F4ToSerial

Envia dados do Falcon BMS para o teu cockpit

Documentação (Português)

Documentação escrita por: Myoda
Traduced by : Paulo Cracel Silva

Versão do documento: 20180812

Conteúdos

1 Programa F4ToSerial

1.1 - Introdução

O programa F4ToSerial foi desenvolvido com o objetivo de incluir numa única aplicação o controlo de todos os elementos "ativos" de um cockpit de simulação.

Por elementos ativos queremos dizer aqueles que recebem informações do simulador e reagem de acordo com: manómetros, luzes, 7 segments displays e ecrãs.

O F4ToSerial foi desenvolvido exclusivamente para o simulador Falcon 4 BMS da versão 4.33 ou superior.

A informação vem da memória compartilhada do simulador e é roteada através da porta serial para controlar placas capazes de conduzir esses chamados elementos "ativos".

O F4ToSerial é um programa "tudo em um" que visa fornecer uma solução alternativa para a multiplicidade de programas atualmente disponíveis para o Falcon BMS.

O programa F4ToSerial, no entanto, tem duas grandes restrições.

- - Só funciona numa direção (Falcon BMS Cockpit)

- - Foi desenvolvido para a versão "Block 52" do F16.

Enviar os dados do cockpit para o simulador (na outra direção) não representa nenhuma dificuldade e não será discutido aqui, pois envolve a simulação de depressão de teclas em 90% dos casos.

O programa F4ToSerial foi desenvolvido para funcionar com placas eletrónicas do tipo Arduino e o código dessas placas é fornecido.

Finalmente, note que este programa foi desenvolvido por mim como parte da criação do cockpit da minha casa e foi colocado online para a comunidade. Não pode ser usado para fins profissionais e permanece grátis.

1.1 - Metodologia

Quando comecei a projetar o meu cockpit simulador F16, percebi rapidamente que o dinheiro é um fator determinante na qualidade do cockpit acabado. Embora o tempo e a especialização importem, um cockpit de alto nível e acabado custa dezenas de milhares de euros... E eu não minto!

Não tendo um orçamento expansível, tive que me certificar que consumiria o menor número possível de componentes para a criação do meu cockpit.

Geralmente, existem duas abordagens para construir um cockpit. Em um deles, o orçamento não é levado em conta e, no outro, prestamos atenção a cada euro investido.

Este princípio aplica-se a "lightBits" e 7 segments displays, por exemplo, mas também de forma mais geral a todos os módulos do cockpit, botões, interruptores, ecrãs, etc., mas também ao ACESII, acelerador, etc.

Por outro lado, quanto mais simples a composição, mais as conexões e essa simplificação torna-se complexa e difícil de desenvolver.

Não sendo um faz-tudo genial ou um engenheiro eletrónico de nascença, a minha abordagem está a meio caminho entre os dois métodos.

É por isso que neste guia eu escolhi usar este ou aquele método que às vezes simplifica e às vezes reduz a despesa.

Claro, tudo pode ser melhorado e eu não pretendo ter a solução definitiva.

1.2 - Que tipo de placa eletrónica deve usar com o F4ToSerial?

Existem vários tipos de placas que são muito úteis para o simulador de cockpit.

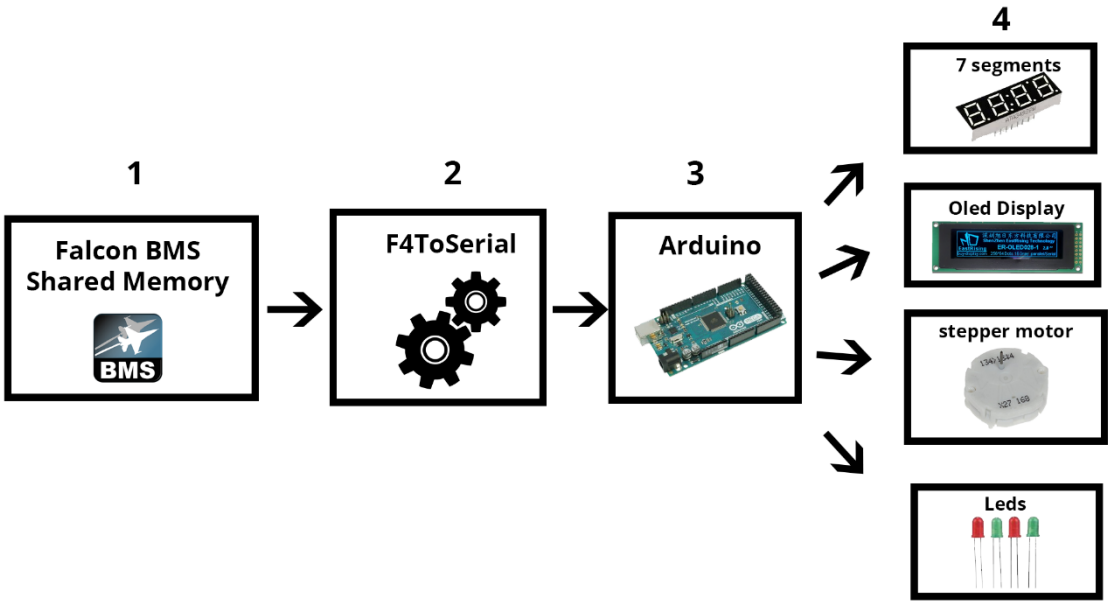
Placas Photon, Arduino e Pokeys, etc. Esta última não será discutida neste documento porque o seu uso com o programa F4ToSerial ainda não está implementado. Cada uma tem vantagens e desvantagens e este documento não pretende fazer uma comparação ou guiá-lo ao escolher a sua placa.

Note, no entanto, que o programa F4ToSerial foi desenvolvido apenas para a placa Arduino Mega.

A placa Arduino UNO é compatível, mas devido à sua pequena quantidade de memória, não pode usar uma das bibliotecas essenciais do programa C ++.

Outras placas ainda não são suportadas e não estão planeadas para o momento.

1.3 – Como são enviados os dados para o cockpit?



Antes de ter os manómetros que funcionam no cockpit do simulador ou no ecrã do DED do F16 que exibe informações, são necessárias várias etapas. O diagrama anterior descreve essas etapas que são detalhadas a seguir.

1.3.1 – Etapa 1: Shared Memory

A memória compartilhada é uma área que os desenvolvedores do Falcon BMS gentilmente implementaram e documentaram para permitir que os construtores de cockpit leiam as informações atuais da aeronave e do voo: (altitude, rumo, posição dos joysticks etc.).

O primeiro passo é, portanto, ter acesso ao registro de informações da memória compartilhada. O F4ToSerial usa o Windows F4SharedMem.dll DLL que é instalado no diretório de instalação do programa F4ToSerial.

1.3.2 – Etapa 2: F4ToSerial

Na etapa 2 são recuperados dados da memória compartilhada, transformando-os e enviando-os para o cockpit. Nesta etapa, o programa F4ToSerial faz um loop permanente nos endereços de memória, recupera e prepara as informações antes de enviá-las para a porta serial.

Os dados são transformados, por exemplo, do binário em string legível pelo cartão Arduino na próxima etapa.

Exemplos de quadros de dados no formato Json.

Configurando um motor de passo:

{"SETUP_STEPPER":{"RPM":{"NAME":"RPM","PINS":[46,44,42,40],"PINSCOUNT":4,"STEPS":600,"MAXSPEED":1000,"ACCELERATION":1000}}}

Atualização de uma matriz de LEDs:

{"UPDATE_MATRIX":{"A":{"MATRICE":[[1,1,0,0],[1,0,0,1],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]}}}

Exibindo informações da ecrã DED:

{"SET_DISPLAY":{"DED":[" UHF 292.30   STPT $ 8","","," VHF   1   10:56:20","","," M1 3 C 6400 MAN T 75X"]}}

1.3.3 – Etapa 3: Recuperação e transformação do sinal

Nesta etapa, a placa Arduino recebe informações da porta serial e a transforma em sinais elétricos compatíveis com os elementos ativos (manómetros, LEDs, 7 segments displays e displays).

1.3.4 - Etapa 4: Ativação dos elementos do cockpit

Nesta última etapa os elementos ativos evoluem de acordo com os sinais recebidos (deslocamento dos manómetros, iluminação dos LEDs etc.).

Note que todas essas etapas ainda funcionam na direção do Falcon BMS para o cockpit. A memória compartilhada é acessível somente para leitura, qualquer interação com o simulador será por meio de depressão sobre uma ou mais teclas.

2 Download e instalação do programa F4ToSerial

2.1 - Download do programa F4ToSerial

Pode fazer o download do programa F4ToSerial do site:

http://f4toserial.mini-cube.fr/.

A versão mais recente da aplicação está disponível aqui:
http://f4toserial.mini-cube.fr/download-latest-version/

Versões anteriores do programa também estão disponíveis.

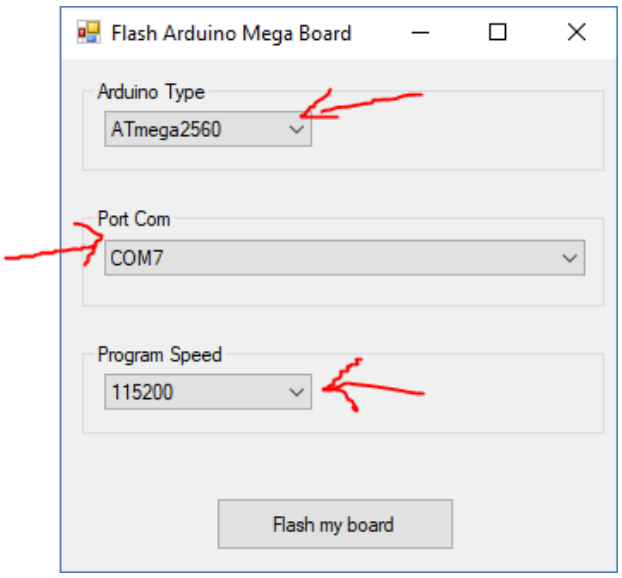
2.2 – Download do código ++ para o Arduino

2.2.1 – Método fácil:

1) A partir do programa F4ToSerial, se possui uma placa Arduino Mega 2560, poderá fazer o upload do programa C ++ diretamente para a sua placa Arduino.

2) Para fazer isso, clique no separador "General Settings" e depois em "Flash Arduino Board".

3) Em seguida, indique o tipo de placa (tipo Arduino), a porta serial (porta Com) e a velocidade (velocidade do programa) e clique em "Flash my board".



O parâmetro "Program Speed" é usado para ajustar a velocidade de leitura da porta serial da placa Arduino.

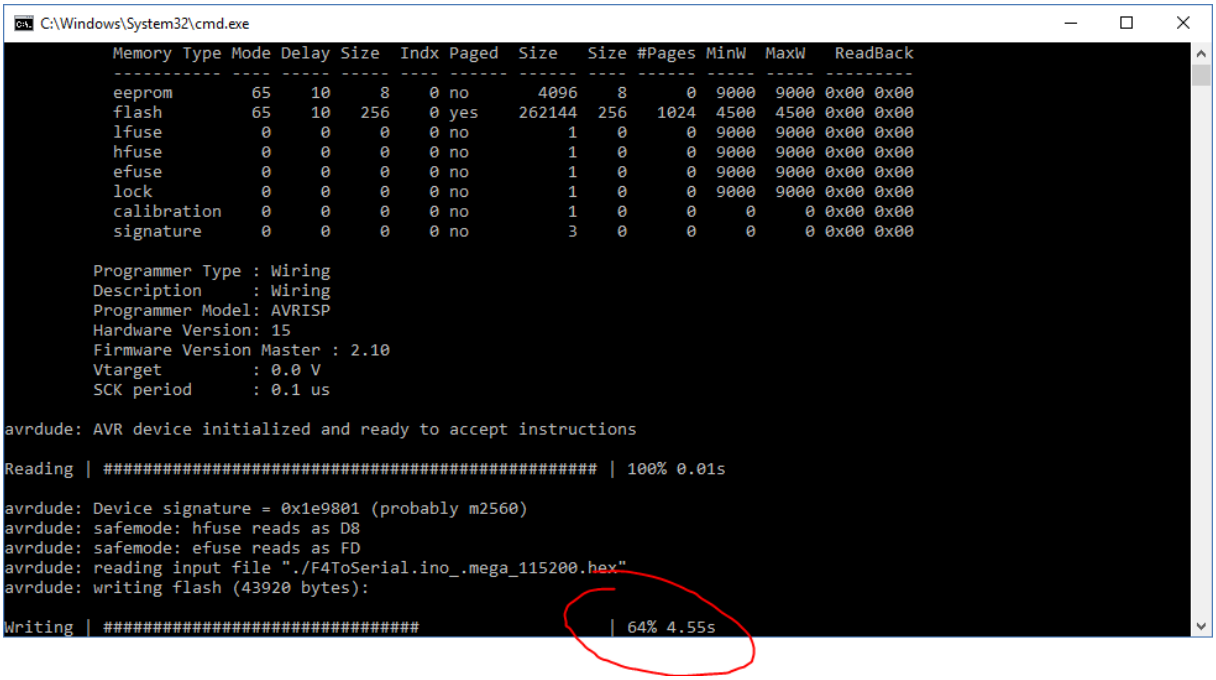
Por padrão, o programa F4ToSerial envia dados na porta serial em 115.200 bauds.

Algumas placas podem ler dados a uma velocidade de 1.000.000 bauds.

Este parâmetro corresponde à instrução Serial.Read () do código C ++ que está na placa do Arduino.

Portanto, ele deve ser idêntico à velocidade de transferência do programa Taxa de velocidade de transmissão serial (F4ToSerial) definida na guia "Configurações gerais".

4) Uma janela de comando do DOS é aberta e o programa está a ser transferido para a placa do Arduino.



2.2.2 – Método alternativo:

> *Este método é de interesse se o programa F4Terial falhar em piscar a sua placa Arduino ou se a velocidade precisar de ser ajustada manualmente.*

1) Faça donwload aqui de um programa que permite transferir um arquivo Hex para uma placa Arduino http://xloader.russemotto.com/
2) Transferir o código Hex diretamente para a placa Arduino (Mega (ATMEGA2560)). http://f4toserial.com/resources/

2.2.3 - Método completo:

Para executar os seus módulos ou elementos ativos totalmente, você também precisará baixar o código C ++ compatível com as placas Arduino.

O link de download do código está aqui:

https://bitbucket.org/falconbms/

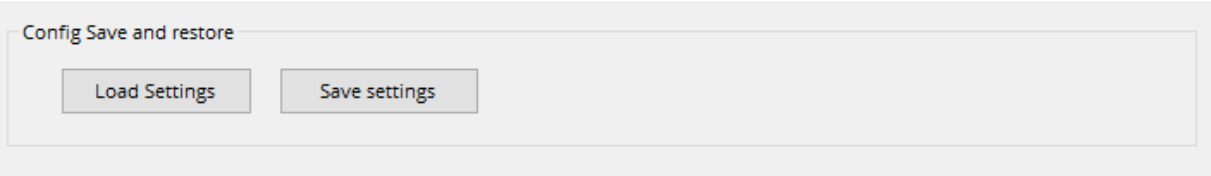
Arquivos de código-fonte C ++ para a placa Arduino estão disponíveis aqui:
https://bitbucket.org/falconbms/arduino-commons.git

> *Convido a clonar os arquivos do repositório com uma ferramenta do tipo SourceTree para manter os arquivos atualizados se novas versões do código C ++ forem colocadas online.*

3 Manual do utilizador F4ToSerial

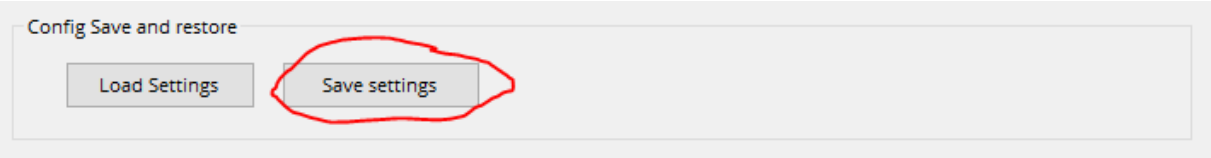
3.1 - Backup e Restauração da Configuração

O programa F4ToSerial permite guardar a última configuração. Este último é tedioso para configurar a primeira vez, razão pela qual é importante registrar o último estado no final da criação de elementos ativos.



3.1.1 - Backup da configuração

O backup da configuração é feito clicando no botão "Save settings".



Deve então especificar um local e um nome para o arquivo de configuração. A extensão do arquivo de backup é ".xml". Não é necessário especificá-lo.

Arquivos de configuração podem ser facilmente modificados com um editor de HTML / XML.

Aqui está um exemplo de um arquivo de configuração:

<?xml version="1.0" encoding="utf-8"?>

<serial_settings>{"7_chaff":"","7_flares":"","7_fuel":"","7_uhf_chan":"","7_uhf_freq":"","aft":"","bcd_decoder_0":"COM1","bcd_decoder_1":"COM1","cabinAlt":"","ded":"","epu":"","ffi":"","ftit":"","fwd":"","hyd_press_a":"","hyd_press_b":"","Matrix_A":"","Matrix_B":"","Matrix_C":"","Matrix_D":"","Matrix_E":"","noz":"","oil":"","oxygen":"","pfl":"","pitch":"","roll":"","rpm":"","yaw":""}</serial_settings>
<steppers_speed_accell>{"aft":[1000,1000],"cabinAlt":[1000,1000],"epu":[1000,1000],"ftit":[1000,1000],"fwd":[1000,1000],"hyd_press_a":[1000,1000],"hyd_press_b":[1000,1000],"noz":[1000,1000],"oil":[1000,1000],"oxygen":[1000,1000],"pitch":[1000,1000],"roll":[1000,1000],"rpm":[1000,1000],"yaw":[1000,1000]}</steppers_speed_accell>

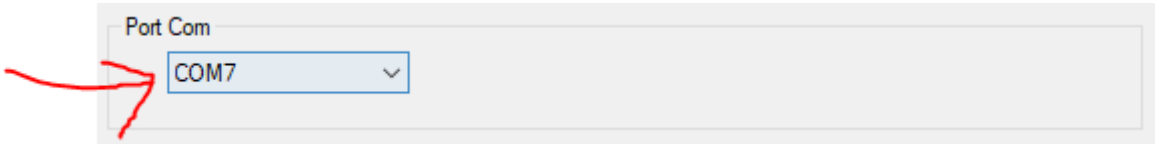
3.1.2 - Restauração da Configuração

Para carregar a configuração é preciso clicar no botão "Load settings".

Qualquer configuração atual será substituída pela configuração no arquivo de configuração durante o carregamento. Portanto, tenha cuidado para não sobrescrever uma configuração correta, porque a operação não pode ser cancelada.

3.2 – Configuração dos elementos ativos

O princípio de definir os elementos ou módulos ativos (manómetros, indicadores, 7 segment displays e ecrãns) é, em primeiro lugar, escolhendo a porta serial nos parâmetros do elemento em questão.



Cada placa do Arduino conectada ao computador possui sua própria porta serial.

Uma vez conectado, a lista exibe os nomes das portas seriais disponíveis.

> *A lista de portas Com não é atualizada em tempo real. Terá de conectar o seu equipamento antes de iniciar o programa F4ToSerial.*

3.2.1 - Manómetros

Visão geral:

Fiz vários testes antes de tomar a decisão de usar os motores x27.168 para manómetros porque:

- Eles não são barulhentos em comparação com servomotores
- O seu ângulo de rotação é alto comparado a um servomotor.
- Eles consomem pouca corrente (podem ser conectados sem fontes de alimentação externas na placa do Arduino)
- Eles são muito rápidos.
- O custo de um x27 é muito baixo (menos de 2 € por peça).

Os manómetros do cockpit que podem ser usados no F4ToSerial são os seguintes:

Consola Central:

- Rpm
- Noz
- Oil
- Ftit

Consola Direita:

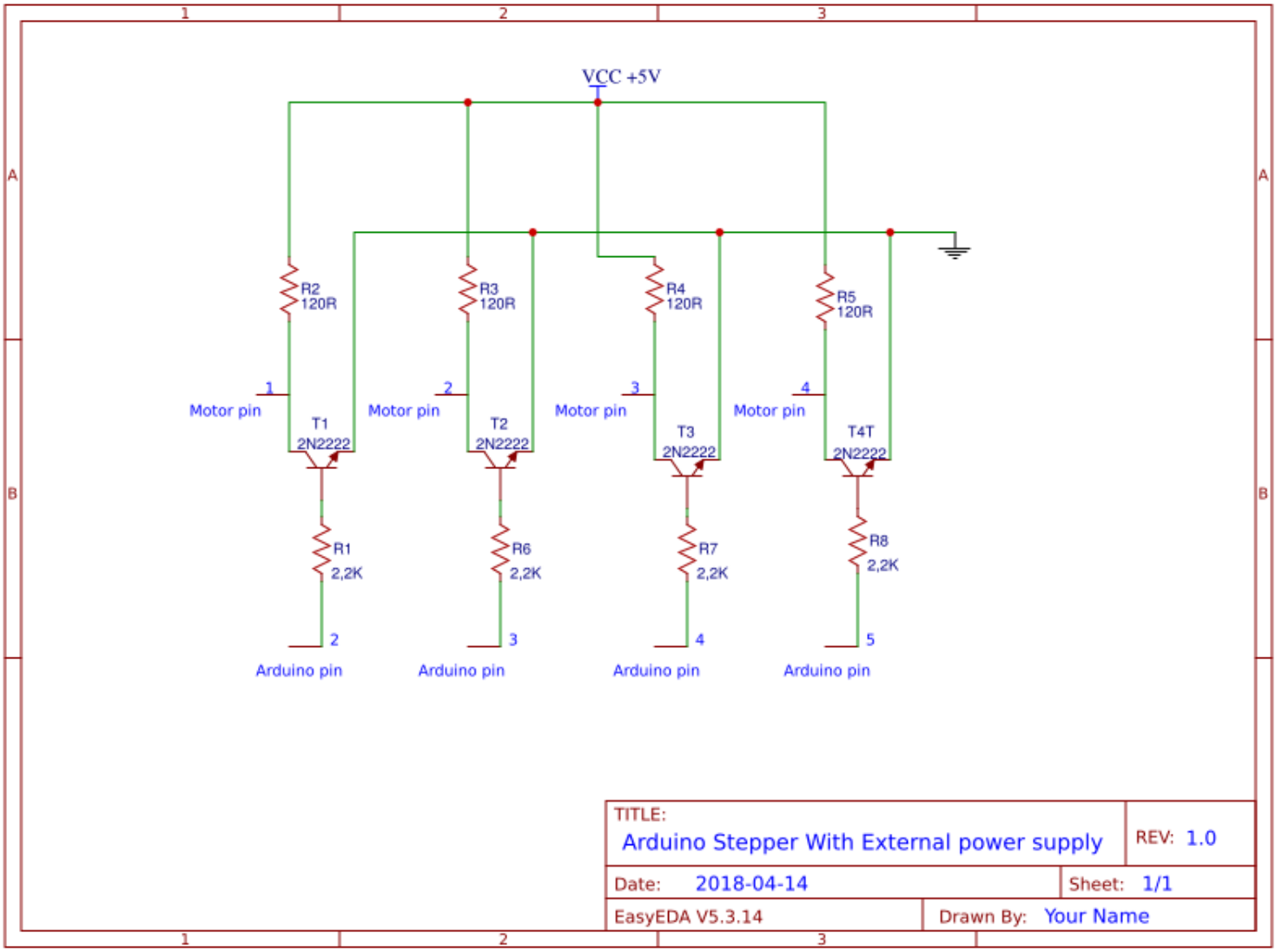
- Epu
- Fwd
- Aft
- Hyd Press A
- Hyd Press B
- Cabin

Consola Esquerda:

- Roll Trim
- Pitch Trim
- Yaw Trim

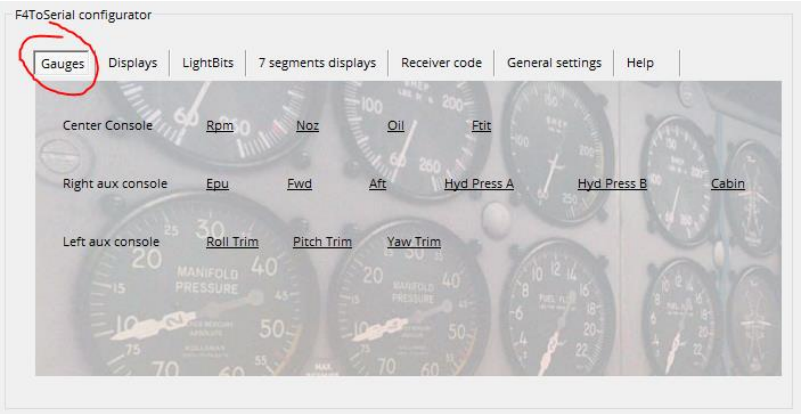
> *Existe uma característica especial para o manómetro de HYD PRESS. Este último usa 2 agulhas no mesmo eixo. Neste caso, outro tipo de motor deve ser usado: O x40.168.*

O diagrama a seguir ilustra como conectar o motor x27 com uma fonte de alimentação externa. Isso torna possível conectar todos os motores à mesma placa Arduino.



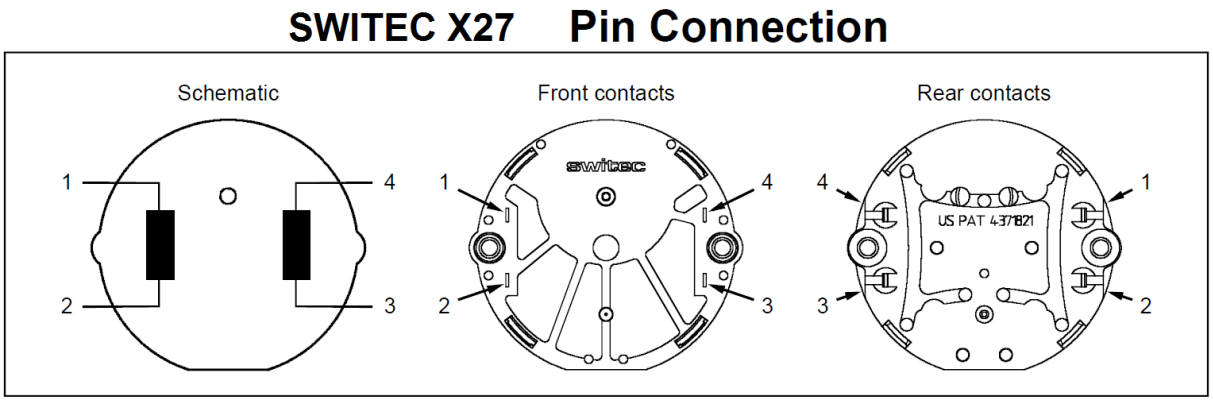
Uso no F4ToSerial.

No F4ToSerial, os manómetros são configurados no primeiro separador.



Clique no link do módulo para entrar na sua configuração.

Os motores X27 têm 4 pinos:

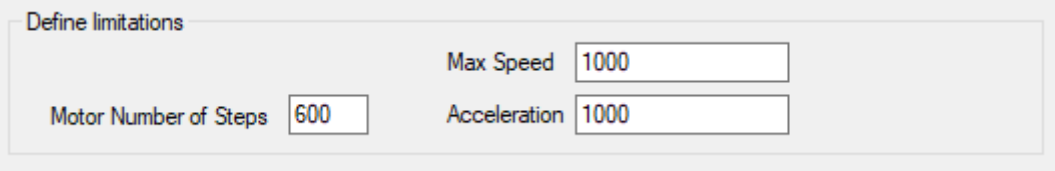


Estes 4 pinos estão representados no F4ToSerial na área "Define pins"

Aqui você deve inserir os números de pinos correspondentes na sua placa Arduino.

> *Neste exemplo, que não está correto, o pino 1 do motor está conectado ao pino 0 da placa Arduino e o pino 4 do motor está conectado ao pino 3 da placa Arduino.*

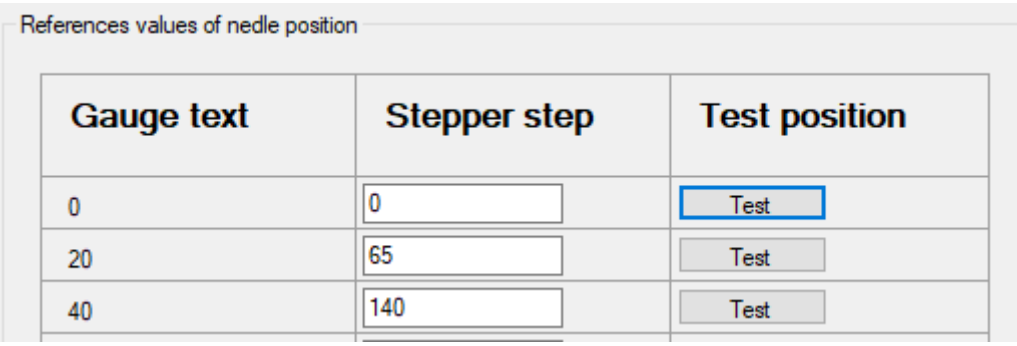
A área a seguir, "Define limitations", permite configurar os limites específicos dos mecanismos do X27.



- **Steps**: Os motores X27.168 são motores de 600 passos, mas você pode alterar o valor aqui. Nenhum interesse a menos que seu motor seja diferente de um x.27.168.
- **Max Speed**: Este parâmetro corresponde à velocidade máxima permitida pelo motor. É calculado com base nas informações da biblioteca AccelStepper: http://www.airspayce.com/mikem/arduino/AccelStepper/
- **Acceleration**: Este parâmetro corresponde à aceleração máxima permitida pelo motor. É calculado com base nas informações da biblioteca AccelStepper: http://www.airspayce.com/mikem/arduino/AccelStepper/

> *Eu recomendo ter cuidado se alterar os valores das limitações dos motores. Você pode danificá-lo ou exibir valores incorretos. Os valores 600, 1000 e 1000 vêm dos meus muitos testes.*

A área de "References values of needle position" permite que cada manómetro do seu cockpits seja combinado com os valores dos manómetros do simulador Falcon 4 BMS.



Como cada cockpit é diferente, este sistema permite que todos os tipos de suportes de manómetros sejam usados. No meu caso eu usei um suporte que vem do site: http://hispapanels.com/

Mas há outro suporte com outros valores de calibre. Portanto, é importante ter um sistema que permita a

correspondência de todos os manómetros de todos os cockpits.

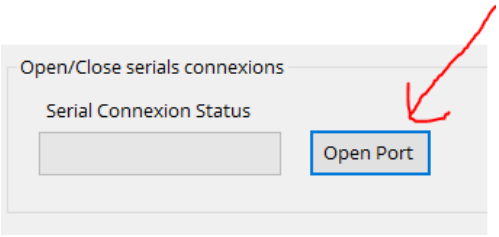
Além disso, alguns manómetros F16 com valores não lineares, essa opção permite corresponder exatamente os valores do seu cockpit aferir ao do Falcon BMS.

Os valores de referência do simulador são mostrados na primeira coluna.

Você deve combinar os seus valores de calibre com os valores do simulador, indicando um número de "etapas" na coluna central.

Clique no botão "teste" para tentar combinar os valores entre eles.

Atenção, para tentar mover as agulhas deve ter conectado corretamente o seu motor no cartão, fechar o Falcon BMS e abrir antes as portas seriais na página principal do programa F4ToSerial.



A foto a seguir mostra comparações entre um manómetro de rotação no Falcon BMS e um manómetro de rotação num cockpit doméstico.

Você deve apontar o seu ponteiro para os valores 70 (ponto vermelho), 80 (ponto verde) e 90 (ponto amarelo) clicando no botão de teste e tendo o mesmo resultado no seu manómetro.

Encontre mais ajuda no meu vídeo aqui:

https://youtu.be/dqMFBQkAozs?t=5m55s

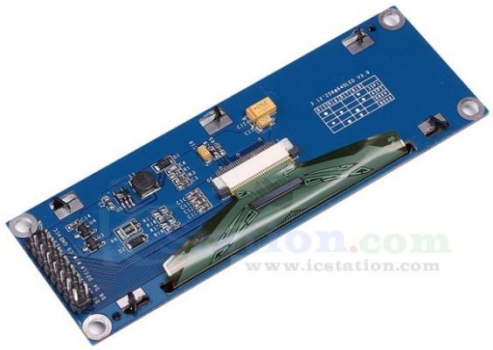
Para verificar se os seus manómetros estão calibrados, pode iniciar o simulador Falcon BMS e verificar o funcionamento dos seus manómetros. Para reconfigurar seus manómetros deve fechar o Falcon BMS.

3.2.2 - Ecrãns

Visão geral:

Não há 36.000 soluções para simular um ecrã DED no seu cockpit. Entre os mais simples de implementar, o uso de ecrãns OLED parece ser o mais interessante.

Exemplo de um ecrã OLED (256x64 px):



Exemplo de FFI e DED usando o excelente programa DEDuino: https://pit.uriba.org/tag/deduino/

O F4ToSerial permite obter o mesmo nível de resultado, exceto o efeito de rolagem dos números no FFI. Planeio desenvolver uma atualização sobre isso.

Para trabalhar com o F4ToSerial, dois formatos de ecrã são aceites: 256x24 e 128x64.

Eles estão disponíveis em todos os lugares da internet, mas o F4ToSerial usa apenas os modelos SPI mais rápidos. O I2C Bus não está implementado no programa C ++ ou no F4ToSerial.

> *Eu recomendo que preste atenção ao comprar o seu ecrã, porque no caso dos modelos I2C às vezes é necessário para dessoldar um resistor para trocar o BUS no SPI.*

Nos exemplos relacionados à minha instalação eu uso dois modelos de ecrãs:

O modelo ER-OLEDM028-1Y cuja ficha de dados está disponível aqui:

https://www.buydisplay.com/download/manual/ER-OLEDM028-1_Series_Datasheet.pdf

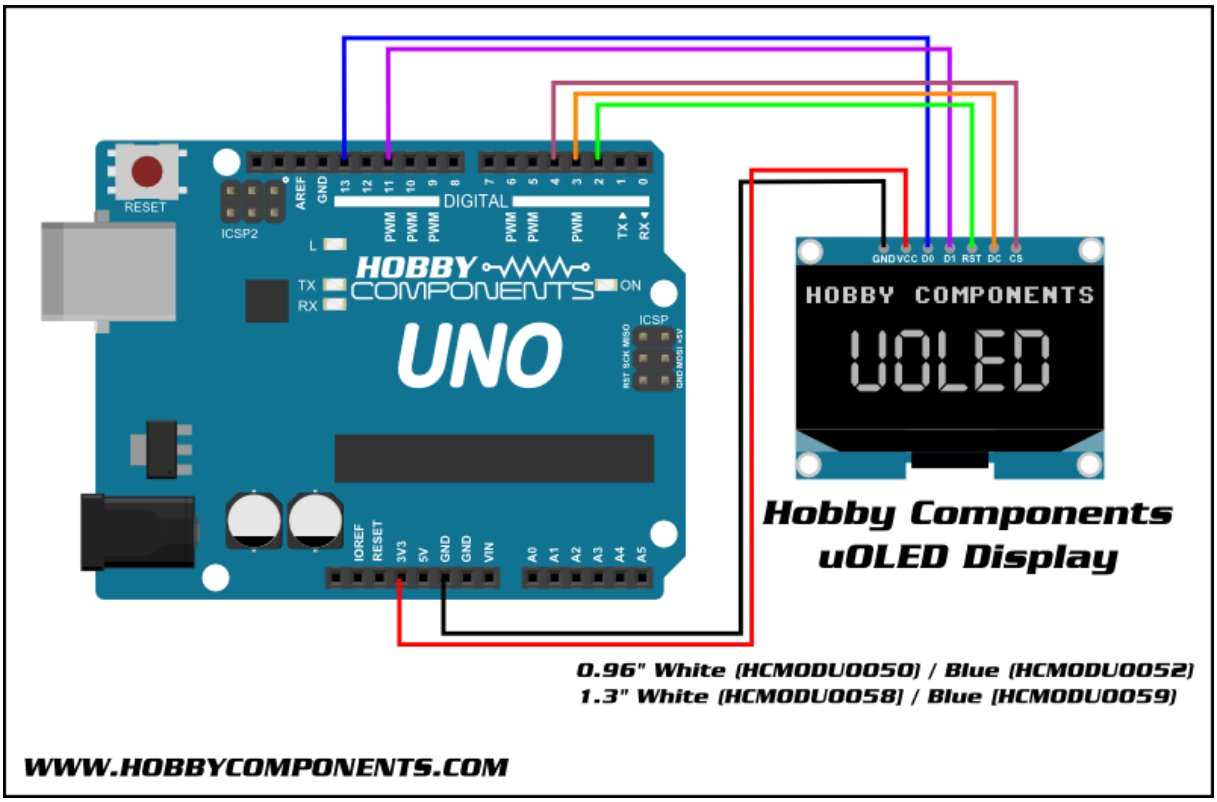
Bem como um modelo NONAME com chipset SSD1322 comprado no AliExpress aqui:

https://fr.aliexpress.com/item/Blue-Color-3-12-3-12inch-OLED-Display-Module-256x64-SPI-Communicate-SSD1322-For-Arduino-STM32/32819511393.html?spm=a2g0s.9042311.0.0.aSjjOi

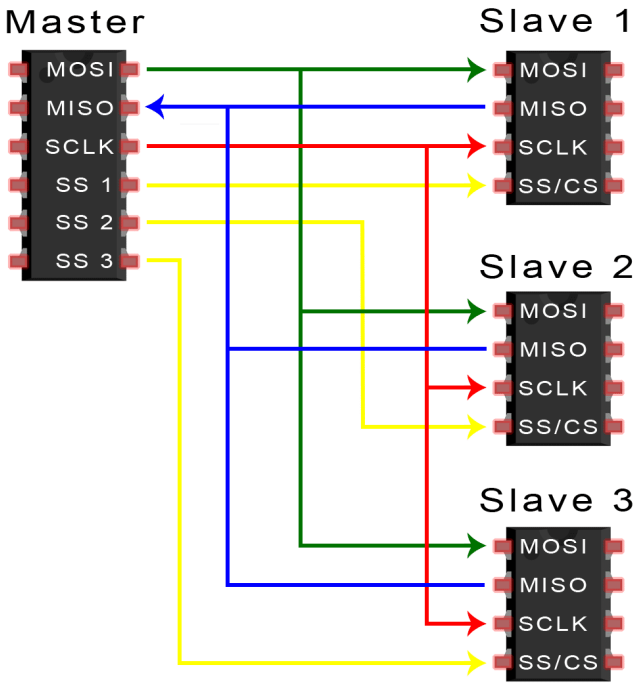
A comunicação no SPI Bus usa 5 fios:

1. Clock
2. Data
3. CS (Cable Select)
4. DC (Data Control Command)
5. Reset

Este exemplo mostra a conexão de um único ecrã de 128x64 a uma placa do Arduino Uno.

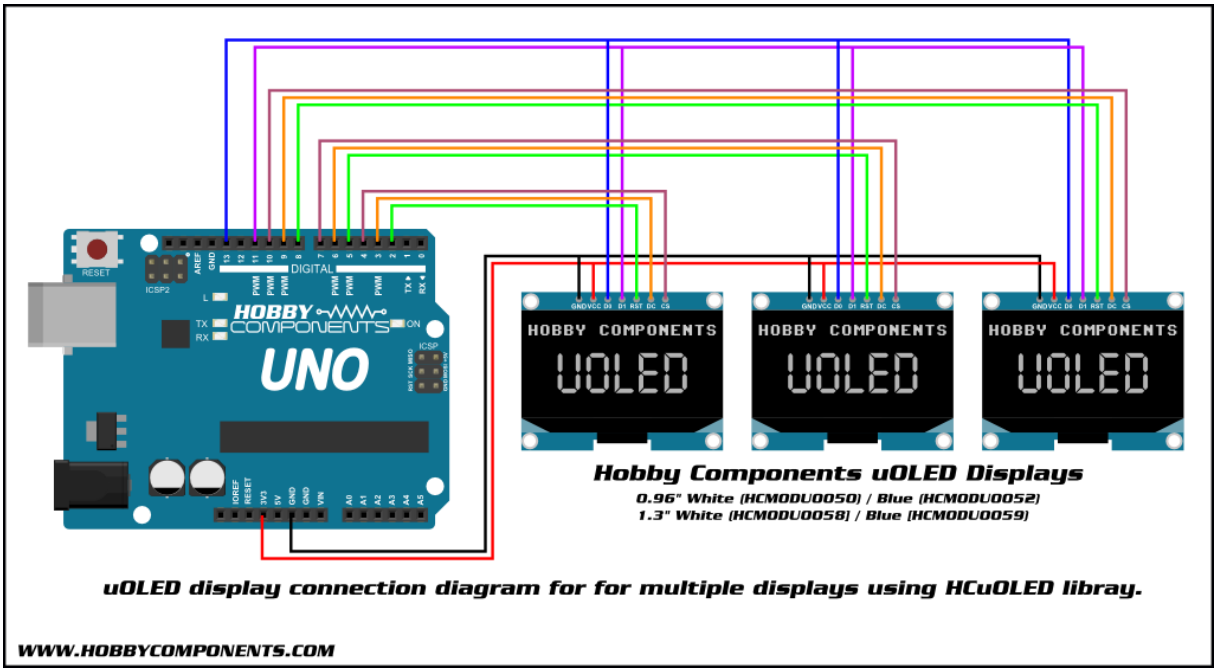


Nos casos em que vários ecrãs OLED são usados na mesma porta serial (a mesma placa Arduino), os relógios (CLOCK / DO) devem estar conectados uns aos outros, assim como os dados (DATA / M0SI / D1).



RESET, DC e CS são conectados separadamente no cartão.

Exemplo de conexão:

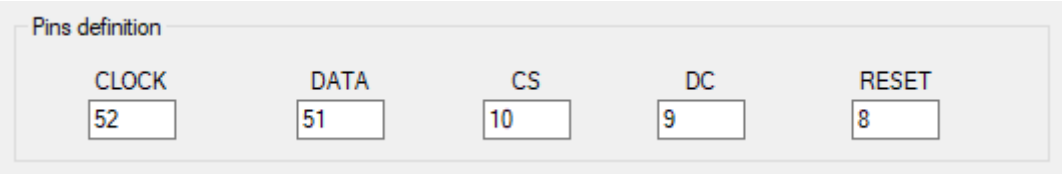


> *Existem especificações para placas Arduino. Apenas determinados pinos na placa do Arduino são fornecidos para transmitir o CLOCK e DATA. Por favor, consulte a documentação do Arduino.*
> *https://www.arduino.cc/en/Reference/SPI*

Uso no F4ToSerial:

Acessível a partir do separador "Displays" do bloco de configuração, esta opção permite configurar os três ecrãs do cockpit: DED, PFL e FFI. No Falcon BMS, o último (FFI) não é um ecrã em si. Mas a sua exibição continua sendo possível.

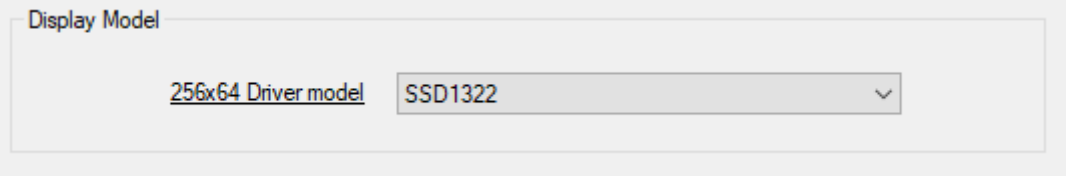
O primeiro passo é a definição da porta serial e, em seguida, os pinos do ecrã OLED.



Indique aqui os pinos da sua placa Arduino correspondentes aos pinos SPI dos seus ecrãs OLED.

Por padrão, os números de pinos CLOCK (52) e DATA (51) correspondem aos números usados em uma placa Mega Arduino.

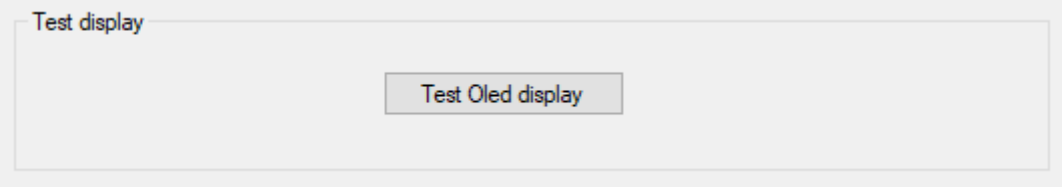
Então indique o modelo do microcontrolador do seu ecrã:



O F4ToSerial é compatível com os seguintes chipsets OLED:

- **Formato : 256x64**
  - SSD1322 (por defeito)
- **Formato : 128x64**
  - SSD11306 (por defeito)
  - SH1106
  - SSD1309
  - SSD1325
  - ST7565

Finalmente, depois de guardar a configuração e abrir as portas Seriais, é possível realizar testes para garantir que os seus monitores OLEDS funcionam corretamente. Clique no botão "Test Oled display".



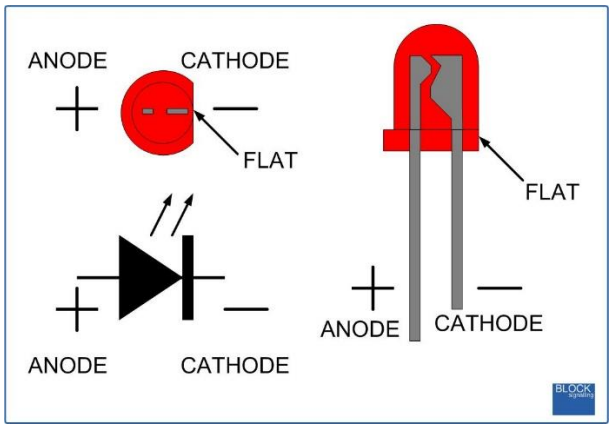
3.2.3 - Leds ou LightBits

Visão global:

O uso de LEDs ou Lightbits num cockpit é essencial para simular as muitas luzes ligadas ou desligadas no simulador Falcon BMS.

Este também é o mesmo tipo de componente usado em qualquer dispositivo real.

No simulador Falcon BMS, os LEDs são chamados de "LightBits" e o seu estado (ligado ou desligado) é alterado para "binário" (O ou 1) na Shared Memory.



Existem duas abordagens para o uso de Lightbits. Uma "económica" e outra "não quero saber".

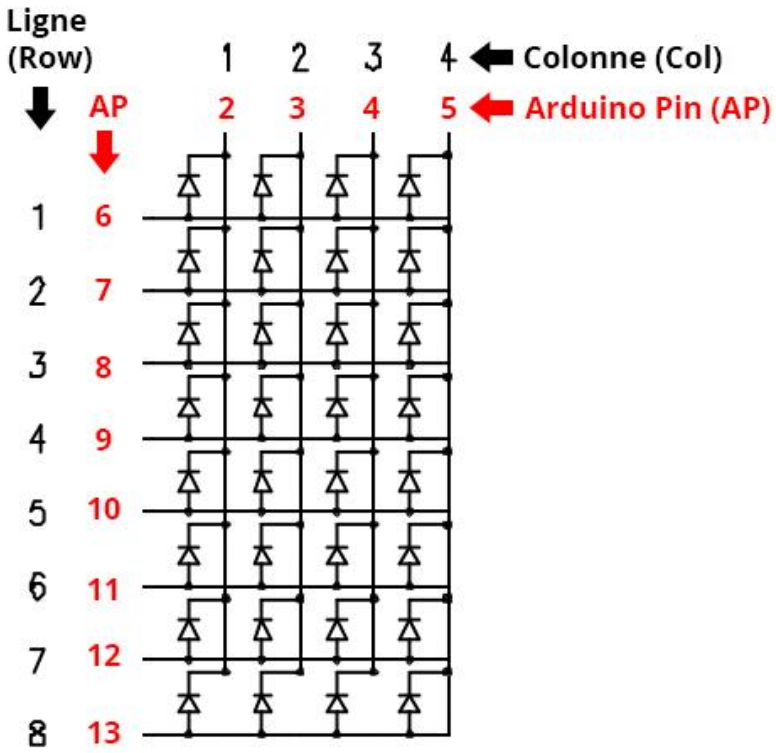
Na abordagem “não quero saber", é preciso contar 1 pino da placa Arduino para cada lightbit. É mais fácil de conectar, mas ocupa rapidamente todos os pinos da sua placa Arduino.

Na abordagem "económica", 4 pinos podem alimentar até 16 lightbits. Isto é chamado de "Matriz" ou "Matrix" em inglês.

Pessoalmente uso ambos os métodos com a aplicação de uma matriz para alguns componentes como o PFL, por exemplo.

A seguir, o PFL do Bloco F16 52.

O PFL pode ser representado com uma matriz deste tipo:



Claro que não devemos esquecer de adicionar as resistências nas entradas de cada coluna.

Eu não vou me debruçar sobre o uso e operação de uma matriz aqui, porque há tutoriais muito bons no Google procurando por "LED Matrix", por exemplo:

(Francês) https://openclassrooms.com/courses/perfectionnez-vous-dans-la-programmation-arduino/concevez-des-matrices-de-led

(Inglês) https://www.arduino.cc/en/Tutorial/RowColumnScanning

A única coisa a saber é que o uso da matriz não permite ter vários LightBits acesos ao mesmo tempo. É, portanto, inteligente iluminar vários LEDs ao mesmo tempo e, para isso, usar o efeito da persistência retiniana.

A boa notícia é que se não sabe como fazer isso, o F4ToSerial faz isso por você!

No meu caso para o uso de LEDs conectados na placa sem passar por uma matriz, utilizo uma outra placa muito prática.

http://www.flightsimparts.eu/shop_ledextension.htm

Esta placa tem 2 vantagens

1. Já tem todas as resistências
2. Ela usa transístores para alimentar os LEDS com uma fonte de alimentação externa.

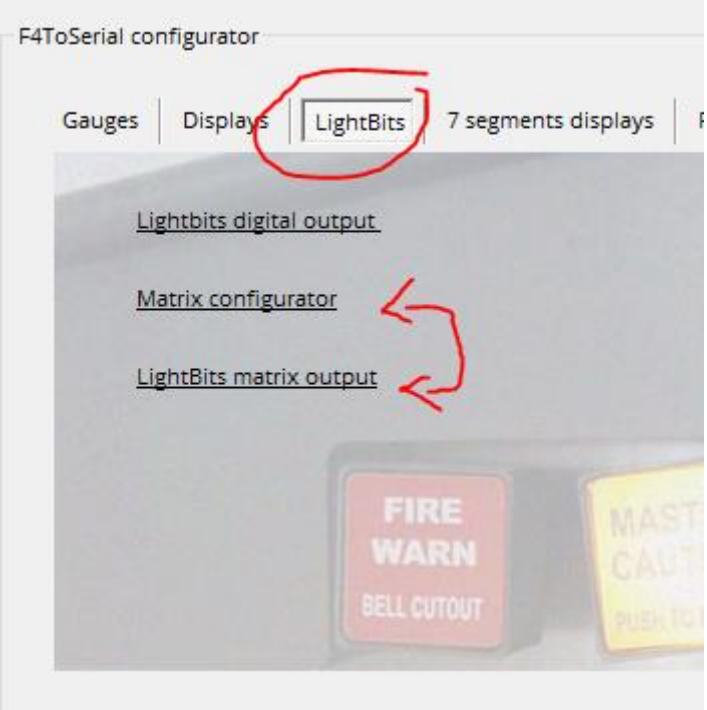
Neste último ponto, lembre-se de que quanto mais componentes você conectar (OLEDS, LEDs, motores, etc.) na sua placa Arduino, mais potência você terá! Uma placa Arduino sozinha não pode acender mais de 5 ou 6 LEDs!

Eu recomendo fortemente o uso de uma fonte de alimentação externa!

No caso de uma matriz, apenas um LED está aceso, o que consome pouca energia!

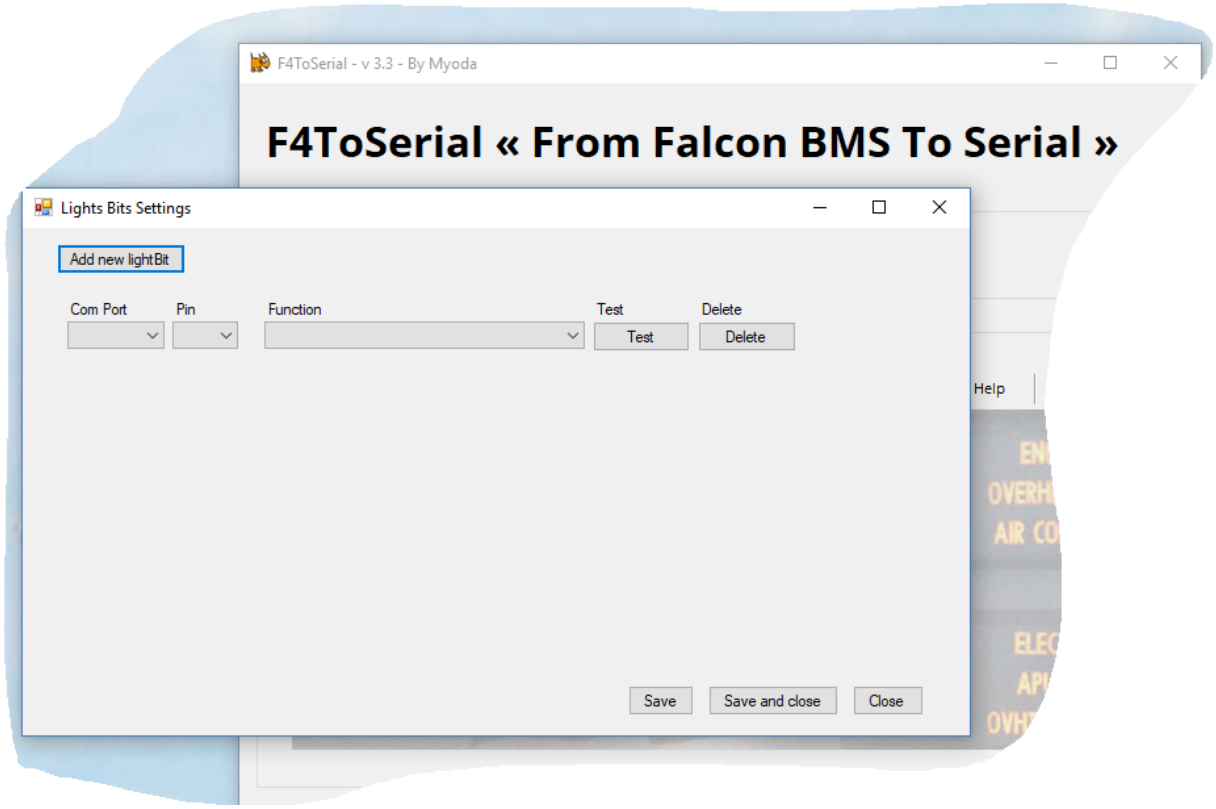
Uso no F4ToSerial:

Para configurar as LightBits no início do F4toSerial, clique no separador "LightBits"



Existem 3 links neste separador. O primeiro link "LightBits digital output" é usado para conectar os LEDs diretamente na placa. Os outros dois são usados para criar e configurar uma matriz.

<u>Criando uma LightBit diretamente na placa usando "LightBits Digital output"</u>



Clicar em "Add New LightBit" adiciona uma nova "entrada" ao nosso quadro de LightBits.

Como sempre, a porta serial (Com Port) deve primeiro ser definida.

Em seguida, indique a saída "pin" da placa Arduino correspondente ao seu LightBit.

Finalmente, defina sua função:

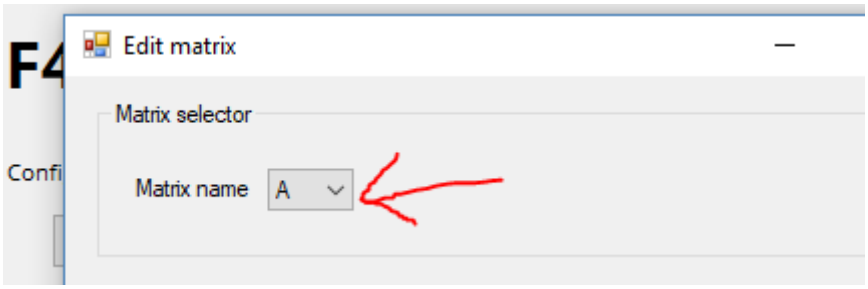
A coluna "Função" corresponde à função do lightBit no simulador Falcon BMS. Aconselho a testar para usar uma função simples como o "Hook" ou “Seat Armed” que ilumina imediatamente o PFL.

Você ainda pode fazer um teste pressionando (MAL & IND LTS) para ligar tudo.

<u>Criando um configurador Matrix LightBits e um output de matriz LightBits</u>

Clique em "Matrix configurator"

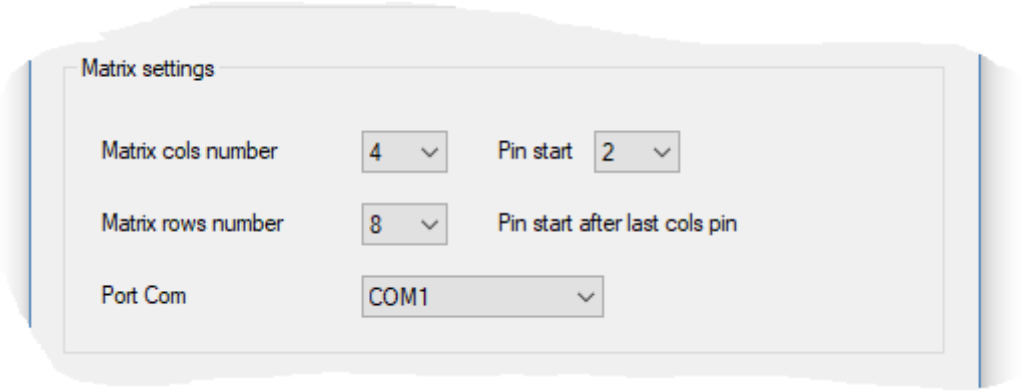
Comece por escolher o nome da sua Matriz



O F4Serial permite criar até 5 Matrizes 8x8 (8 linhas e 8 colunas).

Eles são chamados A, B, C, D ou E.

Em seguida, indique os 4 parâmetros-chave da matriz:



1. O número de colunas (número de cols da matriz)
2. O número de linhas (número de linhas da matriz)
3. A porta serial (porta Com)
4. O pino de partida (Pin Start).

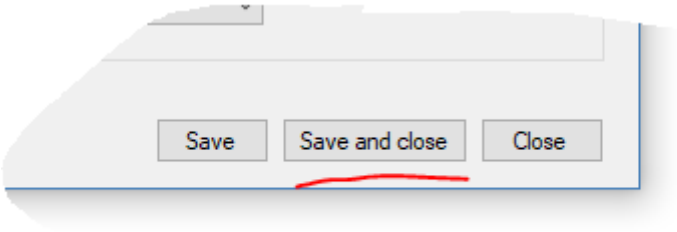
O último parâmetro (PinStart) é muito simples de entender. Ele informa a sua placa Arduino, que é o primeiro pino da sua matriz, sabendo que eles estão todos em ordem.

Você ainda precisa conectar a sua matriz na placa Arduino começando com as colunas com o programa F4ToSerial.

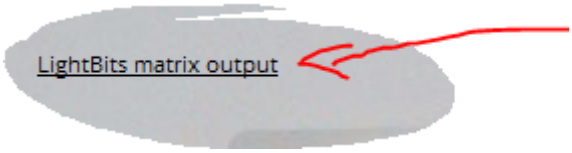
No exemplo acima, o array PFL 4x8 (4 colunas e 8 linhas) é enviado para a porta Com 1 e os pinos são conectados de 2 a 13.

2,3,4 e 5 (4 colunas) + 6,7,8,9,10,11,12 e 13 (8 linhas).

Pode agora guardar a configuração.



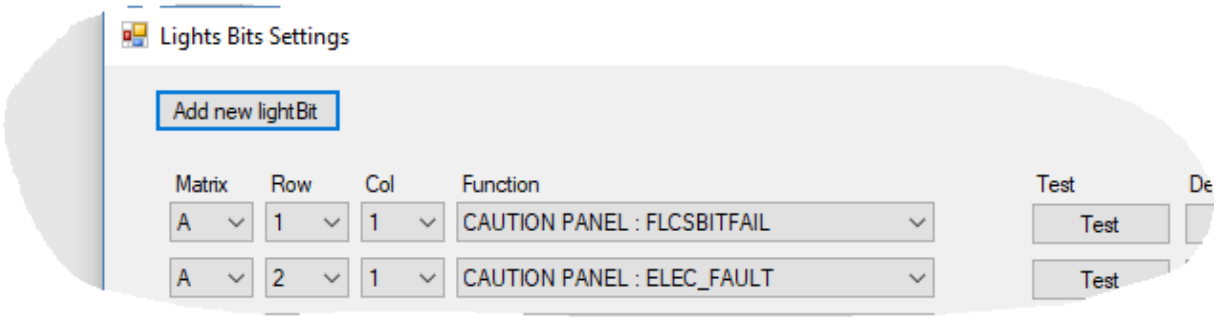
Agora você tem que definir os LightBits e as suas funções na sua matriz. Para fazer isso, clique em "LightBits matrix output" na janela principal.



O princípio é o mesmo dum lightBit conectado diretamente à placa Arduino, exceto que agora você tem que indicar:

- A Matriz
- A coluna
- A linha
- A função

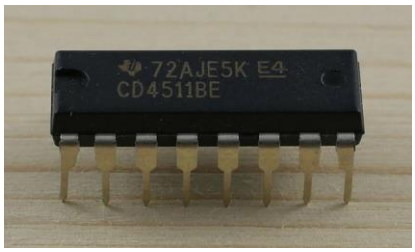
O seu lightBit está na interseção da linha e da coluna.



3.2.4 - 7 Segments displays

7 Segments displays existem há muito tempo e são usados em muitos dispositivos, incluindo o F16 Falcon Block 52.

Este componente, relativamente simples tem, no entanto, um defeito: é ganancioso na conexão! De facto, cada "LED" da ecrã é um LED completo e precisa ser alimentado para funcionar, assim como um lightBit.

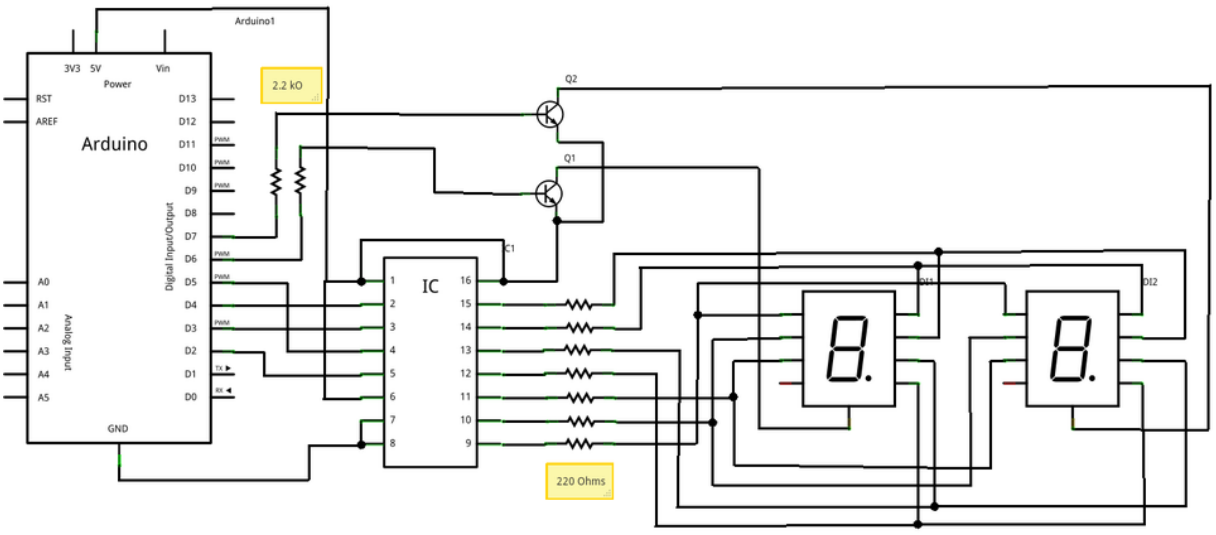


Para evitar o uso de muitos pinos na placa do Arduino, é altamente recomendável passar por um descodificador BCD.

Mais uma vez, eu não vou entrar nos detalhes da operação do descodificador BCD, porque existe uma documentação muito boa no Google com alguma pesquisa.

https://www.carnetdumaker.net/articles/utiliser-un-afficheur-7-segments-avec-une-carte-arduino-genuino/#question-piege-cathode-ou-anode-commune

A seguir, um exemplo de conexão de uma placa Arduino a um descodificador BCD e dois monitores de 7 segmentos.



Note que no exemplo acima, usamos dois transístores NPN bipolares. Eles são usados para alternar a energia da ecrã para reproduzir o efeito da persistência da retina. Porque, assim como para um lightBits, quando se afixam vários displays no mesmo descodificador BCD, apenas um pode ser usado ao mesmo tempo.

Os descodificadores BCD só podem exibir valores de 0 a 9. Convido você a ler este tutorial para entender o princípio de operação:

http://eskimon.fr/tuto-arduino-205-afficheurs-7-segments

No F4ToSerial, para usar um 7 Segments displays, este tem que passar por um descodificador BCD.

Os 7-segment displays que podem ser usados com o F4ToSerial são:

- UHF FREQ
- UHF CHAN
- FUEL (Manómetro)
- CHAFF/FLARES (Painel CMDS)

Atenção:

Para CHAFF e FLARES, quando a quantidade de cápsula de CHAFF e FLARES atinge o nível 0, a indicação "Lo" para "Low" (mínimo) aparece na frente do número de Chaff / Flares restantes.

Visível também aqui (salvo que 01 e 02 não são usados no F 16)

Infelizmente, os descodificadores BCD só podem exibir valores de 0 a 9.

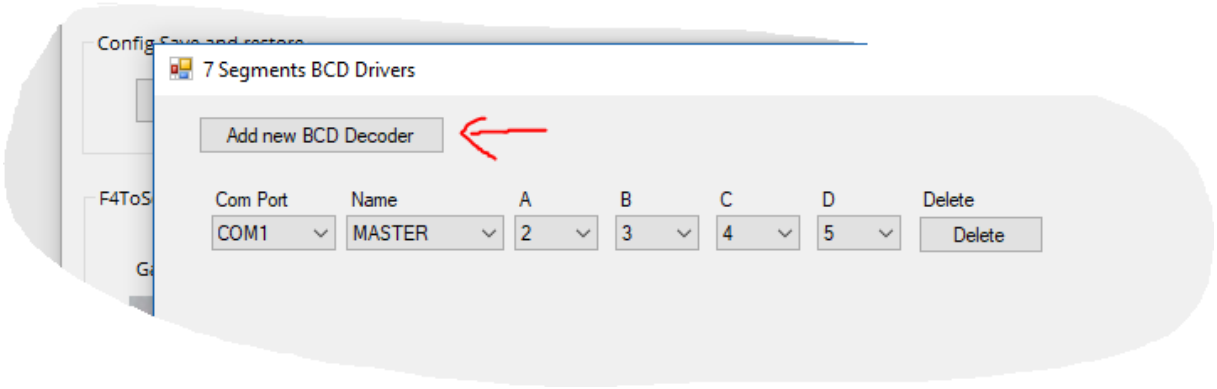
Não está previsto para o momento a atualização disto, porque não interfere com a compreensão do painel CMDS e não é uma restrição chamada "forte".

Uso no F4ToSerial:

Comece por clicar no separador "7 Segments displays ".

Em seguida, clique em "BCD DECODER" para definir e adicionar um novo descodificador BCD.

Clique no botão "Add new BCD decoder" para adicionar uma nova linha.



Para cada linha você deve definir os seguintes parâmetros:

• A porta serial (porta Com)

• O nome do descodificador (Nome)

• O pino de entrada A do descodificador BCD

• Pino de entrada B do descodificador BCD

• O pino de entrada C do descodificador BCD

• O pino de entrada D do descodificador BCD

Como os descodificadores BCD podem ser distribuídos entre várias placas Arduino, pode adicionar várias linhas e terá que escolher um nome para o seu descodificador.

O nome simplesmente define a função do descodificador. Escolhendo por exemplo "Master", você define um descodificador que terá várias funções como UHF, CAN, FLARES etc.

Este nome será usado pela placa Arduino para identificar qual descodificador receberá informações do simulador Falcon BMS.

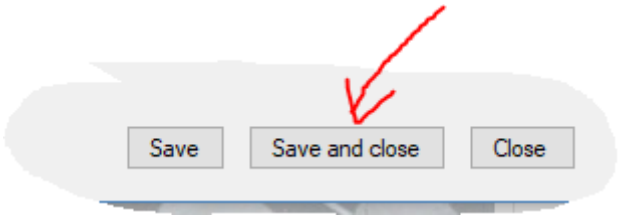
Exemplo:

Numa placa Arduino conectada à porta COM2 é usado um descodificador para os 7 Segments displays CHAFF, FLARES e FUEL. Pode então defini-lo como nome "Master".

Noutra placa Arduino na porta COM3 é usado um descodificador para os 7 Segments displays UHF CHAN e UHF FREQ, Pode definir "UHF" ou "UHFCHAN" ou "UHFFREQ" como o nome.

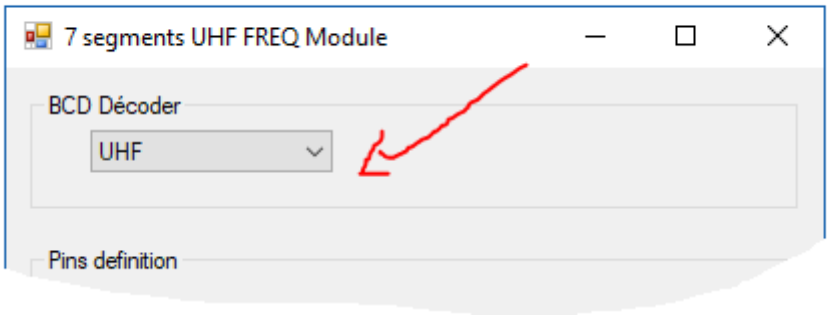
Os pinos A, B, C e D do descodificador BCD devem ser definidos e convido você a procurar a documentação do descodificador para obter mais informações, caso não entenda essa parte.

Depois de concluir as configurações e guardar a configuração:



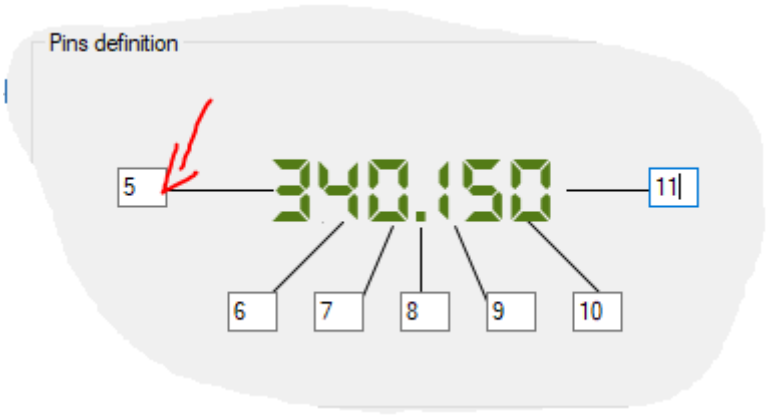
Clique no visor na janela principal.

Em seguida, selecione o descodificador BCD para esta exibição.

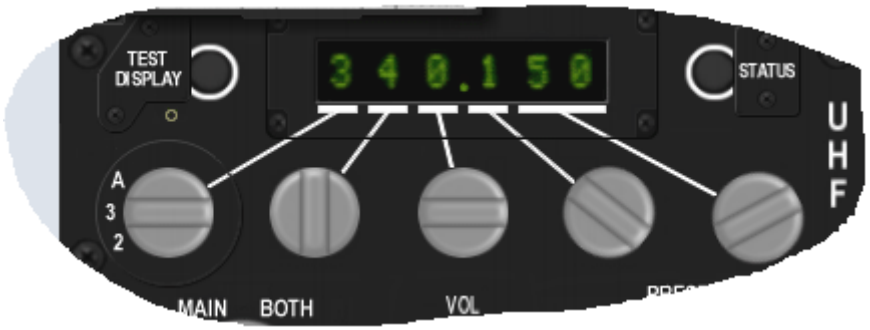


Quando o descodificador for selecionado, digite o número do pino correspondente para cada monitor (representado aqui por um número em verde).

Neste exemplo, são os 6 dígitos que compõem a FREQ UHF (7 com o ponto).

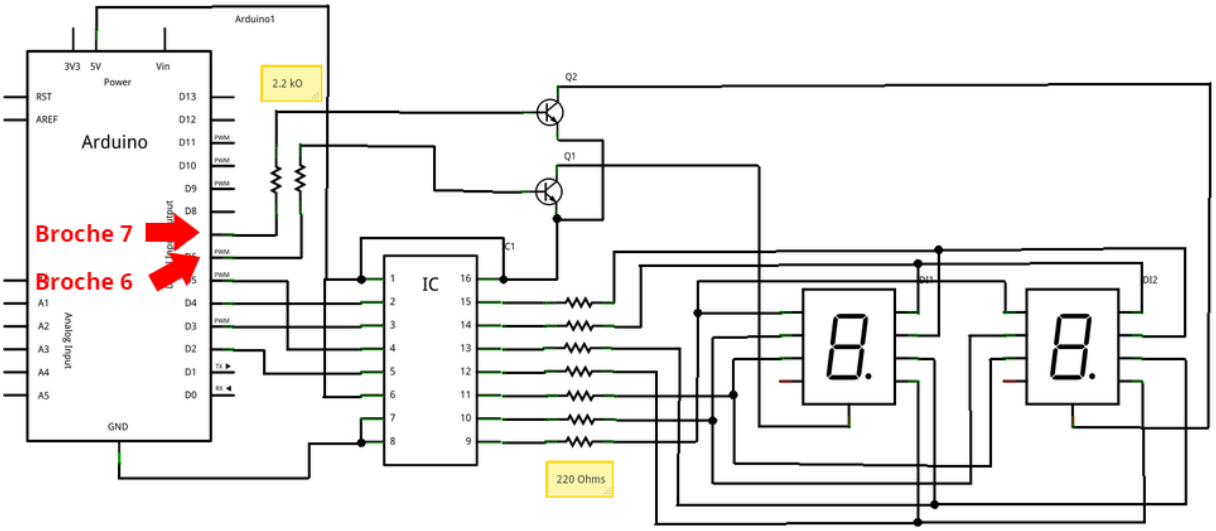


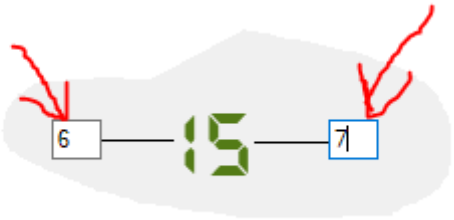
Esta janela representa o painel.



Para entender melhor como definir os pinos que permitem a escolha da ecrã, convido-o a ver este diagrama.

Exemplo:





3.3 – Definições gerais

3.3.1 - Download da versão mais recente

O download da versão mais recente é importante para manter o software atualizado.

Para isso, clique no botão "Check Version".

3.3.2 - Taxa de atualização

A taxa de atualização é o tempo decorrido entre cada novo envio de informações para a sua placa Arduino do programa F4ToSerial.

Atenção, esta velocidade é calculada e depende da velocidade de transferência (Baud Rate).

- Quanto mais você reduzir esse número, mais rápido você enviará informações para sua placa Arduino. **No entanto, arrisca a saturação do "buffer" da sua placa Arduino**.
- Quanto mais você aumentar esse número, mais tempo passa entre as informações. **O risco é observar um atraso nos elementos do seu cockpit em comparação com o Falcon BMS.**

3.3.3 - Velocidade do serial port

A velocidade da porta serial é a taxa na qual cada dado "bit" é transferido do programa F4ToSerial para a placa Arduino.

O Arduino suporta nativamente as seguintes frequências.

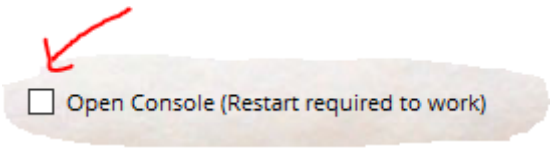
300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200

> *A minha experiência mostrou-me que a placa Arduino pode suportar mais de 115.200 bauds. Na prática, convido-o a usar uma velocidade mais alta para evitar qualquer perda de elementos ativos, incluindo exibições de OLEDS.*
>
> *Eu recomendo o uso de 1.000.000 como a velocidade mínima de transferência.*

3.3.4 – Mostrar a consola de Debug

O consola é um modo muito útil para depurar seu aplicativo. Ao usar o F4ToSerial pela primeira vez, ou quando você estiver configurando um módulo.

Exibe os dados que passam no cartão e avisa se houver algum problema no programa.



Quando clica em "Open Console" tem que reiniciar F4ToSerial.

4 Bugs e possíveis problemas

Nada acontece quando eu abro os serial ports, porquê?

Se nada acontecer quando você abrir as portas seriais e clicar nos itens dos botões de teste (lightbits, motores etc.)

Comece por procurar na consola se você tiver informações enviadas.

Se nada acontecer, provavelmente está a enviar os dados para uma placa na velocidade errada (baud). Para fazer isso, verifique se a sua placa Arduino está a comunicar na mesma velocidade que o programa F4ToSerial.

Quando eu adiciono mais de 6 LEDs ou mais de 2 manómetros, ele pára de funcionar, porquê?

Se você extrair muita energia da sua placa Arduino, ela não funcionará corretamente. Deve alimentar seus elementos com uma fonte externa.

Em geral, precisa conectar todo o seu equipamento a uma fonte externa que não seja a placa Arduino, ela própria alimentada pela porta USB do seu computador.