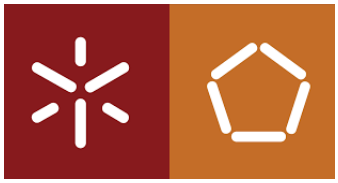

Instrumentação avançada

Relatório de desenvolvimento do Simulador de Aviação

Maria Gabriela Jordão Oliveira[1*], pg50599
Miguel Caçador Peixoto[1†], pg50657

*mgabijo@gmail.com
†miguelpeixoto457@gmail.com
[1]Mestrado em Engenharia Física, Universidade do Minho, Gualtar, 4710-057 Braga, Portugal.

Resumo

Desenvolveu-se um simulador de aviação recorrendo a um acelerómetro e a uma placa integrada com um microcontrolador PIC18F47Q10. A comunicação efetuada entre a STIM e a NCAP segue o norma standard IEEE 1451.0.
Palavras chave: *Instrumentação; Microcontrolador; Simulador;*

Conteúdo

1. Introdução

Nos dias de hoje, a utilização de sistemas ciberfísicos e o interesse nestes por parte das sociedades tem aumentado. Na sua essência, estes sistemas são projetados para interagirem com o mundo físico e realizar tarefas específicas, como automação de sistemas, o que implica o controlo e monitorização de variáveis físicas em constante alteração. As variáveis físicas, usualmente de natureza analógica, são detetadas através de sensores e traduzidas, por estes, para sinais elétricos. Estes sinais serão posteriormente processados conforme a aplicação desejada. Normalmente, recorrem-se a microcontroladores para receberem a informação dos sensores e, posteriormente, enviarem para unidades de processamento com maior capacidade de processamento. É importante não esquecer que estes sistemas também englobam atuadores, ou seja, o processo inverso, em que é a unidade de processamento que, por algum motivo, envia indicações ao microcontrolador para ativar algum atuador.

Neste projeto, no âmbito da unidade curricular de instrumentação avançada do mestrado em engenharia física da Universidade do Minho, desenvolveu-se um simulador de aviação controlado por um acelerómetro. Para tal foi necessário programar um microcontrolador do modelo PIC18F47Q10. A aplicação projetada controlará o movimento de um avião em ambiente de simulação, sendo assim necessária a obtenção de valores para a direção do movimento do objeto. Por simplicidade de utilização do simulador, definiu-se que o avião movimentar-se-á a uma velocidade constante e que a direção do movimento deste deverá ser paralela à do acelerómetro. Com isto, utilizar-se-á a informação proveniente dos 3 eixos do acelerómetro para determinar a direção de movimento do avião.

O protocolo de comunicação utilizado entre a STIM (microcontrolador) e a NCAP (unidade com maior capacidade de processamento), tal como será posteriormente explicado, foi o IEEE 1451.0.

Consequentemente, este documento encontra-se dividido em 4 secções abaixo. Primeiro, na Seção 2 o protocolo utilizado, a sua implementação e a sua utilização são explanados. De seguida, na Seção 3, os periféricos utilizados são apresentados e é explicada a sua funcionalidade no âmbito da aplicação. Finalmente, na Seção 4 o funcionamento final da aplicação e a programação necessária para a obter são apresentados.

2. Protocolo de Comunicação

No projeto desenvolvido, o microcontrolador comunica com o computador, e vice-versa, usando o protocolo standard IEEE 1451. Este define um conjunto de normas e interfaces para comunicação e interoperabilidade entre transdutores inteligentes (sensores e atuadores) e um sistema de controle central (como por exemplo, um computador). É composto por três componentes principais:

- Network Capable Application Processor (NCAP): A NCAP é um dispositivo que atua como uma ponte entre os transdutores inteligentes e o sistema de controle central. É responsável por gerir a comunicação entre os dois e converter os dados dos transdutores, enviados pela STIM, num formato que possa ser entendido pelo sistema de controlo.
- Smart Transducer Interface Module (STIM): A STIM é um dispositivo conectado ao transdutores inteligentes e é responsável por converter a saída destes (por exemplo, temperatura, pressão, etc.) num formato digital que possa ser transmitido para a NCAP.
- Transducer Electronic Data Sheet (TEDS): TEDS é um formato de dados padronizado que contém informações sobre cada transdutor, qual é o seu tipo, alcance, precisão e outras características relevantes. Estas informações são armazenadas na STIM e podem ser requisitadas pela NCAP quando necessário.

Em geral, o standard IEEE 1451 fornece uma estrutura para integrar transdutores inteligentes num sistema de controlo, permitindo que o sistema comunique e controle os transdutores de maneira consistente, confiável e, mais importante, padronizada.

Depois desta breve introdução, explica-se o funcionamento do protocolo e como este é implementado.

2.1. Funcionamento da Comunicação

O protocolo é usado para comunicar entre o computador, que contém o simulador de aviação, e o microcontrolador, que acede diretamente a todos os sensores e atuadores, como ilustra a figura Figura 1.

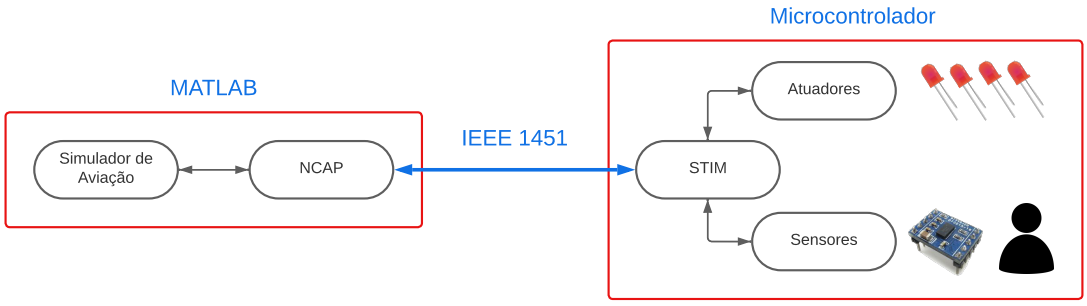

Figura 1: Esquema exemplificativo do funcionamento do protocolo IEEE 1451 na aplicação em questão.

De modo a que o simulador de aviação consiga obter os dados da direção do avião, a partir do acelerómetro, e mostrar o seu estado, este tomará partido da NCAP que irá requisitar dados e controlar atuadores, através da comunicação com a STIM. Cada pedido de comunicação é feito pela NCAP à STIM cai numa destas situações:

- Pedido de MetaTEDS: Pedir informações gerais sobre o microcontrolador, canais disponíveis, versão, entre outros.
- Pedido de TransducerChannel TEDS: Pedir informações gerais sobre um determinado canal, como que unidades este mede/aplica, limites operacionais, versão, entre outros.
- Pedir para ler dados de um sensor (canal).
- Pedir para alterar o estado de um atuador (canal).

Por sua vez a STIM, implementada no microcontrolador, irá responder com a flag de sucesso ou falha de comunicação, o comprimento da mensagem de dados, seguido pelos dados requisitados (em caso de sucesso). No caso de insucesso a mensagem enviada é 00 00 00, uma vez que se falhou (00) e portanto o comprimento da mensagem é 00 00.

Nas subsecções abaixo explana-se como definir as MetaTEDS e as TEDS, qual a estrutura correta das mensagens e como fazer um pedido de (Meta)TEDS, leitura ou escrita num canal e respetiva resposta por parte da STIM.

2.2. Meta TEDS

Meta TEDS é uma TEDS única e obrigatória para o bom funcionamento do protocolo. Esta torna disponível para a interface toda a informação necessária para se ter acesso a qualquer canal e informação comum a todos os canais. Para a correta implementação da Meta TEDS, no caso concreto da aplicação projetada, é necessário definir 3 campos, Tabela 1,: ID da TEDS, que representa a estrutura do bloco da Meta-TEDS; o UUID, o ID único do dispositivo; e o número de canais máximo.

Tipo de Campo	**Descrição**	**Dados (TLV)**
3	Identificador de TEDS	03 04 00 01 00 01
4	UUID	04 10 00 00 00 00 00 00 00 00 00 01
13	Canais Máximos	13 02 00 07

Tabela 1: Definição da MetaTEDS.

Cada campo da MetaTEDS é escrito segundo o formato *type*, *length*, *value* (TLV). Analisando cada campo da MetaTEDS em particular, a explicação do campo identificador de TEDS (ID) encontra-se na Tabela 2, o campo UUID é arbitrário pois representa o ID único do dispositivo e o número de canais máximos é definido de acordo com a Tabela 3. O tipo de cada campo está de acordo com a norma [1].

	Dados	Descrição
T	03	O bloco é do tipo "Identificador de TEDS"
L	04	4 bytes no campo de valores
V	00	Familia, predeterminado como sendo 00
	01	Class é 1 (MetaTEDS)
	00	Versão é 00 (Protótipo)
	01	Comprimento do tuplo é 01 porque todos os campos de comprimento de todos os blocos de dados terão 1 byte/octeto

Tabela 2: Tabela descritiva do campo TEDS ID, identificador de TEDS.

	Dados	Descrição
T	13	O bloco é do tipo "Canais Máximos"
L	04	2 bytes no campo de valores
V	00	O número de canais é 7 (3 sensores e 4 atuadores)
	07	

Tabela 3: Tabela explicativa do campo número de canais máximos.

2.3. TransducerChannel TEDS

Cada canal da STIM necessita de ter uma TEDS definida e são necessários tantos canais da STIM como sensores e atuadores utilizados. No caso da aplicação projetada são utilizados um acelerómetro de 3 eixos e 4 leds, assim é preciso definir 7 TEDS, uma para cada canal. A estrutura tipo de uma TED encontra-se na Tabela 4.

Tipo de Campo	Descrição	Tipo de dados	Número de Bytes no campo de valor
3	Identificador de TEDS	UINT8	4
11	Tipo do Canal	UINT8	1
12	Unidades	UINT8	10
13	Limite inferior de alcance	UINT8	1
14	Limite superior de alcance	UINT8	1
40	Modelo de Dados	UINT8	1
41	Tamanho dos Dados	UINT8	1
42	Bits significativos do tamanho dos Dados	UINT8	1

Tabela 4: Tabela exemplificativa da estrutura das TEDS.

Tendo em conta a tabela acima, pode-se então definir a TEDS para cada um dos canais de acordo com as suas especificações, na Tabela 5 está exemplificada a TEDS para o canal 1, isto é eixo dos xx do acelerómetro. No campo valor do indentificador de TEDS coloca-se 03 00 01, uma vez que a classe do identificador de TEDS é 03 (TransducerChannel TEDS) e 01 uma vez que o tamanho de todos os blocos de dados será 1 byte. O valor do tipo de canal varia entre 0, para sensores, e 1, para atuadores. O campo de unidades é definido de acordo com a Tabela 6, sendo para este exemplo m/s^2 e no caso dos leds V. O limite de operação é entre 0 e 255, tal como descrito nos campos de limites inferior e superior de alcance, no caso dos sensores e 0 e 1 no caso dos atuadores. O modelo de dados usado neste projeto é UInt8 que corresponde a 00 [1]. O valor do tamanho de dados é 1, pois apenas se deseja enviar 1 byte por cada valor fornecido pelo sensor. Finalmente, visto que o sensor envia sempre 1 byte de dados, o numero de bits significativos é 8.

De forma similar, alterando apenas os campos anteriomente referidos, isto é, tipo de canal, unidades e alcance, definiram-se também as restantes 6 TEDs.

Tipo de Campo	Descrição	Dados (TLV)
3	Identificador de TEDS	03 04 00 03 00 01
11	Tipo do Canal	11 01 00
12	Unidades	12 10 0 128 128 130 128 124 128 128 128 128
13	Limite inferior de alcance	13 01 00
14	Limite superior de alcance	14 01 255
40	Modelo de Dados	40 01 00
41	Tamanho dos Dados	42 01 01
42	Bits significativos do tamanho dos Dados	42 01 08

Tabela 5: TEDS do sensor do canal 1 do acelerómetro (eixo do xx).

Field	Description	Data Type	Number of octets
2	(2*<exponent of radians>) + 128	UInt8	1
3	(2*<exponent of steradians>) + 128	UInt8	1
4	(2*<exponent of meters>) + 128	UInt8	1
5	(2*<exponent of kilograms>) + 128	UInt8	1
6	(2*<exponent of seconds>) + 128	UInt8	1
7	(2*<exponent of amperes>) + 128	UInt8	1
8	(2*<exponent of kelvins>) + 128	UInt8	1
9	(2*<exponent of moles>) + 128	UInt8	1
10	(2*<exponent of candelas>) + 128	UInt8	1

Tabela 6: Tabela explicativa do campo unidades.

2.4. Estrutura das Mensagens

Segundo o protocolo utilizado, cada mensagem de comando enviada à STIM pela NCAP segue a estrutura da Tabela 7.

Octeto	Descrição
1	Canal do Transdutor de destino (MSB)
2	Canal do Transdutor de destino (LSB)
3	Classe do Comando
4	Função do Comando
5	Tamanho (MSB)
6	Tamanho (LSB)
7	Octetos dependentes do Comando

Tabela 7: Estrutura das mensagens enviadas à STIM pela NCAP.

Tendo em conta esta estrutura das mensagens, nas subsecções que se seguem explica-se qual a mensagem que a NCAP deve enviar à STIM para pedir a Meta TEDs, TEDS, ler ou escrever num determinado canal e as respetivas mensagens de sucesso com que a STIM deve responder.

2.4.1. Pedido de Meta TEDS

Seguindo a estrutura da tabela Tabela 7, os campos correspondentes aos canais do transdutor são arbitrários pois não se está interessado na TEDS de um canal específico. A classe do comando é colocada a 1 pois é a classe comum para a STIM e transdutor, o octeto 4 tem o valor 2 pois quer-se ler o segmento das TEDS e finalmente especificam-se os octetos dependentes do comando como sendo 01, uma vez que se quer ler a Meta TEDS, e 00, porque não se deseja nenhum offset. Esta mensagem está exemplificada na Tabela 8.

Campo	Valor
Canal do Transdutor de destino (MSB)	xx
Canal do Transdutor de destino (LSB)	xx
Classe do Comando	01
Função do Comando	02
Tamanho (MSB)	00
Tamanho (LSB)	02
Octetos dependentes do Comando	01 00

Tabela 8: Tabela exmplificativa da mensagem de pedido da Meta TEDS. O valor xx em alguns campos representa que o valor desse campo é arbitrário.

Quando a STIM recebe uma mensagem com este formato, deverá responder com uma mensagem de sucesso seguida do envio da MetaTEDS, esta resposta segue o formato TLV. Assim, deverá ser enviado 01 00 22 "identificador de TEDS" UUID "canais máximos", dissecando esta mensagem:

- 01 - mensagem de sucesso;
- 00 22 - tamanho do campo valor, ou seja, tamanho da Meta TEDS;
- "identificador de TEDS" UUID "canais máximos" - Meta TEDS segundo o definido na Tabela 1.

2.4.2. Pedido de TransducerChannel TEDS

Para pedir uma TEDS de um canal é usada uma mensagem muito similar à anterior. Neste caso, os campos relativos ao canal do transdutor de destino não são arbitrários, aqui coloca-se o número do canal da TEDS que se prentende pedir. Por exemplo, se se quiser pedir a TEDS do canal 1, o primeiro octeto deve conter o valor 00 e o segundo 01. Para além disto, os octetos dependentes do comando também mudam, sendo o primeiro 03, uma vez que se deseja a TEDS do canal de um transdutor, e 00 (offset de 0) como se mostra na Tabela 9 para o caso concreto do canal 1.

Campo	Valor
Canal do Transdutor de destino (MSB)	00
Canal do Transdutor de destino (LSB)	01
Classe do Comando	01
Função do Comando	02
Tamanho (MSB)	00
Tamanho (LSB)	02
Octetos dependentes do Comando	03 00

Tabela 9: Tabela exemplificativa da mensagem para pedir TEDS. Neste exemplo solicita-se a TEDS do canal 1.

Novamente, quando a STIM recebe uma mensagem com este formato, deverá responder com uma mensagem de sucesso seguida do envio da TEDS solicitada, esta resposta segue também o formato TLV. Assim, deverá ser enviado 01 00 36 TEDS, neste caso o tamanho é 36 uma vez que é o tamanho das TEDS.

2.4.3. Pedido de leitura de um canal

Para se ler o valor do sensor associado a um determinado canal é necessário que a NCAP mande a mensagem adequada à STIM. Esta mensagem encontra-se exemplificada para o pedido concreto de leitura do canal 1 na Tabela 10. Por palavras, coloca-se os valores para o canal do transdutor de destino como sendo o canal desejado, neste exemplo 1, a classe do comando é colocada a 03 pois é o transdutor está no estado ativo, a função é 1 pois quer-se ler o segmento de dados e os octetos dependentes do comando são 00, uma vez que não se deseja nenhum offset.

Campo	Valor
Canal do Transdutor de destino (MSB)	00
Canal do Transdutor de destino (LSB)	01
Classe do Comando	03
Função do Comando	01
Tamanho (MSB)	00
Tamanho (LSB)	01
Octetos dependentes do Comando	00

Tabela 10: Tabela exemplificativa da mensagem para pedir a leitura de um canal (1).

A mensagem de sucesso por parte da STIM é novamente similar as anteriores. Neste caso seria 01 00 01 "valor do canal", mais uma vez sucesso, tamanho da mensagem 1 byte e o byte lido, porque se definiu que apenas se lia e enviava um bit de cada vez.

2.4.4. Pedido para mudar o estado de um atuador

A mensagem enviada pela NCAP para escrever no canal de um atuador, ou seja, mudar o estado deste, é em tudo semelhante à mensagem para ler o valor de um certo canal.

Seguindo a estrutura do comando anterior, deve-se alterar a função do comando para 02, pois quer-se definir o seu estado como estando ligado, o tamanho do valor passa a ser 02 e, finalmente, especifica-se os octetos dependentes do comando como sendo 00 (offset de 0) e o valor que se deseja colocar no canal. Por exemplo, caso se deseje ligar um led, deve-se colocar 00 01 nos octetos dependentes do comando, este comando encontra-se exemplificado na Tabela 11.

Campo	Valor
Canal do Transdutor de destino (MSB)	00
Canal do Transdutor de destino (LSB)	04
Classe do Comando	03
Função do Comando	02
Tamanho (MSB)	00
Tamanho (LSB)	02
Octetos dependentes do Comando	00 01

Tabela 11: Mensagem exemplificativa para pedir a alteração do estado de um atuador. Neste caso, ligar o atuador do canal 4.

A mensagem de sucesso por parte da STIM seria simplesmente 01 00 00.

3. Periféricos

No desenvolvimento deste sistema teve que se recorrer a periféricos. A montagem utilizada incluindo todos os periféricos e microcontrolador encontra-se na imagem abaixo.

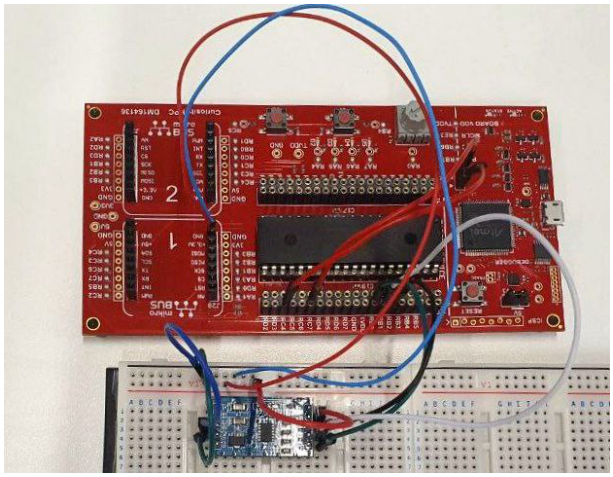

Figura 2: Montagem do acelerómetro e placa de desenvolvimento.

Recorreu-se a um sensor, atuadores, à UART e ao ADC, como se passa a explicar.

O sensor utilizado é um acelerómetro que se encontra num circuito integrado do modelo MMA7361L, Figura 3. Os valores nos canais deste indicam a direção do movimento do avião em ambiente de simulação. Por outras palavras, os valores medidos em cada canal indicam se o avião se movimenta no sentido positivo ou negativo de cada um dos eixos (xx, yy e zz).

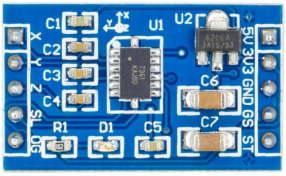

Figura 3: Acelerómetro do modelo MMA7361L.

Tendo em conta a aplicação projetada, alimentou-se o acelerómetro com $3.3V$ e colocou-se a porta g-select (GS) a $0V$. Assim, os valores de saída de cada eixo/canal variam entre $1.485V$ e $1.815V$, sendo a meia escala a $1.65V$. Como os valores obtidos à saída do ADC são sempre positivos, o sentido negativo de um eixo é definido como a primeira metade da escala e o positivo a segunda metade desta, sendo o $a = 0m/s^2$ o meio da escala.

Os atuadores utilizados são os 4 leds presentes na placa em que o microcontrolador se encontra integrado. Estes são utilizados como sinal indicativo da calibração do avião, para que o utilizador deixe permanecer o acelerómetro em repouso enquanto estes estejam acesos.

Tal como referido anteriormente, também se recorreu ao conversor analógico-digital (**ADC**) e ao recetor-transmissor assíncrono universal (**UART**).

- **ADC:** Este periférico será usado, como o nome indica, para converter sinais analógicos em sinais digitais, de tal modo que estes possam ser processados pelo microcontrolador. O ADC converterá os sinais provenientes do acelerómetro, usando um multiplexer para alternar entre os canias deste.

- **UART:** Este periférico é utilizado para fazer as comunicações entre o microcontrolador e o computador conectado via cabo micro-USB ao circuito integrado.

4. Funcionamento da Aplicação

Tal como anteriormente foi referido, elaborou-se uma aplicação onde um avião em ambiente de simulação tem a sua direção controlada pela inclinação de um acelerómetro. Para se construir esta aplicação, foi necessário programar quer o microcontrolador quer um ficheiro matlab de modo a ter o ambiente de simulação.

4.1. Simulador

No que diz respeito ao simulador, o objetivo é mostrar no ecrã do computador do utilizador um avião a voar a uma velocidade constante, sendo a direção do avião controlada pelo acelerómetro. Assim, o simulador a cada iteração do simulador, tomando partido da NCAP, requisita à STIM a leitura dos valores de cada eixo do acelerómetro, tal como descrito na Subsubseção 2.4.3. Por outras palavras, a cada iteração do simulador requer-se a leitura dos canais de 1 a 3 (xx, yy e zz) de modo a que a direção do sensor seja a mesma que a direção do avião.

No entanto, tendo em conta a natureza e morfologia dos valores lidos, no início do programa, antes de se iniciar o ambiente de simulação, é necessária uma fase de calibração. Nesta fase, com o acelerómetro em repouso, obtém-se os valores médios de cada eixo para posteriormente os valores lido poderem ser escalados. Todo este processo será explicado mais detalhadamente ainda nesta secção. O pseudocódigo geral para o simulador encontra-se a baixo.

Algorithm 1 Simulador de Aviação

$simulador \leftarrow iniciar\_simulador()$
$x\_medio, y\_medio, z\_medio \leftarrow calibrar()$
while $True$ **do**
    $x, y, z \leftarrow pedir\_valores()$
    $x, y, z \leftarrow processar\_valores(x, y, z)$
    $atualizar\_avião()$
end while

A fase de calibração consiste essencialmente em dois ciclos while. O primeiro ciclo tem como objetivo descartar as primeiras 50 leituras dos canais de modo a não existirem eventuais flutuações iniciais, pode ser considerado um ciclo de precaução. No segundo ciclo, que pode pode ser considerado a calibração no sentido literal, são pedidos 50 valores de cada eixo do acelerómetro e fazem-se as suas médias, guardando estes valores nas variáveis x_médio, y_médio, z_médio. Durante a fase de calibração os leds ligam e desligam segundo um padrão definido e quando se desligarem todos os leds definitivamente esta acaba.

Por sua vez, já após se iniciar o ambiente de simulação, o processamento dos valores centra-os em zero, e.g $x = x - x\_medio$, arredonda-os à dezena mais próxima, e.g $x = round(x/10)$, e, finalmente, converte-os em ângulos, e.g $x = x * \pi/13$. Estes valores são usados para definir a direção do avião, através de uma matriz de rotação. Note que sendo que o valor, de cada um dos canais, recebido pela NCAP varia de 0 a 255, e no pré-processamento é retirado o valor médio, ou seja, o valor varia de -128 a 128[1], e como se arredonda de seguida à dezena mais próximo pelo método explicado, apenas se podem obter valores de -13 a 13, com isto, definindo que os ângulos variam de $-\pi$ a π, os valores medidos podem ser traduzidos em ângulos de rotação pelo fórmula exposta.

Note que adicionalmente também foi aumentada a resiliência do código da aplicação no pedido de leitura de sensores, no que diz respeito a falhas de comunicação, através da identificação da flag de sucesso/falha de comunicação da resposta do PIC: Se esta tiver falhado durante a fase de calibração, a aplicação irá requisitar novamente os valores até ser bem sucedido; e se tiver falhado durante a simulação, este irá manter o valor anterior do eixo onde falhou a medida e tentar novamente na iteração seguinte.

O funcionamento do programa em matlab encontra-se esquematizado no fluxograma da Figura 4, o ambiente de simulação e o respetivo avião encontra-se na Figura 5.

[1]Com a exceção do canal 3 (eixo dos zz) que lê por defeito 1g. No entanto, foi experimentalmente verificado que este processamento, tal e qual como explicado, funciona perfeitamente neste eixo também. Assim, os todos os canais do acelerómetro são tratados de igual forma.

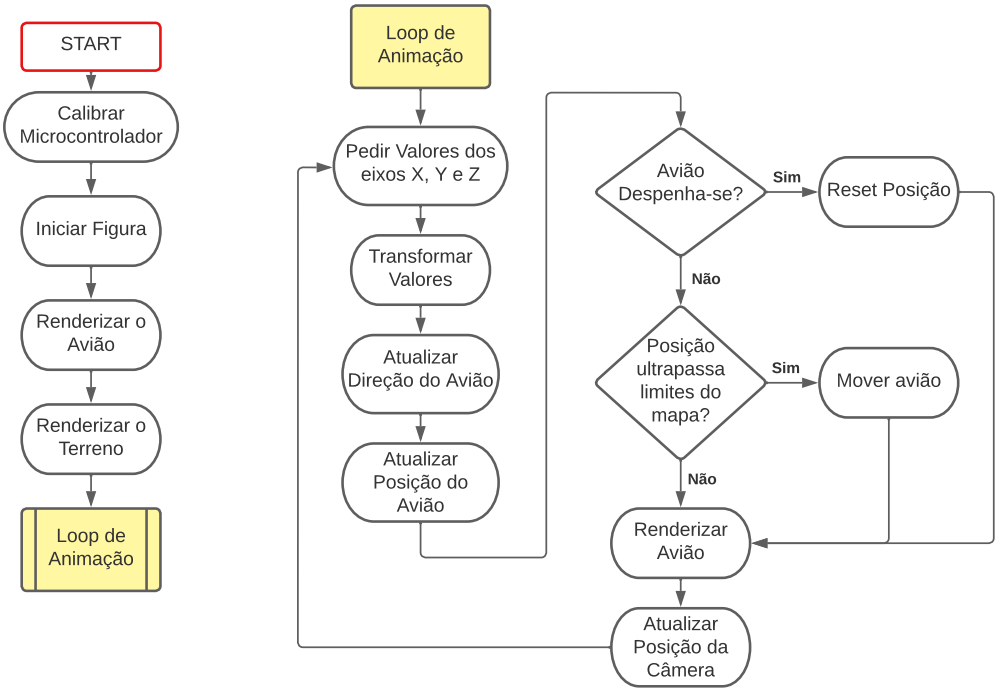

Figura 4: Fluxograma exemplificativo do funcionamento do simulador.

Figura 5: Imagem do avião e cenário de fundo no simulador de aviação.

Acrescente-se que o este simulador encontra-se dentro de uma aplicação construída também em matlab, esta contém as instruções de utilização e um botão para iniciar o simulador.

4.1.1. Frequência de amostragem

O simulador desenvolvido em matlab para uma correta e fluída apresentação das imagens, atualiza-as a uma taxa de cerca de 20Hz, assim os valores de cada canal do acelerómetro precisam de ser atualizados no mínimo a esta taxa. Como se tem 3 canais, a frequência de amostragem é 3 vezes esta, ou seja, 60Hz e é garantida pelos pedidos de valores dos canais por parte da NCAP.

4.2. Microcontrolador

No que diz respeito ao microcontrolador explicar-se-á abaixo as configurações de registos e periféricos utilizadas e as principais funçóes utilizadas, ou seja, a main e a função que é chamada na resposta à interrupção do ADC. Nesta secção também se explicará a média móvel, uma vez que os dados enviados para a NCAP já são pré-processados, ou seja, já se envia a média móvel destes.

4.2.1. Configuração de Registos e Periféricos

Para a programação desejada do microcontrolador foi necessário configurar registos e periféricos. As configuração realizadas encontram-se explanadas nas subsecções abaixo.

4.2.1.1 Configuração do Clock

Fundamentalmente é necessário configurar o clock do microcontrolador, para isso acedeu-se aos seguintes registos:

- **OSCCON1:** Onde se colocou 0b01100000, ou seja, definiu-se que a fonte do clock seria HFINTOSC e que o divisor do clock seria 1;
- **OSCFRQ:** Onde se colocou 0b0000110, isto é, definiu-se o clock como 32MHz;
- **OSCEN:** Onde se colocou 0b01000000 para ativar o clock HFINTOSC.

Escolheu-se usar o clock com frequência de 32MHz, uma vez que a aplicação projetada é relativamente lenta, executa com uma frequência de amostragem de 60Hz, e assim com este clock é conseguida uma correta execução da mesma.

4.2.1.2 Configuração das Portas do Microcontrolador

Foi necessário configurar as entradas analógicas para a medição dos sinais provenientes dos sensores, acelerómetro, e digitais para a escrita nos canais associados aos atuadores, leds.

Para o efeito, consideram-se as portas RB0, RB1 e RB2 para cada um dos canais do acelerómetro, cada uma corresponde a um eixo, xx, yy e zz respetivamente. Para os 4 LEDS foram consideradas as portas RA4, RA5, RA6 e RA7, tal como já está nativamente definida na placa do circuito integrado.

Configuraram-se também as portas responsáveis pela transmissão e receção de informação, isto é, configurou-se a porta RC4 para transmissão (TX) e a porta RC7 para receção (RX).

Todas as portas foram configuradas usando registos do microcontrolador, tal como se ilustra nos seguintes pontos:

- Foram limpos quaisquer valores dos registos dos valores dos portos - Registos LATA, LATB e LATC colocados a 0;
- Foi definida a direção de I/O dos pins relevantes usando os registos TRISA (colocou-se a 0 (OUT)), TRISB (colocaram-se os primeiros 3 bits a 1 (IN) e os restantes a 0 (OUT)) e TRISC (colocou-se 1 o bit 7 (IN) e os restantes a 0 (OUT));
- Foi definido se os pinos eram digitais ou analógicos, para o efeito colocou-se o registo ANSELA a 0 (Digital), os primeiros 3 bits do registo ANSELB a 1 e os restantes a 0 (RB0, RB1 e RB2 como analógico e os restantes como digitais) e definiram-se todos os bits do registo ANSELC a 0 (todos as portas são digitais).

4.2.1.3 Configuração do UART

Configuração dos pinos TX/RX

Usando o módulo *Peripheral Pin Select* (PPS) foram selecionados os pinos para a transmissão (TX) e receção (RX) de comunicações pelo PIC. O pino RC4 foi selecionado para transmissão (colocou-se 0b00001001 no registo **RC4PPS**) e o pino RC7 foi selecionado para a receção (colocou-se 0b00010111 em **RX1PPS**).

Também é necessário configurar o estado da transmissão do microcontrolador colocando o valor 0b00100000 no registo **TX1STA**. Desta forma a transmissão fica configurada como sendo de 8 bits no modo assíncrono. Similarmente configurou-se a receção de comunicações colocando o valor 0b10110000 no registo **RC1STA**. Desta forma a receção de comunicações é configurada como sendo de 8 bits, sendo a sua receção bit a bit e de uma forma constante.

Configuração da BAUD rate

No correto funcionamento da aplicação, a cada pedido de leitura por parte da NCAP, esta envia 7 bytes de pedido de leitura à STIM e recebe como resposta 4 bytes (flag (1), tamanho (2) e valor do sensor (1)). Assim, atendendo que a frequência de amostragem é de $60Hz$ como deduzido anteriormente, consegue-se obter a BAUD rate mínima necessária pela Equação 1.

$$BAUD\ Rate = (4 + 7) \times 8 \times 60 = 5280bps \tag{1}$$

Definiu-se uma BAUD rate de 9600 que é o mínimo valor disponível pelo microcontrolador que garante a BAUD rate mínima requerida pela aplicação, garantido assim o correto funcionamento da aplicação, como verificado experimentalmente.

Para definir uma BAUD rate de 9600 e sabendo que a frequência de clock do microcontrolador foi definida como $32MHz$, o registo SP1BRGL foi configurado de acordo com estas especificações e usando a equação 28-1 do datasheet:

$$SPxBRG = \frac{F_{OSC}}{64 \times BAUDRATE} - 1 \tag{2}$$

Que, ao substituir os valores desejados,

$$(32000000/(64 * 9600)) - 1 = 51 = 0b110011 \tag{3}$$

Tendo isto em conta foi colocado em **SP1BRGL** 0b110011 em **SP1BRGH** todos os bits foram colocados a 0.

4.2.1.4 Configuração do ADC

O ADC gera um valor digital (binário) em função de uma tensão de referência. Esta tem de ser configurada no registo **ADREF**, onde se colocou o valor de 0b00000000. O que representa que a tensão de referência utilizada é a tensão V_{DD}, ou seja $3.3V$.

Segundo o datasheet [2], para o correto funcionamento do ADC o tempo de conversão (t_{AD}) deve estar entre $1\mu s$ e $9\mu s$. Para além disto, este tempo deve ser muito inferior ao tempo de amostragem para garantir que o tempo de amostragem definido não seja alterado pelo tempo de conversão do ADC. Atendendo isto, escolheu-se utilizar o tempo de conversão mínimo que garanta o correto funcionamento do ADC, ou seja $1\mu s$, o que se traduz numa frequência de clock do ADC de $1MHz$. Para configurar a frequência de clock do ADC em $1MHz$ através do registo **ADCLK**, nele tem de constar o valor de n dado pela seguinte fórmula:

$$\text{ADC Clock Frequency} = \frac{F_{OSC}}{2(n + 1)} \tag{4}$$

Assim,

$$n = (32Mhz/2 * 1Mhz) - 1 = 15 = 0b00001111 \tag{5}$$

Finalmente, colocou-se os resultados do ADC justificados à esquerda, em contínua operação e definiu-se que a frequência de clock do ADC seria uma divisão da do microcontrolador de acordo com o registo ADCLK, ou seja, colocou-se 0b00000000 no registo **ADCON0**.

Note que o registo **ADPCH** controla que porta será lida pelo ADC quando se iniciar uma conversão, está será comutada entre as diferentes portas associadas aos sensores ao longo da execução do programa. A tabela abaixo mostra os diferentes valores relevantes para este registo.

Valor do Registo	Porta Correspondente
001010	RB2/ANB2
001001	RB1/ANB1
001000	RB0/ANB0

4.2.1.5 Configuração do Interrupts

A aplicação projetada recorre a 2 tipos de interrupções distintos, a interrupção do ADC, quando acaba uma conversão, e a interrupção externa, quando se recebe algum valor na porta série. Primeiro colocou-se todas as flags a 0:

- No registo PIR1, o bit 0 é colocado a 0 (ADC interrupt flag);
- No registo PIR3, o bit 5 é colocado a 0 (RX1 interrupt flag).

De seguida habilitou-se as interrupções relevantes, ou seja:

- No registo PIE1, o bit 0 é colocado a 1 (Enable ADC interrupt);
- No registo PIE3, o bit 5 é colocado a 1 (Enable RX1 interrupt).

4.2.1.6 Habilitação dos mecanismos configurados

Por último, foi necessário habilitar o ADC e as interrupções globais e de periféricos, ou seja:

- No registo ADCON0, o bit 0 é colocado a 1 (Enable ADC);
- No registo INTCON, o bit 6 e 7 são colocados a 1 (Enable Peripheral and Global Interruptions).

Após explicadas as configurações do microcontrolador, prossegue-se com as funções implementadas neste para o correto e desejado funcionamento da aplicação.

4.2.2. Função *main*

Na implementação concebida a função *main* apenas serve para o propósito de configuração. Configuram-se todas as portas e rotinas importantes explicadas acima (Subsubseção 4.2.1) e definem-se variáveis como as metaTEDS, TransducerChannel TEDS e estruturas auxiliares como as queues, que serão utilizadas para o calculo das médias móveis. De seguida entra-se num ciclo while infinito que não realiza qualquer operação.

Note que se optou por enviar a média móvel dos valores e não os valores em si, uma vez que se pretendem mitigar oscilações indesejadas. O cálculo da média móvel encontra-se explicitado na Subsubseção 4.2.4.

4.2.3. Interrupções

Visto que, na ótica da aplicação projetada, o propósito da STIM é responder a comandos enviados pela NCAP, as interrupções são um processo fulcral. Estão definidas, no total, 2 interrupções (ADC e EUSART) mas apenas a do EUSART irá ser gerida pelo *interrupt_handler*, que, por sua vez, irá chamar a função *Identify_NCAP_cmd* que se passa a explicar.

A função *Identify_NCAP_cmd* irá guardar os primeiros 6 bytes (o número mínimo de bytes segundo Tabela 7) num array principal, assim como os restantes bytes dos octetos dependentes do comando num outro array auxiliar. Em seguida, esta função irá proceder à analise dos dados recebidos. Se o comando for inválido, irá proceder ao envio de uma mensagem de erro e se for válido irá enviar uma mensagem de sucesso e, em seguida, irá executar função correspondente ao comando enviado, caindo num dos possíveis casos expostos na Subseção 2.1.

As funções de envio de Meta TEDS e TransducerChannel TEDS limitam-se apenas a percorrer e a enviar os dados presentes nas estruturas que as definem. A função que interage com atuadores define o seu estado e, finalmente, a função que envia os dados da leitura de um sensor irá realizar uma média móvel sobre o mesmo, como é explicado na próxima secção, e, posteriormente, enviar o resultado da média móvel.

4.2.4. Média Móvel

Ao receber um pedido de leitura de um sensor, o microcontrolador irá iniciar uma conversão no ADC, utilizando o valor do canal desejado, esperar que esta acabe e obter o valor atual do sensor. Tal como referido, para mitigar oscilações indesejadas é realizada uma média móvel sobre este valor. Assim, este valor é colocado numa *queue* sobre a qual, posteriormente, será feita uma média móvel simples (MMS), de modo a obter uma maior estabilidade.

Com o intuito de se usar o menor número possível de valores na MMS que garantam a estabilidade dos resultados, esta é feita com uma janela de 3 valores, sendo o valor enviado calculado usando a fórmula abaixo.

$$MMS_t = \frac{x_{t-2} + x_{t-1} + x_t}{3} \tag{6}$$

Onde x_t é o valor atual, x_{t-1} o último valor e x_{t-2} o penúltimo. No caso de existirem menos de 3 valores na *queue* faz-se simplesmente a média dos valores lá presentes.

O resultado da média móvel é o resultado que é enviado pela STIM à NCAP em resposta a um pedido de leitura de um canal.

4.2.5. Fluxograma

O fluxograma do programa do microcontrolador, explicado textualmente nas secções acima, encontra-se na figura abaixo.

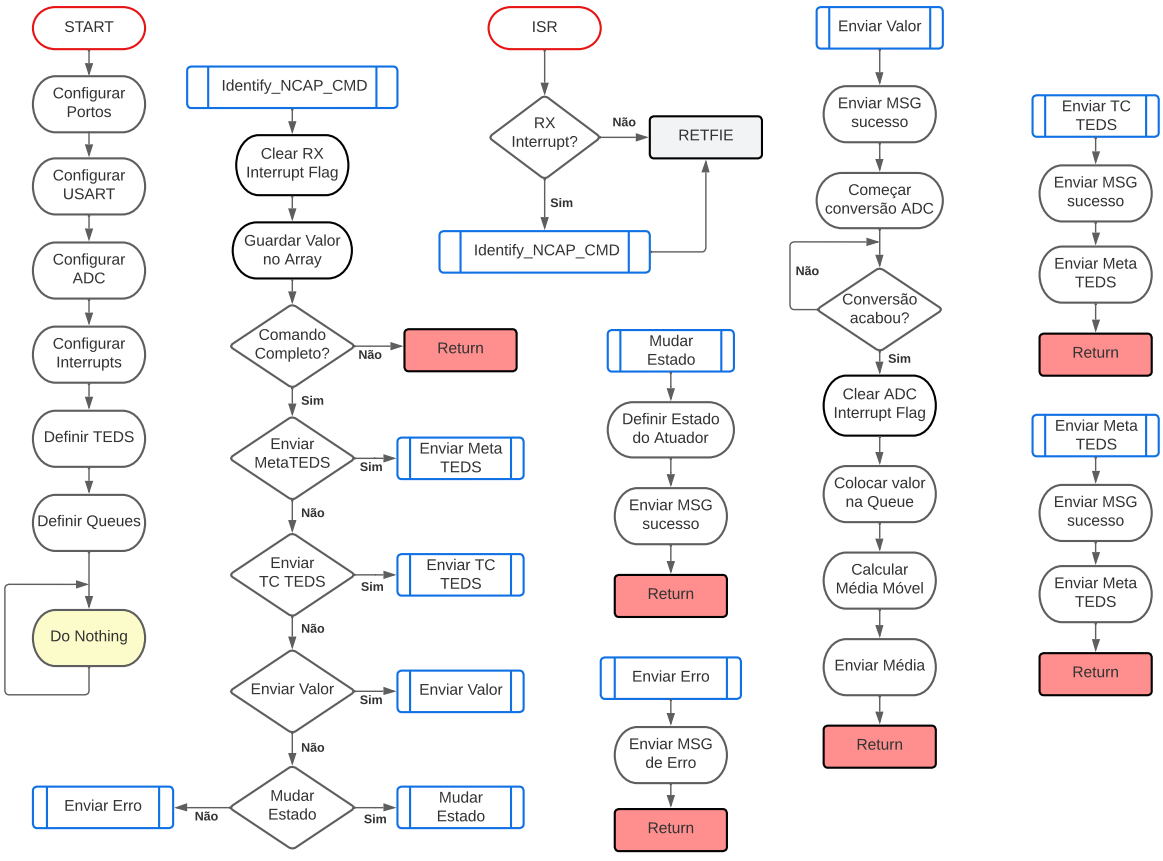

Figura 6: Fluxograma exemplificativo do funcionamento do programa a desenvolver.

5. Conclusão

Numa forma geral, o objetivo do projeto foi alcançado, sendo conseguida a implementação de um Simulador de Aviação controlado por um micro-controlador que comunica usando o protocolo

IEEE 1451. A direcção do avião foi mapeado com sucesso a partir do valor dos canais de um acelerómetro de 3 eixos.

Assim, demonstrou-se que é possível obter um vetor de orientação sobre a posição do acelerómetro e fazer com que o avião no simulador aponte nessa mesma direção, sendo esta atualizada a uma taxa de $20Hz$ devido a barreiras computacionais. Verificou-se então que o simulador poderá ser controlado por um humano, usando como interface de controlo a placa do acelerómetro.

Referências

[1] K Lee et al. "IEEE standard for a smart transducer interface for sensors and actuators common functions, communication protocols, and transducer electronic data sheet (TEDS) formats". Em: *IEEE Instrumentation and Measurement Society* (2007).

[2] *pic18f27/47q10 data sheet - microchip technology*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F27-47Q10-Data-Sheet-40002043B.pdf.