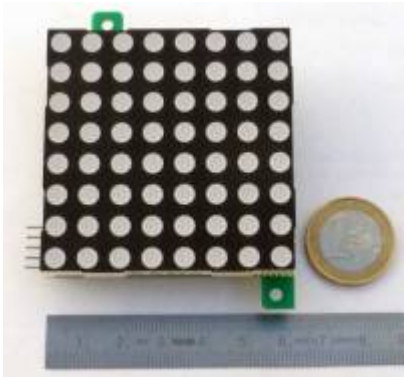
PixBlock



Ein Display, so wie du es brauchst

- Helles, farbstarkes Dot-Matrix-Display, 60 x 60 mm
- 64 Bicolor-Pixel (rot, grün)
- Modularer Aufbau: Einzelne PixBlocks lassen sich verketten und beliebig anordnen
- Präzise LED-Stromquellen
- Randloser Einbau möglich
- Einfache Ansteuerung über SPI-Schnittstelle

Die zugehörige Arduino-Library ermöglicht die einfache Ansteuerung mit folgenden Eigenschaften:

- Betrieb als Text- und Grafik-Display
- 15 verschiedene Farbschattierungen
- Automatische Laufschriften
- Mehrere Zeichensätze (ASCII, ISO8859, Symbole)

Beschreibung

Der PixBlock ist ein zweifarbiges (rot + grün) Dot-Matrix-Display mit 8 x 8 Pixeln. Es können beliebige Mikrocontroller, z. B. Arduino, PIC oder ARM, zur Ansteuerung verwendet werden. Über eine SPI-Schnittstelle lassen sich die 64 Pixel im Multiplex-Verfahren betreiben. Dazu werden periodisch 16 Bit zum Ein- / Ausschalten der 16 Leuchtdioden einer Spalte (8 rote + 8 grüne) übertragen. Das Weiterschalten auf die nächste Spalte geschieht automatisch mit dem Latch-Signal.

Alle grundlegenden Funktionen vom Multiplexing bis zur Laufschrift werden von der Arduino-Library erledigt. Sie wurde in C++ für Atmel-AVR-Prozessoren geschrieben. Eine Anpassung an andere Prozessoren ist leicht möglich.

Jeder PixBlock läßt sich optional mit einem eigenen Mikrocontroller (ATtiny84A) bestücken, der sich

um die Ansteuerung von bis zu 10 verketteten PixBlocks kümmert. Hierdurch erhält man ein intelligentes Display, das völlig selbständig - ohne externen Prozessor - arbeiten kann. Ebenso haben auf der Platine bei Bedarf noch drei Taster und ein EEPROM Platz.

Aufbau

Der Aufbau ist schnell erledigt. Die SMD-Bauteile sind bereits vorbestückt. Entsprechende Fähigkeiten zum Aufbau elektronischer Schaltungen und im sicheren Umgang mit den Werkzeugen werden vorausgesetzt.

Werkzeug

Lötwerkzeug

Man benötigt die üblichen Werkzeuge zum Löten von Elektronikschaltungen. Wer den optionalen Mikrocontroller bestücken will, sollte dünnes Lötzinn(0,5 mm) und eine Pinzette parat haben.

Bauteile

PixBlock-Komponenten

Der Bausatz enthält:

- 1 vorbestückte PixBlock-Platine
- 1 Dot-Matrix-Display
- 1 Stiftleiste 5-polig, gewinkelt
- 1 Buchsenleiste 5-polig, gewinkelt

Je nach geplanter Verwendung können noch zusätzlich Teile sinnvoll sein. Diese sind nicht im Bausatz enthalten.

- Distanzbolzen mit Schrauben
- Schraubensicherungslack

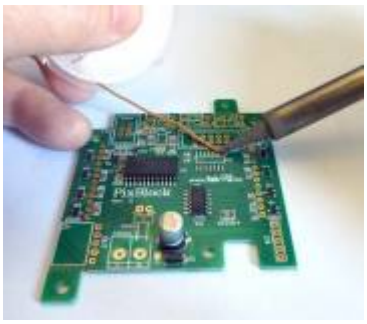
- Stecker für den Stromanschluss
- Wannenstecker 10-polig oder Pfostenleiste 2×5-polig
- Mikrocontroller ATtiny84A (SO14-Bauform)
- Wannenstecker 6-polig für ISP-Anschluss
- serielles EEPROM (z. B. 25LC256, SO8-Bauform)
- Taster (6 x 6 mm, gerade oder gewinkelt)

Bauanleitung

Wichtig: Das Display wird erst ganz zum Schluß aufgelötet.
Vorher sollte man sich genau überlegen, welche zusätzlichen Bauteile man eventuell noch bestücken möchte. Ist das Display erst einmal aufgelötet, verdeckt es die Platine und die Bestückung weiterer Bauteile wird dadurch sehr schwierig bzw. unmöglich.

1. Wer den optionalen Mikrocontroller bestücken möchte, sollte damit beginnen. Anderenfalls kann man gleich zum nächsten Schritt gehen.

   Zuerst ist es ratsam von den Pads das überschüssige Lötzinn mit Lötlitze abzusaugen. Danach ein Pad verzinnen und den Prozessor dort mit nur einem Beinchen festlöten. Falls nötig kann die Position des Prozessors jetzt noch korrigiert werden. Wenn alle Beinchen exakt mittig über dem jeweiligen Pad liegen, die restlichen Pins festlöten.




2. Jetzt werden die 5-polige Stiftleiste und die 5-polige Buchsenleiste eingelötet. Beim ersten PixBlock in einer Kette kann man die Stiftleiste natürlich auch durch einen anderen Verbinder, z. B. eine gerade Stiftleiste, ersetzen, damit sie die Platine nicht überragt.

3. Will man für die Spannungsversorgung eine Steckverbindung benutzen, kann man einen entsprechenden Anschluss (z. B. Stecker 2,54 mm oder Schraubklemme 5 mm) bestücken.

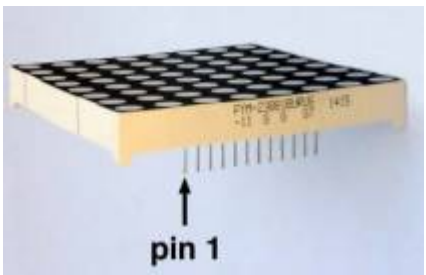


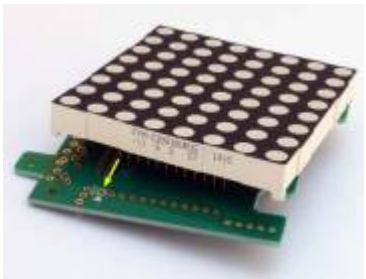
4. Nun muss man sich überlegen, welche Extra-Bauteile man noch bestücken möchte. Hier kommen Befestigungsbolzen, Taster oder zusätzliche Stecker bzw. Stiftleisten in Frage. Die inneren Befestigungsbolzen, deren Schrauben vom Display später überdeckt werden, sollten unter Verwendung von Schraubensicherungslack bzw. Klebstoff montiert werden, um ein versehentliches Lösen zu verhindern.

   Für den optionalen Mikrocontroller wird auf alle Fälle auch der 6-polige Programmieranschluss benötigt. Ausserdem kann die Bestückung einer 2-poligen Stiftleiste für den Reset-Jumper J2b sinnvoll sein. So lässt sich der Controller später auch abschalten, falls man ihn nicht mehr benötigt.



5. Zum Schluss wird das Display eingesetzt. Dabei ist auf die richtige Position zu achten. Wenn man auf die Typenbezeichnung des Displays schaut liegt Pin 1 vorne links. Dieser Pin muss in das eckige Pad, welches durch ein Dreieck gekennzeichnet und mit „pin 1“ beschriftet ist. Das Display so verlöten, dass es an drei Ecken auf der Platine aufliegt. Die vierte Ecke hängt in der Luft, da die Platine hier eine Aussparung besitzt.





So sieht ein fertig bestückter PixBlock aus. Hier wurden noch drei Distanzbolzen montiert und die überstehenden Befestigungslaschen entfernt.

Hardware

Die PixBlock-Hardware wird ausführlich durch den Schaltplan und den Bestückungsplan beschrieben. Beide stehen im Downloadbereich zur Verfügung.

Technische Daten

- Dot-Matrix-Display mit 8×8 zweifarbigen (rot / grün) LEDs
- Spannungsversorgung: 4,0 V - 5,0 V
- Maße: ca. 60 mm x 60 mm (ohne Befestigungslaschen)
- Bestückungsoption für ATtiny24/44/84A vorgesehen

Anschlüsse

Serial-IN

< 8em 2em >	
Pin	**Signal**
1	SDI
2	LATCH
3	CLK
4	GND
5	Vin

Serial-OUT

< 8em 2em >	
Pin	**Signal**
1	SDO
2	LATCH
3	CLK
4	GND
5	Vin

ISP-Anschluss (optional)

Für CON4 kann ein 2×3-Pfostenfeld oder ein 6-poliger Wannenstecker bestückt werden. Über diesen Anschluss ist dann eine Programmierung des Prozessors mit einem Programmiergerät möglich.

< 16em 6em 2em 2em 6em >			
Signal	**Pin**	**Pin**	**Signal**
MISO	1	2	VCC
SCK	3	4	MOSI
RESET	5	6	GND

Extension Port (optional)

Für CON5 kann ein 2×5-Pfostenfeld oder ein 10-poliger Wannenstecker bestückt werden. Er stellt die I/O-Anschlüsse des optionalen Mikrocontrollers zur Verfügung.

< 22em 6em 2em 2em 12em >			
Signal	**Pin**	**Pin**	**Signal**
GND	1	2	VCC (max. 50 mA)
PA0	3	4	PA1
PA2	5	6	PA3
PA4	7	8	PA5
PA6	9	10	PA7

Spannungsversorgung

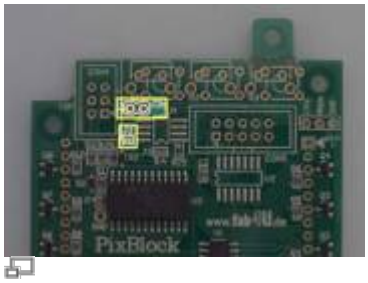
Der PixBlock ist für den Betrieb an einer stabilisierten 5 Volt Gleichspannung ausgelegt.

Ein Betrieb bis hinunter auf ca. 3,3 Volt ist möglich, wenn man die Dioden D1 und D2 überbrückt. Dazu wird Lötjumper J1 geschlossen. Außerdem muss die Diode D2 ausgelötet und durch einen kleinen Draht bzw. einen 0-Ohm-Widerstand der Bauform 0603 ersetzt werden werden.

Wichtig: Nach dem Umbau ist darauf zu achten, dass die Versorgungsspannung niemals verpolt angeschlossen wird!

Bei der Verkettung mehrerer PixBlocks ist zu beachten, dass durch Kontakt- und Leiterbahnwiderstände ein kleiner Spannungsabfall von Block zu Block entsteht. Durch den Einsatz von geregelten Stromquellen ist die LED-Helligkeit relativ unempfindlich gegenüber solchen Spannungsschwankungen. Bei langen Ketten von PixBlocks kann es aber dennoch sinnvoll sein, die Versorgungsspannung an mehreren Punkten einzuspeisen, z. B. alle 5 Pixblocks.

Mikrocontroller-Option



Reset-Jumper

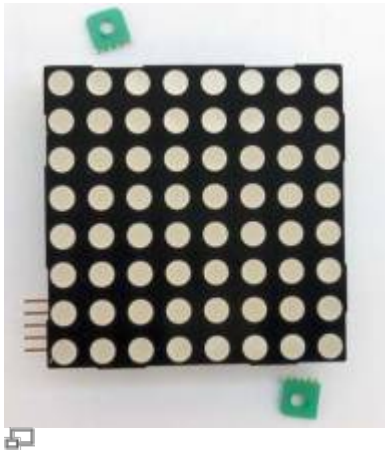
Optional kann auf der PixBlock-Platine ein Prozessor bestückt werden, der sich um die Ansteuerung kümmert. So kann der PixBlock völlig autonom arbeiten. Als Prozessoren sind die Atmel-Typen ATtiny24A, ATtiny44A oder ATtiny84A (2k, 4k oder 8k Flash-Speicher) möglich. Die Programmierung geschieht über den 6-poligen ISP-Anschluss. Hierfür wird ein passendes Programmiergerät benötigt.

Wenn der Prozessor bestückt ist, kann man ihn durch Setzen des Reset-Jumpers J2, bzw. J2b, quasi abschalten. Dann lässt sich der PixBlock wieder als rein passiver Block verwenden.

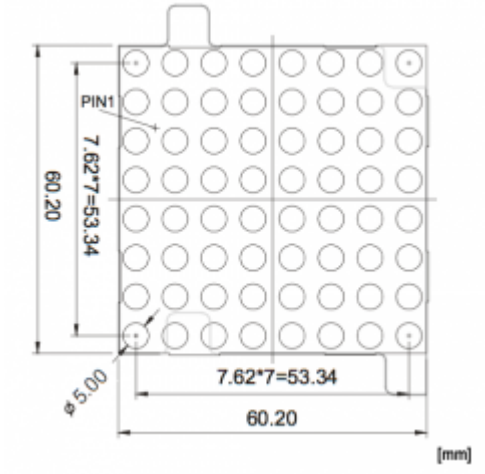
Außerdem bietet die Platine noch Platz für drei Taster (gerade oder gewinkelt) und ein serielles EEPROM (z. B. Microchip 25LC256).

Abmessungen

Der PixBlock ist etwa 60 x 60 mm groß. Die Befestigungslöcher haben einen Durchmesser von 3 mm. Für den Einbau in ein Gehäuse orientiert man sich am besten an der Maßzeichnung im Bestückungsplan.



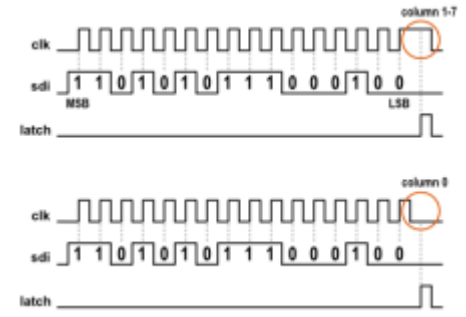
Abbrechbare Haltelaschen ermöglichen einen randlosen Einbau



Maße der LED-Matrix

Beim Anreihen mehrerer Blöcke sollte man den Abstand entsprechend dem Rastermaß der Leuchdioden wählen, d. h. die PixBlocks sollten in einem Raster von ca. 61 mm (8 x 7,62 mm) angeordnet werden.

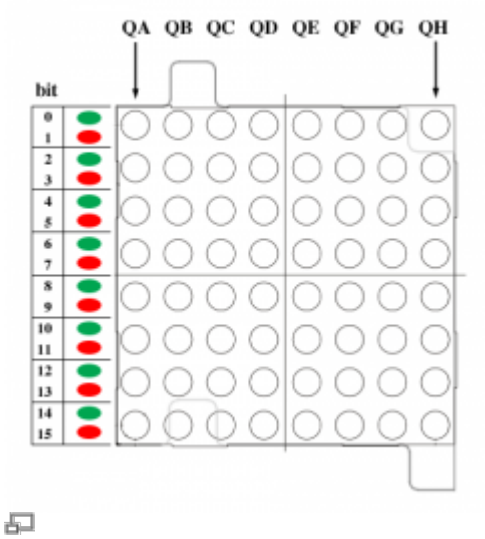
Ansteuerung



Timing-Diagramm

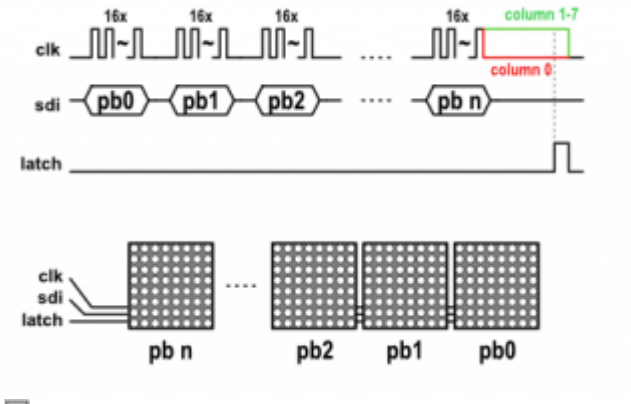
Die Ansteuerung geschieht über ein SPI-artiges Interface. Mit der steigenden Flanke des Takt-Signals (clk) wird der Zustand des Daten-Signals (sdi = serial data in) gelesen. Durch einen High-Pegel auf der Übernahme-Leitung (latch) werden die Daten in den PixBlock übernommen.

Es werden jeweils 16 Bit eingetaktet, die den Zustand der 16 Leuchtdioden einer Spalte bestimmen (1 = an, 0 = aus). Das höchstwertigste Bit (MSB) ist dabei der roten LED des untersten Pixels zugeordnet. Das niederwertigste Bit (LSB) bestimmt den Zustand der grünen LED des obersten Pixels.



Zuordnung von Bits und Leds

Die Weiterschaltung der Spalten geschieht automatisch. Mit jeder steigenden Flanke des Latch-Signals wird die nächste Spalte ausgewählt. Wichtig ist dabei der Zustand des Clock-Signals. Zum Anwählen von Spalte 0 (der Spalte ganz links) setzt man es auf Low. Für alle anderen Spalten muss es High-Pegel aufweisen. Auf diese Weise lässt sich die Ansteuerung der Spalten sehr effizient erledigen.



Verkettung mehrerer PixBlocks

Bei einer Kette von mehreren PixBlocks entsteht ein entsprechend größeres Schieberegister, z. B. 128 Bit für 8 PixBlocks. Es müssen dann jeweils pro PixBlock 16 Bit eingetaktet werden. Die Daten für den Block am Ende der Kette werden zuerst gesendet.

Ein einfaches Code-Beispiel findet man im Abschnitt "Software".

Software

Am einfachsten ist es, wenn man zur Ansteuerung des PixBlocks die Arduino-Library verwendet. Sie nimmt einem die ganzen mühseligen Arbeiten ab, so dass man sich auf seine Anwendung konzentrieren kann.

Zum grundlegenden Verständnis kann der folgende Code dienen. Er illustriert beispielhaft, wie der PixBlock in C angesteuert wird.

```
/*****************************************/
/* Simple example showing how to access  */
/* the PixBlock dot matrix in principle. */
/*****************************************/

#include <inttypes.h>
#include <avr/io.h>


/*************
 * constants *
 *************/

// hardware pins for accessing the dot matrix
#define DM_DATA_DDR       DDRB
#define DM_DATA_PORT      PORTB
#define DM_DATA_BIT       0
#define DM_CLK_DDR        DDRB
#define DM_CLK_PORT       PORTB
#define DM_CLK_BIT        1
#define DM_LATCH_DDR      DDRB
```

```
#define DM_LATCH_PORT     PORTB
#define DM_LATCH_BIT      2


/*********************
 * global variables *
 *********************/

uint16_t pixel_data[8] = {
    0xFFFF, 0xAAAA, 0x5555, 0x0000, 0xC003, 0x2008, 0x0410, 0x03C0
};   // each 16-bit value represents a column of the display


/*************
 * functions *
 *************/

void dot_matrix_shift_out(const uint16_t* screen)
// with each call shift out one column of pixel data
{
    static uint8_t    column = 0;
    uint16_t    pixel_data;

    pixel_data = screen[column];
    for (uint8_t i = 0; i < 16; i++) {
        if (pixel_data & 0x8000) { DM_DATA_PORT |= (1 << DM_DATA_BIT); }
        else { DM_DATA_PORT &= ~(1 << DM_DATA_BIT); }
        pixel_data <<= 1;

        // clock, rising egde
        DM_CLK_PORT &= ~(1 << DM_CLK_BIT);
        DM_CLK_PORT |=  (1 << DM_CLK_BIT);
    }

    // set final state of clock pin
    if (column == 0) {
        DM_CLK_PORT &= ~(1 << DM_CLK_BIT);
    }
    // pulse latch signal
    DM_LATCH_PORT |=  (1 << DM_LATCH_BIT);
    DM_LATCH_PORT &= ~(1 << DM_LATCH_BIT);

    // next column
    column++;
    column &= 0x07;         // limit column range to 0..7
}


/********
 * main *
 ********/
```

```
int main(void)
{
    // initialize i/o pins
    DM_DATA_PORT  &= ~(1 << DM_DATA_BIT);
    DM_CLK_PORT   &= ~(1 << DM_CLK_BIT);
    DM_LATCH_PORT &= ~(1 << DM_LATCH_BIT);
    DM_DATA_DDR   |=  (1 << DM_DATA_BIT);
    DM_CLK_DDR    |=  (1 << DM_CLK_BIT);
    DM_LATCH_DDR  |=  (1 << DM_LATCH_BIT);

    // repeatedly shift out pixel data
    // (usually one would do this in an interrupt routine)
    while(1)
    {
        dot_matrix_shift_out(pixel_data);
    }
}
```

Anwendungsbeispiele

Hier findest du einige Anregungen, was man mit dem PixBlock alles machen kann.

- **Bits of Time** - Die gute alte Sanduhr in moderner Ausführung als Sanduhr 2.0.

Download

Datei	Lizenz
Schaltplan	CC-BY-SA
Bestückungsplan	CC-BY-SA
PixBlock Library	siehe license.md
Bauanleitung	CC-BY-SA

CC-BY-SA = Creative-Commons-Lizenz, Namensnennung, Weitergabe unter gleichen Bedingungen
CC0 = Creative-Commons-Lizenz, ohne Einschränkungen

PixBlock, Elektronik, Bausatz, Led, Dot-Matrix, Anzeige, Display