Falcon BMS to Arduino Interface Tool (BMSAIT)

Beispielprogramm zur Abbildung des Whiskeykompass mit einem OLED Display

Autor	Robin „Hummer“ Bruns
Dokumentversion	1.0
Softwareversion	1.3.17
BMS Version	4.37u3
Datum	06.01.2024

1. Überblick

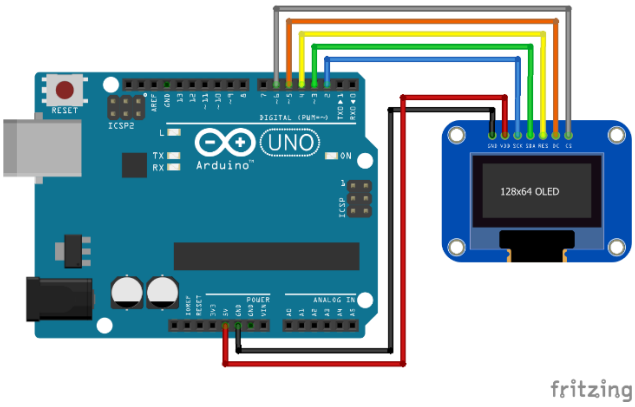
Das vorliegende Beispielprogramm stellt eine Möglichkeit dar, um den Whiskey Kompass mit einem Arduino anzusteuern.

Um dieses Beispiel nachzubauen benötigt ihr:

- Ein Arduino-Board oder anderes Entwicklerboard (z.B. ESP32)
  Um eine möglichst flüssige Kompassanzeige zu erreichen, empfehle ich ein leistungsfähiges Entwicklungsboard zu nutzen (Arduino Mega oder Due; ESP8266 oder ESP32). Bei Nutzung eines Arduino UNO oder NANO wird die Anzeige Störungen aufweisen.
- Ein OLED Display (SSD1306 mit 128x32 oder 128x64 Pixel)
- Verbindungskabel

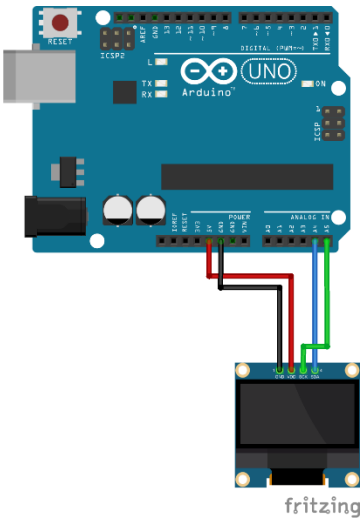
2. Verkabelung

2.1. Variante 1 (SPI Verbindung)



Arduino	OLED
GND	GND
3.3V oder 5V	Vcc
2	D0/SCK
3	D1/SDA
4	RES
5	DC
6	CS

2.2. Variante 2 (I²C Verbindung)



Arduino	OLED
GND	GND
3.3V oder 5V	Vcc
A5	SCL
A4	SDA

3. Programmierung des Arduino

Falls die Arduino IDE noch nicht installiert ist, lest bitte das Kapitel 4.1.4 der BMSAIT Dokumentation. Stellt sicher, dass die U8G2 Bibliothek installiert ist.

Prüft anschließend, welche Hardware ihr nutzen wollt. OLED lassen sich über unterschiedliche Verbindungsarten mit dem Arduino verbinden. Üblich sind I^2C oder SPI. Beim SPI gibt es wiederum mehrere Untervarianten. In diesem Handout beschreibe ich zwei Beispiele, wie die Verbindung abgebildet werden kann.
Variante 1 – Anbindung eines SSD1306 OLED mit 128x64 Pixeln über eine 4-aderige SPI-Verbindung
Variante 2 – Anbindung eines SSD1306 OLED mit 128x32 Pixeln über eine I^2C Verbindung.

Andere Verbindungsarten lassen sich herstellen, indem die entsprechenden Konstruktoren für die U8G2 Bibliothek eingebunden werden. Eine Liste aller Konstruktoren findet ihr unter folgendem Link. Sucht dort zuerst nach dem Chip eures OLED Displays (z.B. SSD1306) anschließend nach der passenden Pixelzahl und zuletzt die beabsichtigte Verbindungsart (z.B. I2C, SPI):
https://github.com/olikraus/u8g2/wiki/u8g2setupcpp#constructor-name

Ruft nun die .ino aus dem jeweiligen Variantenordner (\Arduino Sketch\BMSAIT_OLED_Compass1\ oder BMSAIT_OLED_Compass2\ ) mit einem Doppelklick auf. Der Sketch wird in der Arduino IDE geladen. Wenn ihr die Verkabelung gem. Kapitel 2 durchgeführt habt, sind hier erst einmal keine Anpassungen erforderlich.
Bei OLEDs ist es wichtig den richtigen „Konstruktur“ für die Ansteuerung des OLEDs zu wählen. Das Vorgehen habe ich in der Dokumentation von BMSAIT beschrieben.

Beim Wechsel des OLED-Konstruktors ist dieser zweimal im Sketch anzupassen. Einmal mit der Steuerungsvariante F (Full) für leistungsfähige Entwicklungsboards und einmal mit der Steuerungsvariante 2 (OLED Anzeige wird in zwei nacheinanderfolgenden Schritten zusammengesetzt) für Boards mit wenig Arbeitsspeicher. Beim Wechsel eines Konstruktors ist auch zu prüfen, ob sich die Angabe an Parametern ändert. Wenn ja, ist das auch im Arduino Code anzupassen, damit die richtige Zahl an PINs angegeben wird.

```
9   #if defined(DUE) || defined(DUE_NATIVE) || defined(MEGA) || defined(ESP)
10    //arduino board with enough memory will use the unbuffered mode
11    U8G2_SSD1306_128X64_NONAME_F_4W_SW_SPI displayOLED(U8G2_R2, /*d0*/2,/*d1*/ 3,/*CS*/ 6,/*DC*/ 5 ,/*Res*/4);
12  #else
13    //arduino board with low memory will have to use the buffered mode
14    U8G2_SSD1306_128X64_NONAME_2_4W_SW_SPI displayOLED(U8G2_R2, /*d0*/2,/*d1*/ 3,/*CS*/ 6,/*DC*/ 5 ,/*Res*/4);
15  #endif
16
```

Nachdem der Konstruktor im Sketch geprüft wurde und ihr das richtige Arduino-Board ausgewählt habt, ladet ihr den Sketch auf den Arduino hoch.
Nach dem Hochladen sollte das Display kurz den Text „BMSAIT“ anzeigen. Nach 2 Sekunden wird eine Kompassrose angezeigt. Diese sollte „360“ zentriert anzeigen. Das Display schaltet nach 10 Sekunden in einen Ruhemodus. Das Display aktiviert sich wieder, wenn eine aktive Verbindung zur BMSAIT Windows App besteht.

Sollte die Grafik der Kompassrose nicht richtig angezeigt werden (z.B. ein verschobenes Heading), kann dies über die Optionen *#define OFFSETX xxx* und *#define OFFSETY xxx* im Sketch angepasst werden. Der Standardwert für X ist 335 und für Y ist es die 0.

```
///Layout settings
                          //                          Rotation: U8G2_R0  U8G2_R1  U8G2_R2  U8G2_R3
                          //                                    (0°)     (90°CW)   (180°)    (270°CW)
  #define OFFSETX 335     //increase this to move graphic     left     down      right      up
                          //decrease this to move graphic     right     up       left       down
  #define OFFSETY 0       //increase this to move graphic      up      right     down       left
                          //decrease this to move graphic     down     left       up        right

```

Nach einer Anpassung der Optionen muss der Sketch erneut auf den Arduino hochgeladen werden.

Hinweis: Der geringe Arbeitsspeicher eines Arduino UNO oder NANO kommt mit diesem Beispielprogramm an seine Grenzen. Um dies zu umgehen, erfolgt die Aktualisierung der OLED Anzeige in mehreren Schritten. Das führt aber zu Bildverzerrungen.
Ihr könnt versuchen, ob euer Arduino genug Ressourcen hat, um die Bildaktualisierung doch in einem Schritt durchzuführen. Aktiviert dazu in dem Modul „UserConfig“ im Bereich BOARD SELECTION nicht den Arduino UNO, NANO, MICRO oder LEONARDO aus (auch wenn ihr diesen benutzt), sondern den Eintrag ESP.

```
//BOARD SELECTION

  //#define UNO         //uncomment this if this sketch will be loaded on an UNO
  //#define NANO        //uncomment this if this sketch will be loaded on an NANO
  //#define MICRO       //uncomment this if this sketch will be loaded on an MICRO
  //#define LEONARDO    //uncomment this if this sketch will be loaded on an LEONARDO
  //#define MEGA        //uncomment this if this sketch will be loaded on an MEGA
  //#define DUE         //uncomment this if this sketch will be loaded on an DUE (connected via programming port)
  //#define DUE_NATIVE  //uncomment this if this sketch will be loaded on an DUE (connected via native port)
  #define ESP         //uncomment this if this sketch will be loaded on an ESP32 or ESP8266

```

4. Einstellung des Windows-Programms

Installiert und startet BMSAIT und stellt sicher, dass die Basiseinstellungen richtig vorgenommen wurden. Wichtig ist insbesondere, dass der Verweis auf die Variablendefinition (BMAIT-Variablen.csv) hergestellt wird.

Ladet anschließend die beiliegende Konfiguration (BMSAIT_CompassOLED.ini). BMSAIT sollte nun die geladene Definition anzeigen (ein COM-Port und 2 Variablen).

Macht einen Rechtsklick auf den COM-Port und bearbeitet diesen. Wählt den COM-Port aus, an dem euer Arduino angeschlossen ist. Wenn ihr nicht sicher seid, welcher COM-Port dies ist, dann wählt entweder die SCAN Funktion und schaut, auf welchem COM-Port der Arduino eine Antwort sendet oder ihr schaut in dem Windows-Gerätemanager nach.

Ich empfehle die Änderungen nun zu sichern („Speichern unter“ und Auswahl einer neuen Datei).

Startet nun die Verbindung zum Arduino.

5. Einstellungen in Falcon BMS

Keine Einstellungen erforderlich.

6. Ergebnis

Startet Falcon BMS in der Instant Action. Mit dem Einstieg in die 3D Welt wird das aktuelle Heading des Flugzeugs durch eine Kompassrose auf dem OLED Display angezeigt.