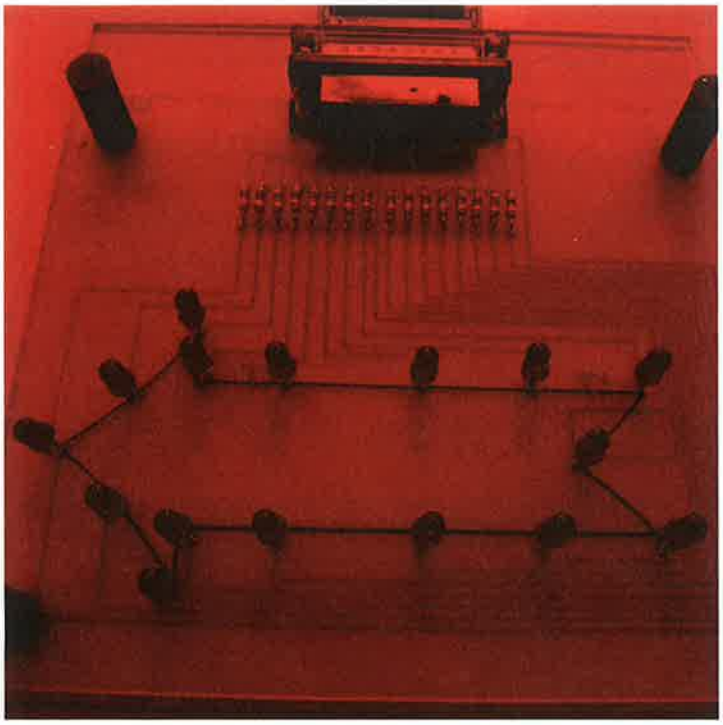
E 3915

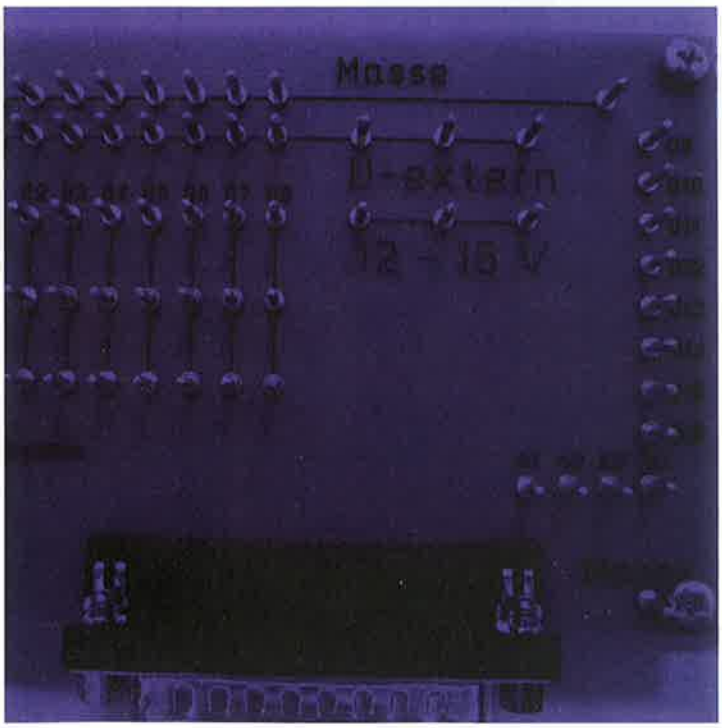
tu 161

Neckar-Verlag

3. Quartal 2016

ZEITSCHRIFT FÜR TECHNIK IM UNTERRICHT





ISSN 0342-6254

Wordclock

Ein Steuerungsprojekt für den Technikunterricht

Von Johannes Lehmke

Abb. 1

Die Uhrzeitangabe einer „Wordclock" erfolgt über Sätze im Klartext. Diese Sätze werden aus Wortbausteinen zusammengesetzt. So liest man auf der „Wordclock" statt 10:10 den Satz „ES IST ZEHN NACH ZEHN UHR". Die Uhr könnte so wie die hier abgebildete aussehen.

Die besondere Attraktivität dieses Projekts liegt darin, dass die Schülerinnen und Schüler ein Projektprodukt konzipieren, planen und fertigen und es dann auch als originelle und sehr schöne Erinnerung mit nach Hause nehmen können.

Die Herausforderung dieses Steuerungsprojekts ist die hohe Anzahl der zu steuernden Elemente, 22 Wortbausteine. Das LaunchPad und auch der Arduino haben aber nur eine begrenzte Anzahl von maximal 16 Anschlusspins. So muss die Anzahl der Steuerleitungen erweitert werden.

Bau der Wordclock

Für die Fertigung einer Wordclock im Unterricht 1 scheint das Format DIN A 4 (20 cm x 30 cm) empfehlenswert, da dann die Wortbausteinblende (siehe Schnittzeichnung) mit einem Textverarbeitungsprogramm wie z. B. Word erstellt werden und auf einer Polyester-Folie mit einem Laserdrucker ausgedruckt werden kann.

In Grenzen ist die Auswahl der Wortbausteine und ihre Aufteilung auf der Fläche frei. Mit „dreiviertel" könnten die Schülerinnen und Schüler auch eine „bayrische" Wordclock bauen.

Auf eine Grundplatte aus Sperrholz oder Forex werden Abschnitte von U-Profilen aus weißem Kunststoff (Baumarkt) geklebt und durch Stege aus 3-mm-Forex in Abschnitte unterteilt.

So entsteht für jeden Wortbaustein ein Ausleuchtungsschacht, in den je ein selbstklebender LED-Streifen geklebt wird. Die LED-Streifen werden an einem 10-pol-Pfostenstecker angeschlossen (siehe Schnittzeichnung – Abb. 2).

Die Wortbausteinblende wird auf die U-Profile gelegt und mit einer Plexiglasplatte abgedeckt. Die Plexiglasplatte, die Wortbausteinblende und die Ausleuchtungsschächte werden durch einen Winkelrahmen aus schwarzem Kunststoff (Baumarkt) zusammengehalten.

Experimentelle Erprobung der Wordclock

Für einen ersten Test soll die Wordclock zunächst von Hand betrieben werden. Dazu kann mit Bausteinen aus dem EBS-System[2] die nachstehende Experimentierumgebung aufgebaut werden. Mit 8 Schiebeschaltern werden die Adern der Leitung zur Wordclock mit +12 V verbunden. Für die drei Leitungen, mit denen die Wortbausteine der Wordclock ausgeleuchtet werden sollen, ist nur jeweils eine Experimentierumgebung vorgesehen, die nacheinander genutzt werden kann (siehe Abb. 3).

Den Zustand der beiden Taster an der Wordclock kann man mit den ersten beiden LEDs prüfen.

Ziel dieser Erprobung ist es, die Funktionstüchtigkeit der Wordclock festzustellen und die Steckerbelegung der drei 10-pol-Stecker zu ermitteln.

Ein denkbares Ergebnis könnte dann so wie in der nachstehenden Darstellung aussehen (siehe Abb. 4).

Programmierumgebung für die Wordclock

Wenn die fertiggestellte und getestete Wordclock mit dem LaunchPad programmiert werden soll, braucht man zwischen LaunchPad und Wordclock das „Unipolar Projektmodul"[2], das auf das LaunchPad gesteckt wird. Mit dem „Unipolar Projektmodul" kann man 8 Signale für eine Steuerung in das LaunchPad einlesen und über drei Stecker 24 Lasten mit etwa 9 V versorgen.

Jeder dieser drei Stecker ist mit je einem Schieberegister (C4094) verbunden. Die drei Schieberegister sind auf dem „Unipolar Projektmodul" in Reihe geschaltet. Das LaunchPad selbst wird über den USB-Stecker vom angeschlossenen Rechner versorgt. Das „Unipolar Projektmodul" und die angeschlossene Wordclock werden per Hohlstecker von einem 12-V-Steckernetzteil versorgt (siehe Abb. 5).

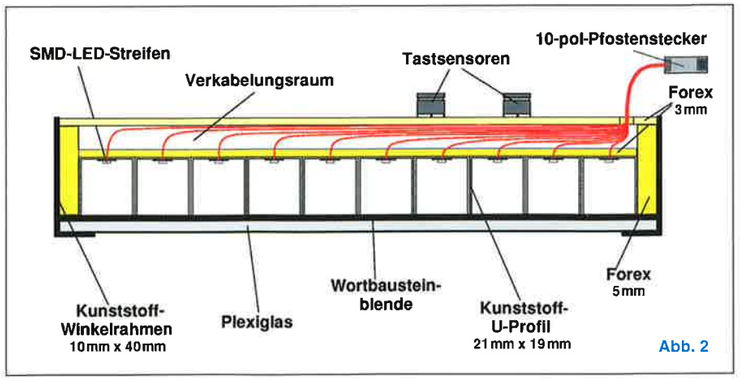


Abb. 2

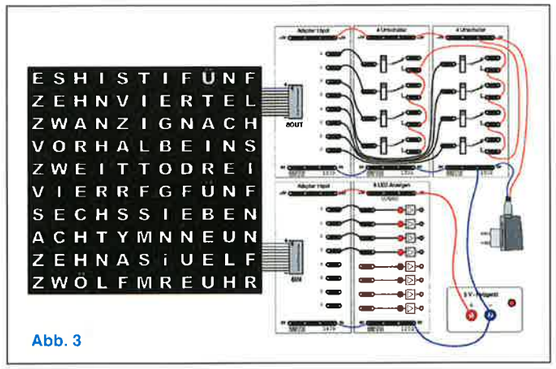


Abb. 3

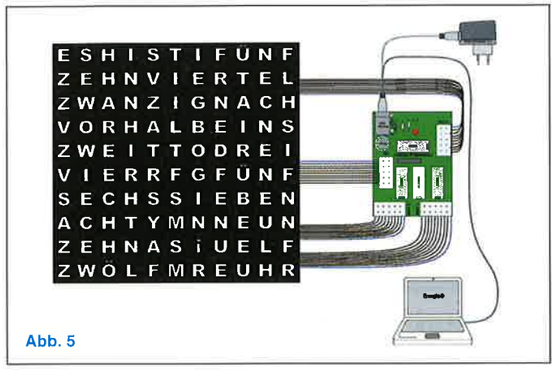


Abb. 5

Abb. 4

Steckerbelegung UZ1

ES	UHR	ZWEI	VIER	GND
+ 12V	IST	EINS	DREI	FÜNF

Steckerbelegung ZUZ

FÜNF	VIERTEL	HALB	NACH	GND
+ 12V	ZEHN	ZWANZIG	VOR	

Steckerbelegung UZ1

SECHS	ACHT	ZEHN	ZWÖLF	GND
+ 12V	SIEBEN	NEUN	ELF	

Steckerbelegung SENS

RESET				GND
+ 12V	START			

Exkurs zum Schieberegister

Schieberegister sind digitalelektronische Schaltungen, die mehrstellige binäre Signale taktgesteuert aufnehmen und abspeichern. Der besondere Vorteil von Schieberegistern ist es, dass sich mit ihnen die Anzahl der Ausgänge fast beliebig vergrößern lässt.

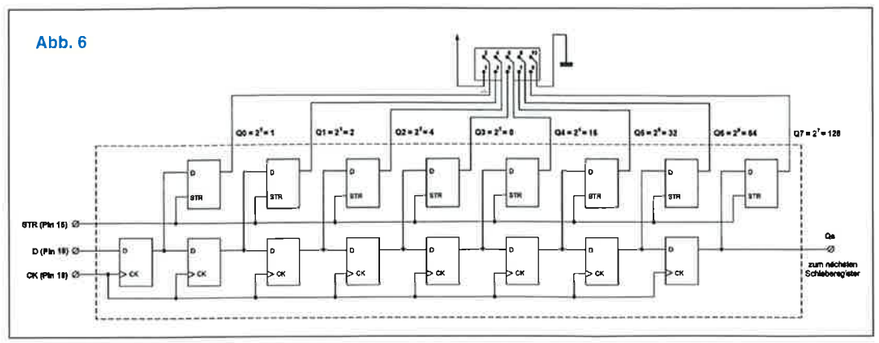


Abb. 6

Über zwei Leitungen (D, CK) werden die Signale seriell in das Schieberegister eingeschoben und dann parallel wieder ausgegeben. So lassen sich, wie im Fall der Wordclock, mit zwei Leitungen 22 Lasten steuern. Erst wenn der Schiebevorgang beendet ist, werden die Signale mit dem Stro-

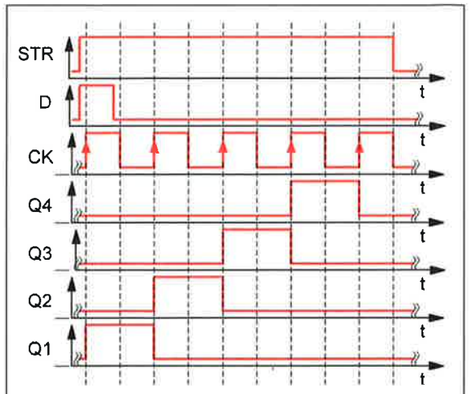


Abb. 7

be-Signal (STR) in den Außenspeicher übernommen und für die Steuerung nutzbar (siehe Abb. 6).

Über die Datenleitung (D) wird ein Signal (1 oder 0) eingelesen und dann bei jedem ansteigenden Clock-Signal (CK) eine Speicherzelle weitergeschoben. Das geschieht dann 8-mal auf die gleiche Weise. Der zeitliche Verlauf für die ersten vier Takte ist nebenstehend dargestellt.

Mit dem Strobe-Signal (STR) werden die Daten im Schieberegister in den Außenspeicher übernommen, dann über die Ausgangsbuchse ausgegeben und schließlich über die Leitungen des Flachbandkabels an das Steuerobjekt weitergegeben (siehe Abb. 7).

Den einzelnen Wortbausteinen der Wordclock sind die einzelnen Bits eines der drei Bytes zugeordnet. Wenn man zum Beispiel über das Kabel UZ1 die Wordbausteine „ES IST FÜNF UHR“ ausleuchten will, dann sieht das Byte folgendermaßen aus (siehe Abb. 8).

Wenn dieses Byte mit dem niedrigstwertigen Bit zuerst in das Schieberegister geschoben wird, landet das niedrigstwertige Bit nach dem 8. Takt auf Schieberegisterposition Q8. Wenn das Byte mit dem höchstwertigen Bit zuerst in das Schieberegister geschoben wird, landet das höchstwertige Bit nach dem 8. Takt auf Schieberegisterposition Q8.

2^7 = MSB	2^6	2^5	2^4	2^3	2^2	2^1	2^0 = LSB
1	0	0	0	0	1	1	1

Abb. 8: Die dieser Bitfolge entsprechende Dezimalzahl ist: 128 + 4 + 2 + 1 = 135

```
//Programmbeispiel                          Zeile 1
  int D = 19;                               Zeile 2
  int CK = 18;                              Zeile 3
  int STR = 15;                             Zeile 4
  int UZ[3] = {15, 23, 39};                 Zeile 5
  int i;                                    Zeile 6
void setup()                                Zeile 7
  {                                         Zeile 8
  pinMode (STR, OUTPUT);                    Zeile 9
  pinMode (D, OUTPUT);                      Zeile 10
  pinMode (CK, OUTPUT);                     Zeile 11
  }                                         Zeile 12
void loop()                                 Zeile 13
  {                                         Zeile 14
  for (i=0; i<3; i=i+1)                     Zeile 15
    {                                       Zeile 16
    digitalWrite (STR, HIGH);               Zeile 17
    shiftOut (D, CK, MSBFIRST, UZ[i]);      Zeile 18
    digitalWrite (STR, LOW);                Zeile 19
    delay (2000);                           Zeile 20
    }                                       Zeile 21
  }                                         Zeile 22
```

Abb. 9:

Zeile 1: Überschriften und Kommentare gehören nicht zum Programm und werden mit // vom Programm abgetrennt.

Zeile 2 - 4: In diesen Zeilen werden die Portvariablen Daten (D), Clock (CK) und Strobe (STR) initialisiert und den entsprechenden Ausgängen zugewiesen.

Zeile 5: In dieser Zeile wird für die Uhrzeitvariable UZ ein sogenanntes Array initialisiert. In diesem Array werden Dezimalzahlen (15, 23, 39) den einzelnen Komponenten UZ[i] zugewiesen. Die Indizes i durchlaufen dabei die Zahlen 0–2.

Zeile 6: Hier wird die Schleifenvariable i initialisiert.

Zeile 7 – Zeile 12: Im Setup-Block werden die Portvariablen STR, D und CK als Ausgänge deklariert. Dieser Block wird nur einmal beim Start des Programms abgearbeitet.

Zeile 13/14 bis Zeile 22: Im Loop-Block findet der eigentliche Steuerablauf in einer Endlosschleife statt.

Zeile 15: In dieser Zeile wird eine for-Schleife mit drei Durchläufen gestartet.

Zeile 16/21: Die Befehle und Anweisungen, die in dieser for-Schleife ausgeführt werden sollen, werden durch die beiden geschweiften Klammern zusammengefasst.

Zeile 18: In dieser Zeile wird die Komponente i der Uhrzeitvariablen UZ über die Datenleitung D und die Clockleitung CK mit dem höchstwertigen Bit (MSB) zuerst in ein Schieberegister geschoben.

Zeilen 20: Durch diese Zeile wird der Programmablauf um 2 s verzögert.

Zeilen 22: Mit dieser Zeile springt das Programm wieder an den Anfang zurück.

Entwicklung des Steuerprogramms für die Wordclock

Ein Steuerprogramm für die Wordclock – bis DREI Uhr – wurde hier als Beispiel ausgewählt, um es ausführlich zu kommentieren. Beim Durcharbeiten dieses Beispielprogramms sollen die Schülerinnen und Schüler die Gelegenheit bekommen, die elementaren Kenntnisse und Fähigkeiten zu erwerben, die notwendig sind, um dann weitgehend selbständig die Programme zum Steuern der Wordclock zu entwickeln (siehe Abb. 9).

Betrieb der Wordclock für die vollen Stunden

Für die Steuerung der Wordclock mit einem Programm wie dem im Programmbeispiel müssen zunächst die Komponenten für die Uhrzeitvektoren UZ1[i] bzw. UZ2[i] ermittelt werden. Für die Berechnung der Komponenten muss man die Codezahlen der Wortbausteine, die in jedem Programmschritt aufleuchten sollen, addieren.

Wenn als Erstes „ES IST ZWEI UHR“ aufleuchten soll, dann berechnet man die Komponente des Uhrzeitvektors UZ1 aus den Codezahlen 1 + 2 + 4 + 16 = 23, oder wenn „ES IST DREI UHR“ aufleuchten soll, dann ergibt sich die Komponente durch: 1 + 2 + 4 + 32 = 39.

Eine Wordclock, die die vollen Stunden anzeigen soll, hat 12 Programmschritte. Jeder der beiden Uhrzeitvektoren (UZ1, UZ2) hat 12 Komponenten, die wie oben stehend bestimmt werden. Damit ergibt sich:

UZ1[12] = {15, 23, 39, 71, 135, 7, 7, 7, 7, 7, 7, 7},

UZ2[12] = {0, 0, 0, 0, 0, 1, 2, 4, 8, 16, 32, 64}.

Mit der Erstellung der Uhrzeitvektoren kann man das Programm für die Anzeige der vollen Stunden aufschreiben.

Nach der Initialisierung und der Deklarierung der Portvariablen (D, CK, STR), der Uhrzeitvariablen (UZ1, UZ2) und der Schleifenvariablen i kommt im Loop-Block eine for-Schleife zum Einsatz, die die einzelnen Uhrzeiten hochzählt (siehe Abb. 10).

```
//Wordclock für volle Stunden
  int D = 19;
  int CK = 18;
  int STR = 15;
  int UZ1[12] = {15, 23, 39, 71, 135, 7, 7, 7, 7, 7, 7, 7};
  int UZ2[12] = {0, 0, 0, 0, 0, 1, 2, 4, 8, 16, 32, 64};
  int i;
void setup()
  {
  pinMode (D, OUTPUT);
  pinMode (CK, OUTPUT);
  pinMode (STR, OUTPUT);
  }
void loop()
  {
  for (i=0; i<12; i=i+1)
    {
    digitalWrite (STR, HIGH);
    shiftOut (D, CK, MSBFIRST, UZ1[i]);
    shiftOut (D, CK, MSBFIRST, UZ2[i]);
    digitalWrite (STR, LOW);
    delay (6000);
    }
  }
```

Abb. 10

In der Schleife wird zwischen den geschweiften Klammern zunächst in die Portvariable STR eine „1“ geschrieben, damit die Ausgänge den Schiebevorgang nicht anzeigen.

Danach wird mit dem Befehl „shiftOut“ über die Datenleitung „D“ und mit der Clockleitung „CK“ jeweils eine Komponente der Uhrzeitvariablen UZ1 und dann eine Komponente der Uhrzeitvariablen UZ2 in die Schieberegister geschoben. Um das Ergebnis der beiden Schiebevorgänge sehen zu können, muss in die Portvariable (STR) eine „0“ geschrieben werden.

Danach sollte das Programm eigentlich eine Stunde Pause machen. Im Unterricht könnte man sich mit den Schülerinnen und Schülern zunächst auf eine Pause von 6 s einigen.

Vollständiger Betrieb der Wordclock

Beim vollständigen Betrieb der Wordclock werden neben den vollen Uhrzeiten auch die Zwischenuhrzeiten angezeigt. Dazu muss ein Vektor (ZUZ) für die Zwischenuhrzeiten initialisiert werden. Auch dieser Vektor hat 12 Komponenten, die überwiegend aus mehreren Wortbausteinen bestehen.

Um zum Beispiel die Komponente für die Zwischenuhrzeit „ FÜNF VOR HALB“ zu ermitteln, nimmt man die Codezahlen der einzelnen Wortbausteine für die Zwischenuhrzeiten aus der vorstehenden Steckerbelegung:

FÜNF = 1, VOR = 32 und HALB = 64. Das ergibt: FÜNF VOR HALB = 97.

Die weiteren Komponenten bestimmen sich in gleicher Weise und ergeben den Vektor für die Zwischenuhrzeiten (siehe Abb. 11)

Wie die beiden Uhrzeitvektoren wird nun auch der Vektor für die Zwischenuhrzeiten über den Stecker „8OUT3“ in die Schieberegister des „Unipolar Projektmodul“ geschoben.

Den schnellen Vorlauf erreicht man durch die Abfrage der Taste 1 innerhalb der „if-Anweisung“. Wenn die Taste nicht gedrückt ist, wird die „for-Schleife“ 100-mal mit jeweils einer

```
//Wordclock vollständig mit schnellem Vorlauf

  int D = 19;                                                    //Daten auf Pin 19
  int CK = 18;                                                   //Clock auf Pin 18
  int STR = 15;                                                  //Strobe auf Pin 15
  int T1 = 2;                                                    //Taste 1 auf Pin 2
  int UZ1[12] = {15, 23, 39, 71, 135, 7, 7, 7, 7, 7, 7, 7};      //Uhrzeiten 1 - 5
  int UZ2[12] = {0, 0, 0, 0, 0, 1, 2, 4, 8, 16, 32, 64};        //Uhrzeiten 6 - 12
  int ZUZ[12] = {97, 64, 81, 40, 36, 34, 33, 0, 17, 18, 20, 24}; //Zwischenuhrzeiten
  int i;
  int k;
  int l;

void setup()
  {
  pinMode (D, OUTPUT);
  pinMode (CK, OUTPUT);
  pinMode (STR, OUTPUT);
  pinMode (T1, INPUT);
  }

void loop()
  {
  for (i=0; i<12; i=i+1)                    //Index für Rahmentext und volle Uhrzeiten
    {
    for (l=0; l<12; l=l+1)                  //Index für Zwischenuhrzeiten
      {
      for (k=0; k<100; k=k+1)               //Index für 100 Durchläufe: 100x3=300s
        {
        digitalWrite (STR, HIGH);           //Ausgang gespeichert
        shiftOut (D, CK, MSBFIRST, UZ1[i]); //8OUT1
        shiftOut (D, CK, MSBFIRST, UZ2[i]); //8OUT2
        shiftOut (D, CK, MSBFIRST, ZUZ[l]); //8OUT3
        digitalWrite (STR, LOW);            //Ausgang durchgestellt
        if (digitalRead (T1) > LOW)         //Tastenabfrage
        delay (2);                          //Pause 2ms schneller Vorlauf
        else
        delay (3000);                       //Pause 3s für den Lauf der Uhr
        }
      }
    }
  }
```

Abb.11

Pause von 3 s x 100 = 300 s = 5 min durchlaufen (Normalbetrieb). Wenn die Taste gedrückt ist, ist die Durchlaufzeit 0,02 s x 100 = 2 s (schneller Vorlauf).

Beim weiteren Programmieren der Wordclock ist es möglich, die Ziele und Aufgaben ansteigend komplexer zu machen, zum Beispiel durch die Verwendung des Befehls „millis“. Die Pausenzeiten sind zwar recht genau, aber zwischen den Pausen geht durch die Bearbeitung der Befehle und Anweisungen auch Zeit verloren, die die Uhr etwas ungenau macht.

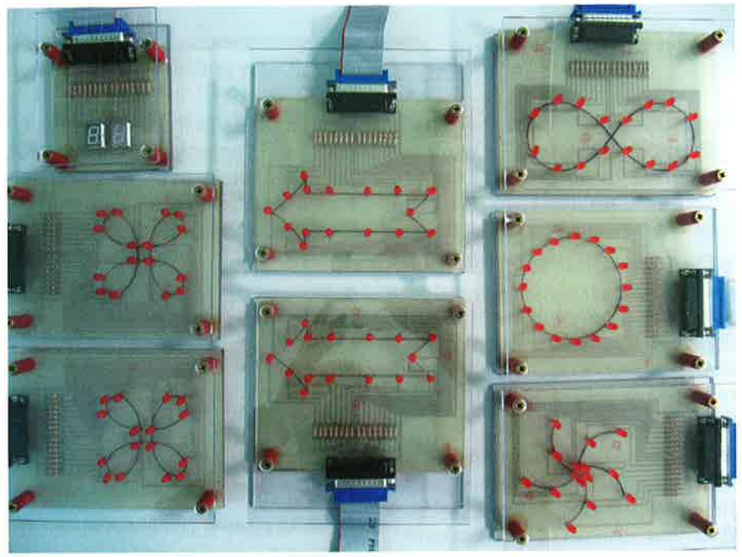
Literatur

1 J. Lehmke, Stt M 2.25 Wordclock – Fertigung und Betrieb, in TU Lehmke. de > Steuerungstechnik > Material

2 t.u.medien, Gesellschaft für Unterrichtsmedien mbH, Händelstraße 7, 59348 Lüdinghausen, Tel.: 02591 940088, eMail: info@tu-medien.de

CIUS 3 Weiterentwicklung des universellen Mediensystems CIUS zum Steuern und Regeln

Von Andreas Marx



Es sind 18 Jahre vergangen, seit CIUS2[1] auch in TU vorgestellt wurde. Das Mediensystem hat an den Erfolg von CIUS1 angeschlossen und seither in vielen Schulen, Hochschulen, Seminaren und darüber hinaus Anwendung gefunden. Man könnte nun sagen, CIUS2 ist volljährig geworden. Aber die Entwicklung bzw. Weiterentwicklung ist damit nicht beendet. Im Gegenteil, es ist ein Zeitpunkt gekommen, CIUS2 als CIUS3 wieder den aktuellen Anforderungen der fortentwickelten IuK-Technik einerseits und den didaktischen Anforderungen andererseits anzupassen. Diese Anpassungen und Weiterentwicklungen soll dieser Beitrag beschreiben.

Zahlreiche inhaltliche Anwendungsvarianten und innovative didaktische Ideen für die unterrichtliche Umsetzung des Themenfeldes Messen-Steuern-Regeln (M-S-R) wurden seither mit CIUS realisiert und teilweise auch publiziert, Beispiele siehe Literatur. CIUS wird für Projektarbeiten und Projektprüfungen ebenso eingesetzt wie für fachinterne Überprüfungen, fachpraktische Jahresarbeiten und andere Prüfungsformen des Technikunterrichts sowie Prüfungsleistungen an Hochschulen.

Die Zukunftsfähigkeit und Optimierung von CIUS sind die besonderen Anliegen der Fortentwicklung und gleichzeitig Beitrag zur Kontinuität und Innovation der Didaktik des allgemeinbildenden Technikunterrichts. Die Bildungspläne und Studienordnungen haben in den über 26 Jahren,[2] seit es CIUS gibt, den Inhaltsbereich IuK gestärkt und erweitert. Dennoch zeigen sich im Unterrichtsalltag und auch Seminaralltag an Hochschulen noch Entwicklungspotenziale auf, da sich die Fertigung von Modellen mit häufig geringer Qualität und Zuverlässigkeit immer wieder als Zentrum der konkreten Lernsituation in den Vordergrund schiebt und dann häufig keine Zeit mehr verbleibt, mit diesen Modellen auch unterschiedliche Steuerungs- und Regelungsaufgaben mit Alltagsbezug zu realisieren. Ein Zitat aus den Leitgedanken zu CIUS1: „Bei der Konzipierung von CIUS sollte ganz bewußt der bisher meist beschrittene Weg verlassen werden, wonach das Interface mit all seinen Problemen vom Aufbau bis zur Programmierung den Hauptanteil unterrichtlicher Aktivitäten ausmachte und für das Lösen von konkreten Problemstellungen […] viel zu wenig Zeit blieb.“[3] Diese Aussage hat nichts an Aktualität eingebüßt. Immer wieder tauchen kometenhafte Neuerscheinungen von Interfaces auf, die meist nur kurzzeitig die Diskussion mitbestimmen, aber bald schon wieder vom Markt bzw. aus der Schule verschwinden. Viele Lehrkräfte können ein Lied davon singen, wie oft sie schon ihre Unterrichtsvorbereitungen einstampfen konnten, weil die Systeme sich (schulisch) nicht bewährt hatten oder untergegangen waren.

Eine Perspektive sollte im Fokus des Technikdidaktikers stehen: Der Technikunterricht muss sich noch stärker darauf konzentrieren, technische Problemlösungen ins Zentrum der technischen Bildung zu stellen, statt die Schüler/-innen überwiegend mit Fertigungsaufgaben zu beschäftigen.[4] Fertige Modelle, stabil, unterrichtsrobust und schnell angeschlossen sollen dazu beitragen.

Das Konzept (Pflichtenheft) von CIUS zusammengefasst:

1 CIUS: Computer Interface für Universelles Steuern (und Regeln).

2 Entwicklungszeitraum CIUS1: 1990-1992.

3 Walter Barth, Wolfdieter Grötzinger, (1994): Messen, Steuern und Regeln, Landesinstitut für Erziehung und Unterricht Stuttgart (Materialien Natur und Technik T 28).

4 Vgl. Helmuth Fies (1995): Bereiten wir unsere Schüler auf eine Technik von gestern vor? In: tu 75, 1995, S. 12 ff.