Falcon BMS to Arduino Interface Tool (BMSAIT)

Example program for driving a simulated whiskey compass with an OLED display

Author	Robin "Hummer" Bruns
Document version	1.0
Version	1.3.17
BMS Version	4.37u3
Date	06.01.2024

1. Overview

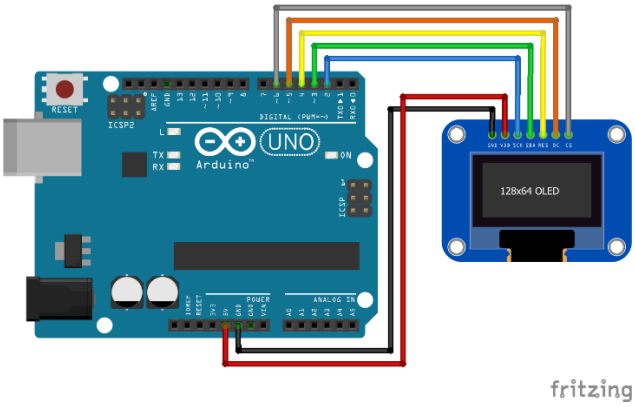
The present example program is a way to simulate a whiskey compass with an Arduino.

To recreate this example you need:

- An Arduino board or other developer board (e.g. ESP32)
  To achieve the smoothest possible compass display, I recommend using a powerful development board (Arduino Mega or Due; ESP8266 or ESP32). When using an Arduino UNO or NANO, the display will show interference.
- An OLED display (SSD1306 with 128x32 or 128x64 pixels)
- Connecting cable

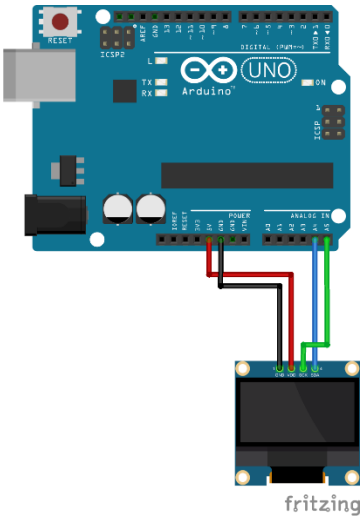
2. Cabling

2.1. Variant 1 (SPI connection)



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

2.2. Variant 2 (I2C connection)



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

3. Programming the Arduino

If the Arduino IDE is not yet installed, please refer to chapter 4.1.4 of the BMSAIT documentation. Ensures that the U8G2 library is installed.

Then check which hardware you want to use. OLEDs can be connected to the Arduino via different connection types. I2C or SPI are common. In SPI, there are several subvariants. In this handout, I describe two examples of how the connection can be mapped.
Variant 1 – Connection of a SSD1306 OLED with 128x64 pixels via a 4-wire SPI connection
Variant 2 – Connection of a SSD1306 OLED with 128x32 pixels via an I2C connection.

Other types of connections can be established by including the corresponding constructors for the U8G2 library. A list of all constructors can be found under the following link. First look for the chip of your OLED display (e.g. SSD1306), then for the appropriate number of pixels and finally for the intended connection type (e.g. I2C, SPI):
https://github.com/olikraus/u8g2/wiki/u8g2setupcpp#constructor-name

Now call up the .ino from the respective variant folder (\Arduino Sketch\BMSAIT_OLED_Compass1\ or BMSAIT_OLED_Compass2\ ) with a double-click. The sketch is loaded in the Arduino IDE. If you have done the wiring according to Chapter 2, no adjustments are necessary here.
With OLEDs, it is important to choose the right "structure" for controlling the OLED. I have described the procedure in the documentation of BMSAIT.

When changing the OLED constructor, it has to be adjusted twice in the sketch. Once with the control variant F (Full) for high-performance development boards and once with the control variant 2 (OLED display is assembled in two consecutive steps) for boards with little RAM. When changing a constructor, it is also necessary to check whether the specification of parameters changes. If so, this must also be adjusted in the Arduino code so that the correct number of PINs is specified.

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

```

After the constructor has been checked in the sketch and you have selected the correct Arduino board, upload the sketch to the Arduino.
After uploading, the display should briefly show the text "BMSAIT". After 2 seconds, a compass rose will appear. This should show "360" centered. The display switches to a sleep mode after 10 seconds. The display reactivates when there is an active connection to the BMSAIT Windows app.

If the graphic of the compass rose is not displayed correctly (e.g. a shifted heading), this can be adjusted via the options *#define OFFSETX xxx* and *#define OFFSETY xxx* in the sketch. The default value for X is 335 and for Y it is 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

```

After adjusting the options, the sketch must be uploaded to the Arduino again.

Note: The low RAM of an Arduino UNO or NANO reaches its limits with this example program. To get around this, the OLED display is updated in several steps. However, this leads to image distortions. You can try to see if your Arduino has enough resources to perform the image update in one step. To do this, do not activate the Arduino UNO, NANO, MICRO or LEONARDO in the "UserConfig" module in the BOARD SELECTION area (even if you are using it), but the ESP entry.

```
//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. Setting up the Windows app

Installs and starts BMSAIT and makes sure that the basic settings have been made correctly. It is particularly important that the reference to the variable definition (BMAIT-Variablen.csv) is made.

Then load the enclosed configuration (BMSAIT_CompassOLED.ini). BMSAIT should now display the loaded definition (one COM port and 2 variables).

Right-click on the COM port and edit it. Select the COM port to which your Arduino is connected. If you are not sure which COM port this is, then either select the SCAN function and see on which COM port the Arduino sends a response or you can look in the Windows Device Manager.

I recommend saving the changes now ("Save as" and selecting a new file).

Now start the connection to the Arduino.

5. Settings in Falcon BMS

No settings required.

6. Result

Launches Falcon BMS in Instant Action. With the entry into the 3D world, the current heading of the aircraft is indicated by a compass rose on the OLED display.