Calliope-Kurs Kinder

04_Tag3

Jogi Künstner, Turbine Brunnen

Herbst 2020


Kurs Programmieren Lernen mit Calliope Mini

Zusammenfassung I

Heute wiederholen wir als erstes nochmal das Gelernte vom letzten Nachmittag vor den Ferien, insbesondere schauen wir uns nochmal **SEHR** genau an,

- wie Variablen verwendet werden,
- wie man sie mit Werten belegt,
- wie man die Werte abruft.
- Und wir schauen uns an, wie wir uns das gut einprägen können mit einem Vergelich aus dem normalen Leben:
- mit Obst-Körben
  - die mit Äpfeln gefüllt werden
  - und deren Apfel-Füllstand angeschaut wird


Zusammenfassung II

Im Anschluss daran schauen wir uns zuerst ein sehr wichtiges Kapitel beim Programmieren an:
Die **Wenn-Dann-Ansonsten** - Konstruktion sicherlich das wichtigste Element überhaupt beim Programmieren!
Dazu müssen wir uns etwas mit Annahmen, Wahrheits-Werten und Logik auseinandersetzen.

Daraufhin setzen wir uns mit dem eingebauten Temperatur-Messer im Calliope auseinander, versuchen eine Temperatur zu ermitteln und sie dann auch anzuzeigen, und zwar so, dass jeder sieht, dass es sich um eine Temperatur handelt.

Im Anschluss daran schauen wir uns noch die einzelne RGB-Leuchtdiode unterhalb des 5x5-LED-Feldes, wie man die in unterschiedlichen Farben leuchten lassen kann.

Und zum Abschluss verwenden wir das gewonnene Wissen um Temperatur, Wenn-Dann und die Ansteuerung der RGB-LED um eine persönliche Temperatur-Ampel zu bauen.

Die einzelnen Kapitel

- 01 Auffrischen: Variablen und Obstkörbe
- 02 Auffrischen: Ports / externe LED
- 03 Etwas Logik: Wenn-Dann
- 04 Der Temperatur-Sensor
- 05 Die RGB-Led, eine Temperatur-Ampel
- 06 Bessere Temperatur-Ampel/Fehlersuche

04_01_Auffrischen_Variablen

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Wiederholung / Auffrischen
Variablen

Platzhalter / Variablen

Da Variablen sozusagen der Grundpfeiler beim Programmieren sind, schauen wir uns in dieser Rückblende **nochmal** genau die Verwendung von Variablen und Ihren Einsatz in einfachen mathematischen Aufgaben an:

Platzhalter / Variablen dienen - wie der Name andeutet - dazu, veränderliche Werte aufzunehmen. Man will aber den Variablen nicht nur Werte “reingeben”, sondern man will dann auch die Variablen wieder “fragen” : Was für einen Wert hast Du denn? In der Programmierung sagt man:

- der Variable wird ein Wert **zugewiesen**
- der Wert der Variablen wird **abgefragt**


Zuweisung an eine Variable

Vorstellen kann man sich das auch als Zuruf mit dem Megaphon:
Hey Du, **Variable_A**, merk Dir doch bitte mal die **2**!

Zuweisung an eine Variable

Oder man stellt sich die Variable als Obstkorb vor, der mit Äpfeln “belegt” wird:

Korb_A :

Zuweisung an eine Variable

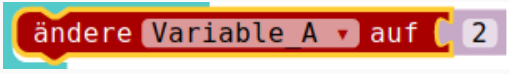
Belege den Korb_A mit 2:

Zuweisung an eine Variable

In einer “normalen” Programmiersprache : **Variable_A** $= 2$
Zuweisung an **Variable_A**, die Variable steht auf der linken Seite des Gleichheitszeichens, der Wert mit dem die Variable belegt werden soll, steht auf der rechten Seite des Gleichheitszeichens.

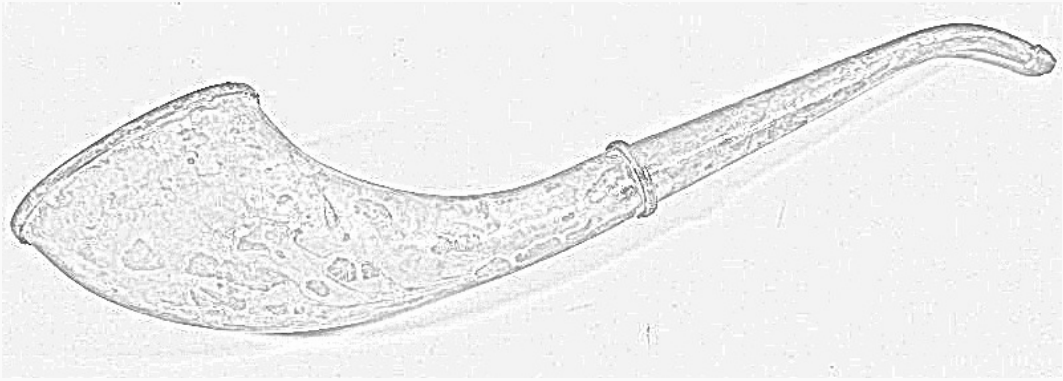

Zuweisung an eine Variable

Die Zuweisung an eine Variable in unsere Calliope-Programmier-Sprache sieht so aus:



Abfrage/Benutzen einer Variable

So wie man sich vorher die Zuweisung mit einem Megaphon vorstellen kann, so kann man sich das Auslesen/Benutzen der Variable vorstellen, dass man mit einem altmodischen Hör-Rohr fragt:
Hey Du, **Variable_A**, was war es, was Du Dir vorher merken solltest? Sag es mir doch bitte.



Abfrage/Benutzen einer Variable

Oder man stellt sich vor, wie man in den Obstkorb reinschaut, wieviele Äpfel denn dort drin sind, man **fragt** die **Belegung ab**:

Was ist der Inhalt von Korb_A :

Abfrage/Benutzen einer Variable

So wie in unserem Korb-Beispiel der Korb nur angeschaut wird, ist das auch beim Programmieren:

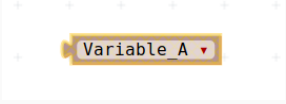
- Der Inhalt des Korbes ändert sich durchs Anschauen nicht
- Der Inhalt der Variable ändert sich durchs Auslesen, durch die Benutzung in einer Formel **nicht**.

In einer "normalen" Programmiersprache wird das üblicherweise z.B. so dargestellt:

- Variable_B = **Variable_A**
  - Der Wert der Variable: **Variable_A** wird erfragt, sie steht auf der rechten Seite des Gleichheitszeichens.

Abfrage/Benutzen einer Variable

In unserer grafischen Calliope-Programmiersprache wird das Benutzen der Variable einfach durch das Puzzleteil mit der Variable dargestellt:



Abfrage/Benutzen einer Variable

Dieses einzelne Puzzleteil kann man dann irgendwo anstatt festen Werten einklicken, also wenn man z.B. die Rechnung mit der **Variable_B** von oben nochmals benutzen will:

- **Variable_B** = **Variable_A**
  - Der Wert der Variable: **Variable_A** wird erfragt, sie steht auf der rechten Seite des Gleichheitszeichens.
  - Auf der linken Seite des Gleichheits-Zeichens wird das der Variable **Variable_B** zugewiesen,
  - die **Variable_B** wird mit dem selben Wert belegt.
- 

Ein paar Rechnungen

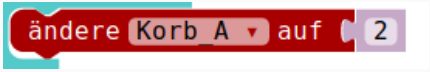
Nun haben wir also mehrere Darstellungs-Möglichkeiten für Variablen-Belegung und Variablen-Abfrage gesehen:

- “Körbchen”-Darstellung
- Darstellung mit Text
- Grafische Calliope-Darstellung

Damit wollen wir noch ein paar ganz einfache Belegungen, Abfragen und Rechnungen zeigen.

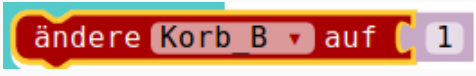

Korb A mit 2 belegen

Korb_A = 2



Korb B mit 1 belegen

Korb_B = 1



Korb C = Summe Korb A + B

Korb_C = Korb_A + Korb_B

Korb C = Summe Korb A + B

=

+

Korb B erhöhen um 1

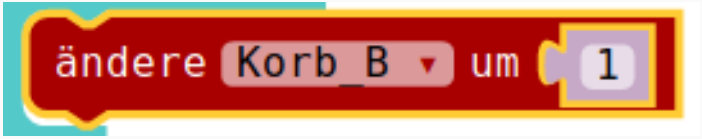
=

+

Korb B erhöhen um 1

Korb_B = Korb_B + 1

oder andere Möglichkeit, die auch leider einfach mit dem Setzen einer Variable zu verwechseln ist:



Was enthält nun Korb C?

Vorher haben wir den Korb C mit der Summe von Korb A und Korb B belegt haben.
Das Ergebnis war 3. Nun haben wir den Korb B um eins erhöht.
Ist nun der Wert des Korb C auch um 1 erhöht worden?
NEIN!
Die Berechnung wurde vorher ausgeführt.
Das Ergebnis ändert sich nicht mehr.
Der Korb C enthält immer noch die 3.

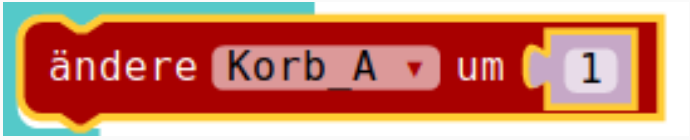
Wir schauen nochmal nach :-)

Korb A erhöhen um 1

Ebenso können wir nun natürlich Korb A um eins erhöhen

Korb_A = Korb_A + 1

oder andere Möglichkeit, die auch leider einfach mit dem Setzen einer Variable zu verwechseln ist:



Korb A erhöhen um 1

=

+

Korb C = Summe Korb A + B

Nun können wir nochmal die Berechnung von vorher durchführen: Korb C soll die Summe von Korb A und von Korb B enthalten.

In “normaler” Programmiersprache:

Korb_C = Korb_A + Korb_B

- Korb A enthält 3 Äpfel
- Korb B enthält 2 Äpfel
- Korb C enthält vor der Berechnung schon 3 Äpfel
- Wenn wir nun die Berechnung durchführen, was enthält denn dann Korb C?
  - **5 Äpfel** = 3 Äpfel von Korb A und 2 Äpfel von Korb B ? Oder etwa:
  - **8 Äpfel** = 3 Äpfel von Korb A und 2 Äpfel von Korb B und die 3 eigenen Äpfel von vorher?
- ?
- ?
- ?


Antwort : Korb C = Summe Korb A + B

Lösung: So wie wir die Berechnung formuliert haben:

Korb C = Korb A + Korb B

ist der vorherige Inhalt von Korb C egal! Er wird quasi vorher ausgeleert!

Also sieht unsere Korb-Rechnung nun so aus:

Antwort : Korb C = Summe Korb A + B

=

+

Antwort : Korb C = Summe Korb A + B

Und in Calliope-Rechnung sieht das ganz wieder genau gleich wie vorher aus:

04_02_Auffrischen_Ampel

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Wiederholung / Auffrischen Ampel

Spannung / Verbraucher 1

Hier nur die wichtigsten Dinge bezüglich Spannung / Verbraucher vom letzten Nachmittag:

- Die Steckdose ist **tabu** !
- Unsere Spannungs-Bereiche liegen zwischen 1.5 Volt und ca 12 Volt, die Steckdose hat **220 Volt**
- Der Calliope arbeitet normalerweise mit 3.3 Volt
- Die Grösse einer Batterie sagt **NICHTS** über ihre Spannung aus!

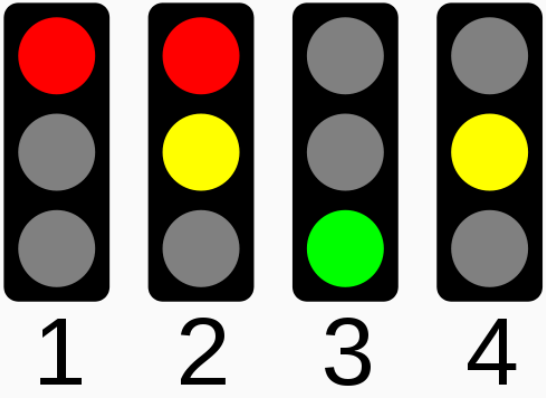
Spannung / Verbraucher 2

- Spannung von Lieferant (Batterie) und Verbraucher (z.B. LED etc) müssen übereinstimmen
- Bei Nichtübereinstimmung kann etwas kaputt gehen, wenn es dumm läuft ist das der Calliope
- Kurzschluss heisst der Pluspol und der Minus-Pol eines Spannungs-Lieferanten werden zusammengehalten/“kurzgeschlossen”
- Beim Kurzschluss geht üblicherweise etwas kaputt, wenn es dumm läuft, ist das der Calliope.

Ampel

- Die Ampel und die Ansteuerung der Pins ging letztes Mal vielleicht ein kleines bisschen (zu?) schnell.
- Darum in diesem Auffrischen das Ganze nochmal Schritt für Schritt

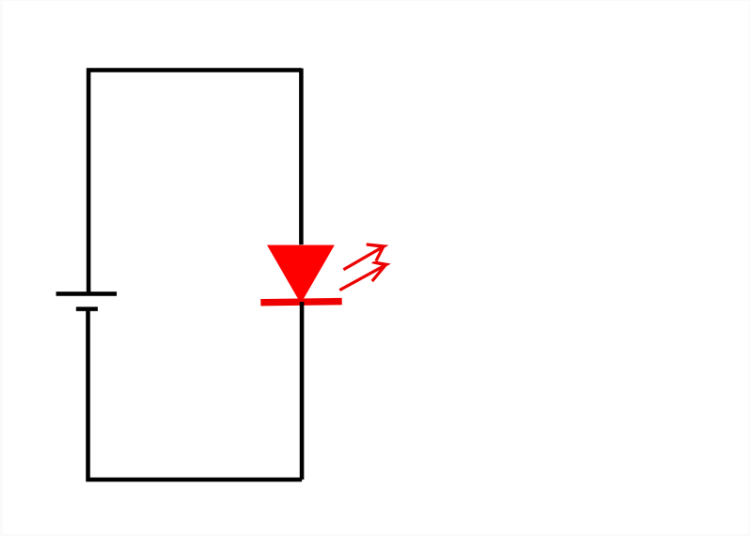
Wir möchten gerne eine Ampel haben, die folgendes ermöglicht:



Ohne Calliope 1

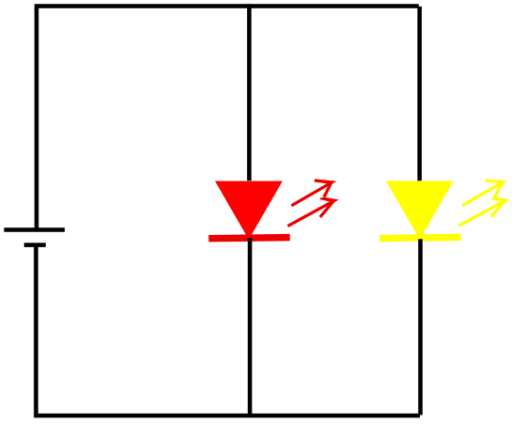
Wir brauchen eine Batterie

Ohne Calliope 2



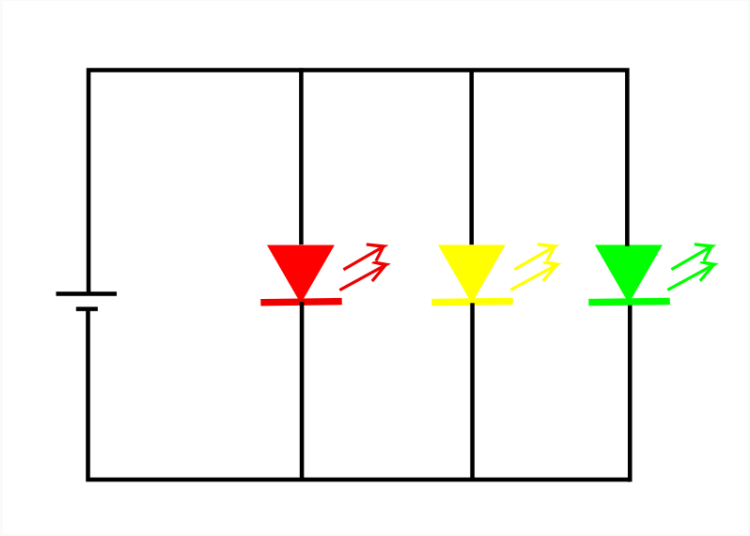
Wir brauchen eine rote LED

Ohne Calliope 3



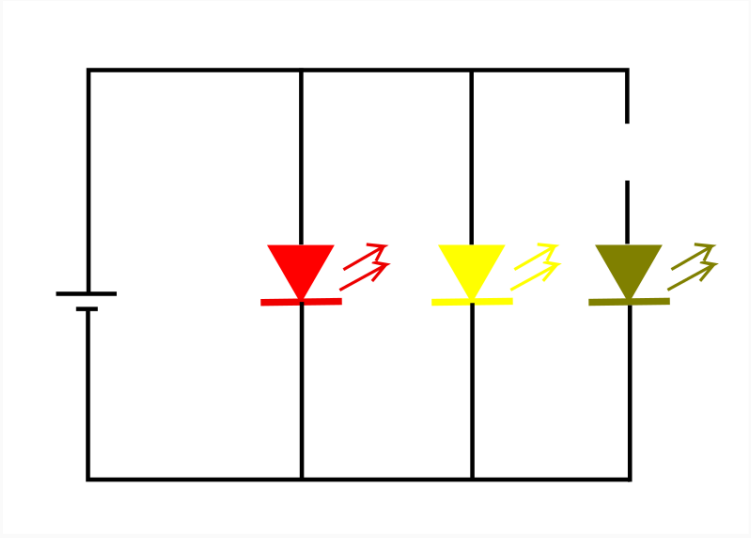
Wir brauchen eine gelbe LED

Ohne Calliope 4



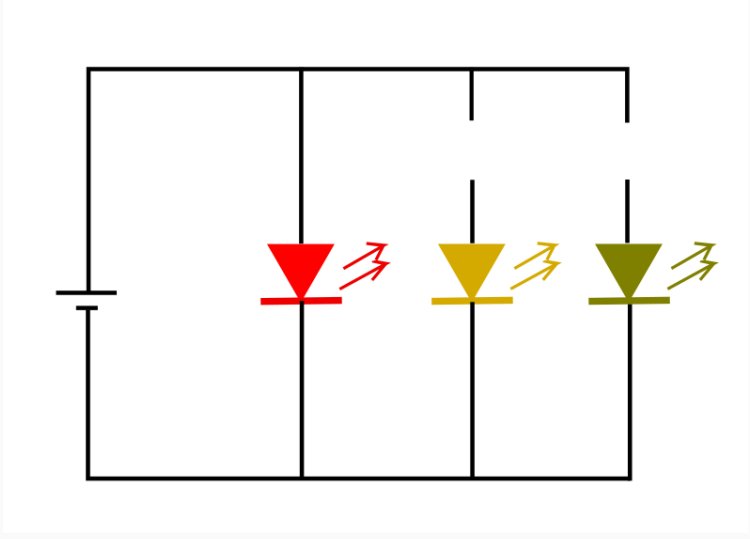
Wir brauchen eine grüne LED

Ohne Calliope 5



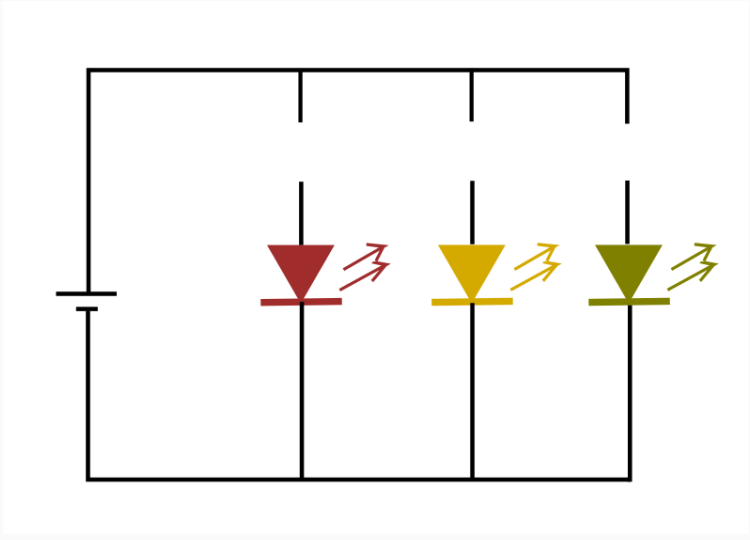
Wir trennen Grün auf

Ohne Calliope 6



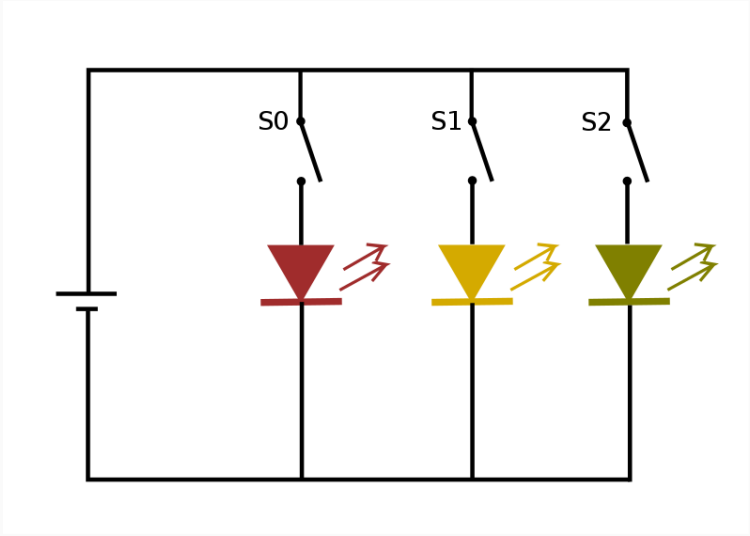
Wir trennen Gelb auf

Ohne Calliope 7



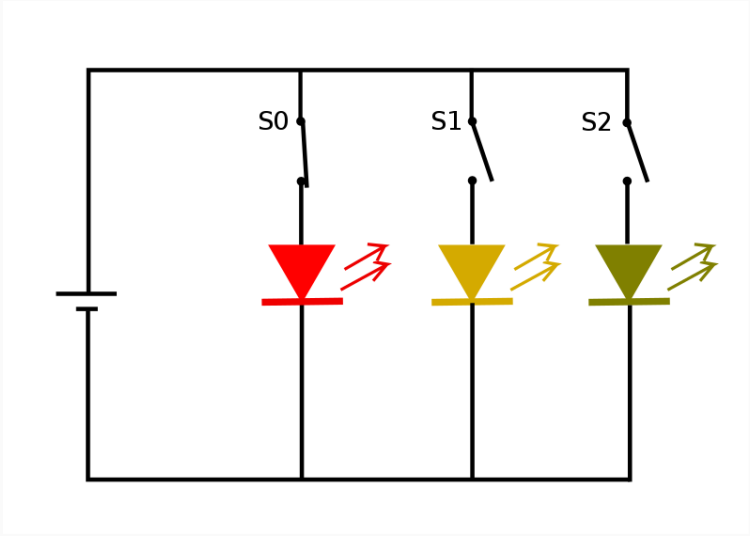
Wir trennen Rot auf

Ohne Calliope 8



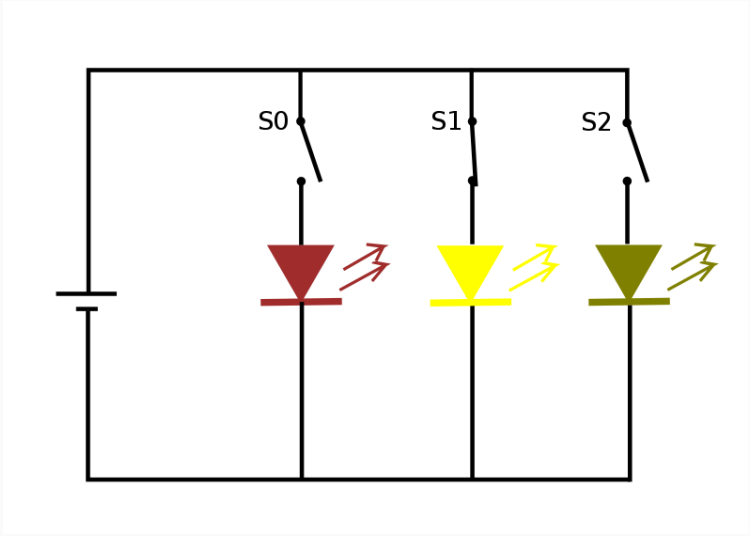
Wir bauen Schalter ein (S0 - S2)

Ohne Calliope 9



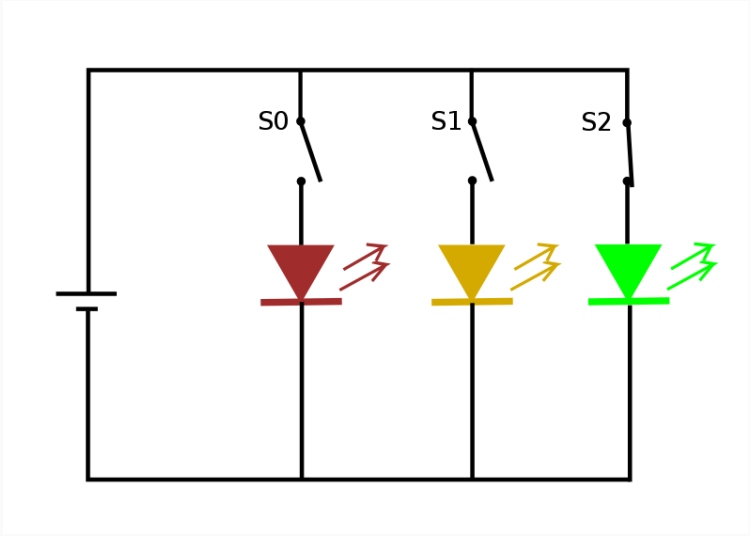
Schalter S0 einschalten : Rot

Ohne Calliope 10



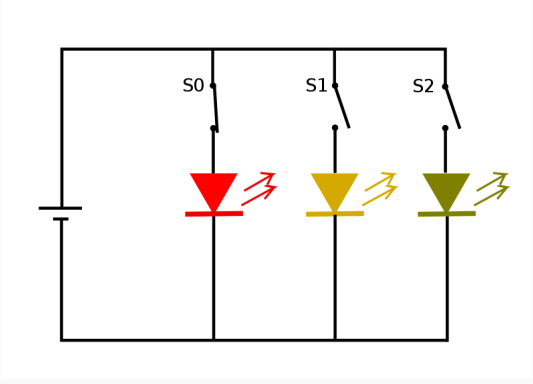
Schalter S1 einschalten : Gelb

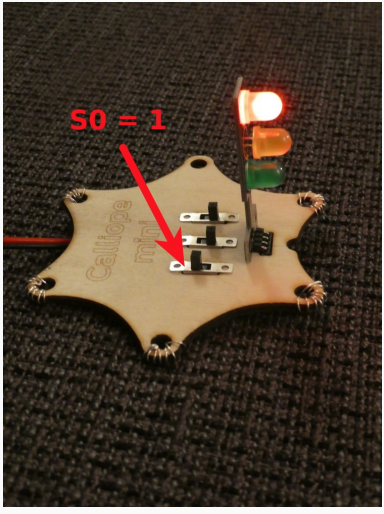
Ohne Calliope 11



Schalter S2 einschalten : Grün

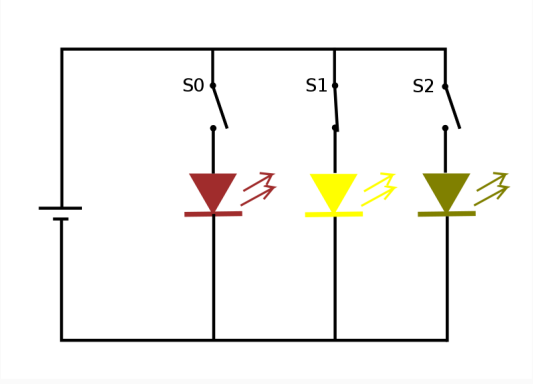
Ohne Calliope 12

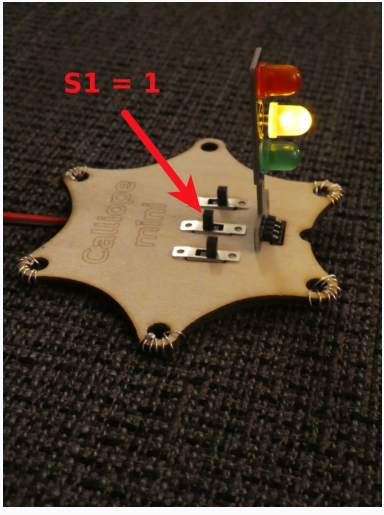




Schalter S0 einschalten : Rot

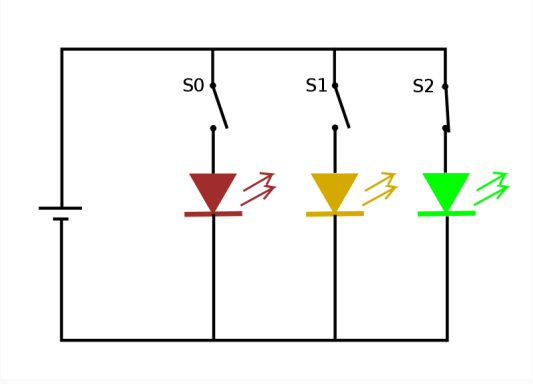
Ohne Calliope 13

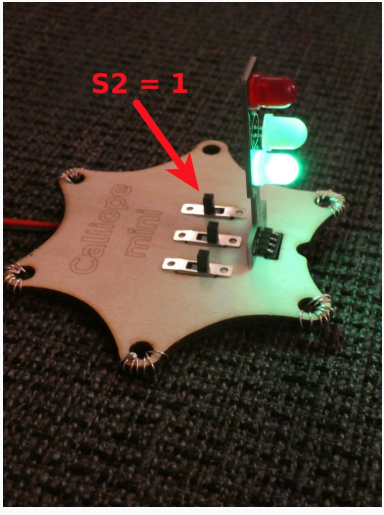




Schalter S1 einschalten : Gelb

Ohne Calliope 14





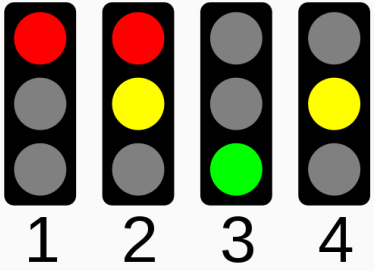
Schalter S2 einschalten : Grün

Wie die Ampel funktioniert

- Alle Schalter aus => Alle Lampen aus
- S0 = 0 : Rot aus
- S0 = **1** : Rot **ein**
- S1 = 0 : Gelb aus
- S1 = **1** : Gelb **ein**
- S2 = 0 : Grün aus
- S2 = **1** : Grün **ein**


Ein Ampel-Zyklus (1)

- Alle Lampen aus
- Rot ein => **Ampel Rot (1)**
- Rotzeit abwarten (z.B. 5 sek)
- Gelb ein => **Ampel Rot-Gelb (2)**
- Rot-Gelbzeit warten (z.B. 1 sek)
- Rot aus
- Gelb aus
- Grün ein => **Ampel Grün (3)**
- Grünzeit warten (z.B. 5 sek)
- Grün aus
- Gelb ein => **Ampel Gelb (4)**
- Gelbzeit warten (z.B. 1 sek)
- Wieder von vorne



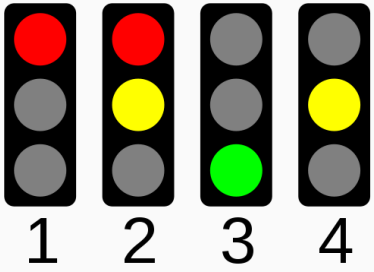
Ein Ampel-Zyklus (2)

- Alle Lampen aus
- Rot ein => **Ampel Rot**
- Rotzeit abwarten (z.B. 5 sek)
- Gelb ein => **Ampel Rot-Gelb**
- Rot-Gelbzeit warten (z.B. 1 sek)
- Rot aus
- Gelb aus
- Grün ein => **Ampel Grün**
- Grünzeit warten (z.B. 5 sek)
- Grün aus
- Gelb ein => **Ampel Gelb**
- Gelbzeit warten (z.B. 1 sek)
- Wieder von vorne

- S0,S1,S2 =0 => Lampen aus
- S0=1 => **Ampel Rot**
- Rotzeit abwarten (z.B. 5 sek)
- S1=1 => **Ampel Rot-Gelb**
- Rot-Gelbzeit warten (z.B. 1 sek)
- S0=0
- S1=0
- S2=1 => **Ampel Grün**
- Grünzeit warten (z.B. 5 sek)
- S2=0
- S1=1 => **Ampel Gelb**
- Gelbzeit warten (z.B. 1 sek)
- Wieder von vorne

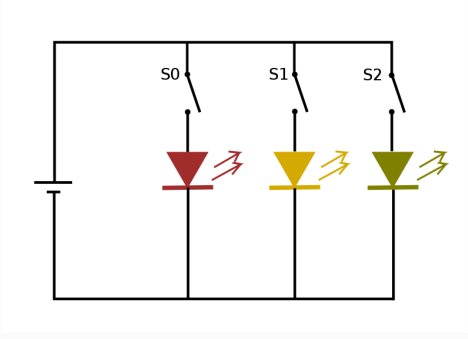
Ein Ampel-Zyklus (3)

- S0,S1,S2 =0 => Lampen aus
- S0=1 => **Ampel Rot (1)**
- Rotzeit abwarten (z.B. 5 sek)
- S1=1 => **Ampel Rot-Gelb (2)**
- Rot-Gelbzeit warten (z.B. 1 sek)
- S0=0
- S1=0
- S2=1 => **Ampel Grün (3)**
- Grünzeit warten (z.B. 5 sek)
- S2=0
- S1=1 => **Ampel Gelb (4)**
- Gelbzeit warten (z.B. 1 sek)
- Wieder von vorne

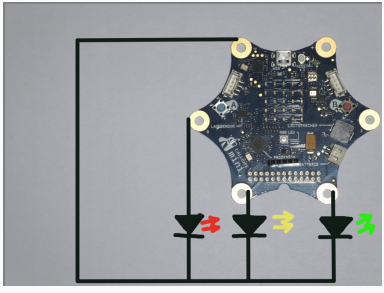


Ersetzen Hardware

Wir ersetzen Schalter

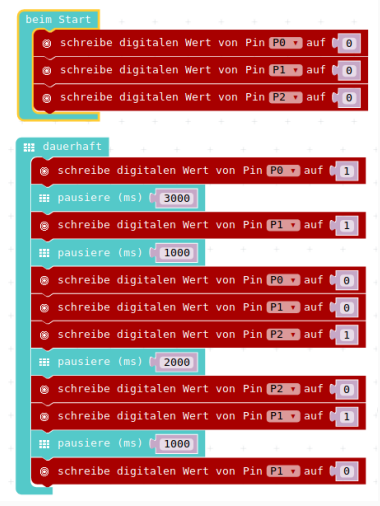


Durch den Calliope



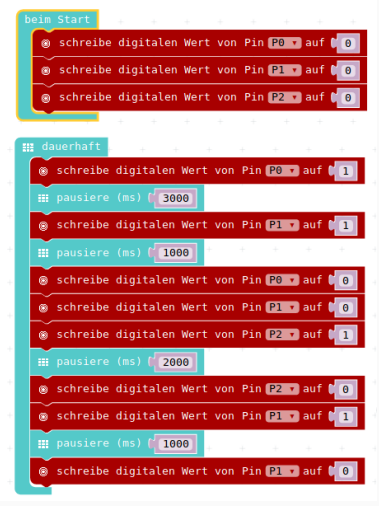
Ersetzen Software : Schalter S durch Pin P

- S0,S1,S2 =0 => Lampen aus
- S0=1 => **Ampel Rot (1)**
- Rotzeit abwarten (z.B. 5 sek)
- S1=1 => **Ampel Rot-Gelb (2)**
- Rot-Gelbzeit warten (z.B. 1 sek)
- S0=0
- S1=0
- S2=1 => **Ampel Grün (3)**
- Grünzeit warten (z.B. 5 sek)
- S2=0
- S1=1 => **Ampel Gelb (4)**
- Gelbzeit warten (z.B. 1 sek)
- Wieder von vorne

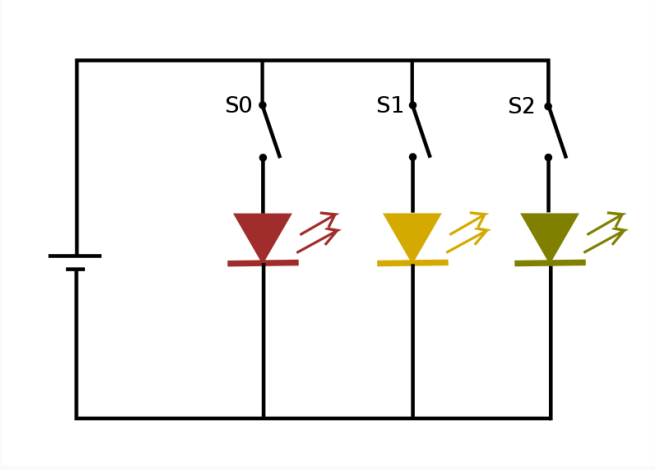


Ersetzen Software : Schalter S durch Pin P

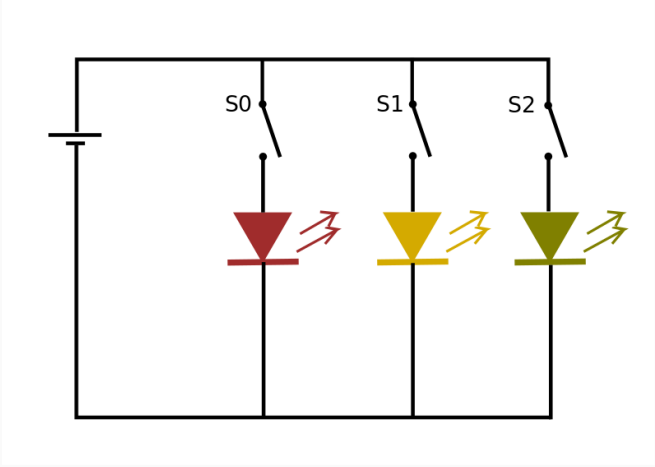
- Das Bild auf der rechten Seite
- ist das Programm vom letzten Mal
- es entspricht fast der Anweisung mit den Schaltern
- Im Programm setzen wir unten Grün auf aus
- In der Schalt-Anweisung:
- Schalten wir oben **ALLE** aus



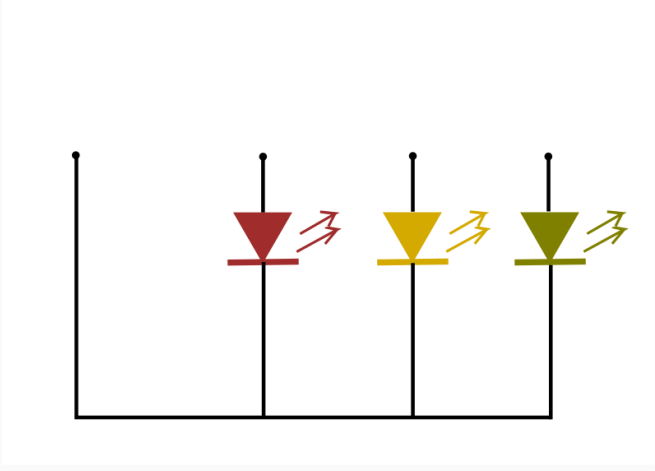
Anschluss LEDs (1)



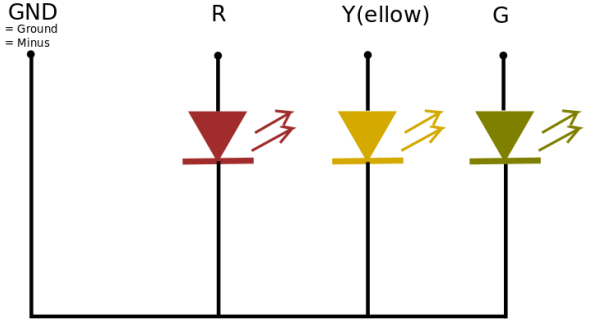
Anschluss LEDs (2)



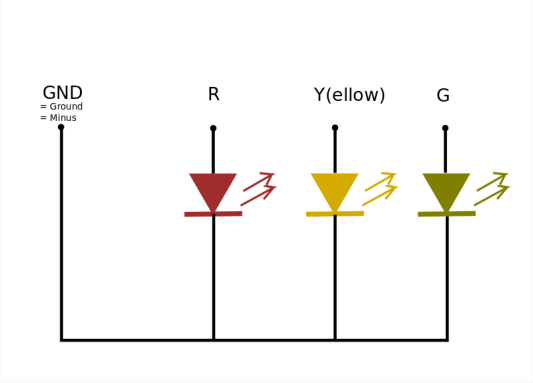
Anschluss LEDs (3)



Anschluss LEDs (4)



Anschluss LEDs (5)



04_03_Wenn-Dann

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Logik, Vergleiche, Wahrheit

Weltmeisterschaft / Fussball-Wetten / Vergleiche

- Wer hat vor 2 Jahren schaut Fussball-WM geschaut?
- Wer hat dabei Wetten gemacht" ?
- **Wenn** die Schweiz ein Tor mehr schiesst als Deutschland,
- **dann** bekomme ich von Dir zwei zusätzliche Panini-Bilder?


- Wer streitet hin und wieder mit seinem Freund?
- **Wenn** Du mir das Lego nicht gibst,
- **dann** bekommst Du von mir keine Schokolade mehr.

und so weiter...


Das alles sind Vergleiche, die normalerweise dann im Anschluss überprüft werden können und sich entweder als wahr oder falsch erweisen.

Je nachdem, ob wahr oder falsch, wird dann etwas gemacht:

- Du bist nicht mehr mein Freund
- Du musst mir zwei Panini geben
- Ich muss Dir eine Schoki geben
- ...


Das ist eines der wichtigen Eigenschaften auch beim Programmieren:

- Einen **Vergleich** machen, der ein Ergebnis hat,
- dies ist im Allgemeinen **wahr** oder **falsch**
- Basierend auf dem **wahr** oder **falsch**
- wird dann etwas unterschiedliches **gemacht**,
- es wird eine **Aktion** ausgelöst.

Das wollen wir jetzt auch machen


Das Menu Logik

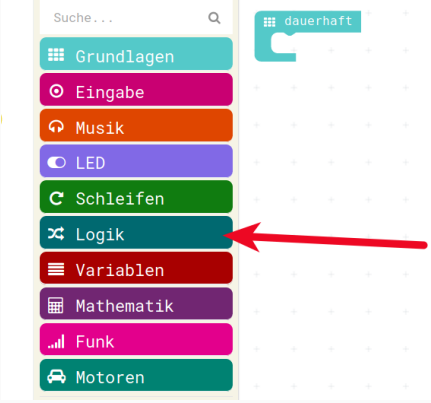


Figure 1: LogikMenu

Das Menu Logik

enthält die Wenn-Dann Programmierung, es enthält die Vergleiche die wir machen wollen und es enthält auch “Wahr” und “Falsch” - Werte

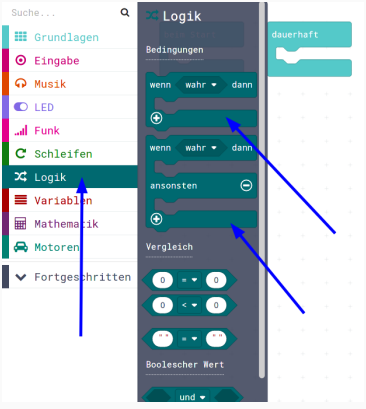


Figure 2: Logik Wenn Dann Ansonsten

Wenn Dann in der Dauerschleife

Das Wenn-Dann ziehen wir in den Arbeitsbereich in die Dauer-Schleife

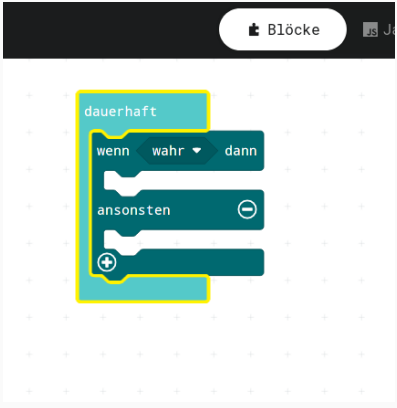


Figure 3: Wenn Wahr

Symbole in der Wenn-Dann

Nun können wir mit zwei einfachen Symbolen auf unserem “Display” anzeigen, wie sich das Wenn-Dann verhaält

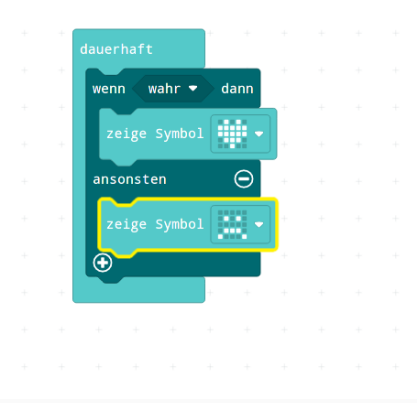


Figure 4: Wenn Wahr Symbol

Wenn-Dann Ergebnis Wahr

In der Wenn-Dann-Abfrage kommt immer oben die **Aktion**, die gemacht werden soll, wenn die Aussage **wahr** ist, darunter kommt das, was gemacht werden soll, wenn die Aussage sich als **falsch** erweist.

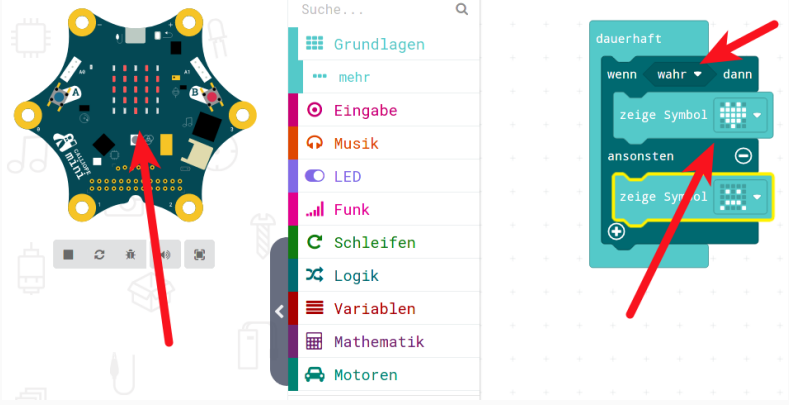


Figure 5: Wenn Wahr Symbol_Wahr

Wenn-Dann Ergebnis Falsch

So sieht das Ganze aus, wenn die Aussage **Falsch** ist.

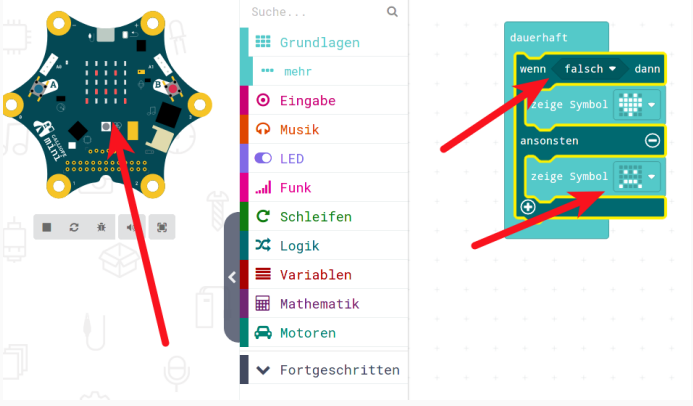


Figure 6: Wenn Wahr Symbol Falsch

Einsatz der farbigen RGB-Leucht-Diode

Wir haben ja auch eine farbige Leuchtdiode auf dem Calliope, die soll nun zum Einsatz kommen.

Anstatt Symbole auf dem 5x5 - roten LED-Display wollen wir die LED in unterschiedlichen Farben leuchten lassen.

Die LED befindet sich auch unter Grundlagen (auch zu erkennen an der Farbe!)

Farbigen RGB-Leucht-Diode in Wenn-Dann

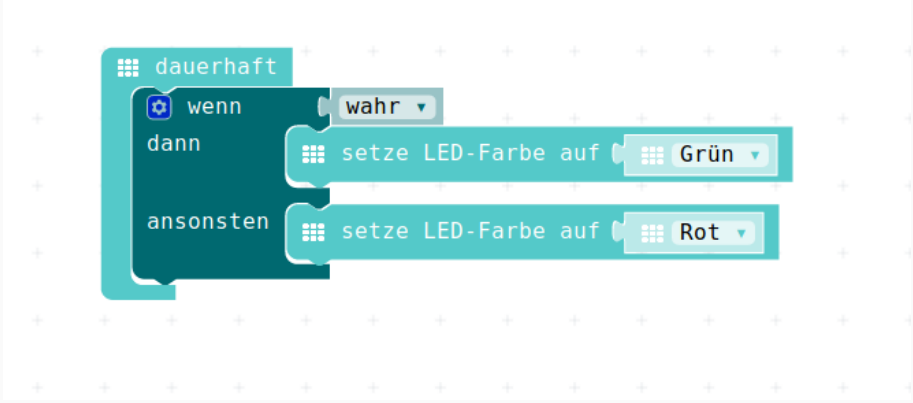


Figure 8: RGB Led In Wenn Wahr

Farbigen RGB-Leucht-Diode leuchtet Grün

Da momentan der Wert **Wahr** in die Abfrage reingeben wird, leuchtet unsere RGB-Led in Grün.

Jetzt ist übrigens ein guter Moment, um das nicht nur im Simulator auszuprobieren, sondern das Programm als HEX-Datei zu speichern und auf dem Calliope-Board auszuprobieren.

Vergleiche

Nun wollen wir aber **echte** Vergleiche machen, dazu können wir zum Beispiel zwei Zahlen miteinander vergleichen und das Ergebnis auswerten:

- Zwei ist grösser als Fünf : **Falsch**
- Sechs ist grösser als Fünf : **Wahr**
- Sechs ist grösser als Sechs : **Falsch**
- Zehn ist kleiner als Sechs : **Falsch**
- Zehn ist gleich Zehn : **Wahr**
- Acht ist gleich Neun : **Falsch**


Vergleiche

Das kleiner-Zeichen habt Ihr in Mathematik wahrscheinlich auch noch nicht gehabt, aber es ist eigentlich **selbst** sprechend:

- Kleinere Zahl $<$ Grössere Zahl
- Grössere Zahl $>$ Kleinere Zahl

Damit wird:

- Zwei ist grösser als Fünf : $2 > 5$: **Falsch**
- Sechs ist grösser als Fünf : $6 > 5$: **Wahr**
- Sechs ist grösser als Sechs : $6 > 6$: **Falsch**
- Zehn ist kleiner als Sechs : $10 < 6$: **Falsch**
- Zehn ist gleich Zehn : $10 = 10$: **Wahr**
- Acht ist gleich Neun : $8 = 9$: **Falsch**

Vergleiche

Figure 10: Logik Vergleiche

Vergleich aus dem Menu holen

und anstelle von “**Wahr**” in die Wenn-Dann reinsetzen

Nun haben wir einen - **noch sinnlosen** - Vergleich:

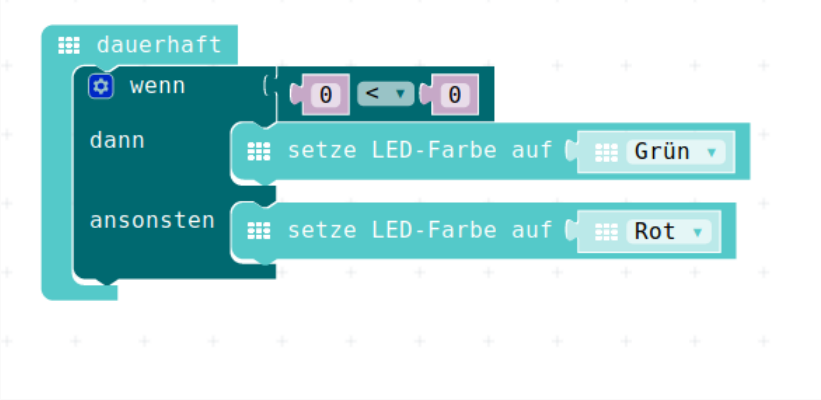


Figure 11: Wenn Dann Vergleich

Vergleich auf > grösser setzen

Mit dem kleinen Dreieck beim Vergleich können wir nun denn kleiner-Vergleich auf einen Grösser-Vergleich umbauen.

Figure 12: Wenn Dann Groesser

Vergleich auf sinnvolle Werte

Nun nehmen wir zwei Werte in den Vergleich.

Die Werte sind eigentlich egal, ich habe 22 und 21 genommen, das wäre gut, wenn Ihr das auch macht, dann können wir später sehen warum...

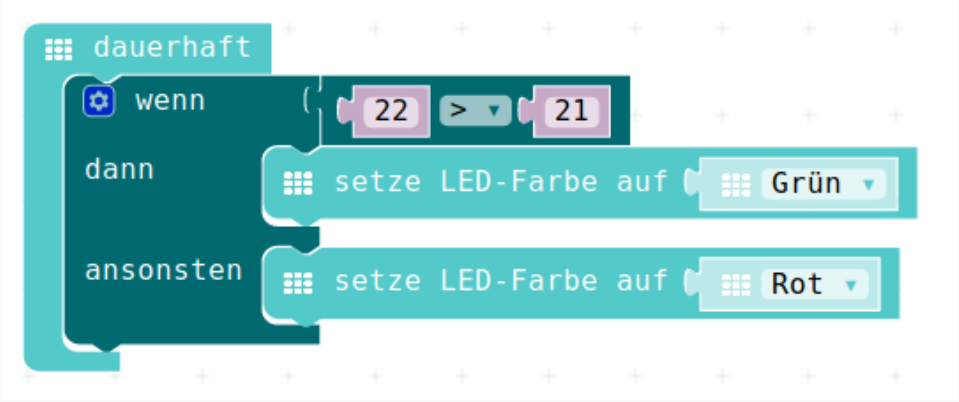


Figure 13: Wenn Dann Groesser:True

Vergleich ergibt grünes LED

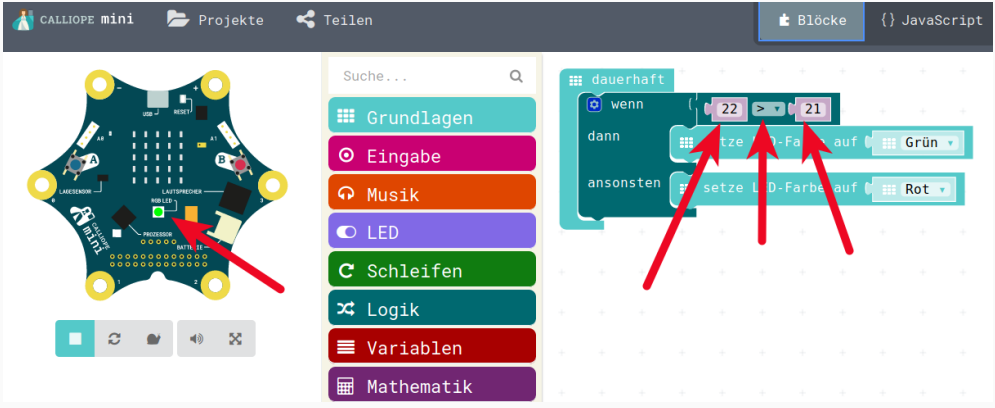


Figure 14: Wenn Dann Groesser:True

JavaScript-Code

Java-Script-Code

```
basic.forever(() => {
    if (22 > 21) {
        basic.setLedColor(Colors.Green)
    } else {
        basic.setLedColor(Colors.Red)
    }
})
```

Download Hex-Code:

Hex-code


04_04_TemperaturSensor

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Der Temperatur-Messer

Sensor/Eingang/Input

Neben vielen anderen Sensoren/Eingängen, ein paar davon haben wir schon kennengelernt, hat der Calliope auch einen **Temperatur-Sensor**.

Diesen wollen wir nun als sinnvollen Eingangs-Wert für unsere Wenn-Dann Abfrage benutzen.

Davor wollen wir aber den Sensor selbst kurz kennenlernen, sehen wie man den abfrägt und was er für Werte liefert.

Dazu räumen wir unsere Dauerschlefe frei:


Freiräumen der Dauerhaft-Schleife

Herausziehen der bisherigen **Wenn-Dann** -Programmierung zur Seite. **NICHT** löschen, wir wollen das später noch benutzen

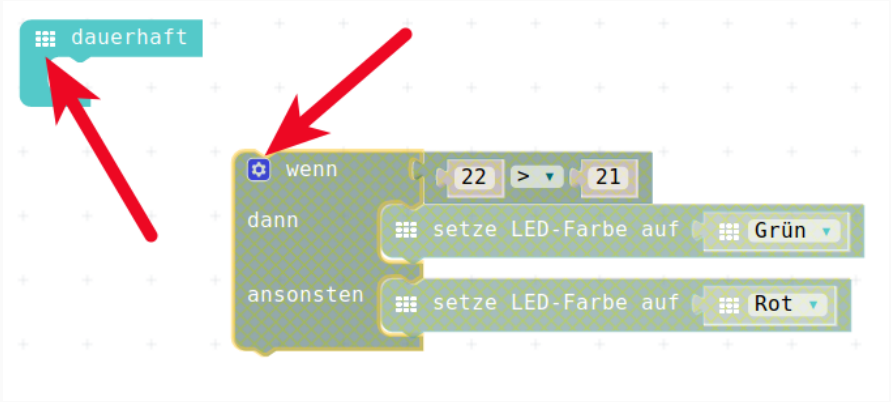


Figure 1: Schleife frei räumen

Das Menu Eingabe

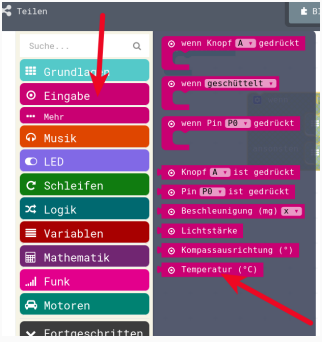


Figure 2: Menu Input

enthält eine Spezial-Variable, eine Input-Variable, namens **Temperatur**. Um diese sinnvoll weiter zu verwenden, legen wir Menu **Variablen** eine eigene Variable an, die wir zum Beispiel ***AktuelleTemperatur*** nennen.

Variable anlegen

Nun legen wir uns wieder eine neue Variable namens ***AktuelleTemperatur*** an.

Figure 3: Variable anlegen

Variable benennen

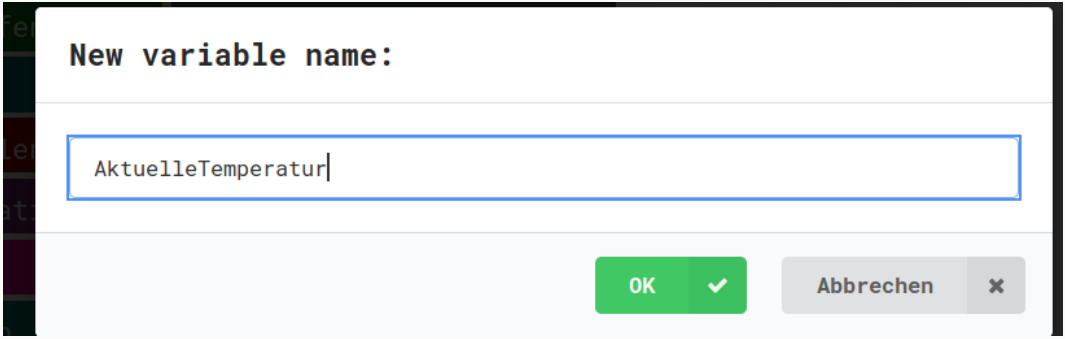


Figure 4: Variable benennen

Temperatur auslesen

Nun belegen wir also die neu angelegte Variable ***AktuelleTemperatur*** mit der Temperatur, wie sie aus dem Eingabe-Menu kommt.

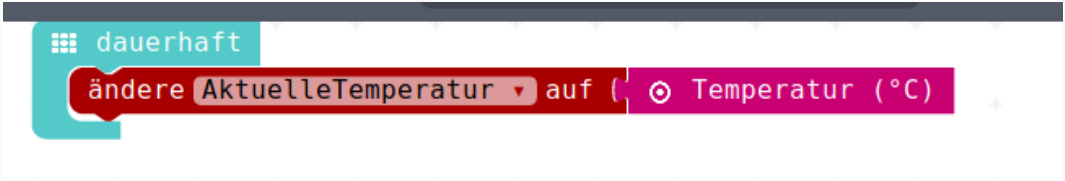


Figure 5: Variable zuweisen

Temperatur anzeigen

und holen uns noch aus dem Grundlagen-Menu das **zeige Nummer** und zeigen damit die Variable an, die die aktuelle Temperatur beinhaltet.

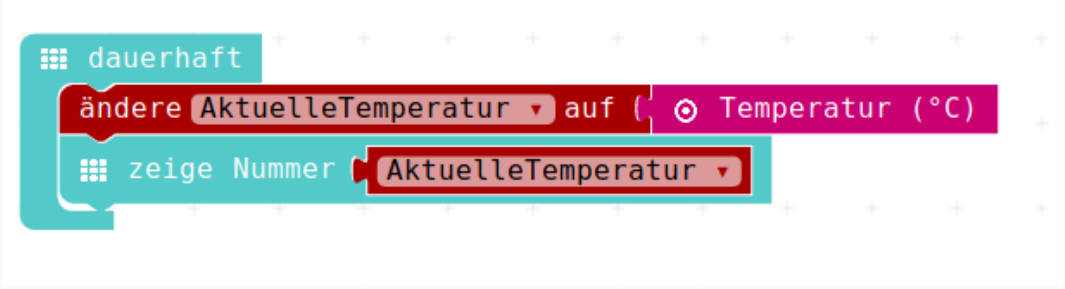


Figure 6: Temperatur anzeigen

Temperatur im Simulator

Nun schauen wir uns das im Simulator an:

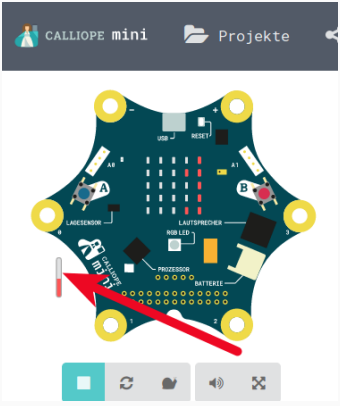


Figure 7: Temperatur im Simulator

Mit der Maus kann man an diesem Thermometer die Temperatur verändern.

Temperatur im Simulator

Die Ausgabe kann noch etwas verbessert werden:

- die Zahlen “rauschen” nur durch und man weiss nicht ob das nun 21 oder 12 sind
- Man sieht nicht, dass es sich um eine Temperatur in Grad Celcius (°C) handelt.

Verbesserung der Anzeige

Mit einem vereinfachten “**°C**” als Symbol und ein paar “warte ms” und Bildschirmlöschen (das ist alles im Menu Grundlagen, zum Teil in “. . . Mehr”) gibt die Anzeige dann schon was her.

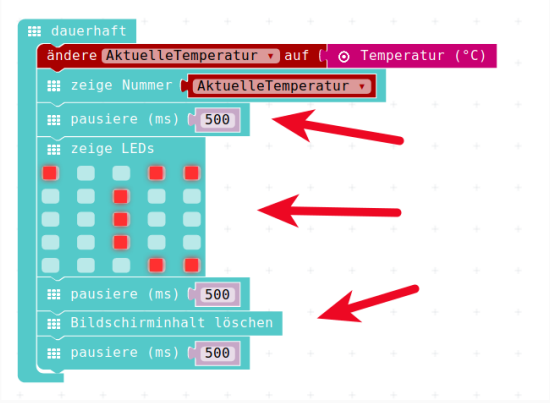


Figure 8: Verbesserte Anzeige

Download in der Calliope

Jetzt ist der Programm-Code eigentlich gut genug, um eine echte Messung in unserem echten Calliope durchzuführen.

Wir laden das Programm dazu auf den Calliope:

- Dem Ganzen unten einen sinnvollen Namen geben (z.B. Temperatur-Messer_01)
- Den Speichern-Knopf (Diskette) drücken
- Das erzeigte und geladene HEX-File im **Download** - Ordner finden
- Das HEX-File kopieren , “STRG-C”
- Den Calliope-Mini anschliessen
- Das HEX-File auf dem Calliope “fallen lassen”, “Einfügen”, “STRG-V”
- Der Calliope sollte anfangen das neue Programm in seinen Speicher zu übertragen.

JavaScript-Code

Java-Script-Code

```
let AktuelleTemperatur = 0
basic.forever(() => {
    AktuelleTemperatur = input.temperature()
    basic.showNumber(AktuelleTemperatur)
    basic.pause(500)
    basic.showLeds(`
        # . . # #
        . . # . .
        . . # . .
        . . # . .
        . . . # #
        `)
    basic.pause(500)
    basic.clearScreen()
    basic.pause(500)
})
```


Download HEX-Datei

Download Hex-Code

Hex-code

04_05_TemperaturAmpel

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Die Temperatur-Ampel

Wiedereinbau Wenn-Dann

Nun können wir aus der Wenn-Dann Übung von vorher und dem kleinen Temperatur-Programm von eben eine kleine Temperatur-“**Ampel**” machen.

Dazu bauen wir das vorher “beiseite” gelegte **Wenn-Dann** wieder ein.

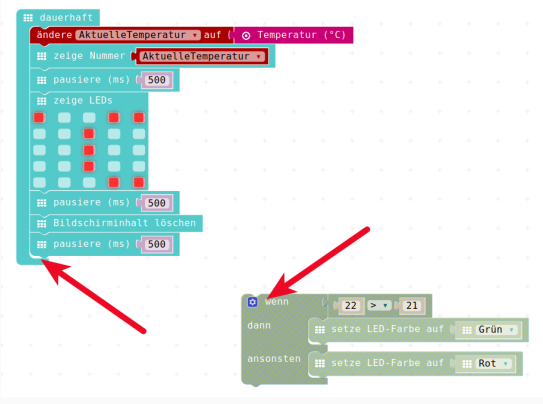


Figure 1: Wieder Einbau

Wiedereinbau Wenn-Dann

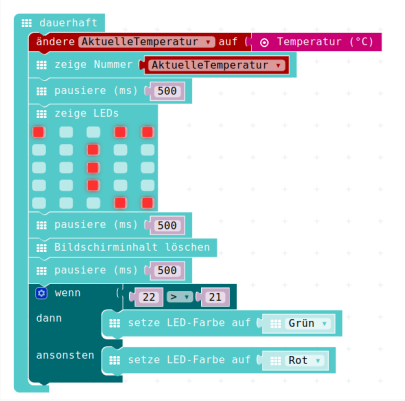


Figure 2: Eingebaut

Verwendung sinnvoller Werte

Wir möchten im ersten Schritt, "**Alles im Grünen Bereich**" anzeigen, wenn die Temperatur eine **gute** Temperatur hat.

Dazu sagen wir : Alles was grösser 21 °C ist, ist gut.

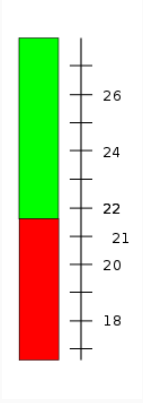


Figure 3: Thermometer

Verwendung sinnvoller Werte

Das heisst: Wir müssen unsere Wenn-Dann-Konstruktion so umbauen, dass folgender Satz/Aussage abgebildet wird:

- Wenn die Temperatur grösser als 21 °C ist,
  - **dann** soll die RGB-LED grün leuchten,
  - **ansonsten** soll sie rot leuchten.

Verwendung sinnvoller Werte

- Nun ergeben auch die vorhin “ganz willkürlich gewählten” Werte 22 und 21 etwas Sinn...
- Wir ersetzen die konstante 22 in der Wenn-Dann-Abfrage durch die jetztige Temperatur.
- Diese befindet sich in der Variable ***AktuelleTemperatur***
- Dazu holen wir uns die Variable ***AktuelleTemperatur*** aus dem Menu Variablen

Verwendung sinnvoller Werte

Figure 4: Variable holen

Verwendung sinnvoller Werte

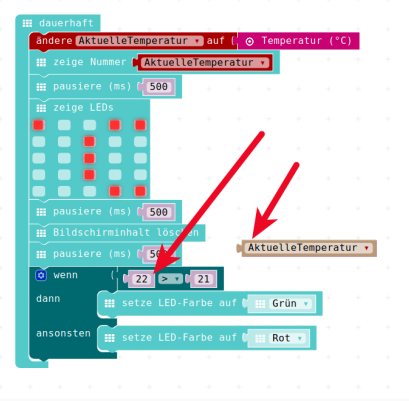


Figure 5: Variable liegt da

Verwendung sinnvoller Werte

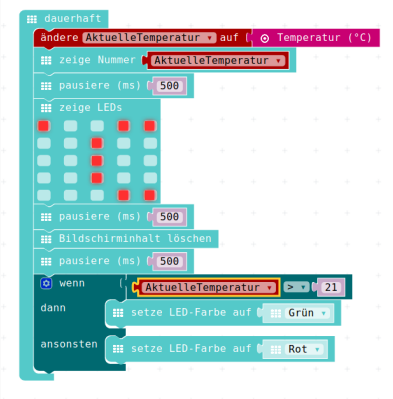


Figure 6: Variable wird benutzt

Damit lässt sich im Simulator schon mal ausprobieren, wie unsere Temperatur-Ampel reagiert.

Benutzung im Simulator

Beim Starten ist im Simulator die Temperatur immer 21 °C, das ist nach unseren Wünschen genau die Grenze. Erst wenn die ***AktuelleTemperatur*** grösser als 21 °C ist, dann wird die Anzeige grün. Das können wir im Simulator ausprobieren und dann natürlich auch wieder in den Calliope laden um es in der richtigen Hardware mit echten Werten zu testen.

Download in der Calliope

Jetzt ist der Programm-Code eigentlich gut genug, um eine echte Messung in unserem echten Calliope durchzuführen.

Wir laden das Programm dazu auf den Calliope:

- Dem Ganzen unten einen sinnvollen Namen geben (z.B. Temperatur-Messer_01)
- Den Speichern-Knopf (Diskette) drücken
- Das erzeigte und geladene HEX-File im **Download** - Ordner finden
- Das HEX-File kopieren
- Den Calliope-Mini anschliessen
- Das HEX-File auf dem Calliope “fallen lassen”, “Einfügen”, “CTRL-V” * Der Calliope sollte anfangen das neue Programm in seinen Speicher zu übertragen.

(So, das war jetzt aber das letzte Mal eine detaillierte Anleitung zum Laden des Programms in den Calliope, in Zukunft kommt nur noch der Hinweis, dass wir das Programm in den Calliope laden...)

JavaScript-Code

Java-Script-Code

```
let AktuelleTemperatur = 0
basic.forever(() => {
    AktuelleTemperatur = input.temperature()
    basic.showNumber(AktuelleTemperatur)
    basic.pause(500)
    basic.showLeds(`
        # . . # #
        . . # . .
        . . # . .
        . . # . .
        . . . # #
        `)
    basic.pause(500)
    basic.clearScreen()
    basic.pause(500)
    if (AktuelleTemperatur > 21) {
        basic.setLedColor(Colors.Green)
    } else {
```


Download HEX-Datei

Download Hex-Code

Hex-code

04_06_TemperaturAmpelBesser

Calliope-Kurs Kinder

Jogi Künstner, Turbine Brunnen

Herbst 2020


Die verbesserte Temperatur-Ampel

Echter grüner Bereich

Nun wollen wir die Temperatur-Ampel etwas verbessern.

- Wir legen einen grünen Bereich zwischen grösser als 21 °C und bis 25 °C fest
- **Wenn** die Temperatur kleiner/gleich 21 °C ist, **dann** soll das Licht blau sein, es ist uns zu kalt (blau wie am Wasserhahn)
- **Wenn** die Temperatur in unserem "grünen" Bereich ist, **dann** soll natürlich auch die LED grün leuchten
- **Wenn** die Temperatur grösser als 25 °C ist, **dann** soll die LED rot leuchten (rot wie am Wasserhahn)

Echter grüner Bereich

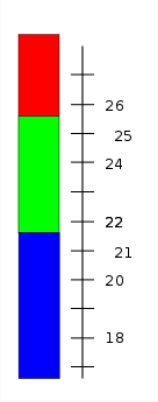


Figure 1: Neue Temperatur-Bereiche

Wenn-Dann erweitert

Ebenso wie im “echten Leben” kann man auch das Wenn-Dann - Programmier-Konstrukt erweitern.

- **Wenn** xyz **Dann** macheDas
- **Ansonsten Wenn** abc **Dann** macheJenes
- **Ansonsten Wenn** def **Dann** machedochnochwasanderes
- **Ansonsten** MacheEinfachIrgendwas

Mit solch einem Konstrukt können wir nun unserer Temperatur-Abfrage erweitern um eine zusätzliche Abfrage grösser 25° C
Und die Farben müssen wir natürlich auch noch anpassen.

Erweiterung in der Programmier-Oberfläche

Um das Wenn-Dann - Konstrukt in der Programmier-Oberfläche zu erweitern, muss im “Wenn-Dann-Puzzle-Stück” das (+) benutzt werden.

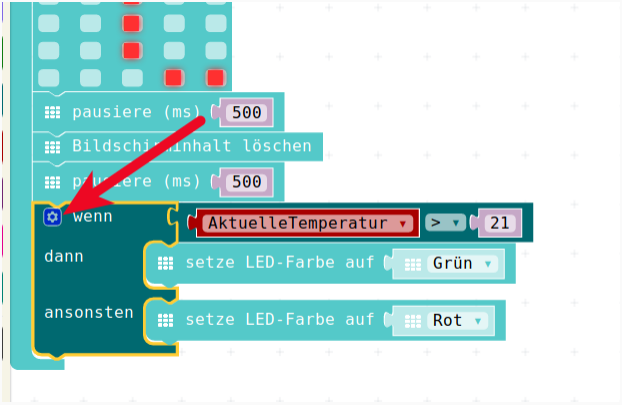


Figure 2: ToolBox

Erweiterung in der Programmier-Oberfläche

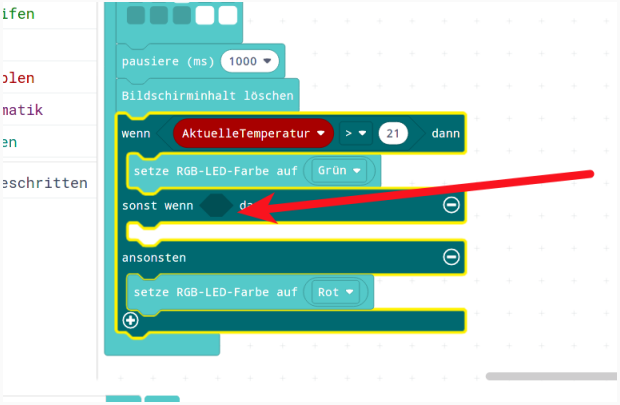


Figure 3: Toolbox Miniatur

Einbau der zusätzlichen Abfragen

Nun können wir also in die zusätzlichen Abfragen unsere weiteren Überprüfungen auf Temperatur $>$ 25 einklicken (am Besten die Überprüfung per rechter Maustaste von oben kopieren) und die LED-Farben-Setzen befehle einklicken und die Farben entsprechend ändern.

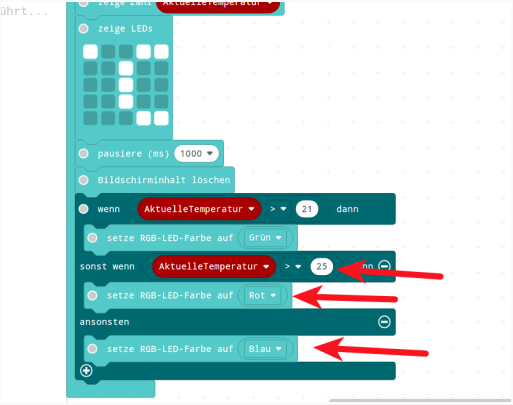


Figure 4: Neues Programm Elself

Ein Bug (ein Fehler) !

- Es geht nicht!
- Wir bekommen kein **rot** zu sehen!
- Was ist falsch?

Dazu können wir mal versuchen, die Temperatur auf $> 26°C$, also z.B. 30°C einzustellen und dann das Programm anschauen / beobachten.

Dazu eignet sich der **Käfer**.


De“bug”ging

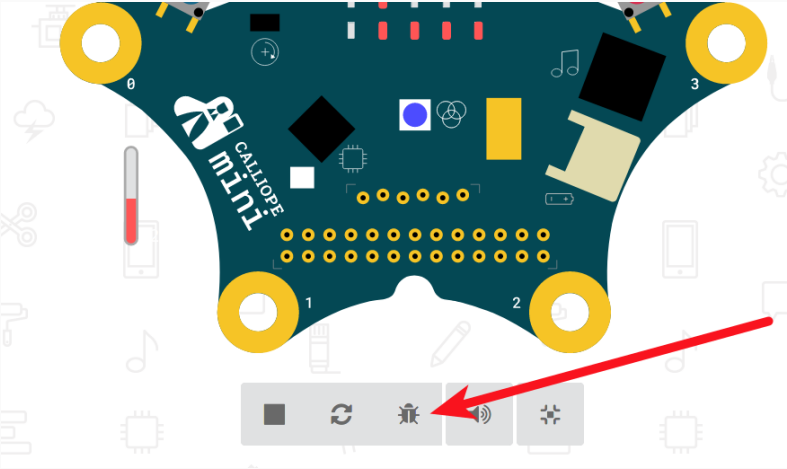
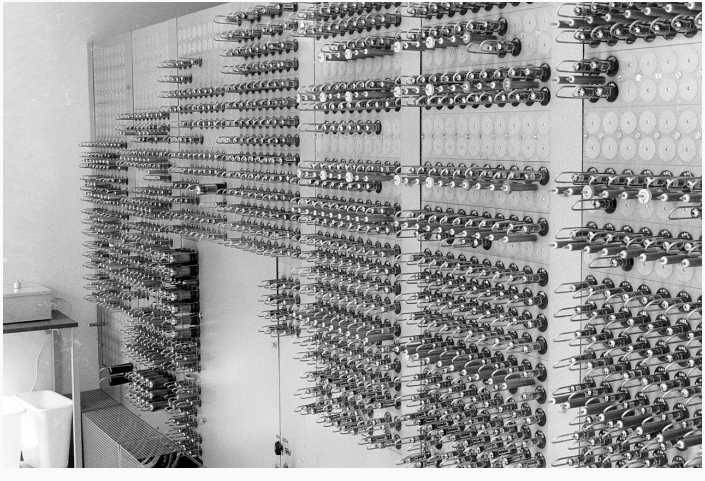


Figure 5: Debugging

Ein Bug (ein Fehler) !

- Ein Programmier-Fehler
- auch “Bug” genannt
- Bug ist english und heisst Käfer / Fliege
- Was hat das mit Programmieren zu tun?

Ein Röhren-Computer



(By Heinz Reutersberg, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=80182474)

Elektronen-Röhre

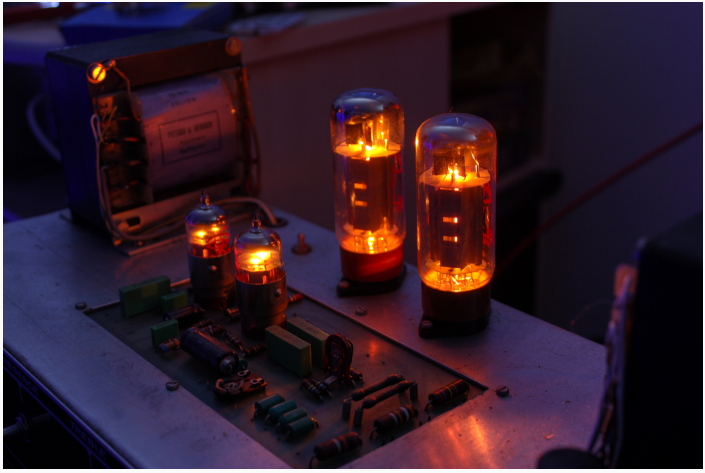


Figure 6: Elekronen-Roehre

(By Christopher Schirner, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=53326983)

Der erste Bug jemals gefunden

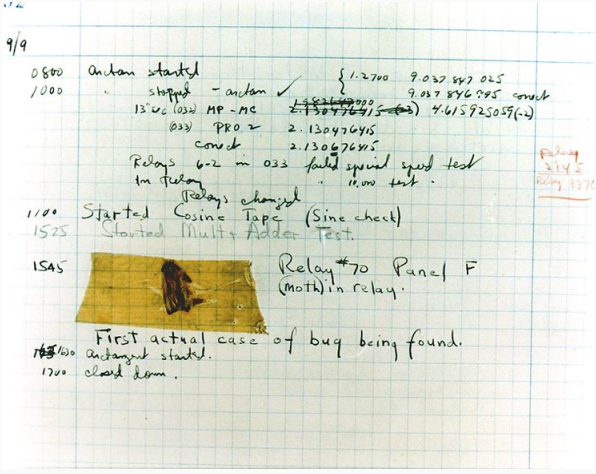


Figure 7: Erster Computer Bug

(U.S. Naval Historical Center Online Library Photograph NH 96566-KN)

Ein Bug (ein Fehler) !

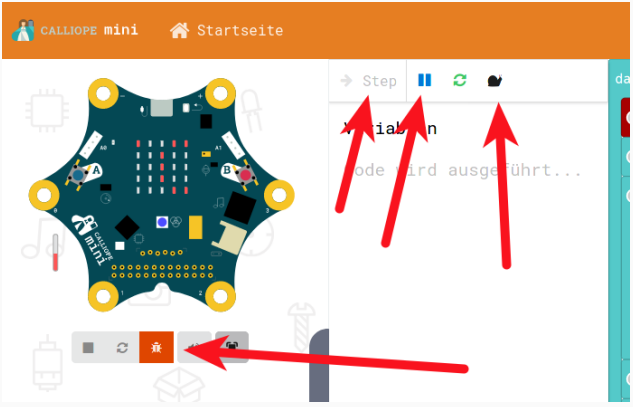


Figure 8: Bugsuche Schnecke

Das ist schonmal eine grosse Hilfe und könnte uns bei der Fehlersuche unter die Arme greifen.

Ein Bug (ein Fehler) !

Was passiert?
Auch eine Temperatur von z.B. 30°C, die ja als rot angezeigt werden soll, geht durch das ganze Wenn-Dann-Konstrukt durch. Wenn irgendeine Bedingung erfüllt ist, dann wird die zugehörige Aktion durchgeführt und dann das Konstrukt verlassen.

- Als erstes wird die gemessene aktuelle Temperatur von 30°C überprüft, ob sie grösser ist als 21 °C.
- 30°C **IST** grösser als 21°C
- Der Vergleich liefert das Ergebnis **WAHR**
- Also wird die zueghörige Aktion durchgeführt: Setzen der Farbe auf grün
- Das **Sonst Wenn** wird gar nicht erreicht und darum dann auch die Überprüfung auf > 25°C erst gar nicht durchgeführt!

Umbau des Wenn-Dann-Konstrukts

Nachdem wir diesen Fehler gefunden haben, müssen wir unser **“Wenn-Dann”** - Konstrukt umbauen:

- als erstes Vergleich auf $>$ 25 °C $=>$ ROT
- als zweites Vergleich auf $>$ 21 °C $=>$ GRÜN
- ansonsten $=>$ BLAU

Vor dem Umbau spielen wir das hier einmal durch :

Vergleich auf “Papier”

gemessener Wert : **30**

- als erstes Vergleich auf $>$ 25 °C : **WAHR** $\Rightarrow$ ROT und Ende
- Ergebnis : **ROT**

Vergleich auf “Papier”

gemessener Wert : **24**

- als erstes Vergleich auf $>$ 25 °C : **FALSCH** $\Rightarrow$ Weiter
- als zweites Vergleich auf $>$ 21 °C : **WAHR** $\Rightarrow$ GRÜN und Ende
- Ergebnis : **GRÜN**

Vergleich auf “Papier”

gemessener Wert : **19**

- als erstes Vergleich auf $>$ 25 °C : **FALSCH** $=>$ Weiter
- als zweites Vergleich auf $>$ 21 °C : **FALSCH** $=>$ Weiter
- ansonsten $=>$ BLAU
- Ergebnis : **BLAU**


Neu zusammensetzen

Nun ziehen wir also unsere Vergleichs-Puzzle-Teile und unsere RGB-LED-Farben-Setz-Puzzle-Teile raus:

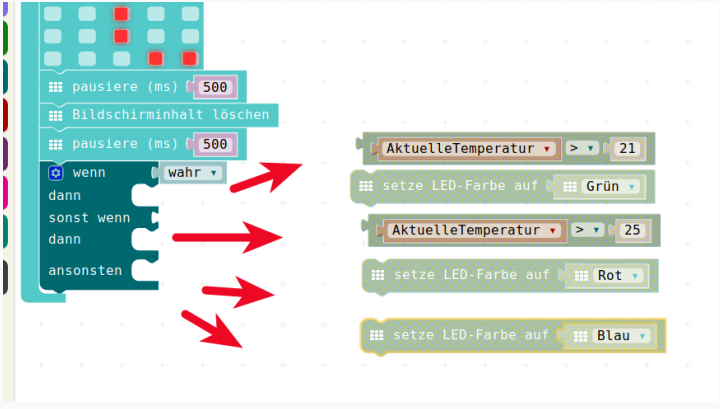


Figure 9: If Else Frei Geraeumt

Neu zusammensetzen

und setzen es wie angedacht wieder zusammen.

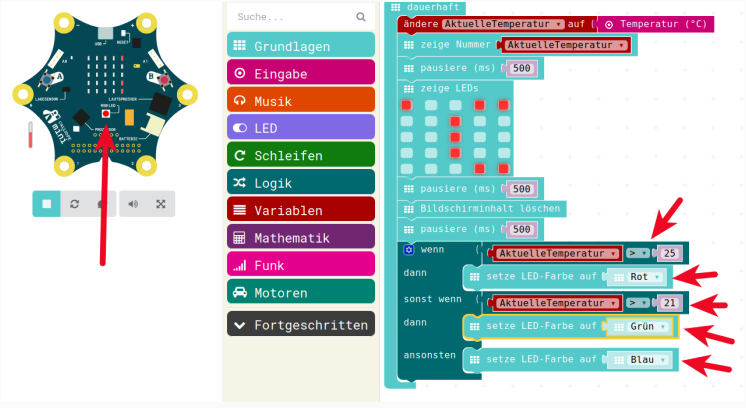


Figure 10: Neue Struktur Und Geht

Wenn wir nun die Temperatur mit der Maus im Simulator ändern, dann sehen wir, dass Farb-Anzeige unsere LED wir gewünscht funktioniert.

Download in der Calliope

Jetzt ist der Programm-Code eigentlich gut genug, um eine echte Messung in unserem echten Calliope durchzuführen.

Wir laden das Programm dazu auf den Calliope

JavaScript-Code

Java-Script-Code

```
let AktuelleTemperatur = 0
basic.forever(() => {
    AktuelleTemperatur = input.temperature()
    basic.showNumber(AktuelleTemperatur)
    basic.pause(500)
    basic.showLeds(`
        # . . # #
        . . # . .
        . . # . .
        . . # . .
        . . . # #
        `)
    basic.pause(500)
    basic.clearScreen()
    basic.pause(500)
    if (AktuelleTemperatur > 25) {
        basic.setLedColor(Colors.Red)
    } else if (AktuelleTemperatur > 21) {
```


Download HEX-Datei

Download Hex-Code

Hex-code

Weitere Schritte

Schaut Euch doch die “Hausaufgabe” an, da wird als neues Eingangs-Element der Licht-Messer verwendet und dann mit Hilfe einer **Wenn-Dann** - Konstruktion und Ton-Ausgabe eine kleine Schubladen-Alarm-Anlage gebaut.

Auf zur Schubladen-Alarm-Anlage


Navigation

- Zurück zu Tag 2 Hausaufgabe
- Hoch zur Übersicht
- Weiter zu Tag 3 Hausaufgabe (Schubladen-Alarm-Anlage)


Lizenz/Copyright-Info

Für alle Bilder auf dieser Seite gilt:

- Autor: Jörg Künstner
- Lizenz: CC BY-SA 4.0