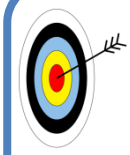


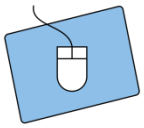
Roboterfernsteuerung über den Lagesensor!

Die erste Fernsteuerung habt ihr geschafft. Super. Und jetzt soll der Roboter auf die Bewegung eures Smartphones reagieren! Er fährt in die Richtung, in die ihr das Smartphone neigt.

Legt dazu ein neues Projekt und benennt es sinnvoll.

Das Layout – das Aussehen eurer App

Viele Bestandteile eurer App kennt ihr diesmal schon. Doch einiges ist auch neu. Neben einem *Button*, dem *Listpicker* und dem *BluetoothClient* verwendet ihr auch noch eine **„Clock“** und den **„AccelerometerSensor“**. Die Uhr (Clock) braucht ihr um regelmäßig die Bewegung eures Smartphones zu überprüfen. Der AccelerometerSensor ist der Lagesensor des Smartphones, der die Neigung des Handys misst. Um die App optisch ansprechender zu gestalten, könnt ihr den ListPicker unsichtbar machen. In dieser App versteckt ihr den ListPicker hinter einem Button der bei der Verbindungsherstellung seine Farbe von rot auf grün ändert.
Damit diese Fernsteuerung angenehmer zu nutzen ist, indem ihr das Smartphone quer haltet, müsst ihr noch die *Ausrichtung des Bildschirms* ändern. Viel zu tun, also los geht's!



- Ändert unter den Properties für Screen1 die ScreenOrientation in *„Landscape“*.
- Fügt die bekannten Bestandteile ein und benennt sie wieder sinnvoll.
- Ordnet den beiden NXTDrive wieder die einzelnen Motoren zu.
- Schreibt als Text auf den Button *„Connection“* und färbt den Hintergrund (Backgroundcolor) unter Properties rot.
- Setzt für den ListPicker das Feld *„Visible“* auf hidden.
- Zieht die *Clock* und den *AccelerometerSensor* vom Menü in die Mitte. (Beides findet ihr unter dem Punkt „Sensors“.)
- Setzt das TimerIntervall unter Properties der Clock auf 100.

Super, das Layout habt ihr geschafft. Auf zu den Funktionen!

Der Blocks Editor – die Funktionen der App

Auch hier wisst ihr schon wie der Anfang zu machen ist. Als Erinnerung nochmal die Abbildung.

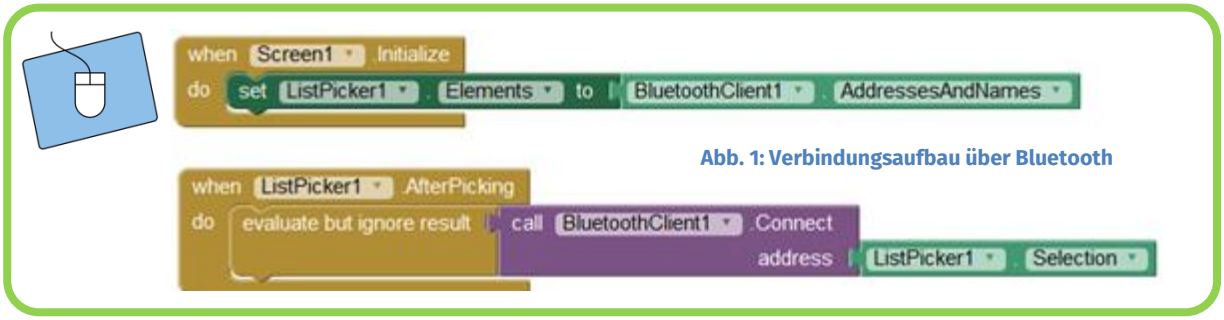


Abb. 1: Verbindungsaufbau über Bluetooth

Den Verbindungsaufbau zum Roboter könnt ihr jetzt etwas eleganter lösen. Denn dafür habt ihr den Button eingeführt. Zusätzlich braucht ihr aber noch eine **Bedingungsabfrage**. Aber keine Angst, auch das nicht schwer. Der Block, den ihr benötigt heißt **„if-then-else“.**

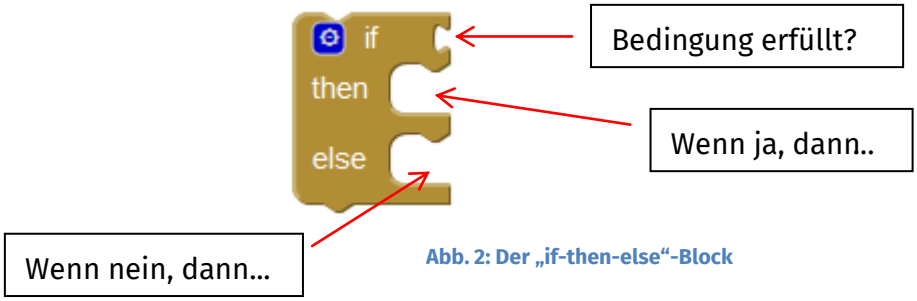
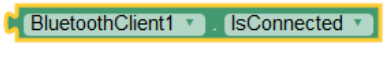
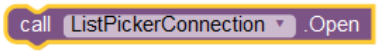


Abb. 2: Der „if-then-else“-Block

Im Menü findet ihr den Button „*if-then*“ unter dem Punkt „*Control*“. Nachdem Auswählen könnt über das Zahnrad das „*else*“ zufügen.

Die Bedingung, die ihr abfragen wollt 

ist . Falls ja, soll der Client *disconnected* werden. Das funktioniert über den *call-Aufruf* unter BluetoothClient1. Anschließend soll die Hintergrundfarbe des *ConnectionButtons* auf rot gesetzt werden. Die dafür nötigen Bausteine findet ihr im Menü unter eurem *Button* und unter *Colors*.

Für den else-Fall braucht ihr nur den Baustein 

, um die Verbindung zum ListPicker herzustellen.

- Fügt den *WhenButton.Click* Baustein ein.
- Fügt in diesen die *If-then-else-Abfrage* mit der Bedingung ein.
- Trennt im then-Fall die Verbindung.
- Stellt im else-Fall die Verbindung her.

Euer Programm sollte jetzt so aussehen:

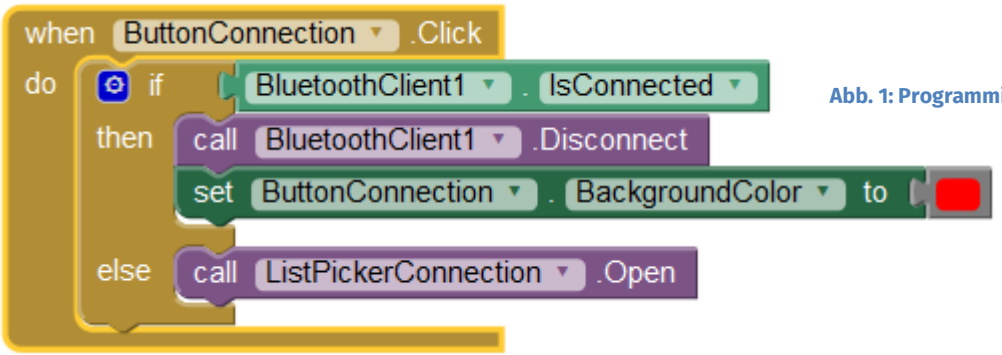


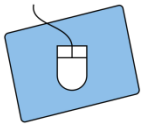
Abb. 1: Programmierung des Buttons

Super, jetzt habt ihr schon ein gutes Stück eurer App fertig. Gut gemacht!

Damit der *ConnectionButton* auch richtig funktioniert, müsst ihr jetzt noch eine Kleinigkeit anpassen. In eurem *AfterPicking-Block* wird der *„ignore result“* Baustein durch eine einfache If-Abfrage (ohne else) ersetzt, bei der die Hintergrundfarbe des Buttons auf grün gesetzt wird.

Passt den *AfterPicking-Block* so an, dass der Knopf grün wird.

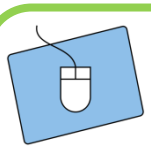
Für den Endspurt braucht ihr noch was Neues – **Variablen**. Falls ihr noch nicht so genau wisst was eine Variable ist, stellt euch einfach einen großen Container vor, der einen Namen hat, damit ihr ihn wiedererkennt und in dem ihr etwas einlagern (speichern) könnt. In diesem Fall lagert ihr Zahlen ein.
Damit ihr die Variablen benutzten könnt, müsst ihr sie erst benennen und ihnen einen Startwert zuweisen. Das macht ihr mit dem Baustein *„initialize“*, den ihr im Menü unter *Variables* findet.



- Deklariert vier Variablen und nennt sie: *Power*, *Lenkung*, *rechterMotor* und *linkerMotor*.
- Weist allen den Wert 0 zu.

So im letzten Schritt müsst ihr noch festlegen, was immer dann geschehen soll, wenn das **TimerIntervall** abgelaufen ist. Dann sollen die Variablen einen Wert zugewiesen bekommen. Dazu braucht ihr die ***set-Funktion*** der einzelnen Variablen, die findet ihr, wenn ihr mit dem Mauszeiger beim initialize-Block über den Namen fahrt. Anschließend braucht ihr wieder eine ***if-Abfrage***, um sicherzustellen, dass die Verbindung zum Roboter besteht. Wenn ja, braucht ihr nur noch die einzelnen Motoren mit ihrer entsprechenden Leistung (rechterMotor bzw. linkerMotor) fahren zu lassen. Wie das geht, wisst ihr ja bereits.

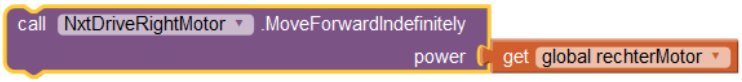
- Da die Werte, die der AccelerometerSensor misst, nur von 1-10 reichen, werden diese bei der Berechnung von Power und Lenkung mit 10 multipliziert.
- Damit man die spezifische Leistung der einzelnen Motoren berechnen kann, muss man die Summe/Differenz zwischen Power und Lenkung bilden.



- Wählt den Block *„When Clock1.Timer“* aus.
- Setzt die Variable *Power*

- Setzt genau so auch die *Lenkung*. Verwendet dabei aber den Sensorwert *.YAccel*.
- Setzt die Variable *linkerMotor*

- Setzt die Variable *rechterMotor*=Power-Lenkung.
- Fügt die *if-Abfrage* mit der Bedingung *BluetoothClient1.isConnected* ein.
- Setzt in den *then-Block* die beiden Motoren ein und lasst sie mit der entsprechenden Leistung vorwärts fahren.



- Aufatmen, ihr habt es geschafft!!

Die App ausprobieren!

Testet eure App und lasst die Roboter die Welt erobern!

Viel Spaß beim Testen!

Falls ihr technische Probleme habt, meldet euch einfach bei einer/m Betreuer/in. Sie helfen euch gerne bei der Verbindungsherstellung.

Herzlichen Glückwunsch!

Und auch hier das Speichern nicht vergessen! ☺

Neues ausprobieren!

Wenn ihr das Layout eurer App langweilig findet, lasst euren Ideen freien Lauf!
Falls euch noch weitere Funktionen einfallen, baut sie ruhig selbstständig ein. Ansonsten geht es jetzt zur Vorbereitung des Wettbewerbs..

Quellenverzeichnis

Abb. 1, 2, 3, sowie die Abbildungen einzelner Puzzleteile – *Quelle: Screenshots aus der MIT AppInventor-Software*

Alle weiteren Grafiken/Icons – Quelle: InfoSphere