Arduino Einführung Teil 9

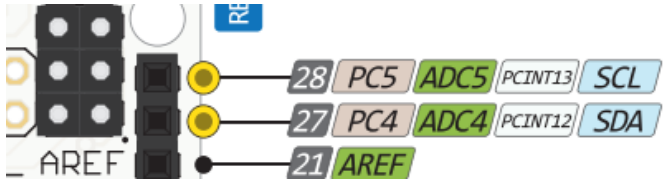
Taster-Platine 4x4

Wie gehe ich am besten vor?

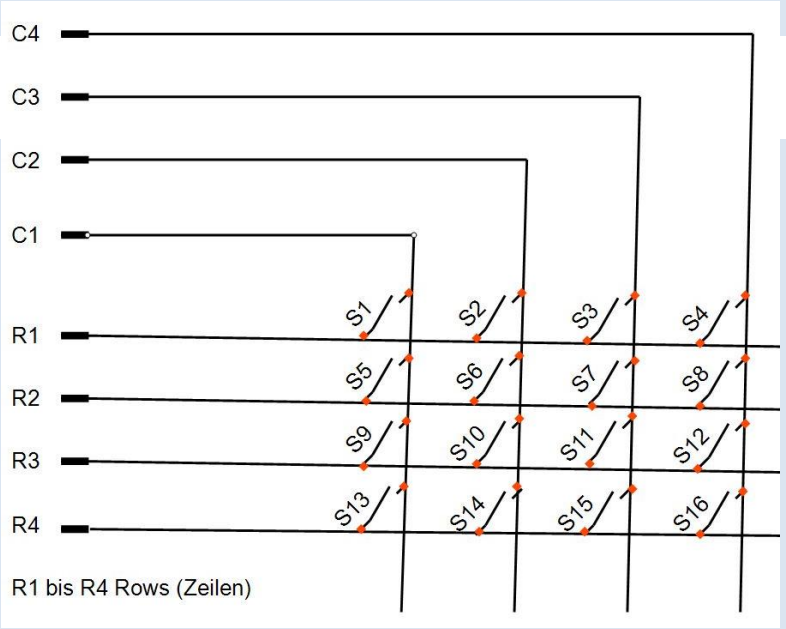
1. Was will ich machen?
2. Bauteile
3. Überlegungen zur Schaltung und Algorithmus
4. Zuordnung Arduino-Pins mit den Pins der Taster-Platine
5. Schaltplan Taster-Platine mit LCD
6. Zu Versuch 1 „ALL_TastPlatine_1.ino“ und „functions.ino“
7. Zu Versuch 2 „ALL_TastPlatine_2.ino“ und „functions.ino“
8. Sketch Versuch 1 Taster-Platine
9. Sketch Versuch 2 Taster-Platine
10. LCD-Funktionen

H39@email.de

Was will ich machen?	
Vorversuch Taster-Platine	Die Matrix der Reihen und Spalten untersuchen.
LCD in Betrieb nehmen	1. Ordner „LiquidCrystal_dfrobot“ in „…\arduino\libraries“ anlegen 2. Dateien kopieren (Anfrage an Enno)
Versuch 1 Taster-Platine	Taster-Platine betreiben Taste auf LCD anzeigen Konventionelle Programmierung
Versuch 2 Taster-Platine	Taster-Platine betreiben Taste auf LCD anzeigen Programmierung mit Ports und Bit-Operatoren

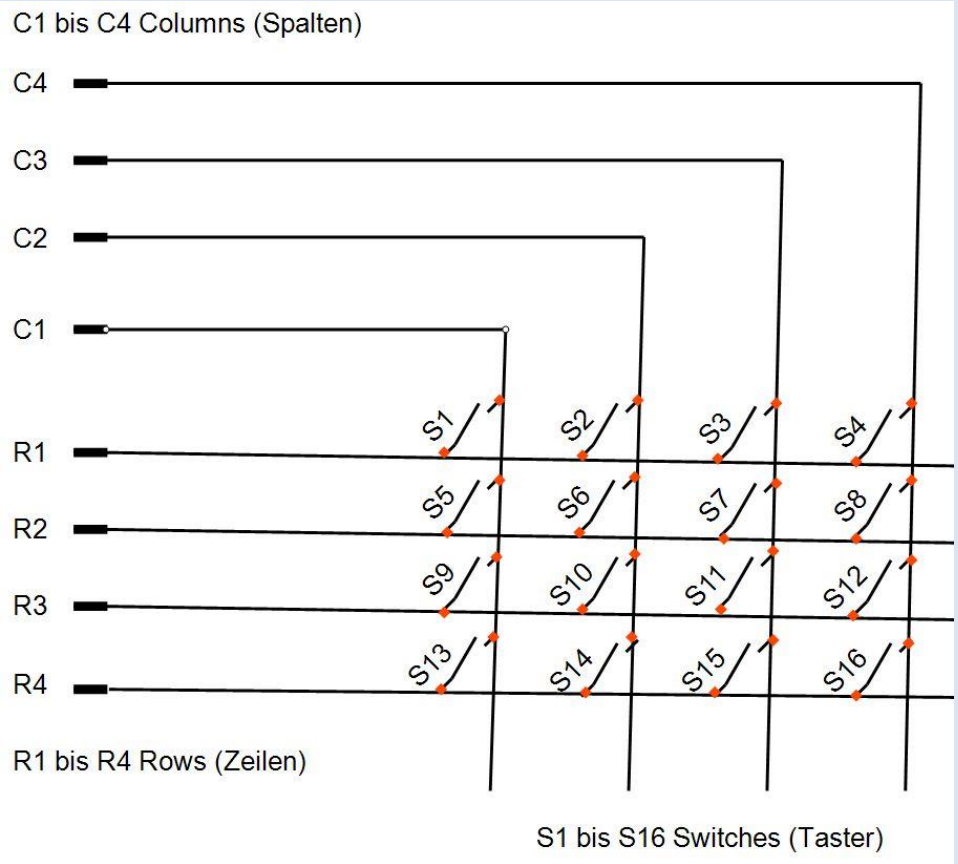
Bauteile?	
Vorversuch Tast-Platine	Steckbrett ALLNET-Bausatz: Tast-Platine Multimeter
LCD in Betrieb nehmen	ARDUINO ALLNET-Bausatz: LCD Anschlüsse: LCD — Arduino GND — Steckbrett minus (-) VCC — Steckbrett plus (+) SDA — Arduino SDA SCL — Arduino SCL 
Tast-Platine Versuch 1	Bauteile s.o.
Tast-Platine Versuch 2	Bauteile s.o.

Überlegungen zur Schaltung und Algorithmus	
Aufgabe	Ein Tastenduck (S1 bis S16) soll erkannt werden.
Stromkreise?	Wird Taster S9 gedrückt, so ergibt sich beginnend bei Stift R3 über den Taster S9 zum Stift C1 ein Stromkreis. Wegen der 16 Taster ergeben sich 16 Stromkreise.
Arduino OUTPUT-Pins	Wir verbinden die Pins R1 bis R4 der Taster-Platine mit Arduino-Pins im OUTPUT-Modus (+ 5V).
Arduino INPUT-Pins	Wir verbinden die Pins C1 bis C4 der Taster-Platine mit Arduino-Pins im INPUT-Modus.
Algorithmus	„S6“ sei gedrückt. In einer äußeren Schleife setzen wir die OUTPUT-Pins (R1 bis R4) nacheinander auf HIGH. In einer inneren Schleife fragen wir die INPUT-Pins (C1 bis C4) auf HIGH ab. Hier wird jeweils im 2. Durchlauf der äußeren und der inneren Schleife „S6“ gefunden.

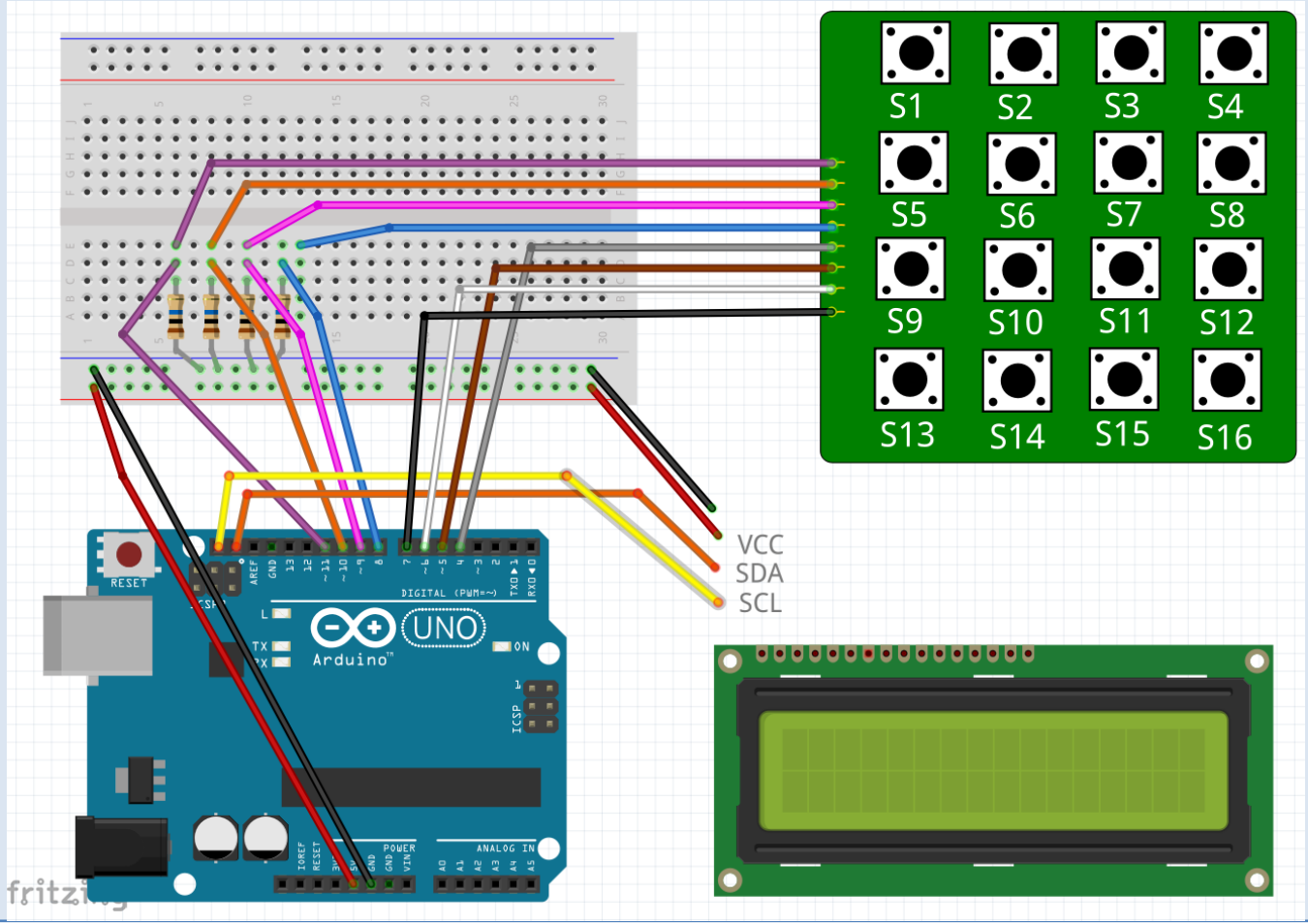


Zuordnung Arduino-Pins mit den Pins der Taster-Platine

Port Pin	Arduino Pins	Platine
PB3	11	C4
PB2	10	C3
PB1	9	C2
PB0	8	C1
PD4	4	R1
PD5	5	R2
PD6	6	R3
PD7	7	R4



Schaltplan Taster-Platine mit LCD



Zu Versuch 1 „ALL_TastPlatine_1.ino“ und „functions.ino“

Beschreibung ALL_TastPlatine1.ino	Der Sketch bindet die Library `<LiquidCrystal_I2C.h>` von dfrobot ein. Initialisierung der Arduino-Pins. In `setup()` Zuweisung der Modi `OUTPUT` und `INPUT`. In `loop()` Aufruf der Funktion `lookForKey()`, die im Tab `functions` programiert ist.
Funktion mit Rückgabewert	Werden Funktionen die mit einem Datentyp deklariert, geben einen Wert an die aufrufende Stelle zurück.: `int lookForKey(){ … }`. Mit dem Schlüsselwort `return` wird der Wert zurück gegeben: `return keyFound`. Dieser Wert kann beim Funktionsaufruf einer Variablen zugewiesen werden. `int key = lookForKey()`
Beschreibung functions.ino	In der äußeren while-Schleife werden die Reihen (`row`) R1 bis R4 durchlaufen. Am Anfang wird der zugehörige Pin auf `HIGH`, am Schleifenende auf `LOW` gesetzt. Am Schleifenende wird dann die Reihe um 1. erhöht (z.B. von `pinC1` auf `pinC2`). In der inneren while-Schleife werden für die aktuelle Reihe, die Spalten (`column`) C1 bis C4 durchlaufen. Wird in einer Spalte ein HIGH gefunden `(digitalRead(column) == HIGH)`, dann wird die Tastennummer `(keyCount) in keyFound` gespeichert. Danach werden in der inneren Schleife Spalte und Tastennummer um 1. erhöht.

Zu Versuch 2 „ALL_TastPlatine_2.ino“ und „functions.ino“	
Bitweise Operatoren	s. dazu: http://playground.arduino.cc/Code/BitMath Die Kommentare im Sketch erklären die Arbeitsweise.
Port Register z.B. PORTD	s. dazu: https://www.arduino.cc/en/Reference/PortManipulation Die digitalen Pins 0 bis 7 machen 8 Bits in einem Byte aus.
z. B. PIND	Das Register `PIND` ersetzt `digitalRead()`. Es enthält den Zustand im INPUT-Modus. Ergibt `Serial.println(PIND) B01000000`, so ist Bit 6 HIGH (Arduino Pin 6).
z.B. PORTD	Das Register `PORTD` ersetzt digitalWrite(). PORTD = `B00010000;` // setzt Bit 4 (Arduino Pin 4) auf HIGH
z.B. DDRD	Das Register DDRD ersetzt `pinMode()`. `DDRD = B11110000;` // setzt Arduino Pins 0 bis 3 als INPUT, Pin 4 bis 7 als OUTPUT `DDRD = DDRD | B11110000;` // wie oben, aber Pin 0 bis 3 bleiben unverändert.
	In der äußeren while-Schleife werden die Reihen (`row`) R1 bis R4 durchlaufen. Am Anfang wird der zugehörige Bit/Pin über das Port-Register `PORTD` auf „1“, am Schleifenende auf „0“ gesetzt. Am Schleifenende wird dann die Reihe um 1. erhöht.
	Um einen gesetzten Taster in den Spalten (`column`) C1 bis C4 zu finden führe ich 4 Zuweisungen aus. Der Trick besteht darin, das die Zuweisung eine Bedingung enthält, die `TRUE`, d.h. „1“ wird, wenn ein Bit/Pin (Reihe) auf `HIGH` steht. Ergibt `PINB` z.B. `B0010`, dann ist Bit 1/Pin 9 auf `HIGH` und die Bedingung `(PINB==B0010)` wird „1“. Damit wird „`column = (PINB== B0010) * 2`” zu `column = 2`.

Sketch Versuch 1 Taster-Platine

```
// ALL_TastPlatine_1.ino
// Library: LiquidCrystal_dfrobot
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

int pinR1 = 4;    //Arduino Pin 4
int pinR2 = 5;    //Arduino Pin 5
int pinR3 = 6;    //Arduino Pin 6
int pinR4 = 7;    //Arduino Pin 7

int pinC1 = 8;    //Arduino Pin 8
int pinC2 = 9;    //Arduino Pin 8
int pinC3 = 10;   //Arduino Pin 10
int pinC4 = 11;   //Arduino Pin 11

void setup() {
  lcd.begin();
  lcd.print("Taster druecken!");

  // digitale PIN 4 bis 7 auf OUTPUT
  pinMode(pinR1, OUTPUT);
  pinMode(pinR2, OUTPUT);
  pinMode(pinR3, OUTPUT);
  pinMode(pinR4, OUTPUT);
  // digitale PIN 8 bis 11 auf INPUT
  pinMode(pinC1, INPUT);
  pinMode(pinC2, INPUT);
  pinMode(pinC3, INPUT);
  pinMode(pinC4, INPUT);
}

void loop() {
  int key = lookForKey();
  if (key != 0) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Taster = S");
    lcd.print(key);
  }
  delay(300);
}
```

```
// functions.ino
int lookForKey(){
  int keyCount = 1;
  int keyFound = 0;

  int row = pinR1;
  while (row <= pinR4){
    digitalWrite(row, HIGH);

    int column = pinC1;
    while (column <= pinC4 ){
      if (digitalRead(column) == HIGH) keyFound = keyCount;
      column = column + 1;
      keyCount = keyCount + 1;
    }
    digitalWrite(row, LOW);
    row = row + 1;
  }
  return keyFound;
}
```

Sketch Versuch 2 Taster-Platine

```
// ALL_TastPlatine_2.ino
// Library: LiquidCrystal_dfrobot
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.begin();
  lcd.print("Taster druecken!");

 DDRD = DDRD | B11110000;
 DDRB = DDRB | B11110000;
}

void loop() {
  int key = lookForKey();
  if (key != 0) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Taster = S");
    lcd.print(key);
  }
  delay(300);
}
```

```
 // functions.ino
int lookForKey(){
  int portB = 0;
  int column = 0;
  int row = 0;
                                                  // die Reihen R1 (hier row = 0) bis
                                                  //              R4 (hier row = 3) durchlaufen
  while (row <= 3){
    PORTD = B00010000 << row;                     // anwenden des bitweise "Linksschieben":   "<<"
                                                  //                   Pin 4 => PD4 => B00010000 << 0 auf HIGH
                                                  //              bis Pin 7 => PD7 => B00010000 << 3 auf HIGH
    delayMicroseconds(1);                         // der Hardware Zeit geben
    portB = PINB;                                 // den Port B auslesen, Ergebnis B0001 bis B0004 und B0000
                                                  // die Spalte ermitteln, d.h.
    column =              (portB == B0001) * 1;   // (portB == B0001) * 1 bedeuted aus B0001 wird 1
    column = column + (portB == B0010) * 2;       // (portB == B0010) * 2 bedeuted aus B0010 wird 2
    column = column + (portB == B0100) * 3;       // (portB == B0100) * 3 bedeuted aus B0100 wird 3
    column = column + (portB == B1000) * 4;       // (portB == B1000) * 4 bedeuted aus B1000 wird 4

    if (column) break;                            // war eine Spalte (column) HIGH, dann Taster gefunden
    row = row + 1;
  }
  return (column != 0) * ( row * 4 + column);     // (column != 0) wenn TRUE, dann wird aus TRUE die "1"
                                                  // (column != 0) wenn FALSE, dann wird aus FALSE die "0"
                                                  // Beispiel: für die Reihe R3 (row = 2) und column = 3 wird
                                                  //          (   3   != 0) * (  2  * 4 +   3)
}                                                 //          (      1       *       11       ) also Taste 11
```

LCD-Funktionen		
LiquidCrystal_I2C	lcd(0x27, 16, 2)	Vor void()-setup einfügen Setzt die Adress des LCD, meist 0x27 Setzt die Anzahl Zeichen Setzt die Anzahl Zeilen Erzeugt ein Objekt der Klasse „LiquidCrystal_I2C“ mit Namen „lcd“
lcd.backlight()		Hintergrundbeleuchtung an
lcd.clear()		LCD löschen, Cursor oben links
lcd.setCursor(0,0) lcd.setCursor(0,1)		Setzt Cursor auf Zeichen 0 in Zeile 0 Setzt Cursor auf Zeichen 0 in Zeile 1
lcd.print("LED = ") lcd.print(sensorValueMap)		Schreibt die Zeichen „LED=„ ab der Cursorposition Schreibt den Wert „sensorValueMap „ ab der Cursorposition