

Yann Dos Santos

Arduino Roboter

Maturaarbeit in Fach Informatik mit Mathias Bolli

Frauenfeld Kantonsschule 2020

# Zusammenfassung der Kapitel des Dossiers

## Kapitel 1: Das Robotergehirn

Das erste Kapitel ist der Gründung der IDE Arduino und ihrer Funktionsweise gewidmet. Die IDE[[1]](#footnote-1) dient als Brücke zwischen dem Programmierer und dem Roboter. Es ermöglicht Ihnen, Code in der Sprache C einzugeben, um ihn zu kompilieren und dann über die serielle Schnittstelle an das Arduino zu senden.

Die erste Übung in der Mappe besteht darin, die Arduino-Karte einzuschalten und an den Computer anzuschließen und den Roboter anzuweisen, “Hello World“ auf dem seriellen Monitor anzuzeigen.

Alles, was Sie dem Roboter befehlen wollen, wird zuerst von der Arduino-IDE kompiliert und dann über die serielle Schnittstelle als Bits an den Roboter gesendet. Die serielle Schnittstelle kann zum Senden und Empfangen von Daten verwendet werden, die Geschwindigkeit, mit der Daten gesendet werden, wird als „Baudrate“ bezeichnet (Bits[[2]](#footnote-2) pro Sekunde). In dieser Übung haben wir die serielle Schnittstelle mit dem Befehl „Serial**.**begin**(**9600**)**“ bei einer Geschwindigkeit von 9600 Baud initialisiert und dann den Befehl „Serial**.**println**(**"Hello world! \n"**)**“ verwendet, um Text an den seriellen Monitor zu senden.

Wir werden später sehen, dass der Arduino-Code in zwei Teilen organisiert ist:

* Das "Setup"-Verfahren: In diesem Teil wird der Code nur einmal zu Beginn des Programms ausgeführt. In diesem Teil werden wir die Erklärungen abgeben und die serielle Schnittstelle und die Eingänge des Roboters initialisieren.
* Das "Schleifen"-Verfahren: In diesem Teil wird der Code auf unbestimmte Zeit wiederholt. Hier werden wir das Hauptprogramm schreiben, im Allgemeinen wollen Roboter Schleifenaufgaben ausführen.

Um Variablen zu verwenden und zu speichern, müssen wir sie zunächst deklarieren und angeben, welchen Typ wir verwenden wollen. Der Mikrocontroller speichert diese Daten dann in binärer Form in seinem RAM[[3]](#footnote-3). Es gibt viele Formen von Variablen, aber hier sind die häufigsten:

* Die "int": Sie enthalten eine ganze Zahl zwischen -32768 und 32767, wenn unser Wert nicht in diesem Bereich liegt, erscheint ein Fehler.
* Das "char": Sie stellen ein Zeichen in Form einer Zahl dar, dank des ASCII-Standards (American Standard Code for Information Exchange).
* Der "float": Sie enthalten eine gerundete Dezimalzahl zwischen ≈ - 3.4 ∙ 1038 und ≈ 3.4 ∙ 1038

Um eine Variable zu definieren, schreiben Sie "Type Name = Wert":

int a **=** 52**;**

Oder wir können zuerst den Typ der Variable definieren und ihr dann ihren Wert zuweisen. Diese Methode ist praktischer, da Sie den Wert jederzeit nur mit seinem Namen ändern können, anstatt zum gesamten Code zurückzugehen, um ihn zu ändern.

int a**;**

a **=** 52**;**

Sie können dann mit diesen Variablen Operationen wie Addition, Subtraktion, Division, Multiplikation, Modulo usw. durchführen.

In jedem Programm ist es notwendig, Bedingungen zu verwenden, das sind Teile des Codes, die ausgeführt werden, wenn eine Voraussetzung erfüllt ist, zum Beispiel eine gedrückte Taste oder ein bestimmter Variablenwert. Wir werden die Funktionen "if" und "else" verwenden, die zunächst bewerten, ob die Bedingung zwischen den Klammern wahr ist, und dann entscheiden, ob der Code zwischen den Klammern ausgeführt werden soll oder nicht. Wenn die Bedingung der "if"-Funktion falsch ist, führt das Programm den Code der "else"-Funktion aus.

**if(**a **>** b**)** //Wenn die Bedingung wahr

**{** // Ausführen

Serial**.**print**(**"a ist großer als b"**);**

**}**

**else** // Ansonsten

**{** // Ausführen

Serial**.**print**(**"b ist großer als a"**);**

**}**

Wir haben dann gelernt, Schleifen mit der "while"-Funktion zu verwenden, da das Programm bei "if" auswertet, ob die Bedingung zwischen den Klammern wahr ist, und solange sie wahr ist, wird der Code zwischen den Klammern ausgeführt, im Gegensatz zu "if", wo der Code nur einmal ausgeführt wurde. Zum Beispiel wird eine "while (true)"-Schleife ihren Code unendlich oft ausführen.

**while** **(true){**

Serial**.**println **(**"Diese Aussage ist wahr"**);**

**}**

Schließlich können Sie, um den Code zu organisieren, Kommentare (grüner Text) verwenden. Diese Kommentare sind notwendig, um zu dokumentieren, was der Code bewirkt, z.B. wenn es sich um ein großes Projekt handelt, ist es nützlich, sich an das zu erinnern, was man getan hat, oder wenn man den Code an jemand anderen weitergeben muss. Kommentare werden nie an das Programm gesendet.

Das folgende Programm veranschaulicht alles, was wir in diesem ersten Kapitel gelernt haben:

//Aussagen

const int SerialBaud **=** 9600**;**

//Geschwindigkeit der Informationsübertragung zwischen dem Arduino und dem seriellen Monitor

int a **=** 52**;**

//Erstellt einen Variable a vom Typ int und gibt ihm den Wert 63.

int b **=** 10**;**

//Erstellt einen Variable b vom Typ int und gibt ihm den Wert 10.

float c **=** sqrt**(**2.0**);**

// Erstellt einen Variable c vom Typ float und gibt ihm den Wert der Quadratwurzel von 2.

char d **=** 'x'**;**

//Erstellt eine Variable a vom Typ char und gibt ihr den Wert "m".

int Addition**,** Subtraktion**,** Multiplikation**,** Modulo**;**

float Teilung**;**

//Geben Sie den Namen der nächsten Berechnungen an.

//Initialisierung

void setup**()** **{**

//Programmfunktion, die zu Beginn einmal ausgeführt wird.

Serial**.**begin**(**SerialBaud**);**

//Legt die Geschwindigkeit der seriellen Schnittstelle fest

Serial**.**println**(**"Hello world! \n"**);**

//Anzeige von "Hello World!" auf dem seriellen Monitor.

Serial**.**println**(**a**);**

Serial**.**println**(**b**);**

Serial**.**println**(**c**);**

Serial**.**println**(**d**);**

//Zeigt die den Variablen zuvor zugewiesenen Werte an.

Serial**.**println**(**d**,** DEC**);**

//Zeigt den dezimalen Wert von "x" entsprechend dem ASCII-Code an.

Serial**.**println**();** //Zeigt einen Zeilenumbruch an

Addition **=** a **+** b**;**

Subtraktion **=** a **-** b**;**

Multiplikation **=** a **\*** b**;**

Modulo **=** a **%** b**;**

Teilung **=** a **/** b**;**

//Definieren Sie den Wert jeder Berechnung

Serial**.**print**(**"a = "**);**

Serial**.**println**(**a**);**

Serial**.**print**(**"b = "**);**

Serial**.**println**(**b**);**

**if** **(** a **<** b **)** **{**

Serial**.**println**(**"b ist großer als a"**);**

**}**

**else** **if** **(** a **>** b **)** **{**

Serial**.**println**(**"a ist großer als b"**);**

**}**

**else** **{**

Serial**.**println**(**"a und b gleich sind"**);**

**}**

//Wertet aus, ob die Bedingung wahr ist, und führt den entsprechenden Code aus. (Hier ist a größer als b)

Serial**.**print**(**"Addition a + b = "**);**

Serial**.**println**(**Addition**);**

Serial**.**print**(**"Subtraktion a - b = "**);**

Serial**.**println**(**Subtraktion**);**

Serial**.**print**(**"Multiplikation a \* b = "**);**

Serial**.**println**(**Multiplikation**);**

Serial**.**print**(**"Modulo a % b = "**);**

Serial**.**println**(**Modulo**);**

Serial**.**print**(**"Teilung a / b = "**);**

Serial**.**println**(**Teilung**);**

Serial**.**println**();**

Serial**.**println**(**"Zahlen von 1 bis 5 :"**);**

//Zeigt mit der Funktion "for" Zahlen von 1 bis 5 an.

**for** **(** int i **=** 1**;** i **<=** 5**;** i **=** i **+** 1**)** **{**

Serial**.**println**(**i**);**

**}**

**}**

void loop**()** **{**

**}**

Was auf dem seriellen Monitor erscheint:

Hello world!

52

10

1.41

x

120

a = 52

b = 10

a ist großer als b

Addition a + b = 62

Subtraktion a - b = 42

Multiplikation a \* b = 520

Modulo a % b = 2

Teilung a / b = 5.00

Zahlen von 1 bis 5:

1

2

3

4

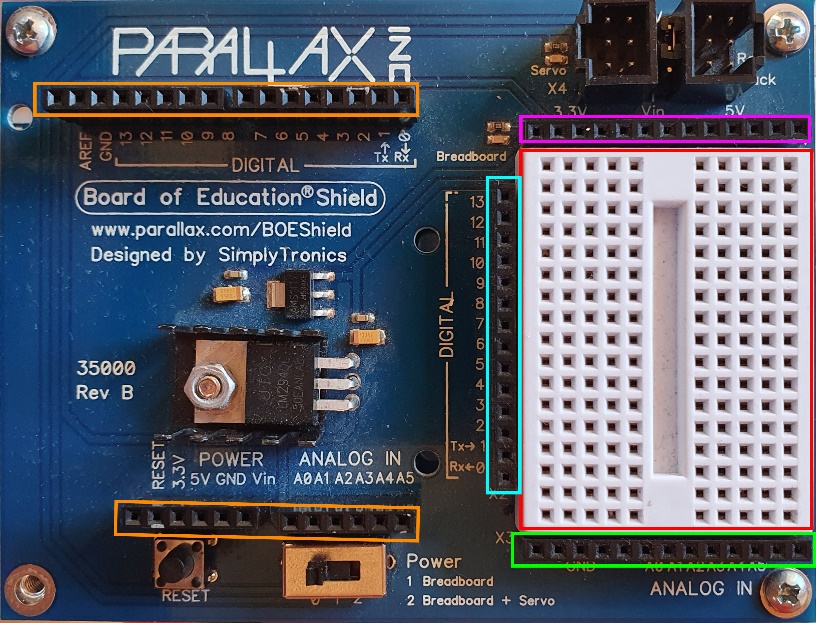
5

## Kapitel 2: Bau und Prüfung von LED-Anzeigen

Dieses zweite Kapitel ist den physikalischen Werkzeugen des Boe-Bots wie LEDs oder Motorservos (abgekürzt "Servos") gewidmet.

Eine LED, Leuchtdiode, ist ein Halbleiter, der Licht ausstrahlt, wenn er mit einem konstanten Strom, in der Regel 20mA, gespeist wird. Diese LEDs werden nützlich sein, um externen Personen zu signalisieren, was im Inneren des Geräts geschieht, z.B. kann eine LED aufleuchten, wenn der Roboter startet, und erlöschen, wenn der Roboter keine Aktion ausführt.

Ein Servo ist ein Motor, den wir mit Hilfe eines Positions- oder Geschwindigkeitsbefehls steuern werden. Die Befehle werden in unserem Code stehen.

Zu Beginn mussten wir das Boe-Bot zusammenbauen. Das Arduino wird unter der Boe-Bot-Platine, die als "Shield" bezeichnet wird, befestigt und leitet die Eingaben des Arduino auf das Shield des Boe-Bots um. Auf diesem Shield befindet sich eine so genannte Lochrasterplatine, mit der wir einfache Schaltungen herstellen können, die kein Löten erfordern. Um das Breadboard herum befinden sich die Anschlüsse, die zum Betrieb einer Schaltung notwendig sind: 5V Stromversorgung, 3,3V Stromversorgung, Batteriespannung und der "Ground"-Anschluss, der die 0V der Schaltung darstellt. Es gibt auch digitale Ein-/Ausgänge, die binär arbeiten (HIGH oder LOW: 5V oder 0V). Es gibt auch analoge Ein-/Ausgänge, die ein Signal von 0V bis 5V in einen digitalen Wert umwandeln.

Lebensmittel

Breadboard

Digitale Ein-/Ausgänge

Analoge Eingänge/Ausgänge

Arduino-Platinen-Pins

In der ersten Aktivität werden wir in unserer Schaltung LEDs verwenden. Um eine LED korrekt zu betreiben, müssen wir einen Widerstand verwenden, der den durch die LED fließenden Strom begrenzt, um sie nicht zu beschädigen.

Mit dem Ohm’schen Gesetz können wir leicht den Wert des Widerstandes berechnen, der erforderlich ist, um eine 2V-LED zu betreiben, ohne 20mA zu überschreiten:

Aus Sicherheitsgründen werden wir einen 220-Widerstand Ω verwenden, um die LED nicht zu beschädigen. Die Schaltung, die wir verwenden werden, besteht aus 2 Widerständen von je 220 Ω und 2 roten LEDs. Jeder Widerstand wird mit einer LED und einem digitalen Ausgang des Arduino verbunden.

Um den Zustand einer Spindel zu ändern, verwenden Sie den Befehl digitalWrite (Pin, Zustand). "Pin" ist die Nummer des Pins, an den die LED angeschlossen ist, und "Status" ist die Pin-Spannung (LOW → 0V, HIGH → 5V).

Das untenstehende Programm erlaubt es einer LED, alle 500 Millisekunden einzeln zu leuchten.

void setup**()** **{**

pinMode**(**12**,** OUTPUT**);** // Ausgangspins 12 und 13

pinMode**(**13**,** OUTPUT**);**

**}**

void loop**()** **{**

digitalWrite**(**12**,** 0**);** // Pin 12: 0V

digitalWrite**(**13**,** 1**);** // Pin 13: 5V

delay**(**500**);** // Warten 0,5s

digitalWrite**(**12**,** 1**);** // Pin 12: 5V

digitalWrite**(**13**,** 0**);** // Pin 13: 0V

delay**(**500**);** // Warten 0,5s

**}**

Die Motorservos werden durch Rechteckimpulse gesteuert. Im Inneren des Motorservos befindet sich ein Potentiometer, mit dem der Motor so eingestellt werden kann, dass bei einem Impuls von 1500µs der Motor gestoppt wird. Dazu schreiben Sie einfach ein Programm, das den Roboter anweist, 1500µs-Impulse an die Servomotoren zu senden, und stellen dann das Potentiometer ein, bis der Motor nicht mehr läuft. Die Impulsreihen, die der Roboter übersetzen kann, reichen von 1300µs bis 1700µs, wobei 1500µs der Parameter ist, bei dem sich der Roboter nicht mehr bewegt. 1300µs und 1700µs sind die Werte, bei denen sich der Motor mit voller Drehzahl dreht, aber die Drehrichtung der Motoren definiert.

Hier ist der Code, der zur Einstellung des Potentiometers verwendet wird:

#include <Servo.h>

Servo servoLeft**;**

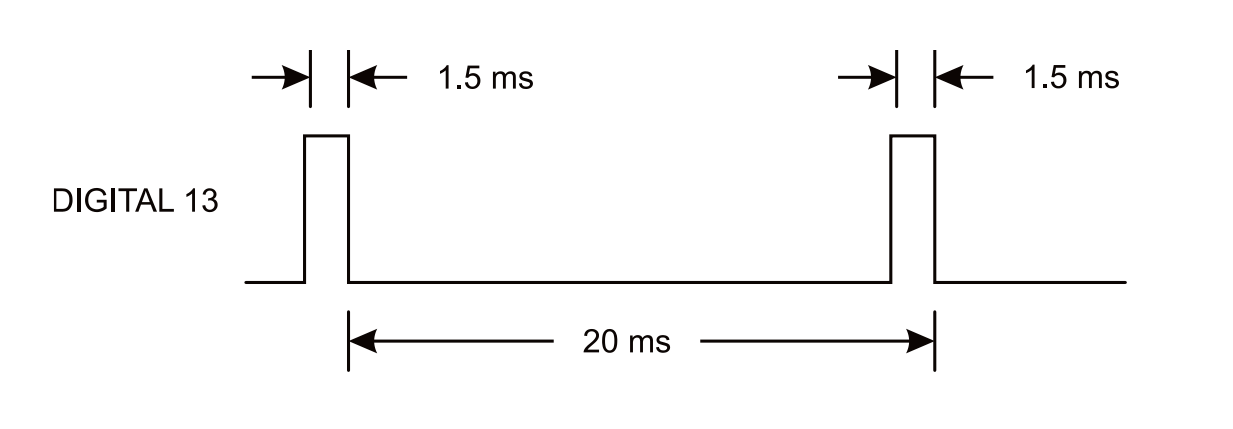
void setup**()** **{**

servoLeft**.**attach**(**13**);**

servoLeft**.**writeMicroseconds**(**1500**);**

**}**

Diese Grafik zeigt das an die Servomotoren gesendete Signal:



Kapitel 3: Zusammenbau und Prüfung von Boe-Bots

Dieses Kapitel ist der Konstruktion des Roboters und einigen grundlegenden Tests gewidmet. Sobald alle Teile zusammengebaut sind, wird der Roboter auf Bestellung rollbereit sein.

Mit dem Code aus dem vorigen Kapitel können wir den Roboter mit der Funktion "<servo>.writeMicroseconds(x)" bewegen. Dank dieser Funktion und der beiden Räder können wir den Roboter vorwärts und rückwärts fahren lassen, den Roboter in die gewünschte Richtung und Geschwindigkeit drehen, wir sind völlig frei von der Art und Weise, wie sich der Roboter vorwärtsbewegt. Und dank der Batterien muss der Code, sobald er in den Arbeitsspeicher des Roboters eingespeist ist, nicht mehr per Kabel an einen Computer angeschlossen werden.

Dann werden wir lernen, wie man eine einfache Schaltung erstellt, die beim Start des Roboters ein Geräusch abgibt. Um einen Ton zu erzeugen, müssen wir einen "piezoelektrischen Wandler" verwenden, d.h. einen Gegenstand aus einem Quarz im Inneren, der bei einer bestimmten Frequenz schwingt, um einen mehr oder weniger hohen Ton zu erzeugen. Dank der Funktion "tone(Pin, Frequenz, Dauer)" des Arduino wird es möglich sein, die Frequenz und die Dauer des Tons mit der Funktion "delay(ms)" zu wählen.

Hier ist der Code, der zur Erzeugung des Tons verwendet wird:

void setup**()** **{**

tone**(**4**,** 3000**,** 1000**);**

delay**(**1000**);**

**}**

void loop**()** **{**

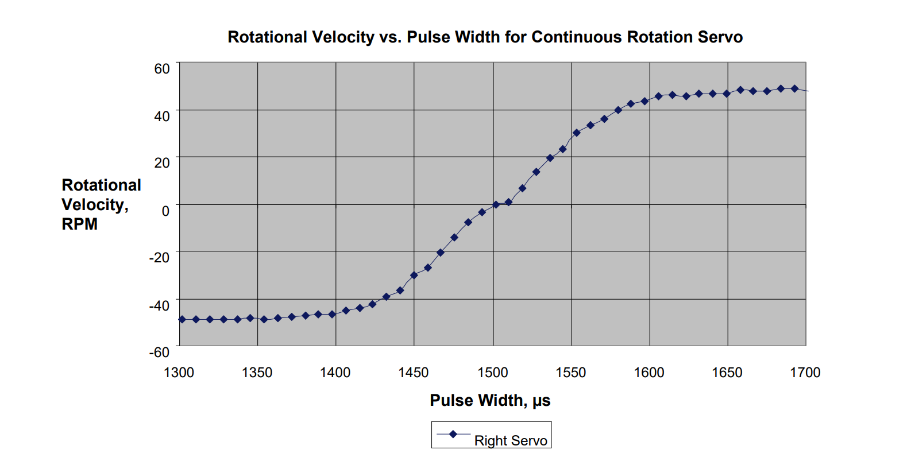
Serial**.**println**(**"Waiting for reset..."**);**

delay**(**1000**);**

**}**

Hier erzeugt die Funktion tone einen 3000Hz[[4]](#footnote-4)-Ton für 1 Sekunde. Der Roboter wartet dann weitere 1 Sekunde, bevor er zum Schleifenteil des Codes übergeht.

Im Schleifenteil wartet der Roboter einfach darauf, dass der Benutzer die physische "Reset"-Taste auf dem Roboterschild drückt. Dieser Knopf befiehlt dem Roboter, den Code von Anfang an zu starten; in diesem Fall bewirkt das Drücken des Reset-Knopfes, dass der Roboter wieder einen Ton abgibt.

Danach werden wir lernen, wie man die Geschwindigkeit der Servos steuert. Die Servos werden durch Pulsweitenmodulation (PWM: pulse with modulation) angetrieben. Dies ist ein Impuls mit fester Frequenz, der Impuls definiert die Drehzahl des Motors. Der Nachteil ist, dass die Geschwindigkeit des Rades nicht linear der Länge des Impulses folgt. Die Geschwindigkeit in Abhängigkeit von der Länge des Impulses ist zwischen 1400µs und 1600µs ziemlich linear, aber sobald diese Werte überschritten werden, tendiert die Kurve zu einer maximalen und minimalen Geschwindigkeit. Es ist wichtig, diesen Versatz zu berücksichtigen, wenn wir eine genaue Geschwindigkeit für unseren Roboter festlegen wollen.

1. Akronym für Integrated Development Environment [↑](#footnote-ref-1)
2. Ein Bit ist ein Wert von 1 oder 0 [↑](#footnote-ref-2)
3. Akronym für Random Access Memory [↑](#footnote-ref-3)
4. Hertz (Hz) ist ein Maß für die Frequenz und entspricht der Anzahl der Wiederholungen eines Signals für eine Sekunde. [↑](#footnote-ref-4)