Falcon BMS to Arduino Interface Tool (BMSAIT)

Speedbrake Indicator using a servo motor

Author	Robin „Hummer“ Bruns
Document version	1.0
Software version	1.3.8
BMS version	4.35u3
Date	21.04.2022

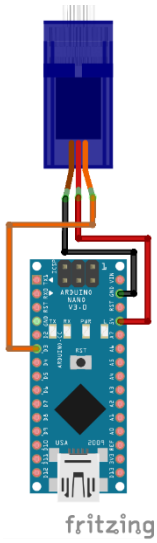
1. Overview

This is an example program to show how to use BMASIT to connect and use a servo motor to drive a speedbrake indicator panel. Primarily, BMSAIT is designed to work with Falcon BMS, but it will also work with DCS if the integration of BMSAIT an DCS-BIOS is set up.

To use this example, you need the following hardware components:

- one Arduino-board (i.e. UNO or NANO)
- a 5V micro servo motor (i.e. SG90 9g servo[1])
- connector cables

2. Wiring



PIN Arduino	PIN servo
GND	GND (brown)
5V	Vcc (red)
3	Signal (orange)

As the servo will be driven by a pwm (pulse width modulation) signal, only specific pins of the arduino are available. If you want to use a different pin, please check if the other pin also supports pwm.[2] For the Arduino Uno or Nano, the available pins are D3, D5, D6, D9, D10 or D11.

3. Program the Arduino

If you haven't installed the Arduino IDE yet, please refer to chapter 4.1.4 of the main BMSAIT documentation.
Load the BUPRadio Arduino sketch \Arduino Sketch\BMSAIT_SBIServo\. Into the Arduino IDE. If you use the wiring of chapter 2, there is no need to change the Arduino code at this point.
After selecting the correct Arduino board and COM port from the Arduino IDE menu, you can upload the Arduino code to the Arduino.

If you want to use a different Arduino pin to connect the servo motor, you need to change the reference in the Arduino code. Open the sketch in the Arduino IDE and change the pin at the following position of the BMSAIT_Servo.h:

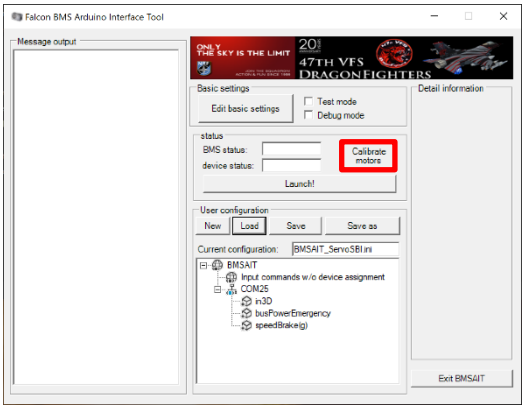
[1] https://www.az-delivery.de/en/products/az-delivery-micro-servo-sg90
[2] https://www.arduino.cc/reference/de/language/functions/analog-io/analogwrite/

```
#include <Servo.h>

#define SERVODELAY 200          //time (ms) the servo will pause after movement
#define SBIDELAY 500            //time (ms) the off-flag will show up when the

#define SBI_PIN              3 //Arduino PIN the servo is conncted to
#define SBI_OFF_ANGLE       90 //servo movement angle for the "off" symbol
#define SBI_CLOSED_ANGLE   135 //servo movement angle for the "closed" symbol
#define SBI_OPEN_ANGLE      55 //servo movement angle for the "open" symbol

```

If you did changes, you need to upload the sketch to the Arduino again.

Immediately after loading the sketch on the Arduino, the Servo will briefly move to the three symbol positions. After that, it will remain in the „off“ position.

If you missed the movement and want to check it again, you can use the BMSAIT windows app to call for another motor calibration.



Use the motor calibration to check for the correct alignment of the SBI symbols. If the symbols are offset, you need to do a manual calibration.

The manual calibration is set in the Arduino sketch in the file BMSAIT_Servo.h:

```
#include <Servo.h>

#define SERVODELAY 200          //time (ms) the servo will pause after movement
#define SBIDELAY 500            //time (ms) the off-flag will show up when the

#define SBI_PIN              3 //Arduino PIN the servo is conncted to
#define SBI_OFF_ANGLE       90 //servo movement angle for the "off" symbol
#define SBI_CLOSED_ANGLE   135 //servo movement angle for the "closed" symbol
#define SBI_OPEN_ANGLE      55 //servo movement angle for the "open" symbol

```

To align the SBI symbols, you need to change the angle values. Try a different value (usually up to 5°) and upload the sketch to the Arduino again. Relaunch the motor calibration to check the effects of your changes until the symbols will be shown correctly.

4. Settings for the BMSAIT windows app

Download and install the BMSAIT windows app. Make sure that the required basic settings are set. For this example, it is imperative that the path to the variable setting is set. Disable auto-launch for this test.

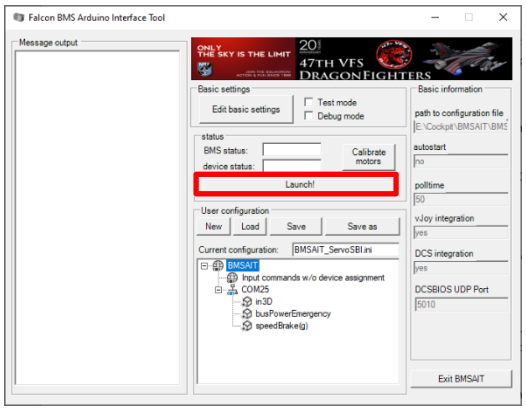
In BMSAIT, use the load button to load the setting file from this example (BMSAIT_demoSBI.ini). BMSAIT should show some information (one COM-Port and three data variables).

Make sure that the correct COM-Port of your Arduino is set. If you don't know the correct COM port, right-click the COM-Port in the configuration window and select "edit device". The pop-up window will allow you to hit the "Scan" button to check your COM ports for connected Arduino devices running a BMSAIT sketch.

I recommend to save both the basic settings and the user configuration at this time.

Now use the "Launch!" button to establish a connection between the windows app and the Arduino device.

The SBI should briefly move and then rest in the „off" position.



5. Result

Now run Falcon BMS. The SBI should remain "off" mode until BMS enters the 3D world. Once in 3D, the SBI should mimic the SBI display of the simulation. Pop the speedbrake to check if the SBI will show the correct symbology (open/closed).

Be advised that the SBI will briefly show the "off" flag when the speedbrake switches from open to closed and vice versa. This is a feature to simulate the real F-16 SBI behavior.

If DCS integration is activated in BMSAIT and DCS-BIOS properly set up, BMSAIT will automatically switch to DCS mode when launching DCS and start to display the speedbrake position when in 3D.