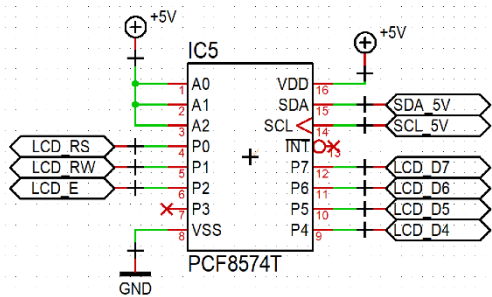


Anschluss des LC-Display am I²C-Bus

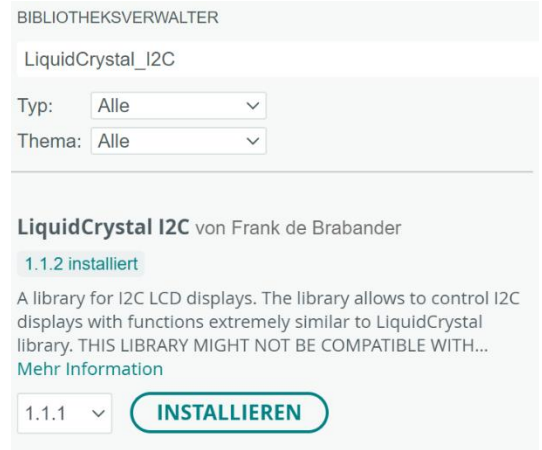


Das LC-Display ist mit einem PCF8574 an den I²C-Bus angeschlossen.

Die Datenübertragung zum Display erfolgt im 4-Bit Modus. Das bedeutet, dass immer jeweils 2 Byte pro Zeichen über den I²C-Bus übertragen werden.

Die Standard LiquidCrystal-Bibliothek von Arduino funktioniert hier leider nicht, da sie nicht mit Bus-Anschluss arbeitet. Daher muss eine Zusatzbibliothek installiert werden. Wir verwenden die **LiquidCrystal_I2C**-Bibliothek

von Frank de Brabander. Sie finden die Bibliothek leicht über den Bibliotheksverwalter:



Die LiquidCrystal_I2C-Bibliothek ist weitgehend kompatibel zur original Arduino LiquidCrystal. Die wichtigsten Funktionen (Methoden)¹ zur Textausgabe sind:

Konstruktor:

```
LiquidCrystal_I2C lcd( i2cADDR, Spalten, Zeilen);
// i2cADDR: 0x27
// Spalten: 16
// Zeilen: 2
```

```
Wire.begin( SDApin, SCLpin); // I2C-Bus initialisieren
// SDApin: GPIO 21          Ist nicht teil von LiquidCrystal_I2C,
// SCLpin: GPIO 22          wird aber benötigt
```

```
lcd.init(); // LCD initialisieren
lcd.clear(); // LCD-Anzeige löschen
lcd.setCursor( spalte, zeile ); // s: 0..15 z: 0..1
lcd.print( *Text ); lcd.print( wert, BASE ); // BASE: DEC, HEX, BIN, OKT (optional)
lcd.println(); // dito. mit CR+LF
lcd.backlight(); lcd.noBacklight(); // Backlight Jumper auf prg/OFF (exp.)
```

Arbeitsaufträge

1. Installieren Sie die LCD-Bibliothek.
2. Testen Sie die Displayausgabe mit dem abgebildeten Beispielprogramm.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
    Wire.begin(21,22);
    lcd.init();
    lcd.clear();

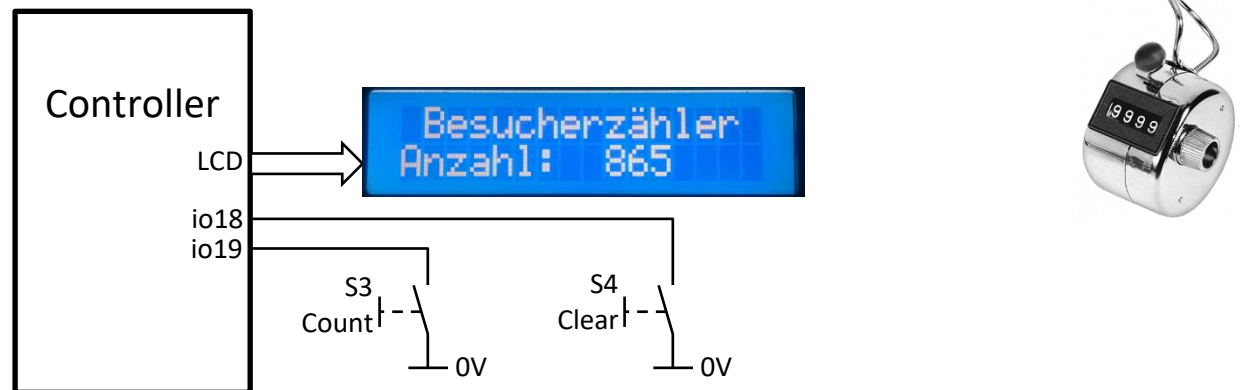
    lcd.setCursor(0,0);
    lcd.print("Hallo Welt");
}

void loop() {}
```

¹ <https://www.arduino.cc/reference/en/libraries/liquidcrystal/>

Projekt: Digitaler Besucherzähler für einen Messestand

Zur mengenmäßigen Erfassung von Besuchern auf dem Messestand Ihrer Firma wurde seither ein Handzähler verwendet. Dieser hat ein mechanisches Zählwerk, eine Zähltaste (Count) und einen Rücksteller (Clear). Sie bekommen die Aufgabe, einen solchen Zähler mit dem Mikrocontroller und dem LC-Display zur Anzeige des Zählwertes zu realisieren.



Zunächst wird die Clear-Taste weggelassen.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
#define PRESS LOW
const int S3_COUNT = 19;

void setup()
{
  Wire.begin(21,22);
  lcd.init();
  lcd.clear();

  lcd.setCursor(0,0);
  lcd.print("Besucherz\u00e4hler");
  lcd.setCursor(0,1);
  lcd.print("Anzahl:");

  pinMode(S3_COUNT, INPUT_PULLUP);
}
```

```
uint16_t count = 0;

void loop()
{
  char buf[6];
  sprintf(buf, "%5u", count);
  lcd.setCursor(8,1);
  lcd.print(buf);

  if(digitalRead(S3_COUNT) == PRESS)
    count++;
}
```

Hinweis: \xe1 ist der Displaycode für den Umlaut ä. Weitere Zeichencodes:

ö: \xef	ß: \xe2	σ: \xe5
ü: \xf5	μ: \xe4	α: \xe0
°: \xdf	€: \xd3	ε: \xe3


Arbeitsauftrag

- Erstellen Sie das abgebildete Programm und testen Sie es auf der Mikrocontroller-Hardware.
- Machen Sie sich die Arbeitsweise der Textausgabe mit `sprintf()`² klar.

```
sprintf(buf, "format", par1,par2,...);
buf:      Name des Zeichenpuffers (array)
format:   Formatstring mit Platzhaltern. Z.B. "%5.2f\ndfC" → 23.54°C
par1, par2, ...: Parameter der/die in die Platzhalter eingesetzt werden. Reihenfolge!
```

- Warum funktioniert das Programm nicht so wie gewünscht?

² <https://www.ibm.com/docs/en/zos/3.1.0?topic=programs-sprintf-format-write-data>

 Friedrich-Ebert-Schule Esslingen FES	IT: Hardwarenahes Programmieren	Name: Rahm Datum: 12.11.2024 1_6_LC-Matrix_Display.docx
	Programmierung: LC-Matrixdisplay	1.6.3

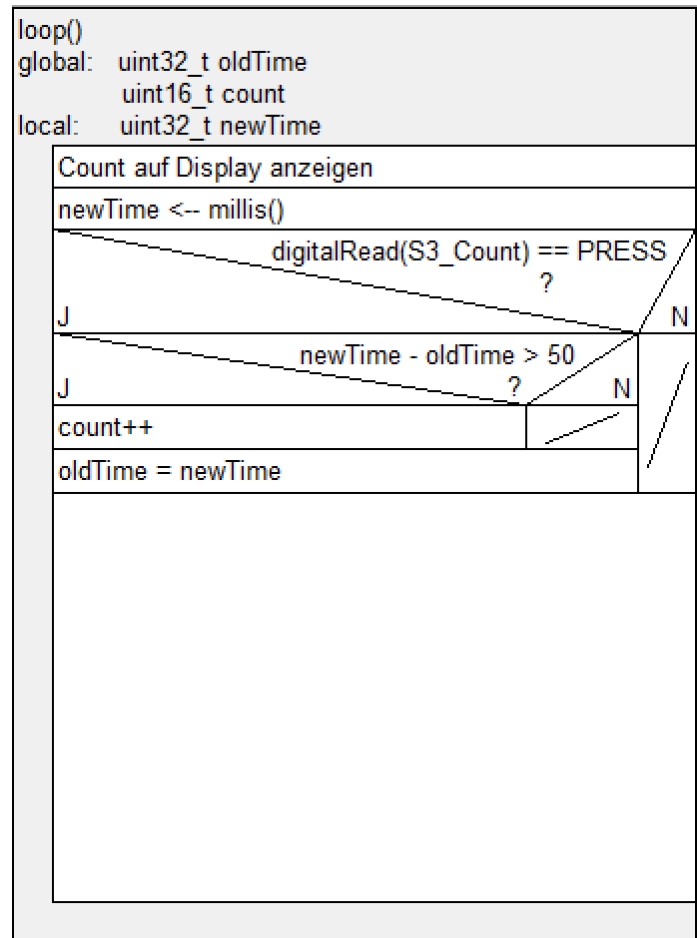
Es darf nur einmal pro Tastendruck gezählt werden. Hier hilft uns wieder unser Entprellalgorithmus

Arbeitsauftrag

1. Erweitern Sie das Programm entsprechend dem abgebildeten Struktogramm.
2. Ergänzen Sie nun die Clear-Funktion. Dazu benötigen Sie keine Entprellung mehr, da es kein Problem ist, wenn count mehrfach auf 0 gesetzt wird.
3. Tragen Sie Ihre Lösung in das freie Anweisungsfeld im Struktogramm ein.
4. Dokumentieren Sie Ihre Ergebnisse.

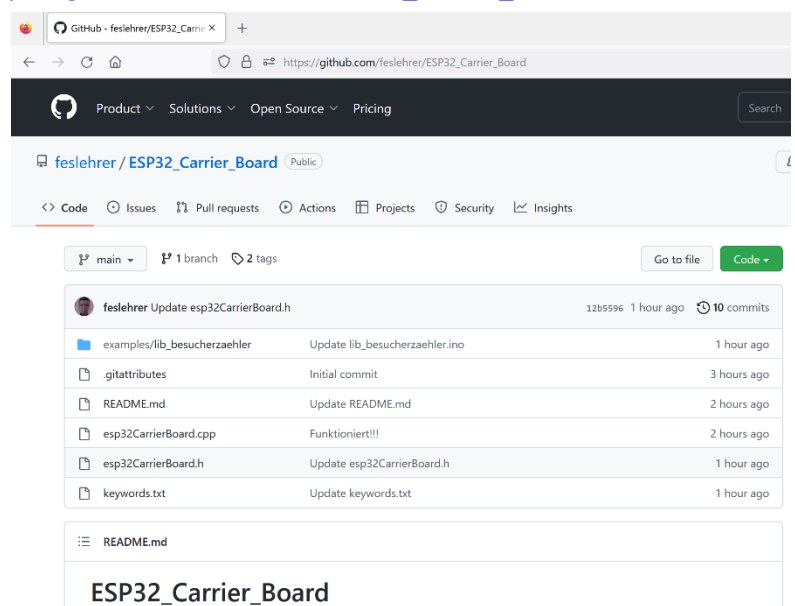
Sollen mehrere Taster eine Entprellung benötigen, kann der gleiche Code dupliziert werden. Es werden dann nur für jeden Taster eigene Variablen (oldTime, newTime) benötigt.

Eine einfache Lösung ist die Installation einer Bibliothek, die speziell für das ESP32-Carrier-Board gemachte Bibliothek, die von GitHub heruntergeladen werden und installiert werden kann.



Vorgehensweise


1. Öffnen Sie das GitHub-Repository: https://github.com/feslehrer/ESP32_Carrier_Board



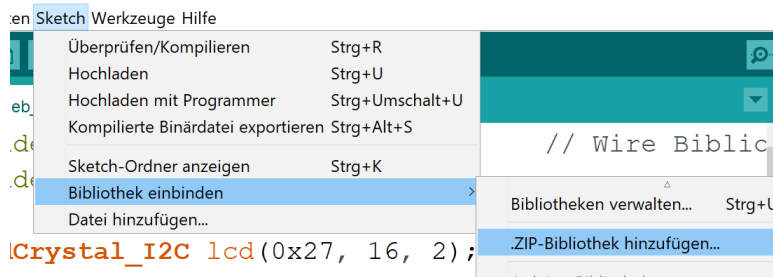
2. Gehen Sie auf „Code“ (Grüner Button) und betätigen Sie:


Download ZIP

3. Nach dem Download befindet sich die zip-Datei im Download-Ordner.

 Friedrich-Ebert-Schule Esslingen FES	IT: Hardwarenahes Programmieren	Name: Rahm Datum: 12.11.2024 1_6_IC-Matrix_Display.docx
	Programmierung: LC-Matrixdisplay	1.6.4

4. Öffnen Sie nun in der Arduino-IDE den Menüpunkt Sketch→ Bibliothek einbinden → .Zip-Bibliothek hinzufügen...



5. Gehen Sie zum Download-Ordner und wählen Sie die Zip-Datei  ESP32_Carrier_Board-1.1.zip aus.
6. Über Sketch→Bibliothek einbinden→ESP32_Carrier_Board-main können Sie die Bibliothek einbinden. Im Code erscheint die folgende Anweisung:
- ```
#include <esp32CarrierBoard.h>
```

7. Das fertige Beispiel zum Besucherzähler finden Sie nun im Menü:  
Datei→Beispiele→ESP32\_Carrier\_Board-main→ libBesucherzaehler

Rufen Sie das Beispiel auf. Es bietet zusätzlich die Möglichkeit den Besucherzähler wieder herunterzählen. Probieren Sie es aus.

Hier der komplette Code des Besucherzählers:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <esp32CarrierBoard.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int CLEAR = S4; // GPIO 18
const int COUNTUP = S3; // GPIO 19
const int COUNTDOWN = S2; // GPIO 23
bool toggleStateUP;
bool toggleStateDOWN;

void setup()
{
 Wire.begin(21,22);
 lcd.init();
 lcd.clear();

 lcd.setCursor(0,0);
 lcd.print("Besucherz\xe1hler");
 lcd.setCursor(0,1);
 lcd.print("Anzahl:");

 pinMode(COUNTUP,INPUT_PULLUP);
 pinMode(COUNTDOWN,INPUT_PULLUP);
 pinMode(CLEAR,INPUT_PULLUP);
}
```

```
uint16_t count = 0;

void loop()
{
 char buf[6];
 sprintf(buf, "%5u", count);
 lcd.setCursor(8,1);
 lcd.print(buf);

 if(pinToggle(COUNTUP, &toggleStateUP) == PRESS)
 {
 count++;
 }

 if(pinToggle(COUNTDOWN, &toggleStateDOWN) == PRESS)
 {
 if(count!=0) count--;
 }

 if(digitalRead(CLEAR) == PRESS)
 {
 count=0;
 }
}
```

### Zur Erklärung:

Die Funktion **pinToggle()** liest den Taster und liefert ein entprelltes Signal als Rückgabewert (Typ bool). Die Statusvariable (toggleStateX) muss als Zeiger (call-by-ref) übergeben werden. Daher wird der Variablen der Referenzier-Operator (&) vorangestellt. Mit der Status-Variablen kann auch die Toggle-Funktion des Tasters realisiert werden, die in diesem Beispiel aber nicht benötigt wird.