4. Praxisaufgaben mit OpenRoberta

Ab jetzt geht es erst so richtig los. In den folgenden Lektionen werden alle Module des Cubes programmiert.

4.1 Lektion 1: Leuchtdiode

Leuchtdioden kennst du aus deinem Alltag. Sie geben dabei Informationen über den Status von elektronischen Geräten an. Ein typisches Beispiel für eine Leuchtdiode in einem elektronischen Gerät ist die Stand-by-Leuchte von einem Bildschirm. Wenn wir die Stand-by-Leuchte sehen können, die sich häufig als roter Punkt darstellt, wissen wir, dass das Gerät z.B. über eine Fernbedienung wieder eingeschaltet werden kann.

Aufgabe 1: Die blinkende LED

Gemeinsam möchten wir uns gleich zu Beginn unserer Reise eines der wichtigsten Programme der Mikroelektronik ansehen: die blinkende Leuchtdiode.

Aufgabenstellung

Schalte die grüne Leuchtdiode auf deinem Funduino Cube ein. Pausiere eine Sekunde und schalte die grüne Leuchtdiode wieder aus. Nach einer weiteren Sekunde soll die grüne Leuchtdiode wieder aufleuchten.

Konfiguration in OpenRoberta

Schritt 1

Wenn du die OpenRoberta Oberfläche geöffnet hast, findest du in der oberen Navigationsleiste zwei Felder wieder: PROGRAMM und ROBOTERKONFIGURATION.

Abbildung Schritt 1

Wir starten mit einem Klick auf die ROBOTERKONFIGURATION. Es öffnet sich an der linken Bildschirmseite ein neues Menü. In diesem Menü findest du die beiden Felder Aktion und Sensoren.
Wir benötigen in unserem ersten Versuchsaufbau einen Baustein aus dem Feld Aktion. Diesen Baustein finden wir, indem wir jetzt das Feld Aktion anklicken.

Schritt 2

Durch den Klick auf das Feld Aktion klappt jetzt ein weiteres Menü auf. Wenn wir in diesem Menü jetzt ein wenig mit dem Mausrad nach unten scrollen, entdecken wir den gewünschten Block.
Wir haben dir diesen Block in der Abbildung rot markiert.

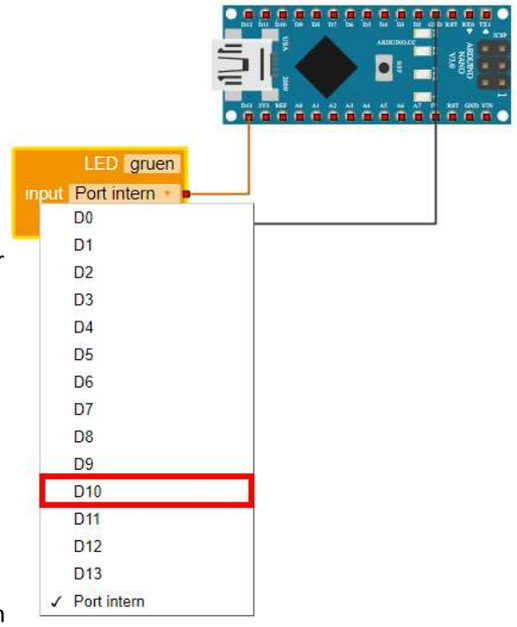
Was genau sagt dieser Block eigentlich aus?

Du wirst erkennen, dass die Worte auf der linken Seite des Blocks farblich anders hinterlegt sind, als die auf der rechten Seite des Blockes . Das liegt daran, dass wir die Informationen auf der rechten Seite des Bausteins anpassen können.

Und wie fügen wir diesen Block jetzt in unsere ROBOTERKONFIGURATION ein?

Ganz einfach! Wir klicken auf den Baustein und ziehen diesen in das weiße Feld.

Schritt 3



Der Block wird direkt nach dem Loslassen des Blockes automatisch durch eine Leitung mit unserem Controller verbunden.

Jetzt haben wir die Möglichkeit, die hell hinterlegten Informationen in dem Baustein anzupassen.

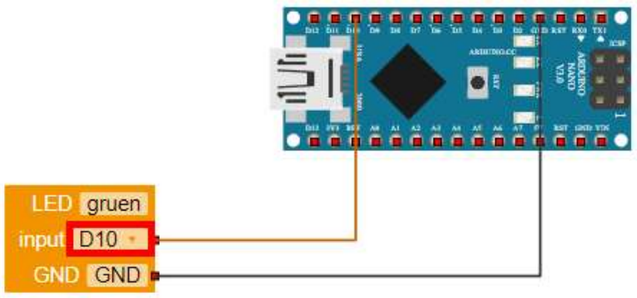
Wir beginnen damit, unserer LED einen Namen zu geben. Da wir zunächst die grüne LED auf unserer Platine blinken lassen möchten, vergeben wir den Namen „gruen“. Diese Bezeichnung wird uns im späteren Ablauf dabei helfen, die LED zielgerichtet anzusteuern.

Weiterhin wählen wir mit einem Klick auf das hell hinterlegte Feld rechts von „input“ den Pin aus, an dem unsere LED mit dem Mikrocontroller verbunden ist.

Schaue doch einmal auf deinen Funduino Cube und überprüfe, welcher Pin wir auswählen müssen, damit wir die LED ansteuern können. An der schwarzen Linie, die auf dem Cube von der LED zum Mikrocontroller führt, wirst du die Nummer des passenden Kontakts erkennen.

Schritt 4

Richtig! Wir wählen natürlich den Pin D10 aus. Deine Roboterkonfiguration müsste jetzt so wie auf dem nachfolgenden Bild aussehen.

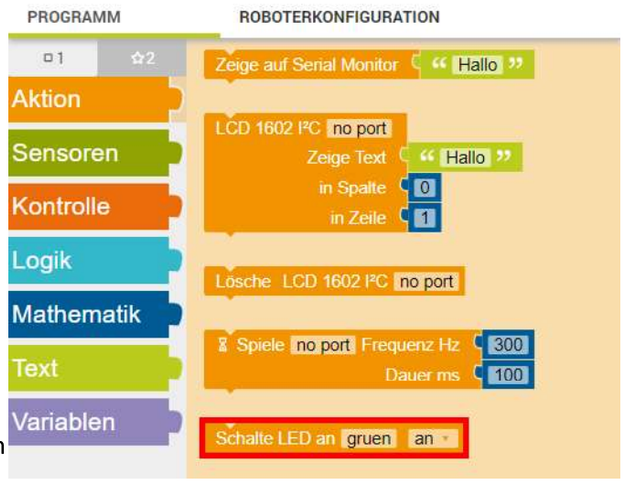


Die ROBOTERKONFIGURATION ist mit dieser letzten Einstellung erfolgreich abgeschlossen. Gratulation!

Beispielprogramm in OpenRoberta

Schritt 1

Jetzt schauen wir uns gemeinsam das zweite Feld in der oberen Navigation an, dass den Namen „PROGRAMM“ trägt. Mit einem Klick auf dieses Menü gelangst du zu den sogenannten Blöcken.



Unsere grüne LED finden wir mit einem Klick auf den Baustein „Aktion“ am linken Bildschirmrand wieder. Nach dem Klick öffnet sich, so wie du es schon aus unserer ROBOTERKONFIGURATION kennst, ein Feld mit unterschiedlichen Blöcken.

Weil wir eine LED ansteuern müssen, wählen wir den Block „Schalte LED an gruen an“ aus und ziehen diesen in das weiße Feld.

Schritt 2:

Sicherlich wirst du jetzt feststellen, dass der Aktionsbaustein „Schalte LED an gruen an“ in dem weißen Feld schwebt. Mit einem weiteren Klick auf den Aktionsbaustein können wir diesen nun an die gewünschte Stelle in unserer Schleife ziehen.

Herzlichen Glückwunsch! Du hast soeben deinen ersten Programmablauf erstellt!

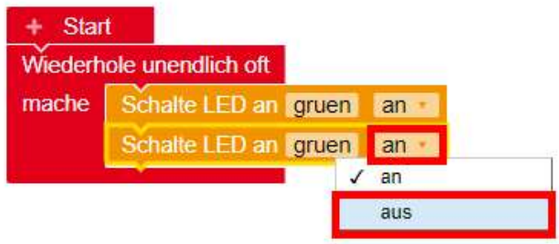
Schritt 3:

Doch bevor wir uns jetzt auf unserem Erfolg ausruhen, erinnern wir uns an unser eigentliches Ziel: Wir möchten eine LED blinken lassen.

Wenn du das Programm jetzt auf deine Platine lädst, wirst du feststellen, dass diese dauerhaft leuchtet. Wir müssen also noch einen weiteren Aktionsbaustein ergänzen, der die grüne LED wieder ausschaltet.

Na, findest du den Baustein wieder? Falls nicht, schaue dir die vorherigen Schritte noch einmal genauer an. Dein Programmlauf sollte jetzt wie auf der nachfolgenden Abbildung aussehen.

Schritt 4:



Schauen wir uns den vorherigen Programmablauf an, so stellen wir fest, dass wir dem Mikrocontroller jetzt Folgendes beigebracht haben:

Schalte die grüne LED an…
…und danach…
…schalte die grüne LED erneut an…

Unser Programmcode stimmt so also bis jetzt nicht ganz mit unserem Ziel überein. Das ist aber gar kein Problem, denn wir wählen jetzt den zweiten Block aus und klicken auf „an“. Es öffnet sich ein Menü, in welchem wir jetzt richtigerweise „aus“ auswählen können.

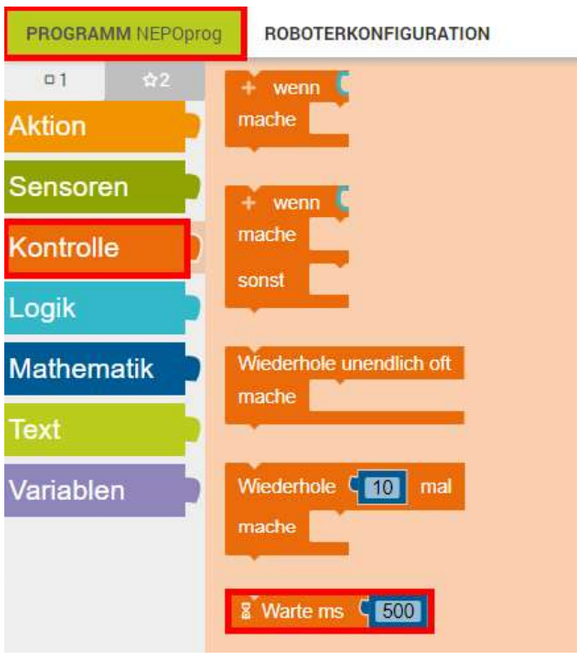
Schritt 5:

Dein Programmablauf sieht jetzt aus, wie auf der nachfolgenden Abbildung.

Toll gemacht! Wir haben eine blinkende LED programmiert und unser Ziel erreicht.

Probiere doch einmal aus und beobachte, was passiert, wenn du das Programm jetzt auf dein Funduino Cube überspielst.

Schritt 6:



Ohje, die LED leuchtet dauerhaft? Woran liegt das?

Wie du in der Einstiegslektion bereits gelernt hast, lesen Mikrocontroller einen Programmcode genauso aus, wie du ein Buch liest:
Von oben nach unten und von links nach rechts.

Im Gegensatz zu dir lesen Mikrocontroller die Programmabläufe in einer wahnsinnig schnellen Geschwindigkeit aus. Diese Lesegeschwindigkeit wird auch *Frequenz* genannt.

Auch wenn wir den Programmablauf richtig programmiert haben, erfolgt das Auslesen so schnell, dass wir mit dem menschlichen Auge nicht erkennen können, ob die LED auch wirklich blinkt. Für unser Auge scheint es so, als würde die LED durchgehend leuchten – auch wenn sie das in Wirklichkeit gar nicht tut.

Für diesen Fall gibt es einen weiteren Block, den wir dir jetzt vorstellen werden. Dieser Baustein nennt sich in OpenRoberta „Warte ms“.
Mithilfe dieses Bausteins können wir dem Mikrocontroller also befehlen, für eine bestimmte Zeitdauer zu warten (zu pausieren).

Schritt 7:

In dem Programmbaustein „Warte ms“ können wir jetzt festlegen, für welche Zeitdauer (auch *Intervall* genannt) der Mikrocontroller warten soll, bis der nächste Befehl ausgewählt wird. Der Zusatz „ms“ steht hierbei für die Zeitangabe in der Einheit Millisekunden.

1 Sekunde entspricht 1000 Millisekunden…
2 Sekunden entspricht 2000 Millisekunden…
10 Sekunden entspricht 10000 Millisekunden…

Da wir nach jedem Ein- oder Ausschalten der LED eine solche Pause benötigen, fügen wir den Programmbaustein „Warte ms“ gleich an zwei Stellen in unser Programm ein.

Was passiert, wenn du das Programm jetzt auf den Funduino Cube überspielst?

Was passiert, wenn du statt der 1000 einen Wert von 5000 in das Programm überträgst und dieses erneut auf den Funduino Cube überspielst?

Aufgabe 2: Der Wechselblinker

Wir haben bei unserer ersten Aufgabe gelernt, wie wir eine einzelne LED auf dem Funduino Cube mithilfe der Entwicklungsumgebung OpenRoberta zum Blinken bringen können. Jetzt möchten wir den Schwierigkeitsgrad ein wenig erhöhen und zwei Leuchtdioden abwechselnd blinken lassen.

Aufgabenstellung

Schalte die grüne Leuchtdiode ein. Warte anschließend eine Sekunde und schalte die grüne Leuchtdiode wieder aus.

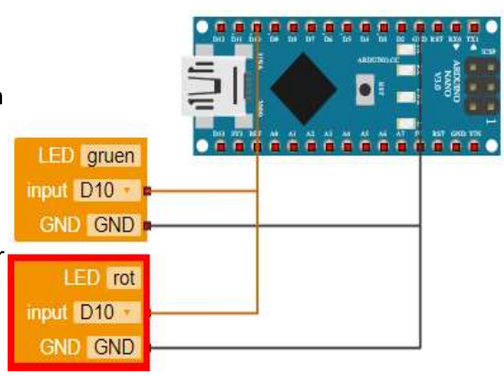
Schalte anschließend die rote Leuchtdiode ein. Warte erneut eine Sekunde und schalte die rote Leuchtdiode wieder aus.

Konfiguration in OpenRoberta

Schritt 1:

Wir haben bereits gelernt, wie wir Blöcke auswählen und diese in unsere ROBOTERKONFIGURATION einbinden können.

Wir erweitern unsere bestehende ROBOTERKONFIGURATION jetzt mit einer zweiten LED. Dazu fügen wir die entsprechenden Blöcke in unsere ROBOTERKONFIGURATION ein und vergeben für die zweite LED den Namen „rot“.



6. Praxisaufgaben mit Arduino

Ab jetzt geht es erst so richtig los. In den folgenden Lektionen werden alle Module des Cubes mit Hilfe der Arduino IDE programmiert.

6.1 Lektion 1: Leuchtdioden

Aufgabe 1: Die blinkende LED

Gemeinsam möchten wir uns gleich zu Beginn unserer Reise eines der wichtigsten Programme der Mikroelektronik ansehen: die blinkende Leuchtdiode.

Leuchtdioden kennst du aus deinem Alltag. Sie geben häufig Informationen über den Status von elektronischen Geräten an. Ein typisches Beispiel für eine Leuchtdiode in einem elektronischen Gerät ist die Stand-by-Leuchte von einem Bildschirm. Wenn wir die Stand-by-Leuchte sehen können, die sich häufig als roter Punkt darstellt, wissen wir, dass das Gerät z.B. über eine Fernbedienung wieder eingeschaltet werden kann.

Aufgabenstellung

Schalte die grüne Leuchtdiode auf deinem Funduino Cube ein. Pausiere eine Sekunde und schalte die grüne Leuchtdiode wieder aus. Nach einer weiteren Sekunde soll die grüne Leuchtdiode wieder aufleuchten.

Programmcode

Schritt 1:
Im ersten Schritt erstellen wir zunächst eine Variable für die grüne Leuchtdiode. Hierzu schreiben wir zuerst das Wort „int“ in den Programmcode.

„Int“ ist dabei die Kurzform für „integer“. Bei einem Integer handelt um einen Datentyp. In diesem Fall um eine Ganzzahl (z.B. 1, 2, 5 oder auch 1337). Es erscheint dir vielleicht ungewöhnlich, aber beim Programmieren ist die Angabe des Datentyps für das Programm sehr wichtig.

Nach der Festlegung des Datentyps folgt der Name für die Variable. Da unsere LED grün ist, nennen wir sie „modul_led_gruen“. Wenn du möchtest, kannst du anstelle von „modul_led_gruen“ auch „x“ oder „LED“ oder „Jessica“ schreiben. Wichtig ist nur, dass die Bezeichnung für dieses Modul im weiteren Programm immer exakt gleich bleibt!

Da unsere grüne LED auf dem Funduino Cube an dem Pin D10 (oder auch an Pin 10) mit dem Mikrocontroller verbunden ist, hinterlegen wir in der Variable den Wert „10“.

```
int modul_led_gruen = 10;
```

Schritt 2:
Im zweiten Schritt ergänzen wir unseren Code um das Setup.

Im Setup legen wir fest, an welchem Pin die grüne LED mit unserem Mikrocontroller verbunden ist. Dies erfolgt durch den Befehl in der nachfolgenden Abbildung.

Der erste Teil der Programmbefehls „PINBELEGUNG“ wird mit dem Namen unserer Variable „modul_led_gruen“ aus Schritt 1 ersetzt. In dieser Variable ist der Wert „10“ gespeichert.

```
pinMode(PINBELEGUNG, INPUT oder OUTPUT);
```

Schritt 3:
Anschließend müssen wir festlegen, ob es sich bei der LED um einen Eingang (Input) oder einen Ausgang (Output) handelt.

Da die LED über eine ausgehende Spannung von dem Mikrocontroller auf deinem Funduino Cube angesteuert wird, hinterlegen wir im „pinMode“ Befehl den Wert „OUTPUT“.

```
pinMode(modul_led_gruen, OUTPUT);
```

Schritt 4:
Das Codebeispiel aus Schritt 3 ergänzen wir jetzt in unserem Setup.

Wenn du alles richtig gemacht hast, sieht dein Code jetzt so aus wie in der nachfolgenden Abbildung.

```
int modul_led_gruen = 10;

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

Schritt 5:
An dieser Stelle erinnern wir uns an unsere Aufgabenstellung zurück: Wir möchten eine LED blinken lassen, also nacheinander ein- und wieder ausschalten. Bisher haben wir jedoch nur Vorbereitungen getroffen. Jetzt schreiben wir das eigentliche Programm. Der Bereich in dem das geschieht, wird als „Loop“ bezeichnet.

```
int modul_led_gruen = 10;

void setup()
{
 pinMode(modul_led_gruen, OUTPUT);
}
void loop()

{


}
```

Schritt 6:
Der Befehl, den wir für das Ein- und Ausschalten benötigen, nennt sich „digitalWrite“. Mit diesem Befehl können wir Module an einem digitalen Pin ansteuern.

Die LED auf unserem Funduino Cube befindet sich an Pin D10. Das „D“ steht dabei für „digital“, also „digital 10“.

Auch dieser Befehl setzt sich aus zwei Informationen zusammen, die durch ein Komma voneinander getrennt sind.

Die erste Information legt fest, an welchem Pin das gewünschte Modul angeschlossen ist. Wir tragen hier also wieder unsere Variable „modul_led_gruen“ ein.

```
digitalWrite(modul_led_gruen, An/Aus)
```

Schritt 7:
Die zweite Information legt fest, welches Signal wir an den digitalen Pin senden möchten.

Wenn wir ein HIGH Signal senden, wird am Pin D10 des Mikrocontrollers eine Spannung von 5V ausgegeben. Damit schalten wir die LED ein.

Wenn wir ein LOW Signal senden, wird der Pin D10 auf eine Spannung von 0V geschaltet. Damit schalten wir die LED aus.

Da wir die LED zunächst einschalten möchten, hinterlegen wir hier zunächst den Wert „HIGH“. Anschließend fügen wir die Codezeile in den Loop-Teil unseres Codes ein.

```
int modul_led_gruen = 10;

void setup()
{
 pinMode(modul_led_gruen, OUTPUT);
}
void loop()

{
digitalWrite(modul_led_gruen, HIGH);

}
```

Schritt 8:
Nachdem wir die Leuchtdiode eingeschaltet haben, möchten wir die Leuchtdiode auch wieder ausschalten. Dies erzielen wir, indem wir die zweite Information aus dem „digitalWrite“ Befehl von HIGH auf LOW umschreiben.

Auch diese Codezeile fügen wir jetzt in unseren Loop-Teil ein.

```
digitalWrite(led_gruen, LOW)
```

Schritt 9:
Wenn du die beiden Codezeilen richtig in deinen Programmcode eingefügt hast, sollte dieser jetzt so wie in der Abbildung aussehen.

Was passiert, wenn du den Code jetzt auf deinen Funduino Cube überspielst?

```
int modul_led_gruen = 10;

void setup()
{
 pinMode(modul_led_gruen, OUTPUT);
}
void loop()

{
digitalWrite(modul_led_gruen, HIGH);
digitalWrite(modul_led_gruen, LOW);
}
```

Hmm… die LED leuchtet dauerhaft. Aber warum?

Der Mikrocontroller auf deinem Funduino Cube verarbeitet den Code in einer sehr schnellen Geschwindigkeit. Die Geschwindigkeit, in welcher der Code verarbeitet wird, nennt man auch Taktfrequenz.

Die Frequenz ist dabei so hoch, dass wir Menschen das Ein- und Ausschalten der Leuchtdiode mit dem Auge nicht wahrnehmen können. Für uns macht es den Anschein, als würde die Leuchtdiode durchgehend leuchten. Was müssen wir in unserem Code ergänzen, damit das Ein- und Ausschalten für uns Menschen sichtbar wird?

Schritt 11:
Genau, wir müssen eine Pause zwischen dem Einschalten und dem Ausschalten ergänzen.

Die Pause gibt eine Zeit vor, in der das Programm stoppt. Diese Angabe erfolgt in Millisekunden (ms). Eine Sekunde entspricht dabei 1000 Millisekunden.

In einem Programmcode wird diese Pause auch „delay“ (engl. für Verzögerung) genannt. Die Codezeile sieht im Programm so aus:

```
Delay(1000);
```

Schritt 12:
Zwischen dem Einschalten und dem Ausschalten der Leuchtdiode fügen wir jetzt diese Pause ein.

Anschließend laden wir das Programm auf wieder auf unseren Funduino Cube hoch.

Und, was kannst du beobachten?

```
int modul_led_gruen = 10;

void setup()
{
 pinMode(modul_led_gruen, OUTPUT);
}
void loop()

{
digitalWrite(modul_led_gruen, HIGH);
delay(1000);
digitalWrite(modul_led_gruen, LOW);
}
```

Schritt 13:

Mist, die Leuchtdiode leuchtet immer noch dauerhaft?

Aber woran liegt das?

In Schritt 9 haben wir gelernt, dass der Mikrocontroller den Programmcode in einer sehr hohen Frequenz ausliest. Und genau liegt hier wieder einmal das Problem. Der Mikrocontroller startet sofort nach dem Verarbeiten des Loop-Teils wieder von vorne. Demnach wird direkt nach dem Ausschalten der Leuchtdiode die Leuchtdiode wieder eingeschaltet. Und das in einer Frequenz, die für das menschliche Auge wieder nicht sichtbar ist.

Schritt 14:

Wir ergänzen unseren Programmcode also um eine weitere Pause, direkt nach dem Ausschalten der Leuchtdiode. Anschließend laden wir das Programm wieder auf unseren Funduino Cube. Mit diesem letzten Schritt ist unser Code endlich fertig!

```
int modul_led_gruen = 10;
```

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

```
void loop()

{
digitalWrite(modul_led_gruen, HIGH);
delay(1000);
digitalWrite(modul_led_gruen, LOW);
delay(1000);
}
```

Gratulation, du hast dein erstes eigenes Programm geschrieben!

Nur eine Frage hätten wir da noch…

… was passiert eigentlich, wenn du die zweite Pause direkt an den Start des Loop-Teils setzt?

Aufgabe 2: Der Wechselblinker

Wir haben bei unserer ersten Aufgabe gelernt, wie wir eine einzelne LED auf dem Funduino Cube mithilfe der Entwicklungsumgebung Arduino DIE zum Blinken bringen können. Jetzt möchten wir den Schwierigkeitsgrad ein wenig erhöhen und zwei Leuchtdioden abwechselnd blinken lassen.

Aufgabenstellung

Schalte die grüne Leuchtdiode ein. Warte anschließend eine Sekunde und schalte die grüne Leuchtdiode wieder aus.

Schalte anschließend die rote Leuchtdiode ein. Warte erneut eine Sekunde und schalte die rote Leuchtdiode wieder aus.

Schritt 1:

Zuerst erweitern wir unserem Code aus Aufgabe 1 um die rote LED. Für diesen Schritt verwenden wir wieder eine Variable.

Dieser Variable weisen wir den Wert „12“ zu. Die Variable nennen wir „led_rot“.

```
int modul_led_rot = 12;
```

Schritt 2:

Unsere neue Variable ergänzen wir nun in unserem Programmcode aus Aufgabe 1.

Wir fügen die Variable in unserem Beispiel in Zeile 2 unseres Programmcodes ein.