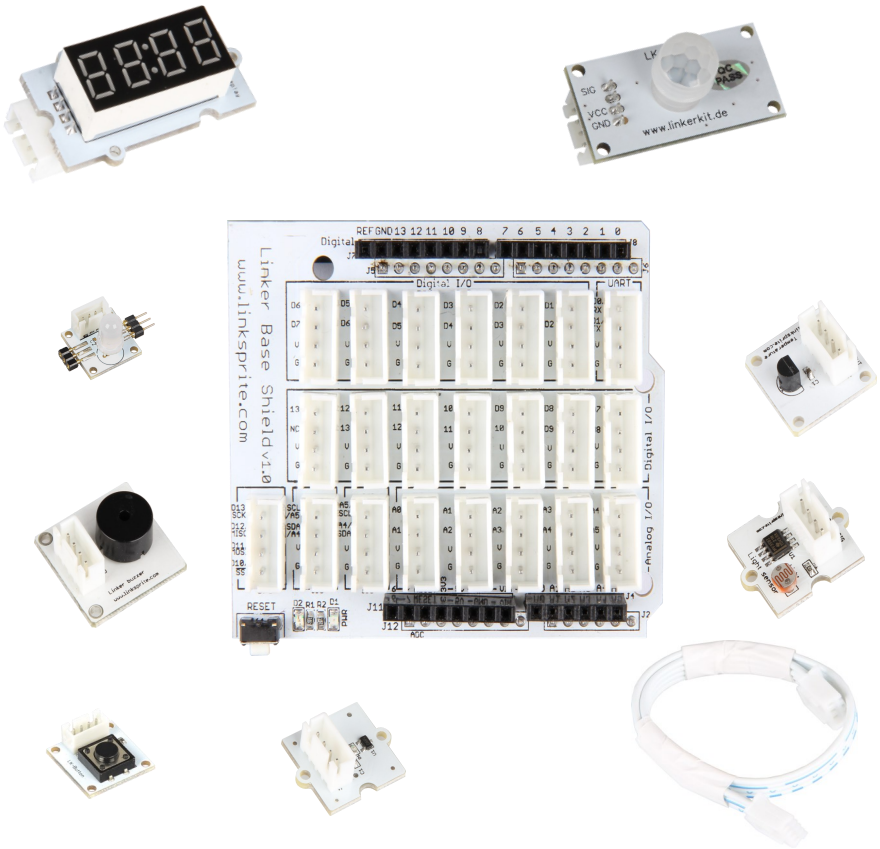
LINKERKIT BASE SET

für Arduino



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.

Das Linker Kit Base Set für den Arduino macht die Verwendung verschiedener Module und Sensoren sehr einfach. Durch das Linker Kit Stecksystem ist eine Verpolung der einzelnen Module ausgeschlossen.
In dieser Anleitung finden Sie detaillierte Erklärungen und Beispiele zu allen in diesem Set enthaltenden Module.

Probieren sie es aus und erweitern sie die Beispiele nach Ihren eigenen Vorstellungen.

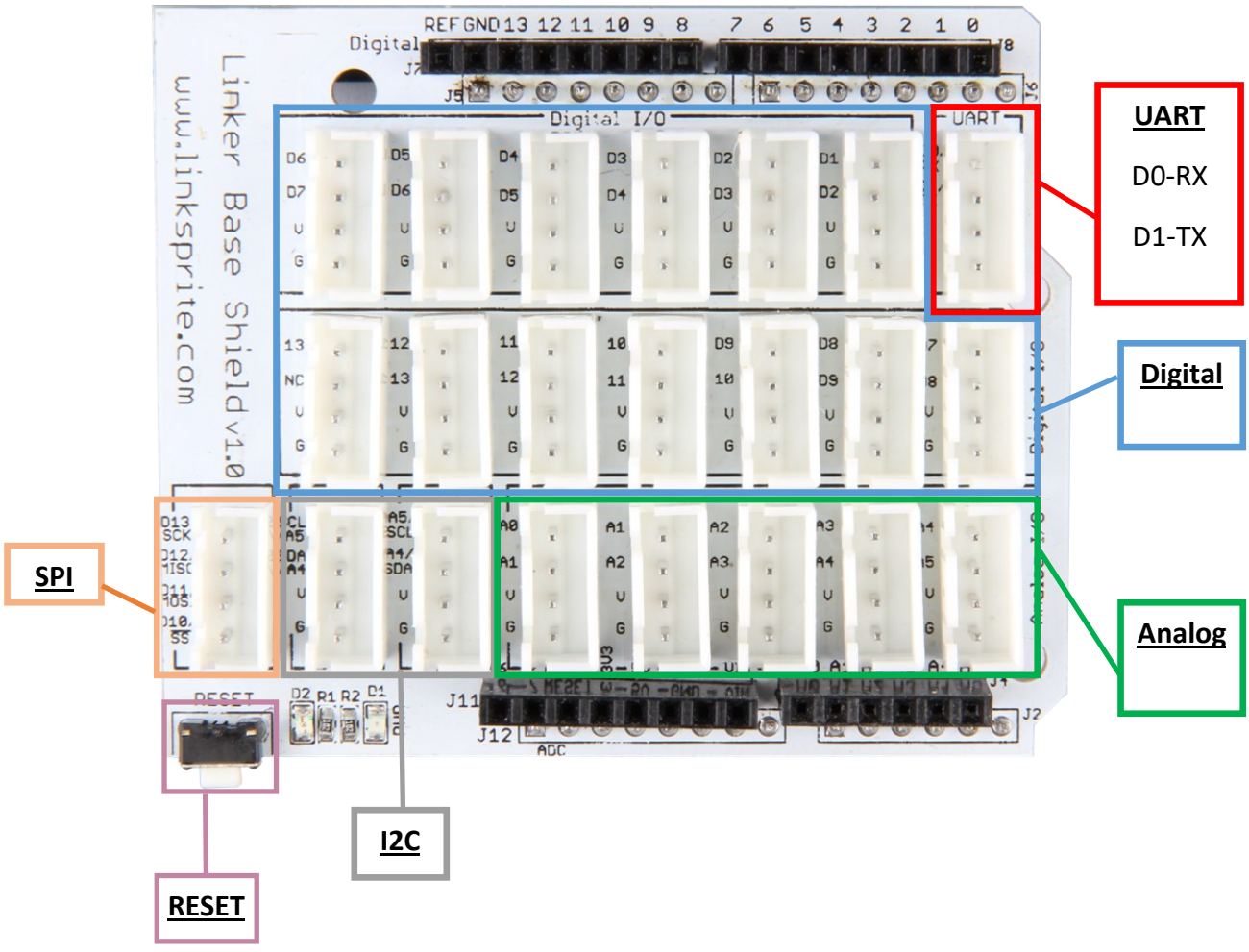
Im Lieferumfang enthalten sind:

Baseboard für Arduino Uno, Buttonmodul, Buzzermodul,
5x Kabel, Digitalanzeigemodul, Hallsensor, RGB LED Modul,
Lichtsensor, Bewegungssensor, Temperatursensor

2. DAS BASESHIELD

Das Baseboard ist das Herzstück dieses Sets, es wird einfach auf Ihren Arduino Uno gesteckt und ist sofort einsatzbereit.
Es können über 18 Module an das Board angeschlossen werden.
Neben den bekannten UART, SPi und I2C können noch 13 weitere digitale und 5 Analoge Module angeschlossen werden.
Der Reset-Button des Arduino wurde um ihn leichter bedienen zu können ebenfalls auf das Baseshield gelegt.
Außerdem werden die Anschlüsse des Arduino durchgeführt.

Dem nachfolgendem Bild können Sie einen detaillierten Anschlussplan für das Baseshield entnehmen

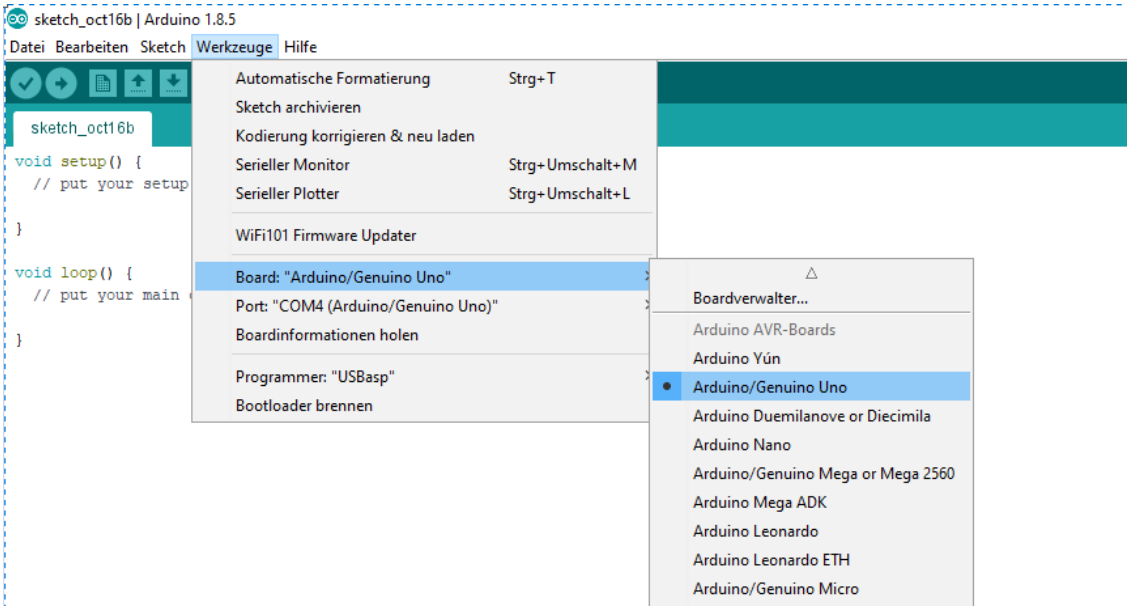


3. DIE ARDUINO IDE

Um den Arduino Uno programmieren zu können, benötigen Sie zunächst die entsprechende Arduino Software. Diese ist hier erhältlich. Laden Sie das Softwarepaket (Arduino IDE) herunter und installieren Sie dieses.

Nachdem Sie die Software gestartet haben muss diese auf das Board eingestellt werden.

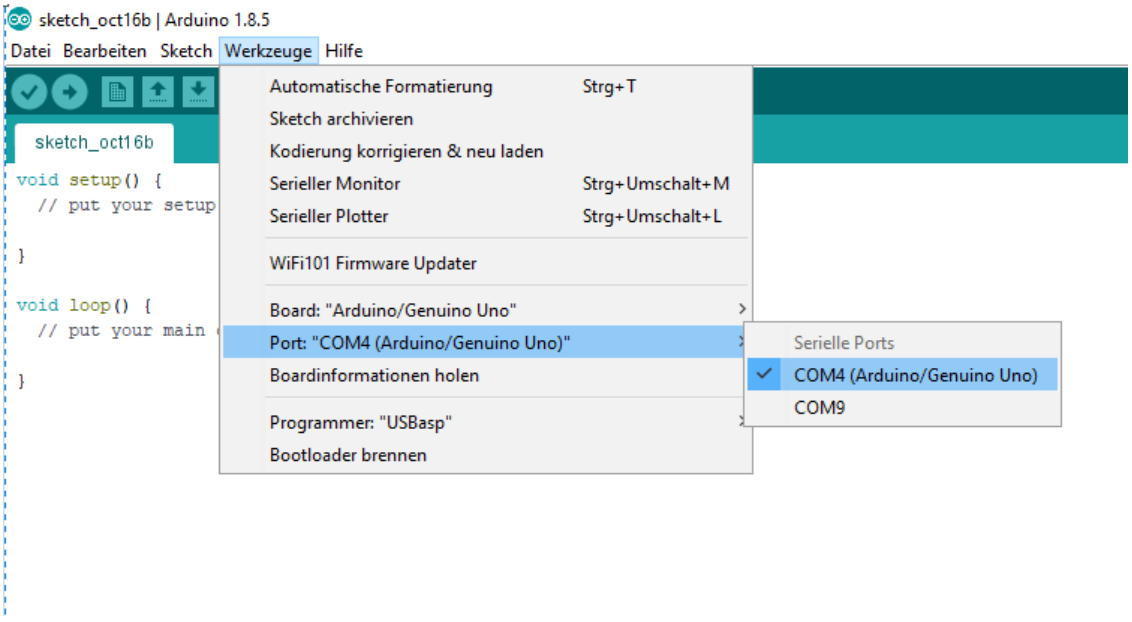
Öffnen Sie dazu den Reiter „**Werkzeuge**“ und wählen Sie unter „**Board**“ das „**Arduino/Genuino UNO**“-Board aus.



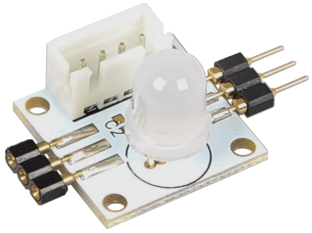
Nun müssen Sie noch den richtigen Port auswählen.

Öffnen Sie dazu den Reiter „**Werkzeuge**“ und wählen Sie unter „**Port**“ den „**COM4**“-Port aus.

Achtung!!! Der Port kann variieren und muss nicht immer „COM4“ sein

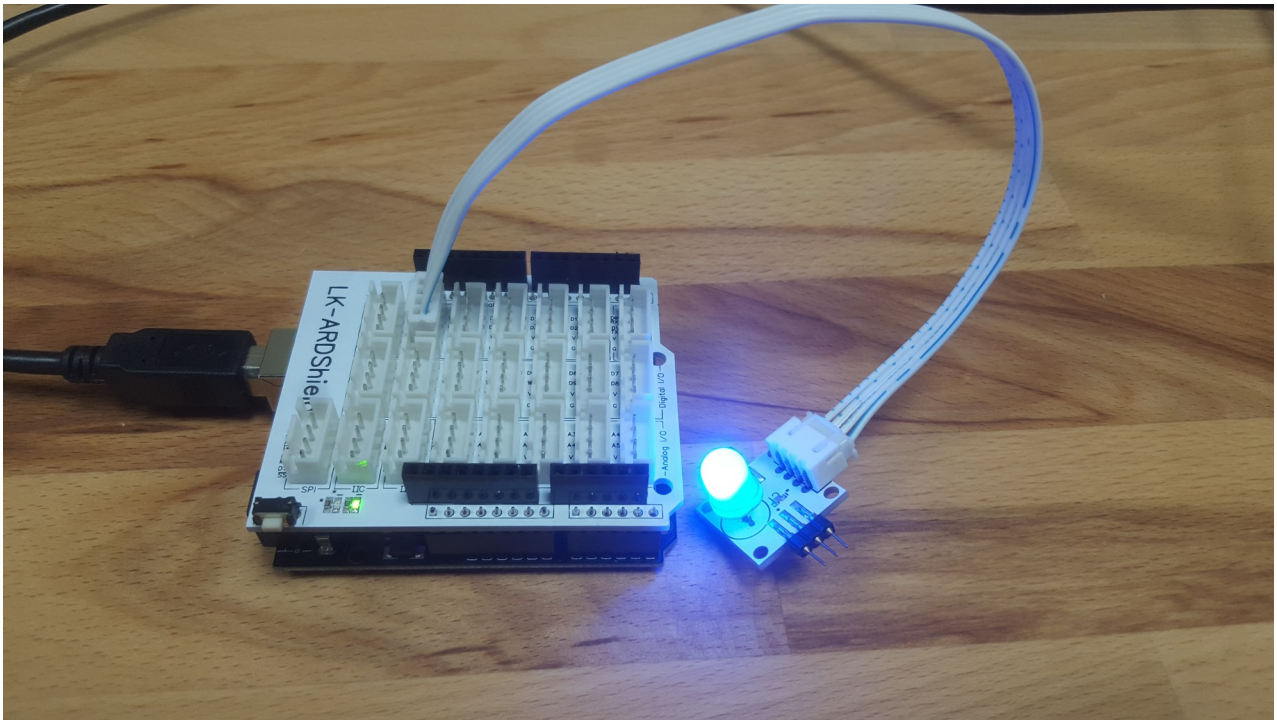


4. DIE RGB-LED



Egal ob zur Beleuchtung oder zur Statusanzeige die LED gehört in jedes Set.

Mehrere LEDs lassen sich auch zusammen stecken und gemeinsam ansteuern.



Die LED kann in vielen verschiedenen Farben leuchten.

4. DIE RGB-LED

In diesem Beispiel zeigen wir wie Sie die LED blinken lassen können. Dafür müssen Sie zunächst die benötigte Bibliothek herunterladen. Diese plus mehrere nützliche Beispieldateien laden Sie hier herunter.

Laden Sie die „Source code (zip)“-Datei der neusten FastLED Version herunter.

Entpacken Sie die Datei und speichern Sie den Ordner, im Ordner „libraries“, in Ihrem Arduino-Verzeichnis. Wichtig ist außerdem, dass Sie den Namen des heruntergeladenen Ordners in „FastLED“ ändern.

Öffnen Sie nun die „Blink“- Beispieldatei aus dem Ordner:

Arduino/libraries/FastLED/examples/Blink/

Diese Datei können Sie nun auf Ihren Arduino hochladen und ausführen.

Schließen Sie dafür die LED an Digital-Pin 3 an.

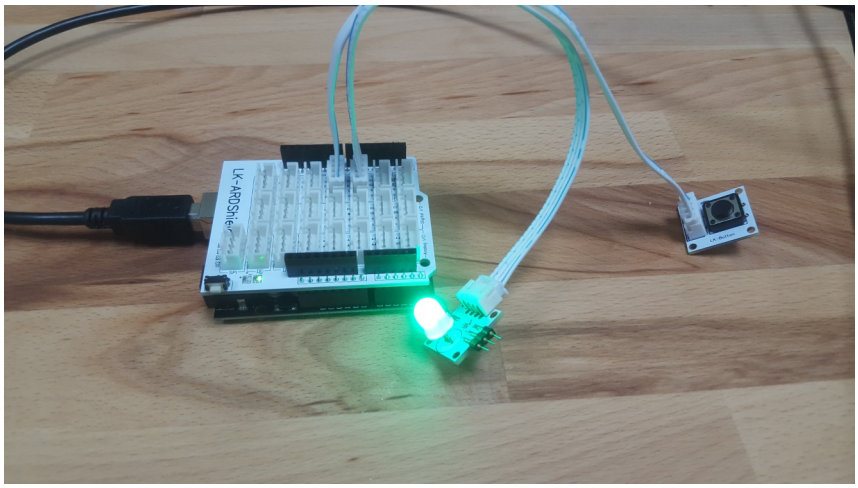
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
#include <FastLED.h>

// Anzahl der LEDs
#define NUM_LEDS 1

#define DATA_PIN 3

CRGB leds[NUM_LEDS];

void setup() {

    FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);

}

void loop() {
  // LED an dann halbe Sekunde Pause
  leds[0] = CRGB::Red;
  FastLED.show();
  delay(500);
  // LED aus dann halbe Sekunde Pause
  leds[0] = CRGB::Black;
  FastLED.show();
  delay(500);
        }
```

5. DAS BUTTON MODUL

Das Buttonmodul eignet sich hervorragend zum steuern anderer Module und kann für viele verschiedene

Aufgaben verwendet werden.



In diesem Beispiel kombinieren wir die bereits kennengelernte LED mit dem Buttonmodul.

Wir werden diesmal die LED mittels Knopfdruck zum Leuchten bringen.

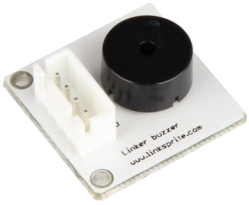
Schließen Sie die LED an Digital-Pin 3 und den Button an Digital-Pin 2 an.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

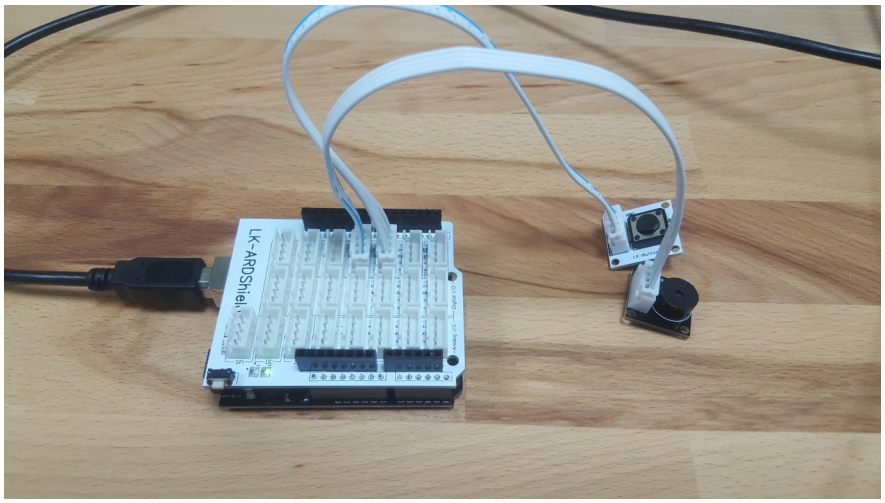
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
#include <FastLED.h>
#define NUM_LEDS 1
#define DATA_PIN 3
#define BUTTON_PIN 2
int buttonstate = 0;
CRGB leds[NUM_LEDS];

void setup() {
      FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);
}
void loop() {
  buttonstate = digitalRead(BUTTON_PIN);

// überprüfe ob Button gedrückt wurde
  if (buttonstate == HIGH){
    leds[0] = CRGB::Red;
    FastLED.show();
  } else{
    leds[0] = CRGB::Black;
    FastLED.show();
  }
                        }
```

6. DAS BUZZER MODUL



Dieser Buzzer kann an Digitale Ausgänge angeschlossen werden, um einen Ton zu erzeugen. Alternativ kann er an einem Analogen Impulsbreitenmodulationsausgang angeschlossen werden um unterschiedliche Töne und Effekte zu erzeugen



In diesem Beispiel nehmen wir das das Buttonmodul und kombinieren es mit dem Buzzermodul, sodass der Buzzer, bei jedem drücken des Knopfes ertönt.

Der Buzzer wird in diesem Beispiel an dem Steckplatz mit dem digitalen Port 3 und der Button an den mit dem digitalen Port 2 angeschlossen.

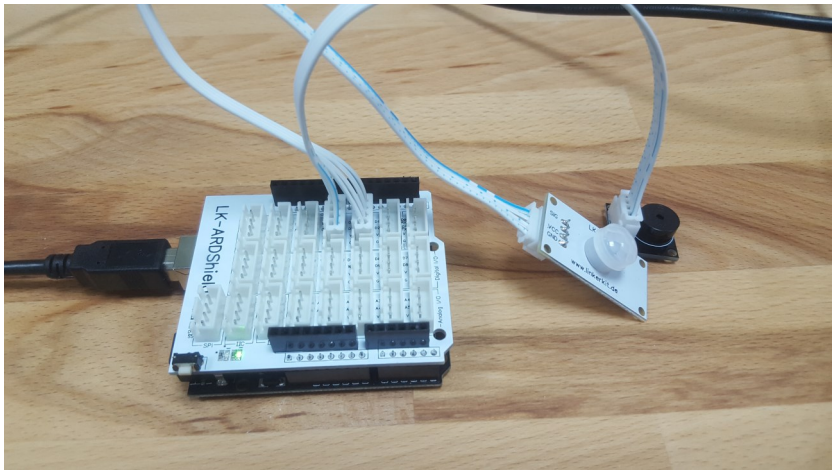
Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
const int buttonPin = 2;    // definiere Button Pin
int Buzzer1 = 3;

// variables will change:
int buttonState = 0;        // variable um pushbutton status zu lesen

void setup() {
  //initialisiere den Buzzer als Output
  pinMode(Buzzer1, OUTPUT);
  //initialisiere den Button als Input
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
     tone(Buzzer1,400,200);
     delay(500);
  }
```

7. DER BEWEGUNGSSENSOR

Dieser Bewegungssensor hat einen Erfassungswinkel von 120° und eine Reichweite von bis zu 6 Metern.
Mit diesem Sensor kann man zum Beispiel Lampen bauen, die sich automatisch einschalten wenn man den Raum betritt.

Bewegungssensoren werden auch oft in Alarmanlagen eingesetzt.



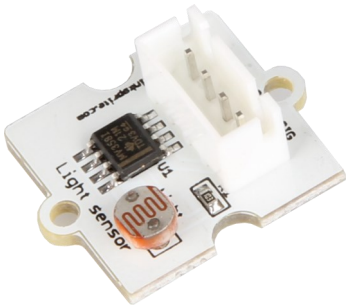
In diesem Beispiel Kombinieren wir den Bewegungssensor mit dem Buzzer, sodass sobald das Modul eine Bewegung erkennt ein Alarmton ausgegeben wird.
Verbinden Sie den Bewegungssensor mit Digital-Pin 2 und den Buzzer mit Digital-Pin 3.

Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.
Speichern Sie die Datei am Besten in Ihrem Dokumente Ordner unter dem Namen Pir.py

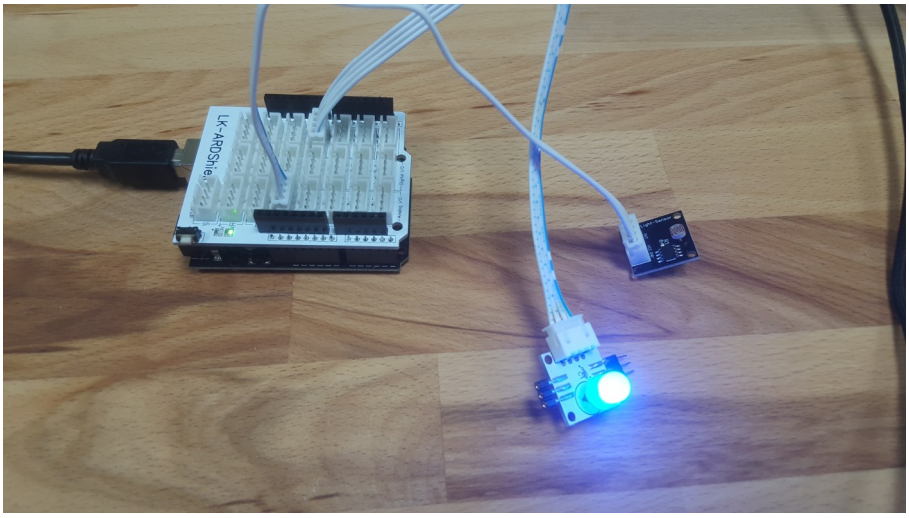
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
// set pin numbers:
const int sensorPin= 2;    // Port des Sensorpins
int Buzzer1 = 3;

int sensorState = 0;       // Variable für Sensorstatus

void setup() {
  // Initialisierung des Piezos als Ausgang
  pinMode(Buzzer1, OUTPUT);
  // Initialisiere Sensor-Pin als Eingang
  pinMode(sensorPin, INPUT);
}
void loop(){
  // den Zustand des Sensorwertes lesen:
  sensorState = digitalRead(sensorPin);

  // Überprüft, ob der Sensor aktiviert ist.
  if (sensorState == HIGH) {
     tone(Buzzer1,400,200);
     delay(1000);
  }
                        }
```

8. DER LICHTSENSOR



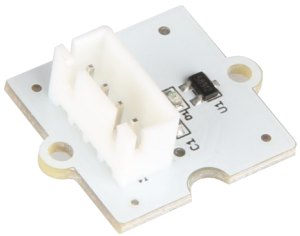
Dieser Lichtsensor ist ein vom Licht abhängiger Widerstand (LDR). Der Widerstand des Sensors verringert sich, sobald die Helligkeit in der Umgebung zu nimmt.
Mit Hilfe dieses Sensors kann man eine Lampe bauen, die automatisch an geht sobald es dunkel wird.



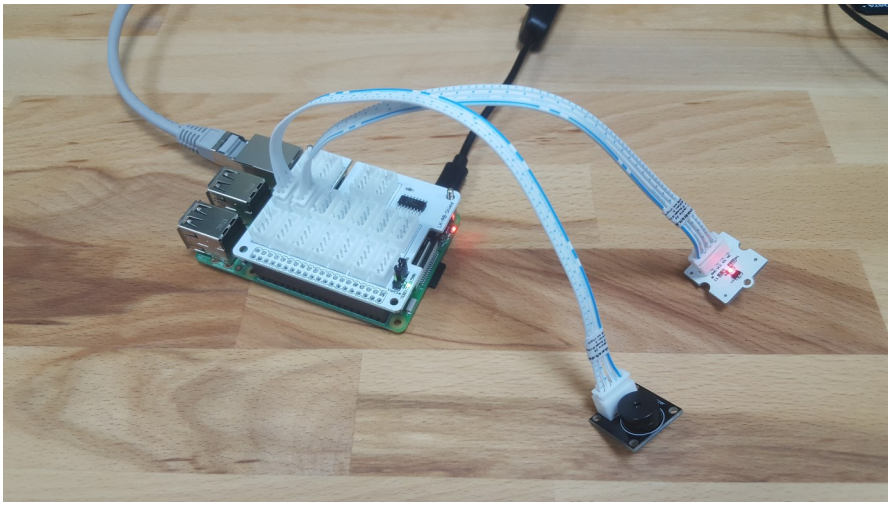
In diesem Beispiel kombinieren wir den Lichtsensor mit der LED, um die LED einzuschalten sobald es dunkel wird. Um Dunkelheit zu simulieren können sie den Lichtsensor einfach mit der Hand abdecken.
Schließen Sie die LED an Digital-Pin 3 und den Lichtsensor an Analog-Pin 0 an.
Bitte kopieren sie den Code vollständig und fügen Sie ihn in Ihr Skript ein.
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
#include <FastLED.h>
#define NUM_LEDS 1
const int ledPin=3;                    // Verbindet die LED mit Digital Pin 3
const int thresholdvalue=10;        // Setzt die Schwelle wann die LED angehen soll. Wenn es gering
                                    // eingestellt wird, dann leuchtet die LED bei viel Licht.
                                 // Wenn es hoch eingestellt wird, leuchtet die LED bei Dunkelheit.
CRGB leds[NUM_LEDS];
void setup() {
  Serial.begin(9600);              //Startet die serielle Verbindung
   FastLED.addLeds<NEOPIXEL, ledPin>(leds, NUM_LEDS);
}
void loop() {
  int sensorValue = analogRead(0);  // Verbindet das Lichtsensor Modul mit A0, Analog 0
  float Rsensor;
  Rsensor=(float)(1023-sensorValue)*10/sensorValue;
  if(Rsensor>thresholdvalue)
  {
   leds[0] = CRGB::Red;
    FastLED.show();
 }
  else
  {
  leds[0] = CRGB::Black;
    FastLED.show();
  }
    Serial.println(Rsensor,DEC);
}
```

9. DER HALLSENSOR



Mit diesem Hallsensor können Sie Magnetfelder in der Umgebung, mit Hilfe des Hall-Effekts, erfassen.

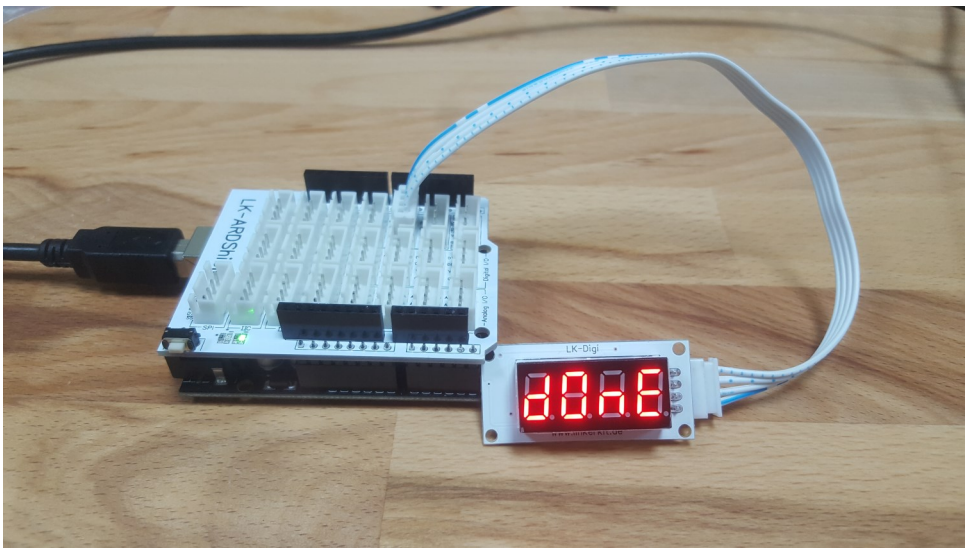


In diesem Beispiel kombinieren wir den Hallsensor und den Buzzer. Wenn der Sensor ein Magnetfeld erkennt wird der Buzzer ein Signal von sich geben.
Verbinden Sie den Hallsensor mit Digital-Pin 2 und den Buzzer mit Digital-Pin 3.
Bitte kopieren sie den folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.
Probieren Sie es aus und erweitern Sie den Code nach Ihren eigenen Vorstellungen.

```
const int HallPin = 2;     // Hallsensor pin
int Buzzer1 = 3;

int SensorState = 0;       // Variabel für den Sensorstatus

void setup() {
  // Initialisiere den Piezo als Ausgang
  pinMode(Buzzer1, OUTPUT);
  // Initialisiere den Sensor als Eingang
  pinMode(HallPin, INPUT);

}

void loop(){
  // Lese Sensorwert aus
  SensorState = digitalRead(HallPin);

  // überprüfen ob Sensor aktiviert wurde
  if (SensorState == LOW) {
     tone(Buzzer1,400,200);
     delay(500);
  }
                                        }
```

10. DIE DIGITALANZEIGE

Mit Hilfe der Digitalanzeige können wir uns die Zeit oder andere Daten anzeigen lassen. Sie wird in vielen industriellen Lösungen wie Aufzügen verwendet! Und wird für Sie in der Zukunft mit Sicherheit von Nutzen sein.



In diesem Beispiel zeigen wir wie Sie etwas auf der Digitalanzeige anzeigen lassen können. Dafür müssen Sie zunächst die benötigte Bibliothek herunterladen. Diese plus eine
nützliche Beispieldatei laden Sie hier herunter.

Laden Sie die „Source code (zip)“-Datei herunter und entpacken Sie diese.

Speichern Sie den Ordner, im Ordner „libraries“, in Ihrem Arduino-Verzeichnis.

Öffnen Sie nun die „TM1637Test“- Beispieldatei aus dem Ordner:

Arduino/libraries/TM163-1.1.0/examples/TM1637Test/

Diese Datei können Sie nun auf Ihren Arduino hochladen und ausführen.

Schließen Sie dafür die Digitalanzeige an Digital-Pin 2 und 3 an.

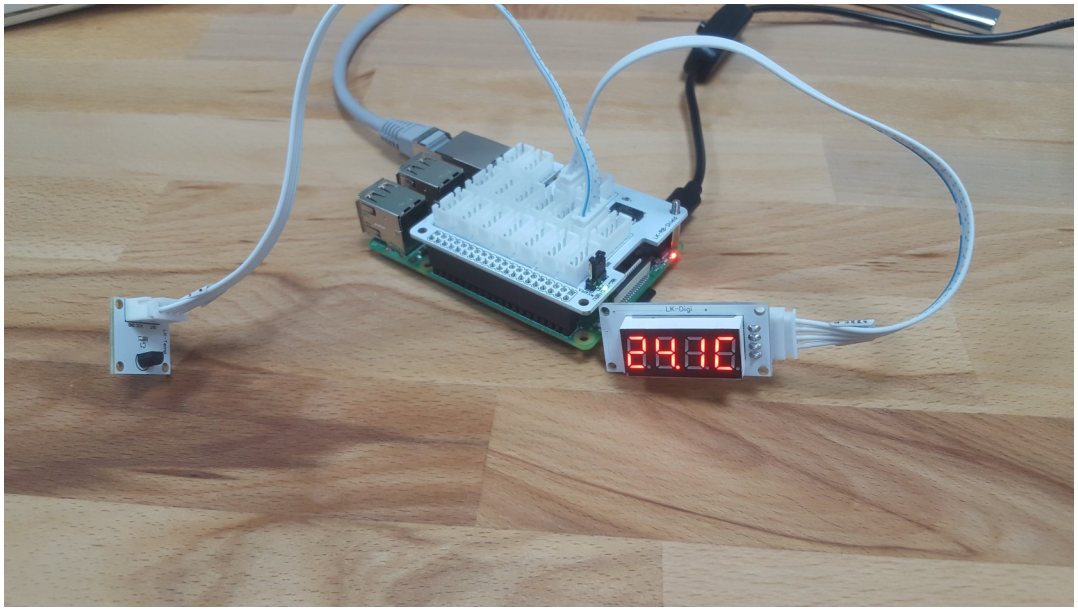
Probieren Sie es aus und erweitern Sie den Code, der auf den folgenden 2 Seiten aufgeführt ist, nach Ihren eigenen Vorstellungen.

```
#include <Arduino.h>
#include <TM1637Display.h>

// Module connection pins (Digital Pins)
#define CLK 2
#define DIO 3

// The amount of time (in milliseconds) between tests
#define TEST_DELAY   2000

const uint8_t SEG_DONE[] = {
    SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,         // d
    SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,   // O
    SEG_C | SEG_E | SEG_G,                         // n
    SEG_A | SEG_D | SEG_E | SEG_F | SEG_G          // E
    };

TM1637Display display(CLK, DIO);

void setup()
{
}

void loop()
{
  int k;
  uint8_t data[] = { 0xff, 0xff, 0xff, 0xff };
  display.setBrightness(0x0f);

  // All segments on
  display.setSegments(data);
  delay(TEST_DELAY);

  // Selectively set different digits
  data[0] = 0b01001001;
  data[1] = display.encodeDigit(1);
  data[2] = display.encodeDigit(2);
  data[3] = display.encodeDigit(3);

  for(k = 3; k >= 0; k--) {
     display.setSegments(data, 1, k);
     delay(TEST_DELAY);
     }

  display.setSegments(data+2, 2, 2);
  delay(TEST_DELAY);
display.setSegments(data+2, 2, 1);
  delay(TEST_DELAY);
  display.setSegments(data+1, 3, 1);
  delay(TEST_DELAY);
```

```
// Show decimal numbers with/without leading zeros
bool lz = false;
for (uint8_t z = 0; z < 2; z++) {
  for(k = 0; k < 10000; k += k*4 + 7) {
          display.showNumberDec(k, lz);
          delay(TEST_DELAY);
  }
  lz = true;
}
// Show decimal number whose length is smaller than 4
for(k = 0; k < 4; k++)
  data[k] = 0;
display.setSegments(data);
  // Run through all the dots
  for(k=0; k <= 4; k++) {
          display.showNumberDecEx(0, (0x80 >> k), true);
          delay(TEST_DELAY);
  }
display.showNumberDec(153, false, 3, 1);
delay(TEST_DELAY);
display.showNumberDec(22, false, 2, 2);
delay(TEST_DELAY);
display.showNumberDec(0, true, 1, 3);
delay(TEST_DELAY);
display.showNumberDec(0, true, 1, 2);
delay(TEST_DELAY);
display.showNumberDec(0, true, 1, 1);
delay(TEST_DELAY);
display.showNumberDec(0, true, 1, 0);
delay(TEST_DELAY);

// Brightness Test
for(k = 0; k < 4; k++)
   data[k] = 0xff;
for(k = 0; k < 7; k++) {
  display.setBrightness(k);
  display.setSegments(data);
  delay(TEST_DELAY);
}

// On/Off test
for(k = 0; k < 4; k++) {
  display.setBrightness(7, false);  // Turn off
  display.setSegments(data);
  delay(TEST_DELAY);
  display.setBrightness(7, true); // Turn on
  display.setSegments(data);
  delay(TEST_DELAY);
}
// Done!
display.setSegments(SEG_DONE);
while(1);
                }
```

11. DER TEMPERATURSENSOR

Dieser Temperatursensor verwendet einen Thermistor, um die Umgebungstemperatur zu erfassen. Der Widerstand des Thermistors erhöht sich, wenn die Umgebungstemperatur abnimmt. Durch diese Charakteristik wird die Umgebungstemperatur ausgemessen.



In diesem Beispiel kombinieren wir den Temperatursensor mit der Digitalanzeige um uns die aktuelle Raumtemperatur anzeigen zulassen.

Der Temperatursensor wird an Analog-Pin 0 und Der Display an Digital-Pin 2 und 3 angeschlossen.

Bitte kopieren sie den auf dieser und auf der nächsten Seite folgenden Code vollständig und fügen Sie ihn in Ihr Skript ein.

Probieren Sie es aus und erweitern Sie den Code auf der nächsten Seite, nach Ihren eigenen Vorstellungen.

```
//TMP36 Pin-Variablen
#include <Arduino.h>
#include <TM1637Display.h>
int sensorPin = 0;     // Der analoge Pin TMP36's Vout, ist mit der Auflösung verbunden
                       // Sie beträgt 10 mV / Grad Celsius mit einem
                       // 500 mV ausgleich um negative Temperaturen zu ermöglichen
#define CLK 2
#define DIO 3
#define TEST_DELAY   2000

TM1637Display display(CLK, DIO);
/*
 * setup() - Diese Funktion startet einmal, wenn der Arduino gestartet wird.
 * Wir initialisieren die Verbindung zum Computer.
 */
void setup()
{
 Serial.begin(9600);  // Startet die serielle Verbindung zum Computer
                      // Um das Ergebnis anzuzeigen müssen Sie ein Monitor öffnen
}

void loop()                  // Immer wieder ausführen (Wiederholschleife)
{
 // Spannung wird anhand des Temperatur-Sensors erfasst.
 int reading = analogRead(sensorPin);

 // Konvertierung der Ausgelesenen Spannung, 3.3V Arduino verwendet 3.3
 float voltage = reading * 5.0;
 voltage /= 1024.0;

 // berechnen der Temperatur
 int temperatureC = (voltage - 0.5) * 1000 ;

 // gebe Temperatur über serielle Konsole aus
Serial.print(temperatureC); Serial.println(" degrees C");


display.setBrightness(0x0f); //einstellen der Display Helligkeit

display.showNumberDecEx(temperatureC, (0x80 >> 2), true,3,0);    // Zeige Temperatur auf
dem Display

display.setSegments(0b01100110, 1, 3);       //Füge das C hinten an



delay(3000);   //drei Sekunde Warten


                              }
```

12. 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.

13. 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: 01.04.2020