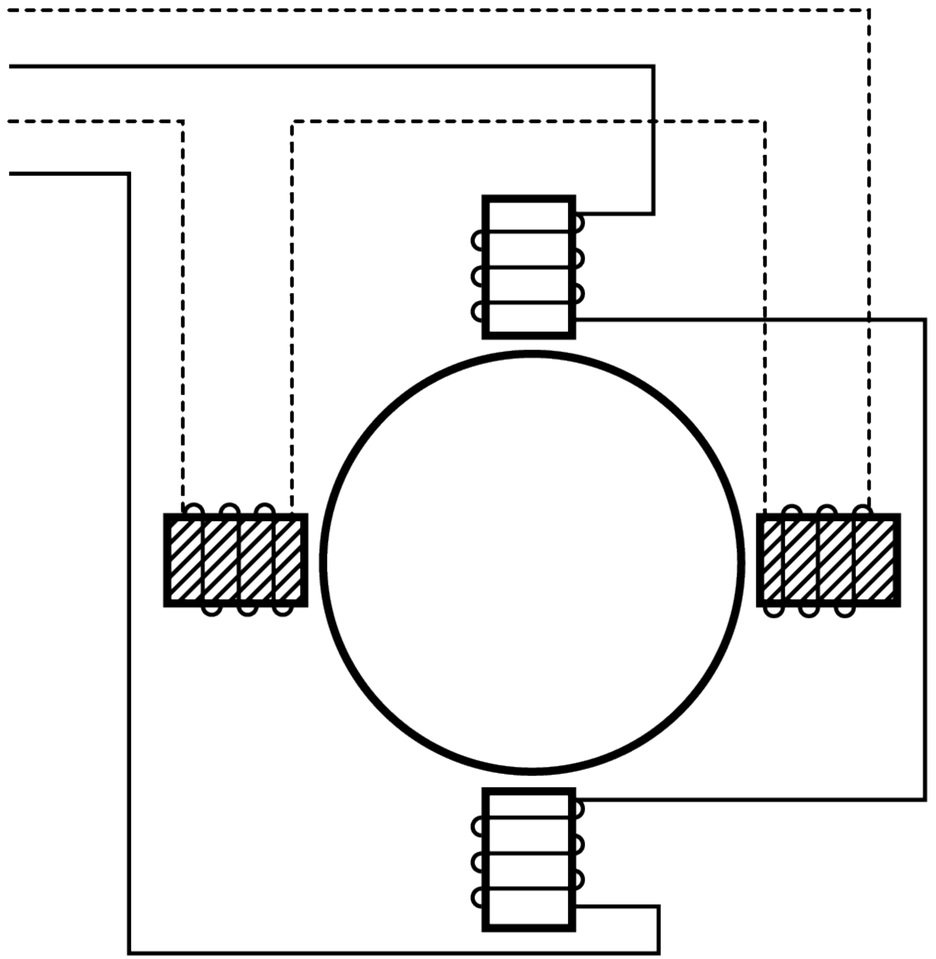
Schrittmotor



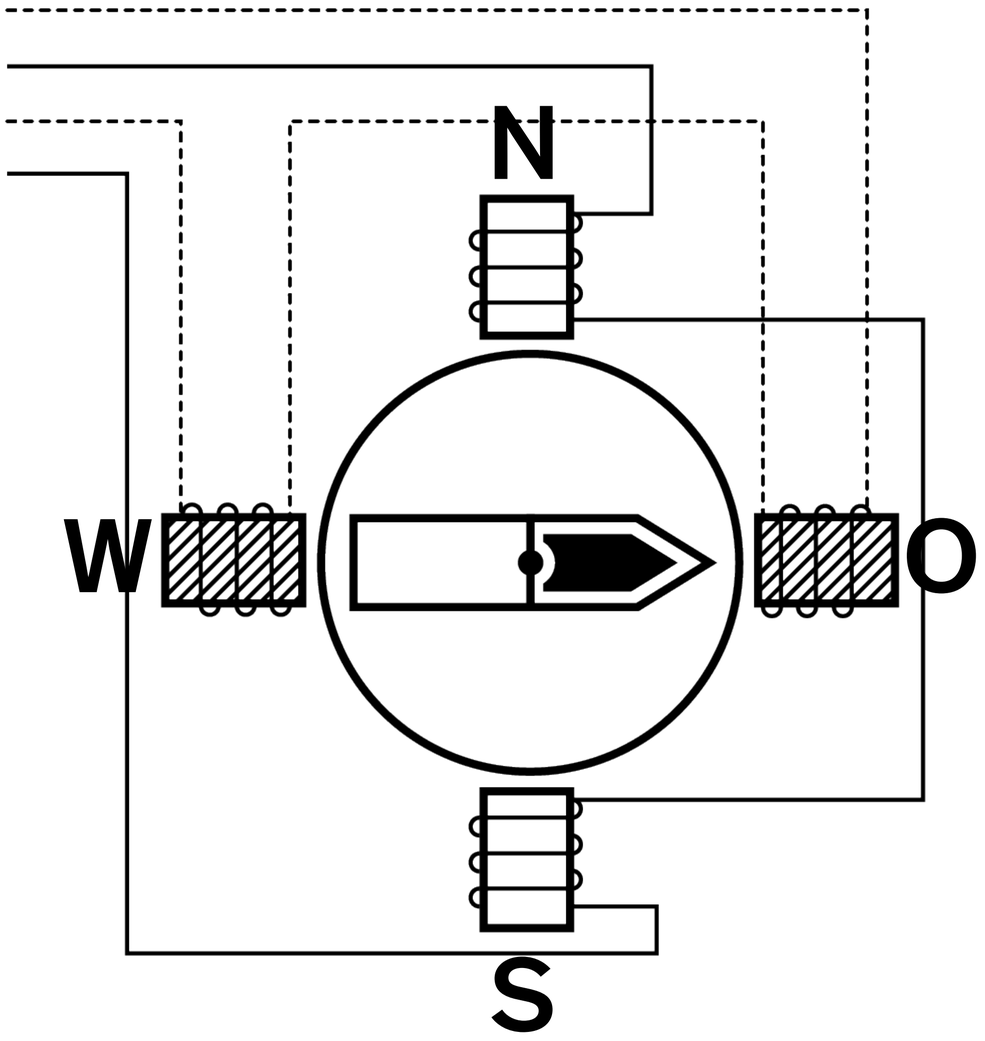
Schrittmotor



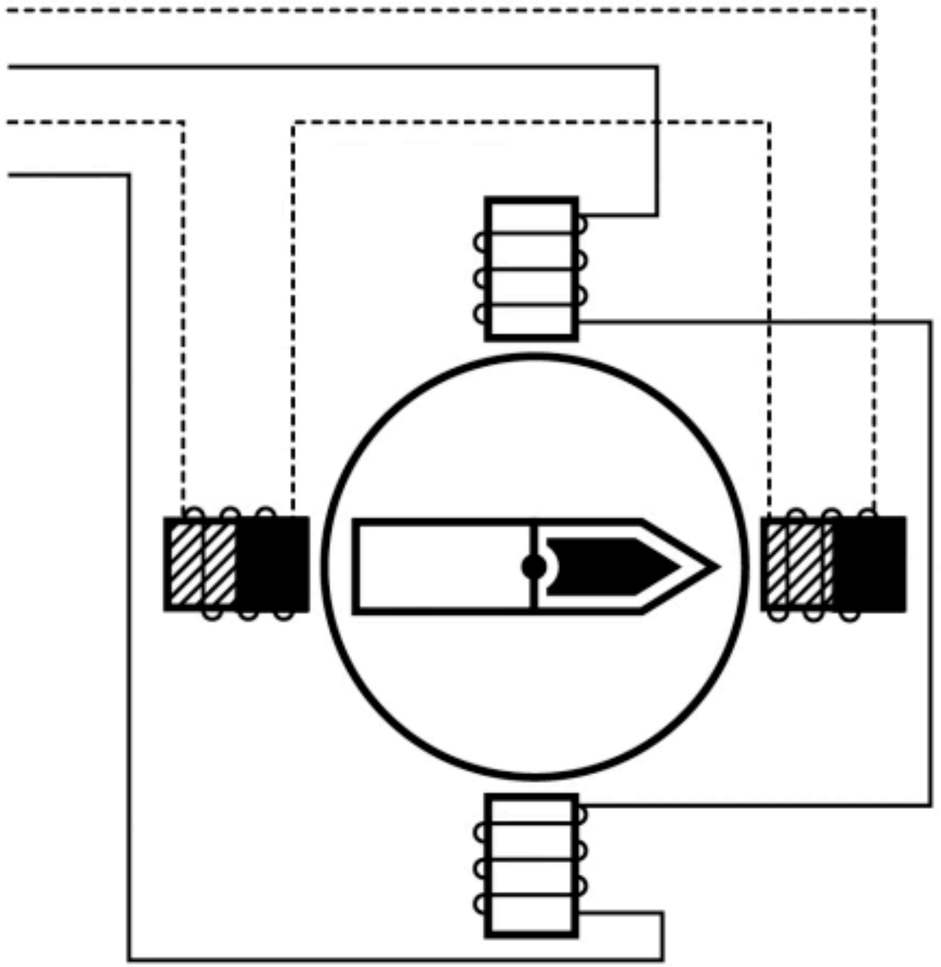
Schrittmotor



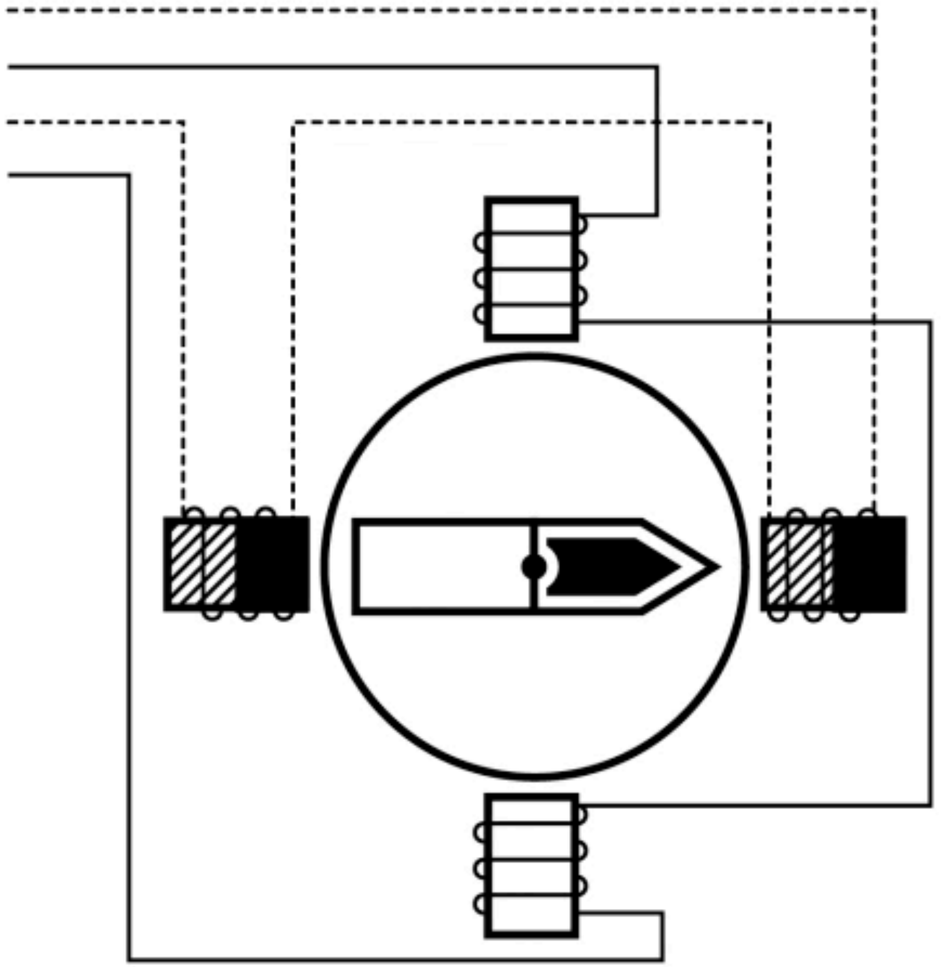
Schrittmotor



Schrittmotor



Schrittmotor



Schrittmotor

Schrittmotor

Non-blocking Code

Pro / Contra delay()

\+

- Einfach zu verstehen

–

- Blocking

Non-blocking Code

Pro / Contra delay()

\+

Einfach zu verstehen (Für Anfänger)

–

Blocking

Non-blocking Code

Pro / Contra delay()

+

Einfach zu verstehen (Für Anfänger)

–

Blocking (Bringt Probleme sobald der Code nur ein *wenig* Komplexer wird)

Non-blocking Code

```
int ledPin = 9;
int sensorPin = A0;
int value = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(sensorPin, INPUT);
}

void loop() {
  value = analogRead(sensorPin);
  value = map(value, 800, 1023, 0, 255);
  analogWrite(ledPin, value);
  delay(50); // unser arduino wird von hier an für 50ms in den *schlaf* gelegt

  // anderen code "parallel" ausführen? nicht möglich. bei zeitkritischem code schlecht
}
```

Non-blocking Code

```
int ledPin1 = 9;
int ledPin2 = 10;
int sensorPin = A0;
int value = 0;

void setup() {
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(sensorPin, INPUT);
}

void loop() {
  value = analogRead(sensorPin);
  value = map(value, 800, 1023, 0, 255);
  analogWrite(ledPin1, value);
  delay(1000); // << block
  analogWrite(ledPin2, value);
  delay(1000); // << block

  // anderen code "parallel" ausführen? nicht möglich. bei zeitkritischem code schlecht
}
```

Non-blocking Code

Interrupt Service Routine (kurz ISR)

1. "interrupts" (greift in den Prozess ein)
2. stellt ein Register auf timer0 um
3. wartet "exklusiv" bis der timer durchgelaufen ist
4. "zeit ist eingefroren"
5. danach wird die loop genau an der stelle fortgesetzt

Non-blocking Code

millis()

Gibt die Millisekunden zurück, die verstrichen sind seit dem unser Arduino "an" ist.

Non-blocking Code

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

void loop() {

}
```

Non-blocking Code

```
long time = 0;
long interval = 1000;

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

void loop() {

}
```

Non-blocking Code

```
long time = 0;
long interval = 1000;

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

void loop() {
  if(millis() - time > interval) {
  }
}
```

Non-blocking Code

```
long time = 0;
long interval = 1000;

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

void loop() {
  if(millis() - time > interval) {
  }
}
```

Cycle n:
if(n – time > interval) {
…
}

Non-blocking Code

Cycle n	millis()	time	interval
0	0	0	1000
1	1	0	1000
2	2	0	1000
3	3	0	1000
4	4	0	1000
…	…	…	…
10	10	0	1000
…	…	…	…
1001	1001	0	1000

Non-blocking Code

```
long time = 0;
long interval = 1000;


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


void loop() {
  if(millis() - time > interval) {
  }
}
```

Cycle 0:

if(0 – 0 > 1000) {

…

}

if(1001 – 0 > 1000) {

…

}

Non-blocking Code

```
long time = 0;
long interval = 1000;

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

void loop() {
  if(millis() - time > interval) {
    time = millis();
  }
}
```

Non-blocking Code

```
long time = 0;
long interval = 1000;

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

void loop() {
  if(millis() - time > interval) {
    time = millis();
    // unser code, jetzt mit timing und ohne delay()
  }
}
```

Non-blocking Code

```
long time0 = 0;
long interval0 = 1000;
long time1 = 0;
long interval1 = 2500;

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

void loop() {
  if(millis() - time0 > interval0) {
    time0 = millis();
    Serial.println("jede sekunde");
  }
  if(millis() - time1 > interval1) {
    time1 = millis();
    Serial.println("alle 2.5 sekunden");
  }
}
```

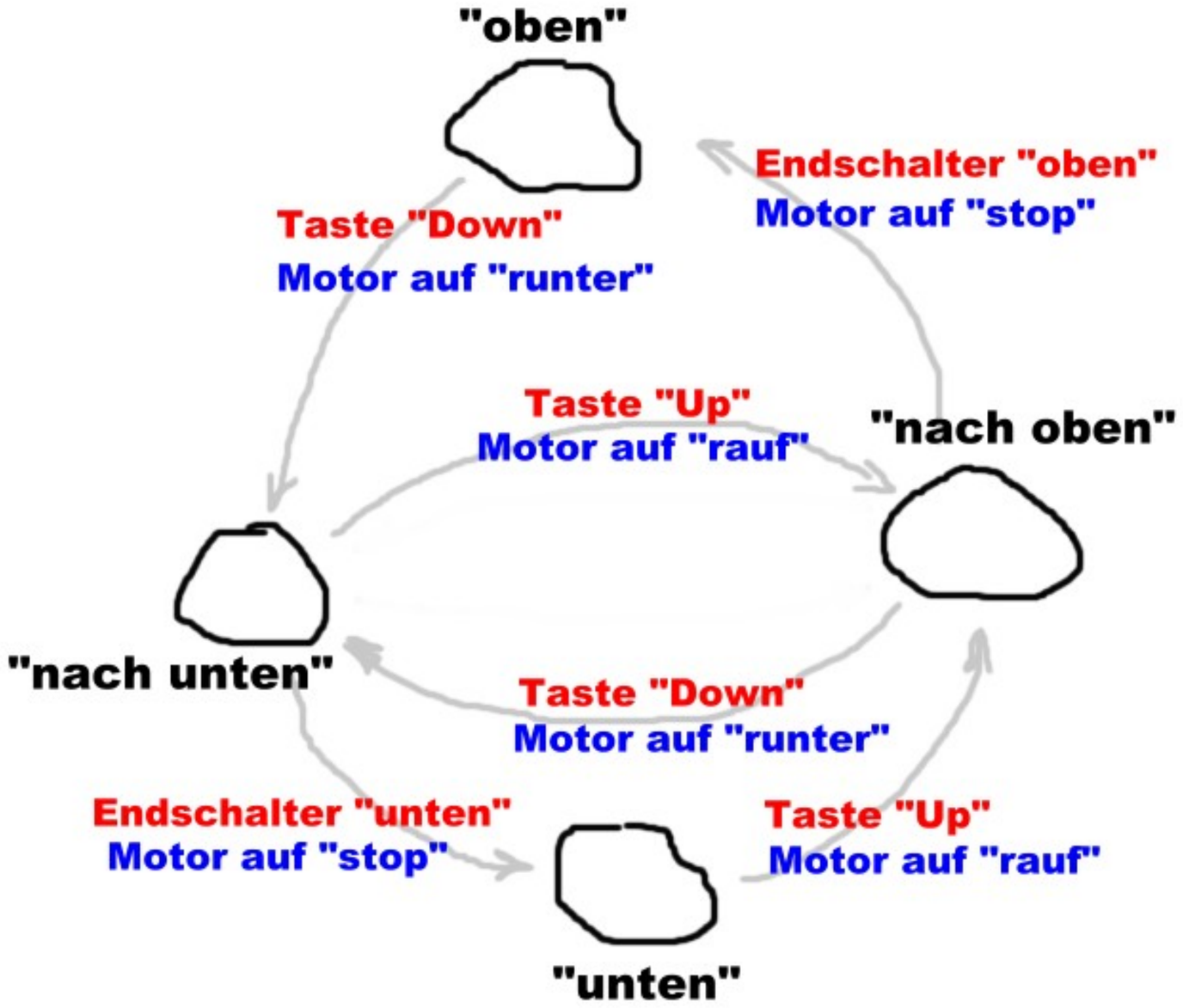
Non-blocking Code

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  // …
  // und noch mehr code
  // und was anderes soll berechnet werden
  // … mehr code
  // außerdem soll ein knopf benutzt werden, der unser arduino durch versch. modi
  // durchschalten kann, z.B. ist unser arduino nur ein "interface"
}
```

Endlicher Zustandsautomat

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  // …
  // und noch mehr code
  // und was anderes soll berechnet werden
  // … mehr code
  // außerdem soll ein knopf benutzt werden, der unser arduino durch versch. modi
  // durchschalten kann, z.B. ist unser arduino nur ein "interface"
}
```

Endlicher Zustandsautomat



Endlicher Zustandsautomat

```
#define WARTEN          0
#define BEARBEITEN      1
#define AUSGABE         2
int state = 0;

void setup() {}

void loop() {
  stateMachine();
}
```

```
void stateMachine() {
  switch( state ) {
    case WARTEN:
      state = BEARBEITEN;
    break;

    case BEARBEITEN:
      state = AUSGABE;
    break;

    case AUSGABE:
      state = WARTEN;
    break;
  }
}
```

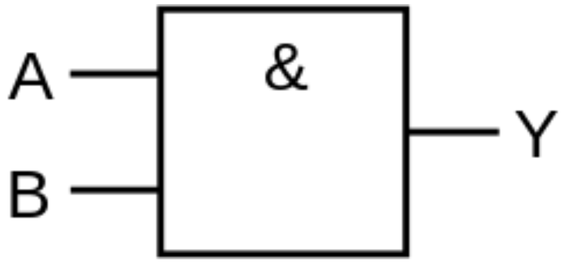
Endlicher Zustandsautomat

```
#define WARTEN          0
#define BEARBEITEN      1
#define AUSGABE         2
int state = 0;

void setup() {}

void loop() {
  stateMachine();
}
```

```
void stateMachine() {
  switch( state ) {
    case WARTEN:
      // neuer code hier
      // oder funktion aufrufen
      state = BEARBEITEN;
    break;

    case BEARBEITEN:
      // neuer code hier
      state = AUSGABE;
    break;

    case AUSGABE:
      // neuer code hier
      state = WARTEN;
    break;
  }
}
```

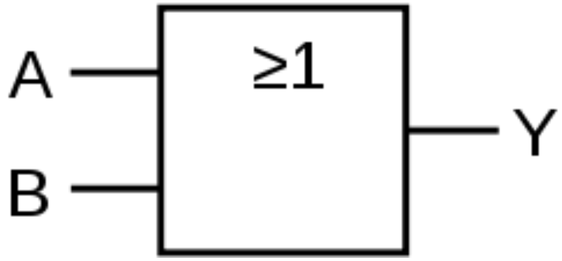
Endlicher Zustandsautomat

Logikgatter



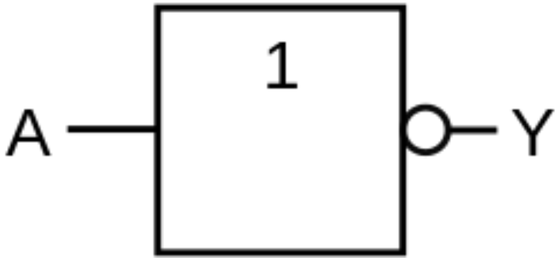
A	B	Y
FALSCH	FALSCH	FALSCH
FALSCH	WAHR	FALSCH
WAHR	FALSCH	FALSCH
WAHR	WAHR	WAHR

AND



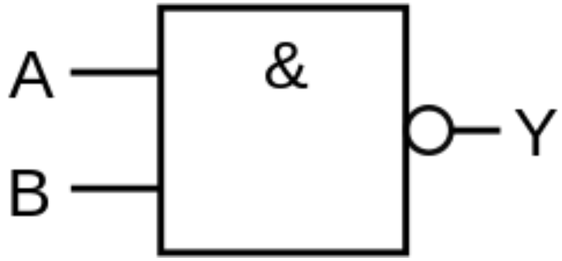
A	B	Y
FALSCH	FALSCH	FALSCH
FALSCH	WAHR	WAHR
WAHR	FALSCH	WAHR
WAHR	WAHR	WAHR

OR



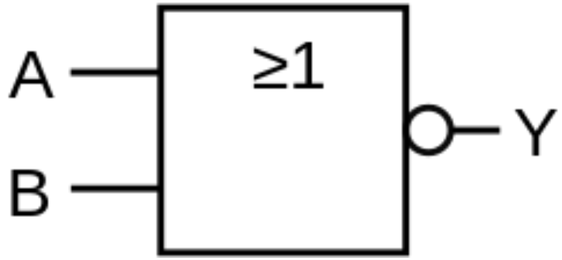
A	Y
FALSCH	WAHR
WAHR	FALSCH

NOT



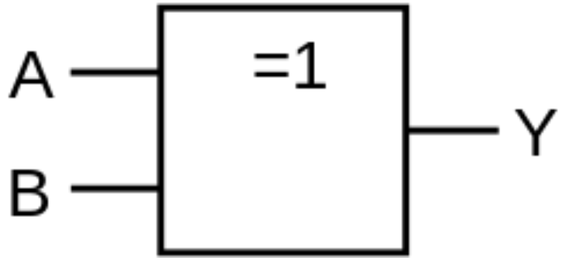
A	B	Y
FALSCH	FALSCH	WAHR
FALSCH	WAHR	WAHR
WAHR	FALSCH	WAHR
WAHR	WAHR	FALSCH

NAND



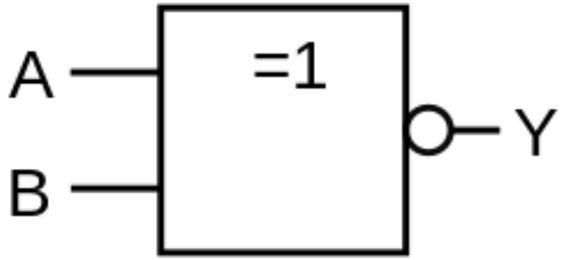
A	B	Y
FALSCH	FALSCH	WAHR
FALSCH	WAHR	FALSCH
WAHR	FALSCH	FALSCH
WAHR	WAHR	FALSCH

NOR



A	B	Y
FALSCH	FALSCH	FALSCH
FALSCH	WAHR	WAHR
WAHR	FALSCH	WAHR
WAHR	WAHR	FALSCH

XOR



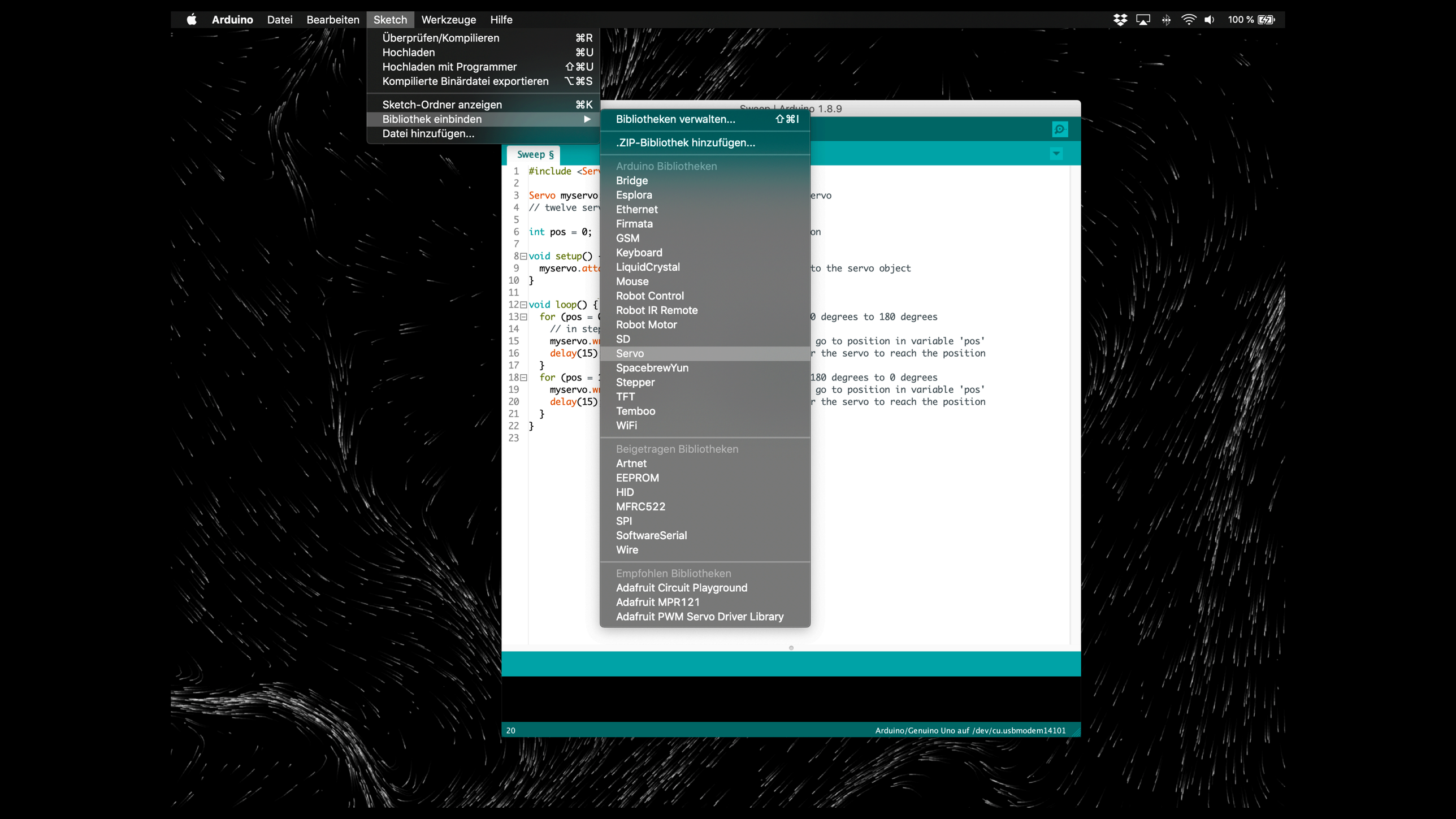
A	B	Y
FALSCH	FALSCH	WAHR
FALSCH	WAHR	FALSCH
WAHR	FALSCH	FALSCH
WAHR	WAHR	WAHR

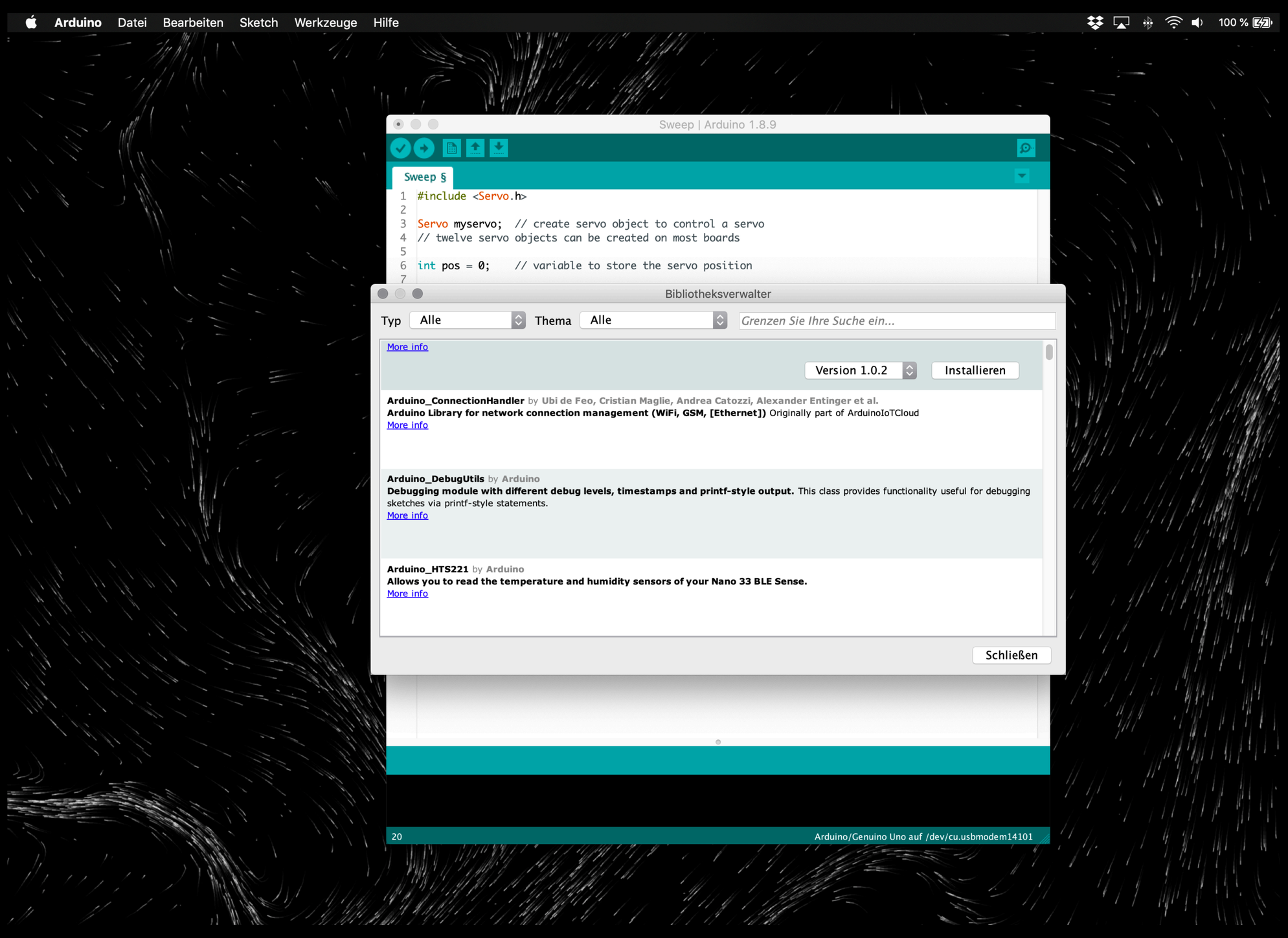
XNOR

NOT

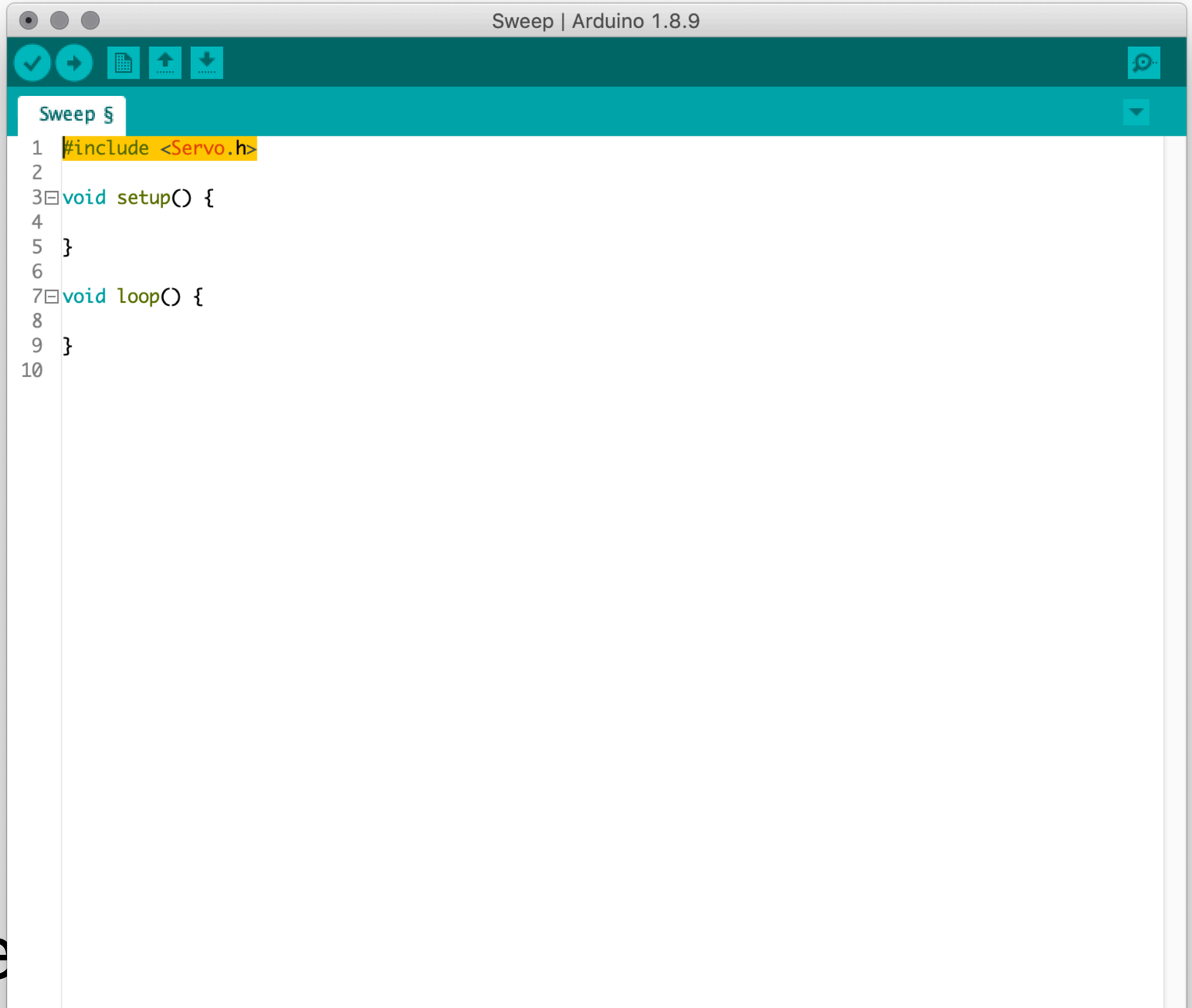
Bibliotheken

Bibliotheken


Arduino Datei Bearbeiten Sketch Werkzeuge Hilfe
100 %
Überprüfen/Kompilieren ⌘R
Hochladen ⌘U
Hochladen mit Programmer ⇧⌘U
Kompilierte Binärdatei exportieren ⌥⌘S
Sketch-Ordner anzeigen ⌘K
Bibliothek einbinden
Datei hinzufügen...
Bibliotheken verwalten... ⇧⌘I
.ZIP-Bibliothek hinzufügen...
Arduino Bibliotheken
Bridge
Esplora
Ethernet
Firmata
GSM
Keyboard
LiquidCrystal
Mouse
Robot Control
Robot IR Remote
Robot Motor
SD
Servo
SpacebrewYun
Stepper
TFT
Temboo
WiFi
Beigetragen Bibliotheken
Artnet
EEPROM
HID
MFRC522
SPI
SoftwareSerial
Wire
Empfohlen Bibliotheken
Adafruit Circuit Playground
Adafruit MPR121
Adafruit PWM Servo Driver Library
Sweep | Arduino 1.8.9
Sweep §
#include <Serv
Servo myservo
// twelve serv
int pos = 0;
void setup() {
myservo.atte
}
void loop() {
for (pos = 0
// in step
myservo.wr
delay(15)
}
for (pos = 1
myservo.wr
delay(15)
}
}
ervo
on
to the servo object
0 degrees to 180 degrees
go to position in variable 'pos'
r the servo to reach the position
180 degrees to 0 degrees
go to position in variable 'pos'
r the servo to reach the position
20
Arduino/Genuino Uno auf /dev/cu.usbmodem14101


Arduino Datei Bearbeiten Sketch Werkzeuge Hilfe
100 %
Sweep | Arduino 1.8.9
Sweep §
1 #include <Servo.h>
2
3 Servo myservo; // create servo object to control a servo
4 // twelve servo objects can be created on most boards
5
6 int pos = 0; // variable to store the servo position
7
Bibliotheksverwalter
Typ Alle Thema Alle Grenzen Sie Ihre Suche ein...
More info
Version 1.0.2 Installieren
Arduino_ConnectionHandler by Ubi de Feo, Cristian Maglie, Andrea Catozzi, Alexander Entinger et al.
Arduino Library for network connection management (WiFi, GSM, [Ethernet]) Originally part of ArduinoIoTCloud
More info
Arduino_DebugUtils by Arduino
Debugging module with different debug levels, timestamps and printf-style output. This class provides functionality useful for debugging sketches via printf-style statements.
More info
Arduino_HTS221 by Arduino
Allows you to read the temperature and humidity sensors of your Nano 33 BLE Sense.
More info
Schließen
20
Arduino/Genuino Uno auf /dev/cu.usbmodem14101

Bibliotheke



Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield, Arduino Ethernet Shield 2 and Arduino Leonardo ETH
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GSM - for connecting to a GSM/GRPS network with the GSM shield.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- Stepper - for controlling stepper motors
- TFT - for drawing text , images, and shapes on the Arduino TFT screen
- WiFi - for connecting to the internet using the Arduino WiFi shield
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

The Matrix and Sprite libraries are no longer part of the core distribution.

Bibliotheken

Contributed Libraries

If you're using one of these libraries, you need to install it first. See these instructions for details on installation. There's also a tutorial on writing your own libraries.

Communication (networking and protocols):

- Messenger - for processing text-based messages from the computer
- NewSoftSerial - an improved version of the SoftwareSerial library
- OneWire - control devices (from Dallas Semiconductor) that use the One Wire protocol.
- PS2Keyboard - read characters from a PS2 keyboard.
- Simple Message System - send messages between Arduino and the computer
- SSerial2Mobile - send text messages or emails using a cell phone (via AT commands over software serial)
- Webduino - extensible web server library (for use with the Arduino Ethernet Shield)
- X10 - Sending X10 signals over AC power lines
- XBee - for communicating with XBees in API mode
- SerialControl - Remote control other Arduinos over a serial connection

Bibliotheken

Sensing:

- Capacitive Sensing - turn two or more pins into capacitive sensors
- Debounce - for reading noisy digital inputs (e.g. from buttons)

Displays and LEDs:

- GFX - base class with standard graphics routines (by Adafruit Industries)
- GLCD - graphics routines for LCD based on the KS0108 or equivalent chipset.
- Improved LCD library fixes LCD initialization bugs in official Arduino LCD library
- LedControl - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.
- LedControl - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
- LedDisplay - control of a HCMS-29xx scrolling LED display.
- Matrix - Basic LED Matrix display manipulation library
- PCD8544 - for the LCD controller on Nokia 55100-like displays (by Adafruit Industries)
- Sprite - Basic image sprite manipulation library for use in animations with an LED matrix
- ST7735 - for the LCD controller on a 1.8", 128x160 TFT screen (by Adafruit Industries)

Bibliotheken

Bibliotheken

Audio and Waveforms:

- FFT - frequency analysis of audio or other analog signals
- Tone - generate audio frequency square waves in the background on any microcontroller pin

Motors and PWM:

- TLC5940 - 16 channel 12 bit PWM controller.

Timing:

- DateTime - a library for keeping track of the current date and time in software.
- Metro - help you time actions at regular intervals
- MsTimer2 - uses the timer 2 interrupt to trigger an action every N milliseconds.

Utilities:

- PString - a lightweight class for printing to buffers
- Streaming - a method to simplify print statements

https://www.arduino.cc/en/Reference/Libraries

Bibliotheken

Baut ein zeitgesteuertes Anzeige-System:

– Arduino
– Breadboard
– Jumperkabel
– LED(s)
– Servomotor
– Widerstände
– millis()
– Endlicher Zustandsautomat (State-Machine)

Aufgabe 4 – Bis 29.11.2019

1. Plant euren Aufbau in Fritzing oder Tinkercad
2. Zeichnet den Schaltplan per Hand auf
3. Dokumentiert:

– Fotos: Echter Aufbau + Schaltplan
– Screenshots: Fritzing
– Video Demonstration
– Code als *.ino Datei (Arduino Dateiformat)

Nutzt die Arduino Referenz und das Tutorium

Aufgabe 4 – Bis 29.11.2019

Fragen?