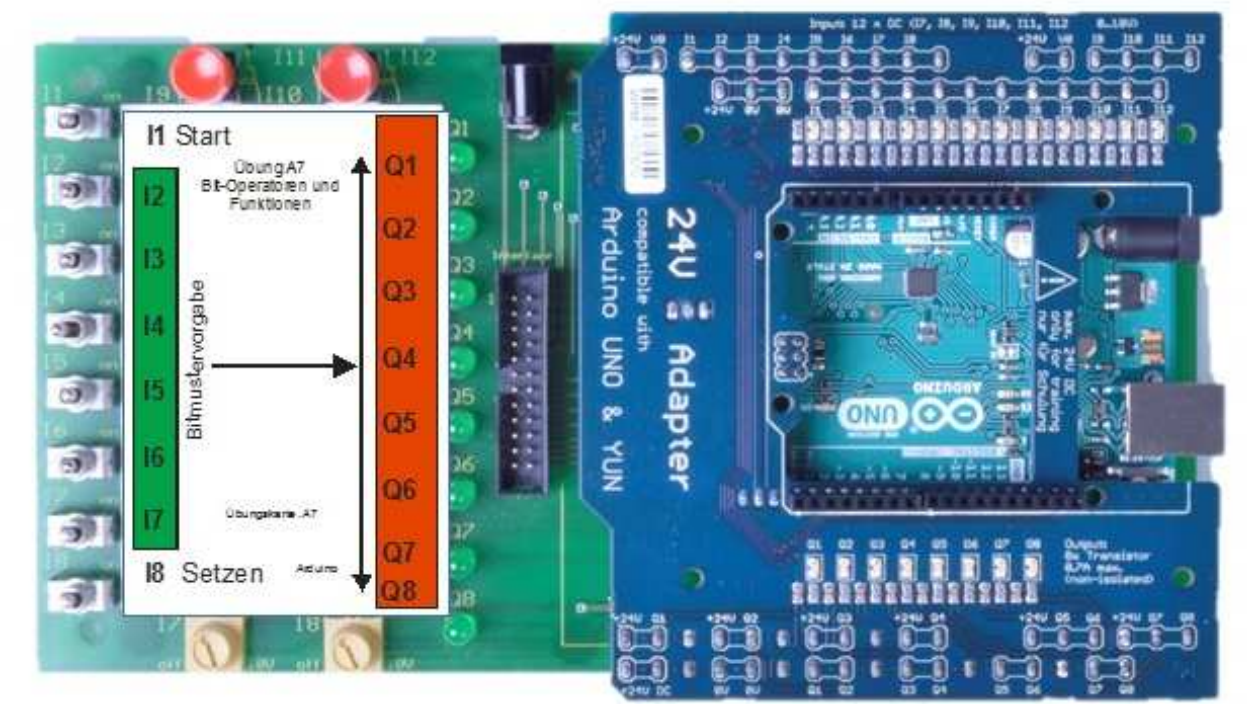
FELTRON ZEISSLER

MINITRAINERSCHULE

-BASIS-

für den ***Arduino*** / ***Genuino***



Modul A

GRUNDLAGEN DER PROGRAMMIERUNG MIT DEM ARDUINO/GENUINO-*MINITRAINER*

Reinhold Grewe / Klaus Machalek

Um die umfangreichen Möglichkeiten zu vermitteln und den unterschiedlichen Vorkenntnissen gerecht zu werden, wurde das Konzept der *MiniTrainerSchule* in mehrere Hefte aufgeteilt.

MiniTrainerSchule-Basis mit die *Module A* und *Modul B*:

Modul A : Arduino-*MiniTrainer* - und Grundlagen der Programmierung

Modul B : Grundschaltungen der Elektrotechnik

MiniTrainerSchule-Aufbau mit dem Modul C (separat zu bestellen!):

Modul C : Programmierung eigener Funktionen (Bausteinen)

Haftungsausschluss

Der Inhalt der *MiniTrainerSchule* ist ausschließlich zu Ausbildungszwecken erstellt. Wir haben den Inhalt und die Übungen sorgfältig getestet. Die Autoren übernehmen keinerlei Gewähr für die Aktualität, Richtigkeit und Vollständigkeit der bereitgestellten Informationen in dieser *MiniTrainerSchule*. Haftungsansprüche gegen die Autoren, welche sich auf Schäden materieller oder ideeller Art beziehen, die durch die Nutzung oder Nichtnutzung der dargebotenen Informationen bzw. durch die Nutzung fehlerhafter und unvollständiger Informationen verursacht wurden, sind grundsätzlich ausgeschlossen, sofern seitens der Autoren kein nachweislich vorsätzliches oder grob fahrlässiges Verschulden vorliegt. Die Autoren behalten es sich ausdrücklich vor, Teile der Seiten ohne gesonderte Ankündigung zu verändern, zu ergänzen oder zu löschen.

Autoren: Reinhold Grewe / Klaus Machalek

1. Auflage 2016 (Stand 07/16) Build 20160720

*MODULE DER ARDUINO/GENUINO-**MINITRAINERSCHULE-**BASIS*

Modul A

Modul A richtet sich an den Einsteiger. Vorkenntnisse sind nicht erforderlich, Kenntnisse der booleschen Algebra sind aber hilfreich.

In diesem Modul wird der Arduino-*MiniTrainer* vorgestellt und in Betrieb genommen. Nach dem Erlernen dieses Moduls sind Sie in der Lage, den Arduino-*MiniTrainer* in Betrieb zu nehmen und kleine Programme zu schreiben. Sie beherrschen die Syntaxregeln, können Variable deklarieren und mit den Datentypen Bit, Byte und Integer umgehen. Sie können Entscheidungen mit "*if ... else*" programmieren und kennen die wichtigsten Operatoren. Um die Einstiegsschwelle niedrig zu halten, wurde bewusst auf eine vollständige Sprachbeschreibung verzichtet. Die für die Beispiele und Übungsaufgaben verwendeten Sprachelemente werden erläutert.

Modul B (Vorkenntnisse : Modul A)

Aufbauend auf Modul A werden die logischen Verknüpfungen von Bit-Variablen und verschiedene Zeit-, Zähl- und Registerfunktionen behandelt. Schwerpunkt ist es, logische Zusammenhänge zu verstehen und in ein Programm umzusetzen. Komplexe Funktionen, wie z.B. Zeitrelais und Zählerrelais, stehen in diesem Modul fertig vorbereitet zur Verfügung und können in den Übungen verwendet werden, ohne die programmtechnische Umsetzung verstehen zu müssen. Das selbständige Gestalten solcher komplexer Funktionen ist Gegenstand von Modul C. Nach dem Erlernen dieses Moduls sind Sie in der Lage, logische Verknüpfungen in einem Programm zu realisieren. Sie verstehen die Funktion von Zeitbausteinen, Zählern und Registern und können diese in Ihre Programme einbinden.

*MODULE DER ARDUINO/GENUINO-**MINITRAINERSCHULE-**AUFBAU*

Modul C (Vorkenntnisse Modul A)

Die Kenntnisse in der Programmiersprache werden um verschiedene Kontrollelemente wie "*for"*, "*while"*, und "*case"* erweitert. Die Gültigkeitsbereiche von Variablen und Funktionen werden behandelt. Nach dem Erlernen dieses Moduls sind Sie in der Lage, eigene komplexe Funktionen zu schreiben und in das Programm einzubinden.

INHALTSVERZEICHNIS MODUL A

1 HARDWARE DES MINITRAINERS FÜR DEN ARDUINO/GENUINO

Der *MiniTrainer* für den Arduino ist ein modulares Bausteinsystem, das auf dem LOGO!-*MiniTrainerXL* bzw. dem EASY700-*MiniTrainer* aufsetzt. Das heißt, ist einer der *MiniTrainer* vorhanden, so kann dieser direkt als Übungsplattform genutzt werden. In der folgenden Liste sind alle Hardwarekomponenten eines kompletten Arduino-*MiniTrainers* aufgeführt. Diese lassen sich je nach Bedarf einzeln bestellen (siehe Bezugsquellen).

1.	LOGO!-*MiniTrainerXL* oder EASY700-*MiniTrainer*	Best.-Nr. 5035-3002 Best.-Nr. 5035-3001
2.	Arduino-Adapter OHNE Arduino/Genuino	Best.-Nr. 5035-3025
3.	Arduino UNO oder Arduino Leonardo oder Arduino YUN	Best.-Nr. A000073 Best.-Nr. A000057 Best.-Nr. A000008
4.	Arduino-*MiniTrainerSchule-Basis – Modul A*	Best.-Nr. 5035-3026
5.	Arduino-*MiniTrainerSchule-Basis – Modul B*	Best.-Nr. 5035-3031
6.	Arduino-MiniTrainerSchule-Aufbau – *Modul C*	Best.-Nr. 5035-3029

Hinweis
Es sind auch weitere Arduino-Boards nutzbar, wenn sie den gleichen mechanischen Aufbau wie der *Arduino UNO* aufweisen.

Bezugsquellen:

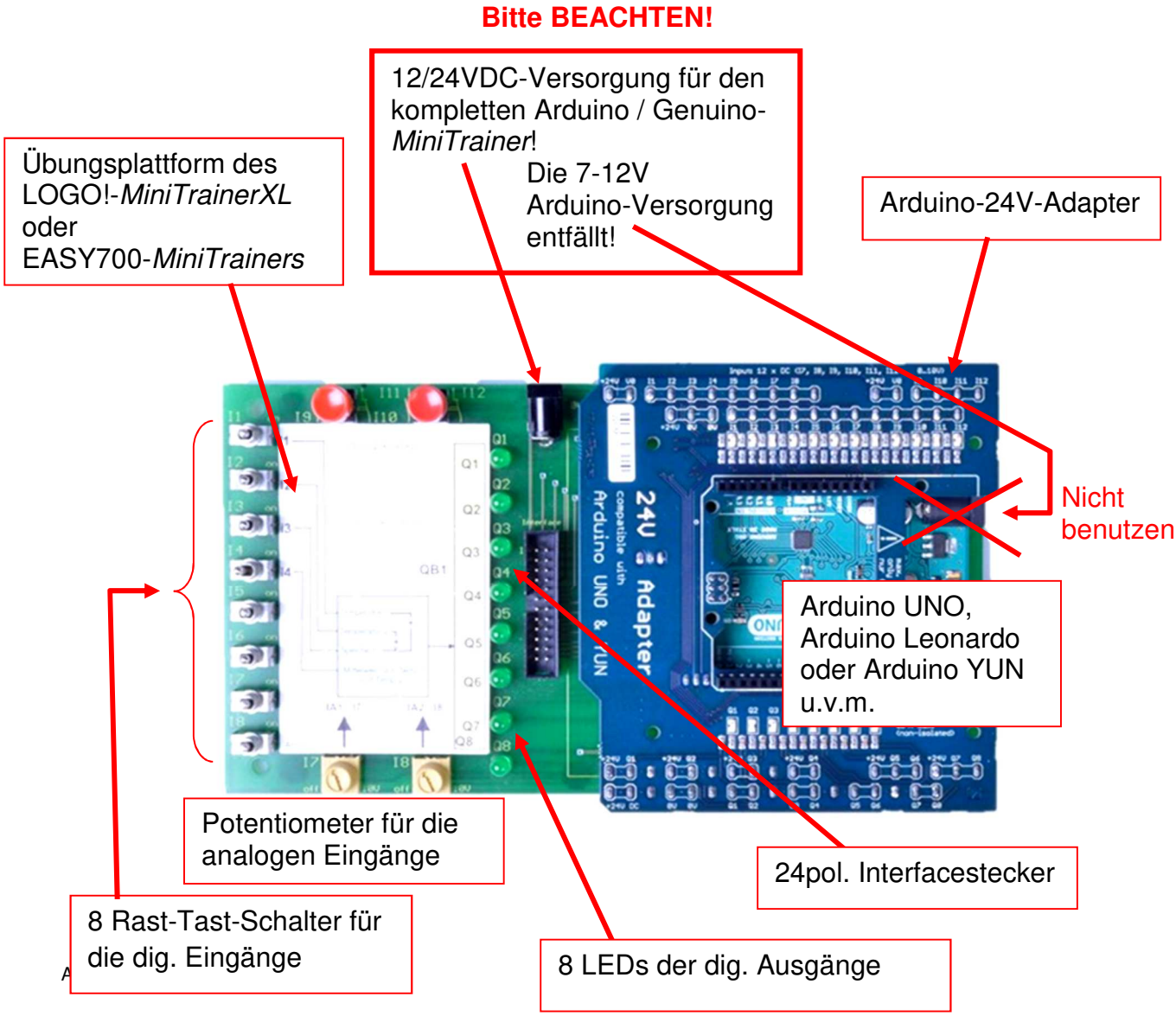
1) FELTRON Elektronik-ZEISSLER & Co. GmbH,
53842 Troisdorf, Auf dem Schellerod 22, Tel: 02241-4867-0,
email feltron@feltron.de,
Internet (online Shop) : www.feltron.de

2) Online Shop: www.minitrainer.de

1.1 Aufbau des MiniTrainers für den Arduino/Genuino

Die Hardware des *MiniTrainers* (LOGO! oder EASY700) verfügt über

12 digitale Eingänge (I1- I8 als Rast-Tast-Schalter und I9 - I12 als Wechsler-Taster)
2 analoge Eingänge (alternativ für I7 und I8)
8 digitale Ausgänge (Q1 - Q8) oder alternativ
1 byte-Ausgang (QB)



Zusätzlich zu den 8 Rast-Tastschaltern, bietet der MiniTrainer 2 Wechsler-Kontakte:

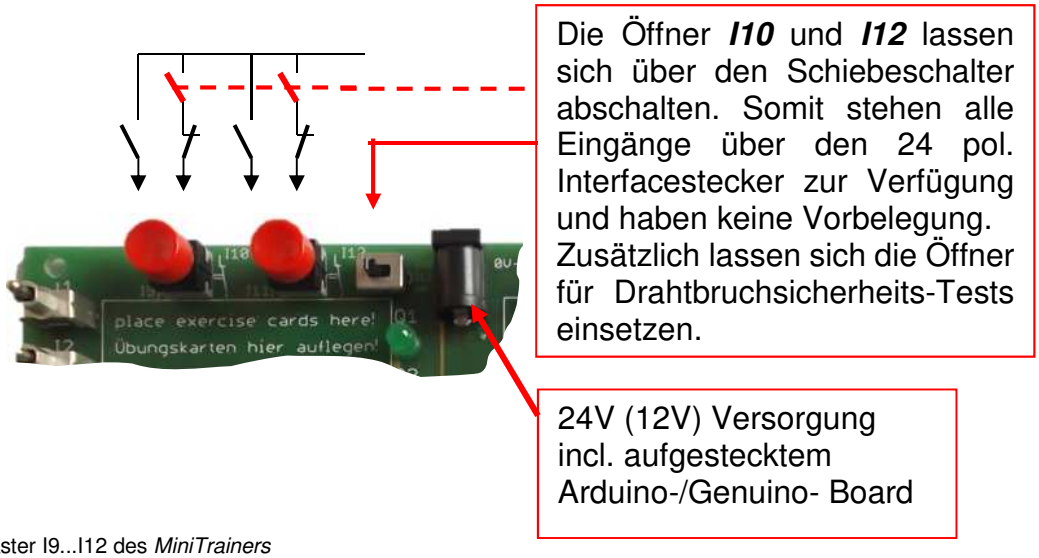


Abb. 2 Taster I9...I12 des *MiniTrainers*

Zum Simulieren von analogen Eingängen werden die Potentiometer I7 und I8 verwenden. Sie liefern eine analoge Spannung von 0-10 VDC.

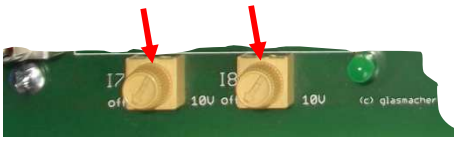


Abb. 3 Potentiometer des *MiniTrainers*

Hinweis
Alle Ein- und Ausgänge sind auf einen 24poligen Interfacestecker geführt. Somit lassen sich Übungsmodule (z.B. Ampel) und 24V-Modelle (z.B. Fischertechnik) direkt an-schließen.

Im Lieferumfang des ***MiniTrainers*** befindet sich:

1 x LOGO!-*MiniTrainerXL* oder EASY700-*MiniTrainer*
1 x 24VDC-Steckernetzgerät (auf Sonderwunsch auch 12VDC)
1 x LOGO! oder EASY-*MiniTrainerSchule* incl. Übungskarten und 1 x Lösungs-CD

Im Lieferumfang des **24V-Adapters** befindet sich:

1 x 24V-Adapter entweder für LOGO!- oder EASY700-*MiniTrainer*
1 x Satz Stiftleisten (LOGO! oder EASY)
1 x Arduino-*MiniTrainerSchule-Basis* (Modul A und B)
incl. Übungskarten (Lösungen online über www.Feltron.de abrufbar).
1 x Montageanleitung

Hinweis
Weiteres *MiniTrainer*-Zubehör unter www.minitrainer.de

1.2 Der 24V-Adapter für den Arduino/Genuino

Der 24V-Adapter bildet zusammen mit dem Arduino-/Genuino-Board und dem LOGO!-*MiniTrainerXL* oder dem EASY700-*MiniTrainer* die Übungsplattform Arduino-*MiniTrainer*.
Auf der Unterseite (Bauteileseite) des 24V-Adapters wird zunächst das Arduino/ Genuino-Board gesteckt. Danach werden entsprechend des *MiniTrainers* die Stiftleisten, wie im der Abb. 4 gezeigt, gesteckt.

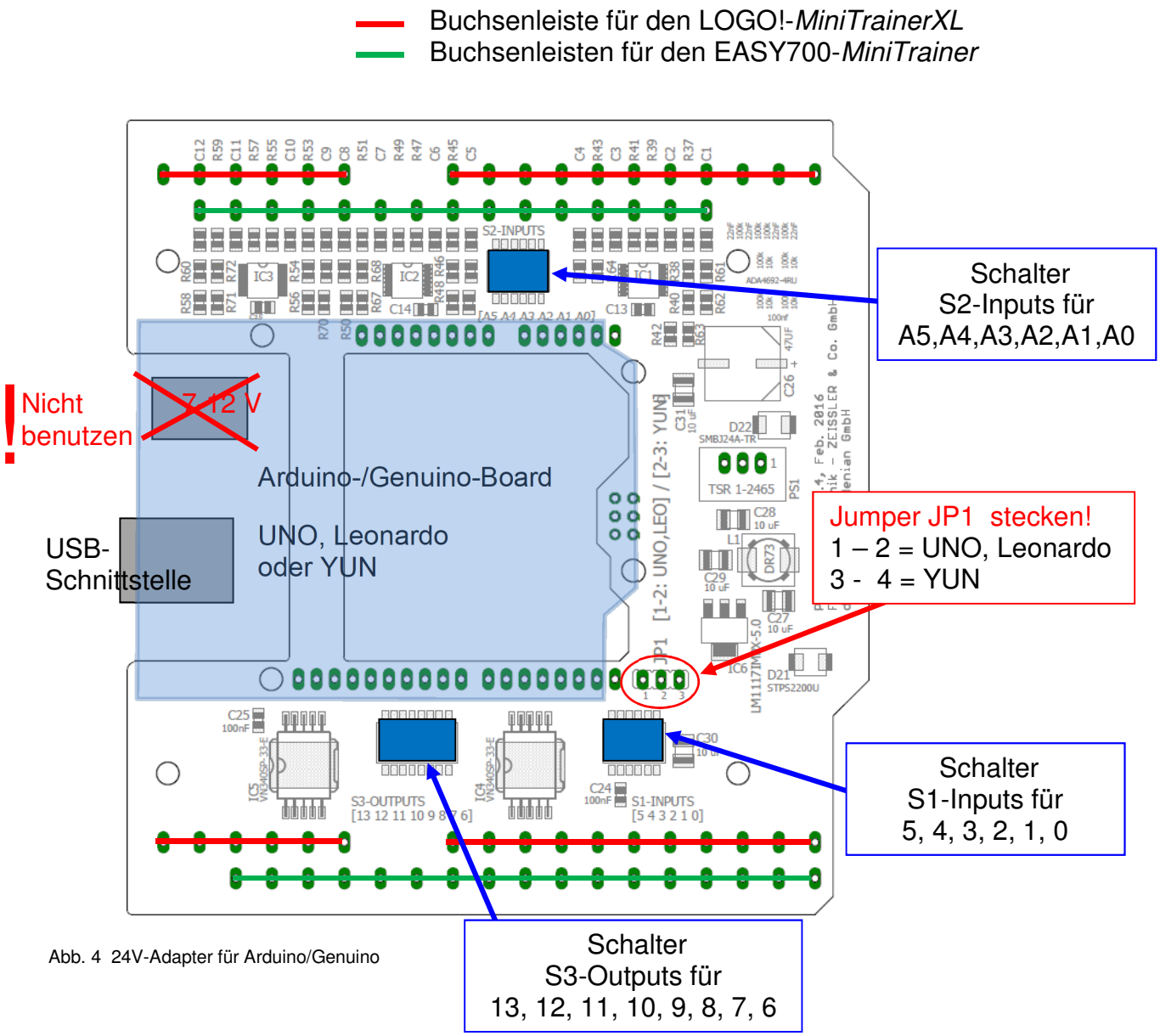


Abb. 4 24V-Adapter für Arduino/Genuino

Hinweis
Die Schalter S1, S2 und S3 dienen zum Abschalten der Inputs und Outputs zur 24 V-E/A-Ebene des *MiniTrainers*. Somit lassen sich bedarfsweise auch Shields, z.B. ein Display, auf den Arduino stecken und die notwendigen Datenleitungen zum *MiniTrainer* unterbrechen**. Im Lieferzustand sind alle Schalter "ON"**.

1.3 ÜBUNGSKARTEN

Zu den Übungsaufgaben der einzelnen Module liegen die entsprechenden Übungskarten dem Buch bei. Die Übungskarten sind im Visitenkartenformat erstellt und lassen sich auf den *MiniTrainer* legen.

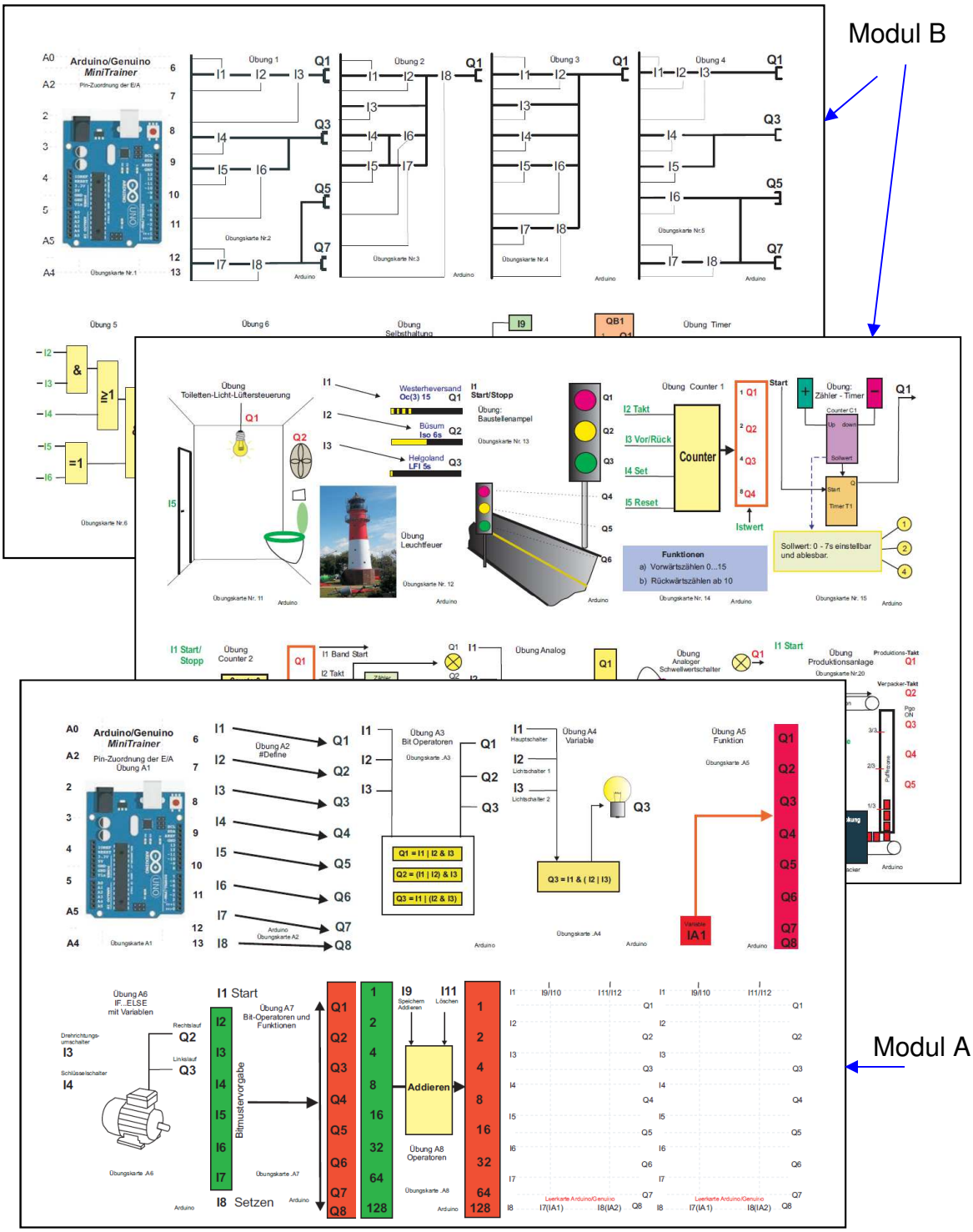


Abb. 5 Übungskarten Modul A und B

2 DIE ENTWICKLUNGSUMGEBUNG

Die Software für die Erstellung der Programme ist die Original **Arduino-Software.** Die **Arduino-Software** ist kostenlos unter **https://www.arduino.cc** aus dem Internet herunterladbar und besteht im Wesentlichen aus:

- dem Editor, mit dem die Programme geschrieben werden,
- dem Übersetzer, der die Schreibweise überprüft und den Mikroprozessor-Code erstellt,
- einem Ladeprogramm, das den fertigen Code in den Mikroprozessor lädt.

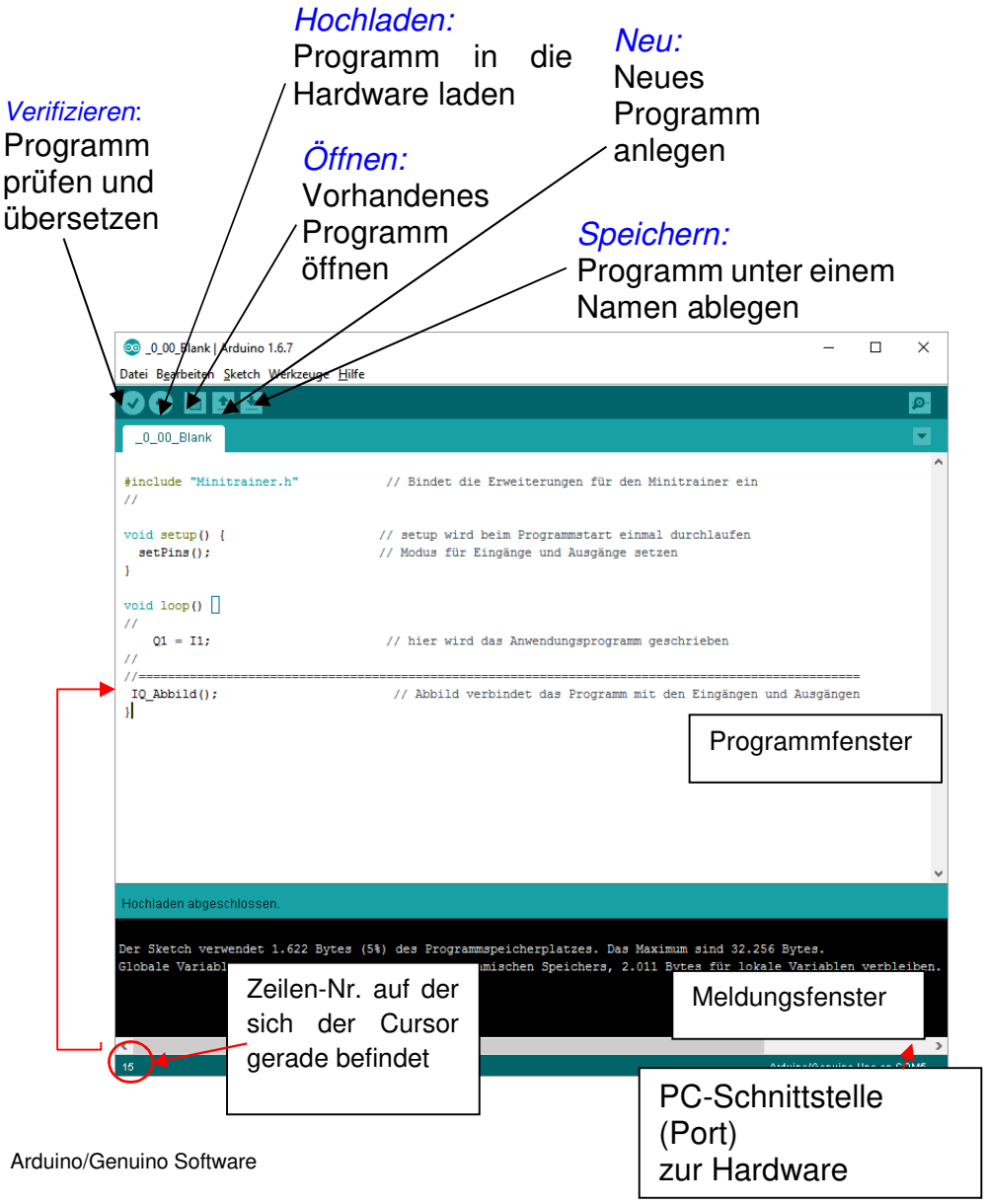


Abb. 6 Arduino/Genuino Software

Im oberen Teil des Programmfensters wird das Programm geschrieben. Im unteren Teil des Fensters werden Hinweise und Fehlermeldungen ausgegeben. Um das Programm in den Arduino zu laden muss die Schnittelle (Port) angewählt werden.

3 STRUKTUR DER PROGRAMME.

Die Grundstruktur ist bei allen Programmen - Sketch genannt - gleich.

3.1 SETUP()

Die Funktion "setup()" wird einmal beim Programmstart durchlaufen und dient der Voreinstellung und Initialisierung.

3.2 LOOP()

Darauf folgt die Funktion "loop()". Diese Funktion und die darin enthaltenen Anweisungen werden zyklisch wiederholt. Die Anwendungsprogramme werden in den Funktionen "setup()" und "loop()" geschrieben. Sie sind jeweils mit einem Paar aus geschweiften Klammern umschlossen { ... }. Innerhalb dieser Klammern wird der Programmcode geschrieben.

3.3 DER ZEITLICHER ABLAUF DES PROGRAMMS

Der zeitliche Ablauf eines Programms ist in der Abbildung dargestellt. Nach dem Start wird zunächst "setup()" aufgerufen. Darin befindet sich die Einrichtung der An-schluss-Pins als Ausgang oder Eingang. Dann folgt zeitlich die Funktion "loop()", die ab jetzt immer wiederholt wird.

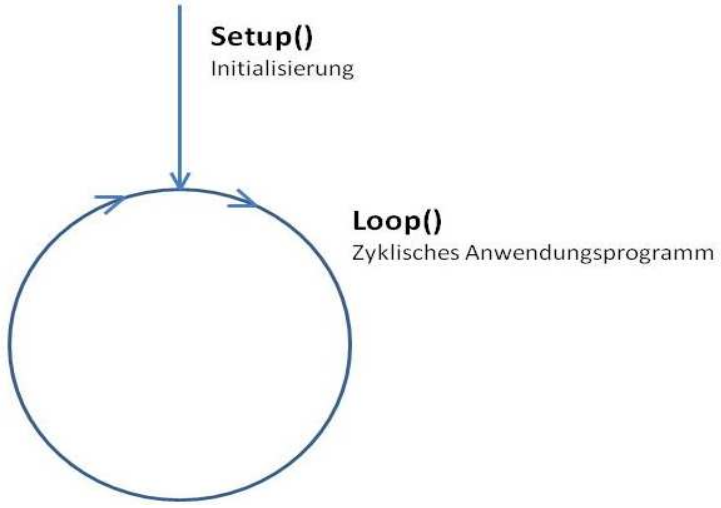


Abb. 7 Programmzyklus

4 ERSTES PROGRAMM

4.1 PROGRAMMIERSCHRITTE

- Voreinstellungen innerhalb der Klammern {} von "setup()" schreiben.
- Programm innerhalb der Klammern {} von "loop()" schreiben.
- Programm unter einem frei wählbaren Namen speichern.
- Programm in die Hardware laden und testen.

4.2 INBETRIEBNAHME

Wir nehmen die Hardware und Software in Betrieb und schreiben ein erstes Programm. Dazu müssen wir wissen, wie die Eingangsschalter bzw. die Ausgangs-LEDs des *MiniTrainers* mit den Prozessor-Pins verbunden sind.

Arduino/ Genuino	**24 V Adapter Inputs (*MiniTrainer*)**											
	I1	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**	**I11**	**I12**
Uno	2	3	4	5	A0	A1	A2	A3	A4	-	A5	-
Leonardo	2	3	4	5	A0	A1	A2	A3	A4	0	A5	1
YUN	2	3	4	5	A0	A1	A2	A3	A4	0	A5	1

Tabelle 1 Zuordnung der Eingänge

Arduino/ Genuino	**24 V Adapter Outputs (*MiniTrainer*)**							
	Q 1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**
Uno	6	7	8	9	10	11	12	13
Leonardo	6	7	8	9	10	11	12	13
YUN	6	7	8	9	10	11	12	13

Tabelle 2 Zuordnung der Ausgänge

Mit drei Funktionen können wir auf die Pins zugreifen.

Funktion	**Beispiel**	**Beschreibung**
pinMode	pinMode(2, INPUT); pinMode (6, OUTPUT);	Pin 0 ist ein Eingang Pin 6 ist ein Ausgang
digitalRead	digitalRead(A0);	Liest der Wert an A0
digitalWrite	digitalWrite(6, HIGH); digitalWrite(6, digitalRead(A0));	Setzt Ausgangspin 6 auf HIGH Statt HIGH kann hier auch ein Ausdruck stehen

Tabelle 3

4.3 PINMODE

Die Pins der Arduino-Baugruppe können als Eingänge oder Ausgänge betrieben werden. Dieses teilen wir mit der Funktion "pinMode(<Pin>, <INPUT oder OUTPUT>" mit.

4.4 DIGITALREAD

Die Funktion "digitalRead(<Pin>") liest den Wert des Pins, der in der Klammer (...) angegeben wird.

4.5 DIGITALWRITE

Die Funktion "digitalWrite(<Pin>, <Wert>) " beschreibt den in der Klammer angegeben Pin mit dem Wert. Der Wert kann einfach HIGH oder LOW sein, es ist aber auch eine Funktion möglich.

4.6 ERSTES PROGRAMM

Wir schreiben ein Programm, bei dem der Ausgang Q1 auf HIGH gesetzt wird.

PROGRAMMBEISPIEL A20

```
void setup()
{
  pinMode (6, OUTPUT); //Hier steht die Initialisierung von PIN 6
}

void loop()
{
  digitalWrite(6, HIGH); //Die LED Q1 an PIN 6 leuchtet.
}
```

Nun erweitern Sie das Programm so, dass der Wert des Eingangs I1 auf den Ausgang Q1 geschrieben wird.

PROGRAMMBEISPIEL A21

```
void setup()
{
  pinMode (2,  INPUT);
  pinMode (6, OUTPUT);
}

void loop()
{
  digitalWrite(6, digitalRead(2));
}
```

Die LED Q1 leuchtet, wenn I1 eingeschaltet wird.

4.7 Programm übersetzen und Fehlermeldungen auswerten

Mit dem Klick auf  wird das Programm geprüft und in einen Code übersetzt, den der Prozessor ausführen kann. Werden dabei Fehler gefunden, werden diese im Meldungsfenster angezeigt.

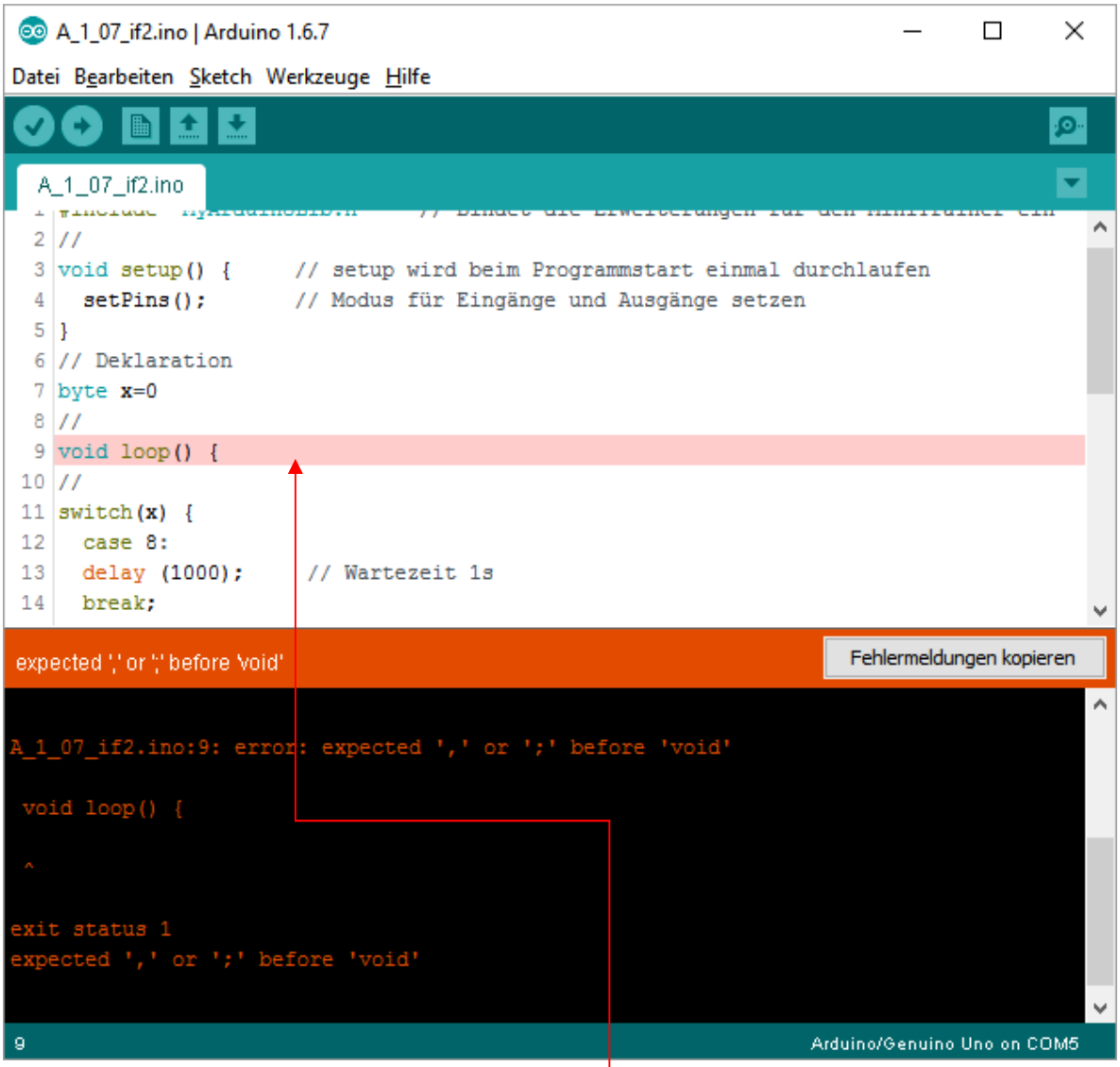


Abb. 8 Fehlermeldungen

Im Meldungsfenster steht die festgestellte Ursache und ein Hinweis auf die betroffene Zeile. Zusätzlich ist die Zeile im Programm farblich hinterlegt.

Häufige Fehler sind:

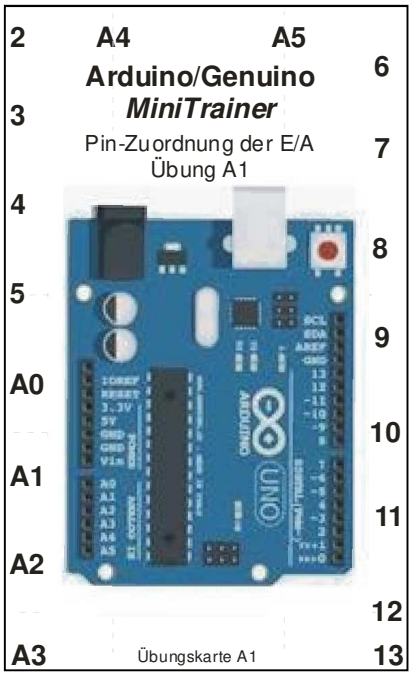
- fehlendes ";" am Ende einer Anweisung
- fehlende schließende Klammer ")" oder "}".
- ungleiche Anzahl von offenen und geschlossenen Klammer.

Wenn das Programm fehlerfrei übersetzt wurde, kann es in geladen und getestet werden.

5.1.1 ÜBUNG A1 - PROGRAMM SCHREIBEN

Erweitern Sie das *erste Beispielprogramm A20* so, dass alle LEDs über je einen Eingang gemäß der Übungskarte A1 angesteuert werden. Kommentieren Sie das Programm und machen Sie sich mit dem Speichern, Laden sowie den Fehlermeldungen vertraut.

Hinweis
2 -> 6
3 -> 7
4 -> 8 usw.



Übungskarte A1

5.2 DEFINITIONEN

In den ersten Programmen haben wir eine Tabelle benötigt, um die PINs des Prozessors den Schaltern und LEDs des *MiniTrainers* zuzuordnen. Diese Aufgabe soll nun ein Programm erledigen.

Dazu verwenden wir die *"*#define" - Anweisung. Sie weist den Übersetzer an, den Wert durch einen Namen zu ersetzen.

```
#define  <Name> <Wert>
```

In unserem Beispiel :

```
#define I1 2
```

Die "#define"-Anweisung wird oberhalb von "setup()" in das Programm geschrieben. Dann können wir in folgenden Programmteilen direkt mit I1 und Q1 arbeiten.