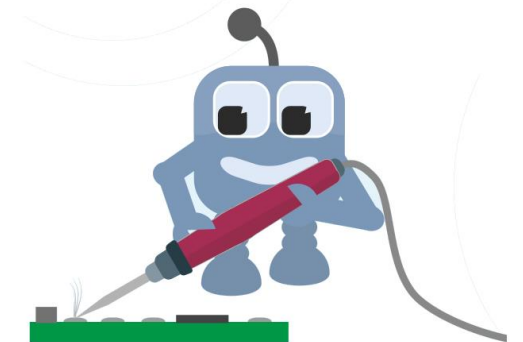
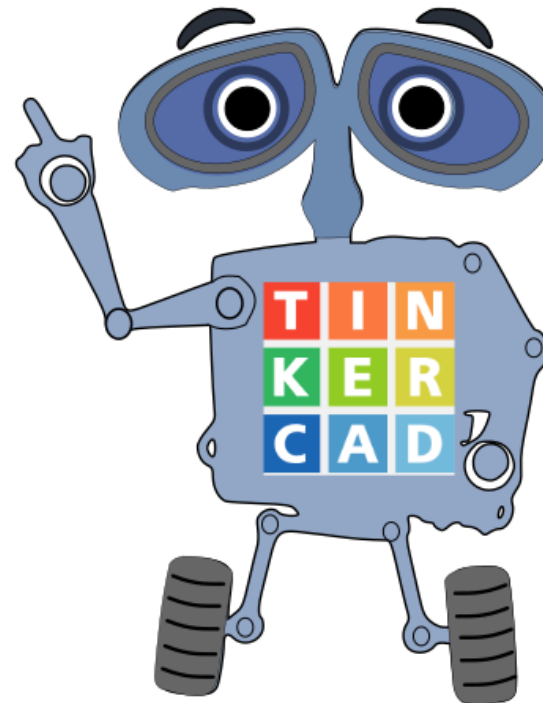
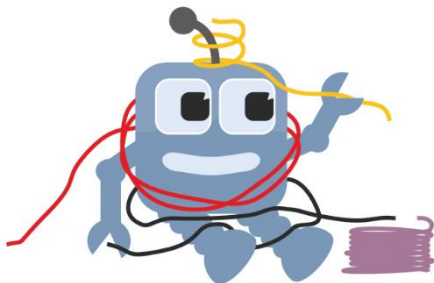
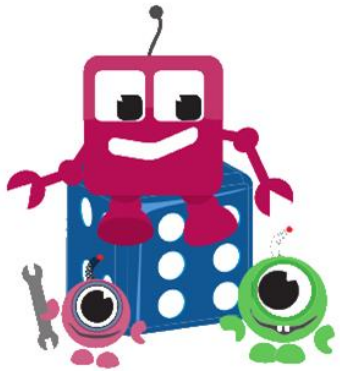




## Tinkercad Workshop



# Makers Monday



# Tinkercad Workshop



Tinkercad Anmeldung

## 3E5 9TD Z6K

**Willkommen zurück**

Wie verwendest du Tinkercad?

In der Schule

Lehrkräfte

**Schüler mit Klassencode**

Schülerkonten

Auf eigene Faust

Persönliche Konten

Du hast noch kein Konto?  
[Tinkercad beitreten](#)

Willkommen bei  
**Schaltkreise 1**

**Mit Spitzname beitreten**

E-Mail oder Benutzername

Other providers

Apple Google Microsoft Facebook

Du trittst keiner Klasse bei?  
[Zurück](#)

Willkommen bei  
**Schaltkreise 1**

Dein Spitzname?

01oberlab

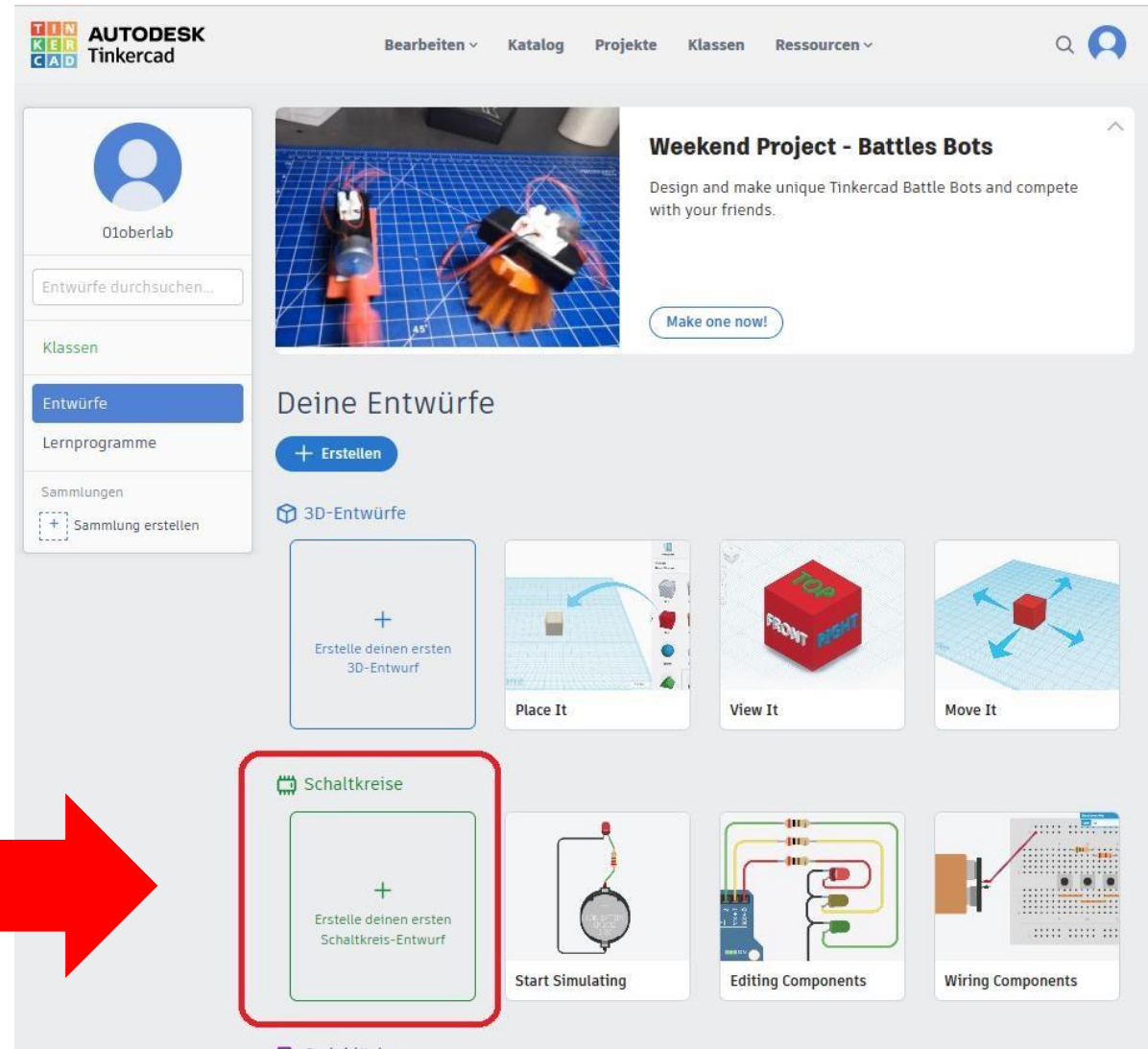
**Das bin ich!**

Du trittst keiner Klasse bei?  
[Zurück](#)

**01oberlab**  
**02oberlab**  
...  
**07oberlab**  
**08oberlab**

# Tinkercad Workshop

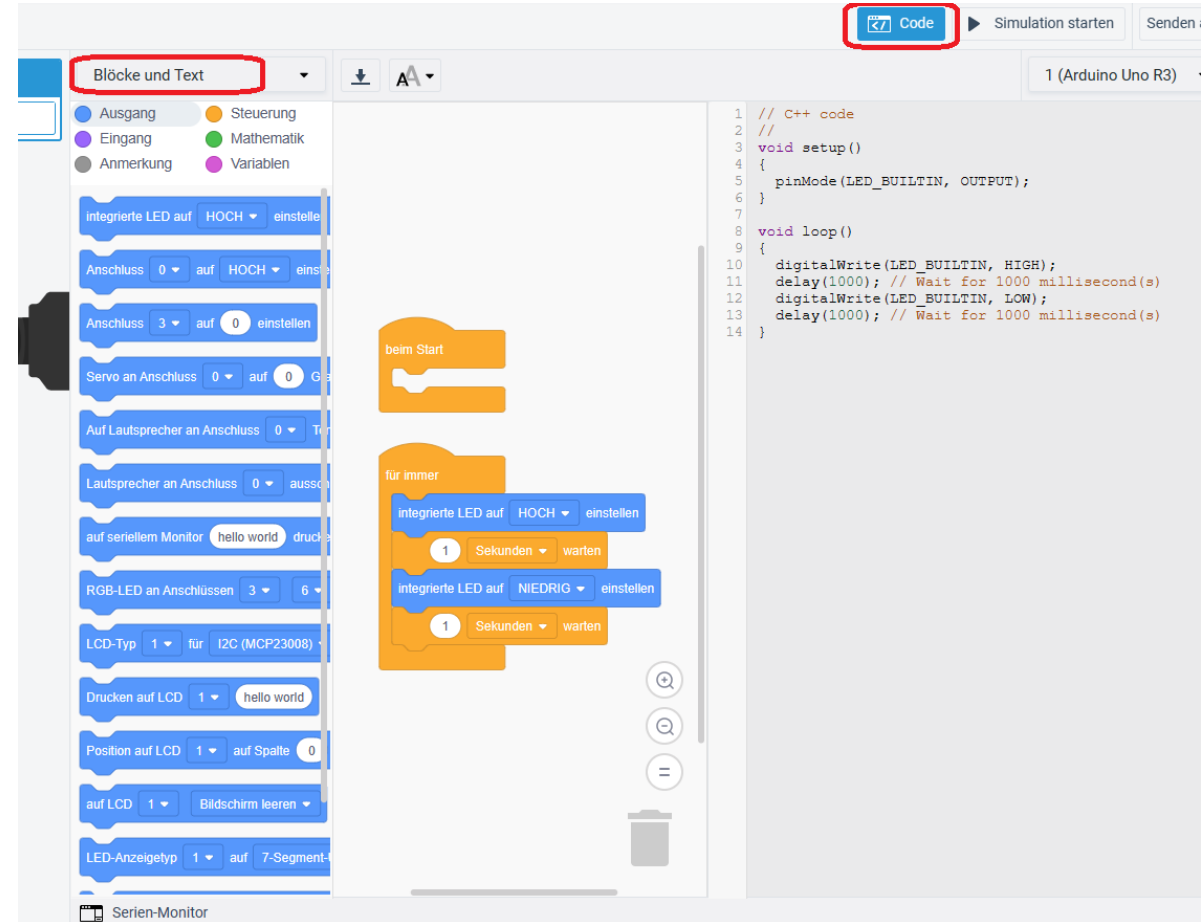
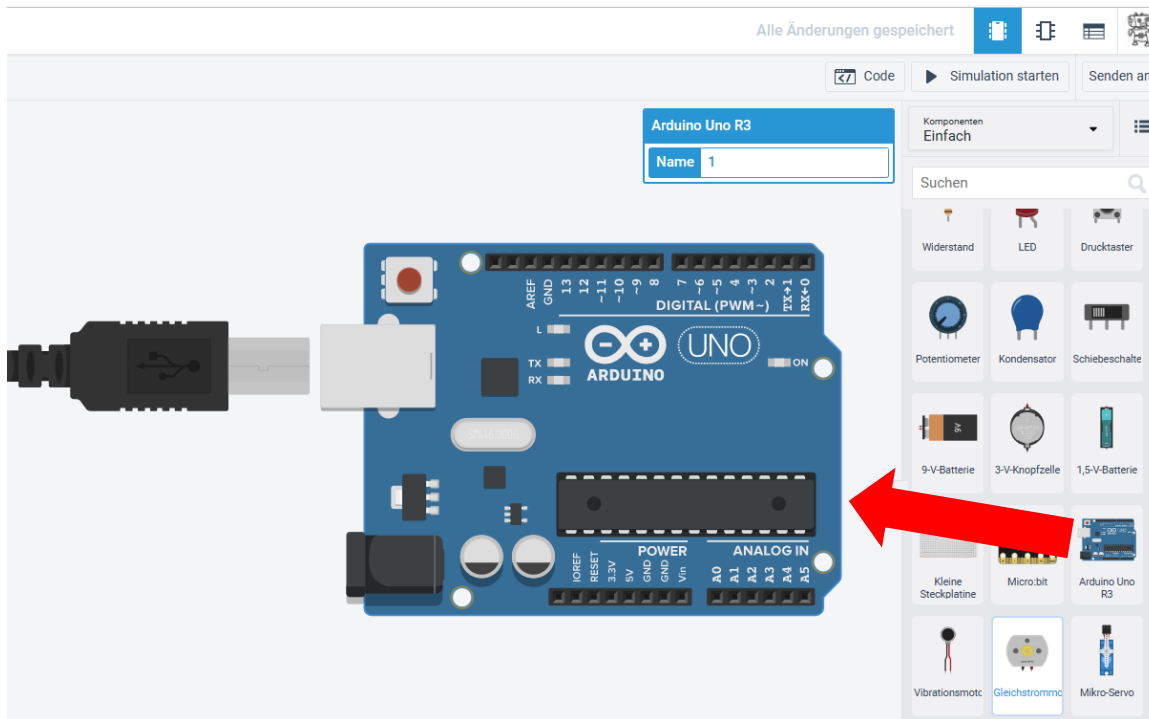
## Tinkercad Anmeldung



# Tinkercad Workshop



Mein erstes Programm:

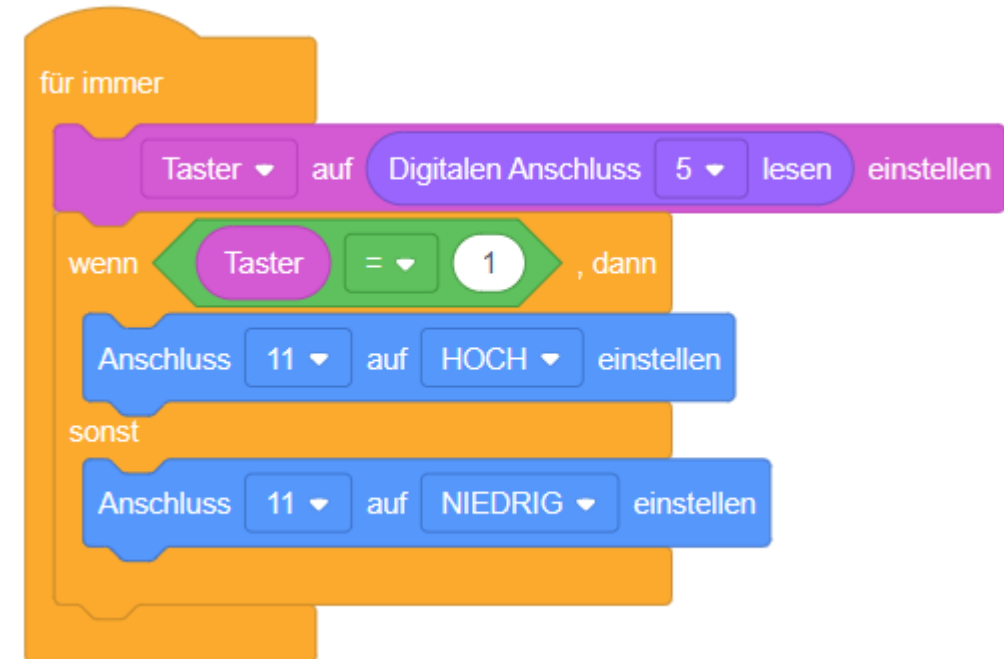
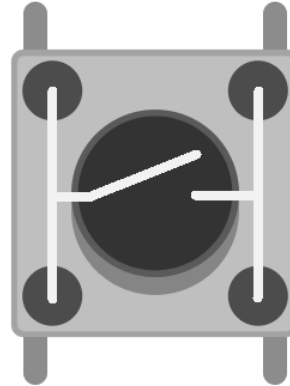
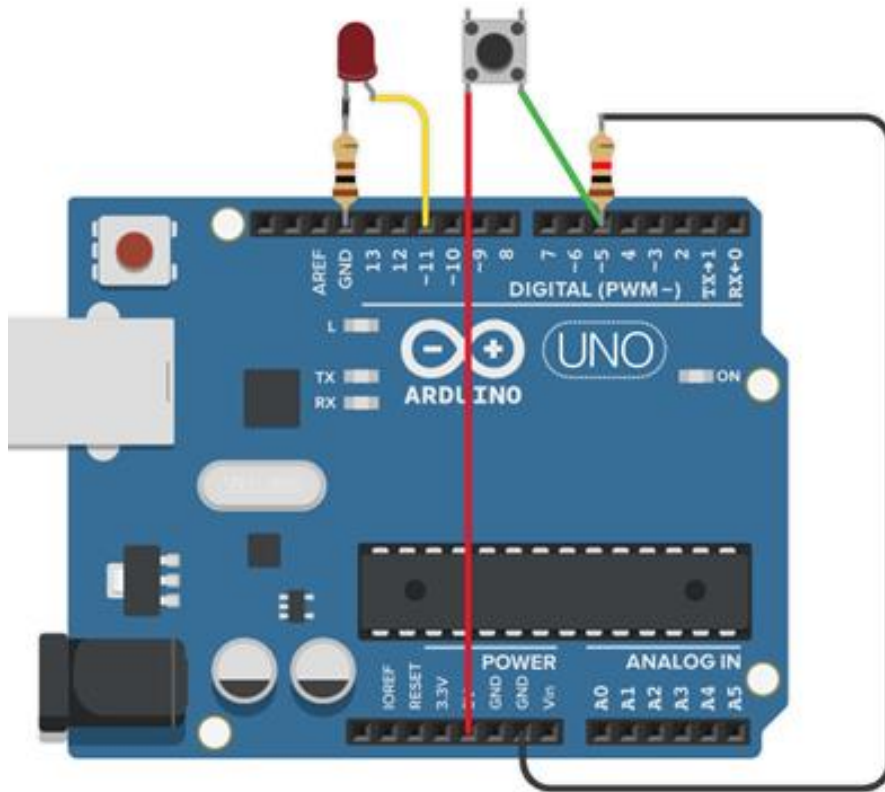


# Tinkercad Workshop



## LED und Taster

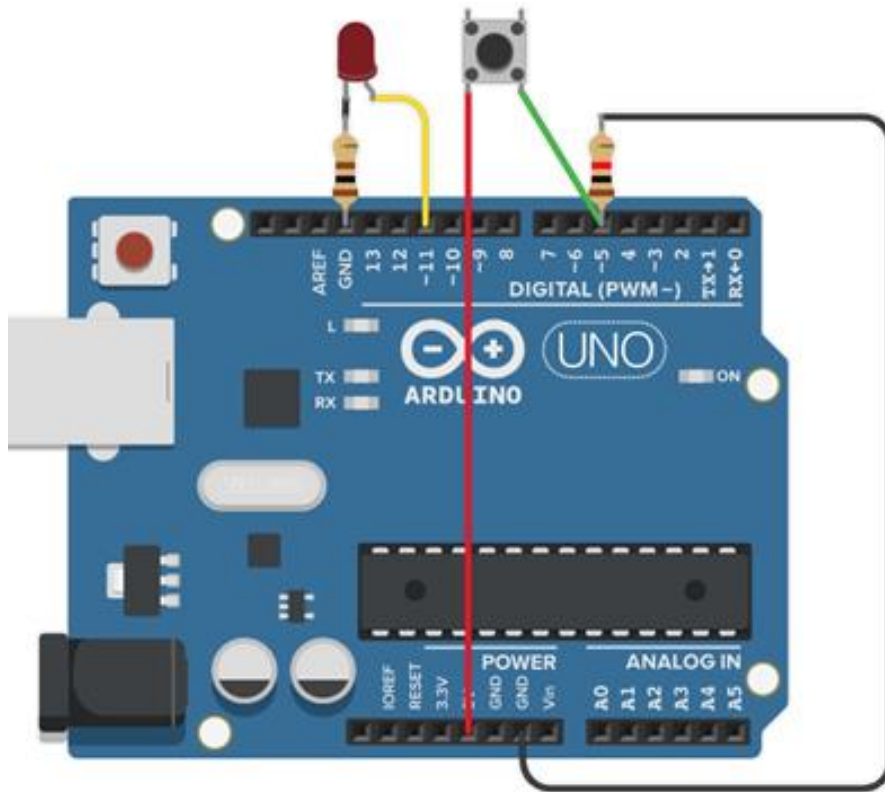
Mit dem Betätigen des Tasters leuchtet die LED.



# Tinkercad Workshop



## LED und Taster mit Schaltfunktion

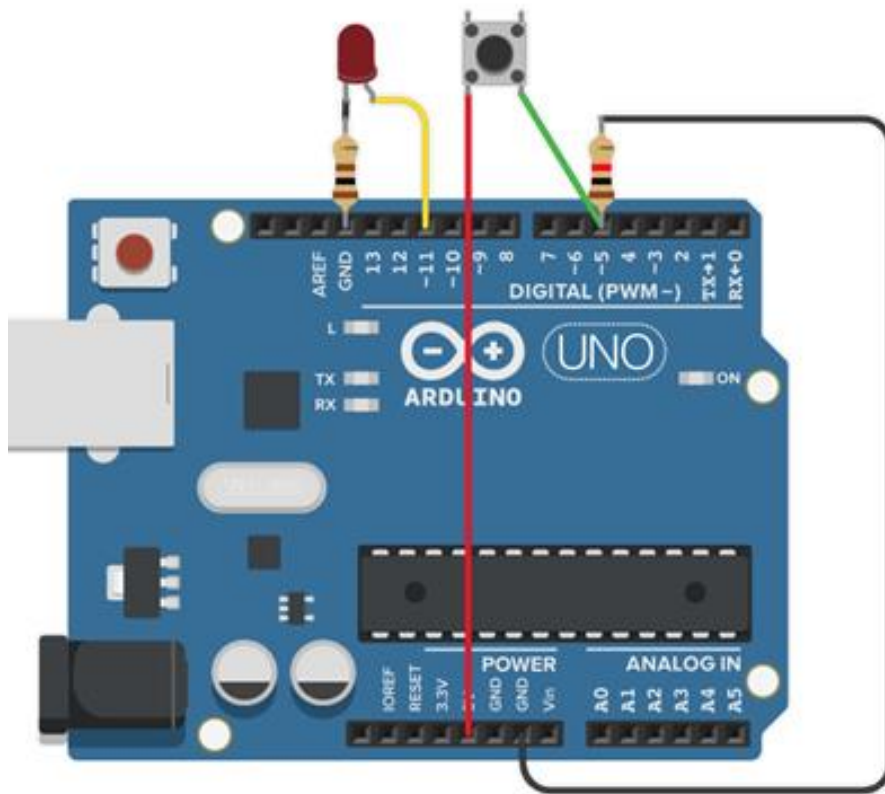


Die LED soll mit dem Taster ein- und ausschaltbar sein.  
Erstelle das Programm in Blöcke.

# Tinkercad Workshop



## LED und Taster mit Schaltfunktion

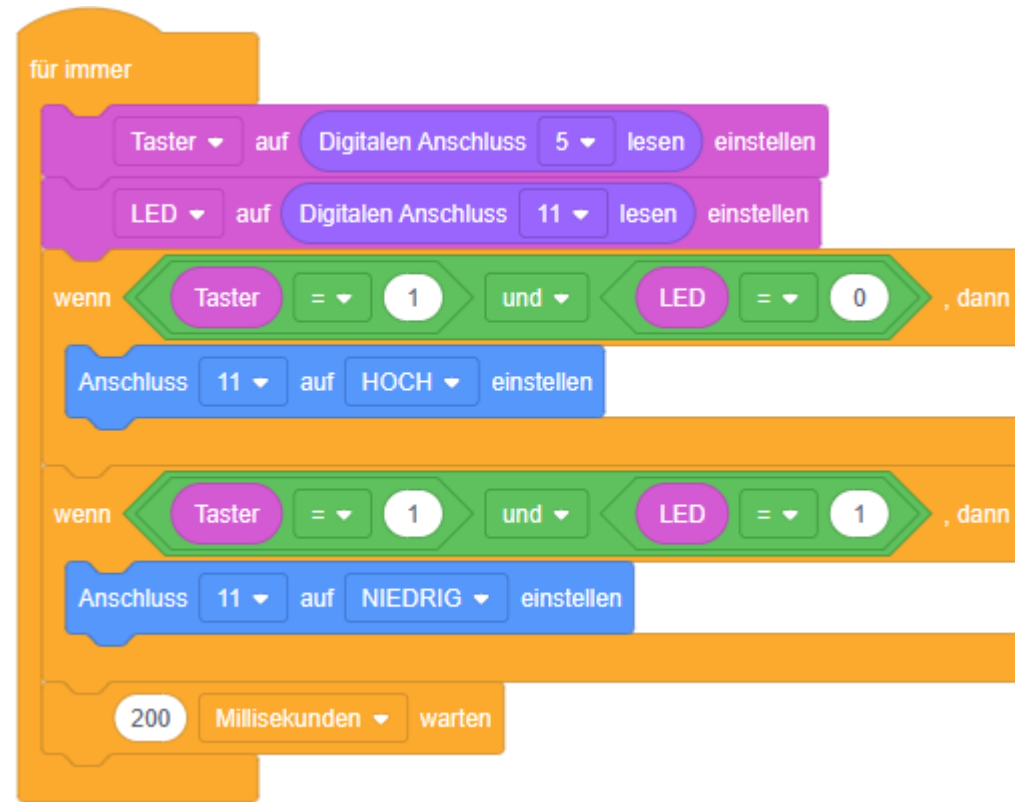
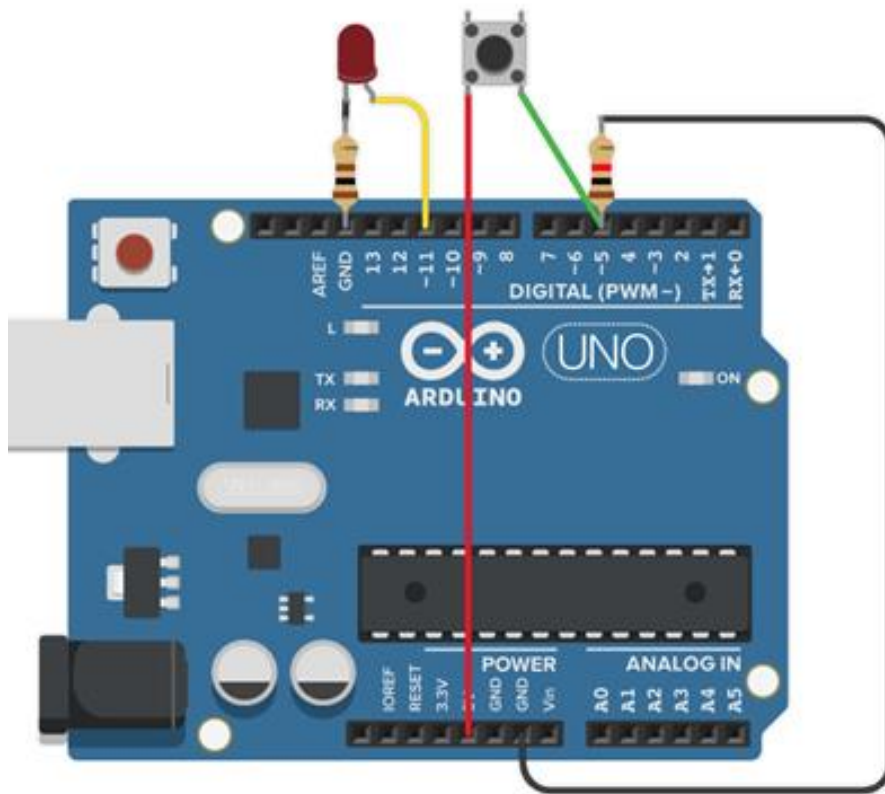


Ergänze die Programmzeilen



# Tinkercad Workshop

## LED und Taster mit Schaltfunktion





# Tinkercad Workshop

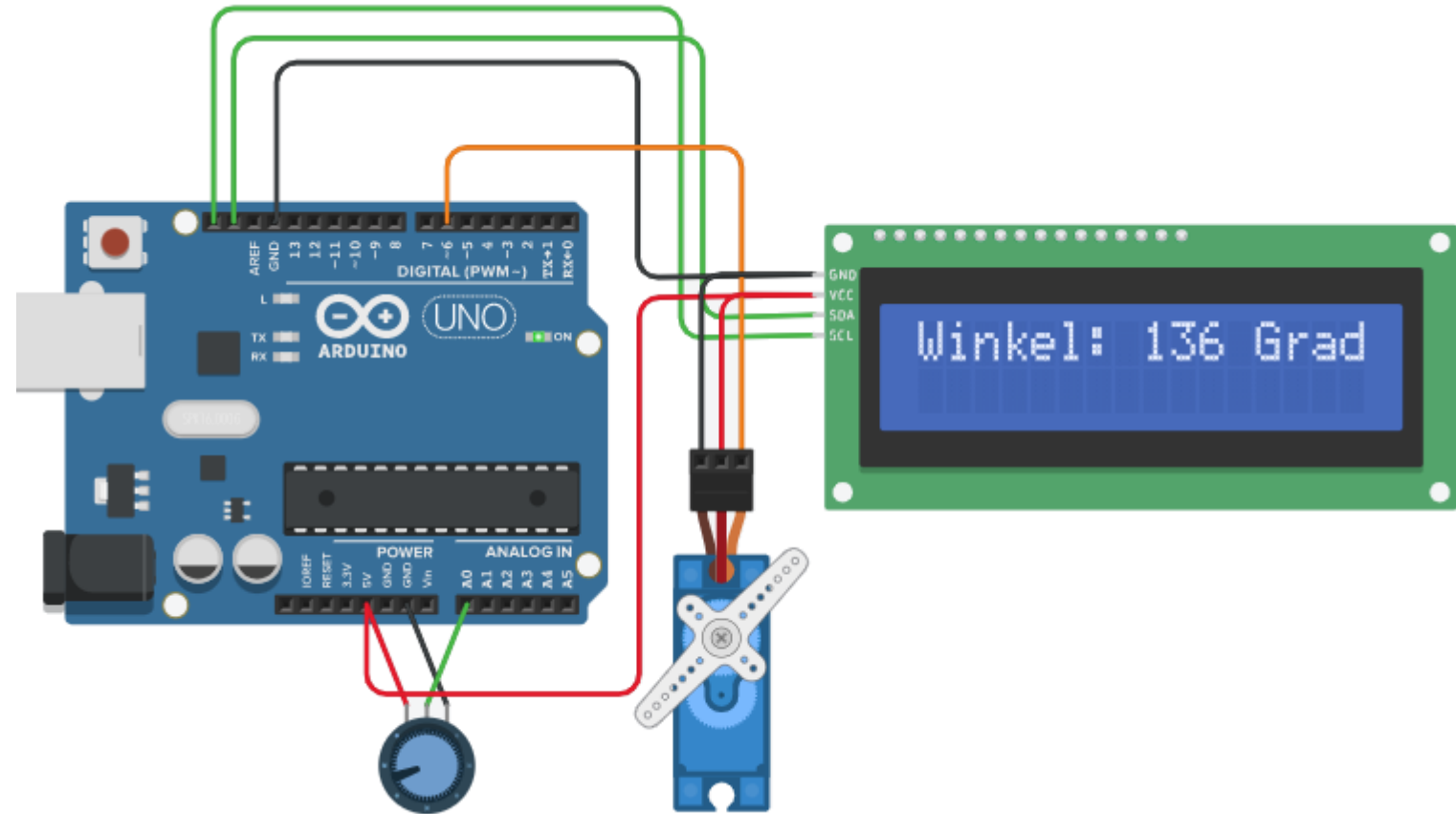


## Servo-Motor mit Poti und Display

Der Servo-Motor wird mit dem Poti im Bereich zwischen  $0^\circ$  und  $180^\circ$  angesteuert.

Das LCD-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Potiwert und der Winkel angezeigt.



Serien-Monitor	
Potwert = 777	Winkel: 136
Potwert = 941	Winkel: 165
Potwert = 859	Winkel: 151
Potwert = 737	Winkel: 129
Potwert = 777	Winkel: 136
Potwert = 777	Winkel: 136
Potwert = 777	Winkel: 136
Potwert = 777	Winkel: 136

# Tinkercad Workshop



## Servo-Motor mit Poti und Display

Der Servo-Motor wird mit dem Poti im Bereich zwischen 0° und 180° angesteuert.

Das LCD-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Potiwert und der Winkel angezeigt.



Ergänze die Programmzeilen

# Tinkercad Workshop

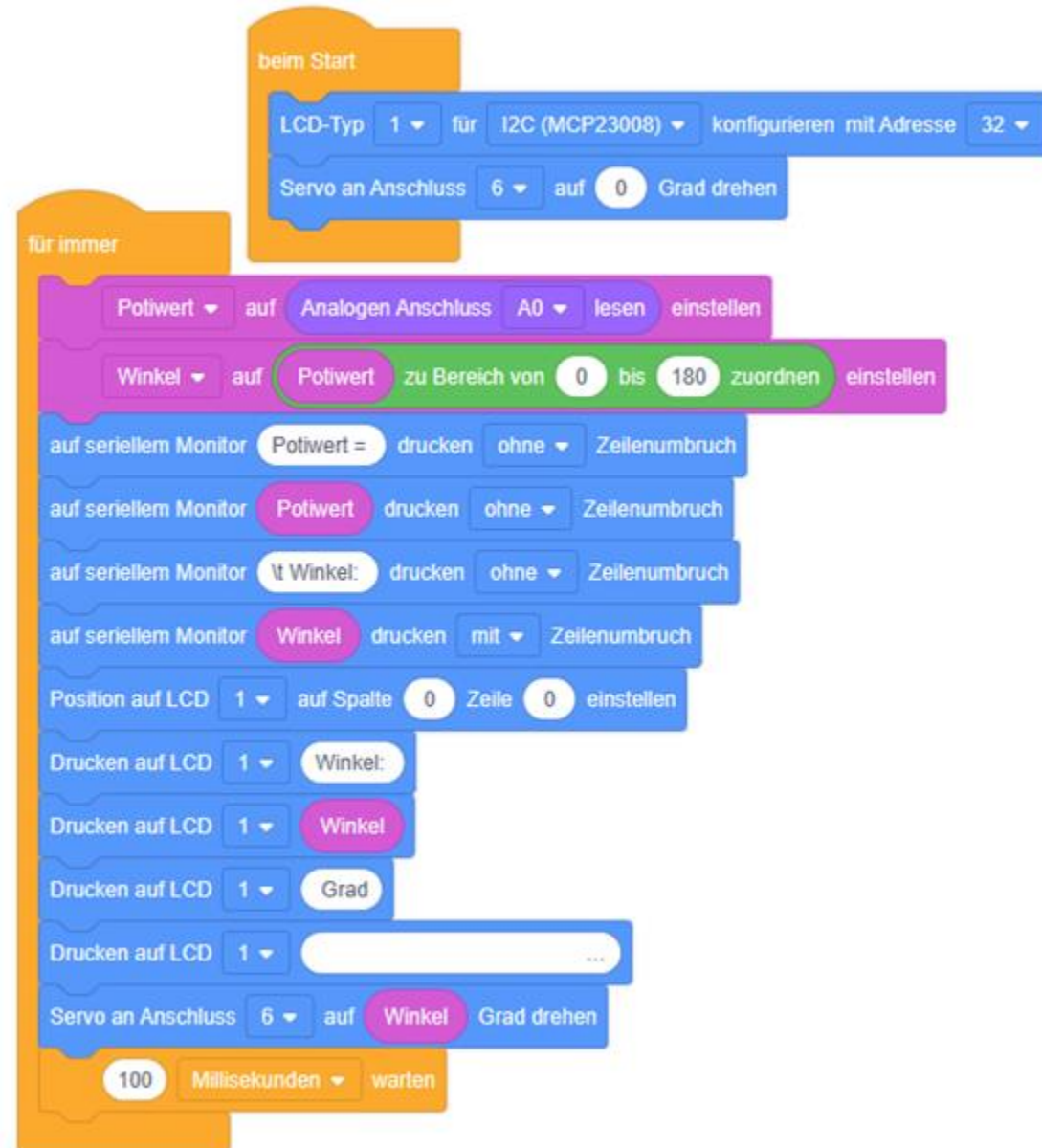


## Servo-Motor mit Poti und Display

Der Servo-Motor wird mit dem Poti im Bereich zwischen 0° und 180° angesteuert.

Das LCD-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Potiwert und der Winkel angezeigt.



# Tinkercad Workshop

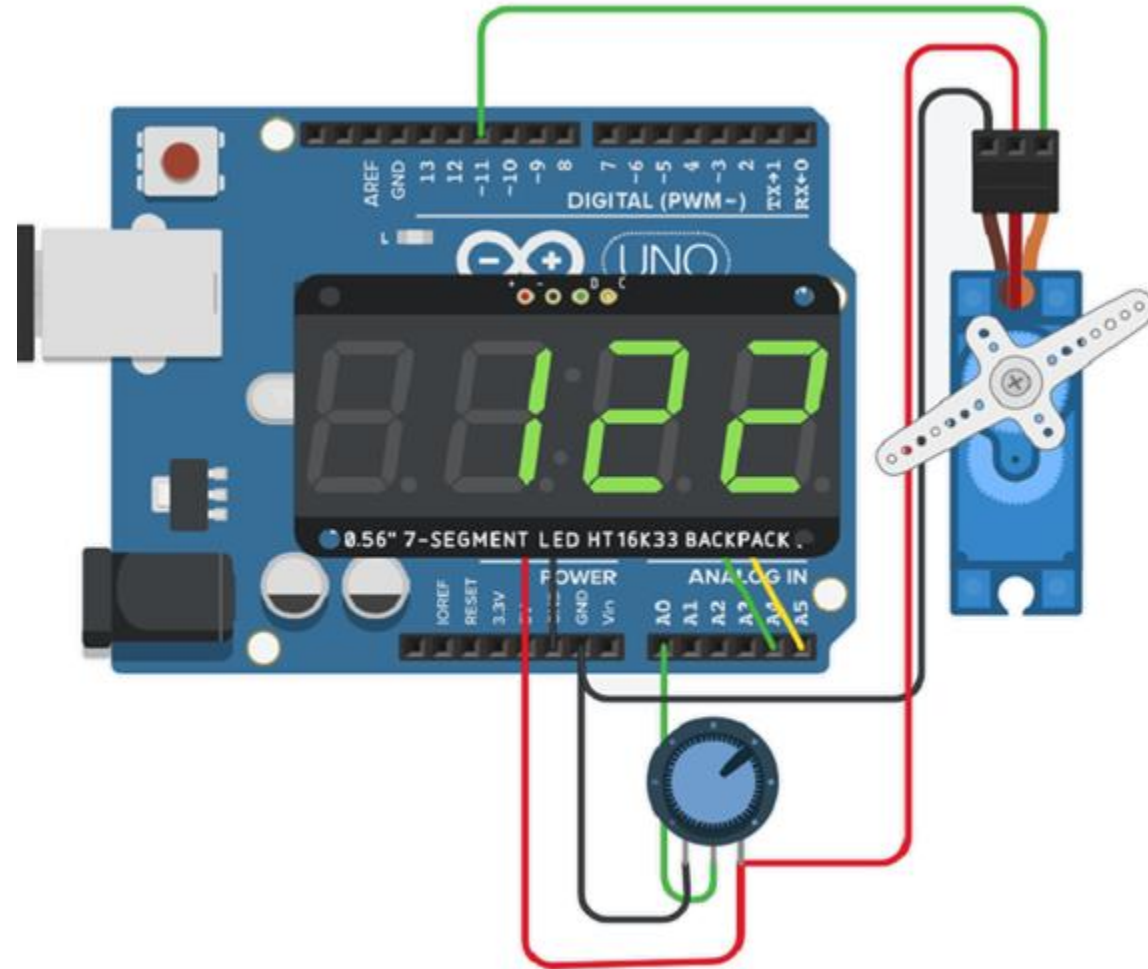


## Servo-Motor mit Poti und Siebensegment-Anzeige

Der Servo-Motor wird mit dem Poti im Bereich zwischen  $0^\circ$  und  $180^\circ$  angesteuert.

Das Sieben-Segment-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Winkel angezeigt.



### Serien-Monitor

```
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad  
Winkel: 122 Grad
```

# Tinkercad Workshop



## Servo-Motor mit Poti und Siebensegment-Anzeige

Der Servo-Motor wird mit dem Poti im Bereich zwischen 0° und 180° angesteuert.

Das Sieben-Segment-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Winkel angezeigt.



Ergänze die Programmzeilen

# Tinkercad Workshop

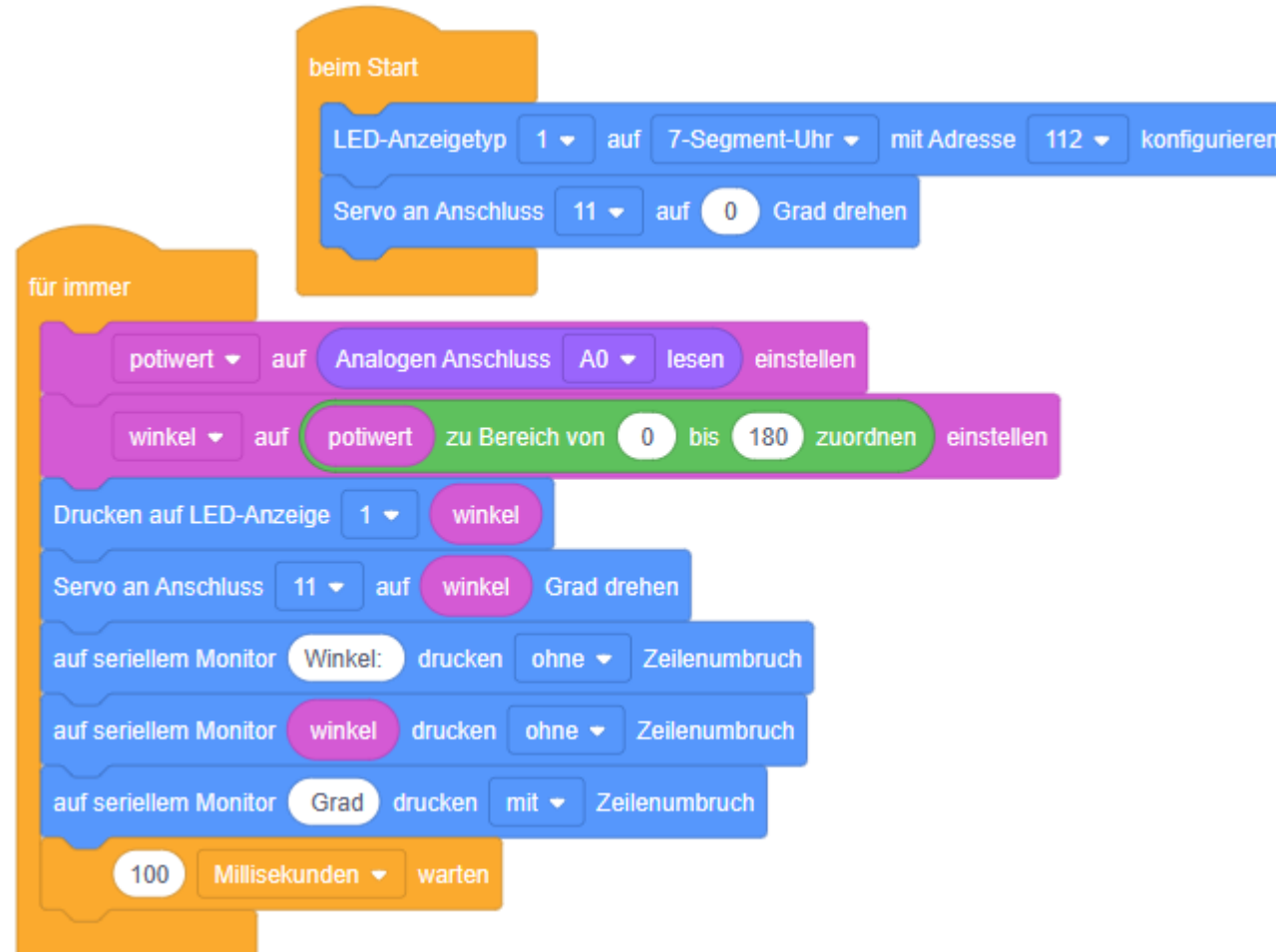


## Servo-Motor mit Poti und Siebensegment-Anzeige

Der Servo-Motor wird mit dem Poti im Bereich zwischen 0° und 180° angesteuert.

Das Sieben-Segment-Display zeigt den Winkel in Grad an.

Auf den Seriellen Monitor wird der Winkel angezeigt.



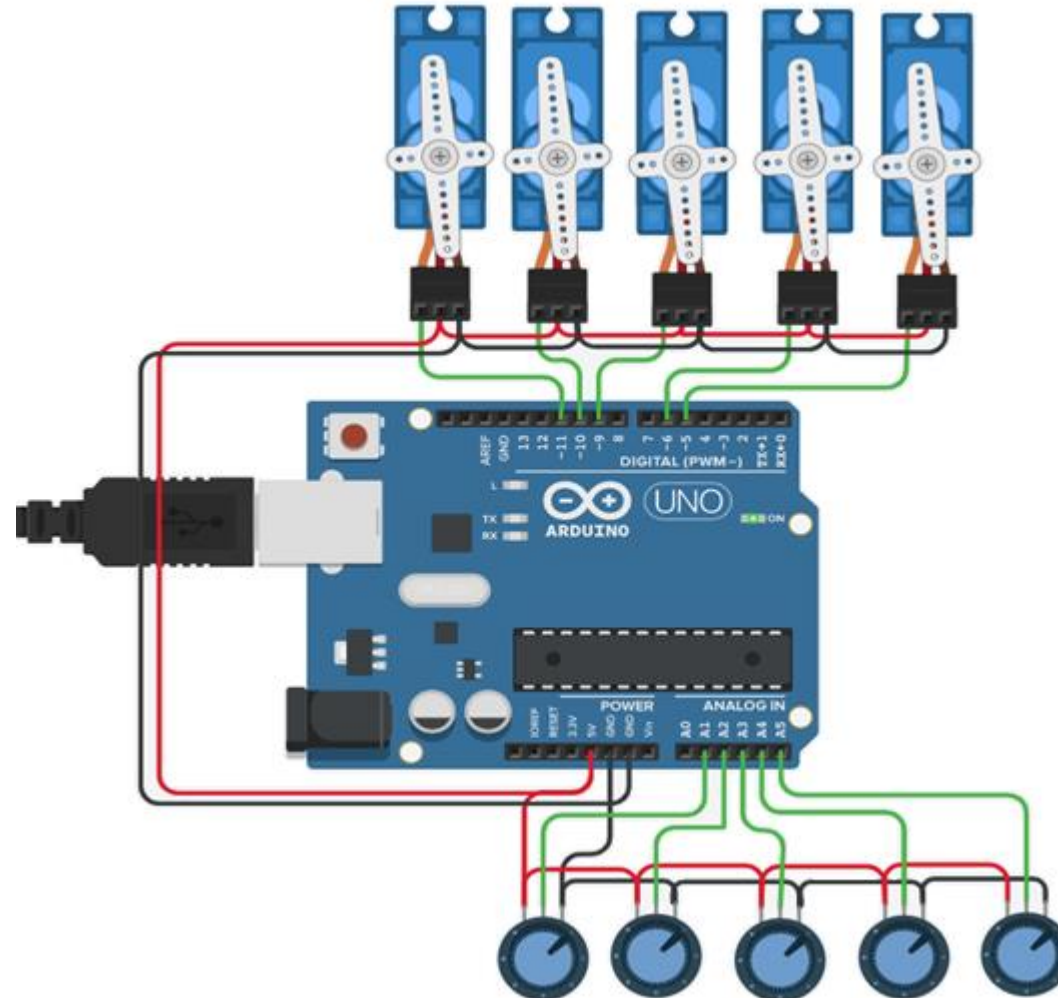


# Tinkercad Workshop



## Greifarm-Steuerung mit 5 Servo-Motore

Fünf Servo-Motore  
werden für einen  
Greifarm mit fünf Potis  
gesteuert.  
Schreibe das Programm in  
Blöcke.



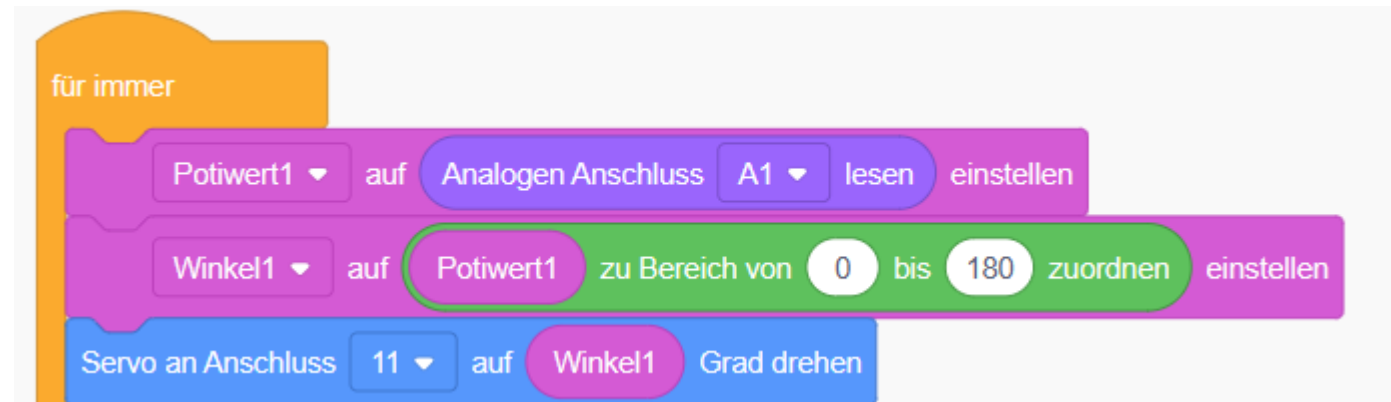


# Tinkercad Workshop



## Greifarm-Steuerung mit 5 Servo-Motore

Fünf Servo-Motore  
werden für einen  
Greifarm mit fünf Potis  
gesteuert.  
Schreibe das Programm in  
Blöcke

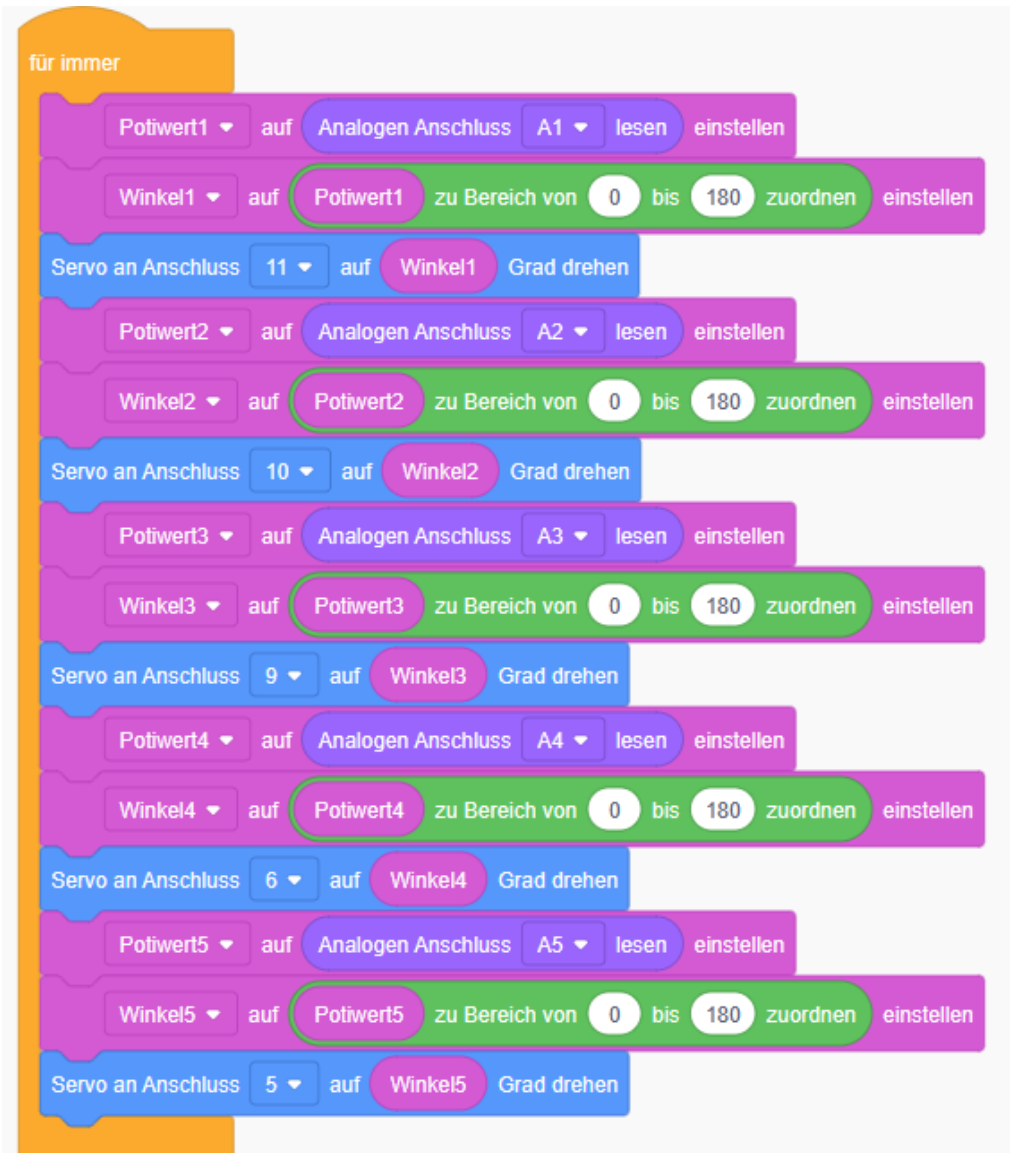


Ergänze die Programmzeilen

# Tinkercad Workshop

## Greifarm-Steuerung mit 5 Servo-Motore

Fünf Servo-Motore werden für einen Greifarm mit fünf Potis gesteuert.  
Schreibe das Programm in Blöcke.



# Tinkercad Workshop



## Greifarm-Steuerung mit 5 Servo-Motore

Fünf Servo-Motore  
werden für einen  
Greifarm mit fünf Potis  
gesteuert.  
Schreibe das Programm in  
Blöcke.

```
// C++ code
#include <Servo.h>
int Potiwert1 = 0;
int Winkel1 = 0;
int Potiwert2 = 0;
int Winkel2 = 0;
int Potiwert3 = 0;
int Winkel3 = 0;
int Potiwert4 = 0;
int Winkel4 = 0;
int Potiwert5 = 0;
int Winkel5 = 0;
Servo servo_11;
Servo servo_10;
Servo servo_9;
Servo servo_6;
Servo servo_5;

void setup()
{
  pinMode(A1, INPUT);
  servo_11.attach(11, 500, 2500);
  pinMode(A2, INPUT);
  servo_10.attach(10, 500, 2500);
  pinMode(A3, INPUT);
  servo_9.attach(9, 500, 2500);
  pinMode(A4, INPUT);
  servo_6.attach(6, 500, 2500);
  pinMode(A5, INPUT);
  servo_5.attach(5, 500, 2500);
}
```

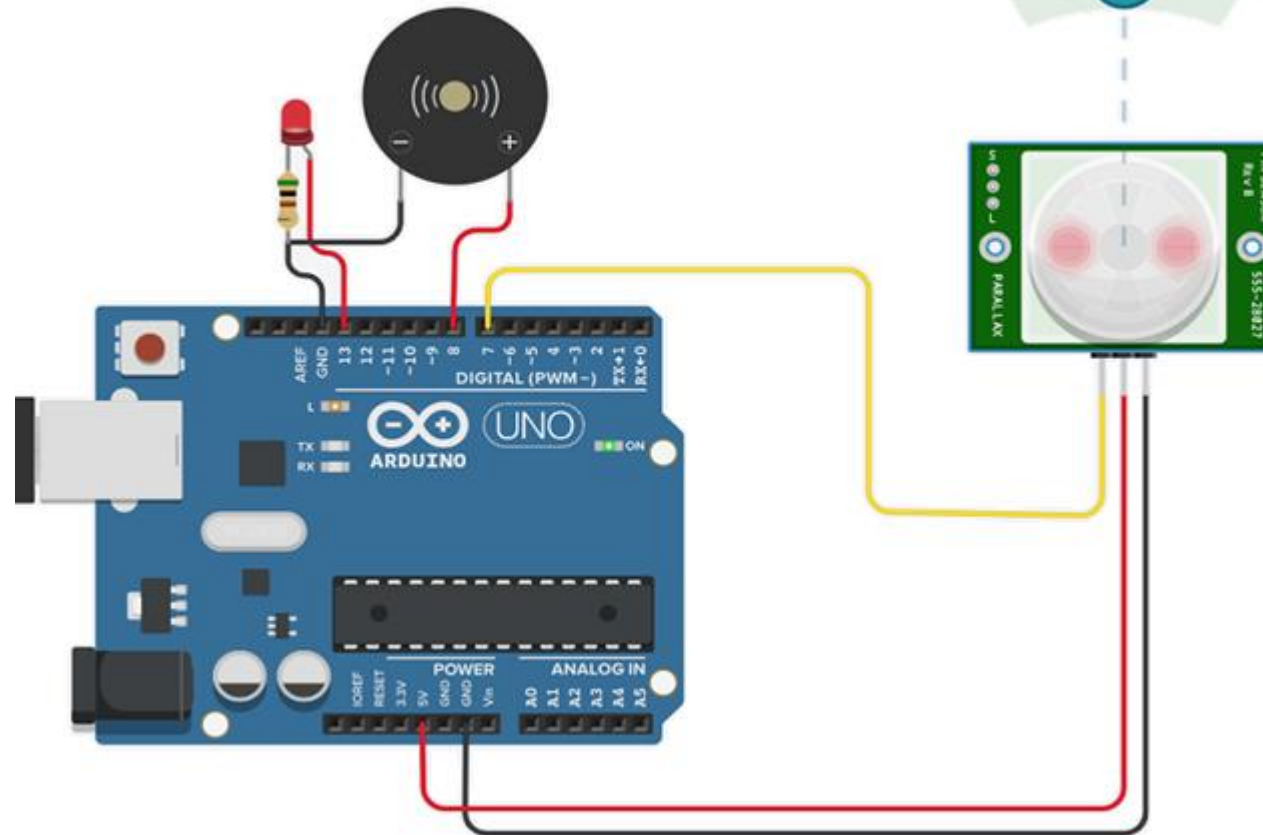
```
void loop()
{
  Potiwert1 = analogRead(A1);
  Winkel1 = map(Potiwert1, 0, 1023, 0, 180);
  servo_11.write(Winkel1);
  Potiwert2 = analogRead(A2);
  Winkel2 = map(Potiwert2, 0, 1023, 0, 180);
  servo_10.write(Winkel2);
  Potiwert3 = analogRead(A3);
  Winkel3 = map(Potiwert3, 0, 1023, 0, 180);
  servo_9.write(Winkel3);
  Potiwert4 = analogRead(A4);
  Winkel4 = map(Potiwert4, 0, 1023, 0, 180);
  servo_6.write(Winkel4);
  Potiwert5 = analogRead(A5);
  Winkel5 = map(Potiwert5, 0, 1023, 0, 180);
  servo_5.write(Winkel5);
  delay(10); // Warte 10ms
}
```

# Tinkercad Workshop



## Alarmanlage

Erkennt der PIR-BWM eine Bewegung, wird die LED und der Buzzer aktiviert.  
Schreibe das Programm in Blöcke.

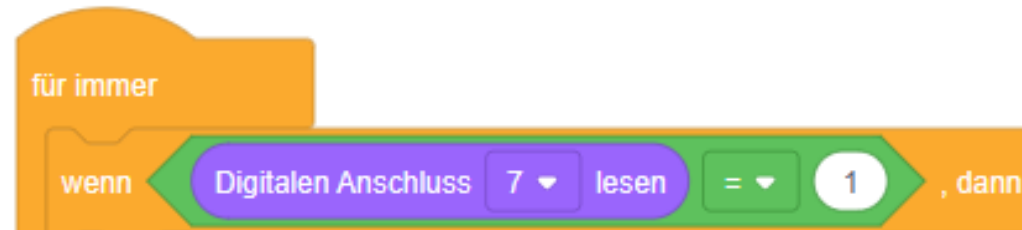


# Tinkercad Workshop



## Alarmanlage

Erkennt der PIR-BWM eine Bewegung, wird die LED und der Buzzer aktiviert.



Ergänze die Programmzeilen

# Tinkercad Workshop

## Alarmanlage

Erkennt der PIR-BWM eine Bewegung, wird die LED und der Buzzer aktiviert.

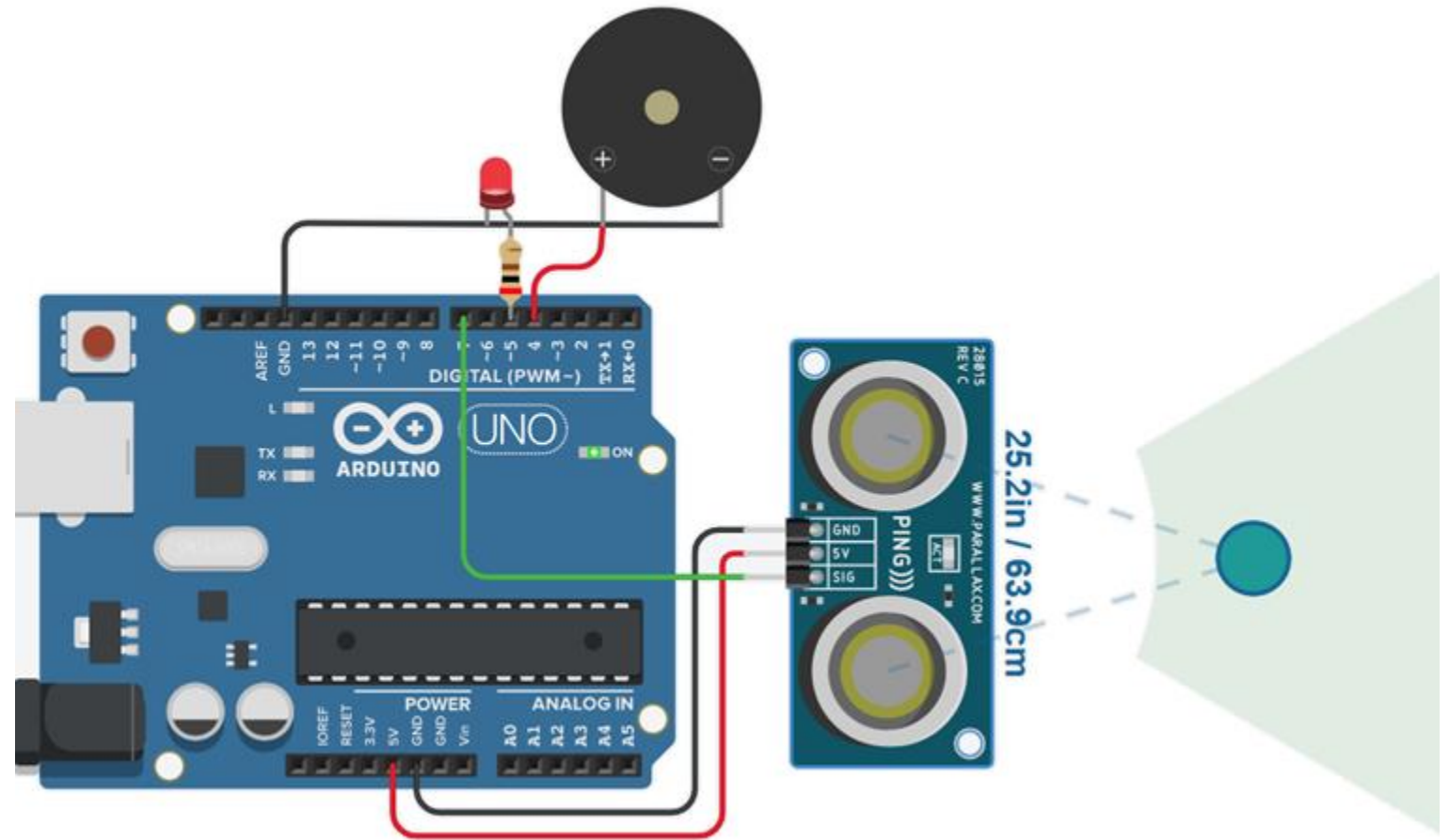


# Tinkercad Workshop



## Einparkhilfe

Bewegt sich ein Objekt auf den Entfernungsmesser zu, soll je nach dem noch verfügbaren Abstand zum Entfernungsmesser ein optisches Signal ( $<70\text{cm}$ ) oder ein optisches Signal und ein akustisches Signal ( $<50\text{cm}$ ) gegeben werden. Schreibe das Programm in Blöcke.



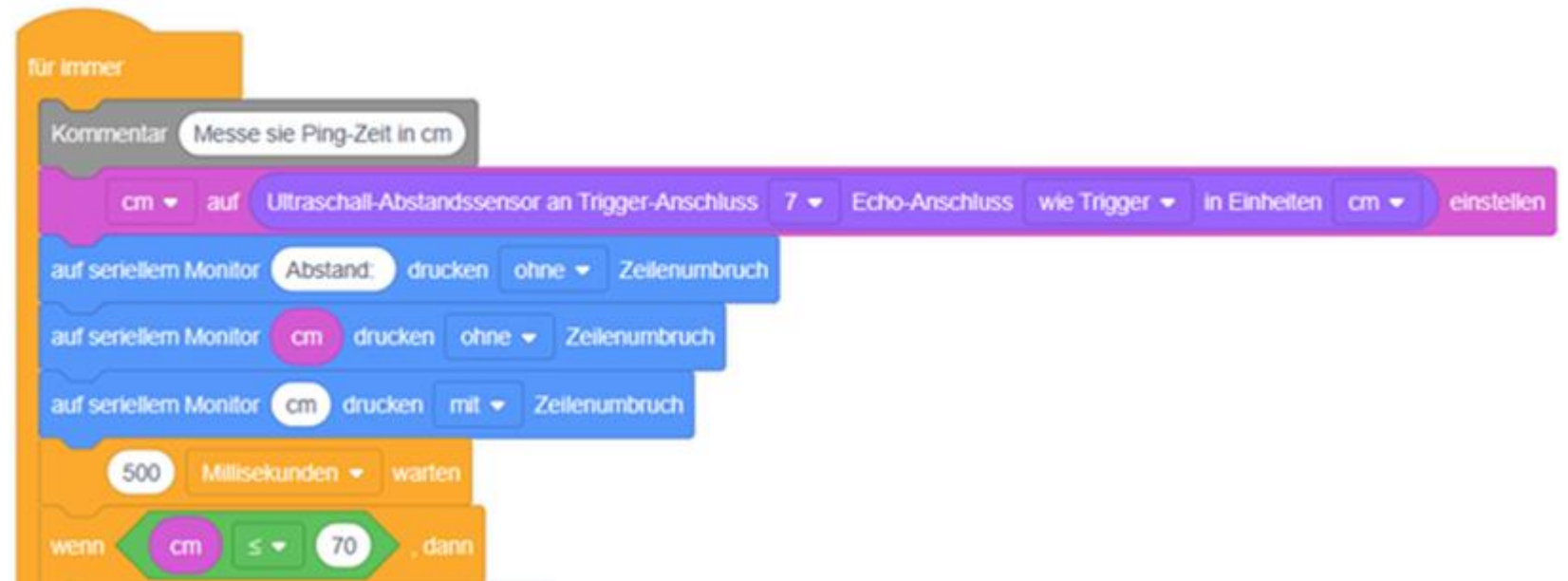


# Tinkercad Workshop



## Einparkhilfe

Bewegt sich ein Objekt auf den Entfernungsmesser zu, soll je nach dem noch verfügbaren Abstand zum Entfernungsmesser ein optisches Signal (<70cm) oder ein optisches Signal und ein akustisches Signal (<50cm) gegeben werden. Schreibe das Programm in Blöcke.



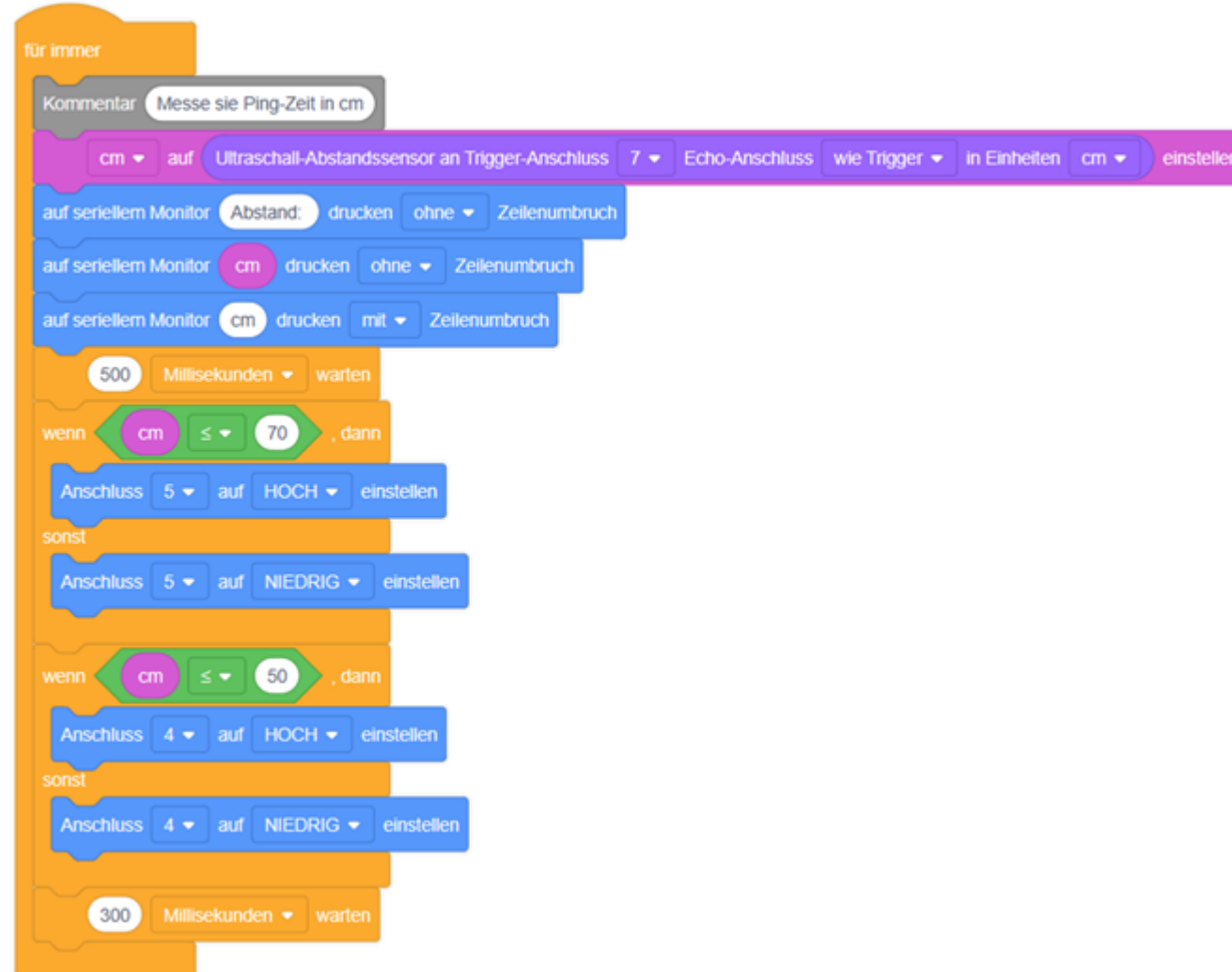
Ergänze die Programmzeilen

# Tinkercad Workshop



## Einparkhilfe

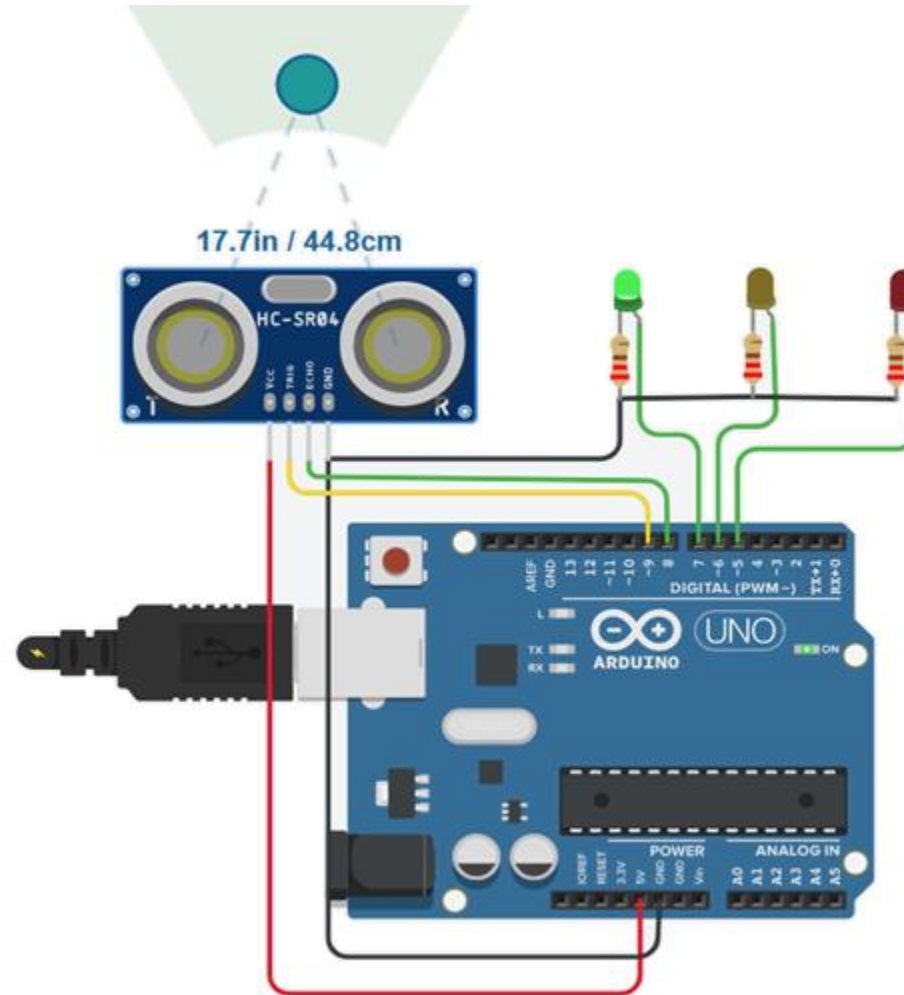
Bewegt sich ein Objekt auf den Entfernungsmesser zu, soll je nach dem noch verfügbaren Abstand zum Entfernungsmesser ein optisches Signal (<70cm) oder ein akustisches Signal (<50cm) gegeben werden. Schreibe das Programm in Blöcke.



# Tinkercad Workshop

## Ultraschallsensor aktiviert eine Ampelschaltung

Ampel mit Ultraschallsensor.  
Die Ampel zeigt zunächst rot.  
Wenn die Entfernung kleiner als  
100 cm ist, soll sie nach 1 Sekunde  
Wartezeit erst auf rot/gelb  
für eine Sekunde, dann auf grün  
für drei Sekunden springen.  
Anschließend folgt wieder eine  
Sekunde gelb und dann wieder  
rot.  
Schreibe das Programm in Text.



# Tinkercad Workshop



## Ultraschallsensor aktiviert eine Ampelschaltung

Ampel mit Ultraschallsensor.

```
//C++
//Ampel mit Ultraschallsensor

# define SENDEN 9
# define ECHO 8

// Ampel
# define ROT 5
# define GELB 6
# define GRUEN 7
long Zeit = 0;
long Entfernung = 0;

void setup()
{
    pinMode(SENDEN, OUTPUT);
    pinMode(ECHO, INPUT);
    pinMode(ROT, OUTPUT);
    pinMode(GELB, OUTPUT);
    pinMode(GRUEN, OUTPUT);
}

int EntfernungMessen()
{
    long Entfernung = 0;
    // Sender kurz ausschalten um Störungen des Signals
    // zu vermeiden
    digitalWrite(SENDEN, LOW);
    delay(5);
    // Signal senden
    digitalWrite(SENDEN, HIGH);
    delayMicroseconds(10);
    digitalWrite(SENDEN, LOW);
    // pulseIn -> Zeit messen, bis das Signal zurückkommt
    long Zeit = pulseIn(ECHO, HIGH);
    // Entfernung in cm berechnen
    Entfernung = (Zeit / 2) * 0.03432;
    return Entfernung;
}
```

```
void AmpelSchalten()
{
    delay(1000);
    digitalWrite(GELB, HIGH);
    delay(1000);
    digitalWrite(GELB, LOW);
    digitalWrite(ROT, LOW);
    digitalWrite(GRUEN, HIGH);
    delay(3000);
    digitalWrite(GRUEN, LOW);
    digitalWrite(GELB, HIGH);
    delay(1000);
    digitalWrite(GELB, LOW);
    digitalWrite(ROT, HIGH);
}

void loop()
{
    digitalWrite(ROT, HIGH);
    // Funktion aufrufen
    Entfernung = EntfernungMessen();
    // Ampel schalten
    if (Entfernung < 100)
    {
        AmpelSchalten();
    }
}
```

# Tinkercad Workshop



## Temperatur-Anzeige

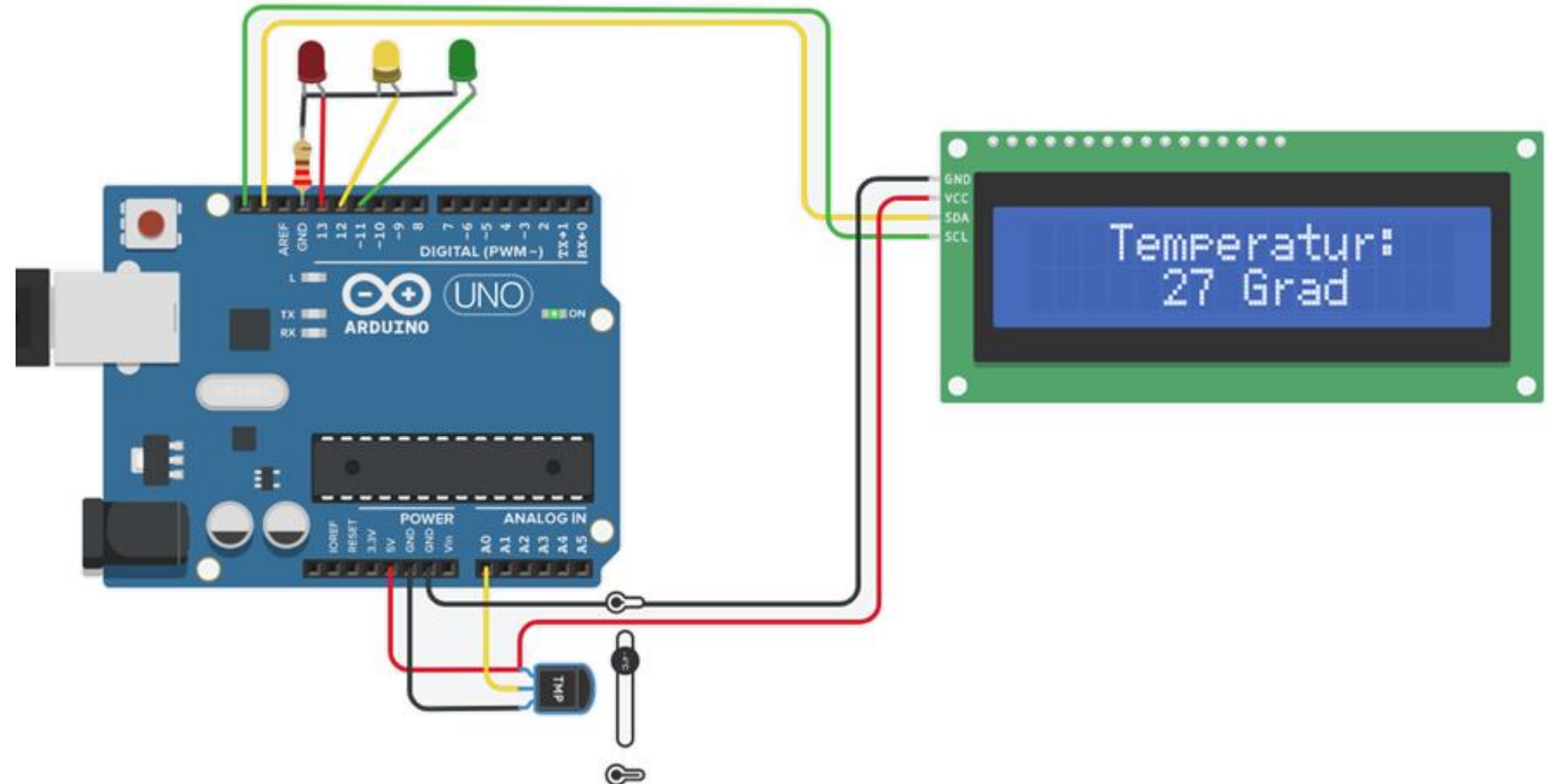
Die Temperatur soll auf einem LCD-Display angezeigt werden.

Die LEDs leuchten bei grün  $<20^\circ$

gelb  $<60^\circ$  und  $>40^\circ$

rot  $>60^\circ$

Schreibe das Programm in Blöcke.



# Tinkercad Workshop



## Temperatur-Anzeige

Die Temperatur soll auf einem LCD-Display angezeigt werden.

Die LEDs leuchten bei grün  $<20^\circ$

gelb  $<60^\circ$  und  $>40^\circ$

rot  $>60^\circ$

Schreibe das Programm in Blöcke.



Ergänze die Programmzeilen



# Tinkercad Workshop



## Temperatur-Anzeige

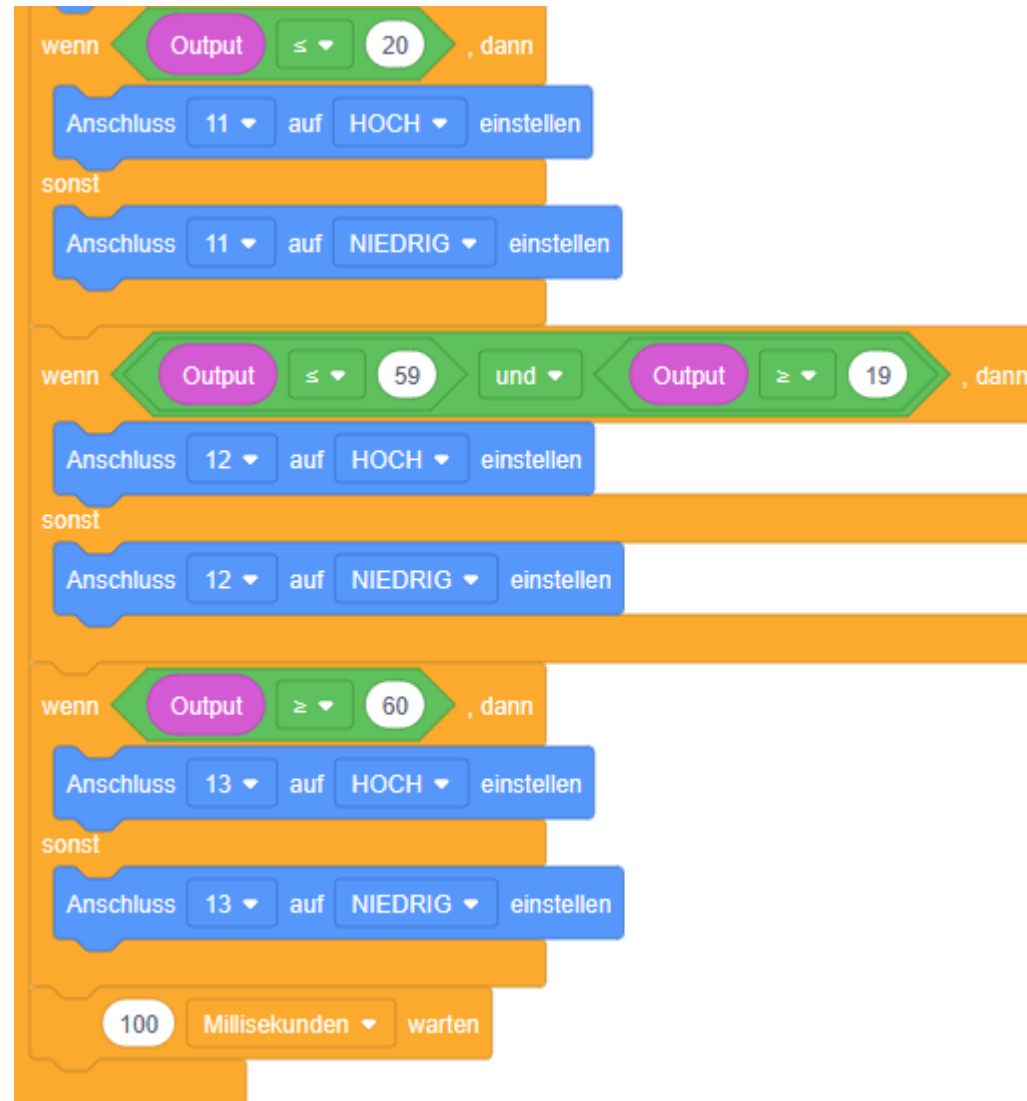
Die Temperatur soll auf einem LCD-Display angezeigt werden.

Die LEDs leuchten bei grün  $<20^{\circ}$

gelb  $<60^{\circ}$  und  $>40^{\circ}$

rot  $>60^{\circ}$

Schreibe das Programm in Blöcke.





# Tinkercad Workshop



## Feuchte-Anzeige

Der Feuchtesensor zeigt  
die Bodenfeuchte mit  
Hilfe der LEDs an.

rot <200

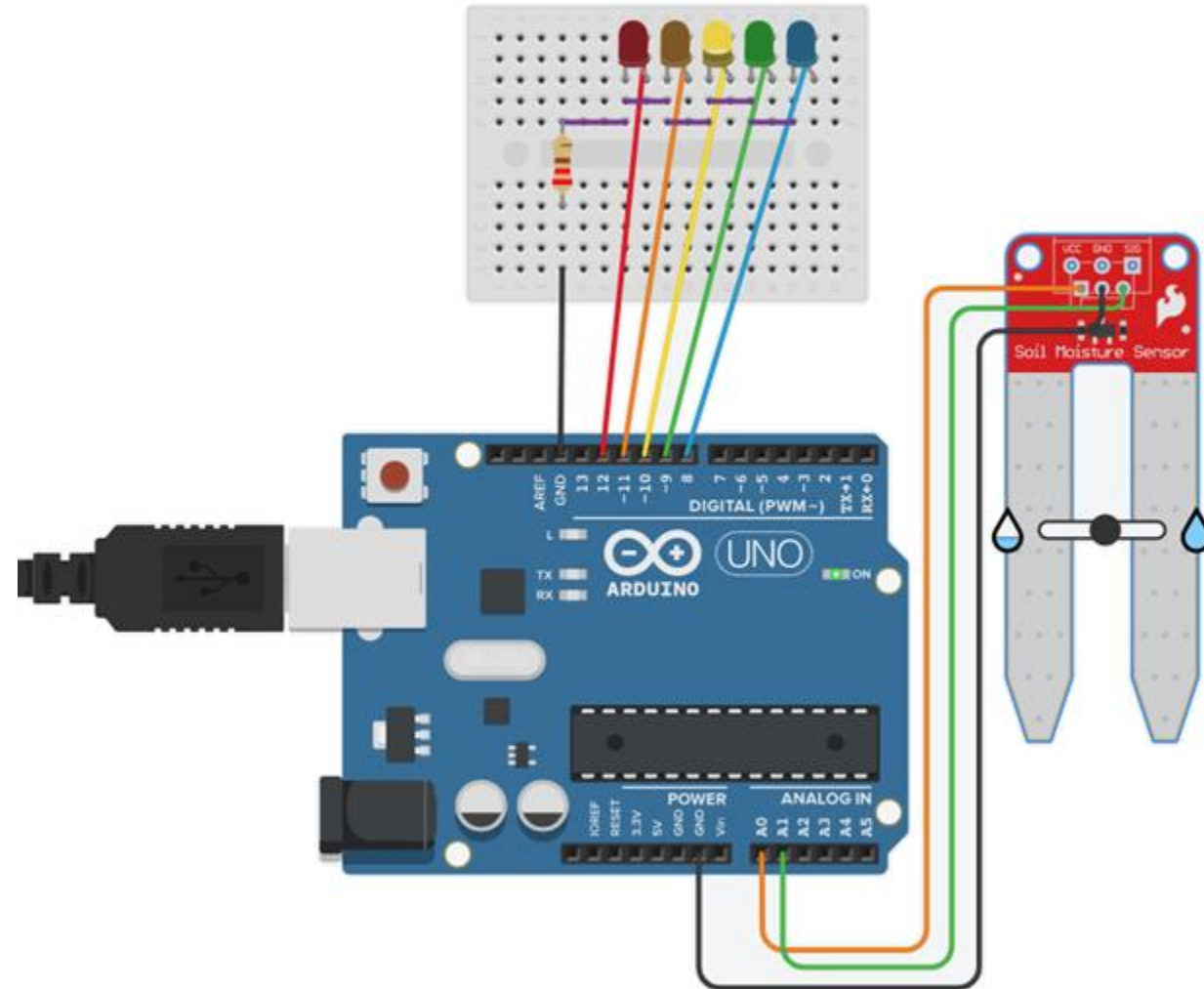
orange <400

gelb <600

grün <800

blau >800

Schreibe das Programm  
in Blöcke.



# Tinkercad Workshop



## Feuchte-Anzeige

Der Feuchtesensor zeigt  
die Bodenfeuchte mit  
Hilfe der LEDs an.

rot <200

orange <400

gelb <600

grün <800

blau >800

Schreibe das Programm  
in Blöcke.



Ergänze die Programmzeilen

# Tinkercad Workshop

## Feuchte-Anzeige

Der Feuchtesensor zeigt die Bodenfeuchte mit Hilfe der LEDs an.

rot <200

orange <400

gelb <600

grün <800

blau >800

Schreibe das Programm in Blöcke.

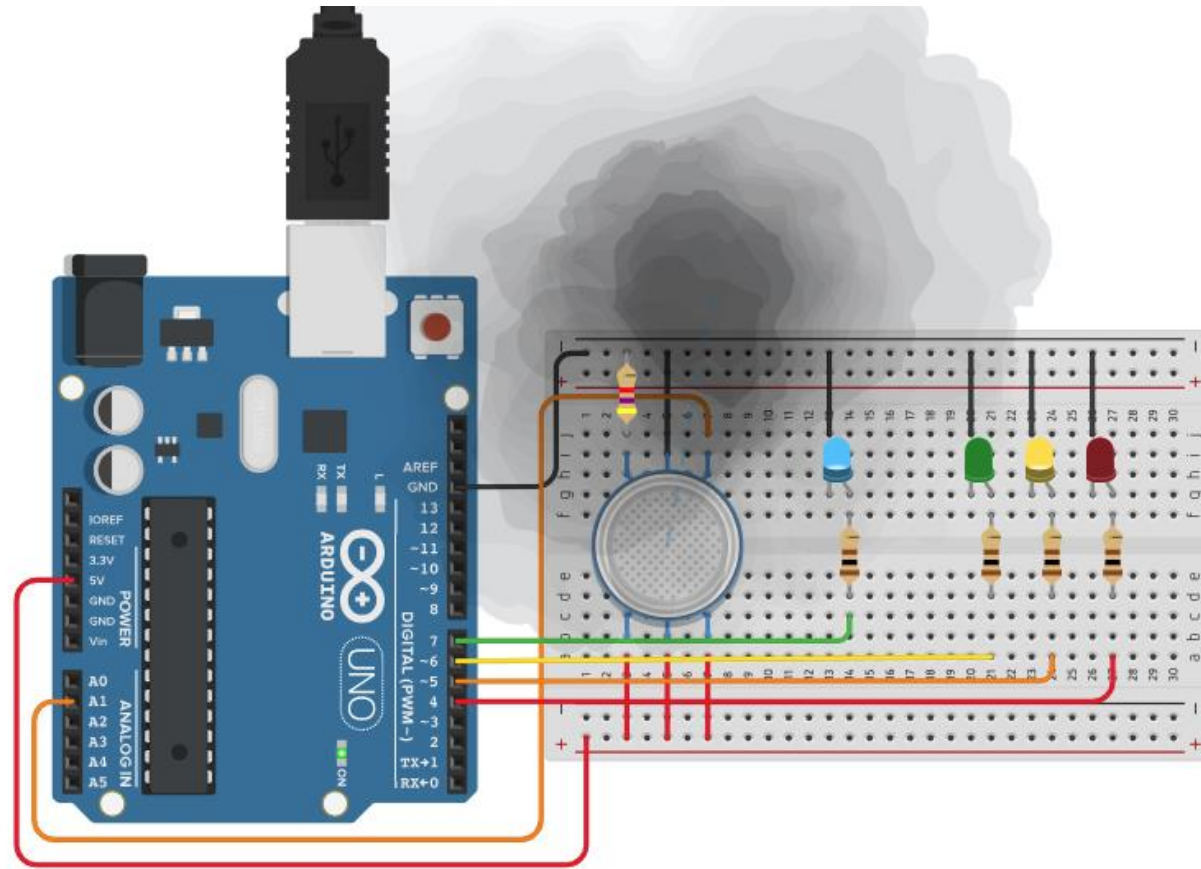


# Tinkercad Workshop



## Gas-Ampel

Die Gas-Ampel zeigt Gas mit Hilfe der drei LEDs an.  
Das Gas liegt am A1 in einem Bereich von 0-1000ppm.  
Die LEDs:  
rot >500ppm  
gelb >350ppm  
grün <350ppm  
blau betriebsbereit  
Schreibe das Programm in Blöcke.



# Tinkercad Workshop



## Gas-Ampel

Die Gas-Ampel zeigt Gas mit Hilfe der drei LEDs an.

Das Gas liegt am A1 in einem Bereich von 0-1000ppm.

Die LEDs:

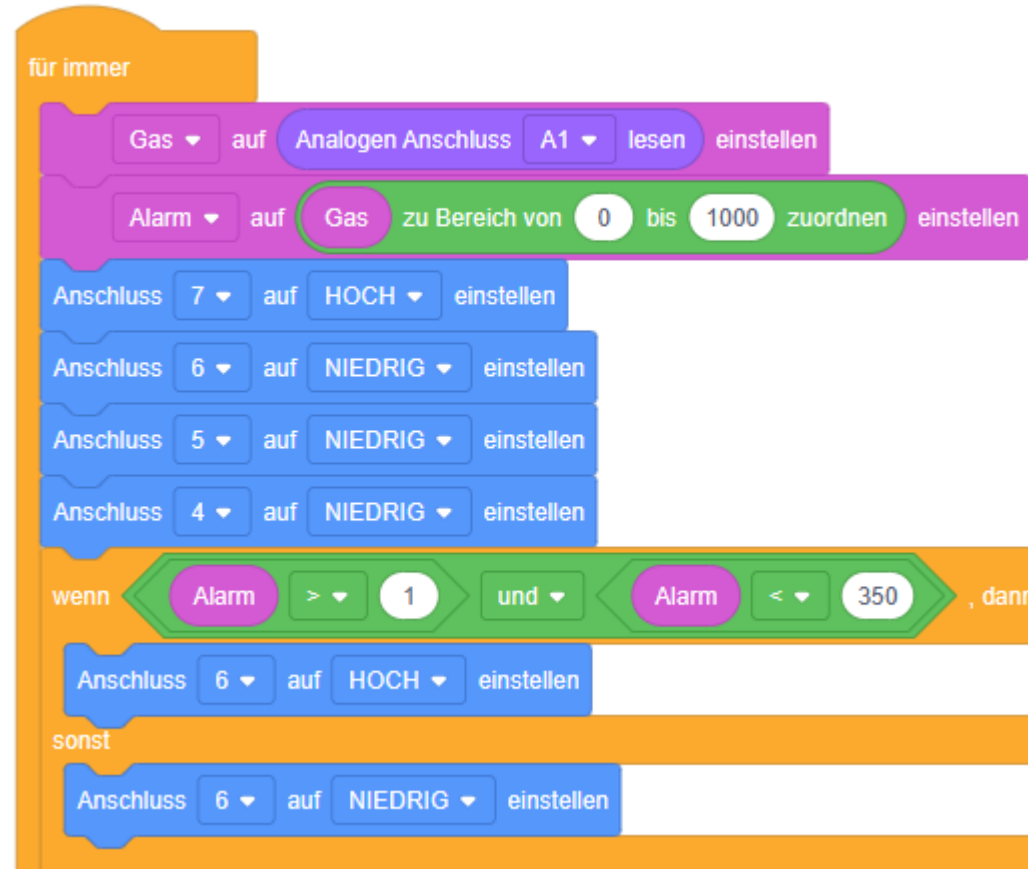
rot >500ppm

gelb >350ppm

grün <350ppm

blau betriebsbereit

Schreibe das Programm in Blöcke.



Ergänze die Programmzeilen

# Tinkercad Workshop



## Gas-Ampel

Die Gas-Ampel zeigt Gas mit Hilfe der drei LEDs an.  
Das Gas liegt am A1 in einem Bereich von 0-1000ppm.  
Die LEDs:  
rot >500ppm  
gelb >350ppm  
grün <350ppm  
blau betriebsbreit  
Schreibe das Programm in Blöcke.



# Tinkercad Workshop



## Gas-Ampel

Die Gas-Ampel zeigt Gas mit Hilfe der drei LEDs an.

Das Gas liegt am A1 in einem Bereich von 0-1000ppm.

Die LEDs:

rot >500ppm

gelb >350ppm

grün <350ppm

blau betriebsbreit

```
// C++ Gas-Ampel
```

```
int const GAS = A1;
int LED_b1 = 7;
int LED_gn = 6;
int LED_ge = 5;
int LED_rt = 4;

void setup(){
  pinMode(LED_b1, OUTPUT);
  pinMode(LED_gn, OUTPUT);
  pinMode(LED_ge, OUTPUT);
  pinMode(LED_rt, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int val = analogRead(GAS);
  val = map(val, 300, 750, 0, 1000);
  digitalWrite(LED_b1, HIGH);
  digitalWrite(LED_gn, val <= 350 ? HIGH : LOW);
  if (val >= 351 && val <= 500) {digitalWrite(LED_ge, HIGH);}
  else {digitalWrite(LED_ge, LOW);}
  digitalWrite(LED_rt, val >= 501 ? HIGH : LOW);
  Serial.println(val);
  delay(250);
}
```

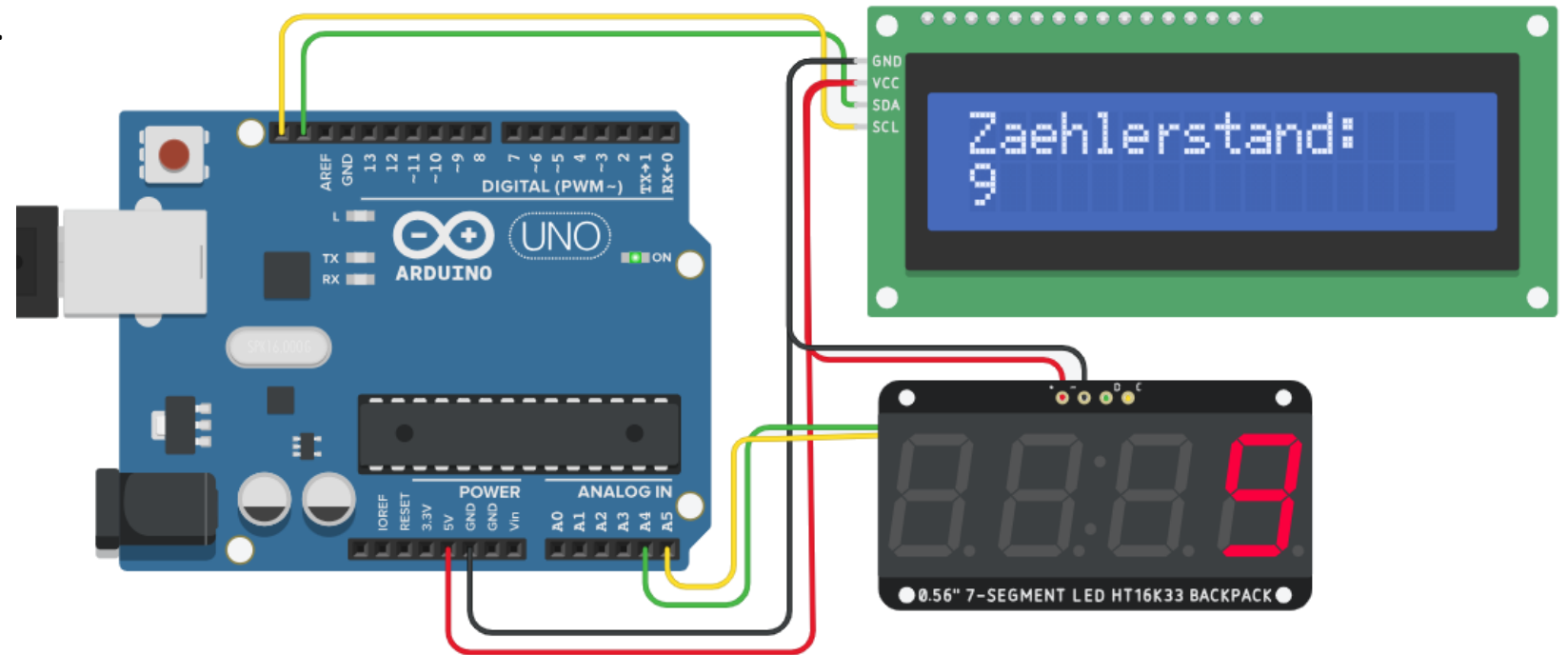


# Tinkercad Workshop



## Zähler

Programmiere einen Zähler der von 1-12 zählt und dann wieder bei 1 beginnt.  
Erstelle das Programm in Block.



# Tinkercad Workshop



## Zähler

Programmiere einen Zähler der von 1-12 zählt und dann wieder bei 1 beginnt.  
Erstelle das Programm in Block.



Ergänze die Programmzeilen

# Tinkercad Workshop

## Zähler

Programmiere einen Zähler der von 1-12 zählt und dann wieder bei 1 beginnt.  
Erstelle das Programm in Block.

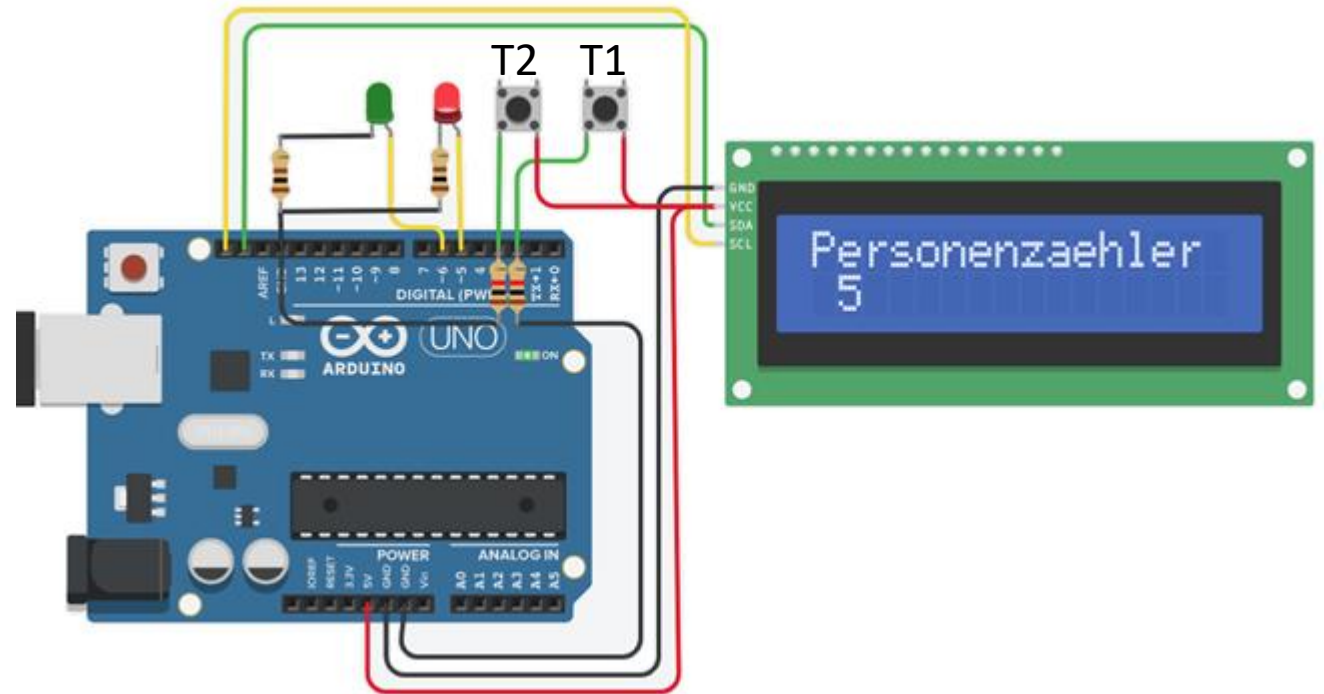


# Tinkercad Workshop



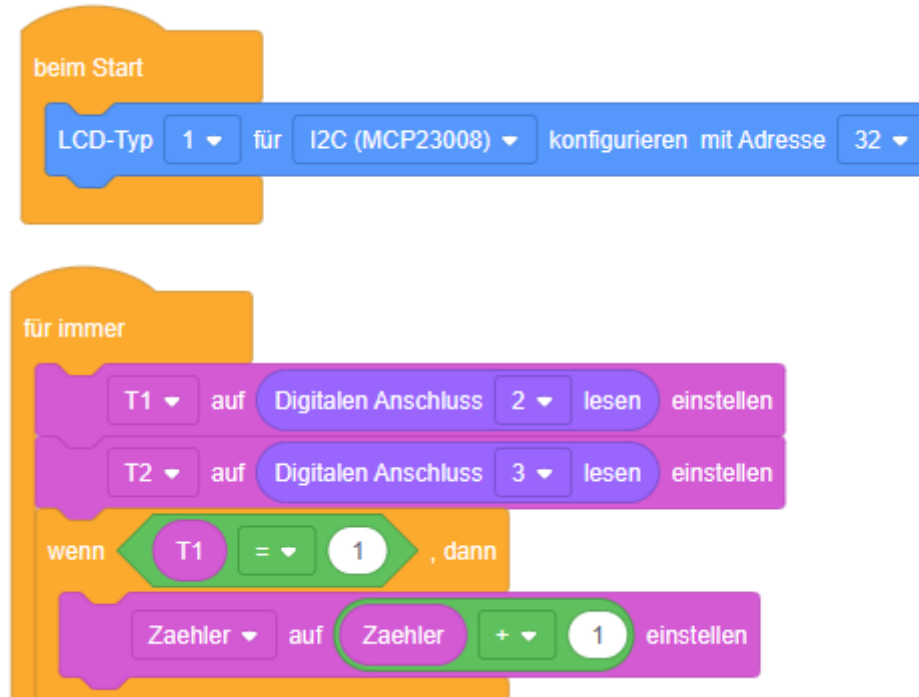
## Personenzähler

Mit der Taste 1 wird der Personenzähler um 1 erhöht, mit der Taste 2 um 1 verringert. Ist der Zählerstand 5 oder  $> 5$ , leuchtet die Rote LED, bei einem Zählerstand  $< 5$  leuchtet die grüne LED. Erstelle das Programm in Block.



# Tinkercad Workshop

## Personenzähler

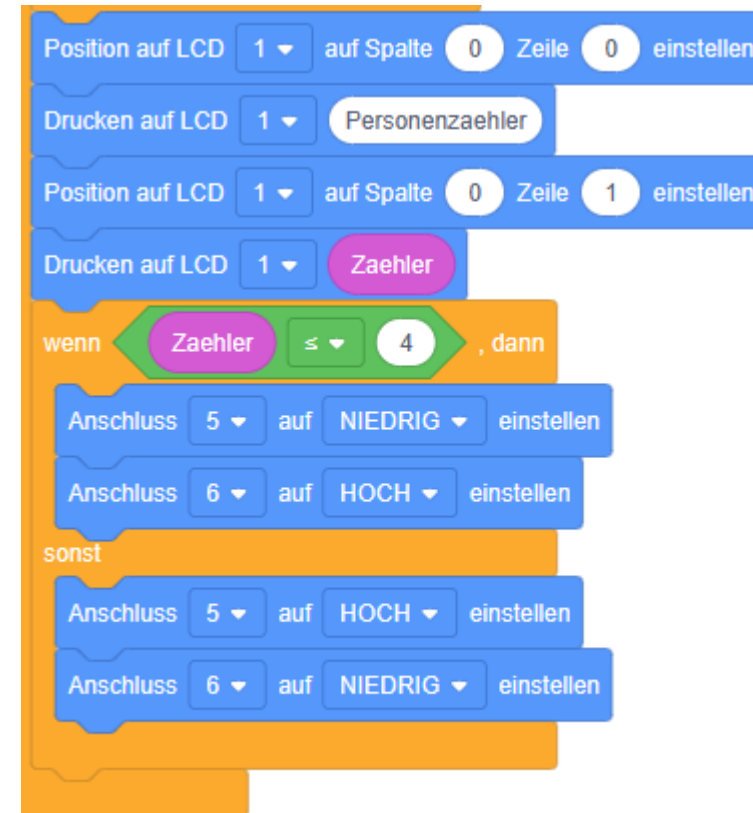
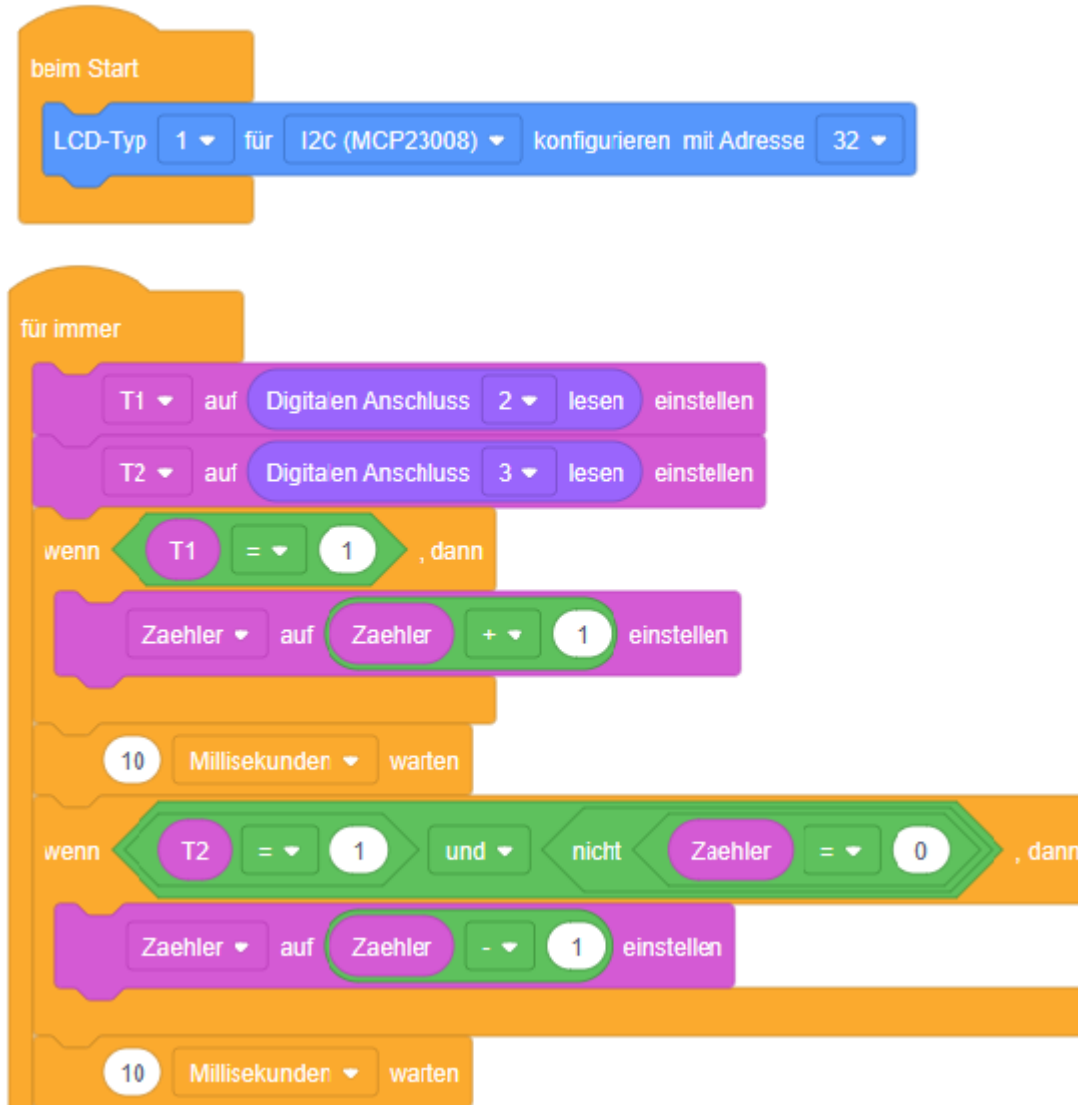


Ergänze die Programmzeilen

# Tinkercad Workshop



## Personenzähler



# Tinkercad Workshop

## Personenzähler

```
// C++ Personenzähler

#include <Adafruit_LiquidCrystal.h>

int Zaehler = 0;

int T1 = 0;

int T2 = 0;

Adafruit_LiquidCrystal lcd_1(0);

void setup()
{
  lcd_1.begin(16, 2);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}
```

```
void loop()
{
  T1 = digitalRead(2);
  T2 = digitalRead(3);
  if (T1 == 1) {
    Zaehler = (Zaehler + 1);
  }
  delay(10); // Warte 10 Millisek.
  if (T2 == 1 && !(Zaehler == 0)) {
    Zaehler = (Zaehler - 1);
  }
  delay(10); // Warte 10 Millisek.
  lcd_1.setCursor(0, 0);
  lcd_1.print("Personenzaehler");
  lcd_1.setCursor(0, 1);
  lcd_1.print(Zaehler);
  if (Zaehler <= 4) {
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
  } else {
    digitalWrite(5, HIGH);
    digitalWrite(6, LOW);
  }
}
```

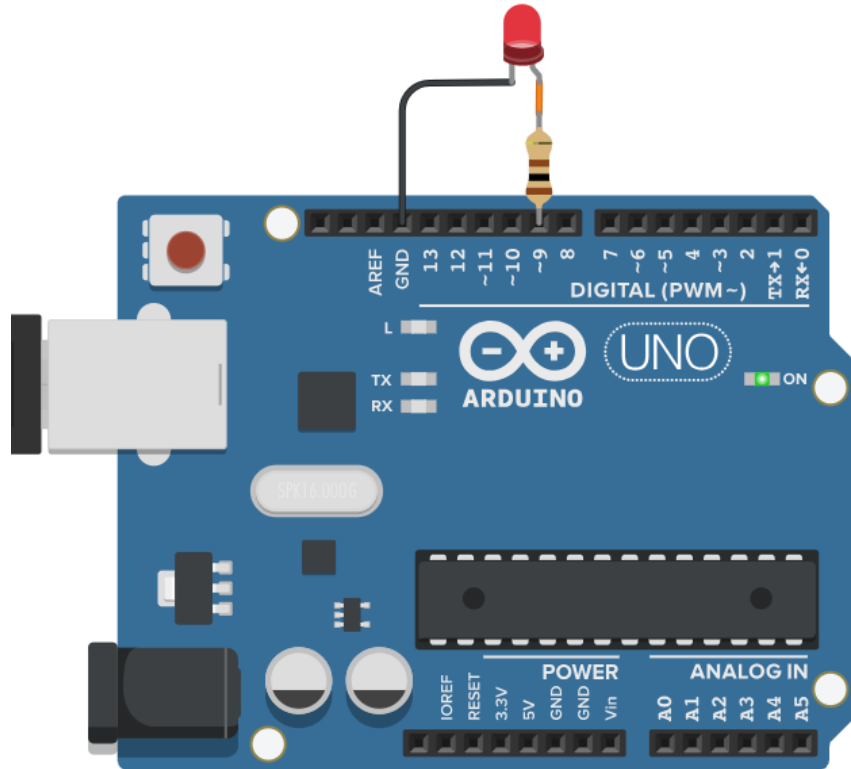




# Tinkercad Workshop



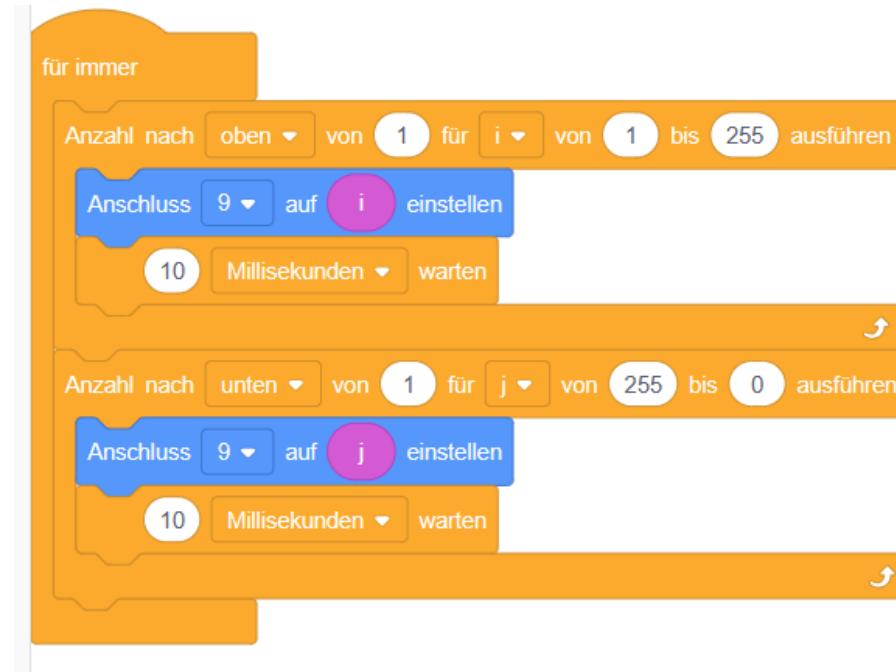
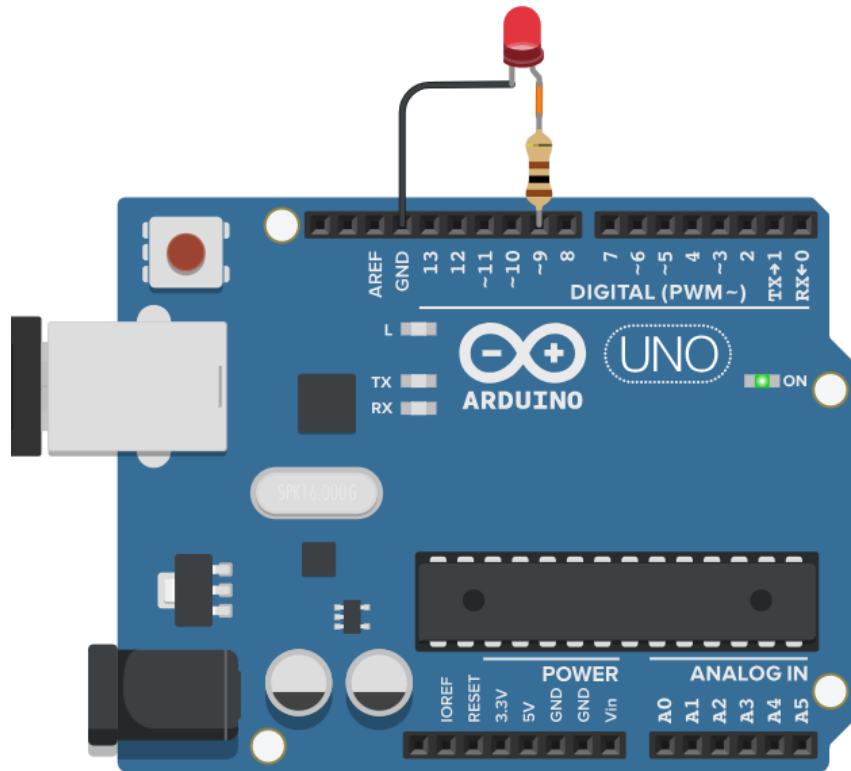
Mit einem Vor-Rückwärtszähler eine LED dimmen



Ergänze die Programmzeilen

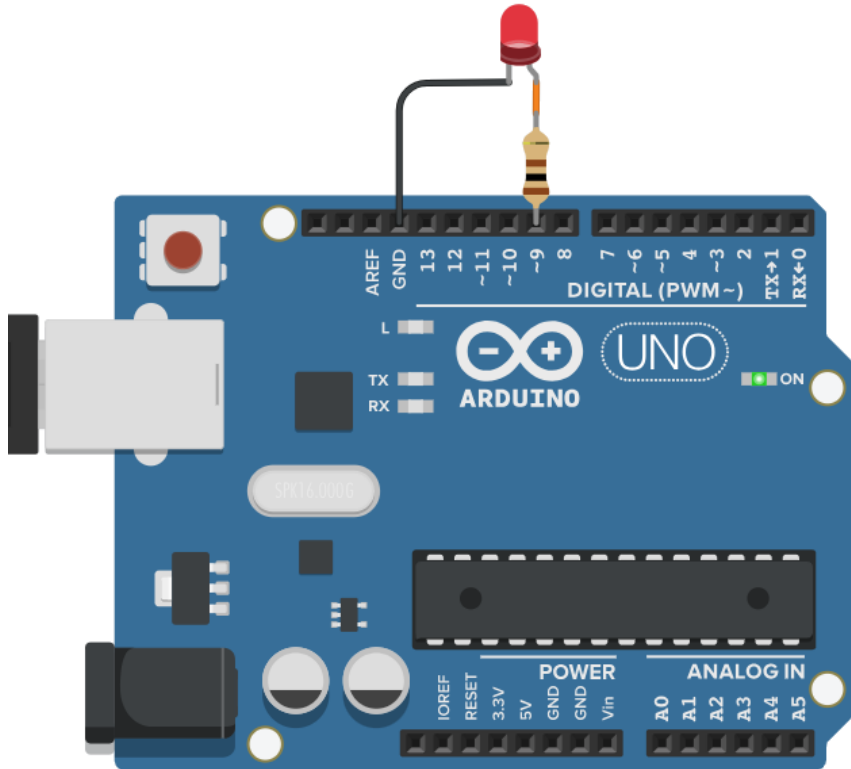
# Tinkercad Workshop

Mit einem Vor-Rückwärtszähler eine LED dimmen



# Tinkercad Workshop

Mit einem Vor-Rückwärtszähler eine LED dimmen



```
// C++ LED dimmen
```

```
int i = 0;
```

```
int j = 0;
```

```
void setup()
```

```
{
```

```
  pinMode(9, OUTPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
  for (i = 1; i <= 255; i += 1) {
```

```
    analogWrite(9, i);
```

```
    delay(10); // Warte 10 Millisek.
```

```
  }
```

```
  for (j = 255; j >= 0; j -= 1) {
```

```
    analogWrite(9, j);
```

```
    delay(10); // Warte 10 Millisek.
```

```
  }
```

```
}
```

# Tinkercad Workshop



**Der Befehlssatz der Blocksprache ist begrenzt,  
deshalb lassen sich nicht alle Programme in Block-  
Sprache lösen.**

# Tinkercad Workshop



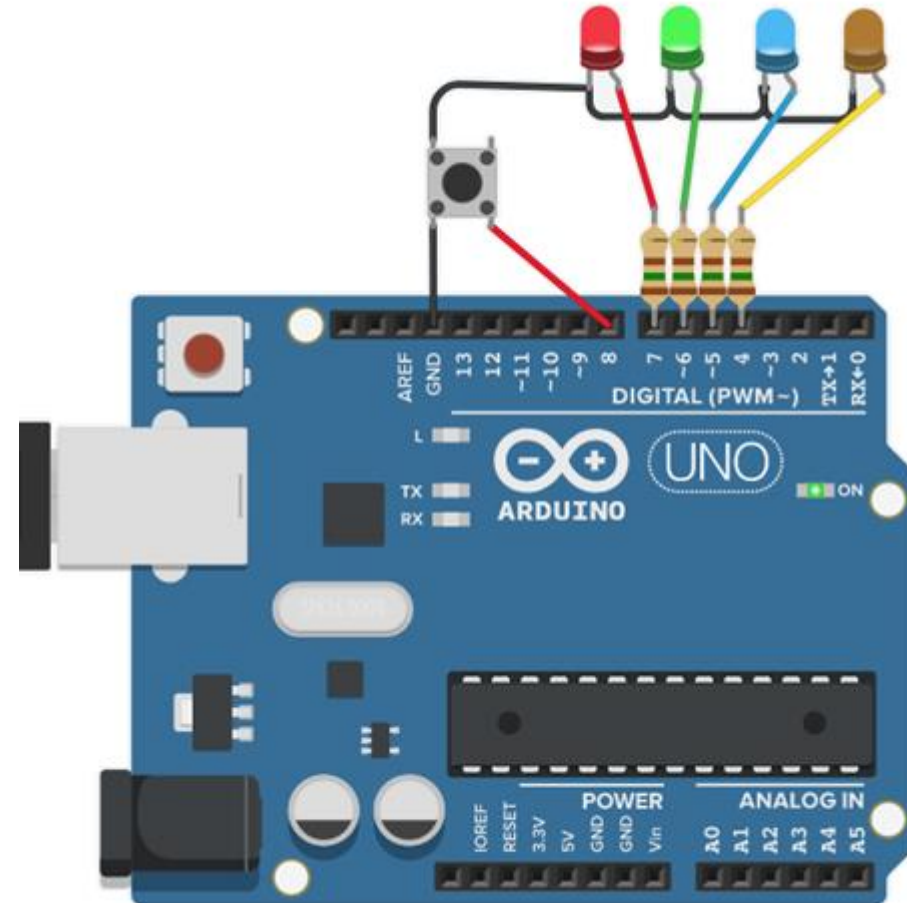
## delay() und millis()

Der Befehl delay() ist in C++ Programmen nicht immer hilfreich. Er blockiert den Programmablauf und so können in der Delay-Phase z. B. keine Eingänge abgefragt werden.

Abhilfe bietet der millis() Befehl.

Am besten erkennt man den Unterschied, wenn die nachfolgenden Programme mit delay() und millis() verglichen werden.

Drei LEDs, die mit unterschiedlichen Frequenzen blinken sollen, sind mit dem delay() Befehl nicht realisierbar. Auch der Taster, der eine LED einschalten soll, wird in der Delay-Phase nicht erkannt.



# Tinkercad Workshop



## Teste das Programm mit delay()

```
//C++ Testprogramm delay()

#define LED1 4
#define LED2 5
#define LED3 6

unsigned long pauseLED1 = 500;
unsigned long pauseLED2 = 1000;
unsigned long pauseLED3 = 5000;

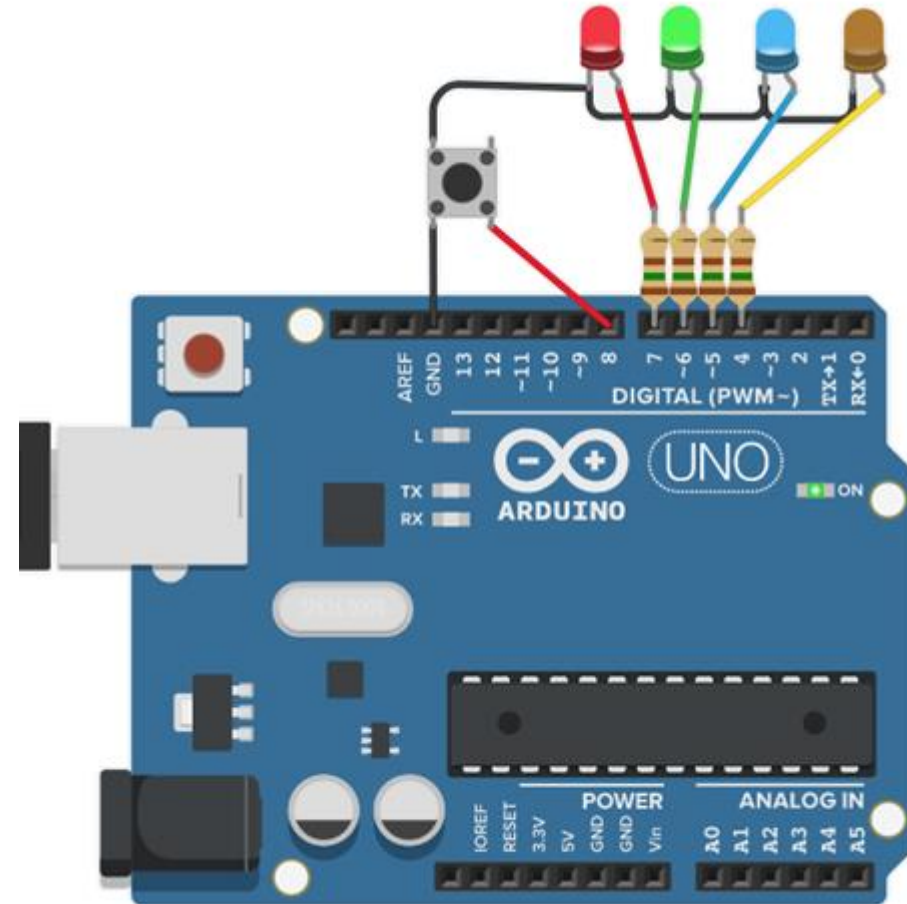
bool LEDstate1 = LOW;
bool LEDstate2 = LOW;
bool LEDstate3 = LOW;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
}

void loop() {
  delay(pauseLED1);
  LEDstate1 = !LEDstate1;
  digitalWrite(LED1, LEDstate1);

  delay(pauseLED2);
  LEDstate2 = !LEDstate2;
  digitalWrite(LED2, LEDstate2);

  delay(pauseLED3);
  LEDstate3 = !LEDstate3;
  digitalWrite(LED3, LEDstate3);
}
```



# Tinkercad Workshop



## Teste das Programm mit millis()

```
//C++ Testprogramm millis()

#define LED1 4
#define LED2 5
#define LED3 6
#define LED4 7
#define TASTER 8

long pauseLED1 = 500; //long-> 4Bytes
long pauseLED2 = 1000;
long pauseLED3 = 5000;

long oldMillis_LED1 = 0;
long oldMillis_LED2 = 0;
long oldMillis_LED3 = 0;

bool LEDstate1 = LOW; //true oder false
bool LEDstate2 = LOW;
bool LEDstate3 = LOW;
bool LEDstate4 = LOW;

bool TASTERstate = HIGH;
bool TASTERstate_old = HIGH;

long prellZeit = 80;
long TASTERmillis_old = 0;
void setup() {
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(TASTER, INPUT_PULLUP);
}
```

```
void loop() {

    if (millis() - oldMillis_LED1 >= pauseLED1) {
        LEDstate1 = !LEDstate1;
        digitalWrite(LED1, LEDstate1);
        oldMillis_LED1 = millis();
    }

    if (millis() - oldMillis_LED2 >= pauseLED2) {
        LEDstate2 = !LEDstate2;
        digitalWrite(LED2, LEDstate2);
        oldMillis_LED2 = millis();
    }

    if (millis() - oldMillis_LED3 >= pauseLED3) {
        LEDstate3 = !LEDstate3;
        digitalWrite(LED3, LEDstate3);
        oldMillis_LED3 = millis();
    }

    TASTERstate = digitalRead(TASTER);

    if (TASTERstate != TASTERstate_old && millis() - TASTERmillis_old > prellZeit)
    {
        LEDstate4 = !LEDstate4;
        digitalWrite(LED4, LEDstate4);
        TASTERstate_old = TASTERstate;
        TASTERmillis_old = millis();
    }
}
```





# Tinkercad Workshop



## Ausgänge schalten mit Arrays

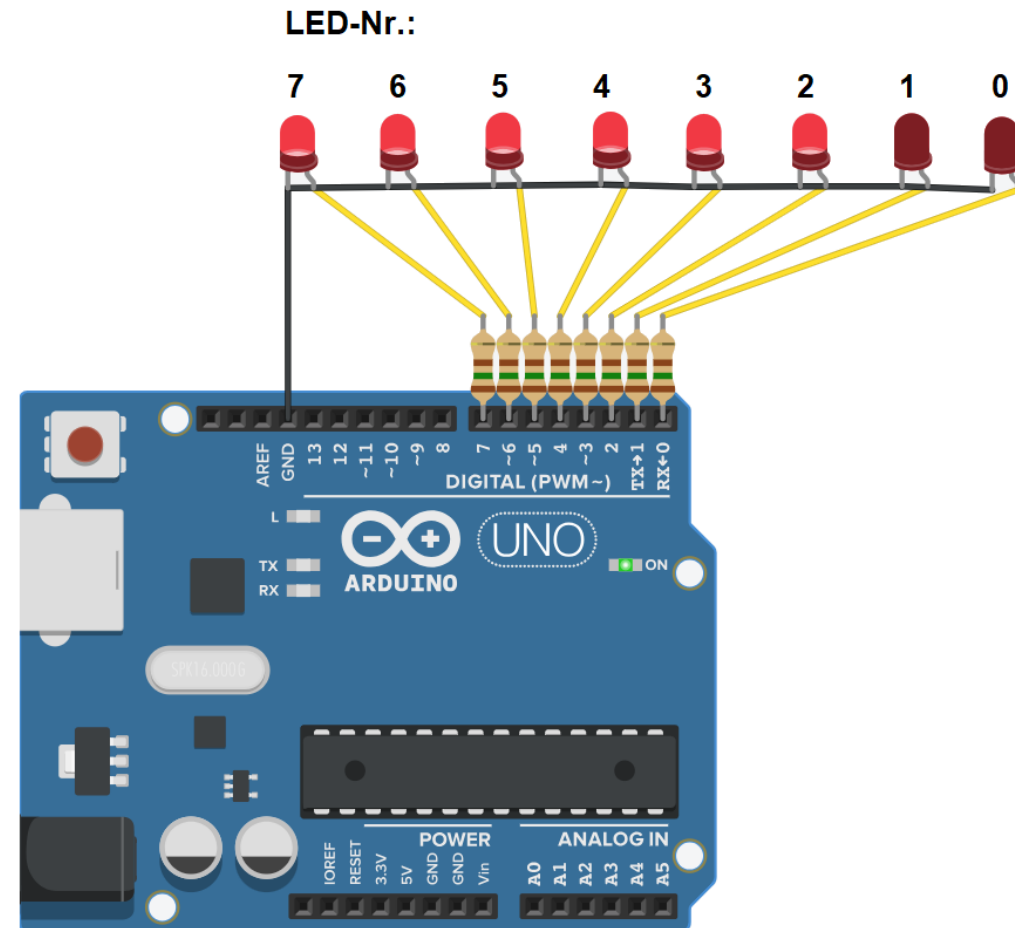
Ein Array ist eine Liste von Variablen, die den gleichen Datentyp haben. Mit einem Array lässt sich sehr einfach ein Lauflicht darstellen.

Das Lauflicht soll die LEDs von links nach rechts und zurück ein/ausschalten.

Array-Befehl:

```
int LEDs[] = {7, 6, 5, 4, 3, 2, 1, 0};
```

Outputs								
int								
Elemente	7	6	5	4	3	2	1	0
Index	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]



# Tinkercad Workshop

## Ausgänge schalten mit Arrays

```
//C++ Lauflicht links-rechts mit Array

#define Anzahl 8 // Anzahl der Array Elemente
int LED[] = {7, 6, 5, 4, 3, 2, 1, 0}; //LEDs von li nach re

void setup(){
    for(int i = 0; i < Anzahl; i++){
        pinMode(LED[i],OUTPUT); // Jeden Pin als Ausgang konfigurieren
    }
}

void loop(){
    for(int i = 0; i < Anzahl; i++){
        digitalWrite(LED[i],HIGH); // Jede einzelne LED anschalten
        delay(250);
    }
    delay(250);
    for(int i = Anzahl; i >= 0 ; i--){
        digitalWrite(LED[i],LOW); // Jede einzelne LED ausschalten
        Serial.println(i);
        delay(250);
    }
    delay(250);
}
```



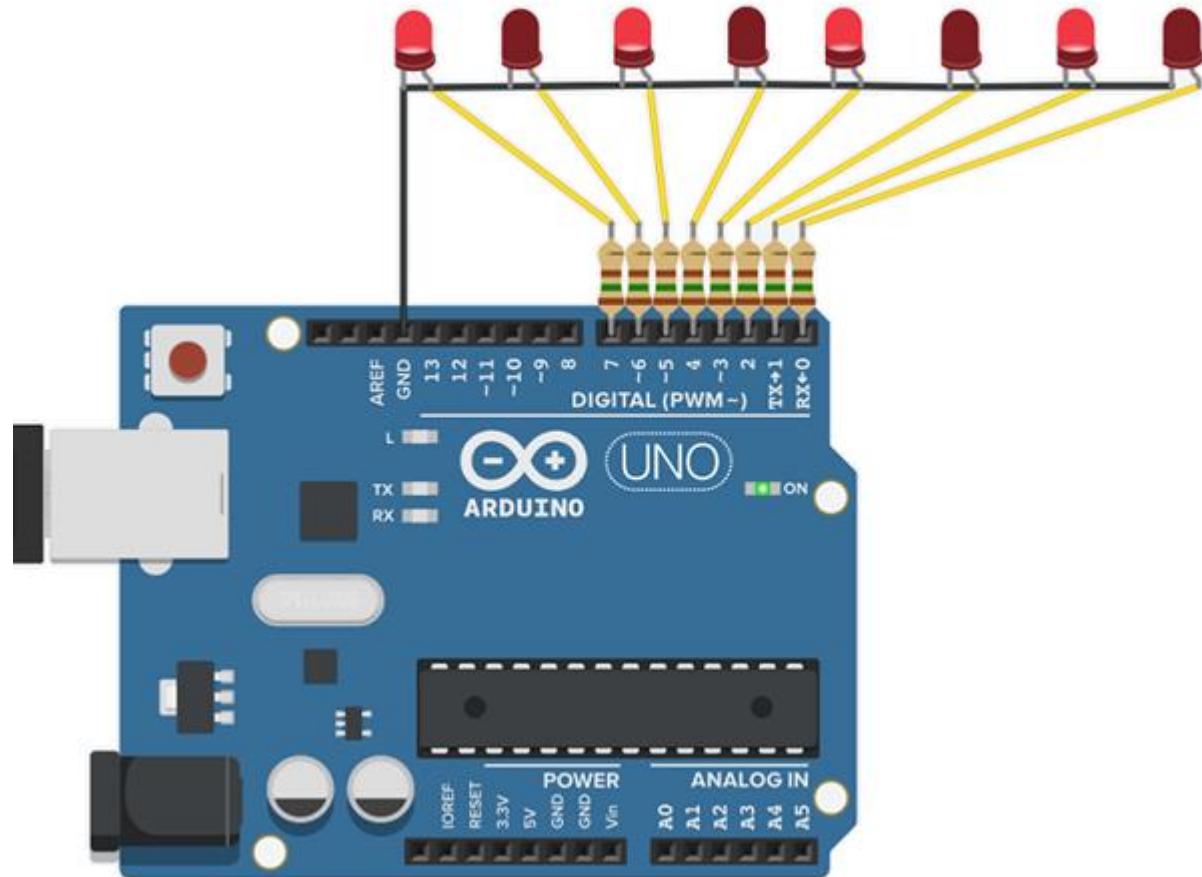
# Tinkercad Workshop



## Ausgänge schalten mit PORT

Funktionen:

- Lauflicht hin und zurück
- alle LEDs ein
- alle LEDs aus
- die LEDs 7 5 3 1 ein
- die LEDs 6 4 2 0 ein
- die beiden äußeren LEDs ein
- die beiden inneren LEDs ein
- alle LEDs aus



# Tinkercad Workshop



## Ausgänge schalten mit PORT

Funktionen wie `digitalWrite()` und `digitalRead()` sind relativ langsam in der Ausführung und manchmal werden Programme mit vielen Ein- oder Ausgängen komplex und schwer überschaubar. Deshalb ist es oft übersichtlicher die Pins mithilfe der Ports zu definieren und zu schalten.

Es existieren drei sogenannte Register:

DDR - Data Direction Register - read/write

PORT - Port Register - read/write

PIN - Port Input Register - read only

→ Pins als INPUT = 0 oder als OUTPUT = 1 definieren

→ Pins als HIGH = 1 oder als LOW = 0 festlegen

→ gibt den Zustand der Pins an, die im DDR-Register auf Input gesetzt wurden

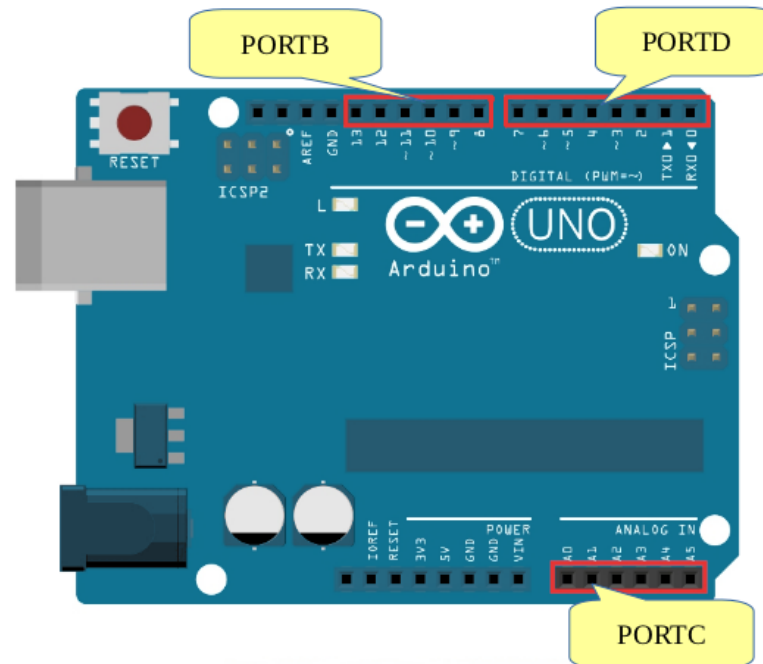
Das Register wird jeweils mit dem Namen des Ports (D, B oder C) ergänzt.

Beim Zugriff auf die Ports wird jeweils ein Bit gesetzt. Sein Wert ist entweder 1 = an, oder 0 = aus.

# Tinkercad Workshop

## Ausgänge schalten mit PORT

Die digitalen und die analogen Pins des Arduinos sind in drei Gruppen aufgeteilt:



Port D	D7	D6	D5	D4	D3	D2	D1	D0
Port B	0	0	D13	D12	D11	D10	D9	D8
Port C	0	0	A5	A4	A3	A2	A1	A0

# Tinkercad Workshop

## Ausgänge schalten mit PORT

Beispiele:

Digitale Pins 7, 6, 5, 4 und 3 als OUTPUT setzen

**DDRD = B11111000;**

Digitale Pins 7, 6, 5, 4 und 3 auf HIGH setzen:

**PORTD = B11111000;**

Taster an Pin 13 als INPUT setzen

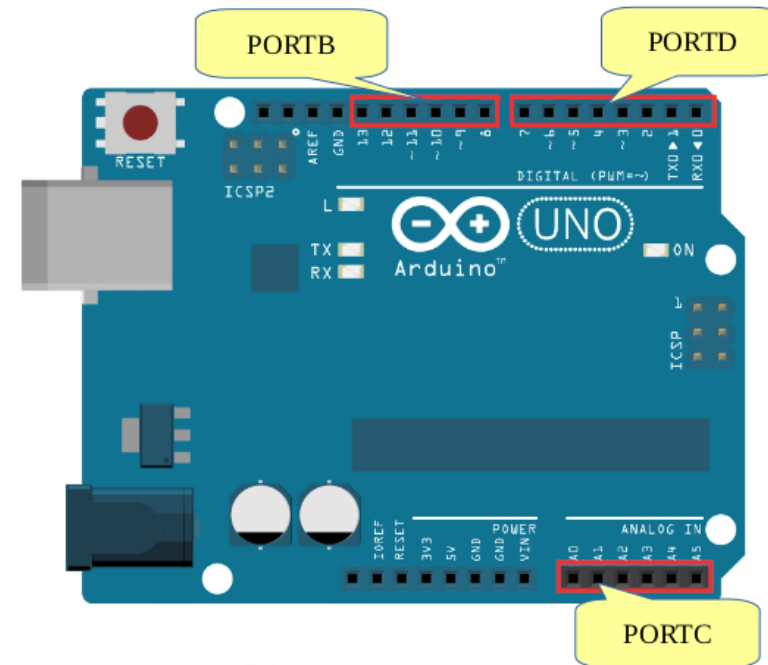
**DDRB = B00100000;**

PULL\_UP-Widerstand an Pin 13 einschalten

**PORTB = B00100000;**

Bits nach rechts schieben:

**PORTD = PORTD >> 1;**



Port D	D7	D6	D5	D4	D3	D2	D1	D0
Port B	0	0	D13	D12	D11	D10	D9	D8
Port C	0	0	A5	A4	A3	A2	A1	A0

# Tinkercad Workshop

## Ausgänge schalten mit PORT

**PORTD** ordnet den digitalen Arduino-Pins 0 bis 7 zu

DDRD - Port D Data Direction Register - read/write

PORTD - Port D Data Register - read/write

PIND - Port D Input Pins Register - read only

**PORTB** ordnet den digitalen Arduino-Pins 8 bis 13 zu

Die beiden hohen Bits (6 & 7) werden nicht zugeordnet und sind auch nicht verwendbar

DDRB - Port B Data Direction Register - read/write

PORTB - Port B Data Register - read/write

PINB - Port B Input Pins Register - read only

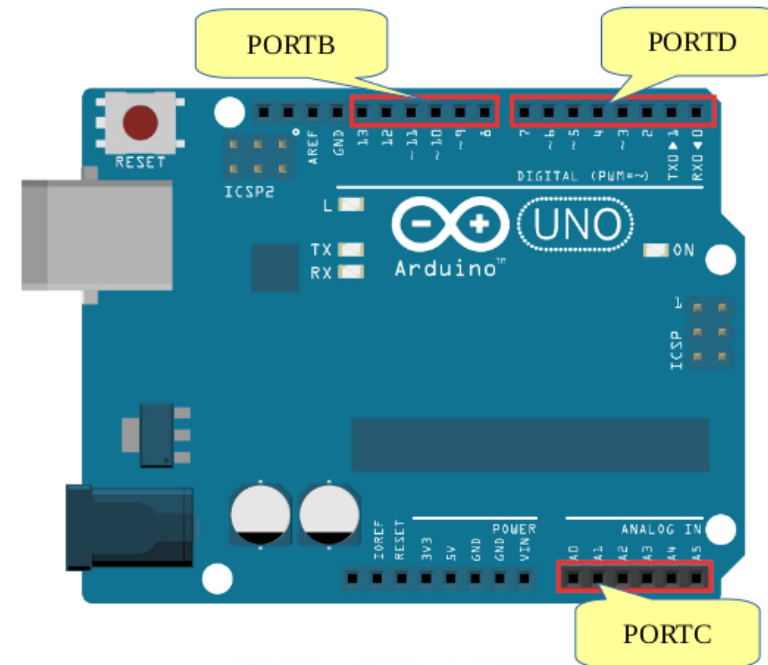
**PORTC** wird den analogen Arduino-Pins 0 bis 5 zugeordnet. Die Pins 6 & 7 sind auf dem

Arduino UNO nicht vorhanden

DDRC - Port C Data Direction Register - read/write

PORTC - Port C Data Register - read/write

PINC - Port C Input Pins Register - read only



Port D	D7	D6	D5	D4	D3	D2	D1	D0
Port B	0	0	D13	D12	D11	D10	D9	D8
Port C	0	0	A5	A4	A3	A2	A1	A0



# Tinkercad Workshop

## Ausgänge schalten mit PORT

Befehl:  
`PORTD = PORTD >> 1;`

Bits nach rechts schieben:

1	1	1	1	1	0	0	0
0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0
0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1

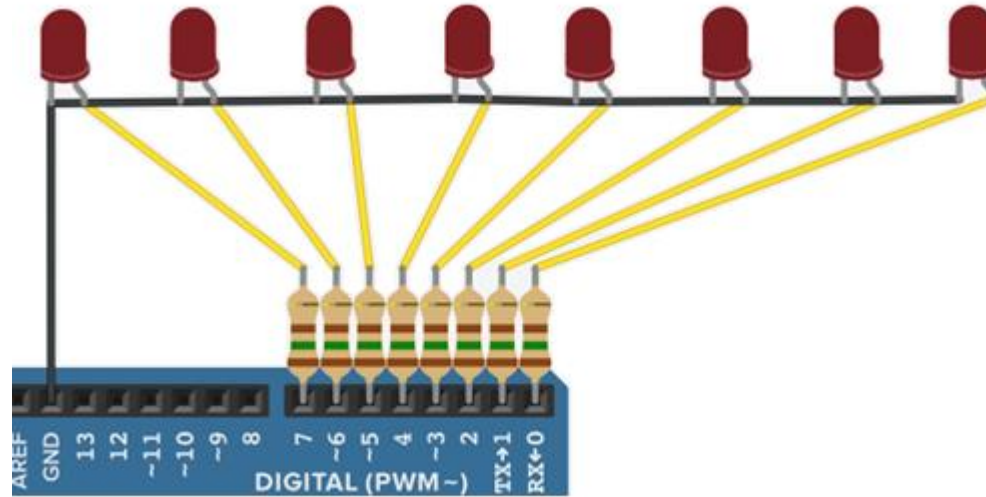
Befehl:  
`PORTD = PORTD << 1;`

Bits nach links schieben:

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	0
1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0

# Tinkercad Workshop

## Ausgänge schalten mit PORT



Der Parameter  $\sim$  kehrt den Wert eines Bits um:

```
// LEDs an den Pins 7 5 3 1 leuchten
```

```
PORTD = B10101010;
```

```
// Werte umkehren, aus 0 wird 1 und aus 1 wird 0
```

```
PORTD =  $\sim$ PORTD;
```

```
// PORTD hat jetzt den Wert B01010101
```

```
// -> Die LEDs an den Pins 6 4 2 0 leuchten
```

# Tinkercad Workshop

## Ausgänge schalten mit PORT

```
//C++ Ausgänge schalten mit PORT

int Leuchtdauer = 200;
void setup()
{
  // Pins 7 bis 0 als OUTPUT definieren
  DDRD = B11111111;
}
void loop()
{
  // Lauflicht hin Start mit LED Pin 7
  PORTD = B10000000;
  delay(Leuchtdauer);
  for (int i = 0; i < 7; i++)
  {
    /*
    1 Bit nach rechts schieben
    B01000000 -> Pin 6
    B00100000 -> Pin 5
    B00010000 -> Pin 4
    B00001000 -> Pin 3...
    */
    PORTD = PORTD >> 1;
    delay(Leuchtdauer);
  }
  // Lauflicht zurück Start mit LED Pin 0
  PORTD = B00000001;
  delay(Leuchtdauer);
  for (int i = 0; i < 7; i++)
  {
```

```
    /*
    1 Bit nach links schieben
    B00000010 -> Pin 1
    B00000100 -> Pin 2
    B00001000 -> Pin 3
    B00010000 -> Pin 4...
    */
    PORTD = PORTD << 1;
    delay(Leuchtdauer);
  }
  delay(Leuchtdauer);
  // alle LEDs
  PORTD = B11111111;
  delay(1000);
  // alle aus
  PORTD = B00000000;
  delay(1000);
  // LEDs an den Pins 7 5 3 leuchten
  PORTD = B10101010;
  delay(1000);
  // umkehren die inneren LEDs leuchten
  PORTD = ~PORTD;
  delay(1000);
  // die beiden äußeren
  PORTD = B10000001;
  delay(1000);
  // die inneren
  PORTD = ~PORTD;;
  delay(1000);
  // nur die beiden mittleren
  PORTD = B00011000;
  delay(1000);
  // alle aus
  PORTD = B00000000;
  delay(1000);
}
```

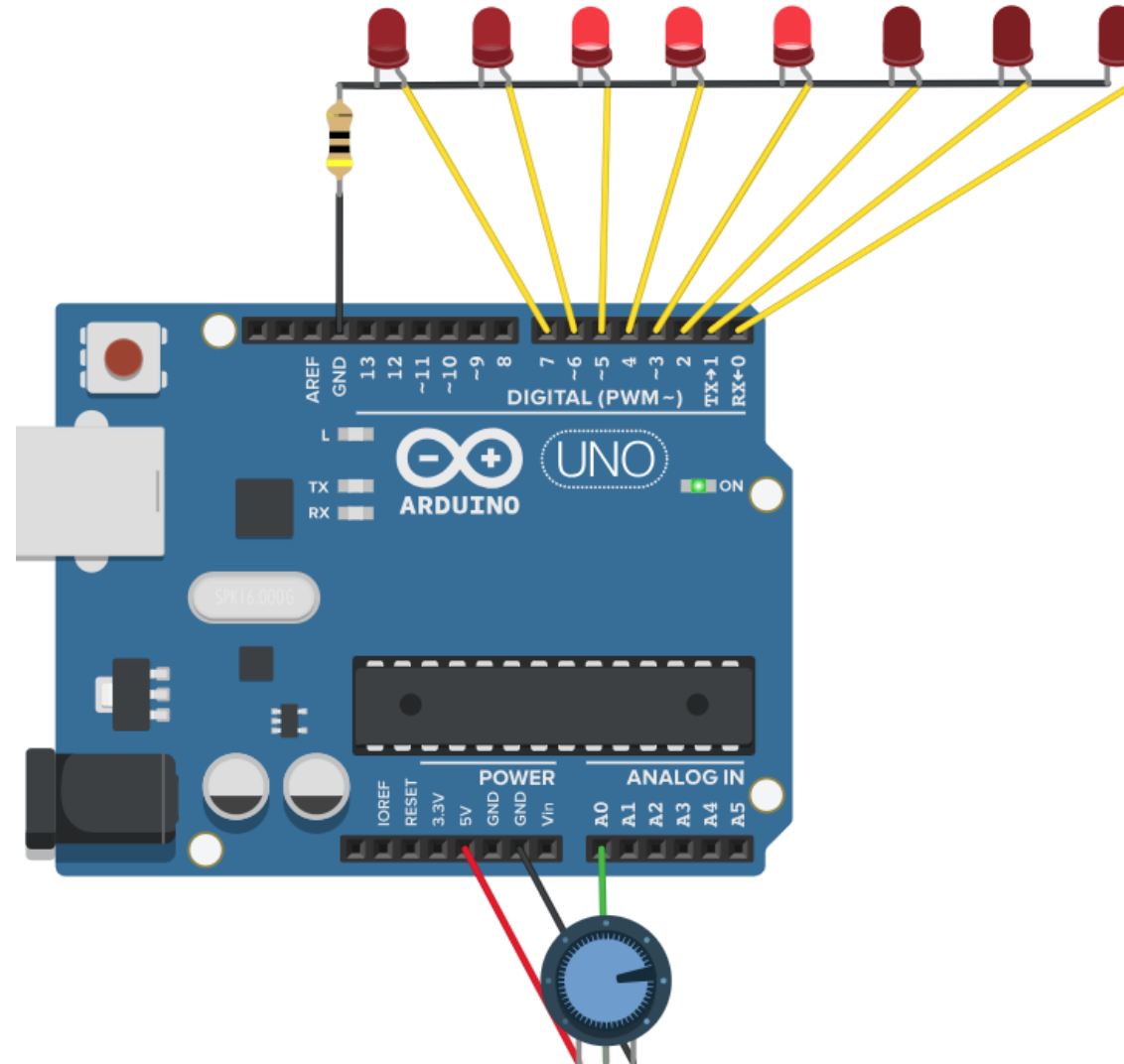


# Tinkercad Workshop



## KITT Laufflicht mit PORT-Register

Jeweils drei LEDs sollen von links nach rechts und zurück laufen.  
Die Geschwindigkeit ist mit dem Poti von 150-400ms einstellbar.  
Schreibe das Programm in Text.

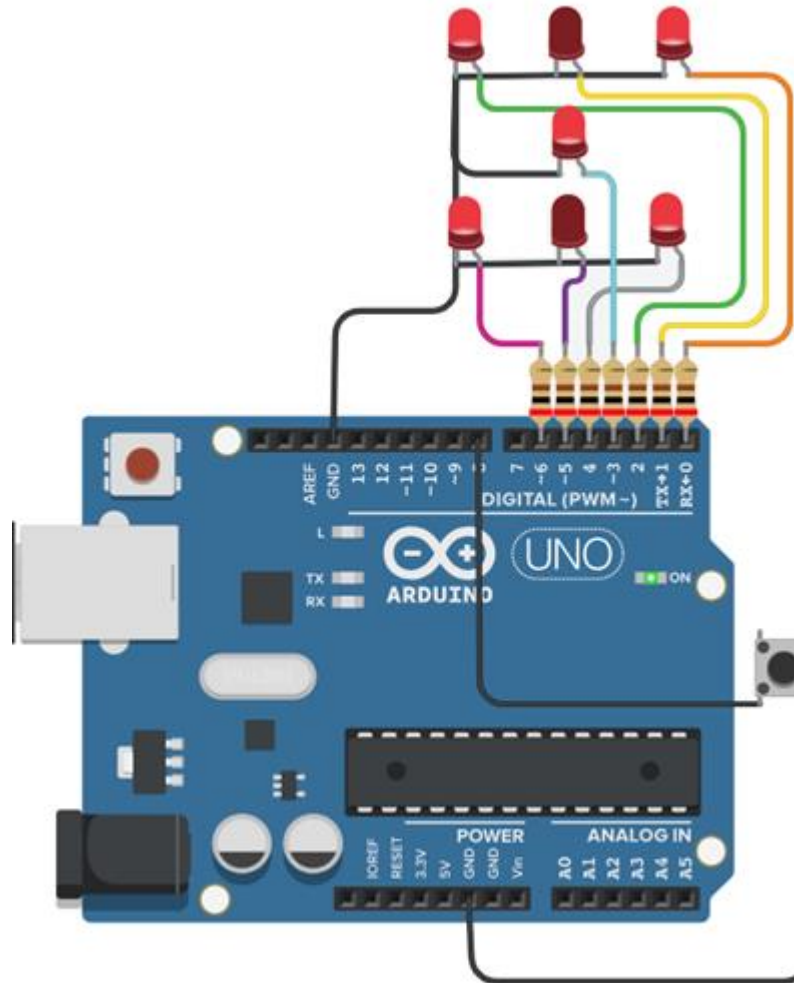


# Tinkercad Workshop



## Würfel mit PORT-Register

Mit einem Tastendruck werden die Zufallszahlen 1-6 erzeugt und mit den LEDs angezeigt.  
Schreibe das Programm in Text.



# Tinkercad Workshop



## Stepper Motor mit WOKWI

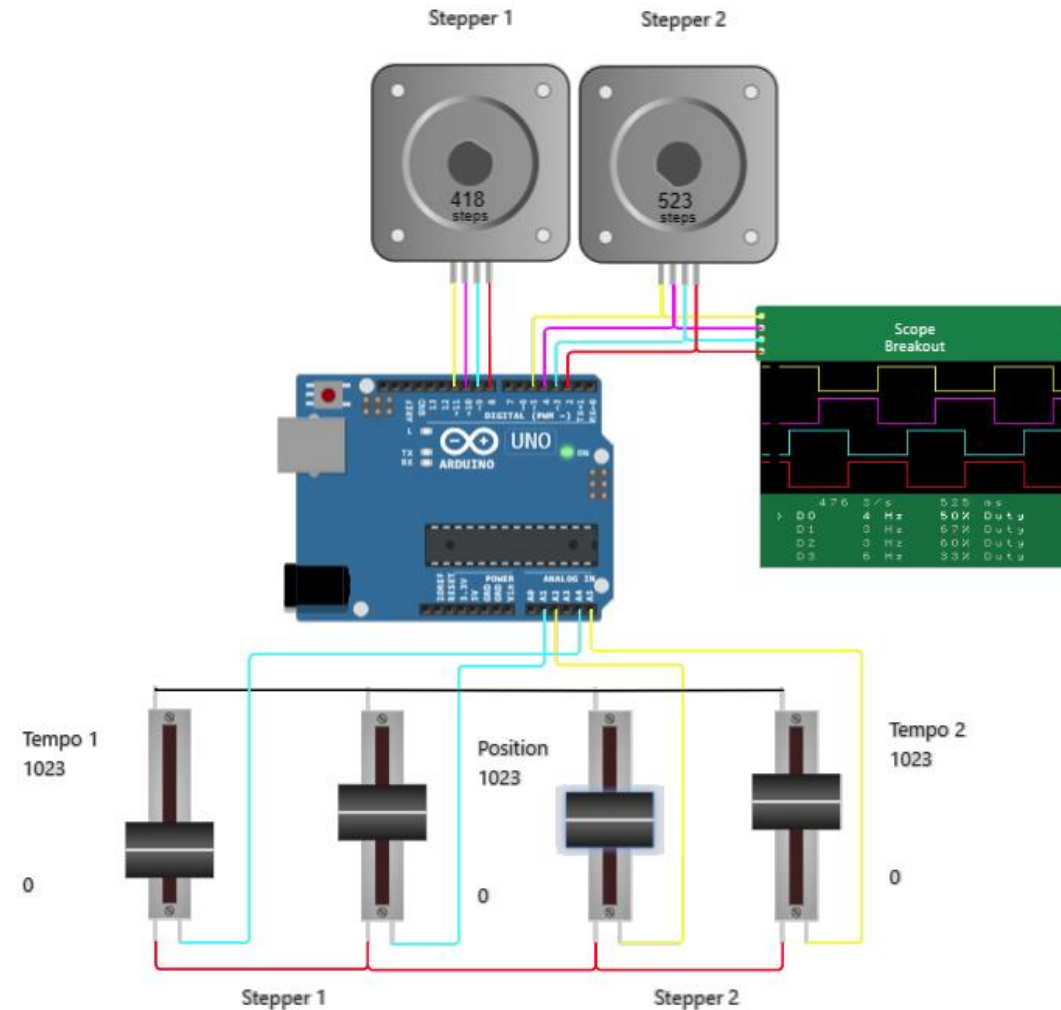
Der WOKWI-Simulator ist eine gute Alternative zum TINKERCAD-Simulator.

Teste den WOKWI-Simulator z.B. mit einem Stepper-Programm.

Link zum WOKWI-Simulator:

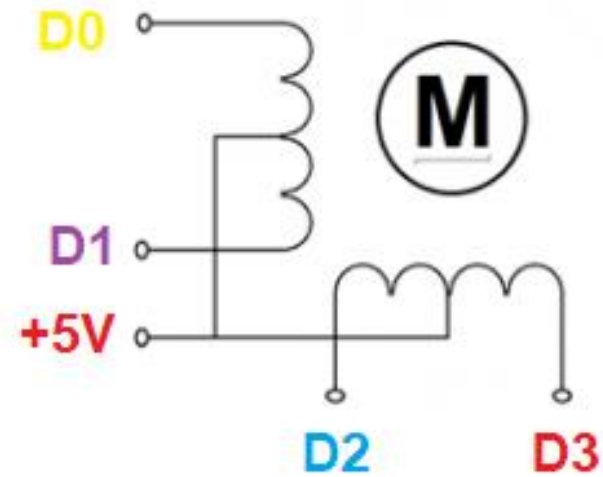
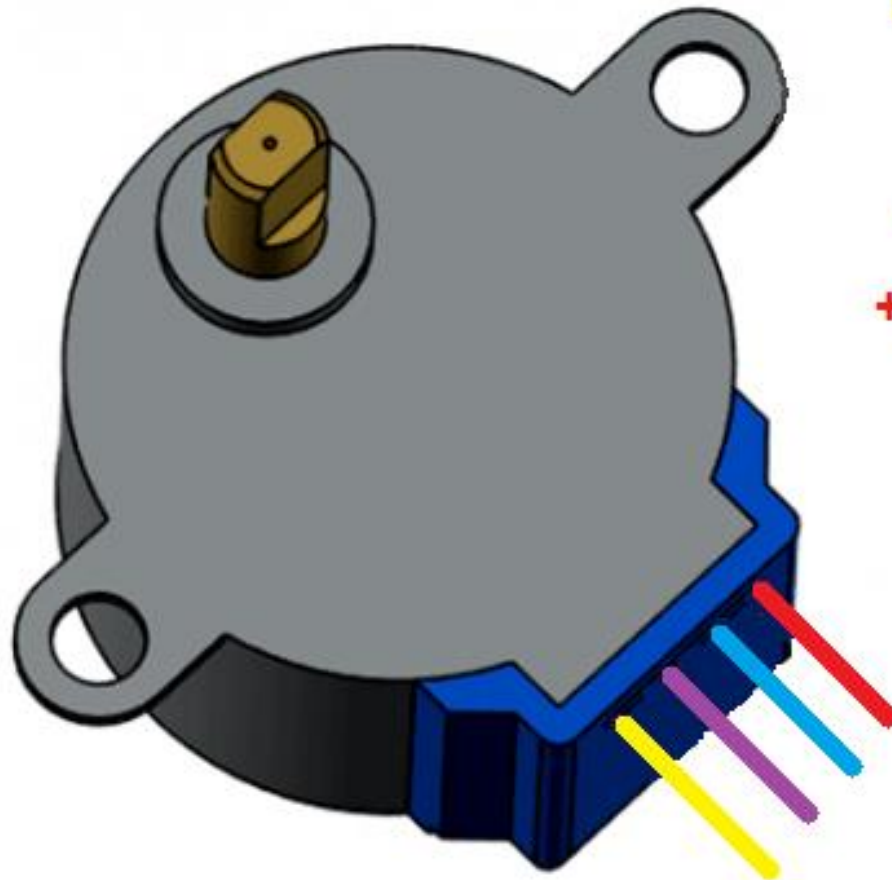
<https://wokwi.com/projects/new/arduino-uno>

Die Schritte der Stepper-Motore folgen den Positions-Poti.  
Mit den Tempo-Potis kann die Geschwindigkeit eingestellt werden.



# Tinkercad Workshop

## Stepper Motor mit WOKWI

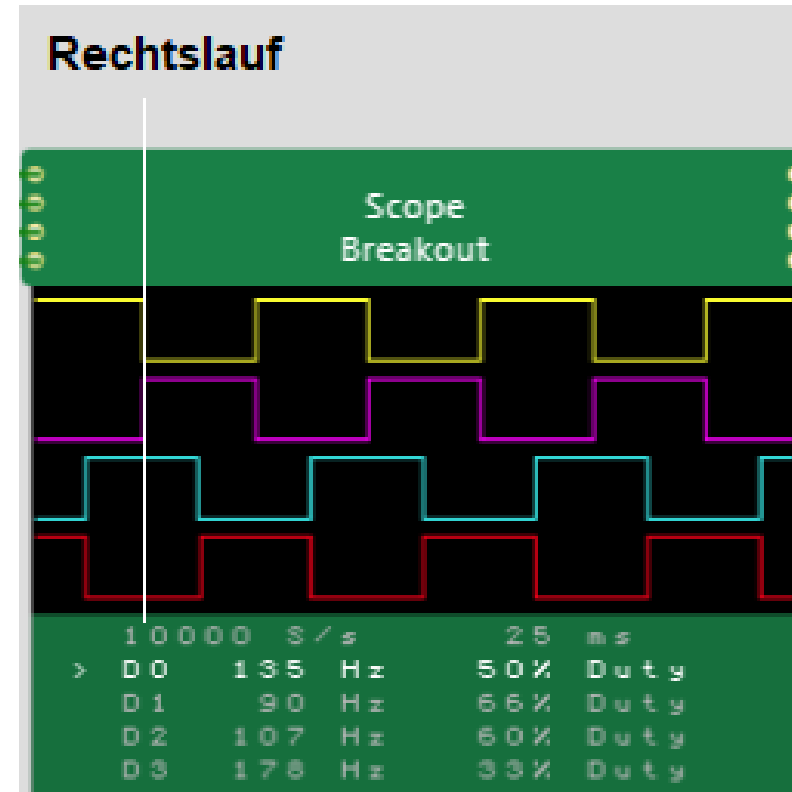
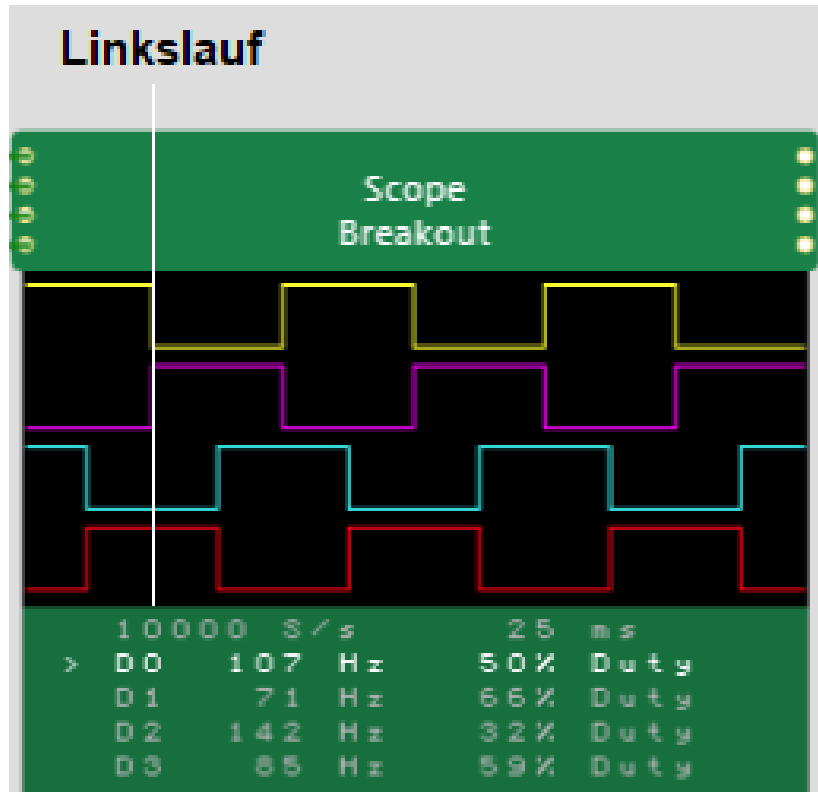


Stepper Schema



# Tinkercad Workshop

## Stepper Motor mit WOKWI



# Tinkercad Workshop



## Stepper Motor mit WOKWI

Die Schritte der Stepper-Motore folgen den Positions-Poti. Mit den Tempo-Potis kann die Geschwindigkeit eingestellt werden.

```
//C++ Stepper-Motor mit Poti
```

```
#include <AccelStepper.h>
```

```
AccelStepper stepper1(AccelStepper::FULL4WIRE, 8, 9, 10, 11);  
AccelStepper stepper2(AccelStepper::FULL4WIRE, 2, 3, 4, 5);
```

```
void setup()  
{  
    stepper1.setMaxSpeed(1000);  
    stepper2.setMaxSpeed(1000);  
}
```

```
void loop()  
{  
    int analog_in1 = analogRead(A1);  
    int TempoPot1 = analogRead(A4);  
    stepper1.moveTo(analog_in1);  
    stepper1.setSpeed(TempoPot1/30);  
    stepper1.runSpeedToPosition();  
  
    int analog_in2 = analogRead(A2);  
    int TempoPot2 = analogRead(A5);  
    stepper2.moveTo(analog_in2);  
    stepper2.setSpeed(TempoPot2/30);  
    stepper2.runSpeedToPosition();  
}
```

# Tinkercad Workshop

**Wir wünschen dir viel Spaß beim Programmieren!**

