Programmieren für Ingenieure
Sommer 2015

Andreas Zeller, Universität des Saarlandes

Anmeldung HISPOS

Prüfungsanmeldung ab sofort möglich
Verpflichtend
Anmeldung bis zum 27. Mai 2015

Felder initialisieren

leds

0	1	2	3	4	5	6
13	12	11	10	9	8	7

```
int leds[7];

void setup() {
   leds[0] = 13;
   leds[1] = 12;
   leds[2] = 11;
   leds[3] = 10;
   leds[4] = 9;
   leds[5] = 8;
   leds[6] = 7;
}
```

```
int leds[] =
    { 13, 12, 11, 10, 9, 8, 7 };

void setup() {
   // Bereits initialisiert
}
```

While-Schleifen

```
i = 1;
while (i < 5) {
   Serial.println(i);
   i = i + 1;
}
Serial.println("ENDE");
```

Ausgeführte Anweisungen

```
i = 1;
Serial.println(i);
i = i + 1; // 2
Serial.println(i);
i = i + 1; // 3
Serial.println(i);
i = i + 1; // 4
Serial.println(i);
i = i + 1; // 5
Serial.println("ENDE");
```

Themen heute

- Zeichenketten
- Interaktion
- Automaten

http://commons.wikimedia.org/wiki/File:Media-markt-automat.jpg

Ein Kauf

Artikel auswählen

- Zeige eine Auswahl (*Menü*) vorhandener Artikel an
- Der Benutzer soll mit Tasten zwischen Artikeln navigieren
- Dabei soll jeweils der Preis angezeigt werden

Ein Kauf

Münzen einwerfen

- Münzen (0,10€–2,00€) erkennen
- Nach Einwurf vom Betrag abziehen

- Zu zahlenden Restbetrag anzeigen
- Wiederholen, bis Restbetrag = 0,00€

Ein Kauf

Artikel ausgeben

- Bei uns etwa:
  LED leuchten lassen

Artikel ausgeben

Interaktion

Fotos: Wikipedia, Pixabay

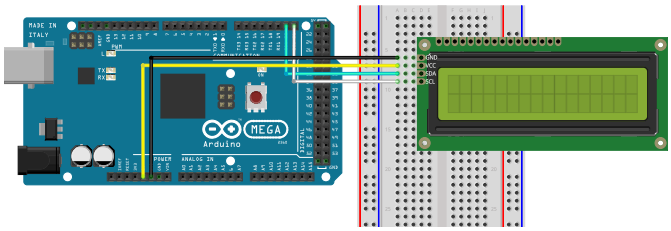
LCD-Anzeige

- Für *Interaktion* mit Käufer:
  - Preis anzeigen
  - Restbetrag anzeigen

Plan

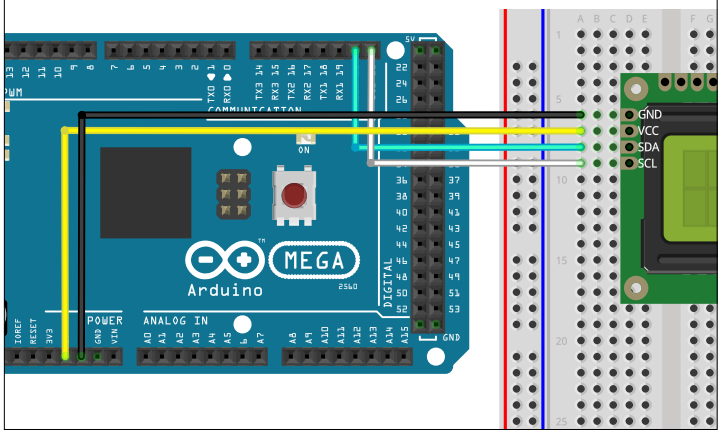
- Wir schließen ein *LCD-Display* an
- Wir schließen *Taster* an…
  - …für die *Auswahl* des Artikels
  - …als Sensoren für den *Münzeinwurf*
- Wir führen den Käufer durch den Kauf

LCD anschließen



http://arduino.cc/en/uploads/Tutorial/LCD_bb.png

LCD anschließen



http://arduino.cc/en/uploads/Tutorial/LCD_bb.png

LCD-Bibliothek

- Eine *Bibliothek* sammelt Funktionen zu einem bestimmten Zweck
- Die *LiquidCrystal*-Bibliothek erlaubt es, ein angeschlossenes LCD anzusprechen
- Um die Bibliothek zu nutzen, muss sie zunächst *eingebunden* werden.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

LCD einrichten

- Dieser Code richtet ein *lcd-Objekt* ein, dessen Funktionen wir dann nutzen können

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Dabei ist 0x27 die I2C-Adresse des LCD-Moduls (dessen Beschreibung zu entnehmen)
- Die anderen beiden Parameter geben die Zeichenzahl des LCDs an (16x2)

Ausgabe auf LCD

- lcd.print() gibt einen Text oder einen Wert auf dem LCD aus

```
void setup() {
    lcd.init();
    lcd.backlight();
    lcd.print("Hello, world!");
}
```

Demo

- Texte und Werte eigener Wahl ausgeben

Schreibmarke

- Die *Schreibmarke* (*Cursor*) bestimmt, wo der nächste Text ausgegeben wird
- Zu Beginn links oben; wird mit jeder Ausgabe weiterbewegt
- Analog: Schreibmarke in Textverarbeitung

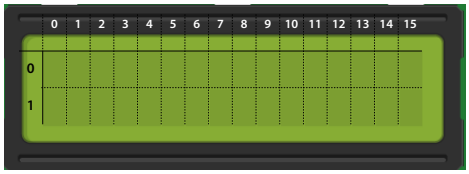
Schreibmarke bewegen

- Die Funktion lcd.setCursor(*x, y*) bewegt die Schreibmarke auf Spalte *x*, Zeile *y*
- Position oben links ist (0, 0)

Schreibmarke bewegen

```
void setup() {
    lcd.init();
    lcd.backlight();
    lcd.clear();  // Bildschirm löschen

    lcd.print("Hello, world!");
    lcd.setCursor(7, 0);
    lcd.print("class");
}
```



Schreibmarke bewegen

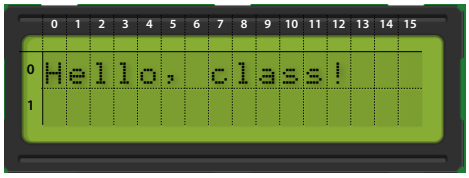
```
void setup() {
    lcd.init();
    lcd.backlight();
    lcd.clear();  // Bildschirm löschen

    lcd.print("Hello, world!");
    lcd.setCursor(7, 0);
    lcd.print("class");
}
```

Schreibmarke bewegen

```
void setup() {
    lcd.init();
    lcd.backlight();
    lcd.clear();  // Bildschirm löschen

    lcd.print("Hello, world!");
    lcd.setCursor(7, 0);
    lcd.print("class");
}
```



Zeit ausgeben

```
void setup() {
    lcd.init();
    lcd.backlight();
    lcd.print("Hello, world!");
}

void loop() {
    lcd.setCursor(0, 1);
    lcd.print(millis() / 1000);
}
```

Demo

- Schreibmarke bewegen; mit cursor() anzeigen lassen

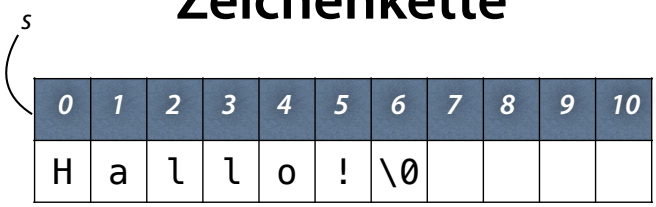
Zeichenketten in C

- Ein einzelnes Zeichen wird in C in einzelne Hochkomma eingeschlossen:

```
char c = 'a';
lcd.print(c);
```

- Die wichtigste Verwendung ist als *Feld* von Zeichen (*Zeichenkette,* auch *String*)
- Zeichenketten enden mit einem speziellen "Null-Zeichen", geschrieben als '\0'

Zeichenkette




```
char s[] = { 'H', 'a', 'l', 'l', 'o', '!', '\0'};
```

oder kürzer

```
char s[] = "Hallo!";
```

Demo

- Texte und Werte eigener Wahl ausgeben

String-Funktionen

- Um mit Zeichenketten umzugehen, bietet C zahlreiche Funktionen:
  - strcpy() – Zeichenkette kopieren
  - strcat() – Zeichenketten verknüpfen
  - strlen() – Länge bestimmen
  - strcmp() – Zeichenketten vergleichen

Zeichenketten kopieren

- strcpy(*ziel*, *quelle*) kopiert *quelle* nach *ziel*
- *ziel* muss groß genug sein für *quelle*

```
char name[20];
strcpy(name, "Test");
```

name

0	1	2	3	4	5	6	7	8	9	10
T	e	s	t	\0						

Zeichenketten verknüpfen

- strcat(*ziel*, *quelle*) hängt *quelle* an *ziel* an
- *ziel* muss groß genug sein

```
char name[20];
strcpy(name, "Test");
strcat(name, "en");
```

name

0	1	2	3	4	5	6	7	8	9	10
T	e	s	t	\0						

Zeichenketten verknüpfen

- strcat(*ziel*, *quelle*) hängt *quelle* an *ziel* an
- *ziel* muss groß genug sein

```
char name[20];
strcpy(name, "Test");
strcat(name, "en");
```

name

0	1	2	3	4	5	6	7	8	9	10
T	e	s	t	e	n	\0				

Länge bestimmen

- strlen(*s*) gibt die Länge von *s* zurück

```
char name[20];
strcpy(name, "Test");
n = strlen(name);  // n = 4
```

name

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
T	e	s	t	\0						

Zeichenketten vergleichen

- Zeichenketten können *nicht* mit ==, != usw. verglichen werden
- strcmp(*s*, *t*) vergleicht *s* und *t*
- Rückgabe:
  0 – bei gleichem Inhalt
  <0 – wenn *s* alphabetisch *vor t* kommt
  >0 – wenn *s* alphabetisch *nach t* kommt

```
int u = strcmp("Anton", "Anton");
int v = strcmp("Anton", "Berta");
```

Zeichenketten als Parameter

- Eine Zeichenkette ***name*** als *Parameter* wird so deklariert:

  char *name*[] (oft auch: char **name*)

```
void print_ten_times(char text[]) {
    lcd.print(text);
    // noch neun Mal ausgeben
}
```

strcpy()

- So könnte eine Implementierung von strcpy() (Zeichenkette kopieren) aussehen:

```
void strcpy(char ziel[], char quelle[]) {
    int i = 0;
    while (quelle[i] != '\0') {
        ziel[i] = quelle[i];
        i++;
    }
    ziel[i] = '\0';
}
```

strcpy()

- Alternative Implementierung für Tippfaule

```
void strcpy(char ziel[], char quelle[]) {
    int i = 0;
    int j = 0;
    while (ziel[i++] = quelle[j++]) {}
}
```

Echte C-Experten können das noch kürzer

Warum nicht ziel[i++] = quelle[i]? Weil nicht exakt definiert ist, wann i erhöht wird.

strcat()

- So könnte eine Implementierung von strcat() (Zeichenketten verknüpfen) aussehen:

```
void strcat(char ziel[], char quelle[]) {
    int i = strlen(ziel);
    int j = 0;
    while (quelle[j] != '\0') {
        ziel[i] = quelle[j];
        i++; j++;
    }
    ziel[i] = '\0';
}
```

strcat(ziel, “en”)

```
void strcat(char ziel[], char quelle[]) {
    int i = strlen(ziel);
    int j = 0;
    while (quelle[j] != '\0') {
        ziel[i] = quelle[j];
        i++; j++;
    }
    ziel[i] = '\0';
}
```

ziel

0	1	2	3	4	5	6	7	8	9	10
T	e	s	t	\0						

strcat(ziel, “en”)

```
void strcat(char ziel[], char quelle[]) {
    int i = strlen(ziel);
    int j = 0;
    while (quelle[j] != '\0') {
        ziel[i] = quelle[j];
        i++; j++;
    }
    ziel[i] = '\0';
}
```

ziel

0	1	2	3	4	5	6	7	8	9	10
T	e	s	t	e	n	\0				

strlen()

- So könnte eine Implementierung von strlen() (Länge bestimmen) aussehen:

Typ des Rückgabewertes

```
int strlen(char s[]) {
    int i = 0;
    while (s[i] != '\0') {
        i++;
    }
    return i;
}
```

Rückgabe

```
int n = strlen(ziel);
```

Artikel auswählen

Artikel auswählen

- Zeige eine Auswahl (*Menü*) vorhandener Artikel an
- Der Benutzer soll mit Tasten zwischen Artikeln navigieren
- Dabei soll jeweils der Preis angezeigt werden

Plan

- In oberer Zeile *Artikel* anzeigen

Artikel anzeigen

```
int DRINKS = 3;
char *drink_name[] = { "Wasser", "Limo", "Bier" };

void print_drinks() {
  int pos = 0;

  for (int i = 0; i < DRINKS; i++) {
    lcd.setCursor(pos, 0);
    lcd.print(drink_name[i]);
    pos += strlen(drink_name[i]) + 1;
  }
}
```

Getränkemenü

Demo

- Schreibmarke bewegen; mit cursor() anzeigen lassen

Artikel auswählen

Artikel auswählen

- Zeige eine Auswahl (*Menü*) vorhandener Artikel an
- Der Benutzer soll mit Tasten zwischen Artikeln navigieren
- Dabei soll jeweils der Preis angezeigt werden

Plan

- In unterer Zeile *Preise* anzeigen
- Jeweils unter dem Artikelnamen
- Problem: Preise als *Zeichenketten* darstellen

Zeichen zu Zahlen

- Die Funktion atoi(*s*) wandelt den Beginn einer Zeichenkette *s* in eine ganze Zahl um
- Führende Leerzeichen werden überlesen
- *s* bleibt unverändert
- Keine Fehlererkennung

```
n = atoi("25");
n = atoi(" 25");
n = atoi(" 25 years");
n = atoi("25years");
```

Zahlen zu Zeichen

- Die Funktion sprintf(*s, format, werte*…) füllt *s* mit *werte,* wie in *format* angegeben:

%d – **Dezimalzahl** (decimal)

```
char buf[128];
sprintf(buf, "%d", 25);      // buf[] == "25"
```

%s – **Zeichenkette** (string)

```
sprintf(buf, "%s", "Hugo"); // buf[] == "Hugo"
```

Zahlen zu Zeichen

- Das sprintf()-Format kann *weiteren Text* enthalten, der dann mitkopiert wird:

```
char buf[128];
int n = 25;
sprintf(buf, "Kaufen Sie %d Pelze", n);
// buf[] == "Kaufen Sie 25 Pelze"
```

Zahlen zu Zeichen

- Die Ausgabe von mehreren Parametern ist ebenfalls möglich:

```
char buf[128];
int n = 25;
int p = 600;
sprintf(buf, "%d Bohrer zu %d Euro", n, p);
// buf[] == "25 Bohrer zu 600 Euro"
```

Maximale Länge

- Zahlen vor der Formatangabe bestimmen die *auszugebende Länge*

```
char buf[128];
int n = 25;

sprintf(buf, "Anzahl:%5d", n);
// buf[] == "Anzahl:   25"      5 Zeichen

sprintf(buf, "Anzahl:%7d", n);
// buf[] == "Anzahl:     25"    7 Zeichen
```

Vorangestellte Nullen

- Beginnt die Längenangabe mit 0, wird mit Nullen anstatt Leerzeichen aufgefüllt

```
char buf[128];
int n = 25;

sprintf(buf, "Anzahl:%05d", n);
// buf[] == "Anzahl:00025"

sprintf(buf, "Anzahl: %03d", n);
// buf[] == "Anzahl: 025"
```

Menü mit Preis

```
int DRINKS = 3;
char *drink_name[] = { "Wasser", "Limo", "Bier" };
int drink_price[] = { 100, 150, 250 };

void print_prices() {
    int x = 0;
    for (int i = 0; i < DRINKS; i++) {
        char buffer[100];

        lcd.setCursor(x, 1);
        sprintf(buffer, "%d.%02d",
                drink_price[i] / 100,
                drink_price[i] % 100);
        lcd.print(buffer);
        x += strlen(drink_name[i]) + 1;
    }
}
```

Getränkemenü

Demo

- Schreibmarke bewegen; mit cursor() anzeigen lassen

Artikel auswählen

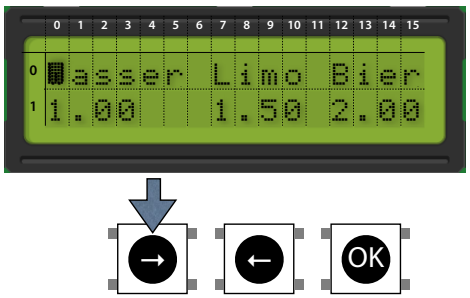
Artikel auswählen

- Zeige eine Auswahl (*Menü*) vorhandener Artikel an
- Der Benutzer soll mit Tasten zwischen Artikeln navigieren
- Dabei soll jeweils der Preis angezeigt werden

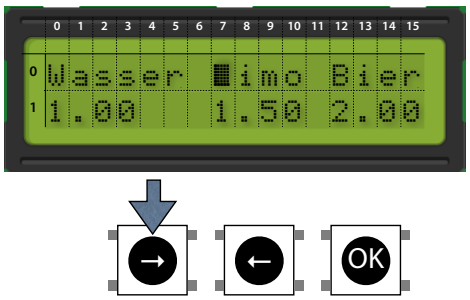
Plan

- Aktuell ausgewählten Artikel durch Blinken sichtbar machen
- Auswahl mit *Tastern* (links, rechts, OK)

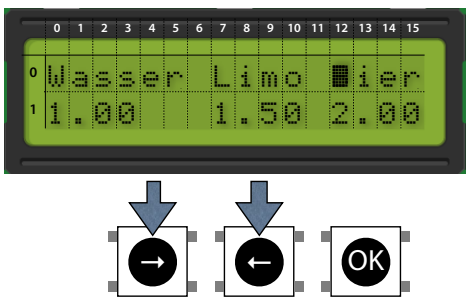
Getränkemenü



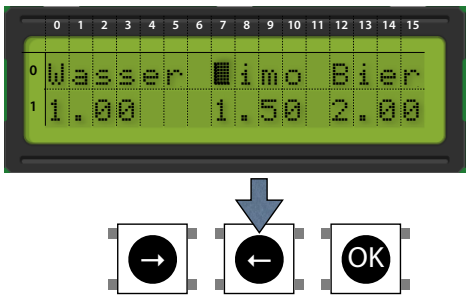
Getränkemenü



Getränkemenü



Getränkemenü



Schreibmarke sichtbar machen

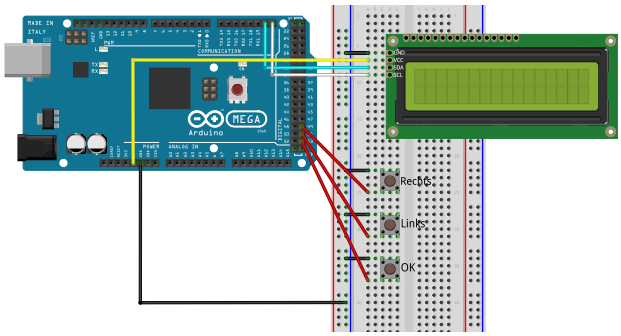
- Mit lcd.cursor() können wir die Schreibmarke als Unterstrich (_) sichtbar machen
- lcd.blink() lässt sie als Block blinken
- lcd.noCursor(), lcd.noBlink() schaltet dies wieder ab

Auswahl anzeigen

```
// Schreibmarke auf Getränkename positionieren
// drink = 0: 1. Getränk,
// drink = 1: 2. Getränk, usw.
void show_selection(int drink) {
  int x = 0;

  for (int i = 0; i < drink; i++)
  {
    x += strlen(drink_name[i]);
    x += strlen(" ");
  }
  lcd.setCursor(x, 0);
  lcd.blink();
}
```

Taster anschließen



http://arduino.cc/en/uploads/Tutorial/LCD_bb.png

Getränk wählen

```
// Getränk auswählen und dessen Nummer zurückgeben
int choose_drink() {
  int current_selection = 0;
  unsigned long last_select = millis();

  show_selection(current_selection);
  while (1) {
    if (millis() - last_select > 20) {

      // Rechts
      if (digitalRead(PIN_RIGHT) == LOW) {
        if (current_selection < DRINKS - 1) {
          current_selection++;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_RIGHT) == LOW) {}
        last_select = millis();
```

```
      // Links
      if (digitalRead(PIN_LEFT) == LOW) {
        if (current_selection > 0) {
          current_selection--;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_LEFT) == LOW) {}
        last_select = millis();
      }

      // Auswahl
      if (digitalRead(PIN_OK) == LOW) {
        lcd.noBlink();
        while (digitalRead(PIN_OK) == LOW) {}
        return current_selection;
      }
    }
  }
}
```

```
// Getränk auswählen und dessen Nummer zurückgeben
int choose_drink() {
  int current_selection = 0;
  unsigned long last_select = millis();

  show_selection(current_selection);
  while (1) {
    if (millis() - last_select > 20) {

      // Rechts
      if (digitalRead(PIN_RIGHT) == LOW) {
        if (current_selection < DRINKS - 1) {
          current_selection++;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_RIGHT) == LOW) {}
        last_select = millis();
      }

      // Links
      if (digitalRead(PIN_LEFT) == LOW) {
        if (current_selection > 0) {
          current_selection--;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_LEFT) == LOW) {}
        last_select = millis();
      }

      // Auswahl
      if (digitalRead(PIN_OK) == LOW) {
        lcd.noBlink();
        while (digitalRead(PIN_OK) == LOW) {}
        return current_selection;
      }
    }
  }
}
```

Auswahl bestätigen

```
void print_selection(int selection) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sie kauften:");
  lcd.setCursor(0, 1);
  lcd.print(drink_name[selection]);
  // 3 Sekunden blinken
  long m = millis();
  while (millis() - m < 3000) {
    if (millis() / 200 % 3)
      lcd.backlight();
    else
      lcd.noBacklight();
  }
  lcd.backlight();
}
```

Alles zusammen

```
void loop() {
  lcd.clear();
  print_drinks();
  print_prices();
  int selection = choose_drink();

  print_selection(selection);

  // Weitere Funktionen, etwa:
  // pay_for_drink(selection);
  // dispense_drink(selection);
}
```

Demo

- Schreibmarke bewegen; mit cursor() anzeigen lassen

Ein Kauf

LCD einrichten

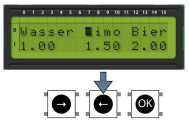
- Dieser Code richtet ein *lcd-Objekt* ein, dessen Funktionen wir dann nutzen können

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Dabei ist 0x27 die I2C-Adresse des LCD-Moduls (dessen Beschreibung zu entnehmen)
- Die anderen beiden Parameter geben die Zeichenzahl des LCDs an (16x2)

Getränkemenü



Zeichenketten in C

- Ein einzelnes Zeichen wird in C in einzelne Hochkomma eingeschlossen:

```
char c = 'a';
lcd.print(c);
```

- Die wichtigste Verwendung ist als *Feld* von Zeichen (*Zeichenkette*, auch *String*)
- Zeichenketten enden mit einem speziellen "Null-Zeichen", geschrieben als '\0'

Menü mit Preis

```
int DRINKS = 3;
char *drink_name[] = { "Wasser", "Limo", "Bier" };
int drink_price[] = { 100, 150, 250 };

void print_prices() {
    int x = 0;
    for (int i = 0; i < DRINKS; i++) {
        char buffer[100];

        lcd.setCursor(x, 1);
        sprintf(buffer, "%d.%02d",
                drink_price[i] / 100,
                drink_price[i] % 100);
        lcd.print(buffer);
        x += strlen(drink_name[i]) + 1;
    }
}
```

Handouts

String-Funktionen

- Um mit Zeichenketten umzugehen, bietet C zahlreiche Funktionen:
  - strcpy() – Zeichenkette kopieren
  - strcat() – Zeichenketten verknüpfen
  - strlen() – Länge bestimmen
  - strcmp() – Zeichenketten vergleichen

Zeichen in C

- Ein einzelnes Zeichen wird in C in einzelne Hochkomma eingeschlossen:

```
char c = 'a';
Serial.println(c);
```

- Die wichtigste Verwendung ist als *Feld* von Zeichen (*Zeichenkette,* auch *String*)
- Zeichenketten enden mit einem speziellen "Null-Zeichen", geschrieben als '\0'

Zeichenkette

s

0	1	2	3	4	5	6	7	8	9	10
H	a	l	l	o	!	\0				

```
char s[] = { 'H', 'a', 'l', 'l', 'o', '!', '\0'};
```

oder kürzer

```
char s[] = "Hallo!";
```

Was ist s[0]?

strlen()

- So könnte eine Implementierung von strlen() (Länge bestimmen) aussehen:

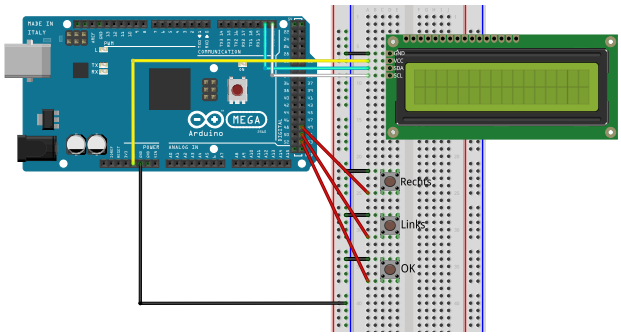
Typ des Rückgabewertes

```
int strlen(char s[]) {
    int i = 0;
    while (s[i] != '\0') {
        i++;
    }
    return i;
}
```

Rückgabe

```
int n = strlen(ziel);
```

Display und Taster anschließen



http://arduino.cc/en/uploads/Tutorial/LCD_bb.png

LCD-Bibliothek

- Eine *Bibliothek* sammelt Funktionen zu einem bestimmten Zweck
- Die *LiquidCrystal*-Bibliothek erlaubt es, ein angeschlossenes LCD anzusprechen
- Um die Bibliothek zu nutzen, muss sie zunächst *eingebunden* werden.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

LCD einrichten

- Dieser Code richtet ein *lcd-Objekt* ein, dessen Funktionen wir dann nutzen können

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Dabei ist 0x27 die I2C-Adresse des LCD-Moduls (dessen Beschreibung zu entnehmen)
- Die anderen beiden Parameter geben die Zeichenzahl des LCDs an (16x2)

Zeichen zu Zahlen

- Die Funktion atoi(*s*) wandelt den Beginn einer Zeichenkette *s* in eine ganze Zahl um
- Führende Leerzeichen werden überlesen
- *s* bleibt unverändert
- Keine Fehlererkennung

```
n = atoi("25");
n = atoi(" 25");
n = atoi(" 25 years");
n = atoi("25years");
```

Zahlen zu Zeichen

- Die Funktion sprintf(*s, format, werte*…) füllt *s* mit *werte,* wie in *format* angegeben:

  %d – **Dezimalzahl** (decimal)

```
char buf[128];
sprintf(buf, "%d", 25);      // buf[] == "25"
```

  %s – **Zeichenkette** (string)

```
sprintf(buf, "%s", "Hugo"); // buf[] == "Hugo"
```

Artikel anzeigen

```
int DRINKS = 3;
char *drink_name[] = { "Wasser", "Limo", "Bier" };

void print_drinks() {
  int pos = 0;

  for (int i = 0; i < DRINKS; i++) {
    lcd.setCursor(pos, 0);
    lcd.print(drink_name[i]);
    pos += strlen(drink_name[i]) + 1;
  }
}
```

Menü mit Preis

```
int DRINKS = 3;
char *drink_name[] = { "Wasser", "Limo", "Bier" };
int drink_price[] = { 100, 150, 250 };

void print_prices() {
    int x = 0;
    for (int i = 0; i < DRINKS; i++) {
        char buffer[100];

        lcd.setCursor(x, 1);
        sprintf(buffer, "%d.%02d",
                drink_price[i] / 100,
                drink_price[i] % 100);
        lcd.print(buffer);
        x += strlen(drink_name[i]) + 1;
    }
}
```

Auswahl anzeigen

```
// Schreibmarke auf Getränkename positionieren
// drink = 0: 1. Getränk,
// drink = 1: 2. Getränk, usw.
void show_selection(int drink) {
  int x = 0;

  for (int i = 0; i < drink; i++)
  {
    x += strlen(drink_name[i]);
    x += strlen(" ");
  }
  lcd.setCursor(x, 0);
  lcd.blink();
}
```

Getränk wählen

```
// Getränk auswählen und dessen Nummer zurückgeben
int choose_drink() {
  int current_selection = 0;
  unsigned long last_select = millis();

  show_selection(current_selection);
  while (1) {
    if (millis() - last_select > 20) {

      // Rechts
      if (digitalRead(PIN_RIGHT) == LOW) {
        if (current_selection < DRINKS - 1) {
          current_selection++;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_RIGHT) == LOW) {}
        last_select = millis();
      }
```

```
      // Links
      if (digitalRead(PIN_LEFT) == LOW) {
        if (current_selection > 0) {
          current_selection--;
        }
        show_selection(current_selection);
        while (digitalRead(PIN_LEFT) == LOW) {}
        last_select = millis();
      }

      // OK
      if (digitalRead(PIN_OK) == LOW) {
        lcd.noBlink();
        while (digitalRead(PIN_OK) == LOW) {}
        return current_selection;
      }
    }
  }
}
```

Auswahl bestätigen

```
void print_selection(int selection) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sie kauften:");
  lcd.setCursor(0, 1);
  lcd.print(drink_name[selection]);
  // 3 Sekunden blinken
  long m = millis();
  while (millis() - m < 3000) {
    if (millis() / 200 % 3)
      lcd.backlight();
    else
      lcd.noBacklight();
  }
  lcd.backlight();
}
```

Alles zusammen

```
void loop() {
  lcd.clear();
  print_drinks();
  print_prices();
  int selection = choose_drink();

  print_selection(selection);

  // Weitere Funktionen, etwa:
  // pay_for_drink(selection);
  // dispense_drink(selection);
}
```