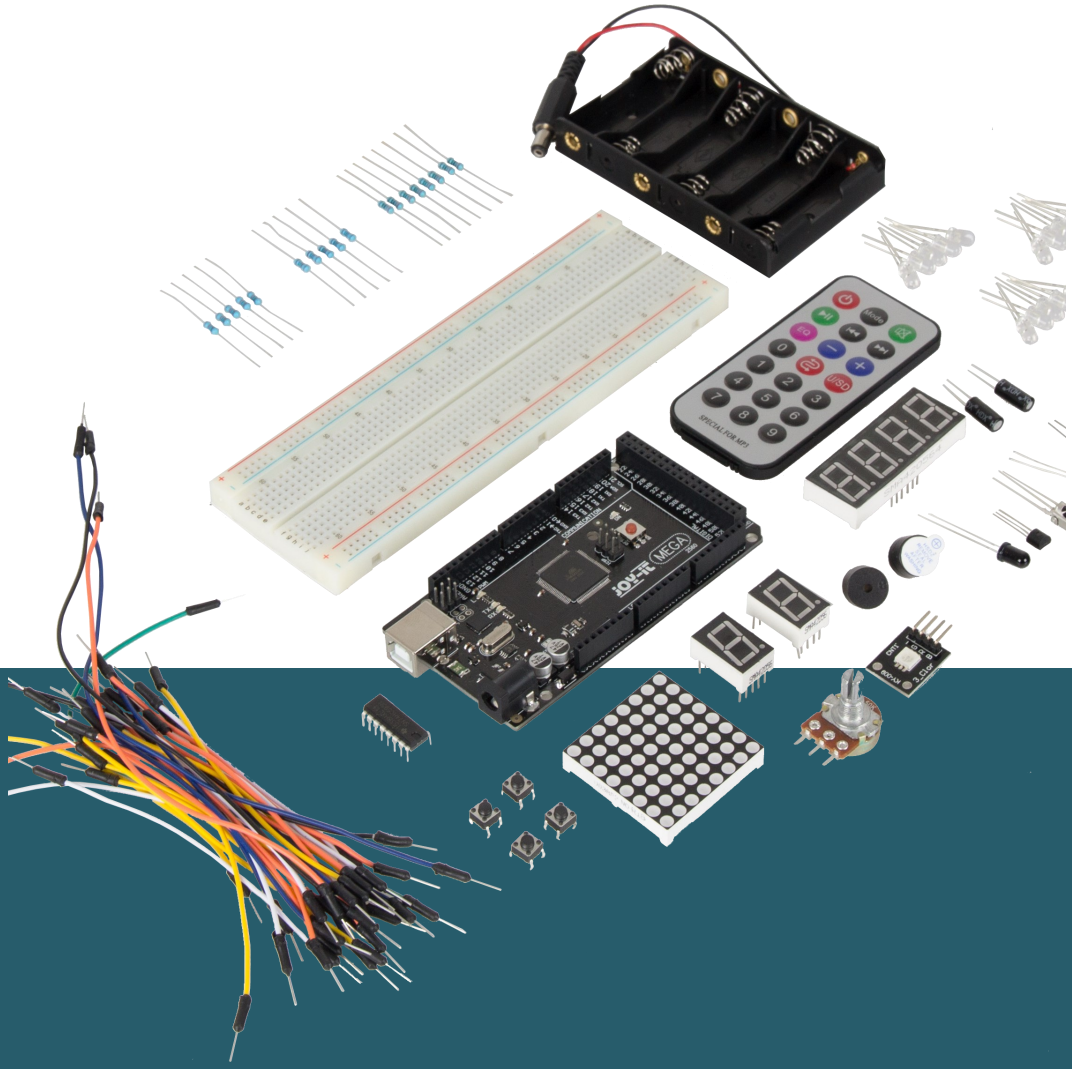


MEGA2560

Mikrocontroller Lernset

1. INHALTSVERZEICHNIS

1. ALLGEMEINE INFORMATIONEN

Sehr geehrter Kunde,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

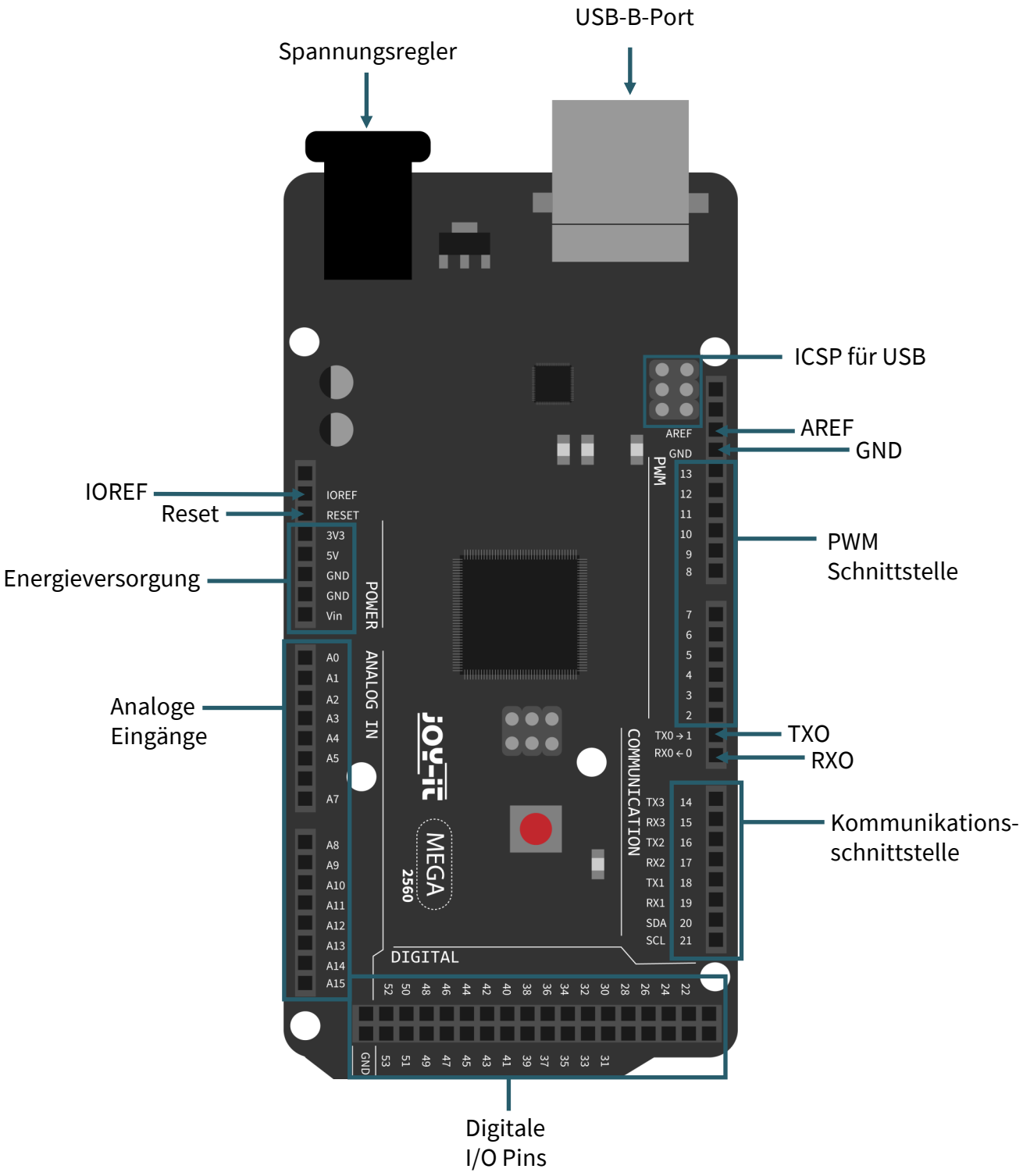
Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

Technische Daten

Unser Board ist ein hochwertiger Nachbau und kompatibel mit dem Arduino Mega 2560. Es handelt sich aber ausdrücklich nicht um einen originalen Arduino.
Das Mega 2560 ist das richtige Mikrocontrollerboard für die, die schnell und unkompliziert in die Programmierwelt einsteigen wollen. Dieses Set führt Sie durch eine Vielzahl von Projekten. Sein ATMega2560-Mikrocontroller bietet Ihnen genügend Leistung für Ihre Ideen und Projekte. Er ist 101,52 x 53,3 mm groß und besitzt mit 54 digitalen Ein- und Ausgängen und 16 analogen Eingängen, viele Verbindungsmöglichkeiten.

Model	ard_Mega2560R3
Mikrocontroller	ATMega2560
Eingangsspannung	7 - 12 V
Maximale Eingangsspannung	6 - 20 V
Digitale IO	54 (14 mit PWM)
Analoge IO	16
DC Strom IO	40 mA
DC Strom 3,3 V	50 mA
Speicher	256 kB (8 kB für Bootloader)
SRAM	8 kB
EEPROM	4 kB
Clock Speed	16 MHz
Abmessungen	101,52 x 53,3 mm

Anschlussbelegung



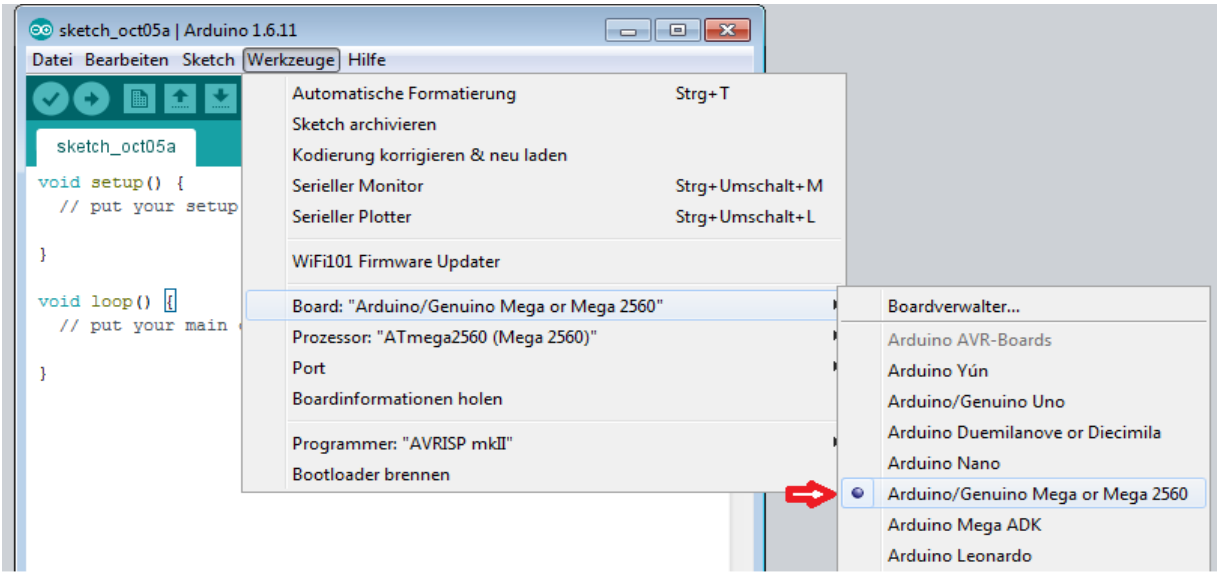
3. SOFTWAREINSTALLATION

Damit man mit der Programmierung des Joy-IT ard_Mega2560R3 beginnen kann, muss vorab auf dem Computer, der für das Programmieren verwendet wird, eine Entwicklungsumgebung, sowie die Treiber für das zugehörige Betriebssystem, installiert werden.

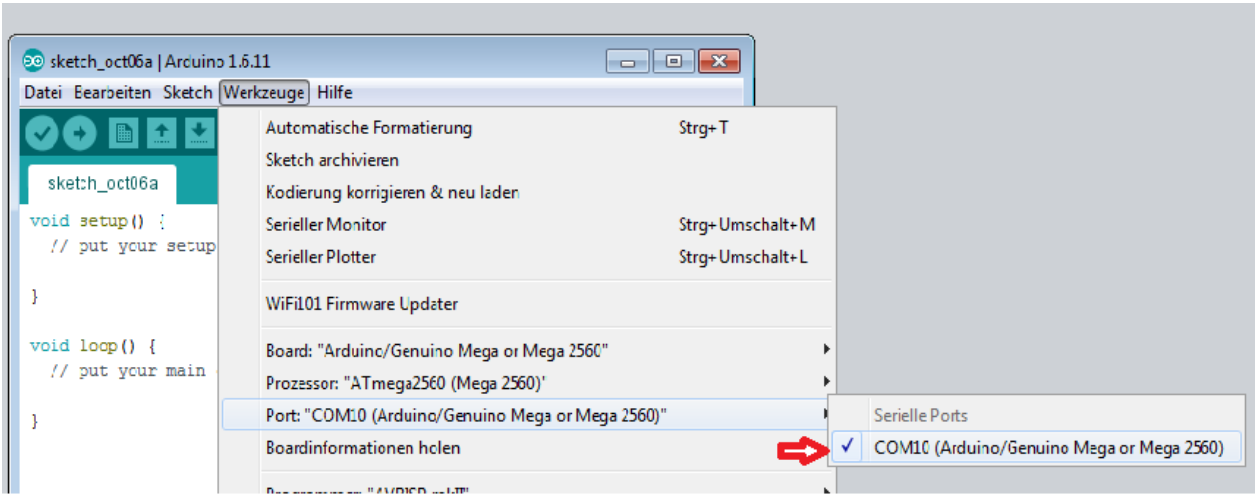
Als Entwicklungsumgebung bietet sich die Arduino DIE an (welche Sie **hier** zum Download finden), die von dem Arduino Hersteller als OpenSource Software unter der GPLv2 veröffentlicht wurde, die sich vom Konzept und Aufbau an Einsteiger richtet.
Diese ist vollständig kompatibel zum Joy-IT ard_Mega2560R3 und beinhaltet neben der Programmierumgebung auch die benötigten Treiber, um direkt loslegen zu können.

Nach der Installation der Software, muss das entsprechende Mikrocontrollerboard in der Programmierumgebung eingerichtet werden. Hierzu befolgen Sie die folgenden zwei Schritte:

1. Unter ***Werkzeuge*** → ***Board*** muss ***Arduino/Genuino Mega or Mega 2560*** ausgewählt sein.



2. Unter ***Werkzeuge*** → ***Port*** wählen Sie dann den Port aus, der mit ***Arduino/Genuino Mega or Mega 2560*** gekennzeichnet ist.



4. WIDERSTÄNDE

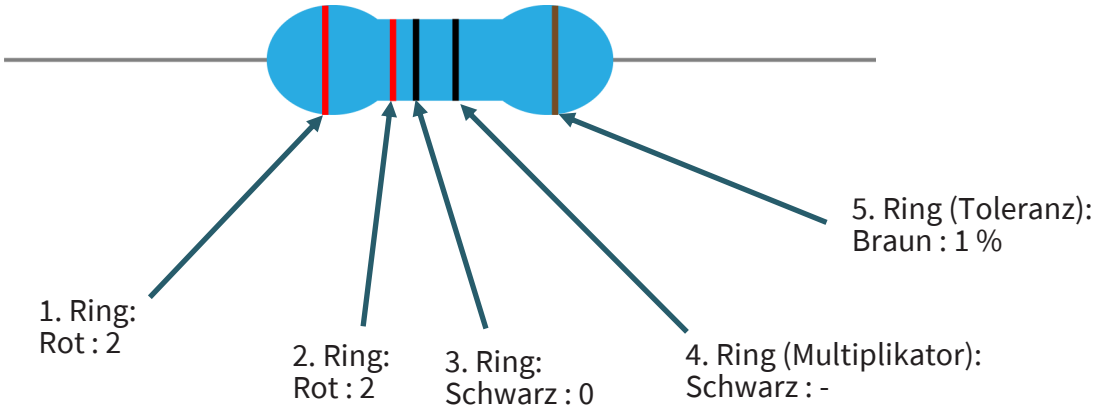
In diesem Set befinden sich drei verschiedene Widerstände 220 Ω, 1 kΩ und 10 kΩ. Auf Widerständen befindet sich eine Farbcodierung mit welcher man den Widerstand berechnen oder erkennen kann, wenn man diesen nicht mittels eines Multimeters nachmessen kann.
In diesem Kapitel lernen Sie wie Sie dies Widerstände berechnen können um die folgenden Lektionen besser durchführen zu können, denn Sie müssen die verschiedenen Widerstände identifizieren können um die Aufbauten korrekt nachzubauen.

Zunächst muss man die Anzahl der Ringe auf den Widerstand erfassen, denn Widerstände können vier oder fünf Ringe besitzen. In diesem Set besitzen die Widerstände 5 Ringe, wodurch der Widerstandswert genauer angegeben ist als bei 4 Ringen.
Nun muss festgestellt werden, welcher Ring der erste ist. Dieser lässt sich dadurch erkennen, dass dieser weiter weg vom Körperende ist.
Man kann nun den Widerstandswert mittels einer Formel berechnen. Bei 5 Ringen dienen die ersten 3 Ringe als Widerstandszähler und der vierte als Multiplikator, was den gesamten Widerstandswert berechnet. Der fünfte Ring ist dabei der Toleranzring, welcher die Abweichung des Widerstandswerts angibt. Bei vier Ringen fällt der dritte Ring weg als Widerstandszähler und dient dann als Multiplikator. Der vierte Ring ist dann als folge der Toleranzring.

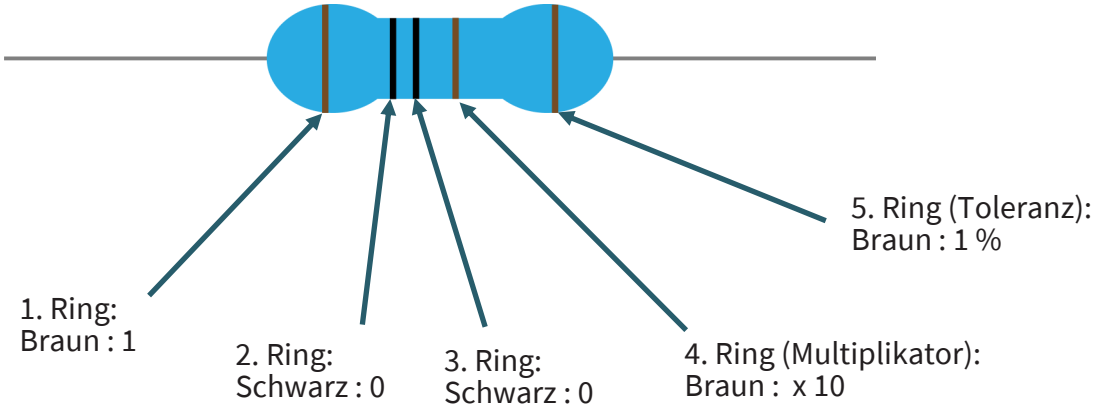
Die Ringe besitzen einen bestimmten Farbcode, wo jede Farbe einen bestimmten Wert besitzt. Im folgenden sehen Sie die Farbtabelle bei 5 Ringen:

Ringfarbe	1. Ring	2.Ring	3. Ring	4. Ring (Multiplikator)	5. Ring (Toleranz)
Schwarz	0	0	0	-	-
Braun	1	1	1	x 10	1 %
Rot	2	2	2	x 100	2%
Orange	3	3	3	x 1.000	-
Gelb	4	4	4	x 10.000	-
Grün	5	5	5	x 100.000	0,5 %
Blau	6	6	6	x 1.000.000	0,25 %
Violett	7	7	7	x 10.000.000	0,1 %
Grau	8	8	8	-	-
Weiß	9	9	9	-	-
Gold	-	-	-	x 0,1	5 %
Silber	-	-	-	x 0,01	10 %

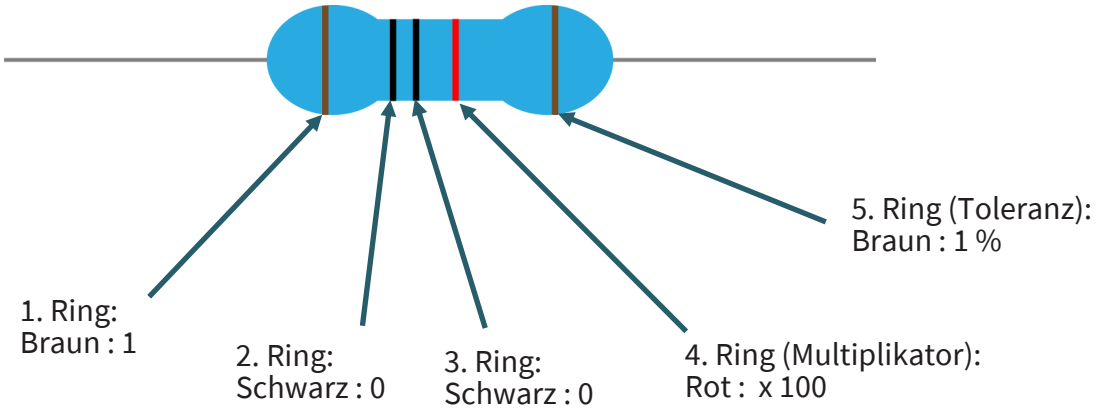
Betrachten wir nun unsere Widerstände lassen sich diese schnell berechnen. Beginnen wir mit dem 220 Ω Widerstand.



Durch die Widerstandszähler ergibt sich die Zahl 220. Dadurch das der Multiplikator keinen Wert besitzt, kann der Widerstand auf 220 Ω bestimmt werden mit einer Toleranz von 1 %, welcher dieser Widerstand abweichen kann.



Die ersten drei Ringe ergeben den ersten Wert von 100, welcher mittels dem Multiplikator (x 10) auf den Widerstandswert von 1000 Ω berechnet wird. Dies kann auch in 1 kΩ umgerechnet werden. Also besitzt dieser Widerstand einen Widerstandswert von 1 kΩ mit einer Toleranz von 1 %.

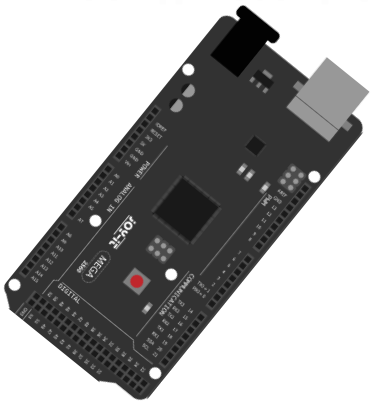


Die ersten drei Ringe ergeben wieder den ersten Wert von 100. Mittels dem Multiplikator von x 100 ergibt sich der Widerstandswert von 10.000 Ω, welcher sich in 10 kΩ umrechnen lässt. Auch dieser Widerstand hat wieder eine Toleranz von 1 %.

5. LEKTIONEN

Lektion 1 : Hallo Welt

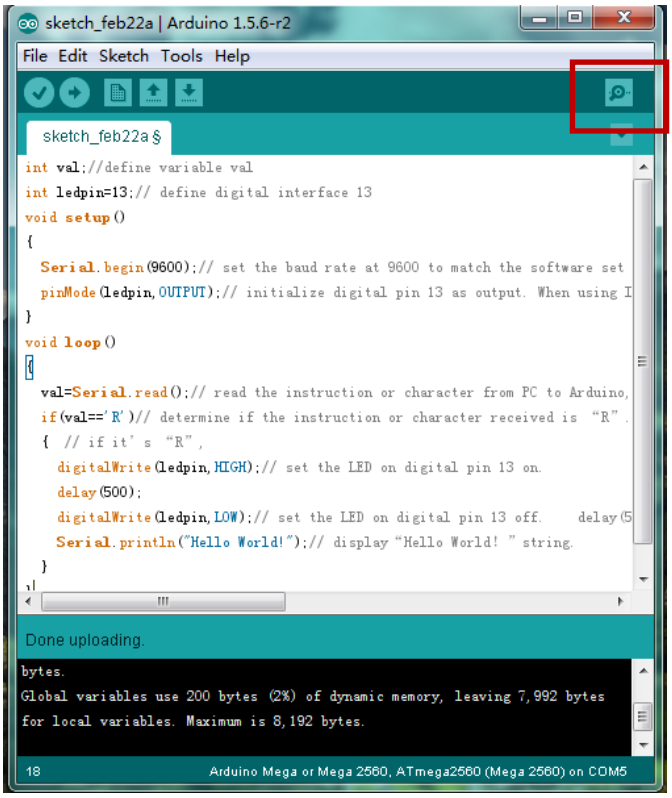
Wir beginnen mit etwas simplem. Für dieses Projekt benötigt man nur die Platine und ein USB Kabel, um die Lektion zu starten. Dies ist ein Kommunikationstest für deinen Mega2560 und deinen PC, sowie ein grundlegendes Projekt für Ihren ersten Versuch in der Arduino Welt!

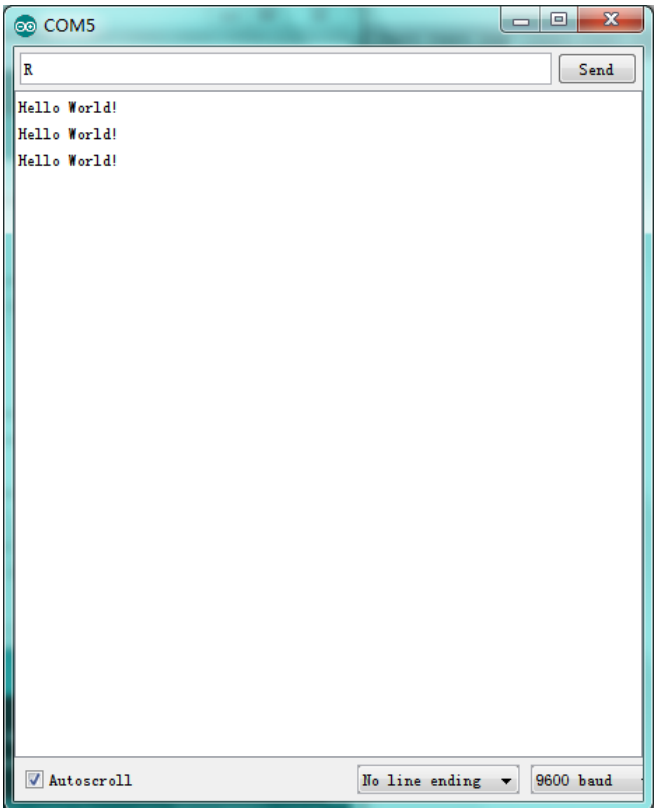


Nachdem die Installation der Treiber abgeschlossen ist, lass uns die Arduino Software öffnen und einen Code verfassen, welcher es dem Mega2560 ermöglicht *Hallo Welt!* unter deiner Anweisung anzuzeigen. Natürlich kannst du auch einen Code verfassen, welcher den Mega2560 ohne Anweisung wiederholt *Hallo Welt!* ausgeben lässt. Ein simple if()-Anweisung wird dies übernehmen. Wir können die LED an Pin 13 anweisen, erst zu blinken und anschließend *Hallo Welt!* anzuzeigen, nachdem der Arduino den Befehl dazu bekommt.

```
int val; // Definiert die variable “Val”
int ledpin=13; // Definiert digitales Interface 13

void setup() {
    Serial.begin(9600);
    // Setzt die Baudrate auf 9600 um der Konfiguration der Software zu
    //entsprechen. Wenn es mit einem bestimmten Gerät verbunden ist,
    //muss die Baudrate übereinstimmen.
    pinMode(ledpin,OUTPUT);
    // Bestimmt den Digital Pin 13 als Ausgang. Wenn I/O Ports an
    // einem Arduino verwendet werden, wird diese Konfiguration immer
    // benötigt.
}

void loop() {
    val=Serial.read();
    // Liest die Anleitung oder Char-Symbole
    // vom PC zum Arduino und ordne sie “Val” zu.
    if(val=='R') {
    // Bestimmt ob die Anweisung oder der erhaltene Buchstabe „R“ ist.
    // Falls es „R” ist,
        digitalWrite(ledpin,HIGH);
        // Schaltet die LED am Digital Pin 13 ein.
        delay(500);
        digitalWrite(ledpin,LOW);
        // Schaltet die LED am Digital Pin 13 aus.
        delay(500);
        Serial.println("Hello World!"); // Zeigt „Hallo Welt!” an.
    }
}
```

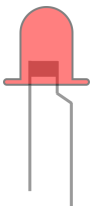
Klicken Sie auf den seriellen Monitor und füge ***R*** , dann wird die LED auf dem Board aufleuchten und Ihr PC wird die Information *Hallo Welt!* vom Arduino erhalten.



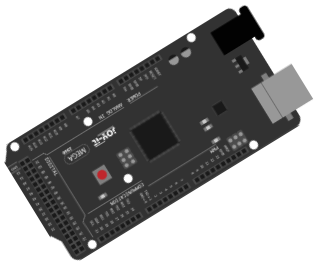


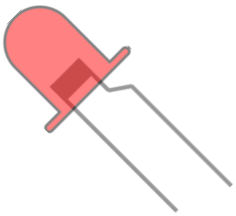
Lektion 2: Blinkende LED

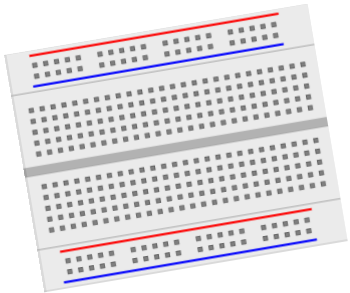
In dem Hallo Welt! Programm sind wir der LED bereits begegnet. Diesmal werden wir eine LED mit einem der digitalen Pins verbinden. Es werden die folgenden Teile benötigt:

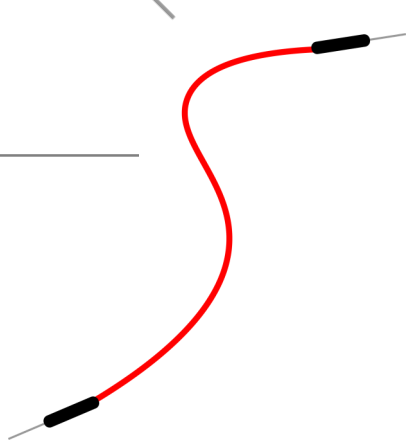


1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	220 Ω Widerstand
1x	Breadboard
2x	Überbrückungskabel

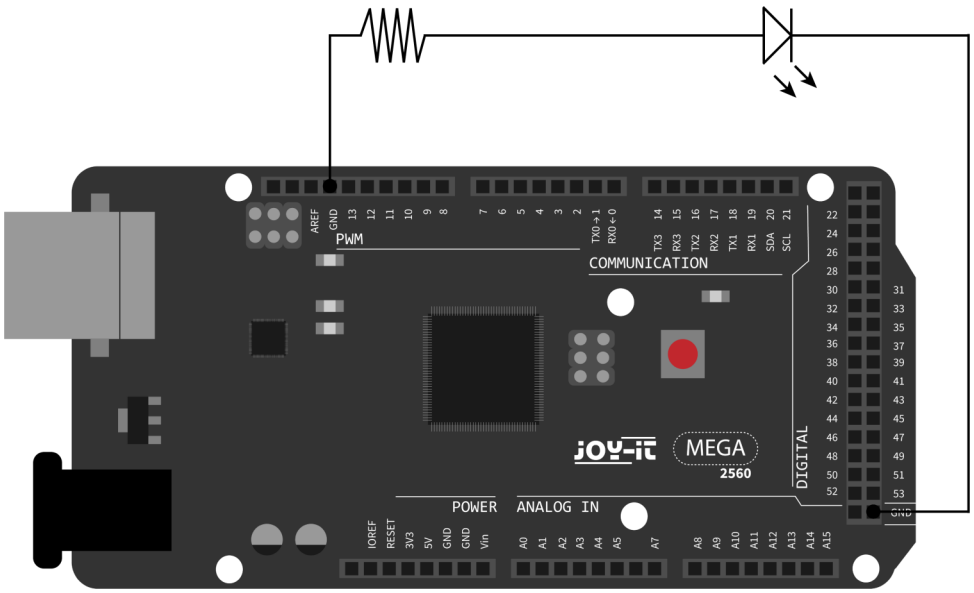


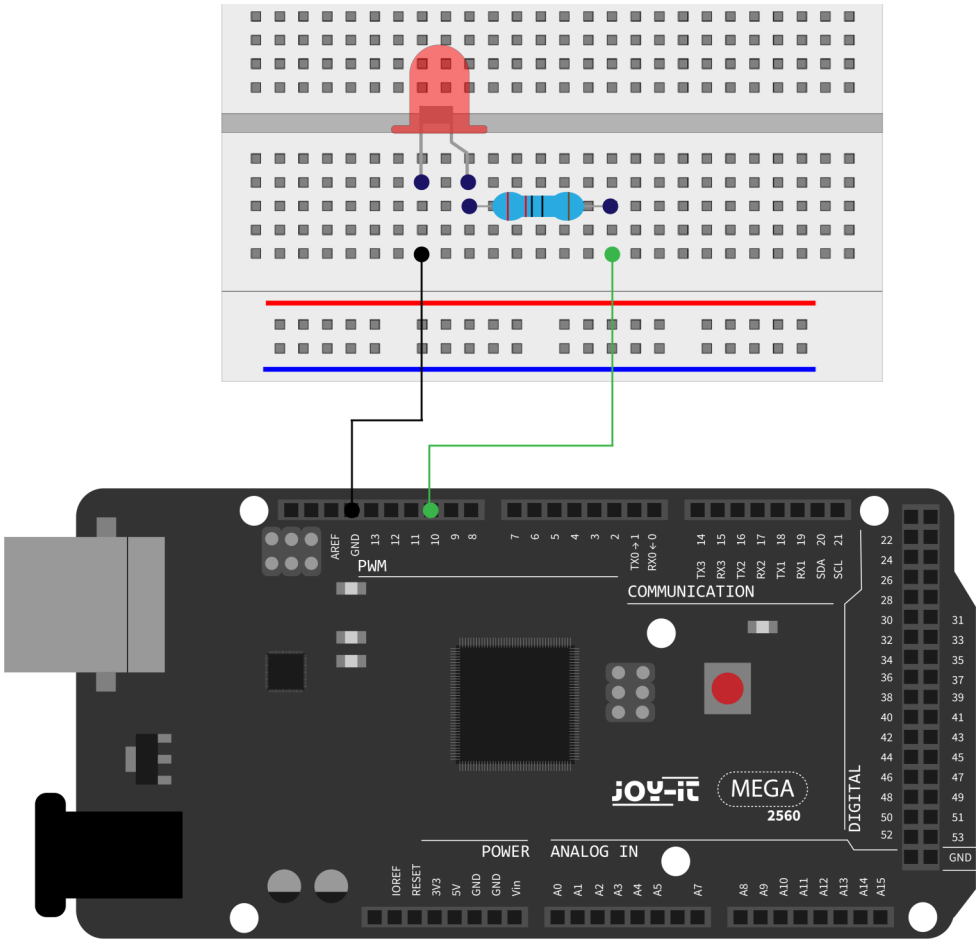






Wir folgen dem folgendem Schaltplan. Hier benutzen wir den digitalen Pin 10. Wir verbinden die LED mit einem 220 Ω Widerstand um Beschädigungen durch zu hohem Strom zu vermeiden.

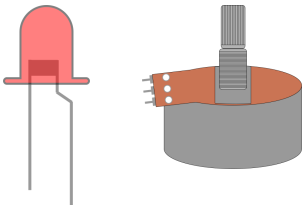




```
int ledPin = 10;      // Definiert Digital Pin 10.

void setup() {
    pinMode(ledPin, OUTPUT);
    // Definiert Pin mit verbundener LED als Ausgang.
}

void loop() {
    digitalWrite(ledPin, HIGH); // Schaltet die LED ein.
    delay(1000); // Wartet eine Sekunde.
    digitalWrite(ledPin, LOW); // Schaltet die LED aus.
    delay(1000); // Wartet eine Sekunde
}
```

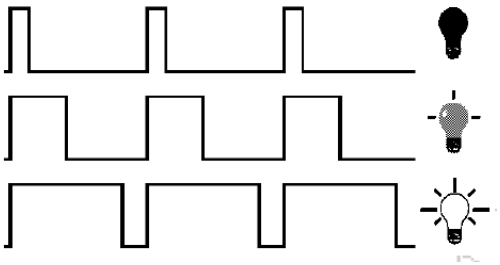
Nachdem Hochladen des Programmes werden Sie die LED, welche mit Pin 10 verbunden ist, im 1-Sekundentakt aufleuchten sehen.

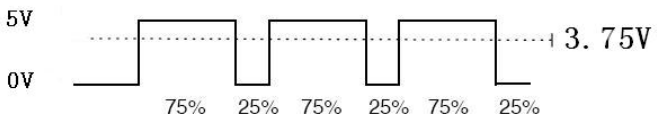
Lektion 3: PWM Lichtkontrolle

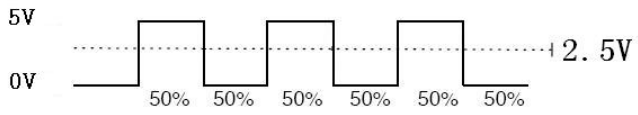


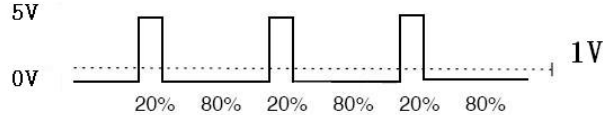
PWM (kurz für Pulse Width Modulation) ist eine Technik, welche benutzt wird um analoge Signalpegel in digitale zu kodieren. Ein Computer ist nicht dazu in der Lage analoge Spannung auszugeben. Er kann nur digitale Spannung mit den Werten wie 0 V oder 5 V ausgeben. Daher wird ein hochauflösender Zähler benutzt, um einen spezifischen analogen Signalpegel zu kodieren, indem man den Auslastungsgrad von PWM moduliert. Das PWM Signal ist ebenfalls digitalisiert, da zu jedem Zeitpunkt die Energieversorgung entweder 5 V (an) oder 0 V (aus) ist.

Die Spannung oder der Strom wird der analogen Last (das Gerät, dass die Energie verbraucht) durch wiederholte Impulsfolgen zugeführt, indem permanent zwischen dem ein- und ausgeschalteten Zustand gewechselt wird. Der Wertz der Ausgangsspannung wird anhand der An- und Aus-Zustände ermittelt.





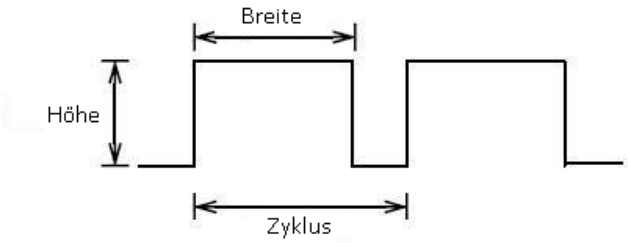




$$Spannung = \frac{AN - Zeit}{Impulszeit} * maximaler\ Spannungswert$$

Es gibt viele Anwendungsfälle für PWM: Regulierung der Lampenhelligkeit, Regulierung der Motorengeschwindigkeit usw..

Das sind die drei Grundparameter von PWM:

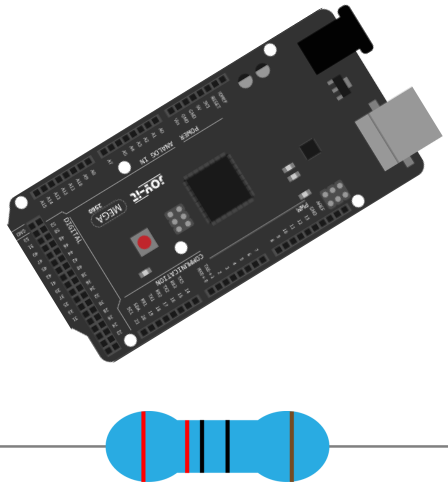


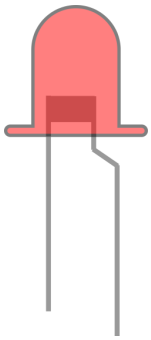
1. Die Amplitude der Impulsbreite (Minimum / Maximum)
2. Die Pulsperiode (Die gegenseitige Pulsfrequenz in einer Sekunde)
3. Der Spannungspegel (wie: 0 - 5 V)

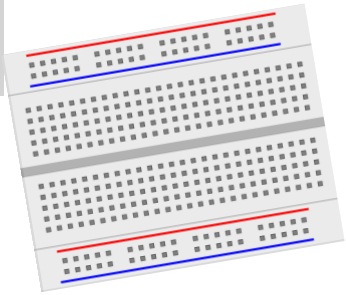
Es gibt 6 PWM Schnittstellen auf dem Mega2560: digitaler Pin 3, 5, 6, 9, 10 und 11.

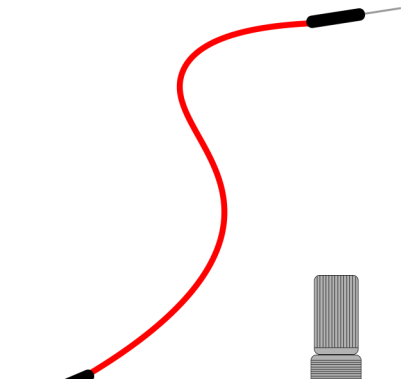
In vorherigen Experiment haben wir die tastenkontrollierte LED kennen gelernt, bei der wir ein digitales Signal verwendet haben, um einen digitalen Pin zu kontrollieren.
Nun werden wir einen Potentiometer verwenden um die Helligkeit der LED kontrollieren zu können. Wir benötigen hierfür:

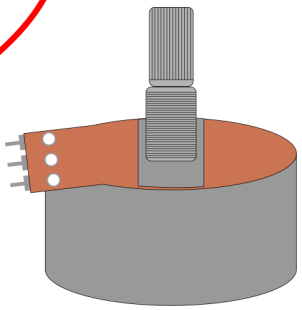
1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	220 Ω Widerstand
1x	Breadboard
6x	Überbrückungskabel
1x	Potentiometer



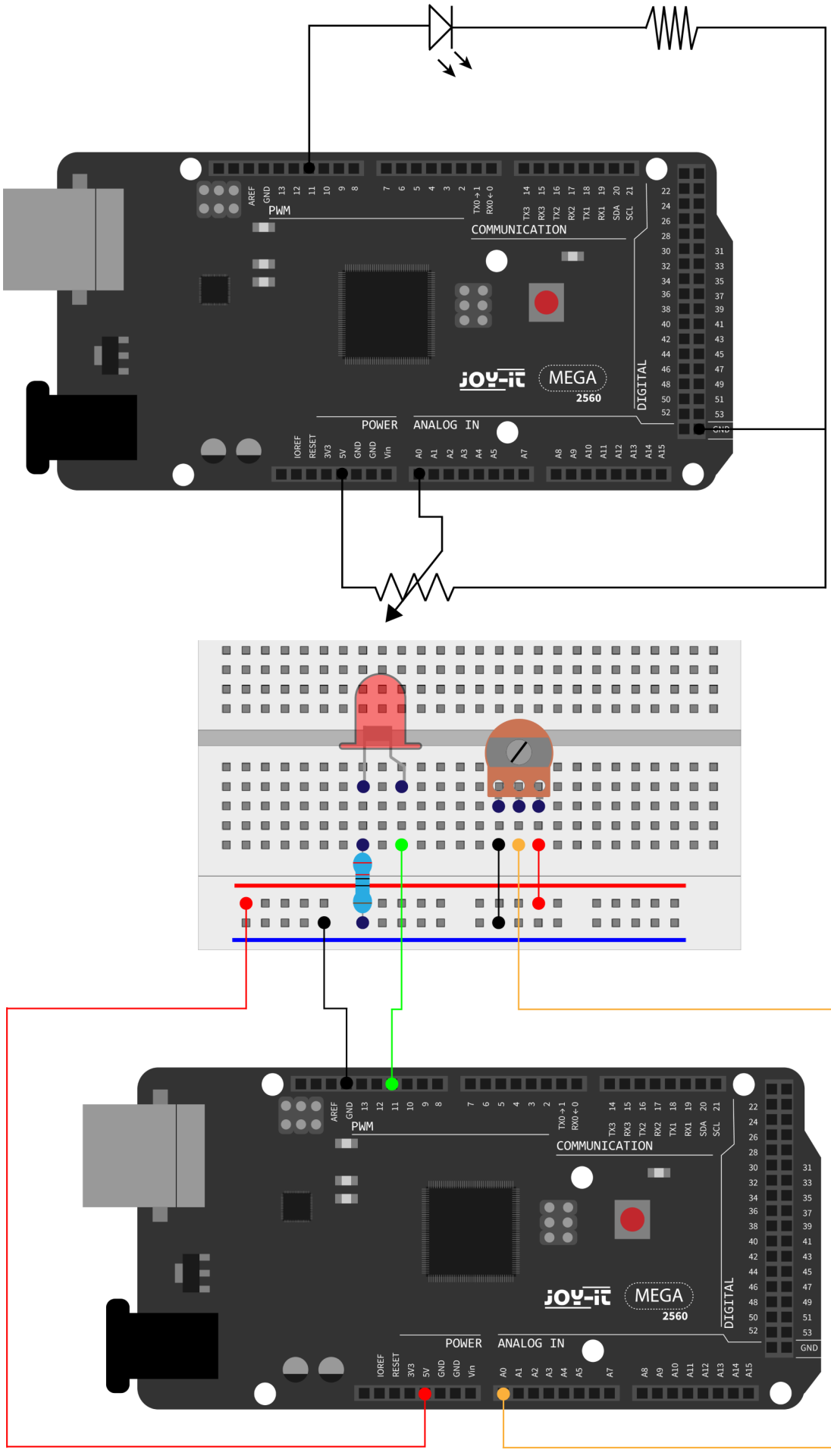








Der Eingang des Potentiometers ist analog, also verbinden wir ihn mit dem analogen Port. Die LED verbinden wir mit dem PWM-Port. Ein anderes PWM-Signal kann die Helligkeit der LED regulieren.

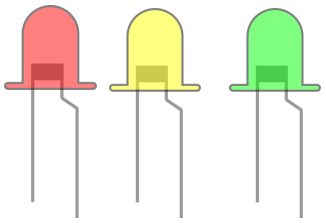


In diesem Experiment werden wir den Analogwert des Potentiometers lesen und den wert dem PWM-Port zuordnen, sodass eine entsprechende Änderung der Helligkeit der LED beobachtet werden kann.
Außerdem werden wir den Analogwert auf dem Bildschirm anzeigen.

```
int potpin = 0; // Initialisiert analogen Pin 0
int ledpin = 11; // Initialisiert den digitalen Pin 11 (PWM Ausgang)
int val = 0; // Speichert den Wert der Variable vom Sensor

void setup() {
    pinMode(ledpin,OUTPUT);
    // Definiert digitalen Pin 11 als „Ausgang“
    Serial.begin(9600); // Setzt Baudrate auf 9600
}

void loop() {
    val = analogRead(potpin);
    // Liest den Analogwert vom Sensor und weist diesen „Val“ zu
    Serial.println(val); // Zeigt den Wert von „Val“
    analogWrite(ledpin,val/4);
    // Schaltet die LED ein und stellt die Helligkeit ein (Maximalwert: 255)
    delay(10); // Wartet 0,01 Sekunden
}
```

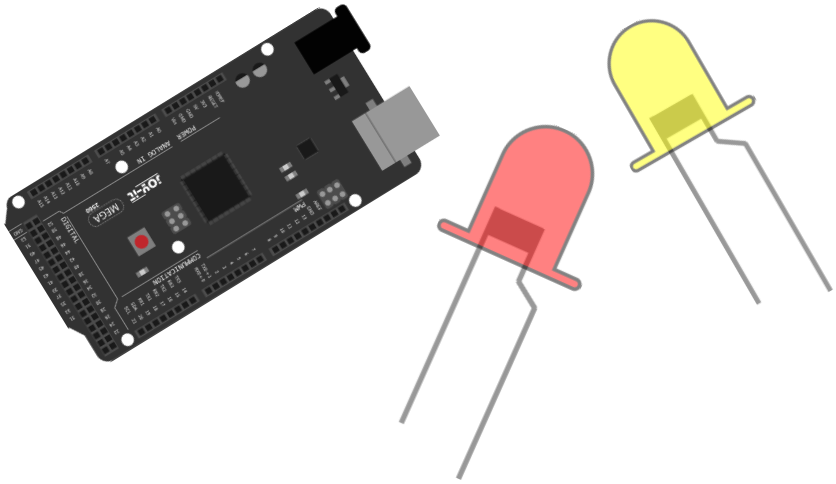
Nach der Übertragung des Programms und bei Bewegung des Potentiometers, können wir Veränderungen bei den angezeigten Werten beobachten. Außerdem können wir eine offensichtliche Änderung der LED-Helligkeit erkennen.

Lektion 4: Ampellichter

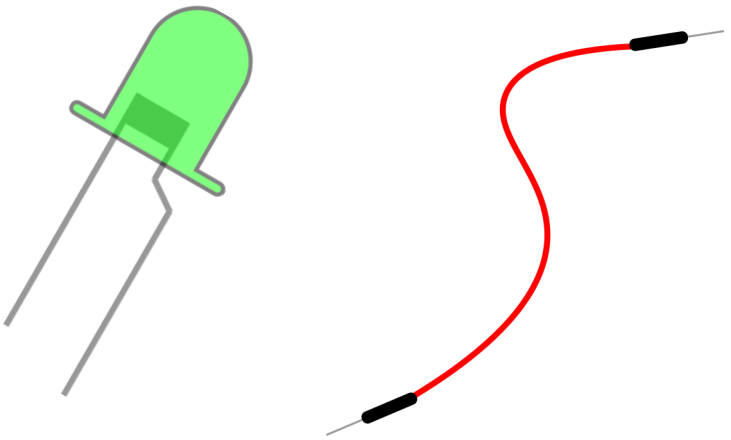


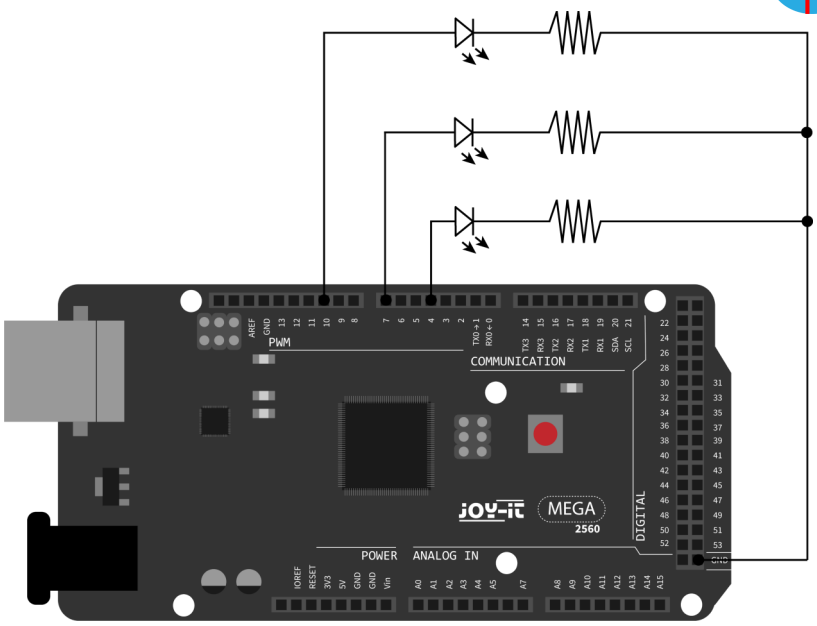
Im vorherigen Programm haben wir das blinkende LED Experiment mit nur einer LED durchgeführt. Jetzt wird es Zeit, ein etwas komplizierteres Experiment durchzuführen: Ampellichter.
Eigentlich sind diese beiden Experimente sich sehr ähnlich. Während dieses Experiments werden wir 3 LEDs mit verschiedenen Farben nutzen, während im letzten nur eine LED zum Einsatz kam. Wir benötigen hierfür:

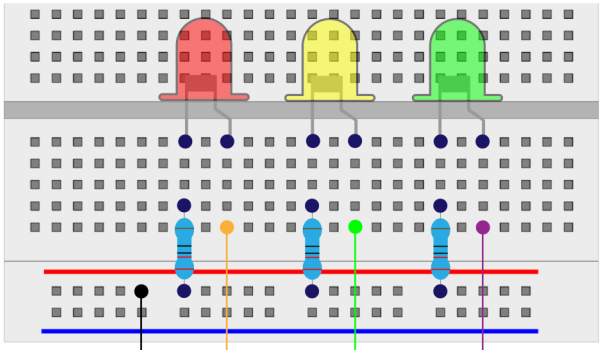
1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	Gelbe M5 LED

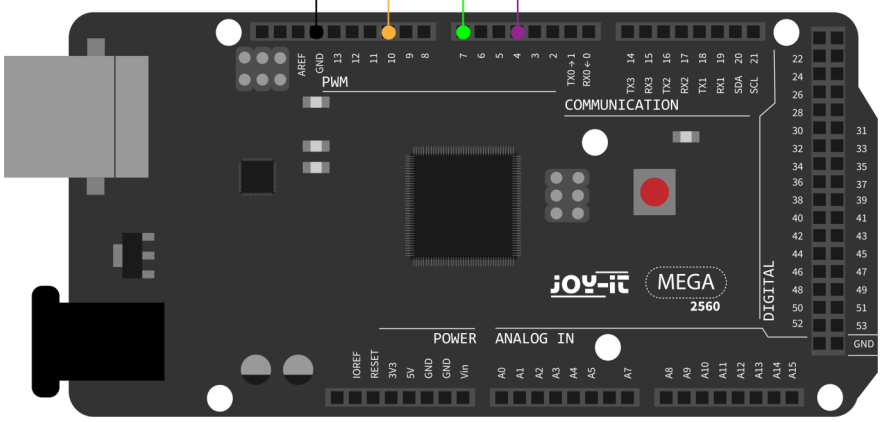


1x	Grüne M5 LED
3x	220 Ω Widerstand
1x	Breadboard
4x	Überbrückungskabel







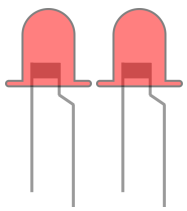


Da dies eine Situation von Ampellichtern ist, sollte die Leuchtzeit jeder einzelnen LED genau so lange, wie bei einer echten Ampel sein. In diesem Programm werden wir die Arduino-Verzögerungsfunktion nutzen, um die Verzögerungszeit zu kontrollieren.

```
int redled =10; // Initialisiert digitalen Pin 8
int yellowled =7; // Initialisiert digitalen Pin 7
int greenled =4; // Initialisiert digitalen Pin 4

void setup() {
    pinMode(redled, OUTPUT); // Setzt den Pin mit der roten LED als „Ausgang“
    pinMode(yellowled, OUTPUT);// Setzt den Pin mit der gelben LED als „Ausgang“
    pinMode(greenled, OUTPUT);// Setzt den Pin mit der grünen LED als „Ausgang“
}

void loop() {
    digitalWrite(greenled, HIGH); // Schaltet die grüne LED ein
    delay(5000); // Wartet 5 Sekunden
    digitalWrite(greenled, LOW); // Schaltet die grüne LED aus
    for(int i=0;i<3;i++) // Blinkt 3x
        {
        delay(500); // Wartet 5 Sekunden
        digitalWrite(yellowled, HIGH); // Schaltet die gelbe LED ein
        delay(500); // Wartet 5 Sekunden
        digitalWrite(yellowled, LOW); // Schaltet die gelbe LED aus
    }
    delay(500); // Wartet 5 Sekunden
    digitalWrite(redled, HIGH); // Schaltet die rote LED ein
    delay(5000); // Wartet 5 Sekunden
    digitalWrite(redled, LOW); // Schaltet die rote LED aus
}
```

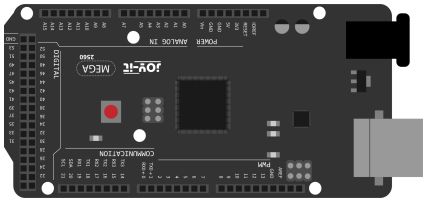
Wenn die Datei hochgeladen wurde, sind die Ampellichter sichtbar. Das grüne Licht wird für 5 Sekunden leuchten und sich dann abschalten. Danach wird das gelbe Licht 3 Mal blinken und anschließend wird das rote Licht für 5 Sekunden leuchten, sodass ein Kreislauf gebildet wird.

Lektion 5: LED Jagd-Effekt

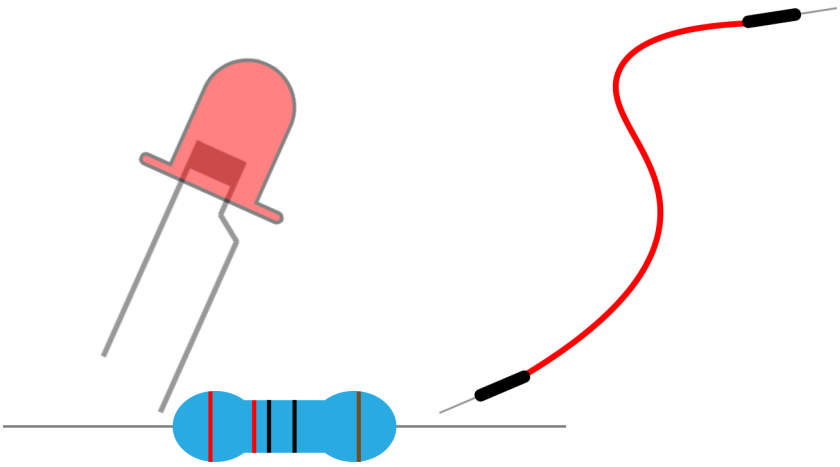


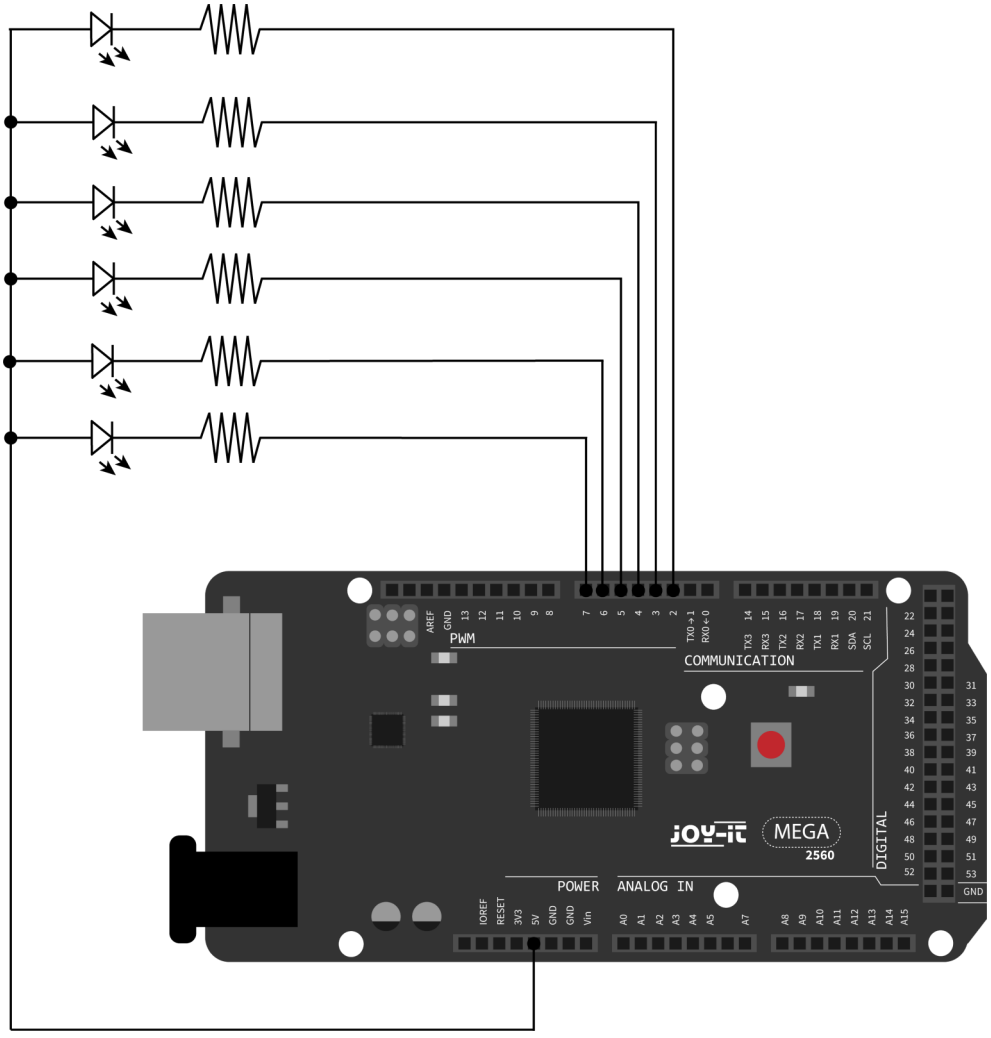
Wir sehen oft Reklametafeln, welche versehen sind mit bunten LEDs. Diese ändern sich ständig um verschiedene Effekte zu formen. In diesem Experiment wird ein Programm erstellt, welches den LED Jagd-Effekt simuliert. Dazu wird benötigt:

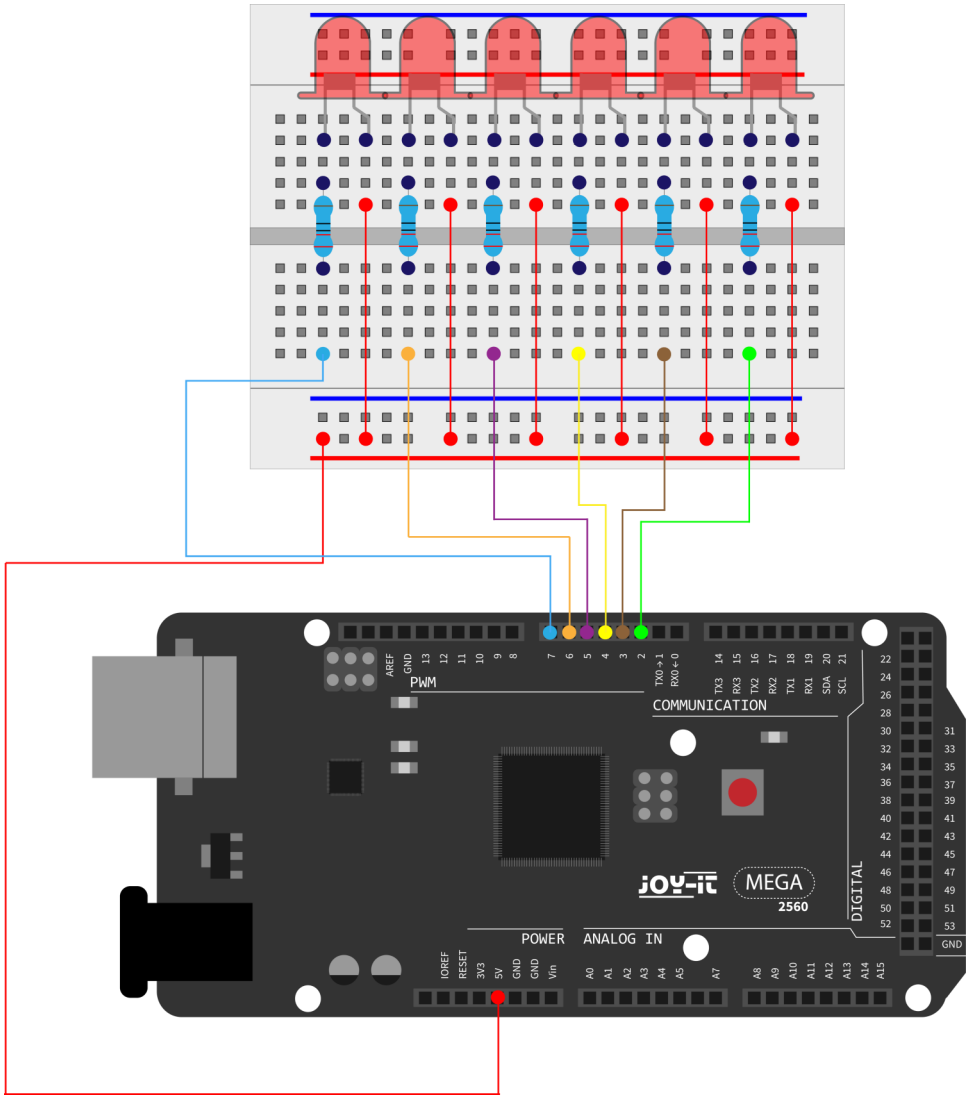
1x	Mega2560 Platine
1x	USB-Kabel



6x	M5 LED
6x	220 Ω Widerstand
1x	Breadboard
13x	Überbrückungskabel

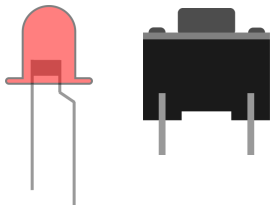






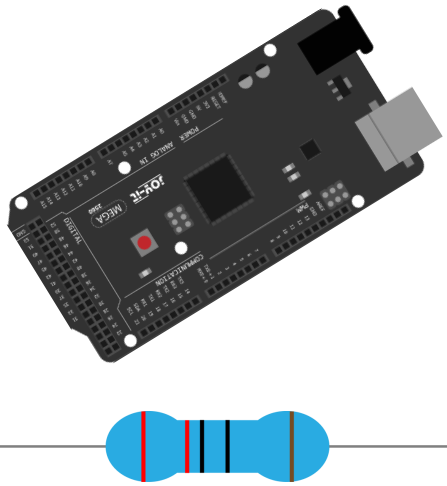

```
int BASE = 2 ; // Der I/O Pin für die erste LED #
int NUM = 6; // Anzahl der LEDs

void setup() {
    for (int i = BASE; i < (BASE + NUM); i ++) {
        pinMode(i, OUTPUT); // Setzt I/O Pinne als Ausgang
    }
}

void loop() {
    for (int i = BASE; i < (BASE + NUM); i ++){
        digitalWrite(i, LOW); // Setzt I/O Pinne auf „niedrig“
        // schaltet die LEDs nacheinander ein
        delay(200); // Verzögerung
    }
    for (int i = BASE; i < (BASE + NUM); i ++) {
        digitalWrite(i, HIGH); // Setzt I/O Pinne auf „hoch“,
        // schaltet die LEDs nacheinander ab
        delay(200); // Verzögerung
    }
}
```

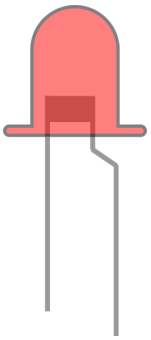
Lektion 6: Tastengesteuerte LED

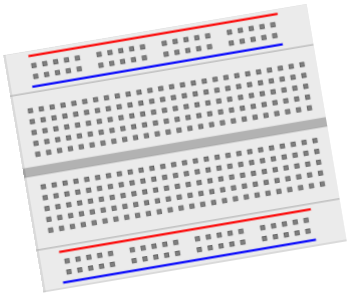


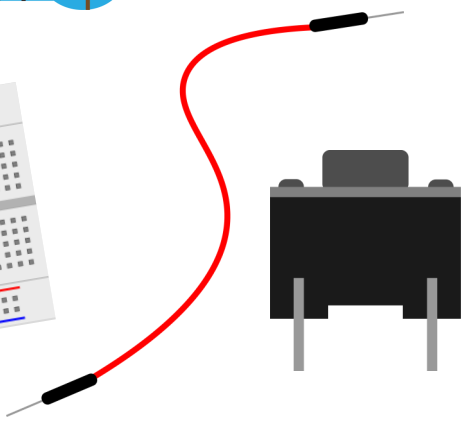
I/O-Port ist eine Schnittstelle, welche als Eingang und als Ausgang fungiert. Bis jetzt haben wir nur den Ausgang benutzt. In diesem Experiment werden wir versuchen den Eingang zu benutzen, um den Ausgangswert des angeschlossenen Gerätes auslesen zu können. Wir werden eine Taste und eine LED als Eingang und als Ausgang verwenden, um ein besseres Verständnis über die I/O-Funktion zu bekommen.
Tastenschalter, den vielen wahrscheinlich bekannt ist, haben einen Schaltwert (digitalen Wert). Wenn der Schalter gedrückt wird schließt sich der Schaltkreis und befindet sich in einem leitenden Status.
Dazu benötigen wir:

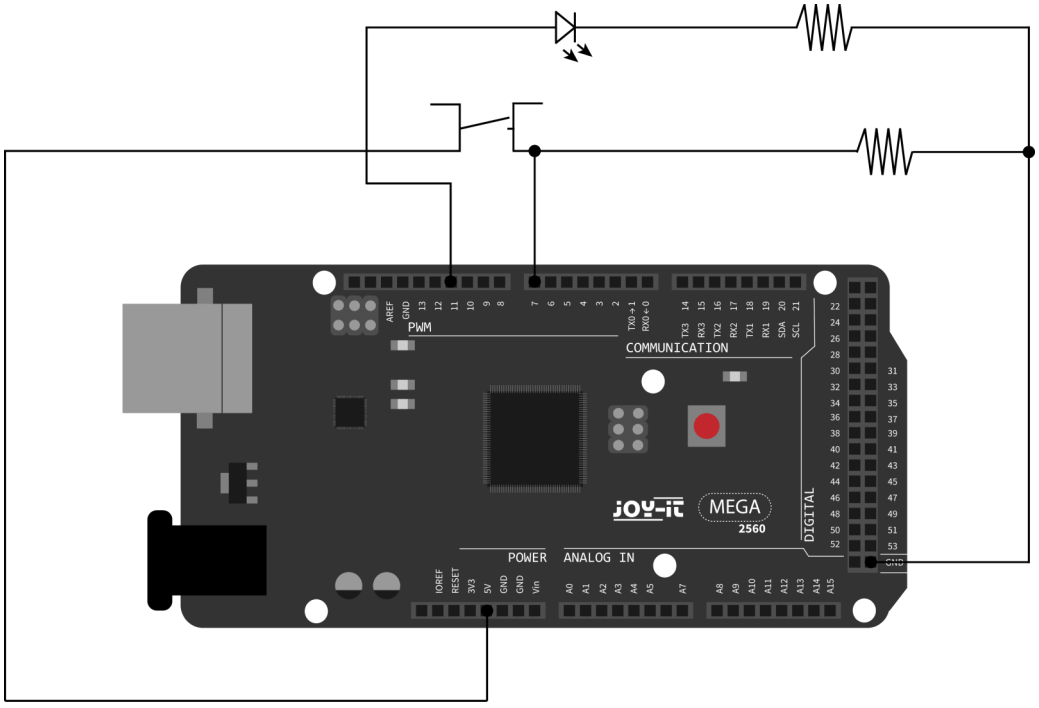
1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	220 Ω Widerstand
1x	10 kΩ Widerstand
1x	Tastenschalter
1x	Breadboard
6x	Überbrückungskabel

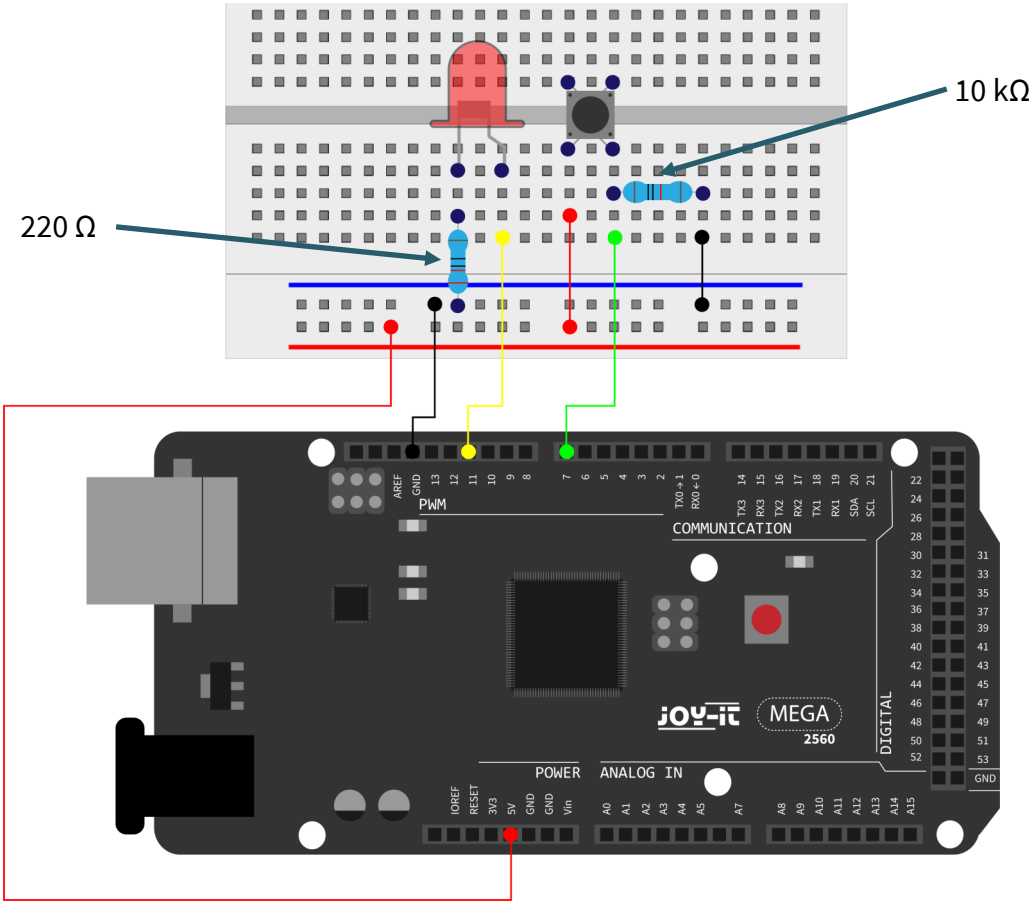












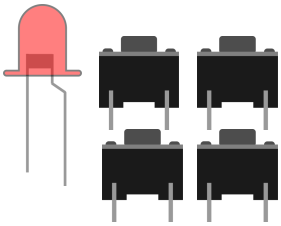

```
int ledpin=11; // Initialisiert Pin 11
int inpin=7; // Initialisiert Pin 7
int val; // Definiert „Val”

void setup() {
    pinMode(ledpin,OUTPUT); // Setzt LED Pin als „Ausgang“
    pinMode(inpin,INPUT); // Setzt Tastenpin als „Eingang“
}

void loop() {
    val=digitalRead(inpin);
    // Liest den Pegelwert von Pin 7 und ordnet diesen „Val“ zu
    if(val==LOW)
    // Prüft ob die Taste gedrückt wird, falls ja wird die LED eingeschalte
    {
        digitalWrite(ledpin,LOW);
    }
    else {
        digitalWrite(ledpin,HIGH);
    }
}
```

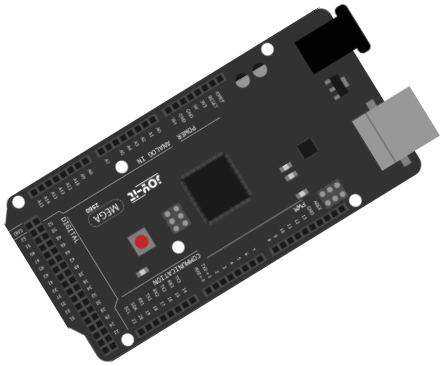
Wenn die Taste gedrückt wird, leuchtet die LED, ansonsten bleibt diese aus. Das simple Prinzip dieses Experimentes wird oft in einer Vielfalt von Schaltkreisen und Elektrogeräten verwendet.

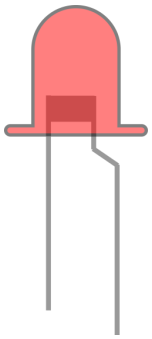
Lektion 7: Responder Experiment

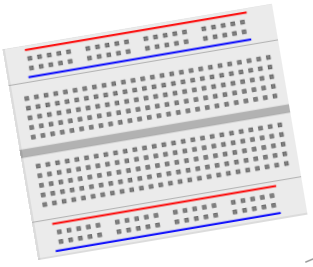
In dieser Lektion gibt es drei Tastenschalter und eine Reset-Taste, welche mit Hilfe von 7 digitalen I/O Pins die 3 entsprechenden LEDs steuern. Hierzu benötigen Sie:

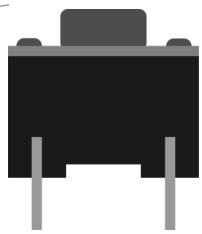
1x	Mega2560 Platine
1x	USB-Kabel
7x	220 Ω Widerstand
1x	Rote M5 LED
1x	Gelbe M5 LED
1x	Grüne M5 LED
4x	Tastenschalter
1x	Breadboard
13x	Überbrückungskabel

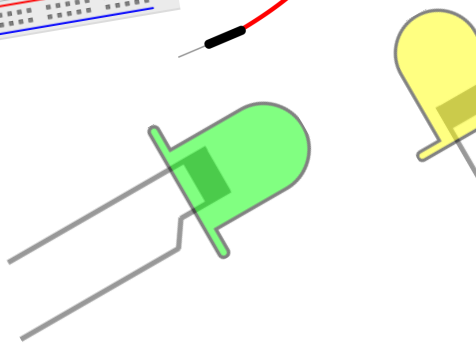


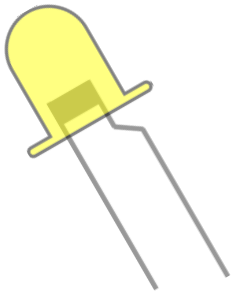


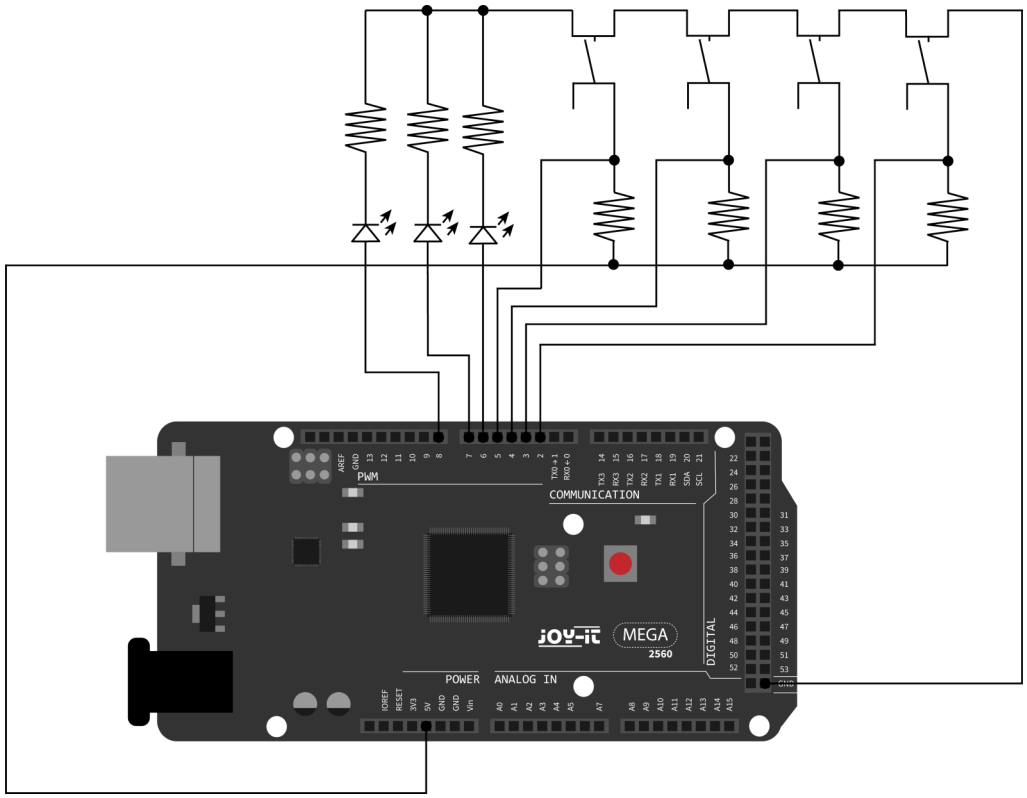


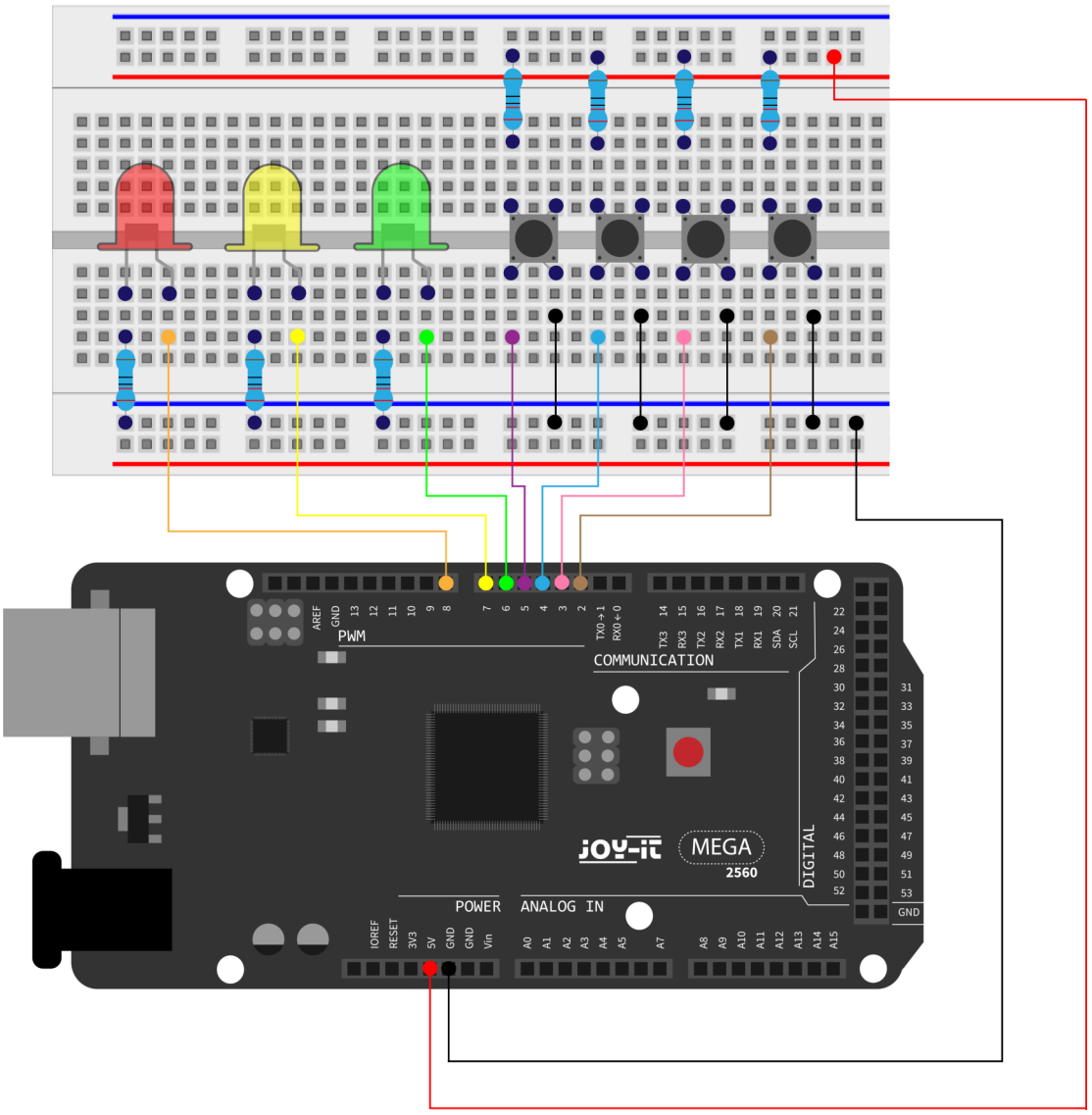












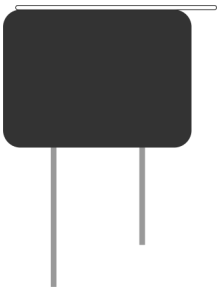

```
int redled=8; // Pin für rote LED
int yellowled=7; // Pin für gelbe LED
int greenled=6; // Pin für grüne LED
int redpin=5; // Pin für rote Taste
int yellowpin=4; // Pin für gelbe Taste
int greenpin=3; // Pin für grüne Taste
int restpin=2; // Pin für Reset-Taste
int red;
int yellow;
int green;

void setup() {
    pinMode(redled,OUTPUT);
    pinMode(yellowled,OUTPUT);
    pinMode(greenled,OUTPUT);
    pinMode(redpin,INPUT);
    pinMode(yellowpin,INPUT);
    pinMode(greenpin,INPUT);
}
```

```
void loop(){ //Liest wiederholt die Pinne der Tasten
    red = digitalRead(redpin);
    yellow = digitalRead(yellowpin);
    green = digitalRead(greenpin);
    if(red==LOW)RED_YES();
    if(yellow==LOW)YELLOW_YES();
    if(green==LOW)GREEN_YES();
}

void RED_YES(){// Führt den Code aus bis rote LED an ist
// endet den Kreislauf wenn die Reset-Taste betaetigt wird
    while(digitalRead(restpin)==1){
        digitalWrite(redled,HIGH);
        digitalWrite(greenled,LOW);
        digitalWrite(yellowled,LOW);
    }
    clear_led();
}

void YELLOW_YES(){// Fuehrt den Code aus bis gelbe LED an ist
// endet den Kreislauf, wenn die Reset-Taste betaetigt wird
    while(digitalRead(restpin)==1){
        digitalWrite(redled,LOW);
        digitalWrite(greenled,LOW);
        digitalWrite(yellowled,HIGH);
    }
    clear_led();
}

void GREEN_YES() // Fuehrt den Code aus bis grüne LED an ist
// endet den Kreislauf wenn die Reset-Taste betaetigt wird
{
    while(digitalRead(restpin)==1){
        digitalWrite(redled,LOW);
        digitalWrite(greenled,HIGH);
        digitalWrite(yellowled,LOW);
    }
    clear_led();
}

void clear_led(){ // Alle LEDs aus
digitalWrite(redled,LOW);
digitalWrite(greenled,LOW);
digitalWrite(yellowled,LOW);
}
```

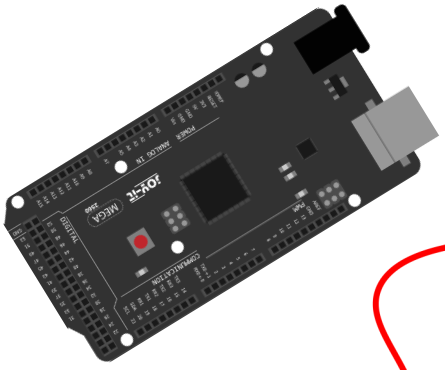
Achten Sie darauf, dass Sie beide Codeteile in Ihrem Sketch der Arduino IDE hinzufügen. Wenn eine Taste betätigt wird, schaltet sich die entsprechende LED ein. Wir die Reset-Taste betätigt, schaltet sich die entsprechende LED wieder aus.

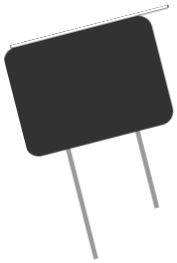
Lektion 8: Aktiver Summer

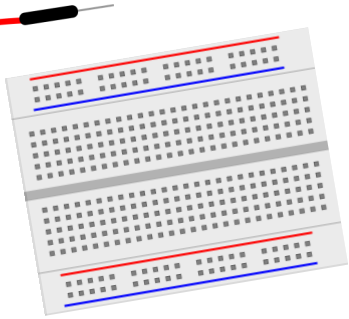
Aktive Summer werden in Computern, Druckern, Weckern, Elektrospielzeug etc. als ein Geräusch emittierendes Element eingesetzt. Es hat eine innere Vibrationsquelle. Mit einer 5V-Energieversorgung verbunden, kann es wiederholt summen.
Hierzu wird benötigt:

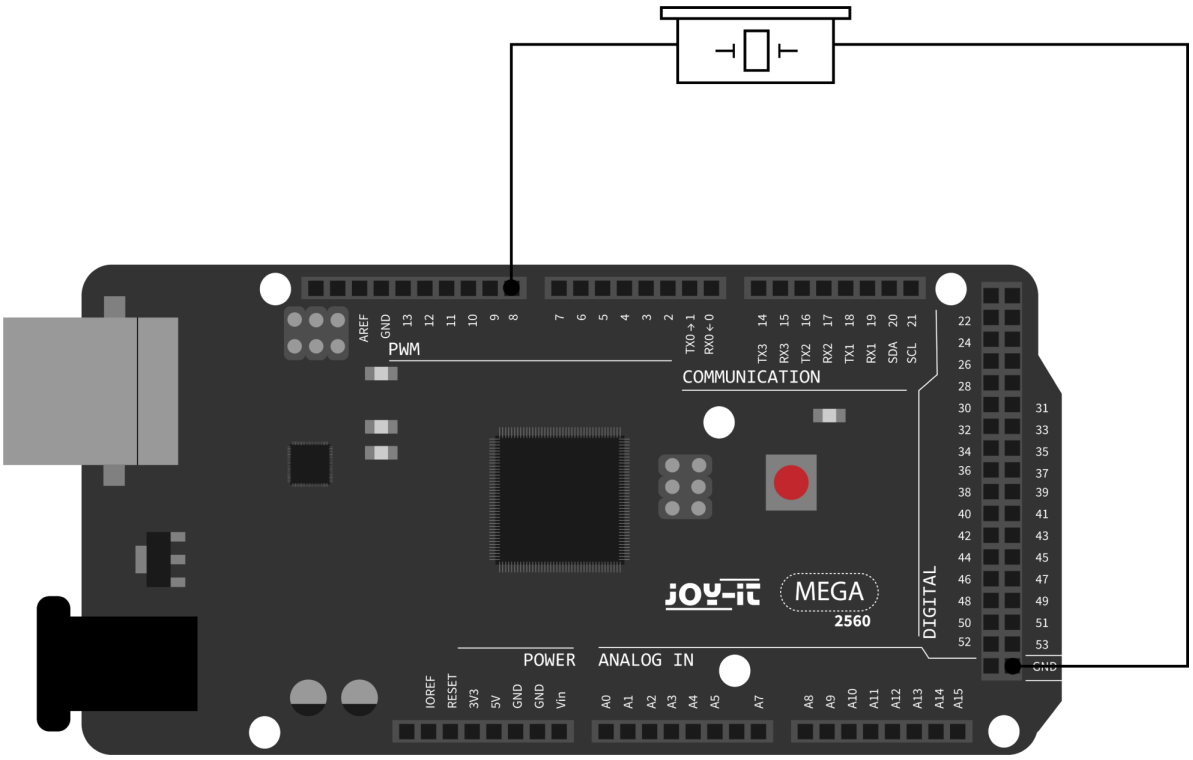


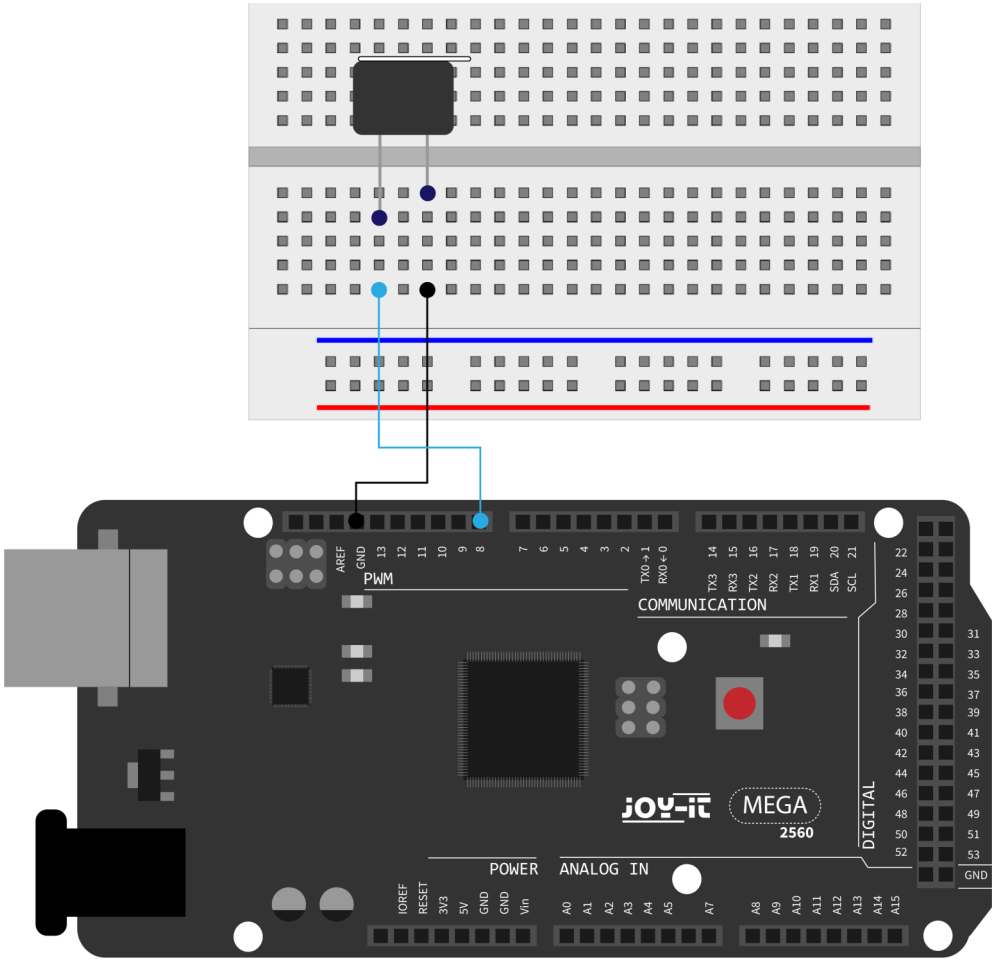
1x	Mega2560 Platine
1x	USB-Kabel
1x	Aktiver Summer (mit Aufkleber)
1x	Breadboard
2x	Überbrückungskabel







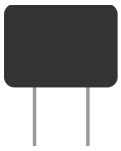





```
int buzzer=8;
// Initialisiere digitalen I/O Pin, welcher den Summer kontrolliert

void setup() {
    pinMode(buzzer,OUTPUT); // Stellt den Pin Modus auf „Ausgang“
}

void loop() {
    digitalWrite(buzzer, HIGH); // Macht Geräusche
}
```

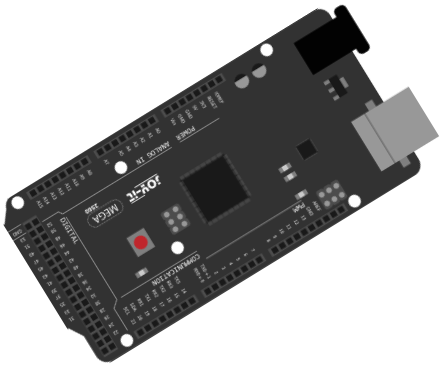
Das Projekt ist nach dem Übertragen des Programms abgeschlossen. Der Summer wird nach der Übertragung mit Strom versorgt und wird Geräusche erzeugen.

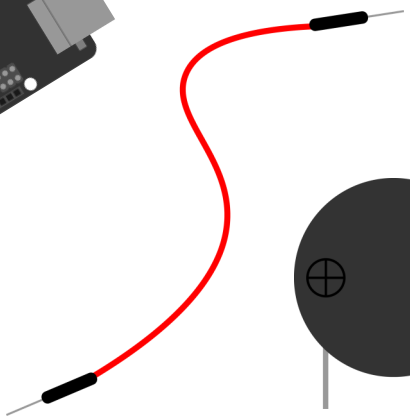
Lektion 9: Passiver Summer

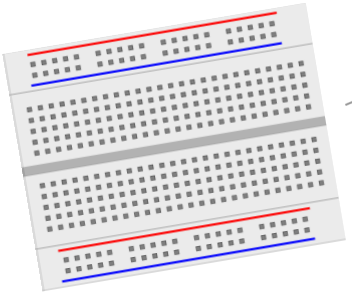


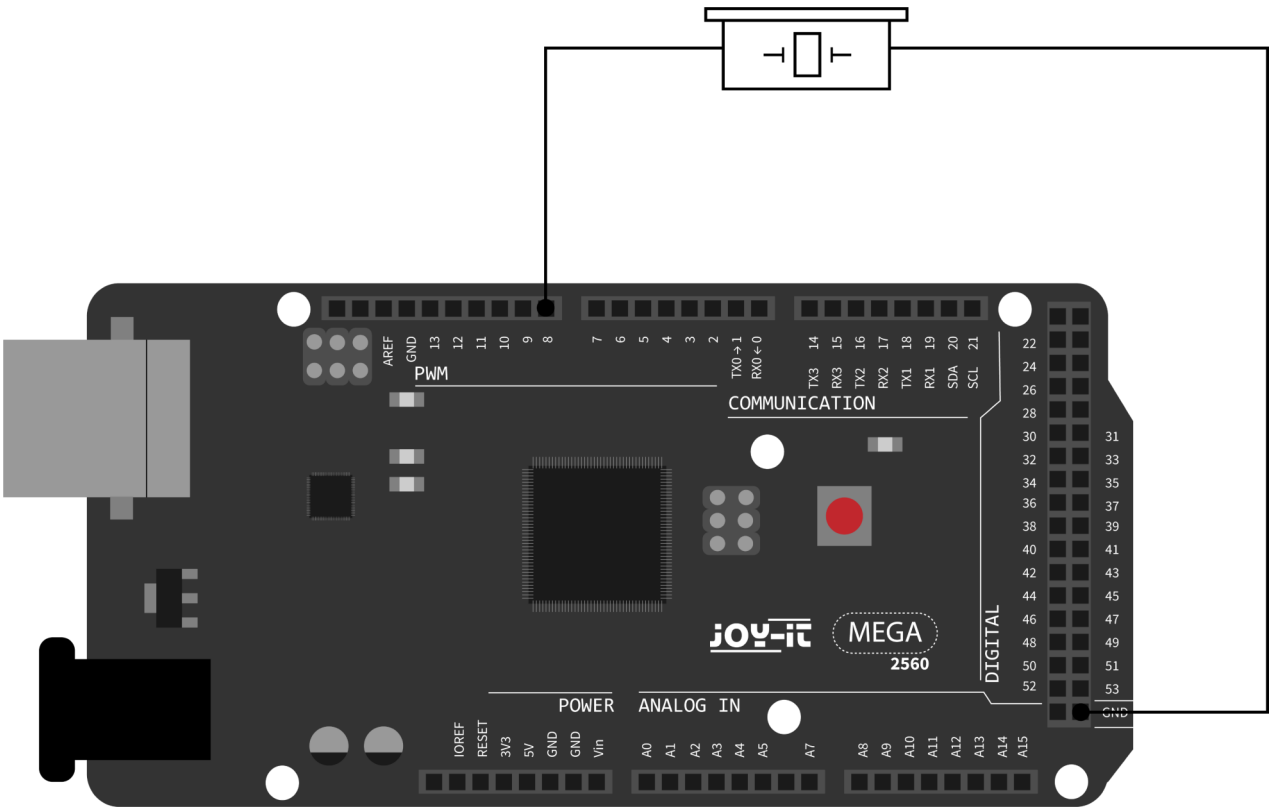
Mit dem Mega2560 sind viele interaktive Projekte möglich. Die bisherigen Projekte haben sich hauptsächlich mit LEDs beschäftigt, doch ein häufig verwendetes Projekt ist das akustisch-optische Display. Hierfür wird ein passiver Summer verwendet, der sich im Gegensatz zum aktiven Summer, nicht selbst aktivieren kann.
Die Aktivierung erfolgt über eine Pulsfrequenz. Unterschiedliche Frequenzen resultieren beim Summer in unterschiedlichen Tönen. Dies kann dafür genutzt werden, um beispielsweise die Melodie eines Liedes wiederzugeben.
Dazu wird benötigt:

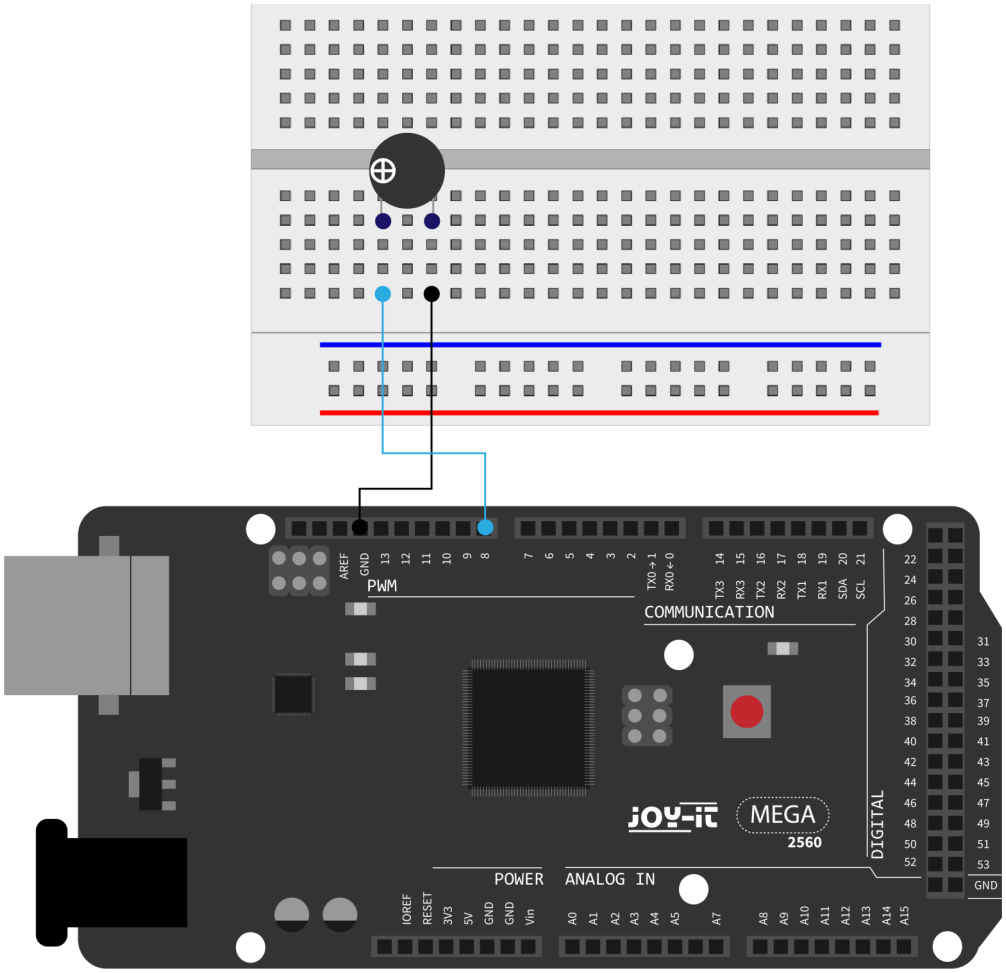
1x	Mega2560 Platine
1x	USB-Kabel
1x	Passiver Summer (ohne Aufkleber)
1x	Breadboard
2x	Überbrückungskabel





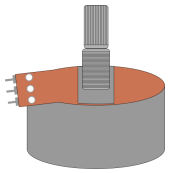






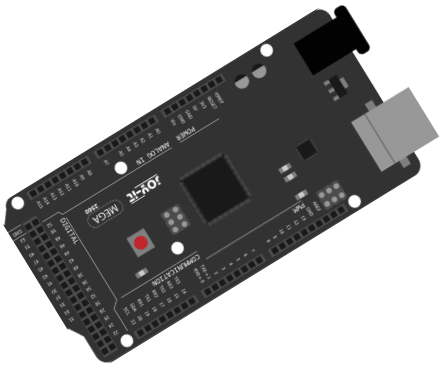

```
int buzzer=8; // Wählt digitalen I/O Pin für den Summer aus

void setup() {
    pinMode(buzzer,OUTPUT); // Setzt die digitale IOs als Ausgang.
}

void loop() {
    unsigned char i,j; // Bestimmt Variable
    while(1){
        for(i=0;i<80;i++) { // Gibt einen Frequenzton aus
            digitalWrite(buzzer,HIGH); // Ton
            delay(1); // 1ms Verzoegerung
            digitalWrite(buzzer,LOW); // Kein Ton
            delay(1); // 1ms Verzoegerung
        }
        for(i=0;i<100;i++) { // Gibt Frequenzton aus
            digitalWrite(buzzer,HIGH); // Ton
            digitalWrite(buzzer,LOW); // Kein Ton
            delay(2); // 2ms Verzögerung

        }
    }
}
```

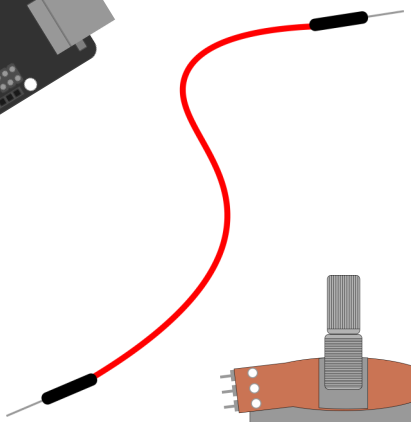
Lektion 10: Analogwert auslesen

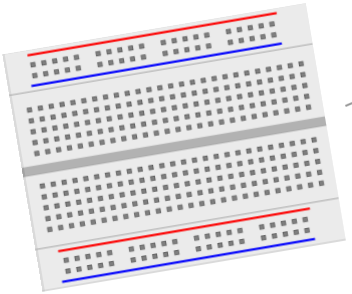


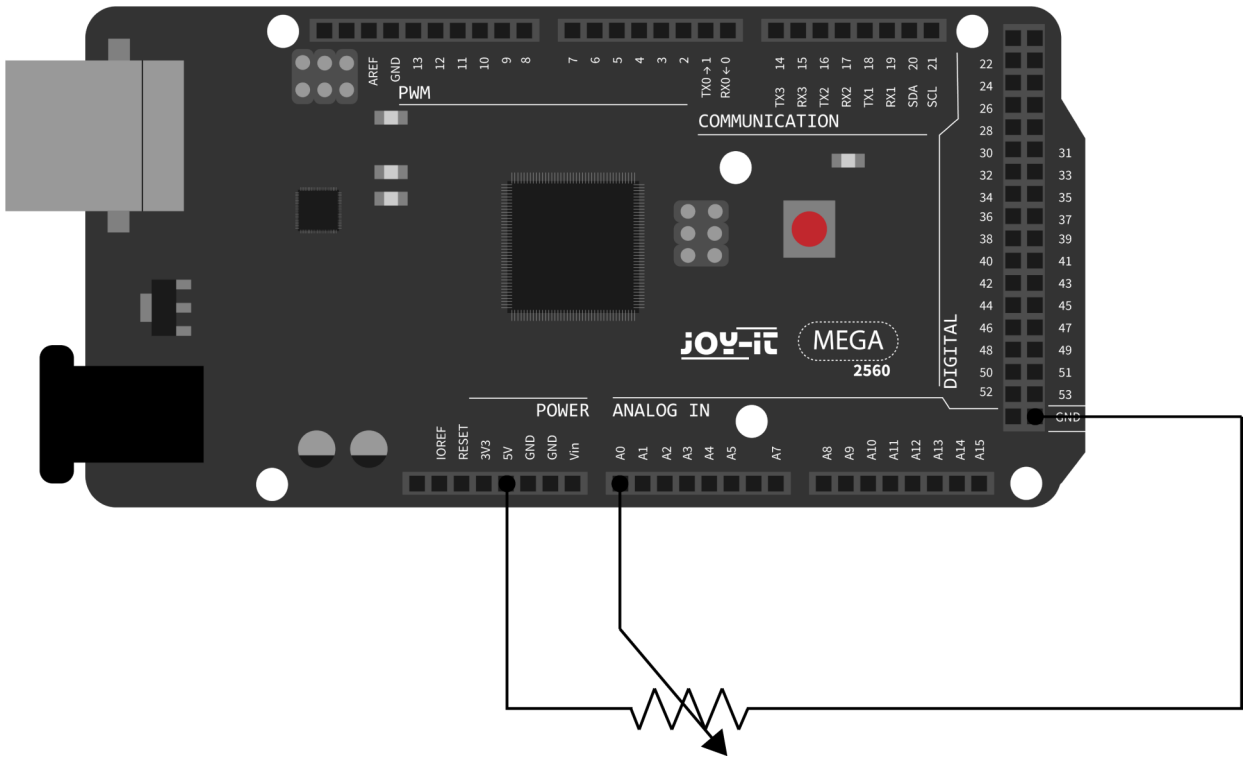
In diesem Projekt geht es um die analogen Schnittstellen des Mega2560. Ein *analogRead()*-Befehl kann den Wert der Schnittstelle lesen. Durch die Analog-Digital-Umwandlung des Mega2560 liegen die ausgelesenen Werte zwischen 0 und 1023.
Um die Werte auslesen zu können, ist es wichtig auf die richtige Baudrate zu achten (hier: 9600). Die Baudrate des Computers sollte der Baudrate des Gerätes entsprechen. Öffnet man den seriellen Monitor der Arduino IDE, so kann man die Baudrate in der unteren rechten Ecke konfigurieren.
In diesem Projekt wird der eingestellte Widerstandswert eines Potentiometers zu einem analogen Signal umgewandelt und anschließend auf dem Bildschirm ausgegeben.
Hierzu wird benötigt:

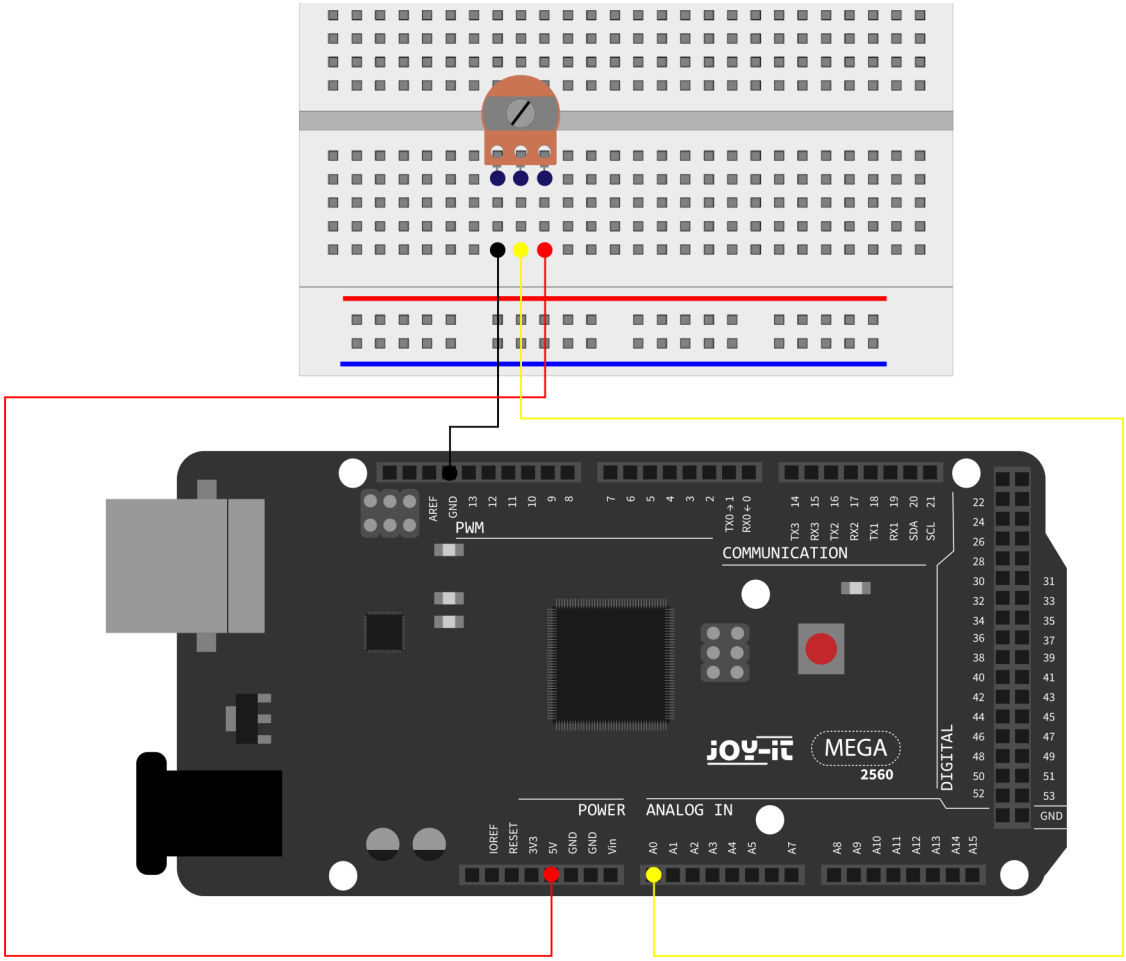
1x	Mega2560 Platine
1x	USB-Kabel
1x	Potentiometer
1x	Breadboard
3x	Überbrückungskabel







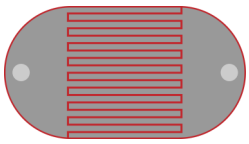





```
int potpin=0; // Initialisiert analogen Pin 0
int ledpin=13; // Initialisiert digitalen Pin 13
int val=0; // Definiert „Val“, ordnet Ursprungswert zu

void setup() {
    pinMode(ledpin,OUTPUT); // Setzt digitalen Pin als „Ausgang“
    Serial.begin(9600); // Setzt Baudrate auf 9600
}

void loop() {
    digitalWrite(ledpin,HIGH); // Schaltet LED auf Pin 13 ein
    delay(50); // Wartet 0,05 Sekunden
    digitalWrite(ledpin,LOW); // Schaltet LED auf Pin 13 aus
    delay(500); // Wartet 0,05 Sekunden
    val=analogRead(potpin); // Liest Analogwert und ordnet ihn „Val“ zu
    Serial.println(val); // Zeigt Wert von „Val“
}
```

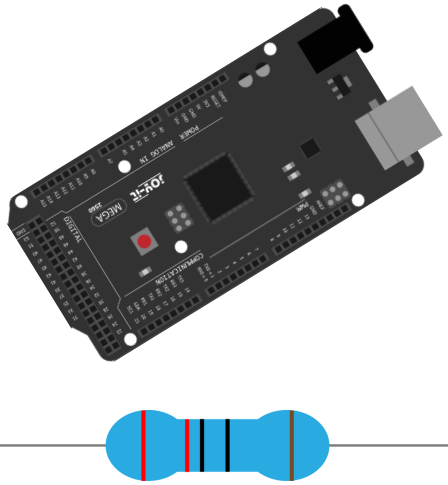
Die ausgelesenen Werte werden im seriellen Monitor ausgegeben.

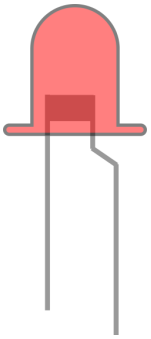
Lektion 11: Fotowiderstand

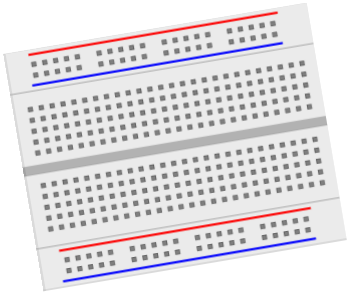


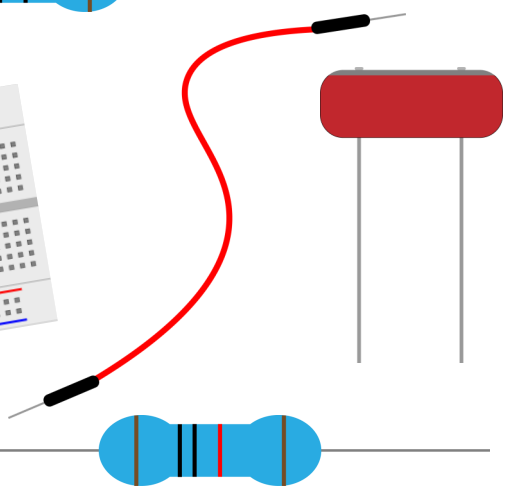
Ein Fotowiderstand ist ein Widerstand, dessen Widerstandskraft je nach einfallender Lichtstärke variiert. Er basiert auf dem fotoelektrischen Effekt von Halbleitern. Wenn das einfallende Licht intensiv ist, reduziert sich die Widerstandskraft. Wenn das einfallende Licht schwach ist, erhöht sich die Widerstandskraft. Fotowiderstände werden normalerweise zur Lichtmessung, Lichtkontrolle und zur Photovoltaik-Umwandlung (konvertiert die Änderung von Licht in eine Änderung der Elektrizität) benutzt. Sie werden in diversen Licht-Steuerung-Schaltkreisen eingesetzt zum Beispiel als optische Schalter. In diesem Projekt wird dieser Effekt genutzt um eine LED der aktuellen Lichtstärke anzupassen.
Hierzu wird benötigt:

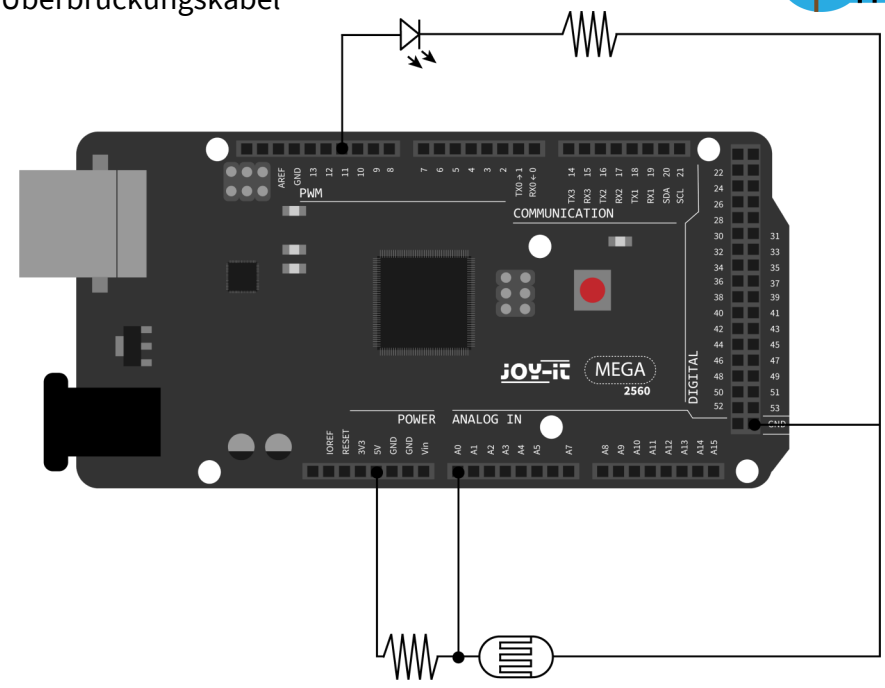
1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	220 Ω Widerstand
1x	10 kΩ Widerstand
1x	Fotowiderstand
1x	Breadboard
5x	Überbrückungskabel

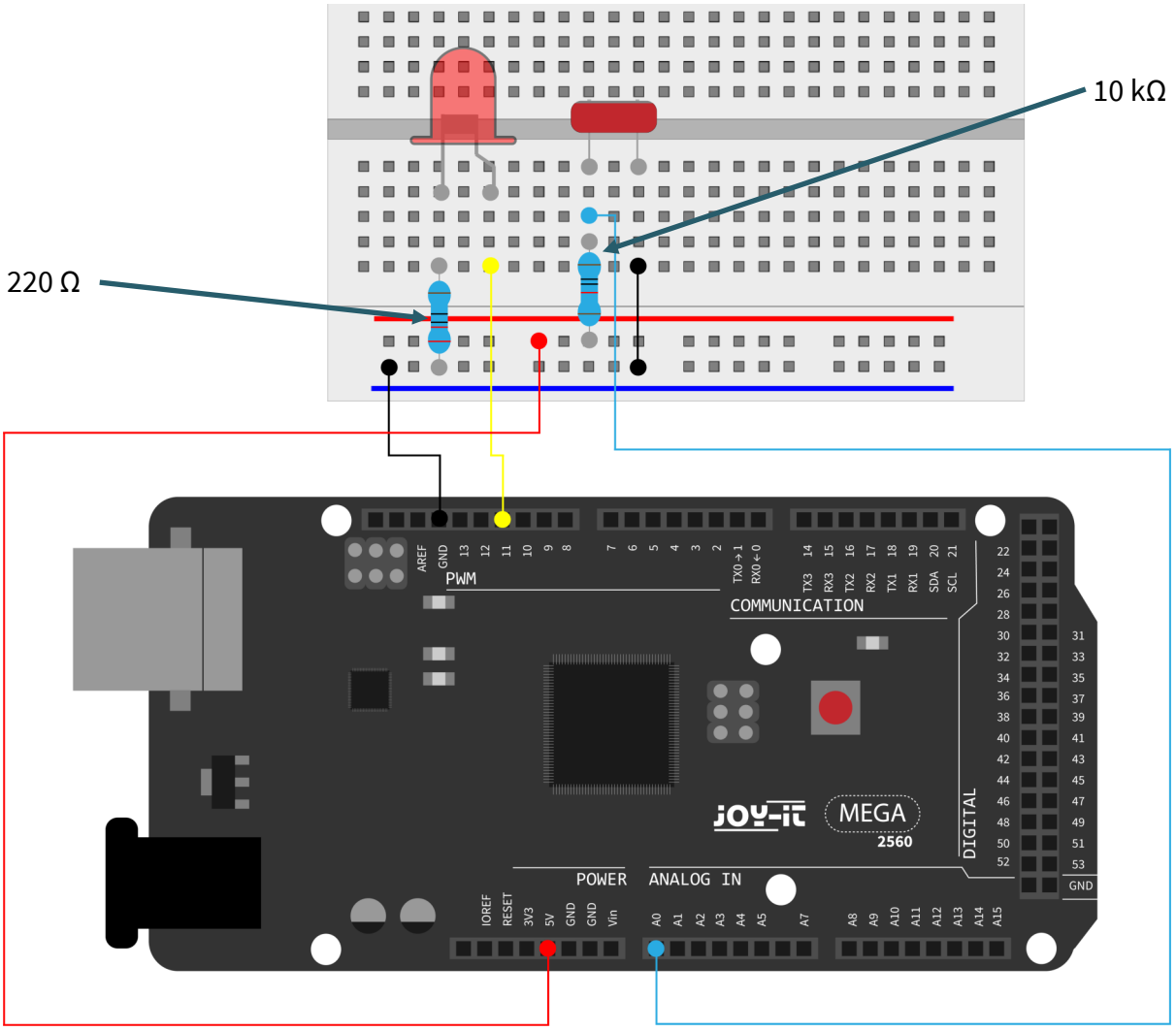






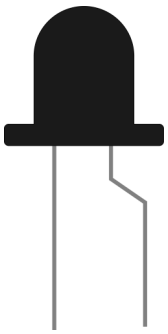






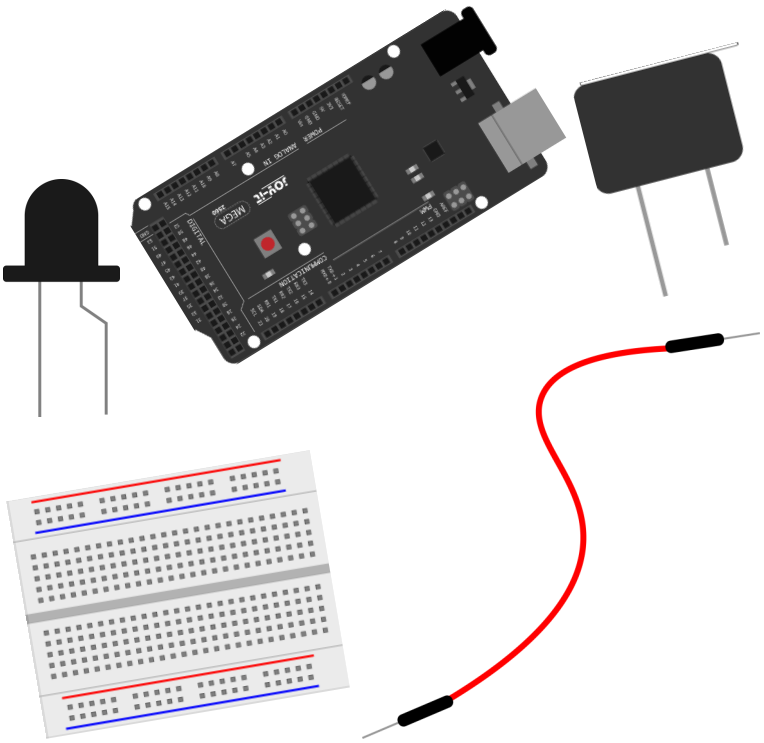

```
int potpin=0;
// Initialisiert analogen Pin 0 an dem Photovaristor angeschlossen ist
int ledpin=11; // Initialisiert digitalen Pin 11
// Ausgang welcher die Helligkeit der LED reguliert
int val=0; // Initialisiert Variable „Val“

void setup() {
    pinMode(ledpin,OUTPUT); // Stellt Pin 11 als Ausgang ein
    Serial.begin(9600); // Setzt Baudrate auf „9600“
}

void loop() {
    val=analogRead(potpin);
    // Liest den Analogwert des Sensors und dieser wird „Val“ zugewiesen
    Serial.println(val); // Zeigt den Wert von „Val“ an
    analogWrite(ledpin,val);
    // Schaltet die LED ein und stellt die Helligkeit ein
    delay(10); // Wartet 0,01 Sekunden
}
```

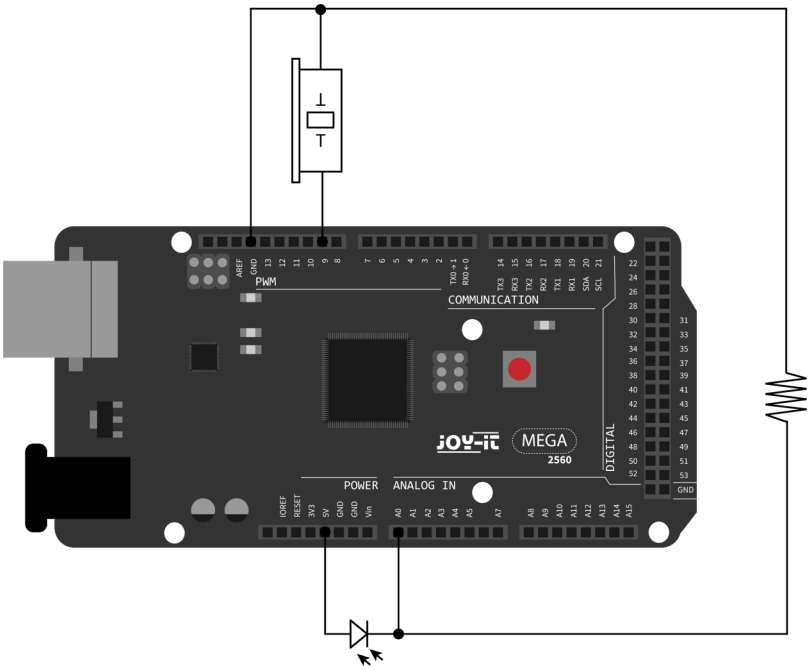
Lektion 12: Flammensensor

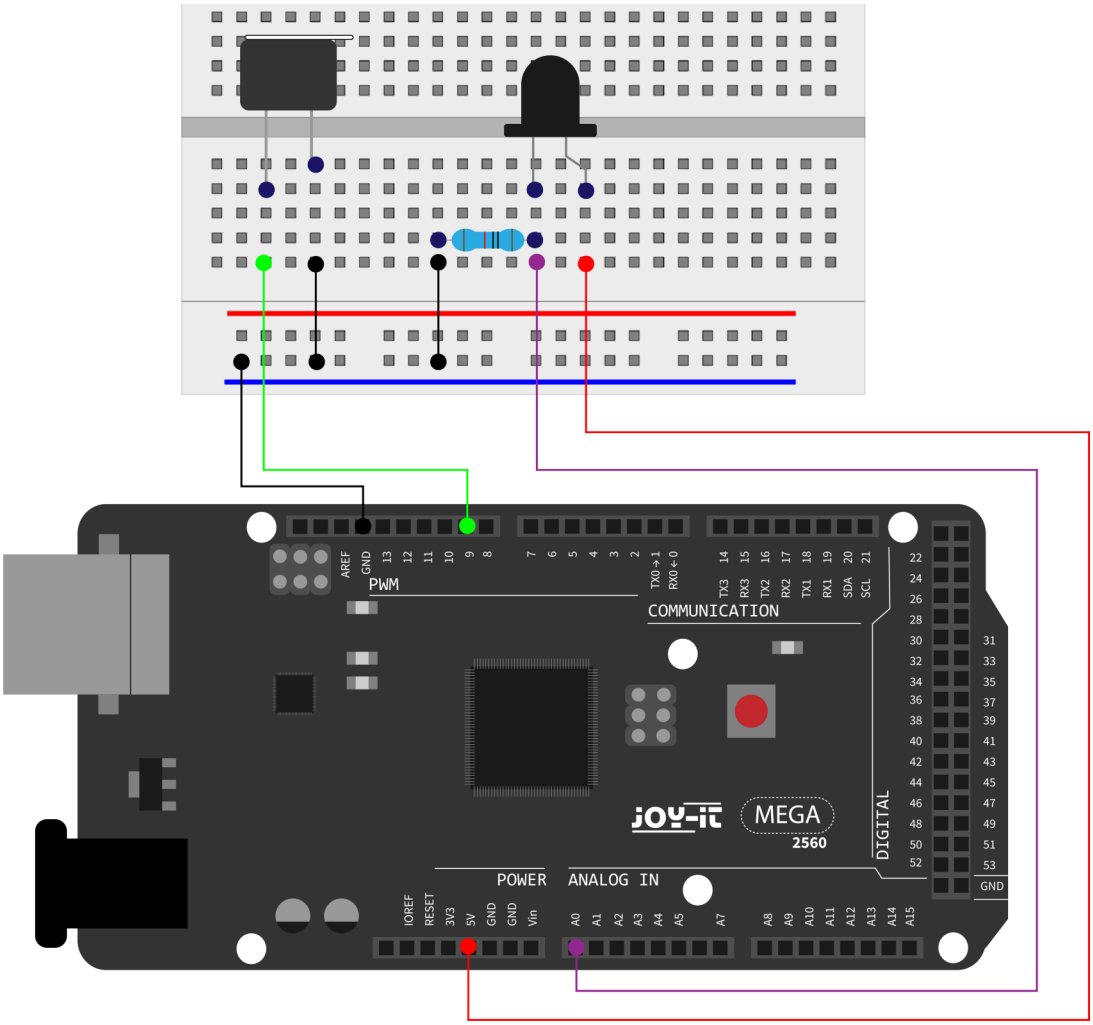


Der Flammensensor (Infrarotempfangende Triode) wird speziell auf Robotern verwendet um Flammenquellen zu finden. Dieser Sensor hat eine hohe Sensitivität zu Flammen. Der Flammensensor ist nach dem Prinzip gebaut, dass Infrarotstrahlung sehr empfindlich auf Feuer reagiert. Er hat ein speziell entworfenes Infrarot-Aufnahmerohr um Feuer zu entdecken, um dann die Helligkeit der Flamme zu einem Signal umzuwandeln. Diese Signale werden dann zum zentralen Prozessor weitergegeben und entsprechend verarbeitet.
Wenn der Sensor sich einem Feuer nähert, ändert sich der analoge Spannungswert. Mit einem Multimeter lässt sich überprüfen, dass die Spannung bei ca. 0,3 V liegt, wenn sich kein Feuer in der Nähe befindet. Befindet sich ein Feuer in der Nähe, so liegt die Spannung bei ca. 1,0 V. Je höher die Spannung, desto näher das Feuer.
Für dieses Projekt wird benötigt:

1x	Mega2560 Platine
1x	USB-Kabel
1x	Flammensensor
1x	Aktiver Summer (mit Aufkleber)
1x	10 kΩ Widerstand
1x	Breadboard
6x	Überbrückungskabel



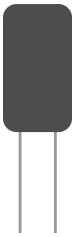




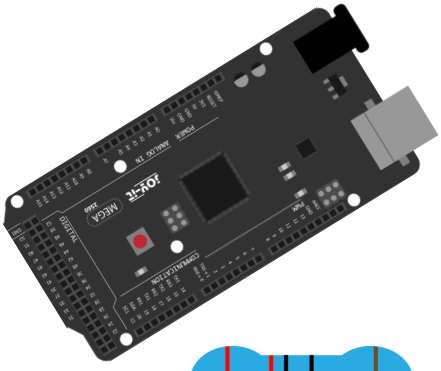

```
int flame=0; // Wählt analogen Pin 0 für Sensor aus
int Beep=9; // Wählt digitalen Pin 9 für Summer aus
int val=0; // Initialisiert Variable

void setup() {
    pinMode(Beep,OUTPUT); // Stellt Summer Pin als „Ausgang“ ein
    pinMode(flame,INPUT); // Stellt Sensor Pin als „Eingang“ ein
    Serial.begin(9600); // Setzt Baudrate auf „9600“
}

void loop() {
    val=analogRead(flame); // Liest den Analogwert des Sensors
    Serial.println(val); // Gibt den Analogwert aus
    if(val>=10) { // Summer summt, wenn Analogwert über 10
    //ggf. angepasst werden an den analogen Werten des Sensors
        digitalWrite(Beep,HIGH);
    }
    else{
        digitalWrite(Beep,LOW);
    }
    delay(500);
}
```

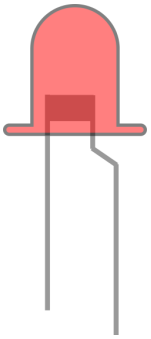
Lektion 13: Tilt-Sensor

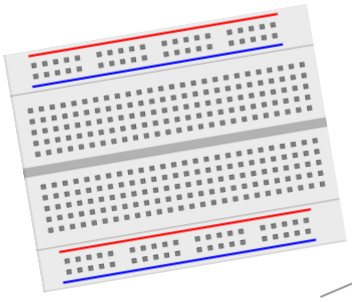
In dieser Lektion fungiert der Tilt-Sensor als ein Ein- und Ausschalter einer LED. Die LED ist dann eingeschaltet, wenn ein Ende vom Sensor unterhalb der horizontalen Position ist. Mittels der analogen Schnittstelle, an welcher der Tilt-Sensor angeschlossen ist, kann geprüft werden in welcher Lage sich der Sensor befindet.
Für diese Lektion benötigen Sie:

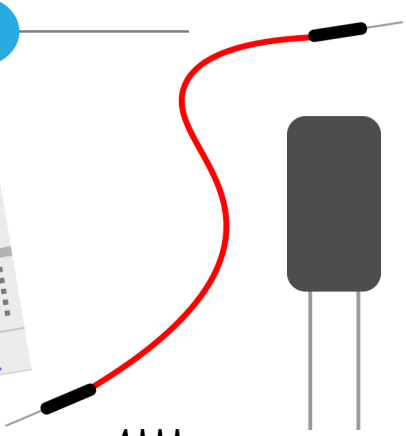


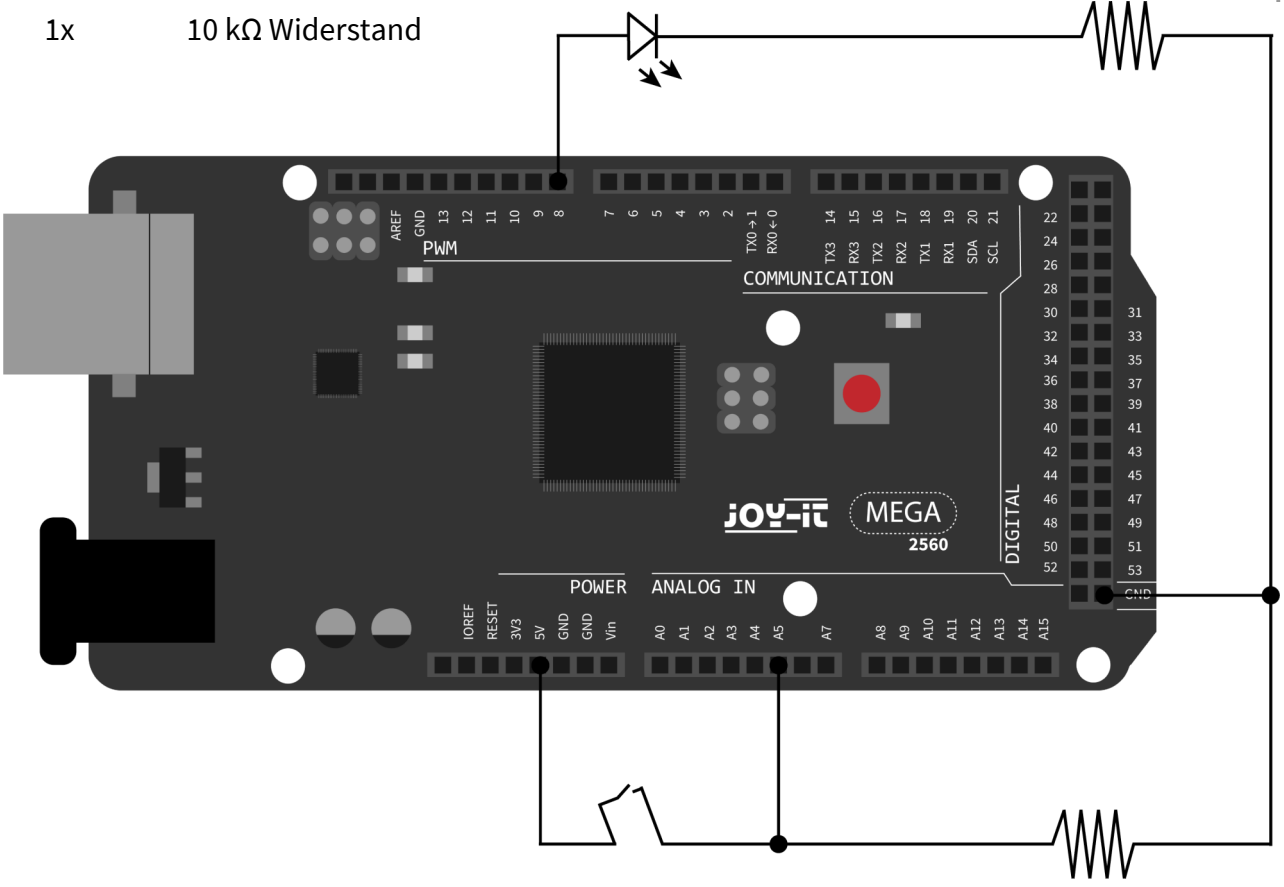
1x	Mega2560 Platine
1x	USB-Kabel
1x	Rote M5 LED
1x	220 Ω Widerstand
1x	Breadboard
5x	Überbrückungskabel
1x	Tilt-Sensor
1x	10 kΩ Widerstand

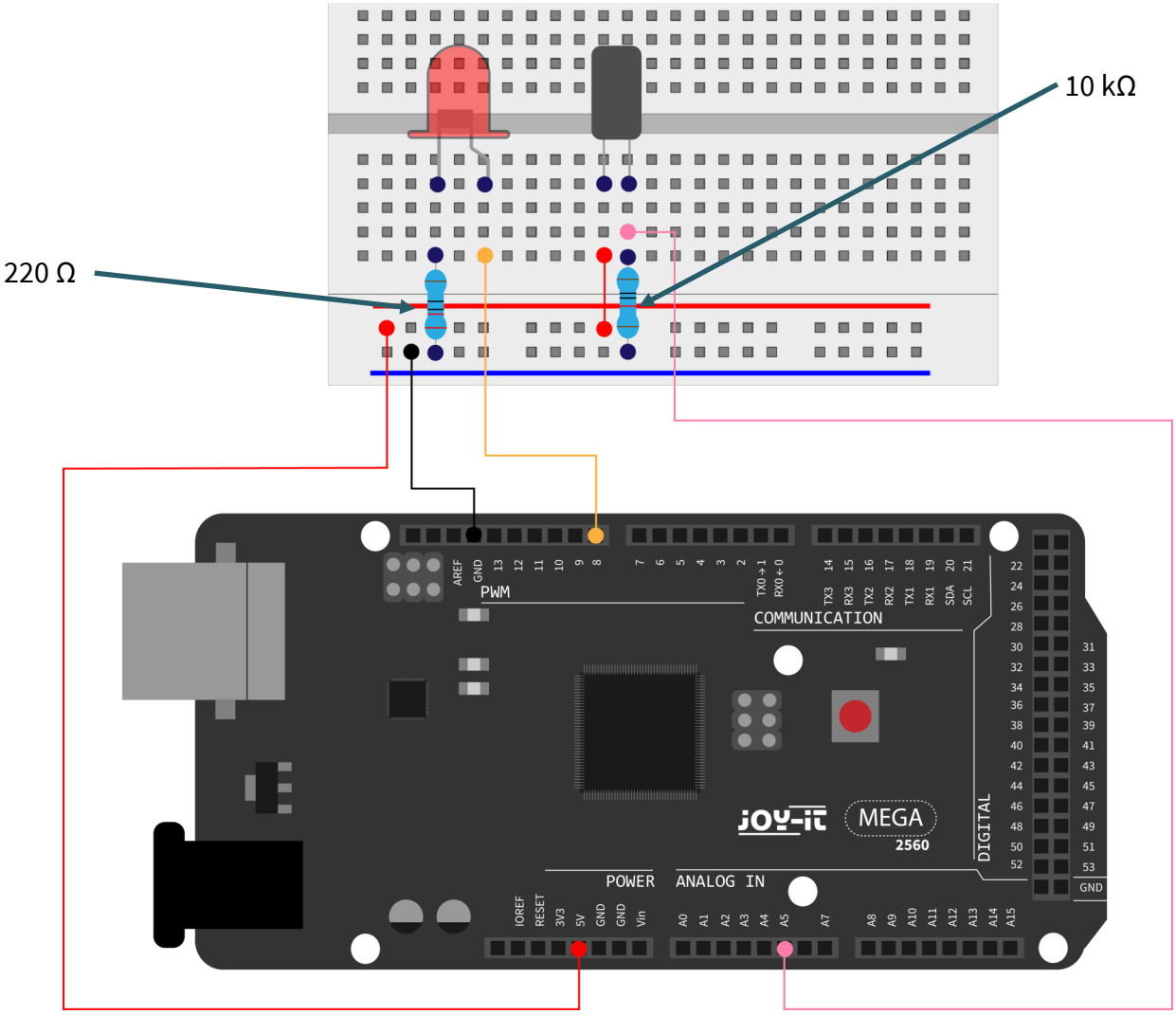






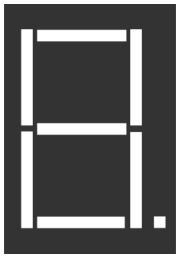






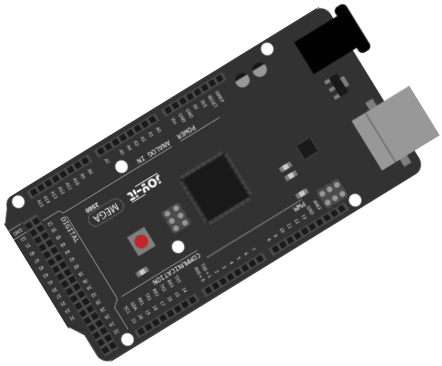

```
void setup() {
    pinMode(8,OUTPUT); // Stellt digitalen Pin 8 als „Ausgang“ ein
}

void loop() {
    int i; // Definiert Variable i
    while(1) {
        i=analogRead(5);// Liest den Spannungswert vom analogen Pin 5
        if(i>512) { // Wenn groeßer als 512 (2.5V)
            digitalWrite(8,LOW); // Schalte LED ein
        }
        else {
            digitalWrite(8,HIGH); // Schalte LED aus
        }
    }
}
```

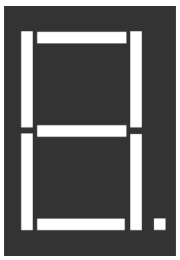
Lektion 14: 1-Ziffer LED Segment Anzeige

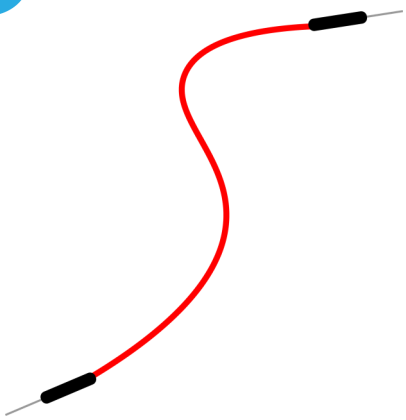


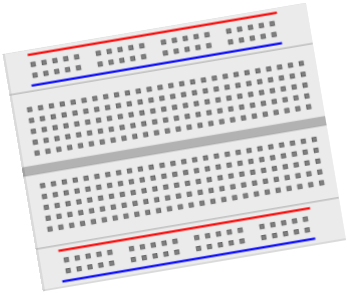
LED Segment Anzeigen sind verbreitet für die Anzeige von numerischen Informationen. Sie werden oft bei Anzeigen von elektromagnetischen Öfen, vollautomatischen Waschmaschinen, Wassertemperaturdisplays, elektronischen Uhren etc. verwendet. Die LED Segment Anzeige ist ein Halbleiter und ein Licht ausgebendes Gerät. Seine Basiseinheit ist eine LED.
Je nach Verdrahtung der LED-Einheiten, kann die LED Segment Anzeige in Anzeigen mit gemeinsamer Anode und Anzeigen mit gemeinsamer Kathode unterteilt werden. Die Anzeige mit gemeinsamer Anode kombiniert alle Anoden der LED-Einheiten in eine gemeinsame Anode (COM). Für die gemeinsame Anoden Anzeige muss die gemeinsame Anode (COM) mit + 5 V verbunden werden. Wenn der Kathodenpegel eines Segments niedrig ist, ist das Segment an. Wenn der Kathodenpegel eines Segments hoch ist, ist das Segment aus. Für die gemeinsame Kathodenanzeige muss die gemeinsame Kathode (COM) mit GND verbunden werden. Wenn der Anodenpegel eines Segments hoch ist, ist das Segment an. Wenn der Anodenpegel eines Segments niedrig ist, ist das Segment aus.
Für das folgende Projekt wird benötigt:

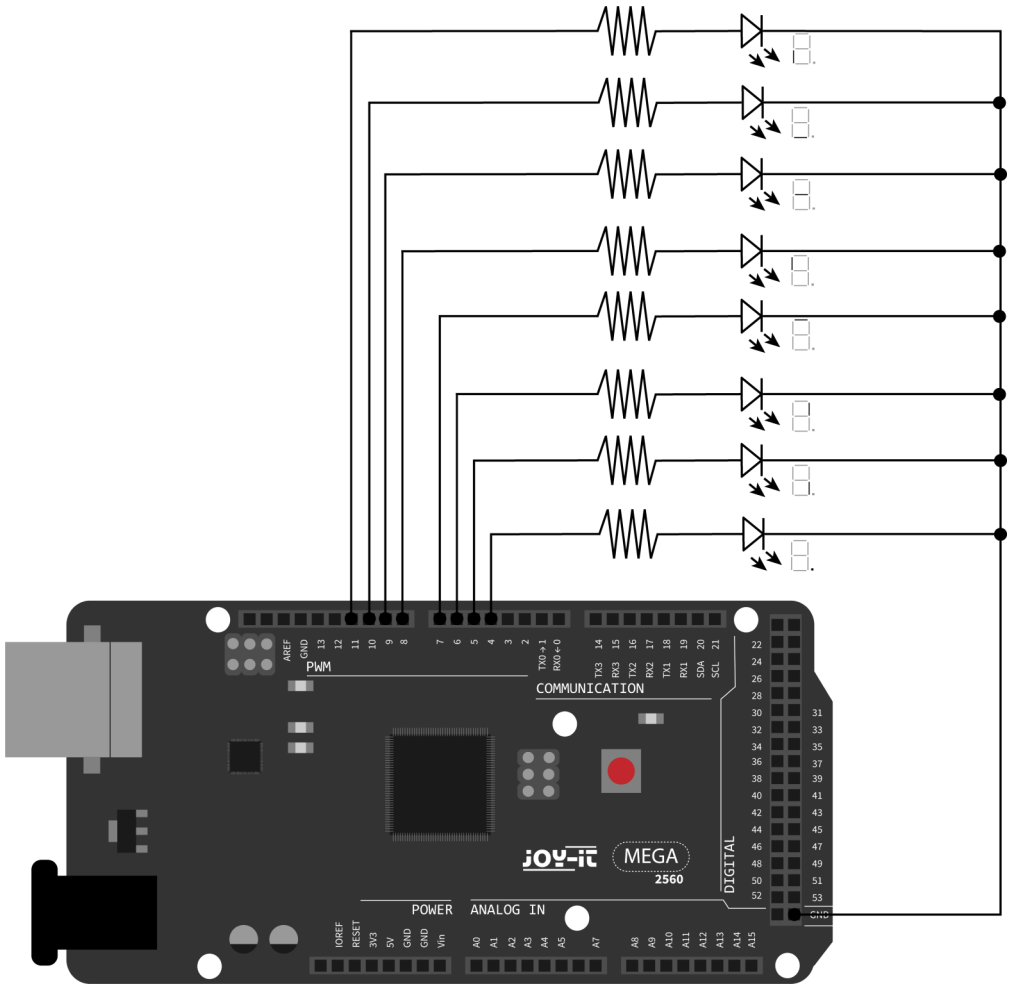
1x	Mega2560 Platine
1x	USB-Kabel
1x	1- Ziffer 7-Segment-Anzeige
8x	220 Ω Widerstand
1x	Breadboard
12x	Überbrückungskabel

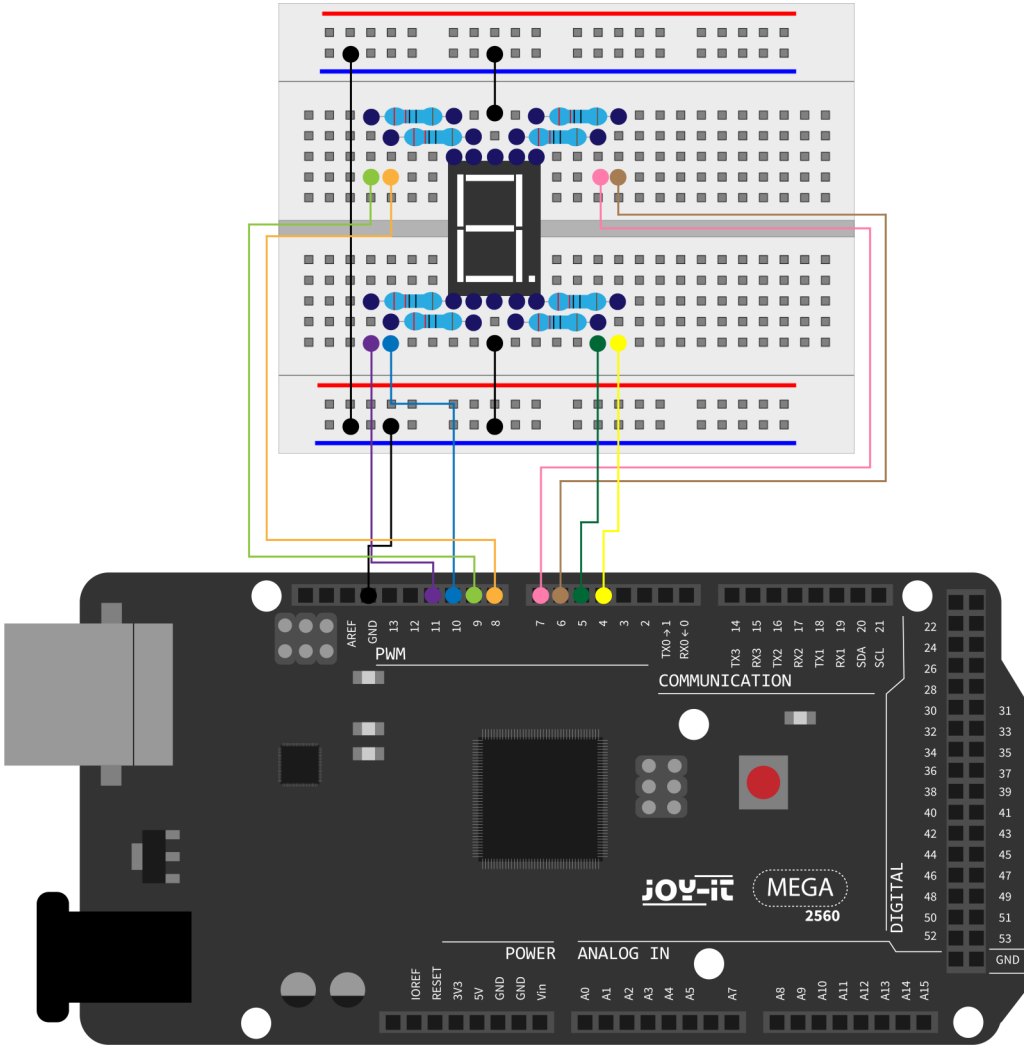









PWM
COMMUNICATION
DIGITAL
POWER
ANALOG IN
MEGA
2560


PWM
COMMUNICATION
DIGITAL
POWER
ANALOG IN
MEGA
2560

```
// Stelle den IO Pin für jedes Segment ein
int a=7; // Stellt digitalen Pin 7 für Segment a ein
int b=6; // Stellt digitalen Pin 6 für Segment b ein
int c=5; // Stellt digitalen Pin 5 für Segment c ein
int d=10; // Stellt digitalen Pin 10 für Segment d ein
int e=11; // Stellt digitalen Pin 11 für Segment e ein
int f=8; // Stellt digitalen Pin 8 für Segment f ein
int g=9; // Stellt digitalen Pin 9 für Segment g ein
int dp=4; // Stellt digitalen Pin 4 für Segment dp ein

void digital_0(void) { // Zeigt Nummer 0 an
    unsigned char j;
    digitalWrite(a,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(d,HIGH);
    digitalWrite(e,HIGH);
    digitalWrite(f,HIGH);
    digitalWrite(g,LOW);
    digitalWrite(dp,LOW);
}

void digital_1(void) { // Zeigt Nummer 1 an
    unsigned char j;
    digitalWrite(c,HIGH); // Setzt Pegel für Pin 5 auf „hoch“
    digitalWrite(b,HIGH); // Schaltet Segment b aus
    for(j=7;j<=11;j++) // Schaltet andere Segmente aus
        digitalWrite(j,LOW);
    digitalWrite(dp,LOW); // Schaltet Segment dp aus
}

void digital_2(void) { // Zeigt Nummer 2 an
    unsigned char j;
    digitalWrite(b,HIGH);
    digitalWrite(a,HIGH);
    for(j=9;j<=11;j++)
        digitalWrite(j,HIGH);
    digitalWrite(dp,LOW);
    digitalWrite(c,LOW);
    digitalWrite(f,LOW);
}
```

```
void digital_3(void) { // Zeigt Nummer 3 an
    digitalWrite(g,HIGH);
    digitalWrite(a,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(d,HIGH);
    digitalWrite(dp,LOW);
    digitalWrite(f,LOW);
    digitalWrite(e,LOW);
}

void digital_4(void) { // Zeigt Nummer 4 an
    digitalWrite(c,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(f,HIGH);
    digitalWrite(g,HIGH);
    digitalWrite(dp,LOW);
    digitalWrite(a,LOW);
    digitalWrite(e,LOW);
    digitalWrite(d,LOW);
}

void digital_5(void) { // Zeigt Nummer 5 an
    unsigned char j;
    digitalWrite(a,HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c,HIGH);
    digitalWrite(d,HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f,HIGH);
    digitalWrite(g,HIGH);
    digitalWrite(dp,LOW);
}

void digital_6(void) { // Zeigt Nummer 6 an
    unsigned char j;
    for(j=7;j<=11;j++)
        digitalWrite(j,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(dp,LOW);
    digitalWrite(b,LOW);
}
```

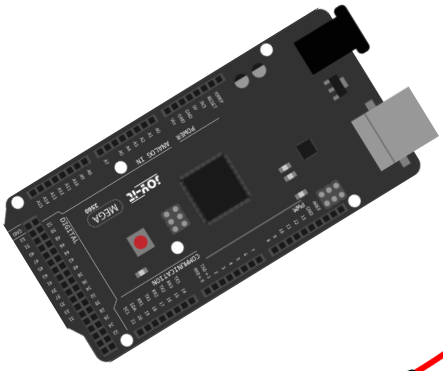
```
void digital_7(void) { // Zeigt Nummer 7 an
    unsigned char j;
    for(j=5;j<=7;j++)
        digitalWrite(j,HIGH);
    digitalWrite(dp,LOW);
    for(j=8;j<=11;j++)
        digitalWrite(j,LOW);
}

void digital_8(void) { // Zeigt Nummer 8 an
    unsigned char j;
    for(j=5;j<=11;j++)
        digitalWrite(j,HIGH);
    digitalWrite(dp,LOW);
}

void digital_9(void) { // Zeigt Nummer 5 an
    unsigned char j;
    digitalWrite(a,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(d,HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f,HIGH);
    digitalWrite(g,HIGH);
    digitalWrite(dp,LOW);
}

void setup() {
    int i; // deklariert eine Variable
    for(i=4;i<=11;i++)
        pinMode(i,OUTPUT); // Stellt Pin 4-11 als „Ausgang“
}

void loop() {
    while(1){
        digital_0(); // Zeigt Nummer 0 an
        delay(1000); // Wartet 1 Sekunde
        digital_1(); // Zeigt Nummer 1 an
        delay(1000); // Wartet 1 Sekunde
        digital_2(); // Zeigt Nummer 2 an
        delay(1000); // Wartet 1 Sekunde
        digital_3(); // Zeigt Nummer 3 an
        delay(1000); // Wartet 1 Sekunde
        digital_4(); // Zeigt Nummer 4 an
```

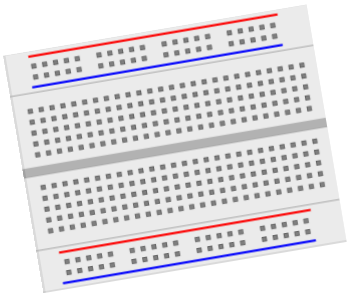
```
        delay(1000); // Wartet 1 Sekunde
        digital_5(); // Zeigt Nummer 5 an
        delay(1000); // Wartet 1 Sekunde
        digital_6(); // Zeigt Nummer 6 an
        delay(1000); // Wartet 1 Sekunde
        digital_7(); // Zeigt Nummer 7 an
        delay(1000); // Wartet 1 Sekunde
        digital_8(); // Zeigt Nummer 8 an
        delay(1000); // Wartet 1 Sekunde
        digital_9(); // Zeigt Nummer 9 an
        delay(1000); // Wartet 1 Sekunde
    }
}
```

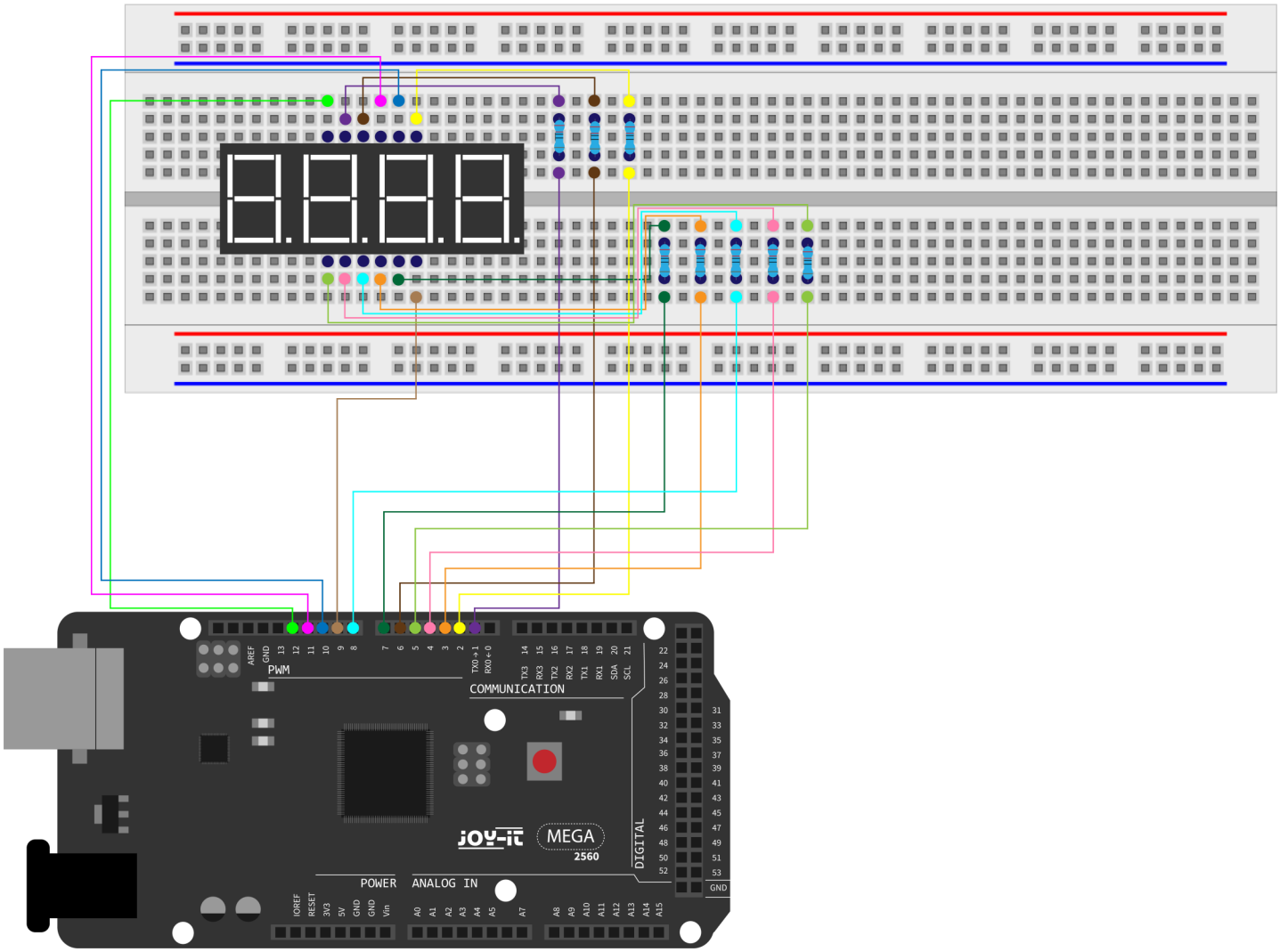
Lektion 15: 4-Ziffern LED Segment-Anzeige

In diesem Projekt wird eine 4-Ziffern 7-Segment-Anzeige betrieben. Für LED Anzeigen sind strombegrenzende Widerstände unverzichtbar. Es gibt zwei Verdrahtungsmethoden für strombegrenzende die d1-d4 Anode. Ein Vorteil dieser Methode ist, dass sie weniger Widerstände benötigt und zwar nur 4 Stück. Aber diese Methode kann keine konstante Helligkeit erhalten. Die zweite Methode ist an jeden Pin einen Widerstand zu verbinden.
Für die erste Methode wird benötigt:

1x	Mega2560 Platine
1x	USB-Kabel
1x	4-Ziffern 7-Segment-Anzeige
8x	220 Ω Widerstand
1x	Breadboard
20x	Überbrückungskabel








```
// PIN für Anode
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int dp = 8;

// PIN für Kathode
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;

// Stellt Variable ein
long n = 1230;
int x = 100;
int del = 55;
```

```
void setup() {
    pinMode(d1, OUTPUT);
    pinMode(d2, OUTPUT);
    pinMode(d3, OUTPUT);
    pinMode(d4, OUTPUT);
    pinMode(a, OUTPUT);
    pinMode(b, OUTPUT);
    pinMode(c, OUTPUT);
    pinMode(d, OUTPUT);
    pinMode(e, OUTPUT);
    pinMode(f, OUTPUT);
    pinMode(g, OUTPUT);
    pinMode(dp, OUTPUT);
}

void loop() {
    Display(1, 1);
    Display(2, 2);
    Display(3, 3);
    Display(4, 4);
}

void position(unsigned char n){
    switch(n) {
        case 1:
            digitalWrite(d1,LOW);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, HIGH);
            break;
        case 2:
            digitalWrite(d1, HIGH);
            digitalWrite(d2, LOW);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, HIGH);
            break;
        case 3:
            digitalWrite(d1,HIGH);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, LOW);
            digitalWrite(d4, HIGH);
            break;
```

```
        case 4:
            digitalWrite(d1, HIGH);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, LOW);
            break;
        default :
            digitalWrite(d1, HIGH);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, HIGH);
            break;
    }
}
void Num_0() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}
void Num_1() {
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}

void Num_2() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
```

```
void Num_3() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}

void Num_4() {
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}

void Num_5() {
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}

void Num_6() {
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
```

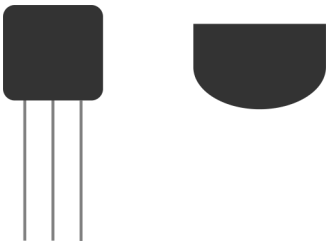
```
void Num_7() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}

void Num_8() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}

void Num_9() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Clear() { // Leert den Bildschirm
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}
```

```
void pickNumber(unsigned char n) { // Wählt Nummer
    switch(n) {
        case 0:
            Num_0();
            break;
        case 1:
            Num_1();
            break;
        case 2:
            Num_2();
            break;
        case 3:
            Num_3();
            break;
        case 4:
            Num_4();
            break;
        case 5:
            Num_5();
            break;
        case 6:
            Num_6();
            break;
        case 7:
            Num_7();
            break;
        case 8:
            Num_8();
            break;
        case 9:
            Num_9();
            break;
        default:
            Clear();
            break;
    }
}

void Display(unsigned char x, unsigned char Number) {
    position(x);
    pickNumber(Number);
    delay(1);
    Clear() ; // Leert den Bildschirm
}
```

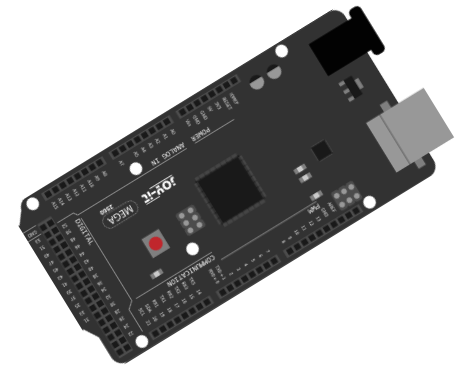
Wir der obiger Code vollständig auf den Mega2560 übertragen, so gibt die Anzeige 1234 aus.

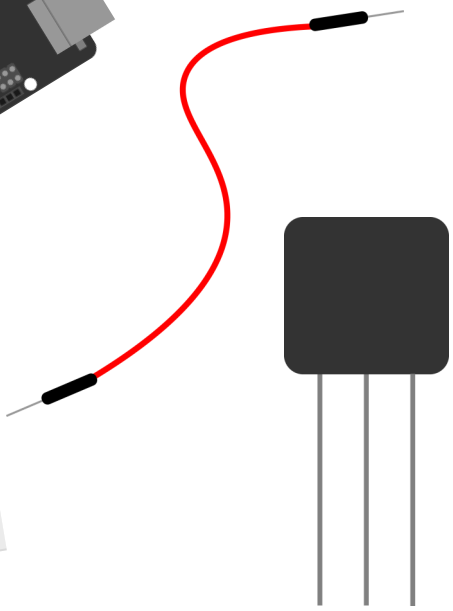
Lektion 16: LM35 Temperatursensor

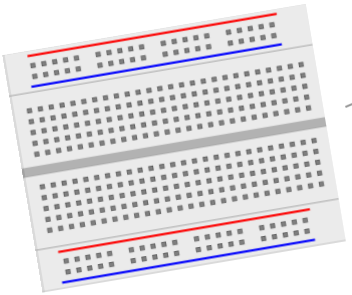
Der LM35 ist ein verbreiteter und einfach zu verwendender Temperatursensor. Man braucht keine sonstige Hardware. Die einzige Schwierigkeit besteht darin, den Code zu schreiben, welcher dafür sorgt, dass die Analogwerte , die er liest, in Celsius umgerechnet werden.
Man benötigt:

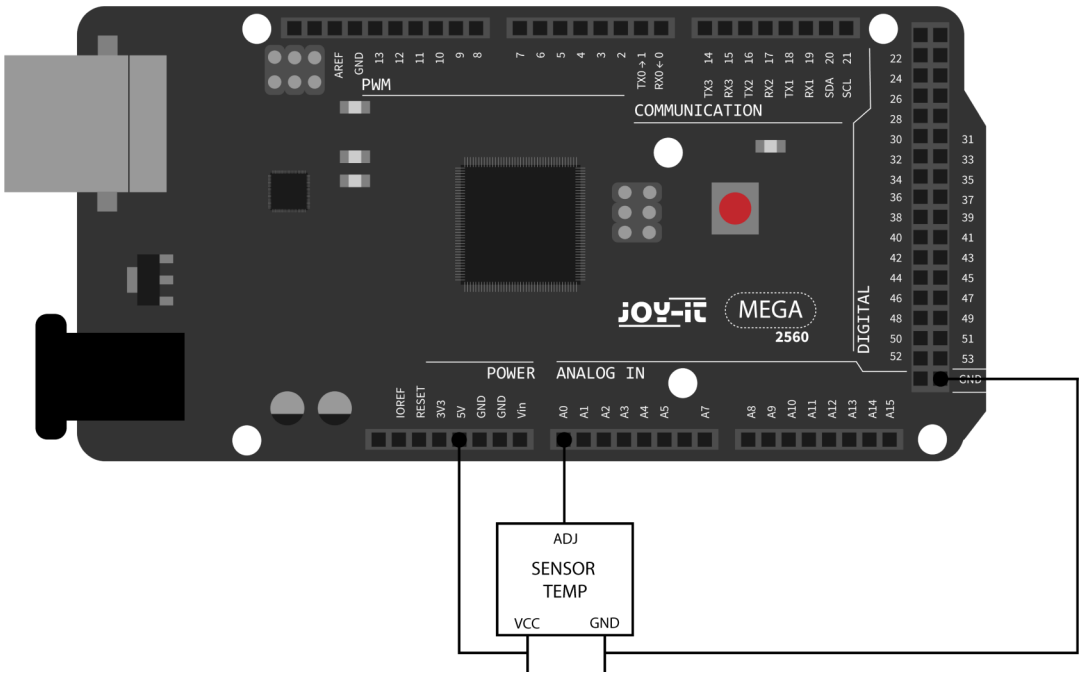


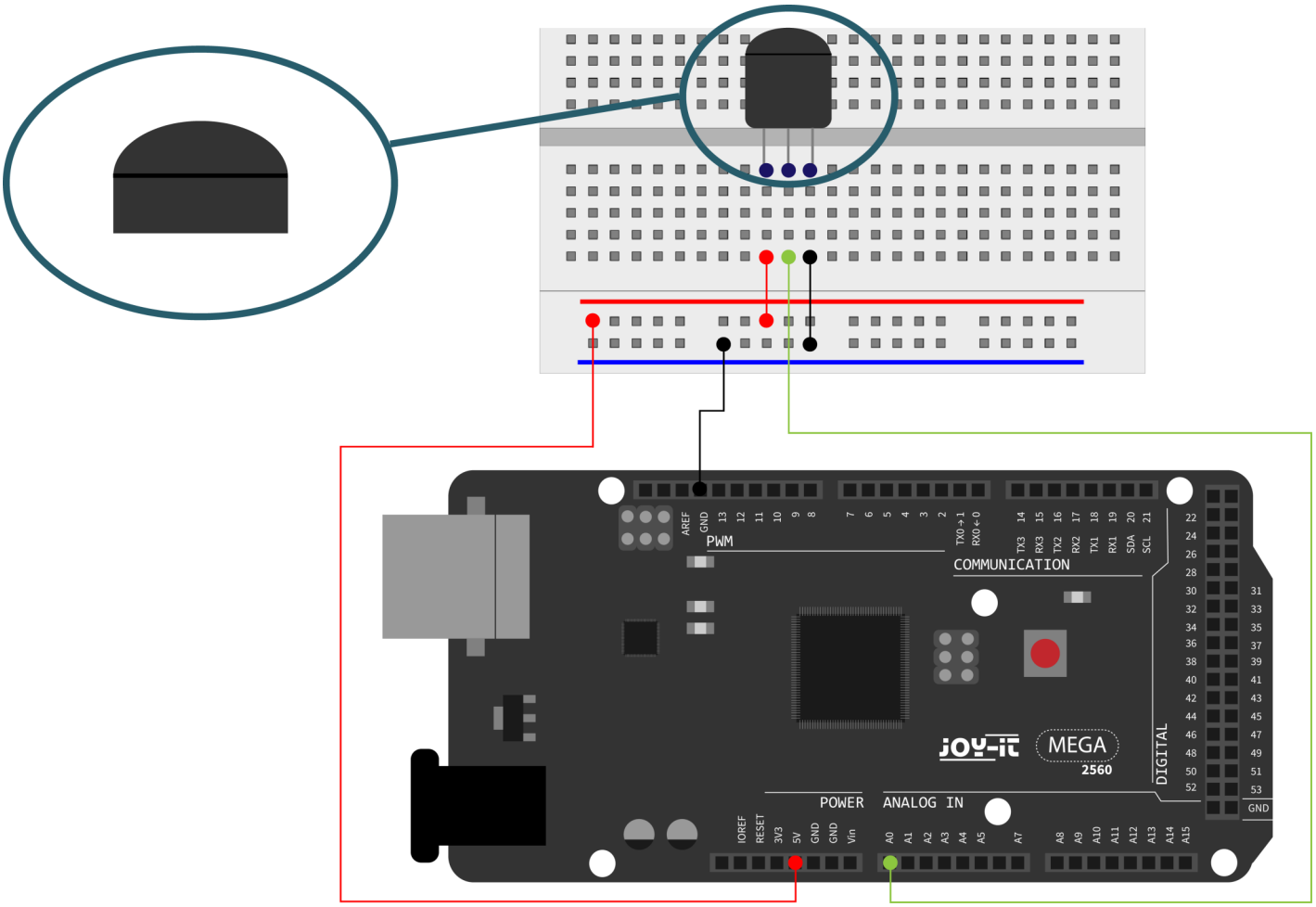
1x	Mega2560 Platine
1x	USB-Kabel
1x	LM35
1x	Breadboard
5x	Überbrückungskabel







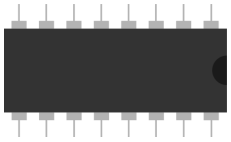





```
int potPin = 0; // Initialisiert Port A0 für Sensor

void setup() {
    Serial.begin(9600); // Stellt die Baudrate auf „9600“
}

void loop() {
    int val; // Definiert Variable
    int dat; // Definiert Variable
    val=analogRead(0); // Liest den Analogwert vom Sensor
    dat=(125*val)>>8; // Temperaturkalkulationsformel
    Serial.print("Temp:"); // Ausgabe beginnt mit Temp
    Serial.print(dat); // Ausgang und Anzeigewert von dat
    Serial.println(" *C"); // Zeigt „*C“ an
    delay(500); // Wartet 0,5 Sekunden
}
```

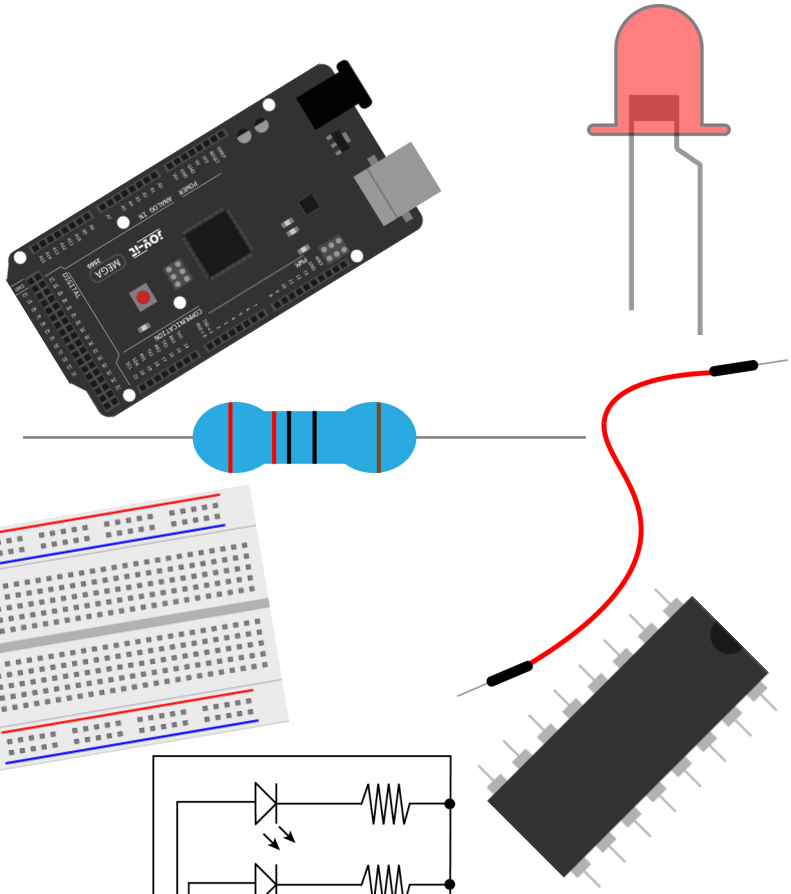
Auf dem seriellen Monitor kann die Temperarturausgabe überwacht werden.

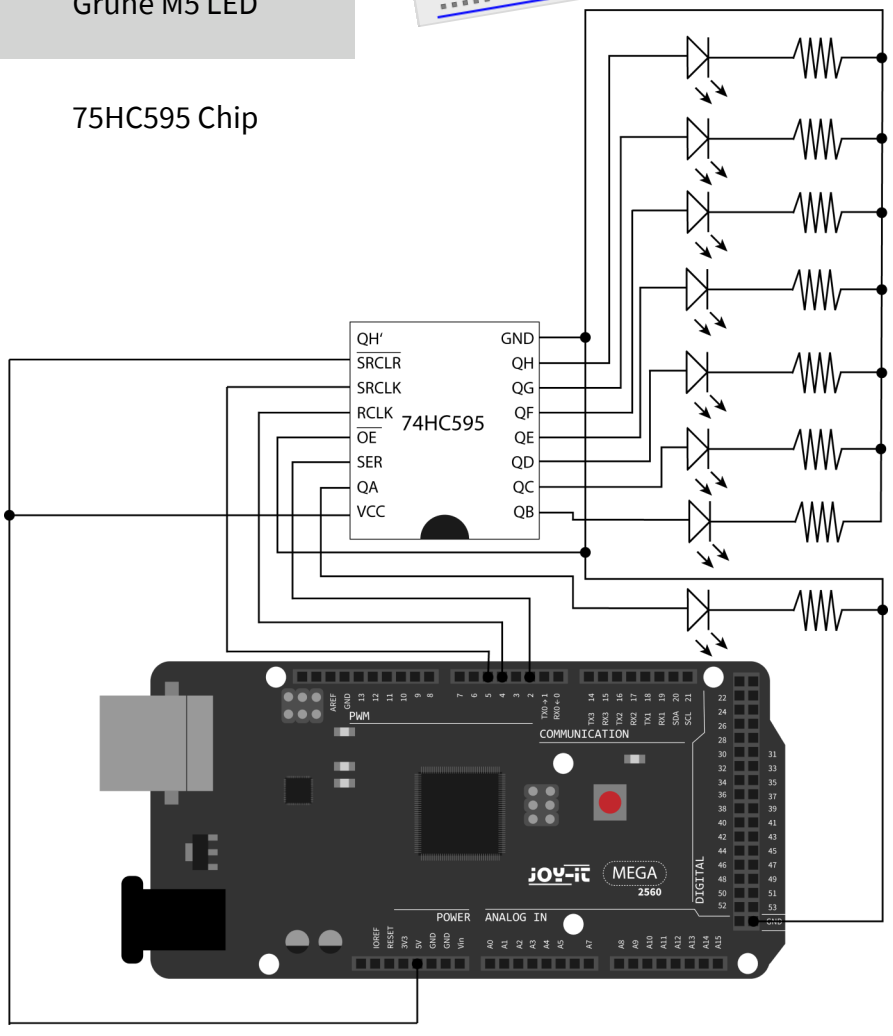
Lektion 17: 74HC595 Schieberegister

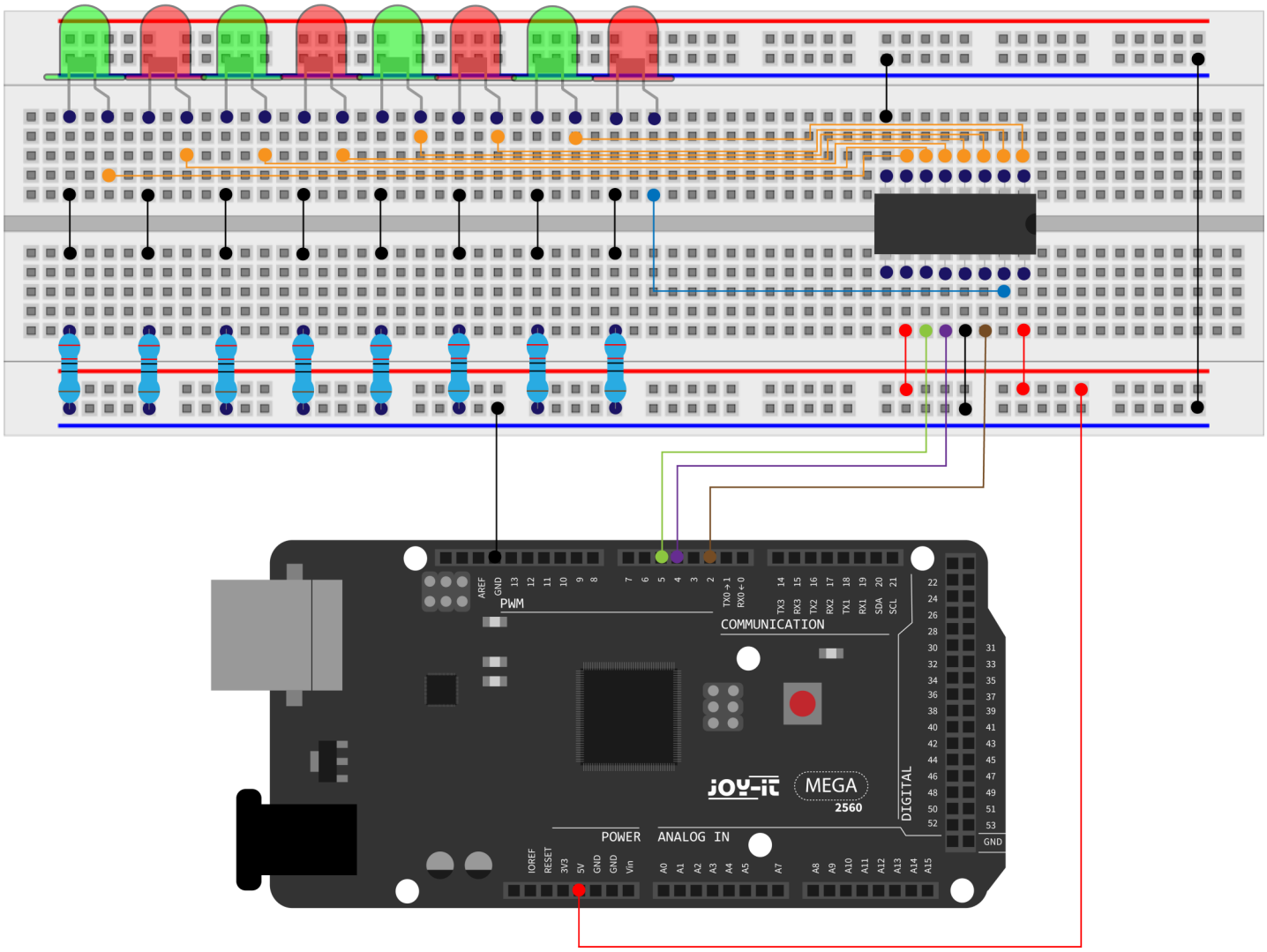


Das 74HC595 ist eine Kombination aus einem 8-Ziffern Schieberegister und Merker. Es ist ausgestattet mit einem Tri-State Ausgang. In diesem Projekt wird der 74HC595 verwendet um 8 LEDs ressourcensparend zu verwenden. Die benötigten I/O Ports reduzieren sich von 8 auf 3 Ports.
Man benötigt:

1x	Mega2560 Platine
1x	USB-Kabel
4x	Rote M5 LED
8x	220 Ω Widerstand
1x	Breadboard
24x	Überbrückungskabel
4x	Grüne M5 LED
1x	75HC595 Chip





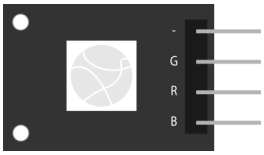



```
int data = 2; // Stellt Pin 14 des 74HC595 als Dateneingang
int clock = 5; // Stellt Pin 11 des 74HC595 als Tackt Pin
int latch = 4; // Stellt Pin 12 des 74HC595 als Ausgang
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;

void setup() {
    pinMode(data, OUTPUT);
    pinMode(clock, OUTPUT);
    pinMode(latch, OUTPUT);
}

void loop() {
    for(int i = 0; i < 256; i++) {
        updateLEDs(i);
        delay(500);
    }
}
```

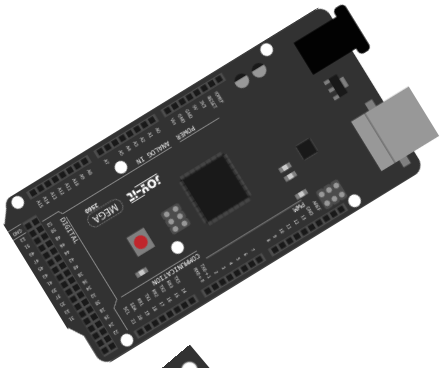
```
void updateLEDs(int value) {
    digitalWrite(latch, LOW);
    shiftOut(data, clock, MSBFIRST, ~value);
    digitalWrite(latch, HIGH); // Verriegeln
}
```

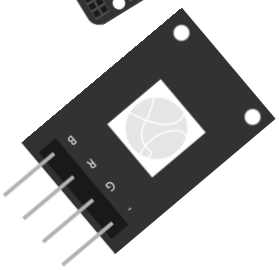
Lektion 18: RGB-LED

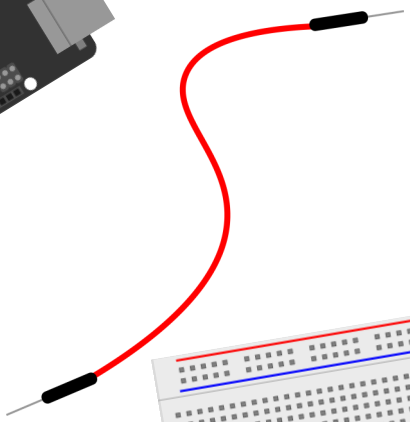
Diese Diode wird durch PWM Signale gesteuert und besitzt ein dreifarbiges System um Farben darzustellen. Das Bauteil kann direkt an die Mega2560 Schnittstellen angeschlossen werden.
Man benötigt:

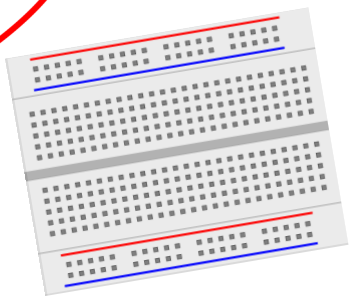


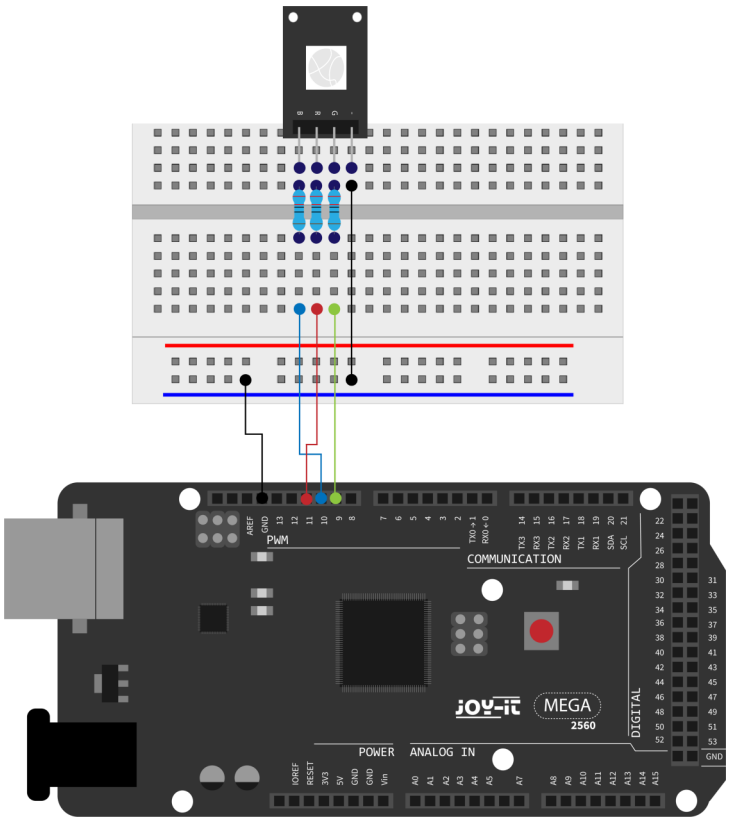
1x	Mega2560 Platine
1x	USB-Kabel
1x	RGB-LED
1x	Breadboard
5x	Überbrückungskabel
3x	220 Ω Widerstand







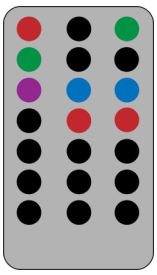




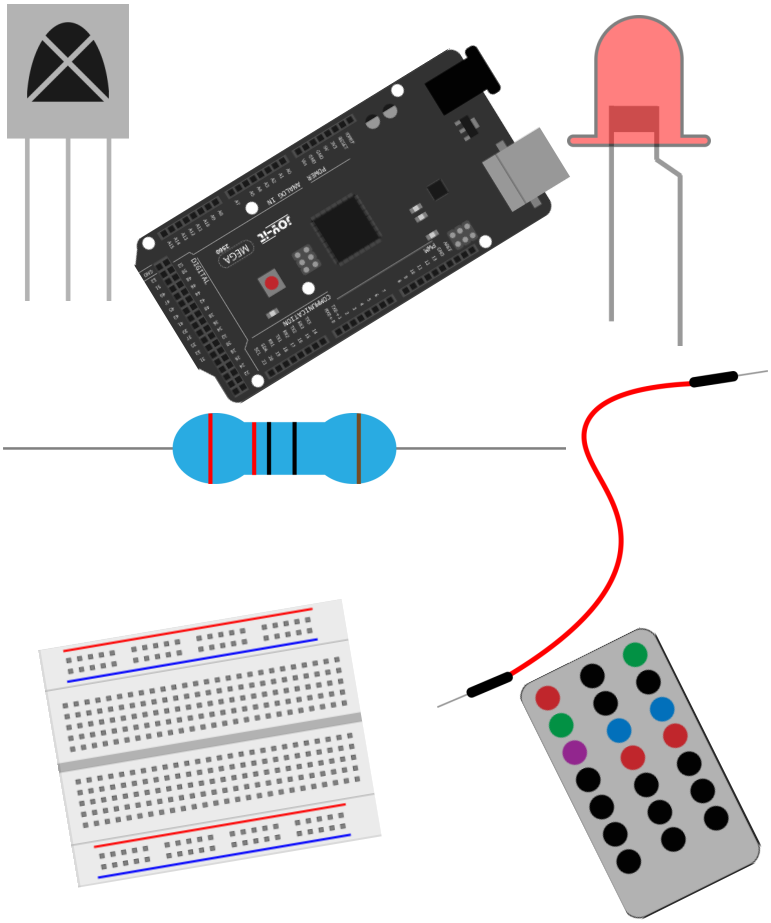

```
int redpin = 11; // Wählt Pin für rote LED aus
int bluepin =10; // Wählt Pin für blaue LED aus
int greenpin =9; // Wählt Pin für grüne LED aus
int val;

void setup() {
    pinMode(redpin, OUTPUT);
    pinMode(bluepin, OUTPUT);
    pinMode(greenpin, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    for(val=255; val>0; val--) {
        analogWrite(11, val);
        analogWrite(10, 255-val);
        analogWrite(9, 128-val);
        delay(1);
    }
    for(val=0; val<255; val++) {
        analogWrite(11, val);
        analogWrite(10, 255-val);
        analogWrite(9, 128-val);
        delay(1);
    }
    Serial.println(val, DEC);
}
```

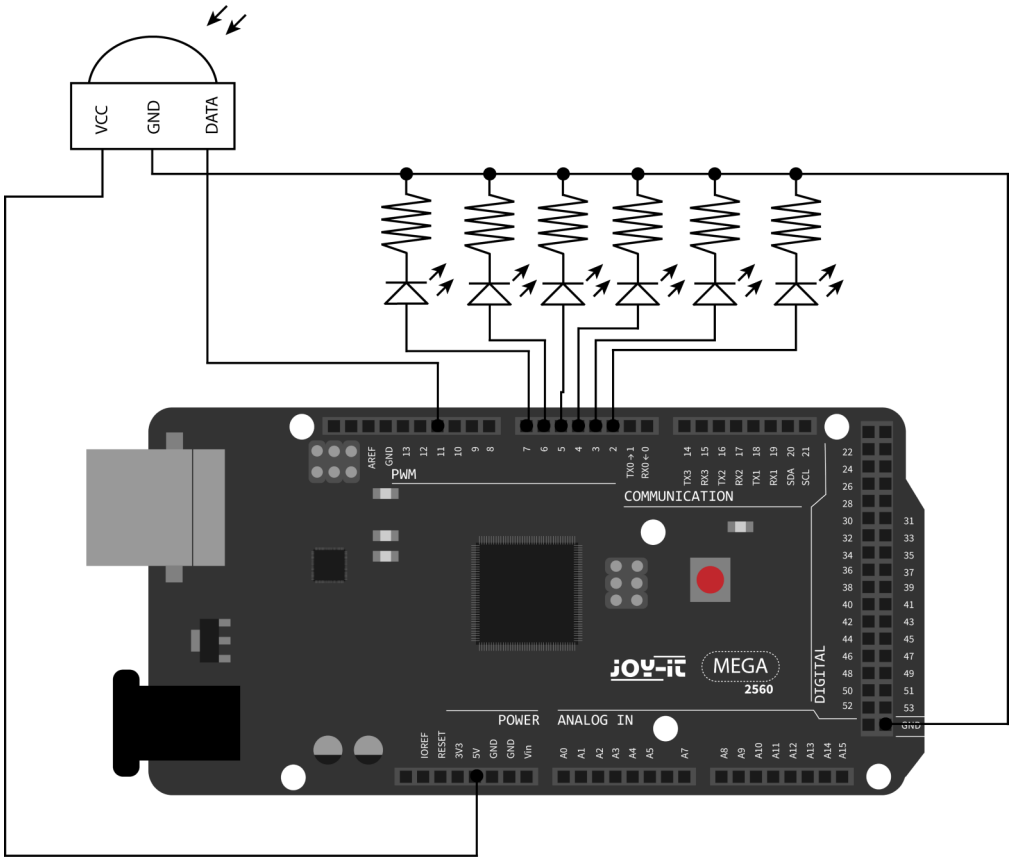
Lektion 19: Infrarot-Fernbedienung

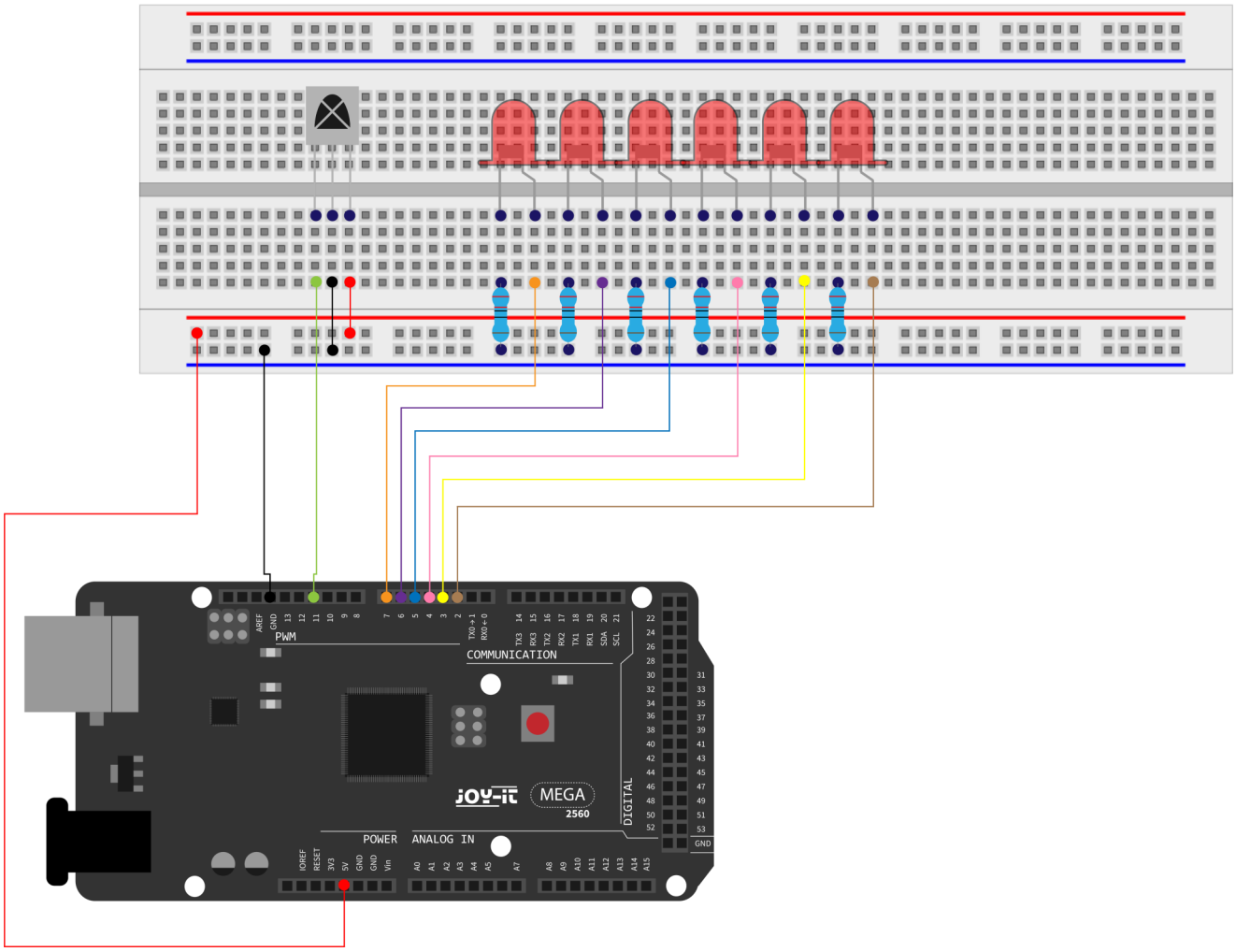
Der IR-Empfänger wandelt das ankommende Lichtsignal in ein schwaches, elektrisches Signal um. Um den Code einer Fernbedienung zu dekodieren, ist es nötig die Kodierungsmethode zu kennen. In diesem Projekt wird dafür das NEC-Protokoll verwendet.
Hierzu benötigen Sie:



1x	Mega2560 Platine
1x	USB-Kabel
6x	M5 LED
6x	220 Ω Widerstand
1x	Breadboard
11x	Überbrückungskabel
1x	Infrarot-Empfänger
1x	Infrarot-Fernbedienung







Der folgende Code benötigt die Bibliothek ***IRremote***, welche sie in der Arduino IDE herunterladen können, unter ***Sketch → Bibliothek einbinden → Bibliothek verwalten...*** . Dort können Sie die mittels der Suchleiste diese Bibliothek finden und installieren. Starten Sie danach Ihre IDE neu.

Der folgende Code wird mittels Infrarot ein Signal empfangen, entschlüsseln und im seriellen Monitor ausgeben. Wenn Sie die mitgelieferte Fernbedienung verwenden, wird Ihnen auch die gedrückte Taste der Fernbedienung korrekt ausgegeben. Beachten Sie dabei, dass bei Verwendung einer anderen Fernbedienung es dabei zur Ausgabe von falschen oder gar keinen Tasten kommen kann. Mit der Fernbedienung lassen sich auch die 6 LEDs ansteuern, also an- und ausmachen.

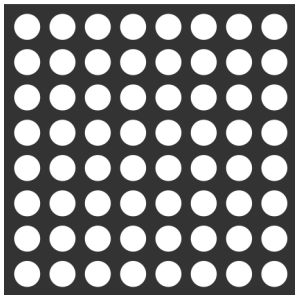
```
#include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
```

```
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB04F;
IRrecv irrecv(RECV_PIN);
decode_results results;

void dump(decode_results *results) {
    int count = results->rawlen;
    if (results->decode_type == UNKNOWN){
        Serial.println("Could not decode message");
    }
    else {
        if (results->decode_type == NEC){
            Serial.print("Decoded NEC: ");
        }
        else if (results->decode_type == SONY) {
            Serial.print("Decoded SONY: ");
        }
        else if (results->decode_type == RC5) {
            Serial.print("Decoded RC5: ");
        }
        else if (results->decode_type == RC6) {
            Serial.print("Decoded RC6: ");
        }
        Serial.print(results->value, HEX);
        Serial.print(" (");
        Serial.print(results->bits, DEC);
        Serial.println(" bits)");


        if(results->value == 0xFFA25D)
            Serial.println("Button: Power");
        else if(results->value == 0xFF629D)
            Serial.println("Button: Mode");
        else if(results->value == 0xFFE21D)
            Serial.println("Button: Mute");
        else if(results->value == 0xFF22DD)
            Serial.println("Button: Play/Stop");
        else if(results->value == 0xFF02FD)
            Serial.println("Button: <<");
```

```
        else if(results->value == 0xFFC23D)
            Serial.println("Button: >>");
        else if(results->value == 0xFFE01F)
            Serial.println("Button: EQ");
        else if(results->value == 0xFFA857)
            Serial.println("Button: -");
        else if(results->value == 0xFF906F)
            Serial.println("Button: +");
        else if(results->value == 0xFF6897)
            Serial.println("Button: 0");
        else if(results->value == 0xFF9867)
            Serial.println("Button: Reload");
        else if(results->value == 0xFFB04F)
            Serial.println("Button: U/SD");
        else if(results->value == 0xFF30CF)
            Serial.println("Button: 1");
        else if(results->value == 0xFF18E7)
            Serial.println("Button: 2");
        else if(results->value == 0xFF7A85)
            Serial.println("Button: 3");
        else if(results->value == 0xFF10EF)
            Serial.println("Button: 4");
        else if(results->value == 0xFF38C7)
            Serial.println("Button: 5");
        else if(results->value == 0xFF5AA5)
            Serial.println("Button: 6");
        else if(results->value == 0xFF42BD)
            Serial.println("Button: 7");
        else if(results->value == 0xFF4AB5)
            Serial.println("Button: 8");
        else if(results->value == 0xFF52AD)
            Serial.println("Button: 9");
    }
}
void setup() {
pinMode(RECV_PIN, INPUT);
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);
    pinMode(LED6, OUTPUT);
    pinMode(13, OUTPUT);
    Serial.begin(9600);
    irrecv.enableIRIn(); // Start the receiver
}
```

```
int on = 0;
unsigned long last = millis();

void loop() {
     if (irrecv.decode(&results)){
          if (millis() - last > 250){
               on = !on; // digitalWrite(8, on ? HIGH : LOW);
               digitalWrite(13, on ? HIGH : LOW);
               dump(&results);
          }


          if (results.value == on1 )
               digitalWrite(LED1, HIGH);
          if (results.value == off1 )
               digitalWrite(LED1, LOW);
          if (results.value == on2 )
               digitalWrite(LED2, HIGH);
          if (results.value == off2 )
               digitalWrite(LED2, LOW);
          if (results.value == on3 )
               digitalWrite(LED3, HIGH);
          if (results.value == off3 )
               digitalWrite(LED3, LOW);
          if (results.value == on4 )
               digitalWrite(LED4, HIGH);
          if (results.value == off4 )
               digitalWrite(LED4, LOW);
          if (results.value == on5 )
               digitalWrite(LED5, HIGH);
          if (results.value == off5 )
               digitalWrite(LED5, LOW);
          if (results.value == on6 )
               digitalWrite(LED6, HIGH);
          if (results.value == off6 )
               digitalWrite(LED6, LOW);
          last = millis();
          irrecv.resume();
     }
}
```

Lektion 20: 8x8 LED-Matrix

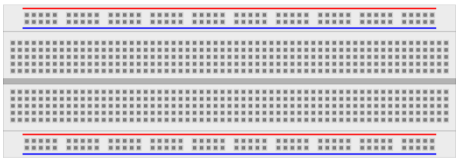


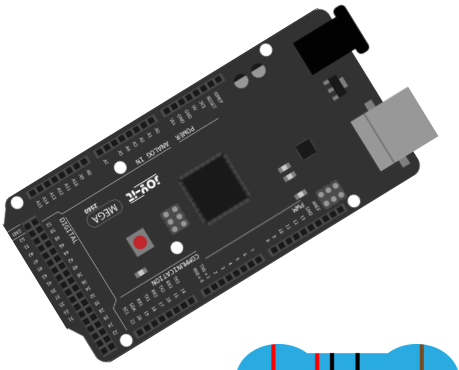
Die 8 x 8 LED Matrix besteht aus 64 LEDs. Jede LED ist im Bereich des Schnittpunktes von Reihe und Spalte platziert. Wenn der Pegel für eine Reihe 1 ist und der Pegel der entsprechenden Spalte 0 ist, wird die LED in deren Schnittpunkt angehen.

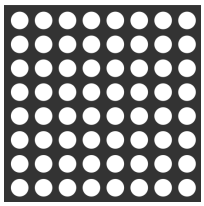
Beispiel:
Wenn Sie die erste LED anschalten wollen, stellen Sie Pin9 auf "HIGHPEGEL" und Pin 13 auf "LOWPEGEL" ein.
Falls Sie die erste Reihe einschalten wollen, stellen Sie Pin 9 auf "HIGHPEGEL" und die Pins 13, 3, 4, 10, 11, 15 und 16 auf "LOWPEGEL".
Um die erste Spalte einzuschalten stellen Sie Pin 13 auf "LOWPEGEL" und die Pins 9, 14, 8, 12, 1, 7, 2, 5 auf "HIGHPEGEL".

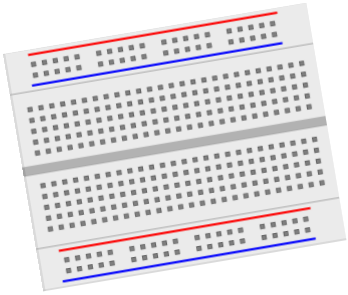
Sie benötigen für den folgenden Code:

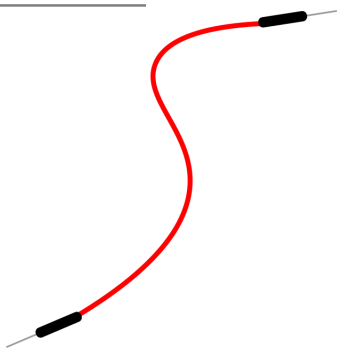
1x	Mega2560 Platine
1x	USB-Kabel
6x	LED Matrix
8x	220 Ω Widerstand
2x	Breadboard
16x	Überbrückungskabel

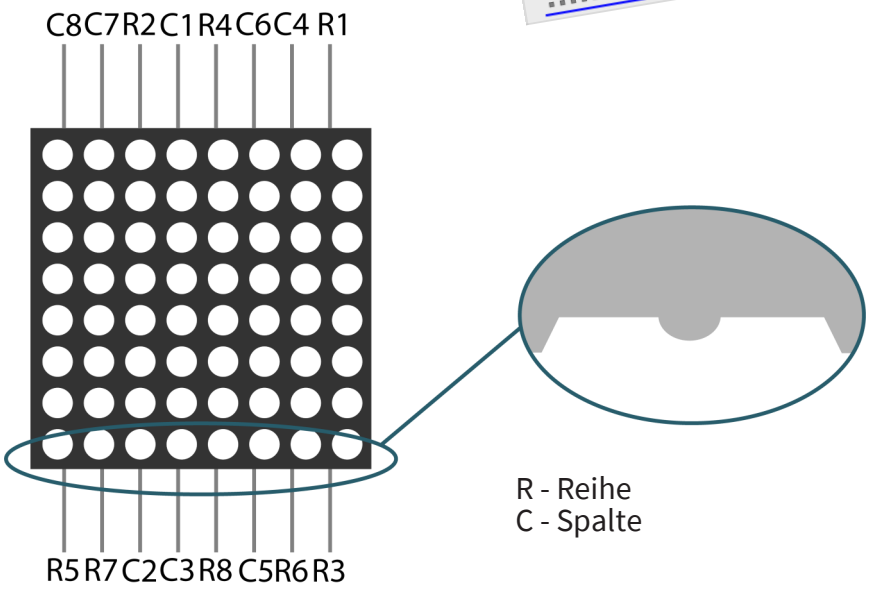


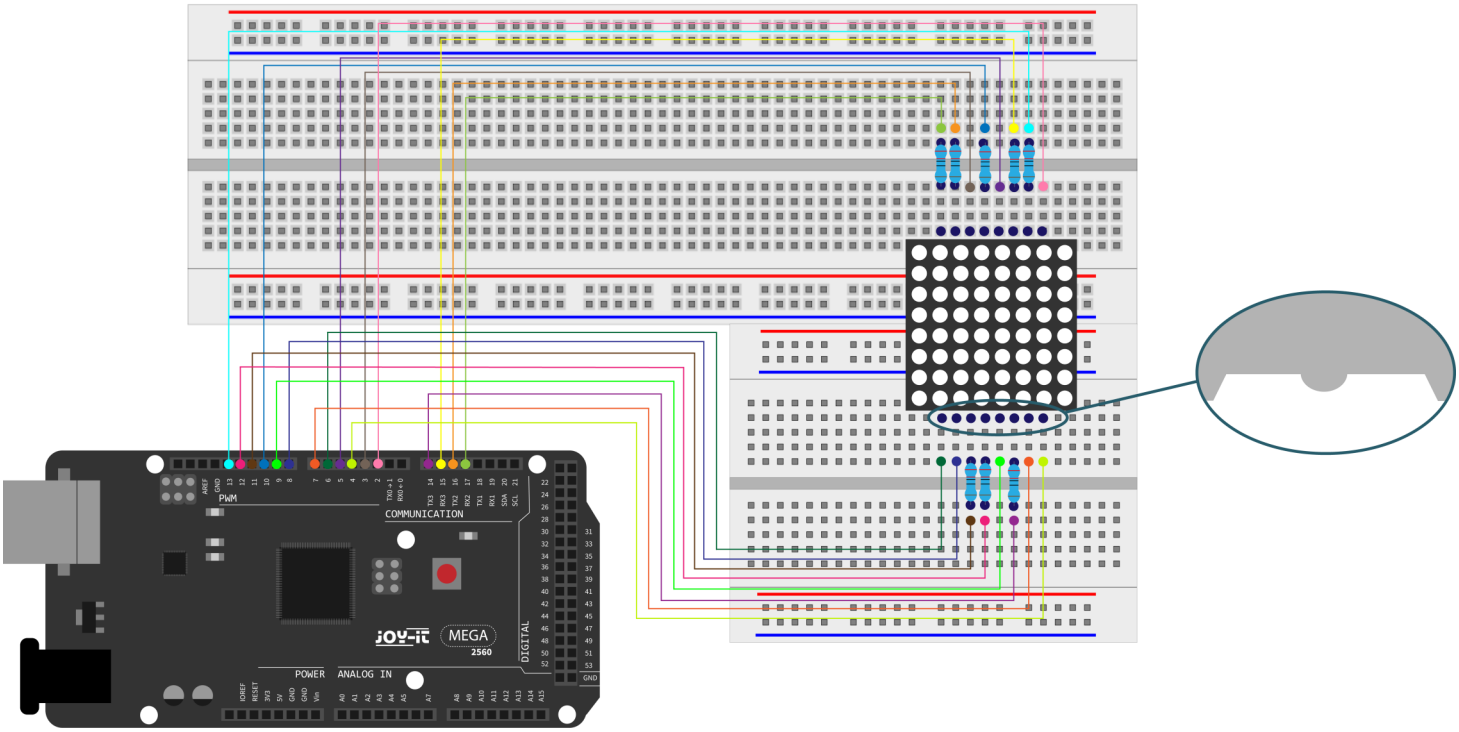












Achten Sie darauf das die LED-Matrix richtig angeschlossen ist. Unter der Matrix befindet sich eine Auswölbung (Abbildung) an welcher Sie sich orientieren können.

```
// Setzen eines Arrays um Buchstaben von „0“ zu speichern
unsigned char Text[]={0x00,0x1c,0x22,0x22,0x22,0x22,0x22,0x1c};

void Draw_point(unsigned char x,unsigned char y) {  // Zeichne-Punkt Funktion
    clear_();
    digitalWrite(x+2, HIGH);
    digitalWrite(y+10, LOW);
    delay(1);
}

void show_num(void) { // Anzeige-Funktion, ruft Zeichne-Punkt Funktion auf
    unsigned char i,j,data;
    for(i=0;i<8;i++) {
        data=Text[i];
        for(j=0;j<8;j++) {
            if(data & 0x01)
                Draw_point(j,i);
            data>>=1;
        }
    }
}
```

```
void setup(){
    int i = 0 ;
    for(i=2;i<18;i++) {
        pinMode(i, OUTPUT);
    }
    clear_();
}

void loop() {
    show_num();
}

void clear_(void) { // Leert Bildschirm
    for(int i=2;i<10;i++)
        digitalWrite(i, LOW);
    for(int i=0;i<8;i++)
        digitalWrite(i+10, HIGH);
}
```

6. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

7. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: 30.07.2020