Falcon BMS to Arduino Interface Tool (BMSAIT)

Beispielprogramm Backup Radio

Autor	Robin „Hummer“ Bruns
Dokumentversion	1.0
Softwareversion	1.0
BMS Version	4.34u3
Datum	4.10.2020

1. Überblick

In den bisherigen Beispielen wurden immer nur einzelne Funktionen beispielhaft abgebildet. Hier wird nun darauf aufgebaut, indem mehrere Einzelbestandteile zu einem komplexen Projekt zusammengesetzt werden. Das Modul BUPRadio gibt es in drei Varianten, die sich durch leicht unterschiedliche Hardwarenutzungen unterscheiden.

Die für alle Varianten gleichen Grundfunktion bestehen darin, dass die Funktionen eines echten UHF Funkgerätes abgebildet werden sollen. Dazu gehört:

- Anzeige des UHF Channel und UHF Preset Frequenz auf 7-Segment-Anzeigen
- Ansteuerung der Switches des Backup Radio

Die Besonderheit besteht hier darin, dass die Anzeige der Displays gesteuert werden kann und dabei die in BMS simulierten Möglichkeiten übertrifft:

- Ist das Funkgerät abgeschaltet, sind die Displays abgeschaltet
- Befindet man sich im manuellen Modus, wird kein Preset-Kanal angezeigt
- Wird der Test-Button gedrückt, wird auf allen Displays eine 8 abgebildet
- Wird der Status-Button gedrückt, wird auf dem Display der manuellen Frequenz die dem aktuellen Preset-Kanal hinterlegte Frequenz angezeigt.

Zur Umsetzung des BUPRadio gehört auch eine Funktion der BMSAIT Windows App. In den Basiseinstellungen wird ein Verweis auf Eure <Pilot>.ini gespeichert. In dieser Datei wird gespeichert, welche Frequenzen ihr in der 2D Welt für die 20 Kanäle gespeichert habt. Mit Einstieg in die 3D Welt werden die Frequenzen aus der .ini gelesen und von der BMSAIT App an den Arduino gesendet, um diese im Status-Mode des BUPRadio anzuzeigen.

Variante 1 der Abbildung des BUPRadio:

> Variante 1 stellt die Minimalvariante zur Abbildung der komplexen Möglichkeiten des BackupRadio dar. Hierbei wird ein einzelnes Max7219 Display Tube genutzt, um sowohl die manuelle Frequenz, als auch den Preset-Kanal anzuzeigen. Die manuelle Frequenz wird auf den ersten sechs Displays angezeigt und die beiden letzten Displays für den Preset-Kanal.
>
> Um eine ordentliche Anzeige zu erhalten, sollten die 8 Displays daher von der Platine entlötet werden, um diese getrennt in einem Panel verbauen zu können. Dies verursacht etwas Lötarbeit und zugegebenermaßen etwas Kabelgewirr, aber man kommt mit einem Display aus.
>
> Benötigte Hardware:
>
> - Ein Arduino-Board (z.B. ein UNO)
> - Ein Max7212 7-Segment-Tube
> - Zwei Taster
> - Einen Drehschalter mit drei Rastungen
> - Einen Kippschalter (ON-OFF-ON)
> - Verbindungskabel

Variante 2 der Abbildung des BUPRadio:

Bei der Variante 2 wird der Aufbau um ein weiteres Bauteil zur Anzeige von 7-Segement-Displays erweitert. Es handelt sich um ein kleineres Bauteil als das Max7219 und enthält 4 Anzeigen.

Das separate Tube ermöglicht eine Abbildung der manuellen Frequenz und des Preset-Channel auf einem Panel, ohne die einzelnen 7-Segment-Elemente umlöten zu müssen. Der Nachteil dieser Lösung besteht darin, dass sowohl das Max7219 als auch das TM1637 zu viele Elemente haben. Dies muss man hinnehmen, die nicht benötigten Elemente von den Tube-Platinen entfernen oder die Elemente irgendwie verdecken.

Ergänzend zu Variante 1 benötigt ihr folgende Hardware:

- Ein TM1637 7-Segment-Tube

Variante 3 der Abbildung des BUPRadio:

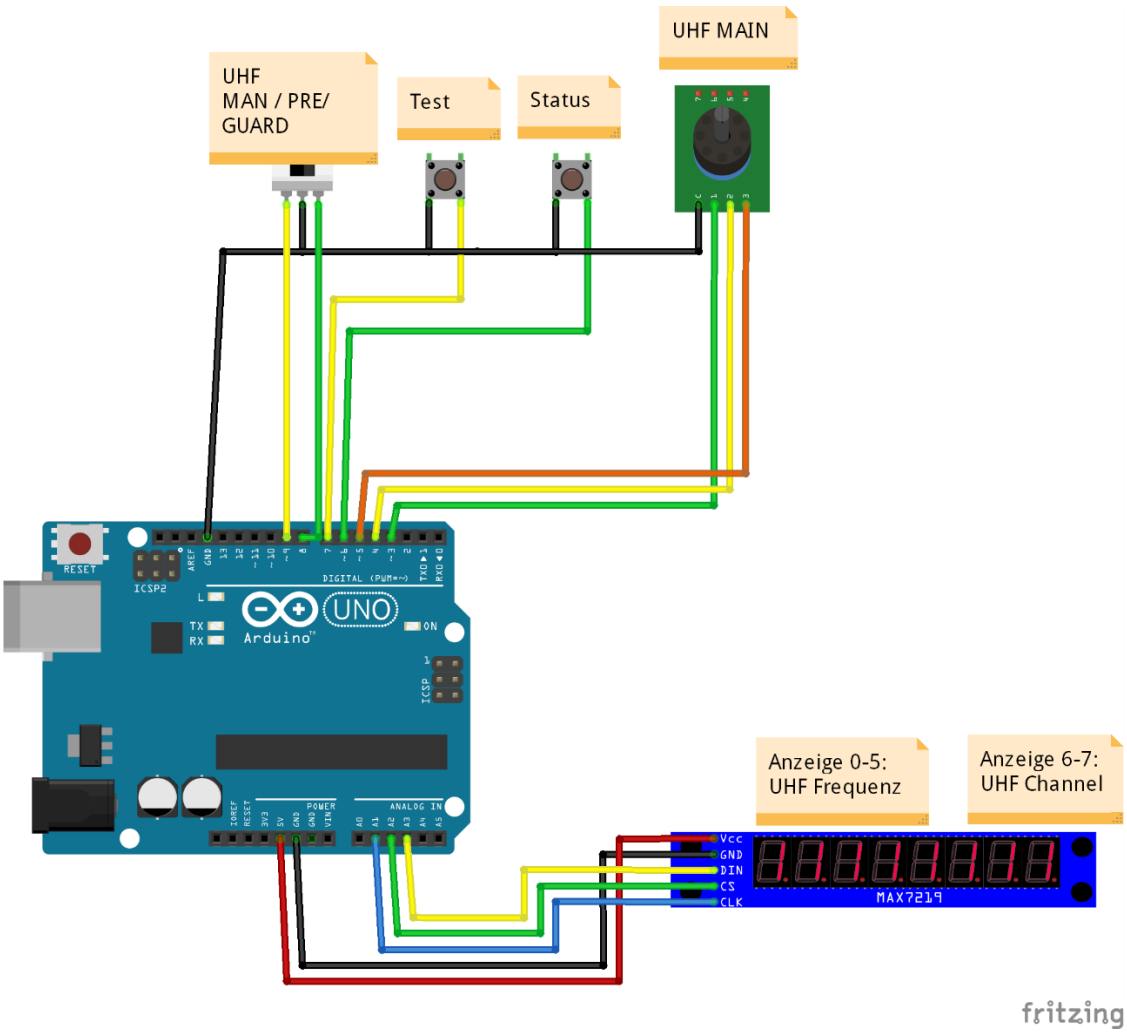
Variante 3 ist die deluxe-Lösung und deckt alle Funktionen des BackupRadio Panel mit allen Schaltern ab. Variante 3 baut auf Variante 1 auf. Hierfür wird ein Drehimpulsgeber für die Einstellung des Preset-Kanal sowie Drehschalter für die Einstellung der manuellen Frequenzen verwendet.

Ergänzend zu Variante 1 benötigt ihr:

- Einen Drehencoder
- Einen Drehschalter mit 2 Rastungen
- Einen Drehschalter mit 4 Rastungen
- Drei Drehschalter mit 10 Rastungen
- 1 Widerstand 4.7kOhm
- 5 Widerstände 2.2kOhm
- 36 Widerstände 1kOhm

2. Verkabelung

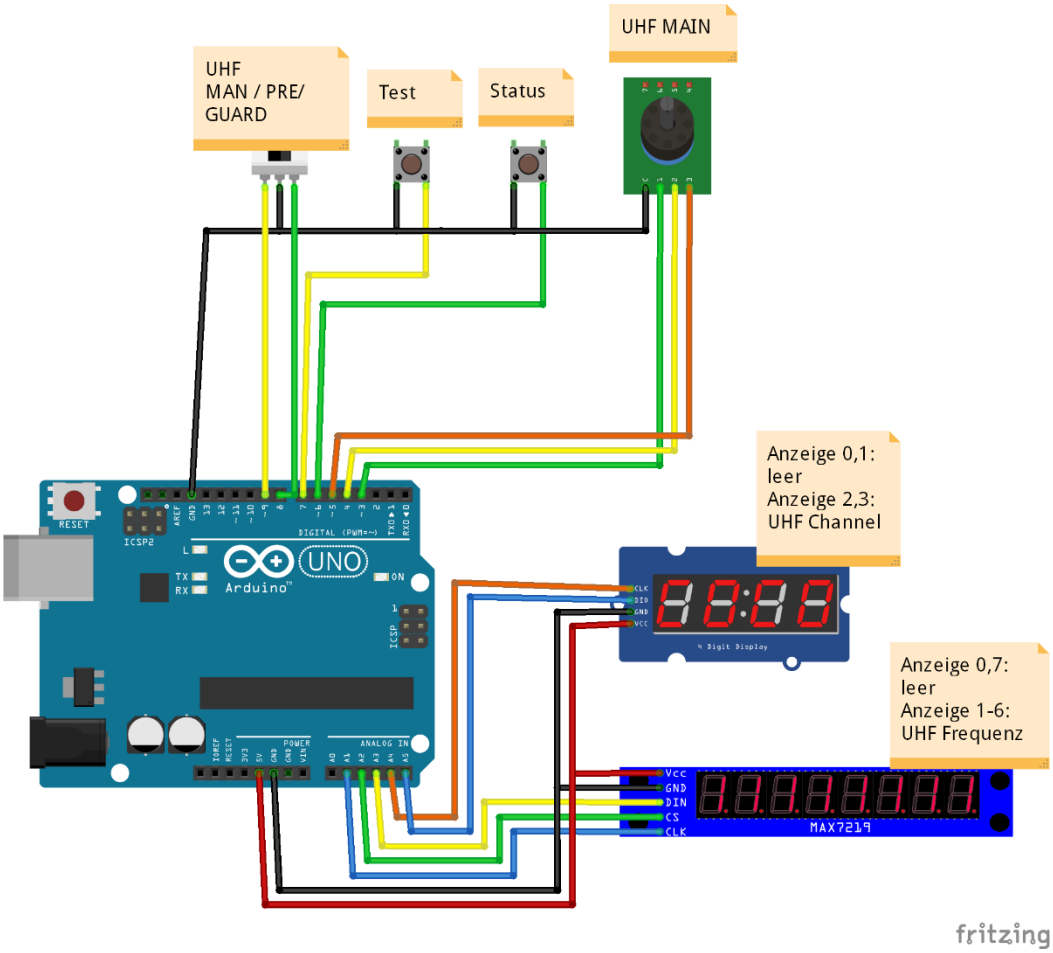
2.1. Variante 1



Arduino	MAX7219
A1	CLK
A2	CS
A3	DIN

Schalter	Arduino PIN	Gegenseite
Taster Test	7	GND
Taster Status	6	GND
UHF MAIN - OFF	3	GND
UHF MAIN - MAIN	4	GND
UHF MAIN - BOTH	5	GND
UHF MODE – MANUAL	8	GND
UHF MODE – PRESET	9	GND
UHF MODE – GUARD	10	GND

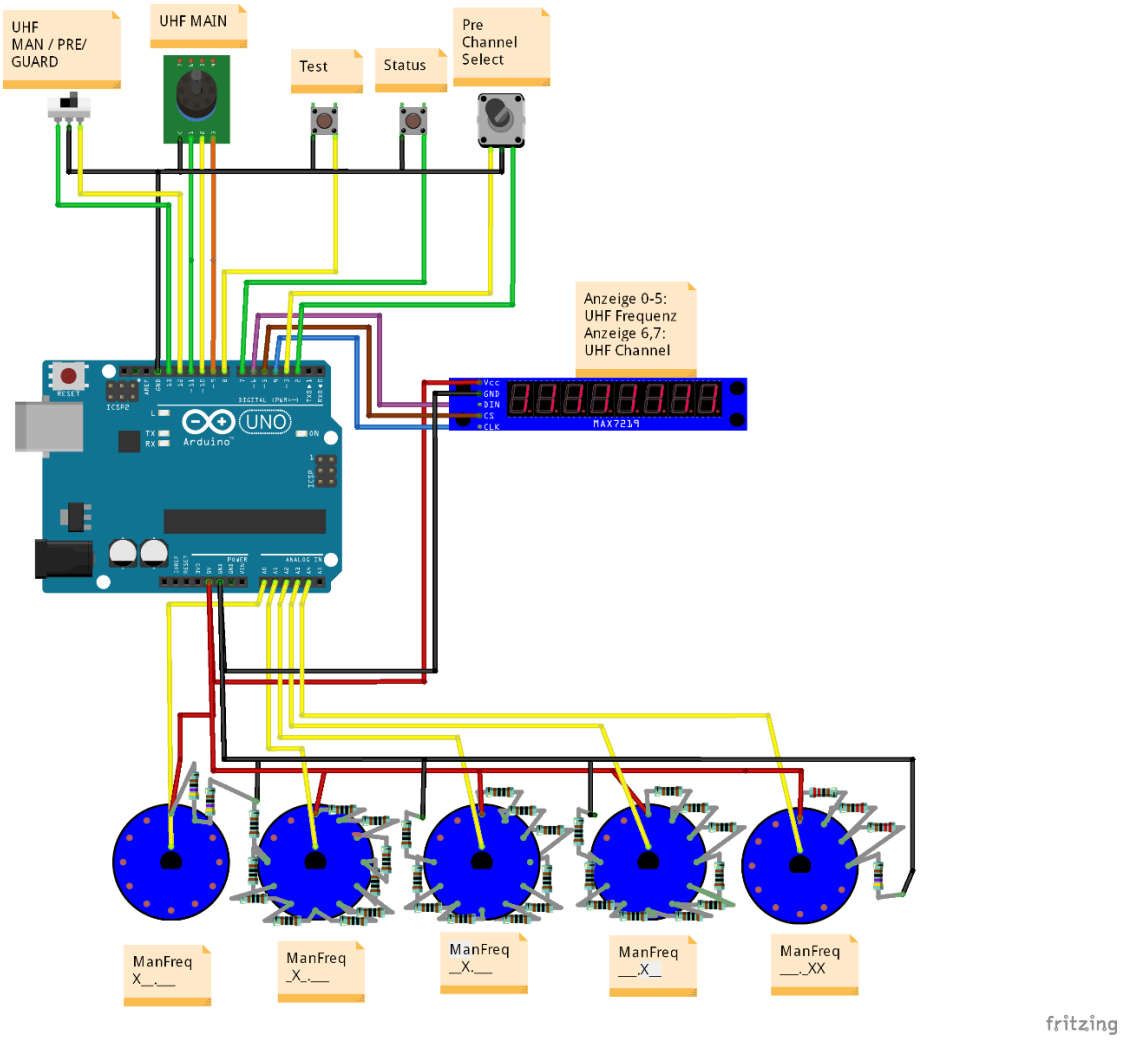
2.2. Variante 2



Arduino	MAX7219	TM1637
A1	CLK	
A2	CS	
A3	DIN	
A4		DIO
A5		CLK

Schalter	Arduino PIN	Gegenseite
Taster Test	7	GND
Taster Status	6	GND
UHF MAIN - OFF	3	GND
UHF MAIN - MAIN	4	GND
UHF MAIN - BOTH	5	GND
UHF MODE – MANUAL	8	GND
UHF MODE – GUARD	9	GND

2.3. Variante 3



Arduino	MAX7219
4	CLK
5	CS
6	DIN

Schalter	Typ	Arduino PIN	Gegenseite
PresetChannelKnob Links	Drehimpulsgeber1	2	GND
PresetChanellKnob Rechts	Drehimpulsgeber1	3	GND
Taster Test	Taster1	8	GND
Taster Status	Taster2	7	GND
UHF MAIN - OFF	Drehschalter1	9	GND
UHF MAIN - MAIN	Drehschalter1	10	GND
UHF MAIN - BOTH	Drehschalter1	11	GND
UHF MODE – MANUAL	Kippschalter1	12	GND
UHF MODE – GUARD	Kippschalter1	13	GND
MNL Freq X__.___	Drehschalter2	A0	GND
MNL Freq _X_.___	Drehschalter3	A1	GND
MNL Freq __X.___	Drehschalter4	A2	GND
MNL Freq ___.X__	Drehschalter5	A3	GND
MNL Freq ___._XX	Drehschalter6	A4	GND

Beachtet die Besonderheit, dass der Drehschalter 1 digital angeschlossen werden, d.h. jede Stellung verfügt über einen eigenen PIN am Arduino, über den die Position ausgelesen werden kann.

Bei den Drehschaltern 2 bis 6 werden die Stellungen über die Software bestimmt. Hierzu sind die Rastungen des Drehschalters mit Widerständen zu verbinden (siehe BMSAIT Doku in der Erklärung des Moduls Switches). Die Summe aller Widerstände bei einem Drehschalter sollte ca. 10kOhm betragen.

2.4. Interne Kommandoverarbeitung

Im Modul BUPRadio ist es erforderlich, dass die Schalter UHF MAIN, UHF MODE, STATUS und TEST an dem Arduino angeschlossen sind, auf dem dieses Modul läuft. Der Grund dafür ist, dass die Schalterstellungen benutzt werden, um Funktionen dieses Moduls zu beeinflussen.

BATT PWR	UHF MAIN	UHF Mode	Test	Status	Preset Anzeige	Manual Anzeige
OFF	Egal	Egal	Egal	Egal	Keine Anzeige	Keine Anzeige
ON	OFF	Egal	Egal	Egal	Keine Anzeige	Keine Anzeige
ON	MAIN/BOTH	MNL	OFF	OFF	Keine Anzeige	Anzeige MNL
ON	MAIN/BOTH	PRESET	OFF	OFF	Anzeige CNL	Anzeige MNL
ON	MAIN/BOTH	GUARD	OFF	OFF	Keine Anzeige	243.000
ON	MAIN/BOTH	Egal	OFF	ON	Anzeige CNL	Anzeige Freq des gewählen CNL
ON	Egal	Egal	ON	Egal	88	888888

Die Battery Power wird durch eine BMSAIT Variable direkt aus der Simulation gelesen. Dieser Schalter muss daher nicht an diesem Arduino angeschlossen sein. Für die Funktion des Moduls muss dafür aber die entsprechende Datenvariable im Bereich UserConfig aufgenommen sein (ID 1260 „PWRST“).

Die Funktionsweise besteht darin, dass den angeschlossenen Schaltern im Modul Switches ein interner Code hinzugefügt wird, über den das Modul BUPRadio die aktuelle Schalterposition erkennen kann.

```
Schalter schalter[]=
{
  {6, "STATU" ,1, 0, 0, "00", "00", 6}   //STATUS button
 ,{7, "TEST"  ,1, 0, 0, "00", "00", 7}   //TEST button
 ,{3, "UHFOF" ,2, 0, 0, "01", "00", 1}   //UHF Main Switch - OFF
 ,{4, "UHFMA" ,2, 0, 0, "02", "00", 2}   //UHF Main Switch - MAIN
 ,{5, "UHFBO" ,2, 0, 0, "03", "00", 2}   //UHF Main Switch - BOTH
 ,{8, "MNL"   ,2, 0, 0, "04", "05", 3}   //UHF Mode Switch MAN
 ,{9, "GRD"   ,2, 0, 0, "06", "05", 5}   //UHF Mode Switch GRD
};
```

Ein Coding wertet die Stellungen aus und bestimmt darüber, ob die Displays aktiviert sind und welcher Wert angezeigt werden soll (Funktionen *BUPRadio_Update* und *CheckPowerOn*).

3. Programmierung des Arduino

Falls die Arduino IDE noch nicht installiert ist, lest bitte das Kapitel 4.1.4 der BMSAIT Dokumentation.

Ruft nun die .ino in der gewünschten Variante aus dem Ordner \Arduino Sketch\BMSAIT_BUPRadio\ mit einem Doppelklick auf. Das Sketch wird in der Arduino IDE geladen. Wenn ihr die Verkabelung gem. Kapitel 2 durchgeführt habt, sind hier keine Anpassungen erforderlich.
Nachdem ihr das richtige Arduino-Board ausgewählt habt, ladet ihr das Sketch auf den Arduino hoch.

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. Wählt das PUSH-Prinzip und schaltet den Autostart aus.

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

Aktiviert den Testmodus und startet die Verarbeitung. Wenn alles geklappt hat, sollte das/sollten die 7-Segment-Anzeigen kurz einen Wert anzeigen.

5. Ergebnis

Beendet den Testmodus und startet die Verarbeitung des BMSAIT. Startet FalconBMS und betretet die 3D Welt. Prüft nun die Schalterstellungen und das Ergebnis der Displayanzeige gem. Tabelle:

Bei der abgebildeten Simulation kann derzeit nicht berücksichtigt werden, dass die Anzeige des Preset Channel und der Frequenz eigentlich nur dann erscheint, wenn die Lautstärke des UHF Radio aufgedreht wurde

Die Prüfung, ob die Stromversorgung im Cockpit auch aktiviert ist, erfolgt erst in der 3D-Welt. Außerhalb der 3D Welt wird nur auf die Stellung der UHF MAIN und UHF Mode Schalter geprüft, um zu bestimmen, ob die Anzeigen aktiviert sind oder nicht.

Die korrekte Anzeige der Frequenz zu einem Channel im Status-Mode funktioniert nur, wenn in den Basiseinstellungen der BMSAIT Windows App der Verweis auf die richtige Pilot.ini hergestellt wurde und man sich in der 3D Welt befindet.

Bumerang war übrigens so freundlich ein schönes Video zu machen, in dem die Funktionsfähigkeit des BUPRadio Moduls in seinem Pitbau-Projekt demonstriert wird:

BUPRadio F-16 Home Cockpit