VORBEMERKUNG

Die folgenden Seiten sind in der Projektphase entstanden.

Schülerinnen und Schüler haben Fragen und Schwierigkeiten beim Programmieren ihres Projektes vorgestellt. Dazu habe ich die folgenden Anleitungen speziell für deren Projekt geschrieben, aber gleichzeitig auch als Anregung für die anderen Teams.

HILFEKARTEN

Spiel geschafft? - Übergabe zwischen Screens

Benötigte Komponenten:

1. Screen 1: ListView „lvSpielauswahl“
2. Screen 2 („Spiel_1“): Button „btGeschafft“

Auf dem ersten Screen ist eine Auswahlliste für verschiedene Aufträge (bzw. Spiele) der App. Wenn man einen Auftrag erfolgreich erledigt hat, soll das in der Liste sichtbar sein.

Für jeden Auftrag wird ein neuer Screen gestartet, der dann den Wert „true“ (wahr) zurückgeben soll, wenn der Nutzer / die Nutzerin erfolgreich war.

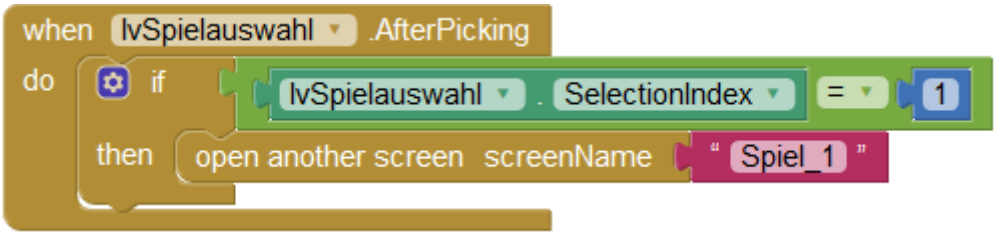
Hier ist nur die Programmierung für einen Auftrag enthalten. Ihr schafft es bestimmt, weitere Aufträge mit einzubinden. Wenn ihr Fragen dazu habt, meldet euch!

Blöcke Screen 1:

Zunächst erstellen wir eine neue Liste, in der die einzelnen Aufträge aufgelistet sind.

Diese wird bei der Initialisierung des Screens der ListView übergeben.

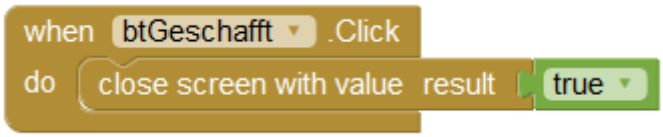
Wird ein Eintrag ausgewählt, soll der entsprechende neue Screen geöffnet werden (hier muss für die weiteren Aufträge ergänzt werden: „else if...“):



Jetzt kümmern wir uns zunächst um den weiteren Screen „Spiel_1“.

Blöcke Screen 2:

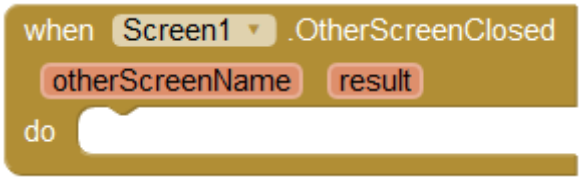
Wenn der Auftrag erledigt ist, wird der Screen geschlossen und der Wert „true“ übergeben. Hier im Beispiel erfolgt das, wenn auf den Button btGeschafft geklickt wird. Je nach Auftrag kann der Block auch in einem anderen Ereignis eingebunden werden.



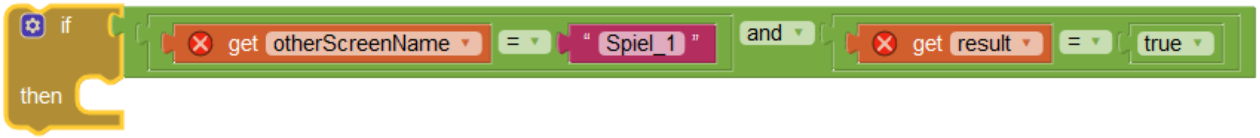
Der Screen wird geschlossen und der Nutzer / die Nutzerin sieht wieder den ursprünglichen Screen mit der Auswahlliste.

Blöcke Screen 1:

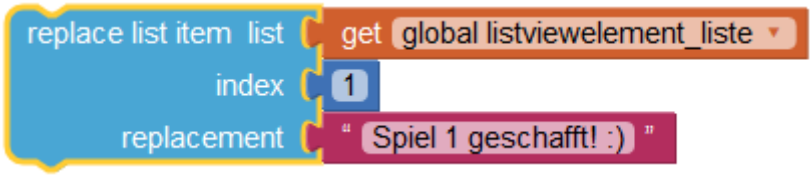
Das Ereignis, das ausgelöst wurde heißt „when … OtherScreenClosed“. Es bekommt sowohl den Name des anderen Screens (otherScreenName) als auch den Übergabewert (result) mit übergeben.



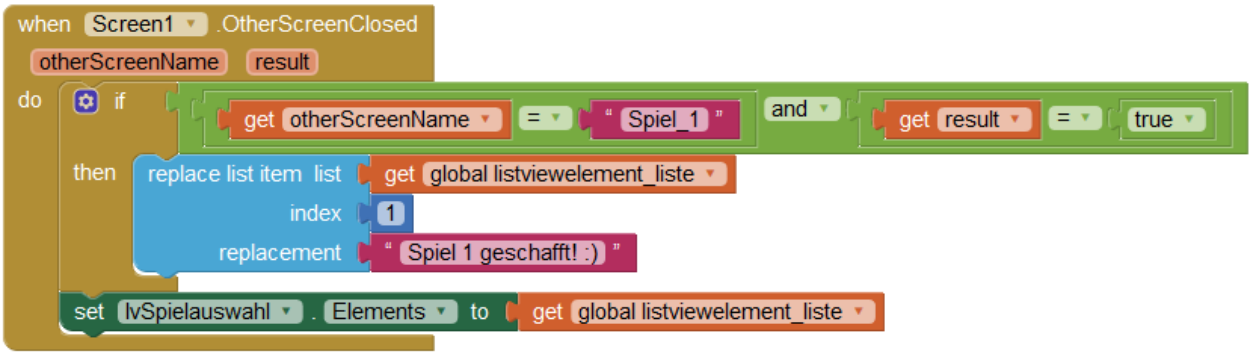
In einer Entscheidung fragen wir ab, welcher Screen geschlossen wurde (also von welchem Auftrag der Nutzer / die Nutzerin zurückkehrt) und ob es erfolgreich war.



Wenn ja, wird der entsprechende Eintrag in der Auswahlliste ersetzt von z.B. „Spiel 1“ zu „Spiel 1 geschafft!“.



Natürlich muss am Ende die veränderte Liste wieder an die ListView zur Anzeige übergeben werden. Insgesamt sehen die Blöcke so aus:

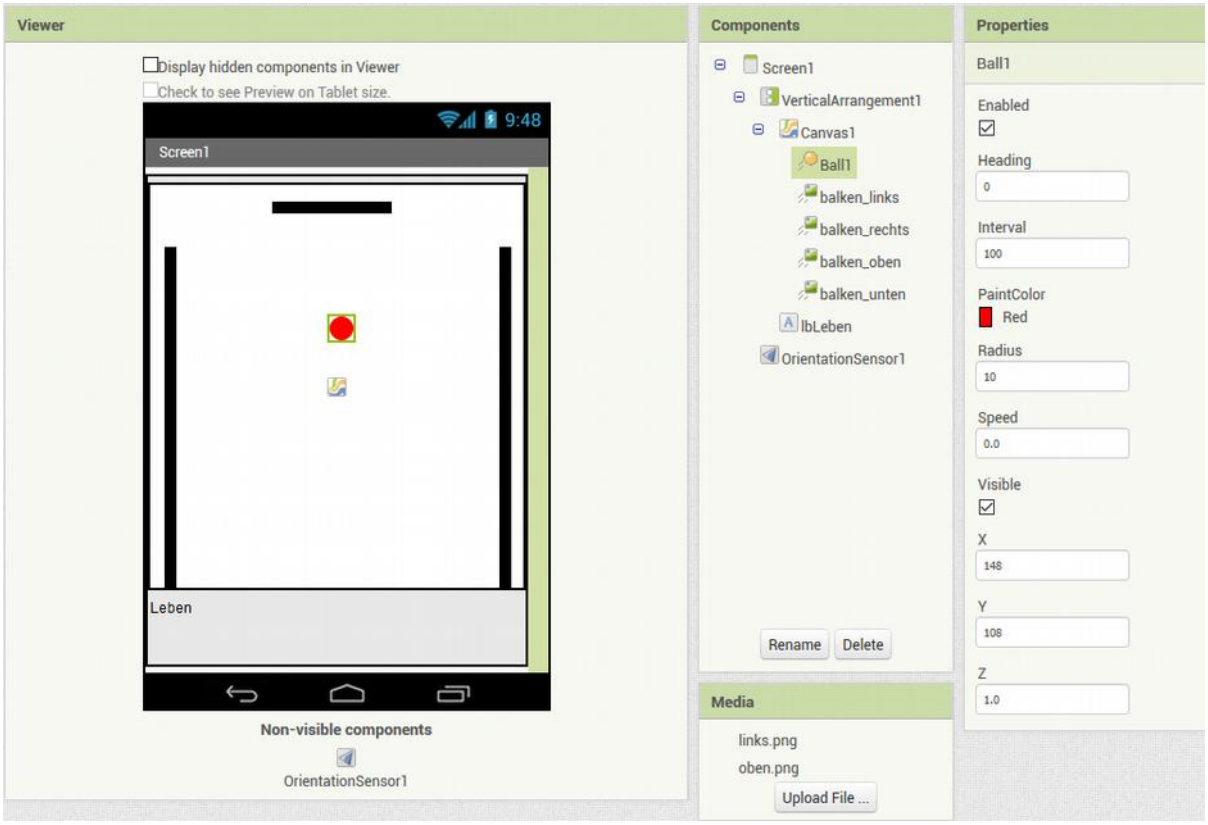


HILFEKARTEN

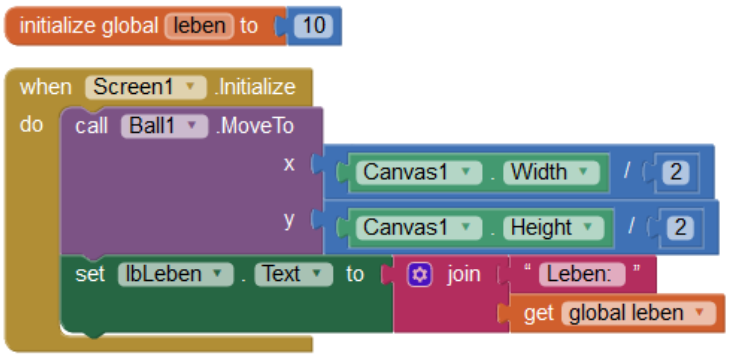
Nutzen des Orientation Sensors

Benötigte Komponenten:

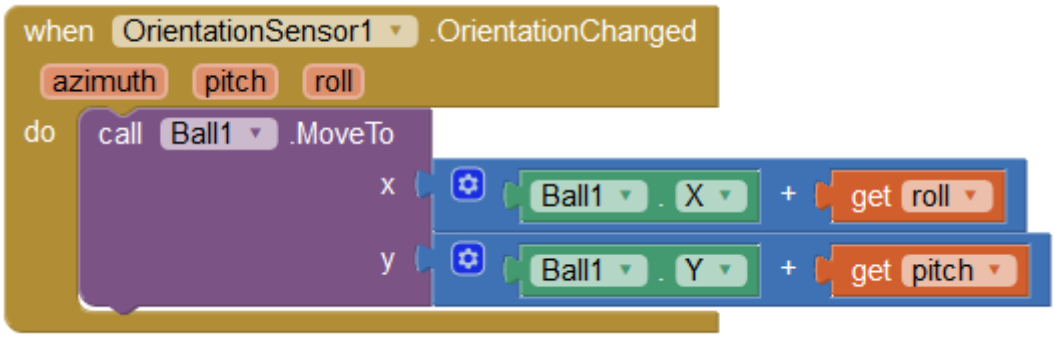
1. Als Layout ein „VerticalArrangement“.
2. Eine Leinwand (Canvas) mit einem Ball und vier ImageSprites (Balken).
3. Ein Label, um die Anzahl der Leben auszugeben.
4. Einen Orientation Sensor.



Bei der Initialisierung des Screens wird der Ball in die Mitte der Leinwand gesetzt und dem Label die Anzahl der Leben mit übergeben.

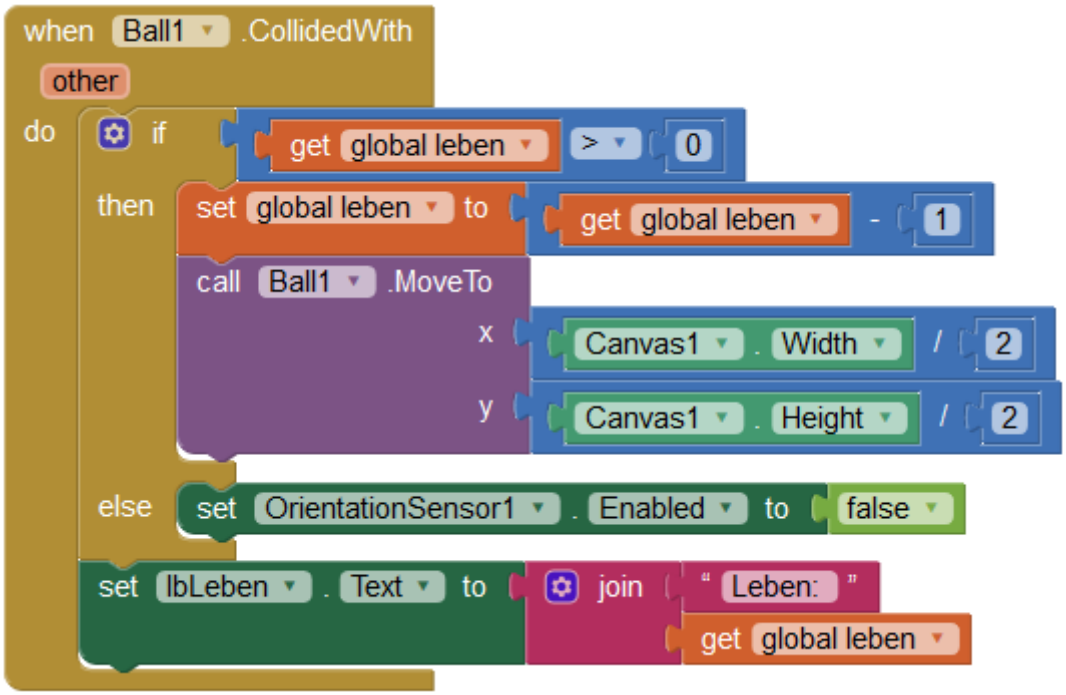


Bewegt sich das Smartphone, wird das Ereignis „when … OrientationChanged“ ausgelöst. Dabei bekommt man drei Werte übergeben. Probiere aus, was passiert, wenn du folgende Blöcke programmierst:



Möchte man noch die Kollision des Balles mit den Balken am Rand programmieren, nutzt man das Ereignis „when … CollidedWith“ des Balles.

Überlege dir zunächst, was in den folgenden Blöcken programmiert wurde und teste es dann aus.

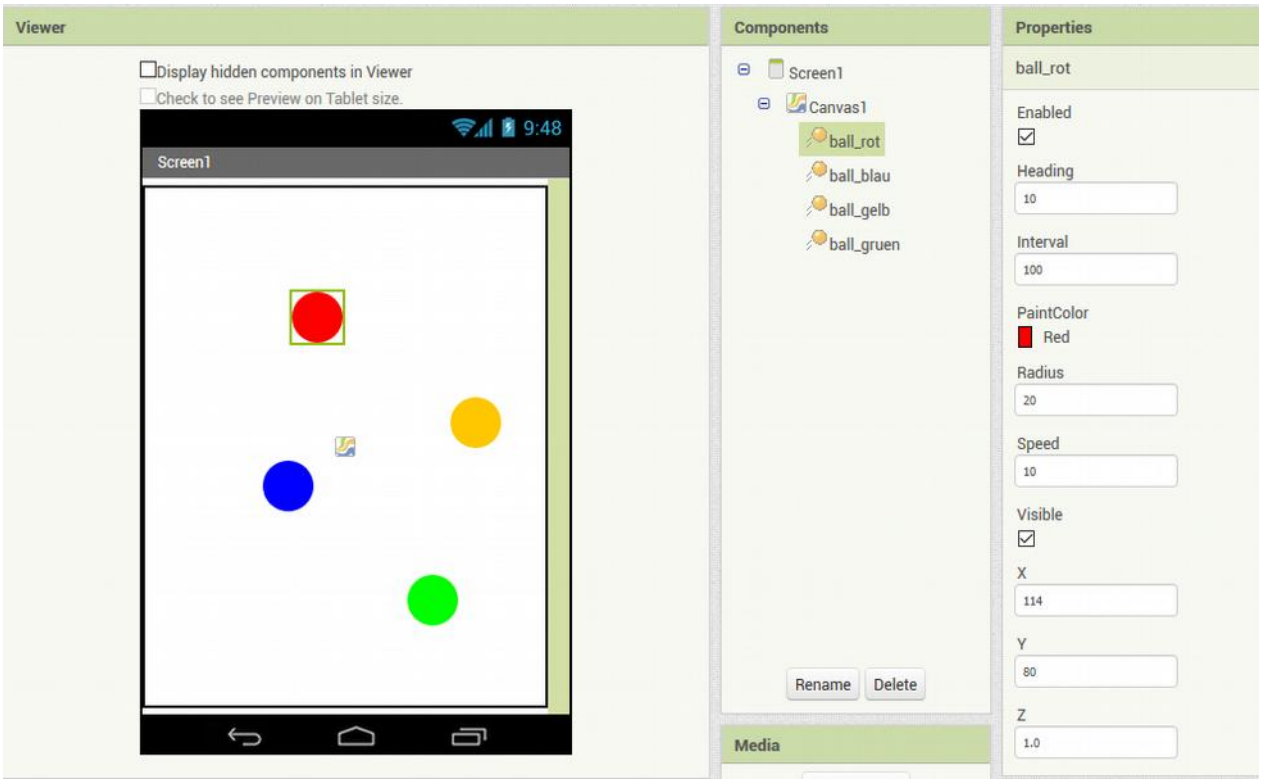


HILFEKARTEN

Kollision von Objekten

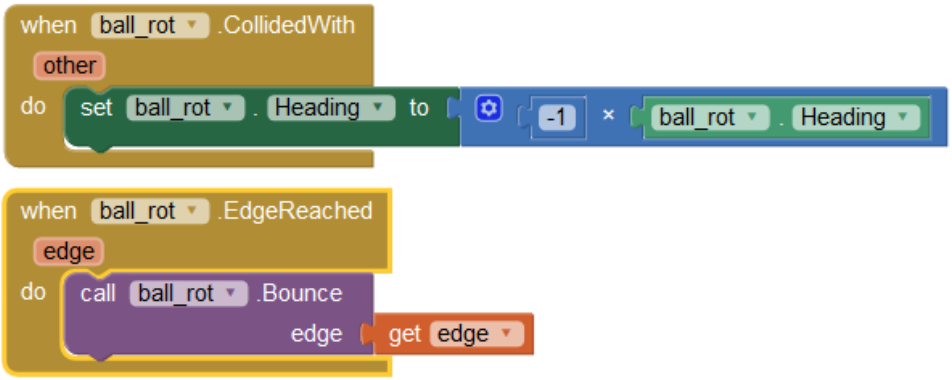
Benötigte Komponenten:

1. Eine Leinwand (Canvas).
2. Vier Bälle mit unterschiedlichen Farben, einer Geschwindigkeit (Speed) von 10 und unterschiedlichen Richtungswinkeln (Heading).



Die Bälle bewegen sich auf dem Screen und sollen voneinander abprallen, d.h. sich beim Zusammenstoß in die Gegenrichtung weiterbewegen. Außerdem sollen sie vom Rand abprallen.

Folgende Blöcke brauchst du für alle vier Bälle:

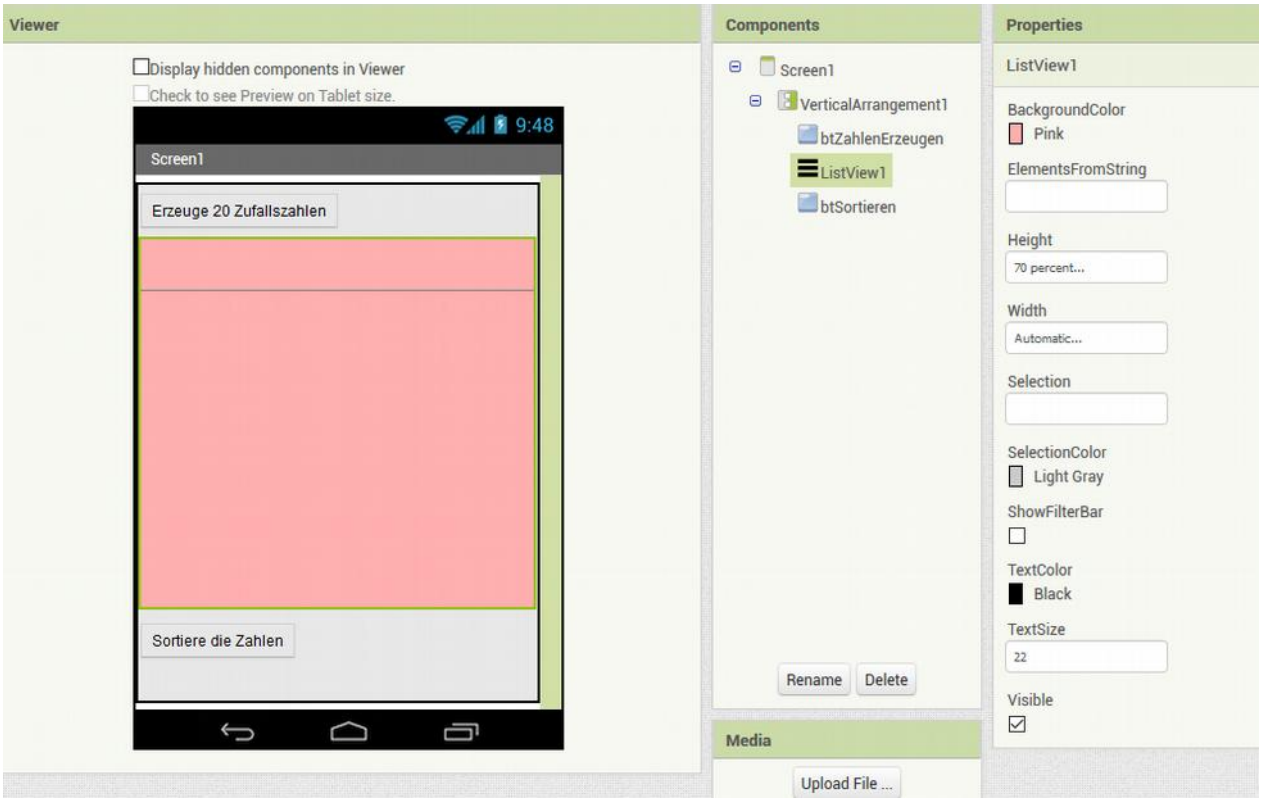


HILFEKARTEN

Sortieren einer Zahlenliste

Benötigte Komponenten:

1. Für das Layout nutzen wir ein VerticalArrangement.
2. Einen Button zum Erzeugen von Zufallszahlen.
3. Eine ListView, in der die Zahlen angezeigt werden.
4. Einen Button zum Sortieren der Zahlen.



Um eine Liste von Zahlen zu sortieren, kann man viele unterschiedliche Ansätze nutzen. Bei einer kleinen Menge kommt es nicht auf das Verfahren an, da selbst zeitaufwendige Verfahren schnell erledigt sind. Deshalb nutzen wir hier jetzt unsere Idee der Maximumsuche.

Wir durchlaufen im ersten Schritt unsere gesamte Liste und suchen nach dem Maximum. Dieses Maximum setzen wir in einer neuen Liste an die erste Stelle und entfernen es aus der ersten Liste. Diese ist nun einen Eintrag kürzer.

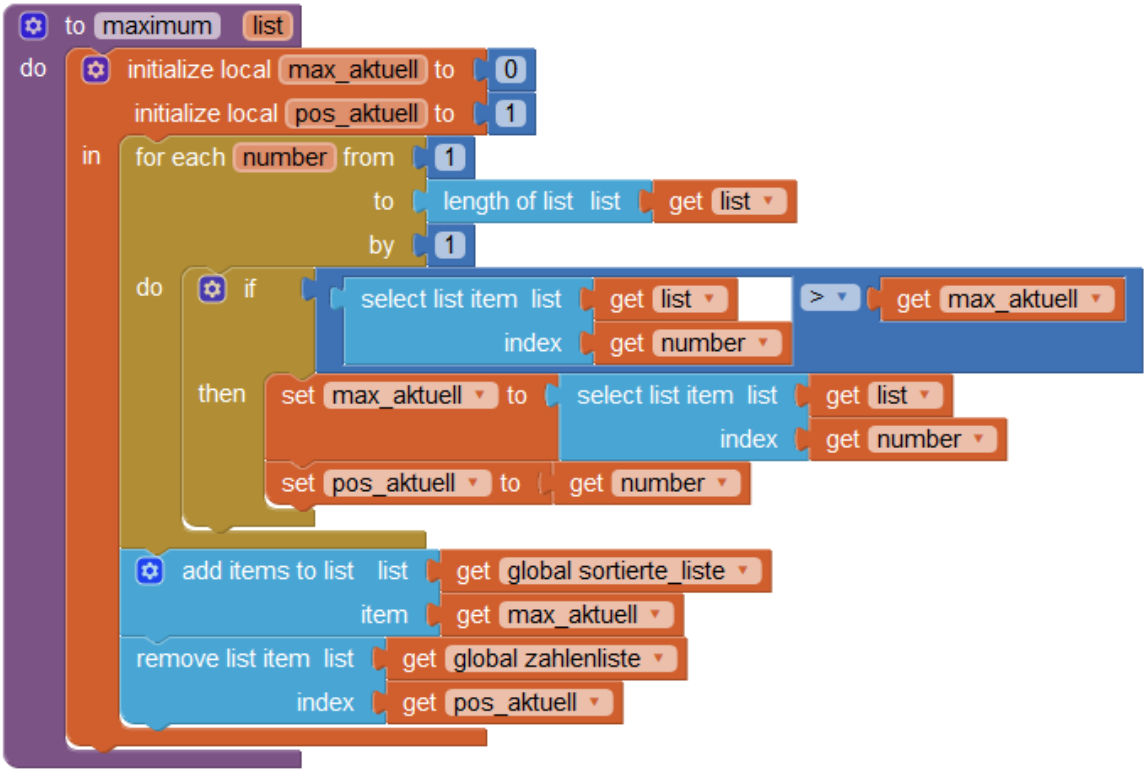
Wir suchen erneut nach dem nächsten Maximum, hängen es an die sortierte Liste (die gerade nur ein Element, nämlich das absolute Maximum enthält) an und entfernen es wieder aus der ursprünglichen Liste. Das machen wir so lange, bis die Liste keine Einträge mehr enthält.

Initialisieren der beiden Listen:

Erzeugen der Zufallszahlen und Eintragen dieser Zahlen in die „zahlenliste“:

Überlege dir, warum hier die beiden Listen geleert werden. (Müsste beim ersten Mal nicht gemacht werden. Warum später?)

Maximumsuche einer Liste – hier als Prozedur „maximum“:



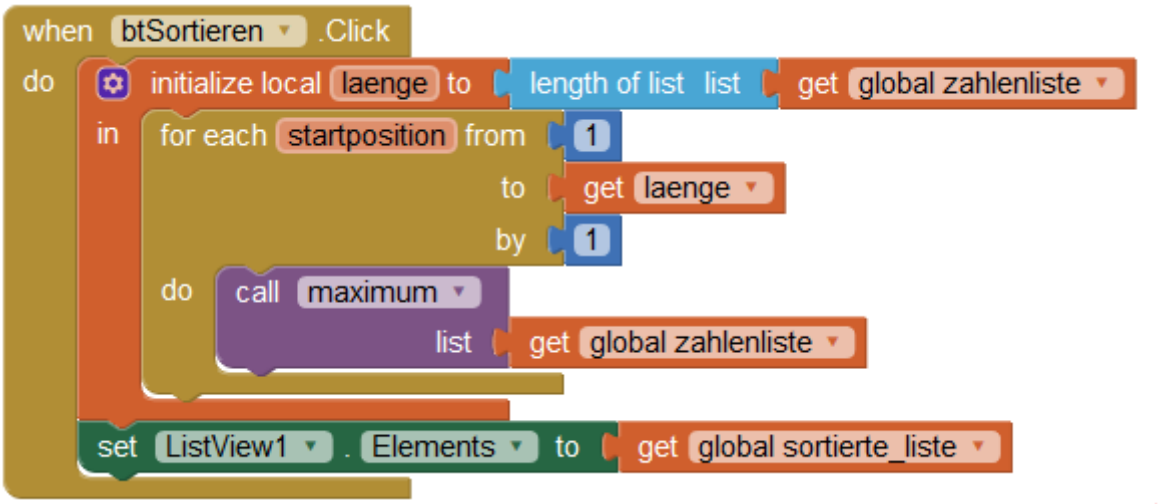
Beschreibe, was in der Prozedur passiert. Nutze dazu eine Beispielliste: 3, 6, 17, 2, 21, 8.

Beim Klick auf den Button „btSortieren“ soll nun die Zahlenliste sortiert werden wie oben beschrieben.

Dazu nutzen wir im Click-Ereignis eine lokale Variable „laenge“, der zunächst die Länge der aktuellen Zahlenliste übergeben wird.

Die Schleife wird „laenge“ Mal durchlaufen und im jedem Durchlauf wird die Prozedur „maximum“ mit der immer weiter verkürzten Zahlenliste aufgerufen.

Am Ende werden die Elemente der sortierten Liste an die ListView zur Anzeige übergeben.



HILFEKARTEN

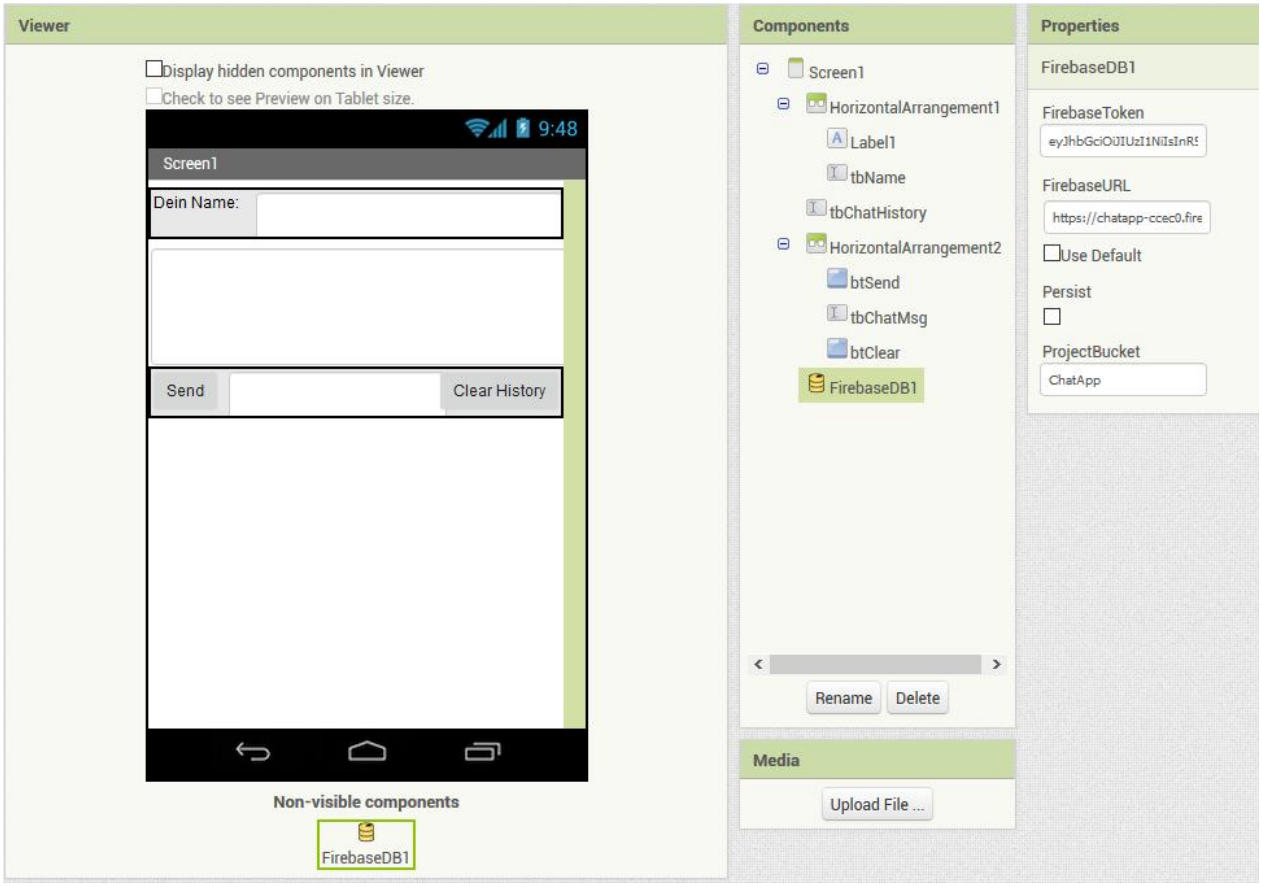
Chat-App mit Nutzen von firebase

Benötigte Komponenten:

1. Horizontales Layout mit einem Label (Text: „Dein Name:“) und einer Textbox.
2. Eine größere Textbox, bei der die Eigenschaft MultiLine (mehrere Zeilen) ausgewählt ist.
3. Horizontales Layout mit einem Button (btSend), einer Textbox (für die Nachricht) und einem weiteren Button, um das mittlere Textfeld löschen zu können.

Außerdem brauchen wir im Internet eine Möglichkeit, die aktuelle Nachricht und den aktuell Schreibenden speichern zu können. Dazu nutzen wir eine so genannte Datenbank. Hier von firebase.

Um auf diese Datenbank zugreifen zu können, brauchen wir auf unserem Screen noch eine unsichtbare Komponente aus der Palette „Experimental“: FirebaseDB



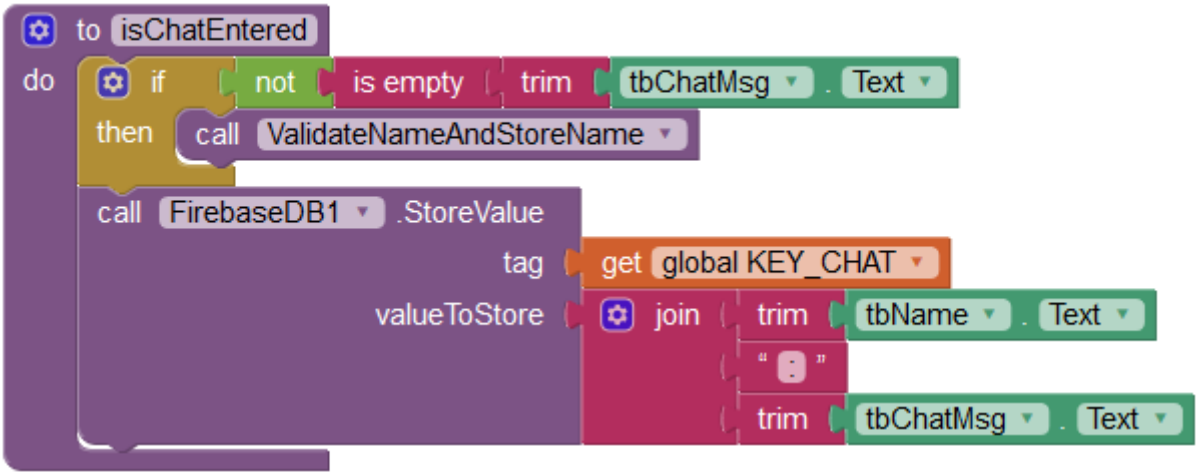
Was bei den Eigenschaften der FirebaseDB-Komponente eingetragen werden muss, erfährst du bei deiner Lehrerin (bzw. an der Tafel).

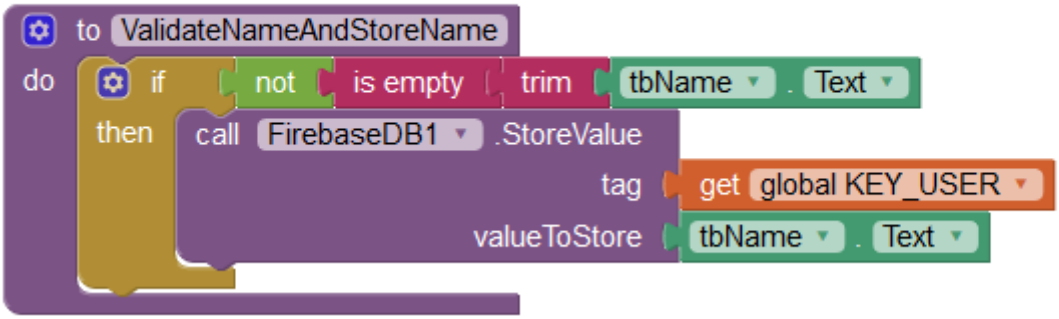
Blöcke:

Zunächst deklarieren und initialisieren wir zwei globale Variable, die wir für den Nutzer und die Nachricht in der Datenbank brauchen:

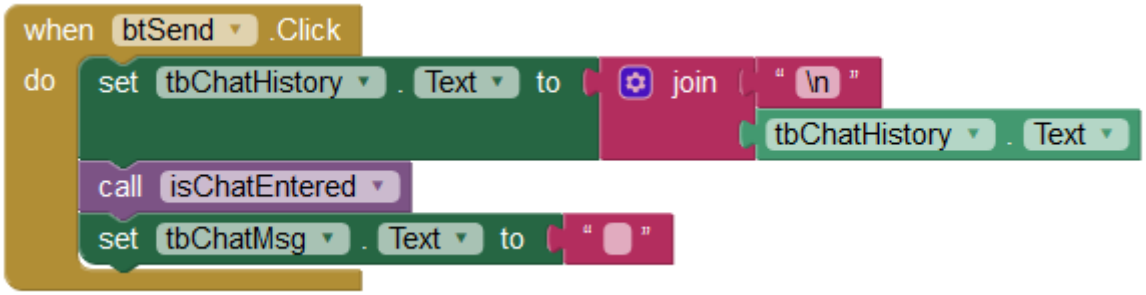
Das einfachste ist die Programmierung des Buttons btClear. Hier müssen wir nur den Text in der mittleren Textbox löschen.

Um das Programm übersichtlicher zu gestalten, schreiben wir uns zwei Prozeduren. Eine, die beschreibt, was passiert, wenn man den Chat betritt. Diese ruft eine weitere Prozedur auf, die sich um die Speicherung des Namens des Chattenden kümmert.



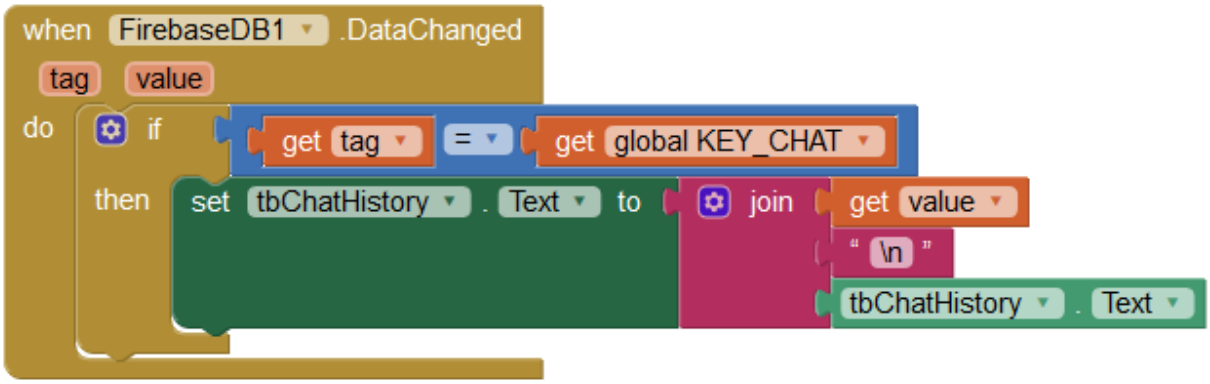


Jetzt müssen wir den Senden-Button programmieren. Hier wird im mittleren Textfeld die aktuelle Nachricht eine Zeile nach unten geschrieben, dann die erste Prozedur von oben aufgerufen und anschließend das Eingabefeld für die Nachricht wieder geleert.



Es fehlt noch die Programmierung der Datenbank. Wenn die Daten verändert wurden, sollen Sie aus der Datenbank ausgelesen und in das mittlere Textfeld geschrieben werden.

Dazu wird der ausgelesene Wert (value) vor den bisherigen Text geschrieben. Eine neue Zeile erreicht man durch „\n“.



Testet die App, indem ihr euch gegenseitig Nachrichten schreibt. Viel Spaß dabei! Und bleibt höflich!

HILFEKARTEN

Steuern eines Lego-Roboters (EV3)

Benötigte Komponenten:

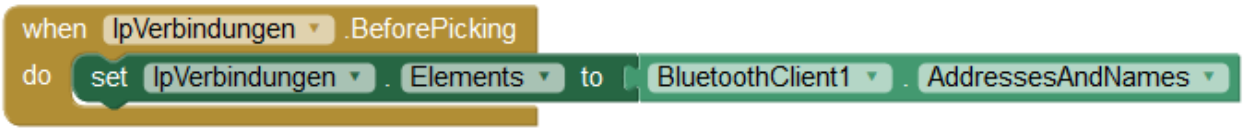
1. Einen BluetoothClient (Palette Connectivity).
2. Einen ListPicker zum Verbinden und einen Button zum Trennen der Verbindung.
3. Je nachdem, was genutzt werden soll, braucht man noch Komponenten für die Motoren oder die Sensoren. Alles dazu findet man in der Palette Lego Mindstorms. Für diese Komponenten muss die Eigenschaft BluetoothClient auf den Namen des eingesetzten BluetoothClients geändert werden

   Bei den Motoren und Sensoren müssen die Ports auch vorab in den Eigenschaften eingestellt werden.

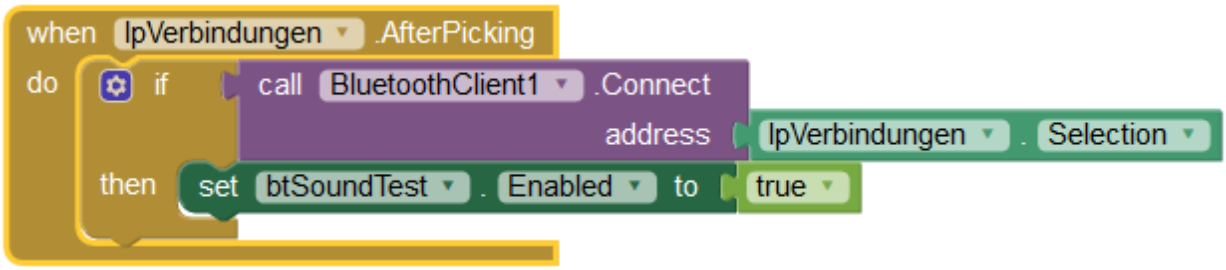
Blöcke:

Um sich mit dem EV3 zu verbinden, brauchen wir zwei Ereignisse des ListPickers.

In die Auswahlliste werden zunächst alle Adressen der Geräte aufgenommen, die in Reichweite sind und Bluetooth angeschaltet haben.

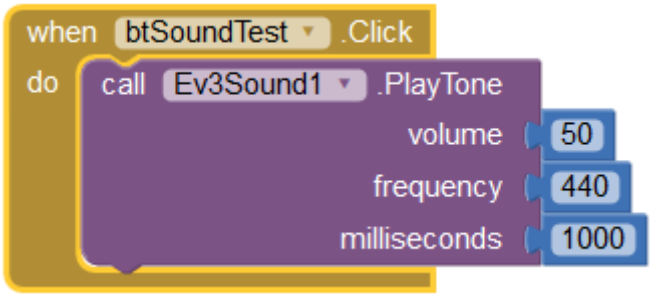


Nachdem ein Gerät ausgewählt worden ist, ruft man eine Methode auf, die die beiden Geräte über Bluetooth verbindet und stellt gleichzeitig sicher, dass das geklappt hat. Die Connect-Methode liefert dann true zurück und die Anweisung(en) in der Verzweigung werden ausgeführt.

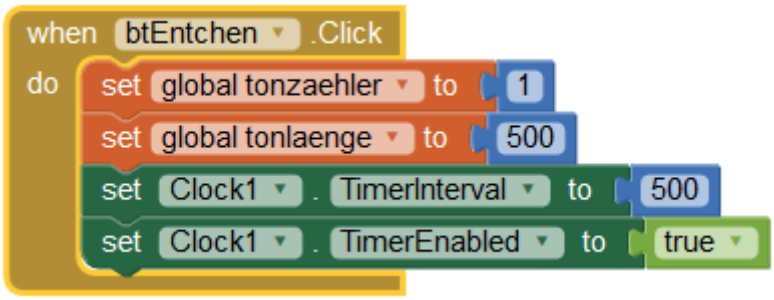


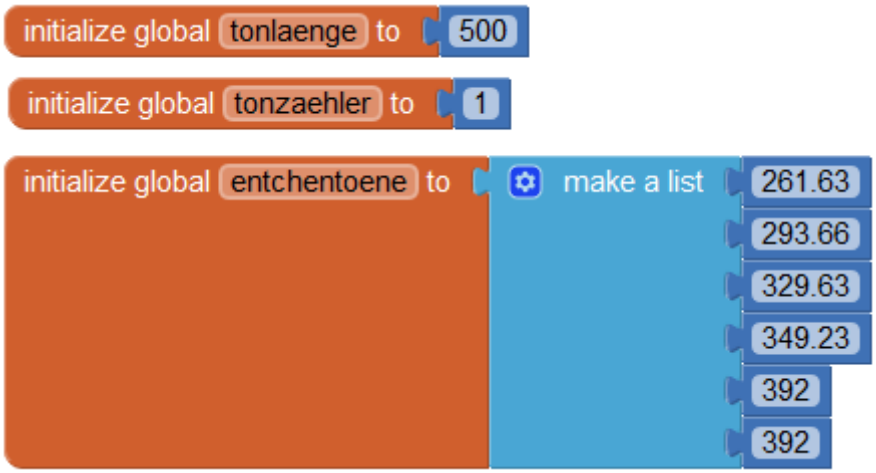
Es ist sinnvoll, hier die Buttons erst zu aktivieren, die eine Bewegung, einen Ton, o.ä. beim Roboter auslösen sollen, um einen Absturz zu vermeiden.

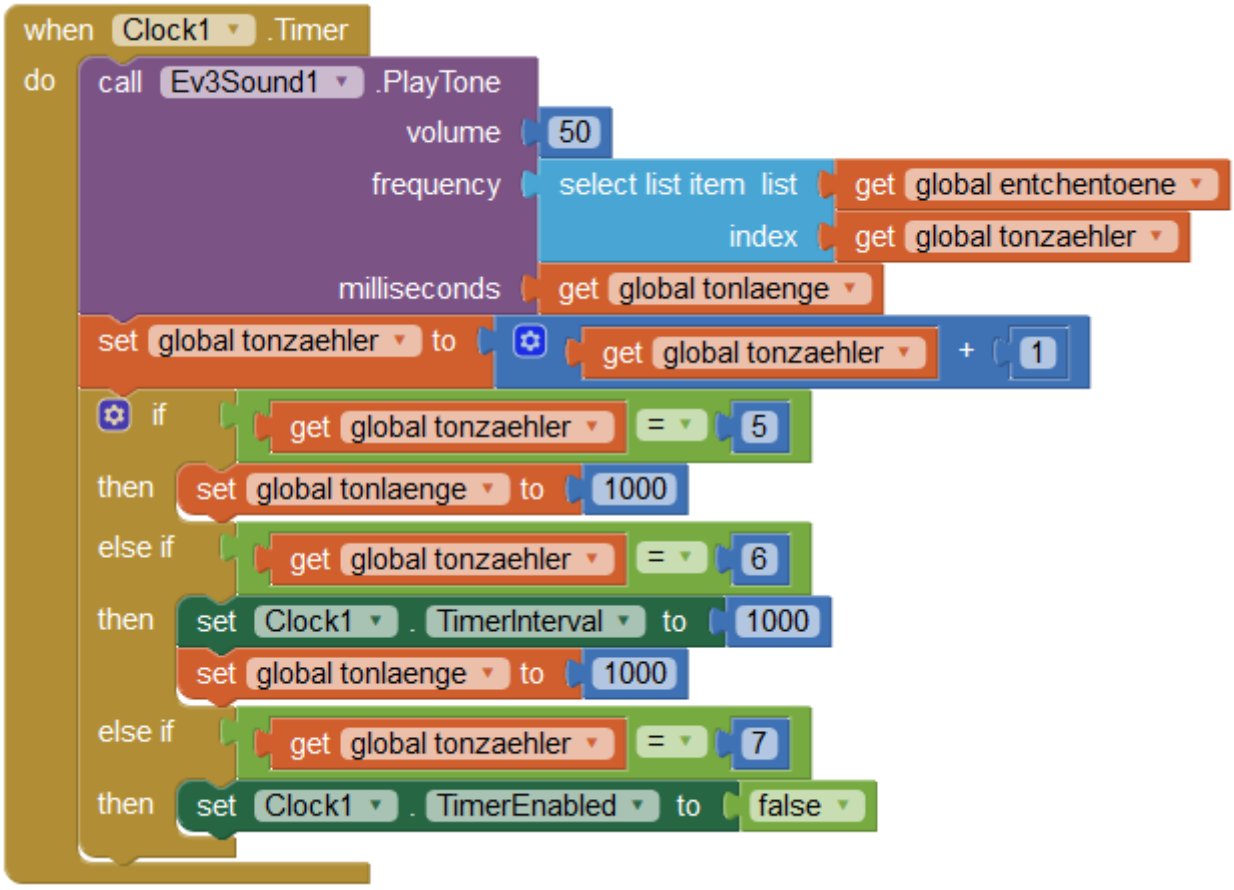
Im folgenden Beispiel wird beim Klick auf den Button btSoundTest der Ton a‘ (440 Hz) eine Sekunde lang mit Lautstärke 50 gespielt.



Die erste Zeile der Melodie von „Alle meine Entchen...“ liefert:





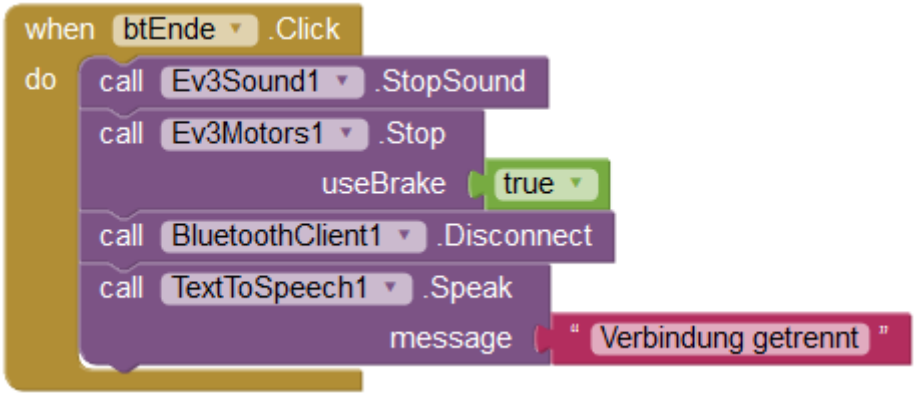


Kannst du den Code erklären?

Versuche, das Lied weiterspielen zu lassen.

Ganz am Ende ist es wichtig, dass die Verbindung wieder getrennt wird. Dabei müssen auch Motoren gestoppt und der Sound ausgeschaltet werden. Wer mag, kann eine Nachricht dazu sprechen lassen.

Dazu braucht ihr auf dem Screen eine TextToSpeach-Komponente (Palette Media).



Um den Roboter fahren zu lassen, müssen wir die Motorkomponente programmieren. Im Designer werden die Motoren bei MotorPorts eingegeben.

Schau dir den Code an. Teste ihn aus und ändere die Werte bei power und turnRatio. Kannst du erklären, für was die beiden Werte stehen?