Chatter Programmierung - Erste Schritte

Einführung

Installation

Willkommen zum Chatter-Programmiertutorial!

Vielen Dank für deine Unterstützung von CircuitMess und willkommen zum Chatter-Programmiertutorial.

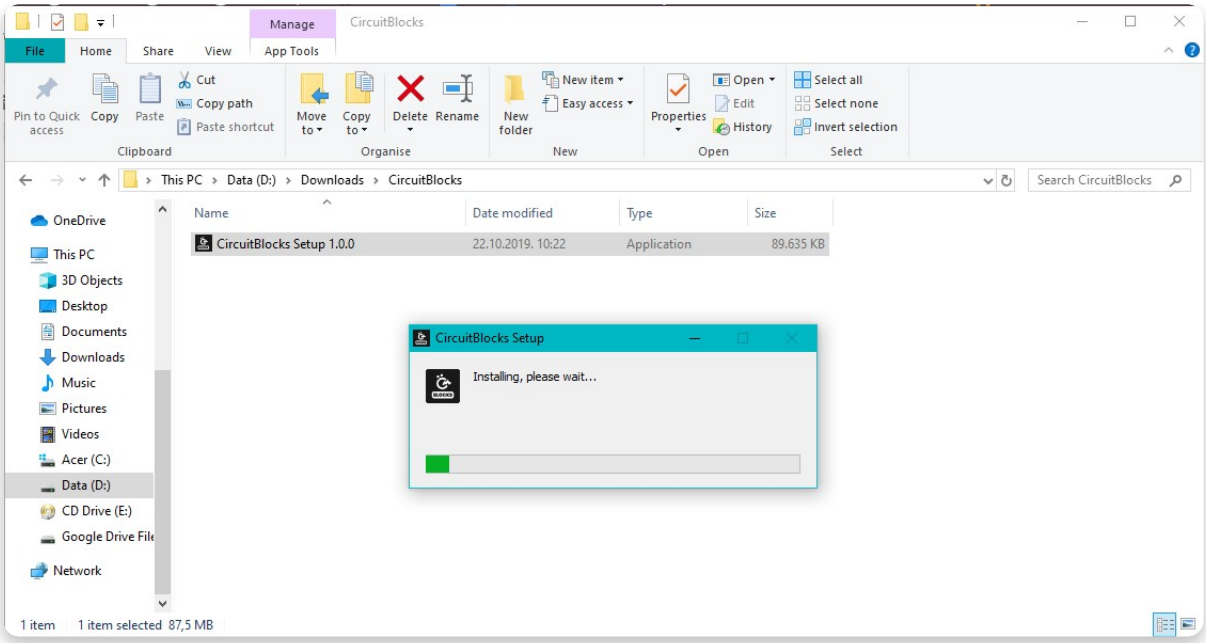
Wir werden **CircuitBlocks** für die Programmierung deiner neu zusammengebauten, verschlüsselnden, drahtlosen Kommunikatoren verwenden.

CircuitBlocks ist eine von uns entwickelte, maßgeschneiderte Programmier-App. Du wirst deinen Chatter in der grafischen, blockbasierten Programmieroberfläche von CircuitBlocks programmieren, die dir bei deinen ersten Schritten in der Welt der Datenverarbeitung helfen wird.

CircuitBlocks läuft derzeit auf Windows-, Linux- und Mac OS-Computern.

Wenn du einen Windows-Computer hast

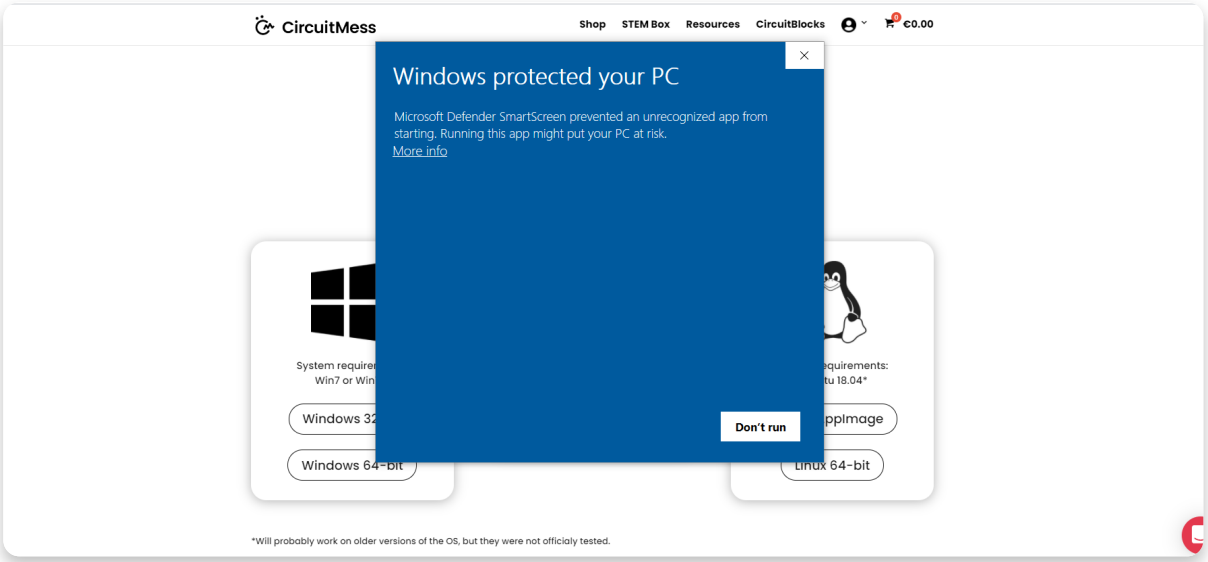
1. **Gehe auf die CircuitBlocks-Download-Seite**
2. **Lade die neueste Version für Windows herunter** - Prüfe dabei, ob du eine 32- oder 64-Version hast. Öffne dazu die "Einstellungen" App, klicke auf die Option "System" und suche den Abschnitt "Info". Hier siehst du den Systemtyp.
3. Starte die heruntergeladene Datei mit dem Namen "CircuitBlocks" durch einen Doppelklick.
4. CircuitBlocks wird automatisch installiert und eine neue Desktop-Verknüpfung wird erstellt.



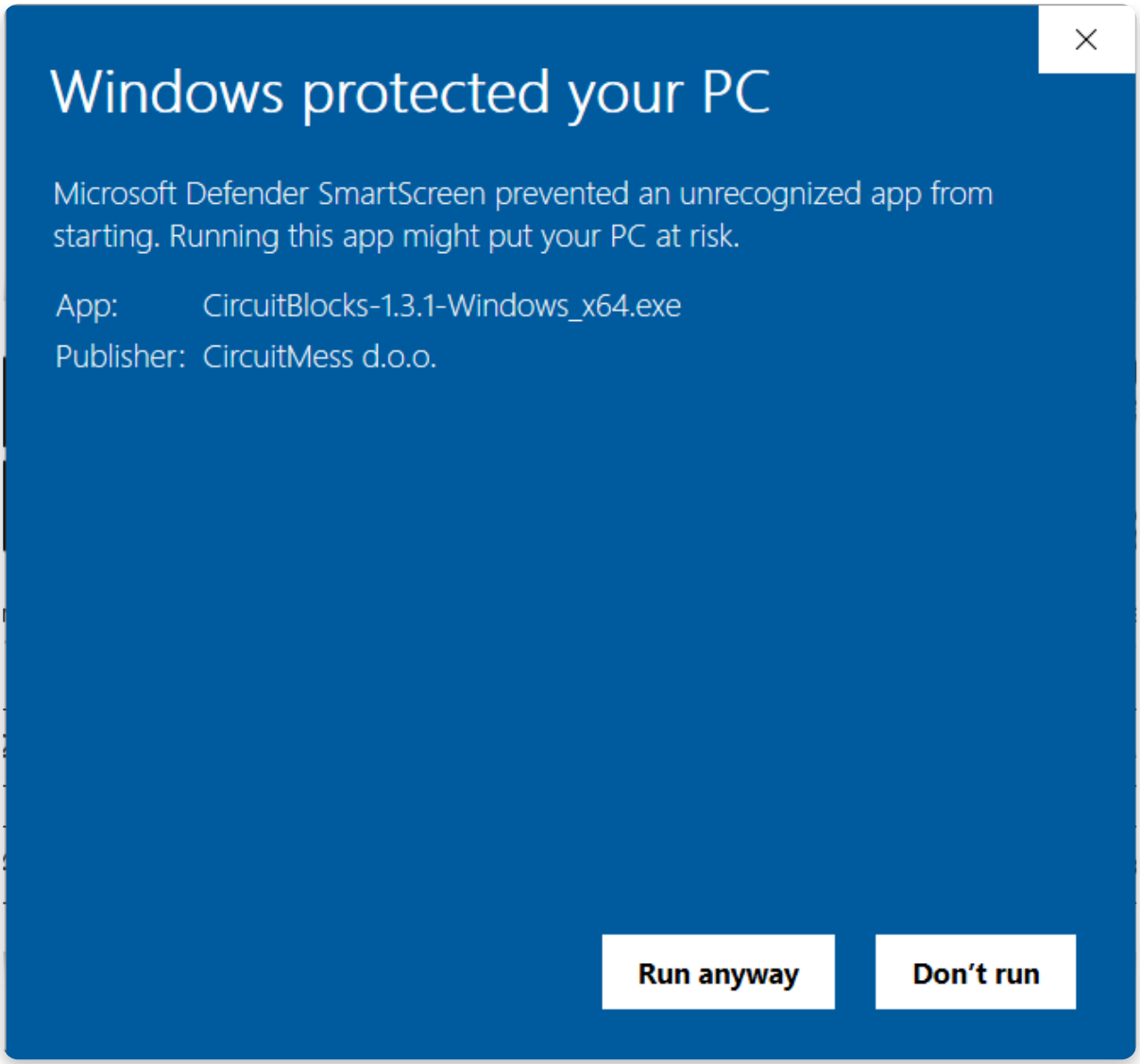
Dein PC ist nicht gefährdet!

Es besteht die Möglichkeit, dass eine Meldung erscheint, die besagt, dass dein PC gefährdet ist, wenn du versuchst CircuitBlocks zu installieren. Mach' dir keine Sorgen! Diese Meldung wird manchmal angezeigt - unabhängig davon, ob CircuitBlocks sicher ist.

Wie du mit dieser Meldung umgehen kannst, erfährst du in den folgenden Anweisungen.



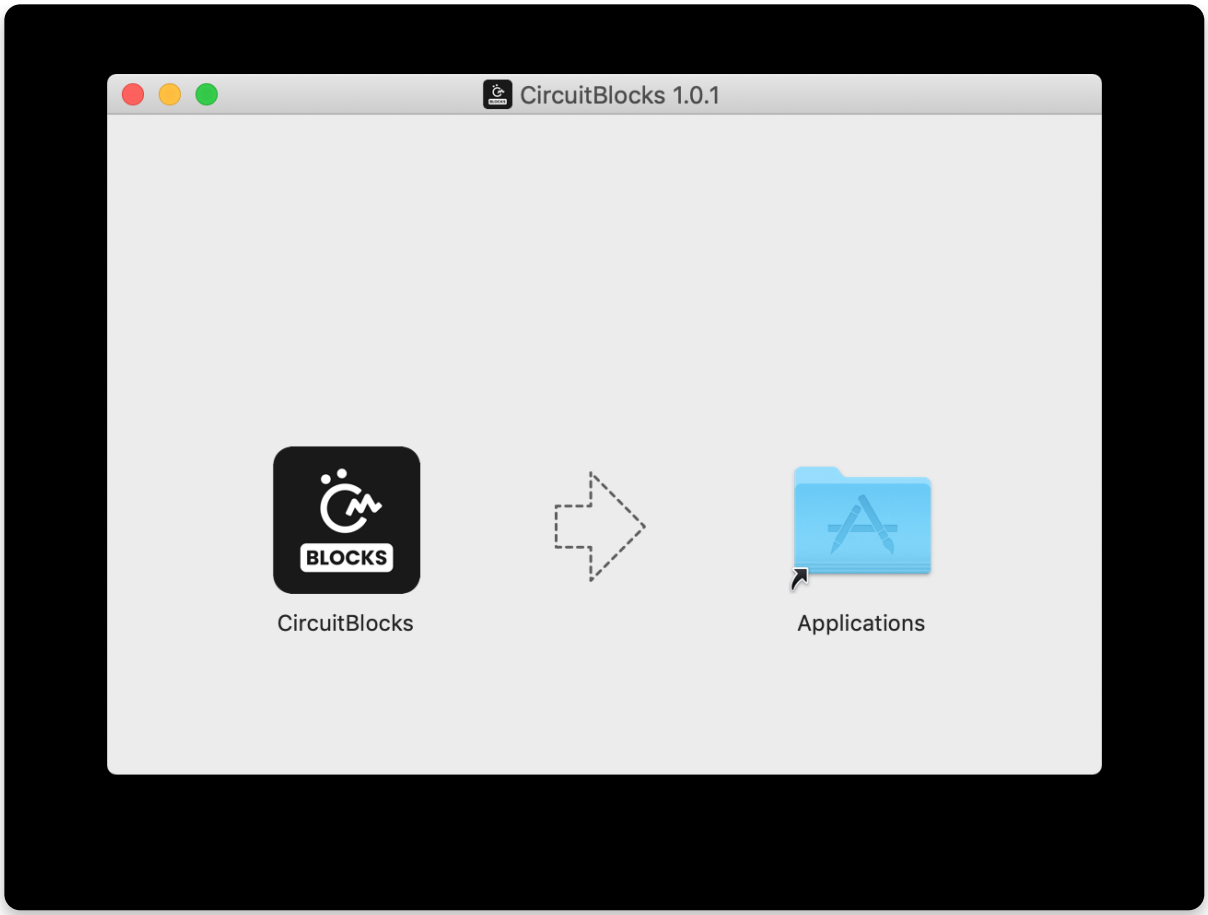
Diese Meldung erhältst du möglicherweise bei der Installation von CircuitBlocks auf deinem PC. Windows meldet eine Bedrohung, obwohl das Programm sicher heruntergeladen und ausgeführt werden kann. Bitte fahre mit der Installation fort, indem Du auf die Option "Mehr Informationen" (bzw. "More info" in englisch) klickst.



Nachdem du auf die Option "Weitere Informationen" geklickt hast, sollte am unteren Rand des Fensters die Option "Trotzdem ausführen" (in englisch: "Run anyway") erscheinen. Klicke auf die Option "Trotzdem ausführen" und fahre mit der Installation fort.

Wenn du einen Mac Computer hast

1. **Gehe auf die CircuitBlocks-Download-Seite**
2. **Lade die neueste Version für MacOs herunter** - Eine Datei mit dem Namen "CircuitBlocks-1.0.1-Mac.dmg" oder ähnlichem sollte heruntergeladen werden.
3. Verschiebe die Datei in den Ordner "Programme".
4. CircuitBlocks wird automatisch installiert.



Wenn du einen Linux Computer hast

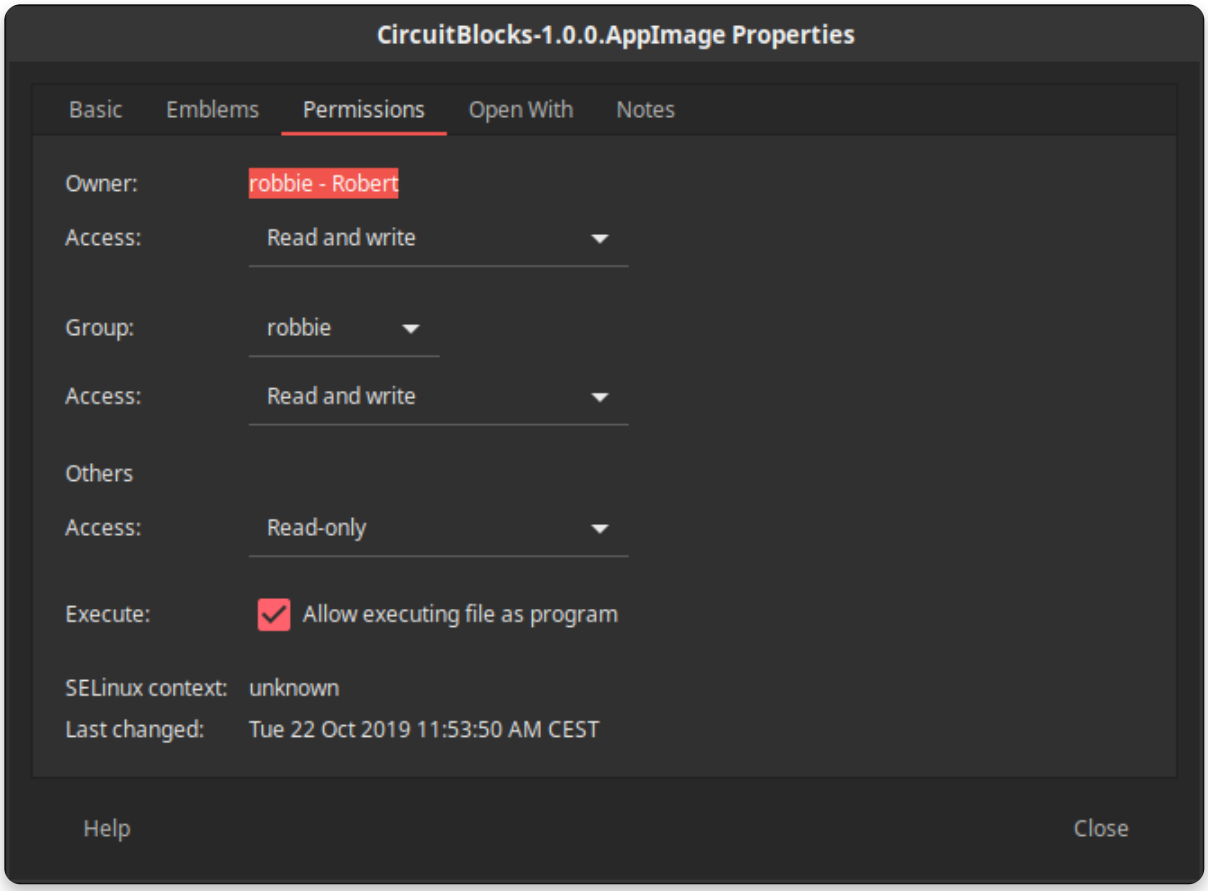
Es gibt zwei Möglichkeiten, CircuitBlocks unter Linux zu installieren.
Linux 64-bit:

1. **Gehe auf die CircuitBlocks-Download-Seite**
2. **K**licke auf den Download-Button "**Linux 64-bit**"
3. Unter Ubuntu startest du die Installation einfach per Doppelklick auf die Datei. Bei anderen Linux-Distributionen öffne das Terminal und führe folgenden Befehl aus:
   `sudo dpkg -i <Pfad zur heruntergeladenen Datei .deb>`
4. CircuitBlocks wird automatisch installiert und ein Desktop-Eintrag wird erstellt.

Eigenständiges AppImage:

1. **Gehe auf die CircuitBlocks-Download-Seite**
2. **Klicke auf den Download-Button "Linux AppImage"**
3. **Klicke mit der rechten Maustaste auf die Datei und wähle "Eigenschaften"**

4. Gehen auf die Seite "Berechtigungen" (bzw. in englisch: "Permissions") und setze ein Häkchen bei "Ausführen der Datei als Programm zulassen" (bzw. "Allow executing file as program").
5. Doppelklicke die Datei und die Installation wird automatisch abgeschlossen.



Solltest du Probleme mit der Installation haben, wende dich bitte per E-Mail an contact@circuitmess.com und sende uns einen Screenshot des Problems sowie alle relevanten Informationen.

Die Grundlagen

Benutzeroberfläche

Wenn du CircuitBlocks öffnest, siehst du ein Fenster, das wie oben gezeigt aussieht.

Es ist ziemlich einfach - Du kannst ein **neues Projekt (wir nennen Projekte auf englisch auch "Sketches"**) beginnen, indem Sie auf die Schaltfläche "Neues Projekt" (englisch "New project") klickst.

Gespeicherte Projekte werden direkt neben dieser Schaltfläche angezeigt und du kannst jederzeit darauf zugreifen.

Wenn du auf ein Problem mit CircuitBlocks stößt, klicke bitte auf den Link "**Fehlerbericht senden**" (auf englisch "Send error report") am unteren Rand des Hauptbildschirms. Hierbei erhältst du dann eine Fehlermeldungsnummer. Bitte kontaktiere uns über contact@circuitmess.com und gib diese Fehlermeldungsnummer an, damit wir dir weiterhelfen können.

Erstellen eines neuen Projekts (Sketch)

Klicke auf die große Schaltfläche "Neues Projekt" (englisch "New project").

Nun bekommst du die Möglichkeit, das Gerät und den Projekttyp auszuwählen.

Für das Gerät wähle bitte: **Chatter**.

Für den Typ des Sketches wähle bitte: **Block**.

Drücke die Schaltfläche "Erstellen" (englisch "**Create**").

Es erscheint ein Bildschirm, der wie folgt aussieht:

Am oberen Rand des Bildschirms befindet sich eine **Symbolleiste** mit einigen Schaltflächen.

Die **Blockauswahlleiste** befindet sich ganz links - Du kannst Blöcke von dort nehmen und sie in den "Zeichenbereich" in der Mitte des Bildschirms ziehen.

In der Mitte des Bildschirms wirst du deinen Programmcode mit bunten Blöcken "zeichnen".

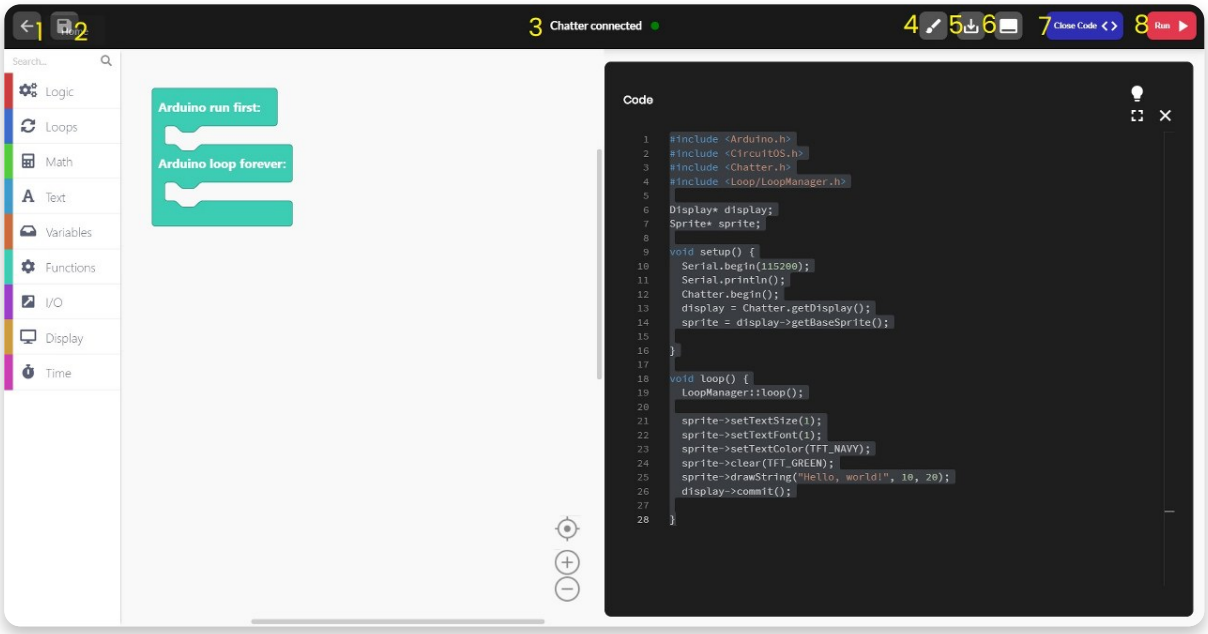
Auf der rechten Seite des Bildschirms siehst du, wie der in der **Programmiersprache C++** geschriebene Code auf magische Weise von selbst erscheint, wenn du die bunten Blöcke in die Mitte ziehst.

C++ ist eine der beliebtesten Programmiersprachen, aber sie ist ziemlich komplex, wenn du noch nie programmiert hast.

Deshalb haben wir CircuitBlocks entwickelt - hier kannst du bunte Blöcke, die Teile des Codes darstellen, mit der Maus verschieben und ablegen und sehen, wie dein Programm in C++ aussehen würde. Wenn du erfahren genug bist, kannst du direkt

zur textuellen Programmierung in C++ wechseln, ohne dass du bunte Blöcke benötigst.

Symbolleiste



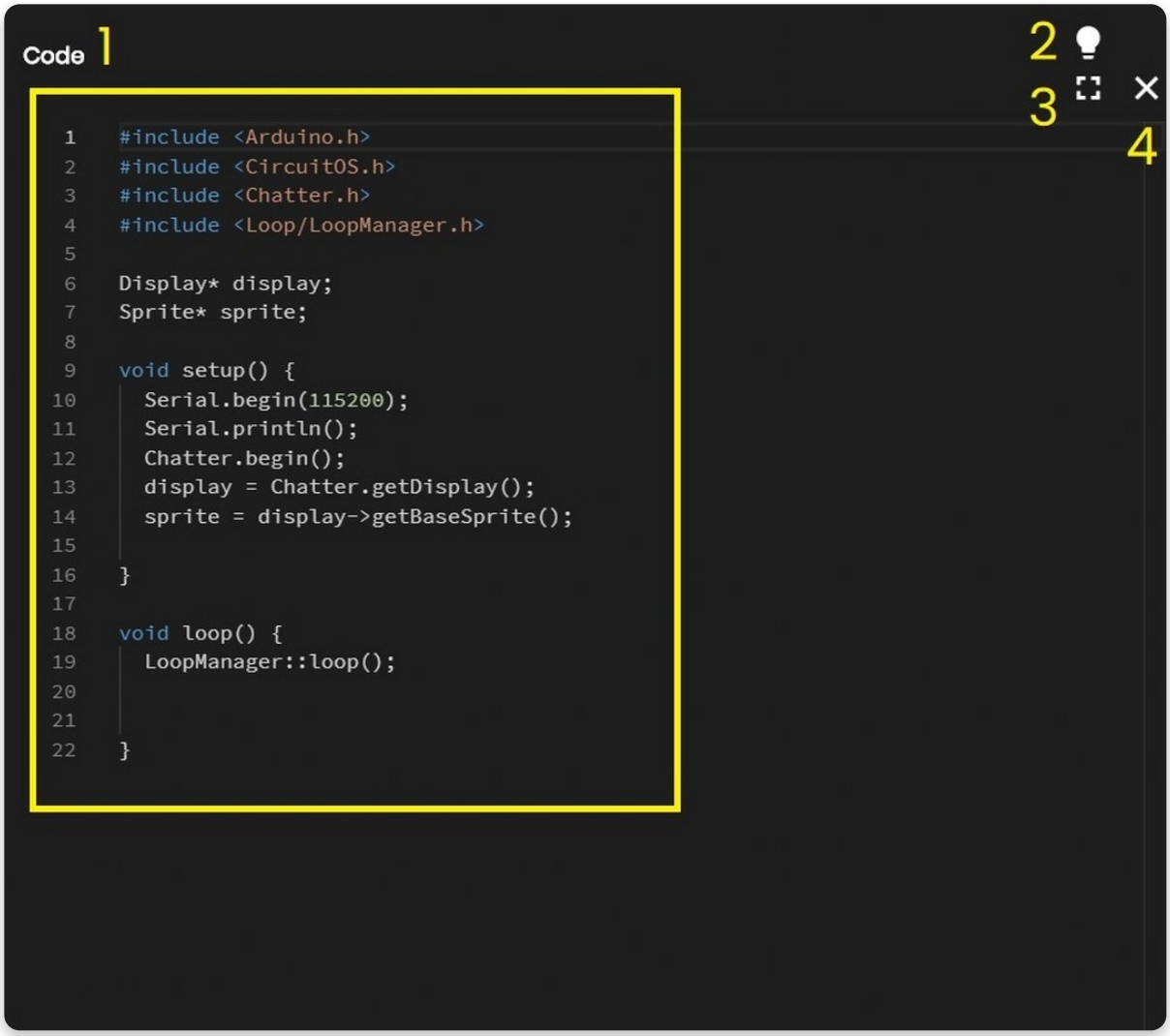
Im Folgenden wird kurz erklärt, was die Schaltflächen in der Symbolleiste des Fensters bewirken:

1. **Back to the main menu/Zurück zum Hauptmenü** – bringt dich zum Startbildschirm zurück, ohne zu speichern.
2. **Save/Save As/Speichern unter** – speichert dein Projekt. Achte darauf, diese Schaltfläche von Zeit zu Zeit und vor dem Schließen von CircuitBlocks zu drücken.
3. **Chatter connection indicator/Chatter-Verbindungsanzeige** - Der rote Punkt wird grün, wenn dein Chatter über ein USB-Kabel mit deinem Computer verbunden ist.
4. **Sprite editor** - zum Zeichnen von Bildern, die du auf deinem Chatter verwenden möchtest.
5. **Export to binary/Exportieren in Binärform** - speichert eine Binärdatei deines Programmcodes auf deinem Computer. Dies ist eine fortgeschrittene Funktion, die du vorerst nicht brauchen wirst.
6. **Serial monitor/Serieller Monitor** - Diese Schaltfläche öffnet ein Fenster, das wir den "Seriellen Monitor" nennen. "Seriell" ist ein Spitzname für eine Art der Kommunikation, die zwischen Chatter und deinem Computer stattfindet. In

diesem Fenster kannst du später die Nachrichten sehen, die von Chatter an deinen Computer über den USB-Anschluss gesendet werden.

7. **Close code/Code schließen** - Mit dieser Schaltfläche kannst du das Code-Fenster auf der rechten Seite des Bildschirms schließen oder wieder öffnen. Dies ist nützlich, wenn du mehr Platz auf dem Bildschirm benötigst, um deine farbigen Blöcke zu sehen.
8. **Run/Ausführen** - Diese Schaltfläche übersetzt den in CircuitBlocks erstellten Code in Maschinencode, den Chatter versteht (beep boop beep boop 1011100101) und sendet den Code über den USB-Anschluss an deinen Chatter.

Code-Fenster



Das so genannte "Code-Fenster" besteht aus den folgenden Teilen:

1. **Main code screen/Haupt-Code-Bildschirm** - in C++ geschriebener (Programm-)Code erscheint hier, wenn du bunte Blöcke auf der linken Seite des Bildschirms einfügst. Du wirst sehen, dass einige Teile des Codes in

lustigen Farben eingefärbt sind. Programmierer nennen dies Syntaxhervorhebung. Im Grunde werden verschiedene Kategorien von Codebefehlen unterschiedlich eingefärbt, damit Programmierer den Code leichter verstehen können.

2. **Light/dark theme switch/Umschalter für helles/dunkles Farbschema** - mit dieser Schaltfläche kannst du die Hintergrund- und Textfarbe des Codefensters umschalten.
3. **Expand/Erweitern** - dehnt das Codefenster auf den gesamten Bildschirm aus. Drücke die Taste erneut, um es wieder auf den halben Bildschirm zu verkleinern.
4. **Close/Schließen** – schließt das Code-Fenster, dieselbe Funktion wie die Schaltfläche "Code schließen" in der Symbolleiste.

Zeichenbereich

Auf dem Zeichenbereich geschieht die Magie.

Er besteht aus den folgenden Teilen:

1. **Search bar/Suchleiste** – gib hier den Namen einer Komponente (in englisch) ein, nach der du suchst.
2. **Component selector/Komponentenauswahl** – die Blöcke sind hier in verschiedene Kategorien unterteilt. Jeder Kategorie ist eine bestimmte Farbe zugewiesen.
3. **Drawing area/Zeichenfläche** – ziehe die Blöcke aus der Komponentenauswahl und lege sie in der Zeichenfläche ab. Auf diese Weise wird der Code erstellt. Kinderleicht!
4. **Center tool/Zentrierwerkzeug** - wenn du dich beim Scrollen durch die Zeichenfläche verirrst, drücke diese Taste. Dadurch wird die Anzeige auf die Blöcke zentriert, die du auf der Zeichenfläche abgelegt hast.
5. **Zoom buttons/Zoom-Tasten** - zum Vergrößern und Verkleinern der Zeichenfläche.

Arten von Blöcken

In CircuitBlocks gibt es insgesamt **neun** Blocktypen. Jeder von ihnen wird durch seine Farbe dargestellt. Jeder Block wird in Code übersetzt, der dann kompiliert und auf deinen Chatter hochgeladen wird, genau wie auf jeder Arduino-basierten Plattform.

Jeden Blocktyp kannst du anklicken um einen Bereich zu öffnen, von dem du die zugehörigen Blöcke per Drag & Drop in die Zeichenfläche ziehen kannst.

Wenn du auf "Mehr" (englisch: "More") drückst, werden noch mehr Blöcke sichtbar, die nicht so häufig verwendet werden.

Es gibt zwei Hauptfunktionen in jedem Arduino-Code - `void setup()` und `void loop()`.

Alles, was in der Funktion `void setup()` enthalten ist, wird **nur einmal** ausgeführt. Sie wird hauptsächlich dazu verwendet, die Software zu starten, Variablen zu initialisieren und zu deklarieren und Funktionen auszuführen, die nur einmal ausgeführt werden müssen (z.B. der Intro-Bildschirm in einem Videospiel).

Die Schleife `void loop()` ist der Ort, an dem sich alles andere abspielt. Sie führt im Grunde jedes Stückchen Code darin wiederholt aus (die Geschwindigkeit hängt vom Gerät ab - stellen dir einfach vor, sie wäre ultraschnell!)

Jeder Block, den du einfügst, wird automatisch in die Funktion `void loop()` eingefügt.

Wenn du etwas in die Funktion `void setup()` einfügen möchtest, musst du den Hauptblock aus Functions herausziehen und deine Blöcke nach Belieben darin platzieren, aber dazu etwas später mehr.

Elliptische Blöcke

Elliptische Blöcke stellen Variablen dar. Ob es sich nun um ganze Zahlen, Zeichenketten oder andere Variablentypen (außer Boolesche) handelt, sie alle sind an der gleichen Form zu erkennen.

Außerdem geben größere Blöcke mit elliptischer Form entweder Integer- oder Float-Werte zurück.

Wann immer du kreisförmige "Löcher" in einigen Blöcken findest, kannst du Variablen einfügen. Dies ist am häufigsten in Vergleichs- oder Aktionsblöcken zu finden.

Dreieckige Blöcke

Dreieckige Blöcke stellen boolesche Variablen dar.

Sowohl Variablen (wahr und falsch) als auch Funktionen, die boolesche Werte zurückgeben, haben die gleiche Form.

Unabhängig von der Farbe gibt jeder dieser Blöcke entweder `true` (wahr) oder `false` (falsch) zurück.

Für dreieckige "Löcher" müssen boolesche Blöcke eingefügt werden.

Programmbausteine

Alles andere sind im Grunde Programmbausteine (englisch "building blocks"). Das sind Funktionen, die keinen Rückgabewert haben. Sowohl elliptische als auch dreieckige Blöcke müssen zunächst innerhalb der Programmbausteine platziert werden, um als Teil des Programms zu fungieren.

Sie haben eine bestimmte "Puzzle"-Form und können ineinander gestapelt werden.

Der **Hauptbaustein** befindet sich im Abschnitt "Funktionen" (englisch "Functions").

Es gibt im Grunde zwei Hauptbausteinabschnitte:

1. "**Arduino run first**" für alles, was zuerst ausgeführt werden soll (der Code landet in `void setup()`) und
2. "**Arduino loop forever**" für die Anteile, die innerhalb der Arduino-Schleife immer wieder wiederholt werden sollen (in `void loop()`).

Blöcke einfügen

Dies ist nun der wichtigste Teil.

Der ganze Sinn der blockbasierten Programmierumgebung ist das Verbinden von Blöcken und deren Platzierung in einem anderen Block.

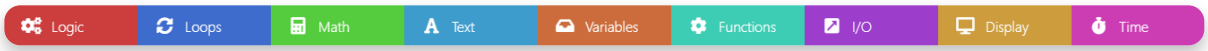
Das alles geschieht durch einfaches **Drag-and-Drop**, also das Verschieben der Blöcke mit der Maus.

Hier ist ein Beispiel für ein Programm, welches die Variable `Var` auf `1` setzt und dann erhöht, solange sie kleiner als `10` ist.

Am Ende des Programms wird **`Var`den Wert `10`haben.**

Dies ist nur ein einfaches Beispiel, und die Blockbildung wird in den folgenden Kapiteln näher erläutert.

Blocktypen



CircuitBlocks bietet insgesamt neun Blocktypen. Wir haben sie so organisiert, dass du alles mit maximal zwei Klicks finden kannst.

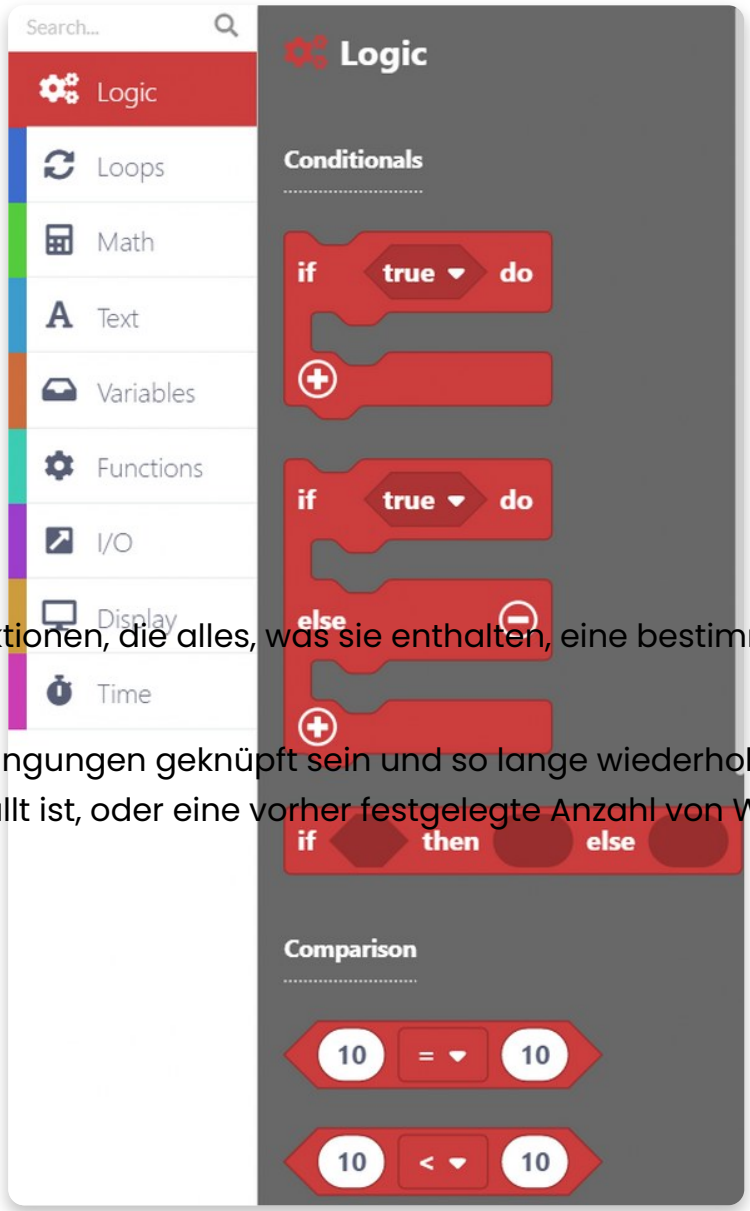
Die Typen selbst sind ziemlich selbsterklärend, aber wir werden sie alle durchgehen, um ein besseres Verständnis für das ganze Konzept zu bekommen.

Einige der Typen bieten **zusätzliche Blöcke** (im Menü "Mehr", englisch "More"), mit Funktionen, die nicht so häufig verwendet werden, aber dennoch nützlich sein können.

Logic

Hier befindet sich die Basis für jeden Code.

Jede Wenn/Dann-Funktion (if, if-else, else), Vergleiche, Und/Oder/Nicht, Wahr/Falsch und andere logische Operatoren.



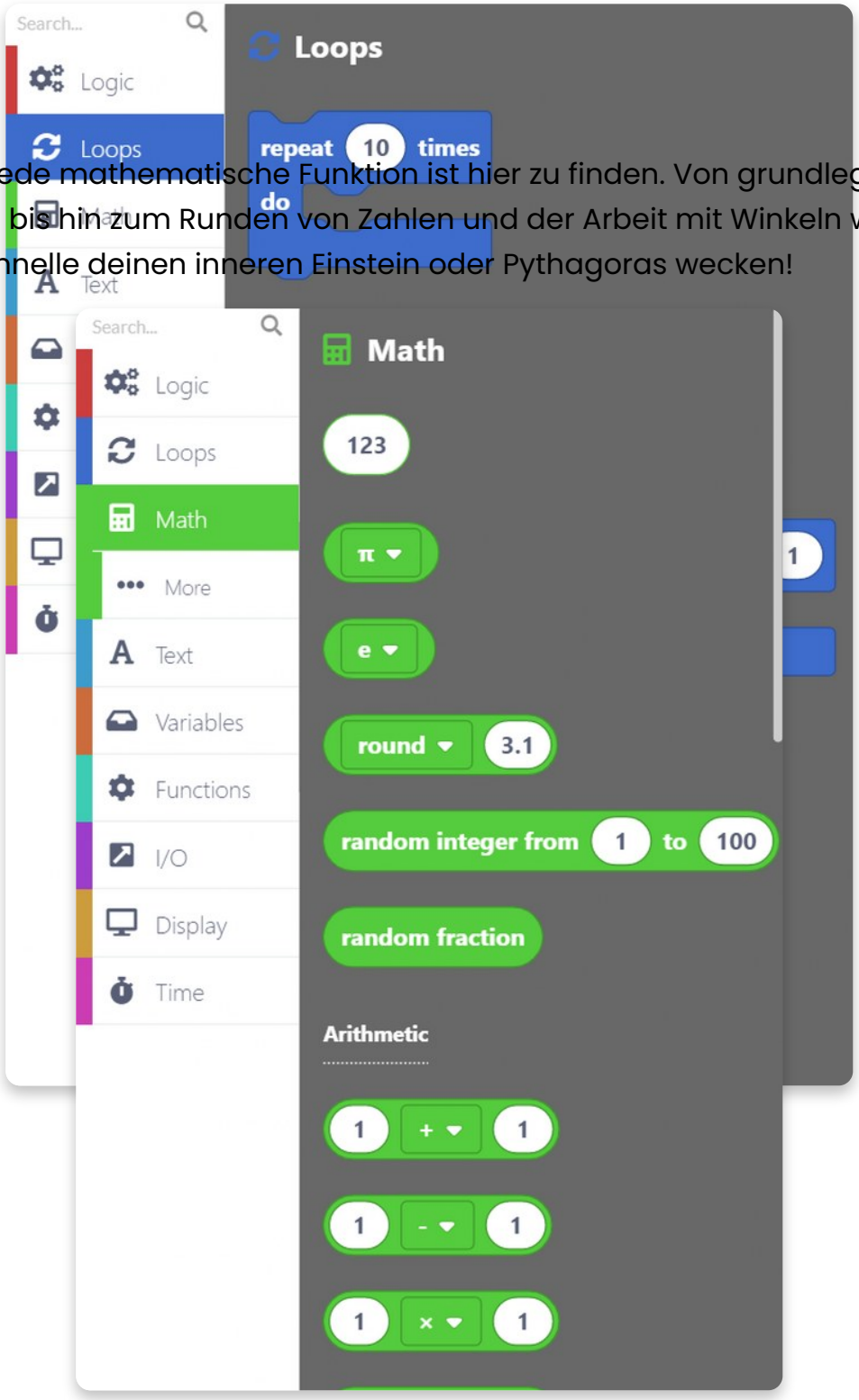
Loops

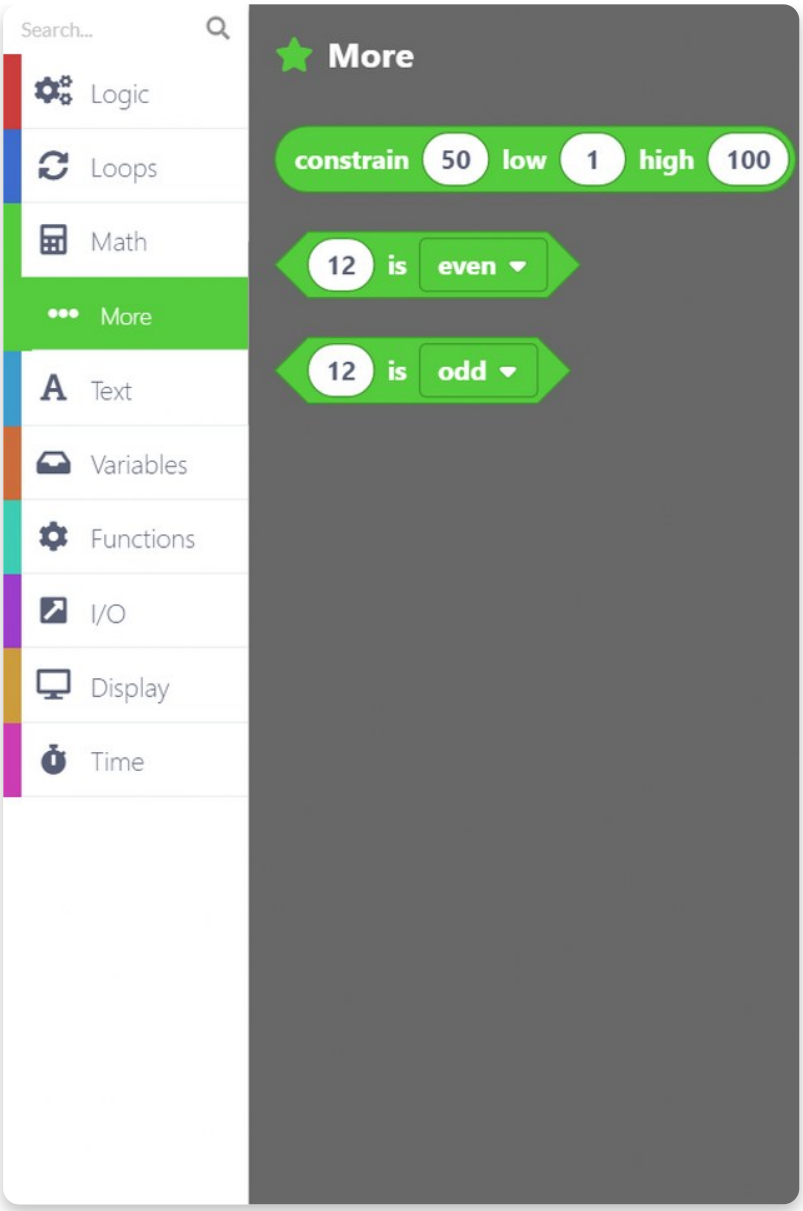
Schleifen sind Funktionen, die alles, was sie enthalten, eine bestimmte Zeit lang wiederholen.

Sie können an Bedingungen geknüpft sein und so lange wiederholt werden, wie die Bedingung erfüllt ist, oder eine vorher festgelegte Anzahl von Wiederholungen haben.

Math

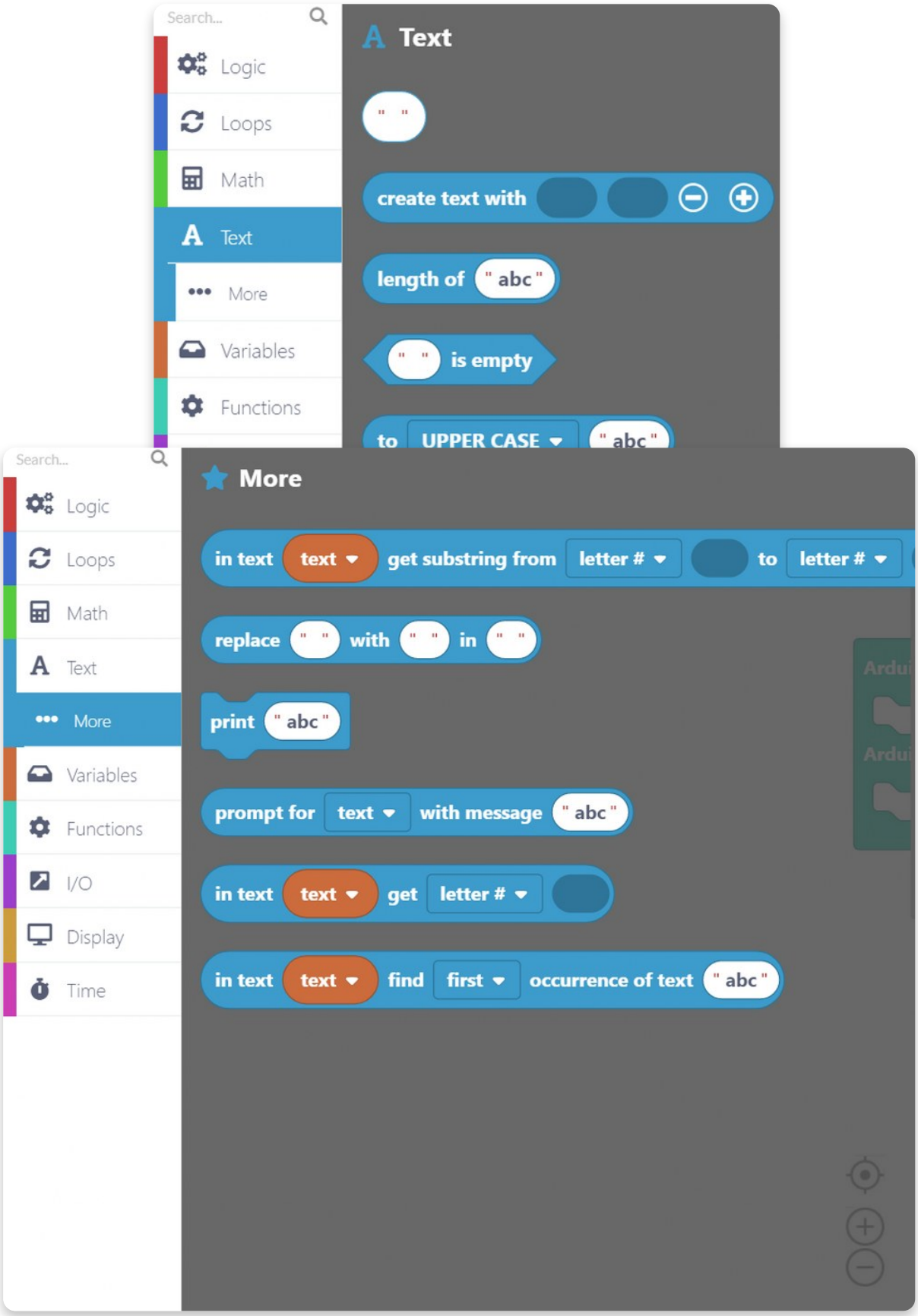
So ziemlich jede mathematische Funktion ist hier zu finden. Von grundlegenden Operationen bis hin zum Runden von Zahlen und der Arbeit mit Winkeln wirst du in sekundenschnelle deinen inneren Einstein oder Pythagoras wecken!





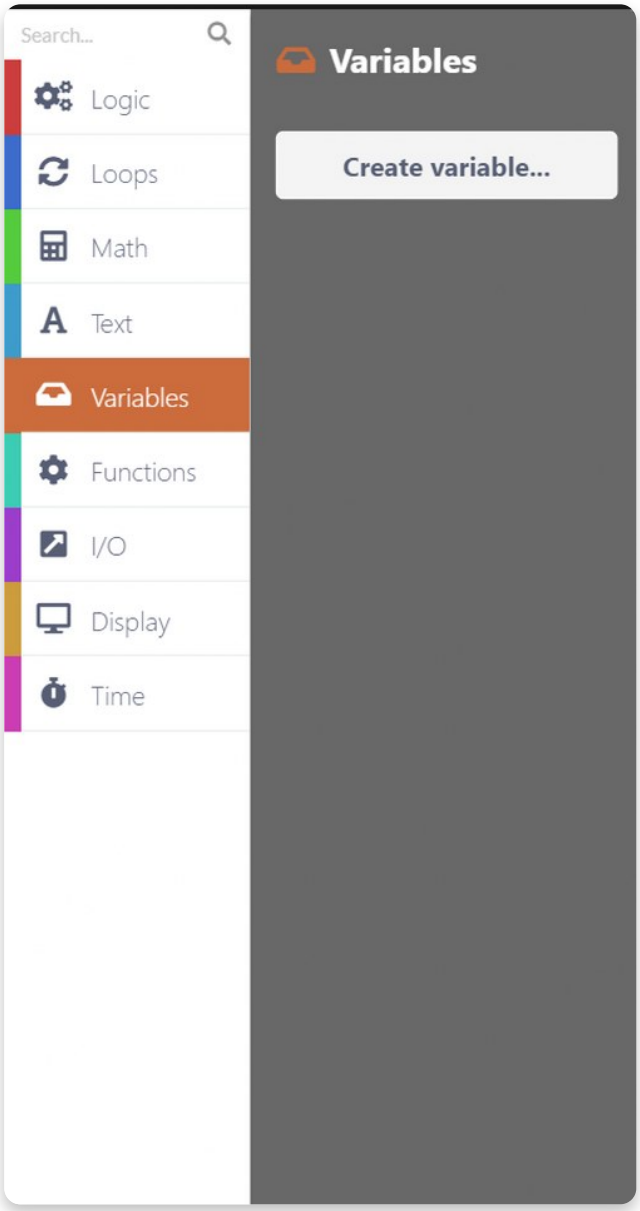
Text

Zeichenketten, Zeichen und Zeichenkettenmanipulation. Ein großartiger Ort, um neuen Text zu erstellen und ihn in deinen Programme zu implementieren.


Search...
Logic
Loops
Math
Text
More
Variables
Functions
Text
" "
create text with
length of " abc "
" " is empty
to UPPER CASE " abc "
Search...
Logic
Loops
Math
Text
More
Variables
Functions
I/O
Display
Time
More
in text text get substring from letter # to letter #
replace " " with " " in " "
print " abc "
prompt for text with message " abc "
in text text get letter #
in text text find first occurrence of text " abc "

Variables

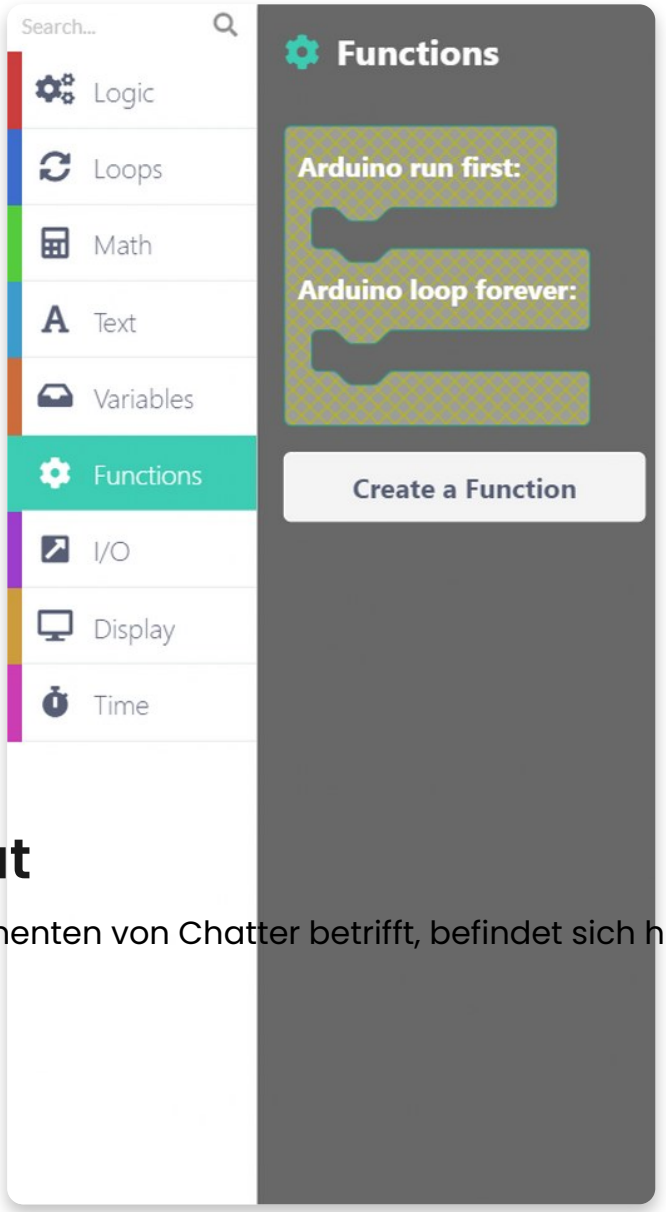
Erstelle eine Variable beliebigen Typs und lege ihren Namen und den gewünschten Wert fest. CircuitBlocks erkennt den Variablentyp (`int, double, string, boolean`) automatisch, so dass du dir darüber keine Gedanken machen musst.



Functions

Die Arduino-Hauptfunktion (die wir im vorherigen Kapitel erklärt hatten) befindet sich hier.

Du kannst auch eigene Funktionen erstellen, die dann als einer der Hauptbestandteile deines Programms eingefügt werden können.



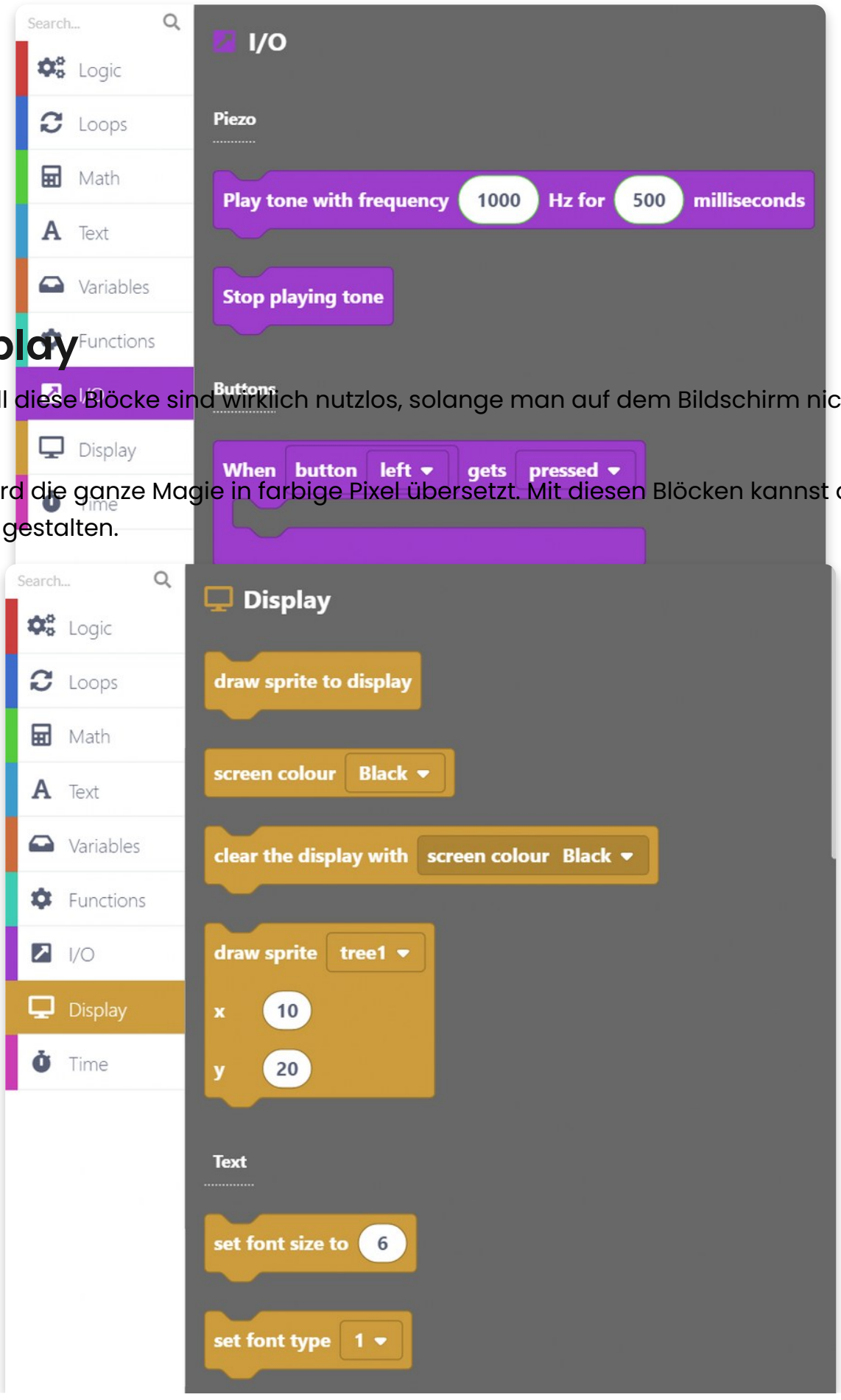
Input/Output

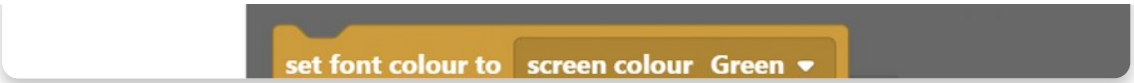
Alles, was die Komponenten von Chatter betrifft, befindet sich hier.

Display

Nun, all diese Blöcke sind wirklich nutzlos, solange man auf dem Bildschirm nichts sieht!

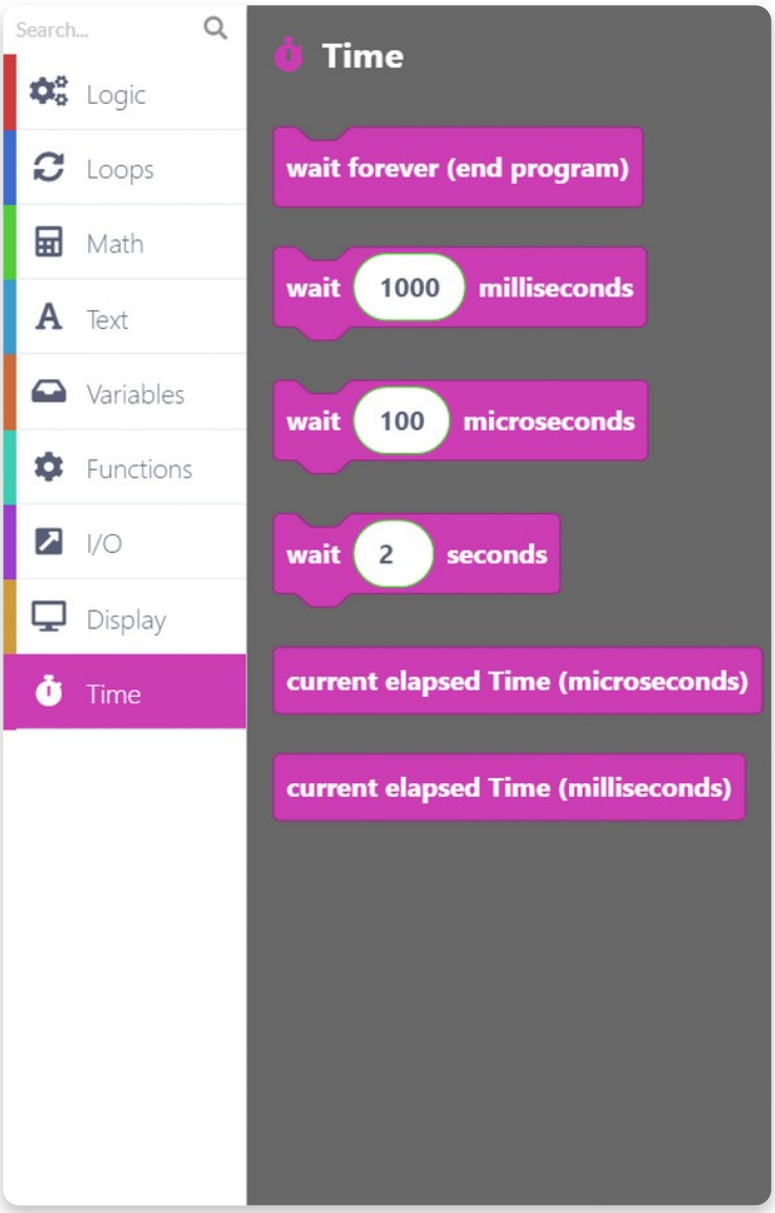
Hier wird die ganze Magie in farbige Pixel übersetzt. Mit diesen Blöcken kannst du so viel gestalten.





Time

Verzögerungen, Timer und andere zeitbezogene Dinge, ideal für die Erstellung cooler Animationen und Videospiele.



Search bar / Suchleiste

Über den Blocktypen befindet sich eine **Suchleiste**, die dir die Suche nach einem bestimmten Block erleichtert, den du einfach nicht finden kannst.

Gib einfach (in englisch) ein, was dir in den Sinn kommt, und alle Blöcke, die etwas mit dem Suchbegriff zu tun haben, werden auf der rechten Seite angezeigt.

Nun kann man wirklich nicht sagen, dass es unmöglich ist, etwas zu finden.

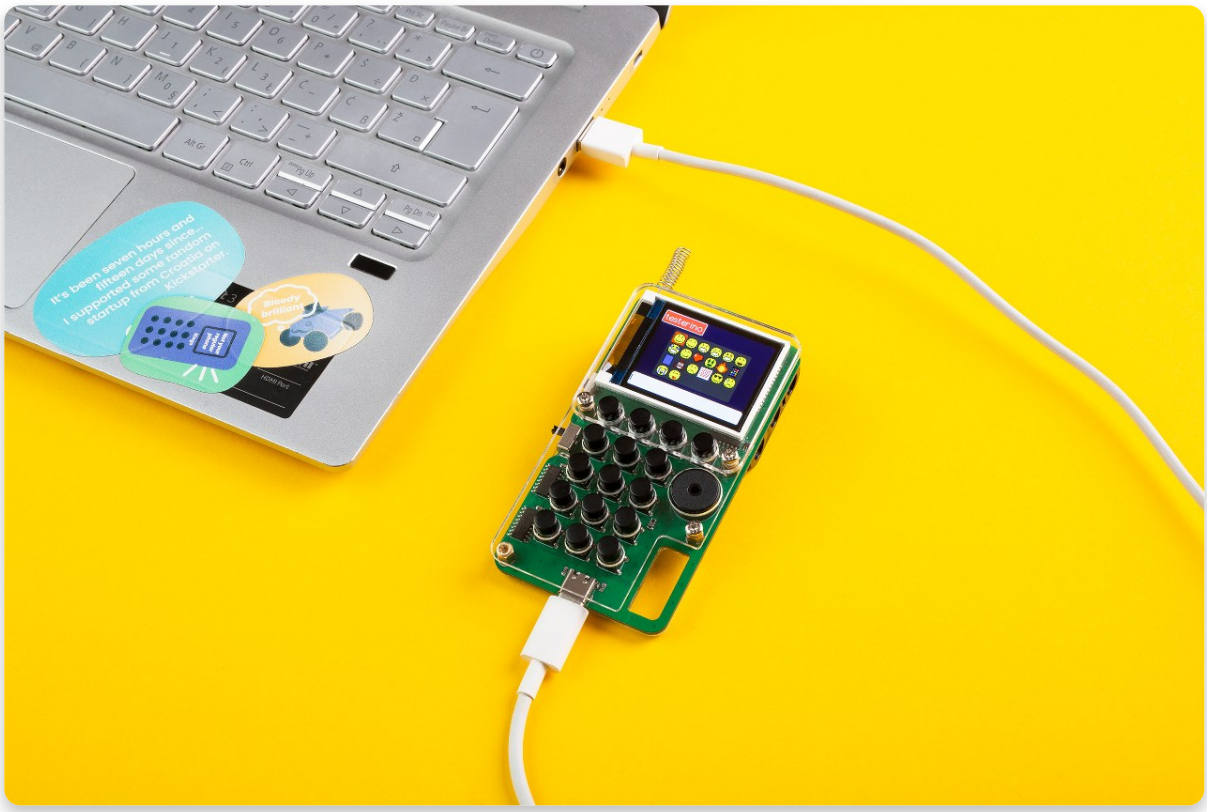
Du hast alles über die Blöcke gelernt! Es ist an der Zeit, zur nächsten Lektion überzugehen...

Los geht's! Schritt für Schritt

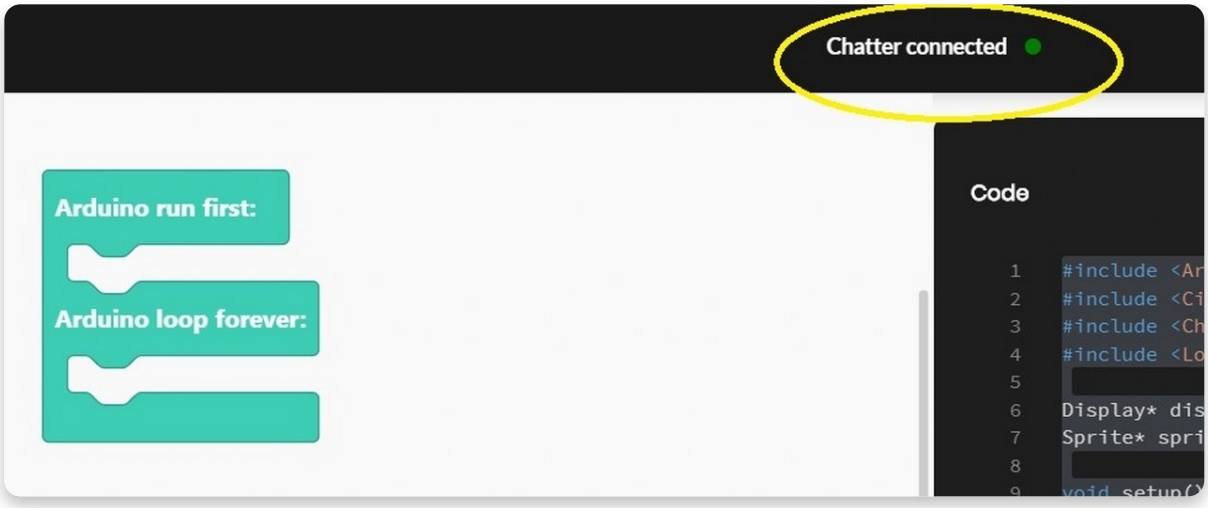
Lass uns auf den Bildschirm schreiben!

Jetzt geht's zur Sache!

Bevor es los gehen kann, musst du deinen Chatter an den USB-Anschluss deines Computers anschließen und ihn einschalten.



Wenn alles in Ordnung ist, sollte CircuitBlocks "**Chatter connected**" anzeigen.



Wenn CircuitBlocks deinen Chatter nicht erkannt hat, überprüfe bitte, ob das USB-Kabel richtig eingesteckt ist und ob du einen funktionierenden USB-Anschluss an deinem Computer verwendest.

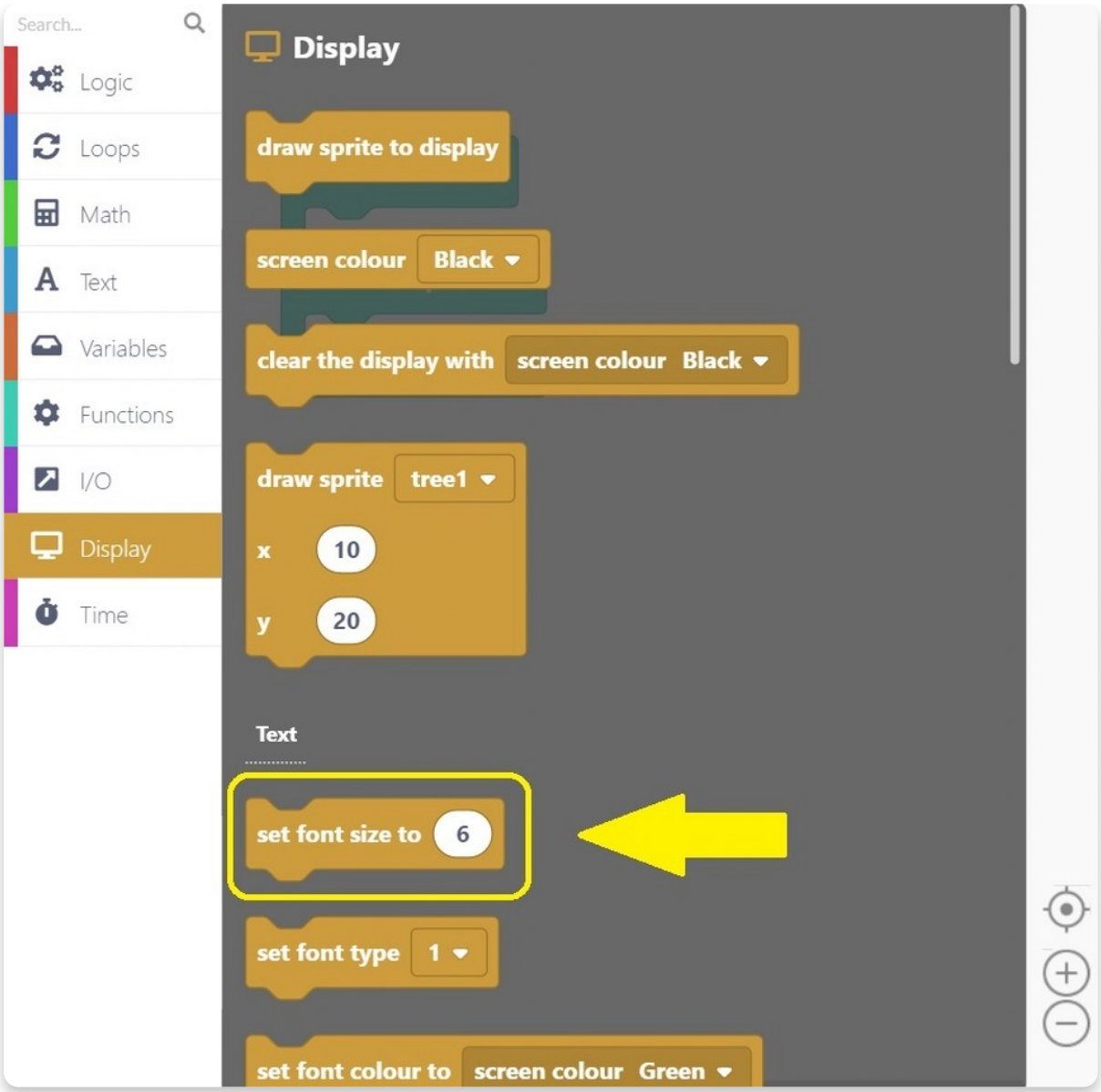
Wenn CircuitBlocks dann deinen Chatter immer noch nicht erkennt, ist möglicherweise etwas mit der Treiberinstallation auf deinem Computer schief gelaufen. Treiber sind diese kleinen Programme, die deinem Computer helfen, mit Chatter zu kommunizieren, und sie verhalten sich manchmal komisch. Wenden dich per E-Mail an contact@circuitmess.com, wenn dein Computer deinen Chatter nicht erkennt.

Lasst uns etwas schreiben!

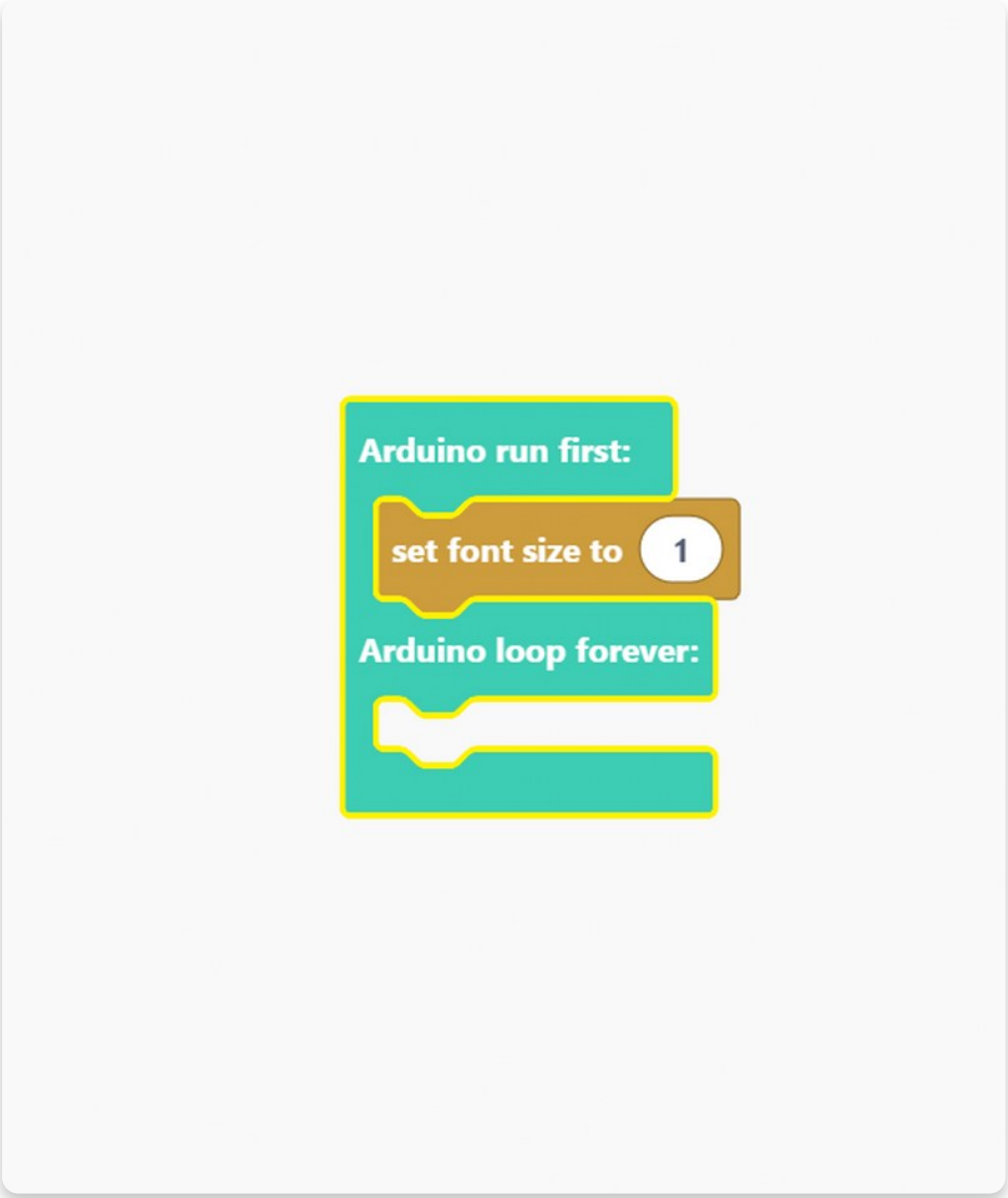
Wir werden die Dinge so einfach wie möglich angehen.

Das erste, was wir lernen werden, ist, wie man den **Bildschirm von Chatter löscht** und darauf **schreibt**!

Dazu benötigen wir nur Blöcke eines einzigen Blocktypes: Bildschrim (Display). Bitte klicke auf den genannten Blocktyp und wähle "Schriftgröße auf 6 setzen" (englisch "set font size to").



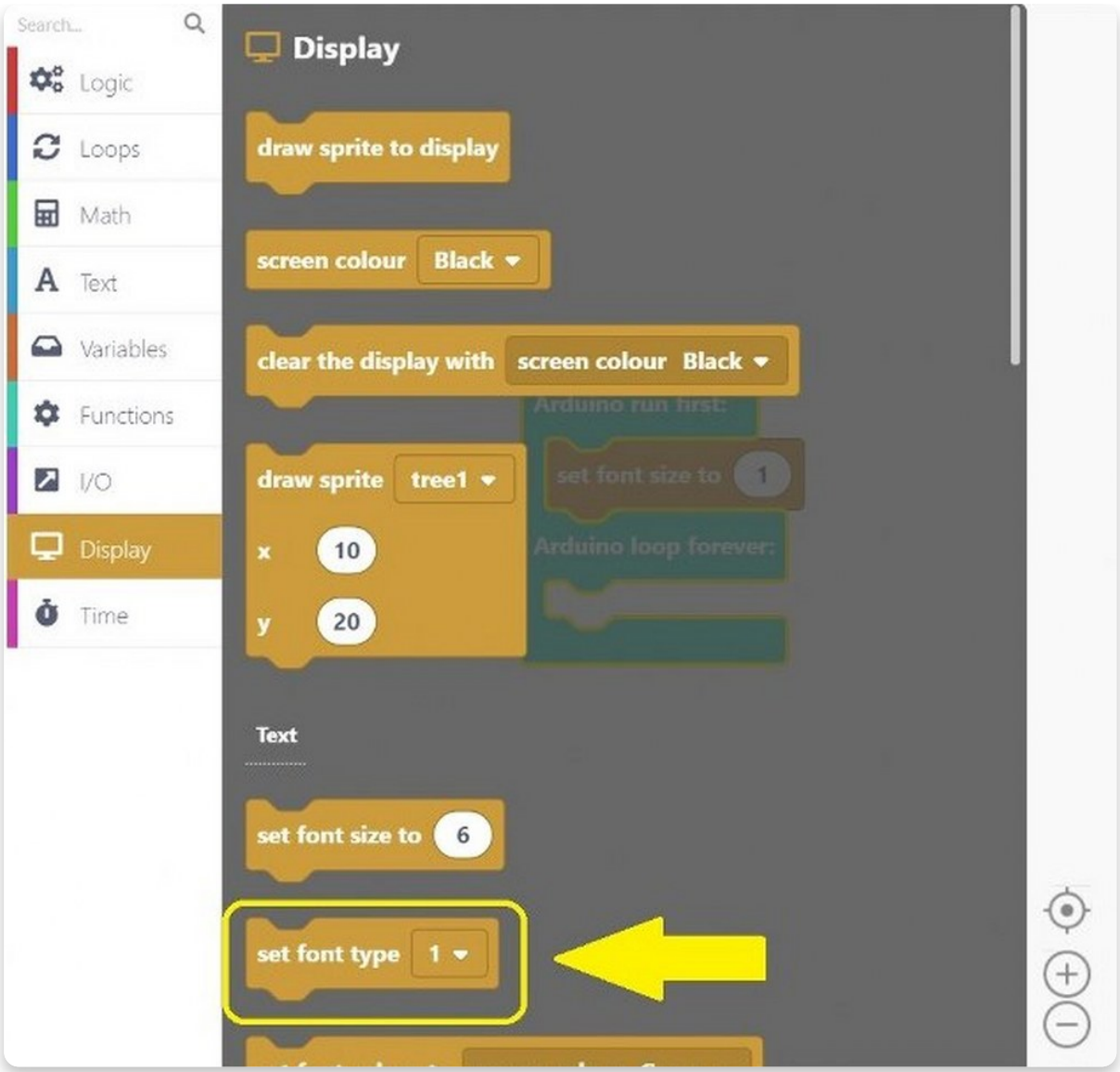
Sobald du auf den Block klickst, erscheint er im **Zeichenbereich** und du musst ihn in den bläulichen "**Arduino run first**"-Block ziehen.



Wir finden, dass die Schriftgröße 6 zu groß sein könnte, also haben wir sie in 1 geändert. Du kannst das tun, indem du einfach die "6" löschst und stattdessen "1" einträgst.

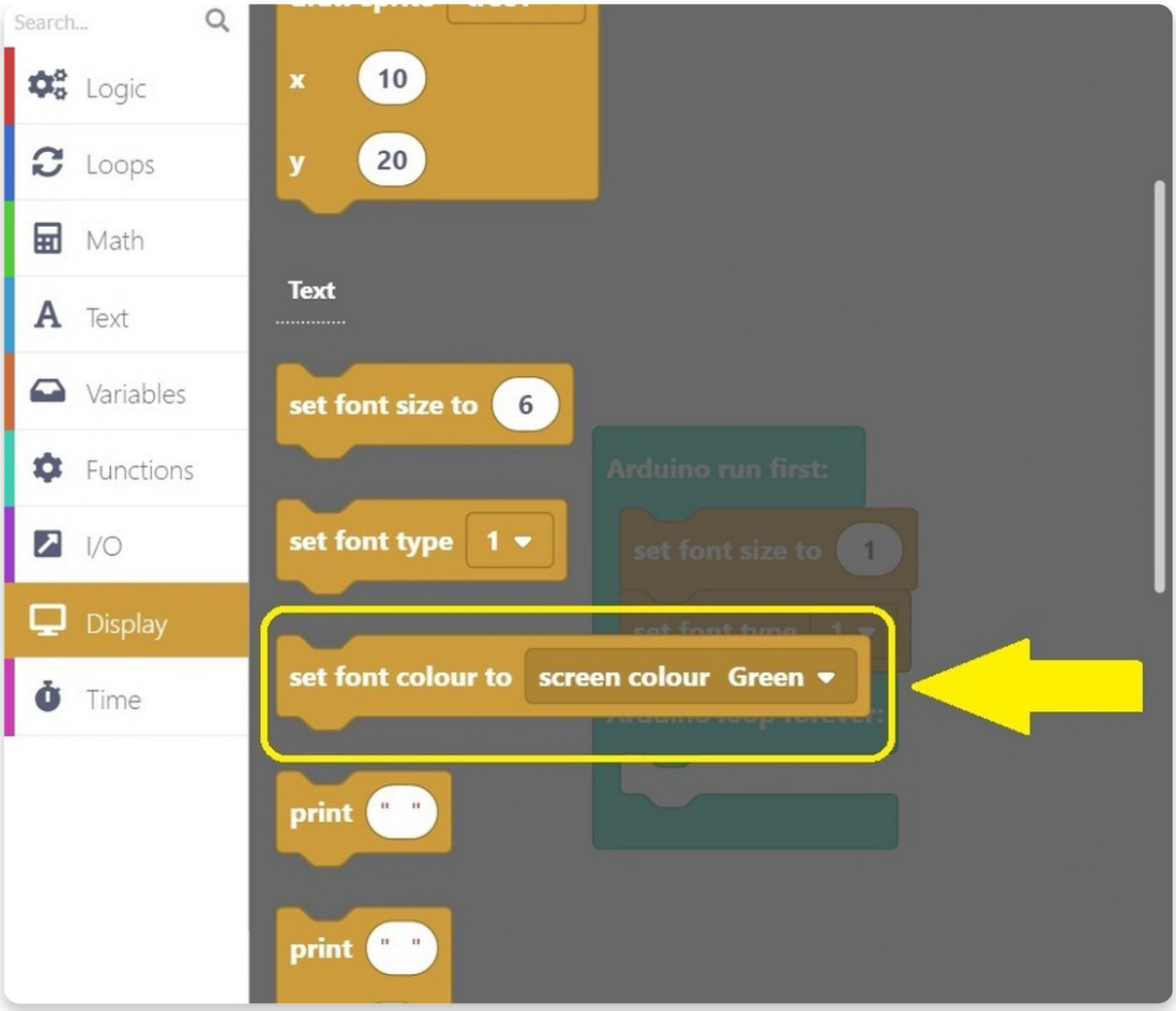
Einfach, nicht wahr?

Als Nächstes müssen wir eine **Schriftart** festlegen. Den Block "set font size" findest du ebenfalls im Abschnitt "Display".



Die Schriftart muss nicht geändert werden, aber wenn du experimentieren möchtest, kannst du das gerne tun!

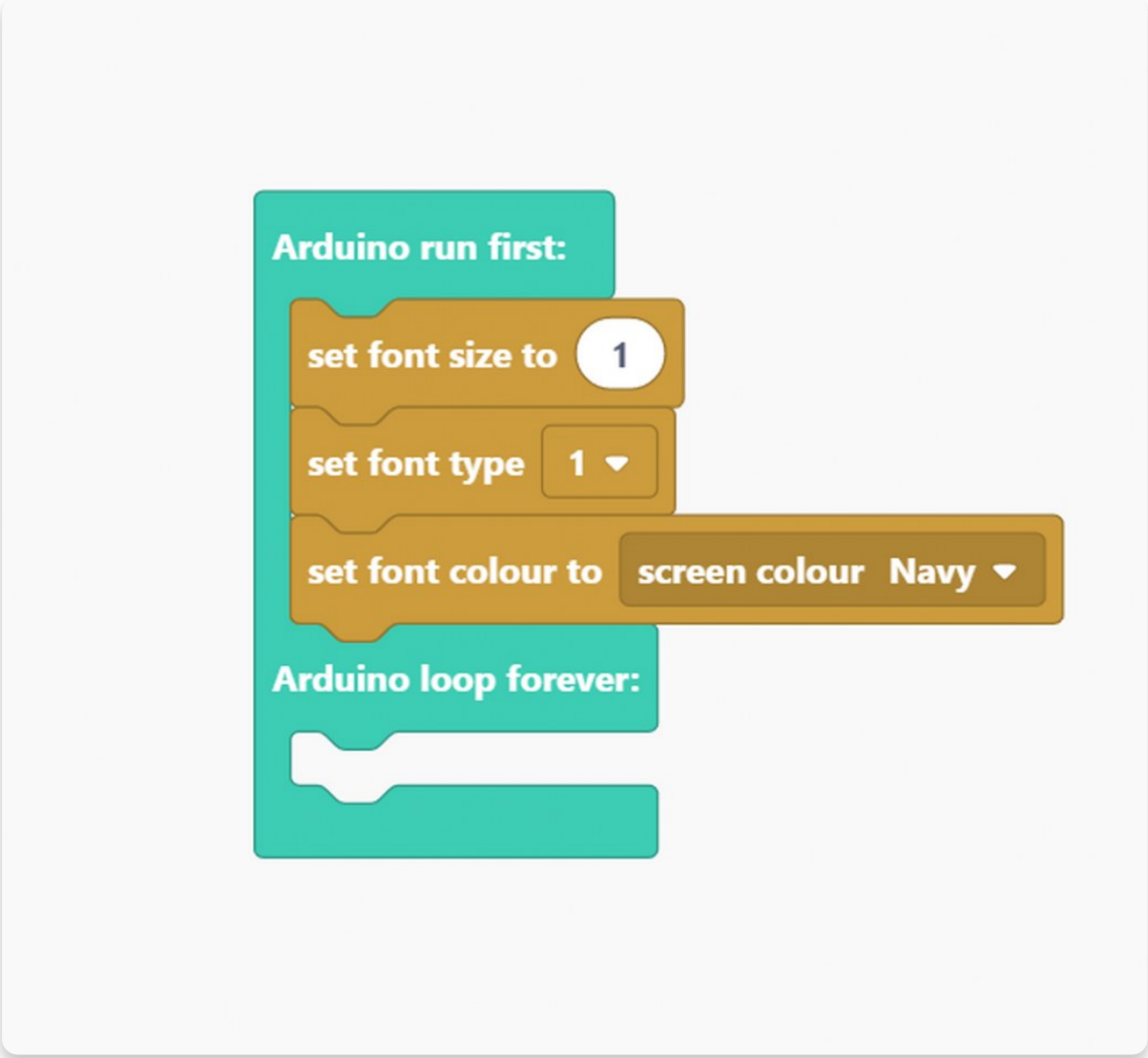
Das dritte Element, das wir aus dem Abschnitt "Display" verwenden werden, ist der Block "**Schriftfarbe auf Bildschirmfarbe setzen**" (englisch "set font colour to screen colour").



Anstelle der vorausgewählten Farbe grün ("Green") wählen wir als Schriftfarbe Marineblau (englisch "Navy").

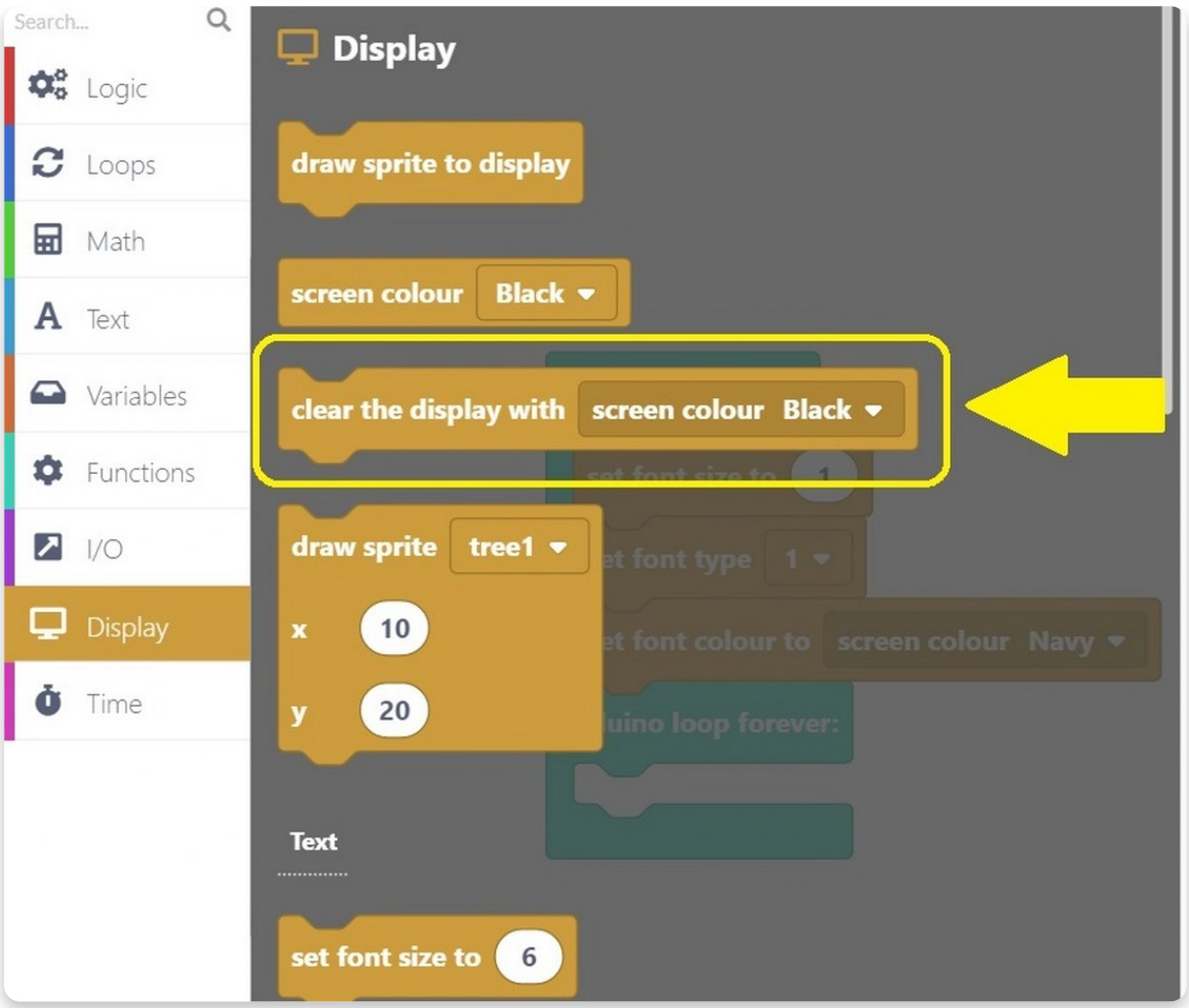
Du kannst jede beliebige Farbe wählen die du magst.

Dein Programm sollte jetzt wie folgt aussehen:



Nachdem du nun alle erforderlichen Größen, Farben und Schriftarten eingestellt hast, ist es an der Zeit, den Bildschirm zu löschen und komplett mit einer bestimmten Farbe zu füllen, damit der anzuzeigende Satz (oder das Wort) deutlich zu sehen ist.

Wie im vorherigen Block kannst Du jede beliebige Farbe wählen.

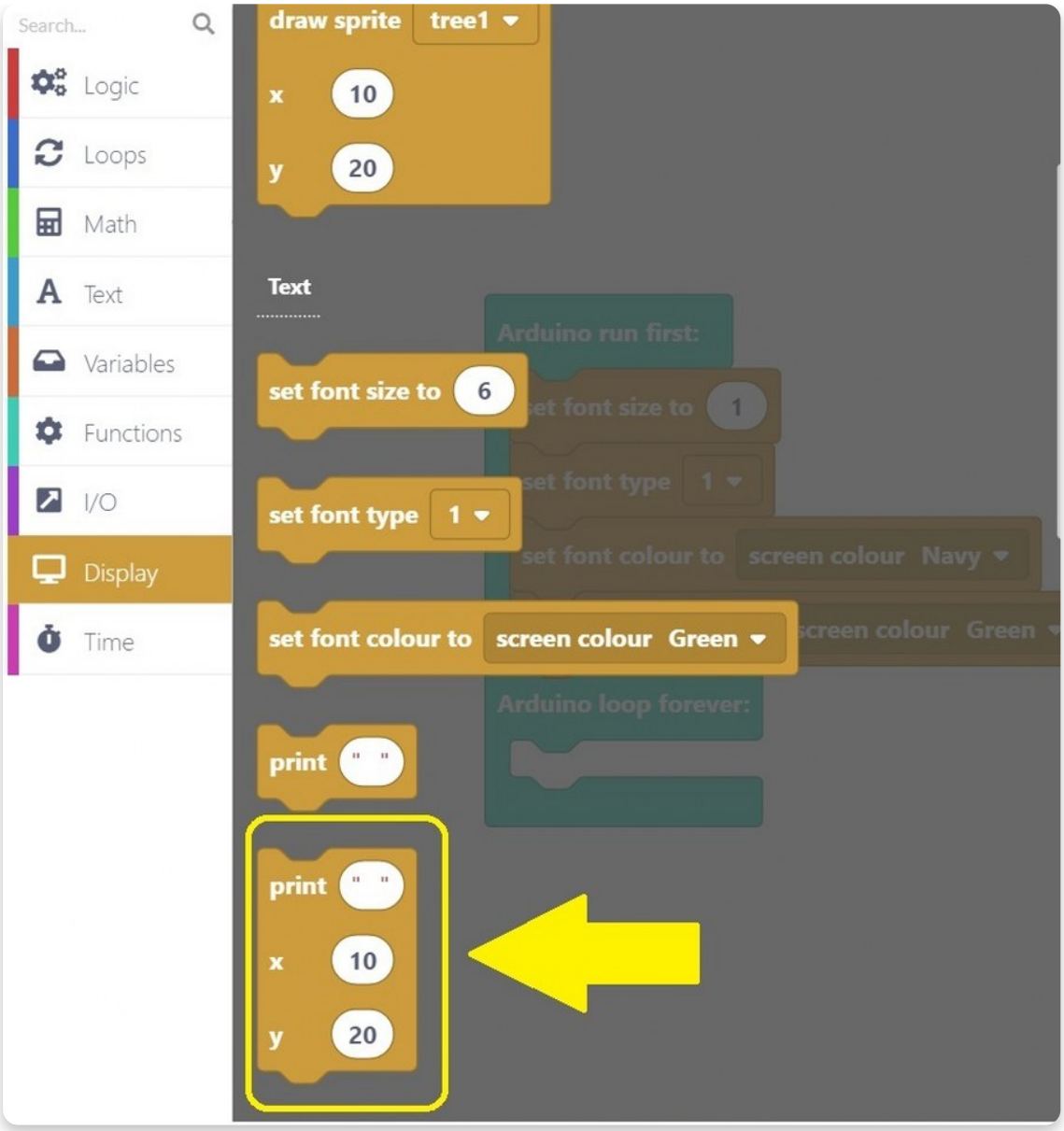


Klicke auf den eingekreisten Block und ziehe ihn in den **Arduino-Run-First**-Block wie die anderen Blöcke.

Wie du sehen kannst, haben wir beschlossen, dass der Bildschirm grün (englisch: "Green") sein soll.

Lass uns den Hauptteil unseres Programms beginnen - schreibe einen Satz (oder ein Wort), dass auf dem Bildschirm erscheinen soll.

Um das zu tun, musst du diesen Block verwenden:



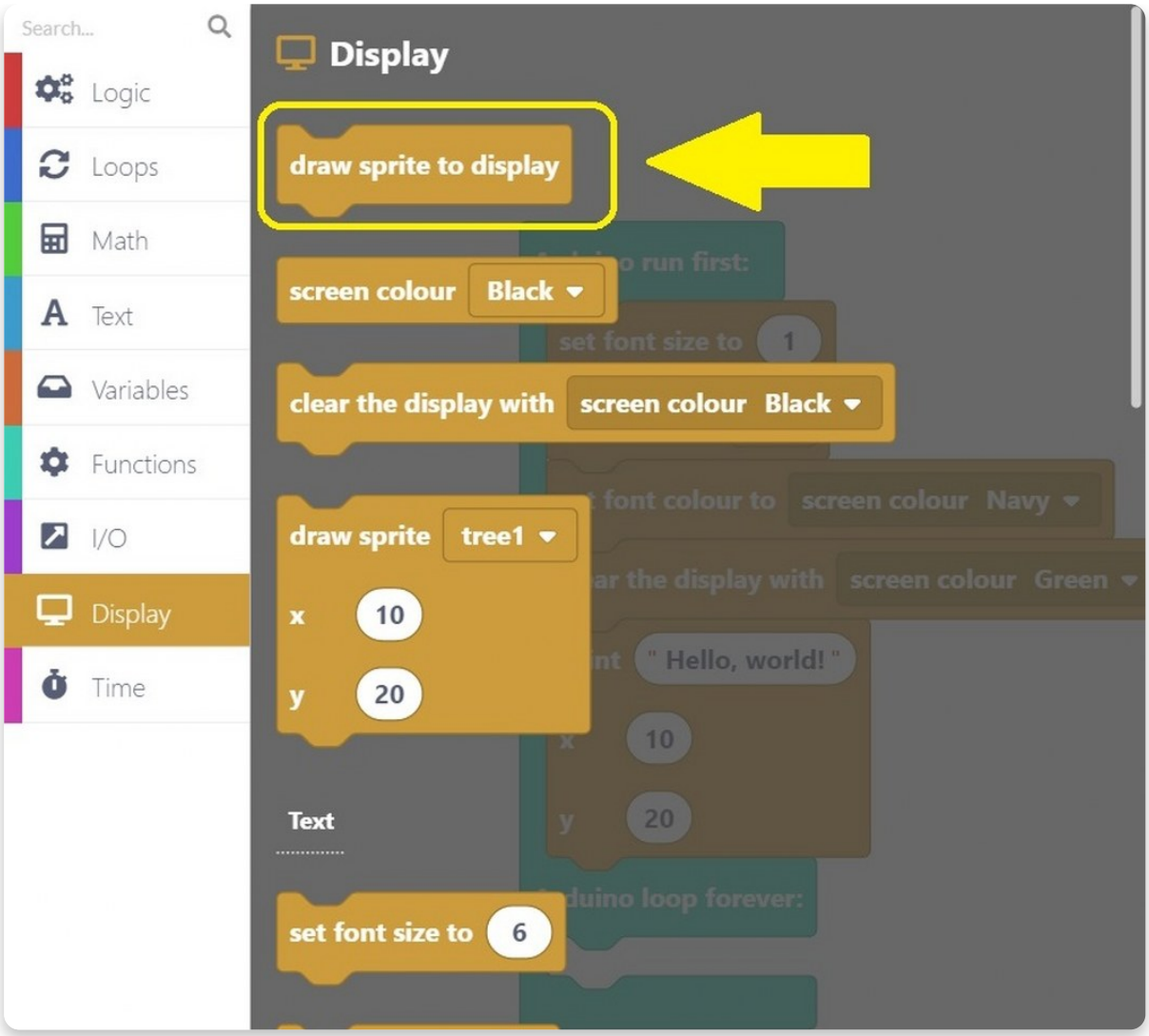
Wie du sehen kannst, gibt es drei weiße Kreise in die man etwas eintragen kann.

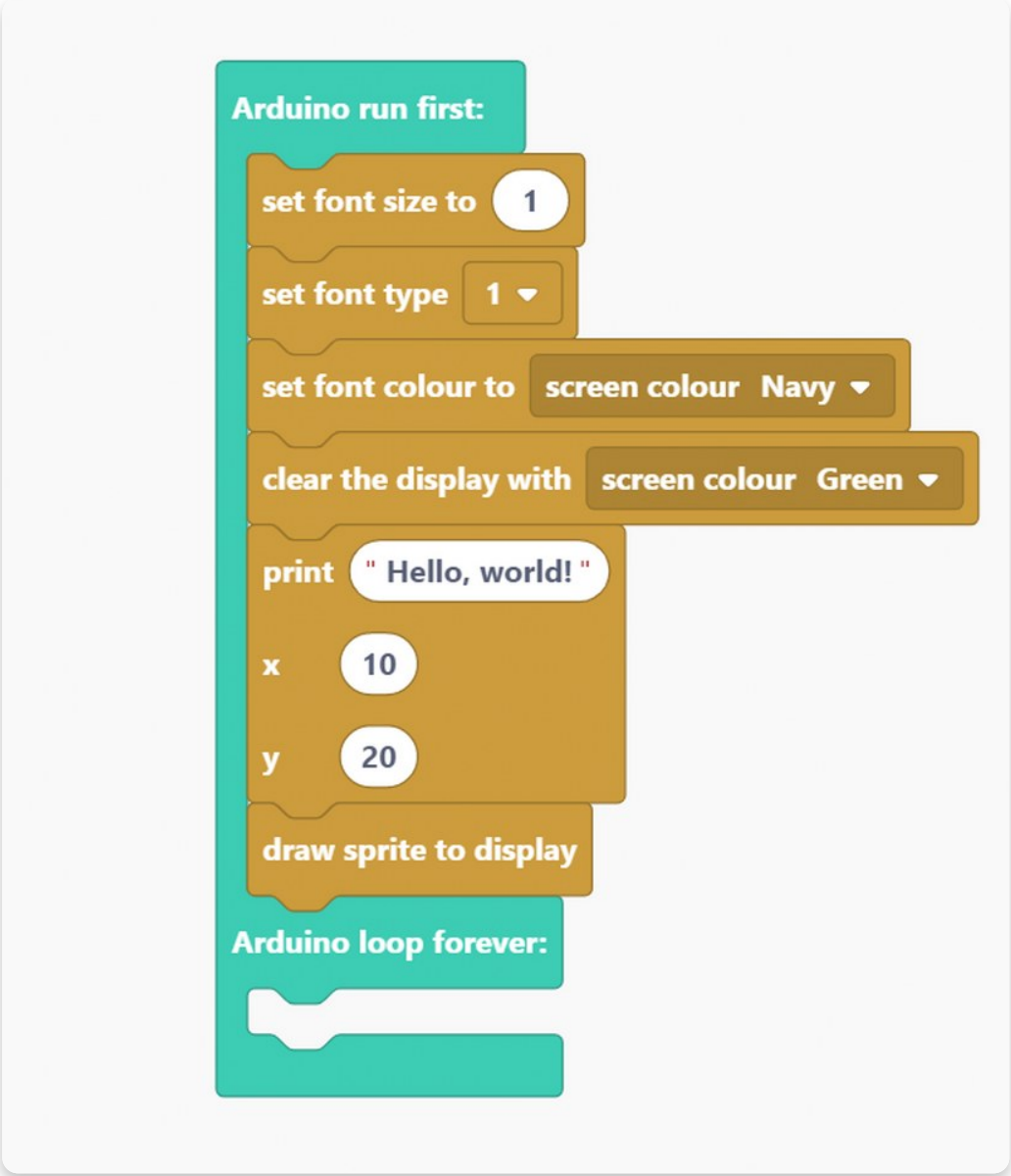
1. Der erste (neben "**print**") bestimmt den Satz, der auf dem Bildschirm erscheinen soll.
2. Die **X- und Y-Koordinaten** geben an, wo der Text auf dem Bildschirm erscheinen soll.

Wir haben uns entschieden, "Hallo, Welt!" auf den Bildschirm zu schreiben, aber wir haben die Koordinaten beibehalten.

Das letzte, was du nun noch tun musst, ist auf einen "**draw sprite to display**"-Block zu klicken.

Wir müssen diesen Block verwenden, um sicherzustellen, dass dieser Code auf dem Bildschirm angezeigt wird.


Search...
Logic
Loops
Math
Text
Variables
Functions
I/O
Display
Time
Display
draw sprite to display
screen colour Black
clear the display with screen colour Black
draw sprite tree1
x 10
y 20
Text
set font size to 6
set font size to 1
font colour to screen colour Navy
screen colour Green
" Hello, world! "
10
20

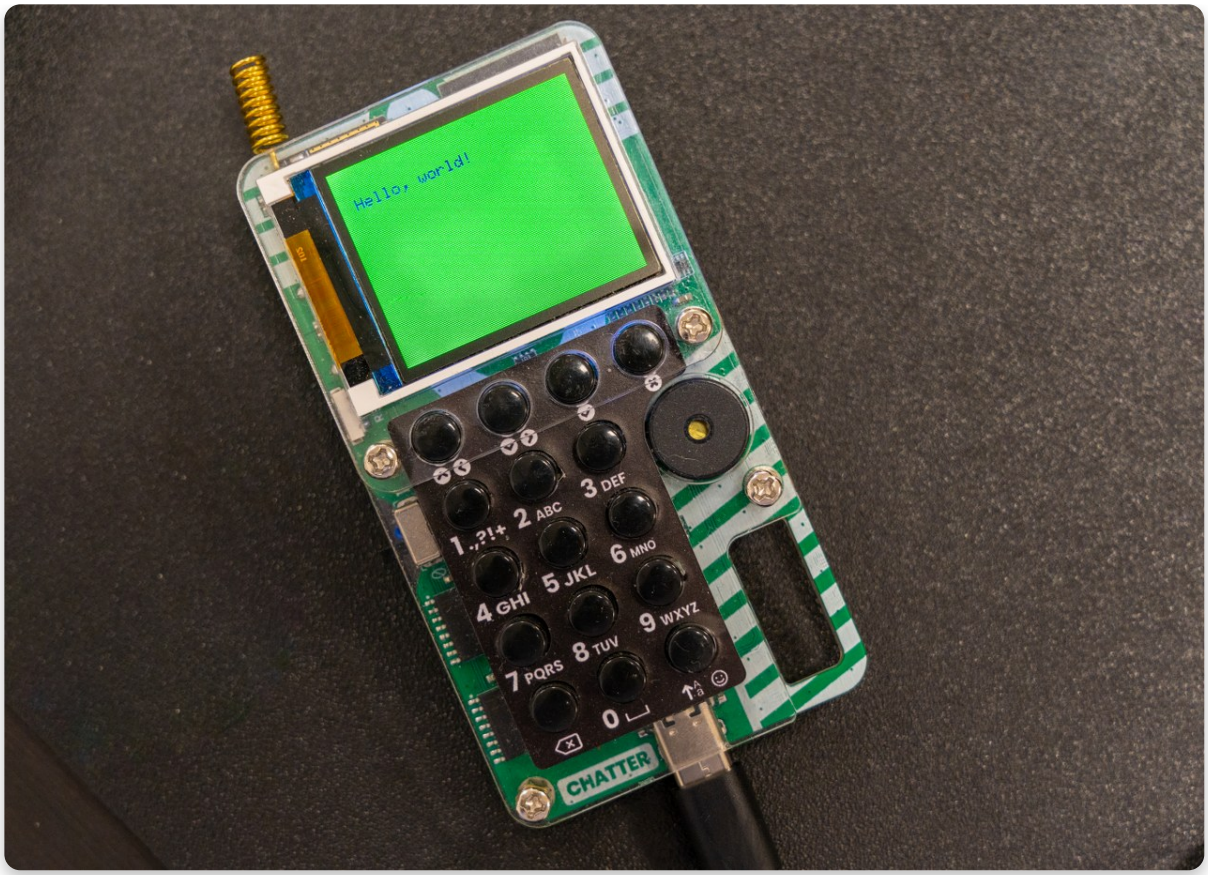


Klicke nun auf die große rote Schaltfläche Ausführen ("**Run**") und warte, bis der **Code kompiliert** ist!

Beim ersten Mal kann es bis zu einer Minute dauern, bis der Code kompiliert ist. Aber keine Sorge, danach sollte das Kompilieren schneller gehen.

Wenn du auf die Schaltfläche Ausführen klickst, erscheint eine rote Linie unter der Symbolleiste, die den Prozentsatz des kompilierten Codes anzeigt. Sobald der Code kompiliert ist, wird dein Chatter neu gestartet, der Bildschirm wird grün und in der Farbe Marine erscheint ein Text, der sagt: "Hallo, Welt!"

So wie auf dem Foto unten sollte es aussehen:



Loading...

Buzzzzzer - der Summer

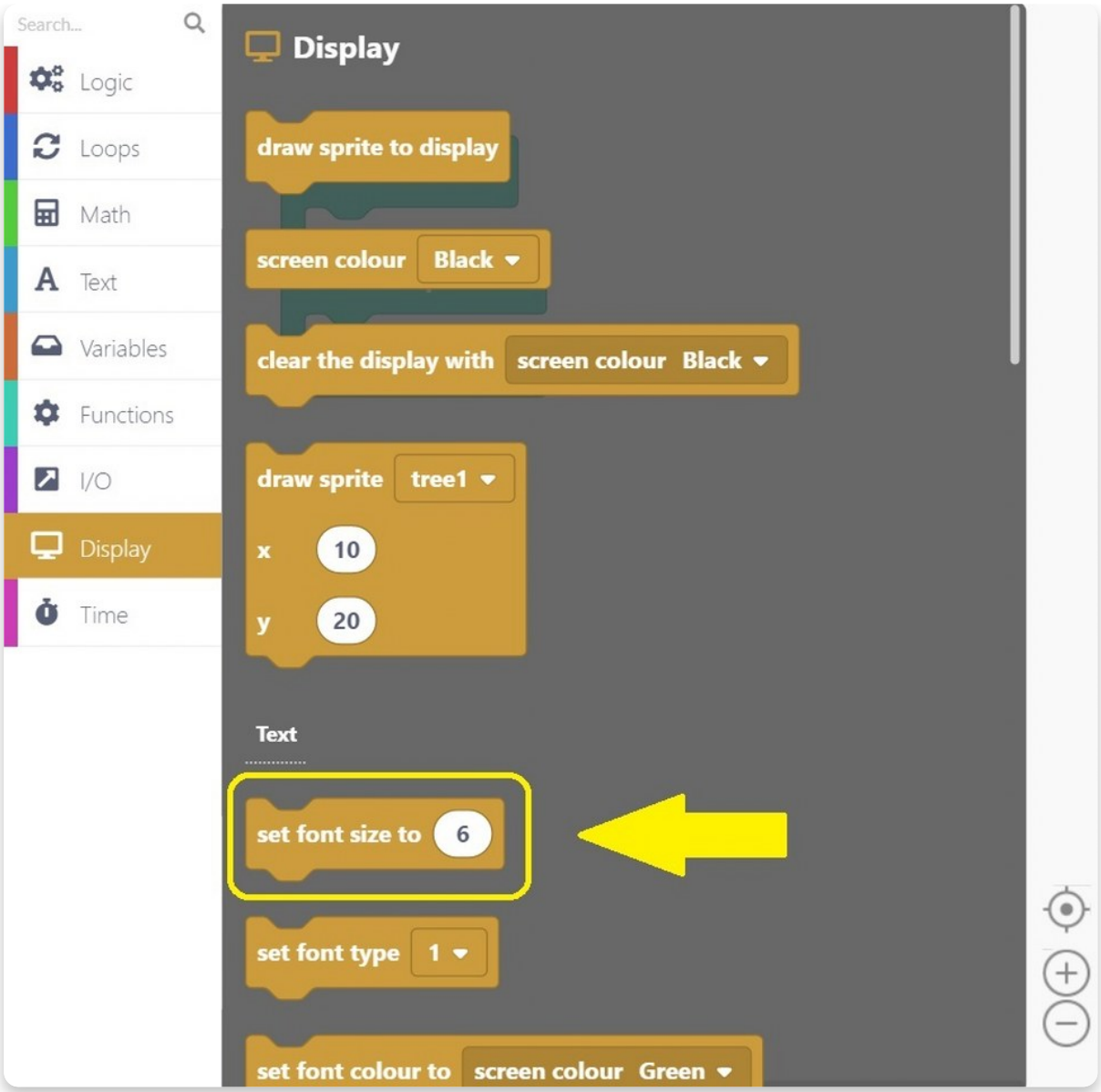
Wir wollen lernen, was man mit dem **Piezo-Summer** macht, den du auf deinen Chatter gelötet hast.

Wie das Wort schon sagt, wird der Summer zur Erzeugung von Summtönen verwendet.

Wir werden eine sehr ähnliches Programm wie das letzte erstellen, **aber dieses Mal wird das Drücken der Tasten einen bestimmten Ton aus dem Summer auslösen.**

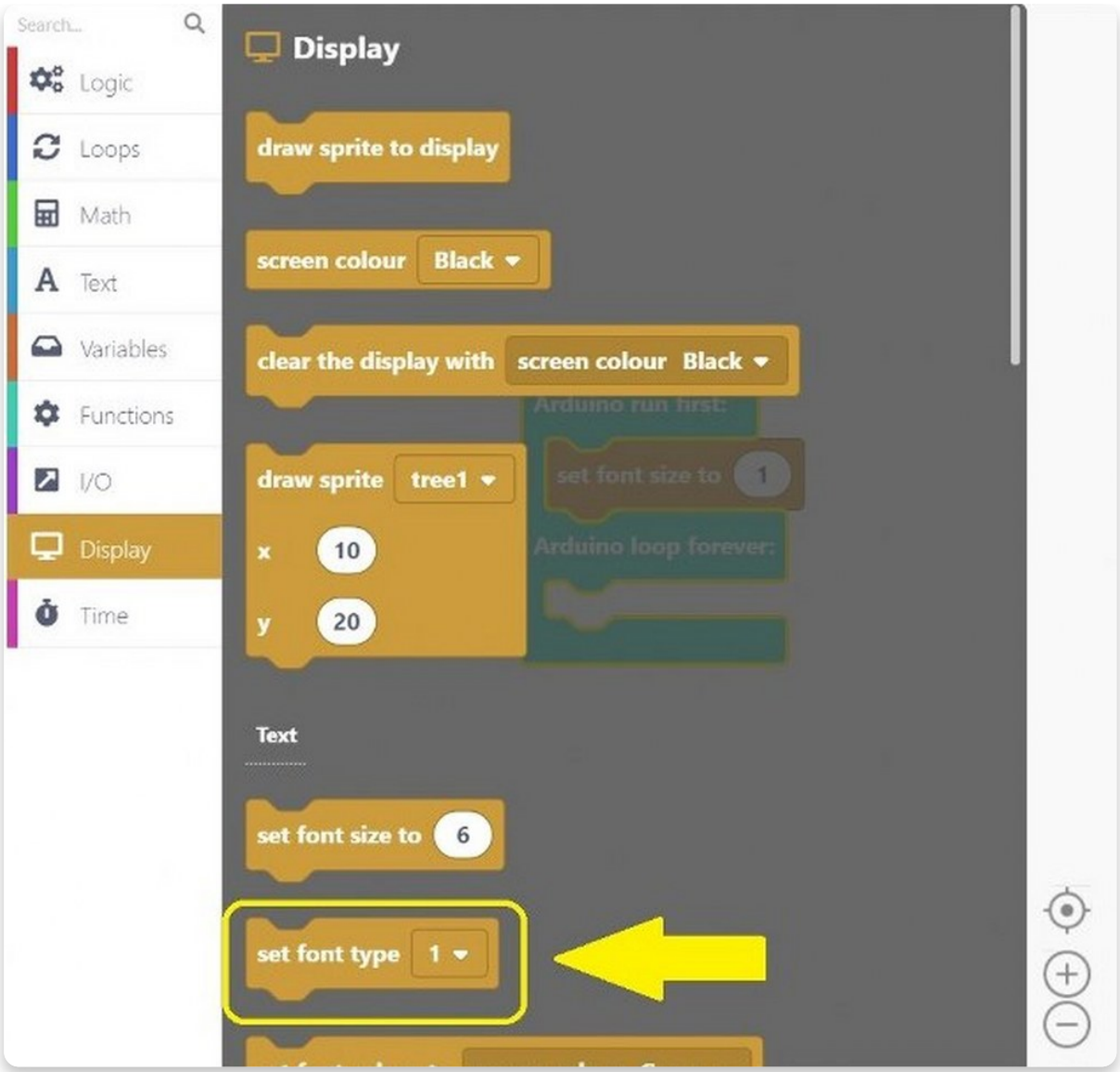
Fangen wir an!

Für den Anfang öffne den Bereich "**Display**" und klicke auf den Block "**Schriftgröße einstellen auf**".



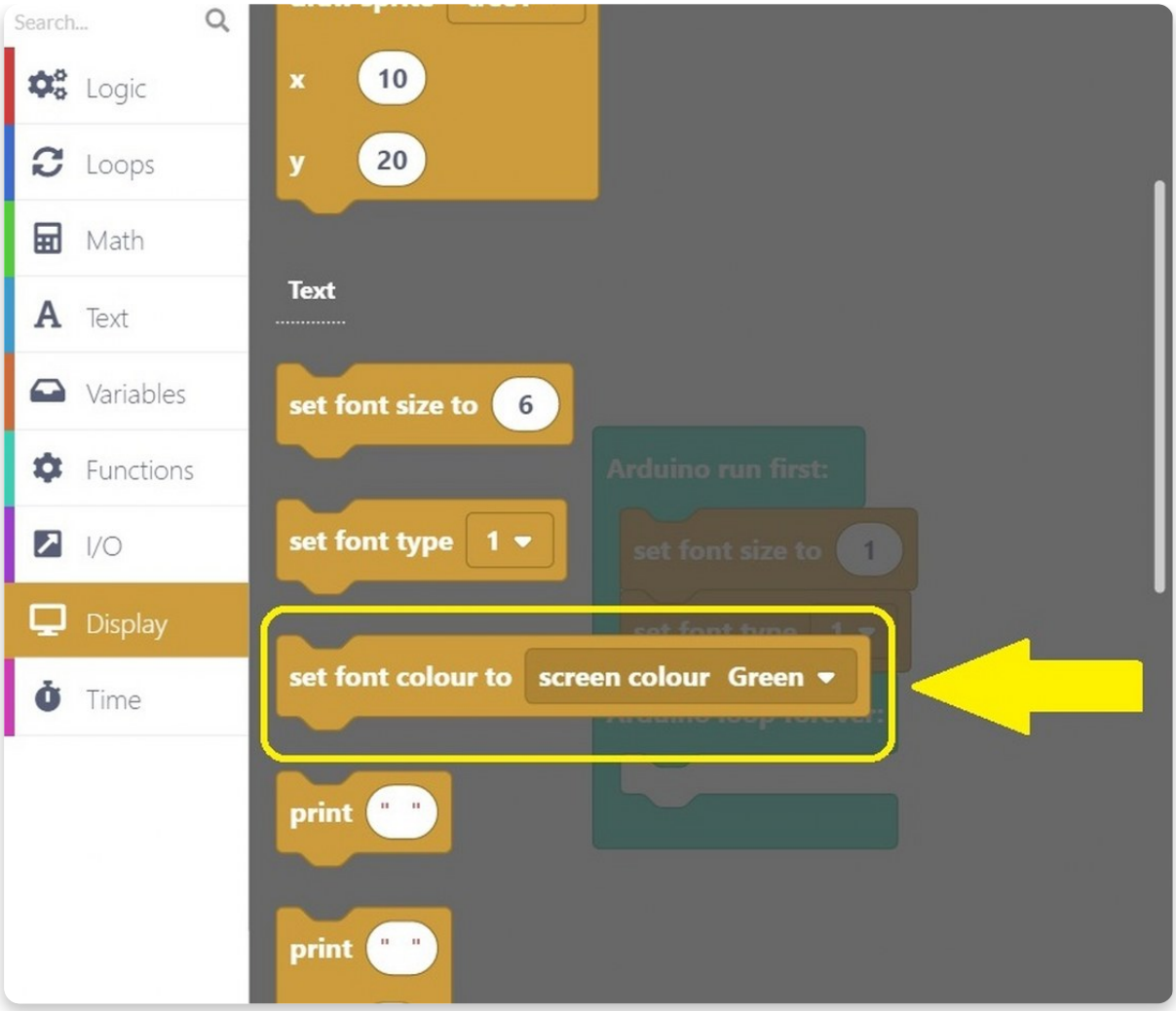
Auch hier ändern wir die Schriftgröße auf 1, wie bei den letzten beiden Programmen.

Als nächstes müssen wir eine **Schriftart** festlegen - über einen weiteren Block aus dem Abschnitt "Display".



Klicke auf den Block und ziehe ihn auf die Zeichenbereich.

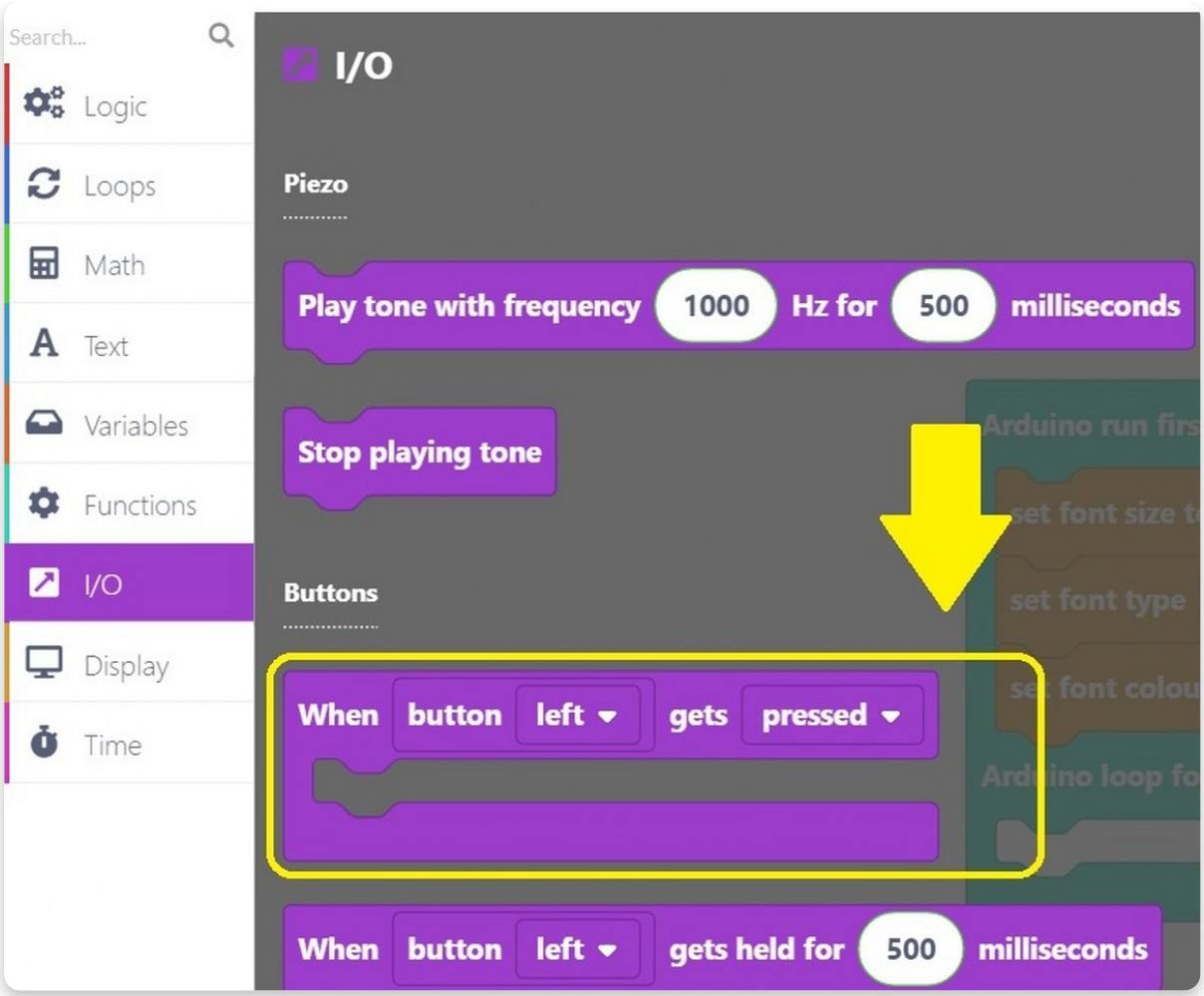
Und jetzt legen wir die **Schriftfarbe** fest.

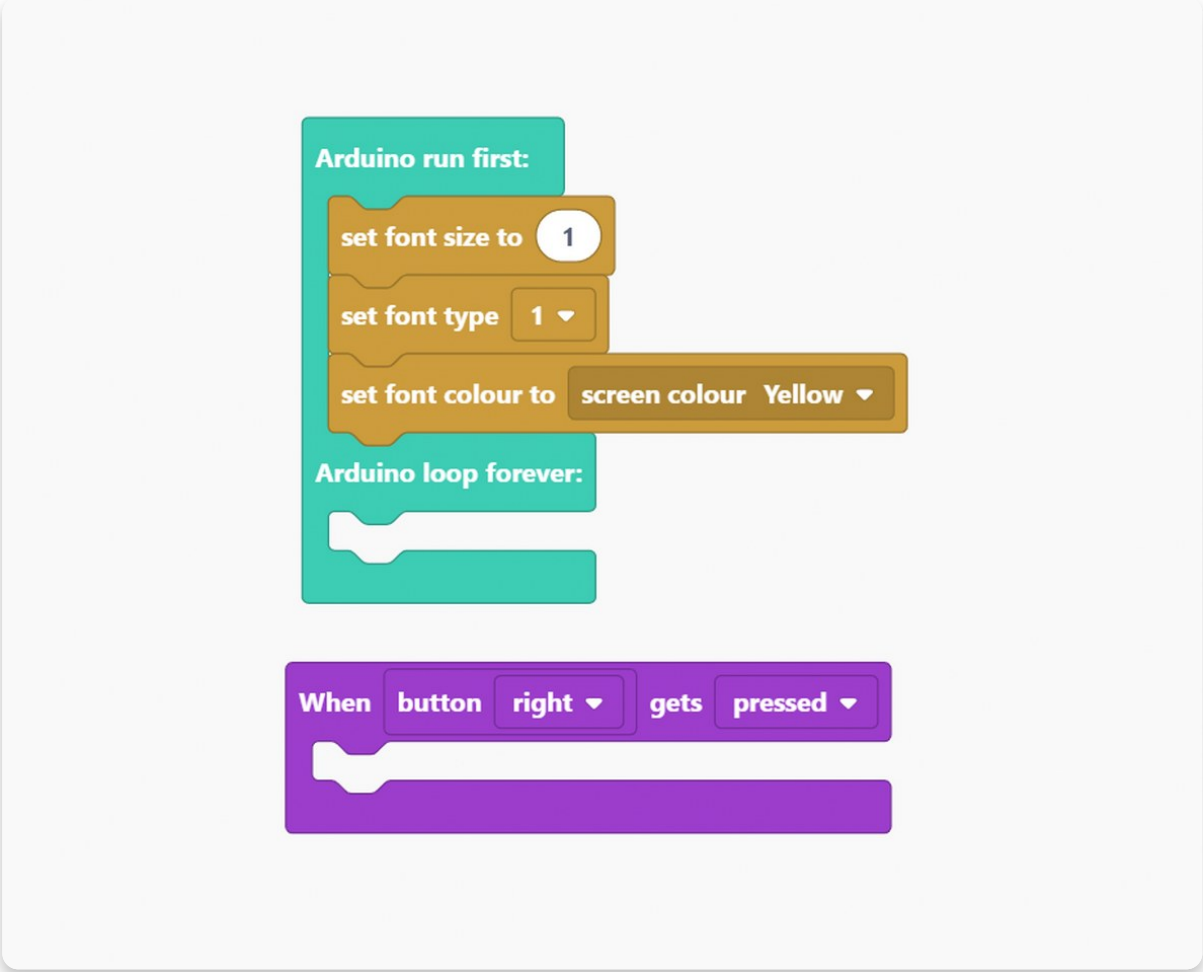


Diesmal wählen wir die Schriftart **Gelb** (englisch: yellow).

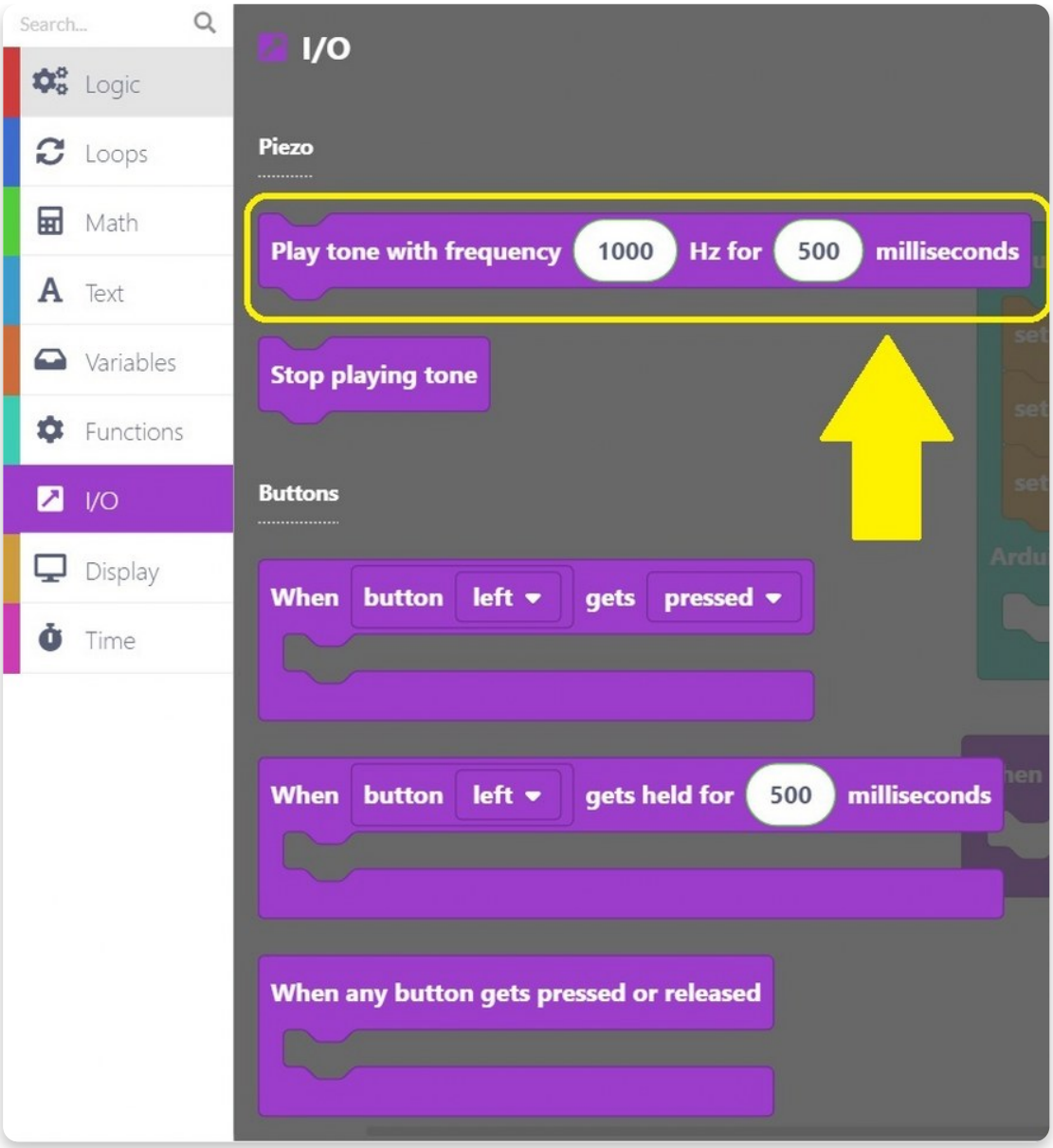
Nun, da wir eine Schrift eingestellt haben, können wir ändern, **was mit dem Summer passiert, wenn eine bestimmte Taste gedrückt oder losgelassen wird**.

Wie im vorherigen Programm verwenden wir Blöcke aus dem Abschnitt **I/O**, um zu bestimmen, was passiert, wenn bestimmte Tasten gedrückt werden.


Search...
Logic
Loops
Math
Text
Variables
Functions
I/O
Display
Time
I/O
Piezo
Play tone with frequency 1000 Hz for 500 milliseconds
Stop playing tone
Buttons
When button left gets pressed
When button left gets held for 500 milliseconds

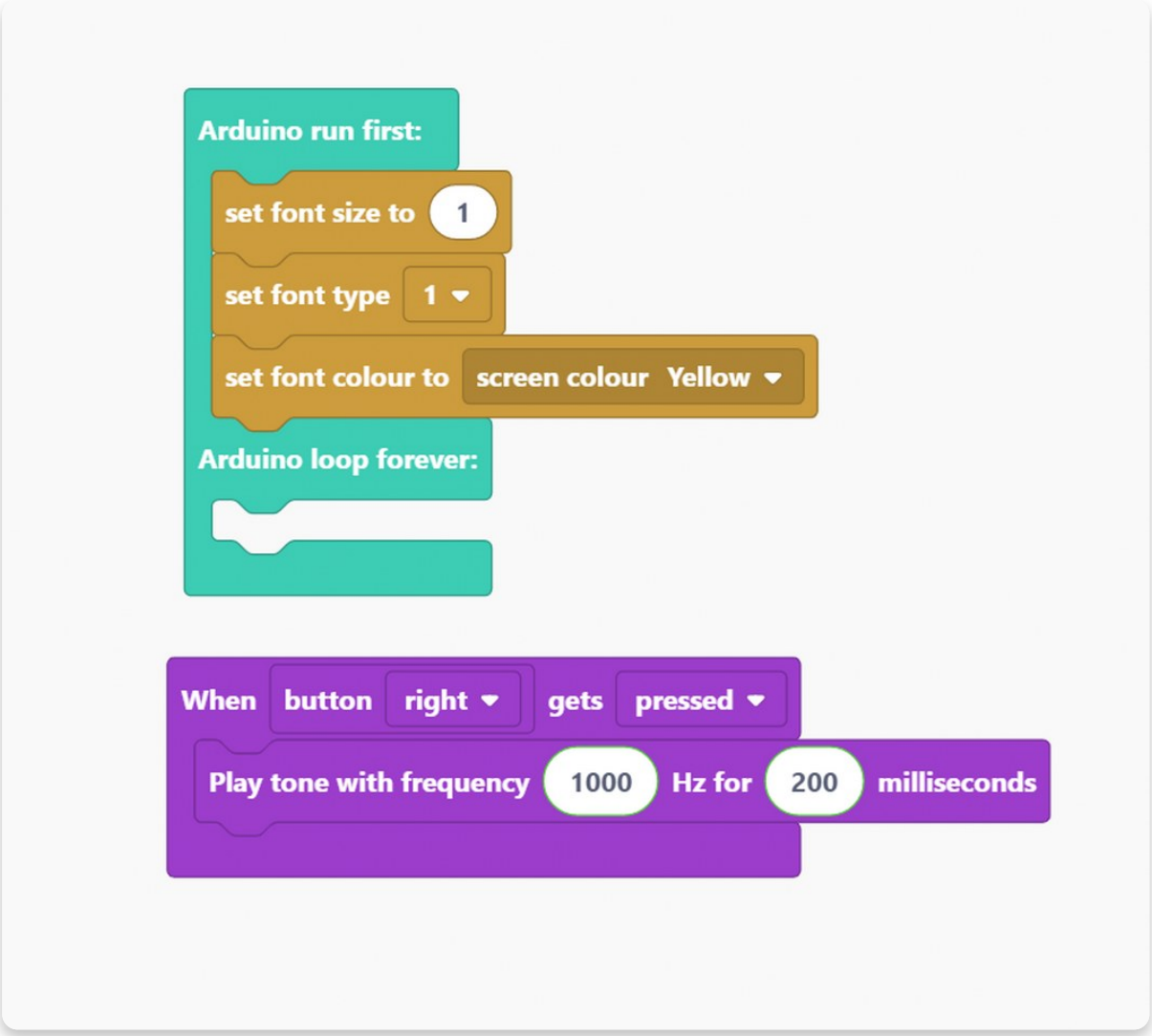


Wir stellen dir einen neuen Block vor. Er hat die Bezeichnung "**Ton mit Frequenz von 1000 Hz für 500 Millisekunden abspielen**" - auf englisch: "Play tone with frequency 1000 Hz for 500 milliseconds"

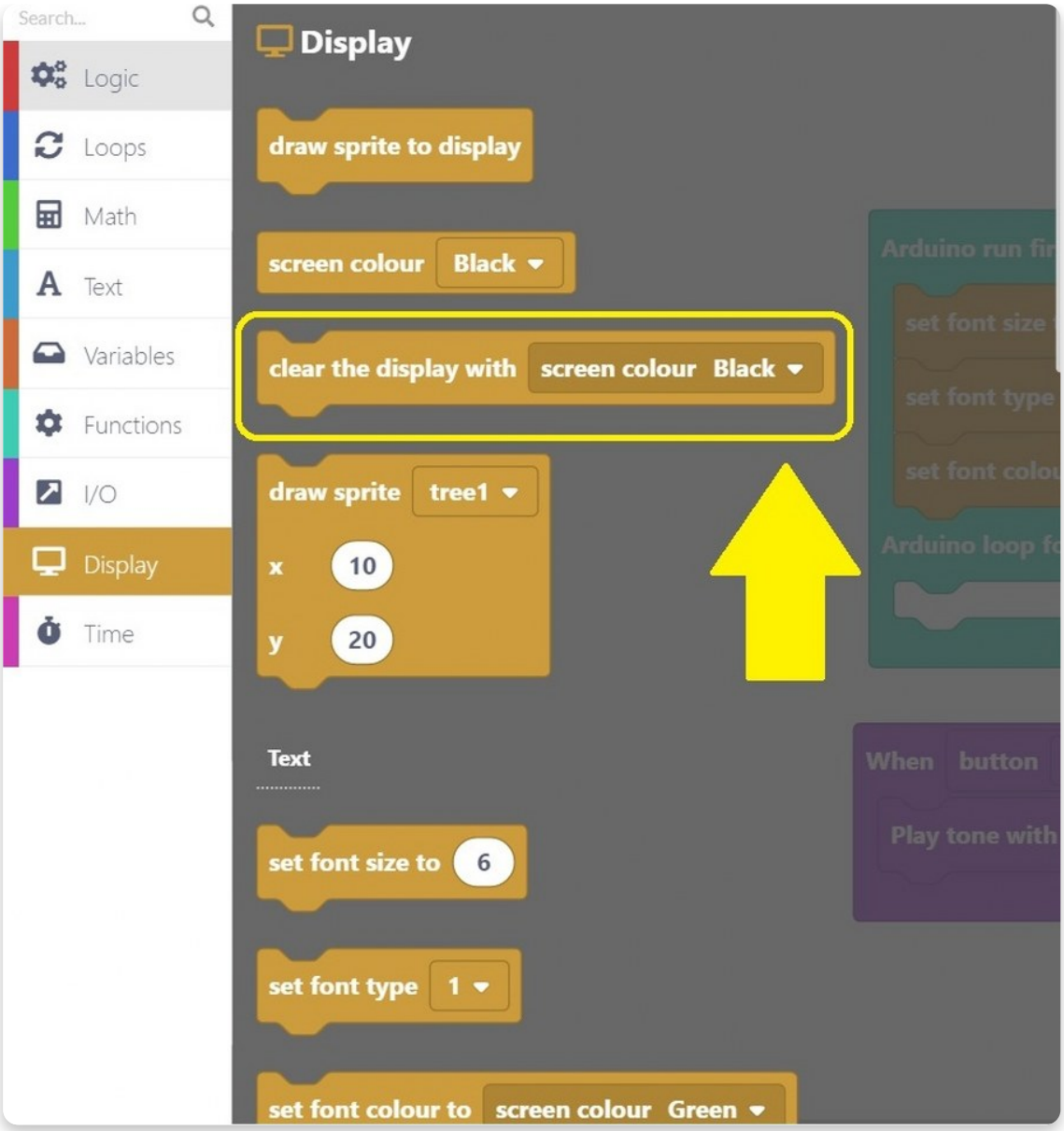


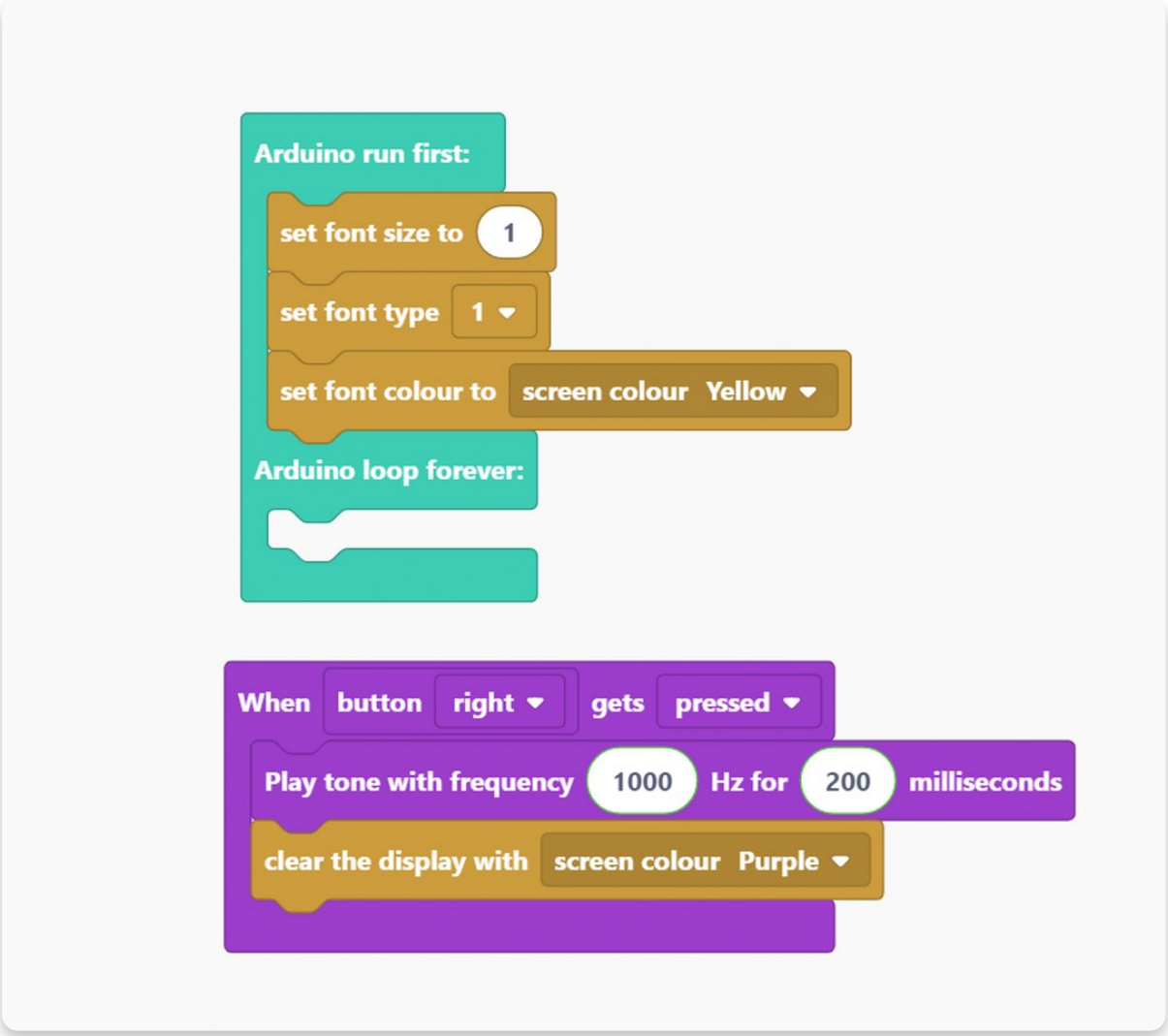
Klicke auf den Block und ziehe ihn auf den Zeichenbereich.

Wir haben die **Millisekunden** auf **200** geändert, aber du kannst den Ton so lang oder kurz machen, wie du magst.

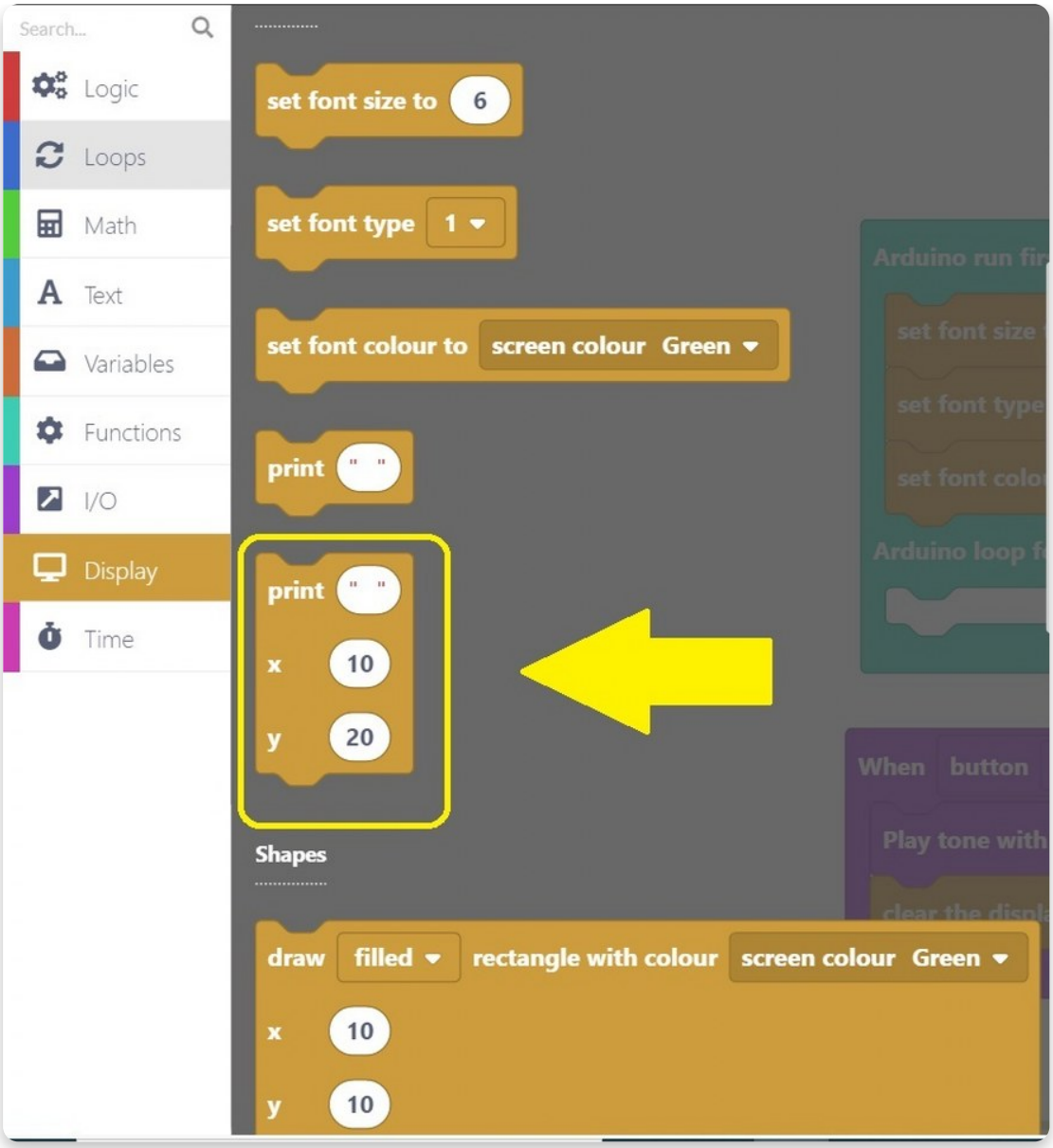


Wir löschen die Anzeige in lila, sobald wir die rechte Taste drücken.

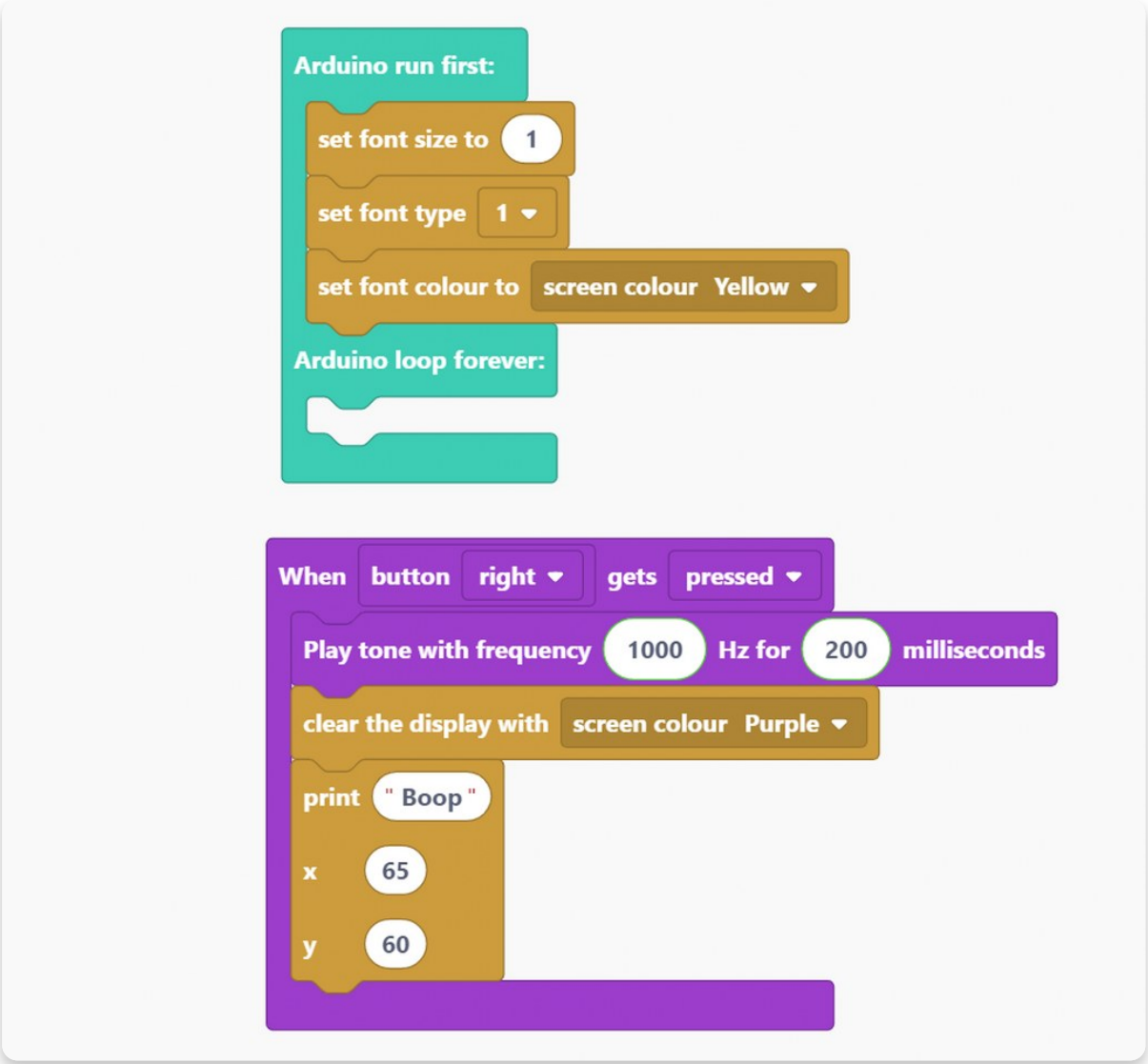

Search...
Logic
Loops
Math
Text
Variables
Functions
I/O
Display
Time
Display
draw sprite to display
screen colour Black
clear the display with screen colour Black
draw sprite tree1
x 10
y 20
Text
set font size to 6
set font type 1
set font colour to screen colour Green
When button
Play tone with



Du kannst auswählen, was auf dem Display angezeigt werden soll, sobald die Taste gedrückt wird.

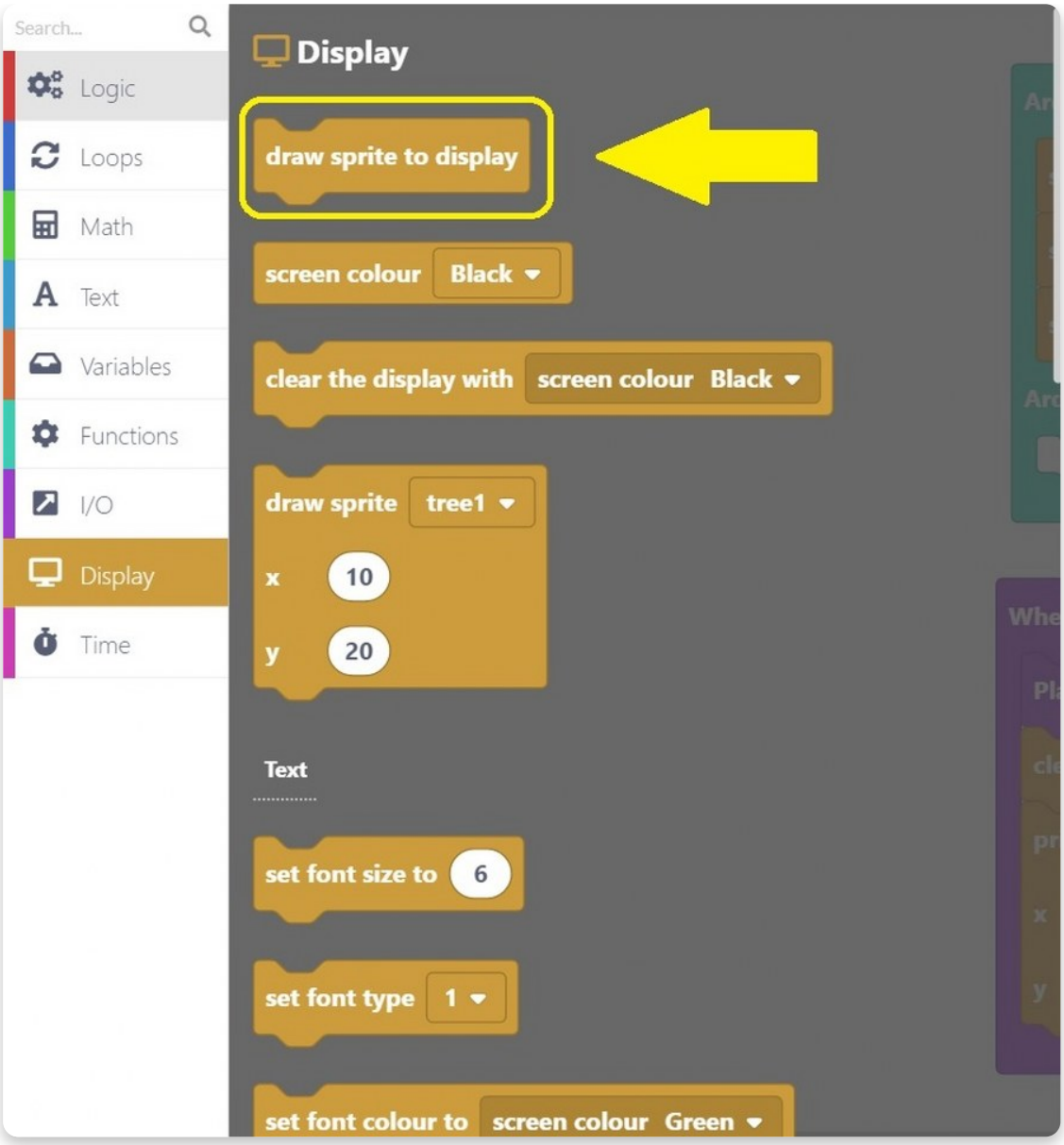


Zum Beispiel kann der Bildschirm "**boop**" anzeigen, wenn wir die **rechte** Taste drücken.

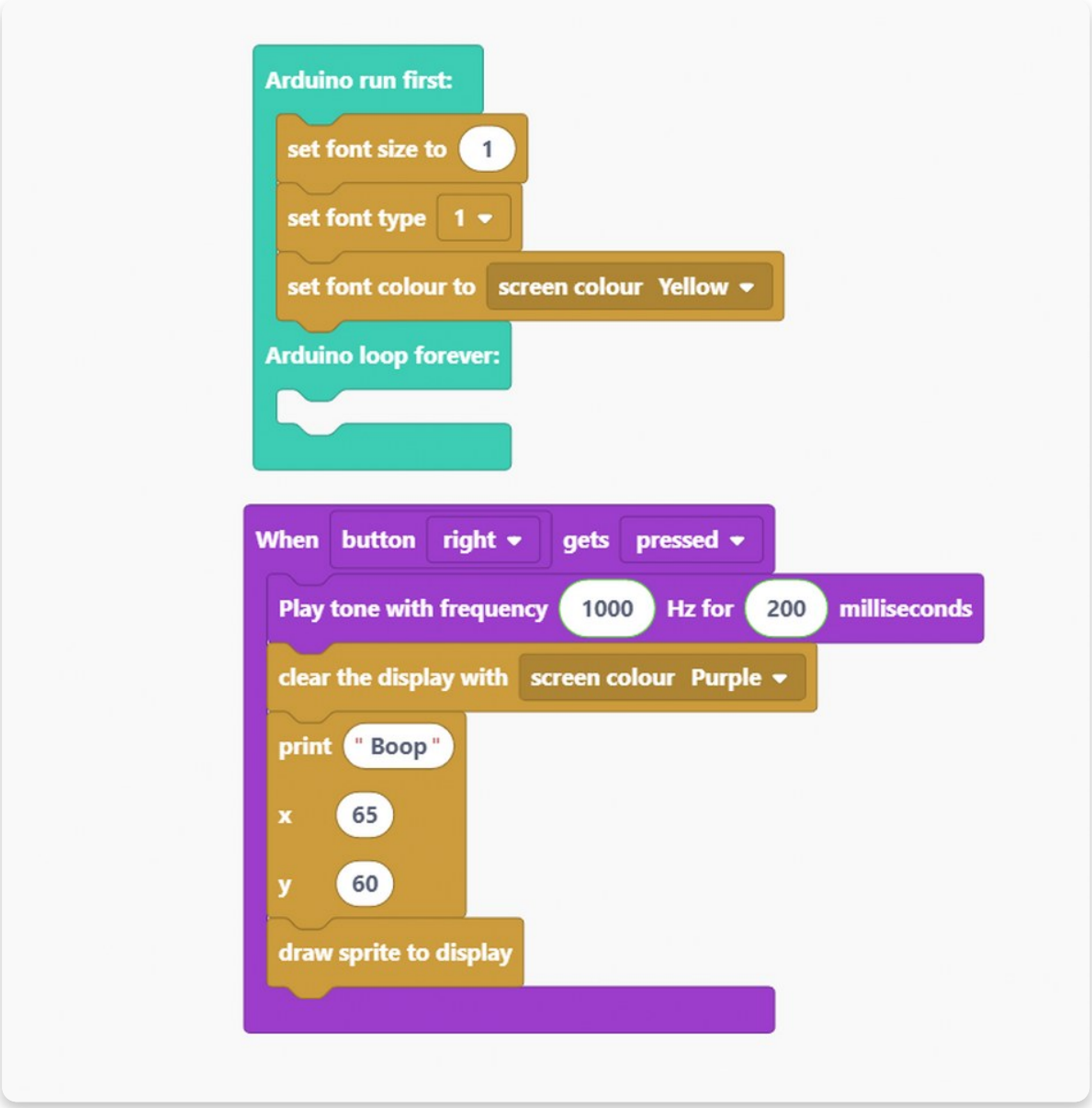


Vergiss die Koordinaten nicht!
Setzen wir x auf 65 und y auf 60.

Und zum Schluss ziehe den "**draw sprite to display**"-Block auf den Zeichenbereich damit das Programm funktioniert.

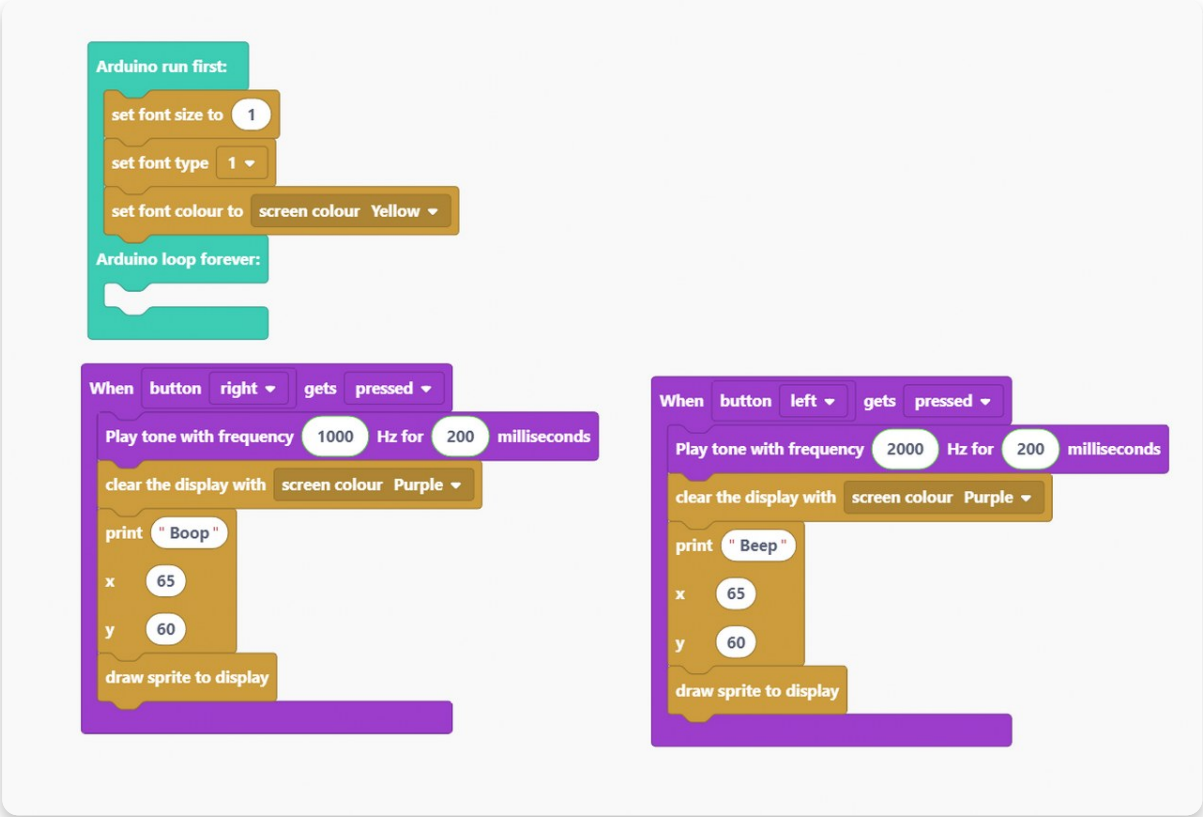


Erste Taste - FERTIG!



Da wir die gleichen Blöcke öfter benötigen, können wir sie einfach duplizieren.

Lass uns eine Taste nach der anderen programmieren.



Der erste neue Block wird ausgeführt, wenn eine **linke Taste** gedrückt wird.

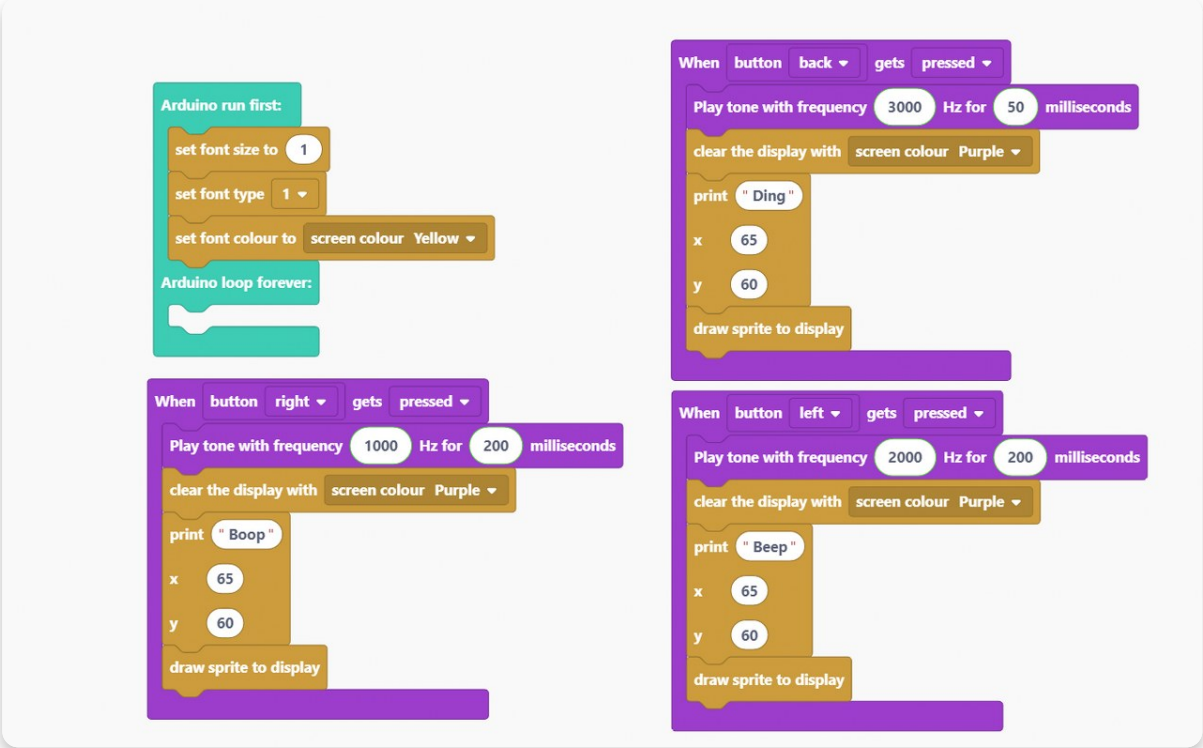
Wir haben uns entschieden, diesmal einen Ton mit einer Frequenz von 2000 Hz abzuspielen, aber wir haben die gleiche Zeitspanne beibehalten, für die wir diesen Ton abspielen werden.

Die Farbe des Bildschirms bleibt lila, aber wir zeigen diesmal "**Beep**" an.

Duplizieren wir den Block noch einmal.

Jetzt verwenden wir die Zurück-Taste und verwenden einen Ton mit einer Frequenz von 3000 Hz für 50 Millisekunden. In der Zwischenzeit wird auf dem Bildschirm "Ding" angezeigt.

Beachte, dass wir die Koordinaten für alle Ausgaben gleich gelassen haben.



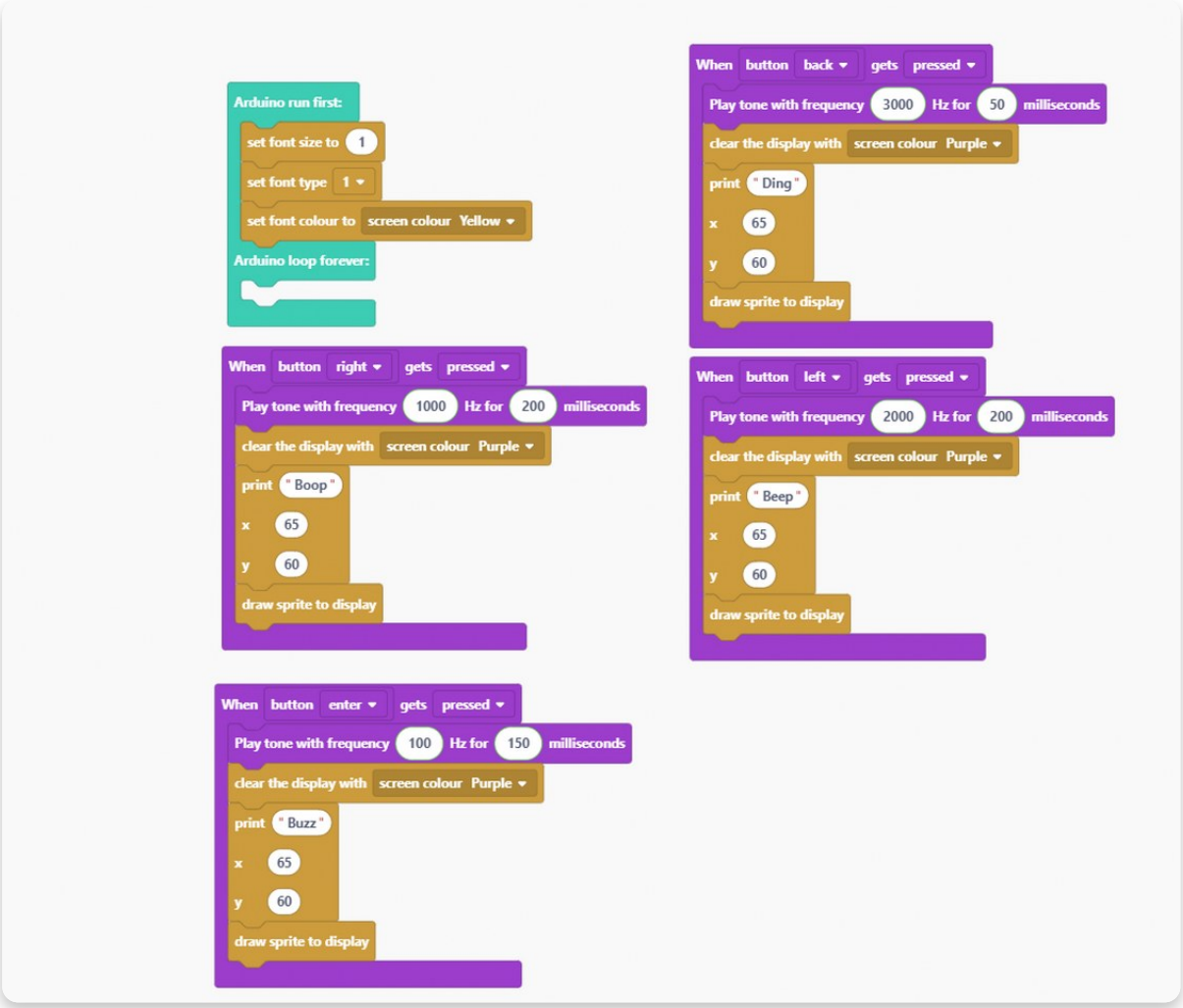
Ja, du hast es richtig erraten!

Es ist an der Zeit, weitere Blöcke zu duplizieren.

Die letzte Taste, die wir verwenden werden, ist die Eingabetaste.

In diesem Programm werden wir nichts beim Loslassen dieser Taste machen.

Wenn du die Enter-Taste drückst, spielt der Buzzer **150 Millisekunden** lang einen Ton mit einer Frequenz von **100 Hz**. Während der Ton abgespielt wird, erscheint auf dem Bildschirm die Meldung "**Buzz**".



Toll!

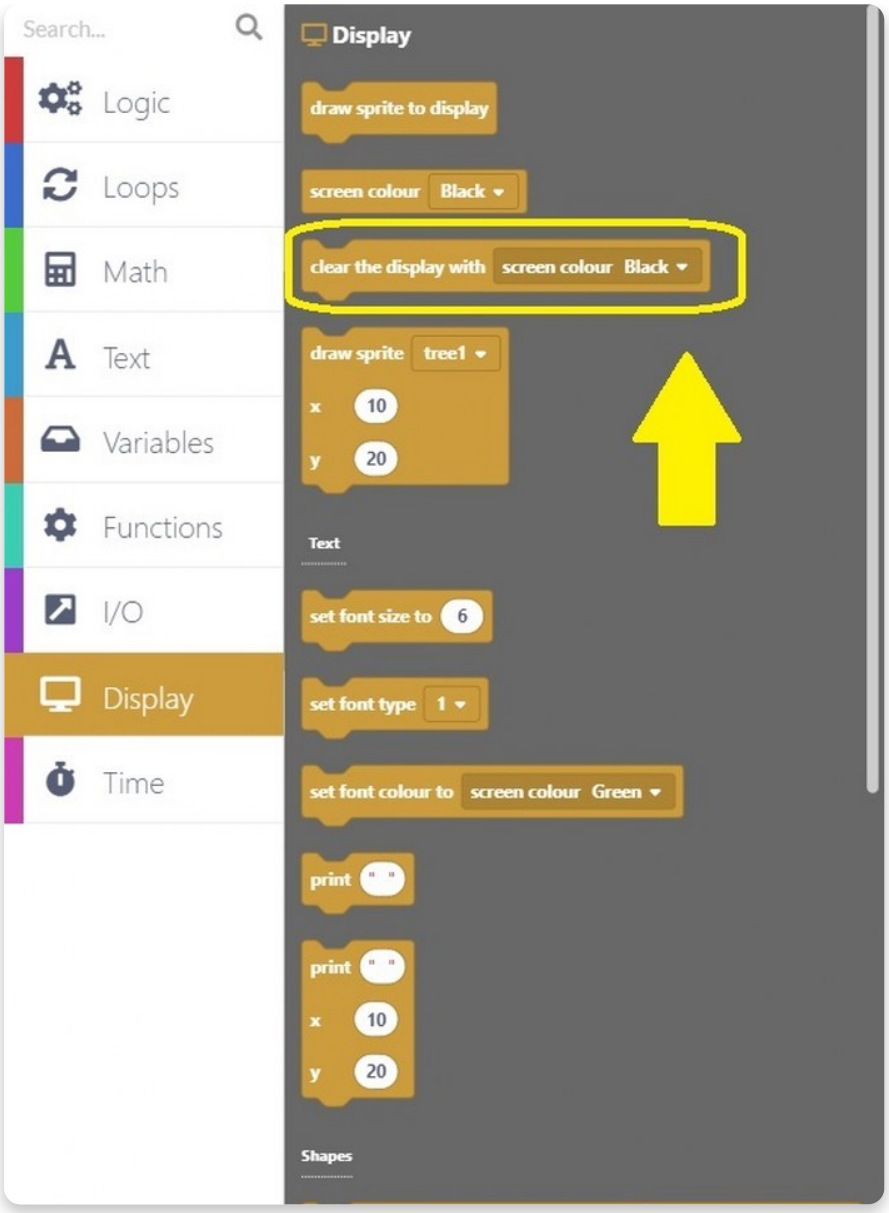
Wir haben alle vier Tasten von Chatters erster Tastenreihe benutzt.

Wir sollten etwas mit dem Bildschirm machen, sobald Chatter eingeschaltet wird. Das Beste, was wir tun können, ist zu schreiben, was als nächstes auf dem Bildschirm passieren wird.

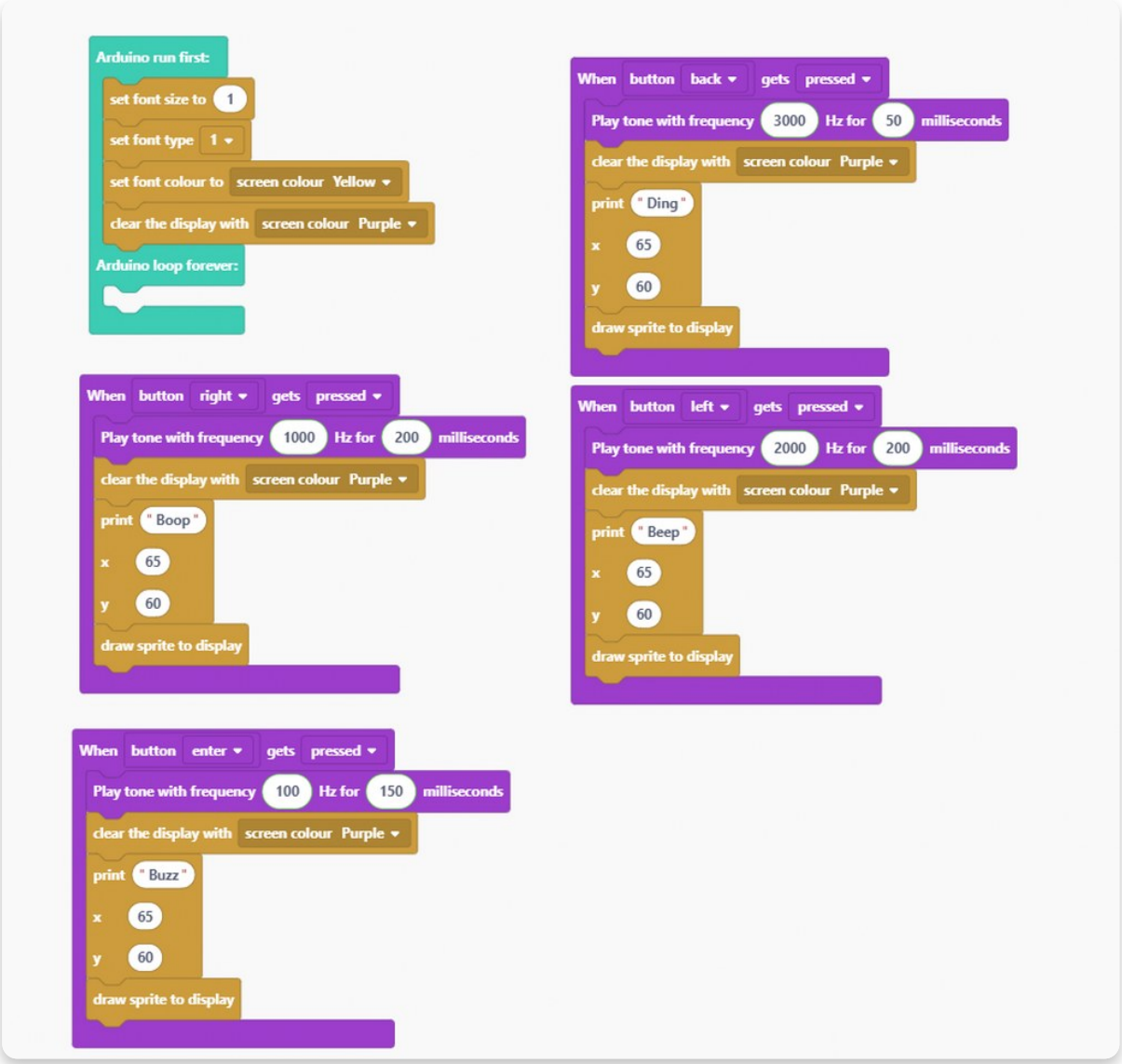
Dazu müssen wir in den **Abschnitt Display** wechseln und einige Blöcke verwenden.

Als Erstes müssen wir den Bildschirm mit seiner aktuellen Farbe füllen damit der bisher angezeigte Text verschwindet.

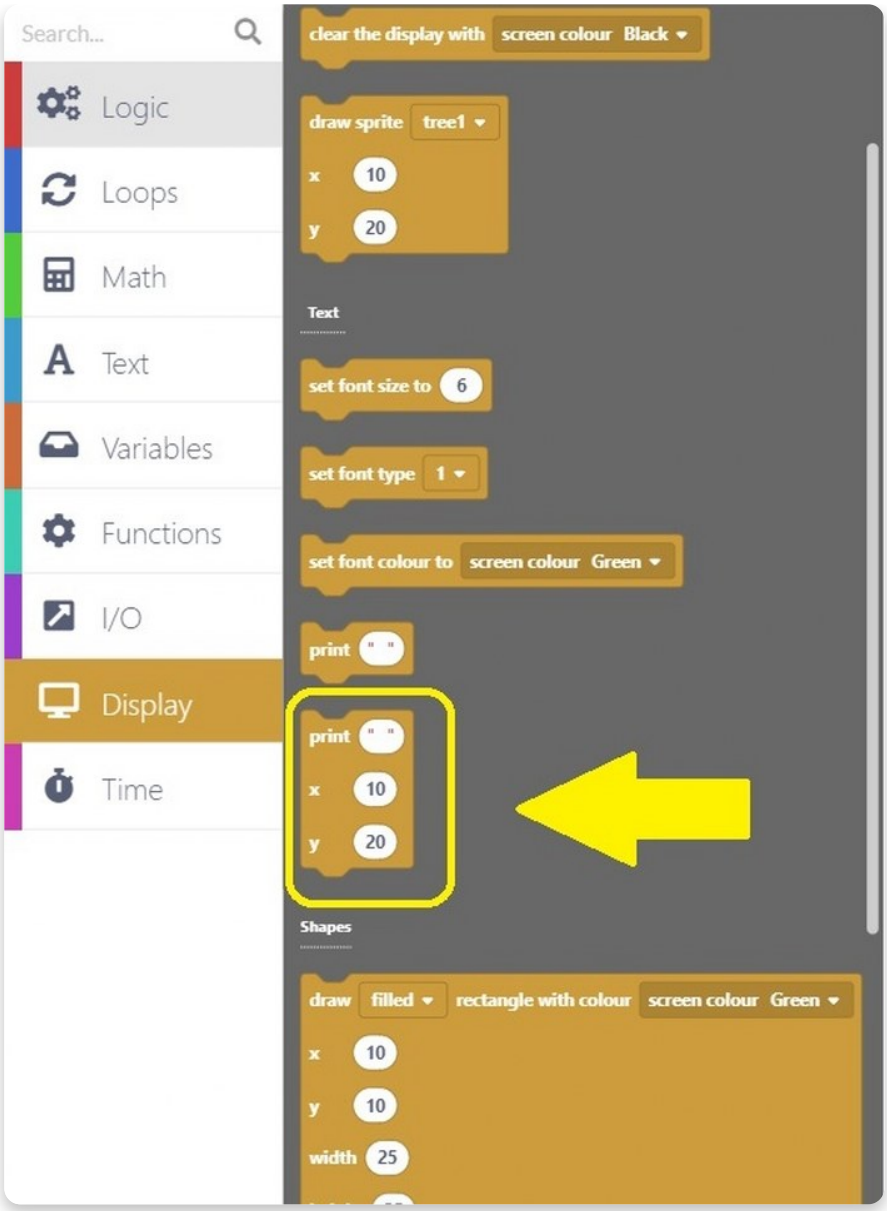
Dazu verwenden wir diesen Block:



Dieser Block muss in den Bereich "**Arduino run first**" des Hauptblocks gezogen werden.



Da wir den Bildschirm gelöscht haben, ist es nun an der Zeit, etwas darauf zu schreiben.



Ziehe den eingekreisten Block in den Abschnitt "Arduino run first".

Du benötigst zwei dieser Blöcke, damit der Text schön in der Mitte des Bildschirms platziert wird.

Der Text, der auf dem Bildschirm angezeigt wird, sobald du deinen Chatter einschaltest, lautet also: "**Press one of the top row buttons!**" ("Drücke eine der Tasten der oberen Reihe!"). Im Grunde ist dies wie eine Anleitung auf dem Bildschirm.

Und wir hoffen, du weißt, was am Ende stehen muss. Der "**draw sprite to display**"-Block!

Du hast es geschafft!
Herzlichen Glückwunsch!

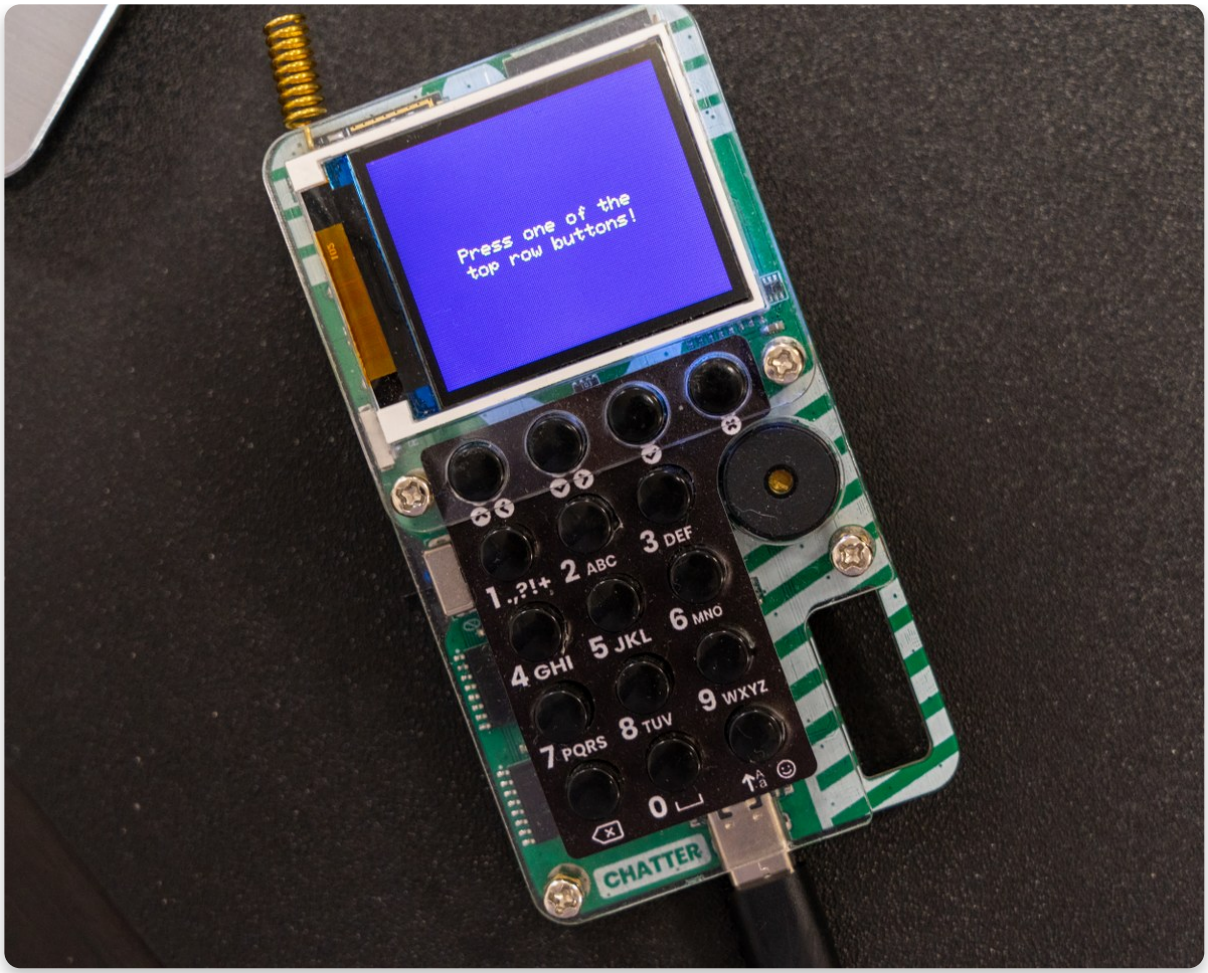
Drücke den großen roten Ausführen-Knopf ("Run"), warte, bis der Code kompiliert ist und probiere ihn aus.

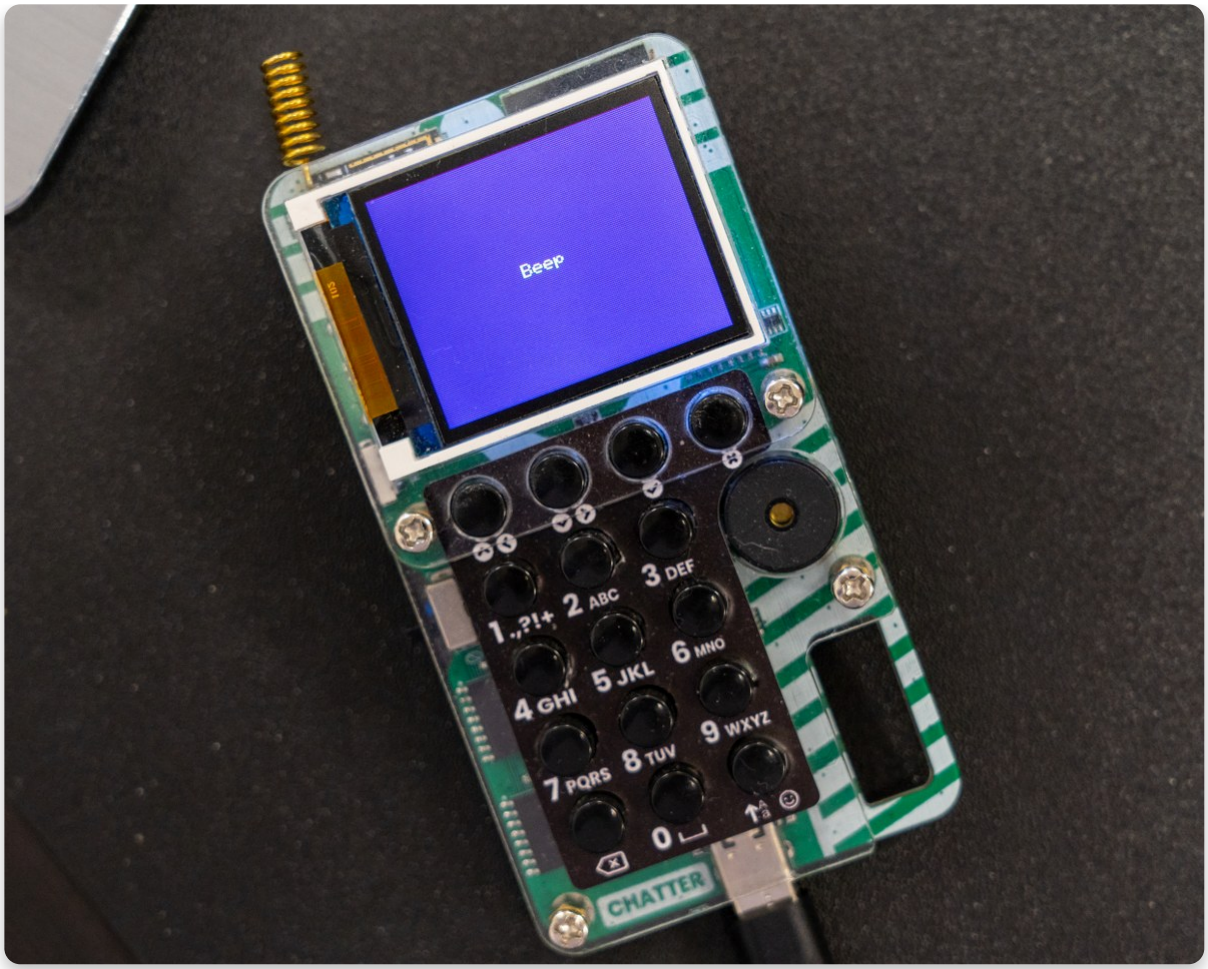
Sobald der Code kompiliert ist, sollte dein Chatter neu starten und der Bildschirm wird gelb mit dem lila Slogan "Drücke einen der Knöpfe in der oberen Reihe!".

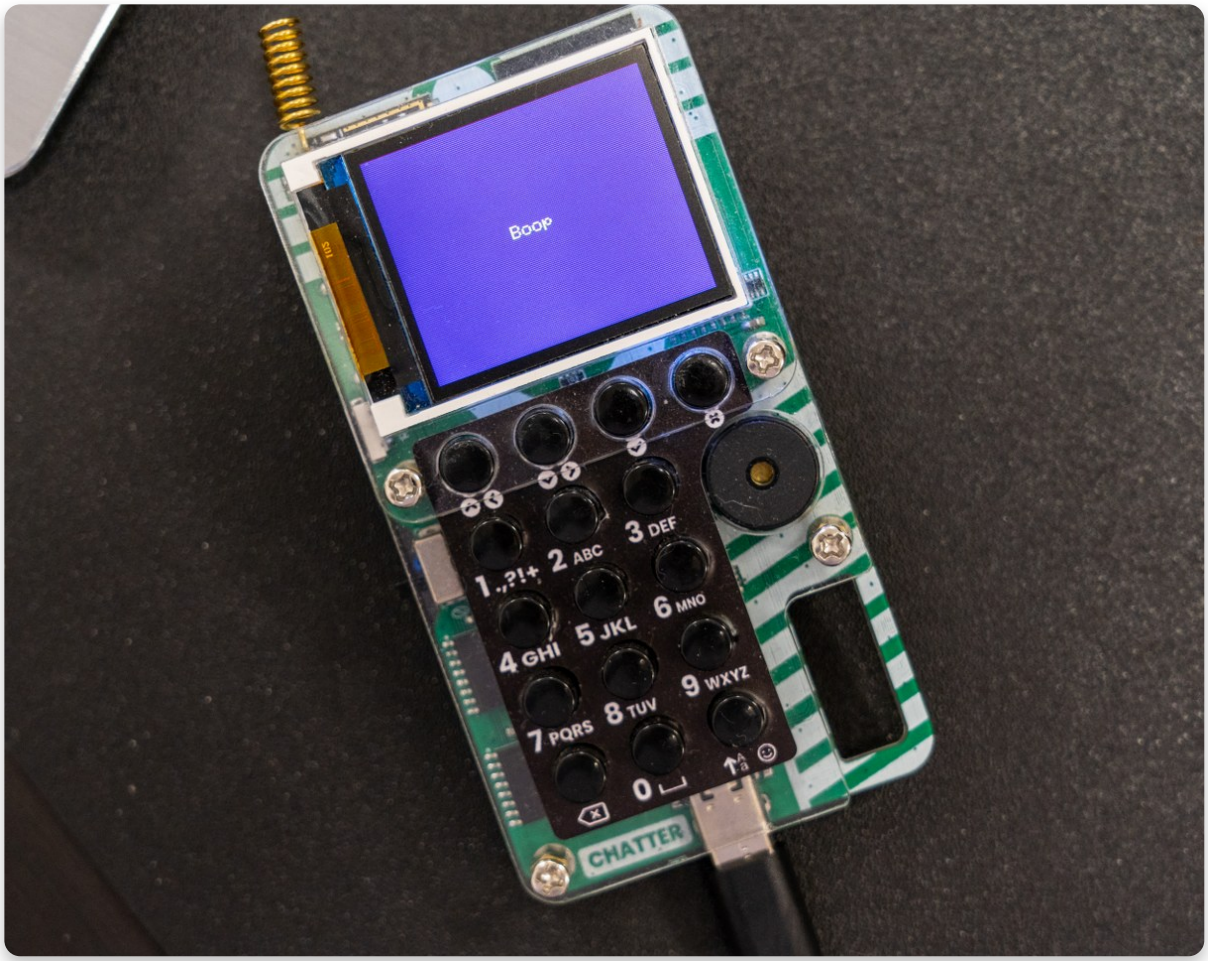
Wenn du dies tust und auf eine der Tasten drückst, werden Töne vom Buzzer abgespielt. Je nachdem, auf welche Taste du gedrückt hast, ist die Tonfrequenz unterschiedlich und die Dauer des Tons ändert sich.

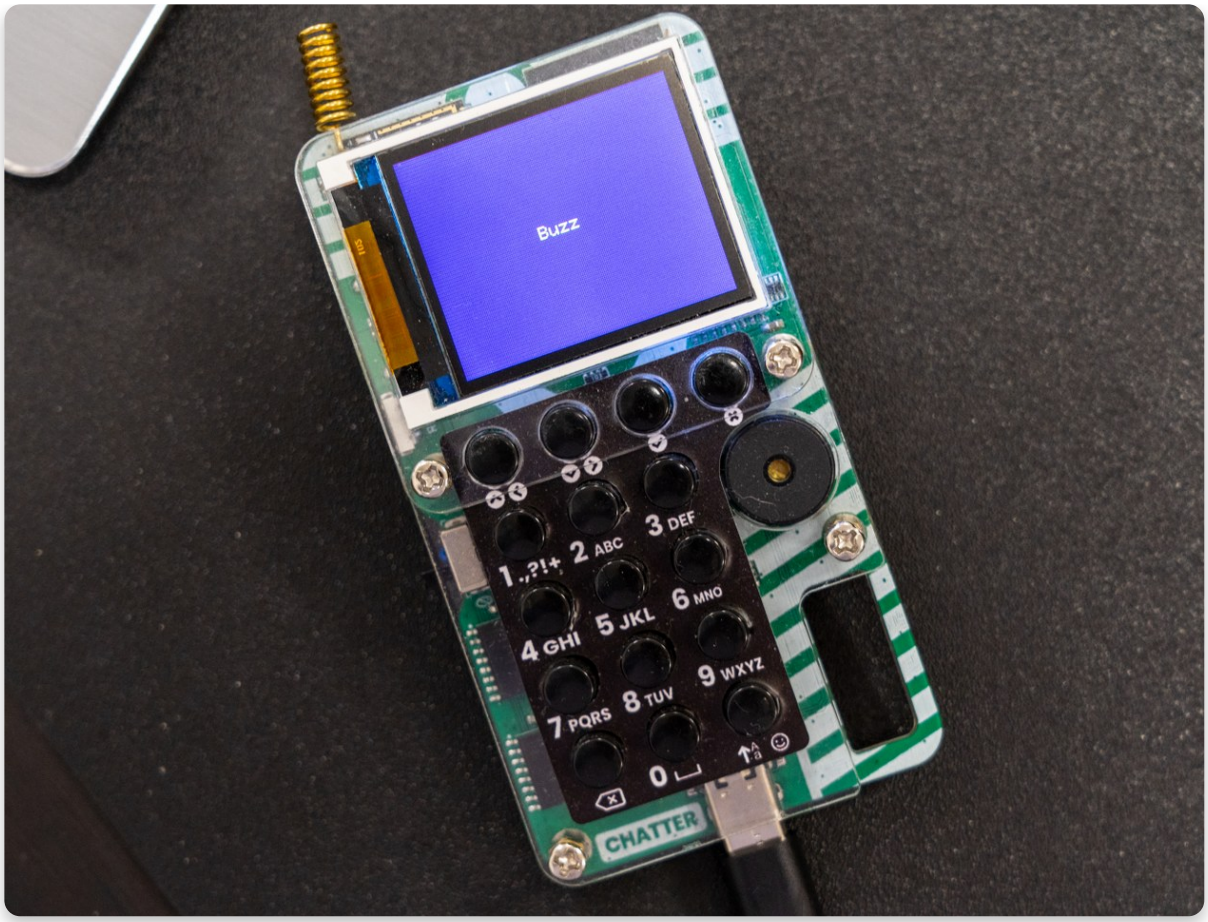
Außerdem wird mit jeder Taste ein anderer Text auf dem Bildschirm angezeigt.

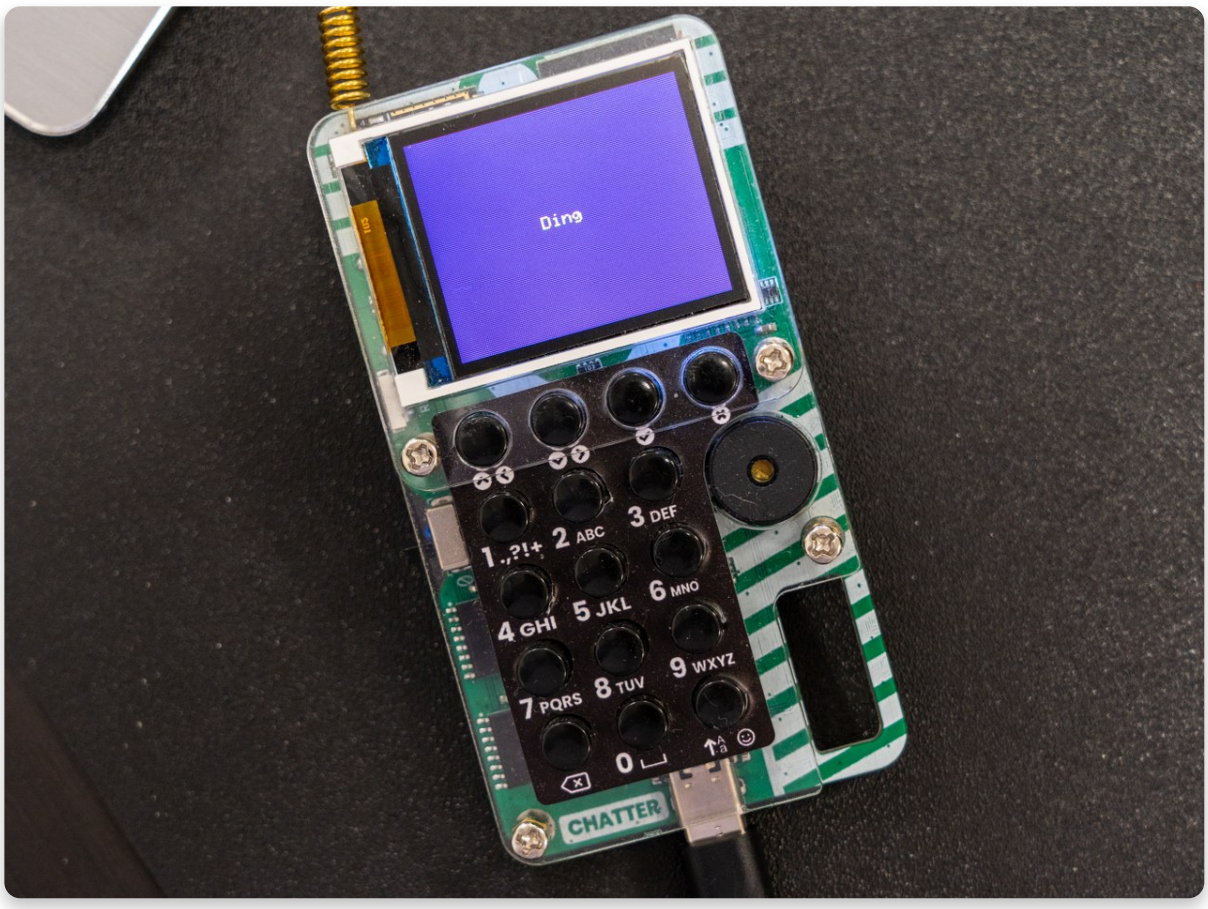
Auf den folgenden Fotos kannst du sehen, wie der Bildschirm jeweils aussehen sollte:


Press one of the
top row buttons!
1 .,?!+
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
CHATTER


Beep
1 .,?!+
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
CHATTER


Boop
1 .,?!+
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
CHATTER


Buzz
1 .,?!+ 2 ABC 3 DEF
4 GHI 5 JKL 6 MNO
7 PQRS 8 TUV 9 WXYZ
0
CHATTER

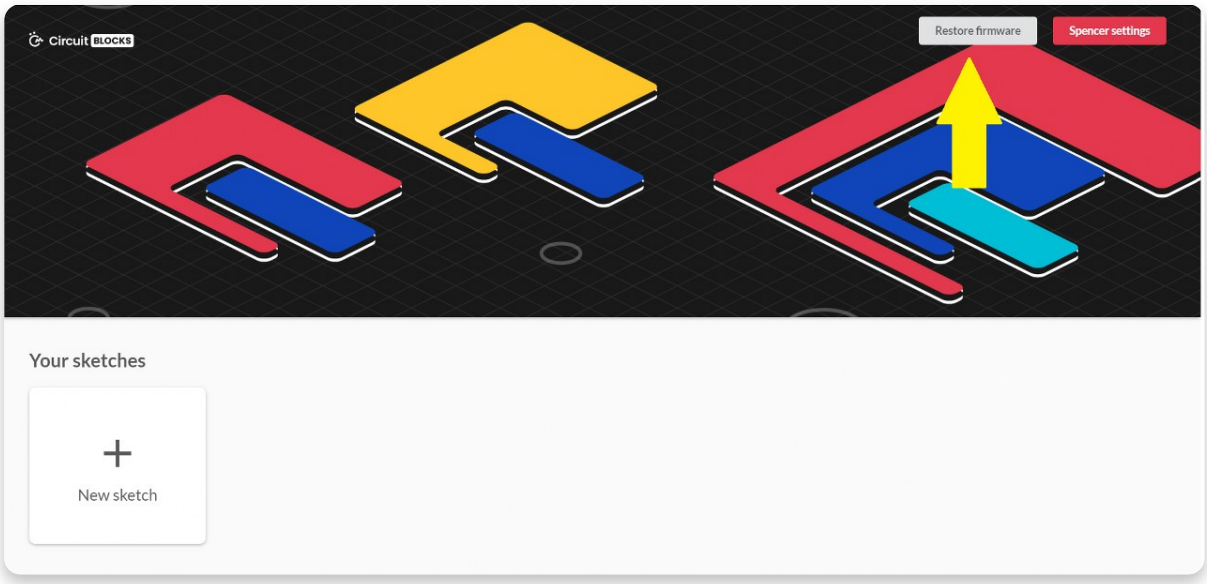

Ding
1 .,?!+ 2 ABC 3 DEF
4 GHI 5 JKL 6 MNO
7 PQRS 8 TUV 9 WXYZ
0
CHATTER

Loading...

Wiederherstellen der Basis-Firmware von Chatter

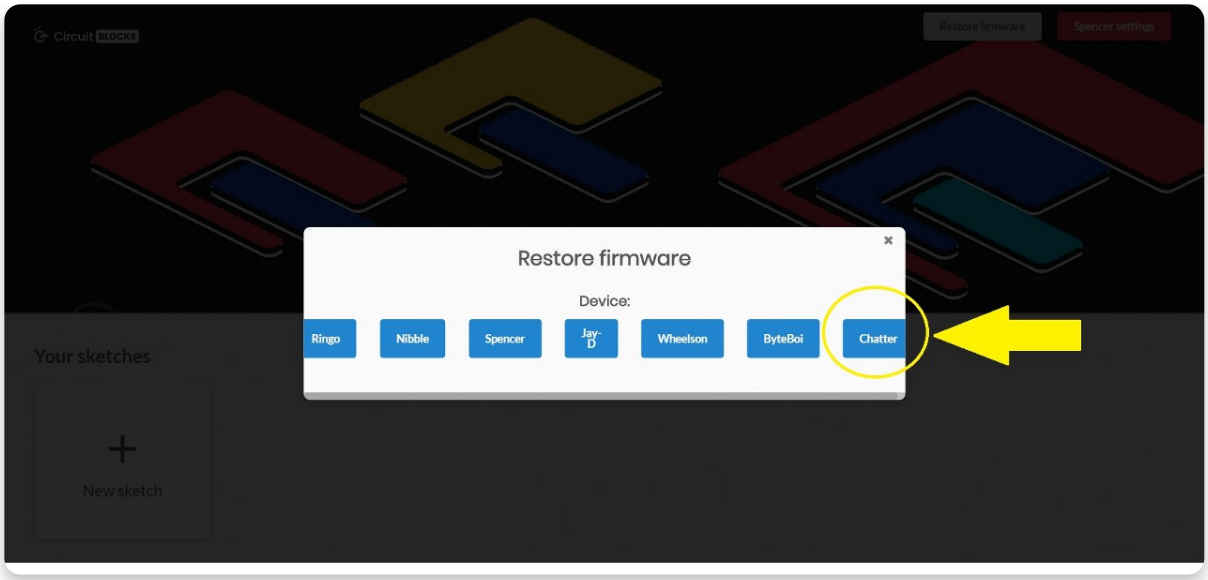
Wenn du mit dem Programmieren fertig bist und deinen Chatter einfach wieder "normal" machen willst, musst du seine Basis-Firmware wiederherstellen.

Das ist ganz einfach: Schließe deinen Chatter an den USB-Anschluss deines Computers an und drücke die Schaltfläche "Firmware wiederherstellen" (englisch: Restore firmware) oben rechts.



Du wirst in einem Fenster aufgefordert, das Gerät auszuwählen, für das du die Firmware wiederherstellen möchtest.

Wähle hier natürlich **Chatter**.



Warte ein paar Sekunden, und dein Chatter wird wieder wie gewohnt funktionieren.

Du musst dies immer dann tun, wenn du mit der Programmierung deines Chatters fertig bist und wenn du möchtest, dass er seine ursprünglichen Funktionalität zurückerhält.

Wie geht's weiter?

Du hast das Ende unseres Chatter-Tutorials erreicht, Herzlichen Glückwunsch!

Wir hoffen, du bist genauso gespannt wie wir auf die Zukunft von Chatter, denn es gibt so viele coole Dinge, die wir in den zukünftigen Firmware- und CircuitBlocks-Updates damit machen wollen.

In der Zwischenzeit kannst du auf eigene Faust weiterforschen und uns zeigen, was du mit deinem Chatter gemacht hast, indem du es im CircuitMess-Community-Forum oder über unseren Discord-Kanal teilst.

Wenn du Hilfe mit deinem Gerät brauchst, wende dich wie immer an uns über contact@circuitmess.com, und wir werden dir so schnell wie möglich helfen.

Vielen Dank und mach' weiter so!