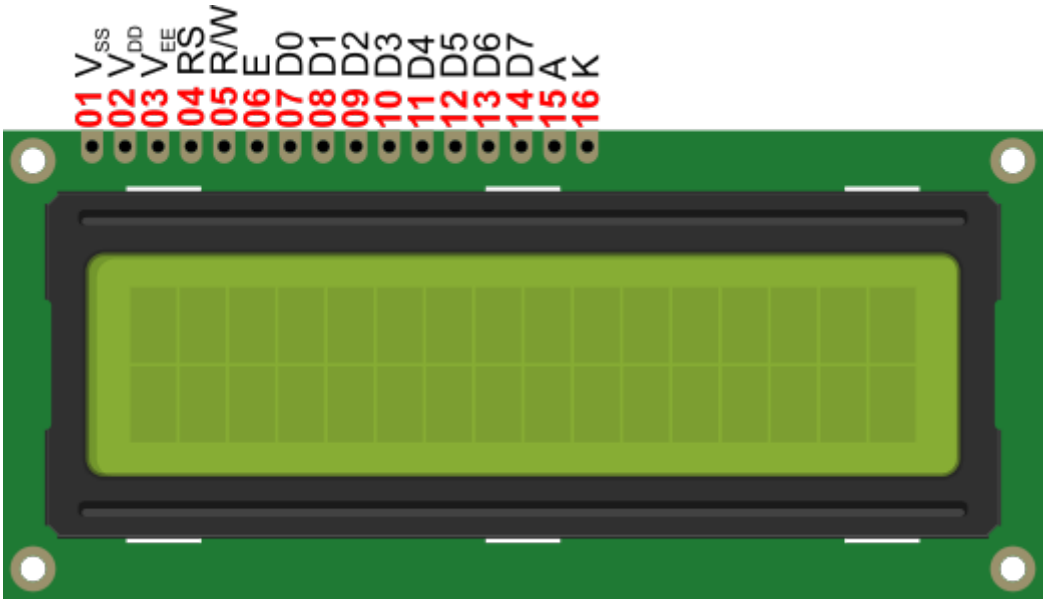
Display 16 Zeichen

Beschreibung

Ein Stellen mit zwei Zeilen Display.

Anschluss



Es werden folgende PINS angeschlossen:

- **Pin 1 (V_{SS})** und **Pin 2 (V_{DD})** dienen der Stromversorgung des Displays und der Ansteuerungselektronik. Pin 1 ist dabei auf Masse zu legen, auf Pin 2 sind +5 V Versorgungsspannung zuzuführen.
- **Pin 3 (V_{EE})** ist ein analoger Eingang und dient der Kontrastregelung des Displays. Der Wert muss zwischen 0 V und +5 V liegen.
- **Pin 4 (RS)** ist ein digitaler Eingang und bestimmt, ob die zum Display übermittelten Datenbits als Befehl (LOW) oder Zeichendaten (HIGH) interpretiert werden sollen.
- **Pin 5 (R/W)** ist ein digitaler Eingang, der entscheidet, ob Daten auf dem Display geschrieben (LOW) oder vom Display eingelesen (HIGH) werden sollen. Es ist also tatsächlich möglich, den Inhalt des Displays wieder mit dem Arduino einzulesen. In der Praxis ist das aber eigentlich nie erforderlich. Daher legt man diesen Pin einfach dauerhaft auf Masse (LOW).
- **Pin 6 (E)** ist ein digitaler Eingang, der auf HIGH geschaltet werden muss, damit das Display die an den Datenpins anliegenden Bits ausliest.
- **Pin 7 – Pin 14 (D0 – D7)** sind die 8 Bits des bidirektionalen, paralellen Datenbusses. Da man ungern ganze 8 Ports des Arduinos nur für die Datenübertragung zum Display verbrauchen möchte, nutzt man die Fähigkeit der Ansteuerungselektronik, in den 4-Bit-

Modus zu schalten. In diesem Fall werden nur die hinteren Pins 11 – 14 (D4 – D7) mit dem Arduino verbunden und die 8 Bit in zwei Schritten (jeweils 4 Bit) nacheinander übertragen. Die Pins 7 – 10 lässt man einfach offen.
- **Pin 15 (A)** und **Pin 16 (K)** existieren nur an LCD mit eingebauter Hintergrundbeleuchtung und dienen der Stromversorgung selbiger. An Pin 15 (Anode) kommt die Versorgungsspannung, Pin 16 (Kathode) wird auf Masse gelegt. Je nach LCD muss hier entweder ein Vorwiderstand für die im LCD verbaute LED vorgeschaltet werden oder aber der entsprechende Widerstand befindet sich bereits im LCD. Wenn man sich unsicher ist und kein Hinweis darauf beim LCD zu finden ist, kann man vorsichtshalber einen 220 Ω-Widerstand einbauen.

LCD-Ansteuerung mit analoger Kontrastregelung (Poti)

- Universal-LCD mit Parallelbus (14 oder 16 Pins)
- (Widerstand 220 Ω)
- Trimmpotentiometer 10 kΩ
- Jumperkabel (18×)
- Beim Poti ist Anschluss 1 immer Masse und Anschluss 2 V+ und die Mitte das eigentliche zu Regelnde Gerät LED etc.

Steckbrettansicht

Die relevaten Pins des LCD werden mit dem Arduino verbunden. In die Spannungszuführung der Hintergrundbeleuchtung (so denn überhaupt vorhanden) wurde vorsichtshalber der oben erwähnte Vorwiderstand eingesetzt. Der analoge Eingang der Kontrastregelung (V_{EE}) wird mit dem Schleifkontakt eines Trimmpotentiometers verbunden, welches auf der einen Seite mit +5 V, auf der anderen Seite mit Masse (0 V) verbunden wird. Hierüber lässt sich der Kontrast manuell regeln.

Zur Ansteuerung des LCD wird die LiquidCrystal-Bibliothek von Adafruit genutzt. Der Beispielcode gibt einen Standardtext aus und zählt anschließend die Sekunden seit dem Start des Programms hoch.

LCD-Ansteuerung mit PWM-Kontrastregelung

- Universal-LCD mit Parallelbus (14 oder 16 Pins)
- (Widerstand 220 Ω)
- Jumperkabel (14×)

Steckbrettansicht

Im Normalfall stellt man den Kontrast des LCD einmalig ein und belässt ihn dann in dieser Einstellung. Damit ist das Trimmpotentiometer eigentlich überflüssig und nimmt nur Platz auf dem Breadboard weg. Einen passenden Widerstand (mit festem Wert) zu finden, kann sich aber unter

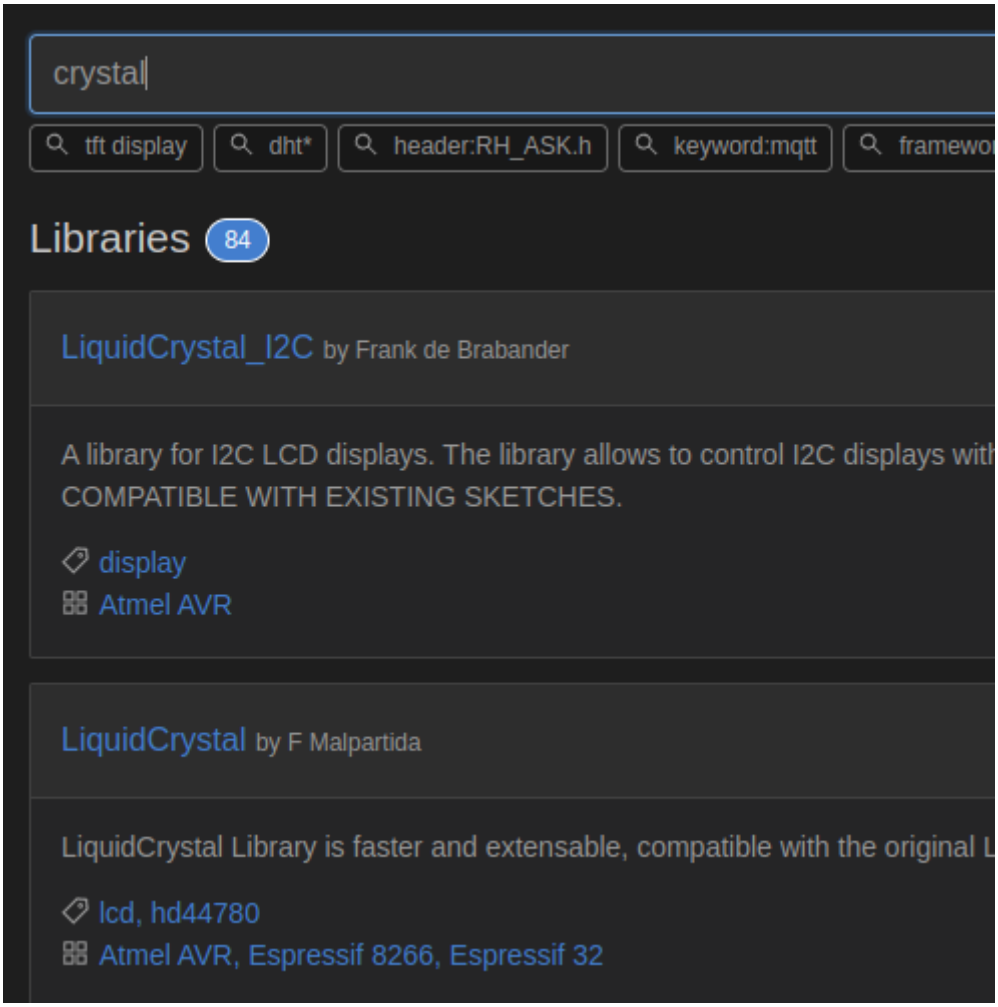
Umständen als schwierig erweisen. Eine Alternative stellt die Kontrastregelung über einen PWM-Ausgang des Arduinos dar. Dazu wird auch der Pin 3 (V_{EE}) an den Arduino angeschlossen und das Trimmpotentiometer kann entfallen. Dafür verliert man natürlich wiederum einen digitalen Ausgang. Man muss von Schaltung zu Schaltung abwägen, was einem lieber ist.

Im Gegensatz zum obigen Beispiel wird bei der Initialisierung mittels der Funktion analogWrite() per PWM der Kontrast des LCD auf einen festen Wert eingestellt. Der optimale Wert muss von Ihnen einmalig auf Ihr LCD angepasst werden.

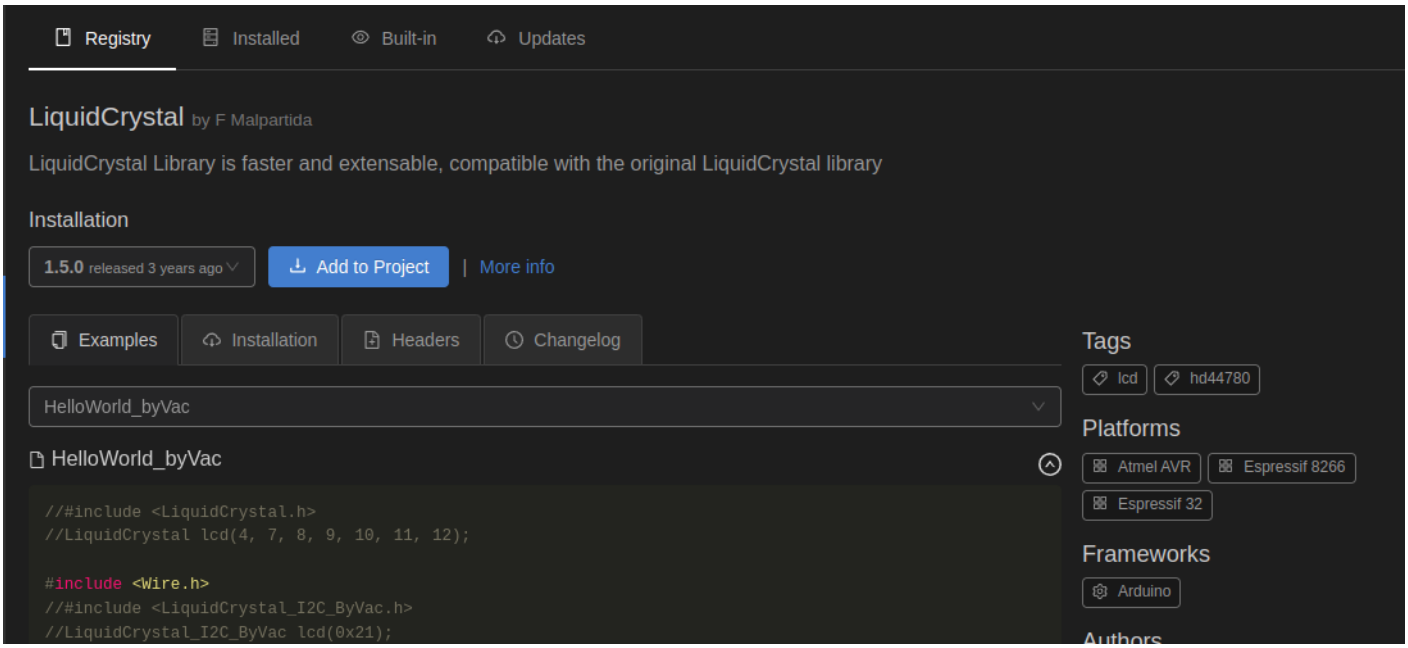
Einbindung 16 Zeichen Display

Bibliotheken installieren

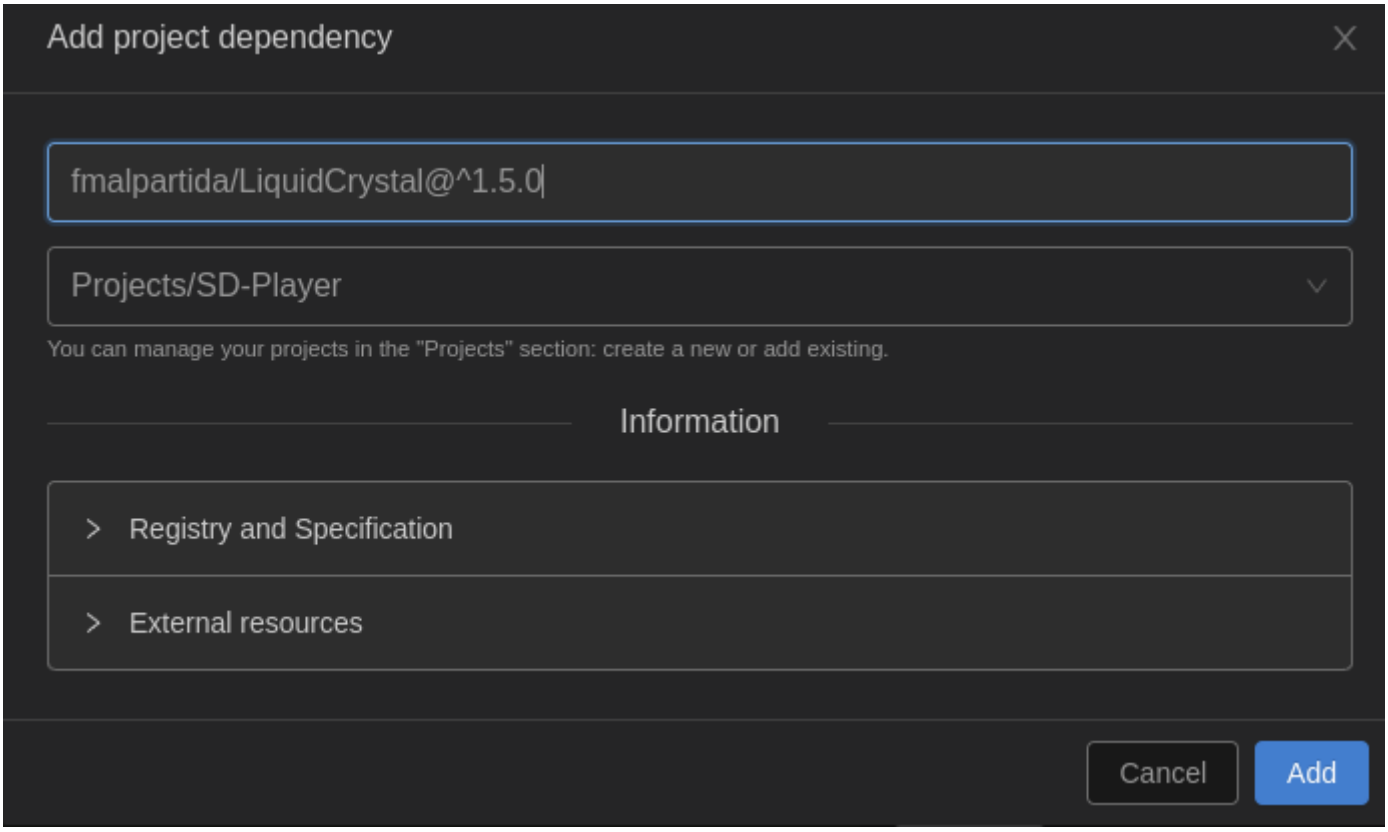
Wir benötigen dazu die LiquidCrystal Library (Diese im PIO-Home Library herunterladen)
Dazu den Suchbegriff crystal eingeben



Dann auf Add to Project klicken



Und nun das Project auswählen



Code

Code Beispiel Direktanschluss 4 Datenpins ohne IC2 BUS:

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

```
#include <SoftwareSerial.h>

#define PIN_LCD_VEE_VO 6                    // Pin für LCD-Pin Vee/Vo (Kontrastwert)
#define PIN_LCD_RS 13                       // Pin für LCD-Pin RS  (Register Select)
#define PIN_LCD_E 12                        // Pin für LCD-Pin E   (Enable)
#define PIN_LCD_D4 2                        // Pin für LCD-Pin D4  (Datenbit 4)
#define PIN_LCD_D5 3                        // Pin für LCD-Pin D5  (Datenbit 5)
#define PIN_LCD_D6 4                        // Pin für LCD-Pin D6  (Datenbit 6)
#define PIN_LCD_D7 5                        // Pin für LCD-Pin D7  (Datenbit 7)

#define LCD_CONTRAST 10                     // Kontrastwert (muss experimentell an das LCD angepasst werden) 0
ganz Dunkel 100 Total Hell, auch nix mehr zu erkennen)
#define LCD_ROWS 2                          // Anzahl der Zeilen des Displays.
#define LCD_COLS 16                         // Anzahl der Spalten des Displays.

LiquidCrystal lcd(PIN_LCD_RS, PIN_LCD_E, PIN_LCD_D4, PIN_LCD_D5, PIN_LCD_D6, PIN_LCD_D7);
int aSeconds = 0; //Als Zähler für die Sekunden, wird fürs Display nichtebenötigt, aber wie bauen hier im Beispiel
einen Zähler
void setup()
{
 analogWrite(PIN_LCD_VEE_VO, LCD_CONTRAST);  // Regele den Kontrast des Display per PWM auf den Wert
LCD_CONTRAST. Wenn ein Poti benützt wird überflüssig
   lcd.begin(LCD_COLS, LCD_ROWS);           // Die Größe des Displays festlegen und das Display intialisieren.
   lcd.setCursor(0, 0);                     // Springe mit dem Cursor in der 1. Zeile an Position 1.
lcd.setCursor(Position,Zeile)
   lcd.print("Sekunden seit");              // Schreibe ab dort den Text "Sekunden seit".
   lcd.setCursor(0, 1);                     // Springe mit dem Cursor in der 2. Zeile an Position 1.
lcd.setCursor(Position,Zeile)
   lcd.print("Start:");                     // Schreibe ab dort den Text "Start:".
}

void loop()
{
 lcd.setCursor(7, 1);                     // Springe mit dem Cursor in der 2. Zeile an Position 8.
   lcd.print(aSeconds);                     // Schreibe ab dort den aktuellen Wert der Variable aSeconds.
   aSeconds++;                              // Erhöhe den Wert der Variablen aSeconds um 1.

   delay(1000);                             // Warte eine Sekunde.
}
```

Quelle

https://rotering-net.de/tut/arduino/lcd-ansteuern.html

Version #12
Erstellt: 24 Dezember 2022 21:08:10 von Admin
Zuletzt aktualisiert: 29 Dezember 2022 02:24:58 von Admin