Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

In Kooperation mit dem VDE XperienceLab

Übungsblatt 8: Programmieren in C (WS 2019/20)

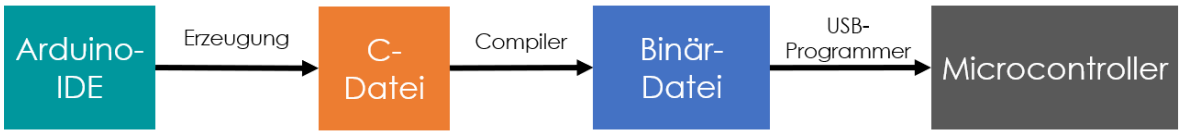
Einführung

Wie kann man einen Mikrocontroller programmieren?

Codes für unseren Mikrocontroller werden in C geschrieben. Ein Compiler kompiliert die C-Datei dann automatisch in eine Binärdatei, die auf den Chip gespielt werden kann.

Um den Mikrocontroller diverse Funktionen ausführen zu lassen, sind spezielle Anweisungen im C-Code nötig (für das Ausgeben von 5V an einem Ausgangs-Pin muss beispielsweise ein Register auf dem Chip angesprochen und manipuliert werden). Den C-Code komplett selbst zu erstellen erfordert einige Erfahrung.

Arduino stellt jedoch in einer Entwicklungssoftware (IDE) viele Funktionen zur Verfügung, die das Erstellen des C-Codes deutlich vereinfachen. Die Programmiersprache in Arduino bleibt dabei C.



Alle Schritte des Kompilierens und Hochladens führt die Software bzw. das Board automatisch aus.

Wie benutzt man die Arduino Entwicklungsumgebung?

1. Arduino-IDE herunterladen (www.arduino.cc > Software > Downloads)
2. Arduino-IDE installieren und öffnen

   Der Code besteht grundsätzlich aus 2 Teilen:

   1. Setup:
      Dieser Teil des Codes wird genau 1x bei jedem Start des µC ausgeführt u.a. werden hier alle In-/Output-Pins initialisiert (festgelegt)

   2. Loop:
      Hier steht der Hauptcode
      Der Loop ist eine Endlosschleife
      -> der Inhalt (Code) wird von Anfang bis Ende durchgeführt und dann wieder von vorne gestartet

3. Code schreiben

4. Board über USB anschließen und Einstellungen treffen

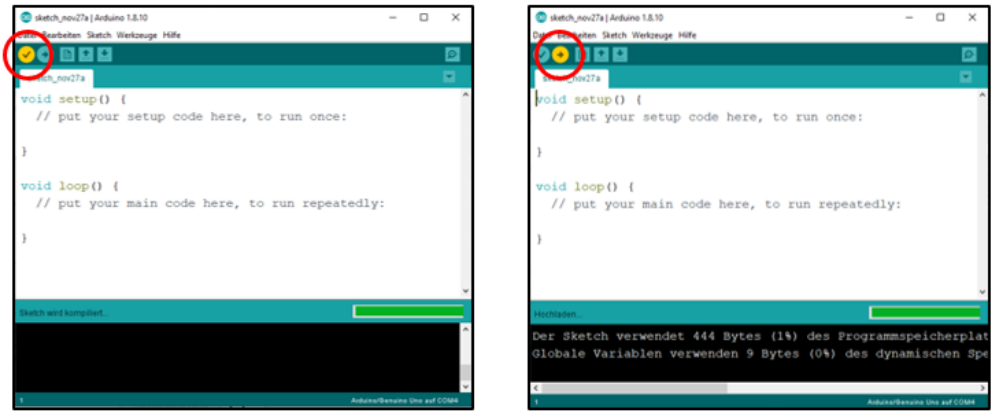
   1. Board auswählen:
      Werkzeuge > Board > Arduino/Genuino Uno

   2. USB-Gerät auswählen:
      Werkzeuge > Port > COM…

   3. Programmer auswählen:
      Werkzeuge > Programmer > AVRISP mkII

5. Code kompilieren und hochladen

Code kompilieren > Code hochladen



Wie sieht ein einfacher Code aus?

Beispiel:
An Pin Nummer 2 hängt eine LED, die alle 0,5 Sekunden an- bzw. ausgeschaltet werden soll. Dafür muss der Ausgang des Mikrocontrollers zwischen 0V und 5V wechseln.

Im Setup muss der Pin als Output deklariert werden:

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

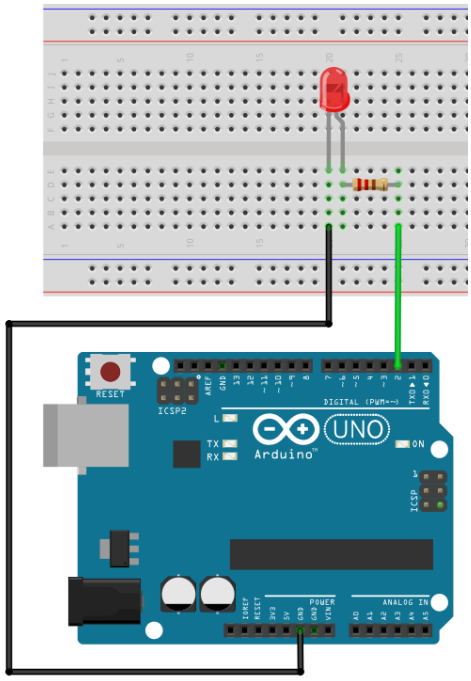
pinMode() ist eine der vielen Funktionen, die die Arduino-IDE bereitstellt. Der Pin und die Funktion des Pins werden der Funktion übergeben. Hinter der Funktion verbergen sich die ganzen Anweisungen, die das gewünschte Verhalten des Mikrocontroller hervorrufen. Darum müssen wir uns also nicht kümmern. Also geht es mit dem Hauptcode weiter…

```
void loop() {
  digitalWrite(2, HIGH);
  delay(500);
  digitalWrite(2, LOW);
  delay(500);
}
```

Hier benötigen wir 2 weitere Funktionen:
digitalWrite() lässt uns den Ausgang steuern (LOW = 0V; HIGH = 5V)
delay() pausiert alles um die angegebene Zeit (in ms)

Aufgabe 1 – LED blinken lassen

Nehmt das Steckbrett mit der einzelnen LED zur Hand und schließt es wie auf dem Bild zu sehen an den Arduino an:

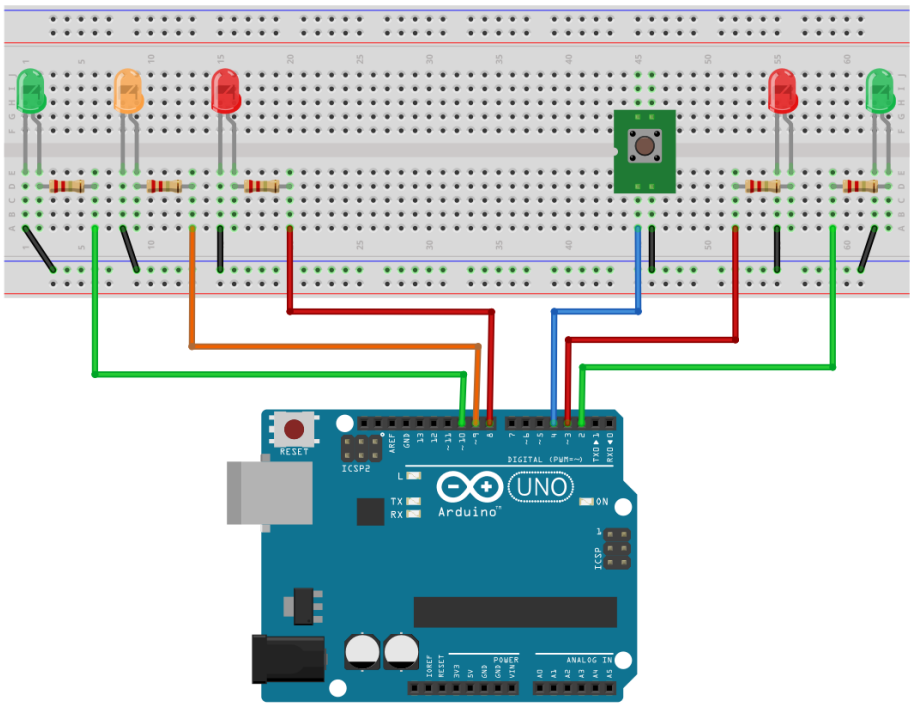


Übernehmt den in der Einführung gezeigten Code und spielt ihn auf das Board.

Nun sollte die LED blinken.

Aufgabe 2 – Ampelsteuerung

Schließt alles wie im Bild gezeigt an:



LED	Pin
Auto Rot	8
Auto Gelb	9
Auto Grün	10
Fußgänger Rot	3
Fußgänger Grün	2
Taster	4

a) Die Ampeln sollen gleichzeitig folgende Phasen haben:

	2 Sek.	0,5 Sek.	2 Sek.	0,5 Sek.
Auto	Rot	Rot Gelb	Grün	Gelb
Fußgänger	Grün	Rot	Rot	Rot

b) Die Auto-Ampel soll dauerhaft grün sein. Wenn der Taster gedrückt wird, soll die Auto-Ampel wie in Teil a.) auf rot springen und die Fußgänger-Ampel grün werden.

Hinweise:
Der Pin des Tasters muss für den Arduino als Input deklariert werden. Wird der Taster gedrückt, so wird der Pin mit GND verbunden (0V).

Ist der Taster nicht gedrückt, so hängt der Pin „in der Luft“ (floating genannt). Um das zu verhindern muss ein sogenannter Pull-Up Widerstand vom Pin zu VCC eingesetzt werden. Dieser „zieht“ den Input auf 5V. Wird der Taster gedrückt, so wird der Input auf 0V gezwungen. Lässt man ihn wieder los, wird er wieder vom Widerstand auf 5V gezogen. Jeder Pin hat einen internen Pull-Up Widerstand verbaut. So muss er nicht auf dem Steckbrett eingebaut werden.

Wichtige Funktionen:

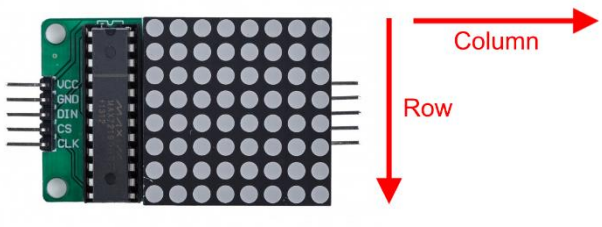
pinMode(*Pin Nummer*, INPUT_PULLUP)
Aktiviert den internen Pull-Up Widerstand

digitalRead(*Pin Nummer*)
Funktion, die 0 oder 1 zurückgibt, je nachdem ob LOW (~0V) oder HIGH (~5V) am Pin anliegen.

Aufgabe 3 – LED-Matrix

Schließt die LED-Matrix folgendermaßen an:

Matrix	Pin
VCC	5V
GND	GND
DIN	12
CS	11
CLK	10



Zudem müsst ihr die Library „*LedControl*“ installieren. Dazu müsst ihr den Ordner nach *Dokumente* > *Arduino* > *libraries* kopieren. Nicht vergessen die Arduino-IDE neu zu starten!

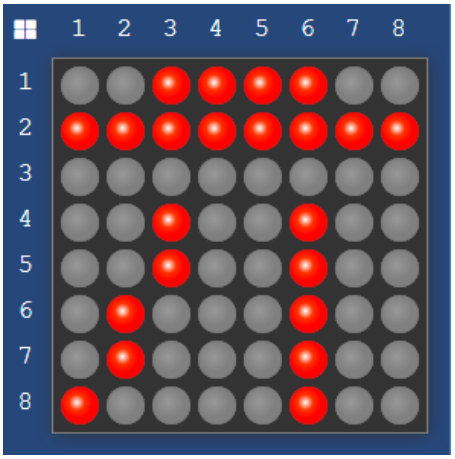
a) Ladet den Beispielcode auf das Board und lasst ihn durchlaufen.
Schaut euch den Code genauer an. Jeder Reihe der Matrix wird ein Byte zugeordnet. Jedes Bit im Byte entspricht einer LED dieser Reihe. 1 bedeutet „an“, 0 „aus“.
Folglich brauchen wir 8 Bytes, um jede LED unserer 8x8-Matrix anzusteuern.
Für jeden Buchstabe wird also ein Array mit (hier 5) Bytes angelegt.

Die Funktion *lc.setRow(0, Reihennummer, Byte)* beschreibt jede LED-Reihe mit dem jeweiligen Byte.

lc.clearDisplay(0) ist ein Befehl, um alle LEDs der Matrix auszuschalten (Reset).

Lasst ein eigenes Wort durchlaufen. Die Daten für alle Buchstaben sind am Anfang des Codes gegeben.

b) Stellt folgendes Logo auf der Matrix dar. Beschreibt jede Reihe dabei mit dem entsprechenden Byte zum an- bzw. ausschalten der einzelnen LEDs der Reihe.



c) Öffnet den 2. Code zur Aufgabe. Legt darin ein Array mit 64 Einträgen an. Diese sollen entweder 1 oder 0 sein. Zu Beginn des Codes setzt ihr alle Stellen 0.
Nun gibt es eine Funktion *display_matrix()*, der ihr eure Matrix übergeben müsst. Daraufhin werden alle LEDs der echten Matrix eurem Array entsprechend beleuchtet. Die Nummerierung ist dabei wie folgt:

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Programmiert ein Lauflicht, dass bei LED 0 beginnt und dann im Zick-Zack über die Matrix durch die einzelnen Reihen läuft.