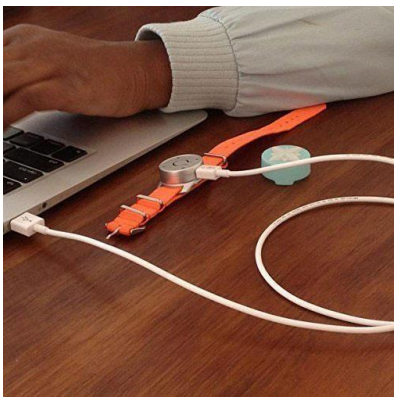
#20
Jewelbot

Einleitung

Der Jewelbot ist ein Freundschaftsarmband, das man programmieren kann, um geheime Botschaften an seine Freunde, die auch so ein Armband besitzen, zu schicken und um andere coole Dinge damit zu machen. Jewelbots ist speziell auf Mädchen ausgerichtet, aber das Design wird auch viele Buben ansprechen. Jewelbots wurde von zwei Frauen entwickelt, die sich von Kindheit an mit Informatik und Programmierung beschäftigten. Sie möchten mit ihrer Entwicklung mehr Mädchen für die MINTFächer begeistern. Sie möchten bei Mädchen "**eine anhaltende Liebe zu Computern und zur Programmierung**" erzeugen, eine "**Liebe, die die Mädchen durch ihre Leben und Karrieren begleiten wird**".



Unter dem Charme des Jewelbots befindet sich ein kleiner Computer. Genauso wie wir ein Programm schreiben können, das auf unserem Computer läuft, können wir den Jewelbot so programmieren, dass er leuchtet und summt. Wir programmieren den Jewelbot mit einem Programm auf unserem Computer namens **Arduino IDE**. **Arduino** ist der Name des winzigen Computers in unserem Jewelbot. Es gibt viele verschiedene Arten von Arduinos (genau wie es viele verschiedene Arten von Laptops gibt), und die im Jewelbot ist speziell für den Jewelbot konzipiert. IDE steht für "**I**ntegrated **D**evelopment **E**nvironment". Jewelbots verwenden die Arduino-Programmiersprache und haben auch spezielle Funktionen, die wir verwenden, um die LEDs und den Buzzer (Summer) zu programmieren. Der Jewelbot hat viele Modi. Die zwei wichtigsten, die wir verwenden werden, sind der **Codiermodus** und der **Freundschaftsmodus**. Im Freundschaftsmodus können wir auf den **Pairing-Modus** und den **Messaging-Modus** zugreifen, mit denen wir uns mit den Jewelbots anderer Freunde verbinden und Nachrichten senden können. Im **Codiermodus** können wir Programme auf unseren Jewelbot hochladen. Sobald wir ein bestimmtes Programm auf unseren Jewelbot hochgeladen haben, ist dies alles, was der Jewelbot ausführen kann, bis wir es durch ein anderes Programm ersetzen, das wir geschrieben und hochgeladen haben.

1) Erste Schritte

a) Ein- und Ausschalten

Den Jewelbot ein- und auszuschalten ist super einfach!

Einschalten: Wir dru cken einfach den „magischen Knopf“! Wir sehen eine „Regenbogenanimation“, wenn der Jewelbot eingeschaltet wird!

Ausschalten: Wir halten den „magischen Knopf“ für 5 Sekunden gedrückt. Wir sehen wieder eine Regenbogenanimation, wenn der Jewelbot sich ausschaltet. b) Aufladen

Bevor wir anfangen ko nnen, mu ssen wir unseren Jewelbot aufladen.

.) Wir heben den Charm an, um an die Micro-USB-Ö ffnung zu kommen.

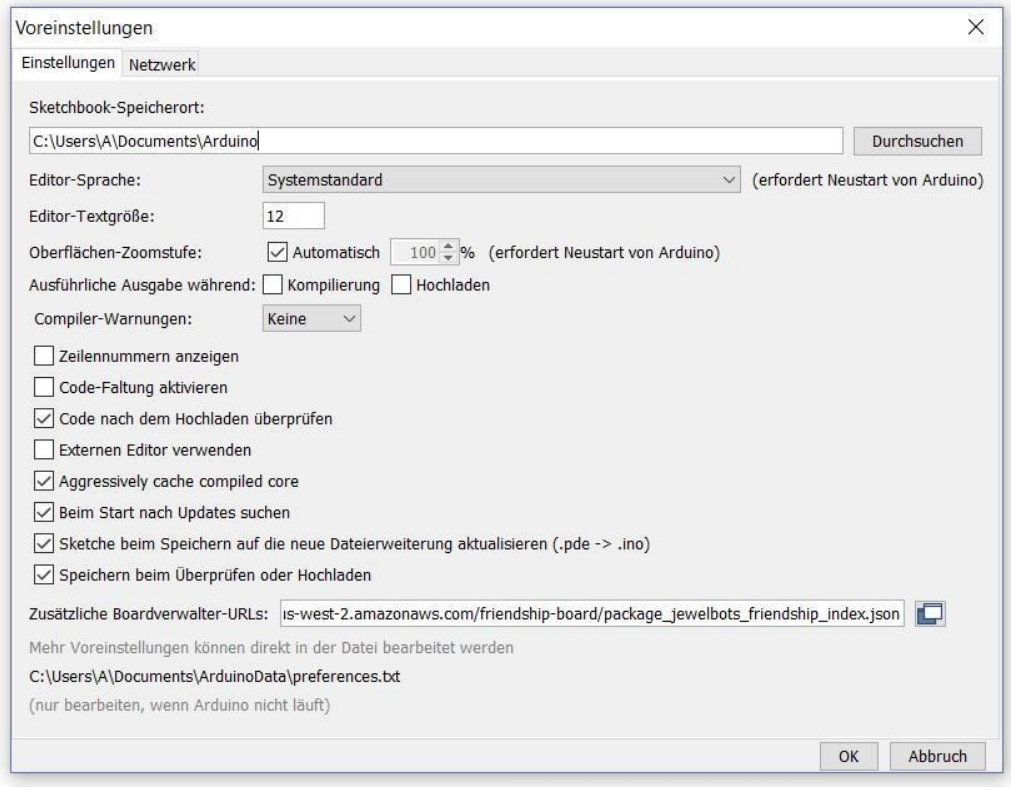
.) Wir schließen den Jewelbot u ber ein USB-Kabel an eine Stromquelle an.

.) Wir dru cken den „Magic Button“, um den Jewelbot zu aktivieren. Ein rotes Licht erscheint, wenn er eingeschaltet ist und aufgeladen wird. Ein gru nes Licht bedeutet eine volle Batterie. Volle Aufladung: 2 Stunden Ladezeit, minimalste Aufladung: 5 Minuten Ladezeit.

2) Setup

a) Wir laden den Jewelbot auf!
b) Um mit der Programmierung des Jewelbot zu beginnen, mu ssen wir die Arduino IDE herunterladen.
c) Die Arduino IDE (Integrated Development Environment / integrierte Entwicklungsumgebung) ist eine Programmierumgebung, mit der wir viele verschiedene Dinge programmieren ko nnen. Wir mu ssen sie installieren, um mit dem Jewelbot arbeiten zu ko nnen
d) Wir installieren die Arduino Software (IDE). Wenn wir mit diesem Schritt fertig sind, fügen wir die Jewelbot-Boards hinzu.
e) Um den Jewelbot programmieren zu können, müssen wir die speziellen Jewelbot-Boards zur Arduino IDE hinzufügen.
f) Wir öffnen **Datei – Voreinstellungen**
Wir kopieren in das Feld **„Zusätzliche Boardverwalter-URLs“** folgende URL:

https://s3-us-west-2.amazonaws.com/solo-board/package_jewelbots_index.json,https://s3-us-west-2.amazonaws.com/friendship-board/package_jewelbots_friendship_index.json



g)

Wir klicken auf „OK“!

h) Wir schließen die Arduino IDE und starten sie neu!

i)Wir scrollen im Menü **Werkzeuge** über "**Board**:" und klicken dann auf "**Boardverwalter**".

j) Wir geben "j" in das Suchfeld ein, um schnell die Jewelbot-Boards zu finden.

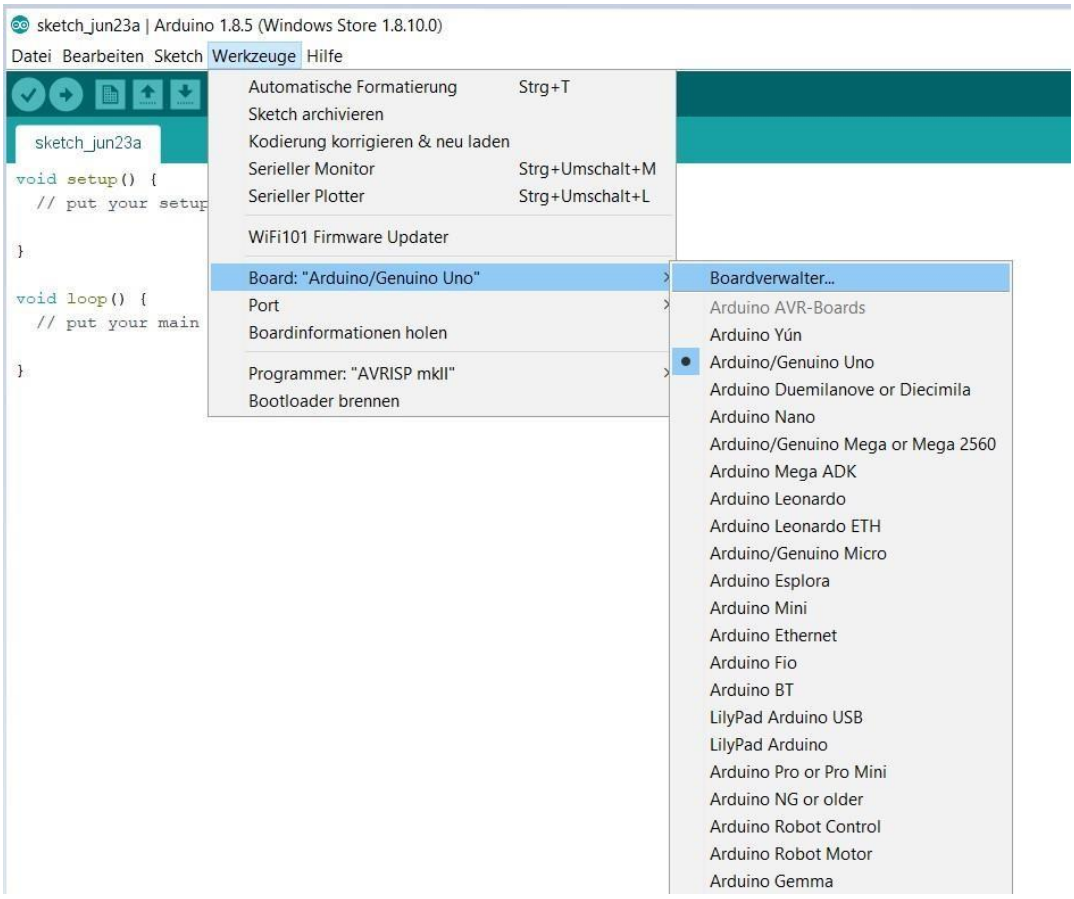
k) Wir klicken auf eine beliebige Stelle in der Box fu r "Jewelbots Firmware Update", "Jewelbots Friendship Mode" und "Jewelbots Solo Mode", um die Schaltfla che "Installieren" anzeigen zu lassen.

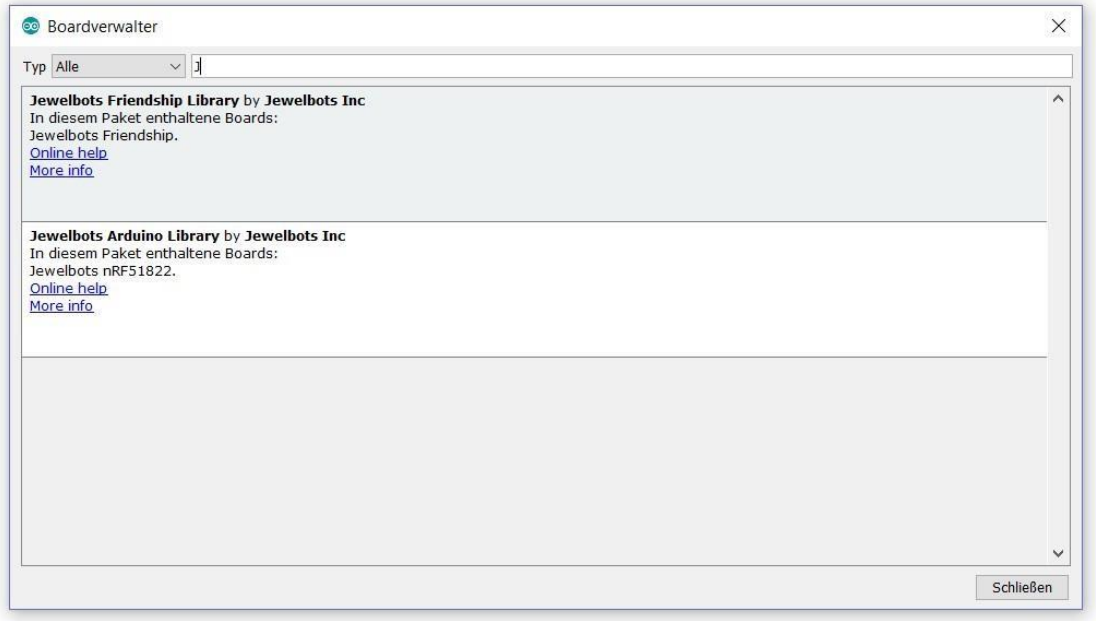
l) Wir stellen sicher, dass wir alle drei installieren, damit wir unseren Jewelbot nach der Aktualisierung der Firmware programmieren ko nnen Wenn die Box „Jewelbot Firmware Update“ nicht erscheint, ist kein Update notwendig!

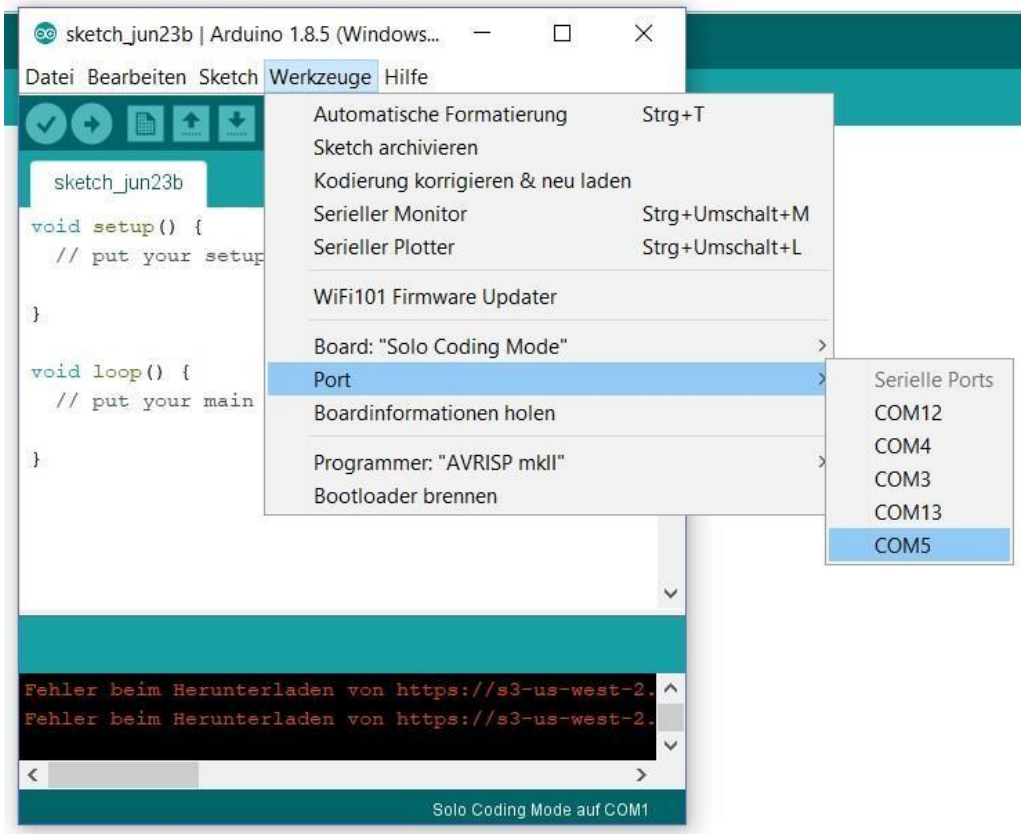
m) Wir klicken der Reihe nach auf die Schaltfla chen "Installieren", um die Bibliotheken herunterzuladen.

n) Wir schließen das Programm und o ffnen es erneut.

o) Der Setup-Prozess ist abgeschlossen, wir ko nnen mit dem Programmieren beginnen.







3) Einen neuen Sketch beginnen

Ein Programm in Arduino heißt **Sketch** Ein Sketch ist der Code der Arduino IDE, den wir dann auf unseren Jewelbot hochladen werden! Es ähnelt dem Erstellen eines neuen Dokuments. a) Wir drücken die Schaltfläche "**Neu**" für einen leeren Arduino-Sketch.

b) Wir gehen zu **„Werkzeuge“**, und suchen im Menüpunkt **"Board"** und wählen das **JewelBots Boards**, das **Solo Coding Board** genannt wird. Es wird am unteren Rand des Menüs von Boards angezeigt.

c) Wir gehen erneut zum Menü „Werkzeuge“ und wählen den richtigen **Port** für den Jewelbot. Am einfachsten ist es, wenn der Jewelbot das einzige USB-Gerät ist, das an den Computer angeschlossen ist.

4) Programme hochladen Ein

fertiges Programm auf den Jewelbot hochladen ist sehr einfach.

a) Wir drücken und halten die „Magic-Taste“ für 2 Sekunden, während der Jewelbot natürlich per USB am Computer angeschlossen ist. Der Jewelbot kommt in den **Upload-Modus**.

b) Wir beachten, dass die 4 Lichter violett werden und sehen, dass die Ladelampe (rot oder grün) erlischt .

c) Wir drücken die **Hochladen-Schaltfläche** auf Arduino und warten, bis wir eine Nachricht erhalten, die uns mitteilt, dass der Upload abgeschlossen ist.

d) Nun warten wir, bis das rote Licht wieder erscheint und trennen dann den Jewelbot vom Computer, um unser Programm in Aktion zu sehen!

5) Ein einfaches Beispiel

```
void setup() {

 // put your setup code here, to run once:

}

void loop() {

 // put your main code here, to run repeatedly:

LED led;   led.turn_on_single(SW, GREEN);

}
```

a) Jeder neue Sketch hat zumindest zwei **Funktionen**: Setup und Loop. Das Wort void wird für Funktionen ohne Rückgabewert (also Prozeduren) verwendet. Deshalb auch die leere Klammer.
Nach einer offenen geschweiften Klammer steht ein hilfreicher Kommentar, der mit zwei Schrägstrichen beginnt. Danach stehen einige Leerzeilen (viele Programmierer nennen das "Whitespace") und danach eine schließende geschweifte Klammer.
Jedes Mal, wenn wir nach zwei Schrägstrichen etwas sehen, bedeutet dies, dass es sich um eine Notiz für die Person handelt, die den Code liest, und nicht um den Code, den der Computer ausführt.
Der Jewelbot versteht die Setup- und Loop-Funktionen und weiß, dass er nach dem Hochladen beim Ausstecken zuerst die Setup-Funktion ausführen soll und dann die Loop-Funktion **immer wieder** ausführen soll. Jeden Code, den wir zwischen den öffnenden und schließenden geschweiften Klammern einer Funktion schreiben, sagt dem Jewelbot, was zu tun ist, wenn diese Funktion ausgeführt wird. Vorerst interessiert uns nur die Loop-Funktion. b) Wir kopieren den Code in unsere Arduino IDE.

c) Wir bringen den Jewelbot in den **Upload-Modus**, indem wir die „**Magic-Taste**“ für 2 Sekunden drücken. Der Jewelbot ist natürlich per USB am Computer angeschlossen. Die vier LEDs leuchten violett auf.

d) Wir drücken die **Hochladen-Schaltfläche** in der Arduino IDE und warten, bis wir eine Nachricht erhalten, die uns mitteilt, dass der Upload abgeschlossen ist. **(„Hochladen abgeschlossen“)**

e) Wir trennen den Jewelbot vom USB Anschluss! Der Jewelbot zeigt uns das Ergebnis unserer Programmierung: **Die LED (im SüdWesten) leuchtet grün**.

f) Die vier LEDs werden nach Himmelsrichtungen benannt! Diese richten sich nach der Lage des USBPorts. Der muss immer nach unten zeigen!

6) Klassen und Objekte

In vielen Programmiersprachen gibt es Dinge, die "Objekte" genannt werden. Objekte werden unter Verwendung von Funktionen einer sogenannten "Klasse" angesprochen.

Wir werden keine neuen Klassen schreiben, wenn wir den Jewelbot programmieren, aber wir werden **Funktionen** aus Klassen verwenden, die die Schöpfer des Jewelbots geschrieben haben. Solche Klassen sind die **LED-Klasse**, die **Animation-Klasse**, die **Buzzer-Klasse** oder auch die **Timer-Klasse**. Diese Klassen enthalten **verschiedene Funktionen** zu den LEDs auf dem Jewelbot. Insbesondere schalten alle Funktionen in der LED-Klasse eine oder mehrere LEDs an einem Jewelbot ein und aus. Damit wir diese Funktionen nutzen können, müssen wir ein LEDObjekt erstellen. LED lampe;

Wir haben also für die Klasse LED ein Objekt mit dem Namen lampe (Achtung: immer in Kleinbuchstaben) erstellt. Aus diesem Beispiel folgt, dass es egal ist, wie der Name eines Objekts ist, solange es einen Namen hat, mit dem es identifiziert werden kann.

lampe.turn_on_single(SW, GREEN);

Nun ordnen wir dem Objekt mit dem Namen „lampe“ eine Funktion aus der LED-Klasse zu. Wir schreiben den Namen des Objekts und dann einen Punkt und dann **turn_on_single**, wenn wir diese Funktion verwenden wollen und eine LED aufleuchten lassen wollen.

Aber das ist nicht alles! Jetzt weiß Jewelbot, dass er eine LED einschalten muss. Aber welche LED soll er einschalten und welche Farbe soll es sein? Wir schreiben dieses als **Argumente für die Funktion** am Ende von turn_on_single in eine Klammer. (LED im **S**üd**W**esten leuchtet grün!)

7) Arbeiten mit den LEDs

a) Es gibt insgesamt 4 LEDs, die mit dem Jewelbot verbunden sind.

b) Um die LED-Funktionen verwenden zu können, müssen wir zuerst die LED-Klasse wählen und danach einem Objekt einen Namen geben.

LED led;

Dies teilt dem Programm mit, dass wir die LEDs des Jewelbot codieren möchten.

c) Dann können wir die folgenden Funktionen verwenden, um mit LEDs interessante Dinge zu tun!

d) Die verschiedenen Funktionen werden als "API" bezeichnet, da sie die Art und Weise darstellen, wie wir mit dem Gerät interagieren. APIs sind Möglichkeiten, mit verschiedenen Schnittstellen zu arbeiten, manchmal auf einem Computer und manchmal online.

e) Um eine bestimmte LED zu aktivieren, müssen wir dem Programm mitteilen, mit welcher LED wir arbeiten möchten. Dazu verwenden wir die Werte NE, NW, SE, SW. Diese stehen für die Richtungen Nordosten, Nordwesten, Südosten, Südwesten. (siehe Abbildung oben!)

f) Es gibt auch viele verschiedene Farben, mit denen wir arbeiten können! Die Liste der Farbnamen, die wir verwenden können, sind:
RED, GREEN, BLUE, MAGENTA, YELLOW, CYAN, WHITE, ORANGE, GOLD, PURPLE, PERIWINKLE, ROSE, OCEAN, and SKY.

g) Funktionen:

.) Diese Funktion schaltet eine einzelne LED des Jewelbot ein:

```
LED led;
led.turn_on_single (SW, GREEN);
```

.) Diese Funktion schaltet alle LEDs des Jewelbot ein:

```
LED led;
led.turn_on_all (GREEN);
```

.) Diese Funktion schaltet eine einzelne LED des Jewelbot aus:

```
LED led;
led.turn_off_single (NE);
```

.) Diese Funktion schaltet alle LEDs des Jewelbot aus:

```
LED led;
led.turn_off_all ();
```

.) Diese Funktion schaltet die LED für eine bestimmte Zeit ein und dann erlischt die LED:

```
LED led;
led.flash_single (SE, BLUE, 1000);
```

.) Diese Funktion schaltet alle LEDs für eine bestimmte Zeit ein und schaltet sie dann aus:

```
LED led;
led.flash  all (BLUE, 1000);
```

.) Diese Funktion lässt eine Regenbogenfarben-Animation blinken.

```
Animation animation;
animation.rainbows ();
```

.) Mit dieser Funktion zeigt der Jewelbot die Farben des Jewelbot-Logos auf allen 4 LEDs an.

```
Animation animation;
animation.jewelbots  logo ();
```

.) Diese Funktion schaltet alle LEDs ein und schaltet sie langsam wieder aus, so dass sie wie ein Muster aussehen.

```
Animation animation;
animation.breathe_single_color (GREEN);
```

.) Diese Funktion ist eine Animation, bei der ein Magenta-Ball, zufällig herumspringt!

```
Animation animation;
animation.bouncing_ball ();
```

8) Arbeiten mit dem Buzzer

Mit dem Buzzer (Summer) des Jewelbot können wir Vibrationen in bestimmter Länge erzeugen. Die Zeit, die der Motor vibriert, wird in **Millisekunden** gemessen. 40 **ms** ist das Minimum, und 2 Sekunden (2000 ms) ist das Maximum. .) Mit dieser Funktion vibriert Jewelbot für eine sehr kurze Zeit (125 ms).

```
Buzzer buzz;
buzz.extra_short_buzz();
```

.) Mit dieser Funktion vibriert Jewelbot für eine kurze Zeit (250 ms).

```
Buzzer buzz;
buzz.short_buzz();
```

.) Mit dieser Funktion vibriert der Jewelbot für eine mittellange Zeit (500 ms).

```
Buzzer buzz;
buzz.medium_buzz();
```

.) Mit dieser Funktion vibriert der Jewelbot für eine lange Zeit (750 ms).

```
Buzzer buzz;
buzz.long_buzz();
```

.) Mit dieser Funktion vibriert der Jewelbot für eine extra lange Zeit (1000 ms).

```
Buzzer buzz;
buzz.extra_long_buzz();
```

.) Mit dieser Funktion vibriert der Jewelbot für eine wirklich lange Zeit (1500 ms).

```
Buzzer buzz;
buzz.really_long_buzz();
```

.) Diese Funktion erstellt einen benutzerdefinierten Buzz! Die Amplitude ist ein Maß dafür, wie stark der Motor vibriert. Die Amplitude kann zwischen 0 (kein Summen) und 127 (maximales Summen) liegen. Die zweite Zahl gibt die Länge des Summens in ms an!

```
Buzzer buzz;
Buzz.custom buzz (63, 500);
```

Die Stärke des Motors ist also auf 50% eingestellt (maximales Summen ist 127), die Dauer des Summens ist ½ Sekunde (500ms)!

9) Arbeiten mit der Magic-Taste

Wir können **Ereignisse** für den Magic-Button programmieren. Je nachdem wie lange der Button gedrückt wird können verschiedene Ereignisse für die LEDs oder den Buzzer programmiert werden.

Die folgenden Funktionen werden als Ereignisse bezeichnet, da diese durch eine Interaktion auf dem Gerät ausgelöst werden. Ereignisse sind Wege, wie wir Dinge auslösen können, nachdem wir etwas anderes getan haben, in diesem Fall nach dem Drücken der Magic-Taste.

.) Dieses Ereignis passiert, wenn wir einmal auf die Magic-Taste drücken.

```
 void button_press(void) {
LED led;
 led.turn_on_single(NW, GREEN);
 }
```

.) Dieses Ereignis passiert, wenn wir die Magic-Taste drücken und für einen Moment gedrückt halten.

```
void button_press_long(void) {   LED led;
 led.turn_on_single(SE, RED);
 }
```

.) Dieses Ereignis passiert, wenn wir die Magic-Taste drücken, wenn das Gerät am Computer angeschlossen ist.

```
   void charging_button_press(void) {
LED led;     led.turn_on_single(NW,
BLUE);
   }
```

10) Kleine Beispiele

a) Beispiel 1: LEDs

```
LED led; Timer

timer; void

setup() {} void

loop() {

  // Arbeiten mit LEDs

led.turn_on_all(BLUE);

timer.pause(500);
```

```
led.turn_on_single(SW, GREEN);

timer.pause(500); led.flash_single(NE,

RED, 1000);
}
```

Zuerst leuchten alle LEDs blau, eine LED (Südwesten) leuchtet nach 500 ms grün, eine zweite LED (Nordosten) leuchtet rot. Danach beginnt das Programm wieder von vorne. Wir benötigen die Bibliotheken LED und Timer!

b) Beispiel 2: Animationen

```
LED led;

Animation animation;

Timer timer; void

setup() {} void loop()

{

  // Arbeiten mit Animationen

animation.rainbows(); timer.pause(500);

animation.jewelbots_logo();

timer.pause(500);

animation.breathe_single_color(MAGENTA);

}
```

Ergebnis: Es erscheint ein Regenbogenmuster (LEDs zeigen verschiedene Farben), dann das JewelbotsLogo (alle LEDs leuchten blau, jede wechselt hintereinander im Uhrzeigersinn auf rot) und am Schluss leuchten alle vier LEDs in der Farbe Magenta und verlöschen wieder (breathe). **c) Beispiel 3: Magic-Button 1**

```
LED led;

Animation animation;

Timer timer; void setup()
{} void loop() {} void button_press

(void){  animation.rainbows();

timer.pause (500);

led.turn_on_all(GREEN);

}
```

Drücken wir den Magic-Button leuchten nach der Regenbogen-Animation alle LEDs grün! **d) Beispiel 4: Magic-Button 2**

```
LED led;

Animation animation;

Timer timer; void setup() {}

void loop() {} void

button_press (void) {

animation.rainbows();

timer.pause (500);

led.turn_on_all(GREEN);

}

void button_press_long (void) {

led.flash_all(RED, 1000);

timer.pause(500);

animation.jewelbots_logo();
```

```
}
```

Drücken wir den Magic-Button leuchten also nach der Regenbogen-Animation alle LEDs grün. Drücken wir den Magic-Button länger, dann leuchten alle LEDs rot, danach zeigen die LEDs das Jewelbots-Logo.

11) Variable

Bei der Programmierung versuchen wir immer, die Dinge so effizient wie möglich mit der geringsten Wiederholung von Codezeilen zu machen. Nehmen wir an, wir haben folgendes Programm für den Jewelbot geschrieben:

```
void setup () {} void
loop () {}
void button_press () {

LED lampe;
Timer timer;

lampe.turn_on_single (NE, BLUE);
timer.pause (2000);
lampe.turn_off_single (NE);
timer.pause (3000);
lampe.turn_on_single (NE, BLUE);
timer.pause (2000);
lampe.turn_off_single (NE);
timer.pause (3000);
lampe.turn_on_single (NE, BLUE);
timer.pause (2000);
lampe.turn_off_single (NE);
timer.pause (3000);
}
```

Dadurch wird eine LED nach Drücken der Magic-Taste zwei Sekunden lang blau leuchten und dann drei Sekunden lang ausgeschaltet. Das Ganze wird dann noch zweimal wiederholt.

Was, wenn wir den Sketch ein bisschen ändern möchten und die LED für eine Sekunde anstelle von zwei einschalten und dann für vier Sekunden statt drei ausschalten möchten?

So wie es jetzt ist, müssten wir jedes Mal, wenn wir 2000 geschrieben haben, die Zahl auf 1000 ändern und dann jedes Mal, wenn wir 3000 geschrieben haben, diese Zahl auf 4000 ändern. Wir könnten das tun, aber wenn das Programm lang wäre, würde es eine Weile dauern, jeden Befehl einzeln zu ändern. Wir könnten auch vergessen, eine der Zahlen, die wir ändern wollten, zu aktualisieren, und das wäre nicht gut!

Zur Vereinfachung können wir die Zahlen 2000 und 3000 als Variable speichern. Variable haben einen **Namen** und einen **Wert**. Sobald wir eine Variable erstellt und auf einen Wert festgelegt haben, erkennt das Programm jedesmal den Variablennamen und den Wert, den wir festgelegt haben.

So definieren wir eine Variable.

```
int EINGESCHALTET = 2000;
```

Der Name, den wir für diese Variable gewählt haben, ist EINGESCHALTET, da es die Zeit ist, die die LED an bleibt. Es ist gut, Namen zu finden, die im Zusammenhang mit unserem Programm Sinn ergeben.

Nach EINGESCHALTET haben wir ein Gleichheitszeichen und dann 2000 geschrieben. Das heißt, dass der Wert dieser Variablen 2000 ist, die Zeitmenge, für die die LED eingeschaltet bleiben soll. Nach 2000 gibt es einen Strichpunkt, denn so enden alle Anweisungen.

Was bedeutet das Wort **int** am Anfang? Das nennt man den **Typ** der Variablen, der Jewelbot muss wissen, ob der Wert der Variablen eine ganze Zahl (wie 1, 3 oder 12) ist, oder eine Zahl mit einem Komma (wie 5,2 oder 11,6) ist oder Buchstaben (wie A oder B) sind.
In unserem Fall ist der Typ der Variablen int, also der Inhalt (Wert) ist eine ganze Zahl, eine Zahl ohne Dezimalwert.
Einen Wert mit Dezimalzahlen hat der Variablentyp, der als **float** bezeichnet wird, und einen Wert mit Buchstaben hat der Variablentyp, der **char** genannt wird. Wir werden wahrscheinlich keinen CharTyp benötigen, wenn wir unseren Jewelbot programmieren.

Wir können die Anweisung **int EINGESCHALTET = 2000;** an den Beginn unseres Programmes setzen.
Wir können auch eine andere Variable definieren, die angibt, wie lange die LED aus ist, nennen wir sie **AUSGESCHALTET**. Da wir möchten, dass die LED für 3 Sekunden aus ist, schreiben wir **int AUSGESCHALTET = 3000;**

Unser Programm sieht dann so aus:

```
int EINGESCHALTET = 2000; int
AUSGESCHALTET = 3000; void
setup () {} void loop () {}
void button_press () {

LED lampe;
```

```
Timer timer;
  lampe.turn_on_single (NE,
BLUE); timer.pause
(EINGESCHALTET);
lampe.turn_off_single (NE);
timer.pause (AUSGESCHALTET);
lampe.turn_on_single (NE,
BLUE); timer.pause
(EINGESCHALTET);
lampe.turn_off_single (NE);
timer.pause (AUSGESCHALTET);
lampe.turn_on_single (NE,
BLUE); timer.pause
(EINGESCHALTET);
lampe.turn_off_single (NE);
timer.pause (AUSGESCHALTET);
}
```

Wenn wir nun die Zeit, in der die LED eingeschaltet ist, auf eine Sekunde und die Zeit, in der die LED ausgeschaltet ist, auf vier Sekunden ändern möchten, brauchen wir einfach die Zahlen nach den Gleichheitszeichen ändern.

```
int EINGESCHALTET = 1000; int
AUSGESCHALTET = 4000;
```

Jetzt beträgt die Dauer der LED-Anzeige eine Sekunde und die Zeit, während der die LED aus ist, beträgt insgesamt vier Sekunden. Ist das nicht viel einfacher als das ganze Programm durchlaufen zu müssen und alle 2000 durch 1000 und alle 3000 durch 4000 zu ersetzen?

12) Informatische Begriffe

Boolesche Variable

Wir verstehen unter einer booleschen Variablen eine Variable, die nur zwei Zustände annehmen kann. Diese beiden Zustände werden „true“ und „false“ („wahr“ und „falsch“) genannt und werden auch als Wahrheitswerte bezeichnet. Solche Variablen finden bei bedingten Anweisungen oder Schleifen Anwendung. Ihren Namen hat die boolesche Variable nach George Boole, einem englischen Mathematiker des 19. Jahrhunderts.

Wir kreieren eineVariable z. B. mit dem Namen „Grün“ mit der Befehlsfolge bool

```
Grün; if(color == GREEN){    Grün=true; }
  else{
  Grün=false; }
```

Kompilieren
Kompilieren bedeutet, dass unser Code so übersetzt wird, dass der Computer ihn verstehen und verwenden kann. Wenn wir die "Kompilier-Schaltfläche" drücken, wird die Arduino IDE auch nach Fehlern in unserem Programm suchen.

Funktionen
Jeder neue Sketch hat zumindest zwei **Funktionen**: Setup und Loop. Das Wort void wird für Funktionen ohne Rückgabewert (also Prozeduren) verwendet. Deshalb auch die leere Klammer.

```
void setup() {} void loop()
{}
```

Der Jewelbot versteht die Setup- und Loop-Funktionen und weiß, dass er nach dem Hochladen unseres Programmes nach dem Ausstecken zuerst die Setup-Funktion ausführen soll und dann die LoopFunktion **immer wieder** ausführen soll. Jeden Code, den wir zwischen den öffnenden und schließenden geschweiften Klammern einer Funktion schreiben, sagt dem Jewelbot, was zu tun ist, wenn diese Funktion ausgeführt wird.

Andere Funktionen sind z.B.

```
void button_press(){}
```

Int
Dies ist der Name von Variablen, die Zahlen sind. Es steht für "Integer". int-Datentypen müssen immer mit Kleinbuchstaben "i" programmiert werden. Die Programmierung unterscheidet zwischen Groß- und Kleinschreibung

Initialisieren
Dies ist der Prozess, einen Anfangswert für eine Variable zu deklarieren. Wenn wir beispielsweise "int x = 2" eingeben, initialisieren wir den Wert von x auf 2.

Pin
Eine Nummer oder eine String-Adresse auf dem Arduino-Board. Es ist wie der Name eines Ortes auf unserem Board. (Die 4 LEDs haben neben den Namen SW, NW, NE und SE auch die Pin-Nummern 0, 1, 2 und 3)

String
Dies ist der Name von Variablen, die Wörter oder Buchstabenwerte sind. "String" -Datentypen müssen immer mit einem Großbuchstaben "S" programmiert werden. Die Programmierung unterscheidet zwischen Groß- und Kleinschreibung.

Hochladen
Wenn wir Code hochladen, laden wir den Code von unserem Computer auf unser zu codierendes Gerät, z.B. unseren Jewelbot, hoch.

Variable
Variablen enthalten einen Datenwert, der vom Programm gespeichert wird. Diese Werte können Strings (char), Ganzzahlen (int) oder Dezimalzahlen (float) sein.

13) Informatische Konzepte

Die Programmierung unterscheidet zwischen Groß- und Kleinschreibung. Zum Beispiel muss der Variablentyp int in Kleinbuchstaben sein, sonst erhalten wir eine Fehlermeldung. Außerdem müssen wir jede Codezeile mit einem Strichpunkt (Semikolon) beenden.

Die Funktion Setup ()
Die Funktion setup () wird beim Start eines Sketches aufgerufen. Wir verwenden diese Funktion z. B. zum Initialisieren von Variablen oder zum Erstellen von Objekten in einer Klasse. Die Setup-Funktion wird nur einmal ausgeführt, nach jedem Einschalten oder Zurücksetzen der Arduino-Karte. Beispiel:

```
void setup () {     int x =

2;

   LED led;

   // Wir haben für die Klasse LED ein Objekt mit dem Namen led (Achtung: immer in Kleinbuchstaben) erstellt.  Der Variablen „x“ weisen wir den Wert 2 zu

}
```

Kommentare
Programmierer stellen häufig Kommentare neben ihren Code, um anderen Benutzern zu helfen, zu verstehen, was ihr Code tut. Alle Kommentare beginnen mit "//" und werden vom Compiler ignoriert, was bedeutet, dass sie unseren Code nicht beeinflussen.

Variablennamen
Variablen können beliebig benannt werden. Zum Beispiel wurde in allen Beispielen in JewelbotTutorials das LED-Objekt immer als „led“ deklariert. Wir können es jedoch vollständig als was auch immer wir wollen deklarieren!

Bedingte Anweisungen
Dies sind logische arithmetische Anweisungen, die wir in unserem Programm verwenden können, um verschiedene Funktionen oder Schleifen zu steuern.
Zum Beispiel können wir diese bedingten Anweisungen verwenden, um den Jewelbot in einen Taschenrechner zu verwandeln! Nehmen wir an, wir haben Variablen namens x und y.

```
void setup () {
  // deklariere hier den Namen deines LED-Objekts.   LED led;
```

```
 // deklariere die Werte deiner Variablen x und y   int x
= (4 * 5 + 3) ^ 2;
 int y = 20x - (8 ^ 2);
}
```

Jetzt können wir die bedingten Anweisungen verwenden, um zu testen, welche Variable größer ist.

If… then… (wenn… dann...)
Dies ist eine bedingte Anweisung, die nur dann eine Aktivität ausführt, wenn etwas Bestimmtes passiert.

```
if (x> y) {
 led.turn_on_all (GREEN);
} if (y <x)
{
 led.turn_on_all (BLUE);
}
```

Dieser Code-Block besagt, dass, wenn die Variable x größer als y ist, unser Jewelbot alle vier LEDs **grün** aufleuchten lässt. Die zweite Bedingung besagt, dass, wenn die Variable y größer als x ist, unser Jewelbot alle vier LEDs **blau** aufleuchten lässt.

If… else… (wenn… ansonsten...)
Dies ist eine bedingte Anweisung, die eine Aktivität ausführt, wenn etwas Spezifisches passiert, aber in einer Situation, in der diese bestimmte Sache nicht passiert, wird sie etwas anderes tun.t

```
if (x <y) {
 led.turn_on_all (RED);
}
else {
 led.turn_on_all (WITHE);
}
```

Dieser Code-Block besagt, wenn die Variable x kleiner als y ist, unser Jewelbot alle vier LEDs **rot** aufleuchten lässt. Andernfalls schaltet unser Jewelbot alle vier LEDs auf weiß.

Schleifen
Eine Schleife ist eine Folge von Anweisungen, die sich ständig wiederholt.
Die Funktion **loop ()** führt eine Schleife aus, d.h. was immer wir programmiert haben und in der Schleife steht, wird immer wieder ausgeführt!

```
void setup () {
 // schreibe deinen Setup-Code hierher:
 LED führte;
}
```

```
void loop () {
  // schreibe deinen Hauptcode hierher:
  led.flash_single (SE, BLUE);
}
```

Es wird eine Schleife aufgerufen, damit eine LED mit der Farbe Blau leuchtet.

For-Schleife

Diese unterscheidet sich von den Bedingungen if-then oder if-else, da durch sie eine Bedingung festgelegt wird, die dazu führt, dass das Programm eine bestimmte Anzahl von Wiederholungen ausführt.

Die for-Schleife besteht 3 Parametern. Einer **Deklaration eines int**, einer **Bedingung zum Ausführen** der Schleife und eines **Inkrements** (oder Dekrements).

Ein **Inkrement** ist ein festgelegter Betrag (z.B. 1) mit der die Größe (der Inhalt) einer Variablen schrittweise **vergrößert** wird. Dem entsprechend wird mit einem **Dekrement** die Größe (der Inhalt) einer Variablen schrittweise **verkleinert**.

Zum Beispiel könnte eine for-Schleife wie folgt beginnen (int z = 0, z <10, z ++). Dies bedeutet, dass es eine Variable z mit dem Wert 0 gibt, die um 1 erhöht wird, solange sie kleiner als 10 ist.

```
for (int a = 2, a <= 20, a ++) {
led.flash_single (NW, GELB, 1000);   led.flash_single
(NE, MAGENTA, 1000);
```

14) Ein Spiel

Dieses Spiel wird "Fang den Kobold" genannt. Den Kobold stellen die 4 LEDs dar, wenn sie grün leuchten. Zu Beginn leuchten die 4 LEDs in verschiedenen Farben (Blau, Magenta, Gelb etc.) Wenn dann die 4 LEDs grün aufleuchten, muss man den Magic-Button drücken. Es handelt sich also um ein Reaktionsspiel. Hat man rechtzeitig gedrückt, bekommt man zur Belohnung eine Animation, in unserem Fall das Jewelbot-Logo. Hat man bei einer anderen Farbe gedrückt, leuchten alles vier LEDs zweimal rot auf.

```
void setup() {}
bool Gruen;

void Farbe (ColorLabel color) {

 if(color == GREEN){
   Gruen=true;
 }
   else{
   Gruen=false;
 }
 LED led;
```

```
 led.turn_on_all(color);   Timer
timer;    timer.pause(500);
 }

void loop() {

 Farbe (MAGENTA);
 Farbe (GREEN);
 Farbe (CYAN);
 Farbe (BLUE);
}
```

Wir kreieren mit dem Befehl „bool Gruen“ eine Variable, die entweder „wahr“ oder „falsch“ bedeuten kann.

Wir erstellen eine Funktion mit dem Namen „Farbe“, sie akzeptiert eine beliebige Farbe, d.h. wir können sie in unserer loop-Funktion als beliebige Farbe aufrufen.

Mit if…then… wird festgelegt, wann ist der boolesche Wert „Gruen“ wahr und wann ist er falsch.

```
void button_press(){

 Timer timer;
 LED led;
 Animation animation;

 if(Gruen){
   animation.jewelbots_logo();
timer.pause(2000);
 }
 else{
   timer.pause(1000);
led.turn_on_all(RED);     timer.pause(500);
led.turn_off_all();
   timer.pause(500);

   led.turn_on_all(RED);
timer.pause(500);     led.turn_off_all();
   timer.pause(500);

} }
```

Drücken wir die Magic-Taste wird festgestellt, ob wir bei Grün oder bei einer anderen Farbe gedrückt haben.

Wenn wir einen booleschen Wert nur für sich selbst in einem "if" -Zustand haben, wird angenommen, dass wir danach suchen, ob er wahr ist. Wenn er wahr ist (wir haben bei Grün gedrückt), dann sehen eine Jewelbot-Logo-Animation.

Wenn er nicht wahr ist (wir haben bei einer anderen Farbe gedrückt), leuchten alle vier LEDs zweimal rot auf.

Aufgaben:

.) Der Jewelbot soll mehr als vier Farben zeigen!

.) Wie können wir das Spiel einfacher oder auch schwerer machen? Wir müssen etwas in der FarbeFunktion ändern!