Daten manuell in das Modul LED & KEY (TM1638) einlesen.

```
const int stb = 7;     // Strobe
const int clk = 9;     // Clock
const int dio = 8;     // Data

const byte NUMBER_FONT[] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7 (Bsp. Array-Index ist 7)
  0b01111111, // 8
  0b01101111  // 9
  };
void setup() {
  pinMode(stb, OUTPUT);
  pinMode(clk, OUTPUT);
  pinMode(dio, OUTPUT);

  digitalWrite(stb, LOW);      // (*)

  shiftOut(dio, clk, LSBFIRST, 0b10001111);

  clearLn();

  printZi(7,0,0);       // (**)
  printZi(3,0,1);
  printZi(4,1,0);
  printZi(1,0,1);
  printZi(2,0,0);

  digitalWrite(stb, HIGH);
 }

void loop() {
}

void clearLn() {
  for(int i = 0; i<8; i++) {
    shiftOut(dio, clk, LSBFIRST, 0b00000000);
    shiftOut(dio, clk, LSBFIRST, 0b10000000);
  }
}
// (***)
void printZi(int number, bool point, bool
led) {
  byte byte2 = NUMBER_FONT[number];
  if (point==1){
    bitSet(byte2, 7);
}
  shiftOut(dio, clk, LSBFIRST, byte2);
  if (led==1){
    shiftOut(dio, clk, LSBFIRST, 0b10000001);
}else{
    shiftOut(dio, clk, LSBFIRST, 0b10000000);
  }
```

Das Programm wurde im Wesentlichen von Dominik und Victor geschrieben.

Vorsicht: Diesen Text muss man sehr konzentriert (mehrmals) lesen. Also Musik aus, Kaugummi raus und los geht´s:

Bis zur Anweisung (*) sollte es keine Probleme geben.
Danach kommt mit shiftOut das Command-Byte, welches nur einmal gesendet wird.
Das anschließende Löschen des Displays mit clearLn(); erfolgt als ausgelagerte Methode (s.u.). In dieser Methode (auch „Prozedur“ genannt) werden durch eine for-Schleife 8-mal „leere“ Datenbytes übertragen, so dass alle 8 Anzeigen gelöscht sind.

Nun geht es bei (**) weiter. Jede einzelne Ziffer wird mit der gleichen Methode „printZi“ übertragen, die wieder ausgelagert wurde. Zusätzlich kann man hier in der zweiten Spalte eingeben, ob ein Komma gesetzt wird und in der 3. Spalte, ob die LED leuchten soll. Die Ziffer ist eine int-Variable, der Dezimalpunkt und die LED sind bool-Variablen, die 0 (false) oder 1 (true) sein können. Diese Prozedur (Methode) beginnt bei (***).

Zunächst kommt die Ziffer „7“. Das NUMBER_FONT-Array sagt uns, welche Segmente gesetzt werden müssen. Die Ziffer 7 steht an der 7. Stelle des Arrays. Das erste übertragende Byte war ja das Commandbyte. Das 2. Byte legt die anzusteuernden Segmente fest. In printZi wird die Ziffer mit „number“ gekennzeichnet. Dieses 2. zu übertragende Byte wählt nun mit
byte byte2 = NUMBER_FONT[number] die Binärzahl 0b00000111 aus dem Array aus. Die erste Stelle (links) dieses Byte besagt ja, ob dort ein DP gesetzt werden soll (1) oder nicht (0). Das wird hier durch bitSet raffiniert gemacht. Wenn (if-Abfrage) also bool point (die zweite Spalte in printZi) 1 ist, muss im byte2 an der 7. Stelle eine „1“ gesetzt werden. Vorsicht: Die Bits im Byte werden von 0 bis 7 durchnummeriert! Das geschieht durch bitSet(byte2, 7); im 2. Byte wird also gegebenenfalls die „1“ ganz links zusätzlich gesetzt. Anschließend wird durch shiftOut das byte2 übertragen.

Ähnlich verfährt man mit dem Setzen der LED über der Ziffer: Es wird abgefragt, ob in printZi an der 3. Stelle (bool led) eine **1** steht. Falls ja überträgt man 0b1000000**1** als 3. Byte, falls nein wird 0b1000000**0** übertragen. Die rechte Stelle des 3. Bytes entscheidet ja, ob die LED leuchtet oder nicht.

Diese Methode wird hier für alle 5 Ziffern (7,3,4,1,2) nacheinander angewendet.