

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung



# Einführung in die Programmierung Tag 2

1

# Kurze Wiederholung von Tag 1

# Einführung: Übersicht

- Was ist ein Programm?
- Was sind Befehle, Schleifen, Variablen und Bedingungen?
- Was ist grafische Programmierung?
- Und wie funktioniert das alles?

# Einführung: Programme

- Was ist eigentlich ein Programm?
- Habt Ihr da Ideen?

# Einführung: Programme

- ▶ Ein Programm ist eine Liste von Befehlen an einen Computer
- ▶ Wie ein Kochrezept oder eine Anleitung
- ▶ Die Befehle werden in der Reihenfolge abgearbeitet
- ▶ Oft sind Programmiersprachen aus simplen Befehlen aufgebaut, diese können aber kombiniert werden
- ▶ In vielen Programmiersprachen erkennst Du Befehle an den runden Klammern danach, Beispiele mp3-player: „play()“, „next()“, „stop()“



# Arduino – Eine open source Prototyping Plattform



- Der Arduino ist ein Mini-Computer
- Mit dem Arduino lassen sich schnell und einfach Sensoren und LEDs oder Motoren ansteuern
- Durch den Arduino vereinfacht sich der Umgang mit Elektronik

Seeed Grove –  
Sensoren (Licht, Temperatur, etc)  
Aktoren (Motoren, Displays, LEDs, etc.)  
Kabel zum Stecken statt löten



# Steckboard

Analoge Steckplätze  
A0 – A3

Digitale Steckplätze  
D2 – D8

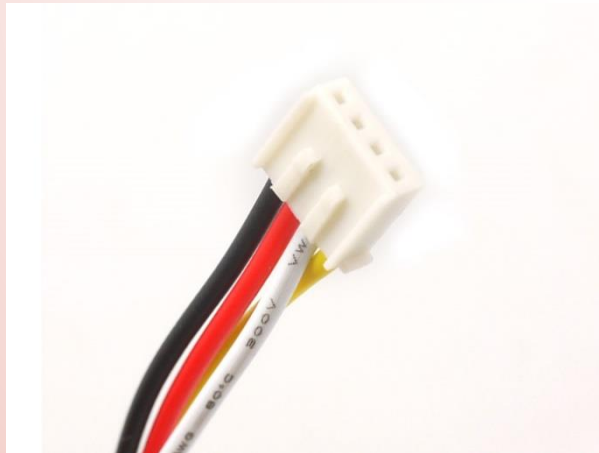


Display Steckplätze  
I2C



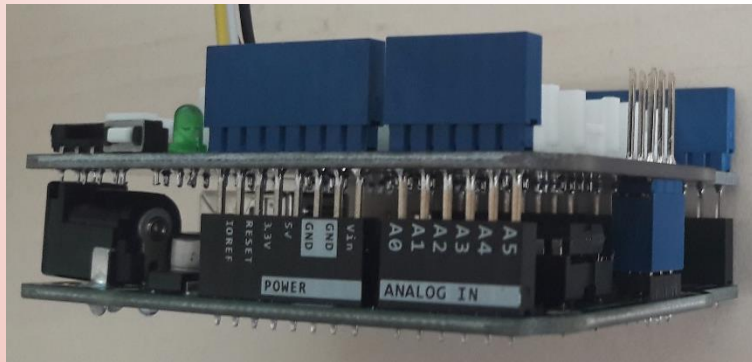
# Kabel

- Die Steckkabel dienen zum Verbinden der einzelnen Komponenten (LEDs, Sensoren, ...)
- Die Stecker haben eine „Richtung“, zu erkennen an den „Nasen“ an einer Seite, die Gegenseite (Buchse) dazu die passenden Ausbuchtungen

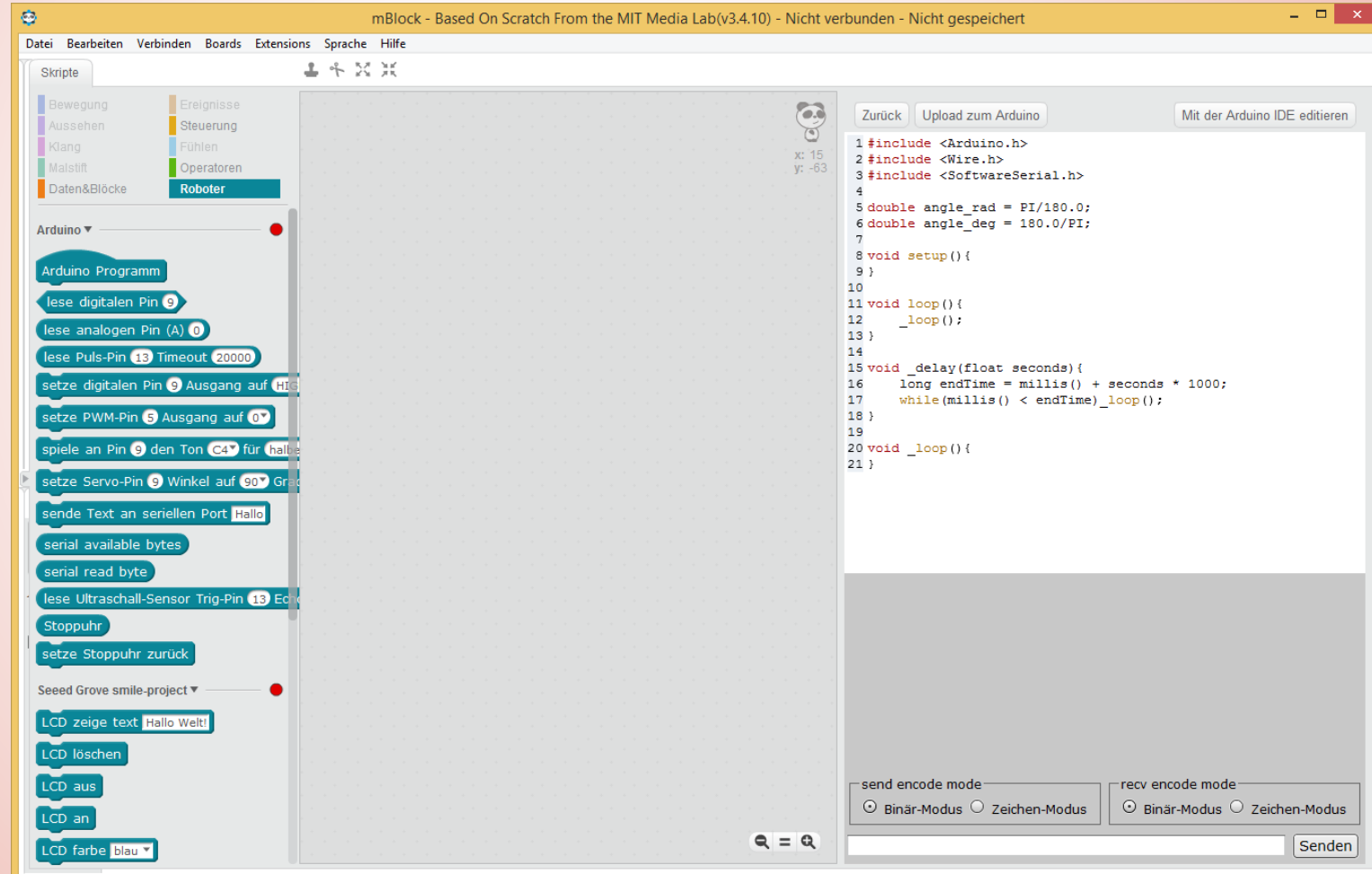


# Steckboard

- Steckt das Steckboard vorsichtig auf den Arduino auf, alle „Stifte“ müssen in die entsprechende Gegenseite
- Der kleine Schalter links unten muss auf 5V (rechte Schalterstellung) stehen

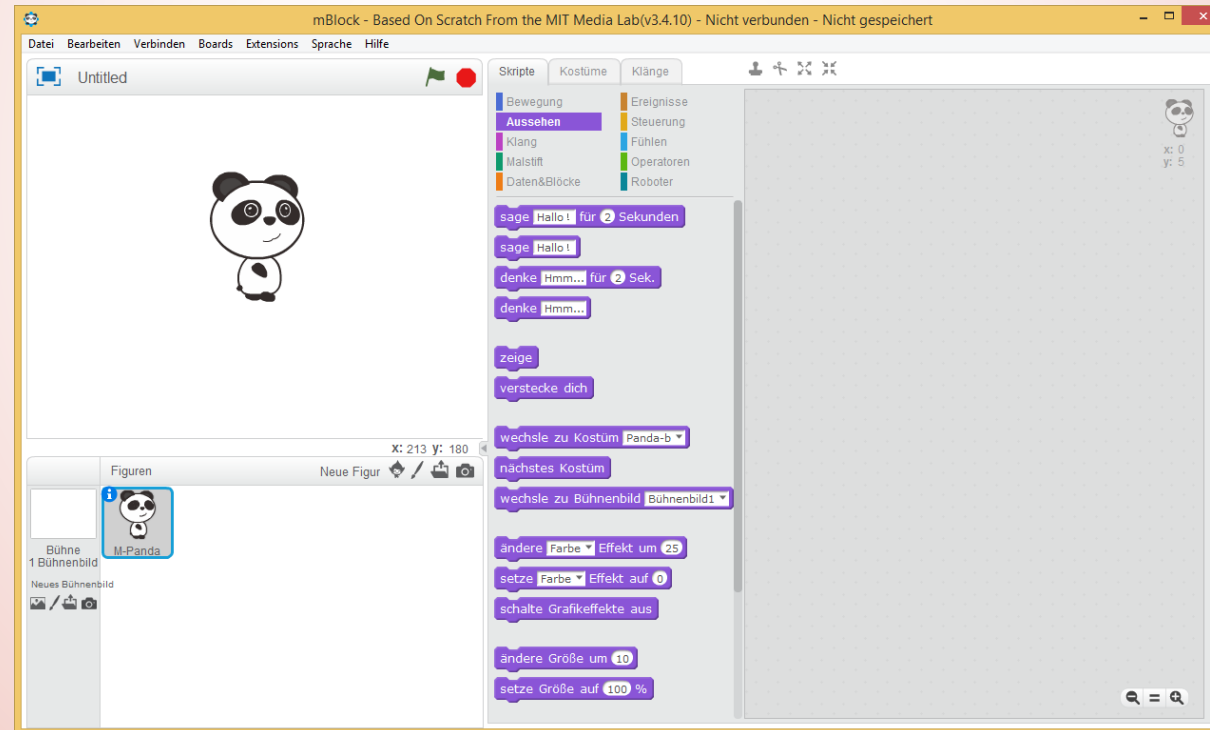


# mBlock: Eine grafische Programmierumgebung (Scratch)



# mBlock: Umschalten auf Arduino Modus

- Über „Bearbeiten“ im Menü gehen, dann „Arduino-Modus auswählen“





# mBlock: Übersicht

The screenshot displays the mBlock software interface, which is based on Scratch. The window title is "mBlock - Based On Scratch From the MIT Media Lab(v3.4.8) - Nicht verbunden - Nicht gespeichert". The interface includes a menu bar (Datei, Bearbeiten, Verbinden, Boards, Extensions, Sprache, Hilfe), a left sidebar with category tabs (Skripte, Bewegung, Aussehen, Klang, Malstift, Daten&Blöcke, Ereignisse, Steuerung, Fühlen, Operatoren, Roboter), and a central workspace. The workspace contains an "Arduino Programm" block, a "wiederhole fortlaufend" loop, and three sub-blocks: "setze light auf lese analogen Pin (A) 0", "sende Text an seriellen Port light", and "warte 1 Sek.". A red rectangle highlights this central workspace, with the text "Programmierbereich" written in red across it. On the right side, there is a code editor showing the corresponding C++ code for the program, including headers, setup, and loop functions. Below the code editor is a terminal window showing the output of the program, including the text "Reading | 100% 0" and "avrduide: verifying ...". At the bottom right, there are buttons for "send encode mode" and "recv encode mode", each with radio buttons for "Binär-Modus" and "Zeichen-Modus", and a "Senden" button.

Arduino Programm

wiederhole fortlaufend

setze light auf lese analogen Pin (A) 0

sende Text an seriellen Port light

warte 1 Sek.

**Programmierbereich**

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7 double light;
8
9 void setup() {
10   Serial.begin(115200);
11   pinMode(A0+0, INPUT);
12 }
13
14 void loop() {
15   light = analogRead(A0+0);
16   Serial.println(light);
17   _delay(1);
18   _loop();
19 }
20
21 void _delay(float seconds) {
22   long endTime = millis() + seconds * 1000;
23   while(millis() < endTime) _loop();
24 }
avrduide: load data flash data from input file C:\Users\ab\AppData\Local\Temp\build1881702473048
avrduide: input file C:\Users\ab\AppData\Local\Temp\build1881702473048
avrduide: reading on-chip flash data:
Reading | ##### | 100% 0
avrduide: verifying ...
avrduide: 5352 bytes of flash verified
avrduide done. Thank you.
```

send encode mode  
☒ Binär-Modus ☐ Zeichen-Modus

recv encode mode  
☒ Binär-Modus ☐ Zeichen-Modus

Senden

# mBlock: Übersicht

The screenshot displays the mBlock software interface, which is based on Scratch. The window title is "mBlock - Based On Scratch From the MIT Media Lab(v3.4.8) - Nicht verbunden - Nicht gespeichert". The interface includes a menu bar (Datei, Bearbeiten, Verbinden, Boards, Extensions, Sprache, Hilfe), a toolbar, and a main workspace. On the left, there is a "Skripte" (Scripts) panel with a red border, containing a "Block-Palette" (Block Palette) with various categories like Bewegung (Movement), Aussehen (Appearance), Klang (Sound), Malstift (Paint), Daten&Blöcke (Data & Blocks), Ereignisse (Events), Steuerung (Control), Fühlen (Sensing), Operatoren (Operators), and Roboter (Robot). The main workspace shows an "Arduino Programm" (Arduino Program) block, a "wiederhole fortlaufend" (Repeat Forever) loop, and a "setze light auf lese analogen Pin (A) 0" (Set light to read analog pin (A) 0) block. The right side of the interface features a "Zurück" (Back) button, an "Upload zum Arduino" (Upload to Arduino) button, and a "Mit der Arduino IDE editieren" (Edit with Arduino IDE) button. Below these buttons is a code editor showing the following C++ code:

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7 double light;
8
9 void setup() {
10   Serial.begin(115200);
11   pinMode(A0+0, INPUT);
12 }
13
14 void loop() {
15   light = analogRead(A0+0);
16   Serial.println(light);
17   _delay(1);
18   _loop();
19 }
20
21 void _delay(float seconds) {
22   long endTime = millis() + seconds * 1000;
23   while(millis() < endTime) _loop();
24 }
```

Below the code editor, there is a terminal window showing the output of the program:

```
avrdude: load data flash data from input file C:\Users\ab\AppData\Local\Temp\build1881702473048
avrdude: input file C:\Users\ab\AppData\Local\Temp\build1881702473048
avrdude: reading on-chip flash data:

Reading | ##### | 100% 0

avrdude: verifying ...
avrdude: 5352 bytes of flash verified

avrdude done. Thank you.
```

At the bottom of the terminal window, there are buttons for "send encode mode" and "recv encode mode", each with radio buttons for "Binär-Modus" (Binary Mode) and "Zeichen-Modus" (Text Mode). A "Senden" (Send) button is also present.

# mBlock: Übersicht

The screenshot displays the mBlock software interface, which is based on Scratch. The main workspace shows a Scratch script with a 'wiederhole fortlaufend' (repeat forever) loop. Inside the loop, the following blocks are present: 'setze light auf lese analogen Pin (A) 0', 'sende Text an seriellen Port light', and 'warte 1 Sek.' (wait 1 second). The right-hand panel, labeled 'Arduino-Bereich' (Arduino Area), contains a code editor and a terminal window. The code editor shows the following C++ code:

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7 double light;
8
9 void setup() {
10   Serial.begin(115200);
11   pinMode(A0+0, INPUT);
12 }
13
14 void loop() {
15   light = analogRead(A0+0);
16   Serial.println(light);
17   _delay(1);
18   _loop();
19 }
20
21 void _delay(float seconds) {
22   long endTime = millis() + seconds * 1000;
23   while(millis() < endTime) _loop();
24 }
```

The terminal window shows the output of the program, including the text 'Reading | ##### | 100% 0' and 'avrduide: verifying ...', indicating that the code has been successfully uploaded and executed.

# Übung

## Das erste Programm



- Startet mBlock auf eurem Rechner
- Wechselt in den Arduino Modus (Bearbeiten -> Arduino Modus)
- Schließt das LCD an einen „I2C“ Steckplatz an
- Verbindet den Arduino und das Notebook über das Kabel
- Verbindet mBlock und den Arduino über (im Menü) „Verbinden“ -> „serieller Port“ -> „Com2“. Die Zahl hinter „Com“ kann unterschiedlich sein, das ist egal
- Erstellt ein erstes Arduino Programm
- Ladet es in den Arduino hoch
- „Arduino Programm“ findet ihr unter „Roboter“
- LCD Befehle auch





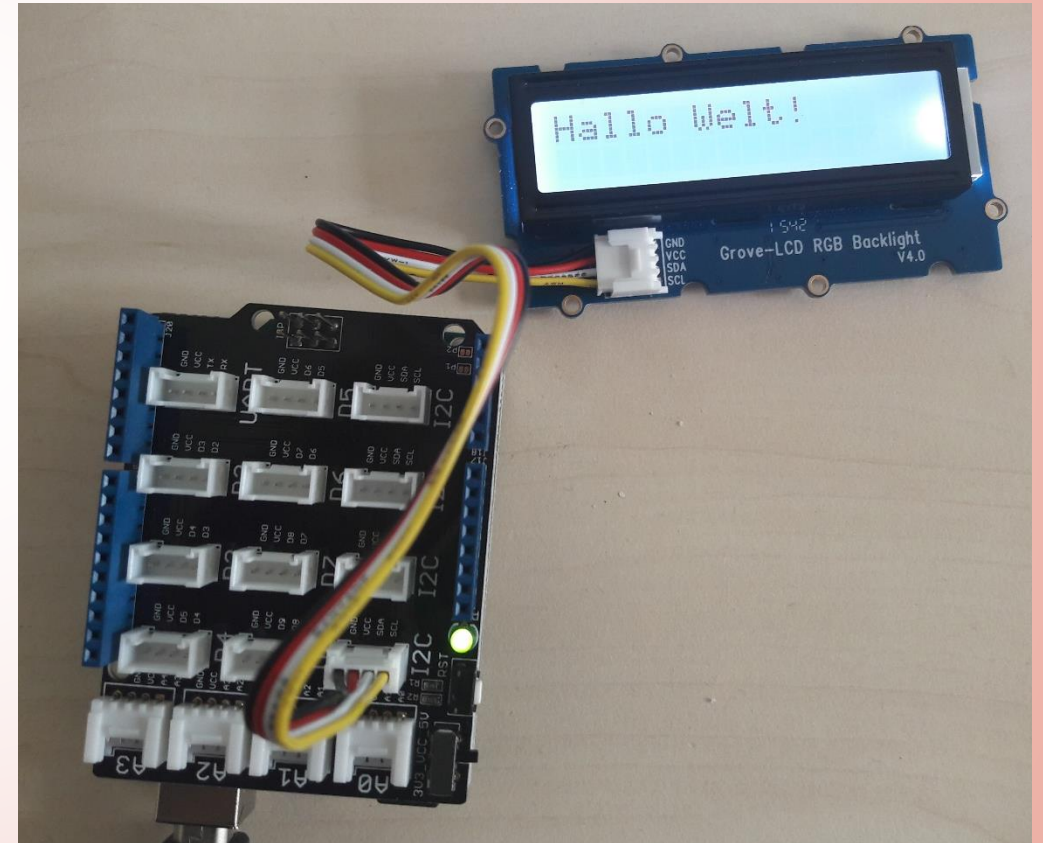
# Übung

## Das erste Programm

- Das Resultat sollte ungefähr so aussehen

Arduino Programm

LCD zeige text `Hallo Welt!`



# Aktor RGB LCD (I2C)



# Übung RGB LCD

- Recherchiert:
  - Wofür steht LCD?
  - Wofür steht RGB?
- Ändert euer Programm:
  - Ändert den Text, den ihr ausgeben
  - Ändert die Hintergrundfarbe auf Rot, Grün, Blau
  - Wie viel Text passt auf das Display?
  - Verändert die Position des Textes mit „LCD Pos“

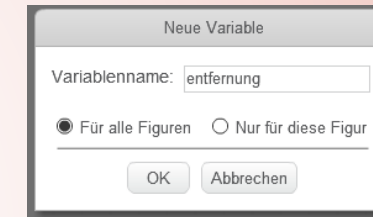
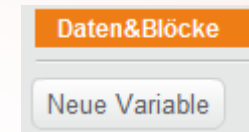
# Einführung: Variablen

- Eine Variable ist ein Platz, um Werte zu speichern
- Ähnlich wie ein Karton, in den ich etwas hinein tun kann
- Dieser Karton hat einen eindeutigen Namen, also nur er heißt so
- Analog zu Variablen in der Mathematik ( $x, \dots$ )
- Gebt den Variablen „sprechende“ Namen, damit ihr nächste Woche noch wisst, was die Variable enthält („abstand“ statt „a“)
- Bei mBlock können Variablen **NUR** Zahlen enthalten (keinen Text)



# Einführung: Variablen anlegen

- Anlegen einer Variablen im Punkt Daten&Blöcke
- Dabei muss der Variable ein Name gegeben werden (Bitte nicht nur a oder x)
- Danach erscheinen neue Befehle unter Daten&Blöcke für diese Variable



# Einführung: Variablen anlegen

- Variablen können gelesen und beschrieben werden
- Zum setzen einer Variable Nutzt das „setze .... auf x“
- Variablen können Zahlen oder Texte beinhalten
- Zum Lesen zieht die Variable in einen andere Operation rein

The image shows a block editor interface with two panels. The left panel, titled 'Daten&Blöcke', contains a 'Neue Variable' button and a list of variable operations: 'Wert' (checked), 'setze Wert auf 0', 'ändere Wert um 1', 'zeige Variable Wert', and 'verstecke Variable Wert'. The right panel shows a sequence of blocks: 'buzzer D4 play tone 1519 for duration', 'temperature sensor A0', 'sound sensor A0', 'rotary angle sensor A0', 'Piezo vibration sensor A0', 'button D4 pressed', 'touch sensor D4 pressed', '3-axis digital accelerometer x', 'Seed Grove smile-project', 'LCD zeige text Hallo Welt!', 'LCD löschen', 'LCD aus', 'LCD an', 'LCD Pos Zeile: 0 Spalte: 0', 'LCD farbe blau', and 'Chain LED LED0 r: 0 g: 0 b: 0'. Red arrows point from the 'setze Wert auf 0' and 'zeige Variable Wert' blocks in the left panel to the 'setze Wert auf 0' and 'LCD zeige text Hallo Welt!' blocks in the right panel, illustrating the variable's use.

# Übung Variablen 1

- ▶ Ändert euer Programm:
  - ▶ Legt eine Variable an und weist dieser einen Wert zu
  - ▶ Gebt den Wert der Variable auf dem Display aus
  - ▶ Benutzt „verbinden“ um zusätzlich noch einen Text auszugeben

# Lösung Übung Variablen 1

- Ändert euer Programm:
  - Legt eine Variable an und weist dieser einen Wert zu
  - Gebt den Wert der Variable auf dem Display aus
  - Benutzt „verbinden“ um zusätzlich noch einen Text auszugeben





# Übung Variablen 2

- ▶ Ändert euer Programm:
  - ▶ Wartet nach der Ausgabe von „Wert“ für eine Sekunde
  - ▶ Erhöht dann den Wert der Variable „Wert“ um eins
  - ▶ Gebt die geänderte Variable aus
  - ▶ Wiederholt dieses drei mal

# Lösung Übung Variablen 2

- Ändert euer Programm:
  - Wartet nach der Ausgabe von „Wert“ für eine Sekunde
  - Erhöht dann den Wert der Variable „Wert“ um eins
  - Gebt die geänderte Variable aus
  - Wiederholt dieses drei mal

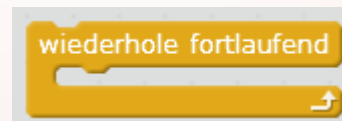
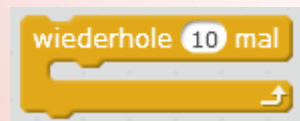


# Übung Variablen

- Wofür braucht Ihr Variablen bei der Umsetzung eurer Pflanze?

# Einführung Schleifen/Wiederholungen

- Wäre es nicht praktisch, die Programmteile immer wieder untereinander schreiben zu müssen?
- Dafür gibt es „Schleifen“ unter dem Punkt „Steuerung“
- Die Befehle Teil in der Schleife werden dann wiederholt (eine bestimmte Anzahl oder fortlaufend, die sogenannte „Endlosschleife“)



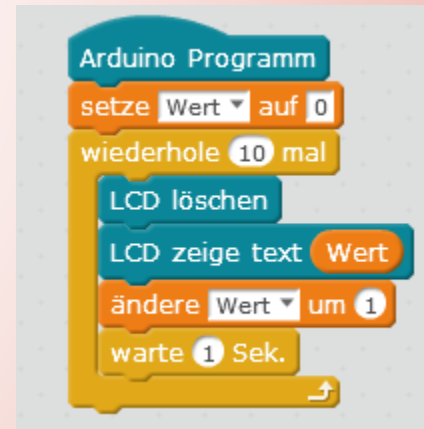
# Übung Schleifen/Wiederholungen

- Ändert Euer Programm so ab, dass ihr anstatt der Wiederholung der Befehle eine Schleife verwendet. Ob 10 Wiederholungen oder fortlaufend ist dabei egal.



# Lösung Übung Schleifen/Wiederholungen

- ▶ Ändert Euer Programm so ab, dass ihr anstatt der Wiederholung eine Schleife verwendet. Ob 10mal oder fortlaufend ist dabei egal



# Übung Schleifen

- ▶ Wofür braucht Ihr Schleifen bei der Umsetzung eurer Pflanze?

# Ende der Wiederholung

# Sensoren

## Ultraschall-Abstand(Digital)



# Übung

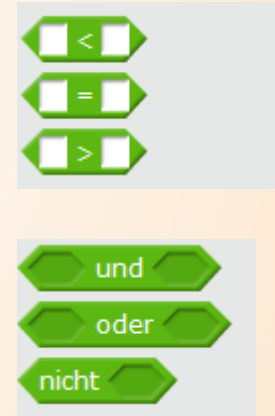
## Ultraschall-Abstand

- Steckt den Abstandssensor an einen digitalen Steckplatz
- Versucht ein Programm zu schreiben um die Daten auszulesen
  - Tipp: schreibt die Sensorwerte in eine Variable und gibt diese auf dem LCD aus
- Testet und Dokumentiert:
  - Wie schnell reagiert der Abstandssensor?
  - Welche Entfernungen erkennt der Sensor?



# Einführung Bedingungen

- ▶ Bedingungen dienen zum Steuern des Programmflusses. Sie sind so etwas wie „Wenn dies zutrifft tue dies, sonst das“.
- ▶ Bedingungen findet ihr unter „Operatoren“
- ▶ Es gibt einfache Bedingungen (kleiner, größer, gleich)
- ▶ Und es gibt logische Bedingungen (Bedingung 1 und/oder/nicht Bedingung2)



# Einführung Bedingungen 2

- Unter „Steuerung“ findet ihr entsprechende Elemente



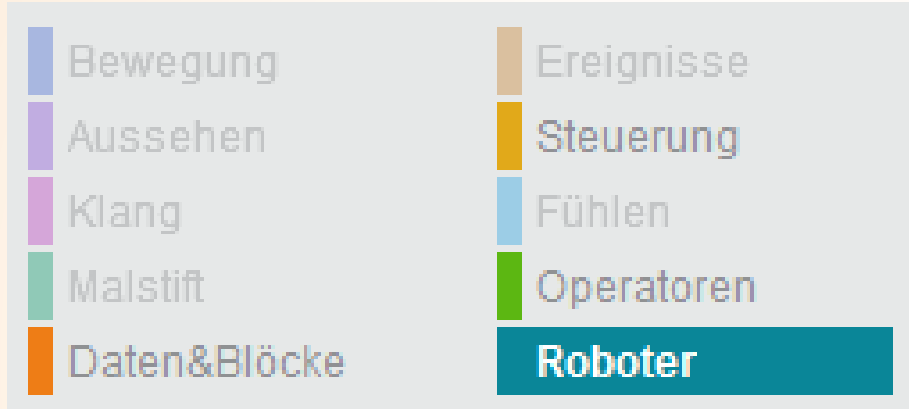
# Übung Bedingungen

- Ändert Euer Programm so ab, dass ihr mit einer Bedingung überprüft, ob der Wert aus der Variable eures Abstandssensors größer als z.B. 20 cm ist. Dann ändert die Hintergrundfarbe des LCD Displays auf blau. Wenn der Abstand kleiner ist, wieder auf eine andere Farbe.

# Übung Bedingungen

- Wofür braucht Ihr Bedingungen bei der Umsetzung eurer Pflanze?

# Einführung: Farben



Für uns Wichtig:

- **Roboter** für die Ansteuerung des Arduinos
- **Daten&Blöcke** zum verwenden von Variablen
- **Operationen** zum Rechnen und Vergleichen
- **Steuerung** zum Kontrollstrukturen zu erstellen



# Einführung: Formen

Die Formen geben Hinweise über die Möglichkeiten der Verwendung des Elements

- Runde Elemente passen in die weißen Flächen:

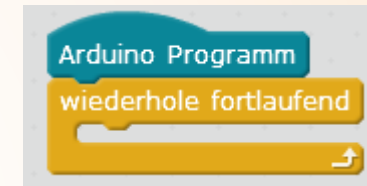


- Eckige Elemente kann man miteinander verbinden



# Einführung: Programm

- Ein Arduino Programm in mBlock hat immer die gleiche Struktur:
- Ein Programm und eine endlos-Schleife
- Alles weitere wird in diese Schleife gepackt
- Wenn Ihr es ausprobieren wollt, müsst ihr auf „Upload zum Arduino“ drücken
- „Arduiono Programm“ findet ihr unter „Roboter“
- Schleifen und warten unter „Steuerung“



# Sensoren Temperatur (Analog)



# Übung Temperatur

- ▶ Steckt den Temperatursensor an den Steckplatz A1 an
- ▶ Versucht ein Programm zu schreiben um die Daten auszulesen
- ▶ Testet und dokumentiert:
  - ▶ Wie schnell reagiert der Sensor?
  - ▶ Wie genau ist der Sensor?
  - ▶ Müsst Ihr Anpassungen der Werte vornehmen? (Eichen mit Thermometer)

# Aktoren


## Chainable LED Digital 7 und 8 (beide!)





# Übung

## Chainable LED

- Steckt die LED an Digital Port D7!
- (D8 ist zwar frei, kann aber nicht mehr verwendet werden!)
- Die Verbindung muss dabei vom Steckboard zum „in“ Stecker einer LED gehen.
  - Wenn Ihr mehrere LEDs anschließen wollt, hintereinander immer von „out“ zu „in“
  - Die Steuerung erfolgt über den Chain LED Befehl 
  - Dabei müssen die Werte für rot (r), grün (g), und blaue (b) als ganze Zahlen zwischen 0 und 255 eingetragen werden.
- Testet und dokumentiert:
  - Wie müssen die rgb-Werte für verschiedene Farben sein?
  - Findet ihr ein „Farbrad“ dafür im Internet? Und wenn ja, wo?
  - Könnt Ihr auch 2 LEDs unterschiedlich ansteuern?

# Sensoren

## Lichtsensor(Analog)

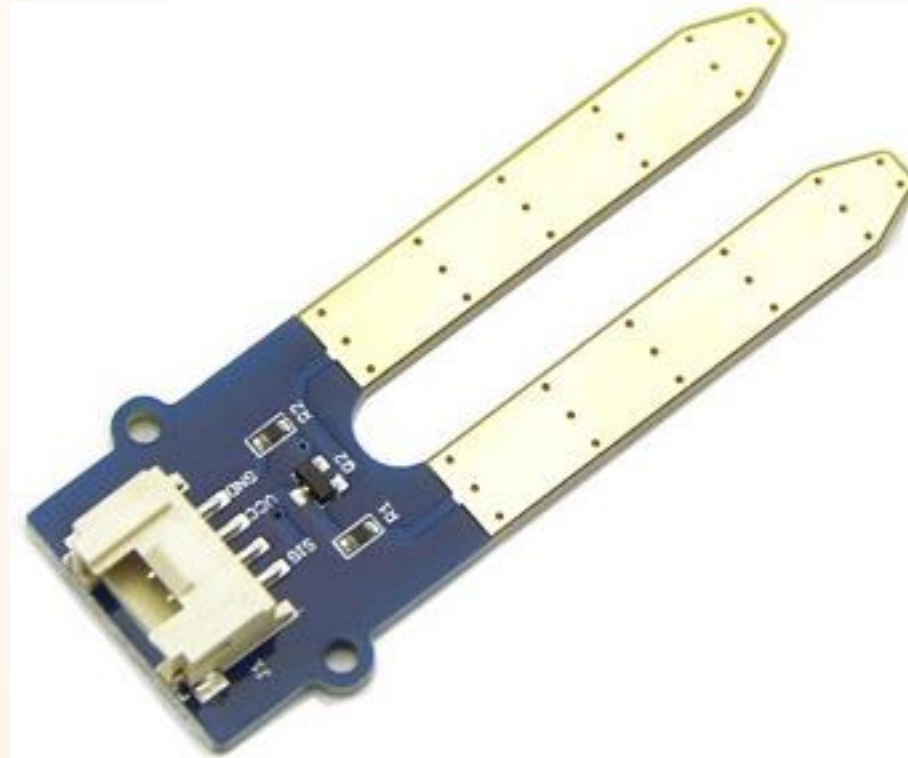


# Übung Lichtsensord

- Steckt den Lichtsensor an den Steckplatz A0 an
- Versucht ein Programm zu schreiben um die Daten auszulesen
  - Tipp: schreibt die Sensorwerte auf das LCD Display
- Testet und Dokumentiert:
  - In welcher Maßeinheit wird Licht allgemein gemessen? (Internet Recherche)
  - Wie schnell reagiert der Lichtsensor?
  - Was für Werte bekommt Ihr in der Sonne, was für welche im Schatten und was für welche wenn Ihr den Sensor abdeckt?
  - Vergleicht die Werte aus eurem Sensor mit einem externen Lichtsensor

# Sensoren

## Feuchtigkeit (Analog)

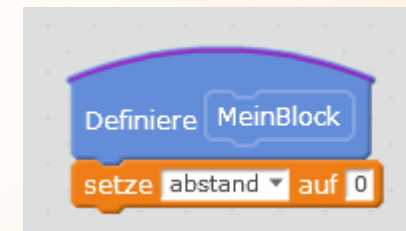
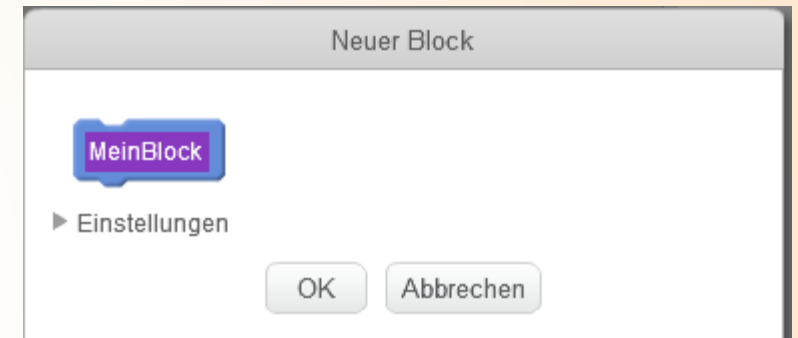


# Übung Feuchtigkeit

- ▶ Steckt den Temperatursensor an den Steckplatz A1 an
- ▶ Versucht ein Programm zu schreiben um die Daten auszulesen
- ▶ Testet und dokumentiert:
  - ▶ Wie schnell reagiert der Sensor?
  - ▶ Wie genau ist der Sensor?

# Einführung Blöcke

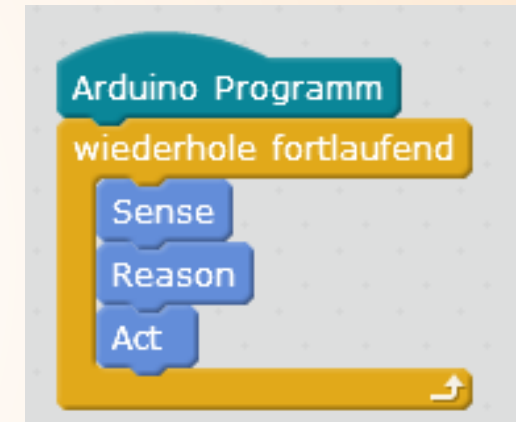
- Blöcke dienen zum Strukturieren des Programmes
- Gebt ihnen sinnvolle Namen
- Blöcke dürfen **NICHT** den gleichen Namen wie eine Variable haben
- Innerhalb eines Blockes habt ihr Zugriff auf alle Variablen des gesamten Programmes





# Einführung Sense-Reason-Act Modell

- Sense-Reason-Act ist ein sinnvoller Ablauf für die Programmierung eurer Pflanze
- Das Programm sollte so aufgebaut werden
- Die Werte von einem Programmteil zum nächsten speichert ihr in Variablen



# Kreativ-Übung

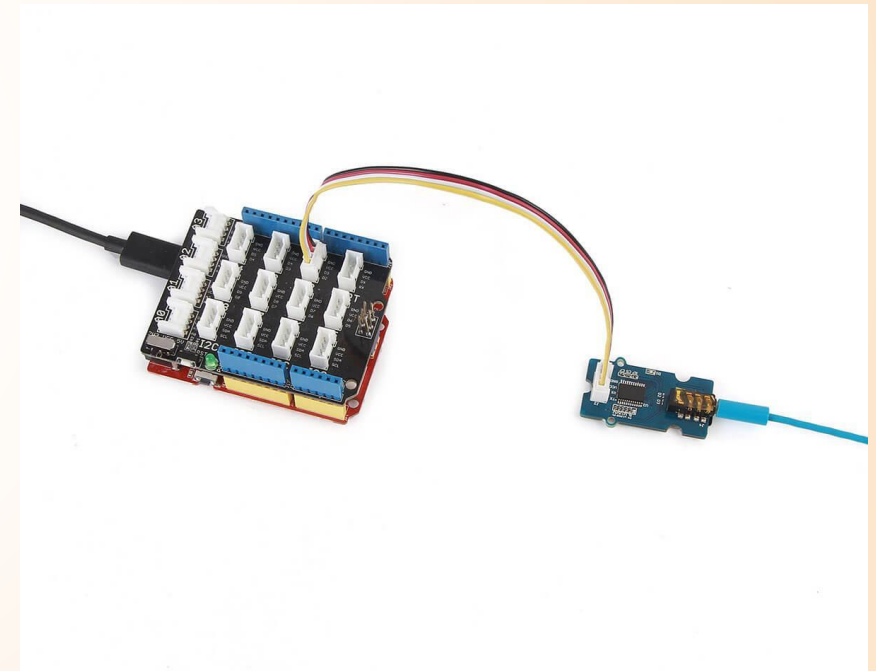
- ▶ Was von den Ideen von der ersten Kreativrunde meint Ihr sind umsetzbar?
- ▶ Was für neue Ideen sind Euch gekommen?

# Aufgaben für die Exkursion und Recherche

- Macht Fotos des Ausflugs, der Pflanzen und vom botanischen Garten
- Macht Euch Stichworte über die Exkursion
- Welche Pflanzen (Name und Art) habt Ihr bekommen?
- Was wisst Ihr über die Bedürfnisse (Licht, Wasser, Boden) der Pflanzen?
- Was wisst Ihr nicht und müsst es noch recherchieren?

# Aktor MP3

- Das MP3-Modul an D2 verbinden
- Sounds als mp3 Speichern
  - Name: 01.mp3 , 02.mp3, 03.mp3, ...
- Ordner auf der SD Karte „MP3“ erstellen
- Dateien in den Ordner kopieren
- Abspielen über MBlock „play mp3 (1)“
- Sounds findet ihr u.a. auf [www.soundbible.com](http://www.soundbible.com)
- Youtube songs umwandeln:  
<https://www.onlinevideoconverter.com/de>



# Feedback Tag 2

➤ Was denkt Ihr?