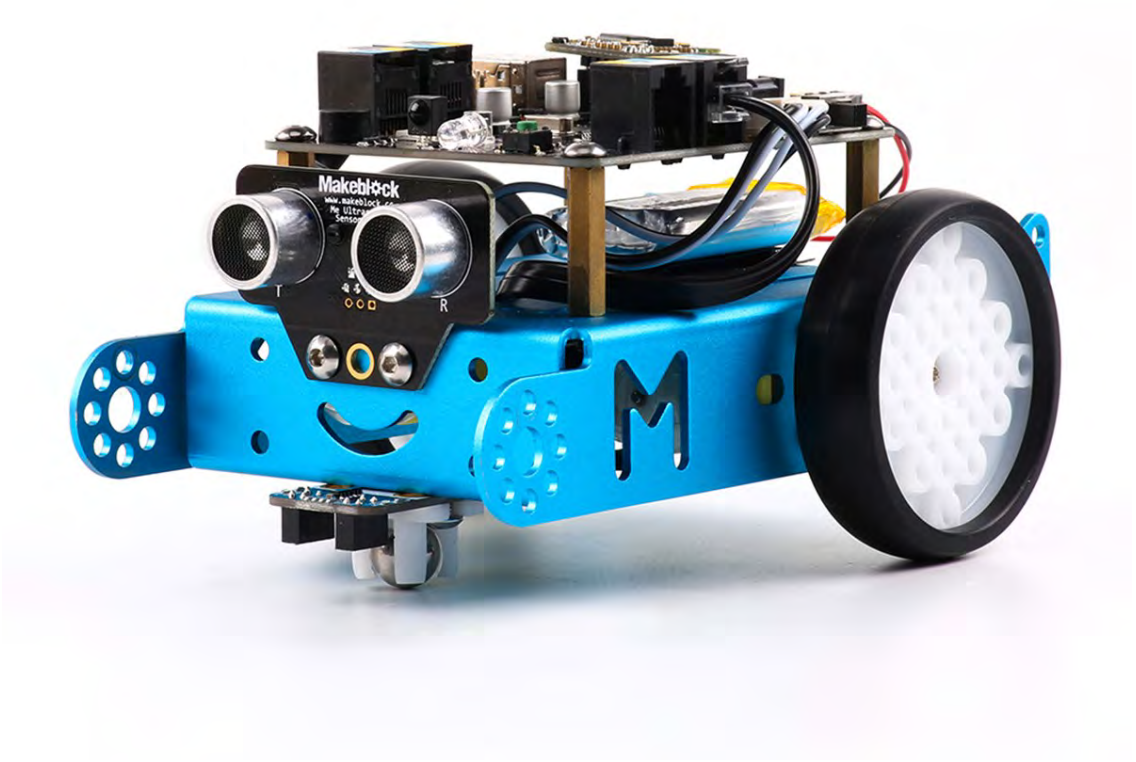
TfK Technik für Kinder e.V.
Begeistern durch Machen

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Modul:

Physical Computing mBot



Version: 1.0

SCRATCH ARDUINO

Dieses Modul wurde an der Lehr- und Forschungseinheit Mathematik und Informatik (LMI) der Universität Passau in Kooperation mit dem Verein „TfK - Technik für Kinder e.V.“ im Rahmen des Projekts Technikgruppen an Schulen - Kinder für Technik begeistern erarbeitet.
Mehr zu diesem Projekt erfahren Sie auf der Internetseite

http://www.fim.uni-passau.de/fim/fakultaet/lehrstuehle-professuren-und-fachgebiete-der-fim/didaktik-der-informatik/projekte/technikgruppen.html .

Autor:

Wolfgang Pfeffer

Mitarbeiter der Didaktik der Informatik / TfK
wolfgang.pfeffer@uni-passau.de

Modulübersicht

Im Rahmen des Kooperationsprojekts zwischen der Universität Passau und dem Verein Technik für Kinder e.V. sind folgende Module entstanden:

Modul Roboter-Programmierung

Dieses Modul bietet einen sehr kurzen Einstieg in die Programmierung eines Lego EV3 Roboters mit der Lego Mindstorms Education Software. Neben Hardware- und Software Anforderungen wird die Entwicklungsumgebung vorgestellt und anschließend Einstiegsaufgaben zu Motoren, Sensoren sowie vermischte Aufgaben bereitgestellt.

Modul App-Entwicklung auf Mobile Devices

Dieses Modul thematisiert umfassend die App-Entwicklung auf Mobile Devices mit dem AppInventor, einer graphischen Programmierumgebung. Ein einfacher und handlungsorientierter Zugang zu dieser Thematik steht dabei im Vordergrund. Neben Hardware- und Softwareanforderungen wird eine ausführliche Installationsanleitung angeboten. Die Einführungsaufgabe 'Schritt für Schritt zur ersten eigenen App' ermöglicht eine Einführung in den Umgang mit dem AppInventor. Gleichzeitig werden fast alle wichtigen Elemente behandelt. Anschließend umfasst das Modul 10 Aufgaben mit unterschiedlichem Schwierigkeitsgrad sowieso ausführlichen Lösungen. Exkurse zu Themen wie GPS und Dijkstra-Algorithmus runden die Handreichung ab.

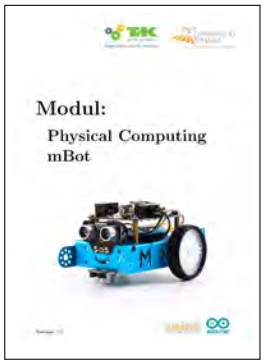
Modul App-Entwicklung mit Java Android

Dieses Modul baut thematisch auf dem Modul 'App-Entwicklung auf Mobile Devices' auf, anstelle der graphischen Programmierung werden die Applikationen nun mit Java Android entwickelt. Da die Einstiegshürde in Java Android vergleichsweise hoch ist, sind grundlegende Java-Kenntnisse unumgänglich. Es wird ein sehr ausführlicher sowie kleinschrittiger Einstieg in die Programmierung mit Java Android geboten. Anschließend stehen verschiedene Projekte mit umfangreichen Lösungen zur Auswahl (z.B. Vokabel-App, Quiz-App, 2048-App, Datenspeicher oder LegoPilot-App). Dabei werden wichtige Konzepte der Sekundarstufe II behandelt.

Modul Physical Computing

Dieses Modul kombiniert Elemente aus der Physik und der Informatik und basiert auf dem Einplatinencomputer Raspberry Pi. Zunächst geht es darum, (einfache) Schaltungen zu erstellen und mit konstanter Spannungsquelle zu arbeiten. Hierbei wird mit verschiedenen Schaltungselementen experimentiert (z.B. LED, Widerstand, LDR, Poti, Servomotor). Anschließend soll an manchen Pins gezielt Spannung angelegt oder nicht angelegt werden. Dazu werden die GPIO-Pins mit der Programmiersprache Python konfiguriert und angesteuert. Die Handreichung beinhaltet ein eigenes Kapitel zu Python, in welchem man sich anhand kleiner Aufgaben mit der Syntax der Programmiersprache vertraut machen kann.

Modul Physical Computing mBot



Dieses Modul kann als Schnittstellenmodul zwischen Roboter-Programmierung und Physical Computing gesehen werden. Bei mBot handelt es sich um ein handliches Roboterfahrzeug, das mit verschiedenen Sensoren und Motoren ausgestattet ist, die mit dem Mikrocontroller Arduino verbunden werden können. Im Bezug auf die Programmierung bietet das Modul zwei Zugangswege: Ähnlich zur App-Entwicklung mit dem AppInventor kann der mBot mithilfe der graphischen Programmierumgebung Scratch angesteuert werden. Ein zweiter Teil des Moduls bietet die Programmierung des mBot mit Hilfe von Arduino-C.

Inhaltsverzeichnis

Kapitel 1

Über das Modul

Das Modul *Physical Computing - mBot* beschäftigt sich mit der Erstellung von kleinen Programmen für einen Roboter. Diese Programme werden sowohl in der leicht zu erfassenden und erlernenden grafischen Oberfläche Scratch, sowie später in dem Hardware nahem C-Dialekt Arduino-C behandelt.

Der mBot eignet sich sehr gut für diesen Zweck, da dieser (insbesondere preislich) auf Schulen zugeschnitten wurde.

Kapitel 2

Was wird benötigt?

Überblick:

Im Folgendem werden die benötigten Komponenten, untergliedert nach Soft- und Hardware erläutert.

2.1 Hardware

Jeder Nutzer benötigt einen mBot und ein zugehöriges USB-Kabel:

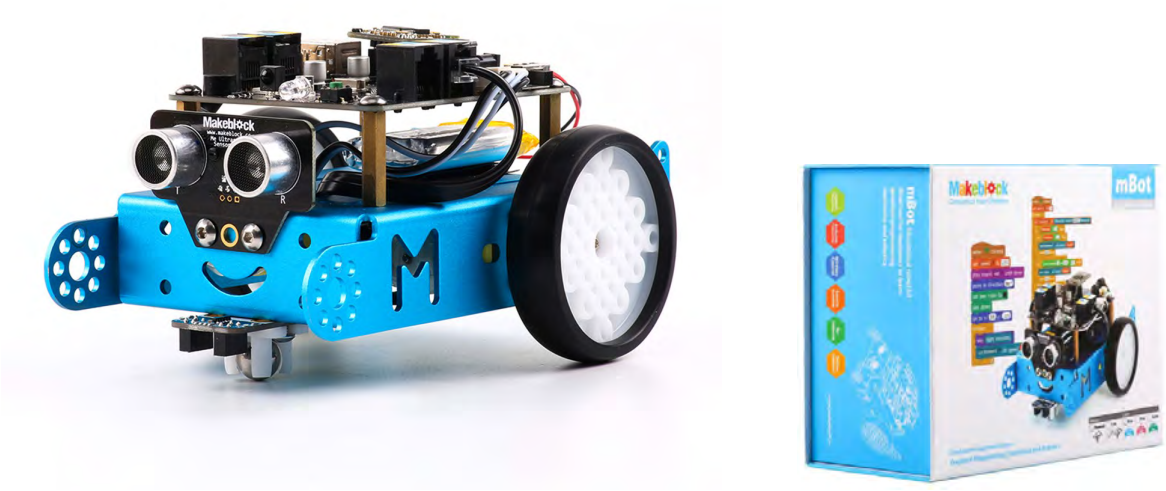


Abbildung 2.1: Roboter mBot (links); Box mit Roboter, Infrarot-Fernbedienung und Kabel (rechts).

Die Box aus Abbildung 2.1 ist bei diversen Anbietern online erhältlich. Der Komplettpreis bewegt sich zwischen 60 € und 80 €.

Zur Programmierung des mBot-Roboters wird einen PC mit Windows oder OSX benötigt. Das Betriebssystem Linux wird von den Herstellern leider nicht unterstützt und kann deshalb nur zur Programmierung mit Arduino-C verwendet werden.

Hinzu kommen vier R6 (AA) Batterien oder alternativ ein Akkupack.

Achtung

Die meisten Akkus sind ungeeignet, da diese nur eine Spannung von 1.2 Volt liefern. Um die Motoren (korrekt) zu betreiben sind allerdings 1.5 Volt notwendig.

Nach Erhalt der Box muss der Roboter vor der ersten Inbetriebnahme zusammengebaut werden. Hierfür liegt eine ausführliche Anleitung bei. Gleichzeitig bekommt man einen ersten Überblick über die Ausstattung des Roboters (verbaute Motoren, Sensoren und Ausgabegeräte). Die Anleitung hält auch verschiedene Grafiken bereit, anhand derer die einzelnen Komponenten visualisiert und beschrieben werden:

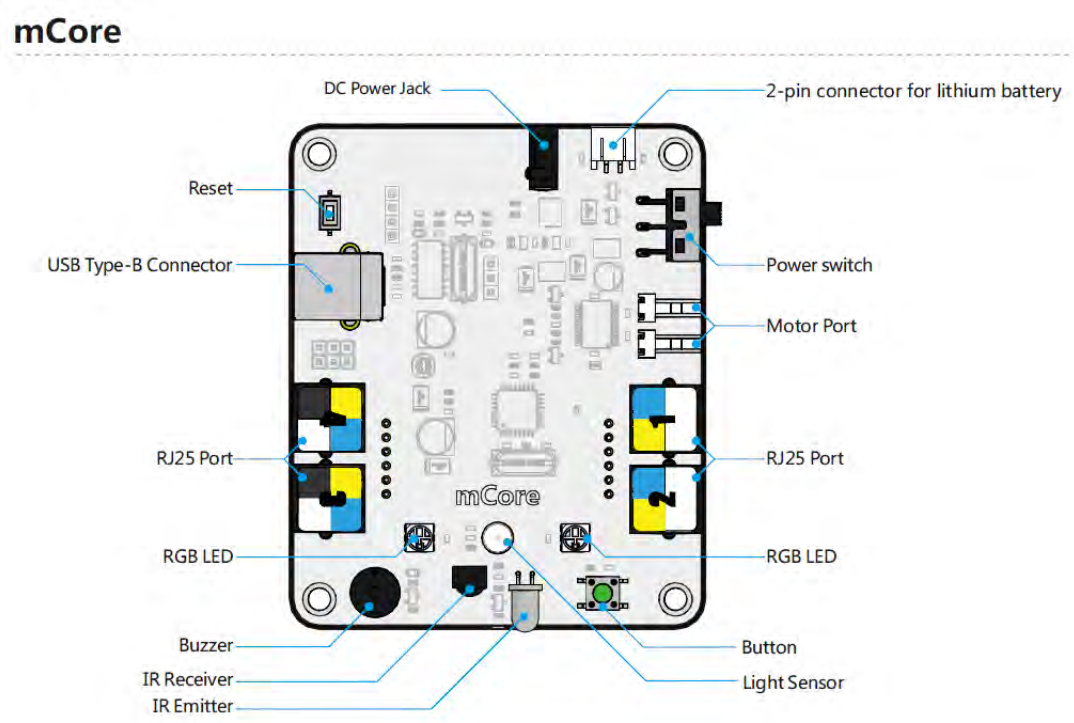


Abbildung 2.2: Beschreibung des mCore in der beiliegenden Anleitung.

2.2 Software

Das Modul bietet zwei Zugänge an, einerseits die graphische Programmierung des mBot mit Hilfe des Programms `mBlock`, andererseits die textbasierte Programmierung des mBot mittels Arduiono-C. Der Hersteller Makeblock liefert alle notwendigen Komponenten mit dem Programm *`mBlock`*. Dieses kann unter `http://www.mblock.cc/download` heruntergeladen werden. Auf die Programmierung des mBot mit der Programmiersprache Arduino-C gehen wir im entsprechenden Abschnitt näher ein.

Kapitel 3

Programmierung mit Scratch4Arduino

Überblick:

Dieses Kapitel beginnt mit einer eine Einführung in die graphische Programmierung des mBot-Roboters. Anschließend gibt es zuerst kleinere Aufgaben zu den verschiedenen Komponenten des Roboters. Dazu zählen die verschiedenen Motoren, Ausgabegeräte sowie Sensoren. Somit sind in den Abschnitten 3.2, 3.3 sowie 3.4 zunächst kurze und übersichtliche Programm zu erstellen, mit denen etwa beide Motoren angesteuert werden, eine LED blinken soll oder der Wert des Ultraschallsensors ausgegeben werden soll. Anschließend gibt es dann einige vermischte Aufgaben, bei denen u.a. Motoren, Ausgabegeräte und Sensoren miteinander interagieren sollen. Lösungshinweise zu den Aufgaben sind im nächsten Kapitel zu finden.

3.1 Scratch4Arduino (mBlock)

Alle Programme in diesem Kapitel werden mit Hilfe der graphischen Entwicklungsumgebung mBlock erstellt. Dieser Abschnitt enthält Installationshinweise sowie erste Schritte mit diesem Programm.

3.1.1 Installation

Die Software für Windows und Mac kann unter `http://www.mblock.cc/download` gefunden werden. Hier kann das Programm für das entsprechende Betriebssystem heruntergeladen und anschließend installiert werden[1]:

[1] Quellenangabe: `https://www.colourbox.de/vektor/zahnrad-teamarbeit-verbindung-vektor-10011069`

3.1.2 Erste Schritte

Beim ersten Start des Programms nach der Installation finden wir die folgende Übersicht:

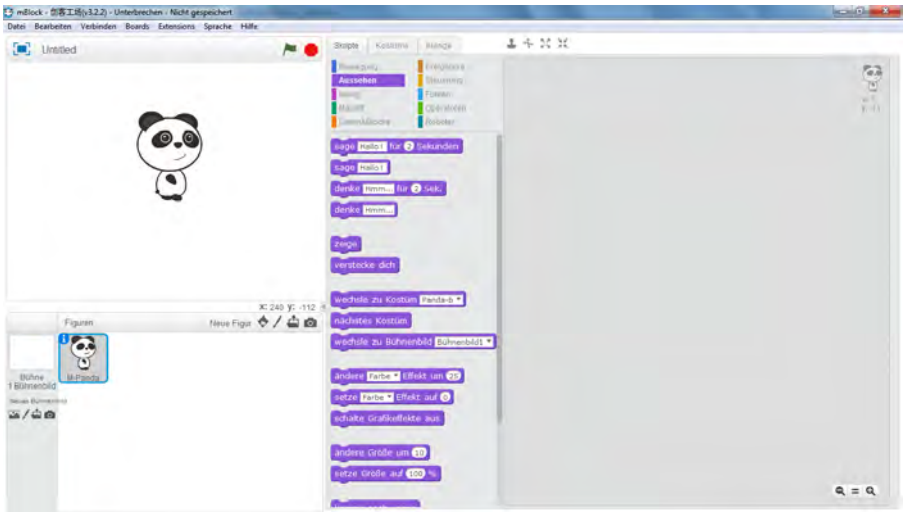


Abbildung 3.1: Standardansicht des Programms mBlock.

In der Mitte finden sich farbige Bausteine, mit denen ein Programm erstellt werden kann. Hierzu müssen die entsprechenden Bausteine im rechten (grauen) Bereich platziert werden. Dies ist via Drag & Drop möglich. Im linken oberen Bereich ist ein Panda zu sehen. Dieser kann mit Hilfe der Bausteine programmiert werden.

Sie können hier erst einmal mit verschiedenen Bausteinen experimentieren, um ein Gefühl für die Funktionsweise der Umgebung zu bekommen. Wichtig hierbei ist der Startbaustein  . Dieser Baustein stellt sicher, dass das erstellte „Programm“ gestartet wird, wenn auf die grüne Flagge rechts oberhalb des Pandas geklickt wird. Über das rote Stopp-Symbol daneben können Sie ihr Programm wieder beenden:

Abbildung 3.2: Ein einfaches Programm

3.1.3 Verbinden des mBot

Bevor der Roboter vom PC erkannt wird, muss noch der entsprechende Treiber für die Hardware installiert werden.
Dieser wird durch Klicken auf die Schaltfläche *Verbinden* → *Installiere Arduino Treiber* installiert.

Installation des Hardware-Treibers

Denken Sie daran, den entsprechenden Treiber für die Hardware zu installieren. Ansonsten ist eine Verbindung zwischen PC und mBot nicht möglich.

Verbindet nun den Roboter via USB-Kabel mit dem PC und stellt die Verbindung durch Klicken auf die Schaltfläche *Verbinden* → *Serieller Port* → *COM* her.

3.1.4 Der erste Test

Wir wollen zunächst überprüfen, ob die Verbindung mit dem mBot erfolgreich hergestellt wurde bzw. ob wir nun Programme auf den mBot spielen können. Dazu nutzen wir folgendes kleines Testprogramm. Die Bedeutung des blauen Bausteins klären wir etwas später:

Abbildung 3.3: Testprogramm, um die Verbindung zum mBot testen zu können.

Durch Klicken auf die grüne Flagge wird das Programm automatisch auf den Roboter gespielt, falls eine Verbindung hergestellt wurde. Alternativ lädt auch ein Doppelklick auf den  Block das Programm auf den mBot. Das Programm wird anschließend automatisch gestartet.

Sobald der Code übersetzt und übertragen ist, sollten die beiden Lämpchen auf der Platine grün aufleuchten.

3.2 Motoren

In diesem Abschnitt beschäftigen wir uns zunächst ausschließlich mit den Motoren. Wir betrachten die Bausteine des Programms, mit denen Motoren gestartet, angesteuert und gestoppt werden können. Nach diesem Abschnitt sollten Sie die entsprechenden Bausteine für die Motor-Steuerung kennen und einsetzen können.

3.2.1 Aufgabe 1 – Ansteuern eines Motors

Aufgabenstellung

Es soll der erste Motor (M1) gestartet werden, 10 Sekunden lange laufen und anschließend wieder gestoppt werden.

Neue Blöcke:

Zur Bearbeitung dieser Aufgabe benötigt ihr folgende neuen Blöcke:

Findet diese Blöcke im mBlock-Programm und setzt diese geeignet zusammen, so dass euer Roboter die vorgegebene Aufgabenstellung erfüllt. Euer Programm könnt ihr wieder durch Klicken auf die grüne Fahne oder durch Doppel-Klick auf  starten.

3.2.2 Aufgabe 2 – Fahre vorwärts

Aufgabenstellung

Bei Aufgabe 1 ist der mBot nur im Kreis gefahren. Jetzt soll er 10 Sekunden vorwärts fahren und anschließend wieder anhalten.

Neue Blöcke: Für die Lösung gibt es verschiedene Varianten. Intuitiv können hier zwei Blöcke wie bei Aufgabe 1 verwendet werden. Alternativ bietet mBlock auch folgenden Block an:

Anmerkung

Falls deine Batterien nicht voll geladen sind oder du Akkus verwendest kann es sein, dass der Roboter nicht geradeaus fährt. Auch der Boden kann dazu führen, dass der Roboter nicht exakt geradeaus fährt. Je nach Boden kann es zudem vorkommen, dass ein Rad blockiert.

3.2.3 Aufgabe 3 – Fahre ein Quadrat ab

Aufgabenstellung

Der Roboter soll nun ungefähr ein Quadrat abfahren. Du kannst hierzu zur Vereinfachung annehmen, dass es bei einer Geschwindigkeit von 100 genau eine Sekunde dauert, um 90 Grad zu drehen.

Neue Blöcke:

3.2.4 Aufgabe 4 – Wiederverwendbarkeit

Wir können in Scratch auch eigene Blöcke definieren. Dadurch werden Programme deutlich kürzer und übersichtlicher. Einen eigenen Block kannst du in der Rubrik *Daten & Blöcke* definieren:

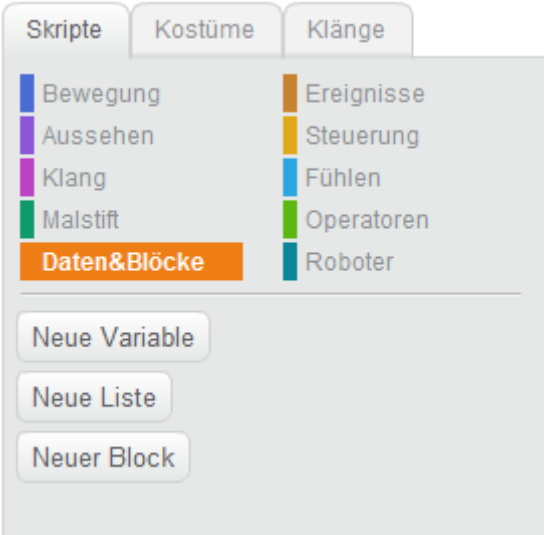


Abbildung 3.4: Menü in der Mitte

Abbildung 3.5: Erstellen eines neuen Blocks

Du musst deinem Block zunächst einen Namen geben. Wir möchten an dieser Stelle einen eigenen Block erstellen, mit dem sich der Roboter um 90 Grad dreht und nennen ihn deshalb *Drehe*. Weiter definieren wir uns einen eigenen Block *Fahre*, mit dem der Roboter für 3 Sekunden vorwärts fährt.

Verschiebe nun die entsprechenden Teile Deines Programms aus Aufgabe 3 in die jeweiligen neuen Blöcke und verwende nun die Blöcke 

und 

, um den Roboter vorwärts fahren bzw. drehen zu lassen.

3.3 Ausgabegeräte

3.3.1 Theorie – LED

Was ist eigentlich eine LED?[2]

Eine LED (engl.: *light-emitting-diode*) ist eine Leuchtdiode, d.h. ein elektrisches Bauelement, welches Strom nur in eine Richtung fließen lässt (deswegen spricht man auch von einem Halbleiter). Eine LED wandelt elektrische Energie in Licht um. Unter https://www.youtube.com/watch?v=xeOdnAeo8E4 findest Du ein Video aus der Sendung mit der Maus. Diese erklärt Funktionsweise von LEDs sowie deren Vorteil gegenüber herkömmlichen Glühbirnen.

Bei der LED des mBot handelt es sich um eine RGB-LED. Der Zusatz RGB ist wichtig, denn dieser besagt, dass unsere Leuchtdiode nicht nur in einer Farbe leuchten kann. RGB steht hierbei für Rot–Grün–Blau. Vielleicht kennst du den RGB-Farbraum ja schon aus dem Kunstunterricht. Dabei handelt es sich um einen additiven Farbraum, d.h. durch das additive Mischen der drei Grundfarben können alle anderen Farben erzeugt werden. Abbildung 3.6 zeigt dir die Mischpalette dieser drei Grundfarben.

Mischpalette der drei Grundfarben rot, grün und blau:

[2]Quellenangabe Bild: https://www.colourbox.de/bild/led-lampe-low-power-bild-7920925
[2]Quellenangabe Bild: http://www.mazi-trendidee.de/ebay/admin/upload/389/16_Die_Maus_guckt.jpg

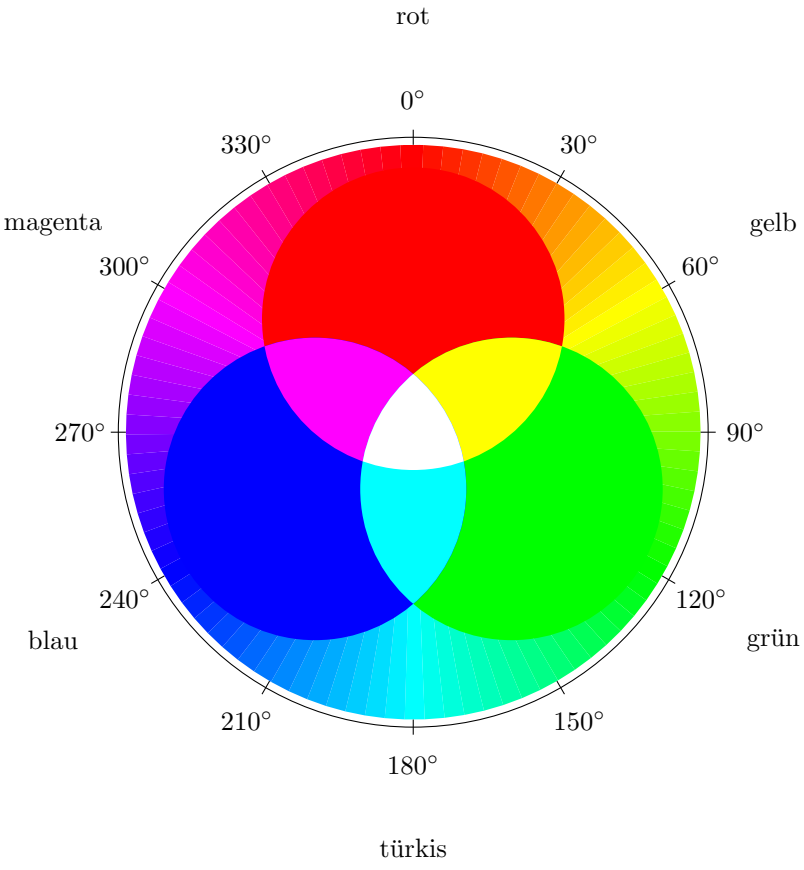


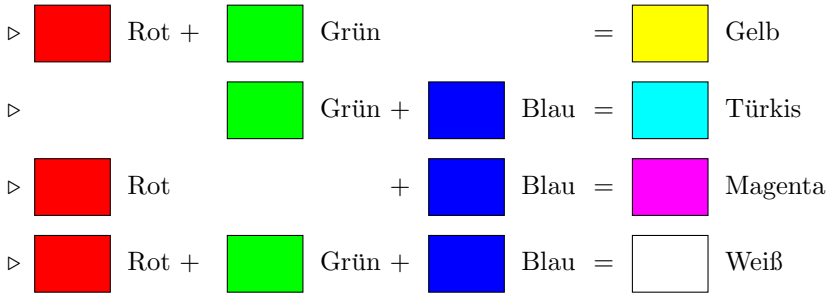
Abbildung 3.6: Mischpalette der drei Grundfarben rot, grün und blau.

Jeder der drei Farben rot, grün und blau kann hier ein Wert zwischen 0 und 255 zugewiesen werden:

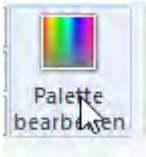
Dabei bedeutet 0 'Aus' und 255 'An'. Werte dazwischen dimmen die entsprechende Farbe. Je größer die Zahl einer bestimmten Farbe ist, umso stärker wird dieser Farbanteil beigemischt.

Beispiele:

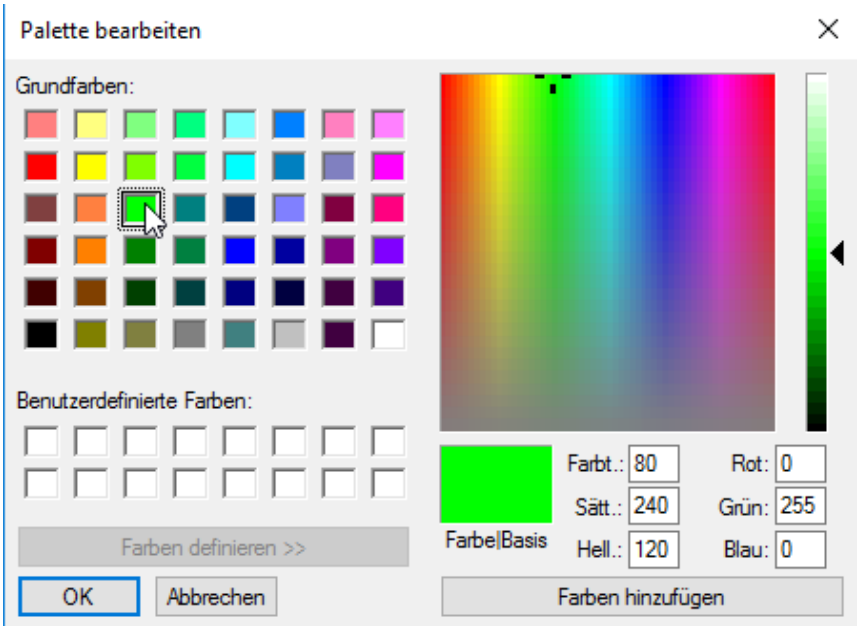
- Der Idealfall der additiven Farbmischung lässt sich durch drei Scheinwerfer mit den Farben rot, grün und blau beschreiben, die auf eine schwarze Fläche leuchten. Die Lichtkegel der Scheinwerfer sollen sich dabei teilweise überschneiden (s. Abbildung 3.6 ohne das Äußere der drei Kreise). Hier ergeben sich also folgende Farbmischungen:



- Die Werte können auch mit Programmen wie GIMP oder Paint ermittelt werden. Wir zeigen Dir hier exemplarisch, wie Du die RGB-Werte einer Farbe in Paint herausfinden kannst. Suche dazu im Menü das folgende *Palette bearbeiten*-Symbol:



Nachdem Du auf dieses Symbol geklickt hast, öffnet sich folgende Farbpalette:



Hier kannst du jede beliebige Farbe auswählen, indem Du den Cursor in dem farbigen Quadrat bewegst. Die Helligkeit kannst Du über den Balken an der rechten Seite festlegen. Sobald Du deine gewünschte Farbe gefunden hast, kannst Du unten rechts die RGB-Werte ablesen und diese dann direkt in das *mBlock*-Programm übertragen.

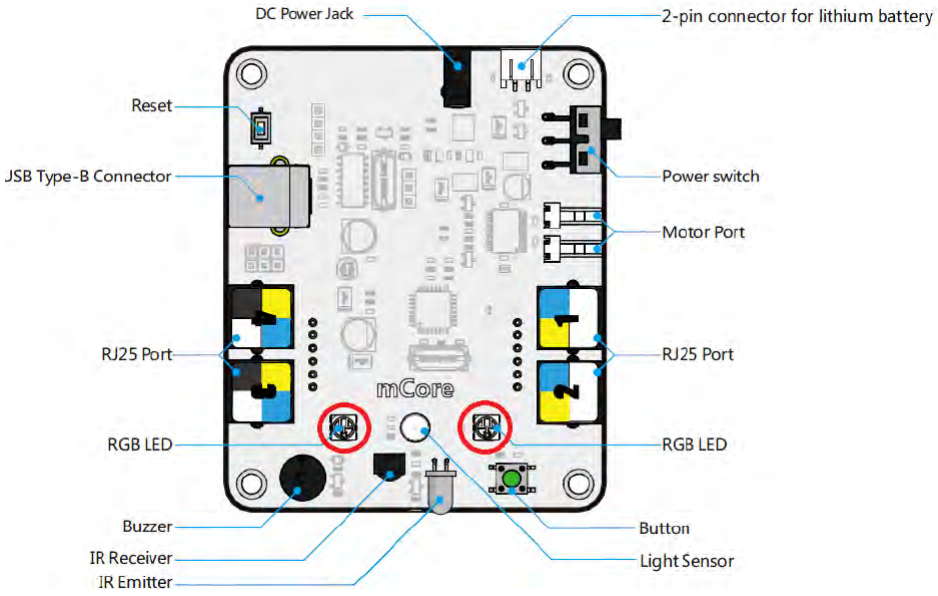
3.3.2 Aufgabe 1 – LED

Aufgabenstellung

Erstelle ein Programm, das die erste LED (*LED 1*) blau und die zweite LED (*LED 2*) rot leuchten lässt.

Zur Erinnerung:

Die beiden RGB-LEDs findest du an folgenden Stellen der Platine:



Wir wollen nun *LED 1* blau und *LED 2* rot färben.

Neue Blöcke:

Dazu benötigen wir einen weiteren Block:

3.3.3 Aufgabe 2 – LED Farbmischung

Aufgabenstellung:

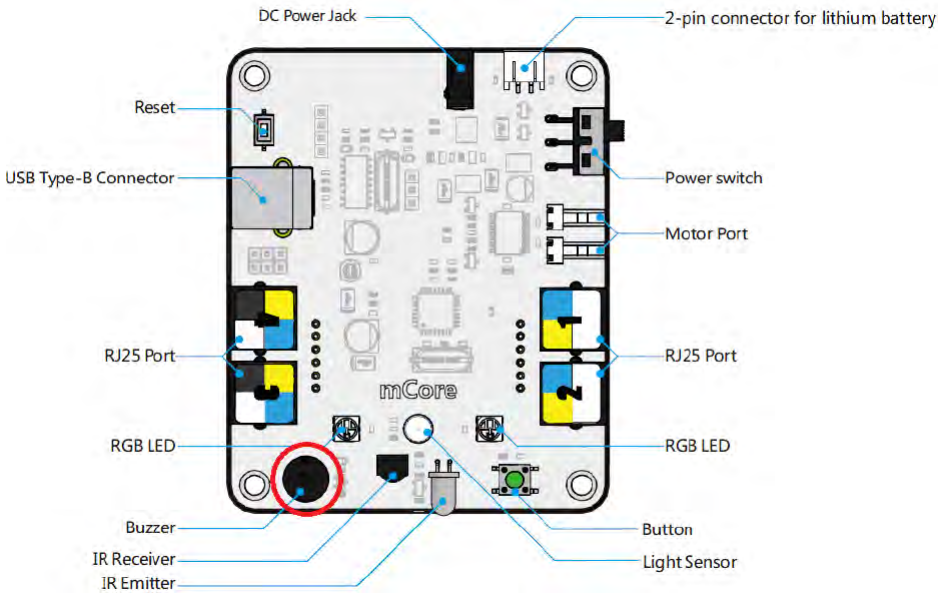
In der vorherigen Aufgabe haben wir einfache Farben gewählt, die direkt eingegeben werden konnten. Diesmal soll *LED 1* Gelb und *LED 2* Magenta leuchten.

Überlege Dir selber eigene Farbmischungen und lass die LEDs darin leuchten.

3.3.4 Aufgabe 3 – Der Buzzer

Aufgabenstellung:

Als zweite Ausgabemöglichkeit befindet sich auf dem Board ein Buzzer, der einfache Töne wiedergeben kann:



Lasse den Buzzer eine ganze Note D4 abspielen. Die Zahl 4 gibt hierbei die entsprechende Oktave an. Dabei ist D4 das „klassische D“.

Neue Blöcke:

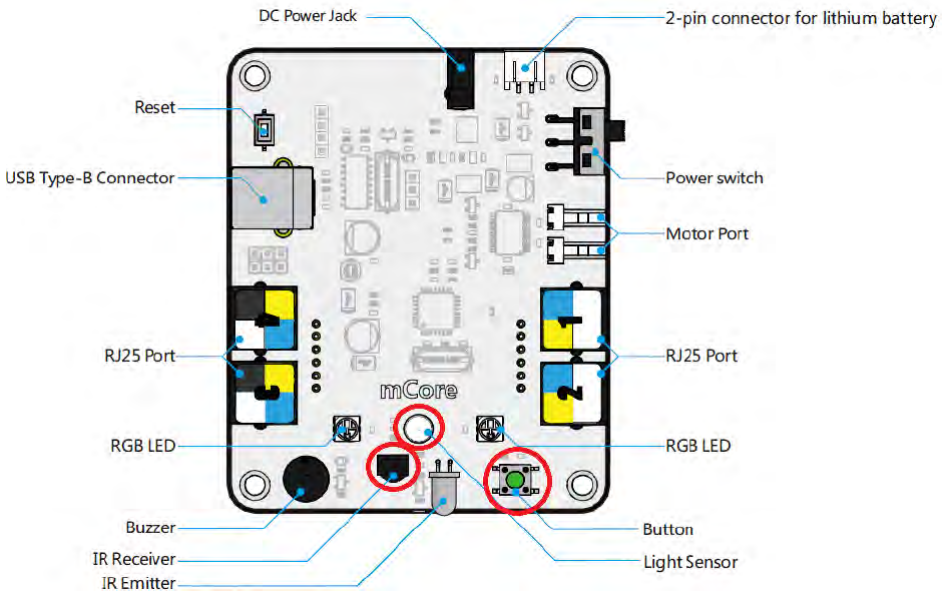
Hierzu benötigst du folgenden neuen Block:

3.4 Sensoren

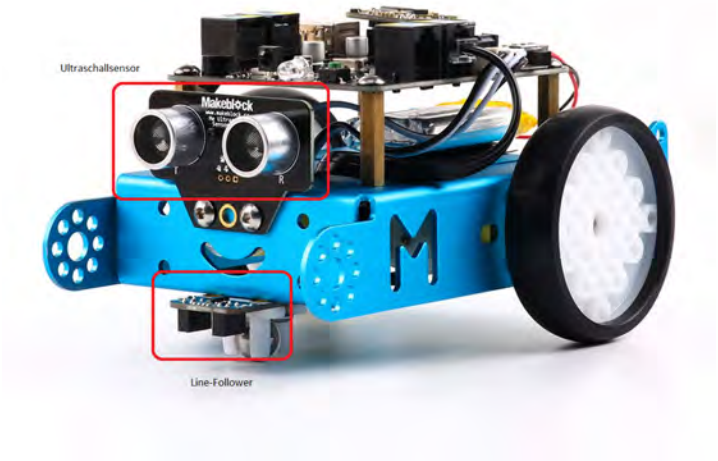
Neben den Motoren und Ausgabegeräten besitzt der mBot auch noch verschiedene Sensoren. Ein Sensor kann auch als Detektor, Aufnehmer oder Fühler bezeichnet werden und ist ein technisches Bauteil, welches bestimmte physikalische oder chemische Eigenschaften (z.B. Temperatur, Feuchtigkeit, Helligkeit, Schallfeldgrößen, usw.) erfassen kann.

Der mBot hat fünf verschiedene Sensoren: einen Lichtsensor, mit dem die Helligkeit gemessen werden kann, einen Ultraschallsensor, mit dem die Distanz zu einem Objekt gemessen werden kann, einen Druckknopf (Button), einen Linienverfolger, mit dem der mBot einer schwarzen Linie folgen kann sowie einen Infrarot-Sensor, welcher Befehle der mitgelieferten Infrarot-Fernbedienung aufnimmt.

Den Button, Infrarot- und Lichtsensor findest Du auf der Platine:



Den Ultraschallsensor und Line-Follower findest du vorne am mBot:



Bei den nachfolgenden Sensoren lernst Du die verschiedenen Sensoren genauer kennen und lernst, wie man auf die gemessenen Werte der Sensoren zugreift. Wir werden diese zunächst nur in *Scratch* auf der Bühne anzeigen lassen. Im Abschnitt zu den kombinierten Aufgaben werden die ermittelten Werte anschließend verarbeitet und

darauf reagiert. Zum Beispiel soll vor einem Hindernis abgebremst und gewendet werden.

Für die nachfolgenden Aufgaben ist es wichtig, dass auf dem mBot die Original-Firmware läuft. Ist dies nicht der Fall, oder bist Du Dir nicht mehr sicher, kannst Du über *Verbinden* → *Default Programm zurücksetzen* das Standardprogramm wiederherstellen.

In den nachfolgenden Aufgaben benötigen wir unter anderem die beiden Blöcke:

3.4.1 Aufgabe 1 – Der Lichtsensor

Mit Hilfe des Lichtsensors kann die Helligkeit in der Umgebung des mBot gemessen werden. Der Lichtsensor ist eine einfache Möglichkeit die Helligkeit eines Raumes zu messen. Dazu wollen wir erst mal herausfinden, welche Werte der Lichtsensor zurückgibt. Wir greifen dazu mit einem geeigneten Block auf den Lichtsensor zu und geben die gemessenen Werte in *Scratch* aus.

Dazu verwenden wir eine Endlos-Wiederholung 

. Der Wert des Lichtsensor kann mit Hilfe des Blockes 

ausgelesen werden. Um den Wert anzeigen zu können, kannst Du diesen Block mit dem Block 

wie folgt kombinieren:

Nachdem das Programm gestartet wurde, sollte der Panda eine relativ konstante Zahl ausgeben. Notiere nun für folgende Fälle die Ausgabewerte des Lichtsensors:

(a) Dunkle den Lichtsensor mit der Hand ab.

(b) Versuche den Lichtsensor ganz abzudunkeln. Das gelingt z.B. sehr gut mit der Kappe eines Textmarkers.

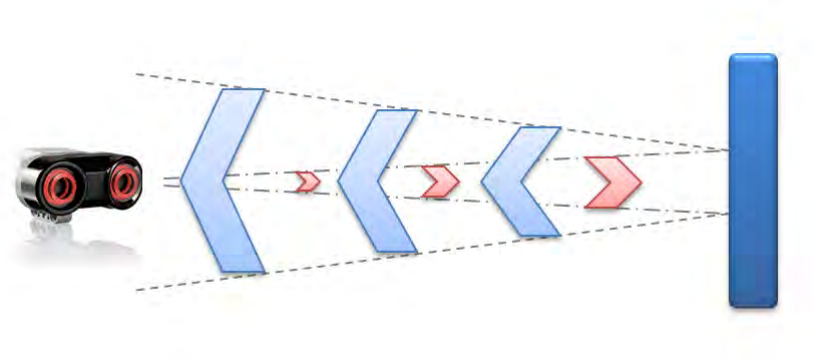
(c) Leuchte den Lichtsensor mit einer starken Lichtquelle an, z.B. mit der Taschenlampe Deines Smartphones.

Was für Zahlenwerte gibt der Lichtsensor Deiner Meinung nach zurück?

3.4.2 Aufgabe 2 – Der Ultraschallsensor

Mit Hilfe des Ultraschallsensors kann die Distanz (in cm) zu einem Objekt gemessen werden. In nachfolgendem Kasten findest Du eine Beschreibung der Funktionsweise eines Ultraschallsensors.

Funktionsweise eines Ultraschall-Sensors



Ein Ultraschallsensor lässt sowohl das Senden als auch das Empfangen von Schallimpulsen zu und kann damit Entfernungen messen. Nachdem der Ultraschallsensor einen Schallimpuls ausgesendet hat, wartet er auf den Empfang dieses Schallimpulses. Der ausgesendete Schallimpuls wird von einem Objekt reflektiert und kann somit vom Ultraschallsensor empfangen werden. Anhand der Zeit, die zwischen Aussenden und Empfangen des Schallimpulses vergeht, berechnet der Ultraschallsensor die Entfernung zu dem detektierten Gegenstand. Fledermäuse orientieren sich mit dem gleichen Prinzip in der Dunkelheit bei der Beutejagd.

Beachte, dass der Ultraschallsensor etwas ungenau misst. Hin und wieder liefert er auch mal komplett unbrauchbare Werte.

Auch hier wollen wir erst mal herausfinden, welche Werte der Ultraschallsensor zurückgibt. Wir greifen dazu mit einem geeigneten Block auf den Ultraschallsensor zu und geben die gemessenen Werte in *Scratch* aus.

Neue Blöcke:
Dazu benötigen wir folgenden neuen Block:

Beachte hier, dass Du den Port angibst, an dem der Ultraschallsensor mit der Platine verbunden ist!

3.4.3 Aufgabe 3 – Button

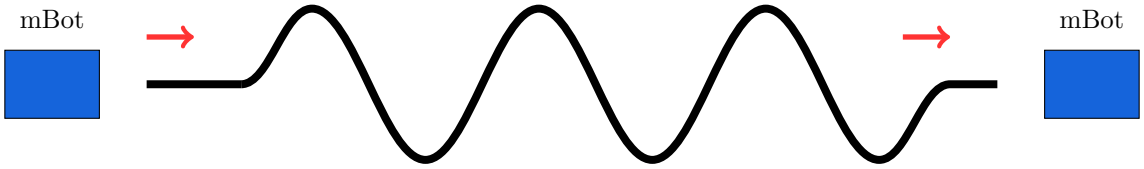
Der Druckknopf (Button) bietet uns die Möglichkeit auf die Benutzeraktion *Drücken des Buttons* zu reagieren. In dieser Aufgaben soll über den Buzzer eine halbe Note C4 ausgegeben werden, wenn der Button gedrückt wird. Diese Aufgabe beinhaltet zum ersten Mal das Warten auf eine bestimmte Aktion. Hierfür benötigst du folgende neuen Blöcke:

Experimentiere mit diesen beiden Blöcken. Was passiert, wenn Du den Button gedrückt hältst?

Du wirst feststellen, dass der Ton nicht ganz abgespielt wird, weil schon wieder der nächste Tastendruck registriert wird. Finde heraus, wie Du dies verhindern kannst!

3.4.4 Aufgabe 4 – Line Follower

Mit Hilfe des Line-Followers kann der mBot eine schwarze Linie „verfolgen“. Er fährt somit auch Schlangenlinien:



Der Sensor gibt die Werte 0,1,2 und 3 zurück. Die Bedeutung der Werte ist in der folgenden Tabelle dargestellt:

Wert	Bedeutung
0	Beide auf Linie (schwarz)
1	Nur links auf Linie
2	Nur rechts auf Linie
3	Keiner auf Linie

Tabelle 3.1: Werte des Line Followers

Versuche wieder wie bei den vorangehenden Aufgaben den Wert des Sensors auszulesen und in *Scratch* auszugeben. Suche dabei zunächst nach einem geeigneten Block!

Um den Sensor zu testen kannst Du Deine Hand oder ein Buch davor halten und vom mBot wegbewegen bzw. wieder hinbewegen. Beachte jedoch, dass der Roboter eine Fehlmessung liefert, wenn Du zu nah an den Sensor kommst.

3.5 Einfache kombinierte Aufgaben

3.5.1 Variablen

Für die folgenden Aufgaben benötigen wir Variablen[3]. Eine Variable kann in *Scratch* eine Gleitkommazahl[4] speichern. Wir werden meist aber sowieso nur mit natürlichen Zahlen rechnen.

In diesem Abschnitt lernst Du, wie Du Variablen erstellst sowie welchen Vorteil sie bieten.

Eine Variable erstellst du mit einem Klick auf *Daten & Blöcke* → *Neue Variable* :

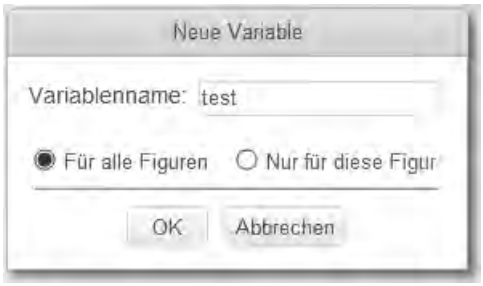


Abbildung 3.7: Erstellen einer neuen Variable

Für die Programmierung des mBot wählen wir hier immer *Für alle Figuren* aus.

Beim Arbeiten mit Variablen sollte diesen grundsätzlich am Anfang ein Wert zugewiesen werden. Bei *Scratch* wird der Wert automatisch auf *0* gesetzt, jedoch musst Du später bei der Arduino-Programmierung selbst dafür sorgen, dass ihr ein Wert zugewiesen wird. Man spricht in diesem Zusammenhang auch davon, dass die Variable *initialisiert* wird.
Folglich weisen wir auch bei der Programmierung mit *Scratch* bereits jeder Variable einen Anfangswert zu:

Den Wert, der in einer Variable zwischengespeichert ist, kannst Du mit dem Block 

ändern.

3.5.2 Aufgabe 1 - Zählen

Aufgabenstellung

Der Roboter soll zählen, wie oft der Button gedrückt wird. Sobald der Lichtsensor abgedunkelt wird, soll der Buzzer so viele Töne ausgeben, wie Tastendrücke gezählt wurden.

Beispiel:

Das Programm wird gestartet und der Schalter auf der Platine des mBot wird fünfmal gedrückt. Anschließend wird der Lichtsensor mit dem Deckel eines Textmarkers verdunkelt. Der mBot spielt nun fünf Töne ab.

Neue Blöcke:

Für diese Aufgabenstellung benötigst Du möglicherweise folgenden Blöcke:

[3] Du kannst Dir eine Variable als eine Art Zwischenspeicher vorstellen, dem Du einen bestimmten Wert zuweisen kannst. Diesen Wert kannst du später mit Hilfe der Variable wieder verwenden. Benötigst Du den Wert nicht mehr, kannst Du in der Variable auch einen neuen Wert abspeichern.

[4] Eine Gleitkommzahl (oder auch Fließkommazahl) ist eine *approximative* (näherungsweise) Darstellung einer reellen Zahl. Die Menge der Gleitkommazahlen ist eine Teilmenge der rationalen Zahlen. Gleitkommazahlen spielen vor allem bei Rechnungen, die von Computern ausgeführt werden, eine wichtige Rolle. Für uns reicht die Vorstellung einer Zahl mit Nachkommastellen aus.

und

3.5.3 Aufgabe 2 - Nachtlicht

Verwende den Roboter um bei einer geringen Umgebungshelligkeit das Licht (LEDs) einzuschalten. Sobald es wieder hell wird, soll die LED ausgehen.

TIPP

Du kannst zum Abdecken des Lichtsensors zum Beispiel die Kappe eines Stifts verwenden.

3.5.4 Aufgabe 3 - Lichtschalter

Jetzt wollen wir einen Lichtschalter simulieren. Hierzu benötigen wir wieder eine Variable "istAn". Diese Variable soll nur zwischen den Werten '1' für 'An' und '0' für 'Aus' wechseln. Abhängig von dieser Variable werden dann die LEDs angesteuert.

3.5.5 Aufgabe 4 - Stille Post

Nun soll eine abgewandelte Version von stiller Post implementiert werden, indem der Roboter eine zufällige Zahl zwischen eins und fünf auswählt und diese dann mittels blauem Blinken anzeigt. Sobald der mBot fertig mit der Anzeige ist, soll der Nutzer den Knopf genau so oft drücken, wie der Roboter aufgeblinkt hat. Die Eingabe wird mit dem Abdunkeln des Lichtsensors bestätigt. Wenn die vom mBot gewählte Zahl mit der vom Nutzer eingegebenen übereinstimmt, sollen die LEDs grün leuchten, wenn die Eingabe falsch war rot.

Neue Blöcke: 

3.5.6 Aufgabe 5 - Abstandsmessung

Lasse den mBot einen Warnton ausgeben, wenn der Ultraschallsensor einen Wert kleiner fünf zurückgibt.

3.5.7 Aufgabe 6 - Bremsen

Fahre langsamer, desto näher du einem Hindernis kommst. Vor dem Objekt soll der Roboter anhalten. Für diese Aufgabe soll Stufenhaft gebremst werden. Eine mögliche Zuordnung von Entfernung und Geschwindigkeit ist unten aufgelistet.

Geschwindigkeit	Entfernung MIN	Entfernung MAX
255	51	10000
100	31	50
50	11	30
0	0	10

Alternativ kann auch eine mathematische Funktion wie zum Beispiel $speed(x) := \frac{1}{2}(x-5)$ verwendet werden. Hier muss aber auf Randwerte kleiner Null beziehungsweise größer 255 geachtet werden.

Wer es mathematisch mag, kann auch die (zweistellige) *min* und *max* Funktion implementieren um dann folgende Gleichung zu verwenden: $speed(x) := \min\{255, \max\{\frac{1}{2}(x-5), 0\}\}$

3.5.8 Aufgabe 7 - Vermessung

Merke dir den Anfangsabstand und fahre vorwärts bis der Knopf gedrückt wird. Messe nun wieder und gib die zurückgelegte Distanz mit dem  Block aus.

Alternative

Du kannst den gerundeten Wert auch mit den LEDs oder dem Buzzer ausgeben.

3.5.9 Aufgabe 8 - Fahre und Erkenne

Lasse den Roboter im Raum umherfahren und weiche Hindernissen, die vom Entfernungssensor erkannt werden aus, indem du den Roboter drehen lässt.

3.5.10 Aufgabe 9 - Linien zählen

Der mBot soll während er vorwärts fährt mittels dem Line-Follower schwarze Linien zählen. Sobald für zwei Sekunden keine Linie mehr gefunden wird, soll der Roboter anhalten und die Anzahl mit dem Buzzer ausgeben.

Bei dieser Aufgabe benötigst du die Stoppuhr. Um brauchbare Werte zu erhalten, solltest du diese am Anfang (und wann immer nötig) zurücksetzten.

Versuche diese Aufgabe erst mit gleich dicken Linien. Danach probiere es mit unterschiedlichen, vermischten Linienstärken. Bis zu welcher Linienbreite erfasst der Roboter diese noch? Wann nicht mehr?

3.5.11 Aufgabe 10 - Sumo Ring

Drucke auf einem Blatt Papier (mindestens A3) einen schwarzen Kreis aus. Setzte den Roboter in diesen und lasse ihn umherfahren. Wenn der mBot die Linie sieht, soll er rückwärts nach links (rechts) drehen und dann weiterfahren.

Tipp

Sollte der Roboter trotz des Drehens aus dem Kreis fahren, versuche nach der Kurve noch kurz rückwärts zu fahren. Auf diese Art funktioniert das Experiment auch mit einem A4 Blatt.

3.5.12 Aufgabe 11 - Tanz

In dieser Aufgabe wollen wir den Roboter zum "tanzen" bringen. Erst soll auf einen Tastendruck gewartet werden. Sobald diese gedrückt wurde, soll der Roboter jede Sekunde neu entscheiden, ob er rechts oder linksherum dreht. Sobald der Lichtsensor abgedunkelt wird, soll er stehen bleiben.

3.5.13 Aufgabe 12 - Folge einer Linie

Drucke einen Pfad auf einem großem Blatt Papier aus. Ziel der Aufgabe ist es jetzt, den mBot diesem folgen zu lassen.

3.5.14 Aufgabe 13 - Rennfahrer

Verwende 3.5.13 Aufgabe 12 - Folge einer Linie und baue die Stoppuhr so ein, dass diese die Zeit misst, die der Roboter für das Befahren des Pfades benötigt. Gib das Ergebnis der Messung am Schluss auf dem Desktop aus. Vergleicht eurer Roboter und den Quellcode um festzustellen, warum einer schneller ist als der andere.

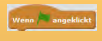
3.5.15 Aufgabe 14 - Rennfahrer (nativ)

Bisher haben wir immer den Block  verwendet. Versuche nun in 3.5.13 Aufgabe 12 - Folge einer Linie diesen Block mit dem  Block zu ersetzen.

Um dieses Programm zu übertragen muss das USB Kabel verwendet werden.

Miss nun mit einer externen Stoppuhr, wie lange der Roboter jetzt benötigt um die Linie zu befahren. Hat sich etwas verändert?

Hinweis

Wenn du bei anderen Programmen wieder den  Block verwenden willst, musst du zuerst den Roboter mit dem USB Kabel verbinden und dann auf *Verbinden to Default Programm zurücksetzten* klicken.

3.5.16 Aufgabe 15 - Fernsteuerung

Jetzt wollen wir die Fernbedienung verwenden, um den mBot zu steuern. Sobald mit der Fernbedienung gearbeitet wird, muss ein natives mBot Programm geschrieben werden.

Die Aufgabe ist jetzt, den Roboter mittels der Pfeiltasten zu steuern.

uf Polizei-

3.5.17 Aufgabe 16* - Polizeiauto

Verwende die Fernsteuerung aus 3.5.16 Aufgabe 15 - Fernsteuerung und erweitere diese so, dass mit einem Druck auf 'A' das Blaulicht und auf 'B' das Horn ein- und ausgeschaltet werden kann.

Ne-

Kapitel 4

Programmierung mit Scratch4Arduino - Lösungen

Überblick:

4.1 Motoren

4.1.1 Aufgabe 1 – Ansteuern eines Motors

Abbildung 4.1: Eine mögliche Lösung

4.1.2 Aufgabe 2 – Fahre vorwärts

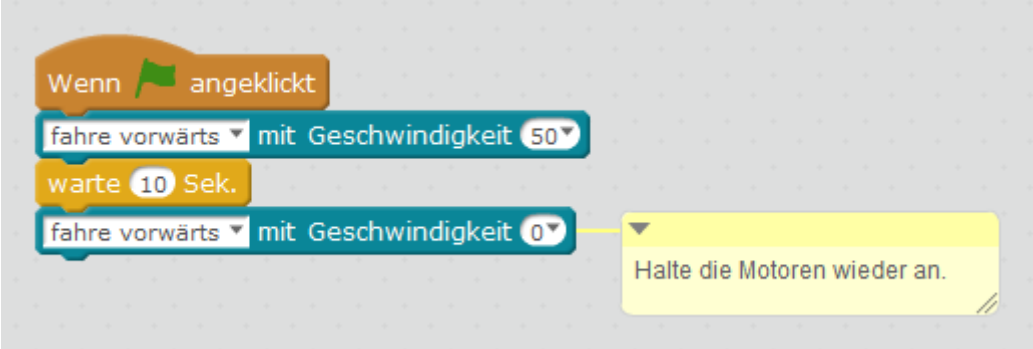


Abbildung 4.2: Die einfache Lösung

4.1.3 Aufgabe 3 – Fahre ein Quadrat ab

Abbildung 4.3: Fahre ein Quadrat

4.1.4 Aufgabe 4 – Wiederverwendbarkeit

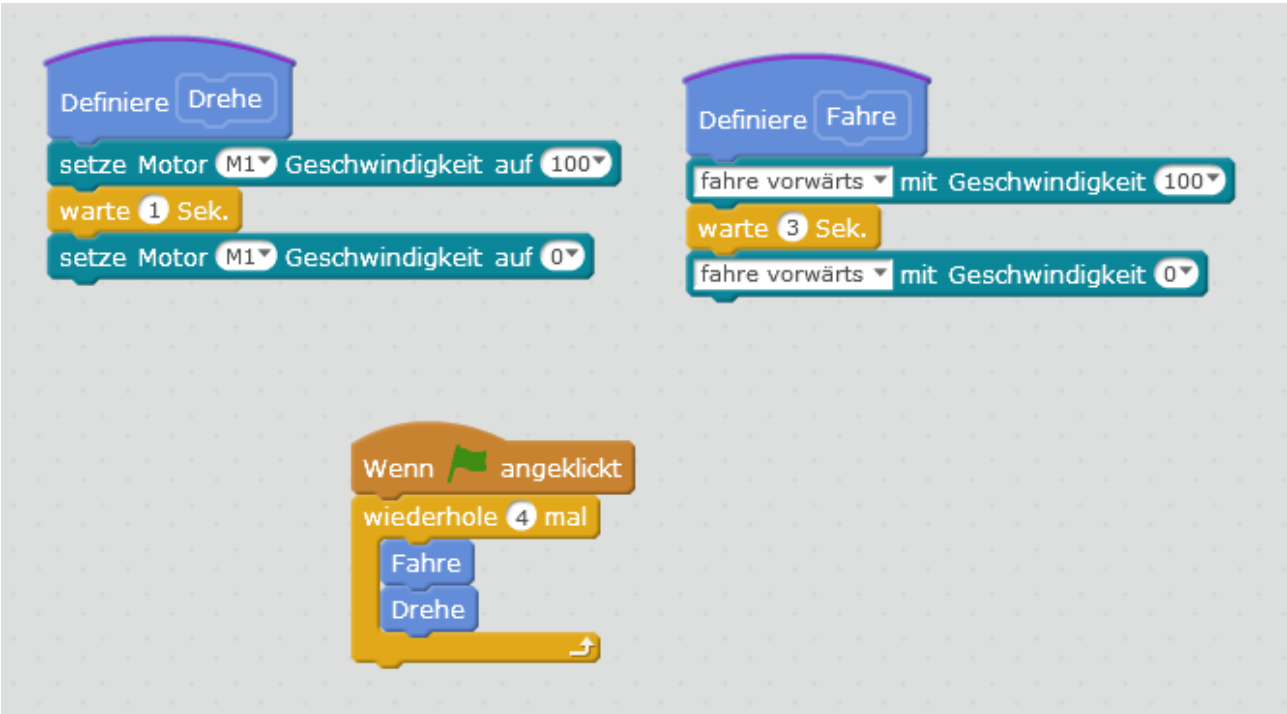


Abbildung 4.4: Fahre ein Quadrat mit Blöcken

4.2 Ausgabegeräte

4.2.1 Aufgabe 1 – LED

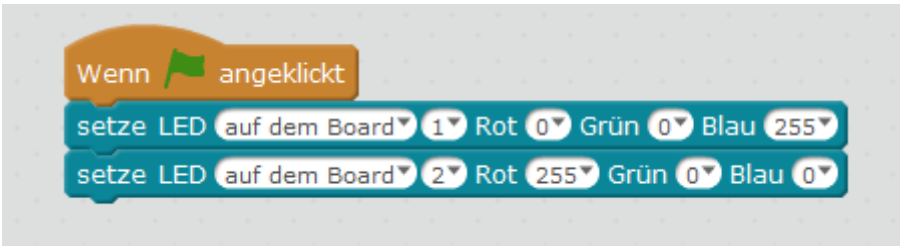


Abbildung 4.5

4.2.2 Aufgabe 2 – LED Farbmischung

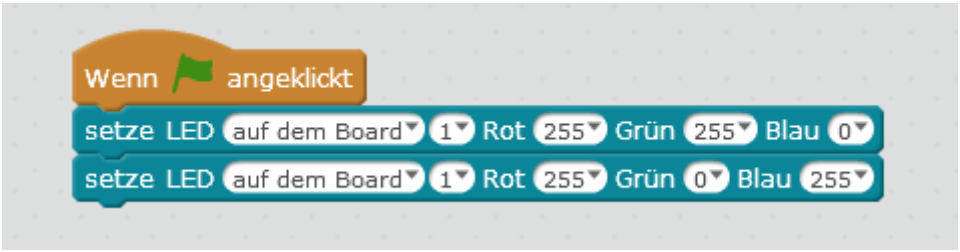


Abbildung 4.6

4.2.3 Aufgabe 3 – Der Buzzer

Abbildung 4.7

4.3 Sensoren

4.3.1 Aufgabe 1 – Der Lichtsensor

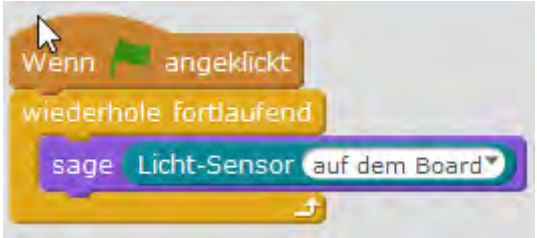


Abbildung 4.8

4.3.2 Aufgabe 2 – Der Ultraschallsensor

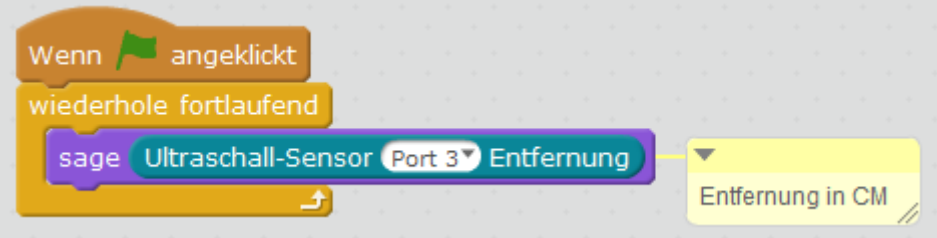


Abbildung 4.9

4.3.3 Aufgabe 3 – Button

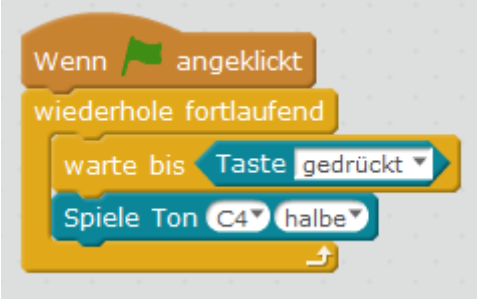


Abbildung 4.10

4.3.4 Aufgabe 4 – Line Follower

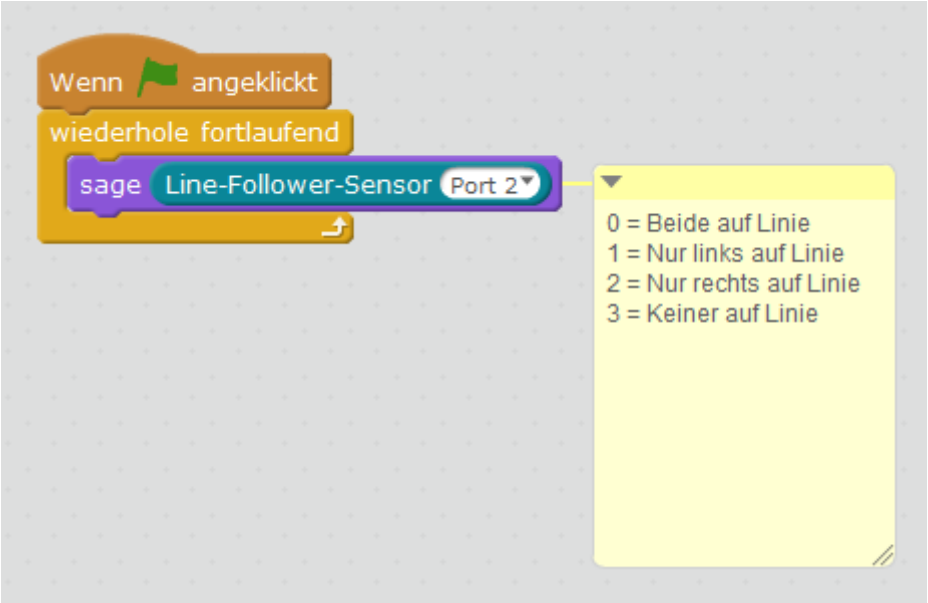


Abbildung 4.11

4.4 Einfache kombinierte Aufgaben

4.4.1 Aufgabe 1 - Zählen

Abbildung 4.12

4.4.2 Aufgabe 2 - Nachtlicht

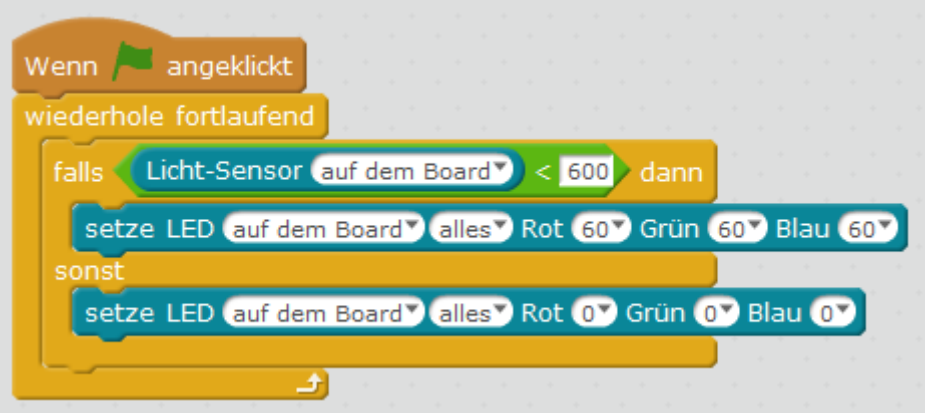


Abbildung 4.13

4.4.3 Aufgabe 3 - Lichtschalter

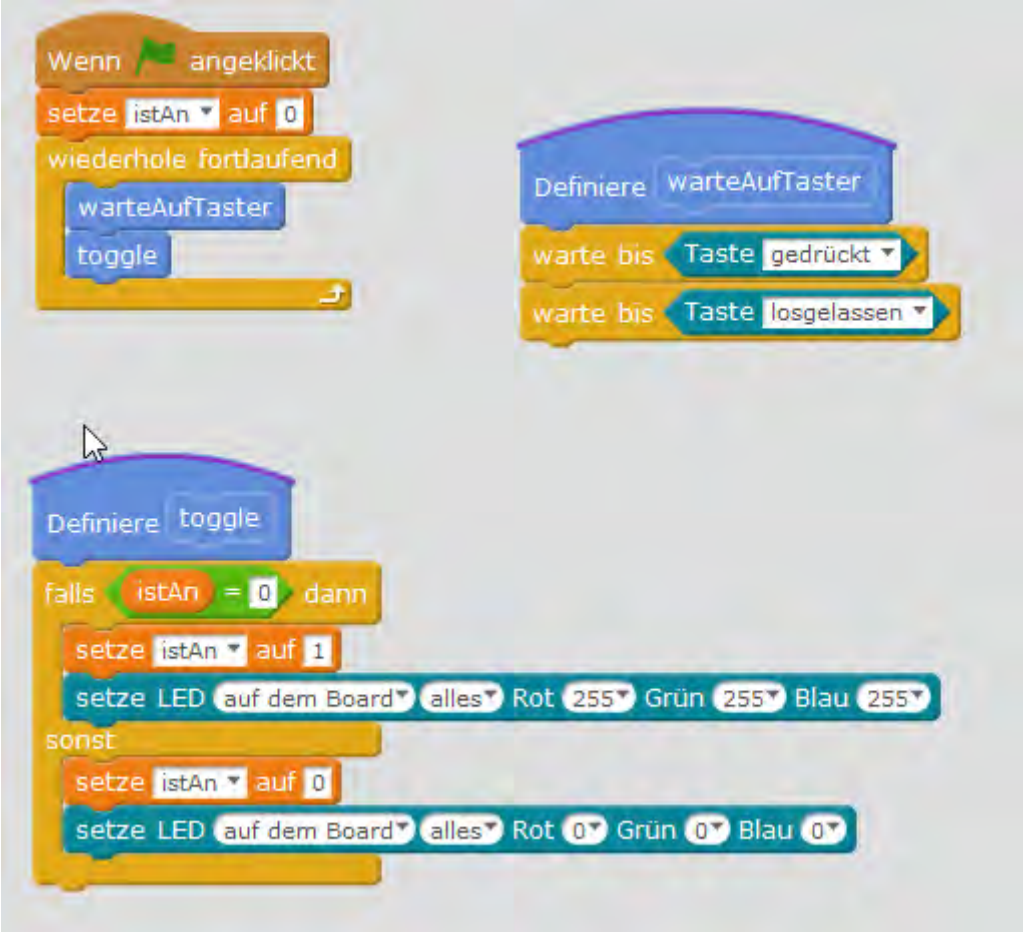


Abbildung 4.14

4.4.4 Aufgabe 4 - Stille Post

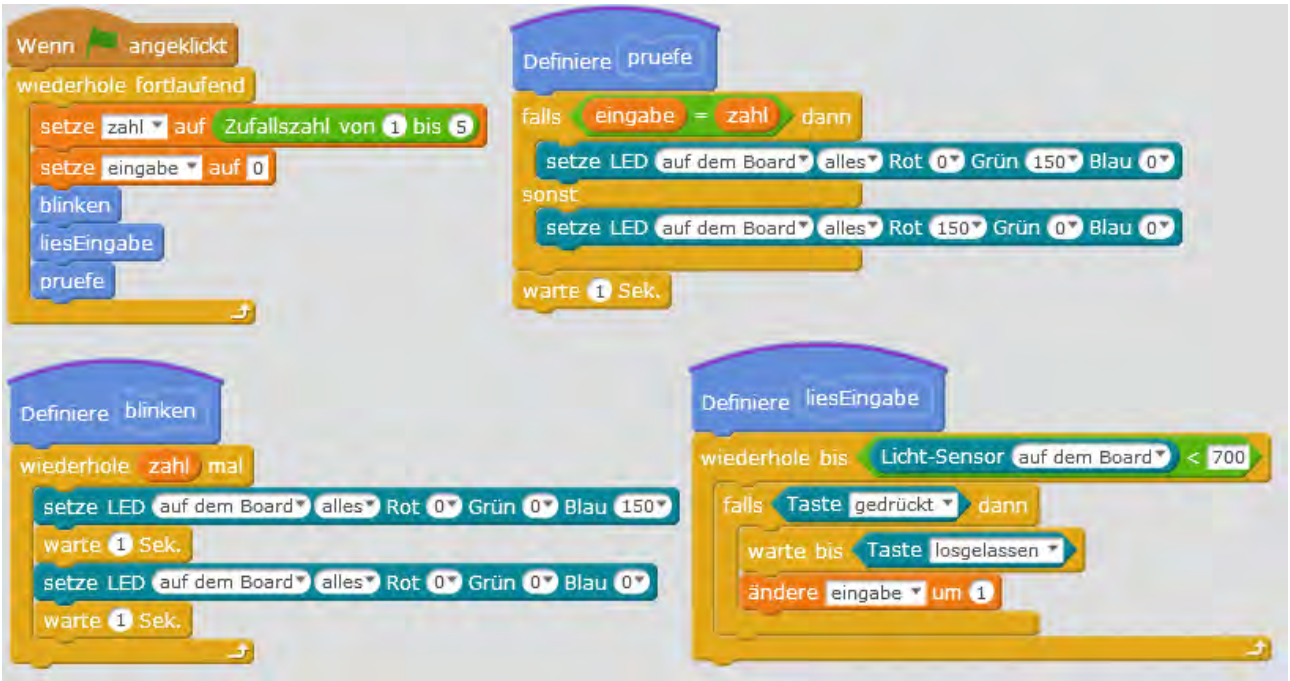


Abbildung 4.15

4.4.5 Aufgabe 5 - Abstandsmessung

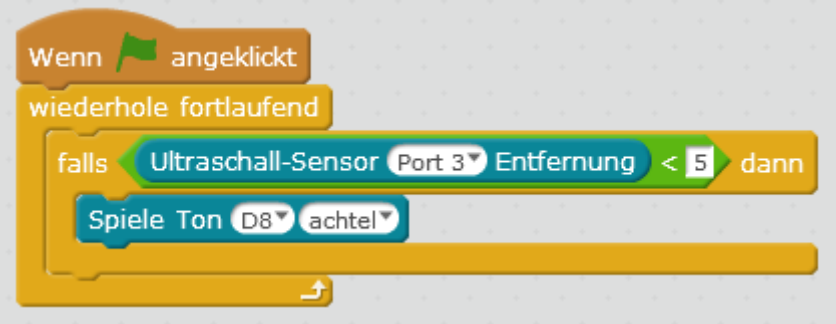


Abbildung 4.16

4.4.6 Aufgabe 6 - Bremsen

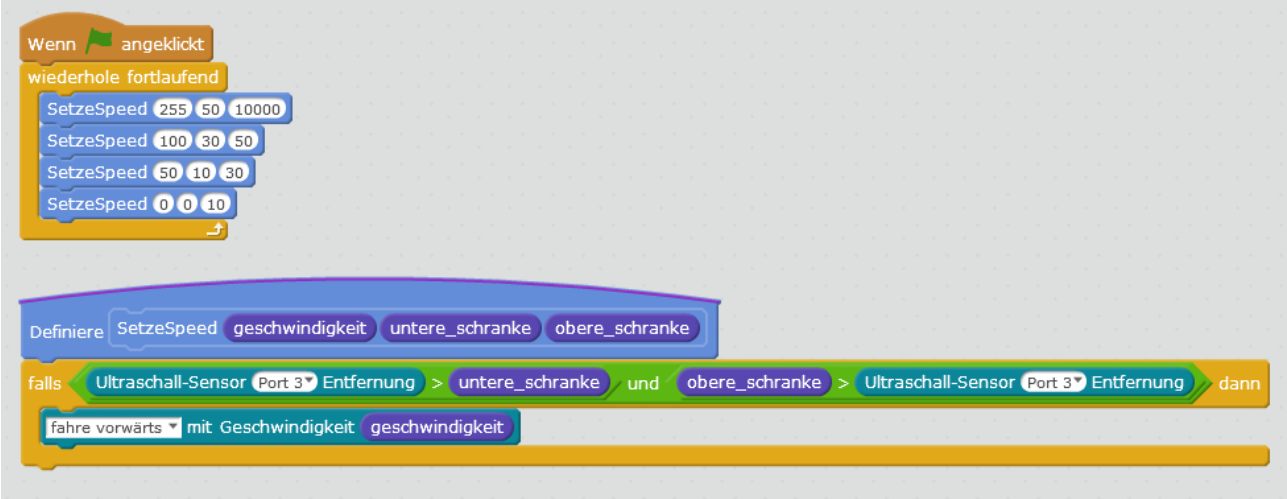


Abbildung 4.17

4.4.7 Aufgabe 7 - Vermessung

Abbildung 4.18

4.4.8 Aufgabe 8 - Fahre und Erkenne

Abbildung 4.19

4.4.9 Aufgabe 9 - Linien zählen

Abbildung 4.20

4.4.10 Aufgabe 10 - Sumo Ring

Abbildung 4.21

4.4.11 Aufgabe 11 - Tanz

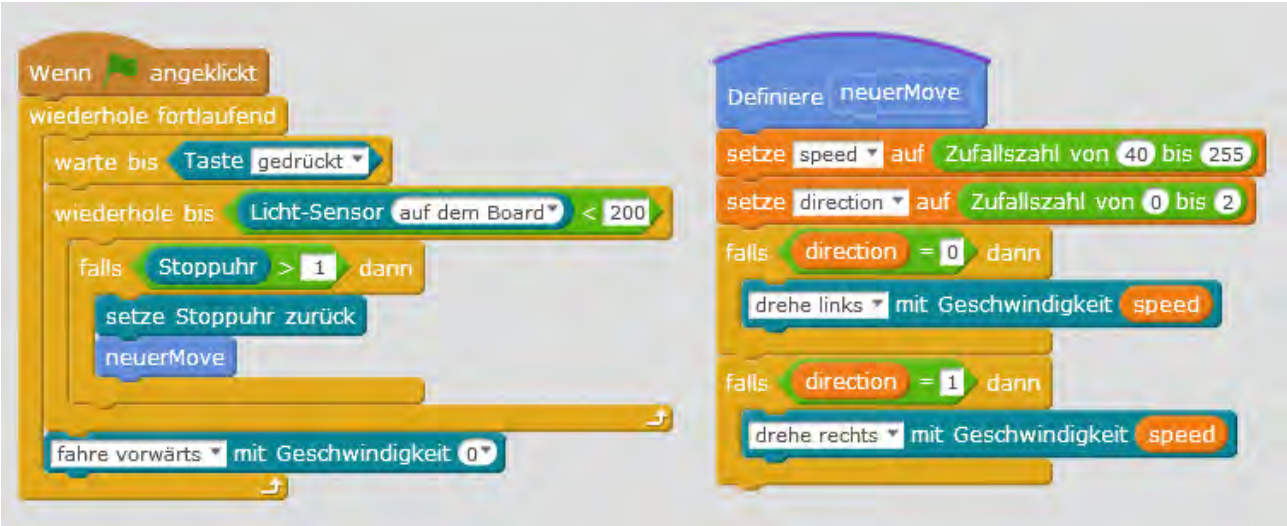


Abbildung 4.22

4.4.12 Aufgabe 12 - Folge einer Linie

Abbildung 4.23

4.4.13 Aufgabe 13 - Rennfahrer

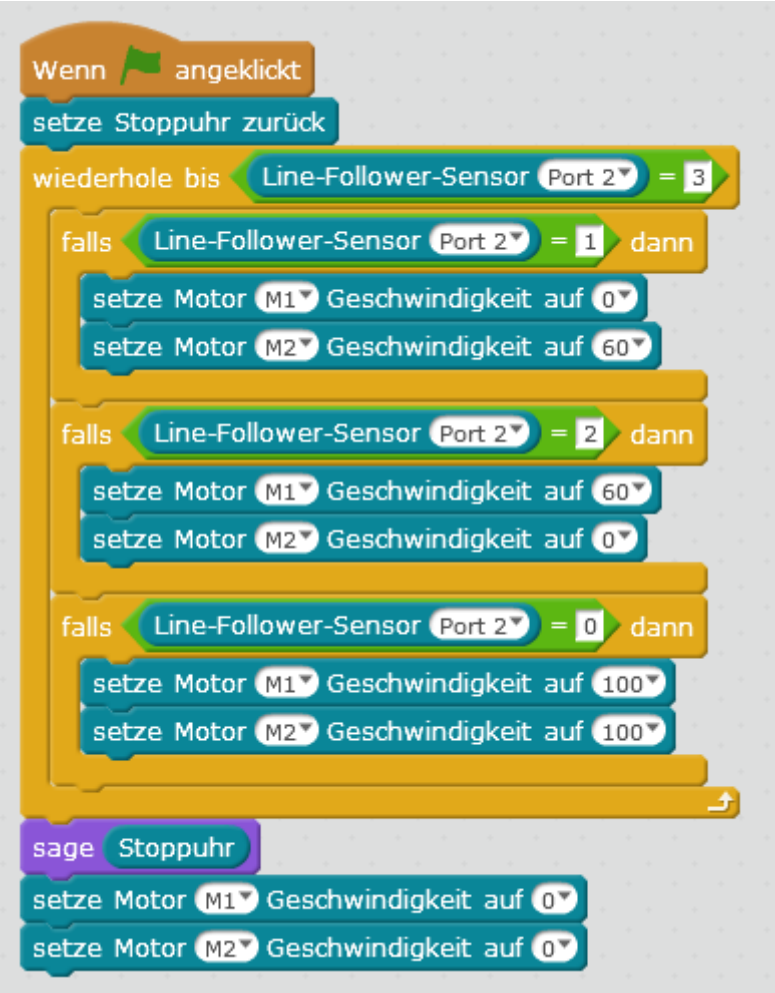


Abbildung 4.24

4.4.14 Aufgabe 14 - Rennfahrer (nativ)

In diesem Experiment solltest du nun festgestellt haben, dass der Roboter etwas schneller ist, als zuvor. Dies liegt daran, dass der mBot jetzt selbstständig arbeitet, während er zuvor immer mit dem PC kommuniziert hat. Da wir nun die Kommunikation nur noch zum hochladen nutzen, ergeben sich gewisse Einschränkungen: Du kannst Blöcke, die auf der Bühne von Scratch operieren nicht mehr verwenden. Dies betrifft unter anderen auch den 

. Der Vorteil ist natürlich, dass der PC nur noch zum programmieren benötigt wird.

4.4.15 Aufgabe 15 - Fernsteuerung

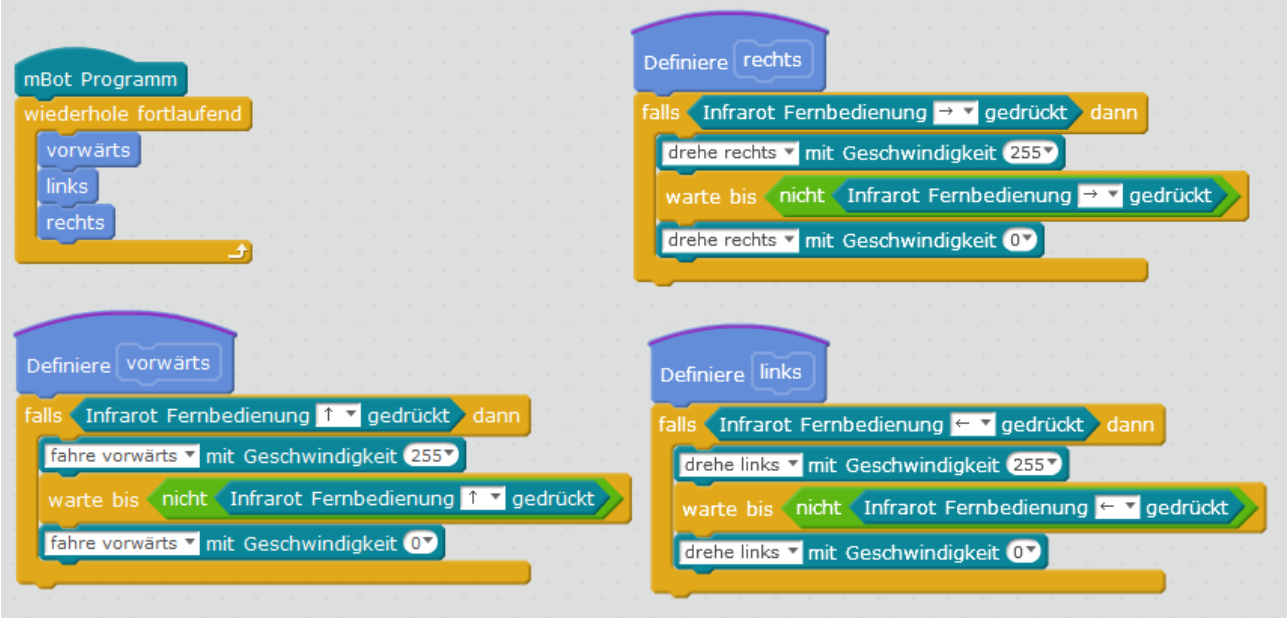


Abbildung 4.25

4.4.16 Aufgabe 16* - Polizeiauto

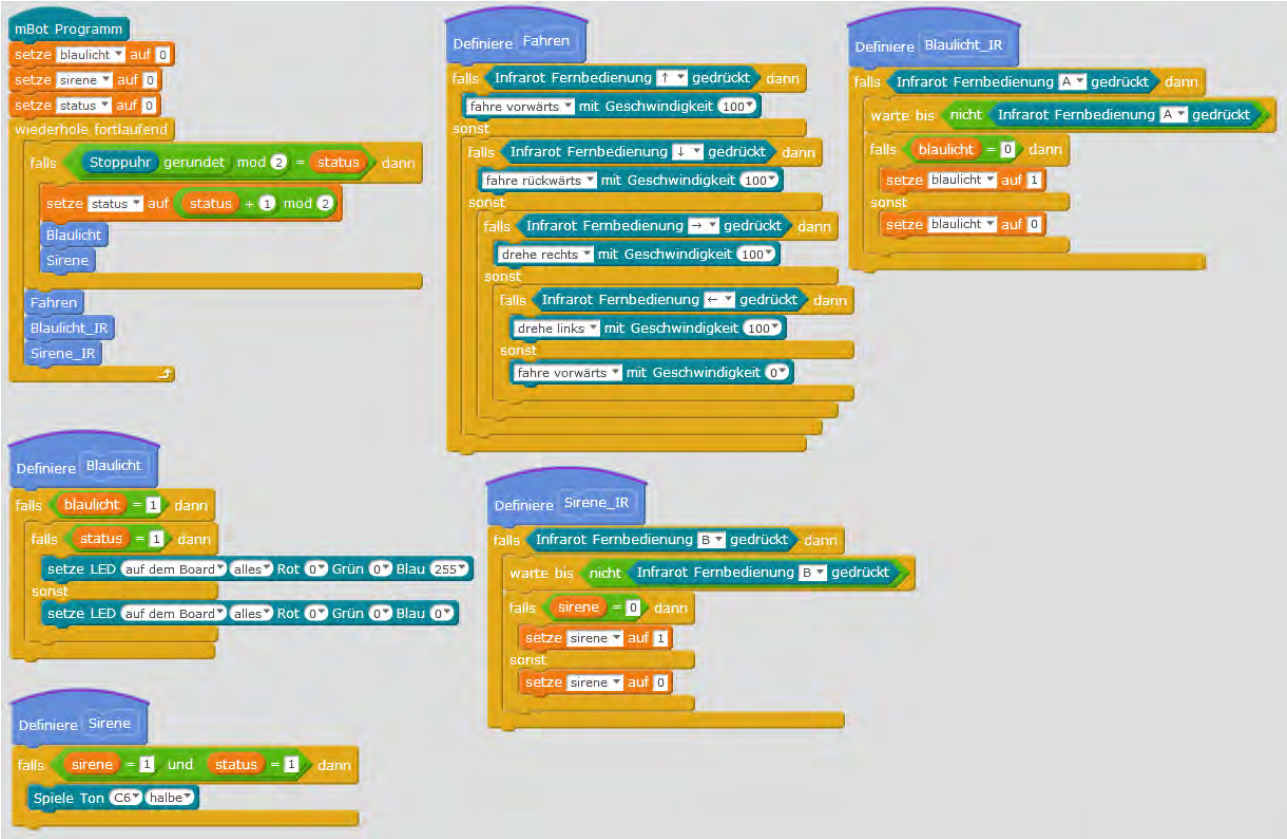


Abbildung 4.26

Kapitel 5

Programmierung mit Arduino-C

Überblick:

5.1 Arduino-C

5.2 Motoren

5.2.1 Aufgabe 1 – ...

5.2.2 Aufgabe 2 – ...

5.3 Ausgabegeräte

5.3.1 Aufgabe 1 – ...

5.3.2 Aufgabe 2 – ...

5.4 Sensoren

5.4.1 Aufgabe 1 – ...

5.4.2 Aufgabe 2 – ...

...